```
In [ ]: # Included Libraries
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.model_selection import KFold
        import re
        from nltk.tokenize import word_tokenize
        from nltk.corpus import stopwords
        from nltk.stem import SnowballStemmer
        import emoji
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.model_selection import train_test_split
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.metrics import *
        import pandas as pd
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.model_selection import cross_validate
        from sklearn.model_selection import cross_val_score, train_test_split
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.model_selection import train_test_split, cross_val_score
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.pipeline import make_pipeline
        from sklearn.svm import SVC
        import joblib
        from sklearn.model_selection import cross_val_predict
```

```
In [ ]: positive_tweets = pd.read_csv('data/Positive_Tweets.tsv', sep='\t', header=None)
        print("We have {} positive tweets in our dataset.".format(len(positive_tweets)))
        negative_tweets = pd.read_csv('data/Negative_Tweets.tsv', sep='\t', header=None)
        print("We have {} negative tweets in our dataset.".format(len(negative_tweets)))
        all_tweets = pd.concat([positive_tweets, negative_tweets])
        all_tweets.columns = ['Label', 'Tweet']
        print("We have {} tweets in our dataset.".format(len(all_tweets)))
        #A random sample of 5 tweets
        print("A sample of the data\n", all_tweets.sample(5))
```

```
We have 22761 positive tweets in our dataset.
We have 22514 negative tweets in our dataset.
We have 45275 tweets in our dataset.
A sample of the data
      Label                                                  Tweet
9954    pos  ... تمارين_المقاومة بأنواعها : حدد العضله المراد#
8791    neg                                  😰 لا عاد الحلم يخووف مره
5059    pos                                   اروى حبيبتي نامي وش مسهرك
9483    pos  ... اللهم اروي قبر ابي بفضلك ونعمتك بنورك و رحمتك
18815   pos  ...البليهي أختصر مشاعر جمهور #الهلال تجاههكك ي عب
```

Preproccesing Data

in this part we will preprocces the data and make it ready for the model

- remove mentions, hashtags, and links
- remove stop words, that are words that are not important in the context of the sentence
- remove punctuations
- replace emojis with their meaning

In [ ]:
```python
def procces_tweet(tweet: str) -> str:

    # Remove all mentions, hashtags, links, and special characters
    tweet = re.sub(r'http\S+|www\S+|https\S+|@\S+|#\S+', '', tweet, flags=re.MULTILINE)
    tweet = tweet.replace('أ', 'ا').replace('إ', 'ا').replace('آ', 'ا')
    tweet = tweet.replace('ة', 'ه').replace('ى', 'ي').replace('ؤ', 'و')

    # Remove all emojis and emoticons and replace them with unicode
    tweet = emoji.demojize(tweet)

    # Tokinize the tweet
    tokens = word_tokenize(tweet)

    # Remove all stop words, stop words are words that do not add any meaning to the sentence
    stop_words = set(stopwords.words('arabic'))
    filtered_tokens = [w for w in tokens if not w in stop_words]

    # Stemming, stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base c
    stemmer = SnowballStemmer('arabic')
    stemmed_tokens = [stemmer.stem(w) for w in filtered_tokens]

    return " ".join(stemmed_tokens)

all_tweets['Filtered_Tweet'] = all_tweets['Tweet'].apply(procces_tweet)
```

Feature Extraction

In this part we will extract the features from the data, we will use the TF-IDF to extract the features, and what TF-IDF does is that it gives a score to each word in the sentence, and the score is based on how many times the word appears in the sentence, and how many times the word appears in the whole dataset, and the score is calculated by this formula:

$$score = \frac{\text{number of times the word appears in the sentence}}{\text{number of words in the sentence}} \cdot \log{\frac{\text{number of sentences}}{\text{number of sentences that contain the word}}}$$
Additonally we are going to use 75% of the data for training, and 25% for testing, and we are going to use 5-fold cross validation to evaluate the model.

In [ ]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(all_tweets['Filtered_Tweet'].values, all_tweets['Label'].values,

# Train on the training data and transform the training and testing data
feature_extraction = TfidfVectorizer(ngram_range=(1, 2), max_features=10000)
X_train_features = feature_extraction.fit_transform(X_train)
X_test_features = feature_extraction.transform(X_test)
joblib.dump(feature_extraction, 'models/features.joblib')
```

Out[ ]:
```
['models/features.joblib']
```

Model Training

We will use three models to train the data, a Naive Bayes model, a Search Tree model, and a not-assigned model, we will train the data in 5-fold cross validation, on a 75% training set and a 25% test set.

Naive Bayes Classifier

Naive Bayes is a probabilistic model that uses Bayes' theorem to predict the class of a given data point. It is a simple model that is easy to implement and is very fast. It is also very effective in text classification problems. The model is based on the assumption that the features are independent of each other, which is not true in most cases, but it still works well in practice.

75 - 25 Split

In [ ]:
```
# Train a Naive Bayes classifier using 5-fold cross-validation
NaiveBayes_Classifer = MultinomialNB()
NaiveBayes_Classifer.fit(X_train_features, y_train)

# Evaluate the classifier using 5-fold cross-validation

# Evaluate the classifier on the test data
accuracy = NaiveBayes_Classifer.score(X_test_features, y_test)
y_pred = NaiveBayes_Classifer.predict(X_test_features)
print(f"Accuracy: {accuracy:.2f}")
print(classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
print("True Positives (TP) = ", cm[0][0])
print("False Positives (FP) = ", cm[0][1])
print("False Negatives (FN) = ", cm[1][0])
print("True Negatives (TN) = ", cm[1][1])
joblib.dump(NaiveBayes_Classifer, 'models/NaiveBayes.joblib')
```

```
Accuracy: 0.86
              precision    recall  f1-score   support

         neg       0.83      0.90      0.86      5555
         pos       0.90      0.82      0.86      5764

    accuracy                           0.86     11319
   macro avg       0.86      0.86      0.86     11319
weighted avg       0.86      0.86      0.86     11319

Confusion Matrix:
[[5001  554]
 [1010 4754]]
True Positives (TP) =  5001
False Positives (FP) =  554
False Negatives (FN) =  1010
True Negatives (TN) =  4754
```

```
Out[ ]: ['models/NaiveBayes.joblib']
```

5 Fold Cross Validation

```
In [ ]: # Define the Random Forest model
naive_bayes_5f = MultinomialNB()

# Define the cross-validation splitter
kf = KFold(n_splits=5, shuffle=True, random_state=42)
# Define the evaluation metrics
metrics = {'accuracy': make_scorer(accuracy_score),
           'precision': make_scorer(precision_score, pos_label='pos'),
           'recall': make_scorer(recall_score, pos_label='pos'),
           'f1': make_scorer(f1_score, pos_label='pos')}

# Fit and evaluate the model with 5-fold cross-validation
X = all_tweets['Filtered_Tweet'].values
y = all_tweets['Label'].values
y_preds = cross_val_predict(naive_bayes_5f, feature_extraction.fit_transform(X), y, cv=kf)
print('Classification Report:')
print(classification_report(y, y_preds))
print('Confusion Matrix:')
print(confusion_matrix(y, y_preds))
```

```
# Fit the model on the full dataset
naive_bayes_5f.fit(feature_extraction.fit_transform(X), y)

# Save the model using joblib
joblib.dump(naive_bayes_5f, 'models/naive_bayes_5f.joblib')
```

```
Classification Report:
              precision    recall  f1-score   support

         neg       0.84      0.90      0.87     22514
         pos       0.90      0.83      0.86     22761

    accuracy                           0.87     45275
   macro avg       0.87      0.87      0.87     45275
weighted avg       0.87      0.87      0.87     45275

Confusion Matrix:
[[20308  2206]
 [ 3835 18926]]
```

```
Out[ ]: ['models/naive_bayes_5f.joblib']
```

Decision Tree Classifier

We use Random Forest Classifier to train the data, Random Forest is an ensemble learning method for classification, regression, and other tasks, that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

75 - 25 Split

```
In [ ]: # Train a decision tree classifier on the features
RandomForest_Classifier = RandomForestClassifier(n_estimators=100)
RandomForest_Classifier.fit(X_train_features, y_train)

# Evaluate the classifier on the test data
accuracy = RandomForest_Classifier.score(X_test_features, y_test)
y_pred = RandomForest_Classifier.predict(X_test_features)
```

```
print(f"Accuracy: {accuracy:.2f}")

print(classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
joblib.dump(RandomForest_Classifier, 'models/RandomFortress.joblib')
```

```
Accuracy: 0.93
              precision    recall  f1-score   support

         neg       0.93      0.93      0.93      5555
         pos       0.93      0.93      0.93      5764

    accuracy                           0.93     11319
   macro avg       0.93      0.93      0.93     11319
weighted avg       0.93      0.93      0.93     11319

Confusion Matrix:
[[5162  393]
 [ 402 5362]]
```

```
Out[ ]: ['models/RandomFortress.joblib']
```

5 Fold Cross Validation

```
In [ ]: # Define the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Define the cross-validation splitter
kf = KFold(n_splits=5, shuffle=True, random_state=42)
# Define the evaluation metrics
metrics = {'accuracy': make_scorer(accuracy_score),
           'precision': make_scorer(precision_score, pos_label='pos'),
           'recall': make_scorer(recall_score, pos_label='pos'),
           'f1': make_scorer(f1_score, pos_label='pos')}

# Fit and evaluate the model with 5-fold cross-validation
X = all_tweets['Filtered_Tweet'].values
y = all_tweets['Label'].values
y_preds = cross_val_predict(rf_model, feature_extraction.fit_transform(X), y, cv=kf)
print('Classification Report:')
print(classification_report(y, y_preds))
print('Confusion Matrix:')
print(confusion_matrix(y, y_preds))

# Fit the model on the full dataset
rf_model.fit(feature_extraction.fit_transform(X), y)

# Save the model using joblib
joblib.dump(rf_model, 'models/RandomFortress_5f.joblib')
```

```
Classification Report:
              precision    recall  f1-score   support

         neg       0.94      0.92      0.93     22514
         pos       0.93      0.94      0.93     22761

    accuracy                           0.93     45275
   macro avg       0.93      0.93      0.93     45275
weighted avg       0.93      0.93      0.93     45275

Confusion Matrix:
[[20800  1714]
 [ 1346 21415]]
```

Out[ ]:
```
['models/RandomFortress_5f.joblib']
```

SVMs (Support Vector Machines)

In [ ]:
```
# Create a pipeline for the SVM classifier
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(X_train_features, y_train)

# Evaluate the classifier on the test data
accuracy = svm_classifier.score(X_test_features, y_test)
print(f"Accuracy: {accuracy:.2f}")
print(classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
joblib.dump(svm_classifier, 'models/SVM.joblib')
```

```
Accuracy: 0.92
              precision    recall  f1-score   support

         neg       0.93      0.91      0.92      5555
         pos       0.92      0.93      0.92      5764

    accuracy                           0.92     11319
   macro avg       0.92      0.92      0.92     11319
weighted avg       0.92      0.92      0.92     11319

Confusion Matrix:
[[5061  494]
 [ 382 5382]]
```

Out[ ]:
```
['models/SVM.joblib']
```

5 Fold Cross Validation

In [ ]:
```
# Define the Random Forest model
svc_model_5f = SVC(kernel='linear')

# Define the cross-validation splitter
kf = KFold(n_splits=5, shuffle=True, random_state=42)
# Define the evaluation metrics
metrics = {'accuracy': make_scorer(accuracy_score),
           'precision': make_scorer(precision_score, pos_label='pos'),
           'recall': make_scorer(recall_score, pos_label='pos'),
           'f1': make_scorer(f1_score, pos_label='pos')}

# Fit and evaluate the model with 5-fold cross-validation
X = all_tweets['Filtered_Tweet'].values
y = all_tweets['Label'].values
y_preds = cross_val_predict(svc_model_5f, feature_extraction.fit_transform(X), y, cv=kf)
print('Classification Report:')
print(classification_report(y, y_preds))
print('Confusion Matrix:')
print(confusion_matrix(y, y_preds))

# Fit the model on the full dataset
svc_model_5f.fit(feature_extraction.fit_transform(X), y)
```

```python
# Save the model using joblib
joblib.dump(svc_model_5f, 'models/svc_model_5f.joblib')
```

```
Classification Report:
              precision    recall  f1-score   support

         neg       0.93      0.91      0.92     22514
         pos       0.91      0.94      0.92     22761

    accuracy                           0.92     45275
   macro avg       0.92      0.92      0.92     45275
weighted avg       0.92      0.92      0.92     45275

Confusion Matrix:
[[20496  2018]
 [ 1440 21321]]
```

```
Out[ ]: ['models/svc_model_5f.joblib']
```