An investigation of social engineering security threat using machine learning classification algorithm

- Mona Khalid Alhafi

أكـاديـمـيـة سـدايا
SDAIA Academy

SDAIA
الهيئة السعودية للبيانات
والذكاء الاصطناعي
Saudi Data & AI Authority

Outline

- Project Definition and Objective
- Dataset & Data Analysis
  - Explanation
  - Cleaning
  - Visualizing
  - Extracting Features
- Machine Learning Algorithm & Results
  - Linear & Logistic Regression & KNN
  - Decision Tree & Random Forest
- Conclusions

Definition and Objective

Social Engineering is the act of manipulating a person to get access to confidential information

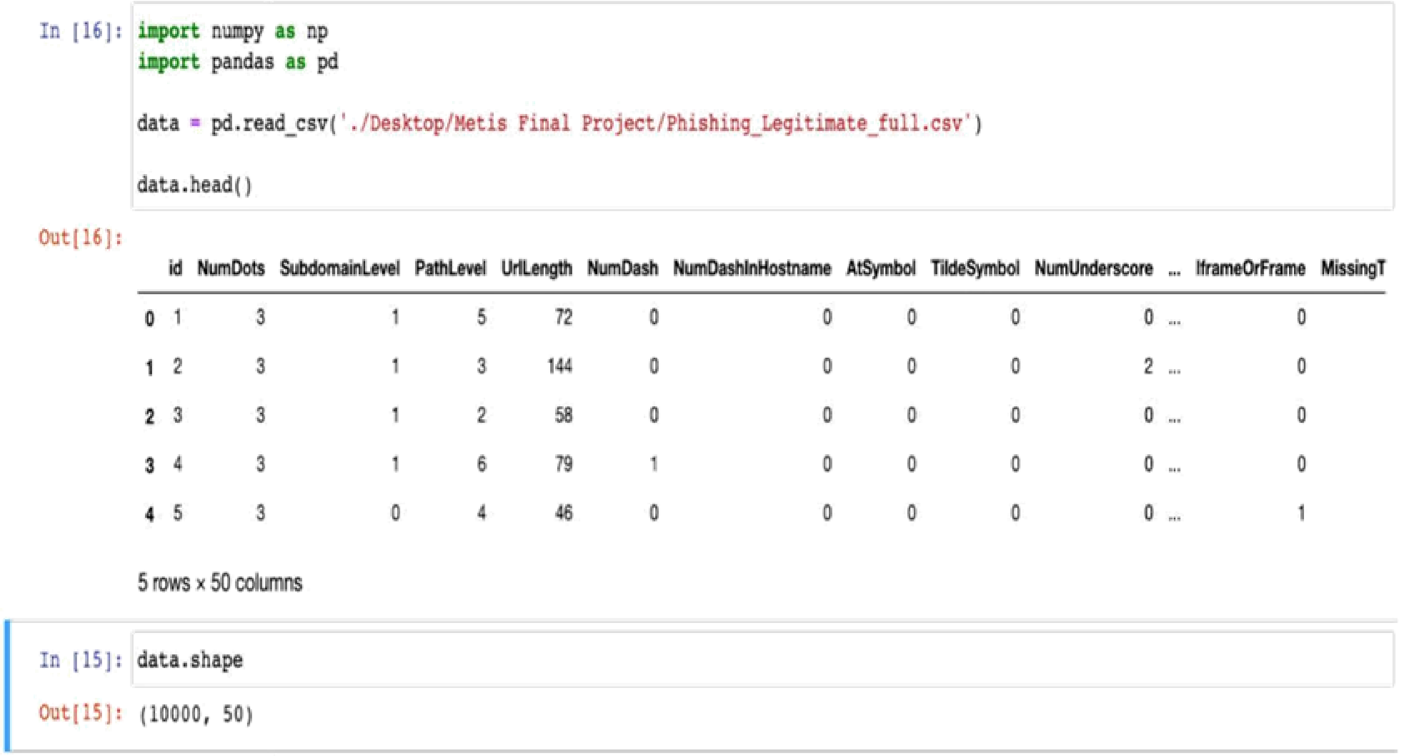
Objectives

- Analyze dataset and visualize data for better understanding.
- Develop a machine learning algorithm to predict either an email is phishing or benign.

