Date: 20th January 2021
Word count: 1208

Course Teacher:
Dr Luca Citi

Comparative Study of a Machine Learning System for a Practical Problem

Syed Omar Faruk (2003385)

Contents

1 Comparative Study

1.1 Investigate the performance

I have got the data CE802_P2_Data.csv contained in the CE802_P2_Data.zip archive and I am going to perform a number of tasks with the dataset and see the accuracy of the claim.

- Create a Decision Tree Classier
- Create a K-Nearest Neighbour Classifier
- Create a Naive Bayes Classifier

- Create a Support Vector Machine Classifier and
- Create a Multi-layer Perceptron Classifier

As of the first look at the data, I can see that there are missing features that I also need to deal with first. (Alpaydin, 2020) The F15 column has a number of instances missing. For those missing values what I can do are as follow:

- I will omit the column F15 as said on the guideline and continue with the rest of the columns (F1 to F14).
- I will fill the blanks of F15 with the mean values of the column.
- I will fill the blanks of F15 with zero (0)

1.2 Data Pre-Processing

For the data pre-processing part I saw the data was already clean. The only problem I had was the empty cells on the F15 column. So, I decided to do the above techniques to get the values to fill these empty cells. To omit out column F15 I excluded the column when I created the test and train data sets.

```
train_data_x = data.iloc[:,:14]
train_data_y = data.iloc[:, 15]
```

To fill the blank cells with the mean values I used:

```
data['F15'].fillna(data['F15'].mean())
data = data.fillna(data.mean())
```

To fill the blank with zero I used:

```
data = data.fillna(0)
```

2 Classification Performance Benchmark

2.1 Performance of Decision Tree

For the decision tree I imported `tree` from `sklearn`. Then I created a classifier object `DTCO` and fit it with the split data. Then predicted the output using `DTCO.predict(x_test)`. For Decision Tree, I got the following values of accuracy and kappa:

When I removed the entire F15 column:

```
Accuracy: 0.7066666666666667
Kappa Statistics: 0.41018766756032166
```

Filling F15 column with mean:

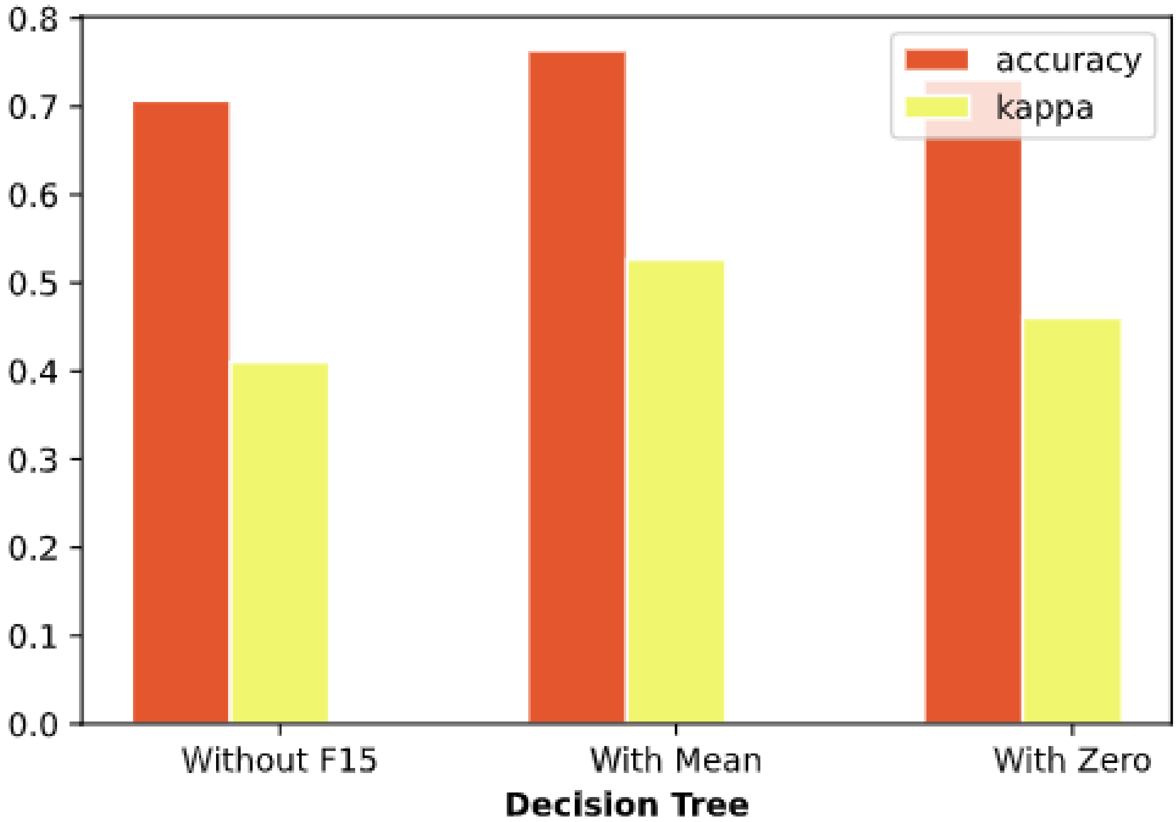

Figure 1: Decision Tree Performance

```
Accuracy: 0.7633333333333333
Kappa Statistics: 0.5264140875133404
```

Filling F15 column with zero:

```
Accuracy: 0.73
Kappa Statistics: 0.4602878464818764
```

I plotted a barplot to see the differences.

2.2 Performance of K-Nearest Neighbour

For the K-Nearest Neighbour I imported `KNeighborsClassifier` from `sklearn.neighbors`. Then I created a classifier object `KNCO` and fit it with the split data. Then predicted the output using `KNCO.predict(x_test)`. For K-Nearest Neighbour I got the following values of accuracy and kappa:

When I removed the entire F15 column:

```
Accuracy: 0.5866666666666667
Kappa Statistics: 0.1626148028092923
```

Filling F15 column with mean:

```
Accuracy: 0.59
```

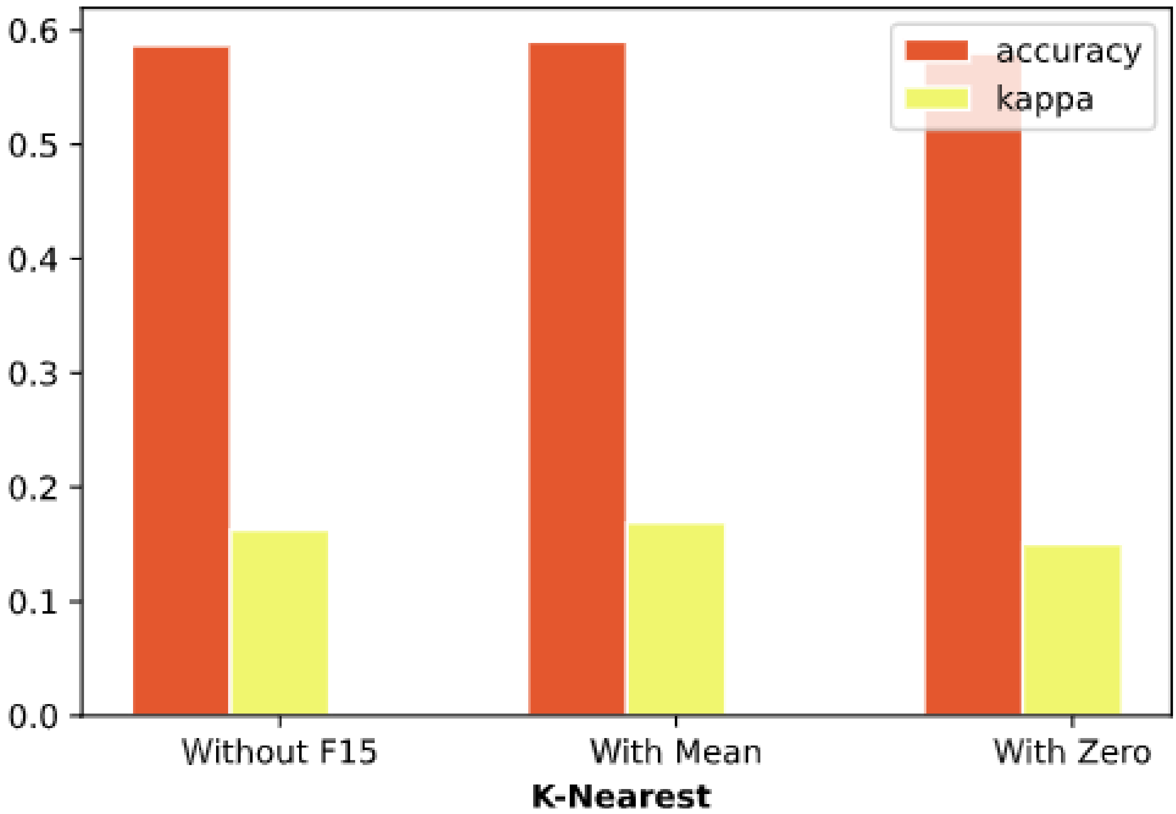

Figure 2: K-Nearest Neighbour Performance

```
Kappa Statistics: 0.16891891891891897
```

Filling F15 column with zero:

```
Accuracy: 0.58
Kappa Statistics: 0.15002698327037245
```

I plotted a barplot to see the differences.

2.3 Performance of Naive Bayes

For the Naive Bayes I imported `GaussianNB` from `sklearn.naive_bayes`. Then I created a classifier object `NBCO` and fit it with the split data. Then predicted the output using `NBCO.predict(x_test)`. For Naive Bayes I got the following values of accuracy and kappa:

When I removed the entire F15 column:

```
Accuracy: 0.5233333333333333
Kappa Statistics: 0.04818956336528224
```

Filling F15 column with mean:

```
Accuracy: 0.5433333333333333
Kappa Statistics: 0.09102972399150744
```

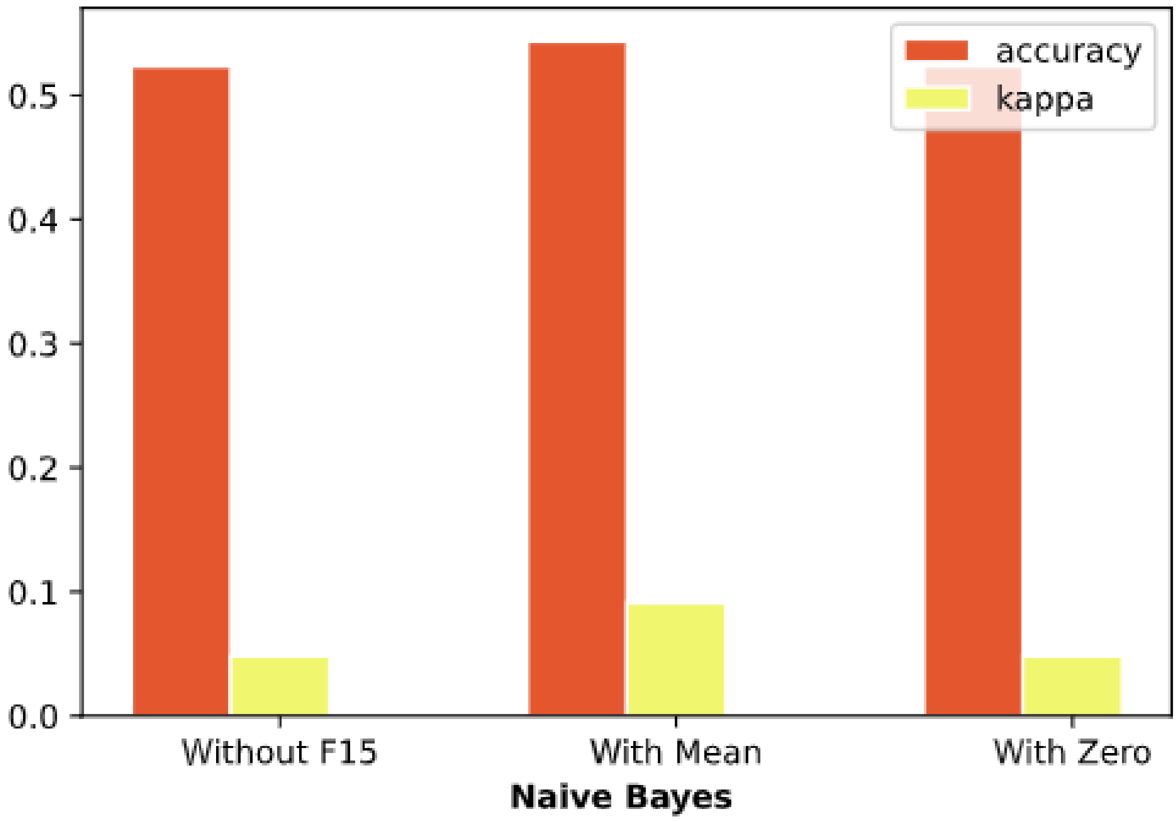

Figure 3: Naive Bayes Neighbour Performance

Filling F15 column with zero:

```
Accuracy: 0.5233333333333333
Kappa Statistics: 0.04818956336528224
```

I plotted a barplot to see the differences.

2.4 Performance of Support Vector Machine

For the Support Vector Machine I imported `svm` from `sklearn`. Then I created a classifier object `SVCO` and fit it with the split data. Then predicted the output using `SVCO.predict(x_test)`. For the Support Vector Machine I got the following values of accuracy and kappa:

When I removed the entire F15 column:

```
Accuracy: 0.5233333333333333
Kappa Statistics: 0.04818956336528224
```

Filling F15 column with mean:

```
Accuracy: 0.5466666666666666
Kappa Statistics: 0.10667367314766163
```

Filling F15 column with zero:

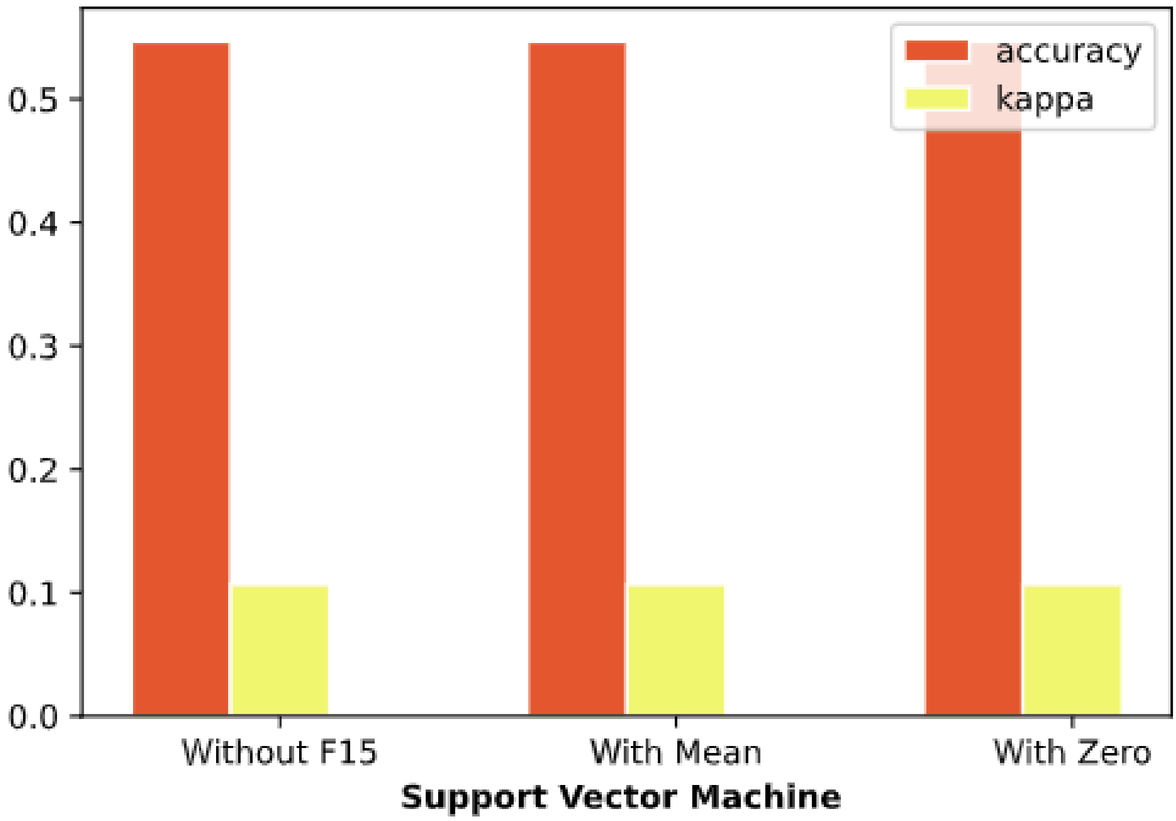

Figure 4: Support Vector Machine Performance

```
Accuracy: 0.5466666666666666
Kappa Statistics: 0.10667367314766163
```

I plotted a barplot to see the differences.

2.5 Performance of Multi-layer Perceptron

For the Multi-layer Perceptron I imported `svm` from `sklearn`. Then I created a classifier object `MPCO` and fit it with the split data. Then predicted the output using `MPCO.predict(x_test)`. For Multi-layer Perceptron I got the following values of accuracy and kappa:

When I removed the entire F15 column:

```
Accuracy: 0.54
Kappa Statistics: 0.006908462867011966
```

Filling F15 column with mean:

```
Accuracy: 0.5533333333333333
Kappa Statistics: 0.09115572436245256
```

Filling F15 column with zero:

```
Accuracy: 0.5433333333333333
```

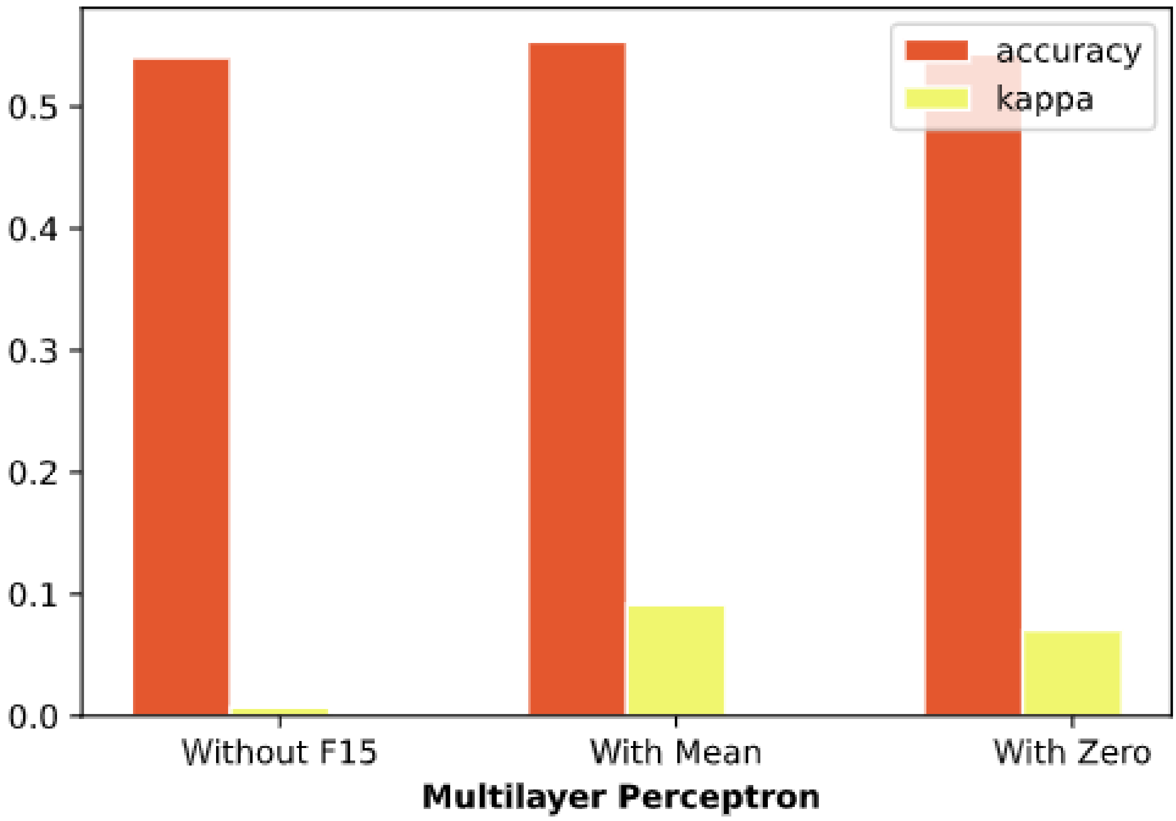

Figure 5: Multi-layer Perceptron Performance

```
Kappa Statistics: 0.07030401737242131
```

I plotted a barplot to see the differences.

2.6 Decision for Classifier Selection

After seeing the accuracy and kappa values I decided to use a Decision Tree with mean values for this project as that gives the accuracy of 76%.

2.7 Prediction on a hold-out test set

After choosing the perfect classifier I predicted the values for the CE802_P2_Test.csv fileś `Class` column.

```
train_data_x = test.iloc[:,:15]
train_data_y = test.iloc[:, 15]
ClassPredict = DTCO.predict(train_data_x)
```

Added the values to the Class column and replaced the test file with the predictions

```
test['Class'] = ClassPredict
```

3 Additional Comparative Study

In this part of the project, I need to predict both the claim and the value of the claims. According to the new company the provided historical data, each customer, and a numerical value representing the value of the clime.

3.1 Investigate the project

For this project, I will approach Linear Regression and two more machine learning procedures. They are as follow:

- Gradient Boosting Regressor and
- Random Forest Regressor

Then I will observe the accuracy of the predictions then choose the best procedure for this project.

3.2 Data Pre-processing

While looking at the data file given to me I saw the data was mostly clean. But the Target column has some null values. So I decided to fill them with the following options. (sci, 2020)

- Use the column with zero
- Fill the zero with mean values column
- Fill the values with median values of the column

Comparing the accuracy I will decide which is the best for that type of data. Also, I noted that F6 and F14 columns have nonnumerical data which is interesting. For making things easier I changed the strings to integers as follow:

```
def replace_country(a,b):
   data.F14.replace(a,b,inplace=True)
replace_country('USA',1)
replace_country('UK',2)
replace_country('Europe',3)
replace_country('Rest',4)

def replace_cF6(a,b):
   data.F6.replace(a,b,inplace=True)
replace_cF6('Low',1)
replace_cF6('High',2)
```

```
replace_cF6('Very high',3)
replace_cF6('Medium',4)
replace_cF6('Very low',5)
```

3.3 Selecting the best ML Procedure

In this part, I will discuss how much score I got for Linear Regression, Gradient Boosting Regressor, and Random Forest.

Linear Regression: For Linear regression I got the following scores for the data:
With zero on the `Target` column: 0.6789168904393641
With mean values on the `Target` column: 0.4203527219307589
With median values on the `Target` column: 0.5829178459399992
Gradient Boosting Regressor: For Gradient Boosting Regressor I got the following scores for the data:
With zero on the `Target` column: 0.8164604167727227
With mean values on the `Target` column: 0.66522042679068
With median values on the `Target` column: 0.78070237240036607
Random Forest Regressor: For Random Forest Regressor I got the following scores for the data:
With zero on the `Target` column: 0.6770847115136522

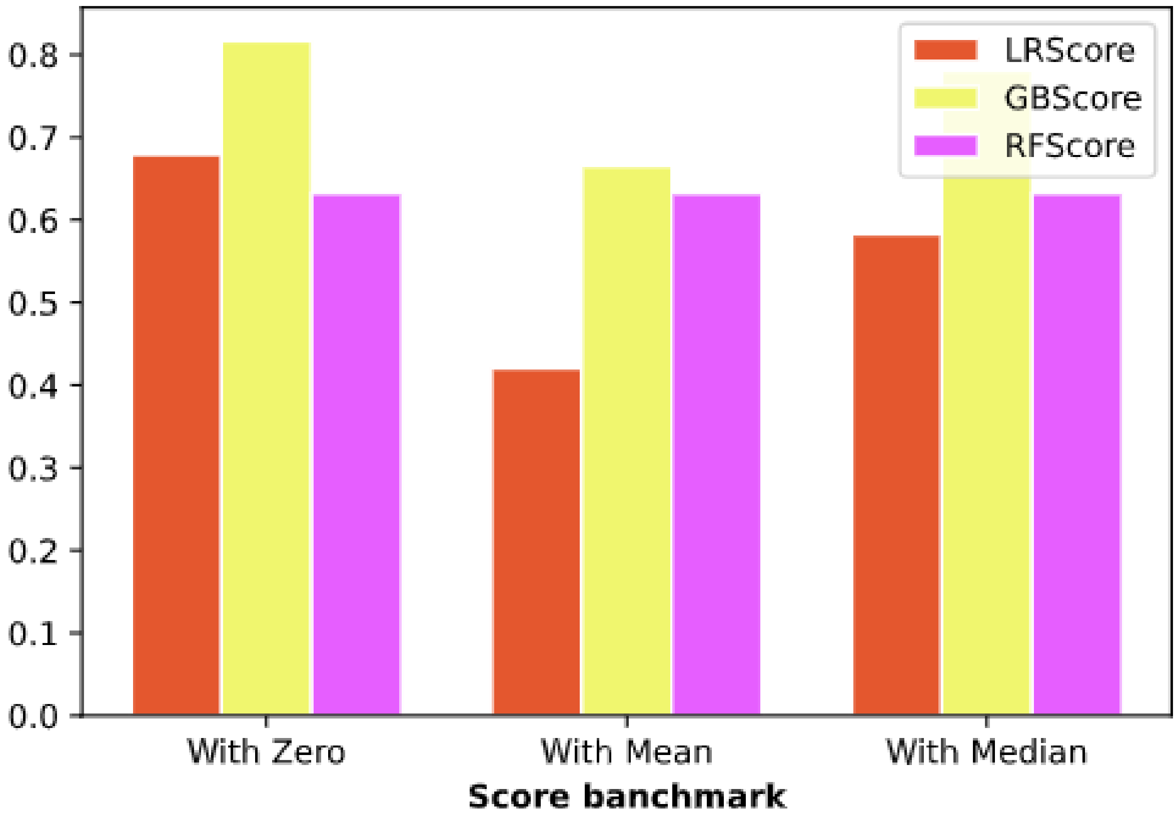

Figure 6: Scores for multiple ML procedures

With mean values on the Target column: 0.5803909303022915
With median values on the Target column: 0.6323511719057406
You can see a comparison barplot for all the scores (Figure 6)

So, the conclusion is I am going to use Gradient Boosting Regressor with zero on the Target column.

3.4 Prediction on a hold-out test set

I used the best scored procedure to the the prediction and predicted the Target columns for the CE802_P3_Test.csv file. I changed back to strings for column F6 and F14 and saved the file.

4 Final Outcome

Finally, I would like to say I enjoyed the projects. I tried my best to get the best predictions but they can be improved by cleaning the data, tuning the feature variables, and for future work it will be a privilege to work with the travel company and the other company who hired me for the project.

References

(2020). learn: machine learning in python - scikit-learn 0.16.1 documentation.

Alpaydin, E. (2020). *Introduction to machine learning*. The MIT Press.