1] General Information on datasets

1) Carbon Dioxide (Numeric Dataset Regressors)

Dataset Link: https://www.kaggle.com/datasets/debajyotipodder/co2-emission-by-vehicles/data?select=CO2+Emissions_Canada.csv

This dataset captures the details of how CO2 emissions by a vehicle can vary with the different features. The dataset has been taken from Canada Government official open data website. This is a compiled version. This contains data over a period of 7 years.

There are one dataset file:

- Carbon_dioxide.CVS with a total of 7385 rows and 12 columns, and its size is 476.09 KB.

There are few abbreviations that has been used to describe the features. I am listing them out here.

Understanding the data	

Model	4WD/4X4 = Four-wheel drive
	AWD = All-wheel drive
	FFV = Flexible-fuel vehicle
	SWB = Short wheelbase
	LWB = Long wheelbase
	EWB = Extended wheelbase

Transmission	A = automatic
	AM = automated manual
	AS = automatic with select shift
	AV = continuously variable
	M = manual
	3 - 10 = Number of gears

Fuel Type	X = regular gasoline
	Z = premium gasoline
	D = diesel
	E = ethanol (E85)
	N = natural gas

Fuel Consumption	City and highway fuel consumption ratings are shown in liters per 100 kilometers (L/100 km) - the combined rating (55% city, 45% hwy) is shown in L/100 km and in miles per imperial gallon (mpg).
CO2 emissions	The tailpipe emissions of carbon dioxide (in grams per kilometer) for combined city and highway driving.

2) Bank Marketing Response (Numeric Dataset Classifiers)

Dataset link: https://www.kaggle.com/datasets/kukuroo3/bank-marketing-response-predict?select=train.csv

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

There are two datasets:

1. train.csv with 70% of the examples (14161) and 16 inputs.
1) test.csv with 30% of the examples (16) inputs.

The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).

Variable Name	Role	Type	Description	Missing Values
age	Feature	Integer		No
job	Feature	Categorical	Type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')	No
marital	Feature	Categorical	Marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)	No
education	Feature	Categorical	(categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')	No
default	Feature	Binary	Has credit in default?	No
balance	Feature	Integer	Average yearly balance	No
housing	Feature	Binary	Has housing loan?	No
loan	Feature	Binary	Has personal loan?	No
contact	Feature	Categorical	contact communication type (categorical: 'cellular', 'telephone')	Yes
day_of_week	Feature	Date	last contact day of the week	
month	Feature	Date	Last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')	No
duration	Feature	Integer	Last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.	No
campaign	Feature	Integer	Number of contacts performed during this campaign and for this client (numeric, includes last contact)	No

pdays	Feature	Integer	Number of days that passed by after the client was last contacted from a previous campaign (numeric; -1 means client was not previously contacted)	Yes
previous	Feature	Integer	Number of contacts performed before this campaign and for this client	No
poutcome	Feature	Categorical	Outcome of the previous marketing campaign (categorical: 'failure', 'nonexistent', 'success')	Yes
y	Target	Binary	Has the client subscribed a term deposit?	No

Dataset Size:

- Overall Dataset (14161 rows x 16 columns) 1.43 MB
- Train data (12870 rows x 16 columns) 1.09 MB
- Test data (4291 rows x 16 columns) 347.61 KB

3) Mammogram Classification for Breast Cancer Detection (Image Dataset Classifiers)

Dataset Link: https://www.kaggle.com/datasets/hayder17/breast-cancer-detection

This dataset contains 3,383 mammogram images focused on breast tumors, annotated in a folder structure. The dataset was exported from Roboflow, a platform for computer vision projects. It is ideal for building and testing Deep-learning models aimed at detecting breast tumors through mammograms.

There are two datasets:

1) **Test directory**
   a) Consists of two directories: 0 directory and 1 directory.
   b) File Types: Images in JPG/PNG format Total Images: 336 images Annotations: Tumor annotations are provided in a folder structure. Image Dimensions: All images are resized to 640 x 640 pixels.
   c) 0 directory has 208 images and 1 directory has 128 images.
2) **Train directory**
   a) Consists of two directories: 0 directory and 1 directory.
   b) File Types: Images in JPG/PNG format Total Images: 336 images Annotations: Tumor annotations are provided in a folder structure. Image Dimensions: All images are resized to 640 x 640 pixels.
   c) 0 directory has 1569 images and 1 directory has 803 images.

2] Implementation details

1) Carbon Dioxide (Numeric Dataset Regressors)

I. Imports and Setup:

The notebook imports essential libraries such as **numpy** and **pandas** (numerical operations and data manipulation), **matplotlib** and **Seaborn** (visualization), and **sklearn** (machine learning).

II. Dataset Loading:

The dataset is loaded into a pandas DataFrame uing the **read_cvs** function.

III. Preprocessing:

- **Handling Missing Values**

Null values in the columns 'Fuel type' and 'Fuel Consumption' are filled using the most frequent value (Since it's a categorical column) and median value (Since it's a numeric column) respectively.

- **Encoding Non-Numeric Values**

Non-numeric (categorical) columns are encoded into numeric form using **LabelEncoder**.

- **Inspecting the Dataset**

The analysis suggests the data is approximately normally distributed (mean and median are close), indicating fewer potential outliers.

- **Dataset Splitting**

Features (X): All columns except the target variable. Target (y): The last column. The dataset is split into training and testing sets: 80% Training Data and 20% Testing Data. Random state or any constant value ensures reproducibility.

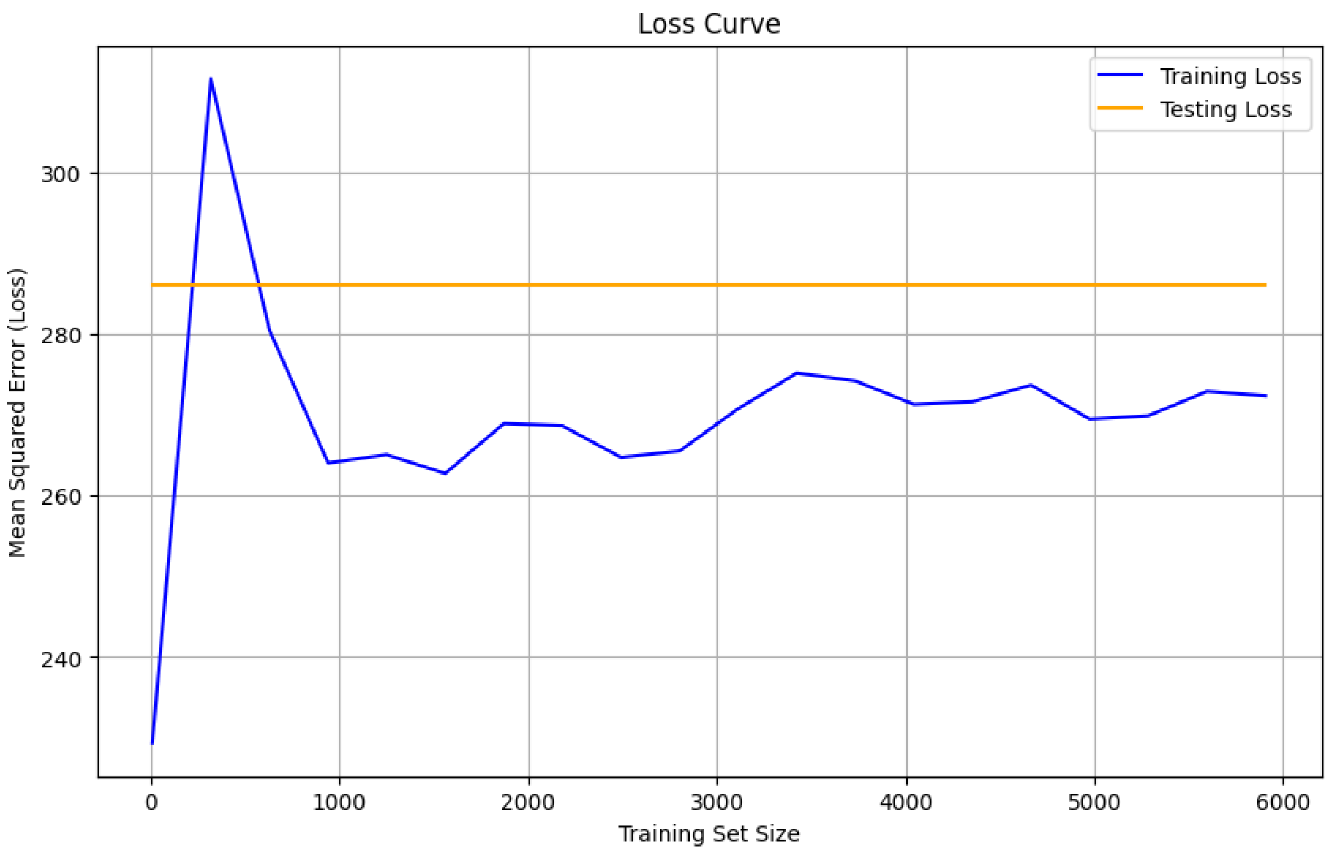
IV. Model Training using Linear Regression:

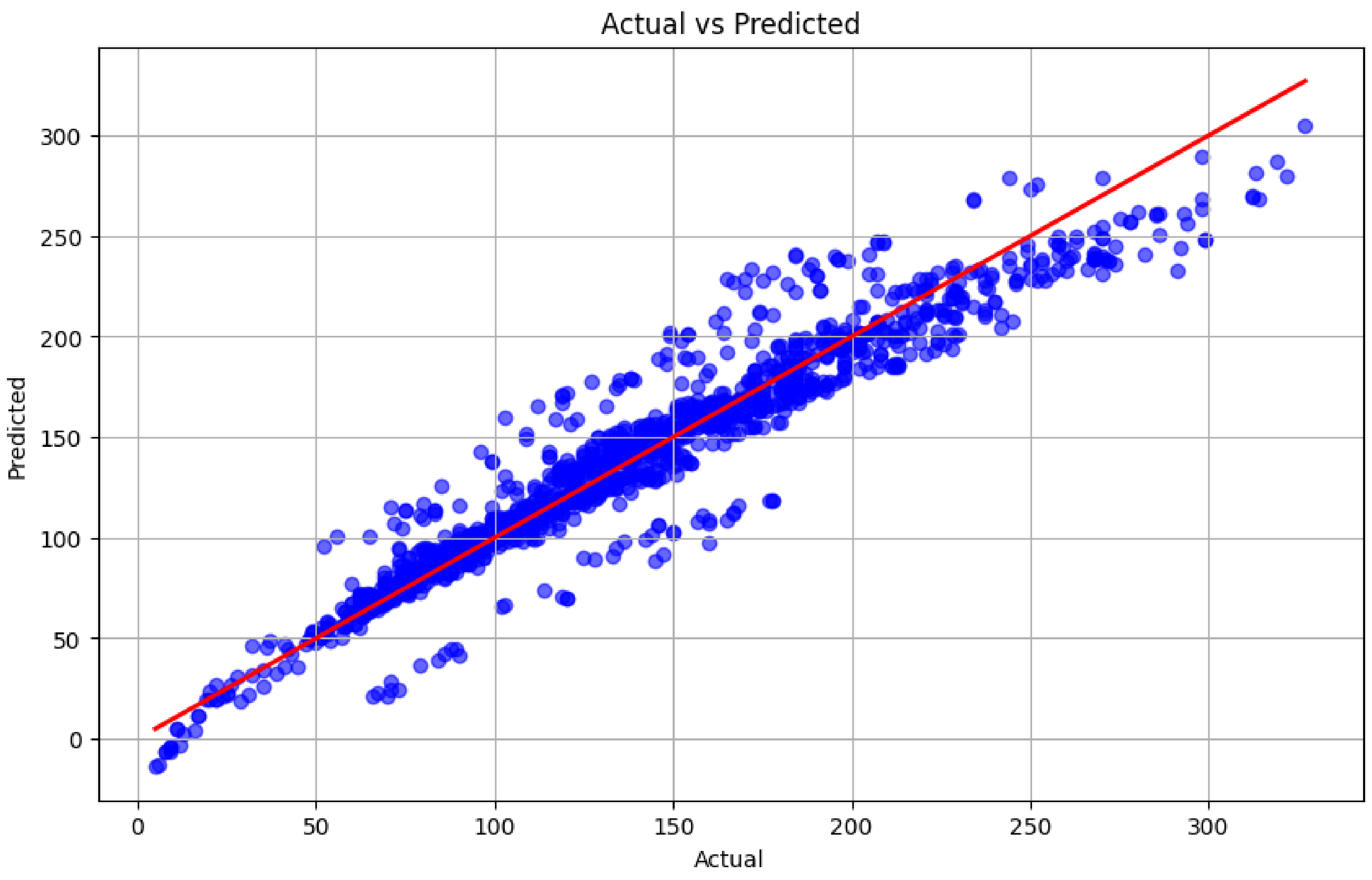
The model is trained on the training. During this step, the algorithm determines the optimal weights for the linear regression equation. The model predicts target values for the test dataset based on the learned coefficients.

V. Linear Regression Model Evaluation:

Mean Squared Error (MSE): 286

$\boldsymbol{R^2}$ **Score:** 91%



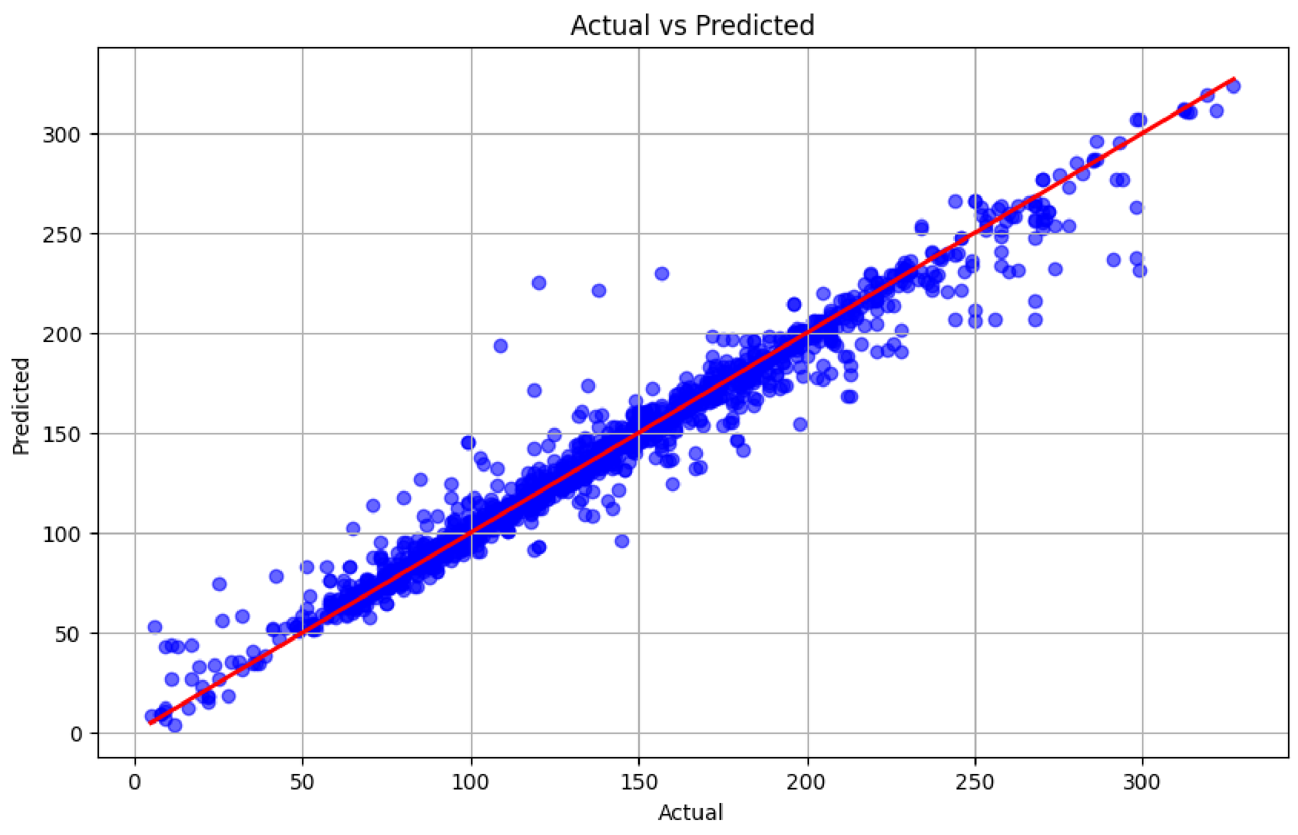


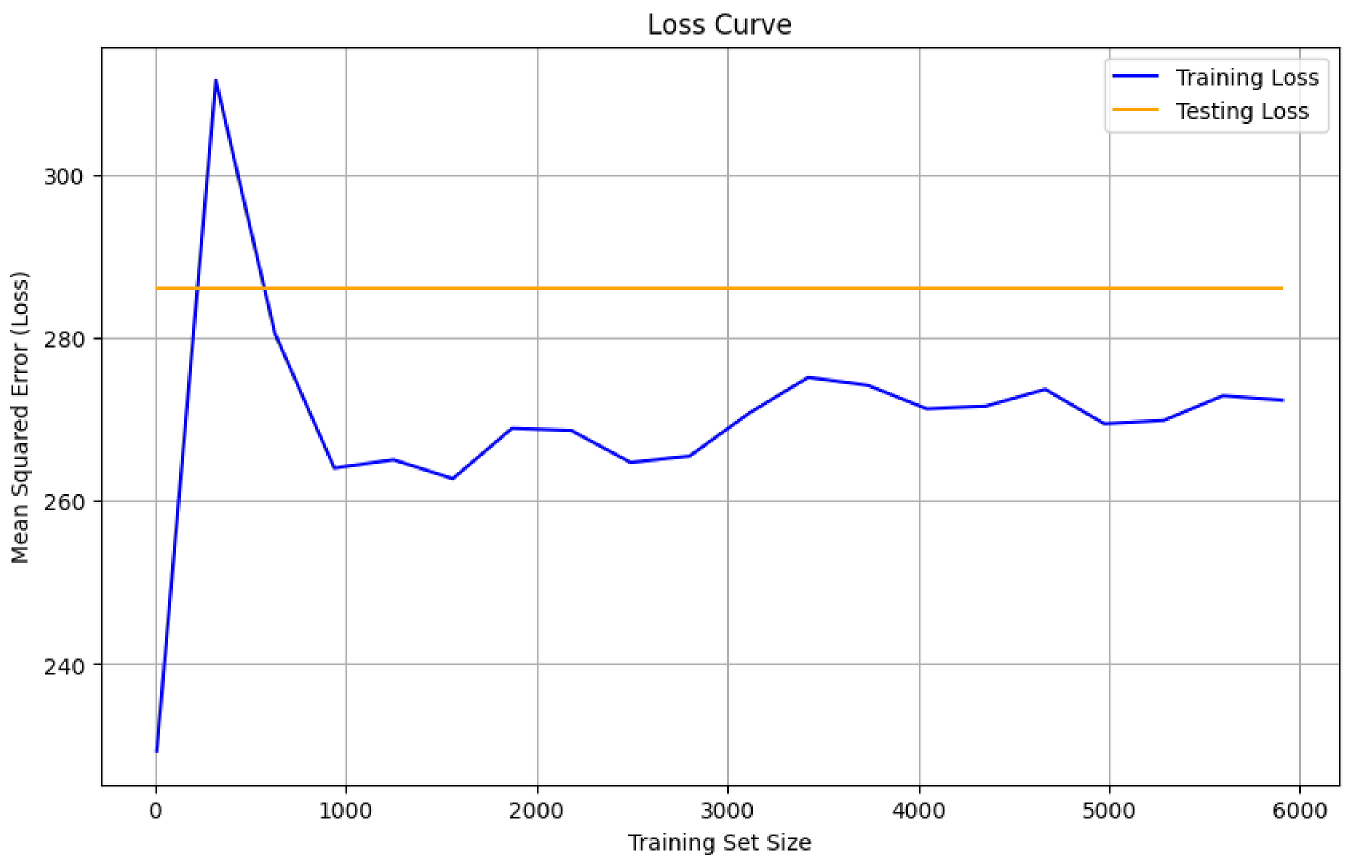
VI. Model Training using KNN classifier:

The K-Nearest Neighbors (KNN) classifier is a non-parametric, instance-based learning algorithm that classifies data points based on the majority label of their k nearest neighbors in the feature space. For a given input, the algorithm calculates distances to all training points, selects the k closest points, and assigns the majority label among these neighbors as the prediction. With k set to 5 in our case, the model considers the 5 nearest neighbors during classification. KNN is a lazy learning algorithm, meaning it does not build an explicit model; instead, the **fit** method stores the training data for efficient distance computations during predictions.

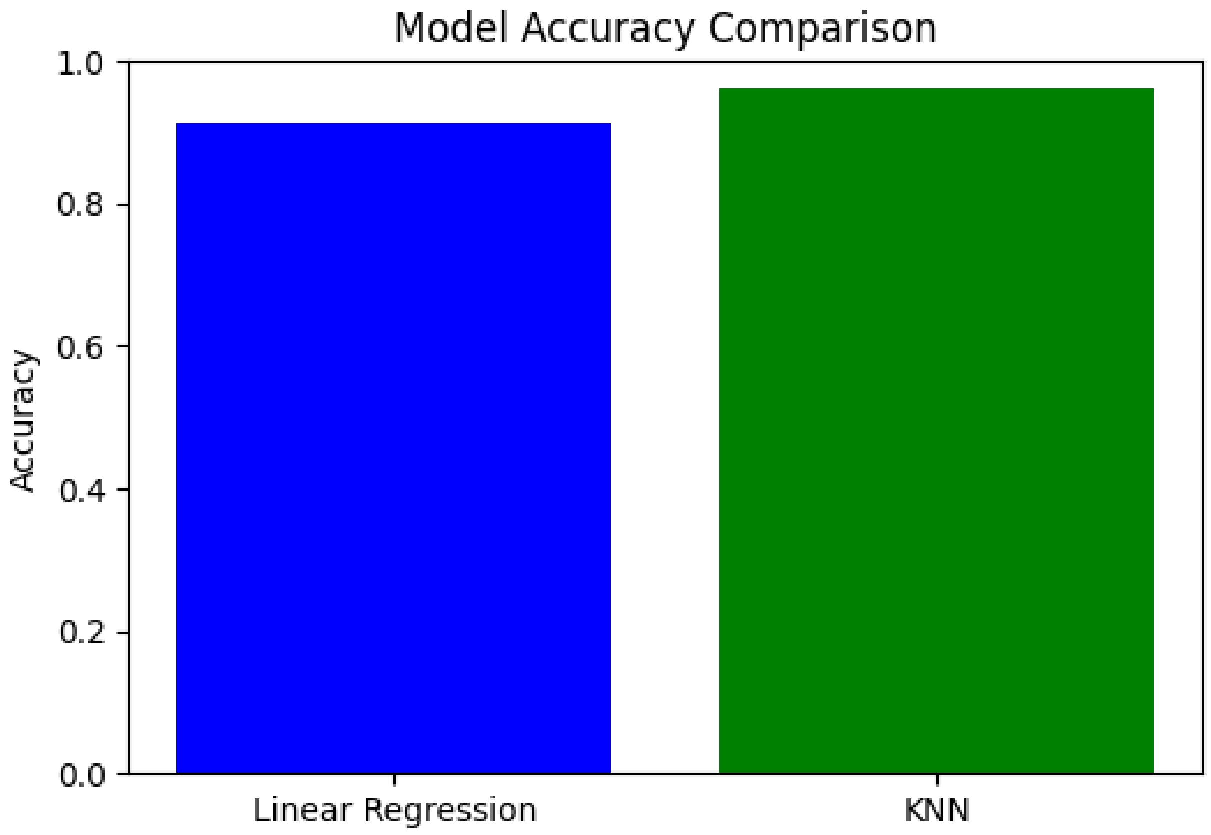
VII. KNN Model Evaluation:

Model Accuracy: 96%


Actual vs Predicted
Predicted
Actual
0
50
100
150
200
250
300


Loss Curve
Training Loss
Testing Loss
Mean Squared Error (Loss)
Training Set Size
240
260
280
300
0
1000
2000
3000
4000
5000
6000

VIII. Comparison between two models:



2) Bank Marketing Response (Numeric Dataset Classifiers)

I. Imports and Setup:

The notebook imports essential libraries such as **numpy** and **pandas** (numerical operations and data manipulation), **matplotlib** and **Seaborn** (visualization), and **sklearn** (machine learning).

II. Dataset Loading:

The dataset is loaded into a pandas DataFrame uing the **read_cvs** function.

III. Preprocessing:

- **Handling Missing Values**

Missing values in categorical features were imputed using the most frequent value (mode). Missing values in numerical features were imputed using the median value.

- **Encoding Categorical Features**

Categorical columns were transformed into numerical representations using Label Encoding for both training and testing datasets.

- **Outlier Detection and Removal**

Outliers in the balance column were identified by focusing on the top 5% and bottom 5% of values using percentile thresholds. Rows with extreme values were removed to ensure cleaner data for modeling. Visualizations, such as KDE plots and box plots, were used to inspect the distribution and confirm the presence of outliers.

Feature Dropping

The default column was removed, as it was likely redundant or unnecessary for modeling. The previous column, dominated by zero values, was dropped due to low information contribution. The poutcome column, with a majority of unknown values, was also dropped as it provided minimal meaningful information.

Exploratory Data Analysis (EDA)

Count plots were used to analyze the distribution of categorical variables like previous and poutcome. Statistical summaries and visualizations such as KDE and box plots provided insights into the distributions of numerical features like balance.

IX. Model Training using Linear Regression:

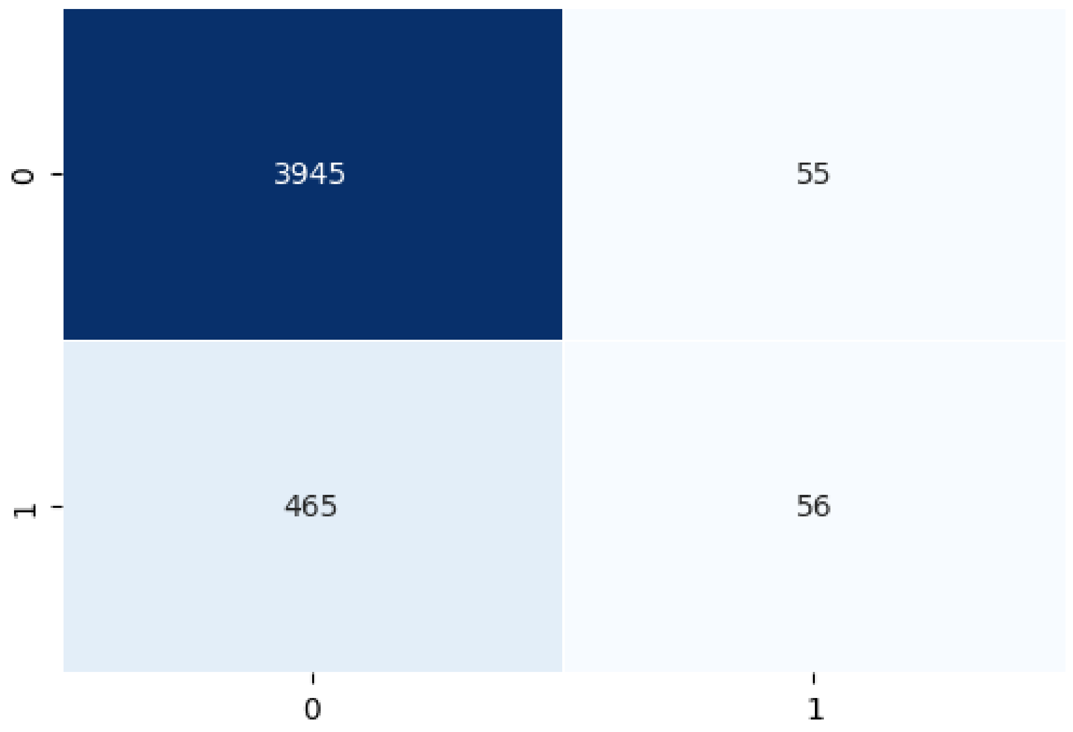
The model is trained on the training. During this step, the algorithm determines the optimal weights for the linear regression equation. The model predicts target values for the test dataset based on the learned coefficients.

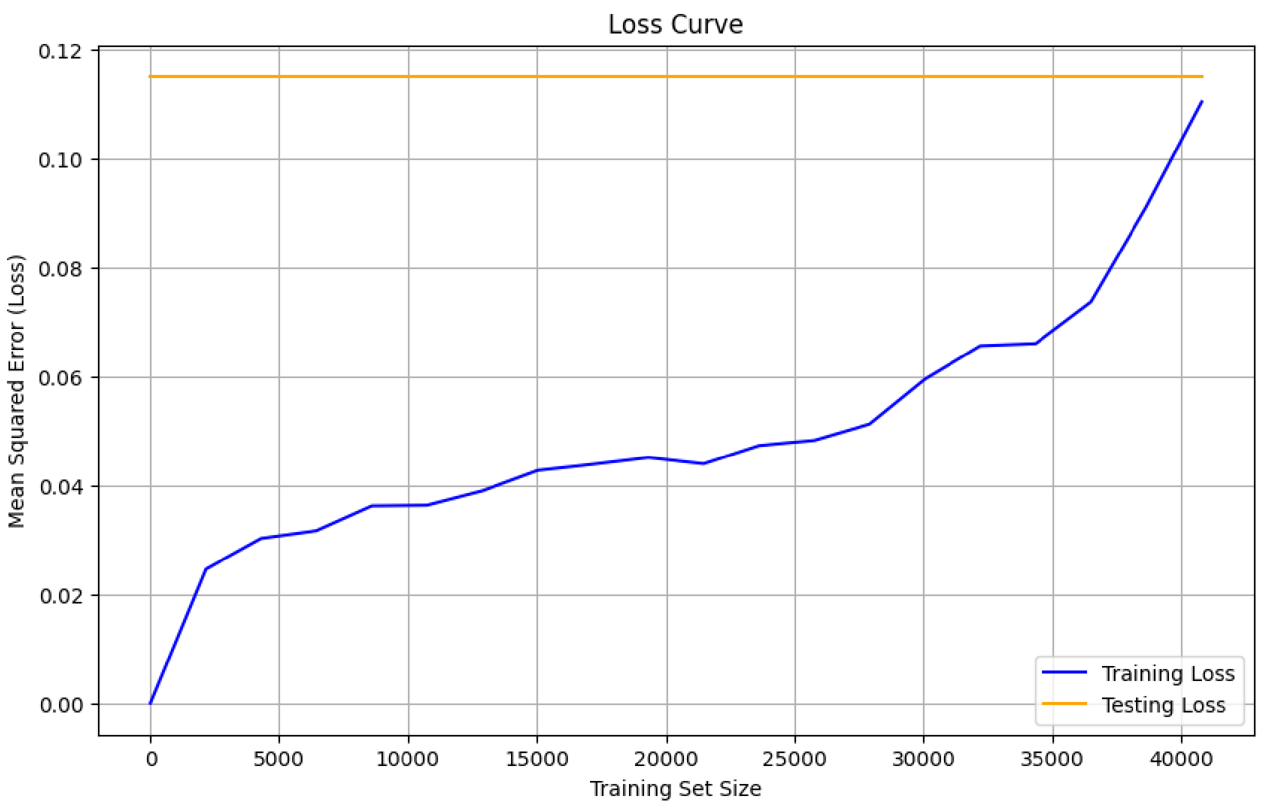
X. Model Training using Logistic Regression:

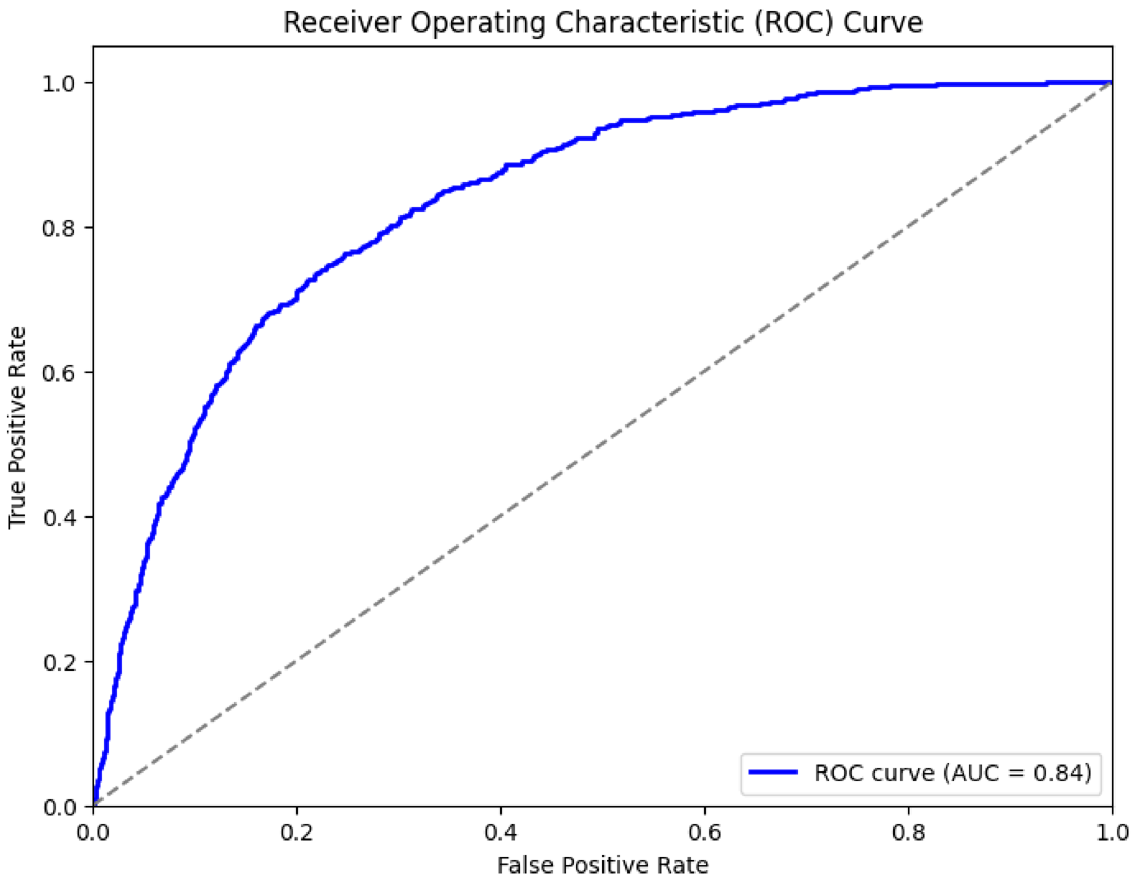
Logistic regression is a classification algorithm designed to predict probabilities for binary outcomes (0 or 1) using the logistic function. The model is trained on normalized training images and their corresponding labels using the **fit** function. During the training process, it optimizes coefficients by minimizing a loss function through iterative optimization. Once trained, the **predict** function is used to classify test data by calculating the probability of each class and assigning the label with the highest probability

XI. Logistic Regression Model Evaluation:

Model Accuracy: 88%




Loss Curve
Mean Squared Error (Loss)
Training Set Size
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
5000
10000
15000
20000
25000
30000
35000
40000
Training Loss
Testing Loss

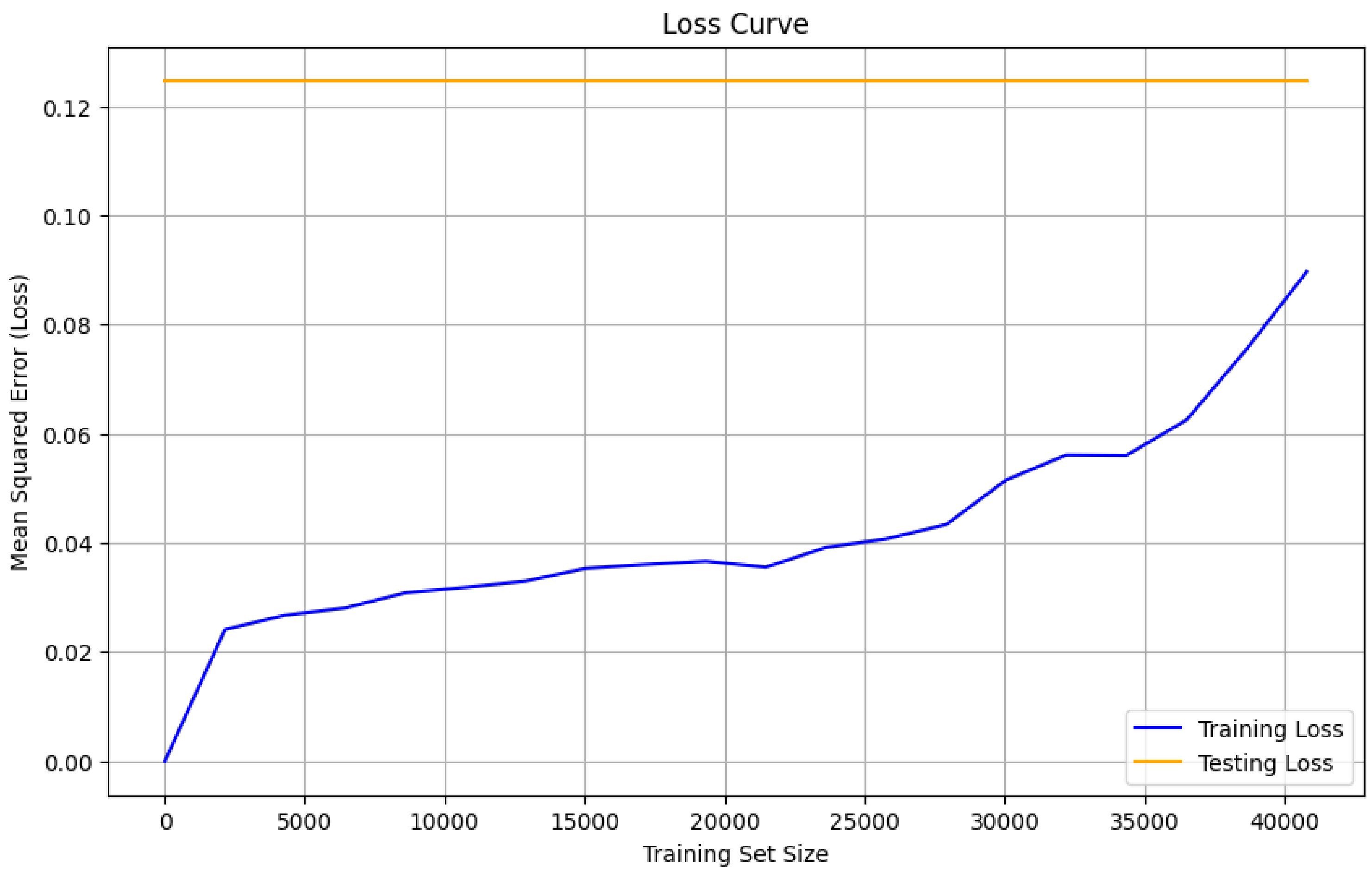

Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.84)

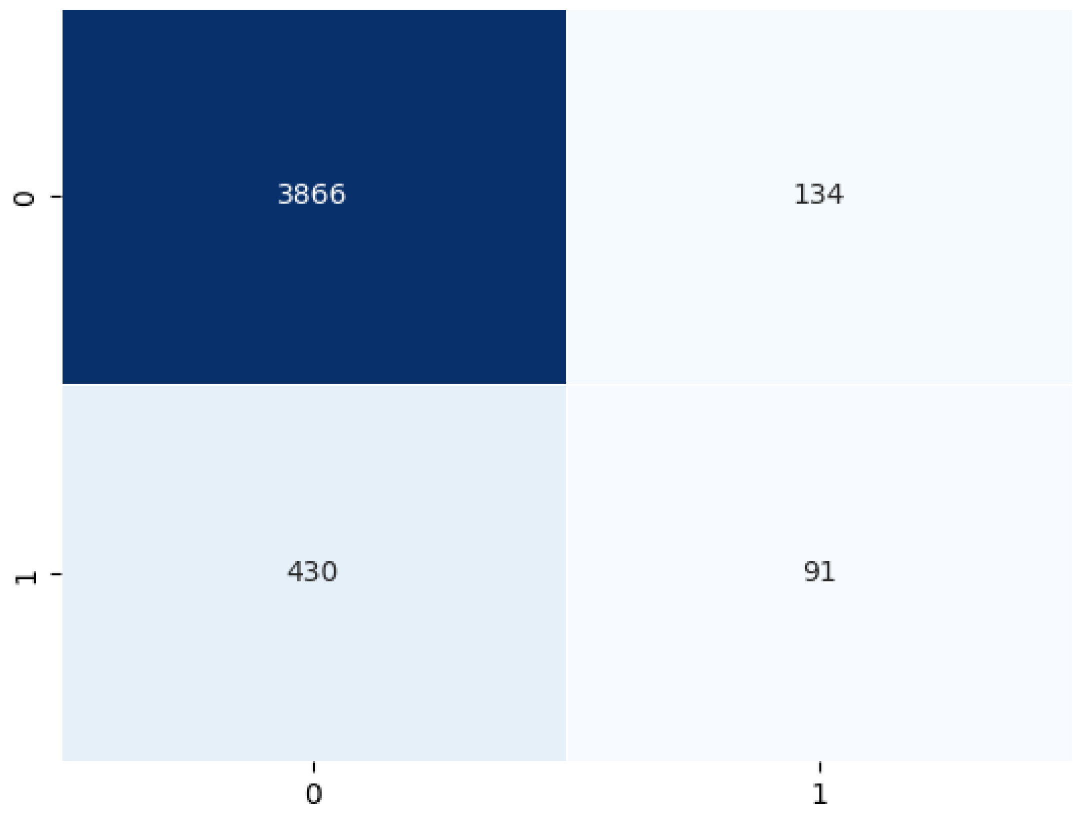
XII. Model Training using KNN classifier:

The K-Nearest Neighbors (KNN) classifier is a non-parametric, instance-based learning algorithm that classifies data points based on the majority label of their k nearest neighbors in the feature space. For a given input, the algorithm calculates distances to all training points, selects the k closest points, and assigns the majority label among these neighbors as the prediction. With k set to 5 in our case, the model considers the 5 nearest neighbors during classification. KNN is a lazy learning algorithm, meaning it does not build an explicit model; instead, the **fit** method stores the training data for efficient distance computations during predictions.

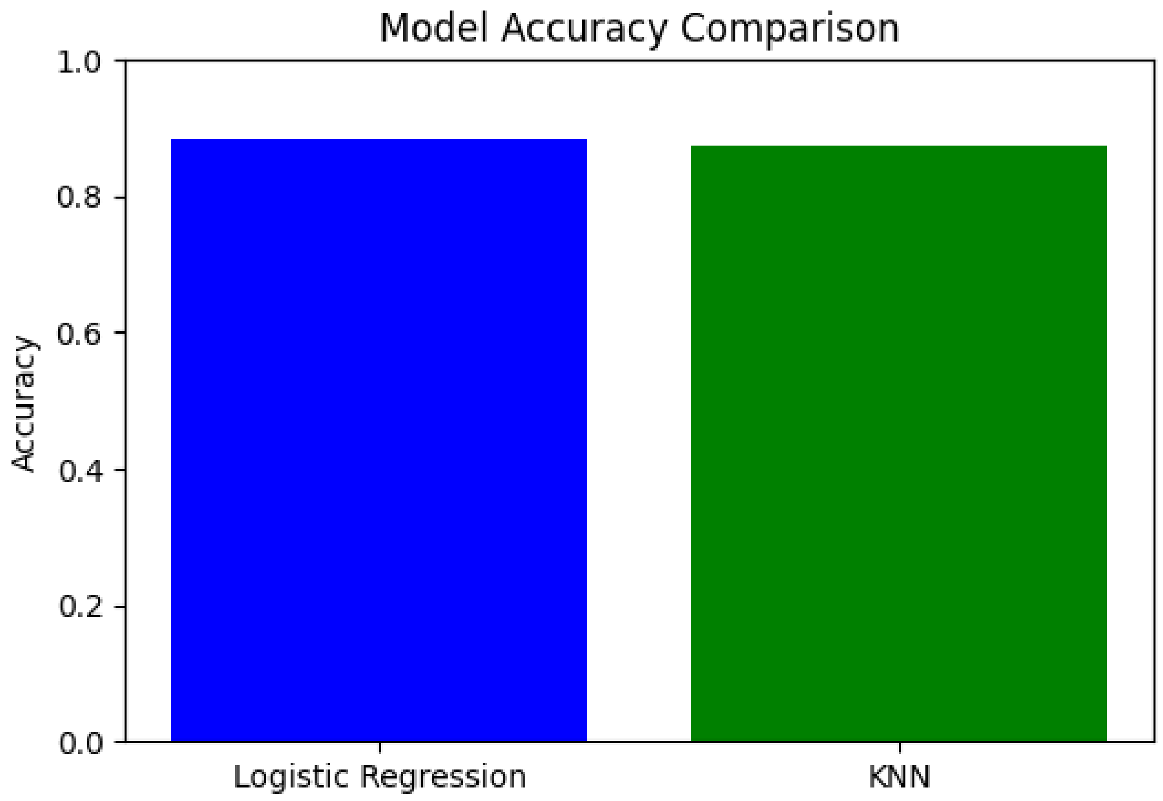
XIII. KNN Model Evaluation:

Model Accuracy: 88%





XIV. Comparison between two models:



3) Mammogram Classification for Breast Cancer Detection (Image Dataset Classifiers)

I. Imports and Setup:

The notebook imports essential libraries such as **os**, **cv2** (for image processing), **numpy** (numerical computations), **matplotlib** (visualization), and **sklearn** (machine learning).

II. Image Dataset Loading:

A function **load_images_from_folder** is defined to read images from a dataset, which appears to be structured with subfolders for labels ('0' and '1'). The dataset is loaded for training and testing using this functions. The images are resized to a fixed size (64, 64).

III. Preprocessing:

- **Normalization**

The division by 255 scales the pixel values (because there is 256 grey shades ranging from 0 to 255) to the range from 0 to 1.

- **Dimensionality Reduction with PCA (Principle Component Analysis)**

n_components = 2: This specifies that we want to reduce the data to 2 dimensions (‘0’ or ‘1’).

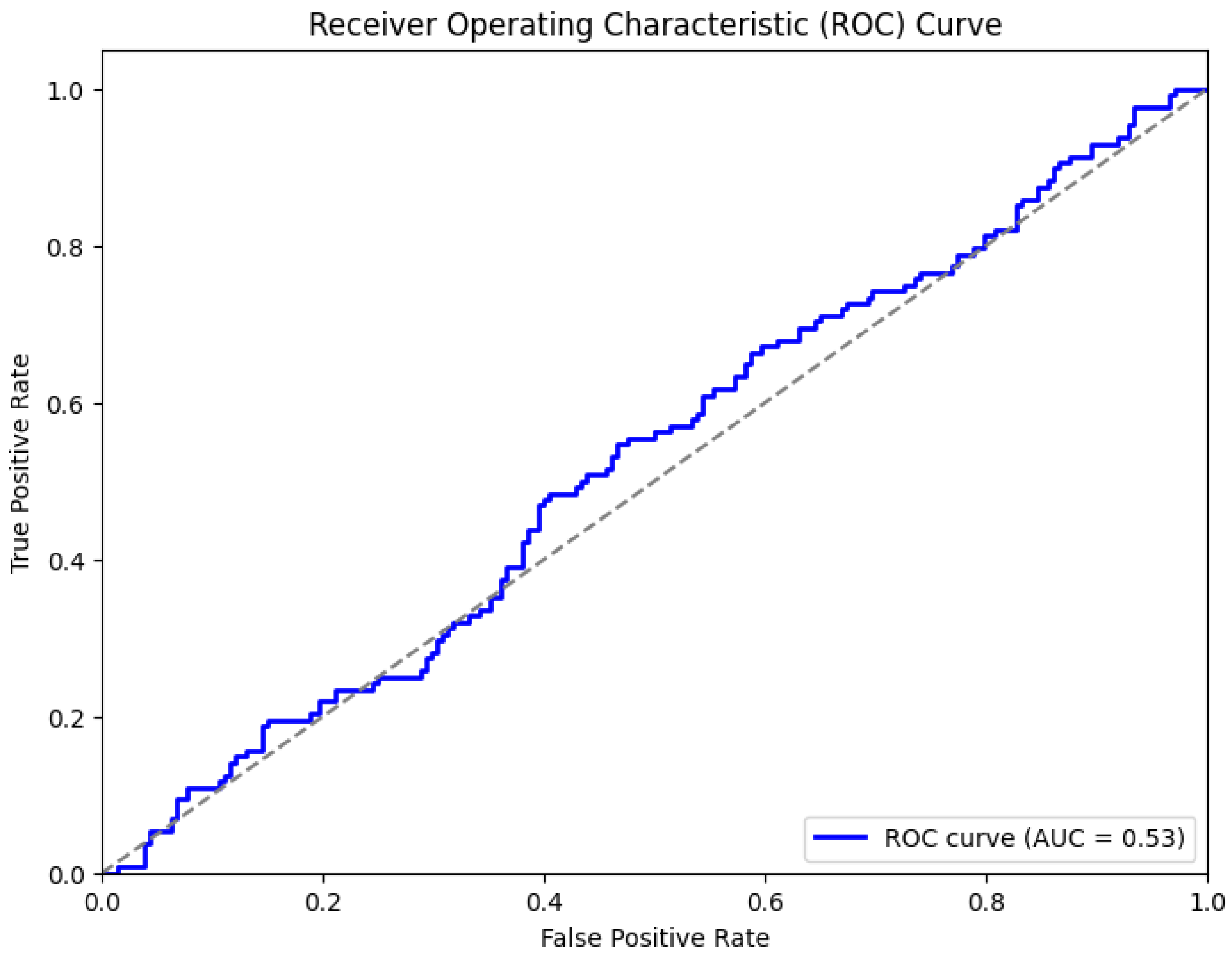
IV. Model Training using Logistic Regression:

Logistic regression is a classification algorithm designed to predict probabilities for binary outcomes (0 or 1) using the logistic function. The model is trained on normalized training images and their corresponding labels using the **fit** function. During the training process, it optimizes coefficients by minimizing a loss function through iterative optimization. Once trained, the **predict** function is used to classify test images by calculating the probability of each class and assigning the label with the highest probability.

V. Logistic Regression Model Evaluation:

Model Accuracy: 62%

Confusion Matrix: array ([[208, 0], [128, 0]])



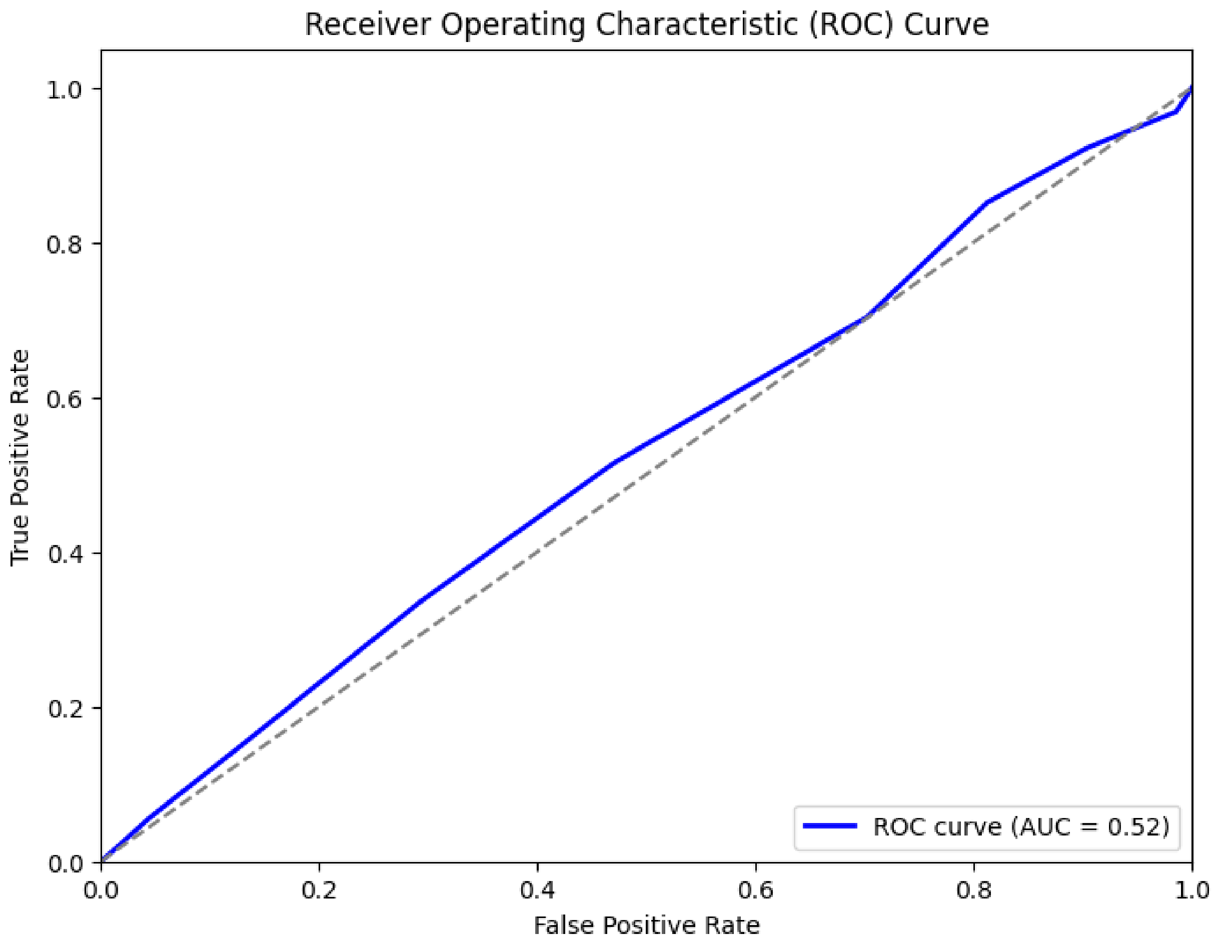
VI. Model Training using KNN classifier:

The K-Nearest Neighbors (KNN) classifier is a non-parametric, instance-based learning algorithm that classifies data points based on the majority label of their k nearest neighbors in the feature space. For a given input, the algorithm calculates distances to all training points, selects the k closest points, and assigns the majority label among these neighbors as the prediction. With k set to 15 in our case, the model considers the 15 nearest neighbors during classification. KNN is a lazy learning algorithm, meaning it does not build an explicit model; instead, the **fit** method stores the training data for efficient distance computations during predictions.

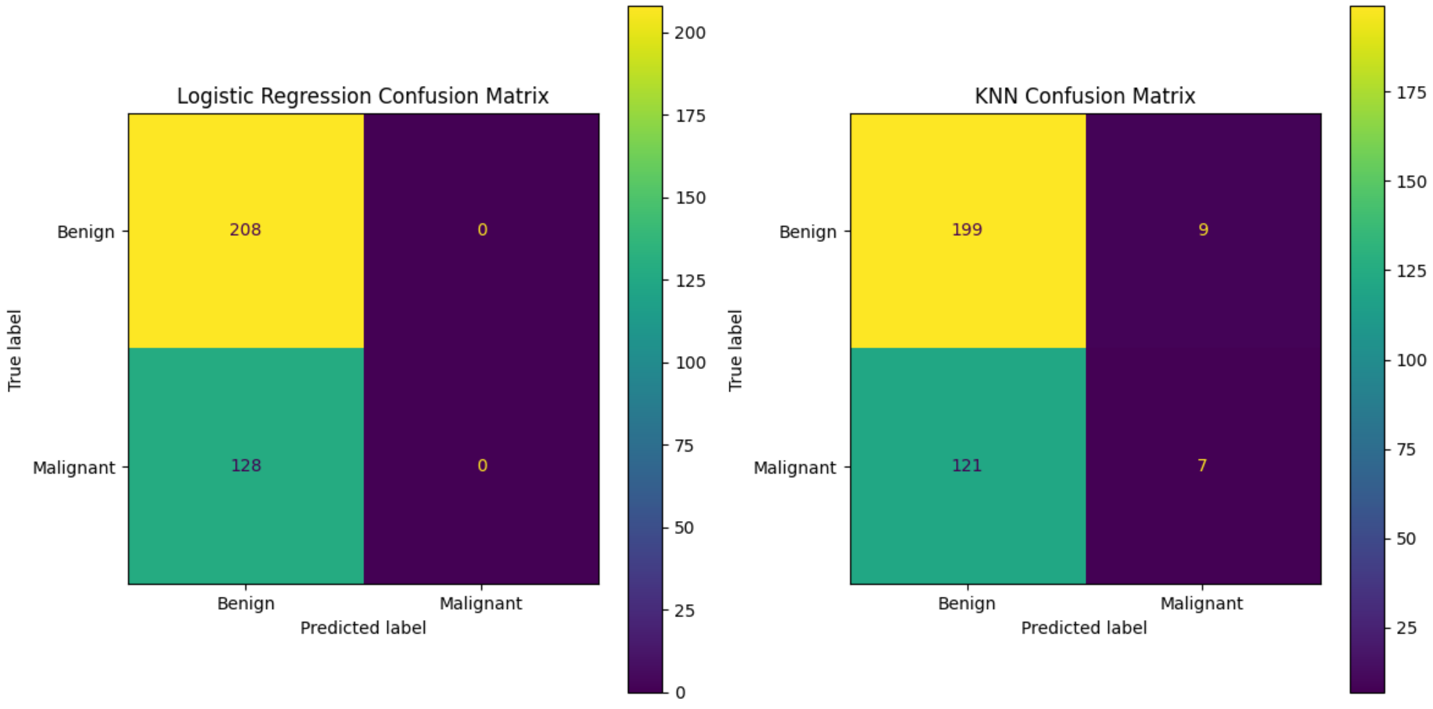
VII. KNN Model Evaluation:

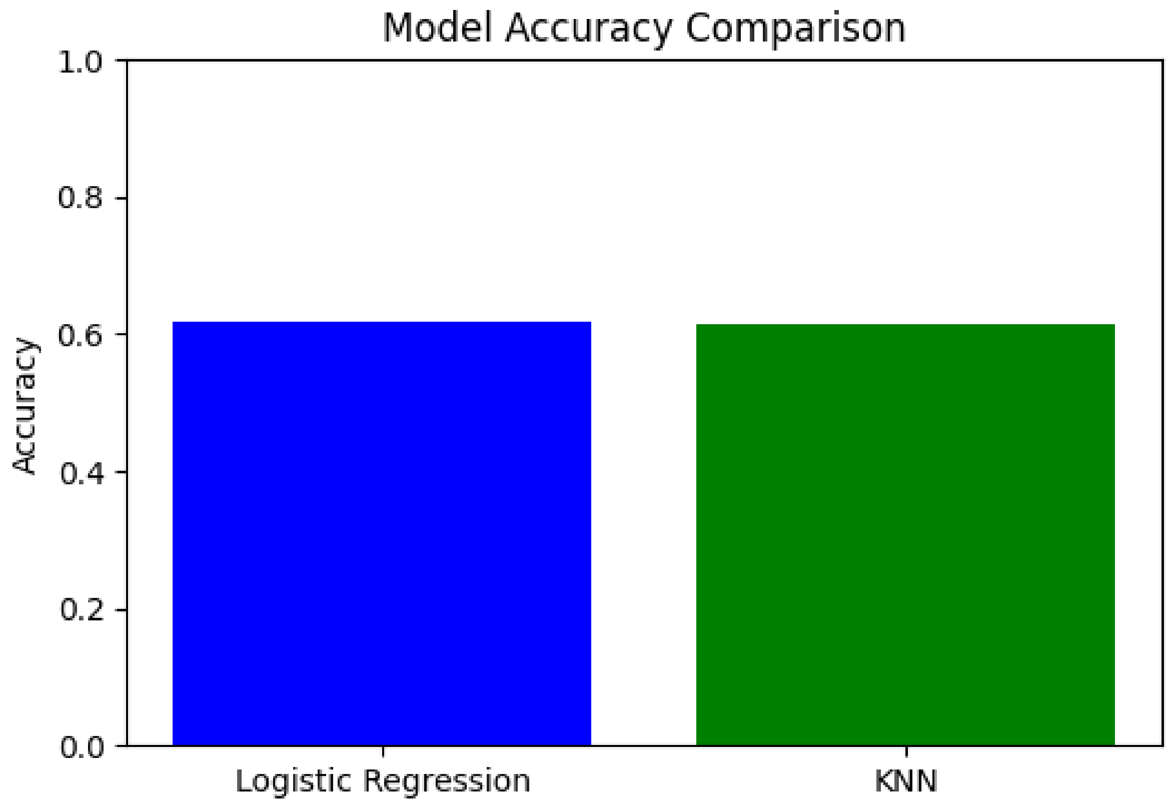
Model Accuracy: 61%

Confusion Matrix: array ([[199, 9], [121, 7]])

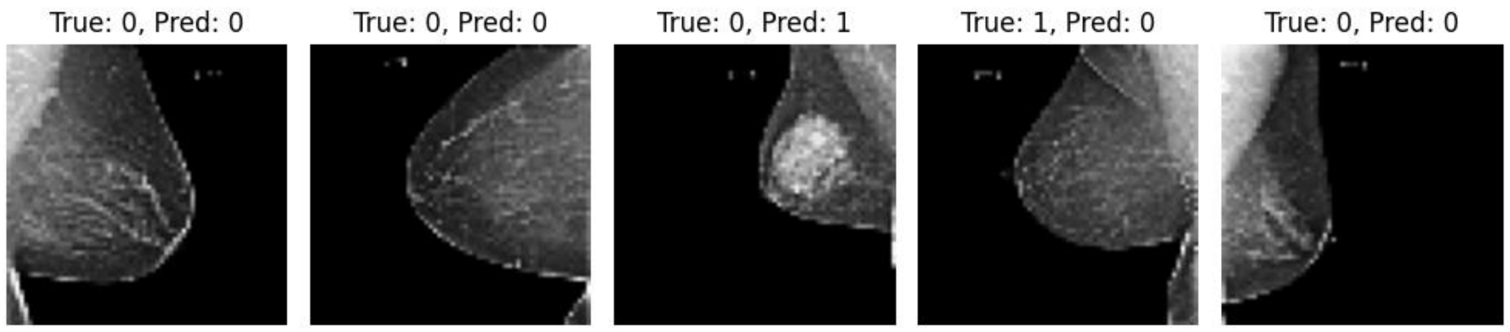


VIII. Comparison between two models:





IX. Sample Logistic Regression Predictions:



X. Sample KNN Predictions: