PROJECT PORTFOLIO

/Research/Machine Learning with Iris

Project Summary:

This a personal project where I use machine learning to predict the species of flowers using dimensions like petal-lenght, petal-width, sepal lenght and sepal width.

This project uses a online csv dataset which is then converted into a pandas dataset and filtered for nulls and outliers. Following this, the K nearest neighbor machine learning algorithm is used to build the model.

The model is 90% accurate under the K nearest neighbor algorithm. Furthermore, filtering the dataset from outliers using the Interquartile range makes it 97% accurate.

Technical Skills Required

Python Programming

Jupyter NoteBook

Welcome to my Iris Machine Learning Project

I will begin by importing the libraries that I woud require for this project. These would include my pandas library to use as a dataset. matplotlib and seaborn for plotting, scipy and numpy for doing stati stical work like outlier detection, and sklearn for machine learning

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
import seaborn as sb
import scipy
import numpy as np
from scipy import stats
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
```

1.) Importing the Data Set

I will begin by using a CSV URL to import the data set and parse it into a pandas data frame.

In [2]:

```
url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'cla
ss']
irisset = pd.read_csv(url, names=names)
```

In [3]:

```
#Just to make sure the data is working
print(irisset.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
```

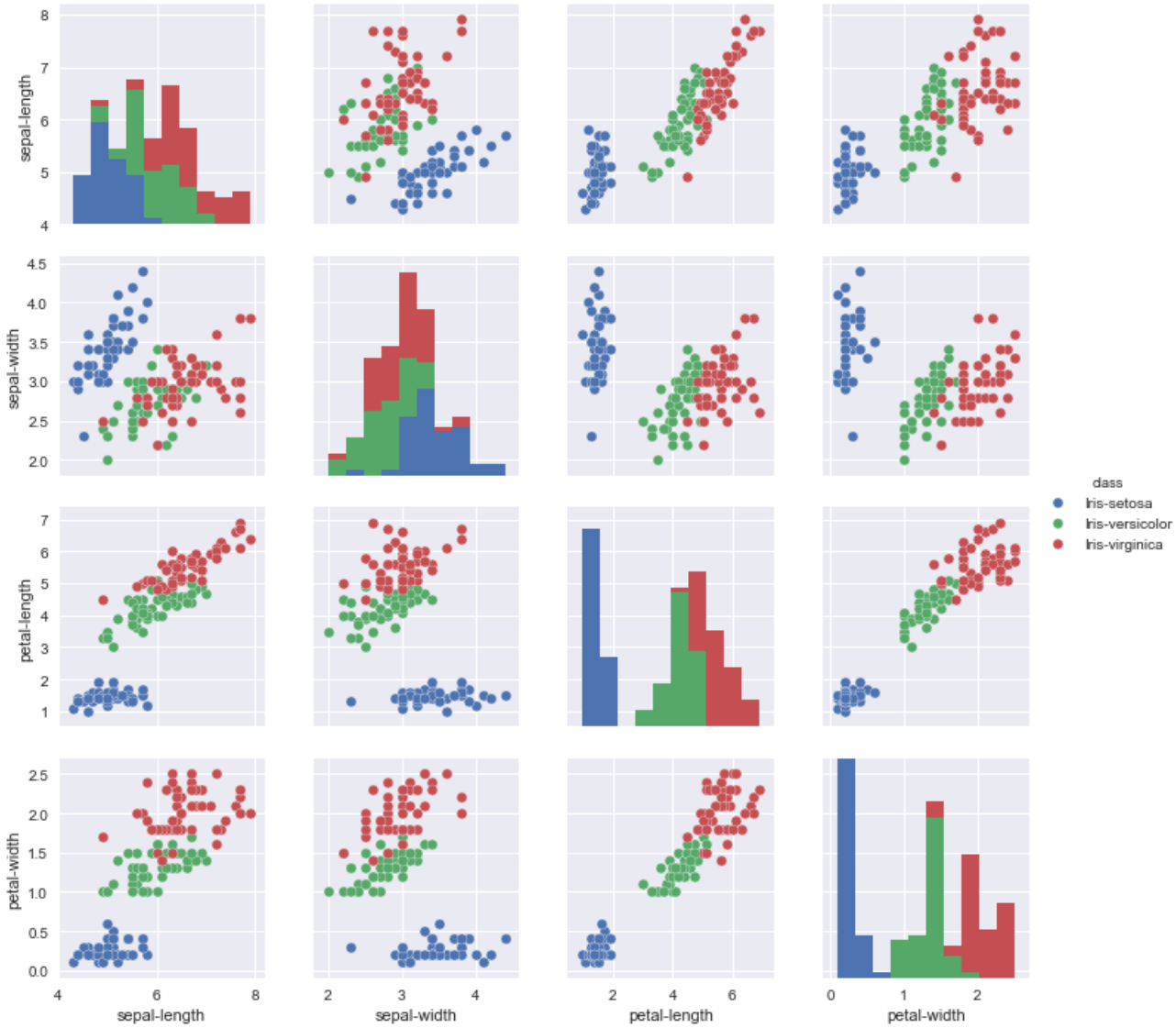
2.) Data Set Visualization and Cleaning

Following this, I will start visualizing the data to see what type o f insights I can get from these vizualizations and how I can deal wi

th nulls and outliers.

In [4]:

```
#I'll start with a seabon pairplot which creates a plot between each column
and row to see which variables have a relationship  with one another. I
can also create differentiate the color of the plot by a certain column. Si
nce I am going to use this project to find our how to find different classe
s of flowers, I will set class as the differentiator
sb.pairplot(irisset, hue="class")
plt.show()
```



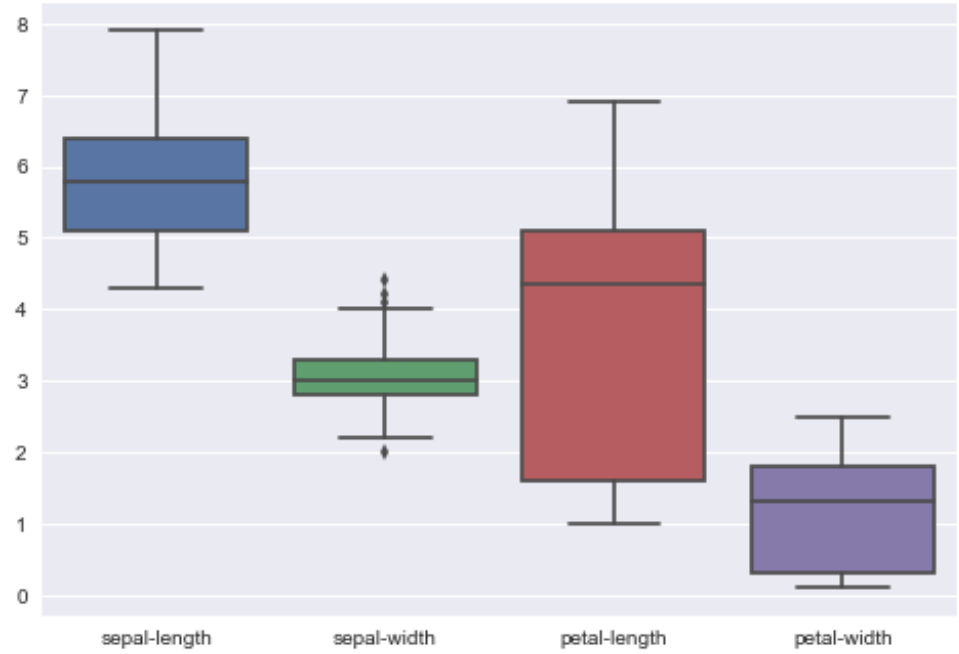
In [5]:

```
#Straightaway I can see some positive relationship between petal-width and
petal-lenght, petal lenght-sepal and lenght, petal-lenght and sepal lenght.
Furthermore, Just by doing this pairplot, I can tell that the Iris-Virginic
a is the largest flower followed by the Iris-versicolo and finally the
Iris-setosa. Looking at these clusters, I can go on to use the K-nearest ne
ighbour classifier. However, I will try out other algorithms as well.
```

In [6]:

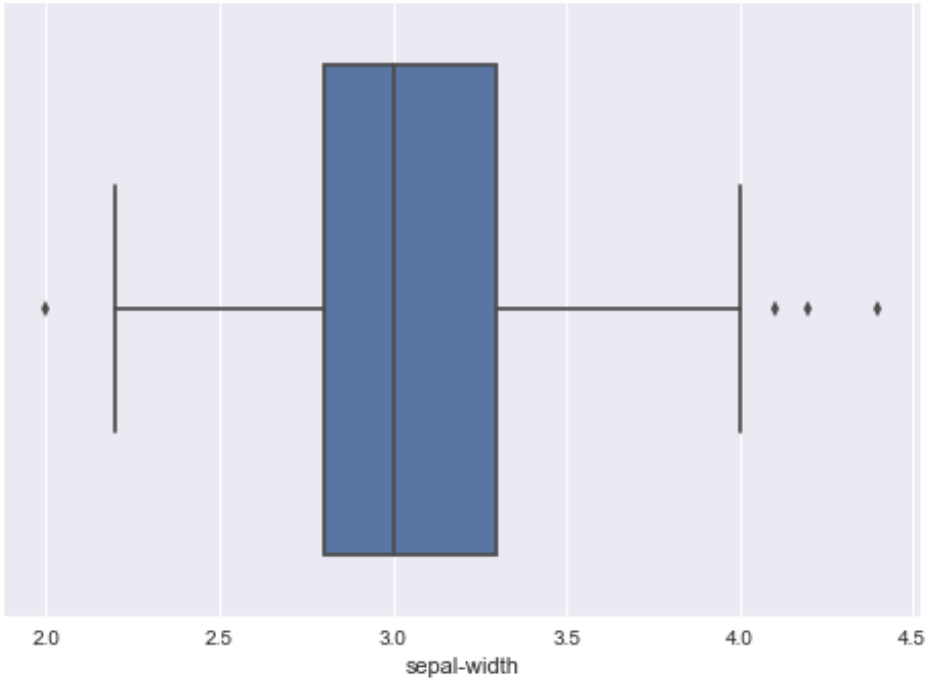
```
#I can also do a bopxplot to illustrate any possible outliers using the
IQR rule
sb.boxplot(data=irisset)
```

```
plt.show()
```



In [7]:

```
#In the above diagram, we can see that sepal-width has some outliers. Particularly 3 data points which is 3/2rd above the upper quartile and one data point 3/2rd below the lower quartile
sb.boxplot(irisset['sepal-width'])
plt.show()
```



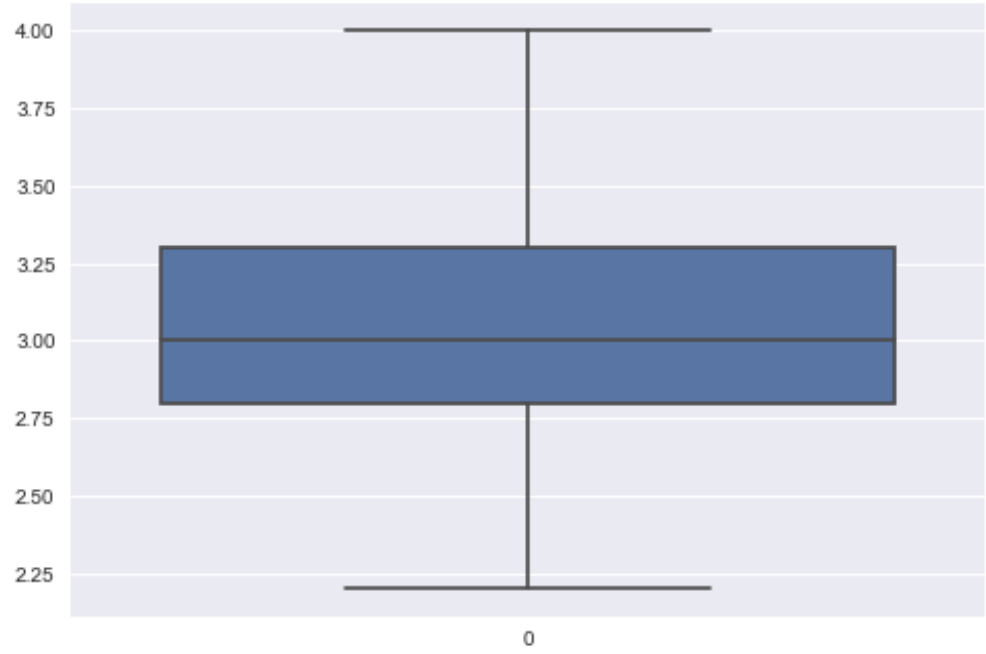
In [8]:

```
#To illustrate this mathematically, I need to find 3/2rd of upper quartile and 3/2rd o flower quartile and return an array. I can create a function to do this.
def outlier_remove(dfname,col):
    q1 = dfname[col].quantile(0.25)
    q3 = dfname[col].quantile(0.75)
    iqr = q3-q1 #Interquartile range
```

```
    fence_low  = q1-1.5*iqr
    fence_high = q3+1.5*iqr
    return dfname.loc[(irisset[col] > fence_low) & (dfname[col] < fence_high
)]
```

In [9]:

```
#Finally, I can use this function to create a new dataset which in turn
will not have the data points that are considered outliers using the IQR c
riteria.
irissetclean = outlier_remove(irisset,'sepal-width')
sb.boxplot(data=irissetclean['sepal-width'])
plt.show()
```



In [10]:

```
#Finally, I can check for any null values
irisset.isnull().values.any()
```

Out[10]:

```
False
```

3.) Machine Learning

Finally, I will begin the machine learning process. To do this I wil l split the data set into two parts. One will be the training set wh ich an algorithm will be used so the program can learn about the dif ferent species. The second set will then test how accurately the alg ortihm can figure out the species.

In [11]:

```
#First I will begin by taking all the columns aside from class and label th
at as X which would be my independant variables while the 4th column or cla
ss will be the variable the machine will try to assign
array = irisset.values
```

```
X = array[:,0:4]
Y = array[:,4]
#I will now split the data set into two parts. 80% of the data will be
held for the machine to learn while the other 20% will be predicted by the
computer and then cross checked with Y_validation.
X_train, X_validation, Y_train, Y_validation =
model_selection.train_test_split(X, Y, test_size=0.2, random_state=7)
```

In [12]:

```
#I will begin by setting up the K nearest neighbour and testing whether the
model works
mod = KNeighborsClassifier()
mod.fit(X_train, Y_train)
print(mod)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

In [13]:

```
#Now, I will crosscheck the model with y_validation and see how much of a
match I get.
Y_pred = mod.predict(X_validation)
print(classification_report(Y_validation,Y_pred))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         7
Iris-versicolor       0.85      0.92      0.88        12
 Iris-virginica       0.90      0.82      0.86        11

    avg / total       0.90      0.90      0.90        30
```

In [14]:

```
#Given that we had a high recalla and high accuracy of about 90%, we can sa
y that the model is near perfect.
#Interestingly, getting rid of outliers yield different results. We can
follow the same code for the filtered dataset
array = irissetclean.values
X = array[:,0:4]
Y = array[:,4]
X_train, X_validation, Y_train, Y_validation =
model_selection.train_test_split(X, Y, test_size=0.2, random_state=7)
mod = KNeighborsClassifier()
mod.fit(X_train, Y_train)
Y_pred = mod.predict(X_validation)
print(classification_report(Y_validation,Y_pred))
```

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         8
Iris-versicolor       0.90      1.00      0.95         9
 Iris-virginica       1.00      0.92      0.96        13

    avg / total       0.97      0.97      0.97        30
```

In [ ]:

```
```