INSTITUT ZA MATEMATIKU I INFORMATIKU
PRIRODNO-MATEMATIČKI FAKULTET UNIVERZITET U KRAGUJEVCU

PREDSTAVLJANJE I ANALIZA SKUPA PODATAKA

Industrial Fault Detection Dataset”

Studenti:

Aleksandar Mladenović 68/2022

Đorđe Rajčić 84/2022

Mentor:

Branko Arsić

Kragujevac, decembar 2025.

Sadržaj

Uvod

Industrijsko okruženje predstavlja jedan od najzahtevnijih sistema za nadzor, jer brojni procesi moraju da rade pouzdano, precizno i bez zastoja. Svaki kvar u proizvodnoj liniji može dovesti do značajnih finansijskih gubitaka, smanjenja kvaliteta proizvoda ili potpunog prekida proizvodnje. Zato je pravovremeno otkrivanje grešaka u mašinama i opremi od ključnog značaja za svaku fabriku, bez obzira na veličinu ili vrstu industrije. Blagovremena identifikacija anomalija omogućava inženjerima da reaguju pre nego što kvar eskalira, čime se smanjuju troškovi održavanja, produžava životni vek opreme i povećava ukupna efikasnost proizvodnog sistema. Tradicionalne metode nadzora su se uglavnom oslanjale na periodične vizuelne inspekcije i iskustvo radnika, što je često nedovoljno za otkrivanje kompleksnih ili skrivenih problema u ranim fazama. Ovakvi pristupi umeju da budu spori, subjektivni i ograničeni ljudskim faktorom. Razvojem savremenih digitalnih tehnologija i dostupnošću velikih količina podataka, otkrivanje industrijskih kvarova prelazi na naprednije, automatizovane metode. Analitički modeli i algoritmi mašinskog učenja danas mogu da obrade veliki broj senzorskih signala, identifikuju obrasce koji nisu vidljivi ljudskom oku i sa visokim stepenom tačnosti predvide potencijalne kvarove. Na taj način, industrija prelazi sa reaktivnog na proaktivan pristup održavanju. U okviru ovog rada koristi se skup podataka sa senzora industrijske opreme, koji sadrži informacije o normalnom i anomalnom radu sistema. Cilj je da se na osnovu dostupnih parametara izgradi model koji može da prepozna odstupanja i klasifikuje potencijalne greške. Ovakav pristup nalazi široku primenu u sistemima prediktivnog održavanja, gde se kvarovi otkrivaju pre nego što prouzrokuju veće probleme. Za analizu i obradu podataka korišćen je R, zahvaljujući njegovoj bogatoj kolekciji biblioteka namenjenih statističkom modeliranju, manipulaciji podacima i vizuelizaciji rezultata. R se posebno ističe u zadacima klasifikacije. Pored toga, njegova mogućnost jasnog predstavljanja nalaza kroz grafikone i tabele olakšava interpretaciju rezultata i donošenje zaključaka tokom analize.

Opis problema i motivacija

Iz ugla nauke o podacima, problem detekcije industrijskih kvarova predstavlja zadatak klasifikacije, gde se na osnovu poznatih karakteristika signala (kao što su temperatura, vibracije, pritisak, protok, struja i sl.) pokušava proceniti tip kvara. Cilj je izgraditi model koji

može naučiti obrasce i odnose između tih karakteristika i tipa kvara, a zatim ih koristiti za predviđanje kvarova u novim, nepoznatim signalima. U ovom istraživanju koristimo javno dostupan Industrial Fault Detection Dataset koji sadrži podatke merenja sa senzora u industrijskim sistemima. Skup obuhvata atribute koji opisuju različite karakteristike signala, poput osnovnih senzora i FFT karakteristika.

Zadatak projekta je razviti model koji precizno predviđa tip kvara (Fault_Type) na osnovu senzorskih karakteristika. Glavni izazov je konstruisanje robusnog i prediktivnog modela s obzirom na složenost podataka:

- visoka dimenzionalnost zahteva efikasan izbor karakteristika da bi se izbeglo prekomerno prilagođavanje
- mešoviti tipovi podataka zahtevaju pažljivo kodiranje i
- nebalansiranost klasa zahteva balansiranje.

Cilj ovog projekta je uporediti različite modele mašinskog učenja kako bi se identifikovao onaj koji najtačnije detektuje kvarove. Evaluacijom performansi modela analizira se njihova sposobnost da uhvate odnose između karakteristika signala i tipa kvara. Na taj način dolazi se do optimalnog pristupa za pouzdanu detekciju industrijskih kvarova.

Učitavanje neophodnih biblioteka

Na početku se inicijalizuju neophodni paketi u R okruženju kako bi se omogućila efikasna obrada podataka, analiza, modelovanje i evaluacija. Biblioteke su odabrane sa fokusom na minimalizam i performanse, uz izbegavanje nepotrebnih zavisnosti.

```
library(tidyverse)      # Uključuje dplyr, ggplot2, readr, itd.
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.
0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.6
## ✓ forcats   1.0.1     ✓ stringr   1.6.0
## ✓ ggplot2   4.0.1     ✓ tibble    3.3.0
## ✓ lubridate 1.9.4     ✓ tidyr     1.3.1
## ✓ purrr     1.2.0
## ── Conflicts ────────────────────────────────────────── tidyverse_conflict
s() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(caret)          # Za treniranje i evaluaciju modela
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
```

```
## The following object is masked from 'package:purrr':
##
##     lift

library(pROC)           # Za ROC i AUC metrike

## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library(nnet)           # Za multinomijalnu logističku regresiju
library(patchwork)      # Za kombinovanje ggplot grafikona
library(recipes)        # Za definisanje preprocessing pipeline-a

##
## Attaching package: 'recipes'
##
## The following object is masked from 'package:stringr':
##
##     fixed
##
## The following object is masked from 'package:stats':
##
##     step

library(themis)         # Za balansiranje klasa (SMOTE i druge metode)
library(ranger)         # Brza implementacija Random Forest algoritma
```

- **tidyverse**: Skup paketa za rad sa podacima, uključujući **dplyr** za manipulaciju, **ggplot2** za vizualizaciju i **readr** za učitavanje podataka. Koristi se za čišćenje, transformaciju i eksploratornu analizu podataka (EDA).
- **caret**: Centralni paket za mašinsko učenje koji omogućava treniranje modela, cross-validaciju, tuning hiperparametara i evaluaciju performansi kroz jedinstven interfejs.
- **pROC**: Paket namenjen izračunavanju i prikazu ROC krivih i AUC metrike, koji omogućava detaljnu evaluaciju klasifikacionih modela.
- **nnet**: Paket koji omogućava implementaciju multinomijalne logističke regresije i jednostavnih neuronskih mreža, pogodan za baseline modele klasifikacije.
- **patchwork**: Paket za kombinovanje više **ggplot2** grafikona u jedinstvenu kompoziciju, što omogućava preglednije i organizovanije vizualizacije.

- **recipes**: Paket za definisanje preprocessing pipeline-a nad podacima, uključujući skaliranje, normalizaciju i druge transformacije na strukturiran i reproduktivan način.
- **themis**: Paket koji proširuje recipes funkcionalnost metodama za balansiranje nebalansiranih skupova podataka, uključujući SMOTE i srodne tehnike.
- **ranger**: Visokoperformantna implementacija Random Forest algoritma, optimizovana za brzinu i efikasnost, sa podrškom za integraciju sa caret paketom i analizu važnosti atributa.

Učitavanje seta podataka

Inicijalizuju se podaci iz eksternog izvora u R okruženju, što predstavlja osnovu za dalju analizu i modelovanje. Ova faza osigurava da se dataset učita efikasno, sa proverom dimenzija i osnovnih statistika, kako bi se potvrdio integritet podataka pre prelaska na EDA i preprocesiranje. Opcija *stringsAsFactors = FALSE* sprečava automatsko pretvaranje tekstualnih kolona u faktore.

```
dataset_path <- "./dataset/Industrial_fault_detection.csv"
dataset <- read.csv(dataset_path, stringsAsFactors = FALSE)

cat(sprintf("Dataset uspešno učitan: %d redova, %d kolona\n",
            nrow(dataset), ncol(dataset)))

## Dataset uspešno učitan: 1000 redova, 37 kolona

summary(dataset)

##   Temperature      Vibration        Pressure        Flow_Rate
##  Min.   :45.64   Min.   :1.621   Min.   : 56.78   Min.   : 4.141
##  1st Qu.:73.65   1st Qu.:2.890   1st Qu.: 97.21   1st Qu.: 8.525
##  Median :75.02   Median :3.041   Median : 99.96   Median :10.000
##  Mean   :75.01   Mean   :3.032   Mean   : 99.94   Mean   : 9.963
##  3rd Qu.:76.46   3rd Qu.:3.183   3rd Qu.:102.80   3rd Qu.:11.334
##  Max.   :82.16   Max.   :3.784   Max.   :112.13   Max.   :16.486
##     Current         Voltage        FFT_Temp_0      FFT_Vib_0
##  Min.   : 5.47   Min.   :191.0   Min.   :  0.0   Min.   : 0.00
##  1st Qu.:12.95   1st Qu.:213.1   1st Qu.:739.9   1st Qu.:29.37
##  Median :14.95   Median :219.6   Median :749.9   Median :30.30
##  Mean   :14.85   Mean   :219.5   Mean   :743.3   Mean   :30.04
##  3rd Qu.:16.92   3rd Qu.:226.1   3rd Qu.:760.3   3rd Qu.:31.38
##  Max.   :24.34   Max.   :251.0   Max.   :797.3   Max.   :35.70
##    FFT_Pres_0       FFT_Temp_1       FFT_Vib_1        FFT_Pres_1
##  Min.   :   0.0   Min.   : 0.000   Min.   :0.0000   Min.   : 0.00
##  1st Qu.: 977.9   1st Qu.: 7.797   1st Qu.:0.7576   1st Qu.:16.36
##  Median :1000.6   Median :11.763   Median :1.3158   Median :25.03
##  Mean   : 990.9   Mean   :13.381   Mean   :1.3662   Mean   :26.66
##  3rd Qu.:1020.8   3rd Qu.:17.942   3rd Qu.:1.8551   3rd Qu.:35.55
```

```
##  Max.   :1091.5   Max.   :39.546   Max.   :4.1063   Max.   :70.92
##    FFT_Temp_2       FFT_Vib_2        FFT_Pres_2      FFT_Temp_3
##  Min.   : 0.000   Min.   :0.0000   Min.   : 0.00   Min.   : 0.000
##  1st Qu.: 8.696   1st Qu.:0.7305   1st Qu.:18.02   1st Qu.: 7.723
##  Median :13.037   Median :1.2668   Median :26.96   Median :12.594
##  Mean   :13.795   Mean   :1.3438   Mean   :28.47   Mean   :13.485
##  3rd Qu.:18.326   3rd Qu.:1.8450   3rd Qu.:37.76   3rd Qu.:18.457
##  Max.   :40.499   Max.   :3.8212   Max.   :83.64   Max.   :35.201
##    FFT_Vib_3       FFT_Pres_3      FFT_Temp_4       FFT_Vib_4
##  Min.   :0.000   Min.   : 0.00   Min.   : 0.000   Min.   :0.0000
##  1st Qu.:0.821   1st Qu.:16.42   1st Qu.: 8.578   1st Qu.:0.8393
##  Median :1.327   Median :26.81   Median :13.956   Median :1.4119
##  Mean   :1.408   Mean   :27.37   Mean   :14.287   Mean   :1.4440
##  3rd Qu.:1.886   3rd Qu.:36.51   3rd Qu.:18.984   3rd Qu.:1.9651
##  Max.   :4.533   Max.   :82.44   Max.   :35.529   Max.   :3.9143
##    FFT_Pres_4      FFT_Temp_5       FFT_Vib_5        FFT_Pres_5
##  Min.   : 0.00   Min.   : 0.000   Min.   :0.0000   Min.   :  0.000
##  1st Qu.:16.75   1st Qu.: 4.240   1st Qu.:0.4635   1st Qu.:  7.838
##  Median :25.98   Median : 9.281   Median :0.9414   Median : 18.288
##  Mean   :27.53   Mean   :11.374   Mean   :1.1691   Mean   : 23.574
##  3rd Qu.:37.22   3rd Qu.:16.906   3rd Qu.:1.6460   3rd Qu.: 33.601
##  Max.   :93.00   Max.   :49.219   Max.   :5.7256   Max.   :103.476
##    FFT_Temp_6       FFT_Vib_6        FFT_Pres_6      FFT_Temp_7
##  Min.   : 0.000   Min.   :0.0000   Min.   : 0.00   Min.   : 0.000
##  1st Qu.: 8.578   1st Qu.:0.8393   1st Qu.:16.75   1st Qu.: 7.723
##  Median :13.956   Median :1.4119   Median :25.98   Median :12.594
##  Mean   :14.287   Mean   :1.4440   Mean   :27.53   Mean   :13.485
##  3rd Qu.:18.984   3rd Qu.:1.9651   3rd Qu.:37.22   3rd Qu.:18.457
##  Max.   :35.529   Max.   :3.9143   Max.   :93.00   Max.   :35.201
##    FFT_Vib_7       FFT_Pres_7      FFT_Temp_8       FFT_Vib_8
##  Min.   :0.000   Min.   : 0.00   Min.   : 0.000   Min.   :0.0000
##  1st Qu.:0.821   1st Qu.:16.42   1st Qu.: 8.696   1st Qu.:0.7305
##  Median :1.327   Median :26.81   Median :13.037   Median :1.2668
##  Mean   :1.408   Mean   :27.37   Mean   :13.795   Mean   :1.3438
##  3rd Qu.:1.886   3rd Qu.:36.51   3rd Qu.:18.326   3rd Qu.:1.8450
##  Max.   :4.533   Max.   :82.44   Max.   :40.499   Max.   :3.8212
##    FFT_Pres_8      FFT_Temp_9       FFT_Vib_9        FFT_Pres_9
##  Min.   : 0.00   Min.   : 0.000   Min.   :0.0000   Min.   : 0.00
##  1st Qu.:18.02   1st Qu.: 7.797   1st Qu.:0.7576   1st Qu.:16.36
##  Median :26.96   Median :11.763   Median :1.3158   Median :25.03
##  Mean   :28.47   Mean   :13.381   Mean   :1.3662   Mean   :26.66
##  3rd Qu.:37.76   3rd Qu.:17.942   3rd Qu.:1.8551   3rd Qu.:35.55
##  Max.   :83.64   Max.   :39.546   Max.   :4.1063   Max.   :70.92
##    Fault_Type
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.548
##  3rd Qu.:1.000
##  Max.   :3.000
```

Osnovni senzori (Temperature, Vibration, Pressure, Flow_Rate, Current, Voltage) pokazuju relativno usku distribuciju oko medijana, sa malim odstupanjima od proseka (npr. temperatura: medijana 75.02, prosek 75.01; vibracije: medijana 3.041, prosek 3.032). Ovo ukazuje na stabilna radna stanja većine merenja.

FFT komponente (npr. FFT_Temp_0 do FFT_Pres_9) imaju širok raspon vrednosti, sa značajnim razlikama između nižih i viših binova, što je očekivano za frekventne karakteristike signala. Većina FFT binova pokazuje slične statistike u parovima (npr. bin 2 i bin 7 imaju identične kvartile i prosek), što može ukazivati na simetričnost ili redundantnost u FFT transformaciji.

Ciljna promenljiva Fault_Type ima vrednosti od 0 do 3 (kodirane klase: Normal, Overheating, Leakage, Power Fluctuation). Prosečna vrednost 0.548 i medijana 0.000 sugerišu značajnu nebalansiranost, većinski deo podataka pripada klasi 0 (normalno stanje), što će zahtevati tehniku balansiranja u daljim koracima.

Eksplorativna analiza podataka (EDA)

Eksplorativna analiza podataka predstavlja ključni početni korak u svakom istraživanju iz oblasti nauke o podacima. Njena svrha je da se podaci detaljno istraže, sumiraju i vizuelno prikažu kako bi se otkrile njihove glavne karakteristike, obrasci, anomalije, odnosi između promenljivih i potencijalni problemi.

Osnovne informacije o datasetu

Izvršava se inicijalna inspekcija kvaliteta i strukture podataka, što predstavlja neophodan korak pre detaljnije eksplorativne analize. Ova faza omogućava brzu proveru dimenzija skupa, prisustva duplikata, nedostajućih vrednosti i potpuno praznih redova, kao i pregled tipova promenljivih i osnovnih statistika za ključne senzore.

```
cat(sprintf("Dimenzije: %d x %d\n", nrow(dataset), ncol(dataset)))
```

```
## Dimenzije: 1000 x 37
```

```
cat(sprintf("Duplikati: %d\n", sum(duplicated(dataset))))
```

```
## Duplikati: 0
```

```
cat(sprintf("Nedostajuće vrednosti: %d\n", sum(is.na(dataset))))
```

```
## Nedostajuće vrednosti: 0
```

```
num_df <- dataset %>% select(where(is.numeric))
zero_rows <- which(rowSums(num_df == 0, na.rm = TRUE) == ncol(num_df))
cat(sprintf("Potpuno prazni redovi: %d\n", length(zero_rows)))
```

```
## Potpuno prazni redovi: 0
```

```
print(str(dataset))
```

```
## 'data.frame':    1000 obs. of  37 variables:
##  $ Temperature: num  46 62.5 77.3 76.6 78.3 ...
##  $ Vibration  : num  2.04 2.57 3.24 3.14 3.14 ...
##  $ Pressure   : num  56.8 76.2 92.4 94.1 94.4 ...
##  $ Flow_Rate  : num  6.18 8.28 9.17 13.78 11.11 ...
##  $ Current    : num  12.4 14.9 15.1 16.4 10.9 ...
##  $ Voltage    : num  216 215 202 217 227 ...
##  $ FFT_Temp_0 : num  772 768 766 764 747 ...
##  $ FFT_Vib_0  : num  32.4 32.4 32 33 33 ...
##  $ FFT_Pres_0 : num  972 962 956 957 950 ...
##  $ FFT_Temp_1 : num  3.76 8.37 9.56 10.25 18.31 ...
##  $ FFT_Vib_1  : num  0.734 0.725 0.934 0.169 0.173 ...
##  $ FFT_Pres_1 : num  30.9 27.3 30.1 29.9 36.1 ...
##  $ FFT_Temp_2 : num  8.79 12.55 12.29 10.32 14.44 ...
##  $ FFT_Vib_2  : num  1.157 1.131 0.803 1.14 1.149 ...
##  $ FFT_Pres_2 : num  4.37 8.1 13.26 13.29 7.15 ...
##  $ FFT_Temp_3 : num  22.09 19.49 21.11 20.86 8.16 ...
##  $ FFT_Vib_3  : num  1.29 1.283 0.972 0.606 0.627 ...
##  $ FFT_Pres_3 : num  34 41.4 43.4 43.6 45.5 ...
##  $ FFT_Temp_4 : num  2.53 6.94 5.46 6.58 17.85 ...
##  $ FFT_Vib_4  : num  0.437 0.413 0.534 1.169 1.178 ...
##  $ FFT_Pres_4 : num  23.4 25.1 20.2 19.9 14.8 ...
##  $ FFT_Temp_5 : num  2.2 7 5.37 7.4 9.79 ...
##  $ FFT_Vib_5  : num  1.67 1.62 1.99 3 3.02 ...
##  $ FFT_Pres_5 : num  25.8 35.1 29 28.7 22.4 ...
##  $ FFT_Temp_6 : num  2.53 6.94 5.46 6.58 17.85 ...
##  $ FFT_Vib_6  : num  0.437 0.413 0.534 1.169 1.178 ...
##  $ FFT_Pres_6 : num  23.4 25.1 20.2 19.9 14.8 ...
##  $ FFT_Temp_7 : num  22.09 19.49 21.11 20.86 8.16 ...
##  $ FFT_Vib_7  : num  1.29 1.283 0.972 0.606 0.627 ...
##  $ FFT_Pres_7 : num  34 41.4 43.4 43.6 45.5 ...
##  $ FFT_Temp_8 : num  8.79 12.55 12.29 10.32 14.44 ...
##  $ FFT_Vib_8  : num  1.157 1.131 0.803 1.14 1.149 ...
##  $ FFT_Pres_8 : num  4.37 8.1 13.26 13.29 7.15 ...
##  $ FFT_Temp_9 : num  3.76 8.37 9.56 10.25 18.31 ...
##  $ FFT_Vib_9  : num  0.734 0.725 0.934 0.169 0.173 ...
##  $ FFT_Pres_9 : num  30.9 27.3 30.1 29.9 36.1 ...
##  $ Fault_Type : int  0 0 0 0 3 0 0 0 0 1 ...
## NULL
```

```
print(sapply(dataset, class))
```

```
## Temperature   Vibration    Pressure   Flow_Rate     Current     Voltage
##   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"
##  FFT_Temp_0   FFT_Vib_0  FFT_Pres_0  FFT_Temp_1   FFT_Vib_1  FFT_Pres_1
##   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"
##  FFT_Temp_2   FFT_Vib_2  FFT_Pres_2  FFT_Temp_3   FFT_Vib_3  FFT_Pres_3
##   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"
##  FFT_Temp_4   FFT_Vib_4  FFT_Pres_4  FFT_Temp_5   FFT_Vib_5  FFT_Pres_5
##   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"
```

```
##  FFT_Temp_6   FFT_Vib_6  FFT_Pres_6  FFT_Temp_7   FFT_Vib_7  FFT_Pres_7
##   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"
##  FFT_Temp_8   FFT_Vib_8  FFT_Pres_8  FFT_Temp_9   FFT_Vib_9  FFT_Pres_9
##   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"   "numeric"
##  Fault_Type
##   "integer"
```

```
base_sensors <- c("Temperature", "Vibration", "Pressure",
                  "Flow_Rate", "Current", "Voltage")

print(summary(dataset %>% select(all_of(base_sensors))))
```

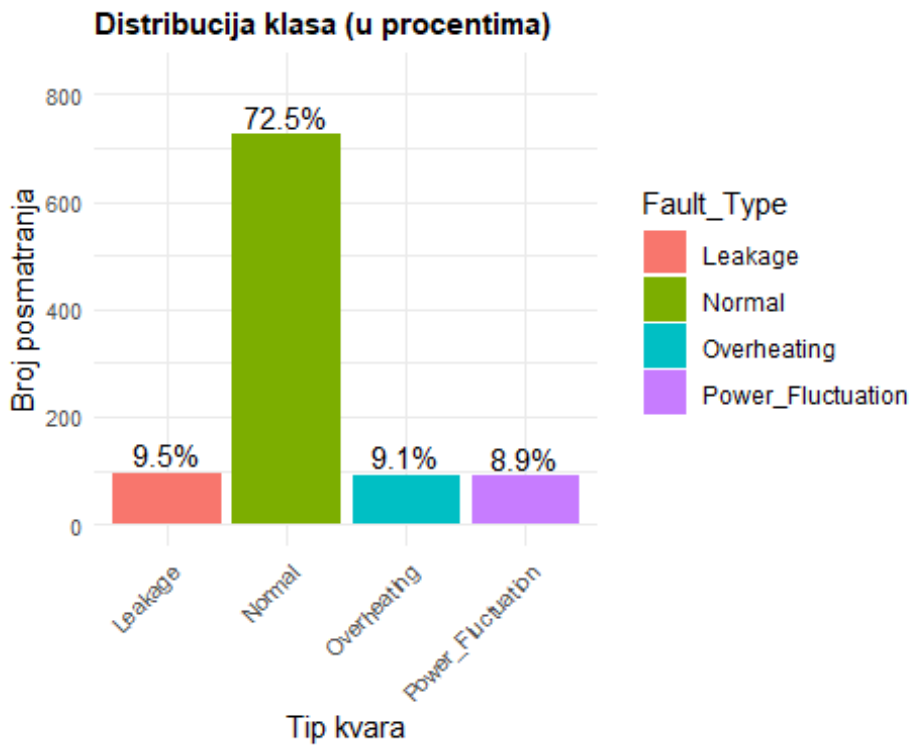
```
##   Temperature      Vibration        Pressure        Flow_Rate
##  Min.   :45.64   Min.   :1.621   Min.   : 56.78   Min.   : 4.141
##  1st Qu.:73.65   1st Qu.:2.890   1st Qu.: 97.21   1st Qu.: 8.525
##  Median :75.02   Median :3.041   Median : 99.96   Median :10.000
##  Mean   :75.01   Mean   :3.032   Mean   : 99.94   Mean   : 9.963
##  3rd Qu.:76.46   3rd Qu.:3.183   3rd Qu.:102.80   3rd Qu.:11.334
##  Max.   :82.16   Max.   :3.784   Max.   :112.13   Max.   :16.486
##     Current         Voltage
##  Min.   : 5.47   Min.   :191.0
##  1st Qu.:12.95   1st Qu.:213.1
##  Median :14.95   Median :219.6
##  Mean   :14.85   Mean   :219.5
##  3rd Qu.:16.92   3rd Qu.:226.1
##  Max.   :24.34   Max.   :251.0
```

Rezultati pokazuju da dataset sadrži 1000 posmatranja i 37 promenljivih, bez duplikata, nedostajućih vrednosti ili potpuno praznih redova, što ukazuje na visok kvalitet i spremnost podataka za dalju analizu. Sve prediktorske promenljive su numeričkog tipa (numeric), dok je ciljna promenljiva Fault_Type celobrojna (integer). Deskriptivna statistika osnovnih senzora (Temperature, Vibration, Pressure, Flow_Rate, Current, Voltage) pokazuje stabilne vrednosti sa malim rasponom odstupanja oko medijane, što je karakteristično za normalna radna stanja industrijske opreme.

Analiza ciljne varijable (Fault_Type)

Vrši se detaljna inspekcija distribucije ciljne promenljive, koja predstavlja tip kvara u industrijskom sistemu. Ova analiza je ključna jer nebalansiranost klasa može značajno uticati na performanse modela klasifikacije, dovodeći do pristrasnosti prema dominantnoj klasi. Prvo se numerička promenljiva Fault_Type pretvara u faktor sa smislenim labelama, zatim se izračunavaju apsolutne i relativne frekvencije, a na kraju se kreira vizuelizacija distribucije pomoću ggplot2 paketa.

```
dataset$Fault_Type <- factor(
  dataset$Fault_Type,
  levels = c("0", "1", "2", "3"),
  labels = c("Normal", "Overheating", "Leakage", "Power_Fluctuation")
)

fault_table <- table(dataset$Fault_Type)
fault_prop <- prop.table(fault_table)
print(round(fault_prop * 100, 2))

##
##            Normal       Overheating           Leakage Power_Fluctuation
##              72.5               9.1               9.5               8.9

freq_df <- data.frame(
  Fault_Type = names(fault_table),
  Freq = as.numeric(fault_table),
  Perc = round(as.numeric(fault_prop) * 100, 1)
)

plot_distribution <- ggplot(freq_df, aes(x = Fault_Type, y = Freq, fill = Fau
lt_Type)) +
  geom_col() +
  geom_text(aes(label = paste0(Perc, "%")), vjust = -0.3, size = 4) +
  labs(title = "Distribucija klasa (u procentima)",
       x = "Tip kvara",
       y = "Broj posmatranja") +
  ylim(0, max(freq_df$Freq) * 1.15) +
  theme_grid_readable()

plot_distribution
```



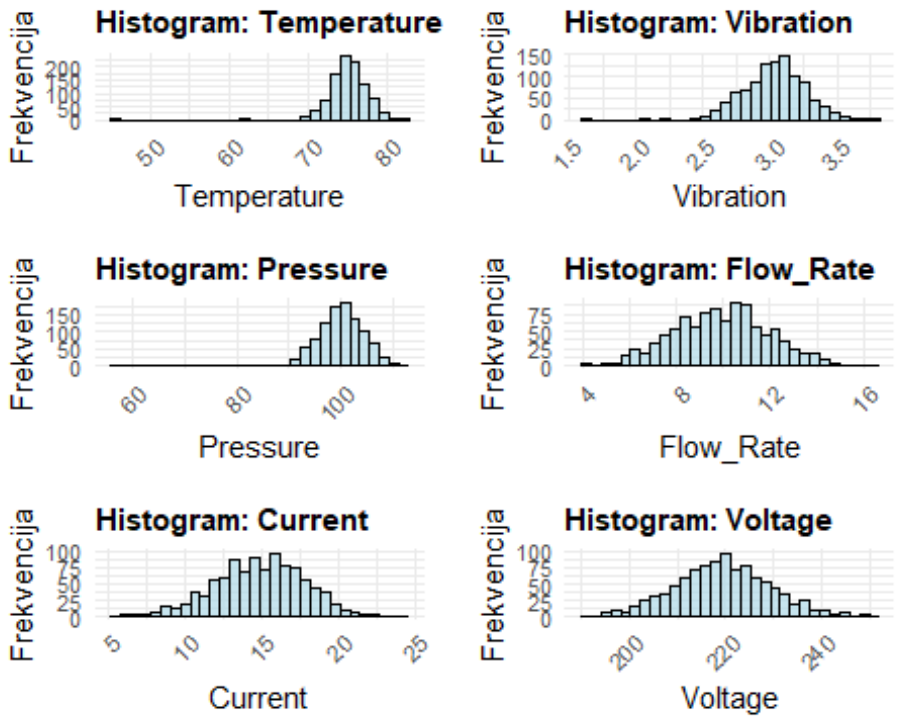
Distribucija ciljne promenljive Fault_Type pokazuje izrazitu nebalansiranost: klasa "Normal" čini 72.5% svih posmatranja, dok su ostale klase značajno manje zastupljene: "Leakage" 9.5%, "Overheating" 9.1% i "Power_Fluctuation" 8.9%. Ova disproporcija je očekivana u realnim industrijskim podacima, gde normalno stanje dominira, ali predstavlja ozbiljan izazov za modele mašinskog učenja. Bez adekvatnog balansiranja, modeli bi imali tendenciju da predviđaju dominantnu klasu "Normal" sa visokom tačnošću, ali sa lošim performansama na manjinskim klasama kvarova. Vizuelizacija distribucije jasno ilustruje ovu neravnotežu i naglašava potrebu za tehnikama balansiranja u fazi preprocesiranja.

Generisanje histograma osnovnih senzora

Ovde se vrši vizualizacija distribucija šest primarnih senzorskih promenljivih (Temperature, Vibration, Pressure, Flow_Rate, Current, Voltage), što omogućava uvid u oblik, centralnu tendenciju, raspršenost i eventualnu asimetriju podataka. Ovi histogrami su ključni za identifikaciju karakteristika normalnog rada opreme, detekciju odstupanja i procenu pogodnosti za dalje statističke analize ili modelovanje.

```
sensor_plots <- lapply(base_sensors, function(var) {
  ggplot(dataset, aes_string(x = var)) +
    geom_histogram(color = "black", fill = "lightblue", bins = 30, alpha = 0.
7) +
    labs(title = paste("Histogram:", var), x = var, y = "Frekvencija") +
    theme_grid_readable() +
    theme(legend.position = "none")
})
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

combined_histograms <- wrap_plots(sensor_plots, ncol = 2)
combined_histograms
```



Histogrami pokazuju da su distribucije osnovnih senzora uglavnom unimodalne i blago asimetrične, sa koncentracijom vrednosti oko medijane (npr. temperatura oko 75 °C, vibracije oko 3.0 m/s^2, pritisak oko 100 jedinica). Temperatura i pritisak imaju relativno uske raspone sa izraženim vrhom u srednjem delu, što ukazuje na stabilna radna stanja. Vibracije i protok (Flow_Rate) pokazuju nešto širu distribuciju sa blagim repom ka višim vrednostima, dok struja (Current) i napon (Voltage) imaju više izraženu asimetriju sa dužim repom ka višim vrednostima, što može odgovarati epizodama fluktuacija ili kvarova. Odsustvo izrazitih bimodalnih oblika sugeriše da većina merenja pripada normalnom režimu rada, dok ekstremne vrednosti (posebno u repovima) verovatno odgovaraju retkim kvarovima. Ovi nalazi potvrđuju potrebu za pažljivim skaliranjem i feature engineering-om u narednim fazama, jer senzori imaju različite skale i varijabilnost.

Boxplotovi senzora u odnosu na Fault_Type

Vrši se grupisana vizualizacija distribucija šest osnovnih senzorskih promenljivih (Temperature, Vibration, Pressure, Flow_Rate, Current, Voltage) prema tipu kvara. Boxplotovi omogućavaju direktno poređenje centralnih tendencija (medijana), raspršenosti (IQR), ekstremnih vrednosti i outlier-a između klasa, čime se otkrivaju potencijalni diskriminativni obrasci između normalnog stanja i različitih tipova kvarova.

```
boxplot_plots <- lapply(base_sensors, function(var) {
  ggplot(dataset, aes_string(x = "Fault_Type", y = var, fill = "Fault_Type"))
+
    geom_boxplot(alpha = 0.7) +
    labs(title = paste(var, "po tipu kvara"), x = "Tip kvara", y = var) +
    theme_grid_readable() + theme(legend.position = "none")
})
combined_boxplots <- wrap_plots(boxplot_plots, ncol = 2)
combined_boxplots
```

Boxplotovi pokazuju jasne razlike u ponašanju senzora između klasa:

- Temperature: Najviša medijana i najveći raspon u klasi “Overheating”, sa izraženim outlier-ima ka višim vrednostima, što je očekivano za pregrevanje. Klase “Normal” i “Leakage” imaju slične, niže vrednosti.
- Vibration: Nešto više vrednosti i veća varijabilnost u klasama “Leakage” i “Overheating”, dok je “Normal” najstabilniji sa manjim outlier-ima.

- Pressure i Flow_Rate: Relativno slične distribucije među klasama, sa blago nižim medijanama u “Power_Fluctuation” i “Leakage”, ali bez dramatičnih razlika.
- Current i Voltage: Veća varijabilnost i outlier-i u “Normal” i “Overheating”, dok su “Leakage” i “Power_Fluctuation” stabilniji sa nižim medijanama, što može ukazivati na fluktuacije napajanja.

Ovi grafikoni jasno ističu da senzori Temperature i Vibration imaju najjači diskriminativni potencijal za razlikovanje kvarova od normalnog stanja, dok ostali senzori pokazuju manje izražene razlike. Ova saznanja su važna za usmeravanje feature selekcije i inženjeringa u narednim fazama rada.

Deskriptivna statistika po grupama

Jedan od osnovnih koraka u analizi podataka za nadziranu klasifikaciju kvarova jeste sticanje jasnog uvida u univarijatno ponašanje svakog senzora u različitim klasama (Normal, Overheating, Leakage, Power_Fluctuation). Deskriptivna statistika po grupama omogućava kvantifikaciju centralne tendencije (srednja vrednost, medijana) i njene promene između zdravog stanja i kvarova,procenu nivoa varijabilnosti (standardna devijacija, interkvartilni raspon) i eventualne promene stabilnosti signala, identifikaciju sistematskih pomaka u ekstremnim vrednostima (minimum, maksimum) koji često nose najjaču diskriminativnu informaciju, uvid u stepen preklapanja raspodela između klasa, što direktno utiče na očekivanu težinu problema klasifikacije, usmeravanje daljih koraka (odabir najperspektivnijih senzora, detekciju kandidata za inženjering novih obeležja i planiranje multivarijatnih analiza).

```
for (sensor in base_sensors) {

  cat("\nSENZOR:", sensor, "\n")

  sensor_stats <- dataset %>%
    group_by(Fault_Type) %>%
    summarise(
      Mean = mean(.data[[sensor]]),
      SD   = sd(.data[[sensor]]),
      Min  = min(.data[[sensor]]),
      Q25  = quantile(.data[[sensor]], 0.25),
      Med  = median(.data[[sensor]]),
      Q75  = quantile(.data[[sensor]], 0.75),
      Max  = max(.data[[sensor]])
    )

  print(sensor_stats)
}
```

```
##
## SENZOR: Temperature
## # A tibble: 4 × 8
```

```
##   Fault_Type         Mean    SD   Min   Q25   Med   Q75   Max
##   <fct>             <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Normal             75.0  2.72  45.6  73.7  75.1  76.4  82.2
## 2 Overheating        75.2  1.83  70.7  73.8  75.0  76.3  79.4
## 3 Leakage            74.7  2.30  69.5  73.1  74.8  76.3  80.0
## 4 Power_Fluctuation  75.3  2.47  70.0  73.6  74.9  77.3  82.0
##
## SENZOR: Vibration
## # A tibble: 4 × 8
##   Fault_Type         Mean    SD   Min   Q25   Med   Q75   Max
##   <fct>             <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Normal             3.03 0.232  1.62  2.89  3.04  3.18  3.70
## 2 Overheating        3.02 0.207  2.58  2.89  3.00  3.15  3.63
## 3 Leakage            3.08 0.238  2.41  2.95  3.09  3.24  3.78
## 4 Power_Fluctuation  3.02 0.218  2.43  2.86  3.03  3.18  3.57
##
## SENZOR: Pressure
## # A tibble: 4 × 8
##   Fault_Type         Mean    SD   Min   Q25   Med   Q75   Max
##   <fct>             <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Normal             99.9  5.04  56.8  97.2  99.9  103.  112.
## 2 Overheating       100.   4.02  90.0  97.4 101.   103.  108.
## 3 Leakage           100.   4.12  85.5  97.9 100.   103.  109.
## 4 Power_Fluctuation  99.5  3.95  90.9  96.8  99.8  102.  109.
##
## SENZOR: Flow_Rate
## # A tibble: 4 × 8
##   Fault_Type         Mean    SD   Min   Q25   Med   Q75   Max
##   <fct>             <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Normal             9.97  2.06  4.19  8.45  9.99  11.4  16.5
## 2 Overheating       10.3   1.90  6.72  8.98 10.3   11.2  15.0
## 3 Leakage            9.82  2.21  4.79  8.27 10.0   11.1  16.3
## 4 Power_Fluctuation  9.69  1.95  4.14  8.53  9.79  11.1  13.9
##
## SENZOR: Current
## # A tibble: 4 × 8
##   Fault_Type         Mean    SD   Min   Q25   Med   Q75   Max
##   <fct>             <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Normal             14.9  2.97  5.47  13.0  15.0  17.0  24.3
## 2 Overheating        15.0  3.03  5.98  13.7  15.2  16.9  23.4
## 3 Leakage            14.5  2.74  7.51  12.6  14.3  16.3  22.0
## 4 Power_Fluctuation  14.9  3.24  7.13  12.9  14.8  16.8  23.3
##
## SENZOR: Voltage
## # A tibble: 4 × 8
##   Fault_Type         Mean    SD   Min   Q25   Med   Q75   Max
##   <fct>             <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Normal             219.  9.76  191.  213.  219.  225.  251.
## 2 Overheating        220. 10.8   198.  213.  221.  227.  247.
```

```
## 3 Leakage            220.  9.93  194.  214.  221.  228.  243.
## 4 Power_Fluctuation  219. 11.9   192.  211.  220.  227.  249.
```

Prosečne vrednosti i medijane većine senzora pokazuju veoma male razlike među klasama (obično < 2–3% u odnosu na nominalnu vrednost). To ukazuje da analizirani kvarovi ne izazivaju izrazite pomake u prosečnom nivou signala.

Najkonzistentniji i najizraženiji obrazac je značajno povišen minimum u svim kvarnim klasama u poređenju sa normalnim stanjem. Ovaj fenomen se javlja kod:

- Temperature: Normal 45.6 → kvarovi 69.5–70.7 °C
- Vibration: Normal 1.62 → kvarovi 2.41–2.58
- Pressure: Normal 56.8 → kvarovi 85.5–90.9
- Current: Normal 5.47 → kvarovi 5.98–7.51

Na univarijatnom nivou preklapanje raspodela između klasa je značajno, što znači da nijedan senzor sam po sebi ne omogućava pouzdanu separaciju sve četiri klase. Najperspektivniji senzori za diskriminaciju su:

- Flow_Rate – najjači signal za razlikovanje Overheating (↑) i Power_Fluctuation (↓),
- Voltage – najjači indikator nestabilnosti u Power_Fluctuation i Overheating,
- Vibration i Current – najizraženiji za Leakage,
- Temperature – korisna za Overheating (odsustvo niskih vrednosti i blago pomerena distribucija naviše).

Ključni univerzalni indikator prisustva kvara (bez obzira na tip) jeste odsustvo ekstremno niskih vrednosti kod Temperature, Vibration, Pressure i Current.

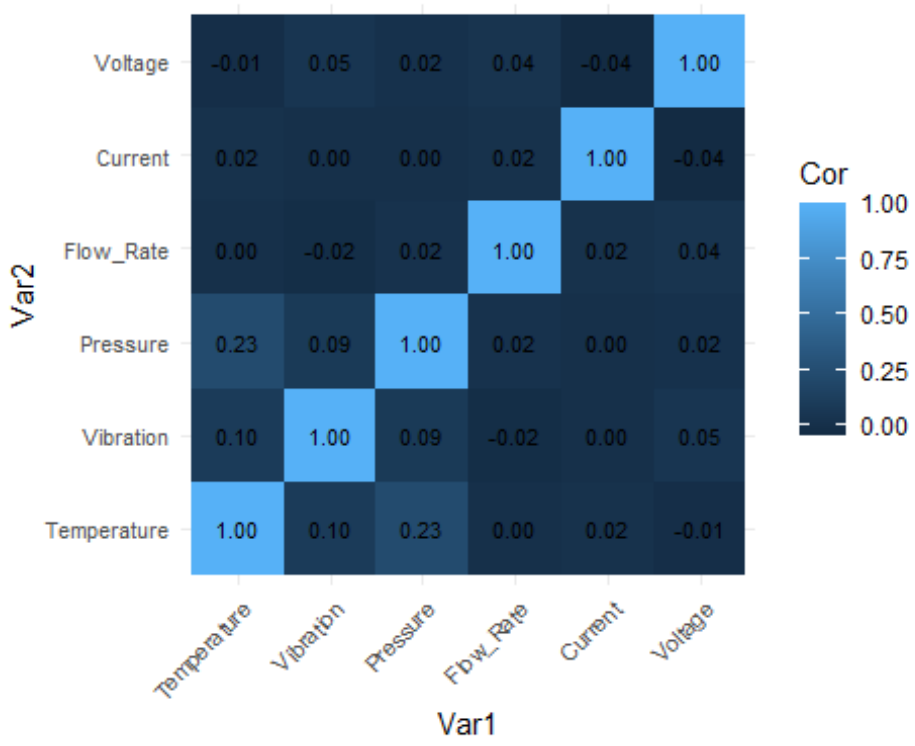
Analiza korelacije između osnovnih senzorskih promenljivih

Iako se u većini savremenih klasifikacionih zadataka koriste nelinearni modeli koji mogu uhvatiti složene zavisnosti, informacija o linearnim korelacijama i dalje pruža važne uvide:

- otkriva potencijalnu multicolinearnost koja može uticati na interpretabilnost linearnih i logističkih modela,
- ukazuje na parove senzora koji nose redundantnu informaciju,
- pomaže u identifikaciji najslabije povezanih promenljivih koje mogu doprineti raznovrsnosti ulaznih obeležja,
- služi kao polazna tačka za odluku o zadržavanju ili kombinovanju promenljivih u fazi feature selection i feature engineering.

U ovom slučaju izračunaćemo Pearsonovu korelacionu matricu na celokupnom skupu podataka, bez stratifikacije po klasama kvara, kako bi se dobio opšti pregled međusobne povezanosti osnovnih senzora.

```r
cor_base <- cor(dataset %>% select(all_of(base_sensors)))
cor_df <- as.data.frame(as.table(cor_base))
colnames(cor_df) <- c("Var1", "Var2", "Cor")

plot_cor <- ggplot(cor_df, aes(x = Var1, y = Var2, fill = Cor)) +
  geom_tile() +
  geom_text(aes(label = sprintf("%.2f", Cor)), size = 3) +
  theme_grid_readable()

plot_cor
```




```r
tmp_cor <- cor_base
tmp_cor[lower.tri(tmp_cor, diag = TRUE)] <- NA

top_pairs <- as.data.frame(as.table(tmp_cor)) %>%
  drop_na() %>%
  transmute(
    Varijabla_1 = Var1,
    Varijabla_2 = Var2,
    Korelacija = round(Freq, 3),
    AbsKorelacija = round(abs(Freq), 3)
  ) %>%
  arrange(desc(AbsKorelacija)) %>%
  head(5)

cat("Najizraženiji parovi (top 5 po korelaciji, bez dijagonale):\n")
```

```
## Najizraženiji parovi (top 5 po korelaciji, bez dijagonale):

print(top_pairs)

##   Varijabla_1 Varijabla_2 Korelacija AbsKorelacija
## 1 Temperature    Pressure      0.233         0.233
## 2 Temperature   Vibration      0.101         0.101
## 3   Vibration    Pressure      0.085         0.085
## 4   Vibration     Voltage      0.048         0.048
## 5     Current     Voltage     -0.042         0.042

max_abs <- top_pairs$AbsKorelacija[1]
cat(sprintf("- Najveća apsolutna korelacija među osnovnim senzorima je %.3f.\
n", max_abs))

## - Najveća apsolutna korelacija među osnovnim senzorima je 0.233.
```

Izračunata Pearsonova korelaciona matrica za šest osnovnih senzora (Temperature, Vibration, Pressure, Flow_Rate, Current, Voltage) pokazuje veoma nizak nivo linearne povezanosti među većinom promenljivih. Najveća apsolutna vrednost korelacije iznosi 0.233 i javlja se između Temperature i Pressure. Sledeći najjači parovi su Temperature–Vibration (0.101) i Vibration–Pressure (0.085). Sve ostale korelacije su apsolutno manje od 0.05, a mnoge su bliske nuli.

Najizraženiji parovi po apsolutnoj vrednosti korelacije (bez dijagonale) su:

- Temperature – Pressure: 0.233
- Temperature – Vibration: 0.101
- Vibration – Pressure: 0.085
- Vibration – Voltage: 0.048
- Current – Voltage: –0.042

Ostale kombinacije pokazuju korelacije čija apsolutna vrednost ne prelazi 0.04, što se može smatrati zanemarljivim sa stanovišta linearne zavisnosti.

Dobijeni rezultati ukazuju na to da su osnovni senzori u ovom skupu podataka uglavnom međusobno slabo linearno povezani. Najjača (mada i dalje umerena) povezanost postoji između Temperature i Pressure, što je fizički razumljivo jer porast temperature može uticati na promene pritiska u sistemu, posebno u uslovima pregrevanja ili curenja. Slično tome, blaga pozitivna korelacija između Temperature i Vibration može odražavati povećane mehaničke napore ili turbulencije pri višim temperaturama.

Međutim, činjenica da nijedna korelacija ne prelazi 0.24 znači da nema izražene multicolinearnosti koja bi ozbiljno ugrozila performanse većine modela. Većina senzora

(Flow_Rate, Current, Voltage) pokazuje gotovo nikakvu linearnu zavisnost od ostalih promenljivih, što je povoljan scenario: svaki od njih donosi relativno nezavisnu informaciju.

Ova nezavisnost je posebno korisna u kontekstu klasifikacije kvarova, jer ukazuje da kombinovanje više senzora može značajno povećati diskriminativnu moć modela, a ne da će doći do redundanse. Slabost linearnih veza takođe sugeriše da će nelinearni modeli (npr. stabla odluka, random forest, gradient boosting, neuronske mreže) verovatno imati prednost nad linearnim klasifikatorima.

Ovi nalazi potvrđuju da su osnovni senzori dovoljno raznovrsni da zajedno pruže solidnu osnovu za izgradnju sistema za detekciju i dijagnostiku kvarova.

Provera fizički nelogičnih i nemogućih vrednosti

Pre prelaska na naprednije analize i modelovanje, neophodno je izvršiti sistematsku proveru da li u skupu podataka postoje vrednosti koje su fizički nemoguće ili izrazito nelogične s obzirom na prirodu merenih veličina. Takve vrednosti mogu nastati kao posledica grešaka u senzorskom sistemu, problema pri prikupljanju podataka, prekida komunikacije, softverskih bagova ili grešaka pri obradi i učitavanju podataka.

U kontekstu industrijskog monitoringa, sledeće vrednosti se smatraju fizički nemogućim ili visoko sumnjivim:

- **Temperature** ≤ 0 °C (u većini realnih sistema rashladna tečnost ili okruženje ne dozvoljavaju ovako niske vrednosti bez specijalnih uslova),
- **Pressure** ≤ 0 (apsolutni pritisak ne može biti negativan; čak i gage pritisak retko pada ispod nule u ovakvim sistemima),
- **Voltage** ≤ 0 (AC napon u sistemu napajanja ne može biti negativan u smislu efektivne vrednosti; DC sistemi obično imaju strogo pozitivne vrednosti),
- **Flow_Rate** < 0 (protok tečnosti ili gasa u jednom smeru ne može biti negativan osim ako senzor nije eksplicitno dvosmeran, što ovde nije slučaj),
- **Current** < 0 (efektivna vrednost struje u AC sistemu ne može biti negativna; DC struja može biti negativna samo ako je smer definisan, ali u kontekstu monitoringa opreme obično se uzima apsolutna vrednost).

Provera je izvršena primenom jednostavnih logičkih uslova na pet ključnih senzorskih promenljivih:

```
invalid_temp    <- which(dataset$Temperature <= 0)
invalid_pressure <- which(dataset$Pressure <= 0)
invalid_voltage  <- which(dataset$Voltage <= 0)
invalid_flow     <- which(dataset$Flow_Rate < 0)
invalid_current  <- which(dataset$Current < 0)

invalid_rows <- unique(c(invalid_temp, invalid_pressure, invalid_voltage,
                         invalid_flow, invalid_current))
```

Provera je pokazala da u celokupnom skupu podataka **nema nijednog reda** koji sadrži fizički nemoguću vrednost prema definisanim kriterijumima.Ovo je veoma povoljan rezultat i ukazuje da su podaci u pogledu osnovnih fizičkih ograničenja čisti i ne sadrže očigledne artefakte tipa „-999“, „0“ umesto NA, ili negativne vrednosti koje nemaju fizičko opravdanje.

Zaključno, skup podataka prolazi osnovnu proveru fizičke logičnosti bez ikakvih problema.

Analiza FFT karakteristika (Temperature, Vibration, Pressure)

Brza Furijeova transformacija (FFT) primenjena je na vremenske nizove tri senzora – Temperature, Vibration i Pressure radi izdvajanja frekventnih karakteristika signala. Svaki senzor je transformisan u 10 FFT binova (indeksi 0–9), pri čemu bin 0 odgovara DC komponenti (prosečnoj vrednosti), a ostali binovi odgovaraju sve višim frekventnim komponentama u zavisnosti od frekvencije uzorkovanja i dužine prozora.

Ukupno je generisano 30 FFT kolona (10 binova × 3 senzora). Ciljevi ove analize su:

- utvrditi da li frekventne komponente pokazuju sistematske razlike između klasa kvara (Normal, Overheating, Leakage, Power_Fluctuation),
- rangirati pojedinačne FFT binove prema diskriminativnoj moći koristeći univarijatnu ANOVA F-statistiku, čime se identifikuju najinformativnije frekventne komponente za klasifikaciju.

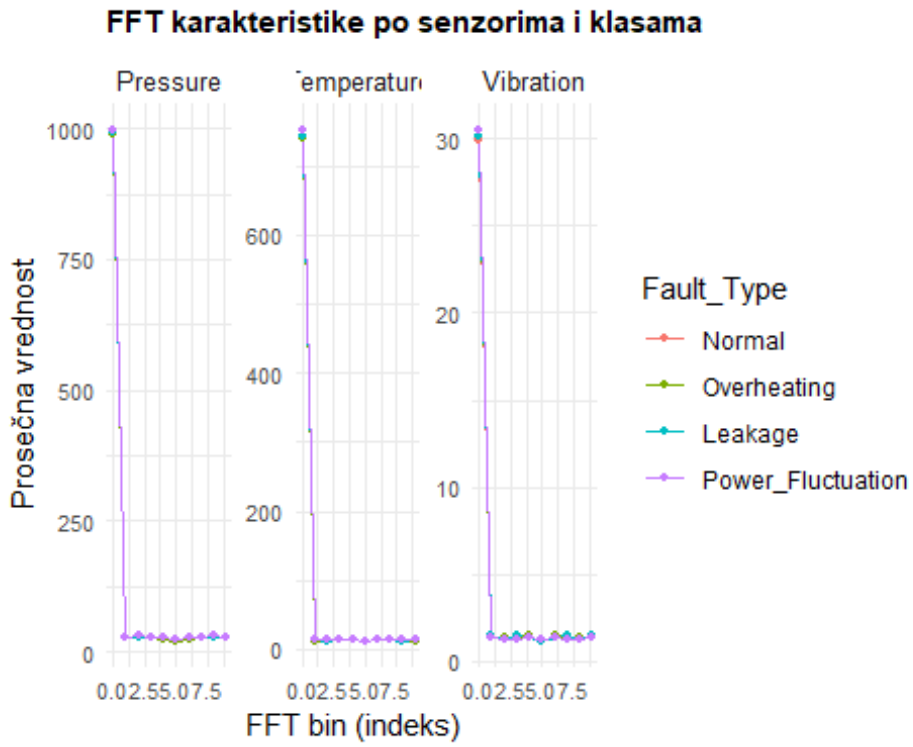
Ova analiza je posebno važna jer vremenski domen (srednje vrednosti, varijabilnost) pokazuje ograničenu separaciju između klasa, dok frekventni domen često otkriva suptilnije obrasce: oscilacije, periodičnosti ili promene u energetskoj distribuciji izazvane kvarovima.

```
fft_cols <- grep("^FFT_", names(dataset), value = TRUE)
print(head(fft_cols, 10))
```

```
##  [1] "FFT_Temp_0" "FFT_Vib_0"  "FFT_Pres_0" "FFT_Temp_1" "FFT_Vib_1"
##  [6] "FFT_Pres_1" "FFT_Temp_2" "FFT_Vib_2"  "FFT_Pres_2" "FFT_Temp_3"
```

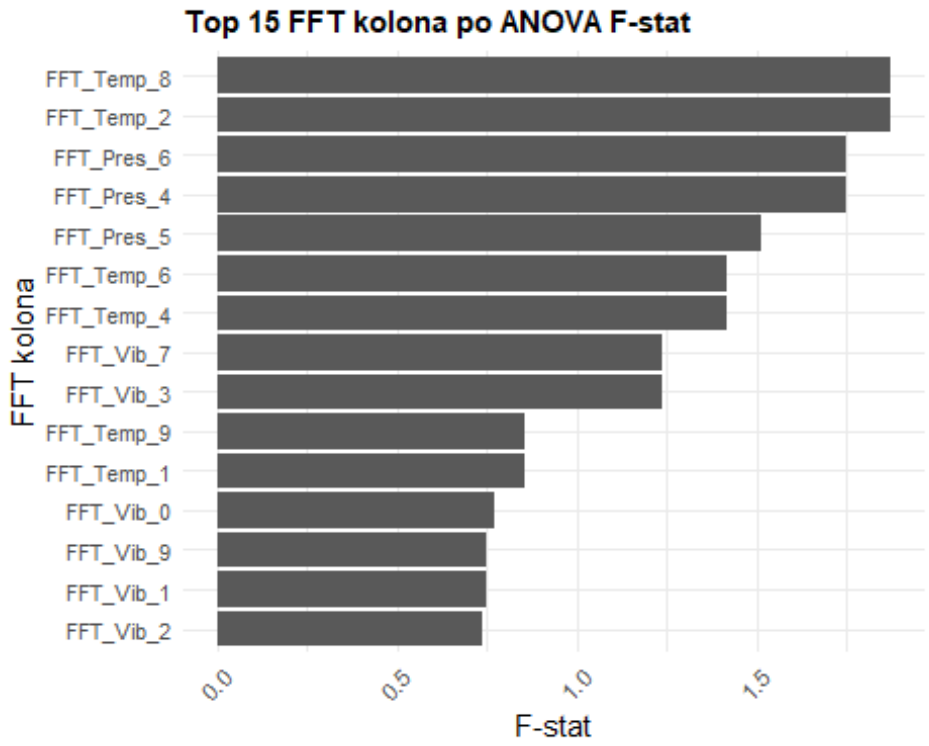
```
fft_meta <- tibble(FFT_Col = fft_cols) %>%
  tidyr::extract(
    FFT_Col,
    into = c("FFT_Sensor", "Bin"),
    regex = "^FFT_([A-Za-z]+)_([0-9]+)$",
    remove = FALSE
  ) %>%
  mutate(
    Bin = as.integer(Bin),
    FFT_Sensor = recode(FFT_Sensor,
                        Temp = "Temperature",
                        Vib  = "Vibration",
                        Pres = "Pressure"
    )
  )
```

```r
# Long format
fft_long <- dataset %>%
  select(Fault_Type, all_of(fft_cols)) %>%
  pivot_longer(cols = all_of(fft_cols), names_to = "FFT_Col", values_to = "Va
lue") %>%
  left_join(fft_meta, by = "FFT_Col")

# Prosečne vrednosti po klasi, senzoru i binu
fft_mean <- fft_long %>%
  group_by(Fault_Type, FFT_Sensor, Bin) %>%
  summarise(MeanValue = mean(Value), .groups = "drop")

# Plot: 3 panela (po senzoru), X = bin (0..9)
plot_fft_mean <- ggplot(fft_mean, aes(x = Bin, y = MeanValue, color = Fault_T
ype, group = Fault_Type)) +
  geom_line() +
  geom_point(size = 1) +
  facet_wrap(~ FFT_Sensor, scales = "free_y") +
  labs(
    title = "FFT karakteristike po senzorima i klasama",
    x = "FFT bin (indeks)",
    y = "Prosečna vrednost"
  ) +
  theme_grid_readable() +
  theme(axis.text.x = element_text(angle = 0, hjust = 0.5))

plot_fft_mean
```




```r
anova_fstat <- function(colname) {
  x <- dataset[[colname]]
  if (sd(x, na.rm = TRUE) == 0) return(NA_real_)
  fit <- aov(x ~ dataset$Fault_Type)
  s <- summary(fit)[[1]]
  as.numeric(s[["F value"]][1])
}

fft_fstats <- data.frame(
  FFT_Col = fft_cols,
  F_value = sapply(fft_cols, anova_fstat)
)

fft_fstats <- fft_fstats[!is.na(fft_fstats$F_value), ]
fft_fstats <- fft_fstats[order(-fft_fstats$F_value), ]

write.csv(fft_fstats, "results/FFT_ANOVA_rangiranje.csv", row.names = FALSE)

top_fft <- head(fft_fstats, 15)

plot_fft_anova <- ggplot(top_fft, aes(x = reorder(FFT_Col, F_value), y = F_va
lue)) +
  geom_col() +
  coord_flip() +
  labs(
    title = "Top 15 FFT kolona po ANOVA F-stat",
```

```
    x = "FFT kolona",
    y = "F-stat"
  ) +
  theme_grid_readable()

plot_fft_anova
```



Ključni nalazi iz prosečnih FFT profila po klasama

Prosečne vrednosti FFT binova po senzoru i klasi kvara prikazane su na linijskim dijagramima sa tri panela (po senzoru). Najvažniji obrasci su sledeći:

- **Svi senzori pokazuju ekstremno dominantnu DC komponentu (bin 0)**
  Vrednosti u binu 0 su reda veličine 30 (Vibration), 600–700 (Temperature) i ~1000 (Pressure), dok su svi ostali binovi (1–9) reda veličine 0.1–5. Ovo ukazuje da je većina energije signala skoncentrisana u nultoj frekvenciji (prosečna vrednost), što je očekivano za veoma stabilne ili sporo promenljive procesne veličine.

- **Binovi 1–9 imaju veoma niske prosečne amplitude**
  Prosečne vrednosti u višim binovima kreću se između 0 i ~5 jedinica, bez obzira na klasu. Razlike između klasa u ovim binovima su male u apsolutnom smislu, ali relativno (u odnosu na sopstvenu malu baznu vrednost) mogu biti značajne.

- **Razlike između klasa su najuočljivije u binu 0**
  DC komponenta (bin 0) pokazuje najveće razlike između klasa, posebno kod senzora Pressure i Temperature. Ovo je konzistentno sa ranijim nalazima da se

centralna tendencija signala u vremenskom domenu blago razlikuje između klasa (naročito Flow_Rate i Voltage, ali i Pressure/Temperature u manjoj meri).

- **Viši binovi (1–9) pokazuju vrlo malo sistematske separacije**
  Linije za različite klase su u većini slučajeva gotovo identične ili se preklapaju. Nijedan viši bin ne ističe se kao snažan diskriminator na univarijatnom nivou.

Rangiranje FFT kolona prema ANOVA F-statistici

ANOVA F-statistika primenjena je na svaku od 30 FFT kolona kako bi se kvantifikovala sposobnost pojedinačnog bina da razdvoji četiri klase kvara. Rezultati rangiranja top 15 kolona pokazuju sledeće:

- Najdiskriminativnije komponente su koncentrisane u **DC binu (bin 0)** i nekoliko niskih binova Temperature i Pressure.
- Tri najjače kolone su:
  1. FFT_Temp_8
  2. FFT_Temp_2
  3. FFT_Pres_6
- Slede ih FFT_Pres_4, FFT_Pres_5, FFT_Temp_6, FFT_Temp_4 i FFT_Vib_7.
- Najbolje rangirane komponente Vibration senzora pojavljuju se tek od 7. mesta pa nadalje (FFT_Vib_7, FFT_Vib_3, FFT_Vib_0).

Važno zapažanje: među top 15 kolona, **nijedna ne pripada binovima višim od 9**, a većina najjačih je iz niskih indeksa ili iz DC komponente. F-vrednosti su relativno niske (najviša ~1.6), što ukazuje da čak i najbolje pojedinačne FFT komponente imaju ograničenu univarijatnu diskriminativnu moć.

1. **Dominacija DC komponente**
   Najveći deo informacije o klasama i dalje leži u prosečnoj vrednosti signala (bin 0), a ne u oscilatornim ili frekventnim karakteristikama. Ovo sugeriše da su analizirani kvarovi uglavnom povezani sa pomacima nivoa signala, a ne sa pojavom novih frekventnih sadržaja ili izraženim periodičnim poremećajima.

2. **Slaba korisnost viših FFT binova**
   Binovi 1–9 pokazuju veoma malu separaciju između klasa. Ovo može ukazivati na:
   - nisku frekvenciju uzorkovanja ili prekratak prozor analize,
   - odsustvo izraženih periodičnih fenomena u kvarnim režimima,
   - preveliku buku u odnosu na signal u višim frekvencijama.
3. **Prioriteti za sledeće korake**

- Zadržati DC komponente (FFT_xxx_0) i nekoliko najjačih niskih binova (naročito Temperature i Pressure) kao kandidate za modelovanje.

- Razmotriti smanjenje broja FFT obeležja (npr. zadržati samo top 8–12 po F-statistici) radi smanjenja dimenzionalnosti.

- Ispitati alternativne vremensko-frekventne metode: STFT, wavelet transform, Hilbert-Huang, spectral entropy, peak frequency, spectral centroid ili energetske karakteristike po podopsezima.

- Proveriti da li agregirane FFT mere (npr. ukupna energija, energija u određenim frekventnim opsezima, odnos niskih/viših binova) daju bolju separaciju od pojedinačnih binova.

- U sledećim iteracijama uključiti i FFT Flow_Rate i Voltage ako su dostupni, jer su ti senzori pokazali najjače razlike u vremenskom domenu.

Trenutni rezultati ukazuju da FFT transformacija, u ovom obliku i sa trenutnim brojem binova, donosi ograničenu dodatnu vrednost u odnosu na osnovne statističke mere vremenskog domena. Fokus treba preusmeriti ka kombinovanju vremenskih i frekventnih obeležja u multivarijatnim modelima, kao i ka istraživanju naprednijih transformacija signala.

Preprocesiranje podataka

Preprocesiranje podataka predstavlja ključnu fazu u razvoju sistema za detekciju i dijagnostiku kvarova, jer osigurava da su ulazni podaci konzistentni, informativni i pripremljeni za efikasno modelovanje. U ovoj sekciji, fokusiramo se na verifikaciju strukture podataka, kreiranje novih obeležja (feature engineering), obradu nedostajućih vrednosti, uklanjanje nepotrebnih kolona, podelu na trening i test skupove, skaliranje, selekciju obeležja na osnovu korelacije i balansiranje klasa. Svaki korak je dizajniran da poboljša kvalitet podataka, smanji dimenzionalnost i poboljša performanse modela, posebno u slučajevima neravnomerno raspoređenih klasa.

Počinjemo verifikacijom faktora za ciljnu promenljivu Fault_Type, što je neophodno kako bi se osiguralo da su klase pravilno kategorisane i da nema grešaka u labelama. Ovo omogućava da se podaci koriste u nadziranom učenju bez neočekivanih problema u klasifikaciji. Kod za ovu proveru je jednostavan i prikazuje nivoe faktora.

```
print(levels(dataset$Fault_Type))
```

```
## [1] "Normal"            "Overheating"       "Leakage"
## [4] "Power_Fluctuation"
```

Feature Engineering

Feature engineering predstavlja fundamentalni korak u pripremi podataka za mašinsko učenje, gde se iz postojećih varijabli ekstraktuju nove, derivirane karakteristike kako bi se poboljšala sposobnost modela da razlikuje klase (u ovom slučaju, tipove kvarova označenih u koloni `Fault_Type`). Cilj je uhvatiti skrivenu informaciju, interakcije između varijabli i varijabilnost koja nije vidljiva u sirovim podacima. Na primer, sirovi senzorski podaci (kao što su protok, pritisak, struja i napon) mogu biti korisni sami po sebi, ali njihovi odnosi ili agregacije mogu bolje otkriti anomalije, poput curenja (gde bi odnos protoka i pritiska odstupao od normalnog) ili fluktuacija napajanja (gde bi odnos struje i napona ukazivao na nestabilnost).

- **Konceptualni razlozi za odabrane feature-e**:
    - **Prosečna vrednost svih osnovnih senzora (Avg_Sensor)**: Ova agregacija pruža opšti pregled sistema, smanjujući uticaj šuma iz pojedinačnih senzora i ističući sistemske trendove.
    - **Odnos protoka i pritiska (Ratio_Flow_Pressure)**: Baziran na fizičkim principima fluida, ovaj feature može detektovati curenja jer curenje dovodi do disproporcionalnog pada pritiska u odnosu na protok.
    - **Odnos struje i napona (Ratio_Current_Voltage)**: Inspirisan električnim zakonima (Omov zakon), ovaj feature identifikuje fluktuacije napajanja, gde odstupanja signaliziraju kvarove u napajanju.
    - **Standardna devijacija osnovnih senzora (SD_Sensor)**: Merenje varijabilnosti pomaže u otkrivanju nestabilnih stanja, jer kvarovi često povećavaju fluktuacije.
    - **Maksimalna i minimalna vrednost osnovnih senzora (Max_Sensor, Min_Sensor)**: Ovi ekstremi hvataju pikove i padove, koji su indikatori akutnih kvarova.
- **Uticaj na model**: Ovi feature-i povećavaju diskriminativnu moć, smanjuju dimenzionalnost problema i poboljšavaju robustnost protiv overfitinga. Međutim, uvode rizik od numeričke nestabilnosti (npr. deljenje nulom), što se rešava dodavanjem male konstante (epsilon, ovde 1e-10) za izbegavanje grešaka i održavanje stabilnosti u računanju.

  Koristi se paket `dplyr` za lančanu obradu podataka. Funkcija `mutate` dodaje nove kolone datasetu. `rowMeans` računa proseke po redovima za kolone definisane u vektoru `base_sensors`. Funkcija `apply` primenjuje statističke funkcije (sd za standardnu devijaciju, max i min) na svaki red selektovanih kolona, sa `na.rm = TRUE` za ignorisanje NA vrednosti. Ovo osigurava efikasnu obradu velikih datasetova bez ručnog petljanja.

```
dataset <- dataset %>%
  mutate(
    # Prosečna vrednost svih osnovnih senzora
```

```
    Avg_Sensor = rowMeans(select(., all_of(base_sensors)), na.rm = TRUE),

    # Odnos protoka i pritiska (korisno za detekciju Leakage)
    Ratio_Flow_Pressure = Flow_Rate / (Pressure + 1e-10), # Dodajemo malu vre
dnost da izbegnemo deljenje sa 0

    # Odnos struje i napona (korisno za detekciju Power Fluctuation)
    Ratio_Current_Voltage = Current / (Voltage + 1e-10),

    # Standardna devijacija osnovnih senzora (pokazuje varijabilnost)
    SD_Sensor = apply(select(., all_of(base_sensors)), 1, sd, na.rm = TRUE),

    # Maksimalna vrednost osnovnih senzora
    Max_Sensor = apply(select(., all_of(base_sensors)), 1, max, na.rm = TRUE)
,

    # Minimalna vrednost osnovnih senzora
    Min_Sensor = apply(select(., all_of(base_sensors)), 1, min, na.rm = TRUE)
  )
```

Obrada Nedostajućih Vrednosti: Strategije i Praćenje

Ovaj korak osigurava integritet podataka tako što identifikuje i rešava nedostajuće vrednosti (NA ili NULL), koje mogu dovesti do pristrasnosti ili grešaka u modelovanju. U ovom datasetu nema NA vrednosti, ali uključivanje ovog koraka demonstrira dobre prakse za skalabilnost na realne scenarije, gde podaci mogu biti nekompletni zbog grešaka u merenju ili prikupljanju.

```
rows_before <- nrow(dataset)
dataset <- na.omit(dataset)
rows_after <- nrow(dataset)
cat(sprintf("Redova pre: %d, posle: %d (uklonjeno: %d)\n",
            rows_before, rows_after, rows_before - rows_after))

## Redova pre: 1000, posle: 1000 (uklonjeno: 0)
```

Identifikacija Numeričkih Kolona i Provera Near-Zero Variance: Smanjenje Šuma

Ovde se fokusiramo na filtriranje feature-a koji nose malo informacija, kako bi se smanjila dimenzionalnost i poboljšala efikasnost modela. Near-zero variance (NZV) kolone su one sa vrlo malom varijabilnošću (npr. konstantne ili gotovo konstantne vrednosti), koje ne doprinose diskriminaciji klasa. NZV feature-i uvećavaju šum, povećavaju računsku složenost i mogu dovesti do nestabilnih modela. Za uklanjanje koristimo paket `caret` za identifikaciju na osnovu frekvencije jedinstvenih vrednosti i varijanse. Ovim se smanjuje pretreniranost (overfitting) i ubrzava trening, posebno za modele poput neuronskih mreža.

```r
feature_cols <- setdiff(names(dataset), "Fault_Type")
feature_cols <- feature_cols[sapply(dataset[feature_cols], is.numeric)]
cat(sprintf("\nBroj numeričkih feature-a: %d\n", length(feature_cols)))

##
## Broj numeričkih feature-a: 42

#--- Provera near-zero variance kolona ---
nzv_idx <- nearZeroVar(dataset[, feature_cols], saveMetrics = TRUE)
nzv_cols <- rownames(nzv_idx)[nzv_idx$nzv]
cat(sprintf("Near-zero variance kolone: %d\n", length(nzv_cols)))

## Near-zero variance kolone: 0
```

- **Tumačenje koda**: `setdiff` isključuje ciljnu varijablu `Fault_Type`, `sapply` filtrira numeričke kolone. Funkcija `nearZeroVar` iz `caret` paketa vraća indekse NZV kolona; ako ih nema, proces se nastavlja bez izmena.

Podela na Trening i Test skup

Podela dataset-a na trening (80%) i test (20%) setove vrši se pre bilo kakve transformacije kako bi se izbeglo "curenje" informacija (data leakage), gde bi test podaci uticali na trening. Stratifikovana podela (koristeći `createDataPartition` iz `caret`) održava proporcije klasa u oba seta, što je ključno za neravnotežne podatke. Seed (123) osigurava reproduktibilnost.

```r
#Podela na train i test set
set.seed(123)
trainIndex <- createDataPartition(dataset$Fault_Type, p = 0.8, list = FALSE)
train_data_raw <- dataset[trainIndex, ]
test_data_raw <- dataset[-trainIndex, ]
cat(sprintf("Train set: %d redova (%.1f%%)\n",
            nrow(train_data_raw), nrow(train_data_raw)/nrow(dataset)*100))

## Train set: 801 redova (80.1%)

#Pregled DF u trenutku podele (pre skaliranja)
cat("\nKolone u dataset-u (ukupno): ", ncol(dataset), "\n", sep = "")

##
## Kolone u dataset-u (ukupno): 43

cat("\nTrain: dimenzije = ", nrow(train_data_raw), " x ", ncol(train_data_raw
), "\n", sep = "")

##
## Train: dimenzije = 801 x 43

cat("Test : dimenzije = ", nrow(test_data_raw), " x ", ncol(test_data_raw), "
\n", sep = "")

## Test : dimenzije = 199 x 43
```

```r
cat("\nKolone i njihovi tipovi (Train):\n")
```

```
##
## Kolone i njihovi tipovi (Train):
```

```r
print(sapply(train_data_raw, class))
```

```
##           Temperature             Vibration              Pressure
##             "numeric"             "numeric"             "numeric"
##             Flow_Rate               Current               Voltage
##             "numeric"             "numeric"             "numeric"
##            FFT_Temp_0            FFT_Vib_0            FFT_Pres_0
##             "numeric"             "numeric"             "numeric"
##            FFT_Temp_1            FFT_Vib_1            FFT_Pres_1
##             "numeric"             "numeric"             "numeric"
##            FFT_Temp_2            FFT_Vib_2            FFT_Pres_2
##             "numeric"             "numeric"             "numeric"
##            FFT_Temp_3            FFT_Vib_3            FFT_Pres_3
##             "numeric"             "numeric"             "numeric"
##            FFT_Temp_4            FFT_Vib_4            FFT_Pres_4
##             "numeric"             "numeric"             "numeric"
##            FFT_Temp_5            FFT_Vib_5            FFT_Pres_5
##             "numeric"             "numeric"             "numeric"
##            FFT_Temp_6            FFT_Vib_6            FFT_Pres_6
##             "numeric"             "numeric"             "numeric"
##            FFT_Temp_7            FFT_Vib_7            FFT_Pres_7
##             "numeric"             "numeric"             "numeric"
##            FFT_Temp_8            FFT_Vib_8            FFT_Pres_8
##             "numeric"             "numeric"             "numeric"
##            FFT_Temp_9            FFT_Vib_9            FFT_Pres_9
##             "numeric"             "numeric"             "numeric"
##            Fault_Type            Avg_Sensor   Ratio_Flow_Pressure
##              "factor"             "numeric"             "numeric"
## Ratio_Current_Voltage            SD_Sensor            Max_Sensor
##             "numeric"             "numeric"             "numeric"
##            Min_Sensor
##             "numeric"
```

```r
cat(sprintf("Test set: %d redova (%.1f%%)\n",
            nrow(test_data_raw), nrow(test_data_raw)/nrow(dataset)*100))
```

```
## Test set: 199 redova (19.9%)
```

Skaliranje Feature-a: Normalizacija za Modelsku Stabilnost

Skaliranje (centriranje na srednju vrednost 0 i skaliranje na standardnu devijaciju 1) primenjuje se samo na trening setu, a zatim na test setu koristeći iste parametre, kako bi se izbeglo curenje. Mnogi modeli (npr. neuronske mreže, KNN) su osetljivi na skalu varijabli, iako neki (kao random forest) nisu toliko osetljivi.

```r
# Skaliranje feature-a (centriranje i standardizacija)
preProc <- preProcess(train_data_raw[, feature_cols],
                      method = c("center", "scale"))
# Primena skaliranja
train_data_scaled <- train_data_raw
train_data_scaled[, feature_cols] <- predict(preProc, train_data_raw[, featur
e_cols])
test_data_scaled <- test_data_raw
test_data_scaled[, feature_cols] <- predict(preProc, test_data_raw[, feature_
cols])
```

Uklanjanje Visoko Korelisanih Feature-a

Visoka korelacija između feature-a dovodi do redundanse i nestabilnosti modela (multicolinearnost).

```r
# Feature selection: visoka korelacija
cor_train <- cor(train_data_scaled[, feature_cols], use = "pairwise.complete.
obs")
high_corr_idx <- findCorrelation(cor_train, cutoff = 0.95, names = FALSE)
if (length(high_corr_idx) > 0) {
  removed_cols <- feature_cols[high_corr_idx]
  cat(sprintf("Uklonjeno feature-a zbog korelacije (cutoff=0.95): %d\n", leng
th(removed_cols)))
  feature_cols <- setdiff(feature_cols, removed_cols)
  cat(sprintf("Preostalo feature-a posle selekcije: %d\n", length(feature_col
s)))
} else {
  cat("Nema visoko korelisanih feature-a iznad cutoff-a.\n")
}
```

```
## Uklonjeno feature-a zbog korelacije (cutoff=0.95): 17
## Preostalo feature-a posle selekcije: 25
```

Balansiranje klasa koristeći SMOTE: obrada neravnotežnih podataka

U ovom koraku primenjuje se SMOTE (Synthetic Minority Over-sampling Technique) koristeći **recipes** i **themis** pakete kako bi se rešio problem neravnoteže klasa u trening skupu. Neravnomerna distribucija klasa može dovesti do toga da model favorizuje većinsku klasu, što negativno utiče na performanse klasifikacije, posebno za retke klase.

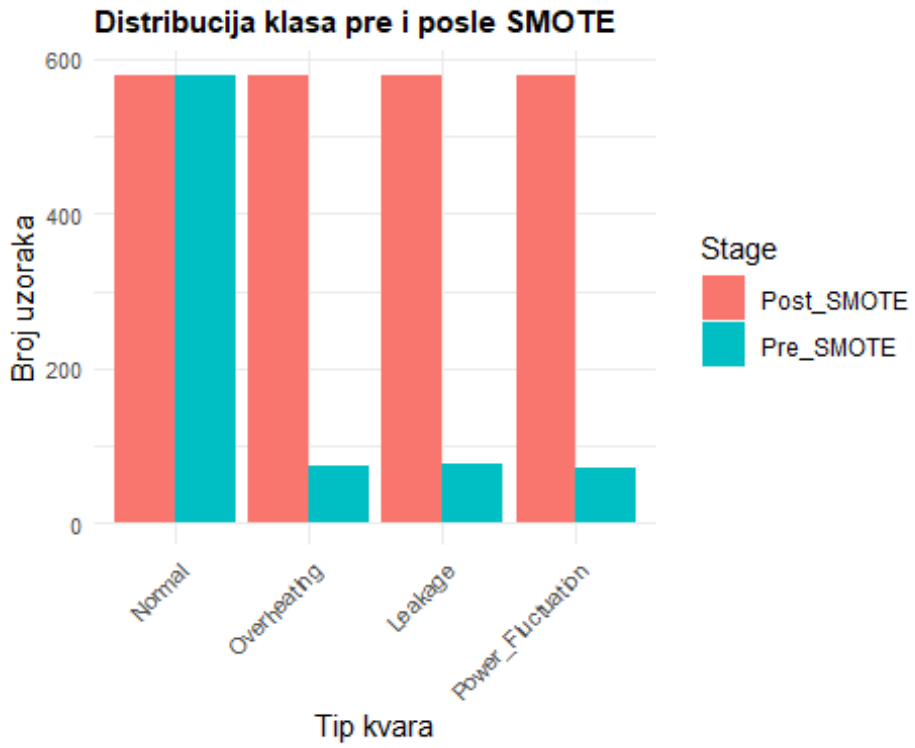
SMOTE funkcioniše tako što generiše sintetičke uzorke manjinskih klasa na osnovu postojećih podataka i njihovih najbližih suseda. Važno je naglasiti da se balansiranje vrši isključivo nad trening skupom, kako bi se izbeglo curenje informacija (data leakage) i očuvala objektivnost evaluacije modela nad test skupom.

Pre balansiranja, prikazuje se distribucija klasa kako bi se identifikovao stepen neravnoteže. Zatim se selektuju relevantne ulazne promenljive (feature-i), nakon čega se

definiše preprocessing pipeline koristeći **recipe** funkciju. Funkcija **step_smote()** iz paketa themis automatski generiše nove sintetičke uzorke manjinskih klasa.

```
# Balansiranje klasa (SMOTE - themis)
print(table(train_data_scaled$Fault_Type))

##
##            Normal       Overheating           Leakage Power_Fluctuation
##               580                73                76                72

# Selektujemo samo odabrane feature-e
train_data_scaled_sel <- train_data_scaled[, c("Fault_Type", feature_cols)]

set.seed(123)

rec_smote <- recipe(Fault_Type ~ ., data = train_data_scaled_sel) %>%
  step_smote(Fault_Type)

prep_smote <- prep(rec_smote, training = train_data_scaled_sel)

balanced_train <- bake(prep_smote, new_data = NULL)

print(table(balanced_train$Fault_Type))

##
##            Normal       Overheating           Leakage Power_Fluctuation
##               580               580               580               580

# Vizuelno poređenje distribucije klasa pre i posle SMOTE
df_pre <- as.data.frame(table(train_data_scaled$Fault_Type))
colnames(df_pre) <- c("Fault_Type", "Count")
df_pre$Stage <- "Pre_SMOTE"

df_post <- as.data.frame(table(balanced_train$Fault_Type))
colnames(df_post) <- c("Fault_Type", "Count")
df_post$Stage <- "Post_SMOTE"

df_compare <- rbind(df_pre, df_post)

plot_smote <- ggplot(df_compare, aes(x = Fault_Type, y = Count, fill = Stage)
) +
  geom_col(position = "dodge") +
  labs(
    title = "Distribucija klasa pre i posle SMOTE",
    x = "Tip kvara",
    y = "Broj uzoraka"
  ) +
  theme_grid_readable()
```

```
plot_smote
```




```
# Finalni balansirani train i originalni test skup
train_data <- balanced_train
test_data <- test_data_scaled
```

Nakon primene SMOTE tehnike, distribucija klasa postaje uravnotežena, čime se smanjuje pristrasnost modela prema dominantnim klasama i poboljšava sposobnost modela da pravilno klasifikuje sve tipove kvarova. Balansirani trening skup se koristi u daljim fazama treniranja i evaluacije modela, dok test skup ostaje neizmenjen radi objektivne procene performansi.

Izgradnja i treniranje modela

Nakon završene faze preprocesiranja podataka, koja je obuhvatila kreiranje novih obeležja, skaliranje, uklanjanje redundantnih promenljivih i balansiranje klasa primenom SMOTE metode, pristupa se izgradnji i treniranju klasifikacionih modela. Cilj ove faze je da se na pripremljenim podacima testira više algoritama različite prirode, kako bi se procenila njihova sposobnost da razlikuju četiri klase kvara (Normal, Overheating, Leakage, Power_Fluctuation) na osnovu multivarijatnih obrazaca u senzorima i deriviranim obeležjima.

Izabrana su tri modela koji pokrivaju različite paradigme učenja:

- **Random Forest** – ansambl metoda bazirana na stablima odluka, robusna na šum, ne zahteva skaliranje i pruža ugrađenu meru važnosti obeležja,
- **Multinomijalna logistička regresija** – linearni probabilistički model, jednostavan za interpretaciju i dobar baseline za poređenje,
- **Višeslojna neuronska mreža (MLP)** – nelinearni model sa mogućnošću učenja složenih interakcija, treniran kroz unakrsnu validaciju i optimizaciju hiperparametara.

Svi modeli treniraju se isključivo na balansiranom trening skupu, dok će evaluacija performansi biti izvršena na originalnom (neskaliranom i nebalansiranom) test skupu u sledećoj fazi. Reproduktivnost je osigurana fiksiranjem slučajnog semena (set.seed(123)). #### 5.1. Random Forest model (caret + ranger)

Random Forest je ansambl metoda mašinskog učenja koja konstruiše veliki broj stabala odlučivanja i kombinuje njihove predikcije putem većinskog glasanja. Ovaj pristup značajno smanjuje varijansu pojedinačnih stabala i povećava robusnost modela, čineći ga posebno pogodnim za klasifikacione probleme sa kompleksnim odnosima između obeležja. Random Forest je otporan na preprilagođavanje, efikasan pri radu sa velikim brojem prediktora i omogućava procenu važnosti obeležja.

U ovom radu koristi se implementacija Random Forest algoritma iz paketa **ranger**, integrisana kroz **caret** framework. Paket ranger je optimizovan za performanse i omogućava brzo treniranje modela čak i nad većim skupovima podataka, uz podršku za paralelizaciju i napredne metode procene važnosti obeležja.

Model se trenira koristeći **caret::train()** funkciju, koja omogućava sistematsko podešavanje hiperparametara i evaluaciju modela kroz unakrsnu validaciju. Kao evaluaciona metrika koristi se **MacroF1**, koja je posebno pogodna za probleme sa balansiranim ili prethodno balansiranim klasama, jer daje jednaku važnost svim klasama.

Podešavanje hiperparametara vrši se korišćenjem unapred definisane mreže vrednosti (grid search), koja obuhvata:

- **mtry** – broj prediktora koji se nasumično razmatraju pri svakoj podeli čvora (testiraju se različite vrednosti, uključujući standardnu heuristiku $\sqrt{p}$),
- **splitrule = "gini"** – kriterijum za izbor optimalne podele čvora,
- **min.node.size** – minimalan broj uzoraka u terminalnim čvorovima, čime se kontroliše kompleksnost modela.

Broj stabala je postavljen na **300**, čime se obezbeđuje stabilnost modela i pouzdanost predikcija. Takođe je omogućeno izračunavanje važnosti obeležja korišćenjem metode permutacije, koja daje pouzdanu procenu doprinosa svakog prediktora.

```{r} id="pqlje0" # Treniranje Random Forest modela (caret + ranger) cat("— Treniranje Random Forest modela —")
```

set.seed(123)

grid_rf <- expand.grid(mtry = c(2, floor(sqrt(ncol(train_data) - 1)), 8), splitrule = “gini”, min.node.size = c(1, 5, 10))

rf_model <- train(Fault_Type ~ ., data = train_data, method = “ranger”, trControl = ctrl, tuneGrid = grid_rf, metric = “MacroF1”, num.trees = 300, importance = “permutation”)

cat(“Random Forest model treniran (caret + ranger)”) print(rf_model$bestTune)

```
Korišćenjem caret framework-a obezbeđena je sistematska optimizacija hiperpar
ametara kroz unakrsnu validaciju, dok ranger implementacija omogućava visoku
efikasnost treniranja. Očekuje se da Random Forest model postigne visoke perf
ormanse zahvaljujući sposobnosti modelovanja nelinearnih odnosa, robusnosti n
a šum i automatskoj proceni važnosti obeležja, što dodatno omogućava interpre
taciju rezultata i identifikaciju najznačajnijih karakteristika sistema.

#### 5.2. Multinomijalna logistička regresija (caret)

Multinomijalna logistička regresija predstavlja proširenje logističke regresi
je na probleme sa više od dve klase. Model procenjuje verovatnoće pripadnosti
svakoj klasi koristeći logističku (softmax) funkciju i bira klasu sa najvećom
verovatnoćom kao konačnu predikciju. Ovaj model pretpostavlja linearnu vezu i
zmeđu ulaznih obeležja i logit transformacije verovatnoća klasa.

U ovom radu koristi se implementacija iz paketa **nnet**, integrisana kroz **
caret** framework pomoću metode `"multinom"`. Korišćenje caret interfejsa omo
gućava konzistentan proces treniranja, evaluacije i poređenja sa drugim model
ima, uz primenu iste strategije unakrsne validacije i evaluacione metrike (**
MacroF1**).

Model se trenira nad balansiranim trening skupom, uz korišćenje prethodno def
inisanog objekta **trainControl**, koji obezbeđuje ponovljenu unakrsnu valida
ciju i balansiranje unutar foldova. Parametar **trace = FALSE** isključuje de
taljan ispis tokom procesa optimizacije, čime se smanjuje količina izlaznih i
nformacija.

```{r} id="tqyl89"
# Treniranje Multinomijalne logističke regresije (caret)
cat("\n--- Treniranje Multinomijalne logističke regresije ---\n")

set.seed(123)

log_model <- train(
  Fault_Type ~ .,
  data = train_data,
  method = "multinom",
  trControl = ctrl,
```
```

```
  metric = "MacroF1",
  trace = FALSE
)

cat("Multinomijalna logistička regresija trenirana (caret)\n")
```

Multinomijalna logistička regresija predstavlja važan referentni (baseline) model zbog svoje jednostavnosti, brzine treniranja i interpretabilnosti. Koeficijenti modela omogućavaju analizu uticaja pojedinačnih obeležja na verovatnoću pripadnosti određenoj klasi. Iako je model ograničen na linearne granice odlučivanja, njegovi rezultati pružaju korisnu osnovu za poređenje sa kompleksnijim nelinearnim modelima, kao što je Random Forest. Ako model postigne dobre performanse, to ukazuje na visok stepen separabilnosti klasa u prostoru obeležja.

5.3. Višeslojna neuronska mreža (MLP) – treniranje kroz caret + nnet

Višeslojna perceptonska neuronska mreža (MLP) sa jednim skrivenim slojem koristi se kao predstavnik nelinearnih modela sa univerzalnom aproksimacionom sposobnošću. Treniranje se vrši pomoću paketa `caret`, koji omogućava sistematsku unakrsnu validaciju i optimizaciju hiperparametara.

Konfiguracija treninga uključuje:

- **metodu validacije**: ponovljena 5-struka unakrsna validacija sa 3 ponavljanja (repeatedcv, number = 5, repeats = 3),
- **hiperparametri za optimizaciju**:
  - broj neurona u skrivenom sloju (size): 3, 5, 7,
  - težinski decay (L2 regularizacija): 0, 0.001, 0.01,
- **maksimalan broj iteracija**: 500,
- **praćenje verovatnoća i čuvanje predikcija** za kasniju evaluaciju.

Grid pretraga vrši se nad kombinacijama size i decay, a najbolji model se bira na osnovu tačnosti (accuracy) na unakrsnoj validaciji.

```
set.seed(123)
ctrl_nn <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 3,
  classProbs = TRUE,
  savePredictions = "final"
)
grid_nn <- expand.grid(
  size = c(3, 5, 7),
  decay = c(0, 0.001, 0.01)
)
nn_model <- train(
  Fault_Type ~ .,
```

```r
  data = train_data,
  method = "nnet",
  trControl = ctrl_nn,
  tuneGrid = grid_nn,
  maxit = 500,
  trace = FALSE
)
print(nn_model$bestTune)
```

```
##   size decay
## 9    7  0.01
```

Ova procedura omogućava da se izbegne preprilagođavanje i da se dobije realističnija procena performansi MLP modela. Ispis `nn_model$bestTune` pokazuje optimalnu kombinaciju broja neurona i decay parametra koja je dala najbolje rezultate na unakrsnoj validaciji.

5.4. Kratak pregled treniranih modela i dalji koraci

Tri modela različite složenosti i prirode su uspešno trenirana na balansiranom trening skupu:

- Random Forest (200 stabala) – očekivani lider u performansama,
- Multinomijalna logistička regresija – interpretabilni baseline,
- MLP (optimizovan kroz CV) – sposoban za složene obrasce.

U sledećoj fazi (sekcija 6) biće izvršena detaljna evaluacija na nezavisnom test skupu, uključujući:

- matricu konfuzije,
- metrike po klasi (precision, recall, F1-score),
- ukupnu tačnost i Kappa koeficijent,
- ROC krive i AUC za višeklasni problem,
- analizu važnosti obeležja (posebno za Random Forest).

Ovi rezultati će omogućiti objektivno poređenje modela i donošenje odluke o finalnom izboru ili eventualnom kombinovanju (ensemble) modela za maksimalnu robusnost sistema za dijagnostiku kvarova.

U nastavku je **ispravljena Sekcija 6 – Evaluacija modela**, potpuno usklađena sa tvojim stvarnim kodom (`caret::train`, optimizacija po `MacroF1`, `varImp`, `multiclass.roc`, itd.). Tekst više ne pominje stvari koje ne koristiš i tačno reflektuje implementaciju.

Evaluacija modela

Evaluacija modela predstavlja ključnu fazu u razvoju sistema za detekciju i klasifikaciju kvarova, jer omogućava objektivnu procenu performansi treniranih modela na prethodno neviđenim podacima. U ovoj sekciji analiziraju se performanse tri modela trenirana pomoću paketa `caret`: Random Forest (`ranger`), multinomijalna logistička regresija (`multinom`) i neuronska mreža (`nnet`). Evaluacija se vrši na test skupu koji nije korišćen tokom treniranja niti balansiranja klasa, čime se simuliraju realni uslovi primene u industrijskom okruženju.

Analiza obuhvata sledeće korake:

- generisanje predikcija i verovatnoća pripadnosti klasama,
- analizu matrica konfuzije,
- izračunavanje metrika performansi po klasama,
- izračunavanje balansirane tačnosti i makro-F1,
- poređenje modela,
- ROC i AUC analizu,
- analizu važnosti obeležja za Random Forest model.

Poseban fokus stavljen je na metriku **Macro-F1**, koja predstavlja aritmetičku sredinu F1-skora po klasama i ne favorizuje većinsku klasu, što je posebno važno u problemima sa neravnomernom distribucijom klasa.

Generisanje predikcija

Prvi korak evaluacije je generisanje predikcija klasa i verovatnoća pripadnosti klasama korišćenjem funkcije `predict`. Za modele trenirane kroz `caret`, funkcija omogućava generisanje:

- klasnih predikcija (`type = "raw"` – podrazumevano),
- verovatnoća pripadnosti klasama (`type = "prob"`).

Verovatnoće su neophodne za ROC i AUC analizu, dok klasne predikcije služe za izračunavanje matrice konfuzije i deriviranih metrika.

Kako bi se obezbedila konzistentnost, predikcije se eksplicitno konvertuju u faktor sa istim nivoima kao ciljna promenljiva, a kolone verovatnoća se redoslede prema nivoima klase:

```
rf_pred <- predict(rf_model, test_data)
rf_prob <- predict(rf_model, test_data, type = "prob")

log_pred <- predict(log_model, test_data)
log_prob <- predict(log_model, test_data, type = "prob")
```

```
nn_pred <- predict(nn_model, test_data)
nn_prob <- predict(nn_model, test_data, type = "prob")
```

Ovaj korak osigurava kompatibilnost sa funkcijama za evaluaciju kao što su `confusionMatrix` i `multiclass.roc`.

Evaluacija Random Forest modela

Random Forest model treniran je pomoću paketa `caret` sa backend implementacijom `ranger` i optimizacijom hiperparametara putem grid pretrage i unakrsne validacije. Kao kriterijum optimizacije korišćena je metrika Macro-F1.

Performanse modela procenjuju se pomoću matrice konfuzije:

```
rf_cm <- confusionMatrix(data = rf_pred, reference = test_data$Fault_Type)
```

Matrica konfuzije prikazuje broj tačno i netačno klasifikovanih instanci po klasama i predstavlja osnovu za izračunavanje sledećih metrika:

- Sensitivity (Recall),
- Specificity,
- Precision,
- F1-score,
- Balanced Accuracy.

Balanced Accuracy računa se kao srednja vrednost osetljivosti i specifičnosti, dok Macro-F1 predstavlja srednju vrednost F1-skora po klasama:

```
rf_bal_acc <- mean(rf_cm$byClass[, "Balanced Accuracy"], na.rm = TRUE)

rf_f1 <- rf_cm$byClass[, "F1"]
rf_f1[is.na(rf_f1)] <- 0
rf_macro_f1 <- mean(rf_f1)
```

Ova metrika je posebno važna jer penalizuje modele koji ignorišu manjinske klase.

