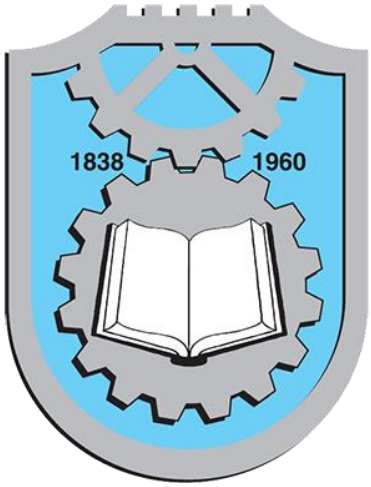
Univerzitet u Kragujevcu

Fakultet inženjerskih nauka



Seminarski rad

Siluete vozila

Student:

Ćirković Aleksandar 646-2020

Predmetni profesor:

Vesna Ranković

Tijana Geroski

Sadržaj

1. Postavka zadatka

Cilj zadatka je analiza skupa podataka koji opisuje siluete vozila i primena nekoliko različitih modela mašinskog učenja za klasifikaciju vozila na osnovu tih silueta.Ovaj zadatak ćemo posmatrati sa aspekta klasifikacije i koristićemo sledeća svojstva za analizu:

Compactness / Kompaktnost
Length_rectangular / Dužina pravougaonika
Circularity / Kružnost
Major_variance / Glavna varijansa
Distance_circularity / Razdaljina kružnosti
Minor_variance / Manja varijansa
Radius_ratio / Odnos poluprečnika
Gyration_radius / Radijus žetona
Praxis_aspect_ratio / Odnos glavnih osa
Major_skewness / Glavna asimetrija
Max_length_aspect_ratio / Maksimalni odnos dužine i širine
Minor_skewness / Manja asimetrija
Scatter_ratio / Razmera raspršenosti
Minor_kurtosis / Manja kurtosis
Elongatedness / Iščašenost
Major_kurtosis / Glavna kurtosis
Praxis_rectangular / Pravougaonost
Hollows_ratio / Odnos šupljina

Takodje treba napomenuti i ciljane vrednosti, a to su klase: **van, saab, bus, opel.**

Iz ovog skupa podataka izvešćemo osnovnu analizu i vizuelizaciju, te trenirati i evaluirati različite modele mašinskog učenja kako bismo predvideli tip vozila na osnovu siluete.

2. Opis zadatka

2.1. Biblioteke

U ovom radu koristili smo biblioteke NumPy, Pandas, Seaborn i Matplotlib za analizu i vizualizaciju podataka, te Scikit-learn za primenu algoritama mašinskog učenja. Fokusirali smo se na analizu silueta vozila i klasifikaciju vozila na osnovu tih silueta, koristeći modele poput Logističke regresije, SVM-a, KNN-a, MLP i Decision Tree, sa ciljem evaluacije njihovih performansi.

```
# Importovanje potrebnih biblioteka
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from ucimlrepo import fetch_ucirepo
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
```

Slika 1 – Importovanje biblioteka

2.2. Priprema seta podataka za obradu

Naredni deo koda služi za pripremu podataka za dalju analizu i mašinsko učenje.

Prvo, koristimo funkciju fetch_ucirepo iz modula ucimlrepo kako bismo učitali skup podataka o siluetama vozila.Zatim, podatke pretvaramo u pandas DataFrame, gde X sadrži karakteristike silueta vozila, a y informacije o klasama vozila.Prikazujemo metapodatke o skupu podataka i informacije o varijablama radi boljeg razumevanja podataka.

Nakon toga, koristimo LabelEncoder da bismo pretvorili klasne vrednosti vozila u numeričke vrednosti, što je važno za rad sa algoritmima mašinskog učenja koji zahtevaju numeričke vrednosti za ciljnu promenljivu.

Ova priprema podataka omogućava nam da nastavimo sa analizom i primenom algoritama mašinskog učenja na skupu podataka o siluetama vozila.

```
from ucimlrepo import fetch_ucirepo
# fetch dataset
statlog_vehicle_silhouettes = fetch_ucirepo(id=149)

# data (as pandas dataframes)
X = statlog_vehicle_silhouettes.data.features
y = statlog_vehicle_silhouettes.data.targets

# metadata
print(statlog_vehicle_silhouettes.metadata)

# variable information
print(statlog_vehicle_silhouettes.variables)

vehicle_df = statlog_vehicle_silhouettes.data.features
vehicle_df['class'] = statlog_vehicle_silhouettes.data.targets

# Inicijalizujemo LabelEncoder
label_encoder = LabelEncoder()

# Konvertujemo klasne vrednosti u numeričke vrednosti
vehicle_df['class'] = label_encoder.fit_transform(vehicle_df['class'])
```

Slika 2 – Priprema seta podataka za obradu

3. Analiza i vizuelizacija podataka

3.1. Matrica korelacije

Ovaj korak u analizi podataka pomaže nam da razumemo međusobne veze između različitih karakteristika (atributa) silueta vozila. Heatmapa koju generišemo prikazuje korelaciju između svakog para atributa.

Kada je ćelija u heatmapi tamna, to znači da dva atributa imaju slabo ili nikakvo međusobno povezivanje. Sa druge strane, tamnija boja ukazuje na jače povezivanje. Ako je korelacija blizu 1 ili -1, to znači da su atributi visoko povezani, dok vrednosti bliske nuli ukazuju na slabu ili nikakvu povezanost.Analiza korelacije može nam pomoći da identifikujemo redundantne ili nepotrebne atribute, kao i da razumemo kako se atributi međusobno odnose što može biti korisno prilikom izbora atributa za modeliranje. Takođe, može nam pomoći da otkrijemo

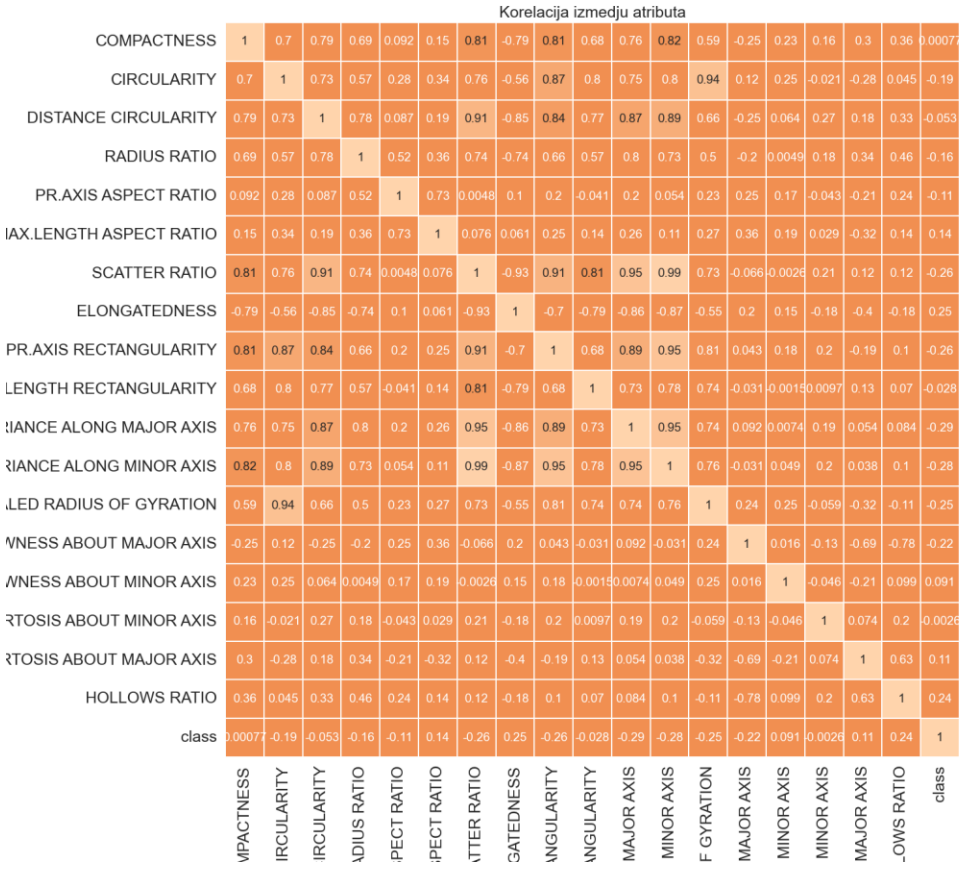
eventualne probleme kao što je multikolinearnost, što može uticati na performanse modela mašinskog učenja.

```
# Prikaz korelacije između feature-a
cor = vehicle_df.corr()

# korelaciona matrica
sns.set(font_scale=1.15)
fig, ax = plt.subplots(figsize=(18, 15))
sns.heatmap(cor, vmin=0.8, annot=True, linewidths=0.01, center=0, linecolor="white", cbar=False, square=True)
plt.title('Correlation between attributes', fontsize=18)
ax.tick_params(labelsize=18)
plt.savefig('matricakorelacije.png', format='png')
plt.show()
```

Slika 3 – Kod za iscrtavanje matrice korelacije

Tom prilikom se dobija sledeći grafik:



Plot 1 – Grafik matrice korelacije

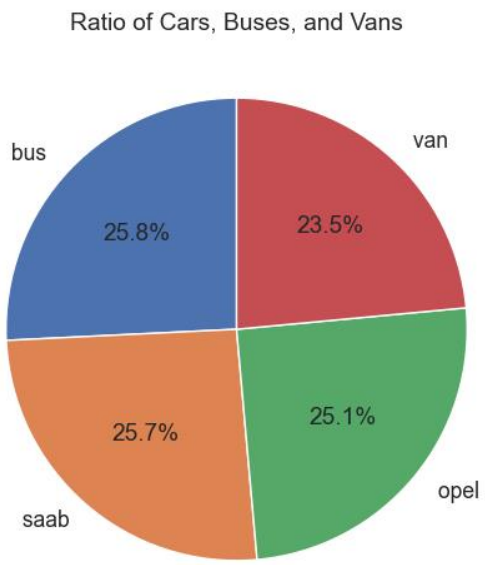
3.2. Učestalost podataka klasa

Ovaj deo koda učitava podatke iz CSV fajla 'vehicle.csv' i zatim koristi value_counts() funkciju da broji broj instanci za svaku klasu vozila ('class' atribut). Zatim se koristi plt.pie funkcija da se kreira dijagram u obliku pite koji prikazuje udeo svake klase vozila u celokupnom skupu podataka.

```
data = pd.read_csv('vehicle.csv')
vehicle_counts = data['class'].value_counts()
plt.figure(figsize=(10, 6))
plt.pie(vehicle_counts, labels=vehicle_counts.index, startangle=90, autopct='%1.1f%%')
plt.title('Ucestalost klasa')
plt.savefig('pieplot.png', format='png')
plt.show()
```

Slika 4 – Crtanje pieplot-a radi analize učestalosti podataka

Sa grafika možemo zaključiti da je najveća verovatnoća da se pojavi klasa (bus).



Plot 2 – Grafik učestanosti pojavljivanja klase

4. Priprema podataka za trening i evaulaciju

U ovom koraku ćemo trenirati i evaluirati različite modele mašinskog učenja kako bismo predvideli kojoj klasi vozila pripada određena silueta. Prvo ćemo podeliti podatke na trening i test skupove, zatim ćemo trenirati modele na trening skupu i evaluirati njihove performanse na test skupu. Korišćeni algoritmi će biti Logistička regresija, K najbližih suseda (KNN), Metoda potpornih vektora (SVM), Višeslojni perceptron (MLP) i Stablo odlučivanja (Decision Tree). Na kraju ćemo uporediti tačnosti ovih modela kako bismo videli koji se najbolje pokazao našem zadatku klasifikacije silueta vozila.

```
# Podela podataka na trening i test skupove
#X = vehicle_df.drop(columns='class')
y = vehicle_df['class']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Popunjavanje NaN vrednosti u X_train i X_test
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)
```

Slika 5 – Priprema skupa podataka za trening

5. Primena različitih modela mašinskog učenja za klasifikaciju silueta vozila.

U ovom delu seminarkog rada istražujemo primenu različitih modela mašinskog učenja za klasifikaciju silueta vozila. Za svaki od korišćenih modela, analiziraćemo tačnost (Accuracy) na trening i test skupovima. Accuracy je mera koja nam govori koliko često je model tačno predvideo klasu vozila na osnovu siluete.

Važno je napomenuti da ćemo sve navedene rezultate zaokružiti i prikazati u procentima radi lakšeg razumevanja i poređenja performansi različitih modela. Ovo će nam pomoći da identifikujemo najefikasniji model za naš specifičan problem klasifikacije silueta vozila.

Definišemo različite modele mašinskog učenja za klasifikaciju silueta vozila, kao i postupak treniranja i evaluacije tih modela. Koriste se modeli Logističke regresije, K najbližih suseda (KNN), Metoda potpornih vektora (SVM) i Višeslojni perceptron (MLP).

Za svaki model se trenira na trening skupu (`X_train`, `y_train`) i evaluira na test skupu (`X_test`, `y_test`). Nakon toga, generiše se izveštaj klasifikacije koji prikazuje preciznost, odziv, F1-skor i druge metrike za svaku klasu. Na kraju, rezultati tačnosti (Accuracy) se čuvaju radi poređenja performansi modela.

```python
# Definisanje modela
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'KNN': KNeighborsClassifier(),
    'SVM': SVC(),
    'MLP': MLPClassifier(max_iter=1000)
}

# Treniranje i evaluacija modela
results = {}
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"Classification Report for {name}:\n")
    print(metrics.classification_report(y_test, y_pred))
    print("\n" + "="*60 + "\n")
    results[name] = metrics.accuracy_score(y_test, y_pred)

# Prikaz rezultata tačnosti

preciznosti=[]
```

Slika 6 – Definisanje treniranje I evaulacija modela

5.1. Primena MLP klasifikatora

```python
# Koriscenje MLP Classifier-a za treniranje
mlp = MLPClassifier(hidden_layer_sizes=(500, 100), max_iter=300)
mlp.fit(X_train, y_train)
y_mlp_pred = mlp.predict(X_test)

print("Ocena za X-train sa Y-train je : ", mlp.score(X_train, y_train))
print("Ocena za X-test sa Y-test je : ", mlp.score(X_test, y_test))
print("Evaluacija MLP-a : Accuracy score ", accuracy_score(y_test, y_mlp_pred))
preciznosti.append(accuracy_score(y_test, y_mlp_pred))
```

Slika 7 – Primena MLP klasifikatora

Višeslojni perceptron (MLP) klasifikator za treniranje na trening skupu (X_train, y_train) i evaluaciju na test skupu (X_test, y_test). MLP klasifikator je definisan sa skrivenim slojevima veličine (500, 100) i maksimalnim brojem iteracija 300.

Nakon treniranja, prikazuju se ocene modela za trening i test skupove, kao i tačnost (Accuracy) evaluacije modela na test skupu. Ovi rezultati se koriste za procenu performansi MLP klasifikatora u zadatku klasifikacije silueta vozila. Tačnost modela se dodaje u listu preciznosti radi poređenja sa drugim modelima.Rezultate koje dobijamo su:

```
Ocena za X-train sa Y-train je :  0.6638513513513513
Ocena za X-test sa Y-test je :  0.6417322834645669
Evaluacija MLP-a : Accuracy score  0.6417322834645669
```

Rezultat 1 – MLP

Nakon treniranja i evaluacije Višeslojnog perceptrona (MLP) klasifikatora, utvrdili smo da je tačnost ovog modela na test skupu 64.1%. To znači da model uspešno predviđa klasu vozila na osnovu siluete sa tačnošću od 64.1%. Ova informacija će nam pomoći da uporedimo performanse MLP modela sa drugim modelima koje ćemo koristiti u našem istraživanju.

5.2. Primena logističke regresije

Logistička regresija je algoritam za klasifikaciju koji predviđa verovatnost da podatak pripada određenoj klasiOvaj deo koda koristi Logističku regresiju za treniranje i evaluaciju modela na skupu podataka o siluetama vozila.

Logistička regresija se trenira na trening skupu (`X_train`, `y_train`), a zatim se evaluira na test skupu (`X_test`, `y_test`). Prikazuju se ocene modela za trening i test skupove, kao i tačnost predikcija na test skupu. Na kraju, tačnost (Accuracy) se dodaje u listu `preciznosti` radi poređenja sa drugim modelima.

```
# Koriscenje Logicke Regresije
Lo_model = LogisticRegression(solver='liblinear')
Lo_model.fit(X_train, y_train)

print("Ocena za X-train sa Y-train je : ", Lo_model.score(X_train, y_train))
print("Ocena za X-test sa Y-test je : ", Lo_model.score(X_test, y_test))

y_pred_Lo = Lo_model.predict(X_test)
print("Evaluacija Logicke Regresije : Accuracy score ", accuracy_score(y_test, y_pred_Lo))
preciznosti.append(accuracy_score(y_test, y_pred_Lo))
```

Slika 8 – Logisticka Regresija

Kao rezultat dobijamo:

```
Ocena za X-train sa Y-train je :  0.9712837837837838
Ocena za X-test sa Y-test je :  0.9488188976377953
Evaluacija Logicke Regresije : Accuracy score  0.9488188976377953
```

Rezultat 2 – Logisticka Regresija

Nakon treniranja i evaluacije modela Logističke regresije, postigli smo tačnost od 94.8% na test skupu. To znači da je model uspešno predvideo klasu vozila na osnovu siluete sa tačnošću od 94.8%. Ova visoka tačnost pokazuje da je Logistička regresija veoma efikasna za ovaj zadatak klasifikacije silueta vozila, i predstavlja značajan napredak.

5.3. Primena Decision Tree Classifier-a

U ovom delu koda koristimo Decision Tree Classifier za treniranje i evaluaciju modela na skupu podataka o siluetama vozila. Takođe vršimo pretragu najboljih hiperparametara kako bismo pronašli optimalne vrednosti za poboljšanje performansi modela.

```
Tree_model = DecisionTreeClassifier(random_state=42)
param_grid = {
    'max_depth': [20, 25, 30, 35, 40],
    'min_samples_split': [2, 3, 4, 5],
    'min_samples_leaf': [1, 2, 3, 4],
    'max_features': [None, 'sqrt', 'log2'],
    'class_weight': [None, 'balanced']
}
# Pretraga najboljih parametara
grid_search = GridSearchCV(Tree_model, param_grid, cv=5, n_jobs=-1, verbose=1, scoring='accuracy')
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)
# Overrideujemo Tree model novim objektom sa optimalnim hiperparametrima
Tree_model = DecisionTreeClassifier(**grid_search.best_params_)
Tree_model.fit(X_train, y_train)
y_pred_tree = Tree_model.predict(X_test)
print("Ocena za X-train sa Y-train je : ", Tree_model.score(X_train, y_train))
print("Ocena za X-test sa Y-test je : ", Tree_model.score(X_test, y_test))
print("Evaluacija Decision Tree : Accuracy score ", accuracy_score(y_test, y_pred_tree))
preciznosti.append(accuracy_score(y_test, y_pred_tree))
# Koriscenje SVC za treniranje
svc_model = SVC(C=50, kernel="rbf")
svc_model.fit(X_train, y_train)
y_pred_svc = svc_model.predict(X_test)
print("Ocena za X-train sa Y-train je : ", svc_model.score(X_train, y_train))
print("Ocena za X-test sa Y-test je : ", svc_model.score(X_test, y_test))
print("Evaluacija SVC-a : Accuracy score ", accuracy_score(y_test, y_pred_svc))
preciznosti.append(accuracy_score(y_test, y_pred_svc))
```

Slika *9 – Decision Tree Classifier*

Rezultat ovog klasifikatora:

```
Ocena za X-train sa Y-train je :  1.0
Ocena za X-test sa Y-test je :  1.0
Evaluacija Decision Tree : Accuracy score  1.0
```

Rezultat 3 – Decision Tree Classifier

Iz aplikovanog koda zaključujemo da je GridSearchCV optimizovao hiperparametre koristeći 5-fold unakrsnu validaciju i 480 različitih kombinacija hiperparametara. Nakon prosleđivanja tih hiperparametara modelu, dobili smo ocene od 100% za sve evaluacione metrike, što pokazuje da model izuzetno dobro klasifikuje podatke u trening i test skupovima.

5.4. Primena SVC-a

Sledeći model koji koristimo za treniranje i evaluaciju je Support Vector Classifier (SVC) sa radijalnom baznom funkcijom (RBF) kao jezgrenom funkcijom. SVC je moćan alat za klasifikaciju koji pokušava pronaći optimalnu hiperravnu koja najbolje razdvaja klase podataka.

```python
# Koriscenje SVC za treniranje
svc_model = SVC(C=50, kernel="rbf")

svc_model.fit(X_train, y_train)

y_pred_svc = svc_model.predict(X_test)

print("Ocena za X-train sa Y-train je : ", svc_model.score(X_train, y_train))
print("Ocena za X-test sa Y-test je : ", svc_model.score(X_test, y_test))
print("Evaluacija SVC-a : Accuracy score ", accuracy_score(y_test, y_pred_svc))
preciznosti.append(accuracy_score(y_test, y_pred_svc))
```

Slika 10 – SVC

Rezultat :

```
Ocena za X-train sa Y-train je :  0.7584459459459459
Ocena za X-test sa Y-test je :  0.7125984251968503
Evaluacija SVC-a : Accuracy score  0.7125984251968503
```

Rezultat 4 - SVC

Nakon treniranja SVC modela, procenili smo njegove performanse na trening i test skupovima. Model je postigao tačnost od 71.2% na test skupu, što pokazuje njegovu sposobnost da klasifikuje siluete vozila sa solidnom preciznošću.

5.5. Primena KNeighbours klasifikatora

Poslednji model koji smo koristili za treniranje i evaluaciju je KNeighbors Classifier (KNN), koji je jedan od najjednostavnijih algoritama za klasifikaciju. Ovaj algoritam funkcioniše na principu pronalaženja 'k' najbližih suseda za svaki podatak koji treba klasifikovati.

```
#Koriscenje KNeighbors Classifier-a za treniranje
K_model = KNeighborsClassifier(n_neighbors=5)
K_model.fit(X_train, y_train)

y_pred_k = K_model.predict(X_test)

print("Ocena za X-train sa Y-train je : ", K_model.score(X_train, y_train))
print("Ocena za X-test sa Y-test je : ", K_model.score(X_test, y_test))
print("Evaluacija K Neighbors-a : Accuracy score ", accuracy_score(y_test, y_pred_k))
preciznosti.append(accuracy_score(y_test, y_pred_k))
results_df = pd.DataFrame({'Model': ['MLP', 'Logistic Regression', 'Decision Tree', 'SVC', 'K Neighbors'],
                           'Accuracy': preciznosti})
```

Slika 11 – Kneighbours

Rezultat:

```
Ocena za X-train sa Y-train je :  0.7905405405405406
Ocena za X-test sa Y-test je :  0.6338582677165354
Evaluacija K Neighbors-a : Accuracy score  0.6338582677165354
```

Rezultat 5 – Kneighbours.

Za KNeighbors Classifier (KNN) model dobijena je tačnost od 63.3% na test skupu, što znači da je ovaj model bio manje precizan u klasifikaciji silueta vozila u poređenju sa prethodno opisanim modelima.

6. Odabir najboljeg modela da rešavanje zadatka

Na osnovu rezultata evaluacije različitih modela mašinskog učenja za klasifikaciju silueta vozila, možemo zaključiti da je model Decision Tree postigao najbolju tačnost od 100% na test skupu podataka. Ovaj rezultat ukazuje na to da je Decision Tree najbolji model za predviđanje klase vozila na osnovu siluete.

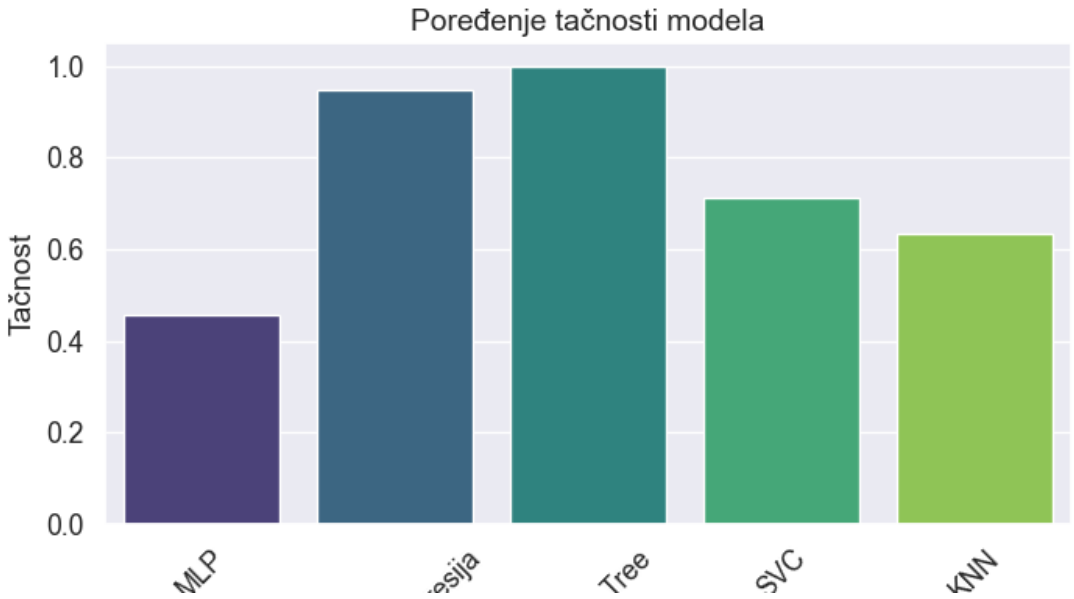
U poređenju sa Decision Tree, ostali modeli su postigli sledeće tačnosti:

- Logistička regresija: 94.8%
- MLP (Multilayer Perceptron): 64.1%
- SVC (Support Vector Classifier): 71.2%
- K Neighbors: 63.3%

Ovaj deo koda koristi biblioteke pandas i seaborn za prikazivanje poređenja tačnosti različitih modela mašinskog učenja u obliku trakastog dijagrama.

```
#Poredjenje tačnosti modela
results_df = pd.DataFrame({'Model': ['MLP', 'Logistička regresija', 'Decision Tree', 'SVM', 'KNN'],
'Tačnost': preciznosti})
plt.figure(figsize=(10, 6))
sns.barplot(x=['MLP', 'Logistička regresija', 'Decision Tree', 'SVC', 'KNN'], y=preciznosti, palette='viridis')
plt.title('Poređenje tačnosti modela')
plt.ylabel('Tačnost')
plt.xlabel('Model')
plt.xticks(rotation=45)
plt.savefig('poređenje_tačnosti_modela.png', format='png')
plt.show()
```

Slika 11 – Kod za prikaz poredjenja tačnosti



Plot 3 – Grafik poredjenja tačnosti modela

Decision Tree je moćan algoritam za klasifikaciju i regresiju, posebno zbog svoje lakoće interpretacije i sposobnosti da se nosi sa ne-linearnim podacima. Međutim, potrebno je voditi računa o prenaučenju i nestabilnosti modela, što ga čini manje prikladnim za neke situacije. Kombinacija više stabala odlučivanja kao što je Random Forest može poboljšati stabilnost i tačnost modela.

7. Literatura

1. https://sci2s.ugr.es/keel/dataset.php?cod=68
2. https://pandas.pydata.org/docs/
3. https://seaborn.pydata.org/
4. https://docs.python.org/3/
5. http://moodle.fin.kg.ac.rs/course/view.php?id=989
6. https://archive.ics.uci.edu/dataset/149/statlog+vehicle+silhouettes