Dataset

Explanation

- 10,000x50 in shape
- Balanced output
- Numerical & Binary Classification



```
In [16]: import numpy as np
         import pandas as pd

         data = pd.read_csv('./Desktop/Metis Final Project/Phishing_Legitimate_full.csv')

         data.head()
Out[16]:
```

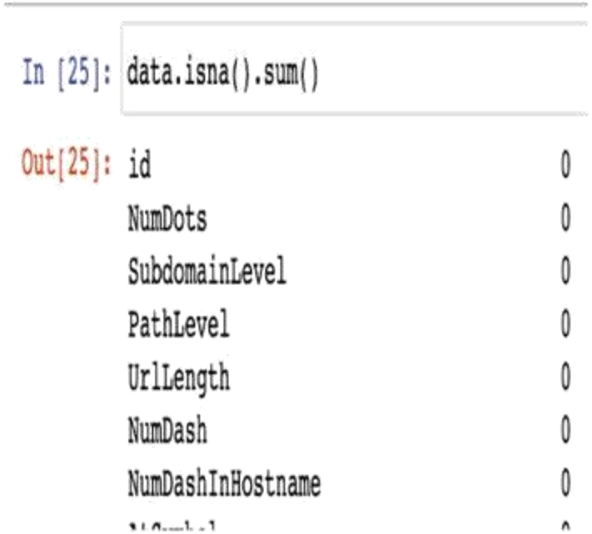
	id	NumDots	SubdomainLevel	PathLevel	UrlLength	NumDash	NumDashInHostname	AtSymbol	TildeSymbol	NumUnderscore	...	IframeOrFrame	MissingT
0	1	3	1	5	72	0	0	0	0	0	...	0	
1	2	3	1	3	144	0	0	0	0	2	...	0	
2	3	3	1	2	58	0	0	0	0	0	...	0	
3	4	3	1	6	79	1	0	0	0	0	...	0	
4	5	3	0	4	46	0	0	0	0	0	...	1	

5 rows × 50 columns

```
In [15]: data.shape
Out[15]: (10000, 50)
```

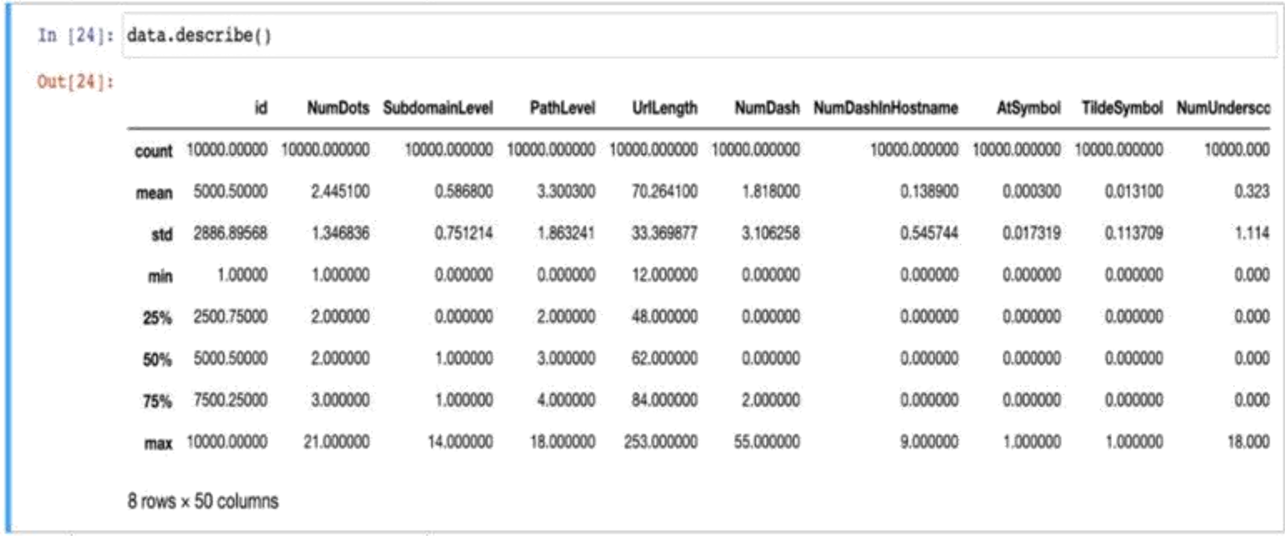
Data Analysis

CHECKING NULL in DATASET



```
In [25]: data.isna().sum()

Out[25]: id                   0
         NumDots              0
         SubdomainLevel       0
         PathLevel            0
         UrlLength            0
         NumDash              0
         NumDashInHostname    0
```

DESCRIBING DATA



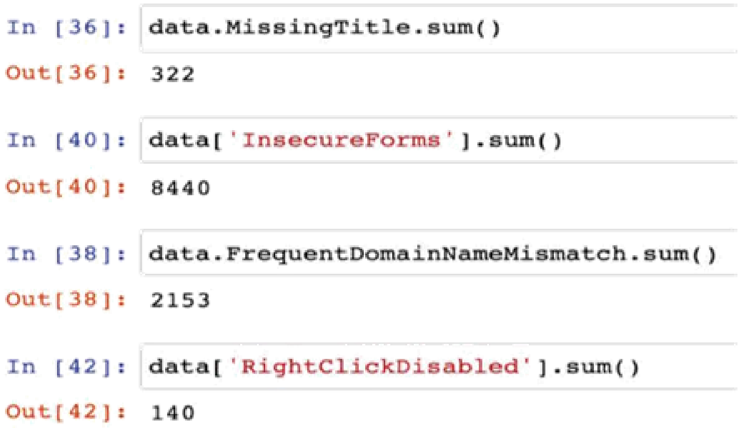
```
In [24]: data.describe()

Out[24]:
```

	id	NumDots	SubdomainLevel	PathLevel	UrlLength	NumDash	NumDashInHostname	AtSymbol	TildeSymbol	NumUndersco
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000
mean	5000.50000	2.445100	0.586800	3.300300	70.264100	1.818000	0.138900	0.000300	0.013100	0.323
std	2886.89568	1.346836	0.751214	1.863241	33.369877	3.106258	0.545744	0.017319	0.113709	1.114
min	1.00000	1.000000	0.000000	0.000000	12.000000	0.000000	0.000000	0.000000	0.000000	0.000
25%	2500.75000	2.000000	0.000000	2.000000	48.000000	0.000000	0.000000	0.000000	0.000000	0.000
50%	5000.50000	2.000000	1.000000	3.000000	62.000000	0.000000	0.000000	0.000000	0.000000	0.000
75%	7500.25000	3.000000	1.000000	4.000000	84.000000	2.000000	0.000000	0.000000	0.000000	0.000
max	10000.00000	21.000000	14.000000	18.000000	253.000000	55.000000	9.000000	1.000000	1.000000	18.000

8 rows × 50 columns

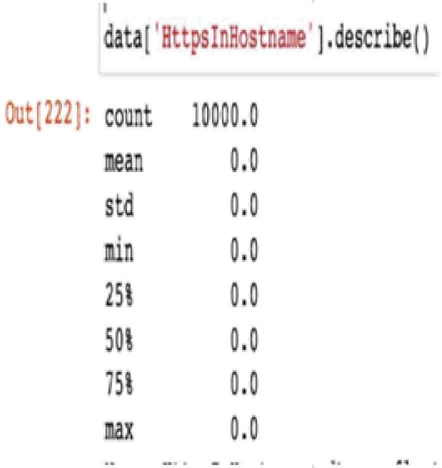
Data Analysis

GETTING INSIGHT ABOUT THE DATA

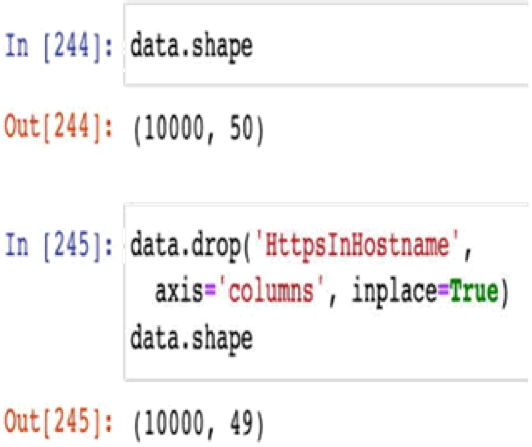


```
In [36]: data.MissingTitle.sum()
Out[36]: 322

In [40]: data['InsecureForms'].sum()
Out[40]: 8440

In [38]: data.FrequentDomainNameMismatch.sum()
Out[38]: 2153

In [42]: data['RightClickDisabled'].sum()
Out[42]: 140
```

CLEANING DATA



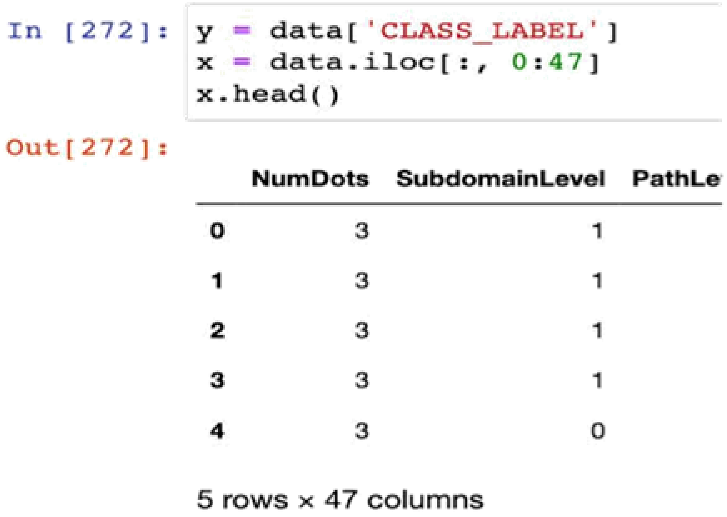
```
          data['HttpsInHostname'].describe()

Out[222]: count    10000.0
          mean         0.0
          std          0.0
          min          0.0
          25%          0.0
          50%          0.0
          75%          0.0
          max          0.0
```



```
In [244]: data.shape

Out[244]: (10000, 50)

In [245]: data.drop('HttpsInHostname',
            axis='columns', inplace=True)
          data.shape

Out[245]: (10000, 49)
```

Data Analysis

EXCTRACT FEATURES AND OUTPUT

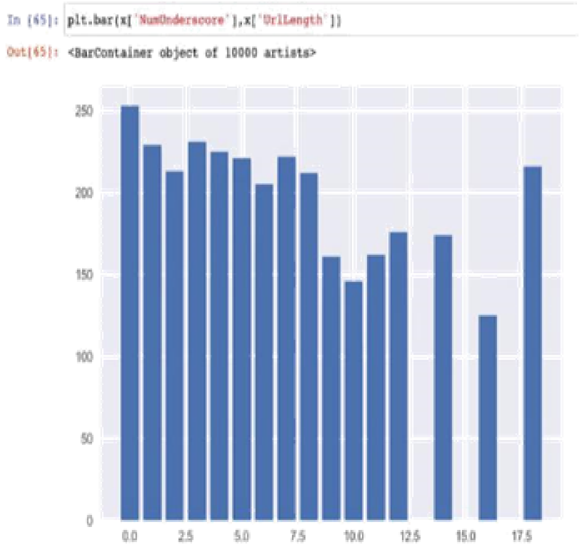


```
In [272]: y = data['CLASS_LABEL']
          x = data.iloc[:, 0:47]
          x.head()
Out[272]:
```

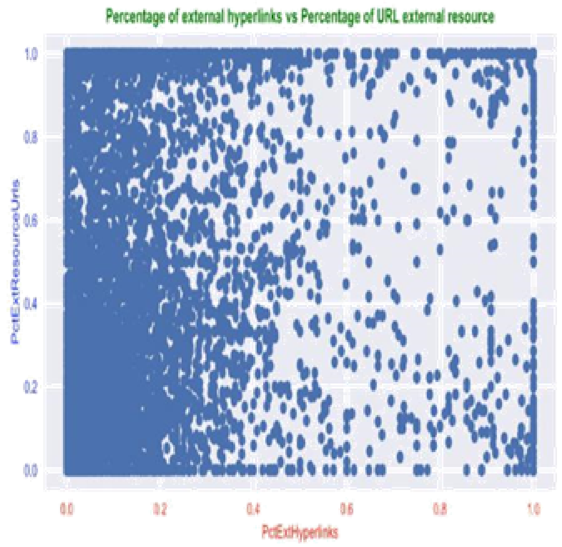
	NumDots	SubdomainLevel	PathLe
0	3	1	
1	3	1	
2	3	1	
3	3	1	
4	3	0	

5 rows × 47 columns

VISUALIZING

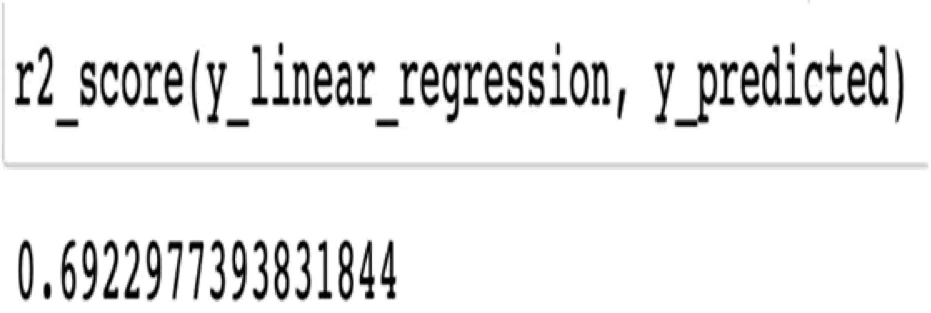


```
In [65]: plt.bar(x['NumUnderscore'],x['UrlLength'])
Out[65]: <BarContainer object of 10000 artists>
```



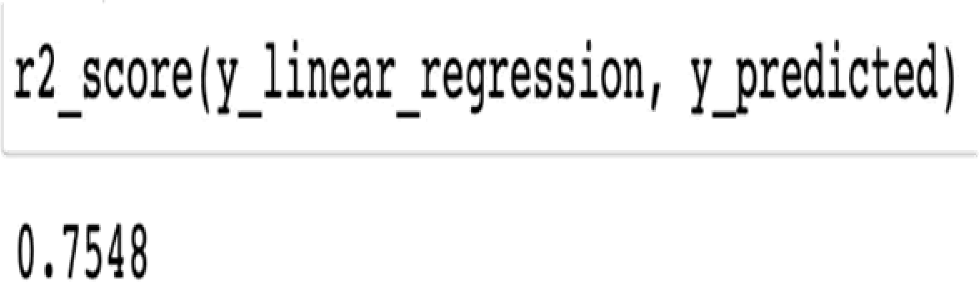
Models

Linear Regression



```
r2_score(y_linear_regression, y_predicted)
```

```
0.6922977393831844
```

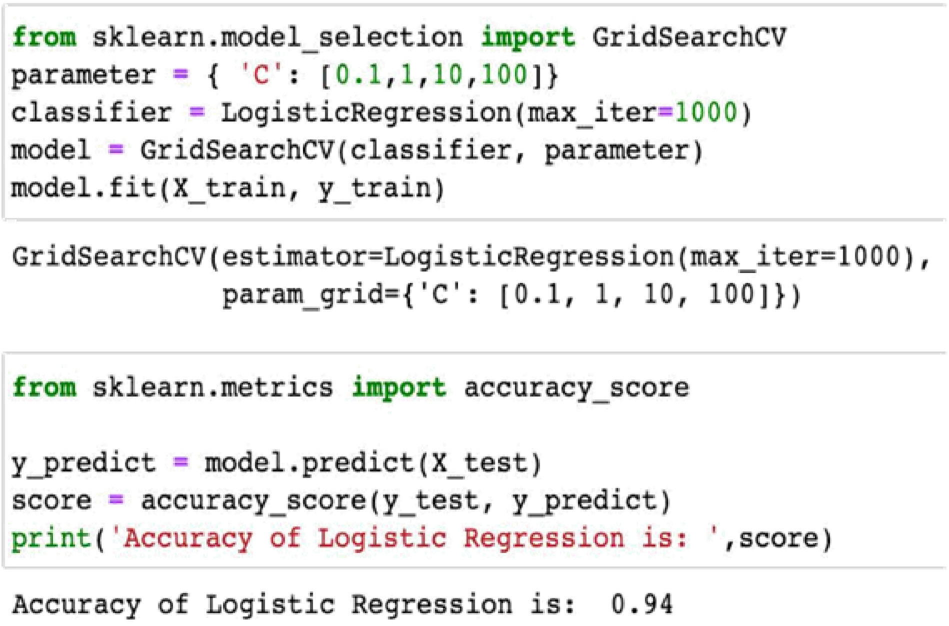


```
r2_score(y_linear_regression, y_predicted)
```

```
0.7548
```

Models

Logistic Regression

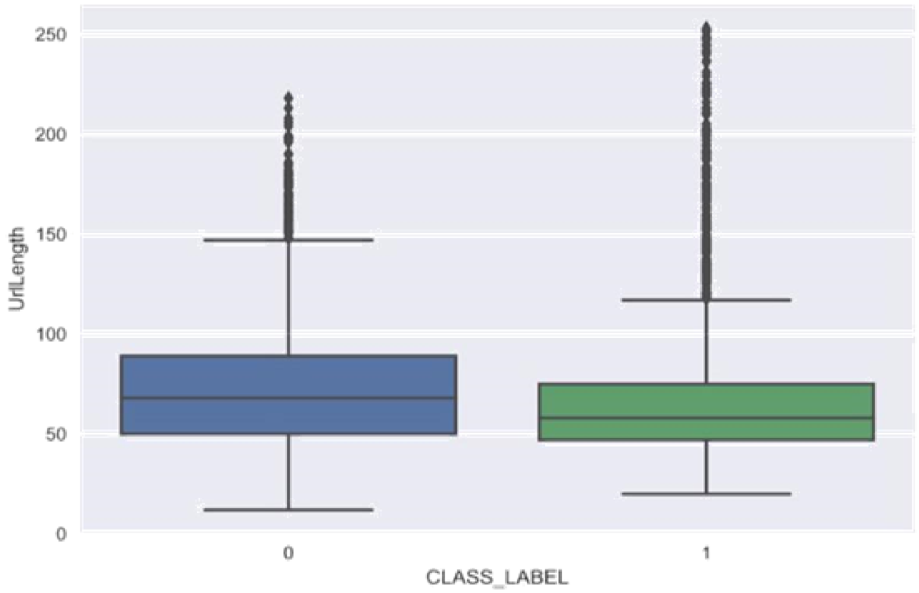


```
from sklearn.model_selection import GridSearchCV
parameter = { 'C': [0.1,1,10,100]}
classifier = LogisticRegression(max_iter=1000)
model = GridSearchCV(classifier, parameter)
model.fit(X_train, y_train)
```

```
GridSearchCV(estimator=LogisticRegression(max_iter=1000),
             param_grid={'C': [0.1, 1, 10, 100]})
```

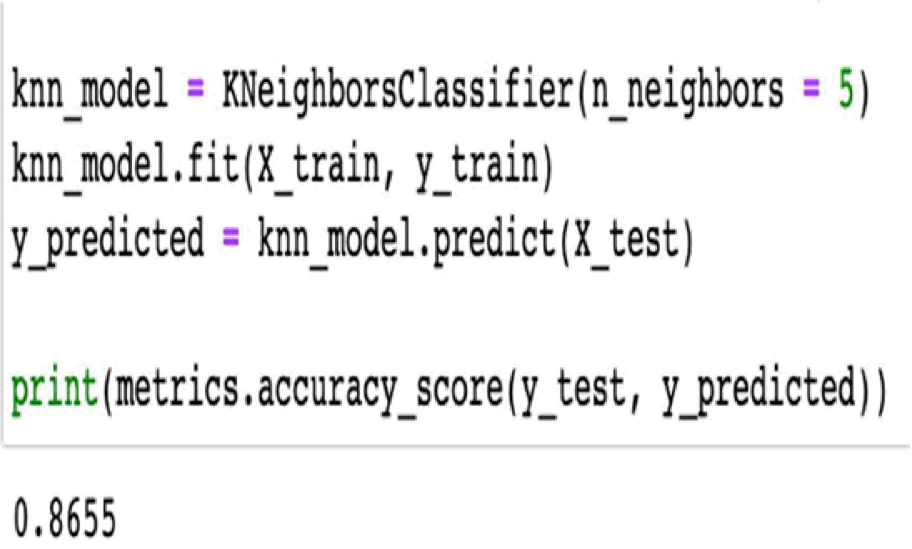
```
from sklearn.metrics import accuracy_score

y_predict = model.predict(X_test)
score = accuracy_score(y_test, y_predict)
print('Accuracy of Logistic Regression is: ',score)
```

```
Accuracy of Logistic Regression is:  0.94
```



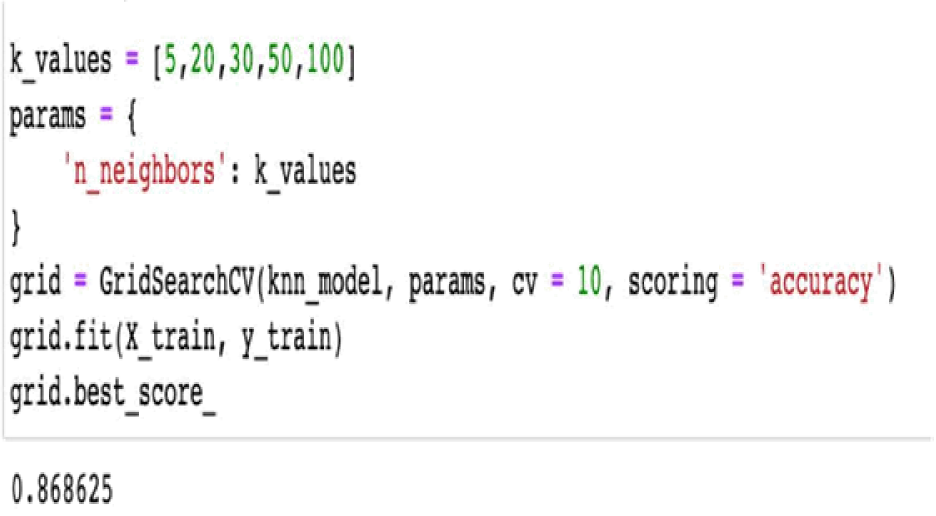
Models

K NEAREST NEIGHBOR



```
knn_model = KNeighborsClassifier(n_neighbors = 5)
knn_model.fit(X_train, y_train)
y_predicted = knn_model.predict(X_test)

print(metrics.accuracy_score(y_test, y_predicted))
```

```
0.8655
```

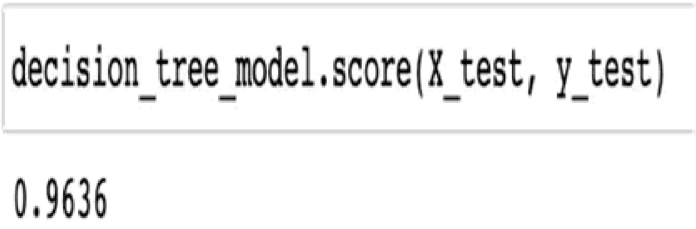


```
k_values = [5,20,30,50,100]
params = {
    'n_neighbors': k_values
}
grid = GridSearchCV(knn_model, params, cv = 10, scoring = 'accuracy')
grid.fit(X_train, y_train)
grid.best_score_
```

```
0.868625
```

Models

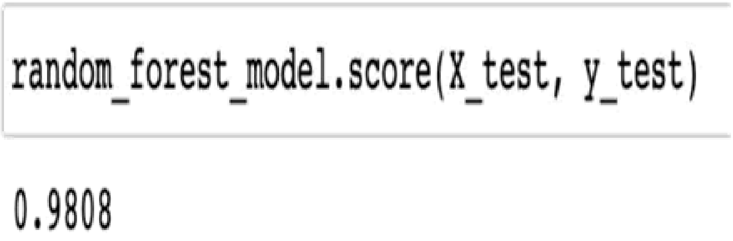
Decision Tree



```
decision_tree_model.score(X_test, y_test)
```

```
0.9636
```

Random Forest



```
random_forest_model.score(X_test, y_test)
```

```
0.9808
```

Conclusion

- Data Analysis
- Comparing accuracy of different machine learning algorithm models

Thanks for your Attention

I hope my project lived up to your expectations