Name: Shahnwaz Khan Roll: AC-1272 Subject: Data Mining Semester: VI Practical: Q5

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
# Import Gaussian Naive Bayes
from sklearn.naive_bayes import GaussianNB
# For K Nearest
from sklearn.neighbors import KNeighborsClassifier
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
# metrics to calculate the accuracy of classifiers
from sklearn import metrics
# seaborn to load the iris data set
import seaborn as sns
```

In [ ]:
```
df=sns.load_dataset('iris')
display(df.head())
```

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

Choosing Training set using HOLD OUT method

In [ ]:
```
# store the feature matrix (X) and response vector (y)
X = df.drop(columns='species')
y = df['species']
# splitting X and y into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Naive Bayes Classifier

In [ ]:
```
# Naive Bayes Classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)
# making predictions on the testing set
y_pred = gnb.predict(X_test)
```

In [ ]:
```
Naive_Accuracy= metrics.accuracy_score(y_test, y_pred)*100
```

KNN

In [ ]:

```
# Create an instance of KNN class. Neighbors size=5 and p=2 means use Euclidean d
classifier=KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
# Train our data by fitting the data into our model
classifier.fit(X_train,y_train)
# Now predict the output
y_pred=classifier.predict(X_test)
```

In [ ]:

```
KNN_Accuracy= metrics.accuracy_score(y_test, y_pred)*100
```

Decision Tree

In [ ]:

```
# Creating instance of Decision Tree Classifier
DC=DecisionTreeClassifier()
# Fit the model
DC.fit(X_train,y_train)
# Predict the result
y_pred=DC.predict(X_test)
```

In [ ]:

```
Decision_Accuracy= metrics.accuracy_score(y_test, y_pred)*100
```

In [ ]:

```
print('Naive Bayes Accuracy: ',Naive_Accuracy)
print('KNN accuracy: ',KNN_Accuracy)
print('Decision Tree Accuracy: ',Decision_Accuracy)
```

```
Naive Bayes Accuracy:  93.33333333333333
KNN accuracy:  93.33333333333333
Decision Tree Accuracy:  93.33333333333333
```

Comparing Different Classifiers using Bar Plot

In [ ]:

```
df=pd.DataFrame([[Naive_Accuracy,KNN_Accuracy,Decision_Accuracy]],columns=['Naive
df
```

Out[ ]:

	Naive	KNN	Decision
classifier	93.333333	93.333333	93.333333

In [ ]:

```
df.plot.bar()
```

Out[ ]: <AxesSubplot:>

(ii) Choose Training set using Random Sub Sampling method

Random subsampling

In [ ]:
```
k=3  # Count of random selection of samples
Decision_Accuracy=0  # Accuracy of decision tree
KNN_Accuracy=0 # Accuracy of KNN
Naive_Accuracy=0 # Accuracy of Naive Bayes
for i in range(k):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Fit the decision tree model
    DC.fit(X_train,y_train)
    # Fit KNN Model
    classifier.fit(X_train,y_train)
    # Fit Naive Bayes Model
    gnb.fit(X_train,y_train)
# Predict the result using each classifier
    y_pred_dc=DC.predict(X_test)
    y_pred_knn=classifier.predict(X_test)
    y_pred_naive=gnb.predict(X_test)

    # Now find the accuracy of each classifier
    #Decision tree
    acc_temp_DC= metrics.accuracy_score(y_test, y_pred_dc)*100
    Decision_Accuracy+=acc_temp_DC/k

    # KNN
    acc_temp_KNN= metrics.accuracy_score(y_test, y_pred_knn)*100
    KNN_Accuracy+=acc_temp_KNN/k

    # Naive Bayes
    acc_temp_Naive= metrics.accuracy_score(y_test, y_pred_naive)*100
    Naive_Accuracy+=acc_temp_Naive/k

print('Naive Bayes Accuracy: ',Naive_Accuracy)
print('KNN accuracy: ',KNN_Accuracy)
print('Decision Tree Accuracy: ',Decision_Accuracy)
```

```
Naive Bayes Accuracy:  95.55555555555556
KNN accuracy:  94.44444444444444
Decision Tree Accuracy:  93.33333333333333
```

Compare different classifier using Bar Plot

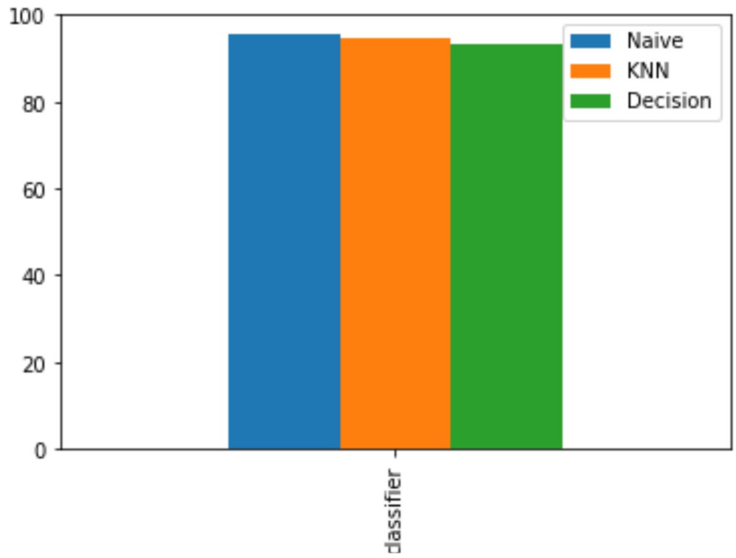
In [ ]:
```
df=pd.DataFrame([[Naive_Accuracy,KNN_Accuracy,Decision_Accuracy]],columns=['Naive
df
```

Out[ ]:

	Naive	KNN	Decision
classifier	95.555556	94.444444	93.333333

In [ ]:
```
df.plot.bar()
```

Out[ ]: <AxesSubplot:>



5.3 Data is scaled to standard format.

In [ ]:

```
from sklearn.preprocessing import StandardScaler
```

In [ ]:

```
object= StandardScaler()
# standardization
scale = object.fit_transform(X_train)
print(scale)
```

```
[[-0.97926616 -1.8617121  -0.216762   -0.22108395]
 [ 1.62684539  0.26595887  1.28069832  0.81931112]
 [-1.45310462  0.73877464 -1.27053038 -1.13142965]
 [ 1.03454731 -0.2068569   0.72608339  0.68926174]
 [-0.50542769  1.92081407 -1.1041459  -1.00138026]
 [-1.69002385 -0.44326479 -1.27053038 -1.26147903]
 [-0.50542769 -0.2068569   0.44877592  0.42916297]
 [-0.97926616  0.97518253 -1.32599188 -1.13142965]
 [ 0.91608769 -0.2068569   0.39331443  0.29911359]
 [-0.86080654  0.50236676 -1.1041459  -0.87133088]
 [-0.74234693  0.73877464 -1.27053038 -1.26147903]
 [-1.09772577  1.21159041 -1.27053038 -1.39152842]
 [ 1.03454731  0.02955099  0.55969891  0.42916297]
 [ 0.44224923 -0.44326479  0.33785293  0.1690642 ]
 [-0.38696808 -1.15248844  0.39331443  0.03901482]
 [-0.03158923 -0.91608056  0.78154488  0.94936051]
 [-0.15004885 -0.67967267  0.22692994  0.1690642 ]
 [-1.45310462  1.21159041 -1.49237636 -1.26147903]
 [ 1.15300693 -0.2068569   1.00339086  1.20945928]
 [-0.97926616  1.21159041 -1.27053038 -1.26147903]
 [ 1.03454731  0.50236676  1.11431384  1.72965682]
 [-1.21618539  0.73877464 -0.99322292 -1.26147903]
 [ 0.67916846  0.02955099  1.00339086  0.81931112]
 [-0.50542769  0.73877464 -1.21506889 -1.00138026]
 [ 0.32378962 -0.67967267  0.55969891  0.03901482]
 [ 2.21914347 -1.15248844  1.77985177  1.46955805]
 [-0.97926616  0.97518253 -1.1596074  -0.74128149]
 [-0.74234693  0.97518253 -1.21506889 -1.26147903]
 [ 1.27146654  0.02955099  0.94792936  1.20945928]
 [ 1.15300693 -0.67967267  0.6151604   0.29911359]
 [-1.21618539 -0.2068569  -1.27053038 -1.13142965]
 [-0.50542769  1.92081407 -1.32599188 -1.00138026]
 [-0.97926616  0.26595887 -1.38145337 -1.26147903]
 [ 0.20533     0.73877464  0.44877592  0.55921235]
 [-0.86080654  1.68440618 -1.21506889 -1.13142965]
 [-0.38696808 -1.62530421  0.00508397 -0.22108395]
```

```
[ 0.56070885 -1.8617121   0.39331443  0.1690642 ]
[-0.97926616 -0.2068569  -1.1596074  -1.26147903]
[-1.334645    0.26595887 -1.1596074  -1.26147903]
[-1.57156424 -1.8617121  -1.32599188 -1.13142965]
[ 1.62684539 -0.2068569   1.16977534  0.55921235]
[-1.45310462  0.02955099 -1.21506889 -1.26147903]
[ 0.67916846 -0.67967267  1.05885235  1.33950866]
[-0.50542769  1.4479983  -1.21506889 -1.26147903]
[ 0.32378962 -1.15248844  1.05885235  0.29911359]
[ 0.79762808 -0.2068569   1.16977534  1.33950866]
[ 0.67916846 -0.44326479  0.33785293  0.1690642 ]
[ 2.10068385 -0.2068569   1.61346729  1.20945928]
[-0.97926616  0.50236676 -1.27053038 -1.26147903]
[-0.50542769  0.73877464 -1.1041459  -1.26147903]
[-1.09772577  0.02955099 -1.21506889 -1.26147903]
[-1.45310462  0.26595887 -1.27053038 -1.26147903]
[-1.21618539  0.02955099 -1.1596074  -1.26147903]
[-0.15004885 -1.38889633  0.72608339  1.07940989]
[-0.15004885 -0.2068569   0.28239144  0.03901482]
[ 0.08687038  0.26595887  0.6151604   0.81931112]
[ 1.38992616  0.26595887  0.55969891  0.29911359]
[ 0.67916846  0.26595887  0.89246787  1.46955805]
[-0.15004885  3.1028535  -1.21506889 -1.00138026]
[ 0.20533    -0.2068569   0.6151604   0.81931112]
[-1.09772577 -0.2068569  -1.27053038 -1.26147903]
[ 0.91608769 -0.44326479  0.50423741  0.1690642 ]
[ 0.20533    -2.09811998  0.17146845 -0.22108395]
[ 2.21914347 -0.67967267  1.66892878  1.07940989]
[-0.26850846 -0.2068569   0.44877592  0.42916297]
[ 1.03454731 -0.2068569   0.83700638  1.46955805]
[-0.86080654  1.68440618 -1.1596074  -1.26147903]
[ 1.03454731  0.50236676  1.11431384  1.20945928]
[-0.62388731  1.4479983  -1.21506889 -1.26147903]
[-1.09772577 -1.38889633  0.44877592  0.68926174]
[ 0.44224923  0.73877464  0.94792936  1.46955805]
[-0.38696808  2.63003773 -1.27053038 -1.26147903]
[-0.15004885  1.68440618 -1.1041459  -1.13142965]
[ 0.67916846  0.26595887  0.44877592  0.42916297]
[-0.86080654  1.68440618 -0.99322292 -1.00138026]
[ 0.79762808  0.26595887  0.78154488  1.07940989]
[ 2.21914347  1.68440618  1.66892878  1.33950866]
[ 0.08687038 -0.2068569   0.78154488  0.81931112]
[ 0.79762808 -0.2068569   1.00339086  0.81931112]
[ 0.56070885  0.50236676  1.28069832  1.72965682]
[-0.97926616  0.73877464 -1.1596074  -1.00138026]
[ 1.03454731  0.02955099  0.39331443  0.29911359]
[-0.15004885 -0.44326479  0.28239144  0.1690642 ]
[-1.69002385  0.26595887 -1.32599188 -1.26147903]
[ 0.32378962 -0.67967267  0.17146845  0.1690642 ]
[ 0.67916846 -0.67967267  1.05885235  1.20945928]
[ 1.62684539  1.21159041  1.33615982  1.72965682]
[ 0.44224923 -0.67967267  0.6151604   0.81931112]
[-1.09772577 -1.62530421 -0.216762   -0.22108395]
[-0.74234693 -0.91608056  0.11600696  0.29911359]
[ 0.56070885 -1.38889633  0.72608339  0.94936051]
[ 0.32378962 -0.2068569   0.67062189  0.81931112]
[-0.26850846 -0.2068569   0.22692994  0.1690642 ]
[-0.86080654  0.97518253 -1.27053038 -1.26147903]
[ 2.21914347 -0.2068569   1.33615982  1.46955805]
[-1.80848347 -0.2068569  -1.43691487 -1.39152842]
[ 1.03454731 -1.38889633  1.16977534  0.81931112]
[-0.26850846 -0.67967267  0.67062189  1.07940989]
[-0.15004885 -0.67967267  0.44877592  0.1690642 ]
[ 1.15300693  0.26595887  1.22523683  1.46955805]
[-0.86080654  0.73877464 -1.21506889 -1.26147903]
[-1.69002385 -0.2068569  -1.32599188 -1.26147903]
```

```
 [ 0.79762808 -0.2068569   0.83700638  1.07940989]
 [-0.03158923 -0.91608056  0.78154488  0.94936051]
 [-0.03158923 -0.91608056  0.22692994 -0.22108395]
 [-0.38696808 -1.62530421  0.06054546 -0.09103457]
 [-0.38696808  0.97518253 -1.32599188 -1.26147903]
 [-0.15004885 -1.15248844 -0.10583902 -0.22108395]
 [ 1.27146654  0.26595887  1.11431384  1.46955805]
 [-0.26850846 -0.44326479 -0.05037752  0.1690642 ]
 [ 1.27146654  0.02955099  0.67062189  0.42916297]
 [ 0.56070885 -0.44326479  1.05885235  0.81931112]
 [-0.03158923 -0.91608056  0.11600696  0.03901482]
 [ 1.50838577 -0.2068569   1.22523683  1.20945928]
 [-1.334645    0.26595887 -1.32599188 -1.26147903]
 [-0.03158923 -1.15248844  0.17146845  0.03901482]
 [ 1.745305   -0.44326479  1.44708281  0.81931112]
 [-0.03158923  2.15722196 -1.38145337 -1.26147903]
 [-0.97926616 -2.57093576 -0.10583902 -0.22108395]
```

In [ ]: