Naive Bayes Classification

Importing the libraries

In [31]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import pandas as pd
```

Importing the dataset

In [38]:
```
dataset = pd.read_csv(r'C:\Users\Bhaskar\Desktop\iris.csv')
```

looking at the first 5 values of the dataset

In [39]:
```
dataset.head()
```

Out[39]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [40]:
```
%matplotlib inline
img=mpimg.imread(r'C:\Users\Bhaskar\Desktop\iris_types.jpg')
plt.figure(figsize=(20,40))
plt.axis('off')
plt.imshow(img)
```

Out[40]:
```
<matplotlib.image.AxesImage at 0x22907948e88>
```

Spliting the dataset in independent and

dependent variables

In [42]:
```
X = dataset.iloc[:,:4].values
y = dataset['Species'].values
```

Splitting the dataset into the Training set and Test set

In [43]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,
```

Feature Scaling to bring the variable in a single scale

In [44]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Fitting Naive Bayes Classification to the Training set with linear kernel

In [45]:
```
from sklearn.naive_bayes import GaussianNB
nvclassifier = GaussianNB()
nvclassifier.fit(X_train, y_train)
```

Out[45]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

Predicting the Test set results

In [46]:
```
y_pred = nvclassifier.predict(X_test)
print(y_pred)
```

```
['Iris-virginica' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-virginica' 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'
 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor' 'Iris-setosa'
 'Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa' 'Iris-virginica'
 'Iris-versicolor']
```

lets see the actual and predicted value side by side

In [47]:
```
y_compare = np.vstack((y_test,y_pred)).T
#actual value on the left side and predicted value on the right hand side
#printing the top 5 values
y_compare[:5,:]
```

Out[47]:
```
array([['Iris-virginica', 'Iris-virginica'],
       ['Iris-virginica', 'Iris-virginica'],
       ['Iris-setosa', 'Iris-setosa'],
       ['Iris-setosa', 'Iris-setosa'],
       ['Iris-setosa', 'Iris-setosa']], dtype=object)
```

Making the Confusion Matrix

In [48]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[11  0  0]
 [ 0  9  0]
 [ 0  0 10]]
```

finding accuracy from the confusion matrix.

In [49]:
```
a = cm.shape
corrPred = 0
falsePred = 0

for row in range(a[0]):
    for c in range(a[1]):
        if row == c:
            corrPred +=cm[row,c]
        else:
            falsePred += cm[row,c]
print('Correct predictions: ', corrPred)
print('False predictions', falsePred)
print ('\n\nAccuracy of the Naive Bayes Clasification is: ', corrPred/(cm.s
```

```
Correct predictions:  30
False predictions 0


Accuracy of the Naive Bayes Clasification is:  1.0
```

In [ ]:

In [ ]:

In [ ]: