Feature Extraction

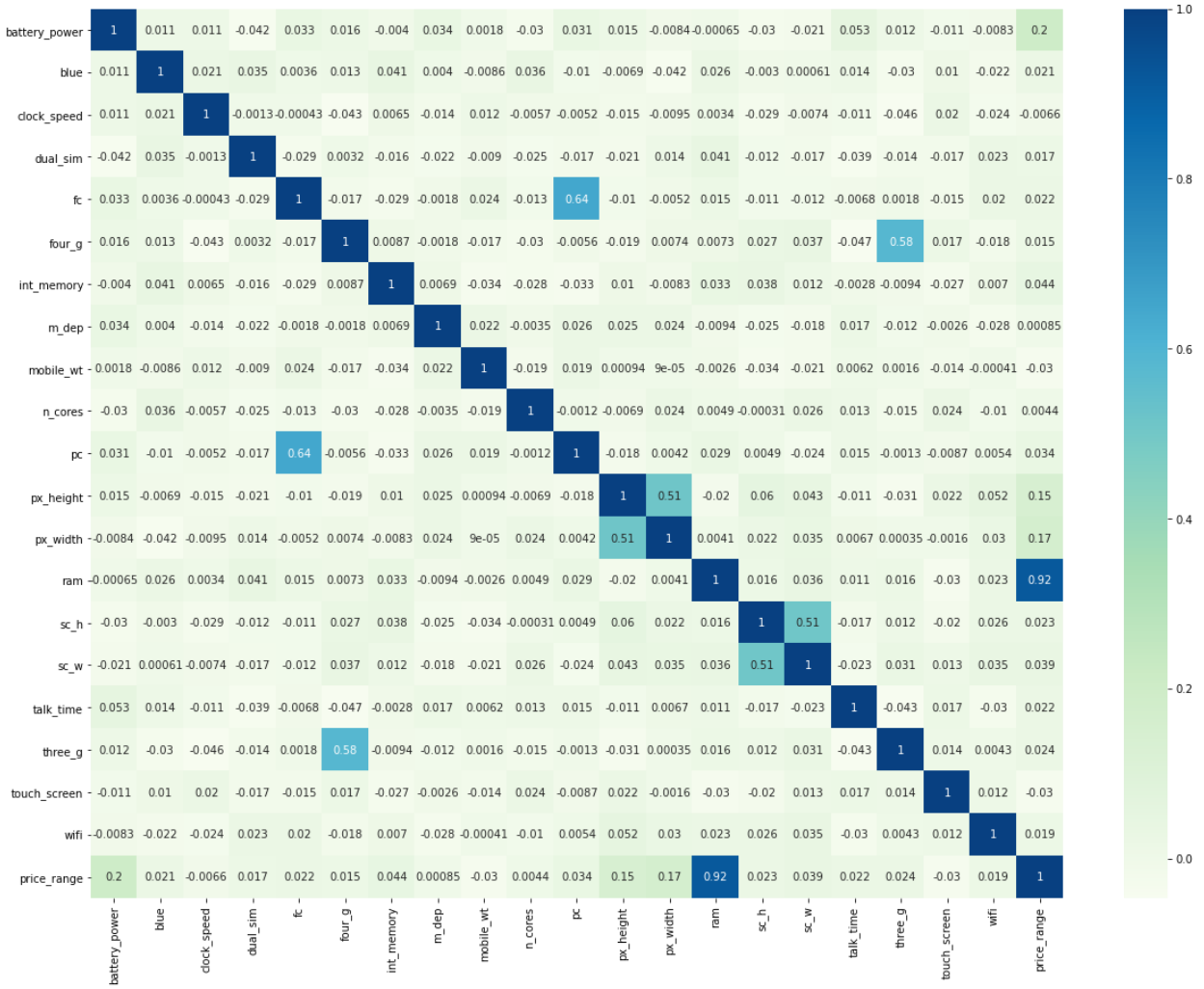
Bob has started his own mobile company. He wants to compete with big companies like Apple, Samsung etc. He does not know how to estimate the price of mobiles that are manufactured by his company. To solve this problem Bob collected sales data of mobile phones manufactured by other companies. Bob wants to find out some relation between the technical features of a mobile phone (eg:- RAM, Internal Memory etc) and its selling price.

You are required to analyse this data and help Bob to estimate the price of the mobiles his company makes. Thus,

1- Start by analysing and understanding the features that have the significant effect on the price of mobile phones. e.g calculate the correlation between the independent variables (features) and the price range.

Answer:

First we need to explore the data that allows us to understand the contents of the given data, starting from the **read_csv** function that reads the (comma separated file) csv file, than perform some statistics with the help of **describe()** function that allows us to understand our data more clear and is used to display summary statistics for the numeric attributes of dataset, including the count, mean, minimum and maximum values. In practice, understanding the context of the data is also considered very important because it will answer the basic problems to be analyzed.
From Train_Data set we extract features that affects the price range of mobile phone, this Data has 21 columns out of which 20 are the features columns that needs to be compare with price range, so we separated these columns in Feature parameter and price_range column in PriceRange with help of **iloc** function.
corr() allows us to find the correlation for each column. This function can determine which data is needed and which data has the most influence on other data features and has the highest correlation rate among the others to see the relationship of various mobile phone conditions on the price of the mobile phone.



After perform correlation we found out that 4 features in our given data affects the price range of mobile phone that are,

Ram
Battery Power
Pixel Width
Pixel Height
Internal Memory.

In [49]:
```
# Taking value from above data
High_value = pd.DataFrame(F_corr_PR)
High_value.columns = ['High_value']
High_value['High_value'].nlargest() # .nlargest show high values in a list.....
# It shows that these features will effect the price range of phone as customers tends to buy a phone with these features
# from above it shows ram has the max effect in purchasing
```

Out[49]:
```
ram             0.917046
battery_power   0.200723
px_width        0.165818
px_height       0.148858
int_memory      0.044435
Name: High_value, dtype: float64
```

2- To simplify the classification task, transform the output variable (price range) from [0, 1, 2, 3] to [0, 1] and remove the target variable from the dataset. i.e 0→0, while 1,2,3→1.

In order to perform Logistic Regression we need to simplify the output variable of price range into the category of 0's and 1's, by simply transform the target variable from dataset i.e 0-→ into 0 while 1,2,3-→ into 1 categories.
I use numpy (**np.where()**) function for the transformation of variables in price range column and passed the argument with desired action.

2: Classification Task

In [51]:
```
New_df = pd.DataFrame(PriceRange) #create new data frame and stores price_range column in it...
Price_Range_Trans = np.array(New_df) # then convert this column into array....
# with the help of numpy where function creates a new data frame with our desire values
df = df2 = np.where(Price_Range_Trans<1,Price_Range_Trans,1)
Trans = pd.DataFrame(df)
Trans.columns = ['Price_Range_Trans']
pd.concat([New_df, Trans], axis=1) # join both data sets to show the comparision........
```

Out[51]:

	price_range	Price_Range_Trans
0	1	1
1	2	1
2	2	1
3	2	1
4	1	1
...	...	...
1995	0	0
1996	2	1
1997	3	1
1998	0	0
1999	3	1

2000 rows × 2 columns

3- Based on the set of features that you identify as the most likely features to affect the price of mobile phones, use Logistic regression to perform the classification task.

Logistics Regression:

Logistic regression is a supervised learning algorithm that uses labels to train the model. Supervised learning algorithms have input variable **X** and a target variable **Y** for training the model. Logistic regression predicts the possibility of a particular event happening. Mainly used for classification and the response variable is discrete.

3: Likely Features that affects the price of mobile phones

In [53]:
```
# for test train split uses correlated columns with classified price_range
X = Correlated_to
Y = Price_Range_Trans
```

In [54]:
```
# take 20% for testing and 80% for training
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=0)
```

In [55]:
```
print(X_train.shape)
print(Y_train.shape)
```

```
(1600, 4)
(1600, 1)
```

In [56]:
```
print(X_test.shape)
print(Y_test.shape)
```

```
(400, 4)
(400, 1)
```

In [57]:
```
# fit logistic regression model....
from sklearn.linear_model import LogisticRegression
logistic_Regression_Model = LogisticRegression()
logistic_Regression_Model.fit(X_train,Y_train)
```

Out[57]:
```
LogisticRegression()
```

In [58]:
```
# predicted the values
print(logistic_Regression_Model.predict(X_test))
```

```
[1 0 1 1 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 0 1 1 0
 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1
 0 0 1 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0
 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1
 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0 0 1 1 1 1 1 1
 0 0 1 1 1 1 0 1 0 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 1 1 1
 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 0 1 1 0 1 1 1
 1 0 1 0 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0
 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0
 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 1 1 1 1 0 1 0]
```

In [59]:
```
Y_pred=logistic_Regression_Model.predict(X_test)
```

In [60]:
```
# confusion metric is used to describe the performance of classifier.......
import sklearn.metrics as metrics
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
Confusion_Matrix = metrics.confusion_matrix(Y_test, Y_pred)
Confusion_Matrix
```

Out[60]:
```
array([[ 91,   4],
       [  2, 303]], dtype=int64)
```

In [61]:
```
# accuracy score of predicted data set....
accuracy_Train_Model = metrics.accuracy_score(Y_test, Y_pred)
accuracy_Train_Model
```

Out[61]:
```
0.985
```

In [62]:
```
Accuracy_Train = (accuracy_Train_Model)*100
print("Accuracy {:0.1f}%:".format(Accuracy_Train))
```

4- Asses the accuracy of your classifier.

Answer:

After performing the logistic regression on Train_Dataset, we got the accuracy of 98.5% from our model, then we join the predicted column of price range into our Test_Dataset in order to predict the accuracy of our classifier, we got 99.5% of accuracy, which means that our model is 99.5% accurate in predicting the price range of mobile phone based on the features that are highly correlated with price range in Train_Dataset.

```
In [77]: y_pred=logistic_Regression_Model_Test.predict(x_test)
```

```
In [78]: # confusion metric is used to describe the performance of classifier.......
import sklearn.metrics as metrics
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
confusion_matrix
```

```
Out[78]: array([[ 51,   1],
                [  0, 148]], dtype=int64)
```

```
In [79]: accuracy_Test_Model = metrics.accuracy_score(y_test, y_pred)
accuracy_Test_Model
```

```
Out[79]: 0.995
```

```
In [80]: # we achieve accuracy for our predicted values....
Accuracy_Test = (accuracy_Test_Model)*100
print("Accuracy {:0.1f}%:".format(Accuracy_Test))
```

```
Accuracy 99.5%:
```

Analysis of Given Data:

For this dataset we observe that the heat map with values 0 corresponds to no correlation while value 1 corresponds to highest correlation. It shows that most of features are not well correlated to the price range while we observe five features that are highly correlated like ram, battery power, pixel height, pixel width and internal memory means that with these features affects price range of the mobile phone while ram has the highest correlated values as this feature is super important in deciding the price range.

After that we perform logistic regression to train our Train data model and predicted the price range for given Test dataset, we perform test train split with the features that are correlated to price range split the data set into test split with 20% while 80% in train split and we achieve 98.5% accuracy with our logistic regression model. After the we take our price range column join into Test data set and again perform Test Train split to calculate the accuracy of model, we achieve 99.5% accuracy means that the data which is not seen by the classifier after seeing, it predicted the accuracy of our model, good precision is achieve by our model and it predicted the price range for mobile phone.

The complete code of this analysis is attach separately in 200218625_SML_Q#1.ipynb file and it is provided with this pdf.

K-means Clustering

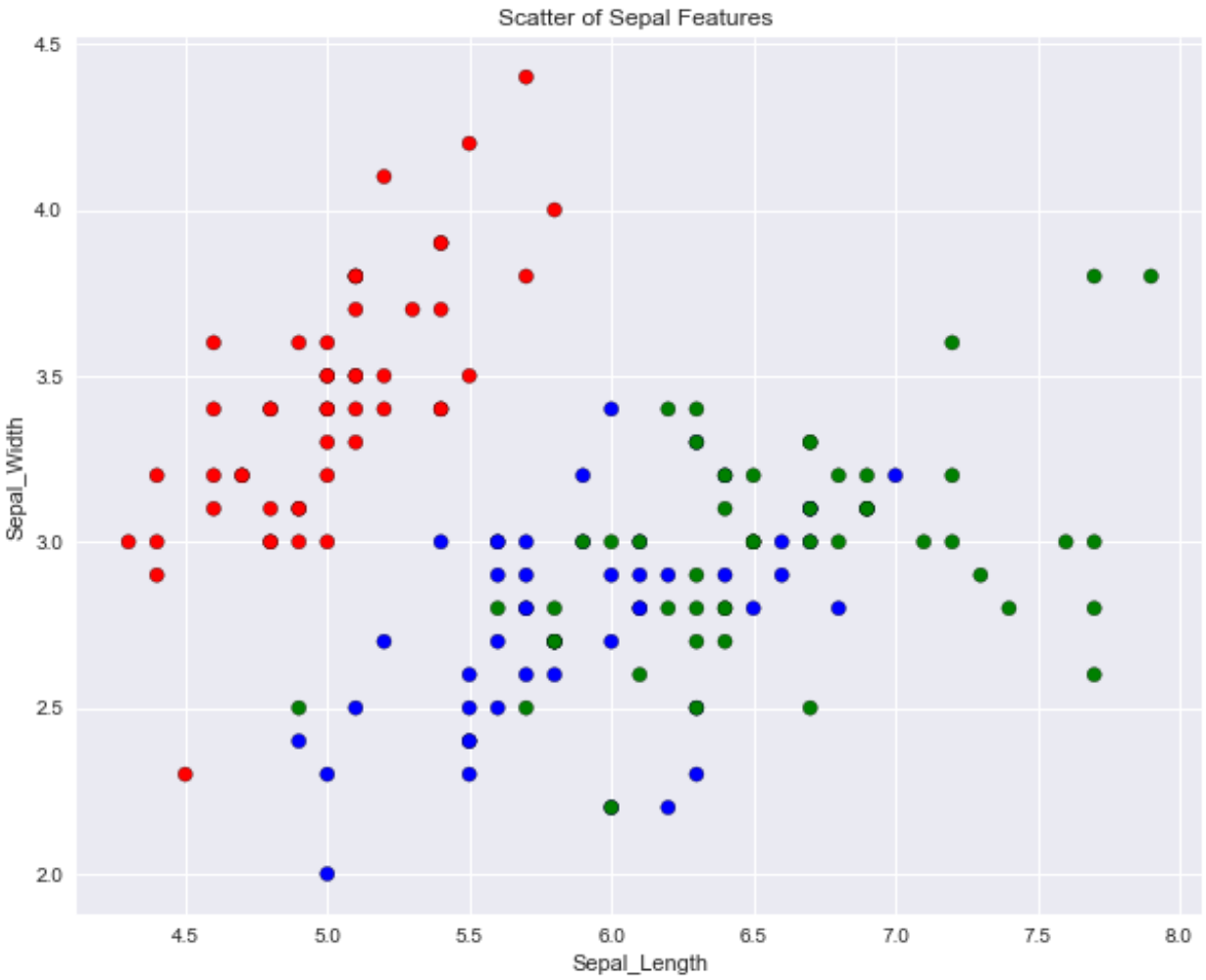
Predicting IRIS flower species with K-Means clustering. The data can be loaded by using the following command in Python

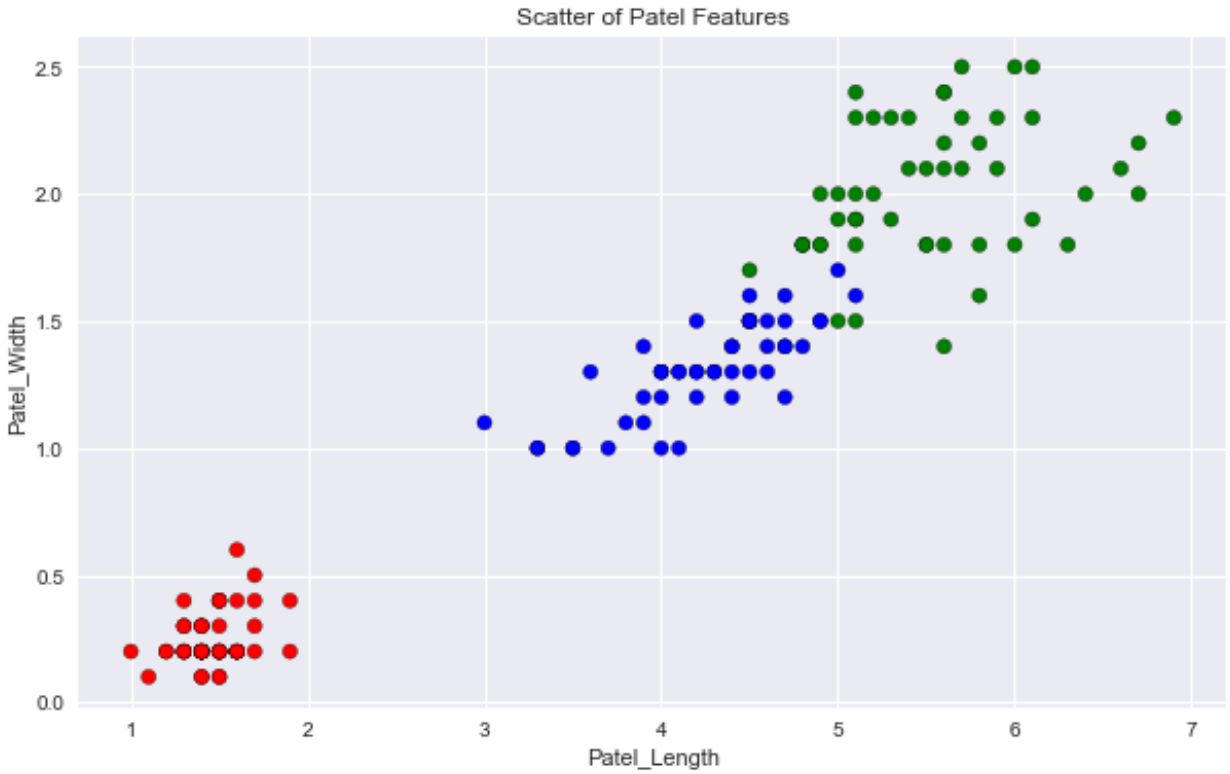
from sklearn.datasets import load_iris

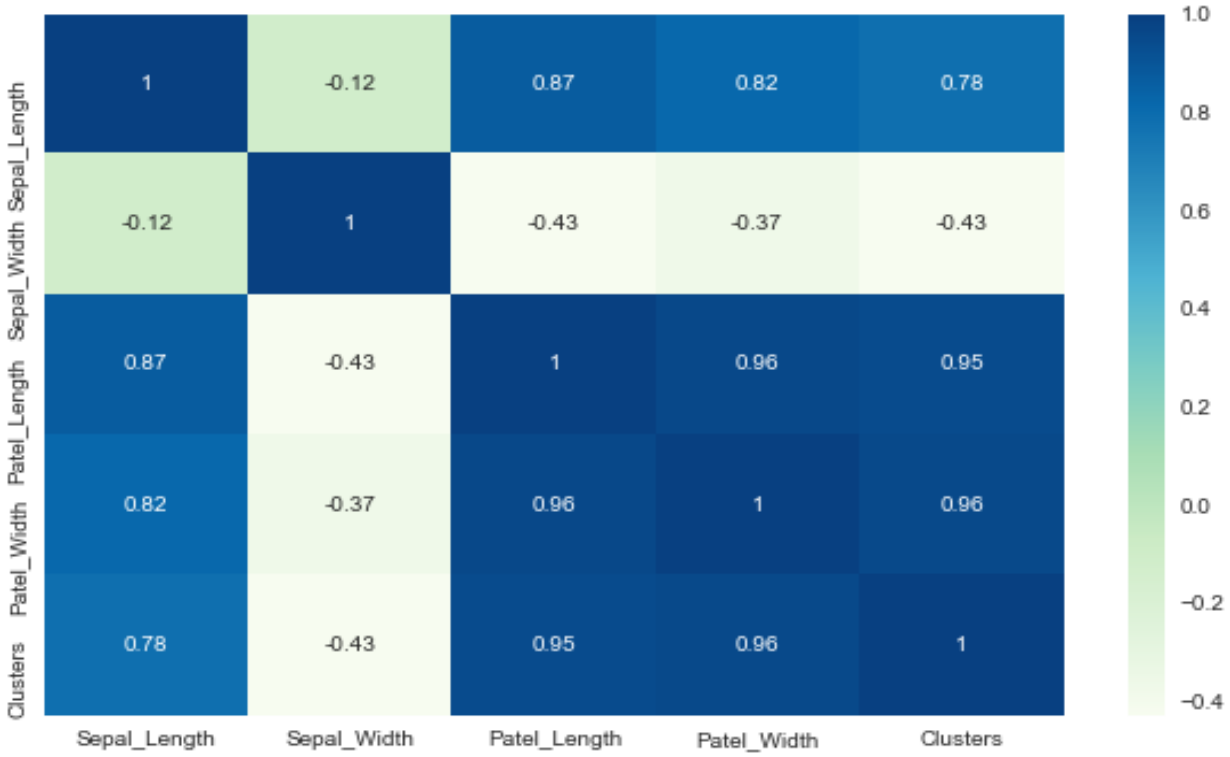
1- Visualise the data set by generating scatter plots of the data features.

Answer:

Visualize features of Sepal_Length with Sepal_Width and Patel_Length with Patel_width using scatter plot.



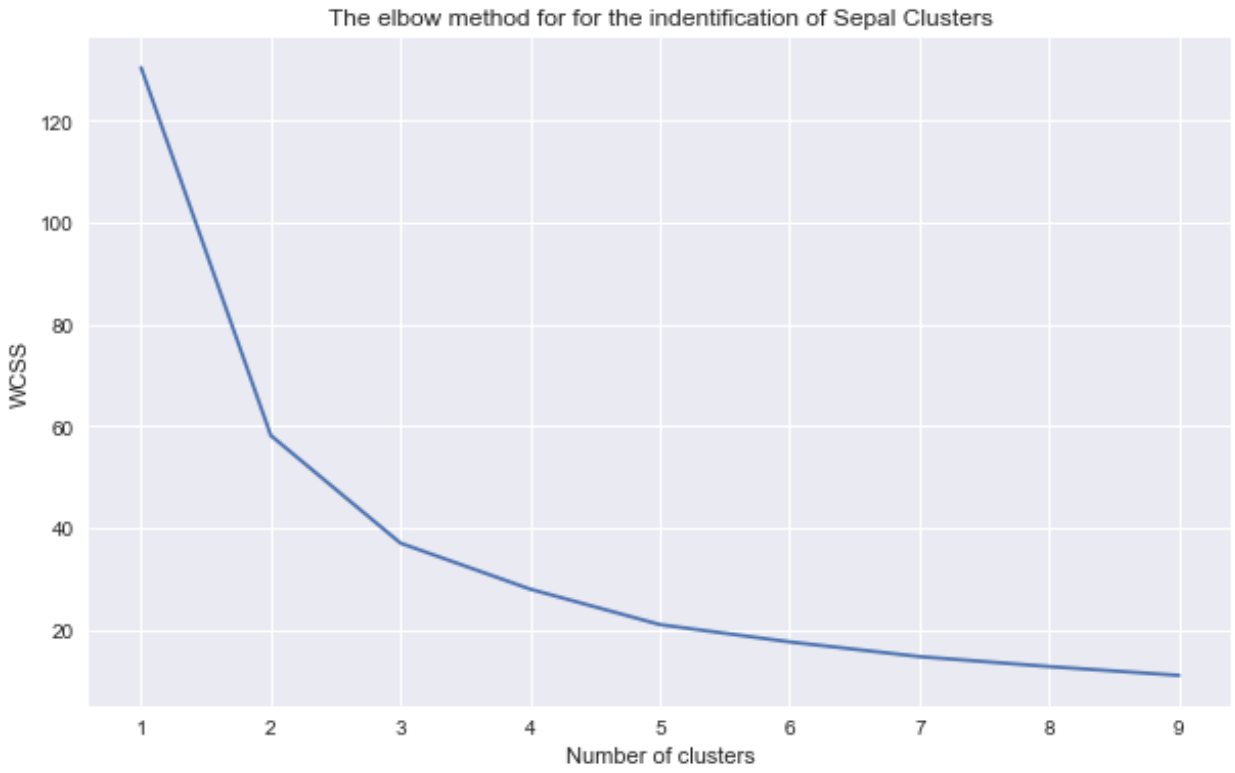




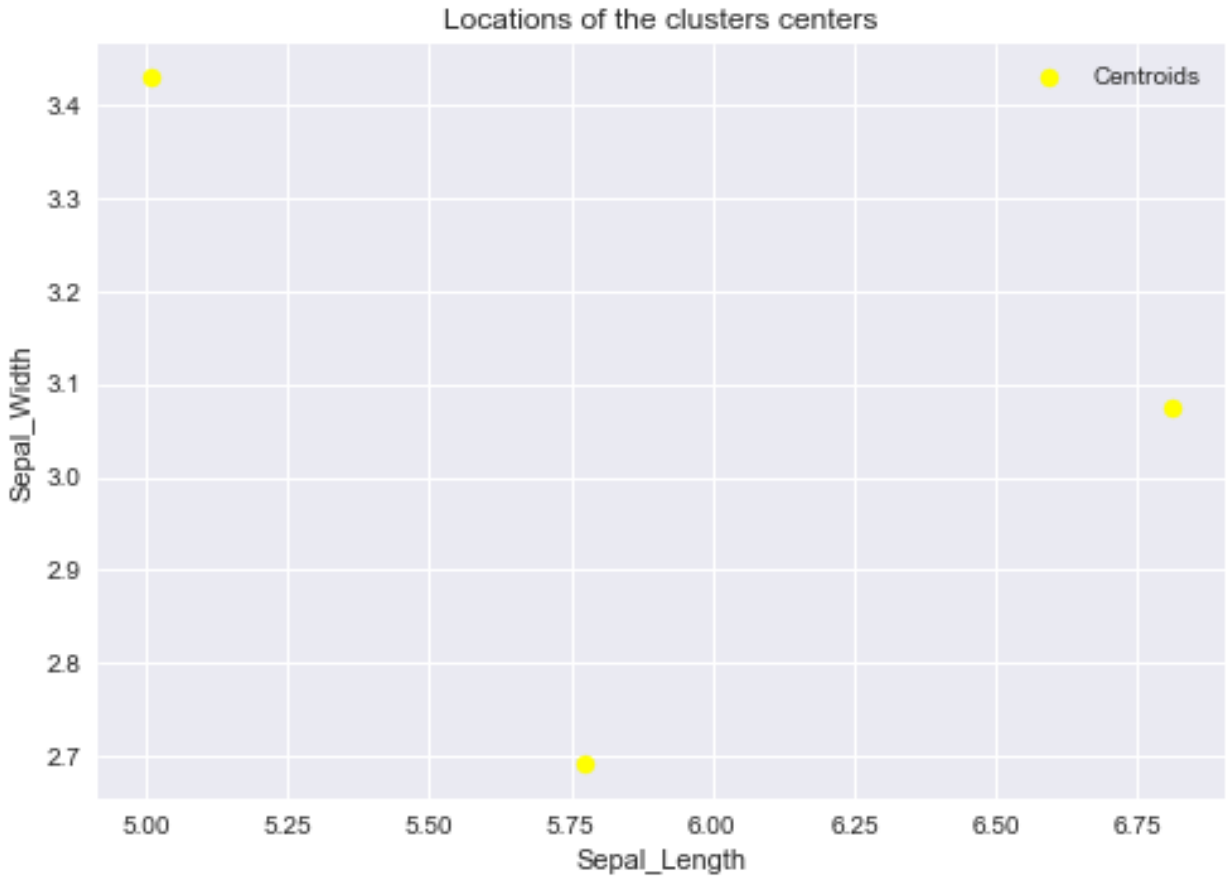
Perform correlations between features columns and Clusters

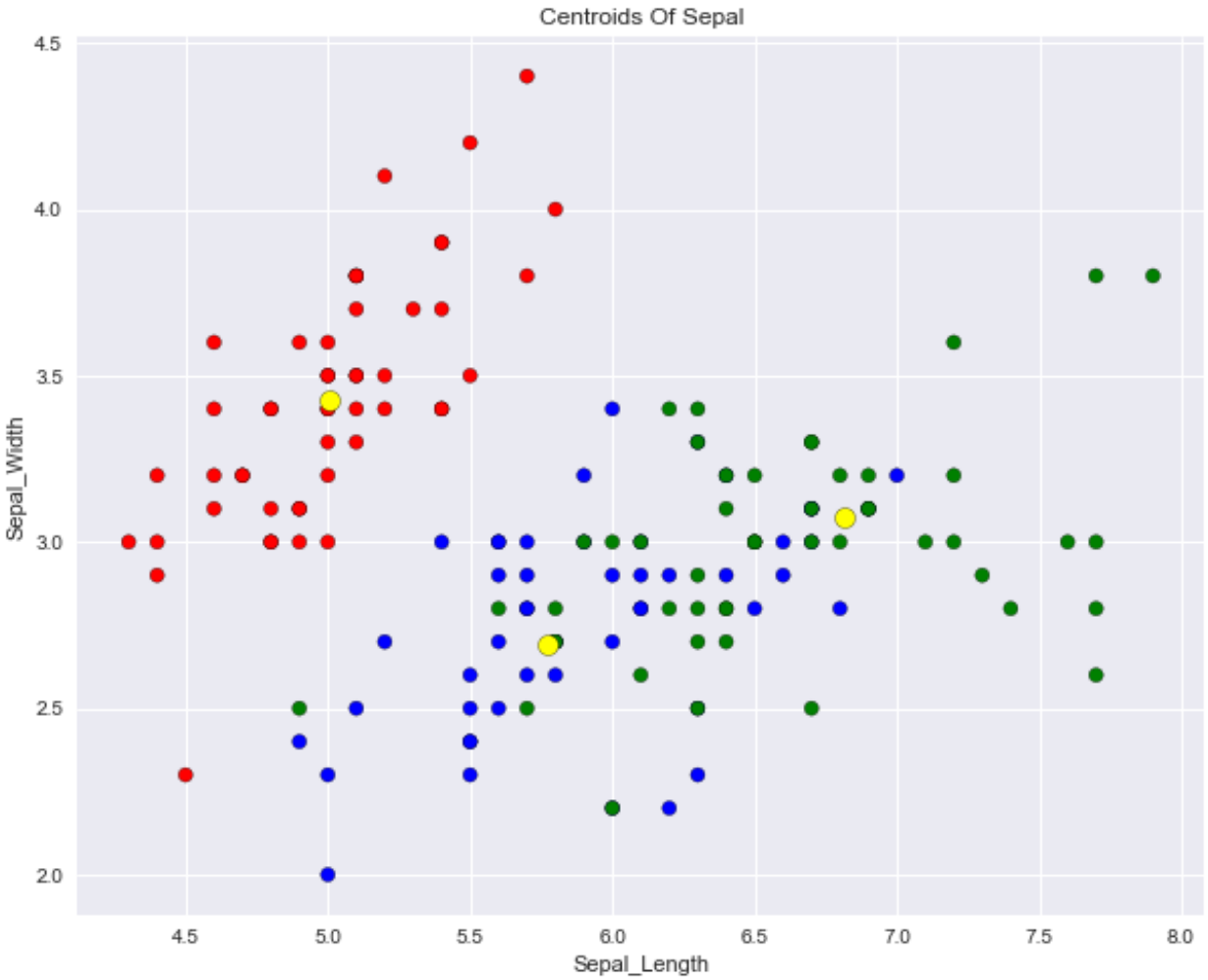
2- Based on the sepal length and sepal width,

• Find the optimal number of clusters using the elbow method.



Using K-Means clustering, identify the optimal locations of the clusters centers.





Evaluate the accuracy of this unsupervised clustering.

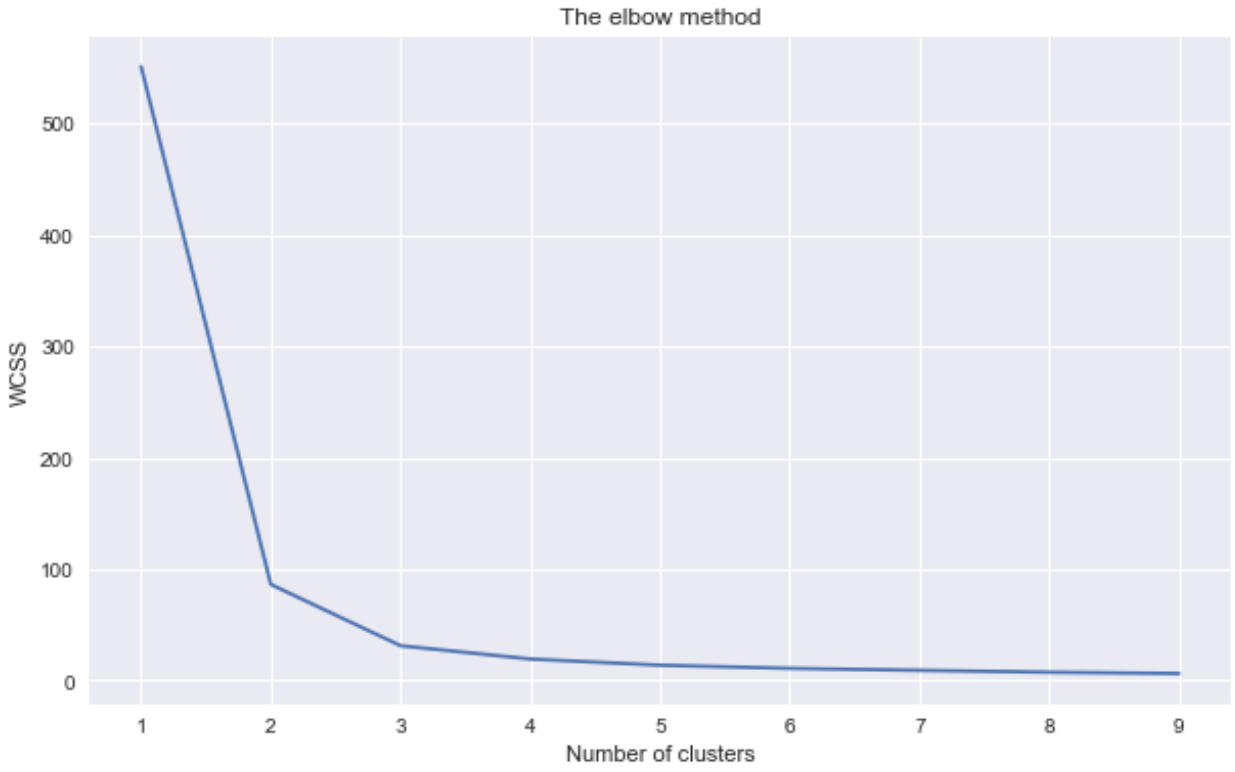
In [129]:
```
from sklearn.metrics import adjusted_rand_score

score = adjusted_rand_score(Array_Cluster,s_kmeans)
score
```

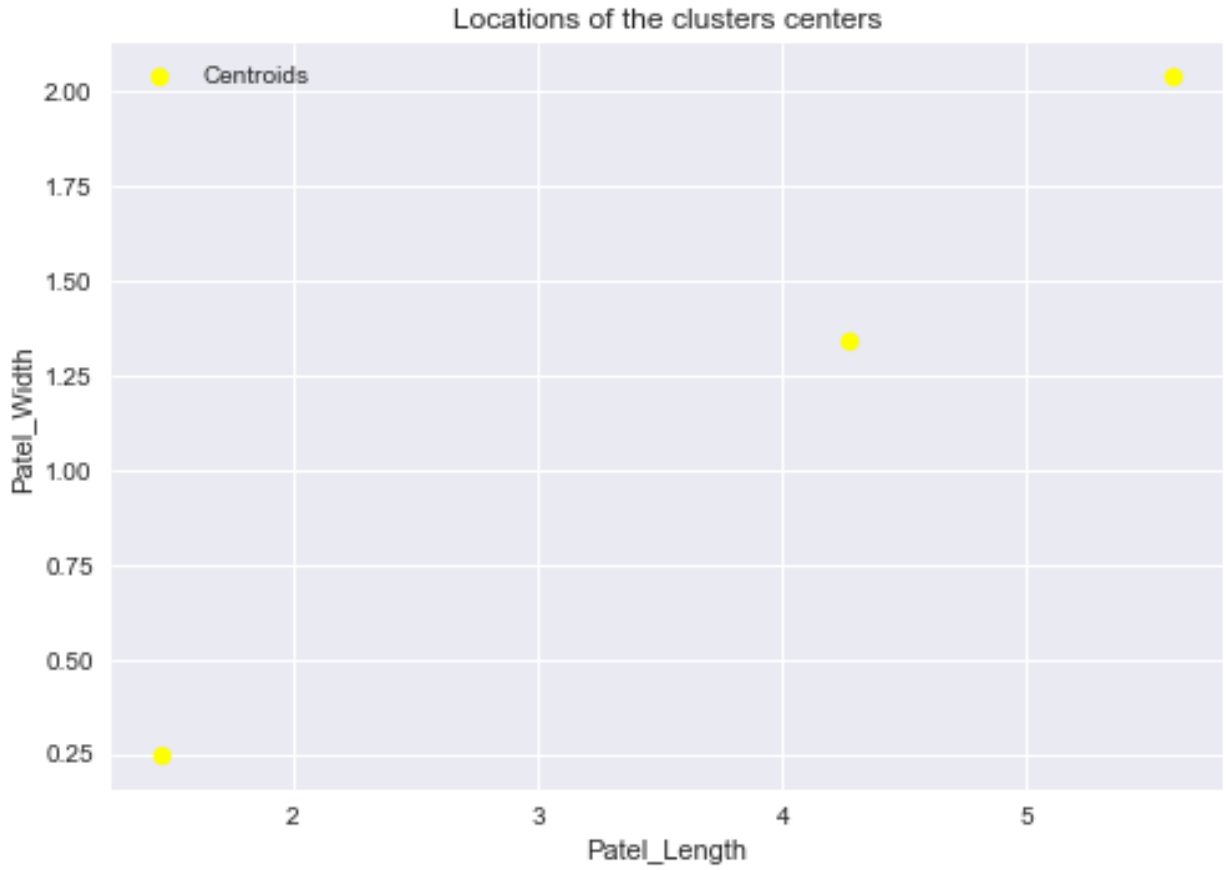
Out[129]: 0.6006861021484542

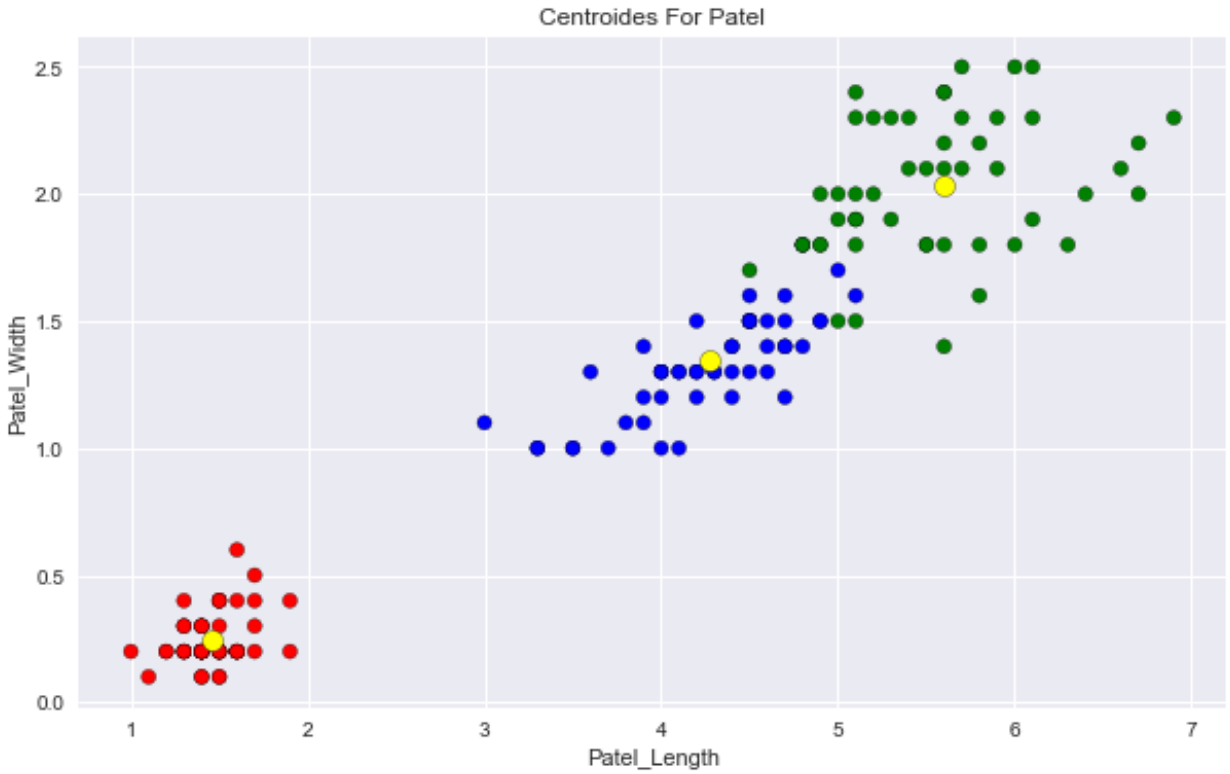
3- Based on the petal length and petal width,

• find the optimal number of clusters using the elbow method.



Using K-Means clustering, identify the optimal locations of the clusters centers.





Evaluate the accuracy of this unsupervised clustering.

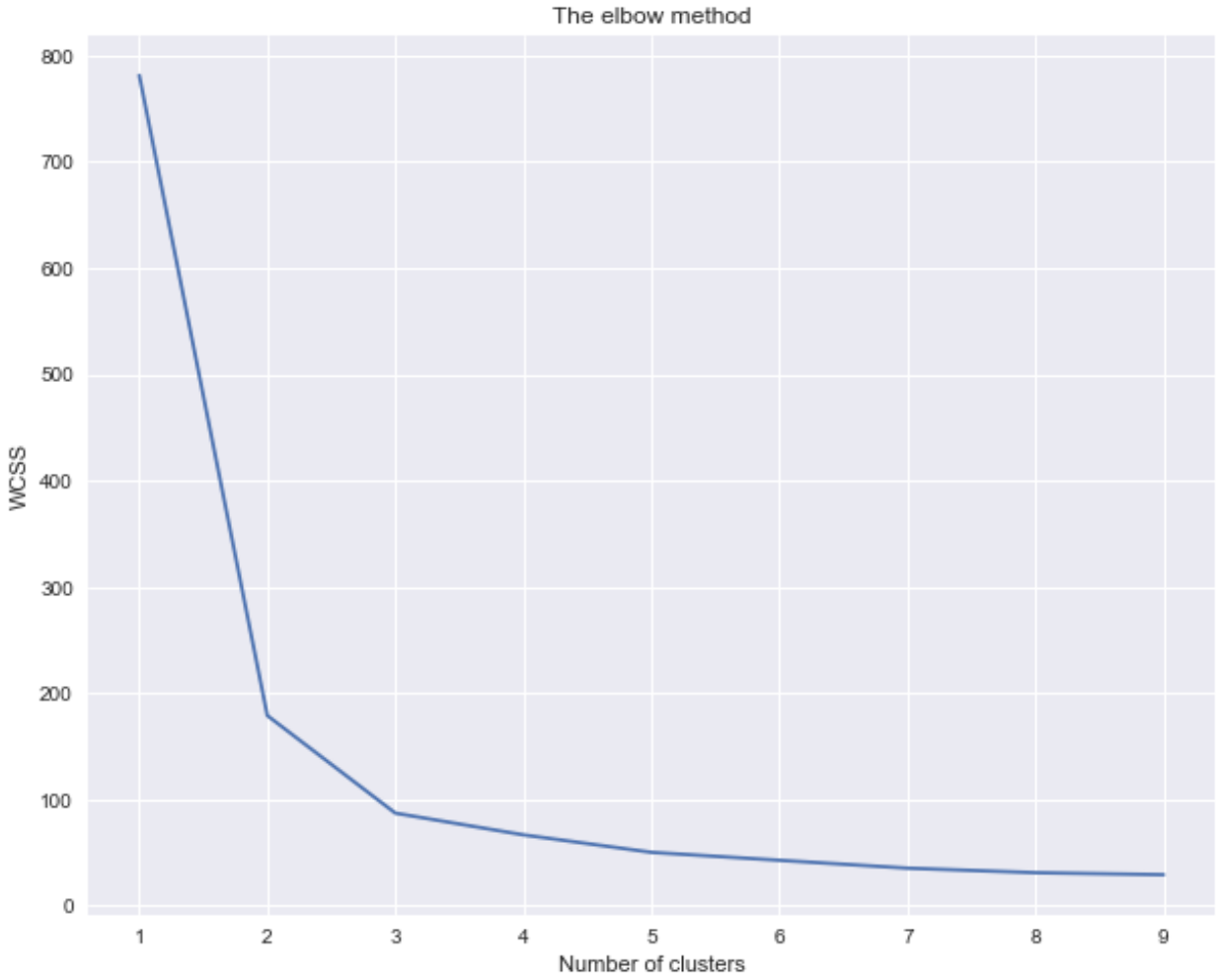
In [136]:
```
from sklearn.metrics import adjusted_rand_score

score = adjusted_rand_score(Array_Cluster,p_kmeans)
score
```

Out[136]: 0.8856970310281228

4- Based on all the features (sepal length, sepal width, petal length, and petal width)

• Find the optimal number of clusters using the elbow method.



Using K-Means clustering, identify the optimal locations of the clusters centers.

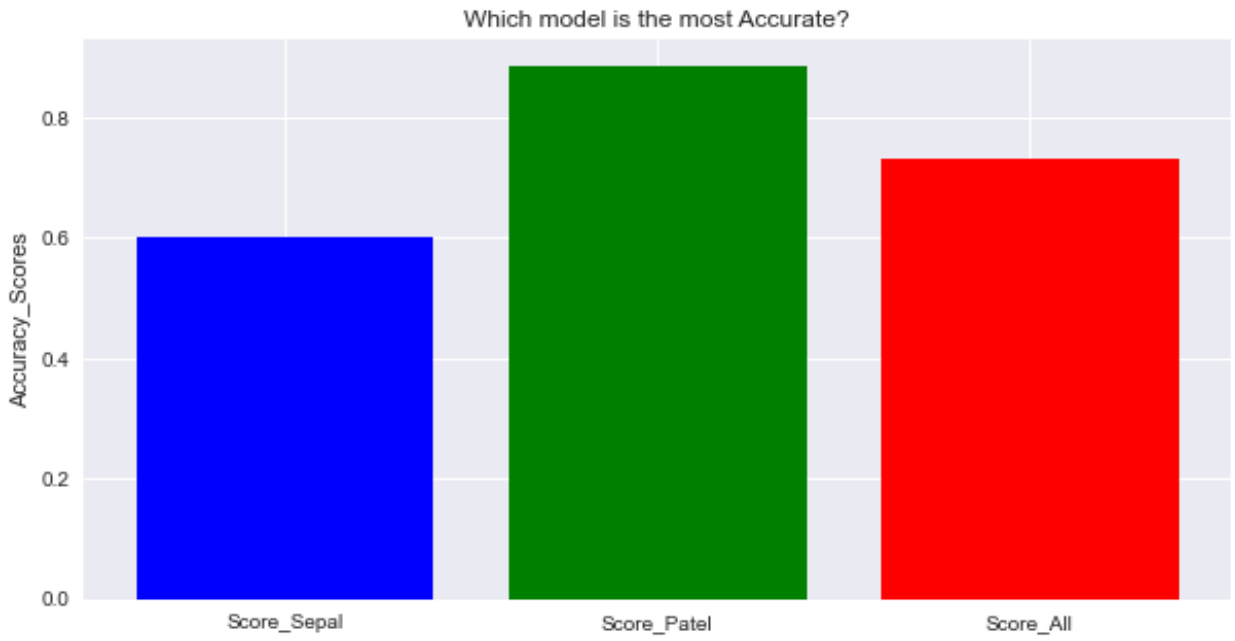
Evaluate the accuracy of this unsupervised clustering.

```
In [141]: from sklearn.metrics import adjusted_rand_score

          score = adjusted_rand_score(Array_Cluster,All_kmeans)
          score

Out[141]: 0.7302382722834697
```

5- Discuss the results obtained from clustering based on the different features in part 2-4.



K-neartest Neighbor

This assignment numerically explores the behavior of the K-nearest neighbor classification algorithm. Assume a two-class problem that is characterized by two features vector x = (x1, x2). The two class-conditional densities, p(x/c1) and p(x|c2) are taken to be two-dimensional Gaussian distributions centered at the points (4, 11) and (10, 3) with covariance's matrices given by $\Sigma = 0.3\ I$ and $\Sigma = I$ (with I being the identity matrix) respectively. In addition, the priors of the data are p(c1) = 0.5 and p(c2) = 0.5. For this problem,

1- Generate data from the two classes specified by the parameters given above.

In [2]:
```
# create two centers
GD_Center_1 = [4,11]
Cov_Matrix_1 = [[0.3, 0], [0, 0.3]]
```

In [3]:
```
GD_Center_2 = [10,3]
Cov_Matrix_2 = [[1, 0], [0, 1]]
```

In [4]:
```
Parameter_1 = np.random.multivariate_normal(GD_Center_1,Cov_Matrix_1,100)
Parameter_2 = np.random.multivariate_normal(GD_Center_2,Cov_Matrix_2,100)
```

In [13]:
```
Dataset = np.concatenate((Parameter_1, Parameter_2), axis = 1)
DataFrame = pd.DataFrame(Dataset)
DataFrame
```

Out[13]:

	0	1	2	3
0	4.682927	11.239517	10.590180	2.287531
1	3.307256	10.239526	11.710047	2.316788
2	3.788919	10.660812	9.287124	2.686872
3	3.980410	11.139398	10.013086	3.616051
4	3.680894	12.797928	8.576772	3.451054

2- Plot the data points using a different symbol for each of the two classes.

3- Derive an equation for the decision boundary along which the posterior probabilities for the two classes are equal.

Density Estimation:-

Probability density functions, says that the probability P that a new vector x will fall inside some region R of x-space is given that it is drawn from the unknown density function p(x) is,

$$P = \int_R p(x')\,dx'$$

where,

$$\text{Prob}(x=k) = \frac{N!}{k!(N-k)!} p^k (1-p)^{N-k}$$

mean fraction of points is shown as,

$$E\left(x = \frac{k}{N}\right) = P$$

and variance (σ)

$$\text{var}\left(x = \frac{k}{N}\right) = \frac{P(1-P)}{N} \quad \text{where } P \approx \frac{k}{N}$$

If $p(x)$ is assumed to be continous constant over R region

$$\Rightarrow P = \int_R p(x')\,dx' \approx p(x)V.$$

$$\Rightarrow p(x) \approx \frac{k}{NV}.$$

Bayes' Theorem:-

Bayes' theorem combines the prior probabilities with the class conditional densities - to give the posterior probabilities of a given class,

$\therefore P(C_K/x)$ as follows,

$$\therefore P(C_K|x) = \frac{P(x/C_K)\,P(C_K)}{P(x)}.$$

where $p(x)$ is unconditional density function,

$$\therefore p(x) = \sum_{K=1}^{c} P(x/C_K)P(C_K).$$

$\therefore$ c is the number of class,

$$\therefore \sum_{K=1}^{c} P(C_K/x) = 1.$$

where $P(x/C_K)$ is referred to as the likelihood.

Now considered,

$$\frac{P(C_1/x)}{P(C_2/x)} \underset{C_2}{\overset{C_1}{\gtrless}} 1 \Rightarrow \frac{P(x/C_1)}{P(x/C_2)} \underset{C_2}{\overset{C_1}{\gtrless}} \frac{P(C_2)}{P(C_1)}.$$

Taking log on both sides,

$$\log(p(x/C_1)) - \log(p(x/C_2)) = \log(P(C_2)) - \log(P(C_1)).$$

By substituting the values in above.

$p(x|C_1) = N(x:\mu_1, \Sigma_1)$, $p(x|C_2) = N(x:\mu_2, \Sigma_2)$

Yields the following quadratic equations,

$$-\frac{1}{2}(x-\mu_1)^T \Sigma_1^{-1}(x-\mu_1) + \frac{1}{2}(x-\mu_2)^T \Sigma_2^{-1}(x-\mu_2)$$

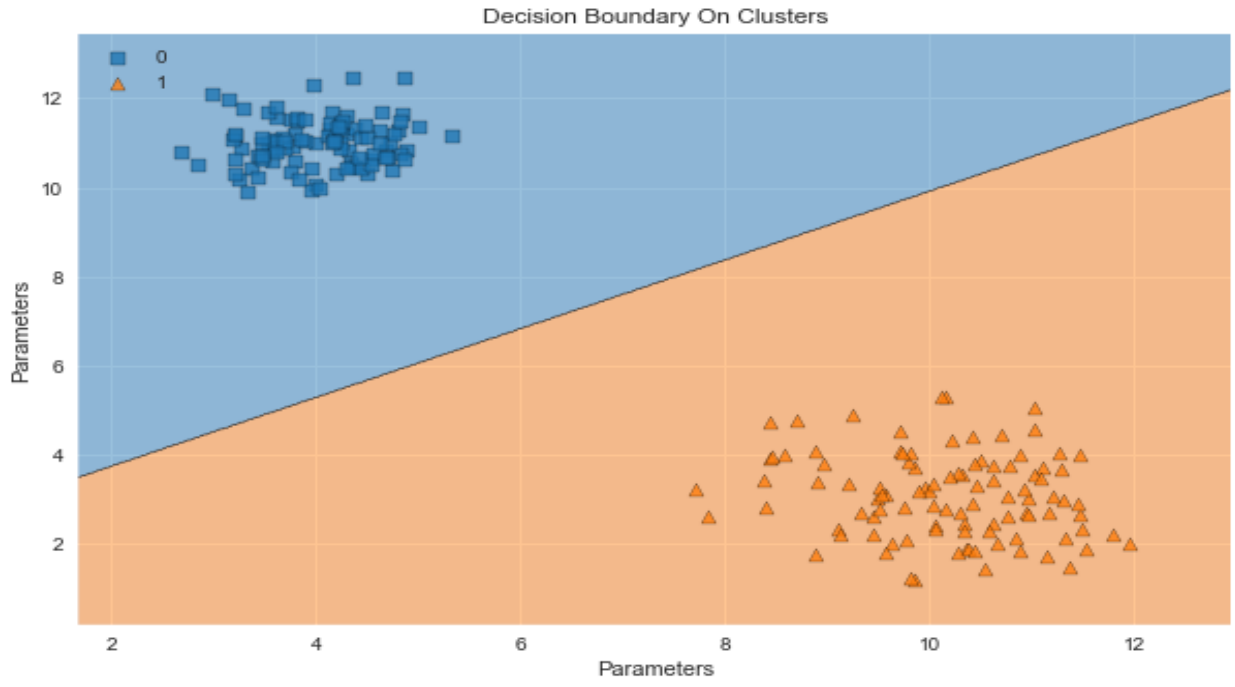
$$+\frac{1}{2}\log\left(\frac{|\Sigma_2|}{|\Sigma_1|}\right) = \log\left(\frac{P(C_2)}{P(C_1)}\right)$$

$$\sim -(x-\mu_1)^T \Sigma_1^{-1}(x-\mu_1) + (x-\mu_2)^T \Sigma_2^{-1}(x-\mu_2) + c'$$

where,

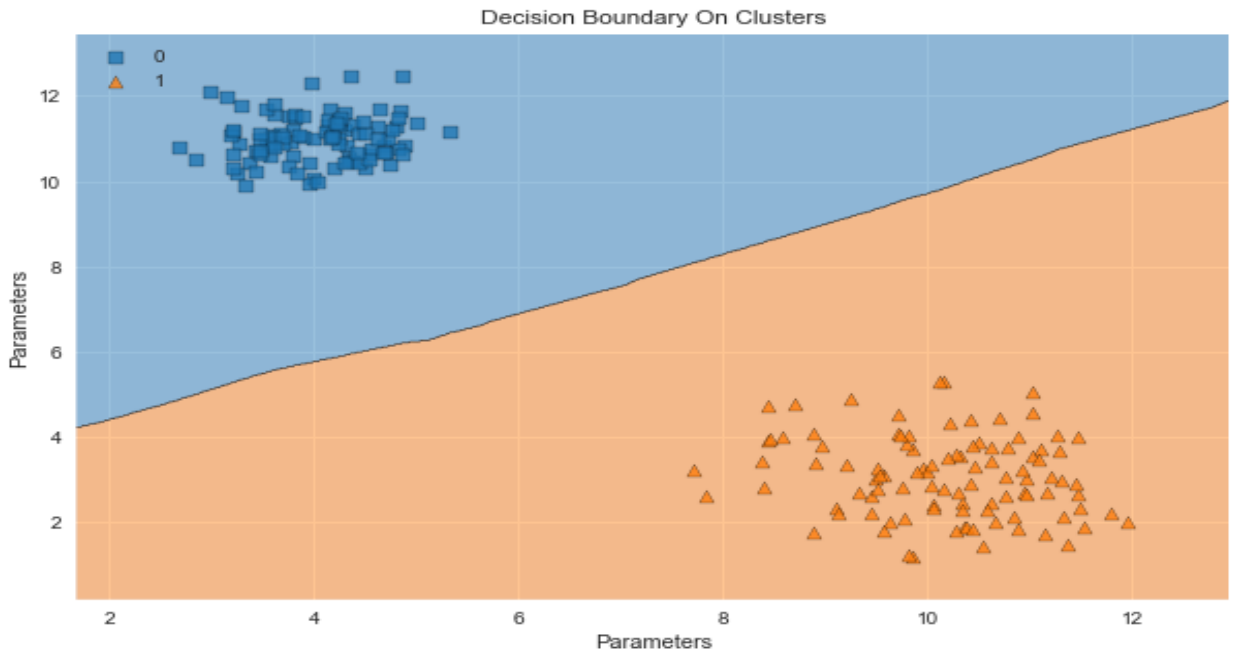
$$c' = \ln\frac{P(C_1)}{P(C_2)} - \frac{1}{2}\log|\Sigma_1| + \frac{1}{2}\log|\Sigma_2|.$$

4- Plot the decision boundary you obtained in part (3).

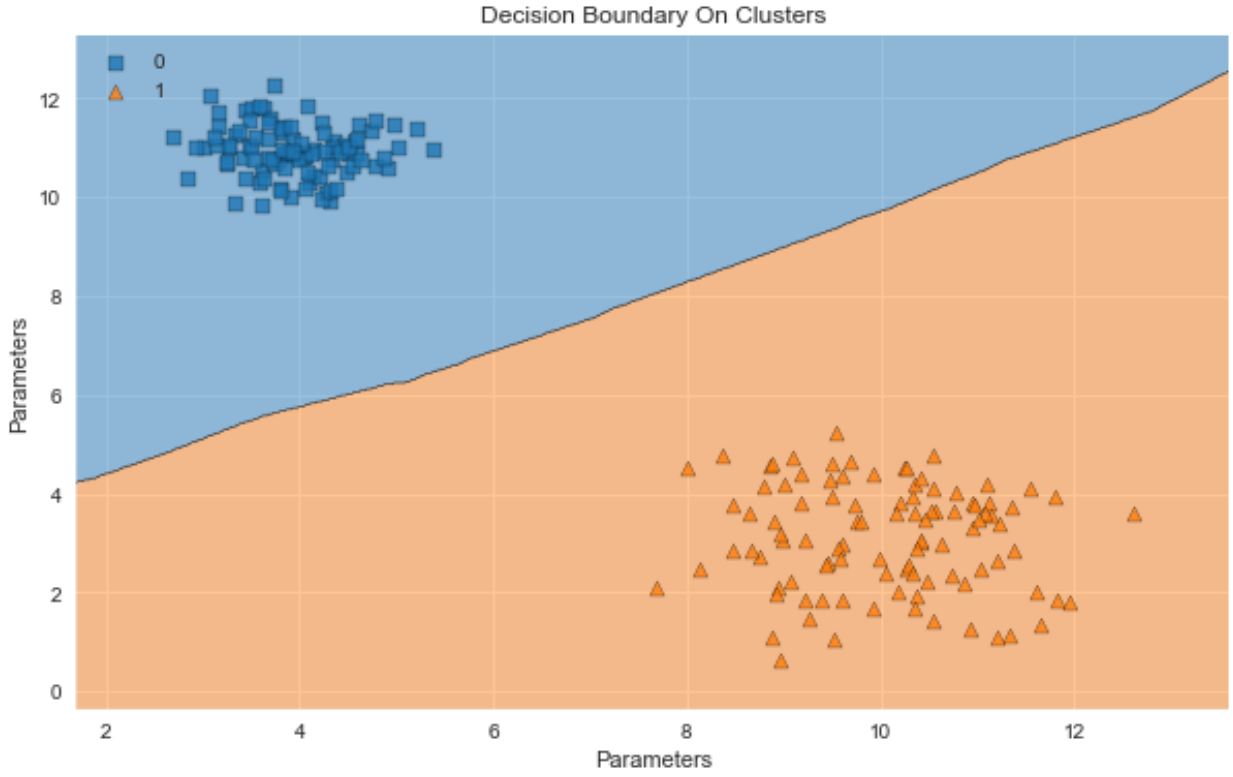


5- Plot the decision boundaries predicted by the KNN classification algorithm for various values of K

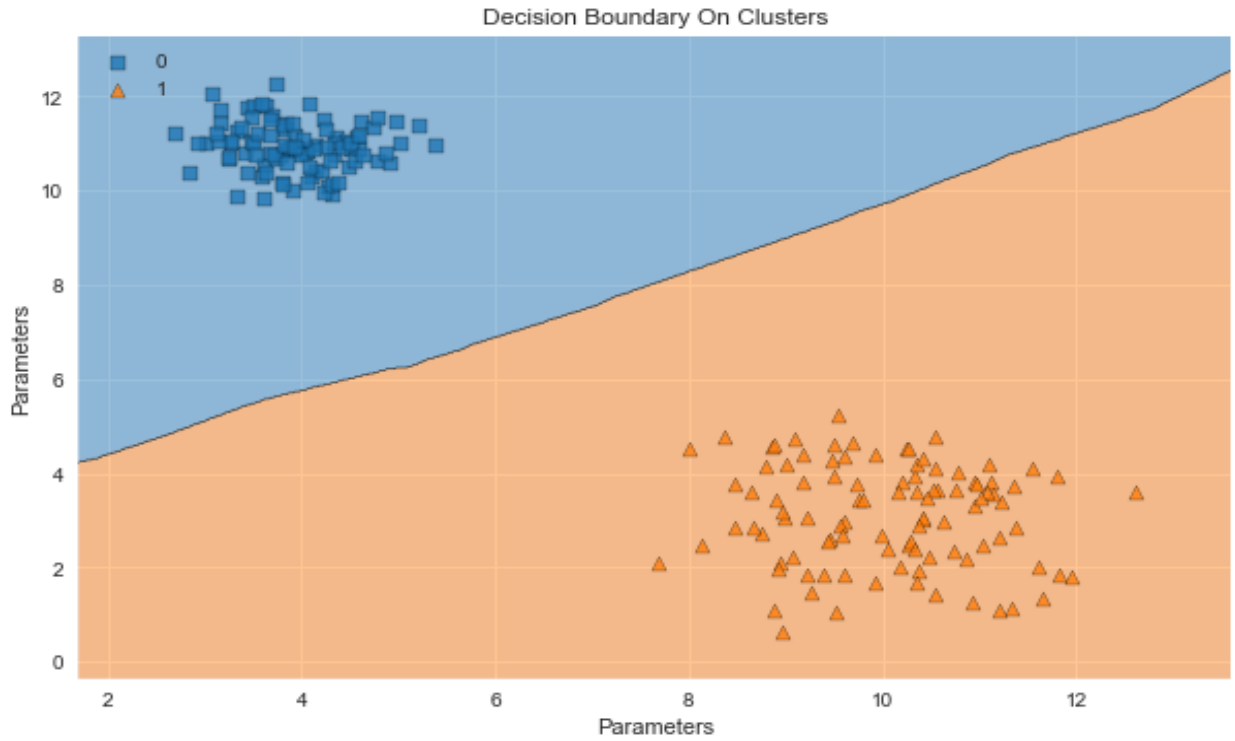
For k = 2:



For k = 3:



For k = 4:



Low values for K (eg. K = 1 or K = 2) can be noisy and subject to the effects of outliers. A small value for K provides the most flexible fit, which will have low bias but high variance. Large values for K on the other hand smooth things over. This means lower variance but increased bias. However, K should not be too large so that a category with only a few samples in it will not always be out voted by other categories.