Introduction to AI & ML

Lab Manual

Department of Computer Science and Engineering
The NorthCap University, Gurugram

Introduction to AI &ML Lab Manual

CSL 236

Department of Computer Science and Engineering

The NorthCap University, Gurugram- 122001, India

Session 2022-23

Published by:

School of Engineering and Technology

Department of Computer Science & Engineering

The NorthCap University Gurugram

• **Laboratory Manual is for Internal Circulation only**

Copying or facilitating copying of lab work comes under cheating and is considered as use of unfair means. Students indulging in copying or facilitating copying shall be awarded zero marks for that particular experiment. Frequent cases of copying may lead to disciplinary action. Attendance in lab classes is mandatory.

Labs are open up to 7 PM upon request. Students are encouraged to make full use of labs beyond normal lab hours.

PREFACE

Machine Learning Lab Manual is designed to meet the course and program requirements of NCU curriculum for B.Tech. III year students of CSE branch. The concept of the lab work is to give brief practical experience for basic lab skills to students. It provides the space and scope for self-study so that students can come up with new and creative ideas.

The Lab manual is written on the basis of "teach yourself pattern" and expected that students who come with proper preparation should be able to perform the experiments without any difficulty. Brief introduction to each experiment with information about self-study material is provided. The pre-requisite is having a basic working knowledge of Python. The laboratory exercises will include familiarization with data pre-processing techniques for ML like handling missing data, duplicate data, outliers feature scaling and encoding. Feature Selection and Dimensionality Reduction are included to enhance the performance and reduce the computational time. Various ML classification and regression techniques are taught. Students would learn the algorithms pertaining to these and implement the same using a high-level language, i.e. Python. Students are expected to come thoroughly prepared for the lab. General disciplines, safety guidelines and report writing are also discussed.

The lab manual is a part of curriculum for the The NorthCap University, Gurugram. Teacher's copy of the experimental results and answer for the questions are available as sample guidelines.

We hope that lab manual would be useful to students of CSE, IT, ECE and BSc branches and author requests the readers to kindly forward their suggestions / constructive criticism for further improvement of the work book.

Author expresses deep gratitude to Members, Governing Body-NCU for encouragement and motivation.

Authors
The NorthCap University
Gurugram, India

CONTENTS

S.N.	Details	Page No.
	Syllabus	6
1	**Introduction**	9
2	**Lab Requirement**	10
3	**General Instructions**	11
4	**List of Experiments**	13
5	**List of Flip Assignment**	14
6	**List of Projects**	15
7	**Rubrics**	16
8	**Annexure 1 (Format of Lab Report)**	18

SYLLABUS

1. Department:	Department of Computer Science and Engineering		
2. Course Name: Machine Learning	**3. Course Code**	**4. L-T-P**	**5. Credits**
	CSL236	3-0-2	4
6. Type of Course (Check one):	Program Core ☐	Program Elective ☐	Open Elective ☐
7. Pre-requisite(s), if any: Introduction to AI and ML			
8. Frequency of offering (check one): Odd ☐ Even ☐ Either semester ☐ Every semester ☐			

9. Brief Syllabus:

Introduction to artificial intelligence, History of AI, Proposing and evaluating AI application, Preprocessing and Feature Engineering, Case study: Exploratory Analysis of Delhi Pollution, Simple Linear Regression, Multiple Regression, Polynomial Regression, Support Vector Regression SVR, Decision Tree Regression, Random Forest Regression, Logistic Regression, K Nearest Neighbors, Support Vector Machine, Kernel SVM, Naïve Bayes, Decision Trees Classification, Random Forest Classification, Basic Terminologies: Overfitting, Underfitting, Bias and Variance model, Bootstrapping, Cross-Validation and Resampling Methods, Performance Measures: Confusion matrix, ROC. Comparing two classification Algorithms: McNamara's Test, paired t-test.

Total lecture, Tutorial and Practical Hours for this course (Take 15 teaching weeks per semester): 75 hours
The class size is maximum 30 learners

	Practice	
Lectures: 40 hours	**Tutorials:** 0 hours	**Lab Work: 35** hours

10. Course Outcomes (COs)

On successful completion of this course students will be able to:

CO 1	Understand and implement the preprocessing of the data to be used for machine learning models.
CO 2	Understand the strengths and limitations of various ML algorithms.

CO 3	Understand why models degrade and how to maintain them.
CO 4	Implement and use model grading metrics.
CO 5	Apply ML techniques and technologies to solve real world business problems.

11. UNIT WISE DETAILS No. of Units: 5

Unit Number: 1	**Title: Introduction to AI and ML**	**No. of hours: 4**
Content Summary: Introduction to artificial intelligence, History of AI, Overview of machine learning, techniques in machine learning, deep learning, differences between deep learning, machine learning and AI, different applications of machine learning, different types of data.		
Unit Number: 2	**Data preprocessing and engineering**	**No. of hours:10**
Content Summary: Introduction to Data Preprocessing, different preprocessing techniques, data cleaning, data transformation: standardization and normalization, data smoothing, dimensionality reduction, different encoding schemes for categorical and numerical features.		
Unit Number: 3	**Title: Regression Techniques**	**No. of hours:12**
Content Summary: Simple Linear Regression, Multiple Regression, Polynomial Regression, Support Vector Regression SVR, Decision Tree Regression, Random Forest Regression		
Unit Number: 4	**Title: Classification algorithm techniques**	**No. of hours:12**
Logistic Regression, K Nearest Neighbors, Support Vector Machine, Kernel SVM, Naïve Bayes, Decision Trees Classification, Random Forest Classification		
Unit Number: 5	**Title: Analysis of various algorithms**	**No. of hours:7**
Basic Terminologies: Over fitting, Under fitting, Bias and Variance model, Bootstrapping, Cross-Validation and Resampling Methods, Performance Measures: Confusion matrix, ROC.		

12. Brief Description of Self-learning components by students (through books/resource material etc.):

Data-preprocessing techniques

13. Advance Learning Components:

Probability and Statistics and Linear Algebra

14. Books Recommended :

Text Books:

1. Michael Bowles, “Machine Learning in Python ” Wiley, Third Edition, 2019
2. Jiawei Han, Micheline Kamber and Jian Pei. Data Mining: Concepts and Techniques, 3rd ed.

Reference Books:

1. Ian H. Witten & Eibe Frank., “Data Mining Practical Machine Learning Tools and Techniques”, Morgan Kauffmann Publishers, Second Edition, 2020
2. Ethem Alpaydin, “Introduction to Machine Learning”, MIT Press, Third Edition, 2015
3. Tom Mitchell. Machine Learning. Mc Graw Hill

Reference Websites: (NPTEL, Swayam, Coursera, Edx, Udemy, LMS, official documentation weblink)

- **https://nculms.ncuindia.edu/**
- **https://www.simplilearn.com/big-data-and-analytics/machine-learning-certification-training-course**
- **https://www.coursera.org/learn/machine-learning**

Faculty teaching the Course:

Dr. Meghna Sharma

Dr. Vidhi Khanduja

Dr. Sujata

Dr. Shraddha Arora

Dr. Meenakshi

Ms. Ruchika Lalit

Dr. Anshul Bhatia

Dr. Nidhi Malik

1. INTRODUCTION

That 'learning is a continuous process' cannot be over emphasized. The theoretical knowledge gained during lecture sessions need to be strengthened through practical experimentation. Thus, practical makes an integral part of a learning process.

OBJECTIVES:
The purpose of conducting experiments can be stated as follows:

- To familiarize the students with the basic concepts of Machine Learning like supervised, unsupervised and reinforcement learning.
- The lab sessions will be based on exploring the concepts discussed in class.
- Learning and understanding Data Preprocessing techniques.
- Learning and understanding regression and classification problems and algorithms.
- Learning and understanding Feature selection and Dimensionality Reduction.
- Learning and understanding performance metrics.
- Hands on experience

2. LAB REQUIREMENTS

S.No.	Requirements	Details
1	**Software Requirements**	Python 3.
2	**Operating System**	Windows(64-bit), Linux
3	**Hardware Requirements**	8 GB RAM (Recommended) 2.60 GHz (Recommended)
4	**Required Bandwidth**	NA

3. GENERAL INSTRUCTIONS

3.1 General discipline in the lab

- Students must turn up in time and contact concerned faculty for the experiment they are supposed to perform.
- Students will not be allowed to enter late in the lab.

- Students will not leave the class till the period is over.
- Students should come prepared for their experiment.
- Experimental results should be entered in the lab report format and certified/signed by concerned faculty/ lab Instructor.
- Students must get the connection of the hardware setup verified before switching on the power supply.
- Students should maintain silence while performing the experiments. If any necessity arises for discussion amongst them, they should discuss with a very low pitch without disturbing the adjacent groups.
- Violating the above code of conduct may attract disciplinary action.
- Damaging lab equipment or removing any component from the lab may invite penalties and strict disciplinary action.

3.2 Attendance

- Attendance in the lab class is compulsory.
- Students should not attend a different lab group/section other than the one assigned at the beginning of the session.
- On account of illness or some family problems, if a student misses his/her lab classes, he/she may be assigned a different group to make up the losses in consultation with the concerned faculty / lab instructor. Or he/she may work in the lab during spare/extra hours to complete the experiment. No attendance will be granted for such case**.**

3.3 Preparation and Performance

- Students should come to the lab thoroughly prepared on the experiments they are assigned to perform on that day. Brief introduction to each experiment with information about self study reference is provided on LMS.
- Students must bring the lab report during each practical class with written records of the last experiments performed complete in all respect.
- Each student is required to write a complete report of the experiment he has performed and bring to lab class for evaluation in the next working lab. Sufficient space in work book is provided for independent writing of theory, observation, calculation and conclusion.
- Students should follow the Zero tolerance policy for copying / plagiarism. Zero marks will be awarded if found copied. If caught further, it will lead to disciplinary action.
- Refer **Annexure 1** for Lab Report Format

4. LIST OF EXPERIMENTS

Sr. No.	Title of the Experiment	Software used	Unit Covered	CO Covered	Time Required
1	To introduce various python libraries used for machine learning.	Python (Jupyter)	1	CO1	4 hours
2	To apply various data pre-processing techniques used for effective machine learning on the given dataset.	Python (Jupyter)	1	CO1	2 hours
3	To apply feature encoding schemes such as label encoder and onehotencoder.	Python (Jupyter)	1	CO1	3 hours
4	To apply different feature selection techniques in machine learning.	Python (Jupyter)	1	CO1	3 hours
5	To apply PCA as feature reduction technique on IRIS dataset.	Python (Jupyter)	2	CO1	2 hours
6	To apply Simple Linear Regression on the given dataset.	Python (Jupyter)	2	CO2, CO3, CO4	2 hours
7	To apply multiple linear regression on any regression dataset.	Python (Jupyter)	2	CO2, CO3, CO4	3 hours
8	To apply Polynomial Linear Regression on the given dataset.	Python (Jupyter)	2	CO2, CO3, CO4	3 hours
9	To solve classification problems using Logistic Regression.	Python (Jupyter)	3	CO2, CO3, CO4	2 hours
10	To solve classification problems using KNN classification.	Python (Jupyter)	3	CO2, CO3, CO4	2 hours
11	To solve classification problems using Naïve Bayes.	Python (Jupyter)	4	CO2, CO3, CO4	2 hours
12	To apply Support Vector Machines (SVM) on classification problems.	Python (Jupyter)	4	CO2, CO3, CO4	3 hours
13	To apply Decision Trees for classification problems.	Python (Jupyter)	4	CO2, CO3, CO4	3 hours
Value Added Experiments					
14	Build a ML model from scratch using data-preprocessing and regression algorithms and	Python (Jupyter)	1,2,3,4	CO1,CO2, CO3,CO4,CO5	5 hours

	calculating various performance metrics.				
15	Build a ML model from scratch using data-preprocessing and classification algorithms and calculating various performance metrics.	Python (Jupyter)	1,2,3,4	CO1,CO2, CO3,CO4,CO5	5 hours

5. LIST OF FLIP EXPERIMENTS

5.1 Project – Dimensionality reduction using LDA.
5.2 Competition on Kaggle

6. LIST OF PROJECTS

1. Titanic Challenge: The sinking of the Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the widely considered "unsinkable" RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren't enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew. While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others. In this project, the students need to build a predictive model that answers the question: "what sorts of people were more likely to survive?" using passenger data (ie name, age, gender, socio-economic class, etc).
2. House Price Prediction Using Advanced Regression Techniques: Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But Kaggle's advanced house price prediction dataset proves that much more influences price negotiations than the number of bedrooms or a white-picket fence. With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this dataset can be used to predict the final price of each home.
3. Mechanism of Action (MoA) Prediction: Mechanism of action means the biochemical interactions through which a drug generates its pharmacological effect. If we know a disease affects some particular receptor or downstream set of cell activity, we can develop drugs faster if we can predict how cells and genes affect various receptor sites. Using a dataset that combines gene expression and cell viability data in addition to the MoA annotations of more than 5,000 drugs. In this, each drug was tested under two dose (cp_dose) and three times (cp_time). So, six samples basically correspond to one drug. We need to train a model that classifies drugs based on their biological activity. This problem is a multi-label classification, which means we have multiple

targets (not multiple classes). In this project, perform explanatory data analysis and then train a model using deep neural networks with Keras.

7. RUBRICS

Marks Distribution	
Continuous Evaluation (30 Marks)	**Project Evaluations (40 Marks)**
Each experiment shall be evaluated for 10 marks and at the end of the semester proportional marks shall be awarded out of total 10.	Both the projects shall be evaluated for 40 marks each and at the end of the semester viva will be conducted related to the projects as well as concepts learned in labs and this component carries 40 marks.
15 Marks: For viva **5 Marks:** MOOC	

Annexure 1

Introduction to AI and ML

(CSL 236)

Lab Practical Report

Faculty name: Ankita Gupta

Student name: Piyush Gambhir

Roll No.: 21CSU349

Semester: V

Group: AIMLB (AL-3)

Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2022-23

NAME : Piyush Gambhir

CLASS : AIML-B (AL-3)

ROLL NO: 21CSU349

Sr. No.	Title of Experiment	Date of Experiment	Date of submission	CO Covered	Sign
1	To introduce various Python libraries used for machine learning.	11-08-23	18.08.2023	CO1	
2	To apply various data pre-processing techniques for effective machine learning on the given dataset.	18-08-23	18.08.2023	CO1	
3	To apply feature encoding schemes such as label encoder and one-hot encoder.	25-08-23	25.08.2023	CO1	
4	To apply different feature selection techniques in machine learning.	08-09-23	08.09.2023	CO1	
5	To apply PCA as a feature reduction technique on the IRIS dataset.	01-09-23	01.09.2023	CO1	27/10/23
6	To apply Simple Linear Regression on the given dataset.	22-09-23	22.09.2023	CO2, CO3, CO4	
7	To apply multiple linear regression on any regression dataset.	29-09-23	29.09.2023	CO2, CO3, CO4	
8	To apply Polynomial Linear Regression on the given dataset.	20-10-23	10.11.2023	CO2, CO3, CO4	10/11/23
9	To solve classification problems using Logistic Regression.	20-10-23	10.11.2023	CO2, CO3, CO4	10/11/23
10	To solve classification problems using KNN classification.	10-10-23	10.11.2023	CO2, CO3, CO4	

11	To solve classification problems using Naïve Bayes.	10-11-23	10.11.2023	CO2, CO3, CO4	Ankita 10/11/23
12	To apply Support Vector Machines (SVM) to classification problems.	17-11-23	20.11.2023	CO2, CO3, CO4	Ankita 20/11/23
13	To apply Decision Trees for classification problems.	17-11-23	20.11.2023	CO2, CO3, CO4	

EXPERIMENT NO. 1

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 11.08.2023
Faculty Signature:
Marks:

Objective(s):

- To understand the basic libraries of python.
- To differentiate between numpys and pandas.

Outcome:

Students will be familiarized with handling dataset using python basic libraries and applying various operations on dataset using these.

Problem Statement:

To introduce various libraries of python used for machine learning.

Background Study:

Basic libraries of python are necessary to import datasets and applying various data pre-processing and machine learning techniques on them.

Question Bank:

1. Pandas for Reading Data:

   - To read data from the internet, you can use pandas' **read_csv()** method with a URL as the file path.
   - To read data from your system, use **read_csv()** or other relevant **read_*()** functions, specifying the local file path.

2. Pandas DataFrame to NumPy and Vice Versa:

   - To convert a Pandas DataFrame to a NumPy array, use the **.to_numpy()** or **.values** attribute of the DataFrame.
   - To convert a NumPy array to a Pandas DataFrame, use **pd.DataFrame()** with the NumPy array as the input.

3. Accessing Rows and Columns with **loc** and **iloc**:

   - **loc** is used for label-based indexing. You can access rows and columns by their labels (names).
   - **iloc** is used for integer-based indexing. You can access rows and columns by their integer positions (0-based).

4. Feature Selection vs. Dimensionality Reduction:

   - Feature Selection: Selecting a subset of relevant features from the original set while preserving their original meaning. It reduces the number of features.
   - Dimensionality Reduction: Transforming the original feature space into a lower-dimensional space while retaining as much information as possible. It creates new features.

5. Advantages of Wrapper Methods over Filter Methods:

   - Wrapper methods use a machine learning model's performance as a criterion for feature selection.
   - They consider feature interactions and dependencies, making them more suitable for complex relationships.
   - They can lead to more accurate feature selection but may be computationally expensive.

6. Regularization Methods for Feature Selection:

   - Regularization methods like L1 (Lasso) and L2 (Ridge) penalize the magnitude of feature coefficients during model training.
   - They encourage sparsity in the feature set by pushing some feature coefficients to zero, effectively selecting a subset of features.

7. Embedded Feature Selection Methods:

   - Embedded methods perform feature selection as part of the model training process.
   - Examples include L1 regularization in linear models, decision tree-based feature importance, and recursive feature elimination with support vector machines.
   - They consider feature relevance within the context of the model's optimization, making them efficient and effective.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 1

Problem Statement

To introduce various libraries of python used for machine learning.

Code

```python
# importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
```

Python

1. Print 2D and 3D array using numpy

```python
# printing 2D array
print("2D Array: ")
print(np.array([[1, 2, 3], [4, 5, 6]]))

print("\n")

# printing 3D array
print("3D Array: ")
print(np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]]))
```

Python

```
2D Array:
[[1 2 3]
 [4 5 6]]


3D Array:
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]
```

2. Creating Dataframe from Dictionary

```python
# creating a data dictionary
sample_data_dict = {'name': ['John', 'Mary', 'Peter', 'Jeff', 'Bill', 'Lisa', 'Joseph'],
                    'age': [23, 78, 22, 19, 45, 33, 20],
                    }

# creating a dataframe from dictionary
sample_data_df = pd.DataFrame(sample_data_dict, columns=['name', 'age'])

# print dataframe
print(sample_data_df.to_markdown())
```

Python

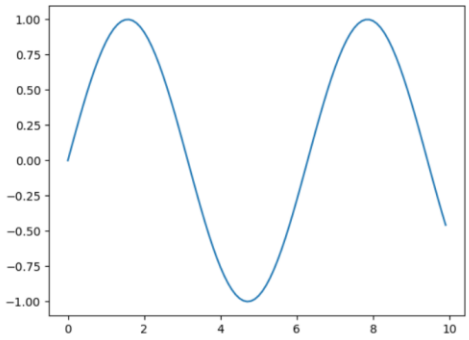
	name	age
0	John	23
1	Mary	78
2	Peter	22
3	Jeff	19
4	Bill	45
5	Lisa	33
6	Joseph	20

3. Plotting Data Using Matplotlib

```python
# data for plotting
x = np.arange(0, 10, 0.1);
y = np.sin(x);

# plotting
plt.plot(x, y);
```

Python



4. Reading Image Using CV2

```python
# reading image using cv2
img = cv2.imread('image.jpg')

# print the image shape
print(img.shape)
```

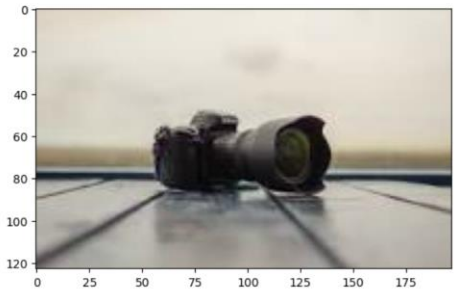
Python

```
(123, 197, 3)
```

5. Implementing imgread, imgsave, imgresize using CV2 Library

```python
# using cv1 library and implementing the functions imgread, imgsave, imgresize
path = 'image.jpg'
img = cv2.imread(path)
plt.imshow(img)
plt.show()
```

Python



6. Using Pandas to Read Data From System and Internet

```python
# using pandas to read the CSV file
csv_file_path = 'annual-enterprise-survey-2021-financial-year-provisional-size-bands-csv.csv'
csv_df = pd.read_csv(csv_file_path)
print("Reading CSV From File: \n", csv_df.head().to_markdown())

print("\n\n")

# reading data from the CSV URL
csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/blood-transfusion/transfusion.data'
data_frame = pd.read_csv(csv_url)
print("Reading CSV From URL: \n", data_frame.head().to_markdown())
```

[ ] Python

```
Reading CSV From File:
|    |   year | industry_code_ANZSIC   | industry_name_ANZSIC              | rme_size_grp   | variable                                          |   value | unit             |
|---:|-------:|:-----------------------|:----------------------------------|:---------------|:--------------------------------------------------|--------:|:-----------------|
|  0 |   2011 | A                      | Agriculture, Forestry and Fishing | a_0            | Activity unit                                     |   46134 | COUNT            |
|  1 |   2011 | A                      | Agriculture, Forestry and Fishing | a_0            | Rolling mean employees                            |       0 | COUNT            |
|  2 |   2011 | A                      | Agriculture, Forestry and Fishing | a_0            | Salaries and wages paid                           |     279 | DOLLARS(millions) |
|  3 |   2011 | A                      | Agriculture, Forestry and Fishing | a_0            | Sales, government funding, grants and subsidies   |    8187 | DOLLARS(millions) |
|  4 |   2011 | A                      | Agriculture, Forestry and Fishing | a_0            | Total income                                      |    8866 | DOLLARS(millions) |


Reading CSV From URL:
|    |   Recency (months) |   Frequency (times) |   Monetary (c.c. blood) |   Time (months) |   whether he/she donated blood in March 2007 |
|---:|-------------------:|--------------------:|------------------------:|----------------:|---------------------------------------------:|
|  0 |                  2 |                  50 |                   12500 |              98 |                                            1 |
|  1 |                  0 |                  13 |                    3250 |              28 |                                            1 |
|  2 |                  1 |                  16 |                    4000 |              35 |                                            1 |
|  3 |                  2 |                  20 |                    5000 |              45 |                                            1 |
|  4 |                  1 |                  24 |                    6000 |              77 |                                            0 |
```

7. Converting Pandas Dataframe Into Numpy Array & Vice-Versa

```python
# converting pandas data frame to numpy array and vice versa

# creating a data frame from numpy array
df = pd.DataFrame(np.random.randn(10, 5), columns=list('ABCDE'))
print("Data Frame Made from Numpy Array: \n", df)
# converting data frame to numpy array
arr = df.to_numpy()

# converting numpy array to data frame
df2 = pd.DataFrame(arr, columns=list('ABCDE'))
```

[ ] Python

```
Data Frame Made from Numpy Array:
          A         B         C         D         E
0  0.645137 -0.865185 -0.709355 -0.311897  0.029653
1 -0.432407 -0.764735  0.396220  0.188073  0.053348
2 -0.328087 -1.134339 -0.149619  1.264454  0.343222
3  0.269465 -0.797453  0.462518 -0.167603 -0.011573
4 -1.300576  0.695887  0.062545  2.110016 -0.991533
5  0.738981  0.596257  0.024780  0.699968 -1.515485
6  0.961992 -0.150316  0.129641 -0.452180 -1.401265
7  0.550747  0.782781 -2.111868  1.187395  0.439801
8 -0.111572  0.622564  0.749648  0.435019 -0.516049
9 -1.088267  0.031369  0.004853  0.910678 -0.903856
```

8. Using loc and ilco to Acces Dataframe Rows and Columns

```python
# accessing rows and columns in a df using iloc and loc

# create a dataframe from a dictionary
df = pd.DataFrame({'a': [1, 2, 3], 'b': [4, 5, 6]})

# access a row using iloc
print("Accessing row using iloc")
print(df.iloc[0].to_markdown())

print("\n")

# access a column using iloc
print("Accessing column using iloc")
print(df.iloc[:, 0].to_markdown())

print("\n")

# access a row using loc
print("Accessing row using loc")
print(df.loc[0].to_markdown())

print("\n")

# access a column using loc
print("Accessing column using loc")
print(df.loc[:, 'a'].to_markdown())

```

[ ] Python

```
Accessing row using iloc
|    |   0 |
|:---|----:|
| a  |   1 |
| b  |   4 |


Accessing column using iloc
|    |   a |
|---:|----:|
|  0 |   1 |
|  1 |   2 |
|  2 |   3 |


Accessing row using loc
|    |   0 |
|:---|----:|
| a  |   1 |
| b  |   4 |


Accessing column using loc
|    |   a |
|---:|----:|
|  0 |   1 |
|  1 |   2 |
|  2 |   3 |
```

EXPERIMENT NO. 2

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 18.08.2023
Faculty Signature:
Marks:

Objective(s):

- To understand the importance of data pre-processing techniques.
- To handle missing values, duplicate values, feature scaling etc.

Outcome:

Students will be familiarized with the understanding and importance of applying various data pre-processing techniques.

Problem Statement:

Write a program to perform data pre-processing techniques for effective machine learning.

Background Study: Data preprocessing in Machine Learning is a crucial step that helps enhance the quality of data to promote the extraction of meaningful insights from the data. Data preprocessing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models.

```
1.data_set= pd.read_csv('Dataset.csv')
```

data_set is a name of the variable to store our dataset, and inside the function, we have passed the name of our dataset. Once we execute the above line of code, it will successfully import the dataset in our code.
Scikit-learn library in our code, which contains various libraries for building machine learning models

Question Bank:

1. What are different ways to handle missing values both for numerical as well as categorical data?
   Numerical Data:
   - Imputation: Replace missing values with statistical measures like mean, median, or mode.
   - Predictive Modeling: Use algorithms like linear regression to predict and fill in missing values.
   - Deletion: Remove rows with missing values, useful when the dataset is large and the missing data is minimal.
   - Assign a Unique Value: Assign a unique value that does not exist in the dataset to indicate missingness.

Categorical Data:

- Mode Imputation: Replace missing values with the most frequent category.
- Create a Missing Category: Treat missing data as a separate category.
- Predictive Modeling: Use classification algorithms to predict the missing categories.
- Deletion: Similar to numerical data, remove rows or columns with missing values.

2. What is the function in python used for finding duplicate rows in data?
   The function DataFrame.duplicated() in Pandas is commonly used to identify duplicate rows in a dataset. It returns a boolean series indicating whether each row is a duplicate or not.

3. Differentiate between two scaling methods used for feature scaling?
   Standardization (Z-Score Normalization) transforms the data to have a mean of 0 and a standard deviation of 1. It is done using the formula (X - mean) / standard deviation. This method is useful in algorithms that assume data is centered around zero and in the same scale, like SVM or PCA.

   Min-Max Scaling rescales the data to a fixed range, usually 0 to 1. The formula is (X - min) / (max - min). This method is sensitive to outliers and often used when the algorithm requires data within a bounded interval, like neural networks.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 2

Problem Statement

Write a program to perform data pre-processing techniques for effective machine learning.

Code

```python
# importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[36] Python

Getting the Datasets

```python
dataset_1_file_path = "dataset1.csv"
dataset_2_file_path = "dataset2.csv"

dataset_1 = pd.read_csv(dataset_1_file_path)
dataset_2 = pd.read_csv(dataset_2_file_path)

print("Dataset 1")
print(dataset_1.head().to_markdown())

print("\nDataset 2")
print(dataset_2.head().to_markdown())
```

[2] ✓ 0.0s Python

```
Dataset 1
|    |   common_id |   feature1 |   feature2 | feature3   |   feature4 |
|---:|------------:|-----------:|-----------:|:-----------|-----------:|
|  0 |           0 |    37.454  |   0.870471 | B          |        971 |
|  1 |           1 |    95.0714 |  -2.99007  | A          |        882 |
|  2 |           2 |    73.1994 |   0.917608 | B          |        470 |
|  3 |           3 |    59.8658 | -19.8757   | A          |        142 |
|  4 |           4 |    15.6019 |  -2.19672  | D          |         91 |

Dataset 2
|    |   common_id |   feature5 | feature6   |   feature7 |   feature8 |
|---:|------------:|-----------:|:-----------|-----------:|-----------:|
|  0 |          50 |    6.00823 | F          |   -4.48618 |       1337 |
|  1 |          51 |   17.094   | H          |    7.62915 |       1012 |
|  2 |          52 |    4.58995 | E          |    1.83455 |       1144 |
|  3 |          53 |    4.70785 | E          |    1.04748 |        732 |
|  4 |          54 |   15.5707  | F          |   -4.37781 |       1023 |
```

Checking For Noise and Removing It (If Any)

```python
# checking for missing values in both the datasets
print("Missing values in the dataset_1: ", dataset_1.isnull().sum().sum())
print("Missing values in the dataset_2: ", dataset_2.isnull().sum().sum())
```

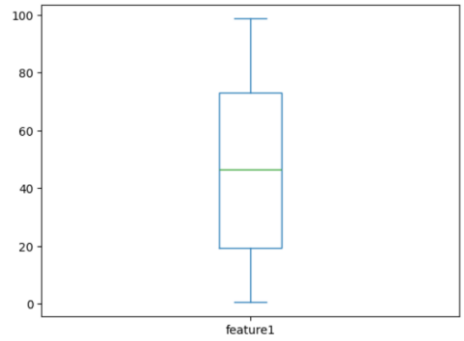
[3] ✓ 0.0s Python

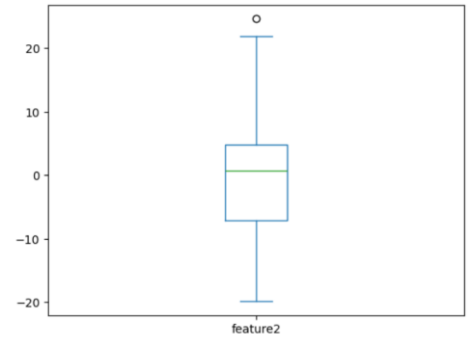
```
Missing values in the dataset_1:  0
Missing values in the dataset_2:  0
```

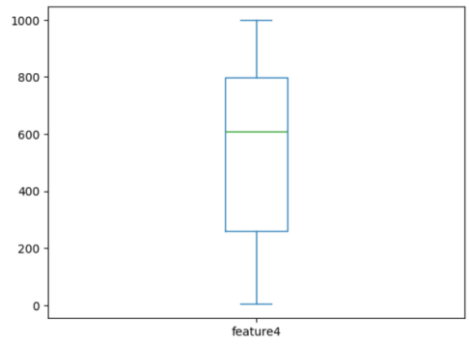
```
# plotting all the features in the dataset_1 to check the outliers

dataset_1['feature1'].plot(kind='box', layout=(5,5))
plt.show()
dataset_1['feature2'].plot(kind='box', layout=(5,5))
plt.show()
dataset_1['feature4'].plot(kind='box', layout=(5,5))
plt.show()
```

[4] ✓ 0.4s Python







```python
def replace_outliers_with_mean(df, column_name):
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    mean_value = df[column_name].mean()
    df[column_name] = df.apply(
        lambda row: mean_value if (
            row[column_name] < lower_bound or row[column_name] > upper_bound) else row[column_name],
        axis=1
    )
    return df


# Handle outliers for dataset1 and dataset2
dataset1_cleaned = dataset_1.copy()
dataset2_cleaned = dataset_2.copy()
for column in ["feature1", "feature2", "feature4"]:
    dataset1_cleaned = replace_outliers_with_mean(dataset1_cleaned, column)
for column in ["feature5", "feature7", "feature8"]:
    dataset2_cleaned = replace_outliers_with_mean(dataset2_cleaned, column)

print("Dataset 1 Cleaned")
print(dataset1_cleaned.head().to_markdown())

print("\n")

print("Dataset 2 Cleaned")
print(dataset2_cleaned.head().to_markdown())

```

[5] ✓ 0.0s Python

```
Dataset 1 Cleaned
|    |   common_id |   feature1 |   feature2 | feature3   |   feature4 |
|---:|------------:|-----------:|-----------:|:-----------|-----------:|
|  0 |           0 |    37.454  |   0.870471 | B          |        971 |
|  1 |           1 |    95.0714 |  -2.99007  | A          |        882 |
|  2 |           2 |    73.1994 |   0.917608 | B          |        470 |
|  3 |           3 |    59.8658 | -19.8757   | A          |        142 |
|  4 |           4 |    15.6019 |  -2.19672  | D          |         91 |


Dataset 2 Cleaned
|    |   common_id |   feature5 | feature6   |   feature7 |   feature8 |
|---:|------------:|-----------:|:-----------|-----------:|-----------:|
|  0 |          50 |    6.00823 | F          |   -4.48618 |       1337 |
|  1 |          51 |   17.094   | H          |    7.62915 |       1012 |
|  2 |          52 |    4.58995 | E          |    1.83455 |       1144 |
|  3 |          53 |    4.70785 | E          |    1.04748 |        732 |
|  4 |          54 |   15.5707  | F          |   -4.37781 |       1023 |
```

Normalizing Data

```python
def normalize_column(df, column_name):
    min_value = df[column_name].min()
    max_value = df[column_name].max()
    df[column_name] = (df[column_name] - min_value) / (max_value - min_value)
    return df

# Normalize numeric features for dataset1_cleaned and dataset2_cleaned
for column in ["feature1", "feature2", "feature4"]:
    dataset1_cleaned = normalize_column(dataset1_cleaned, column)
for column in ["feature5", "feature7", "feature8"]:
    dataset2_cleaned = normalize_column(dataset2_cleaned, column)

print("Dataset 1 After Normalization")
print(dataset1_cleaned.head().to_markdown())

print("\n")

print("Dataset 2 Cleaned After Normalization")
print(dataset2_cleaned.head().to_markdown())
```

[6] ✓ 0.0s Python

```
Dataset 1 After Normalization
|    |   common_id |   feature1 |   feature2 | feature3   |   feature4 |
|---:|------------:|-----------:|-----------:|:-----------|-----------:|
|  0 |           0 |   0.376025 |   0.496554 | B          |   0.972837 |
|  1 |           1 |   0.96314  |   0.404153 | A          |   0.8833   |
|  2 |           2 |   0.740267 |   0.497682 | B          |   0.468813 |
|  3 |           3 |   0.604399 |   0        | A          |   0.138833 |
|  4 |           4 |   0.153354 |   0.423142 | D          |   0.0875252 |
```

Dataset 2 Cleaned After Normalization

	common_id	feature5	feature6	feature7	feature8
0	50	0.106932	F	0.253226	0.850305
1	51	0.333027	H	0.779775	0.519348
2	52	0.0780058	E	0.527933	0.653768
3	53	0.0804103	E	0.493727	0.234216
4	54	0.301959	F	0.257936	0.53055

Integrating the Datasets

```
merged_dataset = pd.merge(dataset1_cleaned, dataset2_cleaned, on="common_id", how="inner")

print("Merged Dataset: ")
print(merged_dataset.head().to_markdown())
```

[7] ✓ 0.0s Python

Merged Dataset:

	common_id	feature1	feature2	feature3	feature4	feature5	feature6	feature7	feature8
0	50	0.982369	0.238637	A	0.634809	0.106932	F	0.253226	0.850305
1	51	0.784225	0.340178	B	0.10161	0.333027	H	0.779775	0.519348
2	52	0.951712	0.499571	C	0.669014	0.0780058	E	0.527933	0.653768
3	53	0.906192	0.355214	B	0.657948	0.0804103	E	0.493727	0.234216
4	54	0.603627	0.104572	B	0.655936	0.301959	F	0.257936	0.53055

Perfroming Reduction on Merged Dataset

```
# Remove rows where feature3 is 'C' or 'D'
reduced_data = merged_dataset[merged_dataset['feature3'].isin(['C', 'D'])]
print("Rows Where Feature3 is 'C' or 'D':")
print(reduced_data.head().to_markdown())

print("\n")

# Updating the merged dataset
merged_dataset = merged_dataset.drop(reduced_data.index)
print("Merged Dataset After Removing Rows Where Feature3 is 'C' or 'D':")
print(merged_dataset.head().to_markdown())
```

[8] ✓ 0.0s Python

Rows Where Feature3 is 'C' or 'D':

	common_id	feature1	feature2	feature3	feature4	feature5	feature6	feature7	feature8
2	52	0.951712	0.499571	C	0.669014	0.0780058	E	0.527933	0.653768
5	55	0.933753	0.49213	C	0.115694	0.983255	G	0.397167	0.349287
6	56	0.0845459	0.22146	D	0.830986	0.163187	F	0.23367	0.0437882
8	58	0.0404591	0.255658	D	0.602616	0.762839	H	0.185451	0.959267
10	60	0.390431	0.28825	D	0.95171	0.337539	E	0.28372	0.740326

Merged Dataset After Removing Rows Where Feature3 is 'C' or 'D':

	common_id	feature1	feature2	feature3	feature4	feature5	feature6	feature7	feature8
0	50	0.982369	0.238637	A	0.634809	0.106932	F	0.253226	0.850305
1	51	0.784225	0.340178	B	0.10161	0.333027	H	0.779775	0.519348
3	53	0.906192	0.355214	B	0.657948	0.0804103	E	0.493727	0.234216
4	54	0.603627	0.104572	B	0.655936	0.301959	F	0.257936	0.53055
7	57	0.194077	0.589073	B	0.78672	0.00189324	E	0.341468	0.552953

EXPERIMENT NO. 3

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 25.08.2023
Faculty Signature:
Marks:

Objective(s):

- To perform different encoding schemes.
- To prepare dataset by converting categorical data to numeric form for machine learning.
- To understand and implement label encoding and one hot encoding.

Outcome:

Students will be able to understand different encoding schemes to prepare data for machine learning.

Problem Statement:

To apply different feature encoding schemes on the given dataset.

Background Study: Machine learning models require all input and output variables to be numeric. This means that if your data contains categorical data, you must encode it to numbers **before you can fit and evaluate a model**. The two most popular techniques are Label Encoding and One-Hot Encoding.

LabelEncoder() class from **preprocessing** library. This class has successfully encoded the variables into digits.

OneHotEncoder class of **preprocessing** library: This class encodes all the variables into numbers 0 and 1 and divided into three columns.

Question Bank:

1. Can ML algorithms handle categorical data directly?
   Most machine learning algorithms cannot handle categorical data directly. They require all input and output variables to be numeric. This necessitates the preprocessing of categorical data into a numerical format before feeding it into an ML model.

2. What are the different schemes for encoding categorical data?
   - Label Encoding: Assigns a unique integer to each category.
   - One-Hot Encoding: Creates a new binary column for each category level.
   - Ordinal Encoding: Assigns integers to categories based on order or hierarchy.
   - Binary Encoding: Combines binary values to represent categories.
   - Frequency or Count Encoding: Replaces categories with their counts or frequency in the dataset.
   - Mean Encoding: Replaces categories with the mean value of the target variable.

3. Differentiate between Label Encoding and One Hot Encoding?
   - **Label Encoding:** Converts each category into a unique integer. It's simple but can introduce a false sense of order or importance among categories (e.g., category 2 is not “higher” or “better” than category 1).
   - **One-Hot Encoding:** Creates a new binary column for each level of the categorical feature. Each category is represented by a vector of 0s and a single 1 indicating the presence of that category. This method avoids introducing a false ordinal relationship but increases the dataset's dimensionality.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 3

Problem Statement

To apply different feature encoding schemes on the given dataset.

Code

```
# importing required libraries
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder
```

[3] Python

```
# reading the dataset
data = pd.read_csv('kaggle_encoding_challenge_dataset.csv')

# printing the first 5 rows of the dataset
print(data.head().to_markdown())
```

[4] Python

	id	bin_0	bin_1	bin_2	bin_3	bin_4	nom_0	nom_1	nom_2	nom_3	nom_4	nom_5	nom_6	nom_7	nom_8	nom_9	ord
0	300000	0	0	1	T	Y	Blue	Triangle	Axolotl	Finland	Piano	0870b0a5d	9ceb19dd6	530f8ecc3	9d117320c	3c49b42b8	
1	300001	0	0	0	T	N	Red	Square	Lion	Canada	Piano	a5c276589	1ad744242	12e6161c9	46ae3059c	285771075	
2	300002	1	0	1	F	Y	Blue	Square	Dog	China	Piano	568550f04	1fe17a1fd	27d6df03f	b759e21f0	6f323c53f	
3	300003	0	0	1	T	Y	Red	Star	Cat	China	Piano	c5725677e	a6542cec0	30c63bd0c	0b6ec68ff	b5de3dcc4	
4	300004	0	1	1	F	N	Red	Trapezoid	Dog	China	Piano	e70a6270d	97b6a3518	a42386065	f91f3b1ee	967cfa9c9	

```
# dropping not required columns from the dataset (bin_0, bin_1, bin_2, bin_3, bin_4)
data.drop(['bin_0', 'bin_1', 'bin_2', 'bin_3', 'bin_4'], axis=1, inplace=True)

# printing the first 5 rows of the dataset
print(data.head().to_markdown())
```

[5] Python

	id	nom_0	nom_1	nom_2	nom_3	nom_4	nom_5	nom_6	nom_7	nom_8	nom_9	ord_0	ord_1	ord_2	ord_3	ord_4
0	300000	Blue	Triangle	Axolotl	Finland	Piano	0870b0a5d	9ceb19dd6	530f8ecc3	9d117320c	3c49b42b8	2	Novice	Warm	j	P
1	300001	Red	Square	Lion	Canada	Piano	a5c276589	1ad744242	12e6161c9	46ae3059c	285771075	1	Master	Lava Hot	l	A
2	300002	Blue	Square	Dog	China	Piano	568550f04	1fe17a1fd	27d6df03f	b759e21f0	6f323c53f	2	Expert	Freezing	a	G
3	300003	Red	Star	Cat	China	Piano	c5725677e	a6542cec0	30c63bd0c	0b6ec68ff	b5de3dcc4	1	Contributor	Lava Hot	b	Q
4	300004	Red	Trapezoid	Dog	China	Piano	e70a6270d	97b6a3518	a42386065	f91f3b1ee	967cfa9c9	3	Grandmaster	Lava Hot	l	W

One Hot Encoding

```
# make a copy of the dataset
data_one_hot_encoding = data.copy()

# performing one hot encoding on the dataset
data_one_hot_encoding = pd.get_dummies(data_one_hot_encoding, columns=['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4', 'ord_1', 'ord_2', 'ord_3', 'ord_4', 'ord_5'])

# printing the first 5 rows of the dataset
print(data_one_hot_encoding.head().to_markdown())
```

[6] Python

	id	nom_5	nom_6	nom_7	nom_8	nom_9	ord_0	day	month	nom_0_Blue	nom_0_Green	nom_0_Red	nom_1_Circle	nom_1_Polygon
0	300000	0870b0a5d	9ceb19dd6	530f8ecc3	9d117320c	3c49b42b8	2	5	11	True	False	False	False	False
1	300001	a5c276589	1ad744242	12e6161c9	46ae3059c	285771075	1	7	5	False	False	True	False	False
2	300002	568550f04	1fe17a1fd	27d6df03f	b759e21f0	6f323c53f	2	1	12	True	False	False	False	False
3	300003	c5725677e	a6542cec0	30c63bd0c	0b6ec68ff	b5de3dcc4	1	2	3	False	False	True	False	False
4	300004	e70a6270d	97b6a3518	a42386065	f91f3b1ee	967cfa9c9	3	4	11	False	False	True	False	False

Label Encoding

```python
# performing label encoding on the dataset

# creating an object of the LabelEncoder class
le = LabelEncoder()

# make a copy of the dataset
data_label_encoding = data.copy()

# performing label encoding on the dataset
data_label_encoding['nom_0'] = le.fit_transform(data_label_encoding['nom_0'])
data_label_encoding['nom_1'] = le.fit_transform(data_label_encoding['nom_1'])
data_label_encoding['nom_2'] = le.fit_transform(data_label_encoding['nom_2'])
data_label_encoding['nom_3'] = le.fit_transform(data_label_encoding['nom_3'])
data_label_encoding['nom_4'] = le.fit_transform(data_label_encoding['nom_4'])

# printing the first 5 rows of the dataset
print(data_label_encoding.head().to_markdown())

```

[7] Python

	id	nom_0	nom_1	nom_2	nom_3	nom_4	nom_5	nom_6	nom_7	nom_8	nom_9	ord_0	ord_1	ord_2	ord_3	ord_4	or
0	300000	0	5	0	3	2	0870b0a5d	9ceb19dd6	530f8ecc3	9d117320c	3c49b42b8	2	Novice	Warm	j	P	be
1	300001	2	2	4	0	2	a5c276589	1ad744242	12e6161c9	46ae3059c	285771075	1	Master	Lava Hot	l	A	RP
2	300002	0	2	2	1	2	568550f04	1fe17a1fd	27d6df03f	b759e21f0	6f323c53f	2	Expert	Freezing	a	G	tP
3	300003	2	3	1	1	2	c5725677e	a6542cec0	30c63bd0c	0b6ec68ff	b5de3dcc4	1	Contributor	Lava Hot	b	Q	ke
4	300004	2	4	2	1	2	e70a6270d	97b6a3518	a42386065	f91f3b1ee	967cfa9c9	3	Grandmaster	Lava Hot	l	W	qK

Dummy Encoding

```python
# performing dummy encoding on the dataset

# make a copy of the dataset
data_dummy_encoding = data.copy()

# performing dummy encoding on the dataset
data_dummy_encoding = pd.get_dummies(data_dummy_encoding, columns=['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4'], drop_first=True)
print(data_dummy_encoding.head().to_markdown())
```

[8] Python

	id	nom_5	nom_6	nom_7	nom_8	nom_9	ord_0	ord_1	ord_2	ord_3	ord_4	ord_5	day	month	nom_0_Green	nom_0_Re
0	300000	0870b0a5d	9ceb19dd6	530f8ecc3	9d117320c	3c49b42b8	2	Novice	Warm	j	P	be	5	11	False	False
1	300001	a5c276589	1ad744242	12e6161c9	46ae3059c	285771075	1	Master	Lava Hot	l	A	RP	7	5	False	True
2	300002	568550f04	1fe17a1fd	27d6df03f	b759e21f0	6f323c53f	2	Expert	Freezing	a	G	tP	1	12	False	False
3	300003	c5725677e	a6542cec0	30c63bd0c	0b6ec68ff	b5de3dcc4	1	Contributor	Lava Hot	b	Q	ke	2	3	False	True
4	300004	e70a6270d	97b6a3518	a42386065	f91f3b1ee	967cfa9c9	3	Grandmaster	Lava Hot	l	W	qK	4	11	False	True

Ordinal Encoding

```python
# performing ordinal encoding on the dataset

# creating a copy of the dataset
data_ordinal_encoding = data.copy()

# creating an object of the OrdinalEncoder class
encoder = OrdinalEncoder()

# performing ordinal encoding on the dataset
data_ordinal_encoding[['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4']] = encoder.fit_transform(
    data_ordinal_encoding[['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4']])
print(  data_ordinal_encoding.head().to_markdown())
```

[9] Python

	id	nom_0	nom_1	nom_2	nom_3	nom_4	nom_5	nom_6	nom_7	nom_8	nom_9	ord_0	ord_1	ord_2	ord_3	ord_4	or
0	300000	0	5	0	3	2	0870b0a5d	9ceb19dd6	530f8ecc3	9d117320c	3c49b42b8	2	Novice	Warm	j	P	be
1	300001	2	2	4	0	2	a5c276589	1ad744242	12e6161c9	46ae3059c	285771075	1	Master	Lava Hot	l	A	RP
2	300002	0	2	2	1	2	568550f04	1fe17a1fd	27d6df03f	b759e21f0	6f323c53f	2	Expert	Freezing	a	G	tP
3	300003	2	3	1	1	2	c5725677e	a6542cec0	30c63bd0c	0b6ec68ff	b5de3dcc4	1	Contributor	Lava Hot	b	Q	ke
4	300004	2	4	2	1	2	e70a6270d	97b6a3518	a42386065	f91f3b1ee	967cfa9c9	3	Grandmaster	Lava Hot	l	W	qK

Binary Encoding

```python
# performing binary encoding on the dataset

# make a copy of the dataset
data_binary_encoding = data.copy()

# mapping
mapping = []

# creating a list of dictionaries containing the mapping for each column
for col in ['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4']:
    unique_values = data_binary_encoding[col].unique()
    value_to_int_mapping = {value: idx for idx,
                            value in enumerate(unique_values)}
    mapping.append({'col': col, 'mapping': value_to_int_mapping})

# performing binary encoding on the dataset
data_binary_encoding = data_binary_encoding.drop(
    ['ord_1', 'ord_2', 'ord_3', 'ord_4', 'ord_5'], axis=1)

for col_map in mapping:
    data_binary_encoding[col_map['col']] = data_binary_encoding[col_map['col']].map(
        col_map['mapping'])


# applying binary format on the dataset columns

# select the columns to be converted to binary format
data_binary_encoding[['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4']] = data_binary_encoding[['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4']].astype(
    'category').apply(lambda x: x.cat.codes)

data_binary_encoding[['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4']] = data_binary_encoding[['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4']].applymap(lambda x: format(x, 'b'))

# printing the first 5 rows of the dataset
```

```python
print(data_binary_encoding.head().to_markdown())
```

[10] Python

```
C:\Users\mainp\AppData\Local\Temp\ipykernel_15068\3578475438.py:31: FutureWarning: DataFrame.applymap has been deprecated. Use DataFrame.map instead.
  data_binary_encoding[['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4']] = data_binary_encoding[['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4']].applymap(lambda x: format(x, 'b'))
```

	id	nom_0	nom_1	nom_2	nom_3	nom_4	nom_5	nom_6	nom_7	nom_8	nom_9	ord_0	day	month
0	300000	0	0	0	0	0	0870b0a5d	9ceb19dd6	530f8ecc3	9d117320c	3c49b42b8	2	5	11
1	300001	1	1	1	1	0	a5c276589	1ad744242	12e6161c9	46ae3059c	285771075	1	7	5
2	300002	0	1	10	10	0	568550f04	1fe17a1fd	27d6df03f	b759e21f0	6f323c53f	2	1	12
3	300003	1	10	11	10	0	c5725677e	a6542cec0	30c63bd0c	0b6ec68ff	b5de3dcc4	1	2	3
4	300004	1	11	10	10	0	e70a6270d	97b6a3518	a42386065	f91f3b1ee	967cfa9c9	3	4	11

Count Encoding

```python
# performing count encoding on the dataset

# creating a copy of the datatset
data_count_encoding = data.copy()

for col in ['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4']:
    count_map = data_count_encoding[col].value_counts().to_dict()
    data_count_encoding[col +
                        '_count'] = data_count_encoding[col].map(count_map)
data_count_encoding.drop(
    columns=['nom_0', 'nom_1', 'nom_2', 'nom_3', 'nom_4'], inplace=True)
print(data_count_encoding.head().to_markdown())
```

[11] Python

	id	nom_5	nom_6	nom_7	nom_8	nom_9	ord_0	ord_1	ord_2	ord_3	ord_4	ord_5	day	month	nom_0_count	nom_1_
0	300000	0870b0a5d	9ceb19dd6	530f8ecc3	9d117320c	3c49b42b8	2	Novice	Warm	j	P	be	5	11	63762	
1	300001	a5c276589	1ad744242	12e6161c9	46ae3059c	285771075	1	Master	Lava Hot	l	A	RP	7	5	51083	
2	300002	568550f04	1fe17a1fd	27d6df03f	b759e21f0	6f323c53f	2	Expert	Freezing	a	G	tP	1	12	63762	
3	300003	c5725677e	a6542cec0	30c63bd0c	0b6ec68ff	b5de3dcc4	1	Contributor	Lava Hot	b	Q	ke	2	3	51083	
4	300004	e70a6270d	97b6a3518	a42386065	f91f3b1ee	967cfa9c9	3	Grandmaster	Lava Hot	l	W	qK	4	11	51083	

EXPERIMENT NO. 4

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 08.09.2023
Faculty Signature:
Marks:

Objective(s):

- To understand the importance of feature selection
- To differentiate between different types of feature selection.
- Build a model using feature selection techniques.

Outcome:

Students will be familiarized with model building using feature selection techniques and optimization.

Problem Statement:

Write a program to apply filter feature selection techniques.

Background Study: Feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.

Question Bank:

1. What are different filter feature selection techniques?
   - Correlation Coefficient (like Pearson or Spearman) for measuring linear relationships between variables.
   - Chi-Square Test for assessing the independence of two categorical variables.
   - ANOVA (Analysis of Variance) for testing differences in means across multiple groups, useful in numerical inputs and categorical outputs.
   - Information Gain for measuring the reduction in entropy from transforming a dataset.
   - Variance Threshold for removing features that do not meet a certain variance threshold, often used for eliminating constant features.

2. How feature selection techniques depend on the data type of input features and output variable?

- For numerical inputs and outputs, techniques like Pearson's correlation and regression models are used.
- For categorical inputs and outputs, methods like the Chi-Square test and contingency tables are applied.
- When dealing with numerical inputs and categorical outputs, ANOVA or logistic regression can be used.
- For categorical inputs and numerical outputs, techniques like the correlation ratio are utilized.

3. What is the mathematics behind Pearson's Correlation to rank features?
   - Pearson's Correlation Coefficient measures the linear correlation between two variables using the formula r = Σ(X - mean of X)(Y - mean of Y) / sqrt[Σ(X - mean of X)^2 * Σ(Y - mean of Y)^2].
   - It ranges from -1 to +1, where +1 indicates a perfect positive linear relationship, -1 a perfect negative linear relationship, and 0 no linear relationship.
   - For feature selection, features with a higher absolute value of r (closer to 1 or -1) are considered more significant.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 4

Problem Statement:

Write a program to apply filter feature selection techniques.

Code:

```
# import the requied libraries
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest, f_regression, RFE, SequentialFeatureSelector
from sklearn.linear_model import LogisticRegression, Lasso
from sklearn.preprocessing import LabelEncoder
```

[42] Python

```
# import the datatset
iris_data = pd.read_csv('iris.csv')

# print the dataset
print(iris_data.head().to_markdown())
```

[43] Python

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5	3.6	1.4	0.2	Iris-setosa

```
# ecnode the categorical data into numerical data
iris_data['Species'] = iris_data['Species'].replace(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], [0, 1, 2])
```

[44] Python

```
# print the datatype of the dataset
print(iris_data.dtypes)
```

[45] Python

```
Id               int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species            int64
dtype: object
```

```
X = iris_data.drop(columns=['Id', 'Species'])
y = iris_data['Species']
```

[46] Python

```python
# Filter Method
fs = SelectKBest(score_func=f_regression, k=2)
X_selected = fs.fit_transform(X, y)
feature_scores = pd.DataFrame(fs.scores_, index=X.columns, columns=['Score'])
print("Filter Method:")
print(feature_scores)
print("\n")
```

[47] Python

```
Filter Method:
                    Score
SepalLengthCm   233.838996
SepalWidthCm     31.597508
PetalLengthCm  1342.159189
PetalWidthCm   1589.559204
```

```python
# Wrapper Method: RFE with Logistic Regression
model = LogisticRegression(max_iter=1000)
rfe = RFE(estimator=model, n_features_to_select=3)
fit = rfe.fit(X, y)
feature_ranking = pd.DataFrame(
    {'Feature': X.columns, 'Ranking': fit.ranking_}).sort_values(by='Ranking')
print("Wrapper Method: RFE")
print(feature_ranking)
print("\n")
```

[48] Python

```
Wrapper Method: RFE
         Feature  Ranking
1   SepalWidthCm        1
2  PetalLengthCm        1
3   PetalWidthCm        1
0  SepalLengthCm        2
```

```python
# Wrapper Method: Backward Elimination
model = LogisticRegression(max_iter=1000)
sfs_backward = SequentialFeatureSelector(estimator=model, n_features_to_select=3, direction='backward')
sfs_backward.fit(X, y)
feature_ranking = pd.DataFrame(
    {'Feature': X.columns, 'Ranking': sfs_backward.get_support()}).sort_values(by='Ranking')
print("Wrapper Method: Backward Elimination")
print(feature_ranking)
```

[49] Python

```
Wrapper Method: Backward Elimination
         Feature  Ranking
0  SepalLengthCm    False
1   SepalWidthCm     True
2  PetalLengthCm     True
3   PetalWidthCm     True
```

```python
# Wrapper Method: Forward Selection
model = LogisticRegression(max_iter=1000)
sfs_forward = SequentialFeatureSelector(estimator=model, n_features_to_select=3, direction='forward')
sfs_forward.fit(X, y)
feature_ranking = pd.DataFrame(
    {'Feature': X.columns, 'Ranking': sfs_forward.get_support()}).sort_values(by='Ranking')
print("Wrapper Method: Forward Selection")
print(feature_ranking)
```

[51] Python

```
Wrapper Method: Forward Selection
         Feature  Ranking
0  SepalLengthCm    False
1   SepalWidthCm     True
2  PetalLengthCm     True
3   PetalWidthCm     True
```

EXPERIMENT NO. 5

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 01.09.2023
Faculty Signature:
Marks:

Objective(s):

- Study Dimensionality Reduction.
- Understand the basic principle behind Principal Component Analysis.

Outcome:

Students will be familiarized with Dimensionality Reduction especially Principal Component Analysis (PCA).

Problem Statement:

Reduce dimensionality of Iris dataset using Principal Component Analysis.

Background Study: Principal component analysis is a statistical technique that is used to analyze the interrelationships among a large number of variables and to explain these variables in terms of a smaller number of variables, called principal components, with a minimum loss of information.

PCA is affected by scale so you need to scale the features in your data before applying PCA. Use StandardScaler to help you standardize the dataset's features onto unit scale (mean = 0 and variance = 1) which is a requirement for the optimal performance of many machine learning algorithms.
StandardScaler to help you standardize the dataset's features onto unit scale (mean = 0 and variance = 1) which is a requirement for the optimal performance of many machine learning algorithms.
explained_variance_ratio_: helps to visualize the first principal component contains of the variance and the second principal component contains of the variance. Together, the two components of the information.

Question Bank:

1. What is dimensionality reduction?
   Dimensionality reduction is the process of reducing the number of input variables in a dataset. It involves transforming high-dimensional data into a lower-dimensional space, making the data easier to analyze and visualize, and often improving the performance of machine learning

models. Techniques like Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) are commonly used for this purpose.

2. Differentiate between Feature Selection, Feature Engineering and Dimensionality Reduction.
   - Feature Selection: Involves selecting a subset of relevant features (variables, predictors) for use in model construction. It aims to reduce the number of input variables to those that are most useful to the model.
   - Feature Engineering: The process of using domain knowledge to extract new features from raw data that make machine learning algorithms work better. It's about creating new input variables from your existing ones.
   - Dimensionality Reduction: Unlike feature selection, which keeps a subset of the original features, dimensionality reduction transforms features into a lower-dimensional space, often creating new combinations of the original features.

3. What are principal components?
   Principal components are new variables that are constructed as linear combinations of the original variables. These components capture the maximum variance in the data in a successive manner, with the first principal component capturing the most variance, and each subsequent component having a lower variance. In Principal Component Analysis (PCA), these components are orthogonal, ensuring no multicollinearity among them. They are used to simplify the data without much loss of information, aiding in visualization, data compression, and improved efficiency in machine learning algorithms.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 5

Problem Statement:

Students will be familiarized with Dimensionality Reduction especially Principal Component Analysis (PCA).

Dataset Link

https://www.kaggle.com/datasets/uciml/iris

Code

```python
# importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVC
```

[1] Python

```python
# importing the dataset
iris_data = pd.read_csv('iris.csv')

# reading the dataset
print(iris_data.head().to_markdown())
```

[2] Python

```
|    |   Id |   SepalLengthCm |   SepalWidthCm |   PetalLengthCm |   PetalWidthCm | Species     |
|---:|-----:|----------------:|---------------:|----------------:|---------------:|:------------|
|  0 |    1 |             5.1 |            3.5 |             1.4 |            0.2 | Iris-setosa |
|  1 |    2 |             4.9 |              3 |             1.4 |            0.2 | Iris-setosa |
|  2 |    3 |             4.7 |            3.2 |             1.3 |            0.2 | Iris-setosa |
|  3 |    4 |             4.6 |            3.1 |             1.5 |            0.2 | Iris-setosa |
|  4 |    5 |             5   |            3.6 |             1.4 |            0.2 | Iris-setosa |
```

```python
# Dropping the Id column
iris_data.drop("Id", axis=1, inplace=True)

# Separating the features and the target variable
X = iris_data.drop("Species", axis=1)
y = iris_data["Species"]

# Scaling the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print(X_scaled[:5])
print(y.unique())
```

[3] Python

```
[[-0.90068117  1.03205722 -1.3412724  -1.31297673]
 [-1.14301691 -0.1249576  -1.3412724  -1.31297673]
 [-1.38535265  0.33784833 -1.39813811 -1.31297673]
 [-1.50652052  0.10644536 -1.2844067  -1.31297673]
 [-1.02184904  1.26346019 -1.3412724  -1.31297673]]
['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

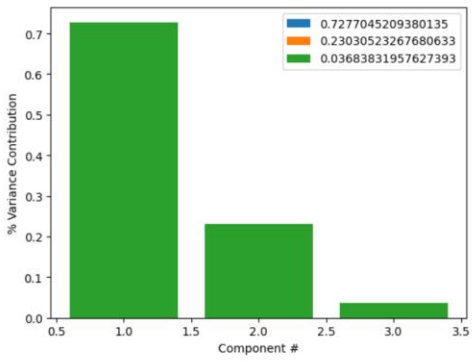
Principal Component Analysis (PCA)

```python
# Applying PCA and reducing the data to 2 principal components
pca = PCA(n_components=3)
principal_components = pca.fit_transform(X_scaled)

explained_var = pca.explained_variance_ratio_
for var in explained_var:
    print(var)
    plt.bar([1, 2, 3], explained_var, label=var)
    plt.xlabel("Component #")
    plt.ylabel("% Variance Contribution")
    plt.legend()
plt.show()
```

[4] Python

```
0.7277045209380135
0.23030523267680633
0.03683831957627393
```




```python
# dropping the third principal component
pca = PCA(n_components=2)
principal_components = pca.fit_transform(X_scaled)

# creating a dataframe of the principal components
principal_df = pd.DataFrame(data=principal_components, columns=["PC1", "PC2"])
print(principal_df.head().to_markdown())
```

[9] Python

	PC1	PC2
0	-2.26454	0.505704
1	-2.08643	-0.655405
2	-2.36795	-0.318477
3	-2.3042	-0.575368
4	-2.38878	0.674767

EXPERIMENT NO. 6

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 22.09.2023
Faculty Signature:
Marks:

Objective(s):

- Understand Simple Linear Regression (SLR).
- Study about the different performance metrics of SLR.

Outcome:

Student will be familiarized with regression problems and SLR as a solution to single feature problem.

Problem Statement:

To apply Simple Linear Regression on the given dataset.

Background Study:

Simple linear regression is an approach for predicting a response using a single feature. It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).
statsmodels — a module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration.
scikit-learn — a module that provides simple and efficient tools for data mining and data analysis.
Calling model.params will show us the model's parameters.

Question Bank:

1. What is a regression problem?
   A regression problem involves predicting a continuous outcome variable (response) based on one or more predictor variables. The goal is to establish a relationship between the predictors and the response, which can be used for forecasting, trend analysis, or determining the influence of certain variables.

2. How Simple Linear Regression (SLR) helps in solving regression problems containing an input feature and an output variable?
   Simple Linear Regression (SLR) is used in regression problems with one input feature and one continuous output variable. It models the relationship between these two variables as a linear equation. The formula for SLR is $Y = \beta0 + \beta1X + \varepsilon$, where Y is the dependent variable, X is the independent variable, β0 is the intercept, β1 is the slope of the line, and ε represents the error term. SLR aims to find the best-fitting line through the data points by minimizing the sum of the

squared differences between the observed values and the values predicted by the linear model. This method is useful for understanding and predicting the behavior of one variable based on another.

3. What are the different performance metrics that can be used for evaluating SLR?
   - Mean Absolute Error (MAE): The average of the absolute differences between the predicted values and actual values.
   - Mean Squared Error (MSE): The average of the squares of the differences between the predicted values and actual values. It gives more weight to larger errors.
   - Root Mean Squared Error (RMSE): The square root of MSE. It is in the same units as the response variable and is sensitive to larger errors.
   - Coefficient of Determination (R^2): Measures the proportion of variability in the dependent variable that can be explained by the independent variable in the model. It ranges from 0 to 1, with higher values indicating a better fit.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 6

Problem Statement:

To apply Simple Linear Regression on the given dataset.

Code

```python
# import required libraries
from sklearn.linear_model import LinearRegression
import pandas as pd
```

[18] Python

```python
# create a dataframe with sample data
df = pd.DataFrame({'X': [95, 85, 80, 70, 60], 'Y': [85, 95, 70, 65, 70]})
```

[19] Python

```python
# computer the mean of X and Y
x_mean = df['X'].mean()
y_mean = df['Y'].mean()

# compute the slope and intercept
slope = ((df['X'] - x_mean) * (df['Y'] - y_mean)).sum() / \
    ((df['X'] - x_mean) ** 2).sum()
intercept = y_mean - slope * x_mean

# taking input from user
x = int(input("Enter the value of X: "))
y = intercept + slope * x
print("The value of Y is: ", y)
```

[20] Python

```
The value of Y is:  33.21917808219178
```

```python
# reshaping the data
X = df['X'].values.reshape(-1, 1)
Y = df['Y'].values.reshape(-1, 1)

# applying linear regression model
reg = LinearRegression().fit(X, Y)

# taking input from user
x = int(input("Enter the value of X: "))
y = reg.predict([[x]])

# printing the value of Y
print("The value of Y is: ", y[0][0])
```

[21] Python

```
The value of Y is:  33.21917808219179
```

EXPERIMENT NO. 7

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 22.09.2023
Faculty Signature:
Marks:

Objective(s):

- Understand mathematics behind Multiple Linear Regression (MLR).
- Solving linear regression problems containing more than one independent feature using MLR.

Outcome:

Students will be familiarized with Multiple Linear Regression for solving linear regression problems.

Problem Statement:

To apply multiple linear regression on any regression dataset.

Background Study:

Multiple Linear Regression attempts to model the relationship between two or more features and a response by fitting a linear equation to observed data. Statsmodels is a Python module that provides classes and functions for the estimation of different statistical models, as well as different statistical tests. statsmodels — a module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration.

Question Bank:

1. What is MLR?
   MLR is a statistical method that models the relationship between two or more independent variables and a dependent variable. It is an extension of simple linear regression to multiple predictors, using the equation $Y = \beta0 + \beta1X1 + \beta2X2 + ... + \betanXn + \varepsilon$, where Y is the response variable, X1 to Xn are predictors, β0 is the intercept, β1 to βn are the coefficients, and ε is the error term.

2. Differentiate between SLR and MLR?
   - SLR (Simple Linear Regression) involves just one independent variable. It models the relationship between a single predictor and a response variable using the equation $Y = \beta0 + \beta1X + \varepsilon$.

- MLR (Multiple Linear Regression) involves multiple independent variables. It models the relationship between several predictors and a response variable using the equation $Y = \beta 0 + \beta 1X1 + \beta 2X2 + ... + \beta nXn + \varepsilon$.
  The key difference is the number of independent variables used: SLR uses one, while MLR uses two or more.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 7

Problem Statement:

To apply multiple linear regression on any regression dataset.

Code:

Using Mathematical Equation

```python
# importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Python

```python
# datatset
y = np.array([140, 155, 159, 179, 192, 200, 212, 215])
x1 = np.array([60, 62, 67, 70, 71, 72, 75, 78])
x2 = np.array([22, 25, 24, 20, 15, 14, 14, 11])

# creating dataframe
df = pd.DataFrame({'y': y, 'x1': x1, 'x2': x2})

# caculating x1^2, x2^2, x1*x2, x1*y, x2*y
df['x1^2'] = df['x1'] ** 2
df['x2^2'] = df['x2'] ** 2
df['x1*x2'] = df['x1'] * df['x2']
df['x1*y'] = df['x1'] * df['y']
df['x2*y'] = df['x2'] * df['y']


# defining number of observations
N = len(df)

# calculating sum of all columns and adding them to dataframe
df.loc['Total'] = df.sum()

# printing dataframe
print(df.to_markdown())

# equation of line
# y = b0 + b1*x1 + b2*x2


# calculating values of sigma_x1_2, sigma_x2_2, sigma_x1_x2, sigma_x1_y, sigma_x2_y
print('\n\nCalculating coefficients:')
sigma_x1_2 = df.loc['Total', 'x1^2'] - (df.loc['Total', 'x1'] ** 2) / N
sigma_x2_2 = df.loc['Total', 'x2^2'] - (df.loc['Total', 'x2'] ** 2) / N
sigma_x1_x2 = df.loc['Total', 'x1*x2'] - \
    (df.loc['Total', 'x1'] * df.loc['Total', 'x2']) / N
sigma_x1_y = df.loc['Total', 'x1*y'] - \
    (df.loc['Total', 'x1'] * df.loc['Total', 'y']) / N
```

```python
sigma_x2_y = df.loc['Total', 'x2*y'] - \
    (df.loc['Total', 'x2'] * df.loc['Total', 'y']) / N
print(f'sigma_x1_2 = {sigma_x1_2}')
print(f'sigma_x2_2 = {sigma_x2_2}')
print(f'sigma_x1_x2 = {sigma_x1_x2}')
print(f'sigma_x1_y = {sigma_x1_y}')
print(f'sigma_x2_y = {sigma_x2_y}')

# calculating y_mean, x1_mean, x2_mean
y_mean = df.loc['Total', 'y'] / N
x1_mean = df.loc['Total', 'x1'] / N
x2_mean = df.loc['Total', 'x2'] / N

print("\n\nMean values:")
print(f'y_mean = {y_mean}')
print(f'x1_mean = {x1_mean}')
print(f'x2_mean = {x2_mean}')

# printing b0, b1, b2
b1 = (sigma_x2_2 * sigma_x1_y - sigma_x1_x2 * sigma_x2_y) / \
    (sigma_x1_2 * sigma_x2_2 - sigma_x1_x2 ** 2)
b2 = (sigma_x1_2 * sigma_x2_y - sigma_x1_x2 * sigma_x1_y) / \
    (sigma_x1_2 * sigma_x2_2 - sigma_x1_x2 ** 2)
b0 = y_mean - b1 * x1_mean - b2 * x2_mean

print('\n\nPrinting b0, b1, b2:')
print(f'b0 = {b0}')
print(f'b1 = {b1}')
print(f'b2 = {b2}')

# calculating line of best fit
y_pred = b0 + b1 * df['x1'] + b2 * df['x2']

# adding line of best fit to dataframe
df['y_pred'] = y_pred

# calculating error
error = df['y'] - y_pred
```

```python
error = df['y'] - y_pred

# adding error to dataframe
df['error'] = error

# printing dataframe
print('\n\nPrinting y_pred:')
print(df.to_markdown())

# calculating mean square error
mse = 0
for i in range(N):
    mse += error[i] ** 2 / N

print(f'\n\nMean square error = {mse}')

# calculating r^2 static
TSS = 0
for i in range(N):
    TSS += (df.loc[i, 'y'] - y_mean) ** 2

RSS = 0
for i in range(N):
    RSS += (df.loc[i, 'y'] - df.loc[i, 'y_pred']) ** 2


r2 = 1 - RSS / TSS
print(f'\n\nr^2 = {r2}')

# calculating adjusted r^2
k = 2  # number of independent variables
adj_r2 = 1 - ((1 - r2) * (N - 1) / (N - k - 1))
print(f'\n\nAdjusted r^2 = {adj_r2}')
```

[ ] Python

```
|       |    y |   x1 |   x2 |   x1^2 |   x2^2 |   x1*x2 |   x1*y |   x2*y |
|:------|-----:|-----:|-----:|-------:|-------:|--------:|-------:|-------:|
| 0     |  140 |   60 |   22 |   3600 |    484 |    1320 |   8400 |   3080 |
| 1     |  155 |   62 |   25 |   3844 |    625 |    1550 |   9610 |   3875 |
| 2     |  159 |   67 |   24 |   4489 |    576 |    1608 |  10653 |   3816 |
| 3     |  179 |   70 |   20 |   4900 |    400 |    1400 |  12530 |   3580 |
| 4     |  192 |   71 |   15 |   5041 |    225 |    1065 |  13632 |   2880 |
| 5     |  200 |   72 |   14 |   5184 |    196 |    1008 |  14400 |   2800 |
| 6     |  212 |   75 |   14 |   5625 |    196 |    1050 |  15900 |   2968 |
| 7     |  215 |   78 |   11 |   6084 |    121 |     858 |  16770 |   2365 |
| Total | 1452 |  555 |  145 |  38767 |   2823 |    9859 | 101895 |  25364 |


Calculating coefficients:
sigma_x1_2 = 263.875
sigma_x2_2 = 194.875
sigma_x1_x2 = -200.375
sigma_x1_y = 1162.5
sigma_x2_y = -953.5


Mean values:
y_mean = 181.5
x1_mean = 69.375
x2_mean = 18.125


Printing b0, b1, b2:
b0 = -6.867487247726768
b1 = 3.147893102683522
```

```
Printing b0, b1, b2:
b0 = -6.867487247726768
b1 = 3.147893102683522
b2 = -1.6561432690175206


Printing y_pred:
```

	y	x1	x2	x1^2	x2^2	x1*x2	x1*y	x2*y	y_pred	error
0	140	60	22	3600	484	1320	8400	3080	145.571	-5.57095
1	155	62	25	3844	625	1550	9610	3875	146.898	8.1017
2	159	67	24	4489	576	1608	10653	3816	164.294	-5.29391
3	179	70	20	4900	400	1400	12530	3580	180.362	-1.36216
4	192	71	15	5041	225	1065	13632	2880	191.791	0.209226
5	200	72	14	5184	196	1008	14400	2800	196.595	3.40519
6	212	75	14	5625	196	1050	15900	2968	206.038	5.96151
7	215	78	11	6084	121	858	16770	2365	220.451	-5.4506
Total	1452	555	145	38767	2823	9859	101895	25364	1500.07	-48.0724

```
Mean square error = 25.430207640275


r^2 = 0.9626163798011393


Adjusted r^2 = 0.9476629317215951
```

Using Built-in Functions

+ Code + Markdown

```python
# importing required libraries
from sklearn.linear_model import LinearRegression
```

Python

```python
# applying Multiple Linear Regression

# creating object of LinearRegression class
regressor = LinearRegression()

# feeding the training data to the model
regressor.fit(df[['x1', 'x2']], df['y'])

# predicting the values of x
y_pred = regressor.predict(df[['x1', 'x2']])

# creating dataframe of x and y
df = pd.DataFrame({'x1': df['x1'], 'x2': df['x2'], 'y': df['y']})

# adding column of predicted y
df['y_pred'] = y_pred

# calculating error in prediction
df['error'] = df['y'] - df['y_pred']

# drop total row
df.drop('Total', inplace=True)

# printing the dataframe
print('\n\nPrinting dataframe:')
print(df.to_markdown())
```

Python

```
...

Printing dataframe:
```

	x1	x2	y	y_pred	error
0	60	22	140	145.967	-5.96712
1	62	25	155	146.889	8.11141
2	67	24	159	163.983	-4.9829
3	70	20	179	180.156	-1.15574
4	71	15	192	191.927	0.0730476
5	72	14	200	196.738	3.26213
6	75	14	212	205.95	6.04959
7	78	11	215	220.383	-5.38318

EXPERIMENT NO. 8

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 20.10.2023
Faculty Signature:
Marks:

Objective(s):

- Study and understand about Polynomial Regression on non-linear regression data.
- Study the mathematics behind Polynomial Regression.

Outcome:

Students will be familiarized with handling of regression data having non-linear relationship between input and output.

Problem Statement:

To apply Polynomial Linear Regression on the given dataset.

Background Study:

Polynomial Regression is a form of linear regression in which the relationship between the independent variable x and dependent variable y is modeled as an nth degree polynomial. Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y, denoted E(y |x). statsmodels — a module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration.

Question Bank:

1. What is non-linear relationship between input and output?
   A non-linear relationship is when the relationship between the input variable(s) and the output variable cannot be accurately described using a straight line. In such cases, the change in the output is not proportional to the change in the input, and the relationship might involve curves, exponential growth or decay, logarithmic patterns, or other complex behaviors.

2. How Polynomial Regression is used to handle non-linear relationship?
   Polynomial Regression extends linear regression by adding polynomial terms, which allows it to model non-linear relationships. Instead of the simple linear equation Y = β0 + β1X, it uses an equation like Y = β0 + β1X + β2X^2 + ... + βnX^n. By including these higher-degree terms, Polynomial Regression can fit a wide range of curvature in the data, thus effectively capturing

the non-linear relationship between the input and output variables. This flexibility makes it suitable for datasets where the relationship between variables is not simply linear.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 8

Problem Statement

To apply Polynomial Linear Regression on the given dataset.

Code

```
# importing required libraries
import pandas as pd
import numpy as np
```

[1] ✓ 0.7s Python

```
# data
X = [-3, -2, -1, -0.2, 1, 3]
y = [0.9, 0.8, 0.4, 0.2, 0.1, 0]

# degree of polynomial
degree = 2

# calculating the number of data points
n = len(X)
print("Number of data points:", n)

```

[2] ✓ 0.0s Python

```
Number of data points: 6
```

```
# calculating the summations for X, Y, and powers of X
sum_x = sum(X)
sum_y = sum(y)
sum_x2 = sum([x**2 for x in X])
sum_x3 = sum([x**3 for x in X])
sum_x4 = sum([x**4 for x in X])

# calculating summations for products of X and Y
sum_xy = sum([x * y_val for x, y_val in zip(X, y)])
sum_x2y = sum([x**2 * y_val for x, y_val in zip(X, y)])

# printing the summations
print("Summations for X, Y, and powers of X")
print("X Sum:", sum_x)
print("Y Sum:", sum_y)
print("X^2 Sum:", sum_x2)
print("X^3 Sum:", sum_x3)
print("X^4 Sum:", sum_x4)
print("Summations for products of X and Y")
print("X*Y Sum:", sum_xy)
print("X^2*Y Sum:", sum_x2y)
```

[3] ✓ 0.0s Python

```
Summations for X, Y, and powers of X
X Sum: -2.2
Y Sum: 2.4000000000000004
X^2 Sum: 24.04
X^3 Sum: -8.008000000000003
X^4 Sum: 180.0016
Summations for products of X and Y
X*Y Sum: -4.640000000000015
X^2*Y Sum: 11.808
```

```python
# creating normal equations matrices
A = [[n, sum_x, sum_x2],
     [sum_x, sum_x2, sum_x3],
     [sum_x2, sum_x3, sum_x4]]
B = [sum_y, sum_xy, sum_x2y]

# solving the system of equations
coefficients = np.linalg.solve(A, B)
print("\nCoefficients (a0, a1, a2):", coefficients)
```

[4] ✓ 0.0s Python

```
Coefficients (a0, a1, a2): [ 0.22906556 -0.16280041  0.02776397]
```

```python
# define the polynomial function
def polynomial(x):
    return coefficients[0] + coefficients[1] * x + coefficients[2] * x**2
```

[5] ✓ 0.0s Python

```python
# evaluating the polynomial function at each data point
predicted_y = [polynomial(x_val) for x_val in X]
```

[6] ✓ 0.0s Python

```python
# display the original data and the predicted values
print("\nOriginal and Predicted Data")
for x_val, orig_y, pred_y in zip(X, y, predicted_y):
    print(f"X: {x_val}, Original Y: {orig_y}, Predicted Y: {pred_y}")
```

[7] ✓ 0.0s Python

```
Original and Predicted Data
X: -3, Original Y: 0.9, Predicted Y: 0.967342489931803
X: -2, Original Y: 0.8, Predicted Y: 0.6657222483828868
X: -1, Original Y: 0.4, Predicted Y: 0.419629938191314
X: -0.2, Original Y: 0.2, Predicted Y: 0.2627362006153428
X: 1, Original Y: 0.1, Predicted Y: 0.09402911188019804
X: 3, Original Y: 0, Predicted Y: -0.009459989001544655
```

EXPERIMENT NO. 9

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 20.10.2023
Faculty Signature:
Marks:

Objective(s):

- Study Logistic Regression.
- How Logistic Regression is used to solve classification problems.

Outcome:

Students will be familiarized with Logistic Regression and performance metrics to calculate its performance on the given dataset.

Problem Statement:

To solve classification problems using Logistic Regression.

Background Study:

Logistic regression is a classification technique which helps to predict the probability of an outcome that can only have two values. Logistic Regression is used when the dependent variable (target) is categorical. A logistic regression produces a logistic curve, which is limited to values between 0 and 1.
confusion_matrix and accuracy_score functions are used to evaluate the model. A confusion matrix is visualized using a heatmap from the seaborn package, and Boxplot from seaborn is used to check for the outliers in the dataset. The confusion matrix consists of the matrix elements with True Positive, True Negative, False Positive, and False Negative values.

Question Bank:

1. What is Logistic Regression?
   Logistic regression is a statistical method used for binary classification problems. It models the probability of a binary outcome (like yes/no, true/false, success/failure) using a logistic function. Despite its name, it's a classification algorithm, not a regression algorithm. It uses a logistic function to squeeze the output of a linear equation between 0 and 1. This output is then interpreted as the probability that a given input point belongs to a certain class.

2. How Logistic Regression is used for solving classification problems?

Logistic regression is employed in classification by predicting the probability that a given data entry belongs to a category. It takes input variables and assigns weights to them. The sum of the weighted inputs is then transformed by the logistic (sigmoid) function to predict the probability. For instance, it can be used to classify emails as spam or not spam by weighing features like word frequency. The decision about the class is made based on whether the predicted probability is above or below a certain threshold (typically 0.5).

3. Why sigmoid function is used in it?

The sigmoid function is crucial in logistic regression because it transforms the output of the linear model (which can have any value from negative to positive infinity) into a probability value between 0 and 1. This is necessary for binary classification. The sigmoid function has an 'S' shaped curve. It outputs a value close to 1 if the input is very positive, close to 0 if the input is very negative, and around 0.5 if the input is near zero. This characteristic makes it appropriate for binary classification tasks.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 9

Problem Statement

To solve classification problems using Logistic Regression.

Code

```python
# importing required libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
from sklearn.datasets import load_iris
```

[34] Python

```python
# loading the dataset
dataset = pd.read_csv('wine_dataset.csv')
print(dataset.head().to_markdown())
```

[35] Python

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alco
0	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	
1	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	
2	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	
3	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	
4	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	

```python
# splitting the dataset into dependent and independent variables
X = dataset.drop('style', axis=1)  # Independent variables
y = dataset['style']               # Dependent variable
```

[36] Python

```python
# print number of unique values in y
print(f"Number of unique values in y: {y.nunique()}")
```

[37] Python

```
Number of unique values in y: 2
```

```python
# splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[38] Python

```python
# implementing the logistic regression model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
```

[39] Python

```
C:\Users\mainp\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression(max_iter=200)
```

```python
# evaluating the model
predictions = model.predict(X_test)

# confusion Matrix
conf_matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:\n", conf_matrix)

# true positive, false positive, false negative, true negative
for i in range(len(conf_matrix)):
    TP = conf_matrix[i, i]
    FP = conf_matrix[:, i].sum() - TP
    FN = conf_matrix[i, :].sum() - TP
    TN = conf_matrix.sum() - (TP + FP + FN)

    print('Class {} -- TP {}, FP {}, FN {}, TN {}'.format(i, TP, FP, FN, TN))

# accuracy
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Precision, Recall, and F1 Score
precision = precision_score(y_test, predictions, average='weighted')
recall = recall_score(y_test, predictions, average='weighted')
f1 = f1_score(y_test, predictions, average='weighted')

print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

[40] Python

```
Confusion Matrix:
 [[305  11]
 [  7 977]]
Class 0 -- TP 305, FP 7, FN 11, TN 977
Class 1 -- TP 977, FP 11, FN 7, TN 305
Accuracy: 0.9861538461538462
Precision: 0.9861190698640091
Recall: 0.9861538461538462
F1 Score: 0.9861237928748684
```

EXPERIMENT NO. 10

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 10.11.2023
Faculty Signature:
Marks:

Objective(s):

- Study K-Nearest Neighbor algorithm (KNN).
- Understand the working principle behind KNN.

Outcome:

Students will be familiarized with classification technique using KNN.

Problem Statement:

To solve classification problems using KNN classification.

Background Study:

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well –

- Lazy learning algorithm – KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
- Non-parametric learning algorithm – KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. The most common parameter used to perform matching is Euclidean distance between the points.

The sklearn library has provided a layer of abstraction on top of Python. Therefore, in order to make use of the KNN algorithm, it's sufficient to create an instance of KNeighborsClassifier. By default, the KNeighborsClassifier looks for the **5** nearest neighbors. We must explicitly tell the classifier to use Euclidean distance for determining the proximity between neighboring points.

Question Bank:

1. What is KNN classifier?
   K-Nearest Neighbors (KNN) is a simple, versatile, and easy-to-implement supervised machine learning algorithm used for classification (and regression). It classifies a new data point based on the majority class among its 'k' nearest neighbors. 'k' is a user-defined constant and the neighbors are taken from a set of objects for which the class (label) is known. This means that a new data point is assigned a value based on how closely it resembles the points in the training set.

2. How KNN makes use of Euclidean distance to calculate nearest neighbor?
   For two points P(x1, y1) and Q(x2, y2), the Euclidean distance is:
   Euclidean distance = sqrt((x2 - x1)^2 + (y2 - y1)^2)

   In a multi-dimensional space, for points P and Q with coordinates (x1, y1, z1, ...) and (x2, y2, z2, ...), the Euclidean distance is:
   Euclidean distance = sqrt((x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2 + ...)

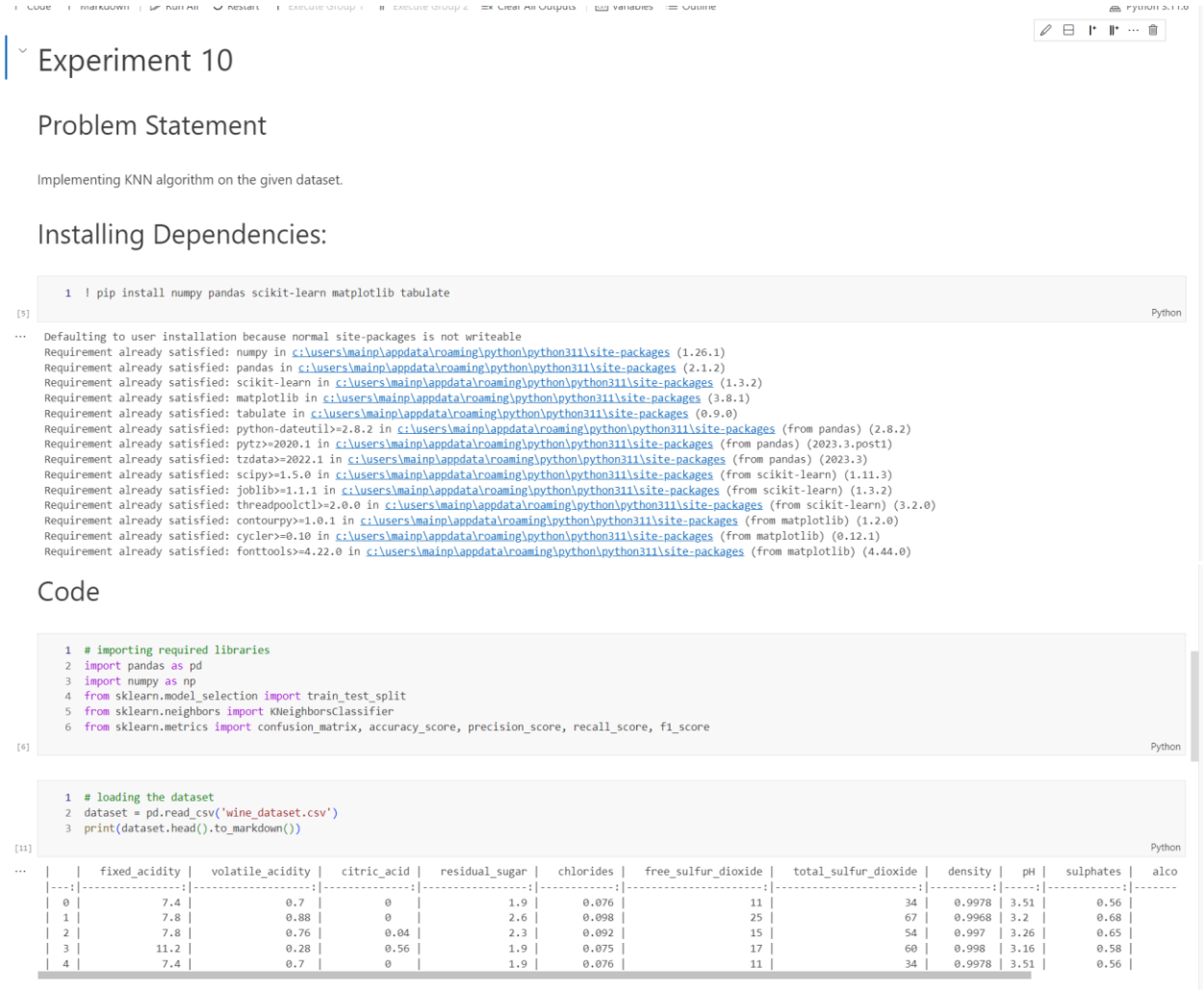
3. What are the other distances that can be used for nearest neighbor?
   - Manhattan Distance: Also known as city block distance, calculates the sum of the absolute differences of their Cartesian coordinates. Used in grid-like path calculations.
   - Chebyshev Distance: The maximum distance along any coordinate dimension. Useful in chess-like or grid-based games.
   - Minkowski Distance: A generalized form of distance calculation. It becomes Manhattan or Euclidean distance with different parameter values.
   - Hamming Distance: Measures the distance between two strings of equal length by counting the number of positions at which the corresponding symbols are different. Commonly used in text or categorical data analysis.

4. What are the various performance metrics used for classification problems?
   - Accuracy: The ratio of correctly predicted instances to the total instances.
   - Precision: The ratio of correctly predicted positive observations to the total predicted positive observations.
   - Recall (Sensitivity): The ratio of correctly predicted positive observations to all observations in the actual class.
   - F1 Score: The weighted average of Precision and Recall, useful when the class distribution is uneven.
   - Confusion Matrix: A table used to describe the performance of a classification model. It outlines the correct and incorrect predictions across different classes.
   - ROC Curve and AUC: The receiver operating characteristic curve shows the diagnostic ability of a binary classifier, and the Area Under the Curve (AUC) represents measure of separability.

- Log Loss: A performance metric for evaluating the predictions of probabilities of membership to a given class.
- Matthews Correlation Coefficient (MCC): A coefficient considering true and false positives and negatives, suitable for unbalanced datasets.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs



Experiment 10

Problem Statement

Implementing KNN algorithm on the given dataset.

Installing Dependencies:

```
[5]  ! pip install numpy pandas scikit-learn matplotlib tabulate
```

Python

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: numpy in c:\users\mainp\appdata\roaming\python\python311\site-packages (1.26.1)
Requirement already satisfied: pandas in c:\users\mainp\appdata\roaming\python\python311\site-packages (2.1.2)
Requirement already satisfied: scikit-learn in c:\users\mainp\appdata\roaming\python\python311\site-packages (1.3.2)
Requirement already satisfied: matplotlib in c:\users\mainp\appdata\roaming\python\python311\site-packages (3.8.1)
Requirement already satisfied: tabulate in c:\users\mainp\appdata\roaming\python\python311\site-packages (0.9.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\mainp\appdata\roaming\python\python311\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\mainp\appdata\roaming\python\python311\site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\mainp\appdata\roaming\python\python311\site-packages (from pandas) (2023.3)
Requirement already satisfied: scipy>=1.5.0 in c:\users\mainp\appdata\roaming\python\python311\site-packages (from scikit-learn) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in c:\users\mainp\appdata\roaming\python\python311\site-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\mainp\appdata\roaming\python\python311\site-packages (from scikit-learn) (3.2.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\mainp\appdata\roaming\python\python311\site-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\mainp\appdata\roaming\python\python311\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\mainp\appdata\roaming\python\python311\site-packages (from matplotlib) (4.44.0)
```

Code

```
[6]  # importing required libraries
     import pandas as pd
     import numpy as np
     from sklearn.model_selection import train_test_split
     from sklearn.neighbors import KNeighborsClassifier
     from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
```

Python

```
[11] # loading the dataset
     dataset = pd.read_csv('wine_dataset.csv')
     print(dataset.head().to_markdown())
```

Python

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alco
0	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	
1	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	
2	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	
3	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	
4	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	

```python
# splitting the dataset into dependent and independent variables
X = dataset.drop('style', axis=1)  # Independent variables
y = dataset['style']               # Dependent variable
```

[17] Python

```python
# print number of unique values in y
print(f"Number of unique values in y: {y.nunique()}")
```

[18] Python

```
Number of unique values in y: 2
```

```python
# splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[19] Python

```python
# implementing the logistic regression model
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
```

[20] Python

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

```python
# evaluating the model
predictions = model.predict(X_test)

# confusion Matrix
conf_matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:\n", conf_matrix)

# true positive, false positive, false negative, true negative
for i in range(len(conf_matrix)):
    TP = conf_matrix[i, i]
    FP = conf_matrix[:, i].sum() - TP
    FN = conf_matrix[i, :].sum() - TP
    TN = conf_matrix.sum() - (TP + FP + FN)

    print('Class {} -- TP {}, FP {}, FN {}, TN {}'.format(i, TP, FP, FN, TN))

# accuracy
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Precision, Recall, and F1 Score
precision = precision_score(y_test, predictions, average='weighted')
recall = recall_score(y_test, predictions, average='weighted')
f1 = f1_score(y_test, predictions, average='weighted')

print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

[22] Python

```
Confusion Matrix:
 [[251  43]
 [ 33 973]]
Class 0 -- TP 251, FP 33, FN 43, TN 973
Class 1 -- TP 973, FP 43, FN 33, TN 251
Accuracy: 0.9415384615384615
Precision: 0.9409701973195929
Recall: 0.9415384615384615
F1 Score: 0.9411773025334574
```

EXPERIMENT NO. 11

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 10.11.2023
Faculty Signature:
Marks:

Objective(s):

- Understand and study Naïve Bayes (NB) Classifier.
- Understand Naïve Bayes theorem behind it.

Outcome:

Students will be familiarized with NB classification technique.

Problem Statement:

To solve classification problems using Naïve Bayes.

Background Study:

Naïve Bayes Classifier is a probabilistic classifier and is based on Bayes Theorem.
In Machine learning, a classification problem represents the selection of the Best Hypothesis given the data.
Given a new data point, we try to classify which class label this new data instance belongs to. The prior knowledge about the past data helps us in classifying the new data point. The Naïve Bayes theorem:

$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)}$$

gives us the probability of Event A to happen given that event B has occurred.
The receiver operating characteristic curve also known as roc_curve is a plot that tells about the interpretation potential of a binary classifier system. It is plotted between the true positive rate and the false positive rate at different thresholds.

Question Bank:

1. What is Bayes theorem?
   Bayes theorem is a fundamental theorem in probability theory. It describes the probability of an event, based on prior knowledge of conditions that might be related to the event.
   Mathematically, it's expressed as:

   P(A|B) = (P(B|A) * P(A)) / P(B)

In this formula:

P(A|B) is the probability of event A given that B is true.
P(B|A) is the probability of event B given that A is true.
P(A) and P(B) are the probabilities of observing A and B independently of each other.

2. How Naïve Bayes classifier helps for solving classification problems?
   This classifier applies Bayes theorem with the "naïve" assumption of conditional independence between every pair of features given the value of the class variable. In simpler terms, it assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. This simplicity allows for the easy computation of probabilities and modeling of the classifier. Naïve Bayes is particularly suited for large datasets and performs surprisingly well, especially in cases like spam filtering and document classification, despite its simplifying assumption.

3. What is the condition on features that should be fulfilled for successful application of Naïve Bayes method?
   The key condition for the successful application of the Naïve Bayes method is the assumption of feature independence. It assumes that all the features are independent of each other given the class label. This means that the effect of a feature value on a given class is independent of the values of other features. While this is a strong and often unrealistic assumption in real-world data, Naïve Bayes can still perform well in practice, particularly when the data dimensions are high, as the class-conditional independence assumption often approximates reality well enough for effective classification.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 11

Problem Statement

To solve classification problems using Naïve Bayes.

Code

```python
# importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
```

[9] Python

```python
# loading the dataset
dataset = pd.read_csv('wine_dataset.csv')
print(dataset.head().to_markdown())
```

[10] Python

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alco
0	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	
1	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	
2	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	
3	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	
4	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	

```python
# splitting the dataset into dependent and independent variables
X = dataset.drop('style', axis=1)  # Independent variables
y = dataset['style']               # Dependent variable
```

[11] Python

```python
# print number of unique values in y
print(f"Number of unique values in y: {y.nunique()}")
```

[12] Python

```
Number of unique values in y: 2
```

```python
# splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[13] Python

```
# training the model
model = GaussianNB()
model.fit(X_train, y_train)
```

[14] Python

```
GaussianNB()
```

```
# evaluating the model
predictions = model.predict(X_test)

# confusion Matrix
conf_matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:\n", conf_matrix)

# true positive, false positive, false negative, true negative
for i in range(len(conf_matrix)):
    TP = conf_matrix[i, i]
    FP = conf_matrix[:, i].sum() - TP
    FN = conf_matrix[i, :].sum() - TP
    TN = conf_matrix.sum() - (TP + FP + FN)

    print('Class {} -- TP {}, FP {}, FN {}, TN {}'.format(i, TP, FP, FN, TN))

# accuracy
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Precision, Recall, and F1 Score
precision = precision_score(y_test, predictions, average='weighted')
recall = recall_score(y_test, predictions, average='weighted')
f1 = f1_score(y_test, predictions, average='weighted')

print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

[15] Python

```
Confusion Matrix:
 [[317   6]
 [ 24 953]]
Class 0 -- TP 317, FP 24, FN 6, TN 953
Class 1 -- TP 953, FP 6, FN 24, TN 317
Accuracy: 0.9769230769230769
Precision: 0.9778109575849158
Recall: 0.9769230769230769
F1 Score: 0.9771285874585437
```

EXPERIMENT NO. 12

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 17.11.2023
Faculty Signature:
Marks:

Objective(s):

- Understand and study Support Vector Machines (SVM).
- To study how linear hyperplane is calculated to differentiate between two classes.
- Basic understanding of the different variants of SVM.

Outcome:

Students will be familiarized with Support Vector Machines classifier.

Problem Statement:

To solve classification problems using SVM.

Background Study:

In machine learning, support-vector machines (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.
Developed at AT&T Bell Laboratories by Vladimir Vapnik with colleagues, SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or VC theory proposed by Vapnik (1982, 1995) and Chervonenkis (1974).
Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.
In Python, scikit-learn is a widely used library for implementing machine learning algorithms. SVM is also available in the scikit-learn library and we follow the same structure for using it(Import library, object creation, fitting model and prediction).
Tuning the parameters' values for machine learning algorithms effectively improves model performance. Let's look at the list of parameters available with SVM.

Question Bank:

1. What is SVM?

   Support Vector Machine (SVM) is a powerful and versatile supervised machine learning algorithm used for both classification and regression tasks. It is particularly well-suited for classification of complex but small- or medium-sized datasets. The main idea behind SVM is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points. To separate the classes, SVM looks for the hyperplane that has the largest margin, i.e., the maximum distance between data points of both classes. In cases where data is not linearly separable, SVM uses a kernel function to transform the data into a higher dimension where a hyperplane can be used to separate the classes.

2. What are the advantages of using SVM over other classifiers?

   Effective in High Dimensional Spaces: SVM is effective in cases where the number of dimensions is greater than the number of samples, which can be a challenge for other classifiers.

   Memory Efficient: It uses a subset of training points (support vectors), which makes it memory efficient.

   Versatility: The ability to use different kernel functions is one of its biggest strengths. This makes it versatile, capable of handling linear and non-linear relationships.

   Robustness: Especially effective when the data has a clear margin of separation. It is less prone to overfitting when compared to some other algorithms.

3. What do you mean by support vectors?

   Support vectors are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. These are the critical elements of the training set because they are the most difficult to classify and they define the margin. The position and orientation of the hyperplane are determined by these points. In simple terms, support vectors help in defining the best decision boundary (hyperplane) that separates different classes in the SVM classifier.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 12

Problem Statement

To solve classification problems using SVM.

Code

```python
# importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
```

[1] Python

```python
# loading the dataset
dataset = pd.read_csv('wine_dataset.csv')
print(dataset.head().to_markdown())
```

[2] Python

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alco
0	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	
1	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	
2	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	
3	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	
4	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	

```python
# splitting the dataset into dependent and independent variables
X = dataset.drop('style', axis=1)  # Independent variables
y = dataset['style']               # Dependent variable
```

[3] Python

```python
# print number of unique values in y
print(f"Number of unique values in y: {y.nunique()}")
```

[4] Python

```
Number of unique values in y: 2
```

```python
# splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[5] Python

```python
# training the model
model = svm.SVC()
model.fit(X_train, y_train)
```

[6] Python

```
SVC()
```

```python
# evaluating the model
predictions = model.predict(X_test)

# confusion Matrix
conf_matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:\n", conf_matrix)

# true positive, false positive, false negative, true negative
for i in range(len(conf_matrix)):
    TP = conf_matrix[i, i]
    FP = conf_matrix[:, i].sum() - TP
    FN = conf_matrix[i, :].sum() - TP
    TN = conf_matrix.sum() - (TP + FP + FN)

    print('Class {} -- TP {}, FP {}, FN {}, TN {}'.format(i, TP, FP, FN, TN))

# accuracy
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Precision, Recall, and F1 Score
precision = precision_score(y_test, predictions, average='weighted')
recall = recall_score(y_test, predictions, average='weighted')
f1 = f1_score(y_test, predictions, average='weighted')

print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

[7] Python

```
Confusion Matrix:
 [[255  62]
 [ 17 966]]
Class 0 -- TP 255, FP 17, FN 62, TN 966
Class 1 -- TP 966, FP 62, FN 17, TN 255
Accuracy: 0.9392307692307692
Precision: 0.9391550059862317
Recall: 0.9392307692307692
F1 Score: 0.9375892782980793
```

EXPERIMENT NO. 13

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL236-IAIML-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 17.11.2023
Faculty Signature:
Marks:

Objective(s):

- Understand and study Decision Trees for classification problems.
- Study about the information gain used to create decision trees.

Outcome:

Students will be familiarized with creation of decision trees.

Problem Statement:

Apply Decision Tree classifier for solving classification problems.

Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decision trees are the most powerful algorithms that falls under the category of supervised algorithms.

They can be used for both classification and regression tasks. The two main entities of a tree are decision nodes, where the data is split and leaves, where we got outcome.

Question Bank:

1. What is a decision tree?
   A decision tree is a tree-like model of decisions and their possible consequences. It is a flowchart-like structure where each internal node represents a "test" on an attribute (e.g., whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from the root to the leaf represent classification rules.

2. How decision tree is created to solve problems?
   To create a decision tree, the algorithm starts at the tree's root and splits the data on the feature that results in the largest information gain (IG). In simpler terms, it chooses the feature that best separates the data. The algorithm then repeats this process recursively for each child until one of the conditions for stopping is met. This could be a maximum depth of the tree, reaching a minimum number of data points in a node, or when all data points in a node belong to the same class. Common algorithms used for constructing decision trees include ID3, C4.5, CART, and CHAID.

3. List out the advantages and disadvantages of Decision Tree Classifiers?

Advantages:

- Simple to Understand and Interpret: Trees can be visualized, making them easy to understand.
- Requires Little Data Pre-processing: No need for normalizing data.
- Can Handle Both Numerical and Categorical Data: Other techniques are usually specialized in analyzing datasets that have only one type of variable.
- Versatile: Can be used for both classification and regression tasks.

Disadvantages:

- Overfitting: Decision trees can create over-complex trees that do not generalize well from the training data.
- Not Robust: A small change in the data can lead to a large change in the structure of the decision tree.
- Can be Biased: If some classes dominate, the decision tree might create a biased tree. Hence, it's recommended to balance the dataset before creating a decision tree.
- Problems with Non-linear Data: Decision trees do not work well with non-linear data where the relationship between features is not well defined.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 13

Problem Statement

Apply Decision Tree classifier for solving classification problems.

Code

```python
# importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
```

[1] Python

```python
# loading the dataset
dataset = pd.read_csv('wine_dataset.csv')
print(dataset.head().to_markdown())
```

[2] Python

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density	pH	sulphates	alco
0	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	
1	7.8	0.88	0	2.6	0.098	25	67	0.9968	3.2	0.68	
2	7.8	0.76	0.04	2.3	0.092	15	54	0.997	3.26	0.65	
3	11.2	0.28	0.56	1.9	0.075	17	60	0.998	3.16	0.58	
4	7.4	0.7	0	1.9	0.076	11	34	0.9978	3.51	0.56	

```python
# splitting the dataset into dependent and independent variables
X = dataset.drop('style', axis=1)  # Independent variables
y = dataset['style']               # Dependent variable
```

[3] Python

```python
# print number of unique values in y
print(f"Number of unique values in y: {y.nunique()}")
```

[4] Python

```
Number of unique values in y: 2
```

```python
# splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[5] Python

```python
# training the model
tree_model = DecisionTreeClassifier(random_state=42)
tree_model.fit(X_train, y_train)
```

[6] Python

```
DecisionTreeClassifier(random_state=42)
```

```python
# evaluating the model
predictions = tree_model.predict(X_test)

# confusion Matrix
conf_matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:\n", conf_matrix)

# true positive, false positive, false negative, true negative
for i in range(len(conf_matrix)):
    TP = conf_matrix[i, i]
    FP = conf_matrix[:, i].sum() - TP
    FN = conf_matrix[i, :].sum() - TP
    TN = conf_matrix.sum() - (TP + FP + FN)

    print('Class {} -- TP {}, FP {}, FN {}, TN {}'.format(i, TP, FP, FN, TN))
```

```python
# accuracy
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Precision, Recall, and F1 Score
precision = precision_score(y_test, predictions, average='weighted')
recall = recall_score(y_test, predictions, average='weighted')
f1 = f1_score(y_test, predictions, average='weighted')

print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

[7] Python

```
Confusion Matrix:
 [[290   8]
 [ 12 990]]
Class 0 -- TP 290, FP 12, FN 8, TN 990
Class 1 -- TP 990, FP 8, FN 12, TN 290
Accuracy: 0.9846153846153847
Precision: 0.9847129817454573
Recall: 0.9846153846153847
F1 Score: 0.984651282051282
```