Phishing URL Detection

The Internet has become an indispensable part of our life, However, It also has provided opportunities to anonymously perform malicious activities like Phishing. Phishers try to deceive their victims by social engineering or creating mockup websites to steal information such as account ID, username, password from individuals and organizations. Although many methods have been proposed to detect phishing websites, Phishers have evolved their methods to escape from these detection methods. One of the most successful methods for detecting these malicious activities is Machine Learning. This is because most Phishing attacks have some common characteristics which can be identified by machine learning methods.

The steps demonstrated in this notebook are:

1. Loading the data
2. Familiarizing with data & EDA
3. Visualizing the data
4. Splitting the data
5. Training the data
6. Comparision of Model
7. Conclusion

```
from google.colab import drive
drive.mount('/content/drive')
```

```
#importing required libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn import metrics
import warnings
warnings.filterwarnings('ignore')
```

1. Loading Data:

The dataset is borrowed from Kaggle, https://www.kaggle.com/eswarchandt/phishing-website-detector .

A collection of website URLs for 11000+ websites. Each sample has 30 website parameters and a class label identifying it as a phishing website or not (1 or -1).

The overview of this dataset is, it has 11054 samples with 32 features. Download the dataset from the link provided.

[ ] ↳ *1 cell hidden*

2. Familiarizing with Data & EDA:

In this step, few dataframe methods are used to look into the data and its features.

[ ] ↳ *7 cells hidden*

3. Visualizing the data:

Few plots and graphs are displayed to find how the data is distributed and the how features are related to each other.

[ ] ↳ *3 cells hidden*

4. Splitting the Data:

The data is split into train & test sets, 80-20 split.

[ ] ↳ *2 cells hidden*

5. Model Building & Training:

Supervised machine learning is one of the most commonly used and successful types of machine learning. Supervised learning is used whenever we want to predict a certain outcome/label from a given set of features, and we have examples of features-label pairs. We build a machine learning model from these features-label pairs, which comprise our training set. Our goal is to make accurate predictions for new, never-before-seen data.

There are two major types of supervised machine learning problems, called classification and regression. Our data set comes under regression problem, as the prediction of suicide rate is a continuous number, or a floating-point number in programming terms. The supervised machine learning models (regression) considered to train the dataset in this notebook are:

1. Logistic Regression

2. k-Nearest Neighbors
3. Support Vector Clasifier
4. Naive Bayes
5. Decision Tree
6. Random Forest
7. Gradient Boosting
8. Catboost
9. Xgboost
10. Multilayer Perceptrons

The metrics considered to evaluate the model performance are Accuracy & F1 score.

[ ] ↳ *1 cell hidden*

5.1. Logistic Regression

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

[ ] ↳ *5 cells hidden*

5.2. K-Nearest Neighbors : Classifier

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

[ ] ↳ *6 cells hidden*

5.3. Support Vector Machine : Classifier

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.

[ ] ↳ *5 cells hidden*

5.4. Naive Bayes : Classifier

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.It is mainly used in text, image classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

[ ] ↳ *5 cells hidden*

5.5. Decision Trees : Classifier

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

[ ] ↳ *6 cells hidden*

5.6. Random Forest : Classifier

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

[ ] ↳ *6 cells hidden*

5.7.Gradient Boosting Classifier

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting. Boosting algorithms play a crucial role in dealing with bias variance trade-off. Unlike bagging algorithms, which only controls for high variance in a model, boosting controls both the aspects (bias & variance), and is considered to be more effective.

[ ] ↳ *7 cells hidden*

5.8. CatBoost Classifier

CatBoost is a recently open-sourced machine learning algorithm from Yandex. It can easily integrate with deep learning frameworks like Google's TensorFlow and Apple's Core ML. It can work with diverse data types to help solve a wide range of problems that businesses face today.

[ ] ↳ *8 cells hidden*

5.9. XGBoost Classifier

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance that is dominative competitive machine learning. In this post you will discover how you can install and create your first XGBoost model in Python

[ ] ↳ *4 cells hidden*

5.10. Multi-layer Perceptron classifier

MLPClassifier stands for Multi-layer Perceptron classifier which in the name itself connects to a Neural Network. Unlike other classification algorithms such as Support Vectors or Naive Bayes Classifier, MLPClassifier relies on an underlying Neural Network to perform the task of classification.

[ ] ↳ *4 cells hidden*

6. Comparision of Models

To compare the models performance, a dataframe is created. The columns of this dataframe are the lists created to store the results of the model.

[ ] ↳ *4 cells hidden*

Storing Best Model

```
#  XGBoost Classifier Model
from xgboost import XGBClassifier

# instantiate the model
gbc = GradientBoostingClassifier(max_depth=4,learning_rate=0.7)

# fit the model
gbc.fit(X_train,y_train)
```

```
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

```
x= data.iloc[:,0:4].values
y= data.iloc[:,4:5].values
```

```
x.shape
```

```
(11054, 4)
```

```
y.shape
```

```
(11054, 1)
```

```
from sklearn.preprocessing import OneHotEncoder
one = OneHotEncoder()
z= one.fit_transform(x[:,3:4]).toarray()
```

```
z
```

```
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       ...,
       [0., 1.],
       [0., 1.],
       [0., 1.]])
```

```
x= np.delete(x,3,axis = 1)
x= np.concatenate((z,x),axis=1)
```

```
x.shape
```

```
(11054, 5)
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2, random_state=0)
```

```
x_train.shape
```

```
(8843, 5)
```

```
x_test.shape
```

```
(2211, 5)
```

```
#model building
from sklearn.linear_model import LinearRegression
mlr = LinearRegression()
mlr.fit(x_train,y_train)
```

```
LinearRegression()
```

```
ypred= mlr.predict(x_test)
ypred
```

```
array([[ 0.9866333 ],
       [ 0.9866333 ],
       [ 0.98516846],
       ...,
       [ 0.9866333 ],
       [-0.73199463],
       [ 0.98516846]])
```

```
from sklearn.metrics import r2_score
acc= r2_score(y_test,ypred)
acc
```

```
0.7374735045878138
```

```
import pickle
pickle.dump(mlr,open("model.pkl","wb"))
```

```
pwd
```

```
'/content'
```

IBM Cloud Deployment (Watson)

```
!pip install watson-machine-learning-client
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: watson-machine-learning-client in /usr/local/lib/python3
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from wa
Requirement already satisfied: lomond in /usr/local/lib/python3.7/dist-packages (from wa
Requirement already satisfied: ibm-cos-sdk in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from wats
Requirement already satisfied: boto3 in /usr/local/lib/python3.7/dist-packages (from wat
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from w
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from w
Requirement already satisfied: tabulate in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: jmespath<2.0.0,>=0.7.1 in /usr/local/lib/python3.7/dist-p
```

```
Requirement already satisfied: botocore<1.30.0,>=1.29.10 in /usr/local/lib/python3.7/dis
Requirement already satisfied: s3transfer<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.7/d
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.7.0 in /usr/local/lib/python3.7
Requirement already satisfied: ibm-cos-sdk-core==2.7.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: docutils<0.16,>=0.10 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/python3.7/dist
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (
```

```
!pip install ibm_watson_machine_learning
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
Requirement already satisfied: ibm_watson_machine_learning in /usr/local/lib/python3.7/d
Requirement already satisfied: lomond in /usr/local/lib/python3.7/dist-packages (from ib
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: ibm-cos-sdk==2.7.* in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from i
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from i
Requirement already satisfied: tabulate in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.7.0 in /usr/local/lib/python3.7
Requirement already satisfied: ibm-cos-sdk-core==2.7.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: docutils<0.16,>=0.10 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /usr/local/lib/python3.7/d
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/python3.7/dist
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist
```

```
from ibm_watson_machine_learning import APIClient
import json
import numpy as np
```

```
pwd
```

```
'/content'
```

Authenticate and Set Space

```
from ibm_watson_machine_learning import APIClient

wml_credentials = {
    "url":"https://us-south.ml.cloud.ibm.com",
    "apikey":"evDfXkMpOENEe-kz_urpPdF2FAabTik0rWBNPcDg4iay"
}
```

```
wml_client = APIClient(wml_credentials)
wml_client.spaces.list()
```

```
Python 3.7 and 3.8 frameworks are deprecated and will be removed in a future release. Us
Note: 'limit' is not provided. Only first 50 records will be displayed if the number of
------------------------------------  ----------------------  ------------------------
ID                                    NAME                    CREATED
9dc94fb2-eeb3-4e7a-b647-38527ecc35c0  web phishing detection  2022-11-16T08:44:25.630Z
------------------------------------  ----------------------  ------------------------
```

```
SPACE_ID="9dc94fb2-eeb3-4e7a-b647-38527ecc35c0"
```

```
wml_client.set.default_space(SPACE_ID)
```

```
'SUCCESS'
```

```
wml_client.software_specifications.list()
```

```
------------------------------  ------------------------------------  ----
NAME                            ASSET_ID                              TYPE
default_py3.6                   0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
kernel-spark3.2-scala2.12       020d69ce-7ac1-5e68-ac1a-31189867356a  base
pytorch-onnx_1.3-py3.7-edt      069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6         09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12      09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
pytorch-onnx_rt22.1-py3.9       0b848dd4-e681-5599-be41-b5f6fccc6471  base
ai-function_0.1-py3.6           0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
shiny-r3.6                      0e6e79df-875e-4f24-8ae9-62dcc2148306  base
tensorflow_2.4-py3.7-horovod    1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6               10ac12d6-6b30-4ccd-8392-3e922c096a92  base
tensorflow_1.15-py3.6-ddl       111e41b3-de2d-5422-a4d6-bf776828c4b7  base
autoai-kb_rt22.2-py3.10         125b6d9a-5b1f-5e8d-972a-b251688ccf40  base
runtime-22.1-py3.9              12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
scikit-learn_0.22-py3.6         154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                    1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
pytorch-onnx_1.3-py3.6          1bc6029a-cc97-56da-b8e0-39c3880dbbe7  base
kernel-spark3.3-r3.6            1c9e5454-f216-59dd-a20e-474a5cdf5988  base
pytorch-onnx_rt22.1-py3.9-edt   1d362186-7ad5-5b59-8b6c-9d0880bde37f  base
tensorflow_2.1-py3.6            1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base
spark-mllib_3.2                 20047f72-0a98-58c7-9ff5-a77b012eb8f5  base
tensorflow_2.4-py3.8-horovod    217c16f6-178f-56bf-824a-b19f20564c49  base
runtime-22.1-py3.9-cuda         26215f05-08c3-5a41-a1b0-da66306ce658  base
```

```
do_py3.8                      295addb5-9ef9-547e-9bf4-92ae3563e720  base
autoai-ts_3.8-py3.8           2aa0c932-798f-5ae9-abd6-15e0c2402fb5  base
tensorflow_1.15-py3.6         2b73a275-7cbf-420b-a912-eae7f436e0bc  base
kernel-spark3.3-py3.9         2b7961e2-e3b1-5a8c-a491-482c8368839a  base
pytorch_1.2-py3.6             2c8ef57d-2687-4b7d-acce-01f94976dac1  base
spark-mllib_2.3               2e51f700-bca0-4b0d-88dc-5c6791338875  base
pytorch-onnx_1.1-py3.6-edt    32983cea-3f32-4400-8965-dde874a8d67e  base
spark-mllib_3.0-py37          36507ebe-8770-55ba-ab2a-eafe787600e9  base
spark-mllib_2.4               390d21f8-e58b-4fac-9c55-d7ceda621326  base
autoai-ts_rt22.2-py3.10       396b2e83-0953-5b86-9a55-7ce1628a406f  base
xgboost_0.82-py3.6            39e31acd-5f30-41dc-ae44-60233c80306e  base
pytorch-onnx_1.2-py3.6-edt    40589d0e-7019-4e28-8daa-fb03b6f4fe12  base
pytorch-onnx_rt22.2-py3.10    40e73f55-783a-5535-b3fa-0c8b94291431  base
default_r36py38               41c247d3-45f8-5a71-b065-8580229facf0  base
autoai-ts_rt22.1-py3.9        4269d26e-07ba-5d40-8f66-2d495b0c71f7  base
autoai-obm_3.0                42b92e18-d9ab-567f-988a-4240ba1ed5f7  base
pmml-3.0_4.3                  493bcb95-16f1-5bc5-bee8-81b8af80e9c7  base
spark-mllib_2.4-r_3.6         49403dff-92e9-4c87-a3d7-a42d0021c095  base
xgboost_0.90-py3.6            4ff8d6c2-1343-4c18-85e1-689c965304d3  base
pytorch-onnx_1.1-py3.6        50f95b2a-bc16-43bb-bc94-b0bed208c60b  base
autoai-ts_3.9-py3.8           52c57136-80fa-572e-8728-a5e7cbb42cde  base
spark-mllib_2.4-scala_2.11    55a70f99-7320-4be5-9fb9-9edb5a443af5  base
spark-mllib_3.0               5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9  base
autoai-obm_2.0                5c2e37fa-80b8-5e77-840f-d912469614ee  base
spss-modeler_18.1             5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b  base
cuda-py3.8                    5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e  base
autoai-kb_3.1-py3.7           632d4b22-10aa-5180-88f0-f52dfb6444d7  base
pytorch-onnx_1.7-py3.8        634d3cdc-b562-5bf9-a2d4-ea90a478456b  base
----------------------------  ------------------------------------  ----
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
```

Save and Deploy the model

```
import sklearn
sklearn.__version__
```

```
'1.0.2'
```

```
MODEL_NAME = 'web phishing detection'
DEPLOYMENT_NAME = 'web phishing detection'
DEMO_MODEL = mlr
```

```
# Set Python Version
software_spec_uid = wml_client.software_specifications.get_id_by_name('runtime-22.1-py3.9')
software_spec_uid
```

```
'12b83a17-24d8-5082-900f-0ab31fbfd3cb'
```

```
# Setup model meta
model_props = {
    wml_client.repository.ModelMetaNames.NAME: 'web phishing detection',
    wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_1.0',
    wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}

#Save model
model_details = wml_client.repository.store_model(
    model=DEMO_MODEL,
    meta_props=model_props,
    training_data=x_train,
    training_target=y_train
)


model_details
```

```
 {'entity': {'hybrid_pipeline_software_specs': [],
   'label_column': 'l0',
   'schemas': {'input': [{'fields': [{'name': 'f0', 'type': 'float'},
       {'name': 'f1', 'type': 'float'},
       {'name': 'f2', 'type': 'float'},
       {'name': 'f3', 'type': 'float'},
       {'name': 'f4', 'type': 'float'}],
      'id': '1',
      'type': 'struct'}],
    'output': []},
   'software_spec': {'id': '12b83a17-24d8-5082-900f-0ab31fbfd3cb',
    'name': 'runtime-22.1-py3.9'},
   'type': 'scikit-learn_1.0'},
  'metadata': {'created_at': '2022-11-16T20:41:44.124Z',
   'id': 'c5946f47-7b64-4b1c-9304-db19d1bf4ac1',
   'modified_at': '2022-11-16T20:41:48.309Z',
   'name': 'web phishing detection',
   'owner': 'IBMid-663003YX57',
   'resource_key': '26c7c4b6-587b-4dc0-aff1-207b085e34e6',
   'space_id': '9dc94fb2-eeb3-4e7a-b647-38527ecc35c0'},
  'system': {'warnings': []}}
```

```
model_id = wml_client.repository.get_model_id(model_details)


model_id
```

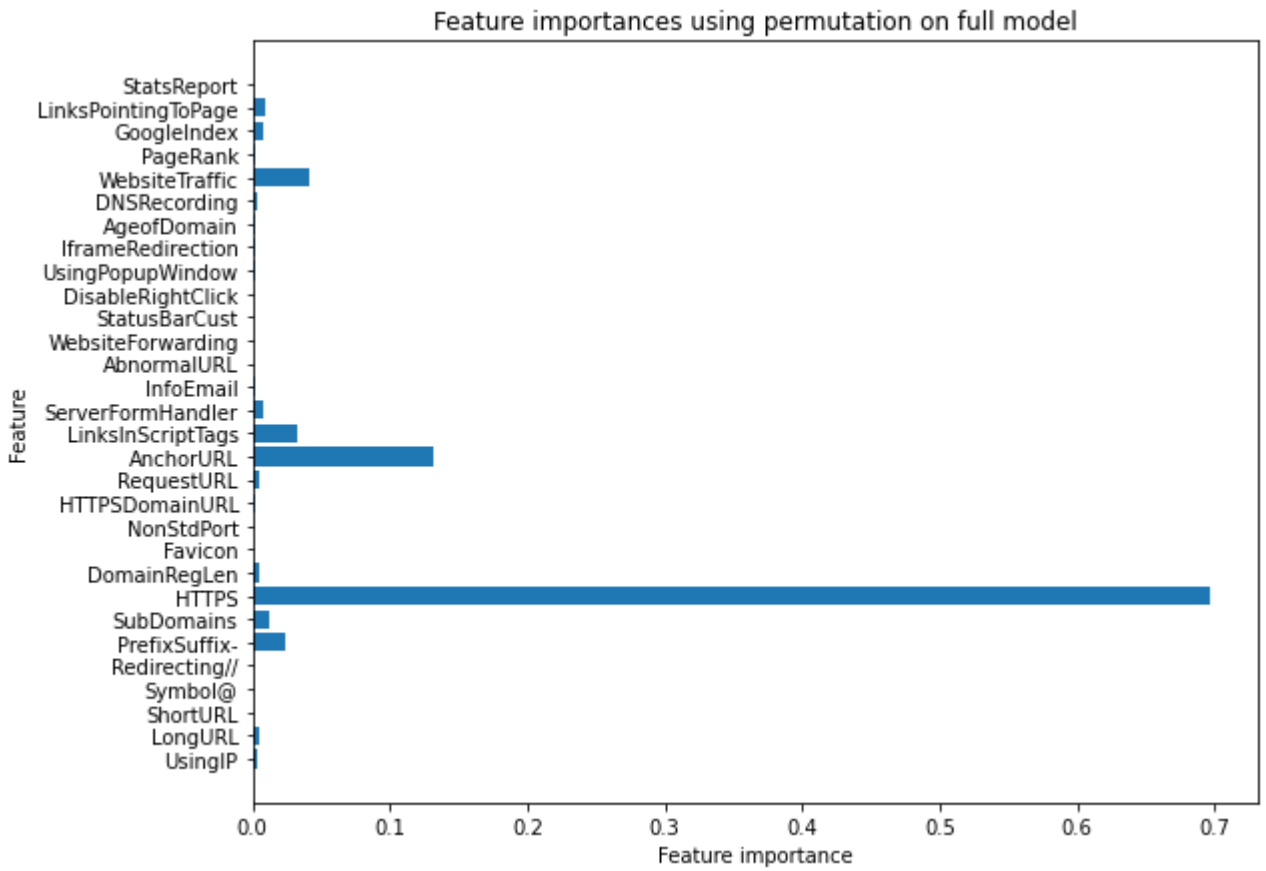
```
 'c5946f47-7b64-4b1c-9304-db19d1bf4ac1'
```

```
# Set meta
deployment_props = {
    wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
    wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
}
```

```
wml_client.repository.download(model_id,"web phishing detection")
```

```
Successfully saved model content to file: 'web phishing detection'
'/content/web phishing detection'
```

```
#checking the feature improtance in the model
plt.figure(figsize=(9,7))
n_features = X_train.shape[1]
plt.barh(range(n_features), gbc.feature_importances_, align='center')
plt.yticks(np.arange(n_features), X_train.columns)
plt.title("Feature importances using permutation on full model")
plt.xlabel("Feature importance")
plt.ylabel("Feature")
plt.show()
```



7. Conclusion

1. The final take away form this project is to explore various machine learning models, perform Exploratory Data Analysis on phishing dataset and understanding their features.
2. Creating this notebook helped me to learn a lot about the features affecting the models to detect whether URL is safe or not, also I came to know how to tuned model and how they

affect the model performance.

3. The final conclusion on the Phishing dataset is that the some feature like "HTTTPS", "AnchorURL", "WebsiteTraffic" have more importance to classify URL is phishing URL or not.
4. Gradient Boosting Classifier currectly classify URL upto 97.4% respective classes and hence reduces the chance of malicious attachments.