

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT on

Machine Learning (23CS6PCMAL)

Submitted by

Nandini A T (1BM23CS411)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING



B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Sep-2024 to Jan-2025

B.M.S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering



CERTIFICATE

This is to certify that the Lab work entitled “Machine Learning (23CS6PCMAL)” carried out by **Nandini A T (1BM23CS411)**, who is a bonafide student of **B.M.S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Machine Learning (23CS6PCMAL) work prescribed for the said degree.

Dr. Pallavi G B Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE
-----------------------------------------------------------------------	------------------------------------------------------------------

Index

Sl. No.	Date	Experiment Title	Page No.
1	21-2-2025	Write a python program to import and export data using Pandas library functions	4
2	3-3-2025	Demonstrate various data pre-processing techniques for a given dataset	7
3	10-3-2025	Implement Linear and Multi-Linear Regression algorithm using appropriate dataset	11
4	17-3-2025	Build Logistic Regression Model for a given dataset	15
5	24-3-2025	Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample.	18
6	7-4-2025	Build KNN Classification model for a given dataset.	21
7	21-4-2025	Build Support vector machine model for a given dataset	24
8	5-5-2025	Implement Random forest ensemble method on a given dataset.	27
9	5-5-2025	Implement Boosting ensemble method on a given dataset.	29
10	12-5-2025	Build k-Means algorithm to cluster a set of data stored in a .CSV file.	31
11	12-5-2025	Implement Dimensionality reduction using Principal Component Analysis (PCA) method.	33

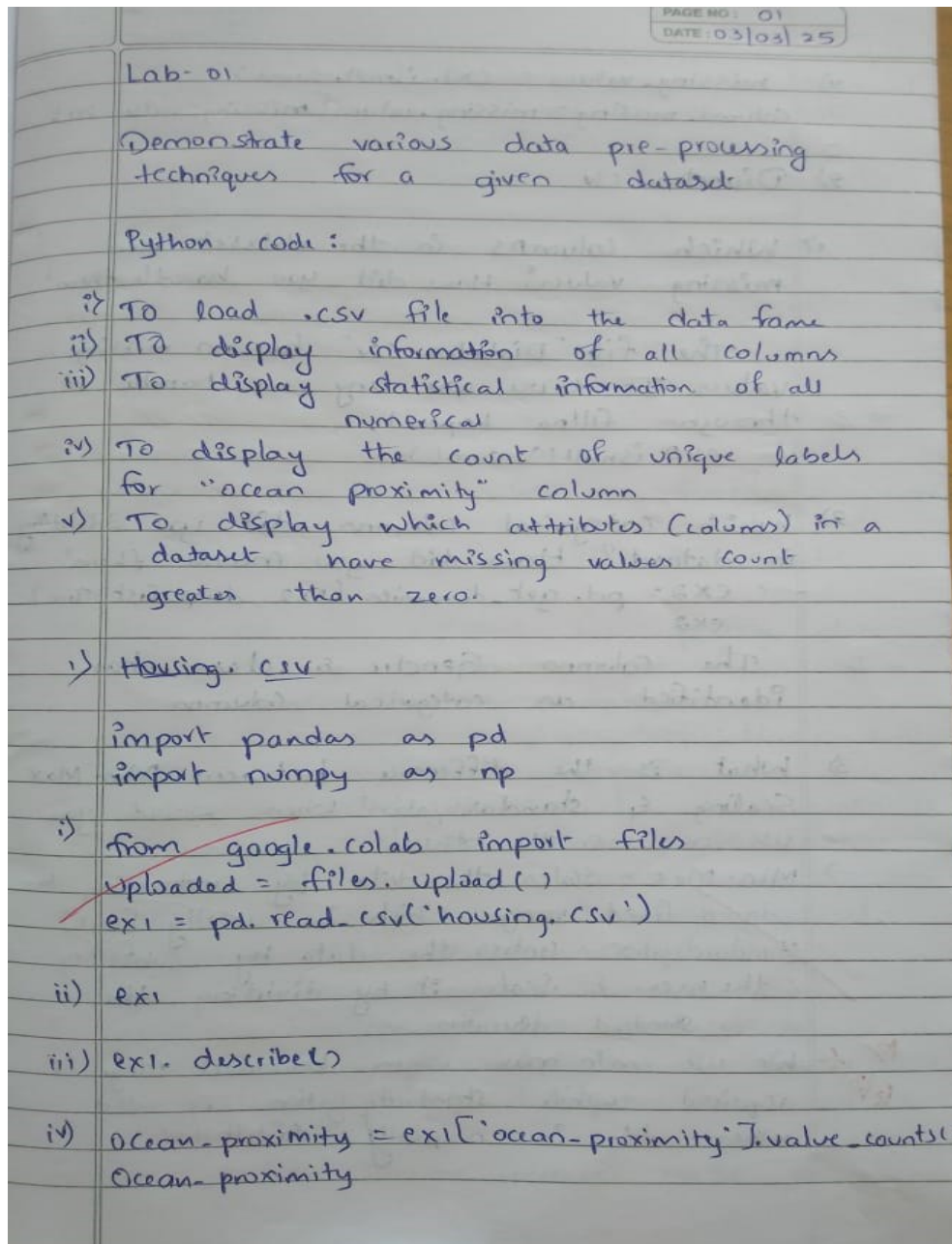
Github Link:

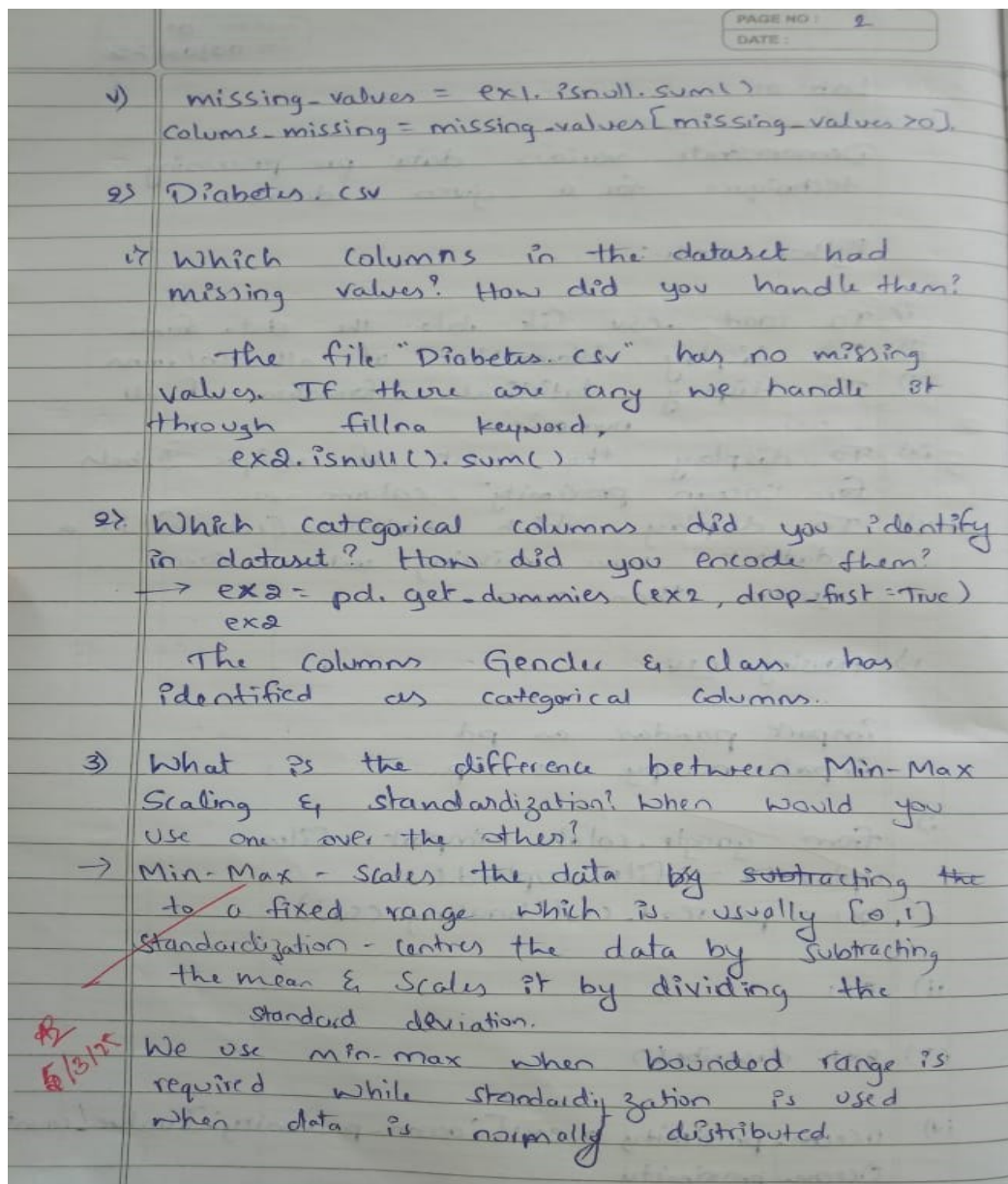
<https://github.com/NandiniAT/ML.git>

Program 1

Write a python program to import and export data using Pandas library functions

Screenshot:





Code:

Housing

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from scipy import stats
from google.colab import files
uploaded=files.upload()
ex1=pd.read_csv('housing.csv')
ex1
ex1.describe()
Ocean_proximity=ex1['ocean_proximity'].value_counts()
Ocean_proximity
```

```

missing_values = ex1.isnull().sum()
columns_missing_values = missing_values[missing_values > 0]
columns_missing_values

```

Diabetes

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from scipy import stats

from google.colab import files
uploaded=files.upload()

ex2=pd.read_csv('Dataset of Diabetes .csv')
ex2
ex2.describe()
# Handling missing values
ex2.isnull().sum()
#Handling categorical data
ex2 = pd.get_dummies(ex2, drop_first=True)
ex2
#Handling Outliers using IQR method (only for numerical columns)
print("Handling Outliers...")
numeric_cols = ex2.select_dtypes(include=[np.number]).columns
Q1 = ex2[numeric_cols].quantile(0.25)
Q1
Q3 = ex2[numeric_cols].quantile(0.75)
Q3
IQR = Q3 - Q1
IQR
lower_bound = Q1 - 1.5 * IQR
lower_bound
upper_bound = Q3 + 1.5 * IQR
upper_bound
ex2 = ex2[~((ex2[numeric_cols] < lower_bound) | (ex2[numeric_cols] > upper_bound)).any(axis=1)]
ex2
ex2.isnull().sum()
ex2.boxplot("AGE")
#Min-Max Scaling (Normalization)
print("Applying Min-Max Scaling...")
scaler = MinMaxScaler()
scaler
df_scaled = pd.DataFrame(scaler.fit_transform(ex2[numeric_cols]), columns=numeric_cols)
df_scaled
#Standard Scaling
print("Applying Standard Scaling...")
scaler = StandardScaler()
scaler
print("\nCleaned Data:")
(ex2.head())
print("\nNormalized Data:")
print(df_scaled.head())

```

Program 2

Demonstrate various data pre-processing techniques for a given dataset

Screenshot:

lab-2

PAGE NO: 3
DATE: 10/03/25

Demonstrate the steps to build a ML-model that predicts the median housing price using the given dataset

1. `print(df.info())`
`print(df.describe())`
2. print x label as 'columns'
print y label as 'frequency'
plot histogram for each plot
interpretation:
median income shows median income values of dataset indicates high/low income
house median age shows distribution of median house age in different blocks.
3. test set is created by splitting data into training & testing set by random sampling. (or) stratified sampling.
random sampling: samples datapoints randomly without considering distribution.
~~stratified~~ sampling: divides data into homogenous sub-groups based on some specific feature.
4. 'ocean-proximity' indicated geographical feature in dataset.
5. median income feature correlates to maximum graph indicates positive correlation
cropped house values data spread & few outliers.

As median income increases median house values

also increases the correlation there are a few outliers

6. Features that could be combined to improve correlation could be:

- rooms per household = total_rooms / house_holds
- bedrooms per room = total_bedrooms / total_rooms
- population per household = population / household

conclusion - Correlation has remained constant even after combining features.

7. Features that need to be cleaned include total_bedrooms, ocean_proximity, median_house_value. Cleaning of data is done by checking missing values, encoding categorical values, features scaling, outlier handling.

8. The ocean_proximity column is categorical that can be converted to numerical data by using One hot encoding to convert where the data is reshaped, transformed & then converted.

9. Feature scaling is highly important due to various factors such as

- * improved model performance
- * faster convergence
- * enhanced interpretability
- * prevention of numerical issues

Common scaling techniques are:

- * min-max scaling
- * standardization
- * robust scaling

PAGE NO: 5
DATE:

10. Numerical pipeline is done by imputation (fills missing value with median value), Custom transformations (adding features), Scaling. Categorical pipelining is done by one hot encoding.

