Name: Tasneem Hoque
CSID: hoque
Banner #: B00841761

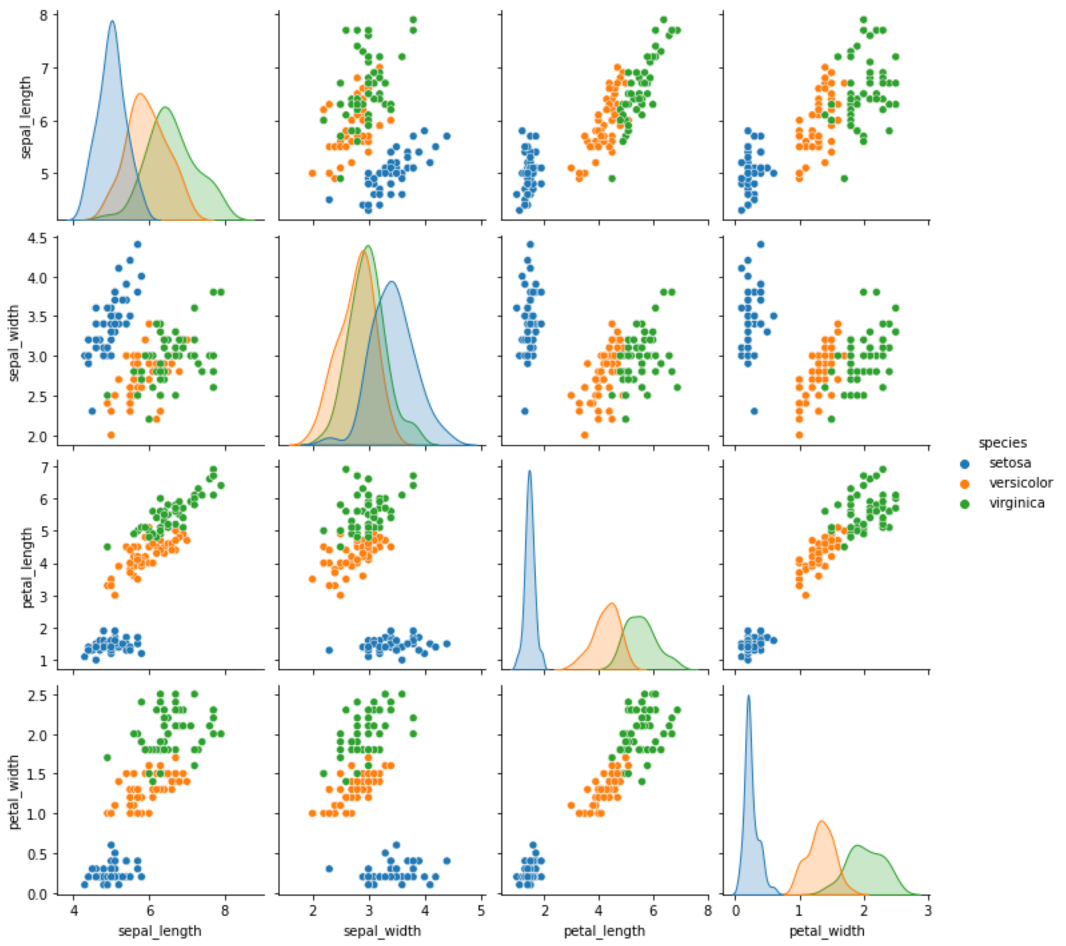
Jupyter Notebook: http://localhost:8888/notebooks/Tasneem_Hoque_Assignment2.ipynb

Part A: Iris Dataset

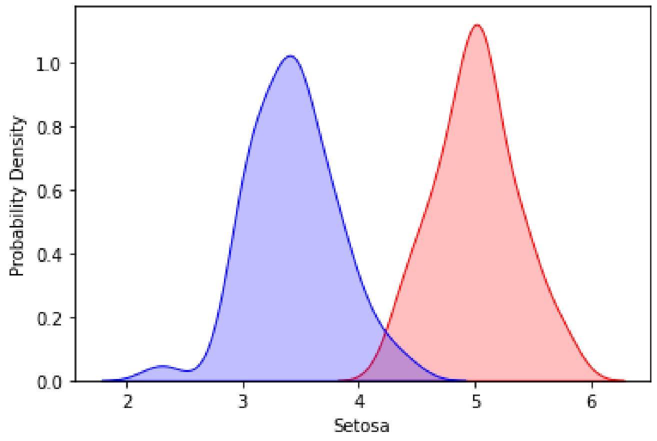
The first step is to import the libraries. For this implantation, I will be using the sklearn, seaborn and matplotlib libraries. I have also imported the Images, requests and BytesIO libraries to display the images of the three different flowers from the dataset.

The dataset I will be working with is the iris flower dataset from sklearn. There are three different categories of flower – Setosa, Virginica and Versicolor. The goal is to be able to train a model that will be able to categorize a flower into a single species depending on their four attributes – sepal length, sepal width, petal length and petal width.

I used the seaborn library to load the iris data set then printed the dataset for examination. I created a Pairplot to find the most distinguishable species of the three. As a result, I found that setosa is the most distinguishable species as seen in the Pairplot below.



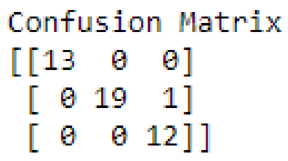
I also made a KDE plot for sepal length and sepal width for the setosa species of flower. From the KDE plot below, we can see that the sepal length and width both have a similar distribution curve, with sepal length having higher values.



I then imported the libraries required to train and test my model. The libraries I had to import were pandas, load_iris, train_test_split, SVC and numpy. I loaded the iris data set, and split it into X and y, where X is the sepal length, sepal width, petal length and petal width. And y is the target values with the column name – species. For this training, I used 30% of the data as my test set.

I made a model that is of the type SVC. I use this support vector classification model to classify the plants into three different species depending on sepal and petal length and width. I did not make use of the parameters to tune the model just yet. I fit the model using my training X and y sets.

I then went on to evaluate my model by creating an array that has predictions made by the model of the X values of the test sets. I also imported the classification_report and confusion_matrix from the sklearn.metrics library. As I compared the results of the prediction and actual values using a classification report and a confusion matrix, the results seemed pretty accurate. Following are images of the confusion matrix and the classification report.



```
Confusion Matrix
[[13  0  0]
 [ 0 19  1]
 [ 0  0 12]]
```

As we can see from the results of the confusion matrix, two categories of flower were 100% accurately classified and only one category of the flower was 83% accurately classified. I will further tune the model to decrease the rate of misclassification.

As we can see from the classification report for species 0 and 1, our precision is 100%, for species 2, we have recall of 100%. Overall, the accuracy of the model is 98% which is a very good number for this kind of classification. Although I am satisfied by the result, it will be worthwhile to tune the model and evaluate it to ensure maximum accuracy has been achieved.

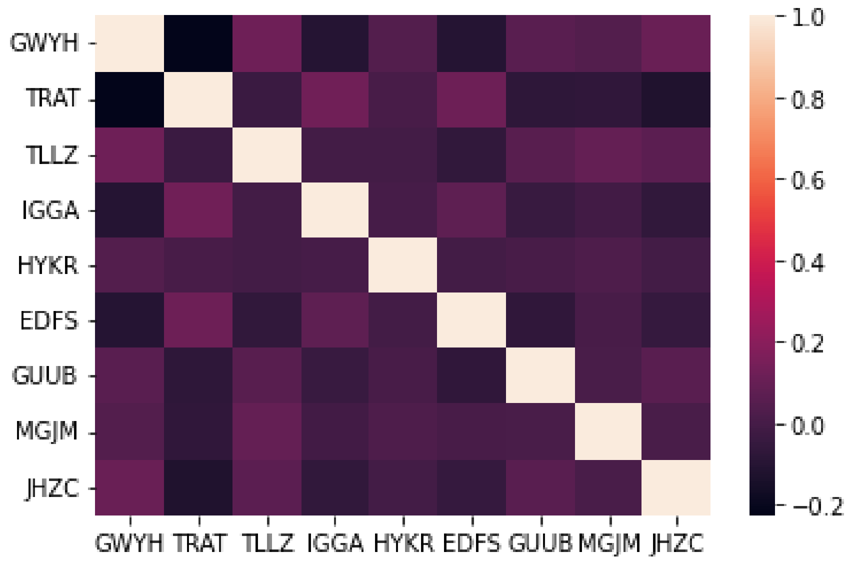
```
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      0.95      0.97        20
           2       0.92      1.00      0.96        12

    accuracy                           0.98        45
   macro avg       0.97      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

To tune the parameters of the model, I will be using the GridSearchCV. First, I evaluated which parameters will give the best results with the SVC model. I created a grid object and fit the data to the grid object. I then checked which parameters will give the best results. The parameters were set to C =1, gamma = 0.1 and kernel = 'rbf'. I then evaluated the model with the grid model that used its best parameters. But as seen from the results, it is very similar to the model I have initially created, therefore, the SVC model initially created is a very good model for predicting the flower species.

Part B: KNN Project Data

I start by importing libraries necessary to create the model, evaluate model, create graphs and import the dataset. I read the dataset then perform some initial analysis before working with the data. This data has 10 columns and a target class column consisting of 0 and 1, which means data can only be categorized into 2 different categories.

The data frame consists of 10 columns and 1000 rows of data. There are no values that are null and all the data in the data frame are numerical therefore, no data preprocessing is required. I then proceeded



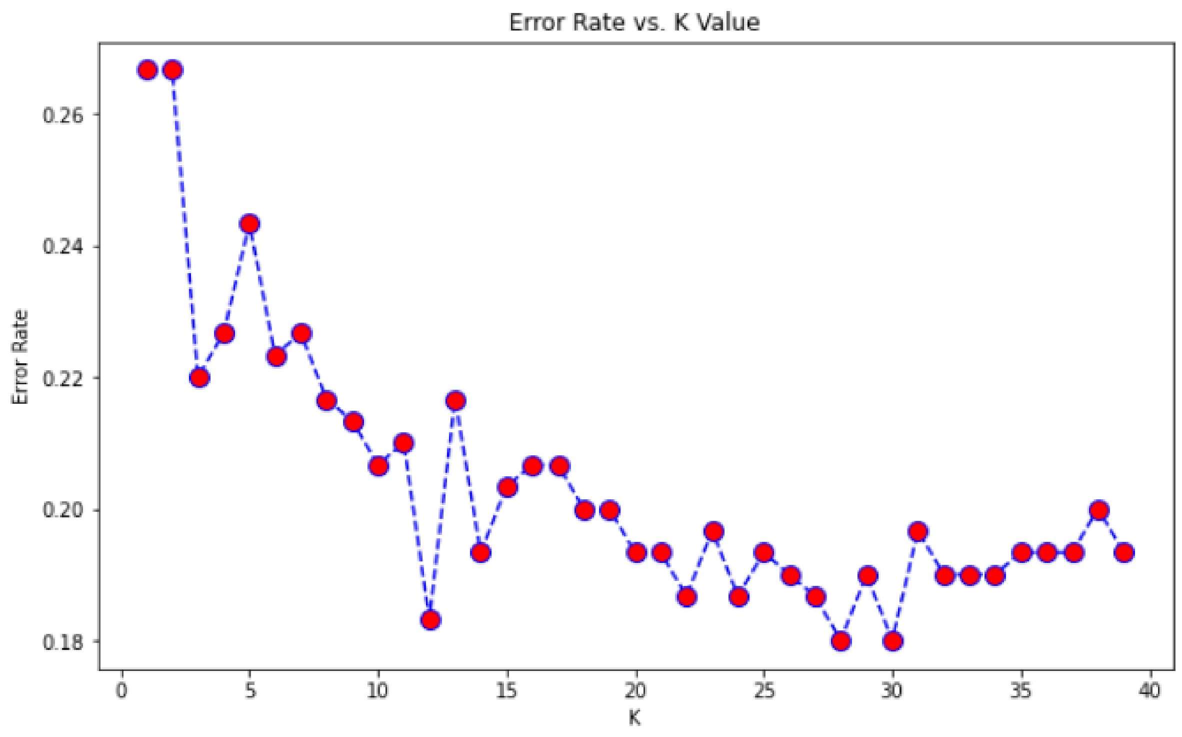
to find the mean and sum of each row, and they did not reveal a lot about the dataset. To do some more exploratory analysis of the dataset, I have tested out the correlation of the various columns. First, I made the correlation matrix and then created a heatmap using seaborn to visualize the correlation matrix. From the matrix we can see that columns TRAT and GWYH are the least closely related and so are columns GWYH and TLLZ and columns JHZC and TRAT. From the correlation matrix we got a clearer idea of the dataset and how each column may be related or unrelated to one another.

I then import the StandardScaler from the sklearn.preprocessing library to use as the model to predict the target class using the training data. I fit the data to the model and transform it while dropping the target class. I then split the data into train and test datasets. I use the k neighbors' classifier to classify the data, in this case I am using number of neighbors equal to 1. I then evaluate my model to get the following results:

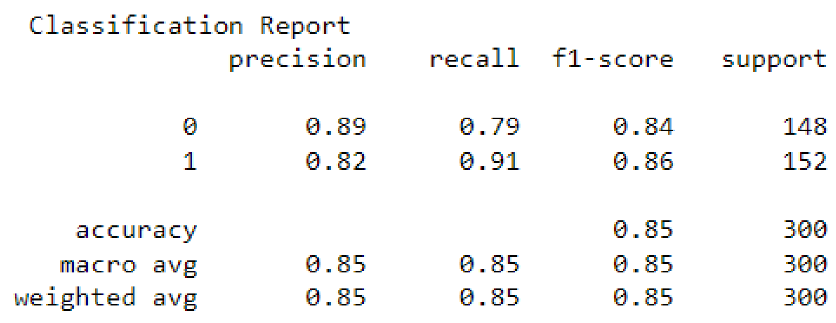
```
Classification Report
              precision    recall  f1-score   support

           0       0.78      0.70      0.74       159
           1       0.69      0.77      0.73       141

    accuracy                           0.73       300
   macro avg       0.74      0.74      0.73       300
weighted avg       0.74      0.73      0.73       300
```

```
Confusion Matrix
[[111  48]
 [ 32 109]]
```

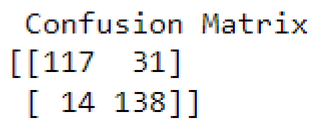
As seen from the results the accuracy is 73% and there is definitely room for improvement. To improve the accuracy of the data I will use the elbow method to pick a good k-value. I created a for loop that trains various KNN models with different k-values, then keep track of the error rate for each of these models with a list. I then create a plot using a for loop. Following is the results from the plot.



As seen from the plot, the k value that most minimizes error is 30, but a k-value of 12 will give us quite the same result. Therefore, we take the lower k-value because we don't want the model to overfit.



```
 Classification Report
              precision    recall  f1-score   support

           0       0.89      0.79      0.84       148
           1       0.82      0.91      0.86       152

    accuracy                           0.85       300
   macro avg       0.85      0.85      0.85       300
weighted avg       0.85      0.85      0.85       300
```



```
 Confusion Matrix
[[117  31]
 [ 14 138]]
```

As seen from the new classification report and confusion matrix, we can see that it has improved a lot. Now the model is giving us an 85% accuracy as opposed to the 73% accuracy we got from the model when k was equal to 1. Therefore, the model has been trained.