## SLIP 1

**Q.1 Write a Python program to implement to find all null values in a given [10M]**

**Data set and remove them.(Use air quality dataset.)**

import pandas as pd

df = pd.read\_csv('AirQuality.csv')

print("Null values before removing:")

print(df.isnull().sum())

df\_cleaned = df.dropna()

print("\nNull values after removing:")

print(df\_cleaned.isnull().sum())

**Q.2 Write a python program to implement k-means algorithm on a synthetic [20M]**

**Dataset.**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

data, labels = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.60,

random\_state=42)

kmeans = KMeans(n\_clusters=4)

kmeans.fit(data)

centers = kmeans.cluster\_centers\_

predicted\_labels = kmeans.labels\_

plt.scatter(data[:, 0], data[:, 1], c=predicted\_labels, cmap='viridis', edgecolors='k')

plt.scatter(centers[:, 0], centers[:, 1], c='red', marker='x')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('Clustering with K-Means')

plt.show()

## 

## SLIP2

**Q.1 Write a python program the Categorical values in numeric format for a given [10M]**

**Dataset.**

import pandas as pd

df = pd.read\_csv('titanic.csv')

print("Original dataset:")

print(df.head())

label\_encoder = LabelEncoder()

for column in df.select\_dtypes(include=['object']):

df[column] = label\_encoder.fit\_transform(df[column].astype(str))

print("\nDataset after label encoding:")

print(df.head())

**Q.2 Write a python program to implement linear SVM.**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

df = pd.read\_csv('heart.csv')

print("Dataset:")

print(df.head())

X = df.drop('target', axis=1)

y = df['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = SVC(kernel='linear')

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print('\nAccuracy:', accuracy)

print('\nClassification Report:')

print(classification\_report(y\_test, y\_pred))

print('\nConfusion Matrix:')

print(confusion\_matrix(y\_test, y\_pred))

## SLIP3

**Q.1 Write a python program to Prepare Scatter Plot (Use Forge Dataset). [10M]**

pip install matplotlib scikit-learn

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

# Generate synthetic dataset (Forge dataset)

X, y = make\_blobs(n\_samples=100, centers=2, random\_state=0, cluster\_std=0.60)

# Scatter plot

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='viridis')

plt.title("Scatter Plot of Forge Dataset")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.colorbar(label='Label')

plt.show()

**Q.2 Write a python program to implement Agglomerative clustering on a synthetic [20M]**

**Dataset.**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.cluster import AgglomerativeClustering

data, true\_labels = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.60,

random\_state=42)

agglomerative = AgglomerativeClustering(n\_clusters=4)

agglomerative.fit(data)

predicted\_labels = agglomerative.labels\_

plt.scatter(data[:, 0], data[:, 1], c=predicted\_labels, cmap='viridis', edgecolors='k')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('Agglomerative Clustering')

plt.show()

## SLIP4

**Q.1 Write a python program to Implement Decision Tree whether or not to play [10M]**

**Tennis.**

import pandas as pd

from sklearn.tree import DecisionTreeClassifier, plot\_tree

import matplotlib.pyplot as plt

df = pd.read\_csv('PlayTennis.csv')

print("Dataset:")

print(df.head())

df['Outlook'] = df['Outlook'].map({'Sunny': 0, 'Overcast': 1, 'Rain': 2})

df['Temperature'] = df['Temperature'].map({'Hot': 0, 'Mild': 1, 'Cool': 2})

df['Humidity'] = df['Humidity'].map({'High': 0, 'Normal': 1})

df['Wind'] = df['Wind'].map({'Weak': 0, 'Strong': 1})

df['PlayTennis'] = df['PlayTennis'].map({'No': 0, 'Yes': 1})

x = df.drop('PlayTennis', axis=1)

y = df['PlayTennis']

clf = DecisionTreeClassifier()

clf.fit(x, y)

plt.figure(figsize=(10, 6))

plot\_tree(clf, feature\_names=['Outlook', 'Temperature', 'Humidity', 'Wind'],

class\_names=['No', 'Yes'], filled=True)

plt.show()

**Q.2 Write a python Program to transform data with Principal Component Analysis [20M]**

**(PCA)**

import pandas as pd

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

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

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

y = data['Species']

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X)

plt.figure(figsize=(8, 6))

targets = data['Species'].unique()

colors = ['r', 'g', 'b']

for target, color in zip(targets, colors):

indicesToKeep = y == target

plt.scatter(X\_pca[indicesToKeep, 0], X\_pca[indicesToKeep, 1], c=color, label=target)

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('2 Component PCA')

plt.legend(targets)

plt.grid()

plt.show()

## SLIP5

**Q.1 Write a python program to Prepare Scatter Plot (Use Iris Dataset). [10M]**

import pandas as pd

import matplotlib.pyplot as plt

iris = pd.read\_csv(r"C:\Users\DELL\Downloads\Iris.csv")

iris.plot(kind= "scatter",x = "SepalLengthCm",y = "PetalLengthCm")

plt.grid()

Q.2 Write a python program to implement Multiple Linear Regression for given dataset. [20M]

(Use Position\_Salaries Dataset)

Data

import pandas as pd

data = {

'Position': ['Business Analyst', 'Junior Consultant', 'Senior Consultant', 'Manager',

'Country Manager', 'Region Manager', 'Partner', 'Senior Partner',

'C-level', 'CEO'],

'Level': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'Salary': [45000, 50000, 60000, 80000, 110000, 150000, 200000, 300000, 500000, 1000000]

}

df = pd.DataFrame(data)

# Independent variable

X = df[['Level']]

# Dependent variable

y = df['Salary']

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

import numpy as np

import matplotlib.pyplot as plt

# Transforming the independent variable for polynomial regression

poly = PolynomialFeatures(degree=4) # Degree is chosen experimentally. You might need to adjust it.

X\_poly = poly.fit\_transform(X)

# Fitting the polynomial regression model

poly\_reg\_model = LinearRegression()

poly\_reg\_model.fit(X\_poly, y)

# Predicting across the range for plotting

X\_grid = np.arange(min(X['Level']), max(X['Level']), 0.1) # choice of 0.1 to ensure smooth curve

X\_grid = X\_grid.reshape(-1, 1)

y\_pred = poly\_reg\_model.predict(poly.fit\_transform(X\_grid))

# Visualizing the polynomial regression results

plt.scatter(X, y, color='red') # Actual points

plt.plot(X\_grid, y\_pred, color='blue') # Regression line

plt.title('Truth or Bluff (Polynomial Regression)')

plt.xlabel('Position Level')

plt.ylabel('Salary')

plt.show()

## SLIP6

**Q.1 Write a Python program to implement to find all null values in a given Dataset [10M]**

**and remove them.(Use air quality dataset.)**

import pandas as pd

df = pd.read\_csv('AirQuality.csv')

print("Null values before removing:")

print(df.isnull().sum())

df\_cleaned = df.dropna()

print("\nNull values after removing:")

print(df\_cleaned.isnull().sum())

**Q.2 Write a python program to implement k-nearest Neighbors ML algorithm to build [20M**

**prediction model (Use Forge Dataset)**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_classification

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

X, y = make\_classification(n\_samples=100, n\_features=2, n\_informative=2,

n\_redundant=0, random\_state=42)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(X\_train, y\_train)

y\_pred = knn.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.1), np.arange(y\_min, y\_max, 0.1))

Z = knn.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.3)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.bwr, edgecolors='k')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('Decision Boundary by KNN with k=5')

plt.show()

## SLIP7

**Q.1 Read Total profit of all months and show it using a line plot. [10M]**

**(create sales\_data.csv file)**

**Total profit data provided for each month. Generated line plot must include the following**

**properties: –X label name = Month Number, Y label name = Total profit**

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_csv(r"C:\Users\DELL\Downloads\sales data.csv")

print(df.head())

plt.figure(figsize=(10,6))

plt.plot(df['Month Number'], df['Total Profit'],marker='o',linestyle='-',color='b')

plt.title("Monthly sales")

plt.xlabel('month number')

plt.ylabel('Total Profit')

plt.grid(True, which= 'both',linestyle='--',linewidth=0.5)

plt.show()

Q.2 Write a python program to implement Eclat Algorithm (OnlineRetail.xlsx dataset)

import numpy as np

import pandas as pd

pip install apyori

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

#importing dataset

ds = pd.read\_csv(r"C:\Users\DELL\Downloads\market basket optimisation.csv", header = None)

transactions = []

for i in range(0, 7501):

transactions.append([str(ds.values[i,j]) for j in range(0, 20)])

#training eclat model to the dataset

from apyori import apriori

rules = apriori(transactions = transactions, min\_support = 0.003, min\_confidence = 0.2, min\_lift = 3, min\_length = 2, max\_length = 2)

#Displaying the first results coming directly from the output of the apriori function

results = list(rules)

results

#Putting the results well organised into a Pandas DataFrame

def inspect(results):

lhs = [tuple(result[2][0][0])[0] for result in results]

rhs = [tuple(result[2][0][1])[0] for result in results]

supports = [result[1] for result in results]

return list(zip(lhs, rhs, supports))

resultsinDataFrame = pd.DataFrame(inspect(results), columns = ['Product 1', 'Product 2', 'Support'])

#Displaying the results sorted by descending supports

resultsinDataFrame.nlargest(n = 10, columns = 'Support')

## SLIP8

**Q.1 Write a python program to Implement Naïve Bayes. [10M]**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

df = pd.read\_csv('iris.csv')

print("Dataset:")

print(df.head())

X = df.drop('Species', axis=1)

y = df['Species']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = GaussianNB()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print('\nAccuracy:', accuracy)

print('\nClassification Report:')

print(classification\_report(y\_test, y\_pred))

print('\nConfusion Matrix:')

print(confusion\_matrix(y\_test, y\_pred))

Q.2 Write a python program to implement Multiple Linear Regression for given [20M]

dataset.(Position\_Salaries Dataset)

Data

import pandas as pd

data = {

'Position': ['Business Analyst', 'Junior Consultant', 'Senior Consultant', 'Manager',

'Country Manager', 'Region Manager', 'Partner', 'Senior Partner',

'C-level', 'CEO'],

'Level': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'Salary': [45000, 50000, 60000, 80000, 110000, 150000, 200000, 300000, 500000, 1000000]

}

df = pd.DataFrame(data)

# Independent variable

X = df[['Level']]

# Dependent variable

y = df['Salary']

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

import numpy as np

import matplotlib.pyplot as plt

# Transforming the independent variable for polynomial regression

poly = PolynomialFeatures(degree=4) # Degree is chosen experimentally. You might need to adjust it.

X\_poly = poly.fit\_transform(X)

# Fitting the polynomial regression model

poly\_reg\_model = LinearRegression()

poly\_reg\_model.fit(X\_poly, y)

# Predicting across the range for plotting

X\_grid = np.arange(min(X['Level']), max(X['Level']), 0.1) # choice of 0.1 to ensure smooth curve

X\_grid = X\_grid.reshape(-1, 1)

y\_pred = poly\_reg\_model.predict(poly.fit\_transform(X\_grid))

# Visualizing the polynomial regression results

plt.scatter(X, y, color='red') # Actual points

plt.plot(X\_grid, y\_pred, color='blue') # Regression line

plt.title('Truth or Bluff (Polynomial Regression)')

plt.xlabel('Position Level')

plt.ylabel('Salary')

plt.show()

## SLIP9

**Q.1 Write a python program to Prepare Scatter Plot (Use Iris Dataset). [10M]**

import pandas as pd

import matplotlib.pyplot as plt

iris = pd.read\_csv(r"C:\Users\DELL\Downloads\Iris.csv")

iris.plot(kind= "scatter",

x = "SepalLengthCm",

y = "PetalLengthCm")

plt.grid()

**Q.2 Write a python program to implement Naive Bayes. (social-network-ads)**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

df = pd.read\_csv('iris.csv')

print("Dataset:")

print(df.head())

X = df.drop('Species', axis=1)

y = df['Species']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = GaussianNB()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print('\nAccuracy:', accuracy)

print('\nClassification Report:')

print(classification\_report(y\_test, y\_pred))

print('\nConfusion Matrix:')

print(confusion\_matrix(y\_test, y\_pred))

## SLIP10

**Q.1 Get total profit of all months and show line plot with the following Style properties. [10M]**

**(create sales\_data.csv file)**

**To generate line plot must include following Style properties: –Line Style dotted and Line-color**

**should be red, Show legend at the lower right location , label name = Month Number ,**

**label name = Sold units number, Add a circle marker, Line marker color as read, Line width**

**should be 3**

import pandas as pd

import matplotlib.pyplot as plt

# Read the dataset

df = pd.read\_csv(r"C:\Users\DELL\Downloads\sales data.csv")

# Plotting

plt.figure(figsize=(10, 6)) # Adjusts the figure size

plt.plot(df['Month Number'], df['Total Profit'],

linestyle=':', # Dotted line

color='red', # Line color red

marker='o', # Circle marker

markerfacecolor='red', # Line marker color red

linewidth=3, # Line width as 3

label='Total Profit') # Label for the legend

# Adding title and labels

plt.title('Monthly Sales Data')

plt.xlabel('Month Number')

plt.ylabel('Total Profit')

# Display grid lines

plt.grid(True, which='both', linestyle='--', linewidth=0.5)

# Show legend at the lower right location

plt.legend(loc='lower right')

# Show the plot

plt.show()

Q.2 Write a python program to implement Multiple Linear Regression for given dataset. [20M]

(Create Own Dataset)

Creating dataset

import pandas as pd

import numpy as np

# Setting a random seed for reproducibility

np.random.seed(0)

# Creating a synthetic dataset

data\_size = 100

feature\_1 = np.random.rand(data\_size) \* 10 # Random values between 0 and 10

feature\_2 = np.random.rand(data\_size) \* 20 # Random values between 0 and 20

feature\_3 = np.random.rand(data\_size) \* 5 # Random values between 0 and 5

# Assuming a linear relationship with some noise

target = 3\*feature\_1 + 2\*feature\_2 + 4\*feature\_3 + np.random.randn(data\_size) \* 2

# Creating a DataFrame

df = pd.DataFrame({

'Feature 1': feature\_1,

'Feature 2': feature\_2,

'Feature 3': feature\_3,

'Target': target

})

# Display the first few rows of the dataset

print(df.head())

from sklearn.model\_selection import train\_test\_split

X = df[['Feature 1', 'Feature 2', 'Feature 3']] # Independent variables

y = df['Target'] # Dependent variable

# Splitting dataset into training and testing set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Creating a model object

model = LinearRegression()

# Fitting the model on the training data

model.fit(X\_train, y\_train)

# Making predictions on the testing set

y\_pred = model.predict(X\_test)

# Evaluating the model

mse = mean\_squared\_error(y\_test, y\_pred)

print(f'Mean Squared Error: {mse:.2f}')

# Model coefficients and intercept

print(f'Coefficients: {model.coef\_}')

print(f'Intercept: {model.intercept\_}')

# The coefficients correspond to the weights of each feature, and the intercept is the bias.

## SLIP11

**Q.1 Read all product sales data and show it using a multiline plot . [10M]**

**(create sales\_data.csv file)**

**Display the number of units sold per month for each product using multiline plots. (i.e., Separate**

**Plotline for each product ).The graph should look like this.**

**#Data**

| Month Number | Product 1 | Product 2 | Product 3 |
| --- | --- | --- | --- |
| 1 | 1200 | 1500 | 1300 |
| 2 | 1800 | 1600 | 1400 |
| 3 | 1400 | 1700 | 1200 |
| 4 | 1500 | 1600 | 1300 |
| 5 | 1600 | 1500 | 1200 |
| 6 | 1700 | 1400 | 1300 |
| 7 | 1800 | 1300 | 1400 |
| 8 | 1600 | 1200 | 1500 |
| 9 | 1400 | 1100 | 1600 |
| 10 | 1300 | 1200 | 1700 |
| 11 | 1200 | 1300 | 1800 |
| 12 | 1100 | 1400 | 1900 |

df = pd.read\_csv(r"C:\Users\DELL\Downloads\sales data.csv")

plt.figure(figsize=(10,6))

#plotting each product

plt.plot(df['Month Number'],df['Product 1'], marker = 'o', label="Product 1")

plt.plot(df['Month Number'],df['Product 2'],marker = 's', label= "Product 2")

plt.plot(df['Month Number'],df['Product 3'], marker = '^',label = "Product 3")

plt.title('Monthly Sales Data')

plt.xlabel('Month Number')

plt.ylabel('Sales Units')

plt.legend()

plt.grid(True, which='both',linestyle="--",linewidth= 0.5)

plt.show()

**Q.2 Write a python program to implement k-means algorithm on a synthetic [20M]**

**Dataset.**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

data, labels = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.60,

random\_state=42)

kmeans = KMeans(n\_clusters=4)

kmeans.fit(data)

centers = kmeans.cluster\_centers\_

predicted\_labels = kmeans.labels\_

plt.scatter(data[:, 0], data[:, 1], c=predicted\_labels, cmap='viridis', edgecolors='k')

plt.scatter(centers[:, 0], centers[:, 1], c='red', marker='x')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('Clustering with K-Means')

plt.show()

## SLIP12

**Q.1 Write a Python program to implement to find all null values in a given [10M]**

**Data set and remove them.(Use air quality dataset.)**

import pandas as pd

df = pd.read\_csv('titanic.csv')

print("Null values before removing:")

print(df.isnull().sum())

df\_cleaned = df.dropna()

print("\nNull values after removing:")

print(df\_cleaned.isnull().sum())

**Q.2 Write a python program to implement Agglomerative clustering on a synthetic [20M]**

**Dataset.**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.cluster import AgglomerativeClustering

data, true\_labels = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.60,

random\_state=42)

agglomerative = AgglomerativeClustering(n\_clusters=4)

agglomerative.fit(data)

predicted\_labels = agglomerative.labels\_

plt.scatter(data[:, 0], data[:, 1], c=predicted\_labels, cmap='viridis', edgecolors='k')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('Agglomerative Clustering')

plt.show()

## 

## SLIP13

Q.1 Read face cream and facewash product sales data and show it using the bar chart. [10M]

The bar chart should display the number of units sold per month for each product. Add a separate

bar for each product in the same chart. (create sales\_data.csv file)

import pandas as pd

import matplotlib.pyplot as plt

# Read the data from CSV

sales\_data = pd.read\_csv(r"C:\Users\DELL\Downloads\sales data slip13.csv")

# Extracting months and sales data for each product

months = sales\_data['Month']

face\_cream\_sales = sales\_data['Face Cream Sales']

face\_wash\_sales = sales\_data['Face Wash Sales']

# Setting the width of each bar

bar\_width = 0.35

# Creating the positions for the bars

index = np.arange(len(months))

# Plotting the side-by-side bar chart

plt.figure(figsize=(10, 6))

plt.bar(index - bar\_width/2, face\_cream\_sales, bar\_width, color='blue', label='Face Cream')

plt.bar(index + bar\_width/2, face\_wash\_sales, bar\_width, color='orange', label='Face Wash')

plt.xlabel('Month')

plt.ylabel('Units Sold')

plt.title('Face Cream and Face Wash Sales per Month')

plt.xticks(index, months, rotation=45)

plt.legend()

plt.tight\_layout()

# Show the plot

plt.show()

Q.2 Write a python program to implement Multiple Linear Regression for given dataset. [20M]

(Use Position\_Salaries Dataset)

Data

import pandas as pd

data = {

'Position': ['Business Analyst', 'Junior Consultant', 'Senior Consultant', 'Manager',

'Country Manager', 'Region Manager', 'Partner', 'Senior Partner',

'C-level', 'CEO'],

'Level': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'Salary': [45000, 50000, 60000, 80000, 110000, 150000, 200000, 300000, 500000, 1000000]

}

df = pd.DataFrame(data)

# Independent variable

X = df[['Level']]

# Dependent variable

y = df['Salary']

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

import numpy as np

import matplotlib.pyplot as plt

# Transforming the independent variable for polynomial regression

poly = PolynomialFeatures(degree=4) # Degree is chosen experimentally. You might need to adjust it.

