CMPSC 445: Applied Machine Learning in Data Science
Spring 2023

Project-1: Iris classification using KNN

Gabriel Nulman – CMPSC Senior
Professor V. Elangovan – Department Chair CMPSC

Goal: For the iris dataset, develop a KNN classifier to classify different iris species.

Brief information about the data:
This dataset consists of 4 attributes: sepal-length, sepal-width, petal-length and petal-width. Our goal is to predict the iris class given the 4 attributes. There are 3 classes in the dataset: Iris-setosa, Iris-versicolor and Iris-virginica. For more detailed information about the dataset, visit the above website.

1. **Introduction:**
   - KNN is a simple supervised machine learning algorithm used primarily for regression and classification. In this project, we will use it for regression to predict values based on k nearest neighbors. We will also test for errors to obtain the most optimal k value(s).
   - The Iris dataset is another dataset present within the Seaborn library. This 150-record dataset is very easy to navigate, in addition to have 3 specification classes which correspond to the three flower species.
2. **Design and Implementation:**
   - Below you can find the design and implementation for this project.

1. Import the dataset and print a sample of the iris dataset.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
cols = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
iris = pd.read_csv(url, header=None, names=cols)
```

```
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

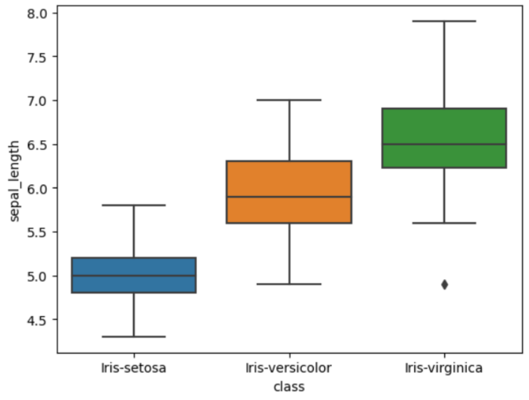
```
iris
```

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

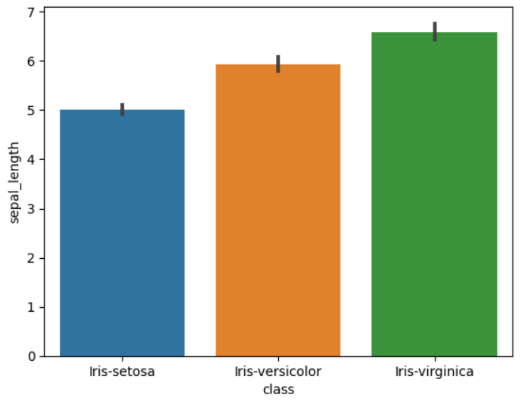
2. Perform various appropriate plots (box plot, bar plot, count plot, distplot, clustermap, pairplot) to visually analyze the data. Record your observations i.e., using your plots, what can you understand about this data?

```
sns.boxplot(x='class', y='sepal_length', data=iris)
plt.show()
```

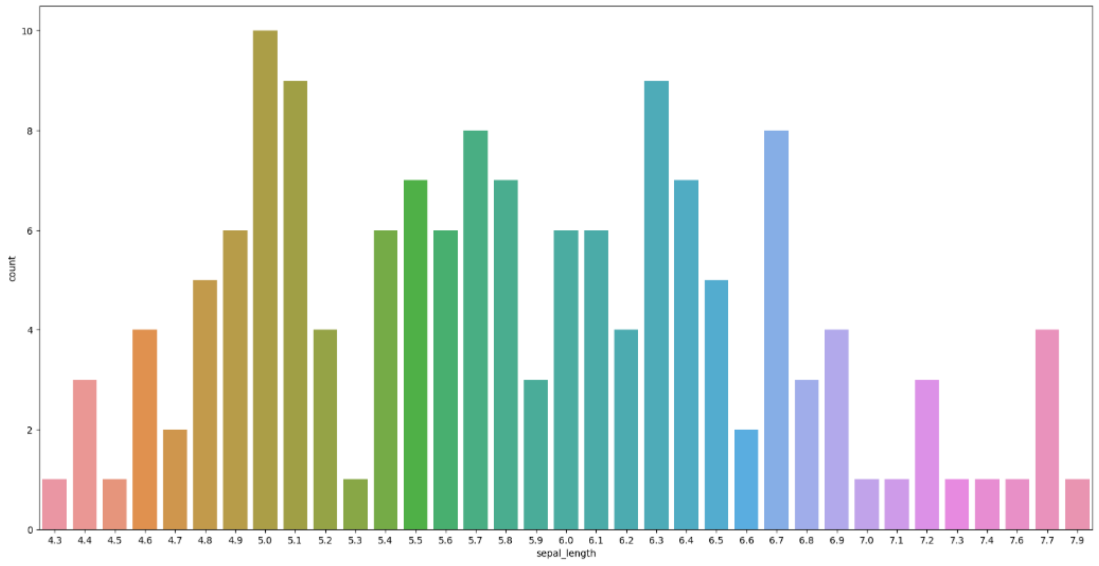


```
sns.barplot(x='class', y='sepal_length', data=iris)
```

```
<AxesSubplot:xlabel='class', ylabel='sepal_length'>
```



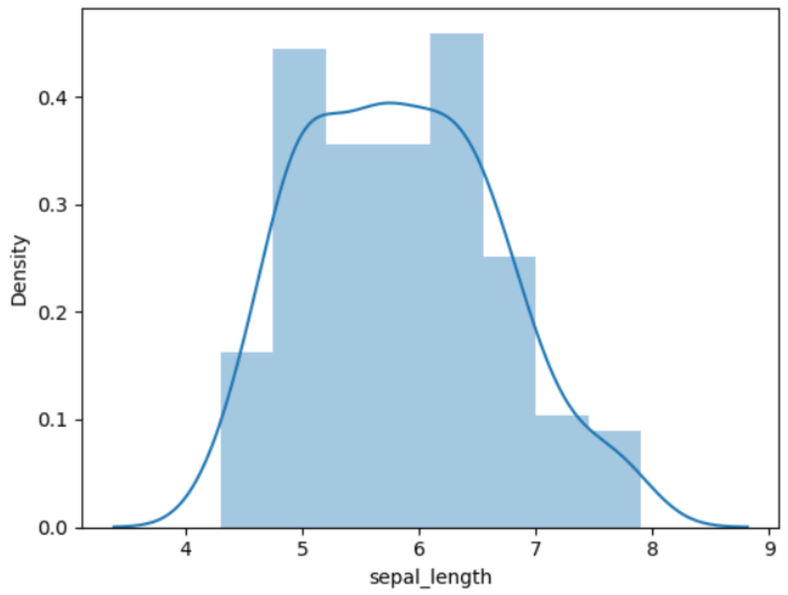
```
sns.countplot(x='sepal_length', data=iris)
plt.show()
plt.rcParams["figure.figsize"] = (30,10)
```



```
sns.distplot(iris['sepal_length'])
```

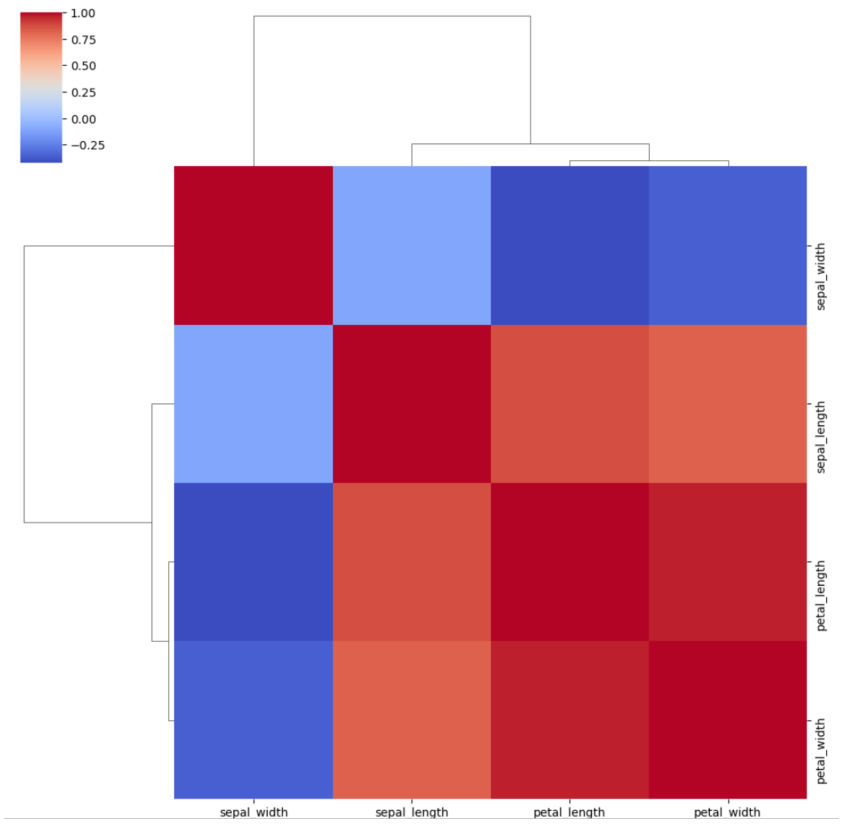
```
C:\Users\gaben\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWa
function and will be removed in a future version. Please adapt your code to use e
nction with similar flexibility) or `histplot` (an axes-level function for histogr
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='sepal_length', ylabel='Density'>
```



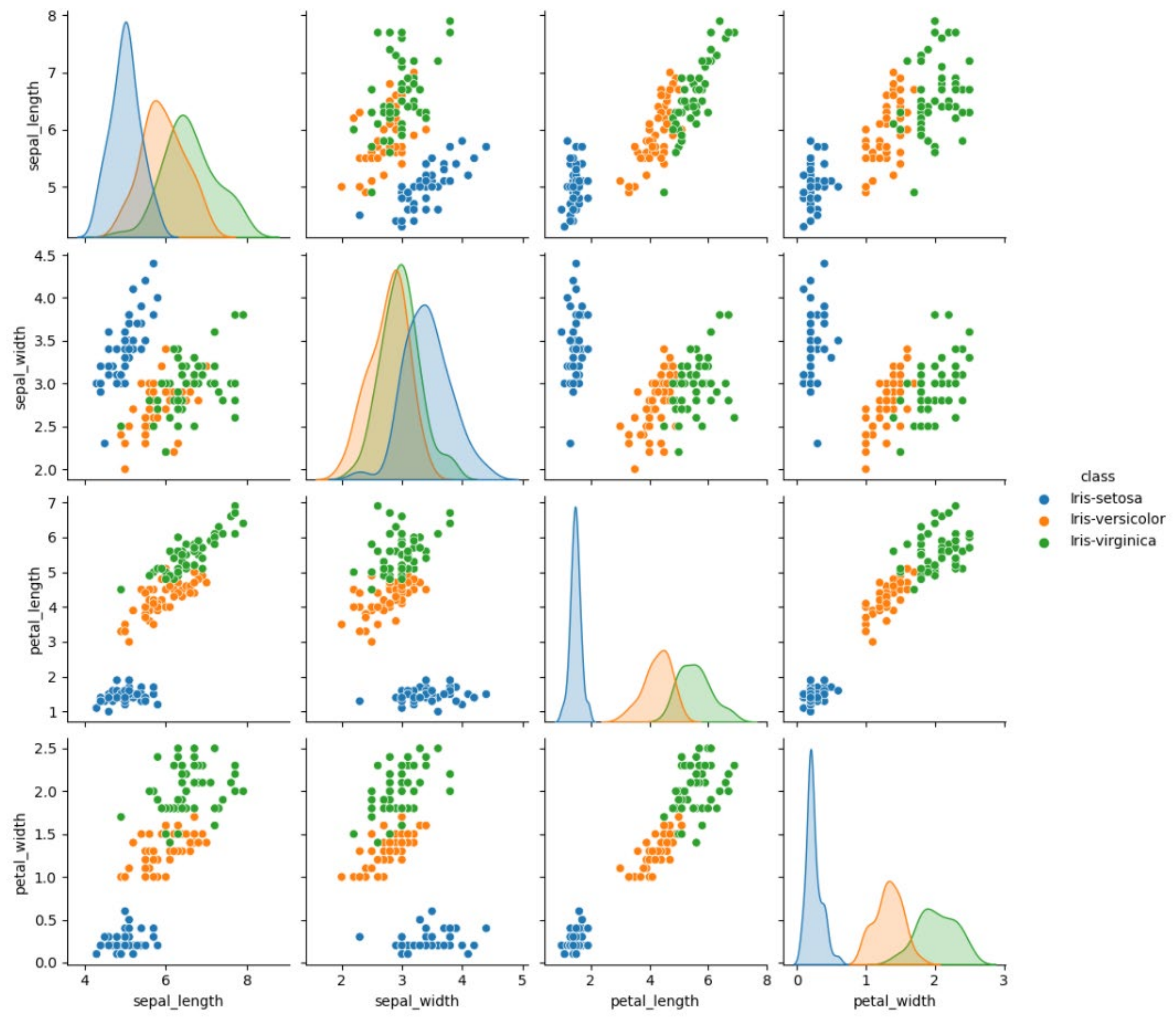
```
sns.clustermap(iris.corr(), cmap='coolwarm')
```

```
<seaborn.matrix.ClusterGrid at 0x1914a904910>
```



```
sns.pairplot(iris, hue='class')
```

```
<seaborn.axisgrid.PairGrid at 0x1914aa61220>
```



3. Preprocessing - split the data for training and testing i.e., 70(training):30(testing)

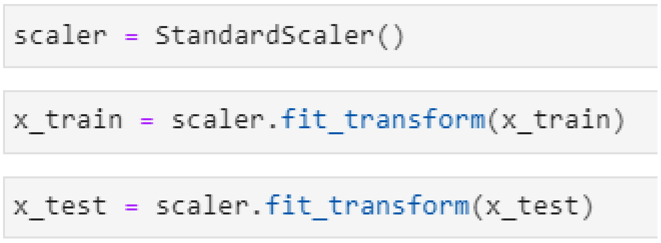
```
from sklearn.model_selection import train_test_split
```

```
x = iris.drop('class', axis=1)
```

```
y = iris['class']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

4. Scale the features i.e., perform normalization. From hereafter, you should be working on the scaled features.

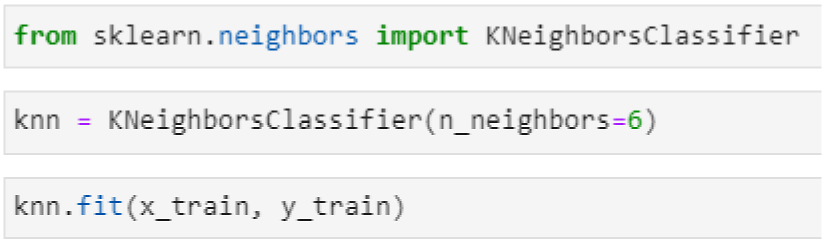


```
scaler = StandardScaler()
```

```
x_train = scaler.fit_transform(x_train)
```

```
x_test = scaler.fit_transform(x_test)
```

5. Perform the training using KNN algorithm.

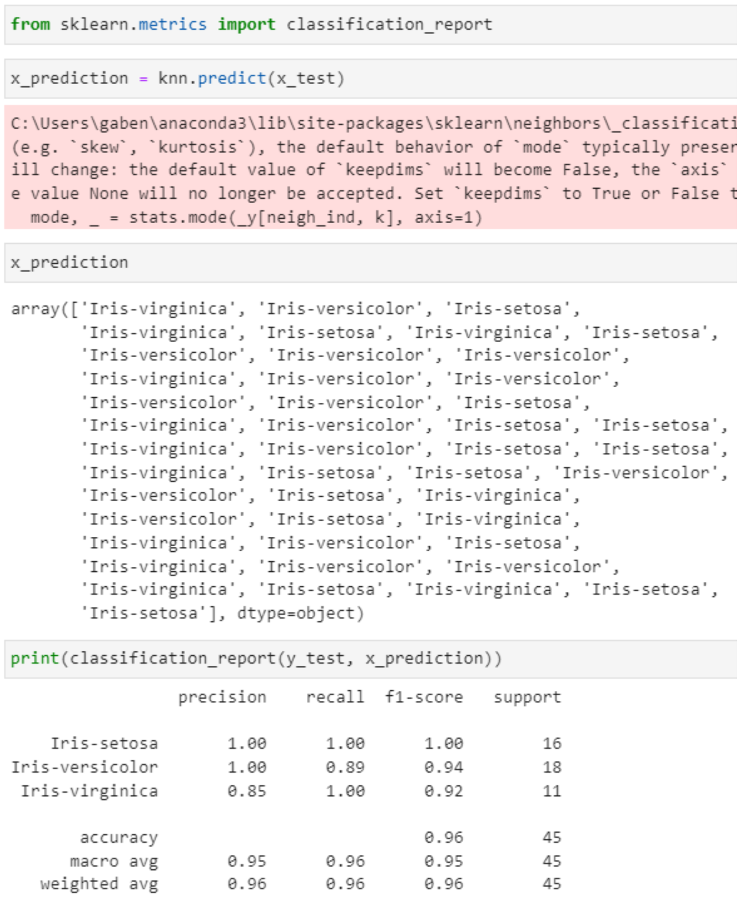


```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier(n_neighbors=6)
```

```
knn.fit(x_train, y_train)
```

6. Performance evaluation. Print the table and briefly explain your understanding.



```
from sklearn.metrics import classification_report
```

```
x_prediction = knn.predict(x_test)
```

```
C:\Users\gaben\anaconda3\lib\site-packages\sklearn\neighbors\_classificati
(e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preser
ill change: the default value of `keepdims` will become False, the `axis`
e value None will no longer be accepted. Set `keepdims` to True or False t
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

```
x_prediction
```

```
array(['Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-setosa'], dtype=object)
```

```
print(classification_report(y_test, x_prediction))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        16
Iris-versicolor       1.00      0.89      0.94        18
 Iris-virginica       0.85      1.00      0.92        11

       accuracy                           0.96        45
      macro avg       0.95      0.96      0.95        45
   weighted avg       0.96      0.96      0.96        45
```

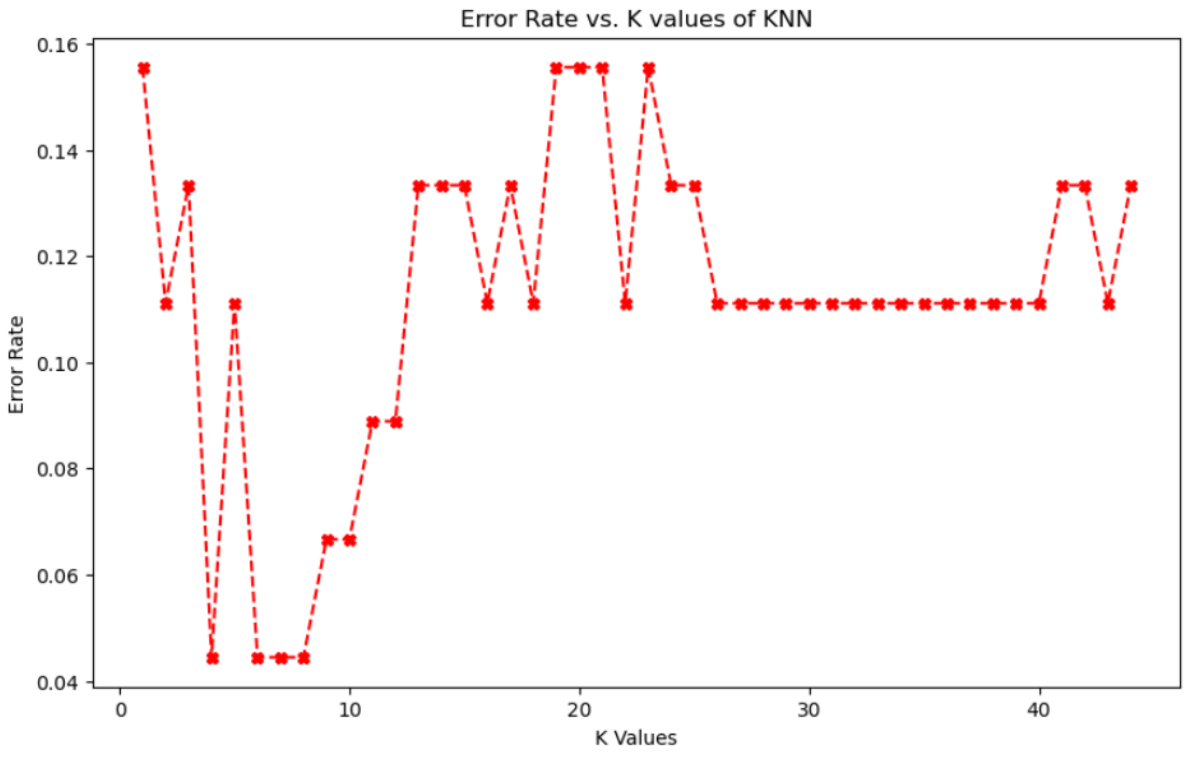
7. Choose appropriate k value by comparing the error rate. Show appropriate plot/s.

```
error_rate = []
```

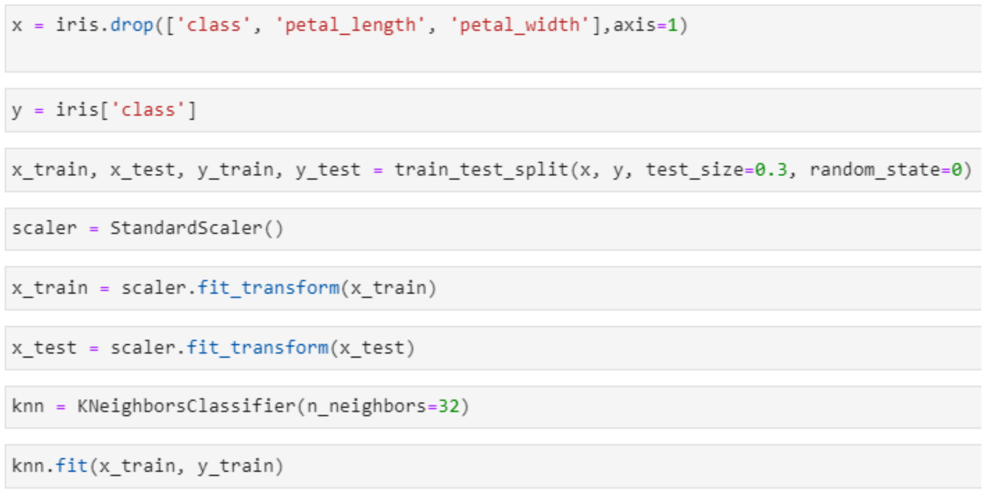
```
for i in range(1, 45):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    predict_i = knn.predict(x_test)
    error_rate.append(np.mean(predict_i != y_test))
```

```
plt.figure(figsize=(10,6))
error_rate
plt.plot(range(1,45), error_rate, color='red', linestyle='dashed', marker='X')
plt.title('Error Rate vs. K values of KNN')
plt.xlabel('K Values')
plt.ylabel('Error Rate')
```

```
Text(0, 0.5, 'Error Rate')
```



8. Now choose only the first 2 attributes of the iris dataset i.e., sepal-length, sepal-width and repeat the steps 3 to 7.



```
x = iris.drop(['class', 'petal_length', 'petal_width'],axis=1)
```

```
y = iris['class']
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

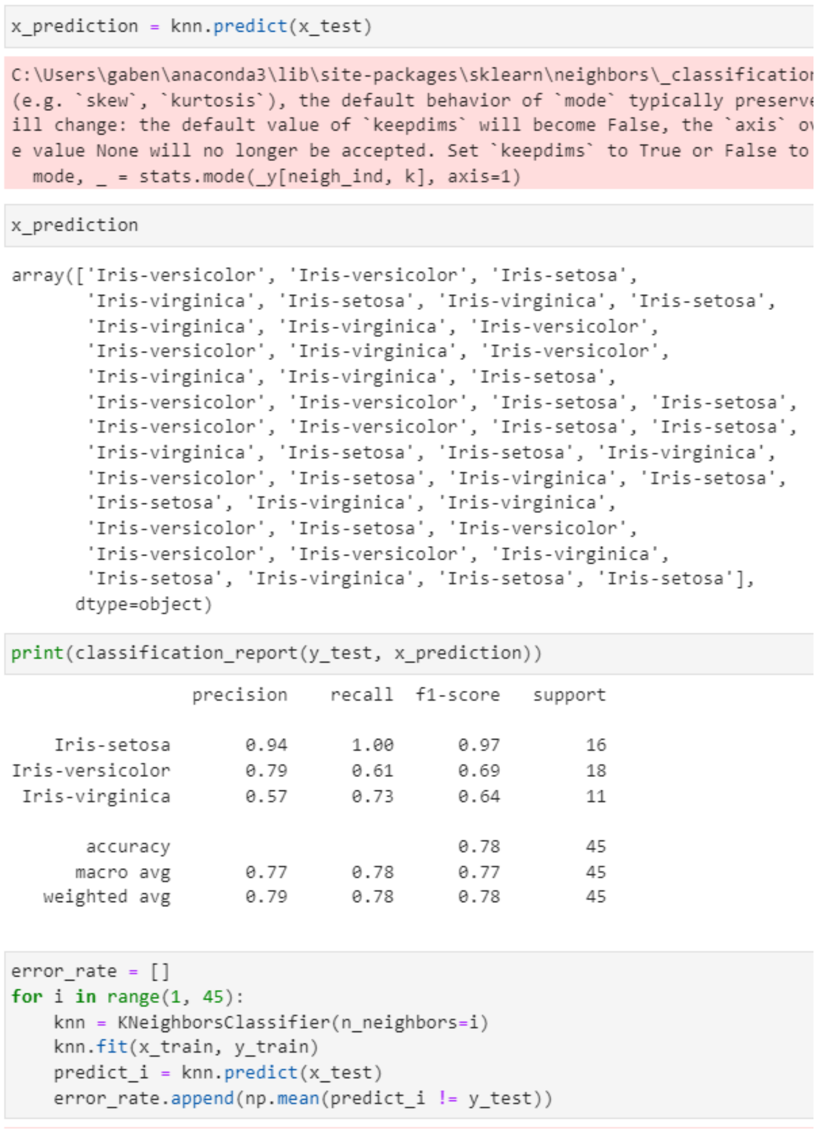
```
scaler = StandardScaler()
```

```
x_train = scaler.fit_transform(x_train)
```

```
x_test = scaler.fit_transform(x_test)
```

```
knn = KNeighborsClassifier(n_neighbors=32)
```

```
knn.fit(x_train, y_train)
```



```
x_prediction = knn.predict(x_test)
```

```
C:\Users\gaben\anaconda3\lib\site-packages\sklearn\neighbors\_classificatio
(e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserv
ill change: the default value of `keepdims` will become False, the `axis` o
e value None will no longer be accepted. Set `keepdims` to True or False to
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

```
x_prediction
```

```
array(['Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-virginica', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-virginica', 'Iris-virginica', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-setosa', 'Iris-virginica', 'Iris-setosa',
       'Iris-setosa', 'Iris-virginica', 'Iris-virginica',
       'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa'],
      dtype=object)
```

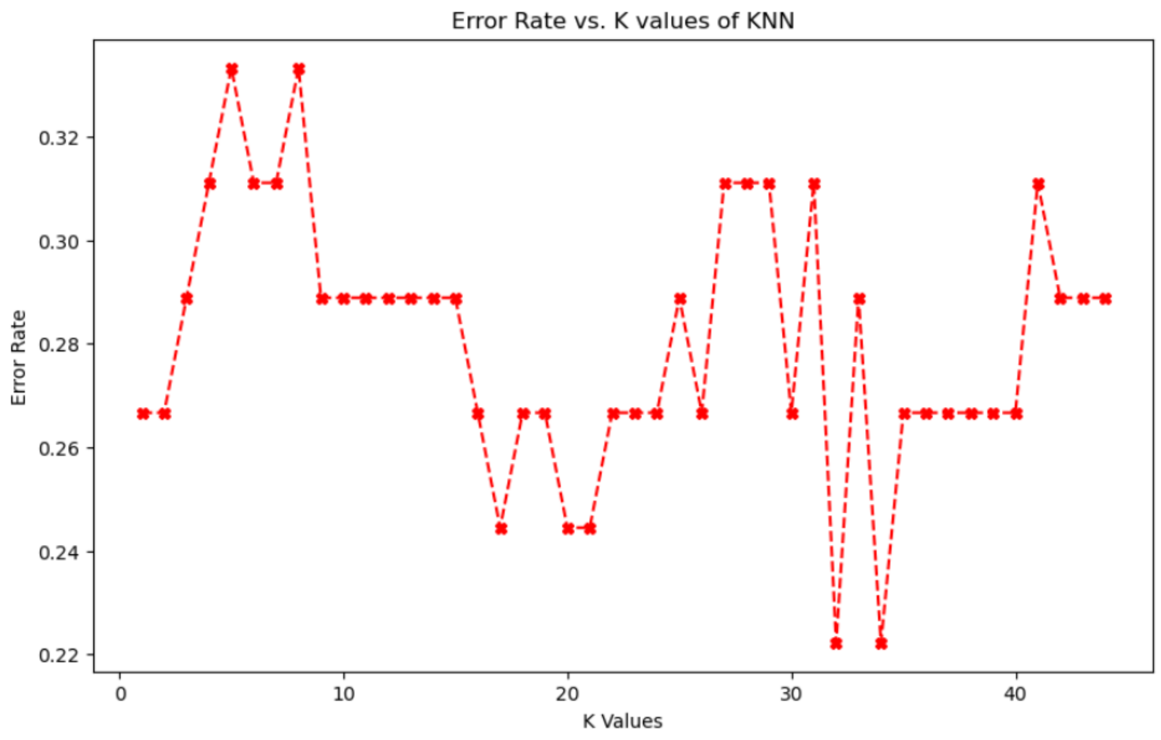
```
print(classification_report(y_test, x_prediction))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       0.94      1.00      0.97        16
Iris-versicolor       0.79      0.61      0.69        18
 Iris-virginica       0.57      0.73      0.64        11

       accuracy                           0.78        45
      macro avg       0.77      0.78      0.77        45
   weighted avg       0.79      0.78      0.78        45
```

```
error_rate = []
for i in range(1, 45):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    predict_i = knn.predict(x_test)
    error_rate.append(np.mean(predict_i != y_test))
```

```
plt.figure(figsize=(10,6))
error_rate
plt.plot(range(1,45), error_rate, color='red', linestyle='dashed', marker='X')
plt.title('Error Rate vs. K values of KNN')
plt.xlabel('K Values')
plt.ylabel('Error Rate')
```

```
Text(0, 0.5, 'Error Rate')
```



9. Compare the performance evaluation of the classifier with 4 attributes and classifier with 2 attributes. Explain your understanding.
    a. Based on the plots, the 4 attribute dataset proved to be more accurate than the 2 attribute dataset. This may be due to valuable information being removed on the second iteration without the last 2 attributes.

Conclusion:

In this project, I learned how to use pandas, seaborn, and pyplot. Even though I had prior experience with Pandas and Dataframes, I never used it in the application for data science and predicting values and related tasks.