IPL First Innings Score Prediction

Problem Statement: This project aims to predict the first innings total score in an IPL match using machine learning. The model will analyze factors such as batting and bowling teams, venue, overs, runs, wickets, and recent performance trends. By leveraging historical data, it will provide real-time score projections to assist teams, analysts, and broadcasters. The predictions can help in strategic decision-making and enhance viewer engagement.

Column Name - Description

mid------------Unique match ID date-----------Date of the match venue-----------Stadium where the match was played bat_team-------Name of the batting team bowl_team------Name of the bowling team batsman--------Name of the batsman on strike bowler---------Name of the current bowler runs-----------Runs scored so far in the innings wickets--------Wickets fallen so far in the innings overs----------Overs completed in the innings runs_last_5----Runs scored in the last 5 overs wickets_last_5-Wickets lost in the last 5 overs striker--------Runs scored by the current batsman non-striker----Runs scored by the non-striker batsman total----------The actual first innings total score (Target variable)

In [24]:

```
# IPL First Innings Score Prediction - Jupyter Notebook
## Step 1: Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sc
from sklearn.preprocessing import OneHotEncoder
from sklearn.feature_selection import RFE
```

In [25]:

```
df = pd.read_csv('ipl.csv')
display(df.head())
print(df.shape)
print(df.dtypes)
```

	mid	date	venue	bat_team	bowl_team	batsman	bowler	runs	wickets	overs
0	1	18-04-2013	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	SC Ganguly	P Kumar	1	0	0.1
1	1	18-04-2013	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	1	0	0.2
2	1	18-04-2013	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.2
3	1	18-04-2013	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.3
4	1	18-04-2013	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.4

```
(76014, 15)
mid                 int64
date               object
venue              object
bat_team           object
bowl_team          object
batsman            object
bowler             object
runs                int64
wickets             int64
overs             float64
runs_last_5         int64
wickets_last_5      int64
striker             int64
non-striker         int64
total               int64
dtype: object
```

In [26]:

```
print("Dataset Info:")
df.info()
print("\nSummary Statistics:")
print(df.describe())
print("\nChecking for Missing Values:")
print(df.isnull().sum())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76014 entries, 0 to 76013
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   mid             76014 non-null  int64  
 1   date            76014 non-null  object 
 2   venue           76014 non-null  object 
 3   bat_team        76014 non-null  object 
 4   bowl_team       76014 non-null  object 
 5   batsman         76014 non-null  object 
 6   bowler          76014 non-null  object 
 7   runs            76014 non-null  int64  
 8   wickets         76014 non-null  int64  
 9   overs           76014 non-null  float64
 10  runs_last_5     76014 non-null  int64  
 11  wickets_last_5  76014 non-null  int64  
 12  striker         76014 non-null  int64  
 13  non-striker     76014 non-null  int64  
 14  total           76014 non-null  int64  
dtypes: float64(1), int64(8), object(6)
memory usage: 8.7+ MB

Summary Statistics:
                mid          runs       wickets         overs   runs_last
_5  \
count  76014.000000  76014.000000  76014.000000  76014.000000  76014.0000
00   
mean     308.627740     74.889349      2.415844      9.783068     33.2164
34   
std      178.156878     48.823327      2.015207      5.772587     14.9141
74   
min        1.000000      0.000000      0.000000      0.000000      0.0000
00   
25%      154.000000     34.000000      1.000000      4.600000     24.0000
00   
50%      308.000000     70.000000      2.000000      9.600000     34.0000
00   
75%      463.000000    111.000000      4.000000     14.600000     43.0000
00   
max      617.000000    263.000000     10.000000     19.600000    113.0000
00   

       wickets_last_5       striker   non-striker         total  
count    76014.000000  76014.000000  76014.000000  76014.000000  
mean         1.120307     24.962283      8.869287    160.901452  
std          1.053343     20.079752     10.795742     29.246231  
min          0.000000      0.000000      0.000000     67.000000  
25%          0.000000     10.000000      1.000000    142.000000  
50%          1.000000     20.000000      5.000000    162.000000  
75%          2.000000     35.000000     13.000000    181.000000  
max          7.000000    175.000000    109.000000    263.000000  

Checking for Missing Values:
mid               0
date              0
venue             0
bat_team          0
bowl_team         0
batsman           0
```

```
bowler            0
runs              0
wickets           0
overs             0
runs_last_5       0
wickets_last_5    0
striker           0
non-striker       0
total             0
dtype: int64
```

In [27]:

```
columns_to_remove = ['mid', 'venue', 'batsman', 'bowler', 'striker', 'non-
existing_columns = [col for col in columns_to_remove if col in df.columns]
print('Before removing unwanted columns:', df.shape)
df.drop(columns=existing_columns, inplace=True)
print('After removing unwanted columns:', df.shape)
```

```
Before removing unwanted columns: (76014, 15)
After removing unwanted columns: (76014, 9)
```

In [28]:

```
# Keeping only consistent teams if 'bat_team' and 'bowl_team' exist
if 'bat_team' in df.columns and 'bowl_team' in df.columns:
    consistent_teams = ['Kolkata Knight Riders', 'Chennai Super Kings', 'R
                        'Kings XI Punjab', 'Royal Challengers Bangalore',
    df = df[(df['bat_team'].isin(consistent_teams)) & (df['bowl_team'].isi
```

In [29]:

```
if 'overs' in df.columns:
    df = df[df['overs'] >= 5.0]
```

In [30]:

```
if 'date' in df.columns:
    df['date'] = pd.to_datetime(df['date'], errors='coerce')
    df = df.dropna(subset=['date'])  # Remove rows where date conversion f
```

```
C:\Users\Jagadeshwar reddy\AppData\Local\Temp\ipykernel_12032\823937039.p
y:2: UserWarning: Parsing dates in %d-%m-%Y format when dayfirst=False (t
he default) was specified. Pass `dayfirst=True` or specify a format to si
lence this warning.
  df['date'] = pd.to_datetime(df['date'], errors='coerce')
```

In [31]:

```
# Step 3: Data Preprocessing - Handling Categorical Variables
if 'bat_team' in df.columns and 'bowl_team' in df.columns:
    encoder = OneHotEncoder(drop='first', sparse=False, handle_unknown='ig
    categorical_features = ['bat_team', 'bowl_team']
    categorical_encoded = encoder.fit_transform(df[categorical_features])
    categorical_df = pd.DataFrame(categorical_encoded, columns=encoder.get
    df = pd.concat([df.drop(columns=categorical_features), categorical_df]

# Splitting data into training and test set based on date
if 'date' in df.columns:
    X = df.drop(columns=['total', 'date'])
    y = df['total']
    X_train = X[df['date'].dt.year <= 2016]
    X_test = X[df['date'].dt.year >= 2017]
    y_train = y[df['date'].dt.year <= 2016]
    y_test = y[df['date'].dt.year >= 2017]

# Handling missing values in X_train and X_test
X_train = X_train.dropna()
X_test = X_test.dropna()
y_train = y_train.loc[X_train.index]  # Aligning target variable
y_test = y_test.loc[X_test.index]
```

```
J:\New folder\Lib\site-packages\sklearn\preprocessing\_encoders.py:972: F
utureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and
will be removed in 1.4. `sparse_output` is ignored unless you leave `spar
se` to its default value.
  warnings.warn(
```

In [32]:

```
# Step 4: Building Simple Linear Regression Model (Using Overs as the Pred
lr_simple = LinearRegression()
lr_simple.fit(X_train[['overs']], y_train)
y_pred_slr = lr_simple.predict(X_test[['overs']])

# Evaluation for SLR
print("Simple Linear Regression Performance:")
print("MAE:", mean_absolute_error(y_test, y_pred_slr))
print("MSE:", mean_squared_error(y_test, y_pred_slr))
print("R2 Score:", r2_score(y_test, y_pred_slr))
```

```
Simple Linear Regression Performance:
MAE: 23.618145881488104
MSE: 866.2753922330978
R2 Score: -0.007003059901813595
```

In [33]:

```python
# Step 5: Multiple Linear Regression (MLR)
lr_multiple = LinearRegression()
lr_multiple.fit(X_train, y_train)
y_pred_mlr = lr_multiple.predict(X_test)

# Evaluation for MLR
print("\nMultiple Linear Regression Performance:")
print("MAE:", mean_absolute_error(y_test, y_pred_mlr))
print("MSE:", mean_squared_error(y_test, y_pred_mlr))
print("R2 Score:", r2_score(y_test, y_pred_mlr))
```

```
Multiple Linear Regression Performance:
MAE: 14.349265728088769
MSE: 358.7804871686855
R2 Score: 0.5829351131488479
```

In [34]:

```python
# Step 6: Feature Selection using Recursive Feature Elimination (RFE)
rfe = RFE(estimator=LinearRegression(), n_features_to_select=5)
rfe.fit(X_train, y_train)
selected_features = X_train.columns[rfe.support_]
print("\nSelected Features after RFE:", selected_features)
```

```
Selected Features after RFE: Index(['wickets', 'bat_team_Sunrisers Hydera
bad', 'bowl_team_Delhi Daredevils',
       'bowl_team_Rajasthan Royals', 'bowl_team_Sunrisers Hyderabad'],
      dtype='object')
```

In [35]:

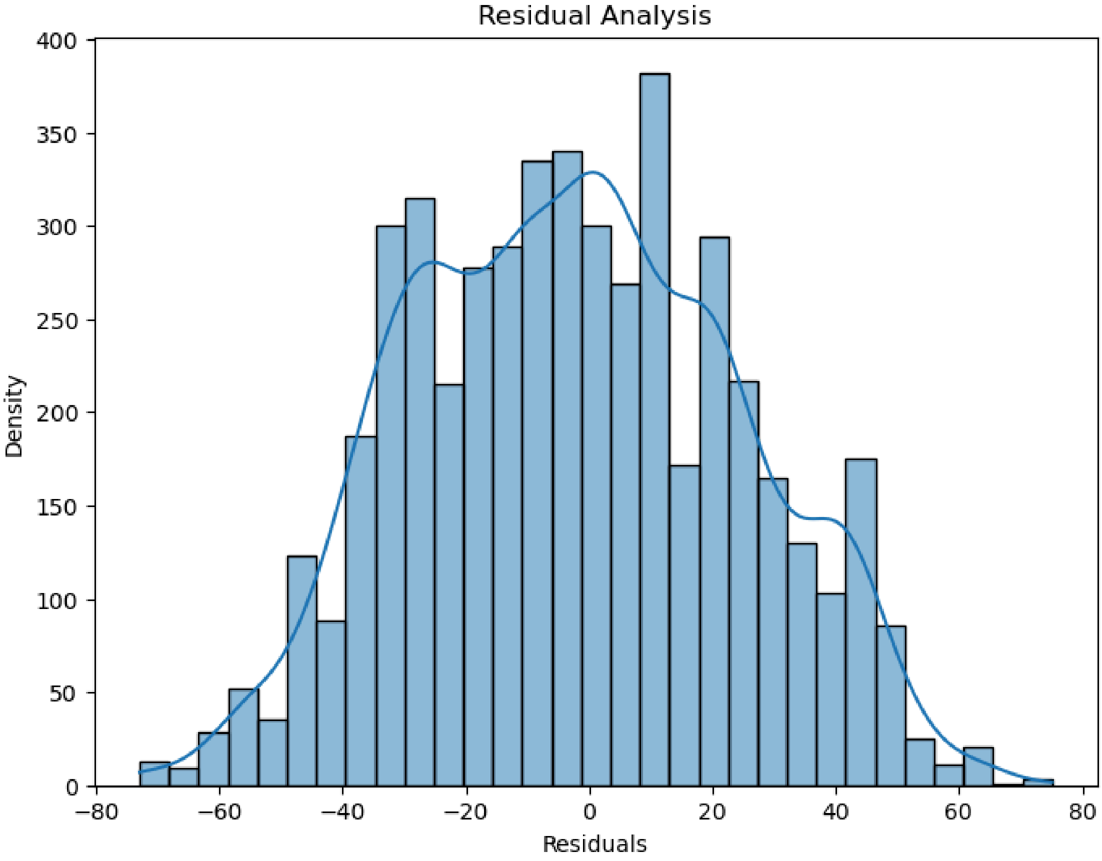
```python
# Step 7: Building Model with Selected Features
X_train_rfe = X_train[selected_features]
X_test_rfe = X_test[selected_features]
lr_rfe = LinearRegression()
lr_rfe.fit(X_train_rfe, y_train)
y_pred_rfe = lr_rfe.predict(X_test_rfe)

# Evaluation for RFE-selected model
print("\nModel Performance after Feature Selection:")
print("MAE:", mean_absolute_error(y_test, y_pred_rfe))
print("MSE:", mean_squared_error(y_test, y_pred_rfe))
print("R2 Score:", r2_score(y_test, y_pred_rfe))
```

```
Model Performance after Feature Selection:
MAE: 21.98015199087654
MSE: 712.9418122110704
R2 Score: 0.17123977787157207
```

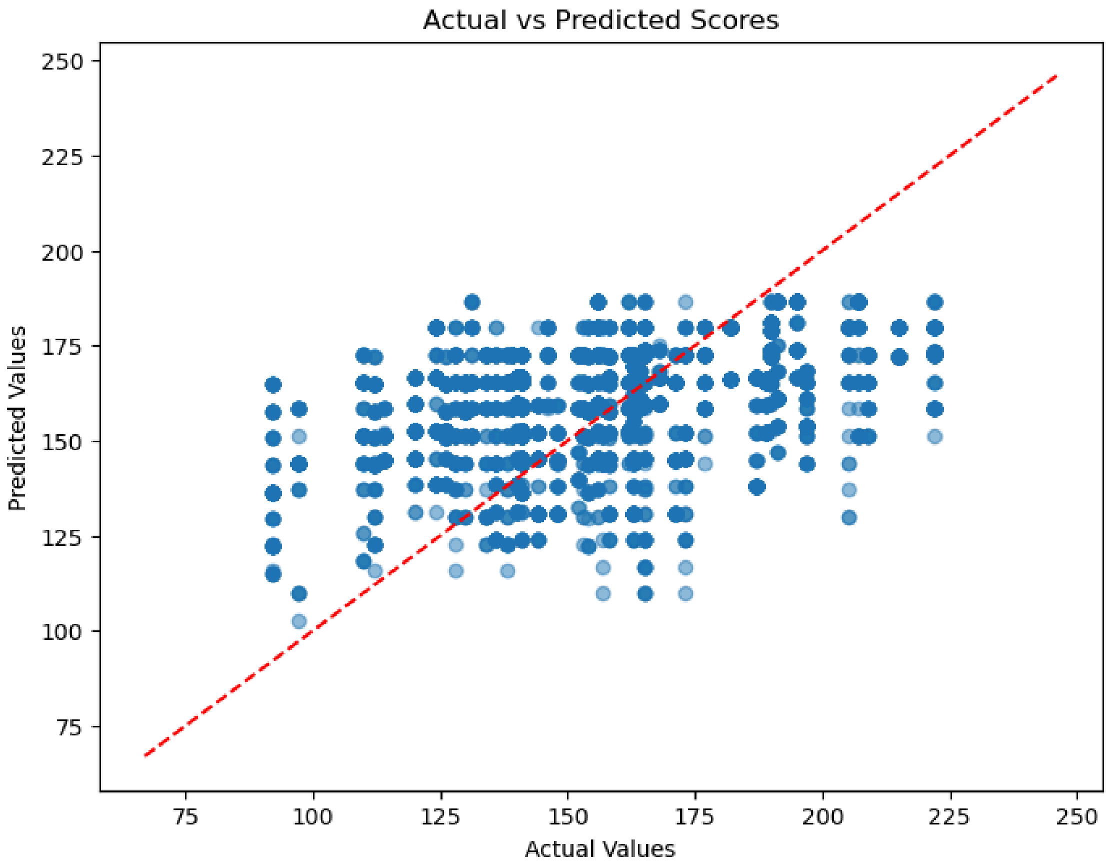
In [36]:

```
# Step 8: Residual Analysis
plt.figure(figsize=(8,6))
sns.histplot(y_test - y_pred_rfe, kde=True)
plt.title("Residual Analysis")
plt.xlabel("Residuals")
plt.ylabel("Density")
plt.show()
```



In [37]:

```
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred_rfe, alpha=0.5)
plt.plot([y.min(), y.max()], [y.min(), y.max()], '--', color='red')
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Actual vs Predicted Scores")
plt.show()
```



In [38]:

```
def predict_score(batting_team, bowling_team, overs, runs, wickets, runs_l
    temp_array = np.zeros(len(selected_features))  # Create an array of ze

    for i, feature in enumerate(selected_features):
        if feature == f'bat_team_{batting_team}':
            temp_array[i] = 1
        elif feature == f'bowl_team_{bowling_team}':
            temp_array[i] = 1
        elif feature == 'overs':
            temp_array[i] = overs
        elif feature == 'runs':
            temp_array[i] = runs
        elif feature == 'wickets':
            temp_array[i] = wickets
        elif feature == 'runs_last_5':
            temp_array[i] = runs_last_5
        elif feature == 'wickets_last_5':
            temp_array[i] = wickets_last_5

    # Convert to DataFrame to match the training data format
    input_df = pd.DataFrame([temp_array], columns=selected_features)

    # Predict the score
    predicted_score = model.predict(input_df)[0]

    return round(predicted_score)
```

#TEAMS FOR PREDICTION 1.Kolkata Knight Riders 2.Chennai Super Kings 3.Rajasthan Royals 4.Mumbai Indians 5.Kings XI Punjab 6.Royal Challengers Bangalore 7.Delhi Daredevils 8.Sunrisers Hyderabad

In [39]:

```
print("Selected Features for Prediction:", selected_features)
```

```
Selected Features for Prediction: Index(['wickets', 'bat_team_Sunrisers H
yderabad', 'bowl_team_Delhi Daredevils',
       'bowl_team_Rajasthan Royals', 'bowl_team_Sunrisers Hyderabad'],
      dtype='object')
```

In [40]:

```
print(df['bat_team'].unique())  # List of valid batting teams
print(df['bowl_team'].unique())  # List of valid bowling teams
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File J:\New folder\Lib\site-packages\pandas\core\indexes\base.py:3653, in Index.get_loc(self, key)
   3652 try:
-> 3653     return self._engine.get_loc(casted_key)
   3654 except KeyError as err:

File J:\New folder\Lib\site-packages\pandas\_libs\index.pyx:147, in pandas._libs.index.IndexEngine.get_loc()

File J:\New folder\Lib\site-packages\pandas\_libs\index.pyx:176, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:7080, in pandas._libs.hashtable.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:7088, in pandas._libs.has
```

In [41]:

```
predicted_score = predict_score(
    batting_team='Kolkata Knight Riders',
    bowling_team='Rajasthan Royals',
    overs=8,
    runs=25,
    wickets=1,
    runs_last_5=15,
    wickets_last_5=0,
    model=lr_rfe,
    selected_features=selected_features
)

print(f"Predicted First Innings Score: {predicted_score}")
```

```
Predicted First Innings Score: 168
```

In [42]:

```
predicted_score = predict_score(
    batting_team='Kings XI Punjab',
    bowling_team='Delhi Daredevils',
    overs=10,
    runs=78,
    wickets=2,
    runs_last_5=38,
    wickets_last_5=1,
    model=lr_rfe,
    selected_features=selected_features
)

print(f"Predicted First Innings Score: {predicted_score}")
```

```
Predicted First Innings Score: 165
```

In [43]:

```
predicted_score = predict_score(
    batting_team='Royal Challengers Bangalore',
    bowling_team='Mumbai Indians',
    overs=17,
    runs=145,
    wickets=5,
    runs_last_5=50,
    wickets_last_5=2,
    model=lr_rfe,
    selected_features=selected_features
)

print(f"Predicted First Innings Score: {predicted_score}")
```

```
Predicted First Innings Score: 151
```

In [44]:

```
predicted_score = predict_score(
    batting_team='Sunrisers Hyderabad',
    bowling_team='Rajasthan Royals',
    overs=6,
    runs=50,
    wickets=0,
    runs_last_5=50,
    wickets_last_5=0,
    model=lr_rfe,
    selected_features=selected_features
)

print(f"Predicted First Innings Score: {predicted_score}")
```

```
Predicted First Innings Score: 169
```