Capstone Project: Final Report
Predicting MLB Players Salaries

By. Christopher Feliz

Table of Contents

Project Proposal + Problem Statement

Earlier last year, the Los Angeles Angels signed Mike Trout to a 12 year, $430 million contract. This is the highest amount ever paid to a player in the history of the MLB. The Angels made an investment with the hopes that he will bring even greater value over the next 12 years in the form of ticket sales and merchandise sales. But, if it doesn't turn out as planned and he gets injured or begins to perform poorly, they would be at a loss. So, how can they be sure they're getting their money's worth?

A rookie, who is called up to play at the big league level is automatically given a minimum starting salary of $500,000. After a couple of years, they are able to begin partially negotiating their own contracts in the form of salary arbitration. If they are

unable to come to an agreement with the ownership, independent third parties are paid large sums to arbitrate. How can a player know their value to a potential team to maximize their negotiation position for a contract?

In this project, I plan to build a model to predict player salaries based on their previous year stats which can be used both by sports franchises, to minimize their risk, and by players, to help them determine their value. The data will be scraped from 'http://www.thebaseballcube.com/', a baseball data warehouse. The final deliverables for the project will be the code in a github repository alongside the final report and slide deck.

Data Collection and Wrangling

For this project I decided to create two separate dataframes, one consisting of pitchers and another for hitters/position players. This was due to the fact that pitchers and hitters each have their own set of recorded metrics. For example, a pitcher's measure of success is heavily weighted towards statistics such as 'W' Wins per season, 'K' Strikeouts per season, and 'ERA' Earned Run Average. Hitters are mostly focused on 'H' Hits per season, 'HR' Home Runs per season, and 'BAVG' Batting Average.

One of the first steps I took towards cleaning my dataset was searching for any missing values. During my search I found a significant amount of null values in both of my datasets. The pitcher dataset had 16% of salaries missing and the hitter dataset had 13%. This was significant because all of the missing values were located in the target variable 'Salary'. After some research i decided not to drop the missing values but to instead fill them. I filled each missing value with the minimum salary for that respective year. Due to the fact that the MLB has been increasing their minimum salary throughout the years, I figured I couldn't just fill the nulls with one single value.

In order to fill in the null values I created a dictionary with years as the key and minimum salary for that year as a value. I then created a new column called 'salary_filled', by using the map function to iterate through each row and fill any missing values with the salary dictionary. The code sample can be seen below.

```
#checking for percentage of missing data
df_pitcher.isnull().sum()/len(df_pitcher)

#replacing salary null values with league minimum for respective years
salary_dict = {2019:555000, 2018:545000, 2017:535000, 2016:507500, 2015:507500, 2014:480000,
               2013:480000, 2012:480000, 2011:414000, 2010:400000, 2009:400000, 2008:390000,
               2007:380000, 2006:327000, 2005:316000, 2004:300000, 2003:300000, 2002:300000,
               2001:200000, 2000:200000, 1999:109000, 1998:109000, 1997:109000, 1996:109000,
               1995:109000, 1994:100000, 1993:100000, 1992:100000, 1991:100000, 1990:100000}

df_pitcher['salary_filled'] = df_pitcher['salary'].fillna(df_pitcher['year'].map(salary_dict))
```

After dealing with all the missing values and rearranging columns, I started to do some feature engineering. I needed to figure out how many years each player had been in the MLB. Both datasets were unorganized and only included players by groups for each season. Implementing this feature in each data set will help with organization and filtering going forward.

In order to create this new feature I sorted the entire dataset by columns ‘playerName’ and ‘year’. Next, I used the Pandas groupby function to split my dataset into groups for each player, and chained it with a cumulative count method. The new feature was called ‘total_years_mlb’, and returned the number of years in the MLB for each player . I also added another feature which returned the minimum salary for each year in ‘total_years_column’. This was built by using the apply method with a lambda function throughout each row in the ‘year’ column. The addition of these new features will help me gain a more valuable insight into my data. The code can be seen below.

```
#add column with number of year in MLB
df_pitcher_final = df_pitcher.sort_values(by=['playerName', 'year'])
df_pitcher_final['year'] = df_pitcher_final['year'].astype(str)
df_pitcher_final['total_years_mlb'] = df_pitcher_final.groupby('playerName').cumcount()+1

#add earned minimum salary for player year in MLB
df_pitcher_final['minimum_year'] = df_pitcher_final['year'].apply(lambda x: salary_dict[int(x)])
```

