```
In [1]: import pandas as pd

        # Load the hitting and pitching data
        hitting = pd.read_csv('hittingstats.csv')
        pitching = pd.read_csv('pitchingstats.csv')

        # Quick look at the data
        print(hitting.head())
        print(pitching.head())
```

```
                      Tm  #Bat  BatAge   R/G    G    PA    AB    R     H   2B  \
0  Arizona Diamondbacks    51    28.6  5.47  162  6284  5522  886  1452  271
1        Atlanta Braves    55    29.3  4.35  162  6075  5481  704  1333  273
2     Baltimore Orioles    60    26.9  4.85  162  6176  5567  786  1391  262
3        Boston Red Sox    57    27.3  4.64  162  6192  5577  751  1404  311
4          Chicago Cubs    56    27.8  4.54  162  6116  5441  736  1318  253

   ...    SLG    OPS  OPS+    TB  GDP  HBP  SH  SF  IBB   LOB
0  ...  0.440  0.777   115  2430  112   84  34  66   17  1111
1  ...  0.415  0.724   100  2275  119   58   9  39   11  1088
2  ...  0.435  0.751   118  2424   71   64   6  45   14  1108
3  ...  0.423  0.741   104  2357  110   73   7  40   30  1116
4  ...  0.393  0.710   100  2139   89   70  17  41   19  1105

[5 rows x 29 columns]
                      Tm  #P  PAge  RA/G   W   L   W-L%   ERA    G   GS  ...  \
0  Arizona Diamondbacks  33  28.6  4.86  89  73  0.549  4.62  162  162  ...
1        Atlanta Braves  30  31.5  3.75  89  73  0.549  3.49  162  162  ...
2     Baltimore Orioles  34  29.6  4.31  91  71  0.562  3.94  162  162  ...
3        Boston Red Sox  35  29.5  4.61  81  81  0.500  4.04  162  162  ...
4          Chicago Cubs  36  29.2  4.13  83  79  0.512  3.78  162  162  ...

     BF  ERA+   FIP   WHIP   H9  HR9  BB9  SO9  SO/W   LOB
0  6195    91  4.09  1.350  9.2  1.1  3.0  8.2  2.73  1096
1  5978   120  3.44  1.196  8.0  0.9  2.8  9.7  3.46  1063
2  6071    96  3.93  1.237  8.1  1.1  3.0  8.6  2.87  1062
3  6154   106  4.10  1.256  8.4  1.2  2.9  8.4  2.93  1080
4  6032   106  4.04  1.248  8.2  1.1  3.0  8.5  2.78  1080

[5 rows x 36 columns]
```

```
In [2]: # Merge datasets on team name
        data = pd.merge(hitting, pitching, on='Tm', suffixes=('_bat', '_pit'))

        # Drop unneeded columns
        data = data.drop(columns=['#Bat', '#P'])

        # Preview merged data
        print(data.head())
```

```
                      Tm  BatAge   R/G  G_bat    PA    AB  R_bat  H_bat   2B  \
0  Arizona Diamondbacks    28.6  5.47    162  6284  5522    886   1452  271
1        Atlanta Braves    29.3  4.35    162  6075  5481    704   1333  273
2     Baltimore Orioles    26.9  4.85    162  6176  5567    786   1391  262
3        Boston Red Sox    27.3  4.64    162  6192  5577    751   1404  311
4          Chicago Cubs    27.8  4.54    162  6116  5441    736   1318  253

   3B  ...    BF  ERA+   FIP   WHIP   H9  HR9  BB9  SO9  SO/W  LOB_pit
0  37  ...  6195    91  4.09  1.350  9.2  1.1  3.0  8.2  2.73     1096
1  15  ...  5978   120  3.44  1.196  8.0  0.9  2.8  9.7  3.46     1063
2  33  ...  6071    96  3.93  1.237  8.1  1.1  3.0  8.6  2.87     1062
3  30  ...  6154   106  4.10  1.256  8.4  1.2  2.9  8.4  2.93     1080
4  29  ...  6032   106  4.04  1.248  8.2  1.1  3.0  8.5  2.78     1080

[5 rows x 62 columns]
```

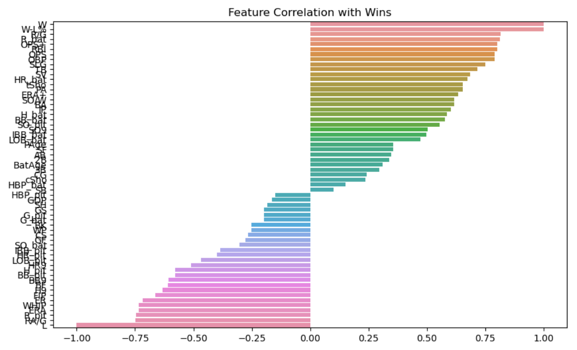
```
In [3]: # Target variable
        y = data['W']

        # Drop columns we won’t use as predictors
        X = data.drop(columns=['Tm', 'W', 'L', 'W-L%'])

        # Optional: Normalize or standardize here

        # Handle categorical data if any (not much here)
```

```
In [4]: from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
        from sklearn.metrics import mean_absolute_error, r2_score

        # Train-test split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

        # Fit a linear regression model
        model = LinearRegression()
        model.fit(X_train, y_train)

        # Predict
        y_pred = model.predict(X_test)

        # Evaluate
        print("R2 Score:", r2_score(y_test, y_pred))
        print("MAE:", mean_absolute_error(y_test, y_pred))
```

```
R2 Score: -0.28802360956497397
MAE: 9.600009082171349
```

```
In [6]: # Select only numeric columns
        numeric_data = data.select_dtypes(include='number')

        # Correlation with wins
        correlation = numeric_data.corr()['W'].sort_values(ascending=False)

        print(correlation)

        # Plot the correlations
        import seaborn as sns
        import matplotlib.pyplot as plt

        plt.figure(figsize=(10, 6))
        sns.barplot(x=correlation.values, y=correlation.index)
        plt.title('Feature Correlation with Wins')
        plt.show()
```

```
W         1.000000
W-L%      0.999928
R/G       0.814450
R_bat     0.811557
OPS+      0.801851
            ...
WHIP     -0.733830
ERA      -0.735495
R_pit    -0.746551
RA/G     -0.748635
L        -0.999803
Name: W, Length: 61, dtype: float64
```



```
In [7]: selected_features = ['R/G', 'R_bat', 'OPS+', 'RA/G', 'R_pit', 'ERA', 'WHIP']
        X_selected = data[selected_features]
```

```
In [8]: from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
        from sklearn.metrics import r2_score, mean_absolute_error

        # Split again with selected features
        X_train, X_test, y_train, y_test = train_test_split(X_selected, y, test_size=0.2, random_state=42)

        # Linear Regression
        model = LinearRegression()
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        # Evaluate
        print("R2 Score:", r2_score(y_test, y_pred))
        print("MAE:", mean_absolute_error(y_test, y_pred))
```

```
R2 Score: 0.599082885415422
MAE: 4.998881286577247
```

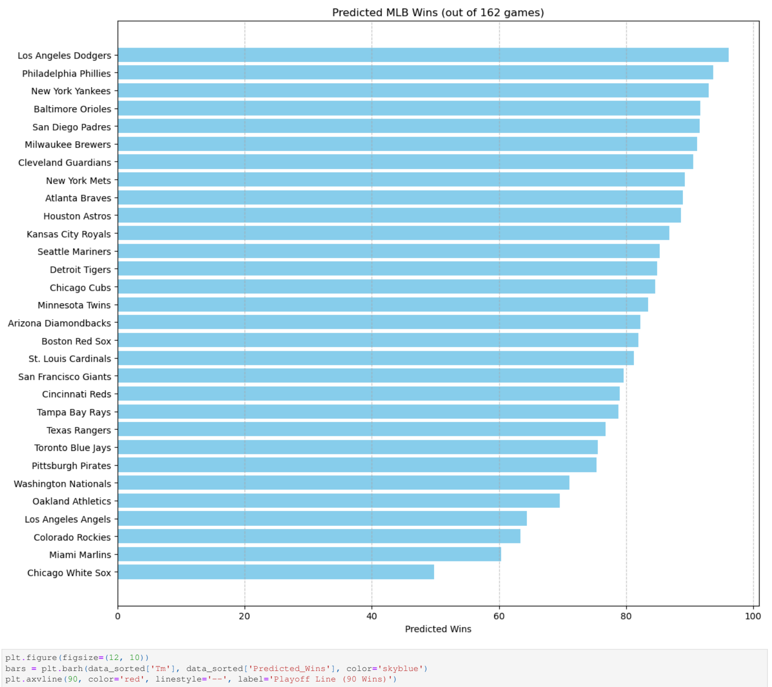
```
In [9]: # Try Random Forest
        from sklearn.ensemble import RandomForestRegressor

        rf = RandomForestRegressor(random_state=42)
        rf.fit(X_train, y_train)
        y_pred_rf = rf.predict(X_test)

        print("Random Forest R2:", r2_score(y_test, y_pred_rf))
        print("Random Forest MAE:", mean_absolute_error(y_test, y_pred_rf))
```

```
Random Forest R2: 0.9817573619631902
Random Forest MAE: 1.1766666666666634
```

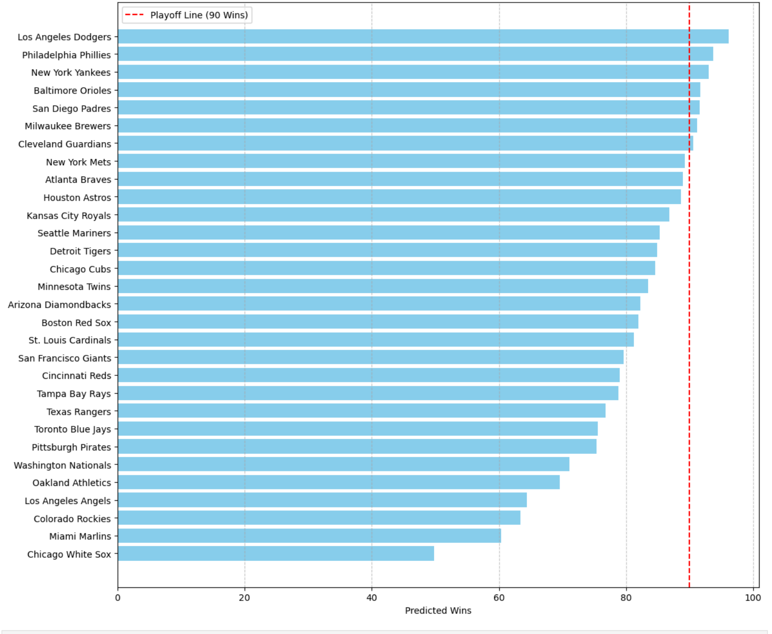
```
In [10]: # Predict wins for all teams in the dataset
         predicted_wins = rf.predict(X_selected)

         # Add predictions to the original DataFrame
         data['Predicted_Wins'] = predicted_wins.round(1)  # Round to 1 decimal for clarity

         # Sort by most predicted wins
         data_sorted = data[['Tm', 'Predicted_Wins']].sort_values(by='Predicted_Wins', ascending=False)

         print(data_sorted)
```

```
                       Tm  Predicted_Wins
13    Los Angeles Dodgers            96.1
20  Philadelphia Phillies            93.7
18       New York Yankees            93.0
2       Baltimore Orioles            91.7
22       San Diego Padres            91.6
15      Milwaukee Brewers            91.2
7     Cleveland Guardians            90.6
17          New York Mets            89.2
1          Atlanta Braves            88.9
10         Houston Astros            88.6
11     Kansas City Royals            86.8
23       Seattle Mariners            85.3
9          Detroit Tigers            84.9
4            Chicago Cubs            84.6
16        Minnesota Twins            83.5
0    Arizona Diamondbacks            82.2
3          Boston Red Sox            81.9
25    St. Louis Cardinals            81.2
24   San Francisco Giants            79.6
6         Cincinnati Reds            79.0
26         Tampa Bay Rays            78.8
27          Texas Rangers            76.8
28      Toronto Blue Jays            75.6
21     Pittsburgh Pirates            75.4
29   Washington Nationals            71.1
19      Oakland Athletics            69.6
12     Los Angeles Angels            64.4
8        Colorado Rockies            63.4
14          Miami Marlins            60.3
5       Chicago White Sox            49.8
```

```
In [11]: import matplotlib.pyplot as plt

         plt.figure(figsize=(12, 10))
         plt.barh(data_sorted['Tm'], data_sorted['Predicted_Wins'], color='skyblue')
         plt.xlabel("Predicted Wins")
         plt.title("Predicted MLB Wins (out of 162 games)")
         plt.gca().invert_yaxis()  # Highest wins on top
         plt.grid(axis='x', linestyle='--', alpha=0.7)
         plt.tight_layout()
         plt.show()
```



```
In [12]: plt.figure(figsize=(12, 10))
         bars = plt.barh(data_sorted['Tm'], data_sorted['Predicted_Wins'], color='skyblue')
         plt.axvline(90, color='red', linestyle='--', label='Playoff Line (90 Wins)')
         plt.xlabel("Predicted Wins")
         plt.title("Predicted MLB Wins with Playoff Threshold")
         plt.gca().invert_yaxis()
         plt.grid(axis='x', linestyle='--', alpha=0.7)
         plt.legend()
         plt.tight_layout()
         plt.show()
```



```
In [ ]:
```