Statistical &Machine Learning Approaches

Individual Assignment

By: Marijose Marcos Iglesias

March 30th, 2022

Introduction

In this individual project, the task of the author is to explain different machine learning models and to implement them successfully to a given dataset through R. Each model should be described and explained to detail in order to show understanding on the models and the logic behind them. The algorithm, the objective function and the fitting of the models should also be discussed with the goal of finding the positive and negative points of each model to also discuss them in this report.

The code for these models will be explained to detail as well as the process and logic of the author when implementing the machine learning models. Once the models are applied to the dataset, the author is expected to setup a benchmark that will allow the comparison of the different models and that will give insights on each model for the author to analyze deeply. The benchmark will be explained as well as the results and their respective conclusions.

The benchmark experiment will include variable selection through feature engineering, the use of a dimensional reduction method (Stepwise), the cross-validation method in order to measure model performance and finally, the AUC as an evaluation metric for the models.

The main objective of this project is for the author to reach a deeper level of understanding on machine learning and statistical models. It is important that the author understands how these machine learning models work as well as to show the ability of setting up a machine learning pipeline.

Dataset and Variable Analysis

The assigned dataset "Bank Marketing" contains 2000 observations and 21 variables. The identified target variable with which the models will be performed is "subscribe", a categorical binary variable containing 1 or 0. The variables presented in the dataset are: client_id, age, job, marital, education, default, housing, loan, contact, month, day_of_week, campaign, pdays, previous, poutcome, emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m, nr.employed and subscribe. There were several identified categorical values in the dataset such as job, marital, education, default, housing, loan, contact, month, day_of_week and poutcome. These categorical variables needed to be treated in order to apply machine learning models to the dataset. Missing values were identified in all of the variables except for variables client_id and subscribe.

Data Pre-processing

To start off with the project, the pre-processing of the data was essential since there were several inconsistencies identified in the data. After setting up the path, the data was split into train and test immediately to treat the data separate since the beginning to avoid data leakage as much as possible. This means the output of the project would consist of 2 separate final base tables. All of

the pre- processing that will be mentioned was applied twice for both split sets. The numeric variables (non- categorical) for both sets were normalized to ensure better model prediction performance. Most of the columns presented more than 100 missing values (NA's), therefore, either the mode or mean of the column depending on the variable were used to replace the missing values of the dataset. For variables such as job, marital, education, default, housing, loan, contact, month, day_of_week, etc., the NA's were replaced by the mode since it seemed more suitable, and it was not possible to replace by the mean due to them being categorical. For other variables such as emp.var.rate, cons.price.idx, cons.conf.idx, nr.employed, euribor3m, etc., the NA's were replaced by the mean of these columns since it seemed more suitable. As mentioned before, the categorical variables needed to be encoded in order for the dataset to be implemented in the models, therefore, these variables were encoded with individual if-else statements. The meaning of the encoding of each variable is as follows:

Job- Management:1, Unemployed:2, Technician:3, Services:4, Unknown:5, Blue-Collar:6, Admin:7, Entrepreneur:8, Retired:9, Housemaid:10, Student: 11, Self-Employed:12.

Marital- Married:1, Signle:2, Divorced:3, Unknown:4.

Education- Basic 9y:1, University Degree:2, High School:3, Unknown:4, Professional Course:5, Basic 6y:6, Basic 4y:7, Illiterate:8.

Default- No:1, Unknown:2, Yes:3.

Housing- No:1, Unknown:2, Yes:3.

Loan- No:1, Unknown:2, Yes:3.

Contact- Cellular:1, Telephone:2.

Month- Jul:7, Jun:6, May:5, Apr:4, Aug:8, Nov:11, Sep:9, Oct:10, Dec:12, Mar:3.

Day_of_week- Thu:4, Mon:1, Tue:2, Wed:3, Fri:5.

Poutcome- Non-existent:1, Failure:2, Success:3.

Contacted_or_not - Dummy variable of pdays to know if the client was contacted or not. Contacted:1, Not contacted:0.

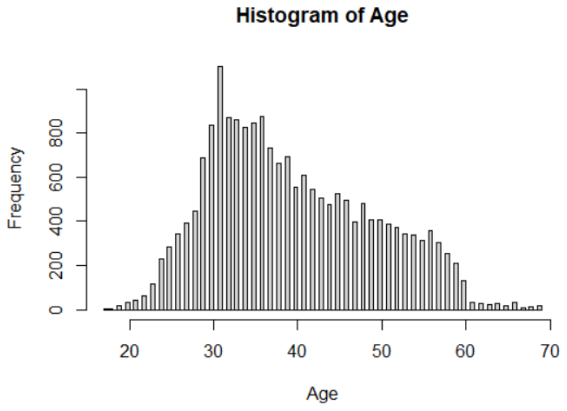
After encoding the categorical variables, these variables were converted to factor class, in order for the model not to identify them as scalar, meaning that 2 is more than 1, which is not the case with categorical variables. Variables client_id and pdays were then dropped since they were not useful for the models.

When this process was done, the outliers were firstly identified with a boxplot per variable. The variables where outliers were identified were treated by calculating the quantiles of these variables which are age and cons.conf.idx.

As an example, below we find a distribution of the variable age before the outliers were treated. It is clear that after 70 years, multiple outliers are displayed:

Using the package "ggstatsplot", a subset of the data was created where the quantiles were subtracted (upper and lower ranges) by leaving out the outliers. This subset then became the final processed dataset without outliers, which has a total of 19585 rows and 22 columns.

Below the age variable is displayed after outliers were treated:



The same process was performed on the cons.conf.idx variable.

Finally, the data was split into train and test for the Logistic Regression, setting a seed to have the same results whenever the code is run. The train set represents 80% of the data and the test set 20%.

Model Explanation

Logistic Regression

The Logistic Regression model is a supervised classification algorithm, which means it uses model training for the model to make predictions and learn overtime to reduce the prediction error as

much as possible. This model is basically used to predict the category of users, in this case clients, based on multiple predictor variables. This model is normally used for binary classification problems. Through the study of the data of the predictor variables, the logistic regression model is able to predict a binary outcome, 1 or 0, 1 being TRUE and 0 being FALSE. This means that for every value of each predictive variable, a prediction can be made for the target variable.

The Logistic Regression model is represented by the following function:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$$

The logistic regression regression is represented by an S-shaped curve called the Sigmoid function, which takes any real value from the inputted variables and converts them in 1 or 0 depending on if the predictor variables fulfill the p value conditions.

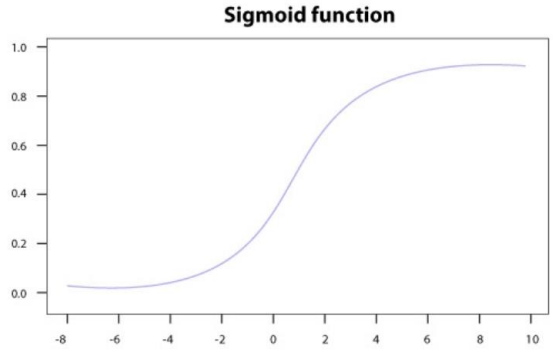


Figure source : https://www.analyticssteps.com/blogs/how-does-linear-and-logistic-regression-work-machine-learning

For example, in this particular case, we are trying to predict with a binary logistic model (0,1) and multiple predictor variables, which are in the bank_filtered table, if a client is subscribed or not, since subscribe is in this case our target variable. The prediction on the subscribe variable is defined by clients that had a p value greater than 0.5 in the predictive variables of the table bank_filtered. If clients fulfill this condition, the model will predict they are subscribed (1), and if they don't, they will predict them as not subscribed (0).

Advantages

- This method is easy to implement and to train, one of the simplest classification models and is very efficient to train and its interpretation is quite straight forward.
- Overfitting can be treated

- It is flexible in a sense that it can be adapted to multiple classes such as multinomial regression.

- It lets you measure the association between two variables as positive or negative.
- When the data is separable, the accuracy is expected to be very good.

Disadvantages

- The assumption of linearity between both the dependent and the independent variables might result in a limitation for this model.

- It is not useful with non-linear problems, which can be non-realistic for real life scenarios.
- Does not work well with poor quality data that is incomplete or is not pre-processed.

Implementation & Results

To start the implementation of the model, the logistic regression model was fitted using a binomial family and the gml function, then the model was summarized to get more insights on the model and the variables. A variable selection method was used, STEPAIC, which helped pick the most suitable predictor variables to use to achieve a better performing model. The model was then fitted one more time with the optimal predictor variables that the STEPAIC function suggested which were around 10 variables. The predictions were then set with a probability >0.5 on the test and model to identify any possible over or underfitting of the model. The accuracy of the model was 89.77%on the test and the train 90.13%. The results of the accuracy are coherent so we can assume there is no overfitting in the model. The confusion matrix of the predictions is as follows:

	pre	
	0	1
0	3489	38
1	371	102

The matrix shows how the model predicted a non-subscribed client 3489 times when it was actually a non-subscribed client. On the other hand, it predicted a subscriber 102 times when it was actually a subscribed user. In both cases the model predicted more correct answers than wrong.

The cross validation is the computed by defining the model and running the k fold, taking the client_id as the id and the subscribe variable as the target variable.

The results of the cross-validation process with the selected features show an AUC mean score of 77.28% (0.7728310):

```
Resample Result
Task: train
Learner: classif.logreg
Aggr perf: auc.test.mean=0.7728310
Runtime: 8.32065
```

Linear Discriminant Analysis (LDA)

The LDA is a linear classification model, normally used for classification problems and/or feature extraction. In this method, the predictive distribution is performed separately for each class, and through the use of the Bayes' theorem, these are converted into estimates for the prediction.

Basically, the LDA model estimates the probability that each input belongs to a class. The output that has the highest probability is considered the final output class from which the LDA will make a prediction.

The LDA method does this through the following steps: Firstly, the separability of the classes is defined (between-class variance), which is calculated by getting the distance between the means of each of the classes. The closer these means are the most complex the classification problem may be.

Secondly, the distance between each class's mean and sample is calculated. This is also a complexity factor, the higher the variance is the more difficult it is to separate the classes.

Finally, using singular value decomposition or a least square method, the between-class variance is maximized and the distance between each class's mean and sample is minimized.

The linear discriminant analysis is also known as the generalized version of the Fisher's linear discriminant and can also be considered as an alternative to the logistic regression model as a less direct approach to predictions. This method might be preferred over logistic regression since this method is more stable when there is a normal distribution of each of the classes and is regularly more used in cases where more than 1 class is presented. When p=1, meaning there is only one predictor, we assume a Gaussian or normal density and a one-dimensional setting. When p> 1, meaning there is more than 1 predictor, a multivariate Gaussian distribution is assumed, meaning each individual predictor follows a one-dimensional setting.

Advantages

- It is a straightforward algorithm, and it is able to do more stable predictions than logistic regression thanks to class separation.

- As mentioned previously, implementation can be more convenient than other models when more than 1 class is presented.

- Minimizes variance through feature reduction
- Reduces high dimensionality of data

Disadvantages

- Requires a normal or Gaussian distribution assumption on predictors.
- Do not perform optimally with some categorical variables.
- Does not work optimally with small size data samples or unbalanced data
- Is likely to overfit
- Can only be used with linear problems

Implementation & Results

For the implementation of this model, we start off by fitting the model with the LDA function, using the predictor variable and the train. Once the model is fitted, the feature selection process is implemented using the train function and defining the parameters method, trControl and tuneGrid.

Once again, the LDA model is fitted with the optimal features that the feature selection process has suggested, which are more than 10 variables. Then, the model is summarized to get more information.

After this, the confusion matrix was calculated using the test set and the prediction (0.5):

```
      0     1
0 13628  1169
1   574   629
```

The matrix shows that the model predicted as not subscribed 13628 users that were actually not subscribed and predicted 629 subscribed users as subscribed.

The predictions on the test and train are then performed to know if there is any overfitting in the variables or not. The LDA model shows a model accuracy of 89.82% on the test prediction and 90.06% on the train model accuracy prediction. With these numbers we can assume the data is not overfitted.

Next, the cross-validation process is performed to get the AUC of the model to measure and compare performance to other models.

The results of the model after the cross-validation process with the selected features show an AUC mean score of 75.70% (0.7570900):

```
Resample Result
Task: train_knn_l
Learner: classif.lda
Aggr perf: auc.test.mean=0.7570900
Runtime: 1.9418
```

K-Nearest Neighbors

The K-Nearest Neighbors method or KNN, is a classification and regression supervised method. The objective of this algorithm is to identify for the test data the class they belong to. As in other models such as the previously presented one, KNN does this by calculating distance between data points. More specifically, KNN calculated the distance between test data points and train data points.

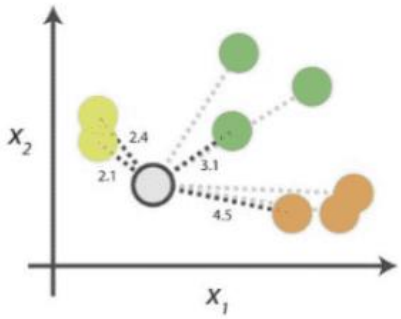


Figure source :https://medium.com/swlh/k-nearest-neighbor .

The model tries to make its prediction based on the chances of the test data points to belong to a class.

For example, as shown in the graph above, the white dot has 8 neighbors and 3 classes it could be classified into. To successfully classify the white dot the distance between the white point and its neighbors is calculated to know which class the white dot is closest to. The distances of the white point and all of its neighbors is calculated and ranked from lowest to highest. The "k" value defines how many neighbors the model should take into account for its prediction. If k=5, then the model will rank the 5 nearest neighbors to make its prediction. Once the neighbors' distance is ranked from low to high, the votes for each of the neighbor's classes are counted and the prediction is made.

The model tries to make its prediction based on the chances of the test data points to belong to "k" neighbors' classes. However, there is no optimal k value, it should be estimated and tried with the algorithm, but the k value should not be too small since some important neighbors might not be taken into consideration.

Advantages

- No testing is needed for the learning part of the model. The model doesn't need to be trained prior to the prediction in order to perform, it learns from the training set at the same time it is making the prediction.

- KNN is a simple model to implement and to interpret since only 2 parameters are needed to perform this model, the "k" value and the distance function.

- New data can be added into the data set, since there is no training process of the model, the prediction can be updated without having to create a new model.

- There is flexibility on what distance function to choose for this algorithm such as Euclidean, Manhattan, etc.

Disadvantages

- This model, however, does not work well with high dimensional data, since as explained before, the method calculates distances between neighbors and high dimensional data would make it difficult for the model to calculate these distances.

- Large datasets might affect the performance of the model for a similar reason that in the previous point, since the complete dataset is considered and processed for each and every prediction.

- Another disadvantage is that outliers can have a big impact on this model's prediction, especially in large dimensions, since all of the information the model needs for the prediction is extracted from the inputted data.

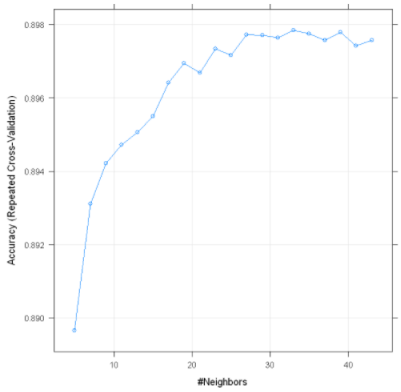
Implementation & Results

As mentioned before, the implementation for this method is quite simple. To begin, the knn model is fitted using the material and method learned in the previous classes. This is done so we can get a summary of what the model would look like, even if this specific first model will not be necessary in general for our prediction. In the fitting of the model, the k value was assigned to 5 to start with and we get the summary of the model: the summary shows that the model has identified 3751 zeros (not subscribed) and 249 ones (subscribed) regarding the target variable.

After this, traincontrol is used to define the control variable which will be used to find the optimal k value later on with a train function. The results of this function are showed below:

```
Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 33.
```

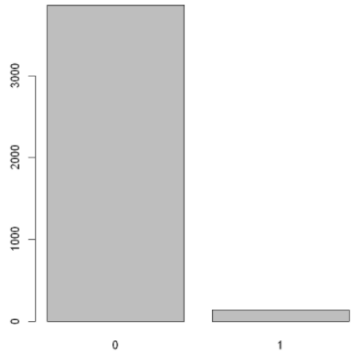
Now it is clear that the optimal k value for our model would be 33, so we proceed to apply this value to our model, and we fit the model again. The plot below corroborates that in fact, the most optimal number of neighbors is 33:



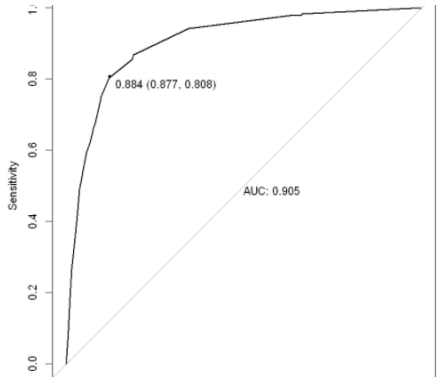
The confusion matrix of our new optimal model is as follows:

predi_knn	0	1
0	3490	364
1	37	109

The matrix shows that the model with the optimal k has predicted 3490 clients as not subscribed when they were indeed not subscribed and 109 as subscribed when they were indeed subscribed. We notice that in the case of the subscribers, the model did predict way more false positives than true positives:



Next, the predictions on the train are performed: the model accuracy of the test prediction is 89.77% and the AUC calculation gives a value of 90.45%.



<u>Decision Tree Algorithm- Classification Trees</u>

As we know, there is multiple types of decision trees, there is decision tree algorithms for regression and classification. Both trees are very similar but have some important differences, one of them being that a regression tree is used to get a quantitative prediction and the classification tree a qualitative one. However, since this project focuses on classification methods, the decision tree algorithm that will be analyzed in this case is the classification tree.

The objective of the classification trees is to predict whether the observation belongs to the most commonly occurring (mode) class of the data or not. Since data is quantitative and not qualitative,

there is no mean to be computed so the model bases on the mode of the classes. When trying to interpret the classification tree results, the class the model predicted is not only relevant but also the actual percentage of the train data that represents this class.

Every decision tree has a node from where the tree splits, edges which connects possible outcomes of each node, a root which is the node where the split begins and leaves which are the final nodes that predict the outcome.

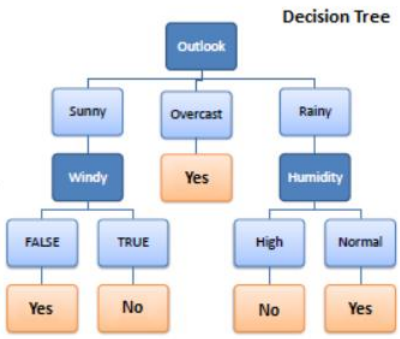


Figure source : https://www.saedsayad.com/

The decision tree involves the partition or separation of data into subsets with similar values. The goal of the algorithm is to analyze these subsets on homogeneity through entropy. Entropy is the decisive factor in a decision tree on how the data will be split. When interpreting the results, the classification error is the value that will tell us what fraction of our train data does not belong to the mode class or the most common class.

Advantages

- Very simple model, easy to implement, interpret, visualize and to explain.
- Does not require feature scaling with non-linear data.

- One of the fastest models to identify variable importance and relationship between variables.
- It is not impacted by outliers since the splitting of nodes makes the model resistant to them.
- Na's in the data do not impact the model's prediction.

Disadvantages

- Prone to overfitting since multiple nodes are created and the tree can become too large/complex, especially when large datasets are used.

- Can be unstable since small data differences might generate a whole new tree creation.

- Not optimal for datasets with many numerical variables since the training process can be very time consuming.

Implementation & Results

The implementation is started by fitting the model with the tree function on the train and defining our target variable, then summarizing the model to get insights on the misclassification rate and mean deviance.

```
Residual mean deviance:  0.579 = 9261 / 16000
Misclassification error rate: 0.1001 = 1601 / 16000
```

As mentioned before, the misclassification error rate is in this case showing that 10.01% of the train data does not correspond to the most common class. The residual mean deviance shows how accurate the response is predicted by the model with the predictor variables, which in this case is not optimal: 57.90%. The model is then refitted with the optimal parameters that are found with the train function and the caret library. The method is set to "ctree" or classification tree and the data is set to the train.

The parameter selection shows what range the mincriterion parameter should be set to in order to get the best results of the model, in this case its 0.99:

```
Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mincriterion = 0.99.
```

Therefore, the model is fitted again with a mincriterion range between 0.90 and 0.1 to get multiple results and compare. The train function then aggregates the results and fits the mincriterion of 0.428 on the full training set:

```
Selecting tuning parameters
Fitting mincriterion = 0.428 on full training set
```

Next, both predictions in the test and train sets are performed and the model accuracy is calculated. Once again, the values for both models are equal to the model accuracy of the previous models.

The test set accuracy is in both the train and test prediction 50.00% The confusion matrix was the computed and shows that the model has predicted 3489 true not subscribed clients and 102 true subscribed clients. It is important to mention the prediction of this model has predicted almost 3 times more false subscribed people than true subscribed clients:

predict_112	0	1
0	3489	371
1	38	102

After calculating the AUC of the model for the benchmarking, the AUC value outputted for this model is 82.64%.

Random Forest

The Random Forest algorithm is a supervised machine learning method that consists of multiple non-related decision trees (previously explained); it can be used for classification or regression problems like other models. By combining multiple decision trees, the aim of the random forest is not only to reduce high variance and bias but to reach a single result based on more information and a majority vote method; the logic behind this model consists of the assumption that decision trees perform better in ensembles than alone since even though decision trees on their own might have errors, most of them will be accurate and should lead the group in the correct direction of the prediction.

In most cases, the training sets are trained with the bootstrap aggregation method, which takes random decision trees from different samples and shuffles and distributes them to other training samples. Random forests have three principal parameters which are normally defined before training the model: number of samples, size of nodes and finally number of decision trees that should be used.

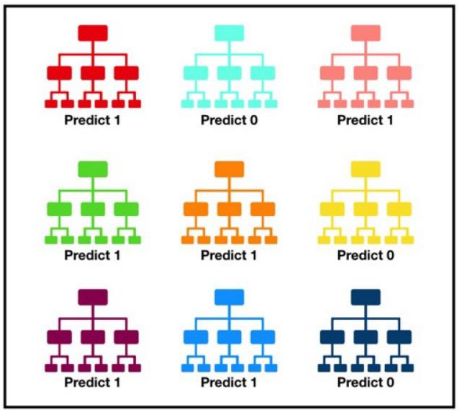


Figure source: Understanding Random Forest. How the Algorithm Works and Why it Is… | by Tony Yiu |

Each tree in this algorithm outputs a class prediction and the class with most votes becomes at the end the final model's prediction. What is interesting in this model is the low correlation between trees since they can create together an ensemble of more accurate predictions than individual predictions sometimes.

Advantages

- Improvement over bagged trees (decorrelation tweak)
- It is more flexible than decision trees on their own
- Easy to identify feature importance through Gini measure
- No normalization/standardization of data is required prior to this model and easy handling of NA's

- Used for both classification and regression problems
- Not sensitive to outliers, outliers do not impact the model importantly
- High prediction accuracy model

Disadvantages

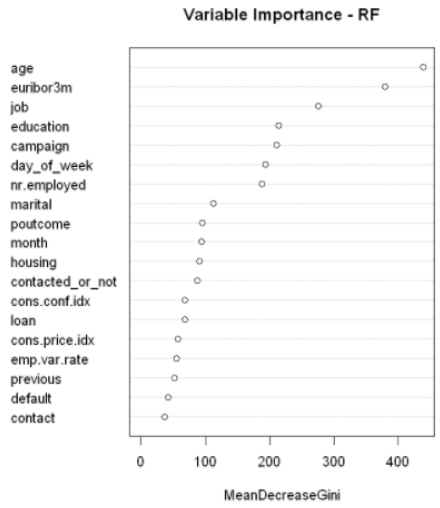
- Their implementation is not optimal with large datasets and can be dense and heavy to process

- It can be considered as a black box model which means they might be difficult to interpret and fully understand their predictions
- Overfitting can happen with unstructured heavy data

- Can be biased if some categorical variables have several levels since the model might think they are more useful than others

Implementation & Results

First, the model was fitted, using the randomForest function and the test as the data. After this, the importance features of the model are calculated by using the function importance and using the previously fitted model. This function gives us the mean decrease accuracy and the mean decrease Gini; the higher the mean decrease Gini, the more important the variable is. Following this logic, the most important features are picked to fit the model once again to get the optimal results of the model's prediction. The graph below shows in a visual way the importance of each of the features in the model:



The top 10 important features were then picked and inputted into the model once again for fitting, hoping for the optimal result. The output gives an out of bag error (OOB) of 11.52%.

Next, the prediction of the model for test and train are performed: The test set gives an accuracy of 88.90% and the test of 99.40%. Finally, the AUC score for the final model is of 73.31%.

Benchmarking- Model Comparison

In this final section of the report, the conclusion of each model is discussed: this includes the model accuracy of both their train and test as well as the AUC score of each model trained with the optimal features/parameters. Based on the results of the AUC values, which is what we are comparing the models with, the KNN algorithm has the highest value with 90.45%. This shows that even though the KNN algorithm is one of the simplest methods compared to some others, its accuracy is in most cases high and does not make many assumptions on data like other models do. The next model with the second-best AUC value is the Decision Tree algorithm with a value of 82.64%, even though the model accuracy on predictions for both train and test were only 50%, which is way lower than other models, this algorithms AUC surpassed the other models (LDA, Logistic Regression and Random Forest. The Logistic Regression model and the LDA model had very similar model accuracies on both train and test predictions and ended up with similar AUC values, however, the logistic regression algorithm seemed to perform better than the LDA algorithm. On the other hand, the Random Forest algorithm, which had the best train and test model accuracy rates ended at the bottom of the best performing models, with an AUC score of 73.31%. It is clear that, even though some model accuracy values seemed higher in some models than others, these did not have an impact on the AUC values at the end. If we take the AIC values as well which was also calculated for logistic regression and LDA, we can see that the conclusions make sense: The AIC score measures the best model from lowest to highest, meaning that the lower the AIC, the better the model performs. The logistic regression has an AIC value of 91.08% and the LDA of 91.11%. These results correspond to the previously drawn conclusions since not only the values correspond but the difference between percentages is minor.

References

https://www.analyticsvidhya.com/blog/2021/04/simple-understanding-and-implementation-of-knn-algorithm/

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://www.analyticsvidhya.com/blog/2021/04/beginners-guide-to-decision-tree-classification-using-python/

https://www.unite.ai/what-is-k-nearest-neighbors/

https://www.ibm.com/cloud/learn/random-forest