Projekt i Data Science

Kunskapskontroll_2: Raport: Football Betting Chatbot

ECUTBILDNING

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

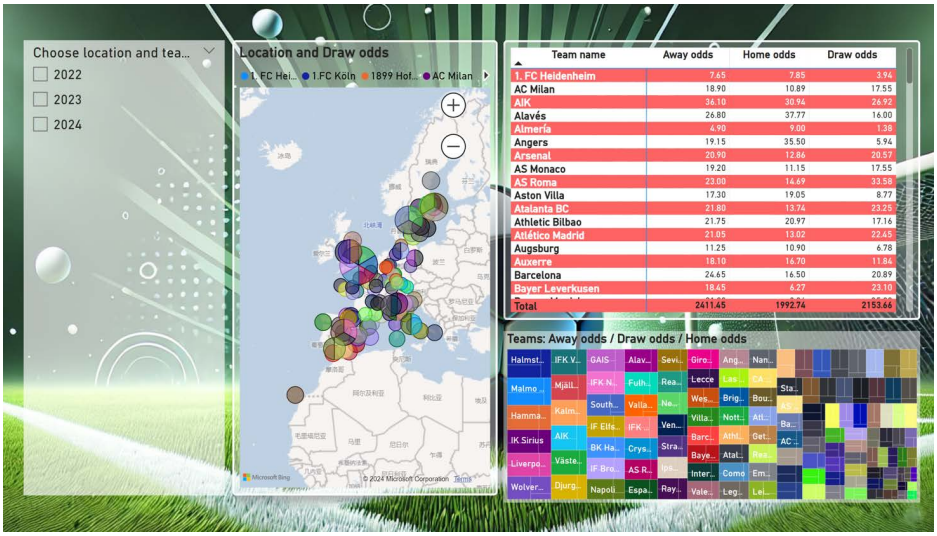
2.3. Min huvudsakliga uppgift

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.

=== Build RandomForestClassifier model (stats_v3)===

```
[426]: # New add features again 2024-10-28

import pandas as pd

# Load the dataset
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)

# Confirm successful loading by displaying the first few rows
print("Data loaded successfully. First few rows:")
print(data.head())
```

```
                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]
```

```
[428]: import numpy as np

# Step 1: Generate 'Win_Percentage' feature
# Simulate win rate between 40% and 70%
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())
```

```
Win_Percentage feature generated. First few rows:
   Win_Percentage
0        0.628610
1        0.492608
2        0.559297
3        0.663224
4        0.488561
```

RandomForestClassifier model kod länk här

[445]:
```
# 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: 0.896551724137931
Precision: 0.8636363636363636
Recall: 1.0
F1 Score: 0.926829268292683
```

[447]:
```
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
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)

# Perform 5-fold cross-validation and compute the accuracy score for each fold
cv_scores = cross_val_score(classifier, X, y, cv=5, scoring='accuracy')

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

```
Cross-validation accuracy scores for each fold: [0.93103448 0.93103448 0.93103448 0.85714286 0.85714286]
Mean cross-validation accuracy: 0.9014778325123153
```

Den ovan bilden visar utvärderingsresultatet från Random Forest-modellen.

• 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.

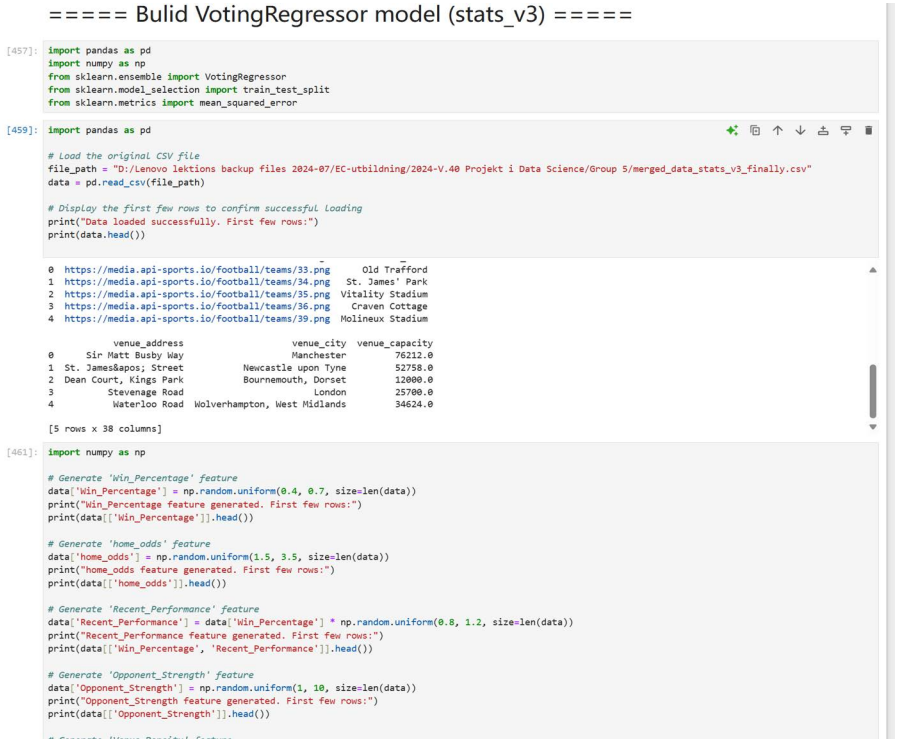
Korsvalideringsnoggrannhet:

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

• **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.

```python
# Cross Validation

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import VotingRegressor, RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Load the cleaned data from CSV
csv_file_path = 'D:/Lenovo lektions backup files 2024-07/EC-utbildning/2024-V.40 Projekt i Data Science/Group 5/merged_data_stats_v3_fin
data = pd.read_csv(csv_file_path)

# Generate features
data['Win_Percentage'] = np.random.uniform(0.4, 0.7, size=len(data))
data['Recent_Performance'] = np.random.uniform(0.8, 1.2, size=len(data))
data['Opponent_Strength'] = np.random.uniform(1, 10, size=len(data))
data['Venue_Density'] = data['venue_capacity'] / 1e6  # Generate density feature
data['Value_Bet'] = (data['Win_Percentage'] * data['home_odds']) - 1  # Example calculation

# Print available columns to verify
print("Columns in the data frame:")
print(data.columns)

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

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize the Voting Regressor with different regressors
voting_regressor = VotingRegressor(estimators=[
    ('rf', RandomForestRegressor(random_state=42)),
    ('lr', LinearRegression())
])

# Train the model
voting_regressor.fit(X_train, y_train)

# Evaluate the model using Mean Squared Error (MSE) on test data
y_pred = voting_regressor.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error of the Voting Regressor model: {mse}")

# Perform cross-validation
cv_scores = cross_val_score(voting_regressor, X, y, cv=5)  # 5-fold cross-validation
print("Cross-validation scores:", cv_scores)
print("Mean cross-validation score:", cv_scores.mean())
```

```
Columns in the data frame:
Index(['match_id', 'league_id', 'season', 'date', 'home_team_id',
       'away_team_id', 'home_score', 'away_score', 'status', 'referee',
       'venue', 'bookmaker', 'home_odds', 'draw_odds', 'away_odds',
       'event_date', 'home_team_name', 'away_team_name', 'player_id',
       'team_id', 'name', 'age', 'nationality', 'position', 'height', 'weight',
       'appearances', 'goals', 'assists', 'yellow_cards', 'red_cards',
       'country', 'founded', 'logo', 'venue_name', 'venue_address',
       'venue_city', 'venue_capacity', 'Win_Percentage', 'Recent_Performance',
       'Opponent_Strength', 'Venue_Density', 'Value_Bet'],
      dtype='object')
Mean Squared Error of the Voting Regressor model: 0.0
Cross-validation scores: [1. 1. 1. 1. 1.]
Mean cross-validation score: 1.0
```

Den ovan bilden visar utvärderingsresultatet för Voting Regressor-modellen.

2.3.4 Om korsvalidering användande

Jag använde korsvalidering i båda modellerna för att säkerställa att de 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 jag bara använder data från tidigare matcher för att göra förutsägelser. Korsvalidering delar upp datamängden i flera delar, så jag tränar och testar modellerna flera gånger. Detta hjälper att se om de fungerar bra på olika data. Det kan också förhindra att modellerna passar för mycket på en specifik datamängd. Därför är korsvalidering viktigt för att förbättra modellernas prestanda och säkerställa att förutsägelserna är korrekta. Genom att använda korsvalidering och göra ytterligare utvärdering fick jag ett bra resultat.

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)

```
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())
```

```
# 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.