AN INTERNSHIP REPORT

on

Data Science and Machine Learning

Submitted in partial fulfilment of 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

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA

KAKINADA-533003, ANDHRA PRADESH, INDIA

2021-2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING KAKINADA

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA

KAKINADA-533003, ANDHRA PRADESH, INDIA

CERTIFICATE

This is to certify that this project report entitled "**DATA SCIENCE AND MACHINE LEARNING**" is a bonafide record of the work being 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. Rama Krishnaiah

Professor & HOD

Department of CSE

UCEK(A)

JNTU KAKINADA

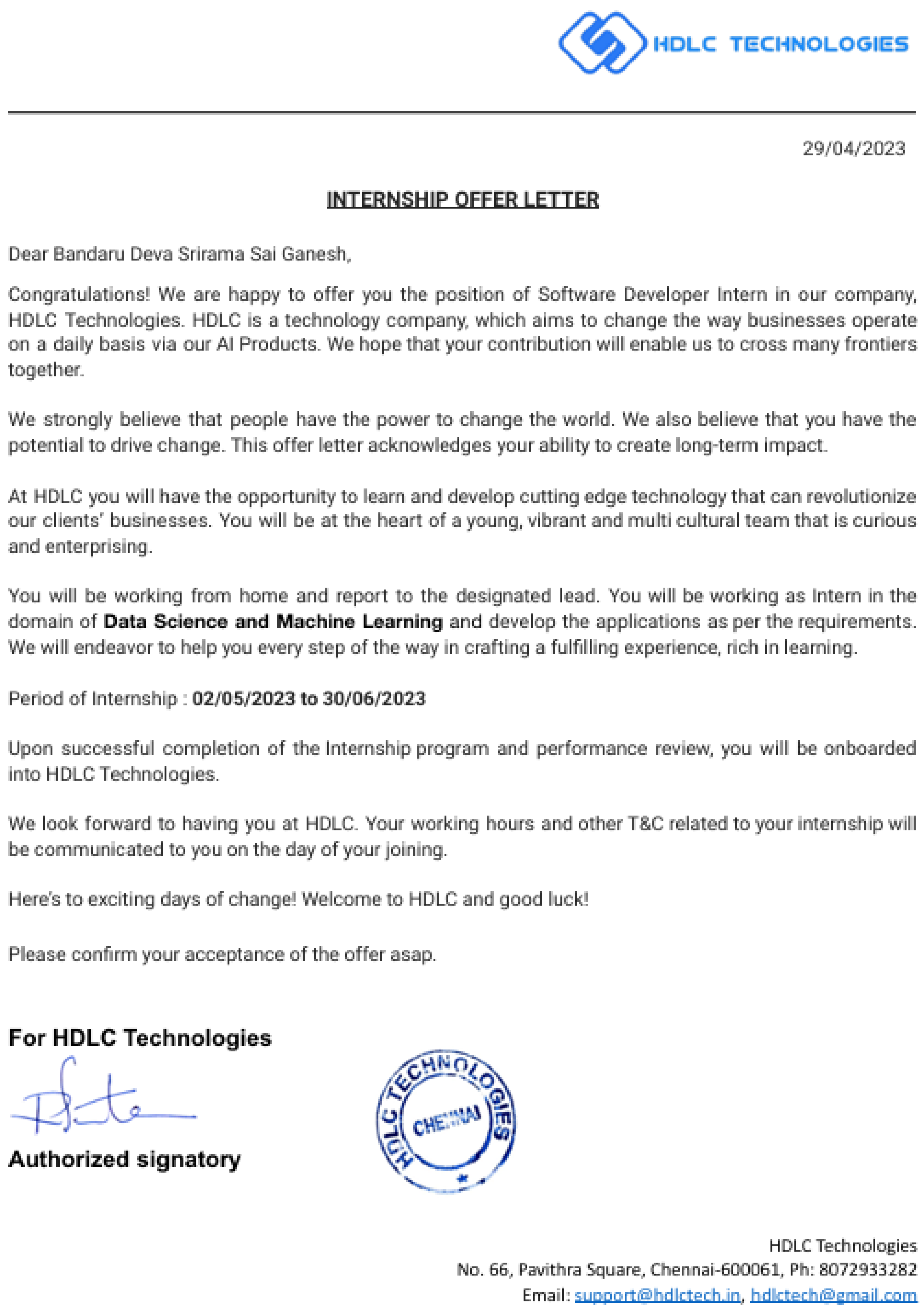
ACKNOWLEDGEMENT

It is with immense pleasure that I take this opportunity to express my heartfelt gratitude to everyone who supported me in successfully completing my internship at HDLC Technologies. I extend my sincere thanks to Dr. Ramakrishnaiah, Professor and Head of the Department of Computer Science and Engineering, for his guidance and encouragement. I am also deeply grateful to our Principal, Dr. Mohan Rao, University College of Engineering Kakinada, for his constant support and motivation throughout this internship.

I would like to extend my profound gratitude to Dr. Ravindranath and Dr. Deekshithulu for their valuable encouragement and support during this journey. My heartfelt appreciation also goes to my family, friends, and companions, whose unwavering support was instrumental in the completion of this report.

This internship has provided me with invaluable practical exposure and an opportunity to bridge the gap between theoretical knowledge and real-world applications. It is a privilege and an honor to present this report, and I hope it serves as a meaningful contribution to bridging the divide between academic concepts and practical implementation.

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

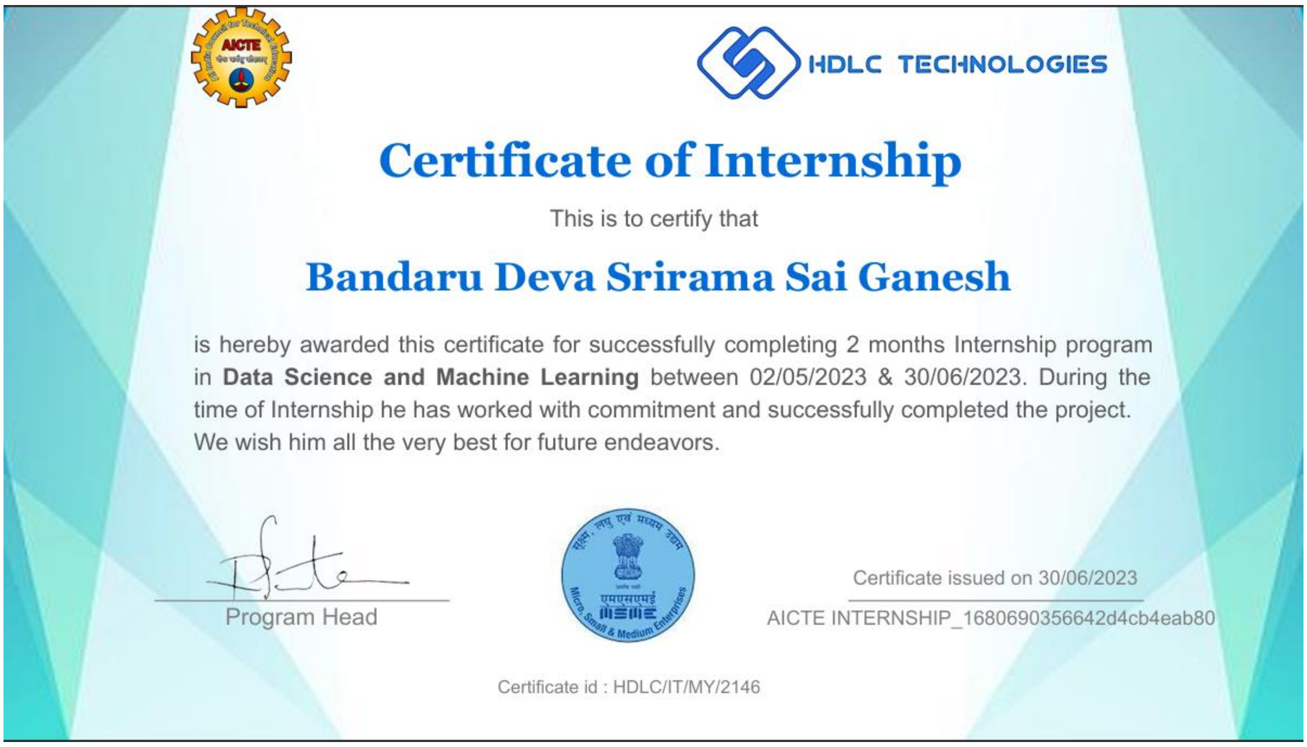
[illegible]

Authorized signatory

HDLC TECHNOLOGIES CHENNAI

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

INTERNSHIP CERTIFICATE



HDLC TECHNOLOGIES

Certificate of Internship

This is to certify that

Bandaru Deva Srirama Sai Ganesh

is hereby awarded this certificate for successfully completing 2 months Internship program in **Data Science and Machine Learning** between 02/05/2023 & 30/06/2023. During the time of Internship he has worked with commitment and successfully completed the project. We wish him all the very best for future endeavors.

[illegible]

Program Head

Certificate issued on 30/06/2023

AICTE INTERNSHIP_1680690356642d4cb4eab80

Certificate id : HDLC/IT/MY/2146

ABSTRACT

This internship at HDLC Technologies emphasized applying data science and machine learning techniques to develop innovative solutions for real-world problems. The mini-project focused on **classifying the Iris dataset** using algorithms such as K-Nearest Neighbors (KNN), demonstrating foundational concepts in supervised learning and model evaluation.

The major project centered on **customer segmentation**, leveraging clustering algorithms like K-Means to group customers based on shared characteristics, providing actionable insights for targeted business strategies.

The development process involved data preprocessing, algorithm selection, model training, evaluation, and deployment. Python libraries, including NumPy, pandas, matplotlib, and scikit-learn, were instrumental in data manipulation, visualization, and implementing machine learning models.

This internship provided practical experience in leveraging machine learning to address business challenges and reinforced the value of data-driven decision-making. These projects not only enhanced technical skills but also demonstrated the ability to deliver impactful solutions tailored to organizational needs.

Index Terms: Data Science, Machine Learning, Customer Segmentation, K-Means, Iris Dataset, KNN, Python, NumPy, pandas, scikit-learn, Data Analysis, Model Evaluation.

Table of Contents

CHAPTER I

INTRODUCTION

INTRODUCTION

HDLC Technologies is a forward-thinking organization specializing in data science and machine learning solutions. The company provides a range of services designed to help businesses leverage data to optimize operations, gain insights, and implement data-driven strategies. During my two-month virtual internship at HDLC Technologies, I was tasked with working on projects that directly applied machine learning techniques to real-world problems. This internship provided valuable exposure to the industry's applications of machine learning, and I gained hands-on experience working with tools and technologies in data science.

The primary goal of this internship was to develop practical skills in data science and machine learning by working on live projects. I sought to apply the theoretical knowledge gained during my academic studies to real-world scenarios, such as classification and clustering tasks. The internship aimed to deepen my understanding of the entire workflow in machine learning projects, from initial data exploration and cleaning to the deployment of predictive models.

The internship was divided into two key projects that allowed me to explore both supervised and unsupervised learning techniques:

1. Mini Project - Classification of the Iris Dataset

This project involved applying supervised learning to classify the Iris dataset, providing insight into classification algorithms and the concept of model evaluation. I gained a better understanding of algorithm selection, training models, and evaluating their performance using various metrics such as accuracy, precision, and recall.

2. Major Project - Customer Segmentation

The second project focused on unsupervised learning, specifically K-Means clustering, to segment customers based on purchasing behaviour. This project aimed to assist businesses in deriving actionable insights for targeted marketing and personalized customer engagement. By analyzing and clustering customer data, I gained valuable insights into how machine learning can be used to uncover patterns and guide strategic decision-making.

This report outlines the details of these projects, discussing the methodologies, tools, and techniques employed. It also reflects on the challenges faced, the solutions implemented, and the key learning outcomes from the internship.

1.1 **Methodology Overview**

The internship work was divided into two key projects, each following a structured approach involving requirement gathering, design, implementation, testing, and final deployment. The projects focused on data preprocessing, model training, and evaluation, with continuous iterations for refining machine learning models based on performance metrics and feedback.

CHAPTER II

LITERATURE SURVEY

LITERATURE SURVEY

To build a strong foundation for the machine learning models developed during this internship, previous research on classification and clustering techniques was reviewed. This provided insights into the strengths, limitations, and real-world applications of algorithms like K-Nearest Neighbors (KNN) and K-Means clustering.

K-Nearest Neighbors (KNN) is a simple and effective classification algorithm widely used in various domains, including image recognition and medical diagnostics. According to **Hastie, Tibshirani, and Friedman (2009)**, KNN performs well on small to medium-sized datasets but may struggle with high-dimensional data due to the 'curse of dimensionality'.

K-Means clustering is a popular algorithm for segmenting data into distinct groups based on feature similarity. **Jain (2010)** noted that K-Means is efficient for large datasets but can struggle with non-globular clusters. Variants like K-Means++ have been developed to address some of these limitations by improving cluster initialization.

The insights gained from these studies guided the selection of KNN for the Iris dataset classification, ensuring that the dataset's small size and clear feature separability made KNN an appropriate choice. Similarly, understanding the limitations of K-Means clustering led to careful data preprocessing to minimize issues related to cluster initialization and outlier influence.

CHAPTER III

DATA SCIENCE AND MACHINE LEARNING OVERVIEW

OVERVIEW OF MACHINE LEARNING AND DATA SCIENCE

Machine Learning is a subset of artificial intelligence (AI) that enables computers to learn from data and make decisions without being explicitly programmed. It focuses on building algorithms that allow systems to automatically identify patterns in data, learn from them, and make predictions or decisions based on new, unseen data. Machine learning can be classified into three main types:

- **Supervised Learning**: In this approach, the model is trained on labeled data, where the correct answers (outputs) are already known. The algorithm learns to map inputs to the correct output by finding patterns in the data. Examples include classification tasks like identifying flowers in the Iris dataset (using KNN).
- **Unsupervised Learning**: This approach involves training models on unlabeled data to find hidden patterns or groupings in the data. One of the most common techniques used in unsupervised learning is clustering, where the algorithm groups data points into clusters based on their similarities. The customer segmentation project using K-Means clustering is an example of this approach.
- **Reinforcement Learning**: In reinforcement learning, an agent learns to make decisions by interacting with an environment and receiving feedback through rewards or penalties. This type of learning is often used in robotics, gaming, and autonomous systems.

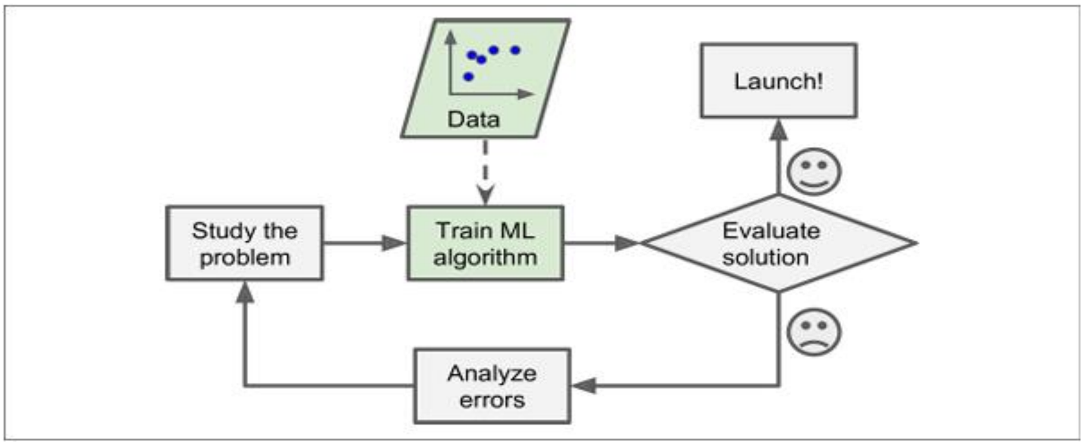
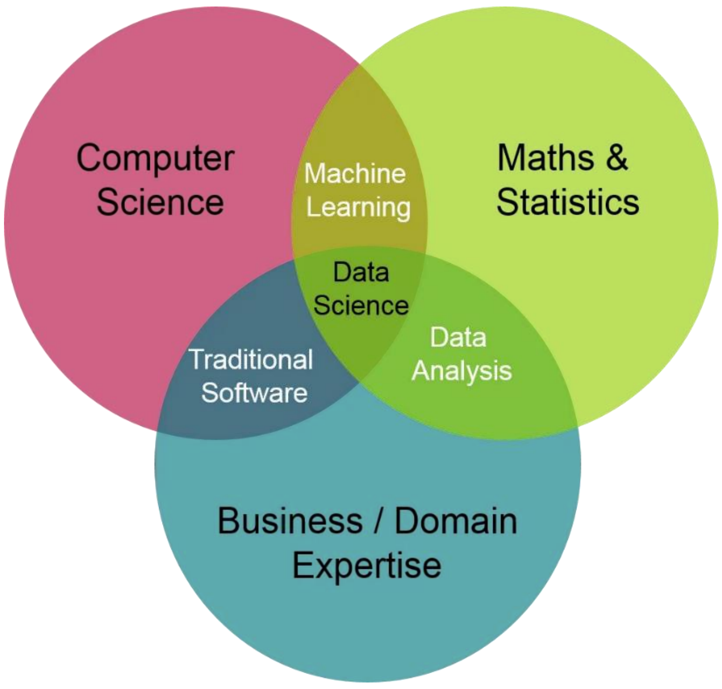


Figure: Machine Learning approach

Data Science is an interdisciplinary field that combines techniques from statistics, mathematics, computer science, and domain expertise to analyze and interpret large volumes of data. It involves data collection, data cleaning, exploratory data analysis (EDA), model building, and the application of machine learning algorithms to solve real-world problems. Data science is essential for deriving actionable insights from both structured (e.g., databases) and unstructured data (e.g., text, images).

The synergy between machine learning and data science has revolutionized industries such as healthcare, finance, marketing, and technology. The ability to process and analyze vast amounts of data has paved the way for predictive analytics, automated decision-making, and personalized services.



CHAPTER IV

PROJECTS

4.1 MINOR PROJECT: IRIS DATASET CLASSIFICATION

4.1.1 Project Objective

The aim of this mini project is to classify the Iris dataset using the K-Nearest Neighbors (KNN) algorithm. By applying KNN, the goal is to predict the species of Iris flowers based on their sepal and petal attributes. The project focuses on understanding the relationship between flower attributes and species, and it demonstrates how KNN can be used for classification tasks in machine learning.

4.1.2 Existing Approach

The existing method of classifying the Iris dataset may involve:

- Basic rule-based or decision tree classifiers with limited accuracy.
- Traditional methods that may struggle with higher dimensionality or large datasets.
- Lack of flexibility in accommodating new and evolving data points efficiently.

4.1.3 Proposed Solution

The proposed system utilizes the K-Nearest Neighbours (KNN) algorithm to classify the Iris dataset:

- The KNN algorithm is a simple yet powerful classification method that works effectively for small datasets like Iris.
- The system processes the data by standardizing it and splitting it into training and test datasets.
- The use of KNN ensures that the classification is based on similarity metrics, making it an intuitive choice for flower species prediction.
- The accuracy and effectiveness of KNN are evaluated using metrics like accuracy and confusion matrix.

4.1.4 Requirements

a) Hardware

- Processor: **Intel i5 or higher** (Recommended for fast processing of large datasets)
- RAM**: 8GB** (Recommended for handling data efficiently during model training)
- Storage**: 1TB Hard Disk** (Recommended for storing datasets and model outputs)
- Display**: Standard resolution display** for visualizing data and model results

b) Software Requirements

- Operating System: **Windows 10/11** (or other Linux-based OS)
- **Python:** A programming language used for machine learning implementation.
- Libraries: pandas, matplotlib, scikit-learn
- **Jupyter Notebook** or similar (For running the Python code and visualizing results)

4.1.5 Methodology

Algorithm Used

K-Nearest Neighbors (KNN) is a simple, supervised machine learning algorithm used for classification tasks. It is based on the principle that similar data points tend to be near each other in feature space.

Why KNN?

KNN is a non-parametric method that does not make assumptions about the underlying data distribution, making it a versatile choice for classification problems. It is particularly effective when the data has clear separability between classes, as seen in the Iris dataset. The algorithm is easy to implement and understand, and works well with small to medium-sized datasets.

Working of KNN

The KNN algorithm operates by finding the k nearest neighbors to a given data point in feature space, and classifying the data point based on the majority class among the neighbors. The distance between data points is typically measured using the Euclidean distance formula. The process can be broken down into the following steps:

1. **Training Phase**: The algorithm stores the entire dataset.
2. **Prediction Phase**: For a given test data point, the algorithm calculates the distance between the test point and all training data points. It selects the k nearest neighbours and assigns the most common class among them as the predicted class for the test data point.

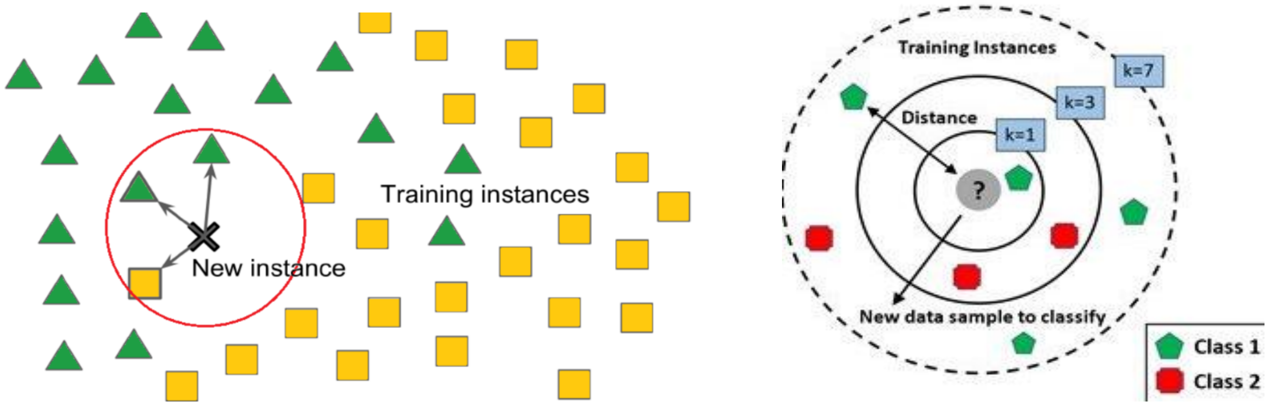


Figure: KNN Working

Advantages of KNN:

1. **Simple to Understand and Implement**:

   KNN is one of the simplest machine learning algorithms to understand and implement. There is no need for a complex training phase; it simply stores the entire training dataset and makes predictions based on nearest neighbours.

2. **No Assumptions About Data**:

   Unlike some algorithms (e.g., linear regression), KNN makes no assumptions about the underlying data distribution, which can be useful when the data is not normally distributed.

3. **Versatile**:

   KNN can be used for both classification and regression tasks, making it a versatile algorithm for various machine learning problems.

4. **Adaptable to Any Data Type**:

   KNN can work with both numeric and categorical data, as long as an appropriate distance metric (like Euclidean or Manhattan) is used.

5. **Efficient for Small Datasets**:

   For smaller datasets, KNN can work efficiently, as it performs predictions quickly without needing extensive training time.

6. **No Training Phase**:

   KNN is a **lazy learner**, meaning it does not involve a training phase and instead relies on the entire dataset for making predictions, which can be a benefit for certain use cases.

Disadvantages of KNN:

1. **Computationally Expensive**:

   KNN requires calculating distances for every data point in the dataset, which can be slow for large datasets.

2. **Struggles with High-Dimensional Data:**

   KNN's performance decreases with an increase in the number of features (curse of dimensionality).

3. **Choice of K and Distance Metric**:

   The algorithm's effectiveness depends on selecting the right value of **K** and the appropriate distance metric.

CLASSIFICATION

Classification is a supervised learning technique used to predict the categorical labels of new data points based on previously seen examples. In classification, the goal is to assign input data into predefined categories or classes. This type of machine learning task involves using labeled datasets, where the outcome (label) is known for each example.

Types of Classification Problems:

- **Binary Classification**: This involves classifying data into two classes, such as spam vs. non-spam emails or fraudulent vs. legitimate transactions.
- **Multiclass Classification**: Here, the data is classified into more than two categories, such as classifying images of animals into dogs, cats, and birds.
- **Multilabel Classification**: This involves assigning multiple labels to an instance, as in tagging a document with multiple topics.

Evaluation Metrics

1. Accuracy

Accuracy is the most straightforward evaluation metric. It is the proportion of correctly classified instances over the total number of instances in the test set.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

While accuracy is easy to understand, it can be misleading in imbalanced datasets, where one class significantly outnumbers the other.

2. Precision

Precision measures how many of the instances that were classified as a particular class are actually that class. It is particularly important when the cost of false positives is high.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

High precision means that when the model predicts a positive class, it is most likely correct.

3. Recall (Sensitivity)

Recall indicates how many of the actual instances of a class were correctly identified by the model. It is particularly useful when the cost of false negatives is high, i.e., when it's crucial not to miss a positive case.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

A high recall means the model does a good job identifying positive instances but may include more false positives.

4. F1-Score

F1-Score is the harmonic mean of precision and recall. It balances the trade-off between precision and recall, especially when dealing with imbalanced datasets where high precision and recall are both necessary.

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

It is often preferred over accuracy when dealing with class imbalances.

5. Confusion Matrix

A confusion matrix is a table that describes the performance of a classification model by showing the counts of true positives, false positives, true negatives, and false negatives.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

It helps in visualizing the performance of a classifier in greater detail and is used to calculate other metrics like precision, recall, and F1-score.

6. Receiver Operating Characteristic (ROC) Curve

The ROC curve plots the true positive rate (recall) against the false positive rate at different classification thresholds. The area under the ROC curve (AUC) provides a single value that summarizes the model's ability to discriminate between positive and negative classes.

$$\text{True Positive Rate (TPR)} = \text{Recall}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

A higher AUC indicates a better-performing model.

7. Logarithmic Loss (Log Loss)

Logarithmic Loss quantifies the accuracy of a classifier by penalizing false classifications. It is particularly useful for models that output probabilities for the classes, such as logistic regression or neural networks.

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right]$$

8. Cross-Validation

Cross-validation is a technique used to assess the generalization ability of a model. In k-fold cross-validation, the dataset is split into k subsets. The model is trained on K - 1 of these subsets and tested on the remaining subset. This process is repeated k times, and the performance is averaged to give a more robust estimate of the model's accuracy.

4.1.6 Dataset Description

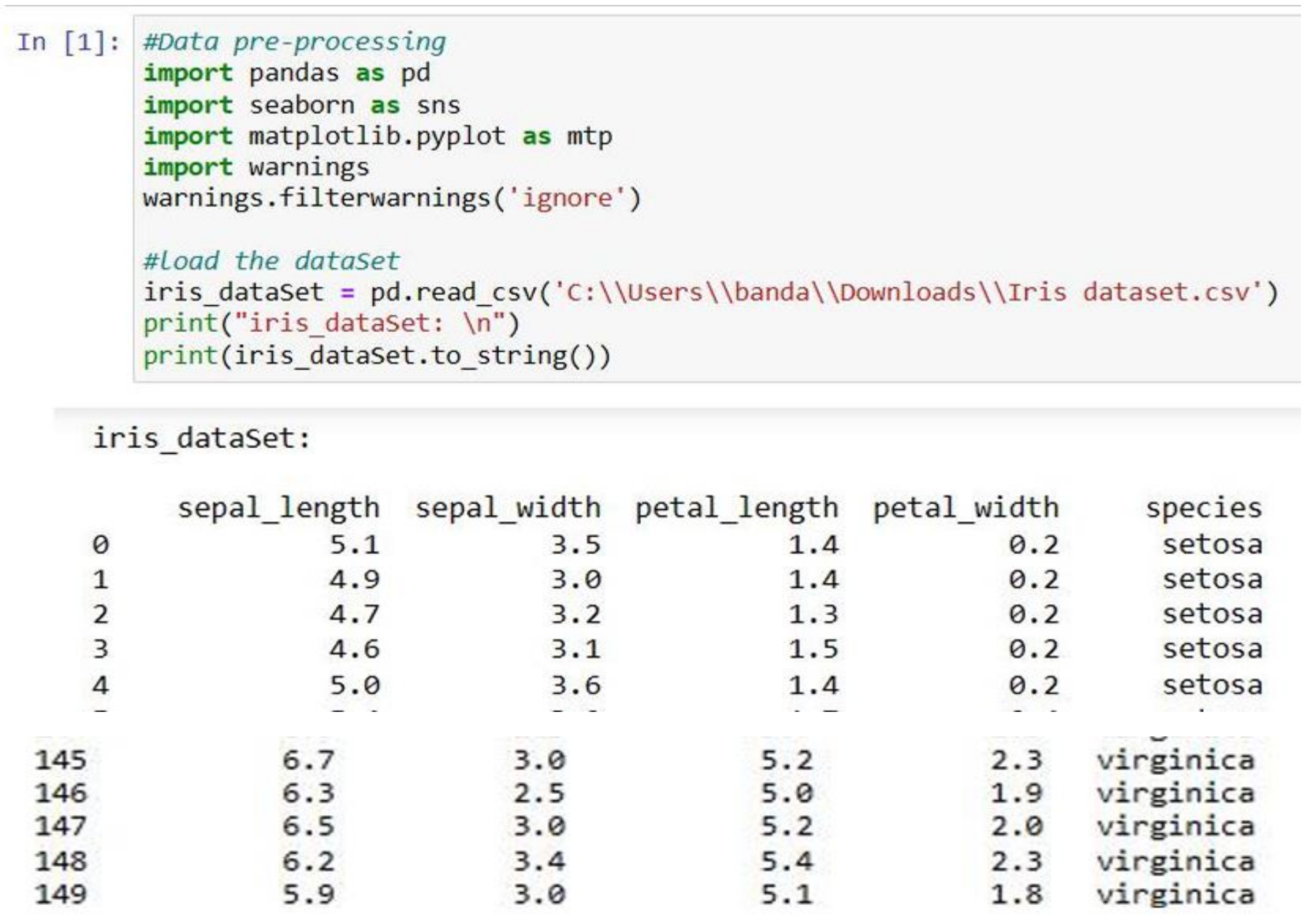
The Iris dataset, a standard benchmark in machine learning, consists of:

- 150 samples equally divided across three species.
- 4 features: sepal length, sepal width, petal length, and petal width (all measured in cm).
- The dataset is well-structured, balanced, and requires minimal preprocessing.

4.1.7 Challenges Faced

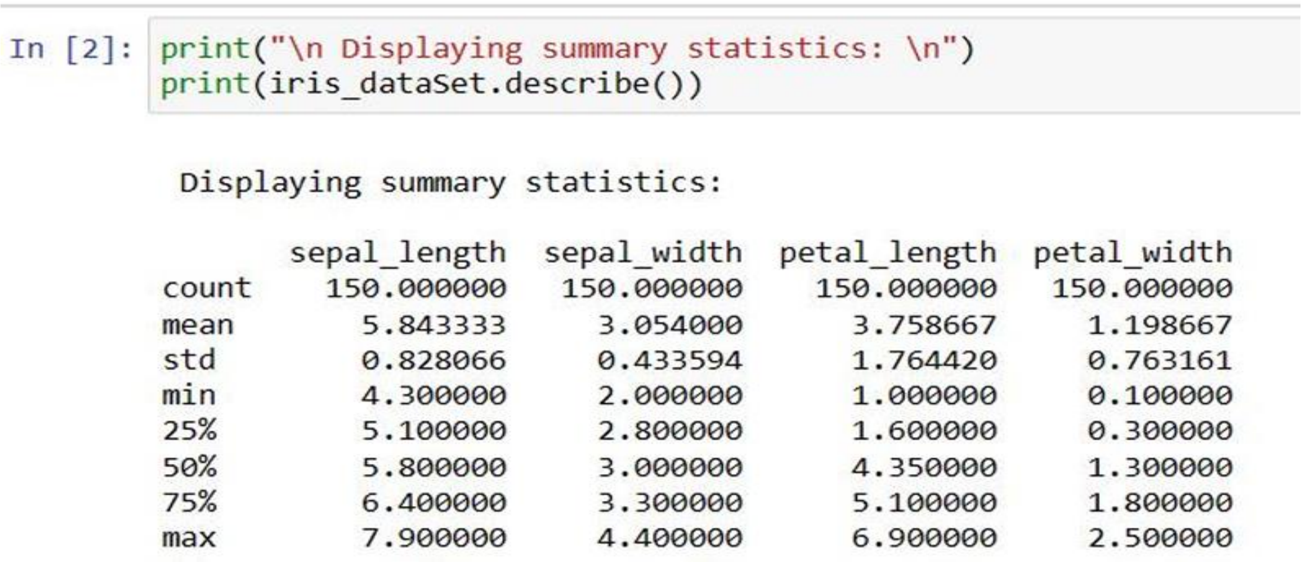
- **Boundary Overlap**: Some samples, particularly between the "Iris-versicolor" and "Iris-virginica" classes, had overlapping feature values, leading to occasional misclassification.
- **Parameter Selection**: Determining the optimal value of k required experimentation. Too small k (e.g., 1) caused overfitting, while a higher k smoothed boundaries excessively, reducing accuracy.
- **Outlier Sensitivity**: KNN was sensitive to outliers, as these points influenced the calculation of nearest neighbours and could mislead the classification.

4.1.8 Code Implementation



```
In [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.to_string())
```

```
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
145           6.7          3.0           5.2          2.3  virginica
146           6.3          2.5           5.0          1.9  virginica
147           6.5          3.0           5.2          2.0  virginica
148           6.2          3.4           5.4          2.3  virginica
149           5.9          3.0           5.1          1.8  virginica
```



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

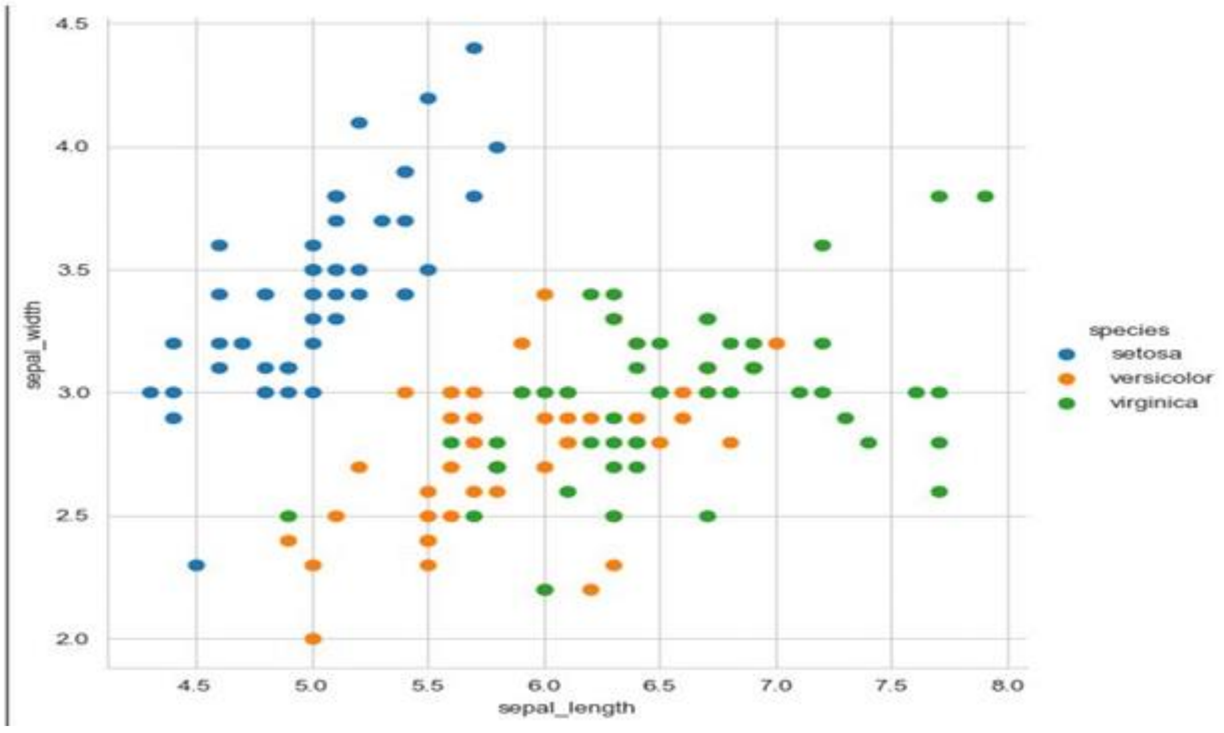
```
 Displaying summary statistics:

       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
```

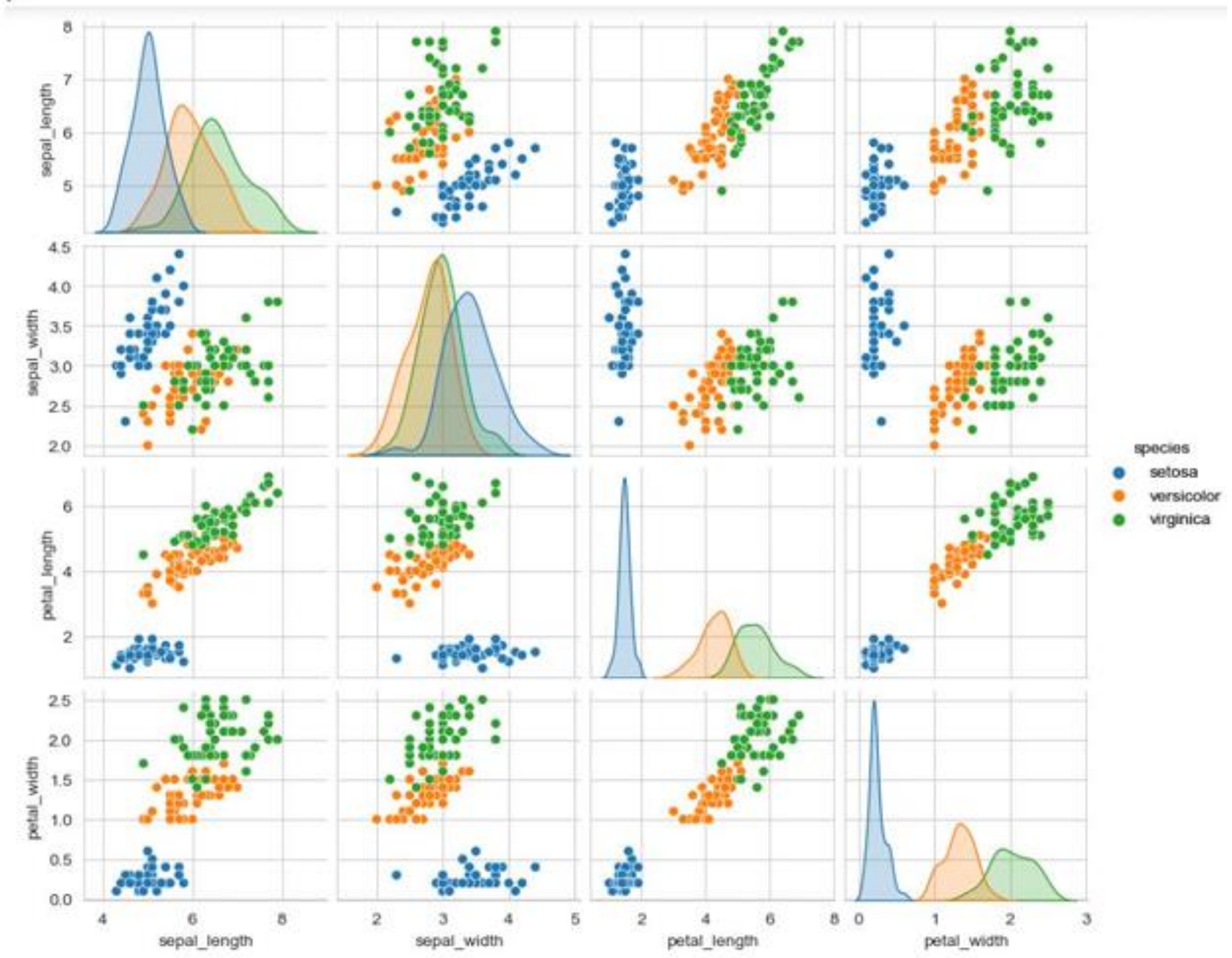
```
In [3]: print("\n Displaying information about the dataset: \n")
        print(iris_dataSet.info())
```

```
 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  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
```

```
In [4]: print("\n Visualizing data: \n")
        sns.set_style('whitegrid')
        sns.FacetGrid(iris_dataSet, hue = "species", height = 6).map(mtp.scatter, 'sepal_length', 'sepal_width').add_legend()
        sns.pairplot(iris_dataSet, hue = 'species', height = 2).add_legend()
        mtp.show()
```

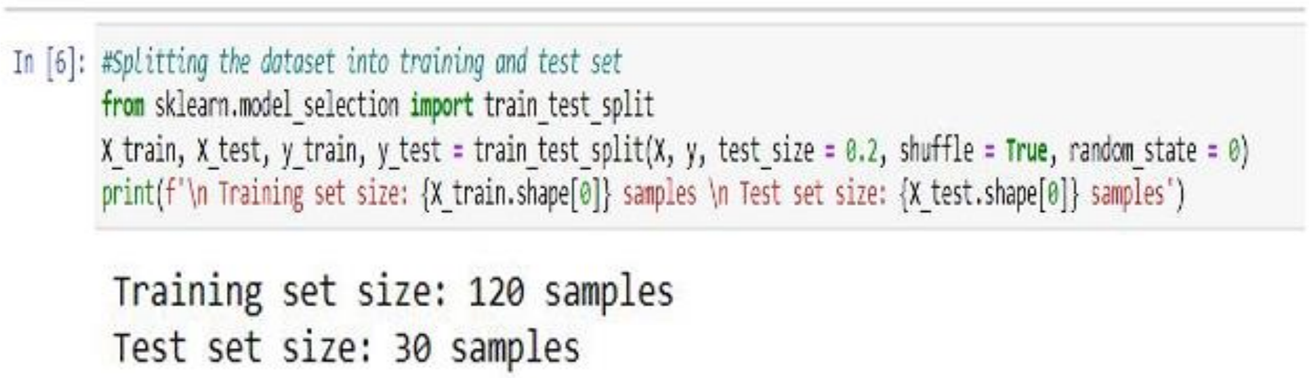
```
Visualizing data:
```





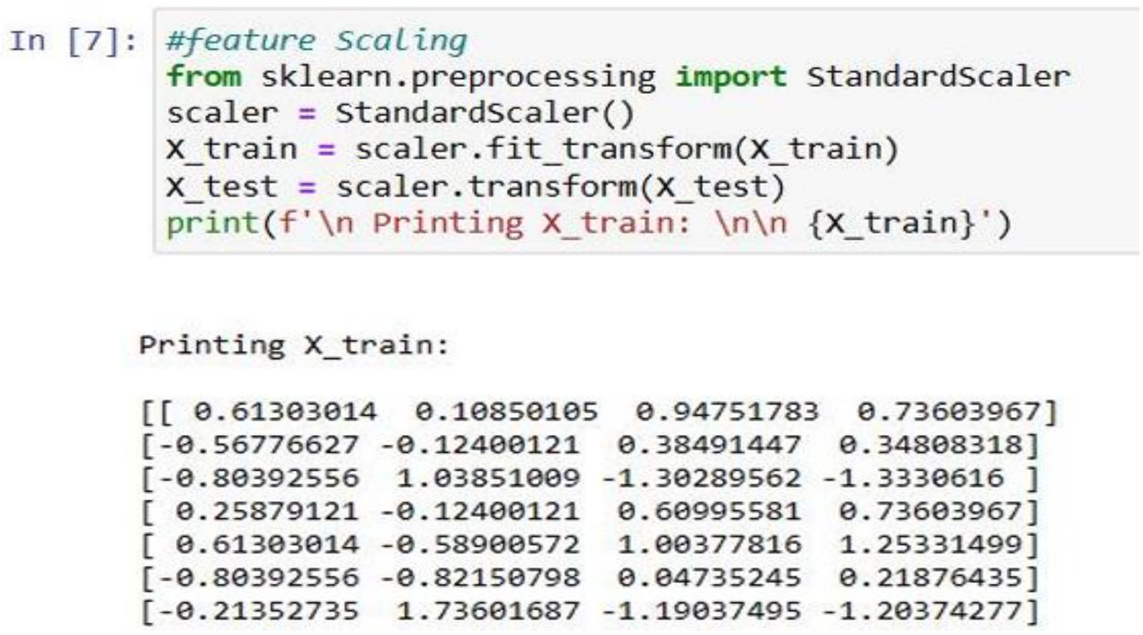
```
In [5]: #Extracting independent variable x and dependent variable y
        X = iris_dataSet.iloc[:, :4].values
        y = iris_dataSet.iloc[:, -1].values
        print(f'Displaying first 5 values of independent variable x: \n {X[:5]}')
        print(f'\n Displaying first 5 values of dependent variable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



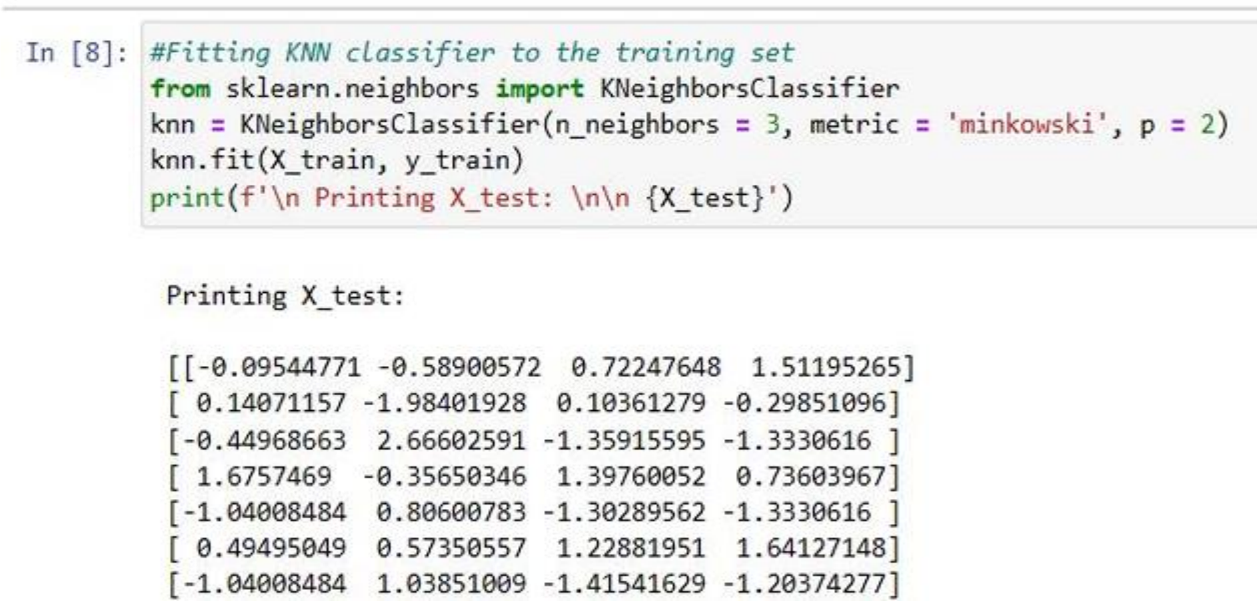
```
In [6]: #Splitting the dataset into training and test set
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, shuffle = True, random_state = 0)
        print(f'\n Training set size: {X_train.shape[0]} samples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

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



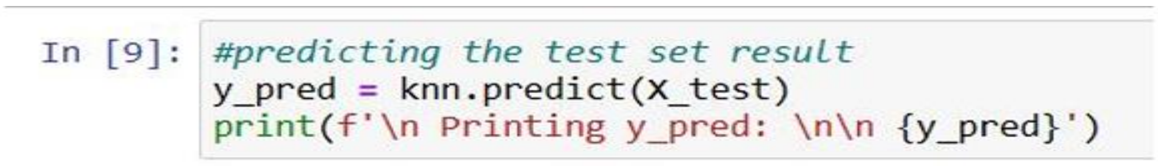
```
In [7]: #feature Scaling
        from sklearn.preprocessing import StandardScaler
        scaler = StandardScaler()
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)
        print(f'\n Printing X_train: \n\n {X_train}')
```

```
 Printing X_train:

 [[ 0.61303014  0.10850105  0.94751783  0.73603967]
 [-0.56776627 -0.12400121  0.38491447  0.34808318]
 [-0.80392556  1.03851009 -1.30289562 -1.3330616 ]
 [ 0.25879121 -0.12400121  0.60995581  0.73603967]
 [ 0.61303014 -0.58900572  1.00377816  1.25331499]
 [-0.80392556 -0.82150798  0.04735245  0.21876435]
 [-0.21352735  1.73601687 -1.19037495 -1.20374277]
```

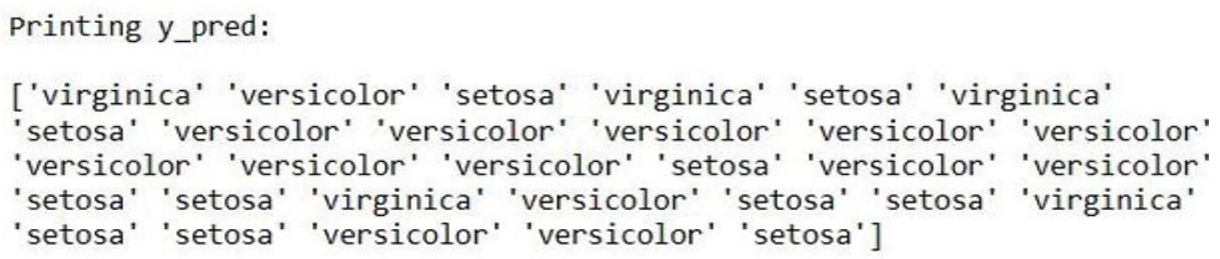


```
In [8]: #Fitting KNN classifier to the training set
        from sklearn.neighbors import KNeighborsClassifier
        knn = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p = 2)
        knn.fit(X_train, y_train)
        print(f'\n Printing X_test: \n\n {X_test}')
```

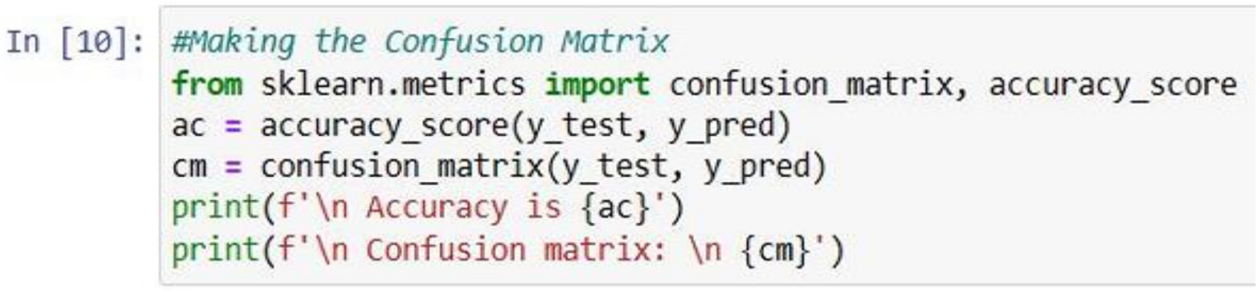
```
 Printing X_test:

 [[-0.09544771 -0.58900572  0.72247648  1.51195265]
 [ 0.14071157 -1.98401928  0.10361279 -0.29851096]
 [-0.44968663  2.66602591 -1.35915595 -1.3330616 ]
 [ 1.6757469  -0.35650346  1.39760052  0.73603967]
 [-1.04008484  0.80600783 -1.30289562 -1.3330616 ]
 [ 0.49495049  0.57350557  1.22881951  1.64127148]
 [-1.04008484  1.03851009 -1.41541629 -1.20374277]
```



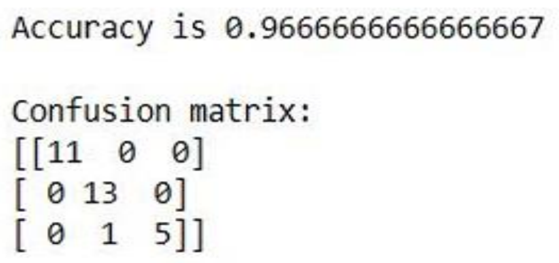
```
In [9]: #predicting the test set result
        y_pred = knn.predict(X_test)
        print(f'\n Printing y_pred: \n\n {y_pred}')
```



```
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']
```



```
In [10]: #Making the Confusion Matrix
         from sklearn.metrics import confusion_matrix, accuracy_score
         ac = accuracy_score(y_test, y_pred)
         cm = confusion_matrix(y_test, y_pred)
         print(f'\n Accuracy is {ac}')
         print(f'\n Confusion matrix: \n {cm}')
```



```
Accuracy is 0.9666666666666667

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

4.1.9 Results and Observations

The KNN classifier yielded the following outcomes:

- **Accuracy**: The model achieved an impressive accuracy of **96.67%** on the test set.
- **Confusion Matrix**: The results showed minimal misclassification across species, validating the model's effectiveness.

Observations:

- KNN effectively classified species with separable features like petal length and width, demonstrating its strength with small datasets.

- The minor misclassifications occurred between **Iris-versicolor** and **Iris-virginica**, likely due to overlapping feature distributions.

4.1.10 Conclusion

This mini project provided valuable insights into:

- The practical application of KNN as a simple yet powerful classification technique.
- The importance of data preprocessing and parameter tuning in improving model performance.
- The role of visualization in understanding data relationships and guiding algorithm selection.

4.2 MAJOR PROJECT: CUSTOMER SEGMENTATION

4.2.1 Project Objective

To segment customers into distinct groups based on purchasing behavior and demographic data using the K-Means clustering algorithm. This segmentation helps businesses tailor marketing strategies and optimize product offerings for specific customer groups.

4.2.2 Existing Approach

- Customer segmentation is typically performed using heuristic methods, which are manual, time-consuming, and less accurate.
- Limited ability to handle large datasets effectively, often leading to inaccurate segment definitions.
- Lack of actionable insights due to insufficient clustering techniques and outdated tools.

4.2.3 Proposed Solution

- Implementation of an automated segmentation system using the K-Means clustering algorithm for high accuracy and scalability.
- Integration of robust preprocessing techniques to handle outliers and normalize data for better cluster formation.
- Provision for visualizing customer segments for easy interpretation and actionable insights.

4.2.4 Requirements

a) Hardware

- Processor: **Intel i5 or higher**
- RAM**: 8GB**
- Storage**: 1TB Hard Disk**

b) Software

- **Python 3.8+**
- Pandas, NumPy, Matplotlib, and Scikit-learn libraries
- Jupyter, Anaconda, Google Colab

4.2.5 Methodology

Algorithm Used

Clustering is a widely used exploratory data analysis technique for understanding the underlying structure of data. It involves identifying subgroups, or clusters, within the dataset such that the data points within the same cluster are highly similar, while those in different clusters are significantly distinct.

The goal is to create homogeneous groups where data points in each cluster are as alike as possible, based on a chosen similarity measure. Common measures include Euclidean distance and correlation-based distance, with the selection depending on the specific application and nature of the data.

Clustering analysis can be performed either by finding subgroups of samples based on features or by grouping features based on samples. In the context of clustering based on features, this technique is widely applied across various domains. For example, in **market segmentation**, clustering helps identify groups of customers with similar behaviors or attributes, which allows businesses to tailor their strategies effectively. In **image segmentation and compression**, clustering is used to group similar regions of an image, facilitating better processing and storage. Similarly, in **document clustering**, it organizes documents into clusters based on topics, aiding in efficient retrieval and analysis.

Unlike supervised learning, clustering is an unsupervised learning method because there is no ground truth to compare the output of the clustering algorithm with true labels, making it difficult to directly evaluate its performance.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

CLUSTERING METHODS

- **Density-Based Methods**: These methods consider the clusters as the dense region having some similarity and different from the lower dense region of the space. These methods have good accuracy and ability to merge two clusters. Example DBSCAN
- **Hierarchical Based Methods**: The clusters formed in this method forms a tree-type structure based on the hierarchy. New clusters are formed using the previously formed one. It is divided into two types: a) Agglomerative b) Divisive
- **Partitioning Methods**: These methods partition the objects into k clusters and each partition forms one cluster. This method is used to optimize an objective criterion similarity function such as when the distance is a major parameter.
- **Grid-based Methods**: In this method the data space is formulated into a finite number of cells that form a grid-like structure. All the clustering operation done on these grids are fast and independent of the number of data objects.

DISTANCE MEASURE

An important component of a clustering algorithm is the distance measure between data points. If the components of the data instance vectors are all in the same physical unit then it is possible that the simple Euclidean distance metric is sufficient to successfully group similar data instances. However, even in this case the Euclidean distance can sometimes be misleading.

Figure shown below illustrates this with an example of the width and height measurements of an object. Despite both measurements being taken in the same physical units, an informed decision has to be made as to the relative scaling. As the figure shows, different scalings can lead to different clusterings.

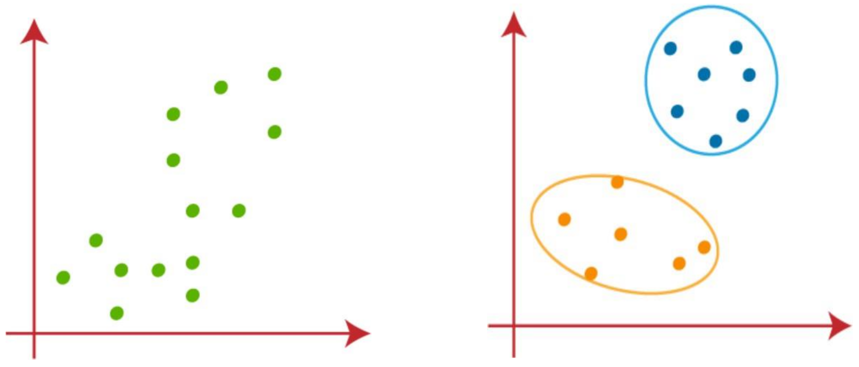


Figure: Formation of clusters

Measuring Algorithm Performance

One of the most important considerations regarding the ML model is assessing its performance, or you can say the model's quality. In the case of supervised learning algorithms, evaluating the quality of our model is easy because we already have labels for every example.

On the other hand, in the case of unsupervised learning algorithms, we are not that much blessed because we deal with unlabeled data. But still, we have some metrics that give the practitioner insight into the happening of change in clusters depending on the algorithm.

Now a good clustering algorithm aims to create clusters whose:

- The **intra-cluster similarity is high** (The data that is present inside the cluster is similar to one another)
- The **inter-cluster similarity is less** (Each cluster holds information that isn't similar to the other)

K-Means Clustering Algorithm

The K-means algorithm is an iterative process that aims to partition a dataset into K predefined, distinct, and non-overlapping clusters, where each data point belongs to only one cluster. The algorithm strives to make the data points within each cluster as similar as possible, while maximizing the differences between clusters. It works by assigning each data point to a cluster in such a way that the sum of the squared distances between the data points and the cluster's centroid (the arithmetic mean of all points in the cluster) is minimized. The less variation there is within clusters, the more homogeneous and similar the data points are within the same group.

The way k-means algorithm works is as follows:

1. Specify number of clusters K.
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

- Compute the sum of the squared distance between data points and all centroids
- Assign each data point to the closest cluster (centroid).
- Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

The approach means follows to solve the problem is called Expectation-Maximization. The E-step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster.

The objective function is:

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \|x^i - \mu_k\|^2 \qquad (1)$$

where w_{ik}=1 for data point xi if it belongs to cluster k; otherwise, w_{ik}=0. Also, u_k is the centroid of x_i's cluster.

It's a minimization problem of two parts. We first minimize **J** w.r.t. w_{ik} and treat u_k fixed.

Then we minimize **J** w.r.t. u_k and treat w_{ik} fixed. Technically speaking, we differentiate **J** w.r.t. w_{ik} first and update cluster assignments (E-step). Then we differentiate **J** w.r.t. u_k and recompute the centroids after the cluster assignments from previous step (M- step). Therefore, E-step is:

$$\frac{\partial J}{\partial w_{ik}} = \sum_{i=1}^{m} \sum_{k=1}^{K} \|x^i - \mu_k\|^2$$

$$\Rightarrow w_{ik} = \begin{cases} 1 & \text{if } k = argmin_j \|x^i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

In other words;

$$\frac{\partial J}{\partial \mu_k} = 2\sum_{i=1}^{m} w_{ik}(x^i - \mu_k) = 0$$

$$\Rightarrow \mu_k = \frac{\sum_{i=1}^{m} w_{ik}x^i}{\sum_{i=1}^{m} w_{ik}}$$

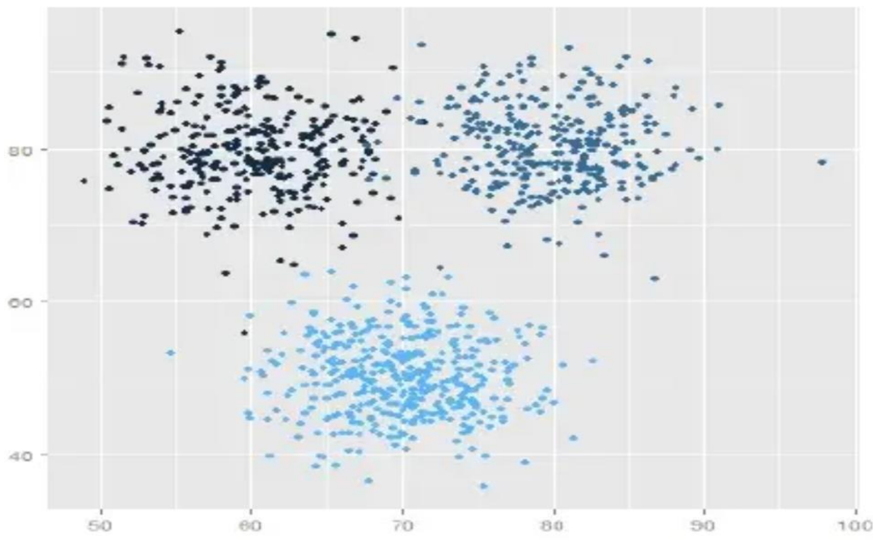


Figure: K-means clustering where k = 3

K-means is an unsupervised clustering algorithm designed to partition un-labelled data into a certain number (thats the "K") of distinct groupings. In other words, k-means finds observations that share important characteristics and classifies them together into clusters. A good clustering solution is one that finds clusters such that the observations within each cluster are more similar than the clusters themselves.

There are countless examples of where this automated grouping of data can be extremely useful. For example, consider the case of creating an online advertising campaign for a brand-new range of products being released to the market. While we could display a single generic advertisement to the entire population, a far better approach would be to divide the population into clusters of people who hold shared characteristics and interests displaying customized advertisements to each group.

K-means is an algorithm that finds these groupings in big datasets where it is not feasible to be done by hand. The intuition behind the algorithm is actually pretty straight forward. To begin, we choose a value for k (the number of clusters) and randomly choose an initial centroid (center coordinates) for each cluster.

We then apply a two step process:

1. Assignment step - Assign each observation to its nearest center.
2. Update step - Update the centroids as being the center of their respective observation.

We repeat these two steps over and over until there is no further change in the clusters. At this point the algorithm has converged and we may retrieve our final clusterings.

As we are only interested in the best clustering solution for a given choice of k, a common solution to this problem is to run k-means multiple times, each time with different randomised initial centroids, and use only the best solution. In other words, always run k-means multiple times to ensure we find a solution close to the global minima.

ADVANTAGES

1. **Guaranteed Convergence**:

   K-Means guarantees convergence as it iteratively minimizes the sum of squared distances within clusters.

2. **Simplicity:**

   K-Means is straightforward to implement and understand, making it a popular choice for clustering tasks.

3. **Scalability:**

   It works efficiently with large datasets due to its low computational complexity.

4. **Effective for Well-Defined Clusters:**

   Works best when the data has clearly defined, globular clusters of roughly equal size.

DISADVANTAGES

1. **Assumes Spherical Clusters**:

   K-Means assumes that clusters are spherical and equally sized, which may not be true for complex datasets with irregularly shaped clusters.

2. **Requires Predefined Number of Clusters (K):**

   The algorithm requires the number of clusters (K) to be specified beforehand, which may not always be known or easy to determine.

3. **Not Suitable for Non-Linear Data**:

   K-Means performs poorly with datasets where clusters are not separated linearly or have overlapping boundaries.

EVALUATION METHODS

Evaluating clustering performance can be challenging since there are no predefined labels in unsupervised learning. However, various evaluation methods exist to assess the quality of clusters. These can be broadly categorized into internal, external, and relative evaluation techniques.

- **Silhouette Score**

  Measures how similar a data point is to its own cluster (cohesion) compared to other clusters (separation).

$$S = \frac{b - a}{\max(a, b)}$$

- **Inertia (Within-Cluster Sum of Squares - WCSS)**

  Quantifies how tightly the data points are grouped around the centroids. Lower inertia suggests better compactness of clusters.

$$WCSS = \sum_{k=1}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2$$

- **Adjusted Rand Index (ARI)**

  Measures the similarity between the predicted clustering and the ground truth by considering all pairs of samples.

- **Elbow Method**

  Plots WCSS against the number of clusters (K) and identifies the "elbow point" where the rate of decrease sharply diminishes. This helps determine the optimal K.

4.2.6 Dataset Description

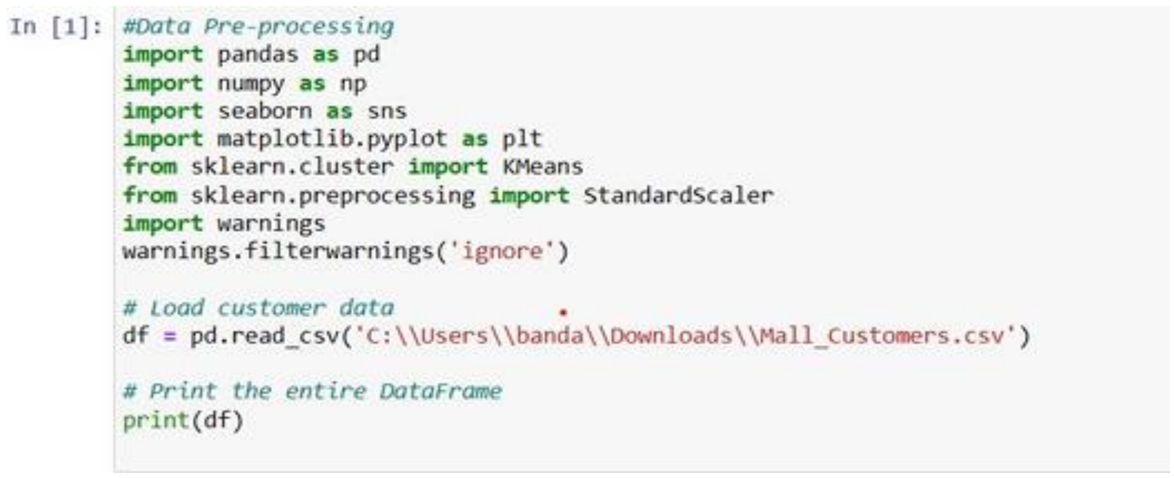
The Mall Customers Dataset consists of customer information including attributes such as Age, Gender, Annual Income, and Spending Score. It contains 200 rows and 5 columns, with the primary features being Annual Income and Spending Score. This dataset is typically used for customer segmentation and marketing analysis, as it provides insights into customer behavior based on income and spending patterns. The dataset helps businesses categorize customers into groups with similar traits, enabling more personalized marketing strategies.

4.2.7 Challenges Faced

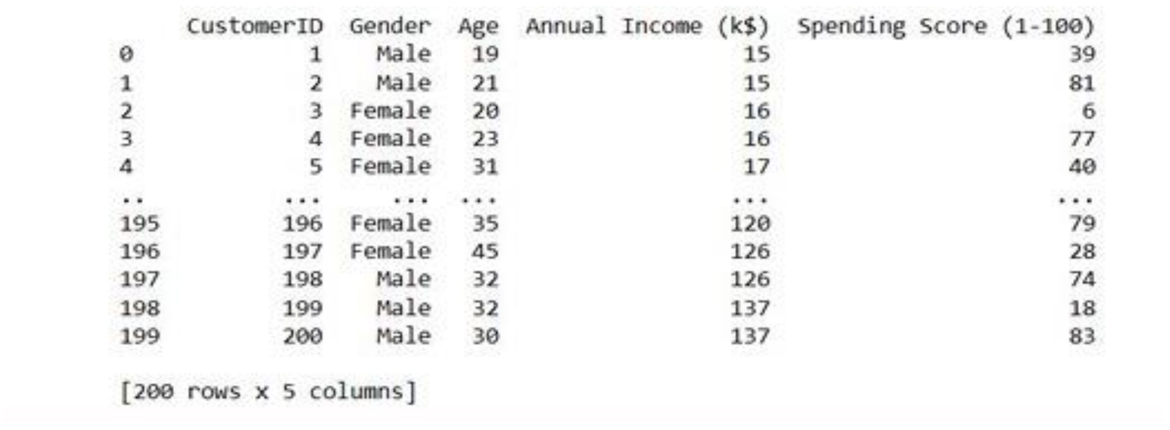
- **Choosing the Number of Clusters (K):** While the elbow method suggested that 4 clusters were optimal, there was no definitive way to confirm this, as the elbow point wasn't very distinct. The process involved trial and error, using domain knowledge to refine the choice of K.

- **Data Imbalance:** The dataset contained more customers in certain ranges of income and spending score, leading to uneven cluster distribution. This caused some clusters to be more populated than others, affecting the quality and interpretability of the results.
- **Handling Missing Values:** There were a few missing values in the dataset, especially in the 'Age' column. Handling missing data by using imputation techniques or removing rows with missing values was necessary to avoid errors during model training.
- **Feature Scaling:** The 'Annual Income (k$)' and 'Spending Score (1-100)' features had different ranges and units, which meant that K-Means could bias the clustering based on the feature with a larger range. Scaling the data using StandardScaler was necessary to bring all features onto a similar scale.
- **Sensitivity to Initial Centroid Placement:** Despite using the K-Means++ initialization method to optimize the placement of initial centroids, the algorithm still struggled with suboptimal cluster assignments in certain iterations. Multiple runs were needed to find the most stable clustering solution.

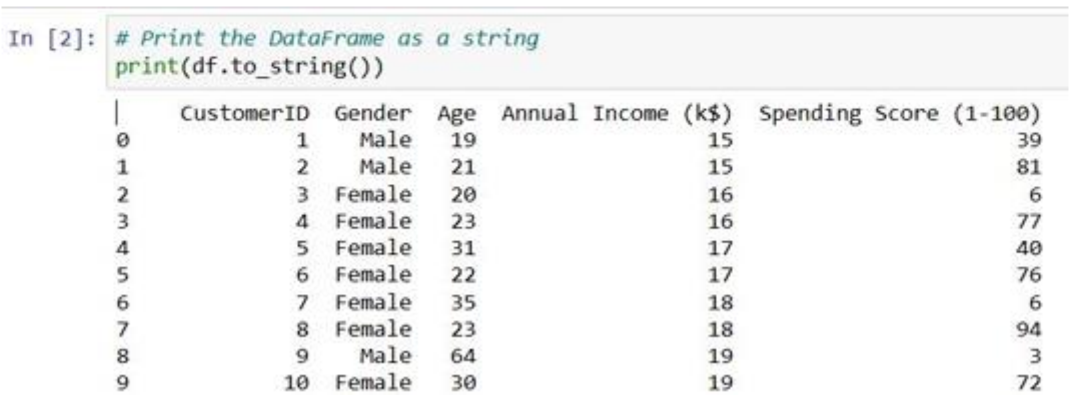
4.2.8 Code Implementation



```
In [1]: #Data Pre-processing
        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 warnings
        warnings.filterwarnings('ignore')

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

        # Print the entire DataFrame
        print(df)
```

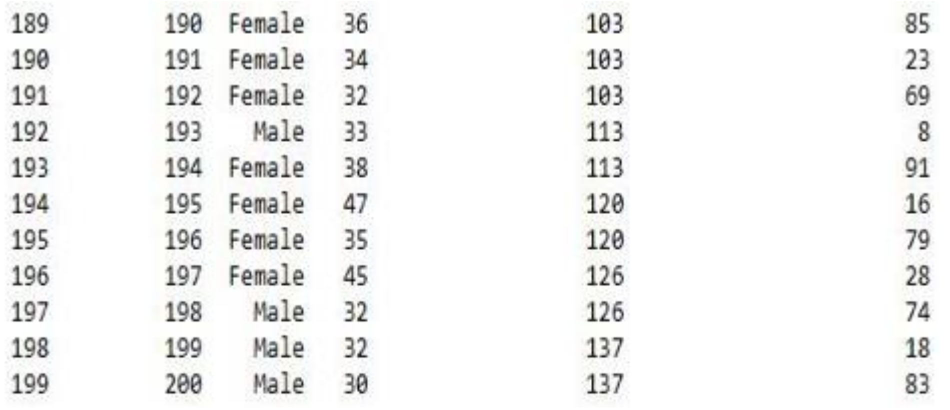


```
     CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0             1    Male   19                  15                      39
1             2    Male   21                  15                      81
2             3  Female   20                  16                       6
3             4  Female   23                  16                      77
4             5  Female   31                  17                      40
..          ...     ...  ...                 ...                     ...
195         196  Female   35                 120                      79
196         197  Female   45                 126                      28
197         198    Male   32                 126                      74
198         199    Male   32                 137                      18
199         200    Male   30                 137                      83

[200 rows x 5 columns]
```

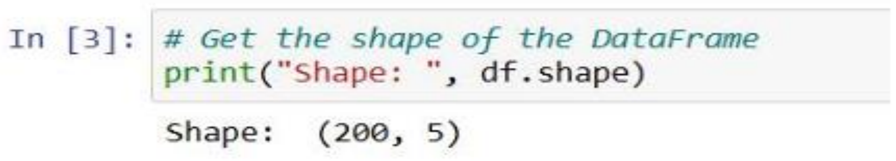


```
In [2]: # Print the DataFrame as a string
        print(df.to_string())
```

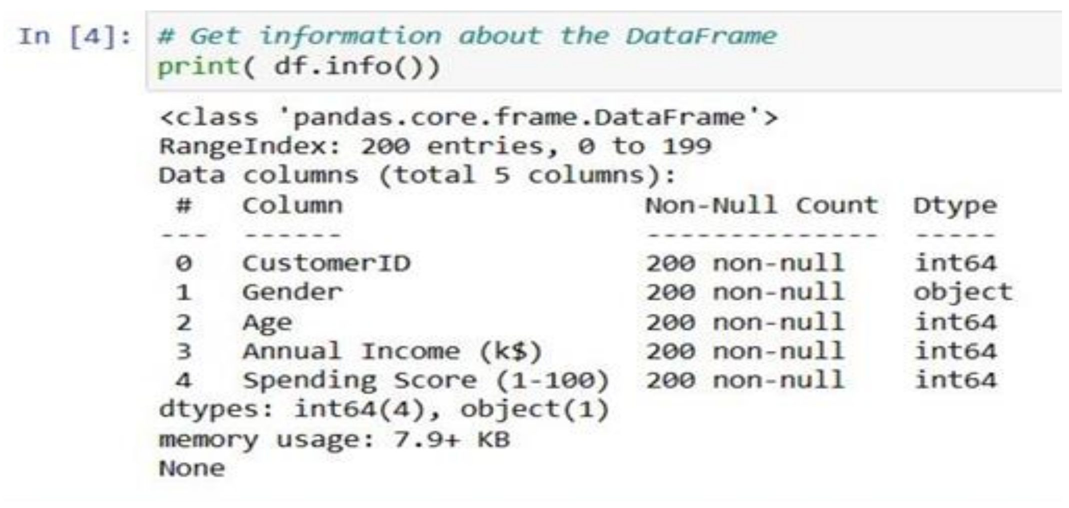
```
     CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0             1    Male   19                  15                      39
1             2    Male   21                  15                      81
2             3  Female   20                  16                       6
3             4  Female   23                  16                      77
4             5  Female   31                  17                      40
5             6  Female   22                  17                      76
6             7  Female   35                  18                       6
7             8  Female   23                  18                      94
8             9    Male   64                  19                       3
9            10  Female   30                  19                      72
```



```
189         190  Female   36                 103                      85
190         191  Female   34                 103                      23
191         192  Female   32                 103                      69
192         193    Male   33                 113                       8
193         194  Female   38                 113                      91
194         195  Female   47                 120                      16
195         196  Female   35                 120                      79
196         197  Female   45                 126                      28
197         198    Male   32                 126                      74
198         199    Male   32                 137                      18
199         200    Male   30                 137                      83
```

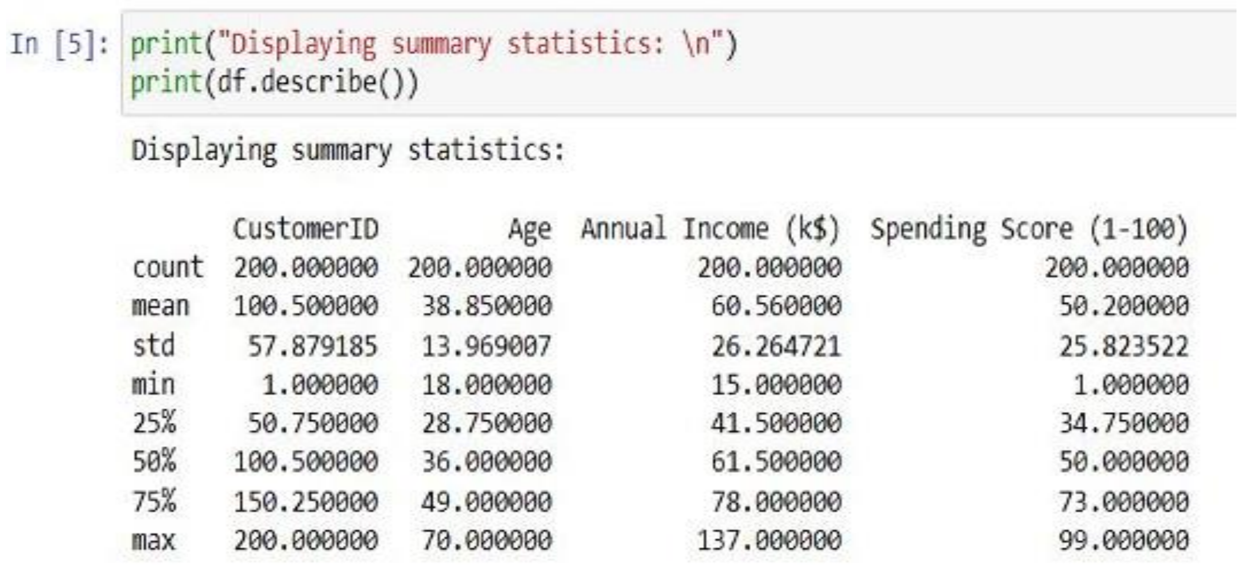


```
In [3]: # Get the shape of the DataFrame
        print("Shape: ", df.shape)

        Shape:  (200, 5)
```



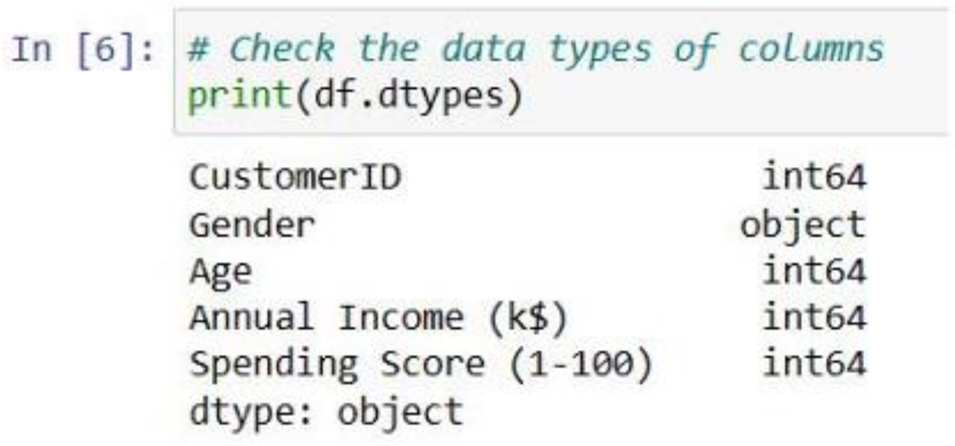
```
In [4]: # Get information about the DataFrame
        print( df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   CustomerID              200 non-null    int64 
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64 
 3   Annual Income (k$)      200 non-null    int64 
 4   Spending Score (1-100)  200 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
```



```
In [5]: print("Displaying summary statistics: \n")
        print(df.describe())
```

```
Displaying 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
```



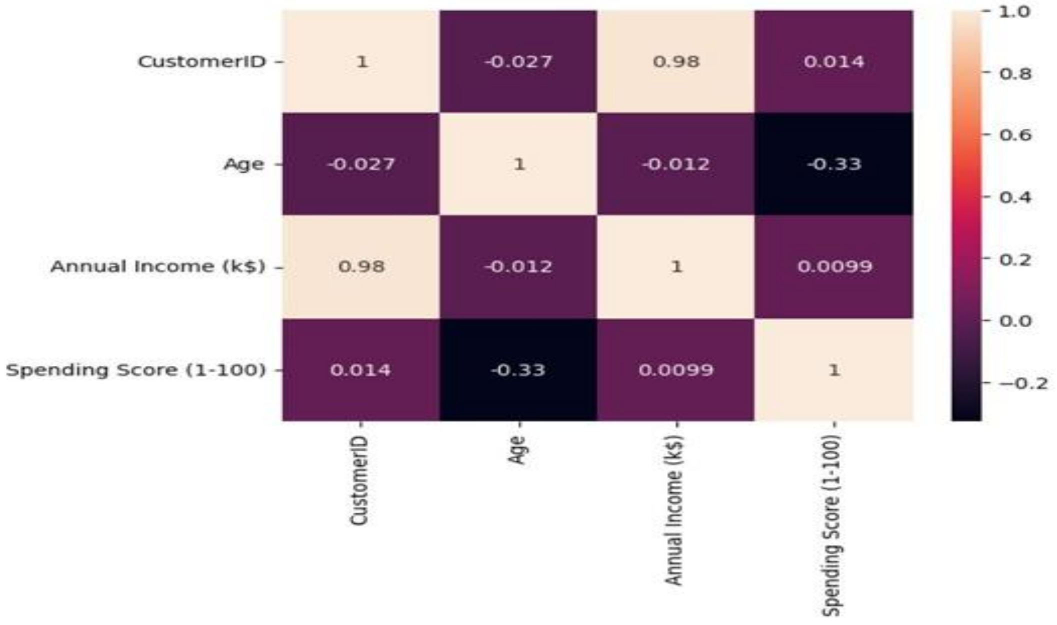
```
In [6]: # Check the data types of columns
        print(df.dtypes)
```

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

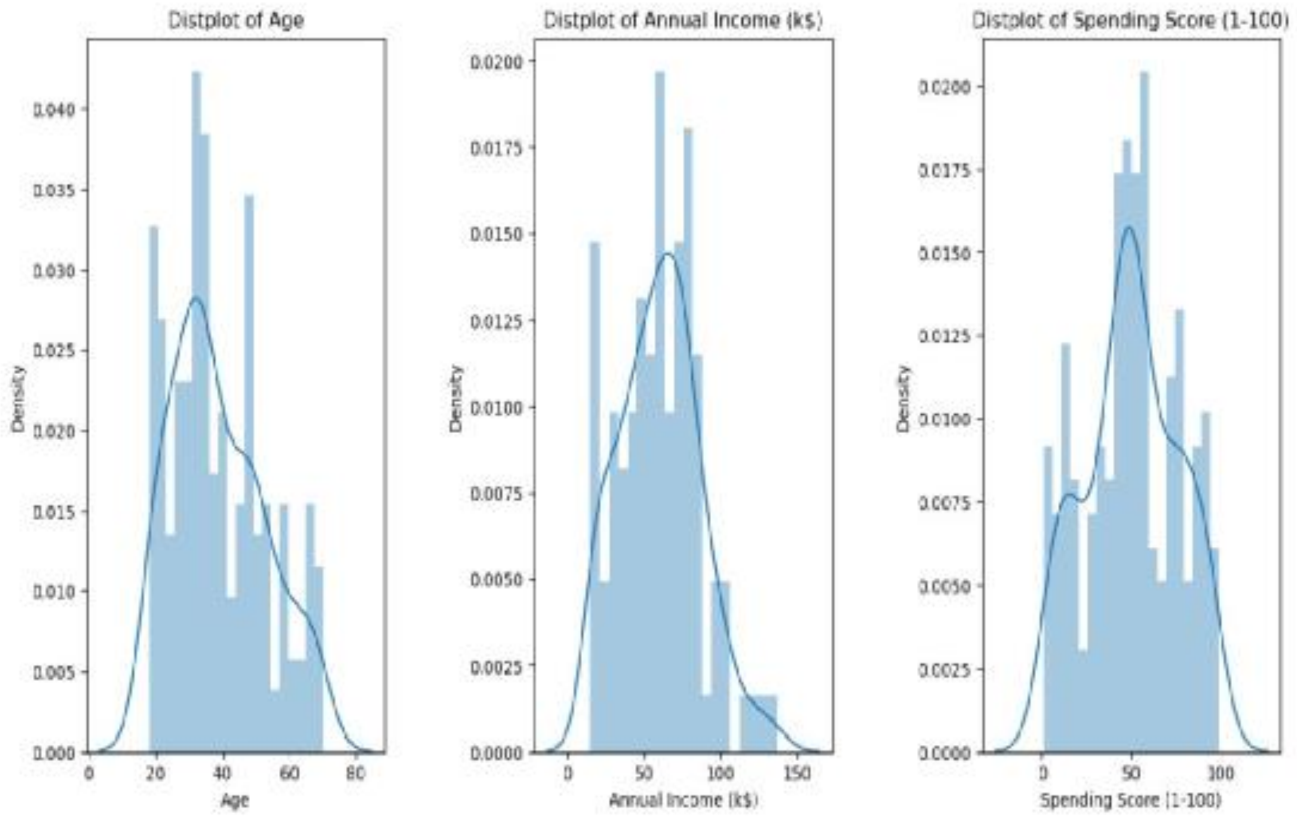
```
In [7]: # Select relevant features for clustering
        features = ['Annual Income (k$)', 'Spending Score (1-100)']
        X = df[features]
        print(X)
```

```
     Annual Income (k$)  Spending Score (1-100)
0                    15                      39
1                    15                      81
2                    16                       6
3                    16                      77
4                    17                      40
..                  ...                     ...
195                 120                      79
196                 126                      28
197                 126                      74
198                 137                      18
199                 137                      83

[200 rows x 2 columns]
```

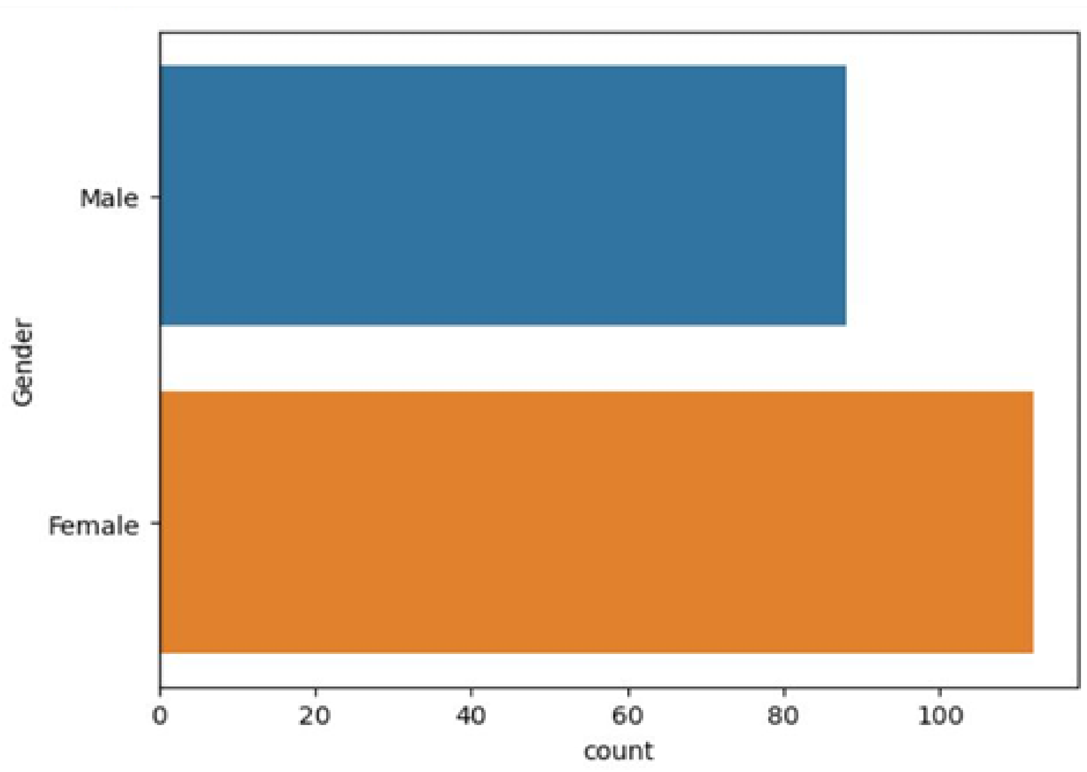
```
In [8]: # Plot correlation matrix heatmap
        plt.figure()
        sns.heatmap(df.corr(), annot = True)
        plt.show()
```



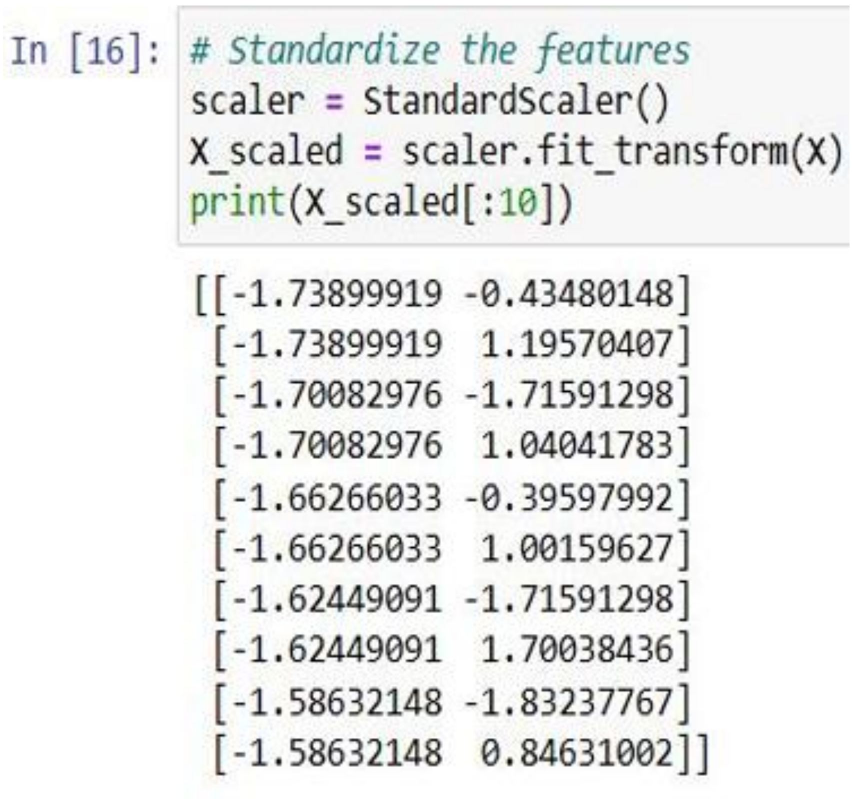
```
In [9]: # Plot distribution plots for selected features
        plt.figure(1, figsize = (15, 6))
        n = 0
        for x in [ 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
            n += 1
            plt.subplot(1, 3, n)
            plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
            sns.distplot(df[x], bins = 20)
            plt.title('Distplot of {}'.format(x))
        plt.show()
```



```
In [13]: # Plot countplot for 'Gender'
         plt.figure()
         sns.countplot(y = 'Gender', data = df)
         plt.show()
```



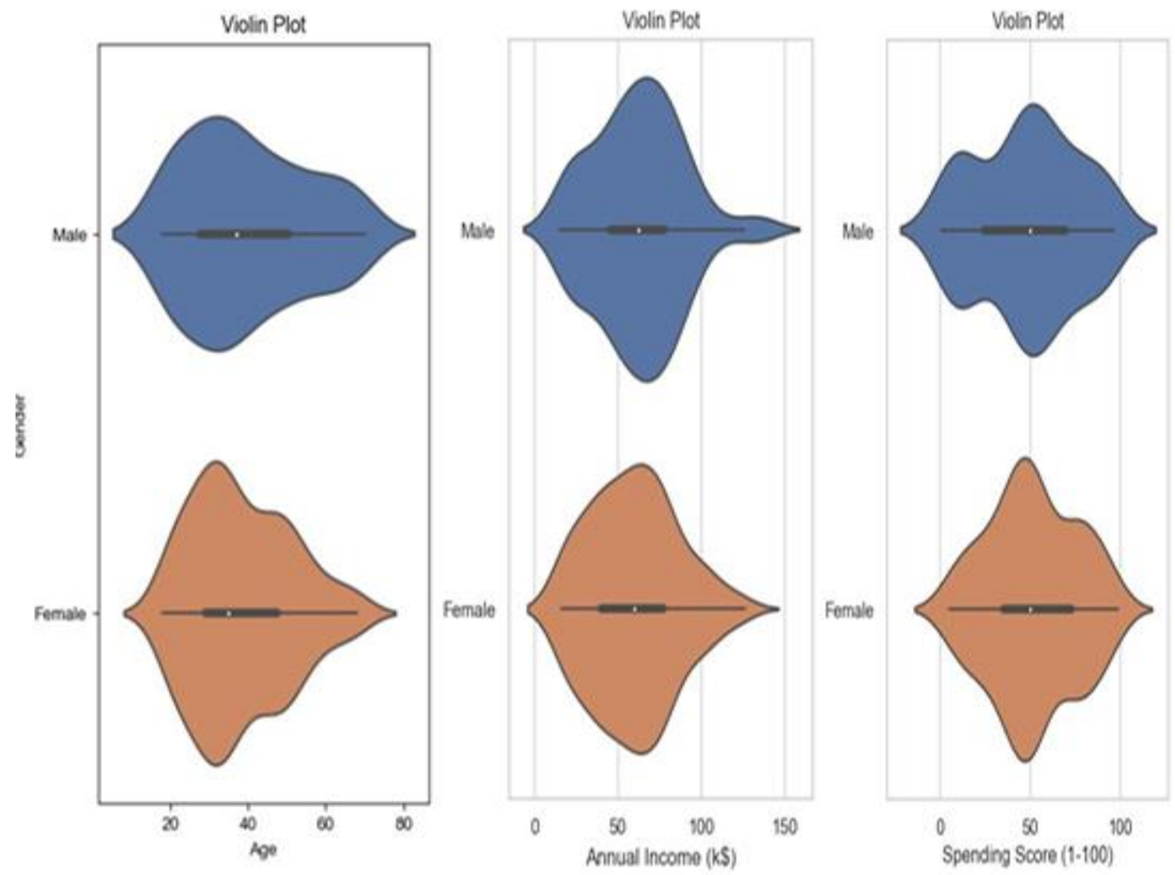
Observation: Female are more likely to shop than male.



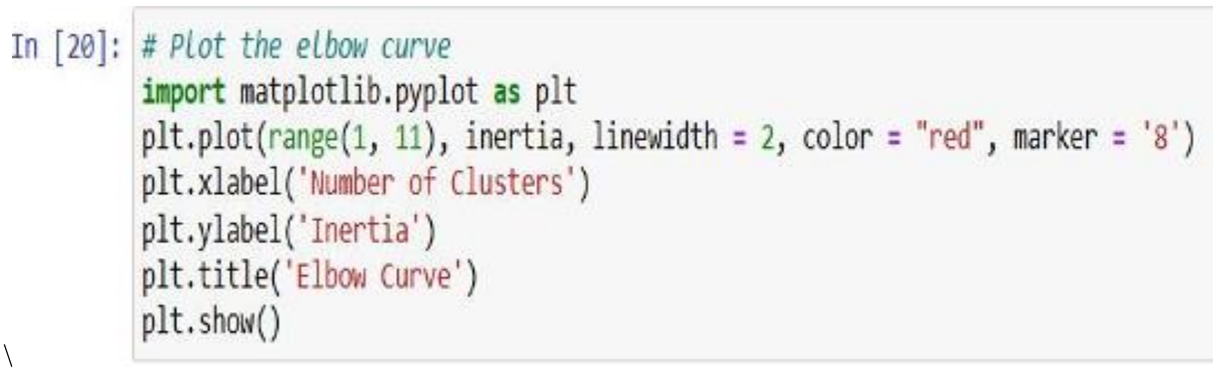
```
In [16]: # Standardize the features
         scaler = StandardScaler()
         X_scaled = scaler.fit_transform(X)
         print(X_scaled[:10])
```

```
[[-1.73899919 -0.43480148]
 [-1.73899919  1.19570407]
 [-1.70082976 -1.71591298]
 [-1.70082976  1.04041783]
 [-1.66266033 -0.39597992]
 [-1.66266033  1.00159627]
 [-1.62449091 -1.71591298]
 [-1.62449091  1.70038436]
 [-1.58632148 -1.83237767]
 [-1.58632148  0.84631002]]
```

```
In [14]: # Plot violin plots for selected features
         plt.figure(1, figsize = (15, 7))
         n = 0
         for cols in [ 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
             n += 1
             plt.subplot(1, 3, n)
             sns.set(style = 'whitegrid')
             plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
             sns.violinplot(x = cols, y = 'Gender', data = df)
             plt.ylabel('Gender' if n == 1 else '')
             plt.title('Violin Plot')
         plt.show()
```

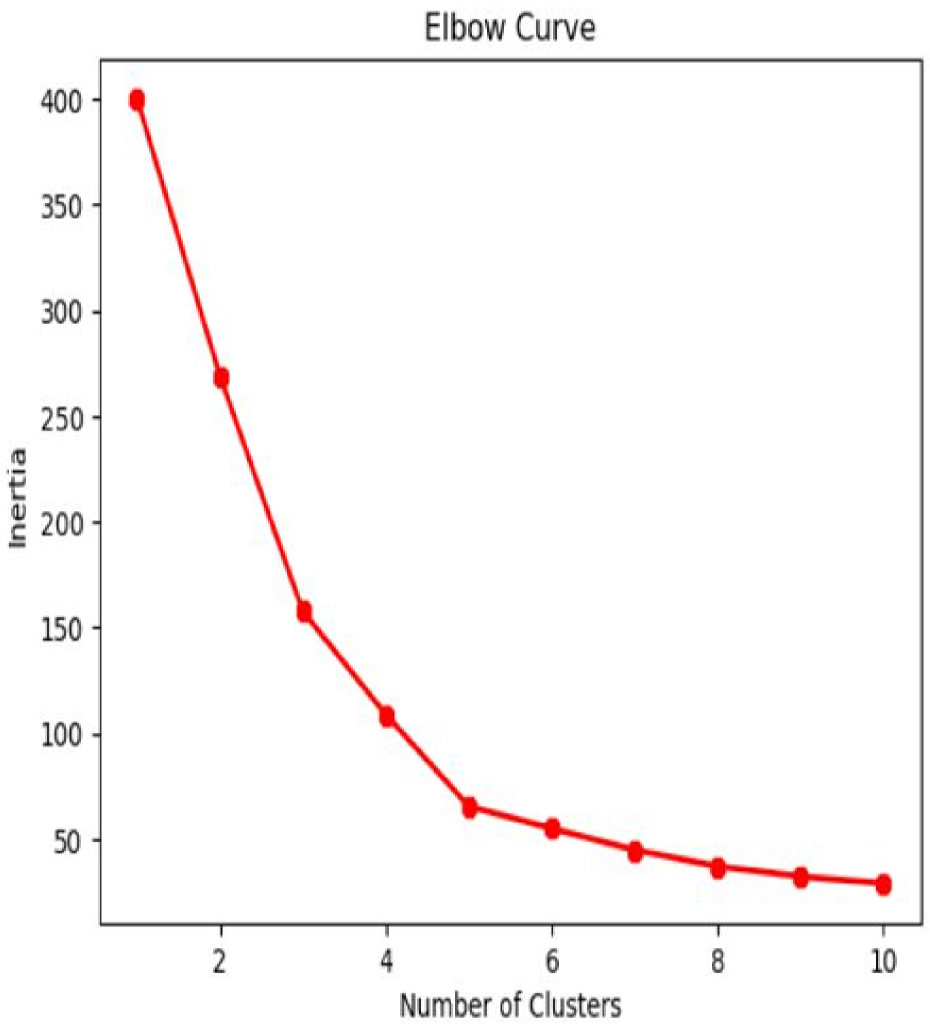


```
In [19]: # Determine the optimal number of clusters using the elbow method
         inertia = []
         for k in range(1, 11):
             kmeans = KMeans(n_clusters=k, init = 'k-means++', random_state = 0)
             kmeans.fit(X_scaled)
             inertia.append(kmeans.inertia_)
```



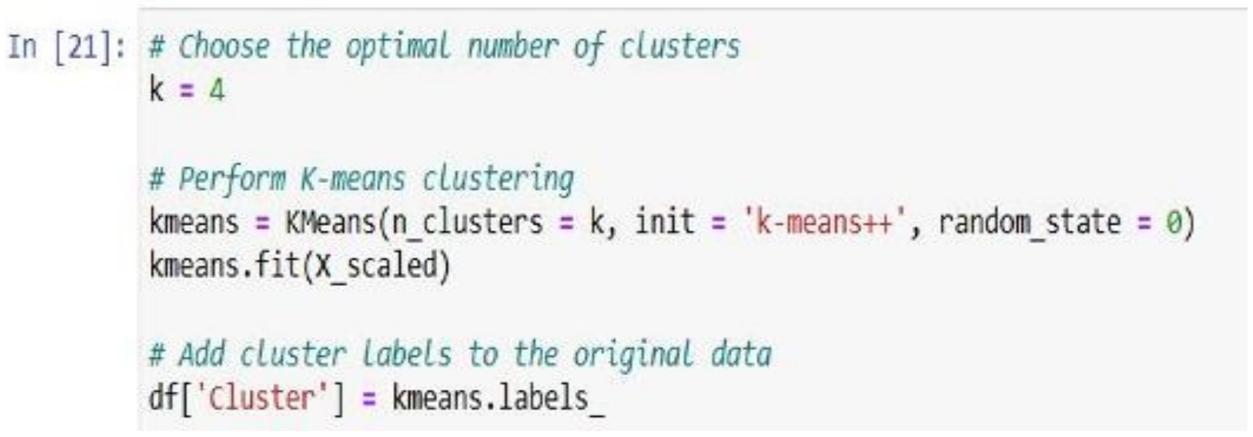
```
In [20]: # Plot the elbow curve
         import matplotlib.pyplot as plt
         plt.plot(range(1, 11), inertia, linewidth = 2, color = "red", marker = '8')
         plt.xlabel('Number of Clusters')
         plt.ylabel('Inertia')
         plt.title('Elbow Curve')
         plt.show()
```

\

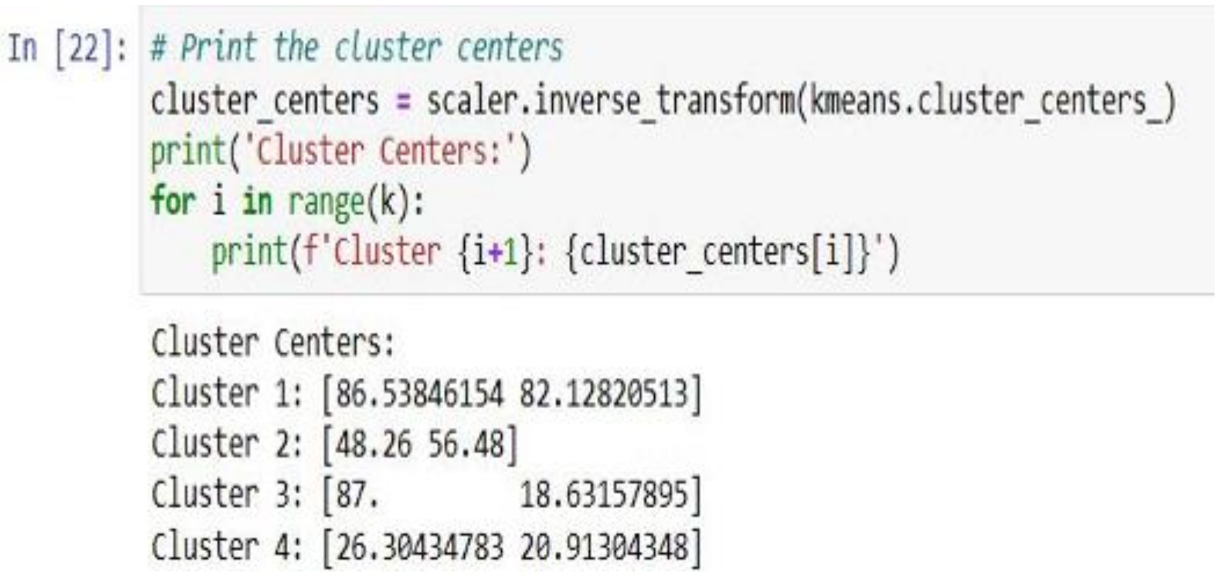


Observations:

- K < 4 Underfitting
- K > 4 Overfitting
- Elbow at K = 4

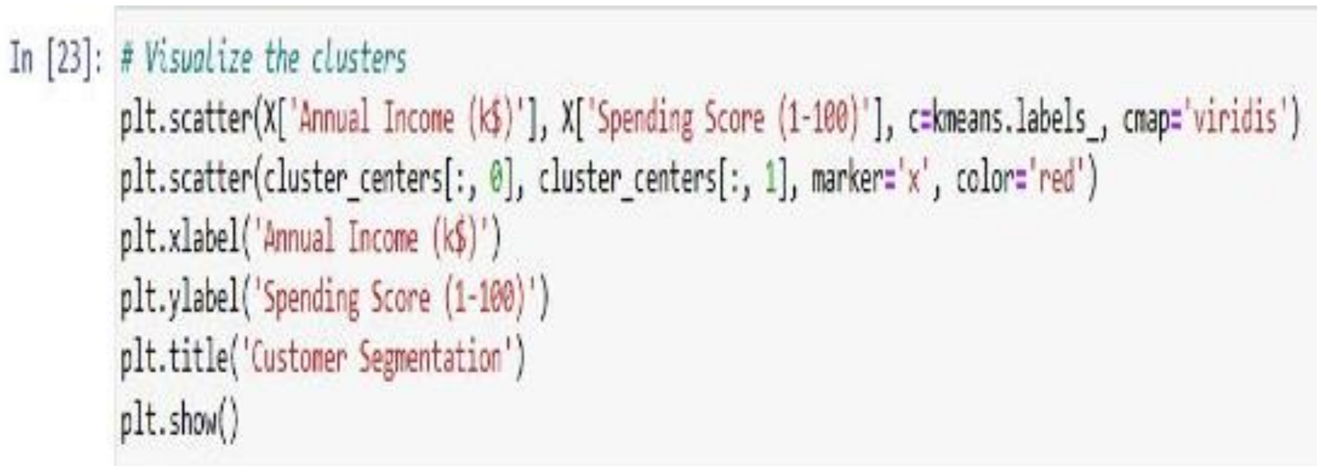


```
In [21]: # Choose the optimal number of clusters
         k = 4

         # Perform K-means clustering
         kmeans = KMeans(n_clusters = k, init = 'k-means++', random_state = 0)
         kmeans.fit(X_scaled)

         # Add cluster labels to the original data
         df['Cluster'] = kmeans.labels_
```

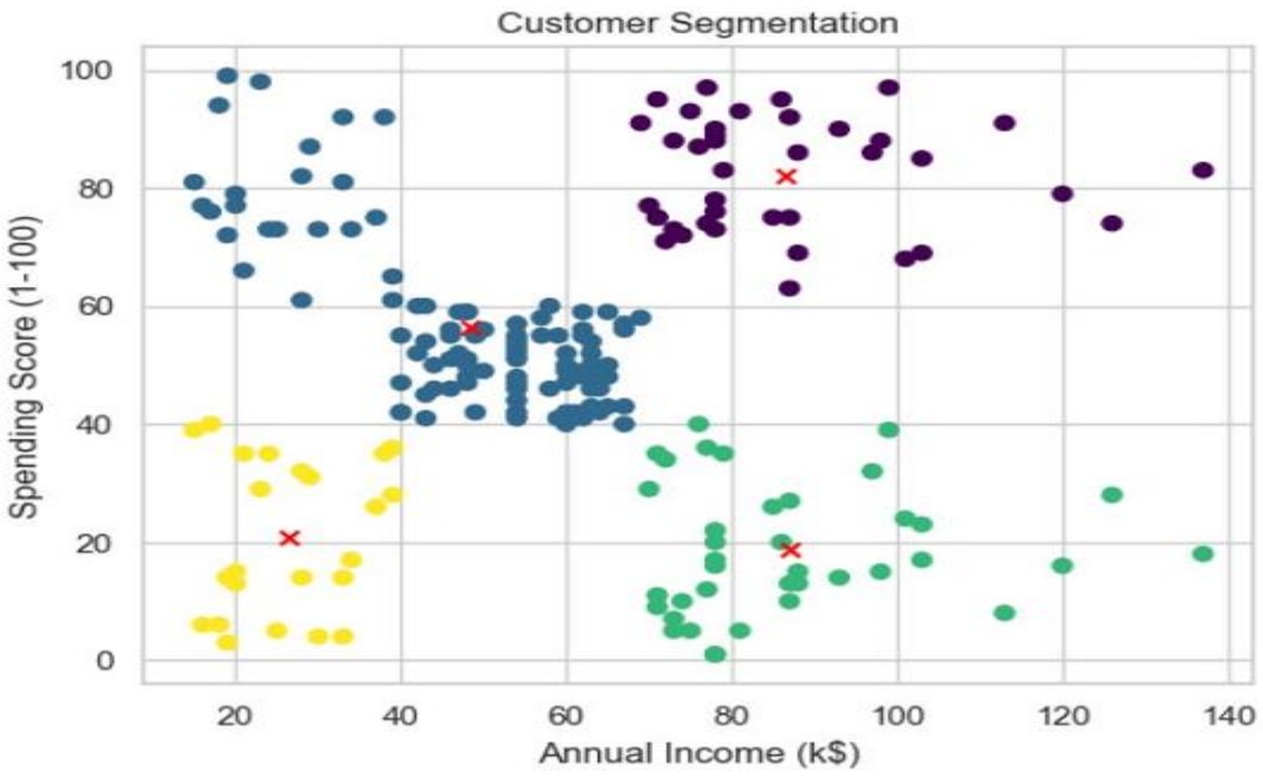


```
In [22]: # Print the cluster centers
         cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)
         print('Cluster Centers:')
         for i in range(k):
             print(f'Cluster {i+1}: {cluster_centers[i]}')
```

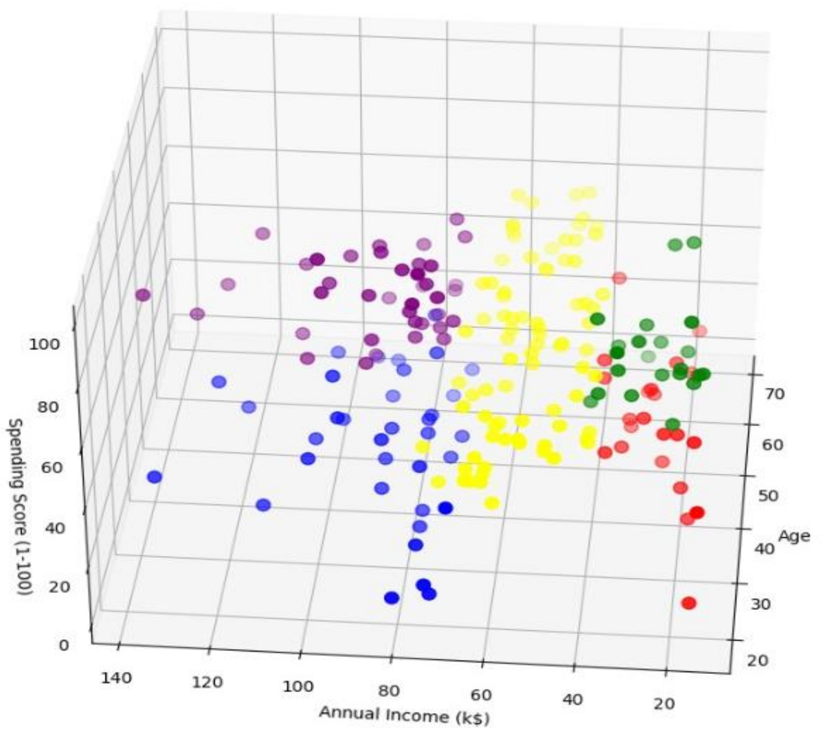
```
Cluster Centers:
Cluster 1: [86.53846154 82.12820513]
Cluster 2: [48.26 56.48]
Cluster 3: [87.         18.63157895]
Cluster 4: [26.30434783 20.91304348]
```



```
In [23]: # Visualize the clusters
         plt.scatter(X['Annual Income (k$)'], X['Spending Score (1-100)'], c=kmeans.labels_, cmap='viridis')
         plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], marker='x', color='red')
         plt.xlabel('Annual Income (k$)')
         plt.ylabel('Spending Score (1-100)')
         plt.title('Customer Segmentation')
         plt.show()
```



3D Plot Visualization



4.2.9 Results

The K-Means clustering algorithm successfully segmented the customer data into four distinct clusters based on annual income and spending score. The optimal number of clusters was determined using the elbow method, and the clusters revealed clear patterns in customer behavior. The visual representation confirmed that the algorithm had effectively categorized customers into well-defined groups, such as low-income, low-spending, and high-income, high-spending.

4.2.10 Conclusion

The results of the K-Means clustering provide valuable insights into customer segmentation, enabling businesses to target different customer groups with tailored marketing strategies. By analyzing the clustering output, companies can identify specific segments and better meet their needs. The approach was effective in revealing actionable patterns, despite challenges like determining the number of clusters.

CHAPTER V

SYSTEM DESIGN

SYSTEM DESIGN

System Architecture

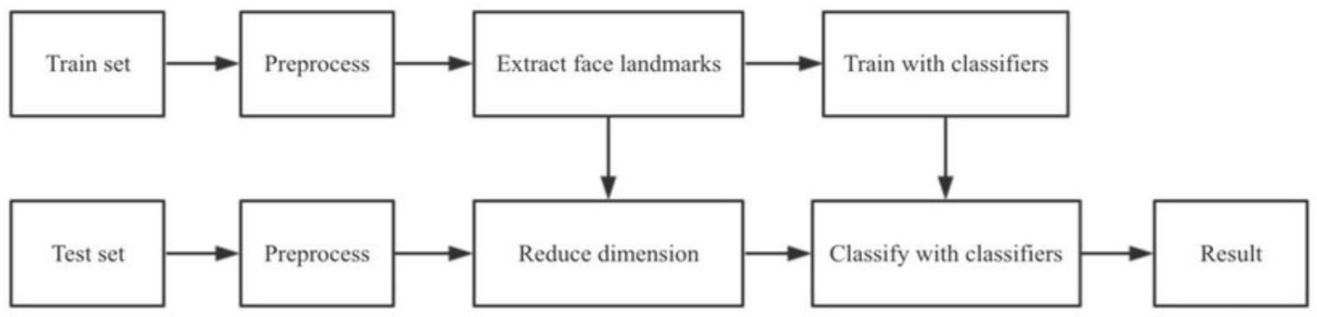


Figure: System architecture of a machine learning algorithm and how it flows

The machine learning architecture defines the various layers involved in the machine learning cycle and involves the major steps being carried out in the transformation of raw data into training data sets capable for enabling the decision making of a system.

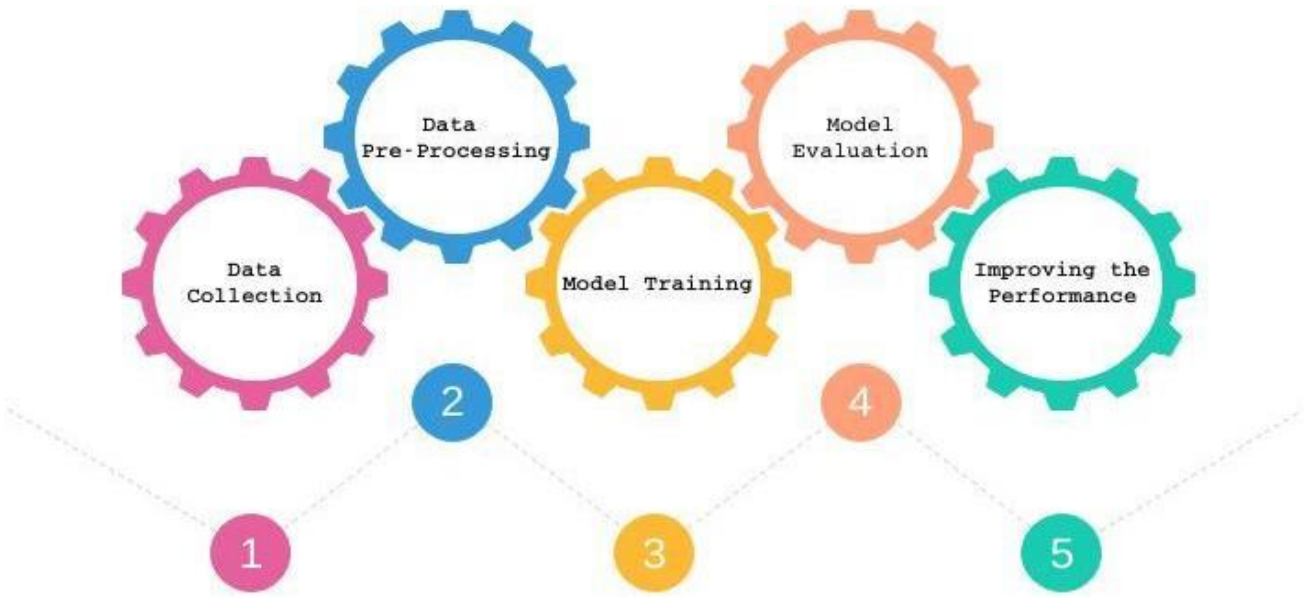
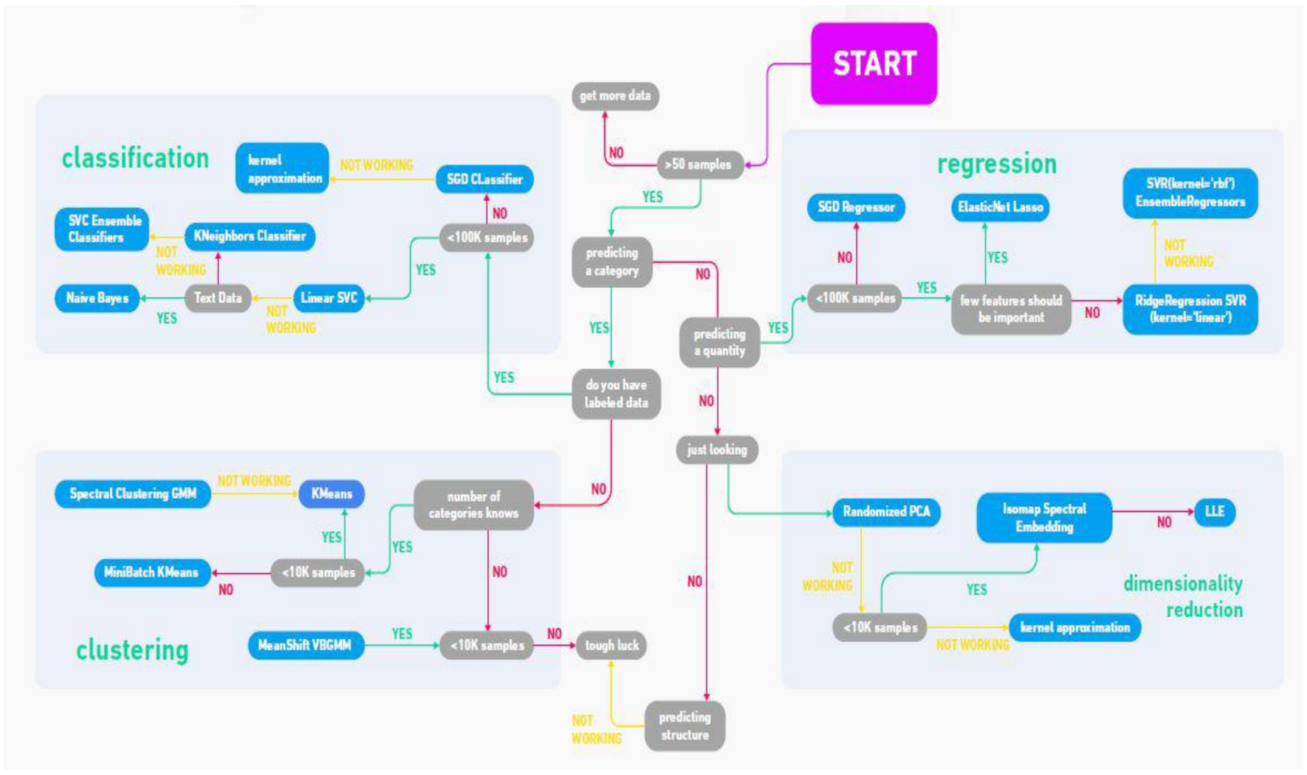


Figure: The dataflow diagram of a how a machine learning algorithm works

Flow chart


START
get more data
NO
>50 samples
YES
predicting a category
NO
predicting a quantity
YES
do you have labeled data
YES
NO
just looking
predicting structure
NOT WORKING
tough luck
classification
kernel approximation
NOT WORKING
SGD CLassifier
NO
<100K samples
YES
SVC Ensemble Classifiers
KNeighbors Classifier
NOT WORKING
Naive Bayes
Text Data
Linear SVC
YES
NOT WORKING
regression
SGD Regressor
ElasticNet Lasso
SVR(kernel='rbf') EnsembleRegressors
NO
YES
NOT WORKING
<100K samples
YES
few features should be important
NO
RidgeRegression SVR (kernel='linear')
clustering
Spectral Clustering GMM
NOT WORKING
KMeans
number of categories knows
YES
YES
MiniBatch KMeans
<10K samples
NO
NO
MeanShift VBGMM
YES
<10K samples
NO
Randomized PCA
Isomap Spectral Embedding
NO
LLE
NOT WORKING
YES
dimensionality reduction
<10K samples
kernel approximation
NOT WORKING

55

CHAPTER VI

CONCLUSION

CONCLUSION

In this internship, I had the opportunity to work on two distinct projects that involved applying machine learning techniques to real-world datasets. Both projects provided valuable insights into the practical application of data science, while also presenting unique challenges that enhanced my problem-solving abilities.

The mini project focused on classifying the Iris dataset using the K-Nearest Neighbors (KNN) algorithm. The project was successful in terms of achieving a high classification **accuracy of 96.67%**. By pre-processing the dataset and using KNN, we were able to accurately classify the flowers into their respective species based on features like sepal and petal length. This project helped solidify my understanding of supervised learning and classification problems, while also allowing me to apply data pre-processing and model evaluation techniques. The challenges faced during this project, including data scaling and handling noisy data, were mitigated through appropriate preprocessing steps, such as feature scaling and careful selection of hyperparameters. Overall, the project demonstrated the power of simple machine learning algorithms, such as KNN, when applied to well-structured datasets.

The major project focused on customer segmentation using the K-Means clustering algorithm. This project aimed to categorize customers based on their annual income and spending scores, a key application of unsupervised learning in marketing and business strategy. By applying the elbow method, we determined that **four clusters were optimal for this dataset**, which was further validated through visualization. The customer segments **revealed distinct patterns**, which could be utilized for targeted marketing and strategic decision-making. Despite challenges such as selecting the right features for clustering and deciding on the optimal number of clusters, the project successfully highlighted the importance of data exploration and the ability to derive actionable insights from raw data. This project was particularly valuable in understanding the nuances of unsupervised learning and the practical difficulties of clustering real-world data.

Both projects allowed me to apply theoretical machine learning concepts to real-world datasets, bridging the gap between classroom learning and practical experience. The Iris classification project reinforced my understanding of supervised learning, while the customer segmentation project deepened my knowledge of unsupervised learning techniques. These experiences provided me with a comprehensive understanding of the strengths and limitations of different machine learning algorithms, as well as the importance of proper data pre-processing and evaluation metrics. Through both projects, I gained a deeper appreciation for the intricacies of machine learning and its ability to uncover valuable insights from data. Moving forward, I am confident that the skills I have acquired will be valuable in solving complex data-driven problems in the field of data science and machine learning.

CHAPTER VII

FUTURE ENHANCEMENTS

FUTURE ENHANCEMENTS

While both the mini project on Iris classification and the major project on customer segmentation were successful in achieving their respective objectives, there are several opportunities for future enhancements to improve the accuracy, efficiency, and applicability of the models.

Minor Project Enhancements

- **Advanced Feature Engineering**: In the current project, we used basic features like sepal and petal length and width for classification. In future iterations, additional feature engineering could be explored by incorporating other attributes such as flower texture or environmental factors that may contribute to more accurate predictions.
- **Model Optimization**: Although KNN performed well in terms of accuracy, exploring alternative classification models such as Support Vector Machines (SVM), Random Forest, or Gradient Boosting could lead to even better performance. Hyperparameter tuning techniques, such as grid search or random search, could also be used to find the optimal parameters for the KNN algorithm.

Major Project Enhancements

- **Incorporating More Features**: In the current customer segmentation project, we only considered annual income and spending score. To further enhance the model, additional features such as customer age, purchase history, or social media activity could be included to create more granular and meaningful customer segments.
- **Real-time Segmentation**: A future enhancement could involve implementing real-time segmentation by continuously updating the customer segments based on new data. This would enable businesses to adapt quickly to changing customer behaviours and market conditions.

- **Other Clustering Techniques**: While K-Means is a popular and effective clustering algorithm, there are several alternatives, such as DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and hierarchical clustering, which might perform better for datasets with complex structures or noise. Testing these algorithms could help in finding more robust clustering solutions.

General Enhancements for Both Projects

- **Automated Data Preprocessing**: Developing an automated pipeline for data cleaning and pre-processing would streamline the process of handling raw datasets, ensuring that the data is ready for analysis with minimal manual intervention.
- **Deployment and Monitoring**: Both projects could be enhanced by deploying the models into production systems. This would allow businesses to make real-time predictions or segmentations based on incoming data. Additionally, monitoring the model's performance over time would help detect any drift in the data or the model's predictive capabilities.
- **Explainable AI**: As machine learning models become increasingly complex, the need for interpretability and transparency grows. Implementing techniques for model explainability, such as LIME (Local Interpretable Model-agnostic Explanations) or SHAP (Shapley Additive Explanations), could help stakeholders better understand the decision-making process behind the models.

CHAPTER VIII

REFERENCES

REFERENCES

[1] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.

[2] Jain, A. K. (2010). *Data Clustering: 50 Years Beyond K-Means*. Pattern Recognition Letters, 31(8), 651–666.

[3] Dhar, V. (2013). "Data science and prediction". *Communications of the ACM*. **56** (12): 64–73.

[4] Bishop, C. M. (2006), *Pattern Recognition and Machine Learning*, Springer, ISBN 978-0-387-31073-2

[5] Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, O'Reilly Publications, 2019.

[6] Data Science and Machine Learning Mathematical and Statistical Methods, Dirk P. Kroese, Zdravko I. Botev, Thomas Taimre, Radislav Vaisman.

[7] Python Docs: 3.13.1 Documentation

[8] Pandas Docs: pandas documentation — pandas 2.2.3 documentation

[9] Matplotlib Docs: Matplotlib documentation — Matplotlib 3.10.0 documentation

[10] Jupyter Docs: Project Jupyter Documentation — Jupyter Documentation 4.1.1 alpha documentation

[11] Bandaru, D. S. S. G. (2025). *Codebase for HDLC Technologies Internship Projects.* GitHub Repository. Available at: https://github.com/deva-04/HDLC-Technologies-Internship