In [ ]:
```
#1)a.Implementation of Find-S algorithm

import pandas as pd
import numpy as np

d = pd.read_csv("dataset.csv")

print(d)

a = np.array(d)[:,:-1]
print(" The attributes are: ",a)

t = np.array(d)[:,-1]
print("The target is: ",t)

def fun(c,t):
    for i, val in enumerate(t):
        if val == "Yes":
            specific_hypothesis = c[i].copy()
            break

    for i, val in enumerate(c):
        if t[i] == "Yes":
            for x in range(len(specific_hypothesis)):
                if val[x] != specific_hypothesis[x]:
                    specific_hypothesis[x] = '?'
                else:
                    pass
        return specific_hypothesis
print(" The final hypothesis is:",train(a,t))
```

In [ ]:
```
#Decison Tree Exercise-3
import numpy as mp
import pandas as pd
d=pd.read_csv(r"C:\Users\20B91A12J0\Downloads\archive\Iris.csv",header=0)

x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 1].values

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=42)

from sklearn.preprocessing import StandardScaler

s=StandardScaler()
X_train=s.fit_transform(X_train)
X_test=s.fit_transform(X_test)

from sklearn.tree import DecisionTreeClassifier

k=DecisionTreeClassifier()
k.fit(X_train,Y_train)
y_pred=k.predict(X_test)

from sklearn.metrics import accuracy_score

print(accuracy_score(Y_test,y_pred)*100)#
```

In [ ]:
```
#4)a.Implementation of Simple Linear Regression Algorithm using Python
import numpy as nm
import matplotlib.pyplot as mtp
```

```
import pandas as pd

data_set= pd.read_csv('Salary_Data.csv')

x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 1].values

# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 1/3, random_st

#Fitting the Simple Linear Regression model to the training dataset
from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(x_train, y_train)

#Prediction of Test and Training set result
y_pred= regressor.predict(x_test)
x_pred= regressor.predict(x_train)

mtp.scatter(x_train, y_train, color="green")
mtp.plot(x_train, x_pred, color="red")
mtp.title("Salary vs Experience (Training Dataset)")
mtp.xlabel("Years of Experience")
mtp.ylabel("Salary(In Rupees)")
mtp.show()
```

In [ ]:

```
#Logistic Regression Exercise-4b
import numpy as mp
import pandas as pd
d=pd.read_csv(r"C:\Users\20B91A12J0\Downloads\archive\Iris.csv",header=0)

x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 1].values

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=42)
from sklearn.preprocessing import StandardScaler
s=StandardScaler()
X_train=s.fit_transform(X_train)
X_test=s.fit_transform(X_test)
from sklearn.linear_model import LogisticRegression
l=LogisticRegression()
l.fit(X_train,Y_train)
y_pred=l.predict(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_test,y_pred)*100)
```

In [ ]:

```
#Binary Classifier Exercise-4c
import numpy as mp
import pandas as pd
d=pd.read_csv(r"C:\Users\20B91A12J0\Downloads\archive (1)\heart.csv",header=0)

x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 1].values

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=42)
from sklearn.preprocessing import StandardScaler
s=StandardScaler()
X_train=s.fit_transform(X_train)
X_test=s.fit_transform(X_test)
from sklearn.ensemble import RandomForestClassifier
```

```
r=RandomForestClassifier()
r.fit(X_train,Y_train)
y_pred=r.predict(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_test,y_pred)*100)
```

In [ ]:
```
# 5.estimate the bias and variance for a regression model
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from mlxtend.evaluate import bias_variance_decomp
# load dataset
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.csv'
dataframe = read_csv(url, header=None)
# separate into inputs and outputs
data = dataframe.values
X, y = data[:, :-1], data[:, -1]
# split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_s
# define the model
model = LinearRegression()
# estimate bias and variance
mse, bias, var = bias_variance_decomp(model, X_train, y_train, X_test, y_test, los
# summarize results
print('MSE: %.3f' % mse)
print('Bias: %.3f' % bias)
print('Variance: %.3f' % var)
```

In [ ]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# load the dataset
data = pd.read_csv(r"C:\Users\20B91A12J0\Downloads\archive\Iris.csv", header=0)

# encode the target variable
label_encoder = LabelEncoder()
label_ids = label_encoder.fit_transform(data['Species'])
onehot_encoder = OneHotEncoder(sparse=False)
reshaped = label_ids.reshape(len(label_ids), 1)
targetvar = onehot_encoder.fit_transform(reshaped)

# get the independent variables
inde_vars = []
for col in data.columns:
    if col not in ['Id', 'Species']:
        inde_vars.append(col)

# split the data into training and testing sets
X = data[inde_vars]
y = targetvar
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2, random_st

# normalize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)

# train the model using Naive Bayes algorithm
```

```
gnb = GaussianNB()
gnb.fit(X_train, Y_train)

# make predictions on test data
y_pred = gnb.predict(X_test)

# evaluate the model accuracy
accuracy = accuracy_score(Y_test, y_pred)
print("Accuracy:", accuracy * 100)
```

In [ ]:
```
#KNN Exercise-8
import numpy as mp
import pandas as pd
d=pd.read_csv(r"C:\Users\20B91A12J0\Downloads\archive\Iris.csv",header=0)

x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 1].values

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=42)
from sklearn.preprocessing import StandardScaler
s=StandardScaler()
X_train=s.fit_transform(X_train)
X_test=s.fit_transform(X_test)
from sklearn.neighbors import KNeighborsClassifier
k=KNeighborsClassifier(n_neighbors=4)
k.fit(X_train,Y_train)
y_pred=k.predict(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_test,y_pred)*100)
```

In [ ]:
```
#NaiveBayes Exercise-10
import numpy as mp
import pandas as pd
d=pd.read_csv(r"C:\Users\20B91A12J0\Downloads\archive\Iris.csv",header=0)

x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 1].values

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=42)
from sklearn.preprocessing import StandardScaler
s=StandardScaler()
X_train=s.fit_transform(X_train)
X_test=s.fit_transform(X_test)
from sklearn.naive_bayes import GaussianNB
g=GaussianNB()
g.fit(X_train,Y_train)
y_pred=g.predict(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_test,y_pred)*100)
```

In [ ]:
```
#11.Python Program to Implement the K-Means and Estimation & MAximization Algorithm

from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
import sklearn.metrics as metrics
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

names = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width', 'Class']
```

```
dataset = pd.read_csv("8-dataset.csv", names=names)

X = dataset.iloc[:, :-1]

label = {'Iris-setosa': 0,'Iris-versicolor': 1, 'Iris-virginica': 2}

y = [label[c] for c in dataset.iloc[:, -1]]

plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])

# REAL PLOT
plt.subplot(1,3,1)
plt.title('Real')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y])

# K-PLOT
model=KMeans(n_clusters=3, random_state=0).fit(X)
plt.subplot(1,3,2)
plt.title('KMeans')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[model.labels_])

print('The accuracy score of K-Mean: ',metrics.accuracy_score(y, model.labels_))
print('The Confusion matrixof K-Mean:\n',metrics.confusion_matrix(y, model.labels_

# GMM PLOT
gmm=GaussianMixture(n_components=3, random_state=0).fit(X)
y_cluster_gmm=gmm.predict(X)
plt.subplot(1,3,3)
plt.title('GMM Classification')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm])

print('The accuracy score of EM: ',metrics.accuracy_score(y, y_cluster_gmm))
print('The Confusion matrix of EM:\n ',metrics.confusion_matrix(y, y_cluster_gmm))
```

In [ ]:

```
#NaiveBayes Exercise-13
import numpy as mp
import pandas as pd
d=pd.read_csv(r"C:\Users\20B91A12J0\Downloads\archive (1)\heart.csv",header=0)

x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 1].values

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=42)
from sklearn.preprocessing import StandardScaler
s=StandardScaler()
X_train=s.fit_transform(X_train)
X_test=s.fit_transform(X_test)
from sklearn.naive_bayes import GaussianNB
g=GaussianNB()
g.fit(X_train,Y_train)
y_pred=g.predict(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_test,y_pred)*100)
```

In [ ]:

```
#PCA and SVM Exercise-14
import numpy as mp
import pandas as pd
d=pd.read_csv(r"C:\Users\20B91A12J0\Downloads\archive\Iris.csv",header=0)

x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 1].values
```

```
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=42)
from sklearn.preprocessing import StandardScaler
s=StandardScaler()
X_train=s.fit_transform(X_train)
X_test=s.fit_transform(X_test)
from sklearn.decomposition import PCA
principal=PCA(n_components=3)
principal.fit(X_train)
principal.fit(X_test)
from sklearn.svm import SVC
s=SVC()
s.fit(X_train,Y_train)
y_pred=s.predict(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(Y_test,y_pred)*100)
```

In [ ]:

```
#PCA Exercise-15
import numpy as mp
import pandas as pd
d=pd.read_csv(r"C:\Users\20B91A12J0\Downloads\archive\Iris.csv",header=0)

x= data_set.iloc[:, :-1].values
y= data_set.iloc[:, 1].values

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=42)
from sklearn.preprocessing import StandardScaler
s=StandardScaler()
X_train=s.fit_transform(X_train)
X_test=s.fit_transform(X_test)
from sklearn.tree import DecisionTreeClassifier
k=DecisionTreeClassifier()
k.fit(X_train,Y_train)
y_pred=k.predict(X_test)
from sklearn.metrics import accuracy_score
print("Before PCA")
print(accuracy_score(Y_test,y_pred)*100)
from sklearn.decomposition import PCA
principal=PCA(n_components=3)
X_Train1=principal.fit_transform(X_train)
X_Test1=principal.fit_transform(X_test)
k1=DecisionTreeClassifier()
k1.fit(X_Train1,Y_train)
Y_pred1=k1.predict(X_Test1)
print("After PCA")
print(accuracy_score(Y_test,Y_pred1)*100)
```

In [ ]: