WEEK-4

Classification of a set of Text Documents into known classes (You may use any of the Classification algorithms like Naive Bayes, Max Entropy, Rochio"s, Support Vector Machine). Standard Datasets will have to be used to show the results.

Introduction to Text Classification

Text classification is a fundamental task in Natural Language Processing (NLP) where a piece of text (e.g., a document, sentence, or paragraph) is assigned to one or more predefined categories. Applications include:

- Spam detection
- Sentiment analysis
- Topic labelling
- News categorization

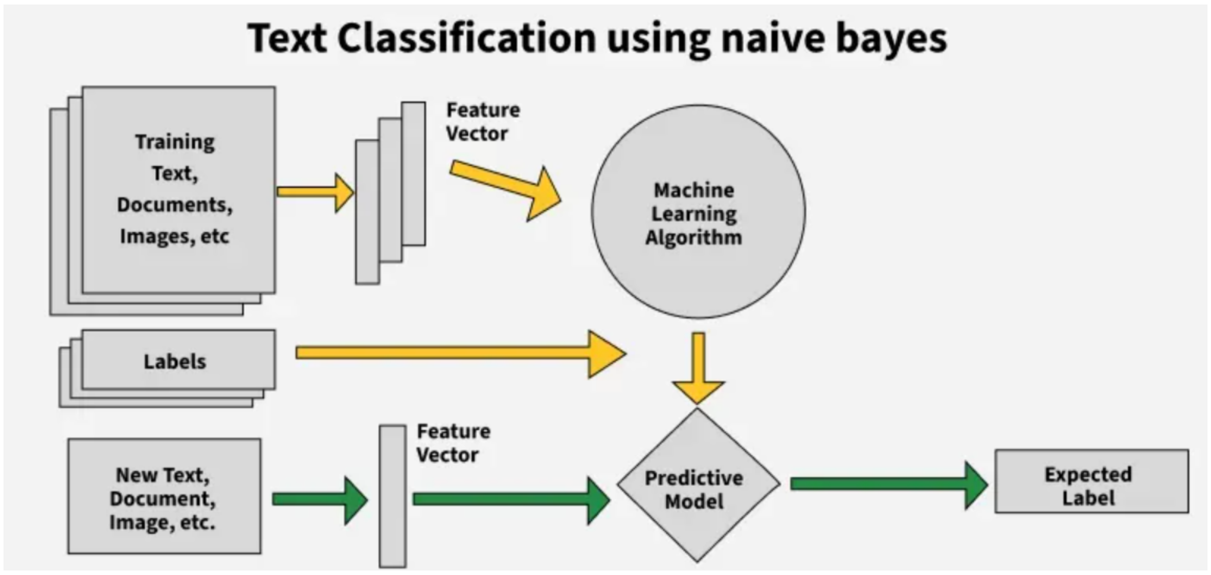
Naive Bayes

Naive Bayes is a classification algorithm that uses probability to predict which category a data point belongs to, assuming that all features are unrelated. It is named as "Naive" because it assumes the presence of one feature does not affect other features.

Why Naive Bayes?

The **Naive Bayes classifier** is a popular probabilistic machine learning algorithm based on **Bayes' Theorem** with an assumption of independence among features. It is particularly well-suited for text classification due to:

- **Efficiency** in handling high-dimensional data (like text)
- **Robustness** even with small training datasets
- **Scalability** and speed of training and prediction


Text Classification using naive bayes
Training Text, Documents, Images, etc
Feature Vector
Machine Learning Algorithm
Labels
New Text, Document, Image, etc.
Feature Vector
Predictive Model
Expected Label

CODE:

```
from sklearn.datasets import fetch_20newsgroups

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, classification_report

categories = ['sci.space', 'rec.sport.hockey', 'comp.graphics', 'alt.atheism']

newsgroups = fetch_20newsgroups(subset='all', categories=categories, shuffle=True,
random_state=42)

print(f"Total documents: {len(newsgroups.data)}")

print(f"Target classes: {newsgroups.target_names}")

X_train, X_test, y_train, y_test = train_test_split (

                          newsgroups.data, newsgroups.target, test_size=0.2, random_state=42 )

vectorizer = TfidfVectorizer(stop_words='english')

X_train_tfidf = vectorizer.fit_transform(X_train)

X_test_tfidf = vectorizer.transform(X_test)

nb = MultinomialNB()

nb.fit(X_train_tfidf, y_train)

y_pred = nb.predict(X_test_tfidf)

print("Accuracy:", accuracy_score(y_test, y_pred))

print("\nClassification Report:\n")

print(classification_report(y_test, y_pred, target_names=newsgroups.target_names))

for i in range(5):

   print("\nText:\n", X_test[i])

   print("Actual:", newsgroups.target_names[y_test[i]])

   print("Predicted:", newsgroups.target_names[y_pred[i]])
```