

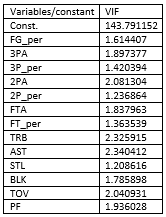
Dropped:

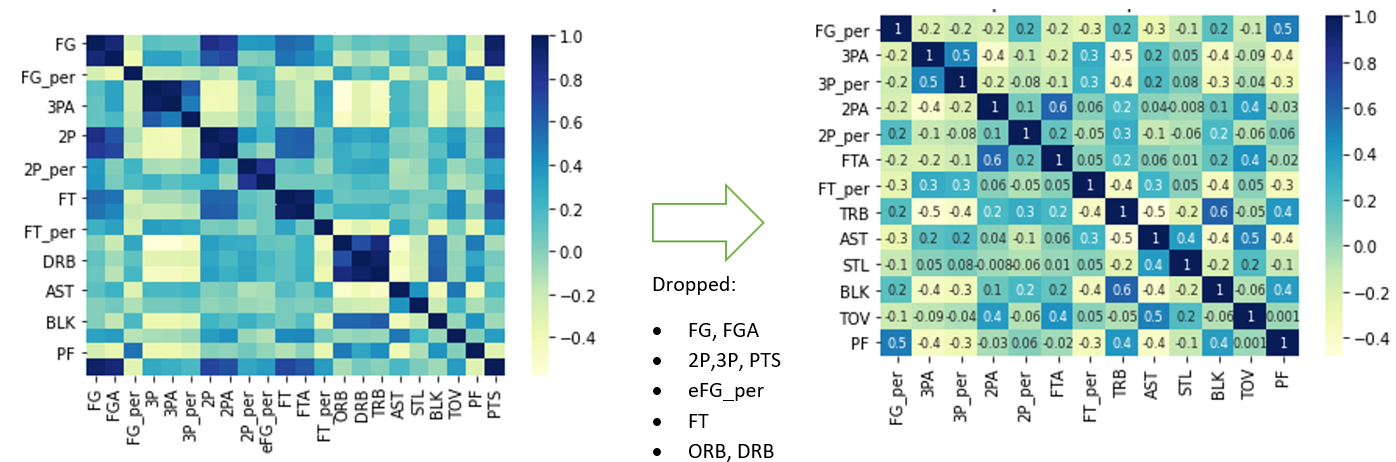
* FG, FGA
* 2P,3P, PTS
* eFG\_per
* FT
* ORB, DRB

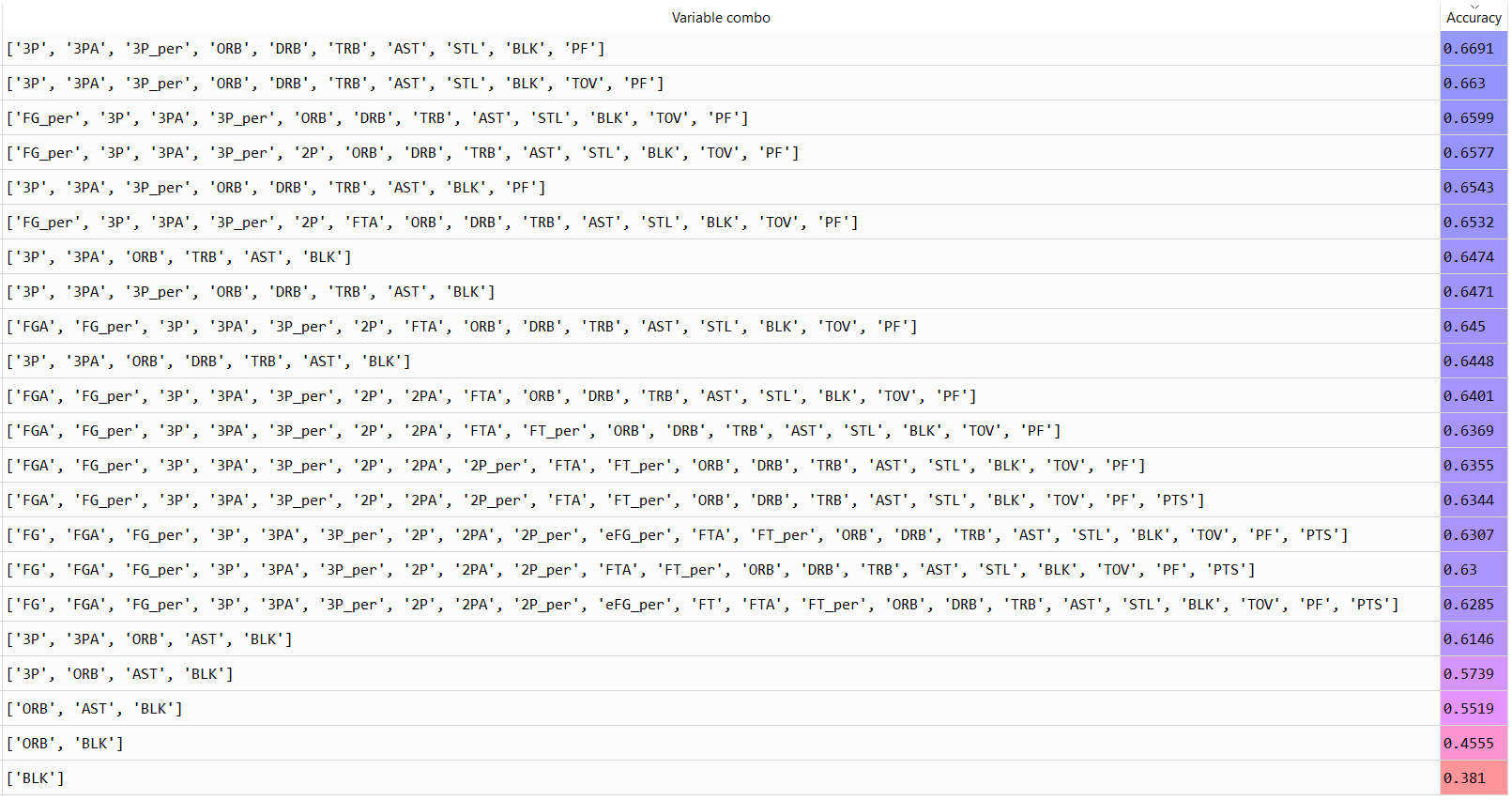
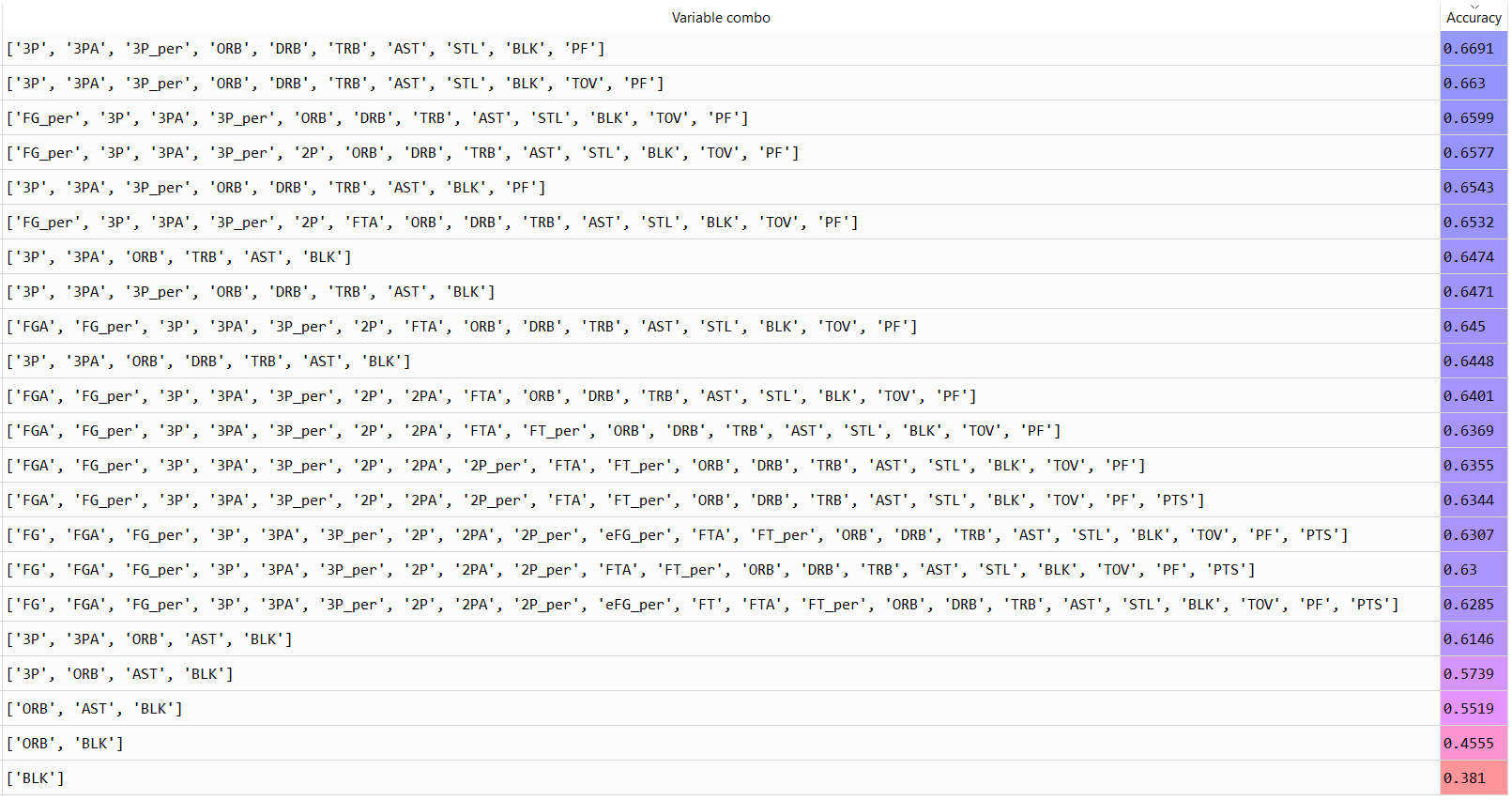
|  |  |
| --- | --- |
| Variables/constant | VIF |
| Const. | 143.791152 |
| FG\_per | 1.614407 |
| 3PA | 1.897377 |
| 3P\_per | 1.420394 |
| 2PA | 2.081304 |
| 2P\_per | 1.236864 |
| FTA | 1.837963 |
| FT\_per | 1.363539 |
| TRB | 2.325915 |
| AST | 2.340412 |
| STL | 1.208616 |
| BLK | 1.785898 |
| TOV | 2.040931 |
| PF | 1.936028 |



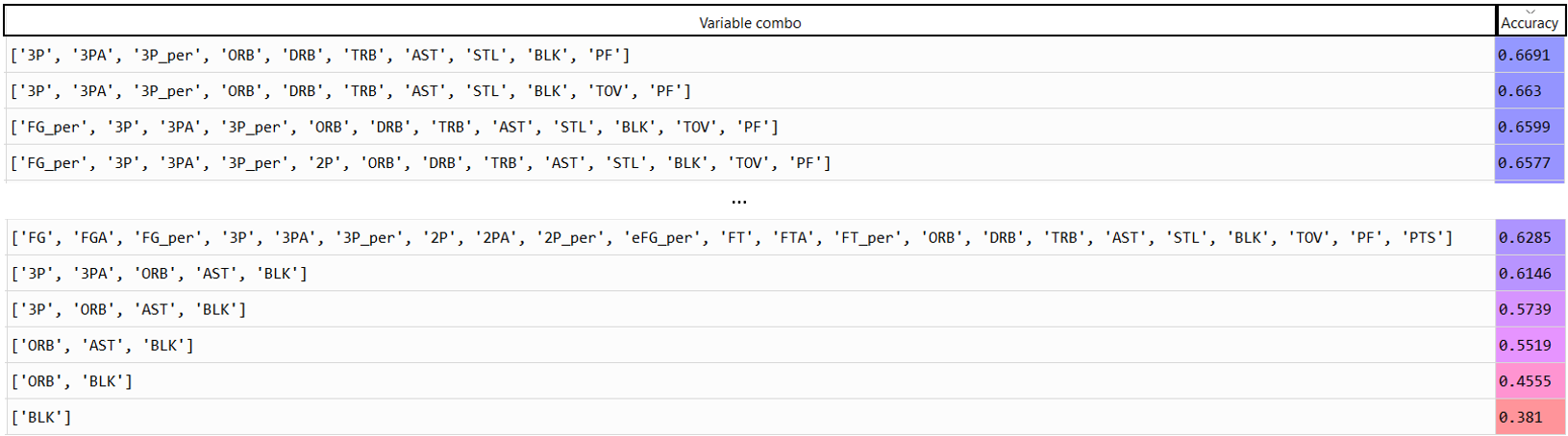




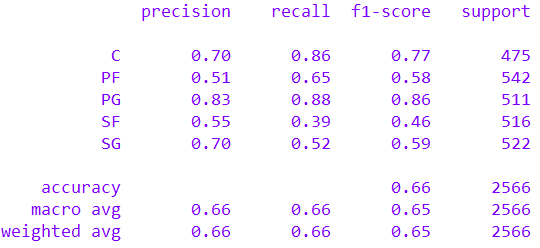
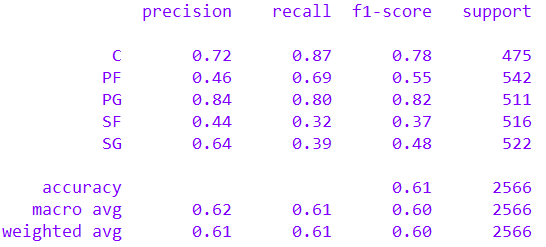




…



**Labels**



**KNN**

**Log\_reg**

**GNB**



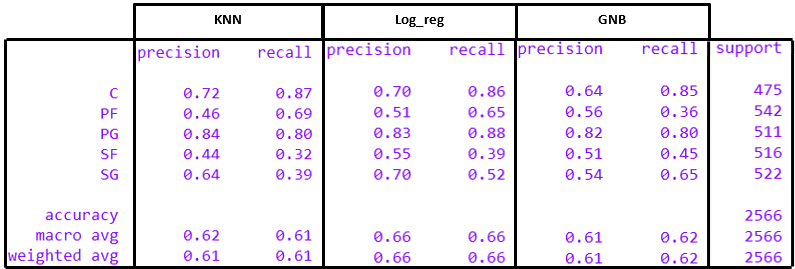
**Label Occurrence**

Figure : KNN

Figure : Log\_reg

Figure : GNB

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | KNN | Log\_reg | GNB |
| Accuracy | 0.61 | 0.66 | 0.62 |
| Weighted Precision | 0.61 | 0.66 | 0.61 |
| Weighted Recall | 0.61 | 0.66 | 0.62 |



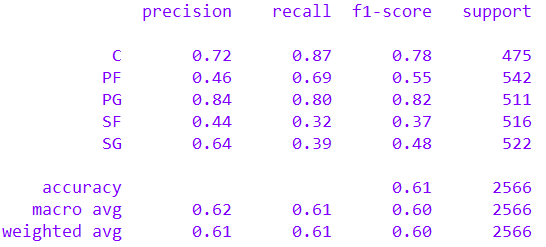


Figure : KNN

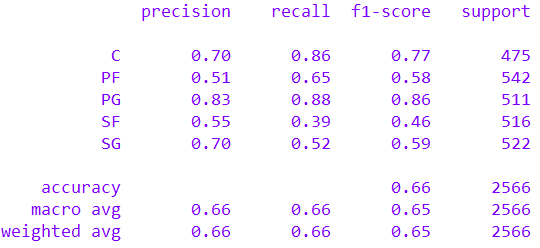
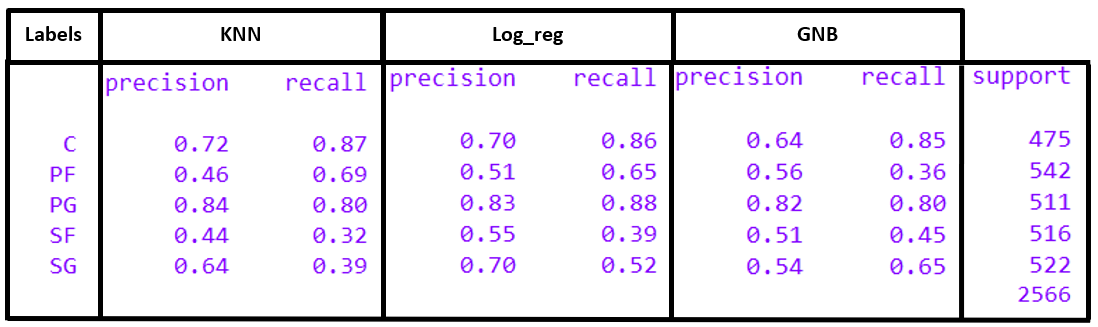
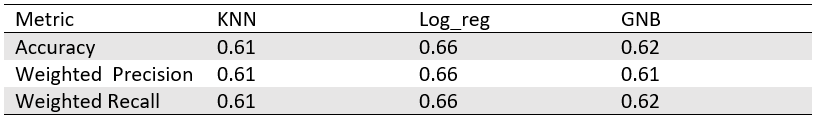


Figure : Log\_reg



Figure : GNB





OPEN FILE: ” Testing\_All\_Models.py” in

Ensemble\Final Model\_ Hard voting classifier

Choose your model from the available options

Show evaluation metrics (Classification Report, Confusion Matrix)?

How many player positions do you want to predict?

One

(**Recommended** )

More than one

**Line 14**

Set <multiple = False>

* F

**Line 14**

Set <multiple = True>

Yes

No

**Line 17**

Set <Eval1 = True>

In **line 61** reference a csv file containing per 36 min player statistics from the current directory. An example of this csv file in the current directory titled ‘df\_10.csv’).

e.g., if the csv file was called ‘df\_10’

set <f = ’ df\_10.csv'>

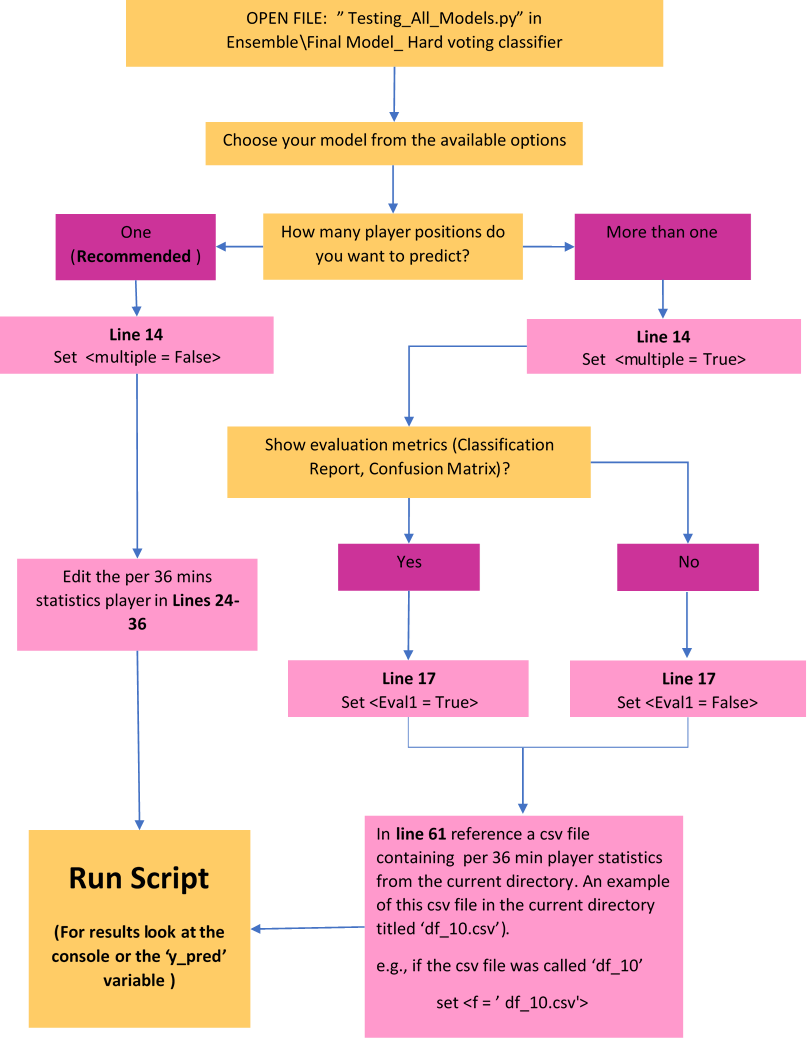
**Run Script**

**(For results look at the console or the ‘y\_pred’ variable )**

**Line 17**

Set <Eval1 = False>

Edit the per 36 mins statistics player in **Lines 24-36**



1. Shooting statistics

* **FG** = Field Goals made; **FGA** = Field Goal(s) Attempted; **FG\_per** = Proportion of field goals made
* **eFG\_per** = Effective proportion of field goals made
* **3P** = Three-point shots made; **3PA** = Three-point shots Attempted; **3P\_per** = Proportion of three-point shots attempted
* **2P** = Two-point shots made; **2PA** = …; **2P\_per** = …
* **FT** = Free Throw shots made; **FTA** = … ; **FT\_per** = …
* **PTS**  = points

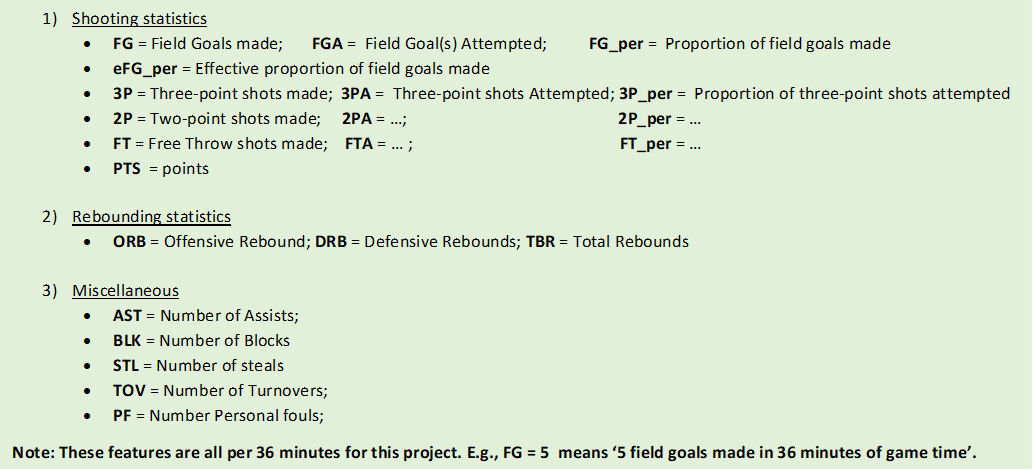
1. Rebounding statistics

* **ORB** = Offensive Rebound; **DRB** = Defensive Rebounds; **TBR** = Total Rebounds

1. Miscellaneous

* **AST** = Number of Assists;
* **BLK** = Number of Blocks
* **STL** = Number of steals
* **TOV** = Number of Turnovers;
* **PF** = Number Personal fouls;

**Note: These features are all per 36 minutes for this project. E.g., FG = 5 means ‘5 field goals made in 36 minutes of game time’.**



# Identifying-Player-Position

## Section 1: Project overview

- Created a tool that enables an NBA scout to identify an NBA player's position based on their per 36minutes statistics using 6 classification models.

- Data for this project was taken from basketballreference.com using the scraper developed from an earlier [Data Pipeline](https://github.com/favourumeh/DATA-PIPELINE) project

- This project uses three different classification algorithms: Naive Bayes, Logistic Regression and K Nearest Neighbours

- In total six models were created:

- three models used the individual algorithms mentioned above. They are called: 'KNN', 'log\_reg' and 'GNB'

- three more models were created using an ensemble approach. These ensemble models use hard voting with some minor tweaks which are explained in section 9. They are called: 'E\_hv1', 'E\_hv2' and 'E\_hv\_flex'

- All models were evaluated by comparing each other against three classification metrics : precision, recall and accuracy. Their confusion matrices were also analysed to gauge how well each model performed for different labels (/classes).

