EMAIL SPAM CLASSIFIER PROJECT

Submitted By:

Lakshmi Rajendra Thute

ACKNOWLEDGEMENT

I am very much Thankful to FlipRobo Technologies for giving me the opportunity to work with them and to work on this project and also, I am very grateful to Data Trained Education Team for their support and help to understand each and every concept of machine learning which helped me a lot while working on this project. I thought, I am fortunate to become a part of FlipRobo Technology.

References:

Google website

Stack overflow

Analytics Vidya

Medium

Data trained notes

INTRODUCTION

The SMS Spam Collection is a set of SMS'S tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according being ham (legitimate) or spam.

Spam Detector is used to detect unwanted, malicious and virus infected texts and helps to separate them from the no spam texts. It uses a binary type of classification containing the labels such as '**ham'** (no spam) and **spam**. Application of this can be seen in Google Mail (GMAIL) where it segregates the spam emails to prevent them from getting into the user's inbox.

ANALYTICAL PROBLEM FRAMING

We got the dataset available from client, and we must build a model that is able to detect the spam emails or SMSs and to sort them from ham emails. The dataset contains total 5 columns, they are v1, v2, Unnamed: 2, Unnamed: 3, Unnamed: 4, from which Unnamed: 2, Unnamed: 3, Unnamed: 4, are irrelevant as they only contain null values, so we dropped them.

- **Mathematical/ Analytical Modelling of the Problem**

Here I firstly read the dataset in jypyter notebook for cleaning the dataset I use pre-processing techniques, then did the Exploratory Data Analysis, then Encoding and lastly model Building and Evaluation.

- **Data Sources and their formats**

I got the dataset in CSV format, and I read the data in Jupyter Notebook using pandas data frame.

- **Data Pre-processing Done**

The dataset contains some irrelevant columns firstly we dropped that columns then we check the unique and duplicate values in the dataset, and removed them

Data Collection And Pre-processing

In [4]:
```
# Collecting data from excel file.
df = pd.read_excel(r"C:\Users\91749\Downloads\email.xlsx")
df
```

Out[4]:

	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

• Hardware and Software Requirements and Tools Used

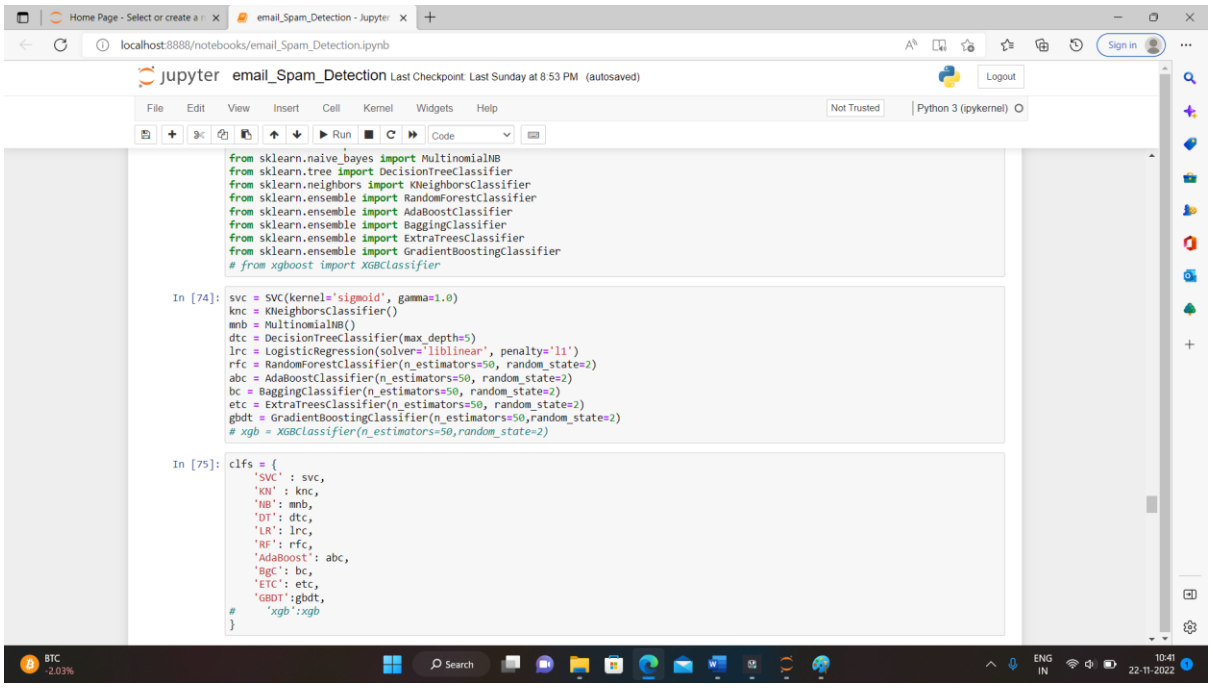
Here for this project, I used Jupyter notebook and libraries such as pandas and NumPy for mathematical operations, matplotlib and seaborn for various type of data visualizations and to explore and for better understanding of the dataset.

• Identification of possible problem-solving approaches (methods)

The statistical summary shows the total count of 1592 rows then mean, min value, max value, standard deviation, and quartiles shows up and down values that means the data contains outliers.

• Testing of Identified Approaches (Algorithms)

- Logistic Regression
- SVC
- Multinomial NB
- Decision Tree Classifier
- KNeighbors Classifier
- Random Forest Classifier
- AdaBoost Classifier
- Bagging Classifier
- Extra-Trees Classifier
- Gradient Boosting Classifier



Home Page - Select or create a n | email_Spam_Detection - Jupyter

localhost:8888/notebooks/email_Spam_Detection.ipynb

jupyter email_Spam_Detection Last Checkpoint: Last Sunday at 8:53 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

In [75]:
```
clfs = {
    'SVC' : svc,
    'KN' : knc,
    'NB': mnb,
    'DT': dtc,
    'LR': lrc,
    'RF': rfc,
    'AdaBoost': abc,
    'BgC': bc,
    'ETC': etc,
    'GBDT':gbdt,
#     'xgb':xgb
}
```

In [76]:
```
def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)

    return accuracy,precision
```

In [77]:
```
accuracy_scores = []
precision_scores = []

for name,clf in clfs.items():

    current_accuracy,current_precision = train_classifier(clf, X_train,y_train,X_test,y_test)

    print("For ",name)
    print("Accuracy - ",current_accuracy)
    print("Precision - ",current_precision)

    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

BTC -2.03% | Search | ENG IN | 10:41 22-11-2022

Home Page - Select or create a n | email_Spam_Detection - Jupyter

localhost:8888/notebooks/email_Spam_Detection.ipynb

jupyter email_Spam_Detection Last Checkpoint: Last Sunday at 8:53 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

```
    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

```
For  SVC
Accuracy -  0.9758220502901354
Precision -  0.9747899159663865
For  KN
Accuracy -  0.9052224371373307
Precision -  1.0
For  NB
Accuracy -  0.9709864603481625
Precision -  1.0
For  DT
Accuracy -  0.9294003868471954
Precision -  0.8282828282828283
For  LR
Accuracy -  0.9584139264990329
Precision -  0.9702970297029703
For  RF
Accuracy -  0.9748549323017408
Precision -  0.9827586206896551
For  AdaBoost
Accuracy -  0.960348162475822
Precision -  0.9292035398230089
For  BgC
Accuracy -  0.9574468085106383
Precision -  0.8671875
For  ETC
Accuracy -  0.9748549323017408
Precision -  0.9745762711864406
For  GBDT
Accuracy -  0.9477756286266924
Precision -  0.92
```

In [78]:
```
clf_performance_df =  pd.DataFrame({'Algorithm': clfs.keys(),'Accuracy_score':accuracy_scores,'Precision_Score':precision_scores
```

BTC -2.03% | Search | ENG IN | 10:41 22-11-2022

Home Page - Select or create a n | email_Spam_Detection - Jupyter

localhost:8888/notebooks/email_Spam_Detection.ipynb

jupyter email_Spam_Detection Last Checkpoint: Last Sunday at 8:53 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

In [78]:
```
clf_performance_df =  pd.DataFrame({'Algorithm': clfs.keys(),'Accuracy_score':accuracy_scores,'Precision_Score':precision_scores
```

In [79]:
```
clf_performance_df
```

Out[79]:

	Algorithm	Accuracy_score	Precision_Score
1	KN	0.905222	1.000000
2	NB	0.970986	1.000000
5	RF	0.974855	0.982759
0	SVC	0.975822	0.974790
8	ETC	0.974855	0.974576
4	LR	0.958414	0.970297
6	AdaBoost	0.960348	0.929204
9	GBDT	0.947776	0.920000
7	BgC	0.957447	0.867188
3	DT	0.929400	0.828283

In [80]:
```
clf_performance_df1 = pd.melt(clf_performance_df, id_vars = 'Algorithm')
```

In [81]:
```
clf_performance_df1
```

Out[81]:

	Algorithm	variable	value
0	KN	Accuracy_score	0.905222
1	NB	Accuracy_score	0.970986
2	RF	Accuracy_score	0.974855
3	SVC	Accuracy_score	0.975822

BTC -2.03% | Search | ENG IN | 10:42 22-11-2022

Home Page - Select or create a n | email_Spam_Detection - Jupyter

localhost:8888/notebooks/email_Spam_Detection.ipynb

jupyter email_Spam_Detection Last Checkpoint: Last Sunday at 8:53 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

3	DT	0.929400	0.828283

In [80]:
```
clf_performance_df1 = pd.melt(clf_performance_df, id_vars = 'Algorithm')
```

In [81]:
```
clf_performance_df1
```

Out[81]:

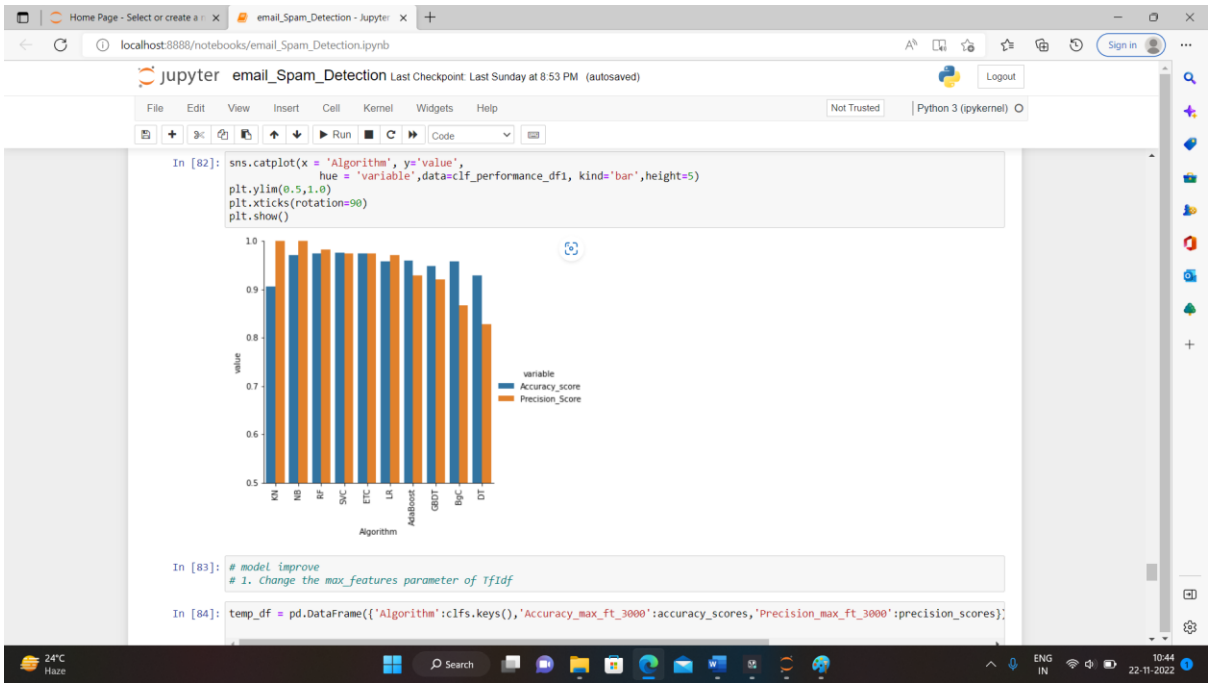
	Algorithm	variable	value
0	KN	Accuracy_score	0.905222
1	NB	Accuracy_score	0.970986
2	RF	Accuracy_score	0.974855
3	SVC	Accuracy_score	0.975822
4	ETC	Accuracy_score	0.974855
5	LR	Accuracy_score	0.958414
6	AdaBoost	Accuracy_score	0.960348
7	GBDT	Accuracy_score	0.947776
8	BgC	Accuracy_score	0.957447
9	DT	Accuracy_score	0.929400
10	KN	Precision_Score	1.000000
11	NB	Precision_Score	1.000000
12	RF	Precision_Score	0.982759
13	SVC	Precision_Score	0.974790
14	ETC	Precision_Score	0.974576
15	LR	Precision_Score	0.970297
16	AdaBoost	Precision_Score	0.929204
17	GBDT	Precision_Score	0.920000
18	BgC	Precision_Score	0.867188

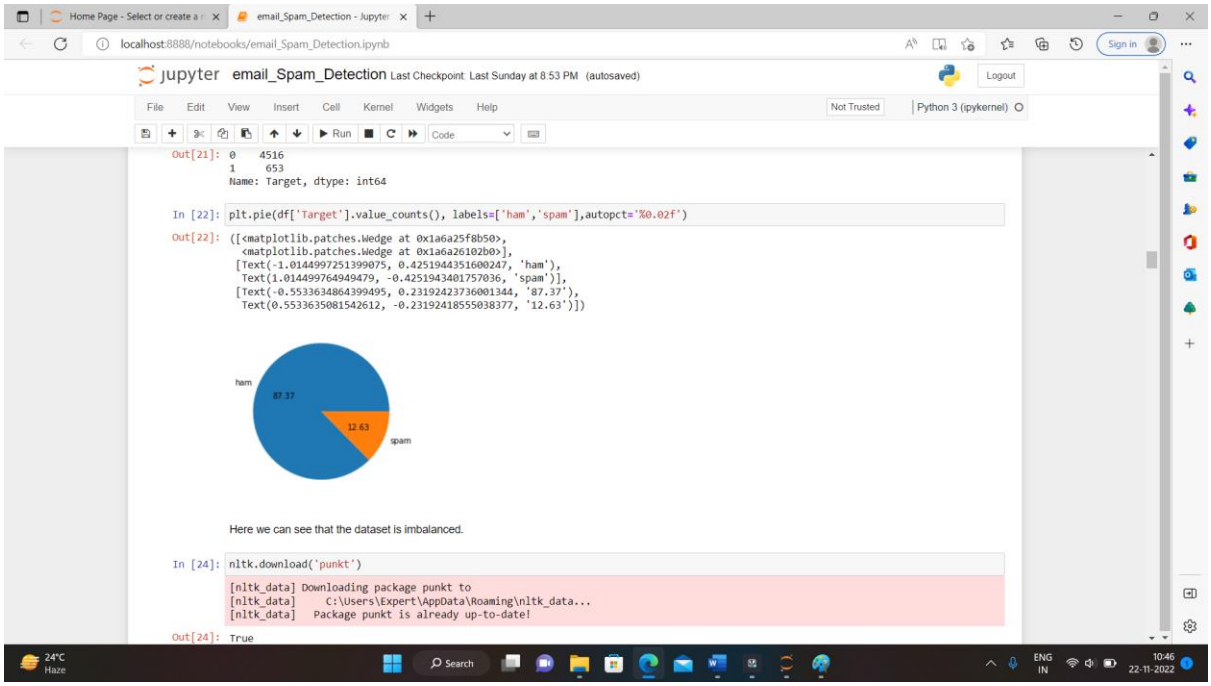
24°C Haze | Search | ENG IN | 10:42 22-11-2022

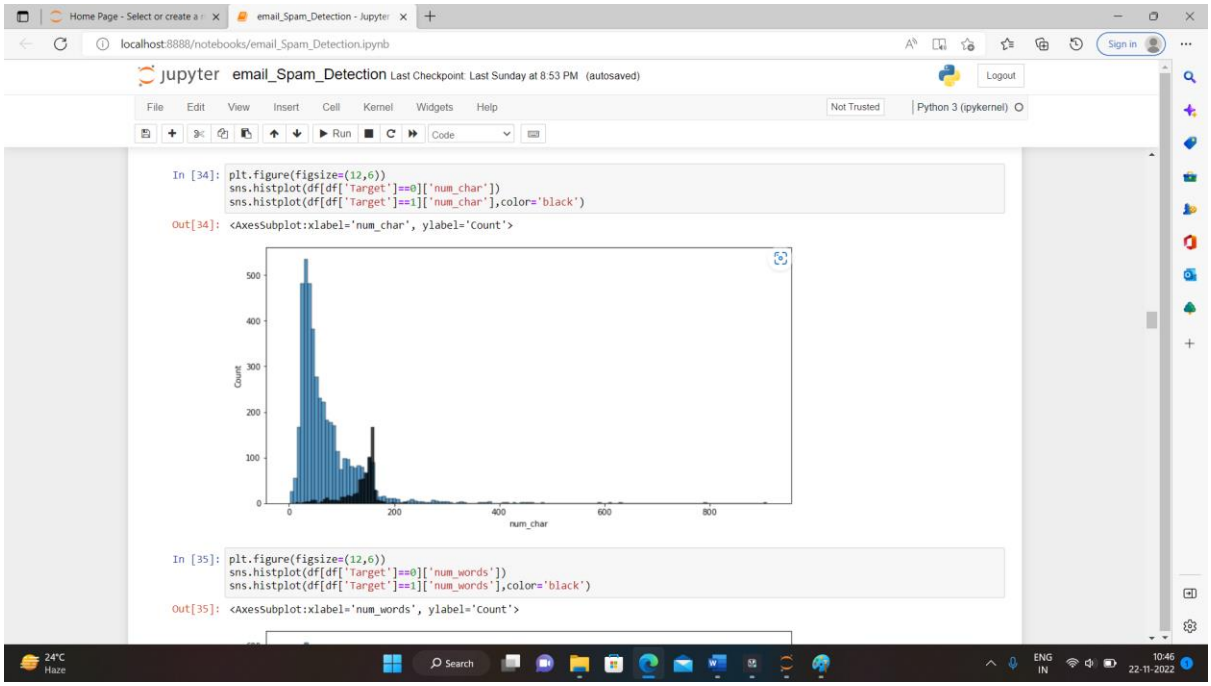
• **Key Metrics for success in solving problem under consideration**

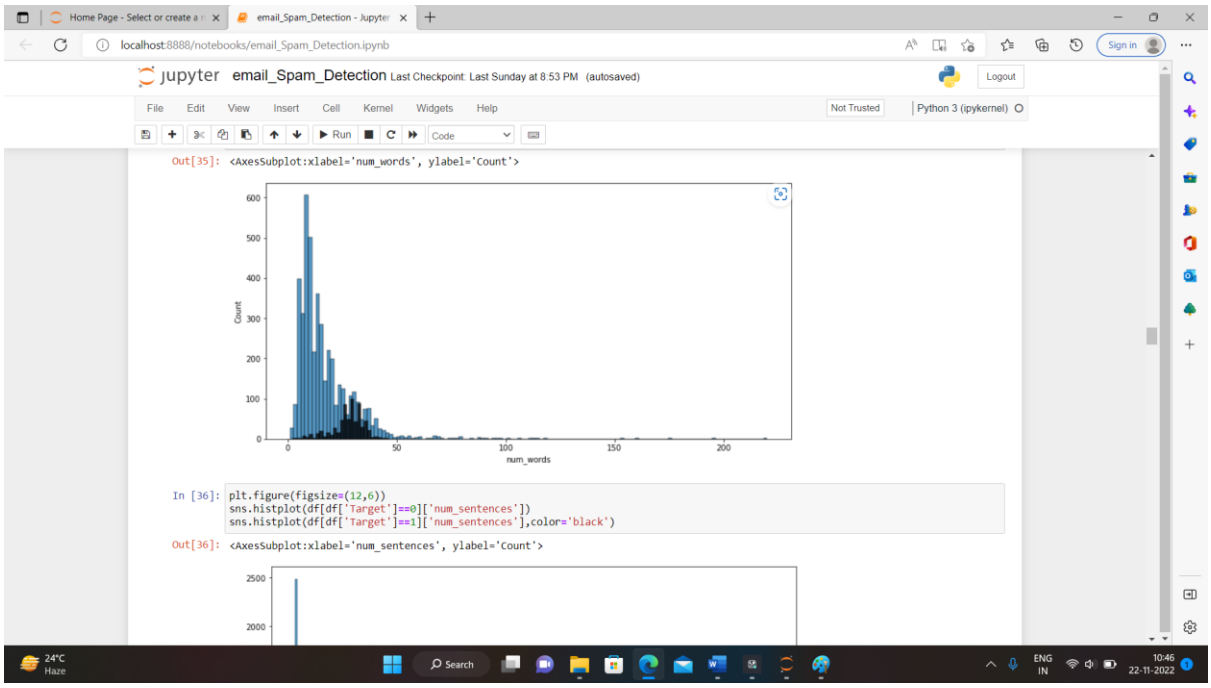
A company's units can use these dashboards to create milestones and monitor their progress by tracking all the most relevant metrics in one location.

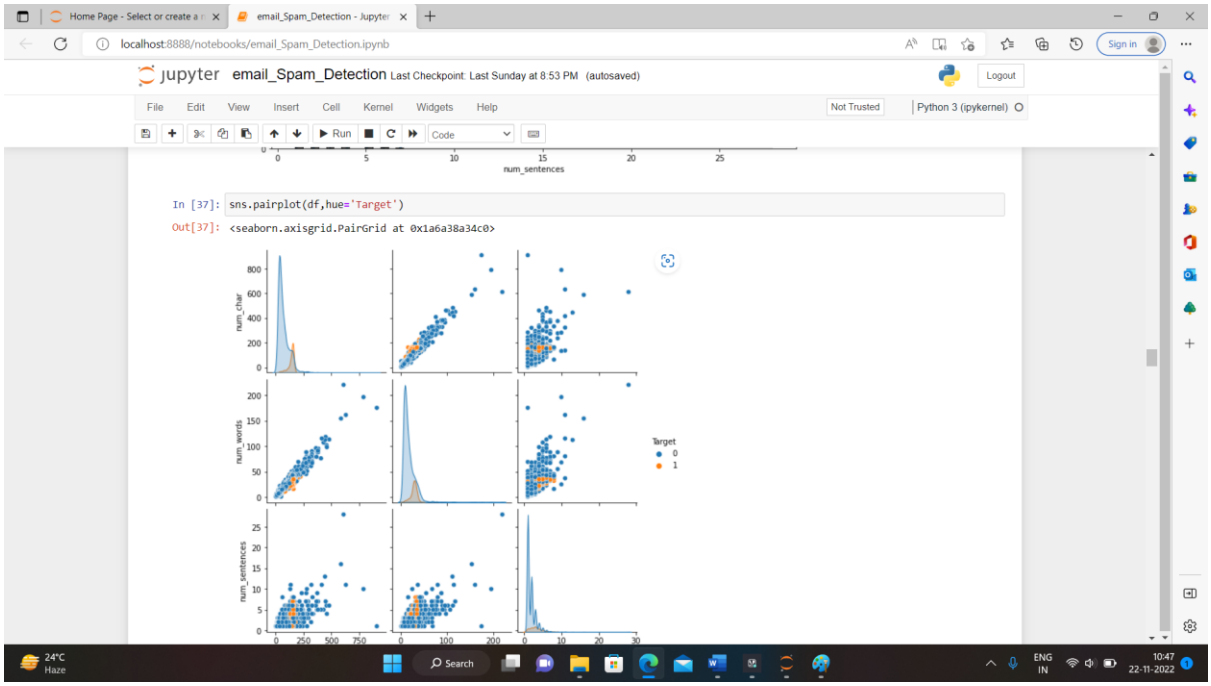
• **Visualizations**

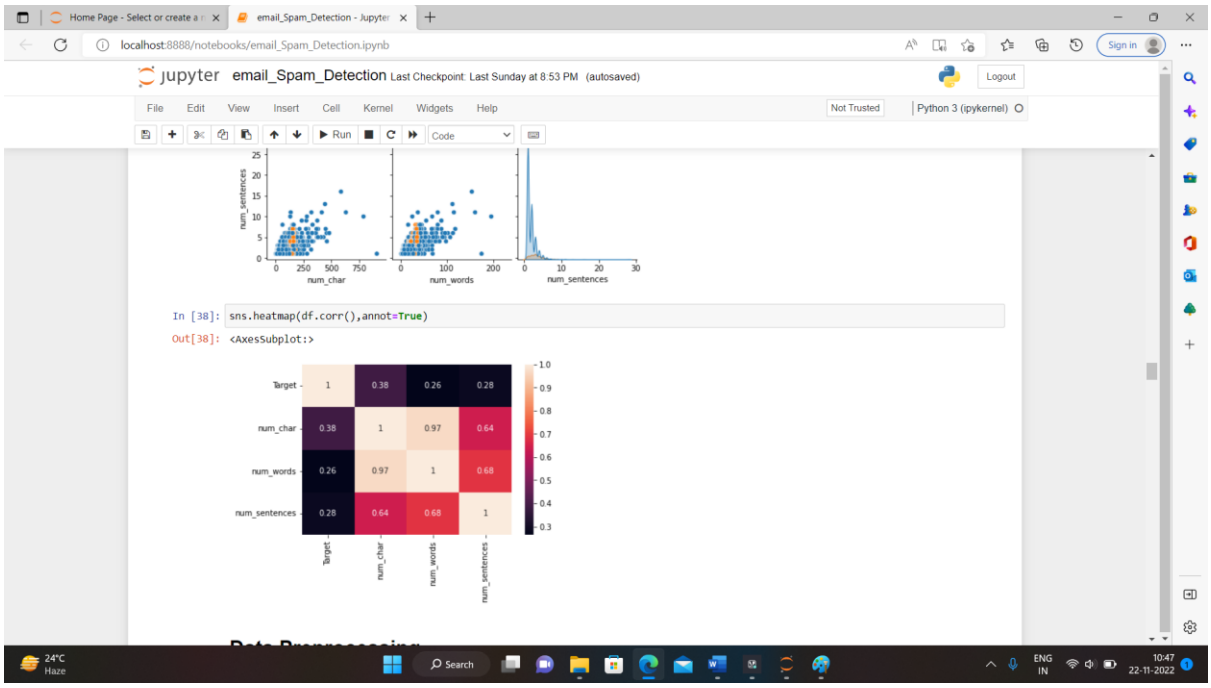





Home Page - Select or create a n
email_Spam_Detection - Jupyter
localhost:8888/notebooks/email_Spam_Detection.ipynb
jupyter email_Spam_Detection Last Checkpoint: Last Sunday at 8:53 PM (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
In [34]: plt.figure(figsize=(12,6))
sns.histplot(df[df['Target']==0]['num_char'])
sns.histplot(df[df['Target']==1]['num_char'],color='black')
Out[34]: <AxesSubplot:xlabel='num_char', ylabel='Count'>
Count
num_char
In [35]: plt.figure(figsize=(12,6))
sns.histplot(df[df['Target']==0]['num_words'])
sns.histplot(df[df['Target']==1]['num_words'],color='black')
Out[35]: <AxesSubplot:xlabel='num_words', ylabel='Count'>


Home Page - Select or create a n
email_Spam_Detection - Jupyter
localhost:8888/notebooks/email_Spam_Detection.ipynb
jupyter email_Spam_Detection Last Checkpoint: Last Sunday at 8:53 PM (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Out[35]: <AxesSubplot:xlabel='num_words', ylabel='Count'>
Count
num_words
In [36]: plt.figure(figsize=(12,6))
sns.histplot(df[df['Target']==0]['num_sentences'])
sns.histplot(df[df['Target']==1]['num_sentences'],color='black')
Out[36]: <AxesSubplot:xlabel='num_sentences', ylabel='Count'>


email_Spam_Detection Last Checkpoint: Last Sunday at 8:53 PM (autosaved)
In [37]: sns.pairplot(df,hue='Target')
Out[37]: <seaborn.axisgrid.PairGrid at 0x1a6a38a34c0>
num_char
num_words
num_sentences
Target
0
1


email_Spam_Detection Last Checkpoint: Last Sunday at 8:53 PM (autosaved)
In [38]: sns.heatmap(df.corr(),annot=True)
Out[38]: <AxesSubplot:>
Target
num_char
num_words
num_sentences
1
0.38
0.26
0.28
0.38
1
0.97
0.64
0.26
0.97
1
0.68
0.28
0.64
0.68
1

- **Observations**
- From the pie chart we observe that the dataset is imbalanced.
- Most of the numeric characters lies in between 0 to 200.
- Most number of words lies in between 0 to 50.
- Most number of sentences lies in between 0 to 5.
- Most of the columns shows linear relationships between each other's.

- **Interpretation Of Results**

  Most of the emails are ham that means non-spam and very less are spam, there are total 87.37 ham emails and 12.63 are spam emails.