FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

Diplomová práce

Detekce anomálií stavů turbíny v datech systému RMS

Dan Široký

PLZEŇ

2023

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

Diplomová práce

Detekce anomálií stavů turbíny v datech systému RMS

Bc. Dan Široký

Vedoucí práce
Ing. Jindřich Liška, Ph.D.

PLZEŇ **2023**

Citace v seznamu literatury:
ŠIROKÝ, Dan. *Detekce anomálií stavů turbíny v datech systému RMS*. Plzeň, 2023. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra kybernetiky. Vedoucí práce Ing. Jindřich Liška, Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Dan ŠIROKÝ**
Osobní číslo:	**A21N0121P**
Studijní program:	**N3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Téma práce:	**Detekce anomálií stavů turbíny v datech systému RMS**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Seznamte se s problematikou sběru dat a nadřazených funkcí systému RMS.
2. Proveďte rešerši diagnostických přístupů pro detekci anomálií v datech.
3. Na základě dostupných informací navrhněte algoritmus pro detekci anomálních provozních stavů.
4. Navržený algoritmus otestujte nad vybranou množinou dostupných dat systému RMS.
5. Navrhněte formy vizualizačních výstupů navrženého algoritmu.

Rozsah diplomové práce: **40-50 stránek A4**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná**

Seznam doporučené literatury:

- Christopher M. Bishop. 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg.
- Liška, J., Vašíček, V., Káš, M 2019. *Real-Time Remote Monitoring of Power Plants in terms of IIoT and Cloud Computing*. The 12th International Workshop on Structural Health Monitoring. Stanford, USA.

Vedoucí diplomové práce: **Ing. Jindřich Liška, Ph.D.**
Katedra kybernetiky

Datum zadání diplomové práce: **24. října 2022**
Termín odevzdání diplomové práce: **22. května 2023**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 24. října 2022

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Plzni dne 31. července 2023

. . . . . .
Dan Široký

Abstrakt

Tato diplomová práce se zabývá úlohou vzdáleného monitorování a diagnostiky točivých strojů, konrétně průmyslových parních turbín. Jejím cílem je navrhnout softwarové řešení detekce anomálií v oblasti, kde se setkáváme s vysokými objemy dat. V průmyslových systémech je úspěšná detekce anomálií klíčovým faktorem pro zajištění jejich bezpečného a spolehlivého provozu. S rostoucím množstvím dat se však tato úloha stáva pro člověka náročnou, a proto je vhodné využít pro její řešení metody strojového učení.

V práci jsou použity dva reprezentativní přístupy k detekci anomálií: Gaussian Mixture Models (GMM) a One-Class Support Vector Machines (OCSVM). S jejich použitím byl navržen balíček implementovaný v jazyce Python, který umožňuje detekci anomálií v provozních datech turbíny.

Významnou částí práce bylo testování navržených metod na reálných operačních datech parních turbín získaných z databáze systému Remote Monitoring System (RMS). Výsledky experimentů ukázaly, že oba analyzované přístupy jsou schopny efektivně detekovat anomálie v takto komplexních datech. GMM i OCSVM dosáhly srovnatelných úrovní úspěšnosti detekce, přičemž každý z nich disponuje vlastními přednostmi a nevýhodami.

Tímto výzkumem se podařilo prokázat nejen použitelnost těchto metod, ale především navrženého řešení jako celku. Vytvořený softwarový balík je navíc možno dále rozšiřovat o nové metody detekce anomálií. Dosažené výsledky mají přínos pro průmyslovou oblast monitorování strojů a mohou být využity pro zlepšení provozní bezpečnosti a spolehlivosti parních turbín.

Abstract

This thesis deals with the task of remote monitoring and diagnostics of rotating machines, industrial steam turbines in particular. Its goal is to design a software solution for anomaly detection in an area where high volumes of data are encountered. Successful anomaly detection is a key factor in ensuring a safe and reliable operation of such industrial systems. However, with the increasing amount of data, this task becomes difficult for human operators, and the use of machine learning methods is therefore advisable for this task.

Two representative approaches to anomaly detection are used in this work: Gaussian Mixture Models (GMM) and One-Class Support Vector Machines (OCSVM). With their use, a package implemented in Python was designed, which enables ano-

maly detection in operational turbine data.

An important part of this work was the testing of the proposed methods on a real sets of operating data of steam turbines obtained from the database of the Remote Monitoring System (RMS). The result of the experiments showed that both analyzed approaches are able to effectively detect anomalies in such complex data. Both GMM and OCSVM achieved comparable levels of detection success, with each having its own advantages and disadvantages.

Through this research, it was possible to prove not only the suitability of these methods, but above all of the proposed solution as a whole. In addition, the created software package can be further expanded with additional anomaly detection methods. The achieved results bring benefits to the field of machine monitoring in industry and can be used to improve the operational safety and reliability of steam turbines.

Klíčová slova

anomaly detection • machine learning • OCSVM • GMM • signal model • remote machine monitoring • rotating machine diagnostics • signal analysis and processing

Poděkování

Za odborné vedení, cenné rady, ochotu a trpělivost, jež mi v průběhu zpracování diplomové práce věnoval, děkuji Ing. Jindřichu Liškovi, Ph.D., vedoucímu mé diplomové práce. Mé poděkování mu patří i za jeho podporu a empatii, jež byly při kompletaci této práce neméně důležité.

Obsah

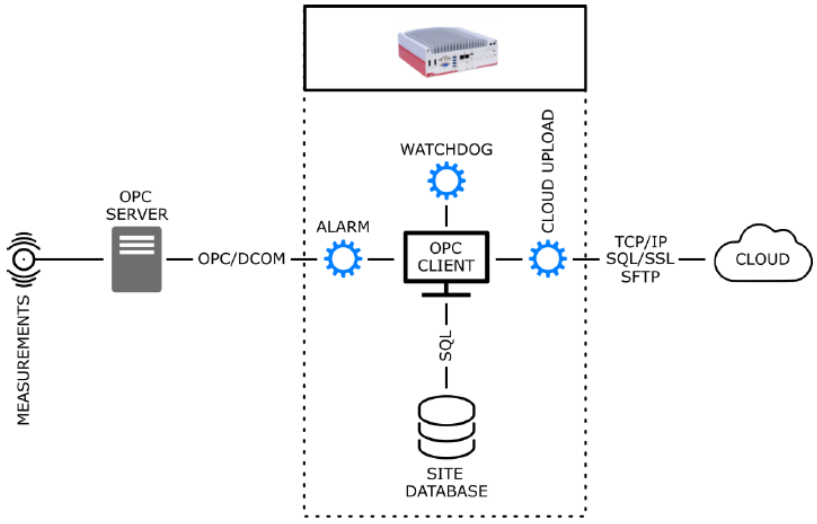
Úvod 1

Sledování stavu a struktury točivých strojů je nezbytnou součástí praxe. Monitorování těchto zařízení umožňuje na základě sledovaných parametrů detekovat, zda nedochází k nežádoucím změnám v činnosti točivého stroje. Analýza a interpretace dat získaných monitorováním daného zařízení pak umožňuje identifikovat nesrovnalosti v těchto datech nebo potenciální poruchy – obecně změny ve stavu zařízení – a rozhodovat o jeho dalším provozu. Díky tomu lze efektivněji plánovat odstávky zařízení či zabránit neočekávaným poruchám, jež by si následné odstavení zařízení vynutily. U průmyslových turbín patří k nejzajímavějším jevům z pohledu diagnostiky zadírání mezi rotorem a statorem (tzv. rubbing), strukturální pevnost závěsů lopatek, vibrace lopatek nebo detekce volných částí v průtočných částech zařízení. Neméně důležitá je ale i diagnostika senzorů použitých při monitorování provozních veličin a dalších parametrů použitých pro řízení a diagnostiku zařízení.

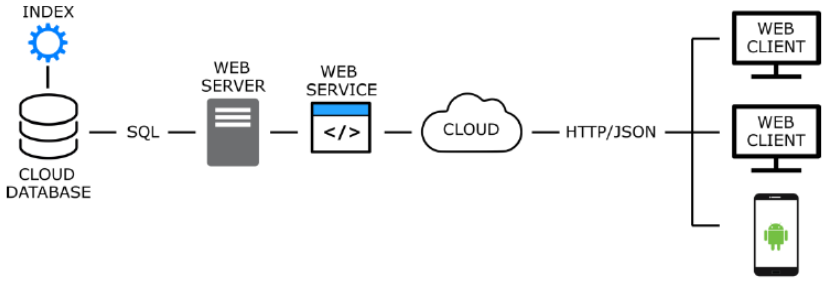
1.1 Systém RMS

Jak vyplývá z úvodu této kapitoly, detekce změn stavu průmyslových turbín je klíčovou součástí jejich bezpečného provozu, zvýšení spolehlivosti a prodloužení životnosti. Monitorování provozních veličin a měření, sledování procesů a následné vyhodnocování stavu zařízení elektráren bylo po mnoho let možné pouze přímo v areálu elektrárny. Příchod Industry 4.0 a průmyslového internetu věcí (IIoT) po roce 2010 umožnily ve všech odvětvích průmyslu, segment výroby elektrické energie nevyjímaje, vývoj nových nástrojů včetně vzdáleného monitorování provozních veličin a procesů. Laboratoř DiagEn výzkumného centra NTIS Západočeské univerzity přinesla vlastní řešení problému vzdáleného monitorování elektrárenských zařízení v reálném čase v podobě systému RMS (Remote Monitoring System) [LVK19].

Tento systém je založen na sběru dat do databázového úložiště, o něž se stará část systému instalovaná přímo v areálu elektrárny (systém měření dat). Pomocí protokolu TCP/IP je tato jednotka propojena s cloudovým serverem (systém monitorování a analýzy dat), hlavní částí systému, který obsluhuje přístup k datům a integruje do systému diagnostické komponenty [LVK19].



Obrázek 1.1: Architektura RMS a komunikačního protolu na straně zařízení. Převzato z [LVK19].



Obrázek 1.2: Architektura RMS a komunikačního protokolu na straně serveru. Převzato z [LVK19].

Měřená data i výstupy diagnostických komponent jsou přehledně zobrazovány uživatelsky přívětivým grafickým rozhraním, a to jak na stolním počítači, tak i v mobilní aplikaci. RMS v současnosti nabízí několik diagnostických modulů, například systém monitorování rubbingu či monitorování kmitání lopatek. Tento modulární design dává RMS potenciál snadné rozšiřitelnosti o libovolné další diagnostické moduly a monitorovací systémy. Škálovatelnost systému měření dat v provozu elektrárny pak nabízí možnost začlenění jakéhokoliv dalšího měřícího systému pod podmínkou existence serveru OPC [LVK19]. Systém RMS je v současnosti provozován na několika desítkách elektráren po celém světě.

1.2 Detekce stavu turbín

Systém RMS umožňuje vzdálený přístup k provozním datům elektráren a poskytuje tak obsluze nástroj umožňující monitorovat stav zařízení bez nutnosti fyzické přítomnosti v provozu. Stále je však zapotřebí aby datům věnoval pozornost expert či tým expertů, a na základě vlastních zkušeností vyhodnocovali, zda je zařízení v normálním stavu nebo se naopak v datech vyskytují anomálie. S použitím konfigurovatelného grafického rozhraní RMS zkoumá expert trendy vybraných monitorovaných veličin a grafické výstupy nasazených diagnostických modulů za určité období. Na základě znalosti zařízení a zkušeností pak identifikuje potenciální problémy, určuje jejich příčinu a predikuje budoucí chování zařízení.

Tento přístup přináší hned několik zjevných negativ. Jedná se o monotónní a v závislosti na množství dat i časově náročnou činnost. Může dojít k přehlédnutí méně očividných vzorců chování nebo ovlivnění závěrů experta jeho dosavadními zkušenostmi a očekáváními. Ruční vyhodnocování dat také neumožňuje efektivně využít historická data. Jako u všech činností takového ražení navíc i zde samozřejmě vyvstává riziko lidské chyby. Tato diplomová práce se proto zaměřuje na zjednodušení tohoto procesu návrhem automatické detekce anomálních stavů průmyslových turbín, detekovaných nad množinou předem definovaných signálů společné domény, jež by mohl být integrován do systému RMS. Expert by tak mohl soustředit svou pozornost pouze na anomální stavy detekované tímto novým modulem systému RMS.

Systém automatické detekce anomálních stavů bude postavený na technikách automatické detekce anomálních stavů[1]. Po počátečním nastavení bude využívat data poskytovaná subsystémem monitorování a analýzy dat systému RMS. Pomocí vybraných metod klasifikace budou data kontinuálně a bez nutnosti asistence obsluhy hodnocena. Na základě hodnocení mohou být data označena jako normální nebo anomální. Výsledky hodnocení budou ukládány zpět do databáze cloudového serveru, odkud mohou být následně přístupné z grafického rozhraní RMS.

[1] Rozpoznávání předmětů a jevů. V kontextu této práce je však tento termín příliš obecný a je vhodnější hovořit o detekci anomálních stavů.

Detekce anomálií 2

Jako anomálii nebo odlehlý vzorek, tzv. outlier, označujeme vzorek, jev či událost, jejíž některá vlastnost se nějakým způsobem odlišuje od normy nebo očekávání pro daný vzorek, jev či událost. Výskyt anomálií může zapříčinit mnoho důvodů od vnějšího vlivu až po selhání části sledovaného systému. Každá anomálie je však vždy analyticky významnou informací. Identifikace těchto neobvyklých odchylek neboli detekce anomálií je proto využívána v mnoha odvětvích. V bankovním sektoru například slouží k odhalování podezřelých finančních transakcí, v průmyslové automatizaci může být použita například právě k odhalování poruch zařízení.

Anomálie je možné na základě jejich projevu kategorizovat do několika skupin. Ty nejjednodušší, které se odlišují oproti očekávání a projevují se například jako extrémní hodnota v řadě měření, označujeme jako bodové anomálie. Je-li vzorek, jev či událost označena jako anomálie na základě jejího vztahu k souvisejícím datům, spadá taková anomálie do kategorie podmíněných anomálií. Takovým anomáliím se také říká kontextuální, neboť je lze chápat pouze v kontextu jejich vztahu k okolním datům. Související data nebo kontext je dán strukturou dat popisující sledované vzorky, jevy či události a musí být součástí formulace problému. Poslední skupinou jsou anomálie kolektivní. Tyto anomálie se nevyskytují osamoceně ale tvoří kontinuální řady. Mohou být proto odhaleny pouze v perspektivě jiných, normálních, řad – je nutné zkoumat je kolektivně [SC08].

Je patrné, že pro detekci anomálií není důležitá jen analýza samotné anomálie ale i dalších aspektů sledovaných vzorků. Anomálie projevující se extrémní hodnotou některé své vlastnosti může být relativně snadné identifikovat. Jsou-li však sledované vzorky, jevy či události popsány velkým množství vlastností, tedy dimenze dat je velká, může být odhalení anomálie poměrně obtížné. Žádná z vlastností anomálie nemusí mít extrémní hodnotu, avšak jejich kombinace nebude odpovídat vzoru nebo chování definovanému normálními vzorky, jevy či událostmi [SC08].

Detekce anomálií může být prováděna prostou vizuální analýzou dat nebo nasazením některého ze široké škály statistických nástrojů a nástrojů pro zpracování časových řad. Pokročilé metody detekce anomálií jsou postaveny na algoritmech strojového učení. Strojové učení je jedním z oborů umělé inteligence, zahrnující

širokou škálu technik a algoritmů. Jednou z konkrétních aplikací strojového učení je automatické detekce anomálních stavů.

2.1 Úloha detekce anomálních stavů

Tato disciplína strojového učení se zaměřuje na třídění či klasifikaci předmětů a jevů, respektive stavů, do tříd na základě jejich charakteristik. Třídou rozumíme množinu stavů vyznačujících se společnými rysy. Za účelem zkoumání charakteristik stavů se zavádí jejich formální popis vhodným výběrem kvantifikovatelných vlastností těchto stavů. Tento výběr je prováděn na základě zkušeností a expertních znalostí zkoumaných stavů a s přihlédnutím k účelu klasifikace a jejímu cíli. Každý stav je pak popsán vektorem hodnot těchto vlastností – svým obrazem. Obrazy pak tvoří příznakový prostor o rozměru, jež odpovídá počtu vybraných vlastností předmětu nebo jevu.

Proces klasifikace obrazů může být realizován mnoha různými algoritmy – klasifikátory. Tyto algoritmy pracují s modelem příznakového prostoru obrazů nebo jeho transformací. Tento model spolu s jednotlivými třídami je nutné definovat procesem nazývaným učení. Ve strojovém učení se rozlišují dva přístupy – učení s učitelem a učení bez učitele. U prvního jsou na vstup použitého klasifikátoru přiváděny obrazy spolu s informací o třídě, do níž náleží. Tato sekvence obrazů je označována jako trénovací množina. V průběhu procesu učení je klasifikátorem vytvořen popis modelu prostoru obrazů a vnitřní parametry klasifikátoru, které mu umožní obrazy správně klasifikovat. Každý klasifikátor má vlastní specifickou reprezentaci těchto naučených znalostí. V případě učení bez učitele jsou na vstup klasifikátoru přiváděny pouze obrazy a algoritmus klasifikátoru rozhodne o jejich zařazení do tříd v prostoru obrazů například na základě jejich podobnosti. Volba režimu učení je závislá na zvoleném klasifikátoru. Po ukončení učení je klasifikátor schopen přiřazovat nové obrazy do definovaných tříd na základě naučených znalostí. Proces učení a reprezentace znalostí klasifikátoru významně přispívají k přesnosti klasifikace.

2.2 Klasifikace do jedné třídy

Speciálním případem úlohy rozpoznávání je klasifikace do jedné třídy. Zásadním rozdílem této modifikace úlohy klasifikace je předpoklad dostupnosti informace pouze o obrazech jedné třídy. Obrazy jsou klasifikovány jako náležící do třídy – inlier – nebo jako outlier. Hranice mezi obrazy náležícími do této jedné cílové třídy a všemi ostatními obrazy musí být odhadnuta pouze na základě znalostí o obrazech, které do této třídy náleží. Úkolem je definovat hranici kolem cílové třídy tak, aby co nejvíce akceptovala objekty, které do této třídy mají náležet a zároveň minimalizovat

počet outlierů chybně zařazených do této třídy [Tax01]. Tento typ klasifikace je tedy velmi výhodný pro aplikace, kde oblast zájmu tvoří pouze obrazy jedné třídy nebo ostatní třídy obsahují neznámé nebo nerelevantní obrazy. Tento přístup je tedy vhodný právě pro detekci anomálií.

Pro řešení této úlohy bylo v minulosti navrženo několik metod. U všech navrhovaných metod lze pozorovat jeden ze tří hlavních přístupů [Tax01]:

- metody založené na odhadu hustoty pravděpodobnosti,
- metody založené na okrajových podmínkách,
- rekonstrukční metody.

Každý z těchto tří přístupů tvoří model svým specifickým způsobem. Společnou charakteristikou všech přístupů je ale existence dvou parametrů s principiálně stejným významem. Prvním z nich je míra podobnosti $p(z)$ či vzdálenost $d(z)$ obrazu z vztažené k cílové třídě reprezentované trénovací množinou. Míru podobnosti $p(z)$ lze chápat jako pravděpodobnost, že obraz z náleží do cílové třídy. Vzdálenost $d(z)$ může být například vzdálenost obrazu z od hranice cílové třídy. Druhým parametrem je prahová hodnota určující hranici, kterou obraz nesmí překonat má-li náležet do cílové třídy. Nové obrazy jsou klasifikovány pomocí rozhodovací funkce $I(\cdot)$ a zařazeny do cílové třídy je-li jejich míra podobnosti $p(z)$ větší než prahová hodnota θ_p nebo je-li jejich vzdálenost od cílové třídy $d(z)$ menší než prahová hodnota θ_d:

$$f(z) = I\left(p(z) > \theta_p\right), \tag{2.1}$$

$$f(z) = I\left(d(z) < \theta_d\right). \tag{2.2}$$

Konkrétní definice obou parametrů se u metod klasifikace do jedné třídy rozchází [Tax01]. Je zřejmé, že u všech metod bude nutné přistoupit nastavením prahové hodnoty na jistý kompromis mezi počtem outlierů mylně zařazených do cílové třídy a počtem obrazů nesprávně klasifikovaných jako nenáležící do cílové třídy.

2.2.1 Metody založené na odhadu hustoty pravděpodobnosti

Budeme-li předpokládat, že trénovací množina obrazů je generována nějakým rozdělením pravděpodobnosti, nabízí se jako nejjednodušší řešení úlohy klasifikace do jedné třídy odhad hustoty pravděpodobnosti tohoto rozdělení z dat trénovací množiny. Prahová hodnota, určující s jakou pravděpodobností nový obraz patří do cílové třídy, pak bude optimalizována podle vhodně zvoleného kritéria. Cílem takovýchto metod je tedy nalezení modelu vhodně popisujícího předpokládané rozdělení pravděpodobnosti trénovací množiny obrazů a optimalizace prahové hodnoty.

Problém odhadu hustoty pravděpodobnosti je však v zásadě špatně položený, neboť existuje nekonečně mnoho pravděpodobnostních rozdělení, ze kterého mohou obrazy trénovací množiny pocházet [Bis06]. Volba rozdělení dat odpovídající charakteru a struktuře dat se tak stává jedním z klíčových předpokladů při realizaci modelu [Agg17]. Velmi často používaným rozdělením v případech modelování spojitých jevů je Gaussovo rozdělení, také označované jako normální rozdělení:

$$\mathcal{N}\left(x\middle|\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \tag{2.3}$$

$$\mathcal{N}\left(\overrightarrow{x}\middle|\overrightarrow{\mu},\Sigma\right) = \frac{1}{(2\pi)^{\frac{n}{2}}}\frac{1}{|\Sigma|^{\frac{1}{2}}}e^{\left(-\frac{1}{2}\left(\overrightarrow{x}-\overrightarrow{\mu}\right)^T\Sigma^{-1}\left(\overrightarrow{x}-\overrightarrow{\mu}\right)\right)} \tag{2.4}$$

kde μ značí střední hodnotu a σ^2 varianci, respektive $\overrightarrow{\mu}$ je n-rozměrný vektor středních hodnot a Σ kovarianční matice $n \times n$ případě n-rozměrného vektoru $\overrightarrow{x}$. Mezi výhody normálního rozdělení patří jeho schopnost maximalizovat entropii, a to jak pro rozdělení jedné proměnné, tak i pro vícerozměrné rozdělení. Další silnou vlastnost Gaussova rozdělení implikuje centrální limitní věta, která říká, že za určitých podmínek má součet sady náhodných proměnných, který je také náhodnou proměnou, rozdělení, které s rostoucím počtem členů konverguje ke Gaussovskému rozdělení. V praxi může být tato konvergence poměrně rychlá [Bis06]. Normální rozdělení je tedy významným pravděpodobnostním rozdělením s širokou škálou aplikací.

Nevýhodou metod založených na odhadu hustoty pravděpodobnosti je, v případě velkého rozměru dat (obrazů), nutnost překonat takzvané prokletí rozměrnosti. V datech vysoké dimenze se od sebe body vzdalují takřka rovnoměrně, což znamená ztrátu kontrastu mezi jejich vzdálenostmi [Bis06]. Tato rovnoměrnost, či řídkost bodů v příznakovém prostoru, negativně ovlivňuje detekci anomálií, neboť výrazně ztěžuje optimalizaci prahové hodnoty. Aby k tomuto jevu nedošlo musí být trénovací množina dostatečně velká [Tax01]. Nutnost zpracovat velký objem dat však může mít negativní dopad na rychlost učení a schopnost modelu generalizovat data.

2.2.2 Metody založené na okrajových podmínkách

Odhadnout kompletní hustotu pravděpodobnosti dat může být úkol vyžadující větší úsilí než řešení samotného úkolu klasifikace do jedné třídy. Pro tuto úlohu klasifikace přitom stačí znát pouze hranici cílové třídy. Na tomto faktu staví metody založené na okrajových podmínkách (boundary methods) [Tax01]. Toto zjednodušení umožňuje použít při procesu učení menší trénovací množinu než u metod založených na odhadu hustoty pravděpodobnosti. Získání prahové hodnoty je navíc snazší je-li modelování cílové třídy zaměřeno pouze na její hranici. Klasifikace obrazů pak probíhá výpočtem jejich vzdálenosti od této hranice, respektive určením,

na které straně této hranice se v příznakovém prostoru nacházejí. Protože se tyto metody silně spoléhají na výpočet vzdáleností mezi obrazy jsou velice citlivé na škálování hodnot jednotlivých atributů obrazů [Tax01]. Je proto zřejmě výhodné data nejprve normalizovat. Je také nutné si uvědomit, že výstup klasifikátoru na rozdíl od předchozí skupiny metod nelze interpretovat jako pravděpodobnost, že obraz náleží do cílové třídy.

Populární třídou metod založených na okrajových podmínkách je Metoda podpůrných vektorů (Support Vector Machines – SVM). Algoritmus SVM hledá rozdělující nadrovinu, která od sebe v příznakovém prostoru optimálně oddělí obrazy dvou tříd. V původní formulaci SVM je cílem tohoto algoritmu nalezení optimálního lineárního klasifikátoru, který bude klasifikovat obrazy do jedné ze dvou tříd [BGV92]. Obecně lze metody SVM rozdělit do tří kategorií podle linearity modelu a separability dat. První kategorie a nejjednodušší verze SVM je již zmíněný lineární klasifikátor. Tuto původní verzi algoritmu je možné použít v případě, že data jsou lineárně separabilní. To znamená že je možné nalézt nadrovinu ležící v co nejširším pásmu necitlivosti, která od sebe oddělí data obou tříd. Druhá kategorie řeší problém lineárně neseparabilních dat. V tomto případě nejsou data plně lineárně separovatelná, například díky přítomnosti šumu. Není možné nalézt lineární oddělující nadrovinu, která by spolehlivě oddělovala data obou tříd. Cílem je nalézt takovou oddělující nadrovinu, která bude minimalizovat počet chybně zařazených obrazů. Poslední kategorii tvoří nelineární SVM. Tato varianta byla rozvinuta v 90. letech aplikací jádrové transformace [BM98], [Mül+01]. Základní myšlenkou této verze SVM je nelézt takovou transformaci příznakového prostoru, ve které budou data lineárně separabilní a bude tedy možné nalézt nadrovinu spolehlivě oddělující obě třídy. Tato nadrovina už však na rozdíl od předchozích dvou případů nebude lineární. Další modifikací SVM vyvinutou v 90. letech je verze pro klasifikaci do jedné třídy, tzv. One-Class Support Vector Machine (OCSVM) [Mül+01].

2.2.3 Rekonstrukční metody

Metody spadající do této kategorie byly primárně určeny k modelování dat. Model je vybírán s využitím apriorních znalostní o datech a s učiněním jistých předpokladů o procesu, který je vygeneroval. Většina těchto metod také činí jisté předpoklady o způsobu, jakým se obrazy stejných tříd v příznakovém prostoru shlukují nebo jakým způsobem jsou distribuovány do podprostorů příznakového prostoru [Tax01]. Model definuje jednotlivé podprostory či prototypy shluků. Pro každý obraz je definována chyba rekonstrukce. Čím je chyba rekonstrukce obrazu pro konkrétní podprostor či prototyp nižší tím spíše obraz náleží do stejné třídy. K těmto metodám patří například metody K-means, PCA nebo auto-enkodéry [Tax01].

2.3 Klasifikátor na bázi modelu Gaussovských směsí

2.3.1 Gaussovské směsi

Gaussovské směsi (Gaussian Mixture Model — GMM) jsou statisckým modelem používaným k popisu rozdělení pravděpodobnosti náhodného výběru dat. Jedná se o lineární kombinaci několika Gaussovských (normálních) rozdělení. Jednotlivá normální rozdělení se nazývají komponentou směsi [DH73], [MB88]. Každá z komponent představuje podskupinu dat s podobnými vlastnostmi a je charakterizována svým středem (střední hodnotou) a měřítkem variability (rozptylem), které jsou odlišné od ostatních složek směsi. Celková pravděpodobnostní distribuce dat je definována jako superpozice distribucí komponent:

$$p(x) = \frac{1}{N} \sum_{j} \alpha_j \cdot \mathcal{N}\left(x;\, \mu_j,\, \Sigma_j\right) \tag{2.5}$$

kde pro N komponent je α_j koeficient směšování j-té komponenty a $\mathcal{N}\left(x;\, \mu_j,\, \Sigma_j\right)$ j-tá komponenta směsi se střední hodnotou μ_j a kovarianční maticí Σ_j. Při vhodném nastavení parametrů směsi je možné modelovat širokou škálu různých rozdělení dat [Bis06].

Pro hledání optimálních parametrů komponent GMM se často používá iterativní algoritmus Expectation-Maximization (EM algoritmus) [Bis06]. Tento algoritmus používá trénovací data k nalezení hodnot parametrů komponent směsi tak, aby pravděpodobnost, že trénovací množina byla generována takovou směsí, byla maximální. Tento proces trénování je velmi efektivní a rychlý, je ovšem nutné si uvědomit, že nezaručuje nalezení globálního ale pouze lokálního maxima [Bis06]. Může být proto žádoucí opakovat trénovací proces s různými počátečními podmínkami a vybrat nejlepší výsledek.

K nevýhodám GMM patří vysoké nároky na výpočetní výkon, citlivost na volbu počtu komponent a riziko přeučení (overfitting). Úlohu volby počtu komponent lze řešit s pomocí Akaikova informačního kritéria (Akaike Information Criterion – AIC), Bayesova informačního kritéria (Bayesian Information Criterion – BIC) či heuristicky [Bis06]. Overfitting je jev, kdy se model příliš přizpůsobí trénovací množině, stává se příliš složitým a nedokáže data dobře generalizovat [Tax01]. Fenomén přeučení je úzce spojen s prokletím rozměrnosti – s rostoucí dimenzí příznakového prostoru narůstá exponenciálně množství dat, která je nutno generalizovat [DH73]. To plyne z již zmíněného řešení tohoto problému použitím velké trénovací množiny obrazů. Dobrých výsledků lze také dosáhnout natrénováním několika GMM na různých trénovacích množinách obrazů a s jiným počtem komponent a výsledný model pak sestavit jako průměr těchto GMM [Tax01], [Vap98].

Mezi výhody GMM patří schopnost identifikovat anomálie v mnoha typech datových struktur, včetně multidimenzionálních dat, kategoriálních dat a nesymetrických datových rozdělení. GMM také poskytují možnost modelovat složité struktury dat a odhalovat skryté vztahy mezi datovými body a jsou také robustní vůči šumu v datech [Bis06]. V neposlední řadě GMM nabízejí možnost relativně dobré interpretace v podobě vizualizace jednotlivých komponent.

Gaussovské směsi nachází široké uplatnění v oblasti analýzy dat. Jednou z nich je rozdělování datových bodů do několika tříd (úloha klasifikace) či seskupování podobných objektů na základě jejich popisu (clustering). Aplikace zajímavá z pohledu této práce je detekce anomálií. V této úloze se GMM používají k určení pravděpodobnosti, že daný bod datasetu je normálním nebo anomálním projevem systému [Tax01].

2.3.2 Gaussovské směsi v detekci anomálií

V této úloze se postupuje tak, že se nejprve předpokládá, že data v datasetu jsou vybrána z několika Gaussovských rozdělení. Poté se určí, jak pravděpodobné je, že každý datový bod pochází z každého z těchto rozdělení. Pokud je pravděpodobnost (likelihood), že daný bod pochází z jakéhokoliv z rozdělení velmi nízká, je označen jako anomální [Tax01]. Místo pravděpodobnosti lze s výhodou použít logaritmus věrohodnosti (log-likelihood). V jeho výpočtu se nevyskytuje součin ale suma, což snižuje výpočetní náročnost metody [Agg17]. Alternativně je možné s využitím Mahalanobisovy míry určovat vzdálenost obrazu od jednotlivých komponent směsi.

2.3.3 Parametrizace a učení

Počet volných parametrů n_p Gaussovské směsi popsané pomocí (2.5) je možné určit pomocí vztahu

$$n_p = \left(n + \frac{1}{2}n(n+1) + 1\right) \cdot N \tag{2.6}$$

kde n je počet parametrů pro střední hodnotu $\overrightarrow{\mu_j}$, $\frac{1}{2}n(n+1)+1$ je počet parametrů pro kovarianční matici Σ_j a jeden parametr v podobě koeficientu směšování α_j pro každou z N komponent směsi.Pro zjednodušení úlohy se často uvažují pouze diagonální kovarianční matice $\Sigma_j = diag(\sigma_j)$ čím se počet volných parametrů výrazně sníží na

$$n_p = (2n+1) \cdot N \tag{2.7}$$

Pokud je počet komponent směsi N předem definován zbývající parametry je možné efektivně odhadnout za pomoci zmiňovaného algoritmu Expectation-Maximization (EM-algoritmus) [Tax01].

EM-algoritmus je elegantní a výkonná metoda odhadu parametrů rozšiřující možnosti metody maximální věrohodnosti. Tento algoritmus má velice široké uplatnění a lze použít v kontextu mnoha metod strojového učení. EM-algoritmus pracuje iterativně a skládá se ze dvou hlavních kroků: očekávání (Expectation) a maximalizace (Maximization). První krok (E-step) vyhodnocuje za použití aktuálních hodnot odhadovaných parametrů aposteriorní pravděpodobnost, ve druhém kroku je tato pravděpodobnost použita k vytvoření nového odhadu hodnot parametrů. Je dokázáno, že hodnota logaritmu věrohodnosti se zvýší s každou iterací. Algoritmus končí, pokud rozdíl dvou po sobě jdoucích hodnot logaritmu věrohodnosti klesne pod předem stanovenou hranici nebo pokud změna odhadovaných parametrů nedosáhne stanovené minimální hranice [Bis06]. Jak již bylo zmíněno EM-algoritmus se používá i pro odhad parametrů hustotních funkcí, například u GMM.

