FINAL PROJECT REPORT

Prepared By: Ayyan Asif

Submitted May 4th, 2024

Prepared for

New Mexico State University

C S 519

**Table of Contents**

Problem Definitions……………………………………………………………………...... 3

The Data …………………… ……..…………………………………………………….. 3

Code and Methodology…..………………………………………………………………. 5

Conclusion ……………….. .…………………………………………………………..... 9

Appendix A: Full Prediction Results ..………………………………………..………... 11

Appendix B: Hyperparameter Tuning ..………………………………..……..………... 17

**Problem Definition**

**Objective**: Develop a machine learning model that accurately predicts the outcomes of Major League Soccermatches, including the final score, based on comprehensive datasets encompassing match conditions, team dynamics, and individual player performance metrics.

The potential value in such an algorithm are as follows:

* Providing sports analysts, betting companies, and soccer enthusiasts with a predictive insight into match outcomes, enhancing strategic betting, fan engagement, and broadcasting strategies.
* Offering teams and coaches data-driven insights into their performance, potential vulnerabilities, and the effectiveness of different strategies or player combinations.

**The Data**

*Overview*

European Soccer Database

Source: <https://www.kaggle.com/datasets/hugomathien/soccer/data>

Summary: A compilation of football data from 3 sources that covers 11 countries, 11 leagues, 25,979 matches, 11,060 players, and 299 teams. The original data is split into 7 tables with varying numbers of dimensions and instances. Record dates range from 2007 to 2016, though not all tables share the same date range.

*Details of each Dataset*

Country:

id: Unique country ID

name: Country name

League:

id: Unique league ID

country id: ID for associated country from “Country” table

name: Leage name

Match:

Columns 1-9: Preliminary data including team, league, country, and match IDs; date; and season.

Columns 10 & 11: Final score of home and away team goals.

Columns 12-55: Correspond to positions for each player for each team. A bit more investigation will be needed to determine if these will be of any use.

Columns 56-77: Player IDs for each player on both the home and away teams.

Columns 78-85: Appears to be XML data for each goal, shot, foul, card, possession, “cross”, and “corner” in the match. We plan to extract this data if possible.

Columns 86-115: Betting odds for a home team win, draw, and away team win for 10 different sports betting companies.

Player:

Contains player data with self-explanatory column headers including id, name, birthday, height, and weight.

Player Attributes:

Columns 1-4: Preliminary data including player ids and date of recorded instance.

Columns 5-42: Appears to contain player ratings on a scale of 1-100 for a variety of attributes including attacking work rate, short passing, acceleration, strength, interceptions, and many more. It should be noted that these ratings were obtained from the FIFA video game, and their accuracy to the real world is questionable. Part of our analysis should include these variables’ validity in our model.

Team:

Contains team data with self-explanatory columns headers for IDs, full name, and shortened name.

Team Attributes:

Columns 1-4: Team ID data and date of recorded instance.

Columns 5-25: Various team metrics, some being categorical and some being measurements on what appears to be a 1-100 scale. Like the player attributes table, this data is sourced from the FIFA videogame. Its validity will likewise need to be determined.

*Data Completeness*

Missing Values:

Country: No missing values

League: No missing values

Match: Several columns contain missing values, though none appear to exceed 16,216. There are 25,979 columns. Even if all rows containing missing values were to be deleted, enough data would remain for the project. However, preprocessing will hopefully keep many of these rows from needing to be removed.

Player: No missing values

Player Attributes: There are 183,978 rows. 30 columns are missing 836 values, 7 columns are missing 2,713 values, and 1 column is missing 6,869 values.

Team: 11 missing FIFA IDs

Team Attributes: 969 values out of 1,458 are missing for the “buildUpPlayDribbling” column. No other columns are missing any values.

Conclusion: There is enough data among each of the tables to be used in analysis. As much data will be kept as possible, so long as its retention does not degrade the quality of our model.

**Summary:**

Based on the initial analysis, this dataset is extremely comprehensive. It provides a variety of variables, a decent timeframe, and excellent consistency among the data. As expected, much cleaning and preprocessing must be done before it is ready to be analyzed. However, once done, it has the potential to be extremely valuable in this project, with numerous analytical paths available. It may also be combined with other datasets to further its completeness should it be beneficial. It should be noted that the player\_attributes and team\_attributes datasets appear to have data gathered from the FIFA soccer video game. It is questionable whether this data is completely accurate; however, given the desire of these video games to be realistic, it is reasonable to assume that they have some ground in

**Code & Methodology**

With our goal being to predict the number of fouls, yellow cards, red cards, and goals (and thereby the winner), it was decided that both classification and regression techniques be pursued. Regression was a natural choice as our prediction could be a range of numbers. However, because the number of goals, fouls, and flags tend to be within a reasonable range, treating each integer as a class was also a reasonable approach.

Our code and methodology follow an expected outline. Each dataset needed to be loaded in, cleaned, analyzed, combined, and reduced dimensionally. From there, multiple prediction algorithms were created and tested, with the results being recorded at the end of this report.

*Loading & Cleaning Match Data*

Our main dataset was the “match” data file which contained data from nearly 26,000 matches from 2008 to 2016. Many of these records were missing the vital pieces of data concerning goals, fouls, and flags which were contained in XML format. Rows missing this data were removed. Column features which had data missing in 10% or more of the rows or that were irrelevant were entirely removed.

Seeing as our data still contained a wealth of instances, rows that had any missing data were also removed. Lastly, because the “team\_attributes” dataset did not have records prior to February 22, 2010, all matches prior to this date were removed. This left us with over 11,000 rows, which was assumed to be more than enough for our analysis.

*Extracting XML Data*

Due to the inconsistency regarding the number of goals, fouls, and cards, this data was stored in XML format. Each datapoint contained within it the type of instance (foul, flag, penalty, shot, possession, etc.), the elapsed time, the team, the player, and other information. Our code went through each row and stored each attribute for each datapoint in a new column as a list.

For example, the “card” column had data on the type of card, the player, and the team each split into a new column with each occurrence stored in a list. The card type was stored in a datapoint with the value “[y, y, r, y]” (y = yellow, r = red). Once this was finished, the original XML columns were deleted.

*Creating New Features*

Seeing as team performance can be gauged based on prior performance, we decided to create new features that recorded the performance of teams from their past 10 matches. For cards, our code took each list of the cards per match and identified the card type and the team. This data was placed into a new column as it would be later used during prediction training. The same was done for fouls and goals.

Once finished, the code created a new column with the sum of each feature for the past 10 matches. It should be noted that our code only summed the data for the home team if they were also the home team for that prior match. The same was true for the away team. So, if a team was playing at their home field, the code only sums match data from their past 10 matches as the home team. This was believed to control for any variance caused by the home team advantage.

*Predicting Missing Data*

Besides the “match” dataset, the only other dataset that had missing data was the “player\_attributes” dataset. Because player performance statistics were to be part of our input variables, it was important that this be as complete as possible. It was decided that an algorithm be created that predicted the missing variables for each feature with missing data.

This was done by splitting the player\_attributes features between those that were missing some data and those that were complete. The features with missing values then had their rows split between those that were missing the data and those that were not. The ones without missing data were to be used in algorithm training.

For each feature, the code created a linear regression, RANSAC, ridge, elastic net, and random forest regression models. Each model was evaluated by its mean-squared-error and r-squared. The algorithm with the highest r-squared on the test data was used for the prediction. If there was a tie, it was broken in the following order: test mean-squared-error, train r-squared, train mean-squared-error. Below is an example of the results of each algorithm for the curve feature:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Train R^2** | **Train MSE** | **Test R^2** | **Train MSE** |
| Linear Regression | 0.79 | 6.31 | 0.79 | 6.32 |
| RANSAC | 0.79 | 6.27 | 0.79 | 6.27 |
| Ridge | 0.79 | 6.31 | 0.79 | 6.32 |
| Elastic Net | 0.79 | 6.33 | 0.79 | 6.33 |
| Random Forest | 0.99 | 0.75 | 0.95 | 1.99 |

From this example, the random forest model performed the best and was used in the prediction. Below is a table showing the results for the best algorithm for each feature:

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Algorithm** | **Test R^2** | **Test MSE** |
| volleys | Random Forest | 0.96 | 1.91 |
| curve | Random Forest | 0.95 | 1.99 |
| agility | Random Forest | 0.93 | 1.87 |
| balance | Random Forest | 0.89 | 2.33 |
| jumping | Random Forest | 0.86 | 2.31 |
| vision | Random Forest | 0.94 | 2.01 |
| sliding\_tackle | Random Forest | 0.98 | 1.46 |

Once the best algorithm was identified, each missing value was predicted and placed into the data. This completed the data cleaning as no other datasets were missing data. All that was left in the preprocessing phase was to combine the datasets into one.

*Combining Datasets*

The goal in this step was to add all the relevant data from each dataset onto the “match” dataset. First, the players on each team were identified. The most recent performance statistics for each player from “player\_attributes” were added onto the team data in “team\_attributes”. The team and player attributes were then added to the “match” dataset, creating one table with very high dimensionality. The goal in this step was to only use the most recent statistics from the perspective of each match. However, an error in our code made it so that the overall most recent statistics were used, even if those statistics were gathered after the match occurred. Unfortunately, there was not enough time to fix this mistake prior to the deadline of the project.

The resulting table was one that had match data, team data, and player data in each row. There were 869 features. This extremely high dimensionality made it so that dimensionality reduction was necessary.

*Reducing Dimensionality*

Three dimensionality reduction techniques were tested, with the highest being linear discriminant analysis. The three techniques were PCA, LDA, and t-SNE. Each was tested with the data using a simple perceptron classification algorithm. The results were as follows:

|  |  |  |
| --- | --- | --- |
| **Technique** | **Train Accuracy** | **Test Accuracy** |
| PCA | 0.277 | 0.276 |
| LDA | 0.351 | 0.349 |
| t-SNE | 0.229 | 0.230 |

In seeking to maximize the variance captured by the dimensions, the number of components for LDA was set to be the number of “classes” or unique values in either the goals, cards, or foul categories minus 1. This would give the LDA the maximum number of dimensions possible to work with.

*Prediction:*

In seeking to identify the best prediction algorithm, both classification and regression algorithms were created using both linear and nonlinear models. Given the straightforward nature of sport performance, it was expected that linear models would perform the best. However, the extraordinarily high dimensionality of our data caused uncertainty as to which methods would perform best. As such, the code is set to create and analyze multiple algorithms for each prediction and display their results.

The algorithms created and used were K-nearest-neighbors, random forest classifier, SVM, random forest regressor, and linear regression. The classification algorithms were evaluated based on their accuracy. The regression algorithms were evaluated using their r-squared, mean-squared-error, and prediction accuracy. The regression prediction accuracy was found by rounding the prediction results to the nearest integer and comparing them to the correct values.

Of the eight predictions (home team goals, away team goals, home team yellow cards, away team yellow cards, home team red cards, away team red cards, home team fouls, and away team fouls) only two had an acceptable test accuracy rate. The full results from the prediction of home team fouls are as follows:

*Classification*

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Train Accuracy** | **Test Accuracy** |
| KNN | 0.541 | 0.484 |
| Random Forest | 1.0 | 0.539 |
| SVM | 0.625 | 0.570 |

*Regression*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Test Accuracy** | **Test R^2** | **Test MSE** | **Train Accuracy** | **Train R^2** | **Train MSE** |
| Random Forest | 0.247 | 0.58 | 3.10 | 0.403 | 0.93 | 1.19 |
| Linear Regression | 0.093 | 0.54 | 3.61 | 0.089 | 0.53 | 3.72 |

The best algorithm for each test and its results are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Algorithm** | **Test Accuracy** | **Train Accuracy** |
| Home Team Goals | SVM | 0.445 | 0.440 |
| Away Team Goals | SVM | 0.446 | 0.445 |
| Home Team Yellow Flags | SVM | 0.421 | 0.421 |
| Away Team Yellow Flags | SVM | 0.408 | 0.394 |
| Home Team Red Flags | KNN | 0.951 | 0.948 |
| Away Team Red Flags | SVM | 0.940 | 0.936 |
| Home Team Fouls | SVM | 0.570 | 0.625 |
| Away Team Fouls | SVM | 0.574 | 0.628 |

As is clear from the results, SVM performs the best amongst each of the algorithms. The full results are available in Appendix A. The parameters for this SVM algorithm are as follows: svm = SVC(kernel = 'linear', random\_state=1, C=1). Every available kernel was tested and “linear” was found to be the best (the full results from the hyperparameter tuning are available in Appendix B).

**Conclusion**

It was expected that our problem would be linear in nature. The theory was that, as player and team attributes improved, so would their performance and vice versa. This may explain why the “linear” kernel yielded the best results. However, our results were not optimal for prediction, many of which did not even cross the 50% accuracy threshold.

It is quite likely that, though our problem is linear in nature, the team and player attributes are not a good enough indicator of team performance. There are too many other variables that are not accounted for in our model. This may make sense as, at the professional level, every player plays near the peak of what is physically possible. Many influential variables then are either outside player performance or are largely based on “luck”.

A good indicator that this is the case is the slack variable, C, being so low. Though it can create a linear model that captures the data best, it must be willing to include a great number of misclassifications to make it happen. This further solidifies the theory that our model fails to capture all the variables influencing team performance.

*Further Improvements*

For further improvement upon this model, three suggestions are to be made, all of which deal with the data. First, the code should be fixed so that the team and player attributes most recent from the date of the match are appended to the data rather than the overall most recent attributes. This error in our code caused matches from 2012 to have team and player data from 2016. How much of an impact this had on our prediction cannot be determined, though it is safe to assume that the results are inaccurate as a result.

Secondly, the player and team attribute data appear to have been gathered from a database of the FIFA video game. Given the desire for realism in video games, it is likely that these statistics have some ground in reality, but they may not be entirely accurate. Finding a better source of data that measures real-time player performance would be a much more reliable indicator.

Lastly, as mentioned earlier, there are likely many variables not included in our dataset that influence team performance. This may include coaching staff statistics, weather, player injuries, and the recency of prior matches. Because professional players perform at such a high level, these other factors may heavily influence the outcome of a match.

Overall, this project appears to have been successful in cleaning, filling, analyzing, and predicting, even if the predictions weren’t as accurate as desired. The highlight of the project is likely the portion of code that identifies columns with missing data in the player\_attributes dataset, creates multiple prediction algorithms, and fills in the missing data with the best prediction. With improvements in the data collected and the correction of the code that combines the tables, we are optimistic about significantly improving the results.

**Appendix A: Full Prediction Results**

Choose Prediction: Goals = 1, Yellow Flags = 2, Red Flags = 3, Fouls = 4

Enter here: 4

Choose team: Home = 1, Away = 2

Enter here: 1

Reducing Dimensions Using LDA...

Train Accuracy KNN: 0.541

Test Accuracy KNN: 0.484

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.539

Train Accuracy SVM: 0.625

Test Accuracy SVM: 0.570

Forest Regressor MSE train: 1.19, test: 3.10

Forest Regressor R^2 train: 0.93, test: 0.58

Train Accuracy Forest Regressor: 0.403

Test Accuracy Forest Regressor: 0.247

Linear Regression MSE train: 3.72, test: 3.61

Linear Regression R^2 train: 0.53, test: 0.54

Train Accuracy Linear Regression: 0.089

Test Accuracy Linear Regression: 0.093

RANSAC MSE train: 3.65, test: 3.53

RANSAC R^2 train: 0.51, test: 0.53

Train Accuracy RANSAC: 0.109

Test Accuracy RANSAC: 0.106

Choose Prediction: Goals = 1, Yellow Flags = 2, Red Flags = 3, Fouls = 4

Enter here: 4

Choose team: Home = 1, Away = 2

Enter here: 2

Reducing Dimensions Using LDA...

Train Accuracy KNN: 0.542

Test Accuracy KNN: 0.479

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.528

Train Accuracy SVM: 0.628

Test Accuracy SVM: 0.574

Forest Regressor MSE train: 1.24, test: 3.24

Forest Regressor R^2 train: 0.93, test: 0.58

Train Accuracy Forest Regressor: 0.387

Test Accuracy Forest Regressor: 0.254

Linear Regression MSE train: 3.85, test: 3.78

Linear Regression R^2 train: 0.52, test: 0.54

Train Accuracy Linear Regression: 0.094

Test Accuracy Linear Regression: 0.092

RANSAC MSE train: 3.78, test: 3.68

RANSAC R^2 train: 0.51, test: 0.53

Train Accuracy RANSAC: 0.103

Test Accuracy RANSAC: 0.109

Choose Prediction: Goals = 1, Yellow Flags = 2, Red Flags = 3, Fouls = 4

Enter here: 1

Choose team: Home = 1, Away = 2

Enter here: 1

Reducing Dimensions Using LDA...

Train Accuracy KNN: 0.469

Test Accuracy KNN: 0.374

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.418

Train Accuracy SVM: 0.440

Test Accuracy SVM: 0.445

Forest Regressor MSE train: 0.35, test: 0.91

Forest Regressor R^2 train: 0.88, test: 0.23

Train Accuracy Forest Regressor: 0.764

Test Accuracy Forest Regressor: 0.338

Linear Regression MSE train: 0.91, test: 0.91

Linear Regression R^2 train: 0.22, test: 0.24

Train Accuracy Linear Regression: 0.337

Test Accuracy Linear Regression: 0.335

RANSAC MSE train: 0.90, test: 0.90

RANSAC R^2 train: 0.21, test: 0.24

Train Accuracy RANSAC: 0.350

Test Accuracy RANSAC: 0.343

Combining Tables & Cleaning Data...

Choose Prediction: Goals = 1, Yellow Flags = 2, Red Flags = 3, Fouls = 4

Enter here: 1

Choose team: Home = 1, Away = 2

Enter here: 2

Reducing Dimensions Using LDA...

Train Accuracy KNN: 0.500

Test Accuracy KNN: 0.407

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.445

Train Accuracy SVM: 0.455

Test Accuracy SVM: 0.446

Forest Regressor MSE train: 0.32, test: 0.83

Forest Regressor R^2 train: 0.87, test: 0.18

Train Accuracy Forest Regressor: 0.809

Test Accuracy Forest Regressor: 0.361

Linear Regression MSE train: 0.82, test: 0.82

Linear Regression R^2 train: 0.18, test: 0.18

Train Accuracy Linear Regression: 0.359

Test Accuracy Linear Regression: 0.367

RANSAC MSE train: 0.81, test: 0.81

RANSAC R^2 train: 0.16, test: 0.16

Train Accuracy RANSAC: 0.382

Test Accuracy RANSAC: 0.386

Combining Tables & Cleaning Data...

Choose Prediction: Goals = 1, Yellow Flags = 2, Red Flags = 3, Fouls = 4

Enter here: 2

Choose team: Home = 1, Away = 2

Enter here: 1

Reducing Dimensions Using LDA...

Train Accuracy KNN: 0.450

Test Accuracy KNN: 0.356

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.404

Train Accuracy SVM: 0.421

Test Accuracy SVM: 0.421

Forest Regressor MSE train: 0.37, test: 0.98

Forest Regressor R^2 train: 0.88, test: 0.21

Train Accuracy Forest Regressor: 0.731

Test Accuracy Forest Regressor: 0.307

Linear Regression MSE train: 0.98, test: 0.97

Linear Regression R^2 train: 0.21, test: 0.23

Train Accuracy Linear Regression: 0.300

Test Accuracy Linear Regression: 0.318

RANSAC MSE train: 0.97, test: 0.96

RANSAC R^2 train: 0.21, test: 0.23

Train Accuracy RANSAC: 0.317

Test Accuracy RANSAC: 0.323

Combining Tables & Cleaning Data...

Choose Prediction: Goals = 1, Yellow Flags = 2, Red Flags = 3, Fouls = 4

Enter here: 2

Choose team: Home = 1, Away = 2

Enter here: 2

Reducing Dimensions Using LDA...

Train Accuracy KNN: 0.435

Test Accuracy KNN: 0.334

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.390

Train Accuracy SVM: 0.394

Test Accuracy SVM: 0.408

Forest Regressor MSE train: 0.41, test: 1.04

Forest Regressor R^2 train: 0.87, test: 0.18

Train Accuracy Forest Regressor: 0.685

Test Accuracy Forest Regressor: 0.300

Linear Regression MSE train: 1.06, test: 1.04

Linear Regression R^2 train: 0.18, test: 0.18

Train Accuracy Linear Regression: 0.289

Test Accuracy Linear Regression: 0.287

RANSAC MSE train: 1.06, test: 1.03

RANSAC R^2 train: 0.18, test: 0.19

Train Accuracy RANSAC: 0.297

Test Accuracy RANSAC: 0.294

Choose Prediction: Goals = 1, Yellow Flags = 2, Red Flags = 3, Fouls = 4

Enter here: 3

Choose team: Home = 1, Away = 2

Enter here: 1

Reducing Dimensions Using LDA...

Train Accuracy KNN: 0.948

Test Accuracy KNN: 0.951

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.950

Train Accuracy SVM: 0.949

Test Accuracy SVM: 0.950

Forest Regressor MSE train: 0.03, test: 0.09

Forest Regressor R^2 train: 0.86, test: 0.08

Train Accuracy Forest Regressor: 0.998

Test Accuracy Forest Regressor: 0.944

Linear Regression MSE train: 0.11, test: 0.11

Linear Regression R^2 train: 0.08, test: 0.10

Train Accuracy Linear Regression: 0.947

Test Accuracy Linear Regression: 0.949

Choose Prediction: Goals = 1, Yellow Flags = 2, Red Flags = 3, Fouls = 4

Enter here: 3

Choose team: Home = 1, Away = 2

Enter here: 2

Reducing Dimensions Using LDA...

Train Accuracy KNN: 0.937

Test Accuracy KNN: 0.936

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.935

Train Accuracy SVM: 0.936

Test Accuracy SVM: 0.940

Forest Regressor MSE train: 0.04, test: 0.11

Forest Regressor R^2 train: 0.86, test: 0.07

Train Accuracy Forest Regressor: 0.995

Test Accuracy Forest Regressor: 0.930

Linear Regression MSE train: 0.13, test: 0.13

Linear Regression R^2 train: 0.08, test: 0.08

Train Accuracy Linear Regression: 0.934

Test Accuracy Linear Regression: 0.937

**Appendix B: Hyperparameter Tuning**

knn = KNeighborsClassifier(n\_neighbors=5, p = 1, metric='minkowski')

forest\_c = RandomForestClassifier(n\_estimators=25, random\_state=1, n\_jobs=2)

forest\_r = RandomForestRegressor(n\_estimators=100, random\_state=1, n\_jobs=-1, criterion='squared\_error')

lr = LinearRegression()

ransac = RANSACRegressor(LinearRegression(), max\_trials=100, min\_samples=0.95, residual\_threshold=None, random\_state=1)

Train Accuracy KNN: 0.595

Test Accuracy KNN: 0.468

Train Accuracy Forest Classifier: 0.999

Test Accuracy Forest Classifier: 0.495

Forest Regressor MSE train: 1.17, test: 3.10

Forest Regressor R^2 train: 0.94, test: 0.58

Train Accuracy Forest Regressor: 0.392

Test Accuracy Forest Regressor: 0.242

Linear Regression MSE train: 3.73, test: 3.61

Linear Regression R^2 train: 0.52, test: 0.54

Train Accuracy Linear Regression: 0.088

Test Accuracy Linear Regression: 0.093

RANSAC MSE train: 3.65, test: 3.52

RANSAC R^2 train: 0.51, test: 0.53

Train Accuracy RANSAC: 0.112

Test Accuracy RANSAC: 0.108

knn = KNeighborsClassifier(n\_neighbors=8, p = 1, metric='minkowski')

forest\_c = RandomForestClassifier(n\_estimators=250, random\_state=1, n\_jobs=-1)

forest\_r = RandomForestRegressor(n\_estimators=250, random\_state=1, n\_jobs=-1, criterion='squared\_error')

lr = LinearRegression()

ransac = RANSACRegressor(LinearRegression(), max\_trials=100, min\_samples=0.95, residual\_threshold=None, random\_state=1)

Train Accuracy KNN: 0.578

Test Accuracy KNN: 0.478

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.529

Forest Regressor MSE train: 1.16, test: 3.10

Forest Regressor R^2 train: 0.94, test: 0.58

Train Accuracy Forest Regressor: 0.394

Test Accuracy Forest Regressor: 0.238

Linear Regression MSE train: 3.73, test: 3.61

Linear Regression R^2 train: 0.52, test: 0.54

Train Accuracy Linear Regression: 0.088

Test Accuracy Linear Regression: 0.093

RANSAC MSE train: 3.65, test: 3.52

RANSAC R^2 train: 0.51, test: 0.53

Train Accuracy RANSAC: 0.112

Test Accuracy RANSAC: 0.108

knn = KNeighborsClassifier(n\_neighbors=12, p = 1, metric='minkowski')

forest\_c = RandomForestClassifier(n\_estimators=500, random\_state=1, n\_jobs=-1)

Train Accuracy KNN: 0.550

Test Accuracy KNN: 0.483

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.535

knn = KNeighborsClassifier(n\_neighbors=50, p = 1, metric='minkowski')

forest\_c = RandomForestClassifier(n\_estimators=1000, random\_state=1, n\_jobs=-1)

Train Accuracy KNN: 0.492

Test Accuracy KNN: 0.468

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.536

knn = KNeighborsClassifier(n\_neighbors=200, p = 1, metric='minkowski')

forest\_c = RandomForestClassifier(n\_estimators=5000, random\_state=1, n\_jobs=-1)

Train Accuracy KNN: 0.431

Test Accuracy KNN: 0.429

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.537

knn = KNeighborsClassifier(n\_neighbors=20, p = 1, metric='minkowski')

forest\_c = RandomForestClassifier(n\_estimators=500, random\_state=1, n\_jobs=-1)

Train Accuracy KNN: 0.529

Test Accuracy KNN: 0.481

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.535

knn = KNeighborsClassifier(n\_neighbors=18, p = 1, metric='minkowski')

Train Accuracy KNN: 0.534

Test Accuracy KNN: 0.483

knn = KNeighborsClassifier(n\_neighbors=15, p = 1, metric='minkowski')

Train Accuracy KNN: 0.543

Test Accuracy KNN: 0.484

Below are results with the best of the above parameters but with the number of dimensions in LDA reduced from len(unique(value)) -1 to len(unique(value)) – 3

Train Accuracy KNN: 0.541

Test Accuracy KNN: 0.485

Train Accuracy Forest Classifier: 1.000

Test Accuracy Forest Classifier: 0.536

Forest Regressor MSE train: 1.21, test: 3.15

Forest Regressor R^2 train: 0.93, test: 0.56

Train Accuracy Forest Regressor: 0.389

Test Accuracy Forest Regressor: 0.238

Linear Regression MSE train: 3.73, test: 3.61

Linear Regression R^2 train: 0.52, test: 0.54

Train Accuracy Linear Regression: 0.089

Test Accuracy Linear Regression: 0.094

RANSAC MSE train: 3.65, test: 3.52

RANSAC R^2 train: 0.51, test: 0.53

Train Accuracy RANSAC: 0.111

Test Accuracy RANSAC: 0.108

**SVM Tuning**

svm = SVC(kernel = 'rbf', random\_state=1, gamma=1.0, C=10)

Train Accuracy SVM: 1.000

Test Accuracy SVM: 0.403

svm = SVC(kernel = 'linear', random\_state=1, gamma=1.0, C=10)

Train Accuracy SVM: 0.627

Test Accuracy SVM: 0.568

svm = SVC(kernel = 'poly', random\_state=1, gamma=1.0, C=10)

Train Accuracy SVM: 1.000

Test Accuracy SVM: 0.521

svm = SVC(kernel = 'sigmoid', random\_state=1, gamma=1.0, C=10)

Train Accuracy SVM: 0.413

Test Accuracy SVM: 0.482

Changing C now…

svm = SVC(kernel = 'linear', random\_state=1, gamma=1.0, C=20)

Train Accuracy SVM: 0.628

Test Accuracy SVM: 0.569

svm = SVC(kernel = 'linear', random\_state=1, gamma=1.0, C=5)

Train Accuracy SVM: 0.626

Test Accuracy SVM: 0.568

svm = SVC(kernel = 'linear', random\_state=1, gamma=1.0, C=1)

Train Accuracy SVM: 0.625

Test Accuracy SVM: 0.570

svm = SVC(kernel = 'linear', random\_state=1, gamma=1.0, C=0.5)

Train Accuracy SVM: 0.622

Test Accuracy SVM: 0.570

svm = SVC(kernel = 'linear', random\_state=1, gamma=1.0, C=100)

Train Accuracy SVM: 0.627

Test Accuracy SVM: 0.568

svm = SVC(kernel = 'linear', random\_state=1, C=1, shrinking=False)

Train Accuracy SVM: 0.625

Test Accuracy SVM: 0.569

svm = SVC(kernel = 'linear', random\_state=1, C=1, class\_weight='balanced')

Train Accuracy SVM: 0.566

Test Accuracy SVM: 0.504