***Project Report***

*Group A*

*Keras & Bokeh*

*Submitted By*

*Sonnet Padayatty Reju*

*Student ID: 18200682*

*MSc. Data & Computational Science*

**Chapter 1**

**Introduction**

* 1. **Project Description**

We are analyzing a dataset with following packages

* Keras : For Deep Learning
* Bokeh : For Visualization

Our aim is to predict the response of customers as explained below. As the desired output is either Yes or No, We need to adapt classify machine learning methods for the analysis part.

**1.2 Problem Statement**

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be subscribed ('yes') or not ('no').

**Data fields**

**Input variables:-**

1. **CustAge** (numeric)
2. **Profession** : types (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3. **Marital** : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4. **Education** (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5. **Default**: has credit in default? (categorical: 'no','yes','unknown')
6. **Housing**: has housing loan? (categorical: 'no','yes','unknown')
7. **Loan**: has personal loan? (categorical: 'no','yes','unknown')
8. **Contact**: contact communication type (categorical: 'cellular','telephone')
9. **Month**: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10. **Day\_of\_week**: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11. **Duration**: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no').
12. **Campaign**: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13. **Pdays**: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14. **Previous**: number of contacts performed before this campaign and for this client (numeric)
15. **Poutcome**: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
16. **Emp.var.rate**: employment variation rate - quarterly indicator (numeric)
17. **Cons.price.idx**: consumer price index - monthly indicator (numeric)
18. **Cons.conf.idx**: consumer confidence index - monthly indicator (numeric)
19. **Euribor3m**: euribor 3 month rate - daily indicator (numeric)
20. **Nr.employed**: number of employees - quarterly indicator (numeric)
21. **Past emails** – count previous mails (numeric)

**Target Variable**

1. **Response** - Has the client subscribed a term deposit? (binary: 'yes' or 'no')

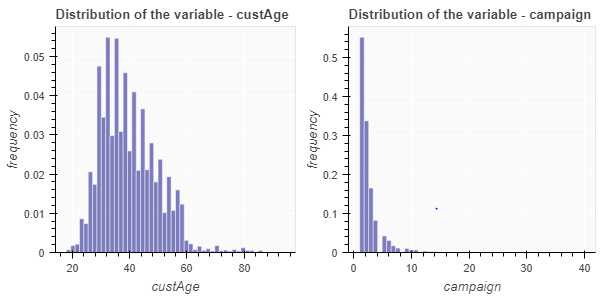
**Chapter 2**

**Methodology**

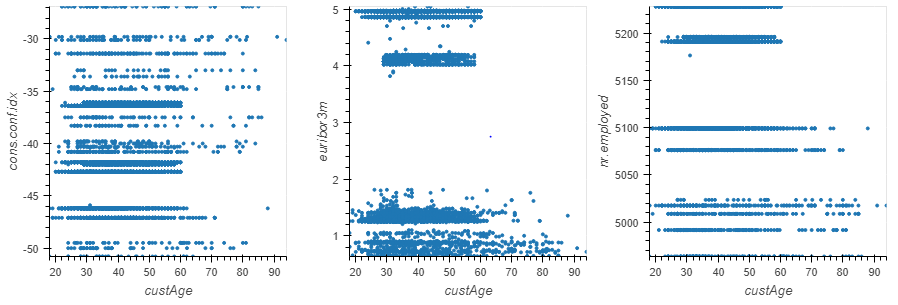
**2.1 Pre-Processing**

Pre-processing of data is an indispensable stage in predictive analysis. Since the predictive model needs to handle a big data set, it is always necessary to eliminate unwanted data. There may be many variables whose data type is incorrect and may create complexity while training the predictive model with train dataset. In order to minimize such issues in modeling stage, we conduct data pre-processing and extract important insights from the raw data. We can extract such information by analyzing the independent variables using scatter plot or by visualizing how the data points have been distributed in each variable. It can be easily achieved by normality checking functions like histograms.

Histograms:-



Scatter plots:-



Following are main pre-processing methods used in predictive analysis

**2.1.1 Missing Value Analysis**

Once we are done with pre-processing steps like “Renaming the variables” and “Converting into proper Data types”, we can conduct the missing value analysis. You may use the combination of both train and test data for the imputation of missing values as it makes the model to predict the values more accurately. Mainly, there are 3 methods for imputation of missing values.

1. Mean method

2. Median method

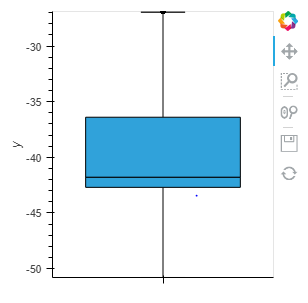
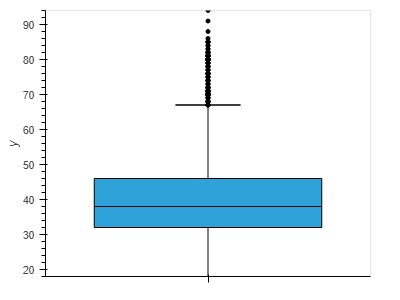
3. KNN imputation

We are not supposed to do imputation if the percentage of missing values in a variable is more than 30%.

