Linear and Logistic Regression on LeBron James

For this homework I wanted to take a look at modeling LeBron's points and wins in the last five seasons. I used a Linear Regression model to analyze his points when compared to his other stats and a Logistic Regression modle to analyze his wins when compared to his stats during that game. My Repository With Complete Code

Gathering Data

Similarly to the first homework, I used the NBA API to scrape data from the official NBA website to gather LeBron's gamelogs from the last five seasons (2020-present). I then saved those gamelogs to a csv file `data/lebron/lebron_gamelogs_last_five.csv` These gamelogs include statistics such as points, rebounds, assists, etc as well as matchup information such as opponent, result, and date.

In [ ]:
```
for season in seasons:
    gamelogs = playergamelog.PlayerGameLog(player_id=lebron_id, season=season)
    gamelogs_df = gamelogs.get_data_frames()[0]
    gamelogs_df['SEASON'] = season
    all_gamelogs = pd.concat([all_gamelogs, gamelogs_df], ignore_index=True)
```

Creating the Linear Model

I wanted to explore the relationship between LeBron's points and the amount of minutes he played, and the number of shots he took.

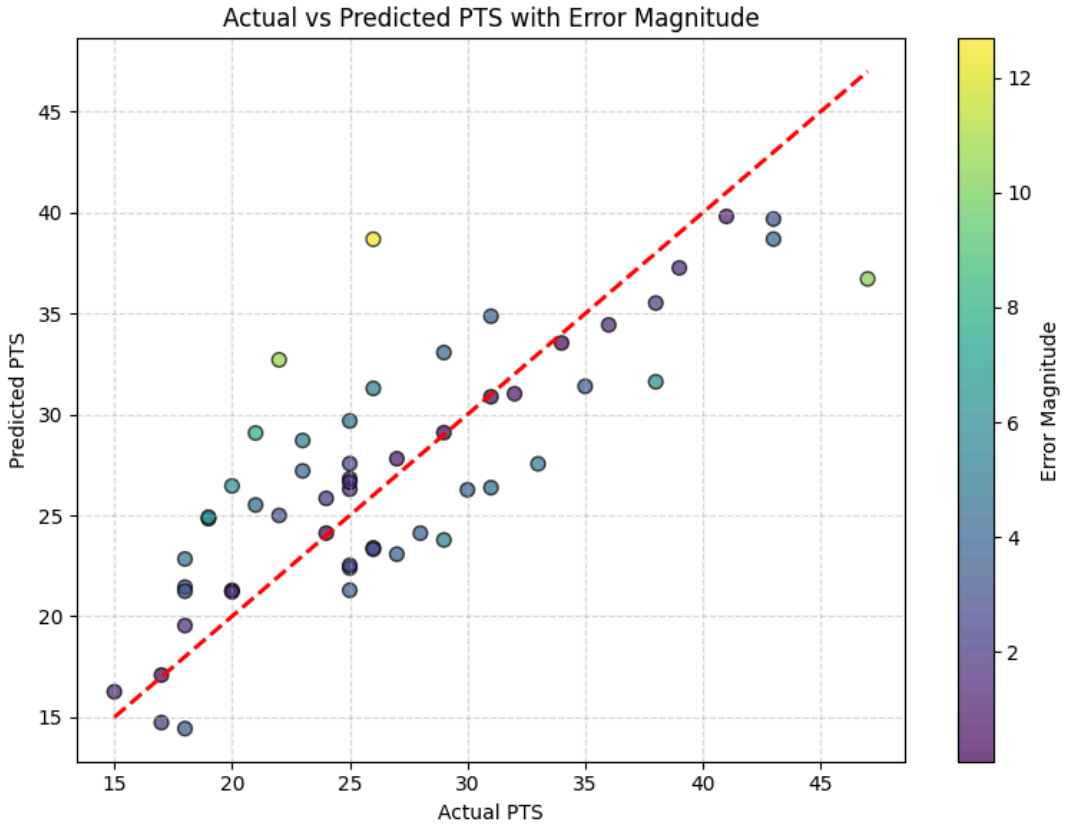
In [ ]:
```
features = ['MIN', 'FGA', 'FTA', 'FG3A']
target = 'PTS'
```

