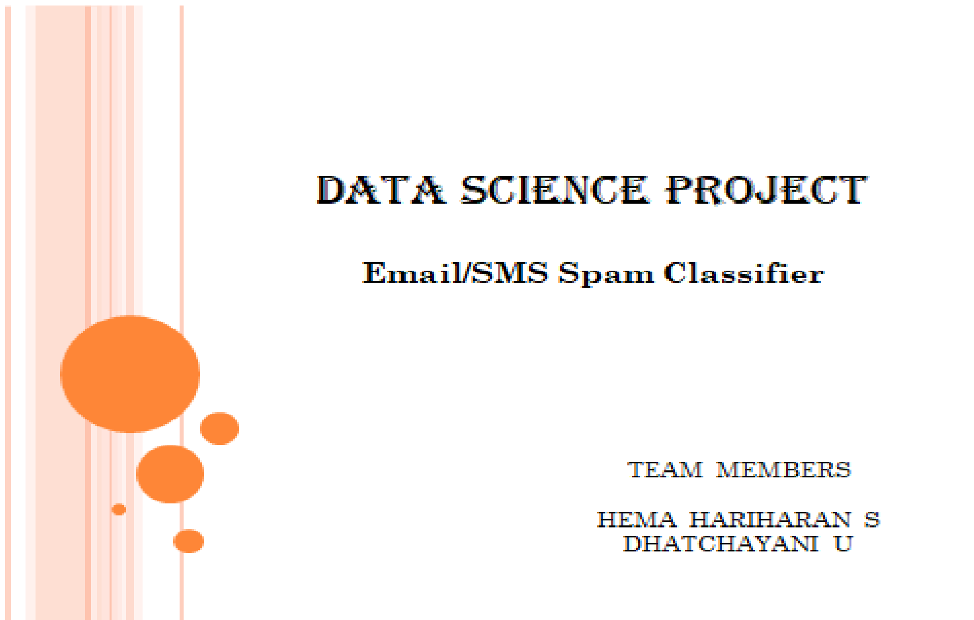
DATA SCIENCE PROJECT

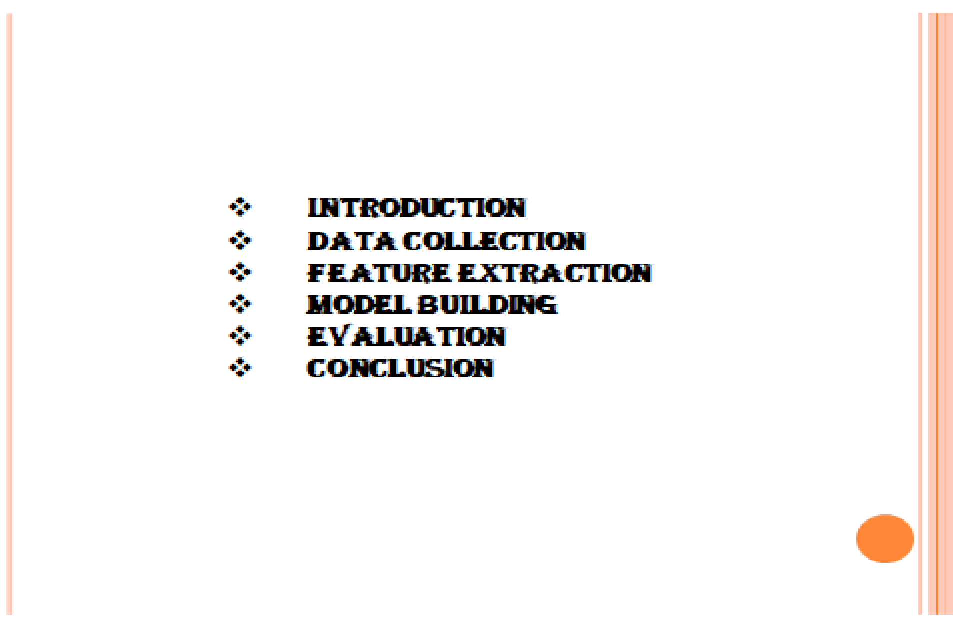
Email/SMS Spam Classifier

Abstract:

The Email/SMS Spam Classifier project aims to develop a robust and accurate system for identifying spam messages in both email and SMS communication channels. With the exponential growth of digital communication, the problem of spam messages has become increasingly prevalent and poses a significant challenge for individuals and organizations. This project leverages machine learning techniques and natural language processing to create a classifier capable of distinguishing between legitimate and spam messages, enabling users to effectively filter unwanted content. To accomplish this goal, the project utilizes a diverse dataset comprising a wide range of legitimate and spam messages. The dataset is preprocessed to extract relevant features from the text, including word frequency, character patterns, and structural attributes. Various machine learning algorithms, such as Naive Bayes, Support Vector Machines, and ensemble methods, are trained and evaluated using the dataset to identify the most effective approach for spam detection. Additionally, advanced natural language processing techniques are employed to enhance the performance of the classifier. These techniques include text tokenization, stemming, and feature engineering, which provide a deeper understanding of the content and context of messages. The project also explores the utilization of domain-specific knowledge, such as known spam keywords and sender reputation, to further improve the classifier's accuracy. The evaluation of the developed classifier involves rigorous testing on a separate test dataset, measuring key performance metrics such as precision, recall, and F1-score. The results are compared against existing spam detection systems to determine the effectiveness and efficiency of the proposed approach. The Email/SMS Spam Classifier project has the potential to significantly alleviate the problem of unwanted messages by providing an intelligent and adaptable solution. By accurately identifying spam messages, users can save time, protect themselves from malicious content, and maintain a clean and secure communication environment. Furthermore, the project's findings and insights can contribute to the ongoing research and development of anti-spam technologies, benefiting both individuals and organizations in the digital age.

PROJECT PRESENTATION:





Introduction:

Email and SMS spam has become a major issue in today's digital world. These unsolicited messages can be annoying, time-consuming, and even harmful. The good news is that data science can help us tackle this problem by building an effective spam classifier.

A spam classifier is a machine learning model that can distinguish between spam and legitimate messages. By analyzing various features of the message such as sender, subject, content, etc., the classifier can predict whether a message is spam or not.

Data Collection

The first step in building a spam classifier is to collect data. This data should include both spam and legitimate messages. There are many publicly available datasets that can be used for this purpose. Some popular ones include the Enron email dataset and the SMS Spam Collection dataset.

Once the data is collected, it needs to be preprocessed. This involves cleaning the data, removing any irrelevant information, and converting it into a format that can be used by the machine learning algorithm.

Feature Extraction

After preprocessing the data, the next step is to extract features from it. Features are the characteristics of the message that the classifier will use to make predictions. Some common features used in spam classification include the sender's email address, the subject line, the presence of certain keywords, and the length of the message. Feature extraction is an important step because it determines the quality of the classifier. The more relevant and informative features we extract, the better our classifier will perform.

Model Building

Once the features are extracted, we can start building our spam classifier model. There are many machine learning algorithms that can be used for this purpose, including Naive Bayes, Support Vector Machines (SVM), and Random Forests.

The model is trained on the preprocessed data with extracted features. During training, the algorithm learns to distinguish between spam and legitimate messages based on the provided features. Once the model is trained, it can be used to predict whether a new message is spam or not.

Evaluation

After building the model, it is important to evaluate its performance. This involves testing the model on a separate dataset that was not used during training. The performance of the model is measured using metrics such as accuracy, precision, recall, and F1 score.

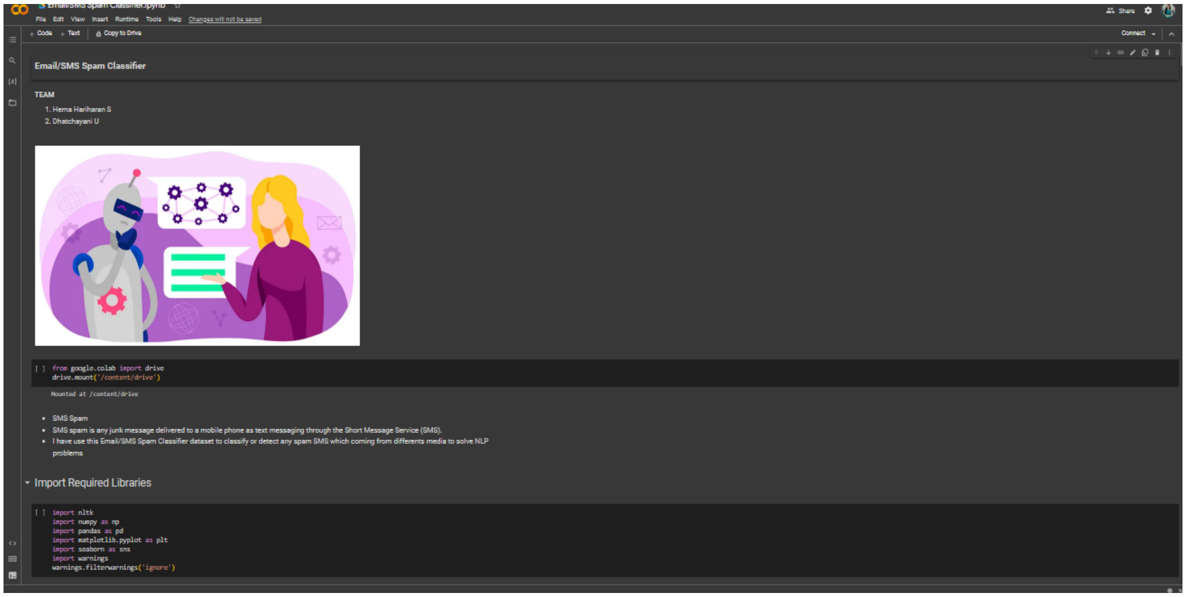
If the model's performance is not satisfactory, we can tweak the feature extraction process or try different machine learning algorithms until we get the desired results.

Conclusion

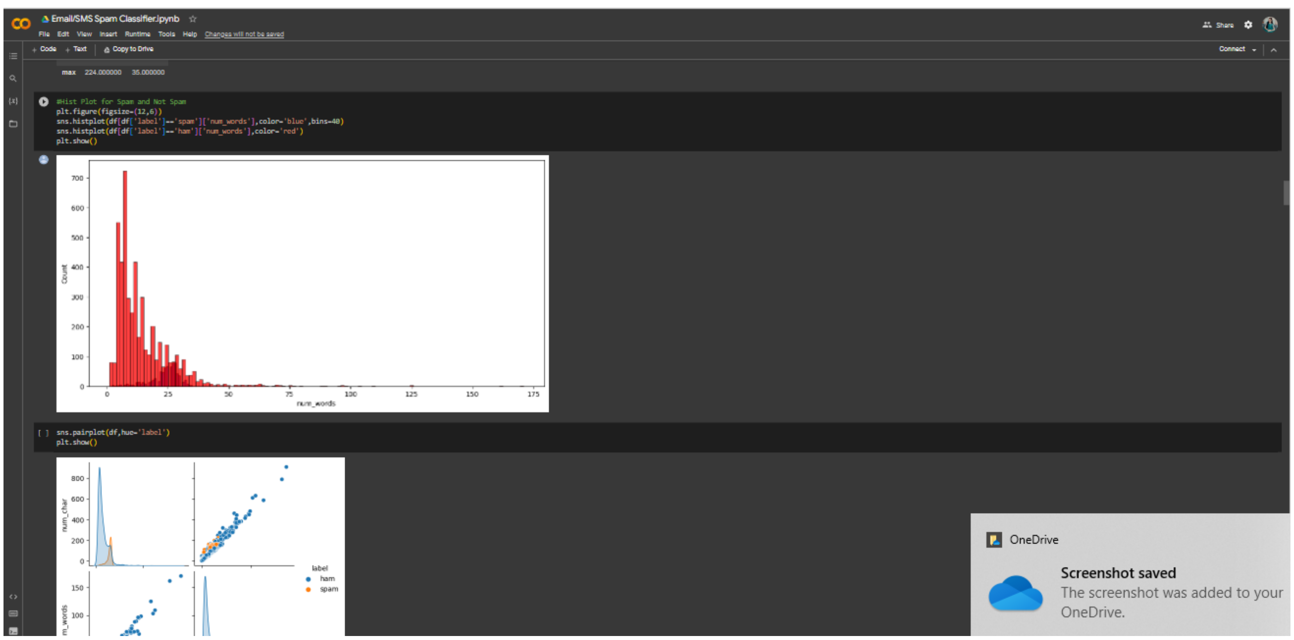
In conclusion, building an email and SMS spam classifier using data science is an effective way to tackle the problem of unsolicited messages. By collecting data, preprocessing it, extracting relevant features, building a model, and evaluating its performance, we can create a classifier that can accurately distinguish between spam and legitimate messages.

This technology can be used by individuals, businesses, and organizations to reduce the amount of time and resources spent on dealing with spam messages.

PROJECT SCREENSHOTS:




Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Share
Connect
(5572, 2)
#Check for Null Values
df.isnull().sum()
label 0
text 0
dtype: int64
df['label'].value_counts()/df.shape[0]*100
ham 86.593683
spam 13.406317
Name: label, dtype: float64
1.1 Feature Engineering
We Create Seperate features to extarct Some Information
1. Total No of Characters
2. Total No of Words
3. Total No of Sentences
#1. Total No of Char
df['num_char']=df['text'].apply(len)
#2. Total No of Words
df['num_words']=df['text'].apply(lambda x: len(str(x).split()))
#3. Total No of Sentences
nltk.download('punkt')
df['num_sen']=df['text'].apply(lambda x: len(nltk.sent_tokenize(x)))
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
df
label text num_char num_words
0 ham Go until jurong point, crazy.. Available only ... 111 20
1 ham Ok lar... Joking wif u oni... 29 6
2 spam Free entry in 2 a wkly comp to win FA Cup fina... 155 28
3 ham U dun say so early hor... U c already then say... 49 11
4 ham Nah I don't think he goes to usf, he lives aro... 61 13
... ...
5567 spam This is the 2nd time we have tried 2 contact u... 161 30
5568 ham Will ü b going to esplanade fr home? 37 8
5569 ham Pity, * was in mood for that. So...any other s... 57 10


Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Share
Connect
max 224.000000 35.000000
#Hist Plot for Spam and Not Spam
plt.figure(figsize=(12,6))
sns.histplot(df[df['label']=='spam']['num_words'],color='blue',bins=40)
sns.histplot(df[df['label']=='ham']['num_words'],color='red')
plt.show()
Count
num_words
sns.pairplot(df,hue='label')
plt.show()
num_char
label
ham
spam
OneDrive
Screenshot saved
The screenshot was added to your OneDrive.

Email/SMS Spam Classifier.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

```
i=df[df['num_char']>500].index
df.drop(i,axis=0,inplace=True)
```

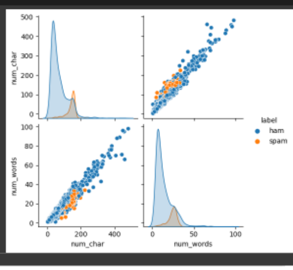
```
df=df.reset_index()
df.drop("index",inplace=True,axis=1)
```

```
df
```

	label	text	num_char	num_words
0	ham	Go until jurong point, crazy.. Available only ...	111	20
1	ham	Ok lar... Joking wif u oni...	29	6
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155	28
3	ham	U dun say so early hor... U c already then say...	49	11
4	ham	Nah I don't think he goes to usf, he lives aro...	61	13
...	...	...	...	...
5561	spam	This is the 2nd time we have tried 2 contact u...	161	30
5562	ham	Will Ì_ b going to esplanade fr home?	37	8
5563	ham	Pity, * was in mood for that. So...any other s...	57	10
5564	ham	The guy did some bitching but I acted like i'd...	125	26
5565	ham	Rofl. Its true to its name	26	6

5566 rows × 4 columns

```
sns.pairplot(df,hue='label')
plt.show()
```



Email/SMS Spam Classifier.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

```
#punctuation
string.punctuation
```

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
#Stopwards
stopwords.words('english')
```

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
 'who',
 'whom',
 'this',
 'that',
 "that'll",
 'these',
 'those',
 'am',
 'is',
 'are',
 'was',
 'were',
 'be',
 'been',
 'being',
 'have',
 'has',
 'had',
 'having',
 'do',
 'does',
```

Email/SMS Spam Classifier.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text Copy to Drive

```
 'weren',
 "weren't",
 'won',
 "won't",
 'wouldn',
 "wouldn't"]
```

Clean or Handle Text Data

1. Remove Web Links
2. Remove Numbers
3. Remove Emails

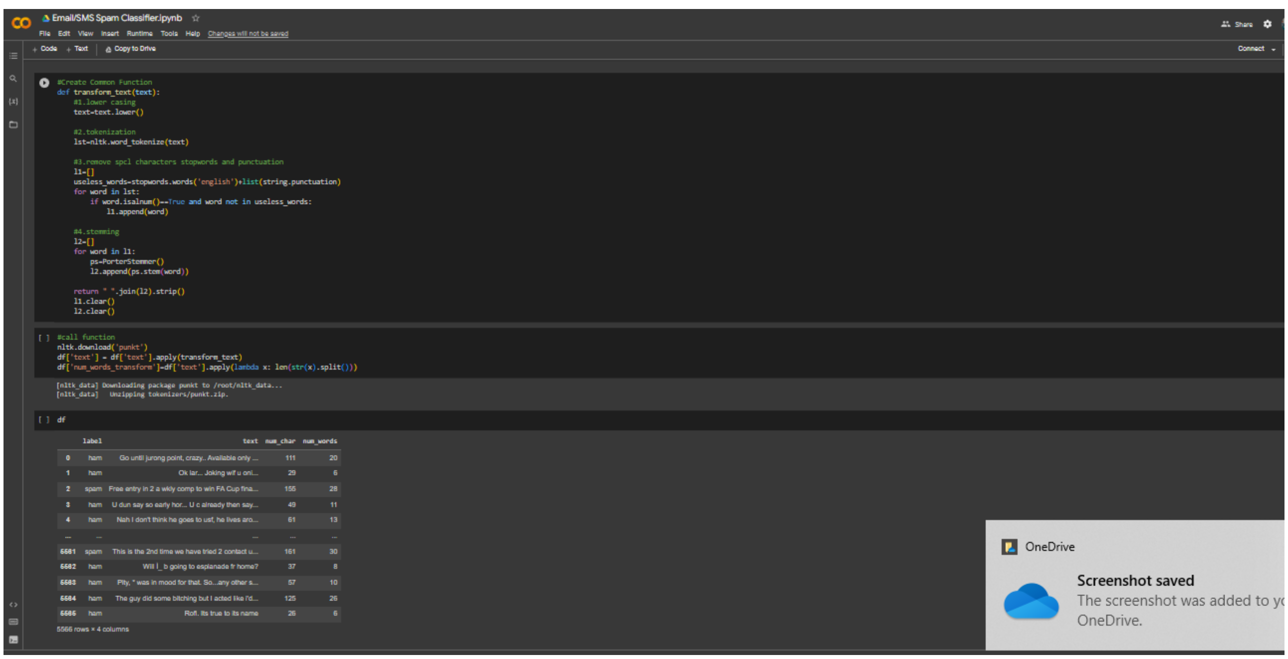
```python
#User Define Funtion for Text processing
def remove_website_links(text):
    no_website_links = text.replace(r"http\S+", "")
    return no_website_links

def remove_numbers(text):
    removed_numbers = text.replace(r'\d+','')
    return removed_numbers

def remove_emails(text):
    no_emails = text.replace(r"\S*@\S*\s?",'')
    return no_emails
```

```python
#Call Function
df['text'] = df['text'].apply(remove_website_links)
df['text'] = df['text'].apply(remove_numbers)
df['text'] = df['text'].apply(remove_emails)
```

Create Common function to Clean or process Text Data

1. Lower casing to avoids duplicates
2. Tokenization sentences
3. Remove Specials characters
4. Remove Stopwords
5. Remove punctuation
6. Stemming

```python
#Create Common Function
def transform_text(text):
    #1.lower casing
    text=text.lower()

    #2.tokenization
    lst=nltk.word_tokenize(text)

    #3.remove spcl characters stopwords and punctuation
    l1=[]
    useless_words=stopwords.words('english')+list(string.punctuation)
    for word in lst:
        if word.isalnum()==True and word not in useless_words:
```



Email/SMS Spam Classifier.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text Copy to Drive

```python
#Create Common Function
def transform_text(text):
    #1.lower casing
    text=text.lower()

    #2.tokenization
    lst=nltk.word_tokenize(text)

    #3.remove spcl characters stopwords and punctuation
    l1=[]
    useless_words=stopwords.words('english')+list(string.punctuation)
    for word in lst:
        if word.isalnum()==True and word not in useless_words:
            l1.append(word)

    #4.stemming
    l2=[]
    for word in l1:
        ps=PorterStemmer()
        l2.append(ps.stem(word))

    return " ".join(l2).strip()
    l1.clear()
    l2.clear()
```

```python
#call function
nltk.download('punkt')
df['text'] = df['text'].apply(transform_text)
df['num_words_transform']=df['text'].apply(lambda x: len(str(x).split()))
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```python
df
```

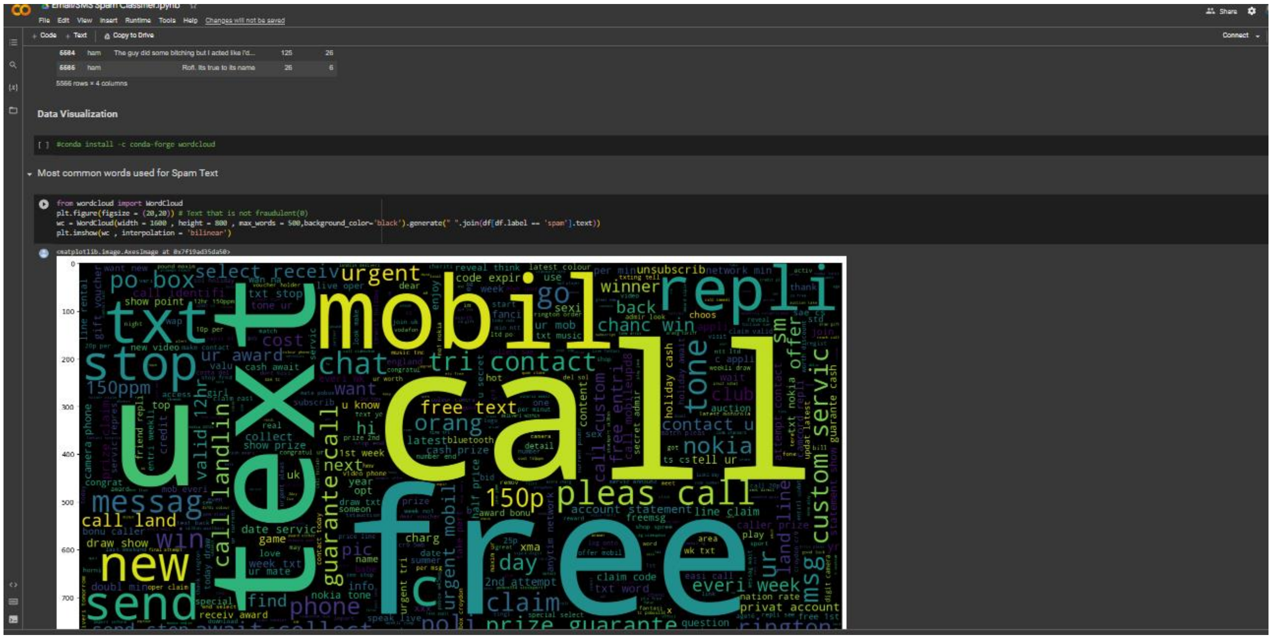
	label	text	num_char	num_words
0	ham	Go until jurong point, crazy.. Available only ...	111	20
1	ham	Ok lar... Joking wif u oni...	29	6
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155	28
3	ham	U dun say so early hor... U c already then say...	49	11
4	ham	Nah I don't think he goes to usf, he lives aro...	61	13
...	...	...	...	...
5561	spam	This is the 2nd time we have tried 2 contact u...	161	30
5562	ham	Will Ì_ b going to esplanade fr home?	37	8
5563	ham	Pity, * was in mood for that. So...any other s...	57	10
5564	ham	The guy did some bitching but I acted like i'd...	125	26
5565	ham	Rofl. Its true to its name	26	6

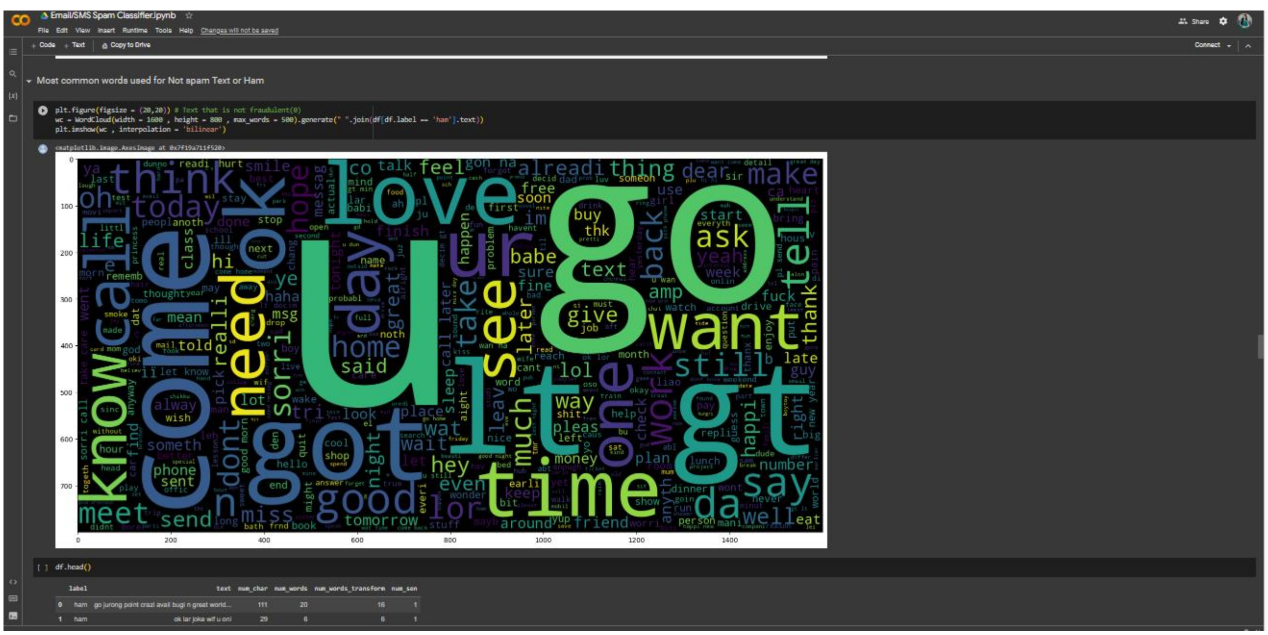
5566 rows × 4 columns

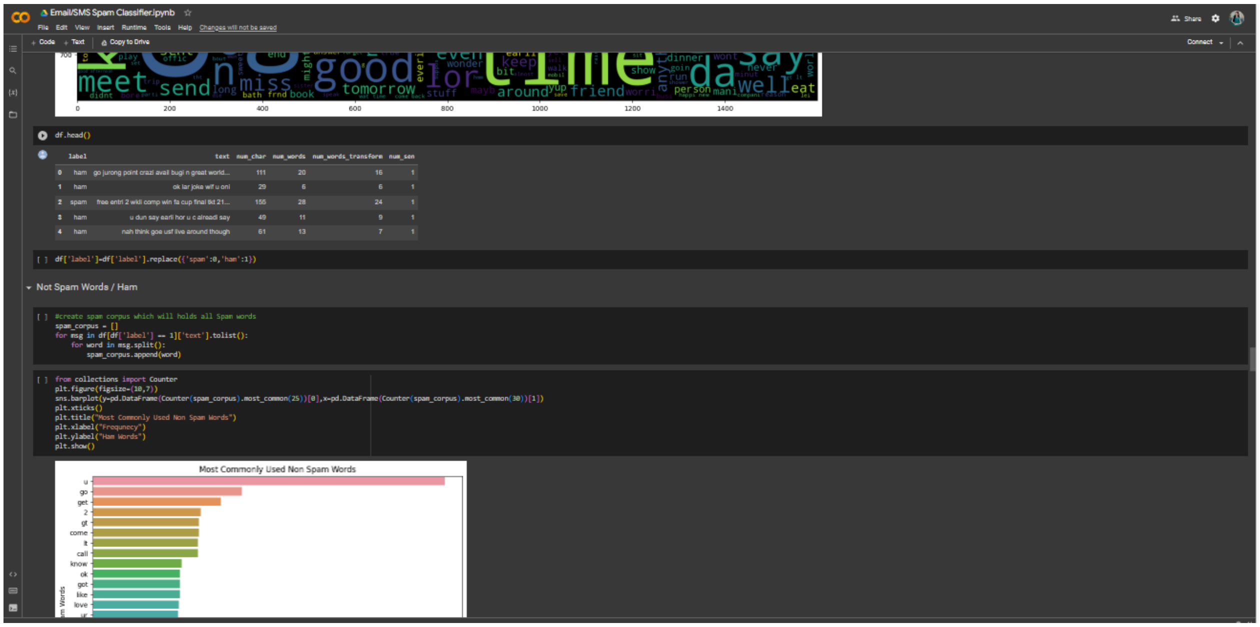
OneDrive

Screenshot saved

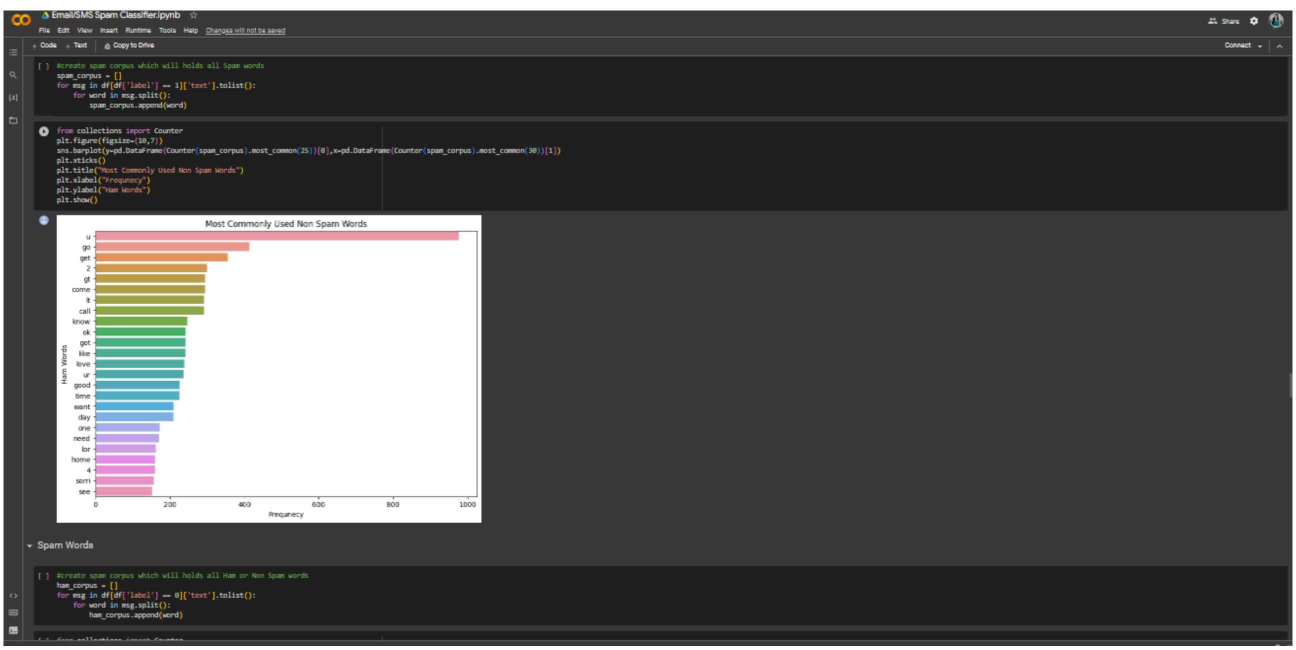
The screenshot was added to y OneDrive.


Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
5584 ham The guy did some bitching but I acted like i'd... 125 26
5585 ham Rofl. Its true to its name 26 6
5566 rows × 4 columns
Data Visualization
[ ] #conda install -c conda-forge wordcloud
Most common words used for Spam Text
from wordcloud import WordCloud
plt.figure(figsize = (20,20)) # Text that is not fraudulent(0)
wc = WordCloud(width = 1600 , height = 800 , max_words = 500,background_color='black').generate(" ".join(df[df.label == 'spam'].text))
plt.imshow(wc , interpolation = 'bilinear')
<matplotlib.image.AxesImage at 0x7f19ad35da50>

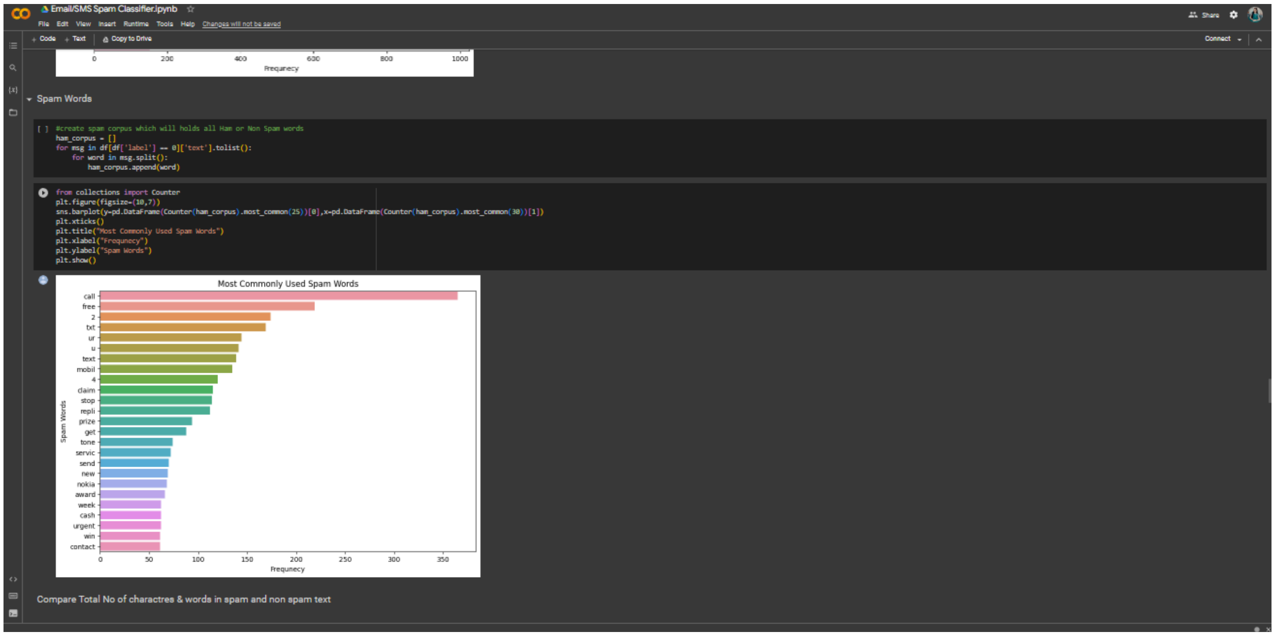

Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Most common words used for Not spam Text or Ham
plt.figure(figsize = (20,20)) # Text that is not fraudulent(0)
wc = WordCloud(width = 1600 , height = 800 , max_words = 500).generate(" ".join(df[df.label == 'ham'].text))
plt.imshow(wc , interpolation = 'bilinear')
<matplotlib.image.AxesImage at 0x7f19a711f520>
[ ] df.head()
label text num_char num_words num_words_transform num_sen
0 ham go jurong point crazi avail bugi n great world... 111 20 16 1
1 ham ok lar joke wif u oni 29 6 6 1

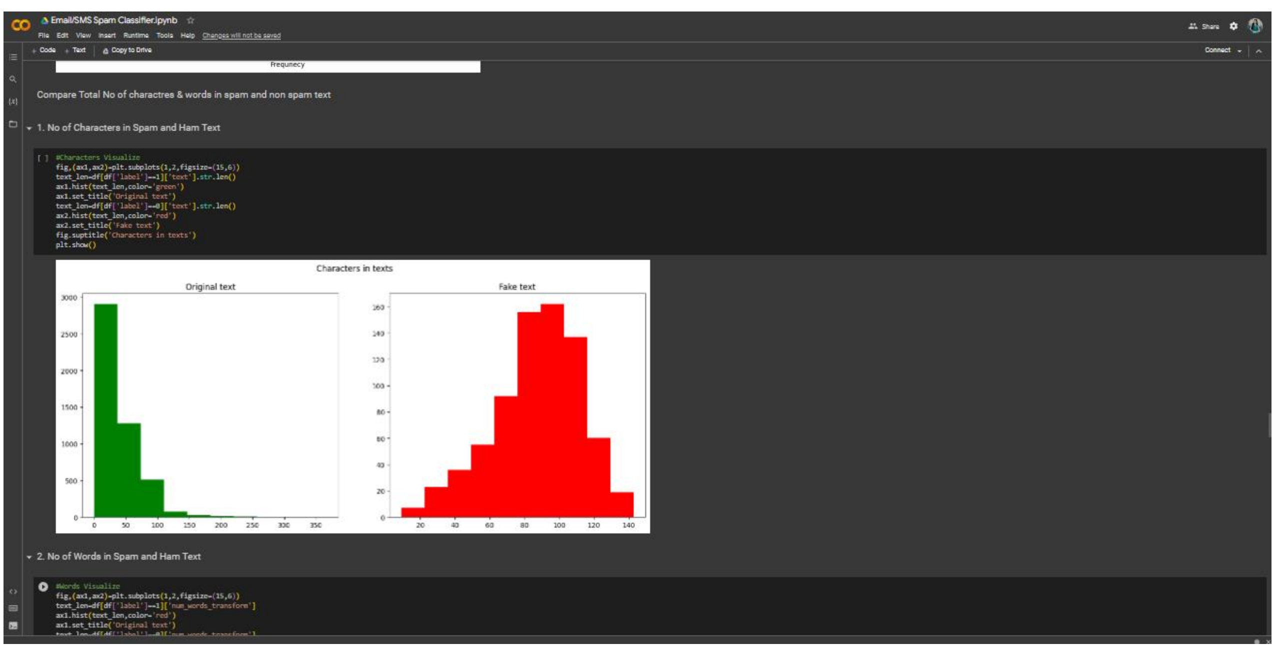


Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Code Text Copy to Drive
Share
Connect
df.head()
label text num_char num_words num_words_transform num_sen
0 ham go jurong point crazi avail bugi n great world... 111 20 16 1
1 ham ok lar joke wif u oni 29 6 6 1
2 spam free entri 2 wkli comp win fa cup final tkt 21... 155 28 24 1
3 ham u dun say earli hor u c alreadi say 49 11 9 1
4 ham nah think goe usf live around though 61 13 7 1
df['label']=df['label'].replace({'spam':0,'ham':1})
Not Spam Words / Ham
#create spam corpus which will holds all Spam words
spam_corpus = []
for msg in df[df['label'] == 1]['text'].tolist():
for word in msg.split():
spam_corpus.append(word)
from collections import Counter
plt.figure(figsize=(10,7))
sns.barplot(y=pd.DataFrame(Counter(spam_corpus).most_common(25))[0],x=pd.DataFrame(Counter(spam_corpus).most_common(30))[1])
plt.xticks()
plt.title("Most Commonly Used Non Spam Words")
plt.xlabel("Frequnecy")
plt.ylabel("Ham Words")
plt.show()
Most Commonly Used Non Spam Words



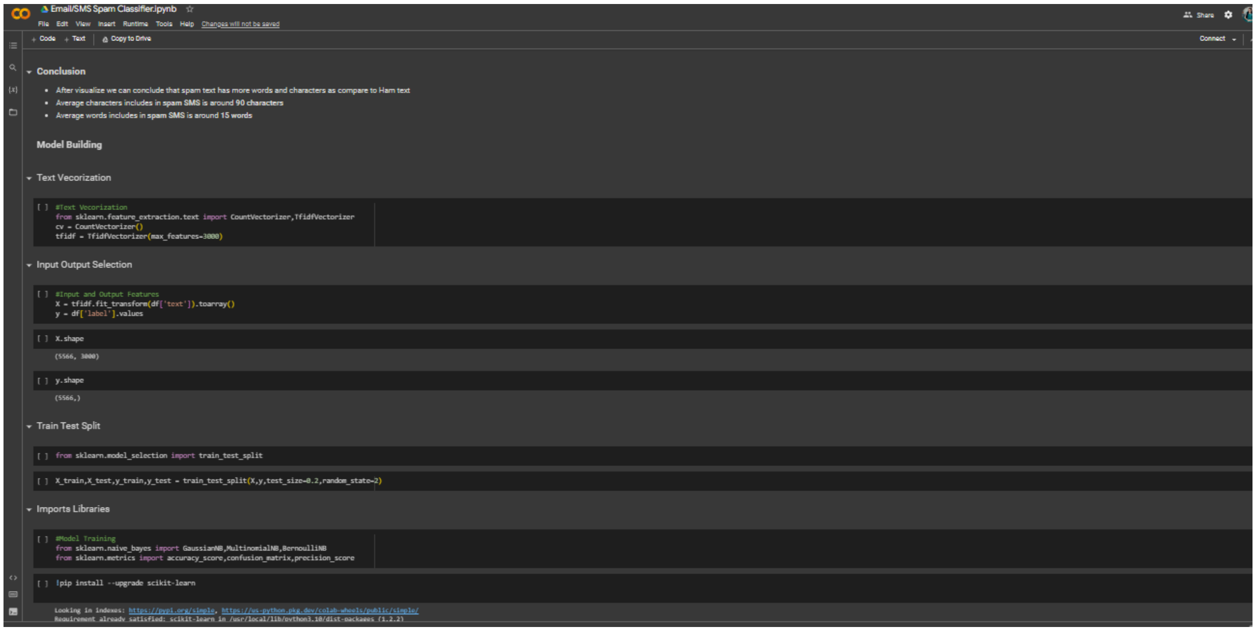
Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Code Text Copy to Drive
Share
Connect
#create spam corpus which will holds all Spam words
spam_corpus = []
for msg in df[df['label'] == 1]['text'].tolist():
for word in msg.split():
spam_corpus.append(word)
from collections import Counter
plt.figure(figsize=(10,7))
sns.barplot(y=pd.DataFrame(Counter(spam_corpus).most_common(25))[0],x=pd.DataFrame(Counter(spam_corpus).most_common(30))[1])
plt.xticks()
plt.title("Most Commonly Used Non Spam Words")
plt.xlabel("Frequnecy")
plt.ylabel("Ham Words")
plt.show()
Most Commonly Used Non Spam Words
u
go
get
2
gt
come
lt
call
know
ok
got
like
love
ur
good
time
want
day
one
need
lor
home
4
sorri
see
Ham Words
0
200
400
600
800
1000
Frequnecy
Spam Words
#create spam corpus which will holds all Ham or Non Spam words
ham_corpus = []
for msg in df[df['label'] == 0]['text'].tolist():
for word in msg.split():
ham_corpus.append(word)


Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Code Text Copy to Drive
Share
Connect
Frequnecy
Spam Words
#create spam corpus which will holds all Ham or Non Spam words
ham_corpus = []
for msg in df[df['label'] == 0]['text'].tolist():
for word in msg.split():
ham_corpus.append(word)
from collections import Counter
plt.figure(figsize=(10,7))
sns.barplot(y=pd.DataFrame(Counter(ham_corpus).most_common(25))[0],x=pd.DataFrame(Counter(ham_corpus).most_common(30))[1])
plt.xticks()
plt.title("Most Commonly Used Spam Words")
plt.xlabel("Frequnecy")
plt.ylabel("Spam Words")
plt.show()
Most Commonly Used Spam Words
call
free
2
txt
ur
u
text
mobil
4
claim
stop
repli
prize
get
tone
servic
send
new
nokia
award
week
cash
urgent
win
contact
Spam Words
Frequnecy
Compare Total No of charactres & words in spam and non spam text


Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Code Text Copy to Drive
Share
Connect
Frequnecy
Compare Total No of charactres & words in spam and non spam text
1. No of Characters in Spam and Ham Text
#Characters Visualize
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,6))
text_len=df[df['label']==1]['text'].str.len()
ax1.hist(text_len,color='green')
ax1.set_title('Original text')
text_len=df[df['label']==0]['text'].str.len()
ax2.hist(text_len,color='red')
ax2.set_title('Fake text')
fig.suptitle('Characters in texts')
plt.show()
Characters in texts
Original text
Fake text
2. No of Words in Spam and Ham Text
#Words Visualize
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,6))
text_len=df[df['label']==1]['num_words_transform']
ax1.hist(text_len,color='red')
ax1.set_title('Original text')


Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Share
Connect
2. No of Words in Spam and Ham Text
#Words Visualize
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,6))
text_len=df[df['label']==1]['num_words_transform']
ax1.hist(text_len,color='red')
ax1.set_title('Original text')
text_len=df[df['label']==0]['num_words_transform']
ax2.hist(text_len,color='green')
ax2.set_title('Fake text')
fig.suptitle('Words in texts')
plt.show()
Words in texts
Original text
Fake text
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,6))
word=df[df['label']==1]['num_words_transform']
sns.distplot(word.map(lambda x: np.mean(x)),ax=ax1,color='red')
ax1.set_title('Original text')
word=df[df['label']==0]['num_words_transform']
sns.distplot(word.map(lambda x: np.mean(x)),ax=ax2,color='green')
ax2.set_title('Fake text')
fig.suptitle('Average word length in each text')


Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Share
Connect
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,6))
word=df[df['label']==1]['num_words_transform']
sns.distplot(word.map(lambda x: np.mean(x)),ax=ax1,color='red')
ax1.set_title('Original text')
word=df[df['label']==0]['num_words_transform']
sns.distplot(word.map(lambda x: np.mean(x)),ax=ax2,color='green')
ax2.set_title('Fake text')
fig.suptitle('Average word length in each text')
Text(0.5, 0.98, 'Average word length in each text')
Average word length in each text
Original text
Fake text
Density
num_words_transform
Conclusion
After visualize we can conclude that spam text has more words and characters as compare to Ham text
Average characters includes in spam SMS is around 90 characters
Average words includes in spam SMS is around 15 words
Model Building
Text Vecorization



Email/SMS Spam Classifier.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text | Copy to Drive

Connect

Conclusion

- After visualize we can conclude that spam text has more words and characters as compare to Ham text
- Average characters includes in spam SMS is around 90 characters
- Average words includes in spam SMS is around 15 words

Model Building

Text Vecorization

```
#Text Vecorization
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
```

Input Output Selection

```
#Input and Output Features
X = tfidf.fit_transform(df['text']).toarray()
y = df['label'].values
```

```
X.shape
```

```
(5566, 3000)
```

```
y.shape
```

```
(5566,)
```

Train Test Split

```
from sklearn.model_selection import train_test_split
```

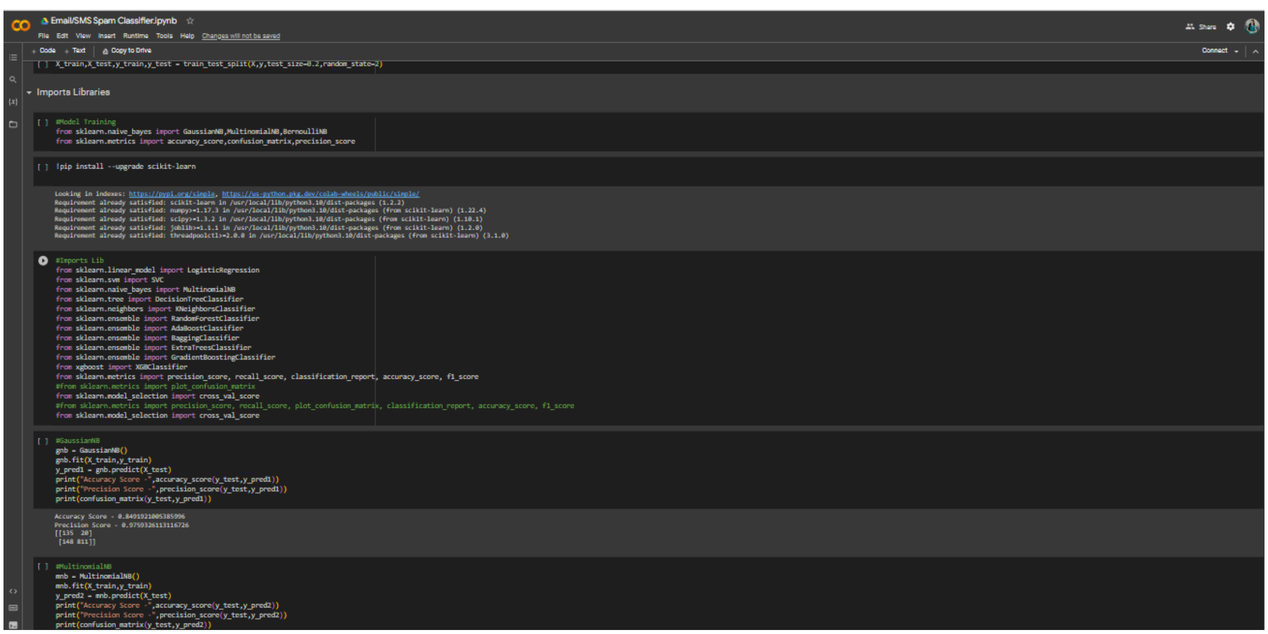
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

Imports Libraries

```
#Model Training
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

```
!pip install --upgrade scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
```



Email/SMS Spam Classifier.ipynb

File Edit View Insert Runtime Tools Help Changes will not be saved

+ Code + Text | Copy to Drive

Connect

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

Imports Libraries

```
#Model Training
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
```

```
!pip install --upgrade scikit-learn
```

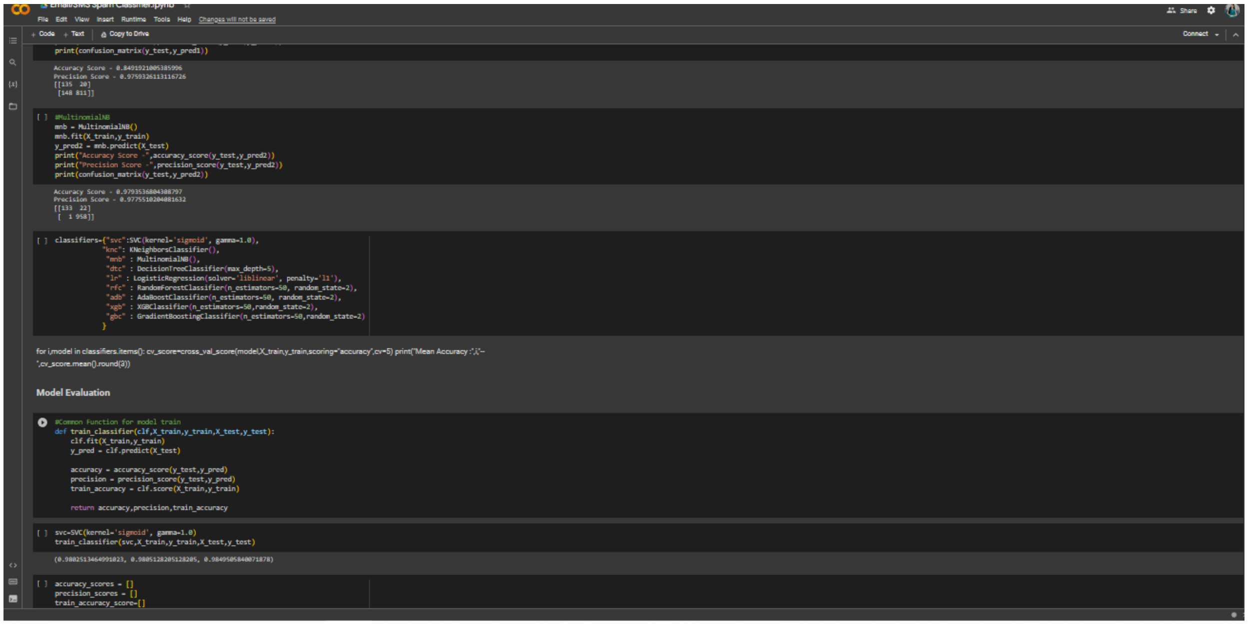
```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.22.4)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.1.0)
```

```
#Imports Lib
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.metrics import precision_score, recall_score, classification_report, accuracy_score, f1_score
#from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import cross_val_score
#from sklearn.metrics import precision_score, recall_score, plot_confusion_matrix, classification_report, accuracy_score, f1_score
from sklearn.model_selection import cross_val_score
```

```
#GaussianNB
gnb = GaussianNB()
gnb.fit(X_train,y_train)
y_pred1 = gnb.predict(X_test)
print("Accuracy Score -",accuracy_score(y_test,y_pred1))
print("Precision Score -",precision_score(y_test,y_pred1))
print(confusion_matrix(y_test,y_pred1))
```

```
Accuracy Score - 0.8491921005385996
Precision Score - 0.9759326113116726
[[135  20]
 [148 811]]
```

```
#MultinomialNB
mnb = MultinomialNB()
mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print("Accuracy Score -",accuracy_score(y_test,y_pred2))
print("Precision Score -",precision_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
```




```
print(confusion_matrix(y_test,y_pred1))
```

```
Accuracy Score - 0.8491921005385996
Precision Score - 0.9759326113116726
[[135  20]
 [148 811]]
```

```
#MultinomialNB
mnb = MultinomialNB()
mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print("Accuracy Score -",accuracy_score(y_test,y_pred2))
print("Precision Score -",precision_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))
```

```
Accuracy Score - 0.9793536804308797
Precision Score - 0.9775510204081632
[[133  22]
 [  1 958]]
```

```
classifiers={"svc":SVC(kernel='sigmoid', gamma=1.0),
             "knc": KNeighborsClassifier(),
              "mnb" : MultinomialNB(),
              "dtc" : DecisionTreeClassifier(max_depth=5),
              "lr" : LogisticRegression(solver='liblinear', penalty='l1'),
              "rfc" : RandomForestClassifier(n_estimators=50, random_state=2),
              "adb" : AdaBoostClassifier(n_estimators=50, random_state=2),
              "xgb" : XGBClassifier(n_estimators=50,random_state=2),
              "gbc" : GradientBoostingClassifier(n_estimators=50,random_state=2)
             }
```

for i,model in classifiers.items(): cv_score=cross_val_score(model,X_train,y_train,scoring="accuracy",cv=5) print("Mean Accuracy :",i,"--",cv_score.mean().round(3))

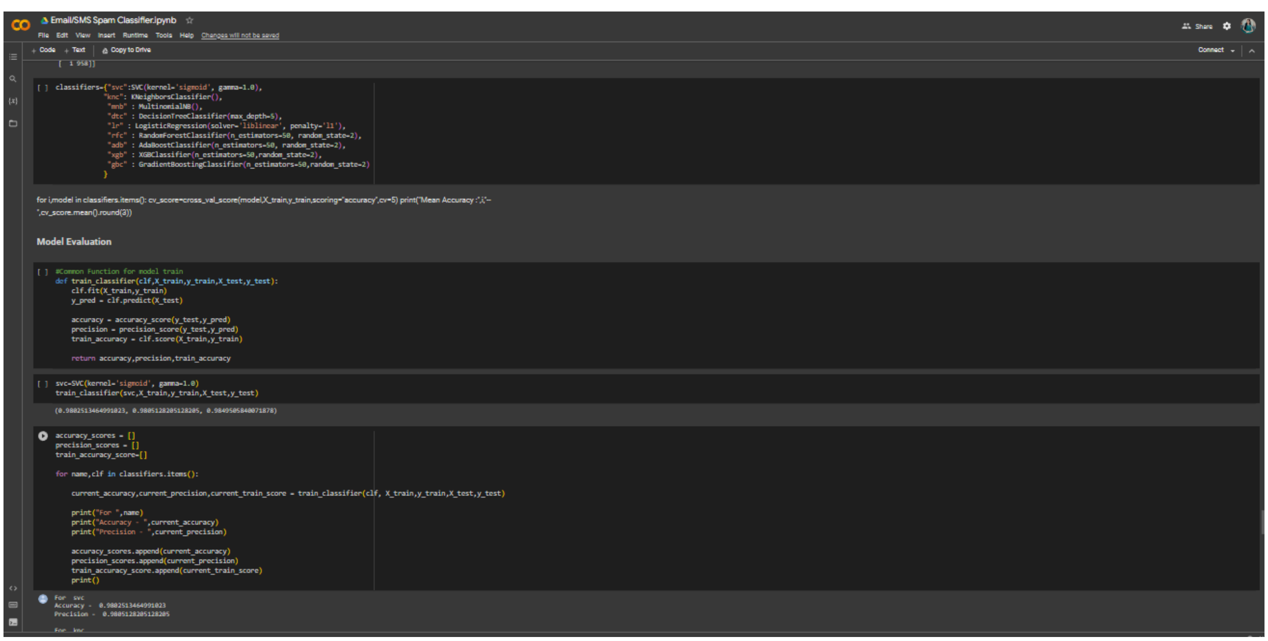
Model Evaluation

```
#Common Function for model train
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)
    train_accuracy = clf.score(X_train,y_train)

    return accuracy,precision,train_accuracy
```

```
svc=SVC(kernel='sigmoid', gamma=1.0)
train_classifier(svc,X_train,y_train,X_test,y_test)
```

```
(0.9802513464991023, 0.9805128205128205, 0.9849505840071878)
```

```
accuracy_scores = []
precision_scores = []
train_accuracy_score=[]
```




```
 [  1 958]]
```

```
classifiers={"svc":SVC(kernel='sigmoid', gamma=1.0),
             "knc": KNeighborsClassifier(),
              "mnb" : MultinomialNB(),
              "dtc" : DecisionTreeClassifier(max_depth=5),
              "lr" : LogisticRegression(solver='liblinear', penalty='l1'),
              "rfc" : RandomForestClassifier(n_estimators=50, random_state=2),
              "adb" : AdaBoostClassifier(n_estimators=50, random_state=2),
              "xgb" : XGBClassifier(n_estimators=50,random_state=2),
              "gbc" : GradientBoostingClassifier(n_estimators=50,random_state=2)
             }
```

for i,model in classifiers.items(): cv_score=cross_val_score(model,X_train,y_train,scoring="accuracy",cv=5) print("Mean Accuracy :",i,"--",cv_score.mean().round(3))

Model Evaluation

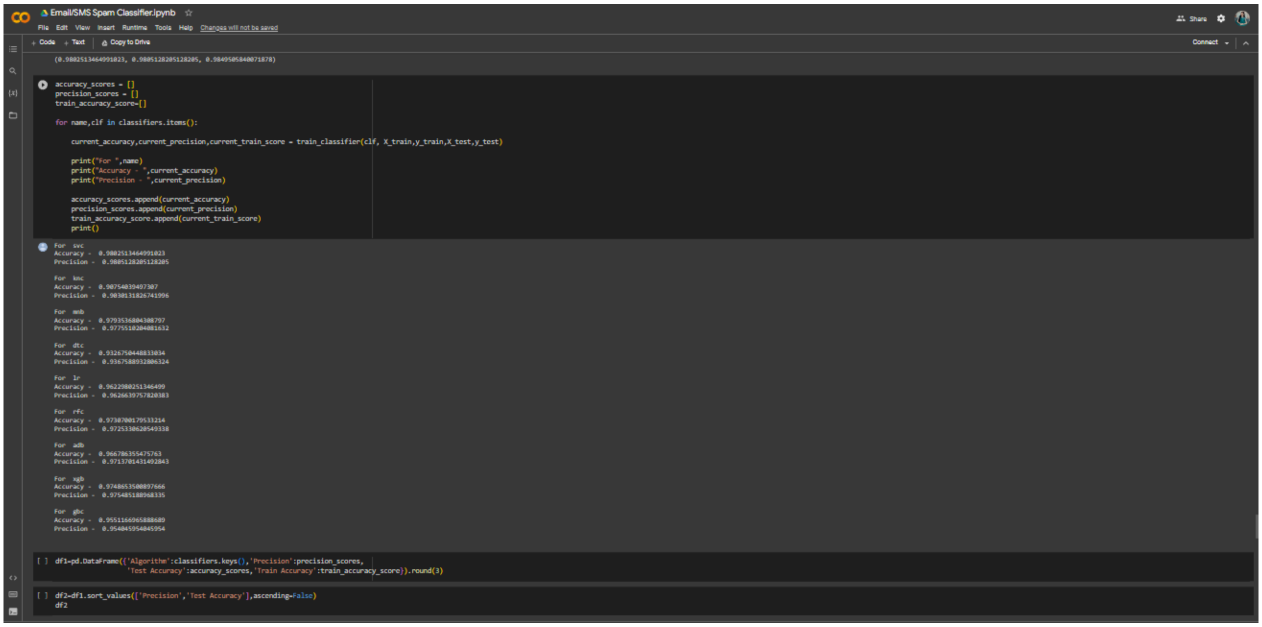
```
#Common Function for model train
def train_classifier(clf,X_train,y_train,X_test,y_test):
    clf.fit(X_train,y_train)
    y_pred = clf.predict(X_test)

    accuracy = accuracy_score(y_test,y_pred)
    precision = precision_score(y_test,y_pred)
    train_accuracy = clf.score(X_train,y_train)

    return accuracy,precision,train_accuracy
```

```
svc=SVC(kernel='sigmoid', gamma=1.0)
train_classifier(svc,X_train,y_train,X_test,y_test)
```

```
(0.9802513464991023, 0.9805128205128205, 0.9849505840071878)
```

```
accuracy_scores = []
precision_scores = []
train_accuracy_score=[]

for name,clf in classifiers.items():

    current_accuracy,current_precision,current_train_score = 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)
    train_accuracy_score.append(current_train_score)
    print()
```

```
For  svc
Accuracy -  0.9802513464991023
Precision -  0.9805128205128205
```



Email/SMS Spam Classifier.ipynb

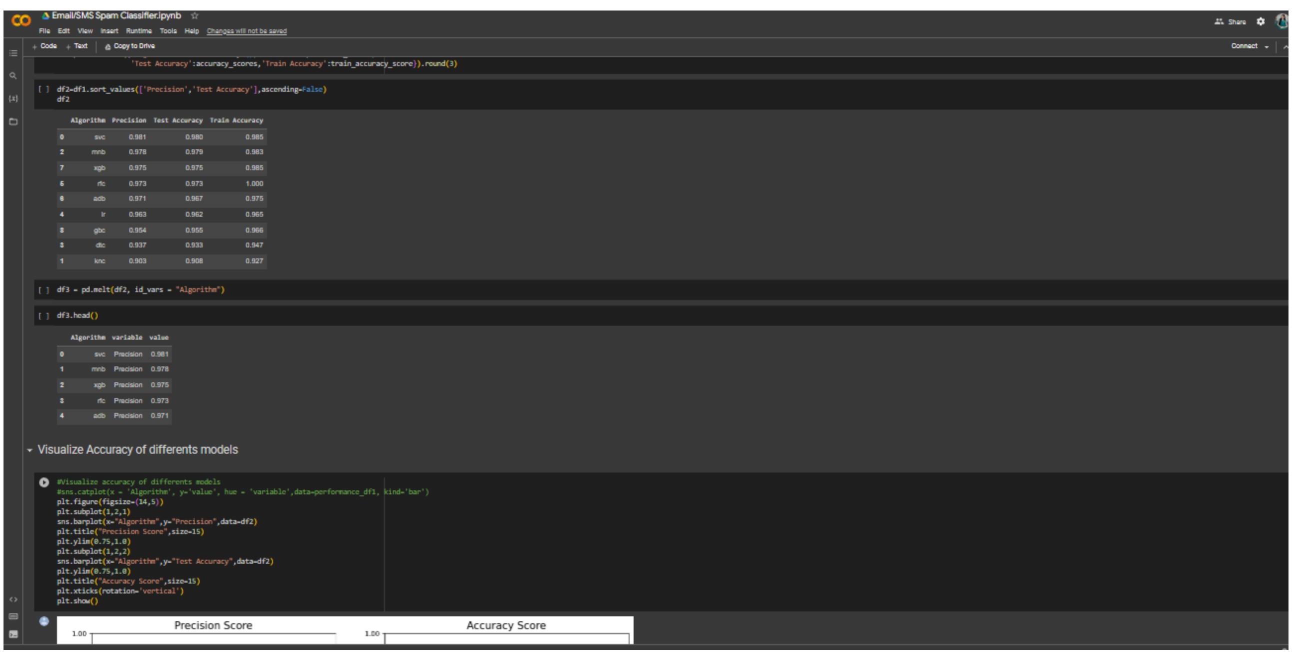
```
(0.9882513464991023, 0.9885128205128205, 0.9849505840071878)
```

```python
accuracy_scores = []
precision_scores = []
train_accuracy_score=[]

for name,clf in classifiers.items():

    current_accuracy,current_precision,current_train_score = 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)
    train_accuracy_score.append(current_train_score)
    print()
```

```
For  svc
Accuracy -  0.9882513464991023
Precision -  0.9885128205128205

For  knc
Accuracy -  0.90754039497307
Precision -  0.9038131826741996

For  mnb
Accuracy -  0.9793536804308797
Precision -  0.9775510204081632

For  dtc
Accuracy -  0.9326750448833034
Precision -  0.9367588932806324

For  lr
Accuracy -  0.9622980251346499
Precision -  0.9626639757820383

For  rfc
Accuracy -  0.9738700179533214
Precision -  0.9725330620549338

For  adb
Accuracy -  0.966786355475763
Precision -  0.9713701431492843

For  xgb
Accuracy -  0.9748653500897666
Precision -  0.975485188968335

For  gbc
Accuracy -  0.9551166965888689
Precision -  0.954045954045954
```

```python
df1=pd.DataFrame({'Algorithm':classifiers.keys(),'Precision':precision_scores,
                  'Test Accuracy':accuracy_scores,'Train Accuracy':train_accuracy_score}).round(3)
```

```python
df2=df1.sort_values(['Precision','Test Accuracy'],ascending=False)
df2
```



Email/SMS Spam Classifier.ipynb

```python
df2=df1.sort_values(['Precision','Test Accuracy'],ascending=False)
df2
```

	Algorithm	Precision	Test Accuracy	Train Accuracy
0	svc	0.981	0.980	0.985
2	mnb	0.978	0.979	0.983
7	xgb	0.975	0.975	0.985
5	rfc	0.973	0.973	1.000
6	adb	0.971	0.967	0.975
4	lr	0.963	0.962	0.965
8	gbc	0.954	0.955	0.966
3	dtc	0.937	0.933	0.947
1	knc	0.903	0.908	0.927

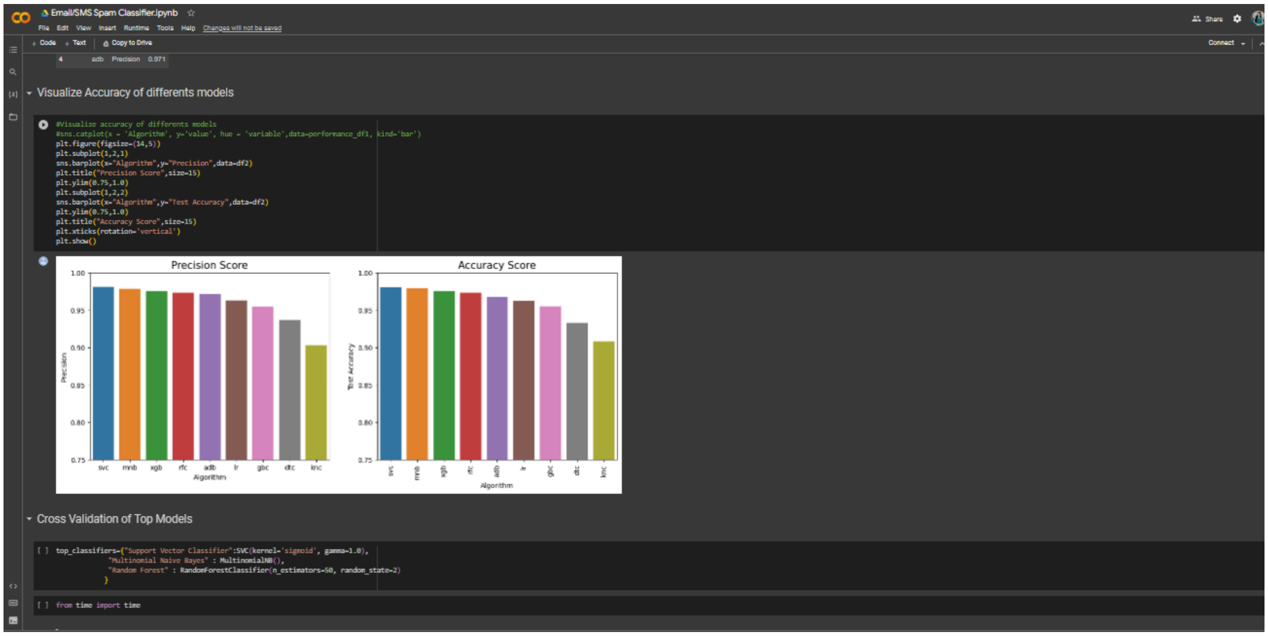
```python
df3 = pd.melt(df2, id_vars = "Algorithm")
```

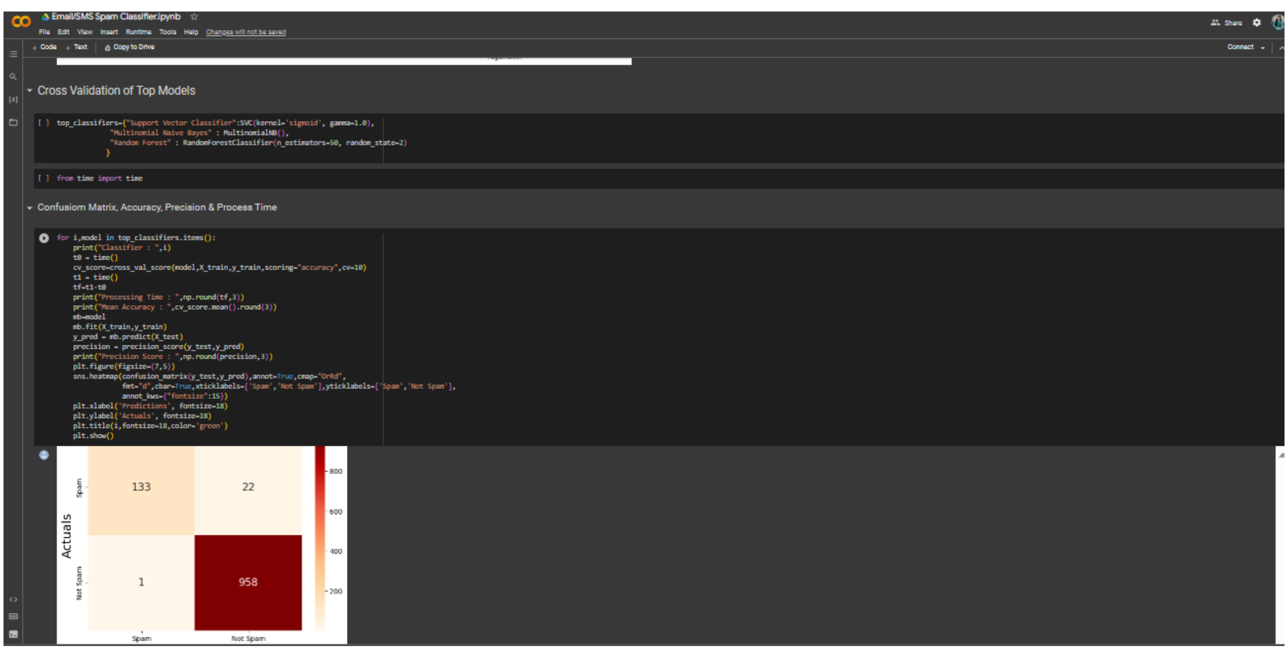
```python
df3.head()
```

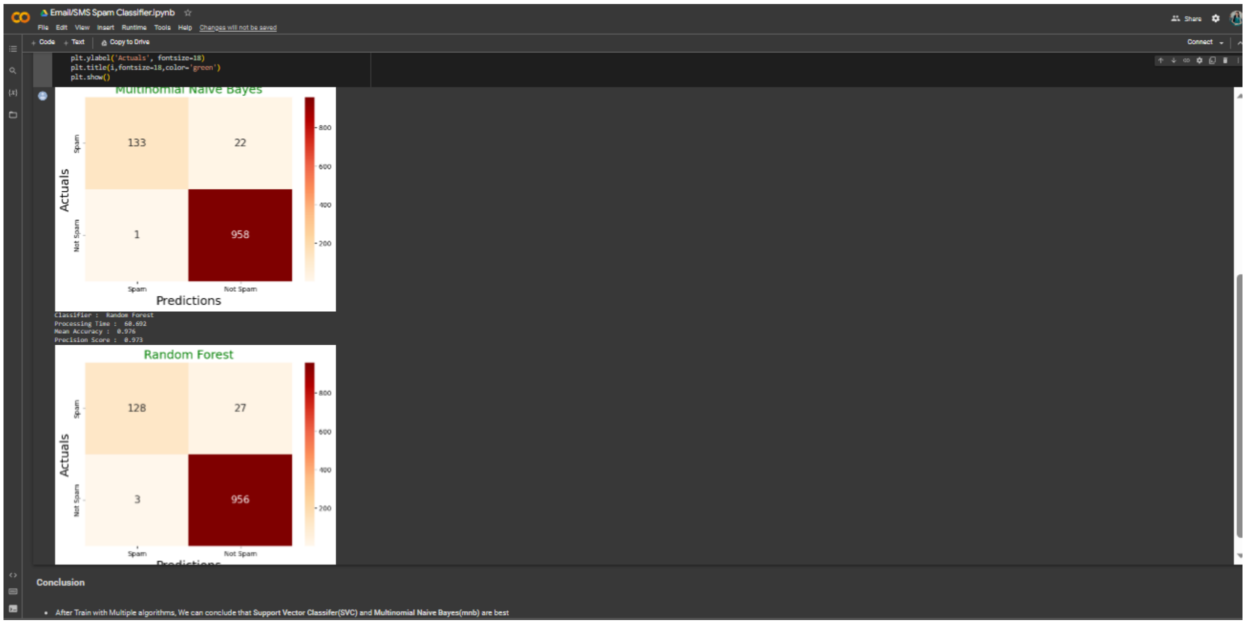
	Algorithm	variable	value
0	svc	Precision	0.981
1	mnb	Precision	0.978
2	xgb	Precision	0.975
3	rfc	Precision	0.973
4	adb	Precision	0.971

Visualize Accuracy of differents models

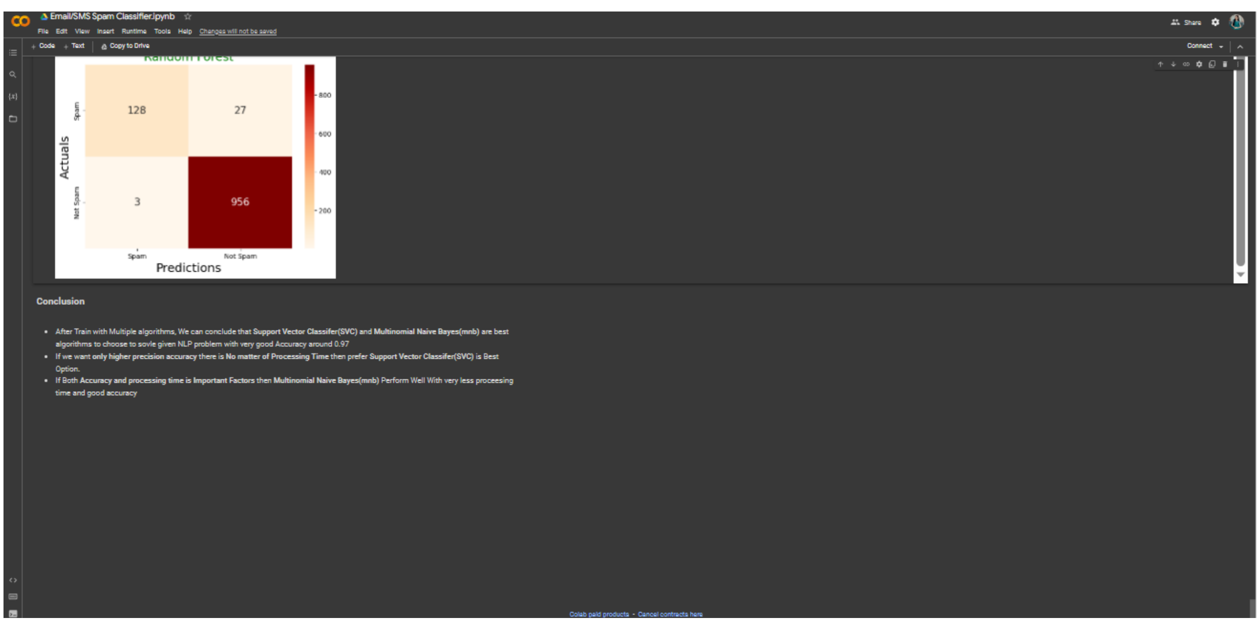
```python
#Visualize accuracy of differents models
#sns.catplot(x = 'Algorithm', y='value', hue = 'variable',data=performance_df1, kind='bar')
plt.figure(figsize=(14,5))
plt.subplot(1,2,1)
sns.barplot(x="Algorithm",y="Precision",data=df2)
plt.title("Precision Score",size=15)
plt.ylim(0.75,1.0)
plt.subplot(1,2,2)
sns.barplot(x="Algorithm",y="Test Accuracy",data=df2)
plt.ylim(0.75,1.0)
plt.title("Accuracy Score",size=15)
plt.xticks(rotation='vertical')
plt.show()
```


Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Code Text Copy to Drive
Share
Connect
4 adb Precision 0.971
Visualize Accuracy of differents models
#Visualize accuracy of differents models
#sns.catplot(x = 'Algorithm', y='value', hue = 'variable',data=performance_df1, kind='bar')
plt.figure(figsize=(14,5))
plt.subplot(1,2,1)
sns.barplot(x="Algorithm",y="Precision",data=df2)
plt.title("Precision Score",size=15)
plt.ylim(0.75,1.0)
plt.subplot(1,2,2)
sns.barplot(x="Algorithm",y="Test Accuracy",data=df2)
plt.ylim(0.75,1.0)
plt.title("Accuracy Score",size=15)
plt.xticks(rotation='vertical')
plt.show()
Precision Score
Accuracy Score
1.00
0.95
0.90
0.85
0.80
0.75
Precision
Test Accuracy
svc mnb xgb rfc adb lr gbc dtc knc
Algorithm
Cross Validation of Top Models
top_classifiers={"Support Vector Classifier":SVC(kernel='sigmoid', gamma=1.0),
"Multinomial Naive Bayes" : MultinomialNB(),
"Random Forest" : RandomForestClassifier(n_estimators=50, random_state=2)
}
from time import time


Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Code Text Copy to Drive
Share
Connect
Cross Validation of Top Models
top_classifiers={"Support Vector Classifier":SVC(kernel='sigmoid', gamma=1.0),
"Multinomial Naive Bayes" : MultinomialNB(),
"Random Forest" : RandomForestClassifier(n_estimators=50, random_state=2)
}
from time import time
Confusiom Matrix, Accuracy, Precision & Process Time
for i,model in top_classifiers.items():
print("Classifier : ",i)
t0 = time()
cv_score=cross_val_score(model,X_train,y_train,scoring="accuracy",cv=10)
t1 = time()
tf=t1-t0
print("Processing Time : ",np.round(tf,3))
print("Mean Accuracy : ",cv_score.mean().round(3))
mb=model
mb.fit(X_train,y_train)
y_pred = mb.predict(X_test)
precision = precision_score(y_test,y_pred)
print("Precision Score : ",np.round(precision,3))
plt.figure(figsize=(7,5))
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,cmap="OrRd",
fmt="d",cbar=True,xticklabels=['Spam','Not Spam'],yticklabels=['Spam','Not Spam'],
annot_kws={"fontsize":15})
plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title(i,fontsize=18,color='green')
plt.show()
Actuals
Spam
Not Spam
133
22
1
958
800
600
400
200



Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Code Text Copy to Drive
Share
Connect
plt.ylabel('Actuals', fontsize=18)
plt.title(i,fontsize=18,color='green')
plt.show()
Multinomial Naive Bayes
Spam
Not Spam
Actuals
133
22
1
958
Spam
Not Spam
Predictions
800
600
400
200
Classifier : Random Forest
Processing Time : 60.692
Mean Accuracy : 0.976
Precision Score : 0.973
Random Forest
Spam
Not Spam
Actuals
128
27
3
956
Spam
Not Spam
Predictions
800
600
400
200
Conclusion
After Train with Multiple algorithms, We can conclude that Support Vector Classifer(SVC) and Multinomial Naive Bayes(mnb) are best



Email/SMS Spam Classifier.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
Code Text Copy to Drive
Share
Connect
Random Forest
Spam
Not Spam
Actuals
128
27
3
956
Spam
Not Spam
Predictions
800
600
400
200
Conclusion
After Train with Multiple algorithms, We can conclude that Support Vector Classifer(SVC) and Multinomial Naive Bayes(mnb) are best algorithms to choose to sovle given NLP problem with very good Accuracy around 0.97
If we want only higher precision accuracy there is No matter of Processing Time then prefer Support Vector Classifer(SVC) is Best Option.
If Both Accuracy and processing time is Important Factors then Multinomial Naive Bayes(mnb) Perform Well With very less proceesing time and good accuracy
Colab paid products - Cancel contracts here

PROJECT CODE:

```
from google.colab import drive
drive.mount('/content/drive')

import nltk
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

#import DataFrame
df = pd.read_csv('/content/drive/MyDrive/Datasets/spam.csv',encoding='latin-1')
df
#Remove Unwanted Columns
df=df.iloc[:,:2]
#Rename Columns
df=df.rename(columns={"v1":"label","v2":"text"})
df
#Total No of Spam and Not Spam Category
```

```
df['label'].value_counts()

import matplotlib.pyplot as plt
plt.figure(figsize=(9,4))
plt.subplot(1,2,1)
plt.pie(df['label'].value_counts(),labels=['Negative','Positive'],autopct="%0.2f")
plt.subplot(1,2,2)
sns.barplot(x=df['label'].value_counts().index,y=df['label'].value_counts(),data=df)
plt.show()
```

#Info

```
df.info()
```

#shape of df

```
df.shape
```

#Check for Null Values

```
df.isnull().sum()

df['label'].value_counts()/df.shape[0]*100
```

#1. Total No of Char

```
df['num_char']=df['text'].apply(len)
```

#2. Total No of Words

```
df['num_words']=df['text'].apply(lambda x: len(str(x).split()))
```

3. Total No of Sentences

```
nltk.download('punkt')
df['num_sen']=df['text'].apply(lambda x: len(nltk.sent_tokenize(x)))
```

#Statistical Info about dataset

```
df.describe()
df[df['label']=='ham'].describe()
df[df['label']=='spam'].describe()
```

#Hist Plot for Spam and Not Spam

```
plt.figure(figsize=(12,6))
sns.histplot(df[df['label']=='spam']['num_words'],color='blue',bins=40)
sns.histplot(df[df['label']=='ham']['num_words'],color='red')
plt.show()

sns.pairplot(df,hue='label')
plt.show()
```

#Remove few Outliers present in dataset

```
i=df[df['num_char']>500].index
df.drop(i,axis=0,inplace=True)

df=df.reset_index()
df.drop("index",inplace=True,axis=1)

sns.pairplot(df,hue='label')
plt.show()
```

#HeatMap

```
sns.heatmap(df.corr(),annot=True)
plt.show()
```

#Import lib required for text processing

```
from nltk.corpus import stopwords
nltk.download('stopwords')
stopwords.words('english')
from nltk.stem import PorterStemmer
from wordcloud import WordCloud
import string,time
string.punctuation
```

#punctuation

```
string.punctuation
```

#Stopwards

stopwords.words('english')

#User Define Funtion for Text processing

```
def remove_website_links(text):
    no_website_links = text.replace(r"http\S+", "")
    return no_website_links


def remove_numbers(text):
    removed_numbers = text.replace(r'\d+','')
    return removed_numbers


def remove_emails(text):
    no_emails = text.replace(r"\S*@\S*\s?",'')
    return no_emails
```

#Call Function

```
df['text'] = df['text'].apply(remove_website_links)
df['text'] = df['text'].apply(remove_numbers)
df['text'] = df['text'].apply(remove_emails)
```

#Create Common Function

```
def transform_text(text):
    #1.lower casing
    text=text.lower()

    #2.tokenization
    lst=nltk.word_tokenize(text)

    #3.remove spcl characters stopwords and punctuation
    l1=[]
    useless_words=stopwords.words('english')+list(string.punctuation)
    for word in lst:
        if word.isalnum()==True and word not in useless_words:
            l1.append(word)

    #4.stemming
    l2=[]
    for word in l1:
        ps=PorterStemmer()
        l2.append(ps.stem(word))

    return " ".join(l2).strip()
    l1.clear()
    l2.clear()
```

#call function

nltk.download('punkt')

df['text'] = df['text'].apply(transform_text)

df['num_words_transform']=df['text'].apply(lambda x: len(str(x).split()))

#conda install -c conda-forge wordcloud

from wordcloud import WordCloud

plt.figure(figsize = (20,20)) # Text that is not fraudulent(0)

wc = WordCloud(width = 1600 , height = 800 , max_words = 500,background_color='black').generate(" ".join(df[df.label == 'spam'].text))

plt.imshow(wc , interpolation = 'bilinear')

plt.figure(figsize = (20,20)) # Text that is not fraudulent(0)

wc = WordCloud(width = 1600 , height = 800 , max_words = 500).generate(" ".join(df[df.label == 'ham'].text))

plt.imshow(wc , interpolation = 'bilinear')

df.head()

df['label']=df['label'].replace({'spam':0,'ham':1})

#create spam corpus which will holds all Spam words

```
spam_corpus = []

for msg in df[df['label'] == 1]['text'].tolist():
    for word in msg.split():
        spam_corpus.append(word)

from collections import Counter
```

```
plt.figure(figsize=(10,7))

sns.barplot(y=pd.DataFrame(Counter(spam_corpus).most_common(25))[0],x=pd.DataFrame(Counter(spam_corpus).most_common(30))[1])

plt.xticks()

plt.title("Most Commonly Used Non Spam Words")

plt.xlabel("Frequnecy")

plt.ylabel("Ham Words")

plt.show()

#create spam corpus which will holds all Ham or Non Spam words

ham_corpus = []

for msg in df[df['label'] == 0]['text'].tolist():

    for word in msg.split():

        ham_corpus.append(word)

from collections import Counter

plt.figure(figsize=(10,7))

sns.barplot(y=pd.DataFrame(Counter(ham_corpus).most_common(25))[0],x=pd.DataFrame(Counter(ham_corpus).most_common(30))[1])

plt.xticks()

plt.title("Most Commonly Used Spam Words")

plt.xlabel("Frequnecy")

plt.ylabel("Spam Words")

plt.show()
```

#Characters Visualize

```
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,6))
text_len=df[df['label']==1]['text'].str.len()
ax1.hist(text_len,color='green')
ax1.set_title('Original text')
text_len=df[df['label']==0]['text'].str.len()
ax2.hist(text_len,color='red')
ax2.set_title('Fake text')
fig.suptitle('Characters in texts')
plt.show()
#Words Visualize
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,6))
text_len=df[df['label']==1]['num_words_transform']
ax1.hist(text_len,color='red')
ax1.set_title('Original text')
text_len=df[df['label']==0]['num_words_transform']
ax2.hist(text_len,color='green')
ax2.set_title('Fake text')
fig.suptitle('Words in texts')
plt.show()
fig,(ax1,ax2)=plt.subplots(1,2,figsize=(15,6))
word=df[df['label']==1]['num_words_transform']
sns.distplot(word.map(lambda x: np.mean(x)),ax=ax1,color='red')
ax1.set_title('Original text')
```

```
word=df[df['label']==0]['num_words_transform']
sns.distplot(word.map(lambda x: np.mean(x)),ax=ax2,color='green')
ax2.set_title('Fake text')
fig.suptitle('Average word length in each text')
#Text Vecorization
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
#Input and Output Features
X = tfidf.fit_transform(df['text']).toarray()
y = df['label'].values
X.shape
y.shape
from sklearn.model_selection import train_test_split
#Model Training
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score
!pip install --upgrade scikit-learn
#Imports Lib
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import AdaBoostClassifier

from sklearn.ensemble import BaggingClassifier

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.ensemble import GradientBoostingClassifier

from xgboost import XGBClassifier

from sklearn.metrics import precision_score, recall_score, classification_report, accuracy_score, f1_score

#from sklearn.metrics import plot_confusion_matrix

from sklearn.model_selection import cross_val_score

#from sklearn.metrics import precision_score, recall_score, plot_confusion_matrix, classification_report, accuracy_score, f1_score

from sklearn.model_selection import cross_val_score

#GaussianNB

gnb = GaussianNB()

gnb.fit(X_train,y_train)

y_pred1 = gnb.predict(X_test)

print("Accuracy Score -",accuracy_score(y_test,y_pred1))

print("Precision Score -",precision_score(y_test,y_pred1))

print(confusion_matrix(y_test,y_pred1))

#MultinomialNB

mnb = MultinomialNB()
```

```
mnb.fit(X_train,y_train)
y_pred2 = mnb.predict(X_test)
print("Accuracy Score -",accuracy_score(y_test,y_pred2))
print("Precision Score -",precision_score(y_test,y_pred2))
print(confusion_matrix(y_test,y_pred2))


classifiers={"svc":SVC(kernel='sigmoid', gamma=1.0),
            "knc": KNeighborsClassifier(),
             "mnb" : MultinomialNB(),
             "dtc" : DecisionTreeClassifier(max_depth=5),
             "lr" : LogisticRegression(solver='liblinear', penalty='l1'),
             "rfc" : RandomForestClassifier(n_estimators=50, random_state=2),
             "adb" : AdaBoostClassifier(n_estimators=50, random_state=2),
             "xgb" : XGBClassifier(n_estimators=50,random_state=2),
             "gbc" : GradientBoostingClassifier(n_estimators=50,random_state=2)
            }
```

#Common Function for model train

```
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)
```

```
    train_accuracy = clf.score(X_train,y_train)

    return accuracy,precision,train_accuracy
svc=SVC(kernel='sigmoid', gamma=1.0)
train_classifier(svc,X_train,y_train,X_test,y_test)
accuracy_scores = []
precision_scores = []
train_accuracy_score=[]

for name,clf in classifiers.items():

    current_accuracy,current_precision,current_train_score = 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)
    train_accuracy_score.append(current_train_score)
    print()
df1=pd.DataFrame({'Algorithm':classifiers.keys(),'Precision':precision_scores,
```

```
'Test Accuracy':accuracy_scores,'Train Accuracy':train_accuracy_score}).round(3)

df2=df1.sort_values(['Precision','Test Accuracy'],ascending=False)

df2

df3 = pd.melt(df2, id_vars = "Algorithm")

df3.head()
```

#Visualize accuracy of differents models

```
#sns.catplot(x = 'Algorithm', y='value', hue = 'variable',data=performance_df1, kind='bar')

plt.figure(figsize=(14,5))

plt.subplot(1,2,1)

sns.barplot(x="Algorithm",y="Precision",data=df2)

plt.title("Precision Score",size=15)

plt.ylim(0.75,1.0)

plt.subplot(1,2,2)

sns.barplot(x="Algorithm",y="Test Accuracy",data=df2)

plt.ylim(0.75,1.0)

plt.title("Accuracy Score",size=15)

plt.xticks(rotation='vertical')

plt.show()

top_classifiers={"Support Vector Classifier":SVC(kernel='sigmoid', gamma=1.0),
```

```
        "Multinomial Naive Bayes" : MultinomialNB(),
        "Random Forest" : RandomForestClassifier(n_estimators=50, random_state=2)
        }
from time import time

for i,model in top_classifiers.items():
    print("Classifier : ",i)
    t0 = time()
    cv_score=cross_val_score(model,X_train,y_train,scoring="accuracy",cv=10)
    t1 = time()
    tf=t1-t0
    print("Processing Time : ",np.round(tf,3))
    print("Mean Accuracy : ",cv_score.mean().round(3))
    mb=model
    mb.fit(X_train,y_train)
    y_pred = mb.predict(X_test)
    precision = precision_score(y_test,y_pred)
    print("Precision Score : ",np.round(precision,3))
    plt.figure(figsize=(7,5))
    sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,cmap="OrRd",
            fmt="d",cbar=True,xticklabels=['Spam','Not Spam'],yticklabels=['Spam','Not Spam'],
```

```
            annot_kws={"fontsize":15})
plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title(i,fontsize=18,color='green')
plt.show()
```

TEAM MEMBERS DETAILS:

TEAM NAME: Shadow knights

MEMBER 1:

NAME : HEMA HARIHARAN S

DEPT : CSE

Email : hemahariharansamson@gmail.com

Mobile Number : 9080602796

MEMBER 2:

NAME : DHATCHAYANI U

DEPT : CSE

Email : dhatchayani8102@gmail.com

Mobile Number : 9385448919