**FUTURE SALES PREDICTIONS**

TEAM MEMBERS

M.KALAIMUTHAMIZHAN

B.DEENADHAYALAN

M.SANTHOSH

S.ARUN KUMAR

P.BASKAR

**PHASE 3 SUBMISSION DOCUMENT**

**PHASE 3 DEVELOPMENT PART 1**



INTRODUCTION

In the analysis of the "Future Sales Prediction" dataset, we conducted a comprehensive series of data analysis steps to create an accurate prediction model. The process began with Exploratory Data Analysis (EDA) to understand the dataset's characteristics. Subsequently, we performed data preprocessing, including outlier detection and handling using the winzoring technique, as well as data normalization using the min-max method. We then developed multiple models, including Linear Regression, Ridge Regression, Lasso Regression, Decision Tree, and Random Forest, all of which were evaluated through cross-validation. The model evaluation results revealed that Random Forest outperformed others, yielding an average Mean Squared Error (MSE) of 10.32%, Root Mean Squared Error (RMSE) of 8.09%, Mean Absolute Error (MAE) of 5.99%, and an R-squared value of 94.27%. Additionally, we conducted classic assumption tests, including tests for linearity, homoscedasticity, normality, multicollinearity, outliers, and independence, to ensure the validity of our model. These results provide in-depth insights into the quality of our prediction model and its relevance in the context of future sales forecasting.

Dataset link: https://www.kaggle.com/datasets/chakradharmattapalli/future-sales-prediction

NECESSARY STEPS TO FOLLOW:

Start by importing libraries:

Pandas:

Pandas is essential for data manipulation and analysis, particularly for loading and handling datasets.

Program:

import pandas as pd

NumPy:

NumPy is used for numerical computations, and it complements Pandas for handling arrays and mathematical operations.

Program:

import NumPy as np

Scikit-Learn (sklearn):

Scikit-Learn provides tools for machine learning, including dataset splitting, preprocessing, and model evaluation. You'll import specific modules as needed for your analysis.

Program:

from sklearn. model selection import train\_test\_split

from sklearn. Preprocessing import StandardScaler # For data scaling (if needed)

LOAD THE DATASET:

To load a dataset for credit card fraud detection, We can use the Pandas library in Python. Here's how we can load a dataset from a CSV file, which is a common data format:

Program:

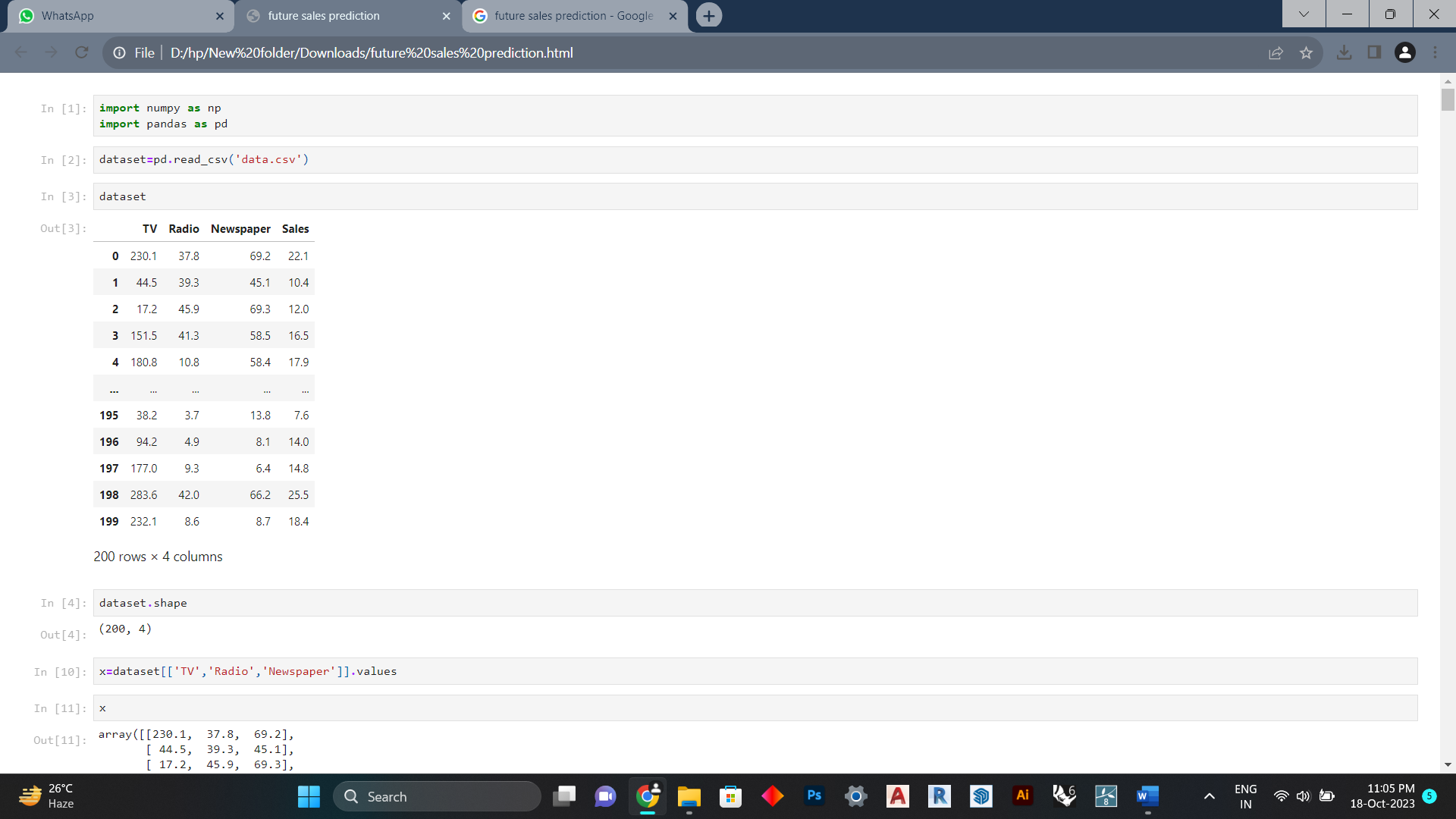
import pandas as pd

# Specify the file path to your dataset

File\_path=("C:\Users\BASKAR LOKESH\Downloads\data.csv ")

# Use Pandas to read the CSV file into a DataFrame

df =pd.read\_csv"(C:\Users\BASKAR LOKESH\Downloads\data.csv ")



# Now, 'df' contains your dataset, and you can start working with it.

In the code above:

Import the Pandas library to work with data.

Replace 'your\_dataset.csv' with the actual file path to our dataset. Make sure that the CSV file is in the same directory as our Python script, or provide the full path to the file if it's located elsewhere.

