## 1) Write a python program to demonstrate linear regression using an appropriate dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import linear_model
df=pd.read_csv("C:\\Users\\User\\Desktop\\Python&ML\\Module-
2\\pizza.csv")
df
plt.xlabel('size')
plt.ylabel('Price')
plt.scatter(df['size'], df['Price'], color='red', marker='+')
plt.plot(df['size'], df['Price'], color='blue')
plt.show()
reg=linear_model.LinearRegression()
reg.fit(df[['size']],df.Price)
reg.predict([[15]])
```

## 2) Write a python program to demonstrate logistic regression using an appropriate dataset.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
df=pd.read_csv("C:\\Users\\User\\Desktop\\Python&ML\\inputinsu
rance.csv")
plt.xlabel('age')
plt.ylabel('insurance')
plt.scatter(df.age,df.insurance,color='red',marker='+')
```

```
plt.show()

train_test_split(df[['age']],df.insurance)

x_train,x_test,y_train,y_test=train_test_split(df[['age']],df.
insurance, test_size=0.2)

model=LogisticRegression()

model.fit(x_train,y_train)

inputdata=np.array([45,23,56,66,77,88,99,12])

inputdata = inputdata.reshape(-1,1)

model.predict(inputdata)
```

**3) Write a python program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample sets . Print both correct and wrong predictions.**

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#%matplotlib inline

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier


data=pd.read_csv("C:\\Users\\User\\Desktop\\Python&ML\\iris.cs
v")

column_names = ['sepal_length', 'sepal_width','petal_length',
'petal_width', 'class']

data.columns = column_names

print(data.isnull().sum())

X = data.drop('class', axis=1)

y = data['class']

print(f"Features shape: {X.shape}")

print(f"Target shape: {y.shape}")
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=42)

print(f"Training set size: {X_train.shape[0]}")

print(f"Test set size: {X_test.shape[0]}")

model = DecisionTreeClassifier(criterion='entropy',random_state=42)

model.fit(X_train, y_train)

train_accuracy = model.score(X_train, y_train)

test_accuracy = model.score(X_test, y_test)

print(f"Test Accuracy: {test_accuracy:.4f}")

new_sample = [1.3, 1.3,4.5,8.0]

print(len(new_sample))

prediction = model.predict([new_sample])

print(f"Predicted Class: {prediction[0]}")
```

**4) Write a python program to implement Naive Bayes algorithm to classify the iris data set. Print both correct and wrong predictions.**

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.naive_bayes import GaussianNB

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import MinMaxScaler


data= pd.read_csv("C:\\Users\\User\\Desktop\\Python&ML\\iris.csv")

data.head()

x=data.drop(["variety"],axis=1)

y=data["variety"]

scalar = MinMaxScaler()

x_scaled=scalar.fit_transform(x)
```

```
X_train, X_test, y_train, y_test = train_test_split( x, y, test_size=0.2, random_state=42)

gnb=GaussianNB()

gnb.fit(X_train,y_train)

y_pred=gnb.predict( X_test)

y_pred
```

**5) Write a python program to implement k-Nearest Neighbor algorithm to classify the iris data set. Print both correct and wrong predictions.**

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import LabelEncoder

df=pd.read_csv("C:\\Users\\User\\Desktop\\Python&ML\\Practical Programs\\knndataset.csv")

df.head()

X = df[['weight','height']].values

y = df['class'].values


le = LabelEncoder()

y_encoded = le.fit_transform(y)

y_encoded

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X, y_encoded)

sample = [[23,56]]  # Example height and weight

prediction = knn.predict(sample)

predicted_label = le.inverse_transform(prediction)[0]

print(f"Predicted status for {sample[0]} (height, weight): {predicted_label}")
```

**6) Write a python program to implement clustering using the k-Means algorithm using an appropriate dataset**

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.datasets import make_blobs


df=pd.read_csv("C:\\Users\\User\\Desktop\\Python&ML\\Practical Programs\\kmeansdata.csv")

df

X=df[['Height','Weight']]

Kmean= KMeans(n_clusters=3,  random_state=30)

Kmean.fit(X)

KMeans(n_clusters=2, random_state=30)

X['cluster']=Kmean.fit_predict(X)

X
```

**7) Write a python program to implement Hierarchical clustering**

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from scipy.cluster.hierarchy import dendrogram,linkage

from sklearn.cluster import AgglomerativeClustering

from sklearn.decomposition import PCA

data
=pd.read_csv("C:\\Users\\User\\Desktop\\Python&ML\\Practical Programs\\HierarchicalData.csv")
```

```
x=data['X']

y=data['Y']

n=range(1,8)

fig,ax=plt.subplots()

ax.scatter(x,y,marker='*',c='red',alpha=0.5)

linked = linkage(data[['X','Y']],'single')

plt.figure(figsize=(10,7))

dendrogram(linked,
orientation='top',distance_sort='descending',show_leaf_counts=
True)

plt.xlabel('Samples')

plt.ylabel('Distances')

plt.show()

model = AgglomerativeClustering(n_clusters=5,
metric='euclidean', linkage='single')

model.fit(data[['X','Y']])

x=data['X']

y=data['Y']

n=range(1,8)

fig, ax = plt.subplots()

ax.scatter(x, y, c=model.labels_, cmap='rainbow')

plt.grid()

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Hierarchical Clustering')

plt.show()
```