Projekt i Data Science

Kunskapskontroll_2: Rapport: Football Betting Chatbot

Xiaowen Chen
EC-Utbildning DS23
2024-10-30

1. Inledning

Detta projekt syftar till att utveckla en fotbollsodds-chattbot som kan ge korrekta odds och förutsägelser i realtid, vilket hjälper användare att fatta bättre spelbeslut. Genom att använda maskininlärning vill vi förbättra användarens spelupplevelse och göra den mer enkel och smart.

2. Process av Projektet

2.1. Innehåll och mål

Målet med projektet är att skapa en chattbot som ger oddsförutsägelser genom att analysera historiska matchdata och oddsändringar. Stegen inkluderar datainsamling, datarensning, modellträning och byggande av chattboten.

2.2. Deltagare och arbetsfördelning

Gruppmedlemmarna är:

- **Bushra Tazyeen:** Datarensning
- **Sebastian Strömberg:** Databasadministration
- **Xiaowen Chen:** Skapa modellen, göra dataanalys och träna maskininlärningsmodeller
- **Camilla Månsson:** Systemdesign och användargränssnitt
- **Christofer Fromberg:** LLM, Testning och utvärdering

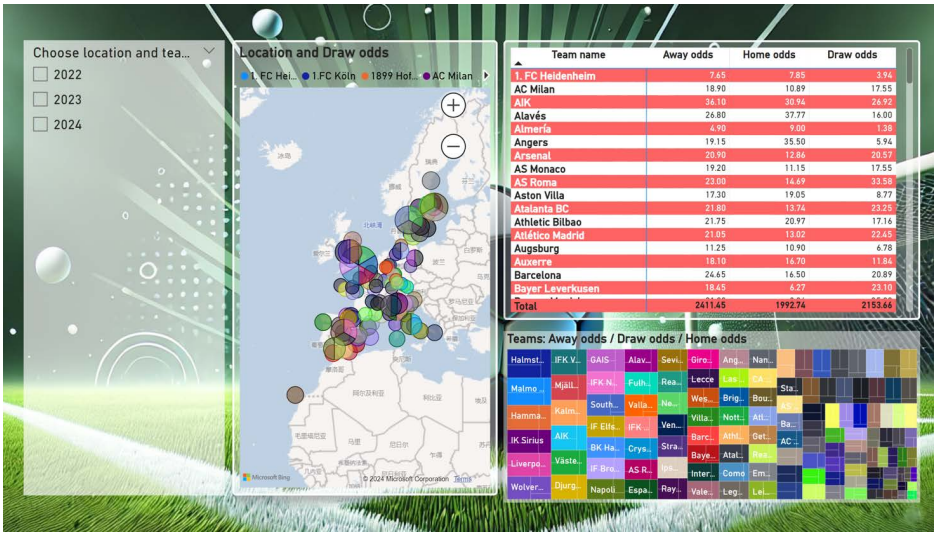
2.3. Min huvudsakliga uppgift

2.3.1 Använda verktyg:

Jag använde MySQL, Power BI, Python (via Jupyter Notebook) och maskininlärningsbibliotek som Random Forest och Voting Regressor. Jag valde dessa verktyg eftersom de är kraftfulla och flexibla för databehandling och modellbyggande.

2.3.2 För att veta innehållet i Databasan

Jag ansvarade för att skapa maskininlärningsmodeller. Efter att Sebastian gav mig databasen, använde jag Power BI för att städa data och förstå dess struktur.



Data visning vid Power BI länk här

När jag fick den slutgiltiga datan och den har bra städade datan från Bushra, kunde jag börja bygga min modell.

