Step 1: Import Required Libraries

Begin by importing all the necessary libraries required for data manipulation, visualization, and modeling. These may include libraries such as `pandas`, `numpy`, `matplotlib`, `seaborn`, `scikit-learn`, etc.

Step 2: Load the Dataset

Load the dataset using `pandas` to bring the data into a structured format (DataFrame). This allows for easy manipulation and exploration of the data.

Step 3: Perform Data Analysis

Conduct an initial data analysis using methods like `.info()`, `.describe()`, and value counts to understand the structure of the dataset. This includes checking for the distribution of variables, understanding the types of features, and identifying any anomalies or patterns.

Step 4: Data Cleaning

- **Identify Missing Values**: Investigate the dataset for any missing values using functions like `.isnull()` and `.sum()` to find columns with missing data.
- **Remove Irrelevant Features**: Eliminate unnecessary features that do not contribute to the output or prediction task to simplify the model and reduce noise.
- **Impute Missing Values**: For numerical features, impute missing values using methods such as the median or `KNNImputer`. The median is robust to outliers, while KNN imputation can use the nearest neighbors to estimate missing values more accurately.

Step 5: Calculate Total Possible Goals

Calculate the total number of goals by summing the `home_goals` and `away_goals` columns in the dataset. For example, if the total possible goals sum to 6,879, you can output this as a key statistic for the dataset.

Step 6: Determine Match Winners

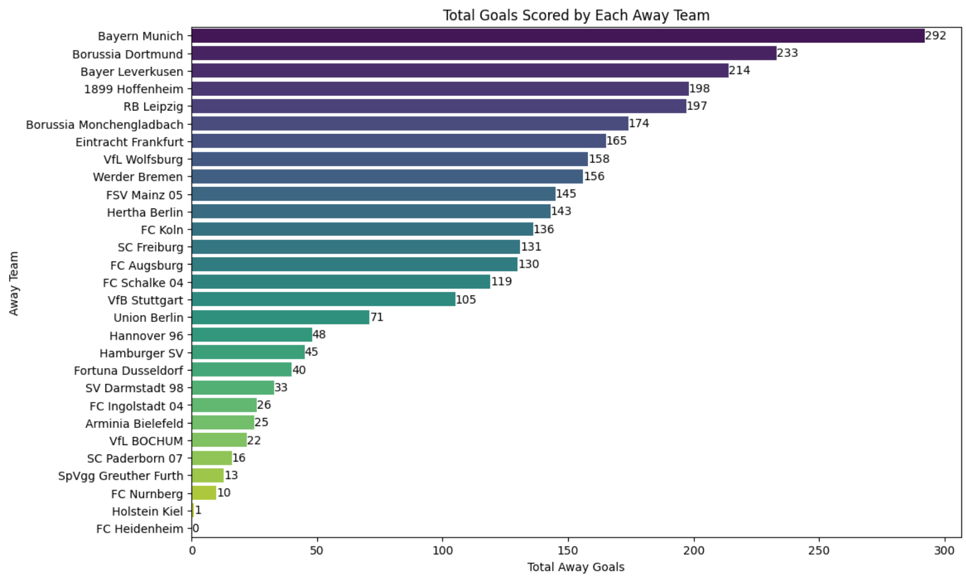
Calculate the winner for each match by comparing `home_goals` and `away_goals`. Create a new column `match_winner` with values 'Home', 'Away', or 'Draw', and then count the occurrences of each result to understand the distribution of match outcomes.

```
Match Winner Counts:
match_winner
Home    1019
Away     707
```

```
Draw     558
```

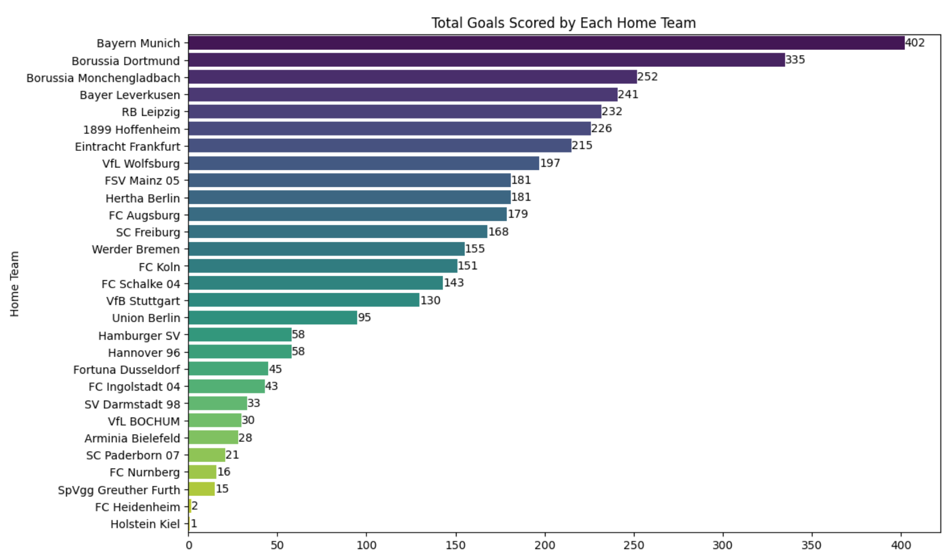
Step 7: Visualize Away Team Performance

Aggregate the total number of goals scored by each away team and sort the teams in descending order based on their total goals. Create a horizontal bar plot to visualize the total goals scored by each away team, with data labels added for clarity.



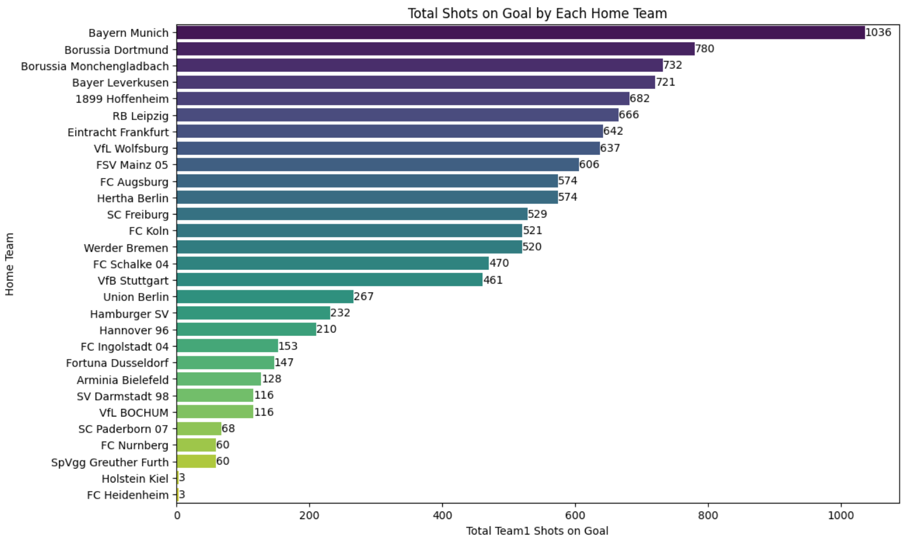
Step 8: Visualize Home Team Performance

Aggregate the total number of goals scored by each home team and sort the teams in descending order based on their total goals. Create a horizontal bar plot to visualize the total goals scored by each home team, with data labels added for clarity.



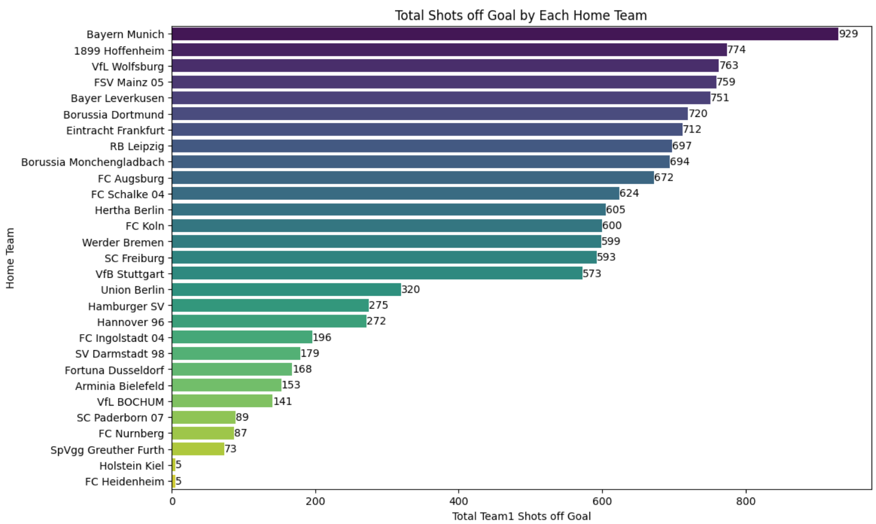
Step 9: Visualize Home Team Shots on Goal

Aggregate the total number of shots on goal for each home team and sort the teams in descending order based on their total shots on goal. Create a horizontal bar plot to illustrate the total shots on goal by each home team, with data labels added to highlight the values.



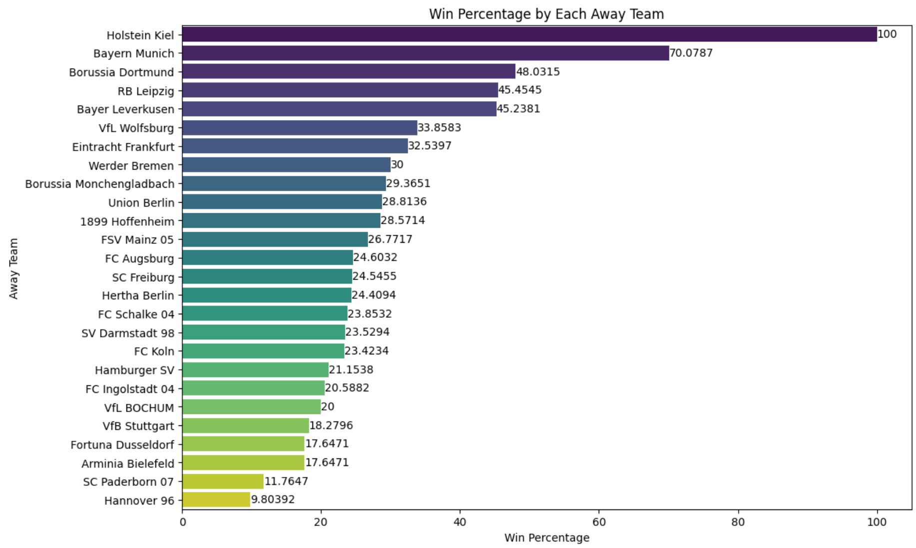
Step 10: Visualize Home Team Shots off Goal

Aggregate and sort the total number of shots off goal for each home team in descending order. Create a horizontal bar plot to display the total shots off goal by each home team, with data labels included to clearly present the values.



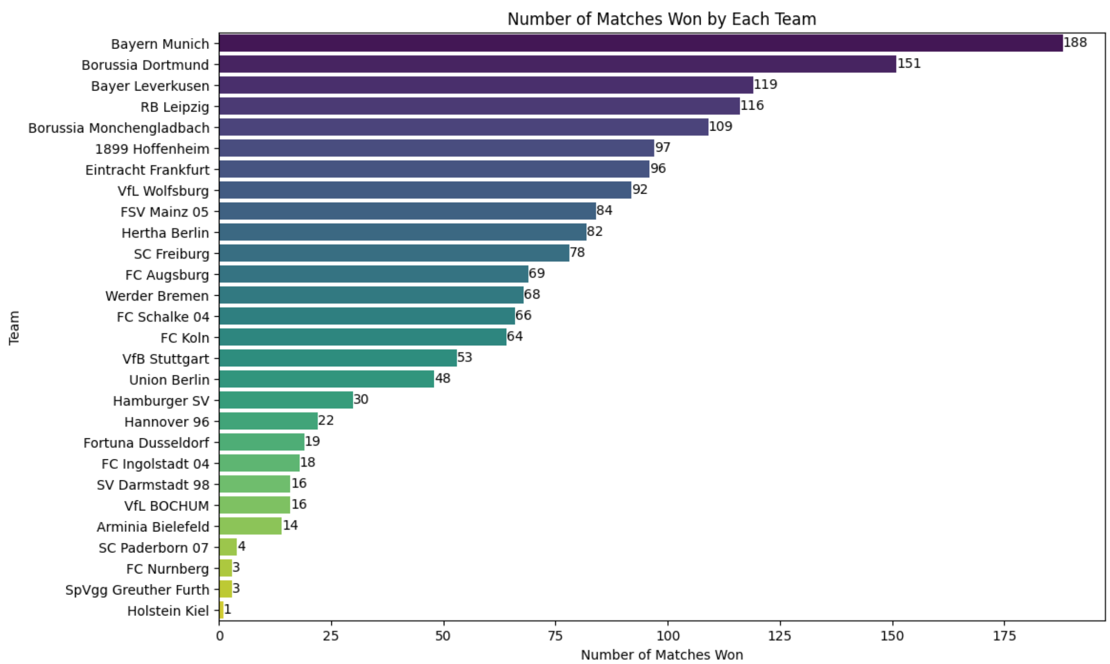
Step 11: Visualize Win Percentage of Away Teams

Calculate and visualize the win percentage for each away team. First, determine the number of wins and total matches for each team, then compute the win percentage. Plot the results in a horizontal bar chart, showing the win percentage of each away team, with data labels to clearly display the values.



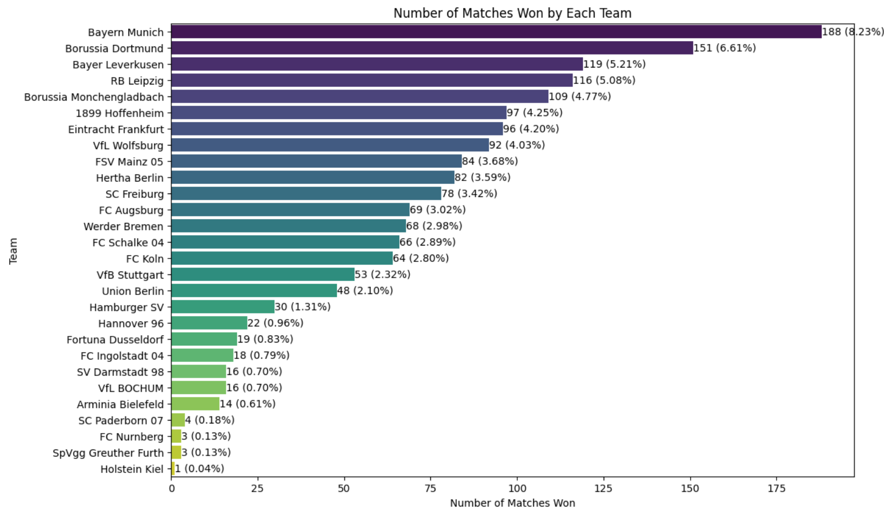
Step 12: Visualize the Number of Matches Won by Each Team

Identify and visualize the number of matches won by each team. Exclude 'Draw' results from the counts, then create a bar plot to display the number of matches won by each team. The plot should include data labels to clearly show the exact number of wins for each team.



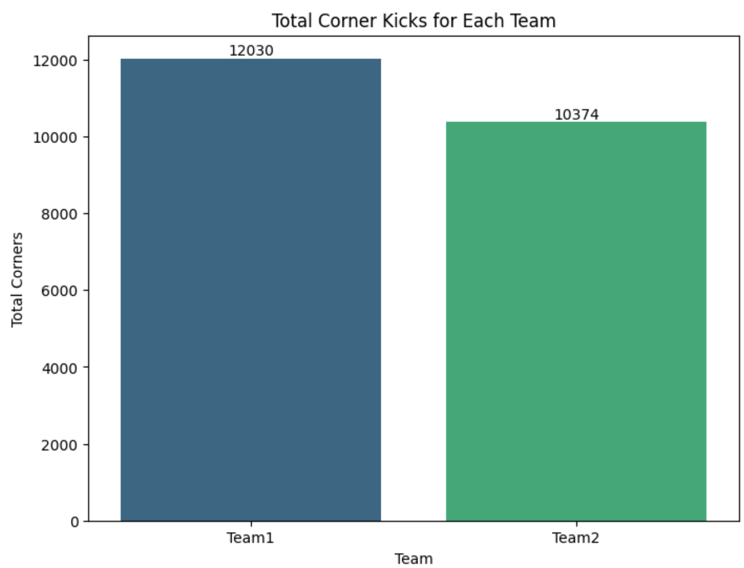
Step 13: Visualize the Percentage of Matches Won by Each Team

Calculate the percentage of matches won by each team and visualize this data. Create a bar plot showing the number of matches won and the percentage of total matches won by each team. Include data labels on the bars to display both the number of wins and the percentage, providing a clear view of each team's performance.



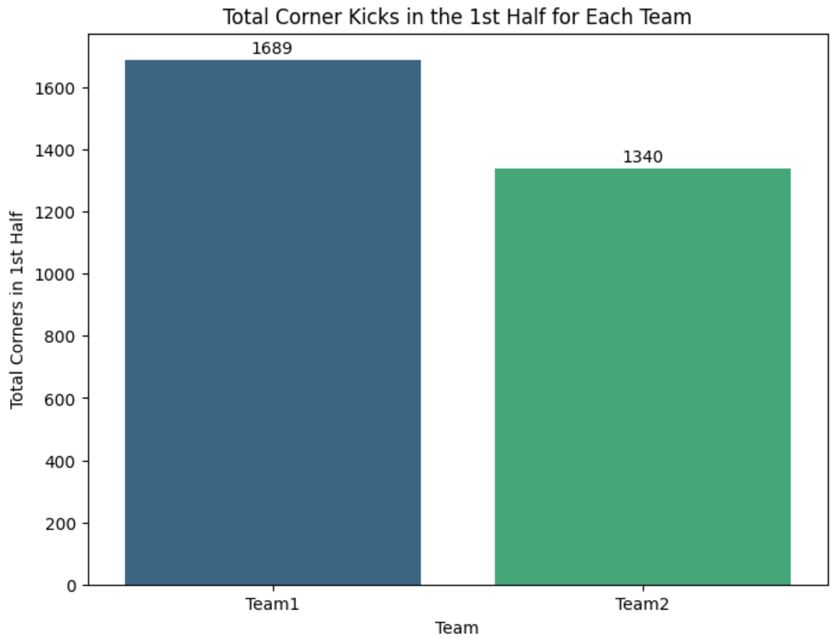
Step 14: Analyze and Visualize Total Corner Kicks

Calculate the total number of corner kicks for both teams (Team1 and Team2) and create a bar plot to visualize this data. Display the total corner kicks for each team with data labels above the bars to clearly show the exact values.



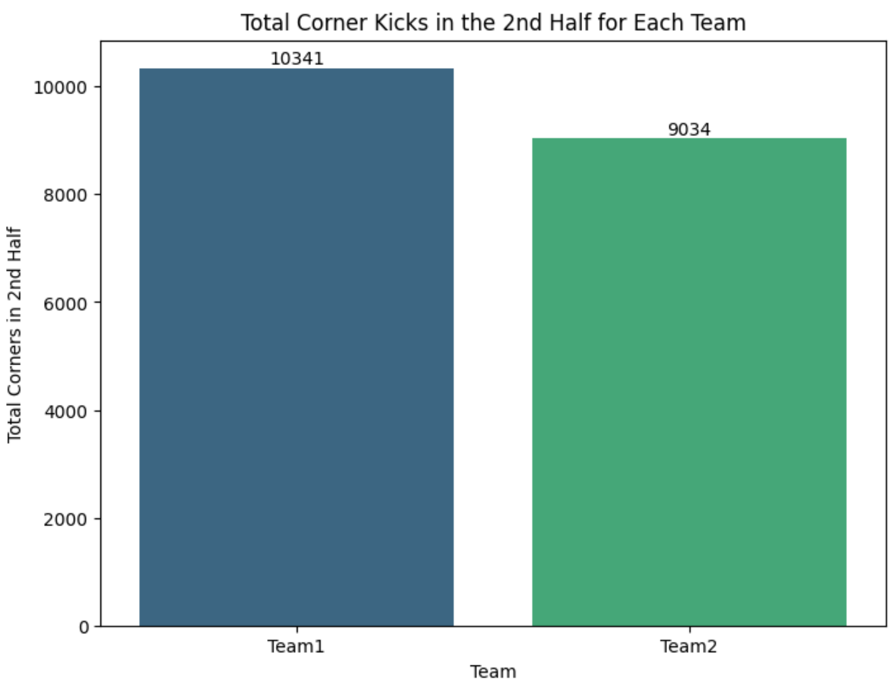
Step 15: Analyze and Visualize Corner Kicks in the 1st Half

Calculate and visualize the total number of corner kicks for each team during the first half of the matches. Display the results using a bar plot, with data labels above each bar indicating the exact number of corner kicks.



Step 16: Analyze and Visualize Corner Kicks in the 2nd Half

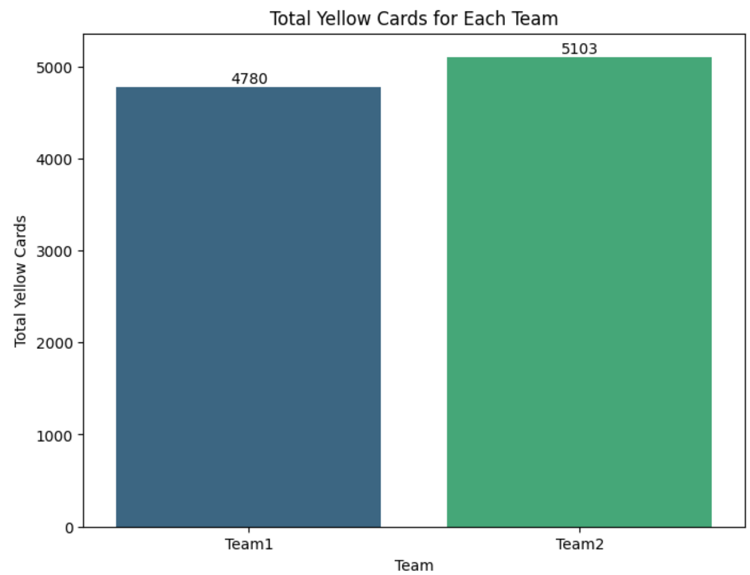
Calculate and visualize the total number of corner kicks for each team during the second half of the matches. Display the results using a bar plot, with data labels above each bar showing the exact number of corner kicks.



Step 17: Analyze and Visualize Yellow Cards

1. **Total Yellow Cards**: Calculate the total number of yellow cards for each team and visualize the results using a bar plot. Add data labels to the bars to show the exact count of yellow cards.

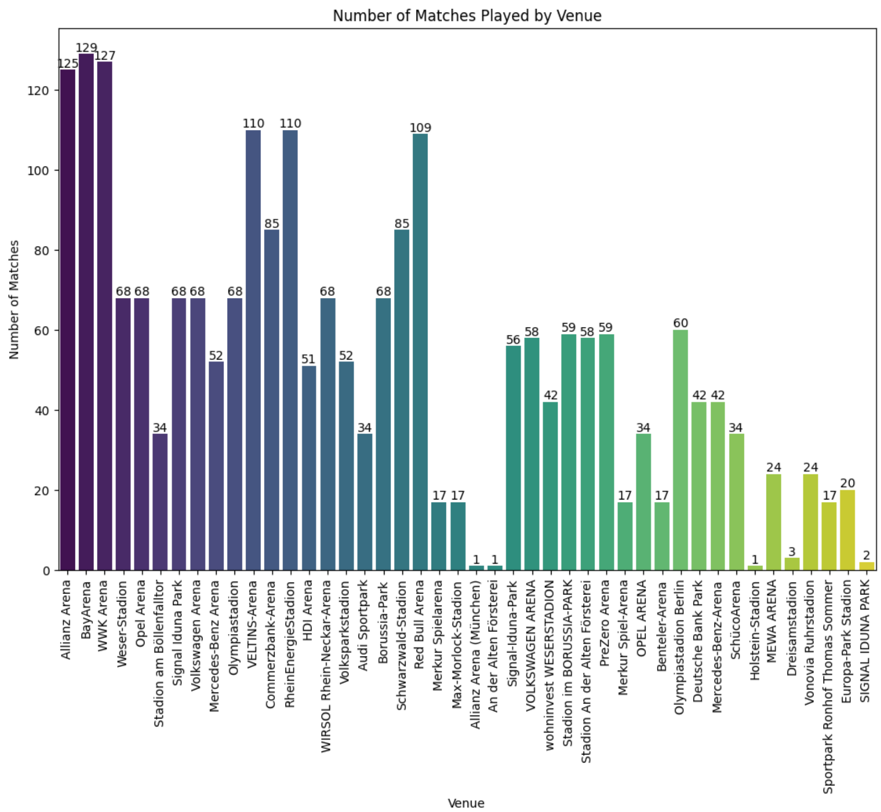
2. **Average Yellow Cards per Match**: Compute the average number of yellow cards per match for each team and display these values. This provides insight into the disciplinary trends of each team during the matches.




```
Average Yellow Cards per Match for Team1: 2.09
Average Yellow Cards per Match for Team2: 2.23
```

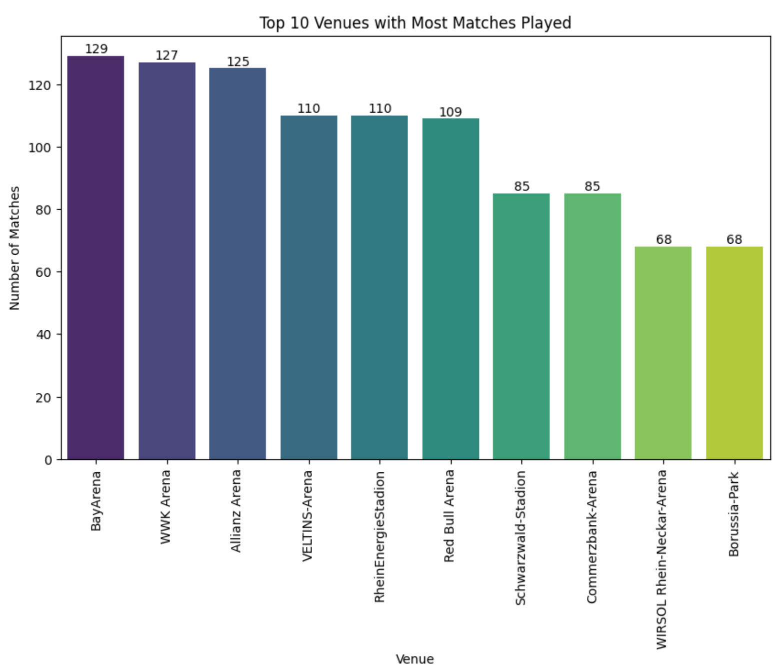
Step 18: Analyze Matches by Venue

1. **Plot**: Use a count plot to display the number of matches played at each venue, with rotated x-axis labels and data labels on the bars.
2. **Purpose**: Identify which venues hosted the most matches and understand venue usage trends.



Step 19: Top 10 Venues Analysis

1. **Plot**: Create a bar plot for the top 10 venues with the highest number of matches played, including rotated x-axis labels for clarity and count labels on the bars.
2. **Purpose**: Highlight the most frequently used venues and visualize venue popularity.



Step 20: Label Encoding

1. **Encoding**: Apply `LabelEncoder` to convert categorical columns ('venue_name', 'city', 'league_name', 'home_name', 'away_name') into numeric values.

Step: Wining Possibility by goal

Step 21:Model Training and Evaluation

1. **Data Preparation**: Encode the `match_winner` column and define features for the model. Split the dataset into training and testing sets.
2. **Model Training**: Train a `RandomForestClassifier` on the training data and make predictions on the test data.
3. **Evaluation**: Print a classification report showing the performance metrics and feature names used in the model.

```
              precision    recall  f1-score   support

        Draw       0.97      0.93      0.95       161
        Home       0.98      1.00      0.99       311
        Away       0.98      0.98      0.98       214

    accuracy                           0.98       686
   macro avg       0.98      0.97      0.97       686
weighted avg       0.98      0.98      0.98       686

Feature Names: ['home_goals', 'away_goals', 'Team1 Shots on Goal',
```

Step 21: Model Training and Evaluation

1. **Feature Definition**:
   - Use features: `home_goals`, `away_goals`, `Team1 Shots on Goal`, `Team2 Shots on Goal`, `Team1 Corner Kicks`, `Team2 Corner Kicks`, `Team1 Fouls`, `Team2 Fouls`.
   - Target: `match_winner_encoded`.
2. **Data Split**:
   - Split data into training and testing sets (30% test).
3. **Train Model**:
   - Train `RandomForestClassifier`.
4. **Evaluate Model**:
   - Print classification report.

```
              precision    recall  f1-score   support

        Draw       0.95      1.00      0.98       161
        Home       1.00      0.99      0.99       311
        Away       1.00      0.98      0.99       214

    accuracy                           0.99       686
   macro avg       0.98      0.99      0.99       686
weighted avg       0.99      0.99      0.99       686
```

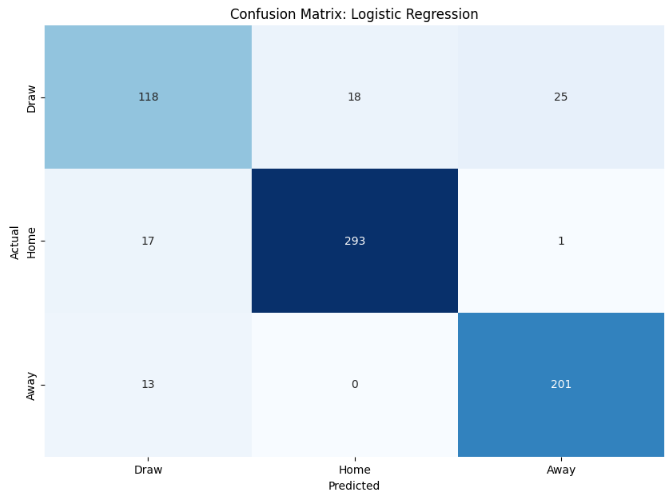
Step 22: Model Comparison and Evaluation

1. **Feature Preparation**:
   - Define the list of features used for modeling.
   - Prepare the dataset and handle missing values using `SimpleImputer`.
2. **Model Training and Evaluation**:
   - Initialize and train several classification models: Logistic Regression, Decision Tree Classifier, Random Forest, Linear Discriminant Analysis, AdaBoost, XGBClassifier, Gradient Boosting, and LGBMClassifier.
   - Evaluate each model's performance using accuracy metrics on both training and validation datasets.
   - Plot confusion matrices to visualize model performance.

```
 # Define models
models = [
    ('Logistic Regression', LogisticRegression(max_iter=1000)),
    ('Decision Tree Classifier', DecisionTreeClassifier()),
    ('Random Forest', RandomForestClassifier()),
    ('Linear Discriminant Analysis', LinearDiscriminantAnalysis()),
    ('Ada Boost', AdaBoostClassifier()),
    ('XGBClassifier', XGBClassifier(eval_metric='mlogloss')),
    ('Gradient Boosting', GradientBoostingClassifier()),
    ('LGBMClassifier', LGBMClassifier())
]
```

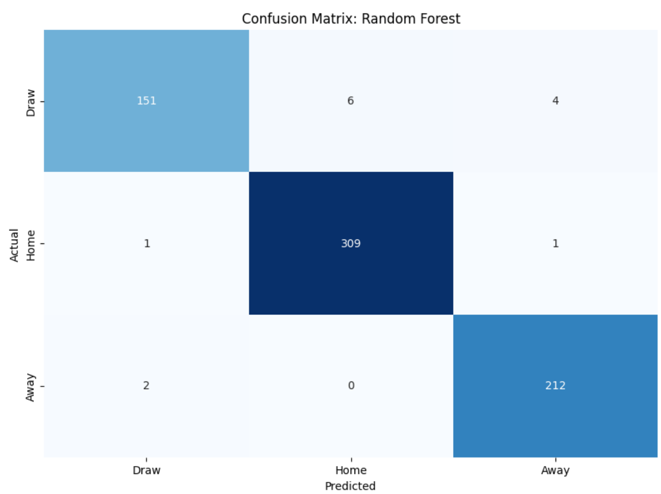
Model: Logistic Regression

Validation Accuracy: 0.892128279883382



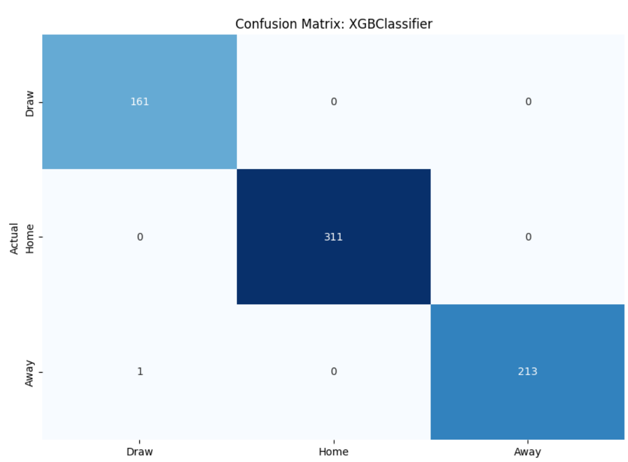
Model: Random Forest

Validation Accuracy: 0.9795918367346939



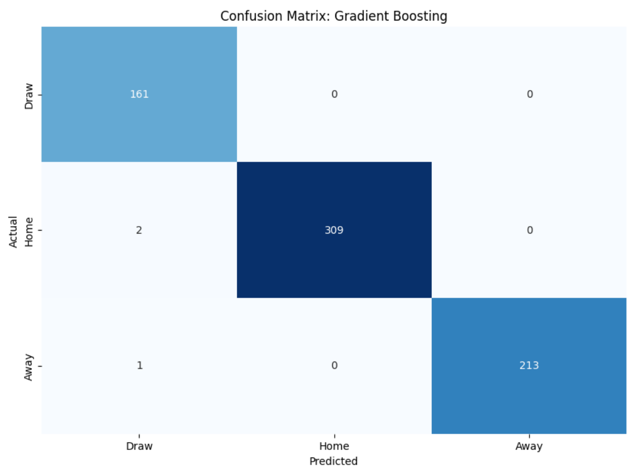
Model: XGBClassifier

Validation Accuracy: 0.9985422740524781



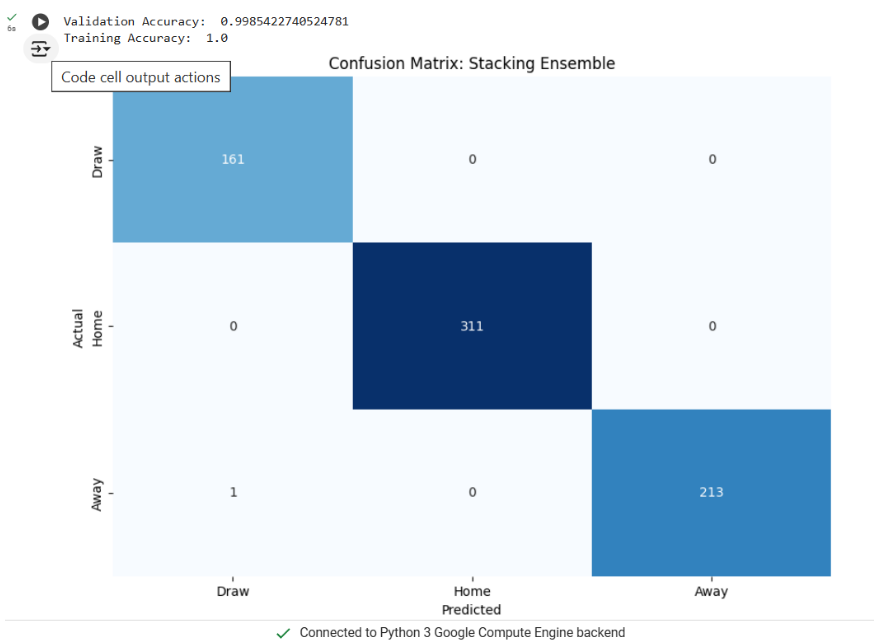
Model: Gradient Boosting

Validation Accuracy: 0.9956268221574344



Step 23: Ensemble Model check combined more than two ML model

1. **Initialize Imputer**: Use `SimpleImputer` with the 'mean' strategy to handle missing values.
2. **Apply Imputer**: Transform training and testing data with the imputer.
3. **Define Models**: Set up `RandomForestClassifier` and `XGBClassifier` as base models, with `LogisticRegression` as the meta-model.
4. **Create and Train Stacking Classifier**: Build and fit a `StackingClassifier` using the base and meta-models.
5. **Predict and Evaluate**: Predict on test data, calculate validation and training accuracies, and display a confusion matrix.

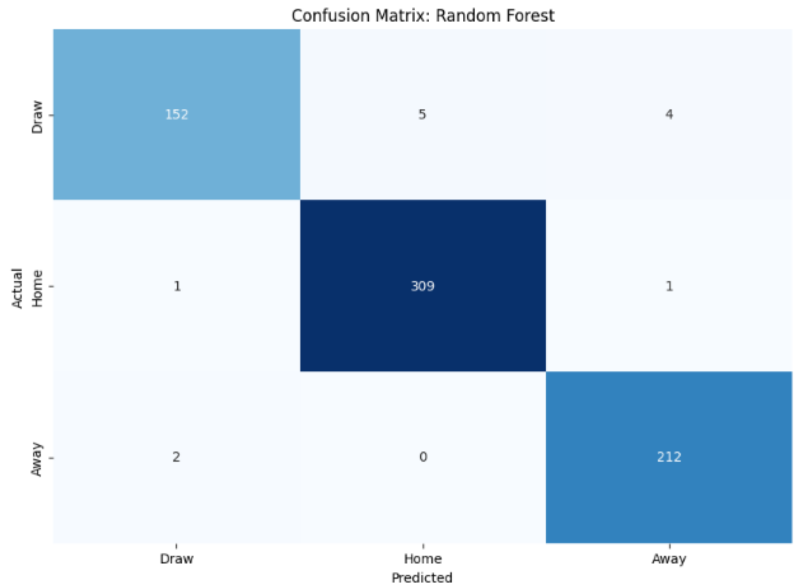


Step 23: Check out with different hyperparameter on Random Forest model

1. **Initialize Imputer**: Use `SimpleImputer` with the 'mean' strategy to handle missing values.

2. **Apply Imputer**: Transform training and testing data with the imputer.
3. **Define Model**: Set up `RandomForestClassifier` with specified parameters.
4. **Train Model**: Fit the RandomForestClassifier on the imputed training data.
5. **Predict**: Make predictions on the test data.
6. **Evaluate**: Calculate and print validation and training accuracies.
7. **Visualize**: Plot and display the confusion matrix to assess model performance.

```
Validation Accuracy:  0.9810495626822158
Training Accuracy:  1.0
```



Step 24: Check out with CNN model

```
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 512)	17,408
dense_6 (Dense)	(None, 3)	1,539

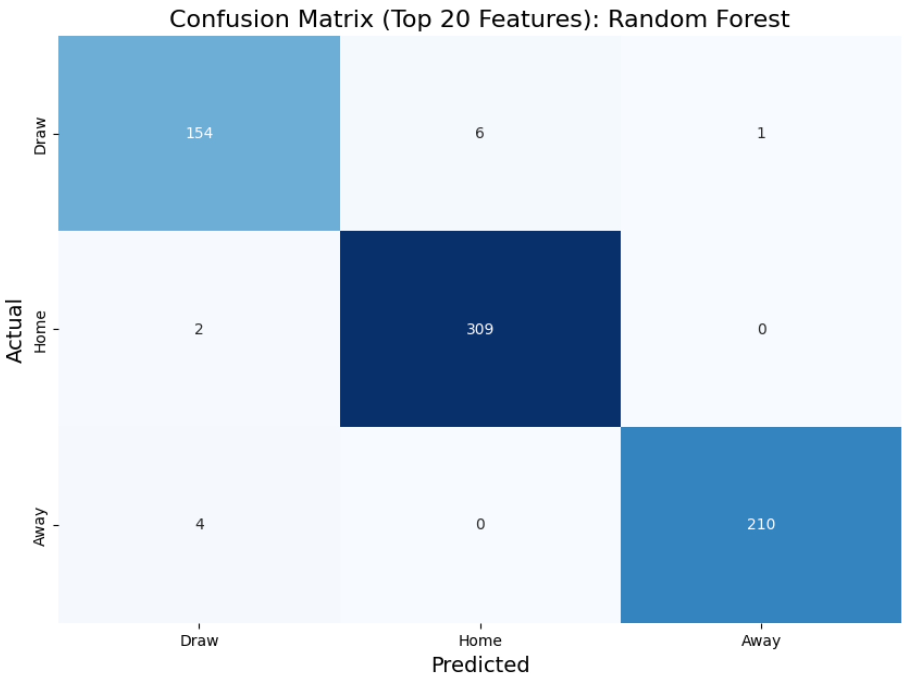
```
Total params: 18,947 (74.01 KB)
Trainable params: 18,947 (74.01 KB)
Non-trainable params: 0 (0.00 B)
Epoch 1/10
40/40 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.5756 - loss: 0.8840 - val_accuracy: 0.7875 - val_loss: 0.5351
Epoch 2/10
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8225 - loss: 0.4981 - val_accuracy: 0.8500 - val_loss: 0.4063
Epoch 3/10
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9057 - loss: 0.3656 - val_accuracy: 0.8969 - val_loss: 0.3268
Epoch 4/10
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9427 - loss: 0.2870 - val_accuracy: 0.9219 - val_loss: 0.2662
Epoch 5/10
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9859 - loss: 0.1927 - val_accuracy: 0.9344 - val_loss: 0.2247
Epoch 6/10
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9860 - loss: 0.1656 - val_accuracy: 0.9625 - val_loss: 0.1798
Epoch 7/10
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9961 - loss: 0.1155 - val_accuracy: 0.9688 - val_loss: 0.1549
Epoch 8/10
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9996 - loss: 0.0895 - val_accuracy: 0.9719 - val_loss: 0.1347
Epoch 9/10
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9998 - loss: 0.0777 - val_accuracy: 0.9781 - val_loss: 0.1125
Epoch 10/10
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0569 - val_accuracy: 0.9781 - val_loss: 0.0996
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9852 - loss: 0.1018
Test Accuracy: 0.9868804812431335
```

Step 25: Validation Accuracy with Top 20 Features: 98.10%

```
Classification Report (Top 20 Features):
               precision    recall  f1-score   support

        Draw       0.96      0.96      0.96       161
        Home       0.98      0.99      0.99       311
        Away       1.00      0.98      0.99       214

    accuracy                           0.98       686
   macro avg       0.98      0.98      0.98       686
weighted avg       0.98      0.98      0.98       686
```



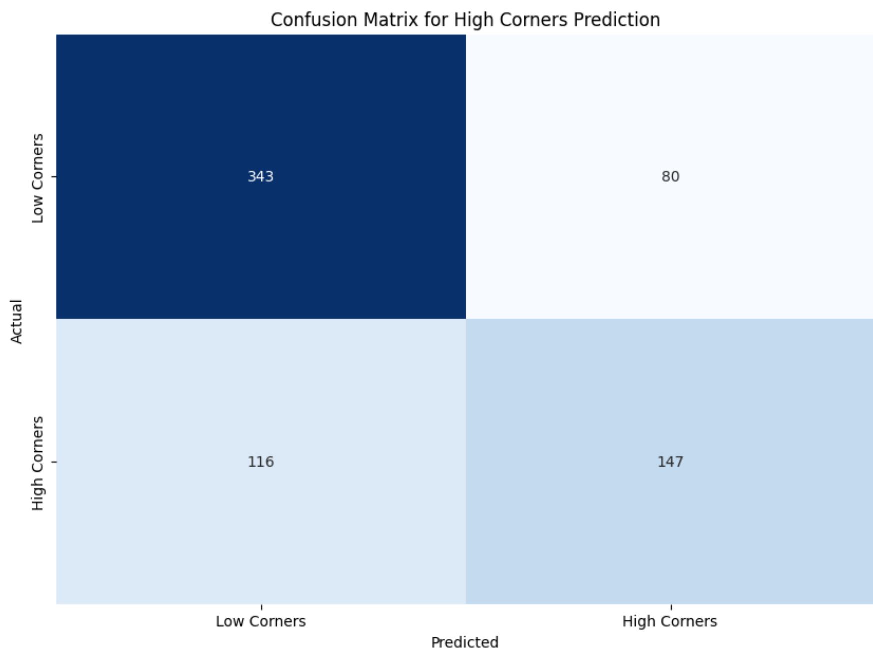
Corner kick Prediction

Step :26, predictions are made using the trained RandomForestClassifier on the test data. The validation accuracy is calculated and printed to evaluate the model's performance. Additionally, a classification report is generated to provide detailed metrics on the model’s accuracy for the 'Low Corners' and 'High Corners' categories.

```
Validation Accuracy:  0.7142857142857143
              precision    recall  f1-score   support

 Low Corners       0.75      0.81      0.78       423
High Corners       0.65      0.56      0.60       263

    accuracy                           0.71       686
   macro avg       0.70      0.68      0.69       686
weighted avg       0.71      0.71      0.71       686
```

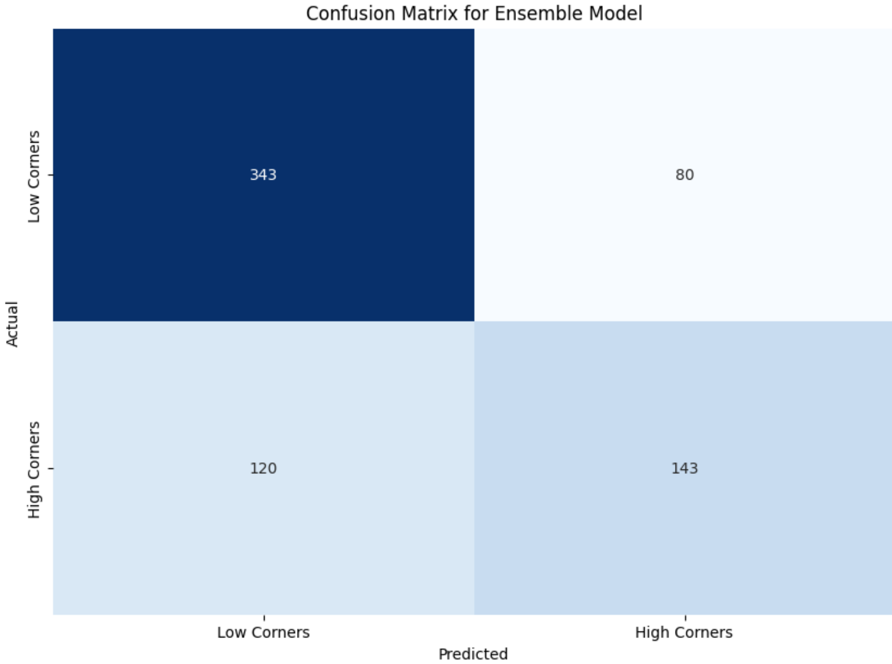


Step 26: **Multiple ML Model Check**

- Model Scores Summary:
- Logistic Regression: 0.7362
- Decision Tree Classifier: 0.6385
- Random Forest: 0.7201
- Linear Discriminant Analysis: 0.7274
- Ada Boost: 0.7041
- XGBClassifier: 0.6647
- Gradient Boosting: 0.7187
- LGBMClassifier: 0.6895

Step:27 Ensemble Model combined Rnadom foreset xgb and logistics regression ml model

`Ensemble Validation Accuracy: 0.7085`



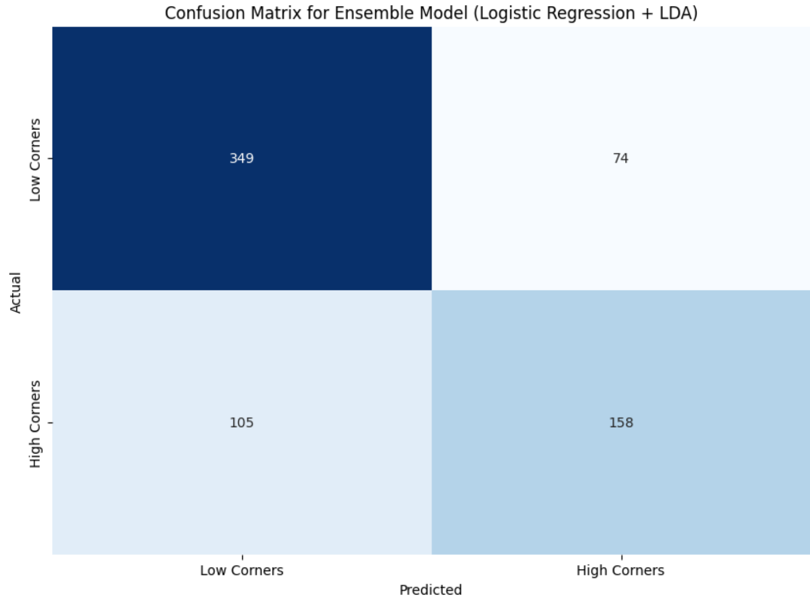
Step 28: Ensemble Model combined Linear Discriminant Analysis and logistics regression ml model

```
Ensemble Validation Accuracy: 0.7391
Cross-Validation Mean Accuracy: 0.7058

Classification Report:
               precision    recall  f1-score   support

 Low Corners       0.77      0.83      0.80       423
High Corners       0.68      0.60      0.64       263

    accuracy                           0.74       686
   macro avg       0.72      0.71      0.72       686
weighted avg       0.74      0.74      0.74       686
```



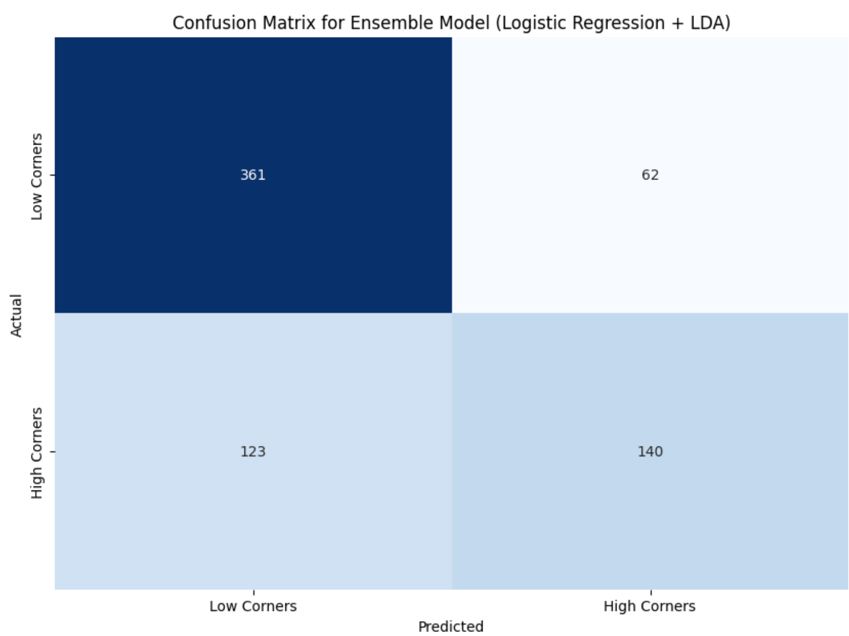
Step 29: Check out with different Hyper parameter

```
Ensemble Validation Accuracy: 0.7303
Best Logistic Regression Parameters: {'C': 0.5, 'penalty': 'l1', 'solver': 'saga'}


Classification Report:
               precision    recall  f1-score   support

 Low Corners       0.75      0.85      0.80       423
High Corners       0.69      0.53      0.60       263

    accuracy                           0.73       686
   macro avg       0.72      0.69      0.70       686
weighted avg       0.73      0.73      0.72       686
```



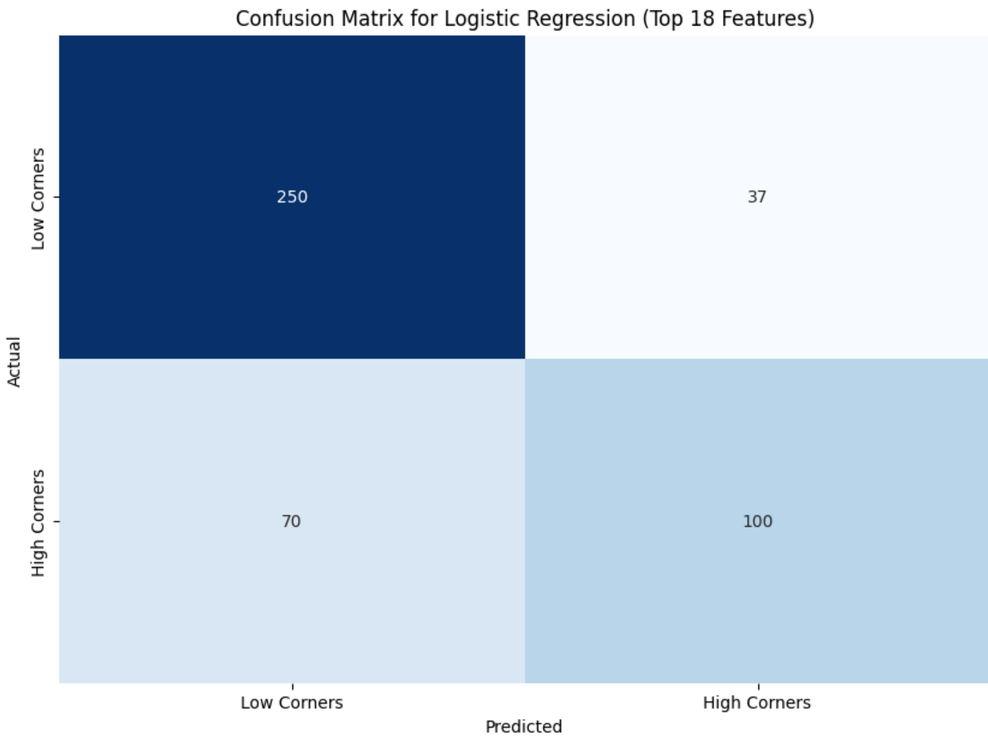
Step 30: Accuracy improves for best 18 features on Logistic Regression (Improve)

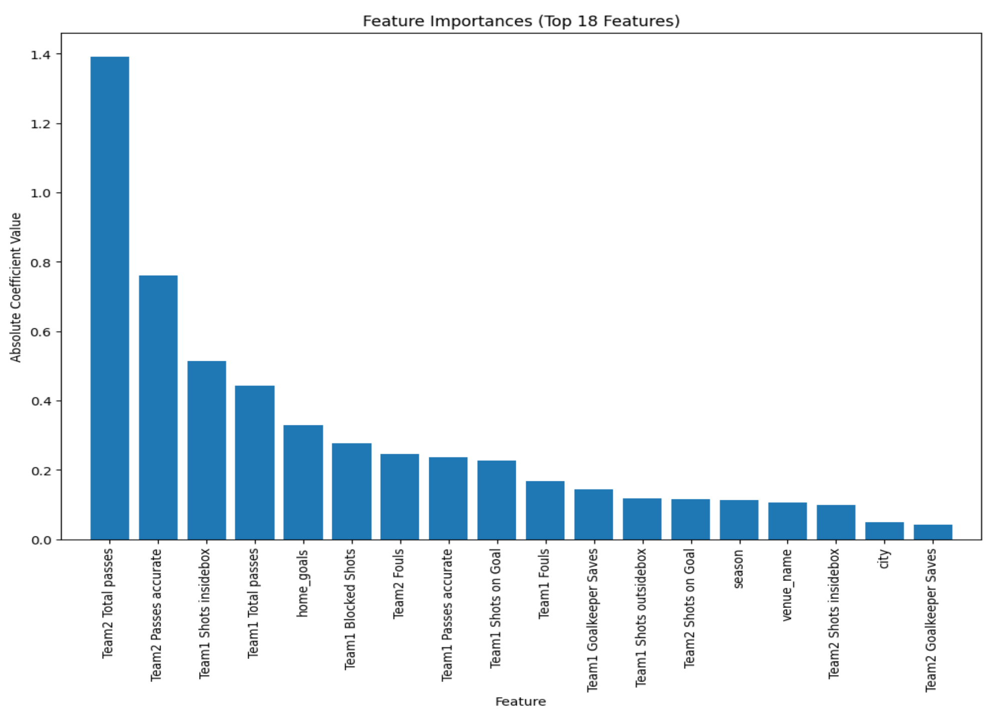
```
Validation Accuracy with Top 18 Features: 0.7659

Classification Report (Top 18 Features):
               precision    recall  f1-score   support

 Low Corners       0.78      0.87      0.82       287
High Corners       0.73      0.59      0.65       170

    accuracy                           0.77       457
   macro avg       0.76      0.73      0.74       457
weighted avg       0.76      0.77      0.76       457
```





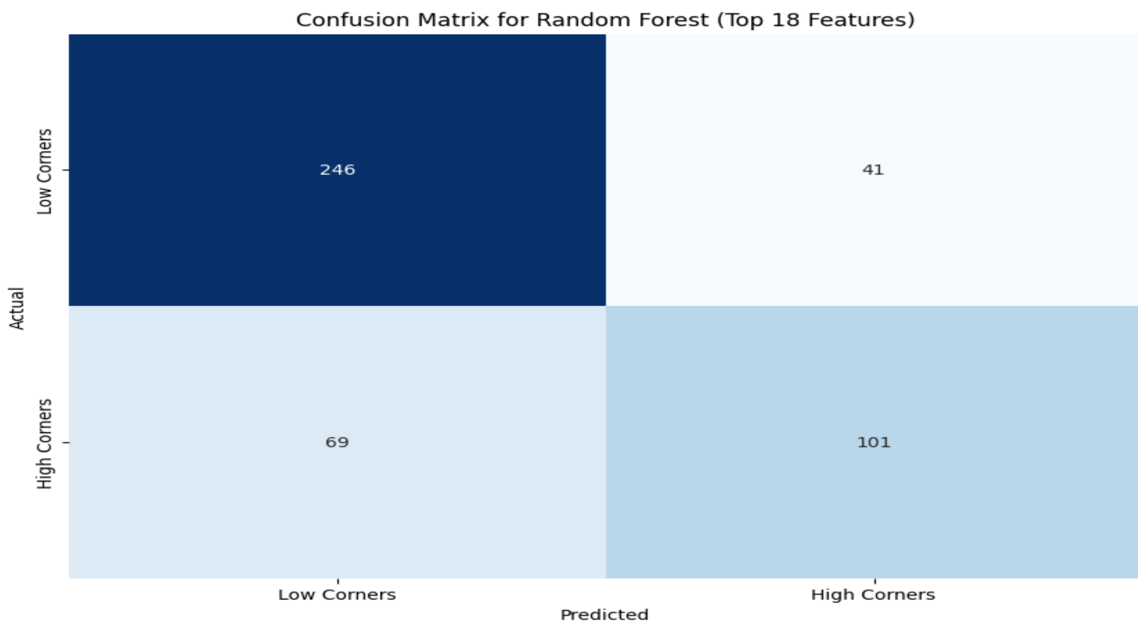
Step 31: Accuracy improves for best 18 features on Random forest model(Improve)

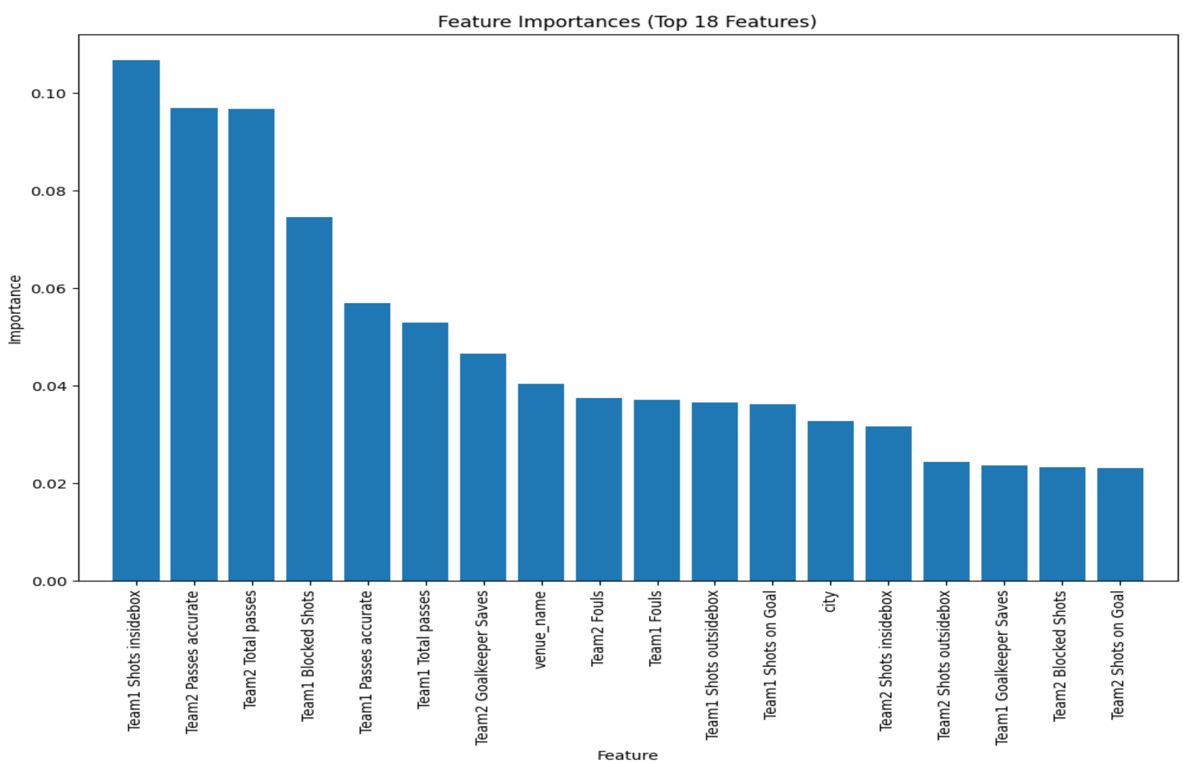
```
Validation Accuracy with Top 18 Features: 0.7593

Classification Report (Top 18 Features):
               precision    recall  f1-score   support

 Low Corners       0.78      0.86      0.82       287
High Corners       0.71      0.59      0.65       170

    accuracy                           0.76       457
   macro avg       0.75      0.73      0.73       457
weighted avg       0.76      0.76      0.75       457
```





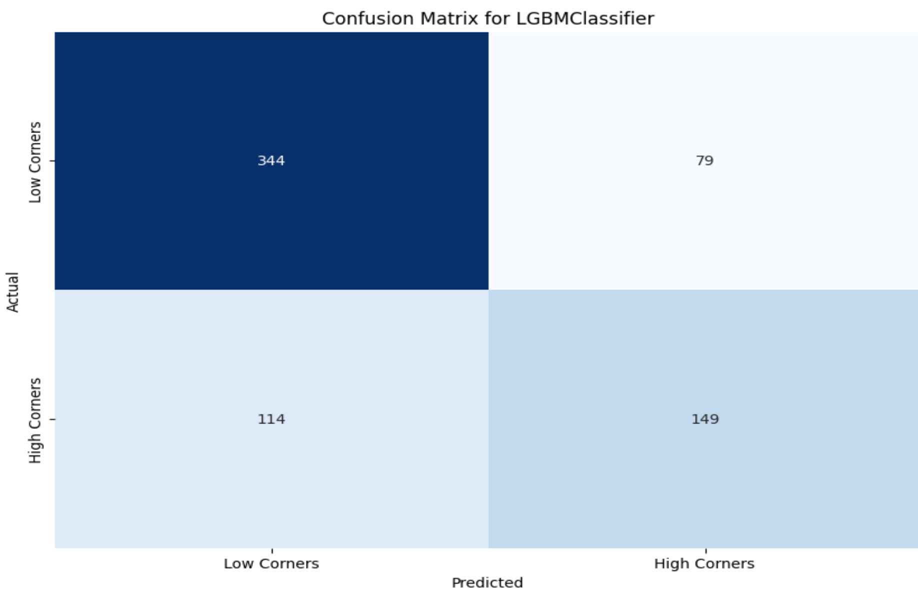
Step 32: Check out using different Hyper *parameter

```
Best LGBMClassifier Validation Accuracy: 0.7187
Best LGBMClassifier Parameters: {'boosting_type': 'dart', 'learning_rate': 0.1,
'n_estimators': 200, 'num_leaves': 31, 'objective': 'binary'}


Classification Report:
               precision    recall  f1-score   support

 Low Corners       0.75      0.81      0.78       423
High Corners       0.65      0.57      0.61       263

    accuracy                           0.72       686
   macro avg       0.70      0.69      0.69       686
weighted avg       0.71      0.72      0.71       686
```



Step 33: Check out with CNN

```
Epoch 47/50
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0018 - val_accuracy: 0.6562 - val_loss: 1.7078
Epoch 48/50
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0018 - val_accuracy: 0.6562 - val_loss: 1.7209
Epoch 49/50
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0017 - val_accuracy: 0.6562 - val_loss: 1.7091
Epoch 50/50
40/40 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0017 - val_accuracy: 0.6531 - val_loss: 1.7492
22/22 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6492 - loss: 1.6232
Test Accuracy: 0.6501457691192627
```

Step 34: Check out specific features on Random forest model

```
# Example feature and target variable preparation
features = ['home_goals', 'away_goals', 'Team1 Shots on Goal', 'Team2 Shots on Goal',
              'Team1 Corner Kicks', 'Team2 Corner Kicks', 'Team1 Fouls', 'Team2
Fouls']  # Add relevant features
```

```
Validation Accuracy: 1.0000

Classification Report:
               precision    recall  f1-score   support

 Low Corners       1.00      1.00      1.00       423
High Corners       1.00      1.00      1.00       263

    accuracy                           1.00       686
   macro avg       1.00      1.00      1.00       686
weighted avg       1.00      1.00      1.00       686
```