- This is a summary of the performance for each model:

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main//Evaluation%20images/Accuracy%2C%20Precision%2C%20Recall%20for%20all%20models%20for%20evaluation%20data.png)

- Feel free to try out the tool created by cloning the repository and opening the file: 'Testing\_All\_Models.py'. Follow the flowchart in the \*\*Section 12\*\* of this README for further guidance.

## Section 2: Python version and packages

Python Version: \*\*3.8.3\*\*

Packages: pandas, numpy, sklearn, matplotlib, seaborn, pickle

## Section 3: Project context

Before diving into the project details, it is important to understand the independent variables (features) and the dependent variable (classes/ labels).

### Dependent variable

- The dependent variable is player position (or 'Pos')

- It is made up of 5 classes:

- 'C' = Centre

- 'PF' = Power Forward

- 'PG' = Point Guard

- 'SF' = Small Forward

- 'SG' = Shooting Guard

### Independent Variables

- The independent variables are per 36 minutes player statistics.

- The ones considered for this project are

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/images%20dump/feature%20definitions.png)

- Go to the file \*\*'Features\_Explained.docx'\*\* for further information behind these features.

## Section 4: Data Splitting

The data used for this report was split into three sets: \*\*1) Training set(70%); 2) Testing set(20%) and 3) Evaluation set (10%)\*\*.

The non-ensemble models were trained and optimised using hyper parameter tuning and/or feature selection for the training set. As a result of this these models could be evaluated based on the testing and evaluations sets.

Making improvements to the ensemble model required the use of data different to the training set because KNN involves telling the model the correct prediction and the non-ensemble models used were already optimised for the training set. The testing set was, thus, used to 'train' the ensemble model.

All models were evaluated using the evaluation set as this data was totally unseen by both sets of models.

## Section 5: Feature Scaling

- Data inputted into KNN and Log\_reg models were scaled using a min-max scaler:

- KNN modelling required feature scaling because it involved the calculation of (Euclidean) distances between data points

- Log\_reg required scaling to speed up convergence of solvers used to find maximum likelihood (e.g. 'Newton-cg')

- Data inputted into the GNB model did not require feature scaling as this model works by determining the approximate gaussian distributions for each label for a given feature.

- The training, testing and evaluation datasets were all scaled separately using the same scaling procedure. This was done to avoid any passage of information between the different datasets.

## Section 6: Feature Selection

- At the beginning of this stage the features considered were as many as 22. Features: \*\*'FG', 'FGA', 'FG\_per', '3P', '3PA', '3P\_per', '2P', '2PA', '2P\_per',

'eFG\_per', 'FT', 'FTA', 'FT\_per', 'ORB', 'DRB', 'TRB', 'AST', 'STL',

'BLK', 'TOV', 'PF', 'PTS'\*\*

### Feature selection for Gaussian Naive Bayes and K-Nearest Neighbours

- As both models are sensitive to 'noisy' (irrelevant) features their tuning process required the detection and removal of any irrelevant features.

- The goal of the feature selection was to narrow down the 22 features to the ones that maximised the classification accuracy for a 5-fold cross validation of the training set. Accuracy was used as the judgement metric because the labels (/classes) -- whilst not equal-- were fairly balanced.

- This was done in 2 steps:

\*\*1) Graphical Assessment: Equivalent Gaussian distribution\*\*

- The mean and standard deviation of a feature for a given label was calculated.

- This was then used to plot an equivalent gaussian distribution to represent each label's feature distribution. Two examples of this type plot are shown below for the BLK and FT features :

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/KNN/final%20images/Spread%20of%20Labels%20for%20BLK%20and%20FT%20features.png)

- From the images above, it is evident that the distribution of the BLK feature for each label is more spread out than for the FT feature. This indicates that BLK is better at predicting a player's position than FT. Additionally, it suggests that the GNB and KNN model will perform better without the FT feature because each model's prediction will become less confident due to the inability of FT to differentiate the labels.

- Graphical assessments are a good starting point in deciding which features were obviously redundant. The next step will quantify how much better the models are without each feature using Relative Standard Deviation (RSD).

\*\*Note: The feature distribution for labels were not necessarily normal. The image above is a schematic to highlight the spread between the labels for a given feature.\*\*

\*\*2) Quantitative Assessment: Relative Standard Deviation(RSD)\*\*

- RSD was used as a measure of the relative spread of labels in a given feature. It is calculated by 1) Calculating the mean of a given feature for each label; 2) Calculating the average spread of the labels relative the mean of all label means for a given feature.

- Note: The 'average spread relative to the mean of all label means' was used to capture the spread as opposed as a simple mean spread calculation because a set of large numbers will naturally have higher spreads despite not actually being more spread out than a set of small numbers.

- Once the RSD was calculated for all features, the features were ordered based on it. A slice of the features with top 3 highest RSD and lowest RSD are shown below

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/Naive%20Bayes/final%20images/Relative%20standard%20deviation%20rankings.png)

- To narrow down the most important features a 5-fold cross validation was conducted using the training set. All 22 features were inputted into a model(i.e. KNN or GNB) and then the mean cross validation accuracy was calculated.

- The process was repeated again and again. With each new iteration the feature with the worst RSD (e.g. FT for the 1st iteration) was removed and the model's accuracy was recalculated. This was done until the model was left with one feature

- In the end the combination of features that produced the highest mean cross validation accuracy was deemed the most relevant. An example of the top 4 variable combinations with the highest accuracy and bottom 6 combination for KNN with number or neighbours = 5 is shown below

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/KNN/final%20images/KNN%20variable%20combo%20accuracy%20feature%20selection.png)

### Feature Selection for Logistic Regression

As Log\_reg is a regression model it had to follow certain assumptions such as: 1) No multicollinearity; 2) Exogeneity. Correlation plots and Variance Inflation Factor calculations were used to assess the multicollinearity. Variables with correlation coefficients >0.7 and VIF> 5 were removed.

\*\*Correlation Plot\*\*

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/Logistic%20Regression/final%20images/correlation\_feature\_selection.png)

## Section 7: Tuning and Hyperparameter Tuning

### Gaussian Naive Bayes

Due to the simplicity of this model the only tuning conducted was the removal of 'noisy' data which has already been explained in the Feature Selection section.

### K-Nearest Neighbours

As well as the removal of 'noisy' data, this model was tuned by adjusting the number of neighbours (k-value) and changing the weight function used to in the prediction from 'uniform' to 'distance'.

A uniform weight function simply chooses the modal label(/class) from the k neighbours. The distance weight function chosen for this model places more emphasis (weight) on labels that are closer to the queried point by setting the weight as the inverse (Euclidean) distance between the query point and the neighbours.

To gauge the effect of adjusting the K-value and weight function a 5-fold cross validation was conducted on the training data and the mean accuracy of the 5 models was calculated. The figure below shows the effect of the weight function used and k-value on the mean accuracy.

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/KNN/final%20images/tuning\_k.png)

From the figure above it is evident that a K-value = 24 and the distance weight function was the most optimal combination as it produced the highest mean cross validation accuracy.

### Logistic Regression

SKLearn's Logistic Regression model has built-in ridge and lasso regularisation. It also offers different solvers that determine the maximum likelihood. To save on time GridsearchCV was used to try different combinations of solvers, types of regularisation(ridge and lasso) and inverse regularisation strengths. Twenty inverse regularisation strength values were uniformly distributed in log space from 10^-4 to 10^4. The mean accuracy and mean log loss of a 5-fold cross-validation of the training set was used to gauge the optimal combination.

The optimal combination used the 'sag' solver, with ridge regularisation and an inverse regularisation strength of 206.913. The mean cross validation accuracy for this combination was 0.6991 and the log loss was 0.7512.

## Section 8: Evaluating KNN, log\_reg and GNB

As the \*\*test data\*\* was not used in the training or tuning of these models. It was used to compare the performance against Accuracy, Precision and Recall

### Broad Analysis on Accuracy, Precision and Recall

- The table below gives the mean accuracy and weighted Precision and Recall for each model

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/Evaluation%20images/Accuracy%2C%20Weighted%20precison%20and%20recall%20for%20non-ensemble%20models.png)

- This table below gives the Precision and Recall for each label and for each model

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/Evaluation%20images/Accuracy%20and%20Precision%20KNN%2C%20Log\_reg%2C%20GNB.png)

- Brief Comments:

- From the tables above, it is evident that the log\_reg model performs the best overall in terms of precision, accuracy and recall. It ranks in the top 2 for precision and recall for all labels and has the highest overall accuracy, weighted precision and weighted recall.

- All Models are good predictors of C and PG with precisions of at least 0.64. The recall scores for these labels were even better (at least 0.80). This is to be expected as the role played by these types of players are well-defined in the data chosen for this project

- All models struggled with predicting SF and PF labels as no model reached over 0.6 precision

- To understand why some labels, have better precision and recall than others it is worth analysing the confusion matrices for each model.

### Confusion Matrix Analysis

The figure below is the confusion matrices for all models:

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/Evaluation%20images/Confusion%20matrices%20for%20all%20non-ensemble%20models%20for%20testing%20data.png)

Brief Comments:

- The reason why KNN and Log\_reg predict the PF label so poorly is because it is confused for the SF label.

- GNB isn't as poor a predictor of the PF label as KNN and Log\_reg, but PF is noticeably underpredicted. This underprediction is caused by overprediction of the C and SF labels which has caused dismal PF recall (0.36). More PFs are classed as Cs by the GNB model than as PFs.

- SF recall was dismal across all models. Both KNN and log\_reg label more SFs as PFs than SFs. Despite having the highest SF recall out of all models more SFs are labelled as SGs and PFs than SFs for the GNB model.

### Contextual observation and Final comments:

Whilst not having amazing classification accuracy the models may have picked up on the fluidity of certain positions in basketball particularly SF and PF. It is this fluidity in positional play that caused lower recall scores for these positions. Additionally, a player's positional classification is not based on any objective metric rather a consensus amongst the player and their observers (coaches, scouts, audience etc). This consensus is typically based on who the player's positional play most resembles historically. However, not every player will fit into a particular box. It is possible for a player's positional play to be different to their identification and the models may capture this.

The following section will outline the creation of ensemble models which will establish different hard voting systems which will enable the models to come to an agreement on a player's position.

## Section 9: Ensemble Model Building

As there was an odd number of algorithms used for this project ensemble models (E\_hv1, E\_hv2 and E\_hv\_flex) were produced using the non-ensemble models (KNN, log\_reg and GNB) optimised for the training set. This was done with the belief that ensemble models would help generalise class prediction by inheriting the best traits of the individual models.

All the ensemble models use hard voting classification and were fed the test data. For this situation there were 3 possible voting scenarios:

- Scenario 1: Consensus Vote (54.8% of test set):

- 1A: All three non-ensemble models agree on the label and the label is correct (42.46% of test data)

- 1B: All three non-ensemble models agree on the label and the label is incorrect (12.34% ...)

- Scenario 2: Majority Vote(41.39% ...):

- 2A: Two non-ensemble models agree on a label and the majority decision is correct (21.09% ...)

- 2B: Two non-ensemble models agree on a label and the majority decision is incorrect (20.3% ...)

- Scenario 3:'Hung Parliament'--The non-ensemble models do not agree on a label (3.82% ...):

- 3A: 'Hung Parliament' and one of the non-ensemble models has chosen the correct label (3.55% ...)

- 3B: 'Hung Parliament' and none of the non-ensemble models has chosen the correct label (0.27% ...)

Scenario 1B and 3B were lost-cause scenarios as they are on the extreme ends of the voting spectrum. At one end the ensemble model is wrong, but highly confident in its classification whilst on the other end the model is wrong and completely unsure of its classification. For both scenarios there is no good resolution, however, it could be argued that if all models reach a consensus then the player's positional play may be different to their current identification. In this case, the ensemble model's classification is sound, and this player is miscategorised. However, there is no way of truly knowing if this is the case without a significant influx of relevant features and a retraining of the non-ensemble models. Scenario 3B is concerning, but very rare (only 0.27% of the test data fell under this scenario).

The E\_hv1 model was used as a baseline ensemble model whilst E\_hv2 and E\_hv\_flex models were 'trained' using the testing set to maximise accuracy. This was achieved by using information from their performance in the test set to guide their voting biases (see the subheadings below for more information). Due to this training the E\_hv2 and E\_hv\_flex models can only be evaluated using the evaluation set as this dataset is unseen by either model.

### E\_hv1

This model accepts the consensus and majority votes from scenario 1 and 2. For scenario 3 the model randomly chooses a non-ensemble model to decide.

\*\*Advantage:\*\* This voting system is fair

\*\*Disadvantage:\*\* The results are not repeatable or reproducible due to the random selection process for Scenario 3.

### E\_hv2

This model accepts the consensus and majority votes from scenario 1 and 2. For scenario 3 the non-ensemble model with the highest accuracy in this scenario for the testing set was chosen. The most accurate model for scenario 3 was log\_reg at 45%.

\*\*Advantage:\*\* This voting system will yield a higher accuracy than E\_hv1

\*\*Disadvantage:\*\* This model is biased towards the optimised log\_reg model for scenario 3.

### E\_hv\_flex

This model deviates slightly from hard voting principles. It accepts the consensus vote from scenario 1 and adopts the same voting for scenario 3 as E\_hv2. The majority vote for scenario 2 is upheld except for the situation where KNN and log\_reg predict a PF label whilst GNB predicts a SF label. In this situation GNB’s SF prediction is upheld because it was found that this prediction had a precision of 49.8% whilst the PF prediction by KNN and log\_reg had a precision of 34.2%. In doing this the model inherits GNB's superior recall for the SF label, but this comes with a trade-off as the model's PF recall reduces (see confusion matrices in the next section to see this trade-off). Nevertheless, the increased recall of the SF label should offset the reduced recall of the PF label.

\*\*Advantage:\*\* This voting system will yield a higher accuracy than E\_hv2

\*\*Disadvantage:\*\* This model is biased towards the optimised log\_reg model for scenario 3. It is also biased towards the optimised GNB model for scenario 2 (where KNN and log\_reg predict the PF label whilst GNB Predicts the SF).

## Section 10: Evaluating KNN, log\_reg and GNB, E\_hv1, E\_hv2, E\_hv\_flex

To evaluate all 6 models produced, the evaluation set was used as this data is unseen by all the models.

### Accuracy, Precision and Recall

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/Evaluation%20images/Accuracy%2C%20Precision%2C%20Recall%20for%20all%20models%20for%20evaluation%20data.png)

### Confusion Matrices

#### 1) Non-Ensemble Models

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/Evaluation%20images/Confusion%20matrices%20for%20all%20non-ensemble%20models%20for%20evaluation%20data.png)

#### 2) Ensemble Models

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/Evaluation%20images/Confusion%20matrices%20for%20all%20ensemble%20models%20for%20evaluation%20data.png)

### Evaluation

All non-ensemble models have low variance because the precision and recall scores are similar to those from the test dataset. Based on these findings, it can be assumed that the ensemble models will also have low variance.

As expected, E\_hv\_flex appears to have the best overall precision, recall and accuracy. However, it is still just 2% better off than the best overall non-ensemble model, log\_reg. Additionally, the confusion matrix for E\_hv\_flex and E\_hv2 shows that E\_hv\_flex has improved SF recall at the cost of PF recall.

There appears to be a limit to the precision, recall and accuracy that can be achieved by the current models. This limit may be caused by the features used in each of the optimised non-ensemble models. The current features could make it difficult for the non-ensemble models to differentiate the PF, SF and C classes. The purpose behind creating the ensemble models was the belief that they would inherit the good traits of the non-ensemble model, and this is evident in high precision and recall of the C and PG classes; but also evident are the poor distinction of the PF, SF and c classes.

To improve the precision and recall of the models, stronger feature engineering is required. The addition of new features that show clear separation of the classes should improve the precision and recall for all classes much more than further tuning or ensemble model tweaks.

## Section 11: Further work

The below sub-sections are suggestions for improvements that could be made to improve precision and recall for all classes.

### Improving precision and recall

This can be done by adding features which help to better distinguish the classes such as 'player height' or 'AST/TOV' ('Assist/Turnover'). Another possible feature is a score that takes into account the fact that the 'C' class was disproportionately affected by nulls in the '3P\_per' feature. Roughly 33% of the C label had a null value for this feature compared to 15% for PF, 2.3% for SF, 0.7% for PG and 0.6% for SG.

### Improve ensemble approach

A soft voting classifier may also prove better than a hard voting classifier for situations where the non-ensemble models have high uncertainty in a classification. However, for this to be fruitful the feature engineering suggestions need to be implemented.

## Section 12: How to use the tool developed

The tool developed allows the user to use all models created in this project (ensemble and non-ensemble) to predict the position of one or more players. The variable 'y\_pred' should give the prediction(s) made. When Identifying a single player, the prediction will also be printed in the console. When identifying more than one player the predictions will not be in the console instead there will be a classification report and confusion matrix.

Follow the flow chart below to use the tool which is in the file 'Testing\_All\_Models.py' in the folder [Final Model\_ Hard voting classifier](https://github.com/favourumeh/Identifying-Player-Position/tree/main/Ensemble/Final%20Model\_%20Hard%20voting%20classifier).

![](https://github.com/favourumeh/Identifying-Player-Position/blob/main/Ensemble/Final%20Model\_%20Hard%20voting%20classifier/How%20to%20use%20the%20tool.png)

# DATA PIPELINE

- Created a data pipeline that can scrape and clean tabular statistical data from BasketballReference.com and store it in a local MySQL server and/or Excel workbook.

- The pipeline was used to store over 16,000 rows of per-game statistics of players in the National Basketball Association (NBA) spanning 31 years (from 1990-2021)

- The stored data was then used in several machine learning projects namely regression and classification (see [Predicting Turnovers](https://github.com/favourumeh/Multiple\_Linear\_Regression---Predicting-Turnovers-) project and [Identifying Player Position](https://github.com/favourumeh/Identifying-Player-Position) project)

- The NBA is the top professional basketball league in North America. It was founded in 1947 and is currently home to the best basketball players in the world.

- Basketball Reference is an online depository of basketball statistics, game logs etc for different basketball leagues.

- Feel free to try out the pipeline by cloning the repository and opening the 'data\_pipeline\_bball\_reference\_to\_SQL\_or\_EXCEL.py' file (see the flowchart below for further details).

# Predicting Turnovers

# Multiple\_Linear\_Regression-- Predicting Turnovers

- For more in-depth analysis see the report ('Multiple Linear Regression\_with\_FTA.docx'). Otherwise, the rest of the README gives a summary of some of the actions undergone and the findings of the project.

- Please feel free to try out the models using the 'Try\_out\_Model.py' file in the 'Try Out Models' folder

## Project Overview

- In this project two multiple linear regression models were created to predict the turnovers committed by an NBA player per 36 minutes (TOV)

- The model results were validated against the linear regression assumptions and using:

- the following metrics: 1) Mean Absolute Error (MAE); 2) Root Mean Squared Error (RMSE); 3) R-squared (R2)

- train-test split and cross-validation

- Ultimately it was found that Two-point attempts per36(2PA), Free throw Attempts per36(FTA) and Assists per36(AST) were the only factors from the dataset that had a notable effect on the TOV:

- \*\*R2 = 0.46\*\* . This is low because the model does not factor a player's skill which is the primary predictor of TOV. The R2 suggests that there is a significant variance in skill level amongst players

- \*\*RMSE = 0.529\*\* . The value of this error is roughly 25% of the mean TOV observed.

- The models created can be used to assists NBA coaches in player development because the model acts as a predictor of a player's expected TOV. A player vastly outperforming their expected TOV (i.e. commits less TOV than expected) could indicate good ball retention skills which would mean that they require more on-ball possessions. It can also be used to pinpoint players who need more training in ball retention.

## Python version and packages

Python Version: 3.8.3

Packages: pandas, numpy, sklearn, statsmodles, matplotlib, seaborn, pickle

## Data used

Slice of table used

![alt text](https://github.com/favourumeh/Multiple\_Linear\_Regression---Predicting-Turnovers-/blob/main/Images/slice%20of%20data.png)

Glossary of headings

![](https://github.com/favourumeh/Multiple\_Linear\_Regression---Predicting-Turnovers-/blob/main/Images/glossary.png)

## Data cleaning, transformation and Feature engineering

- Nulls

- All instances of Null values were found in the percentages columns (i.e. 2P\_per, 3P\_per and FT\_per) and the cause of this was division by zero. All null elements were due to a player not attempting a particular shot (i.e 2PA, 3PA, FTA = 0). Therefore 2P\_per = 2P/2PA = nan.

- Instead of removing the rows that featured nulls the average shot attempts and shots made for each player position was calculated and assigned to the rows were 2PA, 3PA or FTA = 0. The percentage columns were then recalculated from this (e.g. 2P\_per = 2P/2PA).

- For example, if a player played the centre position and they had zero 3PA then the average 3PA and 3P for centres would be assigned to them.

- Early feature selection and engineering:

- Obvious instances of multicollinearity were removed (e.g. The shots made and shot attempted columns are highly correlated so any shots made columns such as 2P was removed)

- 'Minutes played' (MP) highly correlated with most of the other features but it could not be removed without violating Exogeneity assumption, so its effect was dampened by extrapolating all relevant statistics to per 36 minutes statistics

![](https://github.com/favourumeh/Multiple\_Linear\_Regression---Predicting-Turnovers-/blob/main/Images/Feature%20engineering%20and%20selection.png)

- correlation heatmaps and variance inflation factors were used to determine the not-so-obvious correlation amongst features

## Exploratory Data Analysis

Here is the distribution of the player position population used for the final model

![](https://github.com/favourumeh/Multiple\_Linear\_Regression---Predicting-Turnovers-/blob/main/Images/Bar%20chart%20for%20player%20position%20population.png)

These correlation plots show feature selection

![](https://github.com/favourumeh/Multiple\_Linear\_Regression---Predicting-Turnovers-/blob/main/Images/Correlation%20plot.png)

These 'lmplot' show how the relationship between TOV and AST changes for different player positions

![](https://github.com/favourumeh/Multiple\_Linear\_Regression---Predicting-Turnovers-/blob/main/Images/lmplot%20for%20TOV%20v%20AST.png)

## Model Building

The dependent variable was TOV

For model 1: Independent variables = 2PA, AST, FTA

For Model 2: Independent variables = 2PA, AST, FTA, Pos (Player Position)

Pos is a categorical variable, and it was added because I believed that Player position would introduce player skill to the model. I hypothesized that the position a player plays in would lend them to certain skillsets which would impact their likelihood of committing turnovers.

Pos was converted into a dummy variable before any modelling.

I split the data into 80-20 train and tests split.

I used Multiple Linear Regression for both models and evaluated them on the test set using the following metrics: Root Mean Squared, Mean Absolute Error and R-squared. I chose MAE as a baseline error metric and RMSE because it punishes extremely poor predictions. I chose R2 to gauge how well the addition of the Pos variable explains the variance in player skill.

To validate the metric results from the test dataset each metric was recalculated with 5-fold cross validation.

## Model Results and Performance

Here are the summary results for both models

![](https://github.com/favourumeh/Multiple\_Linear\_Regression---Predicting-Turnovers-/blob/main/Images/Model%20Summary%20result.png)

The figure shows the Predicted V Actual turnovers committed per 36minutes

![](https://github.com/favourumeh/Multiple\_Linear\_Regression---Predicting-Turnovers-/blob/main/Images/Model%201--%20Predicted%20V%20Actual%20TOV.png)