GOAL: To classify the given text messages into English, Hindi and Marathi languages.

In [ ]:

```
import pandas as pd

import matplotlib.pyplot as plt
from wordcloud import WordCloud

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import classification_report

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

In [ ]:

```
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/chat.csv")
```

In [ ]:

```
df.head()
```

Out[ ]:

	Texts	Lables
0	how are u	0
1	its good	0
2	nice to hear that	0
3	it was nice meeting u	0
4	how old are u	0

In [ ]:

```
X = df["Texts"]
y = df["Lables"]
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

In [ ]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)
```

Vectorization

Count vectorization

In [ ]:

```
cv = CountVectorizer(stop_words="english")
```

In [ ]:

```
X_train_cv = cv.fit_transform(X_train)
X_test_cv = cv.transform(X_test)
```

In [ ]:

```
dt = DecisionTreeClassifier()
```

In [ ]:

```
dt.fit(X_train_cv, y_train)
```

Out[ ]:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
In [ ]: y_pred = dt.predict(X_test_cv)
```

```
In [ ]: print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.60      1.00      0.75        31
           1       0.83      0.54      0.66        35
           2       0.81      0.57      0.67        30

    accuracy                           0.70        96
   macro avg       0.74      0.70      0.69        96
weighted avg       0.75      0.70      0.69        96
```

TF-IDF

```
In [ ]: tfidf = TfidfVectorizer(stop_words="english")
```

```
In [ ]: X_train_tf = tfidf.fit_transform(X_train)
```

```
In [83]: X_test_tf = tfidf.transform(X_test)
```

```
In [84]: dt = DecisionTreeClassifier()
```

```
In [85]: dt.fit(X_train_tf,y_train)
```

```
Out[85]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
In [87]: y_pred = dt.predict(X_test_tf)
```

```
In [88]: y_pred
```

```
Out[88]: array([0, 0, 0, 2, 1, 1, 0, 0, 1, 1, 0, 2, 0, 2, 0, 1, 0, 0, 0, 0, 0, 0,
       2, 0, 0, 2, 0, 0, 0, 1, 0, 1, 0, 2, 1, 2, 0, 0, 0, 2, 0, 1, 1, 2,
       0, 2, 2, 0, 1, 1, 0, 0, 2, 1, 2, 0, 2, 0, 2, 0, 0, 1, 0, 0, 2, 0,
       0, 0, 1, 0, 2, 1, 0, 0, 1, 1, 0, 2, 2, 0, 0, 0, 0, 2, 2, 0, 2, 1,
       2, 0, 0, 2, 0, 2, 0, 1])
```

```
In [89]: print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.61      1.00      0.76        31
           1       0.95      0.54      0.69        35
           2       0.80      0.67      0.73        30

    accuracy                           0.73        96
   macro avg       0.79      0.74      0.72        96
weighted avg       0.79      0.73      0.72        96
```

Support Vector Machine:

using count vectorization:

```
In [ ]: from sklearn.svm import LinearSVC
```

```
In [ ]: lsv = LinearSVC(random_state=1)
```

```
In [ ]: lsv.fit(X_train_cv, y_train)
```

```
Out[ ]: LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
                  intercept_scaling=1, loss='squared_hinge', max_iter=1000,
                  multi_class='ovr', penalty='l2', random_state=1, tol=0.0001,
                  verbose=0)
```

```
In [ ]: y_pred = lsv.predict(X_test_cv)
```

```
In [ ]: y_pred
```

```
Out[ ]: array([0, 0, 0, 2, 1, 1, 0, 0, 1, 1, 0, 2, 0, 2, 0, 1, 0, 0, 0, 0, 1, 1,
               2, 0, 0, 2, 0, 2, 0, 1, 0, 1, 0, 2, 1, 2, 0, 0, 0, 2, 0, 1, 1, 2,
               1, 2, 1, 0, 1, 1, 0, 0, 2, 1, 2, 0, 2, 0, 2, 0, 0, 1, 0, 0, 2, 0,
               0, 0, 1, 0, 2, 1, 0, 0, 1, 1, 0, 2, 2, 0, 0, 0, 0, 2, 2, 0, 2, 1,
               2, 0, 0, 2, 0, 2, 0, 1])
```

```
In [ ]: print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.66      1.00      0.79        31
           1       0.92      0.63      0.75        35
           2       0.84      0.70      0.76        30

    accuracy                           0.77        96
   macro avg       0.81      0.78      0.77        96
weighted avg       0.81      0.77      0.77        96
```

choosing a soft margin

```
In [ ]: lsvs = LinearSVC(random_state=1, C = 0.5)
```

```
In [ ]: lsvs.fit(X_train_cv, y_train)
```

```
Out[ ]: LinearSVC(C=0.5, class_weight=None, dual=True, fit_intercept=True,
                  intercept_scaling=1, loss='squared_hinge', max_iter=1000,
                  multi_class='ovr', penalty='l2', random_state=1, tol=0.0001,
                  verbose=0)
```

```
In [ ]: y_pred_1 = lsvs.predict(X_test_cv)
```

```
In [ ]: print(classification_report(y_test,y_pred_1))
```

```
              precision    recall  f1-score   support

           0       0.66      1.00      0.79        31
           1       0.92      0.63      0.75        35
           2       0.84      0.70      0.76        30

    accuracy                           0.77        96
   macro avg       0.81      0.78      0.77        96
weighted avg       0.81      0.77      0.77        96
```

using TF-IDF vectorization

```
In [90]: lsv_1 = LinearSVC(random_state=1)
In [91]: lsv_1.fit(X_train_tf, y_train)
```

```
Out[91]: LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
                   intercept_scaling=1, loss='squared_hinge', max_iter=1000,
                   multi_class='ovr', penalty='l2', random_state=1, tol=0.0001,
                   verbose=0)
```

```
In [93]: y_pred = lsv_1.predict(X_test_tf)
```

```
In [94]: print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        31
           1       0.91      0.86      0.88        35
           2       0.86      0.80      0.83        30

    accuracy                           0.89        96
   macro avg       0.88      0.89      0.88        96
weighted avg       0.89      0.89      0.88        96
```

choosing a soft margin

```
In [95]: lsvs = LinearSVC(random_state=1, C = 0.5)
```

```
In [96]: lsvs.fit(X_train_tf, y_train)
```

```
Out[96]: LinearSVC(C=0.5, class_weight=None, dual=True, fit_intercept=True,
                   intercept_scaling=1, loss='squared_hinge', max_iter=1000,
                   multi_class='ovr', penalty='l2', random_state=1, tol=0.0001,
                   verbose=0)
```

```
In [97]: y_pred = lsvs.predict(X_test_tf)
```

```
In [99]: print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        31
           1       0.91      0.86      0.88        35
           2       0.86      0.83      0.85        30

    accuracy                           0.90        96
   macro avg       0.89      0.90      0.89        96
weighted avg       0.90      0.90      0.89        96
```

WE observe that we are getting an accuracy of 90 % with TD-IDF vectorization using Support Vector Machine . Count vectorization and decision tree algorithm are not able to give that good accuracy.