```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Jul 15 20:05:48 2023

@author: myyntiimac
"""

### Customer sentiment analysis using NLP and ML
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv("/Users/myyntiimac/Desktop/Restaurant_Reviews.tsv",delimi

# Cleaning the texts
import re
import nltk
#nltk.download('stopwords')

from nltk.stem.porter import PorterStemmer

corpus = []
for i in range(0, 1000):
    review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])
    review = review.lower()
    review = review.split()
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review]
    review = ' '.join(review)
    corpus.append(review)

# Creating the TFIDF VEctorizer
from sklearn.feature_extraction.text import TfidfVectorizer
cv = TfidfVectorizer()

X = cv.fit_transform(corpus).toarray()
y = dataset.iloc[:, 1].values

#Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, ran

from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score, roc_curve
```

```
# Initialize a dictionary to store the accuracy scores
accuracy_scores = {}
# Define the classifier models
models = {
'Random Forest': RandomForestClassifier(),
'Logistic Regression': LogisticRegression(),
'Support Vector Machine': SVC(),
'K-Nearest Neighbors': KNeighborsClassifier(),
'Decision Tree': DecisionTreeClassifier(),
'Naive Bayes': GaussianNB()
}
# Iterate over each model
for model_name, model in models.items():
# Fit the model on the training data
   model.fit(X_train, y_train)
# Make predictions on the test data
   y_pred = model.predict(X_test)
# Calculate the accuracy score
   accuracy = accuracy_score(y_test, y_pred)
# Store the accuracy score in the dictionary
   accuracy_scores[model_name] = accuracy
# Sort the accuracy scores in descending order
sorted_scores = sorted(accuracy_scores.items(), key=lambda x: x[1], reverse=Tru
# Print the ranking and accuracy scores of the models
print("Model Rankings based on Accuracy Score:")
for rank, (model_name, accuracy) in enumerate(sorted_scores, start=1):
    print(f"Rank {rank}: {model_name} - Accuracy: {accuracy:.4f}")
# Print all the accuracy scores
print("\nAll Accuracy Scores:")
for model_name, accuracy in accuracy_scores.items():
    print(f"{model_name}: {accuracy:.4f}")

#in model rankig outpout we found rank 1 model is support vector machine  we wi
#Lets check the bias and variance of SVM

SVM=SVC()
SVM.fit(X_train, y_train)
# Predicting the Test set results
y_pred = SVM.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)

from sklearn.metrics import accuracy_score
ac = accuracy_score(y_test, y_pred)
print(ac)
bias = SVM.score(X_train,y_train)
bias

variance = SVM.score(X_test,y_test)
variance
```