In [ ]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_sta

model = LinearRegression()
model.fit(x_train, y_train)

y_pred = model.predict(x_test)
```

I also eliminated all outliers in my data in an attempt to generate a more accurate model.

In [ ]:
```
z_scores = zscore(data['PTS'])
abs_z_scores = abs(z_scores)
filtered_entries = (abs_z_scores < 3)
data = data[filtered_entries]
```

This model finished with very modest metrics. The Mean Squared Error was 19.424 and the R2 Score was 0.649. Below is a visual representation of how the model performed, plotting the predicted point totals with the expected point totals.



I do believe Linear Regression could work for predicting point totals but I will need to look at other features or perhaps create my own features that would make the model more accurate.

Creating the Logistic Model

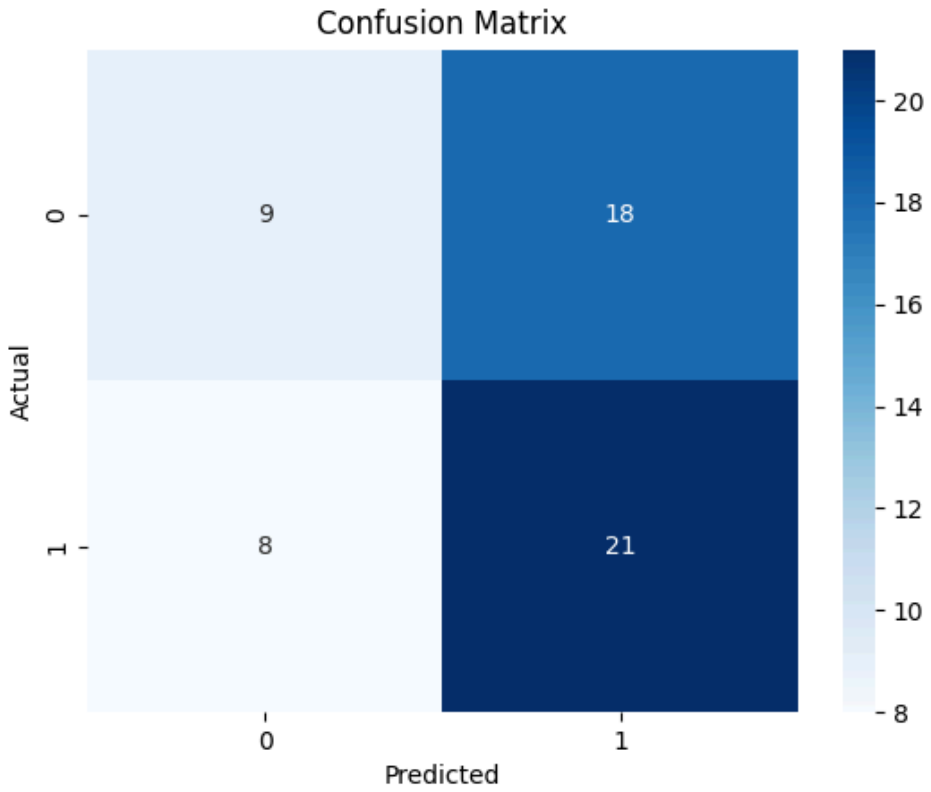
Next I wanted to explore how LeBron's in game performance impacted his team's winning. To do this I used a Logistic Regression that took a look at the relationship between his in game stats and the result of the game.

In [ ]:
```
data['WIN'] = data['WL'].apply(lambda x: 1 if x == 'W' else 0)
data['HOME'] = data['MATCHUP'].apply(lambda x: 0 if '@' in x else 1)

features = ['PTS', 'REB', 'AST', 'STL', 'BLK', 'TOV', 'FGA', 'FTA', 'FG3A', 'HOME']
target = 'WIN'
```

In [ ]:
```
x = data[features]
y = data[target]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_sta
```

```
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
```

This model performed significantly worse than the Linear Regression Model and produced the below Confusion Matrix.



In this graph, a win is denoted as a 1 and a loss is denoted by a 0. As you can see, the model very much overestimates the amount of wins, generating a lot of false positives. Out of the 29 total wins in the test set, the model correctly predicted 21 of them, a respectable rate of 72%. However, out of the 27 losses, the model only predicted 9 of them correctly, yikes. This tells me multiple things, that Logistic Regression may not be the best way to predict wins an losses, or that I need to look at other data points to predict how a team will perform in a game.

Reflection

I definitely feel like I have a better foundation on when to use Linear and Logistic Regression, as well as when NOT to use them. My ultimate goal in this project is to be able to predict a player's stats for their upcoming game so I will definitely need to poke around the API more to gather data that may help with that (injury data, recent performance, etc.). I hope to explore Decision Trees in my next homework.