# PRACTICAL 1

**Q.** Find the central tendency of the dataset by using statistics Measure(Mean, Medium,Mode and Range)

import pandas as pd

df = pd.read\_csv("healthcare-dataset-stroke-data.csv")

ages = df['age'].dropna()

print("mean:",ages.mean())

print("median",ages.median())

print("mode:",ages.mode().tolist())

# PRACTICAL NO 2

**Q.** Implementation of different visualization Techniques on the given dataset to know your characteristics of dataset Classification algorithm (Decision Tree/ Bayesian)

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv("Iris.csv")

*#histogram*

df["PetalWidthCm"].hist(bins = 20,edgecolor='black')

plt.title("Histogram of petal width")

plt.xlabel("Petal Width (cm)")

plt.ylabel("Frequency")

plt.show()

*#Scatter plot*

df.plot.scatter(x='SepalLengthCm', y='PetalLengthCm', title='Sepal Length vs Petal length')

plt.show()

*#pie chart*

df["Species"].value\_counts().plot.pie(autopct='%1.1f%%')

plt.title("Distribution of species")

plt.show()

*#box plot*

df['PetalLengthCm'].plot.box()

plt.title("Boxplot of petal Length")

plt.show()

*#bar plot*

df.groupby('Species')['PetalLengthCm'].mean().plot.bar()

plt.show()

*#line chart*

df.plot(x='SepalLengthCm', y='PetalLengthCm', kind='line',marker='o',title='sepal vs Length')

plt.show()

# PRACTICAL NO 3

**Q.** Implementation Attribute Relevance Analysis using ID3 algorithm.

Code:-

import numpy as np

from sklearn.tree import DecisionTreeClassifier

X = np.array([[2,8],[3,6],[5,7],[1,5],[6,8]])

y = np.array([0,0,1,0,1])

model = DecisionTreeClassifier(criterion='entropy').fit(X,y)

print(f"study importance:{model.feature\_importances\_[0]:.2f}")

print(f"Sleep importance:{model.feature\_importances\_[1]:.2f}")

pred = model.predict([[4,7]])[0]

print(f"Predicted:{'Pass' if pred ==1 else 'Fail'}")

import numpy as np

from sklearn.tree import DecisionTreeClassifier

# Tiny data: 5 students [study hours, sleep hours]

X = np.array([[2, 8], [3, 6], [5, 7], [1, 5], [6, 8]]) # Features

y = np.array([0, 0, 1, 0, 1]) # 0=Fail, 1=Pass

# Train (entropy = info gain)

model = DecisionTreeClassifier(criterion='entropy').fit(X, y)

# Importances (info gain for each feature)

print(f"Study importance: {model.feature\_importances\_[0]:.2f}")

print(f"Sleep importance: {model.feature\_importances\_[1]:.2f}")

# Predict for 4 study hours, 7 sleep

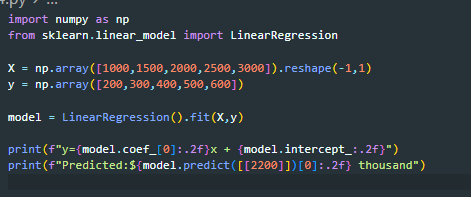
pred = model.predict([[4, 7]])[0]

print(f"Predicted: {'Pass' if pred==1 else 'Fail'}")

# PRACTICAL NO 4

**Q.** Implementation of Linear Regression (Single and

Multidimensional)



1. Simple Linear Regression

import numpy as np

from sklearn.linear\_model import LinearRegression

# Data: sizes (X), prices (y, $thousands)

X = np.array([1000, 1500, 2000, 2500, 3000]).reshape(-1, 1)

y = np.array([200, 300, 400, 500, 600])

# Train model

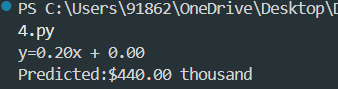
model = LinearRegression().fit(X, y)

# Equation

print(f"y = {model.coef\_[0]:.2f}x + {model.intercept\_:.2f}")

# Predict 2200 sq ft

print(f"Predicted: ${model.predict([[2200]])[0]:.2f} thousand")



1. **Multiple Linear Regression:**-

import numpy as np

from sklearn.linear\_model import LinearRegression

X = np.array([[1000,2],[1500,3],[2000,3],[2500,4],[3000,4]]) # [size, bedrooms]

y = np.array([200,300,350,500,550]) # prices ($thousands)

model = LinearRegression().fit(X, y)

coefs = model.coef\_

intercept = model.intercept\_

print(f"y = {intercept:.2f} + {coefs[0]:.2f}\*size + {coefs[1]:.2f}\*bedrooms")

print(f"Predicted (2200 sq ft, 3 beds): ${model.predict([[2200,3]])[0]:.2f} thousand")

# PRACTICAL NO 5

**Q.** Implementation of Naïve Bayesian classification algorithm Code:-

import numpy as np

from sklearn.naive\_bayes import GaussianNB

# Tiny data: 5 students [study hours, sleep hours]

X = np.array([[2, 8], [3, 6], [5, 7], [1, 5], [6, 8]]) # Features

y = np.array([0, 0, 1, 0, 1]) # 0=Fail, 1=Pass

# Train (assumes independence)

model = GaussianNB().fit(X, y)

# Predict for 4 study hours, 7 sleep (with probabilities)

pred = model.predict([[4, 7]])[0]

probs = model.predict\_proba([[4, 7]])[0]

print(f"Predicted: {'Pass' if pred==1 else 'Fail'}")

print(f"P(Pass): {probs[1]:.2f}, P(Fail): {probs[0]:.2f}")

# PRACTICAL NO 7

# import numpy as np

# import matplotlib.pyplot as plt

# from scipy.cluster.hierarchy import dendrogram, linkage

# from sklearn.datasets import load\_iris

# X=load\_iris().data

# Z=linkage(X,method='ward')

# plt.figure(figsize=(10,5))

# dendrogram(Z,truncate\_mode='level',p=3)

# plt.title('Hierarchical Clustering Dendrogram')

# plt.xlabel('Sample index or cluster size')

# plt.ylabel('Distance')

# plt.show()