10/3/25

Code:

```
from google.colab import files
diabetes=files.upload()
from google.colab import files
adult_income=files.upload()
df1=pd.read_csv("Dataset of Diabetes .csv")
df1.head()
df2=pd.read_csv("adult.csv")
df2.head()
df1.info()
df2.info()
```



```

df1.describe()
df2.describe()

missing_values1=df1.isnull().sum()
print(missing_values1)
missing_values2 = df2.isnull().sum()
print(missing_values2)

df1['Gender']=df1['Gender'].replace('f', 'F')
ordinal_encoder=OrdinalEncoder(categories=[["F",
"M"]])
df1["Gender_Encoded"]=ordinal_encoder.fit_transform(df1[["Gender"]])
onehot_encoder = OneHotEncoder()
encoded_data = onehot_encoder.fit_transform(df1[["CLASS"]])
encoded_array = encoded_data.toarray()
encoded_df = pd.DataFrame(encoded_array, columns=onehot_encoder.get_feature_names_out(["CLASS"]))
df_encoded = pd.concat([df1, encoded_df], axis=1)
df1 = pd.concat([df1, encoded_df], axis=1)
df1.drop("CLASS", axis=1, inplace=True)
df1.drop("Gender", axis=1, inplace=True)
print(df2.head())

from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
df_copy2 = df2
ordinal_encoder = OrdinalEncoder(categories=[["Male", "Female"]])
df_copy2["Gender_Encoded"] = ordinal_encoder.fit_transform(df_copy2[["gender"]])
print(df_copy2[["gender", "Gender_Encoded"]])

onehot_encoder = OneHotEncoder()
encoded_data =
onehot_encoder.fit_transform(df2[["occupation", "workclass", "education", "marital-status", "relationship", "race", "native-country", "income"]])
encoded_array = encoded_data.toarray()
encoded_df = pd.DataFrame(encoded_array,
columns=onehot_encoder.get_feature_names_out(["occupation", "workclass", "education", "marital-status", "relationship", "race", "native-country", "income"]))
df_encoded = pd.concat([df_copy2, encoded_df], axis=1)

df_encoded.drop("gender", axis=1, inplace=True)
df_encoded.drop("occupation", axis=1, inplace=True)
df_encoded.drop("workclass", axis=1, inplace=True)
df_encoded.drop("education", axis=1, inplace=True)
df_encoded.drop("marital-status", axis=1, inplace=True)
df_encoded.drop("relationship", axis=1, inplace=True)
df_encoded.drop("race", axis=1, inplace=True)
df_encoded.drop("native-country", axis=1, inplace=True)
df_encoded.drop("income", axis=1, inplace=True)
print(df_encoded.head())

normalizer = MinMaxScaler()
df_encoded[["fnlwgt", "educational-num", "capital-gain", "capital-loss", "hours-per-week"]] =
normalizer.fit_transform(df_encoded[["fnlwgt", "educational-num", "capital-gain", "capital-loss", "hours-per-week"]
])
df_encoded.head()

```

```
normalizer = MinMaxScaler()
df1[["No_Pation","AGE","Urea","Cr" , "HbA1c" , "Chol","TG","HDL","LDL","VLDL","BMI"]] =
normalizer.fit_transform(df1[["No_Pation","AGE","Urea","Cr" , "HbA1c" ,
"Chol","TG","HDL","LDL","VLDL","BMI"]])
df1.head()
```

Program 3

Implement Linear and Multi-Linear Regression algorithm using appropriate dataset

Screenshot:

PAGE NO : 8
DATE : 17/03/25

Lab-3

1. Implement Linear & Multi-Linear Regression algorithm using appropriate dataset

1. Solve the following Linear Regression Problem using Matrix approach. Find Linear Regression of the data of week & product sales.

x_i (Week)	y_i (Sales in thousands)
1	2
2	4
3	5
4	9

$$\beta = (X^T X)^{-1} X^T Y$$
$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$\rightarrow X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \quad Y = \begin{bmatrix} 2 \\ 4 \\ 5 \\ 9 \end{bmatrix}$

$$X^T X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}$$
$$= \begin{bmatrix} 4 & 10 \\ 10 & 30 \end{bmatrix}$$
$$X^T Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 5 \\ 9 \end{bmatrix} = \begin{bmatrix} 20 \\ 61 \end{bmatrix}$$

Inverse of $X^T X$

$$(X^T X)^{-1} = \begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{bmatrix}$$

→ compute $\beta = (X^T X)^{-1} X^T Y$

$$= \begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{bmatrix} \begin{bmatrix} 20 \\ 61 \end{bmatrix}$$

$$= \begin{bmatrix} -0.5 \\ 9.2 \end{bmatrix}$$

Regression Line Equation $y = -0.5 + 2.2x$

2. Simple Linear Regression

Diameter (X) Price (Y) Predict the price of 20 inch pipe using the data given

X	Y	$X_i \cdot X_i$	$X_i \cdot Y_i$
8	10	64	80
10	13	100	130
12	16	144	192

$$n = 30$$

$$a_1 = \frac{(X Y) - (\bar{X})(\bar{Y})}{(\bar{X}^2) - (\bar{X})^2}$$

$$= \frac{134 - (10)(13)}{(102.67) - 10^2} = \frac{134 - 130}{2.67} = \frac{4}{2.67} = 1.49$$

$$a_1 \approx 1.5$$

$$a_0 = (\bar{Y}) - a_1 \times \bar{X}$$

$$= 13 - 1.5 \times 10$$

$$= -2$$

$$y = -2 + 1.5x$$

The prediction of price of 20 is

$$y = -2 + 1.5(20)$$

$$y = 28$$

After building the regression models, write the following questions

1. Data Preprocessing Steps:

- * Missing Values: Handled by imputing or dropping missing data
- * Scaling: Applied to features with different scales to ensure uniformity
- * Encoding: Categorical variables were encoded to convert them to numeric

2. Regression Line for Canada per-capita income, etc

Yes, the Regression line was plotted

3. Predicted Salary for Hiring Dataset

For a candidate with 12 years of experience, a 10 in test & 10 in interview the model predicted a salary of \$64629.625

4. Encoding: "State" was encoded using one-hot encoding

The rest of features were scaled to ensure consistent range & prevent larger features from dominating model

Code:

```
from google.colab import files
per_capita_income=files.upload()

from google.colab import files
salary=files.upload()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from scipy import stats
from sklearn import linear_model

df1=pd.read_csv("canada_per_capita_income.csv")
df1.head()

df2=pd.read_csv("salary.csv")
df2.head()
df2.YearsExperience.median()
df2.YearsExperience = df2.YearsExperience.fillna(df2.YearsExperience.median())
df2

plt.xlabel("year")
plt.ylabel("per capita income (US$)")
plt.scatter(df1.year, df1['per capita income (US$)'])

plt.xlabel("YearsExperience")
plt.ylabel("Salary")
plt.scatter(df2.YearsExperience, df2.Salary)

reg1 = linear_model.LinearRegression()
reg1.intercept_
reg1.predict([[2020]])

reg2 = linear_model.LinearRegression()
reg2.fit(df2.drop('Salary', axis='columns'), df2['Salary'])
reg2.coef_
reg2.intercept_
reg2.predict([[12]])

from google.colab import files
hiring=files.upload()

from google.colab import files
companies=files.upload()

df3=pd.read_csv("hiring.csv")
df3.head()

df4=pd.read_csv("1000_Companies.csv")
df4.head()
```

```

df3.isnull().sum()
df4.isnull().sum()

df3_copy = df3.copy()
experience_mapping = {'two': 2, 'three': 3, 'five': 5, 'seven': 7, 'ten': 10, 'eleven': 11}
df3_copy['experience'] = df3_copy['experience'].map(experience_mapping)
median_experience = df3_copy['experience'].median()
df3_copy['experience'] = df3_copy['experience'].fillna(median_experience)
df3_copy
df3_copy['test_score(out of 10)'] = df3_copy['test_score(out of 10)'].fillna(df3_copy['test_score(out of 10)'].mean())
reg3 = linear_model.LinearRegression()
reg3.fit(df3_copy.drop('salary($)', axis='columns'), df3_copy['salary($)'])
reg3.coef_
reg3.intercept_
reg3.predict([[2,9,6]])
reg3.predict([[12,10,10]])

ohe = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
state_encoded = ohe.fit_transform(df4[['State']])
state_encoded_df = pd.DataFrame(state_encoded, columns=ohe.get_feature_names_out(['State']))

df4 = pd.concat([df4, state_encoded_df], axis=1).drop(columns=['State'])
print(df4)
reg4 = linear_model.LinearRegression()
reg4.fit(df4.drop('Profit',axis='columns'),df4.Profit)
print(reg4.coef_)
print(reg4.intercept_)
reg4.predict([[91694.48, 515841.3, 11931.24,0,1,0]])

```


Program 4

Build Logistic Regression Model for a given dataset

Screenshot:

Lab-04 Logistic Regression

Consider a binary classification problem where we want to predict whether a student will pass or fail based on their study hours. The logistic regression model has been trained, & the learned parameters are $a_0 = -5$ (intercept) & $a_1 = 0.8$ (coefficient for study hours).

1. Write logistic regression equation.

$$P(\text{pass}|x) = \frac{1}{1 + e^{-(a_0 + a_1 x)}}$$

Given: $a_0 = -5$, $a_1 = 0.8$

$$P(\text{pass}|x) = \frac{1}{1 + e^{-(-5 + 0.8x)}}$$

Calculate probability for $x = 7$ hours

$$P(\text{pass}|7) = \frac{1}{1 + e^{-(-5 + 0.8 \times 7)}}$$

Step 1: Compute linear combination

$$Z = a_0 + a_1 x$$

$$Z = -5 + 0.8 \times 7 = -5 + 5.6 = 0.6$$

Step 2: Plug Z into logistic function

$$P(\text{pass}|7) = \frac{1}{1 + e^{-0.6}}$$

Step 3: Compute $e^{-0.6}$

$$e^{-0.6} \approx 0.5488$$

Step 4: Calculate probability

$$P(\text{pass}|7) = \frac{1}{1 + 0.5488} = \frac{1}{1.5488} \approx 0.6457$$

So the probability that student will pass approximately is 64.57%

3. Determine the predicted class using a threshold of 0.5

If $P(\text{pass}|x) \geq 0.5$, predict pass
If $P(\text{pass}|x) < 0.5$, predict fail

for $x = 7$ hours

$$P(\text{pass}|7) \approx 0.6457 \geq 0.5$$

Thus, the predicted class is pass.

4. Consider $z = [2, 1, 0]$ for 3 classes. Apply soft max funⁿ to find prob of 3 classes

Solⁿ: $\text{SoftMax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$

$$Z = [2, 1, 0]$$

1. Exponentiate each value

$$e^Z = [e^2, e^1, e^0] = [7.3891, 2.7183, 1]$$

2. Sum up the exponentiated values:

$$\text{Sum} = 7.3891 + 2.7183 + 1 = 11.1074$$

3. Divide each exponentiated value by sum:

$$\text{SoftMax}(Z_0) = \frac{7.3891}{11.1074} \approx 0.6652 \text{ class 0}$$

$$\text{SoftMax}(Z_1) = \frac{2.7183}{11.1074} \approx 0.2447 \text{ class 1}$$

$$\text{SoftMax}(Z_2) = \frac{1}{11.1074} \approx 0.0901 \text{ class 2}$$

For dataset file "HR-Comm-sp.csv":

i) Which variables did you identify as having a direct & clear impact on employee retention? Why?

the key variables impact the employees retention are: Satisfaction level - lower satisfaction level increases

Time spent in company - employees 5+ years tend to leave

Salary - low salaries lead to higher turnover
No. of projects & avg monthly hours - very high or low values affect retention.

ii) What was the accuracy of your logistic regression model? Do you think this is a good accuracy? Why or why not?

The accuracy of logistic regression model is 78.40%. This accuracy is overall good. But the model still needs an improvement. The model captures key factors.

Ans: this

Code:

```
from google.colab import files
hr=files.upload()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from scipy import stats
from sklearn import linear_model
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

df1=pd.read_csv("HR_comma_sep.csv")
df1.head()
df1.isnull().sum()
plt.figure(figsize=(12, 6))
sns.barplot(x='Department', y='left', data=df1)
plt.title('Employee Retention Rate by Department')
plt.xlabel('Department')
plt.ylabel('Proportion of Employees Left')
plt.xticks(rotation=45, ha='right')
plt.show()

ohe = OneHotEncoder(handle_unknown='ignore', sparse_output=False)
department_encoded = ohe.fit_transform(df1[['Department']])
department_encoded_df = pd.DataFrame(department_encoded,
columns=ohe.get_feature_names_out(['Department']))
df1 = pd.concat([df1, department_encoded_df], axis=1)
df1 = df1.drop('Department', axis=1)
ordinal_encoder = OrdinalEncoder(categories=[['low', 'medium', 'high']], dtype=np.int64)
salary_encoded = ordinal_encoder.fit_transform(df1[['salary']])
df1['salary_encoded'] = salary_encoded
df1 = df1.drop('salary', axis=1)
df1.head()

correlation_matrix = df1.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Features')
plt.show()
plt.figure(figsize=(8, 6))
sns.barplot(x='salary_encoded', y='left', data=df1)
plt.title('Impact of Employee Salary on Retention')
plt.xlabel('Salary Level (Encoded)')
plt.ylabel('Proportion of Employees Left')
plt.show()
```

```

df_copy = df1[['number_project', 'average_monthly_hours', 'time_spend_company', 'left', 'salary_encoded',
'satisfaction_level', 'Work_accident']]
df_copy.head()
X = df_copy.drop('left', axis=1)
y = df_copy['left']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of the Logistic Regression model: {accuracy}")

```

```

from google.colab import files
zoodata=files.upload()
zootype=files.upload()

```

```

zoo_data = pd.read_csv('zoo-data.csv')
zoo_class = pd.read_csv('zoo-class-type.csv')
merged_data = pd.merge(zoo_data, zoo_class, left_on='class_type', right_on='Class_Number')
merged_data = merged_data.drop(['Animal_Names', 'Number_Of_Animal_Species_In_Class',
'Class_Number', 'class_type', 'animal_name'], axis=1)
X = merged_data.drop('Class_Type', axis=1)
y = merged_data['Class_Type']
print(merged_data.head())
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=np.unique(y_test))
disp.plot(cmap="Blues", values_format="d")
plt.title("Confusion Matrix")
plt.show()

```

Program 5

Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample.

Screenshot:

Lab 04 Decision trees

Consider the following dataset, calculate entropy & information gain w.r.t to target variable 'classification'. Identify whether the splitting node should be a_1 or a_2 attribute.

Instance	a_1	a_2	classification
1	Hot	High	No
2	Hot	High	No
6	Cool	High	No
7	Hot	High	No
8	Hot	Normal	Yes

$Gain(S, a_1) = 0.6079$ - max gain
 $Gain(S, a_2) = 0.1245$
 $Gain(S, a_3) = 0.4199$

```

graph TD
    A((a1)) -- True --> B[1, 2, 6, 7, 8]
    A -- False --> C[3, 4, 5, 9, 10]
  
```

Attribute a_2
 Values (a_2) = Hot, Cool
 $S_{a_1} = [1, 4, \dots]$
 $Entropy(S_{a_1}) = -1/5 \log_2 1/5 - 4/5 \log_2 4/5 = 0.7219$
 $S_{Hot} = [1, 3, \dots]$ Entropy = $-1/4 \log_2 1/4 - 3/4 \log_2 3/4 = 0.9182$
 $S_{Cool} = [0, 1, \dots]$ Entropy = 0.0
 $Gain(S, a_2) = Entropy(S) - \frac{|S_{Hot}|}{|S|} Entropy(S_{Hot}) - \frac{|S_{Cool}|}{|S|} Entropy(S_{Cool})$
 $= 0.9182 - 4/5 \times 0.9182 - 1/5 \times 0 = 0.1245$

$Gain(S, a_2) = Entropy(S) - 4/5 Entropy(S_{Hot}) - 1/5 Entropy(S_{Cool})$
 $= 0.9182 - 4/5 \times 0.9182 - 1/5 \times 0.0$
 $= 0.3219$

Attribute a_3
 value (a_3) = High, Normal
 $S_{a_1} = [1, 4, \dots]$ Entropy (S_{a_1}) = $-1/5 \log_2 1/5 - 4/5 \log_2 4/5 = 0.7219$
 $S_{High} = [0, 4, \dots]$ Entropy (S_{High}) = 0.0
 $S_{Normal} = [1, 0, \dots]$ Entropy (S_{Normal}) = 0.0
 $Gain(S, a_3) = Entropy(S) - \sum_{v \in \{High, Normal\}} \frac{|S_v|}{|S|} Entropy(S_v)$
 $= 0.9182 - 4/5 \times 0.9182 - 1/5 \times 0.0 = 0.3219$

```

graph TD
    A((a1)) -- True --> B[1, 2, 7, 6, 8]
    A -- False --> C((a3))
    C -- High --> D[1, 2, 6, 7, No]
    C -- Normal --> E[Yes]
  
```

For "iris.csv" dataset
 What was the accuracy score
 a) The accuracy score of the model was 1.00 (100%) meaning the model perfectly classified all test samples.

(b) the confusion matrix shows that the model made no errors as all predictions matched the actual class.

```

[[10 0]
 [0 9]]
  
```

No misclassification occurred since all of diagonal values in the confusion matrix are zero every species were classified correctly without confusion.

For "petrol-consumption.csv" dataset:

- The Regression tree structure splits data to minimize mse with leaf node predicting the avg petrol consumption.
- The most important features for predicting petrol consumption are Petrol tax & population driver license.
- The Regression tree predicts continuous value while a classifier predicts categories. The Regression Tree minimizes variance while the classifier minimizes impurity.

Code:

```
from google.colab import files
iris=files.upload()
df1=pd.read_csv("iris.csv")
df1.head()

df1.isnull().sum()

X = df1.drop('species', axis=1)
y = df1['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(classification_report(y_test, y_pred))
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=y.unique())
plt.show()

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=clf.classes_)
cmap = plt.cm.get_cmap('PuBuGn')
disp.plot(cmap=cmap)
plt.show()

drug=files.upload()
df2=pd.read_csv("drug.csv")
df2.head()
df2.isnull().sum()

label_encoders = {}
for column in df2.columns:
    le = LabelEncoder()
    df2[column] = le.fit_transform(df2[column])
    label_encoders[column] = le
X = df2.drop('Drug', axis=1)
y = df2['Drug']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(classification_report(y_test, y_pred))
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=[str(c) for c in y.unique()])
plt.show()

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=clf.classes_)
```



```
cmap = plt.cm.Blues
disp.plot(cmap=cmap)
plt.show()
```

```
pc=files.upload()
df3=pd.read_csv("petrol_consumption.csv")
df3.head()
df3.isnull().sum()
X = df3.drop('Petrol_Consumption', axis=1)
y = df3['Petrol_Consumption']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
regressor = DecisionTreeRegressor(random_state=42)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse:.2f}')
print(f'Root Mean Squared Error: {rmse:.2f}')
print(f'Mean Absolute Error: {mae:.2f}')
print(f'R-squared: {r2:.2f}')
plt.figure(figsize=(30, 30))
plot_tree(regressor, filled=True, feature_names=X.columns, fontsize=10)
plt.show()
```


Program 6

Build KNN Classification model for a given dataset.

Screenshot:

Lab-005

PAGE NO: 15
DATE: 07/02/25

KNN classification Model

* Consider the following dataset. for $K=3$ & test data $(X, 35, 100)$ as (Person, Age, Salary) solve using KNN classifier model & predict the target.

Person	Age	Salary	Target
A	18	50	N
B	23	55	N
C	24	70	N
D	41	60	Y
E	43	70	Y
F	38	40	Y
X	35	100	?

Distance

Distance	Rank
$\sqrt{(35-18)^2 + (100-50)^2} = 52.81$	5
$\sqrt{(35-23)^2 + (100-55)^2} = 46.57$	4
$\sqrt{(35-24)^2 + (100-70)^2} = 31.95$	2
$\sqrt{(35-41)^2 + (100-60)^2} = 40.45$	3
$\sqrt{(35-43)^2 + (100-70)^2} = 31.05$	1
$\sqrt{(35-38)^2 + (100-40)^2} = 61.07$	6

~~$\sqrt{(35-35)^2}$~~

K=1 target = Y
K=2 target = N
K=3 target = Y
target for Person X = Y

∴ for iris dataset

PAGE NO: 16
DATE:

* How to choose the K value? Demonstrate using accuracy rate & error rate.

Soln: Best K value in iris data set =

K values from 1 to 20 are tested for accuracy and error rate the K value of 3 was chosen because of high accuracy & low error rate.

2. for diabetes dataset

* What is the purpose of feature scaling? how to perform it?

Soln: Feature scaling in diabetes to ensure that all the features like glucose, age, insulin are on the same scale so that KNN does not get biased by large numerical values.

Pratibha

Code:

```
from google.colab import files
iris=files.upload()
df1=pd.read_csv("iris (2).csv")
df1.head()
df1.isnull().sum()
X = df1.drop('species', axis=1)
y = df1['species']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
best_k = 1
best_accuracy = 0
for k in range(1, 11):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_k = k
print(f'Best k value: {best_k}')
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=knn.classes_, yticklabels=knn.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

diabetes=files.upload()
df2=pd.read_csv("diabetes.csv")
df2.head()
df2.isnull().sum()
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df2.drop('Outcome', axis=1))
X_train, X_test, y_train, y_test = train_test_split(X_scaled, df2['Outcome'], test_size=0.2, random_state=42)
best_k = 1
best_accuracy = 0
for k in range(1, 11):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f'Accuracy for k={k}: {accuracy}')
    if accuracy > best_accuracy:
```

```

    best_accuracy = accuracy
    best_k = k
print(f'Best k value: {best_k}')

knn = KNeighborsClassifier(n_neighbors=best_k)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

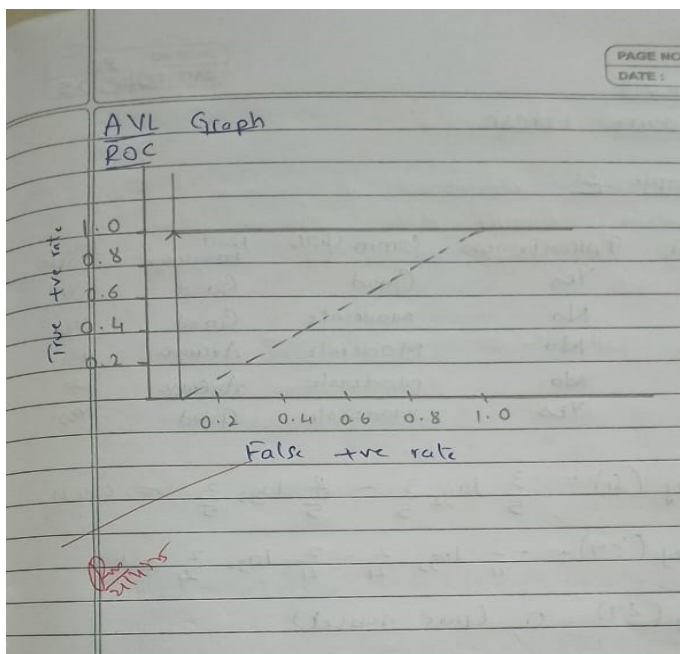
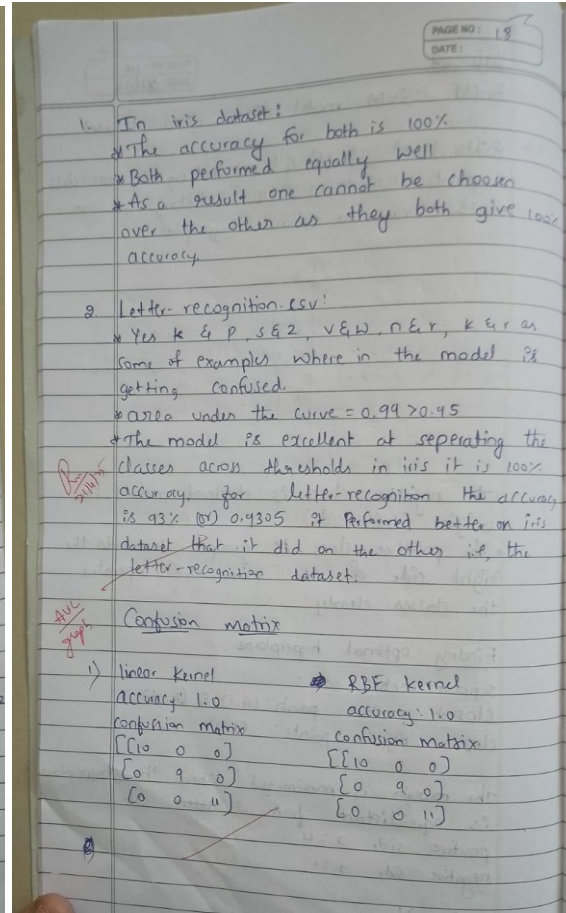
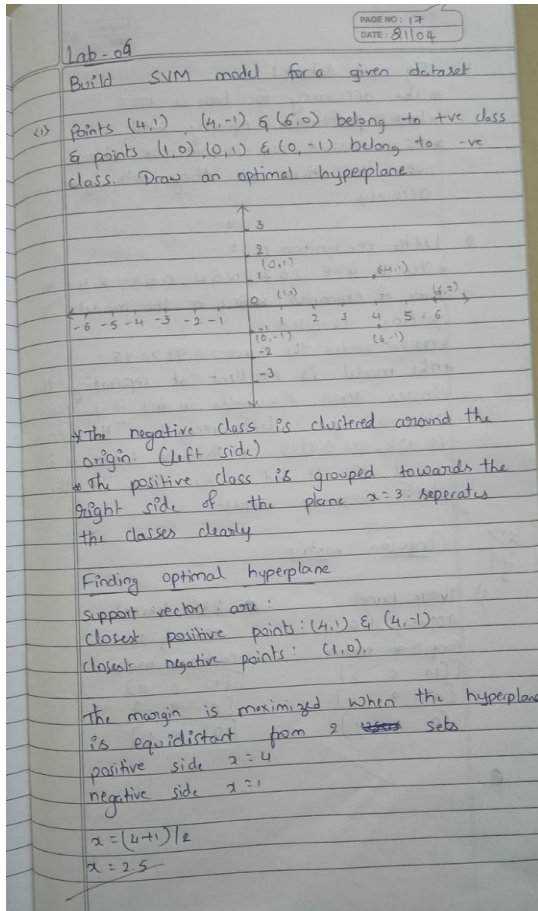
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
heart=files.upload()
df3=pd.read_csv("heart.csv")
df3.head()
df3.isnull().sum()
X = df3.drop('target', axis=1)
y = df3['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
best_k = 1
best_accuracy = 0
for k in range(1, 11):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f'Accuracy for k={k}: {accuracy}, Error Rate for k={k}: {1-accuracy}')
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_k = k
print(f'Best k value: {best_k}')
knn = KNeighborsClassifier(n_neighbors=optimal_k)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
print(cm)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=knn.classes_, yticklabels=knn.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

```

Program 7

Build Support vector machine model for a given dataset

Screenshot:



Code:

```
from google.colab import files
iris=files.upload()
df1=pd.read_csv("iris (1).csv")
df1.head()
X = df1.drop('species', axis=1)
y = df1['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rbf_svm = SVC(kernel='rbf')
rbf_svm.fit(X_train, y_train)
rbf_y_pred = rbf_svm.predict(X_test)
print("RBF Kernel SVM:")
print("Accuracy:", accuracy_score(y_test, rbf_y_pred))
cm = confusion_matrix(y_test, rbf_y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap="Blues")
plt.title('Confusion Matrix for RBF Kernel SVM')
plt.xlabel('Predicted')

plt.ylabel('True')
plt.show()
print(classification_report(y_test, rbf_y_pred))
linear_svm = SVC(kernel='linear')
linear_svm.fit(X_train, y_train)
linear_y_pred = linear_svm.predict(X_test)
print("\nLinear Kernel SVM:")
print("Accuracy:", accuracy_score(y_test, linear_y_pred))
cm = confusion_matrix(y_test, linear_y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap="Blues")
plt.title('Confusion Matrix for Linear Kernel SVM')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
print(classification_report(y_test, linear_y_pred))
letter=files.upload()
df2=pd.read_csv("letter-recognition.csv")
df2.head()
X = df2.drop('letter', axis=1)
y = df2['letter']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
svm_classifier = SVC(kernel='linear', probability=True)
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(10,10))
sns.heatmap(cm, annot=True, fmt='d', cmap="Blues")
plt.title('Confusion Matrix for SVM')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
lb = LabelBinarizer()
lb.fit(y_test)
y_test_lb = lb.transform(y_test)
```

```

y_pred_prob = svm_classifier.predict_proba(X_test)
fpr = {}
tpr = {}
thresh = {}
roc_auc = dict()
n_class = y_test_lb.shape[1]
for i in range(n_class):
    fpr[i], tpr[i], thresh[i] = roc_curve(y_test_lb[:,i], y_pred_prob[:,i])
    roc_auc[i] = auc(fpr[i], tpr[i])
plt.plot(fpr[0], tpr[0], linestyle='--', color='orange', label='SVM (AUC = %0.2f)' % roc_auc[0])
plt.title('ROC Curve for Class 0')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.legend(loc='best')
plt.show()
print(f"AUC score for class 0: {roc_auc[0]}")

