In [61]:
```
# Importing libraries
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as mse, acc
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
```

In [23]:
```
# Loading the dataset
df = pd.read_csv('ipl.csv')   #import file using pandas
```

In [24]:
```
df.columns
```

Out[24]:
```
Index(['mid', 'date', 'venue', 'bat_team', 'bowl_team', 'batsman', 'bowler',
       'runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5', 'striker',
       'non-striker', 'total'],
      dtype='object')
```

In [25]:
```
df.shape    # returns size of dataset
```

Out[25]:
```
(76014, 15)
```

In [26]:
```
df.dtypes   # returns data types of each column
```

Out[26]:
```
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 [27]:
```
df.head()   # returns top 5 rows of a data
```

Out[27]:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | SC Ganguly | P Kumar | 1 | 0 | 0.1 | |
| **1** | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 1 | 0 | 0.2 | |
| **2** | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 2 | 0 | 0.2 | |
| **3** | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 2 | 0 | 0.3 | |
| **4** | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 2 | 0 | 0.4 | |

In [28]:
```
df.columns
```

Out[28]:
```
Index(['mid', 'date', 'venue', 'bat_team', 'bowl_team', 'batsman', 'bowler',
       'runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5', 'striker',
       'non-striker', 'total'],
      dtype='object')
```

In [29]:
```
# Data Cleaning
# Remove unwanted column and keep only consistent teams
# remove first 5 overs of the match

columns_to_remove = ['mid', 'venue', 'batsman', 'bowler', 'striker', 'non-striker']

print('Before removing unwanted columns: {}'.format(df.shape))
df.drop(labels = columns_to_remove, axis = 1, inplace = True)
print('After removing unwanted columns: {}'.format(df.shape))
```

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

In [31]:
```
df.columns   # updated dataset columns
```

Out[31]:
```
Index(['date', 'bat_team', 'bowl_team', 'runs', 'wickets', 'overs',
       'runs_last_5', 'wickets_last_5', 'total'],
      dtype='object')
```

In [32]:
```
df.head()   # updated dataset
```

Out[32]:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Knight Riders | Royal Challengers Bangalore | 1 | 0 | 0.1 | 1 | 0 | 222 |
| **1** | 2008-04-18 | Kolkata Knight Riders | Royal Challengers Bangalore | 1 | 0 | 0.2 | 1 | 0 | 222 |
| **2** | 2008-04-18 | Kolkata Knight Riders | Royal Challengers Bangalore | 2 | 0 | 0.2 | 2 | 0 | 222 |
| **3** | 2008-04-18 | Kolkata Knight Riders | Royal Challengers Bangalore | 2 | 0 | 0.3 | 2 | 0 | 222 |
| **4** | 2008-04-18 | Kolkata Knight Riders | Royal Challengers Bangalore | 2 | 0 | 0.4 | 2 | 0 | 222 |

In [33]:
```
df.index
```

Out[33]:
```
RangeIndex(start=0, stop=76014, step=1)
```

In [34]:
```
df['bat_team'].unique()
```

Out[34]:
```
array(['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals',
       'Mumbai Indians', 'Deccan Chargers', 'Kings XI Punjab',
       'Royal Challengers Bangalore', 'Delhi Daredevils',
       'Kochi Tuskers Kerala', 'Pune Warriors', 'Sunrisers Hyderabad',
       'Rising Pune Supergiants', 'Gujarat Lions',
       'Rising Pune Supergiant'], dtype=object)
```

In [35]:
```
consistent_teams = ['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals
                    'Mumbai Indians', 'Kings XI Punjab', 'Royal Challengers Bangalore'
                    'Delhi Daredevils', 'Sunrisers Hyderabad']
```

In [36]:
```
# Keeping only consistent teams (Remove teams with less season played)
print('Before removing inconsistent teams: {}'.format(df.shape))
df = df[(df['bat_team'].isin(consistent_teams)) & (df['bowl_team'].isin(consistent_tea
print('After removing inconsistent teams: {}'.format(df.shape))
```

```
Before removing inconsistent teams: (76014, 9)
After removing inconsistent teams: (53811, 9)
```

In [37]:
```
df['bat_team'].unique()
```

Out[37]:
```
array(['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals',
       'Mumbai Indians', 'Kings XI Punjab', 'Royal Challengers Bangalore',
       'Delhi Daredevils', 'Sunrisers Hyderabad'], dtype=object)
```

In [38]:
```
# Removing the first 5 overs data in every match
print('Before removing first 5 overs data: {}'.format(df.shape))
df = df[df['overs']>=5.0]
print('After removing first 5 overs data: {}'.format(df.shape))
```

```
Before removing first 5 overs data: (53811, 9)
After removing first 5 overs data: (40108, 9)
```

In [40]:
```
# Converting the column 'date' from string into datetime object
print("Before converting 'date' column from string to datetime object: {}".format(type
df['date'] = df['date'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d'))
print("After converting 'date' column from string to datetime object: {}".format(type(
```

```
Before converting 'date' column from string to datetime object: <class 'str'>
After converting 'date' column from string to datetime object: <class 'pandas._libs.t
slibs.timestamps.Timestamp'>
```

In [42]:
```
# Selecting correlated features using Heatmap
# Get correlation of all the features of the dataset
corr_matrix = df.corr()
top_corr_features = corr_matrix.index

# Plotting the heatmap
plt.figure(figsize=(13,10))
g = sns.heatmap(data=df[top_corr_features].corr(), annot=True, cmap='RdYlGn')
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_7856\1468486487.py:3: FutureWarning: The
default value of numeric_only in DataFrame.corr is deprecated. In a future version, i
t will default to False. Select only valid columns or specify the value of numeric_on
ly to silence this warning.
  corr_matrix = df.corr()
```

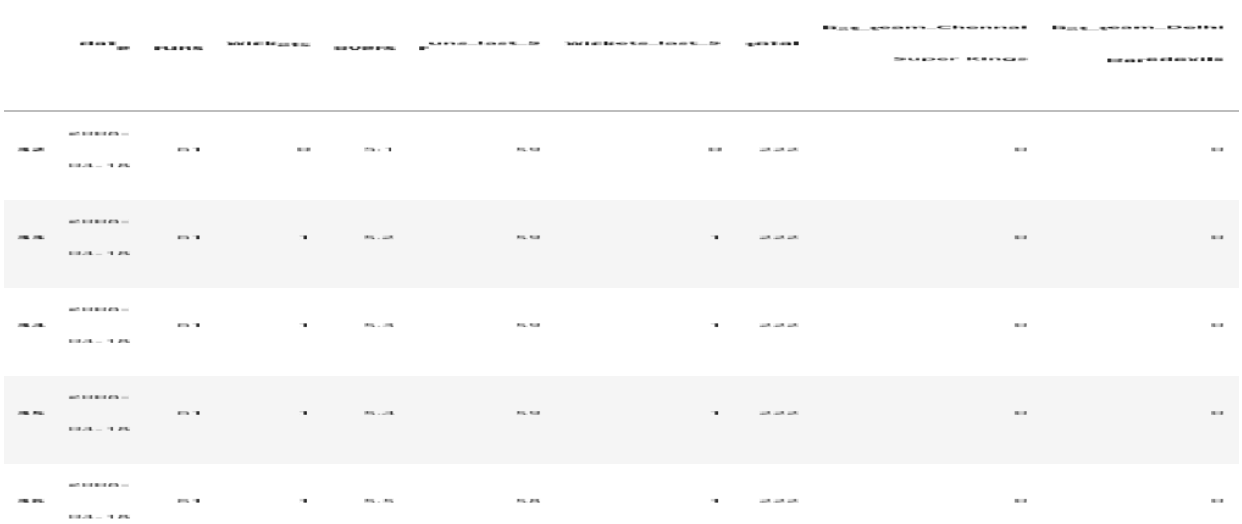

In [43]:
```
# Data Preprocessing
# Handling categorical features
# Splitting dataset into train and test set on the basis of date
```

```
# Converting categorical features using OneHotEncoding method
encoded_df = pd.get_dummies(data=df, columns=['bat_team', 'bowl_team'])
encoded_df.columns
```

Out[43]:
```
Index(['date', 'runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5',
       'total', 'bat_team_Chennai Super Kings', 'bat_team_Delhi Daredevils',
       'bat_team_Kings XI Punjab', 'bat_team_Kolkata Knight Riders',
       'bat_team_Mumbai Indians', 'bat_team_Rajasthan Royals',
       'bat_team_Royal Challengers Bangalore', 'bat_team_Sunrisers Hyderabad',
       'bowl_team_Chennai Super Kings', 'bowl_team_Delhi Daredevils',
       'bowl_team_Kings XI Punjab', 'bowl_team_Kolkata Knight Riders',
       'bowl_team_Mumbai Indians', 'bowl_team_Rajasthan Royals',
       'bowl_team_Royal Challengers Bangalore',
       'bowl_team_Sunrisers Hyderabad'],
      dtype='object')
```

In [44]:
```
encoded_df.head()
```

Out[44]:



5 rows × 23 columns

In [45]:
```
# Rearranging the columns
encoded_df = encoded_df[['date', 'bat_team_Chennai Super Kings', 'bat_team_Delhi Daredevils',
             'bat_team_Kolkata Knight Riders', 'bat_team_Mumbai Indians', 'bat_team_Rajasthan Royals',
             'bat_team_Royal Challengers Bangalore', 'bat_team_Sunrisers Hyderabad',
             'bowl_team_Chennai Super Kings', 'bowl_team_Delhi Daredevils', 'bowl_team_Kings XI Punjab',
             'bowl_team_Kolkata Knight Riders', 'bowl_team_Mumbai Indians', 'bowl_team_Rajasthan Royals',
             'bowl_team_Royal Challengers Bangalore', 'bowl_team_Sunrisers Hyderabad',
             'overs', 'runs', 'wickets', 'runs_last_5', 'wickets_last_5', 'total']]
```

In [46]:
```
# Splitting the data into train and test set
X_train = encoded_df.drop(labels='total', axis=1)[encoded_df['date'].dt.year <= 2016]
X_test = encoded_df.drop(labels='total', axis=1)[encoded_df['date'].dt.year >= 2017]

y_train = encoded_df[encoded_df['date'].dt.year <= 2016]['total'].values
y_test = encoded_df[encoded_df['date'].dt.year >= 2017]['total'].values

# Removing the 'date' column
X_train.drop(labels='date', axis=True, inplace=True)
X_test.drop(labels='date', axis=True, inplace=True)
```

```
print("Training set: {} and Test set: {}".format(X_train.shape, X_test.shape))
```

```
Training set: (37330, 21) and Test set: (2778, 21)
```

In [48]:
```
# Model Building
# •Linear Regression •Decision Tree Regression •Random Forest Regression •Adaptive Boo

## Linear Regression Model
linear_regressor = LinearRegression()
linear_regressor.fit(X_train,y_train)
```

Out[48]:
```
▾ LinearRegression

LinearRegression()
```

In [49]:
```
# Predicting results
y_pred_lr = linear_regressor.predict(X_test)
```

In [52]:
```
# Linear Regression - Model Evaluation
print("---- Linear Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test, y_pred_lr)))
print("Mean Squared Error (MSE): {}".format(mse(y_test, y_pred_lr)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test, y_pred_lr))))
```

```
---- Linear Regression - Model Evaluation ----
Mean Absolute Error (MAE): 12.118617546193294
Mean Squared Error (MSE): 251.00792310417438
Root Mean Squared Error (RMSE): 15.843229566732106
```

In [54]:
```
## Decision Tree Regression Model
decision_regressor = DecisionTreeRegressor()
decision_regressor.fit(X_train,y_train)
```

Out[54]:
```
▾ DecisionTreeRegressor

DecisionTreeRegressor()
```

In [55]:
```
# Predicting results
y_pred_dt = decision_regressor.predict(X_test)
```

In [56]:
```
# Decision Tree Regression - Model Evaluation
print("---- Decision Tree Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test, y_pred_dt)))
print("Mean Squared Error (MSE): {}".format(mse(y_test, y_pred_dt)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test, y_pred_dt))))
```

```
---- Decision Tree Regression - Model Evaluation ----
Mean Absolute Error (MAE): 17.27537796976242
Mean Squared Error (MSE): 544.1004319654428
Root Mean Squared Error (RMSE): 23.32596047251737
```

In [58]:
```
## Random Forest Regression Model
random_regressor = RandomForestRegressor()
random_regressor.fit(X_train,y_train)
```

Out[58]:
```
RandomForestRegressor
RandomForestRegressor()
```

In [59]:
```
# Predicting results
y_pred_rf = random_regressor.predict(X_test)
```

In [60]:
```
# Random Forest Regression - Model Evaluation
print("---- Random Forest Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test, y_pred_rf)))
print("Mean Squared Error (MSE): {}".format(mse(y_test, y_pred_rf)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test, y_pred_rf))))
```

```
---- Random Forest Regression - Model Evaluation ----
Mean Absolute Error (MAE): 13.740630112448148
Mean Squared Error (MSE): 329.3573730235518
Root Mean Squared Error (RMSE): 18.148205779733484
```

In [62]:
```
## Linear Regression model performs best as compared to other two, use this model and

## AdaBoost Model using Linear Regression as the base learner
adb_regressor = AdaBoostRegressor(base_estimator=linear_regressor, n_estimators=100)
adb_regressor.fit(X_train, y_train)
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\ensemble\_base.py:156: FutureWarni
ng: `base_estimator` was renamed to `estimator` in version 1.2 and will be removed in
1.4.
  warnings.warn(
```

Out[62]:
```
AdaBoostRegressor
base_estimator: LinearRegression
LinearRegression
```

In [63]:
```
# Predicting results
y_pred_adb = adb_regressor.predict(X_test)
```

In [64]:
```
# AdaBoost Regression - Model Evaluation
print("---- AdaBoost Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test, y_pred_adb)))
print("Mean Squared Error (MSE): {}".format(mse(y_test, y_pred_adb)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test, y_pred_adb))))
```

```
---- AdaBoost Regression - Model Evaluation ----
Mean Absolute Error (MAE): 12.038415817178553
Mean Squared Error (MSE): 245.2055531391312
Root Mean Squared Error (RMSE): 15.659040620010257
```

In [65]:
```
## Using AdaBoost did not reduce the error to a significant level. Hence, use simple l
```

In [66]:
```
def predict_score(batting_team='Chennai Super Kings', bowling_team='Mumbai Indians', c
  temp_array = list()

  # Batting Team
  if batting_team == 'Chennai Super Kings':
    temp_array = temp_array + [1,0,0,0,0,0,0,0]
```

```
  elif batting_team == 'Delhi Daredevils':
    temp_array = temp_array + [0,1,0,0,0,0,0,0]
  elif batting_team == 'Kings XI Punjab':
    temp_array = temp_array + [0,0,1,0,0,0,0,0]
  elif batting_team == 'Kolkata Knight Riders':
    temp_array = temp_array + [0,0,0,1,0,0,0,0]
  elif batting_team == 'Mumbai Indians':
    temp_array = temp_array + [0,0,0,0,1,0,0,0]
  elif batting_team == 'Rajasthan Royals':
    temp_array = temp_array + [0,0,0,0,0,1,0,0]
  elif batting_team == 'Royal Challengers Bangalore':
    temp_array = temp_array + [0,0,0,0,0,0,1,0]
  elif batting_team == 'Sunrisers Hyderabad':
    temp_array = temp_array + [0,0,0,0,0,0,0,1]

  # Bowling Team
  if bowling_team == 'Chennai Super Kings':
    temp_array = temp_array + [1,0,0,0,0,0,0,0]
  elif bowling_team == 'Delhi Daredevils':
    temp_array = temp_array + [0,1,0,0,0,0,0,0]
  elif bowling_team == 'Kings XI Punjab':
    temp_array = temp_array + [0,0,1,0,0,0,0,0]
  elif bowling_team == 'Kolkata Knight Riders':
    temp_array = temp_array + [0,0,0,1,0,0,0,0]
  elif bowling_team == 'Mumbai Indians':
    temp_array = temp_array + [0,0,0,0,1,0,0,0]
  elif bowling_team == 'Rajasthan Royals':
    temp_array = temp_array + [0,0,0,0,0,1,0,0]
  elif bowling_team == 'Royal Challengers Bangalore':
    temp_array = temp_array + [0,0,0,0,0,0,1,0]
  elif bowling_team == 'Sunrisers Hyderabad':
    temp_array = temp_array + [0,0,0,0,0,0,0,1]

  # Overs, Runs, Wickets, Runs_in_prev_5, Wickets_in_prev_5
  temp_array = temp_array + [overs, runs, wickets, runs_in_prev_5, wickets_in_prev_5]

  # Converting into numpy array
  temp_array = np.array([temp_array])

  # Prediction
  return int(linear_regressor.predict(temp_array)[0])
```

In [67]:

```
# Prediction 1
# • IPL : Season 11, Match number: 13
# • First Innings final score: 200/9

final_score = predict_score(batting_team='Kolkata Knight Riders', bowling_team='Delhi
print("The final predicted score (range): {} to {}".format(final_score-10, final_score
```

```
The final predicted score (range): 159 to 174
C:\Users\admin\anaconda3\lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [68]:

```
# Prediction 2
# • IPL : Season 11, Match number: 39
# • First Innings final score: 146/10
```

```
final_score = predict_score(batting_team='Sunrisers Hyderabad', bowling_team='Royal Ch
print("The final predicted score (range): {} to {}".format(final_score-10, final_score
```

```
The final predicted score (range): 138 to 153
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [69]:
```
# Prediction 3
# • IPL : Season 11, Match number: 50
# • First Innings final score: 186/8

final_score = predict_score(batting_team='Mumbai Indians', bowling_team='Kings XI Punj
print("The final predicted score (range): {} to {}".format(final_score-10, final_score
```

```
The final predicted score (range): 180 to 195
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [70]:
```
# Prediction 4
# • IPL : Season 12, Match number: 9
# • First Innings final score: 176/7

final_score = predict_score(batting_team='Mumbai Indians', bowling_team='Kings XI Punj
print("The final predicted score (range): {} to {}".format(final_score-10, final_score
```

```
The final predicted score (range): 179 to 194
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [71]:
```
# Prediction 5
# • IPL : Season 12, Match number: 25
# • First Innings final score: 151/7

final_score = predict_score(batting_team='Rajasthan Royals', bowling_team='Chennai Sup
print("The final predicted score (range): {} to {}".format(final_score-10, final_score
```

```
The final predicted score (range): 128 to 143
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [72]:
```
# Prediction 6
# • IPL : Season 12, Match number: 30
# • First Innings final score: 155/7

final_score = predict_score(batting_team='Delhi Daredevils', bowling_team='Sunrisers H
print("The final predicted score (range): {} to {}".format(final_score-10, final_score
```

```
The final predicted score (range): 157 to 172
```

```
C:\Users\admin\anaconda3\lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [73]:
```
# Prediction 7
# • IPL : Season 12, Match number: 59
# • First Innings final score: 147/9
```

```
final_score = predict_score(batting_team='Delhi Daredevils', bowling_team='Chennai Sup
print("The final predicted score (range): {} to {}".format(final_score-10, final_score
```

```
The final predicted score (range): 137 to 152
C:\Users\admin\anaconda3\lib\site-packages\sklearn\base.py:464: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [ ]: