E-mail Spam Detection

Big Data Technologies

Submitted to:
Prof. Suhas Joshi

Presented By:
PG 11 Mihir Agarwal
PG 14 Hussain Rangwala
PG 15 Keval Pambhar
PG 28 Viren Kaddam

INTRODUCTION

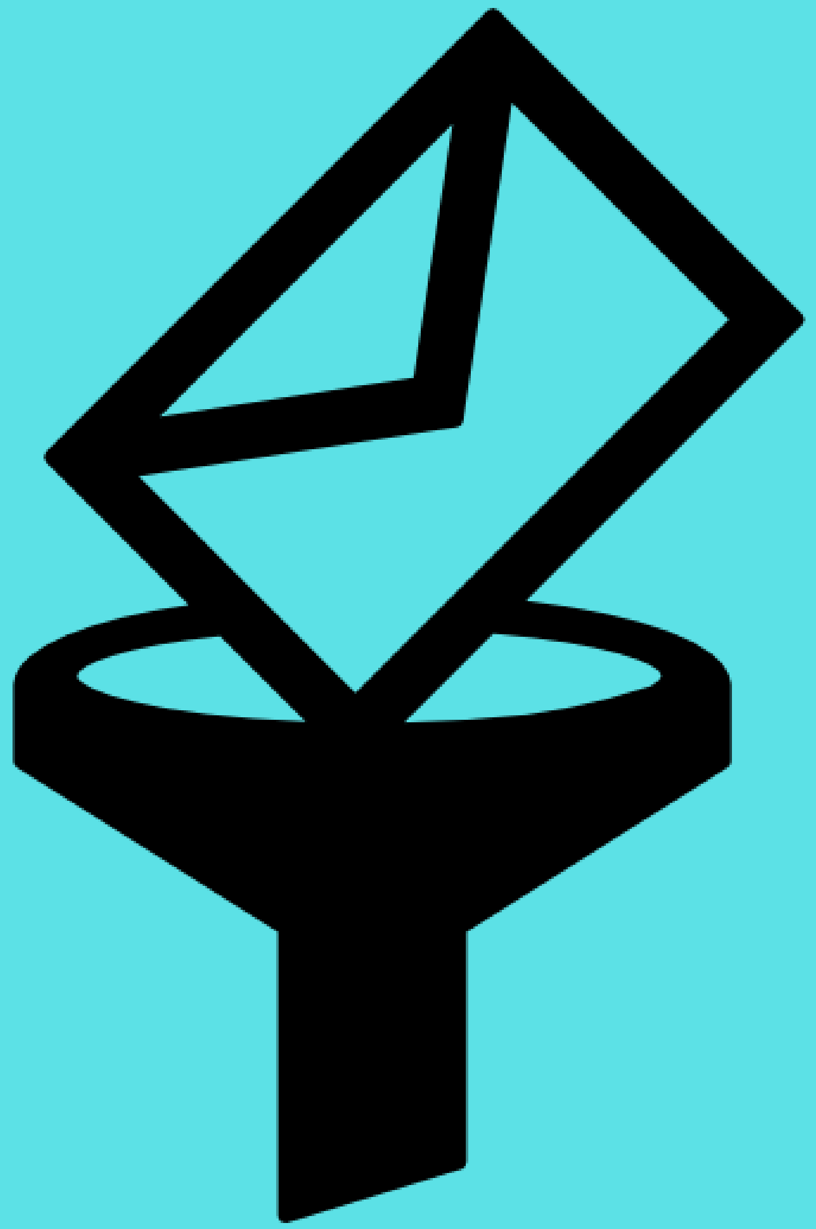
- Email has become extremely popular among people nowadays.
- Unfortunately its usage has been bedeviled with the huge presence of unsolicited and sometimes fraudulent emails which must be promptly detected and isolated.
- Spam detection is used to differentiate between spam email and non-spam emails, thereby making it possible to prevent spam mail from getting into the inbox of users.

Solutions Available

- Content-Based Filtering Technique.
- Case Base Spam Filtering Method.
- Heuristic or Rule-Based Spam Filtering Technique.
- Adaptive Spam Filtering Technique.

Solutions Chosen

Case Base Spam Filtering Method.

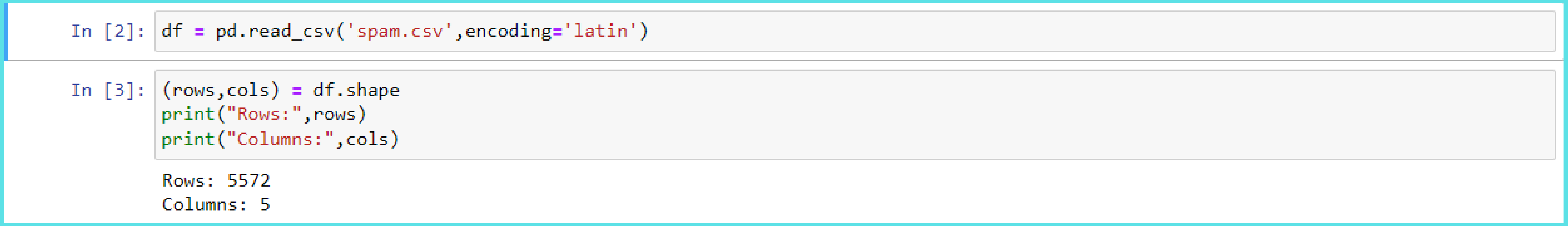


Tech Stack Used

- Numpy
- Pandas
- Scikit-learn
- NLTK
- Matplotlib and Seaborn

Working

- **Importing and Reading Dataset**

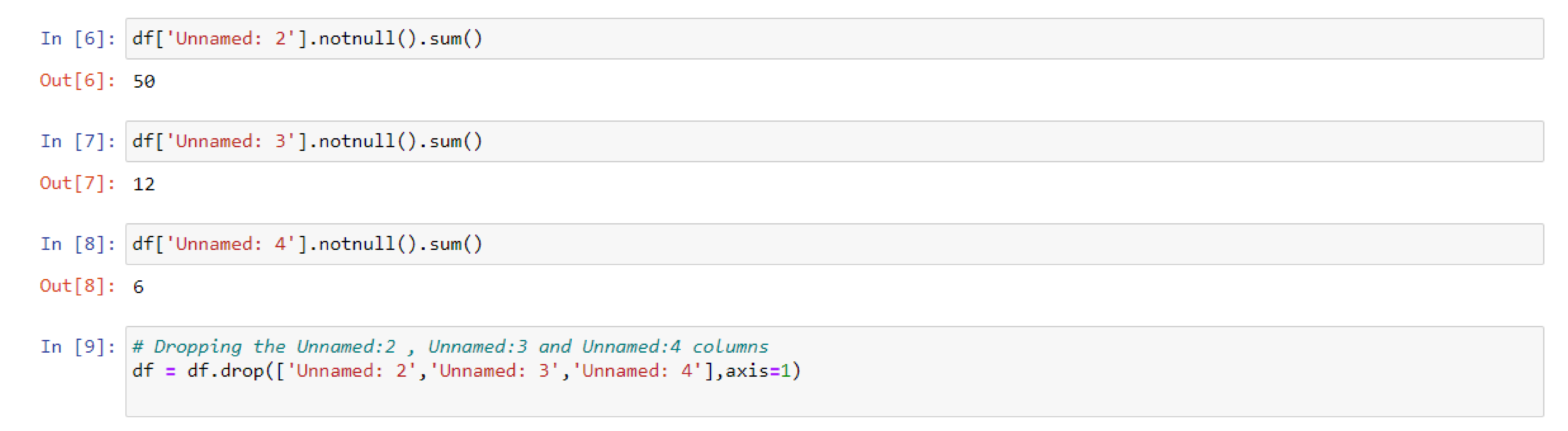


```
In [2]: df = pd.read_csv('spam.csv',encoding='latin')
```

```
In [3]: (rows,cols) = df.shape
        print("Rows:",rows)
        print("Columns:",cols)

        Rows: 5572
        Columns: 5
```

Data Cleaning

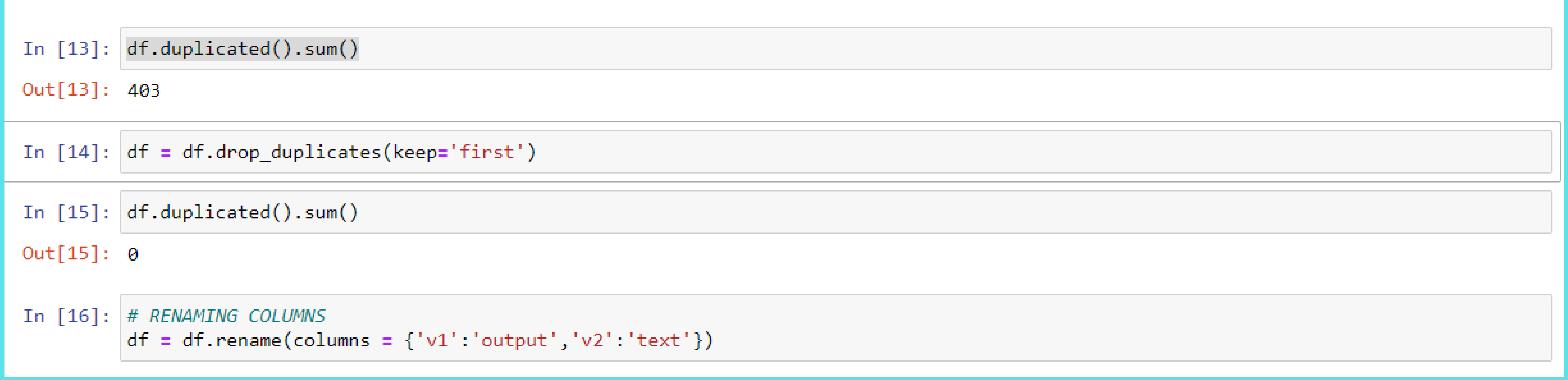
- **Checking For Null Values**



```
In [6]: df['Unnamed: 2'].notnull().sum()

Out[6]: 50

In [7]: df['Unnamed: 3'].notnull().sum()

Out[7]: 12

In [8]: df['Unnamed: 4'].notnull().sum()

Out[8]: 6

In [9]: # Dropping the Unnamed:2 , Unnamed:3 and Unnamed:4 columns
        df = df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)
```

Data Cleaning

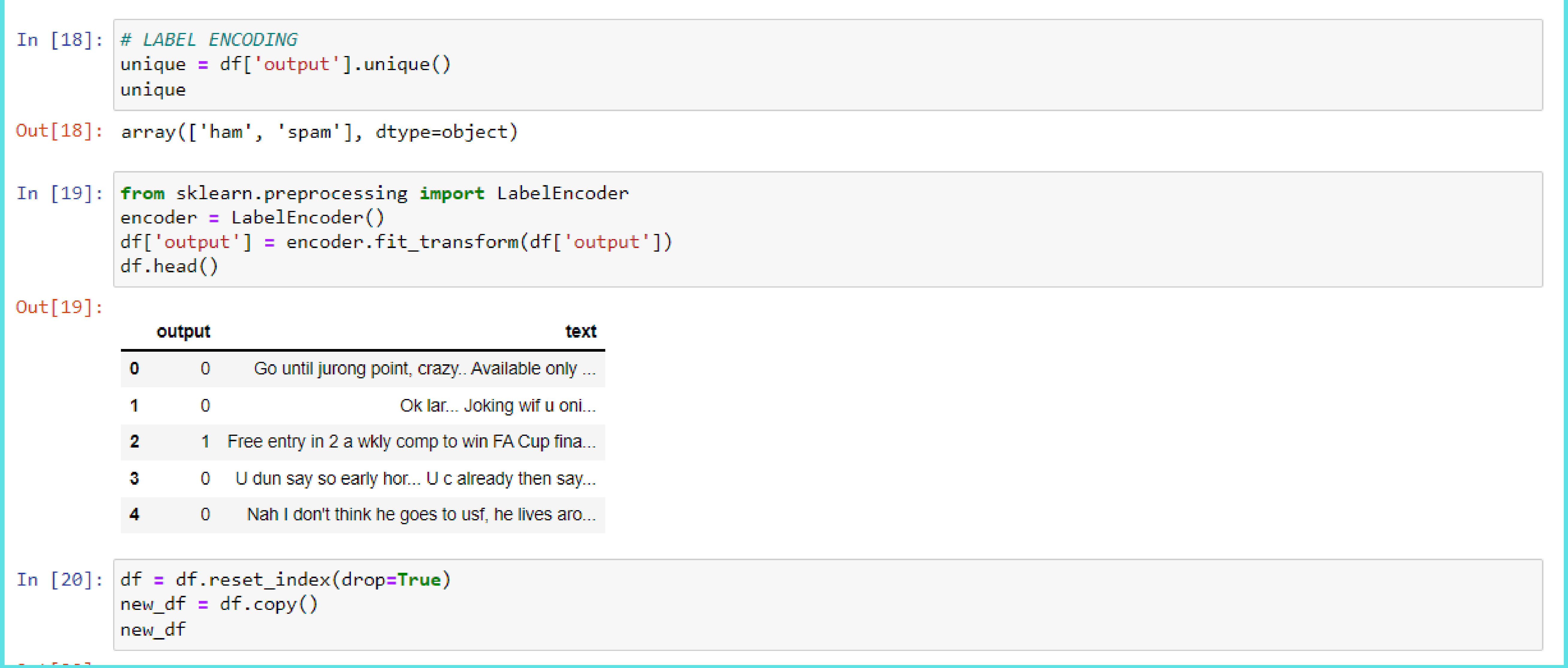
- **Checking For Duplicate Values**



```
In [13]: df.duplicated().sum()

Out[13]: 403

In [14]: df = df.drop_duplicates(keep='first')

In [15]: df.duplicated().sum()

Out[15]: 0

In [16]: # RENAMING COLUMNS
         df = df.rename(columns = {'v1':'output','v2':'text'})
```

Data Cleaning

- **Label Encoding**



```
In [18]: # LABEL ENCODING
         unique = df['output'].unique()
         unique

Out[18]: array(['ham', 'spam'], dtype=object)
```

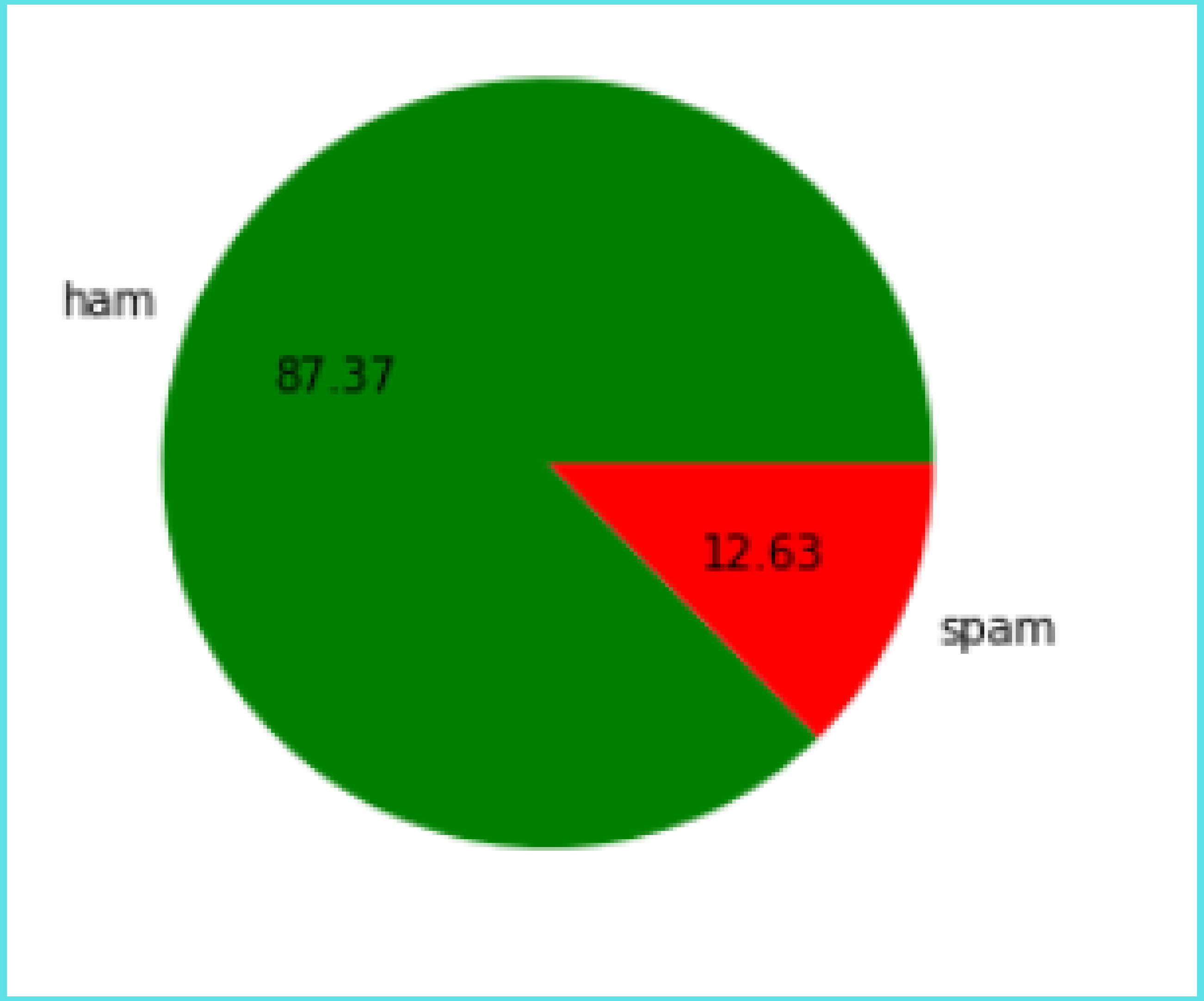
```
In [19]: from sklearn.preprocessing import LabelEncoder
         encoder = LabelEncoder()
         df['output'] = encoder.fit_transform(df['output'])
         df.head()

Out[19]:
```

	output	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

```
In [20]: df = df.reset_index(drop=True)
         new_df = df.copy()
         new_df
```

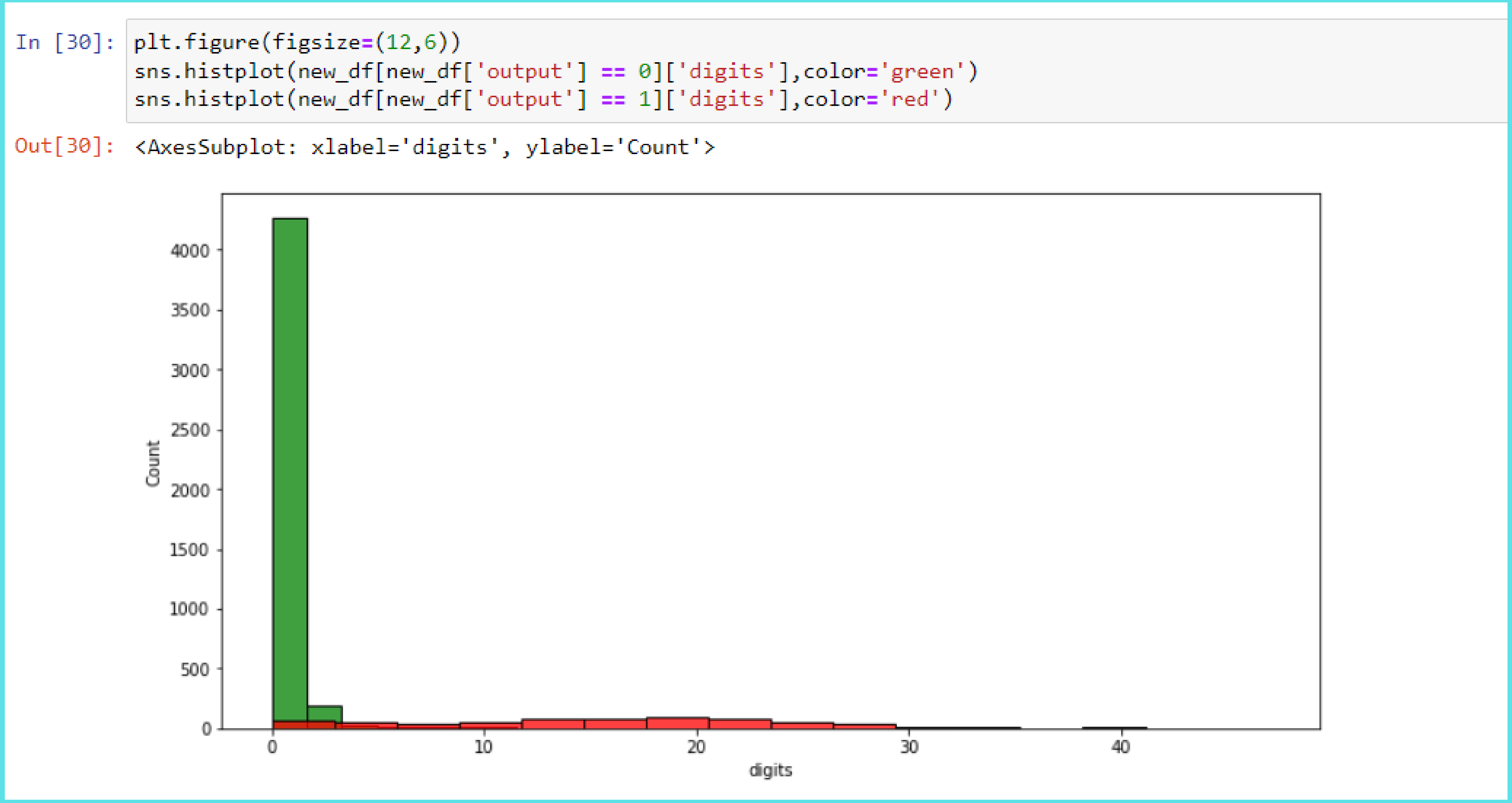
Data Analysis

- Data distribution of Ham vs Spam



Data Analysis

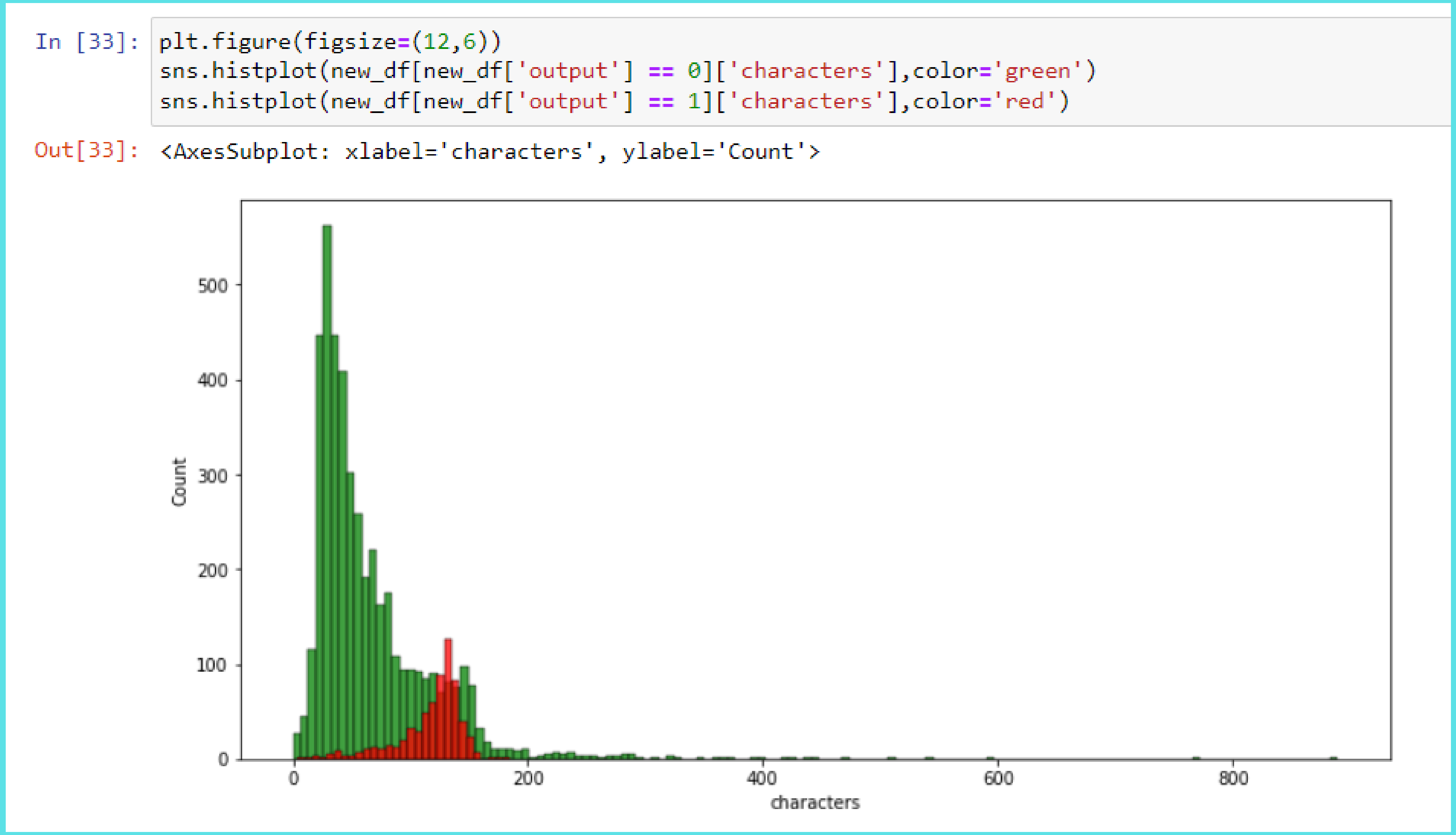
- **No. of digits in HAM Vs No. of digits in SPAM**



```
In [30]: plt.figure(figsize=(12,6))
         sns.histplot(new_df[new_df['output'] == 0]['digits'],color='green')
         sns.histplot(new_df[new_df['output'] == 1]['digits'],color='red')

Out[30]: <AxesSubplot: xlabel='digits', ylabel='Count'>
```

Data Analysis

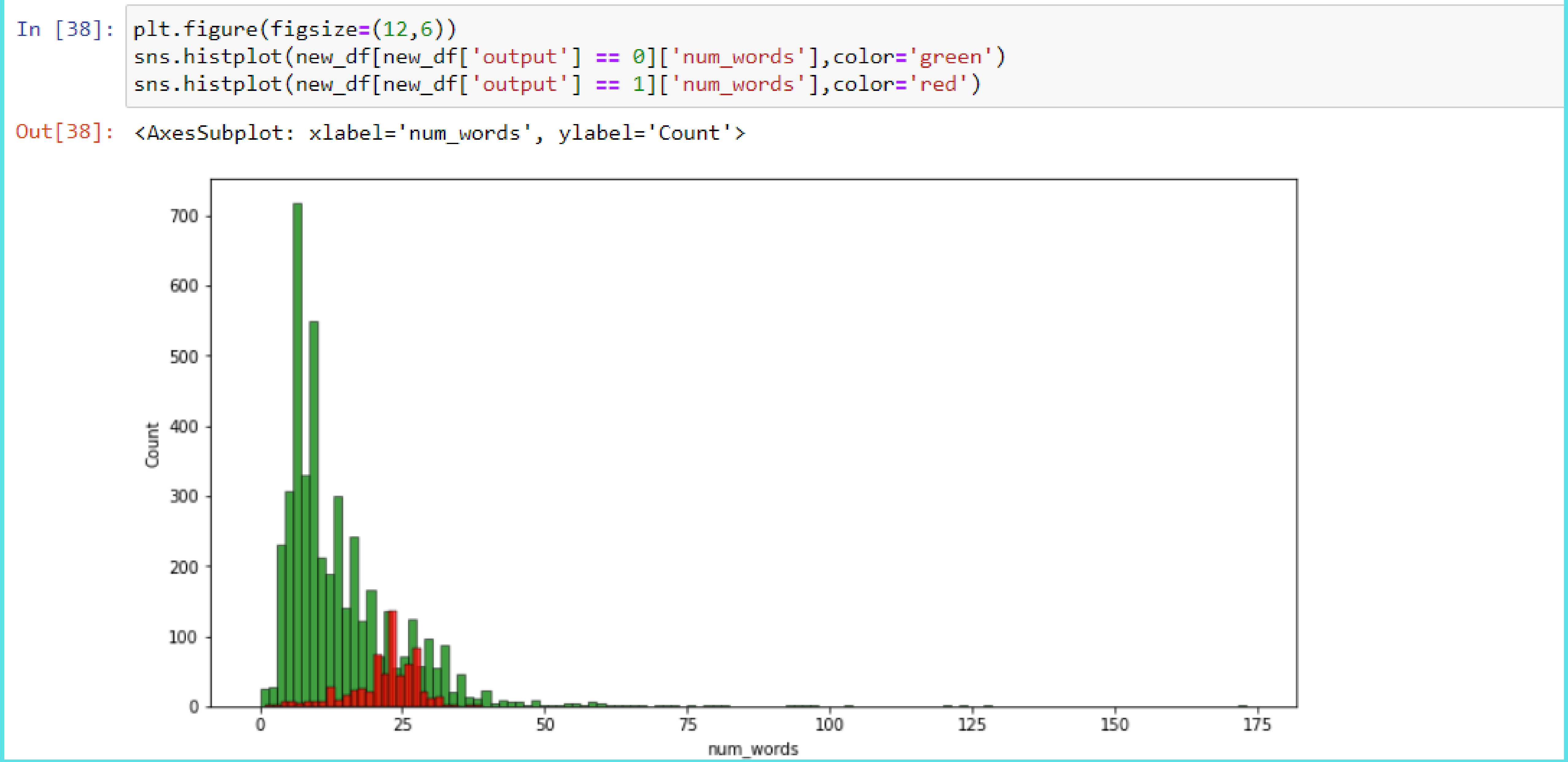
- **No. of characters in HAM Vs No. of characters in SPAM**



```
In [33]: plt.figure(figsize=(12,6))
         sns.histplot(new_df[new_df['output'] == 0]['characters'],color='green')
         sns.histplot(new_df[new_df['output'] == 1]['characters'],color='red')

Out[33]: <AxesSubplot: xlabel='characters', ylabel='Count'>
```

Data Analysis

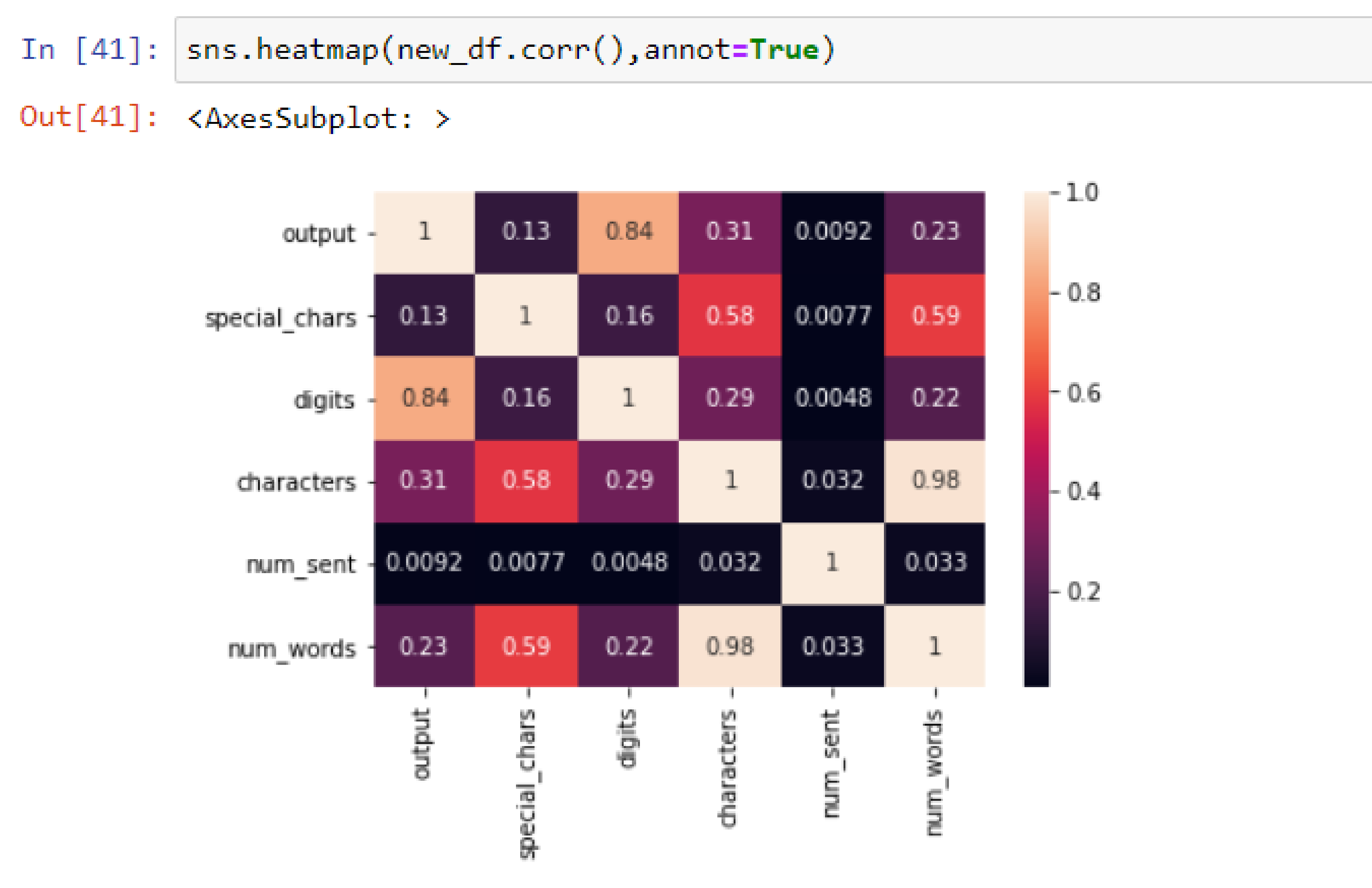
- **No. of words in HAM Vs No. of words in SPAM**



```
In [38]: plt.figure(figsize=(12,6))
         sns.histplot(new_df[new_df['output'] == 0]['num_words'],color='green')
         sns.histplot(new_df[new_df['output'] == 1]['num_words'],color='red')

Out[38]: <AxesSubplot: xlabel='num_words', ylabel='Count'>
```

Data Analysis

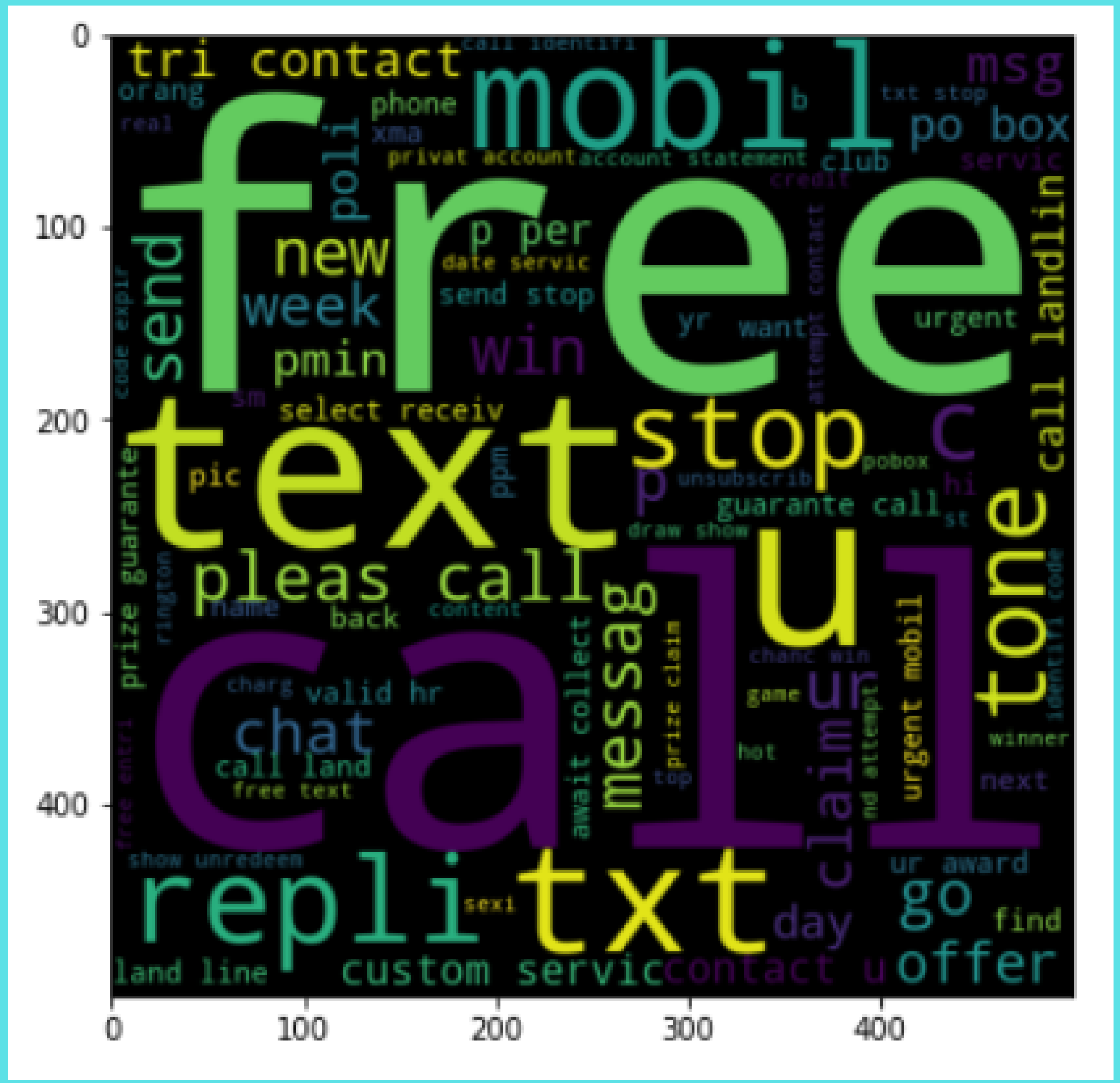
- **Correlation**



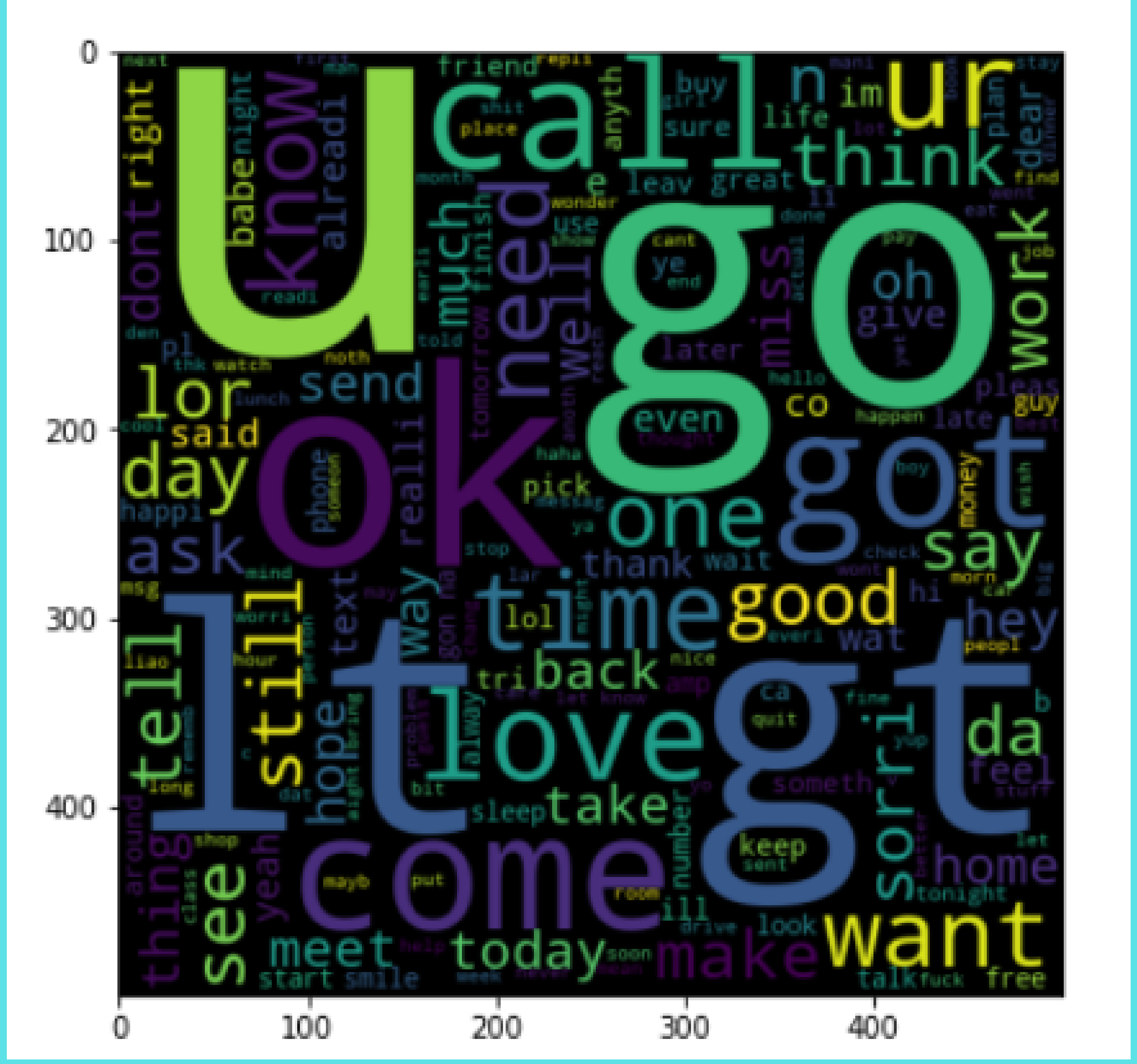
```
In [41]: sns.heatmap(new_df.corr(),annot=True)

Out[41]: <AxesSubplot: >
```

Data Analysis

- **Word Cloud**



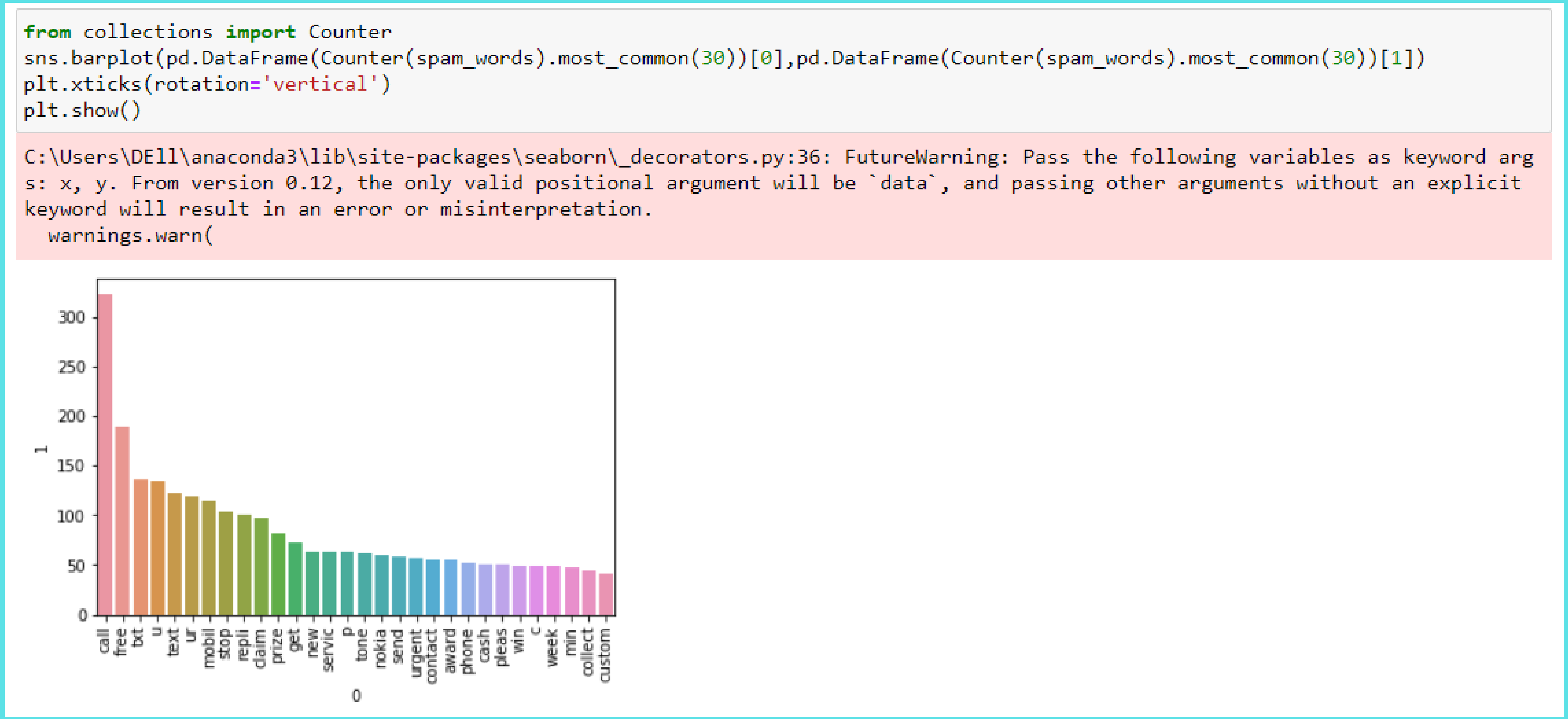
SPAM



HAM

Data Analysis

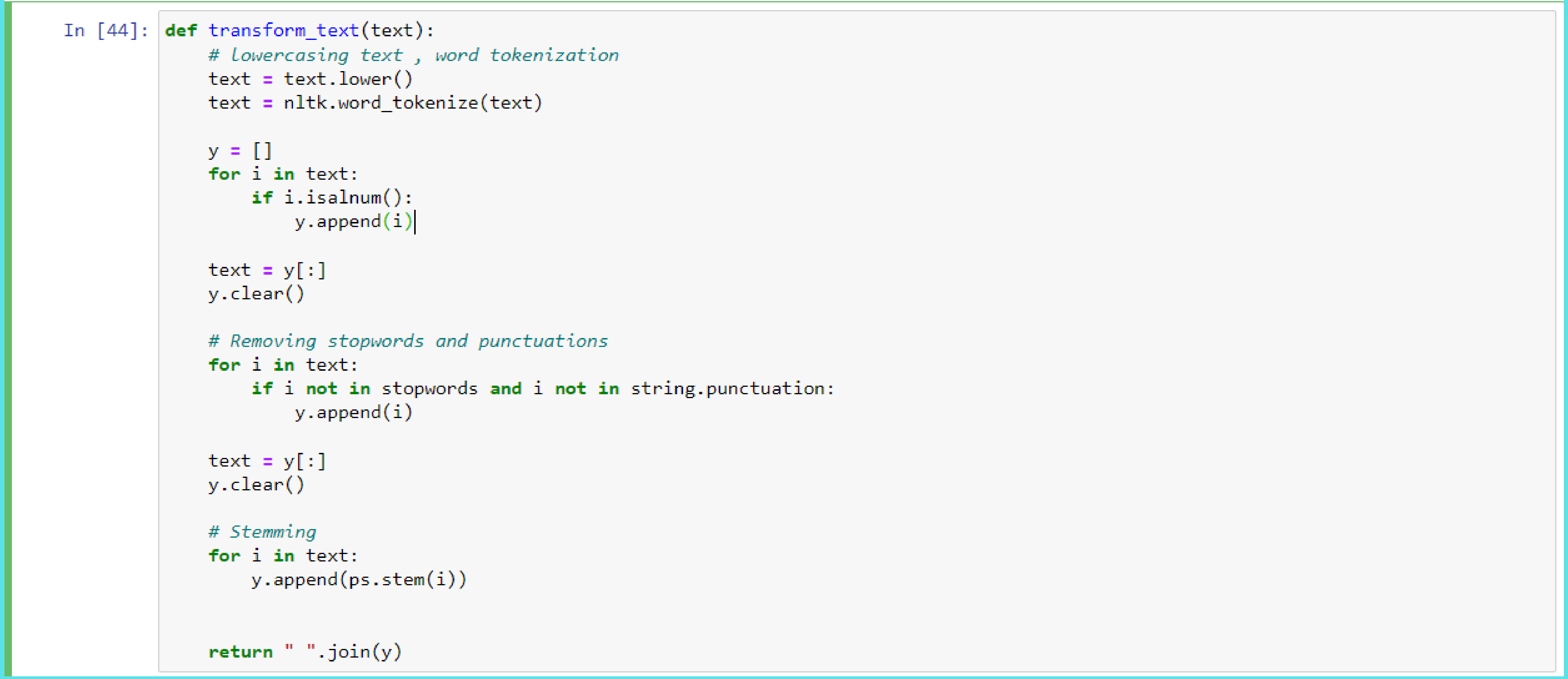
- **Most Words in SPAM**



```python
from collections import Counter
sns.barplot(pd.DataFrame(Counter(spam_words).most_common(30))[0],pd.DataFrame(Counter(spam_words).most_common(30))[1])
plt.xticks(rotation='vertical')
plt.show()
```

```
C:\Users\DEll\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```

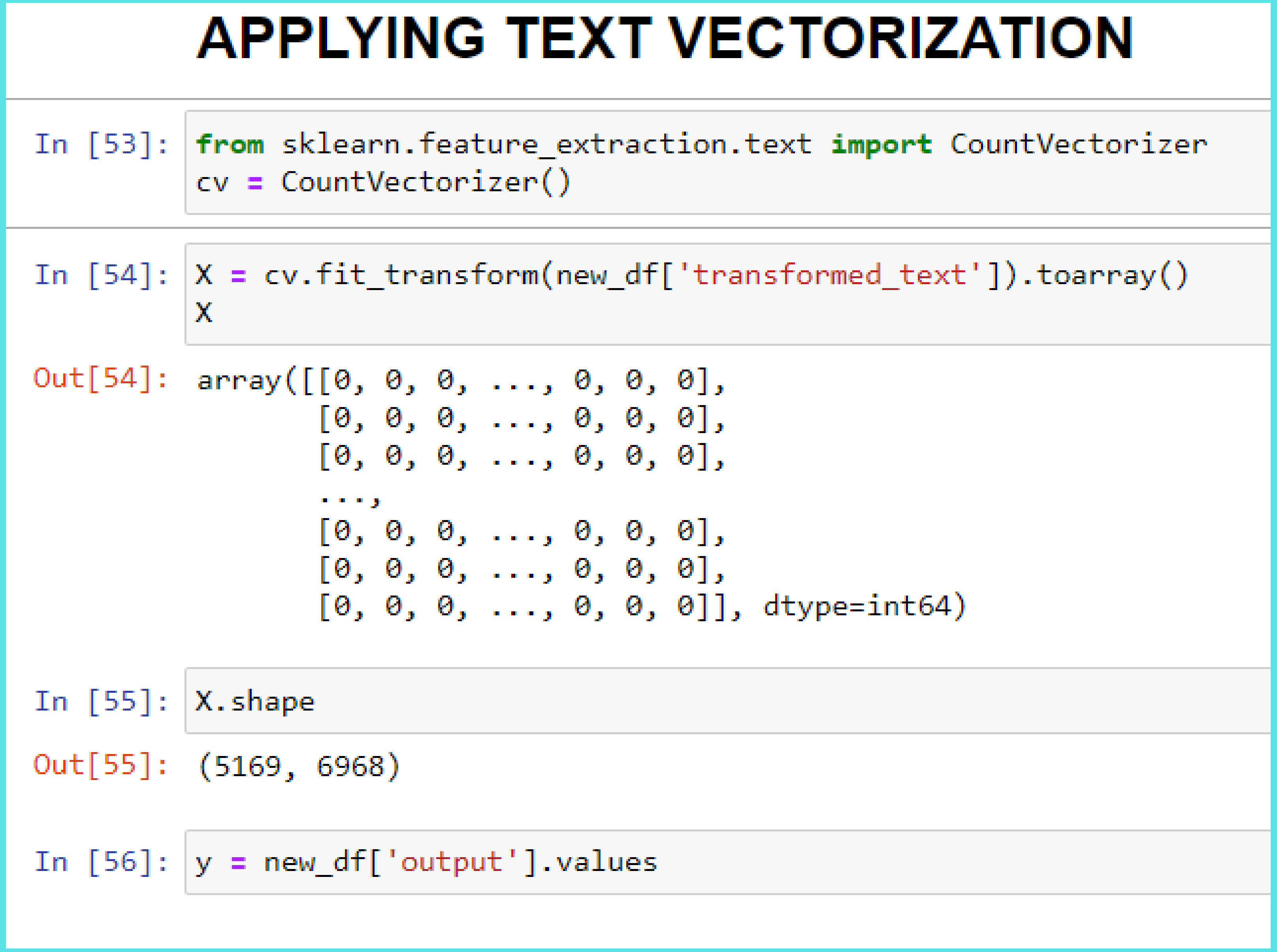
Data Preprocessing



```
In [44]: def transform_text(text):
             # lowercasing text , word tokenization
             text = text.lower()
             text = nltk.word_tokenize(text)

             y = []
             for i in text:
                 if i.isalnum():
                     y.append(i)

             text = y[:]
             y.clear()

             # Removing stopwords and punctuations
             for i in text:
                 if i not in stopwords and i not in string.punctuation:
                     y.append(i)

             text = y[:]
             y.clear()

             # Stemming
             for i in text:
                 y.append(ps.stem(i))


             return " ".join(y)
```

Data Preprocessing

- **Applying Text Vectorization**

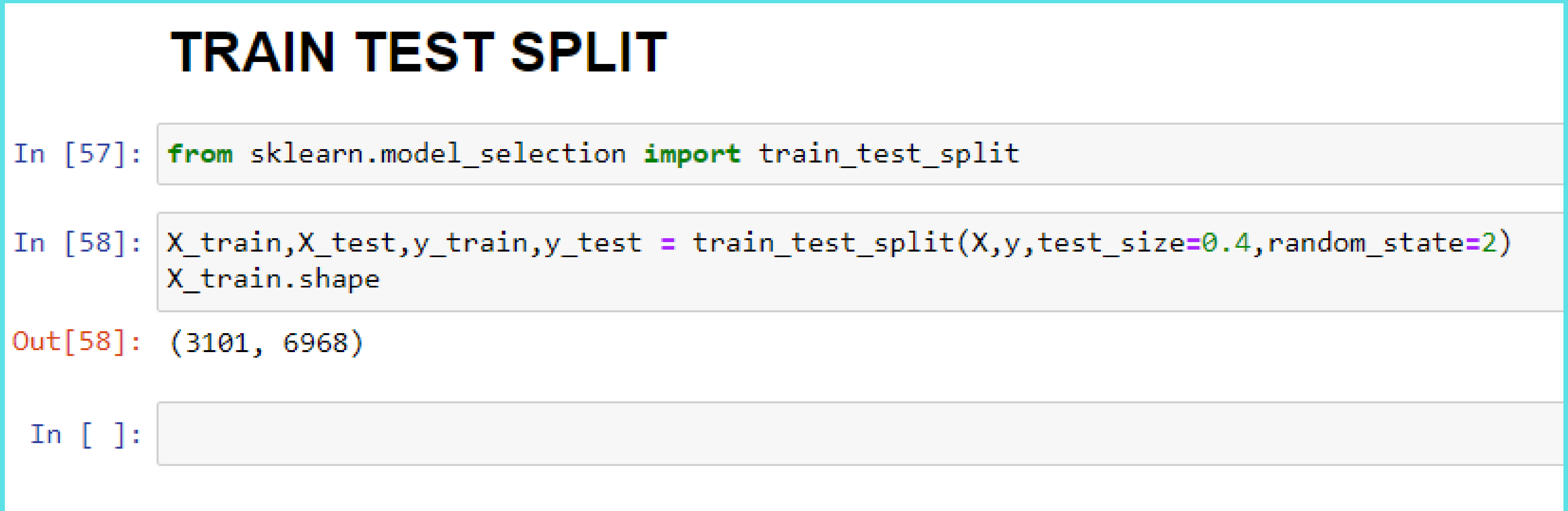


APPLYING TEXT VECTORIZATION

```
In [53]: from sklearn.feature_extraction.text import CountVectorizer
         cv = CountVectorizer()

In [54]: X = cv.fit_transform(new_df['transformed_text']).toarray()
         X

Out[54]: array([[0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                ...,
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0]], dtype=int64)

In [55]: X.shape

Out[55]: (5169, 6968)

In [56]: y = new_df['output'].values
```

Data Preprocessing

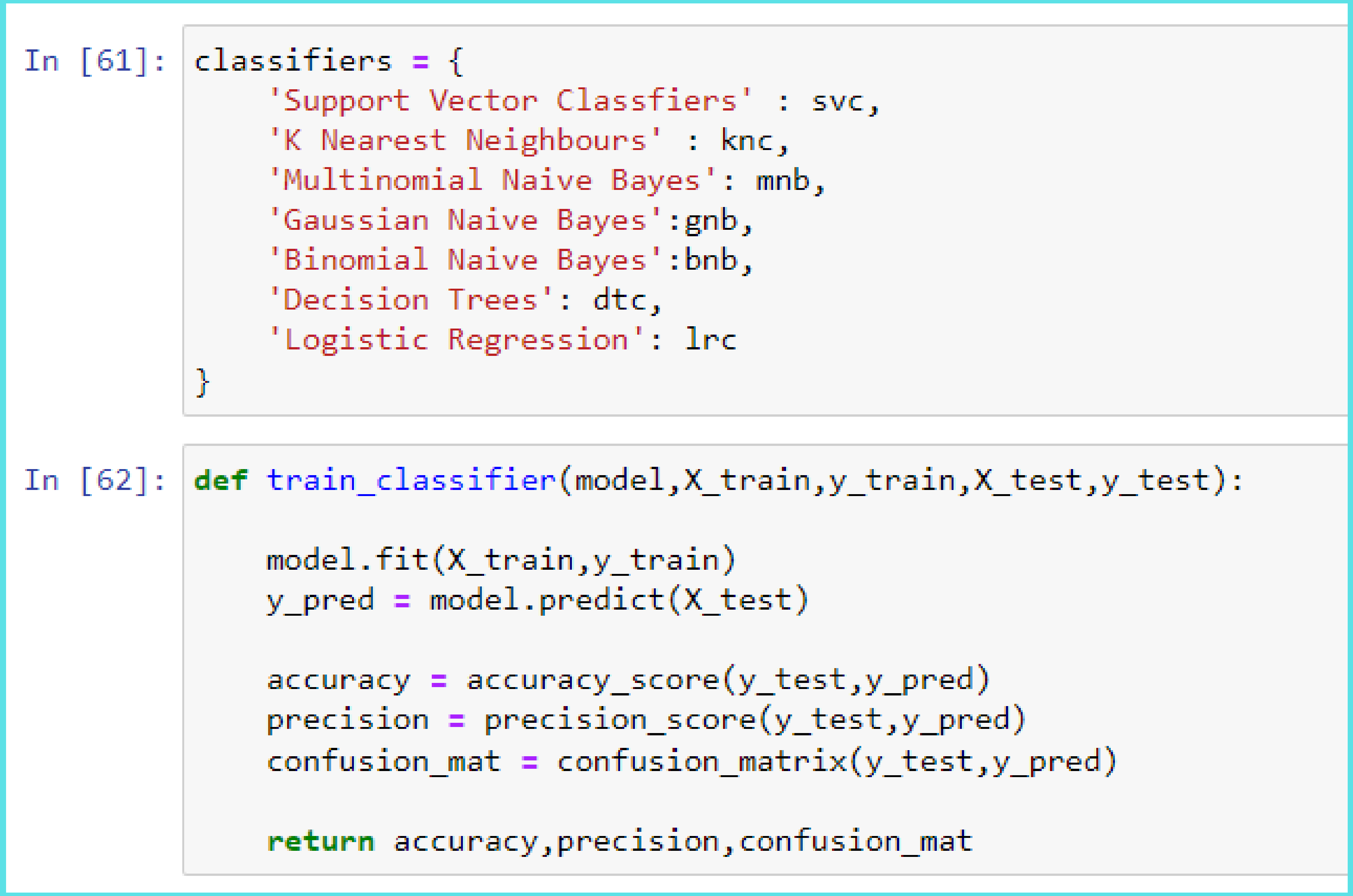
- **Train Test Split**



TRAIN TEST SPLIT

```
In [57]: from sklearn.model_selection import train_test_split

In [58]: X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.4,random_state=2)
         X_train.shape

Out[58]: (3101, 6968)

In [ ]:
```

Classification Models

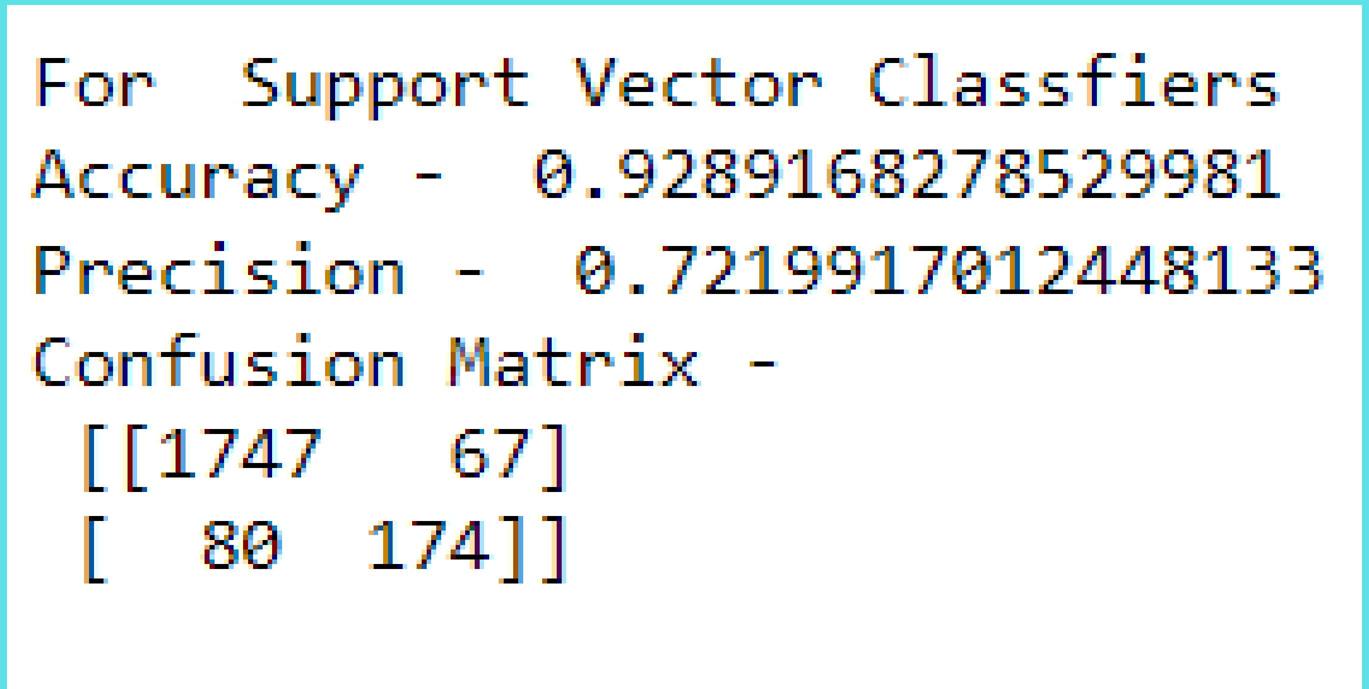


```
In [61]: classifiers = {
             'Support Vector Classfiers' : svc,
             'K Nearest Neighbours' : knc,
             'Multinomial Naive Bayes': mnb,
             'Gaussian Naive Bayes':gnb,
             'Binomial Naive Bayes':bnb,
             'Decision Trees': dtc,
             'Logistic Regression': lrc
         }
```

```
In [62]: def train_classifier(model,X_train,y_train,X_test,y_test):

             model.fit(X_train,y_train)
             y_pred = model.predict(X_test)

             accuracy = accuracy_score(y_test,y_pred)
             precision = precision_score(y_test,y_pred)
             confusion_mat = confusion_matrix(y_test,y_pred)

             return accuracy,precision,confusion_mat
```

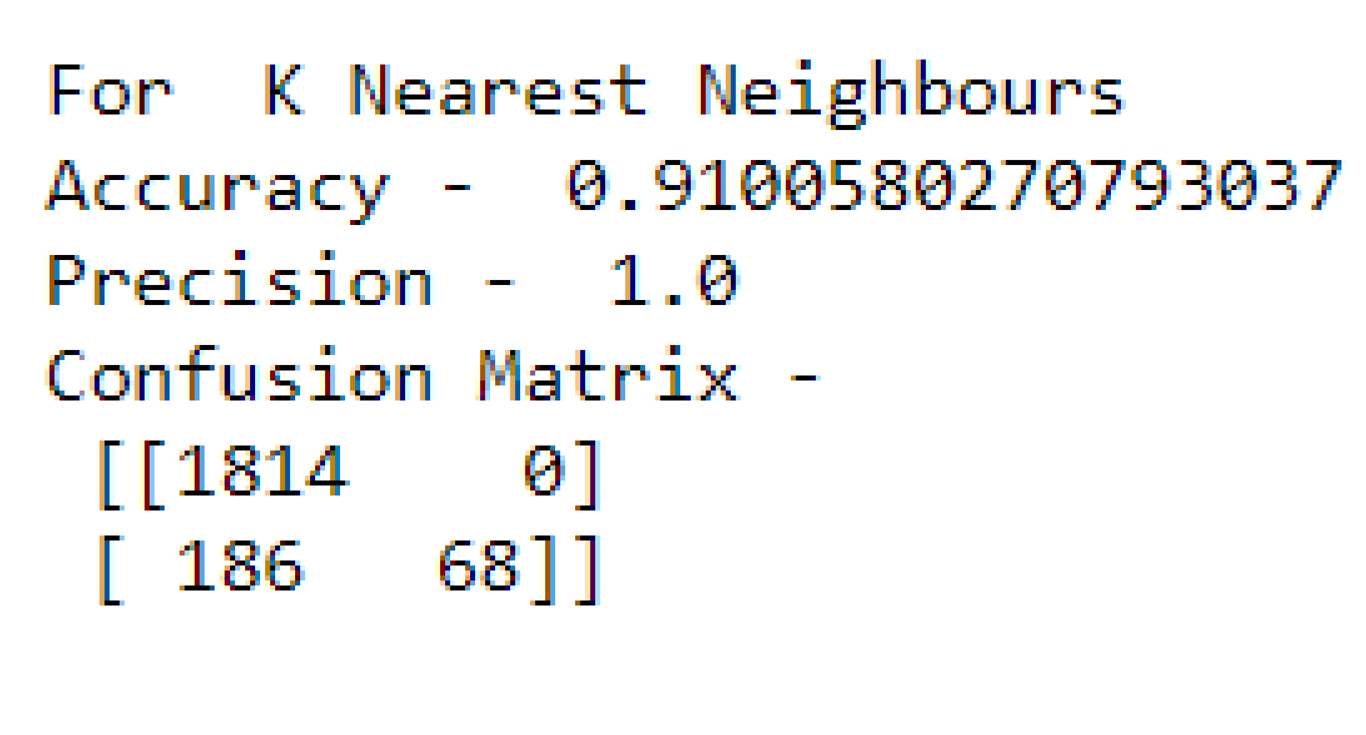
Results

- **Support Vector Classification**



```
For   Support Vector Classfiers
Accuracy -  0.9289168278529981
Precision -  0.7219917012448133
Confusion Matrix -
 [[1747   67]
 [  80  174]]
```

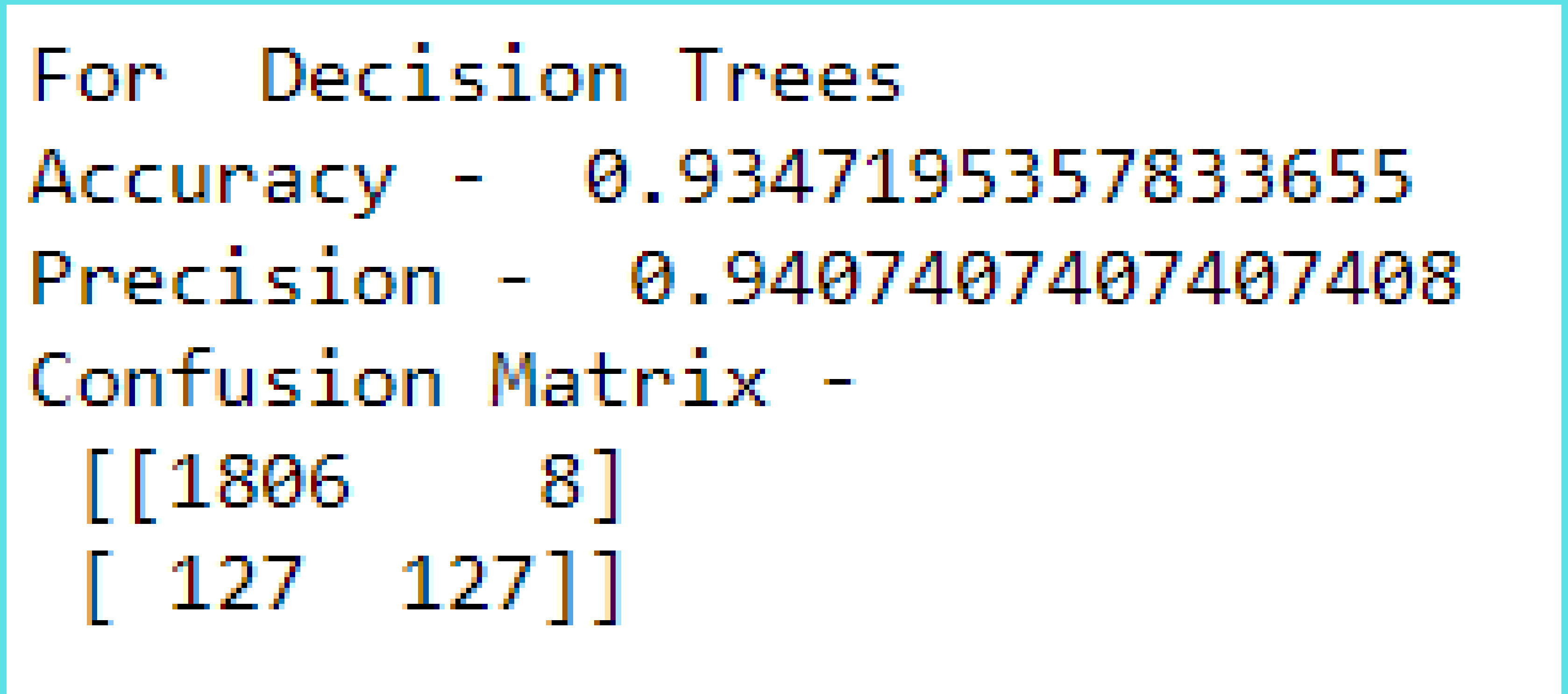
- **K Nearest Neighbours**



```
For   K Nearest Neighbours
Accuracy -  0.9100580270793037
Precision -  1.0
Confusion Matrix -
 [[1814    0]
 [ 186   68]]
```

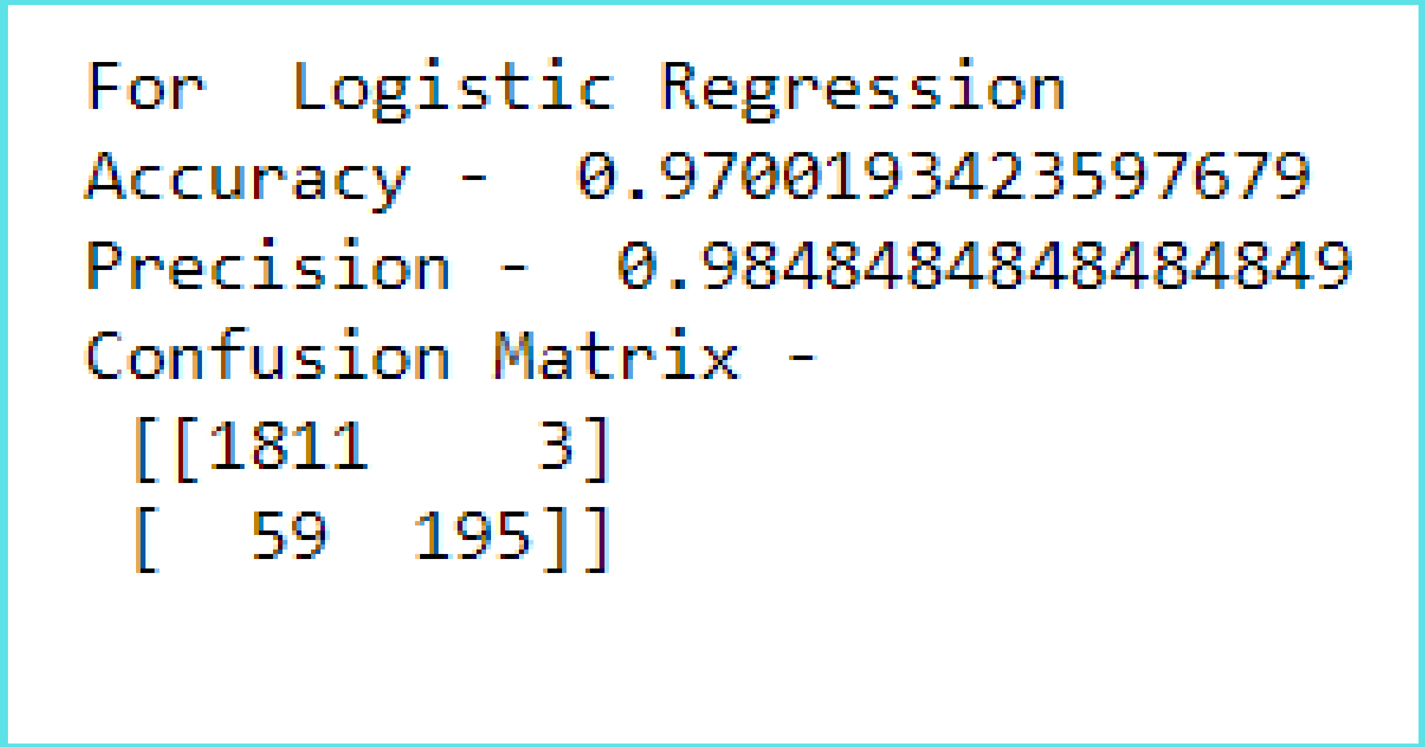
Results

- **Decision Trees**



```
For  Decision Trees
Accuracy -  0.9347195357833655
Precision -  0.9407407407407408
Confusion Matrix -
 [[1806    8]
 [ 127  127]]
```

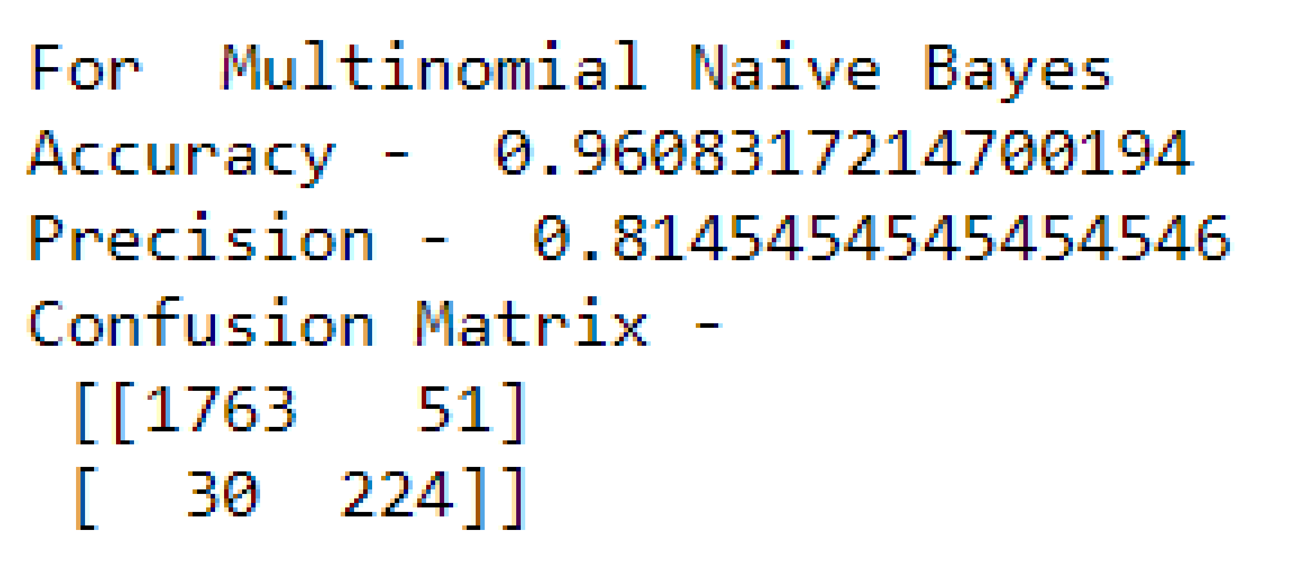
- **Logistic Regression**



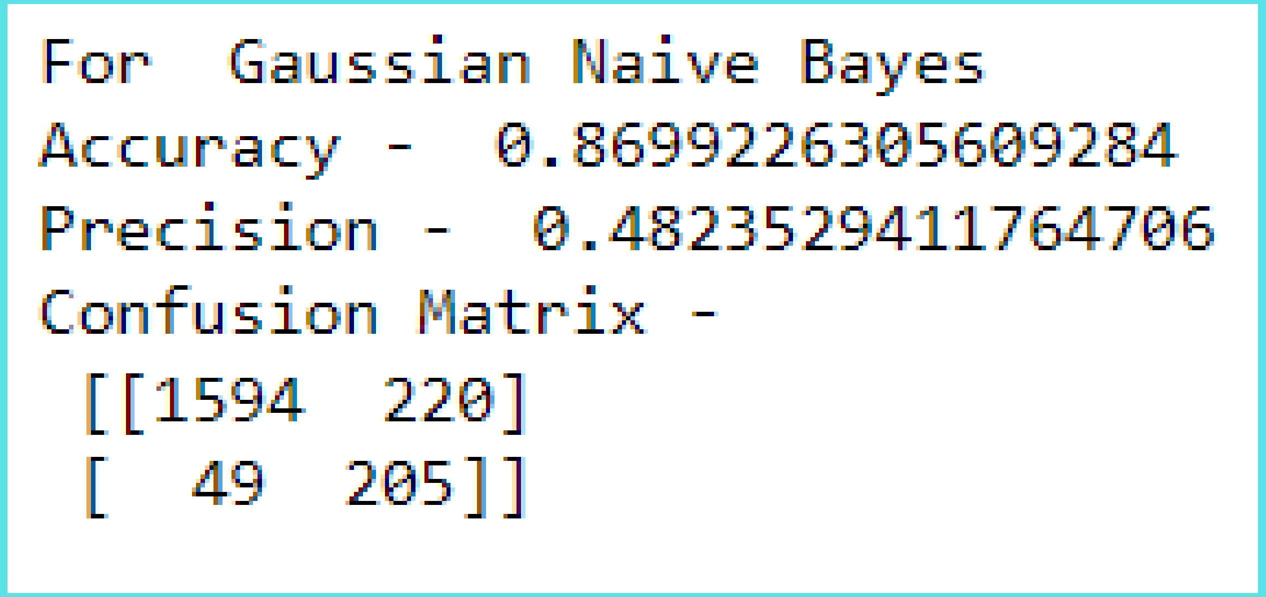
```
For  Logistic Regression
Accuracy -  0.9700193423597679
Precision -  0.9848484848484849
Confusion Matrix -
 [[1811    3]
 [  59  195]]
```

Results

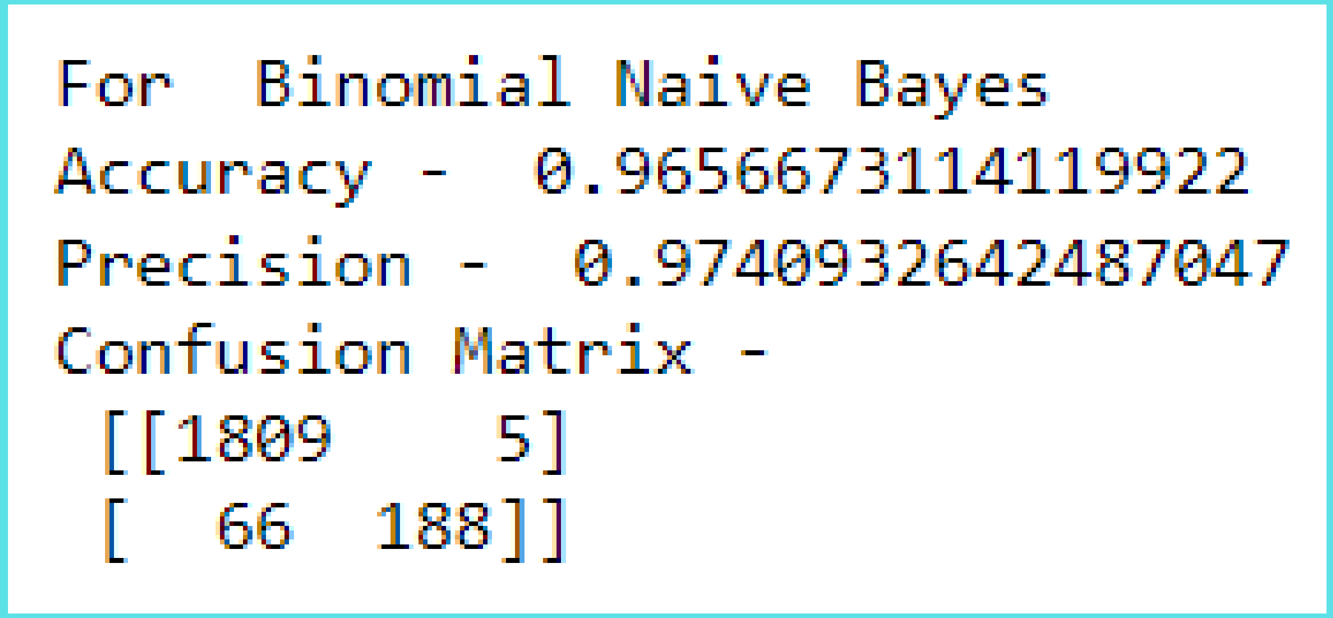
- **Naive Bayes**



```
For  Multinomial Naive Bayes
Accuracy -  0.9608317214700194
Precision -  0.8145454545454546
Confusion Matrix -
 [[1763   51]
 [  30  224]]
```



```
For  Gaussian Naive Bayes
Accuracy -  0.8699226305609284
Precision -  0.4823529411764706
Confusion Matrix -
 [[1594  220]
 [  49  205]]
```



```
For  Binomial Naive Bayes
Accuracy -  0.965667311411992
Precision -  0.9740932642487047
Confusion Matrix -
 [[1809    5]
 [  66  188]]
```

Learnings

- Working with Pandas
- Data Analysis with matplotlib and seaborn
- Natural Language Processing

References

- Mujtaba, Ghulam, et al. "Email classification research trends: Review and open issues." IEEE Access 5 (2017)
- Cihan Varol, Hezha M.Tareq Abdulhadi “Comparison of String Matching Algorithms on Spam Email Detection”, International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism Dec, 2018.
- Thashina Sultana , K A Sapnaz , Fathima Sana , Jamedar Najath, 2020, Email based Spam Detection, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 09, Issue 06 (June 2020)

Thank You!