Configuration Manual

MSc Research Project
Cyber Security

Mamta Sawant
Student ID: X19221134

School of Computing
National College of Ireland

Supervisor: Dr Vanessa Ayala-Rivera

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Mamta Sanjay Sawant
Student ID:	X19221134
Programme:	MSc in Cyber Security **Year:** 2021-22
Module:	Research Project
Lecturer:	Dr Vanessa Ayala-Rivera
Submission Due Date:	16/12/2021
Project Title:	Comparative Analysis of Supervised Machine Learning Models for Phishing Detection
Word Count:	919 **Page Count:** 9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Mamta Sanjay Sawant

Date: 16/12/2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Mamta Sawant
Student ID: X19221134

1 Introduction

The configuration manual focuses on the project's implementation and contribution, namely, comparative analysis of supervised machine learning models for Phishing Detection. This manual also includes information on various hardware and software requirements for the project's successful completion. The primary goal of analyzing and evaluating three supervised Machine learning models that are decision tree, KNN, Logistic Regression is to discover and compare the best phishing detection solution in terms of accuracy and time required to train the model. For this project, the phishing dataset is collected from a cloud-based repository, the dataset as the total number of observations equal to 88647. It consists of data of phishing and legitimate websites.

2 Requirements

To implement the code, a system should have an essential set of tools and settings configured.

2.1 System Requirement

It is important to select a system with hardware specifications that can handle the implementation of Machine learning algorithms. The following are the required System Specifications:

For Windows:

- CPU: Intel i5 5th Gen and above
- RAM: 16GB DDR4 and above
- Storage: 1 HDD

2.2 Software Requirements

- Anaconda Navigator- Jupyter Notebook v6. 4.6
- Python 3.7. 6- Because it is open-source software, it is easily available for download online.
- MS Excel- To analyse the dataset .csv files.

3 Dataset Information

The Phishing website dataset is used for this research. The dataset is available on the online repository under the name dataset_full.csv. It comprises phishing and legitimate instances. The total number of instances is 88647 (Vrbančič, 2020).

Figure 1 shows the license of the dataset i.e., CC BY 4.0 which means the dataset can be shared, copied, or modified as long as the appropriate credits are given.

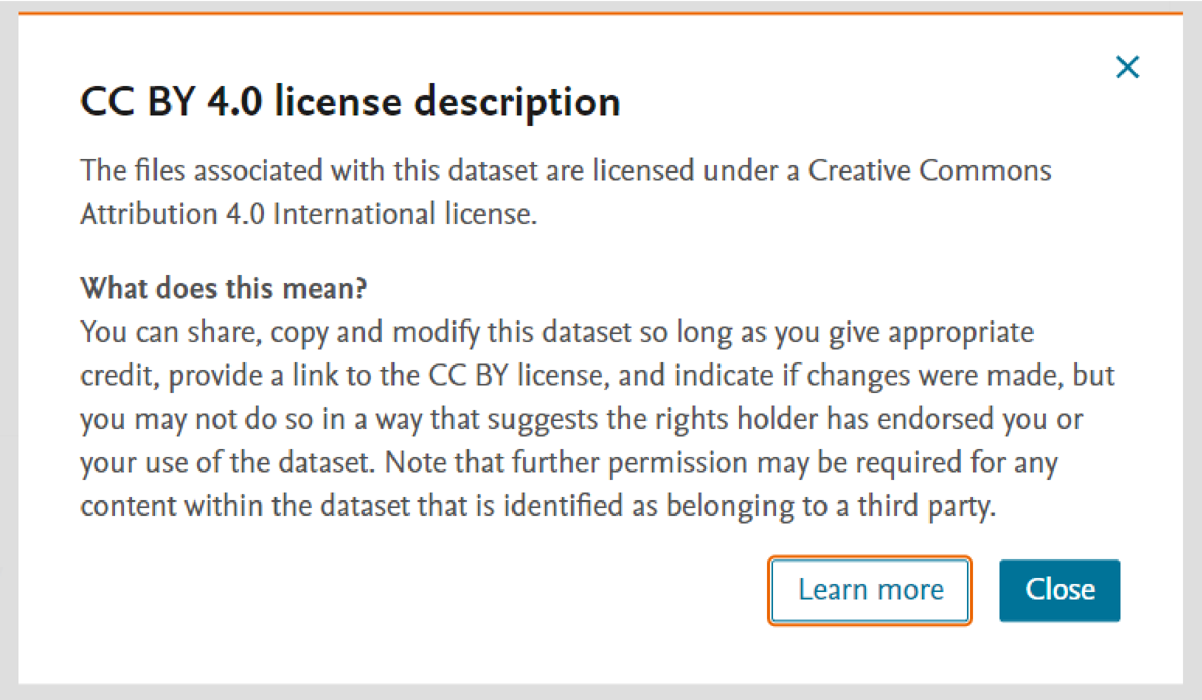
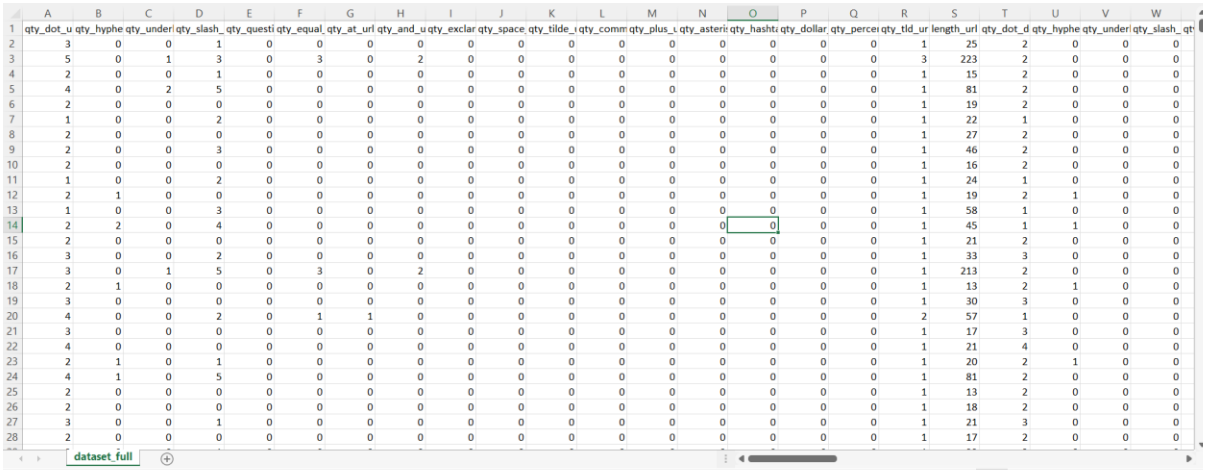


Figure 1: License of the dataset

Figure 2 illustrates the sample content of the dataset_full.csv dataset



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	
1	qty_dot_u	qty_hyphe	qty_underl	qty_slash_	qty_questi	qty_equal_	qty_at_url	qty_and_u	qty_exclar	qty_space_	qty_tilde_	qty_comm	qty_plus_	qty_asteri	qty_hashta	qty_dollar_	qty_percei	qty_tld_ur	length_url	qty_dot_d	qty_hyphe	qty_underl	qty_slash_	qt
2	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	25	2	0	0	0	
3	5	0	1	3	0	3	0	2	0	0	0	0	0	0	0	0	0	3	223	2	0	0	0	
4	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	15	2	0	0	0	
5	4	0	2	5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	81	2	0	0	0	
6	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	19	2	0	0	0	
7	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	22	1	0	0	0	
8	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	27	2	0	0	0	
9	2	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	46	2	0	0	0	
10	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	16	2	0	0	0	
11	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	24	1	0	0	0	
12	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	19	2	1	0	0	
13	1	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	58	1	0	0	0	
14	2	2	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	1	45	1	1	0	0	
15	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	21	2	0	0	0	
16	3	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	33	3	0	0	0	
17	3	0	1	5	0	3	0	2	0	0	0	0	0	0	0	0	0	1	213	2	0	0	0	
18	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	13	2	1	0	0	
19	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	30	3	0	0	0	
20	4	0	0	2	0	1	1	0	0	0	0	0	0	0	0	0	0	2	57	1	0	0	0	
21	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	17	3	0	0	0	
22	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	21	4	0	0	0	
23	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	20	2	1	0	0	
24	4	1	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	81	2	0	0	0	
25	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	13	2	0	0	0	
26	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	18	2	0	0	0	
27	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	21	3	0	0	0	
28	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	17	2	0	0	0	

dataset_full

Figure 2: dataset_full.csv in MS Excel

4 Implementation of Code

4.1 Packages Required for code execution

The model is code using Python 3 language and implemented on the Jupyter notebook. Python code includes below list of packages that are imported:

- Numpy
- Pandas version 1.4.0
- Matplotlib

- Seaborn
- Sklearn
- Train_test_split
- DecisionTreeClassifier
- Sklearn.metrics
- LogisticRegression
- KNeighborsClassifier

4.2 Evaluation of Code:

The entire model is implemented in several sections. Steps involved in the successful execution of code are as follows:

Step 1: Import the dataset_full.csv

```
In [53]: phish_df = pd.read_csv('dataset_full.csv')
         phish_df.dataframeName = 'dataset_full.csv'

         nRow, nCol = phish_df.shape
         print(f'There are {nRow} rows and {nCol} columns')

         There are 88647 rows and 112 columns
```

Figure 3: Importing the dataset

Step 2: Feature selection using correlation matrix

Initially, the feature selection is done on the entire dataset, where the correlation coefficient method is used to select the best possible features. These features are then used to train and test the three supervised machine learning models, and the results are evaluated in terms of accuracy and speed to train the model for phishing detection

```
In [52]: # Correlation matrix
         def plotCorrelationMatrix(df, graphWidth):
             #filename = df.dataframeName
             df = df.dropna('columns') # drop columns with NaN
             df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique values
             if df.shape[1] < 2:
                 print(f'No correlation plots shown: The number of non-NaN or constant columns ({df.shape[1]}) is less than 2')
                 return
             corr = df.corr()
             plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolor='w', edgecolor='k')
             corrMat = plt.matshow(corr, fignum = 1)
             plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
             plt.yticks(range(len(corr.columns)), corr.columns)
             plt.gca().xaxis.tick_bottom()
             plt.colorbar(corrMat)
             plt.title(f'Correlation Matrix for ', fontsize=15)
             plt.show()
```

Fig 4: Correlation matrix function

```
In [54]: plotCorrelationMatrix(phish_df, 19)
```

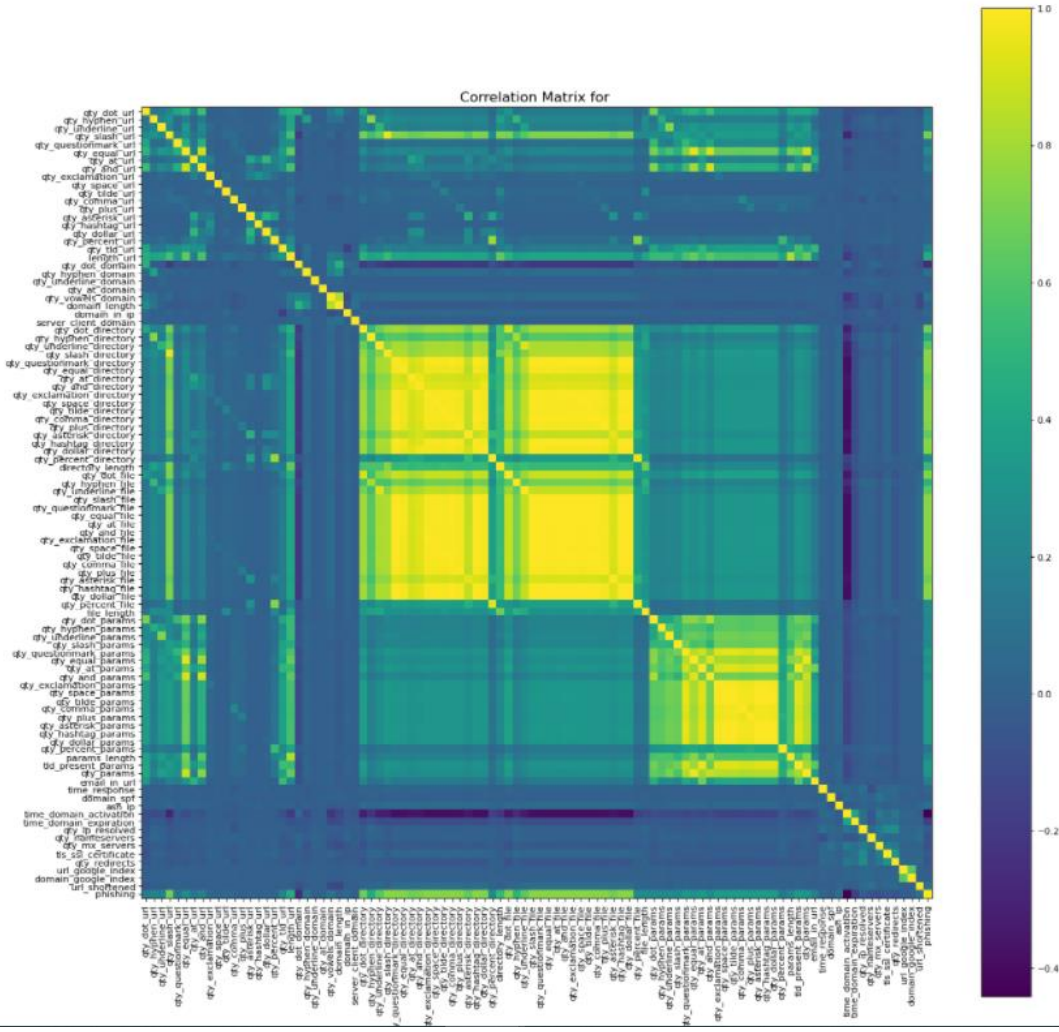


Fig 5: Plotting of Correlation Matrix

Step 3: Check for infinite values, missing values, or NaN(Not a Number) values in the dataset.

```
In [55]: assert isinstance(phish_df, pd.DataFrame)
         phish_df.dropna(inplace=True)

         print(phish_df)
```

Fig 6 : Removing noisy or missing data

Step 4: Implementation of feature selelction to extract highly correlated features using threshold=0.99

```
In [56]: # with the following function we can select highly correlated features
         # it will remove the first feature that is correlated with anything other feature

         def correlation(dataset, threshold):
             col_corr = set()  # Set of all the names of correlated columns
             corr_matrix = dataset.corr()
             for i in range(len(corr_matrix.columns)):
                 for j in range(i):
                     if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                         colname = corr_matrix.columns[i]  # getting the name of column
                         col_corr.add(colname)
             return col_corr
```

```
In [57]: corr_features = correlation(phish_df, 0.99)
         len(set(corr_features))

Out[57]: 21
```

```
In [58]: corr_features

Out[58]: {'qty_and_file',
          'qty_asterisk_params',
          'qty_at_file',
          'qty_comma_directory',
          'qty_comma_file',
          'qty_comma_params',
          'qty_dollar_file',
          'qty_dollar_params',
          'qty_equal_file',
          'qty_exclamation_directory',
          'qty_exclamation_file',
          'qty_hashtag_directory',
          'qty_hashtag_file',
          'qty_hashtag_params',
          'qty_plus_file',
          'qty_questionmark_file',
          'qty_slash_file',
          'qty_space_directory',
          'qty_space_file',
          'qty_tilde_file',
          'qty_tilde_params'}
```

Fig 7: 21 features are extracted from the dataset

Step 5: New dataset with 21 features and 88647 observations is created from the main dataset i.e., dataset_full.csv

```
In [60]: data = pd.read_csv('21Features_80kValues_Dataset.csv')
         data.head()

Out[60]:
```

	qty_exclamation_directory	qty_space_directory	qty_comma_directory	qty_hashtag_directory	qty_slash_file	qty_questionmark_file	qty_equal_file	qty_at_file
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	-1	-1	-1	-1	-1	-1	-1	-1

5 rows × 22 columns

```
In [3]: data.shape

Out[3]: (88647, 22)
```

```
In [4]: assert isinstance(data, pd.DataFrame)
        data.dropna(inplace=True)
        #indices_to_keep = ~data.isin([np.nan, np.inf, -np.inf]).any(1)

        #print(data)
        data.shape

Out[4]: (88647, 22)
```

Fig 8: New dataset with 21 features

Step 6: After data preprocessing and feature selection, data is split into a training set and testing set. X is defined as an independent variable and Y as a target variable.

```
In [61]: x = data.drop('phishing',axis=1).values
         y = data['phishing'].values
```

```
In [62]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=15)
```

Fig 9: Splitting data into 70:30 Ratio for training and Testing the model.

Step 7: Analysis of three supervised machine learning models and evaluation on performance measures.

```
In [64]: from sklearn import tree
         model_tree = tree.DecisionTreeClassifier()
         start=time.time()
         model = model_tree.fit(x_train, y_train)
         elapsed_time=(time.time()-start)
         str(elapsed_time)

Out[64]: '0.061273813247680664'

In [65]: from sklearn.metrics import matthews_corrcoef
         from sklearn.metrics import accuracy_score
         from sklearn.metrics import f1_score
         from sklearn.metrics import precision_score
         from sklearn.metrics import recall_score
         #Test the model using testing data
         predictions = model.predict(x_test)
         from sklearn.metrics import confusion_matrix
         confusion_matrix(y_test,predictions)
         #print(classification_report(y_test,predictions))

Out[65]: array([[13933,  3326],
                [  242,  9094]], dtype=int64)

In [66]: print("f1 score of the Decision tree model is: ",100.0 *f1_score(y_test,predictions,average='weighted'))
         print("Accuracy score of the Decision tree model is: ",100.0 *accuracy_score(y_test,predictions))
         print("Precision score of the Decision tree model is: ",100.0 *precision_score(y_test,predictions))
         print("Recall score of the Decision tree model is: ",100.0 *recall_score(y_test,predictions))

         f1 score of the Decision tree model is:  86.87670731993857
         Accuracy score of the Decision tree model is:  86.58394435044181
         Precision score of the Decision tree model is:  73.22061191626409
         Recall score of the Decision tree model is:  97.40788346186804
```

Fig 10 : Decision tree model

```
In [26]: from sklearn.linear_model import LogisticRegression

         #create logistic regression object
         Classifier=LogisticRegression(random_state= 0, multi_class='multinomial' , solver='newton-cg')

         start=time.time()
         #Train the model using training data
         Classifier.fit(x_train,y_train)
         elapsed_time=(time.time()-start)
         str(elapsed_time)

Out[26]: '1.8445639610290527'

In [27]: predictions = Classifier.predict(x_test)
         predictions
         from sklearn.metrics import confusion_matrix
         confusion_matrix(y_test,predictions)

Out[27]: array([[18762,  4445],
                [  306, 11946]], dtype=int64)

In [28]: print(" f1 score of Logistic Regression model:",100.0 *f1_score(y_test,predictions,average='weighted'))
         print("Accuracy score of Logistic Regression model: ",100.0 *accuracy_score(y_test,predictions))
         print("Precision score of the Logistic Regression model is: ",100.0 *precision_score(y_test,predictions))
         print("Recall score of the Logistic Regression is: ",100.0 *recall_score(y_test,predictions))

          f1 score of Logistic Regression model: 0.86913590007293
         Accuracy score of Logistic Regression model:  86.6014270001974
         Precision score of the Logistic Regression model is:  72.88145933744129
         Recall score of the Logistic Regression is:  97.50244857982369
```

Fig11: Logistic Regression model

```
In [29]: from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()
         xScaler = scaler.fit_transform(x)

In [30]: x_train, x_test, y_train, y_test = train_test_split(xScaler,y, test_size = 0.4)

In [31]: from sklearn.neighbors import KNeighborsClassifier
         from sklearn import metrics

In [32]: k =5
         knn = KNeighborsClassifier(n_neighbors=k)
         start=time.time()
         knn.fit(x_train, y_train)
         y_pred = knn.predict(x_test)
         print(metrics.accuracy_score(y_test, y_pred)) #accuracy score on the train data
         elapsed_time=(time.time()-start)
         str(elapsed_time)

         0.8640965622267971

Out[32]: '37.66502785682678'

In [33]: from sklearn.model_selection import cross_val_predict, cross_val_score
         score = cross_val_score(knn, xScaler, y, cv = 8)
         print(score)

         [0.86210631 0.86255753 0.86427218 0.72728093 0.86734049 0.86508438
          0.87293566 0.86444043]

In [34]: y_pred = cross_val_predict(knn, xScaler, y, cv = 10)
         conf_mat = metrics.confusion_matrix(y , y_pred)
         print(conf_mat)

         [[46774 11226]
          [  747 29900]]

In [40]: f1 = 100.0 *metrics.f1_score(y,y_pred,average="weighted")
         print("f1 score of KNN model is:",f1)

         acc = 100.0 *metrics.accuracy_score(y, y_pred)
         print("Accuracy score of KNN model is:",acc)

         precision = 100.0 *metrics.precision_score(y,y_pred)
         print("Precision score of KNN model is:",precision)

         recall = 100.0 *metrics.recall_score(y,y_pred)
         print("recall score of KNN model is:",recall)

         f1 score of KNN model is: 86.80895914186695
         Accuracy score of KNN model is: 86.49362076550814
         Precision score of KNN model is: 72.70339930943929
         recall score of KNN model is: 97.56256729859366
```

Fig 12: K- Nearest Neighbors (K-NN model)

The results of all the models are recorded after they have been successfully executed. When compared to the other two models, the Logistic Regression model has higher accuracy and the decision tree takes less time to train the model, so it is faster.

References

Vrbančič, G., 2020. Phishing Websites Dataset 1. https://doi.org/10.17632/72ptz43s9v.1