Data Science with Python Major Project

```
import re
import string
import spacy
import nltk
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
from spacy import displacy
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report,accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
import warnings
warnings.filterwarnings('ignore')


class PredictReview:

    def vectorize(self,train_data,test_data):

        tfidf = TfidfVectorizer()
        train = tfidf.fit_transform(train_data.values.astype('U'))
        test = tfidf.transform(test_data.values.astype('U'))
        return train,test,tfidf

    def split(self,data,train_size,shuffle):

        input_data = data['reviews']
        output_data = data['sentiment']
        train_data, test_data, train_output, test_output =
train_test_split(input_data, output_data, test_size=train_size,
random_state=shuffle)
        return train_data, test_data, train_output, test_output

    def base_logisticRegression(self,data):

        log_reg = LogisticRegression()
        data = self.prepare_data_for_train(data)
        train_data, test_data, train_output, test_output =
self.split(data,0.5,101)
        train,test,tfidf = self.vectorize(train_data,test_data)
        log_reg.fit(train,train_output)
        pred = log_reg.predict(test)
        self.performance(pred,test,test_output,log_reg)
        return log_reg,tfidf

    def base_MultinomialNB(self,data):

        Mult_NB = MultinomialNB()
        data = self.prepare_data_for_train(data)
        train_data, test_data, train_output, test_output =
self.split(data,0.25,400)
        train,test,tfidf = self.vectorize(train_data,test_data)
        Mult_NB.fit(train,train_output)
        pred = Mult_NB.predict(test)
        self.performance(pred,test,test_output,Mult_NB)
        return Mult_NB,tfidf

    def base_KNN(self,data):

        KNN =  KNeighborsClassifier(n_neighbors=3)
        data = self.prepare_data_for_train(data)
        train_data, test_data, train_output, test_output =
self.split(data,0.75,19)
        train,test,tfidf = self.vectorize(train_data,test_data)
        KNN.fit(train,train_output)
        pred = KNN.predict(test)
        self.performance(pred,test,test_output,KNN)
        return KNN,tfidf

    def prepare_data_for_train(self,input_data):

        stopword = nltk.corpus.stopwords.words('english')
        empty_list  = []
        for text in input_data.reviews:
            text = text.lower()
            text = re.sub('\[.*?\]', '', text)
            text = re.sub('https?://\S+|www\.\S+', '', text)
            text = re.sub('<.*?>+', '', text)
            text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
            text = re.sub('\n', '', text)
            text = re.sub('\w*\d\w*', '', text)
            text = re.sub(r'[^\w\s]', '',str(text))
            text=re.split("\W+",text)
            text=[word for word in text if word not in stopword]
            text = ' '.join(text)
            empty_list.append(text)
        input_data['review'] = empty_list
        return input_data


    def performance(self,_data,test_data,test_output,model):
        print(classification_report(_data,test_output))
        print('Overall accuracy is {}%
\n'.format(round(accuracy_score(_data,test_output)*100),0))


    def test_sample(self,text,tfidf,base_model):

        text = self.clean_df(text)
        text_sample = tfidf.transform([text])
        pred = base_model.predict(text_sample)
        if pred[0] == 1:
            return 'positive'
        else:
            return 'negative'

    def  clean_df(self,text):
        text = text.lower()
        stopword = nltk.corpus.stopwords.words('english')
        text = re.sub(r'[^\w\s]', '',str(text))
        text=re.split("\W+",text)
        text=[word for word in text if word not in stopword]
        text = ' '.join(text)
        return text

review_predictor = PredictReview()
data = pd.read_csv("D:\\DS Major Project\\amazon_alexa.csv")
print("LOGISTIC REGESSION")
model,coverter = review_predictor.base_logisticRegression(data)
print("NAIVE BAYES")
model,coverter = review_predictor.base_MultinomialNB(data)
print("KNN")
model,coverter = review_predictor.base_KNN(data)
```