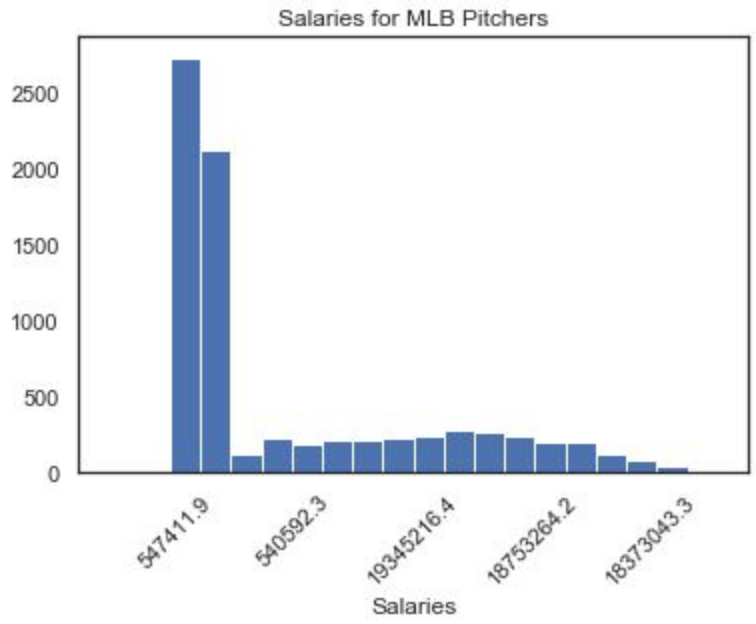
Another issue that had to be accounted for was monetary inflation. The salaries paid toward athletes in the 1990's was very different than that paid today. In order to account for inflation I installed a python library called CPI. CPI is a library that adjusts U. S. dollars by the current consumer price index. A combination of the apply method and a lambda function was used by passing the salary and year as arguments. The code can be seen below.

```
#adjust salary for inflation
df_pitcher['adj_salary'] = df_pitcher.apply(lambda x: cpi.inflate(x.salary, x.year), axis=1)
df_pitcher['adj_salary_filled'] = df_pitcher.apply(lambda x: cpi.inflate(x.salary_filled, x.year),
axis=1)
df_pitcher['adj_salary'] = df_pitcher['adj_salary'].round(1)
df_pitcher['adj_salary_filled'] = df_pitcher['adj_salary_filled'].round(1)
```

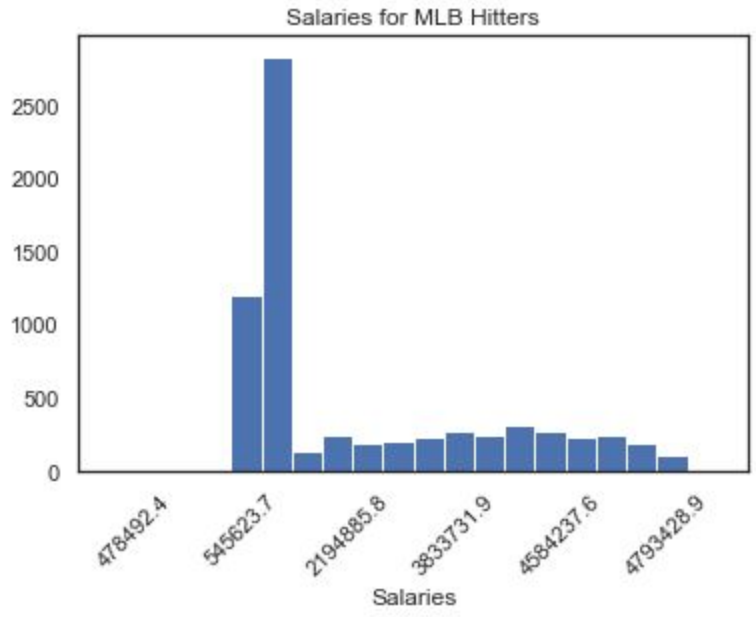
After looking through both datasets I did find a couple of data points that would be considered 'outliers'. I ultimately decided to leave the outliers within my data for a couple of reasons. The first reason is because the salaries paid towards athletes are always going to increase year by year. For example, throughout the last 20 years the minimum salary for MLB players has increased from $200,000 to $555,000. So, what might be considered an outlier today might not be in a couple of years. The final step I took during my data wrangling process was subsetting the data and using players only after the 2010 season. MLB is an expanding and ever changing sport and a lot has happened over the last 10 years. Some minor and major changes, such as including new rules.

Exploratory Data Analysis

Distribution of Salaries(Target Variable) for Pitchers and Hitters



The chart above depicts the salary distribution for pitchers in the MLB. According to the chart most of the pitchers earn a salary around $550,000, which is close to the minimum salary amount. As we move to the right, we can see a drop off for players earning more than the league minimum. Towards the end of the chart there are less than 100 players earning a salary of $20 million or more.

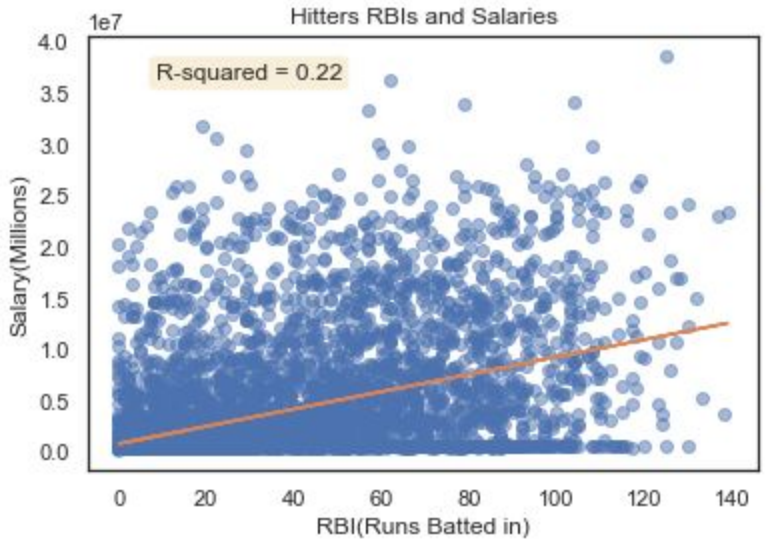


Similarly to the pitchers distribution of salaries, hitters too mostly earn the league minimum around $550,000. As we move to the right, we can see a drop off for players

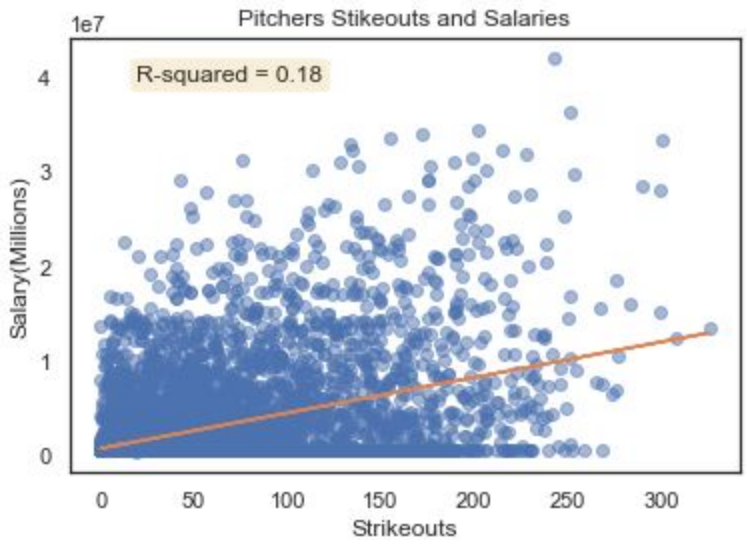
earning more than the league minimum. Towards the end of the chart we can see that the distribution of salaries for hitters goes all the way up $50 million.

Independent Variables Vs. Dependant Variable

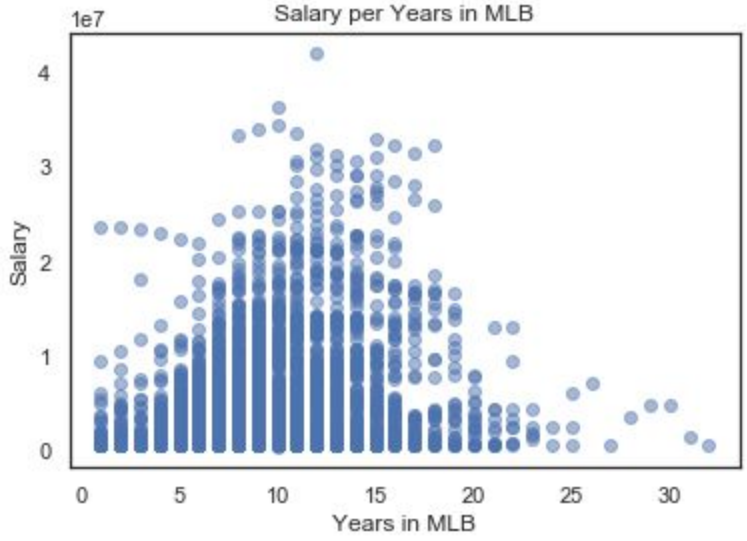
The following charts will be a comparison between multiple independent variables and our target variable ‘salary’. These relationships will be depicted by a scatter plot.



The chart above represents a positive correlation between a player's RBI(runs batted in) and their salary. This feature also has the strongest level of correlation with the target variable within the hitter dataset. As the number of RBI’s increases so does a player's salary. We can also see a couple of data points with high RBI totals and a low salary. This could be due to exceptionally good rookies just joining the league and being paid the minimum.



The chart above represents a positive correlation between a pitcher's strikeouts and their salary. This feature also has the strongest level of correlation with the target variable within the pitcher dataset. As the number of strikeouts increases so does a player's salary. The data includes different types of pitchers such as starting pitchers, relief pitchers, and closers. Number of strikeouts is one of the top measured pitching metrics per season.

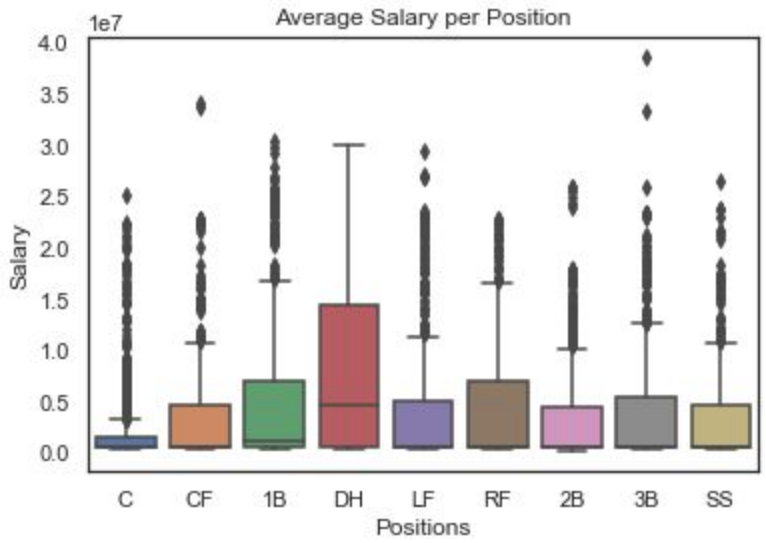


The chart above depicts a weak or non-existent positive correlation between ‘Years in MLB’ and ‘Salary’. During the first couple of years in the MLB we can see a gradual increase in salary. We can also see that a player's maximum salary tends to peak after

7-10 years of MLB service. After 15 years of MLB service a player's salary begins to decrease rapidly. This could be due to aging and below average performance.

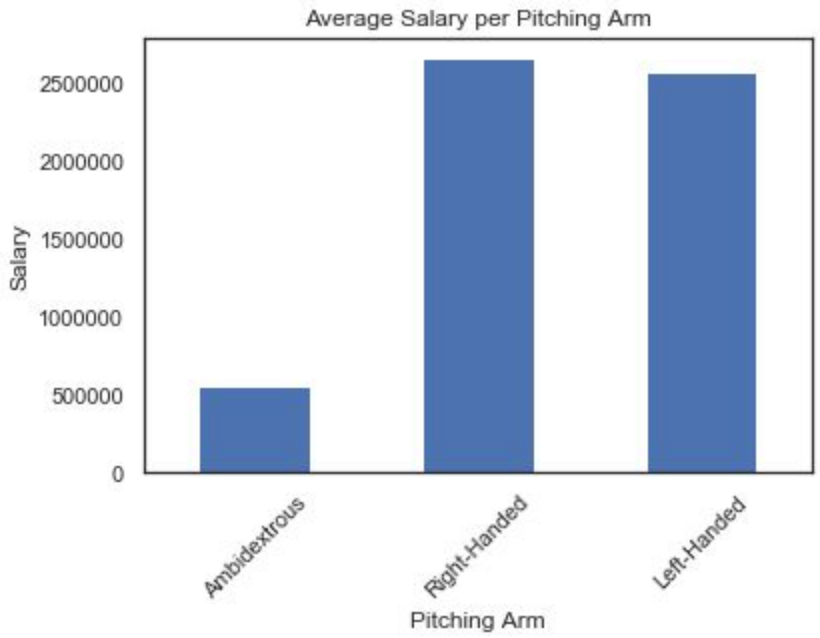
Categorical Data vs Target Variable

➔ In the following graph I will be creating a box and whisker plot, depicting the salaries paid to each position.



The chart above depicts the average salaries per player position. Beginning with catchers, it looks like they are the least paid on the diamond. We can also see that there are a couple of outlier points reaching up to salaries of $2.5million. This was rather interesting to see because catchers are in charge of controlling the game, and calling every pitch. On the other we can see that DH or Designated hitters are paid on average a higher salary. The salary paid to a DH seems to be more consistent than the salaries paid to other position players, due to the presence of less outliers.

Statistical Inference

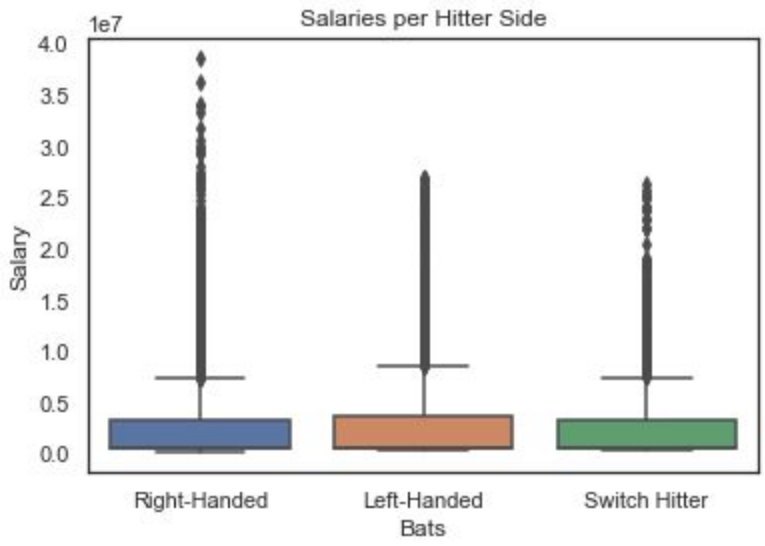


Is there a difference between the salaries paid towards right handed and left handed pitchers?

H_0 : The salaries for both left and right handed pitchers are the same.

H_1 : The salaries for left and right handed pitchers are different.

An independent-samples t-test was conducted to compare the salaries paid to left and right handed pitchers, with an alpha level set at .05. Results of the independent sample t-tests indicated that there were not significant differences in salaries paid to pitchers who threw with their left or right arm, (t(7689) = -0.75, p = 0.77). Specifically, our results suggest that pitchers are paid the same regardless of throwing arm.



Is there a difference between the salaries paid towards right handed and left handed batters?¶

H_0 : The salaries paid to both left and right handed hitters the same.

H_1 :The salaries paid to left and right handed hitters are different.

An independent-samples t-test was conducted to compare the salaries paid to left and right handed batters, with an alpha level set at .05. Results of the independent sample t-tests indicated that there were not significant differences in salaries paid to hitters who batted with their left or right side, (t(5973) = 1.35, p = 0.17). Our results suggest that hitters are paid the same regardless of batting side.

```
lefties = df_hitter.adj_salary_filled[df_hitter['Bats'] == 'L']
righties = df_hitter.adj_salary_filled[df_hitter['Bats'] == 'R']


ttest,pval = ttest_ind(lefties,righties)
print("p-value",pval, 'ttest', ttest)
if pval <0.05:
  print("We reject the null hypothesis.")
else:
  print("We fail to reject the null hypothesis.")

p-value 0.17474496847305468 ttest 1.3572755073329725
We fail to reject the null hypothesis.
```

Machine Learning: Preprocessing + Model Selection

Preprocessing

In order to predict the salaries of MLB players, I chose to use several regression algorithms. Considering our target variable was continuous and our data had labels, it was best to use a supervised learning technique. I then needed to create a predictive model that best fit each of my player datasets. One of the key differences between the two datasets was the number of features used. A pitchers performance is measured using more metrics than that for a hitter. Thus leading to an increase in features.

Before building any machine learning models I had to prepare my dataset. The first issue I dealt with was scaling my data. Both datasets had values ranging from .100 to 1000+ . So it was important to have all the features within the same scale to avoid any adverse effect on the model. Both datasets were scaled using the Standard Scaler from Scikit-learn.

After scaling my data I then had to handle the categorical variables. In the data, the feature “positions” contained a range of nine positions that a player could play. To convert my categorical variables to numbers I used the pandas get dummies function. I also decided to use the log transformation of my target variable to limit the effect of any outliers. After completion of my preprocessing I began to build my models. The code and metrics can be seen below.

Model Selection

The first regression model I built was using a DummyRegressor from scikit-learn. I used this model to get a baseline estimate for my metric of choice ‘Root Mean Squared Error’. The next algorithm I used was a Linear Regression model from Sci-kit learn. This model's performance was subpar, due to a lot of multicollinearity in my features. To correct this I used a dimensionality reduction technique known as Principal Component Analysis. Using PCA helped me reduce the number of features from 156 to 7, and the model's root mean squared error.

I then continued with other regression models, especially tree based ensemble methods. These models are known to be effective against outliers. A model robust to outliers was the Random Forest Regressor . I also made use of some of the best

models currently available such as XGBoost, Gradient Boosting, LightGBM, HistGradientBoosing, and CatBoost. In order to find the right combination of hyperparameters for each model I used Grid Search or Randomized Search.

The best performing model for the hitter dataset was the LightGBM Regressor. This model's final metric was an R-squared of 0.80 and RMSE of 3,236,633. The hyperparameters were 'n_estimators' which were set to 200 and 'max_depth' set to 6. The code for the model with the best hyperparameters, as well as the predictions can be seen below.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error

#drop useless columns
df_hitter_finall = df_hitter.drop(columns=['salary', 'borndate',
                                    'Place', 'playerName', 'HT',
'teamName'], axis=1)
#encode categorical features
dfh_obj = df_hitter_finall.select_dtypes(include=['object']).copy()
dfh_obj.columns
dummies = pd.get_dummies(df_hitter[['Bats', 'Throws', 'LeagueAbbr',
'posit']], drop_first=True)


# Drop the column with the independent variable (Salary), and columns
for which we created dummy variables
X_ = df_hitter_finall.drop(['Bats', 'Throws', 'LeagueAbbr', 'posit',
'adj_salary_filled' ], axis=1).astype('float64')

# Define the feature set X, y
X = pd.concat([X_, dummies], axis=1)
y = df_hitter_finall.adj_salary_filled
y_log = np.log(df_hitter_finall.adj_salary_filled)


X_train, X_test, y_train, y_test = train_test_split(X, y_log,
test_size=0.2, random_state=42)

#scale data
std = StandardScaler()
X_train_scaled = std.fit_transform(X_train)
X_test_scaled = std.transform(X_test)

from lightgbm import LGBMRegressor

lgbm = LGBMRegressor(n_estimators=200, max_depth=6, num_leaves=15)
lgbm.fit(X_train_scaled, y_train)
y_pred = lgbm.predict(X_test_scaled)
y_pred_train_ = lgbm.predict(X_train_scaled)

rmse = np.sqrt(mean_squared_error(np.exp(y_test), np.exp(y_pred)))

print(rmse)
print('Train r2 score: ', r2_score(y_pred_train_, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))

3236633.8627096354
Train r2 score:  0.8753271859626053
Test r2 score:  0.8003821458146158
```

In the pitcher dataset the best model was the CatBoost Regressor. This model's final metric was an R-squared of 0.77 and RMSE of 2,754,297. The hyperparameters 'learning rate' were set to 0.03 and 'iterations' to 3000. I also implemented the use of early stopping rounds in this model to minimize any possible overfitting.
Working with this model was rather interesting because it encodes any categorical variables in the dataset. The code for the model with the best hyperparameters, as well as the predictions can be seen below.

```
from catboost import CatBoostRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split

df_pitcher_fin = df_pitcher.drop(columns=['salary', 'borndate',
                                    'Place', 'playerName', 'HT',
'teamName'], axis=1)


X = df_pitcher_fin.drop(columns=['adj_salary_filled'])
y = df_pitcher_fin.adj_salary_filled
y_log = np.log(df_pitcher_fin.adj_salary_filled)

X_train, X_test, y_train, y_test = train_test_split(X, y_log,
test_size=0.2, random_state=42)


cat = CatBoostRegressor(iterations=3000, verbose=200,
                        learning_rate=0.03,loss_function='RMSE',
                        early_stopping_rounds= 200,
                        random_seed=42)
cat.fit(X_train, y_train, cat_features=['Bats', 'Throws',
'LeagueAbbr', 'posit'],  eval_set=(X_test, y_test)
       )
y_pred = cat.predict(X_test)
y_pred_train = cat.predict(X_train)
rmse = np.sqrt(mean_squared_error(np.exp(y_test), np.exp(y_pred)))
print('Train r2 score: ', r2_score(y_pred_train, y_train))
print('Test r2 score: ', r2_score(y_test, y_pred))
print(rmse)
Train r2 score:  0.8225891937818299
Test r2 score:  0.7773951221427647
2754297.969240993
```

Machine Learning: Final Results

- **Hitter Dataset**

Models	RMSE	R2
Baseline Model	$5,397,560	0.00
Linear Regression w PCA	$4,644,008	0.55
RandomForest Regressor	$3,510,004	0.78
XGBoost Regressor	$3,315,287	0.80
HistGradientBoosting Regressor	$3,293,081	0.79
CatBoost Regressor	$3,272,132	0.80
LightGBM Regressor	$3,236,633	0.80

- **Pitcher Dataset**

Models	RMSE	R2
Baseline Model	$4,683,385	0.00
XGBoost Regressor	$2,832,288	0.76
LightGBM Regressor	$2,818,405	0.76
RandomForest Regressor	$2,796,006	0.77
HistGradientBoost Regressor	$2,757,535	0.77
CatBoost Regressor	$2,754,287	0.77

Conclusion

After looking into each model's individual feature importance attribute, I was able to uncover some interesting results. There was only one model that had ranked the feature ‘positions’ within its top 10 of feature importance. This model was the CatBoost Regressor used in the hitter dataset. Surprisingly, other models did not view this feature as informative enough. From a general standpoint a players position is thought of to be an important factor in their salary.

Overall I was able to get over a 40% reduction in the root mean squared error of the baseline models in the hitter and pitcher datasets respectively. For future iterations I will gather data on player injuries and medical history. Collecting information on a player's past injuries will help predict any future injuries as well as affect their total value. Therefore leading to a more precise salary prediction. Another feature that could be added is a player's place of origin. This could be within the states or at the international level, as they each get paid different bonuses.