The pd.read\_csv(file\_path) function reads the CSV file and stores its contents in a Pandas DataFrame called df. This DataFrame is a two-dimensional table-like data structure that you can manipulate and analyze.

After loading the dataset into a Data Frame, we can perform various data analysis tasks, such as data exploration, preprocessing, and modelling, depending on your specific objectives in credit card fraud detection.

Data processing :

Data processing occurs when data is collected and translated into usable information. Usually performed by a data scientist or team of data scientists, it is important for data processing to be done correctly as not to negatively affect the end product, or data output.

Basic Summary Statistics:

Use Pandas to obtain summary statistics of the dataset, which can give a quick overview of the data, including counts, means, standard deviations, and percentiles.

print(df.describe())

Data Shape:

Use to see how many Rows and columns in our Dataset.

df.shape

(200, 4)

Dependent and Independent Variables

So, y is referred to as dependent feature or variable of total\_vaccinations and x is referred to as independent features or variables of location,date,vaccine,total\_vaccinations. Any predictive mathematical model tends to divide the observations (data) into dependent/ independent features in order to determine the causal effect.

Program:

x=dataset[['TV','Radio','Newspaper']].values

# x for independent variables

x

array([[230.1, 37.8, 69.2],

[ 44.5, 39.3, 45.1],

[ 17.2, 45.9, 69.3],

[151.5, 41.3, 58.5],

[180.8, 10.8, 58.4],

[ 8.7, 48.9, 75. ],

[ 57.5, 32.8, 23.5],

[120.2, 19.6, 11.6],

[ 8.6, 2.1, 1. ],

[199.8, 2.6, 21.2],

[ 66.1, 5.8, 24.2],

[214.7, 24. , 4. ],

[ 23.8, 35.1, 65.9],

[ 97.5, 7.6, 7.2],

[204.1, 32.9, 46. ],

[195.4, 47.7, 52.9],

[ 67.8, 36.6, 114. ],

[281.4, 39.6, 55.8],

[ 69.2, 20.5, 18.3],

[147.3, 23.9, 19.1],

[218.4, 27.7, 53.4],

[237.4, 5.1, 23.5],

[ 13.2, 15.9, 49.6],

[228.3, 16.9, 26.2],

[ 62.3, 12.6, 18.3],

[262.9, 3.5, 19.5],

[142.9, 29.3, 12.6],

[240.1, 16.7, 22.9],

[248.8, 27.1, 22.9],

[ 70.6, 16. , 40.8],

[292.9, 28.3, 43.2],

[112.9, 17.4, 38.6],

[ 97.2, 1.5, 30. ],

[265.6, 20. , 0.3],

[ 95.7, 1.4, 7.4],

[290.7, 4.1, 8.5],

[266.9, 43.8, 5. ],

[ 74.7, 49.4, 45.7],

[ 43.1, 26.7, 35.1],

[228. , 37.7, 32. ],

[202.5, 22.3, 31.6],

[177. , 33.4, 38.7],

[293.6, 27.7, 1.8],

[206.9, 8.4, 26.4],

[ 25.1, 25.7, 43.3],

[175.1, 22.5, 31.5],

[ 89.7, 9.9, 35.7],

[239.9, 41.5, 18.5],

[227.2, 15.8, 49.9],

[ 66.9, 11.7, 36.8],

[199.8, 3.1, 34.6],

[100.4, 9.6, 3.6],

[216.4, 41.7, 39.6],

[182.6, 46.2, 58.7],

[262.7, 28.8, 15.9],

[198.9, 49.4, 60. ],

[ 7.3, 28.1, 41.4],

[136.2, 19.2, 16.6],

[210.8, 49.6, 37.7],

[210.7, 29.5, 9.3],

[ 53.5, 2. , 21.4],

[261.3, 42.7, 54.7],

[239.3, 15.5, 27.3],

[102.7, 29.6, 8.4],

[131.1, 42.8, 28.9],

[ 69. , 9.3, 0.9],

[ 31.5, 24.6, 2.2],

[139.3, 14.5, 10.2],

[237.4, 27.5, 11. ],

[216.8, 43.9, 27.2],

[199.1, 30.6, 38.7],

[109.8, 14.3, 31.7],

[ 26.8, 33. , 19.3],

[129.4, 5.7, 31.3],

[213.4, 24.6, 13.1],

[ 16.9, 43.7, 89.4],

[ 27.5, 1.6, 20.7],

[120.5, 28.5, 14.2],

[ 5.4, 29.9, 9.4],

[116. , 7.7, 23.1],

[ 76.4, 26.7, 22.3],

[239.8, 4.1, 36.9],

[ 75.3, 20.3, 32.5],

[ 68.4, 44.5, 35.6],

[213.5, 43. , 33.8],

[193.2, 18.4, 65.7],

[ 76.3, 27.5, 16. ],

[110.7, 40.6, 63.2],

[ 88.3, 25.5, 73.4],

[109.8, 47.8, 51.4],

[134.3, 4.9, 9.3],

[ 28.6, 1.5, 33. ],

[217.7, 33.5, 59. ],

[250.9, 36.5, 72.3],

[107.4, 14. , 10.9],

[163.3, 31.6, 52.9],

[197.6, 3.5, 5.9],

[184.9, 21. , 22. ],

[289.7, 42.3, 51.2],

[135.2, 41.7, 45.9],

[222.4, 4.3, 49.8],

[296.4, 36.3, 100.9],

[280.2, 10.1, 21.4],

[187.9, 17.2, 17.9],

[238.2, 34.3, 5.3],

[137.9, 46.4, 59. ],

[ 25. , 11. , 29.7],

[ 90.4, 0.3, 23.2],

[ 13.1, 0.4, 25.6],

[255.4, 26.9, 5.5],

[225.8, 8.2, 56.5],

[241.7, 38. , 23.2],

[175.7, 15.4, 2.4],

[209.6, 20.6, 10.7],

[ 78.2, 46.8, 34.5],

[ 75.1, 35. , 52.7],

[139.2, 14.3, 25.6],

[ 76.4, 0.8, 14.8],

[125.7, 36.9, 79.2],

[ 19.4, 16. , 22.3],

[141.3, 26.8, 46.2],

[ 18.8, 21.7, 50.4],

[224. , 2.4, 15.6],

[123.1, 34.6, 12.4],

[229.5, 32.3, 74.2],

[ 87.2, 11.8, 25.9],

[ 7.8, 38.9, 50.6],

[ 80.2, 0. , 9.2],

[220.3, 49. , 3.2],

[ 59.6, 12. , 43.1],

[ 0.7, 39.6, 8.7],

[265.2, 2.9, 43. ],

[ 8.4, 27.2, 2.1],

[219.8, 33.5, 45.1],

[ 36.9, 38.6, 65.6],

[ 48.3, 47. , 8.5],

[ 25.6, 39. , 9.3],

[273.7, 28.9, 59.7],

[ 43. , 25.9, 20.5],

[184.9, 43.9, 1.7],

[ 73.4, 17. , 12.9],

[193.7, 35.4, 75.6],

[220.5, 33.2, 37.9],

[104.6, 5.7, 34.4],

[ 96.2, 14.8, 38.9],

[140.3, 1.9, 9. ],

[240.1, 7.3, 8.7],

[243.2, 49. , 44.3],

[ 38. , 40.3, 11.9],

[ 44.7, 25.8, 20.6],

[280.7, 13.9, 37. ],

[121. , 8.4, 48.7],

[197.6, 23.3, 14.2],

[171.3, 39.7, 37.7],

[187.8, 21.1, 9.5],

[ 4.1, 11.6, 5.7],

[ 93.9, 43.5, 50.5],

[149.8, 1.3, 24.3],

[ 11.7, 36.9, 45.2],

[131.7, 18.4, 34.6],

[172.5, 18.1, 30.7],

[ 85.7, 35.8, 49.3],

[188.4, 18.1, 25.6],

[163.5, 36.8, 7.4],

[117.2, 14.7, 5.4],

[234.5, 3.4, 84.8],

[ 17.9, 37.6, 21.6],

[206.8, 5.2, 19.4],

[215.4, 23.6, 57.6],

[284.3, 10.6, 6.4],

[ 50. , 11.6, 18.4],

[164.5, 20.9, 47.4],

[ 19.6, 20.1, 17. ],

[168.4, 7.1, 12.8],

[222.4, 3.4, 13.1],

[276.9, 48.9, 41.8],

[248.4, 30.2, 20.3],

[170.2, 7.8, 35.2],

[276.7, 2.3, 23.7],

[165.6, 10. , 17.6],

[156.6, 2.6, 8.3],

[218.5, 5.4, 27.4],

[ 56.2, 5.7, 29.7],

[287.6, 43. , 71.8],

[253.8, 21.3, 30. ],

[205. , 45.1, 19.6],

[139.5, 2.1, 26.6],

[191.1, 28.7, 18.2],

[286. , 13.9, 3.7],

[ 18.7, 12.1, 23.4],

[ 39.5, 41.1, 5.8],

[ 75.5, 10.8, 6. ],

[ 17.2, 4.1, 31.6],

[166.8, 42. , 3.6],

[149.7, 35.6, 6. ],

[ 38.2, 3.7, 13.8],

[ 94.2, 4.9, 8.1],

[177. , 9.3, 6.4],

[283.6, 42. , 66.2],

[232.1, 8.6, 8.7]])

# y for dependent variables

y=dataset[['Sales']].values

array([[22.1],

[10.4],

[12. ],

[16.5],

[17.9],

[ 7.2],

[11.8],

[13.2],

[ 4.8],

[15.6],

[12.6],

[17.4],

[ 9.2],

[13.7],

[19. ],

[22.4],

[12.5],

[24.4],

[11.3],

[14.6],

[18. ],

[17.5],

[ 5.6],

[20.5],

[ 9.7],

[17. ],

[15. ],

[20.9],

[18.9],

[10.5],

[21.4],

[11.9],

[13.2],

[17.4],

[11.9],

[17.8],

[25.4],

[14.7],

[10.1],

[21.5],

[16.6],

[17.1],

[20.7],

[17.9],

[ 8.5],

[16.1],

[10.6],

[23.2],

[19.8],

[ 9.7],

[16.4],

[10.7],

[22.6],

[21.2],

[20.2],

[23.7],

[ 5.5],

[13.2],

[23.8],

[18.4],

[ 8.1],

[24.2],

[20.7],

[14. ],

[16. ],

[11.3],

[11. ],

[13.4],

[18.9],

[22.3],

[18.3],

[12.4],

[ 8.8],

[11. ],

[17. ],

[ 8.7],

[ 6.9],

[14.2],

[ 5.3],

[11. ],

[11.8],

[17.3],

[11.3],

[13.6],

[21.7],

[20.2],

[12. ],

[16. ],

[12.9],

[16.7],

[14. ],

[ 7.3],

[19.4],

[22.2],

[11.5],

[16.9],

[16.7],

[20.5],

[25.4],

[17.2],

[16.7],

[23.8],

[19.8],

[19.7],

[20.7],

[15. ],

[ 7.2],

[12. ],

[ 5.3],

[19.8],

[18.4],

[21.8],

[17.1],

[20.9],

[14.6],

[12.6],

[12.2],

[ 9.4],

[15.9],

[ 6.6],

[15.5],

[ 7. ],

[16.6],

[15.2],

[19.7],

[10.6],

[ 6.6],

[11.9],

[24.7],

[ 9.7],

[ 1.6],

[17.7],

[ 5.7],

[19.6],

[10.8],

[11.6],

[ 9.5],

[20.8],

[ 9.6],

[20.7],

[10.9],

[19.2],

[20.1],

[10.4],

[12.3],

[10.3],

[18.2],

[25.4],

[10.9],

[10.1],

[16.1],

[11.6],

[16.6],

[16. ],

[20.6],

[ 3.2],

[15.3],

[10.1],

[ 7.3],

[12.9],

[16.4],

[13.3],

[19.9],

[18. ],

[11.9],

[16.9],

[ 8. ],

[17.2],

[17.1],

[20. ],

[ 8.4],

[17.5],

[ 7.6],

[16.7],

[16.5],

[27. ],

[20.2],

[16.7],

[16.8],

[17.6],

[15.5],

[17.2],

[ 8.7],

[26.2],

[17.6],

[22.6],

[10.3],

[17.3],

[20.9],

[ 6.7],

[10.8],

[11.9],

[ 5.9],

[19.6],

[17.3],

[ 7.6],

[14. ],

[14.8],

[25.5],

[18.4]])

Sklearn.impute

The SimpleImputer class provides basic strategies for imputing missing values. Missing values can be imputed with a provided constant value, or using the statistics (mean, median or most frequent) of each column in which the missing values are located. This class also allows for different missing values encodings.

Program:

**from** sklearn.impute **import** SimpleImputer

In [18]:

imputer**=**SimpleImputer(missing\_values**=**np**.**nan,strategy**=**'mean')

In [19]:

imputer**=**imputer**.**fit(x[:,1:3])

In [20]:

x[:,1:3]**=**imputer**.**transform(x[:,1:3])

In [21]:

x

output: array([[230.1, 37.8, 69.2],

[ 44.5, 39.3, 45.1],

[ 17.2, 45.9, 69.3],

[151.5, 41.3, 58.5],

[180.8, 10.8, 58.4],

[ 8.7, 48.9, 75. ],

[ 57.5, 32.8, 23.5],

[120.2, 19.6, 11.6],

[ 8.6, 2.1, 1. ],

[199.8, 2.6, 21.2],

[ 66.1, 5.8, 24.2],

[214.7, 24. , 4. ],

[ 23.8, 35.1, 65.9],

[ 97.5, 7.6, 7.2],

[204.1, 32.9, 46. ],

[195.4, 47.7, 52.9],

[ 67.8, 36.6, 114. ],

[281.4, 39.6, 55.8],

[ 69.2, 20.5, 18.3],

[147.3, 23.9, 19.1],

[218.4, 27.7, 53.4],

[237.4, 5.1, 23.5],

[ 13.2, 15.9, 49.6],

[228.3, 16.9, 26.2],

[ 62.3, 12.6, 18.3],

[262.9, 3.5, 19.5],

[142.9, 29.3, 12.6],

[240.1, 16.7, 22.9],

[248.8, 27.1, 22.9],

[ 70.6, 16. , 40.8],

[292.9, 28.3, 43.2],

[112.9, 17.4, 38.6],

[ 97.2, 1.5, 30. ],

[265.6, 20. , 0.3],

[ 95.7, 1.4, 7.4],

[290.7, 4.1, 8.5],

[266.9, 43.8, 5. ],

[ 74.7, 49.4, 45.7],

[ 43.1, 26.7, 35.1],

[228. , 37.7, 32. ],

[202.5, 22.3, 31.6],

[177. , 33.4, 38.7],

[293.6, 27.7, 1.8],

[206.9, 8.4, 26.4],

[ 25.1, 25.7, 43.3],

[175.1, 22.5, 31.5],

[ 89.7, 9.9, 35.7],

[239.9, 41.5, 18.5],

[227.2, 15.8, 49.9],

[ 66.9, 11.7, 36.8],

[199.8, 3.1, 34.6],

[100.4, 9.6, 3.6],

[216.4, 41.7, 39.6],

[182.6, 46.2, 58.7],

[262.7, 28.8, 15.9],

[198.9, 49.4, 60. ],

[ 7.3, 28.1, 41.4],

[136.2, 19.2, 16.6],

[210.8, 49.6, 37.7],

[210.7, 29.5, 9.3],

[ 53.5, 2. , 21.4],

[261.3, 42.7, 54.7],

[239.3, 15.5, 27.3],

[102.7, 29.6, 8.4],

[131.1, 42.8, 28.9],

[ 69. , 9.3, 0.9],

[ 31.5, 24.6, 2.2],

[139.3, 14.5, 10.2],

[237.4, 27.5, 11. ],

[216.8, 43.9, 27.2],

[199.1, 30.6, 38.7],

[109.8, 14.3, 31.7],

[ 26.8, 33. , 19.3],

[129.4, 5.7, 31.3],

[213.4, 24.6, 13.1],

[ 16.9, 43.7, 89.4],

[ 27.5, 1.6, 20.7],

[120.5, 28.5, 14.2],

[ 5.4, 29.9, 9.4],

[116. , 7.7, 23.1],

[ 76.4, 26.7, 22.3],

[239.8, 4.1, 36.9],

[ 75.3, 20.3, 32.5],

[ 68.4, 44.5, 35.6],

[213.5, 43. , 33.8],

[193.2, 18.4, 65.7],

[ 76.3, 27.5, 16. ],

[110.7, 40.6, 63.2],

[ 88.3, 25.5, 73.4],

[109.8, 47.8, 51.4],

[134.3, 4.9, 9.3],

[ 28.6, 1.5, 33. ],

[217.7, 33.5, 59. ],

[250.9, 36.5, 72.3],

[107.4, 14. , 10.9],

[163.3, 31.6, 52.9],

[197.6, 3.5, 5.9],

[184.9, 21. , 22. ],

[289.7, 42.3, 51.2],

[135.2, 41.7, 45.9],

[222.4, 4.3, 49.8],

[296.4, 36.3, 100.9],

[280.2, 10.1, 21.4],

[187.9, 17.2, 17.9],

[238.2, 34.3, 5.3],

[137.9, 46.4, 59. ],

[ 25. , 11. , 29.7],

[ 90.4, 0.3, 23.2],

[ 13.1, 0.4, 25.6],

[255.4, 26.9, 5.5],

[225.8, 8.2, 56.5],

[241.7, 38. , 23.2],

[175.7, 15.4, 2.4],

[209.6, 20.6, 10.7],

[ 78.2, 46.8, 34.5],

[ 75.1, 35. , 52.7],

[139.2, 14.3, 25.6],

[ 76.4, 0.8, 14.8],

[125.7, 36.9, 79.2],

[ 19.4, 16. , 22.3],

[141.3, 26.8, 46.2],

[ 18.8, 21.7, 50.4],

[224. , 2.4, 15.6],

[123.1, 34.6, 12.4],

[229.5, 32.3, 74.2],

[ 87.2, 11.8, 25.9],

[ 7.8, 38.9, 50.6],

[ 80.2, 0. , 9.2],

[220.3, 49. , 3.2],

[ 59.6, 12. , 43.1],

[ 0.7, 39.6, 8.7],

[265.2, 2.9, 43. ],

[ 8.4, 27.2, 2.1],

[219.8, 33.5, 45.1],

[ 36.9, 38.6, 65.6],

[ 48.3, 47. , 8.5],

[ 25.6, 39. , 9.3],

[273.7, 28.9, 59.7],

[ 43. , 25.9, 20.5],

[184.9, 43.9, 1.7],

[ 73.4, 17. , 12.9],

[193.7, 35.4, 75.6],

[220.5, 33.2, 37.9],

[104.6, 5.7, 34.4],

[ 96.2, 14.8, 38.9],

[140.3, 1.9, 9. ],

[240.1, 7.3, 8.7],

[243.2, 49. , 44.3],

[ 38. , 40.3, 11.9],

[ 44.7, 25.8, 20.6],

[280.7, 13.9, 37. ],

[121. , 8.4, 48.7],

[197.6, 23.3, 14.2],

[171.3, 39.7, 37.7],

[187.8, 21.1, 9.5],

[ 4.1, 11.6, 5.7],

[ 93.9, 43.5, 50.5],

[149.8, 1.3, 24.3],

[ 11.7, 36.9, 45.2],

[131.7, 18.4, 34.6],

[172.5, 18.1, 30.7],

[ 85.7, 35.8, 49.3],

[188.4, 18.1, 25.6],

[163.5, 36.8, 7.4],

[117.2, 14.7, 5.4],

[234.5, 3.4, 84.8],

[ 17.9, 37.6, 21.6],

[206.8, 5.2, 19.4],

[215.4, 23.6, 57.6],

[284.3, 10.6, 6.4],

[ 50. , 11.6, 18.4],

[164.5, 20.9, 47.4],

[ 19.6, 20.1, 17. ],

[168.4, 7.1, 12.8],

[222.4, 3.4, 13.1],

[276.9, 48.9, 41.8],

[248.4, 30.2, 20.3],

[170.2, 7.8, 35.2],

[276.7, 2.3, 23.7],

[165.6, 10. , 17.6],

[156.6, 2.6, 8.3],

[218.5, 5.4, 27.4],

[ 56.2, 5.7, 29.7],

[287.6, 43. , 71.8],

[253.8, 21.3, 30. ],

[205. , 45.1, 19.6],

[139.5, 2.1, 26.6],

[191.1, 28.7, 18.2],

[286. , 13.9, 3.7],

[ 18.7, 12.1, 23.4],

[ 39.5, 41.1, 5.8],

[ 75.5, 10.8, 6. ],

[ 17.2, 4.1, 31.6],

[166.8, 42. , 3.6],

[149.7, 35.6, 6. ],

[ 38.2, 3.7, 13.8],

[ 94.2, 4.9, 8.1],

[177. , 9.3, 6.4],

[283.6, 42. , 66.2],

[232.1, 8.6, 8.7]])

Onehotencoder:

One-Hot Encoding is another popular technique for treating categorical variables. It simply creates additional features based on the number of unique values in the categorical feature. Every unique value in the category will be added as a feature. One-Hot Encoding is the process of creating dummy variables.

Program:

**from** sklearn.preprocessing **import** OneHotEncoder

In [30]:

onehotencoder**=**OneHotEncoder()

In [31]:

onehotencoder**.**fit\_transform(dataset**.**TV**.**values**.**reshape(**-**1,1))**.**toarray()

Out[31]:

array([[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.],

...,

[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.],

[0., 0., 0., ..., 0., 0., 0.]])

Label encoder

Label encoding is a technique used in machine learning and data analysis to convert categorical variables into numerical format. It is particularly useful when working with algorithms that require numerical input, as most machine learning models can only operate on numerical data.

LabelEncoder Fit\_transform

LabelEncoder , we can use the fit\_transform function. This function fits the LabelEncoder object to the input data, and then transforms the data into encoded values. By default, fit\_transform assigns a unique numerical value to each category in the input data.

Program:

labelencoder\_y**=**LabelEncoder()

In [33]:

y**=**labelencoder\_y**.**fit\_transform(y)

C:\Users\BASKAR LOKESH\anaconda3\Lib\site-packages\sklearn\preprocessing\\_label.py:114: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples, ), for example using ravel().

y = column\_or\_1d(y, warn=True)

In [34]:

Y

Output:

array([106, 28, 40, 66, 80, 12, 38, 47, 2, 61, 45, 75, 21,

51, 86, 109, 44, 115, 35, 54, 81, 76, 5, 96, 25, 71,

57, 100, 85, 29, 102, 39, 47, 75, 39, 79, 117, 55, 26,

103, 67, 72, 98, 80, 18, 64, 30, 111, 91, 25, 65, 31,

110, 101, 95, 112, 4, 47, 113, 84, 16, 114, 98, 52, 63,

35, 34, 49, 85, 108, 83, 43, 20, 34, 71, 19, 10, 53,

3, 34, 38, 74, 35, 50, 104, 95, 40, 63, 46, 68, 52,

13, 88, 107, 36, 70, 68, 96, 117, 73, 68, 113, 91, 90,

98, 57, 12, 40, 3, 91, 84, 105, 72, 100, 54, 45, 41,

22, 62, 8, 60, 11, 67, 58, 90, 30, 8, 39, 116, 25,

0, 78, 6, 89, 32, 37, 23, 99, 24, 98, 33, 87, 94,

28, 42, 27, 82, 117, 33, 26, 64, 37, 67, 63, 97, 1,

59, 26, 13, 46, 65, 48, 92, 81, 39, 70, 15, 73, 72,

93, 17, 76, 14, 68, 66, 120, 95, 68, 69, 77, 60, 73,

19, 119, 77, 110, 27, 74, 100, 9, 32, 39, 7, 89, 74,

14, 52, 56, 118, 84], dtype=int64)

sklearn model\_selection in train\_test\_split

train\_test\_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train\_test\_split will make random partitions for the two subsets.

Program:

**from** sklearn.model\_selection **import** train\_test\_split

In [36]:

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.2,random\_state**=**0)

In [37]:

x\_train

Out[37]:

array([[ 26. , 38.6, 65.6],

[ 25. , 24.6, 2.2],

[ 94. , 29.3, 12.6],

[131. , 20.6, 10.7],

[137. , 23.6, 57.6],

[ 69. , 29.6, 8.4],

[ 6. , 2.1, 1. ],

[ 11. , 43.7, 89.4],

[ 81. , 36.9, 79.2],

[ 70. , 5.7, 34.4],

[ 72. , 14.3, 31.7],

[152. , 32.3, 74.2],

[167. , 21.3, 30. ],

[114. , 21. , 22. ],

[ 33. , 25.8, 20.6],

[ 40. , 12.6, 18.3],

[187. , 28.3, 43.2],

[108. , 18.1, 30.7],

[126. , 22.3, 31.6],

[ 3. , 28.1, 41.4],

[172. , 2.9, 43. ],

[122. , 3.5, 5.9],

[142. , 5.4, 27.4],

[ 95. , 23.9, 19.1],

[107. , 39.7, 37.7],

[140. , 33.5, 59. ],

[170. , 28.8, 15.9],

[101. , 36.8, 7.4],

[ 68. , 9.6, 3.6],

[ 53. , 27.5, 16. ],

[114. , 43.9, 1.7],

[ 85. , 4.9, 9.3],

[175. , 28.9, 59.7],

[189. , 36.3, 100.9],

[ 65. , 14.8, 38.9],

[ 72. , 47.8, 51.4],

[168. , 26.9, 5.5],

[127. , 32.9, 46. ],

[161. , 16.7, 22.9],

[120. , 35.4, 75.6],

[118. , 28.7, 18.2],

[ 60. , 9.9, 35.7],

[ 30. , 25.9, 20.5],

[ 28. , 3.7, 13.8],

[ 9. , 0.4, 25.6],

[158. , 15.5, 27.3],

[ 12. , 45.9, 69.3],

[132. , 29.5, 9.3],

[ 21. , 39. , 9.3],

[111. , 9.3, 6.4],

[130. , 8.4, 26.4],

[ 41. , 5.8, 24.2],

[ 96. , 35.6, 6. ],

[ 82. , 5.7, 31.3],

[ 63. , 4.9, 8.1],

[176. , 2.3, 23.7],

[177. , 48.9, 41.8],

[ 4. , 38.9, 50.6],

[166. , 36.5, 72.3],

[110. , 15.4, 2.4],

[ 8. , 36.9, 45.2],

[ 52. , 10.8, 6. ],

[125. , 3.1, 34.6],

[153. , 37.8, 69.2],

[ 71. , 14. , 10.9],

[148. , 8.2, 56.5],

[100. , 31.6, 52.9],

[ 83. , 42.8, 28.9],

[129. , 5.2, 19.4],

[111. , 33.4, 38.7],

[139. , 43.9, 27.2],

[ 42. , 11.7, 36.8],

[149. , 15.8, 49.9],

[119. , 18.4, 65.7],

[ 67. , 7.6, 7.2],

[ 57. , 35.8, 49.3],

[151. , 16.9, 26.2],

[ 91. , 2.1, 26.6],

[ 34. , 47. , 8.5],

[141. , 27.7, 53.4],

[121. , 47.7, 52.9],

[ 2. , 29.9, 9.4],

[157. , 34.3, 5.3],

[138. , 41.7, 39.6],

[146. , 4.3, 49.8],

[ 23. , 1.6, 20.7],

[ 98. , 41.3, 58.5],

[ 89. , 14.3, 25.6],

[ 76. , 14.7, 5.4],

[181. , 42. , 66.2],

[ 38. , 32.8, 23.5],

[156. , 27.5, 11. ],

[135. , 43. , 33.8],

[ 15. , 21.7, 50.4],

[ 1. , 11.6, 5.7],

[102. , 20.9, 47.4],

[ 62. , 43.5, 50.5],

[ 24. , 1.5, 33. ],

[154. , 8.6, 8.7],

[136. , 24. , 4. ],

[ 16. , 16. , 22.3],

[178. , 10.1, 21.4],

[186. , 4.1, 8.5],

[ 87. , 19.2, 16.6],

[ 45. , 9.3, 0.9],

[ 32. , 39.3, 45.1],

[ 93. , 26.8, 46.2],

[117. , 18.1, 25.6],

[188. , 27.7, 1.8],

[ 88. , 46.4, 59. ],

[ 5. , 27.2, 2.1],

[105. , 7.1, 12.8],

[180. , 39.6, 55.8],

[ 31. , 26.7, 35.1],

[143. , 33.5, 45.1],

[113. , 46.2, 58.7],

[ 97. , 1.3, 24.3],

[144. , 49. , 3.2],

[ 64. , 1.4, 7.4],

[165. , 27.1, 22.9],

[ 55. , 46.8, 34.5],

[ 79. , 8.4, 48.7],

[ 74. , 17.4, 38.6],

[ 13. , 37.6, 21.6],

[ 56. , 0. , 9.2],

[164. , 30.2, 20.3],

[ 66. , 1.5, 30. ],

[145. , 33.2, 37.9],

[182. , 10.6, 6.4],

[163. , 49. , 44.3],

[ 47. , 16. , 40.8],

[ 86. , 41.7, 45.9],

[ 51. , 20.3, 32.5],

[ 75. , 7.7, 23.1],

[ 50. , 35. , 52.7],

[ 27. , 40.3, 11.9],

[104. , 42. , 3.6],

[ 22. , 33. , 19.3],

[ 78. , 28.5, 14.2],

[171. , 3.5, 19.5],

[155. , 3.4, 84.8],

[159. , 4.1, 36.9],

[183. , 13.9, 3.7],

[146. , 3.4, 13.1],

[ 29. , 41.1, 5.8],

[150. , 37.7, 32. ],

[133. , 49.6, 37.7],

[ 48. , 17. , 12.9],

[ 59. , 25.5, 73.4],

[124. , 30.6, 38.7],

[ 73. , 40.6, 63.2],

[174. , 43.8, 5. ],

[156. , 5.1, 23.5],

[125. , 2.6, 21.2],

[116. , 17.2, 17.9],

[ 90. , 14.5, 10.2],

[ 12. , 4.1, 31.6],

[ 54. , 0.8, 14.8],

[160. , 41.5, 18.5],

[ 17. , 20.1, 17. ]])

In [38]:

x\_test

Out[38]:

array([[ 46. , 20.5, 18.3],

[ 35. , 11.6, 18.4],

[ 61. , 0.3, 23.2],

[185. , 42.3, 51.2],

[106. , 7.8, 35.2],

[ 37. , 5.7, 29.7],

[ 7. , 48.9, 75. ],

[161. , 7.3, 8.7],

[ 18. , 35.1, 65.9],

[122. , 23.3, 14.2],

[169. , 42.7, 54.7],

[ 58. , 11.8, 25.9],

[ 99. , 2.6, 8.3],

[115. , 21.1, 9.5],

[ 54. , 26.7, 22.3],

[ 77. , 19.6, 11.6],

[173. , 20. , 0.3],

[ 0. , 39.6, 8.7],

[ 49. , 49.4, 45.7],

[134. , 24.6, 13.1],

[184. , 43. , 71.8],

[ 92. , 1.9, 9. ],

[109. , 22.5, 31.5],

[ 84. , 18.4, 34.6],

[ 36. , 2. , 21.4],

[ 80. , 34.6, 12.4],

[103. , 10. , 17.6],

[128. , 45.1, 19.6],

[147. , 2.4, 15.6],

[ 20. , 25.7, 43.3],

[ 43. , 36.6, 114. ],

[123. , 49.4, 60. ],

[179. , 13.9, 37. ],

[162. , 38. , 23.2],

[ 10. , 15.9, 49.6],

[ 14. , 12.1, 23.4],

[ 39. , 12. , 43.1],

[112. , 10.8, 58.4],

[ 44. , 44.5, 35.6],

[ 19. , 11. , 29.7]])

In [39]:

y\_train

Out[39]:

array([ 32, 34, 57, 100, 72, 52, 2, 19, 62, 28, 43, 90, 77,

96, 26, 25, 102, 65, 67, 4, 78, 68, 73, 54, 63, 88,

95, 81, 31, 40, 98, 52, 99, 113, 42, 68, 91, 86, 100,

87, 74, 30, 24, 14, 3, 98, 40, 84, 23, 56, 80, 45,

74, 34, 52, 69, 120, 8, 107, 72, 13, 39, 65, 106, 36,

84, 70, 63, 73, 72, 108, 25, 91, 95, 51, 48, 96, 27,

37, 81, 109, 3, 98, 110, 68, 10, 66, 41, 39, 118, 38,

85, 104, 11, 1, 76, 59, 13, 84, 75, 8, 91, 79, 47,

35, 28, 60, 92, 98, 57, 6, 68, 115, 26, 89, 101, 26,

116, 39, 85, 54, 37, 39, 15, 39, 95, 47, 94, 93, 117,

29, 73, 35, 34, 45, 33, 89, 20, 53, 71, 70, 74, 100,

66, 32, 103, 113, 33, 46, 83, 63, 117, 76, 61, 90, 49,

7, 22, 111, 14], dtype=int64)

In [40]:

y\_test

Out[40]:

array([ 35, 17, 40, 117, 68, 19, 12, 82, 21, 67, 114, 30, 60,

97, 38, 47, 75, 0, 55, 71, 119, 27, 64, 46, 16, 58,

77, 110, 67, 18, 44, 112, 64, 105, 5, 9, 25, 80, 50,

12], dtype=int64)

StandardScaler

Program:

**from** sklearn.preprocessing **import** StandardScaler

In [44]:

sc\_x**=**StandardScaler()

In [45]:

x\_train**=**sc\_x**.**fit\_transform(x\_train)

In [46]:

x\_test**=**sc\_x**.**transform(x\_test)

In [47]:

x\_train

Out[47]:

array([[-1.32545544, 1.0355176 , 1.65941078],

[-1.34400301, 0.08249594, -1.30629738],

[-0.06422095, 0.40243892, -0.81980897],

[ 0.622039 , -0.18979597, -0.90868666],

[ 0.73332439, 0.01442296, 1.28518893],

[-0.5279101 , 0.42286082, -1.01627544],

[-1.69640676, -1.44914602, -1.36243065],

[-1.60366893, 1.38268978, 2.77272078],

[-0.30533931, 0.91979354, 2.29558792],

[-0.50936253, -1.20408331, 0.19994556],

[-0.4722674 , -0.61865571, 0.07364569],

[ 1.01153788, 0.60665785, 2.06169926],

[ 1.28975138, -0.14214488, -0.00587645],

[ 0.30673037, -0.16256678, -0.3800983 ],

[-1.19562248, 0.16418351, -0.44558712],

[-1.06578951, -0.73437977, -0.55317591],

[ 1.6607027 , 0.33436595, 0.6115896 ],

[ 0.19544498, -0.35997841, 0.02686796],

[ 0.52930117, -0.07407191, 0.06896792],

[-1.75204946, 0.32075135, 0.52738968],

[ 1.38248921, -1.39468764, 0.60223405],

[ 0.4551109 , -1.35384385, -1.13321977],

[ 0.82606222, -1.2245052 , -0.12749855],

[-0.04567338, 0.03484485, -0.51575372],

[ 0.17689741, 1.11039787, 0.35431208],

[ 0.78896709, 0.68834542, 1.35067775],

[ 1.34539407, 0.36840244, -0.66544246],

[ 0.06561201, 0.91298624, -1.06305318],

[-0.54645767, -0.9385987 , -1.24080855],

[-0.82467116, 0.27990757, -0.66076469],

[ 0.30673037, 1.39630437, -1.32968624],

[-0.23114904, -1.25854169, -0.97417549],

[ 1.4381319 , 0.37520973, 1.38342216],

[ 1.69779783, 0.87894975, 3.31066468],

[-0.60210036, -0.58461923, 0.41044535],

[-0.4722674 , 1.66178898, 0.99516699],

[ 1.30829894, 0.23906378, -1.15193086],

[ 0.54784873, 0.64750164, 0.74256725],

[ 1.17846598, -0.45528057, -0.33799834],

[ 0.41801577, 0.81768408, 2.12718809],

[ 0.38092064, 0.36159514, -0.55785368],

[-0.69483819, -0.91817681, 0.26075661],

[-1.25126518, 0.17099081, -0.4502649 ],

[-1.28836031, -1.34022926, -0.7636757 ],

[-1.64076406, -1.56487008, -0.21169847],

[ 1.12282328, -0.53696814, -0.13217633],

[-1.58512136, 1.53245032, 1.83248838],

[ 0.64058656, 0.41605352, -0.97417549],

[-1.41819327, 1.06274679, -0.97417549],

[ 0.25108768, -0.95902059, -1.10983091],

[ 0.60349143, -1.02028627, -0.17427628],

[-1.04724195, -1.19727601, -0.27718729],

[-0.02712582, 0.83129867, -1.128542 ],

[-0.28679174, -1.20408331, 0.0549346 ],

[-0.6391955 , -1.25854169, -1.03030876],

[ 1.45667947, -1.43553142, -0.30057616],

[ 1.47522704, 1.73666925, 0.54610077],

[-1.73350189, 1.05593949, 0.95774481],

[ 1.27120381, 0.89256435, 1.97282157],

[ 0.23254011, -0.54377544, -1.29694183],

[-1.65931163, 0.91979354, 0.70514506],

[-0.84321872, -0.85691113, -1.128542 ],

[ 0.5107536 , -1.38107304, 0.20930111],

[ 1.03008545, 0.98105922, 1.82781061],

[-0.49081497, -0.63907761, -0.89933112],

[ 0.93734762, -1.03390087, 1.23373342],

[ 0.04706445, 0.55900677, 1.06533359],

[-0.26824417, 1.3214241 , -0.05733196],

[ 0.58494386, -1.23811979, -0.5017204 ],

[ 0.25108768, 0.68153812, 0.40108981],

[ 0.77041953, 1.39630437, -0.1368541 ],

[-1.02869438, -0.79564545, 0.31221212],

[ 0.95589519, -0.51654625, 0.9250004 ],

[ 0.3994682 , -0.33955651, 1.66408855],

[-0.56500523, -1.07474465, -1.07240872],

[-0.75048089, 0.84491327, 0.89693376],

[ 0.99299032, -0.44166598, -0.18363183],

[-0.11986365, -1.44914602, -0.16492074],

[-1.17707491, 1.6073306 , -1.01159767],

[ 0.80751466, 0.29352216, 1.08872246],

[ 0.43656334, 1.65498168, 1.06533359],

[-1.77059703, 0.44328271, -0.96949771],

[ 1.10427571, 0.7428038 , -1.16128641],

[ 0.75187196, 1.24654382, 0.44318977],

[ 0.90025249, -1.29938547, 0.92032262],

[-1.38109814, -1.48318251, -0.44090935],

[ 0.00996932, 1.21931463, 1.32728888],

[-0.15695878, -0.61865571, -0.21169847],

[-0.39807714, -0.59142652, -1.15660864],

[ 1.5494173 , 1.26696572, 1.68747741],

[-1.10288465, 0.64069434, -0.3099317 ],

[ 1.08572815, 0.27990757, -0.89465334],

[ 0.69622926, 1.33503869, 0.17187893],

[-1.52947867, -0.11491569, 0.94838926],

[-1.78914459, -0.80245275, -1.14257532],

[ 0.08415958, -0.16937407, 0.80805607],

[-0.65774306, 1.36907518, 0.95306704],

[-1.36255057, -1.4899898 , 0.13445674],

[ 1.04863302, -1.00667168, -1.00224213],

[ 0.71477683, 0.04165215, -1.22209746],

[-1.5109311 , -0.50293165, -0.36606498],

[ 1.4937746 , -0.90456221, -0.40816494],

[ 1.64215513, -1.31300007, -1.01159767],

[-0.19405391, -0.28509813, -0.63269805],

[-0.97305168, -0.95902059, -1.36710843],

[-1.21417004, 1.08316868, 0.70046729],

[-0.08276851, 0.23225648, 0.75192279],

[ 0.36237307, -0.35997841, -0.21169847],

[ 1.67925026, 0.29352216, -1.32500847],

[-0.17550634, 1.56648681, 1.35067775],

[-1.71495433, 0.25948567, -1.31097515],

[ 0.13980228, -1.10878114, -0.81045343],

[ 1.53086973, 1.10359057, 1.20098901],

[-1.23271761, 0.22544919, 0.23268998],

[ 0.84460979, 0.68834542, 0.70046729],

[ 0.28818281, 1.55287221, 1.33664443],

[-0.00857825, -1.5036044 , -0.27250952],

[ 0.86315736, 1.74347655, -1.25951965],

[-0.62064793, -1.4967971 , -1.06305318],

[ 1.25265624, 0.25267838, -0.33799834],

[-0.78757602, 1.593716 , 0.20462334],

[-0.34243444, -1.02028627, 0.86886712],

[-0.43517227, -0.40762949, 0.39641204],

[-1.5665738 , 0.96744462, -0.39880939],

[-0.76902846, -1.59209927, -0.97885326],

[ 1.23410868, 0.4637046 , -0.45962044],

[-0.5835528 , -1.4899898 , -0.00587645],

[ 0.88170492, 0.66792353, 0.36366762],

[ 1.56796487, -0.87052572, -1.10983091],

[ 1.21556111, 1.74347655, 0.6630451 ],

[-0.93595655, -0.50293165, 0.49932304],

[-0.21260148, 1.24654382, 0.73788947],

[-0.86176629, -0.21021786, 0.11106788],

[-0.4166247 , -1.06793735, -0.3286428 ],

[-0.88031385, 0.79045489, 1.05597804],

[-1.30690787, 1.15124166, -0.85255339],

[ 0.12125471, 1.26696572, -1.24080855],

[-1.3996457 , 0.65430893, -0.50639818],

[-0.360982 , 0.34798054, -0.7449646 ],

[ 1.36394164, -1.35384385, -0.49704263],

[ 1.06718058, -1.36065115, 2.55754321],

[ 1.14137085, -1.31300007, 0.31688989],

[ 1.58651243, -0.6458849 , -1.23613078],

[ 0.90025249, -1.36065115, -0.79642011],

[-1.26981274, 1.20570004, -1.13789755],

[ 0.97444275, 0.97425192, 0.08767901],

[ 0.65913413, 1.78432033, 0.35431208],

[-0.91740899, -0.43485868, -0.80577565],

[-0.71338576, 0.14376162, 2.02427708],

[ 0.49220603, 0.49093379, 0.40108981],

[-0.45371983, 1.17166355, 1.54714422],

[ 1.41958434, 1.38949707, -1.17531973],

[ 1.08572815, -1.24492709, -0.3099317 ],

[ 0.5107536 , -1.41510953, -0.41752049],

[ 0.34382551, -0.42124408, -0.571887 ],

[-0.13841121, -0.60504112, -0.93207553],

[-1.58512136, -1.31300007, 0.06896792],

[-0.80612359, -1.53764089, -0.71689797],

[ 1.15991841, 1.23292923, -0.54382036],

[-1.49238353, -0.22383245, -0.61398696]])

In [48]:

x\_test

Out[48]:

array([[-0.95450412, -0.19660326, -0.55317591],

[-1.15852735, -0.80245275, -0.54849813],

[-0.67629063, -1.57167737, -0.32396502],

[ 1.62360756, 1.28738761, 0.98581145],

[ 0.15834985, -1.06113006, 0.23736775],

[-1.12143221, -1.20408331, -0.01990977],

[-1.67785919, 1.73666925, 2.09912145],

[ 1.17846598, -1.09516654, -1.00224213],

[-1.47383597, 0.79726218, 1.67344409],

[ 0.4551109 , -0.00599893, -0.7449646 ],

[ 1.32684651, 1.3146168 , 1.14953351],

[-0.73193333, -0.78883815, -0.19766515],

[ 0.02851688, -1.41510953, -1.02095322],

[ 0.32527794, -0.15575948, -0.96481994],

[-0.80612359, 0.22544919, -0.36606498],

[-0.37952957, -0.25786894, -0.8665867 ],

[ 1.40103677, -0.23063975, -1.39517507],

[-1.80769216, 1.10359057, -1.00224213],

[-0.89886142, 1.77070574, 0.72853393],

[ 0.6776817 , 0.08249594, -0.79642011],

[ 1.60506 , 1.33503869, 1.94943271],

[-0.10131608, -1.46276061, -0.98820881],

[ 0.21399254, -0.06045731, 0.06429014],

[-0.24969661, -0.33955651, 0.20930111],

[-1.13997978, -1.45595332, -0.40816494],

[-0.32388687, 0.7632257 , -0.82916452],

[ 0.10270715, -0.91136951, -0.58592032],

[ 0.5663963 , 1.47799194, -0.49236486],

[ 0.91880005, -1.42872413, -0.67947578],

[-1.43674084, 0.15737621, 0.61626737],

[-1.01014682, 0.89937165, 3.92345296],

[ 0.47365847, 1.77070574, 1.39745548],

[ 1.51232217, -0.6458849 , 0.32156767],

[ 1.19701354, 0.99467381, -0.32396502],

[-1.6222165 , -0.50973895, 0.91096708],

[-1.54802623, -0.76841626, -0.31460948],

[-1.08433708, -0.77522356, 0.60691183],

[ 0.26963524, -0.85691113, 1.32261111],

[-0.99159925, 1.43714816, 0.25607884],

[-1.4552884 , -0.84329653, -0.01990977]])

Conclusion

In the future of sales prediction is poised to undergo a profound transformation as we move forward into an increasingly data-driven and technologically advanced business landscape. With the continued evolution of artificial intelligence and machine learning, organizations are better equipped than ever to harness the power of predictive analytics. These tools enable businesses to uncover valuable insights, identify emerging trends, and make informed decisions, ultimately leading to more accurate sales forecasts. Moreover, as the world becomes more interconnected, real-time data streams and IoT devices will play an integral role in sales prediction, offering instantaneous feedback and adjusting forecasts on the fly. Ethical considerations surrounding data privacy and security will continue to be paramount in shaping the future of sales prediction. Overall, the future holds immense promise for businesses that can adapt to these advancements, as they stand to gain a competitive edge and optimize their sales strategies for enhanced profitability and customer satisfaction.