**Juan Manuel Gomez Gonzalez - Natalia Hoyos Velasquez - Ruoxuan Xu**

# ECE 657A Assignment 2: Parameter Tuning

The Assignment aims to use the Bank Marketing dataset and a splice junction on DNA sequence dataset to gain experience on the use of classification methods.

The file bank-additional.zip taken from [1] contains two .csv file (bank-additional.csv and bank-additional-full.csv) and the data set documentation (bank-additional-names.txt). The file bank-additional.csv was used as the data source and the documentation was used to understand the dataset's contents and to identify the column labels.

The splice junction on DNA sequence dataset was given for the assignment.

Python 3.7 was used for this homework, in Windows 10 using Anaconda and Jupyter Notebook. The Pandas, Matplotlib and Scipy.stats libraries for Python were used.

## Libraries used

Pandas was imported to handle the data frames. The Pandas 10-minute tutorial, Pandas Cheat Sheet and Pandas CookBook were used to learn how to use the basic functions of Pandas. These documents can be obtained from [2]. One of Pandas' premise is that it permits the easy creation of a component called Data Frame, which facilitates the manipulation of data and many operations that can be carried out with it.

For the statistical component of the homework, Scipy.stats and numpy were used. The tutorial for Scipy.stats' statistical functions can be found in [3], and Numpy's statistical functions can be seen in [4]. These references indicate the way of using the methods for obtaining the normality test and Z-Score.

Matplotlib's API guide [5] as well as its gallery [6] and Real Python's matplotlib guide [7] were used to understand matplotlib's basics and how to graph using Python. Matplotlib eases the construction and display of figures, including histograms and many types of plots, in an easy to use scheme, similar even to Matlab's own.

Scikitlearn library is also used in the project, this is an open-source machine learning library for Python and contains a helpful preprocessing module that can help us do the normalization and standardization. Sebastian Raschka's blog on the use of standardization techniques is also used [8, 9]. The Seaborn library is also used in order to plot and graph information found using the other libraries. Seaborn's documentation can be found in [10].

Additionally, the following libraries were also used:

- Seaborn was used to build the plots.
- Collections: It is a library that allows the use of more container datatypes for high performance projects. Collections' counter method is used in order to count the number of apparitions an element from an array appears, e.g. there are two blues in the array: (blue, green, red, blue, orange).
- imbalanced-learn: It is a python library that allows for the balance of the different classes of a dataset by using techniques as over-sampling and under-sampling. In this work, imbalanced-learn's SMOTENC method is used, which will be explained further in the document.
- Graphviz: It is a package created by AT&T Labs for the drawing of graphs using DOT language scripts. It's used in the project for the creation of the graphs for the decision trees.

- Pydot: It is an interface for Graphviz that can convert the DOT language used in that library into code usable by python.

## Question I: Revisiting HW4 Bank Classification with New Tools (for dataset A)

### 1. *Load a simple dataset and perform some basic data preprocessing to fill out "unknowns", outliers or other invalid data. Explain what preprocessing was performed and why. Also, change categorical data into numerical features using pandas.get dummies.*

The dataset was inspected and it was noticed that it is stored as comma separated values (csv) file and each value is separated by a semi-colon (;). The file contains a header with the column names.

The file was taken directly from the dataset URL and loaded as a pandas data frame with parameters for the header and the delimiter were specified to 0 and ";" respectively. The first 5 rows can be seen in the table below.

*Table 1. First five rows of the dataset*

| age | job | marital | education | ... | cons.conf.idx | euribor3m | nr.employed | y |
|---|---|---|---|---|---|---|---|---|
| **30** | blue-collar | married | basic.9y | ... | -46.2 | 1.313 | 5099.1 | no |
| **39** | services | single | high.school | ... | -36.4 | 4.855 | 5191 | no |
| **25** | services | married | high.school | ... | -41.8 | 4.962 | 5228.1 | no |
| **38** | services | married | basic.9y | ... | -41.8 | 4.959 | 5228.1 | no |
| **47** | admin. | married | university.degree | ... | -42 | 4.191 | 5195.8 | no |

The dataset is related to the direct marketing campaigns based on phone calls of a banking institution located in Portugal. The dataset contains 4119 instances and 21 client and contact attributes. The goal attribute (variable y) describes if the client will subscribe to a term deposit.

The dataset is highly unbalanced having a total of 4119 samples from which 10.95% are of class 'yes' and 89.05% are of class 'no'.

The documentation found in [1] states that the duration column is not known before a call is performed and therefore it should be used for benchmark purposes only. It was deleted since it was advised to discard this column if the intention is trying to create a realistic predictive model.

#### a) Missing Values

The data frame has missing values denoted as “unknown” that have to be changed to Not a number (NaN) for an easier analysis. It can also be checked for rows that only contain NaN values and delete them, as these samples do not add any information to the dataset.

It can be seen in the information in Table 2 that no variable has a significant number of missing values that would require to drop the whole column.

The features that contain missing values are “job”, "marital", “education”, "default", "housing" and "loan". Replacing the missing values of these attributes with the mean would not make sense since these are unordered categorical values and calculating the mean will prove to be impossible. Two strategies could be used in this case: to treat the missing values as a separate class or to replace them with the mode.

In this assignment they will be treated as a separate class since replacing the missing values with the mode could potentially add noise to the model.

*Table 2. Missing values by feature*

| Feature | NaN values | % of NaN values |
|---|---|---|
| age | 0 | 0% |
| **job** | 39 | 1% |
| **marital** | 11 | 0% |
| **education** | 167 | 4% |
| **default** | 803 | 19% |
| **housing** | 105 | 3% |
| **loan** | 105 | 3% |
| contact | 0 | 0% |
| month | 0 | 0% |
| day_of_week | 0 | 0% |
| campaign | 0 | 0% |
| pdays | 0 | 0% |
| previous | 0 | 0% |
| poutcome | 0 | 0% |
| emp.var.rate | 0 | 0% |
| cons.price.idx | 0 | 0% |
| cons.conf.idx | 0 | 0% |
| euribor3m | 0 | 0% |
| nr.employed | 0 | 0% |
| y | 0 | 0% |

### b) Handling Outliers

Removing outliers from numerical data:

The boxplots of the numerical features can be seen in Figure 1. This figure shows the outliers of each feature as dots outside the boxplot.

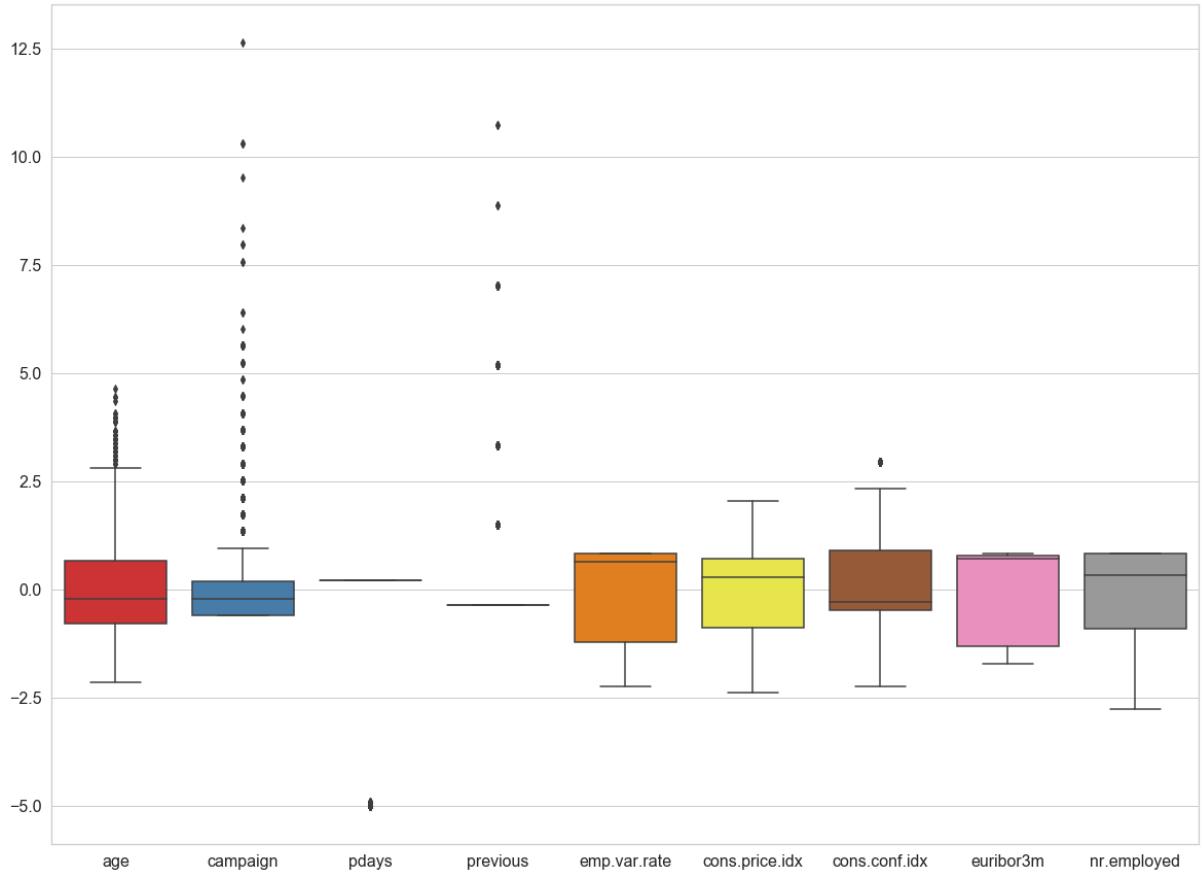


*Figure 1. Boxplot of numerical features*

For the removal of numerical outliers, the Z-Score technique is the rule of thumb. The Z-Score uses the principle that, once the data has been centered and rescaled, any values outside of an n amount of standard deviations from the mean is an outlier [11]. A graphical example can be seen in Figure 2.

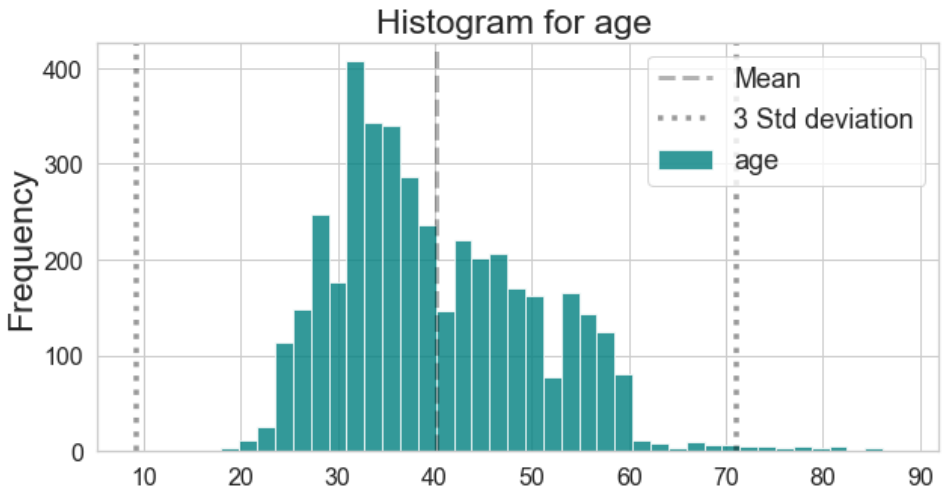


*Figure 2. Histogram of feature "age" showing its mean and 3 standard deviations range.*

At the beginning, 321 outliers were deleted from the sample. Of this, 120 (37%) were of class 'yes'. Due to the highly unbalanced nature of the dataset, it is possible that many samples of class 'yes' were detected as outliers due to the lack of information. It was decided to delete the outliers detected by the Z-score technique except if they were of class 'yes'. In this way, a total of 201 samples were deleted.

Removing outliers from categorical data:

After dealing with the numerical outliers it was questioned if the categorical variables might also have outliers. To solve this concern, it is first needed to define what an outlier in a categorical data might be since an unordered categorical data does not have extreme values. Categorical outliers represent those samples which have values that are infrequent in the dataset [12].

When analyzing the frequency of the categorical values, the features 'Education' and 'Default' drew attention as seen in Table 3 and Table 4.

*Table 3. Education value's frequency*

| Education | Frequency |
|---|---|
| university.degree | 1264 |
| high.school | 921 |
| basic.9y | 574 |
| professional.course | 535 |
| basic.4y | 429 |
| basic.6y | 228 |
| unknown | 167 |
| **illiterate** | **1** |

*Table 4. Default values' frequency*

| Default | Frequency |
|---|---|
| no | 3315 |
| unknown | 803 |
| **yes** | **1** |

It can be observed in Table 3 that there is only one sample with the value 'illiterate' in the 'Education' column. This sample is considered an outlier and is deleted from the dataset. Something similar happens with the 'default' feature. Table 4 shows that the most common value of the 'default' feature is 'no' followed by an unknown value and only one sample with 'yes'. This sample is also considered an outlier and deleted from the dataset.

After all the modifications, the dataset is comprised of 3916 samples.

The dataset is almost done with the preprocessing and ready to start the training of the classification algorithms. One of the final steps is to translate the categorical features into numerical values that the model can understand. function get_dummies converts these non-numerical variables into indicator variables. This function creates one column for each possible instance of a given feature and then fills the columns with zeros and ones, which specify the presence or absence of the corresponding instance.

## *2. Divide data into train and test portions, justify your split decision*

Before splitting the data into train and test portions it is necessary to fix the unbalance on the classes of the dataset. The SMOTENC function uses the Synthetic Minority Oversampling Technique (SMOTE) method which does an over-sampling of the class that is under-represented [13]. One special characteristic of the SMOTENC

function is that it is able to handle mixed data such as continuous and categorical. In the case of the categorical data, the samples generated belong to the same categories as originally presented without any interpolation.

The numerical data is interpolated from the samples present in the dataset and their values are created as floats. The resulting values are formatted to have the same number of decimals as the original dataset (e.g. the age column does not have decimals)

Prior to applying the SMOTENC function, the dataset contained 3465 (89%) 'no' samples and 451 (11%) 'yes' samples. After applying the function, the dataset is comprised of 3465 'no' samples and the same number of 'yes' samples.

After the preprocessing is finished, the dataset is ready to start the training of the classification algorithm. First it needs to be split into training and testing sets in order to train the algorithm and test its performance.

Choosing the splitting data ratio is a difficult task. A training set that is too small would underfit the model due to the lack of information and a small testing set would not be robust enough to test every possible scenario. For big datasets, 80/20 is quite a commonly occurring ratio and is referred to as the Pareto principle.

The dataset is divided using an 80/20 partition to obtain the train set and test set. The min-max scaler estimator is fitted with the training set and scales both the training and test set.

3. *Apply classification using Decision Trees (DT), Random Forests (RF) and Neural Networks (NN) and run using standard libraries in your language or choice. Indicate the classification properties (example: depth of tree, size of neural network, ensembles for RF) you have chosen and justify. Briefly describe the algorithms employed by the libraries you are using (example ID3, C4.5, C5.0 and CART for DT). Make sure to run the classification on 10 features or more*

In order to tune the hyperparameters, various experiments have to be performed fitting the model with different values, and evaluating the result with an appropriate method based on the consequences of misclassifying samples. In this particular case, the cost of misclassifying a "no" sample as a "yes" (false positive) has a low cost, as it would mean calling a client without achieving a successful response. On the other hand, misclassifying a "yes" sample as a "no" (false negative) means that the bank is missing the opportunity of detecting a client that would say "yes" to the subscription of a term deposit. As a result of the above, the metric to measure the performance of the model and to decide on the best hyperparameter is the recall which measures the ability of the classifier to find all the positive samples.

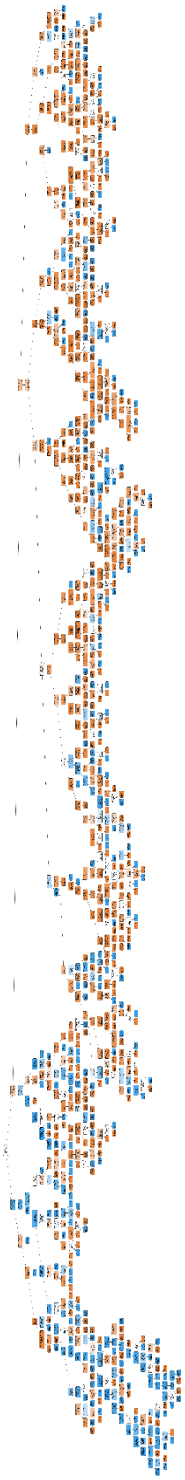

*Figure 3. Decision Tree with default hyperparameters.*

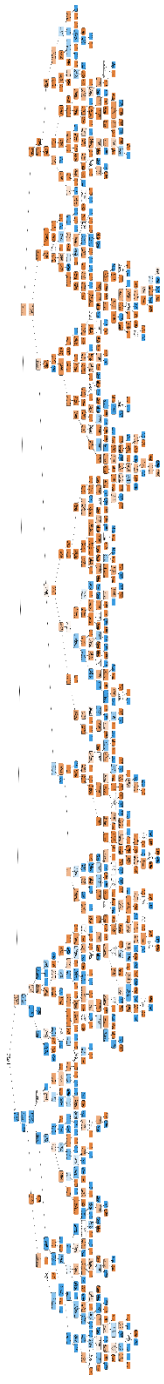

*Figure 4. Decision Tree with tuned hyperparameters*

### a) Decision Tree

To obtain a decision tree classification model the Scikit learn module was used, which employs the CART decision tree. The CART decision tree technique is a procedure developed in 1984, which is capable of analyzing both continuous and nominal attributes as predictors or targets for the decision tree. The trees grow following the basic rules of the other decision trees algorithms (ID3, C4.5, etc.) and would afterwards need pruning if left to grow to their maximum depth [14]. The pruning is made via a method called cost-complexity pruning, where the previously made splits get pruned based on the least contributing splits first, all the way up to the roots. It also differentiates itself from the other algorithms in that it does not use training data previously utilized to grow the trees in order to select the best performing one. Instead, it used cross-validation to pick the "honest" tree [15]. If this cross-validation is not performed, the algorithm does not pick the best tree in the sequence, compared to the C4.5 classifier. The CART algorithm also includes methods for automatic class balancing and the handling of missing values, while being one of the pioneering techniques for the use of cross-validation to assess the performance of the trees during the pruning process [15].

First, the baseline performance of the model is obtained by training the decision tree with its default values using the samples from the train set and evaluating with the test set. Table 5 shows the classification report of the performance of this model and Figure 3 shows the graph of the decision tree.

The hyperparameters of the decision tree model that need to be tuned are:

- Max_depth: represents how deep the tree can continue, allowing more splits. The maximum depth was tested between 2 and 40.
- Class_weight: represents the weights associated with the classes. Can be None or 'balanced'.
- Criterion: indicates the function used to measure the quality of a split. Can be Gini or Entropy.

One way to tune these hyperparameters is using a greedy algorithm that chooses the best value for the first hyperparameter, fixes it, looks up the best value for the second hyperparameter and so on. The tuning is done by using the training set to do a cross-validation and the best value is the one that maximizes the recall. After tuning, the best hyperparameters found were:

- Max_depth: 20
- Class_weight: None
- Criterion: 'entropy'

Figure 4 shows the graph of the decision tree. It shows that the depth has changed from 24 to 21 and the node count from 963 to 843 compared to the baseline tree.

Table 6 shows the classification report of the performance of this model when it is trained with the training and validation set and evaluated with the testing set. It can be seen that the recall for both classes and the precision of the 'yes' class have an increase of 0.1 points or more without decreasing the other measures.

*Table 5. Performance of the baseline decision tree model*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **no** | 0.91 | 0.89 | 0.90 | 693 |
| **yes** | 0.90 | 0.91 | 0.90 | 693 |
| **micro avg** | 0.90 | 0.90 | 0.90 | 1386 |
| **macro avg** | 0.90 | 0.90 | 0.90 | 1386 |
| **weighted avg** | 0.90 | 0.90 | 0.90 | 1386 |

*Table 6. Performance of the tuned decision tree model*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **no** | 0.91 | 0.91 | 0.91 | 693 |
| **yes** | 0.91 | 0.90 | 0.91 | 693 |
| **micro avg** | 0.91 | 0.91 | 0.91 | 1386 |
| **macro avg** | 0.91 | 0.91 | 0.91 | 1386 |
| **weighted avg** | 0.91 | 0.91 | 0.91 | 1386 |

### b) Random Forest

Scikit-learn's Random Forest Classifier Algorithm is based on the same CART decision tree technique, with the addition of subsampling the dataset to increase its accuracy and reduce overfitting. These sub-samples are the same size as the original input sample size but they are drawn from the pool with replacement, which is also known as bootstrapping [16].

Random forest classification is an estimator based on an ensemble of decision trees created with subsamples of the training data. Each tree generates its own decision and afterwards a consensus is made based on the average of the decisions of the different trees, hence the name "Random Forest" [17].

First, the baseline performance of the model is obtained by training the random forest with its default values using the train set and evaluating with the test set. Table 7 shows the classification report of the performance of this model.

For random forest the hyperparameters include:

- N_estimators: number of decision trees in the forest. The number of estimators was tested between 2 and 150.

- Max_depth: represents how deep the tree can continue allowing more splits. The maximum depth was tested between 2 and 40.
- Criterion: indicates the function used to measure the quality of a split. Can be Gini or Entropy.
- Bootstrap: Bootstrapping is a type of resampling where smaller samples of the same size are repeatedly selected, with replacement, from a single original set of samples. By default, bootstrap is on.

After tuning based on the recall, the performance of the model obtained can be seen in Table 8.

The best hyperparameters found were:

- N_estimators: 102
- Max_depth: 13
- Criterion: 'entropy'
- Bootstrap: True

It can be seen in **Error! Reference source not found.** that this model has an increase in the "no" precision, "yes" recall and f1-score.

*Table 7. Performance of the baseline random forest model*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **no** | 0.91 | 0.96 | 0.94 | 693 |
| **yes** | 0.96 | 0.91 | 0.93 | 693 |
| | | | | |
| **micro avg** | 0.94 | 0.94 | 0.94 | 1386 |
| **macro avg** | 0.94 | 0.94 | 0.94 | 1386 |
| **weighted avg** | 0.94 | 0.94 | 0.94 | 1386 |

*Table 8. Performance of the tuned random forest model*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **no** | 0.92 | 0.96 | 0.94 | 693 |
| **yes** | 0.96 | 0.92 | 0.94 | 693 |
| | | | | |
| **micro avg** | 0.94 | 0.94 | 0.94 | 1386 |
| **macro avg** | 0.94 | 0.94 | 0.94 | 1386 |
| **weighted avg** | 0.94 | 0.94 | 0.94 | 1386 |

### c) Neural Networks

A Neural network (NN) is an interconnected group of simple processing units that use mathematical models to process information. For the NN classifier, Scikit-learn's application uses the Multi-layer Perceptron, which trains using backpropagation, specifically using a form of gradient descent calculated using backpropagation. For the classification of samples, it uses the cross-entropy function as a loss function for its minimization, returning a vector of probability estimates per sample [18]. This in turn makes sure that the algorithm learns from its mistakes faster compared to other functions, like the quadratic [19]. Scikit-learn's NN classifier also supports multi-class classification through the use of the normalized exponential function as the output function, which tries to normalize an input vector by diving its elements by the sum of the exponential of each of its elements, ensuring that the sums of the components of the output vector equals one [20]. Furthermore, this specific application of NN supports multi-label classification through the use of the logistic function [18].

First, the baseline performance of the model is obtained by training the NN with its default values using the train set and evaluating with the test set.

Table 9 shows the classification report of the performance of this model.

There is no standard methodology for tuning a neural network and therefore grid-search is a good tool to select the best hyperparameters. The grid_search function systematically works through multiple combinations of parameter values, cross validates them and determines which one gives the best performance.

The hyperparameters of the NN model that need to be tuned are:

- Hidden_layer_sizes: the number of neurons in each hidden layer. There are some recommendations to consider when selecting the size of the hidden layer:
    - between the input and output layer size (2 and 54 in this case)
    - 2/3 (inputs + outputs): 37.33
    - never larger than twice the size of the input layer (108) [21]
- Activation: Represent the activation function of the hidden layer. Can be 'identity', 'logistic', 'tanh', or 'relu'.
- Solver: Determines the solver weight optimization. Can be 'lbfgs', 'sgd' or 'adam'.

The first grid search was done with the parameters

{'hidden_layer_sizes': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110],

'activation': ['identity', 'logistic', 'tanh', 'relu'],

'solver': ['lbfgs', 'sgd', 'adam']}

And found that the best parameters were

{'activation': 'logistic', 'hidden_layer_sizes': 50, 'solver': 'lbfgs'}

After obtaining this result, a grid further refined from the previous results can be used with the parameters as follows:

{'hidden_layer_sizes': [40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59],

'activation': ['identity', 'logistic', 'tanh', 'relu'],

'solver': ['lbfgs', 'sgd', 'adam']}

The best parameters found

{'activation': 'logistic', 'hidden_layer_sizes': 42, 'solver': 'lbfgs'}

It can be seen in
Table 10 that this model does not have an increase in its performance, compared with the default values.

*Table 9. Performance of the baseline neural network model*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **no** | 0.92 | 0.92 | 0.93 | 693 |
| **yes** | 0.92 | 0.92 | 0.92 | 693 |
| | | | | |
| **micro avg** | 0.92 | 0.92 | 0.92 | 1386 |
| **macro avg** | 0.92 | 0.92 | 0.92 | 1386 |
| **weighted avg** | 0.92 | 0.92 | 0.92 | 1386 |

*Table 10. Performance of the tuned neural network model*

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **no** | 0.91 | 0.92 | 0.92 | 693 |
| **yes** | 0.92 | 0.91 | 0.91 | 693 |
| | | | | |
| **micro avg** | 0.91 | 0.91 | 0.91 | 1386 |
| **macro avg** | 0.91 | 0.91 | 0.91 | 1386 |
| **weighted avg** | 0.91 | 0.91 | 0.91 | 1386 |

4. *Create a few plots of your model on the test data, two of the data dimensions at a time, indicating the predicted elements of each class using different colors or shapes. You may need to try plotting various pairs of dimensions to see which provide some interesting result. Be sure to label your axis and legend. Why is separation better on some plots than others?*

It seems that even if there is no clear distinction between the different classes of the dataset, it is possible to discern in some of the pair of variables a slight separation. In the campaign and age plot shown in Figure 5, for example, it is possible to separate some values of "no" from the majority of the "yes" class if a horizontal line is traced through 0.4 or 0.6 of campaign. A vertical line through 0.7 or 0.8 of age would also separate some values of the "yes" class.

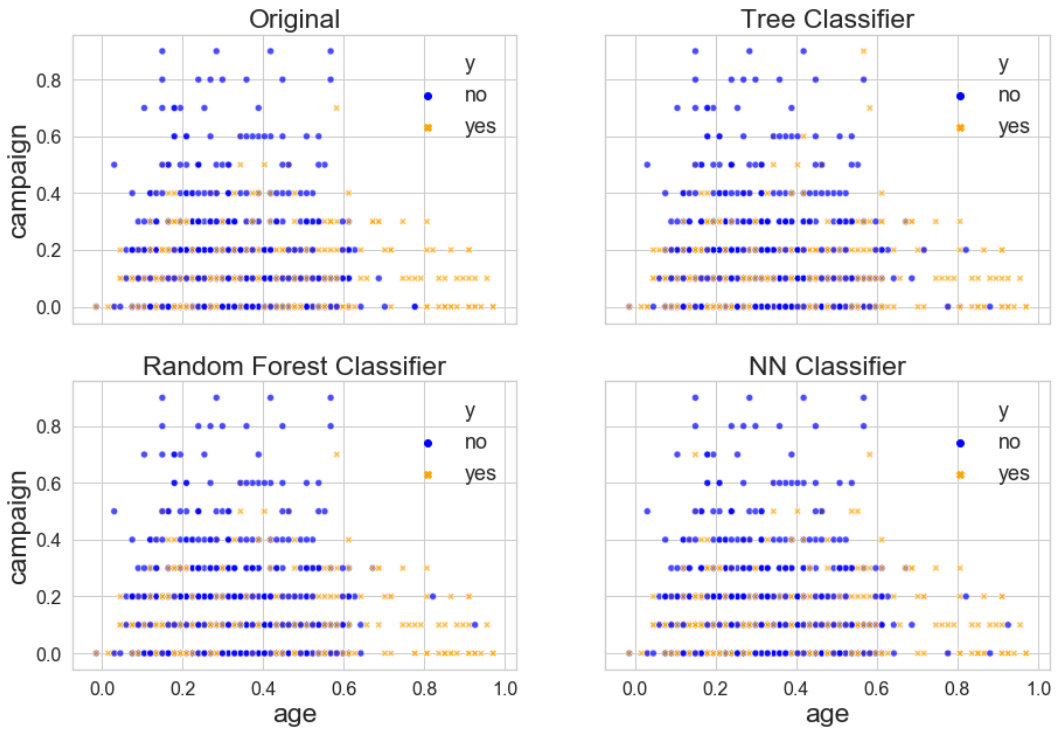


*Figure 5. Age vs Campaign with various models*

For the campaign and cons.price.idx plot shown in Figure 6, it is also possible to separate some "no" values from the rest by tracing a horizontal line through 0.6 of campaign. Nonetheless, for this instance it is not possible to trace a vertical line as in the previous plot.

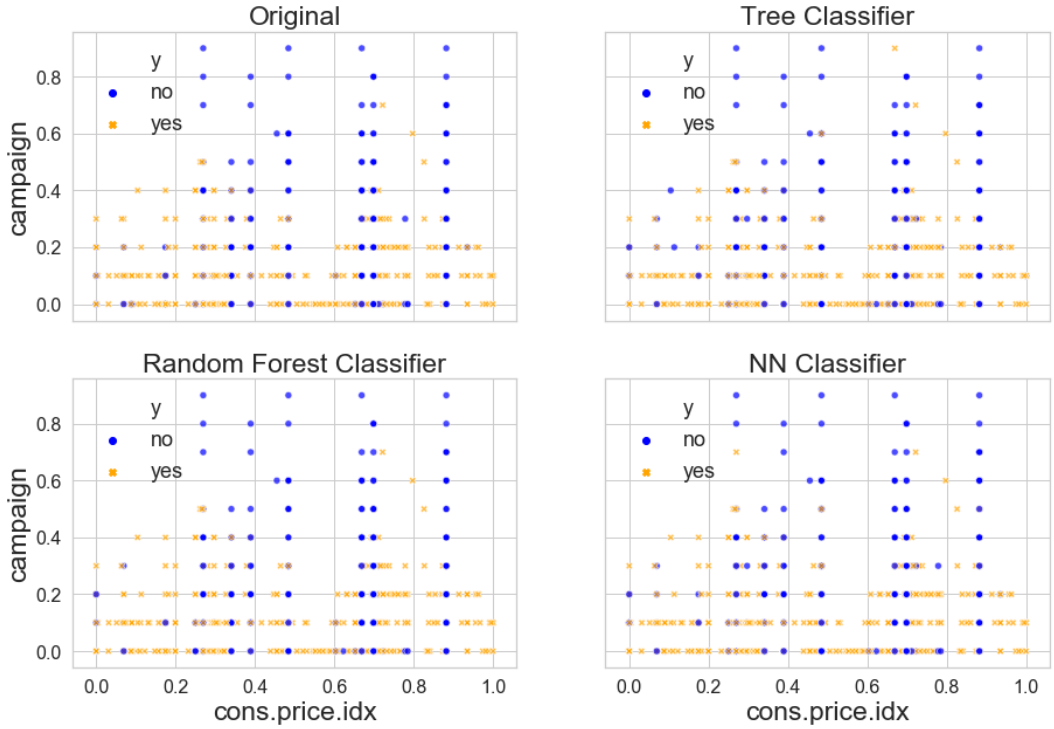


*Figure 6. campaign vs cons.price.idx with various models*

Figure 7 shows the third case where it seems it is not possible to trace a vertical line similar to the first plot, but it seems it is possible to trace a diagonal line that goes from campaign=0.6 and cons.conf.idx=0 to campaign=0.2 and cons.conf.dix=1.0, separating some "no" values from the rest.

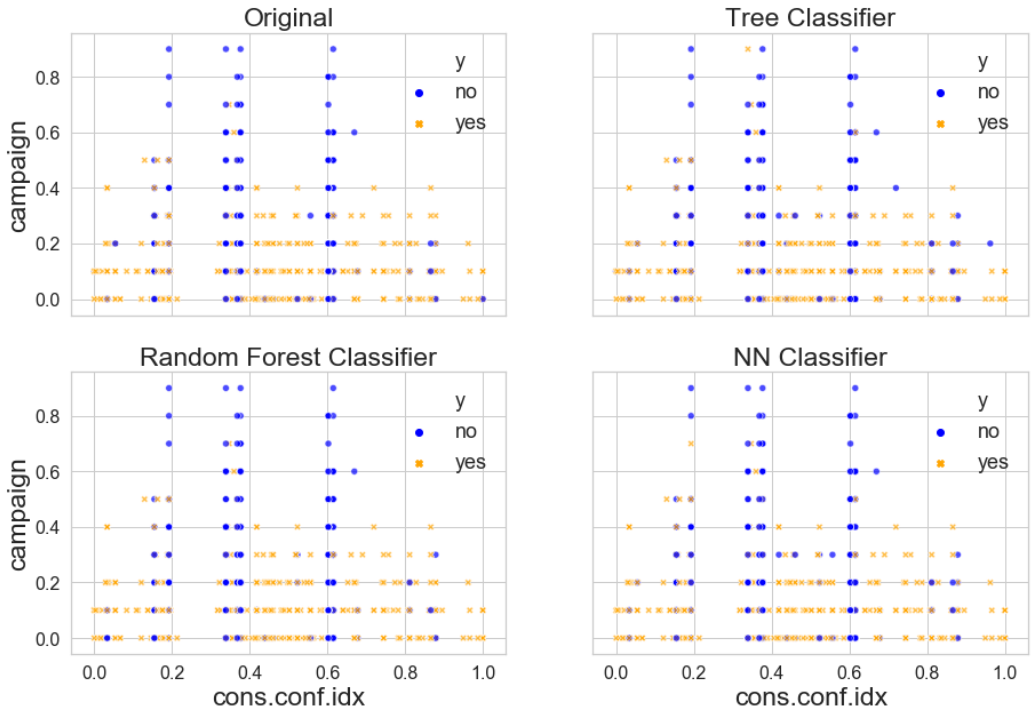


*Figure 7. campaign vs cons.conf.idx with various models*

In the fourth case shown in Figure 8, it can be seen that the "no" class is more concentrated in some particular values of con.price.idx and cons.conf.idx. It seems that it is not possible to separate them using a straight line between a pair of coordinates, nonetheless the two outcomes seem to be discriminable using non-linear boundaries.

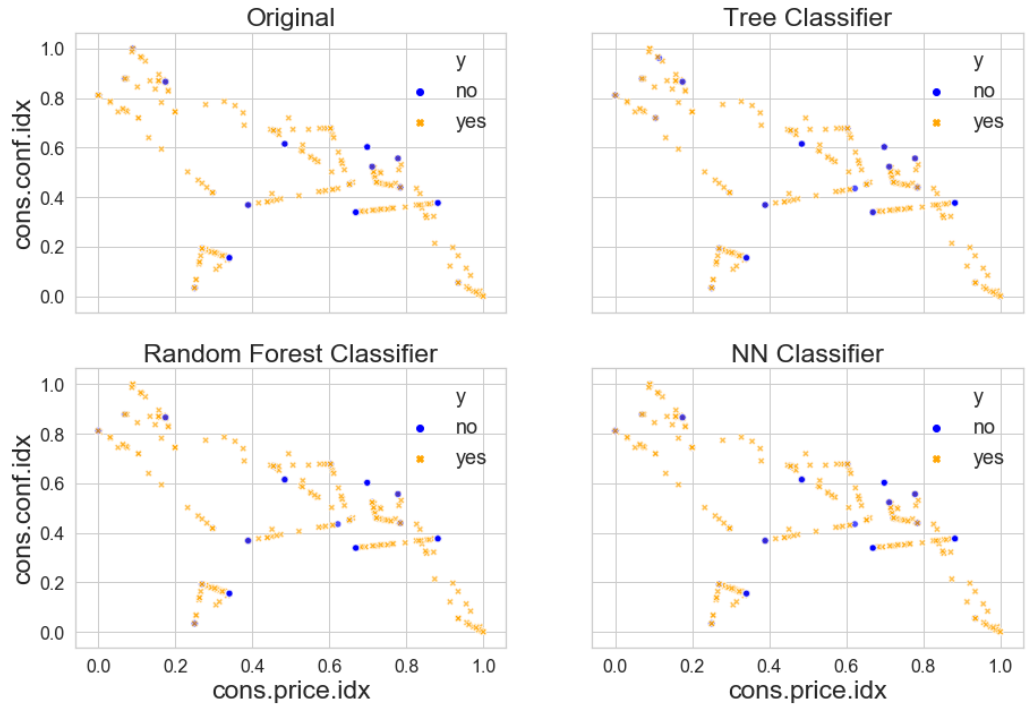


*Figure 8. cons.conf.idx vs cons.price.idx with various models*

It can be seen that the separation is better in some plots than in others because some attributes have a bigger influence in the class prediction than others, making them better at making a distinction between the outputs.

5. *Produce a table with the true/false positive/negative metrics as well as accuracies. Compare the values using bar charts. HINT: classification report from sklearn.metrics.*

*Table 11. Confusion Matrix Structure*

| | | Predicted | |
|---|---|---|---|
| | | no | yes |
| Real | no | True Negative | False Positive |
| | yes | False Negative | True Positive |

The confusion matrix of a model allows to distinguish the true/false positive/negative values of the results. The general structure of a confusion matrix can be seen in Table 11.

*Table 12. Confusion matrix of the models: Decision tree, random forest and neural network.*

| | Decision Tree | | Random Forest | | Neural Network | | |
|---|---|---|---|---|---|---|---|
| | no | yes | no | yes | no | yes | Total |
| no | 633 | 60 | 665 | 28 | 636 | 57 | 693 |
| yes | 66 | 627 | 57 | 636 | 61 | 632 | 693 |
| Total | 699 | 687 | 722 | 664 | 697 | 689 | 1386 |
| | | | | | | | |
| Accuracy | 90.9% | | 93.9 | | 91.5 | | |
| Precision | 0.906 | 0.913 | 0.921 | 0.958 | 0.912 | 0.917 | |
| Recall | 0.913 | 0.905 | 0.960 | 0.918 | 0.918 | 0.912 | |
| f1-score | 0.909 | 0.909 | 0.940 | 0.937 | 0.915 | 0.915 | |

Table 12 shows the confusion matrix of the decision tree, random forest and neural networks models.


*Figure 9. Bar charts showing the true/false negative/positive values from the confusion matrix*

Figure 9 shows four bar charts comparing the confusion matrix values for the three models. The best model would be the one that maximizes the true negatives/positives and minimizes the false negatives/positives. Figure 10 shows the precision, recall and f1-score performance per classifier. The ideal model would be the one that maximizes all of them, the best performer would have to be chosen in a case-by-case scenario depending on the application. In the present case it would be the one that maximizes the recall.

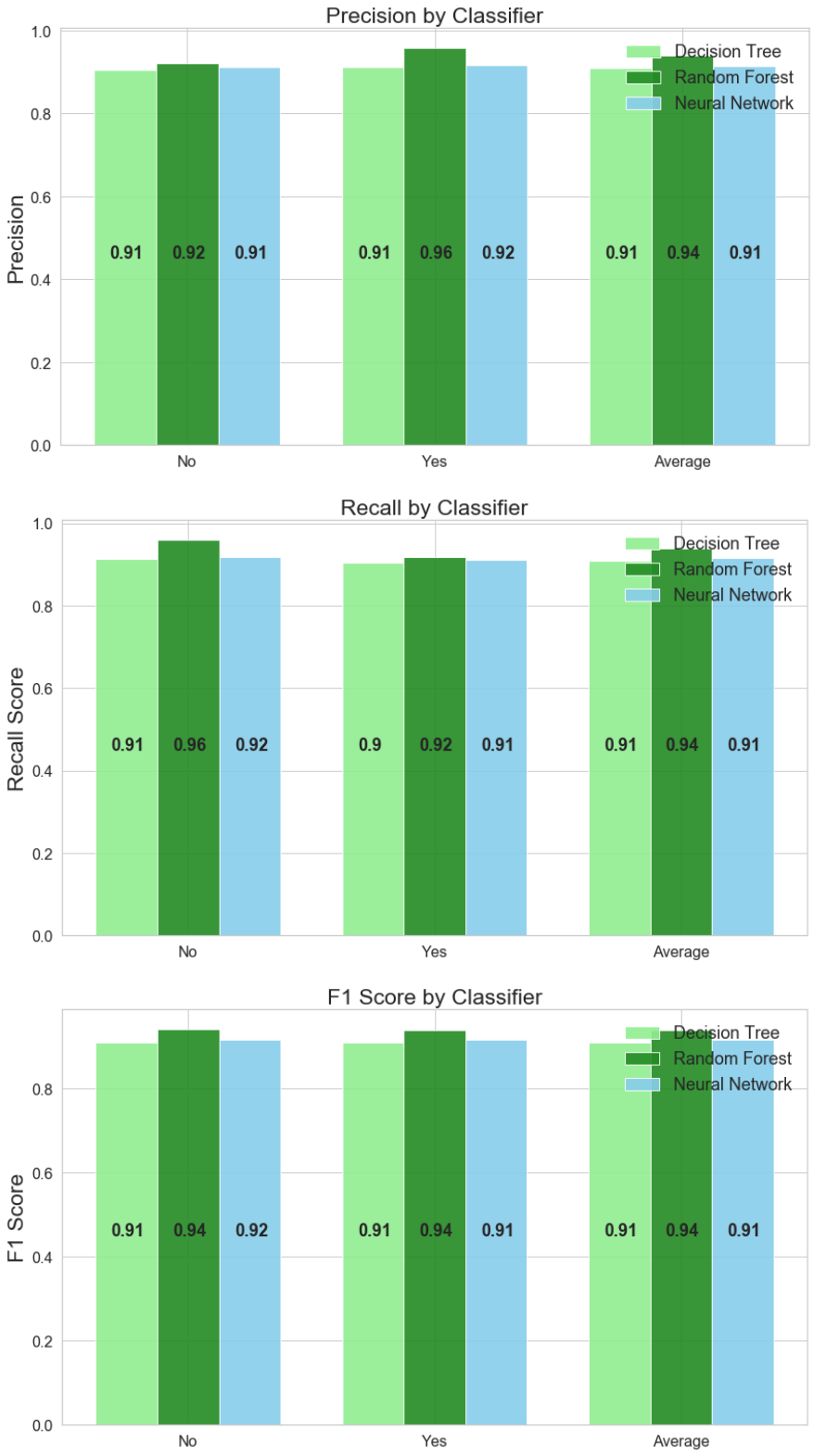


*Figure 10. Precision, Recall and F1-score performance by classifier*

6. *Provide a short explanation of the results you have shown and what it means. Which classification method performed better? Why? Contrast performance with classification from the previous homework and comment on the difference, if any.*

As for the bank wasting a call can be considered trivial compared to losing a prospective client, the intention of the classifier would be to minimize false negatives as well as increase true positives. With this in mind, the plots above show that the Random Forests technique allows for the reduction of false negatives while keeping the highest values of true positives, as well as giving a high value for true negatives and low value for false positives. Due to this, the Random Forests classifier had the best values for recall and precision, and thus also for F1 Score, making it the best option for the correct classification of the bank's potential clients.

To contrast the results of the present assignment with the classification from the previous homework, it has to be taken into account that while this assignment deals with imbalanced data, the previous homework did not and therefore the results are very different. The k-NN model from the previous homework achieved a recall of 0.97 for the “no” class, better than any of the models in the present document, but the recall for the “yes” class was only 0.28. For the SVM the results were 0.99 and 0.2 respectively.

Although it seems like a big difference, it can be proved that is it caused by the imbalanced data. For example, the weighted recall, which takes into account the number of samples of each class and weights their respective recall accordingly, for the homework k-NN was 0.89 , for the homework SVM 0.9 and for this assignment the minimum weighted recall achieved was 0.91 which is not a profound difference after all.

7. *For Fun/Bonus: attempt at least one method to tackle the discrepancy in the size of the classes (imbalanced data).*

The discrepancy in the size of the classes was addressed in question 2, using the SMOTENC method previously described.

## Question 2: Parameter Selection and Classification (for dataset B). Classify dataset B using four classifiers: k-NN, Support Vector Machine (with RBF kernel), Random Forests and simple Neural Networks (MLPs). The objective is to experiment with parameter selection in training classifiers and to compare the performance of these well- known classification methods.

The dataset was inspected and it was noticed that it is stored as a comma separated values (csv) file and each value is separated by a comma (,). The file contains no header with the column names.

The file was taken directly from the file provided and loaded as a pandas data frame with parameters for the header and the delimiter were specified to None and "," respectively. The first 5 rows can be seen in the table below.

1. *Preprocess the given data using the Z-score normalization on the data. Justify the choice of Z-score normalization here, as opposed to min-max normalization. Why do you need normalization in general? Justify why you would normally split the test and training set randomly. What is the distribution of the +1, -1 classes in the dataset?*

The dataset was divided using an 80/20 partition to obtain the train set and test set. The z-score estimator is fitted with the training set and scales both the training and test set.

Z-Score was selected for the normalization of the data due to the possibility of the existence of outliers. Min-Max does not handle outliers correctly, as the existence of an outlier would mark that datapoint as the min or max of the scale and therefore distorting the distribution of the transformation. In other words, as min-max is bounded to [0,1], the existence of an outlier would limit the available space between the data, making the non-outlier datapoints be condensed in the new scale. Min-Max normalization also removes the intrinsic distance relations existing between the different features. On the contrary, Z-Score does not force limits on the scale of the dataset, maintaining the distance relations between the different features.

The random splitting of the data makes sure that the classifier does not bias toward a certain space of the data, and instead generalizes over the whole space available to it. If the data is not randomly split, it would be possible that the classifier learned from a specific part of a dataset (e.g. from the people who are only between 20 and 22 years old found in a voting poll), returning predictions skewed to certain samples and increasing the possibility of underfitting on the data.

The dataset is a little unbalanced with 1137 samples of the +1 class, and 1063 samples of the -1 class (7% less). The parameter ‘stratify’ was used for the split function. This parameter ensures the proportion of values in the sample produced to be the same as the values provided to the parameter ‘stratify’. By using the output column in this parameter, it will make sure that the random split has the same distribution as the original dataset and thus will be better for the training and testing processes.

## 2. *Parameter Selection:*

a) For k-NN you need to evaluate the best value k to use. Using 5-fold cross validation on the training set evaluate k-NN on the values k = [1, 3, 5, 7,…, 31]. The following link can be helpful: https://scikit-learn.org/stable/modules/cross validation.html Plot a figure that shows the relationship between the accuracy and the parameter k. Report the best k in terms of classification accuracy. Explain why you didn't evaluate directly on the test set.

Various classifiers were trained using the cross-validation and the training set with different number of neighbors. The scoring parameter was set to 'balanced_accuracy' to deal with the unbalanced classes. The average accuracy for each number of neighbors used can be seen in Figure 11. The classifier with the best performance was the one with 13 neighbors.

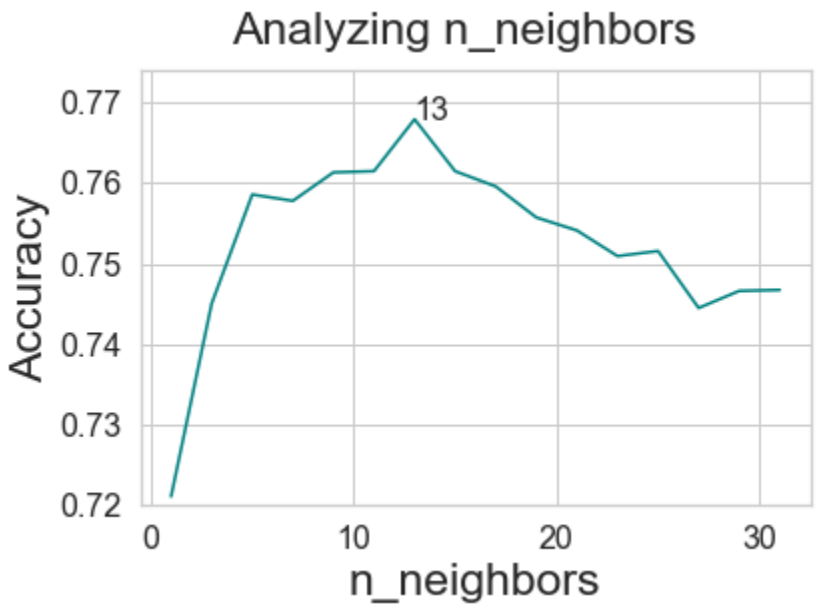


*Figure 11. Relationship between accuracy and number of neighbors*

Selecting the hyperparameters using the test set would increase the risk of overfitting or adding bias since the classifier would be able to recognize the set before the real evaluation happens. The tuning of hyperparameters is part of the classifier building process which should be done using a different set to the testing set because, ideally, the model should be evaluated on unseen samples that were not used to build the classifier.

b) For the RBF kernel SVM, there are two parameters to be decided: the soft margin penalty term "c" and the kernel width parameter "sigma". Again use 5-fold cross validation on the training set to select the parameter "c" from the set [0.1, 0.5, 1, 2, 5, 10, 20, 50] and select the parameter "sigma" from the set [0.01, 0.05, 0.1, 0.5, 1, 2, 5, 10]. Report the best parameters in terms of classification accuracy including plotting the ROC curves [7.5].

The best values for parameters 'c' and 'sigma' were probed using GridSearchCV and were found to be C=10 and sigma=0.01, with a resulting accuracy of 0.89. Figure 12 and Figure 13 show the Receiver Operating Characteristic (ROC) curves and area under the curves (AUC) for the different combinations by fixing either C or sigma.

AUC is different from accuracy and thus explains why, the selected model with C = 10 and sigma = 0.01, is not the one with the highest AUC. AUC is the area under the ROC curve and is related to the probability of a classifier to rank a randomly chosen positive sample higher than a randomly chosen negative sample. It describes the

discriminative ability of the classifier. The accuracy describes the ability of the classifier to classify the samples correctly.

The previously mentioned figures show that the performance of a classifier increases as the AUC increases. It can be seen that the AUC is inversely proportional to the sigma value as AUC decreases when sigma increases. It can also be observed that the AUC is proportional to the C parameter as when one increases the other one does the same.

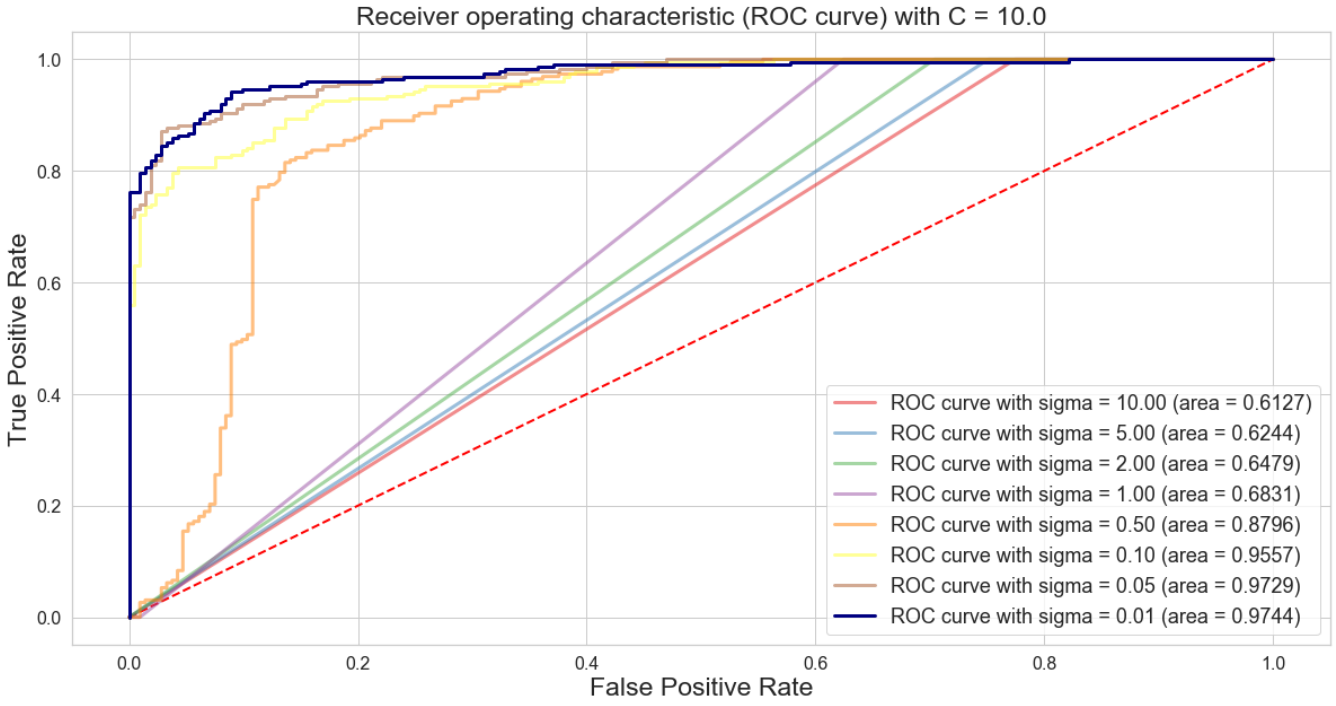


*Figure 12. ROC curves with the C fixed to 10*

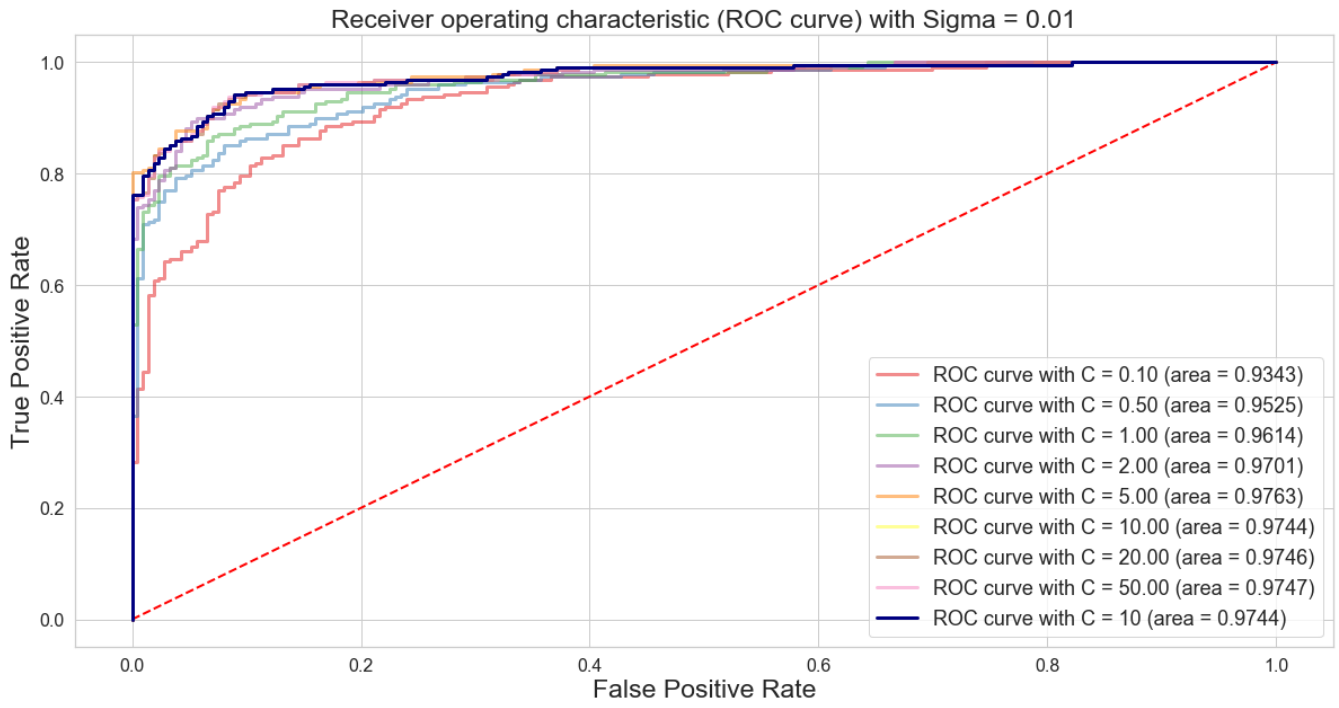


*Figure 13. ROC curves with the Sigma fixed*

## 3. *Train (at least) six classifiers and report the results:*

### a) Classify the test set using k-NN, SVM, Random Forests and Neural Networks. Use the chosen parameters from the parameter selection process in question 2 for k-NN and SVM. For the next two classifiers use the default setups listed at the end for Random Forests and Neural Networks.

The test set is used to evaluate the performance of the k-NN, SVM, random forest and neural network classifiers. K-NN and SVM are used with the parameters found in question 2, and random forest and neural networks are used with the default parameters. The confusion matrix and results can be observed in Table 13.

*Table 13. Confusion matrix of the first four models: k-NN, SVM, Random Forest and Neural Network*

| | k-NN | | SVM | | Random Forest | | Neural Network | | |
|---|---|---|---|---|---|---|---|---|---|
| | -1 | 1 | -1 | 1 | -1 | 1 | -1 | 1 | Total |
| no | 206 | 7 | 196 | 17 | 210 | 3 | 200 | 13 | 213 |
| yes | 86 | 141 | 19 | 208 | 22 | 205 | 20 | 207 | 227 |
| Total | 292 | 148 | 215 | 225 | 232 | 208 | 220 | 220 | 440 |
| | | | | | | | | | |
| Accuracy | 78.9% | | 91.8% | | 94.3% | | 92.5% | | |
| Precision | 0.705 | 0.953 | 0.912 | 0.924 | 0.905 | 0.986 | 0.909 | 0.941 | |
| Recall | 0.967 | 0.621 | 0.920 | 0.916 | 0.986 | 0.903 | 0.939 | 0.912 | |
| f1-score | 0.816 | 0.752 | 0.916 | 0.920 | 0.944 | 0.943 | 0.924 | 0.926 | |

b) For the fifth and sixth classifiers, you should explore the parameters of the Random Forests and Neural Network models to devise your own classifier instance that does better than the other methods. For example, you could consider a deeper neural network with multiple layers, use different optimization/solver algorithms, you could modify the Random Forests using different parameter settings for depth and number of trees or enable boosting. Play around with options and choose a setting for RFs and NNs that performs better.

The parameters for the random forest and neural network classifiers were explored using GridSearchCV using the balanced accuracy as scoring. For the random forest algorithm, the following parameters were tested:

"n_estimators": [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150],

'max_depth': list( range( 2, 40, 1)),

'criterion': ["gini", "entropy"]

And found that the best parameters were

{'criterion': 'entropy', 'max_depth': 10, 'n_estimators': 130} with an accuracy of 0.97

After obtaining this result, a grid further refined from the previous results can be used with the parameters as follows:

"n_estimators": [120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140]

'max_depth': list(range(2,40,1)),

'criterion': ["gini", "entropy"]

The best parameters found were

{'criterion': 'entropy', 'max_depth': 10, 'n_estimators': 136} with an accuracy score of 0.97

For the Neural network classifier, the following parameters were tested:

'hidden_layer_sizes': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120],

'activation': ['identity', 'logistic', 'tanh', 'relu'],

'solver' : ['adam','lbfgs', 'sgd']

And found that the best parameters were

{'activation': 'relu', 'hidden_layer_sizes': 50, 'solver': 'lbfgs'} with an accuracy score of 0.91

After obtaining this result, a grid further refined from the previous results can be used with the parameters as follows:

'hidden_layer_sizes': [41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60]

'activation': ['identity', 'logistic', 'tanh', 'relu'],

'solver': ['adam','lbfgs', 'sgd']

The best parameters found were the same as before.

*Table 14. Confusion matrix of the tuned random forest and neural network*

| | Random Forest | | Neural Network | | |
|---|---|---|---|---|---|
| | no | yes | no | yes | Total |
| no | 209 | 4 | 200 | 13 | 213 |
| yes | 12 | 215 | 17 | 210 | 227 |
| Total | 221 | 219 | 217 | 223 | 440 |
| | | | | | |
| Accuracy | 96.4% | | 93.2% | | |
| Precision | 0.946 | 0.982 | 0.922 | 0.942 | |
| Recall | 0.981 | 0.947 | 0.939 | 0.925 | |
| f1-score | 0.963 | 0.964 | 0.930 | 0.933 | |

It can be seen that by exploring and tuning the parameters, both classifiers had an increase in the accuracy score.

a) Repeat each classification method 20 times by varying the split of the training-test set as in question 2-2. Report the average and standard deviation of classification performance on the test set regarding: accuracy, precision, recall, and F- Measure. Also report the training time and classification time of all the methods. Explain why the classification was repeated 20 times.

Each classification method was repeated using a k = 4, 5, 6 and 7 for cross-validation 20 times. The accuracy, precision, recall and f1-score were recorded and averaged among the same k. The average measure and standard deviation per chosen k in cross-validation for each classifier can be seen in the following figures and tables.

Figure 14 shows the average balanced accuracy performance on the test set after doing cross-validation 20 times for six models. The results are shown per each k of cross-validation. The standard deviation between the 20 cross-validations is shown as a vertical line in each point. Figure 15, Figure 16 and Figure 17 show results for the precision, recall and f1-score in the same manner. It is shown that the standard deviation within a set of results using the same k is not substantially different but that it does change when changing the value of k.

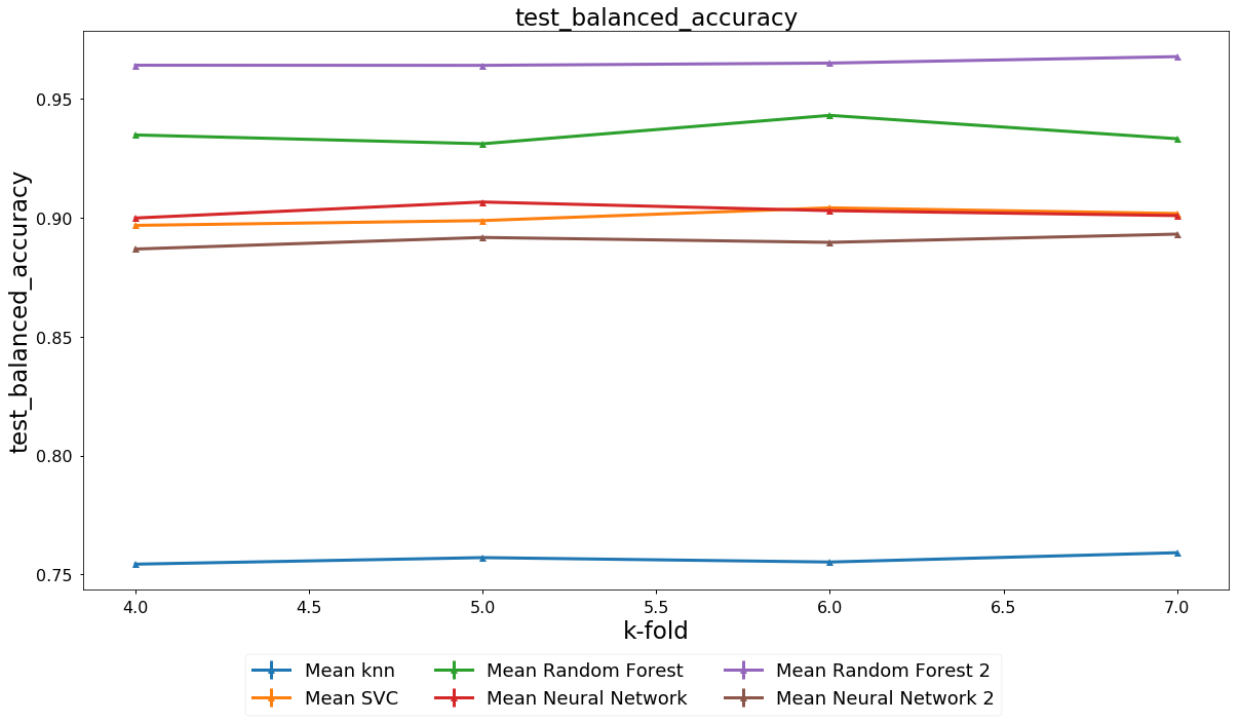


*Figure 14. Error bar of the balanced accuracy average performance on the test set for six models: K-NN, Random Forest, Random Forest Tuned, Support Vector Classifier, Neural Network, Neural Network Tuned per k in cross-validation*

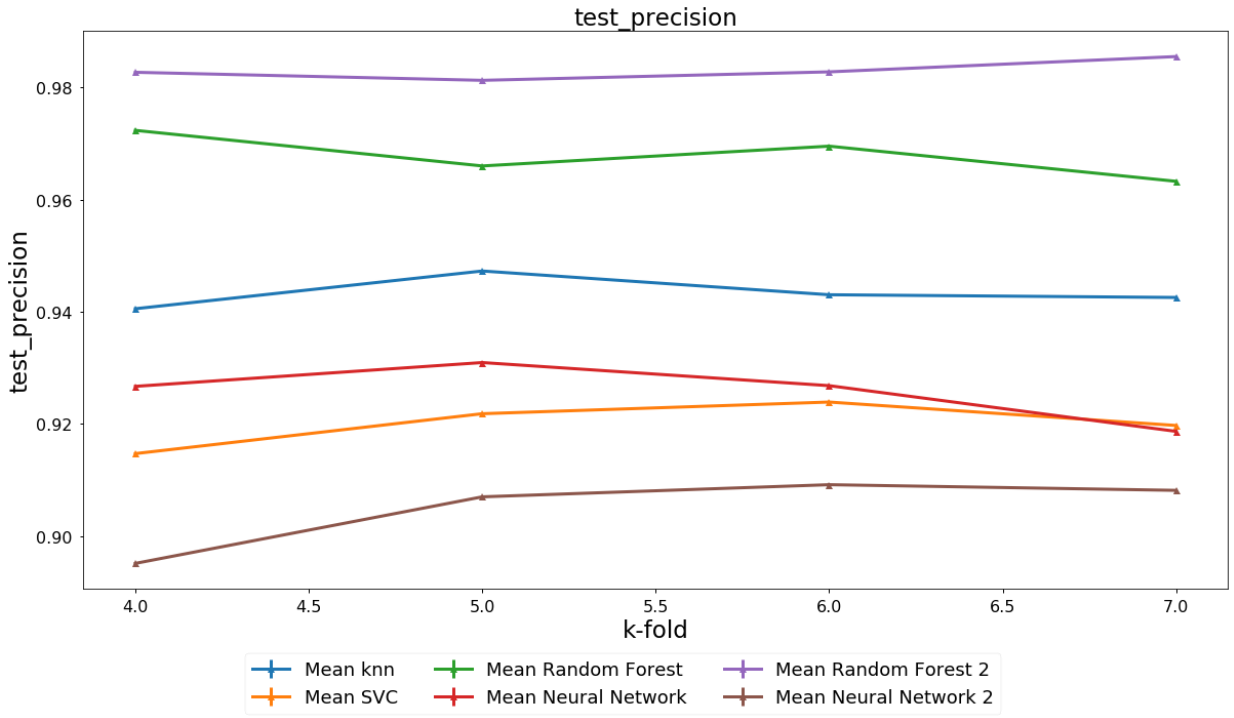


*Figure 15. Error bar of the Precision average performance on the test set for six models: K-NN, Random Forest, Random Forest Tuned, Support Vector Classifier, Neural Network, Neural Network Tuned per k in cross-validation*

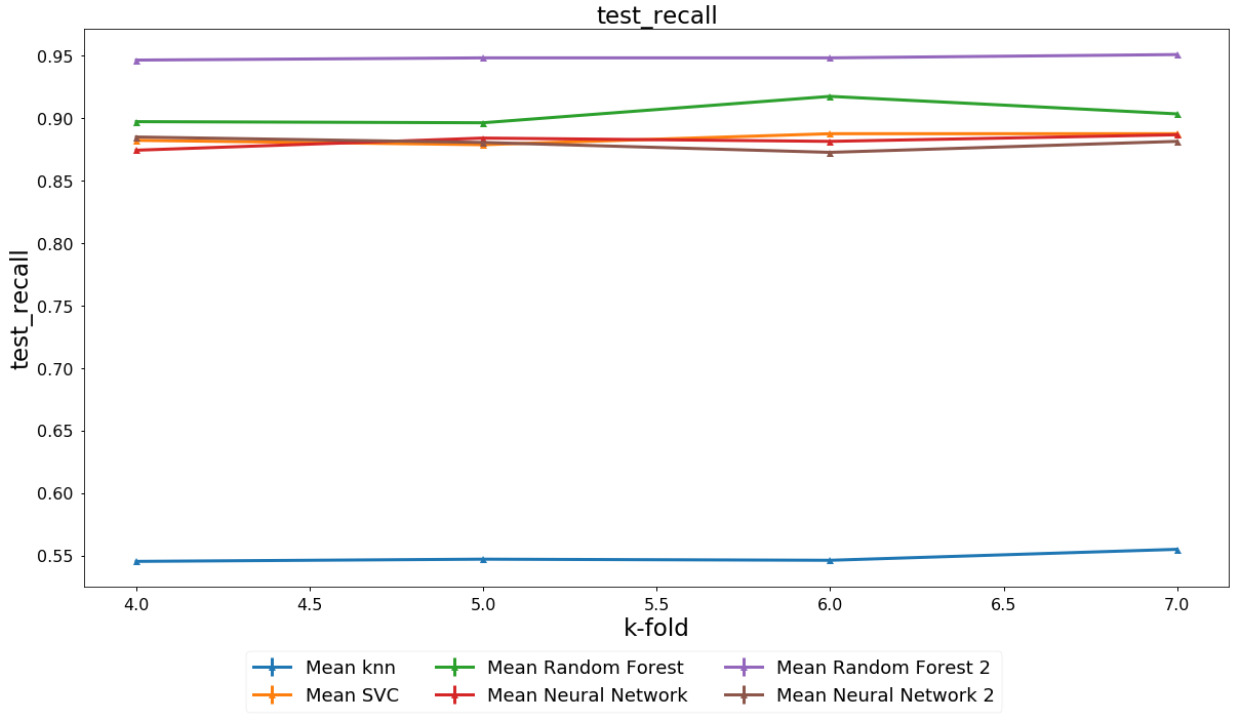


*Figure 16. Error bar of the recall average performance on the test set for six models: K-NN, Random Forest, Random Forest Tuned, Support Vector Classifier, Neural Network, Neural Network Tuned per k in cross-validation*

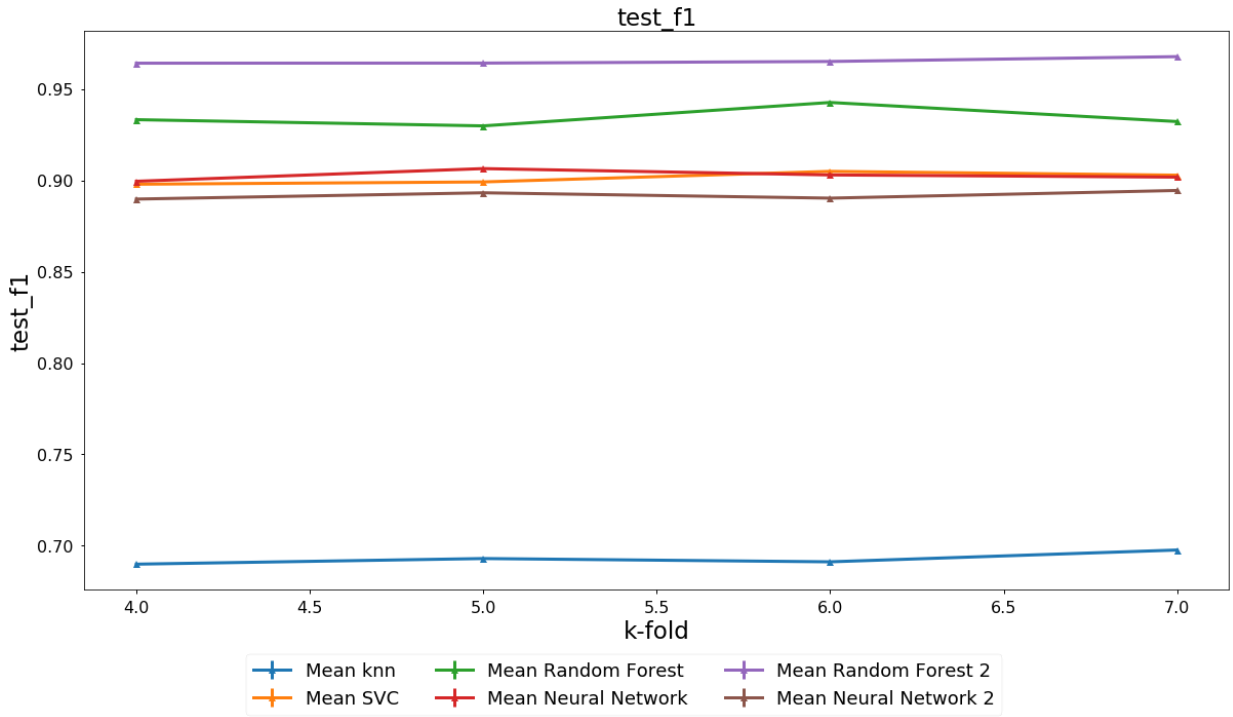


*Figure 17. Error bar of the f1-score performance average performance on the test set for six models: K-NN, Random Forest, Random Forest Tuned, Support Vector Classifier, Neural Network, Neural Network Tuned per k in cross-validation*

Tables 15 to 20 show the numerical results of the balanced accuracy, precision, recall and f1-score performance on the test set after doing cross-validation 20 times for six models. The results are shown per each k of cross-validation. The standard deviation between the 4 different k's is shown in the last row. It can be seen than although the results are different when using a different k, the standard deviation is not significant.

*Table 15. Results of the balanced accuracy, precision, recall and f1-score average performance on the test set for k-NN per k in cross-validation*

| k-Nearest Neighbors | | | | |
|---|---|---|---|---|
| **k** | **Balanced_Accuracy** | **Precision** | **Recall** | **F1-score** |
| **4** | 0.75431 | 0.940565 | 0.545302 | 0.689932 |
| **5** | 0.757066 | 0.947293 | 0.547063 | 0.693006 |
| **6** | 0.75521 | 0.943057 | 0.546176 | 0.691212 |
| **7** | 0.759123 | 0.942573 | 0.554944 | 0.697669 |
| **std** | 0.002132253 | 0.002827904 | 0.00444042 | 0.003386295 |

*Table 16. Results of the balanced accuracy, precision, recall and f1-score average performance on the test set for SVM per k in cross-validation*

| SVM | | | | |
|---|---|---|---|---|
| k | Balanced_Accuracy | Precision | Recall | F1-score |
| **4** | 0.896861 | 0.914723 | 0.882163 | 0.897871 |
| **5** | 0.898877 | 0.921837 | 0.878669 | 0.899129 |
| **6** | 0.904247 | 0.923917 | 0.887469 | 0.904915 |
| **7** | 0.901858 | 0.919739 | 0.887455 | 0.902827 |
| **std** | 0.003253432 | 0.003942108 | 0.004310848 | 0.003256726 |

*Table 17. Results of the balanced accuracy, precision, recall and f1-score average performance on the test set for Random Forest per k in cross-validation*

| Random Forest | | | | |
|---|---|---|---|---|
| **k** | **Balanced_Accuracy** | **Precision** | **Recall** | **F1-score** |
| **4** | 0.934911 | 0.97238 | 0.897109 | 0.933203 |
| **5** | 0.931193 | 0.966033 | 0.896232 | 0.929809 |
| **6** | 0.943159 | 0.969562 | 0.917344 | 0.942616 |
| **7** | 0.933325 | 0.963293 | 0.903328 | 0.932181 |
| **std** | 0.005234553 | 0.00397975 | 0.009752856 | 0.005625116 |

*Table 18. Results of the balanced accuracy, precision, recall and f1-score average performance on the test set for Neural Network per k in cross-validation*

| Neural Network | | | | |
|---|---|---|---|---|
| **k** | **Balanced_Accuracy** | **Precision** | **Recall** | **F1-score** |
| **4** | 0.899963 | 0.926717 | 0.874253 | 0.899487 |
| **5** | 0.906696 | 0.930963 | 0.883955 | 0.906427 |
| **6** | 0.903052 | 0.92684 | 0.881333 | 0.902955 |
| **7** | 0.900978 | 0.918687 | 0.886617 | 0.901754 |
| **std** | 0.002974631 | 0.005137244 | 0.005315116 | 0.00289632 |

*Table 19. Results of the balanced accuracy, precision, recall and f1-score average performance on the test set for Tuned Random Forest per k in cross-validation*

| Tuned Random Forest | | | | |
|---|---|---|---|---|
| **k** | **Balanced_Accuracy** | **Precision** | **Recall** | **F1-score** |
| **4** | 0.964248 | 0.982738 | 0.946368 | 0.964116 |
| **5** | 0.964194 | 0.981304 | 0.948141 | 0.964167 |
| **6** | 0.965142 | 0.982815 | 0.948144 | 0.965056 |
| **7** | 0.967861 | 0.985545 | 0.950785 | 0.967701 |
| **std** | 0.001722268 | 0.00177162 | 0.001820559 | 0.001683596 |

*Table 20. Results of the balanced accuracy, precision, recall and f1-score average performance on the test set for Tuned Neural Network per k in cross-validation*

| Tuned Neural Network | | | | |
|---|---|---|---|---|
| **k** | **Balanced_Accuracy** | **Precision** | **Recall** | **F1-score** |
| **4** | 0.886903 | 0.895138 | 0.884819 | 0.889745 |
| **5** | 0.891784 | 0.907009 | 0.880458 | 0.893183 |
| **6** | 0.889717 | 0.909175 | 0.872533 | 0.890253 |
| **7** | 0.893175 | 0.908164 | 0.881336 | 0.894453 |
| **std** | 0.002727115 | 0.006549061 | 0.00518945 | 0.002274514 |

Figure 18 shows the average training time taken for each classifier to be trained per each k of cross-validation and its standard deviation as vertical lines. It can be observed that, in average, the model that takes longer to train is the Neural Network, either with the default or the selected parameters. This behavior is expected since the Neural Network does a back-propagation process which takes time to converge.

Figure 19 shows the average classification time taken for each classifier to classify the testing set per each k of cross validation and its standard deviation as vertical lines. It can be observed that, in average, the model that takes more time to classify the samples is the k-NN. It was expected since a k-NN classifier has to measure the distance of the sample to all the samples in the trained model and make a comparison in order to make a decision.

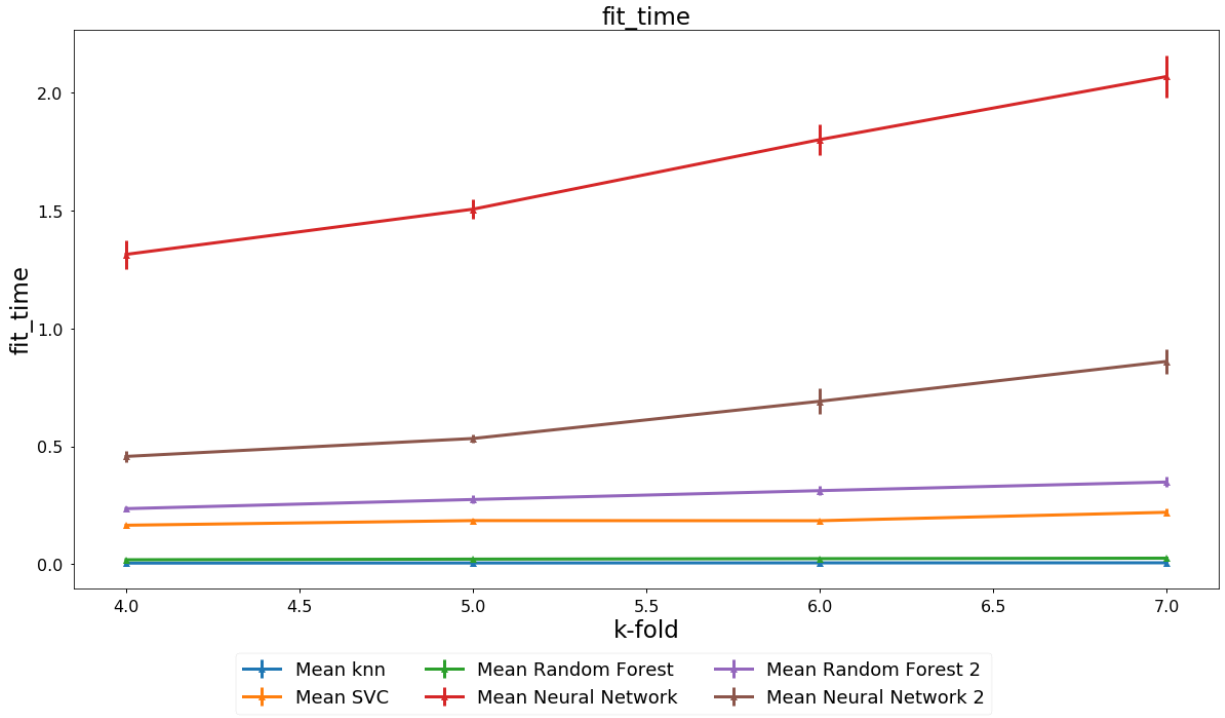


*Figure 18. Error bar of the training time average performance on the test set for six models: K-NN, Random Forest, Random Forest Tuned, Support Vector Classifier, Neural Network, Neural Network Tuned per k in cross-validation*

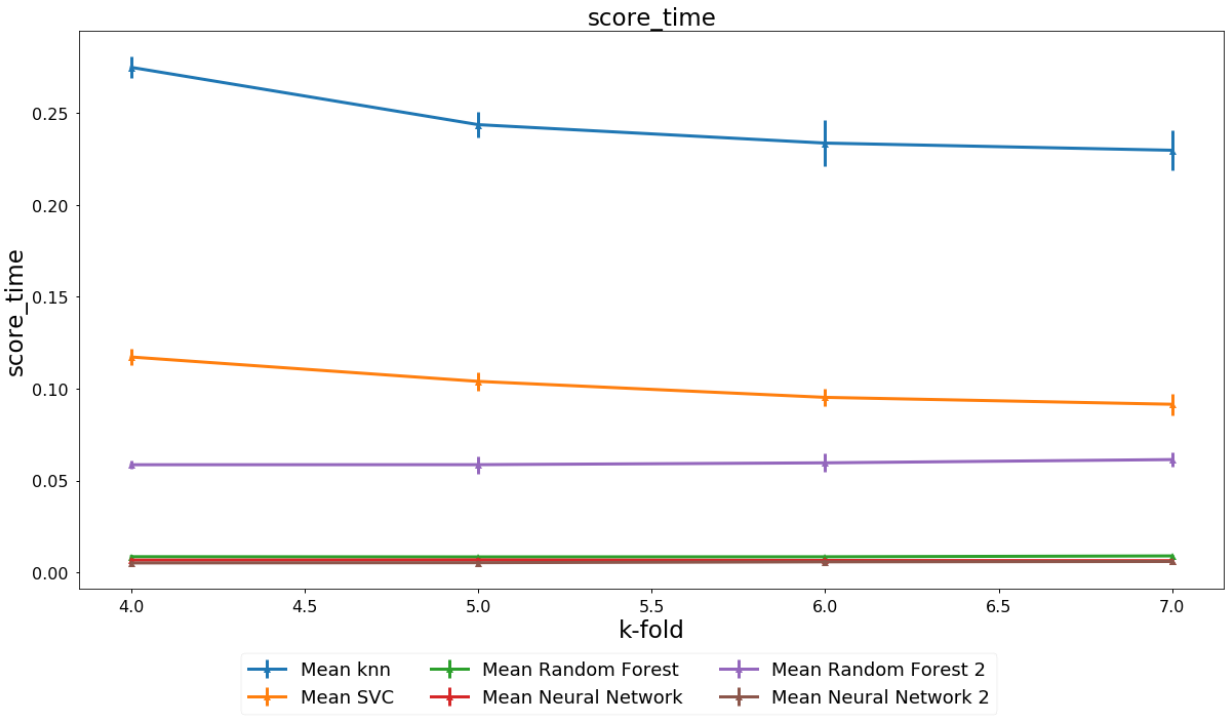


*Figure 19. Error bar of the classification time average performance on the test set for six models: K-NN, Random Forest, Random Forest Tuned, Support Vector Classifier, Neural Network, Neural Network Tuned per k in cross-validation*

The cross-validation process was performed 20 times because classifiers might be unstable, i.e. have different results depending on the data they were trained on. Although cross validation deals with this fact by iterating over the split sets, this sets are not independent of one another. Doing cross-validation 20 times may lower the bias and give a better estimate of the expected error of the classifier.

## *4. Comment on the obtained results, what are the benefits and weaknesses of each method on this dataset. How could this analysis help to make the choice of the right method to use for a dataset of this type in the future?*

The classifiers in order of better accuracy performance were: Tuned random forest, random forest, neural network, SVM, neural network 2 and K-NN. The model with the best results in all performance measurements (i.e. balanced accuracy, precision, recall and f1-score) is the Random Forest with the tuned parameters 'criterion': 'entropy', 'max_depth': 10, 'n_estimators': 136 having an accuracy score of 0.965 ±0.002. The classifier's performance can be seen in Table 21.

*Table 21. Tuned Random Forest Performance*

| Tuned Random Forest | | | | |
|---|---|---|---|---|
| **k** | **Balanced_Accuracy** | **Precision** | **Recall** | **F1-score** |
| **4** | 0.964248 | 0.982738 | 0.946368 | 0.964116 |
| **5** | 0.964194 | 0.981304 | 0.948141 | 0.964167 |
| **6** | 0.965142 | 0.982815 | 0.948144 | 0.965056 |
| **7** | 0.967861 | 0.985545 | 0.950785 | 0.967701 |
| **mean** | 0.965361 | 0.983101 | 0.94836 | 0.96526 |
| **std** | 0.001722 | 0.001772 | 0.001821 | 0.001684 |

Regarding the training time, as it was mentioned before, it can be observed that the model that takes longer to train is the Neural Network and that this behavior is expected since the Neural Network does a back-propagation process which takes time to converge. In contrast, this classifier, together with the default random forest, are the models that take the shortest time to classify samples.

On the subject of the classification time, the model that takes the most time in classifying the samples is the k-NN. This behavior was expected since, in order for a k-NN classifier to make a decision, it has to compute the distance of the test sample to all the samples in the trained model and make a comparison to find the neighbors with the shortest distance.

It is important to take into account different parameters that were not considered before this assignment, as the total training time of the system, a parameter considerably important when dealing with big datasets in which its training could take more time to converge.

The general benefits and weaknesses of the different classification methods are:

KNN

Benefits:

- Has a mathematical explanation that makes it easy to implement and understand.
- It is not very sensitive to outliers.

Weaknesses:

- Computational expensive because it needs to compute the distance of each sample to all the other samples and make a comparison to find the neighbors with the shortest distance.
- Sensitive to imbalanced data. The majority class might dominate the voting. This dataset might have been affected by this fact since it has more samples of class +1.
- Sensitive to the range of features. The feature with the shortest range can shadow the other features since it will affect the calculation of the distance. This dataset might have been affected because the normalization sued was Z-score, which does not leave the data with a fixed range like min-max does.

SVM with RBF Kernel

Benefits:

- By intrinsically mapping the data to a higher space, the features become linearly separable and the decision boundary is created.
- The method is the third fastest to train and to classify.

Weaknesses:

- Prone to overfitting

Random Forest

Benefits:

- Easy to implement.
- The resulting model is easy to understand even to non-technical people.
- One of the fastest to train and to classify this dataset.

Weaknesses:

- Prone to overfitting

Neural Network

Benefits:

- On this dataset, it was the one with the shortest classification time.
- Has the ability to learn and model non-linear data and complex relationships.

Weaknesses:

- On this dataset, it was the classifier that took the longest to train. Might be difficult with larger datasets.
- Difficult to train, might never converge.
- Does not have a mathematical explanation of why it works.

Analyzing the performance and training and classification times can help to make decisions in relation to the right method to use in the future. The right method would have to be chosen based on the application as some processes need a fast classifier such as real time systems like credit card fraud detection, while others might need a more accurate model such as medical diagnosis.

## 5. *If you had to remove 1 feature from the dataset, which feature would you select to remove from the dataset and why? What would have happened if you did classification on two dimensions only?*

The goal of univariate feature selection is to individually examine each feature and determine the strength of its relationship with the outcome variable. There are various techniques to perform a feature selection process:

- Try all possible subsets of features and evaluate their performance. Very good results but not always feasible.
- Backward approach: build a model with all the features and start removing one feature at a time. Keep removing features until removing more features damages the classifier performance.
- Build many decision trees with different subsets of features. Decision trees normally make a node split with the feature that has more information gain. Select the features that consistently are selected to be on the root of the tree.
- Evaluate each feature according to how good it is for discriminating the class and eliminate the ones with the worst performance [22].

For this assignment, the last approach was used. The features were ranked to identify the feature with the lowest score using the python function SelectKBest which evaluates and returns a score depending on the selected score

function. The selected score function was the Chi-square testing because the target is categorical. In this case, feature '53' was the feature with the lowest score.

This method can also be used to reduce the dataset to have only two features by selecting the two highest scoring features. Table 22 shows the results of training a random forest with the two most meaningful features. These results are fairly high for having such a small feature information. Nonetheless, it comes with no surprise that this classifier has significantly worse results when dealing with unseen data than the classifier that was trained with all the features.

*Table 22. Results of a Random Forest classifier using only two features.*

| Random Forest with 2 features | | | |
|---|---|---|---|
| | **no** | **yes** | **Total** |
| **no** | 206 | 7 | 213 |
| **yes** | 86 | 141 | 227 |
| **Total** | 292 | 148 | 440 |
| **Accuracy** | 78.9% | | |
| **Precision** | 0.705 | 0.953 | |
| **Recall** | 0.967 | 0.621 | |
| **f1-score** | 0.816 | 0.752 | |

# References


[1] University of California, Irvine, "UCI Machine Learning Repository: Bank Marketing Data Set," [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Bank+Marketing. [Accessed 26 Feb 2019].

[2] Pandas Project, "Tutorials - pandas 0.23.4 documentation," Documentation on the use of the pandas library, [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/tutorials.html. [Accessed 19 January 2019].

[3] The Scipy Community, "Statistical functions (scipy.stats) - SciPy v0.14.0 Reference Guide," Documentation on the use of the SciPy library, 11 May 2014. [Online]. Available: https://docs.scipy.org/doc/scipy-0.14.0/reference/stats.html. [Accessed 8 January 2019].

[4] The SciPy Community, "Statistics - NumPy v.1.14.1 Manual," Documentation on the use of the NumPy library, 16 April 2018. [Online]. Available: https://docs.scipy.org/doc/numpy-1.14.1/reference/routines.statistics.html. [Accessed 8 January 2019].

[5] The Matplotlib Development Team, "matplotlib.pyplot.hist - Matplotlib 3.0.2 documentation," Documentation on the use of the Matplotlib library, 11 November 2018. [Online]. Available: https://matplotlib.org/api/_as_gen/matplotlib.pyplot.hist.html. [Accessed 8 January 2019].

[6] The Matplotlib Development Team, "Histograms - Matplotlib 3.0.2 documentation," Documentation on the use of the Matplotlib library, 11 November 2018. [Online]. Available: https://matplotlib.org/gallery/statistics/hist.html. [Accessed 9 January 2019].

[7] B. Solomon, "Python Plotting with Matplotlib (Guide)," Real Python, 28 February 2018. [Online]. Available: https://realpython.com/python-matplotlib-guide/. [Accessed 15 January 2018].

[8] S. Raschka, "About Feature Scaling and normalization," Sebastian Raschka, 11 July 2014. [Online]. Available: http://sebastianraschka.com/Articles/2014_about_feature_scaling.html#about-standardization. [Accessed 15 January 2019].

[9] M. Waskom, "seaborn: statistical data visualization - seaborn 0.9.0 documentation," Documentation on th euse of the seaborn library, [Online]. Available: https://seaborn.pydata.org/. [Accessed 22 January 2019].

[10] Pandas Project, "Working with missing data - pandas 0.24.1 documentation," Documentation on the use of the Pandas library, [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html#interpolation. [Accessed 15 January 2019].

[11] Tallahassee Community College Learning Commons, "Z-scores-and-the-Empirical-Rule," Instructional aid on Z scores and the empirical rule, 24 May 2016. [Online]. Available: https://www.tcc.fl.edu/media/divisions/learning-commons/top-5-resources/math/statistics/Z-scores-and-the-Empirical-Rule.pdf. [Accessed 15 January 2019].

[12] A. Koufakou, E. Ortiz, M. Georgiopoulos, G. Anagnostopoulos and K. Reynolds, "A Scalable and Efficient Outlier Detection Strategy for Categorical Data," in *19th IEEE International Conference on Tools with Artificial Intelligence(ICTAI 2007)*, Patras, Greece, 2007.

[13] G. Lemaitre, F. Nogueira and C. Oliveira, "2. Over-sampling - Imbalancerd-learn 0.4.3 documentation," Over-sampling Imbalanced-learn user guide, [Online]. Available: https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html#smote-adasyn. [Accessed 14 March 2019].

[14] scikit-learn Developers, "1.10 Decision Trees -- scikit-learn 0.20.3 documentation," 2018. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart. [Accessed 14 March 2019].

[15] X. Wu and V. Kumar, The Top Ten Algorithms in Data Mining, 1st ed., New York, New York: Taylor & Francis Group, 2009, p. 208.

[16] scikit-learn Developers, "3.2.4.3.1. sklearn.ensemble.RandomForestClassifier -- scikit-learn 0.20.3 documentation," 2018. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed 14 Month 2019].

[17] SciKit-learn developers, "3.2.4.3.1. sklearn.ensemble.RandomForestClassifier - scikit-learn 0.20.3 documentation," Scikit-learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed 14 March 2019].

[18] scikit-learn Developers, "1.17. Neural network models (supervised) -- scikit-learn 0.20.3 documentation," 2018. [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html#classification. [Accessed 15 March 2019].

[19] M. Nielsen, "Neural Networks and Deep Learning," Determination Press, October 2018. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap3.html#introducing_the_cross-entropy_cost_function. [Accessed 15 March 2019].

[20] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, C. Suen, A. Coates, A. Maas, Hannun Awni, B. Huval, T. Wang and S. Tandon, "Unsupervised Feature Learning and Deep Learning Tutorial: Softmax Regression," University of Stanford, 25 September 2013. [Online]. Available: http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression/. [Accessed 15 March 2019].

[21] Stack Overflow, "Estimating the number of neurons and number of layers of an artificial neural network [closed]," Stack Exchange, 27 July 2015. [Online]. Available: https://stackoverflow.com/a/3345770. [Accessed 15 March 2019].

[22] I. Guyon and A. Eliseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research,* vol. 3, pp. 1157-1182, 2003.