```


Program 8

Implement Random forest ensemble method on a given dataset

Screenshot:

PAGE NO: 20
DATE: 05/05/25

Lab-07
Random Forest

1) Sample S₁

S _N	CGPA	Interactiveness	Comm. Skills	Pract. knowledge	Job offer
1	≥ 9	Yes	Good	Good	Yes
2	< 9	No	Moderate	Good	Yes
3	≥ 9	No	Moderate	Average	No
4	≥ 9	No	Moderate	Average	No
5	≥ 9	Yes	Moderate	Good	Yes

$$\text{Entropy}(S_1) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.971$$

$$\text{Entropy}(\geq 9) = -\frac{2}{4} \log_2 \frac{2}{4} - \frac{2}{4} \log_2 \frac{2}{4} = 1$$

$$\text{Entropy}(< 9) = 0 \text{ (pure subset)}$$

$$\text{IG}(\text{CGPA}) = 0.971 - \left(\frac{4}{5} \times 1 + \frac{1}{5} \times 0 \right) = 0.971 - 0.8 = 0.171$$

Decision Tree

```

graph TD
    A[CGPA?] -->|< 9| B[Job offers: Yes]
    A -->|≥ 9| C[Job offers: Yes]
    C -->|Interactiveness: Yes| D[Job offers: Yes]
    C -->|No| E[Job offers: No]
  
```

PAGE NO: 21
DATE:

2. Sample S₂

S _N	CGPA	Interactiveness	Comm. skills	Practical knowledge	Job offers
1	< 9	No	Moderate	Good	Yes
2	≥ 9	No	Moderate	Avg	No
3	≥ 9	No	Moderate	Avg	No
4	≥ 9	Yes	Moderate	Good	Yes
5	≥ 9	Yes	Moderate	Good	Yes

$$\text{Entropy}(S_2) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.971$$

$$\text{Entropy}(\text{Interactiveness}) = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.918$$

$$\text{IG}(\text{Interactiveness}) = 0.971 - \left(\frac{2}{5} \times 0 + \frac{3}{5} \times 0.918 \right) = 0.971 - 0.551 = 0.42$$

Decision Tree

```

graph TD
    A[Interactiveness] -->|No| B[Practical: Good knowledge]
    A -->|Yes| C[Job offers: Yes]
    B -->|Job offers: Yes| D[Job offers: Yes]
    B -->|Job offers: No| E[Job offers: No]
  
```

What is the best acc. Acc. of confusion matrix of classifier you observed & using how many trees?

Best accuracy: 1.000 With 1 tree

Confusion matrix:

10	0	0
0	9	0
0	0	11

Code:

```

from google.colab import files
iris=files.upload()
df1=pd.read_csv("iris (4).csv")
df1.head()
X = df1.drop('species', axis=1)
y = df1['species']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
rf_classifier = RandomForestClassifier(random_state=0)
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
default_accuracy = accuracy_score(y_test, y_pred)
  
```

```

print(f'Accuracy with default n_estimators: {default_accuracy}')
best_accuracy = 0
best_n_estimators = 0
for n_estimators in range(1, 101):
    rf_classifier = RandomForestClassifier(n_estimators=n_estimators, random_state=0)
    rf_classifier.fit(X_train, y_train)
    y_pred = rf_classifier.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    if accuracy > best_accuracy:

        best_accuracy = accuracy
        best_n_estimators = n_estimators
print(f'\nBest accuracy: {best_accuracy} achieved with n_estimators = {best_n_estimators}')
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=np.unique(y_test), yticklabels=np.unique(y_test))
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

```

Program 9

Implement Boosting ensemble method on a given dataset

Screenshot:

Lab-08
AdaBoost Algorithm

Considering AdaBoost Algorithm. For the following sample data, Show the Decision stump calculations steps for the attribute CGPA

CGPA	Interactiveness	Practical Knowledge	Comm Skills	Job Profile
≥ 9	Yes	Good	Good	Yes
< 9	No	Good	Moderate	Yes
≥ 9	No	Avg	Moderate	No
< 9	No	Avg	Good	No
≥ 9	Yes	Good	Moderate	Yes
≥ 9	Yes	Good	Moderate	Yes

→ Initial weight (w_i) = $\frac{1}{6}$

Decision stump on CGPA
if CGPA $\geq 9 \rightarrow$ Yes
if CGPA $< 9 \rightarrow$ No

Weighted error calculation
 $E = w_2 + w_5 = \frac{1}{6} + \frac{1}{6} = \frac{2}{6} = 0.333$

Computing model weight (α)
 $\alpha = \frac{1}{2} \ln \left(\frac{1-E}{E} \right)$
 $= \frac{1}{2} \ln \left(\frac{0.667}{0.333} \right)$
 $= \frac{1}{2} \ln(2)$
 ≈ 0.3466

PAGE NO: _____
DATE: _____

$$Z_{CGPA} = \frac{1}{6} \times 4 \times e^{-0.3466} + \frac{1}{6} \times 2 \times e^{0.3466}$$
$$= 0.9428$$

