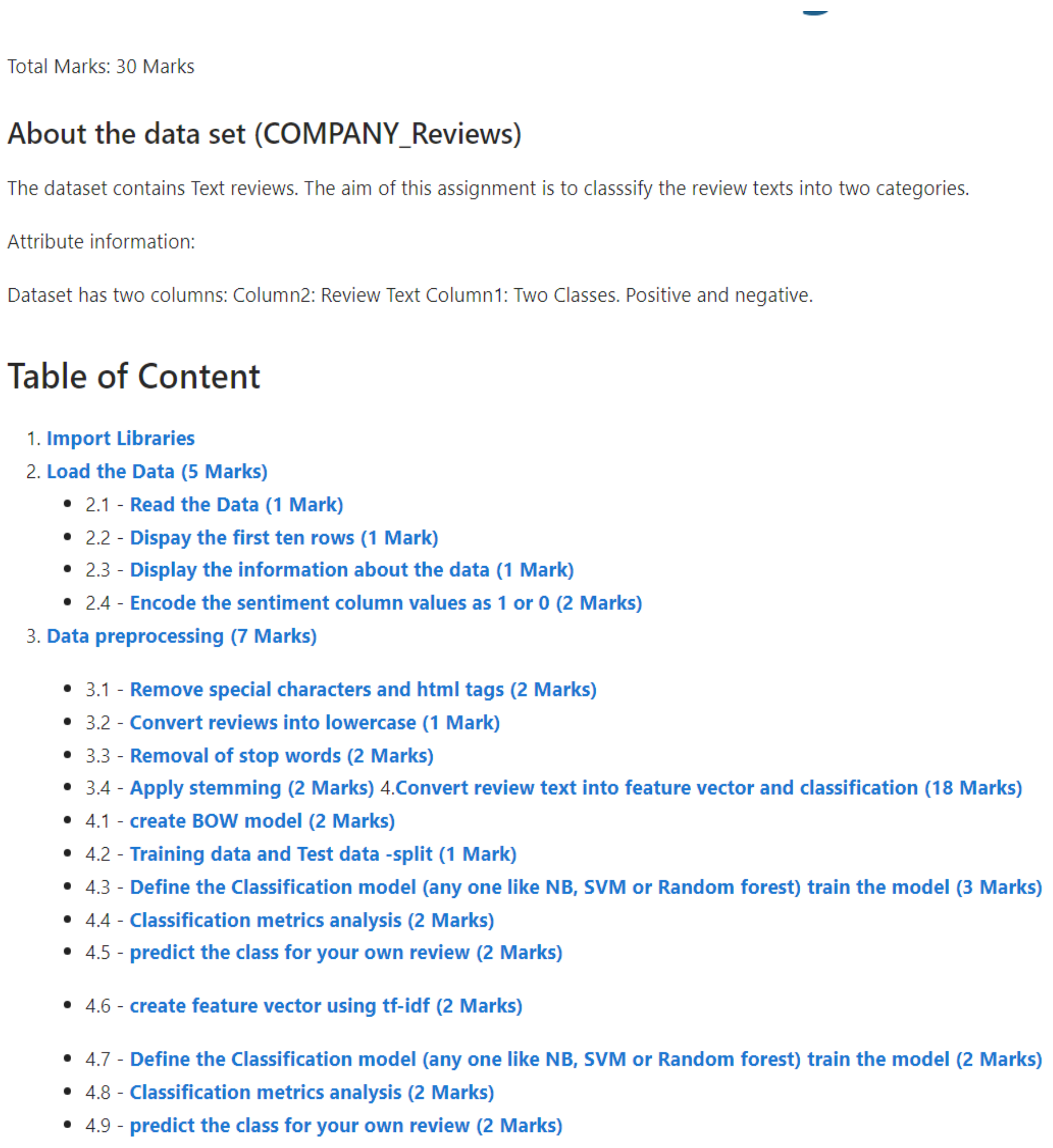
NLP
Natural language processing

Shafeena Farheen
Project By



Total Marks: 30 Marks

About the data set (COMPANY_Reviews)

The dataset contains Text reviews. The aim of this assignment is to classsify the review texts into two categories.

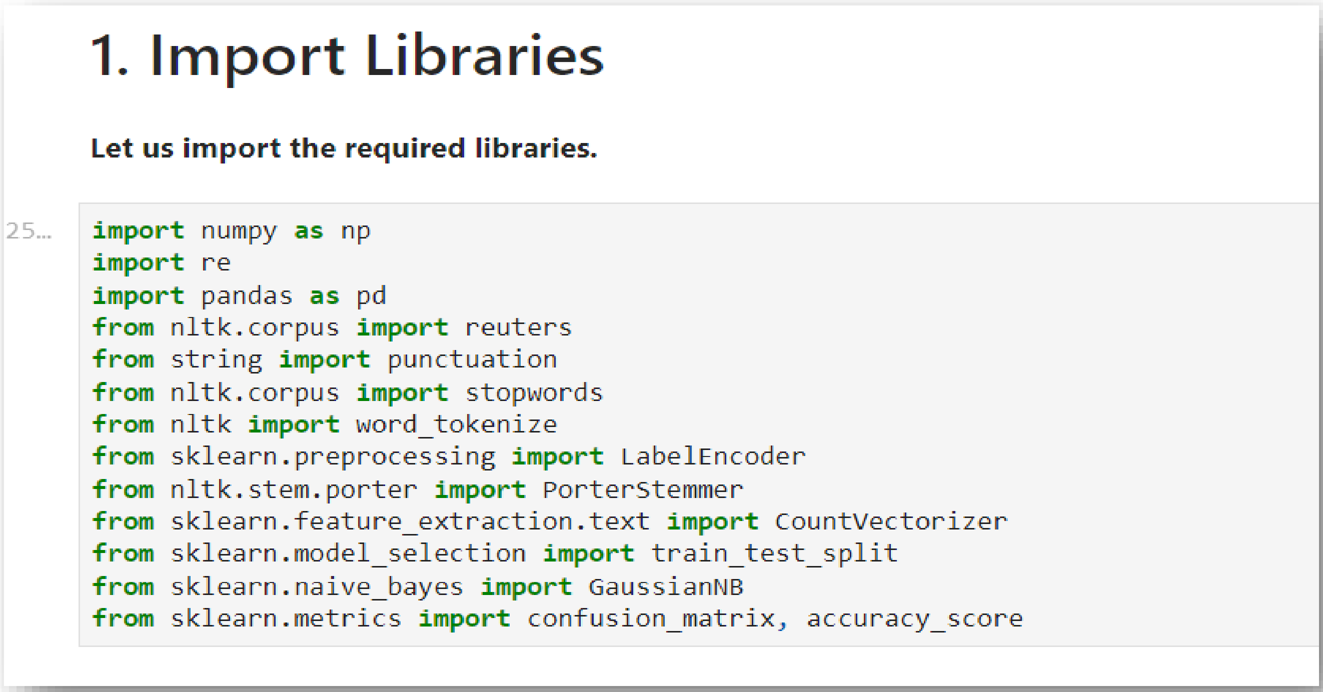
Attribute information:

Dataset has two columns: Column2: Review Text Column1: Two Classes. Positive and negative.

Table of Content

Introduction:

- Objective: To classify text reviews into two distinct categories: positive and negative sentiments.
- Dataset: Textual review data with two attributes: review text and sentiment label (Positive or Negative).
- Goal: To gain valuable insights into the sentiment conveyed by these reviews, which can be instrumental in various applications, such as customer feedback analysis and product sentiment tracking.



1. Import Libraries

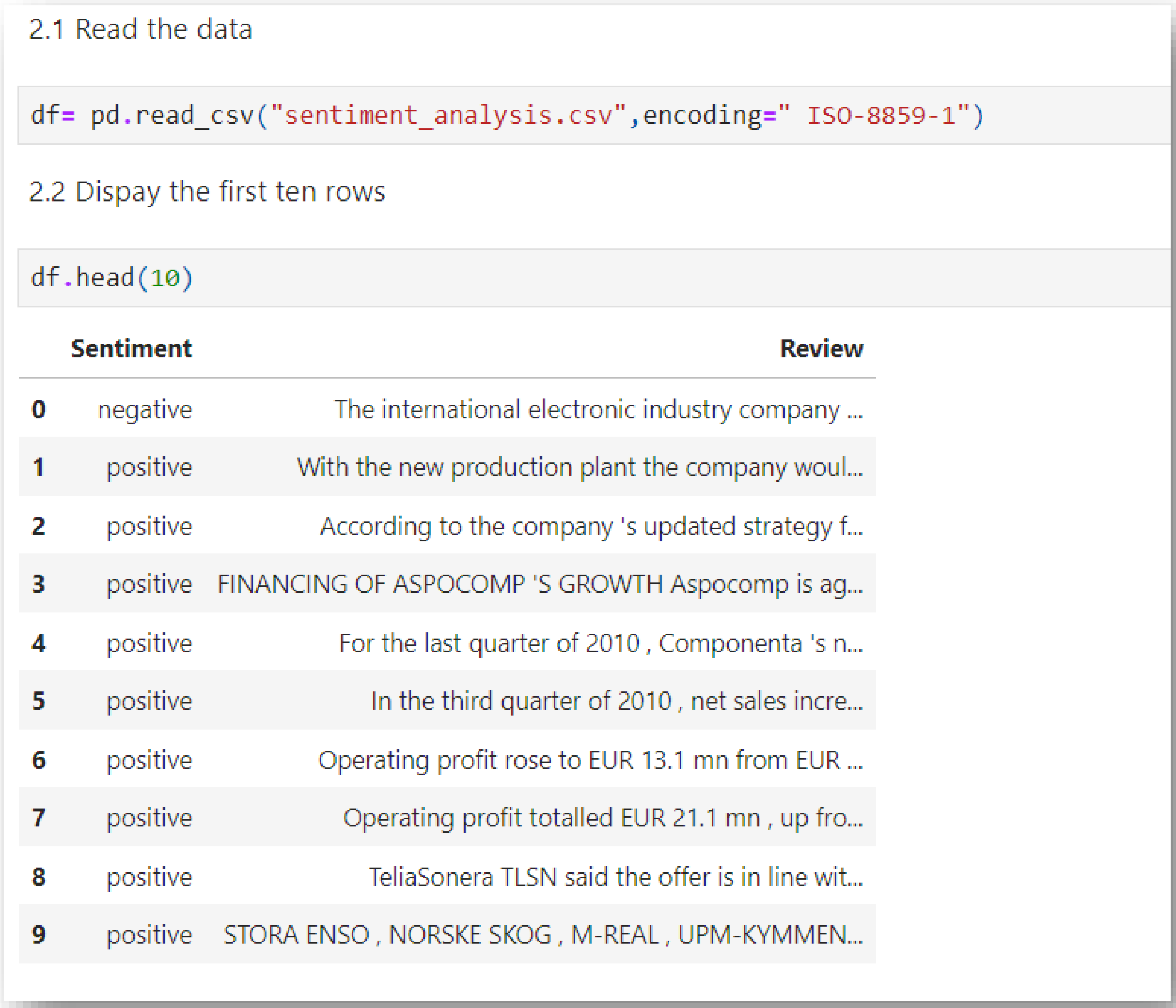
Let us import the required libraries.

```
import numpy as np
import re
import pandas as pd
from nltk.corpus import reuters
from string import punctuation
from nltk.corpus import stopwords
from nltk import word_tokenize
from sklearn.preprocessing import LabelEncoder
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score
```

Inference:

Importing libraries for data analysis and machine learning:

- numpy (np): numerical operations and array handling
- re: regular expressions for text preprocessing
- pandas (pd): data manipulation, loading, and exploration
- nltk: natural language processing tools, including stopwords and a Porter Stemmer
- string.punctuation: punctuation removal from text
- sklearn: machine learning tasks, including data preprocessing, feature extraction, model building, and evaluation
- sklearn.preprocessing.LabelEncoder: encoding the 'Sentiment' column into numerical values
- nltk.stem.PorterStemmer: word stemming
- sklearn.feature_extraction.text.CountVectorizer and sklearn.feature_extraction.text.TfidfVectorizer: converting text data into numerical feature vectors using Bag of Words (BoW) and TF-IDF (Term Frequency-Inverse Document Frequency) approaches
- sklearn.model_selection.train_test_split: splitting the dataset into training and testing sets
- sklearn.naive_bayes.GaussianNB: classification model
- sklearn.metrics.confusion_matrix and sklearn.metrics.accuracy_score: evaluating the classification model's performance



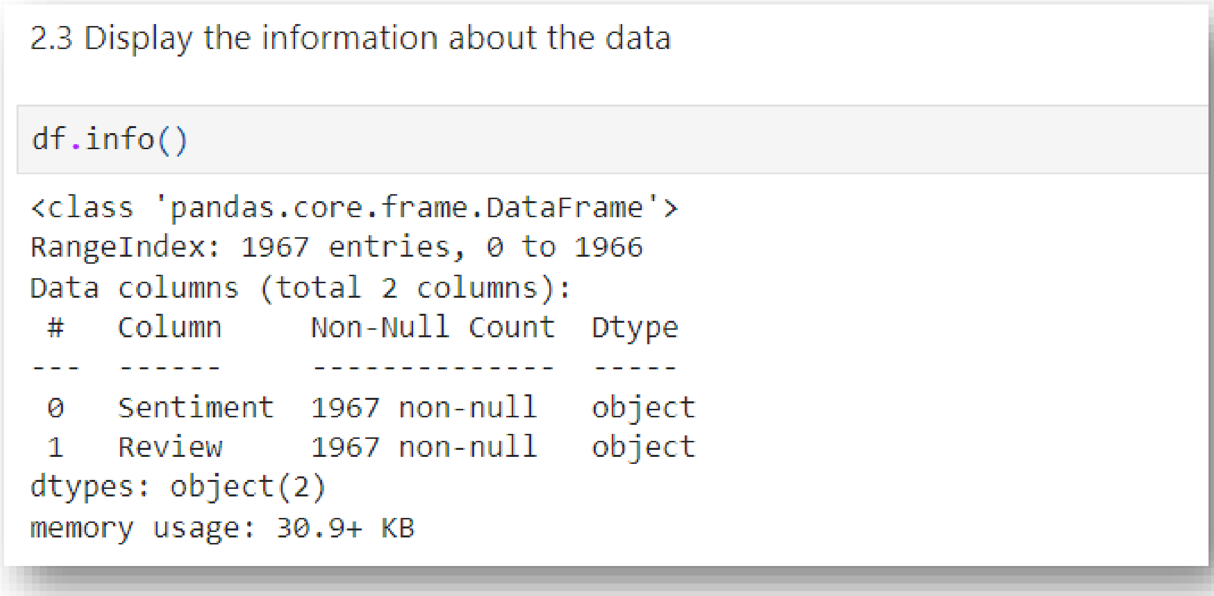
2.1 Read the data

```
df= pd.read_csv("sentiment_analysis.csv",encoding=" ISO-8859-1")
```

2.2 Dispay the first ten rows

```
df.head(10)
```

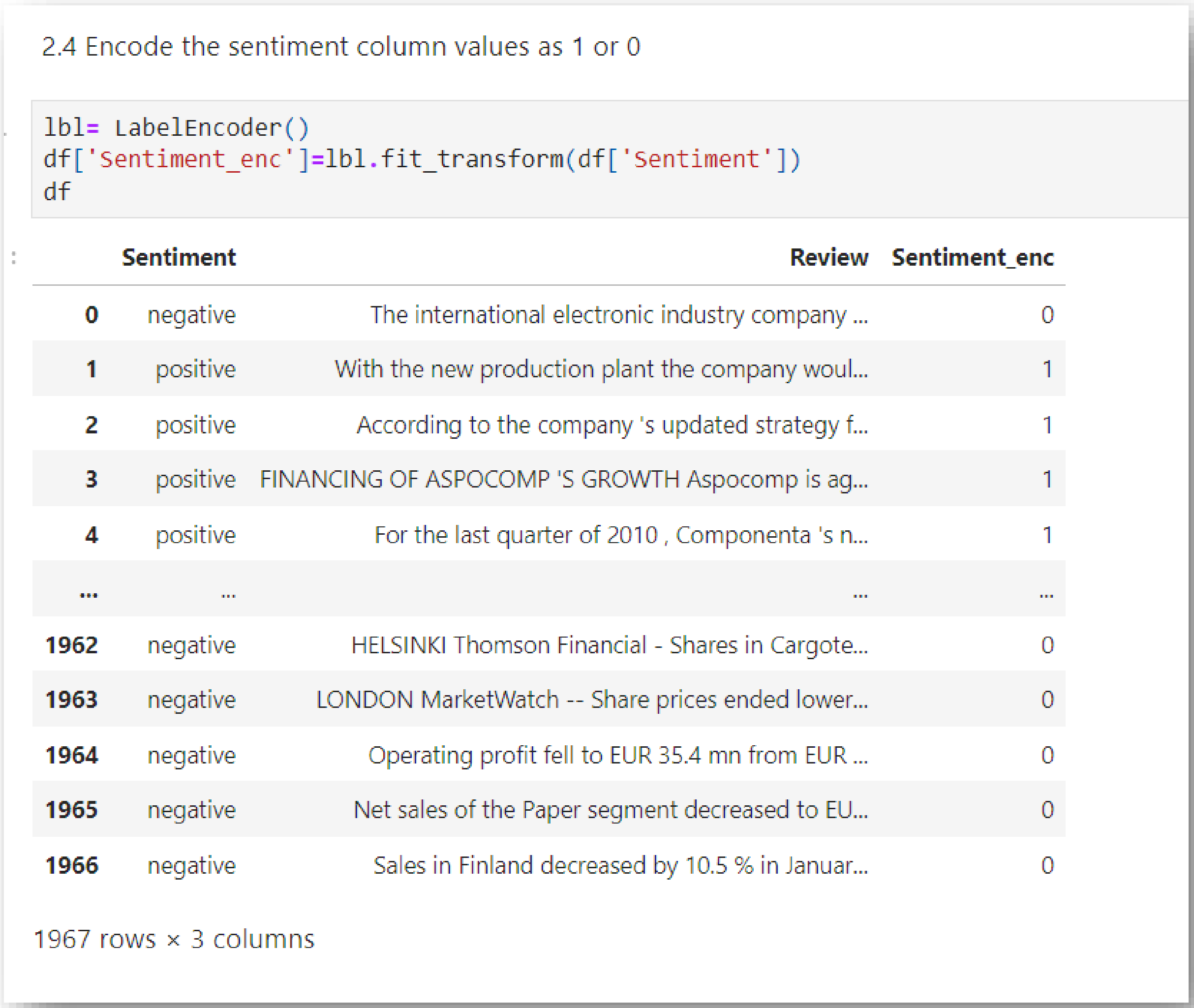
	Sentiment	Review
0	negative	The international electronic industry company ...
1	positive	With the new production plant the company woul...
2	positive	According to the company 's updated strategy f...
3	positive	FINANCING OF ASPOCOMP 'S GROWTH Aspocomp is ag...
4	positive	For the last quarter of 2010 , Componenta 's n...
5	positive	In the third quarter of 2010 , net sales incre...
6	positive	Operating profit rose to EUR 13.1 mn from EUR ...
7	positive	Operating profit totalled EUR 21.1 mn , up fro...
8	positive	TeliaSonera TLSN said the offer is in line wit...
9	positive	STORA ENSO , NORSKE SKOG , M-REAL , UPM-KYMMEN...



2.3 Display the information about the data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1967 entries, 0 to 1966
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   Sentiment  1967 non-null   object
 1   Review     1967 non-null   object
dtypes: object(2)
memory usage: 30.9+ KB
```


2.4 Encode the sentiment column values as 1 or 0

