Fraud URL Analyser

July 24, 2020

1 Fraud URL Analyzer

[1]:
```
import pandas as pd
import numpy as np
import random

##from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

Load Url Data classified as Good or Bad

[2]:
```
urls_data = pd.read_csv("URLdata.csv")
type(urls_data)
urls_data.head()
```

[2]:

	url	label
0	diaryofagameaddict.com	bad
1	espdesign.com.au	bad
2	iamagameaddict.com	bad
3	kalantzis.net	bad
4	slightlyoffcenter.net	bad

Check for missing data

[3]:
```
urls_data.isnull().sum().sum()
```

[3]:
```
0
```

Data Vectorization Using TfidVectorize Create A tokenizer

[6]:
```
def makeTokens(f):
    tkns_BySlash = str(f.encode('utf-8')).split('/')
    total_Tokens = []
    for i in tkns_BySlash:
        tokens = str(i).split('-')
        tkns_ByDot = []
```

```
        for j in range(0,len(tokens)):
            temp_Tokens = str(tokens[j]).split('.')
            tkns_ByDot = tkns_ByDot + temp_Tokens
        total_Tokens = total_Tokens + tokens + tkns_ByDot
    total_Tokens = list(set(total_Tokens))
    if 'com' in total_Tokens:
        total_Tokens.remove('com')
    return total_Tokens
```

[7]:
```
y = urls_data["label"]
url_list = urls_data["url"]
```

[8]:
```
# Using Default Tokenizer
#vectorizer = TfidfVectorizer()

# Using Custom Tokenizer
vectorizer = TfidfVectorizer(tokenizer=makeTokens)
X = vectorizer.fit_transform(url_list)
```

[9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
logit = LogisticRegression()
logit.fit(X_train, y_train)
```

[9]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

[10]:
```
print("Accuracy ",logit.score(X_test, y_test))
```

```
Accuracy  0.96163771063
```

Predicting With Our Model

[11]:
```
X_predict = [
   "oprahsearch.com/scripts/net19.exe",
   "nobodyspeakstruth.narod.ru/upload/main.exe",
  "server1.extra-web.cz/dbm.exe ",
   "directxex.com/uploads/565785830.been.exe",]
```

[12]:
```
X_predict = vectorizer.transform(X_predict)
New_predict = logit.predict(X_predict)
```

[13]:
```
print(New_predict)
```

```
['bad' 'bad' 'bad' 'bad']
```

[14]:
```
X_predict1 = ["best100catfights.com",
"femalewrestlingnow.com",
"jeansvixens.com",
"bakerrealestateinspections.com/file/wr/cd/",
            "en.wikipedia.org/wiki/Women_Who_Work"]
```

[15]:
```
X_predict1 = vectorizer.transform(X_predict1)
New_predict1 = logit.predict(X_predict1)
print(New_predict1)
```

```
['bad' 'bad' 'bad' 'bad' 'good']
```

[16]:
```
vectorizer = TfidfVectorizer()
```

[17]:
```
X = vectorizer.fit_transform(url_list)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[18]:
```
logitmodel = LogisticRegression()
logitmodel.fit(X_train, y_train)
```

[18]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

[19]:
```
X_predict2 = [
             "en.wikipedia.org/wiki/Women_Who_Work"]
```

[20]:
```
X_predict2 = vectorizer.transform(X_predict2)
New_predict2 = logitmodel.predict(X_predict2)
print(New_predict2)
```

```
['good']
```

[21]:
```
   # Accuracy of Our Model with our Custom Token
    print("Accuracy ",logitmodel.score(X_test, y_test))
```

```
Accuracy  0.964634392875
```

[ ]:
```

```