In [ ]:

```
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from textblob import Word
import re
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score
```

In [ ]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

data = pd.read_csv('NLP_Lab6_text_emotion.csv')
```

In [ ]:

```
data.head(10)
```

In [ ]:

```
data = data.drop('author', axis=1)
```

In [ ]:

```
data.head(10)
```

In [ ]:

```
# Making all letters lowercase
data['content'] = data['content'].apply(lambda x: " ".join(x.lower() for x in
x.split()))
```

In [ ]:

```
# Removing Punctuation, Symbols
data['content'] = data['content'].str.replace('[^\w\s]',' ')
```

In [ ]:

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

In [ ]:

```
# Removing Stop Words using NLTK
stop = stopwords.words('english')
data['content'] = data['content'].apply(lambda x: " ".join(x for x in x.split
() if x not in stop))
```

In [ ]:

```
#Lemmatisation
data['content'] = data['content'].apply(lambda x: " ".join([Word(word).lemmati
ze() for word in x.split()]))
#Correcting Letter Repetitions
```

In [ ]:

```
def de_repeat(text):
    pattern = re.compile(r"(.)\1{2,}")
    return pattern.sub(r"\1\1", text)

data['content'] = data['content'].apply(lambda x: " ".join(de_repeat(x) for x
in x.split()))
```

In [ ]:

```
# Code to find the top 10,000 rarest words appearing in the data
freq = pd.Series(' '.join(data['content']).split()).value_counts()[-10000:]

# Removing all those rarely appearing words from the data
freq = list(freq.index)
data['content'] = data['content'].apply(lambda x: " ".join(x for x in x.split
() if x not in freq))
```

In [ ]:

```
#Encoding output labels
lbl_enc = preprocessing.LabelEncoder()
y = lbl_enc.fit_transform(data.sentiment.values)
```

In [ ]:

```
# Splitting into training and testing data in 80:20 ratio
X_train, X_val, y_train, y_val = train_test_split(data.content.values, y, stra
tify=y,
                                                  random_state=42, test_size=
0.2, shuffle=True)
```

In [ ]:

```
# Extracting TF-IDF parameters
tfidf = TfidfVectorizer(max_features=1000, analyzer='word',ngram_range=(1,3))
X_train_tfidf = tfidf.fit_transform(X_train)
X_val_tfidf = tfidf.fit_transform(X_val)

# Extracting Count Vectors Parameters
count_vect = CountVectorizer(analyzer='word')
count_vect.fit(data['content'])
X_train_count =  count_vect.transform(X_train)
X_val_count =  count_vect.transform(X_val)
```

In [ ]:

```
# Model 1: Multinomial Naive Bayes Classifier
nb = MultinomialNB()
nb.fit(X_train_tfidf, y_train)
y_pred = nb.predict(X_val_tfidf)
print('naive bayes tfidf accuracy %s' % accuracy_score(y_pred, y_val))
```

In [ ]:

```
# Model 2: Linear SVM
lsvm = SGDClassifier(alpha=0.001, random_state=5, max_iter=15, tol=None)
lsvm.fit(X_train_tfidf, y_train)
y_pred = lsvm.predict(X_val_tfidf)
print('svm using tfidf accuracy %s' % accuracy_score(y_pred, y_val))
# svm tfidf accuracy 0.5404624277456648

# Model 3: logistic regression
logreg = LogisticRegression(C=1)
logreg.fit(X_train_tfidf, y_train)
y_pred = logreg.predict(X_val_tfidf)
print('log reg tfidf accuracy %s' % accuracy_score(y_pred, y_val))
# log reg tfidf accuracy 0.5443159922928709

# Model 4: Random Forest Classifier
rf = RandomForestClassifier(n_estimators=500)
rf.fit(X_train_tfidf, y_train)
y_pred = rf.predict(X_val_tfidf)
print('random forest tfidf accuracy %s' % accuracy_score(y_pred, y_val))
# random forest tfidf accuracy 0.5385356454720617

## Building models using count vectors feature
# Model 1: Multinomial Naive Bayes Classifier
nb = MultinomialNB()
nb.fit(X_train_count, y_train)
y_pred = nb.predict(X_val_count)
print('naive bayes count vectors accuracy %s' % accuracy_score(y_pred, y_val))
# naive bayes count vectors accuracy 0.7764932562620424

# Model 2: Linear SVM
lsvm = SGDClassifier(alpha=0.001, random_state=5, max_iter=15, tol=None)
lsvm.fit(X_train_count, y_train)
y_pred = lsvm.predict(X_val_count)
print('lsvm using count vectors accuracy %s' % accuracy_score(y_pred, y_val))
# lsvm using count vectors accuracy 0.7928709055876686

# Model 3: Logistic Regression
logreg = LogisticRegression(C=1)
logreg.fit(X_train_count, y_train)
y_pred = logreg.predict(X_val_count)
print('log reg count vectors accuracy %s' % accuracy_score(y_pred, y_val))
# log reg count vectors accuracy 0.7851637764932563

# Model 4: Random Forest Classifier
rf = RandomForestClassifier(n_estimators=500)
rf.fit(X_train_count, y_train)
y_pred = rf.predict(X_val_count)
print('random forest with count vectors accuracy %s' % accuracy_score(y_pred,
y_val))
# random forest with count vectors accuracy 0.7524084778420038
```

In [ ]:

```python
#Below are 8 random statements. The first 4 depict happiness. The last 4 depict sadness

tweets = pd.DataFrame(['I am very happy today! The atmosphere looks cheerful',
'Things are looking great. It was such a good day',
'Success is right around the corner. Lets celebrate this victory',
'Everything is more beautiful when you experience them with a smile!',
'Now this is my worst, okay? But I am gonna get better.',
'I am tired, boss. Tired of being on the road, lonely as a sparrow in the rain. I am tired of all the pain I feel',
'This is quite depressing. I am filled with sorrow',
'His death broke my heart. It was a sad day'])

# Doing some preprocessing on these tweets as done before
tweets[0] = tweets[0].str.replace('[^\w\s]',' ')
from nltk.corpus import stopwords
stop = stopwords.words('english')
tweets[0] = tweets[0].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
from textblob import Word
tweets[0] = tweets[0].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))

# Extracting Count Vectors feature from our tweets
tweet_count = count_vect.transform(tweets[0])

#Predicting the emotion of the tweet using our already trained linear SVM
tweet_pred = lsvm.predict(tweet_count)
print(tweet_pred)
## result
## [0 0 0 0 1 1 1 1]
```

In [ ]: