AN INTERNSHIP REPORT

on

Data Science and Machine Learning

Submitted in partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

BANDARU DEVA SRIRAMA SAI GANESH

(21021A0548)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING KAKINADA (A)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA

KAKINADA - 533 003, ANDHRA PRADESH, INDIA

2021 - 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING KAKINADA (A)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA

KAKINADA - 533 003, ANDHRA PRADESH, INDIA

2021 - 2025

CERTIFICATE

This is to certify that this internship project report entitled "**DATA SCIENCE AND MACHINE LEARNING**" is a Bonafide record of the work submitted by **BANDARU DEVA SRIRAMA SAI GANESH** bearing the roll number **21021A0548**, in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING** to the **UCEK(A)**, **JNTUK**, Kakinada, Andhra Pradesh, India. It has been found satisfactory and hereby approved for submission.

Signature of Head of the Department

Dr. N. Ramakrishnaiah

Professor & HOD

Department of CSE

UCEK(A)

JNTUK, Kakinada

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING KAKINADA (A)

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA

KAKINADA - 533 003, ANDHRA PRADESH, INDIA

2021 - 2025

DECLARATION

I hereby declare that the internship project work entitled "**DATA SCIENCE AND MACHINE LEARNING,**" submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from University College of Engineering, Kakinada, is based on the work I undertook during my internship at HDLC Technologies.

I further declare that this project is a record of original work completed by me during the internship period. All information, software, and references used in this project are duly acknowledged and cited. I am fully responsible for the contents of this project report.

With Gratitude

BANDARU DEVA SRIRAMA SAI GANESH

21021A0548

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to **HDLC Technologies** for providing me with the opportunity to undertake an internship in the field of **Data Science and Machine Learning**. This experience has significantly enhanced my technical proficiency and deepened my practical understanding of real-world applications in data analytics and clustering techniques. It enabled me to apply theoretical knowledge to practical scenarios and strengthened my ability to work independently using industry-relevant tools and methodologies.

I am deeply thankful to **Dr. N. Mohan Rao**, Principal, *University College of Engineering Kakinada*, for his constant support and motivation throughout this internship. I also extend my sincere thanks to **Dr. N. Ramakrishnaiah**, Professor and Head of the Department of *Computer Science and Engineering*, for his valuable guidance and encouragement.

My appreciation extends to all my **faculty members and mentors**, as well as the team at **HDLC Technologies**, for their timely feedback and continued support, which helped me stay focused and grow professionally.

Finally, I am truly grateful to my **family, friends, and peers** for their unwavering support and encouragement throughout this learning journey.

INTERNSHIP OFFER LETTER

HDLC TECHNOLOGIES

29/04/2023

INTERNSHIP OFFER LETTER

Dear Bandaru Deva Srirama Sai Ganesh,

Congratulations! We are happy to offer you the position of Software Developer Intern in our company, HDLC Technologies. HDLC is a technology company, which aims to change the way businesses operate on a daily basis via our AI Products. We hope that your contribution will enable us to cross many frontiers together.

We strongly believe that people have the power to change the world. We also believe that you have the potential to drive change. This offer letter acknowledges your ability to create long-term impact.

At HDLC you will have the opportunity to learn and develop cutting edge technology that can revolutionize our clients' businesses. You will be at the heart of a young, vibrant and multi cultural team that is curious and enterprising.

You will be working from home and report to the designated lead. You will be working as Intern in the domain of **Data Science and Machine Learning** and develop the applications as per the requirements. We will endeavor to help you every step of the way in crafting a fulfilling experience, rich in learning.

Period of Internship : **02/05/2023 to 30/06/2023**

Upon successful completion of the Internship program and performance review, you will be onboarded into HDLC Technologies.

We look forward to having you at HDLC. Your working hours and other T&C related to your internship will be communicated to you on the day of your joining.

Here's to exciting days of change! Welcome to HDLC and good luck!

Please confirm your acceptance of the offer asap.

For HDLC Technologies

Authorized signatory

HDLC TECHNOLOGIES CHENNAI

HDLC Technologies
No. 66, Pavithra Square, Chennai-600061, Ph: 8072933282
Email: support@hdlctech.in, hdlctech@gmail.com

INTERNSHIP CERTIFICATE

ABSTRACT

This internship report outlines the practical implementation of machine learning techniques through two structured projects at HDLC Technologies. The focus was on applying core concepts in supervised and unsupervised learning to extract insights from real-world data.

The mini project involved classifying the Iris dataset using the K-Nearest Neighbors (KNN) algorithm. It encompassed data preprocessing, feature scaling, model training, and evaluation, serving as a hands-on introduction to supervised learning and performance measurement.

The major project addressed customer segmentation through K-Means clustering. By analyzing customer income and spending behavior, the project aimed to identify distinct consumer groups. Key steps included exploratory data analysis, feature standardization, cluster optimization via the elbow method, and visualization of segmentation results.

Tools such as pandas, NumPy, matplotlib, seaborn, and scikit-learn were used throughout the development process. These projects strengthened both analytical and technical competencies, emphasizing the role of data-driven strategies in solving practical business problems.

Keywords: Machine Learning, Data Science, KNN, K-Means Clustering, Iris Dataset, Customer Segmentation, Python, pandas, NumPy, scikit-learn, Data Preprocessing, Model Evaluation

TABLE OF CONTENTS

CHAPTER - 01

INTRODUCTION

The integration of **data science** and **machine learning (ML)** into modern industries has revolutionized the way organizations process data, make decisions, and create value. Data-driven methodologies are no longer optional—they are essential for optimizing business operations, enhancing customer experience, predicting trends, and automating complex tasks. Within this ecosystem, data science provides the analytical and statistical foundation to extract insights from raw data, while machine learning focuses on building adaptive systems that improve through experience.

During this internship, I had the opportunity to explore these domains through two targeted projects that provided hands-on experience with both supervised and unsupervised learning techniques. The first project involved the classification of the Iris flower dataset using the **K-Nearest Neighbors (KNN)** algorithm, a fundamental supervised learning model. This task not only introduced the process of labeled data classification but also emphasized model evaluation metrics like accuracy and confusion matrix interpretation.

The second project addressed customer segmentation using **K-Means clustering**, a widely-used unsupervised learning algorithm. By analyzing patterns in customers' annual income and spending behavior, this project demonstrated how clustering techniques can be used to derive meaningful groupings in unlabeled data—insights that are especially valuable for businesses seeking to optimize marketing strategies and improve customer targeting.

Both projects followed a rigorous development pipeline: **data preprocessing, feature selection, algorithm implementation, model training, evaluation, and visualization**. This iterative process provided a deep understanding of the end-to-end lifecycle of a machine learning workflow, mirroring practices used in industry-level projects.

Python was the core programming language used for development, primarily due to its versatility and strong ecosystem of open-source

libraries. Tools such as **NumPy**, **pandas**, **matplotlib**, and **seaborn** were employed for data manipulation and visualization, while **scikit-learn** served as the backbone for model building and evaluation.

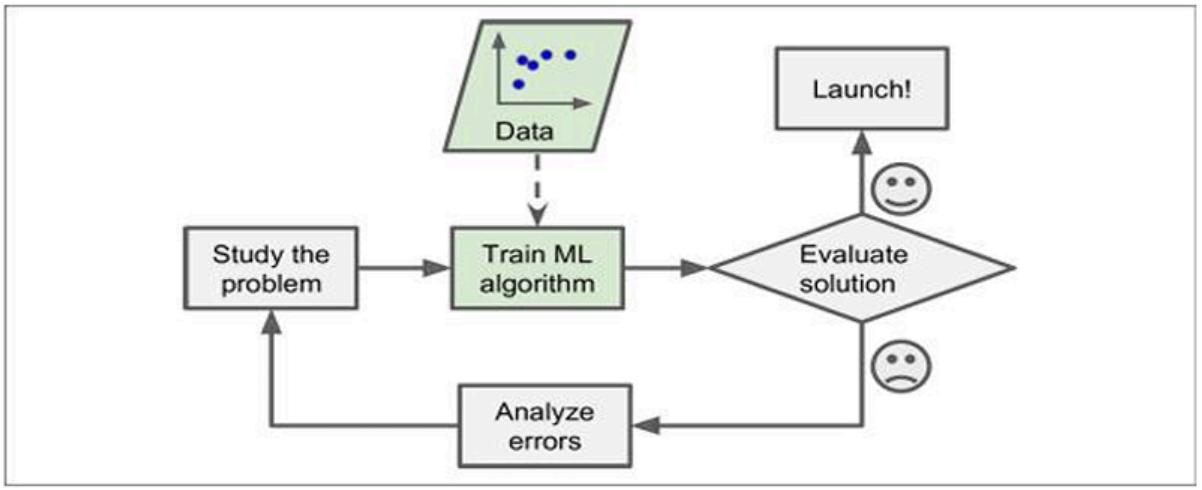


Figure: Machine Learning Approach

CHAPTER - 02

LITERATURE REVIEW

Data science is an interdisciplinary field that integrates statistical methods, programming, and domain expertise to derive insights from data. A core component of data science is machine learning (ML), which enables systems to learn patterns from existing data and make informed predictions or decisions without being explicitly programmed. Mitchell (1997) defines machine learning as "a computer program [that] is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [1]. Machine learning is extensively applied in real-world domains including finance, healthcare, marketing, and security, with classification and clustering being among the most utilized tasks.

In supervised learning, algorithms are trained using labeled data, and one commonly used method is the K-Nearest Neighbors (KNN) algorithm. Originally proposed by Fix and Hodges in 1951 [2] and later formalized by Cover and Hart in 1967 [3], KNN is a non-parametric algorithm that classifies new instances based on the majority class among their k-nearest neighbors in feature space. It is particularly suitable for small datasets and is widely used in education and benchmarking. A typical example is its application on the Iris dataset, introduced by Fisher in 1936 [4], which remains a standard dataset for evaluating classification algorithms. While KNN is intuitive and effective, it can be computationally expensive for large datasets and is sensitive to the scale of features.

In contrast, unsupervised learning techniques such as K-Means clustering are used to identify inherent groupings within unlabeled data. K-Means, introduced by Lloyd (1957) and later popularized by MacQueen in 1967 [5], partitions data into k clusters by minimizing the within-cluster variance. It is widely used in customer segmentation, anomaly detection, and image processing. In the major project of this internship, K-Means clustering was applied to customer data to group consumers based on their annual income and spending scores, uncovering insights into behavioral patterns

that could drive targeted marketing strategies. Although K-Means is efficient and easy to implement, it assumes spherical clusters of similar size and requires the number of clusters (k) to be predefined. Techniques like the Elbow Method, Silhouette Score, and Gap Statistic are commonly employed to determine the optimal value of k [6][7].

Beyond academic exploration, both KNN and K-Means algorithms have demonstrated their effectiveness across diverse practical domains. KNN is widely used in handwriting recognition systems such as optical character recognition (OCR), where characters are classified based on similarity to labeled examples. It has also proven useful in medical diagnostics, such as classifying patients based on symptom profiles, and in recommendation systems where it can suggest products or content by identifying user similarity. On the other hand, K-Means is extensively applied in customer segmentation within the retail and e-commerce sectors, helping businesses tailor marketing strategies to specific consumer groups. It also plays a role in image compression by reducing color spaces, in anomaly detection for cybersecurity, and in organizing documents or search results by topic. These applications reflect the algorithms' versatility and importance in enabling data-driven decision-making across industries.

The implementation of these algorithms was facilitated by Python and its rich ecosystem of libraries. NumPy was used for numerical operations [8], pandas for data manipulation [9], and matplotlib and seaborn for data visualization [10]. Scikit-learn served as the primary framework for implementing and evaluating both KNN and K-Means [11]. Its simplicity and comprehensive API make it ideal for both research and applied machine learning tasks.

CHAPTER - 03

Mini Project - Classification of Iris Data set

3.1 System Analysis

3.1.1 Problem Statement

The classification of plant species based on morphological measurements is a classic challenge in the field of supervised machine learning. This project aims to develop a machine learning model that can accurately classify iris flowers into one of three species—Setosa, Versicolor, or Virginica—using four input features: sepal length, sepal width, petal length, and petal width. The primary objective is to implement a classification algorithm that learns from labeled data and generalizes well to unseen inputs.

3.1.2 Objectives

- To apply the K-Nearest Neighbors (KNN) algorithm for multi-class classification.
- To preprocess and standardize the input data to ensure optimal model performance.
- To evaluate the performance of the trained model using confusion matrix and accuracy metrics.
- To gain practical exposure to building and validating a supervised learning pipeline using Python and scikit-learn.

3.1.3 Existing System

Manual classification of iris species based on visual inspection or measurement lacks consistency and scalability. Traditional rule-based systems are inflexible and require domain-specific thresholds, making them prone to error, especially with overlapping class boundaries.

3.1.4 Proposed System

The proposed system automates the classification process using a data-driven approach. By training a KNN model on the Iris dataset, the system can predict the species of new flower samples with high accuracy. It

incorporates preprocessing (such as feature scaling), model fitting, and prediction stages, providing a robust framework for classification.

3.1.5 Scope

This project is limited to the Iris dataset and focuses solely on implementing and evaluating a supervised learning model. It does not involve model deployment, hyperparameter tuning automation, or advanced ensemble techniques. However, it demonstrates foundational skills in machine learning pipelines, from preprocessing to evaluation.

3.1.6 Requirements

A. Software Requirements

- Programming Language: Python
- Libraries: pandas, NumPy, matplotlib, seaborn, scikit-learn
- IDE/Tools: Jupyter Notebook or any Python-supported IDE

B. Hardware Requirements

A standard personal computer or laptop with at least:

- 4 GB RAM
- Intel i3 processor or equivalentIntel i3 processor or equivalent
- Internet connection for library installations (initial setup only)

3.2 System Design

3.2.1 Methodology and Architectural Overview

The mini-project on Iris dataset classification follows a structured supervised learning pipeline. The architecture is centered around the modular and sequential execution of core stages in a machine learning workflow: data preprocessing, model training, evaluation, and visualization. Python, along with its associated libraries, serves as the development environment due to its extensive support for scientific computing and machine learning.

Methodology Steps:

1) Dataset Loading:

The Iris dataset is loaded from a reliable source, such as scikit-learn's

built-in datasets module. It contains labeled data with four numerical features and three class labels.

2) **Data Preprocessing:**

The dataset undergoes preprocessing including:

a) Feature normalization using StandardScaler to ensure equal influence of features on distance calculations.

b) Splitting into training and testing sets using train_test_split.

3) **Model Selection and Training:**

The K-Nearest Neighbors (KNN) algorithm is selected for its simplicity and effectiveness in small-scale classification tasks. The model is trained using the training set, with k as a tunable hyperparameter.

4) **Model Evaluation:**

Predictions on the test set are compared to actual labels. Evaluation metrics include:

a) Accuracy Score

b) Confusion Matrix

c) Classification Report (precision, recall, F1-score)

5) **Visualization:**

Data distribution and classification boundaries are visualized using libraries such as matplotlib and seaborn, aiding interpretability.

Architectural View:

The overall system follows a linear, modular architecture implemented in a Jupyter Notebook or Python script. Each stage is encapsulated in standalone blocks—data preprocessing, training, testing, and visualization—making the pipeline transparent, testable, and easy to extend.

3.2.2 Data Description

The Iris dataset is a well-known multivariate dataset introduced by Ronald A. Fisher in 1936. It contains 150 samples of iris flowers from three different species: *Setosa*, *Versicolor*, and *Virginica*. Each sample includes four numerical features:

- ➢ Sepal Length (in cm)
- ➢ Sepal Width (in cm)
- ➢ Petal Length (in cm)
- ➢ Petal Width (in cm)

There are 50 instances for each species, making it a balanced dataset. The dataset is widely used for demonstrating classification algorithms due to its simplicity, well-separated classes, and interpretability.

3.2.3 Algorithm Description

The classification task in this mini-project is performed using the **K-Nearest Neighbors (KNN) algorithm**, a fundamental and intuitive method in supervised machine learning. KNN operates on the principle of proximity, assuming that data points with similar features are likely to belong to the same class.

Working Principle:

KNN is a **non-parametric, instance-based, lazy learning algorithm**. Unlike algorithms that build a general model during training (eager learners), KNN **stores the entire training dataset** and makes predictions only when a new input is provided. This lazy learning behavior makes the training phase fast but can increase prediction time for large datasets.

To classify a new sample:

1) The algorithm calculates the **distance** between the input sample and all samples in the training dataset (commonly using **Euclidean distance**).
2) It identifies the '**k**' **nearest neighbors** to the new sample.
3) It assigns the class label based on **majority voting** among these neighbors.

Mathematical Formulation:

Given a test point x, the distance to a training point x_i is computed using:

$$d(x, x_i) = \sqrt{\sum_{j=1}^{n} (x_j - x_{i,j})^2}$$

where nnn is the number of features.

The predicted class y^ is:

$$\hat{y} = \text{mode}(y_1, y_2, ..., y_k)$$

where y_1,..., y_k are the labels of the k nearest training samples.

Hyperparameter 'k':

The choice of **k** (number of neighbors) significantly impacts the model's performance:

- A small **k** can lead to overfitting (high variance).
- A large **k** may cause underfitting (high bias).

  In this project, **k was chosen empirically**, balancing accuracy and generalization.

Advantages:

- Simple and easy to implement.
- Effective for small and clean datasets.
- No assumptions about data distribution.

Limitations:

- Computationally expensive for large datasets (due to distance calculations at prediction time).
- Sensitive to feature scaling and irrelevant features.
- Performance can degrade with high-dimensional data (curse of dimensionality).

3.2.4 Tools and Technologies Used

This project leverages the Python programming language and several well-established libraries within its data science ecosystem. The following tools and technologies were instrumental in implementing the classification pipeline:

1) **Python 3.x**

   A high-level, general-purpose programming language known for its simplicity and extensive support for data science and ML.

2) **Jupyter Notebook**

An open-source interactive development environment that allows for code execution, visualization, and documentation in a single interface—ideal for iterative ML experimentation.

3) **NumPy**

A library for numerical computing that provides support for arrays, matrices, and a collection of mathematical functions.

4) **pandas**

Used for data manipulation and analysis. It provides data structures like DataFrame, which are convenient for handling structured data.

5) **scikit-learn (sklearn)**

A powerful and widely-used machine learning library offering tools for model selection, preprocessing, evaluation, and a broad range of algorithms—including K-Nearest Neighbors.

6) **matplotlib & seaborn**

Libraries for data visualization. Matplotlib enables creation of static, animated, and interactive plots, while Seaborn is built on top of it to simplify the generation of aesthetically pleasing statistical graphics.

7) **StandardScaler (from sklearn.preprocessing)**

Used for standardizing features by removing the mean and scaling to unit variance, which is crucial for distance-based models like KNN.

3.2.5 Summary of Design Decisions

1) **Why KNN over Other Classifiers:** KNN was chosen for its simplicity, ease of interpretation, and strong performance on small, well-structured datasets like Iris. It requires minimal training time and no assumptions about data distribution.
2) **Use of Normalization:** Feature normalization was essential since KNN relies on distance calculations, and unscaled features could bias the model toward higher-magnitude attributes.
3) **Evaluation Approach:** Accuracy, confusion matrix, and classification report were used to provide a balanced view of model performance, covering both overall correctness and class-wise behavior.

3.3 Implementation

3.3.1 Data Preprocessing

The implementation starts with importing necessary libraries such as pandas for data handling, seaborn and matplotlib.pyplot for visualization, and warnings to suppress unnecessary warning messages. The Iris dataset is then loaded from a CSV file into a pandas DataFrame for further analysis. The full dataset is printed to verify successful loading and to inspect the data structure and values. This step ensures that the data is correctly formatted and provides an overview of all samples before any transformation or modeling is performed.

```
[1]: # Data Pre-processing
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as mtp
     import warnings
     warnings.filterwarnings('ignore')

     # Load the dataset
     iris_dataSet = pd.read_csv("C:\\Users\\banda\\Downloads\\iris_dataset.csv")
     print("iris_dataSet: \n")
     print(iris_dataSet.head())
```

```
iris_dataSet:

   sepal_length  sepal_width  petal_length  petal_width     species
0           5.1          3.5           1.4          0.2      setosa
1           4.9          3.0           1.4          0.2      setosa
2           4.7          3.2           1.3          0.2      setosa
3           4.6          3.1           1.5          0.2      setosa
4           5.0          3.6           1.4          0.2      setosa
```

3.3.2 Exploratory Data Analysis – Summary Statistics

To understand the distribution and characteristics of each feature in the dataset, summary statistics are computed using the describe() function in pandas. This method provides key descriptive metrics such as mean, standard deviation, minimum, maximum, and quartile values for each numerical attribute. This statistical overview helps identify the scale, variance, and potential anomalies within the features, which are crucial for selecting suitable preprocessing techniques and models.

```
[2]: print("\nDisplaying summary statistics: \n")
     print(iris_dataSet.describe())
```

```
Displaying summary statistics:

       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

3.3.3 Dataset Metadata Overview

The info() method is used to display metadata about the dataset, including the number of entries, column names, data types, and memory usage. This step verifies the dataset's completeness by checking for null values and ensures that all features are of appropriate types for further processing. This diagnostic step is crucial for confirming data integrity before performing transformations or applying machine learning algorithms.

```
[3]: print("\nDisplaying information about the dataset: \n")
     print(iris_dataSet.info())
```

```
Displaying information about the dataset:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

3.3.4 Data Visualization

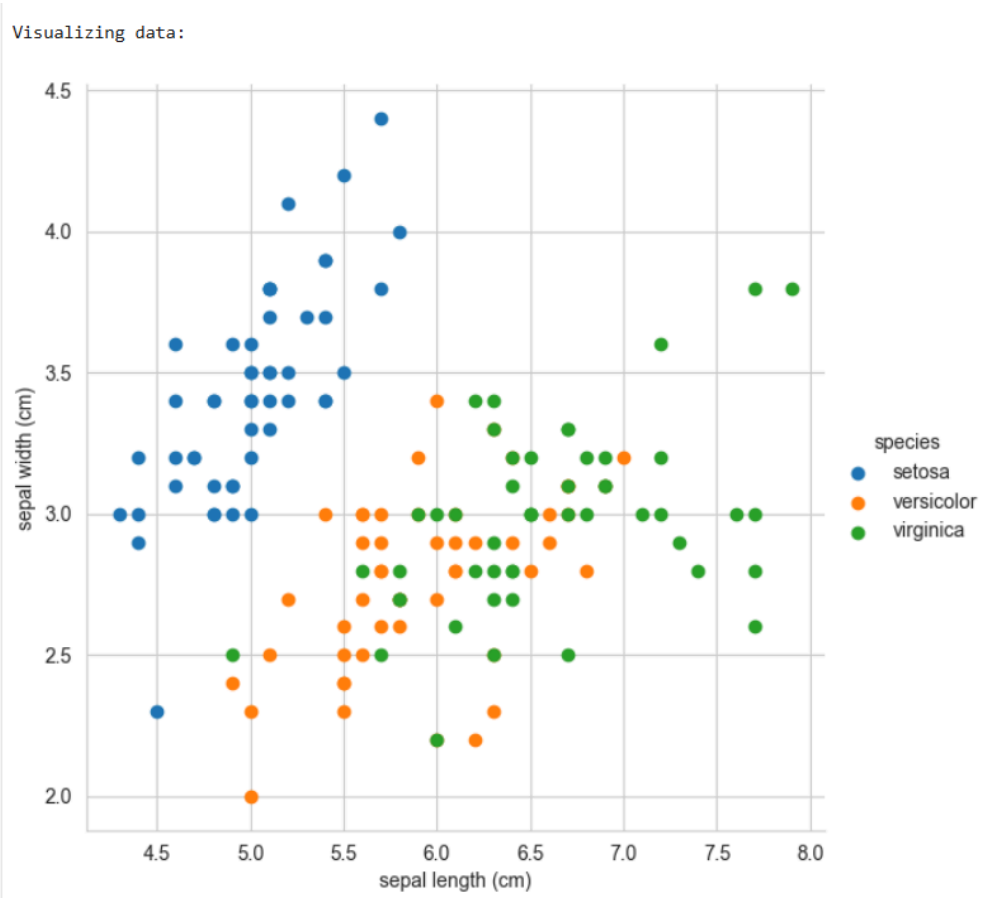
To gain an intuitive understanding of the feature distribution and inter-class relationships, various plots are generated using seaborn and matplotlib. These visualizations help assess the separability of classes and the influence of different features on classification.

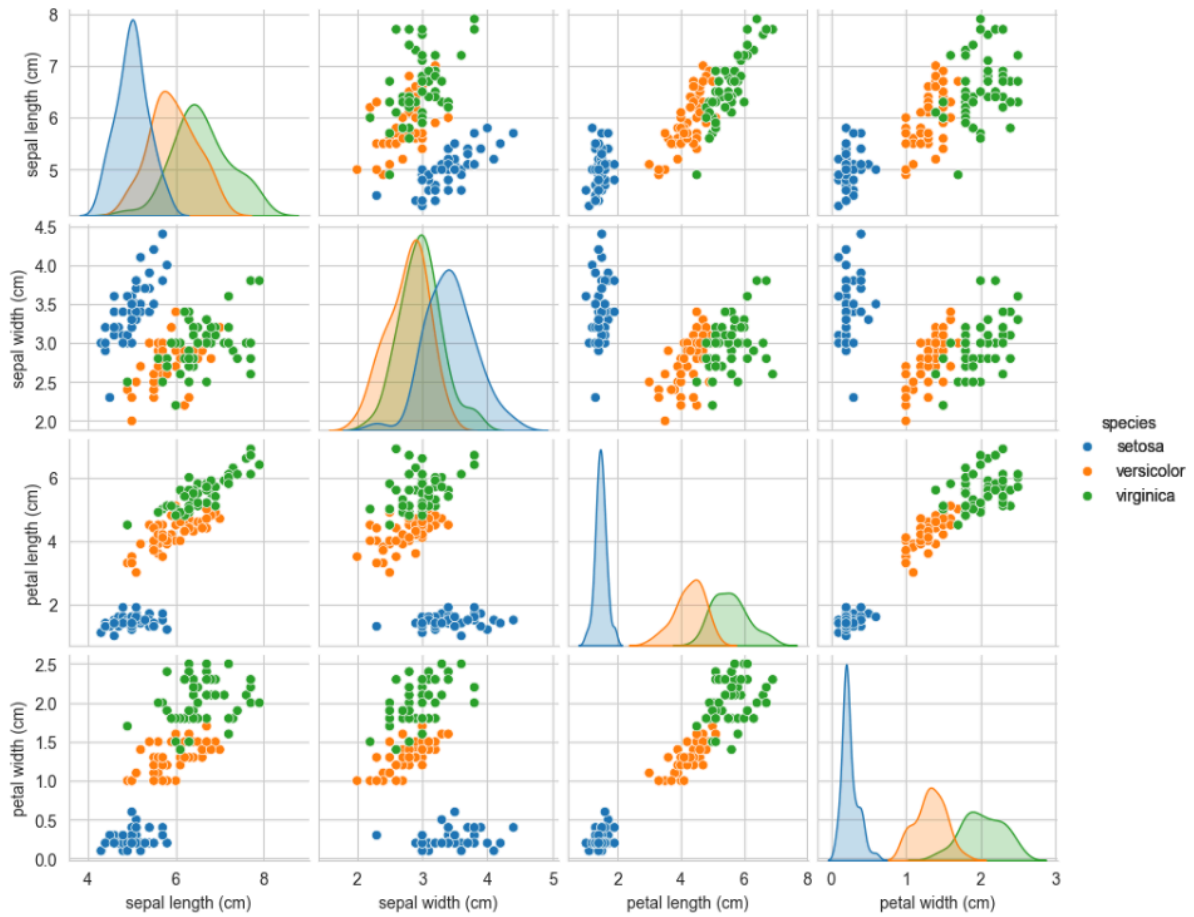
- A **scatter plot** of sepal length vs. sepal width is created using FacetGrid, highlighting the species-wise spread in two dimensions.
- A **pairplot** is generated to visualize all feature combinations across species, helping to identify which features contribute most to class separability.

These plots reveal distinct clusters, especially when petal measurements are involved, supporting the dataset's suitability for supervised classification.

```
[4]: # Visualizing data
print("\nVisualizing data: \n")
sns.set_style('whitegrid')
sns.FacetGrid(iris_dataSet, hue="species", height=6).map(mtp.scatter, 'sepal length (cm)', 'sepal width (cm)').add_legend()
sns.pairplot(iris_dataSet, hue='species', height=2).add_legend()
mtp.show()
```

```
Visualizing data:
```





3.3.5 Feature and Label Extraction

The dataset is divided into two components:

- **Independent Variables (X):** The first four columns representing sepal and petal measurements.
- **Dependent Variable (y):** The last column, which indicates the iris species.

This step prepares the data for model training by separating the input features from the target labels. Displaying the first few values of both X and y helps verify correct extraction and understand sample structure.

[5]:
```
# Extracting independent variable X and dependent variable y
X = iris_dataSet.iloc[:, :4].values
y = iris_dataSet.iloc[:, -1].values
print(f'\nDisplaying first 5 values of independent variable X: \n{X[:5]}')
print(f'\nDisplaying first 5 values of dependent variable y: \n{y[:5]}')
```

```
Displaying first 5 values of independent variable X:
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]

Displaying first 5 values of dependent variable y:
['setosa' 'setosa' 'setosa' 'setosa' 'setosa']
```

3.3.6 Train-Test Split

To evaluate the model's ability to generalize, the dataset is split into training and testing subsets using train_test_split from scikit-learn.

- **Training Set:** Used to train the model.
- **Testing Set:** Used to validate the model's performance on unseen data.

A test size of 20% is chosen, and shuffling is enabled to ensure randomness. A fixed random_state ensures reproducibility of results.

```
[6]: # Splitting the dataset into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True, random_state=0)
print(f'\nTraining set size: {X_train.shape[0]} samples \nTest set size: {X_test.shape[0]} samples')
```

```
Training set size: 120 samples
Test set size: 30 samples
```

3.3.7 Feature Scaling

Feature scaling is performed using StandardScaler to normalize the input features. This step is particularly important for distance-based algorithms like K-Nearest Neighbors (KNN), where varying feature scales can distort distance calculations. The scaler is first fitted on the training data to compute the mean and standard deviation, and then the same transformation is applied to both the training and test sets. This standardization ensures that each feature has a mean of zero and a standard deviation of one, allowing all features to contribute equally to the model's performance.

```
[7]: # Feature Scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
print(f'\nPrinting X_train: \n\n{X_train}')
```

```
Printing X_train:

[[ 0.61303014  0.10850105  0.94751783  0.736072  ]
 [-0.56776627 -0.12400121  0.38491447  0.34752959]
 [-0.80392556  1.03851009 -1.30289562 -1.33615415]
 [ 0.25879121 -0.12400121  0.60995581  0.736072  ]
 [ 0.61303014 -0.58900572  1.00377816  1.25412853]
```

3.3.8 Model Training – KNN Classifier

The K-Nearest Neighbors (KNN) classifier is instantiated with k = 3, using the Minkowski distance metric with p = 2, which corresponds to the Euclidean distance. The model is then trained on the scaled training data. By setting n_neighbors = 3, the model predicts the class of a test sample based on the majority label of its three nearest neighbors.

```
[8]: # Fitting KNN classifier to the training set
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3, metric='minkowski', p=2)
knn.fit(X_train, y_train)
```

```
[8]: KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

3.3.9 Model Prediction

After training the KNN classifier, predictions are made on the test dataset using the predict() method. This step produces the model's output classes for each instance in the test set based on the majority label of the nearest neighbors.

The predicted labels (y_pred) are then printed alongside the test features (X_test) to observe the relationship between input values and predicted outputs.

[9]:
```
# Predicting the test set results
y_pred = knn.predict(X_test)
print(f'\nPrinting X_test: \n\n{X_test}')
print(f'\nPrinting y_pred: \n\n{y_pred}')
```

```
Printing X_test:

[[-0.09544771 -0.58900572  0.72247648  1.5131568 ]
 [ 0.14071157 -1.98401928  0.10361279 -0.30004108]
 [-0.44968663  2.66602591 -1.35915595 -1.33615415]
 [ 1.6757469  -0.35650346  1.39760052  0.736072  ]
 [-1.04008484  0.80600783 -1.30289562 -1.33615415]

Printing y_pred:

['virginica' 'versicolor' 'setosa' 'virginica' 'setosa' 'virginica'
 'setosa' 'versicolor' 'versicolor' 'versicolor' 'versicolor' 'versicolor'
 'versicolor' 'versicolor' 'versicolor' 'setosa' 'versicolor' 'versicolor'
 'setosa' 'setosa' 'virginica' 'versicolor' 'setosa' 'setosa' 'virginica'
 'setosa' 'setosa' 'versicolor' 'versicolor' 'setosa']
```

3.3.10 Model Evaluation

After training the K-Nearest Neighbors (KNN) model, its performance was assessed by comparing the predicted labels against the actual test labels. The evaluation included numerical outputs and summary statistics to verify how well the model generalized to unseen data.

[10]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

ac = accuracy_score(y_test, y_pred)
print(f'\nAccuracy is: {ac:.4f}')

cm = confusion_matrix(y_test, y_pred)
print(f'\nConfusion Matrix:\n{cm}')

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy is: 0.9667

Confusion Matrix:
[[11  0  0]
 [ 0 13  0]
 [ 0  1  5]]
```

```
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        11
  versicolor       0.93      1.00      0.96        13
   virginica       1.00      0.83      0.91         6

    accuracy                           0.97        30
   macro avg       0.98      0.94      0.96        30
weighted avg       0.97      0.97      0.97        30
```

3.4 Results and Evaluation

3.4.1 Evaluation Metrics Used

1) **Accuracy:** Ratio of correctly predicted samples to the total number of samples.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where: **TP** = True Positives, **TN** = True Negatives, **FP** = False Positives, **FN** = False Negatives.

2) **Confusion Matrix:** A confusion matrix is a 3×3 table (for 3 classes) that summarizes the number of correct and incorrect predictions by class. For classes Setosa, Versicolor, and Virginica, it visually depicts:

	Predicted: Setosa	Predicted: Versicolor	Predicted: Virginica
Actual: Setosa	TP	FP	FP
Actual: Versicolor	FN	TP	FP
Actual: Virginica	FN	FN	TP

Each diagonal cell represents correctly classified instances, while off-diagonal cells represent misclassifications.

3) **Precision:** Ratio of correctly predicted positive observations to the total predicted positives.

It answers: *"Of all instances predicted as class X, how many were actually class X?"*

$$\text{Precision} = \frac{TP}{TP + FP}$$

4) **Recall (Sensitivity or True Positive Rate):** Ratio of correctly predicted positive observations to all actual observations of that class. It answers: *"Of all instances that were actually class X, how many did the model correctly predict?"*

$$\text{Recall} = \frac{TP}{TP + FN}$$

5) **F1-Score:** The harmonic mean of precision and recall. It provides a balance between them, especially useful when class distribution is uneven.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.4.2 Observed Results

The trained K-Nearest Neighbors (KNN) model exhibited robust classification performance on the Iris dataset, achieving an **accuracy of 96.67%** on the test set. **Out of 30 samples, 29 were correctly predicted**, with only a single misclassification involving a Virginica flower being labeled as Versicolor. Notably, the Setosa and Versicolor classes achieved perfect precision and recall, reflecting the algorithm's ability to distinguish between well-separated classes. The confusion matrix provides further clarity on prediction distribution across actual and predicted classes, while the classification report revealed strong performance metrics across all three categories. Precision, recall, and F1-scores were all high, with a macro-average F1-score of 0.96 and a weighted average F1-score of 0.97, indicating balanced performance even with slight class imbalance. These

results collectively indicate the model's reliability in handling multi-class classification tasks on well-separated datasets like Iris.

3.5 Challenges and Limitations

3.5.1 Challenges faced during development

1) Selecting the optimal value of *k* for the KNN algorithm.
2) Interpreting overlapping data points during visualization.
3) Managing minor inconsistencies in the dataset structure (e.g., column names or formatting in different sources).

3.5.2 Limitations of the current system

1) KNN is computationally expensive for large datasets due to runtime distance calculations.
2) The dataset is small and ideal; performance on real-world noisy or imbalanced data may differ.
3) No deployment or user interface was developed, as the focus was purely on model accuracy in a controlled setting.

3.6 Future Scope and Enhancements

1) **Algorithm Comparison:** Future iterations could compare the performance of KNN with other classification algorithms such as Support Vector Machines (SVM), Decision Trees, Random Forests, or Logistic Regression to evaluate trade-offs in accuracy, complexity, and interpretability.
2) **Model Deployment:** Integrating the trained model into a user-friendly web application or API using tools like Flask or Streamlit would make it accessible for real-time classification tasks.

CHAPTER - 04

Major Project - Customer Segmentation Using Clustering

4.1 System Analysis

4.1.1 Problem Statement

In the retail industry, understanding customer behavior is critical for designing personalized marketing strategies and optimizing product offerings. However, customers often exhibit diverse purchasing patterns that are difficult to segment manually. This project aims to solve this issue by leveraging K-Means clustering to segment customers based on key behavioral attributes such as age, annual income, and spending score. The primary goal is to help businesses understand different customer groups and enable targeted marketing strategies that improve customer satisfaction and profitability.

4.1.2 Objectives

- To analyze customer behavioral data and identify natural groupings using unsupervised learning techniques (K-Means clustering).
- To determine the optimal number of clusters using the Elbow Method.
- To evaluate and visualize customer clusters using both 2D (matplotlib, seaborn) and 3D (Plotly) plots for better interpretability.
- To derive actionable insights from the resulting clusters to assist in targeted marketing strategies.

4.1.3 Existing System

The traditional approach to customer segmentation is largely manual or heuristic. Businesses often group customers based on singular features such as age or gender without analyzing patterns in combined behaviors. This results in:

- Inaccurate or overly broad segment definitions.
- Poor targeting of marketing campaigns.
- Inefficiencies in customer relationship management.

4.1.4 Proposed System

The proposed system is a Python-based application designed to automate customer segmentation using unsupervised learning. It incorporates a structured pipeline including data ingestion, feature selection, standardization, clustering, and visual analysis. The final output includes both numerical cluster assignments and visual dashboards that provide insights into customer groupings. The system is built for interpretability and reproducibility, making it suitable for retail analytics and strategic decision-making.

4.1.5 Scope

This project focuses on behavioral segmentation using K-Means clustering applied to the Mall Customers dataset. It covers data preprocessing, exploratory analysis, and customer grouping based on age, income, and spending score. While it does not involve predictive modeling or integration with live platforms, the project demonstrates core clustering techniques and visualization strategies, providing a solid foundation for future customer analytics solutions.

4.1.6 Requirements

A. Hardware

- Processor: Intel Core i3 or above
- RAM: Minimum 4 GB
- Storage: At least 500 MB of free disk space
- Display: Any standard screen resolution supporting Jupyter Notebook or IDE

B. Software

- Operating System: Windows 10 / Linux / macOS
- Python (v3.7 or above)
- Jupyter Notebook / VS Code / Any Python IDE
- Required Libraries: pandas, numpy, matplotlib, seaborn, plotly, scikit-learn

4.2 System Design

4.2.1 Methodology and Architectural Overview

The customer segmentation pipeline consists of the following core stages:

1) **Data Acquisition:** Customer data is imported from a structured CSV file containing demographic and behavioral features such as age, annual income, and spending score.
2) **Exploratory Data Analysis (EDA):** Summary statistics and visualizations (histograms, violin plots, count plots) are used to understand feature distributions and gender-based patterns.
3) **Preprocessing:** Selected features are standardized using StandardScaler to ensure uniform scaling and eliminate feature bias during clustering.
4) **K-Means Clustering:** The algorithm is applied to discover natural groupings within the customer data. The Elbow Method is used to identify the optimal number of clusters.
5) **Evaluation and Interpretation:** Clusters are analyzed through 2D scatter plots (matplotlib, seaborn) and interactive 3D plots (Plotly) to assess separation and interpret behavior patterns.
6) **Insight Extraction:** The characteristics of each cluster are summarized to aid in business decision-making and targeted marketing.

4.2.2 Data Description

The system uses the **Mall Customers Data set**, which includes 200 customer records with the following attributes:

- **CustomerID**: Unique identifier (ignored during clustering).
- **Gender**: Male/Female (used for visual differentiation but not for clustering).
- **Age**: Customer age in years.
- **Annual Income (k$)**: Customer's annual income in thousands of dollars.
- **Spending Score (1–100)**: Score assigned by the mall based on customer behavior and spending.

For clustering, the features Age, Annual Income (k$), and Spending Score (1–100) are selected due to their relevance in behavioral segmentation.

4.2.3 Algorithm Description

The core algorithm used in this project is K-Means Clustering, an unsupervised machine learning technique designed to partition data into distinct groups based on similarity. It is particularly effective for customer segmentation because it identifies natural groupings in the data without requiring predefined labels.

K-Means Clustering: Working Principle

K-Means aims to minimize intra-cluster variance (i.e., how close the data points in a cluster are to the cluster center) and maximize inter-cluster variance (i.e., separation between different clusters). The algorithm follows these steps:

1) **Initialization**:

   Select k cluster centroids either randomly or using heuristics like **k-means++** (used in this project for better convergence).

2) **Assignment Step**:

   Assign each data point to the nearest centroid using a distance metric (Euclidean distance is used here):

$$d(x_i, c_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - c_{jk})^2}$$

3) **Update Step**:

   Recalculate centroids as the mean of all points assigned to each cluster:

$$c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

4) **Repeat**:

   Steps 2 and 3 are repeated until convergence (i.e., no change in cluster assignments or a maximum number of iterations is reached).

Elbow Method: Optimal Cluster Selection

To determine the optimal number of clusters (k), the Elbow Method is used. It involves:

- Calculating the Within-Cluster Sum of Squares (WCSS) for different values of k.
- Plotting k vs. WCSS.
- Identifying the "elbow point" where the rate of decrease sharply drops, indicating diminishing returns from adding more clusters.

Advantages of K-Means:

- Simple and computationally efficient.
- Works well with large datasets.
- Easily interpretable results.

Limitations:

- Sensitive to the initial selection of centroids.
- Struggles with non-spherical clusters and varying densities.
- Requires prior knowledge of the number of clusters (k).

4.2.4 Tools and Technologies Used

1) **Programming Language**: Python 3.10
2) **Libraries**:
    a) pandas, numpy – Data manipulation and numerical operations
    b) matplotlib, seaborn – Static visualizations
    c) scikit-learn – Clustering (KMeans), preprocessing
    d) plotly – Interactive 3D visualizations
3) **Development Environment**: Jupyter Notebook

4.2.5 Summary of Design Decisions

In this project, K-Means clustering was chosen due to its simplicity, scalability, and ease of interpretation for customer segmentation tasks. To ensure fair contribution from all features, standardization was applied using the StandardScaler, which mitigates the influence of differing feature magnitudes. The optimal number of clusters was identified as four using the Elbow Method, which helped balance within-cluster compactness and model

generalizability. To further enhance the interpretability of the resulting clusters, an interactive 3D visualization was implemented using Plotly, offering a comprehensive view of customer distribution across age, income, and spending behavior.

4.3 Implementation

4.3.1 Importing Dataset

The required libraries such as pandas, numpy, seaborn, matplotlib, scikit-learn, plotly, and others are imported at the beginning of the implementation. These libraries support data manipulation, visualization, clustering, and rendering of interactive plots. The dataset is then loaded using pandas.read_csv() from a local path. It consists of customer demographic and behavioral attributes such as Gender, Age, Annual Income, and Spending Score. A preview of the first few records is printed to ensure the data has been successfully imported and is in the expected format.

[1]:
```
# Import libraries
import plotly.io as pio
pio.renderers.default = 'notebook'
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import plotly.express as px
import warnings
warnings.filterwarnings('ignore')

# Load customer data
df = pd.read_csv('C:\\Users\\banda\\Downloads\\Mall_Customers.csv')
print(df.head())
```

```
   CustomerID   Genre  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
```

4.3.2 Data Inspection and Feature Selection

Before analysis, the dataset is examined to ensure its structure is suitable for clustering. The column 'Genre' is renamed to 'Gender' for clarity and consistency. The shape, data types, summary statistics, and null value counts are displayed to validate the dataset's integrity. No missing values were found, confirming it is clean and ready for processing. For clustering, three relevant numerical features are selected: Age, Annual Income (k$), and Spending Score (1–100). These features are chosen as they capture key behavioral and demographic traits necessary for effective customer segmentation.

[2]:
```
# Rename column if necessary
df.rename(columns={'Genre': 'Gender'}, inplace=True)

# Display dataset info
print("Dataset Shape:", df.shape)
print("\nData Types:\n", df.dtypes)
print("\nSummary Statistics:\n", df.describe())
print("\nMissing Values:\n", df.isnull().sum())

# Select features for clustering
features = ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']
X = df[features]
```

```
Dataset Shape: (200, 5)

Data Types:
 CustomerID                int64
Gender                    object
Age                        int64
Annual Income (k$)         int64
Spending Score (1-100)     int64
dtype: object

Summary Statistics:
        CustomerID         Age  Annual Income (k$)  Spending Score (1-100)
count  200.000000  200.000000          200.000000              200.000000
mean   100.500000   38.850000           60.560000               50.200000
std     57.879185   13.969007           26.264721               25.823522
min      1.000000   18.000000           15.000000                1.000000
25%     50.750000   28.750000           41.500000               34.750000
50%    100.500000   36.000000           61.500000               50.000000
75%    150.250000   49.000000           78.000000               73.000000
max    200.000000   70.000000          137.000000               99.000000
```

```
Missing Values:
 CustomerID                0
Gender                     0
Age                        0
Annual Income (k$)         0
Spending Score (1-100)     0
dtype: int64
```

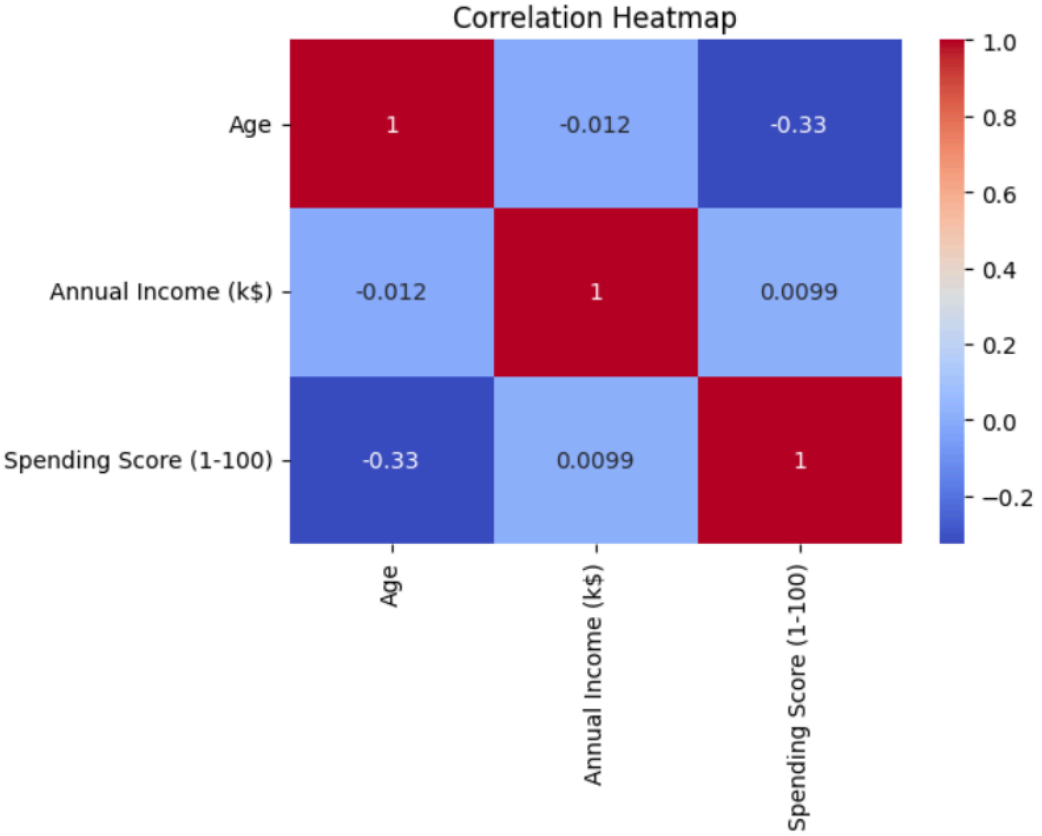
4.3.3 Exploratory Data Analysis (EDA)

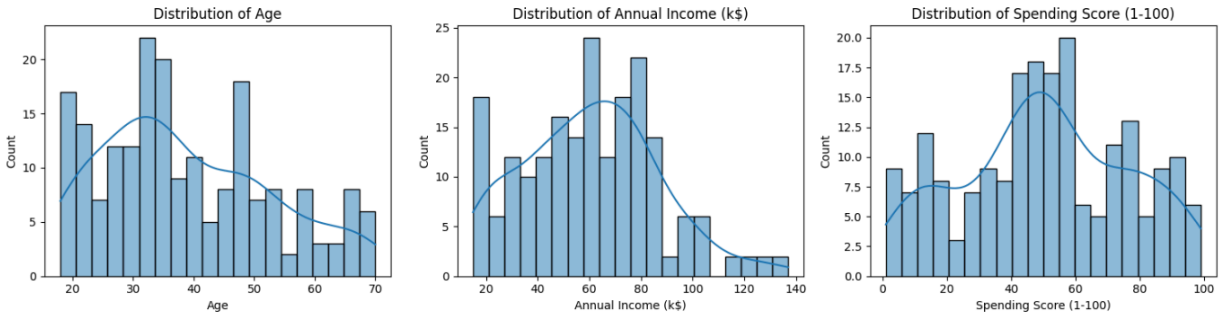
To understand the underlying patterns in the dataset, several visualizations are performed:

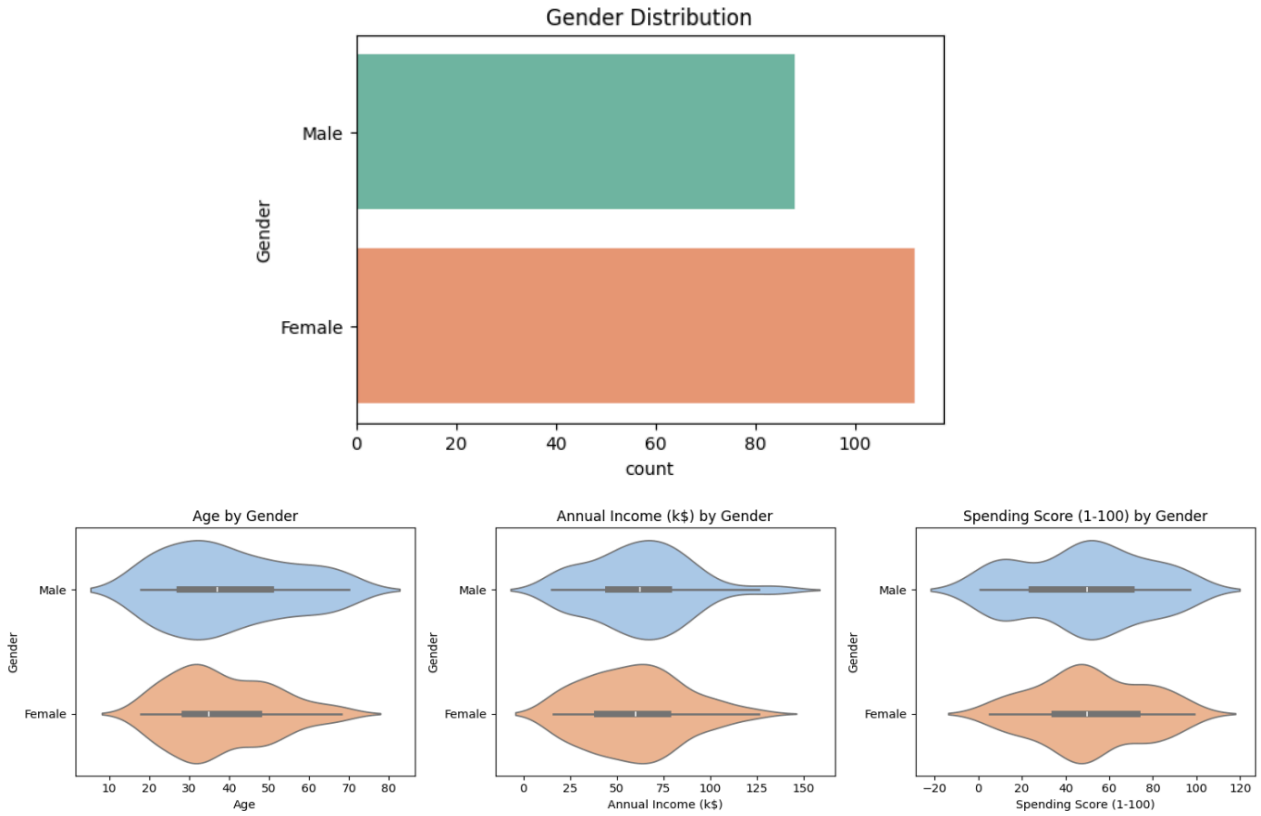
- **Correlation Heatmap**: A heatmap is generated to examine linear relationships between the selected numerical features. The strongest correlation is typically observed between **Annual Income** and **Spending Score**, offering insight into consumer behavior.
- **Distribution Plots**: Histograms with KDE curves display the distribution of **Age**, **Annual Income**, and **Spending Score**. These plots help identify skewness, outliers, and the overall spread of data.
- **Gender Distribution**: A count plot reveals the gender composition of the dataset, assisting in assessing demographic balance.
- **Violin Plots by Gender**: To explore gender-specific behavior, violin plots are used to compare the distributions of each selected feature across **male** and **female** groups. This helps in identifying if spending habits or income levels differ by gender.

[3]:
```python
# Correlation heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(X.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# Distribution plots
plt.figure(figsize=(15, 4))
for idx, col in enumerate(features):
    plt.subplot(1, 3, idx+1)
    sns.histplot(df[col], bins=20, kde=True)
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()
```

```python
# Gender distribution
plt.figure(figsize=(6, 4))
sns.countplot(data=df, y='Gender', palette='Set2')
plt.title('Gender Distribution')
plt.show()

# Violin plots by gender
plt.figure(figsize=(15, 4))
for idx, col in enumerate(features):
    plt.subplot(1, 3, idx+1)
    sns.violinplot(data=df, x=col, y='Gender', palette='pastel')
    plt.title(f'{col} by Gender')
plt.tight_layout()
```







4.3.4 Feature Standardization

Standardization transforms the data into a common scale with a mean of 0 and standard deviation of 1, thereby preventing features with larger magnitudes from disproportionately influencing the clustering process.

```
[4]: # Standardize selected features
     scaler = StandardScaler()
     X_scaled = scaler.fit_transform(X)
```

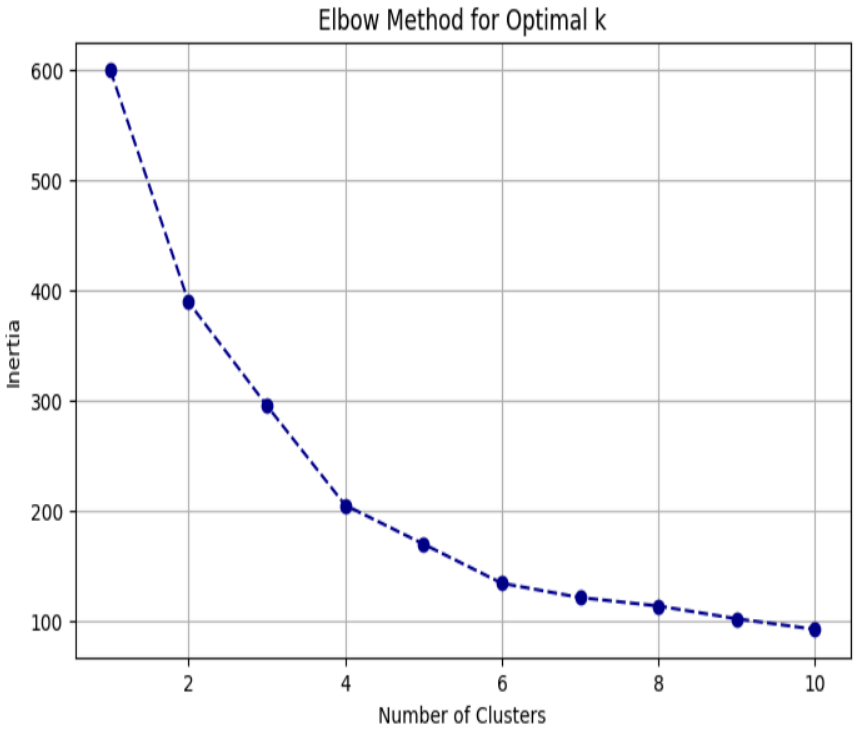
4.3.5 Determining Optimal Number of Clusters: Elbow Method

To determine the most appropriate number of clusters (k) for customer segmentation, the **Elbow Method** is employed. This technique involves fitting the K-Means algorithm for a range of values of *k* (from 1 to 10), and recording the **inertia** (sum of squared distances from each point to its assigned cluster center) for each value.

The inertia values are then plotted against *k*. As the number of clusters increases, inertia decreases. The "elbow point" on the graph — where the rate of decrease sharply slows — indicates the optimal number of clusters. This is the point where adding more clusters yields diminishing returns in terms of intra-cluster compactness.

In this project, the elbow is observed around **k = 4**, suggesting that four clusters best capture the underlying structure of the customer data.

[5]:
```
# Elbow Method
inertia = []
for k in range(1, 11):
    km = KMeans(n_clusters=k, init='k-means++', random_state=42)
    km.fit(X_scaled)
    inertia.append(km.inertia_)

plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), inertia, marker='o', linestyle='--', color='darkblue')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal k')
plt.grid(True)
plt.show()
```



4.3.6 K-Means Clustering and Cluster Centers

After identifying **k = 4** as the optimal number of clusters using the Elbow Method, the K-Means algorithm is applied to the standardized dataset. The algorithm partitions customers into four distinct clusters based on their similarities in age, annual income, and spending score.

Once clustering is complete, each customer in the dataset is assigned a **cluster label**, which is stored in a new column named 'Cluster'. This label indicates the segment to which the customer belongs.

To interpret the clusters meaningfully, the **cluster centers**—originally computed in the standardized feature space—are **inverse transformed** back to the original scale using the fitted StandardScaler. These centers represent the average profile of customers within each cluster and serve as key reference points for understanding the characteristics of each group.

[6]:
```
# Apply K-Means (k=4 as optimal from Elbow Method)
k = 4
kmeans = KMeans(n_clusters=k, init='k-means++', random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)

# Inverse transform cluster centers
cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)
print('\nCluster Centers (Original Scale):')
for i, center in enumerate(cluster_centers):
    print(f'Cluster {i+1}: {center.round(2)}')
```

```
Cluster Centers (Original Scale):
Cluster 1: [53.98 47.71 39.97]
Cluster 2: [32.88 86.1  81.53]
Cluster 3: [25.44 40.   60.3 ]
Cluster 4: [39.37 86.5  19.58]
```

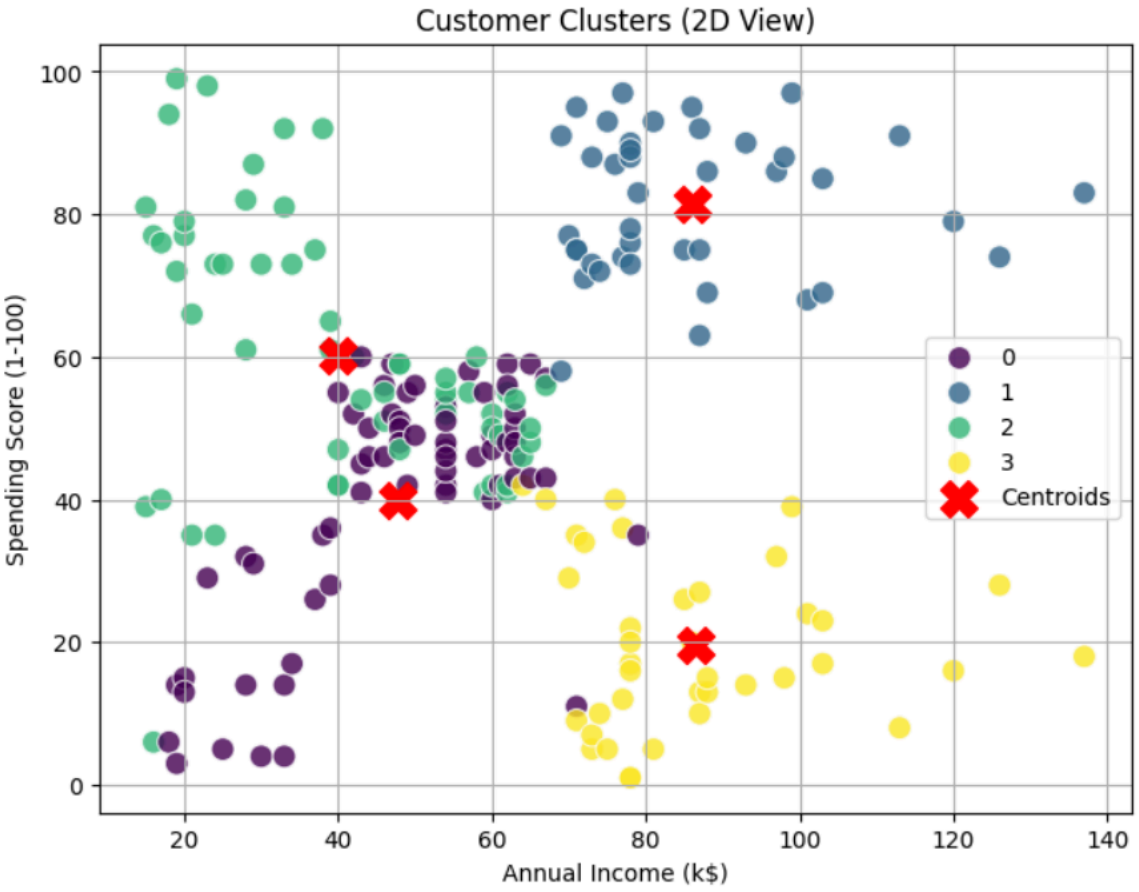
4.3.7 Cluster Visualization in 2D and 3D

To interpret the clustering results visually, both 2D and 3D plots are used to display the distribution of customers across the identified clusters. In the **2D scatter plot**, customers are plotted based on their **annual income** and **spending score**, with distinct colors representing different

clusters. Red 'X' markers denote the **cluster centroids**, highlighting the central tendency of each group. This visualization provides an intuitive understanding of how customer segments differ based on purchasing behavior.

To further enhance interpretability, a **3D scatter plot** is generated using Plotly, incorporating **age** as an additional dimension. Each data point represents a customer and is color-coded by cluster and symbol-coded by gender. This interactive 3D view allows for more nuanced insights into how customer attributes—age, income, and spending—jointly influence segmentation, enabling a deeper exploration of group boundaries and overlaps.

[7]:
```
# 2D Cluster Visualization
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Annual Income (k$)', y='Spending Score (1-100)',
                hue='Cluster', palette='viridis', data=df, s=100, alpha=0.8)
plt.scatter(cluster_centers[:, 1], cluster_centers[:, 2], s=250, c='red', marker='X', label='Centroids')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Customer Clusters (2D View)')
plt.legend()
plt.grid(True)
plt.show()

# 3D Plot using Plotly
fig = px.scatter_3d(df, x='Age', y='Annual Income (k$)', z='Spending Score (1-100)',
                    color=df['Cluster'].astype(str),
                    title='Customer Segmentation (3D View)',
                    symbol='Gender', opacity=0.8,
                    color_discrete_sequence=px.colors.qualitative.Vivid)
fig.update_layout(legend_title_text='Cluster', scene=dict(
    xaxis_title='Age',
    yaxis_title='Income (k$)',
    zaxis_title='Score'
))
fig.show()
```



Customer Segmentation (3D View)

Cluster
2, Male
2, Female
0, Male
0, Female
3, Male
3, Female
1, Male
1, Female

4.4 Results and Evaluation

The K-Means clustering algorithm was applied to a dataset of 200 mall customers, each described by age, annual income, and spending score. The model successfully segmented the customers into four well-defined clusters. The effectiveness of the segmentation was assessed both quantitatively and visually.

4.4.1 Dataset Insights

1) The dataset consisted of **200 entries** and **5 attributes**, with **no missing values**.
2) Key numerical attributes had the following characteristics:
    a) **Age** ranged from 18 to 70 years (**mean ≈ 38.85**)
    b) **Annual Income** ranged from 15k to 137k (**mean ≈ 60.56k**)
    c) **Spending Score** ranged from 1 to 99 (**mean ≈ 50.20**)

This diversity in age, income, and spending behavior justified the need for segmentation.

4.4.2 Elbow Method and Cluster Selection

A plot of the Within-Cluster Sum of Squares (WCSS) showed a distinct elbow at **k = 4**, indicating that four clusters were optimal for this dataset. The WCSS, also known as **inertia**, is calculated as:

$$\text{Inertia} = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Where:

- C_i is the i-th cluster
- μ_i is the centroid of C_i
- $\|x - \mu_i\|$ is the Euclidean distance between a point and its cluster centroid

4.4.3 Cluster Interpretation

The final cluster centroids (in original scale) revealed distinct customer segments:

Cluster	Age	Annual Income (k$)	Spending Score (1–100)	Interpretation
1	53.98	47.71	39.97	Older customers with average spending
2	32.88	86.10	81.53	Young professionals, high spenders
3	25.44	40.00	60.30	Youthful shoppers with moderate income
4	39.37	86.50	19.58	High-income but low-spending customers

These clusters help businesses tailor marketing strategies, such as offering loyalty programs to high-spenders or personalized promotions to low-engagement segments.

4.5 Challenges and Limitations

Despite its effectiveness, the K-Means clustering approach presents several inherent challenges. Firstly, the requirement to specify the number of clusters (k) a priori introduces a degree of subjectivity; although the Elbow Method was used, it still involves interpretive judgment and may not always reflect the true structure of the data. Additionally, K-Means assumes that clusters are convex, isotropic, and of similar density—an assumption that may not hold in complex customer behavior patterns where overlaps and irregular shapes exist.

The algorithm is also sensitive to feature scaling and initialization. While standardization was applied to mitigate feature bias, different centroid

seeds can lead to different clustering results, affecting reproducibility. Furthermore, categorical variables such as gender were not incorporated directly into clustering due to K-Means' reliance on Euclidean distance, which limits its ability to fully capture mixed-type data.

Lastly, while 2D and 3D visualizations improved interpretability, these projections may not fully capture the intricacies of customer segmentation in higher-dimensional space, especially when latent behavioral factors are not explicitly represented in the dataset.

4.6 Future Scope and Enhancements

The current model provides meaningful customer segmentation using K-Means clustering. However, the following key improvements can enhance its practical utility:

- **Incorporating Categorical Features:** Future iterations can include variables like gender or membership type by using algorithms that support mixed data types (e.g., K-Prototypes).
- **Advanced Cluster Validation:** In addition to the Elbow Method, metrics like the Silhouette Score or Davies–Bouldin Index can be adopted to validate the optimal number of clusters more effectively.
- **Real-Time Segmentation:** Deploying the model in a real-time environment would allow businesses to dynamically update customer segments based on recent behavior.
- **Integration with Business KPIs:** Mapping clusters to strategic indicators such as customer lifetime value or churn risk can make segmentation more actionable for marketing teams.

CHAPTER - 05

CONCLUSION

This internship experience provided a comprehensive foundation in practical machine learning through two structured projects—each designed to explore distinct paradigms: **supervised learning** and **unsupervised learning**.

In the **mini project**, the K-Nearest Neighbors (KNN) algorithm was implemented for classification on the Iris dataset. The model demonstrated high predictive accuracy and excellent class-wise performance. The results reflected the effectiveness of KNN in handling multi-class classification tasks when working with well-structured and separable datasets.

The **major project** focused on applying unsupervised learning for customer segmentation using K-Means clustering. Through exploratory data analysis, standardization, and optimal cluster selection, meaningful customer groups were discovered based on age, income, and spending score. The clusters offered actionable insights into customer behavior, ranging from high-spending young professionals to low-spending but high-income individuals. Interactive 3D visualization further enhanced interpretability, making the segmentation results more intuitive and business-friendly.

Together, these projects helped bridge theoretical knowledge with hands-on implementation. Core skills such as data preprocessing, visualization, model evaluation, and clustering interpretation were effectively applied. More importantly, the projects demonstrated how machine learning models can be tailored to address both classification problems and behavioral pattern discovery—making them directly applicable in business intelligence and decision-making contexts.

REFERENCES

[1] T. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[2] E. Fix and J. L. Hodges Jr., “Discriminatory analysis: Nonparametric discrimination: Consistency properties,” *USA Air Force School of Aviation Medicine*, Technical Report 4, 1951.

[3] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[4] R. A. Fisher, “The use of multiple measurements in taxonomic problems,” *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[5] J. B. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proc. 5th Berkeley Symp. Mathematical Statistics and Probability*, vol. 1, pp. 281–297, 1967.

[6] R. Tibshirani, G. Walther, and T. Hastie, “Estimating the number of clusters in a data set via the Gap statistic,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[7] P. J. Rousseeuw, “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[8] C. R. Harris et al., “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020.

[9] W. McKinney, “Data structures for statistical computing in Python,” in *Proc. 9th Python in Science Conf. (SciPy)*, 2010, pp. 51–56.

[10] J. D. Hunter, “Matplotlib: A 2D graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[11] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] D. S. S. G. Bandaru, *Codebase for HDLC Technologies Internship Projects*, Available: https://github.com/deva-04/HDLC-Technologies-Internship