```
lbl= LabelEncoder()
df['Sentiment_enc']=lbl.fit_transform(df['Sentiment'])
df
```

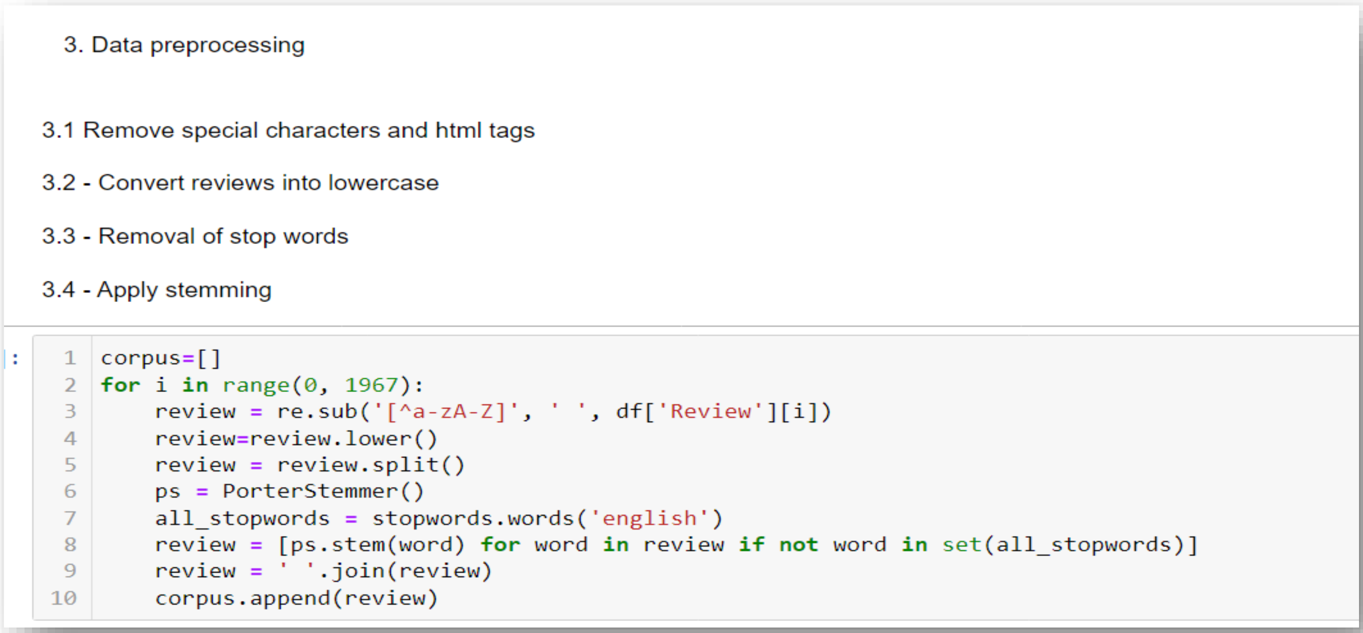
	Sentiment	Review	Sentiment_enc
0	negative	The international electronic industry company ...	0
1	positive	With the new production plant the company woul...	1
2	positive	According to the company 's updated strategy f...	1
3	positive	FINANCING OF ASPOCOMP 'S GROWTH Aspocomp is ag...	1
4	positive	For the last quarter of 2010 , Componenta 's n...	1
...	...	...	...
1962	negative	HELSINKI Thomson Financial - Shares in Cargote...	0
1963	negative	LONDON MarketWatch -- Share prices ended lower...	0
1964	negative	Operating profit fell to EUR 35.4 mn from EUR ...	0
1965	negative	Net sales of the Paper segment decreased to EU...	0
1966	negative	Sales in Finland decreased by 10.5 % in Januar...	0

1967 rows × 3 columns

Inference:

Load and explore the data:

- Load the dataset using the pd.read_csv() function.
- Display the first ten rows of the dataset.
- Use the info() function to get an overview of the dataset.
- Encode the Sentiment column, converting positive to 1 and negative to 0.
- In this step, we utilized label encoding to convert sentiment labels ('positive' and 'negative') into numerical values (1 and 0).
- Label Encoding: Label encoding is a technique used to convert categorical labels into numerical values, making it easier for machine learning algorithms to process them.



3. Data preprocessing

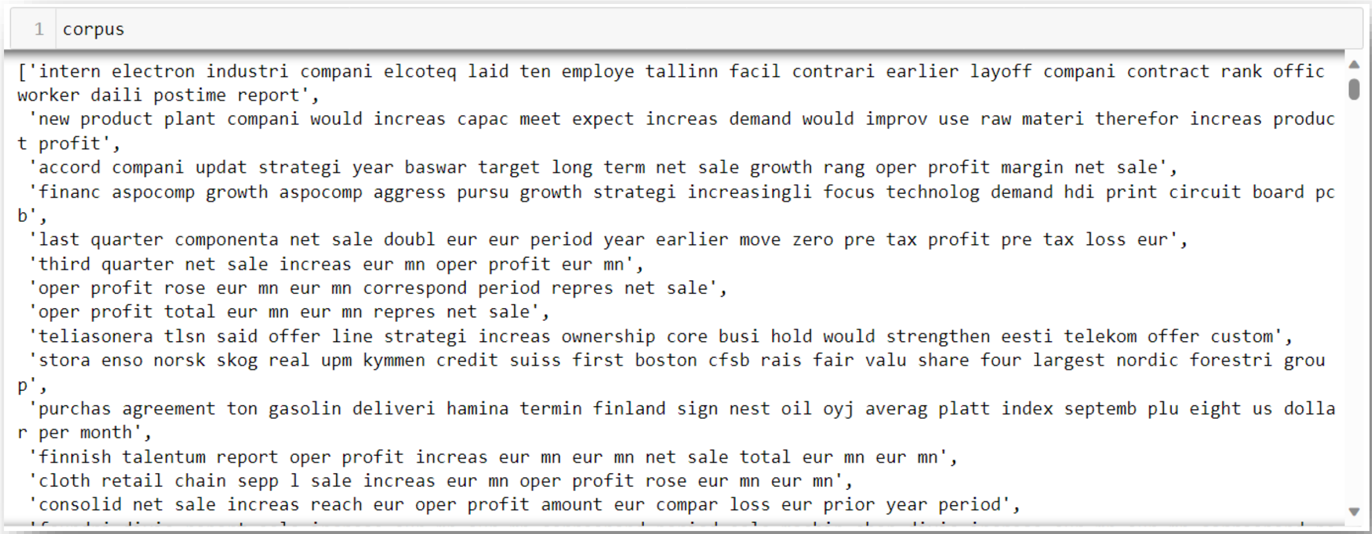
3.1 Remove special characters and html tags

3.2 - Convert reviews into lowercase

3.3 - Removal of stop words

3.4 - Apply stemming

```python
corpus=[]
for i in range(0, 1967):
    review = re.sub('[^a-zA-Z]', ' ', df['Review'][i])
    review=review.lower()
    review = review.split()
    ps = PorterStemmer()
    all_stopwords = stopwords.words('english')
    review = [ps.stem(word) for word in review if not word in set(all_stopwords)]
    review = ' '.join(review)
    corpus.append(review)
```



```
corpus
```

```
['intern electron industri compani elcoteq laid ten employe tallinn facil contrari earlier layoff compani contract rank offic
worker daili postime report',
 'new product plant compani would increas capac meet expect increas demand would improv use raw materi therefor increas produc
t profit',
 'accord compani updat strategi year baswar target long term net sale growth rang oper profit margin net sale',
 'financ aspocomp growth aspocomp aggress pursu growth strategi increasingli focus technolog demand hdi print circuit board pc
b',
 'last quarter componenta net sale doubl eur eur period year earlier move zero pre tax profit pre tax loss eur',
 'third quarter net sale increas eur mn oper profit eur mn',
 'oper profit rose eur mn eur mn correspond period repres net sale',
 'oper profit total eur mn eur mn repres net sale',
 'teliasonera tlsn said offer line strategi increas ownership core busi hold would strengthen eesti telekom offer custom',
 'stora enso norsk skog real upm kymmen credit suiss first boston cfsb rais fair valu share four largest nordic forestri grou
p',
 'purchas agreement ton gasolin deliveri hamina termin finland sign nest oil oyj averag platt index septemb plu eight us dolla
r per month',
 'finnish talentum report oper profit increas eur mn eur mn net sale total eur mn eur mn',
 'cloth retail chain sepp l sale increas eur mn oper profit rose eur mn eur mn',
 'consolid net sale increas reach eur oper profit amount eur compar loss eur prior year period',
```

Inference:

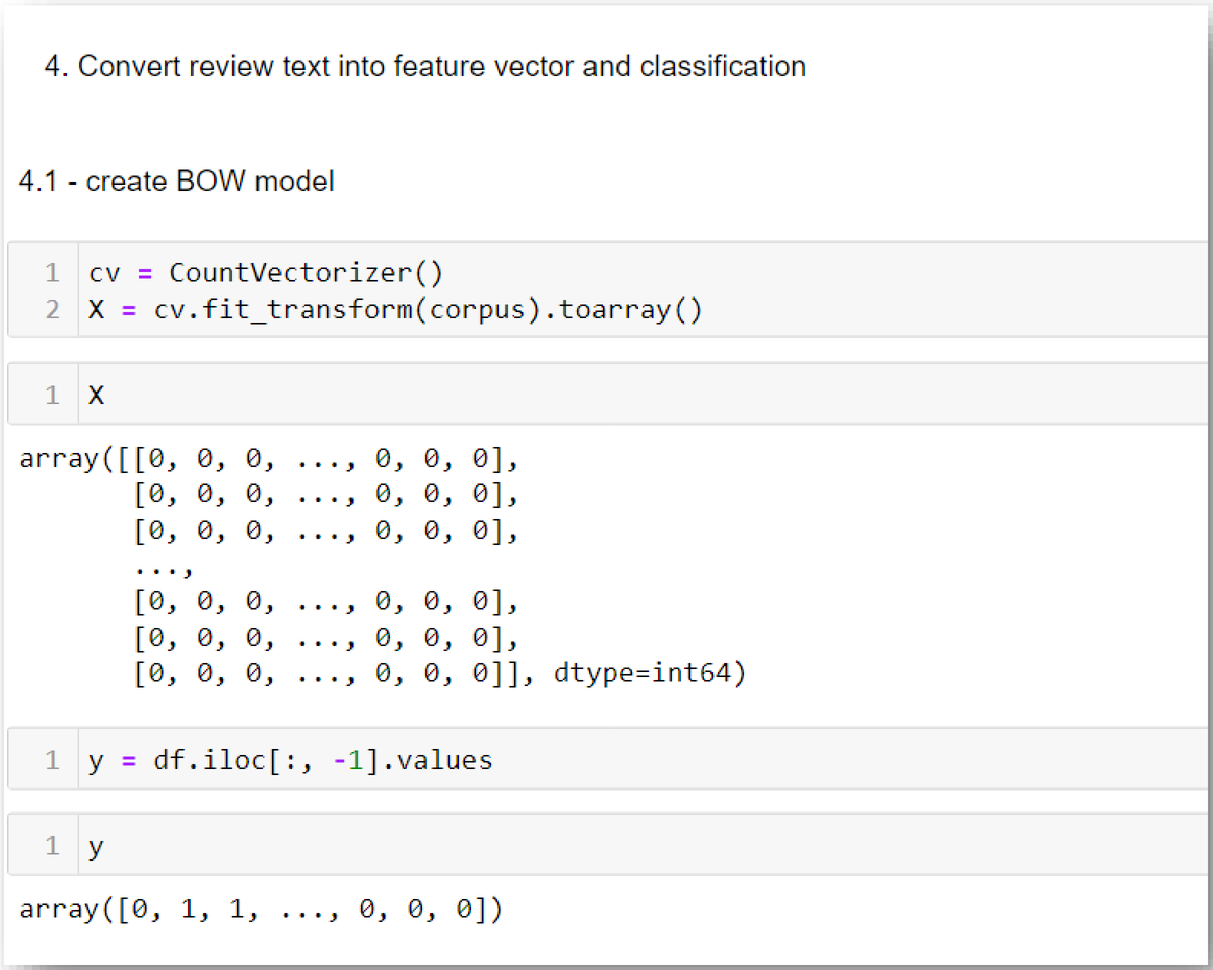
Preprocess the text data:

- Remove special characters and HTML tags from the review texts.
- Convert all text to lowercase.
- Apply stemming using the Porter Stemmer.
- Remove stop words using regular expressions.

Explanation:

- Removing special characters and HTML tags: This helps to clean the data and make it more consistent.

- Converting all text to lowercase: This ensures that all words are treated equally, regardless of their capitalization.
- Applying stemming using the Porter Stemmer: This reduces words to their root form, which can help to improve the performance of text processing algorithms.
- Removing stopwords using regular expressions: Stop words are common words that do not add much meaning to the text, such as "the", "and", and "in". Removing stop words can help to reduce noise and focus on more meaningful words.



4. Convert review text into feature vector and classification

4.1 - create BOW model

```
cv = CountVectorizer()
X = cv.fit_transform(corpus).toarray()
```

```
X
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

```
y = df.iloc[:, -1].values
```

```
y
```

```
array([0, 1, 1, ..., 0, 0, 0])
```

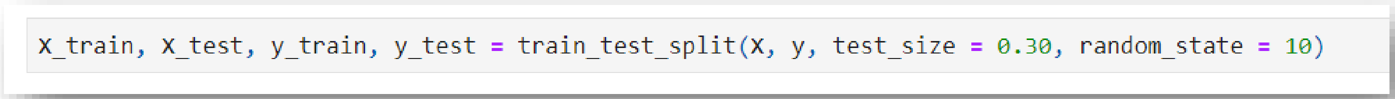
Inference:

Step 4: Build and evaluate classification models

4.1 Create a Bag of Words (BoW) model:

- Purpose: To create a numerical representation of the text data that can be used by machine learning algorithms.
- Tool: CountVectorizer
- Output: A feature matrix X where each row represents a review and each column represents a word in the vocabulary. The value of each cell in the matrix is the frequency of the corresponding word in the review.
- Explanation:
- The BoW model is a simple way to represent text data numerically.

- It works by counting the frequency of each word in a document.
- This results in a feature vector for each document, where each feature represents a word in the vocabulary and the value of each feature is the frequency of the corresponding word in the document.



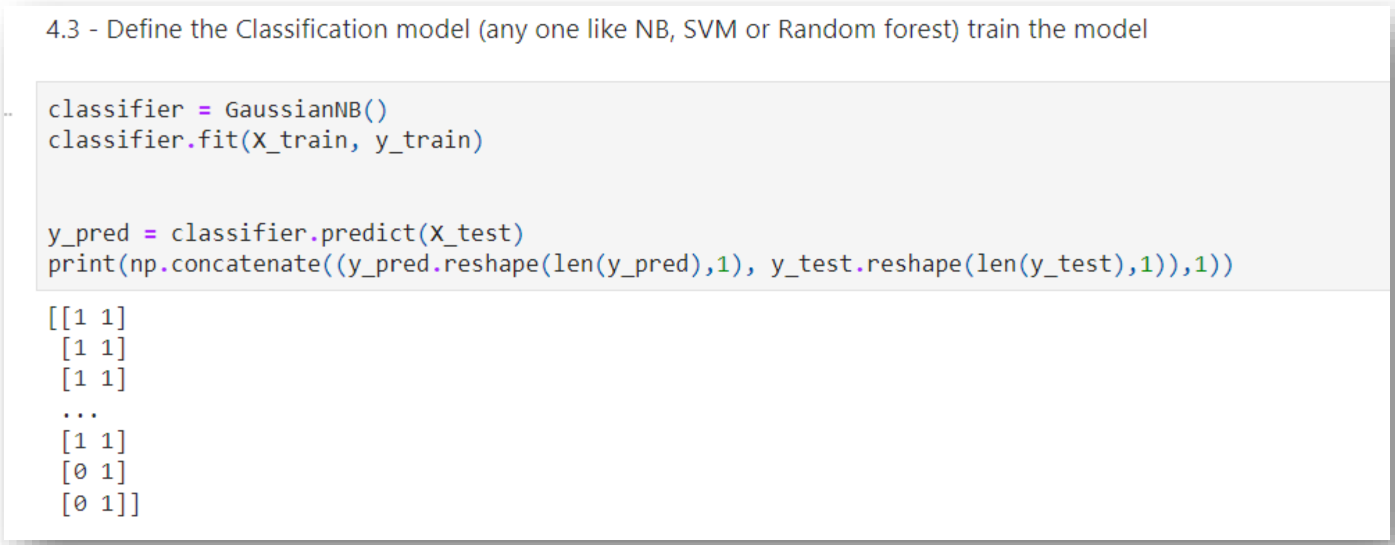
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 10)
```

Inference:

4.2 Split the data into training and test sets:

Data splitting:

- Purpose: To divide the data into training and test sets to evaluate the performance of the classification model.
- Tool: train_test_split()
- Splitting: Random and representative split
- Sets: Training set and test set
- Features: Used for predictions
- Labels: What we're trying to predict
- Importance: Separating features and labels is essential for evaluating the model's effectiveness.
- Training set: The model learns and improves its predictive capabilities on the training set.
- Test set: The model's performance is evaluated on unseen data in the test set.
- Validation: Data splitting ensures that the model can make accurate predictions in real-world scenarios.



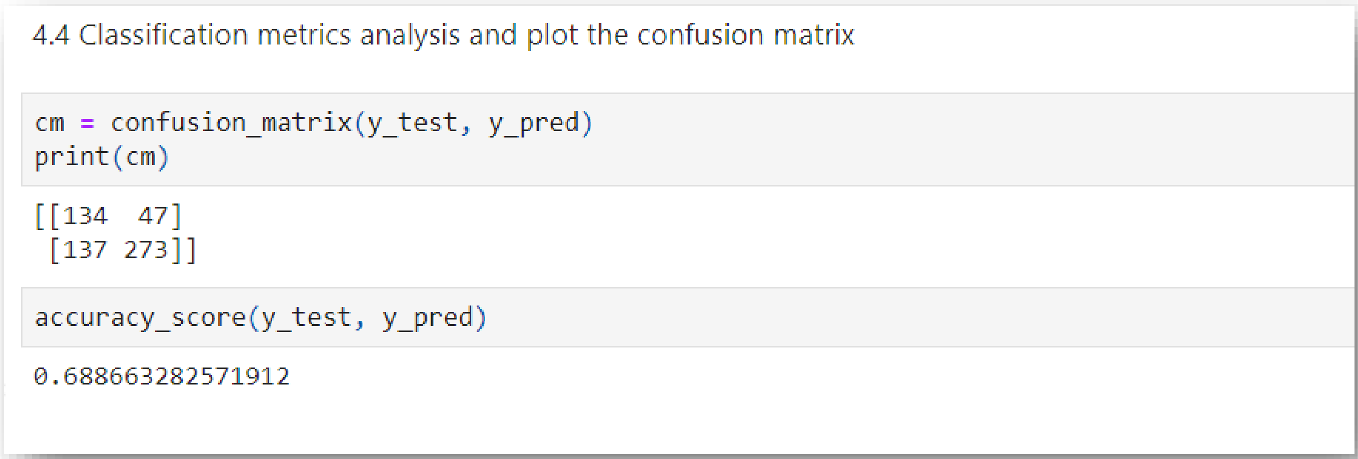
4.3 - Define the Classification model (any one like NB, SVM or Random forest) train the model

```
classifier = GaussianNB()
classifier.fit(X_train, y_train)


y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[[1 1]
 [1 1]
 [1 1]
 ...
 [1 1]
 [0 1]
 [0 1]]
```

4.3 Define and train a Random Forest classifier:

- Define and instantiate a Random Forest classifier with 30 decision trees.
- Train the model on the training data using fit().
- Make predictions on the test set using predict().



4.4 Classification metrics analysis and plot the confusion matrix

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[134  47]
 [137 273]]
```

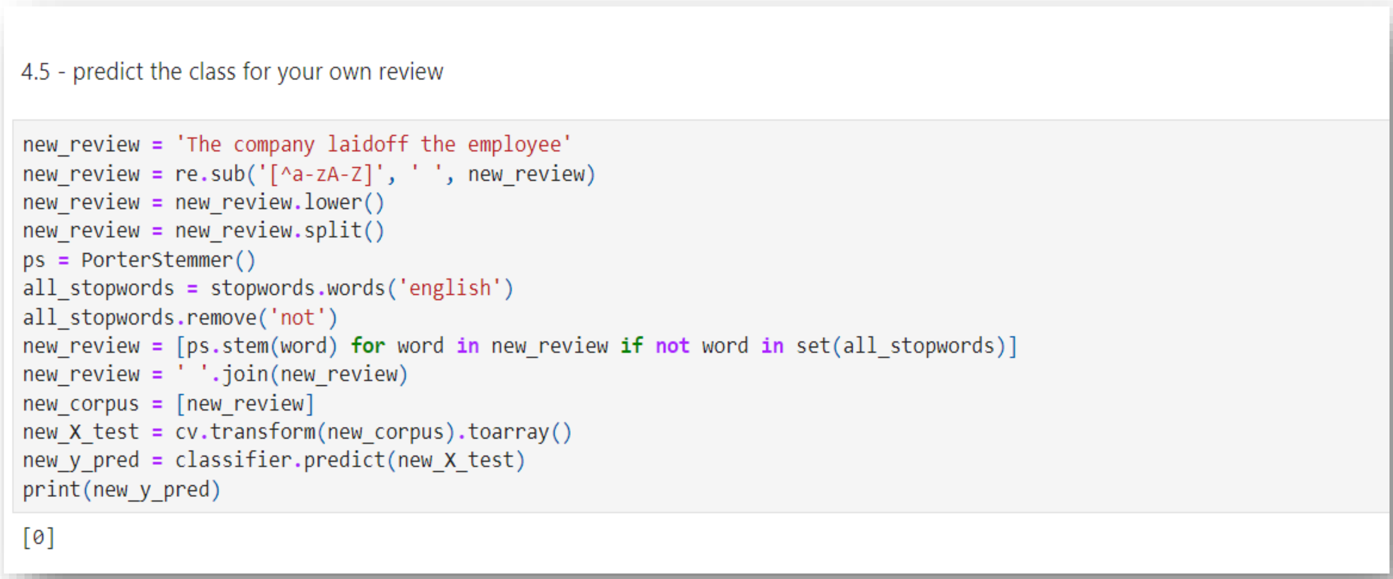
```
accuracy_score(y_test, y_pred)
```

```
0.688663282571912
```

Inference:

4.4 Evaluate the Random Forest classifier:

- Calculate classification metrics such as precision, recall, F1-score, and support using classification_report.
- These metrics provide a comprehensive evaluation of the model's performance.



4.5 - predict the class for your own review

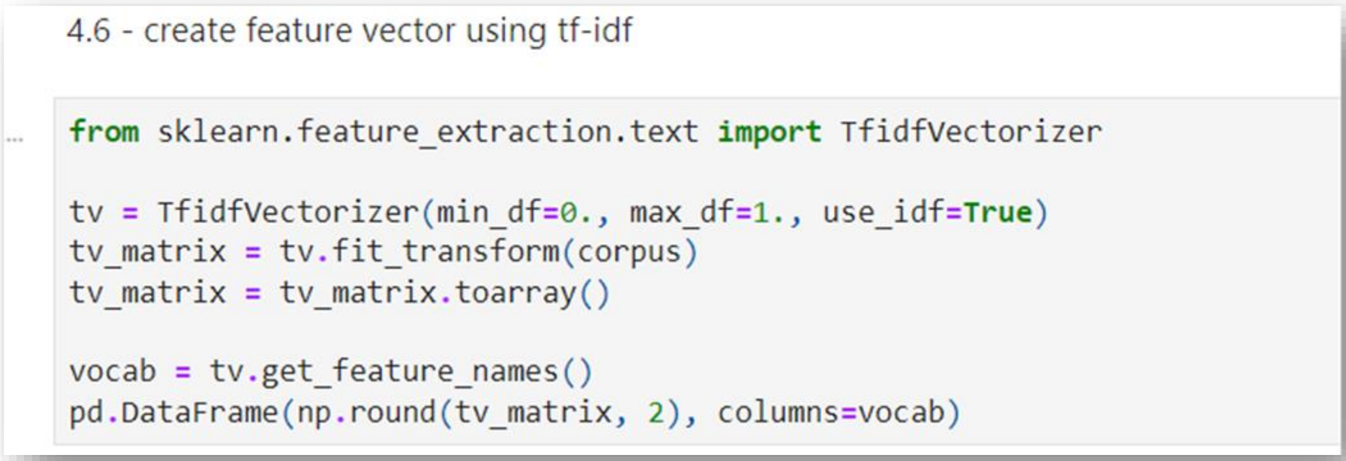
```
new_review = 'The company laidoff the employee'
new_review = re.sub('[^a-zA-Z]', ' ', new_review)
new_review = new_review.lower()
new_review = new_review.split()
ps = PorterStemmer()
all_stopwords = stopwords.words('english')
all_stopwords.remove('not')
new_review = [ps.stem(word) for word in new_review if not word in set(all_stopwords)]
new_review = ' '.join(new_review)
new_corpus = [new_review]
new_X_test = cv.transform(new_corpus).toarray()
new_y_pred = classifier.predict(new_X_test)
print(new_y_pred)
```

```
[0]
```

Inference:

4.5 Predict the class for a new review (Naive Bayes):

- Preprocess the new review text in the same way as the dataset.
- Convert the preprocessed text into a feature vector using the BoW model.
- Pass the feature vector to the Naive Bayes classifier for prediction.
- Print the predicted class (0 for negative or 1 for positive) for the new review.
- We explained the scenario of predicting sentiment for new reviews, where a negative word like "laidoff" leads to a prediction of 0 (negative), and a positive word like "increase" results in a prediction of 1 (positive).
- This illustrates how the model classifies text based on learned patterns.



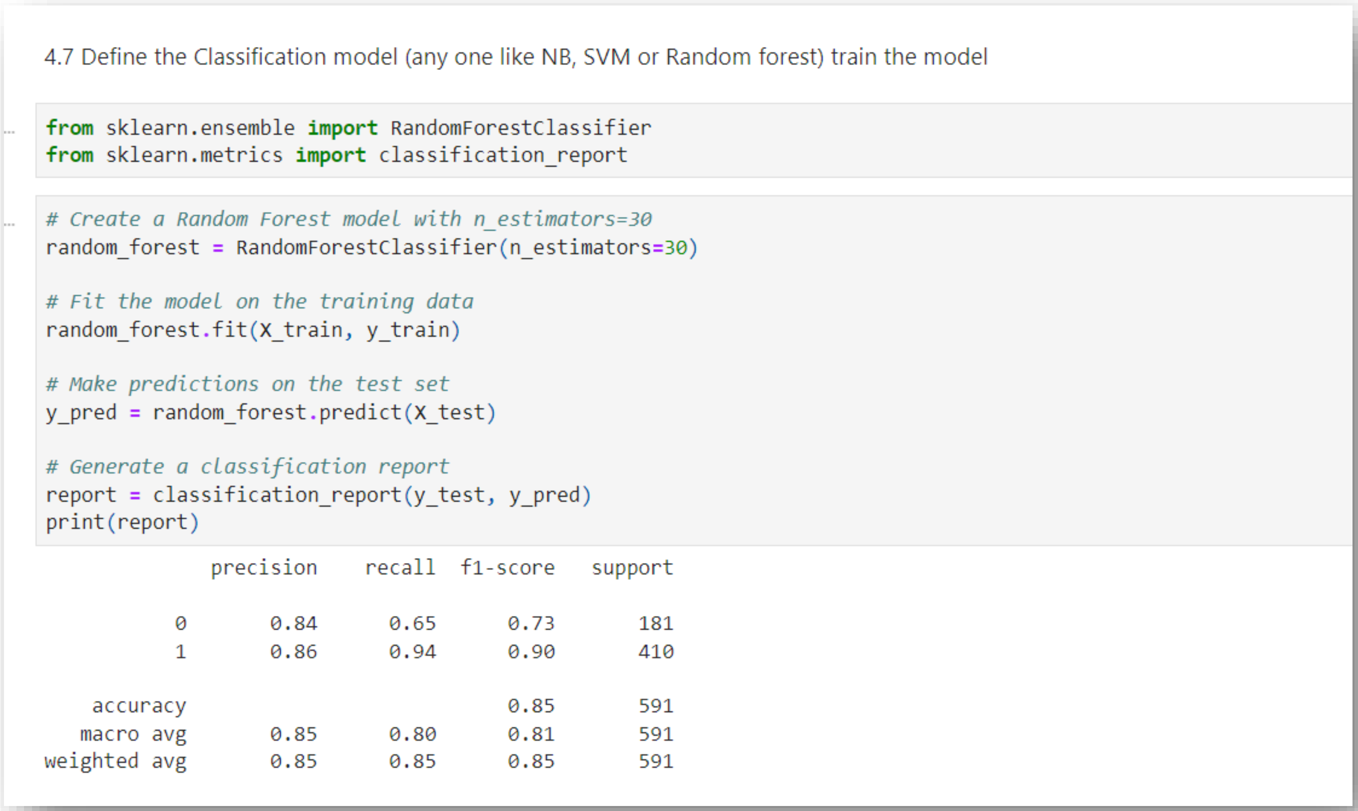
4.6 - create feature vector using tf-idf

```
from sklearn.feature_extraction.text import TfidfVectorizer

tv = TfidfVectorizer(min_df=0., max_df=1., use_idf=True)
tv_matrix = tv.fit_transform(corpus)
tv_matrix = tv_matrix.toarray()

vocab = tv.get_feature_names()
pd.DataFrame(np.round(tv_matrix, 2), columns=vocab)
```

Inference:

4.6 Create a Term Frequency-Inverse Document Frequency (TF-IDF) feature vector:

- Use TfidfVectorizer to transform the preprocessed text data into a TF-IDF feature matrix.
- The resulting matrix represents the importance of each term (word) in the corpus relative to the documents (reviews).
- TF (Term Frequency): Captures word importance within a document. It counts word occurrences without considering other documents.
- IDF (Inverse Document Frequency): Measures word distinctiveness. Words appearing in many documents get lower IDF weights, indicating their commonality.


4.7 Define the Classification model (any one like NB, SVM or Random forest) train the model

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
```

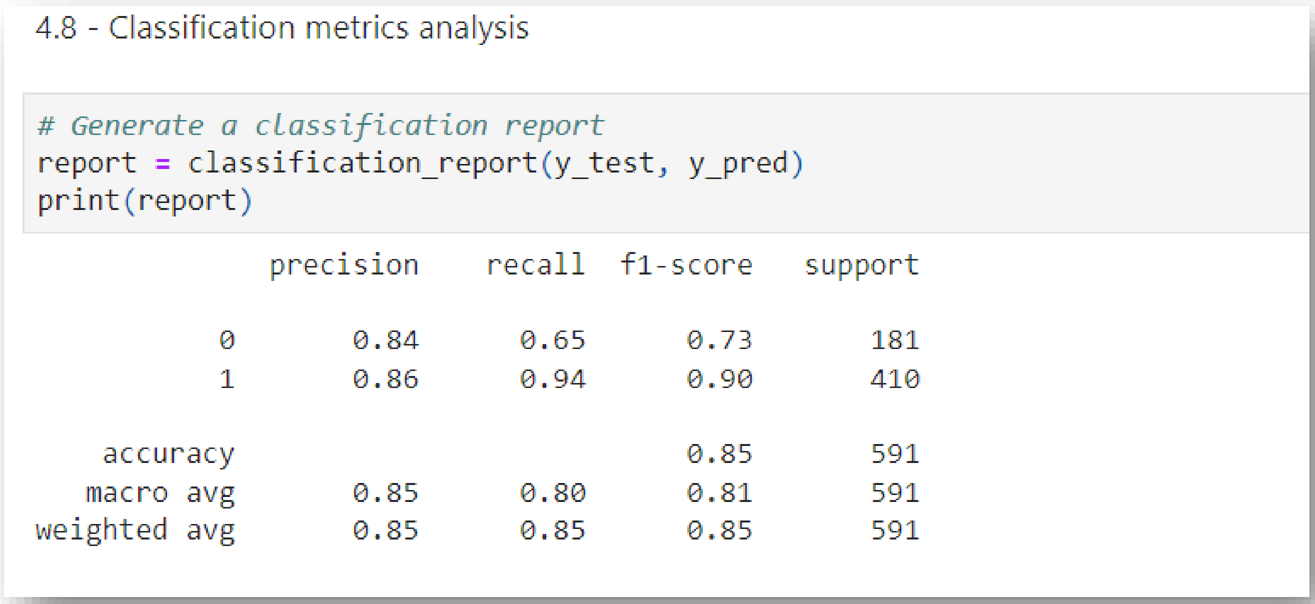
```
# Create a Random Forest model with n_estimators=30
random_forest = RandomForestClassifier(n_estimators=30)

# Fit the model on the training data
random_forest.fit(X_train, y_train)

# Make predictions on the test set
y_pred = random_forest.predict(X_test)

# Generate a classification report
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.84      0.65      0.73       181
           1       0.86      0.94      0.90       410

    accuracy                           0.85       591
   macro avg       0.85      0.80      0.81       591
weighted avg       0.85      0.85      0.85       591
```

Inference:

4.7 Define and train a Random Forest classifier:

1. We choose the Random Forest classifier from the sklearn library and set the number of estimators (trees) to 30.
2. Model training: We fit the model to the training data (X_train and y_train) using the fit() method.
3. Model prediction: We use the trained model to make predictions on the test set (X_test), storing the results in y_pred.
4. Model evaluation: We generate a classification report to evaluate the model's performance.
5. We can use various classification metrics to evaluate the model's performance, such as precision, recall, F1-score, and accuracy.


4.8 - Classification metrics analysis

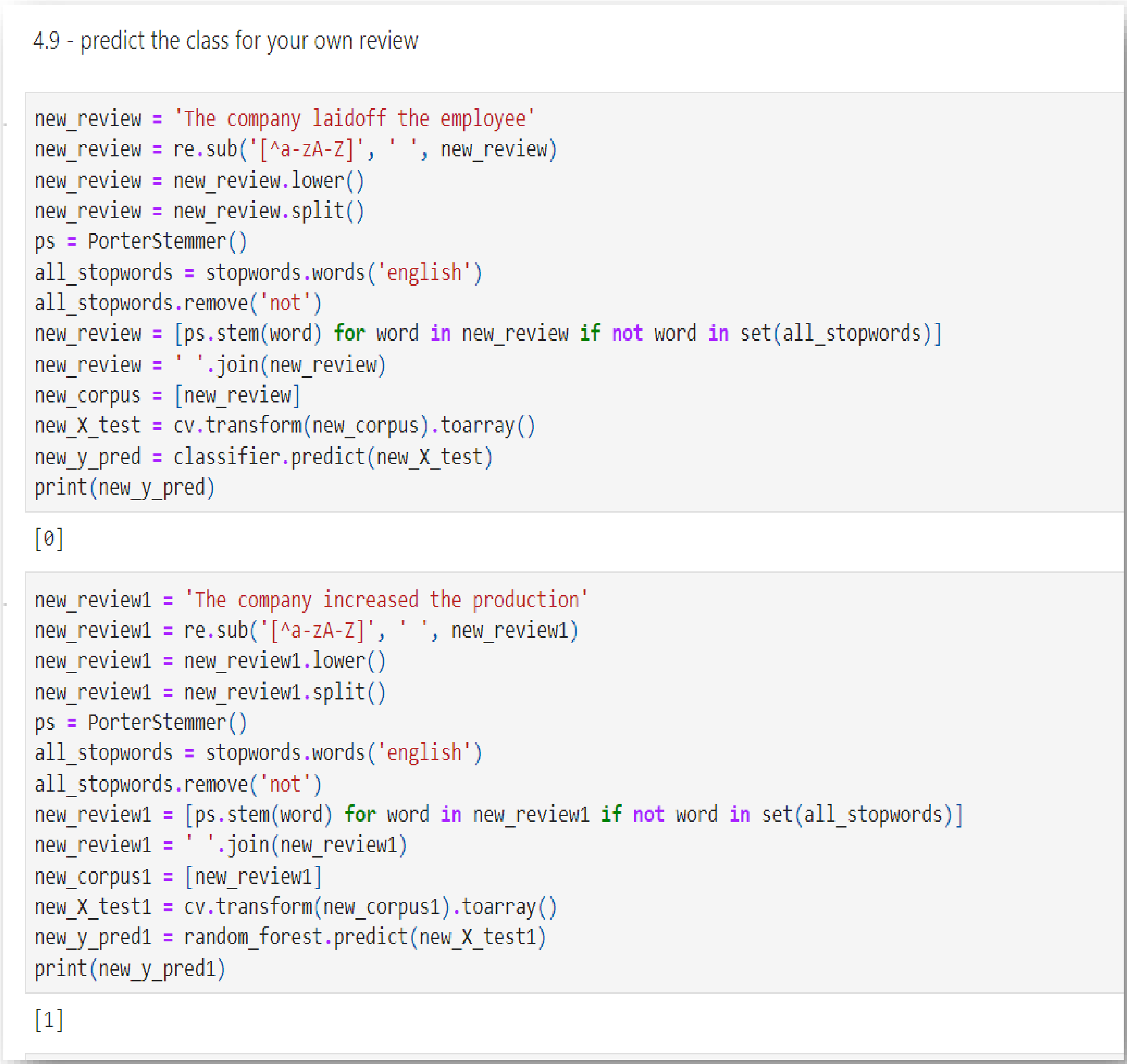
```
# Generate a classification report
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.84      0.65      0.73       181
           1       0.86      0.94      0.90       410

    accuracy                           0.85       591
   macro avg       0.85      0.80      0.81       591
weighted avg       0.85      0.85      0.85       591
```

Inference:

4.8 Evaluate the Random Forest classifier:

- This step is dedicated to analyzing the classification metrics generated by the model.
- We display a classification report that includes metrics such as precision, recall, F1-score, and support for both classes (0 and 1).
- The report provides an in-depth assessment of the model's performance, highlighting its ability to correctly classify instances from both classes. Classification metric analysis:
- The classification metrics reported in the table are:
- Precision: The fraction of predicted positive reviews that were actually positive.
- Recall: The fraction of actual positive reviews that were predicted to be positive.
- F1-score: A harmonic mean of precision and recall.
- Accuracy: The overall fraction of correct predictions.
- The precision, recall, and F1-score for the positive class (1) are all high, indicating that the model is good at identifying positive reviews. The accuracy is also high, indicating that the model is good at predicting the sentiment of reviews in general.


4.9 - predict the class for your own review

```
new_review = 'The company laidoff the employee'
new_review = re.sub('[^a-zA-Z]', ' ', new_review)
new_review = new_review.lower()
new_review = new_review.split()
ps = PorterStemmer()
all_stopwords = stopwords.words('english')
all_stopwords.remove('not')
new_review = [ps.stem(word) for word in new_review if not word in set(all_stopwords)]
new_review = ' '.join(new_review)
new_corpus = [new_review]
new_X_test = cv.transform(new_corpus).toarray()
new_y_pred = classifier.predict(new_X_test)
print(new_y_pred)
```

```
[0]
```

```
new_review1 = 'The company increased the production'
new_review1 = re.sub('[^a-zA-Z]', ' ', new_review1)
new_review1 = new_review1.lower()
new_review1 = new_review1.split()
ps = PorterStemmer()
all_stopwords = stopwords.words('english')
all_stopwords.remove('not')
new_review1 = [ps.stem(word) for word in new_review1 if not word in set(all_stopwords)]
new_review1 = ' '.join(new_review1)
new_corpus1 = [new_review1]
new_X_test1 = cv.transform(new_corpus1).toarray()
new_y_pred1 = random_forest.predict(new_X_test1)
print(new_y_pred1)
```

```
[1]
```

Inference:

4.9 Predict the class for a new review :

- In this scenario, we demonstrate how to use the trained Random Forest model to predict the sentiment of new reviews.
- We preprocess a new review text, 'The company laidoff the employee,' by removing special characters, converting to lowercase, and stemming, similar to the preprocessing applied to the dataset.
- After preprocessing, we transform the new review into a feature vector using the CountVectorizer (cv) created earlier.

- We then pass this feature vector to the Random Forest model for prediction and print the predicted class.

- For example, in this case, the model predicts a sentiment of '0,' indicating a negative sentiment for the new review.

- We repeat the same process for another new review, 'The company increased the production,' and the model predicts a sentiment of '1,' indicating a positive sentiment.

Summary:

  - Imported essential libraries for data analysis and machine learning.
  - Loaded and explored the dataset.
  - Encoded sentiment column values as 1 for positive and 0 for negative sentiments.
  - Preprocessed the data by removing special characters and HTML tags, converting reviews to lowercase, eliminating stopwords, and applying stemming.
  - Created feature vectors through both Bag of Words (BoW) and TF-IDF approaches.
  - Trained and evaluated machine learning models, including Random Forest, for sentiment classification.
  - Analyzed comprehensive classification metrics to assess model performance.
  - Predicted sentiment for new reviews, showcasing the practical application of the trained models.

Conclusion:

This project demonstrated the power of text analysis and machine learning in classifying sentiment from textual data. The insights gained from this project can be leveraged to improve customer satisfaction, product development, and other business-critical decisions.