import pandas as pd.

Exercise : 1

1. Open the file, 'rotten_tomato_train.csv',

```
rotten_tomato_train = pd.read_csv('rotten_tomato_train.tsv',
                                  sep='\t')
rotten_tomato_train_train.head()
```

2) `rotten_tomato_train.tail()`

```
rotten_tomato_train.shape.

rotten_tomato_train.describe

rotten_tomato_train.columns
```

3) `review = rotten_tomato_train.groupby('sentiment').count()`

```
review.phrase.
```

Exe : 2

1)

```
a = rotten_tomato_train.loc[rotten_tomato_train.
                            sentiment==0]
b = rotten_tomato_train.loc[rotten_tomato_train.
                            sentiment==1]
c = rotten_tomato_train.loc[rotten_tomato_train.
                            sentiment==2]
d = rotten_tomato_train.loc[:rotten_tomato_train.
                            sentiment==3]
e = rotten_tomato_train.loc[:rotten_tomato_train.
                            sentiment==4]

Small_rotten_train = pd.concat([a[:200], b[:200],
                                c[:200], d[:200], e[:200]])
```

Natural Language Processing Lab
Lab7. Sentiment Analysis on Movie Reviews

In this lab, you will build Multinomial Naïve Bayes model for movie reviews from Rotton Tomotto Dataset.

EXERCISE-1

1. Open the file, 'rotten_tomato_train.tsv' and read into a DataFrame

2. Print the basic statistics such as head, shape, describe, and columns

3. How many reviews exist for each sentiment?

EXERCISE-2

1. Extract 200 reviews for each sentiment, store them into a new dataframe and create a smaller dataset. Save this dataframe in a new file, say, **"small_rotten_train.csv"**.

EXERCISE-3

1. Open the file, **"small_rotten_train.csv".**

2. The review text are stored in "Phrase" column. Extract that into a separate DataFrame, say "X".

3. The "sentiment" column is your target, say "y".

4. Perform pre-processing: convert into lower case, remove stop words and lemmatize. The following function will help you.

```
def clean_review(review):
      tokens = review.lower().split()
      filtered_tokens = [lemmatizer.lemmatize(w)
            for w in tokens if w not in stop_words]
      return " ".join(filtered_tokens)
```

5. Apply the above function to X

6. Split X and y for training and testing (Use 20% for testing).

7. Create TfidfVectorizer as below and perfrom vectorization on X_train using fit_perform() method.

```
TfidfVectorizer(min_df=3, max_features=None,
      ngram_range=(1, 2), use_idf=1)
```

8. Create MultinomialNB model and perform training using X_train_lemmartized and y_train.

9. Perform validation on X_test lemmatized and predict output.

10. Print classification_report and accuracy score.

Exe:3

1) small_rotten_train

2) x = small_rotten_train.phrase

3) y = small_rotten_train.sentiment

4) import nltk
from nltk.corpus import stopwords.
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
def clean_review(review):
tokens = review.lower().split()
filtered_tokens = [lemmatizer.lemmatize(w) for w in tokens. if w not in stop.
return " ".join(filtered_tokens)

5) Apply the above function:
temp = X.tolist()
~~tokens = review.lower().split()~~
~~filtered_tokens = [lemmatizer.lemmatize(w) for w in tokens. if w not in~~
data = []
for i in temp:
data.append(clean_review(i))
n_X = pd.Series(data)

EXERCISE-4

1. Open, 'rotten_tomato_test.tsv' file into dataframe
2. Clean this test data, using the function `clean_review()`, as before.
3. Build TFIDF values using transform() method.
4. Perform prediction using predict() method.

6)
```
x_train, x_test, y_train, y_test = train_test_split
                    (X, y, train_size = 0.8,
```

7)
```
from sklearn.feature_extraction.text import
                    TfidfVectorizer.
tf = TfidfVectorizer(min_df=3, max_features=None,
                    ngram_range=(1,2), use_idf=1)
tf.
m = tf.fit_transform(x_train)
m.shape.
```

8)
```
from sklearn.feature_extraction.text import
                    CountVectorizer.
cv = CountVectorizer()
x_train_dtm = cv.fit_transform(x_train)
x_test_dtm = cv.transform(x_test)
from sklearn.naive_bayes import MultinomialNB
    clf = MultinomialNB()
    clf.fit(x_train_dtm, y_train)
```

a)

```
y_real_pred = clf.predict(x_test_dtm)
y_real_pred
```

b)

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_real_pred))
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_real_pred)
```

Exercise-4

```
rotten_tomato_test = pd.read_csv('rotten_tomato_test.tsv',
                                 sep='\t')
rotten_tomato_test.shape
x_ = rotten_tomato_test.phrase
t_temp = x_.tolist()
t_fax = []
for i in t_temp:
    t_fax.append(clean_review(i))
nt_x = pd.Series(t_fax)
```

NOTES

```
nt_X
from.sklearn.feature_extraction.text import
                                TfidfVectorizer
tf2=TfidfVectorizer(use_idf=True, n_gram_range=
                                (1,3), min_df=1)
tf2.
my2 = tf2.fit_transform(nt_X)
my2
```