1.**start by importing libraries**

```
import pandas as pd

import numpy as np

import re

import nltk

from nltk.corpus import stopwords

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.svm import SVC

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, precision_score, recall_score
```

2.**load and preprocess data**

```
# Load dataset

Df = pd.read_csv('path/to/dataset.csv')


# Preprocess text data

Def clean_text(text):

   Text = text.lower()

   Text = re.sub(r'\d+', '', text) # remove numbers

   Text = re.sub(r'[^\w\s]', '', text) # remove punctuation

   Text = re.sub(r'\s+', ' ', text) # remove extra spaces

   Text = ' '.join([word for word in text.split() if word not in stopwords.words('english')]) # remove stop
words

   Return text


Df['text'] = df['text'].apply(lambda x: clean_text(x))
```

3.**split the dataset into training and testing**

```
# Split dataset into training and testing sets
```

X_train, X_test, y_train, y_test = train_test_split(df[‘text’], df[‘sentiment’], test_size=0.2, random_state=42)

4.**Transfirm text data into numerical features**

```
# Transform text data into numerical features

Vectorizer = CountVectorizer()

X_train = vectorizer.fit_transform(X_train)

X_test = vectorizer.transform(X_test)
```

5.**We can now train and evaluate the performance of the Naïve Bayes, SVM, and Logistic Regression algorithms.**

```
# Train and evaluate Naive Bayes

nb = MultinomialNB()

nb.fit(X_train, y_train)

nb_pred = nb.predict(X_test)

print('Naive Bayes Accuracy:', accuracy_score(y_test, nb_pred))

print('Naive Bayes Precision:', precision_score(y_test, nb_pred, pos_label='positive'))

print('Naive Bayes Recall:', recall_score(y_test, nb_pred, pos_label='positive'))


# Train and evaluate SVM

svm = SVC(kernel='linear', probability=True)

svm.fit(X_train, y_train)

svm_pred = svm.predict(X_test)

print('SVM Accuracy:', accuracy_score(y_test, svm_pred))

print('SVM Precision:', precision_score(y_test, svm_pred, pos_label='positive'))

print('SVM Recall:', recall_score(y_test, svm_pred, pos_label='positive'))


# Train and evaluate Logistic Regression

lr = LogisticRegression(max_iter=1000)

lr.fit(X_train, y_train)

lr_pred = lr.predict(X_test)
```

```
print('Logistic Regression Accuracy:', accuracy_score(y_test, lr_pred))
print('Logistic Regression Precision:', precision_score(y_test, lr_pred, pos_label='positive'))
print('Logistic Regression Recall:', recall_score(y_test, lr_pred, pos_label='positive'))
```