Hello Kagglers!!

This is a very basic tutorial to Machine Learning for complete Beginners using the Iris Dataset. You can learn how to implement a machine learning to a given dataset by following this notebook. I have explained everything related to the implementation in detail . Hope you find it useful.

For a more advanced notebook that covers some more detailed concepts, have a look at this notebook (https://www.kaggle.com/ash316/ml-from-scratch-part-2/notebook)

If this notebook to be useful, **Please Upvote!!!**

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries insta
lled
# It is defined by the kaggle/python docker image: https://github.com/kaggle/
docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will li
st the files in the input directory

from subprocess import check_output
print(check_output(["ls", "../input"]).decode("utf8"))

# Any results you write to the current directory are saved as output.
```

```
Iris.csv
database.sqlite
```

In [2]:

```
iris = pd.read_csv("../input/Iris.csv") #load the dataset
```

In [3]:

```
iris.head(2) #show the first 2 rows from the dataset
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWid
0	1	5.1	3.5	1.4	
1	2	4.9	3.0	1.4	

In [4]:

```
iris.info()  #checking if there is any inconsistency in the dataset
#as we see there are no null values in the dataset, so the data can be proces
sed
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
Id               150 non-null int64
SepalLengthCm    150 non-null float64
SepalWidthCm     150 non-null float64
PetalLengthCm    150 non-null float64
PetalWidthCm     150 non-null float64
Species          150 non-null object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.1+ KB
```

Removing the unneeded column

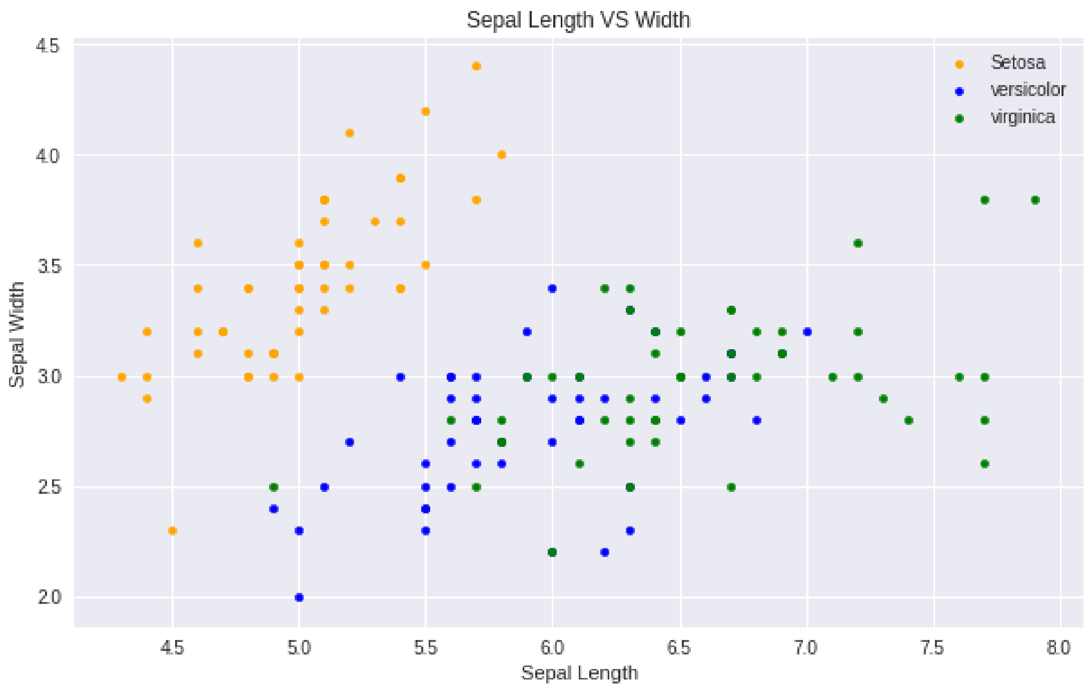
In [5]:

```
iris.drop('Id',axis=1,inplace=True) #dropping the Id column as it is unecessa
ry, axis=1 specifies that it should be column wise, inplace =1 means the chan
ges should be reflected into the dataframe
```

Some Exploratory Data Analysis With Iris

In [6]:

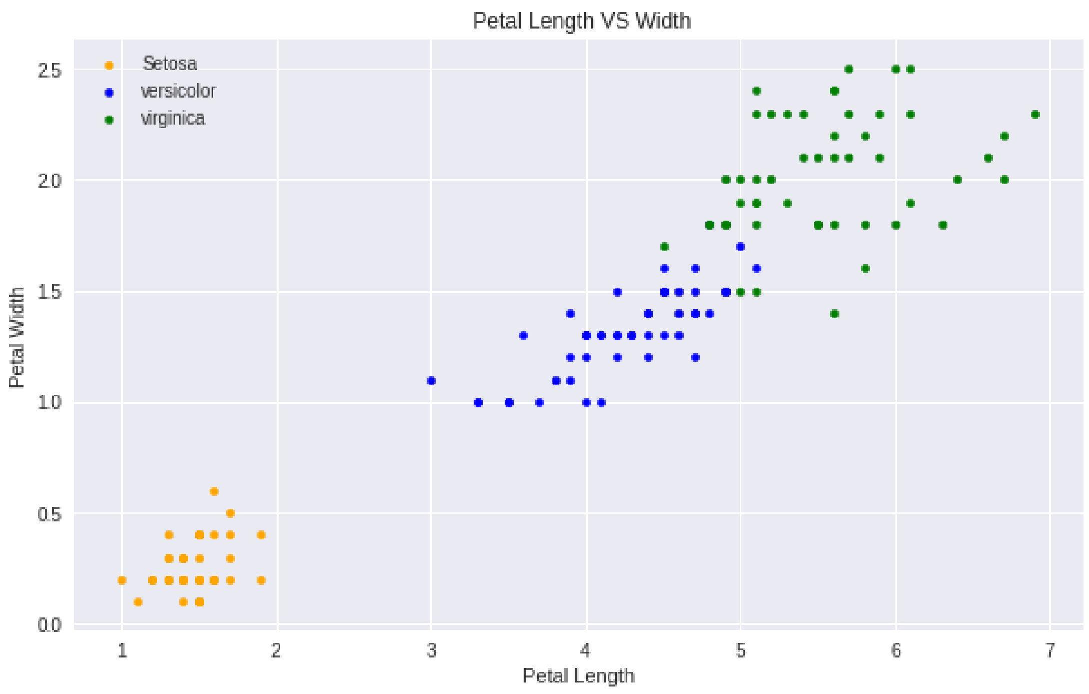
```
fig = iris[iris.Species=='Iris-setosa'].plot(kind='scatter',x='SepalLengthCm'
,y='SepalWidthCm',color='orange', label='Setosa')
iris[iris.Species=='Iris-versicolor'].plot(kind='scatter',x='SepalLengthCm',y
='SepalWidthCm',color='blue', label='versicolor',ax=fig)
iris[iris.Species=='Iris-virginica'].plot(kind='scatter',x='SepalLengthCm',y=
'SepalWidthCm',color='green', label='virginica', ax=fig)
fig.set_xlabel("Sepal Length")
fig.set_ylabel("Sepal Width")
fig.set_title("Sepal Length VS Width")
fig=plt.gcf()
fig.set_size_inches(10,6)
plt.show()
```



The above graph shows relationship between the sepal length and width. Now we will check relationship between the petal length and width.

In [7]:

```
fig = iris[iris.Species=='Iris-setosa'].plot.scatter(x='PetalLengthCm',y='PetalWidthCm',color='orange', label='Setosa')
iris[iris.Species=='Iris-versicolor'].plot.scatter(x='PetalLengthCm',y='PetalWidthCm',color='blue', label='versicolor',ax=fig)
iris[iris.Species=='Iris-virginica'].plot.scatter(x='PetalLengthCm',y='PetalWidthCm',color='green', label='virginica', ax=fig)
fig.set_xlabel("Petal Length")
fig.set_ylabel("Petal Width")
fig.set_title(" Petal Length VS Width")
fig=plt.gcf()
fig.set_size_inches(10,6)
plt.show()
```

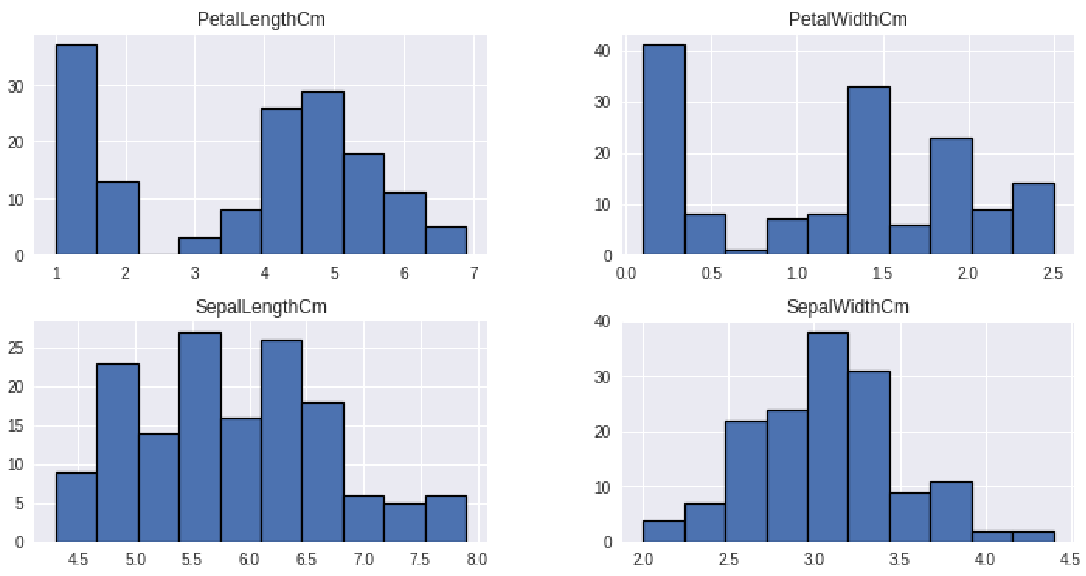


As we can see that the Petal Features are giving a better cluster division compared to the Sepal features. This is an indication that the Petals can help in better and accurate Predictions over the Sepal. We will check that later.

Now let us see how are the length and width are distributed

In [8]:

```
iris.hist(edgecolor='black', linewidth=1.2)
fig=plt.gcf()
fig.set_size_inches(12,6)
plt.show()
```



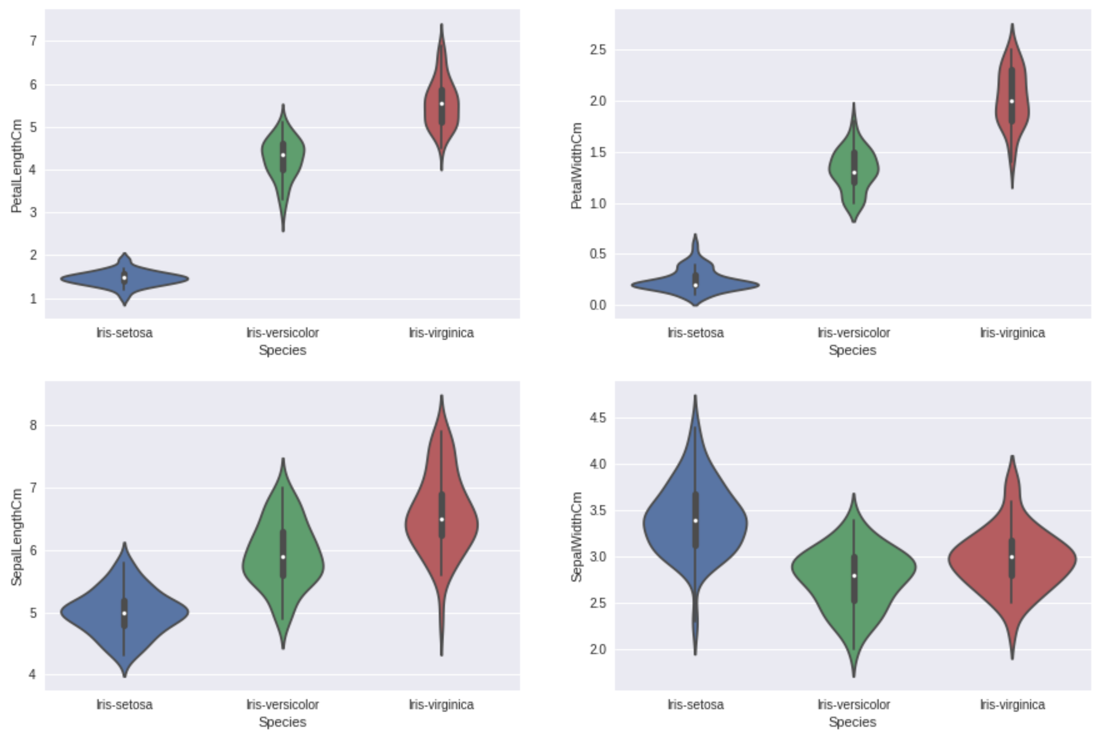
Now let us see how the length and width vary according to the species

In [9]:

```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='Species',y='PetalLengthCm',data=iris)
plt.subplot(2,2,2)
sns.violinplot(x='Species',y='PetalWidthCm',data=iris)
plt.subplot(2,2,3)
sns.violinplot(x='Species',y='SepalLengthCm',data=iris)
plt.subplot(2,2,4)
sns.violinplot(x='Species',y='SepalWidthCm',data=iris)
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f846388ee10>
```



The violinplot shows density of the length and width in the species. The thinner part denotes that there is less density whereas the fatter part conveys higher density

Now the given problem is a classification problem.. Thus we will be using the classification algorithms to build a model.

Classification: samples belong to two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data

Regression: if the desired output consists of one or more continuous variables, then the task is called regression. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight.

Before we start, we need to clear some ML notations.

attributes-->An attribute is a property of an instance that may be used to determine its classification. In the following dataset, the attributes are the petal and sepal length and width. It is also known as **Features**.

Target variable, in the machine learning context is the variable that is or should be the output. Here the target variables are the 3 flower species.

In [10]:

```
# importing alll the necessary packages to use the various classification alg
orithms
from sklearn.linear_model import LogisticRegression  # for Logistic Regressio
n algorithm
from sklearn.cross_validation import train_test_split #to split the dataset f
or training and testing
from sklearn.neighbors import KNeighborsClassifier  # for K nearest neighbour
s
from sklearn import svm  #for Support Vector Machine (SVM) Algorithm
from sklearn import metrics #for checking the model accuracy
from sklearn.tree import DecisionTreeClassifier #for using Decision Tree Algo
ithm
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/cross_validatio
n.py:41: DeprecationWarning: This module was deprecated in ver
sion 0.18 in favor of the model_selection module into which al
l the refactored classes and functions are moved. Also note th
at the interface of the new CV iterators are different from th
at of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [11]:

```
iris.shape #get the shape of the dataset
```

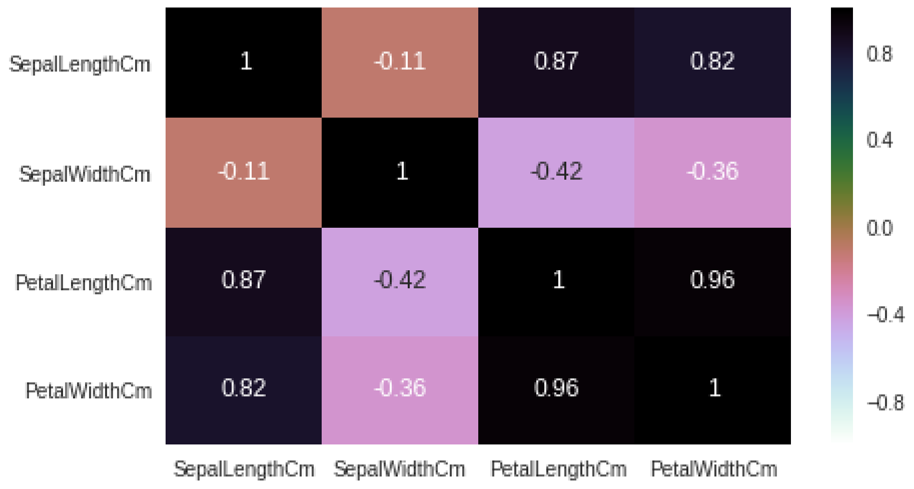
Out[11]:

```
(150, 5)
```

Now, when we train any algorithm, the number of features and their correlation plays an important role. If there are features and many of the features are highly correlated, then training an algorithm with all the featues will reduce the accuracy. Thus features selection should be done carefully. This dataset has less featues but still we will see the correlation.

In [12]:

```
plt.figure(figsize=(7,4))
sns.heatmap(iris.corr(),annot=True,cmap='cubehelix_r') #draws  heatmap with i
nput as the correlation matrix calculted by(iris.corr())
plt.show()
```



Observation--->

The Sepal Width and Length are not correlated The Petal Width and Length are highly correlated

We will use all the features for training the algorithm and check the accuracy.

Then we will use 1 Petal Feature and 1 Sepal Feature to check the accuracy of the algorithm as we are using only 2 features that are not correlated. Thus we can have a variance in the dataset which may help in better accuracy. We will check it later.

Steps To Be followed When Applying an Algorithm

1. Split the dataset into training and testing dataset. The testing dataset is generally smaller than training one as it will help in training the model better.
2. Select any algorithm based on the problem (classification or regression) whatever you feel may be good.
3. Then pass the training dataset to the algorithm to train it. We use the **.fit()** method
4. Then pass the testing data to the trained algorithm to predict the outcome. We use the **.predict()** method.
5. We then check the accuracy by **passing the predicted outcome and the actual output** to the model.

Splitting The Data into Training And Testing Dataset

In [13]:

```
train, test = train_test_split(iris, test_size = 0.3)# in this our main data
 is split into train and test
# the attribute test_size=0.3 splits the data into 70% and 30% ratio. train=7
0% and test=30%
print(train.shape)
print(test.shape)
```

```
(105, 5)
(45, 5)
```

In [14]:

```
train_X = train[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthC
m']]# taking the training data features
train_y=train.Species# output of our training data
test_X= test[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']]
# taking test data features
test_y =test.Species   #output value of test data
```

Lets check the Train and Test Dataset

In [15]:

```
train_X.head(2)
```

Out[15]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidth
75	6.6	3.0	4.4	
16	5.4	3.9	1.3	

In [16]:

```
test_X.head(2)
```

Out[16]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidt
91	6.1	3.0	4.6	
134	6.1	2.6	5.6	

In [17]:

```
train_y.head()  ##output of the training data
```

Out[17]:

```
75     Iris-versicolor
16         Iris-setosa
25         Iris-setosa
101     Iris-virginica
125     Iris-virginica
Name: Species, dtype: object
```

Support Vector Machine (SVM)

In [18]:

```
model = svm.SVC() #select the algorithm
model.fit(train_X,train_y) # we train the algorithm with the training data an
d the training output
prediction=model.predict(test_X) #now we pass the testing data to the trained
algorithm
print('The accuracy of the SVM is:',metrics.accuracy_score(prediction,test_y
))#now we check the accuracy of the algorithm.
#we pass the predicted output by the model and the actual output
```

```
The accuracy of the SVM is: 1.0
```

SVM is giving very good accuracy . We will continue to check the accuracy for different models.

Now we will follow the same steps as above for training various machine learning algorithms.

Logistic Regression

In [19]:

```
model = LogisticRegression()
model.fit(train_X,train_y)
prediction=model.predict(test_X)
print('The accuracy of the Logistic Regression is',metrics.accuracy_score(pre
diction,test_y))
```

```
The accuracy of the Logistic Regression is 0.9777777777778
```

Decision Tree

In [20]:

```
model=DecisionTreeClassifier()
model.fit(train_X,train_y)
prediction=model.predict(test_X)
print('The accuracy of the Decision Tree is',metrics.accuracy_score(predictio
n,test_y))
```

```
The accuracy of the Decision Tree is 0.9777777777778
```

K-Nearest Neighbours

In [21]:

```
model=KNeighborsClassifier(n_neighbors=3) #this examines 3 neighbours for put
ting the new data into a class
model.fit(train_X,train_y)
prediction=model.predict(test_X)
print('The accuracy of the KNN is',metrics.accuracy_score(prediction,test_y))
```

```
The accuracy of the KNN is 0.955555555556
```

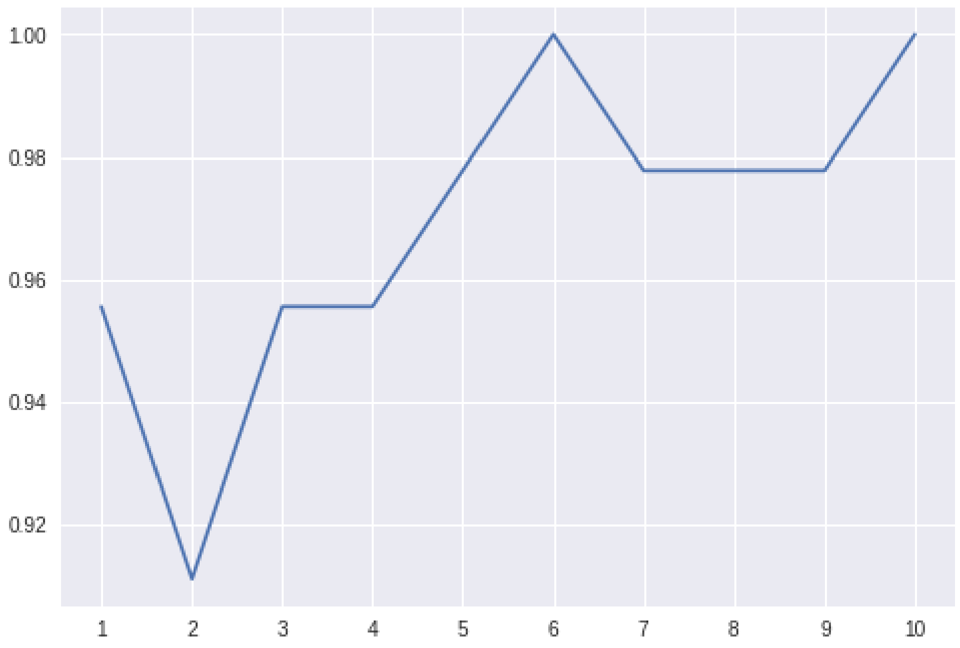
Let's check the accuracy for various values of n for K-Nearest nerighbours

In [22]:

```
a_index=list(range(1,11))
a=pd.Series()
x=[1,2,3,4,5,6,7,8,9,10]
for i in list(range(1,11)):
    model=KNeighborsClassifier(n_neighbors=i)
    model.fit(train_X,train_y)
    prediction=model.predict(test_X)
    a=a.append(pd.Series(metrics.accuracy_score(prediction,test_y)))
plt.plot(a_index, a)
plt.xticks(x)
```

Out[22]:

```
([<matplotlib.axis.XTick at 0x7f8456e94668>,
  <matplotlib.axis.XTick at 0x7f8456f0a8d0>,
  <matplotlib.axis.XTick at 0x7f8456eb7ba8>,
  <matplotlib.axis.XTick at 0x7f8463954b70>,
  <matplotlib.axis.XTick at 0x7f8463b695f8>,
  <matplotlib.axis.XTick at 0x7f8463b69278>,
  <matplotlib.axis.XTick at 0x7f8463baa048>,
  <matplotlib.axis.XTick at 0x7f8463bb56d8>,
  <matplotlib.axis.XTick at 0x7f8463bb55c0>,
  <matplotlib.axis.XTick at 0x7f8463bfb908>],
 <a list of 10 Text xticklabel objects>)
```



Above is the graph showing the accuracy for the KNN models using different values of n.

We used all the features of iris in above models. Now we will use Petals and Sepals Seperately

Creating Petals And Sepals Training Data

In [23]:

```
petal=iris[['PetalLengthCm','PetalWidthCm','Species']]
sepal=iris[['SepalLengthCm','SepalWidthCm','Species']]
```

In [24]:

```
train_p,test_p=train_test_split(petal,test_size=0.3,random_state=0)  #petals
train_x_p=train_p[['PetalWidthCm','PetalLengthCm']]
train_y_p=train_p.Species
test_x_p=test_p[['PetalWidthCm','PetalLengthCm']]
test_y_p=test_p.Species


train_s,test_s=train_test_split(sepal,test_size=0.3,random_state=0)  #Sepal
train_x_s=train_s[['SepalWidthCm','SepalLengthCm']]
train_y_s=train_s.Species
test_x_s=test_s[['SepalWidthCm','SepalLengthCm']]
test_y_s=test_s.Species
```

SVM

In [25]:

```
model=svm.SVC()
model.fit(train_x_p,train_y_p)
prediction=model.predict(test_x_p)
print('The accuracy of the SVM using Petals is:',metrics.accuracy_score(predi
ction,test_y_p))

model=svm.SVC()
model.fit(train_x_s,train_y_s)
prediction=model.predict(test_x_s)
print('The accuracy of the SVM using Sepal is:',metrics.accuracy_score(predic
tion,test_y_s))
```

```
The accuracy of the SVM using Petals is: 0.9777777777777778
The accuracy of the SVM using Sepal is: 0.8
```

Logistic Regression

In [26]:

```
model = LogisticRegression()
model.fit(train_x_p,train_y_p)
prediction=model.predict(test_x_p)
print('The accuracy of the Logistic Regression using Petals is:',metrics.accu
racy_score(prediction,test_y_p))

model.fit(train_x_s,train_y_s)
prediction=model.predict(test_x_s)
print('The accuracy of the Logistic Regression using Sepals is:',metrics.accu
racy_score(prediction,test_y_s))
```

```
The accuracy of the Logistic Regression using Petals is: 0.688
888888889
The accuracy of the Logistic Regression using Sepals is: 0.644
444444444
```

Decision Tree

In [27]:

```
model=DecisionTreeClassifier()
model.fit(train_x_p,train_y_p)
prediction=model.predict(test_x_p)
print('The accuracy of the Decision Tree using Petals is:',metrics.accuracy_s
core(prediction,test_y_p))

model.fit(train_x_s,train_y_s)
prediction=model.predict(test_x_s)
print('The accuracy of the Decision Tree using Sepals is:',metrics.accuracy_s
core(prediction,test_y_s))
```

```
The accuracy of the Decision Tree using Petals is: 0.955555555
556
The accuracy of the Decision Tree using Sepals is: 0.644444444
444
```

K-Nearest Neighbours

In [28]:

```
model=KNeighborsClassifier(n_neighbors=3)
model.fit(train_x_p,train_y_p)
prediction=model.predict(test_x_p)
print('The accuracy of the KNN using Petals is:',metrics.accuracy_score(predi
ction,test_y_p))

model.fit(train_x_s,train_y_s)
prediction=model.predict(test_x_s)
print('The accuracy of the KNN using Sepals is:',metrics.accuracy_score(predi
ction,test_y_s))
```

```
The accuracy of the KNN using Petals is: 0.977777777778
The accuracy of the KNN using Sepals is: 0.733333333333
```

Observations:

- Using Petals over Sepal for training the data gives a much better accuracy.
- This was expected as we saw in the heatmap above that the correlation between the Sepal Width and Length was very low whereas the correlation between Petal Width and Length was very high.

Thus we have just implemented some of the common Machine Learning. Since the dataset is small with very few features, I didn't cover some concepts as they would be relevant when we have many features.

I have compiled a notebook covering some advanced ML concepts using a larger dataset. Have a look at that tooo.

I hope the notebook was useful to you to get started with Machine Learning.

If find this notebook, **Please Upvote**.

Thank You!!

```
In [29]:
```