2. Best accuracy score & Confusion matrix of classifier & how many trees?

Best accuracy score = 0.8385
With N estimation = 80

Confusion matrix = $\begin{bmatrix} 7130 & 284 \\ 1343 & 1012 \end{bmatrix}$

Code:

```
from google.colab import files
income=files.upload()
df1=pd.read_csv("income.csv")
df1.head()
X = df1.drop('income_level', axis=1)

y = df1['income_level']
X = pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
abc = AdaBoostClassifier(n_estimators=10, random_state=42)
abc.fit(X_train, y_train)
y_pred = abc.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Initial AdaBoost accuracy (10 trees): {accuracy}')
param_grid = {'n_estimators': [50, 100, 150, 200]}
grid_search = GridSearchCV(AdaBoostClassifier(random_state=42), param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
print(f'Best parameters: {grid_search.best_params_}')
print(f'Best cross-validation score: {grid_search.best_score_}')
best_abc = grid_search.best_estimator_
y_pred_best = best_abc.predict(X_test)
best_accuracy = accuracy_score(y_test, y_pred_best)
print(f'Accuracy of the best model on the test set: {best_accuracy}')
cm = confusion_matrix(y_test, y_pred_best)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['<=50K', '>50K'], yticklabels=['<=50K', '>50K'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Program 10

Build k-Means algorithm to cluster a set of data stored in a .CSV file

Screenshot:

Lab-07		DATE: 12/5/25	DATE:
Build k-means algorithm to cluster a set of data stored in a .CSV file.			
→ Compute 2 clusters using k-means algo for clustering initial cluster centres are (1,1) & (5,7). Execute for 2 iterations			
Iteration 1:			
record num	cluster 1 (C1)	cluster 2 (C2)	assign to cluster
R1 (1,1)	0	7.21	C1
R2 (1.5,2)	1.12	6.12	C1
R3 (3,4)	3.61	3.61	C1
R4 (5,7)	7.21	0.0	C2
R5 (3.5,5)	4.12	2.5	C2
R6 (4.5,5)	5.31	2.06	C2
R7 (3.5,4.5)	4.30	2.92	C2
New centroids are:			
$C1 = \frac{5.5}{3}, \frac{7.0}{3}$		$C2 = \frac{16.5}{4}, \frac{21.5}{4}$	
$C1 = 1.83, 2.33$		$C2 = 4.12, 5.37$	
Iteration 2: (1.83, 2.33) (4.12, 5.37)			
record num	Closest to C1	Closest to C2	assign to cluster
R1	1.57	5.37	C1
R2	0.47	4.27	C1
R3	2.04	1.77	C2
R4	5.64	1.85	C2
R5	3.15	0.72	C2
R6	2.78	0.52	C2
R7	2.74	1.07	C2

new clusters are:
 $C1 (R1, R2)$ & $C2 (R3, R4, R5, R6, R7)$

new centroids are:
 $C1 = \frac{2.5}{2}, \frac{3}{2}$ $C2 = \frac{19.5}{5}, \frac{25.5}{5}$
 $C1 = 1.25, 1.5$ $C2 = 3.9, 5.1$

1. For "iris.csv" dataset:
The elbow plot (inertia vs k) shows a sharp "elbow" at $k=3$, indicating that 3 clusters is the optimal choice for the "iris" dataset.

Code:

```
from google.colab import files
iris=files.upload()
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from scipy import stats
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

df1=pd.read_csv("iris (4).csv")
df1.head()
df = df1.drop(['sepal_length','sepal_width','species'],axis=1)
```

```
scaler = StandardScaler()
scaled_df = scaler.fit_transform(df)
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(scaled_df)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state=0)
pred_y = kmeans.fit_predict(scaled_df)
df['cluster'] = pred_y
plt.scatter(df['petal_length'], df['petal_width'], c=df['cluster'])
plt.title('Clusters of Iris Flowers')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()
```


Program 11

Implement Dimensionality reduction using Principal Component Analysis (PCA) method.

Screenshot:

Lab-10
Principal Component Analysis (PCA)

→ Reduce the dimension from 2 to 1 using PCA, compute 1st principle component

data matrix:

x_1	4	8	13	7
x_2	11	4	5	14

$\lambda_1 = 30.3849$ $e_1 = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$ $e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$
 $\lambda_2 = 6.6151$

Mean of $x_1 = \frac{4+8+13+7}{4} = 8$
Mean of $x_2 = \frac{11+4+5+14}{4} = 8.5$

X-centred - $X - \text{Mean} = \begin{bmatrix} -4 & 0 & 5 & -1 \\ 2.5 & -4.5 & -3.5 & 5.5 \end{bmatrix}$

Since λ_1 is larger, e_1 is 1st principle component

$e_1 = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix}$

Let $z_i = e_i - \bar{z}_i$

$z_1(-4, 0.5) = 0.5574(-4) + (-0.8303)(2.5) = -4.30535$
 $z_2(0, -4.5) = 0.5574(0) + (-0.8303)(-4.5) = 3.73635$
 $z_3(5, -3.5) = 0.5574(5) + (-0.8303)(-3.5) = 5.69305$
 $z_4(-1, 5.5) = 0.5574(-1) + (-0.8303)(5.5) = -5.12405$

$z = \begin{bmatrix} -4.305 \\ 3.736 \\ 5.693 \\ -5.124 \end{bmatrix}$

1. For "heart.csv" dataset:

* Accuracy before PCA:
Logistic regression: 0.9016
SVM: 0.8525
Random Forest: 0.8361

* Accuracy after PCA:
Logistic regression: 0.8689
SVM: 0.8689
Random Forest: 0.8852

Code:

```
from google.colab import files
heart=files.upload()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from scipy import stats
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA

df1=pd.read_csv("heart(1).csv")
df1.head()
text_cols = df1.select_dtypes(include=['object']).columns
label_encoder = LabelEncoder()
for col in text_cols:
    df1[col] = label_encoder.fit_transform(df1[col])
print(df1.head())
X = df1.drop('HeartDisease', axis=1)
y = df1['HeartDisease']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Support Vector Machine
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)
svm_predictions = svm_model.predict(X_test)
svm_accuracy = accuracy_score(y_test, svm_predictions)
print(f"SVM Accuracy: {svm_accuracy}")

# Logistic Regression
lr_model = LogisticRegression(random_state=42)
```

```

lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)
lr_accuracy = accuracy_score(y_test, lr_predictions)
print(f"Logistic Regression Accuracy: {lr_accuracy}")

# Random Forest
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)
print(f"Random Forest Accuracy: {rf_accuracy}")

models = {
    "SVM": svm_accuracy,
    "Logistic Regression": lr_accuracy,
    "Random Forest": rf_accuracy
}

best_model = max(models, key=models.get)
print(f"\nBest Model: {best_model} with accuracy {models[best_model]}")
pca = PCA(n_components=0.95)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

svm_model_pca = SVC(kernel='linear', random_state=42)
svm_model_pca.fit(X_train_pca, y_train)
svm_predictions_pca = svm_model_pca.predict(X_test_pca)
svm_accuracy_pca = accuracy_score(y_test, svm_predictions_pca)
print(f"SVM Accuracy (with PCA): {svm_accuracy_pca}")

lr_model_pca = LogisticRegression(random_state=42)
lr_model_pca.fit(X_train_pca, y_train)
lr_predictions_pca = lr_model_pca.predict(X_test_pca)
lr_accuracy_pca = accuracy_score(y_test, lr_predictions_pca)
print(f"Logistic Regression Accuracy (with PCA): {lr_accuracy_pca}")

rf_model_pca = RandomForestClassifier(random_state=42)
rf_model_pca.fit(X_train_pca, y_train)
rf_predictions_pca = rf_model_pca.predict(X_test_pca)
rf_accuracy_pca = accuracy_score(y_test, rf_predictions_pca)
print(f"Random Forest Accuracy (with PCA): {rf_accuracy_pca}")

models_pca = {
    "SVM": svm_accuracy_pca,
    "Logistic Regression": lr_accuracy_pca,
    "Random Forest": rf_accuracy_pca
}

best_model_pca = max(models_pca, key=models_pca.get)
print(f"\nBest Model (with PCA): {best_model_pca} with accuracy {models_pca[best_model_pca]}")

```