8th May 2022

Predicting the Success of Bank Telemarketing

Keerthi Vakkalaggada

Student Number: 18222774

Course:Artificial Intelligence

University of Missouri-St. Louis

Contents

1 Abstract

How will the financial institution's future marketing strategies be more effective? To answer this, we must review the bank's most recent marketing strategy and find trends that will assist us in drawing conclusions and implementing future strategies. The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y) using Neural networks. In the process of achieving the goal, I have used Portuguese bank-marketing dataset to develop a strong Neural Network model that is less complex in architecture wise and can be able to produce efficient accuracy. We can use this model to predict which people the bank should market to, for their marketing campaign to get people to sign up for a term deposit.

2 Introduction

A common strategy for growing a company is to run marketing sale campaigns. The focus of marketing strategies is on the needs of consumers and their overall satisfaction. Nonetheless, a number of factors affect whether or not a marketing strategy will be successful. When planning a marketing strategy, there are a few things to consider. "Direct marketing" one of the strategies used by companies to contact particular groups of consumers in order to accomplish a specific objective. Customer remote communications may be centralized in a contact center, making campaign management simpler. Customers can connect with these centers through a variety of networks, including telephone (fixed-line or mobile).

What is a Term Deposit? A Term deposit is a deposit that a bank or a financial institution offers with a fixed rate (often better than just opening deposit account) in which your money will be returned back at a specific maturity time.

3 About Dataset

The dataset is related to a Portuguese banking institution's direct marketing campaigns. Phone calls were used in the marketing campaigns. More than one interaction with the same client was frequently needed in order to decide if the product (bank term deposit) will be subscribed ('yes') or not ('no'). The classification aim is to predict whether a client will sign up for a term deposit (variable) by answering yes or no.
The "Bank Marketing datset" was obtained from the UCI Machine learning repository [1].The considered dataset is unbalanced, as only 4640(11.28 percent) records are related with successes and remaining 36548 are of failures in which total dataset comprises of 41,188 data samples. The out class labels used are 0 (no) and 1 (yes).

Following fields are the input features and their description:

1 - age

2 - job : type of job

3 - marital : marital status

4 - education

5 - default: has credit in default?

6 - housing: has housing loan?

7 - loan: has personal loan?

related with the last contact of the current campaign:

8 - contact: contact communication type

9 - month: last contact month of year

10 - day-of-week: last contact day of the week

11 - duration: last contact duration, in seconds

other attributes:

12 - campaign: number of contacts performed during this campaign and for this client

13 - pdays: number of days that passed by after the client was last contacted from a previous campaign

14 - previous: number of contacts performed before this campaign and for this client

15 - poutcome: outcome of the previous marketing campaign

social and economic context attributes

16 - emp.var.rate: employment variation rate - quarterly

indicator 17 - cons.price.idx: consumer price index - monthly indicator

18 - cons.conf.idx: consumer confidence index - monthly indicator

19 - euribor3m: euribor 3 month rate - daily indicator

20 - nr.employed: number of employees - quarterly indicator

21 - y: Output variable - has the client subscribed a term deposit? (yes, no)

	age	job	marital	education	...	cons.conf.idx	euribor3m	nr.employed	y
0	56	1	1	1	...	-36.4	4.857	5191.0	0
1	57	2	1	4	...	-36.4	4.857	5191.0	0
2	37	2	1	4	...	-36.4	4.857	5191.0	0
3	40	3	1	2	...	-36.4	4.857	5191.0	0
4	56	2	1	4	...	-36.4	4.857	5191.0	0

Figure 1: Input features Statistics

4 Data Pre-processing

4.1 Data cleaning

The process of correcting or deleting inaccurate, corrupted, improperly formatted, duplicate, or incomplete data from a dataset is known as data cleaning. Although data cleaning strategies vary depending on the types of data use, However, we can use these basic steps to clean your dataset.
1.Remove duplicate or irrelevant observations.
2.Fix structural errors.
3.Filter unwanted outliers.
4.Handle missing data.

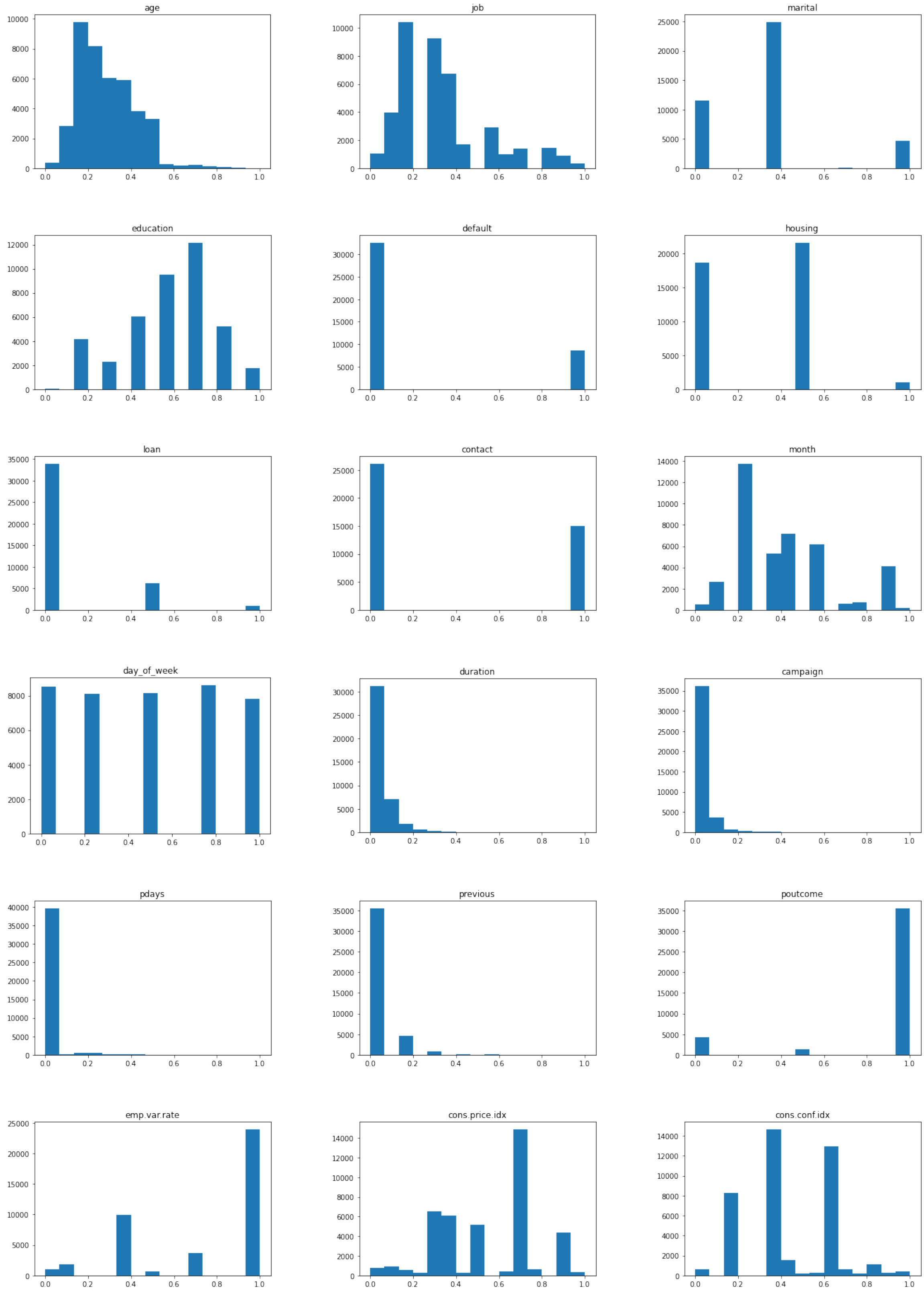
4.2 Data Normalization

Next step will be Data Normalization and it plays a crucial role. As we can see, the data was not distributed uniformly. So, we have to pre-process the data applying normalization techniques. Normalization makes the optimization problem more numerically stable and makes training less sensitive to the scale of features. When we normalize the data , all the values are scaled between "0 and 1", and the outliers will be eliminated from datsset, However they remain visible within our normalized data.
There are several normalization techniques like Rescaling, Mean Normalization, Standardization, Min-Max Normalization that can be utilized and each of them has its own consequences,but for now, any of them is sufficient. The technique that I used is Min-Max Normalization Formula which can be calculated using:

$$\mathbf{Xnew = (X - Xmin)/(Xmax - Xmin)} \tag{1}$$

4.3 Data Visualization

The following plots depicts the distribution of normalized data:



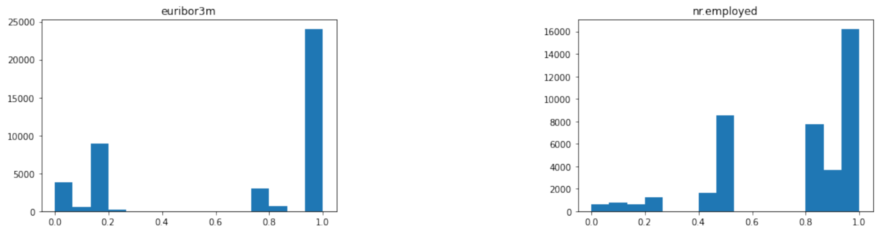
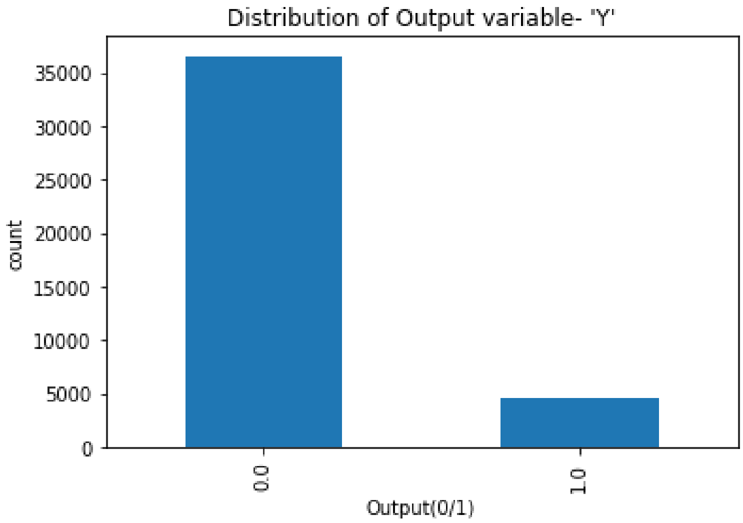


Figure 8: Histogram plots of Normalized data

4.4 Distribution of Output variable

The output variable ’Y’ tells us whether the client subscribed a term deposit? (binary: ’yes’,’no’). The output class labels used are 0 (for ’no’) and 1 (for ’yes’).
Output value ’1’ represents the client has subscribed to term deposit and ’0’ represents the client hasn’t subscribed to term deposit.



5 Data and Network Modelling

5.1 Data modelling

I have divided my whole dataset into Training set and validation set, in which Training set consists of 70 percent of the data and validation set has remaining 30 percent of the data. Data is shuffled whenever data split is performed to avoid biased/unfair prediction. Thus the result may vary every time. All models are compiled and fit on April 23, 2021.
Before going to feed the data to model it is important to Normalize or standardize the input dataset in order to obtain an efficient classification accuracy.
Min-Max normalization is the technique that I used for normalizing my dataset. I have also applied standardization which involves using mean and standard-deviation parameters to normalize the data. However mean for my dataset is ’0.0’ and standard deviation seemed to be ’1.0’. From my observation data before and after applying normalization remained same. That

is why I chose Min-max normalization technique.

5.2 Baseline accuracy

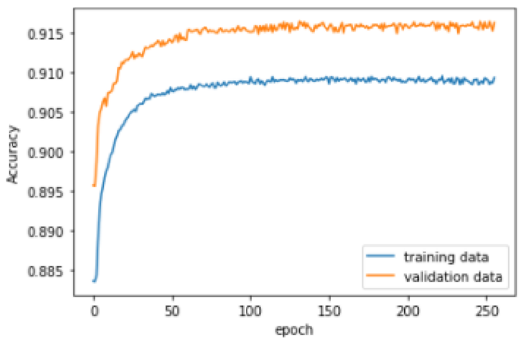
Baseline accuracy for my dataset is 89.1 percent. This clearly tells my dataset is hugely imbalanced.

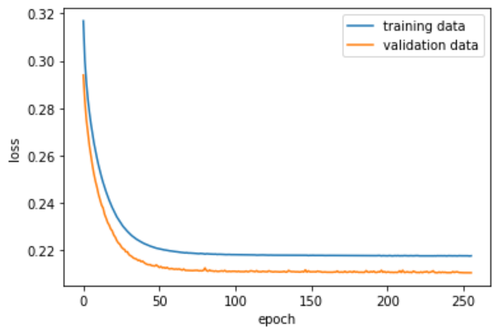
```
high = len(YVALID) - sum(YVALID)
print("Baseline accuracy : ",high/len(YVALID))
```

```
Baseline accuracy :  0.8924870466321243
```

5.3 Logistic regression

Using logistic regression I have obtained an accuracy around **91.3 percent on validation set** and the accuracy started to saturate after training the model over 128 epochs





5.4 Neural Network

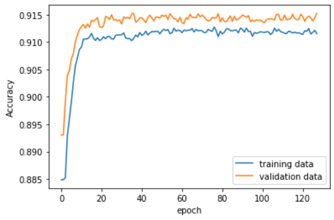
I have performed various experiments which involved trying different combinations of neural layers (by adding/changing layer count), various number of neurons combinations in each layer (4, 8, 16, 32, etc.) along with trying different optimizers and activation fucntions.

5.5 Performance summary and Learning curves of different Neural Networks

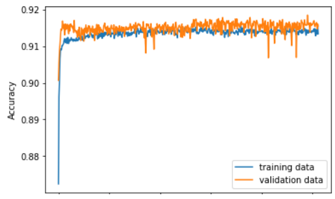
Model.No	Model name	Layers	Layers – Neuron count – Activation function	Epochs	Optimizer	Accuracy	Loss
Model-1	Model_2_r	2	Layer1 - 4-relu Layer2 – 1 – sigmoid	128	Rmsprop	91.5%	19.1%
Model-2	Model_r1	3	Layer1 – 8 – relu Layer2 – 4 – relu Layer3 – 1 – sigmoid	512	Rmsprop	91.5%	17.8%
Model-3	Model_r5	5	Layer1 – 32 – relu Layer2 – 16 – relu Layer3 – 8 – relu Layer4 – 4 – relu Layer5 – 1 – sigmoid	128	rmsprop	91.2%	18.6%
Model-4	Model_r4	6	Layer1 – 16 – relu Layer2 – 8 – relu Layer3 – 4 – relu Layer4 – 4 – relu Layer5 – 4 – relu Layer6 – 1 – sigmoid	128	rmsprop	91.3%	18.9%
Model-5	Model_2	2	Layer1 - 4-relu Layer2 – 1 – sigmoid	128	adam	91.6%	18.42%
Model-6	Model_bst1	3	Layer1 – 8 – relu Layer2 – 4 – relu Layer3 – 1 – sigmoid	256	adam	91.77%	17.4%
Model-7	Model_es1	3	Layer1 – 8 – relu Layer2 – 4 – relu Layer3 – 1 – sigmoid	256-with early stopping	Adam	91.92%	17.7%
Model-8	Model5	4	Layer1 – 8 – relu Layer2 – 4 – relu Layer2 – 4 – relu Layer3 – 1 – sigmoid	128	Adam	91.6%	17.9%
Model-9	Model10	4	Layer1 – 32 – relu Layer2 – 8 – relu Layer2 – 4 – relu Layer3 – 1 – sigmoid	64	adam	91.6%	18.25%
Model-10	Model9	5	Layer1 – 32 – relu Layer2 – 16 – relu Layer3 – 8 – relu Layer4 – 4 – relu Layer5 – 1 – sigmoid	32	adam	91.5%	18.09%
Model-11	Model7	6	Layer1 – 16 – relu Layer2 – 8 – relu Layer3 – 4 – relu Layer4 – 4 – relu Layer5 – 4 – relu Layer6 – 1 – sigmoid	128	adam	91.3%	18.25%
Model-12	Model8	9	Layer1 – 16 – relu Layer2 – 8 – relu Layer3 – 4 – relu Layer4 – 4 – relu Layer5 – 4 – relu Layer6 – 4 – relu Layer7 – 4 – relu Layer8 – 4 – relu Layer9 – 1 – sigmoid	64	Adam	91.5%	18.28%
Model-13	Model	3	Layer1 – 8 – elu Layer2 – 4 – elu Layer3 – 1 – sigmoid	128	adam	91.24%	18.50%

Figure 9: Performance summary of different models

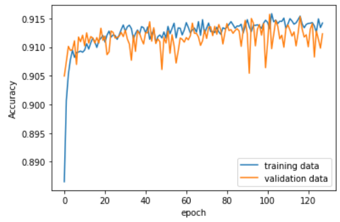
5.5.1 Learning Curves for Neural Network models



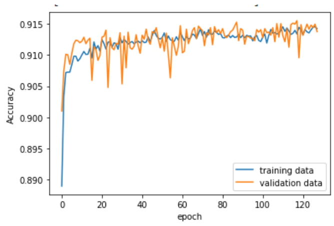
(a) Model1



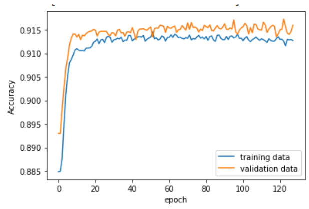
(b) Model2



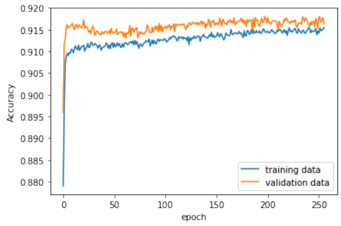
(c) Model3



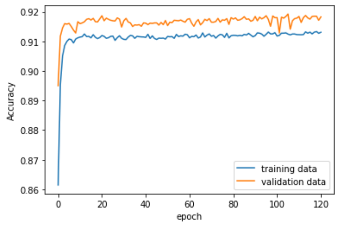
(d) Model4



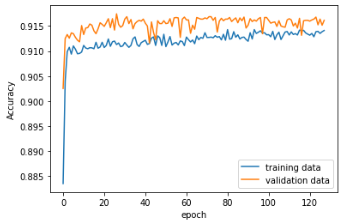
(e) Model5



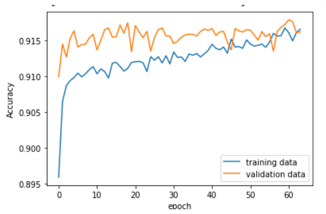
(f) Model6



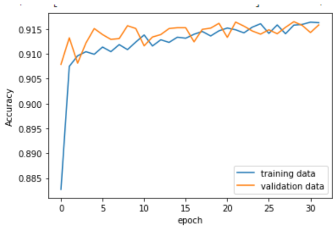
(g) Model7



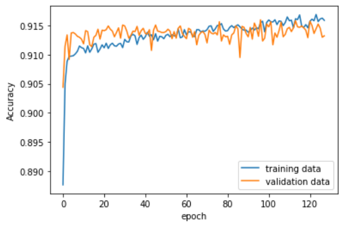
(h) Model8



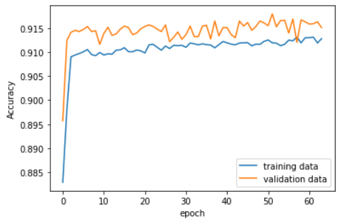
(i) Model9



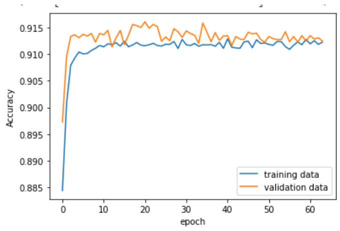
(j) Model10



(k) Model11



(l) Model12

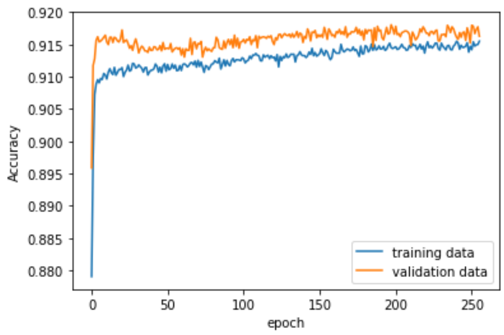


(m) Model13

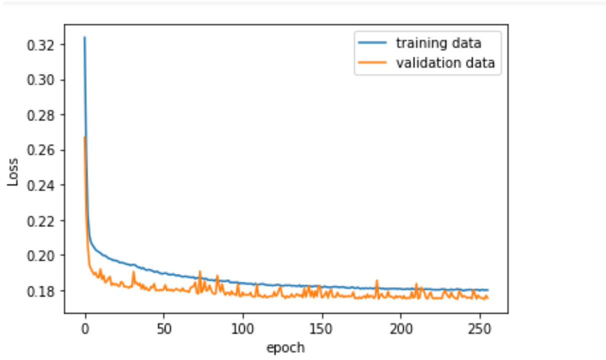
6 Best Neural Network Architecture and Model Evaluation

6.1 Best Neural Network Architecture

From performance summary table **"model-best"** performed well by achieving an accuracy of 91.7 percent.



(a) learning curve of Accuracy



(b) Learning curve of loss

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_75 (Dense)             (None, 8)                 168
_________________________________________________________________
dense_76 (Dense)             (None, 4)                 36
_________________________________________________________________
dense_77 (Dense)             (None, 1)                 5
=================================================================
Total params: 209
Trainable params: 209
Non-trainable params: 0
_________________________________________________________________
```

Figure 12: Summary of Model-bst1

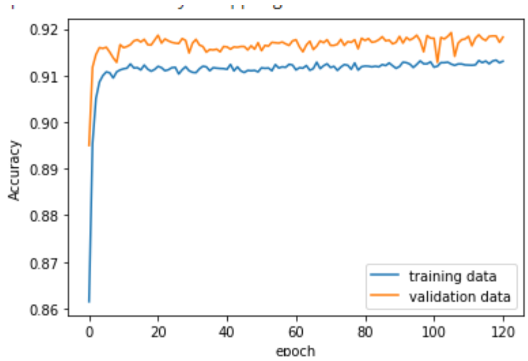
The summary of Model-bst1 depicts the count of the total parameters in the best model neural network. I have considered this model as an optimal model based on these parameters, which are comparatively less than the models that achieve high accuracy and have complex neural network architecture.

6.2 Performance of Neural Network with Callbacks

Applying early stopping concept to my best neural network model helped in achieving higher accuracy. Validation set accuracy raised from 91.7 percent to 91.98 percent. It also prevented my model from overfitting by saving the best accuracy of the model.

6.3 Model Evaluation

The obtained best neural network model can be evaluated using different metrics like Precision, F1-score, recall. The below figure shows the Precision, F1-score, recall values for the best neural network model.



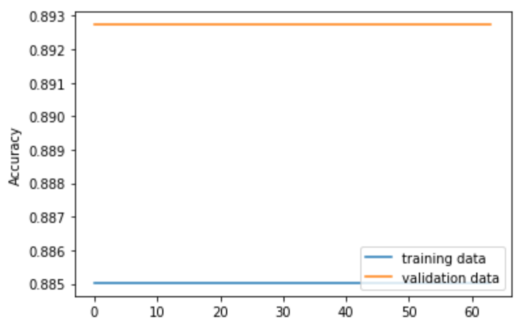
(a) learning curve of Accuracy

```
Epoch 00116: val_accuracy did not improve from 0.91920
Epoch 117/256
961/961 [==============================] - 2s 2ms/step - loss: 0.1817 - accuracy: 0.9130 - val_loss: 0.1768 - val_accuracy: 0.9184

Epoch 00117: val_accuracy did not improve from 0.91920
Epoch 118/256
961/961 [==============================] - 2s 2ms/step - loss: 0.1802 - accuracy: 0.9141 - val_loss: 0.1767 - val_accuracy: 0.9185

Epoch 00118: val_accuracy did not improve from 0.91920
Epoch 119/256
961/961 [==============================] - 2s 2ms/step - loss: 0.1835 - accuracy: 0.9139 - val_loss: 0.1773 - val_accuracy: 0.9185

Epoch 00119: val_accuracy did not improve from 0.91920
Epoch 120/256
961/961 [==============================] - 2s 2ms/step - loss: 0.1791 - accuracy: 0.9140 - val_loss: 0.1811 - val_accuracy: 0.9171

Epoch 00120: val_accuracy did not improve from 0.91920
Epoch 121/256
961/961 [==============================] - 2s 2ms/step - loss: 0.1836 - accuracy: 0.9115 - val_loss: 0.1772 - val_accuracy: 0.9182

Epoch 00121: val_accuracy did not improve from 0.91920
Epoch 00121: early stopping
```

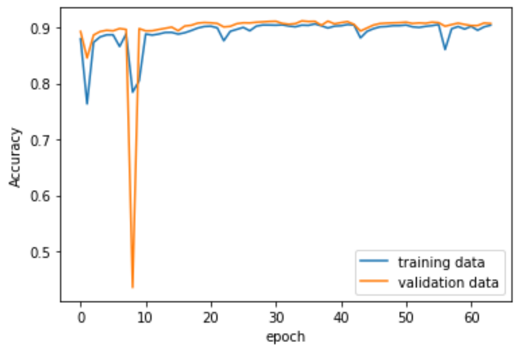
(b) Early stopping at 121st epoch

```
Accuracy Value: 0.92%
Precision Value: 0.62%
Recall Value: 0.54%
F1-score Value: 0.58
```

6.4 Using Linear as an Activation Function

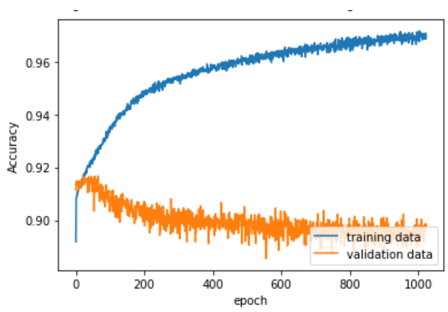


(a) All layers activation as Linear



(b) Activation function of last layer as Linear

7 Neural Network architecture to observe Overfitting



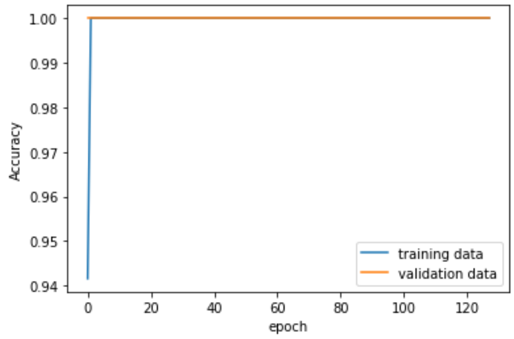
(a) Learning curve of the overfitted model

```
model_o = Sequential()
model_o.add(Dense(64, input_dim = 20,activation='relu'))
model_o.add(Dense(32,activation='relu'))
model_o.add(Dense(16,activation='relu'))
model_o.add(Dense(8,activation='relu'))
model_o.add(Dense(4,activation='relu'))
model_o.add(Dense(1,activation='sigmoid'))
model_o.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
history = model_o.fit(XTRAIN, YTRAIN, validation_data=(XVALID, YVALID), epochs=1024)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.ylabel('Accuracy')
plt.xlabel('epoch')
plt.legend(['training data', 'validation data'], loc='lower right')
plt.show()
```

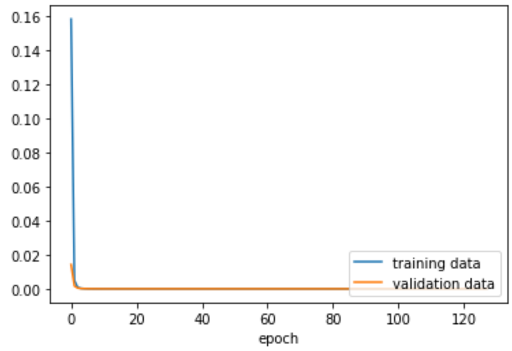
(b) Architecture

The above mentioned model achieved an **accuracy of "97 percent" on training set** when trained over 1024 epochs.

7.1 Overfit Model when output variable is given as Input



(a) Learning curve of Accuracy



(b) Learning curve of Loss

8 Custom function for Best obtained Neural Network

After training the model and obtaining best possible accuracy all the weights and bias were extracted. I have built a function that serves as the obtained model.The output of the custom function and the predictions that I have obtained from actual Neural network model are almost similar. The below images shows the accuracy of custom model and the vales for Custom model, keras model and actual YVALID values.

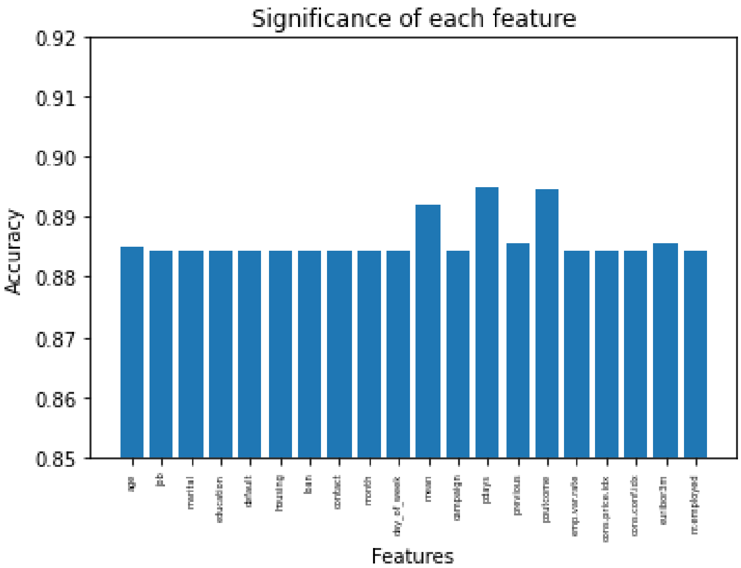
```
Accuracy: 91.63%
Precision: 62.16%
Recall: 54.30%
F1-score: 57.97

Custom Model Predicted Values
====================================================================
X=[0.2 0.2 0.0 0.7 0.0 0.5 0.0 0.0 0.9 0.8 0.0 0.0 1.0 0.0 1.0 0.7 0.4 0.4 0.8 0.9],
Predicted=[0.0]
X=[0.4 0.2 0.3 0.4 0.0 0.5 0.0 0.0 0.9 0.5 0.0 0.0 1.0 0.0 1.0 0.7 0.4 0.4 0.8 0.9],
Predicted=[0.0]
X=[0.2 0.2 0.3 0.6 0.0 0.5 0.0 1.0 0.3 0.2 0.1 0.2 1.0 0.0 1.0 1.0 0.9 0.4 1.0 1.0],
Predicted=[0.0]
X=[0.1 0.9 0.0 0.6 0.0 0.0 0.0 1.0 0.3 0.0 0.2 0.0 1.0 0.0 1.0 1.0 0.9 0.4 1.0 1.0],
Predicted=[0.6]
X=[0.4 0.1 0.3 0.6 1.0 0.0 0.0 1.0 0.3 0.2 0.0 0.1 1.0 0.0 1.0 1.0 0.9 0.4 1.0 1.0],
Predicted=[0.0]

Keras Model Predicted Values
==============================================================
X=[0.2 0.2 0.0 0.7 0.0 0.5 0.0 0.0 0.9 0.8 0.0 0.0 1.0 0.0 1.0 0.7 0.4 0.4 0.8 0.9],
Y=[0.0]
X=[0.4 0.2 0.3 0.4 0.0 0.5 0.0 0.0 0.9 0.5 0.0 0.0 1.0 0.0 1.0 0.7 0.4 0.4 0.8 0.9],
Y=[0.0]
X=[0.2 0.2 0.3 0.6 0.0 0.5 0.0 1.0 0.3 0.2 0.1 0.2 1.0 0.0 1.0 1.0 0.9 0.4 1.0 1.0],
Y=[0.0]
X=[0.1 0.9 0.0 0.6 0.0 0.0 0.0 1.0 0.3 0.0 0.2 0.0 1.0 0.0 1.0 1.0 0.9 0.4 1.0 1.0],
Y=[0.6]
X=[0.4 0.1 0.3 0.6 1.0 0.0 0.0 1.0 0.3 0.2 0.0 0.1 1.0 0.0 1.0 1.0 0.9 0.4 1.0 1.0],
Y=[0.0]

Actual Xvalid Yvalid Values
================================================================
X=[0.2 0.2 0.0 0.7 0.0 0.5 0.0 0.0 0.9 0.8 0.0 0.0 1.0 0.0 1.0 0.7 0.4 0.4 0.8 0.9],
Y=0.0
X=[0.4 0.2 0.3 0.4 0.0 0.5 0.0 0.0 0.9 0.5 0.0 0.0 1.0 0.0 1.0 0.7 0.4 0.4 0.8 0.9],
Y=0.0
X=[0.2 0.2 0.3 0.6 0.0 0.5 0.0 1.0 0.3 0.2 0.1 0.2 1.0 0.0 1.0 1.0 0.9 0.4 1.0 1.0],
Y=0.0
X=[0.1 0.9 0.0 0.6 0.0 0.0 0.0 1.0 0.3 0.0 0.2 0.0 1.0 0.0 1.0 1.0 0.9 0.4 1.0 1.0],
Y=1.0
X=[0.4 0.1 0.3 0.6 1.0 0.0 0.0 1.0 0.3 0.2 0.0 0.1 1.0 0.0 1.0 1.0 0.9 0.4 1.0 1.0],
Y=0.0
```

Figure 17: Custom model output

9 Feature Importance and Reduction

9.1 Feature Importance

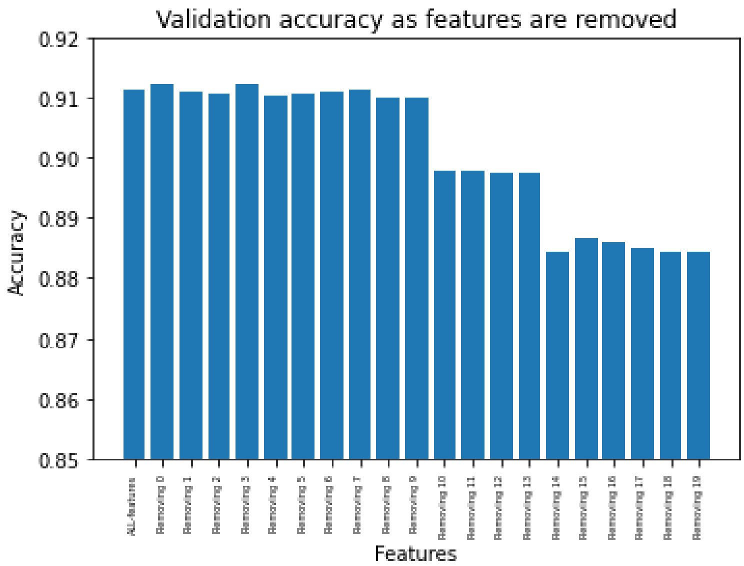
The dataset has 20 independent input features and one output feature. The importance of each feature is obtained by training the best model with feature of interest(one at a time)and by evaluating the model's performance on the validation set. By training our model with one feature at time will help us to determine how strong the influence of that particular feature is there on deciding the output. The below figure shows the relation between single input feature and output in terms of accuracy which is obtained by that feature. From the above figure we



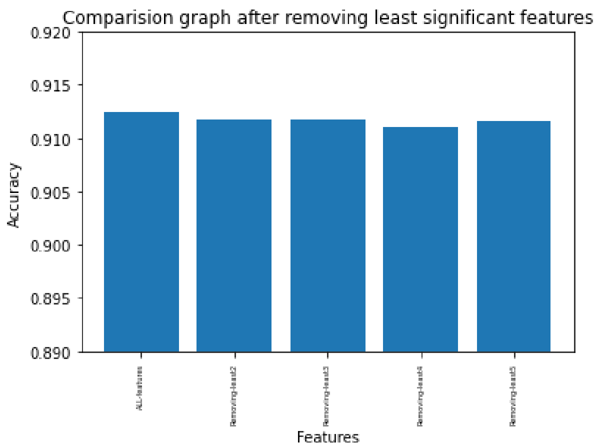
can see the accuracy remained almost constant around 89 percent, for all the features.However accuracy at **"pdays and mean"** features has been increased and to 91 percent and this clearly tells us that these are most important features and have strong influence in deciding the output class label.

9.2 Feature Reduction

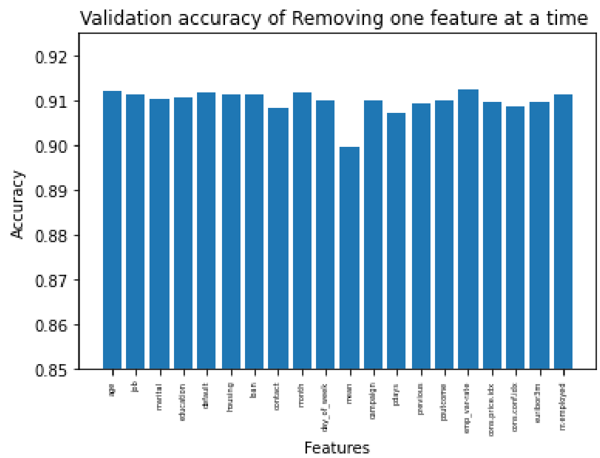
After understanding the importance of each feature I experimented on removing one feature at a time and observed how the value of the accuracy is affected in the absence of that respective feature.



The first bar represents the accuracy of the model which is trained with all the features. We can see the fluctuations in accuracy when first features are removed attractively until ninth feature. However,In the above figure a significant drop in accuracy is observed after removing tenth feature **"mean"**, after that moderate change in accuracy is seen when features like **'cons.price.idx','cons.conf.idx', 'euribor3m'** are removed along with previous features while training the model. Again, sudden drop in accuracy is observed when the model is trained without including first fourteen features. After that the accuracy remained almost constant.



(a) Comparison plots



(b) Accuracy after removing one feature at a time

10 Conclusion

To summarize, the motivation of this project is to try to develop a neural network model to predict whether a customer subscribes for a term term deposit or not. During different stages I have observed various observations. In training phase, I tested different activation function patterns and recorded their effects on the performances and observed how data pre-processing steps like Data cleaning, Normalization and data splitting could make significant changes in accuracy. After experimenting on different neural network architectures, I have realised how over training or under training can lead to overfitting, underfitting problem. By carefully observing learning curves and accordingly adjusting the values for hyper-parameters(neurons count,layers,epochs, batch size),**model-bst1** Neural network performed well. It achieved an accuracy of **91.7 percent** and with the help of model check-point and early stopping the validation accuracy raised to **91.912 percent**. Thus, further experiments are done by including feature importance and applying feature reduction.

11 References

[1].Dataset source-https://archive.ics.uci.edu/ml/datasets/Bank+Marketing
[2].[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014
[3].https://www.kaggle.com/joshuashamouil/bank-marketing-with-classification-modeling.
[4].https://www.tableau.com/learn/articles/what-is-data-cleaning.
[5].https://youtu.be/Tu8Dl3zorgg.