X\_poly = poly.fit\_transform(X)

# Fitting the polynomial regression model

poly\_reg\_model = LinearRegression()

poly\_reg\_model.fit(X\_poly, y)

# Predicting across the range for plotting

X\_grid = np.arange(min(X['Level']), max(X['Level']), 0.1) # choice of 0.1 to ensure smooth curve

X\_grid = X\_grid.reshape(-1, 1)

y\_pred = poly\_reg\_model.predict(poly.fit\_transform(X\_grid))

# Visualizing the polynomial regression results

plt.scatter(X, y, color='red') # Actual points

plt.plot(X\_grid, y\_pred, color='blue') # Regression line

plt.title('Truth or Bluff (Polynomial Regression)')

plt.xlabel('Position Level')

plt.ylabel('Salary')

plt.show()

## SLIP14

Q.1 Write a python program to Prepare Scatter Plot (Use Iris Dataset). [10M]

import pandas as pd

import matplotlib.pyplot as plt

data=pd.read\_csv("iris.csv")

x=data["SepalLengthCm"]

y=data["SepalWidthCm"]

plt.scatter(x,y,c="red")

plt.title("IRIS")

plt.xlabel("SepalLengthCm")

plt.ylabel("SepalWidthCm")

plt.show()

Q.2 Write a python program to implement Multiple Linear Regression for given dataset. [20M]

(Use Position\_Salaries Dataset)

Data

import pandas as pd

data = {

'Position': ['Business Analyst', 'Junior Consultant', 'Senior Consultant', 'Manager',

'Country Manager', 'Region Manager', 'Partner', 'Senior Partner',

'C-level', 'CEO'],

'Level': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'Salary': [45000, 50000, 60000, 80000, 110000, 150000, 200000, 300000, 500000, 1000000]

}

df = pd.DataFrame(data)

# Independent variable

X = df[['Level']]

# Dependent variable

y = df['Salary']

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

import numpy as np

import matplotlib.pyplot as plt

# Transforming the independent variable for polynomial regression

poly = PolynomialFeatures(degree=4) # Degree is chosen experimentally. You might need to adjust it.

X\_poly = poly.fit\_transform(X)

# Fitting the polynomial regression model

poly\_reg\_model = LinearRegression()

poly\_reg\_model.fit(X\_poly, y)

# Predicting across the range for plotting

X\_grid = np.arange(min(X['Level']), max(X['Level']), 0.1) # choice of 0.1 to ensure smooth curve

X\_grid = X\_grid.reshape(-1, 1)

y\_pred = poly\_reg\_model.predict(poly.fit\_transform(X\_grid))

# Visualizing the polynomial regression results

plt.scatter(X, y, color='red') # Actual points

plt.plot(X\_grid, y\_pred, color='blue') # Regression line

plt.title('Truth or Bluff (Polynomial Regression)')

plt.xlabel('Position Level')

plt.ylabel('Salary')

plt.show()

## SLIP15

**Q.1 Write a python program to Implement Naïve Bayes. [10M]**

**import pandas as pd**

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

df = pd.read\_csv('iris.csv')

print("Dataset:")

print(df.head())

X = df.drop('Species', axis=1)

y = df['Species']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = GaussianNB()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print('\nAccuracy:', accuracy)

print('\nClassification Report:')

print(classification\_report(y\_test, y\_pred))

print('\nConfusion Matrix:')

print(confusion\_matrix(y\_test, y\_pred))

Q.2 Write a python program to implement linear SVM.

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

df = pd.read\_csv('heart.csv')

print("Dataset:")

print(df.head())

X = df.drop('target', axis=1)

y = df['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

model = SVC(kernel='linear')

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print('\nAccuracy:', accuracy)

print('\nClassification Report:')

print(classification\_report(y\_test, y\_pred))

print('\nConfusion Matrix:')

print(confusion\_matrix(y\_test, y\_pred))

## SLIP16

**Q.1 Write a Python program to implement to find all null values in a given [10M]**

**Data set and remove them.(Use air quality dataset.)**

import pandas as pd

df = pd.read\_csv('titanic.csv')

print("Null values before removing:")

print(df.isnull().sum())

df\_cleaned = df.dropna()

print("\nNull values after removing:")

print(df\_cleaned.isnull().sum())

Q.2 Write a python program to implement Multiple Linear Regression for given dataset. [20M]

(Use Position\_Salaries Dataset)

Data

import pandas as pd

data = {

'Position': ['Business Analyst', 'Junior Consultant', 'Senior Consultant', 'Manager',

'Country Manager', 'Region Manager', 'Partner', 'Senior Partner',

'C-level', 'CEO'],

'Level': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],

'Salary': [45000, 50000, 60000, 80000, 110000, 150000, 200000, 300000, 500000, 1000000]

}

df = pd.DataFrame(data)

# Independent variable

X = df[['Level']]

# Dependent variable

y = df['Salary']

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

import numpy as np

import matplotlib.pyplot as plt

# Transforming the independent variable for polynomial regression

poly = PolynomialFeatures(degree=4) # Degree is chosen experimentally. You might need to adjust it.

X\_poly = poly.fit\_transform(X)

# Fitting the polynomial regression model

poly\_reg\_model = LinearRegression()

poly\_reg\_model.fit(X\_poly, y)

# Predicting across the range for plotting

X\_grid = np.arange(min(X['Level']), max(X['Level']), 0.1) # choice of 0.1 to ensure smooth curve

X\_grid = X\_grid.reshape(-1, 1)

y\_pred = poly\_reg\_model.predict(poly.fit\_transform(X\_grid))

# Visualizing the polynomial regression results

plt.scatter(X, y, color='red') # Actual points

plt.plot(X\_grid, y\_pred, color='blue') # Regression line

plt.title('Truth or Bluff (Polynomial Regression)')

plt.xlabel('Position Level')

plt.ylabel('Salary')

plt.show()

## SLIP17

**Q.1 Write a python program to Implement Decision Tree whether or not to play [10M]**

**Tennis.**

import pandas as pd

from sklearn.tree import DecisionTreeClassifier, plot\_tree

import matplotlib.pyplot as plt

df = pd.read\_csv('PlayTennis.csv')

print("Dataset:")

print(df.head())

df['Outlook'] = df['Outlook'].map({'Sunny': 0, 'Overcast': 1, 'Rain': 2})

df['Temperature'] = df['Temperature'].map({'Hot': 0, 'Mild': 1, 'Cool': 2})

df['Humidity'] = df['Humidity'].map({'High': 0, 'Normal': 1})

df['Wind'] = df['Wind'].map({'Weak': 0, 'Strong': 1})

df['PlayTennis'] = df['PlayTennis'].map({'No': 0, 'Yes': 1})

x = df.drop('PlayTennis', axis=1)

y = df['PlayTennis']

clf = DecisionTreeClassifier()

clf.fit(x, y)

plt.figure(figsize=(10, 6))

plot\_tree(clf, feature\_names=['Outlook', 'Temperature', 'Humidity', 'Wind'],

class\_names=['No', 'Yes'], filled=True)

plt.show()

Q.2 Write a python Program to transform data with Principal Component Analysis [20M]

(PCA)

import pandas as pd

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

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

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

y = data['Species']

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X)

plt.figure(figsize=(8, 6))

targets = data['Species'].unique()

colors = ['r', 'g', 'b']

for target, color in zip(targets, colors):

indicesToKeep = y == target

plt.scatter(X\_pca[indicesToKeep, 0], X\_pca[indicesToKeep, 1], c=color, label=target)

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('2 Component PCA')

plt.legend(targets)

plt.grid()

plt.show()

## SLIP18

Q.1 Read sales data of bathing soap of all months and show it using a bar chart. [10M]

Save this plot to your hard disk. (create sales\_data.csv file)

Data

| Month | Bathing soap sales | |
| --- | --- | --- |
| 1 | 1200 |  |
| 2 | 1300 |  |
| 3 | 1350 |  |
| 4 | 1400 |  |
| 5 | 1450 |  |
| 6 | 1520 |  |
| 7 | 1580 |  |
| 8 | 1640 |  |
| 9 | 1700 |  |
| 10 | 1760 |  |
| 11 | 1820 |  |
| 12 | 1880 |  |
| 13 | 1940 |  |
| 14 | 2000 |  |
| 15 | 2060 |  |

df =pd.read\_csv(r"C:\Users\DELL\Downloads\sales data.csv")

plt.figure(figsize=(10,6))

plt.bar(df['Month'],df['Bathing soap sales'],color = "Skyblue")

plt.title("Monthly sales of Bathing Soap")

plt.xlabel("Month")

plt.ylabel("sales of Bathing Soap")

plt.savefig("Bath soap sale.png")

plt.show()

**Q.2 Write a python program to implement Multiple Linear Regression for given dataset. [20M]**

(Create Own Dataset)

Creating dataset

import pandas as pd

import numpy as np

# Setting a random seed for reproducibility

np.random.seed(0)

# Creating a synthetic dataset

data\_size = 100

feature\_1 = np.random.rand(data\_size) \* 10 # Random values between 0 and 10

feature\_2 = np.random.rand(data\_size) \* 20 # Random values between 0 and 20

feature\_3 = np.random.rand(data\_size) \* 5 # Random values between 0 and 5

# Assuming a linear relationship with some noise

target = 3\*feature\_1 + 2\*feature\_2 + 4\*feature\_3 + np.random.randn(data\_size) \* 2

# Creating a DataFrame

df = pd.DataFrame({

'Feature 1': feature\_1,

'Feature 2': feature\_2,

'Feature 3': feature\_3,

'Target': target

})

# Display the first few rows of the dataset

print(df.head())

from sklearn.model\_selection import train\_test\_split

X = df[['Feature 1', 'Feature 2', 'Feature 3']] # Independent variables

y = df['Target'] # Dependent variable

# Splitting dataset into training and testing set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Creating a model object

model = LinearRegression()

# Fitting the model on the training data

model.fit(X\_train, y\_train)

# Making predictions on the testing set

y\_pred = model.predict(X\_test)

# Evaluating the model

mse = mean\_squared\_error(y\_test, y\_pred)

print(f'Mean Squared Error: {mse:.2f}')

# Model coefficients and intercept

print(f'Coefficients: {model.coef\_}')

print(f'Intercept: {model.intercept\_}')

## SLIP19

**Q.1 Write a python program to Prepare Scatter Plot (Use Forge Dataset). [10M]**

pip install matplotlib scikit-learn

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

# Generate synthetic dataset (Forge dataset)

X, y = make\_blobs(n\_samples=100, centers=2, random\_state=0, cluster\_std=0.60)

# Scatter plot

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='viridis')

plt.title("Scatter Plot of Forge Dataset")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.colorbar(label='Label')

plt.show()

**Q.2 Write a python program to implement k-means algorithm on a synthetic Dataset. [20M]**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

data, labels = make\_blobs(n\_samples=300, centers=4, cluster\_std=0.60,

random\_state=42)

kmeans = KMeans(n\_clusters=4)

kmeans.fit(data)

centers = kmeans.cluster\_centers\_

predicted\_labels = kmeans.labels\_

plt.scatter(data[:, 0], data[:, 1], c=predicted\_labels, cmap='viridis', edgecolors='k')

plt.scatter(centers[:, 0], centers[:, 1], c='red', marker='x')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.title('Clustering with K-Means')

plt.show()

## SLIP20

**Q.1 Calculate total sale data for last year for each product and show it using a Pie chart. [10M]**

**(create sales\_data.csv file)**

import pandas as pd

import matplotlib.pyplot as plt

# Read the CSV file

sales\_data = pd.read\_csv("sales\_data.csv")

# Group by product and sum the sales

product\_sales = sales\_data.groupby('Product')['Sale'].sum()

# Plotting

plt.figure(figsize=(8, 8))

plt.pie(product\_sales, labels=product\_sales.index, autopct='%1.1f%%', startangle=140)

plt.title('Total Sales Data for Each Product')

plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()

Q.2 Write a python program to implement Eclat Algorithm (OnlineRetail.xlsx dataset)

import numpy as np

import pandas as pd

!pip install apyori

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

#importing dataset

ds = pd.read\_csv(r"C:\Users\DELL\Downloads\market basket optimisation.csv", header = None)

transactions = []

for i in range(0, 7501):

transactions.append([str(ds.values[i,j]) for j in range(0, 20)])

#training eclat model to the dataset

from apyori import apriori

rules = apriori(transactions = transactions, min\_support = 0.003, min\_confidence = 0.2, min\_lift = 3, min\_length = 2, max\_length = 2)

#Displaying the first results coming directly from the output of the apriori function

results = list(rules)

results

#Putting the results well organised into a Pandas DataFrame

def inspect(results):

lhs = [tuple(result[2][0][0])[0] for result in results]

rhs = [tuple(result[2][0][1])[0] for result in results]

supports = [result[1] for result in results]

return list(zip(lhs, rhs, supports))

resultsinDataFrame = pd.DataFrame(inspect(results), columns = ['Product 1', 'Product 2', 'Support'])

#Displaying the results sorted by descending supports

resultsinDataFrame.nlargest(n = 10, columns = 'Support')