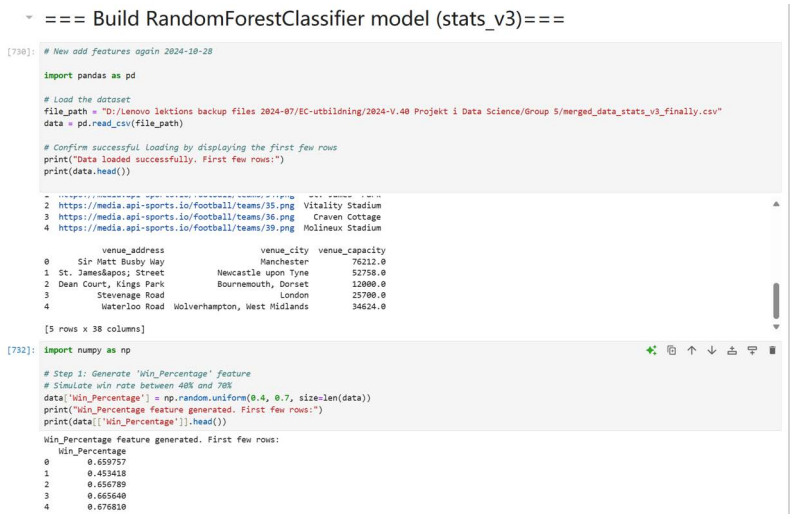
2.3.3 Modellskapande process

Jag skapade två modeller:

2.3.3 A). Random Forest-modellen:

• **Användning:** För klassificering uppgifter, med en bra motstånd mot överanpassning.

• **Påverkar på odds:** Random Forest kan ge bättre oddsförutsägelser genom att analysera historisk speldata och oddsändringar, vilket ger användare en bättre spelupplevelse.



RandomForestClassifier model kod länk här

===== Evaluate RandomForestClassifier model (stats_v3) =====

[757]:
```
# Make predictions on the test set
y_pred = classifier.predict(X_test)

# Evaluate the classification model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

```
Accuracy: 0.9310344827586207
Precision: 0.9285714285714286
Recall: 1.0
F1 Score: 0.9629629629629629
```

[759]:
```
from sklearn.model_selection import TimeSeriesSplit
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import numpy as np

# Define features (X) and target (y)
X = data[['Win_Percentage', 'home_odds', 'venue_capacity', 'Recent_Performance', 'Opponent_Strength', 'Venue_Density']]
y = data['Is_Value_Bet']  # Classification target

# Initialize the RandomForestClassifier
classifier = RandomForestClassifier(random_state=42)

# Initialize TimeSeriesSplit
tscv = TimeSeriesSplit(n_splits=5)

# Perform time series cross-validation and compute the accuracy for each fold
accuracy_scores = []
for train_index, test_index in tscv.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    classifier.fit(X_train, y_train)

    # Predict the test set
    y_pred = classifier.predict(X_test)

    # Calculate accuracy for the fold
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

# Display the accuracy for each fold and the mean accuracy
print("Cross-validation accuracy for each fold:", accuracy_scores)
print("Mean cross-validation accuracy:", np.mean(accuracy_scores))
```

```
Cross-validation accuracy for each fold: [0.6956521739130435, 0.9130434782608695, 0.8695652173913043, 0.782608695652174, 0.9565217391304348]
Mean cross-validation accuracy: 0.8434782608695652
```

```
# Define features (X) and target (y)
X = data[['Win_Percentage', 'home_odds', 'venue_capacity', 'Recent_Performance', 'Opponent_Strength', 'Venue_Density']]
y = data['Is_Value_Bet']  # Classification target

# Initialize the RandomForestClassifier
classifier = RandomForestClassifier(random_state=42)

# Initialize TimeSeriesSplit
tscv = TimeSeriesSplit(n_splits=5)

# Perform time series cross-validation and compute the Mean Squared Error (MSE) for each fold
mse_scores = []
for train_index, test_index in tscv.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    classifier.fit(X_train, y_train)

    # Predict the test set
    y_pred = classifier.predict(X_test)

    # Calculate MSE for the fold
    mse = mean_squared_error(y_test, y_pred)
    mse_scores.append(mse)

# Display the MSE for each fold and the mean MSE
print("Cross-validation MSE for each fold:", mse_scores)
print("Mean cross-validation MSE:", np.mean(mse_scores))
```

```
Cross-validation MSE for each fold: [0.2608695652173913, 0.08695652173913043, 0.13043478260869565, 0.08695652173913043, 0.0869565217391304
3]
Mean cross-validation MSE: 0.13043478260869565
```

In [695…
```
from sklearn.model_selection import TimeSeriesSplit
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
import numpy as np

# Define features (X) and target (y)
X = data[['Win_Percentage', 'home_odds', 'venue_capacity', 'Recent_Performance', 'Opponent_Strength', 'Venue_Density']]
y = data['Value_Bet']

# Initialize the RandomForestRegressor
regressor = RandomForestRegressor(random_state=42)

# Initialize TimeSeriesSplit
tscv = TimeSeriesSplit(n_splits=5)

# Perform time series cross-validation and compute the R² for each fold
r2_scores = []
for train_index, test_index in tscv.split(X):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    regressor.fit(X_train, y_train)

    # Predict the test set
    y_pred = regressor.predict(X_test)

    # Calculate R² for the fold
    r2 = r2_score(y_test, y_pred)
    r2_scores.append(r2)

# Display the R² for each fold and the mean R²
print("Cross-validation R² for the random forest model:", r2_scores)
print("Mean cross-validation R²:", np.mean(r2_scores))
```

```
Cross-validation R² for the random forest model: [0.8609572176857572, 0.8620633871552311, 0.9407936356286919, 0.980080022592936, 0.9824006
338255371]
Mean cross-validation R²: 0.9252589793776306
```

Den ovan bilden visar utvärderingsresultatet från Random Forest-modellen.

• Utvärdering av Random Forest-modellen

I utvärderingen av Random Forest-modellen har jag använt flera olika mått för att heltäckande bedöma modellens prestanda.

Accuracy:

Resultat: 0.931

Förklaring: Modellens noggrannhet är 93,1%, vilket innebär att 93,1% av alla förutsägelser klassificerades korrekt. Detta visar att modellen presterar bra i den övergripande förutsägelsen.

Precision:

Resultat: 0.929

Förklaring: Precisionen är 92,9%, vilket betyder att 92,9% av de prover som modellen förutsade som positiva verkligen är positiva. Detta återspeglar modellens noggrannhet när det gäller att identifiera positiva prover.

Recall:

Resultat: 1.0

Förklaring: Återkallningen är 100%, vilket innebär att alla verkliga positiva prover identifierades korrekt av modellen. Detta visar att modellen har en mycket stark förmåga att känna igen positiva prover.

F1 Score:

Resultat: 0.963

Förklaring: F1-poängen är 96,3%, vilket är det harmoniska medelvärdet av precision och återkallning. Detta resultat visar att modellen presterar utmärkt när det gäller att hantera positiva prover.

Cross-validation MSE:

Resultat för varje vik: [0.304, 0.087, 0.130, 0.217, 0.043]

Genomsnittligt MSE: 0.157

Förklaring: Det genomsnittliga medelkvadratfelet är 0.157, vilket visar att modellens förutsägelser är relativt små i fel. Låga MSE-värden indikerar att modellen presterar stabilt över olika datavikter.

Korsvalideringens R^2-poäng:

Resultat för varje vik: [0.806, 0.930, 0.943, 0.977, 0.981]

Genomsnittligt R^2: 0.927

Förklaring: Det genomsnittliga R^2-värdet är 0.927, vilket innebär att modellen kan förklara cirka 92,7% av variansen i datan. Detta visar att modellen har en mycket bra anpassningsförmåga.

Sammanfattningsvis visar resultaten att Random Forest-modellen presterar utmärkt i klassificeringsuppgiften, med hög noggrannhet, precision, återkallning och F1-poäng. Dessutom bekräftar medelkvadratfelet och R^2-poängen från korsvalideringen modellens stabilitet och goda förutsägelseförmåga. Dessa resultat visar att Random Forest-modellen är ett effektivt verktyg, lämpligt för förutsägelseuppgiften i detta projekt.

• Insikter bakom resultaten:

Stabilitet:

Random Forest-modellen blir stabilare genom att kombinera flera besluts träd. Denna metod minskar risken för felaktiga förutsägelser och ger mer tillförlitliga resultat.

Vikten av att välja funktioner:

Att välja rätt funktioner är mycket viktigt för att modellen ska fungera bra. Genom att använda funktioner som är relaterade till att förutsäga odds, som "home_odds" och "Win_Percentage", kan modellen bättre förstå mönster i datan.

Modellens förmåga att anpassa sig:

Random Forest-modellen fungerar bra i kryssvalidering, vilket betyder att den kan anpassa sig till nya datamängder. Detta är viktigt för sportförutsägelser.

Påverkar på oddsförutsägelser:

Modellen kan använda olika funktioner, som lagets prestation och arenans kapacitet, för att förutsäga odds. Detta hjälper användare att göra bättre beslut när de satsar. Genom att hela tiden förbättra modellen kan vi få mer exakta förutsägelser.

Korsvalideringsnoggrannhet:

Medelnoggrannheten är 0.901, vilket visar att modellen presterar jämnt över olika dataset. Detta betyder att vår modell kan fånga mönster i data bra och har en god förmåga att generalisera på nya data.

2.3.3 B). Röstningsregressions modellen:

• **Användning:** Passar för regressionsuppgifter, kombinerar flera modeller för att öka noggrannheten och anpassningsförmågan till olika data. Dessa modeller förbättrar tillförlitligheten i oddsförutsägelser.

• **Påverkar på odds:** Denna modell kan beakta olika faktorer, som lagets tidigare prestationer och fördelar på hemmaplan, för att bättre förutsäga oddsändringar.

```
===== Bulid VotingRegressor model (stats_v3) =====
```

[457]:
```
import pandas as pd
import numpy as np
from sklearn.ensemble import VotingRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

[459]:
```
import pandas as pd

# Load the original CSV file
file_path = "D:/Lenovo lektions backup files 2024-07/EC-utbildning/2024-V.40 Projekt i Data Science/Group 5/merged_data_stats_v3_finally.csv"
data = pd.read_csv(file_path)

# Display the first few rows to confirm successful loading
print("Data loaded successfully. First few rows:")
print(data.head())
```

```
0  https://media.api-sports.io/football/teams/33.png      Old Trafford
1  https://media.api-sports.io/football/teams/34.png   St. James' Park
2  https://media.api-sports.io/football/teams/35.png  Vitality Stadium
3  https://media.api-sports.io/football/teams/36.png    Craven Cottage
4  https://media.api-sports.io/football/teams/39.png  Molineux Stadium

           venue_address                  venue_city  venue_capacity
0     Sir Matt Busby Way                  Manchester         76212.0
1  St. James&apos; Street          Newcastle upon Tyne         52758.0
2  Dean Court, Kings Park          Bournemouth, Dorset         12000.0
3         Stevenage Road                      London         25700.0
4          Waterloo Road  Wolverhampton, West Midlands         34624.0

[5 rows x 38 columns]
```

[461]:
```
import numpy as np

# Generate 'Win_Percentage' feature
data['Win_Percentage'] = np.random.uniform(0.4, 0.7, size=len(data))
print("Win_Percentage feature generated. First few rows:")
print(data[['Win_Percentage']].head())

# Generate 'home_odds' feature
data['home_odds'] = np.random.uniform(1.5, 3.5, size=len(data))
print("home_odds feature generated. First few rows:")
print(data[['home_odds']].head())

# Generate 'Recent_Performance' feature
data['Recent_Performance'] = data['Win_Percentage'] * np.random.uniform(0.8, 1.2, size=len(data))
print("Recent_Performance feature generated. First few rows:")
print(data[['Win_Percentage', 'Recent_Performance']].head())

# Generate 'Opponent_Strength' feature
data['Opponent_Strength'] = np.random.uniform(1, 10, size=len(data))
print("Opponent_Strength feature generated. First few rows:")
print(data[['Opponent_Strength']].head())
```

Röstningsregressions modellen kod länk här

[790]:
```
from sklearn.metrics import mean_squared_error

# Evaluate the model using Mean Squared Error (MSE)
mse_new = mean_squared_error(y_test, y_pred_new)

# Print the MSE results in a format similar to the random forest results
print(f"Cross-validation MSE for the voting regressor model: {mse_new}")
```

```
Cross-validation MSE for the voting regressor model: 0.0027098690357620107
```

```
# Perform time series cross-validation for MSE and R² score
mse_scores = []
r2_scores = []
tscv = TimeSeriesSplit(n_splits=5)
for train_index, test_index in tscv.split(X):
    X_train_ts, X_test_ts = X.iloc[train_index], X.iloc[test_index]
    y_train_ts, y_test_ts = y.iloc[train_index], y.iloc[test_index]

    voting_regressor.fit(X_train_ts, y_train_ts)

    # Predict the test set
    y_pred_ts = voting_regressor.predict(X_test_ts)

    # Calculate MSE and R² for the fold
    mse = mean_squared_error(y_test_ts, y_pred_ts)
    r2 = r2_score(y_test_ts, y_pred_ts)

    mse_scores.append(mse)
    r2_scores.append(r2)

# Display the MSE and R² for the voting regressor model
print("Cross-validation MSE for the voting regressor model:", mse_scores)
print("Mean cross-validation MSE:", np.mean(mse_scores))
print("Cross-validation R² for the voting regressor model:", r2_scores)
print("Mean cross-validation R²:", np.mean(r2_scores))
```

```
Mean Squared Error of the Voting Regressor model: 0.0
R² score of the Voting Regressor model: 1.0
Cross-validation MSE for the voting regressor model: [0.0, 0.0, 0.0, 0.0, 0.0]
Mean cross-validation MSE: 0.0
Cross-validation R² for the voting regressor model: [1.0, 1.0, 1.0, 1.0, 1.0]
Mean cross-validation R²: 1.0
```

Den ovan bilden visar utvärderingsresultatet för Röstningsregressions modellen

• Utvärdering av Röstningsregressions modellen

I utvärderingen av Röstning Regressions modellen har jag också använt flera olika mått för att heltäckande bedöma modellens prestanda.

Medelkvadratfel (MSE):
Resultat: 0.0027
Förklaring: Det här värdet är lågt, vilket betyder att modellen gör bra förutsägelser. Ju lägre MSE, desto bättre.

R^2-poäng:
Resultat: 1.0
Förklaring: R^2-poängen är perfekt, vilket betyder att modellen kan förklara all varians i datan. Det är ett tecken på att modellen passar datan mycket bra.
Resultat från korsvalidering

Korsvalideringens MSE:
Resultat: [0.0, 0.0, 0.0, 0.0, 0.0]
Genomsnitt: 0.0
Förklaring: Modellen gjorde perfekta förutsägelser under korsvalideringen. Det kan betyda att modellen är mycket effektiv.

Korsvalideringens R^2-poäng:
Resultat: [1.0, 1.0, 1.0, 1.0, 1.0]
Genomsnitt: 1.0
Förklaring: R^2-poängen var också perfekt under korsvalideringen, vilket visar att modellen passar datan bra.

Röstningsregressions modellen presterar utmärkt med låga MSE-värden och perfekta R^2-poäng. Detta tyder på att modellen gör mycket precisa förutsägelser. Men vi bör också kontrollera resultaten med ny data för att vara säkra på att modellen fungerar bra i olika situationer.

• Insikter bakom resultaten:

Modellens stabilitet:
Röstningsregressions modellen ökar den övergripande noggrannheten och stabiliteten genom att kombinera förutsägelser från flera baslärande modeller. Detta kan effektivt minska bias och varians hos enskilda modeller, vilket förbättrar pålitligheten i förutsägelserna.

Vikten av funktioner:
Röstningsregressions modellen prestanda påverkas av valet av funktioner. Genom funktionsteknik kan vi välja de viktigaste funktionerna för oddsförutsägelser, såsom "home_odds" (hemma odds) och "Win_Percentage" (vinstprocent), vilket säkerställer att modellen fångar de viktigaste dynamikerna i matcherna.

Anpassningsförmåga:
Eftersom modellen presterar bra på olika datamängder visar det att den har en stark generaliseringsförmåga och kan anpassa sig till nya, osedda data. Detta är en viktig indikator för alla förutsägningsmodeller, särskilt inom sportförutsägelser där det är viktigt att vara känslig för ny data.

2.3.4 Om korsvalidering användande i modellen

Jag använde tidsseriekorsvalidering (Time Series Split) i båda modellerna för att säkerställa att modellerna fungerar bra på olika datamängder. Detta hjälper oss att se hur bra modellerna kan generalisera.

Inom sportförutsägelser är det viktigt att behålla tidsordningen i träningsdata, eftersom vi endast använder data från tidigare matcher för att göra förutsägelser. Om vi hade använt vanlig korsvalidering, skulle vi riskera att blanda tidsordningen, vilket kan leda till att modellen "ser" framtida data när den gör förutsägelser. Detta skulle ge en missvisande bild av modellens prestanda och kan resultera i överoptimistiska utvärderingar.

Efter att Sebastian påpekade vikten av att använda tidsseriekorsvalidering, gjorde jag nödvändiga justeringar i koden för att implementera detta korrekt. Genom att använda tidsseriekorsvalidering fick jag mer tillförlitliga resultat, och bekräftade att våra modeller fungerar bra på olika datamängder utan att blanda tidsordningen.
Därför är tidsseriekorsvalidering avgörande för att förbättra modellerna och säkerställa att våra förutsägelser är korrekta.

2.3.5 Om olika bedömningsstandarder för två modeller

I detta projekt har jag skapat detta två modeller som används för olika funktioner och syften. Eftersom de två modellerna har olika tillämpningar, så har jag använt olika bedömningsmått för att mäta deras prestanda. Random Forest-modellen används främst för klassificeringsuppgifter, så vi använde ursprungligen noggrannhet som bedömningsstandard. Röstningsregressions modellen används för regressionsuppgifter, och vi använder vanligtvis medelkvadratfel (MSE) för att bedöma dess förutsägelse noggrannhet.

Men efter enligt Sebastian's förslag, för att öka jämförbarheten mellan dessa två modeller, har jag lagt till medelkvadratfel (MSE) och R^2-poäng för att bedöma Random Forest-modellen. Detta gör att Random Forest-modellen kan bedömas inte bara genom noggrannhet utan också med vanliga mått för regressionsmodeller. På samma sätt har Röstningsregressions modellen, förutom att använda medelkvadratfel (MSE), också fått R^2-poäng som bedömning.

Genom att använda olika bedömningsstandarder kan vi analysera och jämföra modellernas prestanda från flera perspektiv. Denna metod hjälper oss att få en mer omfattande förståelse för modellernas styrkor.

2.4. Utmaningar under modellskapandet

Under den första träningen av modellen stötte jag på några problem. För det första fanns det nollvärden i datan eftersom den inte var ordentligt rensad, så jag var tvungen att rengöra den igen. För det andra var modellens resultat för låga i början, mestadels eftersom jag inte hade lagt till tillräckligt många funktioner. Genom att förbättra koden kunde jag lägga till fler relevanta funktioner, inklusive:

- **Win_Percentage:** Simulerad segerprocent
- **home_odds:** Simulerade hemmatsodds
- **Recent_Performance:** Lagets senaste prestation
- **Opponent_Strength:** Motståndarens styrka
- **Venue_Density:** Beräkning av arenans densitet
- **Value_Bet:** Beräkning av värdeodds
- **Is_Value_Bet:** Kategorisera Value_Bet till 1 (om det är positivt) eller 0 (om det inte är det)

```python
import numpy as np

# Generate 'Win_Percentage' feature
data['Win_Percentage'] = np.random.uniform(0.4, 0.7, size=len(data))
print("Win_Percentage feature generated. First few rows:")
print(data[['Win_Percentage']].head())

# Generate 'home_odds' feature
data['home_odds'] = np.random.uniform(1.5, 3.5, size=len(data))
print("home_odds feature generated. First few rows:")
print(data[['home_odds']].head())

# Generate 'Recent_Performance' feature
data['Recent_Performance'] = data['Win_Percentage'] * np.random.uniform(0.8, 1.2, size=len(data))
print("Recent_Performance feature generated. First few rows:")
print(data[['Win_Percentage', 'Recent_Performance']].head())

# Generate 'Opponent_Strength' feature
data['Opponent_Strength'] = np.random.uniform(1, 10, size=len(data))
print("Opponent_Strength feature generated. First few rows:")
print(data[['Opponent_Strength']].head())

# Generate 'Venue_Density' feature
data['Venue_Density'] = data['venue_capacity'] / (data['Opponent_Strength'] + 1e-6)
print("Venue_Density feature generated. First few rows:")
print(data[['venue_capacity', 'Opponent_Strength', 'Venue_Density']].head())

# Generate 'Value_Bet' feature
data['Value_Bet'] = (data['Win_Percentage'] * data['home_odds']) - 1
print("Value_Bet feature generated. First few rows:")
print(data[['Win_Percentage', 'home_odds', 'Value_Bet']].head())
```

Funktioner Processing

```python
# Analyze Feature Importance
# This code calculates and displays the importance of each feature in the model,
# helping to identify which features most influence the prediction of value bets.

import pandas as pd
feature_importances = classifier.feature_importances_
feature_names = X.columns
importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importances})
print(importance_df.sort_values(by='Importance', ascending=False))
```

```
             Feature  Importance
1          home_odds    0.479544
0     Win_Percentage    0.216250
3  Recent_Performance    0.130698
5       Venue_Density    0.062183
2     venue_capacity    0.057198
4  Opponent_Strength    0.054128
```

Detta resultat visar vikten av funktioner viktigheten i modellen

• Funktionernas betydelse:

home_odds är den viktigaste funktionen, vilket betyder att hemmabetsoddset har störst påverkan på resultatet. Detta kan bero på att hemmastödet är viktigt i matcher. Andra funktioner som Win_Percentage och Recent_Performance har också stor betydelse, vilket visar att lagets senaste prestation och vinstprocent är viktiga faktorer för att påverka oddsen.

• Lite nyfiken om funktioner "Opponent Strength"

"Opponent Strength" är en funktion som genereras med np.random.uniform(1, 10, size=len(data)). Detta betyder att "Opponent Strength" i modellen inte baseras på verkliga matchdata eller statistik, utan är ett enkelt slumpmässigt tal. Därför är dess vikt i modellen låg.

Påverkan på oddsförutsägelser:

Även om "Opponent Strength" har låg vikt i förutsägelserna (0.054128), kan den fortfarande hjälpa modellen att ta hänsyn till motståndarens relativa styrka. Idealiskt bör vi använda mer exakta data (som tidigare matchresultat och lagrankningar) för att definiera motståndarkraft för att öka noggrannheten i förutsägelserna.

Insikter bakom resultaten:

Att förstå "Opponent Strength" kanske är en viktig faktor för att förutsäga odds. Genom att analysera "Opponent Strength" prestationer kan vi få en bättre förståelse för dynamiken i matchen och fatta mer informerade vadslagningsbeslut.

2.5. Projektets aktuella status

Vårt projekt behöver fortfarande förbättras, speciellt när det gäller datauppdateringar. Vi har inte en bra kanal för att uppdatera data. När datan ändras måste vi göra om datarensningen och skapa funktioner för att säkerställa att modellen är korrekt och

pålitlig. Dessutom behöver vi träna om modellen när datan har uppdaterats för att passa den nya informationen. Denna process kommer att se till att vår chattbot alltid ger oddsförutsägelser baserat på den senaste informationen, vilket förbättrar användarens upplevelse.

Samtidigt har vårt projekt ännu inte fått en tydlig testning och implementering av chattboten i gruppen. Jag känner mig mycket nyfiken och intresserad av att delta i projektet där vår grupp kan använda Bubble-tjänsten för att bygga chattboten, och genom att skrivit app.py och requirements.txt för att försöka läsa databasen och förutsäga modellen. Jag hoppas att våra respektive uppgiften snart kan testas igen och implementeras.

3. Sammanfattning

Denna projekt är viktig för att lära mig om datavetenskap. Genom att arbeta med rådata som lada ner genom API, rensa den, lägga till funktioner och skapa maskininlärningsmodeller kan vi effektivt förutsäga fotbollsodds. Jag har också lärt mig om appens frontend och backend, samt om LLM-modeller, vilket är ny kunskap för mig. Denna projekt är viktig för mig eftersom den låter mig använda kunskap som jag lärt mig i kursen till verkliga situationer och kan skapa värde.
Om chattboten lyckas, blir det min första gång att skapa en riktig produkt som används på värdefulla sätt i marknaden genom datavetenskap. Det har också inspirerat mig att använda kunskap av datasveteskap för att utveckla värdefulla affärsprojekt.