ipl-score

August 1, 2024

[57]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from seaborn import heatmap
```

[58]:
```
df=pd.read_csv('ipl_data.csv')
```

[59]:
```
df.head(10)
```

[59]:

	mid	date	venue	bat_team	bowl_team	batsman	bowler	runs	wickets	overs	runs_last_5	wickets_last_5	striker	non-striker	total
0	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	SC Ganguly	P Kumar	1	0	0.1	1	0	0	0	222
1	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	1	0	0.2	1	0	0	0	222
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					
5	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.5					
6	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	3	0	0.6					
7	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	Z Khan	3	0	1.1					
8	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	Z Khan	7	0	1.2					
9	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	Z Khan	11	0	1.3					

```
2            2              0        0            0    222
3            2              0        0            0    222
4            2              0        0            0    222
5            2              0        0            0    222
6            3              0        0            0    222
7            3              0        0            0    222
8            7              0        4            0    222
9           11              0        8            0    222
```

[60]:
```
df['bat_team'].unique()
```

[60]:
```
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)
```

[61]:
```
df.isnull().sum()
```

[61]:
```
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
```

[62]:
```
df.duplicated().sum()
```

[62]:
```
0
```

[63]:
```
df.describe()
```

[63]:
```
                mid          runs       wickets         overs   runs_last_5  \
count  76014.000000  76014.000000  76014.000000  76014.000000  76014.000000
mean     308.627740     74.889349      2.415844      9.783068     33.216434
std      178.156878     48.823327      2.015207      5.772587     14.914174
```

```
min        1.000000      0.000000      0.000000      0.000000      0.000000
25%      154.000000     34.000000      1.000000      4.600000     24.000000
50%      308.000000     70.000000      2.000000      9.600000     34.000000
75%      463.000000    111.000000      4.000000     14.600000     43.000000
max      617.000000    263.000000     10.000000     19.600000    113.000000

       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
```

[64]:
```
df.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
```

[65]:
```
df.shape
```

[65]: (76014, 15)

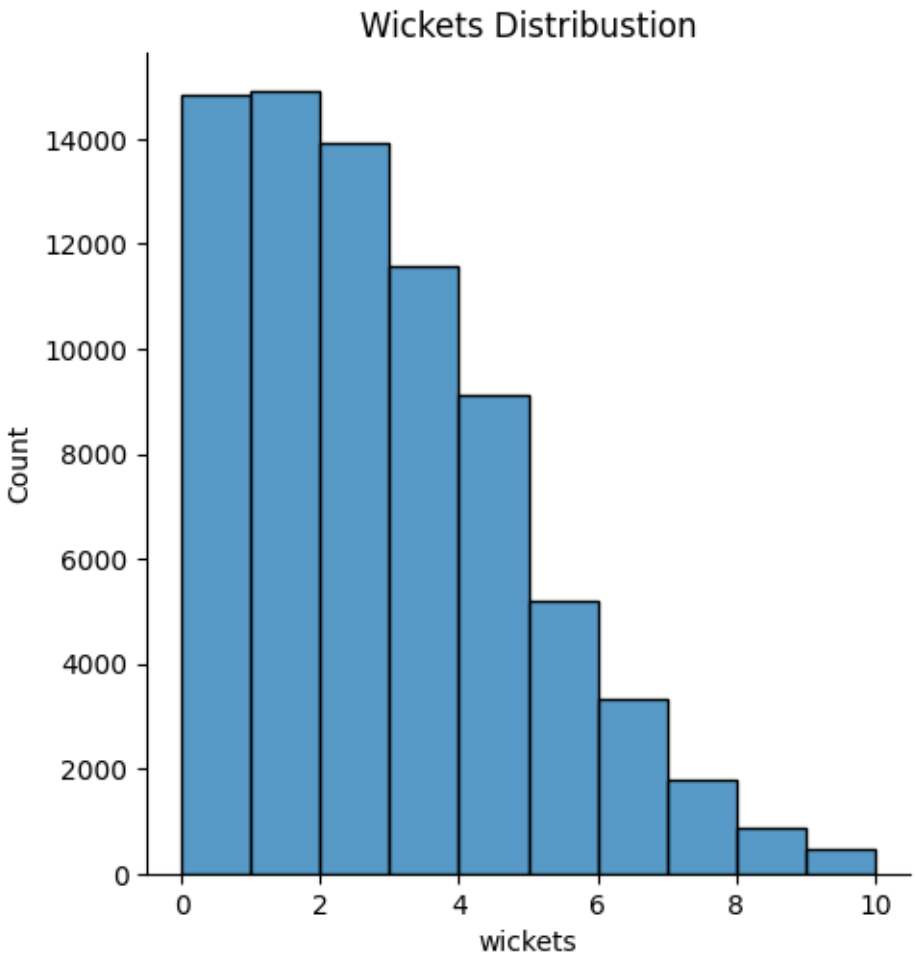
[66]:
```
df.nunique()
```

[66]:
```
mid               617
date              442
venue              35
bat_team           14
bowl_team          14
batsman           411
bowler            329
runs              252
wickets            11
overs             140
runs_last_5       102
wickets_last_5      8
striker           155
non-striker        88
total             138
dtype: int64
```

wicket distribustribustion coundt of run per wickets

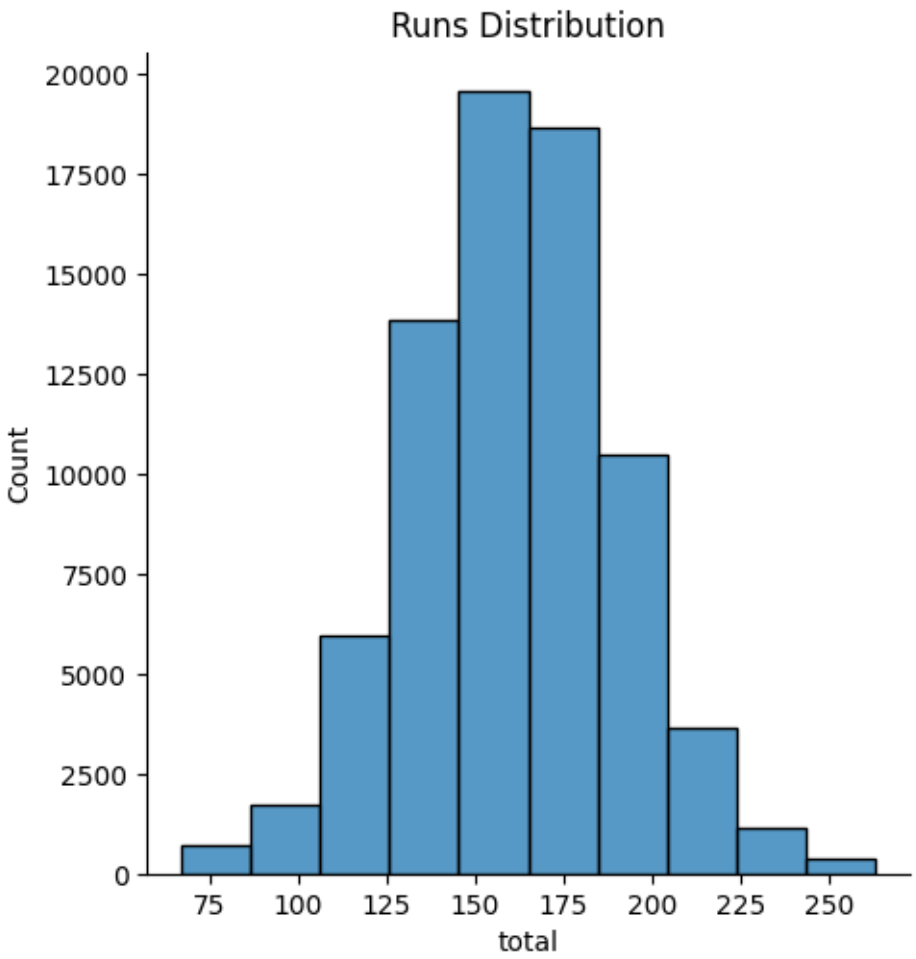
[67]:
```
sns.displot(df['wickets'],kde=False,bins=10)
plt.title('Wickets Distribustion')
```

[67]:
```
Text(0.5, 1.0, 'Wickets Distribustion')
```



[68]:
```
sns.displot(df['total'],kde=False,bins=10)
plt.title('Runs Distribution')
```

[68]:
```
Text(0.5, 1.0, 'Runs Distribution')
```



[69]:
```
df.columns
```

[69]:
```
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')
```

[70]:
```
irrelevant=['mid', 'date', 'venue','batsman', 'bowler','striker',
       'non-striker']
print("Before removing col",df.shape)

df=df.drop(irrelevant,axis=1)
print("After removing col",df.shape)
```

```
Before removing col (76014, 15)
After removing col (76014, 8)
```

[71]:
```
df.columns
```

[71]:
```
Index(['bat_team', 'bowl_team', 'runs', 'wickets', 'overs', 'runs_last_5',
       'wickets_last_5', 'total'],
      dtype='object')
```

[ ]:
```

```

[72]:
```
# keeping only consistent team

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

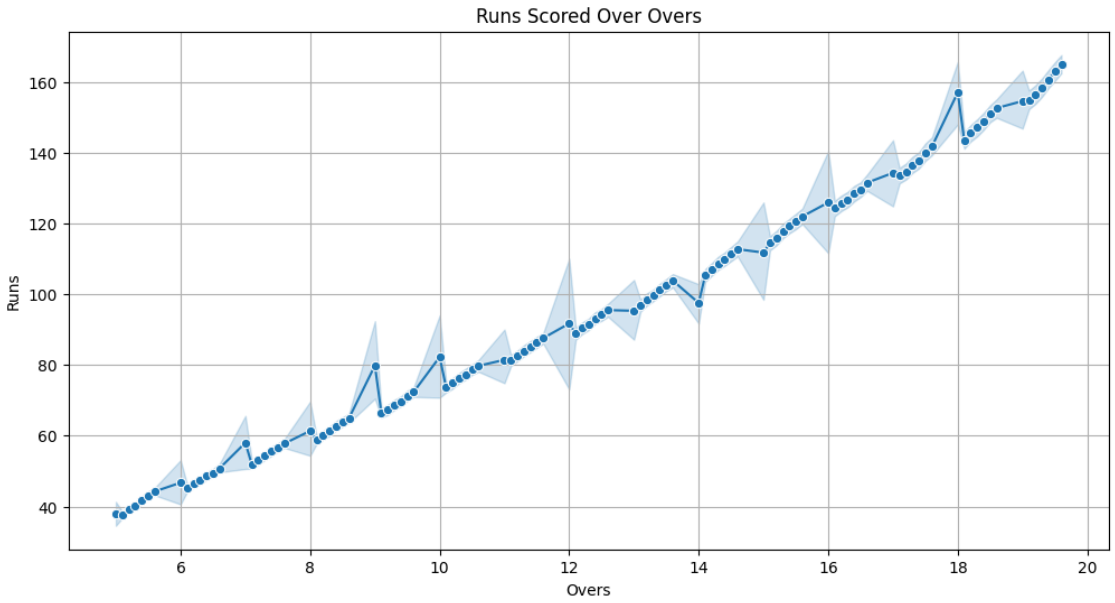
[73]:
```
df.head(20)
```

[73]:

	bat_team	bowl_team	runs	wickets	overs	runs_last_5	wickets_last_5	total
0	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.1	1	0	222
1	Kolkata Knight Riders	Royal Challengers Bangalore	1	0	0.2	1	0	222
2	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.2	2	0	222
3	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.3	2	0	222
4	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.4	2	0	222
5	Kolkata Knight Riders	Royal Challengers Bangalore	2	0	0.5	2	0	222
6	Kolkata Knight Riders	Royal Challengers Bangalore	3	0	0.6	3	0	222
7	Kolkata Knight Riders	Royal Challengers Bangalore	3	0	1.1	3	0	222
8	Kolkata Knight Riders	Royal Challengers Bangalore	7	0	1.2	7	0	222
9	Kolkata Knight Riders	Royal Challengers Bangalore	11	0	1.3	11	0	222
10	Kolkata Knight Riders	Royal Challengers Bangalore	17	0	1.4	17	0	222
11	Kolkata Knight Riders	Royal Challengers Bangalore	21	0	1.5	21	0	222
12	Kolkata Knight Riders	Royal Challengers Bangalore	21	0	1.6	21	0	222
13	Kolkata Knight Riders	Royal Challengers Bangalore	21	0	2.1	21	0	222
14	Kolkata Knight Riders	Royal Challengers Bangalore	21	0	2.2	21	0	222
15	Kolkata Knight Riders	Royal Challengers Bangalore	22	0	2.3	22	0	222
16	Kolkata Knight Riders	Royal Challengers Bangalore	26	0	2.4	26	0	222
17	Kolkata Knight Riders	Royal Challengers Bangalore	27	0	2.5	27	0	222
18	Kolkata Knight Riders	Royal Challengers Bangalore	27	0	2.6	27	0	222
19	Kolkata Knight Riders	Royal Challengers Bangalore	32	0	3.0	32	0	222

8	7	0	222
9	11	0	222
10	17	0	222
11	21	0	222
12	21	0	222
13	21	0	222
14	21	0	222
15	22	0	222
16	26	0	222
17	27	0	222
18	27	0	222
19	32	0	222

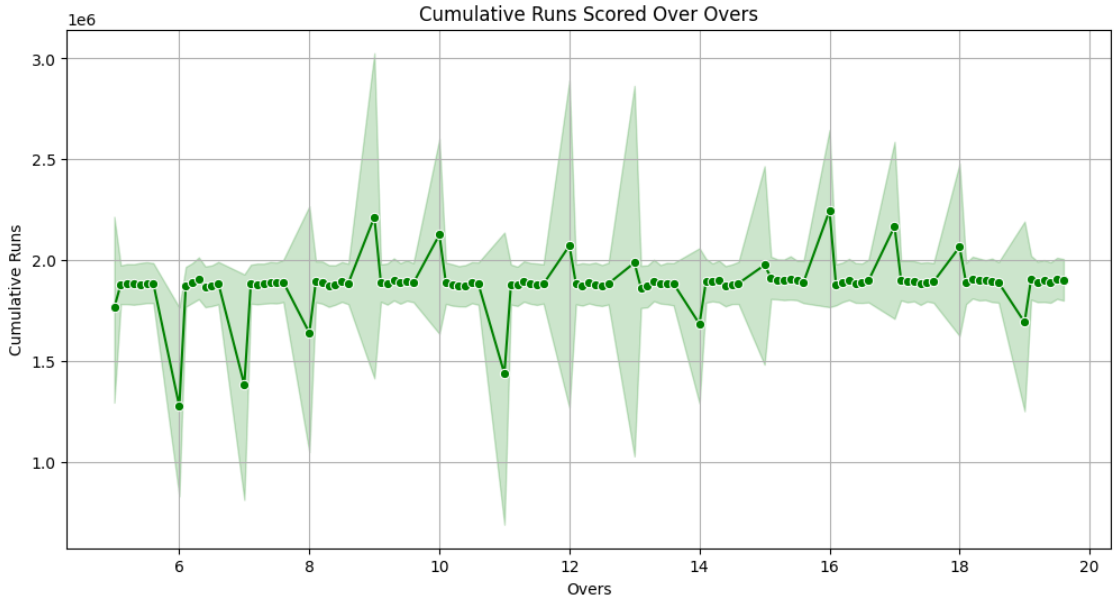
[113]:
```
plt.figure(figsize=(12, 6))
sns.lineplot(x='overs', y='runs', data=df, marker='o')
plt.title('Runs Scored Over Overs')
plt.xlabel('Overs')
plt.ylabel('Runs')
plt.grid(True)
plt.show()
```



[112]:
```
plt.figure(figsize=(12, 6))
sns.lineplot(x='overs', y='runs', data=df, marker='o', label='Runs Scored')
sns.lineplot(x='overs', y='runs_last_5', data=df, marker='o', color='orange',␣
↪label='Runs in Last 5 Deliveries')
plt.title('Runs Scored vs Runs in Last 5 Deliveries')
plt.xlabel('Overs')
```

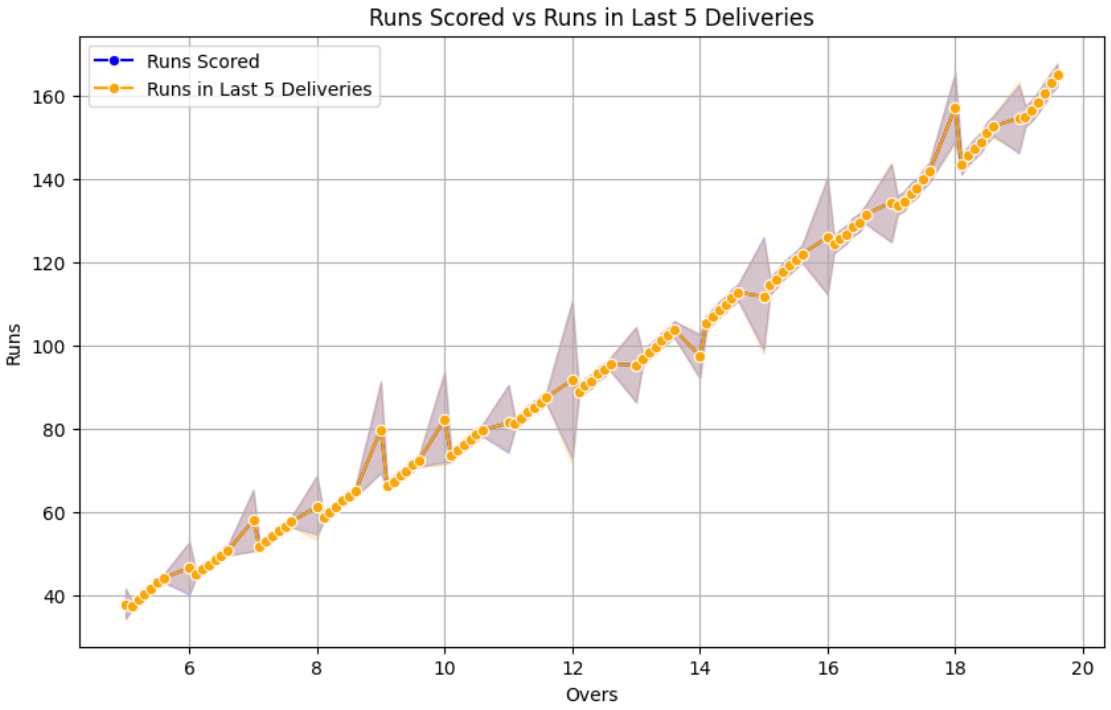
```
plt.ylabel('Runs')
plt.legend()
plt.grid(True)
plt.show()
```

[114]:
```
df['cumulative_runs'] = df['runs'].cumsum()
plt.figure(figsize=(12, 6))
sns.lineplot(x='overs', y='cumulative_runs', data=df, marker='o', color='green')
plt.title('Cumulative Runs Scored Over Overs')
plt.xlabel('Overs')
plt.ylabel('Cumulative Runs')
plt.grid(True)
plt.show()
```



[115]:
```
# Adding runs_last_5 for demonstration
df['runs_last_5'] = df['runs']  # Assuming same for simplicity

# Plot: Runs Scored vs Runs in Last 5 Deliveries
plt.figure(figsize=(10, 6))
sns.lineplot(x='overs', y='runs', data=df, marker='o', label='Runs Scored',␣
 ↪color='blue')
sns.lineplot(x='overs', y='runs_last_5', data=df, marker='o', label='Runs in␣
 ↪Last 5 Deliveries', color='orange')
plt.title('Runs Scored vs Runs in Last 5 Deliveries')
plt.xlabel('Overs')
plt.ylabel('Runs')
plt.legend()
plt.grid(True)
plt.show()
```



[74]:
```
print("Before removing inconsistent team : ",df.shape)

df=df[(df['bat_team'].isin(const_teams)) & (df['bowl_team'].isin(const_teams))]

print("after removing inconsistent team : ",df.shape)
```

```
Before removing inconsistent team :  (76014, 8)
after removing inconsistent team :  (53811, 8)
```

[75]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 53811 entries, 0 to 75888
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   bat_team        53811 non-null  object 
 1   bowl_team       53811 non-null  object 
 2   runs            53811 non-null  int64  
 3   wickets         53811 non-null  int64  
 4   overs           53811 non-null  float64
 5   runs_last_5     53811 non-null  int64  
 6   wickets_last_5  53811 non-null  int64  
 7   total           53811 non-null  int64  
```

```
dtypes: float64(1), int64(5), object(2)
memory usage: 3.7+ MB
```

[76]:
```
df.head()
```

[76]:
```
                bat_team                    bowl_team  runs  wickets  overs  \
0  Kolkata Knight Riders  Royal Challengers Bangalore     1        0    0.1
1  Kolkata Knight Riders  Royal Challengers Bangalore     1        0    0.2
2  Kolkata Knight Riders  Royal Challengers Bangalore     2        0    0.2
3  Kolkata Knight Riders  Royal Challengers Bangalore     2        0    0.3
4  Kolkata Knight Riders  Royal Challengers Bangalore     2        0    0.4

   runs_last_5  wickets_last_5  total
0            1               0    222
1            1               0    222
2            2               0    222
3            2               0    222
4            2               0    222
```

[77]:
```
# Remove first 5 over

print('Before removing over',df.shape)

df=df[df['overs']>=5.0]

print('After removing over',df.shape)
```

```
Before removing over (53811, 8)
After removing over (40108, 8)
```

[78]:
```
num_col = df.select_dtypes(include=['float64', 'int64'])

# Compute correlation matrix
corr_matrix = num_col.corr()

# Plot heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

Perform Label Encoding

[79]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

[80]:
```
le=LabelEncoder()
```

[81]:
```
for col in ['bat_team','bowl_team']:
  df[col]=le.fit_transform(df[col])
```

[82]:
```
df.head()
```

[82]:

	bat_team	bowl_team	runs	wickets	overs	runs_last_5	wickets_last_5	\
32	3	6	61	0	5.1	59	0	
33	3	6	61	1	5.2	59	1	
34	3	6	61	1	5.3	59	1	
35	3	6	61	1	5.4	59	1	
36	3	6	61	1	5.5	58	1	

```
    total
32    222
33    222
34    222
35    222
36    222
```

[83]:
```
from sklearn.compose import ColumnTransformer
```

[84]:
```
columnTransformer = ColumnTransformer([('encoder',
                                        OneHotEncoder(),
                                        [0, 1])],
                                      remainder='passthrough')
```

[85]:
```
ipl_df = np.array(columnTransformer.fit_transform(df))
```

[86]:
```
cols = ['batting_team_Chennai Super Kings', 'batting_team_Delhi Daredevils',␣
 ↪'batting_team_Kings XI Punjab',
             'batting_team_Kolkata Knight Riders', 'batting_team_Mumbai␣
 ↪Indians', 'batting_team_Rajasthan Royals',
             'batting_team_Royal Challengers Bangalore',␣
 ↪'batting_team_Sunrisers Hyderabad',
             'bowling_team_Chennai Super Kings', 'bowling_team_Delhi␣
 ↪Daredevils', 'bowling_team_Kings XI Punjab',
             'bowling_team_Kolkata Knight Riders', 'bowling_team_Mumbai␣
 ↪Indians', 'bowling_team_Rajasthan Royals',
             'bowling_team_Royal Challengers Bangalore',␣
 ↪'bowling_team_Sunrisers Hyderabad', 'runs', 'wickets', 'overs',
       'runs_last_5', 'wickets_last_5', 'total']
df = pd.DataFrame(ipl_df, columns=cols)
```

[87]:
```
print(df.shape)
```

```
(40108, 22)
```

[88]:
```
df = pd.DataFrame(df, columns=cols)
```

[89]:
```
x=df.drop('total',axis=1)
y=df['total']
```

[90]:
```
from sklearn.model_selection import train_test_split
```

[91]:
```
X_TRAIN, X_TEST, Y_TRAIN, Y_TEST=train_test_split(x,y,test_size=0.
 ↪20,random_state=42)
```

[92]:
```
print(X_TRAIN.shape)
print(X_TEST.shape)
print(Y_TRAIN.shape)
print(Y_TEST.shape)
```

```
(32086, 21)
(8022, 21)
(32086,)
(8022,)
```

[93]:
```
from sklearn.metrics import accuracy_score
```

[94]:
```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
```

[95]:
```
from sklearn.metrics import mean_squared_error as mse, r2_score as r2,␣
 ↪mean_absolute_error as mae
```

[96]:
```
LR_model=LinearRegression()
DST_model=DecisionTreeRegressor()
RF_model=RandomForestRegressor()
SVR_model=SVR()
SG_model=XGBRegressor()
```

Linear Regrssion

[97]:
```
LR_model.fit(X_TRAIN,Y_TRAIN)
```

[97]:
```
LinearRegression()
```

[98]:
```
Y_Pred_LR=LR_model.predict(X_TEST)
```

[99]:
```
LR_rmse=np.sqrt(mse(Y_TEST,Y_Pred_LR))
LR_r2=r2(Y_TEST,Y_Pred_LR)
```

[100]:
```
y_train_pred=LR_model.predict(X_TRAIN)
y_test_pred=LR_model.predict(X_TEST)
```

[101]:
```
train_r2_percentage = r2(Y_TRAIN, y_train_pred) * 100
test_r2_percentage = r2(Y_TEST, y_test_pred) * 100
```

[102]:
```
print('Trainning score',train_r2_percentage)
print('Testing score',test_r2_percentage)
```

```
Trainning score 65.69923425260747
Testing score 66.78598203684052
```

[103]:
```
print("LinearRegression RMSE:", LR_rmse)
print("LinearRegression r2:", LR_r2)
```

```
LinearRegression RMSE: 17.23030069769983
LinearRegression r2: 0.6678598203684052
```

Decision Tree

[104]:
```
DST_model.fit(X_TRAIN,Y_TRAIN)
```

[104]:
```
DecisionTreeRegressor()
```

[105]:
```
Y_Pred_DST=DST_model.predict(X_TEST)
```

[106]:
```
train_score_tree = str(DST_model.score(X_TRAIN, Y_TRAIN) * 100)
test_score_tree = str(DST_model.score(X_TEST, Y_TEST) * 100)
print(f'Train Score : {train_score_tree[:5]}%\nTest Score : {test_score_tree[:
 ↪5]}%')
```

```
Train Score : 99.98%
Test Score : 85.52%
```

Random Forest

[107]:
```
RF_model.fit(X_TRAIN,Y_TRAIN)
```

[107]:
```
RandomForestRegressor()
```

[108]:
```
Y_Pred_RF=RF_model.predict(X_TEST)
```

[109]:
```
train_score_RF = str(RF_model.score(X_TRAIN, Y_TRAIN) * 100)
test_score_RF = str(RF_model.score(X_TEST, Y_TEST) * 100)
print('Train Score :',train_score_RF)
print('Test Score',test_score_RF)
```

```
Train Score : 99.04072899010858
Test Score 93.56599512396242
```

