In [ ]:

```
'''1. Write a program to implement the naïve Bayesian classifier for a
sample training data set stored as a .CSV file.
Compute the accuracy of the classifier, considering few test data sets.'''
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
diabetes=pd.read_csv("diabetes.csv")
#only selecting Outcome column from the above table
y=diabetes["Outcome"]
#Deleting outcome column from the table
x=diabetes.drop("Outcome",axis=1)
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.25)
gnb=GaussianNB()
gnb.fit(X_train,Y_train)
prediction=gnb.predict(X_test)
accuracy=accuracy_score(Y_test,prediction)
accuracy
```

In [ ]:

```
'''2. Write a program to demonstrate the working of the decision tree algorithm.
Use an appropriate data set for building the decision tree and apply this knowledge
to classify a new sample.'''
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
iris=load_iris()
x=pd.DataFrame(iris.data)
y=pd.DataFrame(iris.target)
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.25)
dtc=DecisionTreeClassifier()
dtc.fit(X_train,Y_train)
prediction=dtc.predict(X_test)
accuracy=accuracy_score(Y_test,prediction)
accuracy
```

In [ ]:

```
'''3. Write a program to implement k-Nearest Neighbour algorithm to
classify the iris data set.
Print both correct and wrong predictions.'''
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
iris=load_iris()
x=pd.DataFrame(iris.data)
y=pd.DataFrame(iris.target)
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.25)
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,Y_train)
Y_pred=knn.predict(X_test)
accuracy=accuracy_score(Y_test,Y_pred)
accuracy
```

In [ ]:

```
'''4. Write a program to implement Support Vector Machine algorithm
to classify the iris data set.Print both correct and wrong predictions.'''
import pandas as pd
import numpy as numpy
from sklearn.datasets import load_iris
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
iris=load_iris()
x=pd.DataFrame(iris.data)
y=pd.DataFrame(iris.target)
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.25)
svm=SVC()
svm.fit(X_train,Y_train)
prediction=svm.predict(X_test)
accuracy=accuracy_score(Y_test,prediction)
accuracy
```

In [ ]:

```
'''5.Apply EM algorithm to cluster a set of data stored in a .CSV file.
Use the same data set for clustering using k-Means algorithm.
Compare the results of these two algorithms and comment on the quality
of clustering.'''
import pandas as pd
import numpy as np
from sklearn.mixture import GaussianMixture
from sklearn.metrics import accuracy_score
from scipy.stats import mode
diabetes=pd.read_csv("diabetes.csv")
y=diabetes['Outcome']
x=diabetes.drop("Outcome", axis=1)
gmm = GaussianMixture(n_components=3)
gmm.fit(x)
clusters=gmm.fit_predict(x)
labels = np.zeros_like(clusters)
for i in range(4):
    cat = (clusters == i)
    labels[cat] = mode(y[cat])[0]
accurracy=accuracy_score(y,labels)
accurracy
```

In [ ]:

```
 '''K_MEANS'''
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
from scipy.stats import mode
diabetes=pd.read_csv("diabetes.csv")
y=diabetes['Outcome']
x=diabetes.drop("Outcome", axis=1)
k = KMeans(n_clusters=3)
k.fit(x)
clusters=k.fit_predict(x)
labels = np.zeros_like(clusters)
for i in range(4):
    cat = (clusters == i)
    labels[cat] = mode(y[cat])[0]
accurracy=accuracy_score(y,labels)
accurracy
```

In [ ]:

```
'''6. Apply Hierarchical Clustering algorithm to cluster a set
```

```
of data stored in a .CSV file.Use the same data set for clustering
using k-Means algorithm. Compare the results of these two algorithms
and comment on the quality of clustering.'''
import pandas as pd
import numpy as np
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import accuracy_score
from scipy.stats import mode
diabetes=pd.read_csv("diabetes.csv")
y=diabetes['Outcome']
x=diabetes.drop("Outcome", axis=1)
hir = AgglomerativeClustering(n_clusters=3)
hir.fit(x)
clusters=hir.fit_predict(x)
labels = np.zeros_like(clusters)
for i in range(4):
    cat = (clusters == i)
    labels[cat] = mode(y[cat])[0]
accurracy=accuracy_score(y,labels)
accurracy
```

In [ ]:

```
'''8. Build an Artificial Neural Network by implementing
the Backpropagation algorithm and test the same using
appropriate data sets.'''
import tensorflow as tf
from keras import Sequential
from keras.layers import Dense
from keras.models import save_model
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
iris=load_iris()
x=iris.data
y=iris.target
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size = 0.25)
ann = Sequential()
ann.add(Dense(units = 128, activation = "sigmoid"))
ann.add(Dense(units = 64 , activation = "sigmoid"))
ann.add(Dense(units = 1,   activation = "softmax"))
ann.compile(optimizer = "adam", loss = "binary_crossentropy", metrics = ["accuracy"]
ann.fit(X_train, Y_train, batch_size = 30, epochs = 500)
prediction = ann.predict(X_test)
accuracy=accuracy_score(Y_test, prediction)
accuracy
```

In [ ]:

```
'''9. Write a program to implement AdaBoost algorithm to
classify the iris data set.
Print both correct and wrong predictions.'''
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
iris=load_iris()
x=pd.DataFrame(iris.data)
y=pd.DataFrame(iris.target)
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.25)
abc=AdaBoostClassifier()
abc.fit(X_train,Y_train)
prediction=abc.predict(X_test)
```

```
accuracy=accuracy_score(Y_test,prediction)
accuracy
```

In [ ]:

```
'''10. Perform model aggregation on MNIST digit dataset.'''
import keras
from keras.datasets import mnist
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
(x_train,y_train),(x_test,y_test)=mnist.load_data()
x_train.shape
x=x_train.reshape(60000,28*28)
X_train,X_test,Y_train,Y_test=train_test_split(x,y_train,test_size=0.25)
svm=SVC()
svm.fit(X_train,Y_train)
prediction=svm.predict(X_test)
accuracy=accuracy_score(Y_test,prediction)
accuracy
```