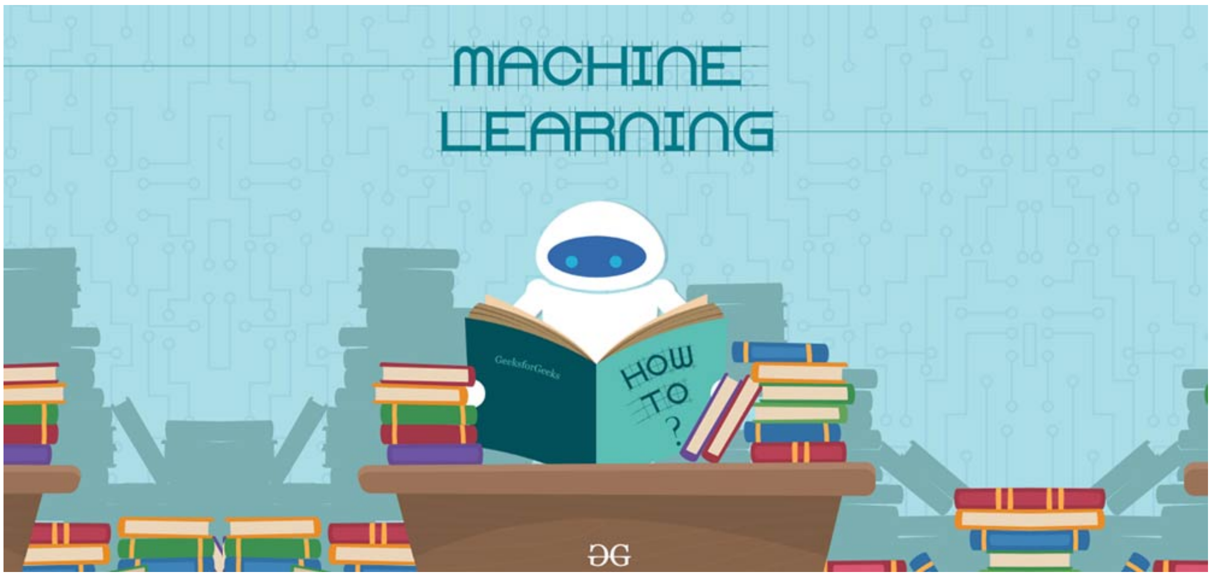
Professional Academy Certificate in Data Analytics: Machine Learning

Link to colab notebook;

Machine Learning analysis of student performance.ipynb

Abstract

Machine learning classification analysis of whether or not students currently enrolled at a University will either Graduate, Drop Out or are still enrolled at the University.
Supervised classification is one of the tasks most frequently carried out by Intelligent Systems *(S. B. Kotsiantis &I. D. Zaharakis & P. E. Pintelas, 2006)*. It is for this reason that I chose to pick a classification problem for this project. The dataset I have chosen provides information that was supplied by the students during the enrollment process (demographics, social-economic factors and their academic path.) as well as the academic performance of each student at the end of their first and second semesters.



Introduction

The machine learning problem to be explored by this project is an example of classification. Classification in machine learning is a supervised method in which our model tries to put the correct target variable (Y) label on each data point in the test set based on the information it gains from the corresponding (X) attributes.in the training set. (*Charbuty, B., & Abdulazeez, A., 2021*)
Once the preliminary analysis and model training is complete we then try to maximise the performance of our model by performing hyperparameter tuning and utilising ensemble learning methods.

Data

The dataset that I have chosen for this project is titled “Predict students' dropout and academic success” and I found it on the well renowned and respected archive, UC Irvine Machine Learning repository. You can find the dataset here and also in the bibliography of this report. The shape of this dataset is 4424 data points (rows) with 37 fields (columns) consisting of 1 target variable and 36 attributes. Almost all of the data types in the data set are either integers or floats despite containing many categorical attributes, this is discussed further in the data preparation section.

Data Preparation

1. **Importing**
   In order to use this dataset for my analysis the first step is to call pandas .read_csv function on the csv file and assign it to a data frame (Studentdf is the name of the original unmodified data set). I then call .head(), .shape and .describe() on the data frame in an attempt to get an initial overview of what the dataset looks like and how I will begin to analyse them

2. **Data sanity checks**
   The dataset mostly consisted of integer or float data types but I converted the target variable into integer form as well in the csv file itself using Excel. In order to achieve this the categorical data was converted into an integer for ease of analysis, for example instead of the target variable being a string (Dropout, Enrolled,Graduated) I converted them into the integer values 1,2 and 3.

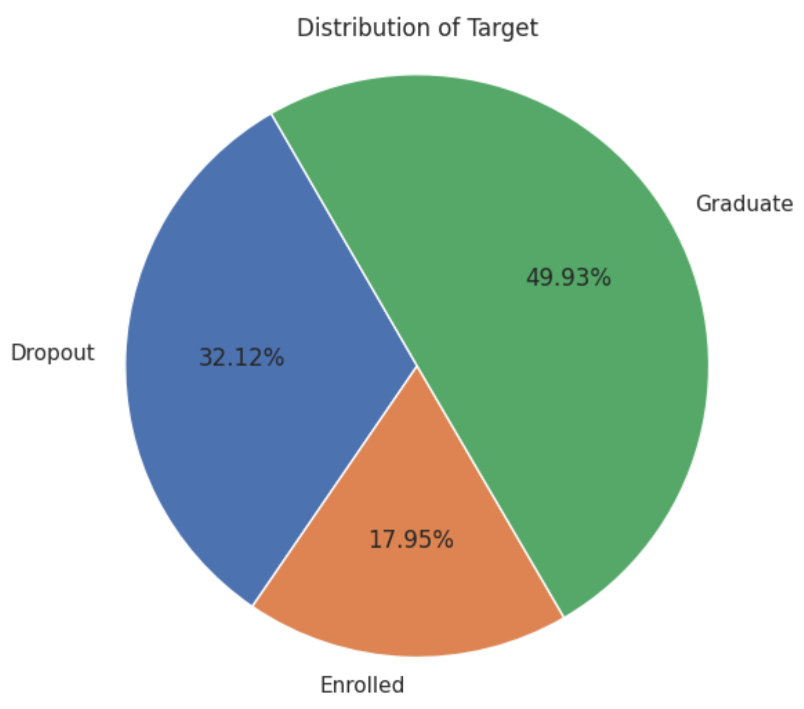


Fig 1. Pie chart showing distribution of target variable

   The next step to do is to perform some sanity checks. I begin by calling the drop_duplicates pandas method. Following on from this I call the isnull() method to see if our data is missing any values that we would need to fill or delete. The data has no duplicates or missing values so we proceed to identifying the target variable and splitting it from our attributes. Finally I used the train_test_split() method to split my data into 80% train and 20% test data for the model. The data is now ready for analysis

3. **Custom function initialisation**
   As I will be performing hyperparameter tuning on most of the models that I train, I felt

it was important to instantiate a custom function; hyperparam_tuning. This function takes as arguments a classifier, a list of parameter options to be hypertuned, X_train and Y_train. It will then perform a 5-fold GridSearchCV using negative mean squared error as the scoring method (This performed better in my testing than accuracy). It will finally train the Grid Search model on our X and Y training data sets and output the hyperparameters which performed the best

4. Feature Importance to subset data

My data set has 37 attributes (1 target and 36 normal attributes) Having so many attributes can introduce a lot of “noise” and makes creating visualisations such as pair plots and heatmaps much more cumbersome. The solution that I have chosen is to use a random forest to create a panda series of all of our attributes and rank them in order of importance. I then take the top 10 attributes and use those for our refined training and test sets. Below is a line graph indicating the results of my feature importance test. As can be seen one feature is deemed by far the most important while others have almost no importance whatsoever.

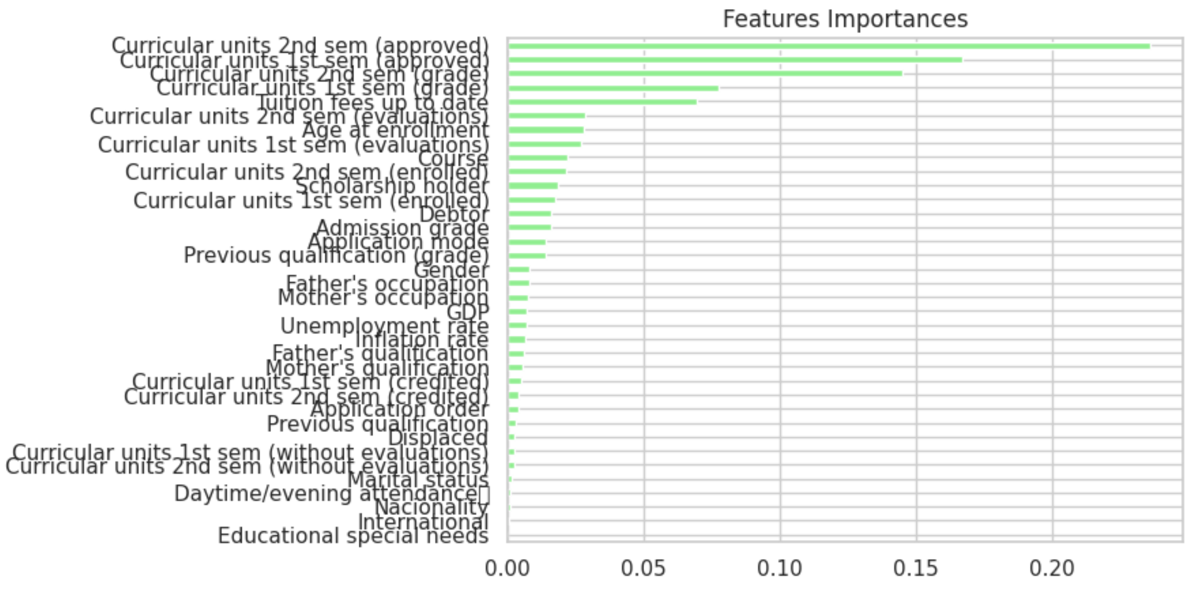


Fig 2. Plot of feature importance ranked by Random Forest

I finally also performed a heatmap correlation of our remaining 10 attributes. This helps to visualise how closely correlated our remaining top 10 attributes are and may reveal some of the patterns our models will identify.

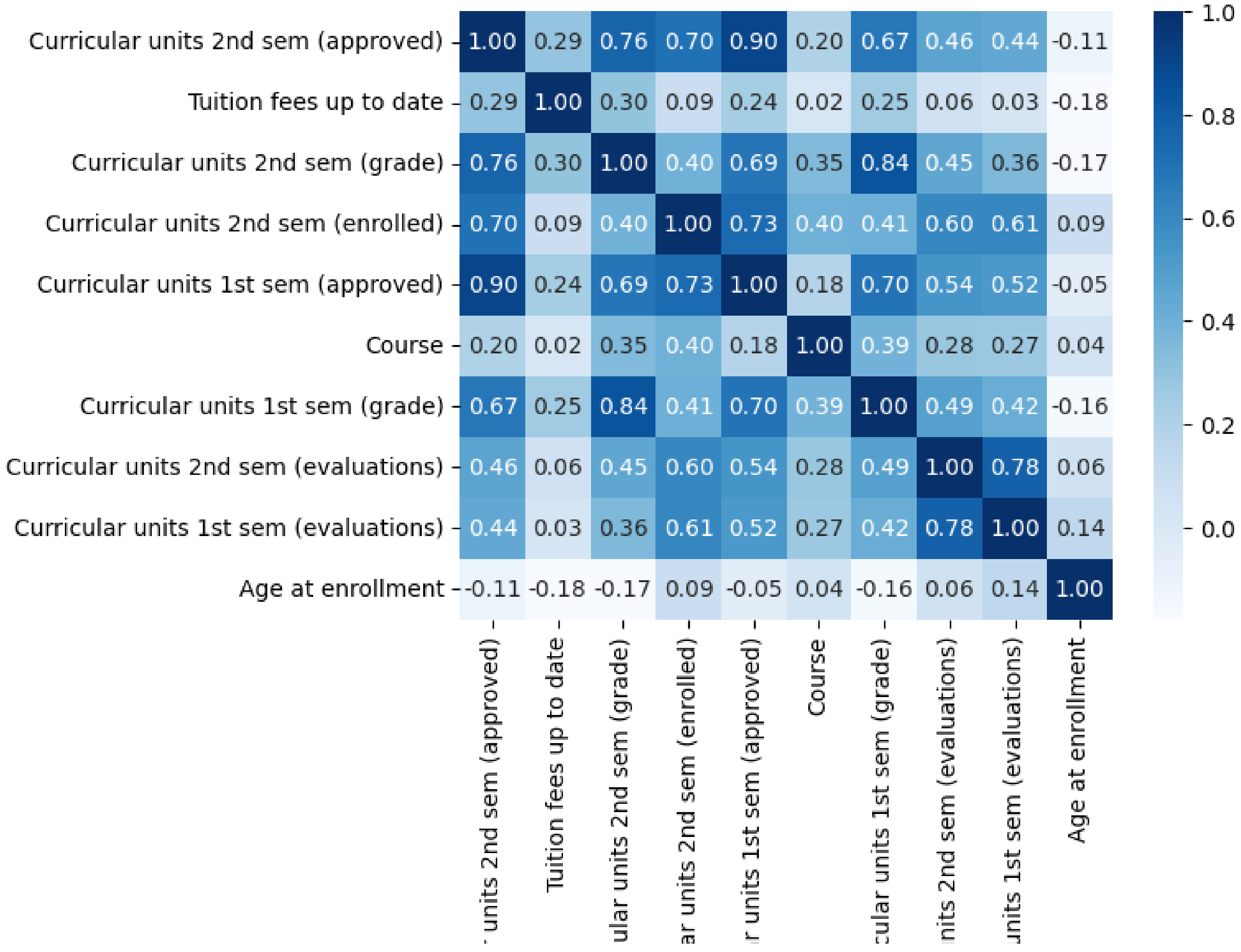
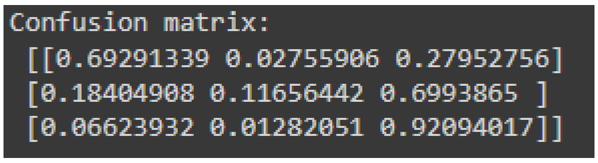


Fig 3. Heatmap correlation of top 10 features

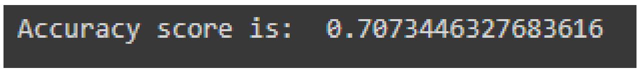
Machine Learning

In this section I will outline the various learning approaches I took in order to find the model with the best accuracy. I utilised 7 different techniques in total, 4 basic and 3 ensemble learning methods.

1. **Base model**
   Logistic regression tries to discern the relationship between our dependent target variable and our independent attributes. I instantiated a logistic regressor as base_model and fit it on the training data. In the interest of clarity I will be basing the models performance primarily from their sk_learn.metrics.accuracy_score and sk_learn.metrics.confusion_matrix. This is due to accuracy being the most important metric in classification problems (*Osisanwo, F. Y et al., 2017*)
   The base model has the following confusion matrix and accuracy score;



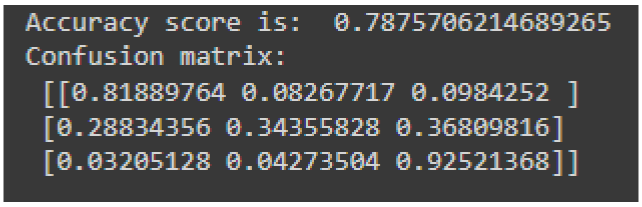
```
Confusion matrix:
 [[0.69291339 0.02755906 0.27952756]
 [0.18404908 0.11656442 0.6993865 ]
 [0.06623932 0.01282051 0.92094017]]
```



```
Accuracy score is:  0.7073446327683616
```

   Our confusion matrix output is 3x3 as we have three possible states for our target variable; Dropout, Enrolled, Graduate.The way to read this output is row 1 and column 1 is Dropout, row 2 and column 2 is Enrolled, row 3 and column 3 is Graduated so (1,1),(2,2) and (3,3) are our true positives.
   We can see that our model has the highest accuracy for predicting Graduates and the worst prediction rate for Enrolled with an overall accuracy of ~70%.

2. **Dimension Scaling of Logistic Regression**
   Following on from our logistic regression model I performed dimension scaling on the X_train and X_test sets. This involves using the sklearn preprocessing module. Dimension scaling is a process that standardises a data set to more closely resemble a normal (Gaussian) distribution. This is a common step in classification problems as classification estimators work best on normally distributed data (*S. B. Kotsiantis, 2007*).



```
Accuracy score is:  0.7875706214689265
Confusion matrix:
 [[0.81889764 0.08267717 0.0984252 ]
 [0.28834356 0.34355828 0.36809816]
 [0.03205128 0.04273504 0.92521368]]
```

   After dimension scaling we can see that our overall accuracy has increased by 8% and also, our models accuracy on enrolled students has more than tripled. Going forward all of our models will use these scaled X_train and X_test sets. In figure 4 below you can see how each prediction affected the running accuracy of each model.

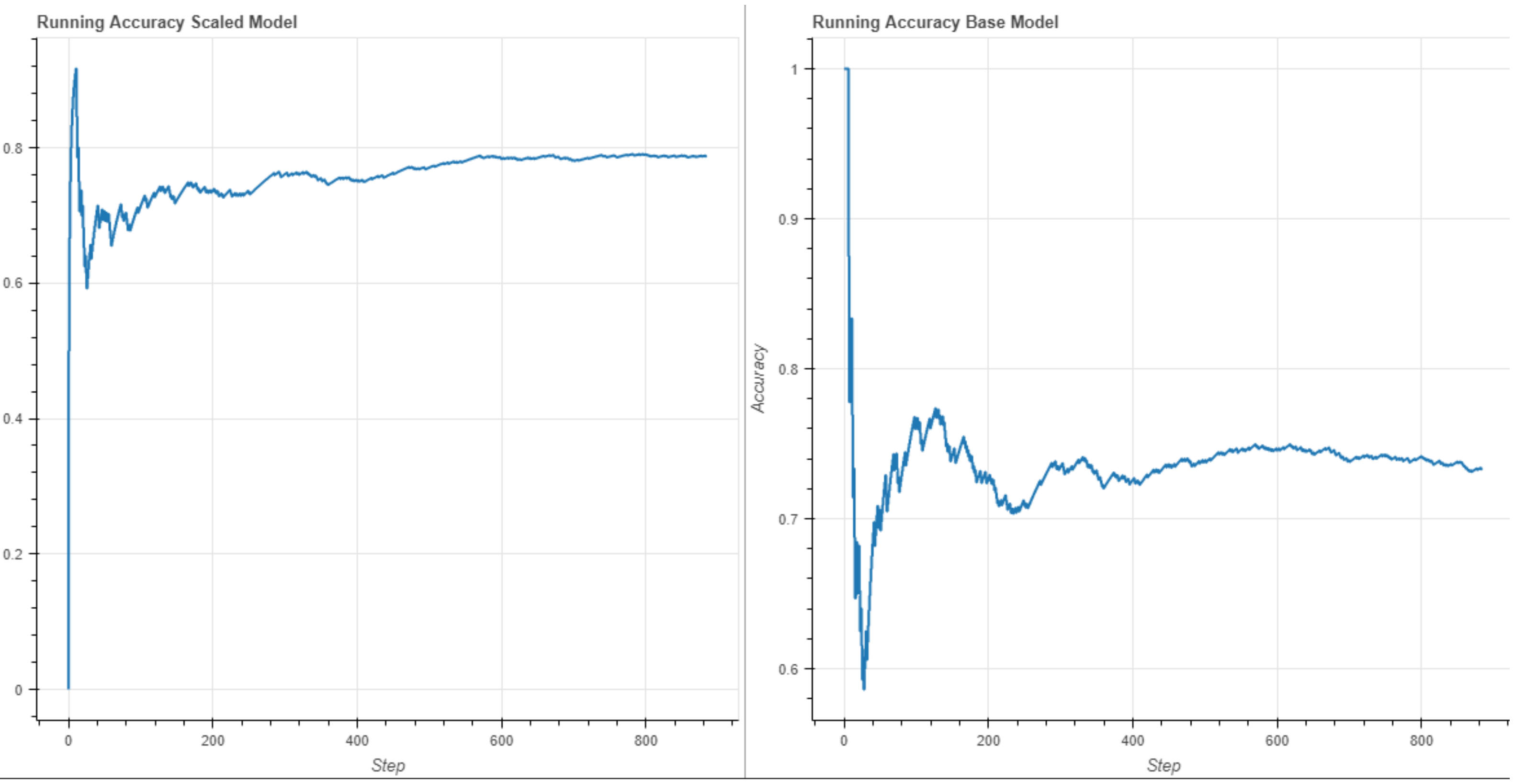


Figure 4. Step line plot showing each decision affecting overall model accuracy

3. Decision Tree

A decision Tree is a supervised learning pattern that tries to build a series of rules for a model to follow based on patterns in the data in order to predict a target variable. This will be the first time I am performing hyperparameter tuning so I trained both a control model with default parameters and a tuned model to demonstrate whether or not hyperparameter tuning increases performance.

My null hypothesis ***H0*** is that hyperparameter tuning does not increase performance.

My alternative hypothesis ***Ha,*** is that it does increase performance

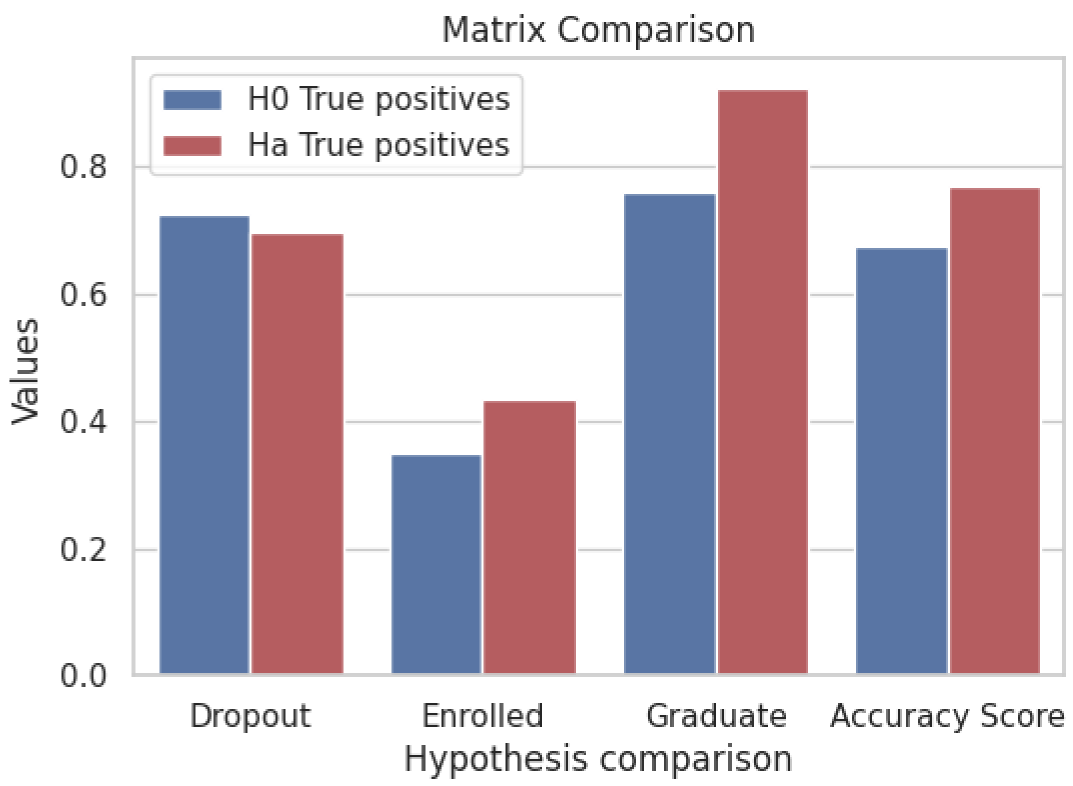


*Fig 5. Comparing results from **H0** and **Ha***

After fitting both the default model and tuned model we can see that the tuned model is the clear winner, though it does not outperform the logistic regression model. Although H0 marginally performs better on Dropout true positives it is safe to say we can reject the Null hypothesis.

4. K Nearest Neighbours

KNN is a supervised learning technique in which data points are grouped into clusters of size k in order to try to identify patterns. My main issue with this Classifier was what value I should use for K as different values would have wildly different accuracies. To identify the best possible value for k I created a for loop that would iterate over the values k=1 to k=20.
Each value of k would be assigned to our classifier and fit to the training data. I plotted the results on a bokeh plot as can be seen in Figure 6

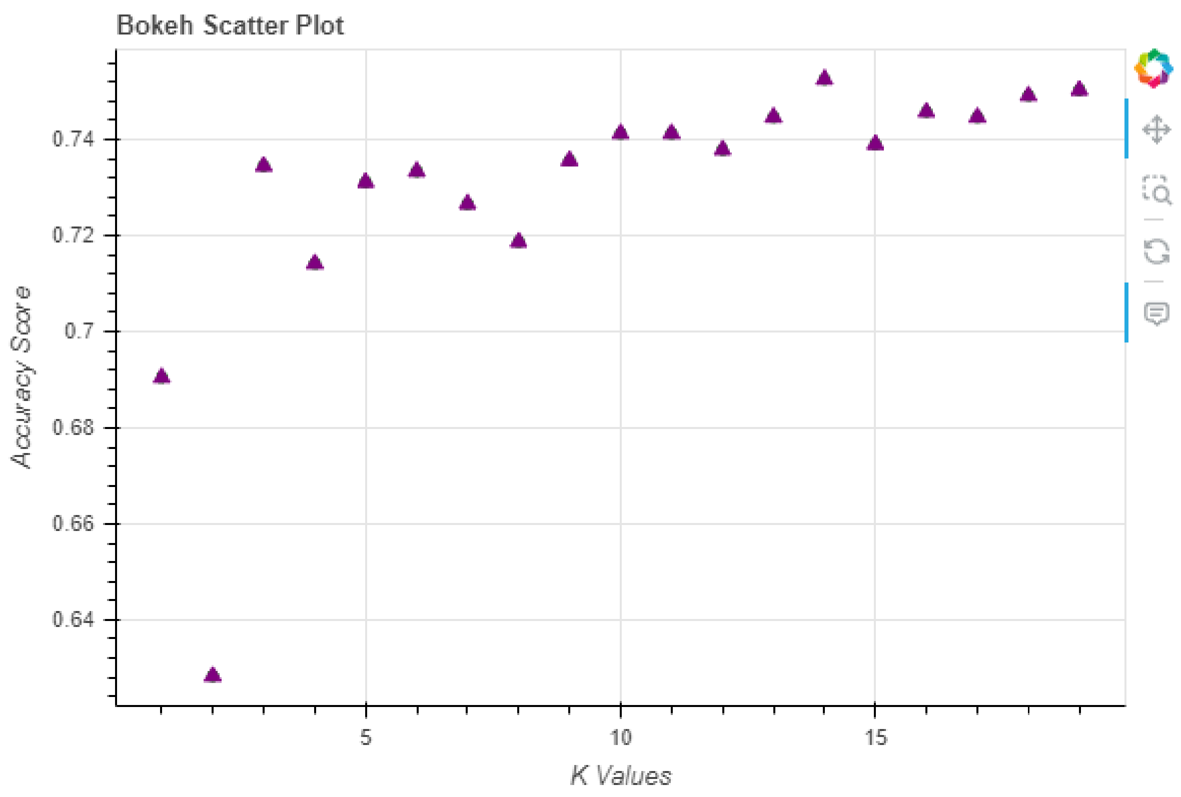


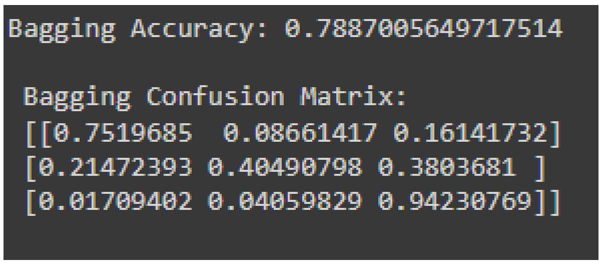
Fig 6. Bokeh plot of K values tested for best accuracy

As can be seen above k=14 is the best value of K so that is what we shall proceed with. However, at just over 75% accuracy it is still weaker than our logistic regressor. With some basic models trained it is now time to move on to a more complex way of training classification models; Ensemble learning

Ensemble learning methods

1. Bagging

Bagging (Bootstrap Aggregating) is an ensemble method that aims to improve performance by reducing the variance in a model's predictions and avoiding overfitting. Bagging involves training a number of smaller models on subsets of our training data and each smaller model contributes to a voting system to give the overall verdict.



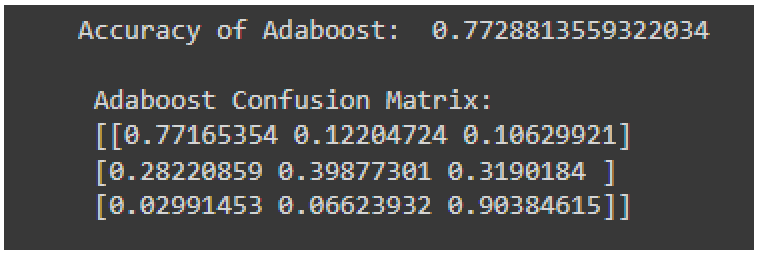
```
Bagging Accuracy: 0.7887005649717514

 Bagging Confusion Matrix:
 [[0.7519685  0.08661417 0.16141732]
 [0.21472393 0.40490798 0.3803681 ]
 [0.01709402 0.04059829 0.94230769]]
```

Our bagging accuracy score is 78.8%, this is very slightly higher than our scaled model accuracy score and is our new best model

2. Adaboost

Boosting is a different ensemble method in which many predictors are trained and each predictor learns from the errors of its predecessors. Adaboost in particular works by training many predictors sequentially and each successor tries to correct errors made by its predecessor. This is done by changing the weights of each training instance. As with Bagging, the predictors all then "vote" on the most likely target variable for each set of attributes.



```
Accuracy of Adaboost:  0.7728813559322034

 Adaboost Confusion Matrix:
 [[0.77165354 0.12204724 0.10629921]
 [0.28220859 0.39877301 0.3190184 ]
 [0.02991453 0.06623932 0.90384615]]
```

In order to demonstrate how Adaboosts successive learning works I created Figure 6 below which shows the learning curve of the model and both the training and validation accuracy. Although our training accuracy is decreasing over a greater data set size and this can be a sign of overfitting, our validation accuracy is steadily increasing so I do not think overfitting is a problem for our model.

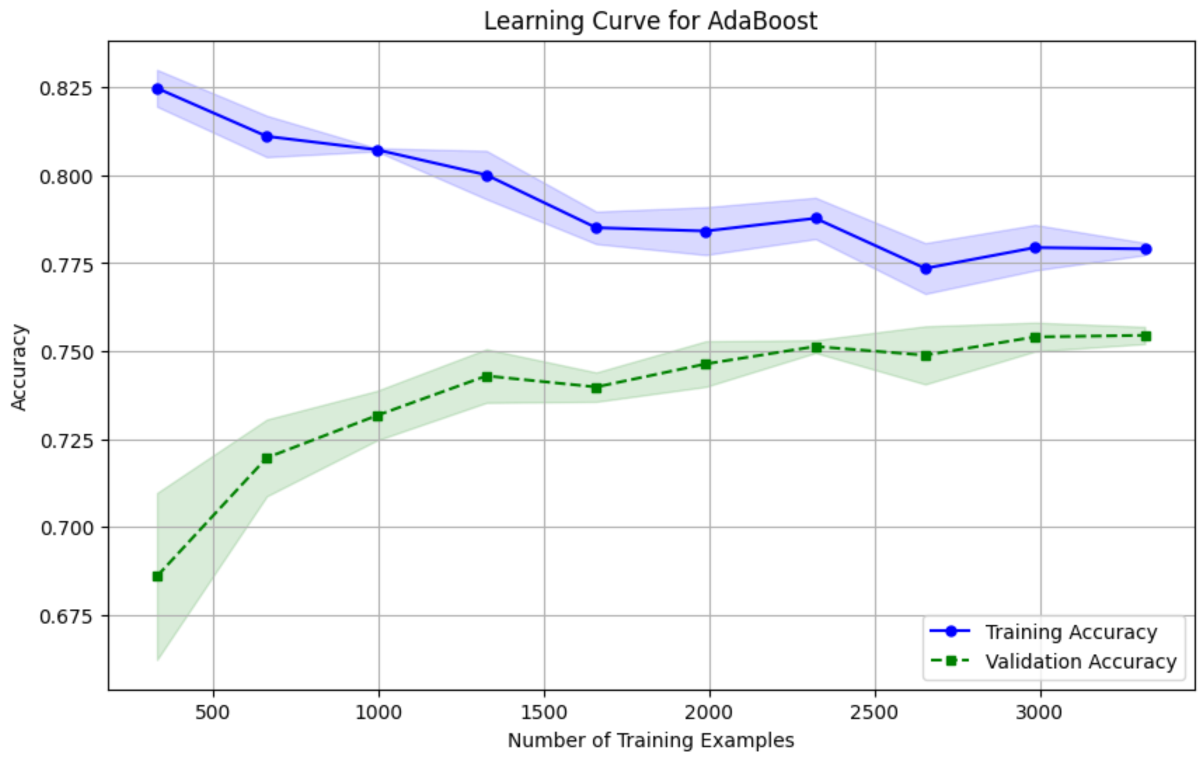
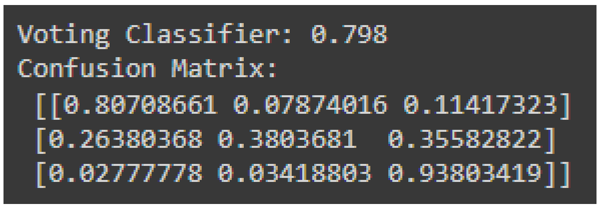


Fig 7. Learning curve of Adaboost

3. Voting Classifier

We now have several different models trained on our data set with varying accuracy. We will now use our final ensemble learning method to combine the strengths of the best 3 from our models. Much like with Bagging and Boosting, a Voting Classifier uses a voting system to inform its choices. However, instead of using an army of weak learner models we are using our 3 best models from our analysis thus far. Two of them are our ensemble methods; bagging and boosting, while the third is our dimension scaled Logistic Regression model. I instantiated the voting classifier with weights equal to the distribution of our target variables summed to 1 (see figure 1.)



```
Voting Classifier: 0.798
Confusion Matrix:
 [[0.80708661 0.07874016 0.11417323]
 [0.26380368 0.3803681  0.35582822]
 [0.02777778 0.03418803 0.93803419]]
```

This results in the voting classifier presenting our overall best model performance of 80% accuracy.

Insights

1. It is important to analyse the distribution of your target variable, see Figure 1. This knowledge can be used to assign weighting to your Classifiers (see Voting classifier) that can compensate for an uneven distribution in your data set
2. Ensemble methods perform much better on average than any of the basic classifiers as evidenced by figure 8 below. All 3 of our ensemble methods are in the top 4 models we trained

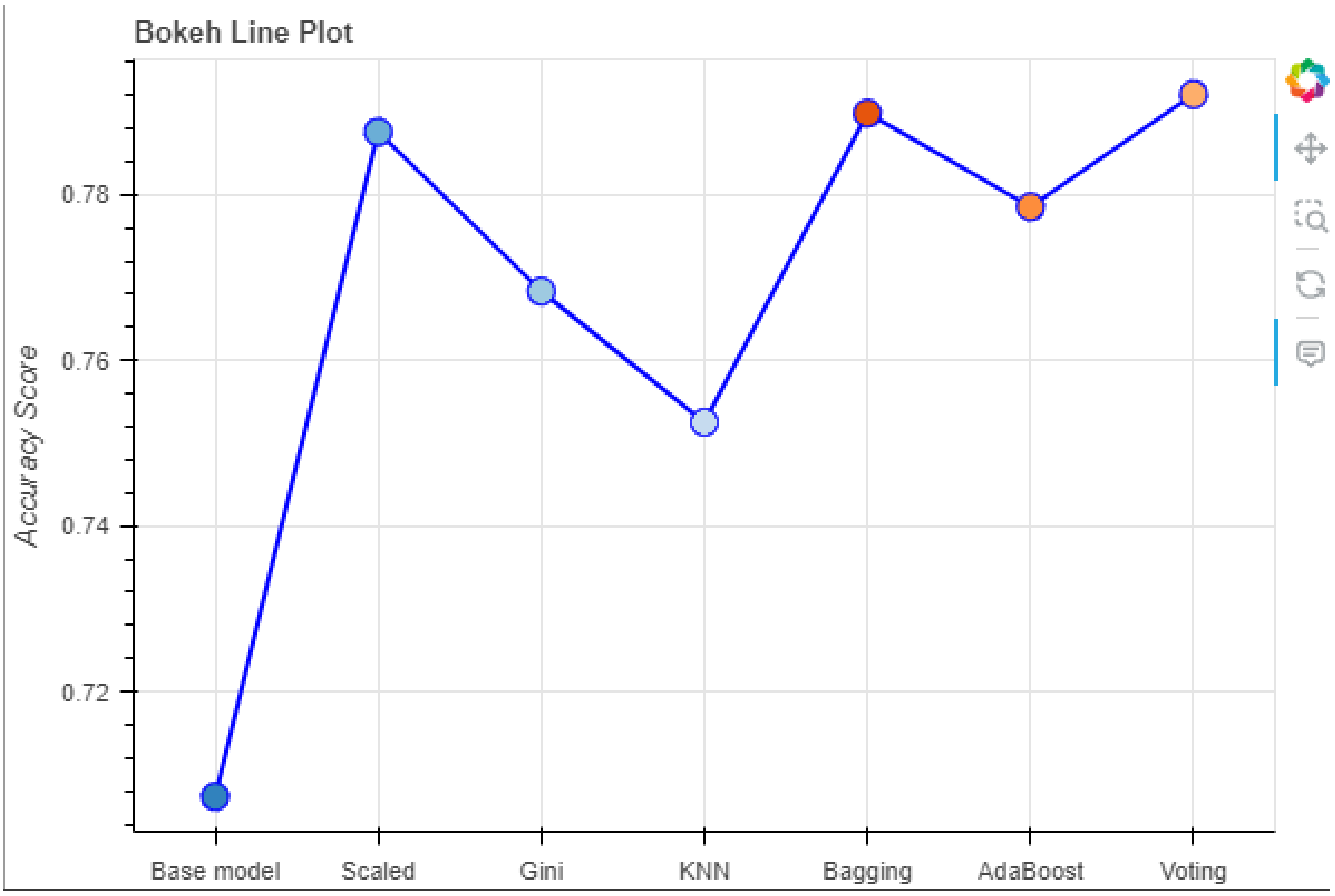


Fig 8. Line plot of each models accuracy score

3. Hyperparameter tuning clearly improves the performance of a model as demonstrated by Figure 5.
4. Certain Classifiers are not suited to all classification problems. KNN, for example, was our worst performing model by a considerable margin on the scaled training sets despite hyperparameter tuning and k-value optimisation, see Figure 6.
5. When considering if a model is overfitted it can be useful to do a learning curve plot. By examining Figure 7 it initially appears that my Adaboost model is overfitted as the training accuracy is decreasing but by also looking at the steady increase in validation accuracy we can see that Adaboost is not overfitting and is actually increasing in performance on unseen data as the data set grows larger
6. Using Random Forest feature importance can be extremely beneficial in cutting through the “noise” in a data set. As can be seen in Figure 2, nearly 90% of our importance weighting falls on only the top 10 out of 36 attributes
7. Heatmaps can be very useful to visualise how closely correlated our attributes are to each other, See Figure 3. This can reveal some of the patterns and trends that our models will find in the data and gives an insight into how they will learn
8. Accuracy is not the only metric to judge a classification model on. In figure 9 below you can see the dotted line occurs at the highest point for metrics where a high score indicates the best model and vice versa. Our voting classifier (pink bar) consistently achieves the best (or tied with another model) for every metric.

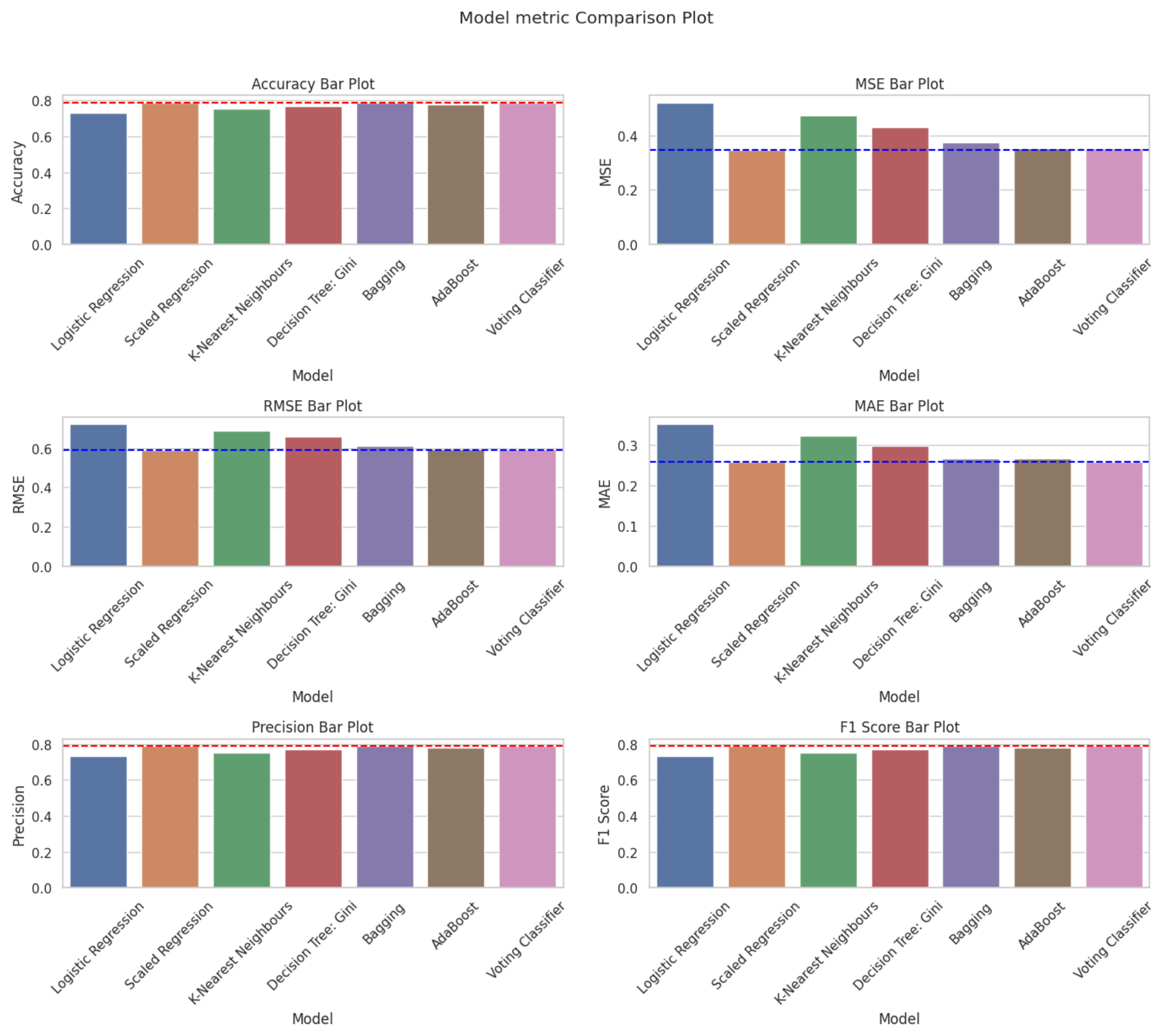


Fig 9. Pairplot of metrics used to determine strength of model

Conclusion

This project aimed to provide a comprehensive overview of classification learning techniques and visualisations for machine learning problems and in that regard I deem it to be successful. From our original base model of Logistic Regression to our Voting Classifier we managed to improve the overall performance by 14% to achieve a prediction accuracy of ~80%. This is a fairly accurate model score and from my investigation of others using the data set on Kaggle 80% is better than many have achieved but I believe it could be improved upon.

The performance of our models is slightly limited by the data set size and also the attributes contained within, I believe that more data to train on would result in a more accurate model based on the learning curve projections from Figure 6, and the high degree of correlation between some of the attributes shown in Figure 3. In addition for a future study, many of the attributes should be re-examined and perhaps additional attributes should be collected. As evidenced by Figure 2, more than 2 thirds of the attributes had essentially no importance whatsoever to our model's predictive capability.

References & Bibliography

S. B. Kotsiantis & I. D. Zaharakis & P. E. Pintelas, 2006, "Machine learning: a review of classification and combining techniques", Artificial Intelligence Review 26 (2006): 159-190.

Charbuty, B., & Abdulazeez, A. (2021). Classification based on decision tree algorithm for machine learning. Journal of Applied Science and Technology Trends, 2(01), 20-28.

Osisanwo, F. Y., Akinsola, J. E. T., Awodele, O., Hinmikaiye, J. O., Olakanmi, O., & Akinjobi, J. (2017), "Supervised machine learning algorithms: classification and comparison. International Journal of Computer Trends and Technology (IJCTT)", 48(3), 128-138.

S. B. Kotsiantis, 2007, "Supervised Machine Learning: A Review of Classification Techniques", Emerging artificial intelligence applications in computer engineering, 160(1), 3-24.

Dataset

https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success