Text preparation, classification, and model assessment

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#import text data from csv file
df=pd.read_csv('/content/Text_data.csv')
#df.head()
df.tail()
#pd.set_option('max_colwidth', 100)
```

```
sns.countplot(df.ADR_label) #Imbalanced Target
```

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

```
import nltk
import string
import re
from nltk.stem.snowball import SnowballStemmer
stopwords=nltk.corpus.stopwords.words('english')
snowball_stemmer=SnowballStemmer(language='english')
def treat_text(text):
    edited_text=re.sub('\W'," ",text) #replace any sumbol with whitespace
    edited_text=re.sub("  "," ",edited_text) #replace double whitespace with single whitespace
    edited_text=edited_text.split(" ") #split the sentence into array of strings
    edited_text=" ".join([char for char in edited_text if char!= ""]) #remove any empty string from text
    edited_text=edited_text.lower() #lowercase
    edited_text=re.sub('\d+',"",edited_text) #Removing numerics
    edited_text=re.split('\W+',edited_text) #spliting based on whitespace or whitespaces
    edited_text=" ".join([snowball_stemmer.stem(word) for word in edited_text if word not in stopwords]) #Snowball Stemmer
    return edited_text
df['Treated_Tweet']=df.Tweet.apply(lambda x: treat_text(x))
df.head()
```

```
# Most Frequent words in the dataset
df.Treated_Tweet.str.split(expand=True).stack().value_counts()[:10]
```

```
freq_words=df.Treated_Tweet.str.split(expand=True).stack().value_counts()[:10]
freq_words=list(freq_words.index)
```

```
# Rare words in the dataset appearing only once in the whole dataset
df.Treated_Tweet.str.split(expand=True).stack().value_counts()[-20:]
```

```
rare_words=df.Treated_Tweet.str.split(expand=True).stack().value_counts()
rare_words=list(rare_words.loc[lambda x: x==1].index)
```

```
#Remove Frequent and Rare words
def remove_noise_words(text):
    edited_text=text.split()
    edited_text=[word for word in edited_text if word not in freq_words]
    edited_text=[word for word in edited_text if word not in rare_words]
    edited_text=" ".join(edited_text)
    return edited_text
```

```
df['Final_Treated_words']=df.Treated_Tweet.apply(lambda x: remove_noise_words(x))
```

```
df.head()
```

```
X=df.Final_Treated_words
y=df.ADR_label
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20, random_state=10)
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
#Creating a list of Pipeline with well-known ML models
from sklearn.pipeline import make_pipeline
from sklearn.naive_bayes import MultinomialNB,ComplementNB
from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.tree import DecisionTreeClassifier

pipelines=[]
for model in [DecisionTreeClassifier(), MultinomialNB(), ComplementNB(),
              LogisticRegression(solver='saga'), RidgeClassifier(solver='auto'), SVC(),RandomForestClassifier()]:
    pipeline=make_pipeline(TfidfVectorizer(), model)
    pipelines.append(pipeline)
```

```
#Training the model
```

```
import time
training_time=[]
for pipeline in pipelines:
    start=time.time()
    pipeline.fit(X_train, y_train)
    stop=time.time()
    training_time.append(stop-start)
```

```
#Prediction from test dataset
from sklearn.metrics import classification_report, confusion_matrix, f1_score, precision_score, recall_score
model_name=[]
precision_array=[]
recall_array=[]
f1_array=[]
test_time=[]
print("Classifiation Report\n")
print("**************************************************")
for i, pipeline in enumerate(pipelines):
    start=time.time()
    y_pred=pipeline.predict(X_test)
    stop=time.time()
    test_time.append(stop-start)
    print(pipelines[i].steps[1][0].upper())
    model_name.append(pipelines[i].steps[1][0].upper())
    f1_array.append(round(f1_score(y_test, y_pred, average='weighted'),2))
    precision_array.append(round(precision_score(y_test, y_pred, average='binary'),2))
    recall_array.append(round(recall_score(y_test, y_pred, average='binary'),2))
    print("\n",classification_report(y_test, y_pred))
    print("**************************************************")
```

```
#Plotting the various performance metrix of all models
training_time=np.array(training_time)/np.max(training_time)
test_time=np.array(test_time)/np.max(test_time)
score_df=pd.DataFrame({'F1 Score(Weighted)':f1_array,
                       'Precision Score of Class 1':precision_array,
                       'Recall of Class 1':recall_array,
                        'Training Time': training_time,
                        'Test Time':test_time}, index=model_name)

f=plt.figure(figsize=(15,10))
plt.title('Comparing Performance of various ML Models on Text Classification Dataset', color='black',
          fontdict={'fontsize':23})
score_df.plot(kind='barh', ax=f.gca(), cmap='gnuplot')
plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
plt.show()
```

Text summary

```
import logging
logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=logging.INFO)
from gensim.summarization import summarize
# load data
filename = '/content/Europe_Covid.txt'
file = open(filename, 'rt')
text = file.read()
file.close()
print ('Summary:')
print (summarize(text))
```

```
2020-10-25 15:33:47,048 : INFO : 'pattern' package not found; tag filters are not available for English
2020-10-25 15:33:47,065 : INFO : adding document #0 to Dictionary(0 unique tokens: [])
2020-10-25 15:33:47,068 : INFO : built Dictionary(171 unique tokens: ['affect', 'case', 'class', 'contin', 'countri']...) from 23 documents (total 257 co
Summary:
European countries are reporting record numbers of Covid-19 cases as the continent prepares for the pandemic to intensify through winter.Those affected i
Polish President Andrzej Duda tested positive for Covid-19 on Friday, according to a tweet from Presidential Minister Blazej Spychalski Saturday.
Duda's diagnosis comes as the country reported 13,632 new cases Friday, the highest daily tally since the pandemic began.
On the same day France reported 42,032 new cases in 24 hours, a new record, according to the French Health Agency.
```