Baseball Homerun prediction

Tirumalesh Nagothi
A20520569
tnagothi@hawk.iit.edu

Mohana uma sushmanth Penumarthi
A20525576
mpenumarthi@hawk.iit.edu

Tarun sai Varanasi
A20526965
tvaranasi@hawk.iit.edu

1. Abstract

This project focuses on developing machine learning models to predict home runs in professional baseball games. Through comprehensive data exploration, cleaning, and feature engineering, we prepared a dataset comprising various player and game-related features. Three classification models—Logistic Regression, Random Forest Classifier, and Decision Tree Classifier—were trained and evaluated, along with ensemble techniques like StackingClassifier. Evaluation metrics such as accuracy, precision, recall, and F1-score were used to assess model performance. The study provides valuable insights for coaches, analysts, and enthusiasts, enhancing strategic decision-making and the excitement of baseball games.

2. Problem statement

In today's global landscape, baseball tournaments captivate audiences worldwide. Predicting whether a player will hit a home run in the next at-bat is a pivotal task for coaches and analysts. Using machine learning, this project aims to develop accurate classification models to predict home runs, empowering teams with strategic insights and enhancing the excitement for fans and broadcasters alike during every game

3. Introduction

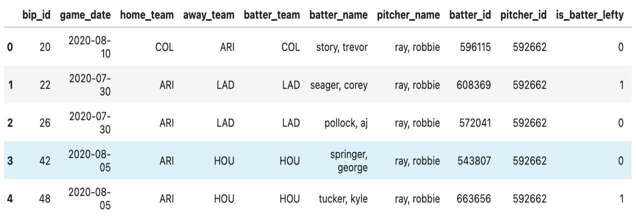
In professional baseball, accurately predicting whether a player will hit a home run during a game is crucial for teams to strategize effectively. However, despite advancements in player performance analytics, there remains a need for a reliable machine learning model that can predict the likelihood of a home run based on various factors such as player statistics, environmental conditions, pitcher performance, and game situation. Developing such a model would not only provide valuable insights for coaches and managers to optimize their game strategies but also aid broadcasters and fans in enhancing their viewing experience by anticipating exciting game-changing moments. The goal of this project is to design and implement a robust machine learning solution capable of accurately predicting home runs in real-time, leveraging historical game data and advanced statistical analysis.

4. Methodology

The proposed methodology includes five stages. They are Data collection, Data Exploration, Data Cleaning, Model Selection, and Model Evaluation. In data collection, we have gathered the dataset from Kaggle. Next step is data exploration. we used exploratory data analysis (EDA) techniques to identify trends, patterns in the data. In data cleaning, we process the data by cleaning and removing null values and transforming it into a suitable format. When the data is ready, we go to model Selection stage. This stage involves the development of various machine learning and statistical models, such as Logistic Regression, Random Forest Classifier, and Decision tree classifier also employing model selection techniques to determine the most suitable model for our dataset. In model evaluation, we evaluated the performance of our models. We used variety of evaluation metrics, such as accuracy, Precision, Recall, and F1-Score to assess the performance of our models

4.1 Data collection:

We have collected the dataset from Kaggle. It contains 7894 entries and 25 features.

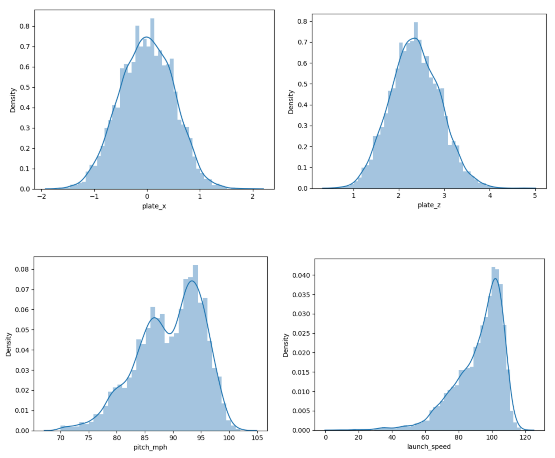


	bip_id	game_date	home_team	away_team	batter_team	batter_name	pitcher_name	batter_id	pitcher_id	is_batter_lefty
0	20	2020-08-10	COL	ARI	COL	story, trevor	ray, robbie	596115	592662	0
1	22	2020-07-30	ARI	LAD	LAD	seager, corey	ray, robbie	608369	592662	1
2	26	2020-07-30	ARI	LAD	LAD	pollock, aj	ray, robbie	572041	592662	0
3	42	2020-08-05	ARI	HOU	HOU	springer, george	ray, robbie	543807	592662	0
4	48	2020-08-05	ARI	HOU	HOU	tucker, kyle	ray, robbie	663656	592662	1

4.2 Data Exploration

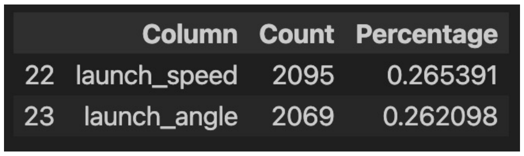
Data exploration stage involves examining and understanding the structure and patterns within the baseball dataset through various techniques such as summary statistics and data visualization. It aims to uncover insights, identify trends, and patterns, providing a foundational understanding that informs subsequent analysis and decision-making processes.

Understanding the distribution of the values in a dataset is so crucial. It helps to identify the patterns and skewness of the features. Here we have performed and visual represented the distribution of each feature which has float data type.



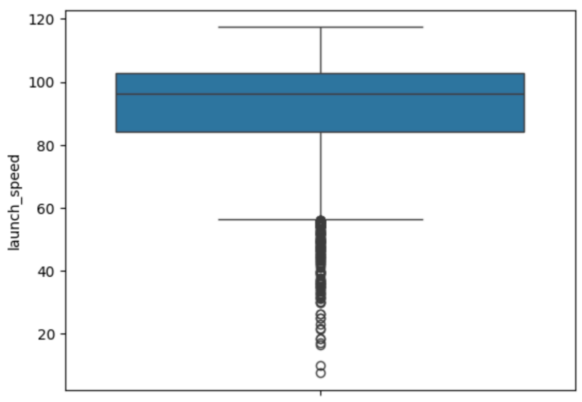
From the above graphs, we can show that there is skewness for pitch_mph, launch_speed features. We have performed the tranformation for the pitch_mph, launch_speed features using StandardScaler to check the skewness whether there is some change or not. But there is no change in the distribution, even after transforming the values.

The most commom problem in every machine learning prject are null values. Accordingly, we have checked for null values within the dataset. Subsequently, we identified null values only within the 'launch_speed' and 'launch_angle' features. Specifically, the 'launch_speed' feature contained 2095 null values, while the 'launch_angle' feature exhibited 2069 null values.

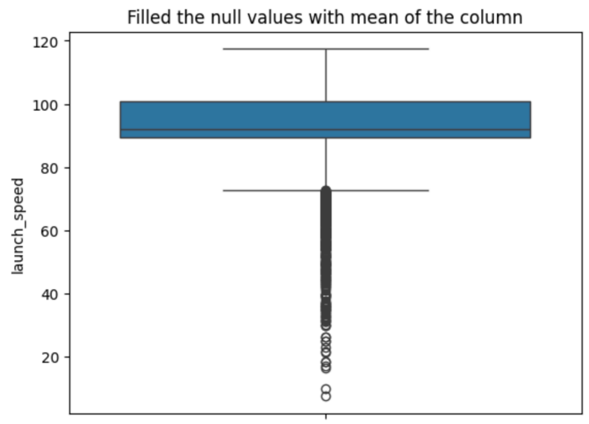


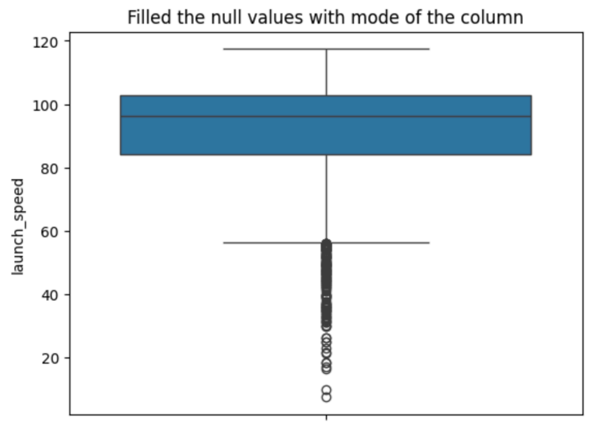
	Column	Count	Percentage
22	launch_speed	2095	0.265391
23	launch_angle	2069	0.262098

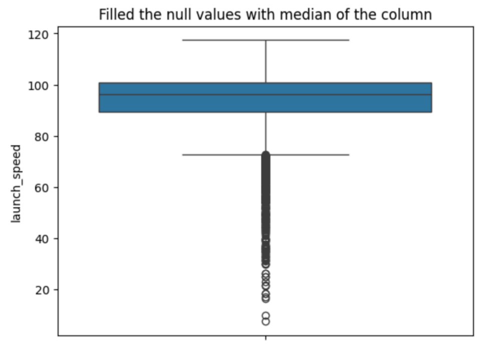
Handling these null values effectively is a crucial task. When dealing with those null values, we wanted to keep things smooth without messing up the rest of the dataset. So, we started by looking for the outliers in the 'launch_speed' feature. Then, we replaced those null values by using the mean value, and checked again for any outliers. We performed the same thing, but this time swapping in the median value and mode value too.



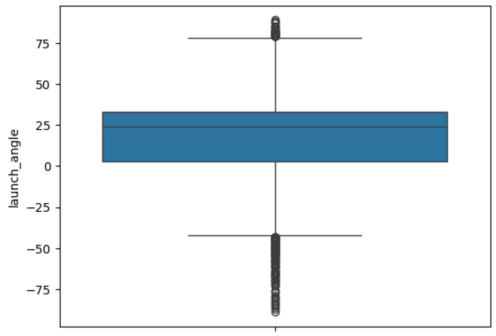
Above boxplot indicates the outliers in launch_speed feature before filling the null values.



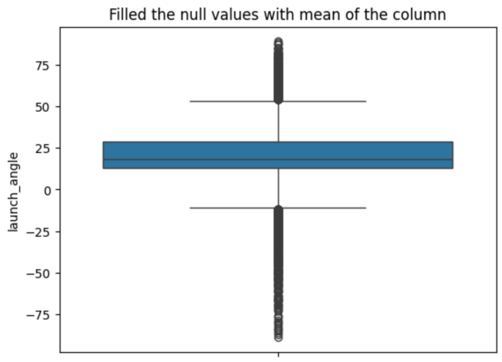


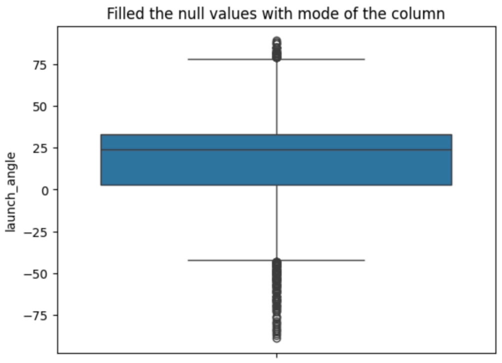


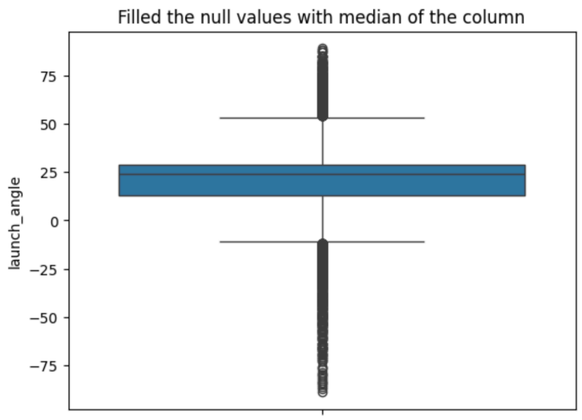
From the above observation, we found that when the values are filled with mode imputation there are no addition of outlier values according to the boxplot. Performed same operation with the feature launch_angle.



Above boxplot indicates the outliers in launch_angle feature before filling the null values.

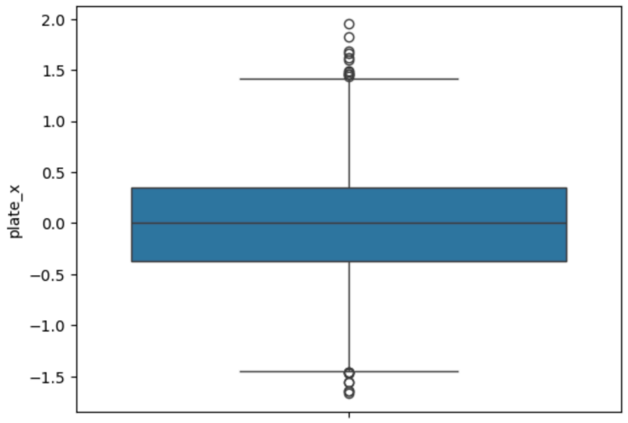


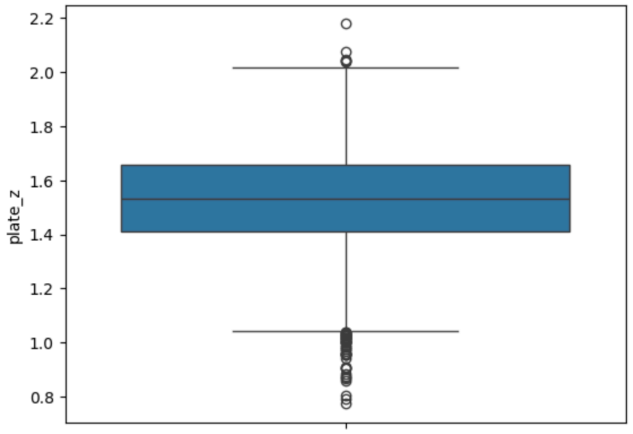




From the above observation, we found that we need to fill the null values using the mode value of the launch_angle column. So, there no more outliers added to the column

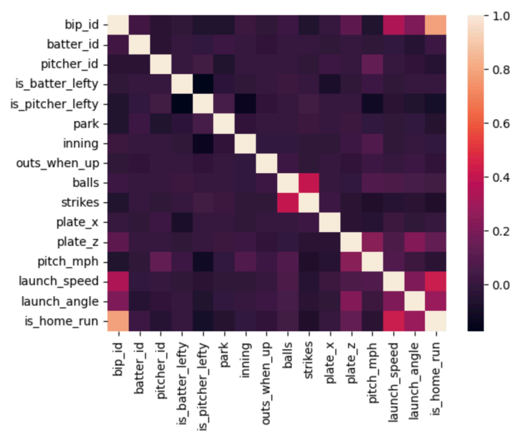
We have checked outliers for the other continous features. We are going to consider the continous features such as plate_x, plate_z, pitch_mph, launch_speed, launch_angle for checking the outliers.





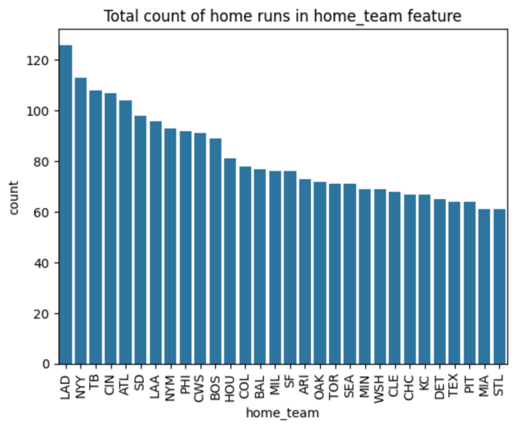
We found few outliers in each feature, but We are not supposed to change the outlier values since the values are standard and they are defined to be those values. For this issue, we can use scaling techniques to transform the data which will be used to train the model with these values.

Correlation: It measures the strength and direction of the relationship between two variables. A correlation value close to 1 indicates a strong positive correlation, while a value close to -1 indicates a strong negative correlation. A value near 0 suggests little to no linear relationship between the variables. It helps in understanding how changes in one variable affect the other, aiding feature selection and model interpretability.

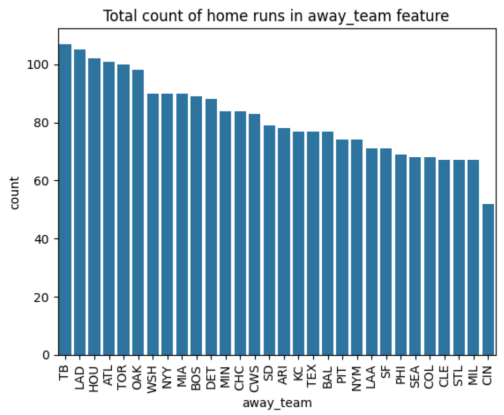


There is not much correlation between the features. Couldn't find the positive nor the negative correlation.
Checking the relationship between the input features and the output feature is very important. We have visually represented the number of home runs achieved within each category for every feature.

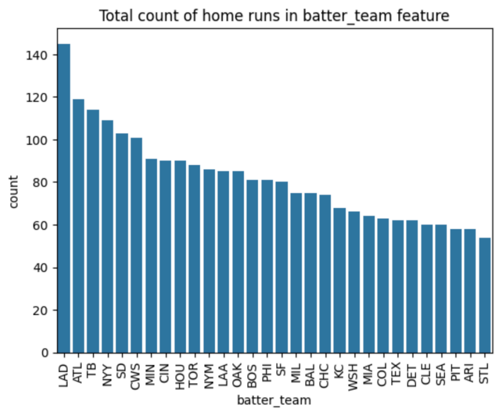
```
LAD has highest count in homeruns i.e 126; MIA has less homeruns i.e 61
```



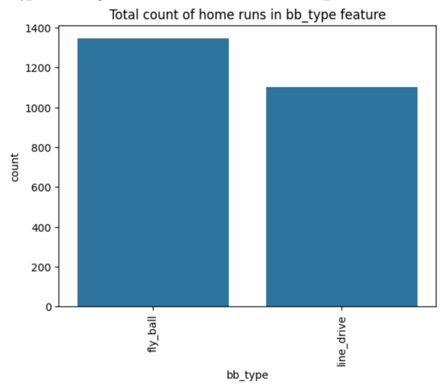
```
TB has highest count in homeruns i.e 107; CIN has less homeruns i.e 52
```



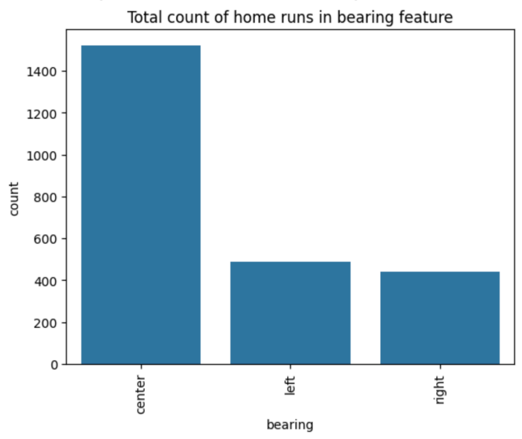
```
LAD has highest count in homeruns i.e 145; STL has less homeruns i.e 54
```



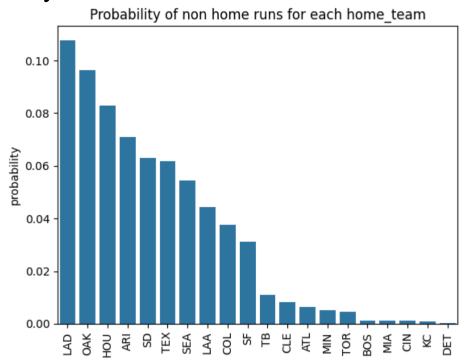
```
fly_ball has highest count in homeruns i.e 1345; line_drive has less homeruns i.e 1102
```

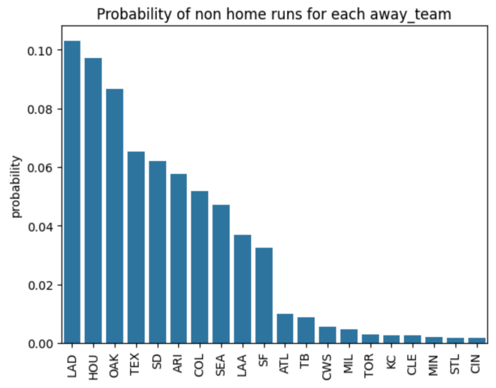


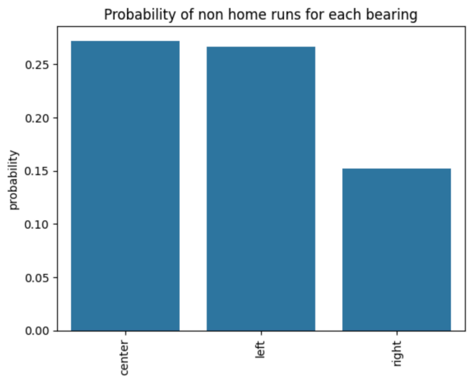
```
center has highest count in homeruns i.e 1520; right has less homeruns i.e 440
```

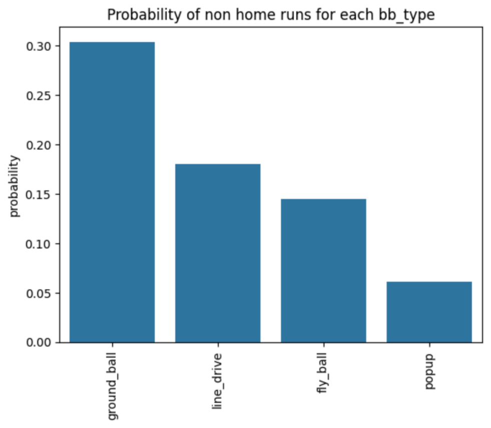


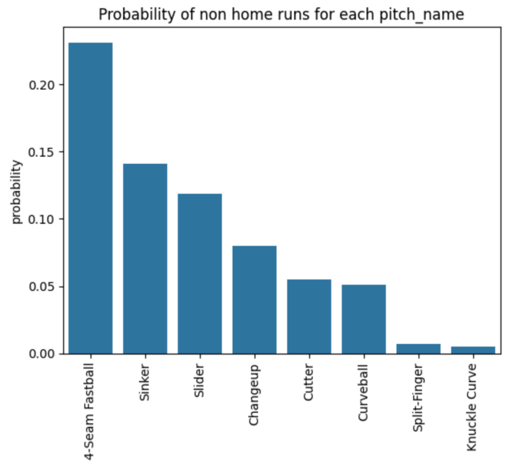
Similarly, we have visually represented the number of non-home runs achieved within each category for every feature.











4.3 Data Cleaning

Dropping columns: keeping the unnecessary data in dataset increases uncertainty while building the model. So here we have removed the unwanted column that are bip_id, batter_name, pitcher_name, game_date and removed the duplicate values.

```
data_frame = data_frame.drop(['bip_id','batter_name','pitcher_name','game_date'],axis=1)
```

```
#Removing the duplicate values
data_frame.drop_duplicates(inplace=True)
```

Handling null values: As we explored, the null values in launch_speed and launch_angle are replaced with their mode value since that was not causing any additional outliers.

Feature Encoding: We have performed label encoding in this stage. It is a technique used to convert categorical data into numerical format. Each unique category is assigned a unique integer, allowing algorithms to work with categorical variables.

```
home_team
--------
{'LAD': 1, 'OAK': 2, 'HOU': 3, 'ARI': 4, 'SD': 5, 'TEX': 6, 'SEA': 7, 'LAA': 8, 'CO
L': 9, 'SF': 10, 'TB': 11, 'ATL': 12, 'CLE': 13, 'CIN': 14, 'NYY': 15, 'MIN': 16, 'TO
R': 17, 'BOS': 18, 'NYM': 19, 'PHI': 20, 'CWS': 21, 'BAL': 22, 'MIL': 23, 'KC': 24,
'MIA': 25, 'WSH': 26, 'CHC': 27, 'DET': 28, 'PIT': 29, 'STL': 30}
away_team
--------
{'LAD': 1, 'HOU': 2, 'OAK': 3, 'TEX': 4, 'SD': 5, 'ARI': 6, 'COL': 7, 'SEA': 8, 'LA
A': 9, 'SF': 10, 'ATL': 11, 'TB': 12, 'CWS': 13, 'TOR': 14, 'MIL': 15, 'MIN': 16, 'NY
Y': 17, 'KC': 18, 'CHC': 19, 'MIA': 20, 'WSH': 21, 'DET': 22, 'BOS': 23, 'CLE': 24,
'NYM': 25, 'PIT': 26, 'STL': 27, 'PHI': 28, 'BAL': 29, 'CIN': 30}
batter_team
--------
{'COL': 1, 'LAA': 2, 'SD': 3, 'HOU': 4, 'SF': 5, 'LAD': 6, 'ARI': 7, 'SEA': 8, 'TEX':
9, 'OAK': 10, 'TB': 11, 'ATL': 12, 'MIN': 13, 'CWS': 14, 'NYY': 15, 'MIL': 16, 'CIN':
17, 'NYM': 18, 'KC': 19, 'PHI': 20, 'BOS': 21, 'TOR': 22, 'CHC': 23, 'BAL': 24, 'MI
A': 25, 'STL': 26, 'PIT': 27, 'DET': 28, 'WSH': 29, 'CLE': 30}
bearing
--------
{'center': 1, 'left': 2, 'right': 3}
pitch_name
--------
{'4-Seam Fastball': 1, 'Sinker': 2, 'Slider': 3, 'Changeup': 4, 'Cutter': 5, 'Curveba
ll': 6, 'Split-Finger': 7, 'Knuckle Curve': 8}
```

Transforming the data: We performed standard scaling for standardization in this stage. It is a preprocessing technique in machine learning used to standardize features by removing the mean and scaling to unit variance. This ensures that all features have a mean of 0 and a standard deviation of 1. Standard scaling helps algorithms converge faster and prevents features with larger scales from dominating the learning process.

Feature selection: We used SelectKBest in this stage. SelectKBest is a feature selection technique in machine learning used to select the top k features based on their importance scores. This method helps reduce dimensionality and improve model performance by focusing on the most relevant features for prediction.

4.4 Model Selection

Now the data ready for the Model development. Since it is a classification problem, we have performed three models in this project. They are Logistic Regression, Random Forest Classifier, Decision Tree Classifier. First, we split the data using train_test_split method.

Logistic Regression:
Logistic Regression is a statistical method used for binary classification. It models the probability that a given input belongs to a certain category.
The logistic function, also known as the sigmoid function, is used in logistic regression. The formula for the logistic function is:

$$f(x) = \frac{1}{1 + e^{-z}}$$

Where z is the linear combination of the input features and their respective coefficients:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$$

The coefficients β are estimated through methods like maximum likelihood estimation.

Random Forest Classifier:
Random Forest is an ensemble learning method used for classification tasks. It operates by constructing a multitude of decision trees during training and outputting the mode (for classification) or the average prediction (for regression) of the individual trees. Each tree in the forest is built using a random subset of features and data samples, and the final prediction is determined by aggregating the predictions of all individual trees. Random Forests are known for their robustness, scalability, and ability to handle high-dimensional datasets with noisy or correlated features. They are effective for both classification and regression tasks and are widely used in various domains including finance, healthcare, and bioinformatics.

Decision tree classifier:
A Decision Tree Classifier is a non-parametric supervised learning method used for classification tasks. It builds a tree-like structure where each internal node represents a feature (or attribute), each branch represents a decision based on that feature, and each leaf node represents the class label. The decision tree algorithm recursively splits the dataset based on the values of input features, aiming to maximize the homogeneity (purity) of the resulting subsets with respect to the target variable.

$$IG(D, f) = H(D) - \sum_{v \in values(f)} \frac{|D_v|}{|D|} H(D_v)$$

where,

- $IG(D, f)$ is the information gain achieved by splitting dataset D on feature f.
- $H(D)$ is the entropy of dataset D, defined as

  $$H(D) = -\sum_{k=1}^{K} p_k log_2(p_k)$$

  Where p_k is the proportion of sample belonging to class k in dataset D
- Values(f) represent the possible values of f
- |D| represents the total number of samples in Dataset D
- $|D_v|$ represents the total number of samples in Dataset D_v where feature f has value v
- $H(D_v)$ is the entropy of subset of D_v, calculated similarity to H(D) using the class distribution within D_v.

4.5 Model Evaluation

After building these models, we evaluated the accuracy for each model. Accuracy measures the proportion of correctly classified samples out of the total number of samples.

$$\text{Accuracy} = \frac{\text{Total Number of Predictions}}{\text{Number of Correct Predictions}}$$

```
log_regression:  0.883495145631068
random_forest:  0.926129168425496
decision_tree:  0.8940481215702828
```

We have used StackingClassifier in this stage. The StackingClassifier is a type of ensemble learning technique that combines multiple base classifiers with a meta-classifier to improve prediction performance. It involves training several base classifiers on the original training data and then using the predictions from these base classifiers as input features for a meta-classifier, which makes the final prediction.

```
stacking_clf.score(x_test,y_test)
```

```
0.926129168425496
```

Also, for each model we have evaluted the Classification report which has Precision, recall, f1 score and confusion matrix.

Classification_report:
The classification_report function generates a comprehensive report that includes several key metrics for each class in the classification problem. It provides metrics such as precision, recall, F1-score, and support for each class, as well as the macro and weighted averages of these metrics across all classes. This report helps in understanding the performance of the classifier for each class and overall.

Confusion_matrix:
The confusion_matrix function creates a table that shows the actual classes against the predicted classes. It helps visualize the performance of the classifier by showing the number of true positives, false positives, true negatives, and false negatives for each class. This matrix provides a detailed view of where the classifier is making errors, such as confusing one class for another.

Precision:
Precision measures the proportion of true positive predictions among all positive predictions made by the model.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Recall (Sensitivity):
Recall measures the proportion of true positive predictions among all actual positive samples in the dataset.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

F1-Score:
F1-score is the harmonic mean of precision and recall. It balances precision and recall and is useful when the class distribution is imbalanced.

$$\text{F1} - \text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Classification report for Random Forest Classifier:

```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      1620
           1       0.92      0.84      0.88       749

    accuracy                           0.93      2369
   macro avg       0.92      0.90      0.91      2369
weighted avg       0.93      0.93      0.93      2369

[[1566   54]
 [ 120  629]]
```

Classification report for Logistic Regression:

```
              precision    recall  f1-score   support

           0       0.89      0.95      0.92      1620
           1       0.88      0.73      0.80       749

    accuracy                           0.88      2369
   macro avg       0.88      0.84      0.86      2369
weighted avg       0.88      0.88      0.88      2369

[[1544   76]
 [ 200  549]]
```

Classification report for Decision Tree Classifier

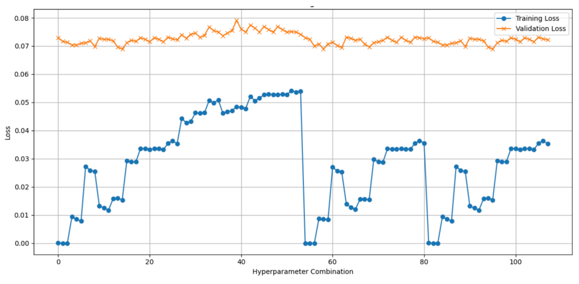
```
              precision    recall  f1-score   support

           0       0.92      0.93      0.92      1620
           1       0.84      0.83      0.83       749

    accuracy                           0.90      2369
   macro avg       0.88      0.88      0.88      2369
weighted avg       0.90      0.90      0.90      2369

[[1499  121]
 [ 127  622]]
```

Cross-Validation:
GridSearchCV typically performs k-fold cross-validation, where the dataset is split into k equal-sized folds, and each fold is used as a validation set once while the rest of the data is used for training. We create a GridSearchCV object, specifying the estimator, parameter grid, cross-validation strategy (cv), and the scoring metric. We call the fit method on the GridSearchCV object to perform the grid search. Finally, we retrieve the best parameters and the best score obtained during the grid search.

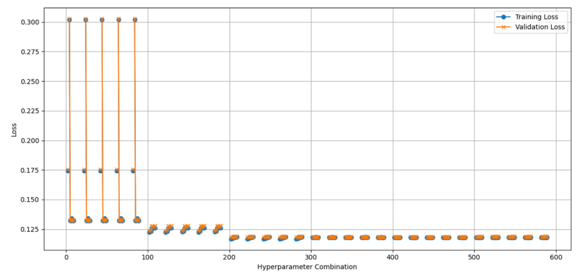
```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
Best Hyperparameters for Random Forest: {'max_depth': None, 'min_samples_leaf': 2,
'min_samples_split': 5, 'n_estimators': 200}
```

The Random Forest model demonstrates superior performance with an accuracy of 92.61%, outperforming both the Logistic Regression model with 88.35% accuracy and the Decision Tree model with 89.40% accuracy. Based on the results, the Random Forest model is recommended for further analysis or deployment due to its higher predictive accuracy.

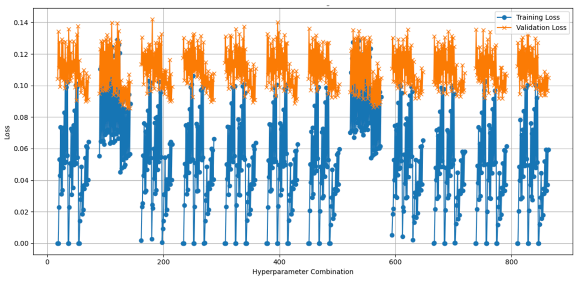
Training vs Validation loss for Random Forest



Training vs Validation loss for Logistic Regression



Training vs Validation loss for Decision Tree Classifier



Project link – https://github.com/tirumaleshn2458/CS584-Final-project/

5. Future work

In further extending this project, we are going to deploy this machine learning model into a web application which takes few details like home_team, away_team, batter_team, pitch etc as input and predicts whether a player will hit a home run in the next at-bat or not.

6. Conclusion

In conclusion, this project successfully tackled the challenge of predicting homerun in professional baseball games through comprehensive data exploration, cleaning, and modeling. By leveraging machine learning techniques such as Logistic Regression, Random Forest Classifier, and Decision Tree Classifier, along with ensemble methods like StackingClassifier, we achieved promising accuracy rates. The insights gained from this study offer valuable strategic implications for coaches, analysts, and enthusiasts, ultimately enhancing the excitement and decision-making process in the realm of baseball. The superior performance of the Random Forest model underscores its potential for real-world applications and further advancements in sports analytics.

7. References

[1] Huang, M.-L.; Li, Y.-Z. Use of Machine Learning and Deep Learning to Predict the Outcomes of Major League Baseball Matches. Appl. Sci. 2021, 11, 4499. https://doi.org/10.3390/app11104499

[2] Koseler, K., & Stephan, M. (2017). Machine Learning Applications in Baseball: A Systematic Literature Review. Applied Artificial Intelligence, 31(9–10), 745–763. https://doi.org/10.1080/08839514.2018.1442991

[3] sklearn Logistic Regression documentation - https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[4] sklearn Random Forest Classifier - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[5] sklearn Decision Tree Classifier - https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[6] sklearn Stacking Classifier - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html

[7] sklearn GridSearchCV - https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html