Technological Institute of the Philippines	Quezon City - Computer Engineering
Course Code:	CPE 019
Code Title:	Emerging Technologies in CpE 2
2nd Semester	AY 2023-2024

ACTIVITY	Midterm Quiz 1
Name	Cuevas, Christian Jay L.
Section	CPE32S3
Date Performed:	4/3/2024
Date Submitted:	4/3/2024
Instructor:	Engr. Roman M. Richard

INSTRUCTION

- Use this provided dataset: PhiUSIIL Phishing URL (Website) - UCI Machine Learning Repository Links to an external site.
- Perform:

  Task 1: Exploratory Data Analysis (Cleaning + Prepping the dataset)

  Task 2: Data modelling using ANN

DATASET

- This dataset contains 134,850 legitimate URLS and 100,945 phishing URLS. The total number of the entries in the dataset is 235,795. There are also a total of 55 variables, 54 features and 1 target variable. The goal of this model is to distinguish legitimate and phishing URLs.

Importing libraries and the dataset

```
""" Install and Import UCIMLRepo Library """
!pip install ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.6-py3-none-any.whl (8.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.6
```

```
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
```

```
import matplotlib.pyplot as plt
from ucimlrepo import fetch_ucirepo
from sklearn.preprocessing import MinMaxScaler
from keras.optimizers import Adam, SGD, RMSprop
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Activation
from sklearn.metrics import confusion_matrix, precision_recall_curve, roc_auc_score, roc_curve, ac
```

```
# fetch dataset
phiusiil_phishing_url_website = fetch_ucirepo(id=967)

# data (as pandas dataframes)
X = phiusiil_phishing_url_website.data.features
y = phiusiil_phishing_url_website.data.targets
```

```
all_data = phiusiil_phishing_url_website.data.original
```

Performing Exploratory Data Analysis

```
""" Checking the .info """

all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 235795 entries, 0 to 235794
Data columns (total 56 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   FILENAME                    235795 non-null  object
 1   URL                         235795 non-null  object
 2   URLLength                   235795 non-null  int64
 3   Domain                      235795 non-null  object
 4   DomainLength                235795 non-null  int64
 5   IsDomainIP                  235795 non-null  int64
 6   TLD                         235795 non-null  object
 7   URLSimilarityIndex          235795 non-null  float64
 8   CharContinuationRate        235795 non-null  float64
 9   TLDLegitimateProb           235795 non-null  float64
 10  URLCharProb                 235795 non-null  float64
 11  TLDLength                   235795 non-null  int64
 12  NoOfSubDomain               235795 non-null  int64
 13  HasObfuscation              235795 non-null  int64
 14  NoOfObfuscatedChar          235795 non-null  int64
 15  ObfuscationRatio            235795 non-null  float64
 16  NoOfLettersInURL            235795 non-null  int64
 17  LetterRatioInURL            235795 non-null  float64
 18  NoOfDegitsInURL             235795 non-null  int64
 19  DegitRatioInURL             235795 non-null  float64
 20  NoOfEqualsInURL             235795 non-null  int64
 21  NoOfQMarkInURL              235795 non-null  int64
 22  NoOfAmpersandInURL          235795 non-null  int64
 23  NoOfOtherSpecialCharsInURL  235795 non-null  int64
 24  SpacialCharRatioInURL       235795 non-null  float64
 25  IsHTTPS                     235795 non-null  int64
 26  LineOfCode                  235795 non-null  int64
 27  LargestLineLength           235795 non-null  int64
```

```
 28  HasTitle                  235795 non-null  int64
 29  Title                     235795 non-null  object
 30  DomainTitleMatchScore     235795 non-null  float64
 31  URLTitleMatchScore        235795 non-null  float64
 32  HasFavicon                235795 non-null  int64
 33  Robots                    235795 non-null  int64
 34  IsResponsive              235795 non-null  int64
 35  NoOfURLRedirect           235795 non-null  int64
 36  NoOfSelfRedirect          235795 non-null  int64
 37  HasDescription            235795 non-null  int64
 38  NoOfPopup                 235795 non-null  int64
 39  NoOfiFrame                235795 non-null  int64
 40  HasExternalFormSubmit     235795 non-null  int64
 41  HasSocialNet              235795 non-null  int64
 42  HasSubmitButton           235795 non-null  int64
 43  HasHiddenFields           235795 non-null  int64
 44  HasPasswordField          235795 non-null  int64
 45  Bank                      235795 non-null  int64
 46  Pay                       235795 non-null  int64
 47  Crypto                    235795 non-null  int64
 48  HasCopyrightInfo          235795 non-null  int64
 49  NoOfImage                 235795 non-null  int64
 50  NoOfCSS                   235795 non-null  int64
 51  NoOfJS                    235795 non-null  int64
 52  NoOfSelfRef               235795 non-null  int64
```

Observation:

- We can see here that there are a total of 235795 entries for the dataset. The number of variables is 55, there are 54 features and 1 target variable which is the label. We can also look at the data types, there are 5 objects, 10 float64, and 41 int64.

```
1 """Check if there are null values"""
2 all_data.isnull().sum()
```

```
FILENAME                    0
URL                         0
URLLength                   0
Domain                      0
DomainLength                0
IsDomainIP                  0
TLD                         0
URLSimilarityIndex          0
CharContinuationRate        0
TLDLegitimateProb           0
URLCharProb                 0
TLDLength                   0
NoOfSubDomain               0
HasObfuscation              0
NoOfObfuscatedChar          0
ObfuscationRatio            0
NoOfLettersInURL            0
LetterRatioInURL            0
NoOfDegitsInURL             0
DegitRatioInURL             0
NoOfEqualsInURL             0
NoOfQMarkInURL              0
```

```
NoOfAmpersandInURL            0
NoOfOtherSpecialCharsInURL    0
SpacialCharRatioInURL         0
IsHTTPS                       0
LineOfCode                    0
LargestLineLength             0
HasTitle                      0
Title                         0
DomainTitleMatchScore         0
URLTitleMatchScore            0
HasFavicon                    0
Robots                        0
IsResponsive                  0
NoOfURLRedirect               0
NoOfSelfRedirect              0
HasDescription                0
NoOfPopup                     0
NoOfiFrame                    0
HasExternalFormSubmit         0
HasSocialNet                  0
HasSubmitButton               0
HasHiddenFields               0
HasPasswordField              0
Bank                          0
Pay                           0
Crypto                        0
HasCopyrightInfo              0
NoOfImage                     0
NoOfCSS                       0
NoOfJS                        0
NoOfSelfRef                   0
NoOfEmptyRef                  0
NoOfExternalRef               0
label                         0
dtype: int64
```

Observation:

- We can observe that there are no null values, so we don't have to do any fill or drop any rows to clean this dataset.

```
1 """Display the first fifteen rows"""
2 all_data.head(15)
```

	FILENAME	URL	URLLength	Domain	DomainLength	IsDom
0	521848.txt	https://www.southbankmosaics.com	31	www.southbankmosaics.com	24	
1	31372.txt	https://www.uni-mainz.de	23	www.uni-mainz.de	16	
2	597387.txt	https://www.voicefmradio.co.uk	29	www.voicefmradio.co.uk	22	
3	554095.txt	https://www.sfnmjournal.com	26	www.sfnmjournal.com	19	
4	151578.txt	https://www.rewildingargentina.org	33	www.rewildingargentina.org	26	

5	23107.txt	https://www.globalreporting.org	30	www.globalreporting.org	23
6	23034.txt	https://www.saffronart.com	25	www.saffronart.com	18
7	696732.txt	https://www.nerdscandy.com	25	www.nerdscandy.com	18
8	739255.txt	https://www.hyderabadonline.in	29	www.hyderabadonline.in	22
9	14486.txt	https://www.aap.org	18	www.aap.org	11
10	167350.txt	https://www.religionenlibertad.com	33	www.religionenlibertad.com	26
11	mw42508.txt	http://www.teramill.com	22	www.teramill.com	16
12	515489.txt	https://www.socialpolicy.org	27	www.socialpolicy.org	20
13	858208.txt	https://www.aoh61.com	20	www.aoh61.com	13
14	712305.txt	https://www.bulgariaski.com	26	www.bulgariaski.com	19

15 rows × 56 columns

Observation:

- We can observe here that there are links that are in values in the column of FILENAME, URL, URLLength, Domain, TLD. The rest are int64 and float64, and in the label column, there are values of 0 and 1. The value 0 represents the legitimate websites, while the value 1 represents the phishing websites.

```
1 """Observing the count, mean, std, min, 25%, 50%, 75% and max to identify outliers """
2 all_data.describe()
```

	URLLength	DomainLength	IsDomainIP	URLSimilarityIndex	CharContinuationRate	TLDL
count	235795.000000	235795.000000	235795.000000	235795.000000	235795.000000	
mean	34.573095	21.470396	0.002706	78.430778	0.845508	
std	41.314153	9.150793	0.051946	28.976055	0.216632	
min	13.000000	4.000000	0.000000	0.155574	0.000000	
25%	23.000000	16.000000	0.000000	57.024793	0.680000	
50%	27.000000	20.000000	0.000000	100.000000	1.000000	
75%	34.000000	24.000000	0.000000	100.000000	1.000000	
max	6097.000000	110.000000	1.000000	100.000000	1.000000	

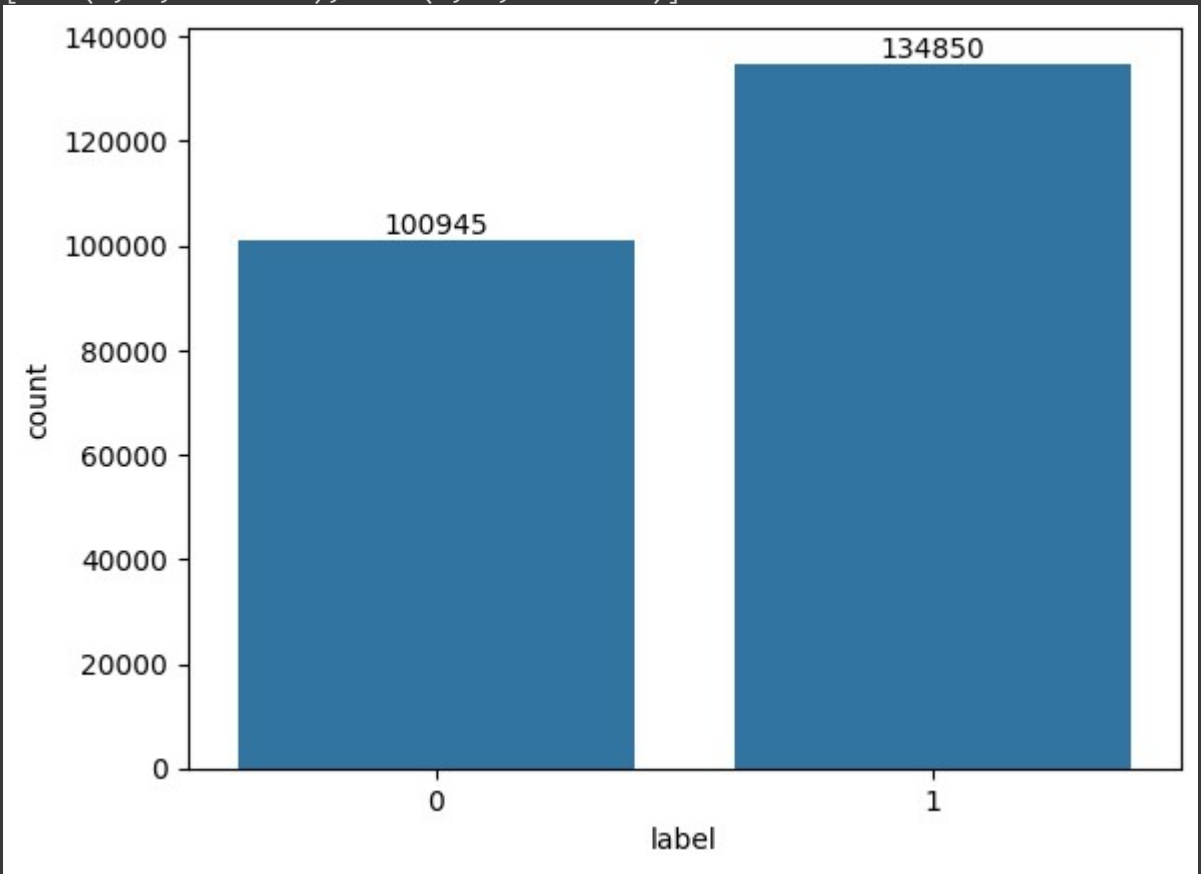
8 rows × 51 columns

Observation:

- Here we can oberve that the 75% and max of some values are too far apart, this signifies that there are some outliers and may affect the overall performance of the model. We can use scalers to normalize this dataset, I will use MinMaxScaler() in particular to scale this dataset.

```
1 ax = sns.countplot(data=all_data, x='label')
2 ax.bar_label(ax.containers[0])
```

```
[Text(0, 0, '100945'), Text(0, 0, '134850')]
```



Observation:

- You can observe that the instance of the class "0" is higher than the instance of the class "1". The number of legitimate website is 100,945 while the number of mines instances is 134,850.
- Why do we need to check for imbalance? We check for class imbalance because this can affect the predictive performance of our model [2] and it can also produce bias when the model is making a decision [1,2].
- To check for the degree of imbalance, we can use the Imbalance Ratio, which is denoted by IR = total of negative class / total of positive class [1]. Positive class is the class we're testing for [3] which is the "0" class, and the negative class is the class is the other possibility that our model is also testing [4].
- In our case, IR = 100,945 / 134,850 is equal to 0.7486, which is close to 1. When the result of the ratio is close to 1, this means that our dataset is not imbalanced [5].

```
1 """Removing the columns with Object"""
2
```

```
3 columns = [ 'FILENAME', 'URL', 'Domain', 'TLD', 'Title' ]
4 all_data_features = all_data.drop(columns, axis = 1)
```

```
1 all_data_features.corr(method="pearson", numeric_only = True)
```

	URLLength	DomainLength	IsDomainIP	URLSimilarityIndex	CharContinua
URLLength	1.000000	0.243091	0.277272	-0.421104	
DomainLength	0.243091	1.000000	-0.041398	-0.420694	
IsDomainIP	0.277272	-0.041398	1.000000	-0.121439	
URLSimilarityIndex	-0.421104	-0.420694	-0.121439	1.000000	
CharContinuationRate	-0.241104	-0.577425	-0.101307	0.572938	
TLDLegitimateProb	-0.016656	0.027209	-0.053220	0.162042	
URLCharProb	-0.245215	-0.178104	-0.111769	0.571622	
TLDLength	0.033339	0.152669	-0.011125	-0.053915	
NoOfSubDomain	0.071058	0.399157	0.084619	-0.058481	
HasObfuscation	0.161980	0.041643	0.001239	-0.100159	
NoOfObfuscatedChar	0.459094	0.011304	0.000354	-0.032257	
ObfuscationRatio	0.127472	0.021627	0.000977	-0.078989	
NoOfLettersInURL	0.956047	0.277602	0.244300	-0.449120	
LetterRatioInURL	0.312632	0.547854	-0.002630	-0.491761	
NoOfDegitsInURL	0.835809	0.131709	0.224703	-0.299591	
DegitRatioInURL	0.304379	0.289628	0.161257	-0.561238	
NoOfEqualsInURL	0.685091	0.020033	0.288351	-0.161075	
NoOfQMarkInURL	0.474901	0.064477	0.389523	-0.341722	
NoOfAmpersandInURL	0.500387	0.016320	0.178814	-0.075346	
NoOfOtherSpecialCharsInURL	0.782582	0.272930	0.285846	-0.540554	
SpacialCharRatioInURL	0.199112	0.182803	0.115484	-0.604885	
IsHTTPS	0.013117	-0.019218	-0.011741	0.346389	
LineOfCode	-0.060636	-0.075171	-0.016211	0.231147	
LargestLineLength	0.045219	0.067131	0.002159	-0.078496	
HasTitle	-0.075456	-0.107133	-0.004602	0.352123	
DomainTitleMatchScore	-0.215877	-0.296493	-0.052033	0.604045	
URLTitleMatchScore	-0.193234	-0.328603	-0.054458	0.543969	
HasFavicon	-0.094548	-0.148317	-0.030890	0.402334	
Robots	-0.073770	-0.082594	-0.028261	0.313293	
IsResponsive	-0.082549	-0.119737	-0.006481	0.433413	

NoOfURLRedirect	0.029975	0.019387	0.023497	-0.060840
NoOfSelfRedirect	-0.005168	-0.046803	-0.010231	-0.049634
HasDescription	-0.149104	-0.202821	-0.021518	0.590195
NoOfPopup	-0.010952	-0.012502	-0.002795	0.041042
NoOfiFrame	-0.044875	-0.051687	-0.009741	0.191208
HasExternalFormSubmit	-0.035443	-0.038560	-0.007988	0.139202
HasSocialNet	-0.180909	-0.216055	-0.046432	0.673690
HasSubmitButton	-0.067420	-0.111077	-0.011489	0.448227
HasHiddenFields	-0.072551	-0.111882	-0.011962	0.404655
HasPasswordField	0.021561	0.009409	-0.008419	0.061553
Bank	-0.031734	-0.039944	-0.015708	0.151290
Pay	-0.057816	-0.065895	-0.013863	0.291542
Crypto	-0.025163	-0.034773	-0.005919	0.087228
HasCopyrightInfo	-0.126420	-0.196860	-0.022796	0.613620
NoOfImage	-0.064639	-0.083458	-0.016502	0.235728
NoOfCSS	-0.014507	-0.017596	-0.004149	0.057683
NoOfJS	-0.078643	-0.102216	-0.023126	0.315679
NoOfSelfRef	-0.074722	-0.096712	-0.018981	0.271590
NoOfEmptyRef	-0.023206	-0.031673	-0.006793	0.091556
NoOfExternalRef	-0.059055	-0.070099	-0.015269	0.221960
label	-0.233445	-0.283152	-0.060202	0.860358

51 rows × 51 columns

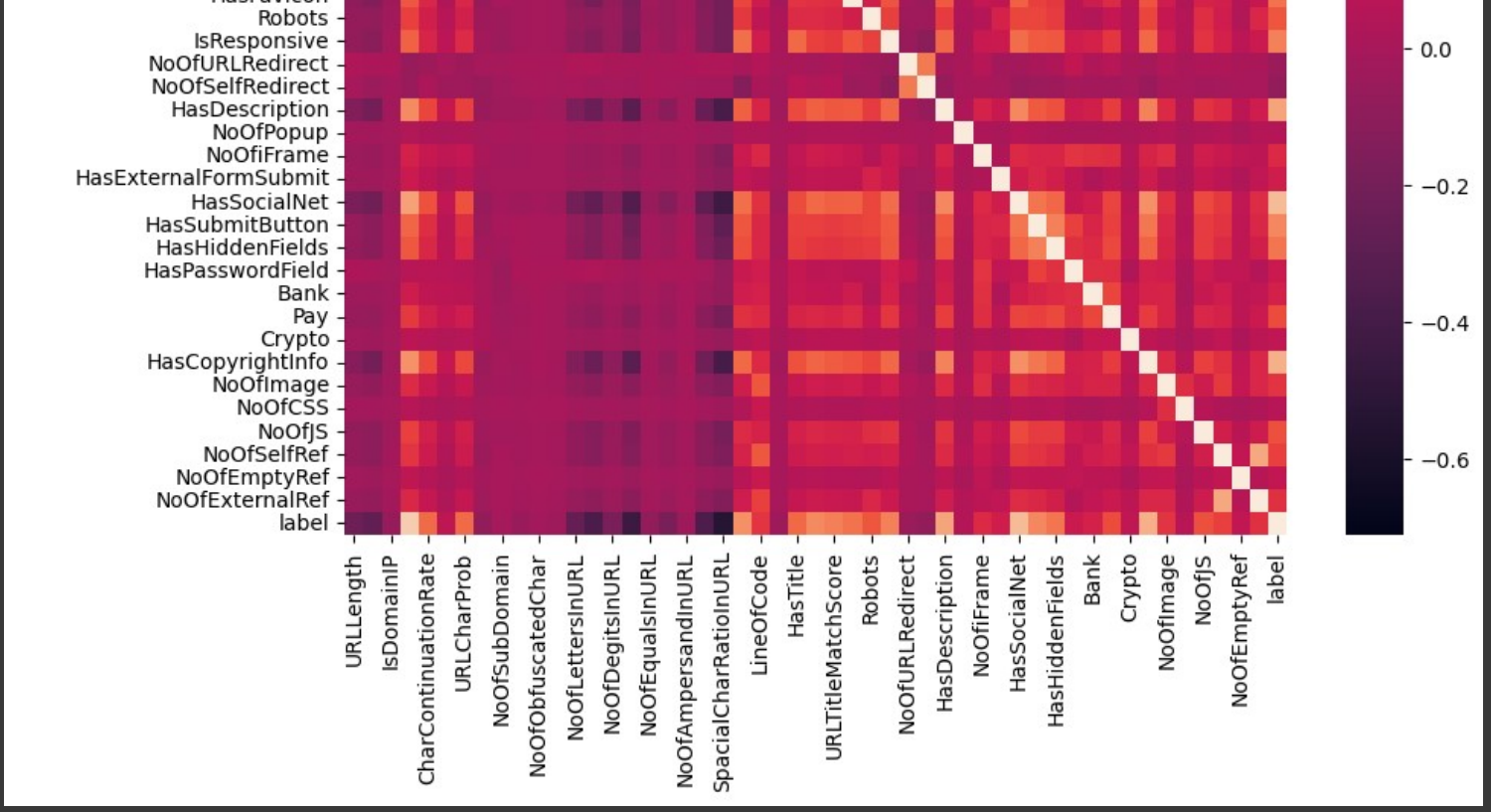
```
1 all_data_features.corr(method="pearson", numeric_only = True)['label'].sort_values(ascending=False
```

```
label                      1.000000
URLSimilarityIndex         0.860358
HasSocialNet               0.784255
HasCopyrightInfo           0.743358
HasDescription             0.690232
IsHTTPS                    0.609132
DomainTitleMatchScore      0.584905
HasSubmitButton            0.578561
IsResponsive               0.548608
URLTitleMatchScore         0.539419
HasHiddenFields            0.507731
HasFavicon                 0.493711
URLCharProb                0.469749
CharContinuationRate       0.467735
HasTitle                   0.459725
Robots                     0.392620
NoOfJS                     0.373500
```

```
Pay                          0.359747
NoOfSelfRef                  0.316211
NoOfImage                    0.274658
LineOfCode                   0.272257
NoOfExternalRef              0.258627
NoOfiFrame                   0.225822
Bank                         0.188959
HasExternalFormSubmit        0.167574
HasPasswordField             0.138183
NoOfEmptyRef                 0.109235
Crypto                       0.099610
TLDLegitimateProb            0.097389
NoOfCSS                      0.068109
NoOfPopup                    0.047391
NoOfSubDomain               -0.005955
NoOfObfuscatedChar          -0.015315
NoOfAmpersandInURL          -0.034622
LargestLineLength           -0.041111
ObfuscationRatio            -0.041915
NoOfURLRedirect             -0.046456
HasObfuscation              -0.052473
IsDomainIP                  -0.060202
NoOfSelfRedirect            -0.076463
NoOfEqualsInURL             -0.076963
TLDLength                   -0.079159
NoOfQMarkInURL              -0.175621
NoOfDegitsInURL             -0.177980
URLLength                   -0.233445
NoOfLettersInURL            -0.258090
DomainLength                -0.283152
NoOfOtherSpecialCharsInURL  -0.358891
LetterRatioInURL            -0.367794
DegitRatioInURL             -0.432032
SpacialCharRatioInURL       -0.533537
Name: label, dtype: float64
```

```
1 fig = plt.figure(figsize=(10,10), dpi=100)
2
3 sns.heatmap(all_data_features.corr())
```

```
<Axes: >
```



Observation:

- We can observe from the above the different correlation of the features to the target variable "label". There are features that has high positive correlation with the target variable and there are also feature that has high negative correlation with the target variable. We can use this later for feature selection after training our first model.

DATA PRE-PROCESSING

- We pre-processed the data using the MinMaxScaler(). This is done because there is a significant difference between the 75% of the data and the maximum of the data. We also use this to avoid bias and to give equal weights to every feature.

```
1 """Normalizing the data"""
```

```
minmax = MinMaxScaler()
all_data_norm = minmax.fit_transform(all_data_features)
```

```
df_all_data = pd.DataFrame(all_data_norm)
df_all_data.head()
```

	0	1	2	3	4	5	6	7	8	9	...	41	42	43	
0	0.002959	0.188679	0.0	1.0	1.000000	1.000000	0.678068	0.090909	0.1	0.0	...	0.0	0.0	1.0	0.
1	0.001644	0.113208	0.0	1.0	0.666667	0.062440	0.547403	0.000000	0.1	0.0	...	0.0	0.0	1.0	0.
2	0.002630	0.169811	0.0	1.0	0.866667	0.054608	0.702533	0.000000	0.2	0.0	...	0.0	0.0	1.0	0
3	0.002137	0.141509	0.0	1.0	1.000000	1.000000	0.629847	0.090909	0.1	0.0	...	1.0	1.0	1.0	0.
4	0.003287	0.207547	0.0	1.0	1.000000	0.152920	0.650301	0.090909	0.1	0.0	...	1.0	0.0	1.0	0.

5 rows × 51 columns

```
df_all_data.columns = all_data_features.columns
```

```
df_all_data.head()
```

	URLLength	DomainLength	IsDomainIP	URLSimilarityIndex	CharContinuationRate	TLDLegitimatePr
0	0.002959	0.188679	0.0	1.0	1.000000	1.0000
1	0.001644	0.113208	0.0	1.0	0.666667	0.0624
2	0.002630	0.169811	0.0	1.0	0.866667	0.0546
3	0.002137	0.141509	0.0	1.0	1.000000	1.0000
4	0.003287	0.207547	0.0	1.0	1.000000	0.1529

5 rows × 51 columns

SPLITTING THE DATASET INTO TRAINING, VALIDATION AND TEST DATA:

- We are splitting the dataset into training, validation and test data so that we can see the predictive power of our model. We will split the data into 60/20/20, 60 for the training, 20 for the testing and 20 for the validation [8][9].

```
X = df_all_data.drop(["label"], axis = 1)
y = df_all_data["label"]

X, X_test, y, y_test = train_test_split(X, y, test_size =0.2, random_state=10)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size =0.2, random_state=10)
```

```
1
```

```
1 X_train.shape, X_test.shape, X_val.shape
```

```
((150908, 50), (47159, 50), (37728, 50))
```

```
1 y_train.shape, y_test.shape, y_val.shape
```

```
((150908,), (47159,), (37728,))
```

TRAINING THE MODEL

- Here we trained our model with 1 hidden layer that has 33 hidden layer nodes. I got this value by multiplying 50 by 2/3, as according to the "rule of thumb" [7].

```
1 model = tf.keras.models.Sequential([
2     tf.keras.layers.Dense(33, input_shape=(50,), activation = "relu"),
3     tf.keras.layers.Dense(1, activation = "sigmoid")
4 ])
```

```
1 model.compile(Adam(learning_rate = 0.001),
2               loss = "binary_crossentropy",
3               metrics=["accuracy"]
4               )
```

```
1 URL_model1 = model.fit(X_train, y_train, validation_data = (X_val, y_val), batch_size = 320 ,epoch
```

```
Epoch 1/10
472/472 [==============================] - 2s 3ms/step - loss: 0.1607 - accuracy: 0.9571 - val_lc
Epoch 2/10
472/472 [==============================] - 1s 2ms/step - loss: 0.0248 - accuracy: 0.9938 - val_lc
Epoch 3/10
472/472 [==============================] - 1s 2ms/step - loss: 0.0119 - accuracy: 0.9977 - val_lc
Epoch 4/10
472/472 [==============================] - 1s 2ms/step - loss: 0.0072 - accuracy: 0.9986 - val_lc
Epoch 5/10
472/472 [==============================] - 1s 2ms/step - loss: 0.0049 - accuracy: 0.9991 - val_lc
Epoch 6/10
472/472 [==============================] - 1s 3ms/step - loss: 0.0036 - accuracy: 0.9993 - val_lc
Epoch 7/10
472/472 [==============================] - 2s 3ms/step - loss: 0.0027 - accuracy: 0.9995 - val_lc
Epoch 8/10
472/472 [==============================] - 1s 2ms/step - loss: 0.0022 - accuracy: 0.9996 - val_lc
Epoch 9/10
472/472 [==============================] - 1s 2ms/step - loss: 0.0018 - accuracy: 0.9997 - val_lc
Epoch 10/10
472/472 [==============================] - 1s 2ms/step - loss: 0.0015 - accuracy: 0.9997 - val_lc
```

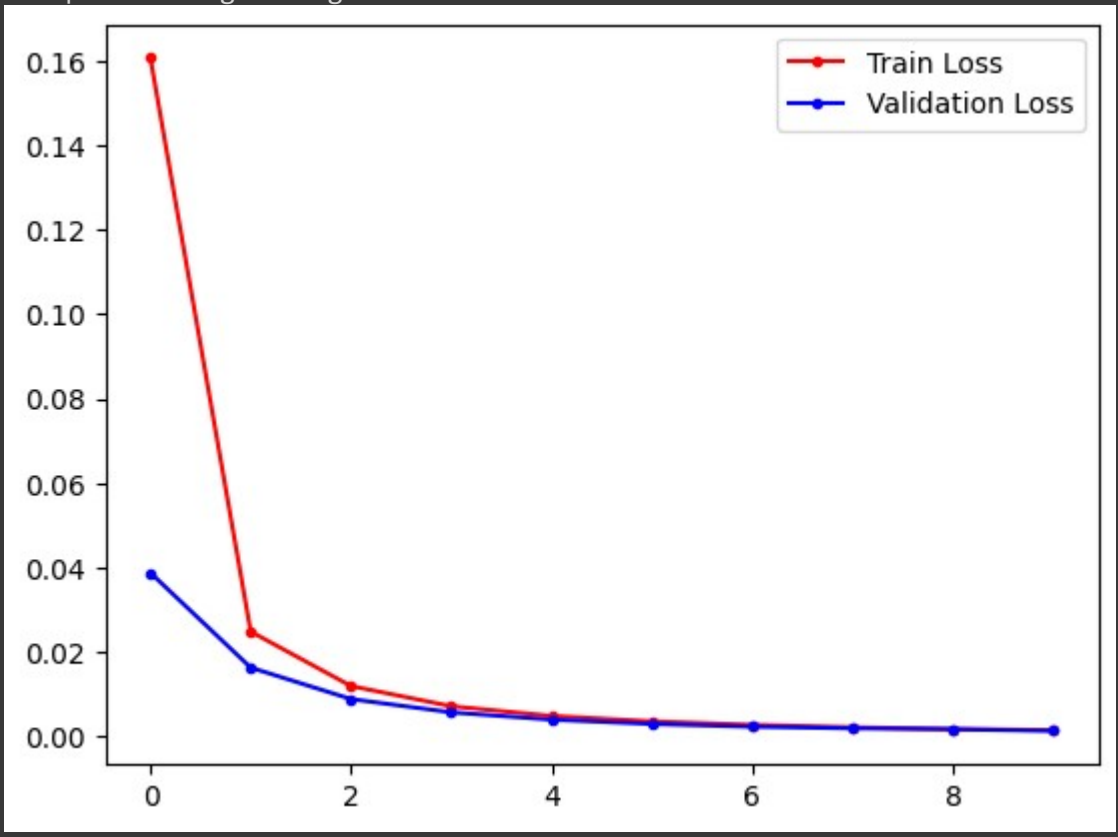
```
1 # The accuracy of the model with the Test Data
2 model.evaluate(X_test, y_test)
```

```
1474/1474 [==============================] - 2s 1ms/step - loss: 0.0015 - accuracy: 0.9997
[0.0014898621011525393, 0.9996607303619385]
```

```
1 fig, ax = plt.subplots()
2 ax.plot(URL_model1.history["loss"],'r', marker='.', label="Train Loss")
```

```
ax.plot(URL_model1.history["val_loss"],'b', marker='.', label="Validation Loss")
ax.legend()
```

```
<matplotlib.legend.Legend at 0x79ac7ad033a0>
```



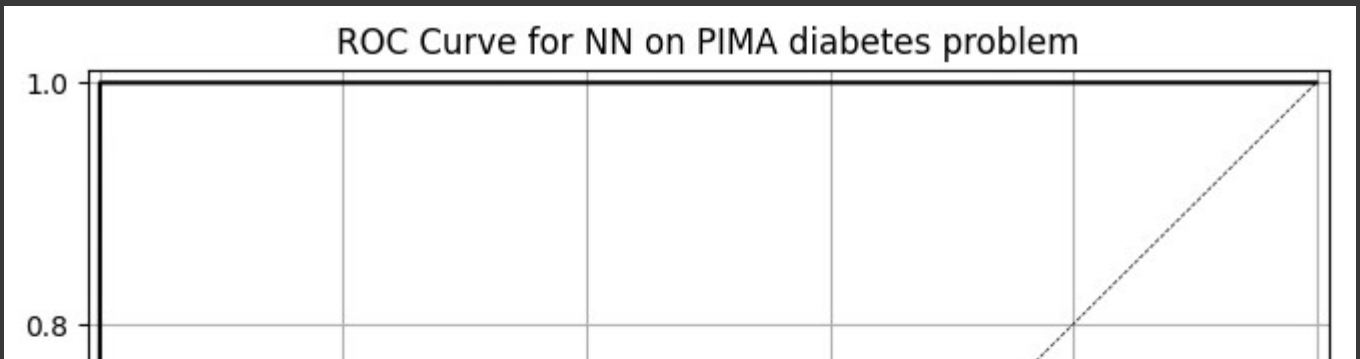

```
y_pred_prob_nn_1 = model.predict(X_test)
```

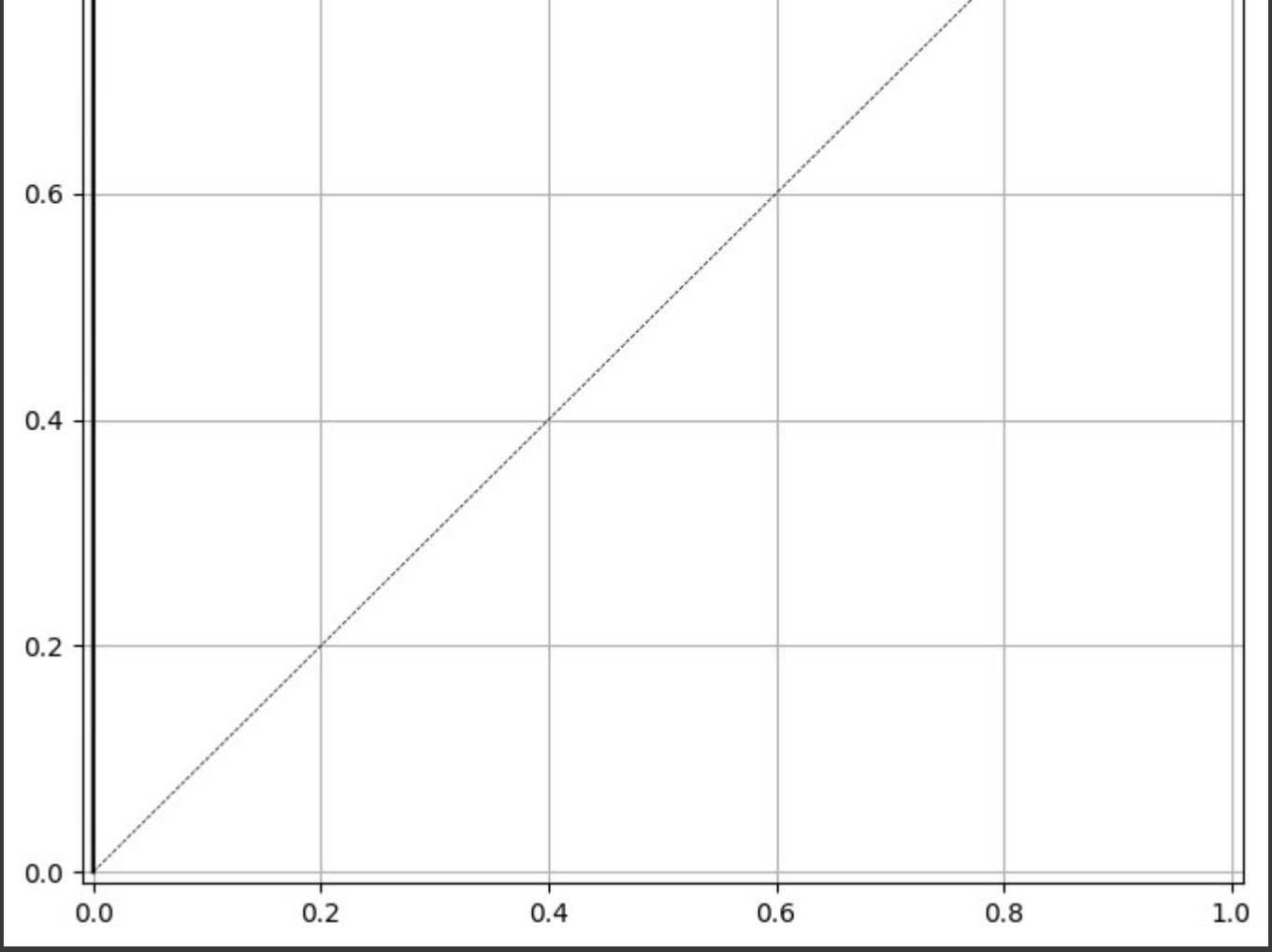
```
1474/1474 [==============================] - 7s 4ms/step
1474/1474 [==============================] - 4s 3ms/step
```

```
def plot_roc(y_test, y_pred, model_name):
    fpr, tpr, thr = roc_curve(y_test, y_pred)
    fig, ax = plt.subplots(figsize=(8, 8))
    ax.plot(fpr, tpr, 'k-')
    ax.plot([0, 1], [0, 1], 'k--', linewidth=.5)  # roc curve for random model
    ax.grid(True)
    ax.set(title='ROC Curve for {} on PIMA diabetes problem'.format(model_name),
           xlim=[-0.01, 1.01], ylim=[-0.01, 1.01])

```

```
print('roc-auc is {:.3f}'.format(roc_auc_score(y_test,y_pred_prob_nn_1)))

plot_roc(y_test, y_pred_prob_nn_1, 'NN')
```

```
roc-auc is 1.000
```





Observation:

- We can observe here that the model is performing very well and the accuracy is 0.99 and the loss is nearly 0.0001. We can also look at the accuracy and the ROC curve above, the accuracy of this model in the test data is 0.997 and the value of its ROC curve is 1.000. We can try to further optimize it through feature selection.

Optimizing the model by feature selection

- We will train the second model using only the features that has high coefficient with our target variable. High positive coefficient is in the range of 0.7 - 1 [6], we will only use the features with values greater than or equal 0.7. According to the correlation that we did earlier, these values are the following:
- URLSimilarityIndex = 0.860358
- HasSocialNet = 0.784255
- HasCopyrightInfo = 0.743358

```
1 df_all_data1 = df_all_data.filter(["URLSimilarityIndex", "HasSocialNet", "HasCopyrightInfo"])
```

```
X1 = df_all_data1
y1 = df_all_data["label"]

X1, X1_test, y1, y1_test = train_test_split(X1, y1, test_size =0.2, random_state=10)
X1_train, X1_val, y1_train, y1_val = train_test_split(X1, y1, test_size =0.2, random_state=10)
```

```
model2 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(3, input_shape=(3,), activation = "relu"),
    tf.keras.layers.Dense(1, activation = "sigmoid")
])
```

```
model2.compile(Adam(learning_rate = 0.001),
               loss = "binary_crossentropy",
               metrics=["accuracy"]
               )
```

```
URL_model2 = model2.fit(X1_train, y1_train, validation_data = (X1_val, y1_val), batch_size = 320 ,
```

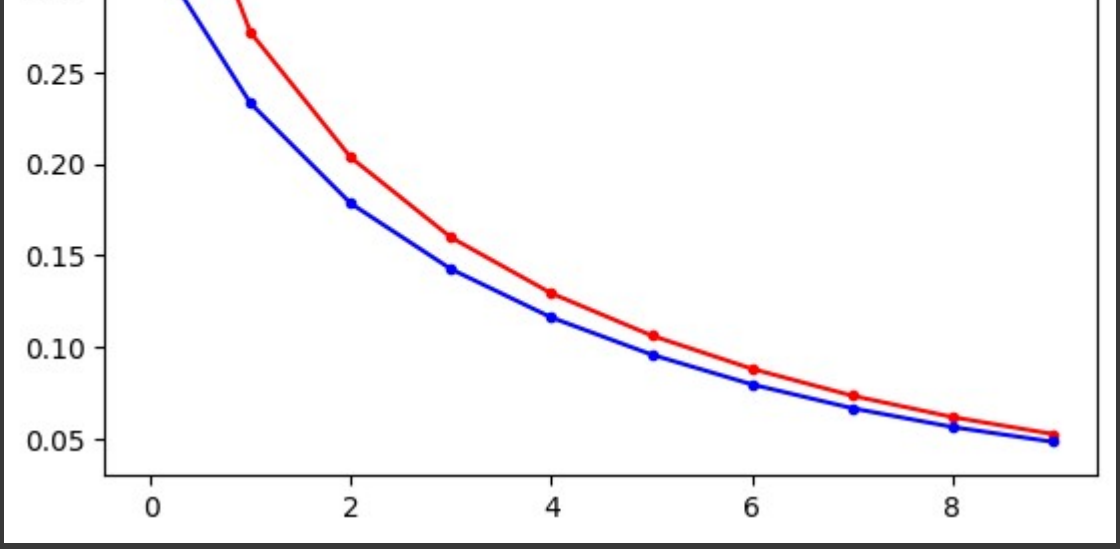
```
Epoch 1/10
472/472 [==============================] - 2s 3ms/step - loss: 0.4150 - accuracy: 0.7906 - val_lc
Epoch 2/10
472/472 [==============================] - 1s 2ms/step - loss: 0.2716 - accuracy: 0.9465 - val_lc
Epoch 3/10
472/472 [==============================] - 1s 2ms/step - loss: 0.2033 - accuracy: 0.9531 - val_lc
Epoch 4/10
472/472 [==============================] - 1s 2ms/step - loss: 0.1597 - accuracy: 0.9620 - val_lc
Epoch 5/10
472/472 [==============================] - 1s 2ms/step - loss: 0.1293 - accuracy: 0.9639 - val_lc
Epoch 6/10
472/472 [==============================] - 1s 2ms/step - loss: 0.1063 - accuracy: 0.9737 - val_lc
Epoch 7/10
472/472 [==============================] - 1s 2ms/step - loss: 0.0881 - accuracy: 0.9912 - val_lc
Epoch 8/10
472/472 [==============================] - 1s 3ms/step - loss: 0.0735 - accuracy: 0.9927 - val_lc
Epoch 9/10
472/472 [==============================] - 2s 3ms/step - loss: 0.0618 - accuracy: 0.9936 - val_lc
Epoch 10/10
472/472 [==============================] - 1s 2ms/step - loss: 0.0526 - accuracy: 0.9941 - val_lc
```

```
model2.evaluate(X1_test, y1_test)
```

```
1474/1474 [==============================] - 2s 1ms/step - loss: 0.0489 - accuracy: 0.9939
[0.048905566334724426, 0.9939141869544983]
```

```
fig, ax = plt.subplots()
ax.plot(URL_model2.history["loss"],'r', marker='.', label="Train Loss")
ax.plot(URL_model2.history["val_loss"],'b', marker='.', label="Validation Loss")
ax.legend()
```

```
<matplotlib.legend.Legend at 0x79ac7dc41630>
```

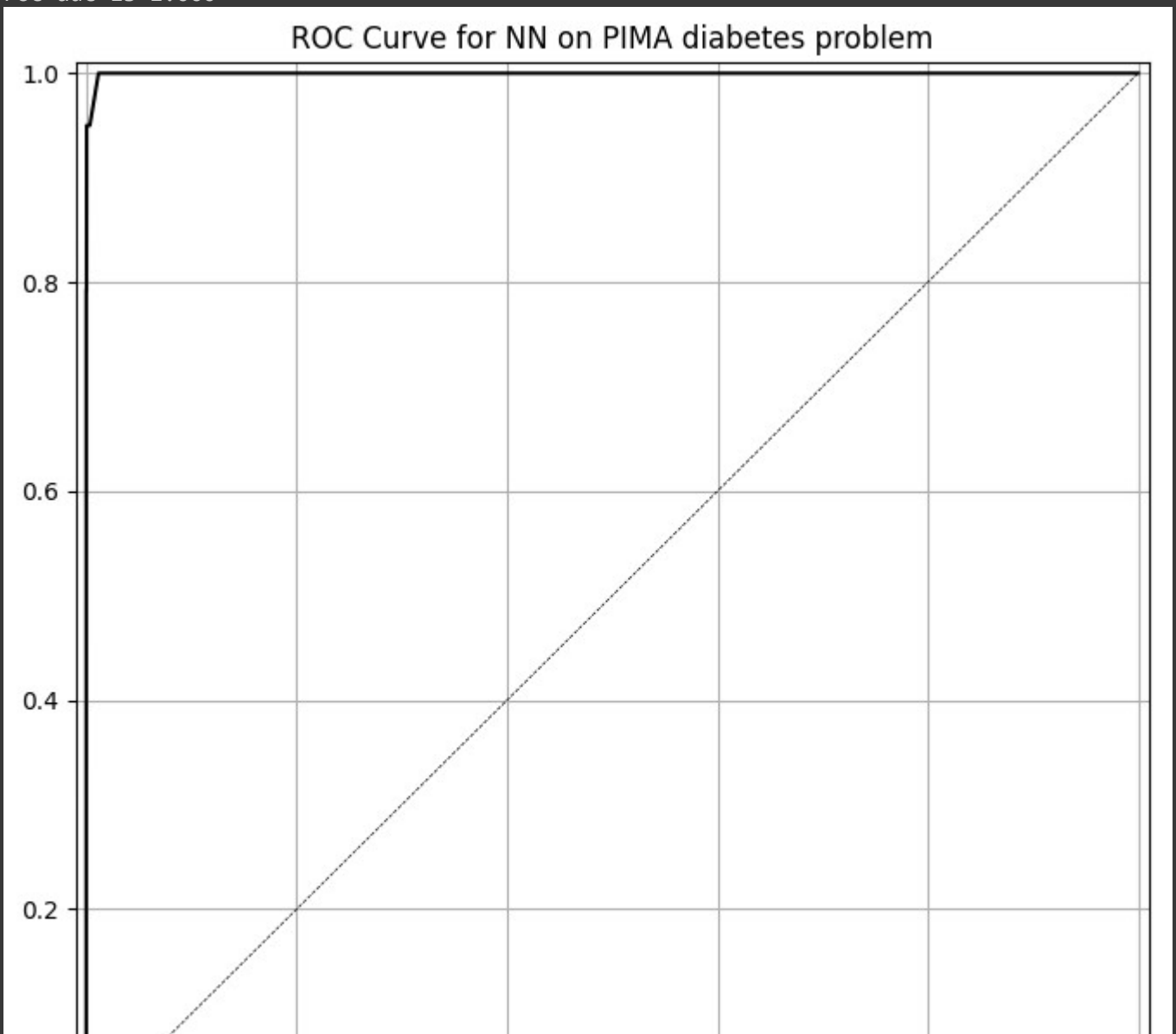


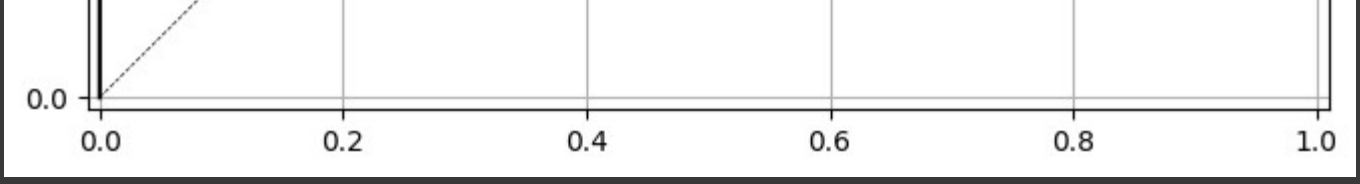

```
1 y_pred_prob_nn_2 = model2.predict(X_test)
```

```
1474/1474 [==============================] - 2s 1ms/step
```

```
1 print('roc-auc is {:.3f}'.format(roc_auc_score(y_test,y_pred_prob_nn_2)))
2
3 plot_roc(y_test, y_pred_prob_nn_2, 'NN')
```

```
roc-auc is 1.000
```





Observation:

- Here, we can see that even if we have only used 3 features with high positive correlation with our target variable, we have a high accuracy and low loss. Our accuracy for the evaluation of our model using the test data is 0.9939 and the loss is 0.0849. Our ROC curve also reached 1.000, indicating that there is a balance between the true positive and true negative predictions of our model.

SUMMARY / CONCLUSION

- In this laboratory quiz, we used a dataset which contains legitimate and phishing website data. The goal of this quiz is to correctly predict if a website is legitimate or if it is a phishing website. The first thing I did is to perform Exploratory Data Analysis. I found our while doing EDA that there are no null value in this dataset and there are in total 56 columns and 235,795 entries in this dataset. I also looked at the distribution of the class values to look for an imbalanced, but I found out that there is no imbalance. Then I did the .describe() method, to look at the distribution of the values and found out that there is a large gap between the 75% and the max value of some attributes. I also did pearson correlation and heatmap to further look into the correlation of the attributes.
- Utilizing the information that I gathered during the EDA, I proceed in pre-processing the data. The first thing I did is to normalize the data by using minmaxscaler and then I also dropped the objects from the dataset. After this I split the data into 60/20/20, 60 for training, 20 for testing and 20 for validation data.
- I trained my first model with 50 input layer nodes, 33 hidden layer nodes, and 1 output layer node. I used relu activation function for the hidden layer while I used sigmoid for the output. My learning rate is 0.001 and I used adam as the optimizer. I used binary_crossentropy for the loss and accuracy for the metric. I used 320 batch size because of the large dataset and I only set 10 epoch. After training, the model has 99% training accuracy and validation accuracy. After evaluating the model, I also got 99% accuracy for the testing dataset, and 0.08 loss for the testing dataset.
- For the second model, I used the same parameters from the first model but I limited the inputs into 3 by using feature selection. The performance is the same, it is performing very well and it has a high training and validation accuracy. After testing it, it also has a high testing accuracy.
- Overall, I think I did my best with regards to EDA and Pre-processing of the dataset. I also did my best in optimizing the model so that the speed and the memory that it takes while training will be lessened.

REFERENCES

[1]A. Kulkarni, F. Batarseh, and D. Chong, "Chapter 5: Foundations of Data Imbalance and Solutions for a Data Democracy." Available: https://arxiv.org/ftp/arxiv/papers/2108/2108.00071.pdf

[2]T. Hasanin, T. M. Khoshgoftaar, J. L. Leevy, and R. A. Bauder, "Severely imbalanced Big Data challenges: investigating data sampling approaches," Journal of Big Data, vol. 6, no. 1, Nov. 2019, doi: https://doi.org/10.1186/s40537-019-0274-4.

[3]"Machine Learning Glossary," Google for Developers. https://developers.google.com/machine-learning/glossary#positive_class (accessed Mar. 23, 2024).

[4]"Machine Learning Glossary," Google for Developers. https://developers.google.com/machine-learning/glossary#negative_class (accessed Mar. 23, 2024).

[5]"Machine Learning Glossary," Google for Developers. https://developers.google.com/machine-learning/glossary#class-imbalanced-dataset (accessed Mar. 23, 2024).

[6]B. Ratner, "The correlation coefficient: Its values range between +1/−1, or do they?," Journal of Targeting, Measurement and Analysis for Marketing, vol. 17, no. 2, pp. 139–142, May 2009, doi: https://doi.org/10.1057/jt.2009.5.

[7]S. Karsoliya, "Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture," 2012. Accessed: Mar. 24, 2024. [Online]. Available: https://ijettjournal.org/volume-3/issue-6/IJETT-V3I6P206.pdf

[8]I. Guyon, "A scaling law for the validation-set training-set size ratio." Accessed: Nov. 05, 2023. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=452e6c05d46e061290fefff8b46d0ff161998677

[9]"Splitting into train, dev and test sets," cs230.stanford.edu. https://cs230.stanford.edu/blog/split/