[110]:
```
from sklearn.metrics import classification_report
```

[111]:
```
print(classification_report(Y_TEST,Y_Pred_RF))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[111], line 1
----> 1 print(classification_report(Y_TEST,Y_Pred_RF))
```

```
File␣
↪~\AppData\Roaming\Python\Python312\site-packages\sklearn\utils\_param_validat␣on.
↪py:213, in validate_params.<locals>.decorator.<locals>.wrapper(*args, **kwarg␣)
    207 try:
    208     with config_context(
    209         skip_parameter_validation=(
    210             prefer_skip_nested_validation or global_skip_validation
    211         )
    212     ):
--> 213         return func(*args, **kwargs)
    214 except InvalidParameterError as e:
    215     # When the function is just a wrapper around an estimator, we allow
    216     # the function to delegate validation to the estimator, but we␣
↪replace
    217     # the name of the estimator by the name of the function in the erro␣
    218     # message to avoid confusion.
    219     msg = re.sub(
    220         r"parameter of \w+ must be",
    221         f"parameter of {func.__qualname__} must be",
    222         str(e),
    223     )

File␣
↪~\AppData\Roaming\Python\Python312\site-packages\sklearn\metrics\_classificat␣on.
↪py:2612, in classification_report(y_true, y_pred, labels, target_names,␣
↪sample_weight, digits, output_dict, zero_division)
   2477 @validate_params(
   2478     {
   2479         "y_true": ["array-like", "sparse matrix"],
   (…)
   2503     zero_division="warn",
   2504 ):
   2505     """Build a text report showing the main classification metrics.
   2506
   2507     Read more in the :ref:`User Guide <classification_report>`.
   (…)
   2609     <BLANKLINE>
   2610     """
-> 2612     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
   2614     if labels is None:
   2615         labels = unique_labels(y_true, y_pred)

File␣
↪~\AppData\Roaming\Python\Python312\site-packages\sklearn\metrics\_classificat␣on.
↪py:108, in _check_targets(y_true, y_pred)
    105     y_type = {"multiclass"}
    107 if len(y_type) > 1:
--> 108     raise ValueError(
```

```
    109         "Classification metrics can't handle a mix of {0} and {1}␣
 ↪targets".format(
    110             type_true, type_pred
    111         )
    112     )
    114 # We can't have more than one value on y_type => The set is no more␣
 ↪needed
    115 y_type = y_type.pop()

ValueError: Classification metrics can't handle a mix of multiclass and␣
 ↪continuous targets
```

[ ]:
```
import numpy as np

def score_predict(batting_team, bowling_team, runs, wickets, overs,␣
 ↪runs_last_5, wickets_last_5, model):
    prediction_array = []

    # One-hot encoding for batting team
    teams = ['Chennai Super Kings', 'Delhi Daredevils', 'Kings XI Punjab',␣
 ↪'Kolkata Knight Riders',
             'Mumbai Indians', 'Rajasthan Royals', 'Royal Challengers␣
 ↪Bangalore', 'Sunrisers Hyderabad']

    for team in teams:
        if batting_team == team:
            prediction_array.append(1)
        else:
            prediction_array.append(0)

    # One-hot encoding for bowling team
    for team in teams:
        if bowling_team == team:
            prediction_array.append(1)
        else:
            prediction_array.append(0)

    # Append match-specific features
    prediction_array += [runs, wickets, overs, runs_last_5, wickets_last_5]

    # Convert to numpy array and reshape for prediction
    prediction_array = np.array([prediction_array])

    # Make the prediction
    pred = RF_model.predict(prediction_array)
```

```
    return int(round(pred[0]))
```

[ ]:
```
batting_team = 'Delhi Daredevils'
bowling_team = 'Chennai Super Kings'
score = score_predict(batting_team, bowling_team, overs=10.2, runs=68,␣
 ↪wickets=3, runs_last_5=29, wickets_last_5=1, model=RF_model)
print(f'Predicted Score : {score} || Actual Score : 147')
```

```
  Cell In[1], line 4
    print(f'Predicted Score : {score})
          ^

SyntaxError: unterminated f-string literal (detected at line 4)
```

[ ]:
```

```