Fake News

Detection

NLP Project

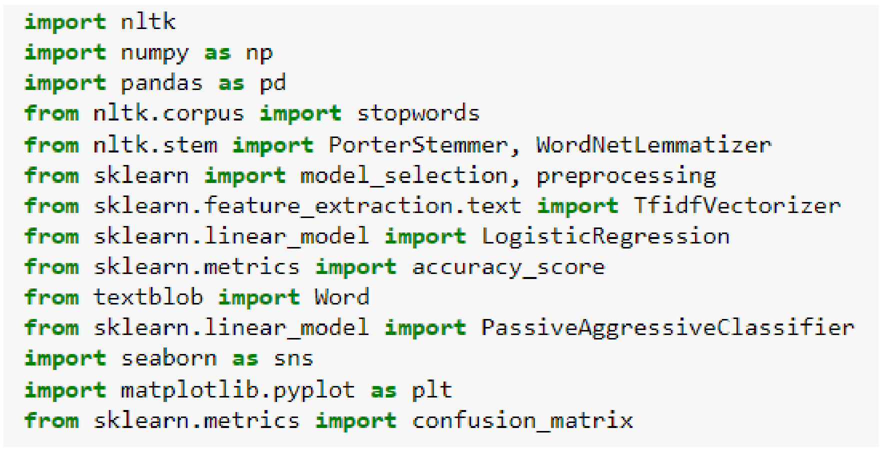
Team Information

	Name	Section	Department	ID
1	يوسف أحمد فؤاد عبدالعزيز	10	CS	20201700997
2	محمد خالد سيد معوض	7	CS	20201700688
3	عبدالله مصطفى عبدالسلام أحمد	5	CS	20201700493
4	يوسف عماد الدين مسعد علي	10	CS	20201701023
5	كريم السيد عبدالقوي	6	CS	20201700600

Introduction

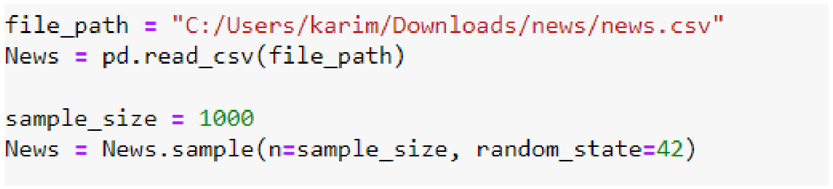
- **Fake news spreads like a wildfire and this is a big issue in this era You can learn how to distinguish fake news from a real one. fake news has become a significant problem in today's digital age. Machine learning, particularly supervised learning, can be helpful in identifying and distinguishing fake news from real news**

Used libraries



```python
import nltk
import numpy as np
import pandas as pd
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from sklearn import model_selection, preprocessing
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from textblob import Word
from sklearn.linear_model import PassiveAggressiveClassifier
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
```

Import The Data

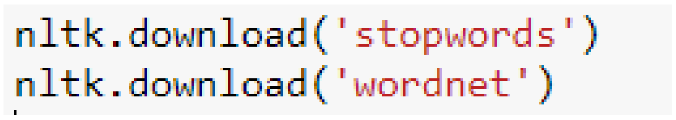


```python
file_path = "C:/Users/karim/Downloads/news/news.csv"
News = pd.read_csv(file_path)

sample_size = 1000
News = News.sample(n=sample_size, random_state=42)
```

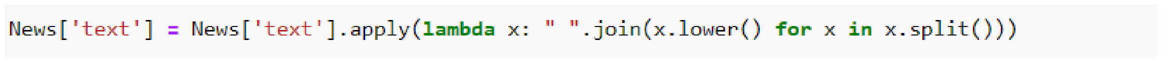
Preprocessing

Downloads stop words & WordNet lexical database:



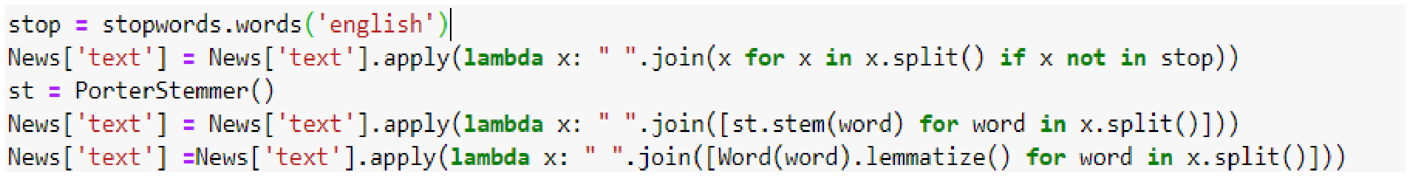
```python
nltk.download('stopwords')
nltk.download('wordnet')
```

Converts all text column to lowercase:



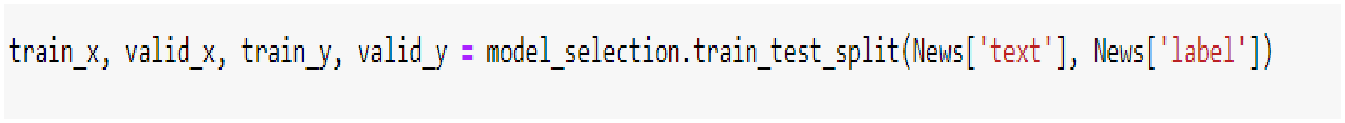
```python
News['text'] = News['text'].apply(lambda x: " ".join(x.lower() for x in x.split()))
```

Removes all stop words & applies lemmatization and stemming to all words:



```
stop = stopwords.words('english')
News['text'] = News['text'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
st = PorterStemmer()
News['text'] = News['text'].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))
News['text'] =News['text'].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))
```

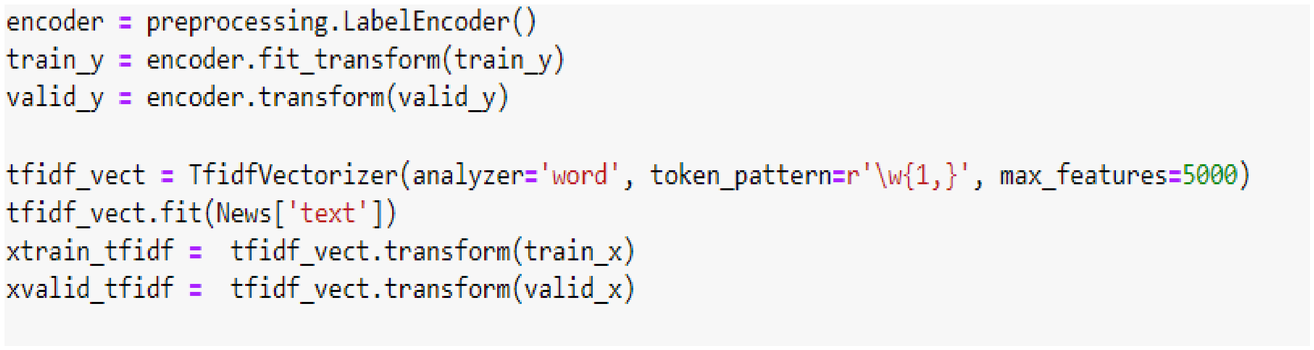
Splits the preprocessed News dataset into training and validation sets:



```
train_x, valid_x, train_y, valid_y = model_selection.train_test_split(News['text'], News['label'])
```

Feature Extraction

Label encoding and TF-IDF vectorization :

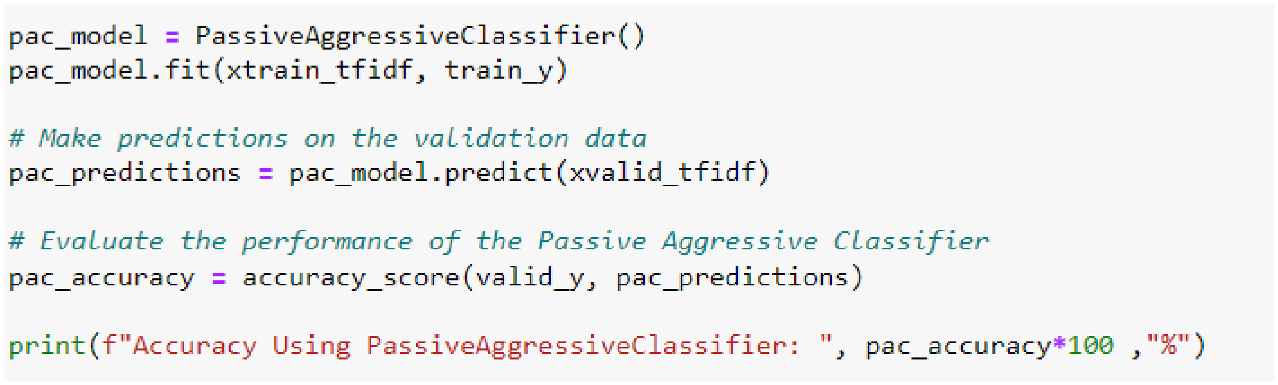


```
encoder = preprocessing.LabelEncoder()
train_y = encoder.fit_transform(train_y)
valid_y = encoder.transform(valid_y)

tfidf_vect = TfidfVectorizer(analyzer='word', token_pattern=r'\w{1,}', max_features=5000)
tfidf_vect.fit(News['text'])
xtrain_tfidf =  tfidf_vect.transform(train_x)
xvalid_tfidf =  tfidf_vect.transform(valid_x)
```

Classification models

Training a Passive Aggressive Classifier (PAC) on the preprocessed text data and evaluating its performance on a validation set:



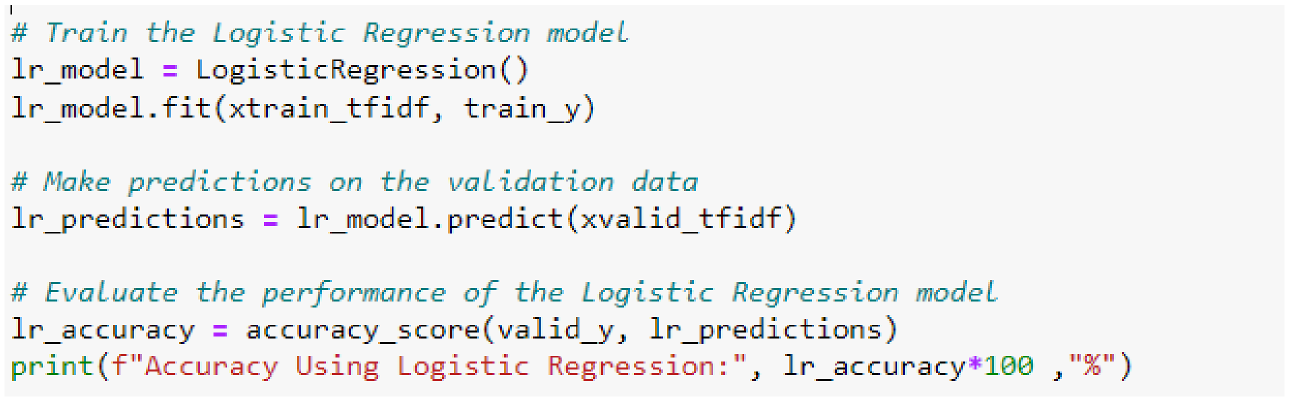
```python
pac_model = PassiveAggressiveClassifier()
pac_model.fit(xtrain_tfidf, train_y)

# Make predictions on the validation data
pac_predictions = pac_model.predict(xvalid_tfidf)

# Evaluate the performance of the Passive Aggressive Classifier
pac_accuracy = accuracy_score(valid_y, pac_predictions)

print(f"Accuracy Using PassiveAggressiveClassifier: ", pac_accuracy*100 ,"%")
```

Training a Logistic Regression Classifier (PAC) on the preprocessed text data and evaluating its performance on a validation set:



```python
# Train the Logistic Regression model
lr_model = LogisticRegression()
lr_model.fit(xtrain_tfidf, train_y)

# Make predictions on the validation data
lr_predictions = lr_model.predict(xvalid_tfidf)

# Evaluate the performance of the Logistic Regression model
lr_accuracy = accuracy_score(valid_y, lr_predictions)
print(f"Accuracy Using Logistic Regression:", lr_accuracy*100 ,"%")
```

Results Visualization

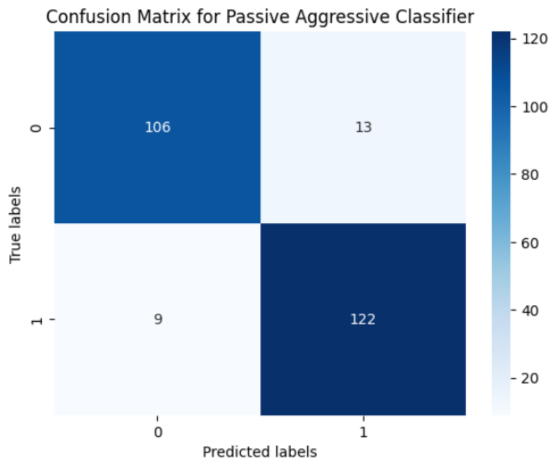
Visualizing the confusion matrices for the PAC and Logistic Regression models on the validation set:

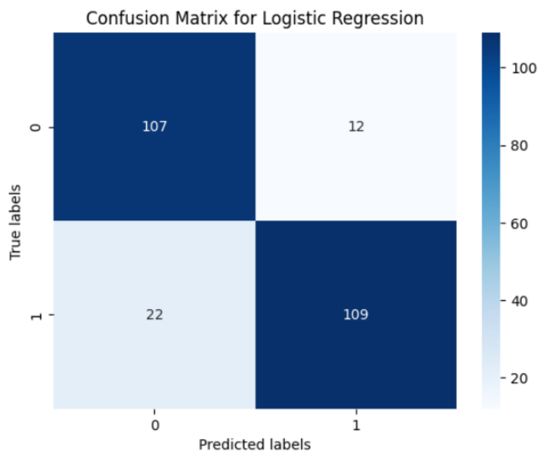
```python
# Print the confusion matrix for the Passive Aggressive Classifier
pac_cm = confusion_matrix(valid_y, pac_predictions)
print("Confusion Matrix for Passive Aggressive Classifier:")
print(pac_cm)

# Create a heatmap of the confusion matrix for the Passive Aggressive Classifier
sns.heatmap(pac_cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix for Passive Aggressive Classifier')
plt.show()

# Print the confusion matrix for the Logistic Regression model
lr_cm = confusion_matrix(valid_y, lr_predictions)
print("Confusion Matrix for Logistic Regression:")
print(lr_cm)

# Create a heatmap of the confusion matrix for the Logistic Regression model
sns.heatmap(lr_cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix for Logistic Regression')
plt.show()
```


Confusion Matrix for Passive Aggressive Classifier
True labels
Predicted labels


Confusion Matrix for Logistic Regression
True labels
Predicted labels