Razvoj modela za prediktivno održavanje rotacijske opreme temeljenog na ultrazvučnoj analizi

Ugarković, Lovro

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Lovro Ugarković

Zagreb, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

Dr. sc. Davor Kolar, docent

Student:

Lovro Ugarković

Zagreb, 2024.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru Davoru Kolaru na pruženoj pomoći, strpljenju i vođenju tokom izrade ovog rada.

Lovro Ugarković

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za diplomske ispite studija strojarstva za smjerove:
Proizvodno inženjerstvo, inženjerstvo materijala, industrijsko inženjerstvo i menadžment, mehatronika i robotika, autonomni sustavi i računalna inteligencija

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 - 04 / 24 - 06 / 1**	
Ur.broj: **15 - 24 -**	

DIPLOMSKI ZADATAK

Student: **Lovro Ugarković** JMBAG: 0035214999

Naslov rada na hrvatskom jeziku: **Razvoj modela za prediktivno održavanje rotacijske opreme temeljenog na ultrazvučnoj analizi**

Naslov rada na engleskom jeziku: **Development of a Predictive Maintenance Model for Rotating Equipment Based on Ultrasound Analysis**

Opis zadatka:

Prediktivno održavanje podrazumijeva prikupljanje i analizu podataka o stanju opreme kako bi se spriječili kvarovi i optimizirali troškovi održavanja. Ultrazvučna analiza je jedna od naprednih metoda koja se koristi za nadzor stanja rotacijske opreme, omogućujući rano otkrivanje potencijalnih kvarova. Korištenjem hand-held uređaja za ultrazvučno mjerenje, moguće je prikupljati podatke te ih iskoristiti za razvoj prediktivnih modela.

U skladu s navedenim, u radu je potrebno:

1.Opisati tehnologiju ultrazvučne analize i mogućnost primjene u prediktivnom održavanju

2.Predložiti i razviti programsko rješenje prediktivnog modela temeljenog na analizi ultrazvuka

3.Na konkretnom primjeru provjeriti funkcionalnost razvijenog prediktivnog modela

4.Na temelju rezultata istraživanja definirati zaključak i dati prijedlog daljnjeg istraživanja

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
26. rujna 2024.	28. studeni 2024.	5., 6. i 9. prosinca 2024.

Zadatak zadao:

Doc.dr.sc. Davor Kolar

Predsjednik Povjerenstva:

Prof. dr. sc. Ivica Garašić

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

Ovaj diplomski rad istražuje primjenu strojnog učenja u prediktivnom održavanju rotacijske opreme kroz prepoznavanje kvarova na ležajevima temeljem ultrazvučne analize. Glavni cilj istraživanja je razviti aplikaciju koja automatski detektira kvarove na ležajevima te korisnicima omogućuje pravovremeno održavanje čime se smanjuje rizik od neočekivanih zastoja i povećava učinkovitost održavanja. Prikupljanje podataka provedeno je pomoću uređaja SDT 340 koji je snimao ultrazvučne signale u različitim stanjima ležajeva, uključujući normalno stanje i kvarove na unutarnjem prstenu, vanjskom prstenu te kuglici ležaja. Signalni podaci obrađeni su u *Matlab* okruženju, a kroz razvijenu aplikaciju omogućeno je izdvajanje ključnih značajki signala te prikaz samog signala. Za klasifikaciju kvarova korišteni su različiti modeli strojnog učenja, uključujući SVM s kvadratnom jezgrom i neuronske mreže. Modeli su testirani na temelju preciznosti klasifikacije te su postigli visok stupanj točnosti od 96.4%. Rezultati istraživanja potvrđuju prikladnost ovih metoda za prediktivno održavanje te postavljaju temelje za daljnji razvoj sustava za industrijsku primjenu.

Ključne riječi: prediktivno održavanje, ležajevi, ultrazvučna analiza, strojno učenje, Matlab.

SUMMARY

This thesis explores the application of machine learning in predictive maintenance for rotary equipment through bearing fault detection based on ultrasonic analysis. The primary goal of the research is to develop an application that can automatically detect bearing faults, thereby enabling timely maintenance, reducing the risk of unexpected downtimes, and enhancing maintenance efficiency. Data collection was conducted using the SDT 340 device, which recorded ultrasonic signals under different bearing conditions, including normal state and faults in the inner ring, outer ring, and bearing ball. Signal data were processed in the *Matlab* environment, where the developed application enabled the extraction of key features and the display of the signal. Various machine learning models, including an SVM with a quadratic kernel and neural networks, were used for fault classification. The models were evaluated based on classification accuracy and the models achieved a high accuracy of 96.4%. The research findings confirm the suitability of these methods for predictive maintenance and lay the groundwork for further development of industrial maintenance systems.

Key words: predictive maintenance, bearings, ultrasonic analysis, machine learning, Matlab.

1. UVOD

U ovome poglavlju je opisana tema i cilj rada, svrha istraživanja te je dan uvod u samo područje prediktivnog održavanja.

1.1. Tema i cilj rada

U ovom diplomskom radu prikazan je razvoj modela za prediktivno održavanje rotacijske opreme temeljenog na ultrazvučnoj analizi. Trenutno postoji skup alata koji služi za ručnu analizu stanja rotacijske opreme gdje se pomoću uređaja prikupljaju podatci s opreme za testiranje, ubace u određeni program i ručno se provodi analiza stanja. Cilj ovog rada je automatiziranje procesa prikupljanja grešaka kako bi se pravovremeno mogao predvidjeti i otkloniti kvar kada je potrebno.

1.2. Prediktivno održavanje

Tradicionalni pristupi održavanju, kao što su korektivno i preventivno održavanje, često ne pružaju optimalna rješenja za sprječavanje neočekivanih kvarova i smanjenje troškova održavanja. U tom kontekstu, prediktivno održavanje postaje sve važnije kao metoda koja koristi napredne tehnologije za praćenje stanja opreme i predviđanje potencijalnih kvarova prije nego što se dogode.

Korektivno održavanje (engl. *corrective maintenance*) kao strategija podrazumijeva otklanjanje oštećenja i kvarova nakon što se oni pojave. To uzrokuje visoke troškove jer su veći proizvodni gubitci u vremenu i sredstvima zbog neočekivanih i duljih zastoja, potreban je veći broj zaposlenika na održavanju u rezervi te povećani troškovi skladištenja rezervnih dijelova.

Preventivno održavanje (engl. *preventive maintenance*) je strategija održavanja kojom se redovno planira izvođenje radnji održavanja kako bi se spriječili potencijalni kvarovi. Osnovni cilj preventivnog održavanja je očuvanje ispravnosti opreme i produljenje njezinog životnog vijeka kroz redovito održavanje. Osnovna podjela ove strategije održavanja je [1]:

- Održavanje po konstantnom ciklusu (plansko – preventivno održavanje),
- Održavanje po stanju (prediktivno održavanje),
- Kontrolni pregledi.

Za prediktivno održavanje (engl. *predictive maintenance*), kao što se može vidjeti iz gornje podjele, koristi se i pojam održavanje po stanju.

Održavanje po stanju (engl. *condition based maintenance*) obuhvaća proces kojim određujemo stanje svakog dijela tehničkog sustava koji se može pratiti i čije ponašanje se može kontrolirati određenim parametrima poput razine vibracija, razine buke, temperatura itd. Općenito, cilj održavanja po stanju je razviti odgovarajuće metode, postupke i opremu za mjerenje određenih parametara sustava koji pokazuju na očekivanu pojavu oštećenja. Uglavnom, postupak uvođenja održavanja po stanju na nekom stroju uključuje: izbor lokacija mjerenja, izbor mjernih parametara, određivanje dopuštenih vrijednosti parametara prije pojave oštećenja ili kvara, mjerenje parametara, dijagnoza (analiza parametara i uspoređivanje s dopuštenim vrijednostima), donošenje odluke o planu održavanja Slika 1.

Slika 1. Postupak uvođenja održavanja po stanju kontrolom parametara [1]

Dakle, vidljivo je sa slike iznad, nakon što se odaberu parametri, uspoređuju se s dopuštenim vrijednostima te nakon mjerenja, ako je dijagnoza dobra, donosi se odluka o planu održavanja, odnosno usvajaju se strategije održavanja po stanju. Prednosti ove strategije su: povećana sigurnost, raspoloživost održavatelja je veća, poslovi održavanja su smanjeni i kvaliteta

proizvoda je veća. Naravno postoje i nedostaci: menadžeri za organiziranje poslova održavanja imaju veći obim posla, veliki je vremenski razmak između vremena uvođenja i ostvarivanja koristi primjene ove strategije te postoji nesigurnost hoće li održavanje po stanju predvidjeti pojavu oštećenja. Najčešće kontrole pri održavanju po stanju su: vizualna kontrola, kontrola temperature, kontrola pukotina, kontrola mjerenjem vibracija, kontrola mjerenja buke, kontrola korozije te kontrola podmazivanjem.

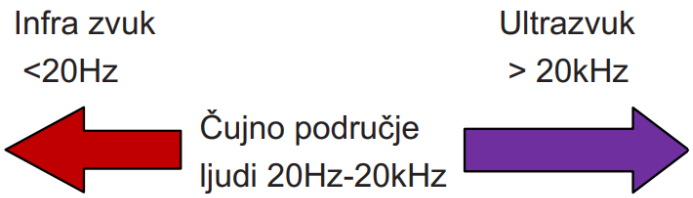
Jedna od najučinkovitijih tehnologija koja se koristi u prediktivnom održavanju je ultrazvučna analiza. Ova metoda omogućuje rano otkrivanje problema u rotacijskoj opremi putem identifikacije visokofrekventnih zvučnih valova koje proizvodi oprema tijekom rada. Ultrazvučna analiza pruža precizne informacije o stanju ležajeva, omogućujući tako pravovremene intervencije i smanjenje rizika od neočekivanih zastoja. Korištenje ultrazvučne analize u prediktivnom održavanju donosi brojne prednosti, uključujući produljenje životnog vijeka opreme, smanjenje operativnih troškova i povećanje sigurnosti rada. Uvođenje ove tehnologije zahtijeva stručnost u interpretaciji ultrazvučnih podataka te implementaciju odgovarajućih alata i strategija za kontinuirano praćenje stanja opreme. U nastavku su detaljnije opisani principi ultrazvučne analize.

2. ULTRAZVUČNA ANALIZA

Ovo poglavlje opisuje samu fiziku ultrazvuka, njegovu primjenu u održavanju, standarde i postupke ispitivanja ultrazvukom te principe rada ultrazvučnih senzora.

2.1. Fizika ultrazvuka

Ultrazvuk je mehanički val koji opisuje područje frekvencija iznad praga čujnosti ljudskog uha od 20 kHz Slika 2.



Slika 2. Frekvencijska područja zvuka [2]

Fizikalne pojave koje se javljaju u procesnoj opremi (trenje, turbulencija, udari, kavitacija, eksplozija, implozija, parcijalna pražnjenja i ionizacija) imaju izražene karakteristike na ultrazvučnim frekvencijama. Kao rezultat, stvaraju se ultrazvučne emisije koje su idealan parametar za praćenje performansi strojeva, stanja strojeva i za dijagnosticiranje anomalija strojeva. Ultrazvuk je idealna tehnologija jer pruža učinkovit način za brzo i neinvazivno određivanje lokacije anomalije uz minimalnu pripremu i u vrlo kratkom vremenu.

Može se prenositi zrakom i preko konstrukcije. Ultrazvuk u zraku se prenosi kroz atmosferu (zrak ili plin) i detektira ultrazvučnim mikrofonom, dok se konstrukcijski nošen ultrazvuk generira unutar strukture i prenosi kroz nju te se obično detektira kontaktnim senzorom, iako se mogu koristiti i drugi senzori. Konstrukcija može biti stroj ili bilo koja komponenta stroja ili sustava.

Uz to, ultrazvuk se dijeli na pasivni i aktivni koji se razlikuju po načinu na koji se ultrazvučni signali generiraju i koriste za detekciju i dijagnostiku. Pasivni ultrazvuk se oslanja na prirodne ultrazvučne emisije iz opreme ili materijala te se koristi za praćenje i dijagnostiku bez potrebe za vanjskim izvorom ultrazvuka. Aktivni ultrazvuk koristi vanjski izvor ultrazvučnih valova za generiranje signala te se koristi za detaljnije ispitivanje i mjerenje svojstava materijala i struktura. U tablici ispod su prikazane različite primjene ultrazvučne analize podijeljene na aktivni i pasivni ultrazvuk.

Tablica 1. Primjene ultrazvuka podijeljene na aktivni i pasivni ultrazvuk [3]

Aktivni		Pasivni	
Puls, Eho, itd..	Snaga	Bez kontaktni - u prostoru	Kontaktni - u strukturi materijala
• NDT - kontrole bez razaranja • Debljine stjenke • Medicinske aplikacije • Ispitivanje propusnosti	• Čišćenja • Zavarivanje • Mehanička obrada • Rezanja	• Propuštanja / vakuum • Električne inspekcije	• Mehaničke inspekcije • Unutarnja propuštanja

U sklopu ovog rada koristi se pasivni ultrazvuk za otkrivanje mehaničkih grešaka na kugličnim ležajevima. Vezano za nadzor stanja korištenjem ultrazvuka, postoje dvije kategorije: online i offline nadzor. Online nadzor podrazumijeva kontinuirano praćenje opreme tijekom njenog rada, koristeći senzore i sustave za prikupljanje podataka za otkrivanje i dijagnosticiranje problema u stvarnom vremenu. Dok offline nadzor uključuje isključivanje opreme, provođenje detaljne kontrole te zatim vraćanje opreme u radni režim.

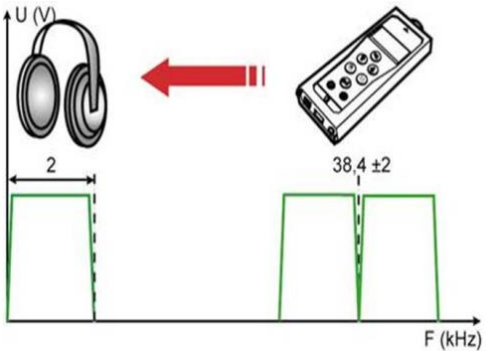
2.2. Standardi i postupci ispitivanja ultrazvukom (ISO 29821)

U ISO 29821 je opisan općeniti postupak ispitivanja ultrazvukom za potrebe nadzora stanja i dijagnostike opreme te koje fizikalne pojave generira ultrazvuk. Uz to je navedena i najčešće korištena oprema za nadzor stanja pomoću ultrazvuka. Ultrazvučni instrumenti su obično ručni, prijenosni i rade na baterije radi lakše upotrebe na terenu. Također se koriste online, neprenosivi sustavi za praćenje stanja gdje se anomalija može pojaviti i treba biti riješena na početku, a ne kada je zakazana inspekcija po planu. Preporučuje se da sustav sadrži instrument, ultrazvučne pretvarače i slušalice. Visoko se preporučuje da se izlazni signal procijeni preko slušalica kako bi se omogućila diskriminacija između različitih izvora. Ovo omogućuje održavatelju da prepozna i spriječi prikupljanje podataka loše kvalitete. Sustav treba omogućiti detekciju akustične energije koja je u zraku ili konstrukciji u rasponu iznad 20 kHz i treba prevesti ovu energiju u čujni signal koji se može vidjeti na indikatoru jačine signala i čuti preko slušalica. Kako bi se ultrazvuk mogao čuti, primjenjuje se heterodina transformacija koja omogućuje slušanje ultrazvučnih događaja na izabranoj frekvenciji. Ova metoda uključuje modulaciju ulaznog signala sa signalom lokalnog oscilatora kako bi se dobio novi signal na različitoj frekvenciji odnosno međufrekvencija Slika 3.

$$sin\theta \times sin\varphi = \frac{\cos(\theta-\varphi)-\cos(\theta+\varphi)}{2}$$

Θ – frekvencija jednog signala [Hz]

Φ – frekvencija drugog signala [Hz]



Slika 3. Heterodina transformacija [2]

Prikazana jednadžba se koristi pri množenju dviju različitih frekvencija gdje se onda stvaraju dvije nove frekvencije, jedna na frekvenciji razlike i jedna na frekvenciji zbroja. Množenje zajedno dvaju sinusoidalnih signala stvara dva kosinusna signala čije su frekvencije zbroj i razlika dva izvorna signala. Signal frekvencije 36kHz se modulira sa signalom frekvencije 38kHz i proizvodi se signal od 2kHz i 74kHz. Dobiveni signal od 2 kHz se može čuti pomoću slušalica. Jakost signala se obično prikazuje u decibelima (dB). Heterodinirani signal omogućuje održavatelju prepoznavanje relevantnog izvora zvuka i određivanje stanja koje proizvodi ultrazvuk. Signal se također može koristiti za određivanje lokacije nepotrebnog ultrazvuka koji bi mogao dovesti do lažnog očitanja. Ultrazvučni moduli za detekciju otkrivaju samo visokofrekventnu buku uzrokovanu trenjem ili turbulentnim protokom te ne reagiraju na niskofrekventne zvukove. Kod ležajeva, ultrazvuk nastaje kretanjem rotirajućih elemenata. Kako se ležaj troši, greške se formiraju na rotirajućim površinama, a kada rotirajući element naiđe na lokalno oštećenje, stvara se zvuk. Stvarne frekvencije kvara oštećenog ležaja moduliraju visokofrekventne komponente od generiranog ultrazvučnog signala. Signal nakon demodulacije ili heterodine transformacije ostavlja samo izvornu modulaciju. Na primjer, kod ležaja, ako je frekvencija kvara 48 Hz, instrument detektira ultrazvučni signal koji je moduliran frekvencijom kvara od 48 Hz. Kada se taj signal demodulira ili heterodinizira, audio signal u slušalicama ne sadrži ultrazvučni signal, već sadrži signal frekvencije kvara od 48 Hz [4].

2.3. Ultrazvučni senzori

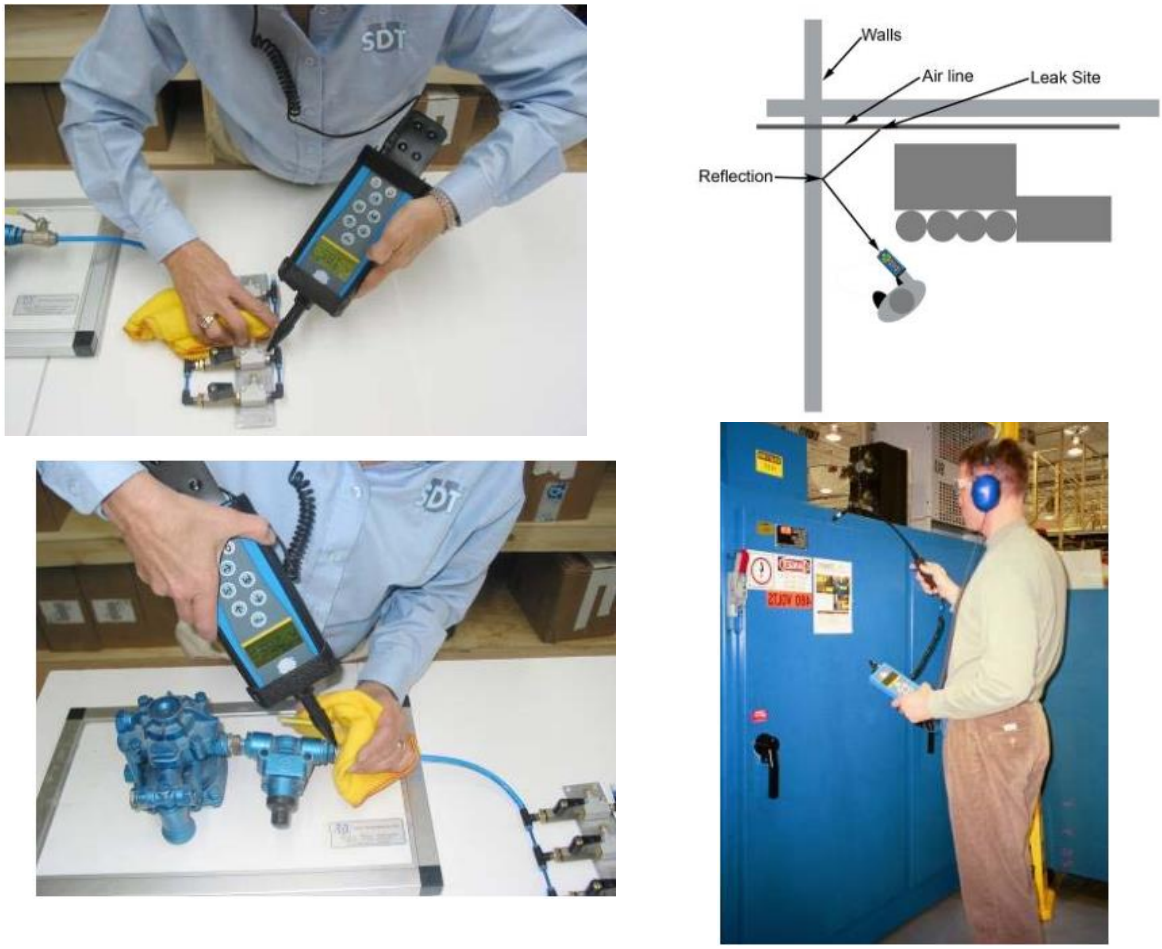
Kao što je već napomenuto, ultrazvuk koji se prenosi zrakom detektira se pomoću ultrazvučnog mikrofona koji spada u beskontaktne senzore, dok se za detekciju konstrukcijski nošenog ultrazvuka najčešće koriste kontaktni senzori.

2.3.1. Beskontaktni senzori

Beskontaktni senzori, zajedno sa ultrazvučnim instrumentom, detektiraju i analiziraju ultrazvučni signal koji se širi zrakom. Ultrazvučni instrument s fiksnim senzorima ima ograničenja u pogledu polja prijema i nije pogodan za sve primjene pa se zato koriste izmjenjivi senzori. Za ultrazvučne instrumente s izmjenjivim senzorima obično postoji izbor između dvije vrste senzora: širokokutni i parabolični. Postoje još i fleksibilni senzori koji olakšavaju detekciju zrakom propagiranog ultrazvuka u skučenim mjestima. Parabolični ultrazvučni senzori dizajnirani su za sigurno otkrivanje i identifikaciju izvora ultrazvuka na velikim udaljenostima do 90 metara. S druge strane, širokokutni senzori su namijenjeni za kratke udaljenosti, ali su zato osjetljivi te održavaju potrebnu razinu točnosti za vrijeme mjerenja [5]. Na slici ispod su prikazane spomenute tri vrste beskontaktnih senzora. Fleksibilni senzor (lijevo), širokokutni senzor (sredina) i parabolični ultrazvučni senzor (desno).

Slika 4. Beskontaktni senzori [3]

Važno je napomenuti da pri pozicioniranju ovih vrsta senzora, kako bi mjerenja bila validna i usporediva, treba imati na umu faktore koji utječu na rezultat, a to su: udaljenost od izvora, mogući vjetar (njegov smjer i brzina) te relativna vlažnost u zraku. Uz to, postoje tehnike zaštite ultrazvuka pri mjerenju sa beskontaktnim senzorima: zaklanjanje, pozicioniranje, prekrivanje i korištenje refleksije Slika 5.



Slika 5. Tehnike zaštite ultrazvuka [3]

Dakle, ove tehnike se koriste kako bi se blokirali svi izvori zvuka osim onog koji nam je potreban za prepoznavanje grešaka.

2.3.2. Kontaktni senzori

Kontaktni senzori se koriste za neinvazivno otkrivanje grešaka i anomalija na strojevima ili određenoj komponenti. Postoje ručni kontaktni senzori, senzori s magnetskim spojem ili trajno ugrađeni (navojni) senzori. Kontaktni senzor (stetoskop) najčešće se koristi kada je potrebno brzo pregledati stroj kako bi se odredilo gdje se nalazi anomalija ili stanje kvara. Također se učinkovito koristi za ulazak u uske prostore kako bi se dobio pristup do točke za praćenje stanja. Za točke koje su izvan dosega mogu se koristiti produžni kontaktni štapovi. Senzori s magnetskim spojem uklanjaju varijacije mjerenja povezane s ručnim kontaktnim senzorima. Stoga su idealni u okolnostima gdje je potrebno dugo vrijeme uzorkovanja ili gdje više

održavatelja očituje podatke na istoj točki uzorkovanja [4]. Na slici ispod su prikazani kontaktni senzori gdje su lijevo prikazani obični kontaktni senzori, a desno navojni senzor.

Slika 6. Kontaktni senzori [6]

Isto kao i kod beskontaktnih senzora, pri pozicioniranju ovih vrsta senzora, da bi mjerenja bila validna i usporediva, treba imati na umu faktore koji utječu na rezultat, a to su: isti senzor, ista lokacija, opterećenje, materijal površine, slojevi boje, brzina itd.

2.4. Mjerenje i analiza ultrazvučnog signala

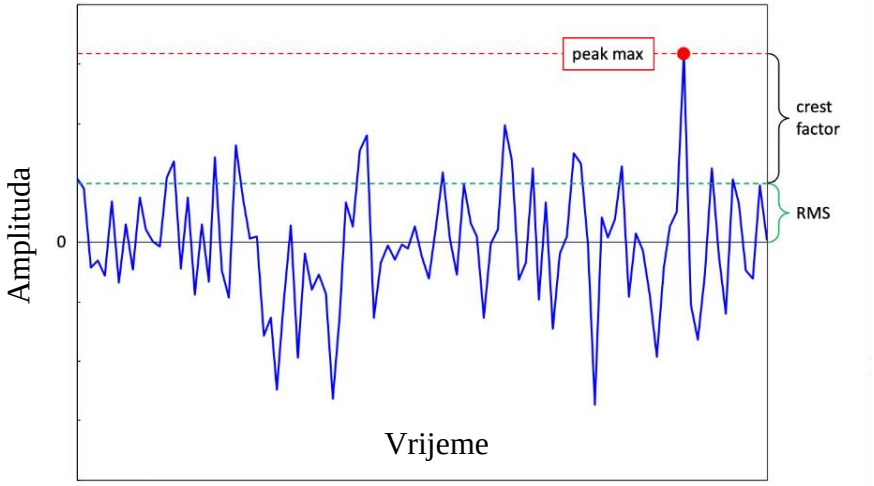
Važne vrijednosti koje opisuju izmjereni ultrazvučni signal su:

- Vršna vrijednost
- RMS vrijednost
- Crest faktor

Vršna vrijednost je najveća izmjerena trenutna amplituda u cjelokupnom trajanju uzorka te predstavlja udarce. Potrebno je koristiti veću frekvenciju za uzorkovanje signala kako bi bila bolja detekcija vrhova.

RMS (root mean square) vrijednost je ukupna snaga signala te predstavlja trenje. Dok max RMS vrijednost opisuje najveću energiju u intervalu od 0,25 sekunde.

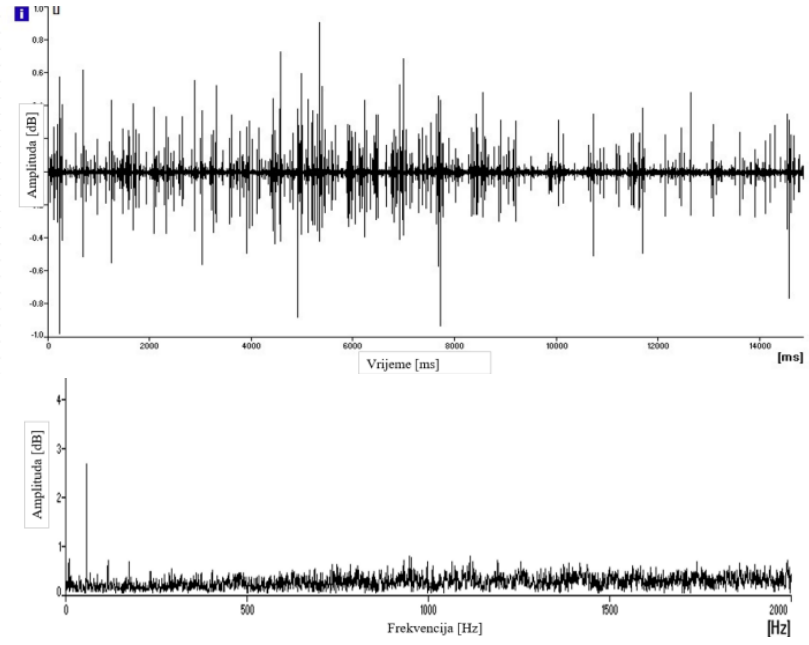
Crest faktor je omjer vršne i RMS vrijednosti Slika 7. Izražen je kao linearna vrijednost te raste, kulminira i onda pada kroz životni vijek defekta ležaja. Uz ove tri najčešće korištene značajke mogu se koristiti i ostale značajke poput faktora oblika, srednje vrijednosti, standardne devijacije, stupnja skraćenosti i asimetrije i sl.



Slika 7. Crest faktor [9]

Pomoću ovih pokazatelja stanja se određuje numerička klasifikacija stanja ležaja koja se može pratiti i na koju se mogu primijeniti alarmi. Prije su održavatelji pratili ultrazvučne podatke na temelju jedne mjere decibela uzete u proizvoljnom trenutku. Ako je mjerenje bilo izvršeno u pogrešno vrijeme, kvar je bio ili propušten ili pretjerano naglašen u odnosu na stvarno stanje. Ovi indikatori stanja omogućuju održavateljima da unaprijed postave vrijeme prikupljanja prije nego što izvrše mjerenje. Rezultat je reprezentativnija i preciznija zbirka podataka.

Ultrazvučni signali mogu biti prikupljeni na dva načina, u vremenskoj i frekvencijskoj domeni. Ispravno je da se ultrazvučni signal signal prvo uhvati u vremenskoj domeni pa se pomoću Fourierove transformacije prebaci u frekvencijsku domenu radi detaljnije analize signala.



Slika 8. Prikaz vremenske i frekvencijske domene oštećenog ležaja [10]

3. METODOLOGIJA

U ovom poglavlju objašnjena je metodologija razvoja prediktivnog modela gdje je opisano: postupak prikupljanja podataka, eksperimentalno okruženje, alat unutar *Matlab*-a, *Predictive Maintenance Toolbox* te postupak obrade podataka putem aplikacije *Diagnostic Feature Designer*.

3.1. Prikupljanje podataka

U industriji se koristi nekoliko tehnika za prikupljanje podataka. Najnoviji ultrazvučni detektori postali su mnogo sofisticiraniji i evoluirali su od starijih uređaja za slušanje s ručno pisanom dokumentacijom do sustava koji mogu pohranjivati prikupljene podatke, snimati zvučne uzorke i analizirati podatke putem softvera za upravljanje podacima. Najčešća metoda za prepoznavanje anomalija je komparativna metoda. Prema ISO 29821, to je najbolja metoda za korištenje kada nisu postavljene osnovne vrijednosti te kada je potrebno procijeniti niz točaka na stroju bez prethodnih kriterija procjene. Kao i kod svih tehnologija, razina povjerenja u dobivene informacije ovisi o primijenjenoj metodi detekcije, korištenoj opremi te obuci i iskustvu održavatelja. Komparativni ultrazvuk može biti kvantitativan ili kvalitativan. Mnoge primjene ne zahtijevaju kvantitativne podatke za praćenje stanja komponenata stroja. U slučaju curenja komprimiranog zraka i električnih pražnjenja, kvalitativne tehnike su obično preferirane metode jer se u tim slučajevima ne proizvodi ultrazvuk ako ne postoji. Tema ovog rada je bazirana na kvantitativnoj metodi pa je ona objašnjena u nastavku.

3.1.1. Kvantitativna metoda

Kvantitativna metoda je najčešće korištena metoda za mnoge ultrazvučne primjene. Ova metoda koristi jakost signala, izraženu u decibelima, kako bi se odredila ozbiljnost stanja komponente. Ovisno o vrsti kvara, održavatelj može zabilježiti podatke kao vrijednost decibela te zabilježiti zvučne uzorke. Može analizirati podatke i zabilježene zvučne uzorke koristeći tehnike vremenske i frekvencijske domene. Dobivena vrijednost decibela uspoređuje se s vrijednošću slične komponente ili osnovnom vrijednošću originalne komponente [4]. Kada dodatni pregledi pokažu povećanje vrijednosti decibela ili promjenu karakteristika zvuka, osnovne vrijednosti mogu se koristiti za usporedbu. To je korisno za prepoznavanje težih oštećenja prije nego što zahtijevaju veće održavanje ili postanu katastrofalni. Metoda osnovnih vrijednosti može uključivati vrijednost decibela i osnovne zvučne uzorke. Prednost osnovnog

zvučnog uzorka je mogućnost pregleda i analize bilo kakvih promjena na predmetnoj opremi temeljenih na spektralnim i vremenskim prikazima koje možda nisu očite samo s vrijednošću decibela. Kada se ultrazvučne inspekcije provode u manje idealnim uvjetima s značajnom pozadinskom bukom, povjerenje u dobivene informacije ovisi o obuci i iskustvu održavatelja te o primijenjenoj metodi detekcije. Vještine i stručnost održavatelja koji provodi mjerenja i analizira podatke ključni su za učinkovitu primjenu ultrazvuka [4].

Nakon više mjerenja na istom uređaju u istim radnim uvjetima, zabilježeni podaci mogu se koristiti za postavljanje parametara za praćenje trendova i kao pomoć u predviđanju kvara te komponente. Primjer kriterija ozbiljnosti temeljenih na povećanju vrijednosti decibela iznad utvrđene referentne vrijednosti ili osnovne vrijednosti za visokobrzinske ležajeve rotirajuće opreme je sljedeći [4]:

Faza prije kvara - 8 dB: Ovo je najranija faza kvara. Ležajevi mogu imati mikropukotine ili mikroskopske mrlje koje nisu vidljive golim okom. Ovo također može ukazivati na potrebu za podmazivanjem ležajeva.

Faza kvara - 16 dB: U ovoj fazi razvijaju se vidljivi nedostaci zajedno s naglim porastom akustične energije. U ovoj fazi ležajeve treba zamijeniti ili provoditi češće preglede.

Katastrofalna faza - katastrofalni kvar - 35 dB do 50 dB: U ovoj fazi je neizbježan brzi kvar. Razina akustičnog zvuka je toliko intenzivna da je čujna, a temperatura ležajeva može toliko porasti da se može izmjeriti. Ovo je vrlo opasna faza jer promjene u zračnostima i tolerancijama ležajeva mogu uzrokovati dodatno trenje unutar stroja, potencijalno oštećujući druge komponente.

Kada ultrazvučni instrument detektira odstupanja od osnovne vrijednosti prethodnog očitanja ili komparativne razlike, ta odstupanja se zabilježe. Podaci o vrijednosti decibela ili heterodinizirane ultrazvučne anomalije, ili oboje, se bilježe i analiziraju radi određivanja ozbiljnosti i poduzimanja korektivnih mjera. Vrlo je korisno korištenje analize vremenske i frekvencijske domene za određivanje ozbiljnosti anomalije i za prikaz stanja stroja kao zvučne slike. Ultrazvučna očitanja i zabilježeni heterodinizirani podatci se prikupljaju s lokacija odabranih tako da se minimiziraju pogreške uzrokovane ultrazvukom iz drugih izvora, poput zvučnih refleksija. Također se pazi da se očitanja ne uzimaju tijekom rada strojeva kada se proizvodi konkurentni ultrazvuk. Korištena oprema za prikupljanje podataka je opisana u nastavku.

3.2. Eksperimentalno okruženje

Za proučavanje grešaka na ležajevima korišten je simulator kvarova ležaja. Na simulator su bili postavljeni tri ležaja s oštećenjima i jedan ležaj u normalnom stanju. Ultrazvučni podaci ležajeva su prikupljeni pomoću instrumenta SDT 340 i navojnog kontaktnog senzora te nakon toga prebačeni u software pod nazivom Ultranalysis Suite 3. Oprema je detaljnije opisana u nastavku.

3.2.1. Simulator kvarova ležaja

Simulator kvarova ležaja je napravljen za proučavanje kvarova ležajeva i neuravnoteženosti u kontroliranim uvjetima. Simulator ima promjenjivu brzinu te se može koristiti za generiranje svake vrste kvara pojedinačno ili u kombinaciji, pružajući stabilnu platformu za proučavanje. Budući da su problemi povezani s ležajevima vrlo česti, ključno je temeljito razumijevanje pripadajućih znakova kvara koje se javljaju u različitim radnim uvjetima. Isto vrijedi i za neuravnoteženost, gdje će pravilno balansiran stroj uštedjeti tvornici vrijeme zastoja stroja, zamjenske dijelove, zalihe i potrošnju energije. Simulator pruža osnovnu postavu za izvođenje eksperimenata te za kontrolu vibracijskih i ultrazvučnih znakova neuravnoteženosti i kvarova ležajeva [7].

Slika 9. Simulator kvarova ležaja [7]

Značajke simulatora [7]:

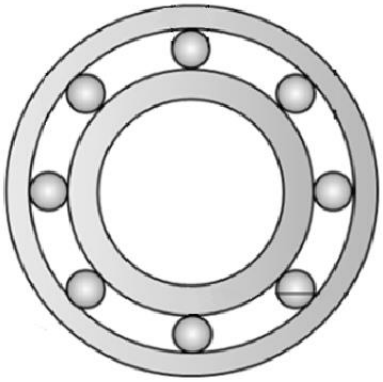
- Prijenosni simulator vibracija za ravnotežu i ležaj
- Koristi se za balansiranje u više ravnina sa središnjim ili visoko postavljenim rotorima
- Može se postaviti tako da pokazuje frekvencije kvara ležaja ovisno o brzini vrtnje vratila
- Mogu se razviti tehnike obrade signala za prepoznavanje frekvencija kvarova ležajeva u prisutnosti nedostataka, pri višestrukim brzinama vratila, bez korištenja spektra visoke rezolucije
- Koristi se za prepoznavanje spektra vibracija različitih grešaka ležaja

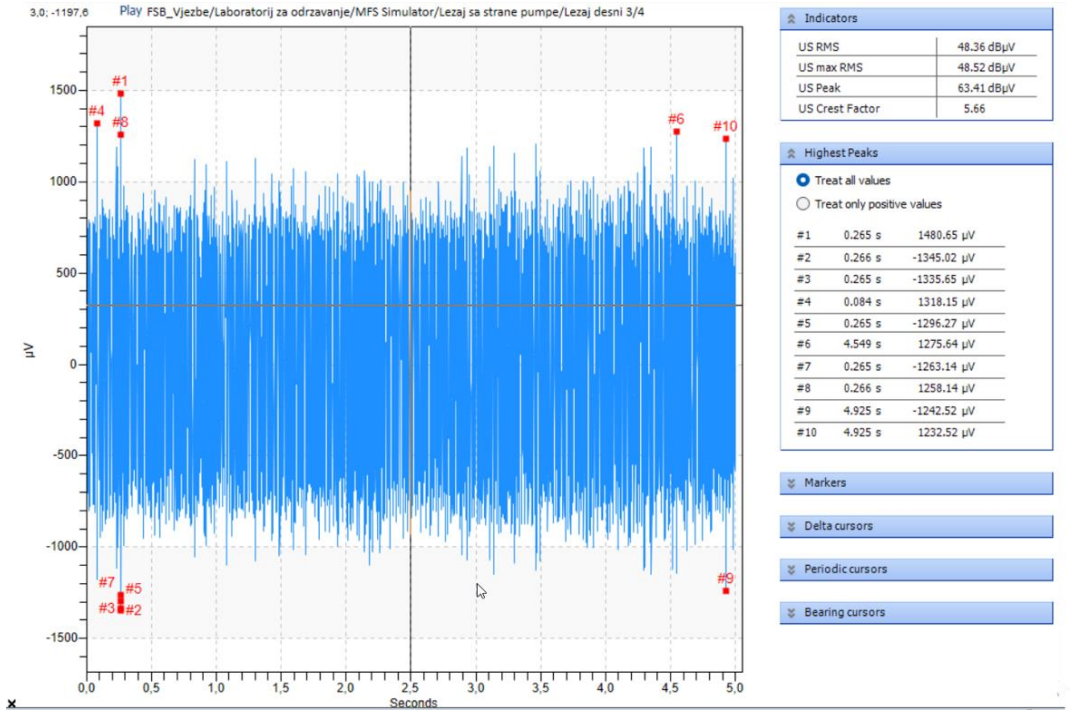
3.2.2. Kuglični ležejavi ER12

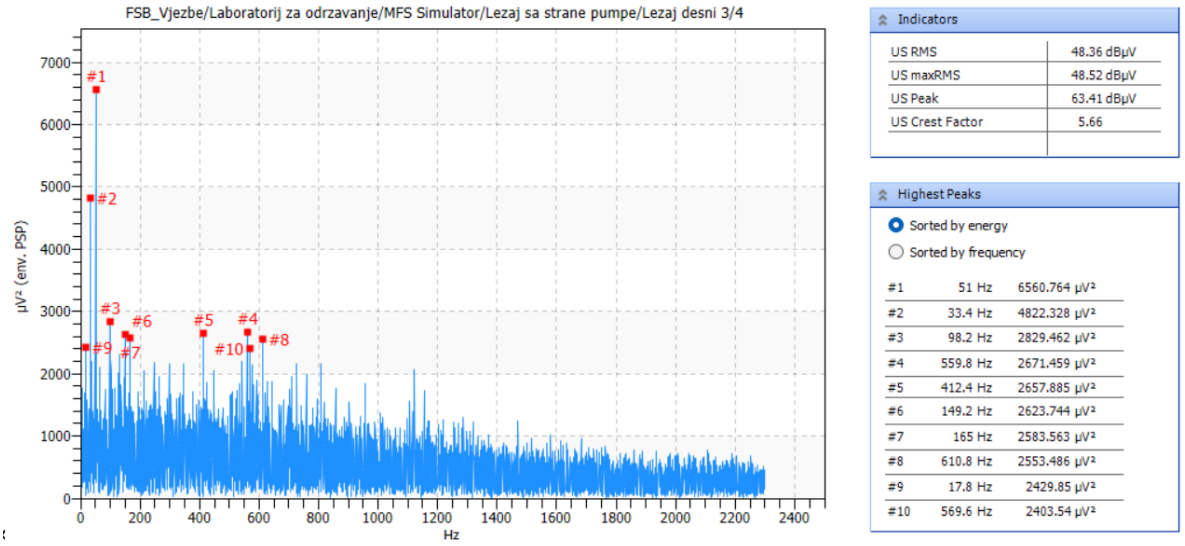
Kao što je već napomenuto, podaci su prikupljeni s tri kuglična ležaja s oštećenjima i s jednog koji je u normalnom stanju radi usporedbe. Najčešća oštećenja koja se javljaju na ležaju su oštećenje vanjskog prstena, unutarnjeg prstena i kuglice pa su zato testirane te tri vrste. Korišten je tip ležaja ER12 Slika 10.

Slika 10. Kuglični ležaj ER12

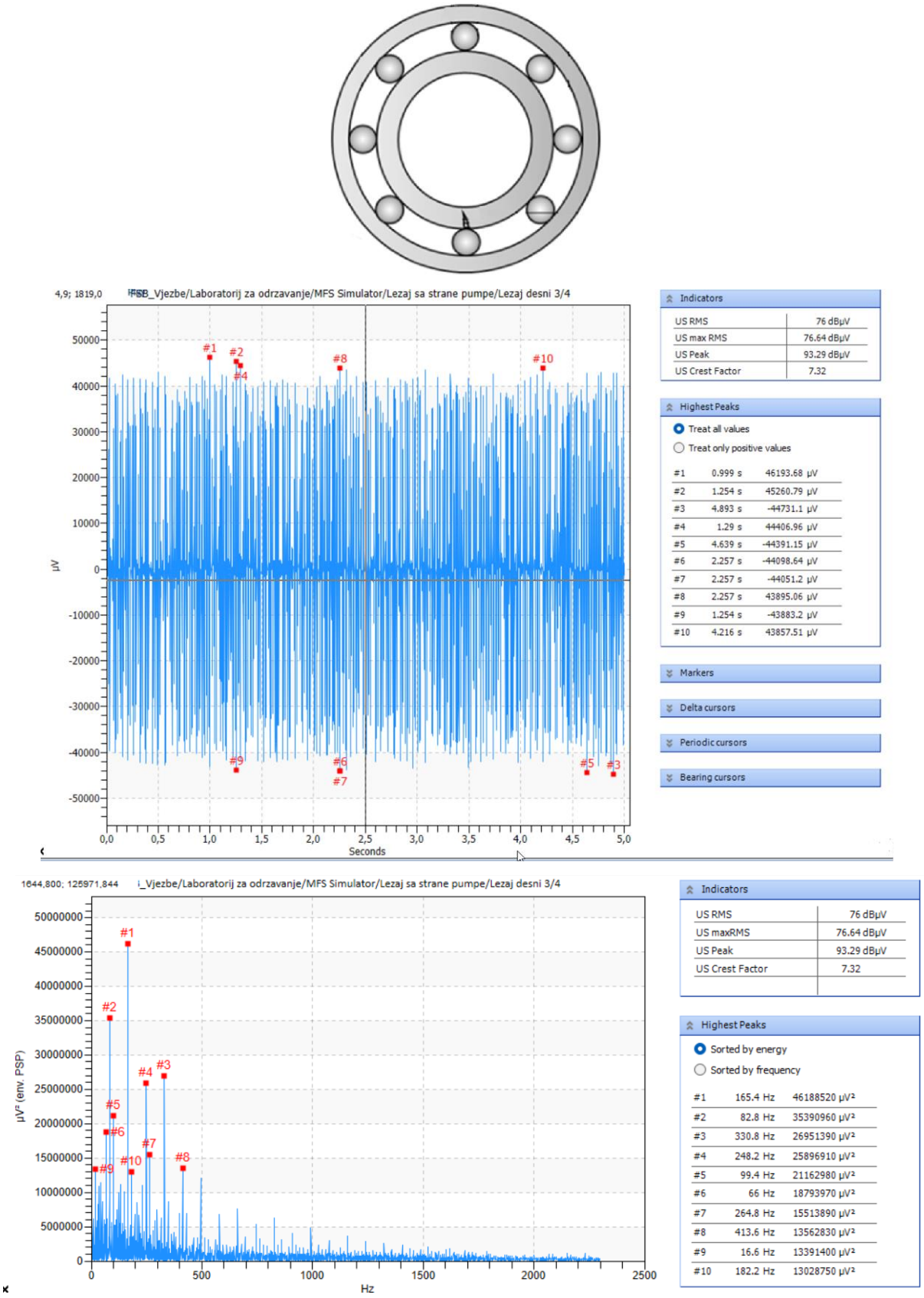
U nastavku su prikazani prikupljeni podaci za svaki ležaj u vremenskoj i frekvencijskoj domeni, uzeti iz programa Ultranalysis Suite 3. Uz to su i izvučeni spomenuti indikatori stanja koji opisuju izmjereni ultrazvučni signal.



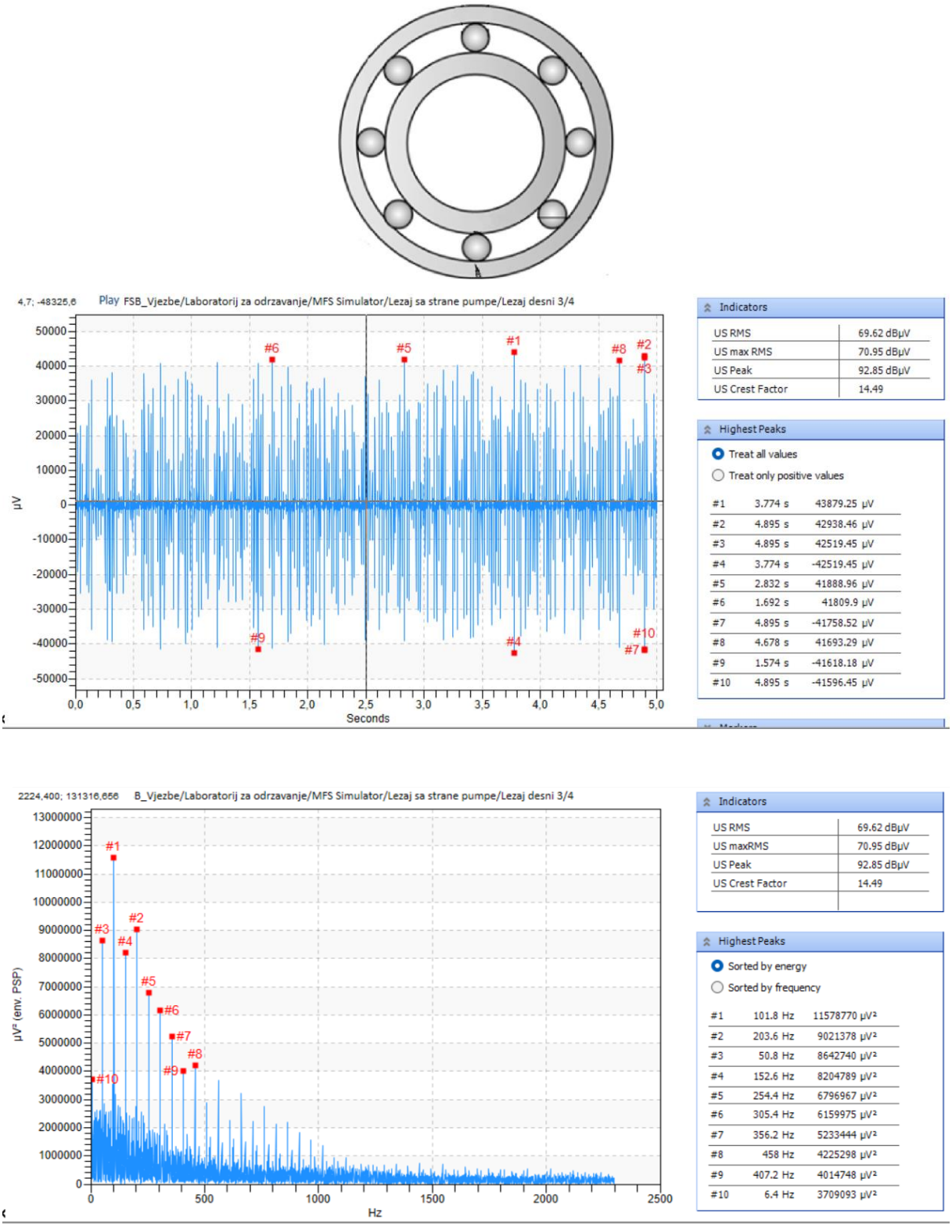




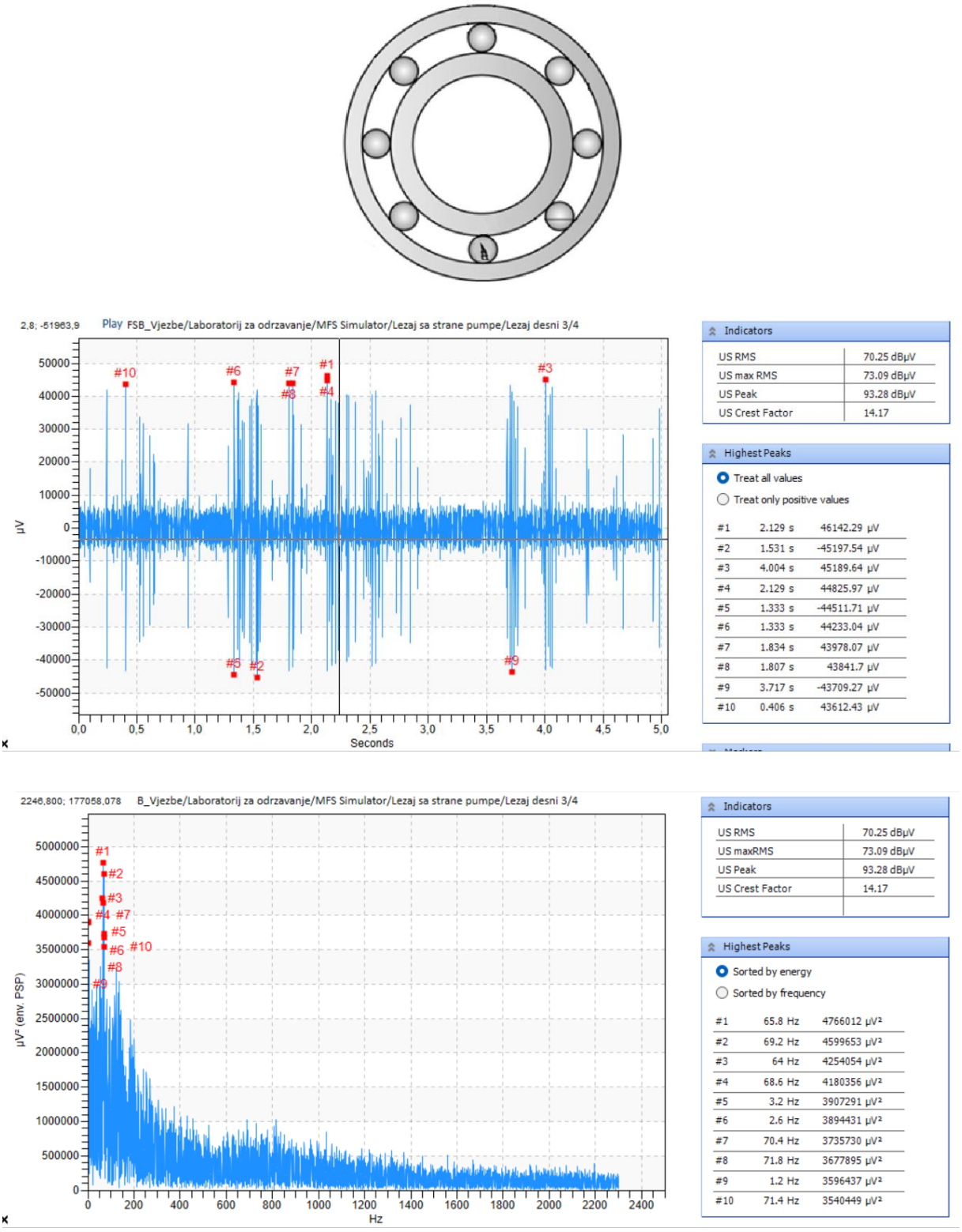
Slika 11. Podaci o ležaju u normalnom stanju prikazani u vremenskoj i frekvencijskoj domeni



Slika 12. Podaci o ležaju s oštećenjem unutarnjeg prstena prikazani u vremenskoj i frekvencijskoj domeni



Slika 13. Podaci o ležaju s oštećenjem vanjskog prstena prikazani u vremenskoj i frekvencijskoj domeni



Slika 14. Podaci o ležaju s oštećenjem kuglice prikazani u vremenskoj i frekvencijskoj domeni

Vidljivo je sa slika da postoje razlike u amplitudama između stanja ležajeva, a i iz samih vrijednosti pokazatelja stanja.

3.2.3. SDT 340

SDT340 je instrument za nadzor stanja, dizajniran za inspekciju i prikupljanje podataka sa svih potencijalno neispravnih industrijskih strojeva prisutnih u nekom pogonu te pomoću njega su prikupljeni svi podaci sa simulatora kvarova ležaja.

Slika 15. SDT 340 instrument [8]

Značajke SDT 340 instrumenta [8]:

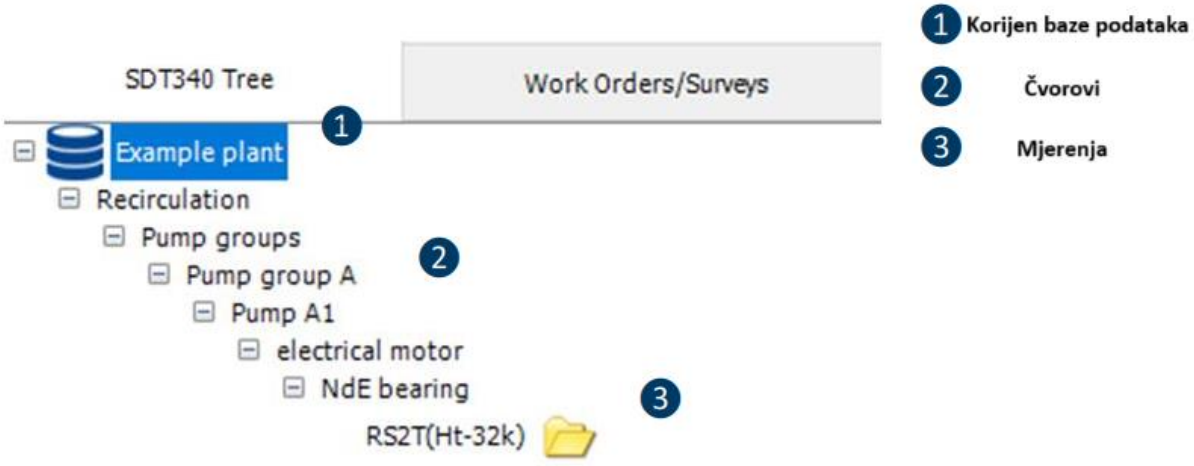
- Može snimiti vremenske podatke u trajanju do 10 minuta s poboljšanim signalnim podacima
- Podaci se mogu spremiti za praćenje trendova i alarmiranje prilikom budućih prikupljanja podataka
- Moguća analiza signala pomoću spomenutih pokazatelja stanja (vršna vrijednost, RMS vrijednost i crest faktor)
- Moguć je prikaz vremenske i frekvencijske domene na ekranu
- Moguće je zapisivanje najvećih vrijednosti pokazatelja stanja

- Može se koristiti sa bluetooth slušalicama
- Kompatibilan sa beskontaktnim i kontaktnim senzorima

Nakon što se prikupe podaci, prebacuju se u software *Ultranalysis Suite 3* (*UAS3*) radi organizacije.

3.2.4. Ultranalysis Suite 3

UAS3 se prvenstveno koristi kao baza podataka za upravljanje prikupljenim podacima, obradom i analizom podataka. Općenito, baza podataka je integrirana zbirka logički povezanih zapisa ili datoteka, konsolidirana u zajednički spremnik koji pruža podatke za jednu ili više različitih namjena. Koristi se za pohranu i organizaciju informacija na način koji olakšava njihovo pronalaženje. *UAS3* koristi hijerarhijski model baze podataka u kojem su podaci organizirani u strukturu nalik na stablo. U ovoj vrsti strukture, naziv baze podataka, ujedno i korijen baze podataka se nalazi na vrhu strukture stabla, dok su mjerenja, također nazvana "lišće", na njegovom kraju. Mjerenja predstavljaju kombinaciju odabira senzora i postavki mjerenja. Grane između korijena baze podataka i kategorija mjerenja nazivaju se čvorovi Slika 16.

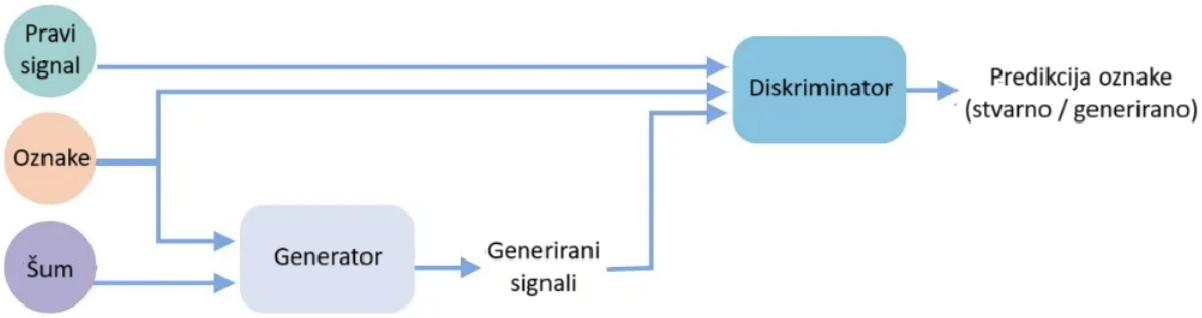


Slika 16. Struktura stabla u programu Ultranalysis Suite 3 [11]

U ovom radu je program *UAS3* korišten samo za spremanje prikupljenih podataka, a za analizu je korišten *Matlab,* odnosno alat *Predictive Maintenance Toolbox* koji je objašnjen u nastavku.

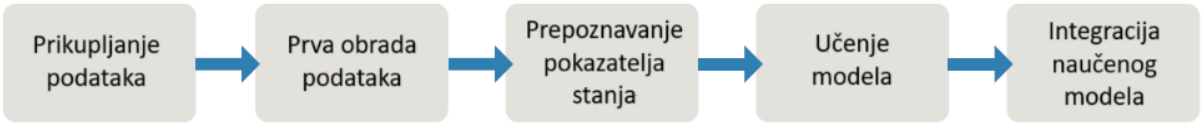
3.3. Matlab – Predictive Maintenance Toolbox

Alat za prediktivno održavanje (engl. *Predictive Maintenance Toolbox, PMT*) kao dio programa *Matlab* služi za predviđanje kvara opreme i za procjenu njenog vijeka trajanja. Ideja alata je omogućiti i poboljšati održavanje pravovremenim djelovanjem radi sprječavanja zastoja i smanjenja gubitaka. Uglavnom se koristi kod praćenja stanja motora, mjenjača, ležajeva, baterija, ali i drugih. S alatom je moguće upravljati podacima sa senzora (signalima) i izravno pratiti stanje opreme. Također, alat se koristi za organizaciju i analizu ulaznih podataka spremljenih u lokalnim datotekama ili u oblaku. Uz programiranje je moguće izdvojiti potrebne pokazatelje stanja, klasificirati ih, rangirati i onda izvesti potrebne zaključke. To uključuje filtriranje i pretprocesiranje signala senzora, izdvajanje značajki vremenske i frekvencijske domene te višerazredno klasificiranje za otkrivanje različitih kombinacija grešaka [12]. Temelji se na umjetnoj inteligenciji, točnije konvolucijskim neuronskim mrežama. Signali sa senzora se generiraju uz pomoć konvolucijskih neuronskih mreža zvane GAN (engl. *Generative adversarial networks*) koje se sastoje od tzv. generatora i diskriminatora. Generator stvara nove uzorke koji nalikuju danom skupu podataka, a diskriminator pokušava razlikovati generirane uzorke od pravih uzoraka. Generator se ažurira na temelju povratnih informacija diskriminatora, s ciljem generiranja uzoraka koje je sve teže razlikovati od pravih. Rezultat naučene mreže je generiranje realnih uzoraka koji su slični izvornom skupu podataka. Ovaj način je povoljan u slučajevima kada su računalne simulacije skupe. Unutar PMT-a se koriste CGAN (engl. *Conditional GAN*), a predstavljaju jednodimenzionalne konvolucijske neuronske mreže. One generiraju podatke koji pripadaju određenim kategorijama te koriste signale, koji su simulirani, kao stvarne podatke koji se koriste za treniranje ove mreže [13] Slika 17.



Slika 17. CGAN [14]

Općenito, uobičajeni proces izrade algoritma za preventivno održavanje je prikazan na slici ispod.



Slika 18. Proces izrade algoritma za preventivno održavanje [15]

Proces započinje prikupljanjem podataka sa stroja u različitim uvjetima rada i stanjima kvarova. Sirovi podaci se zatim obrađuju i formiraju na način da se mogu izdvojiti pokazatelji stanja, odnosno značajke. Dalje, te izdvojene značajke postaju ulazni podaci za trening modela strojnog učenja. Važno je da je skup značajki kvalitetan kako bi program mogao lakše odrediti greške u radu opreme te trenutno stanje iste pri unosu novih podataka s opreme u program. Ako značajke nisu karakteristične, program može netočno procijeniti stanje opreme. Može se napraviti model koji prepoznaje kada je oprema zdrava i koji može identificirati kvarove. Teži je drugi slučaj, odnosno identifikacija kvara jer se može sastojati od kombinacije kvarova kojih može biti više i sličnog karaktera ponašanja.

3.3.1. Izdvajanje značajki

Ključni korak u razvoju algoritma za prediktivno održavanje je identificiranje pokazatelja stanja jer oni pomažu razlikovati zdrav rad od kvarova. Oni se izvlače iz prethodno obrađenih podataka i koriste se za klasifikaciju kvarova i procjenu preostalog radnog vijeka (engl. *remaining useful life, RUL*). Kada se gledaju sirovi, neobrađeni podaci sa senzora nekog stroja, teško je razlikovati zdrav rad od stanja kvara, ali s pokazateljima stanja je to olakšano Slika 19.

Slika 19. Korištenje pokazatelja stanja za detekciju kvarova [15]

Pokazatelji stanja se mogu izvući iz podataka pomoću spomenutih značajki vremenske, frekvencijske i vremensko-frekvencijske domene. Značajke vremenske domene uključuju srednju vrijednost, standardnu devijaciju, koeficijent asimetrije, koeficijent spljoštenosti i dr.,

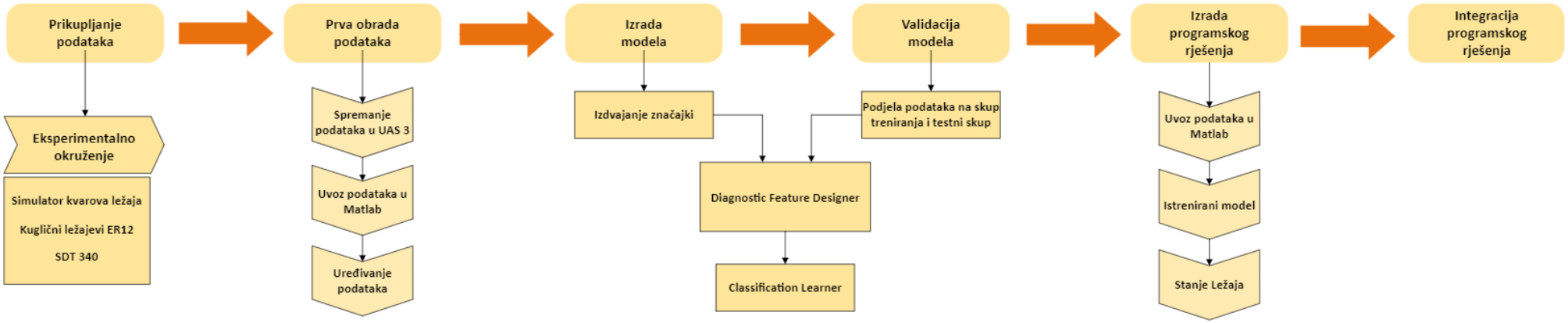
dok se u frekvencijskoj domeni mogu izolirati različiti izvori vibracija. Izdvajanje značajki u vremensko-frekvencijskoj domeni karakterizira promjene u spektralnom sadržaju signala tijekom vremena poput spektralnog koeficijenta spljoštenosti i spektralne entropije. Potrebno je napomenuti da kada se istražuju značajke, nije samo bitno pronaći različite grupe podataka, već je poželjno da one međusobno budu što udaljenije kako bi se osigurala visoka točnost istreniranog modela pri prepoznavanju novih podataka. Izdvajanje značajki je pogodno napraviti preko aplikacije, unutar *Matlab*-a, zvane *Diagnostic Feature Designer*-a.

3.3.2. Diagnostic Feature Designer

Aplikacija omogućuje izdvajanje značajki u okviru prediktivnog održavanja koristeći višenamjensko grafičko sučelje. Značajke se mogu međusobno uspoređivati i odrediti koje su najprikladnije za razlikovanje podataka iz normalnih stanja od onih s kvarom. Najučinkovitije značajke na kraju postaju indikatori stanja za dijagnostiku kvarova i prognozu. Korištenjem ove aplikacije mogu se [16]:

- Uvesti prikupljeni podaci iz pojedinačnih datoteka, skupnih datoteka ili podatkovnih spremišta koje referencira datoteke izvan aplikacije.
- Interaktivno vizualizirati podaci za iscrtavanje varijabli unutar skupa koje uvozite ili izračunate unutar aplikacije.
- Generirati značajke iz varijabli i može se vizualizirati njihova učinkovitost koristeći histogram.
- Rangirati značajke kako bi se odredilo koje su najbolje za prepoznavanje različitih stanja u podacima.
- Izvesti najbolje rangirane značajke izravno u *Classification Learner* za dublji uvid u učinkovitost značajki, za učenje algoritama, izradu samog modela te validaciju modela.

Faze razvoja modela za prediktivno održavanje rotacijske opreme su prikazane u nastavku, a detaljno opisane u sljedećem poglavlju.



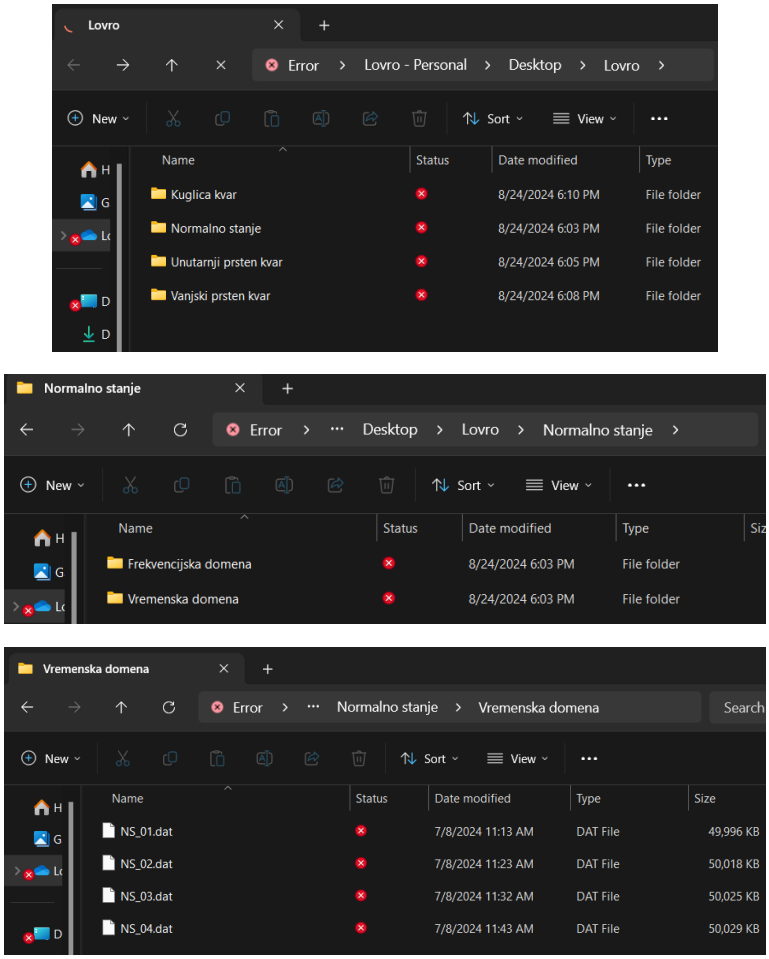
- **1. faza:** Prikupljanje podataka u eksperimentalnom okruženju
- **2. faza:** Prva obrada podataka u *Matlab* programu
- **3. faza:** Izrada modela pomoću aplikacije *Diagnostic Feature Designer*
- **4. faza:** Validacija modela pomoću aplikacije *Diagnostic Feature Designer*
- **5. faza:** Izrada kompletnog programskog rješenja za prediktivno održavanje rotacijske opreme temeljenog na ultrazvučnoj analizi
- **6. faza:** Integracija programskog rješenja

4. RAZVOJ PREDIKTIVNOG MODELA

U ovom poglavlju je opisan postupak obrade prikupljenih podataka, postupak izdvajanja značajki te na koji način se trenira i validira model koristeći program *Matlab*.

4.1. Obrada podataka

Iz programa *UAS3* su spremljeni podaci u obliku tekstualnih datoteka (.dat), a sadrže vrijeme i amplitudu signala za vremensku i frekvencijsku domenu. Prije uvoza podataka u *Matlab* potrebno je urediti same direktorije radi lakšeg snalaženja Slika 20.



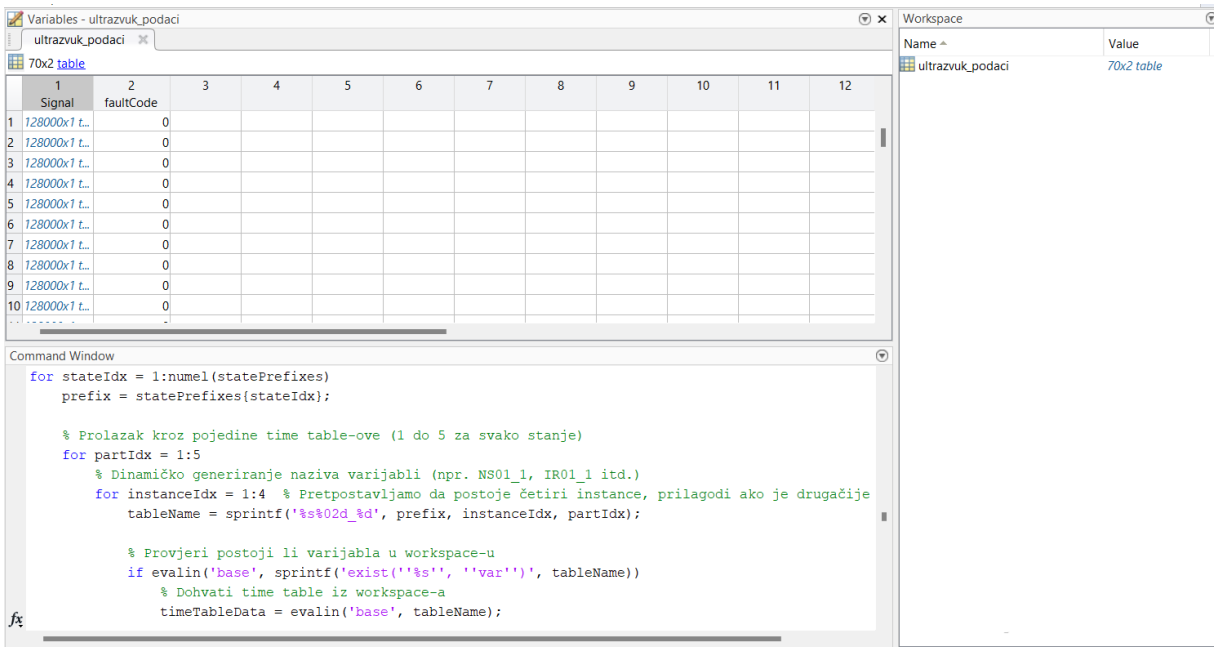
Slika 20. Uređene mape za skupove podataka svakog stanja ležaja

Sada se podaci ubacuju u *Matlab* i dalje uređuju. Napravljen je program za učitavanje podataka, izbacivanje nepotrebnih stupaca, podjelu podataka na 5 dijelova, dodavanje vremenskog stupca s frekvencijom uzorkovanja od 128 kHz, prebacivanje iz podatkovne strukture *table* u

podatkovnu strukturu *timetable* te dodavanje *faultCode* stupca, odnosno dodani su različiti brojevi u ovisnosti o stanju ležaja:

- 0 – normalno stanje
- 1 – kvar na unutarnjem prstenu ležaja
- 2 – kvar na vanjskom prstenu ležaja
- 3 – kvar na kuglici ležaja.

Ovaj program je zajedno s ostalim programskim kodom priložen na kraju diplomskog radu. Zatim je razvijen program za spajanje svih vremenskih tablica u jednu tablicu koje su u prvom stupcu, a drugi stupac je *faultCode* stupac. Pri izvršavanju programskog koda kreira se 70 x 2 tablica svih ultrazvučnih podataka Slika 21.

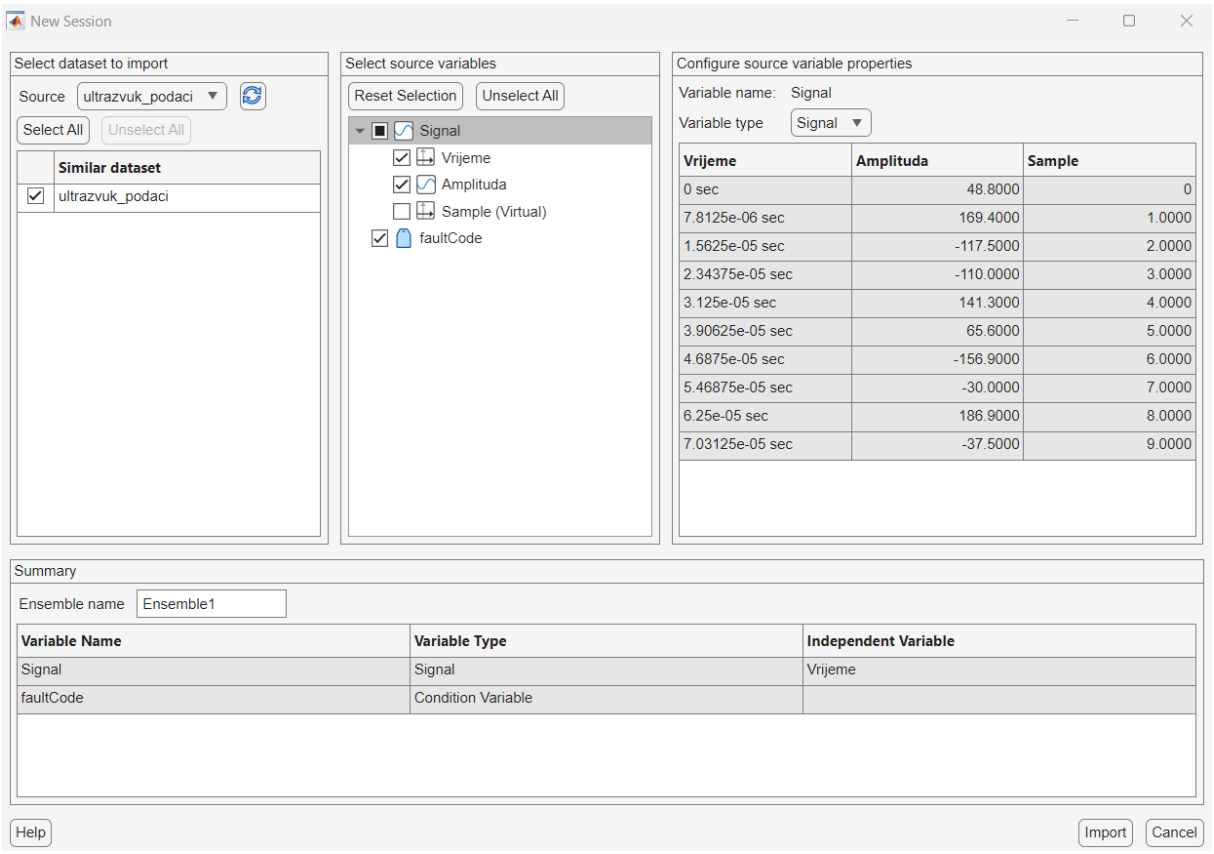


Slika 21. Pripremljeni podaci za uvoz u *Diagnostic Feature Designer*

Na ovaj način su uređeni podaci za svako stanje ležaja i sljedeći korak je izdvajanje značajki preko aplikacije *Diagnostic Feature Designer*-a.

4.2. Izdvajanje značajki pomoću aplikacije Diagnostic Feature Designer

Za jednostavno izdvajanje značajki koristi se već spomenuta aplikacija *Diagnostic Feature Designer* koji se nalazi u programu *Matlab* među aplikacijama koje spadaju pod vrstu *Control System Design*. Otvara se *New Session* i odabere skup podataka iz radnog prostora, u ovom slučaju *ultrazvuk_podaci*. Na Slika 22 se može vidjeti prozor nove sesije u kojem je potrebno kategorizirati unesene podatke. Prikupljene amplitude u mikrovoltima spadaju pod varijablu *Signal*, dok *faultCode* označuje samo prisutnost i vrstu kvara što je kategorizirano kao *Condition Variable*.

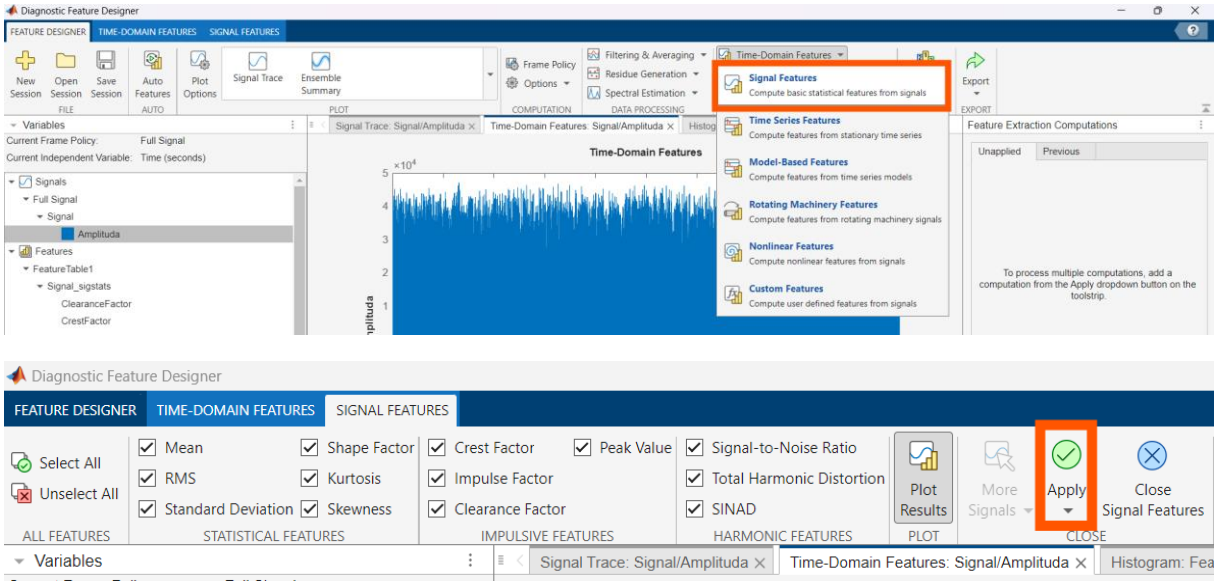


Slika 22. Kategorizacija unesenih podataka

Uneseni podaci se sada mogu grafički prikazati u vremenskoj domeni. Pomoću *Signal Trace* funkcije se mogu prikazati svi signali. Moguće ih je podijeliti po vrsti kvara, odabirom *Group By: faultCode*.

Slika 23. Grafički prikaz svih podataka razvrstanih po vrsti kvara

Sa slika je vidljivo da postoje razlike u amplitudama signala između stanja ležaja pa se može zaključiti da je visoka vjerojatnost da će modeliranje uspjeti. Da se dobije detaljniji uvid u ponašanje pojedine kategorije kvara, signalne značajke koje aplikacija nudi se mogu odvojiti u vremenskoj domeni pod izbornikom i odabirom na *Time-Domain Features*. Izabrane su sve značajke signala Slika 24.



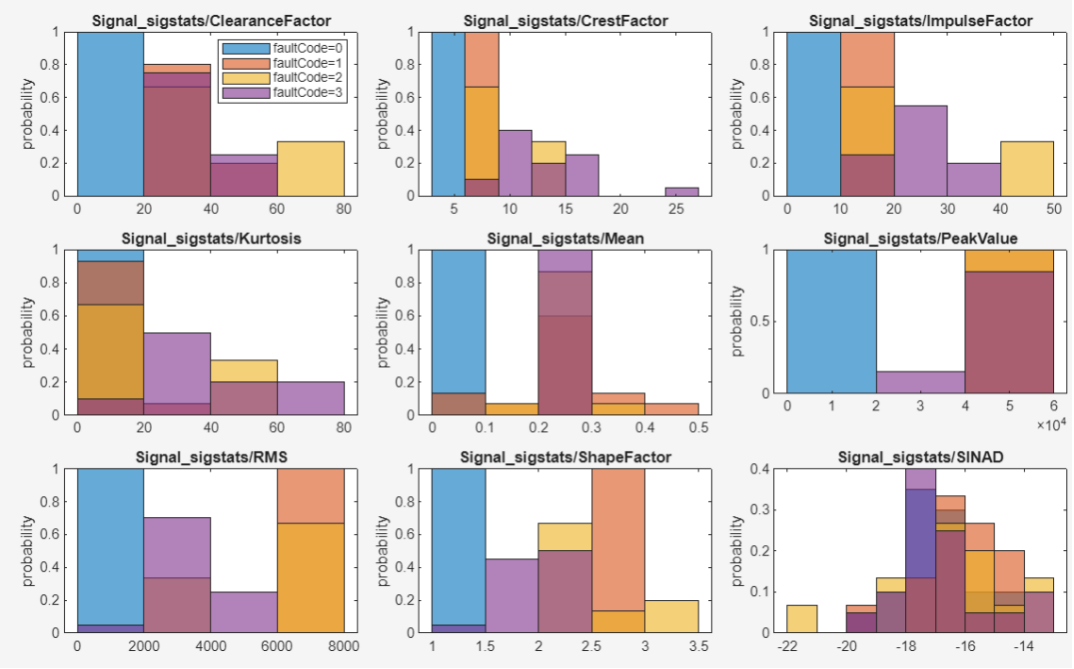
Slika 24. Odabir značajki signala u vremenskoj domeni

Izdvojene značajke se automatski spremaju u tablicu pod imenom *FeatureTable1* u ovom slučaju i prikazana je na Slika 25. Prvi stupac označava kodove kvarova, a svaki sljedeći stupac prikazuje podatke pojedine značajke. Dodane značajke su iz vremenske domene za amplitudu signala.

	faultCode	Signal_sigstats/ClearanceFactor	Signal_sigstats/CrestFactor	Signal_sigstats/ImpulseFactor	Signal_sigstats/Kurto
1	0	8.4584	5.6914	7.1594	
2	0	6.7744	4.5432	5.7223	
3	0	8.0718	5.4043	6.8201	
4	0	6.3438	4.2776	5.3692	
5	0	6.4556	4.3303	5.4519	
6	0	6.5105	4.3787	5.5050	
7	0	6.5666	4.4095	5.5487	
8	0	7.3047	4.9116	6.1796	
9	0	6.7255	4.5332	5.6915	
10	0	6.1889	4.1871	5.2428	
11	0	6.7045	4.5082	5.6696	
12	0	7.0363	4.7529	5.9615	
13	0	6.7843	4.5541	5.7353	
14	0	6.3294	4.2520	5.3511	
15	0	6.5805	4.4397	5.5723	
16	0	7.4323	5.0083	6.2881	
17	0	7.2686	4.8717	6.1405	
18	0	6.5319	4.4032	5.5289	
19	0	7.0372	4.7377	5.9536	
20	0	6.3800	4.2949	5.3972	
21	1	40.4538	7.2652	19.1431	18
22	1	37.9988	7.1043	18.2635	17

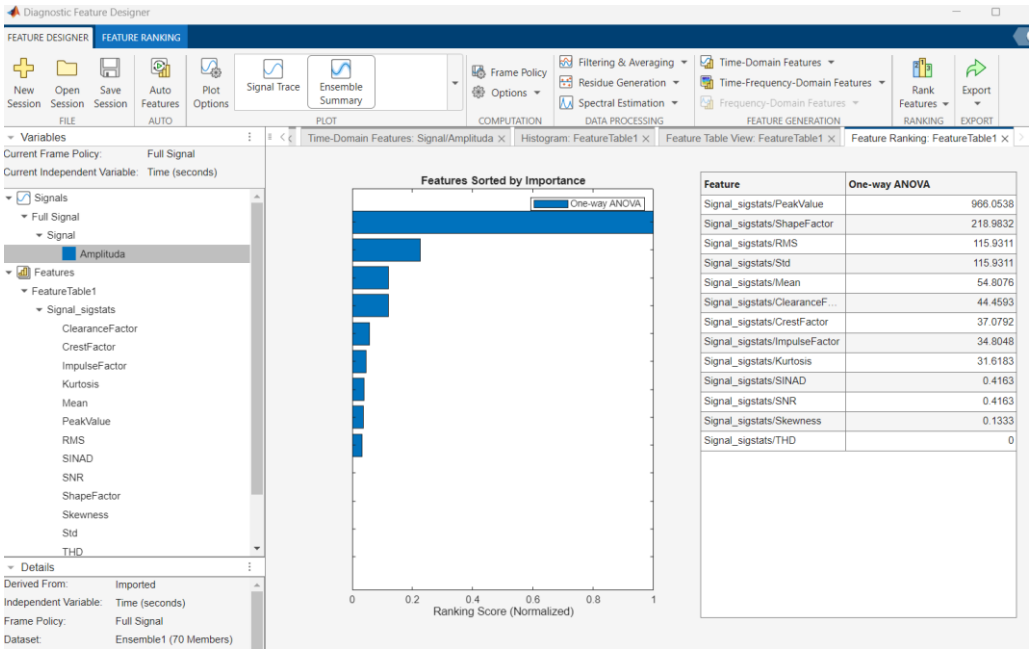
Slika 25. Tablica vrijednosti izdvojenih značajki

Distribuciju vrijednosti značajki moguće je prikazati po vrsti kvara kroz histogram Slika 26. Kvarovi su odvojeni različitom bojom, a visina stupaca predstavlja vjerojatnost ili učestalost pojavljivanja određene vrijednosti značajke unutar te kategorije. Preko histograma se mogu usporediti kako se značajke raspoređuju između različitih stanja kvarova. Na primjer, ako *CrestFactor* ima više vrijednosti u plavoj kategoriji (normalno stanje), to može značiti da u tom stanju vršni faktori nisu toliko izraženi kao u kvarnim stanjima. Na slici ispod su prikazani histogrami za sljedeće značajke: clearance factor (vršna vrijednost podijeljena s kvadratom srednje vrijednosti kvadratnih korijena apsolutnih amplituda), crest factor, impulse factor (vršna vrijednost podijeljena sa srednjom vrijednosti amplituda), koeficijent spljoštenosti (mjera za debljinu krajeva raspodjele vjerojatnosti), srednja vrijednost, vršna vrijednost, RMS vrijednost, shape factor (RMS podijeljen sa srednjom vrijednošću apsolutne vrijednosti amplituda) te SINAD (omjer ukupne snage signala i ukupne snage šuma i izobličenja).



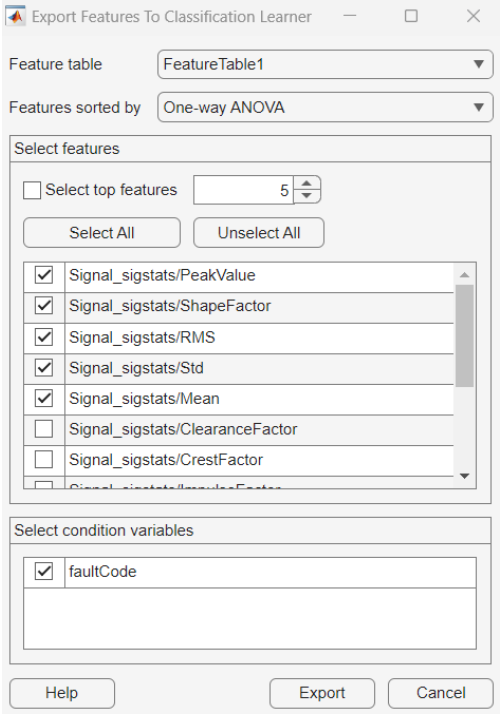
Slika 26. Histogrami značajki iz vremenske domene

Vidljivo je da se ne mogu detaljno procijeniti razlike značajki unutar grupe pa se inače koristi mogućnost automatskog pronalaženja i odvajanja najboljih značajki kroz izbornik *Feature Ranking*. Pomoću njega se značajke usporede i rangiraju prema najvećem razlikovanju kvarova Slika 27.



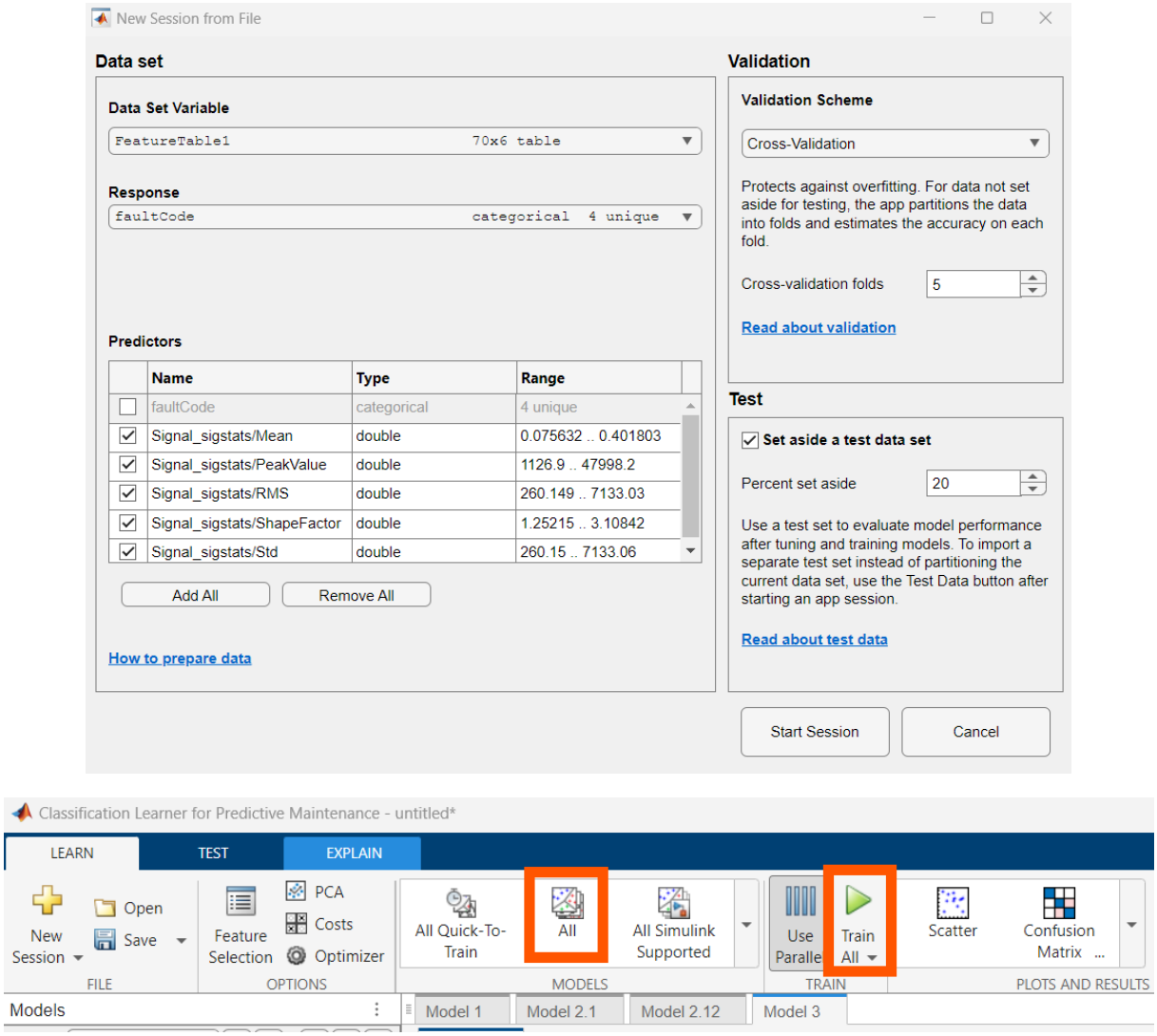
Slika 27. Feature ranking - rangiranje značajki prema ANOVA analizi za stanja sustava

Razlika između kvarova se računa preko ANOVA analize (analiza varijance) u svakoj značajki te se ona s većom razlikom, a time i višim rezultatom analize varijance, rangira više. Vidljivo je kako je vršna vrijednost (engl. *PeakValue*) rangirana kao najznačajnija. Faktor oblika (engl. *ShapeFactor*) je druga po redu značajnosti te onda korijen srednjeg kvadrata (engl. *root mean square, RMS*). Uz njih su uzete još i standardna devijacija (engl. *standard deviation, Std*) te srednja vrijednost (engl. *mean*). Te značajke onda postaju ulazni podaci za treniranje modela strojnog učenja za buduće klasifikacije kvarova. Nakon odabira značajki se otvara izbornik *Classification Learner* u kojem se bira način ocjenjivanja treninga, odnosno validacija Slika 28.



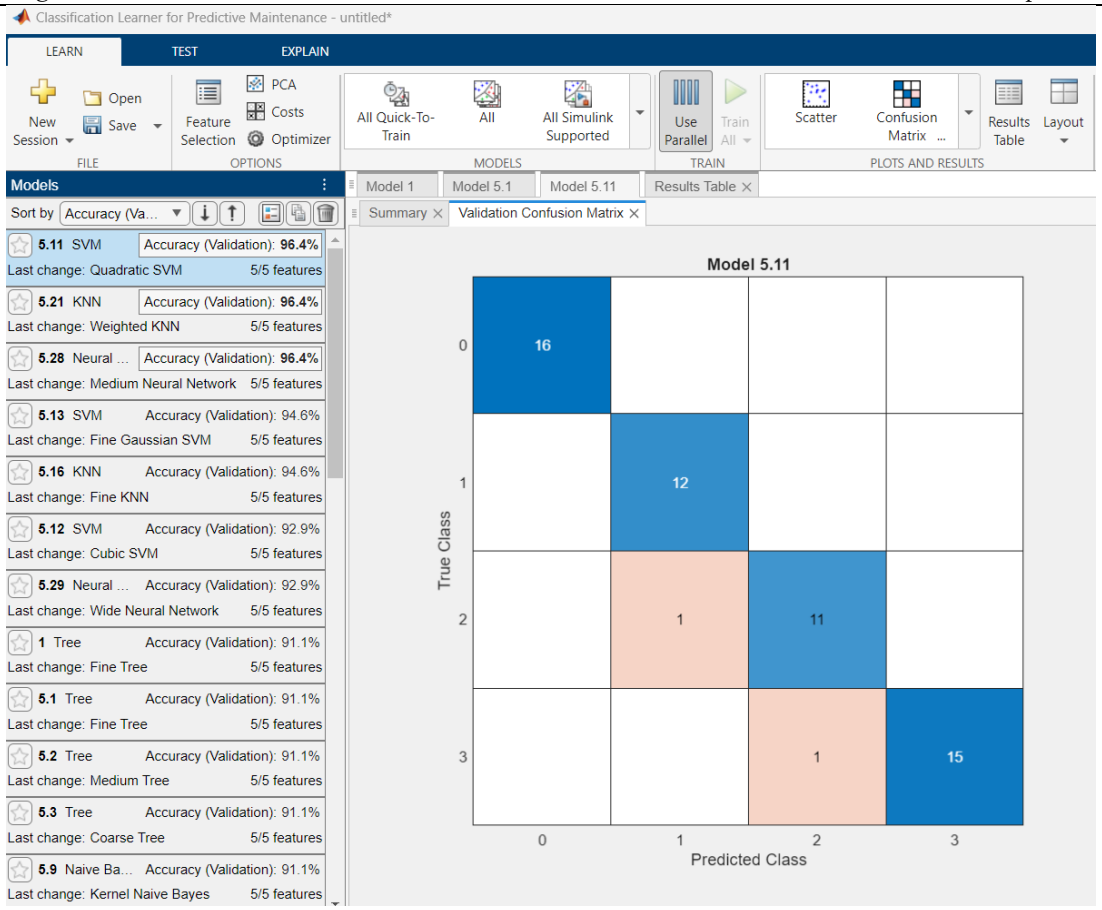
Slika 28. Odabir željenih značajki

Validacija je nužan dio treninga kako bi se model strojnog učenja mogao samostalno ispravljati te kako bi se osiguralo da ne dođe do pretreniranosti modela (engl. *overfitting*). Pretreniranost može dovesti do manje pouzdanog predviđanja budućih kvarova. U ovom slučaju je odabrana peterostruka unakrsna validacija (engl. *5-fold cross validation*) koja slučajno raspoređuje primjere za učenje u pet odvojenih skupova. Za treniranje je uzeto 80% podataka, a za testiranje 20% podataka. Uzorkovanje je stratificirano što osigurava da svaki podskup podataka bude proporcionalno zastupljen u trening i test skupovima podataka. Kako bi se odabrao najbolji model strojnog učenja, odnosno najbolji algoritam, odabrani su svi i po svakom je napravljen trening i evaluacija Slika 29.



Slika 29. Odabir vrste validacije i treniranja pomoću svih vrsta algoritama

Nakon toga se algoritmi mogu rangirati prema evaluaciji (engl. *sort by: Accuracy*). Evaluacija je validiranje razlike predviđanja i stvarnog rezultata na još neviđenom skupu podataka. Rezultati evaluacija se razlikuju od treninga do treninga zbog različitog početnog stanja mreže koje je najčešće slučajno. Najbolje su se pokazali SVM algoritam (engl. *Support Vector Machines*) linearnog tipa, metoda k – najbližih susjeda (engl . *K – nearest neighbor*) i neuronska mreža (engl. *neural network*) s postotkom točnosti od 96.4% Slika 30. Na slici je ujedno prikazana i matrica zabune (engl. *confusion matrix*) pomoću koje se može provjeriti točnost modela na način da se analizira učinak klasifikatora strojnog učenja na skupu podataka kvarova.



Slika 30. Rangiranje različitih istreniranih modela strojnog učenja po točnosti predviđanja i prikaz matrice zabune najbolje istreniranog modela

Dakle, matrica zabune ilustrira koliko dobro model klasificira pojedini kvar, iznosom i intenzitetom boje. Vrijednosti na dijagonali matrice predstavljaju količinu kvara koji je ispravno klasificiran. Ostale vrijednosti predstavljaju netočno određene kvarove. Model se može inače poboljšati odabirom više ili manje značajki za trening ili pripremom značajki iz frekvencijske domene. Modeli su testirani i u Tablica 2 su prikazani te rangirani po točnosti validacije. Prikazana je i količina netočno određenih kvarova za validirane i testirane modele.

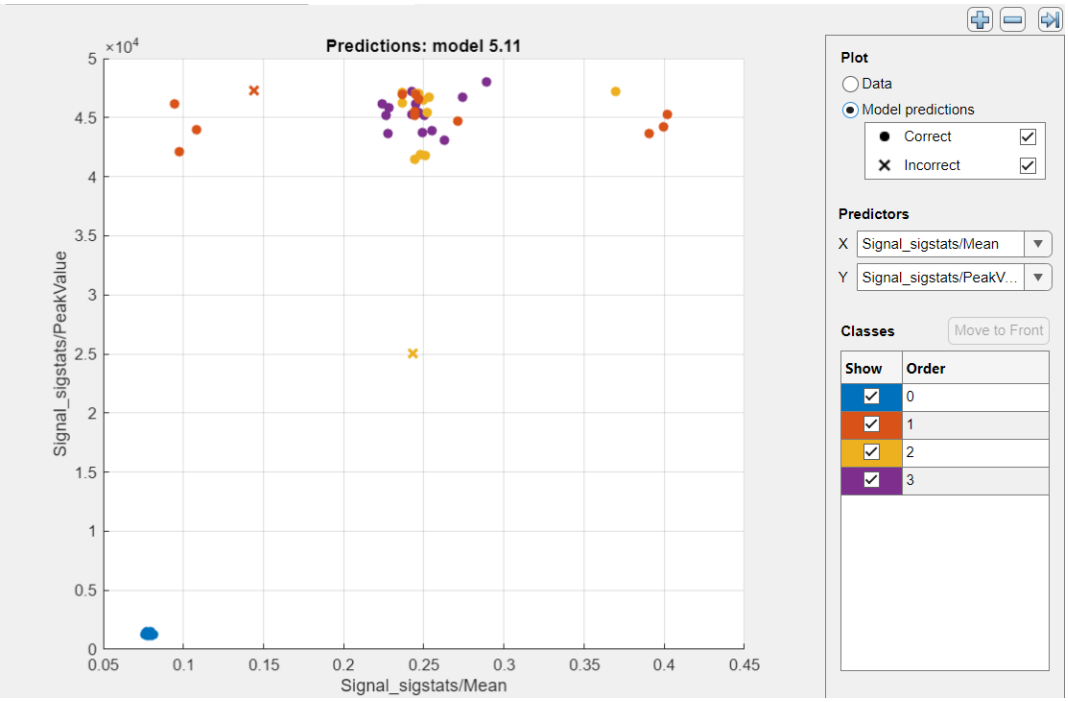
Tablica 2. Tablica modela rangiranih po točnosti validacije

Broj modela	Vrsta modela	Status	Točnost % (Validacija)	Greška (Validacija)	Točnost % (Test)	Greška (Test)
5.11	SVM	Tested	96.42857143	2	100	0
5.21	KNN	Tested	96.42857143	2	100	0
5.28	Neural Network	Tested	96.42857143	2	100	0
5.13	SVM	Tested	94.64285714	3	100	0
5.16	KNN	Tested	94.64285714	3	100	0
5.12	SVM	Tested	92.85714286	4	92.85714286	1
5.29	Neural Network	Tested	92.85714286	4	100	0
1	Tree	Tested	91.07142857	5	85.71428571	2
5.1	Tree	Tested	91.07142857	5	85.71428571	2
5.2	Tree	Tested	91.07142857	5	85.71428571	2
5.3	Tree	Tested	91.07142857	5	85.71428571	2
5.9	Naive Bayes	Tested	91.07142857	5	100	0
5.23	Ensemble	Tested	91.07142857	5	85.71428571	2
5.27	Neural Network	Tested	91.07142857	5	100	0
5.31	Neural Network	Tested	91.07142857	5	92.85714286	1
5.14	SVM	Tested	89.28571429	6	100	0
5.3	Neural Network	Tested	89.28571429	6	100	0
5.1	SVM	Tested	87.5	7	92.85714286	1
5.25	Ensemble	Tested	87.5	7	64.28571429	5
5.32	Kernel	Tested	87.5	7	100	0
5.33	Kernel	Tested	87.5	7	100	0
5.8	Naive Bayes	Tested	82.14285714	10	92.85714286	1
5.4	Discriminant	Tested	80.35714286	11	92.85714286	1
5.7	Efficient Linear SVM	Tested	76.78571429	13	78.57142857	3
5.17	KNN	Tested	73.21428571	15	85.71428571	2
5.24	Ensemble	Tested	73.21428571	15	92.85714286	1
5.19	KNN	Tested	71.42857143	16	78.57142857	3

Pomoću ovog primjera su objašnjene funkcije aplikacije *Diagnostic Feature Designer* te kako ju koristiti za odvajanje značajki i treniranje modela koji uspješno predviđa kvarove u svrhu prediktivnog održavanja. Dalje se izvozi najbolji model na radnu površinu *Matlab*-a koji je sada spreman za klasificiranje novih skupova podataka. Premda prva tri modela imaju isti postotak uspješnosti, za daljnju aplikaciju izabrani su modeli 5.11 i 5.28.

4.3. Značajke modela 5.11.

Model 5.11 koristi SVM (engl. *Support Vector Machine*) s kvadratnom (engl. *quadratic*) funkcijom jezgre. SVM je klasifikacijski model koji traži optimalnu hiperravninu koja razdvaja različite klase podataka na temelju značajki. Dimenzija hiperravnine ovisi o broju značajki. Na primjer, ako postoje dvije ulazne značajke, hiperravnina je jednostavno pravac, a ako postoje tri ulazne značajke, hiperravnina postaje 2D ravnina. Kako broj značajki raste iznad tri, raste i složenost vizualizacije hiperravnine. Kvadratna funkcija jezgre omogućava nelinearnu razdiobu podataka, što znači da može razvrstati složenije skupove podataka gdje klase nisu linearno razdvojive. Model je postigao točnost od 96.4% na validacijskom skupu podataka. To pokazuje vrlo dobru sposobnost modela da ispravno predvidi stanja ležaja na temelju danih značajki. Na slici ispod se može vidjeti točnost predikcija za različite klase gdje su korišteni srednja i vršna vrijednost kao prediktori, a prikazane su sve četiri klase.

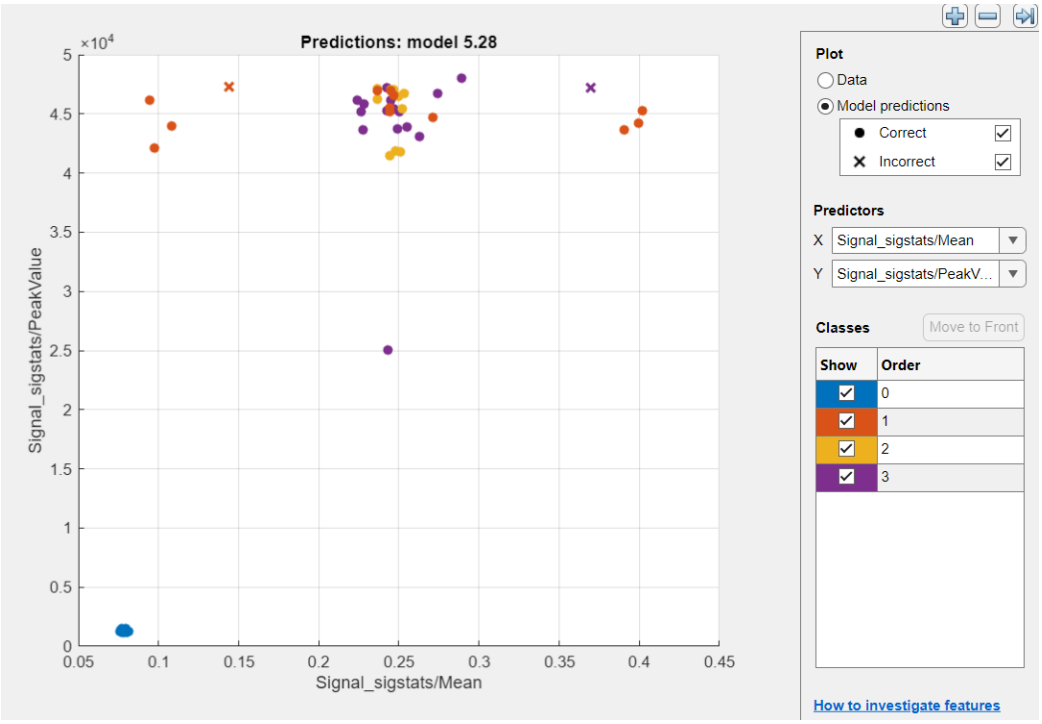


Slika 31. Točnost predikcija modela 5.11 za različite klase s obzirom na vršnu i srednju vrijednost

Može se zaključiti da model 5.11 pokazuje izvrsne rezultate u klasifikaciji stanja ležaja s visokom točnošću i sposobnošću razvrstavanja složenih nelinearnih podataka.

4.4. Značajke modela 5.28.

Model 5.28 je neuronska mreža srednje složenosti koja se koristi za klasifikaciju stanja ležaja. Neuronske mreže su inspirirane načinom rada ljudskog mozga i koriste slojeve neurona kako bi učile kompleksne uzorke u podacima. Ovaj model sadrži više skrivenih slojeva i neurona u svakom sloju, čineći ga dovoljno složenim da prepozna složene obrasce u ultrazvučnim podacima ležaja. Model 5.28 također je postigao točnost od 96.4% na validacijskom skupu podataka, što pokazuje da je model vrlo učinkovit u klasifikaciji stanja ležaja. Na slici ispod su prikazane ispravne i neispravne predikcije modela 5.28.

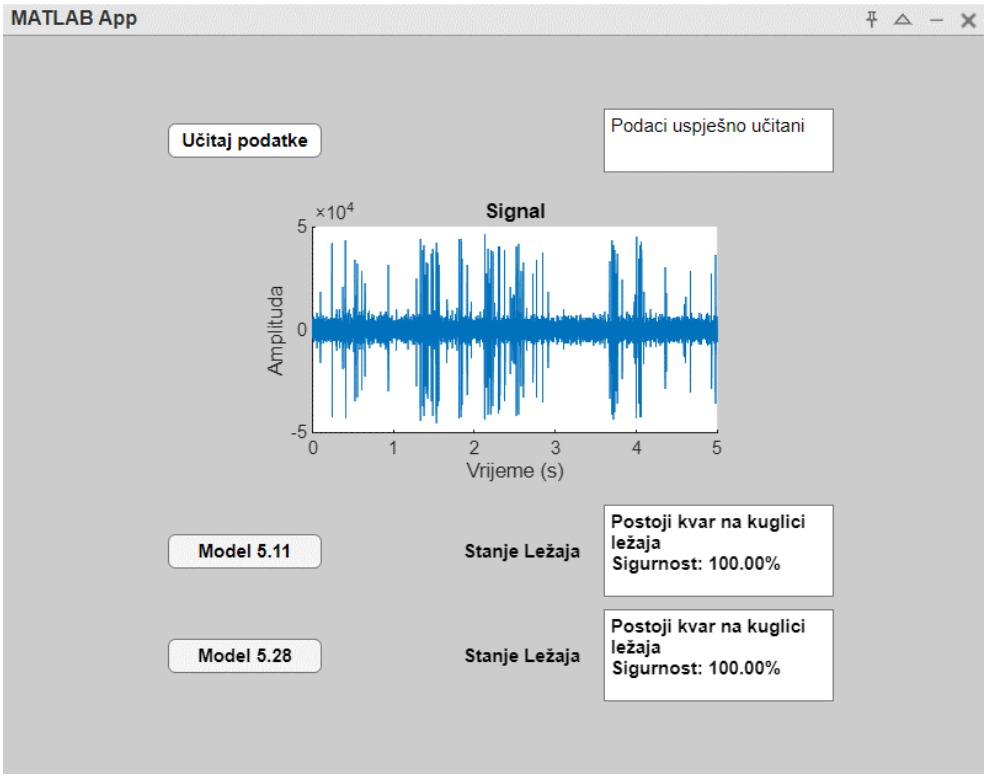


Slika 32. Točnost predikcija modela 5.28 za različite klase s obzirom na vršnu i srednju vrijednost

Neuronske mreže su poznate po svojoj sposobnosti da uče složene nelinearne odnose u podacima te se mogu prilagoditi širokom rasponu problema zbog svoje fleksibilne strukture pa zbog toga i postižu visoku točnost usporedivu s drugim najboljim modelima.

5. RAZVOJ GRAFIČKOG SUČELJA ZA PREDIKTIVNO ODRŽAVANJE

Na temelju razultata opisanih u poglavlju 4 i pripremljenih modela temeljenih na strojnom učenju izrađena je aplikacija u *Matlab-u* koja služi za predikciju stanja ležaja. Aplikacija omogućuje učitavanje podataka, grafički prikaz signala te klasifikaciju stanja ležaja (normalno stanje, kvar na unutarnjem prstenu, vanjskom prstenu ili kuglicama ležaja) Slika 33.



Slika 33. Aplikacija za predviđanje stanja ležaja

Aplikacija je izrađena pomoću *Matlab App Designer-a*, što omogućuje intuitivno korisničko sučelje za interakciju s modelima strojnog učenja. Sučelje se sastoji od tipke za učitavanje podataka, grafičkog prikaza učitanog signala, tipke za pokretanje primjene modela strojnog učenja te tekstualnog prikaza rezultata klasifikacije. Aplikacija je strukturirana na način da se svi ključni koraci izvršavaju preko tipki unutar korisničkog sučelja. Glavne funkcionalnosti aplikacije su:

- **Učitavanje podataka:** omogućuje korisniku da odabere .dat datoteku koja sadrži podatke o ultrazvučnom signalu ležaja.
- **Grafički prikaz signala:** nakon učitavanja, aplikacija prikazuje vremensku domenu amplituda signala u obliku grafa.

- **Predikcija stanja ležaja:** aplikacija koristi unaprijed istrenirani model strojnog učenja (SVM) kako bi klasificirala stanje ležaja.

5.1. Učitavanje podataka

Prvi korak u aplikaciji je omogućiti korisniku da učita ultrazvučne podatke. To se postiže korištenjem Matlab funkcije *uigetfile* koja omogućava odabir datoteke.

```
>>    % Button pushed function: UitajpodatkeButton
        function UitajpodatkeButtonPushed(app, event)
    % Odabir datoteke i učitavanje podataka
[file, path] = uigetfile('*.dat', 'Odaberi datoteku');
filePath = fullfile(path, file);
```

Slika 34. Aplikacija: učitavanje podataka

Nakon što korisnik odabere datoteku, podaci se učitavaju u obliku tablice. Datoteka sadrži četiri stupca, no aplikacija koristi samo stupac koji sadrži vrijednosti amplituda ultrazvučnog signala. Ostali stupci nisu relevantni za analizu.

```
try
    % Učitavanje podataka
    opts = delimitedTextImportOptions('NumVariables', 4);
    opts.DataLines = [1 Inf];  % Uključuje prvi redak podataka
    opts.Delimiter = '\t';     % Definiranje delimitera (\t za tab)
    opts.VariableNames = {'Var1', 'Var2', 'Var3', 'Amplituda'};  % VarName4 je Amplituda
    opts.SelectedVariableNames = {'Amplituda'};  % Zadržavamo samo stupac amplituda
    opts.VariableTypes = {'double', 'double', 'double', 'string'};  % Amplituda kao string
```

Slika 35. Aplikacija: učitavanje vrijednosti amplituda ultrazvučnog signala

Kada se podaci učitaju, zamjenjuju se svi zarezi s točkama jer su podaci zapisani u formatu gdje se decimalne vrijednosti zapisuju sa zarezima, a vrijednosti se pretvaraju u *double* format radi daljnje obrade.

```
>>  % Zamjena zareza u amplitudi s točkama i konverzija u double
    app.dataTable = readtable(filePath, opts);
    app.dataTable.Amplituda = strrep(app.dataTable.Amplituda, ',', '.');
    app.dataTable.Amplituda = str2double(app.dataTable.Amplituda);  % Pretvorba u double
```

Slika 36. Aplikacija: zamjena decimalnih zareza s decimalnim točkama te pretvorba vrijednosti iz tekstualnog u numerički format

Dalje se učitane amplitude prikazuju grafički.

5.2. Grafički prikaz signala

Nakon učitavanja podataka, aplikacija omogućuje korisniku da vizualizira signal u vremenskoj domeni. Prvo je potrebno dodati vrijeme svakom uzorku, a znamo sa uređaja preko kojega su prikupljeni podaci da je frekvencija uzorkovanja 128 000 Hz. Korištenjem frekvencije uzorkovanja za svaki uzorak generirano je vrijeme da se bolje razumije kako se amplituda signala mijenja kroz vrijeme. Nakon toga se grafički može prikazati signal u vremenskoj domeni.

```
% Crtanje grafa
% Vremenska domena
fs = 128000;  % Frekvencija uzorkovanja
vrijeme = (0:height(app.dataTable)-1)' / fs;  % Vrijeme u sekundama

% Crtanje na UIAxes
plot(app.UIAxes2, vrijeme, app.dataTable.Amplituda);
title(app.UIAxes2, 'Signal');
xlabel(app.UIAxes2, 'Vrijeme (s)');
ylabel(app.UIAxes2, 'Amplituda');
```

Slika 37. Aplikacija: grafički prikaz signala

Sljedeći korak je obrada prikupljenih podataka.

5.3. Izdvajanje značajki

Kako bi se predikcija mogla izvršiti, iz signala se trebaju izračunati relevantne značajke koje opisuju njegovu dinamiku. Prvo se podaci dijele na 5 dijelova radi lakše analize te se za svaki dio računaju spomenute statističke značajke:

- Mean (srednja vrijednost): prosječna vrijednost amplituda.
- Peak Value (vršna vrijednost): maksimalna apsolutna vrijednost amplituda.
- RMS (korijen srednjeg kvadrata): kvadratna sredina.
- Shape Factor (faktor oblika): odnos RMS-a i apsolutne srednje vrijednosti.
- Standard Deviation (standardna devijacija): varijacija u amplitudama.

Ove značajke se koriste kao ulazne varijable za model strojnog učenja.

```
 % Podjela podataka na 5 dijelova
broj_uzoraka = height(app.dataTable);
uzorci_po_dijelu = floor(broj_uzoraka / 5);  % broj dijelova = 5

% Inicijalizacija tablice za pohranu značajki
app.features = table();

for i = 1:5
    start_idx = (i-1)*uzorci_po_dijelu + 1;
    if i == 5
        end_idx = broj_uzoraka;  % Posljednji dio ide do kraja
    else
        end_idx = i*uzorci_po_dijelu;
    end

    % Izvlačenje podataka za trenutni dio
    dio_podataka = app.dataTable(start_idx:end_idx, :);

    % Izdvajanje značajki
    meanValue = mean(dio_podataka.Amplituda);          % Mean
    peakValue = max(abs(dio_podataka.Amplituda));      % Peak Value
    rmsValue = rms(dio_podataka.Amplituda);            % RMS
    shapeFactor = rmsValue / mean(abs(dio_podataka.Amplituda));  % Shape Factor
    stdValue = std(dio_podataka.Amplituda);            % Standard Deviation

    % Dodavanje značajki u tablicu
    app.features = [app.features; table(meanValue, peakValue, rmsValue, shapeFactor, stdValue)];
end
```

Slika 38. Aplikacija: izdvajanje značajki signala

Izdvojene značajke moraju biti spremljene u tablicu po stupcima pod istim imenom kao što su i bile u *Diagnostic Feature Designer-u* kako bi ih model mogao prepoznati. Na kraju se u polju za tekst prikazuje jesu li podaci uspješno učitani ili se dogodila greška.

```
    % Dodavanje imena stupaca za značajke
    app.features.Properties.VariableNames = {'Signal_sigstats/Mean', 'Signal_sigstats/PeakValue',
    'Signal_sigstats/RMS', 'Signal_sigstats/ShapeFactor', 'Signal_sigstats/Std'};

    % Pohrana značajki u aplikaciju
    app.features = app.features;

    % Prikaz uspjeha
    app.TextArea.Value = 'Podaci uspješno učitani';

catch ME
    app.TextArea.Value = ['Greška pri učitavanju: ', ME.message];
end
```

Slika 39. Aplikacija: dodavanje imena stupaca za značajke i prikaz uspjeha

Slijedi kod za predikciju stanja ležaja.

5.4. Predikcija stanja ležaja

Nakon što su podaci učitani i značajke izračunate, aplikacija koristi prethodno istrenirani model strojnog učenja za predikciju stanja ležaja. Pritiskom na gumb *Obrada podataka* pomoću modela se dobije trenutno stanje ležaja: normalno stanje ili kvar na unutarnjem prstenu, vanjskom prstenu, ili kuglicama ležaja.

```
% Učitavanje treniranog modela
modelPath = 'model.mat';
load(modelPath, 'trainedModel');  % Učitavanje modela

% Korištenje modela za predikciju na temelju izdvojenih značajki
predictedFaultCodes = trainedModel.predictFcn(app.features);

finalFaultCode = mode(predictedFaultCodes);
```

Slika 40. Aplikacija: predikcija stanja ležaja pomoću prethodno istreniranog modela 5.11

Ovisno o izlazu predikcije (0, 1, 2, ili 3), aplikacija ispisuje odgovarajuću poruku u tekstualnom polju:

- 0 – Stanje je normalno
- 1 – Postoji kvar na unutarnjem prstenu ležaja
- 2 – Postoji kvar na vanjskom prstenu ležaja
- 3 – Postoji kvar na kuglici ležaja

Uz to, aplikacija prikazuje i sigurnost modela u postotku. To se radi tako da se podijeli broj predikcija koje su jednake konačnom stanju (*finalFaultCode*) s ukupnim brojem predikcija, a zatim se taj rezultat pomnoži sa 100 kako bi se dobio postotak. Funkcija *numel* izračunava ukupan broj predikcija koje je model napravio, dok funkcija *sum* broji koliko puta je model dao predikciju koja odgovara konačnom stanju (*finalFaultCode*).

```
>>    % Izračun sigurnosti modela (postotak najčešće predikcije)
        numPredictions = numel(predictedFaultCodes);
        numFinalFaultCode = sum(predictedFaultCodes == finalFaultCode);
        confidencePercentage = (numFinalFaultCode / numPredictions) * 100;

    %% Prikaz predikcije
    % Kategorije su spremljene unutar modela
    switch finalFaultCode
        case '0'
            app.StanjeLeajaTextArea.Value = sprintf
    ('Stanje je normalno\nSigurnost: %.2f%%', confidencePercentage);
        case '1'
            app.StanjeLeajaTextArea.Value = sprintf
    ('Postoji kvar na unutarnjem prstenu ležaja\nSigurnost: %.2f%%', confidencePercentage);
        case '2'
            app.StanjeLeajaTextArea.Value = sprintf
    ('Postoji kvar na vanjskom prstenu ležaja\nSigurnost: %.2f%%', confidencePercentage);
        case '3'
            app.StanjeLeajaTextArea.Value = sprintf
    ('Postoji kvar na kuglici ležaja\nSigurnost: %.2f%%', confidencePercentage);
        otherwise
            app.StanjeLeajaTextArea.Value = 'Nepoznat faultCode';
    end
catch ME
    app.StanjeLeajaTextArea.Value = ['Greška pri predikciji: ', ME.message];
  end
```

Slika 41. Aplikacija: izračun sigurnosti modela 5.11 i prikaz trenutnog stanja ležaja

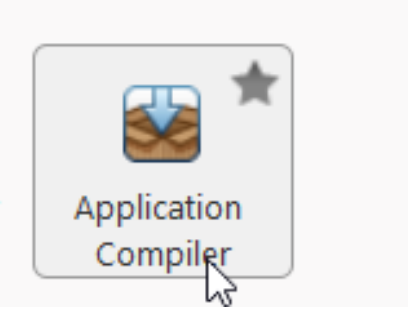
Za model 5.28 vrijedi isti postupak programiranja kao i za model 5.11. Aplikacija omogućuje upotrebu modela strojnog učenja za predikciju stanja ležaja s dodatnim grafičkim prikazom signala i izračunom sigurnosti modela. Time se korisnicima omogućuje jednostavan nadzor nad stanjem ležaja te pravovremeno donošenje odluka o održavanju ili zamjeni komponenti.

5.5. Kompajliranje aplikacije

Proces kompajliranja aplikacije u *Matlab*-u omogućuje kreiranje samostalne aplikacije koja se može koristiti na računalima koja nemaju instaliran *Matlab*. U ovom dijelu opisan je postupak kompajliranja aplikacije pomoću *Application Compiler* alata u *Matlab*-u.

5.5.1. Pokretanje Application Compiler alata

Prvi korak u procesu kompajliranja je otvaranje *Application Compiler* alata. Ovaj alat omogućava pretvaranje *.mlapp* datoteka u izvršne programe *.exe*. Alat se može pokrenuti iz *Matlab*-a u izborniku *apps*, kao što je prikazano na slici ispod.



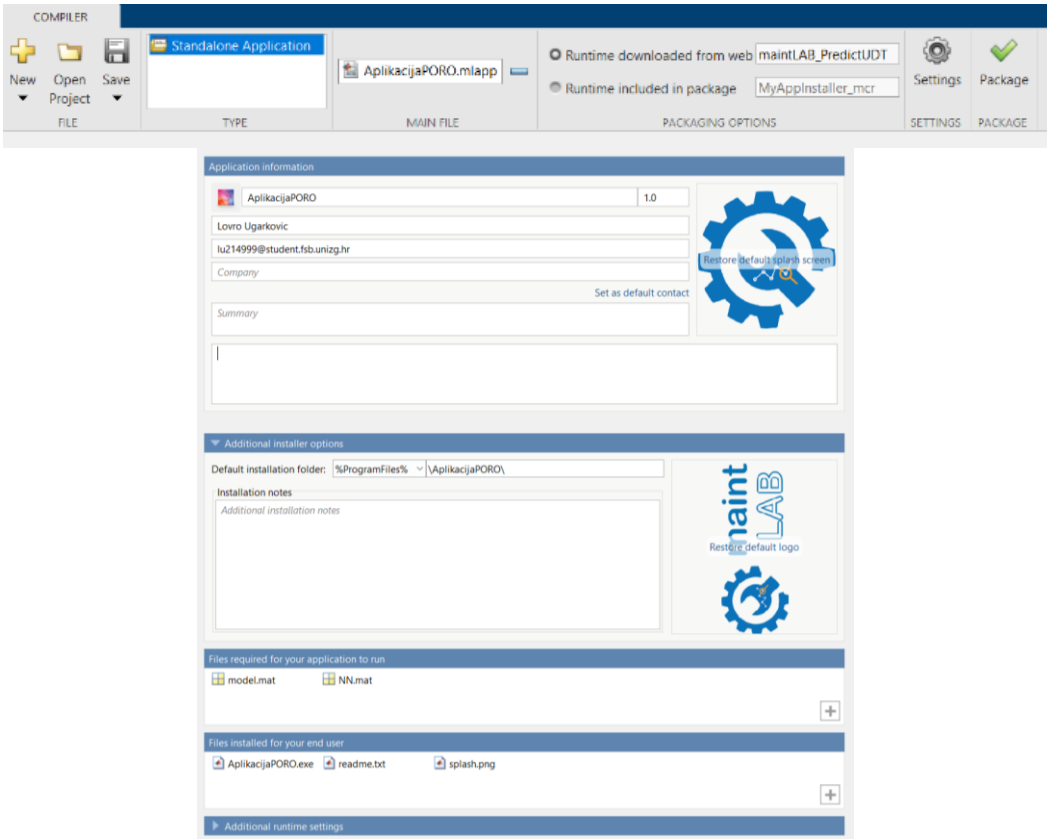
Slika 42. Pokretanje *Application Compiler* alata

5.5.2. Postavke za kompajliranje

Nakon otvaranja alata, potrebno je unijeti osnovne informacije o aplikaciji:

- Ime aplikacije - definirano je ime aplikacije koje će se prikazivati tijekom instalacije.
- Glavna datoteka - *.mlapp* datoteka koja sadrži kod aplikacije.
- Datoteke potrebne za rad aplikacije – datoteke modela koje su ključne za funkcionalnost aplikacije.
- Instalacijske opcije - mjesto gdje će aplikacija biti instalirana, kao i dodatne informacije za korisnika.

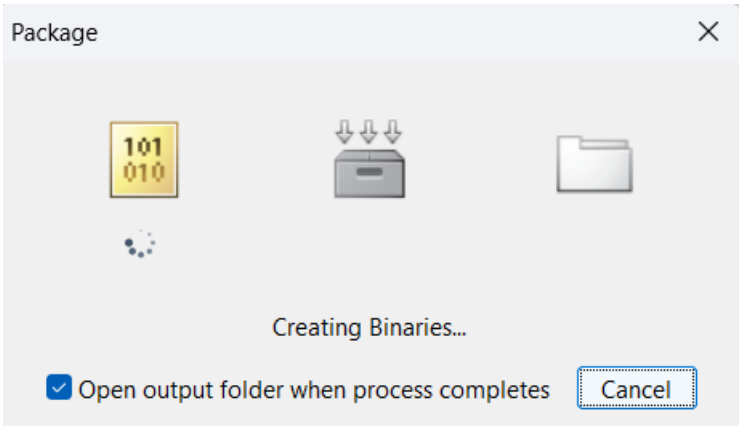
Primjer popunjenih postavki prikazan je na slici ispod.



Slika 43. Postavke za kompajliranje aplikacije

5.5.3. Generiranje izvršne datoteke

Nakon što su postavke ispravno definirane, potrebno je pokrenuti proces kompajliranja klikom na gumb *Package*. Tijekom ovog procesa *Matlab* kreira binarne datoteke potrebne za instalaciju i izvršenje aplikacije. Proces kreiranja binarnih datoteka prikazan je na slici ispod.



Slika 44. Kreiranje binarne datoteke za instalaciju i izvršenje aplikacije

Kada proces završi, izlazna mapa se automatski otvara, omogućujući pristup generiranoj .exe datoteci i pripadajućim instalacijskim datotekama.

5.6. Mogućnosti daljnjeg razvoja aplikacije

Razvoj aplikacije ovisi o specifičnom cilju pa je navedeno nekoliko mogućih smjerova i ideja koje mogu poslužiti kao inspiracija za daljnji razvoj.

5.6.1. Proširenje funkcionalnosti

- Integracija s različitim tipovima senzora: Aplikacija se temelji na podacima senzora pa se mogu dodati mogućnosti zab rad s različitim vrstama senzora ili dodati mogućnosti za proširenje s dodatnim senzorima za praćenje novih parametara.

- Uvođenje sustava automatskog ažuriranja: Omogućiti aplikaciji da automatski ažurira modele s novim podacima, čime bi bila relevantna i prilagođena za različite uvjete.

5.6.2. Optimizacija performansi

- Unapređenje brzine obrade podataka: Optimizirati programski kod kako bi bio brži i efikasniji, što je posebno korisno ako aplikacija obrađuje velike količine podataka.

- Paralelna obrada i distribuirani sustavi: Omogućiti aplikaciji da koristi paralelnu obradu podataka ili distribuirane sustave, što može smanjiti vrijeme obrade i povećati skalabilnost.

5.6.3. Korisničko sučelje

- Modernizacija korisničkog sučelja: Razviti intuitivnije i vizualno privlačnije sučelje koje olakšava rad korisnicima.
- Personalizacija: Dodati mogućnost prilagođavanja prikaza i funkcionalnosti prema specifičnim potrebama korisnika.
- Interaktivni izvještaji i vizualizacija: Razviti interaktivne grafikone, izvještaje i vizualizacije koje će korisnicima omogućiti jednostavno analiziranje podataka.

5.6.4. Analitika i praćenje performansi

- Napredna analitika i statistika: Dodati alate za analizu performansi modela kako bi se pratile ključne metrike poput brzine i učinkovitosti.
- Praćenje povijesnih podataka: Uvesti funkcionalnost za spremanje povijesnih podataka i praćenje promjena kroz vrijeme, čime bi se olakšalo prepoznavanje dugoročnih trendova.
- Nadzorna ploča s ključnim pokazateljima: Kreirati nadzornu ploču gdje bi korisnici mogli vidjeti najvažnije podatke i ključne performanse aplikacije na jednom mjestu.

5.6.5. Sigurnost i zaštita podataka

- Autentifikacija i autorizacija: Dodati slojeve zaštite za pristup aplikaciji kako bi samo ovlašteni korisnici mogli pristupiti osjetljivim podacima i funkcijama.
- Šifriranje podataka: Implementirati šifriranje kako bi se osigurala zaštita osjetljivih podataka.
- Praćenje aktivnosti korisnika: Omogućiti praćenje i evidentiranje aktivnosti korisnika radi poboljšanja sigurnosti i prevencije zloupotrebe.

5.6.6. Poboljšanje modela

- Samostalno učenje modela: Uvesti mogućnost za automatsku optimizaciju modela što bi omogućilo prilagodbu modela novim podacima bez ljudske intervencije.
- Uvođenje novih izvora podataka: Povezivanje s vanjskim bazama podataka.
- Kontinuirano učenje: Razviti sustav za kontinuirano ažuriranje modela kako bi se prilagodio novim uvjetima i podacima.
- Uklapanje više značajki: Analiza dodatnih značajki (npr. značajke iz frekvencijske domene) može pomoći u poboljšanju modela.