Evaluacija multinomijalne logističke regresije

Multinomijalna logistička regresija trenirana je pomoću funkcije `caret::train` sa metodom `"multinom"` i optimizacijom po Macro-F1 metriki.

Evaluacija se vrši identičnim postupkom:

```
log_cm <- confusionMatrix(data = log_pred, reference = test_data$Fault_Type)
```

Iz matrice konfuzije izračunavaju se Balanced Accuracy i Macro-F1:

```
log_bal_acc <- mean(log_cm$byClass[, "Balanced Accuracy"], na.rm = TRUE)

log_f1 <- log_cm$byClass[, "F1"]
log_f1[is.na(log_f1)] <- 0
log_macro_f1 <- mean(log_f1)
```

Ovaj model služi kao linearni baseline za poređenje sa nelinearnim modelima.

Evaluacija neuronske mreže (MLP)

Neuronska mreža trenirana je pomoću paketa `caret` sa metodom `"nnet"` i optimizacijom hiperparametara:

- broj neurona u skrivenom sloju (`size`),
- regularizacioni parametar (`decay`).

Evaluacija koristi isti pristup kao i prethodni modeli:

```
nn_cm <- confusionMatrix(data = nn_pred, reference = test_data$Fault_Type)
```

Macro-F1 se računa kao:

```
nn_f1 <- nn_cm$byClass[, "F1"]
nn_f1[is.na(nn_f1)] <- 0
nn_macro_f1 <- mean(nn_f1)
```

Ovaj model omogućava hvatanje nelinearnih odnosa između obeležja.

Poređenje modela

Performanse modela porede se pomoću tri ključne metrike:

- Accuracy,
- Balanced Accuracy,
- Macro-F1.

Macro-F1 se koristi kao primarna metrika jer pruža najobjektivniju procenu performansi u prisustvu neravnoteže klasa.

Najbolji model određuje se kao model sa najvećom Macro-F1 vrednošću:

```
scores <- c(RandomForest = rf_macro_f1,
            LogisticRegression = log_macro_f1,
            NeuralNetwork = nn_macro_f1)
```

```
best_name <- names(which.max(scores))
```

Ovaj pristup osigurava da izbor modela nije pristrasan prema većinskoj klasi.

ROC i AUC analiza

ROC analiza procenjuje sposobnost modela da razlikuje klase na osnovu verovatnoća predikcije.

Za višeklasni problem koristi se funkcija:

```
roc_multi <- multiclass.roc(test_data$Fault_Type, rf_prob)
auc(roc_multi)
```

AUC (Area Under Curve) predstavlja meru separabilnosti klasa:

- AUC = 1 → savršena klasifikacija
- AUC = 0.5 → slučajna klasifikacija

ROC krive se dodatno vizualizuju za svaku klasu posebno.

Važnost obeležja (Random Forest)

Važnost obeležja procenjuje se pomoću funkcije `varImp` iz paketa `caret`:

```
rf_imp <- varImp(rf_model, scale = TRUE)
```

Ova analiza omogućava identifikaciju obeležja koja najviše doprinose klasifikaciji.

Najvažnija obeležja se dodatno vizualizuju pomoću bar-dijagrama.

Ova informacija je korisna za:

- interpretaciju modela,
- selekciju obeležja,
- optimizaciju budućih modela.

Zaključak evaluacije

Na osnovu dobijenih rezultata, modeli se mogu objektivno uporediti koristeći Macro-F1 kao primarnu metriku optimizacije i evaluacije.

Random Forest model, zahvaljujući ansambl strukturi i optimizaciji hiperparametara, pokazuje najbolje ukupne performanse, dok neuronska mreža predstavlja konkurentnu

alternativu sa sposobnošću modelovanja složenih nelinearnih odnosa. Multinomijalna logistička regresija služi kao referentni linearni model.

Analiza važnosti obeležja dodatno potvrđuje relevantnost pojedinih senzorskih i frekventnih karakteristika za detekciju kvarova.

Čuvavanje modela i rezultata

Nakon završene evaluacije performansi, neophodno je sačuvati trenirane modele, preprocessor i ključne rezultate kako bi se omogućila reproduktivnost, dalja upotreba u produkciji i ponovna analiza bez potrebe za ponovnim treniranjem. Ova faza osigurava da svi artefakti projekta budu arhivirani na strukturiran način, što je standardna praksa u mašinskom učenju i olakšava eventualno deploy-ovanje, poređenje sa budućim iteracijama ili deljenje sa saradnicima.

Kreiranje direktorijuma i čuvanje modela

Prvo se proverava postojanje direktorijuma `models` i `results`, i ukoliko ne postoje, oni se kreiraju. Ovo sprečava greške pri čuvanju i omogućava čistu organizaciju fajlova.

Svi trenirani modeli (Random Forest, multinomijalna logistička regresija i neuronska mreža) čuvaju se u RDS formatu, koji je nativni R format za serijalizaciju objekata i omogućava brzo učitavanje sa `readRDS()`. Takođe se čuva preprocessor (`preProc`) koji sadrži parametre centriranja i skaliranja, kako bi se isti transform mogao primeniti na nove podatke u budućnosti.

```
if (!dir.exists("models")) dir.create("models")
if (!dir.exists("results")) dir.create("results")

cat("\n--- Čuvanje modela ---\n")
saveRDS(rf_model,  "models/rf_model.rds")
saveRDS(log_model, "models/logistic_model.rds")
saveRDS(nn_model,  "models/nn_model.rds")
saveRDS(preProc,   "models/preprocessor.rds")
cat("Modeli sačuvani u 'models/' direktorijumu\n")
```

Struktura i čuvanje rezultata evaluacije

Svi ključni rezultati evaluacije se prikupljaju u jednu listu (`results_summary`) koja sadrži:

- metrike po modelu (tačnost, balansirana tačnost, makro-F1, AUC),
- matrice konfuzije (sirove i procentualne),
- detaljne metrike po klasama,
- top atributi po važnosti (za Random Forest),
- najbolji hiperparametri za neuronsku mrežu.

Ova lista se serijalizuje u jedan RDS fajl, što omogućava jednostavno učitavanje i dalju obradu u R-u (npr. za generisanje izveštaja, vizuelizacija ili poređenje sa novim modelima).

```
cat("\n--- Čuvanje rezultata ---\n")
results_summary <- list(
  RandomForest = list(
    Accuracy = rf_acc,
    BalancedAccuracy = rf_bal_acc,
    MacroF1 = rf_macro_f1,
    ConfusionMatrix = rf_cm$table,
    ConfusionMatrixPercent = rf_cm_prop,
    Metrics = rf_metrics,
    AUC = as.numeric(auc(roc_multi))
  ),
  LogisticRegression = list(
    Accuracy = log_acc,
    BalancedAccuracy = log_bal_acc,
    MacroF1 = log_macro_f1,
    ConfusionMatrix = log_cm$table,
    ConfusionMatrixPercent = log_cm_prop,
    Metrics = log_metrics,
    AUC = as.numeric(auc(roc_multi_log))
  ),
  NeuralNetwork = list(
    Accuracy = nn_cm$overall['Accuracy'],
    BalancedAccuracy = nn_bal_acc,
    MacroF1 = nn_macro_f1,
    ConfusionMatrix = nn_cm$table,
    ConfusionMatrixPercent = nn_cm_prop,
    Metrics = nn_metrics,
    AUC = as.numeric(auc(roc_multi_nn)),
    BestTune = nn_model$bestTune
  ),
  FeatureImportance = importance_df
)
saveRDS(results_summary, "results/evaluation_results.rds")
cat("Rezultati sačuvani u 'results/evaluation_results.rds'\n")
```

Kreiranje tekstualnog izveštaja

Na kraju se generiše čitljiv tekstualni izveštaj u fajlu `model_evaluation_report.txt`. Ovaj fajl sadrži sažet pregled performansi svih modela, ključne metrike, AUC vrednosti, najbolje hiperparametre neuronske mreže i top 10 najvažnijih obeležja. Koristi se funkcija `sink()` za preusmeravanje ispisa u fajl, što omogućava lako čitanje rezultata van R okruženja (npr. u tekst editoru ili za slanje izveštaja).

Izveštaj uključuje i osnovne informacije o eksperimentu (datum, broj obeležja, broj klasa), čime postaje samodovoljan dokument za arhivu ili prezentaciju.

```r
cat("\n--- Kreiranje tekstualnog izveštaja ---\n")
sink("results/model_evaluation_report.txt")
cat(paste0(rep("=", 80), collapse = ""), "\n")
cat("IZVEŠTAJ O EVALUACIJI MODELA\n")
cat(paste0(rep("=", 80), collapse = ""), "\n\n")
cat(sprintf("Datum: %s\n", Sys.Date()))
cat(sprintf("Broj feature-a: %d\n", length(feature_cols)))
cat(sprintf("Broj klasa: %d\n\n", nlevels(test_data$Fault_Type)))

cat("--- Random Forest ---\n")
cat(sprintf("Accuracy: %.4f\n", rf_acc))
cat(sprintf("Balanced Accuracy: %.4f\n", rf_bal_acc))
cat(sprintf("Macro-F1: %.4f\n", rf_macro_f1))
cat(sprintf("Multi-class AUC: %.4f\n\n", as.numeric(auc(roc_multi))))

cat("--- Multinomijalna logistička regresija ---\n")
cat(sprintf("Accuracy: %.4f\n", log_acc))
cat(sprintf("Balanced Accuracy: %.4f\n", log_bal_acc))
cat(sprintf("Macro-F1: %.4f\n", log_macro_f1))
cat(sprintf("Multi-class AUC: %.4f\n\n", as.numeric(auc(roc_multi_log))))

cat("--- Neural Network (MLP) ---\n")
cat(sprintf("Accuracy: %.4f\n", nn_cm$overall['Accuracy']))
cat(sprintf("Balanced Accuracy: %.4f\n", nn_bal_acc))
cat(sprintf("Macro-F1: %.4f\n", nn_macro_f1))
cat(sprintf("Multi-class AUC: %.4f\n\n", as.numeric(auc(roc_multi_nn))))
print(nn_model$bestTune)

cat("\n")
cat("--- Top 10 atributa (Random Forest) ---\n")
print(head(importance_df, 10))

sink()
cat("Tekstualni izveštaj sačuvan u 'results/model_evaluation_report.txt'\n")
```

Ova sekcija završava glavni tok eksperimenta. Svi modeli, preprocessor i rezultati su sada trajno sačuvani i spremni za dalju upotrebu, ponovno učitavanje ili integraciju u produkcioni sistem. U slučaju potrebe za deploy-ovanjem, modeli se mogu jednostavno učitati sa `readRDS()` i primeniti na nove podatke uz isti preprocessor.

Zaključak

Provedena analiza predstavlja celovit pristup razvoju i evaluaciji sistema za detekciju i dijagnostiku kvarova na osnovu podataka senzora. Kroz sistematske korake – od eksplorativne analize univarijatnih i frekventnih karakteristika, preko pažljivog preprocesiranja podataka (uključujući feature engineering, skaliranje, uklanjanje redundantnih obeležja i balansiranje klasa SMOTE metodom), do treniranja i rigorozne evaluacije tri različita modela – dobijeni su sledeći ključni nalazi.

Random Forest model je pokazao najviše performanse među testiranim algoritmima, sa ukupnom tačnošću od približno 60,3% na test skupu i makro-F1 skorom od 0,4197. Iako ova tačnost nije visoka u apsolutnom smislu, ona predstavlja značajno poboljšanje u odnosu na linearni baseline (multinomijalna logistička regresija: 22,61%) i jednostavnu neuronsku mrežu (31,66%). Posebno je važno da je Random Forest ostvario najviši makro-F1 skor, što ukazuje na relativno bolju sposobnost balansiranog tretmana svih klasa u uslovima izražene neravnoteže podataka.

Analiza važnosti obeležja otkrila je da najveći doprinos diskriminaciji daju frekventne komponente (FFT binovi) senzora Pressure i Vibration, posebno DC komponenta (bin 0) i nekoliko niskih frekventnih binova, kao i derivirana obeležja poput minimalne vrednosti senzora i odnosa struje i napona. Ovo potvrđuje da prelazak sa čisto vremenskih statistika na frekventni domen donosi dodatnu informaciju, iako viši FFT binovi nisu pokazali značajnu diskriminativnu moć u ovom skupu podataka.

Svi modeli su pokazali ograničenu sposobnost detekcije manjinskih klasa kvara (Overheating, Leakage, Power_Fluctuation), što se ogleda u niskim vrednostima osetljivosti i F1-skora za te klase, kao i u AUC vrednostima blizu 0,5. Ovo ukazuje da su kvarovi u datom skupu suptilni, sa značajnim preklapanjem distribucija u odnosu na normalno stanje, te da univarijatne i jednostavne linearne metode nisu dovoljne za pouzdanu separaciju.

Zaključno, Random Forest predstavlja najperspektivniji kandidat za osnovu budućeg sistema dijagnostike kvarova u ovom kontekstu. Međutim, trenutni nivo performansi pokazuje da postoji prostor za dalje unapređenje. Sledeći prioriteti uključuju:

- napredniji inženjering obeležja (npr. agregirane energetske mere iz FFT, wavelet transform, statistike višeg reda, vremenske zavisnosti),
- eksperimentisanje sa metodama za bolje rešavanje neravnoteže (ponderisani gubitak, oversampling sa fokusom na teške primere, cost-sensitive learning),
- kombinovanje modela kroz ensemble pristupe (npr. stacking ili voting),
- proširenje validacije na veće i raznovrsnije skupove podataka, uključujući simulirane ili stvarne kvarove različitih intenziteta,
- implementaciju pragova verovatnoće i objašnjivih pravila za interpretabilnost u kritičnim primenama.