Aplikaci EM-algoritmu na hledání parametrů GMM lze rozepsat do jednotlivých iteračních kroků následovně:

1. Inicializace. V tomto kroku jsou zvoleny výchozí hodnoty všech hledaných parametrů, tedy střední hodnoty μ_j, kovarianční matice Σ_j a koeficienty směšování α_j.

2. E-step. Aktuální hodnoty parametrů jsou použity pro vyhodnocení aposteriorní pravděpodobnosti

$$\gamma\left(z_{nj}\right) = \frac{\alpha_j \cdot \mathcal{N}\left(x_n \middle| \mu_j, \Sigma_j\right)}{\sum_{k=1}^{K} \alpha_k \cdot \mathcal{N}\left(x_n | \mu_k, \Sigma_k\right)} \tag{2.8}$$

3. M-step. Nový odhad parametrů s použitím aktuální hodnoty aposteriorní pravděpodobnosti

$$\mu_j^* = \frac{1}{N_j} \sum_{n=1}^{N} \gamma\left(z_{nj}\right) x_n \tag{2.9}$$

$$\Sigma_j^* = \frac{1}{N_j} \sum_{n=1}^{N} \gamma\left(z_{nj}\right) \left(x_n - \mu_j^*\right) \left(x_n - \mu_j^*\right)^T \tag{2.10}$$

$$\alpha_j^* = \frac{N_j}{N} \tag{2.11}$$

$$N_j = \sum_{n=1}^{N} \gamma\left(z_{nj}\right) \tag{2.12}$$

4. Vyhodnocení logaritmické věrohodnostní funkce

$$\ln p\left(X | \mu, \Sigma, \alpha\right) = \sum_{n=1}^{N} \ln \left[\sum_{k=1}^{K} \alpha_k \cdot \mathcal{N}\left(x_n | \mu_k, \Sigma_k\right) \right] \tag{2.13}$$

a kontrola konvergence. Není-li splněno kritérium konvergence algoritmus pokračuje bodem 2.

K volbě počátečních hodnot parametrů může být použit algoritmus K-means. Střední hodnoty budou v tom případě inicializovány jako středy shluků nalezených K-means algoritmem a kovarianční matice budou inicializovány jako kovariance nalezených shluků. Směšovací parametry mohou být inicializovány jako poměry obrazů trénovací množiny zařazených do příslušného shluku k počtu všech obrazů trénovací množiny [Bis06].

2.4 Klasifikátor s podpůrnými vektory

2.4.1 Metoda podpůrných vektorů

Algoritmus metody podpůrných vektorů (Support Vector Machine — SVM) byl poprvé představen Vladimírem Vapnikem a jeho týmem v roce 1992. Metoda staví na Vapnik-Červoněnkisově teorii (VC theory), jež se snaží o vysvětlení učícího procesu z pohledu statistiky [VC74]. Základní myšlenka SVM spočívá v hledání optimální rozdělující nadroviny, která bude rozdělovat obrazy do dvou tříd. Tuto lineární nadrovinu D popisuje

$$D(x) = w \cdot x + b \tag{2.14}$$

kde w a x jsou n-rozměrné vektory parametrů nadroviny a obrazů a b) je bias [BM98]. Rozdělující nadrovina by zároveň měla být v co největší vzdálenosti od datových bodů ležících na okraji svých jednotlivých tříd [BGV92]. Je-li separace trénovací množiny s pásmem necitlivosti M mezi třídami A a B možná, potom všechny trénovací obrazy x_k splňují následující nerovnost:

$$\frac{y_k D\left(x_k\right)}{\|w\|} \geq M \tag{2.15}$$

$$y_k = \begin{cases} +1 & x_k \in A \\ -1 & x_k \in B \end{cases}$$

Cílem metody je nalezení takových parametrů w, které maximalizují

$$M^* = \max_{w, \|w\|=1} M \tag{2.16}$$

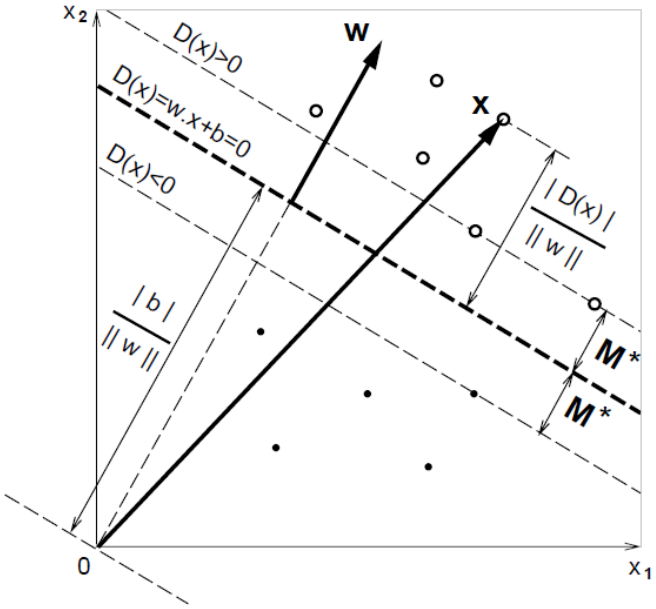
za podmínky

$$y_k D\left(x_k\right) = M; k = 1..p. \tag{2.17}$$

Maximální pásmo necitlivosti je získáno takovými obrazy trénovací množiny, které budou splňovat

$$\min_k y_k D(x_k) = M^*; k = 1..p. \tag{2.18}$$

Tyto obrazy trénovací množiny jsou nazývány podpůrné vektory (support vectors) [BM98]. Výsledkem tohoto procesu je optimální lineární klasifikátor. Současná podoba tohoto algoritmu používaná v softwarových knihovnách byla představena v roce 1993 Cortesovou a Vapnikem.



Obrázek 2.1: Ilustrace dělící nadroviny SVM. Převzato z [BGV92].

Jak již bylo řečeno v 2.2.2, SVM lze rozšířit na nelineární klasifikaci pomocí takzvané jádrové transformace (kernel trick). Tato technika transformuje data z původního prostoru na prostor, ve kterém jsou data lépe separovatelná. Tento nový prostor je obvykle vyšší dimenze než prostor původní [BM98]. Existuje množství různých jader (kernelů), jejichž použití se odvíjí od charakteristik zkoumaného datového souboru. K univerzálním a nejčastěji používaným jádrům se řadí Gaussian RBF (Radial Basis Function) kernel (2.19) [SC08], mezi další často používaná jádra se řadí polynomiální kernel (2.20) a sigmoidální kernel (2.21) [Mül+01]:

$$k(x,y) = e^{\frac{-\|x-y\|^2}{2\sigma^2}}, \tag{2.19}$$

$$k(x,y) = ((x \cdot y) + \theta)^d, \tag{2.20}$$

$$k(x,y) = \tanh(\varkappa(x \cdot y) + \theta). \tag{2.21}$$

Použití jádrové transformace se projeví v rovnici popisující rozdělující nadrovinu:

$$D(x) = w \cdot \Phi(x) + b \tag{2.22}$$

kde $\Phi(x)$ představuje takzvanou feature map sloužící k transformaci dat do prostoru vyššího rozměru, kterého pak bude dosahovat i hledaná rozdělující nadrovina [Agg17].

Problém nalezení dělící nadroviny s optimálním pásmem necitlivosti (2.18) směřuje na minimax problém

$$\max_{w,\|w\|=1} \min_{k} y_k D(x_k). \tag{2.23}$$

Tento problém má nekonečně mnoho řešení, jež se od sebe liší pouze ve škále. Zafixování normy vektoru parametrů $\|w\|$ umožní vybrat jedno konkrétní řešení. Pokud bude fixován součin této normy s velikostí pásma necitlivosti M

$$M \cdot \|w\| = 1 \tag{2.24}$$

bude maximalizace M odpovídat minimalizaci $\|w\|$. Problém hledání dělicí nadroviny s maximálním pásmem necitlivosti (2.16) je pak redukován na řešení kvadratické úlohy

$$\min_{w} \|w\|^2 \tag{2.25}$$

za podmínky (2.17). Maximální šířka pásma necitlivosti pak bude $M^* = \frac{1}{\|w\|^2}$. Toto řešení však může být nepraktické pro prostory vysoké dimenze a neposkytuje žádné informace o podpůrných vektorech. Postup učení je proto výhodné převézt na duální problém pomocí Lagrangiánu

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{k=1}^{p} \alpha_k \cdot [y_k D(x_k) - 1] \tag{2.26}$$

za podmínky $\alpha_k \geq 0$ pro $k = 1..p$, kde α_k jsou Lagrangeovy multiplikátory splňující podmínku $\alpha_k (y_k D(x_k) - 1) = 0$ pro $k = 1..p$. Optimalizačnímu problému (2.25) pak odpovídá hledání sedlového bodu $L(w, b, \alpha)$ jako jeho minima s ohledem na w a maxima s ohledem na α [BGV92].

2.4.2 One-class SVM

Ačkoliv použití SVM pro detekci odlehlých bodů (outlierů) bylo nastíněno už v původním návrhu algoritmu nebyla tato úloha dále rozvíjena [Sch+99]. Je-li SVM trénován v režimu učení bez učitele (unsupervised training) na datové sadě obsahující pouze „normální data“, tedy zástupce pouze jedné třídy, hovoříme o One-Class SVM

(OCSVM) [BS99]. Aby bylo možné tuto úlohu řešit, předpokládá se, že počátek souřadnicového systému prostoru transformovaného jádrovou funkcí (feature space) nepatří do cílové třídy [Agg17]. Cílem je vytvořit model, který dokáže rozlišit, zda každá nová data jsou generována stejným rozdělením jako data z trénovací množiny, nebo zda mohou být považována za anomální. Stejně jako základní algoritmus SVM in OCSVM může využívat jádrovou transformaci [Sch+99].

Při použití jádrové transformace jsou obrazy x transformovány funkcí feature map $\Phi(x)$ do feature space s dimenzí vyšší než původní příznakový prostor, a respektování předpokladu o počátku souřadnicového systému feature space, je dělicí nadrovina popsána následovně [Agg17]:

$$0 = w\phi(x) - b. \tag{2.27}$$

2.4.3 Parametrizace a učení

Řešení Lagrangiánu (2.26), platného i pro OCSVM, lze zapsat jako

$$\min_{w,\ \alpha,b} \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{i=1}^{N} \max\left[b - w\phi(x), 0\right] - b \tag{2.28}$$

kde $C > 1$ je váha inlierů v porovnání s outliery a řídí kompromis mezi falešně negativními a falešně pozitivními výsledky [Agg17]. Protože hodnota parametru C je shora neomezená je výhodnější ji nahradit apriorní pravděpodobností $\nu = \frac{1}{C} \in (0, 1\rangle$, která může být interpretována jako pravděpodobnost, s jakou je obraz trénovací množiny outlierem [Agg17], [BM98]. Řešení Lagrangiánu (2.26) lze s využitím ν přepsat do tvaru

$$\min_{w,\ \alpha,b} \frac{1}{2} \|w\|^2 + \frac{1}{\nu N} \sum_{i=1}^{N} \max\left[b - w\phi(x), 0\right] - b. \tag{2.29}$$

Dosavadní postup byl založen na předpokladu separovatelných dat. Tento předpoklad však nemusí být vždy splněn. Pro tyto případy je používána technika umožňující změkčit tvrdá omezení kladená na dělicí nadrovinu zavedením penalizačních proměnných ξ_i (slack-variables) pro které platí [Agg17], [BM98]:

$$\xi_i \geq b - w \cdot \Phi(x). \tag{2.30}$$

S použitím (2.30) lze řešení Lagrangiánu (2.26) ještě dále upravit do tvaru

$$\min_{w,\ \alpha,b} \frac{1}{2} \|w\|^2 + \frac{1}{\nu N} \sum_{i=1}^{N} \xi_i - b. \tag{2.31}$$

za podmínky $w \cdot \Phi(x) \geq b - \xi_i, \xi_i \geq 0$ [BM98], [Sch+99].

Duální problém, na který vede řešení (2.31), je popsán následovně:

$$\min_{\alpha} \frac{1}{2} \sum_{ij} \alpha_i \cdot \alpha_j \cdot k\left(x_i, x_j\right), \tag{2.32}$$

$$0 \leq \alpha_i \leq \frac{1}{\nu N},$$

$$\sum_i \alpha_i = 1,$$

kde $k(x, y)$ je zvolená jádrová funkce. Obrazy s koeficienty $\alpha_i \neq 0$ jsou podpůrné vektory. Rozhodovací funkce klasifikátoru je popsána funkcí [Sch+99]

$$f(x) = sgn\left(\sum_i \alpha_i \cdot k(x_i, x) - b\right). \tag{2.33}$$

Parametr b je možné získat z (2.27) jako $b = w \cdot \Phi(x_i)$ kde x_i jsou podpůrné vektory, pro jejichž koeficienty platí $0 < \alpha_i < \frac{1}{\nu N}$ [Sch+99]. Algoritmus OCSVM má tedy pouze jeden volný parametr $\nu \in (0, 1\rangle$, některé softwarové implementace tohoto algoritmu však volbu parametru b umožňují.

2.5 Hodnocení anomality

V některých případech může být žádoucí přiřazovat detekovaným anomáliím jisté hodnocení vyjadřující míru anomality. Míra anomality říká, jak moc se daná anomálie odlišuje od obrazů považovaných za normální. Volba metody pro výpočet této míry bude zřejmě záviset na použitém algoritmu detekce anomálií. Výstup některých algoritmů lze přímo použít k ohodnocení anomality zkoumaného obrazu. Příkladem může být algoritmus Isolation Forest – čím kratší je cesta grafem nutná k izolování obrazu tím větší je míra jeho anomality. U jiných metod je nutné tuto míru vypočítat, například jako odchylku od průměru nebo mediánu normálních obrazů nebo pomocí Mahalanobisovy vzdálenosti.

2.5.1 Mahalanobisova míra

U přístupu využívajícího GMM je model složen z normálních rozdělení, v nichž jsou obrazy distribuovány kolem středu tohoto rozdělení vyjádřeného střední hodnotou $\overrightarrow{\mu_j}$. Každé z těchto rozdělení pravděpodobnosti je definováno podle (2.4). Člen v exponentu této rovnice vyjadřuje polovinu druhé mocniny vzdálenosti obrazu $\overrightarrow{x}$ od středu $\overrightarrow{\mu_j}$ tohoto rozdělení [Agg17]:

$$\frac{1}{2}\left(\overrightarrow{x} - \overrightarrow{\mu}\right)^T \Sigma^{-1} \left(\overrightarrow{x} - \overrightarrow{\mu}\right) \tag{2.34}$$

neboli polovinu mocniny Mahalanobisovy vzdálenosti

$$M(\vec{x}, \vec{\mu}, \Sigma) = \sqrt{\left(\vec{x} - \vec{\mu}\right)^T \Sigma^{-1} \left(\vec{x} - \vec{\mu}\right)}. \tag{2.35}$$

Mahalanobisova vzdálenost je obdobou Eukleidovské vzdálenosti bodu k těžišti dat, je však normalizována na základě korelací mezi jednotlivými složkami dat [Agg17]. Díky tomu je zohledněna různá variance různých složek obrazů. Samotná Mahalanobisova vzdálenost ale nevypovídá o míře odlišnosti anomálie. V porovnání se vzdálenostmi normálních obrazů však bude větší, neboť se anomální obraz nachází za hranicí Gaussovské směsi.

Mahalanobisova míra se na první pohled může zdát velice jednoduchá, avšak její schopnost zohlednit závislosti mezi jednotlivými složkami je velmi důležitá, zejména s rostoucí dimenzí obrazů. Stran výpočetní náročnosti je Mahalanobisova vzdálenost se složitostí $O(N)$ pro N obrazů úspornější než jiné metody, které obvykle dosahují složitosti $O\left(N^2\right)$ [Agg17]. Další silnou výhodou je absence jakékoliv parametrizace. Tato vlastnost je žádoucí zejména v úlohách jako je detekce anomálií, kde není k dispozici referenční hodnota příslušnosti obrazu k cílové třídě, ground-truth, která by umožnila optimalizaci jakéhokoliv parametru [Agg17].

2.5.2 Vzdálenost od rozdělující nadroviny

Jak již bylo zmíněno, metody SVM modelují jednotlivé třídy v příznakovém prostoru pomocí rozdělujících nadrovin popsaných podpůrnými vektory. Rozhodovací funkce je pak založena na výpočtu vzdálenosti obrazu od této nadroviny. Je-li vzdálenost od nadroviny kladná patří obraz do třídy jíž tato nadrovina vymezuje, v opačném případě patří do jiné třídy. V úloze detekce anomálií se však vyskytuje pouze jedna třída. Hodnotu vzdálenosti obrazu, jež je vyhodnocen jako anomální, lze tedy interpretovat jako míru jeho anomality – čím je od rozdělující nadroviny dále tím více se odlišuje od normálních obrazů. Protože je výpočet vzdálenosti součástí samotného procesu klasifikace není pro zjištění míry odlišnosti anomálie nutné investovat další zdroje.

Návrh modulu pro systém RMS

3

Design modulu byl navržen pomocí jazyka UML. Samotná implementace proběhla v jazyce Python za použití vývojového prostředí PyCharm. Modul je koncipován tak, aby fungoval jak samostatně s daty poskytnutými ve formátu Comma Separated Values tak i jako součást systému RMS s připojením na SQL databázi systému monitorování a analýzy dat.

3.1 Návrhová kritéria

Současná distribuce systém RMS je implementována v jazyce Python verze 3.7.8. Datový sklad je realizován zabezpečenou SQL databází realizovanou pomocí Microsoft SQL Sever. Nová data jsou do této databáze přijímána s relativně nízkou periodou desítek sekund až jednotek minut v závislosti na realizaci měření konkrétního signálu. Jednotlivé signály ze zájmové skupiny signálů společné domény nemusejí být měřeny a přijímány se stejnou periodou.

Navrhovaný modul detekce anomálií bude pro dosažení maximální kompatibility také implementován v jazyce Python verze 3.7.8. Modul musí být schopen komunikace s SQL databází. Signály nemohou být z databáze čteny v závislosti na čase, neboť každý signál může být měřen v jinou dobu a s jinou periodou. Díky relativně dlouhým časům mezi novými měřeními není nutné klást významné restrikce na maximální dobu potřebnou pro zpracování dat. Protože celý systém musí být schopný operovat bez dozoru je samozřejmě nutné ošetřit všechny očekávané chybové stavy, které by mohly způsobit zastavení činnosti modulu, tak, aby byl modul vždy uveden do bezpečného stavu, v němž je schopný dále operovat nebo alespoň neovlivnit činnost nadřazeného systému.

Pro zachování důvěry uživatelů ve výsledky klasifikace musí klasifikátor produkovat minimum falešně pozitivních výsledků, tedy chybných označení normálních dat jako anomální. Proces učení a nastavení prahových hodnot tedy musí proběhnout tak, aby klasifikátor identifikoval co nejméně normálních obrazů jako anomálie i za cenu klasifikace některých anomálií jako normálních obrazů. Součástí návrhu

je i zobrazení výsledků klasifikace v čase, které pak může být použito v grafickém rozhraní RMS.

3.2 SQL v jazyce Python

SQL (Structured Query Language) je jazyk navržený pro správu a manipulaci s relačními databázemi, vyvinutý v 70. letech firmou IBM. Jazyk je založen na matematickém relačním modelu dat. Umožňuje reprezentovat data v databázi ve formě tabulek s řádky a sloupci. Řádky reprezentují jednotlivá data, sloupce pak atributy dat. SQL poskytuje sadu příkazů umožňující dotazování, vytváření, úpravu a odstraňování tabulek a dat [Cod70]. Jednou z konkrétních implementací databázového systému s použitím jazyka SQL je například Microsoft SQL Server.

Jazyk Python sám o sobě neposkytuje přímou podporu komunikace s SQL servery, komunita vývojářů tohoto jazyka však již v roce 1996 definovala prostřednictvím dokumentů Python Enhancement Proposals (PEP) první standardní rozhraní pro práci s relačními databázovými systémy Python Database API 1.0. Tento standard umožnil vznik prvních knihoven umožňujících komunikaci s databázovým serverem pomocí standardních dotazů SQL. V roce 1999 byla vydána nová verze standardu Python Database API 2.0 která definovala nový objekt kurzor, který umožňuje efektivnější zpracování dat z databáze [Fou23]. V současnosti je k dispozici několik knihoven postavených na standardu Python Database API 2.0, ne všechny jsou však vhodné pro komunikaci s MS SQL Server. Z hlediska popularity a rozsahu podpory lze uvažovat pouze o knihovnách pmyssql a odbc.

3.2.1 Knihovna pymssql

Pymssql je navržena specificky pro práci s Microsoft SQL Server. Instalace je snadná a knihovna nemá žádné závislosti na kódu třetích stran. Je vyvíjena jako open-source projekt pod MIT licencí, která umožňuje neomezené použití. Rozhraní pymssql je jednoduché a umožňuje snadnou optimalizaci a rychlou komunikaci s SQL serverem. Současná distribuce podporuje Python až do verze 3.11 [Mik22]. Autoři ke knihovně poskytují dobře vypracovanou dokumentaci v dostatečném rozsahu pro běžné použití.

Nevýhodou knihovny je malá flexibilita plynoucí z jejího zaměření na Microsoft SQL Server. Knihovna je stále aktualizována, avšak frekvence aktualizací se pohybuje v řádu jednotek za rok. Nové distribuce se navíc občas potýkají s problémy, zejména distribuce pro OS Microsoft Windows.

3.2.2 Knihovna pyodbc

Tato knihovna poskytuje standardní ODBC API, implementovanou v souladu se specifikací Python Database API 2.0, umožňující komunikaci s mnoha databázovými systémy, včetně MySQL nebo Microsoft SQL Sever. Instalace je opět snadná. Distribuce pro OS MS Windows nevyžaduje instalaci dalších balíků či jiných závislostí, distribuce pro MacOS a Unix vyžaduje instalaci ovladače ODBC. Pyodbc je také vyvíjena jako open-source projekt pod MIT licencí. V současnosti jsou k dispozici verze pro Python 2.7, Python 3.7 a vyšší [Kle23]. K dispozici je rozsáhlá dokumentace včetně příkladů použití.

Četnost aktualizací pyodbc je ještě nižší než u pymssql, verze jsou však zpravidla stabilní a nutné opravy vycházejí v relativně krátkém čase. Velkou výhodou je použití standardního rozhraní ODBC, usnadňující komunikaci s databázovým systémem. Tuto knihovnu v současnosti používá i RMS, dává proto smysl zvolit pyodbc i pro modul detekce anomálií a nezvyšovat tak počet knihoven potřebných pro běh RMS.

3.3 Detekce anomálií v jazyce Python

Jazyk Python se díky své jednoduchosti, flexibilitě, rychlosti a rozsáhlé komunitě, jež se kolem něj vytvořila, stal jedním z nejpopulárnějších programovacích jazyků současnosti. S rostoucím zájmem o analýzu dat, strojové učení a detekci anomálií vzniká mezi výzkumníky a vývojáři potřeba efektivních nástrojů, které by tyto procesy umožnily automatizovat. Do bohatého ekosystému Pythonu proto přibývají knihovny, které poskytují širokou škálu specializovaných algoritmů a nástrojů zaměřených na tyto oblasti. Mezi nejvýznamnější patří knihovny Time-series Outlier Detection Systém (TODS), Python Streaming Anomaly Detection (PySAD), Python Outlier Detection (PyOD) a Scikit-learn. Všechny zmíněné knihovny jsou open-source a volně poskytované pod BSD licencí.

3.3.1 Knihovna TODS

Knihovna TODS je zaměřena na detekci anomálií v časových řadách, ale nabízí i širokou paletu nástrojů pro zpracování dat a časových řad, feature analysis a kalibrování systému na základě expertních znalostí (rozhraní human-in-the-loop). Poskytuje tak full-stack řešení pro detekci anomálií v praxi [Lai+21]. Aktuálně poskytovaná verze 0.0.2 nabízí 22 modelů včetně OCSVM. V nabídce však chybí model využívající GMM, což je v kontextu této práce nevýhodné. K dalším nevýhodám patří absence nástrojů pro vyhodnocování kvality modelů či návrh jejich parametrů. Většina modelů je navíc převzatá z knihovny PyOD a nejsou tedy nativní k této knihovně. V neposlední řadě je důležitým argumentem proti použití této knihovny

momentální stav jejího vývoje. V době tvorby této práce nejsou přidávány žádné nové funkčnosti a neprobíhá ani údržba kódu s výjimkou drobných oprav.

3.3.2 Knihovna PySAD

Další populární knihovna, PySAD, se specializuje na online a sekvenční zpracování dat a detekci anomálií. Výhodou tohoto přístupu je nízká náročnost na zdroje výpočetního systému [YK20]. Paleta nabízených nástrojů odpovídá rozsahem a možnostmi knihovně TODS. Navíc PySAD nabízí modul obsahující metody pro statistické zpracování dat a několik tříd pro vyhodnocení a kombinování výstupu modelů. V aktuální verzi 0.1.1 je v nabídce 16 nativních modelů a modul pro integraci modelů knihovny PyOD. V základním repertoáru modelů však chybí OCSVM i GMM, což činí knihovnu nevhodnou pro potřeby této práce. Stejně jako TODS i PySAD postrádá nástroje pro vyhodnocení kvality modelů a návrh jejich parametrů. Velkou nevýhodou je také absence dalšího vývoje – poslední aktualizace byla zveřejněna v roce 2020.

3.3.3 Knihovna PyOD

Nejobsáhlejší ze zatím zmíněných knihoven je PyOD. Knihovna se zaměřením na detekci anomálií ve vícerozměrných datech nabízí ve verzi 1.0.9 více než 40 modelů [ZNL19]. Až na výjimky jsou všechny modely převzaty z knihovny scikit. Přínosem PyOD není implementace samotných detekčních algoritmů ale poskytnutí unifikovaného rozhraní, wrapperu, který umožňuje realizovat detekci anomálií velice snadno bez ohledu na zvolený model. Slabinou PyOD je malá nabídka nástrojů pro práci s daty a výstupy modelů. Stejně jako předchozí knihovny ani PyOD nenabízí možnosti hodnocení modelů a nástroje pro návrh jejich parametrů. Knihovna je i v současnosti dále vyvíjena, její autoři nabízejí rozsáhlou dokumentaci, návody a příklady použití.

3.3.4 Knihovna sklearn

Scikit-learn není zaměřená přímo na detekci anomálií ale na strojové učení jako takové [Ped+11]. Ve verzi 1.2.2 je na výběr více než 50 modelů pro učení s učitelem i bez učitele. Navíc knihovna nabízí bohatou paletu nástrojů pro zpracování vstupních i výstupních dat, výběr a hodnocení modelu, diagnostiku modelů a vizualizaci dat. K dispozici je rozsáhlá dokumentace podrobně popisující všechny třídy. Z pohledu detekce anomálií v kontextu této práce prezentuje použití knihovny scikit-learn oproti ostatním dvě nevýhody. Za prvé, je nutné vybudovat nadstavbu, která na základě výstupu modelu binárně vyhodnotí, zda byl vstup anomální nebo ne. Za druhé, zaměnitelnost modelů v konkrétní aplikaci není možná bez implementace

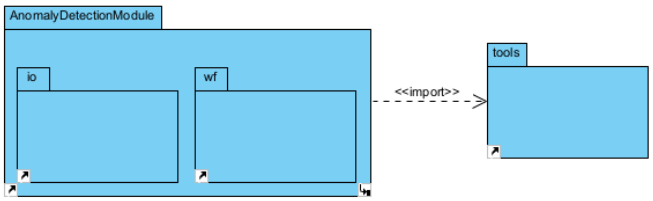
unifikovaného rozhraní. Nabídka funkcí pro návrh a diagnostiku modelů, jež u jiných rozšířených knihoven chybí, dělá z této knihovny i přes zmíněné nevýhody nejlepšího kandidáta. Scikit-learn je navíc stále aktivně vyvíjen nejen týmem autorů ale i rozsáhlou komunitou.

3.4 Architektura modulu pro systém RMS

V této kapitole je prezentována struktura a architektura navrženého modulu detekce anomálií pro systém RMS. V jednotlivých sekcích jsou popsány jednotlivé komponenty, vztahy mezi nimi a návrhové vzory použití pro jejich implementaci.

3.4.1 Struktura modulu

Modul detekce anomálií byl navržen s ohledem na přehlednost, modularitu, rozšiřitelnost a snadnou konfigurovatelnost. Odpovídá tomu hierarchická struktura balíku. Každá část zastává v modulu určitou roli a implementuje specifickou funkčnost.

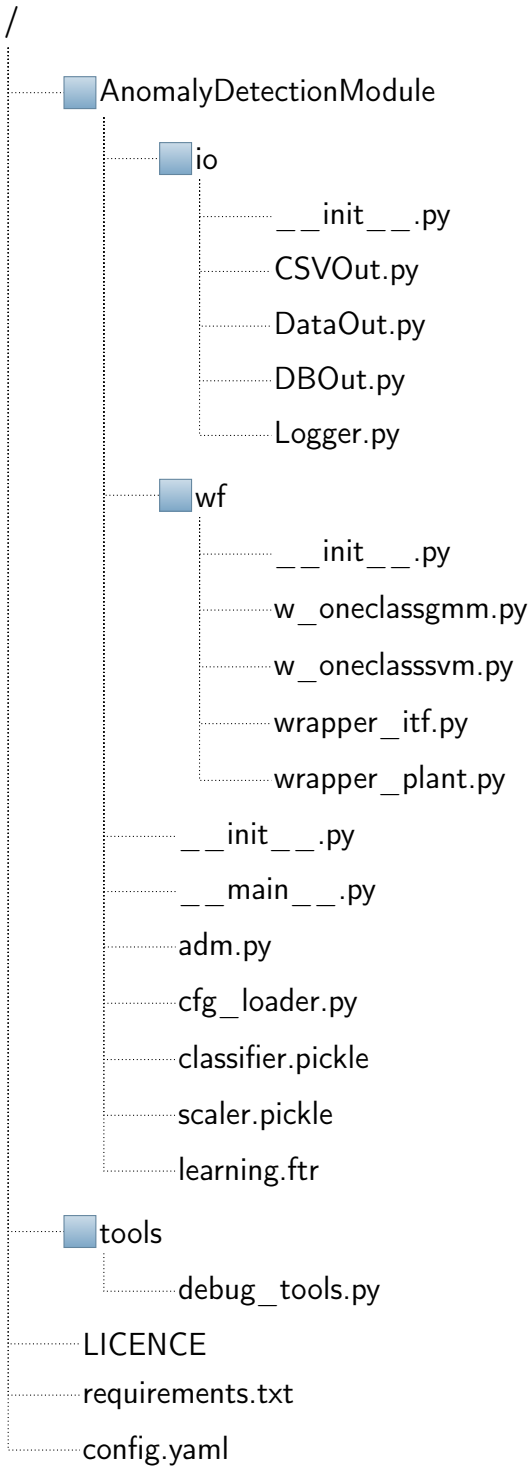


Obrázek 3.1: UML package diagram modulu detekce anomálií

Hlavní část modulu, balík AnomalyDetectionModule, obsahuje jádro modulu v podobě stejnojmenné třídy a dva podružné balíky nazvané io a wf. Podružný balík io obsahuje třídy pro komunikaci a zpracování dat, podružný balík wf pak obalové třídy klasifikátorů. Vedle balíku AnomalyDetectionModule obsahuje modul ještě balík tools, který obsahuje různé funkce použité pro vývoj a ladění modulu. Jeho obsah není z funkčního hlediska podstatný a nebude proto blíže popisován. Hierarchické struktuře modulu odpovídá i hierarchická struktura projektu, znázorněná na Obr. 3.2 pro ilustraci a k pozdějšímu nahlédnutí.

Součástí projektu je konfigurační soubor, obsahující veškeré informace nutné k bezchybnému běhu modulu. K těmto informacím patří například IP adresa SQL serveru RMS. Dále je pomocí něho možné vynutit učení klasifikátoru, přepínat mezi vývojovým a provozním módem a zapínat režim ladění.

Je-li kdekoliv v rámci balíku pracováno se soubory je k těmto operacím použita standardně používaná knihovna pickle. Výjimku tvoří pouze operace s konfiguračním souborem a textová výstupní třída v podružném balíku io.



Obrázek 3.2: Souborová struktura projektu

3.4.2 Balík AnomalyDetectionModule

Tato část modulu zahrnuje veškeré jeho funkční komponenty. Krom zmíněných podružných balíků obsahuje jádro celého projektu. To sestává jednak z magických[1] metod jazyka Python, konkrétně dunder[2] init a dunder main, dále pak ze skriptu `cfg_loader.py` pro zpracování konfiguračního souboru `config.yaml` a nakonec hlavní třídy celého balíku AnomalyDetectionModule v souboru `adm.py`. Metoda dunder init slouží k inicializaci celého balíku, metoda dunder main pak jako vstupní bod pro jeho samostatné spouštění mimo systém RMS.

Třída AnomalyDetectionModule.

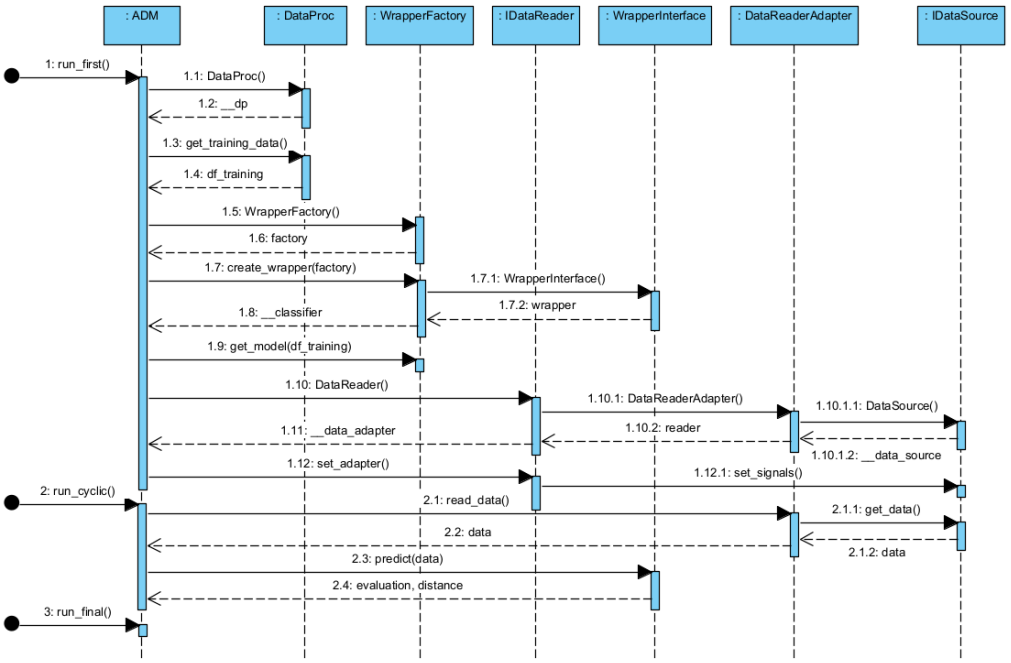
Tato třída, implementovaná v souboru `adm.py`, obsahuje tři metody, které zajišťují běh celého programu. První metoda *run_first()* inicializuje adaptér pro čtení dat a klasifikátor pro detekci anomálií nad daty. Instance obou těchto tříd, adaptéru i klasifikátoru, jsou atributy třídy *AnomalyDetectionModule*. Klasifikátor je dodán podružným balíkem wf, jeho výběr a trénování probíhají na základě nastavení v `config.yaml`. Tento proces bude detailně popsán v kapitole zabývající se tímto podružným balíkem. Pokud vytváření klasifikátoru selže metoda končí s návratovou hodnotou 1. V opačném případě se pokračuje inicializací adaptéru. Adaptér je implementován podružným balíkem io a slouží ke čtení dat z vybraného zdroje. Detailnímu popisu adaptéru se bude věnovat jedna z pozdějších kapitol. Nepodaří-li se adaptér inicializovat metoda končí s návratovou hodnotou 1. Jsou-li všechny operace dokončeny úspěšně metoda končí s návratovou hodnotou 0. Je-li aktivní režim ladění jsou konkrétní chybová hlášení uložena do textového výstupu. Cesta k tomuto souboru musí být definována v `config.yaml`. Metoda *run_first()* musí být úspěšně zavolána právě jednou před voláním dalších metod.

Druhá metoda *run_cyclic()* je určena k cyklickému volání a provádí pomocí adaptéru a klasifikátoru samotnou detekci anomálií v datech. Její činnost lze rozdělit do 4 kroků: čtení nových dat, kontrola operace čtení, klasifikace, uložení výsledku. Čtení nových dat je realizováno voláním metody adaptéru *read_data()*. Je-li adaptérem signalizováno neúspěšné obdržení dat metoda končí s návratovou hodnotou 1. Je-li čtení úspěšné pokračuje běh voláním metody klasifikátoru *predict()*. V posledním kroku je výsledek tohoto volání uložen. Protože integrace do systému RMS nebyla zatím provedena a SQL databáze není připravena pro ukládání výstupu modulu detekce anomálií, je výsledek vypsán do konzole. Opět platí, že jakékoliv chybové hlášení je v režimu ladění uloženo do textového výstupu.

[1] Magické metody, neboli magic methods, označují v jazyce Python speciální typ metod implementujnící specifickou funkčnost jako např. inicializace třídy. Jejich jména jsou rezervovaná klíčová slova ve formátu dunder.

[2] Dunder je zkrácený výraz pro klíčová slova začínající a končící dvěmi podtržítky (double underscore). Tento výraz by mohlo být možné považovat za součást žargonu komunity vývojářů Python.

Poslední metoda *run_final*() musí být zavolána po posledním volání metody *run_cyclic()*. V současné verzi tato metoda neobsahuje žádný výkonný kód a slouží pouze pro ukončení výstupu ve vývojovém módu. Je však možné ji využít v budoucnu pro provedení jakýchkoli operací, jež by mohlo být nutné provézt pro ukončením běhu modulu. Posloupnost jednotlivých operací v popsaných metodách hlavní třídy *AnomalyDetectionModule* ilustruje následující sekvenční diagram.



Obrázek 3.3: Sekvenční diagram pro volání metod hlavní třídy

3.4.3 Balík io

Tento balík obsahuje veškeré programové nástroje pro práci s daty. Obsahuje třídu adaptéru *DataReaderAdapter*, která implementuje rozhraní *IDataReader*. Adaptér volí na základě nastavení v config.yaml příslušný zdroj dat, který má modul používat. Jednotlivé zdroje dat reprezentují třídy *CSVOut* a *DBOut*, které implementují rozhraní *IDataSource*. Vždy může být použita právě jedna z nich. Další třídou je *Logger*, který umožňuje přesměrování standardního výstupu do textového souboru. Toho je využíváno nejen ve vývojovém módu ale i v režimu ladění a normálním režimu. Poslední třídou balíku je *DataProc*, která ve vývojovém módu slouží jako zdroj trénovacích dat pro klasifikátory. Z důvodu chybějící integrace do RMS slouží prozatím i jako zdroj trénovacích dat mimo vývojový mód.

Třída DataReaderAdapter.

Třídu *DataReaderAdapter* by bylo možné označit za hlavní třídu tohoto balíku. Nachází se v souboru `DataOut.py`. Tato třída implementuje rozhraní *IDataReader*, což umožňuje jednak dobré funkční oddělení od třídy *AnomalyDetectionModule*, ale také možnost nahrazení *DataReaderAdapter*u libovolnou jinou třídou, která bude implementovat stejné rozhraní a plnit stejnou funkci. Rozhraní *IDataReader* je součástí souboru `DataOut.py`.

Za účelem konzistentního přístupu k datům byla tato třída konstruována podle třídního návrhového vzoru Adapter (class adapter pattern). Tento návrhový vzor slouží jako forma obalové třídy převádějící rozhraní jedné ze specifických tříd, takzvaných Adaptee, na rozhraní Target Interface očekávané klientskou třídou. V kontextu tohoto návrhového vzoru je *DataReaderAdapter* Adapter a *IDataReader* odpovídá Target Interface, které očekává klient *AnomalyDetectionModule*. Třídy *DBOut* a *CSVOut* figurují jako Adaptee. Pro zjednodušení implementace adaptéru a lepší rozšiřitelnost musí navíc každý Adaptee implementovat rozhraní *IDataSource*, které je také součástí souboru `DataOut.py`.

Voláním *run_first()* třídy *AnomalyDetectionModule* je vytvořena instance třídy *DataReaderAdapter*. Jako součást inicializace této instance je vytvořena instance třídy *DBOut*, respektive třídy *CSVOut* pokud je aktivní vývojový mód, jako atribut třídy *DataReaderAdapter*. Po dokončení inicializace adaptéru je ještě nutné zavolat metodu *set_adapter()*, která je součástí rozhraní *IDataReader*. Tato metoda slouží k vykonání jakýchkoliv dalších operací, které mohou být vyžadovány konkrétní implementací tříd typu Adapter nebo Adaptee, ale není možné je vykonat během jejich inicializace. Metoda *set_adapter()* nemá žádné parametry ani návratovou hodnotu.

Druhou veřejnou metodou, specifikovanou rozhraním *IDataReader*, kterou tato třída implementuje, je metoda *read_data()*. V těle této metody je voláno rozhraní třídy, která figuruje jako aktuálně použitý Adaptee. Návratové hodnoty jsou následně přeformátovány v souladu s deklarací *IDataReader* a vráceny klientské třídě. *DataReaderAdapter* ještě implementuje dvojici soukromých metod, které slouží právě k formátování návratových hodnot Adaptee.

Třída DBOut.

Tato třída, nacházející se v souboru `DBOut.py`, poskytuje rozhraní pro čtení dat z SQL databáze. IP adresa této databáze a iniciále nutné k navázání připojení jsou definovány v `config.yaml`. Aby nemohlo dojít ke konfliktnímu přístupu k databázi je třída *DBOut* konstruována podle návrhového vzoru Singleton. To znamená že je možné vytvořit pouze jednu instanci této třídy, čímž je zajištěno, že v rámci modulu nikdy nemůže existovat více než jedno připojení k databázi.

Během inicializace k níž dochází při vytvoření instance třídy *DBOut* jsou všechny informace týkající se cílové SQL databáze přečteny z config.yaml a uloženy do sou-

kromých atributů. Pokud by mělo dojít k vytvoření další instance této třídy, zajistí aplikovaný návrhový vzor, že místo vytvoření nové instance dojde k vrácení odkazu na již existující instanci. K další inicializaci proto také nedojde. Instance *DBOut* není po inicializaci ještě připravena k použití, je nejprve nutné navázat první spojení s databází a inicializovat seznam signálů společné domény.

Jak již bylo zmíněno třída *DBOut* implementuje rozhraní *IDataSource*. Tímto rozhraním jsou specifikovány dvě povinné veřejné metody. První z nich je metoda *set_signals()*, jež je volána adaptérem v metodě *set_adapter()*. Právě tato metoda slouží k inicializaci seznamu signálů společné domény. Je-li volání úspěšné metoda vrátí návratovou hodnotu 0x00 a instance třídy je připravena k použití. V opačném případě vrátí volaná metoda *set_signals()* v návratové hodnotě celočíselný chybový kód jež může volající dále zpracovávat.

Druhou metodou specifikovanou rozhraním *IDataSource* je metoda *get_data()*. Ta je implementována voláním soukromé metody *get_signal_data_last()*, jež realizuje připojení k SQL databázi, dotaz na poslední zaznamenaná měření všech signálů společné domény, ukončení připojení a vrácení obdržených hodnot. Metoda zároveň vrací celočíselný chybový kód. Pokud není připojení nebo dotaz na hodnoty úspěšné chybový kód specifikuje nastalý problém. V opačném případě je chybový kód roven 0x00. Třída dále obsahuje soukromé metody pro navázání a ukončení připojení k SQL databázi a soukromou metodu realizující SQL dotaz na obdržení seznamu signálů společné domény.

Třída CSVOut.

Třída slouží zejména ve vývojovém módu jako rozhraní pro čtení dat z exportu databáze RMS ve formátu Comma Separated Values. Název tohoto souboru a cesta k němu jsou specifikovány v `config.yaml`. I třída *CSVOut* je realizována podle návrhového vzoru Singleton. Tím je zabráněno konfliktnímu přístupu k souboru exportu databáze RMS. Kód třídy se nachází v souboru `CSVOut.py`.

V průběhu inicializace této třídy je z `config.yaml` kromě názvu a cesty k souboru CSV ještě vyčten rozsah dat s nímž má třída pracovat. Instance třídy *CSVOut* je ihned po inicializaci připravena k použití. Povinná metoda *set_signals()* specifikovaná rozhraním *IDataSource* je implementována jen jako pahýl, neboli stub, a vrací vždy hodnotu 0x00. Druhá povinná metoda *get_data*() otevře soubor CSV, vyčte požadovaný záznam a vrátí jej ve formátu Pandas DataFrame.

Třída Logger.

Logger je kompaktní třída sloužící k přesměrování standardního výstupu *stdout* do textového souboru definovaného v `config.yaml`. Z důvodu prevence konfliktního přístupu k tomuto souboru a nutnosti zachování stavu instance třídy *Logger* je realizována podle speciálního návrhového vzoru Borg, někdy též nazývaný Mo-

nostate. Tento návrhový vzor zajišťuje, že všechny instance dané třídy sdílí stejný stav – mají tedy stejné vlastnosti a shodné hodnoty atributů. Díky tomu je možné vytvářet instance třídy *Logger* libovolněkrát v libovolné třídě balíku a inicializaci provézt pouze jednou. Všechny instance budou zapisovat do stejného textového souboru, aniž by ho bylo nutné po každém zápisu zavřít a před každým zápisem znovu otevřít. Tím dochází i k drobné úspoře zdrojů, neboť pro činnost *Logger*u je potřeba méně operací. Použití třídy *Logger* je uživatelsky velmi přívětivé – každý zdrojový kód, jehož výstup je žádoucí přesměrovat, stačí zapouzdřit příkazem `with` s instancí třídy *Logger*. Implementaci třídy *Logger* lze nalézt v souboru `Logger.py`.

3.4.4 Balík wf

Podružný balík wf sdružuje všechny dostupné obalové třídy klasifikátorů a nabízí jednoduché rozhraní pro práci s nimi. Středobodem balíku je třída *WrapperFactory* jež zprostředkovává vytvoření žádané obalové třídy. Aby byl tento proces co možná nejjednodušší a robustní jsou třídy v tomto balíku koncipovány tak, aby umožňovaly použití návrhového vzoru Factory method. Tento návrhový vzor umožňuje vytvářet instance tříd, aniž by bylo nutné specifikovat jaká třída má být instanciována. Objekt vytvářející instanci tak nevolá konstruktor ale metodu jiné třídy, takzvanou factory method, která se postará o vytvoření instance. Všechny obalové třídy v tomto balíku musejí implementovat rozhraní *WrapperInterface*, jež je součástí tohoto podružného balíku. Aktuálně jsou k dispozici obalové třídy *WOneClassGMM* a *WOneClassSVM*.

Třída WrapperFactory.

Tato třída, implementovaná v souboru `wrapper_plant.py`, působí v kontextu návrhového vzoru Factory method jako Creator. To znamená, že implementuje samotnou factory method *create_wrapper()*, která se stará o tvorbu instancí další, specializovaných, tříd. Jedná se zároveň o jedinou metodu, kterou tato třída implemetuje.

Třída WOneClassGMM.

Obalová třída *WOneClassGMM* umístěná v souboru `w_oneclassgmm.py` implementuje rozhraní *WrapperInterface* ze souboru `wrapper_itf.py`. Třída slouží jako obal pro třídu Gaussian Mixture Model knihovny sklearn. K tomu používá primárně dvě povinné veřejné metody *get_model()* a *predict()* definované implementovaným rozhraním. První z těchto metod slouží k získání klasifikátoru buď jeho vytvořením a natrénováním nebo načtením již hotového klasifikátoru ze souboru. Na Obr. 3.2 je tento soubor ilustrován souborem `classifier.pickle`. Stejným způsobem je získán i objekt Standard Scaler knihovny sklearn, který slouží k normalizaci dat pro trénování i klasifikaci (ilustrován souborem `scaler.pickle`). Metoda *predict()*

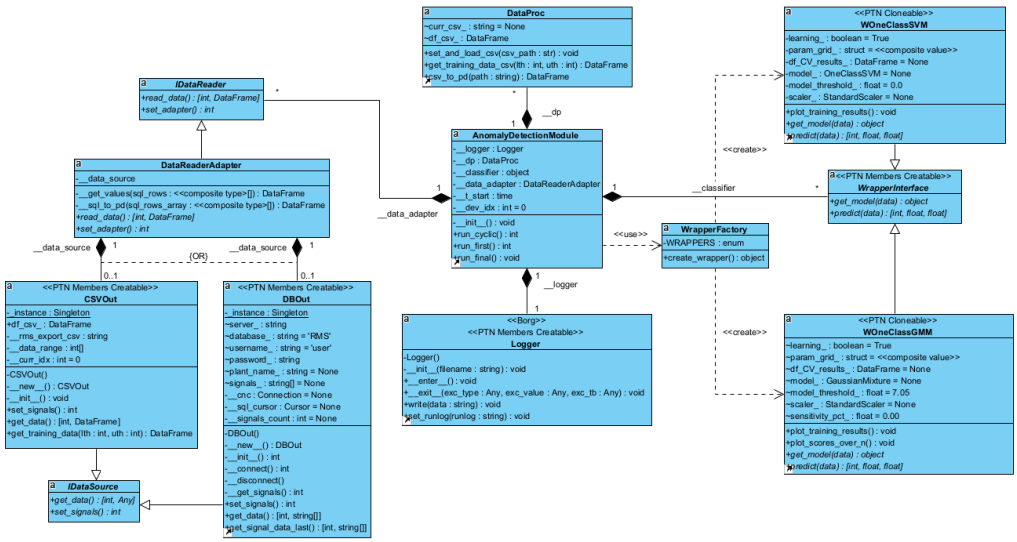
je pak používána pro klasifikaci předaných dat – obrazu – pomocí připraveného klasifikátoru. Třída ještě obsahuje soukromé metody pro výpočet AIC a BIC při procesu učení a metody generující grafické znázornění výsledků procesu učení. Ty jsou však volány pouze ve vývojovém módu.

Třída WOneClassSVM.

Druhá obalová třída *WOneClassSVM* také implementuje rozhraní *WrapperInterface*. Nachází se v souboru `w_oneclasssvm.py` a slouží jako obal pro třídu OneClassSVM knihovny sklearn. Ta je obsluhována, stejně jako v případě *WOneClassGMM*, pomocí implementovaných metod rozhraní. I v tomto případě je možné získat klasifikátor voláním metody *get_model()* a provádět klasifikaci dat voláním metody *predict()*.

3.4.5 UML Class Diagram

Strukturu popsaných tříd a vazby mezi nimi ilustruje následující UML diagram tříd. Jsou zde dobře patrné implementace jednotlivých rozhraní a způsob, jakým jsou třídy podružných balíků navázány na hlavní třídu *AnomalyDetectionModule*. Ve vazbách mezi třídou *DataReaderAdapter* a třídami *DBOut* a *CSVOut*, rozhraními, jež implementují, a jejich začleněním do hlavní třídy, lze snadno rozeznat návrhový vzor Adapter. Obdobně je možné pozorovat prvky návrhového vzoru Factory method mezi třídami *WrapperFactory*, *WOneClassGMM* a *WOneClassSVM* a jejich začleněním do hlavní třídy.



Obrázek 3.4: UML diagram tříd modulu detekce anomálií

3.5 Implementace řešení

V této části bude detailně popsána konkrétní realizace klíčových funkčností a struktur představených v popisu architektury. Vzhledem k zaměření této práce bude důraz kladen zejména na řešení komunikace s databází a implementaci obalových tříd klasifikátorů, zejména pak na proces učení.

3.5.1 Komunikace s databází

O komunikaci s SQL databází se stará třída *DBOut* podružného balíku io. Využívá k tomu prostředky knihovnu pyodbc a dotazy v jazyce SQL.

Inicializace třídy a první připojení.

Při vytváření třídy instance je volána metoda *__new__*(), jež v kombinaci s atributem třídy *_instance* zajišťuje implementaci návrhového vzoru Singleton. Není-li hodnota atributu None je místo vytvoření nové instance vrácen ukazatel na již existující instanci třídy *DBOut*. Následuje volání metody *__init__()*, kde jsou atributy sloužící k připojení k databázi inicializovány hodnotami z config.yaml. Těmito atributy jsou IP adresa serveru *server_*, jméno databáze *database_*, uživatelské iniciály *user_* a *password_*, název provozu *plant_name_* a v neposlední řetězec string *signal_list_*. Poslední jmenovaný atribut je podstatný i pro samotnou detekci anomálií, neboť vyjmenovává všechny signály společné domény, které mají sloužit k detekci stavu turbíny. Posledním krokem inicializace je volání metody *set_signals()*, které musí uskutečnit klientská třída. V metodě *set_signals()* je navázáno první spojení s databází pomocí soukromé metody *__connect()*.

Metoda *__connect()* využívá k připojení k databázi nástroje knihovny pyodbc. Nejprve je pomocí příkazu connect vytvořen objekt Connection. Tento objekt spravuje všechny operace provedené pomocí připojení ODBC. Každý objekt Connection vždy spravuje právě jedno připojení ODBC a provádí všechny transakce s databází. Pro úspěšné připojení musí příkaz connect obsahovat následující řetězec:

```
DRIVER={ODBC Driver 18 for SQL Server}; ENCRYPT=yes;
TrustServerCertificate=yes; Trusted_Connection=no;
ColumnEncryption=Enabled; SERVER=server_;
DATABASE=database_; UID=username_; PWD=password_
```

Tento řetězec definuje ovladač, který má být k připojení použitý, dále že připojení bude šifrované k šifrování má být použit podepsaný certifikát vydaný serverem. Klíčové slovo ColumnEncryption zapíná funkčnost ovladače Always Encrypted. V tomto režimu ovladač šifruje i hodnoty parametrů svázané se šifrovanými daty sloupců tabulek databáze. Dále je v řetězci klíčovým slovem Trusted_Connection specifikováno, že k autorizaci nemá být použit Windows Authentication Mode ale

iniciály dodané klíčovými slovy UID a PWD. Poslední klíčová slova – SERVER a DATABASE – specifikují adresu serveru a jméno databáze, k níž má být připojení vytvořeno.

Je-li úspěšně navázáno zabezpečené připojení k databázi pokračuje se vytvořením objektu Cursor příkazem cursor knihovny pyodbc. Tento objekt reprezentuje ukazatel do databáze. Neprovádí žádné transakce s databází, slouží pouze k výběru dat v databázi a říká objektu Connection zda a jakou transakci má vykonat. Po vykonání příkazu pak ukládá její výsledek spolu s posledním provedeným příkazem. Jeden objekt Connection může vytvořit a obsluhovat libovolný počet Cursorů i v případě, že databáze umožňuje pouze jedno aktivní připojení. V případě, že vytvoření kteréhokoli z těchto dvou objektů selže, informuje o tom metoda *__connect()* chybovým kódem 0x10 v návratové hodnotě.

Provádění metody *set_signals()* v případě úspěšného volání metody *__connect()* pokračuje voláním soukromé metody *__get_signals()*. V této metodě je vytvořen a pomocí cursoru proveden následující SQL příkaz využívající hodnotu *plant_name_*:

```
SELECT [PlantID] FROM [RMS].[dbo].[PLANT]
  WHERE [PlantName]={plant_name_}
```

Tedy výběr záznamu o aktuálním zařízení specifikovaném svým jménem v proměnné *plant_name_*, na němž se provádí detekce anomálií stavů turbíny, z tabulky všech zařízení v systému RMS. SQL příkaz je vykonán příkazem execute objektu Cursor. Není-li zařízení v databázi nalezeno metoda *__get_signals()* končí s chybovým kódem 0x11 v návratové hodnotě. V opačném případě jsou takto získané údaje, spolu s řetězcem *signal_list_*, použity v dalším SQL příkazu:

```
SELECT [SigID],[SigName],[SigDesc],[SigUnit],[PlotType],
  [SigDec],[Display], [SignalGroupID]
  FROM [RMS].[dbo].[SIGNAL]
  WHERE [PlantID]=plantID AND [SigName] IN (signal_list_)
```

Jsou tedy získány informace o signálech společné domény specifikovaných jejich jménem v řetězci *signal_list_* na vybraném zařízení. Všechny obdržené informace jsou uloženy do atributu třídy *signals_*. Tento krok je nutný mimo jiné k získání identifikátoru jednotlivých signálu. Konkrétní signály, například rychlost turbíny, mají zpravidla na různých zařízeních různé označení. Doposud totiž nebyla výrobci ani provozovateli průmyslových turbín definována společná a závazná nomenklatura měřených signálů. Je-li počet vrácených záznamů z tabulky signálů SQL databáze nenulový metoda úspěšně končí s návratovou hodnotou 0x00. V opačném případě je v návratové hodnotě chybový kód 0x12. Po návratu zpět do metody *set_signals()* zbývá poslední krok v podobě volání soukromé metody *__disconnect()*.

V metodě *__disconnect()* je nejprve příkazem close uzavřen vytvořený Cursor aby mohlo být následně příkazem close ukončeno spojení s SQL databází a uzavřen objekt Connection. Tyto kroky je nutné učinit, neboť oba objekty jsou atributem

třídy *DBOut* a nedošlo by tedy k jejich smazání při opuštění současného kontextu. Připojení k SQL databázi je vždy ukončováno, aby nedocházelo ke zbytečnému blokování přístupu u serverů s omezeným počtem připojení. Metoda *set_signals()* tímto krokem končí, instance třídy *DBOut* je nyní plně inicializována a připravena k použití.

Dotazování dat z databáze.

DataReaderAdapter nyní může na požadavek hlavní třídy *AnomalyDetectionModule* poskytovat prostřednictvím *DBOut* z SQL databáze hodnoty vybraných signálů společné domény pro klasifikaci. Slouží k tomu metoda *get_data()*. Ta po zavolání nejprve zkontroluje, zda atribut *signals_* není prázdný. Pokud ano, je vrácena chybová hodnota 0x24. V opačném případě je voláním *__connect()* navázáno spojení s databází. V případě neúspěchu metoda končí s návratovou hodnotou 0x10, v opačném případě pokračuje metoda voláním *get_signal_data_last()*.

Zde je prostřednictvím cyklu pro každý signál v *signals_* za použití objektu Cursor dotazována SQL databáze na poslední uloženou hodnotu:

```
SELECT TOP 1 [SigID], [MTime], [MValue]
  FROM [RMS].[dbo].[plant_name_]
  WHERE [SigID]="signal" AND [MValue] IS NOT NULL
  ORDER BY [MTime] DESC
```

Protože hodnoty měření signálů jsou vkládány na konec tabulky, je v příkazu SQL nutné záznamy seřadit sestupně podle času uložení. Hodnoty NULL, tedy záznamy, ke kterým neexistuje validní hodnota měření, jsou ignorovány. Hodnoty nejsou synchronizovány v čase, neboť měření probíhají s různou periodou – pokud mezi jednotlivými voláními metody *get_data()* nebylo obdrženo nové měření některého ze signálů je vybrána nejnovější platná hodnota. Je zde učiněn předpoklad, že hodnota měření v tomto čase by byla shodná s předchozím měřením. Umělá synchronizace hodnot měření, například použitím Whittaker-Shannonova interpolačního vzorce, by do hodnot mohla vnášet artefakty, které by mohly znehodnotit výsledek procesu klasifikace. V závislosti na počtu sledovaných signálů by také vzrostly nároky na zdroje využité modulem detekce anomálií a došlo by k prodloužení doby provádění každého cyklu detekce anomálií.

Obdržené hodnoty měření signálů jsou uloženy do pole ve stejném pořadí, v jaké jsou signály seřazeny v *signals_*. Je-li délka tohoto pole nulová, tedy nebylo možné získat nová měření, metoda *get_signal_data_last()* končí s chybovou hodnotou 0x20. Pokud ne, je spojení s databází ukončeno voláním metody *__disconnect()* a metoda *get_data()* vrací hodnotu 0x00 a zmíněné pole hodnot měření. Tyto návratové hodnoty jsou předány zpět volajícímu, zpravidla metodě *read_data()* instance třídy *DataReaderAdapter*. Zde je pole hodnot převedeno do formátu Pandas DataFrame. Tento formát byl vybrán z důvodu kompatibility s metodami knihovny sklearn, jež je pou-

žita pro samotnou detekci anomálií. Vytvořený DataFrame je pak předán do hlavní třídy, kde jsou data použita pro samotný proces klasifikace.

3.5.2 Signály společné domény

Signály v databázi systému RMS lze s ohledem na detekci anomálních stavů turbíny rozdělit do skupin podle příslušnosti k jednotlivým funkčním celkům turbín nebo podle fyzikálních veličin měřených na turbínách jako celku či na jejich součástech. Jednotlivé skupiny byly definovány následovně:

1. Provozní veličiny
2. Pára a voda
3. Relativní rotorové vibrace
4. Absolutní statorové vibrace
5. Posuvy
6. Ucpávka a ventilační pára
7. Teplota ložisek – turbína
8. Teplota ložisek – převodovka
9. Teplota ložisek – generátor
10. Mazací a olejový systém
11. Hydraulický systém
12. Systém ochran
13. Vakuovací systém

Signály v těchto skupinách jsou označovány jako signály společné domény. Na různých zařízeních se mohou signály ve skupinách lišit, neboť na každém zařízení mohou být měřeny jiné signály a veličiny.

Pro validaci řešení navrženého v této práci budou použity signály skupiny Relativní rotorové vibrace, neboť lze na všech zařízeních identifikovat signály náležící do této skupiny. V této skupině jsou také známé anomálie označené expertem, na kterých bude možné řešení validovat.

3.5.3 Rozhraní pro knihovnu scikit-learn

Jádrem detekce anomálií vytvořeného řešení jsou algoritmy knihovny scikit-learn, pro něž byly vytvořeny obalové třídy popsané v předešlých kapitolách. Tyto obalové třídy, spolu s třídou *WrapperFactory*, tvoří jednotné rozhraní pro tyto algoritmy,

jež umožňuje jejich snadnou integraci do zbytku modulu. Aktuálně nabízí modul dvě obalové třídy pro dva vybrané algoritmy: GaussianMixture a OneClassSVM. První jmenovaný byl vybrán z důvodu dobré interpretovatelnosti jeho výstupu a jednoduché struktuře, které tento algoritmus činí člověku snadno představitelným. OneClassSVM byl vybrán jako zástupce běžně používaných algoritmů detekce anomálií navržený přímo pro řešení této úlohy.

GaussianMixture je třída balíku mixture knihovny sklearn, která implementuje algoritmy pro modelování Gaussovských směsí. Tato konkrétní třída umožňuje modelovat pravděpodobnostní rozložení Gaussovských směsí na základě poskytnutých trénovacích dat. Z hotového modelu je pak možné vzorkovat data či predikovat, zda neznámá data mohla být vygenerována modelovaným pravděpodobnostním rozdělením. Třída ještě poskytuje metody pro výpočet hodnoty Akaikova informačního kritéria a Bayesova informačního kritéria.

Třída OneClassSVM balíku svm knihovny sklearn implementuje Metodu podpůrných vektorů ve verzi s jednou třídou, a tedy učením bez učitele. Procesem učení je zkonstruována „měkká" rozdělující nadrovina. To znamená že při její konstrukci jsou použity penalizační proměnné místo tvrdých omezení kladených Lagrangiánem (2.29). Natrénovaný klasifikátor je pak možné použít k detekci outlierů v nových datech.

Inicializace obalové třídy.

V případě obou tříd probíhá při tvorbě jejich instance inicializace jejich atributů. V tomto kroku jsou nastavena výchozí hodnota prahu *model_treshold_*, jež se používá jako prahová hodnota v rozhodovací funkci klasifikátoru. Následně probíhá kontrola existence souborů s uloženým natrénovaným klasifikátorem a scalerem. Pokud alespoň jeden z nich neexistuje nastaví se interní přepínač *learning_*, kterým bude aktivováno učení nového klasifikátoru a vytvoření nového scaleru. Stejný efekt má i nastavení nuceného učení jedním z parametrů v config.yaml.

Jakmile je instance obalové třídy vytvořena je ještě nutné připravit scaler a klasifikátor. Toho je dosaženo voláním metody *get_model()*, jež jako argument obdrží trénovací množinu dat. Protože integrace do systému RMS nebyla zatím provedena slouží jako zdroj těchto dat třída *DataProc*, která připraví Pandas DataFrame ze souboru exportu RMS specifikovaného v config.yaml, v rozsahu dat taktéž specifikovaným parametrem v config.yaml. Zvažovanou alternativou je vždy k učení používat CSV soubor exportu RMS.

Metoda začíná kontrolou hodnoty přepínače *learning_*. Pokud byl tento přepínač nastaven dojde k vytvoření nového objektu StandardScaler knihovny sklearn. Následně je volána metoda *fit()* tohoto objektu s trénovacími daty jako argumentem. Scaler automaticky zjistí střední hodnotu a směrodatnou odchylku dat v každém rozměru a tyto parametry uloží do vnitřních proměnných. Ty jsou pak v budoucnu

použity pro centrování a škálování podle předpisu

$$z = \frac{x - \mu}{\sigma}$$

Normalizace dat je běžným požadavkem v mnoha úlohách strojového učení. Centrovaná data s jednotkovým rozptylem se zpracovávají daleko snáze než data, jejichž složky jsou v hodnotách řádově rozdílné. Takto transformovaná data usnadní učícímu procesu algoritmu GaussianMixture hledání parametrů jednotlivých komponent, protože jednotlivé složky dat se budou podobat datům normálního rozdělení (nulová střední hodnota, jednotkový rozptyl). Učící algoritmus OneClassSVM s jádrem RBF implementovaný knihovnou sklearn přímo předpokládá, že všechny složky dat jsou soustředěny kolem 0 a mají varianci stejného řádu. Není-li proces učení v metodě *get_model()* aktivní je scaler načten ze souboru, jež je na Obr. 3.2 ilustrován jako `scaler.pickle`.

Dalším krokem je příprava klasifikátoru. Pokud byl přepínač *learning_* nastaven dochází k vytvoření nového objektu klasifikátoru knihovny sklearn a následně ke spuštění učícího procesu za použití normalizovaných trénovacích dat. Proces učení obou nabízených klasifikátorů bude podrobně popsán v samostatné kapitole. V případě, že učení není aktivní, je i klasifikátor načten ze souboru. Tento soubor je v Obr. 3.2 ilustrován jako `classifier.pickle`. Pokud proběhl proces učení jsou na závěr volání metody *get_model()* scaler i klasifikátor uloženy do příslušných souborů.

Rozhodovací funkce.

Rozhodovací funkce obou klasifikátorů jsou v obalových třídách zapouzdřeny funkcí *predict()*. Provádění této metody, která jako argument přijímá nová data, začíná vždy normalizací těchto dat. Následuje výpočet vzdálenosti nových dat od třídy normálních dat. Podoba tohoto výpočtu se liší v závislosti na klasifikátoru a bude popsána v příslušných kapitolách. Dalším krokem je klasifikace nových dat. I tato operace závisí na konkrétním klasifikátoru a bude také podrobněji popsána v pozdějších kapitolách. U obalové třídy pro GaussianMixture je navíc ještě odhadnuta pravděpodobnost, s jakou patří nová data do třídy normálních dat.

3.5.4 Výběr a učení klasifikátoru

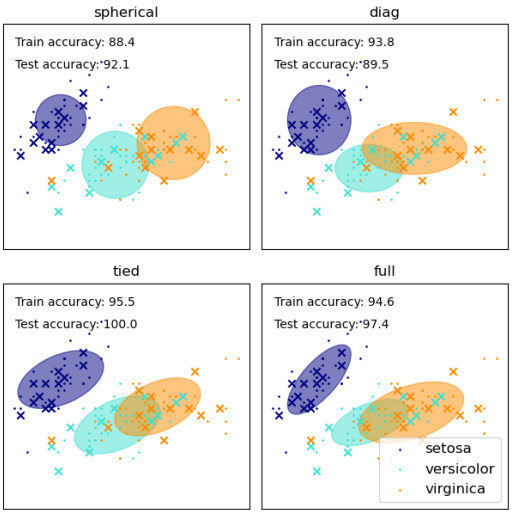
Modul detekce anomálií umožňuje použití pouze jednoho klasifikátoru, klasifikátory není možné za běhu modulu měnit. Toto omezení plyne z architektury modulu a nutnosti inicializovat obalovou třídu klasifikátoru a samotný klasifikátor. Je tedy nutné předem určit, který z podporovaných klasifikátorů má být použit. Za tímto účelem definuje config.yaml parametr *model*. Je-li nastaven na hodnotu „GMM“ bude použita obalová třída pro *GaussianMixture*, je-li nastaven na hodnotu „SVM“ bude

použita obalová třída pro *OneClassSVM*. Tento rozhodovací proces probíhá v metodě *create_wrapper()* třídy *WrapperFactory*, volané z metody *run_first()* hlavní třídy. Inicializace obdržené instance obalové třídy je pak dokončena voláním *get_model()*. Právě zde, jak popisuje předchozí kapitola, je implementován proces učení klasifikátoru.

Proces trénování GMM.

Všechny modely knihovny sklearn implementují jednotné rozhraní pro práci s modelem. K tomuto rozhraní patří metoda *fit()*, která provede učení modelu, respektive v případě třídy *GaussianMixture* nalezení odhadu parametrů komponent směsi pomocí EM-algoritmu. Při volání konstruktoru této třídy je nutné zadat parametry, které určí základní vlastnosti modelu a chování EM-algoritmu.

Parametry modelu zahrnují požadovaný počet komponent směsi a typ kovariance komponent. Na výběr jsou kovariance typu spherical, diagonal, tied a full. Pro typ spherical bude mít každá komponenta směsi vlastní varianci, která nebude nijak vázaná na ostatní komponenty. U typu diagonal budou kovarianční matice všech komponent diagonální. Je-li vybrán typ tied budou všechny komponenty sdílet stejnou diagonální kovarianční matici. Konečně pro typ full bude mít každá komponenta svou vlastní kovarianční matici bez jakýchkoliv restrikcí. Praktický dopad tohoto nastavení ilustruje následující srovnání.



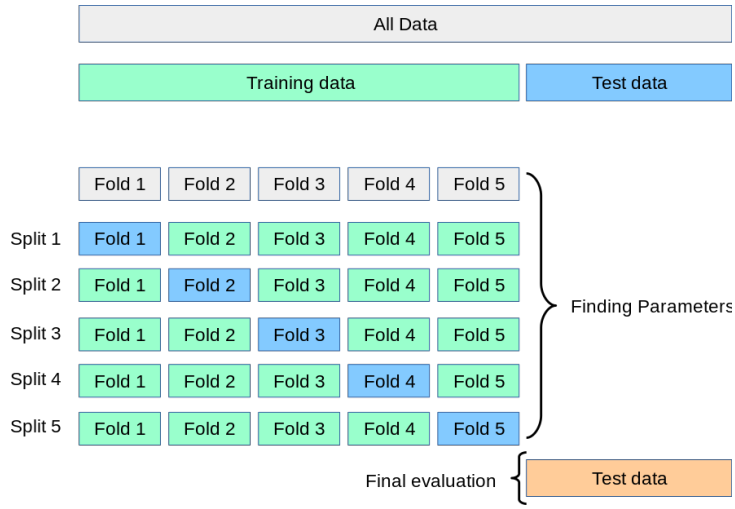
Obrázek 3.5: Ukázka různých typů kovariance komponent. Převzato z [Ped+11].

V konstruktoru je ještě možné zadat regularizační konstantu, která je přičtena k diagonále kovariančních matic, čímž lze zajistit pozitivní definitnost těchto matic. Výchozí hodnota tohoto parametru je 1e-6.

Možnosti nastavení procesu hledání odhadu parametrů je o poznání bohatší. Je možné zadat maximální počet iterací EM-algoritmu, zvolit hodnotu kritéria konvergence, při jejímž dosažení algoritmus skončí nebo určit, jakým způsobem bude EM-algoritmus inicializován. Uživatel může poskytnout vlastní počáteční hodnoty volných parametrů směsi tedy střední hodnoty, inverze kovariančních matic a hodnoty koeficientů směšování. Pokud tak neučiní proběhne inicializace EM-algoritmu pomocí jedné z podporovaných metod. Použitou metodu může uživatel nastavit jedním z parametrů konstruktoru. Na výběr jsou metody kmeans, kmeans++, náhodný výběr z trénovací množiny a pseudonáhodná čísla. Výchozí metodou pro inicializaci EM-algoritmu je kmeans. Počet provedených inicializací lze nastavit další parametrem konstruktoru, jehož výchozí hodnota je 1. Je-li zadána vyšší hodnota bude odhad parametrů směsi proveden pro každou inicializaci. Pro finální sestavení modelu budou použity takové parametry, se kterými dosáhl model nejvyšší hodnoty věrohodnosti.

Hledání optimálních parametrů směsi touto cestou by při předpokládané velikosti trénovací množiny a možných kombinací parametrů modelu a algoritmu bylo implementačně velice náročné. Řešení nabízí jeden ze širokého repertoár nástrojů knihovny sklearn v podobě třídy *GridSearchCV*. Tato třída realizuje přístup k hledání hodnot parametrů zvaný exhaustive grid search. Ten spočívá ve vyčerpávajícím hledání kombinace hodnot parametrů poskytnutých uživatelem, který vede k nejlepšímu výsledku z hlediska určeného kritéria. Většina trénovací množiny je použita pro trénování kandidátů, menší část je rezervována jako testovací množina pro hodnocení natrénovaných klasifikátorů. Každý kandidát může být navíc podroben křížové validaci. Křížová validace je implementována algoritmem k-fold. Část trénovací množiny, která je použita pro vlastní trénování, je dále rozdělena na k částí přibližně stejné velikosti. Každý kandidát je pak postupně natrénován k krát na $k - 1$ částech trénovací množiny. Poslední $k$tá část je vždy použita jako validační množina pro výpočet hodnotícího kritéria. Proces křížové validace slouží jako ochrana před overfittingem. Po skončení celého procesu hledání parametrů je nejlepší kandidát znovu natrénován na celé trénovací množině. Tím algoritmus končí. V atributech objektu *GridSearchCV* je v tuto chvíli k dispozici mnoho detailů k jeho průběhu a instance *GaussianMixture* s „vítěznou“ kombinací parametrů a skóre, kterého dosáhla. Tento způsob hledání odhadu parametrů je použit v implementaci třídy *WOneClassGMM*.

Struktura hodnot parametrů, jež má *GridSearchCV* použít, byla zvolena následovně: počet komponent směsi 2 až 20, všechny typy kovariance. Celkem tedy 76 kandidátů. Způsob inicializace EM-algoritmu byl ponechán ve výchozím nastavení, tedy kmeans. Obdobně bylo ponecháno i výchozí nastavení maximálního počtu iterací EM-algoritmu, tedy 100, hodnota kritéria konvergence a počet inicializací byly též ponechány ve výchozím nastavení. Jako hodnotící funkce pro výběr nejlepší



Obrázek 3.6: Dělení trénovací množiny algoritmem k-fold. Převzato z [Ped+11].

kombinace parametrů bylo vybráno Bayesovo informační kritérium. Křížová validace byla ponechána ve výchozí hodnotě, tedy pětinásobná křížová validace, čím se celkový počet nutných procesů trénování dostal na 380. Navíc bylo v parametrech *GridSearchCV* nastaveno spouštění dvou úloh paralelně, což výrazně zrychlilo proces hledání parametrů. Proces výběru nejlepšího ze 76 kandidátů s trénovací množinou o 225 000 obrazech rozměru 14 s pětinásobnou validací (380 procesů učení) tak trvá pouze v řádu jednotek hodin.

Shrnutí algoritmu exhaustive grid search.

1. Inicializace: vygeneruj kandidáty jako všechny možné kombinace zadaných parametrů
2. Rozděl trénovací množinu na k částí.
3. Každého kandidáta natrénuj $k$krát na $k-1$ částech trénovací množiny a pomocí $k$té části vypočti hodnotu hodnotícího kritéria.
4. Vypočti celkové skóre kandidáta jako průměr k hodnot hodnotícího kritéria.
5. Vyber kandidáta s nejvyšším celkovým skóre jako nejlepšího kandidáta.
6. Natrénuj nejlepšího kandidáta na celé trénovací množině. Algoritmus končí.

Proces trénování OCSVM.

Trénování klasifikátoru *OneClassSVM* funguje analogicky k hledání optimálních parametrů *GaussianMixture*. I pro tuto třídu je nutné nastavit některé parametry klasifikátoru již v parametrech konstruktoru. Samotný proces učení opět realizuje

metoda *fit()*. U třídy *OneClassSVM* definují parametry konstruktoru kernel, který má OCSVM používat, volné parametry ν a γ, který definuje vliv jednotlivých obrazů trénovací množiny. Velmi malé hodnoty γ odpovídají omezenému modelu s horší schopností zachytit komplexitu dat, výsledný model by měl podobné chování jako model se sadou lineárních nadrovin. Příliš velké hodnoty γ snižuje se vliv podpůrných vektorů a model naopak nebude schopen generalizovat. Dále je možné nastavit parametry specifické pro jednotlivé jádrové funkce, maximální počet iterací a prahovou hodnotu pro zastavovací kritérium optimalizačního algoritmu. Proces trénování OCSVM je v knihovně sklearn implementován tak jak jej popisují Schölkopf et al. ve své práci a jak bylo popsáno v kapitole 2.4.

Samozřejmě i v tomto případě by hledání optimálních parametrů OCSVM pouze s využitím metody *fit()* bylo implementačně náročné. Přirozeně se opět nabízelo použití objektu *GridSearchCV*. Během testování toho řešení však proces učení s 32 kandidáty s trojnásobnou křížovou validací, tedy pouhých 96 procesů trénování, a relativně malou trénovací množinou 100 000 obrazů trval neúnosně dlouho, až v řádu nízkých desítek hodin. Implementace byla proto změněna a třída *GridSearchCV* byla nahrazena experimentální třídou *HalvingGridSearchCV*, která používá přístup označovaný jako successive halving. Jedná se o iterativní algoritmus pracující se zdroji, kandidáty a hodnotícím kritériem. Zdroje označují obrazy trénovací množiny, kandidáti specifické kombinace hodnot parametrů. Hodnotící kritérium volí uživatel. V první iteraci použije algoritmus pouze malou část zdrojů k natrénování všech kandidátů a výpočtu hodnotícího kritéria pro každého z nich. Kandidáti jsou seřazeni podle jejich hodnocení a poměrný počet nejhůře hodnocených je diskvalifikován. Ve druhé iteraci je počet zdrojů navýšen o stejný poměr, jako poměr kandidátů, kteří nepostoupili, a zbývající kandidáti jsou s jejich využitím natrénování a ohodnoceni. Zbývající kandidáti jsou opět seřazeni podle jejich hodnocení a poměrný počet nejhůře hodnocených je vyřazen. Tento proces se opakuje, dokud nezbydou poslední dva kandidáti nebo dokud algoritmus nedosáhne předem definovaného maximálního počtu iterací. Stejně jako *GridSearchCV* i tento algoritmus umožňuje použití k-fold křížové validace pro snížení rizika overfittingu. *HalvingGridSearchCV* umožňuje celou řadu nastavení od maximálního či minimálního počtu zdrojů použitého v každé iteraci až po faktor dělení, o nějž má být množina kandidátů v každé iteraci snížena a počet zdrojů navýšen.

Nakonec byla struktura hodnot parametrů pro proces successive halving definována takto: kernel RBF, $\nu \in \{0,001; 0,01; 0,1\}$ a $\gamma \in \{0,001; 0,018; 0,316; 5,623; 100\}$. Počet kandidátů byl tak záměrně snížen z 32 na 15 vyřazením lineárního kernelu, který vždy přinášel horší výsledky než kernel RBF, a ubráním nejvyšší hodnoty z množiny možných hodnot ν. Počet křížových validací, jež mají být provedeny, byl naopak zvýšen na výchozích 5. Jako hodnotící funkce pro *HalvingGridSearchCV* bylo vybráno recall score a faktor byl nastaven na 3. Toto skóre počítá poměr po-

zitivních výsledků (normální obrazy) k součtu pozitivních výsledků a falešně negativních výsledků, tedy schopnost klasifikátoru správně zařadit všechny inliery. Recall score bylo vybráno na základě požadavku na minimalizaci falešně pozitivních výsledků. I *HalvingGridSearchCV* byl nastaven tak, aby zpracovával paralelně dvě úlohy, což vedlo k dalšímu zrychlení procesu. Výběr nejlepší kombinace parametrů OCSVM s jádrem RBF pro 15 kandidátů na trénovací množině 225 000 obrazů s pětinásobnou křížovou validací (75 procesů učení) nyní trvá méně než dvě hodiny. Rychlost hledání nejvhodnějšího kandidáta je srovnatelná s výběrem nejlepšího modelu GMM.

Shrnutí algoritmu successive halving.

1. Inicializace: vygeneruj kandidáty jako všechny možné kombinace zadaných parametrů
2. Urči počet iterací n postupným dělením počtu kandidátů faktorem f.
3. Rozděl trénovací množinu na počet $m = n \cdot f$ částí.
4. Rozděl každou část trénovací množiny na k částí.
5. Každého kandidáta x natrénuj $k$krát na $k - 1$ částech množiny m_x a pomocí $k$té části vypočti hodnotu hodnotícího kritéria.
6. Vypočti celkové skóre každého kandidáta jako průměr k hodnot hodnotícího kritéria.
7. Vyber $1/f$ kandidátů s nejhorším skóre a vyřaď je.
8. Zvětši trénovací množinu $f$krát.
9. Pokud zbývá f a více kandidátů a nebyl dosažen maximální počet iterací, pokračuj krokem 3.
10. Pokud zbyl více než 1 kandidát aplikuj algoritmus exhaustive grid search.
11. Natrénuj nejlepšího kandidáta na celé trénovací množině. Algoritmus končí.

Příklad vývoje množství zdrojů a počtu kandidátů pro 15 kandidátů, trénovací množině s 225 000 obrazy a faktorem 3:

Iterace	Množství zdrojů na kandidáta	Počet kandidátů
1	25 000	15
2	$25\,000 \cdot 3 = 75\,000$	$15//3 = 5$
3	$75\,000 \cdot 3 = 225\,000$	$5//3 = 2$

Tabulka 3.1: Příklad postupů iterací succesive halving

3.5.5 Detekce anomálií

Natrénovaný klasifikátor knihovny sklearn nabízí v rámci jednotného rozhraní několik metod použitelných v procesu klasifikace. Všechny tyto metody mají jediný argument – obraz, který má být klasifikátorem vyhodnocen. Nejpřímější výsledek poskytuje metoda *predict()*, jejíž návratová hodnota vypovídá o zařazení obrazu do některé z tříd, respektive v případě detekce anomálií vypovídá o identifikaci obrazu jako outliera nebo inliera. Další možností jak získat výstup klasifikátoru je voláním metody *score_samples()*. Ta vrací nezpracovaný výstup rozhodovací funkce, jež může uživatel dále zpracovat dle vlastního uvážení. Klasifikátory sklearn však zpřístupňují i samotnou rozhodovací funkcí klasifikátoru prostřednictví metody *decision_function()*.

V navrženém modulu detekce anomálií je klasifikace nových vzorků dat, obdržených prostřednictvím *DataReaderAdapter* z SQL databáze nebo ve vývojovém módu ze souboru CSV exportu databáze RMS, zprostředkována obalovými třídami podružného balíku wf. Obalové třídy k tomuto účelu implementují metodu *predict()* rozhraní *WrapperInterface*. První operací v této metodě je vždy normalizace nového obrazu, tím ale podobnost mezi implementacemi obalových tříd končí.

Detekce anomálií s použitím GMM.

Obalová třída *WOneClassGMM* nabízí dva způsoby klasifikace nových obrazů. O tom, který z nich bude použit, rozhoduje hodnota *model_gmm_sensitivity_pct* v config.yaml. Je-li nastavena na jinou hodnotu než 0,00 bude klasifikace probíhat pomocí metody sklearn *score_samples()*. Ta pro *GaussianMixture* vrací hodnotu logaritmu věrohodnosti, spočtenou jako průměr logaritmu věrohodnosti nového obrazu vůči všem komponentám směsi. Výsledek je pak porovnán s hodnotou s prahovou hodnotou spočtenou jako *model_gmm_sensitivity_pct*-percentil výsledku *score_samples()* pro celou trénovací množinu. Tato prahová hodnota je vypočítána předem po skončení procesu učení. Je-li logaritmus věrohodnosti nového obrazu menší než prahová hodnota je obraz prohlášen za outliera. Tento způsob zpracování nových obrazů velmi spolehlivě detekuje outliery za cenu vyššího počtu falešně pozitivních výsledků, tedy chybně označených normálních obrazů.

Pokud je hodnota *model_gmm_sensitivity_pct* rovna 0,00 probíhá klasifikace na základě Mahalanobisovy vzdálenosti nového obrazu. Nejprve je spočtena Mahalanobisova vzdálenost mezi novým obrazem a každou komponentou směsi pomocí vztahu (2.7). Inverze kovarianční matice v tomto vztahu je počítána standardní metodou *inv()* numerické knihovny numpy, neboť její pozitivní definitnost je zaručena trénovacím algoritmem třídy GaussianMixture. Ze všech vypočtených vzdáleností je vybrána ta nejmenší, která je pak porovnávána s hodnotou *module_threshold* zadanou v config.yaml. Pokud je vzdálenost větší než tato hodnota je obraz prohlášen

za outliera. Tento přístup ke klasifikaci nových obrazů umožňuje prostřednictvím změny hodnoty *module_threshold* měnit citlivost klasifikace a upravit tak chování modulu detekce anomálií.

Poslední operací v metodě *predict()* je výpočet pravděpodobnosti, s jakou nový obraz náleží do třídy inlierů, tedy normálních obrazů. K tomuto výpočtu je použita metoda *predict_proba()* poskytovaná třídou *GaussianMixture*. Ta vrací pravděpodobnosti příslušnosti obrazu k jednotlivým komponentám směsi. Z nich je pak vybrána ta nejvyšší. Tato hodnota není použita pro klasifikaci nového obrazu a v kontextu metody *predict()* má pouze informativní charakter.

Metoda *predict()* obalové třídy *WOneClassGMM* má tři návratové hodnoty. První je označení nového obrazu hodnotou 1 pro outlier nebo 0 pro inlier, druhou je určená Mahalanobisova vzdálenost a třetí je pravděpodobnost příslušnosti ke třídě normálních obrazů.

Detekce anomálií s použitím OCSVM.

Obalová třída *WOneClassSVM* nabízí pouze jeden způsob klasifikace nových obrazů. Ten je založen na vzdálenosti obrazu od dělící nadroviny. Tuto vzdálenost poskytuje třídy *OneClassSVM* knihovny sklearn prostřednictvím metody *decision_function()*. K získané vzdálenosti je přičtena hodnota *model_threshold*. Pokud je tento součet záporné číslo, tedy i po přičtení prahové hodnoty se obraz nachází za hranicí třídy, je obraz označen za outliera. Tento způsob klasifikace přináší stejnou flexibilitu jako způsob detekce anomálií s použitím GMM za pomoci Mahalanobisovy vzdálenosti.

Metoda *predict()* žádné další operace neprovádí. V souladu s definicí rozhraní *WrapperInterface* má opět tři návratové hodnoty. První je označení nového obrazu hodnotou 1 nebo 0 pro outlier, respektive inlier. Druhou je vzdálenost obrazu od rozdělující nadroviny bez přičtené hodnoty *model_threshold*. Jelikož algoritmus OCSVM nepracuje s pravděpodobností není možné vypočítat pravděpodobnost příslušnosti obrazu k třídě normálních obrazů, proto je třetí návratová hodnota zafixována na hodnotu 0.

Míra anomality.

Druhá návratová hodnota metody *predict()* rozhraní *WrapperInterface* je vždy vzdálenost. Tu lze v kontextu vzdáleností normálních obrazů použít k vyhodnocení míry anomality obrazů klasifikovaných jako outlier. Čím vyšší je hodnota vzdálenosti outliera oproti běžným vzdálenostem tím větší je míra jeho anomality.

Validace řešení 4

Navržený modul detekce anomálií byl validován na reálných datech naměřených na turbínách dvou elektráren které provozují systém RMS. Na obou zařízeních byly expertem identifikovány dvě anomální události. Z databáze RMS byly ve formátu CSV exportovány časové úseky měření, v nichž se tyto události nacházejí. Anomální úseky byly ve všech čtyřech exportovaných sadách označeny expertem. Každá sada byla rozdělena na část pro trénování obou klasifikátorů a nastavení vhodné prahové hodnoty a na část použitou pro test detekce anomálií. Proces validace bude zahrnovat natrénování klasifikátoru, určení prahové hodnoty na vzorcích reprezentující normální stav, detekce anomálie a porovnání se vzorky označenými expertem. K výstupu modulu detekce anomálií bude vždy vypracován klasifikační report složený z několika základních metrik používaných pro hodnocení kvality klasifikace.

Použité metriky kvality klasifikace.

Precision score – schopnost klasifikátoru neoznačit obraz jako náležící do dané třídy, pokud do ní nenáleží. Hodnota pro každou třídu je určena poměrem $\frac{TP}{TP+FP}$, kde TP značí obraz, který náleží do třídy a byl klasifikován správně a FP značí obraz, který nenáleží do třídy a byl klasifikován špatně. Hodnota precision score · 100 je procento správně klasifikovaných obrazů třídy ze všech obrazů přiřazených této třídě.

Recall score – schopnost klasifikátoru najít všechny obrazy náležící do dané třídy. Hodnota je definována pro každou třídu jako poměr $\frac{TP}{TP+FN}$, kde TP značí obraz, který náleží do třídy a byl klasifikován správně a FN značí obraz, který do třídy náleží ale byl klasifikován špatně. Hodnota Recall score · 100 je procento správně klasifikovaných obrazů třídy ze všech správně klasifikovaných obrazů.

F1-score – vážený harmonický průměr hodnoty Precision score a Recall score. Nejlepší výsledek odpovídá hodnotě 1, nejhorší výsledek hodnotě 0.

Accuracy – souhrnná přesnost všech správných klasifikací.

4.1 Detekce anomálií na TG 36MW

Oba exporty z databáze systému RMS pro toto zařízení obsahují data pocházející z roku 2021. V CSV exportu pro událost 1 je 518 766 vzorků, v exportu pro událost 2 je 514 144 vzorků měření. Exportovaná měření patří do signálových skupin Provozní veličiny a Relativní rotorové vibrace. Detekce anomálií bude testována na signálech domény Relativních rotorových vibrací. Protože vibrace se mohou na zařízení projevovat odlišně v závislosti na aktuálním režimu, budou ke sledovaným signálům jako reference přidány měření výkonu a otáček ze signálové skupiny Provozní veličiny. Následující tabulka obsahuje kompletní seznam signálů jež budou použity pro detekci anomálního stavu turbíny. Jelikož identifikátory signálů jsou poměrně dlouhé bylo každému z nich přiřazeno písmeno, pod kterým bude figurovat ve všech grafech.

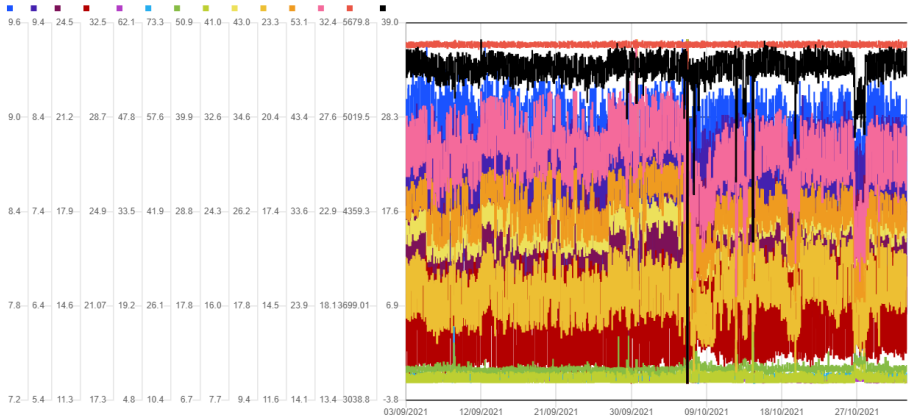
Index	Jméno	Popis signálu	Označení
1	VIBR.FRONT HP ROTOR - X	Turbína, vysokotlaká část, vibrace rotoru, osa X, vpředu	A
2	VIBR.FRONT HP ROTOR - Y	Turbína, vysokotlaká část, vibrace rotoru, osa Y, vpředu	B
3	VIBR.REAR HP ROTOR - X	Turbína, vysokotlaká část, vibrace rotoru, osa X, vzadu	C
4	VIBR.REAR HP ROTOR - Y	Turbína, vysokotlaká část, vibrace rotoru, osa Y, vzadu	D
5	VIBR.TR.DRIV.ROT - HS TUR SIDE-X	Převodovka, rychlootáčková část turbíny, vibrace rotoru, osa X	E
6	VIBR.TR.DRIV.ROT - HS TUR SIDE-Y	Převodovka, rychlootáčková část turbíny, vibrace rotoru, osa Y	F
7	VIBR.TR.DRIV.ROT - LS GEN SIDE-X	Převodovka, pomalootáčková část turbíny, vibrace rotoru, osa X	G
8	VIBR.TR.DRIV.ROT - LS GEN SIDE-Y	Převodovka, pomalootáčková část turbíny, vibrace rotoru, osa Y	H
9	VIBR.FRONT GEN. ROTOR - X	Generátor, vibrace rotoru, osa X, vpředu	I
10	VIBR.FRONT GEN. ROTOR - Y	Generátor, vibrace rotoru, osa Y, vpředu	J
11	VIBR.REAR GEN. ROTOR - X	Generátor, vibrace rotoru, osa X, vzadu	K

Index	Jméno	Popis signálu	Označení
12	VIBR.REAR GEN. ROTOR - Y	Generátor, vibrace rotoru, osa Y, vzadu	L
13	POWER OF TG	Výkon TG, aktuální	M
14	TURBINE SPEED	Otáčky turbíny	N

Tabulka 4.1: Přehled sledovaných signálů

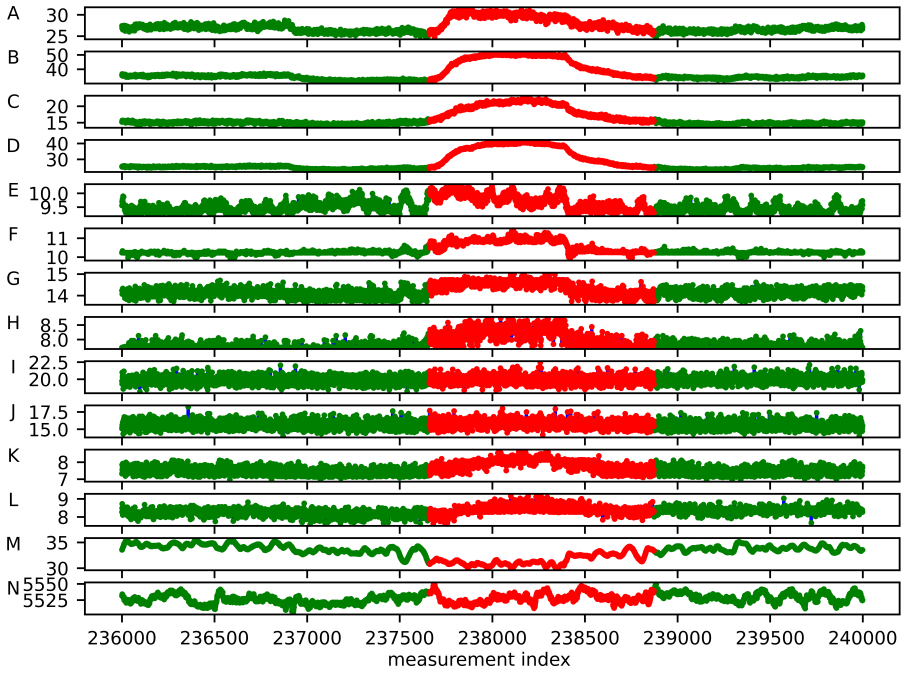
4.1.1 Událost 1 - popis

První událost byla expertem identifikována v rozsahu indexů 237 666 až 238 878. V grafech na Obr. 4.2 a Obr. 4.3 je tento úsek zachycen v kontextu okolních hodnot. Zelenou barvou jsou označeny vzorky označené expertem za normální, červenou pak vzorky identifikované expertem jako anomální. Stejný úsek dat bude použit jako testovací množina. Obr. 4.1 zachycuje celou sadu v grafickém rozhraní systému RMS.

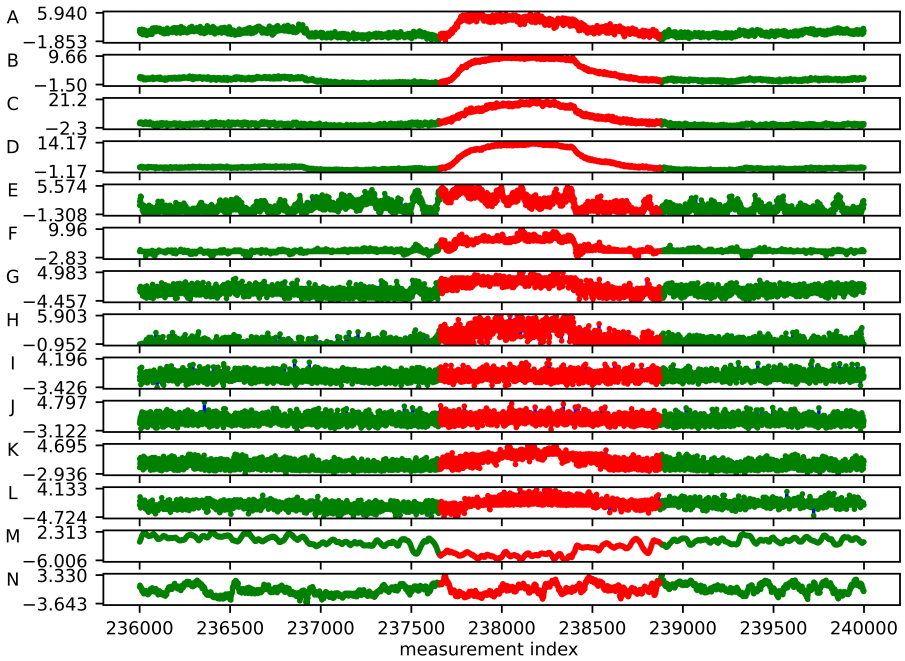


Název	Popis	Jednotka	Dolní	Horní	Min	Max
00CBP01CE900_XQ01	POWER OF TG	MW	-3.80	38.99	-1.85	37.05
00MAD20CS901	TURBINE SPEED	rpm	3038.75	5679.79	3158.80	5559.74
00MAD10CY001_XQ01	VIBR.FRONT HP ROTOR - X	µm	13.40	32.36	14.26	31.49
00MAD10CY002_XQ01	VIBR.FRONT HP ROTOR - Y	µm	14.10	53.14	15.88	51.36
00MAD20CY001_XQ01	VIBR.REAR HP ROTOR - X	µm	11.58	23.31	12.12	22.77
00MAD20CY002_XQ01	VIBR.REAR HP ROTOR - Y	µm	9.37	42.99	10.89	41.47
00MAK01CY001_XQ01	VIBR.TR.DRIV.ROT.- HS TUR SIDE-X	µm p-p	7.66	40.97	9.18	39.46
00MAK01CY002_XQ01	VIBR.TR.DRIV.ROT.- HS TUR SIDE-Y	µm p-p	6.72	50.91	8.73	48.91
00MAK01CY007_XQ01	VIBR.TR.DRIV.ROT.- LS GEN SIDE-X	µm p-p	10.43	73.29	13.28	70.44
00MAK01CY008_XQ01	VIBR.TR.DRIV.ROT.- LS GEN SIDE-Y	µm p-p	4.83	62.15	7.44	59.54
00MKD10CY001_XQ01	VIBR.FRONT GEN. ROTOR - X	µm	17.26	32.52	17.95	31.83
00MKD10CY002_XQ01	VIBR.FRONT GEN. ROTOR - Y	µm	11.29	24.54	11.89	23.94
00MKD20CY001_XQ01	VIBR.REAR GEN. ROTOR - X	µm	5.37	9.41	5.55	9.22
00MKD20CY002_XQ01	VIBR.REAR GEN. ROTOR - Y	µm	7.18	9.60	7.29	9.49

Obrázek 4.1: Celá datová sada dostupná pro událost 1



Obrázek 4.2: Testovací množina pro událost 1

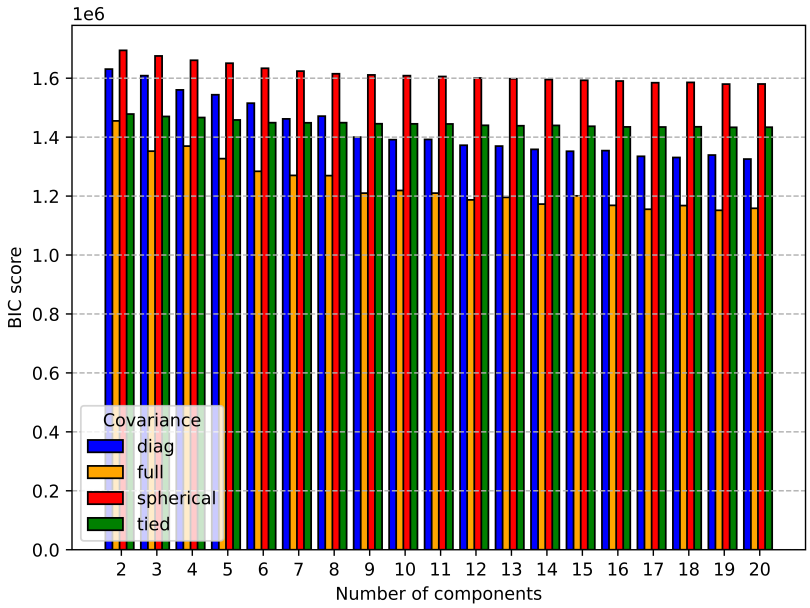


Obrázek 4.3: Normalizovaná testovací množina pro událost 1

4.1.2 Událost 1 - GMM

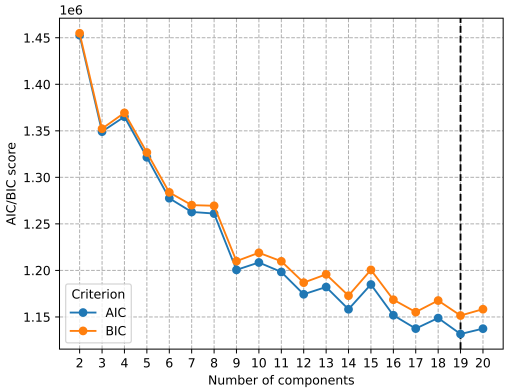
Trénování klasifikátoru.

Klasifikátor byl natrénován na prvních 225 000 vzorcích z exportu z celkových 518 766, tedy přibližně na 43 % dostupných dat. Tento úsek byl vybrán úmyslně s cílem získat co největší kontinuální sadu dat použitelnou pro proces učení s dostatečným odstupem od vyznačené anomálie a to pro případ, že by její rozsah nebyl expertem označen správně. Data byla na začátku trénování normalizována objektem StandardScaler vytvořeného nad daty trénovací množiny. Algoritmus exhaustive grid search prohledával strukturu hodnot obsahující počet komponent směsi $n \in \langle 2; 20 \rangle$ a všechny čtyři podporované typy kovariance komponent. Jako hodnotící kritérium bylo vybráno Bayesovo informační kritérium.



Obrázek 4.4: Srovnání BIC skóre všech kandidátů

Jako nejlepší byla algoritmem označena kombinace parametrů $n = 19$, typ kovariance full. Klasifikátor s touto kombinací parametrů dosáhl skóre BIC = 1 151 459,66. Porovnání BIC skóre pro všechny kandidáty zobrazuje Obr. 4.4, vývoj BIC skóre pro typ kovariance nejlepšího kandidáta je detailně zobrazen na Obr. 4.5. Pro srovnání je v grafu vyneseno i AIC skóre. K odhadnutí parametrů komponent směsi bylo potřeba 43 iterací EM-algoritmu. Celková doba běhu algoritmu exhaustive grid search byla 224,4 minut.

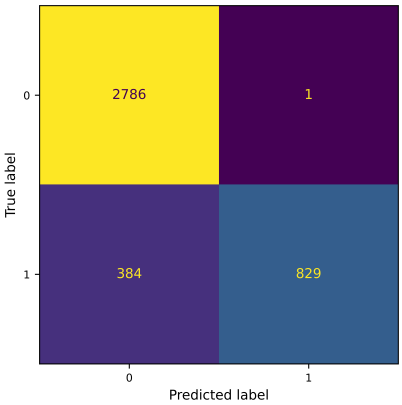


Obrázek 4.5: Vývoj BIC skóre pro typ kovariance full

Z obou grafů patrné, že nejvyšší počet komponent nemusí nutně mít nejnižší skóre. Dobrých výsledků by zřejmě dosahovala i směs s typem kovariance full a počtem komponent 17. Optimalizace výsledku algoritmu exhaustive grid search s ohledem na počet komponent by však vyžadovala vyřazení posledního kroku algoritmu, implementaci alternativního automatického vyhodnocení výsledků a také vlastní funkci pro přetrénování výsledného klasifikátoru.

Detekce anomálií.

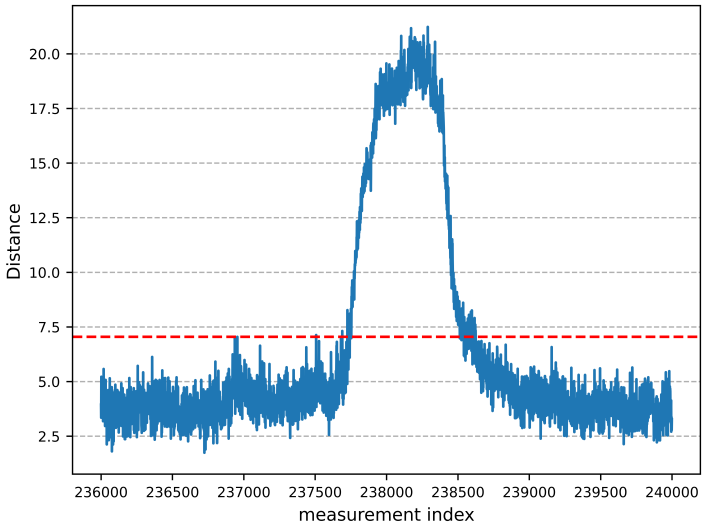
Protože je pozice události označené expertem v datech známá, byla detekce anomálií spuštěna na okolí této události, a to v rozsahu indexů 236 000 až 240 000. Celkem tedy bylo zpracováno 4 000 vzorků dat. Jejich zpracování trvalo modulu detekce anomálií 56,6 minut, tedy přibližně 0,85 sekund na jeden vzorek. Ze 4 000 zpracovávaných vzorku bylo špatně klasifikováno 385 neboli 9,62 %, z toho pouze jeden výsledek byl falešně pozitivní. Výsledky klasifikace shrnuje matice záměn:



Obrázek 4.6: Matice záměn[1]

[1] 0 značí normální obraz, 1 značí anomální obraz.

Následující graf vykresluje Mahalanobisovy vzdálenosti jednotlivých vzorků. Červená přerušovaná čára označuje prahovou hodnotu 7,05 stanovenou na základě pozorovaných vzdáleností normálních dat.



Obrázek 4.7: Mahalanobisovy vzdálenosti vzorků testovací množiny

Na grafu Mahalanobisových vzdáleností je zřetelně rozpoznatelná hledaná událost. Obrazy dat v tomto úseku se vyznačují o poznání větší vzdáleností od středu nejbližší komponenty směsi. S postupným rozvojem anomálie tato vzdálenost strmě roste, což indikuje rostoucí anomalitu obrazů.

Kvalitu provedené detekce anomálií lze vyhodnotit pohledem na klasifikační report.

	Precision score	Recall score	F1-score	Počet vzorků
Normální vzorky	0.8789	0.9996	0.9354	2787
Anomální vzorky	0.9988	0.6834	0.8116	1213
Vážený průměr[2]	0.9152	0.9038	0.8735	4000
Accuracy			0.9038	4000

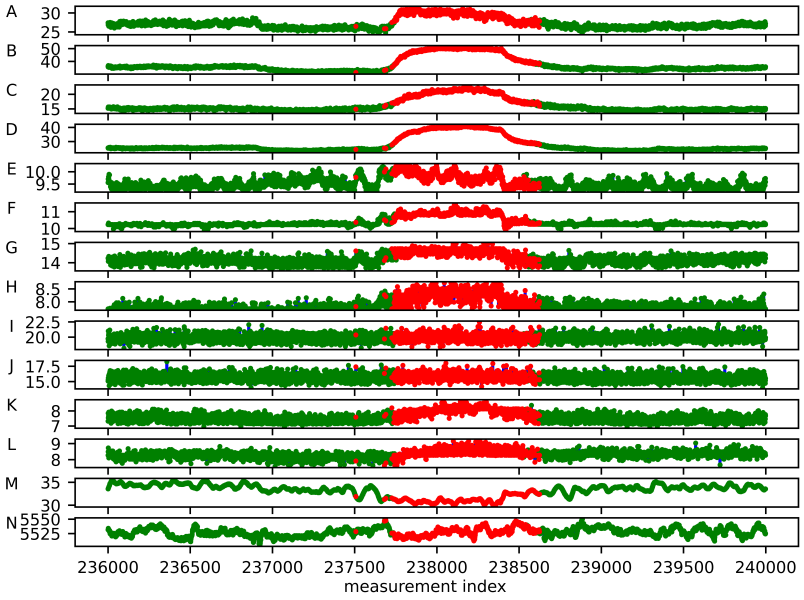
Tabulka 4.2: Klasifikační report

Z hodnoty precision score je patrné, že téměř 100 % obrazů identifikovaných jako anomálie byly skutečně anomálie. Pouze 1 normální obraz byl mylně klasifikován jako anomální, tedy 0,04 % všech normálních obrazů. Oproti tomu přibližně

[2]Vážený průměr je počítán jako součet počtu normálních vozků krát průměrné precision/recall/F1 score, a počtu anomálních vzorků krát průměrné precision/recall/F1 score.

32 % anomálií byla chybně klasifikována jako normální obrazy. To může být částečně způsobeno tím, že tyto obrazy skutečně anomální nejsou a byly expertem označeny chybně. Hledání přesného začátku a konce události v datech pouhým okem nemusí vést k jejímu přesnému ohraničení. Dalším důvodem může být konzervativní nastavení prahové hodnoty odkazující na snahu dosáhnout co nejnižšího počtu „planých poplachů“. Celková přesnost klasifikace byla 90,4 %.

Na posledním grafu je zobrazen stejný úsek dat jako v kapitole 4.1.1. Tentokrát není barevné označení vzorků vztaženo k rozhodnutí experta ale k výstupu modulu detekce anomálií.



Obrázek 4.8: Testovací množina označená klasifikátorem

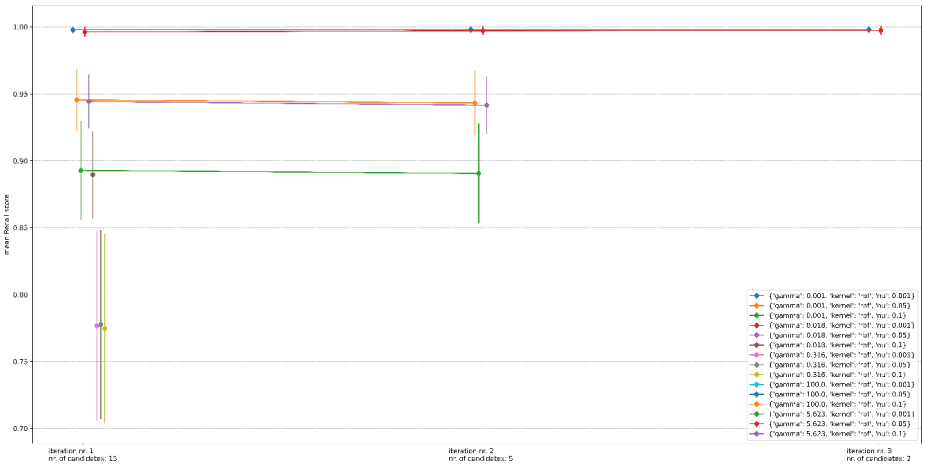
Srovnáním s Obr. 4.2 je patrné, že výstup klasifikátoru se s názorem experta rozchází pouze na samém vzniku události a krátce na jejím konci.

4.1.3 Událost 1 - SVM

Trénování klasifikátoru.

Klasifikátor byl opět natrénován na prvních 225 000 vzorcích a data byla před tréningem opět normalizována. Použití stejné trénovací množiny umožní lépe porovnat výkon obou klasifikátorů. Algoritmus successive halving procházel strukturou obsahují hodnoty $\nu \in \{0,001; 0,01; 0,1\}$, $\gamma \epsilon \{0,001; 0,018; 0,316; 5,623; 100\}$ pro zvolený kernel RBF. Jako hodnotící kritérium bylo použito Recall score. Algoritmus vyhodnotil jako nejlepší kombinaci parametrů $\nu = 0,001$; $\gamma = 0,001$. S touto kombinací parametrů dosáhl klasifikátor Recall score 0,9979. K dosažení výsledku bylo potřeba tří iterací algoritmu successive halving s celkovou dobou trvání

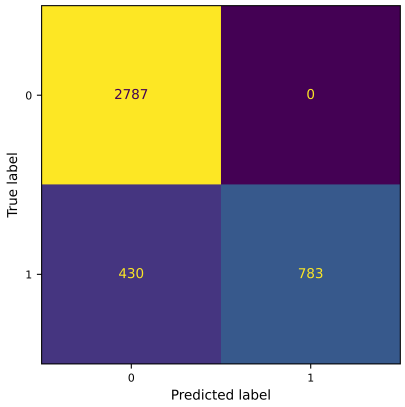
109,6 minut. Přehled vývoje průměrného skóre všech kandidátů včetně standardní odchylky v průběhu iterací algoritmu nabízí Obr. 4.9. Kombinace parametrů, jejichž skóre bylo rovno nebo blízké nule, jsou v grafu ignorovány.



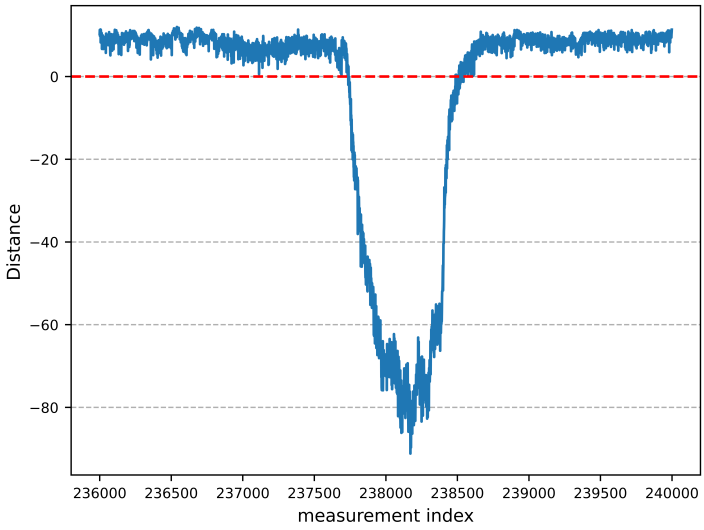
Obrázek 4.9: Vývoj skóre kandidátů v iterací successive halving

Detekce anomálií.

Detekce probíhala na stejném intervalu indexů v exportu RMS tedy 236 000 až 240 000, tedy 4 000 vzorků. Prahová hodnota byla ponechána ve výchozím nastavení 0,00. Při rozhodování o zařazení obrazu tedy nebude k vypočtené vzdálenosti od dělící nadroviny přičítán žádný ofset. Zpracování dat trvalo modulu detekce anomálií 54,2 minut. S časem přibližně 0,8 sekundy na jeden vzorek byl teda algoritmus OCSVM nepatrně rychlejší. Z celkového počtu 4 000 vzorků bylo špatně klasifikováno 430 vzorků, tedy 10,75 %. Žádný výsledek nebyl falešně pozitivní. Výsledky shrnuje matice záměn na Obr. 4.10.



Obrázek 4.10: Matice záměn



Obrázek 4.11: Vzdálenosti vzorků testovací množiny od dělící nadroviny

Vzdálenosti všech 4 000 vzorků jsou vykresleny na grafu na Obr. 4.11, kde červená přerušovaná čára zvýrazňuje nulovou vzdálenost od dělící nadroviny. Stejně jako u Mahalanobisových vzdáleností i zde je pozice anomální události v datech jasně patrná. Vzdálenost anomálních vzorků od dělicí nadroviny je výrazně větší než u normálních dat.

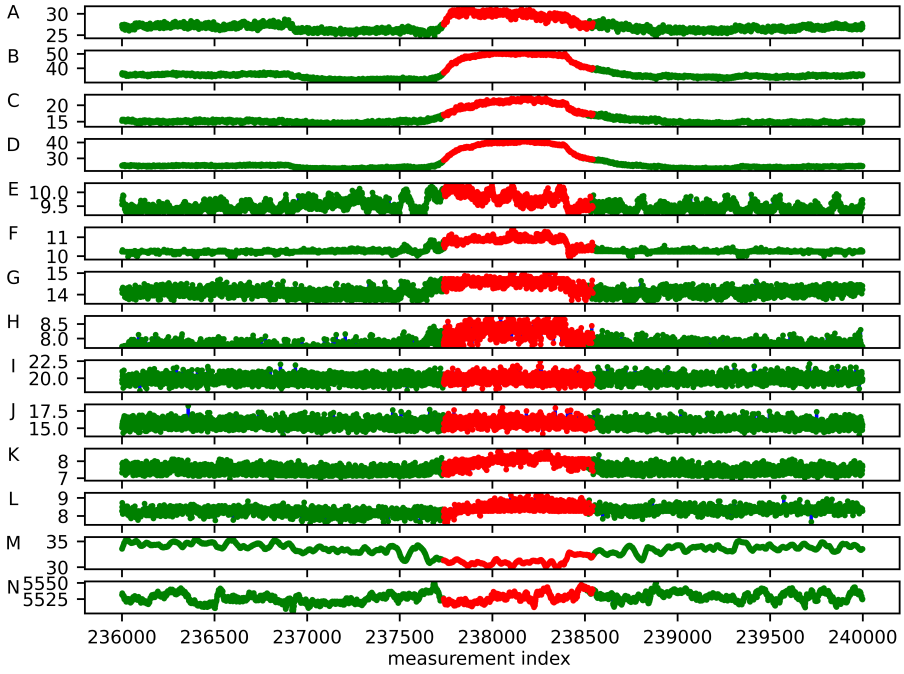
Kvalitu klasifikace opět přibližuje klasifikační report vypracovaný z výsledků klasfikace a značení experta.

	Precision score	Recall score	F1-score	Počet vzorků
Normální vzorky	0.8663	1.0000	0.9284	2787
Anomální vzorky	1.0000	0.6455	0.7846	1213
Vážený průměr	0.9069	0.8925	0.8848	4000
Accuracy			0.8925	4000

Tabulka 4.3: Klasifikační report

Všechny vzorky označené jako anomálie byly skutečně anomáliemi. Oproti tomu 36 % anomálií bylo chybně klasifikováno jako normální obrazy. Tato hodnota je srovnatelná s výsledkem GMM a opět je možná ji částečně přičíst nedokonalým označením dat expertem. Vliv na tento výsledek má i použité hodnotící kritérium Recall score, jež se zaměřuje na eliminaci falešně pozitivních výsledků. To lze vyčíst i z klasifikačního reportu, podle kterého bylo 100 % obrazů správně klasifikováno jako normální. Na tento fakt poukazuje už matice záměn.

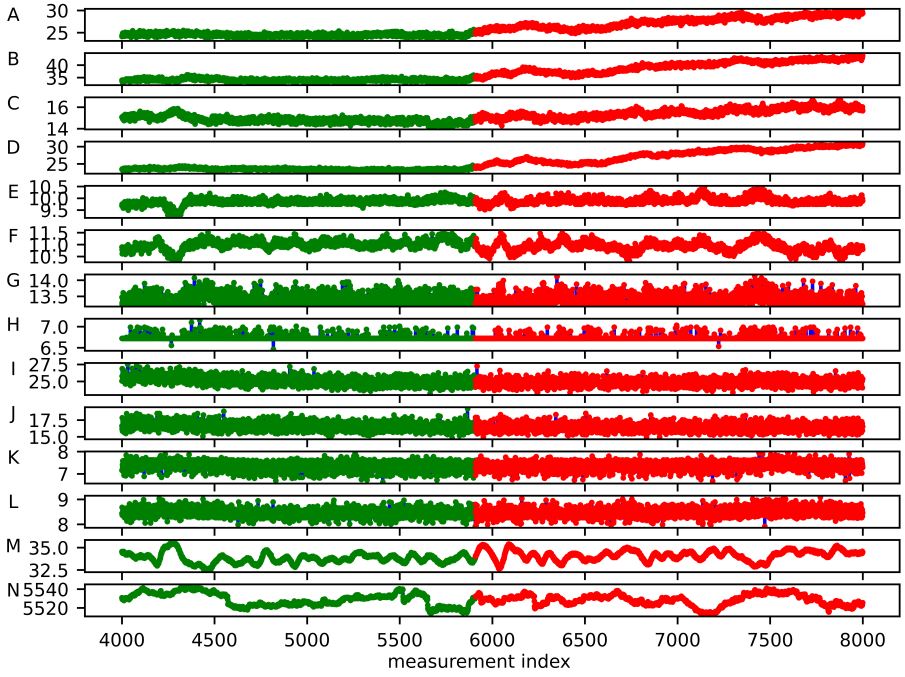
Poslední graf zobrazuje úsek s anomální událostí z kapitoly 4.1.1. Barevné označení vzorků se nyní neřídí rozhodnutím experta ale výstupem modulu detekce anomálií. Detekovaná anomálie je přesně označena. Se značením experta se opět rozchází pouze na samých okrajích události.



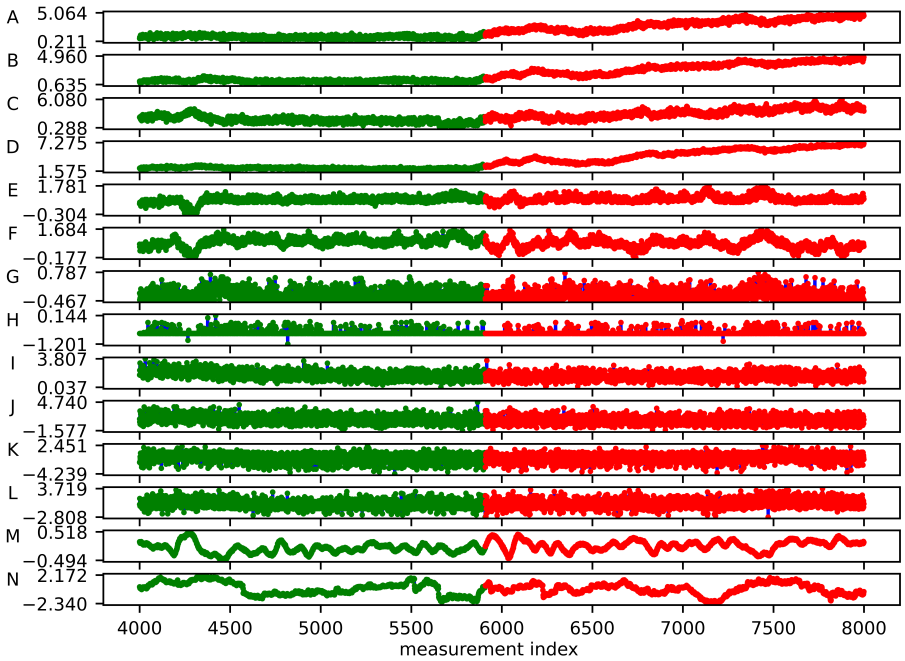
Obrázek 4.12: Testovací množina označená klasifikátorem

4.1.4 Událost 2 - popis

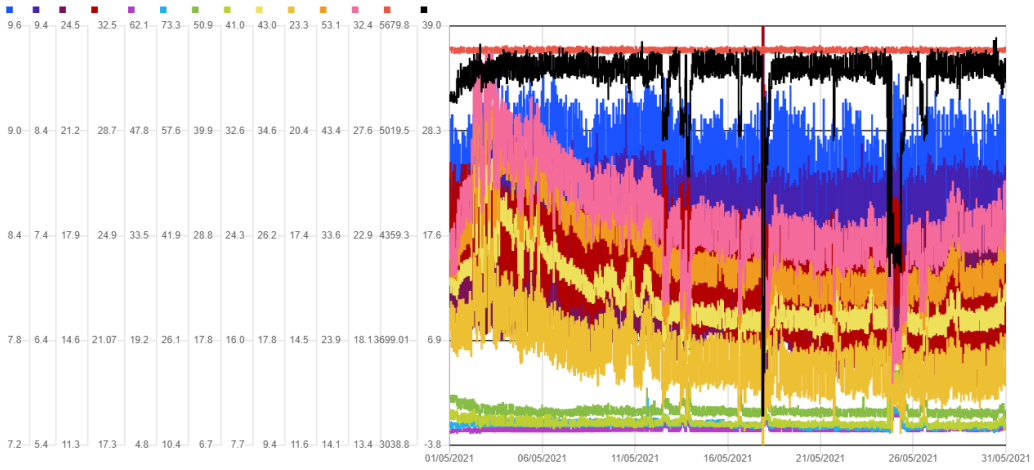
Druhá událost byla expertem identifikována v rozsahu indexů 5 915 až 68 873. V následujícím grafu je tento úsek zachycen v kontextu okolních hodnot. Zelenou barvou jsou označeny vzorky označené expertem za normální, červenou pak vzorky identifikované expertem jako anomální. Počátek této sady, kde se zároveň i začíná projevovat označená událost, zahrnuje konec nájezdu zařízení na jmenovitý výkon. Tento jev nebyl v sadě dat události 1 zahrnutý. Na Obr. 4.15 je zobrazení celé datové sady v grafickém rozhraní systému RMS.



Obrázek 4.13: Testovací množina pro událost 2



Obrázek 4.14: Normalizovaná testovací množina pro událost 2



Název	Popis	Jednotka	Dolní	Horní	Min	Max
00CBP01CE900_XQ01	POWER OF TG	MW	-3.80	38.99	-0.79	37.79
00MAD20CS901	TURBINE SPEED	rpm	3038.75	5679.79	5494.77	5557.36
00MAD10CY001_XQ01	VIBR.FRONT HP ROTOR - X	µm	13.40	32.36	16.21	31.28
00MAD10CY002_XQ01	VIBR.FRONT HP ROTOR - Y	µm	14.10	53.14	18.10	46.49
00MAD20CY001_XQ01	VIBR.REAR HP ROTOR - X	µm	11.58	23.31	10.87	17.40
00MAD20CY002_XQ01	VIBR.REAR HP ROTOR - Y	µm	9.37	42.99	13.83	32.31
00MAK01CY001_XQ01	VIBR.TR.DRIV.ROT.- HS TUR SIDE-X	µm p-p	7.66	40.97	8.82	18.05
00MAK01CY002_XQ01	VIBR.TR.DRIV.ROT.- HS TUR SIDE-Y	µm p-p	6.72	50.91	9.25	19.50
00MAK01CY007_XQ01	VIBR.TR.DRIV.ROT.- LS GEN SIDE-X	µm p-p	10.43	73.29	12.28	27.35
00MAK01CY008_XQ01	VIBR.TR.DRIV.ROT.- LS GEN SIDE-Y	µm p-p	4.83	62.15	6.46	17.24
00MKD10CY001_XQ01	VIBR.FRONT GEN. ROTOR - X	µm	17.26	32.52	19.22	33.97
00MKD10CY002_XQ01	VIBR.FRONT GEN. ROTOR - Y	µm	11.29	24.54	13.33	25.09
00MKD20CY001_XQ01	VIBR.REAR GEN. ROTOR - X	µm	5.37	9.41	6.23	8.71
00MKD20CY002_XQ01	VIBR.REAR GEN. ROTOR - Y	µm	7.18	9.60	7.68	9.58

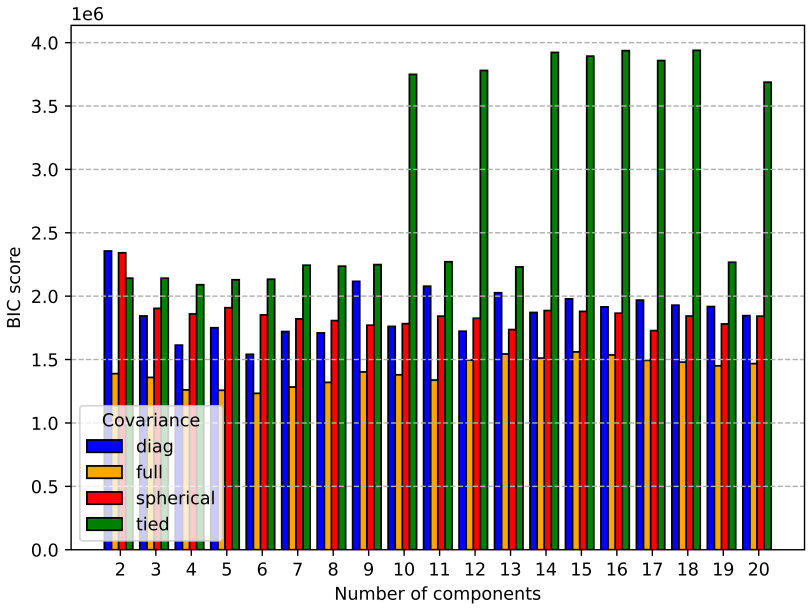
Obrázek 4.15: Celá datová sada dostupná pro událost 2

4.1.5 Událost 2 - GMM

Trénování klasifikátoru.

Trénování klasifikátoru proběhlo na datech od indexu 100 000 až 350 000, tedy 250 000 vzorcích, což tvoří přibližně 48,6 % dostupných dat. Velikost trénovací množiny 220 000 až 250 000 vzorků se ukázala jako dobrým kompromisem mezi časem potřebným k výběru parametrů a přesností modelu. Použitý úsek dat byl opět vybrán tak, aby měla trénovací množina určitý odstup od identifikované anomálie pro případ, že její rozsah nebyl expertem identifikován správně. Set byl opět před startem procesu učení normalizován. Algoritmus exhaustive grid search prohledával stejnou strukturu hodnot jako v předchozím případě, tedy počet komponent směsi $n \in \langle 2; 20 \rangle$ a všechny čtyři podporované typy kovariance komponent. Jako hodnotící kritérium bylo znovu vybráno Bayesovo informační kritérium. Algoritmus vybral jako nejlepší kombinaci počet komponent $n = 6$ a typ kovariance komponent

full. S touto kombinací parametrů dosahuje GMM skóre BIC = 1 233 399,68. Porovnání BIC skóre pro všechny kandidáty zobrazuje Obr. 4.15, vývoj BIC skóre pro typ kovariance nejlepšího kandidáta je detailně zobrazen na Obr. 4.16. Pro srovnání je v grafu opět vyneseno i AIC skóre. K odhadnutí parametrů komponent směsi bylo potřeba 28 iterací EM-algoritmu. Celková doba běhu algoritmu exhaustive grid search byla 204,9 minut.



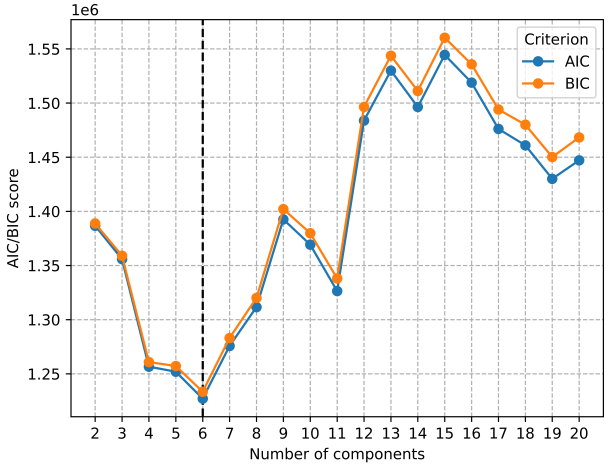
Obrázek 4.16: Srovnání BIC skóre všech kandidátů

V porovnání všech kandidátů je patrné, že směsi s typy kovariance komponent tied nebyly schopné dobře vyjádřit tvar dat napříč počtem komponent. Jejich zvyšující se BIC skóre s rostoucím počtem komponent je možné vysvětlit jak nevhodnými parametry modelu, tak možnými problémy s konvergencí EM-algoritmu či přílišné komplexnosti modelu, která vedla ke ztrátě schopnosti generalizace.

Vývoj BIC skóre s rostoucím počtem komponent pro typ kovariance komponent full také ukazuje náhlý nárůst hodnoty skóre při vyšším počtu komponent. Zvyšování počtu komponent zřejmě nejen nepřinášelo dodatečný přínos, ale naopak snižovalo věrohodnost modelu a zvyšovalo jeho komplexitu. Pro vysoký počet komponent již mohlo docházet k přetrénování nebo nebyl EM-algoritmus schopen konvergovat k lokálnímu optimu v nastaveném limitu 100 iterací.

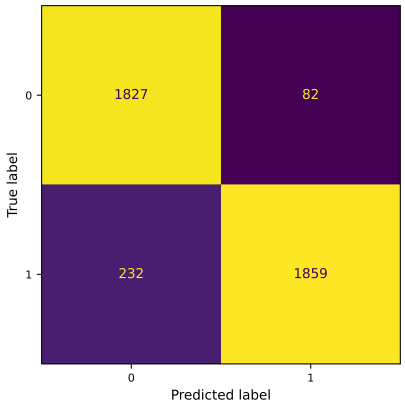
Detekce anomálií.

Z důvodu délky této události proběhla detekce anomálií pouze na intervalu začínajícím před začátkem události označené expertem na množině o 4 000 vzorcích tak, aby interval zasahoval do vyznačené události. Testování rozumného výřezu



Obrázek 4.17: Vývoj BIC skóre pro typ kovariance full

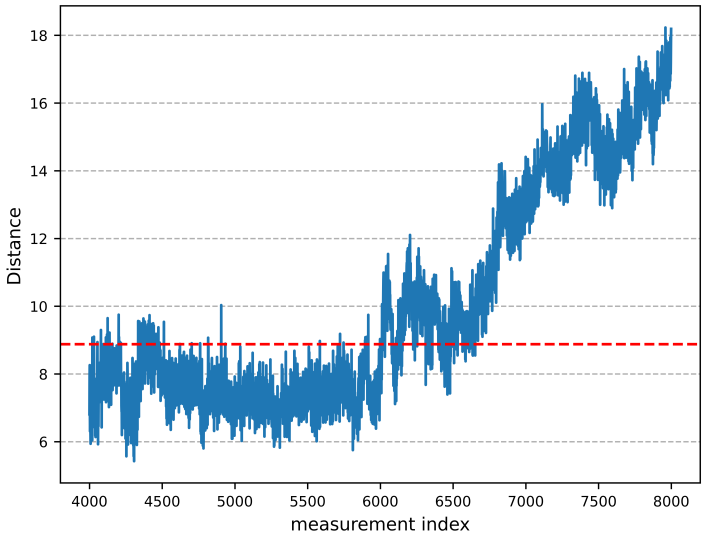
dat, který by pokryl celou událost, by trvalo odhadem 17 hodin. Celkem tedy bylo opět zpracováno 4 000 vzorků dat. Zpracování všech vzorků trvalo modulu detekce anomálií 57,6 minut, tedy přibližně 0,86 sekund na jeden vzorek. Ze 4000 zpracovávaných vzorku bylo špatně klasifikováno 314 neboli 7,85 %, z toho 82 výsledků bylo falešně pozitivní. Výsledky klasifikace shrnuje matice záměn:



Obrázek 4.18: Matice záměn

Následující graf vykresluje Mahalanobisovy vzdálenosti jednotlivých vzorků. Červená přerušovaná čára označuje prahovou hodnotu 8,88 stanovenou na základě pozorovaných vzdáleností normálních dat. Prahová hodnota byla oproti první události zvýšena, neboť přítomnost jevu nájezdu zařízení na jmenovitý výkon nebyla součástí trénovací množiny a klasifikátor tak s původní prahovou hodnotou produkoval příliš falešně pozitivních výsledků (v řádu vysokých stovek). Celá datová sada zahrnovala normální projevy v předchozí sadě nepozorované. Jako řešení by

se nabízelo natrénovat klasifikátor na delším úseku, který bude zahrnovat rozličné normální projevy zařízení. Vhodná datová sada však nebyl k dispozici.



Obrázek 4.19: Mahalanobisovy vzdálenosti vzorků testovací množiny

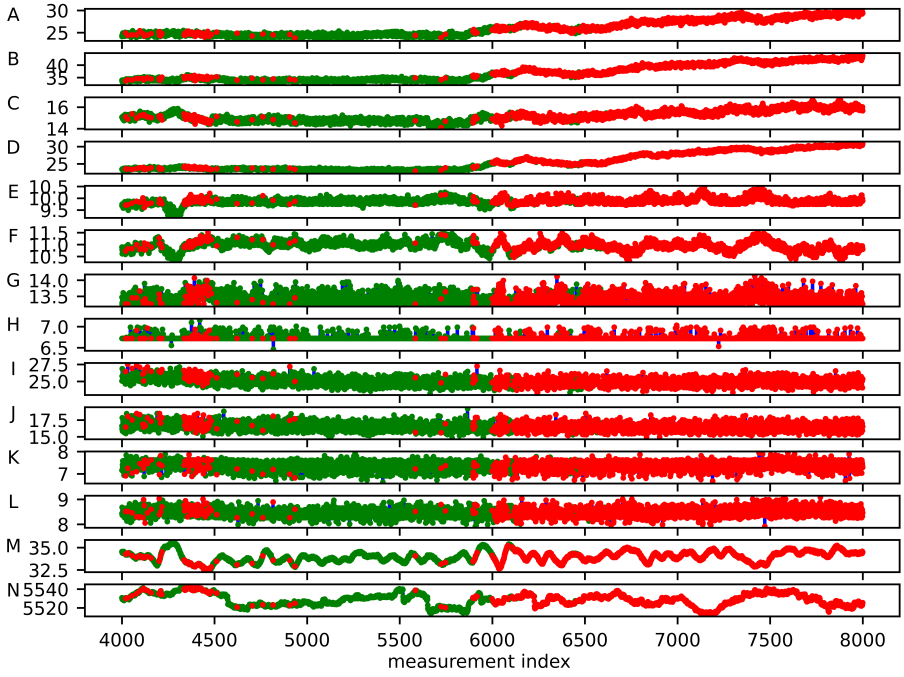
Začátek události je v grafu Mahalanobisových vzdáleností dobře rozpoznatelná, a to i přes úseky, kde se vzdálenost vracela k normálním hodnotám. To mohlo být způsobeno postupným projevem anomálního stavu v jeho začátku. S dalším rozvojem anomálie vzdálenost opět výrazně roste, což indikuje rostoucí anomalitu obrazů.

Celkově shrnuje kvalitu detekce anomálií klasifikační report:

	Precision score	Recall score	F1-score	Počet vzorků
Normální vzorky	0.8873	0.9570	0.9209	1909
Anomální vzorky	0.9578	0.8890	0.9221	2091
Vážený průměr	0.9241	0.9215	0.9215	4000
Accuracy			0.9215	4000

Tabulka 4.4: Klasifikační report

I přes zvýšení prahové hodnoty ukazuje Recall score, že 4,3 % normálních obrazů byly mylně klasifikovány jako anomálie. Oproti tomu pouze 11,1 % anomálních obrazů nebylo odhaleno. I přesto, že bylo nutné zvýšit prahovou hodnotu, stále bylo správně identifikováno 95,8 % anomálií. Celková úspěšnost klasifikace dosáhla 92 %. Poslední graf znovu zobrazuje zkoumaný úsek dat, tentokrát obarvený na základě výstupu klasifikátoru – zelené body byly označeny jako normální, červené jako anomální.



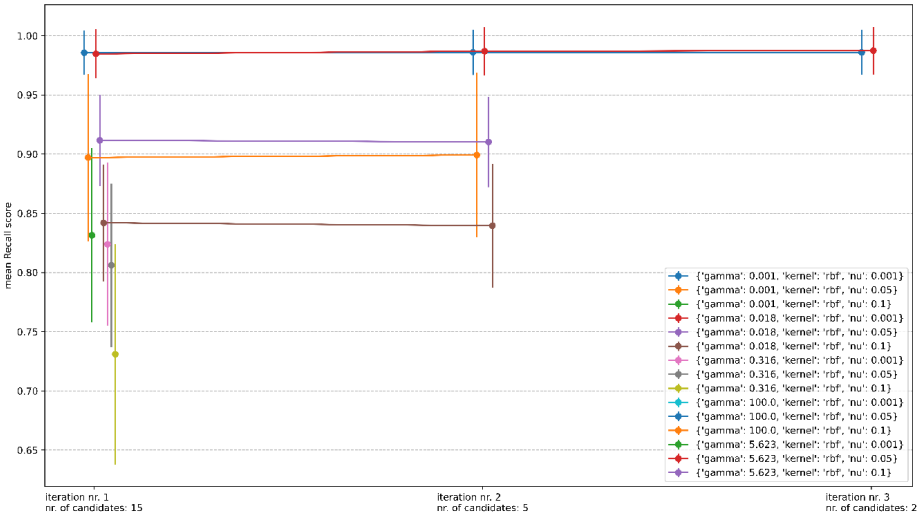
Obrázek 4.20: Testovací množina označená klasifikátorem

Začátek událost nalezené expertem byl klasifikátorem identifikován správně. Na počátku zkoumaného intervalu jsou patrné skupiny falešně pozitivních klasifikací, které jsou důsledkem přítomnosti dat, jež jsou ve skutečnosti normální, ale jejichž odlišný projev nebyl pozorován v trénovací množině. Klasifikátor je proto nebyl schopen správně identifikovat.

4.1.6 Událost 2 - SVM

Trénování klasifikátoru.

Učení probíhalo na stejné trénovací množině 250 000 vzorků, data byla před tréningem opět normalizována. Algoritmus successive halving procházel strukturou obsahují hodnoty $\nu \in \{0,001; 0,01; 0,1\}$, $\gamma \epsilon \{0,001; 0,018; 0,316; 5,623; 100\}$ pro zvolený kernel RBF. Jako hodnotící kritérium bylo použito Recall score. Algoritmus vyhodnotil jako nejlepší kombinaci parametrů $\nu = 0,001;\ \gamma = 0,018$. S touto kombinací parametrů dosáhl klasifikátor Recall score 0,9874. K dosažení výsledku bylo potřeba tří iterací algoritmu successive halving s celkovou dobou trvání 123,7 minut. Přehled vývoje průměrného skóre všech kandidátů včetně standardní odchylky v průběhu iterací algoritmu nabízí Obr. 4.20. Kombinace parametrů, jejichž skóre bylo rovno nebo blízké nule, jsou v grafu ignorovány.

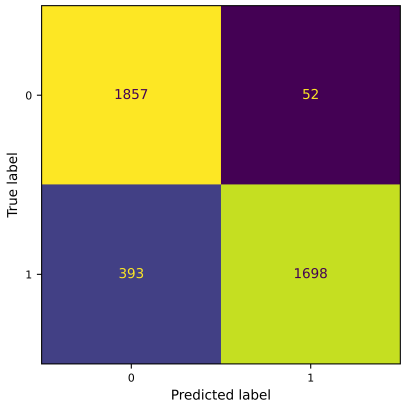


Obrázek 4.21: Vývoj skóre kandidátů v iterací successive halving

Detekce anomálií.

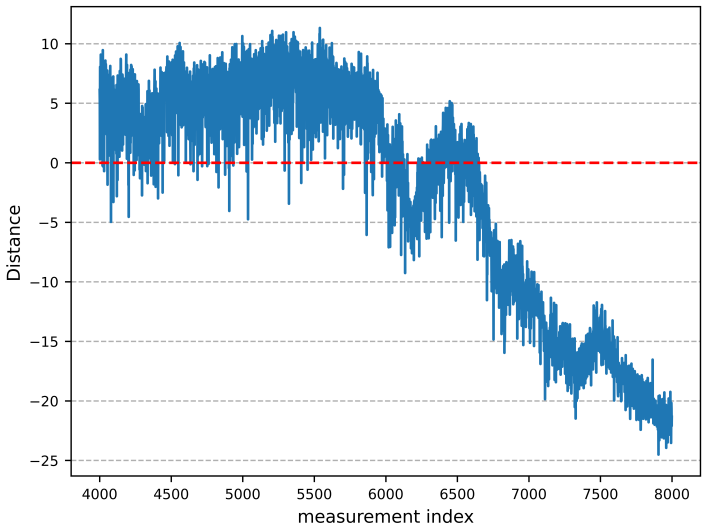
Detekce probíhala na stejném intervalu dat jako v případě GMM, tedy 4 000 až 8 000. Prahová hodnota byla ponechána ve výchozím nastavení 0,00. Při rozhodování o zařazení obrazu tedy nebude k vypočtené vzdálenosti od dělící nadroviny přičítán žádný ofset.

Zpracování dat trvalo modulu detekce anomálií 70,4 minut. Zvýšení doby zpracování vzorku na 1 sekundu mohlo zapříčinit vytížení systému procesy běžícími v pozadí. Doba zpracování je však stále vzhledem k periodě měření dat akceptovatelná. Z celkového počtu 4 000 vzorků bylo špatně klasifikováno 445 vzorků, tedy 11,12 %. Z tohoto počtu bylo 52 výsledků falešně pozitivní. Celkovou statistiku poskytuje matice záměn:



Obrázek 4.22: Matice záměn

Vypočítané vzdálenosti všech 4 000 vzorků jsou vykresleny na následujícím grafu, kde červená přerušovaná čára zvýrazňuje nulovou vzdálenost od dělící nadroviny. Událost je opět snadno rozpoznatelná, vzdálenost anomálních vzorků je opět výrazně odlišná od vzdálenosti normálních vzorků.



Obrázek 4.23: Vzdálenosti vzorků testovací množiny od dělící nadroviny

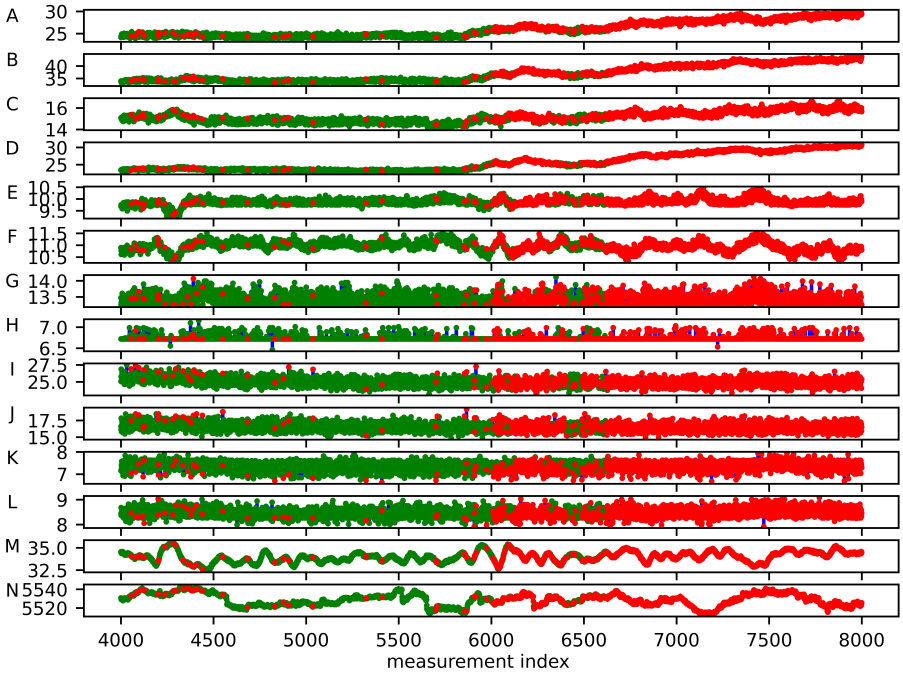
Z grafu vzdáleností vzorků od dělící roviny je patrné, že i malý ofset ve výpočtu vzdálenosti nastavením nenulové prahové hodnoty by výsledek výrazně zlepšilo. Malá prahová hodnota by navíc neohrozila úspěšné detekování skutečných anomálií. Jak plyne z klasifikačního reportu, i bez změny prahové hodnoty však bylo pouze 2,7 % normálních obrazů nesprávně klasifikováno jako anomální.

	Precision score	Recall score	F1-score	Počet vzorků
Normální vzorky	0.8253	0.9728	0.8930	1909
Anomální vzorky	0.9703	0.8121	0.8841	2091
Vážený průměr	0.9011	0.8888	0.8884	4000
Accuracy			0.8888	4000

Tabulka 4.5: Klasifikační report

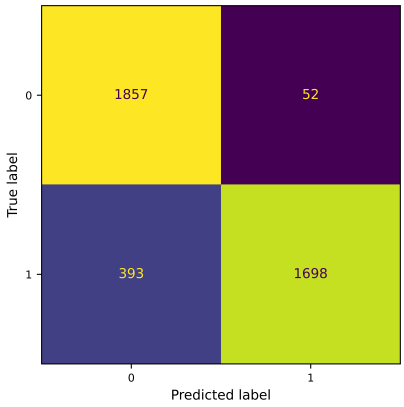
Vlivem vyššího počtu falešně pozitivních výsledků oproti předchozím experimentům však celková přesnost klasifikace klesla na 89 %. Procento úspěšně detekovaných anomálií stále zůstává s 97 % vysoké. V následujícím detailu zkoumaného úseku jsou opět zeleně označeny vzorky klasifikované jako normální a červeně vzorky klasifikované jako anomální. Oblasti výskytu falešně pozitivních výsledků

se shodují s výstupem GMM. Detekce anomálií pomocí OCSVM s parametry vybranými na základě Recall score se ukázala jako robustnější než GMM. I bez změny prahové hodnoty produkovala méně falešně pozitivních výsledků.



Obrázek 4.24: Testovací množina označená klasifikátorem

Po nastavení prahové hodnoty na 1.00 došlo k výraznému snížení počtu falešně pozitivních výsledků. Počet normálních obrazů nesprávně klasifikovaných jako anomálie klesl na 1,5 %, počet neodhalených anomálií stoupl z 19 % na 32,5 %.

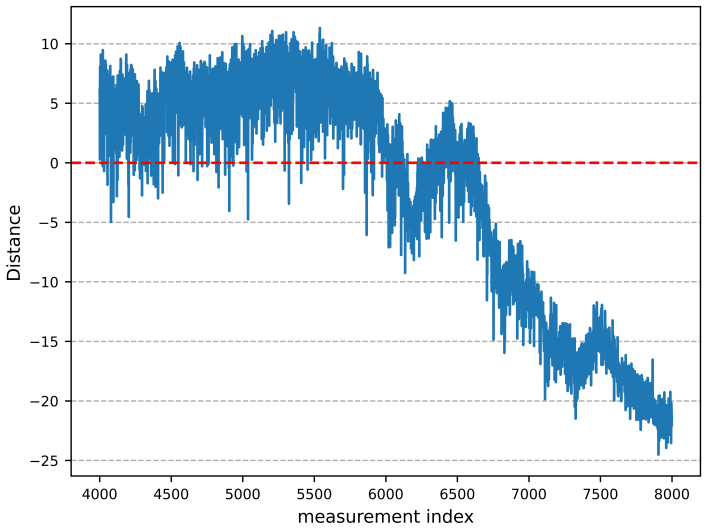


Obrázek 4.25: Matice záměn

	Precision score	Recall score	F1-score	Počet vzorků
Normální vzorky	0.8000	0.9848	0.8828	1909
Anomální vzorky	0.9824	0.7752	0.8666	2091
Vážený průměr	0.8954	0.8752	0.8744	4000
Accuracy			0.8752	4000

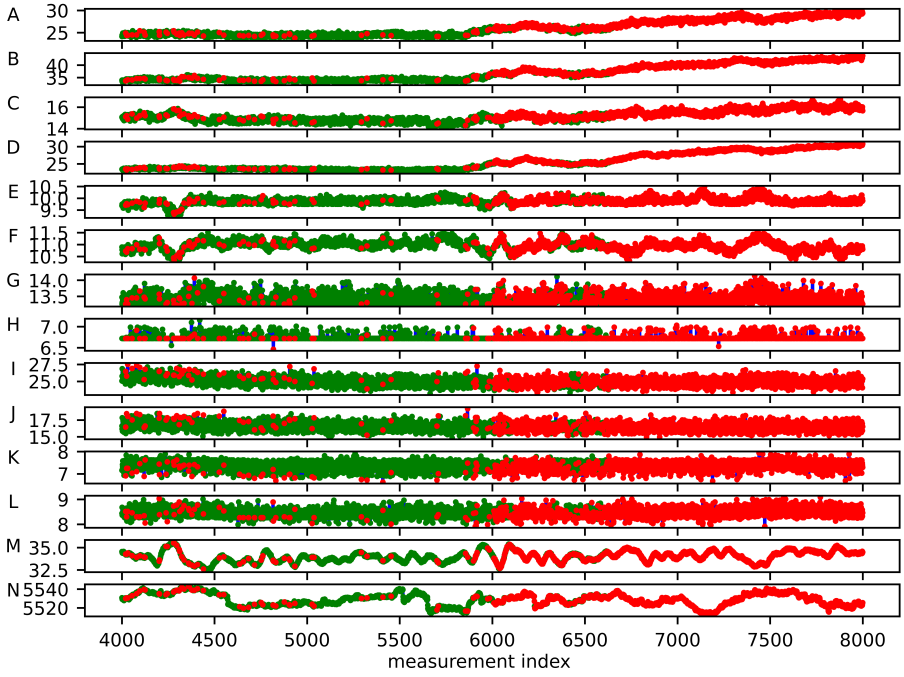
Tabulka 4.6: Klasifikační report

Na grafu vzdáleností vzorků i na znázorněném zkoumaném úseku na Obr. 4.27 je však patrné, že podstatnou část začátku události se i přesto podařilo úspěšně identifikovat. Procento neodhalených anomálií by s dalšími vzorky dosahujícími obdobně vysoké anomality jako na konci zkoumaného úseku klesalo.



Obrázek 4.26: Vzdálenosti vzorků testovací množiny od dělící nadroviny

Na vzorcích testovací množiny označených podle výstupu klasifikátoru jsou patrné shluky falešně pozitivních výsledků na začátku testovací množiny. Jedná se o zmíněné normální vzorky, patřící ke stavu zařízení, jež nebyl součástí trénovací množiny.



Obrázek 4.27: Testovací množina označená klasifikátorem

4.2 Detekce anomálií na turbíně TG 55MW

Obě sady dat exportované z databáze RMS pro toto zařízení pocházejí z roku 2022. Export události 1 obsahuje 623 482 vzorků, export druhé události obsahuje 1 004 913 vzorků měření. I v případě tohoto zařízení patří exportovaná měření do signálových skupin Provozní veličiny a Relativní rotorové vibrace a detekce anomálií bude tedy opět probíhat na signálech domény Relativních rotorových vibrací. Pro referenci budou do sady znovu přidány shodné signály z domény Provozních veličin jako předešlého zařízení. Následuje tabulka použitých signálů a jejich značení.

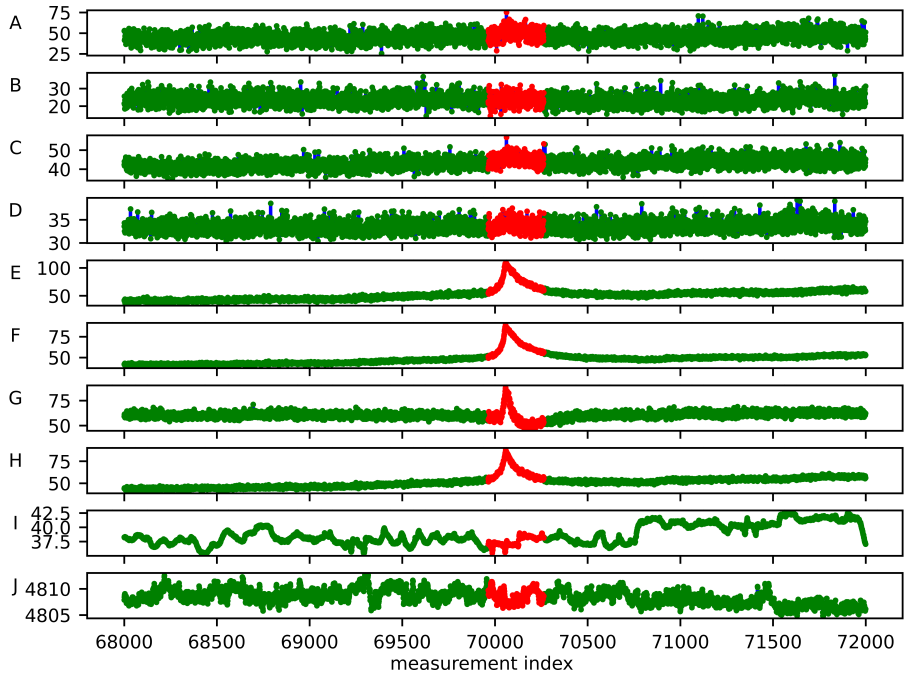
Index	Jméno	Popis signálu	Označení
1	VIBR.FRONT HP ROTOR - X	Vibrace rotoru, osa X, vpředu, vysokotlaká část	A
2	VIBR.FRONT HP ROTOR - Y	Vibrace rotoru, osa Y, vpředu, vysokotlaká část	B
3	VIBR.REAR HP ROTOR - X	Vibrace rotoru, osa X, vzadu, vysokotlaká část	C
4	VIBR.REAR HP ROTOR - Y	Vibrace rotoru, osa Y, vzadu, vysokotlaká část	D

5	VIBR.FRONT GEN. ROTOR - X	Vibrace rotoru, osa X, vpředu, generátor	E
6	VIBR.FRONT GEN. ROTOR - Y	Vibrace rotoru, osa Y, vpředu, generátor	F
7	VIBR.REAR GEN. ROTOR - X	Vibrace rotoru, osa X, vzadu, generátor	G
8	VIBR.REAR GEN. ROTOR - Y	Vibrace rotoru, osa Y, vzadu, generátor	H
9	POWER OF TG	Výkon TG, aktuální	I
14	SPEED TURBINE	Otáčky, turbína	J

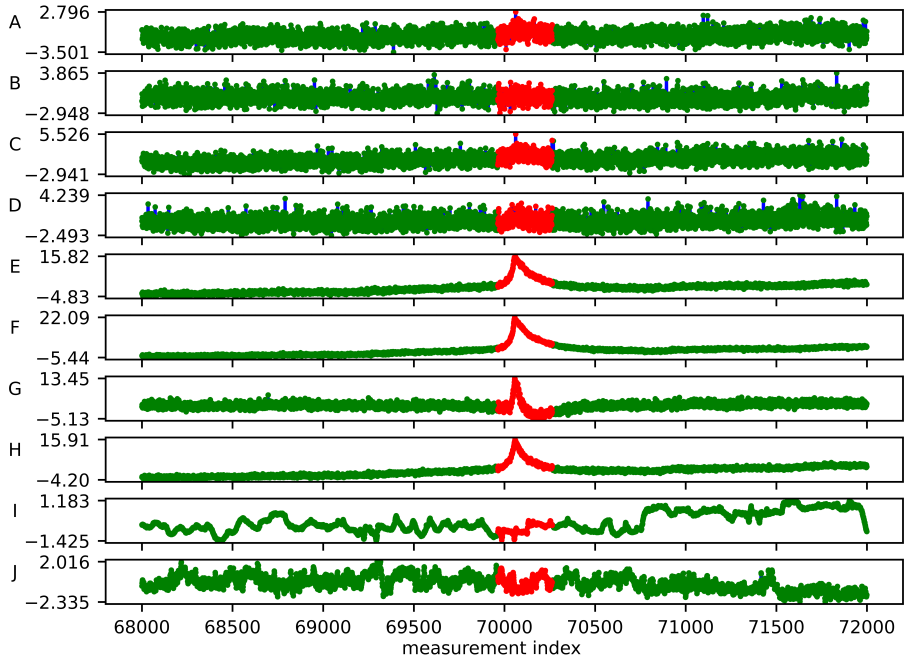
Tabulka 4.7: Přehled sledovaných signálů

4.2.1 Událost 1 - popis

První z událostí identifikovaných expertem na tomto zařízení se nachází v rozsahu indexů 69 970 až 70 269 na první datové sadě o velikosti 623 482. Níže jsou opět vyneseny hodnoty signálů společné domény s vyznačeným úsekem celé události tak, jak ji označil expert. Následuje pak opět vyobrazení celého exportovaného úseku v grafickém rozhraní RMS. Na něm je patrný výrazný krátký výkyv otáček turbíny přibližně ve třetině datové sady.



Obrázek 4.28: Testovací množina pro událost 1



Obrázek 4.29: Normalizovaná testovací množina pro událost 1

	ID	Název	Popis	Jednotka	Dolní	Horní	Min	Max
	1	75MAD10CS901_XQ01	SPEED TURBINE	rpm	-236.08	5106.66	6.78	4863.81
	2	75MKA10CE001_XQ01	POWER OF TG	MW	-2.44	51.56	0.01	49.10
	3	75MKD10CY001_XQ01	VIBR.FRONT GEN.ROTOR - X	µm	-4.66	115.40	0.79	109.94
	4	75MKD10CY002_XQ01	VIBR.FRONT GEN.ROTOR - Y	µm	-3.73	93.53	0.69	89.11
	5	75MKD20CY001_XQ01	VIBR.REAR GEN.ROTOR - X	µm	-3.53	92.60	0.84	88.23
	6	75MKD20CY002_XQ01	VIBR.REAR GEN.ROTOR - Y	µm	-3.72	92.16	0.64	87.80
	7	75MAD10CY001_XQ01	VIBR.FRONT HP ROTOR - X	µm	-3.29	96.63	1.25	92.09
	8	75MAD10CY002_XQ01	VIBR.FRONT HP ROTOR - Y	µm	-0.93	46.08	1.21	43.95
	9	75MAD20CY001_XQ01	VIBR.REAR HP ROTOR - X	µm	-2.19	60.08	0.64	57.25
	10	75MAD20CY002_XQ01	VIBR.REAR HP ROTOR - Y	µm	-1.19	43.36	0.84	41.34

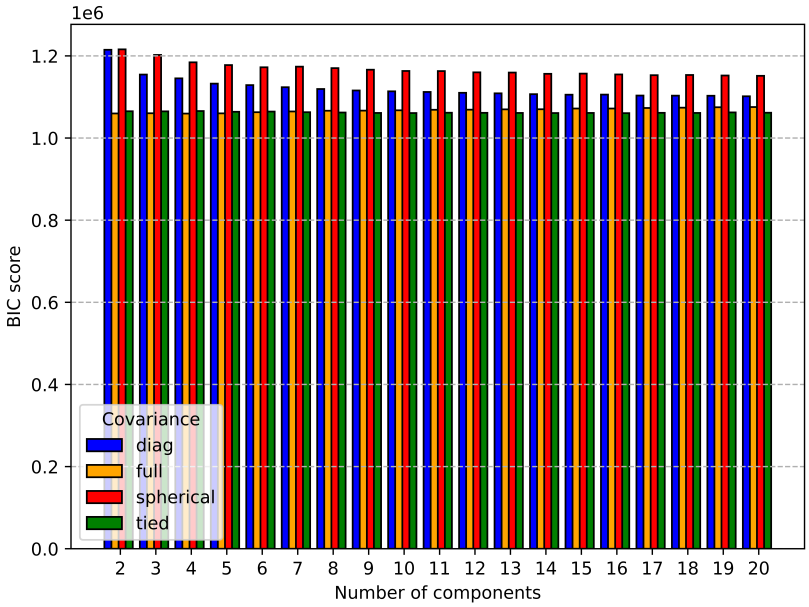
Obrázek 4.30: Celá datová sada dostupná pro událost 1

4.2.2 Událost 1 - GMM

Trénování klasifikátoru.

Původně byl klasifikátor trénován na intervalu dat od indexu 90 000 do indexu 340 000, tedy 250 000 vzorků, což představuje 40 % dostupných dat. S touto trénovací množinou však nebyl algoritmus exhaustive grid search schopný nalézt žádnou kombinaci parametrů, pro níž by EM-algoritmus nedivergoval. Tento jev byl nejpravděpodobněji způsobený extrémními hodnotami v trénovací množině, konkrétně krátkým skokovým poklesem otáček turbíny zmíněným v popisu události. Optimalizační algoritmus se pokoušel přizpůsobit těmto obrazům s extrémními složkami na úkor ostatních dat a nebyl schopen najít v nastaveném limitu iterací konvergující řešení.

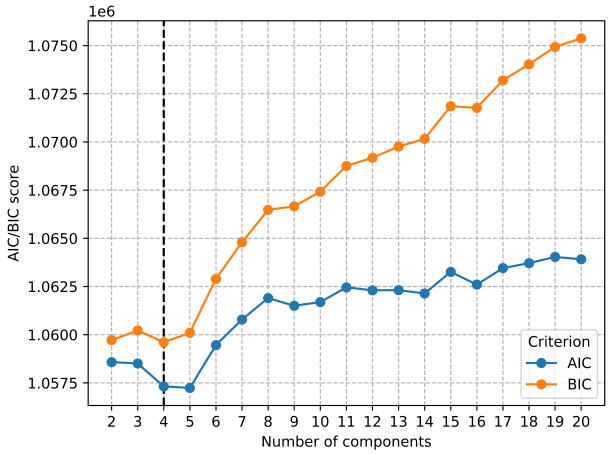
Trénovací množina byla proto změněna na interval dat mezi indexy 400 000 a 620 000, tedy pouze 35 % dostupných dat. Struktura hodnot parametrů použitá pro algoritmus exhaustive grid search byla stejná jako u předchozích událostí: počet komponent směsi $n \in \langle 2; 20 \rangle$ a všechny čtyři podporované typy kovariance komponent. Kandidáti byli opět hodnoceni Bayesovým informačním kritériem.



Obrázek 4.31: Srovnání BIC skóre všech kandidátů

Nejnižšího skóre BIC = 1 059 602,10 dosáhla směs s počtem komponent $n = 4$ a typem kovariance komponent full. Proces hledání nejvhodnějšího kandidáta trval 153,1 minut a k nalezení optimálních parametrů směsi bylo třeba 18 iterací EM-algoritmu. Výrazně kratší doba učení ve srovnání s předchozími událostmi ukazuje vliv počtu zkoumaných signálů společné domény, která je na tomto zařízení o čtyři

signály menší, a velikosti trénovací množiny na dobu běhu algoritmu exhaustive grid search. Porovnání BIC skóre všech kandidátů a vývoj BIC skóre pro vybraný typ kovariance ukazují grafy na Obr. 4.32 a Obr. 4.33.

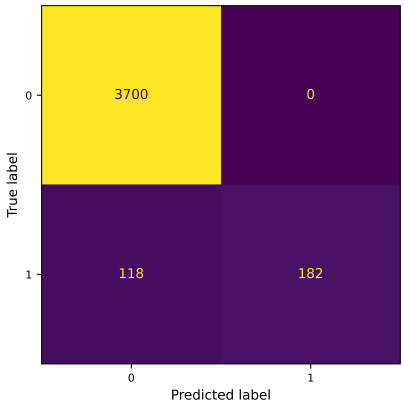


Obrázek 4.32: Vývoj BIC skóre pro typ kovariance full

Všechny typy kovariance komponent dosahovaly v tomto případě dobrých výsledků, avšak nejlepší volbou byla opět kovariance komponent typu full. S rostoucím počtem komponent směsi hodnota BIC mírně rostla což svědčí o dosažení dostatečné komplexity modelu pro popis dat trénovací množiny – přidání další komponenty již nepřináší zlepšení věrohodnosti modelu či přímo zhoršuje schopnost modelu dobře popisovat zkoumaná data.

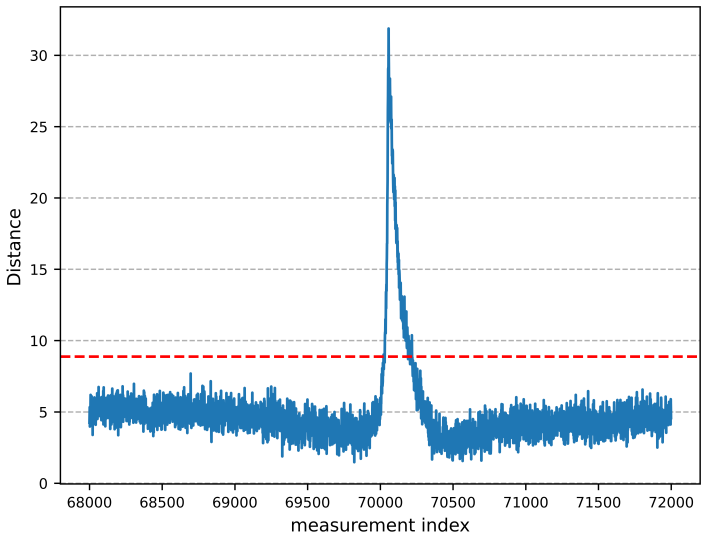
Detekce anomálií.

Detekce anomálií byla spuštěna na intervalu indexů 68 000 až 72 000 v exportu z databáze RMS. Tento interval pokrývá místo výskytu událost s jeho nejbližším okolím. Prahová hodnota klasifikátoru byla nastavena na hodnotu 8,88. Zpracování vzorků testovací množiny trvalo 56,3 minuty. Počet sledovaných signálů v množině signálů společné domény tedy zřejmě nemá vliv na rychlost klasifikace. Ze 4 000 zkoumaných vzorků bylo nesprávně klasifikováno pouze 2,95 % přitom žádný normální obraz nebyl klasifikován nesprávně.



Obrázek 4.33: Matice záměn

V grafu určených Mahalanobisových vzdáleností na Obr. 4.34 vzorků je hledaná událost velmi dobře rozpoznatelná. Strmý nárůst vzdálenosti obrazů indikuje vysokou míru anomality této události.



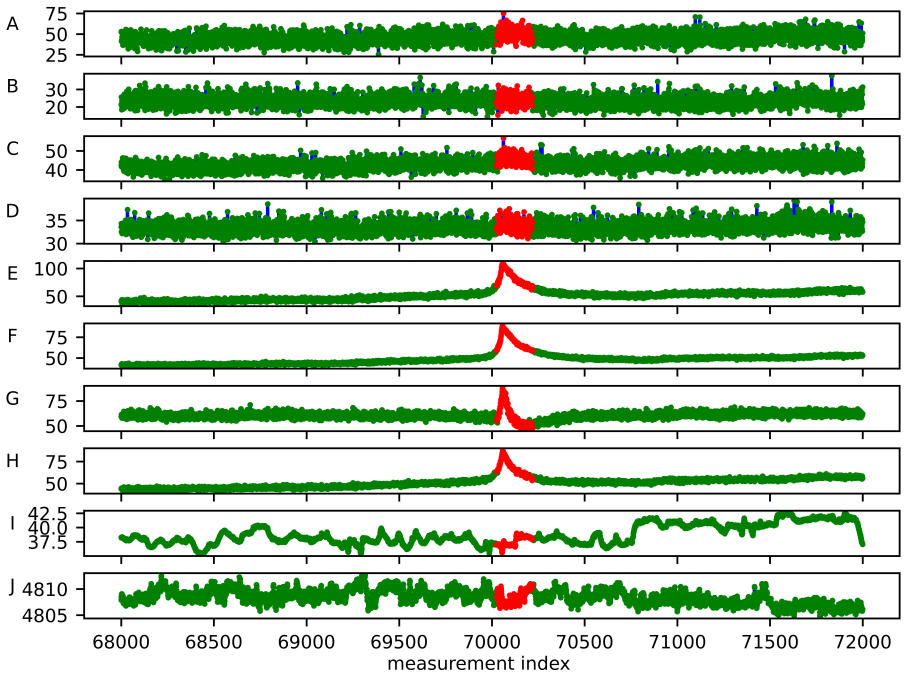
Obrázek 4.34: Mahalanobisovy vzdálenosti vzorků testovací množiny

O dobrém počínání klasifikátoru svědčí i klasifikační report. Celková přesnost provedené klasifikace byla 97 %. Všechny obrazy klasifikované jako anomálie byly skutečně podle experta anomáliemi a všechny normální obrazy byly identifikovány správně. Schopnost plně popsat obrazy nenáležící do cílové třídy je s hodnotou 0,6067 poměrně nízká, avšak na úspěšné odhalení anomálie neměla vliv. Její velikost lze přičíst zejména velmi krátkému trvání anomálního stavu, kde každý vzorek nesprávně označený expertem má výrazný vliv na vyhodnocení správně klasifikovaných anomálií v množině všech správně klasifikovaných obrazů.

	Precision score	Recall score	F1-score	Počet vzorků
Normální vzorky	0.9691	1.0000	0.9843	3700
Anomální vzorky	1.0000	0.6067	0.7552	300
Vážený průměr	0.9714	0.9705	0.9671	4000
Accuracy			0.9705	4000

Tabulka 4.8: Klasifikační report

Poslední graf opět vyobrazuje testovací množinu se značením vzorků podle výstupu klasifikátoru.



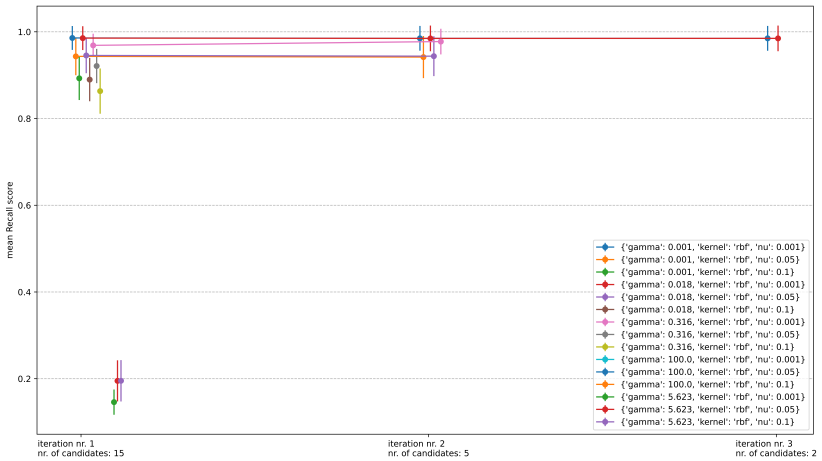
Obrázek 4.35: Testovací množina označená klasifikátorem

4.2.3 Událost 1 - SVM

Trénování klasifikátoru.

Jako trénovací množina byl zvolen úsek dat, na němž hledání nejvhodnějšího modelu GMM selhalo, tedy rozsah indexů 90 000 až 340 000. Klasifikátor OCSVM bylo možné úspěšně natrénovat i za použití této trénovací množiny, která u klasifikátoru s GMM způsobovala problémy s konvergencí. Struktura hodnot pro hledání optimálních parametrů byla ponechána stejná jako u předchozího zařízení: $\nu \in \{0,001; 0,01; 0,1\}$, $\gamma \in \{0,001; 0,018; 0,316; 5,623; 100\}$ pro zvolený kernel RBF.

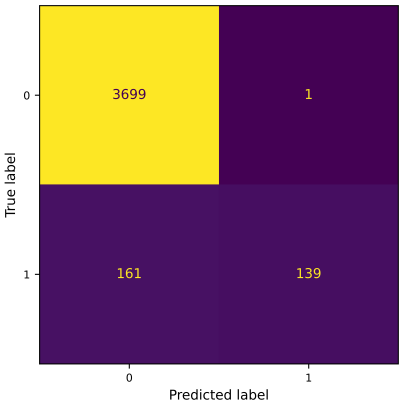
Algoritmus successive halving vybral kombinaci parametrů $\nu = 0,001; \gamma = 0,001$, která dosáhla Recall score 0,9850. Výběr trval 81,3 minut, tedy opět kratší dobu než v případě předchozího zařízení, u nějž obsahuje skupina signálů společné domény relativních vibrací o čtyři měření více. Přehled vývoje Recall score všech kandidátů je opět vyobrazen na Obr. 4.35. Kandidáti, jejichž skóre se blížilo nule, jsou v grafu ignorováni.



Obrázek 4.36: Vývoj skóre kandidátů v iterací successive halving

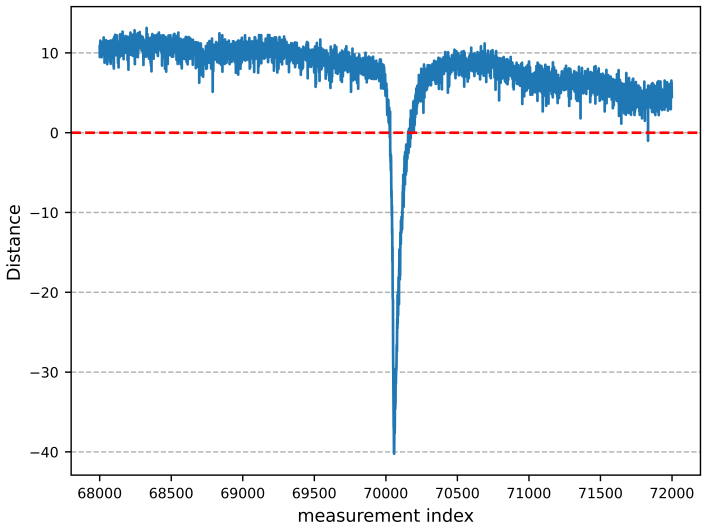
Detekce anomálií.

Testovací množina pro OCSVM byla zvolena shodně s klasifikátorem s GMM. Prahová hodnota byla ponechána na hodnotě 0,00. Zpracování obrazů trvalo 57,2 minuty a nevymyká se tedy nijak z trendu předchozích experimentů. Z celkového počtu 4 000 vzorků bylo chybně klasifikováno 162 vzorků, tedy 4,05 %, z toho pouze jeden výsledek byl falešně pozitivní.



Obrázek 4.37: Matice záměn

Vzdálenosti zkoumaných obrazů od dělící nadroviny zobrazuje následující graf. Událost je v grafu vzdáleností znovu dobře patrná.



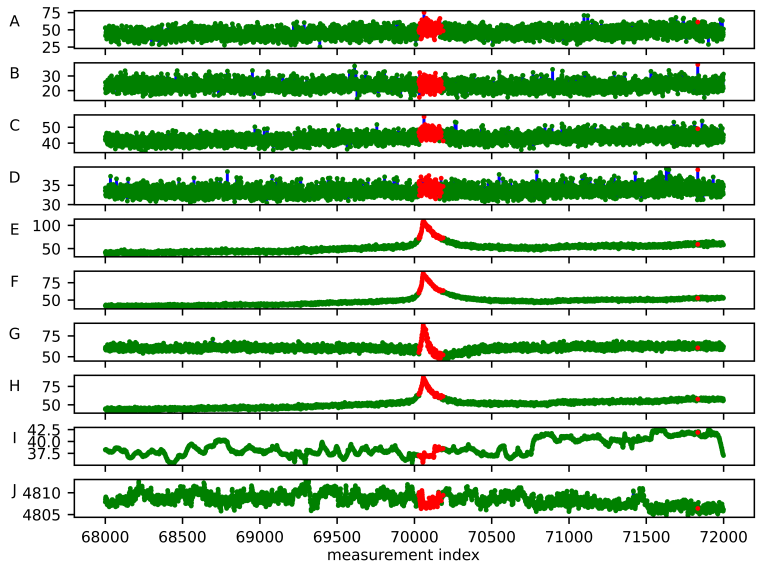
Obrázek 4.38: Vzdálenosti vzorků testovací množiny od dělící nadroviny

Kvalita provedené klasifikace byla opět shrnuta klasifikačním reportem. Výsledky jsou srovnatelné s klasifikátorem s GMM, avšak procento všech správně klasifikovaných anomálií ze všech správně klasifikovaných obrazů je ještě nižší a to 46 %. Tuto hodnotu lze vysvětlit kombinací popsaného jevu, kdy počet vzorků špatně označených expertem v případě anomálního stavu trvajícího relativně krátkou dobu výrazně ovlivní tuto hodnotu a použitým hodnotícím kritériem pro výběr parametrů klasifikátoru, jenž optimalizuje hodnotu Recall score pro normální vzorky.

	Precision score	Recall score	F1-score	Počet vzorků
Normální vzorky	0.9583	0.9997	0.9786	3700
Anomální vzorky	0.9929	0.4633	0.6318	300
Vážený průměr	0.9609	0.9595	0.9526	4000
Accuracy			0.9595	4000

Tabulka 4.9: Klasifikační report

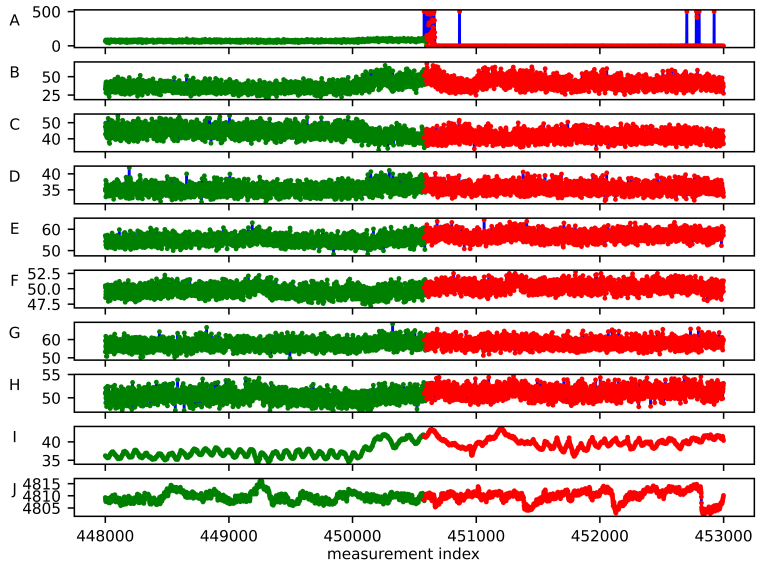
Graf na Obr. 4.39, který zobrazuje testovací množinu označenou výstupem klasifikátoru, je téměř shodný s Obr. 4.29 kde jsou vzorky testovací množiny označené expertem.



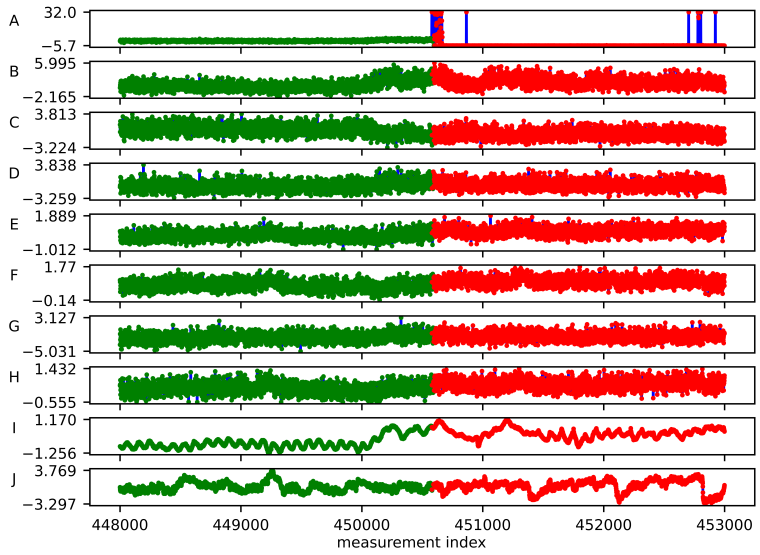
Obrázek 4.39: Testovací množina označená klasifikátorem

4.2.4 Událost 2 - popis

Druhou událost identifikovat expert na intervalu indexů 450 585 až 1 004 913. Anomální stav tedy zabírá více než polovinu dostupného exportu z databáze RMS. Celý export zobrazuje Obr. 4.42, na Obr. 4.39 a Obr. 4.41 je opět testovací množina před a po normalizaci pomocí objektu StandardScaler. Barevné značení odpovídá informaci experta.



Obrázek 4.40: Testovací množina pro událost 1



Obrázek 4.41: Normalizovaná testovací množina pro událost 1

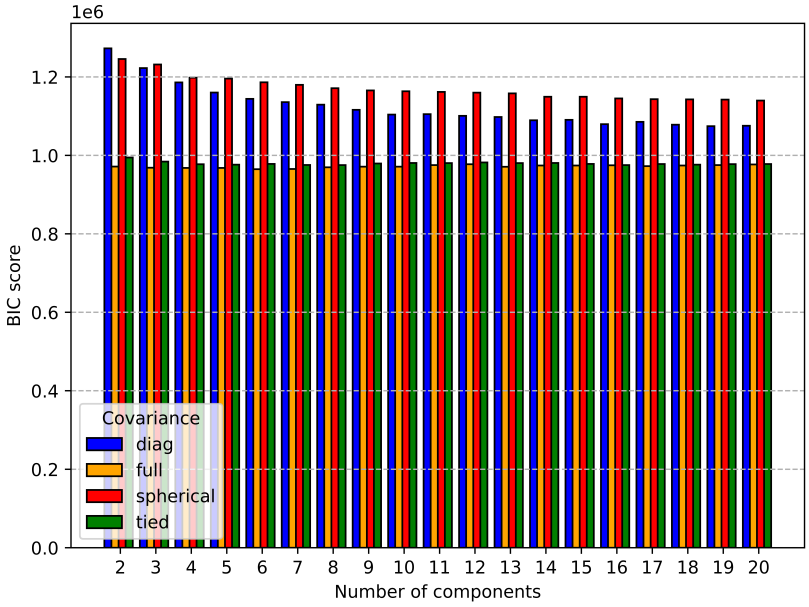
	Název	Popis	Jednotka	Dolní	Horní	Min	Max
	75MAD10CY001_XQ01	VIBR.FRONT HP ROTOR - X	µm	-25.00	525.00	0.00	500.00
	75MAD10CY002_XQ01	VIBR.FRONT HP ROTOR - Y	µm	13.80	75.40	16.60	72.60
	75MAD20CY001_XQ01	VIBR.REAR HP ROTOR - X	µm	30.77	59.78	32.09	58.46
	75MAD20CY002_XQ01	VIBR.REAR HP ROTOR - Y	µm	29.74	45.60	30.46	44.88
	75MKD10CY001_XQ01	VIBR.FRONT GEN.ROTOR - X	µm	34.74	74.55	36.55	72.74
	75MKD10CY002_XQ01	VIBR.FRONT GEN.ROTOR - Y	µm	36.47	57.65	37.43	56.69
	75MKD20CY001_XQ01	VIBR.REAR GEN.ROTOR - X	µm	48.16	82.29	49.71	80.74
	75MKD20CY002_XQ01	VIBR.REAR GEN.ROTOR - Y	µm	34.51	64.23	35.86	62.88
	75MKA10CE001_XQ01	POWER OF TG	MW	18.44	50.95	19.92	49.47
	75MAD10CS901_XQ01	SPEED TURBINE	rpm	4794.96	4823.35	4796.25	4822.06

Obrázek 4.42: Celá datová sada dostupná pro událost 1

4.2.5 Událost 2 - GMM

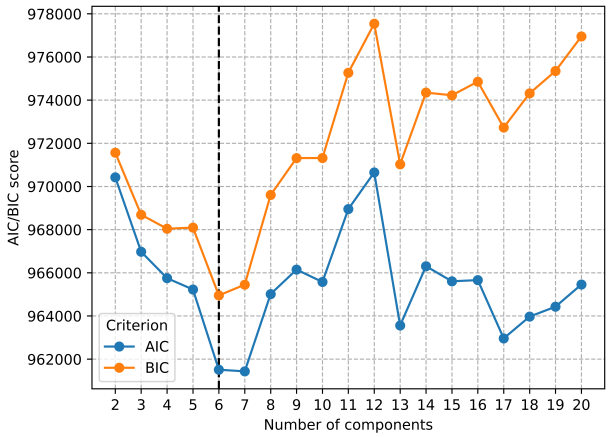
Trénování klasifikátoru.

Trénovací množina byla vybrána v rozsahu indexů 0 až 225 000 neboli 22 % dostupných vzorků a 49,9 % vzorků označených expertem jako normální. Struktura hodnot zůstává stejná, tedy počet komponent směsi $n \in \langle 2; 20 \rangle$ a všechny čtyři podporované typy kovariance komponent, stejně tak i použité hodnotící kritérium opět Bayesovo informační kritérium. Jako nejlepší byla algoritmem exhaustive grid search vybrán počet komponent $n = 6$ a typ kovariance komponent full, jež dosáhla skóre BIC = 964 950,82. K nalezení optimálních parametrů směsi bylo třeba 25 iterací. Proces hledání trval 109,4 minuty.



Obrázek 4.43: Srovnání BIC skóre všech kandidátů

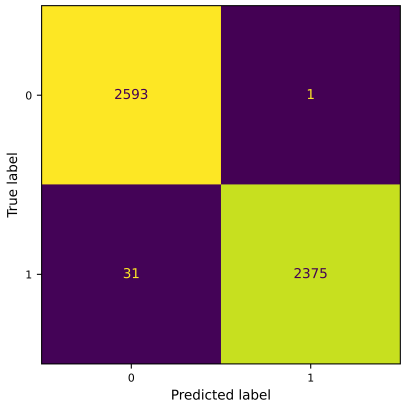
Na srovnání BIC skóre všech kandidátů je patrný malý rozdíl mezi typy kovariance full a tied, opět však byla jako nejlepší vyhodnocena varianta full. Její BIC skóre se s rostoucím počtem komponent příliš neměnilo.



Obrázek 4.44: Vývoj BIC skóre pro typ kovariance full

Detekce anomálií.

Testovací množina obsahuje pouze začátek označené události, neboť hledaná událost má dlouhé trvání. Z předchozích experimentů vyplývá, že pokud klasifikátor úspěšně detekuje anomální stav v jeho počátku bude jej úspěšně detekovat i kdekoliv v jeho průběhu, neboť anomalita vzorků vyjádřená jejich vzdálenosti zpravidla po vzniku anomálního stavu vždy výrazně narůstá. Zvolený rozsah indexů byl proto 448 000 až 453 000. Prahová hodnota byla ponechána na konzervativní hodnotě 8,88. Proces detekce anomálií trval 114,8 minut a z 5 000 vzorků bylo pouze 32 klasifikováno chybně.



Obrázek 4.45: Matice záměn

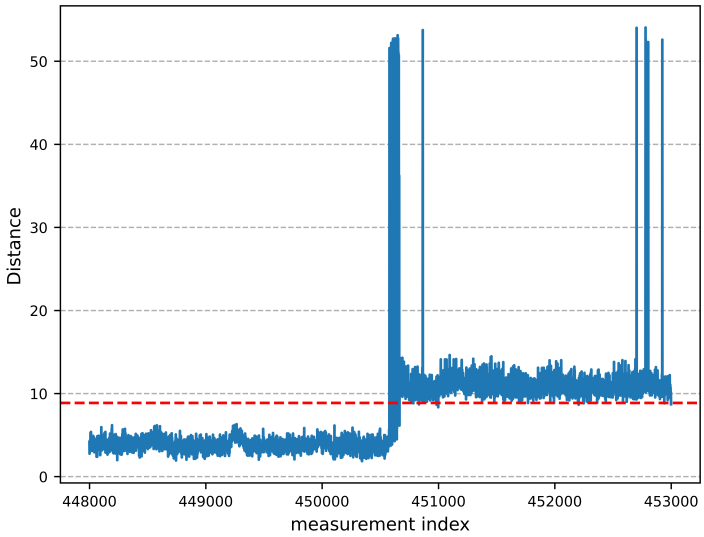
Dobrý výkon klasifikátoru potvrzuje klasifikační report. Celková přesnost klasifikace dosáhla 99 %. Ze všech vzorků klasifikovaných jako anomální bylo téměř

100 % opravdu anomální, navíc i 99 % vzorků klasifikovaných jako normální byly opravdu normální.

	Precision score	Recall score	F1-score	Počet vzorků
Normální vzorky	0.9882	0.9996	0.9939	2594
Anomální vzorky	0.9996	0.9871	0.9933	2406
Vážený průměr	0.9937	0.9936	0.9936	5000
Accuracy			0.9936	5000

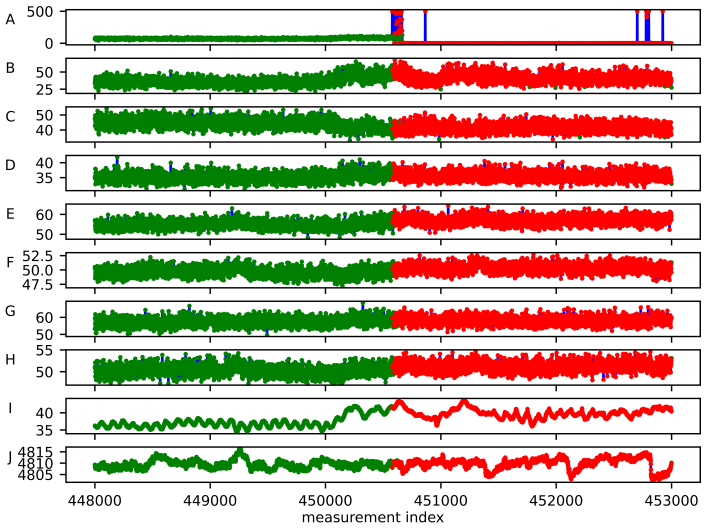
Tabulka 4.10: Klasifikační report

Graf Mahalanobisových vzdáleností jasně zachycuje náhlý nárůst anomality vzorků. Díky jasnému ohraničení této události v datech lze očekávat i nepatrný či žádný rozpor výsledků klasifikace s odhadem experta.



Obrázek 4.46: Vzdálenosti vzorků testovací množiny od dělící nadroviny

Označení vzorků trénovací množiny klasifikátorem na Obr. 4.47 je téměř shodné s označením poskytnutým expertem.

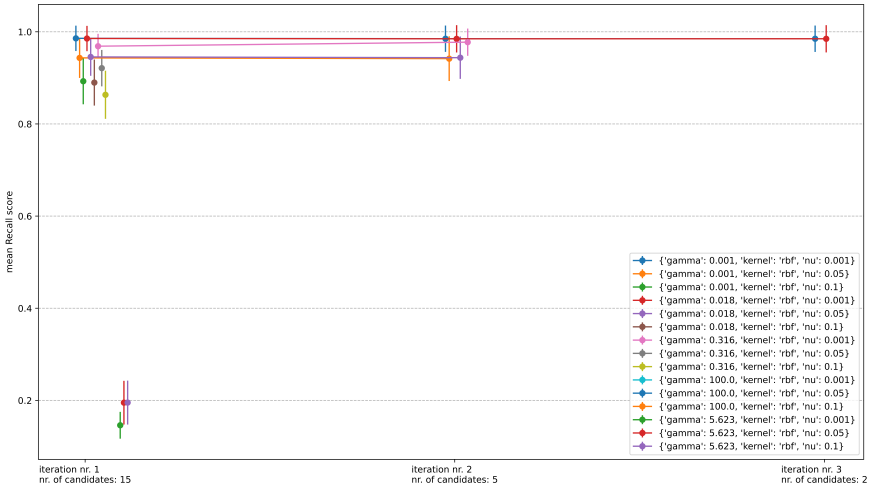


Obrázek 4.47: Testovací množina označená klasifikátorem

4.2.6 Událost 2 - SVM

Trénování klasifikátoru.

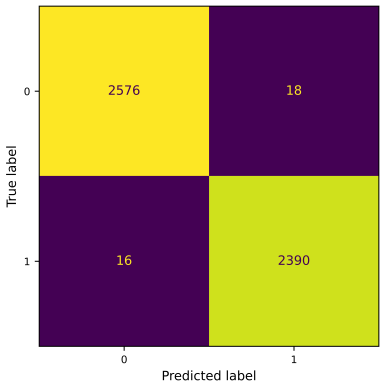
Pro proces učení byla použitá stejná trénovací množina jako pro klasifikátor s GMM, struktura hodnot parametrů zůstává stále stejná – $\nu \in \{0,001; 0,01; 0,1\}$, $\gamma \epsilon \{0,001; 0,018; 0,316; 5,623; 100\}$ pro RBF kernel – a pro výběr nejlepší kombinace parametrů bylo znovu jako kritérium použito Recall score. Successive halving algoritmus vybral v čase 112,2 minut jako nejlepší kombinaci $\nu = 0,001$ a $\gamma = 0,018$. Recall score klasifikátoru s těmito parametry bylo 0,9776. Skóre všech kandidátů je zobrazeno na Obr. 4.48.



Obrázek 4.48: Vývoj skóre kandidátů v iterací successive halving

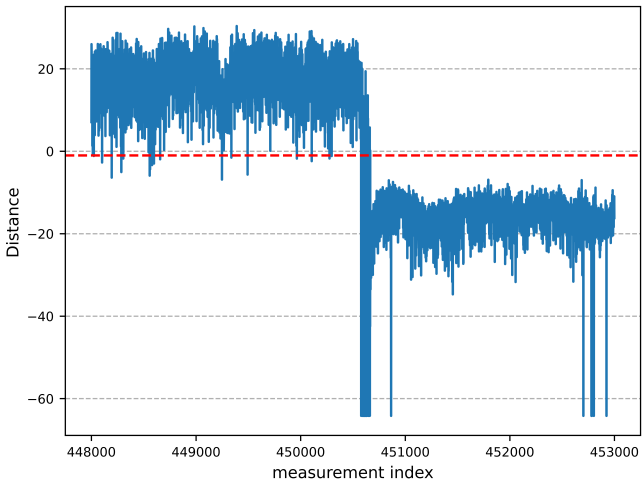
Detekce anomálií.

Interval, na němž probíhala detekce anomálií, zůstává stejný, tedy vzorky exportu s index v rozpětí 448 000 až 453 000. Prahová hodnota byla ponechána na hodnotě 0,00. Detekce anomálií trvala 149,5 minuty. Se zmíněnou prahovou hodnotou bylo 35 vzorků klasifikováno nesprávně, většina chybných výstupů spadal do kategorie falešně pozitivních. Po zvýšení prahové hodnoty na 1,00 jako v případě druhé události u předchozího zařízení došlo k mírnému zlepšení výsledků. Tyto výsledky shrnuje matice záměn.



Obrázek 4.49: Matice záměn

Na grafu vzdáleností vzorků od dělicí nadroviny lze pozorovat velké výkyvy vzdáleností normálních obrazů. Stejné výkyvy lze pozorovat i v grafu Mahalanobisových vzdáleností klasifikátoru s GMM, díky konzervativnímu nastavení prahové hodnoty ale tyto obrazy nebyly klasifikovány jako anomální.



Obrázek 4.50: Vzdálenosti vzorků testovací množiny od dělící nadroviny

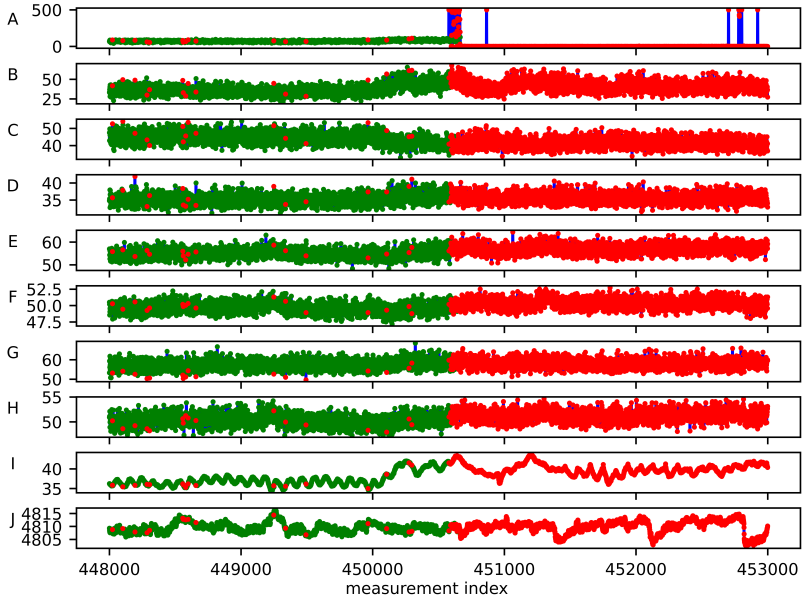
Při pohledu na Obr. 4.39 je možné tyto výkyvy vysvětlit zvyšující se amplitudou měřených vibrací v blízkosti přechodu zařízení do anomálního stavu, jež záměrně nebyla součástí trénovací množiny. Ta je vždy vybírána tak, aby byl mezi ní a testovací množinou určitý odstup pro případ, že by začátek identifikované události byl expertem označen špatně.

Chybná identifikace některých normálních obrazů se promítá i do klasifikačního reportu. Přesto byla kvalita klasifikace srovnatelná s klasifikátorem s GMM. Celková přesnost dosáhla 99 %. Stejně tak i úspěšnost klasifikace obrazů jako normálních, respektive anomálních, dosahovala 99 %.

	Precision score	Recall score	F1-score	Počet vzorků
Normální vzorky	0.9938	0.9931	0.9934	2594
Anomální vzorky	0.9925	0.9933	0.9929	2406
Vážený průměr	0.9932	0.9932	0.9932	5000
Accuracy			0.9932	5000

Tabulka 4.11: Klasifikační report

Chybně identifikované normální obrazy jsou zřetelně rozeznatelné v grafu signálů označených na základě výsledků klasifikace.



Obrázek 4.51: Testovací množina označená klasifikátorem

4.3 Srovnání použitých klasifikátorů

Na základě provedené validace navrženého modulu detekce anomálií pro systém RMS je možné porovnat oba použité klasifikátory z hlediska rychlosti i kvality klasifikace, ale i z pohledu jednoduchosti a použitelnosti v kontextu návrhových kritérií. U porovnávaných výsledků je třeba brát v potaz možná zkreslení uvedených časů plynoucí z parametrů a nerovnoměrného vytížení použitého hardware a operačního systému. Velikost trénovacích a testovacích množin byla ze stejných důvodů omezena s cílem dosáhnout praktické doby trvání trénovacích procesů klasifikátorů a samotného procesu detekce anomálií umožňující experimenty několikrát opakovat. V praxi by byla trénovací množina větší a musela by pokrýt všechny známé normální stavy zařízení.

Délka trvání procesů trénování nehraje v kontextu aplikace řešení velkou roli, neboť lze očekávat, že tréning bude probíhat pouze v průběhu nasazení modulu na konkrétní zařízení. Je však stále zajímavým údajem, neboť je ovlivněna parametry jako je velikost trénovací množiny nebo počet sledovaných signálů společné domény. Pomáhá tak ilustrovat jednoduchost nastavení modulu detekce anomálií pro konkrétní zařízení. Z hlediska doby trvání procesu trénování se jako lepší volba jeví OCSVM. Celková doba tréningu byla s výjimkou posledního experimentu vždy výrazně kratší než u klasifikátoru s GMM. Po přepočtu doby učení na velikost trénovací množiny je patrné, že nejpomalejší čas učení OCSVM se vyrovnává nejrychlejšímu času učení klasifikátoru s GMM. Oba časové údaje pro oba klasifikátory a všechny čtyři experimenty shrnuje Tab. 4.12. Velkou nevýhodou klasifikátoru s GMM se ukázala být citlivost na obrazy trénovací množiny s extrémními hodnotami některých složek.

	Klasifikátor s GMM		OCSVM	
	Tréning [min] / [s]	Klasifikace [s]	Tréning [min] / [s]	Klasifikace [s]
TG 36MW, událost 1	224,4 / 0,06	0,70	109,6 / 0,03	0,81
TG 36MW, událost 2	204,9 / 0,05	0,86	123,7 / 0,03	0,81
TG 55MW, událost 1	153,1 / 0,04	0,84	81,3 / 0,02	0,86
TG 55MW, událost 2	109,4 / 0,03	1,38	112,2 / 0,03	1,79
Průměr		0,95		1,07

Tabulka 4.12: Souhrn rychlosti trénování a klasifikace

Samotná rychlost klasifikace jednotlivých vzorků je podstatná z hlediska nasazení modulu na reálné zařízení, neboť musí být rychlejší než nejmenší perioda měření signálu zařízení. Signály jsou však do databáze přidávány s poměrně velkou

periodou; některé až v řádu minut. V tomto ohledu jsou oba klasifikátory plně vyhovující, neboť v nejhorším případě nepřekročila rychlost klasifikace 2 s a ve většině případů byla kratší než 1 s. Vypočtené průměrné doby klasifikace jednoho vzorku uvádí Tab. 4.12.

Důležitějšími hodnotícími parametry jsou kvalita klasifikace a počet falešně pozitivních výsledků. Klasifikátor s GMM dosáhl průměrné přesnosti 94,74 % a průměrného podílu správně identifikovaných inlierů a outlierů 94,57 %. V těchto statistikách těsně překonává OCSVM s průměrnou přesností klasifikace 93,01 % a průměrným podílem správně identifikovaných obrazů 93,01 %, tedy rozdílem 1,73 %, respektive 1,56 %. Naopak v správně identifikovaných normálních obrazů se jako lepší ukázal OCSVM s hodnotou 99,44 % oproti 98,91 % pro klasifikátor s GMM. Výrazný rozdíl je v počtu falešně pozitivních výsledků. Klasifikátor s GMM jich celkem provedl 84 zatímco OCSVM pouze 48. Uvedené statistiky shrnuje Tab. 4.13. Je také třeba zohlednit nutnost nastavení prahové hodnoty. Zatímco u klasifikátoru s GMM je nutné z dostupných normálních dat odhadnout vhodnou nenulovou prahovou hodnotu, OCSVM si ve většině případů vystačil s nulovou prahovou hodnotou, respektive nulovým ofsetem vzdálenosti obrazu od dělící nadroviny. Jako praktičtější z obou klasifikátorů se proto jeví OCSVM, navíc lépe splňuje návrhové kritérium minimalizace falešně pozitivních výsledků. Klasifikátor s GMM je oproti tomu marginálně celkově přesnější. Oba dva klasifikátory jsou však pro použití v navrženém řešení akceptovatelné.

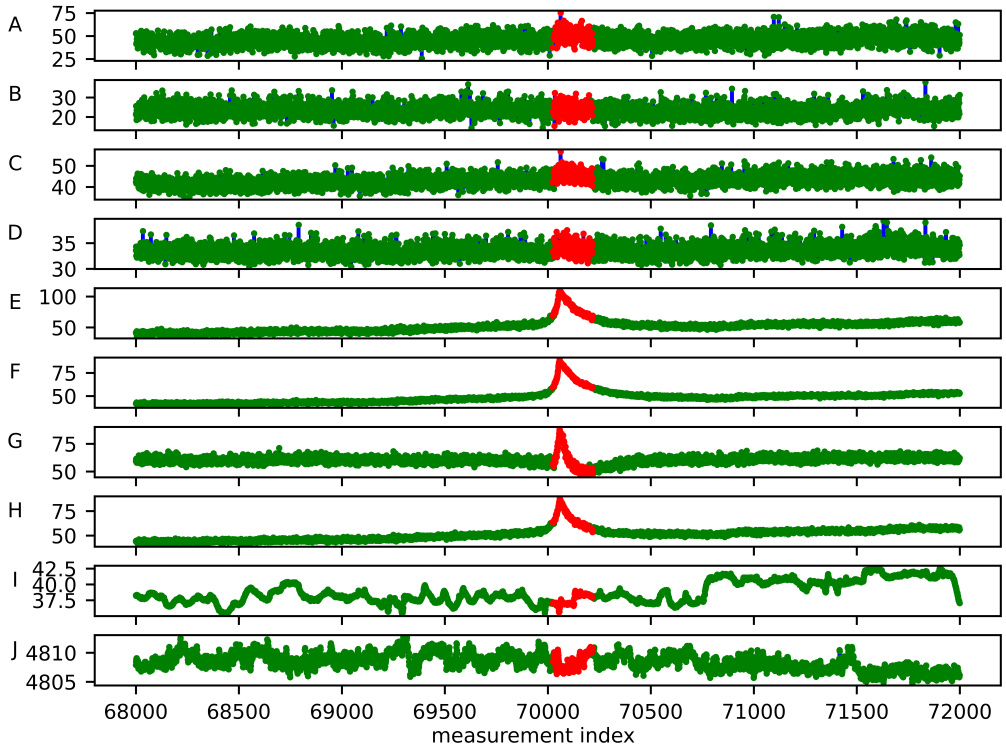
	Klasifikátor s GMM			
	Accuracy	Recall inliers	Avg. Recall	Falešně pozitivní
TG 36MW, událost 1	0,9038	0,9996	0,9038	1
TG 36MW, událost 2	0,9215	0,9570	0,9215	82
TG 55MW, událost 1	0,9705	1,0000	0,9705	0
TG 55MW, událost 2	0,9936	0,9996	0,9871	1
Průměr	0,9474	0,9891	0,9457	84
	OCSVM			
TG 36MW, událost 1	0,8925	1,0000	0,8925	0
TG 36MW, událost 2	0,8752	0,9848	0,8752	29
TG 55MW, událost 1	0,9595	0,9997	0,9595	1
TG 55MW, událost 2	0,9932	0,9931	0,9932	18
Průměr	0,9301	0,9944	0,9301	48

Tabulka 4.13: Souhrnné výsledky kvality klasifikace

4.4 Grafické výstupy navrženého řešení

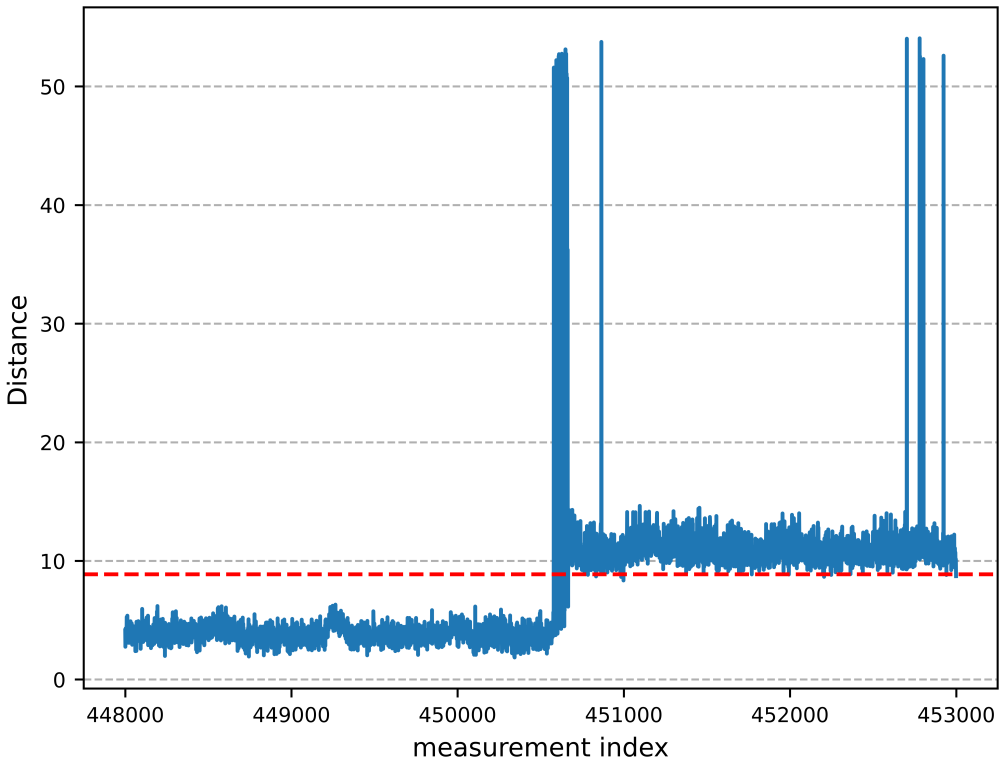
Grafický výstup je neodmyslitelnou součástí nejen procesu validace a interpretace detekčních metod v rámci detekce anomálií, ale celého systému monitorování a diagnostiky průmyslových zařízení. Součástí této práce bylo i zkoumání způsobů, jak vizualizovat výsledky implementovaných detekčních metod, aby bylo možné lépe porozumět detekci anomálií a interpretovat její výstupy.

V průběhu celé kapitoly validace řešení byly používány zejména dvě z navržených vizualizací výstupu. Prvním z nich je značkovaný graf průběhu sledovaných signálů v čase. Tato vizualizace umožňuje detailní zkoumání signálů a detekcí anomálií v kontextu časových úseků. Anomální události jsou v grafech vizuálně zvýrazněny, což usnadňuje identifikaci a analýzu jejich výskytu v datech.



Obrázek 4.52: Testovací množina označená klasifikátorem, událost 1 na TG 55MW

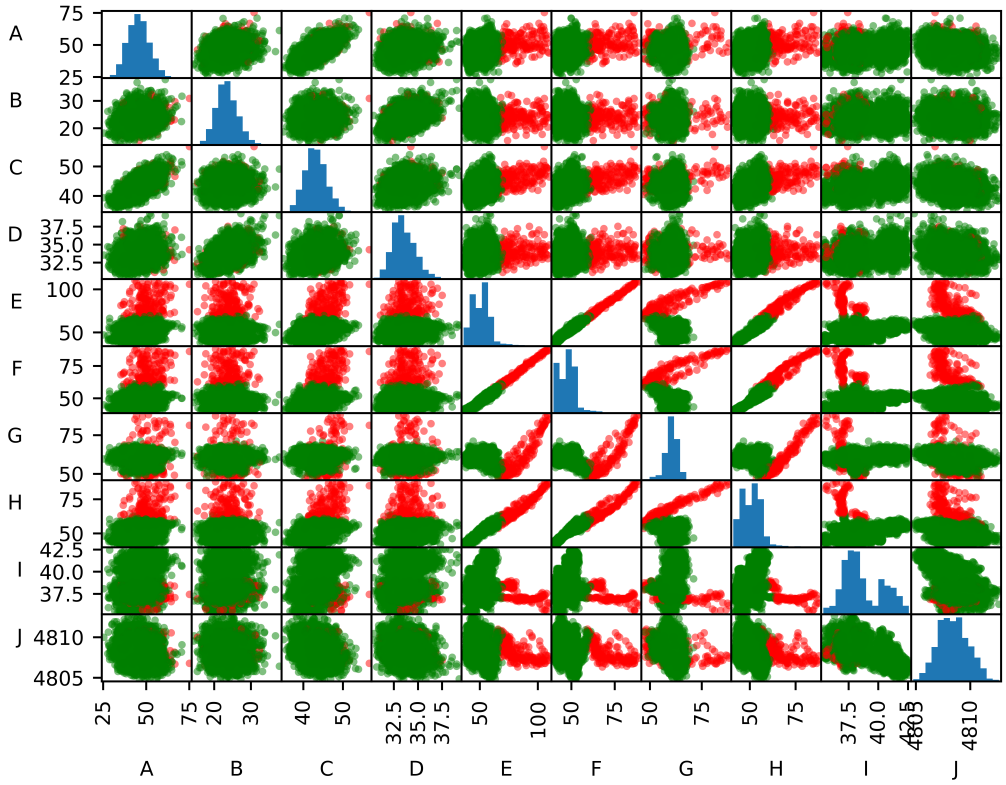
Druhým používaným způsobem vizualizace výstupu byl graf Mahalanobisových vzdáleností obrazů, respektive graf vzdáleností obrazů od dělící nadroviny. Vzdálenost anomálních obrazů je výrazně odlišná od normálních obrazů a proto je tento způsob vizualizace vhodný pro rychlé identifikování časového úseku výskytu anomálie.



Obrázek 4.53: Graf Mahalanobisových vzdáleností pro událost 2 na TG 55MW

Poté, co operátor přijme informaci o detekované anomálii nebo bude revidovat zaznamenané výsledky, může tento typ zobrazení využít k rychlé identifikaci časových údajú a k bližší analýze reportované anomálie pak použít například značkovaný graf průběhu sledovaných signálů.

Poslední navrženou zobrazovací metodou je scatter matrix graf, také známý jako pair plot nebo splom, který ilustruje vztahy mezi různými sledovanými signály. Scatter matrix je tvořen maticí grafů, kde na diagonále jsou histogramy jednotlivých signálů. V ostatních buňkách matice jsou umístěny scatter ploty, které zobrazují vztahy mezi jednotlivými páry signálů. Lze jej použít pro podrobnější analýzu dat a to i díky tomu, že jednotlivá měření jsou také označkována výstupem klasifikátoru. To umožňuje identifikovat, jaké typy vzorků jsou správně nebo nesprávně klasifikovány jako anomální. Scatter matrix graf poskytuje komplexní pohled na vzájemné vztahy mezi proměnnými a jejich vliv na klasifikaci. Tuto zobrazovací metodu je možné degradovat na graf jednotlivých párů signálů, což umožňuje detailnější pohled na korelaci jejich hodnot a přispět tak k odhalení původu anomálie.



Obrázek 4.54: Scatter matrix graf pro událost 2 na TG 55MW

Závěr 5

Úvodem práce byla představena problematika monitorování a diagnostiky turbín a její současný stav. Cílem práce bylo pokračovat v automatizaci tohoto důležitého procesu, a to navržením modulu pro automatickou detekci anomálních stavů turbín, který by mohl být integrován do již existujícího a používaného systému vzdáleného monitorování elektrárenských zařízení v reálném čase. Na základě několika návrhových kritérií bylo navrženo řešení založené na detekci anomálií pomocí balíku v jazyce Python. Samotná úloha detekce anomálií byla představena v rámci úvodu. Ze známých algoritmů detekce anomálií byly vybráni dva zástupci, a to klasifikátor s GMM a klasifikátor OCSVM. Jejich princip funkce a proces jejich tvorby byly v krátkosti představeny.

Dále byla provedena rešerše dostupných řešení pro komunikaci s SQL databází v jazyce Python a dostupných nástrojů pro práci se strojovým učením. Vybrané balíky byly pak použity při návrhu architektury a designu modulu detekce anomálií pro systém RMS. Při návrhu architektury projektu byl kladen důraz na modularitu, oddělení zodpovědností a snadnou rozšiřitelnost. Hierarchická struktura balíčků umožňuje jednoduchou správu a údržbu projektu. Použité návrhové vzory přispívají k elegantnímu řešení požadavků detekce anomálií. Funkčnost všech součástí balíku byla podrobně popsána v jednotlivých kapitolách.

Vytvořený software byl otestován na čtyřech sadách reálných dat měřených na elektrárnách provozujících systém RMS. V některé části všech čtyř datových setů se projevoval anomální stav zařízení, který byl identifikován expertem. Součástí datových sad pak byl popis každého vzorku expertem. Testování na datech načítaných přímo z databáze RMS nebylo možné, neboť integrace modulu detekce anomálií do RMS nebyla včas dokončena.

Provedením experimentů nad všemi čtyřmi sadami dat bylo validováno jejich správné zpracování. Trénování vybraných klasifikátorů bylo ve všech případech úspěšné, v jednom případě však bylo nutné hledat vhodnou trénovací množinu pro klasifikátor s GMM, s níž by byl Expectation-Maximization algoritmus nedivergoval. Kromě citlivosti na data s extrémními hodnotami složek v trénovací množině ukázal klasifikátor s GMM drobnou nevýhodu v délce trvání procesu učení. S do-

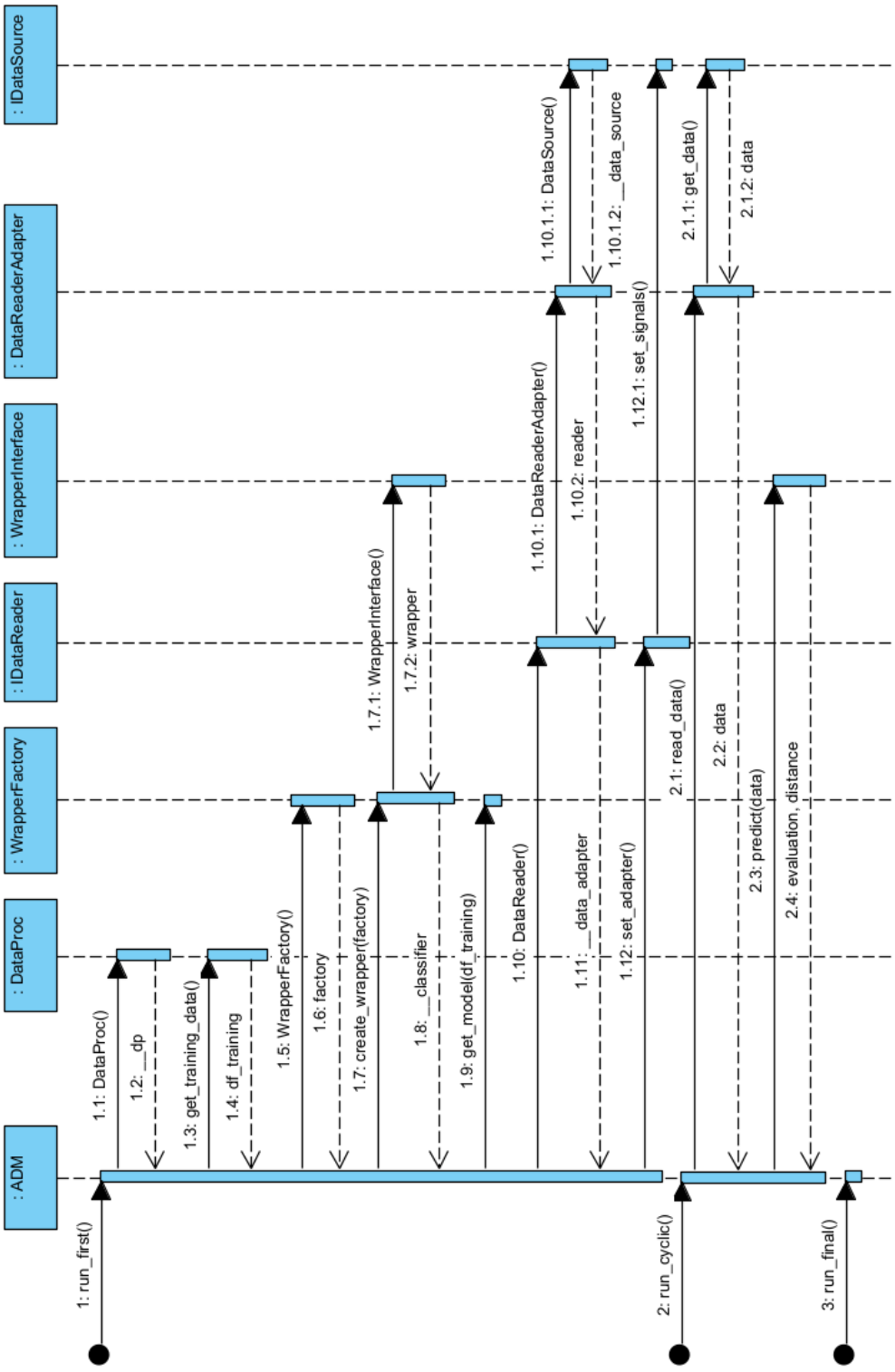
stupnými prostředky nebylo možné provést proces učení nad takovou trénovací sadou, která by zastupovala všechny normální stavy turbín. Použití takové datové sady je velice důležité pro správnou funkci modulu detekce anomálií v praktickém prostředí neomezeném na krátké úseky dat. Hledání optimálního modelu pro klasifikátor s GMM by bylo možné dále rozšířit o vlastní implementaci výběru kombinace parametrů z výsledků algoritmu exhaustive grid search. Tento krok by zřejmě nevedl ke zkvalitnění klasifikace odpovídající času nutnému pro jeho implementaci, mohl by však snížit výpočetní nároky modulu detekce anomálií.

Během samotné detekce anomálií na vzorcích testovací množiny podaly oba klasifikátory srovnatelný výkon. OCSVM klasifikátor dosahoval nižšího počtu falešně pozitivních výsledků, klasifikátor s GMM byl však celkově přesnější. Rychlost klasifikace jednotlivých vzorků byla pro potřeby systému RMS zcela vyhovující v obou případech.

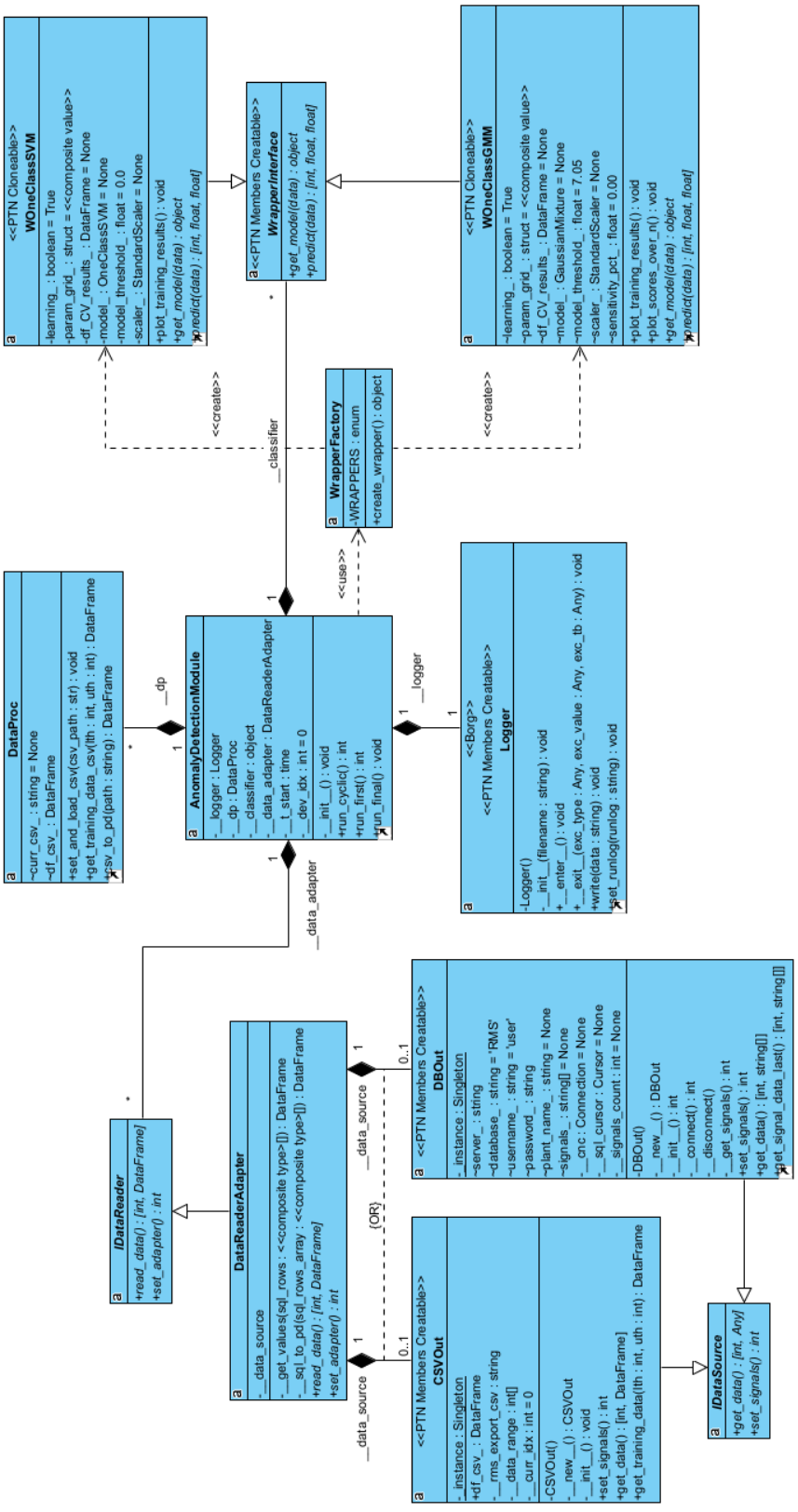
Volba prahové hodnoty pro klasifikaci inlierů, respektive outlierů, proběhla odhadem z vypočítaných vzdáleností normálních dat. Klasifikátor s GMM vyžadoval nastavení nenulové prahové hodnoty vždy, oproti tomu OCSVM si ve většině případů vystačil s výchozí hodnotou 0. Výběr prahové hodnoty by bylo možné alespoň v případě klasifikátoru s GMM automatizovat, avšak výpočetní náročnost určení dobré hodnoty by byla velmi vysoká, neboť by musela probíhat nad trénovací množinou, jež je zpravidla velmi obsáhlá. Navíc by nebylo zaručeno, že vypočítaná hodnota bude zajišťovat dostatečně nízký počet falešně pozitivních výsledků. Je důležité si uvědomit, že detekce probíhala na datových sadách, v nichž byl expert schopen anomální událost identifikovat. V praxi by však bylo navržené řešení schopné identifikovat i anomálie, které expertovi uniknou, neboť se neprojevují viditelnou odchylkou v průběhu hodnot některého signálu v čase.

Provedenými experimenty se tedy podařilo prokázat použitelnost nejen zvolených metod ale celého navrženého řešení. Závěrem byly nastíněny možné způsoby zobrazení výstupu modulu detekce anomálií v grafickém rozhraní systému RMS.

UML diagramy



Obrázek A.1: Sekvenční diagram volání hlavní třídy modulu detekce anomálií



Obrázek A.2: Digram tříd modulu detekce anomálií

Seznam symbolů a zkratek

B

AIC Akaike Information Criterion - Akaikeho informační kritérium

API Application Programming Interface - rozhraní pro programování aplikací

b ofset dělící nadroviny

BIC Bayesian Infromation Criterion - Bayesovo informační kritérium

C váha inlierů v porovnání s outliery

CSV Comma Separated Values - souborový formát určený pro výměnu tabulkových dat

D dělící nadrovina

EM-algorithm Expectation-Maximization algoritmus - iterační metoda pro hledání maximálně věrohodného odhadu

FN false-negative - obraz, který do třídy náleží ale byl klasifikován špatně

FP false-positive - obraz, který nenáleží do třídy a byl klasifikován špatně

GMM Gaussian Mixture Model - směs normálních rozdělení

$I(\cdot)$ rozhodovací funkce

IIoT Industrial Internet of Things průmyslový internet věcí

M, M^* pásmo necitlivosti, optimální pásmo necitlivosti
Mahalanobisova vzdálenost

n počet komponent směsi GMM

NTIS Nové technologie pro informační společnost

OCSVM One-Class Support Vector Machine - jednotřídní metoda podpůrných vektorů

OPC Open Platform Communications - sada protokolů pro průmyslovou automatizaci

RBF Radial Basis Function - radiální bazická funkce - funkce k modelování vzorů a vzdáleností mezi daty

RMS Remote Monitoring System

$sgn(\cdot)$ funkce signum

SQL Structured Query Language - standardizovaný strukturovaný dotazovací jazyk pro práci s daty v relačních databázích

TCP/IP Transmission Control Protocol/Internet Protocol

TP true-positive - obraz, který náleží do třídy a byl klasifikován správně

UML Unified Modeling Language - grafický jazyk pro vizualizaci, specifikaci, navrhování a dokumentaci programových systémů

w vektor parametrů dělící nadroviny

x obraz

y výstup klasifikátoru

α koeficient směšování komponenty GMM

γ vzdálenost vybraných podpůrných vektorů od sebe
aposteriorní pravděpodobnost v E-step EM-algoritmu

Θ prahová hodnota pro klasifikátoru

μ střední hodnota

ν pravděpodobnost, že obraz trénovací množiny je outlier

ξ penalizační proměnná (slack-variable)

σ rozptyl

Σ kovarianční matice

Φ jádrová funkce

Bibliografie

[Agg17] AGGRAWAL, Charu C. *Outlier Analysis*. Cham: Springer International Publishing, 2017. ISBN 978-3-319-47577-6.

[BS99] BERNHARD SCHÖLKOPF Robert Williamson, Alex Smola; SHAWE-TAYLOR, John. *Single-class Support Vector Machines*. Max-Planck-Gesellschaft, 1999.

[BM98] BERNHARD SCHÖLKOPF, Alexander Smola; MÜLLER, Klaus-Robert. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*. 1998, roč. 10, č. 5, s. 1299–1319.

[Bis06] BISHOP, Christopher M. *Pattern Recognition and Machine Learning*. New York: Springer Science Business Media, 2006. ISBN 978-0387-31073-2.

[BGV92] BOSER, Bernhard E.; GUYON, Isabelle M.; VAPNIK, Vladimir N. A Training Algorithm for Optimal Margin Classifiers. In: *Proceedings of the fifth annual workshop on Computational learning theory*. New York, NY, USA, 1992, s. 144–152.

[Cod70] CODD, E. F. A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*. 1970, roč. 13, č. 6, s. 377–387.

[DH73] DUDA, Richard O.; HART, Peter E. *Pattern Classification and Scene Analysis*. New York: John Wiley Sons, 1973. ISBN 978-0471223610.

[Fou23] FOUNDATION, Python Software. *Python*. Python Software Foundation, 2023-07-02. Dostupné také z: `https://www.python.org`.

[Kle23] KLEEHAMMER, Michael. *pyodbc*. pyodbc developer community, 2023-11-17. Dostupné také z: `https://github.com/mkleehammer/pyodbc/wiki`.

[Lai+21] LAI, Kwei-Harng et al. TODS: An Automated Time Series Outlier Detection System. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence*. Washington, DC, USA: AAAI Press, 2021, s. 16060–16062.

[LVK19] LIŠKA, Jindřich; VAŠÍČEK, Vojtěch; KÁŠ, Martin. Real-Time Remote Monitoring of Power Plants in terms of IIoT and Cloud Computing. In: *Proceedings of Structural Heath Monitoring*. Stanford, CA, USA, 2019, s. 534–541.

[MB88] MCLACHLAN, G.; BASFORD, K. *Mixture Models: Inference and Applications to Clustering*. New York: Dekker, 1988. ISBN 0-8247-7691-7.

[Mik22] MIKHAIL TEREKHOV Alex Hagerman, Ramiro Morales. *pymssql*. pymssql developer community, 2022-11-16. Dostupné také z: `https://pymssql.readthedocs.io/en/stable/`.

[Mül+01] MÜLLER, Klaus-Robert; MIKA, Sebastian; RÄTSCH, Gunnar; TSUDA, Koji; SCHÖLKOPF, Bernhard. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*. 2001, roč. 12, č. 2, s. 181–201.

[Ped+11] PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011, roč. 12, s. 2825–2830.

[Sch+99] SCHÖLKOPF, Bernhard; WILLIAMSON, Robert; SMOLA, Alex; SHAWE-TAYLOR, John; PLATT, John. Support Vector Method for Novelty Detection. In: *Proceedings of the 12th International Conference on Neural Information Processing Systems*. Denver, CO, USA, 1999, s. 582–588.

[SC08] STEINWART, Ingo; CHRISTMANN, Andreas. *Support vector machines*. New York: Springer Science Business Media, 2008. ISBN 978-0-387-77241-7.

[Tax01] TAX, David Martinus Johannes. *One-class classification*. 2001. Dis. pr. Delft University of Technology.

[Vap98] VAPNIK, Vladimir N. *Statistical Learning Theory*. Hoboken: Wiley-Interscience, 1998. ISBN 978-0-471-03003-4.

[VC74] VAPNIK, Vladimir N.; CHERVONENKIS, Alexej A. *Pattern Recognition. Theory, Statistical Learning Problems*. Moskva: Nauka, 1974.

[YK20] YILMAZ, Selim F; KOZAT, Suleyman S. PySAD: A Streaming Anomaly Detection Framework in Python. *arXiv preprint arXiv:2009.02572*. 2020.

[ZNL19] ZHAO, Yue; NASRULLAH, Zain; LI, Zheng. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research*. 2019, roč. 20, č. 96, s. 1–7.

Seznam obrázků

Seznam tabulek