

A Project On
Development of Graphical User Interface
using tkinter in Python

A Project Report Submitted by Partial Fulfillment of Requirements
for the Degree of M.Sc(Statistics)
with Specialisation in Industrial Statistics

Submitted by

Mr. Chaudhari Jayesh Suresh (Seat No. 387407)

Mr. Nikam Vikrant Dipak (Seat No. 387424)

Ms. Patil Vaishnavi Sharad (Seat No. 387440)

Under the Guidance of
Asst. Prof. M. C. PATIL

Submitted to



DEPARTMENT OF STATISTICS
SCHOOL OF MATHEMATICAL SCIENCES
KAVAYITRI BAHINABAI CHAUDHARI
NORTH MAHARASHTRA UNIVERSITY, JALGAON - 425001
YEAR 2022-2023

**DEPARTMENT OF STATISTICS
KAVAYITRI BAHINABAI CHAUDHARI
NORTH MAHARASHTRA UNIVERSITY, JALGAON.**



CERTIFICATE

This is to certify that **Mr. Chaudhari Jayesh Suresh (Seat No. 387407)**, **Mr. Nikam Vikrant Dipak (Seat No. 387424)** and **Ms. Patil Vaishnavi Sharad (Seat No. 387440)** are the students of M.Sc.(Statistics) with specialization in Industrial Statistics at Kavayitri Bahinabai Chaudhari North Maharashtra University, Jalgaon and have successfully completed their project entitled “**Development of Graphical User Interface using tkinter in Python**”, under my guidance and supervision during the academic year 2022-2023.

Place :- Jalgaon

Date :-

Prof. Manoj C Patil

(Project Guide)

Department Of Statistics

Kavayitri Bahinabai Chaudhari

North Maharashtra University,

Jalgaon.

DECLARATION

We Chaudhari Jayesh Suresh, Nikam Vikrant Dipak, Patil Vaishnavi Sharad hereby declare that the project report **Development of Graphical User Interface using tkinter in Python**, submitted for partial fulfillment of the requirements for the award of degree of Master of Science in Statistics of the Kavayitri Bahinabai Chaudhari North Maharashtra University, Jalgaon, Maharashtra is a bonafide work done by us under supervision of Prof. Manoj Patil, Department Of Statistics, KBC NMU, Jalgaon.

This submission represents our ideas in our own words and where ideas or words of others have been included, We have adequately and accurately cited and referenced the original sources.

We also declare that We have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Mr. Chaudhari Jayesh Suresh (Seat No. 387407)

Mr. Nikam Vikrant Dipak (Seat No. 387424)

Ms. Patil Vaishnavi Sharad (Seat No. 387440)

Acknowledgement

We are thankful to Prof. R. L. Shinde, Head of Department of Statistics, Kavayitri Bahinabai Chaudhari North Maharashtra University, Jalgaon and project guide Prof. M. C. Patil, Department of Statistics for their valuable guidance, suggestions, cooperation and constant encouragement which enable us to take every forward step in our project.

We would like extend our gratitude to Prof. K. K. Kamalja, Department of Statistics and Prof. R. D. Koshti, Department of Statistics who help us time to time during the project. We would also like to thanks specially Mr. Mukund Kulkarni Sir, Senior Manager at Bajaj Allianz Life Insurance Co Limited, Pune for suggesting the topic and his precious time.

Our classmates also deserve delightful thanks for their continuous encouragement to us. Finally, we thank our parents for the moral supports and blessing.

Mr. Chaudhari Jayesh Suresh (Seat No. 387407)

Mr. Nikam Vikrant Dipak (Seat No. 387424)

Ms. Patil Vaishnavi Sharad (Seat No. 387440)

Contents

Acknowledgement	i
1 Introduction	1
1.1 Motivation	1
1.2 Objective	1
1.3 What is GUI?	2
2 Python GUI - tkinter	3
2.1 Standard Attributes	9
2.2 Geometry Management	10
3 Introduction to Machine Learning	11
3.1 What is Machine Learning?	11
3.1.1 Supervised Learning	11
3.1.2 UnSupervised Learning	12
4 Classification Models	13
4.1 Decision Tree:	13
4.1.1 Introduction :	13
4.1.2 Advantages	14
4.1.3 Limitations	14
4.1.4 Applications	14
4.2 K - Nearest Neighbourhood	15
4.2.1 Introduction :	15
4.2.2 Advantages	15
4.2.3 Limitations	15
4.2.4 Applications	16
4.3 Support Vector Classifier	16
4.3.1 Introduction :	16
4.3.2 Advantages	17
4.3.3 Limitations	17
4.3.4 Applications	17
4.4 Naive Bayes Classifier	18
4.4.1 Introduction :	18
4.4.2 Advantages	18
4.4.3 Limitations	19
4.5 Logistic Regression	19
4.5.1 Introduction :	19
4.5.2 Advantages	20
4.5.3 Limitations	20
4.5.4 Applications	21

5	Regression Models	22
5.1	Decision Tree Regression	22
5.1.1	Introduction :	22
5.1.2	Advantages	23
5.1.3	Limitations	23
5.1.4	Applications	24
5.2	Lasso Regression	24
5.2.1	Introduction :	24
5.2.2	Advantages	25
5.2.3	Limitations	25
5.2.4	Applications	26
5.3	Linear Regression :	26
5.3.1	Advantages	27
5.3.2	Limitations	27
5.3.3	Applications	28
5.4	Ridge Regression	28
5.4.1	Introduction :	28
5.4.2	Advantages	29
5.4.3	Limitations	29
5.4.4	Applications	29
5.5	Polynomial Regression	30
5.5.1	Introduction :	30
5.5.2	Advantages	30
5.5.3	Limitations	31
5.5.4	Applications	31
5.6	Support Vector Regression	31
5.6.1	Introduction :	31
5.6.2	Advantages	32
5.6.3	Limitations	32
5.6.4	Applications	32
6	Usage and Demonstration	33
6.1	Steps to Use GUI	33
6.2	Results on Datasets	36
6.2.1	Decision Tree Classification	36
6.2.2	Support Vector Classification	38
6.2.3	K - Nearest Neighbourhood	41
6.2.4	Naive Bayes Classifier	45
6.2.5	Linear Regression	47
6.2.6	Lasso Regression	51
6.2.7	Support Vector Regression	54
6.2.8	Polynomial Regression	57
	References	60
	Annexure	61

Chapter 1

Introduction

1.1 Motivation

The motivation behind this project is to develop a user-friendly graphical interface for machine learning using the Python programming language. Machine learning has revolutionized various industries, enabling automated decision-making and predictive analysis. However, its implementation often requires a deep understanding of programming and complex algorithms, posing a barrier for non-technical users.

By creating a graphical interface, we aim to democratize access to machine learning tools and techniques. Our interface will allow users with limited programming knowledge to interact with and utilize machine learning algorithms effortlessly. This will open up new possibilities for individuals and organizations to harness the power of data-driven insights without the need for extensive coding expertise.

The graphical interface will provide an intuitive and visually appealing environment, enabling users to easily load datasets, select algorithms, tune parameters, and visualize results. The aim is to bridge the gap between machine learning experts and domain specialists, enabling collaboration and knowledge sharing.

Overall, this project seeks to empower a wider audience to leverage the potential of machine learning, fostering innovation and problem-solving across diverse fields such as healthcare, finance, and marketing.

1.2 Objective

The objective of this project is to develop a graphical user interface (GUI) for machine learning using the Python programming language and the tkinter library. The primary goal is to provide a user-friendly platform that simplifies the process of implementing and experimenting with machine learning algorithms.

The specific objectives include :

1. Designing an intuitive and visually appealing GUI that enables users to interact with machine learning functionalities without the need for extensive coding knowledge.
2. Integrating popular machine learning libraries such as scikit-learn into the GUI, allowing users to access a wide range of algorithms and models.

3. Implementing data preprocessing capabilities within the interface, allowing users to clean, transform, and preprocess datasets easily.
4. Enabling users to train, evaluate, and fine-tune machine learning models by providing options for algorithm selection, parameter tuning, and cross-validation.
5. Visualizing and presenting the results of machine learning experiments through interactive charts, graphs, and metrics, aiding in the interpretation and analysis of the outcomes.

By achieving these objectives, we aim to democratize machine learning by making it more accessible to users with limited programming experience, thereby facilitating innovation and problem-solving in various domains.

1.3 What is GUI?

GUI stands for Graphical User Interface. It refers to a visual interface that allows users to interact with a computer system using graphical elements such as windows, icons, buttons, and menus, rather than relying solely on text-based commands. GUIs are designed to provide a more intuitive and user-friendly experience, enabling users to perform tasks and access information through visual representations.

GUIs enable users to interact with software applications by simply clicking on buttons, dragging and dropping items, selecting options from menus, and visually navigating through different screens or windows. They provide a visual representation of the underlying functionality and data, making it easier for users to understand and control the software.

GUIs have become an integral part of modern operating systems and software applications across various domains. They have revolutionized the way users interact with computers, enabling individuals with varying levels of technical expertise to access and utilize computer systems effectively. GUIs have played a crucial role in making technology more accessible, enhancing productivity, and improving the overall user experience.

Chapter 2

Python GUI - tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI using tkinter is an easy task. All you need to do is perform the following steps :-

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the below-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

```
#!/usr/bin/python
import tkinter
top = tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

This would create a following window :

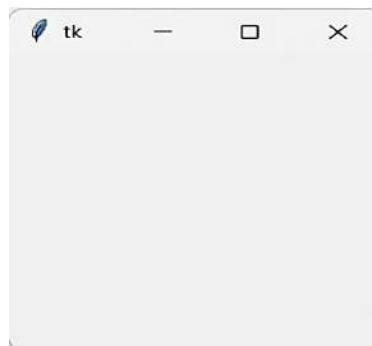


Figure 2.1: Tkinter Mainwindow

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

Button

The Button widget is used to display buttons in your application.



Figure 2.2: Tkinter Button

Canvas

The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.

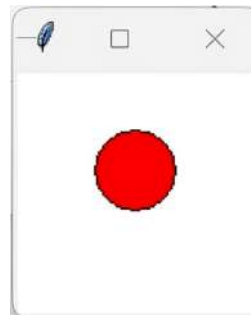


Figure 2.3: Tkinter Canvas

Checkbutton

The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.

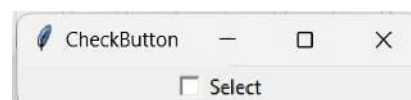


Figure 2.4: Tkinter Checkbutton

Entry

The Entry widget is used to display a single-line text field for accepting values from a user.

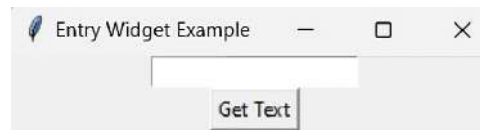


Figure 2.5: Tkinter Entry

Frame

The Frame widget is used as a container widget to organize other widgets.



Figure 2.6: Tkinter Frame

Label

The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

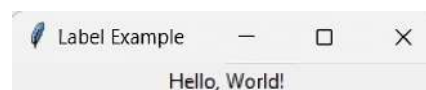


Figure 2.7: Tkinter Label

Listbox

The Listbox widget is used to provide a list of options to a user.

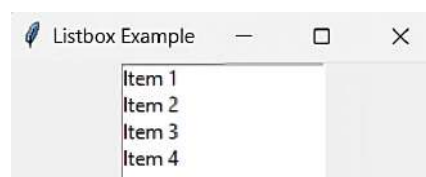


Figure 2.8: Tkinter Listbox

Menubutton

The Menubutton widget is used to display menus in your application.

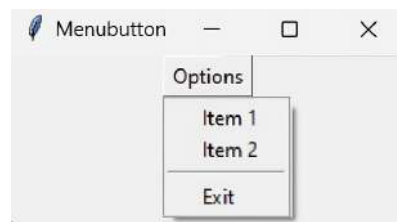


Figure 2.9: Tkinter Menubutton

Menu

The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.

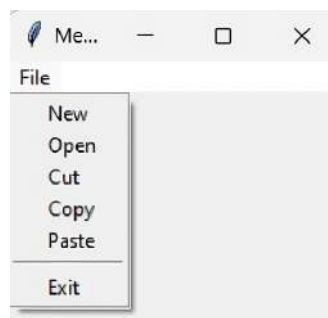


Figure 2.10: Tkinter Menu

Message

The Message widget is used to display multiline text fields for accepting values from a user.

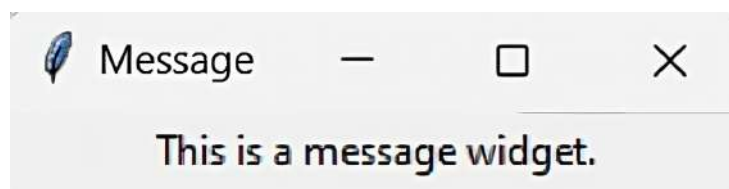


Figure 2.11: Tkinter Message

Radiobutton

The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.

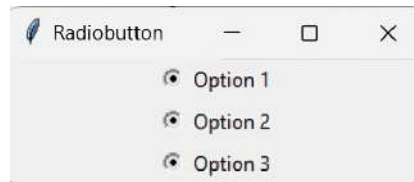


Figure 2.12: Tkinter RadioButton

Scale

The Scale widget is used to provide a slider widget.

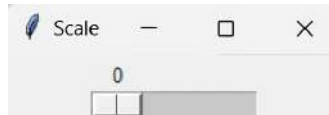


Figure 2.13: Tkinter Scale

Scrollbar

The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.

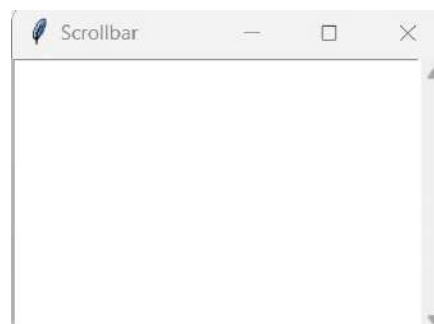


Figure 2.14: Tkinter Scrollbar

Text

The Text widget is used to display text in multiple lines.

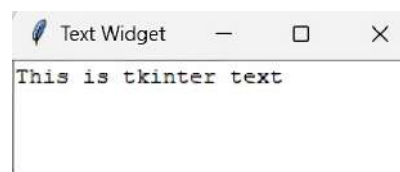


Figure 2.15: Tkinter Text

Toplevel

The Toplevel widget is used to provide a separate window container.

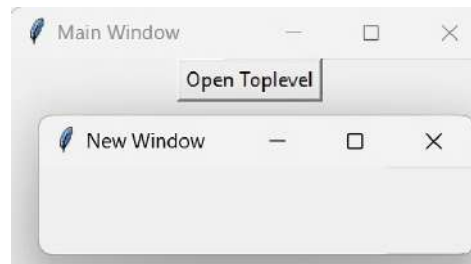


Figure 2.16: Tkinter Toplevel

Spinbox

The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.



Figure 2.17: Tkinter Spinbox

PanedWindow

A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.



Figure 2.18: Tkinter Panedwindow

LabelFrame

A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.



Figure 2.19: Tkinter Panedwindow

tkMessageBox

This module is used to display message boxes in your applications.

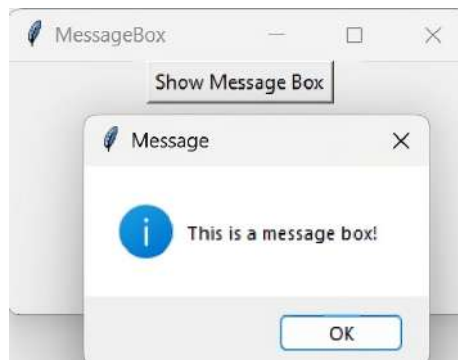


Figure 2.20: Tkinter Messagebox

2.1 Standard Attributes

Let us take a look at how some of their common attributes, such as sizes, colors and fonts are specified.

1. Dimensions
2. Colors
3. Fonts
4. Anchors
5. Relief styles
6. Bitmaps
7. Cursors

2.2 Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- The `.pack()` Method :- This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The `.grid()` Method :- This geometry manager organizes widgets in a tablelike structure in the parent widget.
- The `.place()` Method :- This geometry manager organizes widgets by placing them in a specific position in the parent widget.

Chapter 3

Introduction to Machine Learning

3.1 What is Machine Learning?

Machine learning is a subfield of artificial intelligence that focuses on developing algorithms and models that allow computers to learn from data and make predictions or decisions without being explicitly programmed. It is based on the idea that machines can analyze and interpret patterns in data to extract meaningful insights and improve their performance over time.

The process of machine learning typically involves several key steps. First, a dataset is collected, which consists of input data and corresponding output or target values. This data is then used to train a machine learning model, where the model learns the underlying patterns and relationships between the input and output variables. The model adjusts its parameters and structure based on the provided training data, with the goal of minimizing errors and accurately predicting outcomes.

Machine learning algorithms can be categorized into various types, such as supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the model learns from labeled data, where the inputs and their corresponding outputs are known. Unsupervised learning, on the other hand, deals with unlabeled data and aims to discover patterns or structures within the data. Reinforcement learning involves an agent interacting with an environment, learning through trial and error to maximize a reward signal.

Machine learning has numerous applications across different domains. It is used in image and speech recognition, natural language processing, recommendation systems, fraud detection, predictive analytics, autonomous vehicles, and many other areas. The ability of machine learning models to handle vast amounts of data and detect complex patterns has made them invaluable for solving real-world problems and driving innovation.

Overall, machine learning plays a crucial role in transforming industries, enabling computers to learn from data and make intelligent decisions, and paving the way for advancements in artificial intelligence.

3.1.1 Supervised Learning

Supervised learning is a type of machine learning where the algorithm learns from labeled data to make predictions or classifications on new, unseen data. In supervised learning, the data is labeled with the correct output, which is used to train the algorithm.

Supervised learning can be further divided into two main categories : classification and regression.

Classification is a supervised learning task where the algorithm learns to predict a discrete class label for new data points based on the features of the training data. Examples of classification tasks include email spam detection, image recognition, and sentiment analysis. The output of a classification model is a categorical variable, which represents the class label of the new data point.

Regression is a supervised learning task where the algorithm learns to predict a continuous numerical value for new data points based on the features of the training data. Examples of regression tasks include predicting housing prices, stock prices, and weather forecasts. The output of a regression model is a numerical variable, which represents the predicted value of the new data point.

Both classification and regression tasks involve learning from labeled data and building a model that can make accurate predictions on new, unseen data. The performance of the model can be evaluated using various metrics such as accuracy, precision, recall, and mean squared error. The choice of the appropriate supervised learning technique depends on the nature of the problem and the type of data available.

3.1.2 UnSupervised Learning

Unsupervised learning is a type of machine learning where the algorithm learns from unlabeled data to discover hidden patterns or groupings. Unlike supervised learning, unsupervised learning does not have labeled data or a predefined output variable.

Clustering is a common unsupervised learning task where the algorithm learns to group similar data points together based on the features of the data. Examples of clustering tasks include customer segmentation, image segmentation, and anomaly detection. The output of a clustering model is a set of clusters, where each cluster represents a group of similar data points.

Dimensionality reduction is another unsupervised learning task where the algorithm learns to reduce the number of features or variables in the data while preserving the important information. Examples of dimensionality reduction techniques include Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE).

Unsupervised learning is used in various fields such as data mining, image processing, and natural language processing. The performance of unsupervised learning algorithms is evaluated using metrics such as clustering accuracy, silhouette score, and reconstruction error.

The choice of the appropriate unsupervised learning technique depends on the nature of the problem and the type of data available. Unsupervised learning techniques can also be used in combination with supervised learning techniques for better performance.

Chapter 4

Classification Models

4.1 Decision Tree:

4.1.1 Introduction :

Decision Trees are a popular supervised machine learning algorithm used for both classification and regression tasks. They create a tree-like model of decisions and their possible consequences, allowing for the exploration of multiple decision paths and capturing complex relationships within the data.

The Decision Tree model works as follows :

- **Attribute Selection :** The algorithm selects the best attribute (feature) from the dataset as the root node of the tree based on a selected criterion (e.g., information gain, Gini index).
- **Node Splitting :** The dataset is split into subsets based on the chosen attribute value, creating branches or child nodes.
- **Recursion :** The above steps are recursively applied to each child node until a stopping criterion is met. This can be based on a maximum depth limit, a minimum number of samples per leaf, or other conditions.
- **Leaf Node Assignment :** At the end of the recursion, to assign an observation in a given leaf node to the most commonly occurring class. The classification error rate is simply the fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - \max_k(\hat{p}_{mk}) \quad (4.1)$$

Here \hat{p}_{mk} represents the proportion of training observations in the m^{th} leaf that are from the k^{th} class. However, it turns out that classification error is not sufficiently sensitive for tree growing, and in practice two other measures are preferable. The Gini index is defined by

$$G = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk}) \quad (4.2)$$

a measure of total variance across the K classes. An alternative to the Gini index is entropy, given by

$$D = - \sum_{k=1}^K \hat{p}_{mk} \log(\hat{p}_{mk}) \quad (4.3)$$

- **Prediction :** Given a new data point, it traverses the decision tree based on the attribute values, following the decision path until reaching a leaf node, and then outputs the corresponding class label or predicted value.

4.1.2 Advantages

- **Interpretability :** Decision Trees provide human-interpretable rules, making it easy to understand and explain the decision-making process.
- **Nonlinear Relationships :** Decision Trees can handle nonlinear relationships between features and the target variable.
- **Feature Importance :** Decision Trees can rank the importance of features based on their contribution to the decision-making process.
- **Handling Missing Data :** Decision Trees can handle missing values without the need for data imputation techniques.
- **Robustness :** Decision Trees are less affected by outliers or noise in the data compared to some other algorithms.

4.1.3 Limitations

- **Overfitting :** Decision Trees are prone to overfitting, especially when the tree becomes deep or when there are noisy or irrelevant features.
- **Lack of Smoothness :** Decision Trees create step-like boundaries, which may lead to unstable predictions in regions with slight changes in input values.
- **Lack of Generalization :** Decision Trees may not generalize well to unseen data if they have learned specific patterns or outliers in the training set.
- **Bias towards Features with More Levels :** Decision Trees tend to favor attributes with more levels or categories when selecting the best split, potentially neglecting other relevant features.
- **Scalability :** Decision Trees can grow large and complex, making them computationally expensive and challenging to handle with massive datasets.

4.1.4 Applications

- **Classification :** Decision Trees are widely used for classification tasks, such as spam detection, sentiment analysis, or medical diagnosis.
- **Regression :** Decision Trees can also be applied to regression problems, such as predicting house prices or stock market trends.
- **Feature Selection :** Decision Trees can assist in identifying the most informative features for subsequent modeling or analysis.
- **Anomaly Detection :** Decision Trees can be employed for detecting outliers or anomalies in datasets.
- **Recommender Systems :** Decision Trees can be utilized to build recommendation systems, suggesting items or services based on user preferences.

4.2 K - Nearest Neighbourhood

4.2.1 Introduction :

The K-Nearest Neighbors (KNN) classifier is a popular supervised machine learning algorithm used for classification tasks. It is a non-parametric algorithm that makes predictions based on the similarity of the input data to its k nearest neighbors in the feature space. KNN is considered an instance-based learning method, as it doesn't explicitly learn a model but instead stores the training data to make predictions.

The KNN classifier works based on the following steps :

- **Training :** The algorithm stores the labeled training data in memory, preserving the feature vectors and corresponding class labels.
- **Distance Calculation :** When a new, unlabeled data point is provided for classification, the KNN classifier calculates the distance (e.g., Euclidean distance) between the new data point and each training data point in the feature space.
- **Neighbor Selection :** The KNN classifier selects the k nearest neighbors based on the calculated distances.
- **Voting or Weighting :** For classification, the class labels of the k nearest neighbors are considered. The KNN classifier either assigns the class label by majority voting or weights the labels based on the neighbors' distances.
- **Prediction :** The KNN classifier assigns the class label of the new data point based on the selected neighbors' labels.

4.2.2 Advantages

- **Simplicity and Ease of Implementation :** KNN is relatively simple to understand and implement, making it accessible for beginners.
- **No Training Phase :** The KNN classifier doesn't require an explicit training phase, as it directly uses the labeled training data for predictions.
- **Non-Parametric :** KNN makes no assumptions about the underlying data distribution, making it suitable for a wide range of applications.
- **Adaptability to New Data :** KNN can easily adapt to new data points without the need for retraining the model.
- **Interpretable :** The KNN classifier provides interpretability, as the predicted class is based on the labels of the nearest neighbors.

4.2.3 Limitations

- **Computational Complexity :** As the number of training samples increases, the computation time and memory requirements of KNN can become significant.
- **Sensitivity to Feature Scaling :** KNN is sensitive to the scale and range of features, so data normalization or scaling may be necessary.
- **Curse of Dimensionality :** KNN performance can deteriorate in high-dimensional feature spaces, as the notion of distance becomes less informative.

- **Choosing an Appropriate K Value :** The selection of the optimal k value can impact the model's performance, and it may require experimentation or tuning.
- **Imbalanced Data :** KNN can be biased towards the majority class in imbalanced datasets, leading to less accurate predictions for minority classes.

4.2.4 Applications

- **Text Categorization :** KNN can be used for tasks like sentiment analysis, document classification, and spam filtering based on text features.
- **Image Recognition :** KNN can classify images by comparing their feature vectors or using techniques like k-d trees for efficient nearest neighbor search.
- **Recommendation Systems :** KNN can be applied in collaborative filtering for generating personalized recommendations based on similar users or items.
- **Healthcare :** KNN can assist in disease diagnosis or predicting patient outcomes based on similar cases in the dataset.
- **Anomaly Detection :** KNN can identify anomalies by identifying data points that are dissimilar to their neighbors.

4.3 Support Vector Classifier

4.3.1 Introduction :

The Support Vector Classifier (SVC), also known as Support Vector Machine (SVM) for binary classification, is a powerful supervised machine learning algorithm. It aims to find an optimal hyperplane that separates different classes in the feature space with the largest margin. SVC is widely used due to its ability to handle complex datasets and generalize well to unseen data.

The SVC model works as follows :

- **Training Phase :** The SVC algorithm takes labeled training data and maps the input features to a high-dimensional feature space.
- **Margin Maximization :** SVC identifies the optimal hyperplane that maximizes the margin between the classes. The margin is the distance between the hyperplane and the nearest training data points from each class (known as support vectors).
- **Soft Margin Classification :** If the classes are not perfectly separable, SVC allows for soft margin classification, where a penalty is introduced for misclassifications or overlapping instances.
- **Optimization Problem :** SVC Separate most of the training observations into the two classes, but may misclassify a few observations. It is the solution to the optimization problem

$$\max_{\beta_0, \beta_1, \dots, \beta_p, \epsilon_1, \dots, \epsilon_n, M} M \quad (4.4)$$

subject to

$$\sum_{j=1}^p \beta_j^2 = 1, \quad (4.5)$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \quad (4.6)$$

$$\epsilon_i \geq 0, \sum_{i=1}^n \epsilon_i \leq C, \quad (4.7)$$

where C is a non negative tuning parameter. M is the width of the margin; we seek to make this quantity as large as possible. $\epsilon_1 \dots \epsilon_n$ are slack variables that allow individual observations to be on slack the wrong side of the margin or the hyperplane.

- **Nonlinear Transformation :** SVC can employ the kernel trick to transform the data into a higher-dimensional space, enabling the separation of nonlinearly separable classes.
- **Prediction :** Once the hyperplane is determined, SVC uses it to classify new, unlabeled data points based on their position relative to the hyperplane.

4.3.2 Advantages

- **Effective in High-Dimensional Spaces :** SVC performs well in datasets with a large number of features, such as text classification or image recognition tasks.
- **Robust to Overfitting :** SVC handles overfitting by maximizing the margin, which promotes better generalization to unseen data.
- **Versatility with Kernels :** SVC can employ various kernel functions (e.g., linear, polynomial, radial basis function) to handle nonlinear relationships between features.
- **Support for Different Data Types :** SVC can handle numerical as well as categorical data through appropriate kernel functions.
- **Strong Theoretical Foundation :** SVC is supported by a solid mathematical framework, providing theoretical guarantees for its performance.

4.3.3 Limitations

- **Computationally Intensive :** SVC can be computationally expensive, particularly for large datasets, as it requires solving a quadratic optimization problem.
- **Sensitivity to Hyperparameters :** SVC's performance can be sensitive to the choice of hyperparameters, such as the kernel type and regularization parameter.
- **Interpretability :** SVC models can be challenging to interpret and explain due to the complex decision boundaries in high-dimensional spaces.
- **Memory Usage :** SVC requires storing support vectors and associated parameters, which can consume significant memory resources.
- **Imbalanced Data :** SVC may struggle with imbalanced datasets, where the majority class dominates the training process.

4.3.4 Applications

- **Text Classification :** SVC is commonly used in tasks like sentiment analysis, spam detection, and document categorization based on textual features.

- **Image Recognition** : SVC can classify images based on visual features, making it suitable for applications such as object detection or facial recognition.
- **Bioinformatics** : SVC has been applied to gene expression analysis, protein structure prediction, and disease diagnosis.
- **Finance** : SVC can assist in credit scoring, fraud detection, and stock market prediction by modeling complex relationships between variables.
- **Medical Diagnosis** : SVC has been used for disease classification, such as cancer detection, based on various medical imaging or genomic data.

4.4 Naive Bayes Classifier

4.4.1 Introduction :

The Naive Bayes classifier is a popular supervised machine learning algorithm used for classification tasks. It is based on the principle of Bayes' theorem and assumes that features are conditionally independent of each other given the class label. Despite its simple assumption, Naive Bayes has been proven to perform well in many real-world scenarios.

The Naive Bayes classifier works as follows :

- **Training Phase** : The algorithm analyzes the labeled training data and calculates the prior probabilities of each class label and the likelihood of each feature given the class label.
- **Feature Independence Assumption** : The Naive Bayes classifier assumes that all features are conditionally independent of each other given the class label. This assumption simplifies the calculation of the posterior probability.
- **Posterior Probability Calculation** : When a new, unlabeled data point is provided for classification, the Naive Bayes classifier calculates the posterior probability for each class label using Bayes' theorem. It combines the prior probabilities and the likelihoods of the features.
- **Bayes' Theorem** : Let y is class variable and X is a dependent feature vector (of size n) where: $X = (x_1, x_2, x_3, \dots, x_n)$. We can apply Bayes' theorem in following way:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \quad (4.8)$$

- **Class Prediction** : The Naive Bayes classifier assigns the class label with the highest posterior probability as the predicted class for the new data point.

4.4.2 Advantages

- **Simplicity and Speed** : Naive Bayes is a simple and computationally efficient algorithm, making it suitable for large datasets and real-time applications.
- **Scalability** : The Naive Bayes classifier scales well with the number of features and works efficiently even with high-dimensional data.
- **Good Performance with Small Training Data** : Naive Bayes performs well even with limited training data and can handle missing values effectively.

- **Robust to Irrelevant Features :** Naive Bayes is resilient to irrelevant features in the dataset, as it assumes feature independence.
- **Interpretable :** Naive Bayes provides interpretability, as the classification decision is based on probabilistic reasoning.

4.4.3 Limitations

- **Strong Feature Independence Assumption :** The assumption of feature independence may not hold in some real-world scenarios, leading to suboptimal predictions.
- **Sensitivity to Feature Correlations :** Naive Bayes may struggle when features are highly correlated, as it cannot capture complex relationships between features.
- **Lack of Model Calibration :** Naive Bayes tends to produce well-calibrated probability estimates, although the probabilities may not reflect the true probabilities in the data.
- **Limited Expressive Power :** Naive Bayes may not be suitable for tasks where complex interactions between features significantly affect the class probabilities.
- **Biased Output :** Naive Bayes tends to favor the majority class in imbalanced datasets, leading to biased predictions for minority classes.

Applications

- **Text Classification :** Naive Bayes is widely used in spam filtering, sentiment analysis, topic classification, and document categorization based on textual data.
- **Email Filtering :** Naive Bayes can be used for classifying emails as spam or legitimate based on email content and metadata.
- **Medical Diagnosis :** Naive Bayes has been applied in disease prediction and diagnosis based on patient symptoms, medical history, and test results.
- **Recommendation Systems :** Naive Bayes can be used in collaborative filtering to generate personalized recommendations for users based on their preferences.
- **Customer Segmentation :** Naive Bayes can assist in grouping customers into segments based on demographic, behavioral, or transactional data.

4.5 Logistic Regression

4.5.1 Introduction :

Logistic Regression is a widely used statistical model for binary classification. Despite its name, it is primarily used for classification tasks rather than regression. Logistic Regression estimates the probability of an event occurring based on input features and maps the output to a binary outcome.

The Logistic Regression model works as follows :

- **Hypothesis Function :** Logistic Regression uses a logistic or sigmoid function to model the relationship between the input features and the probability of the binary outcome.

The logistic function maps the linear combination of input features to a value between 0 and 1. The logistic regression model given by,

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} \quad (4.9)$$

- **Training Phase :** During training, the model learns the optimal coefficients (weights) for each feature through the process of maximum likelihood estimation. It aims to maximize the likelihood of observing the training data given the estimated probabilities.
- **Decision Boundary :** Logistic Regression calculates a decision boundary that separates the two classes based on the estimated probabilities. This boundary can be linear or nonlinear, depending on the complexity of the data.
- **Prediction :** For new, unlabeled data points, Logistic Regression calculates the probabilities using the learned coefficients and applies a threshold (usually 0.5) to assign the class label.

4.5.2 Advantages

- **Simplicity and Interpretability :** Logistic Regression is a straightforward and interpretable model that allows for easy understanding of the relationship between features and the probability of the outcome.
- **Efficiency :** Logistic Regression can handle large datasets efficiently and can be easily updated with new data.
- **Probability Estimation :** Logistic Regression provides predicted probabilities, allowing for more nuanced interpretation and decision-making.
- **Handles Nonlinear Relationships :** By using techniques like polynomial features or interaction terms, Logistic Regression can model nonlinear relationships between features and the outcome.
- **Robust to Noise :** Logistic Regression can handle noisy data and outliers by using regularization techniques.

4.5.3 Limitations

- **Limited to Binary Classification :** Logistic Regression is primarily designed for binary classification tasks and may require modifications or extensions for multiclass problems.
- **Linear Decision Boundaries :** Logistic Regression assumes linear decision boundaries, which may limit its ability to capture complex relationships between features.
- **Sensitive to Outliers :** Logistic Regression can be sensitive to outliers that deviate significantly from the majority of the data points.
- **Assumption of Feature Independence :** Logistic Regression assumes that the features are independent of each other, which may not hold in some cases.
- **Lack of Robustness to Irrelevant Features :** Logistic Regression may be influenced by irrelevant features, leading to suboptimal predictions.

4.5.4 Applications

- **Medical Diagnosis :** Logistic Regression is used in various medical applications such as disease prediction, risk assessment, and diagnostic models.
- **Credit Scoring :** Logistic Regression is widely employed in credit scoring models to assess the creditworthiness of individuals based on various factors.
- **Marketing Analysis :** Logistic Regression helps predict customer behavior, such as the likelihood of purchasing a product or churn prediction.
- **Fraud Detection :** Logistic Regression is utilized in fraud detection systems to identify suspicious activities based on historical patterns.
- **Social Sciences :** Logistic Regression finds applications in fields like psychology, sociology, and economics to study factors that influence certain behaviors or outcomes.

Chapter 5

Regression Models

5.1 Decision Tree Regression

5.1.1 Introduction :

Decision trees can also be applied to regression problems. In regression tasks, decision trees create a tree-like model of decisions and their possible consequences, allowing for the exploration of multiple decision paths and capturing complex relationships within the data.

How the Model Works : The basic principles of decision tree construction remain the same for both classification and regression tasks. However, there are some differences in how the model works for regression problems. Here is an overview of the process:

- **Attribute Selection :** Similar to classification, the algorithm selects the best attribute (feature) from the dataset as the root node of the tree based on a selected criterion. In regression, the common criterion for attribute selection is usually based on reducing the variance or mean squared error (MSE) of the target variable.
- **Node Splitting :** The dataset is split into subsets based on the chosen attribute value, creating branches or child nodes. The splitting process continues recursively, considering different attributes and their values, until a stopping criterion is met.
- **Recursion :** The splitting process is applied to each child node until a stopping criterion is met. This can be based on a maximum depth limit, a minimum number of samples per leaf, or other conditions. The goal is to create a tree structure that captures the relationships between the features and the target variable.
- **Leaf Node Assignment :** At the end of the recursion, instead of assigning a class label, the leaf nodes in the decision tree for regression contain predicted values. These values can be the mean or median of the target variable values in that leaf node. When predicting a new data point, it traverses the decision tree based on the attribute values, following the decision path until reaching a leaf node, and then outputs the corresponding predicted value.
- **Prediction :** In the case of decision trees for regression, the prediction process is slightly different from classification. When given a new data point, the decision tree traverses the tree structure based on the attribute values of the data point, following the decision path until reaching a leaf node.

At the leaf node, instead of outputting a class label, the decision tree for regression provides a predicted value. This predicted value represents the expected value or

outcome for the given data point based on the learned patterns and relationships in the training data.

The predicted value at the leaf node can be determined in different ways. One common approach is to assign the mean or median value of the target variable for the training samples that fall into that leaf node. This means that the predicted value for a new data point will be the average or median value of the target variable observed in the training data points that share similar attribute values and end up in the same leaf node as the new data point.

By traversing the decision tree and obtaining the predicted value at the leaf node, decision trees for regression provide a quantitative prediction for the target variable of a new data point based on the learned patterns in the training data.

5.1.2 Advantages

- **Interpretability :** Decision Trees provide human-interpretable rules, making it easy to understand and explain the decision-making process.
- **Nonlinear Relationships :** Decision trees can capture nonlinear relationships between features and the target variable, making them suitable for regression tasks where the underlying relationship is complex.
- **Feature Importance :** Decision trees can rank the importance of features based on their contribution to the decision-making process in regression. This can help identify the most influential features in predicting the target variable.
- **Robustness :** Decision trees are less affected by outliers or noise in the data compared to some other regression algorithms. They can handle noisy data and still provide reasonable predictions.

5.1.3 Limitations

- **Overfitting :** Decision Trees are prone to overfitting, especially when the tree becomes deep or when there are noisy or irrelevant features.
- **Lack of Smoothness :** Decision Trees create step-like boundaries, which may lead to unstable predictions in regions with slight changes in input values.
- **Lack of Generalization :** Decision Trees may not generalize well to unseen data if they have learned specific patterns or outliers in the training set.
- **Bias towards Features with More Levels :** Decision Trees tend to favor attributes with more levels or categories when selecting the best split, potentially neglecting other relevant features.
- **Scalability :** Decision Trees can grow large and complex, making them computationally expensive and challenging to handle with massive datasets.

5.1.4 Applications

- **Predictive Modeling:** Decision Trees are widely used for regression tasks, where the goal is to predict a continuous target variable. Some common applications include predicting house prices, stock market trends, customer lifetime value, or demand forecasting.
- **Feature Importance:** Decision Trees can be used to identify the most informative features in regression problems. By examining the structure of the decision tree and the splits made at each node, we can determine which features have the most significant impact on the predicted outcome. This information can be valuable for feature selection and subsequent modeling or analysis.
- **Anomaly Detection:** Decision Trees can be employed for detecting outliers or anomalies in regression datasets. By learning the normal patterns and relationships in the training data, decision trees can identify data points that deviate significantly from the expected behavior. This can be useful in various domains, such as fraud detection, quality control, or anomaly detection in sensor data.
- **Recommendation Systems:** Decision Trees can be utilized to build recommendation systems for regression tasks. For example, in e-commerce, decision trees can be used to recommend products or services to users based on their preferences and historical data. By analyzing user attributes and past behaviors, decision trees can generate personalized recommendations that match users' interests and needs.
- **Customer Segmentation:** Decision Trees can be applied to segment customers based on their behaviors or characteristics. By analyzing customer attributes and transaction data, decision trees can identify distinct groups of customers with similar preferences or purchase patterns. This information can be useful for targeted marketing campaigns or customer relationship management.
- **Risk Assessment:** Decision Trees can be used for risk assessment and decision-making in various industries, such as insurance or finance. By considering relevant factors and historical data, decision trees can help evaluate the likelihood and impact of different outcomes, enabling organizations to make informed decisions and manage risks effectively.

5.2 Lasso Regression

5.2.1 Introduction :

Lasso Regression, short for Least Absolute Shrinkage and Selection Operator, is a linear regression model that performs both feature selection and regularization. It adds a penalty term to the ordinary least squares objective function, encouraging sparsity by shrinking the coefficients of less important features towards zero.

The Lasso Regression model works as follows :

- **Objective Function :** Lasso Regression minimizes the sum of squared residuals (similar to ordinary linear regression) with an additional regularization term. The regularization term is the L1 norm (absolute value) of the coefficient vector multiplied

by a regularization parameter, λ . The lasso coefficients, $\hat{\beta}_\lambda^L$, minimize the quantity,

$$RSS + \lambda \sum_{j=1}^p |\beta_j| \quad (5.1)$$

- **Regularization and Feature Selection :** By tuning the regularization parameter, Lasso Regression encourages sparsity by driving some coefficients to exactly zero. This leads to automatic feature selection, where less important features are effectively excluded from the model.
- **Tuning the Regularization Parameter :** The λ parameter controls the degree of regularization. Larger values of λ result in more coefficients being shrunk to zero, leading to sparser models.
- **Coefficient Shrinkage :** The Lasso Regression algorithm uses an optimization algorithm, such as coordinate descent, to minimize the objective function and determine the coefficients for the remaining features. The optimization problem is given by,

$$\max_{\beta} \left\{ \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \right\} \quad (5.2)$$

subject to

$$\sum_{j=1}^p |\beta_j| \leq s$$

- **Prediction :** Once the model is trained, Lasso Regression predicts the outcome for new data points by multiplying the feature values with the learned coefficients.

5.2.2 Advantages

- **Feature Selection :** Lasso Regression performs automatic feature selection by shrinking less important features to zero, effectively identifying the most relevant features for the prediction task.
- **Improved Interpretability :** Lasso Regression produces a sparse model, making it easier to interpret and understand the impact of each feature on the outcome.
- **Handles Multicollinearity :** Lasso Regression can handle highly correlated features and select one representative feature while shrinking the coefficients of the others.
- **Regularization for Improved Generalization :** The regularization term in Lasso Regression helps prevent overfitting, leading to better generalization performance on unseen data.
- **Works Well with High-Dimensional Data :** Lasso Regression is particularly effective when dealing with datasets that have a large number of features compared to the number of observations.

5.2.3 Limitations

- **Selection Bias :** Lasso Regression may introduce selection bias, as it tends to favor features with larger effects. Smaller but still important features may be unnecessarily excluded from the model.

- **Sensitive to the Choice of Regularization Parameter :** The performance of Lasso Regression heavily depends on the choice of the regularization parameter. Selecting an appropriate value requires careful tuning.
- **Limited to Linear Relationships :** Lasso Regression assumes a linear relationship between the features and the outcome, which may not hold in all cases. Nonlinear relationships may require other regression techniques.
- **Instability with Highly Correlated Features :** Lasso Regression may be unstable when features are highly correlated, as slight changes in the data can result in different feature selections.
- **Requires Scaling :** Lasso Regression is sensitive to the scale of the features, so it is recommended to scale the features before fitting the model.

5.2.4 Applications

- **Feature Selection :** Lasso Regression is commonly used for feature selection in high-dimensional datasets, such as genomics, image analysis, and text classification.
- **Predictive Modeling :** Lasso Regression can be applied to various predictive modeling tasks, including stock market prediction, customer churn prediction, and credit scoring.
- **Economics and Social Sciences :** Lasso Regression is used in economics and social science research to analyze the impact of various factors on outcomes such as income, employment, or educational attainment.
- **Medical Research :** Lasso Regression has been applied to medical research, including disease prediction, gene expression analysis, and biomarker selection.
- **Signal Processing :** Lasso Regression is used in signal processing applications such as image denoising and signal reconstruction.

5.3 Linear Regression :

Introduction :

Linear Regression is a statistical model used to describe the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the variables, where the dependent variable is a linear combination of the independent variables, along with an error term.

The Linear Regression model works as follows :

- **Hypothesis Function :** Linear Regression lives up to its name: it is a very straightforward simple linear approach for predicting a quantitative response Y on the basis of a single predictor variable X . It assumes that there is approximately a linear relationship between X and Y . Mathematically, we can write this linear relationship as

$$Y \approx \beta_0 + \beta_1 X \quad (5.3)$$

β_0 and β_1 are unknown.

- **Training Phase :** During training, the model learns the optimal values of the coefficients by minimizing the sum of squared residuals between the predicted and actual values. Let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ be the prediction for Y based on the i^{th} value of X . Then $e_i = y_i - \hat{y}_i$ represents the i^{th} residual this is the difference between the i^{th} observed response value and the i^{th} response value that is predicted by our linear model. We define the residual sum of squares (RSS) as

$$RSS = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \dots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2 \quad (5.4)$$

- **Coefficient Estimation :** The coefficients are estimated in such a way that they minimize the difference between the predicted values and the actual values. The least squares approach chooses $\hat{\beta}_0$ and $\hat{\beta}_1$ to minimize the RSS. Using some calculus, one can show that the minimizers are

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \quad (5.5)$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

where $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ and $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ are the sample means.

- **Prediction :** Once the model is trained, it can be used to make predictions on new, unseen data by substituting the feature values into the learned coefficients.

5.3.1 Advantages

- **Simplicity and Interpretability :** Linear Regression is a simple and interpretable model that provides insight into the relationships between variables.
- **Efficiency :** The model can be trained quickly, even with large datasets, and can handle high-dimensional feature spaces.
- **Baseline for Comparison :** Linear Regression provides a baseline against which more complex models can be evaluated.
- **Feature Importance :** The model can indicate the relative importance of features based on the magnitude of their coefficients.
- **No Assumptions on Feature Distribution :** Linear Regression does not assume a specific distribution for the features, unlike some other models.

5.3.2 Limitations

- **Linearity Assumption :** Linear Regression assumes a linear relationship between the independent variables and the dependent variable. It may not capture complex, nonlinear relationships without additional transformations.
- **Sensitivity to Outliers :** The model is sensitive to outliers, which can heavily influence the estimated coefficients and predictions.
- **Multicollinearity :** Linear Regression can struggle with highly correlated features, leading to unstable and unreliable coefficient estimates.
- **Independence of Errors :** Linear Regression assumes that the errors (residuals) are independent and normally distributed, which may not hold in all cases.

- **Overfitting :** The model can overfit the training data if the number of features is large compared to the number of observations, resulting in poor generalization to new data.

5.3.3 Applications

- **Predictive Modeling :** Linear Regression is widely used for predicting continuous outcomes, such as sales forecasting, house price prediction, and stock market analysis.
- **Economics and Finance :** Linear Regression is used to model the relationships between economic factors, such as GDP, inflation, and unemployment rates.
- **Social Sciences :** Linear Regression finds applications in social science research to analyze factors influencing behaviors or outcomes, such as education attainment or crime rates.
- **Health and Medicine :** Linear Regression is used to study the relationships between medical factors and outcomes, such as the impact of lifestyle factors on disease risk.
- **Environmental Sciences :** Linear Regression is employed in environmental studies to analyze the relationships between environmental variables, such as temperature, rainfall, and biodiversity.

5.4 Ridge Regression

5.4.1 Introduction :

Ridge Regression is a linear regression model that incorporates L2 regularization to address the problem of multicollinearity and improve the model's stability. It adds a penalty term to the ordinary least squares objective function, which helps to prevent overfitting and provides more reliable coefficient estimates.

The Ridge Regression model works as follows :

- **Objective Function :** Ridge Regression minimizes the sum of squared residuals (similar to ordinary linear regression) with an additional regularization term. The regularization term is the L2 norm (squared magnitude) of the coefficient vector multiplied by a regularization parameter, lambda. The ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

$$RSS + \lambda \sum_{j=1}^p \beta_j^2 \quad (5.6)$$

where $\lambda \geq 0$ is a tuning parameter, to be determined separately. The second term, $\lambda \sum_j \beta_j^2$, called a shrinkage penalty.

- **Regularization and Shrinking Coefficients :** By tuning the regularization parameter, Ridge Regression shrinks the coefficients towards zero without eliminating them entirely. The degree of shrinkage is controlled by the lambda parameter, with larger values resulting in more significant shrinkage.
- **Bias-Variance Trade-off :** The addition of the regularization term helps to strike a balance between minimizing the squared residuals and reducing the magnitude of the coefficients. This trade-off is particularly useful in situations where multicollinearity exists among the independent variables.

- **Ridge Regression Solution :** The model uses an optimization algorithm to minimize the objective function and find the optimal values of the coefficients that best fit the data.
- **Prediction :** Once the model is trained, Ridge Regression predicts the outcome for new data points by multiplying the feature values with the learned coefficients.

5.4.2 Advantages

- **Improved Stability :** Ridge Regression provides more stable coefficient estimates by reducing the impact of multicollinearity, making it less sensitive to slight changes in the input data.
- **Reduces Overfitting :** The regularization term in Ridge Regression helps to prevent overfitting by adding a penalty for complex models, leading to improved generalization performance on unseen data.
- **Handles Multicollinearity :** Ridge Regression is effective in handling situations where the independent variables are highly correlated, as it reduces the impact of multicollinearity on the coefficient estimates.
- **Controls Feature Importance :** Ridge Regression allows for the identification of important features by shrinking less relevant coefficients closer to zero. This can aid in feature selection and interpretability of the model.
- **Works Well with Large Feature Spaces :** Ridge Regression is suitable for datasets with a large number of features compared to the number of observations.

5.4.3 Limitations

- **Linearity Assumption :** Like linear regression, Ridge Regression assumes a linear relationship between the independent variables and the dependent variable. Nonlinear relationships may require other regression techniques.
- **Sensitivity to Scaling :** Ridge Regression is sensitive to the scale of the features. It is recommended to scale the features before fitting the model to ensure fair treatment of different variables.
- **Limited Feature Elimination :** Unlike some other regularization techniques, such as Lasso Regression, Ridge Regression does not eliminate features entirely but shrinks them towards zero. Therefore, it may not be effective for feature selection tasks requiring sparsity.
- **Choice of Regularization Parameter :** The performance of Ridge Regression depends on the choice of the regularization parameter. Selecting an appropriate value requires careful tuning, such as using cross-validation.

5.4.4 Applications

- **Economics and Finance :** Ridge Regression finds applications in economic and financial studies, such as predicting stock prices, analyzing factors affecting economic growth, or assessing credit risk.

- **Healthcare and Medicine :** Ridge Regression is used in medical research to model the relationship between various clinical or genetic factors and health outcomes, such as disease progression or treatment response.
- **Environmental Sciences :** Ridge Regression is employed to analyze the impact of environmental factors on ecological processes, such as species distribution modeling or climate change studies.
- **Marketing and Customer Analytics :** Ridge Regression is applied in marketing research to analyze consumer behavior, predict customer preferences, or optimize marketing strategies.
- **Social Sciences :** Ridge Regression is used in social science research to study the relationships between socioeconomic factors, education, or public health outcomes.

5.5 Polynomial Regression

5.5.1 Introduction :

Polynomial Regression is a regression technique that models the relationship between the independent variable(s) and the dependent variable as an n th-degree polynomial. It extends the linear regression model by including higher-order polynomial terms, allowing for more flexible curve fitting.

The Polynomial Regression model works as follows :

- **Polynomial Features :** The original features (independent variables) are transformed into polynomial features by raising them to different powers. For example, a second-degree polynomial regression would include the original features and their squared terms.
- **Model Fitting :** The polynomial features and the corresponding target values are used to fit a linear regression model. This is done by finding the coefficients that minimize the sum of squared residuals between the predicted and actual values.
- **Prediction :** Once the model is trained, it can be used to predict the outcome for new data points. The polynomial features of the new data points are computed, and the model applies the learned coefficients to make predictions.

5.5.2 Advantages

- **Flexibility :** Polynomial Regression can capture nonlinear relationships between the independent and dependent variables, making it suitable for more complex data patterns.
- **Improved Fit :** By allowing for higher-degree polynomial terms, Polynomial Regression can better fit curvilinear data compared to linear regression models.
- **Feature Engineering :** Polynomial Regression automatically includes polynomial features, eliminating the need for manual feature engineering.
- **Interpretability :** Polynomial Regression allows for the interpretation of the coefficients associated with each polynomial term, providing insights into the relationship between variables.

5.5.3 Limitations

- **Overfitting :** Higher-degree polynomials can lead to overfitting, where the model fits the training data too closely and performs poorly on new, unseen data.
- **Interpretability :** As the degree of the polynomial increases, the model becomes more complex, making it harder to interpret the relationships between variables.
- **Extrapolation :** Polynomial Regression is not reliable for extrapolation beyond the range of the observed data, as the fitted curve may diverge from the true relationship.
- **Model Selection :** Determining the appropriate degree of the polynomial is challenging and requires trade-offs between bias and variance.
- **Sensitivity to Outliers :** Polynomial Regression can be sensitive to outliers, as they can heavily influence the fitted curve.

5.5.4 Applications

- **Physics and Engineering :** Polynomial Regression is commonly used in physics and engineering to model physical phenomena and capture nonlinear relationships in experimental data.
- **Economics and Finance :** Polynomial Regression can be applied in economic and financial studies to analyze trends, forecast economic indicators, or predict stock prices.
- **Environmental Sciences :** Polynomial Regression is used in environmental sciences to model and predict phenomena such as temperature changes, rainfall patterns, or pollutant concentrations.
- **Image Processing :** Polynomial Regression can be employed in image processing tasks, such as image reconstruction or edge detection.
- **Medicine and Healthcare :** Polynomial Regression finds applications in medical research, such as modeling disease progression, analyzing the impact of risk factors, or predicting patient outcomes.

5.6 Support Vector Regression

5.6.1 Introduction :

Support Vector Regression aims to find a function that approximates the mapping from input variables to the target variable in a regression problem. It identifies a hyperplane that best fits the training data, while also minimizing the margin violations. SVR uses support vectors, which are data points close to the hyperplane that define the decision boundary.

How the Model Works :

- **Kernel Function :** SVR starts by selecting a suitable kernel function that can transform the input data into a higher-dimensional feature space. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid.
- **Training Process :** SVR aims to minimize the margin violations while fitting as many data points as possible within a given margin. The margin is controlled by two hyperparameters: C (trade-off between margin size and error tolerance) and epsilon (tolerance for errors).

- **Regression Function :** SVR finds a hyperplane that maximizes the margin while including as many data points within the margin as possible. The hyperplane is represented by a regression function that predicts the target variable for new input data.

5.6.2 Advantages

- **Effective with Non-Linear Data :** SVR can handle non-linear and complex relationships between variables by using different kernel functions.
- **Robust to Outliers :** SVR is less sensitive to outliers in the training data due to its focus on support vectors close to the hyperplane.
- **Flexibility in Kernel Selection :** SVR offers various kernel functions, allowing users to choose the most suitable one for their specific problem.
- **Good Generalization :** SVR tends to generalize well to unseen data by finding the hyperplane with the largest margin.
- **Few Hyperparameters :** SVR has relatively few hyperparameters to tune, making it easier to implement and optimize.

5.6.3 Limitations

- **Computationally Intensive :** SVR can be computationally expensive, especially for large datasets or complex kernel functions.
- **Sensitivity to Kernel Choice :** The performance of SVR heavily depends on the choice of the kernel function and its parameters.
- **Difficulty in Interpretability :** The interpretation of SVR models is often challenging due to the non-linear transformations performed by the kernel functions.
- **Memory Requirements :** SVR requires storing a subset of support vectors, which can consume a significant amount of memory for large datasets.

5.6.4 Applications

- **Financial Forecasting :** SVR can be used to predict stock prices, market trends, or asset values.
- **Energy Demand Prediction :** SVR can forecast energy demand based on historical data and external factors.
- **Medicine and Healthcare :** SVR can assist in predicting patient outcomes, disease progression, or drug response.
- **Environmental Modeling :** SVR can be applied to predict pollution levels, climate patterns, or ecosystem dynamics.
- **Time Series Analysis :** SVR can be used for time series forecasting, such as predicting sales or website traffic.

Chapter 6

Usage and Demonstration

6.1 Steps to Use GUI

Step 1

Select anyone of the model from Classification or Regression menubar of the GUI main window.(e.g. "Decision Tree Classifier")



Figure 6.1: GUI Mainwindow

Step 2

A new window (i.e., GUI) for executing your selected model has popped up, which includes different options such as parameters, hyperparameters, and attributes specific to the model.

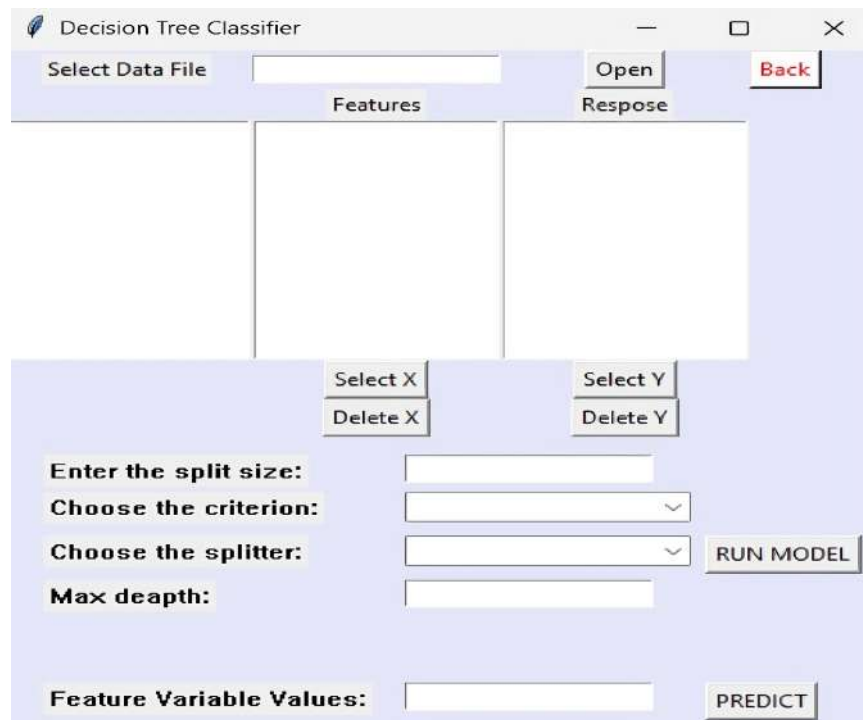


Figure 6.2: Model MainWindow

Step 3

Select the dataset on which you want to work by clicking the "Open" button.(e.g. "IRIS")

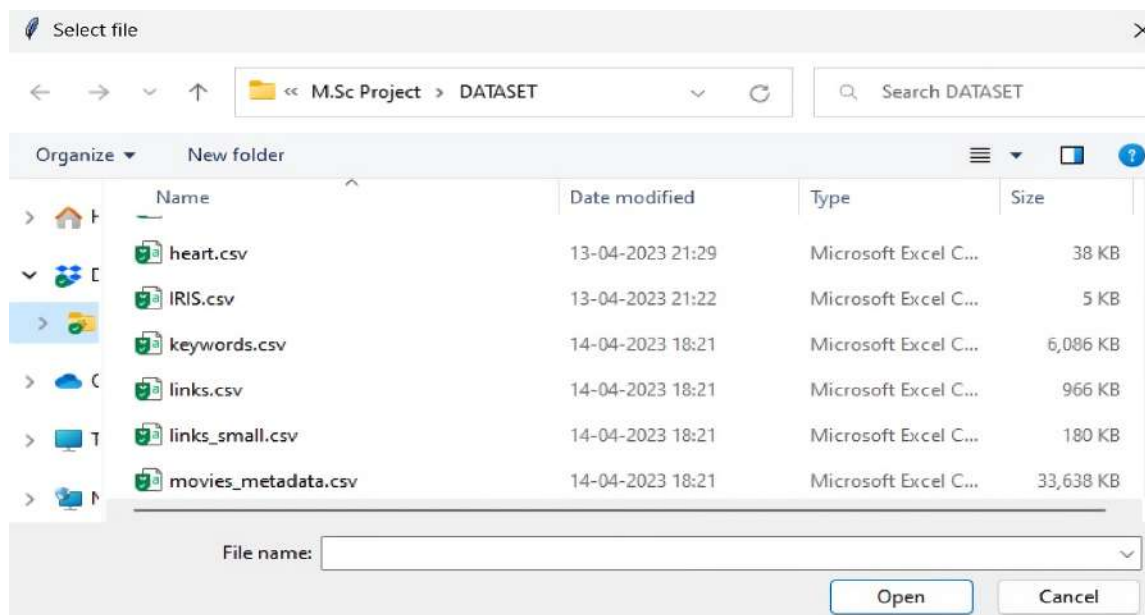


Figure 6.3: Dataset Selection

Step 4

The selected dataset will be displayed in a table format, allowing one to explore the data and variables included in it.

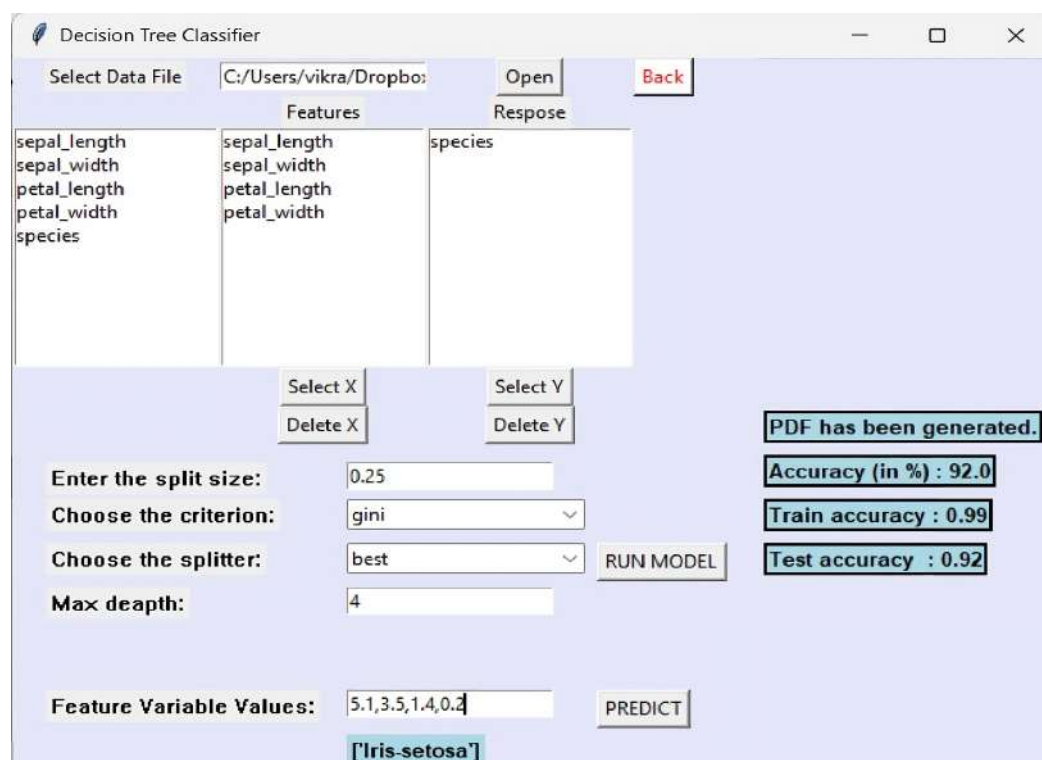


	Id	SepalLen	SepalWid	PetalLen	PetalWid	Species
1	1	5.10	3.50	1.40	0.2	Iris-setosa
2	2	4.90	3.00	1.40	0.2	Iris-setosa
3	3	4.70	3.20	1.30	0.2	Iris-setosa
4	4	4.60	3.10	1.50	0.2	Iris-setosa
5	5	5.00	3.60	1.40	0.2	Iris-setosa
6	6	5.40	3.90	1.70	0.4	Iris-setosa
7	7	4.60	3.40	1.40	0.3	Iris-setosa
8	8	5.00	3.40	1.50	0.2	Iris-setosa
9	9	4.40	2.90	1.40	0.2	Iris-setosa
10	10	4.90	3.10	1.50	0.1	Iris-setosa
11	11	5.40	3.70	1.50	0.2	Iris-setosa
12	12	4.80	3.40	1.60	0.2	Iris-setosa

Figure 6.4: Data Display

Step 5

By selecting the response (dependent) variable and the features (independent) variables, and choosing the parameters and hyperparameters according to the model, we can generate the model output in PDF format. Additionally, by providing values for the feature variables and clicking on the "Predict" button, we can obtain the response for test data.



Decision Tree Classifier

Select Data File: C:/Users/vikra/Dropbox/ Open Back

Features: sepal_length, sepal_width, petal_length, petal_width, species

Response: species

Select X, Delete X, Select Y, Delete Y

Enter the split size: 0.25

Choose the criterion: gini

Choose the splitter: best

Max depth: 4

RUN MODEL

PDF has been generated.

Accuracy (in %) : 92.0

Train accuracy : 0.99

Test accuracy : 0.92

Feature Variable Values: 5.1,3.5,1.4,0.2

PREDICT

[Iris-setosa]

Figure 6.5: Output and Prediction

6.2 Results on Datasets

6.2.1 Decision Tree Classification

About Dataset :

- Context :
This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.
- Content : The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.
- Inspiration : A data frame with 400 observations on the following 11 variables.

The screenshot shows the 'Decision Tree Classifier' GUI. At the top, there's a 'Select Data File' section with a text box containing 'C:/Project/ML Models/' and buttons for 'Open' and 'Back'. Below this is a table for selecting features and the response variable. The 'Features' column lists 11 variables: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunc, Age, and Outcome. The 'Response' column is empty. To the right of the table are buttons for 'Select X', 'Delete X', 'Select Y', and 'Delete Y'. Below the table, there are input fields for 'Enter the split size:' (0.30), 'Choose the criterion:' (gini), 'Choose the splitter:' (random), and 'Max deapth:' (4). A 'RUN MODEL' button is to the right of these inputs. On the far right, a box displays the results: 'PDF has been generated.', 'Accuracy (in %) : 73.0', 'Train accuracy : 0.76', and 'Test accuracy : 0.73'. At the bottom, there's a 'Feature Variable Values:' text box and a 'PREDICT' button.

Select Data File	C:/Project/ML Models/	Open	Back																						
<table border="1"> <thead> <tr> <th>Features</th> <th>Response</th> </tr> </thead> <tbody> <tr> <td>Unamed: 0</td> <td>Pregnancies</td> </tr> <tr> <td>Pregnancies</td> <td>Glucose</td> </tr> <tr> <td>Glucose</td> <td>BloodPressure</td> </tr> <tr> <td>BloodPressure</td> <td>SkinThickness</td> </tr> <tr> <td>SkinThickness</td> <td>Insulin</td> </tr> <tr> <td>Insulin</td> <td>BMI</td> </tr> <tr> <td>BMI</td> <td>DiabetesPedigreeFunc</td> </tr> <tr> <td>DiabetesPedigreeFunc</td> <td>Age</td> </tr> <tr> <td>Age</td> <td></td> </tr> <tr> <td>Outcome</td> <td></td> </tr> </tbody> </table>				Features	Response	Unamed: 0	Pregnancies	Pregnancies	Glucose	Glucose	BloodPressure	BloodPressure	SkinThickness	SkinThickness	Insulin	Insulin	BMI	BMI	DiabetesPedigreeFunc	DiabetesPedigreeFunc	Age	Age		Outcome	
Features	Response																								
Unamed: 0	Pregnancies																								
Pregnancies	Glucose																								
Glucose	BloodPressure																								
BloodPressure	SkinThickness																								
SkinThickness	Insulin																								
Insulin	BMI																								
BMI	DiabetesPedigreeFunc																								
DiabetesPedigreeFunc	Age																								
Age																									
Outcome																									
<input type="button" value="Select X"/> <input type="button" value="Delete X"/>		<input type="button" value="Select Y"/> <input type="button" value="Delete Y"/>																							
Enter the split size: <input type="text" value="0.30"/>		<input type="button" value="PDF has been generated."/>																							
Choose the criterion: <input type="text" value="gini"/>		<input type="button" value="Accuracy (in %) : 73.0"/>																							
Choose the splitter: <input type="text" value="random"/>		<input type="button" value="Train accuracy : 0.76"/>																							
Max deapth: <input type="text" value="4"/>		<input type="button" value="Test accuracy : 0.73"/>																							
<input type="button" value="RUN MODEL"/>																									
Feature Variable Values: <input type="text"/>		<input type="button" value="PREDICT"/>																							

Figure 6.6: Decision Tree Classification GUI Window

- PDF Output

Classification Report

	precision	recall	f1-score	support
Yes	0.86	0.71	0.78	160.0
No	0.54	0.75	0.62	71.0
accuracy	0.72	0.72	0.72	0.72
macro avg	0.7	0.73	0.7	231.0
weighted avg	0.76	0.72	0.73	231.0

Figure 6.7: Classification Report

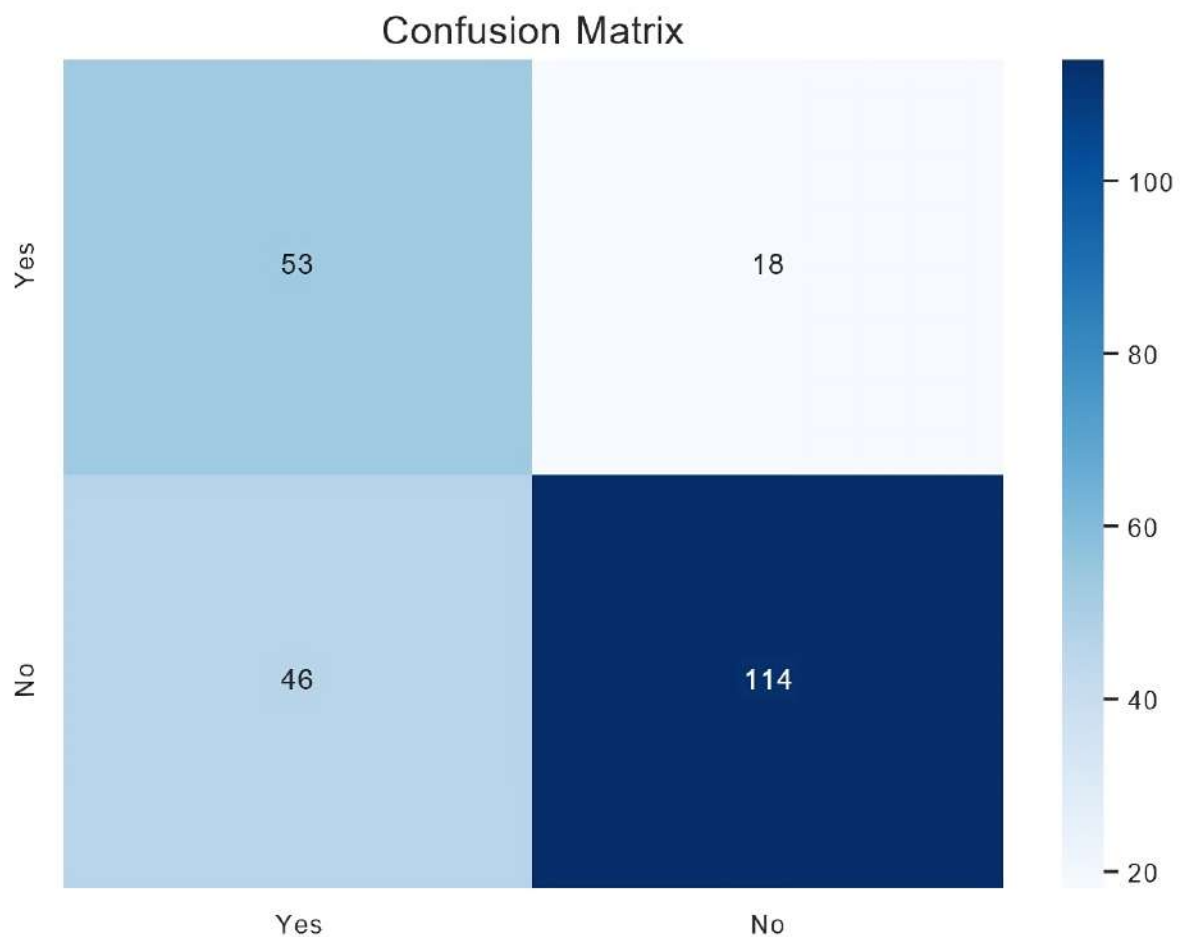


Figure 6.8: Confusion Matrix Plot

Decision Tree

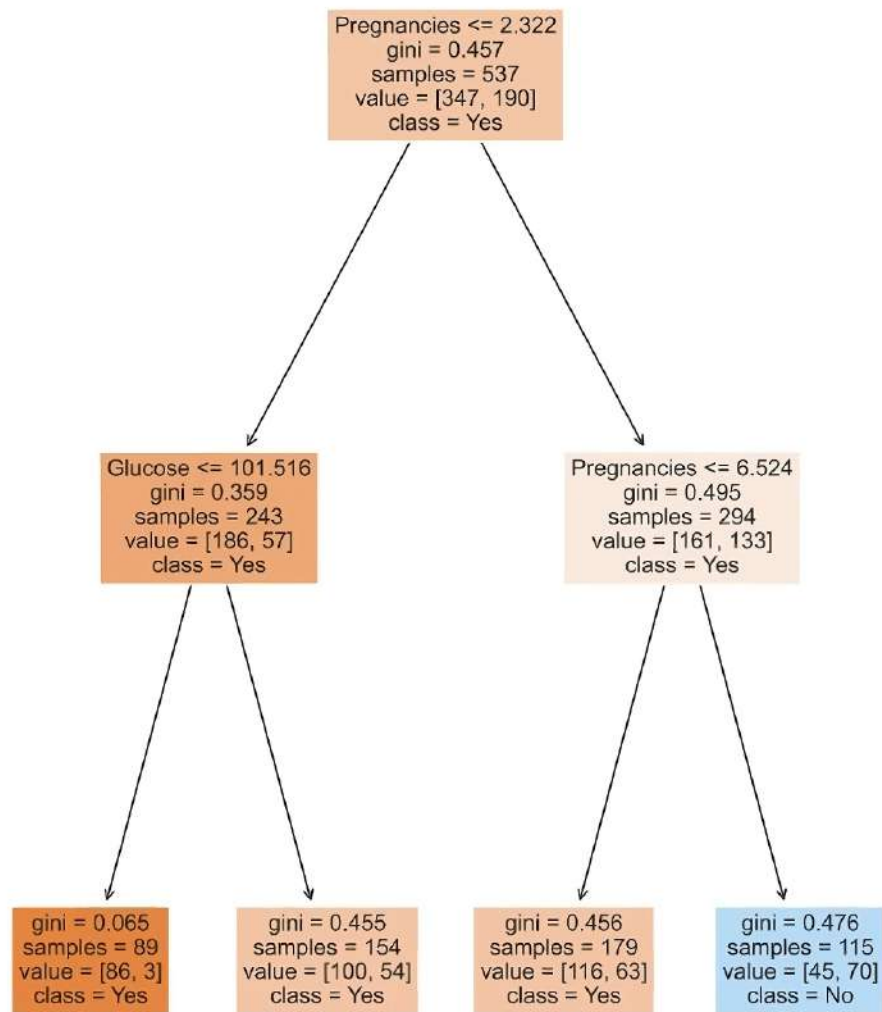


Figure 6.9: Tree

6.2.2 Support Vector Classification

- **About Dataset :**

We have used bill authentication dataset, this were extracted from images that were taken from genuine and forged banknote-like specimens. Attributes used here are

- Variance of Wavelet transformed image(continuous)
- Skewness of Wavelet Transformed image(continuous)
- Kurtosis of Wavelet Transformed image(continuous)
- Entropy of Wavelet Tranformed image(continuous)
- Class as response variable

In this case there are multiple hyperparameters like **split size**, **Penalty** (the penalty parameter C controls the trade-off between achieving a low training error and a low testing error that is the ability of the model to generalize to new data. A smaller value of C creates a wider margin hyperplane at the cost of more margin violations (i.e., misclassifications), while a

larger value of C creates a narrower margin hyperplane and fewer margin violations), **Kernel** (is used to map the input space into a higher-dimensional space where it is easier to classify the data), **Decision Function Shape(dca)** (This method basically returns a Numpy array, In which each element represents whether a predicted sample for x_{test} by the classifier lies to the right or left side of the Hyperplane and also how far from the HyperPlane).

After selecting the appropriate hyperparameters and clicking the "RUN MODEL" button, the model is executed, generating a PDF report. The output includes the overall accuracy of the model, as well as the training and testing accuracies. The PDF report comprises descriptive statistics for numeric variables, a classification report, and a confusion matrix. In addition, we can specify the unknown or test data in the "Feature variable value" section. After clicking the "PREDICT" button, we can obtain the desired output.

Figure 6.10: Support Vector Classifier GUI Window

- PDF Output

Descriptive Statistics for Numeric Variables

	Variance	Skewness	Kurtosis	Entropy
count	1372.0	1372.0	1372.0	1372.0
mean	0.43	1.92	1.4	-1.19
std	2.84	5.87	4.31	2.1
min	-7.04	-13.77	-5.29	-8.55
25%	-1.77	-1.71	-1.57	-2.41
50%	0.5	2.32	0.62	-0.59
75%	2.82	6.81	3.18	0.39
max	6.82	12.95	17.93	2.45

Figure 6.11: Descriptive Statistics for Numerical Variables

Classification Report

	precision	recall	f1-score	support
0	0.98	0.91	0.95	164.0
1	0.89	0.97	0.93	111.0
accuracy	0.94	0.94	0.94	0.94
macro avg	0.93	0.94	0.94	275.0
weighted avg	0.94	0.94	0.94	275.0

Figure 6.12: Classification Report

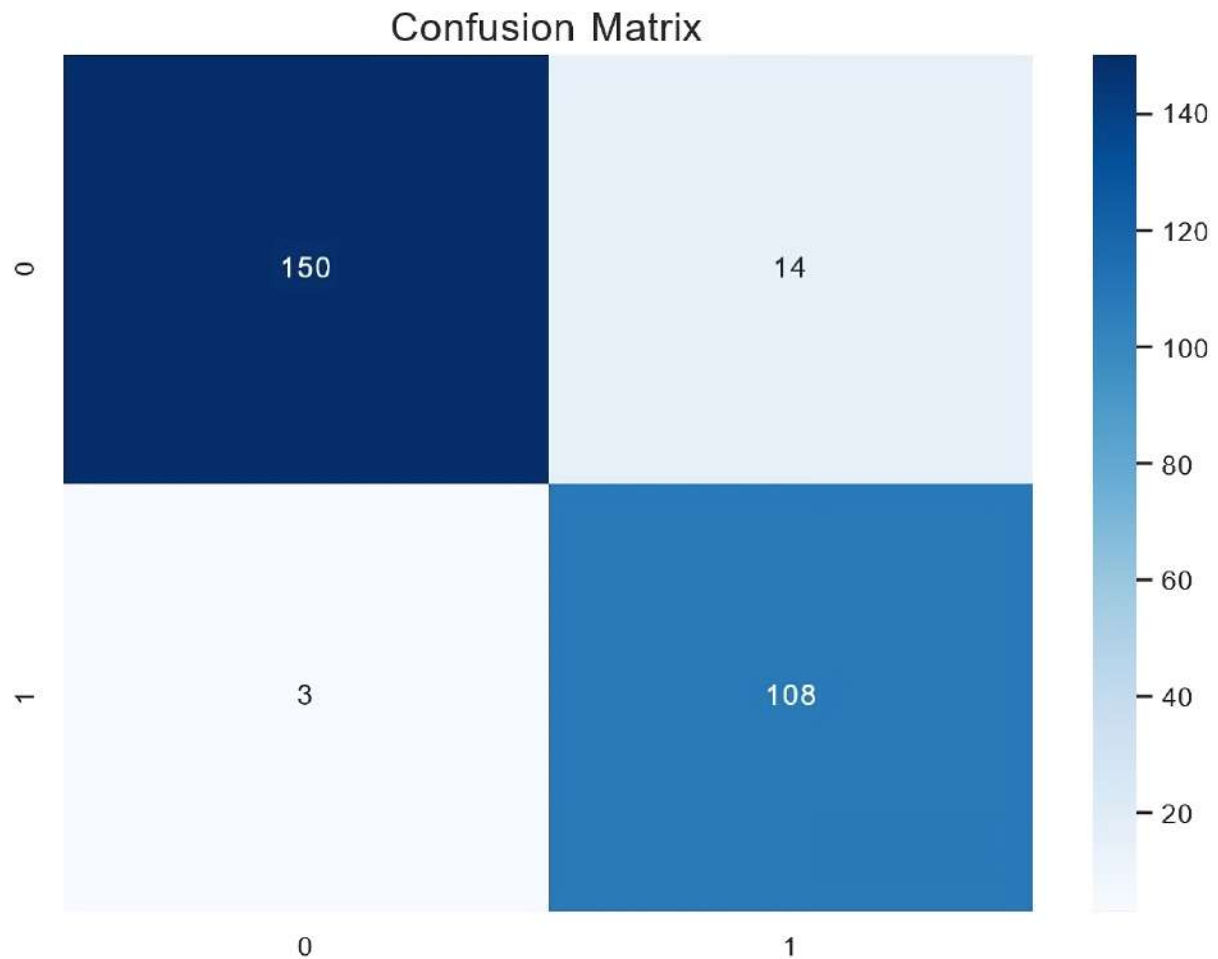


Figure 6.13: Confusion Matrix

6.2.3 K - Nearest Neighbourhood

- **About Dataset :**

- The Iris dataset is a well-known and frequently used dataset in machine learning and data analysis. It was introduced by the British statistician and biologist Ronald Fisher in 1936 and has since become a benchmark dataset for classification algorithms.
- **Origin :** The Iris dataset was derived from a study conducted by Fisher, in which he measured the sepal and petal lengths and widths of three different species of Iris flowers. The study involved collecting data from 150 Iris flowers, with 50 samples from each of the following species: Iris setosa, Iris virginica, and Iris versicolor.
- **Structure :** The dataset consists of four features/attributes: sepal length, sepal width, petal length, and petal width. These measurements were taken from the flowers' sepals and petals, which are the leaf-like structures surrounding the reproductive parts of the flower.
- **Target variable :** The target variable or the class label represents the species of the Iris flower. It is encoded as integers, with 0 representing Iris setosa, 1 representing Iris versicolor, and 2 representing Iris virginica.

- **Dataset size :** The Iris dataset contains 150 instances or samples, each having four attributes (sepal length, sepal width, petal length, and petal width) and a corresponding class label.
- **Purpose :** The dataset is commonly used for classification tasks to train and evaluate machine learning algorithms. It serves as a benchmark dataset for evaluating the performance of various classification algorithms, as it is relatively small, well-structured, and has distinct classes.
- **Insights :** The Iris dataset is valuable because it allows for the exploration of different classification techniques, feature selection, and visualization methods. Its distinct species and well-separated clusters of data points make it a suitable dataset for demonstrating classification algorithms' effectiveness.
- **Usage :** The Iris dataset is widely used in machine learning education, research, and benchmarking. Many machine learning frameworks and libraries provide built-in functions to load and work with the dataset, making it easily accessible.

K-Nearest Neighbors

Select Data File: C:/Users/vikra/Dropbo... Open Back

Features	Response
sepal_length	species
sepal_width	
petal_length	
petal_width	

Select X Select Y
Delete X Delete Y

Enter the split size: 0.2
The number of neighbors: 3
Choose the weight type: distance
Choose algorithm type: auto

RUN MODEL

Accuracy (in %) : 93.0
Test accuracy : 0.93
Train accuracy : 1.0

Feature Variable Values: 5.1,3.6,2.3,.2 PREDICT

[Iris-setosa]

Figure 6.14: KNN MAIN GUI

- **PDF Output :**

Descriptive Statistics

	sepal length	sepal width	petal length	petal width
count	150.0	150.0	150.0	150.0
mean	5.84	3.05	3.76	1.2
std	0.83	0.43	1.76	0.76
min	4.3	2.0	1.0	0.1
25%	5.1	2.8	1.6	0.3
50%	5.8	3.0	4.35	1.3
75%	6.4	3.3	5.1	1.8
max	7.9	4.4	6.9	2.5

Figure 6.15: KNN Descriptive Statistics

Classification Report

	precision	recall	f1-score	support
Iris-setosa	1.0	1.0	1.0	9.0
Iris-versicolor	0.91	0.91	0.91	11.0
Iris-virginica	0.9	0.9	0.9	10.0
accuracy	0.93	0.93	0.93	0.93
macro avg	0.94	0.94	0.94	30.0
weighted avg	0.93	0.93	0.93	30.0

Figure 6.16: KNN Classification Report

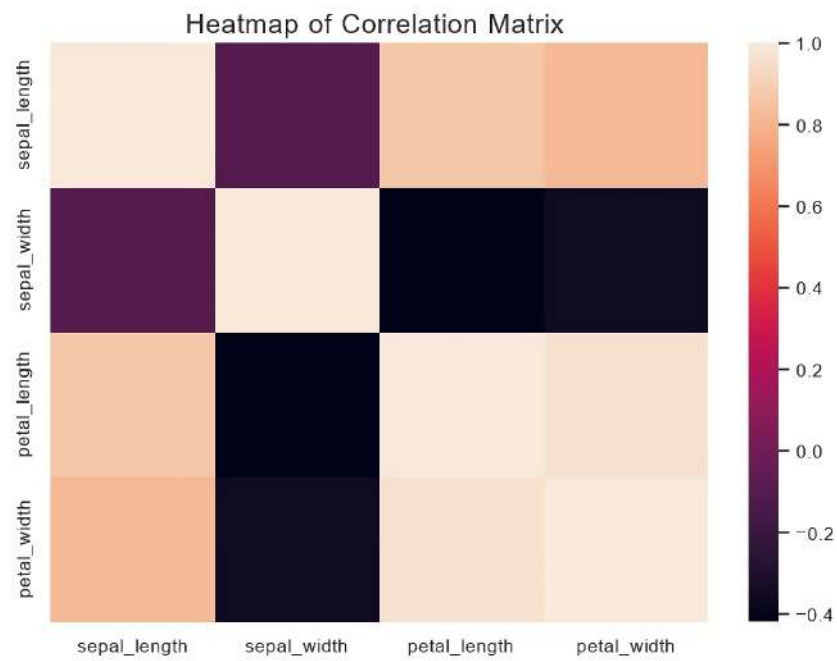


Figure 6.17: KNN Correlation Matrix

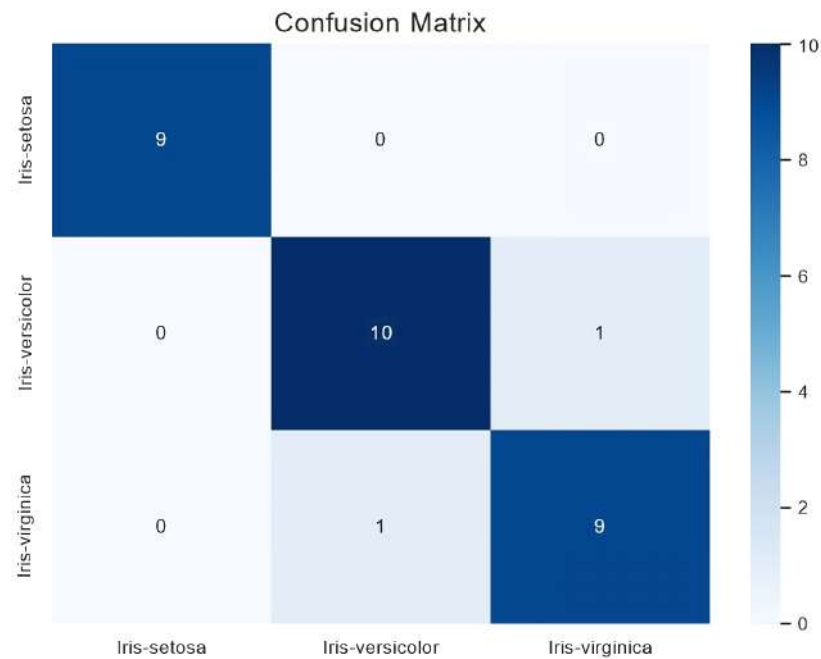


Figure 6.18: KNN Confusion Matrix

6.2.4 Naive Bayes Classifier

- **About Dataset :**

- The Breast Cancer dataset is commonly used dataset in machine learning and data analysis, particularly for classification tasks related to breast cancer diagnosis.
- **Origin:** The Breast Cancer dataset was created by Dr. William H. Wolberg, a physician at the University of Wisconsin Hospital, to aid in the diagnosis of breast cancer. It was first made publicly available in 1992.
- **Structure :** The dataset consists of several features or attributes that describe characteristics of breast mass samples. The attributes include various measurements such as radius, texture, perimeter, area, smoothness, compactness, concavity, symmetry, and fractal dimension. These measurements were derived from digitized images of Fine Needle Aspirate (FNA) samples obtained from breast mass images.
- **Target variable :** The target variable represents the diagnosis of the breast mass, indicating whether it is malignant (cancerous) or benign (non-cancerous). It is encoded as integers, with 0 representing benign and 1 representing malignant.
- **Dataset size :** The Breast Cancer dataset contains 569 instances or samples, each having 30 attributes and a corresponding class label.
- **Purpose :** The dataset is primarily used for classification tasks, particularly for developing models that can accurately diagnose breast masses as malignant or benign. It allows researchers and machine learning practitioners to explore different classification algorithms, feature selection methods, and model evaluation techniques specific to breast cancer diagnosis.
- **Insights :** The Breast Cancer dataset provides an opportunity to study and understand the factors that contribute to the diagnosis of breast cancer. By analyzing the relationships between the various attributes and the target variable, researchers can gain insights into the importance of different features and identify patterns that can aid in accurate diagnosis.
- **Usage :** The Breast Cancer dataset has been widely used in machine learning research, particularly in the field of medical diagnostics. It serves as a benchmark dataset for evaluating the performance of classification algorithms in breast cancer diagnosis and has been extensively used to develop predictive models for early detection of breast cancer.

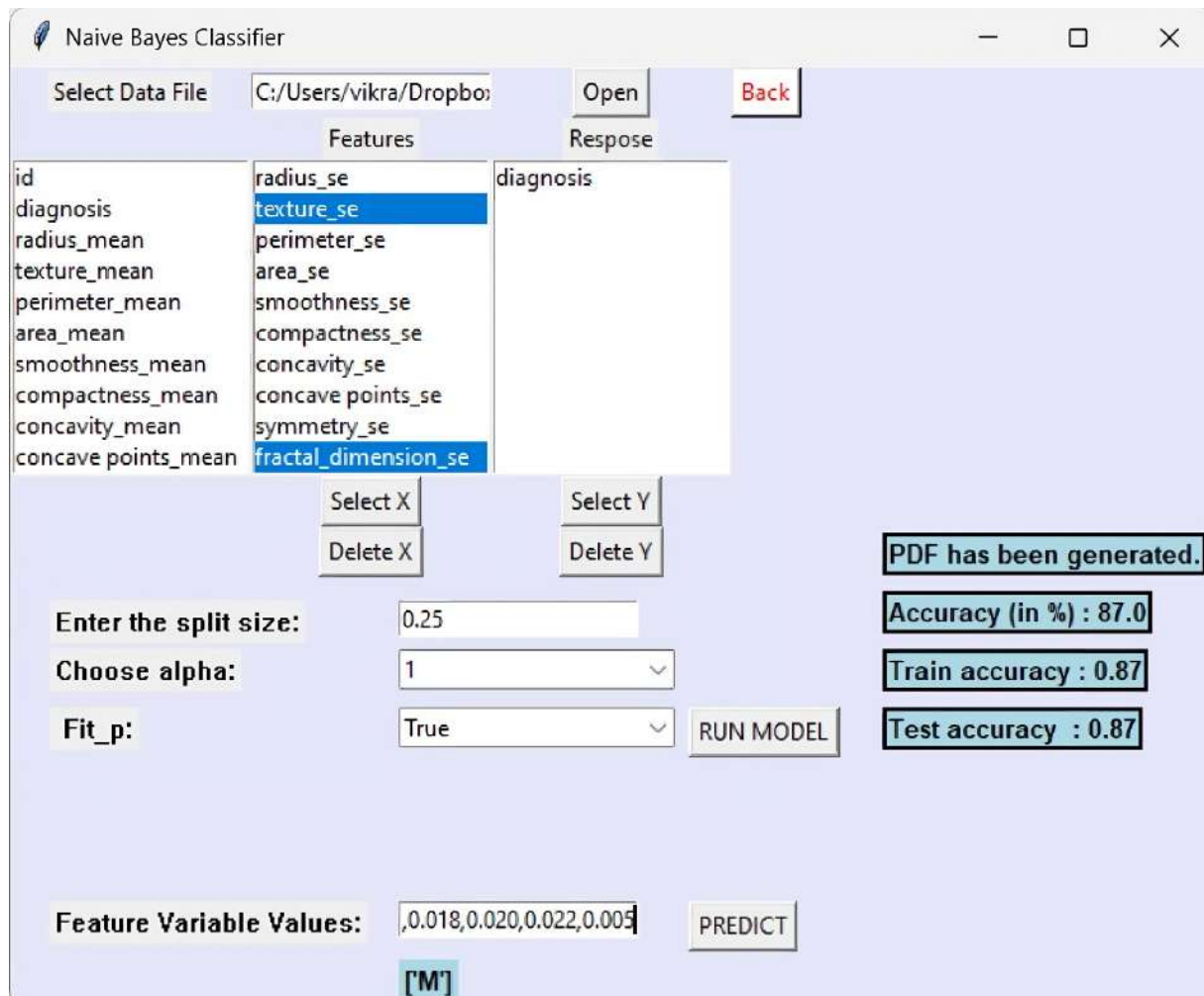


Figure 6.19: NBC GUI Mainwindow

- **PDF Output**

Classification Report

	precision	recall	f1-score	support
M	0.82	0.96	0.88	75.0
B	0.95	0.76	0.85	68.0
accuracy	0.87	0.87	0.87	0.87
macro avg	0.88	0.86	0.86	143.0
weighted avg	0.88	0.87	0.87	143.0

Figure 6.20: NBC Classification Report

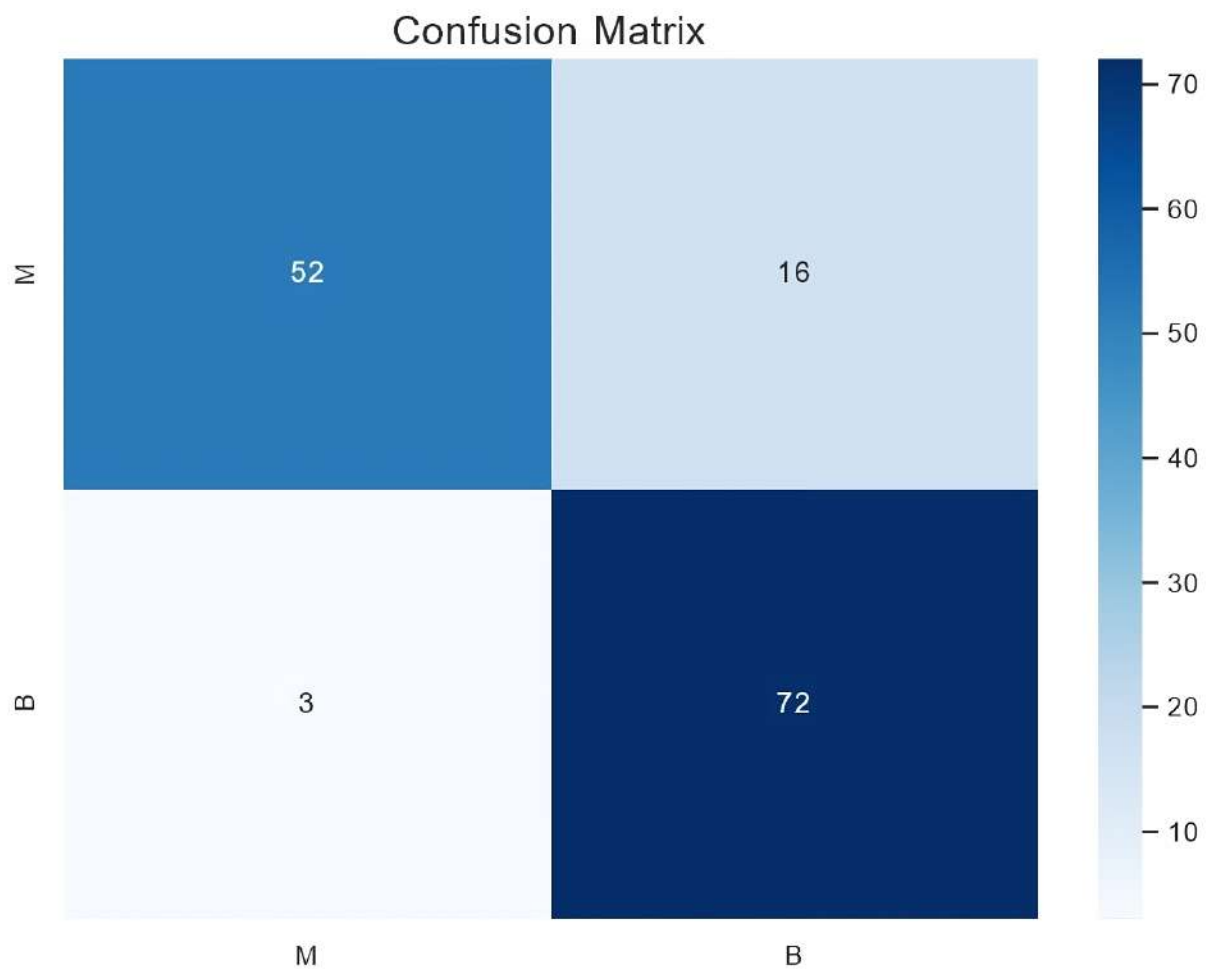


Figure 6.21: NBC Confusion Matrix

6.2.5 Linear Regression

- **About Dataset :**

The following dataset contains information about the years of experience of employees

and their salaries. We applied a linear regression tool to this data, considering years of experience as the independent variable and salary as the dependent variable.

In this case, there is only one hyperparameter that needs to be specified, which is the split size. We chose a split size of 0.25, meaning that the data is divided into 75% training data and 25% test data.

After specifying the split size, we clicked on the "RUN MODEL" button, which executed the model. The output includes the parameters of the model, accuracy metrics, and the generation of a PDF report. The report includes descriptive statistics for numeric variables, performance metrics, and a heatmap of the correlation matrix. We can also specify the unknown or the test data in Feature variable value section and after clicking the "PREDICT" button we can also the specified output.

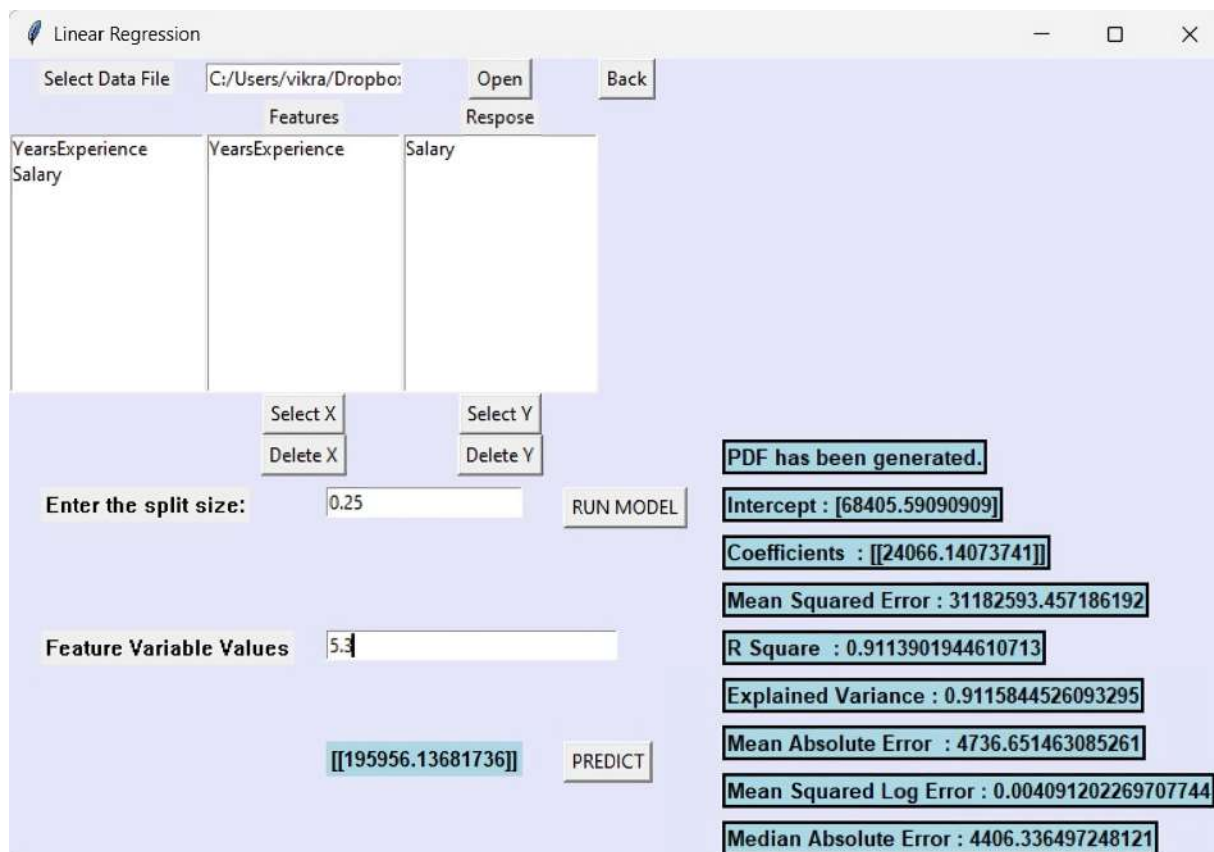


Figure 6.22: Linear Regression GUI Window

- **PDF Output**

In the output section we have several scoring parameter which measures the performance of model.

1. **R^2 score, the coefficient of determination**

The *r2_score* function computes the coefficient of determination, usually denoted as R^2 . It represents the proportion of variance (of y) that has been explained by the independent variables in the model. As such variance is dataset dependent, R^2 may not be meaningfully comparable across different datasets. Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected (average) value of y , disregarding the input features, would get an R^2 score of 0.0. If \hat{y}_i is the predicted value of the i -th sample and y_i is the corresponding true value for total n samples, the estimated R^2 is defined as:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

where $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ and $\sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n \epsilon_i^2$

2. **Mean absolute error**

The *mean_absolute_error* function computes mean absolute error. If \hat{y}_i is the predicted value of the i -th sample and y_i is the corresponding true value then the mean absolute error (MAE) estimated over $n_{samples}$ is defined as

$$MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|.$$

3. **Mean squared error**

The *mean_squared_error* function computes mean squared error. If \hat{y}_i is the predicted value of the i -th sample and y_i is the corresponding true value then the mean absolute error (MSE) estimated over $n_{samples}$ is defined as

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2.$$

4. **Mean squared logarithmic error** The *mean_squared_log_error* function computes a risk metric corresponding to the expected value of the squared logarithmic (quadratic) error or loss. If \hat{y}_i is the predicted value of the i -th sample and y_i is the corresponding true value then the mean square logarithmic error (MSLE) estimated over $n_{samples}$ is defined as

$$MSLE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (\log_e(1 + y_i) - \log_e(1 + \hat{y}_i))^2.$$

Where $\log_e(x)$ means the natural logarithm of x . This metric is best to use when targets having exponential growth

5. **Median absolute error**

The *median_absolute_error* is particularly interesting because it is robust to outliers. The loss is calculated by taking the median of all absolute differences between the target and the prediction. If \hat{y}_i is the predicted value of the i -th sample and y_i is the corresponding true value then the median absolute error

(MedAE) estimated over $n_{samples}$ is defined as

$$\text{MedAE}(y, \hat{y}) = \text{median}(|y_1 - \hat{y}_1|, \dots, |y_n - \hat{y}_n|).$$

The *median_absolute_error* does not support multioutput.

6. Explained variance score

The *explained_variance_score* computes the explained variance regression score. If \hat{y}_i is the estimated target output, y the corresponding (correct) target output, and Var is Variance, the square of the standard deviation, then the explained variance is estimated as follow:

$$\text{explained_variance}(y, \hat{y}) = 1 - \frac{\text{Var}\{y - \hat{y}\}}{\text{Var}\{y\}}$$

The best possible score is 1.0, lower values are worse.

Descriptive Statistics for Numeric Variables

	Years of Experience	Salary
count	30.0	30.0
mean	5.31	76003.0
std	2.84	27414.43
min	1.1	37731.0
25%	3.2	56720.75
50%	4.7	65237.0
75%	7.7	100544.75
max	10.5	122391.0

Figure 6.23: Summary of Numeric Variables

Performance Metrics

Metric	Value
Intercept	[68405.59090909]
Coefficient	[[24066.14073741]]
R Square	0.9113901944610713
Mean Squared Error	31182593.457186192
Explained Variance	0.9115844526093295
Mean Squared Log Error	0.004091202269707744
Mean Absolute Error	4736.651463085261
Median Absolute Error	4406.336497248121

Figure 6.24: Parameters and Metrics of the Data

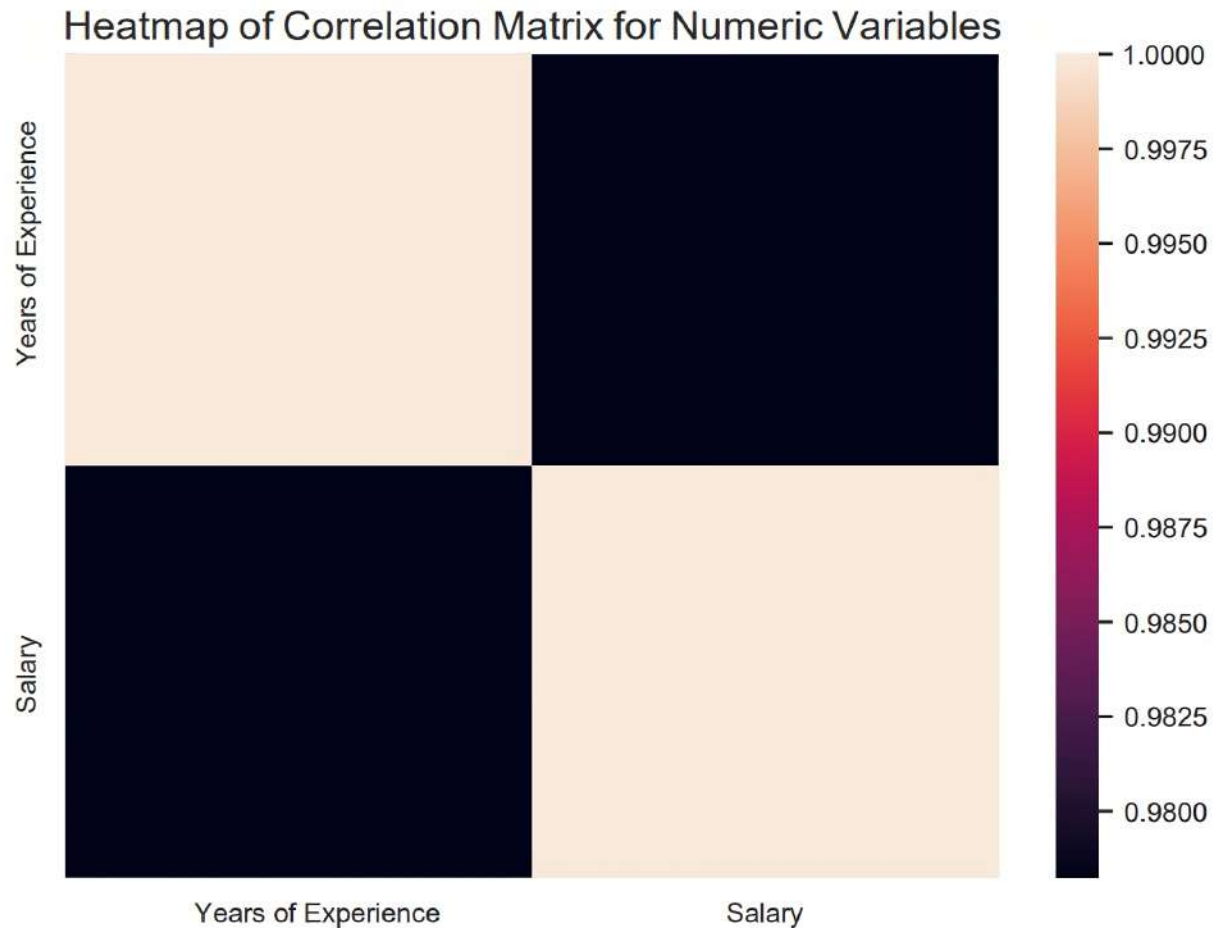


Figure 6.25: Correlation Matrix of the Numeric variables

6.2.6 Lasso Regression

- **About Dataset :**

This Ice Cream Revenue dataset is a collection of information that provides insights into the revenue generated by an ice cream business over a certain period of time. The purpose of this dataset is to analyze the relationship between temperature and ice cream revenue and potentially make predictions or draw insights. The dataset likely includes two variables or columns and 500 rows. The variables are

- **Temperature:** This variable represents the recorded temperature during a specific time period or location. It may be measured in Celsius .
- **Revenue:** This variable represents the generated revenue from ice cream sales during the corresponding temperature period. It is typically measured in a specific currency, such as dollars.

- **Parameters :** These are the terms which are important while running the given model.

1. **max iter : int, default= 1000**

The maximum number of iterations.

2. **alpha float, default= 1.0**

Constant that multiplies the $L1$ term, controlling regularization strength. alpha must be a non-negative float i.e. in $[0, \text{inf})$.

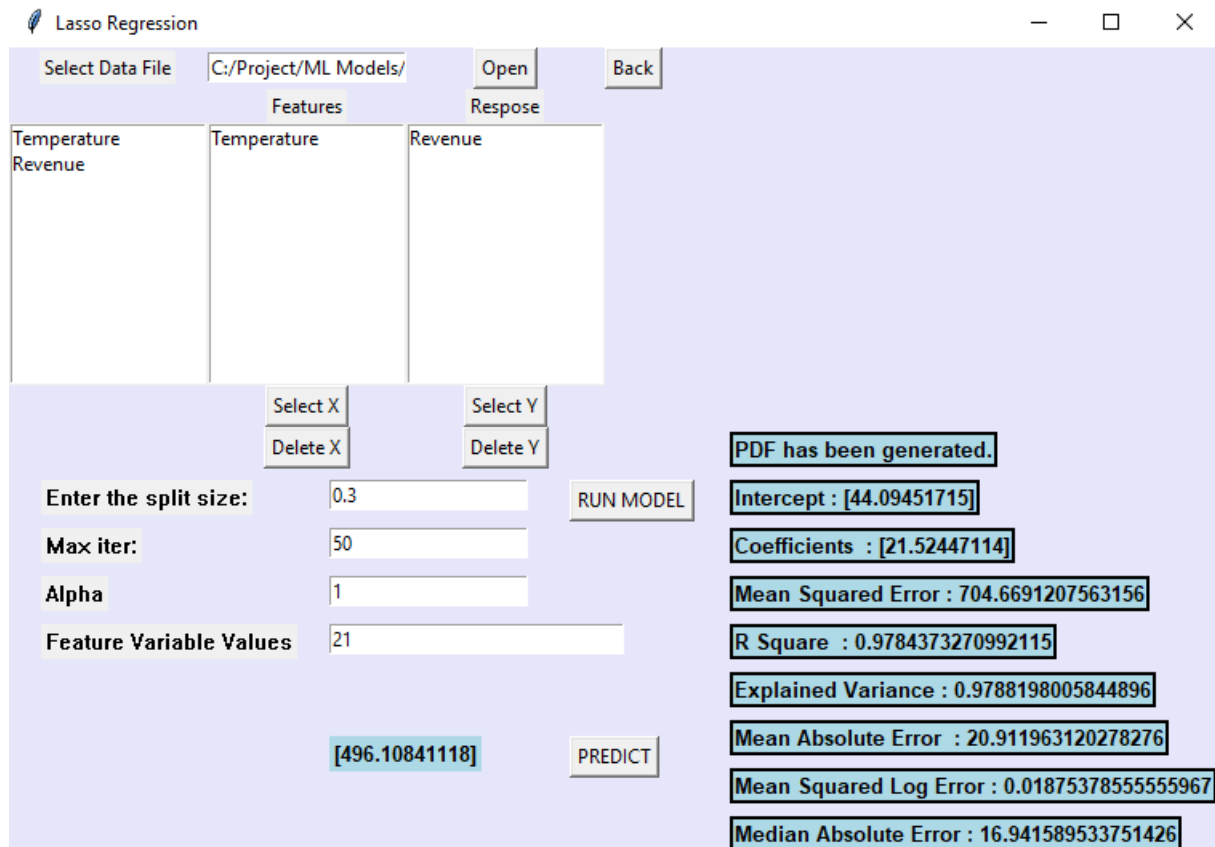


Figure 6.26: Lasso Regression GUI Window

- **PDF Output**

Descriptive Statistics for Numeric Variables

	Temperature	Revenue
count	500.0	500.0
mean	22.23	521.57
std	8.1	175.4
min	0.0	10.0
25%	17.12	405.56
50%	22.39	529.37
75%	27.74	642.26
max	45.0	1000.0

Figure 6.27: Summary of Numeric Variables

Performance Metrics

Metric	Value
Intercept	[44.09451715]
Coefficient	[21.52447114]
R Square	0.9784373270992115
Mean Squared Error	704.6691207563156
Explained Variance	0.9788198005844896
Mean Squared Log Error	0.01875378555555967
Mean Absolute Error	20.911963120278276
Median Absolute Error	16.9421589533751426

Figure 6.28: Parameters and Metrics of the Data

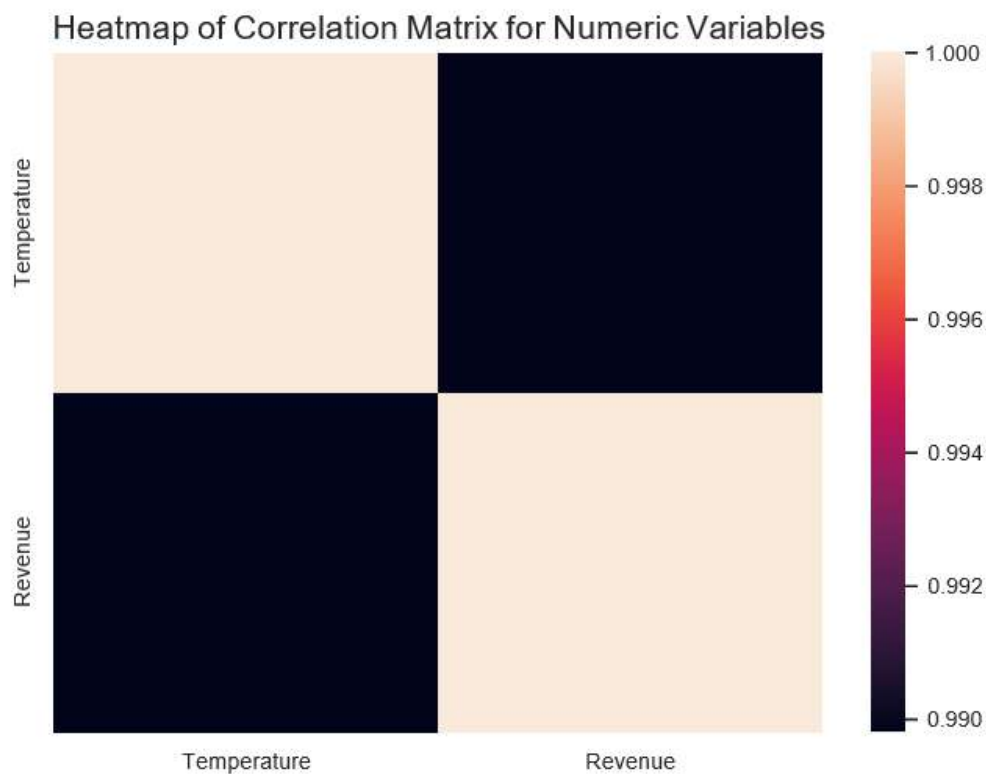


Figure 6.29: Correlation Matrix of the Numeric variables

- **Role of alpha**

To better understand the role of alpha, we plot the lasso coefficients as a function of alpha (max_iter are the maximum number of iterations). Remember that if $\alpha = 0$, then the lasso gives the least squares fit, and when alpha becomes very large, the lasso gives the null model in which all coefficient estimates equal zero.

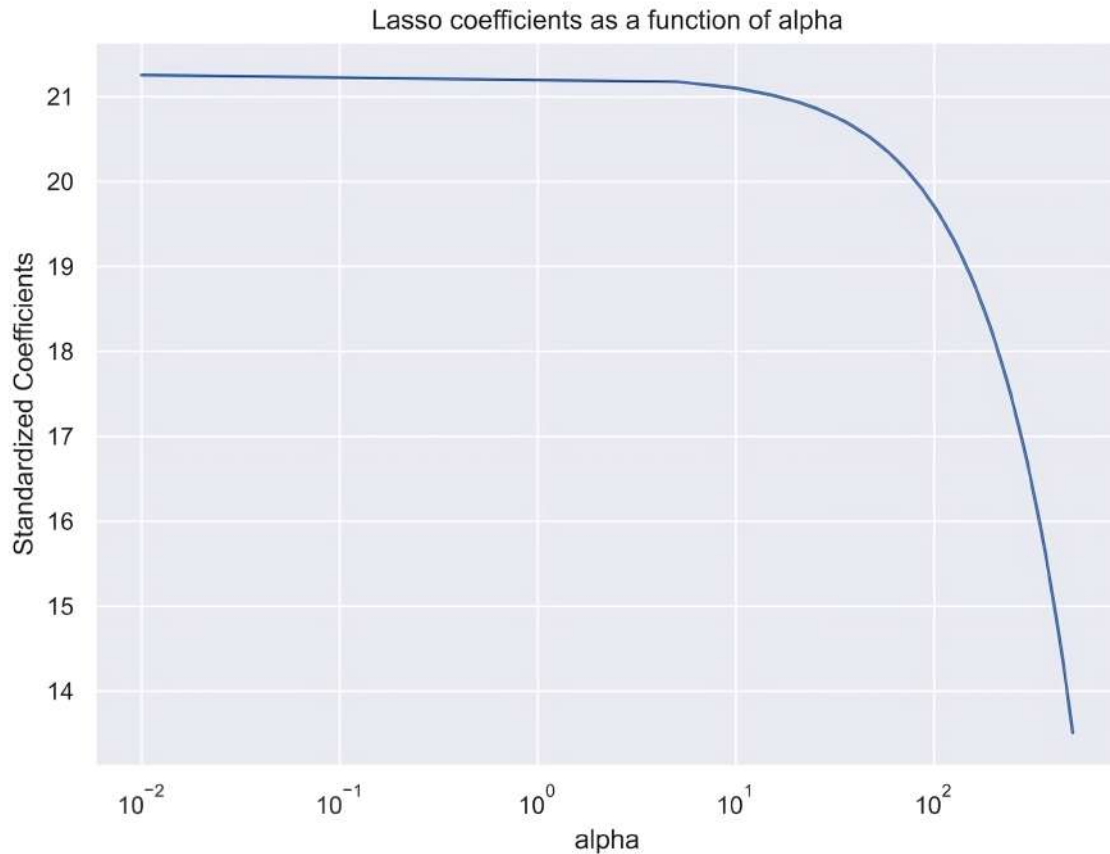


Figure 6.30: Role of alpha

6.2.7 Support Vector Regression

- **About Dataset :**

Here we use same dataset which is used in previous example. i.e. Ice Cream Revenue dataset

- **Parameters :** These are the terms which are important while running the given model .

1. **kernel 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or callable, default='rbf'**
Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used. If a callable is given it is used to precompute the kernel matrix.
2. **epsilon float, default= 0.1**
Epsilon in the epsilon-SVR model. It specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value. Must be non-negative.

Figure 6.31: Support Vector Regression GUI Window

- **PDF Output**

Descriptive Statistics for Numeric Variables

	Temperature	Revenue
count	500.0	500.0
mean	22.23	521.57
std	8.1	175.4
min	0.0	10.0
25%	17.12	405.56
50%	22.39	529.37
75%	27.74	642.26
max	45.0	1000.0

Figure 6.32: Summary of Numeric Variables

Performance Metrics	
Metric	Value
Intercept	[48.977386]
Coefficient	[[21.25302641]]
R Square	0.9800026691155783
Mean Squared Error	0.9800026691155783
Explained Variance	0.9800040897103236
Mean Squared Log Error	0.020594922183365897
Mean Absolute Error	21.567008412069335
Median Absolute Error	21.567008412069335

Figure 6.33: Parameters and Metrics of the Data

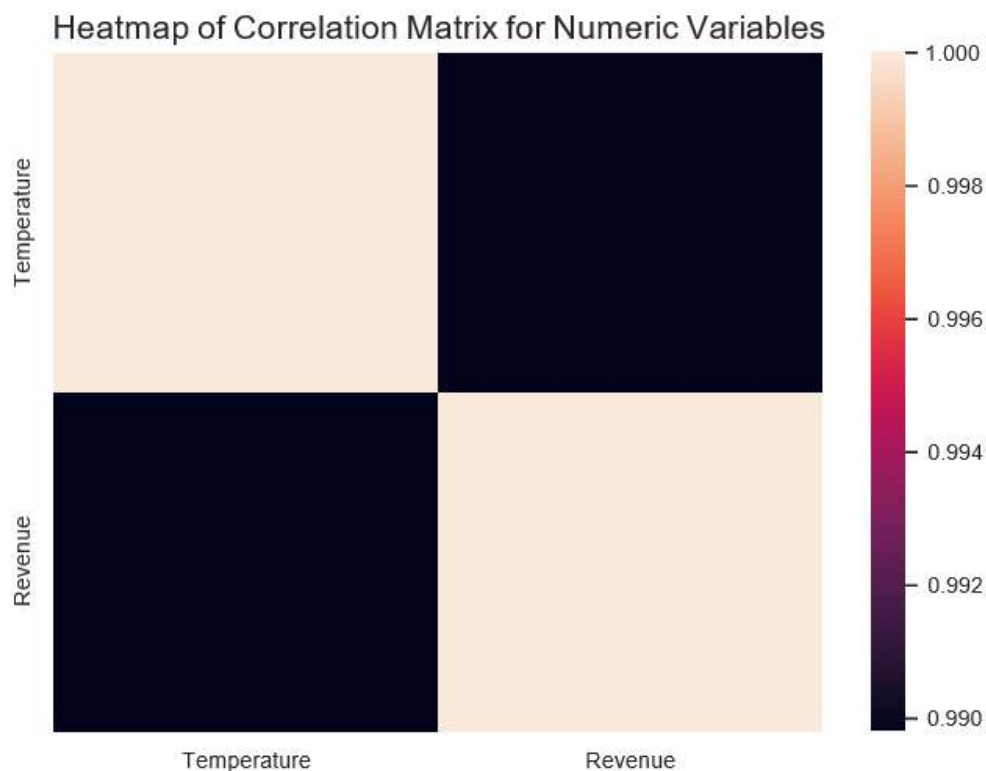


Figure 6.34: Correlation Matrix of the Numeric variables

- **The fitted model graph of Support Vector Regression:**

First we will fit a Support vector Regression model using a linear kernel.

1. Data Points :

The actual data points from the test set are plotted on the graph. Each data point represents an instance with its corresponding input features and target output values.

2. Regression Line :

The regression line represents the best-fit line determined by the Support Vector

Regression(SVR) algorithm. It is the hyperplane that passes through the middle of the margin and minimizes the errors between the predicted values and the actual target values.

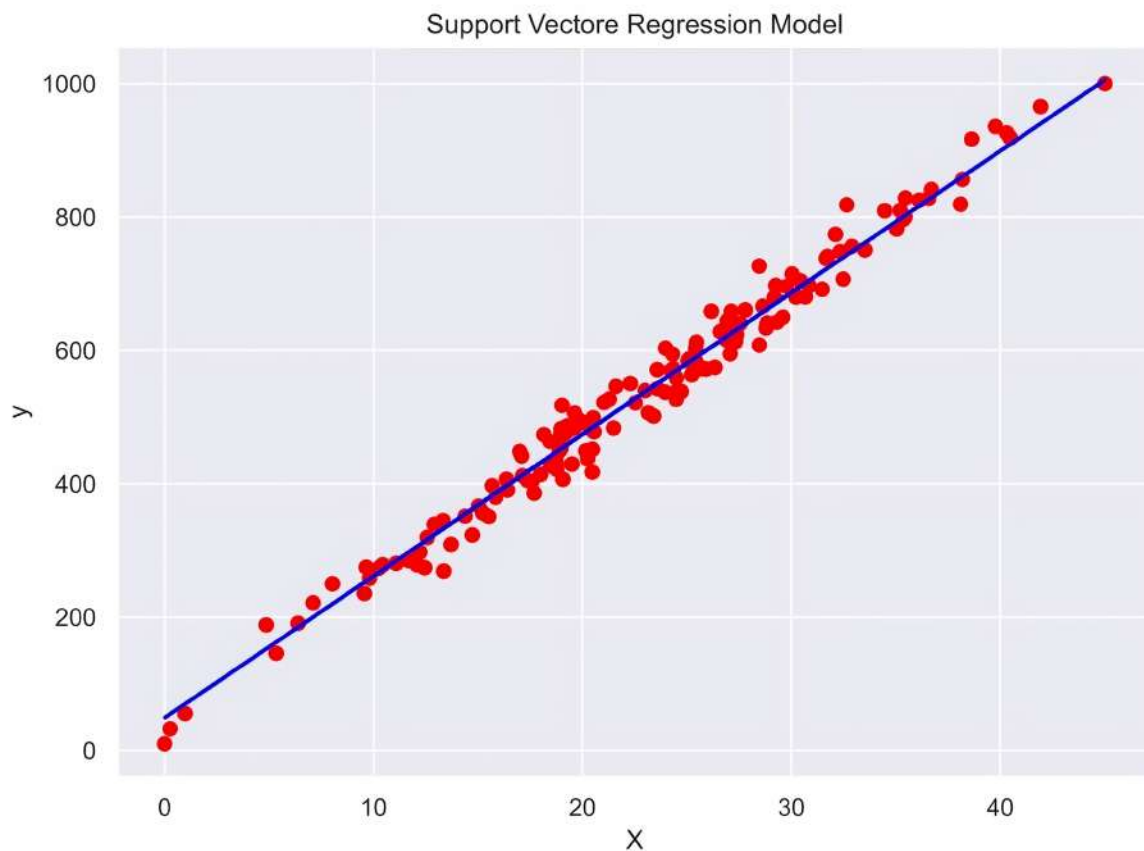


Figure 6.35: Support Vector Model

6.2.8 Polynomial Regression

- **About Dataset :**

Here we use same dataset which is used in previous example. i.e. Ice Cream Revenue dataset

- **Parameters :** These are the terms which are important while running the given model

.

1. **degree int or tuple (min_degree, max_degree), default=2**

If a single int is given, it specifies the maximal degree of the polynomial features. If a tuple (min_degree, max_degree) is passed, then min_degree is the minimum and max_degree is the maximum polynomial degree of the generated features. Note that min_degree and max_degree are equivalent as outputting the degree zero term is determined by include_bias.

Polynomial Regression

Select Data File: C:/Project/ML Models/ Open Back

Features: Temperature Revenue

Response: Revenue

Select X Select Y
Delete X Delete Y

Enter the split size: 0.3 RUN MODEL

Degree: 2

Feature Variable Values: 21,23,20

[[523.91642411]] PREDICT

PDF has been generated.

Intercept : [58.33395593]

Coefficients : [[0. 20.22146106 0.02444319]]

Mean Squared Error : 643.5212360722147

R Square : 0.9783906929541418

Explained Variance : 0.9783907459492891

Mean Absolute Error : 20.24730828822372

Mean Squared Log Error : 0.022787155221839238

Median Absolute Error : 16.77248195914035

Figure 6.36: Polynomial Regression GUI Window

- PDF Output

Descriptive Statistics for Numeric Variables

	Temperature	Revenue
count	500.0	500.0
mean	22.23	521.57
std	8.1	175.4
min	0.0	10.0
25%	17.12	405.56
50%	22.39	529.37
75%	27.74	642.26
max	45.0	1000.0

Figure 6.37: Summary of Numeric Variables

Performance Metrics

Metric	Value
Intercept	[58.3339]
Coefficient	[[0. 20.22146106 0.02444319]]
R Square	0.9783906929541418
Mean Squared Error	643.5212360722147
Explained Variance	0.9783907459492891
Mean Squared Log Error	0.022787155221839238
Mean Absolute Error	20.24730828822372
Median Absolute Error	16.7724819591403

Figure 6.38: Parameters and Metrics of the Data

- **Density plot of the difference between actual and fitted values:**

A density plot of the difference between actual and fitted values is a graphical representation that helps to visualize the distribution and patterns of the discrepancies between the predicted values and the true values in a regression model. This plot provides insights into the accuracy and performance of the model by examining the distribution and spread of the residuals.

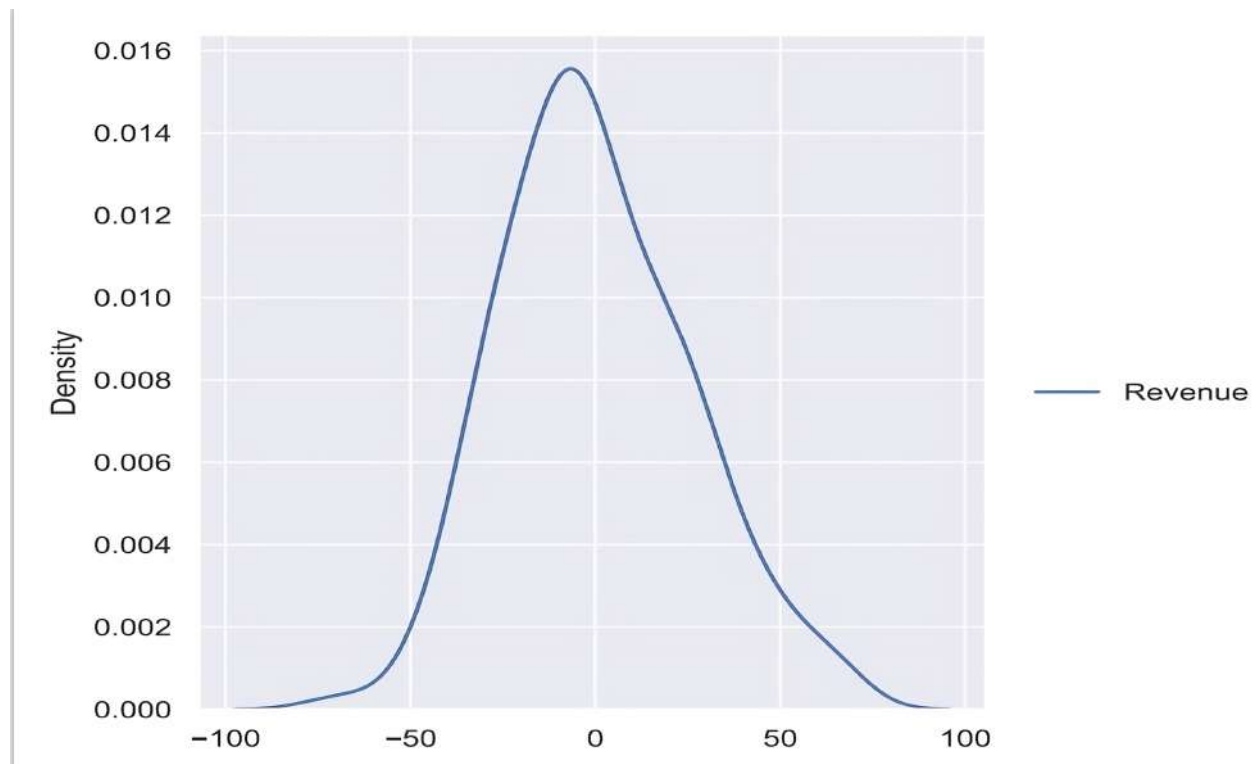


Figure 6.39: Density plot

References

- [1] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani(Second Edition) *An Introduction To Statistical Learning with applications in R.* , 2023,(Springer). Press.
- [2] <https://chat.openai.com/>
- [3] <https://docs.python.org/3/library/tkinter.html>
- [4] <https://www.youtube.com/@krishnaik06>
- [5] <https://www.youtube.com/@Codemycom>
- [6] <https://www.geeksforgeeks.org/>
- [7] <https://www.tutorialspoint.com/>

Softwares and IDE Used

1. Python
2. Jupyter Notebook

Annexure