Since none of the variables exceed 30%, we can proceed with imputation. From the analysis, it is found that KNN imputation is the effective technique for this dataset.

**2.1.2 Outlier Analysis**

By definition, outliers are points that are distant from remaining observations. As a result, they can potentially skew or bias any analysis performed on the dataset. It is therefore very important to detect and adequately deal with outliers. In order to show the impact of outliers, we use a technique called box plot in which the distribution of data points is visualized as shown below.



The outliers that we found in boxplot cannot always be considered as irrelevant data. The outliers could be relevant if we found that the prediction power is dependent on them. In such cases, outliers cannot be considered as data entry errors, so it is important that we need to keep those data in model creation. In our dataset, we found that the removal of outliers doesn’t make much difference in Accuracy of model.

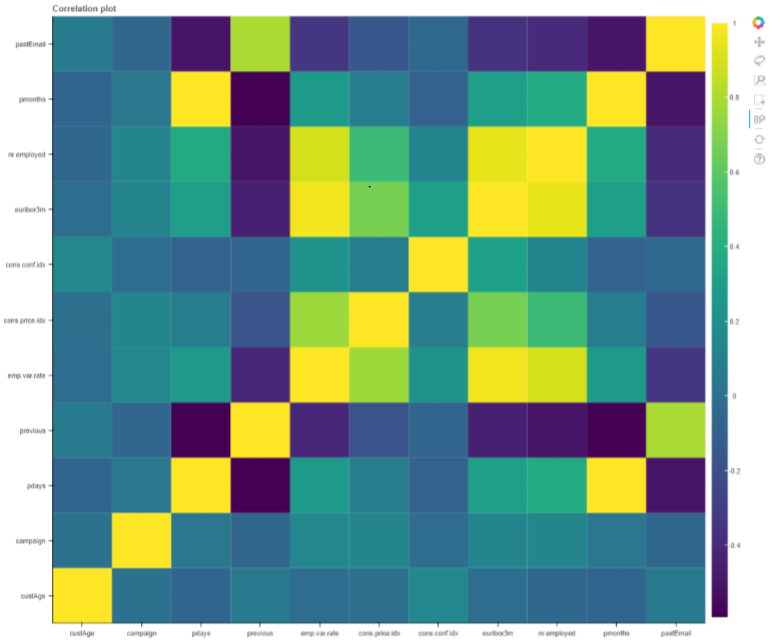
If we are in position that the removal of outliers is mandatory, then we can replace those values by NaN and do the step - KNN imputation again. The distribution of data became more normal / less skewed after the removal of outliers. This illustrates that the removal of outliers from the data may improve the accuracy of the predictive model, but not always.

**2.1.3 Feature Selection**

Feature selection is extremely important in machine learning primarily because it serves as a fundamental technique to direct the use of variables to what's most efficient and effective for a given machine learning system. It helps to minimize the curse of dimensionality or help deal with over fitting feature selection helps to give developers the tools to use only the most relevant and useful data in machine learning training sets, which dramatically reduces costs and data volume. There are many ways to do feature selection, but in this project we use Chi-Square test and Correlation Analysis for the feature selection of Categorical and Continuous variables respectively.

The correlation plot of numerical variables is shown below. From the figure, it is clear that the following variables are highly correlated to each other.

1. Pmonths & pdays
2. Emp.var.rate & euribor3m
3. Emp.var.rate & nr.employed



**Correlation plot:**-

When it comes to categorical variables, we need to perform Chi-Square test of Independence in order to extract unwanted variables from the data set. The result of Chi-Square test is shown below.

profession

1.6242589537712605e-34

marital

4.693435664866768e-05

schooling

4.071800591948163e-08

default

1.626010224096433e-15

housing

0.5469213692385477

loan

0.09547438986454948

contact

4.416434113838791e-36

month

3.3492292613263237e-139

day\_of\_week

0.038266146752773816

poutcome

3.898575747043989e-181

Since the p values of the variables- “day\_of\_week” "housing" and "loan" are greater than 0.05, As per the rule of Chi-Square test, it is found that these variables can be eliminated from modeling stage.

Therefore, as part of Dimension reduction, the following variables are removed from the data set.

Variables: - 'pdays', 'emp.var.rate', 'day\_of\_week', 'loan', 'housing'

**2.1.4 Feature Scaling**

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization. For example, the majority of classifiers calculate the distance between two points by the Euclidean distance. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. Another reason why feature scaling is applied is that gradient descent converges much faster with feature scaling than without it.

In this project, we apply feature scaling method- Normalization.

**Chapter 3**

**Modeling**

For the model creation, we use the package – **Keras**.

Keras is a powerful easy to use python library for developing and evaluating deep learning models. It has the efficient numerical libraries Theano and TensorFlow and allows us to define and train neural networks in a few short lines of code.

There are 2 types of model available in keras

1. Sequential model
2. Model class used with functional API

The model we selected in our case is Sequential and the model creation comprises of following steps.

1. **Loading the data**

We are already done with this step as mentioned above.

1. **Splitting the dataset**

The dataset is spitted into 16 input variables (X) and 1 output variable (Y).

1. **Model creation**

Creation of model can be done by sequence of layers after importing sequential and dense packages. We create sequential model and add layers one at a time till we are satisfied with our network topology. The efficient topology can be found by trial and error method. Generally, we need a network large enough to capture the structure of the problem.

Here, we use fully connected network structure with 4 layers as follows

model = Sequential()

model.add(Dense(40, input\_dim=16, activation='relu'))

model.add(Dense(50, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

Fully connected layers are defined using dense class. We can specify number of neurons in the layer as first argument, input dimension as second argument, and activation function as third argument.

In this case, we initialize network weights to a small random number generated from a uniform distribution, i.e 0 and 0.05 as it is default uniform weight initialization in Keras.

We will use relu activation function on first 2 layers and sigmoid function in the output layer. We use a sigmoid function on output layer to ensure the output is between 0 and 1. The first layer has 18 neurons expecting 16 inputs , the second layer has 15 neurons and the output layer has 1 neuron to predict the class.

1. **Compile the Model**

Compiling the model uses efficient numerical libraries from backend like theano or tensorflow. The backend automatically chooses the best way to represent the network for training and making predictions to run on hardware, such as GPU or CPU. Training a network means finding the best set of weights to make predictions. We must specify the loss function to use to evaluate set of weights, the optimizer is used to search through different set of weights for the network and any optional metrics is used to collect the information and report during the training. Here the loss function is binary cross entropy and metric used is accuracy, as it is a classification problem.

1. **Fit the model**

We can fit or train our model on the loaded data by calling the fit function on the model. The training process will run for a fixed number of iterations through the dataset called epochs. We set number of instances that are evaluated before weight update by mentioning the batch size.

1. **Evaluate the model**

This will give a rough idea about the performance of our model. Even though it gives the accuracy of the model, it has no idea how well it would perform for a new dataset. In order to evaluate the performance on new data, we have already split our dataset into train and test. The model evaluation can be done by using evaluate () function by mentioning the required arguments X and Y.

1. **Predict the output of new data**

This can be easily achieved by using the model.predict () function with required argument.

**Chapter 4**

**Conclusion**

The Accuracy of the trained model is **96.55%**

Loss = **0.1%**

The Summary of the model is shown below:-

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_4 (Dense) (None, 40) 680

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_5 (Dense) (None, 50) 2050

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_6 (Dense) (None, 1) 51

=================================================================

Total params: 2,781

Trainable params: 2,781

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The accuracy obtained after running the test dataset is **88%.**

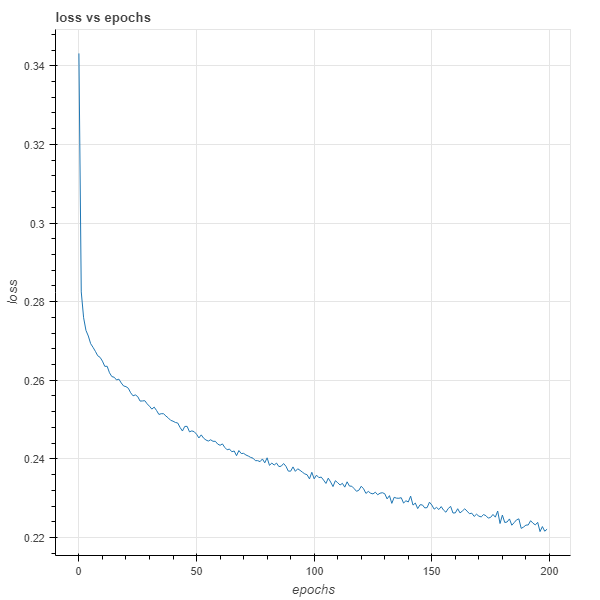
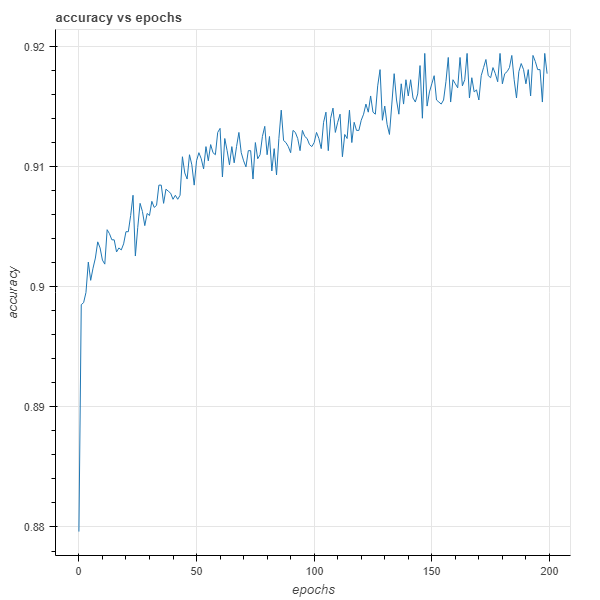
By using the confusion matrix, we found the False Negative Rate = **62.5%**

**Confusion matrix** obtained is shown below.

array([[1237, 86],

[ 100, 60]], dtype=int64)

**The line graph of Epoch vs Loss & Epoch vs Accuracy**

**Via Pytorch**

The same problem is solved by using Pytorch:-

The Accuracy of the trained model is **93.4%**

The results of the same are show below

Loss = **0.19**

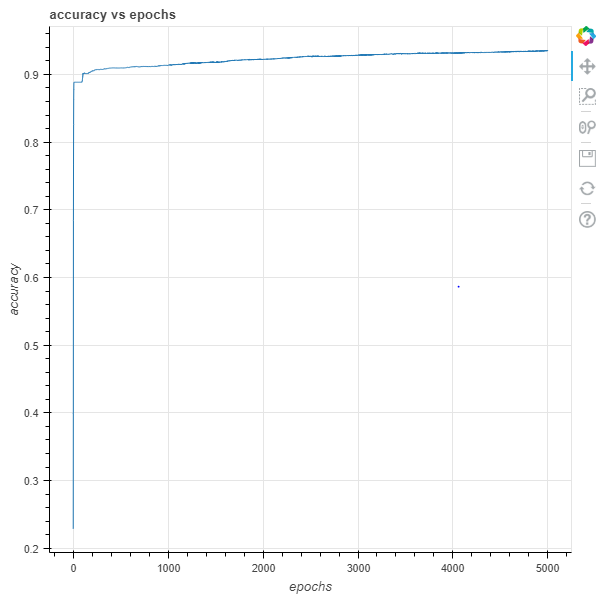
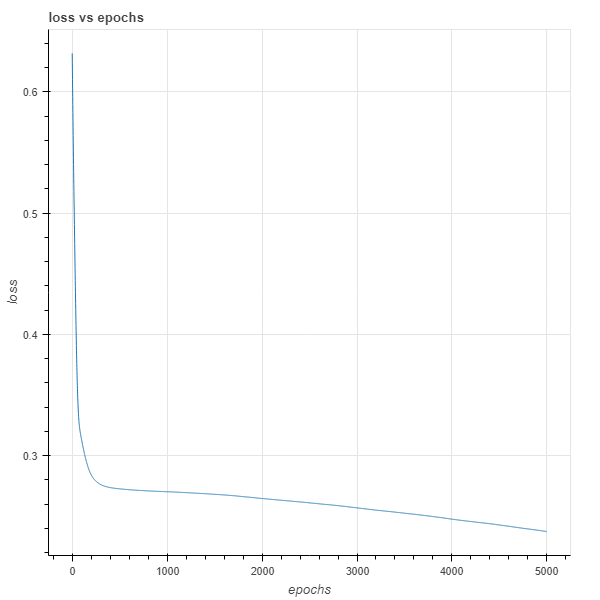
Accuracy obtained after running the test dataset is **88%**

False Negative Rate is **76%**

**Confusion metrics:**

array([[1895, 69],

[ 199, 62]], dtype=int64)



**Comparison Pytorch vs Keras**

|  |  |
| --- | --- |
| **Keras** | **Pytorch** |
| Accuracy of model (in train dataset)= 96% | Accuracy of model (in train dataset)= 93.4% |
| Obtained acceptable accuracy within 500 epochs | Obtained acceptable accuracy within 5000 epochs |
| Loss = 0.1 | Loss = 0.19 |
| FNR = 62.5% | FNR = 76% |
| Acceptable speed when training the model | Better speed when training the model |
| Static Plotting | Dynamic Plotting is possible |

False negative rate is one of the vital factors that take into account while comparing different models. Accuracy doesn’t show the whole picture as higher FNR may have significant negative effects on the business aspects of a company. So when we take False negative rate into account, it is evident that the model created via Keras package has outclassed the one created through Pytorch.