Explaining machine learning models in Phrase Detection

1.1. REQUIRED MODULES

- Numpy
- Sklearn
- Pandas
- re
- Nltk

● SOFTWARE REQUIREMENT

	Operating system	:	Windows 10
	Coding Language	:	PYTHON
	Tool	:	JUPYTER NOTEBOOK

1. BASIC METHODOLOGY

The given training_data.tsv is read by help of pands

```
# Reading data
train_data = pd.read_csv('../data/training_data.tsv',header=0,delimiter="\t" ,quoting=3)
```

Then we create a function that can detect lebel found or not found

```
# Function to define if there is reminder
def found(x):
    if (x[0] == "Not Found"):
        return "Not Found"
    else:
        return "Found"
```

In the next line I applied the found function on training_data.tsv

```
#Applying the found function
```

```
train_data['label_found'] = train_data[['label']].apply( found , axis = 1)
```

The clean_data function is created for clear emojis and Digits

```
#This Function help us to remove the Digits and Emojis from the data
def clean_txt(texts):
    letters = re.sub("[^a-zA-Z]",' ', str(texts))
    lower_case = letters.lower()
    words = lower_case.split()
    stopword = stopwords.words('english')
    meaning_words = [w for w in words if not w in stopword]
    return (" ".join(meaning_words))
```

Then I applied the Clean_txt function on training_Data.tsv

```
#remove the Digits and Emojis from the data
train_data['sent_clean'] = [clean_txt(review) for review in train_data["sent"].values]
```

Then I load the eval_data.txt file by panda

```
#Loading test data as test_data
test_data = pd.read_csv('../data/eval_data.txt',header=0,delimiter="\t" ,quoting=3)
```

Then I applied the Clean_txt file on it

```
#remove the Digits and Emojis from the data
test_data['sent_clean'] = [clean_txt(review) for review in test_data["sent"].values]
```

Then I used CountVectorizer function and bow_train, bow_test

```
#Convert a collection of text documents to a matrix of token counts
#Bow train
vectorizer = CountVectorizer(analyzer = "word", tokenizer = None,
max_features = 12000)
bow_train = (vectorizer.fit_transform(train_data['sent_clean'])).toarray()
bow_test = (vectorizer.transform(test_data['sent_clean'])).toarray()
```

Spliting The data in test and Train with ratio of 10% AND 90%

```
#Splitting the data in test and train with ratio of 10% and 90%
t_train , t_test , s_train , s_test = train_test_split(bow_train
,train_data['label_found'] , test_size = 0.10 , random_state=101)
```

Then I applied Logistic Regression and Two Classifier one is Random Forest Classifier and Decision Tree Classifier

```
#Applying the Logistic regrssion
logreg = LogisticRegression()
logreg = logreg.fit(t_train, s_train)
#Applying Random Forest Classifier
rfc = RandomForestClassifier()
rfc.fit(t_train , s_train)
print(accuracy_score(rfc.predict(t_test),s_test))
#Applying Decision Tree Classifier
dtc = DecisionTreeClassifier()
dtc.fit(t_train , s_train)
print(accuracy_score(s_test, dtc.predict(t_test)))
```

The output

```
#Output Data in TSV format
output = pd.DataFrame( data={ "label_found":pred} )
output.to_csv( "result.tsv", index=False, quoting=3 )
output.sample(10)
```