6. ZAKLJUČAK

U ovom radu razvijena je aplikacija koja koristi metode strojnog učenja za ranu detekciju kvarova na ležajevima, koristeći ultrazvučnu analizu kao osnovni dijagnostički alat. Kroz analizu podataka prikupljenih uređajem SDT 340 obuhvaćena su četiri stanja ležajeva: normalno stanje, kvar na unutarnjem prstenu, kvar na vanjskom prstenu i kvar na kuglici. Ovi su podaci obrađeni u *Matlab* okruženju gdje je preko aplikacije omogućeno automatsko izdvajanje značajki signala (srednja vrijednost, vršna vrijednost, RMS, faktor oblika i standardna devijacija) koje su ključne za prepoznavanje različitih stanja ležajeva. Glavni dio istraživanja bio je usmjeren na evaluaciju učinkovitosti različitih modela strojnog učenja u klasifikaciji kvarova. Među razmatranim modelima, SVM s kvadratnom jezgrom i neuronske mreže pokazali su najvišu točnost od 96.4%. Visoka točnost klasifikacije ukazuje na potencijal modela za praktičnu primjenu u industriji, gdje bi ovakav sustav mogao osigurati pravovremeno održavanje, smanjiti rizik od neplaniranih zastoja te povećati sigurnost i efikasnost operacija. Budući razvoj ovakvog sustava nudi brojne mogućnosti. Prvo, proširenje aplikacije za analizu podataka s različitih senzora može dodatno povećati pouzdanost modela. Nadalje, dodatna prilagodba algoritama i njihova optimizacija omogućit će veću točnost u identifikaciji specifičnih kvarova, čime bi se sustav mogao proširiti na druge vrste rotacijske opreme poput pumpi, kompresora i turbina. Implementacija aplikacije u stvarnom vremenu mogla bi donijeti još veću vrijednost, omogućujući kontinuirano praćenje i predikciju kvarova bez potrebe za ručnim unosom podataka. Ovaj rad doprinosi razvoju alata za prediktivno održavanje u industrijskim okruženjima te pokazuje kako se metodologije strojnog učenja mogu uspješno primijeniti u svrhu optimizacije industrijskog održavanja. Daljnje istraživanje moglo bi se usmjeriti na povećanje obuhvata podataka u bazi, uključivanje podataka iz drugih industrijskih procesa te razvoj naprednih modela koji mogu predvidjeti kvarove u različitim radnim uvjetima. Primjena ovakvih sustava u industrijskoj praksi donosi značajne prednosti, uključujući smanjenje troškova održavanja, povećanje sigurnosti radnika i smanjenje vremena zastoja, što u konačnici doprinosi dugoročnoj održivosti i uspješnosti industrijskih pogona.

LITERATURA

[1] D. Lisjak. Održavanje. Podloge za predavanja. Fakultet strojarstva i brodogradnje. 2023.

[2] D. Dobranović. Upravljanje stanjem imovine počiva na mjernim metodama. Mind Ability d.o.o. 2019.

[3] D. Lisjak. Dijagnostika u održavanju. Ultrazvuk u nadzoru stanja opreme. Fakultet strojarstva i brodogradnje. 2023.

[4] International standard ISO 29821. Condition monitoring and diagnostics of machines – Ultrasound – General guidelines, procedures and validation. 2018.

[5] Airborne ultrasound sensors. https://sdtultrasound.com/products/sensors-transmitters/airborne-ultrasound-sensors/ pristupljeno: 2.8.2024.

[6] Contact ultrasound sensors. https://sdtultrasound.com/products/sensors-transmitters/contact-ultrasound-sensors/ pristupljeno: 2.8.2024.

[7] Balancing and Bearing Fault Simulator. https://spectraquest.com/simulators/details/bbs/ pristupljeno: 14.8.2024.

[8] SDT 340. https://sdtultrasound.com/products/sdt340/ pristupljeno: 16.8.2024.

[9] Crest factor part 1: peak to average ratio. https://www.merlijnvanveen.nl/en/study-hall/191-crest-factor-part-1 pristupljeno: 17.8.2024.

[10] SDT Ultrasound Solutions: Level 1 Ultrasound inspector training, 2024.

[11] SDT Ultrasound Solutions: Ultranalysis Suite 3 (UAS3) user manual. Version 10 – 2023.

[12] MathWorks, Predictive Maintenance Toolbox, https://www.mathworks.com/products/predictive-maintenance.html, pristupljeno: 19.8.2024.

[13] MathWorks, Help Center, Generate Synthetic Signals Using Conditional GAN, https://www.mathworks.com/help/predmaint/ug/generate-synthetic-signals-using-conditional-generative-adversarial-network.html, pristupljeno: 19.8.2024.

[14] DataScientest, What is a Conditional Generative Adversarial Network (cGAN)?, https://datascientest.com/en/what-is-a-conditional-generative-adversarial-network-cgan, pristupljeno: 19.8.2024.

[15] MathWorks, Predictive Maintenance: Extracting Condition Indicators with MATLAB, https://6377406.fs1.hubspotusercontent-na1.net/hubfs/6377406/%5Bebook%5DPredictive%20Maintenance_ExtractingConditionIndicatorswithMATLAB/%5BeBook%5DPredictive%20Maintenance%20Extracting%20Condition%20Indicators%20with%20MATLAB.pdf?hsCtaTracking=c83daea3-b16e-49c7-b2a1-eec761206070%7C76208b57-5361-4fa8-87e4-b2b43949f66a, pristupljeno: 19.8.2024.

[16] MathWorks, Diagnostic Feature Designer, https://www.mathworks.com/help/predmaint/ref/diagnosticfeaturedesigner-app.html pristupljeno: 14.9.2024.

PRILOG

Program za učitavanje i pripremu podataka za Diagnostic Feature Designer

```
% Definiranje osnovnih parametara
fs = 128000;  % Frekvencija uzorkovanja
broj_dijelova = 5;  % Broj dijelova na koje ćemo podijeliti podatke

% Odabir datoteke i učitavanje podataka
[file, path] = uigetfile('*.dat', 'Select a data file');
filePath = fullfile(path, file);

try
    % Učitavanje podataka
    opts = delimitedTextImportOptions('NumVariables', 4);
    opts.DataLines = [1 Inf];  % Uključuje prvi redak podataka
    opts.Delimiter = '\t';     % Definiranje delimitera (\t za tab)
    opts.VariableNames = {'Var1', 'Var2', 'Var3', 'Amplituda'};  % VarName4 je Amplituda
    opts.SelectedVariableNames = {'Amplituda'};  % Zadržavamo samo stupac amplituda
    opts.VariableTypes = {'double', 'double', 'double', 'string'};  % Amplituda kao string

    % Učitavanje podataka
    dataTable = readtable(filePath, opts);

    % Zamjena zareza u amplitudi s točkama i konverzija u double
    dataTable.Amplituda = strrep(dataTable.Amplituda, ',', '.');
    dataTable.Amplituda = str2double(dataTable.Amplituda);  % Pretvorba u double

    % Podjela podataka na 5 dijelova
    broj_uzoraka = height(dataTable);
    uzorci_po_dijelu = floor(broj_uzoraka / broj_dijelova);

    for i = 1:broj_dijelova
        start_idx = (i-1)*uzorci_po_dijelu + 1;
        if i == broj_dijelova
```

```
        end_idx = broj_uzoraka;  % Posljednji dio ide do kraja
    else
        end_idx = i*uzorci_po_dijelu;
    end

    % Izvlačenje podataka za trenutni dio
    dio_podataka = dataTable(start_idx:end_idx, :);

    % Generiranje vremenskog stupca za ovaj dio (od 0 s korakom 1/fs)
    broj_uzoraka_dio = height(dio_podataka);
    vrijeme_dio = (0:(broj_uzoraka_dio-1))' / fs;  % Vremenski stupac za ovaj dio
    dio_podataka.Vrijeme = seconds(vrijeme_dio);  % Pretvaranje u duration

    % Pretvaranje u timetable
    timetable_ime = sprintf('NS01_%d', i);  % Ime za svaki timetable
    eval([timetable_ime ' = table2timetable(dio_podataka, "RowTimes", "Vrijeme");']);
end

% Dodavanje faultCode stupca
faultCode = zeros(broj_dijelova, 1);  % Primjer s faultCode koji je 0 za sve dijelove
NS_01 = table({NS01_1; NS01_2; NS01_3; NS01_4; NS01_5}, faultCode, ...
'VariableNames', {'Signal', 'faultCode'});
End

% Popis svih prefiksa stanja (NS, IR, OR, BF itd.)
statePrefixes = {'NS', 'IR', 'OR', 'BF'};
faultCodesDict = [0, 1, 2, 3];  % Fault codeovi: NS=0, IR=1, OR=2, BF=3

% Inicijalizacija prazne ćelije za spremanje svih time table tablica
sveTablice = {};
faultCodes = [];
```

```
% Prolazak kroz sva stanja
for stateIdx = 1:numel(statePrefixes)
   prefix = statePrefixes{stateIdx};

   % Prolazak kroz pojedine time table-ove (1 do 5 za svako stanje)
   for partIdx = 1:5
      % Dinamičko generiranje naziva varijabli (npr. NS01_1, IR01_1 itd.)
      for instanceIdx = 1:4
         tableName = sprintf('%s%02d_%d', prefix, instanceIdx, partIdx);

         % Provjera postoji li varijabla u workspace-u
         if evalin('base', sprintf('exist("%s", "var")', tableName))
            % Dohvati time table iz workspace-a
            timeTableData = evalin('base', tableName);

            % Dodaj time table i pripadajući fault code u popis
            sveTablice{end+1, 1} = timeTableData; %#ok<AGROW>
            faultCodes(end+1, 1) = faultCodesDict(stateIdx); %#ok<AGROW>   % Dodijeli
odgovarajući fault code
         end
      end
   end
end

% Kombinacija svih time table-ova u konačnu tablicu
ultrazvuk_podaci = table(sveTablice, faultCodes, 'VariableNames', {'Signal', 'faultCode'});
% Prebaciti stupac faultCode iz numeričkog u kategorijski tip
ultrazvuk_podaci.faultCode = categorical(ultrazvuk_podaci.faultCode);
% Očisti sve ostale varijable iz workspace-a osim konačne tablice
clearvars -except ultrazvuk_podaci;
```

```
classdef AplikacijaPORO < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                matlab.ui.Figure
        TextArea3               matlab.ui.control.TextArea
        Model528Button          matlab.ui.control.Button
        Model511Button          matlab.ui.control.Button
        StanjeleajaTextArea_2   matlab.ui.control.TextArea
        StanjeleajaTextArea_2Label  matlab.ui.control.Label
        StanjeleajaTextArea     matlab.ui.control.TextArea
        StanjeleajaTextAreaLabel    matlab.ui.control.Label
        UitajpodatkeButton      matlab.ui.control.Button
        UIAxes                  matlab.ui.control.UIAxes
    end

    properties (Access = private)
        dataTable
        features
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: UitajpodatkeButton
        function UitajpodatkeButtonPushed(app, event)
          % Odabir datoteke i učitavanje podataka
[file, path] = uigetfile('*.dat', 'Odaberi datoteku');
filePath = fullfile(path, file);
```

```
try
    % Učitavanje podataka
    opts = delimitedTextImportOptions('NumVariables', 4);
    opts.DataLines = [1 Inf];  % Uključuje prvi redak podataka
    opts.Delimiter = '\t';     % Definiranje delimitera (\t za tab)
    opts.VariableNames = {'Var1', 'Var2', 'Var3', 'Amplituda'};  % VarName4 je Amplituda
    opts.SelectedVariableNames = {'Amplituda'};  % Zadržavamo samo stupac amplituda
    opts.VariableTypes = {'double', 'double', 'double', 'string'};  % Amplituda kao string

    % Učitavanje podataka
    app.dataTable = readtable(filePath, opts);

    % Zamjena zareza u amplitudi s točkama i konverzija u double
    app.dataTable.Amplituda = strrep(app.dataTable.Amplituda, ',', '.');
    app.dataTable.Amplituda = str2double(app.dataTable.Amplituda);    % Pretvorba u
double

    % Crtanje grafa
    % Vrijeme će biti generirano na temelju uzorkovanja i broja uzoraka
    fs = 128000;  % Frekvencija uzorkovanja
    vrijeme = (0:height(app.dataTable)-1)' / fs;  % Vrijeme u sekundama

    % Crtanje na UIAxes
    plot(app.UIAxes, vrijeme, app.dataTable.Amplituda);
    title(app.UIAxes, 'Signal');
    xlabel(app.UIAxes, 'Vrijeme (s)');
    ylabel(app.UIAxes, 'Amplituda');

    % Podjela podataka na 5 dijelova
    broj_uzoraka = height(app.dataTable);
    uzorci_po_dijelu = floor(broj_uzoraka / 5);  % broj dijelova = 5

    % Inicijalizacija tablice za pohranu značajki
    app.features = table();
```

```
    for i = 1:5
        start_idx = (i-1)*uzorci_po_dijelu + 1;
        if i == 5
            end_idx = broj_uzoraka;  % Posljednji dio ide do kraja
        else
            end_idx = i*uzorci_po_dijelu;
        end

        % Izvlačenje podataka za trenutni dio
        dio_podataka = app.dataTable(start_idx:end_idx, :);

        % Izdvajanje značajki
        meanValue = mean(dio_podataka.Amplituda);          % Mean
        peakValue = max(abs(dio_podataka.Amplituda));      % Peak Value
        rmsValue = rms(dio_podataka.Amplituda);            % RMS
        shapeFactor = rmsValue / mean(abs(dio_podataka.Amplituda));  % Shape Factor
        stdValue = std(dio_podataka.Amplituda);            % Standard Deviation

        % Dodavanje značajki u tablicu
        app.features = [app.features; table(meanValue, peakValue, rmsValue, shapeFactor,
stdValue)];
    end

    % Dodavanje imena stupaca za značajke
    app.features.Properties.VariableNames = {'Signal_sigstats/Mean',
'Signal_sigstats/PeakValue', 'Signal_sigstats/RMS', 'Signal_sigstats/ShapeFactor',
'Signal_sigstats/Std'};

    % Pohrana značajki u aplikaciju
    app.features = app.features;

    % Prikaz uspjeha
    app.TextArea3.Value = 'Podaci uspješno učitani';
```

```
catch ME
    app.TextArea3.Value = ['Greška pri učitavanju: ', ME.message];
end
        end

        % Button pushed function: Model511Button
        function Model511ButtonPushed(app, event)
            try
    % Učitavanje treniranog modela
    modelPath = 'model.mat';
    load(modelPath, 'trainedModel');  % Učitavanje modela

    % Korištenje modela za predikciju na temelju izdvojenih značajki
    predictedFaultCodes = trainedModel.predictFcn(app.features);

        % Najčešća predikcija (mode)
        finalFaultCode = mode(predictedFaultCodes);

        % Izračun sigurnosti modela (postotak najčešće predikcije)
        numPredictions = numel(predictedFaultCodes);
        numFinalFaultCode = sum(predictedFaultCodes == finalFaultCode);
        confidencePercentage = (numFinalFaultCode / numPredictions) * 100;

    %% Prikaz predikcije
    % Kategorije su spremljene unutar modela
    switch finalFaultCode
        case '0'
            app.StanjeleajaTextArea.Value = sprintf('Stanje je normalno\nSigurnost: %.2f%%',
confidencePercentage);
        case '1'
            app.StanjeleajaTextArea.Value = sprintf('Postoji kvar na unutarnjem prstenu
ležaja\nSigurnost: %.2f%%', confidencePercentage);
        case '2'
```

```
            app.StanjeleajaTextArea.Value = sprintf('Postoji kvar na vanjskom prstenu
    ležaja\nSigurnost: %.2f%%', confidencePercentage);
        case '3'
            app.StanjeleajaTextArea.Value = sprintf('Postoji kvar na kuglici ležaja\nSigurnost:
    %.2f%%', confidencePercentage);
        otherwise
       app.StanjeleajaTextArea.Value = 'Nepoznat faultCode';
    end
catch ME
    app.StanjeleajaTextArea.Value = ['Greška pri predikciji: ', ME.message];
        end
    end

    % Button pushed function: Model528Button
    function Model528ButtonPushed(app, event)
        try
    % Učitavanje treniranog modela
    modelPath = 'NN.mat';
    load(modelPath, 'NM');  % Učitavanje modela

    % Korištenje modela za predikciju na temelju izdvojenih značajki
    predictedFaultCodes = NM.predictFcn(app.features);

    % Najčešća predikcija (mode)
    finalFaultCode = mode(predictedFaultCodes);

    % Izračun sigurnosti modela (postotak najčešće predikcije)
    numPredictions = numel(predictedFaultCodes);
    numFinalFaultCode = sum(predictedFaultCodes == finalFaultCode);
    confidencePercentage = (numFinalFaultCode / numPredictions) * 100;

    %% Prikaz predikcije
    % Kategorije su spremljene unutar modela
    switch finalFaultCode
```

```
        case '0'
            app.StanjeleajaTextArea_2.Value = sprintf('Stanje je normalno\nSigurnost: %.2f%%', confidencePercentage);
        case '1'
            app.StanjeleajaTextArea_2.Value = sprintf('Postoji kvar na unutarnjem prstenu ležaja\nSigurnost: %.2f%%', confidencePercentage);
        case '2'
            app.StanjeleajaTextArea_2.Value = sprintf('Postoji kvar na vanjskom prstenu ležaja\nSigurnost: %.2f%%', confidencePercentage);
        case '3'
            app.StanjeleajaTextArea_2.Value = sprintf('Postoji kvar na kuglici ležaja\nSigurnost: %.2f%%', confidencePercentage);
        otherwise
       app.StanjeleajaTextArea_2.Value = 'Nepoznat faultCode';
    end
catch ME
    app.StanjeleajaTextArea_2.Value = ['Greška pri predikciji: ', ME.message];
            end
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Color = [0.8 0.8 0.8];
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'MATLAB App';

            % Create UIAxes
```

```
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, 'Title')
            xlabel(app.UIAxes, 'X')
            ylabel(app.UIAxes, 'Y')
            zlabel(app.UIAxes, 'Z')
            app.UIAxes.Position = [171 179 300 185];

            % Create UitajpodatkeButton
            app.UitajpodatkeButton = uibutton(app.UIFigure, 'push');
            app.UitajpodatkeButton.ButtonPushedFcn = createCallbackFcn(app,
    @UitajpodatkeButtonPushed, true);
            app.UitajpodatkeButton.FontWeight = 'bold';
            app.UitajpodatkeButton.Position = [109 399 100 22];
            app.UitajpodatkeButton.Text = 'Učitaj podatke';

            % Create StanjeleajaTextAreaLabel
            app.StanjeleajaTextAreaLabel = uilabel(app.UIFigure);
            app.StanjeleajaTextAreaLabel.HorizontalAlignment = 'right';
            app.StanjeleajaTextAreaLabel.FontWeight = 'bold';
            app.StanjeleajaTextAreaLabel.Position = [304 134 77 22];
            app.StanjeleajaTextAreaLabel.Text = 'Stanje ležaja';

            % Create StanjeleajaTextArea
            app.StanjeleajaTextArea = uitextarea(app.UIFigure);
            app.StanjeleajaTextArea.FontWeight = 'bold';
            app.StanjeleajaTextArea.Position = [396 115 150 60];

            % Create StanjeleajaTextArea_2Label
            app.StanjeleajaTextArea_2Label = uilabel(app.UIFigure);
            app.StanjeleajaTextArea_2Label.HorizontalAlignment = 'right';
            app.StanjeleajaTextArea_2Label.FontWeight = 'bold';
            app.StanjeleajaTextArea_2Label.Position = [304 56 77 22];
            app.StanjeleajaTextArea_2Label.Text = 'Stanje ležaja';
```

```
            % Create StanjeleajaTextArea_2
            app.StanjeleajaTextArea_2 = uitextarea(app.UIFigure);
            app.StanjeleajaTextArea_2.FontWeight = 'bold';
            app.StanjeleajaTextArea_2.Position = [396 37 150 60];

            % Create Model511Button
            app.Model511Button = uibutton(app.UIFigure, 'push');
            app.Model511Button.ButtonPushedFcn = createCallbackFcn(app,
    @Model511ButtonPushed, true);
            app.Model511Button.FontWeight = 'bold';
            app.Model511Button.Position = [109 134 100 22];
            app.Model511Button.Text = 'Model 5.11';

            % Create Model528Button
            app.Model528Button = uibutton(app.UIFigure, 'push');
            app.Model528Button.ButtonPushedFcn = createCallbackFcn(app,
    @Model528ButtonPushed, true);
            app.Model528Button.FontWeight = 'bold';
            app.Model528Button.Position = [109 56 100 22];
            app.Model528Button.Text = 'Model 5.28';

            % Create TextArea3
            app.TextArea3 = uitextarea(app.UIFigure);
            app.TextArea3.Position = [423 380 150 60];

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
```

```
        function app = AplikacijaPORO

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```