EMAIL SPASM CLASSIFIER

Submitted by:

SAYAN KUMAR BHUNIA

ACKNOWLEDGMENT

This includes mentioning of all the references, research papers, data sources, professionals and other resources that helped me and guided me in completion of the project.

- https://towardsdatascience.com/
- https://anshikaaxena.medium.com/
- https://medium.com/https://medium.com/

INTRODUCTION

➢ Business Problem Framing

Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the nonspam texts. It uses a binary type of classification containing the labels such as 'ham' (nonspam) and spam. Application of this can be seen in Google Mail (GMAIL) where it segregates the spam emails in order to prevent them from getting into the user's inbox.

➢ Conceptual Background of the Domain Problem

The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text.

This corpus has been collected from free or free for research sources at the Internet:

-> A collection of 5573 rows SMS spam messages was manually extracted from the Grumbletext Web site. This is a UK forum in which cell phone users make public claims about SMS spam messages, most of them without reporting the very spam message received. The identification of the text of spam messages in the claims is a very hard and time-consuming task, and it involved carefully scanning hundreds of web pages.

➢ Motivation for the Problem Undertaken

The main goal of these two parts of article is to show how you could design a spam filtering system from scratch.

Analytical Problem Framing

➢ Mathematical/ Analytical Modelling of the Problem

We shall build a supervised classification model to predict the spam.

Now, when we talk about building a supervised classifier catering to certain use-case, for example, classifying risk of loan default, following three things come into our minds:

- **Data** appropriate to the business requirement or use-case we are trying to solve
- A **classification model** which we think (or, rather assess) to be the best for our solution.
- **Optimize** the chosen model to ensure best performance.

➢ Data Sources and their formats

I am using CSV (comma-separated values) format file which is having 5572 rows × 5 columns

```
In [6]: df=pd.read_csv(r"C:\Users\sayan\OneDrive\Desktop\fliprobo projects\Spam Project\spam.csv",encoding = "ISO-8859-1")
        df
```

Out[6]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will ì_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

5572 rows × 5 columns

➢ Data Pre-processing

Following steps have been performed on the data.

- **checking missing values**-
  - If there is any missing value present in your data set then for a better and correct accuracy you have to impute it.
  - If missing data present in object type column, then you have to take most frequent value for your missing data.
  - If missing data present in int or float type column then use mean/median for missing value.

  In the following case no missing value present:

```
[24]: df.isnull().sum()

t[24]: v1    0
       v2    0
       dtype: int64
```

- **Encoding categorical variables** -as we can see there are 3 object data type columns present so we will encode it into (int) format.
  - Apply **Label Encoding**, if number of categories in a categorical variable is equal to 2.
  - Apply **One-Hot Encoding**, if number of categories in a categorical variable is greater than 2.

In the following case Label Encoding is used.

```
In [26]: import sklearn
         from sklearn.preprocessing import LabelEncoder,OneHotEncoder
```

```
In [27]: le=LabelEncoder()
         list1=['v1']
         for i in list1:
             df[i]=le.fit_transform(df[i].astype(str))
         df
```

- **Feature scaling-** Feature Scaling ensures that all features will get equal importance in supervised classifier models. Standard scaler was used to scale all features in the data.

- **Reducing dimension of the data-** Sklearn's pca can be used to apply principal component analysis on the data. This helped in finding the vectors of maximal variance in the data.

- **Outliers detection-** In simple words, an outlier is an observation that diverges from an overall pattern on a sample.

There are many types of outlier detection techniques such as Z-Score or Extreme Value Analysis, Probabilistic and Statistical Modelling, Information Theory Models, Standard Deviation etc.

➢ Outliers Removal

In our dataset, we observed variations in the relation between values of some attributes.

So that these types of rows are dropped from the dataset.

➢ Software Requirements and library Used

```
[1]: import pandas
     import pandas as pd
     import numpy as np
     import seaborn as sns
     import matplotlib.pyplot as plt
     import warnings
     warnings.filterwarnings('ignore')
```

```
10]: import sklearn
     from sklearn.preprocessing import LabelEncoder,OneHotEncoder
```

- **NumPy**
  NumPy is a popular Python library for multi-dimensional array and matrix processing because it can be used to perform a great variety of mathematical operations. Its capability to handle linear algebra, Fourier transform, and more, makes NumPy ideal for machine learning and artificial intelligence (AI) projects, allowing users to manipulate the matrix to easily improve machine learning performance. NumPy is faster and easier to use than most other Python libraries.

- **Scikit-learn**
  Scikit-learn is a very popular machine learning library that is built on NumPy and SciPy. It supports most of the classic supervised and unsupervised learning algorithms, and it can also be used for data mining, modelling, and analysis.

- **Seaborn**
  Seaborn is another open-source Python library, one that is based on Matplotlib (which focuses on plotting and data visualization) but features Pandas' data structures. Seaborn is often used in ML projects because it can generate plots of learning data. Of all the Python libraries, it produces the most aesthetically pleasing graphs and plots, making it an effective choice if you'll also use it for marketing and data analysis.

- **Pandas**
  Pandas is another Python library that is built on top of NumPy, responsible for preparing high-level data sets for machine learning and training. It relies on two types of data structures, one-dimensional (series) and two-dimensional (Data Frame). This allows Pandas to be applicable in a variety of industries including finance, engineering, and statistics. Unlike the slow-moving animals themselves, the Pandas library is quick, compliant, and flexible.

➢ Class imbalance problem

The first challenge we hit upon exploring the data, is class imbalanced problem. Imbalance data will lead to a bad accuracy of a model. To achieve better accuracy, we'll balance the data by using Smote Over Sampling Method.
In this case we are not using it.

Model/s Development and Evaluation

➢ Run and evaluate selected models

Let's select our classification model for this project:

- Random Forest Classifier
- Gradient Boosting Classifier
- KNeighborsClassifier
- DecisionTreeClassifier
- SVC
- LogisticRegression

➢ Testing of Identified Approaches (Algorithms)

```
                with 43478 stored elements in Compressed Sparse Row format>

In [126]: import sklearn
          from sklearn.neighbors import KNeighborsClassifier
          from sklearn.ensemble import GradientBoostingClassifier
          from sklearn.svm import SVC
          from sklearn.ensemble import RandomForestClassifier
          from sklearn.linear_model import LogisticRegression
          from sklearn.tree import DecisionTreeClassifier
          from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
          from sklearn.model_selection import train_test_split

In [120]: x1_train,x1_test,y_train,y_test=train_test_split(x1,y,test_size=.30,random_state=50)

In [127]: rf=RandomForestClassifier(n_estimators=100,random_state=50)
          lg=LogisticRegression()
          knn=KNeighborsClassifier()
          dtc=DecisionTreeClassifier()
          svc=SVC()
          model=[rf,lg,knn,dtc,svc]
          for m in model:
              m.fit(x1_train,y_train)
              predm=m.predict(x1_test)
              print('accuracy_score of',m ,'is:')
              print(accuracy_score(y_test,predm))
              print(confusion_matrix(y_test,predm))
              print(classification_report(y_test,predm))

          accuracy_score of RandomForestClassifier(random_state=50) is:
          0.97188995215311
```

➢ Key Metrics for success in solving problem under consideration

Selection of a model requires evaluation and evaluation requires a good metric. This is indeed important. If we optimize a model based on incorrect metric, then, our model might not be suitable for the business goals.

We have a number of metrics, for example, accuracy, recall, precision, F1 score, area under receiver operating characteristic curve, to choose from.

CONCLUSION

Key aspects of building successful classifier are:

- Selecting correct data according to the purpose or problem statement.
- Proper processing and understanding of the data
- Selecting the model and optimizing the model.

I have used label encoder for encoding object data type into int datatype as machine doesn't understand object type data.

I have used three classification model and found Random Forest Classifier to be the best fit. It is giving 98% accuracy.