A Project Report On

BMW Car Sales Data Using Classification

Submitted in partial fulfillment of the requirement for the award of the degree
MASTER OF COMPUTER APPLICATION
from
Marwadi University

Academic Year 2025 – 26

Harshal Vadher(92400584126)
Ronit Pilojpara(92400584132)

Internal Guide
Gehna Sachdeva

Rajkot-Morbi Road, At & PO : Gauridad, Rajkot 360 003. Gujarat. India.

Faculty of Computer Applications (FoCA)

Certificate

This is to certify that the project work entitled

Heart Failure Prediction Using Logistic Regression

submitted in partial fulfillment of the requirement for the award of the degree of

Master of Computer Application

of the

Marwadi University

is a result of the bonafide work carried out by

Harshal Vadher(92400584126)
Ronit Pilojpara(92400584132)

during the academic year 2025 – 2026

PROF. Gehna Sachdeva	*Dr. Sunil Bajeja*	*Dr. R. Sridaran*
Faculty Guide	**HOD**	**Dean**

DECLARATION

We hereby declare that this project work entitled **BMW Car Sales** is a record done by us.

We also declare that the matter embodied in this project is genuine work done by us and has not been submitted whether to this University or to any other University / Institute for the fulfillment of the requirement of any course of study.

Place: Rajkot

Date:

Harshal Vadher (92400584126) Signature:_____

Ronit Pilojpara (92400584132) Signature:_____

ACKNOWLEDGEMENT

It is indeed a great pleasure to express our thanks and gratitude to all those who helped us. No serious and lasting achievement or success one can ever achieve without the help of friendly guidance and co-operation of so many people involved in the work.

We are very thankful to our guide **PROF. GEHNA SACHDEVA,** the person who makes us to follow the right steps during our project work. We express our deep sense of gratitude to for his guidance, suggestions and expertise at every stage. A part from that his valuable and expertise suggestion during documentation of our report indeed help us a lot.

Thanks to our friend and colleague who have been a source of inspiration and motivation that helped to us during our project work.

We are heartily thankful to the Dean of our department **Dr. R. Sridaran** for giving us an opportunity to work over this project and for their end-less and great support to all other people who directly or indirectly supported and help us to fulfil our task.

Harshal Vadher (92400584126) Signature:_____
Ronit Pilojpara (92400584132) Signature:_____

CONTENTS

Chapters	Particulars	Page No.
1	**Introduction** 1.1. Objective of the New System 1.2. Problem Definition 1.3. Core Components 1.4. Project Profile 1.5. Assumptions and Constraints 1.6. Advantages and Limitations of the Proposed System	6
2	**Requirement Determination & Analysis** 2.1. Requirement Determination 2.2. Targeted Users 2.3. Tool details (Python / PowerBI/ Tableau) 2.4. Library description (Details on various libraries / packages used)	7
3	**System Design** 3.1. Flowchart / Algorithm with steps 3.2. Dataset Design 3.3. Details on preprocessing steps applied	8
4	**Development** 4.1 Script details / Source code 4.2. Screen Shots / UI Design of simulation (if applicable) 4.3. Test reports	10
5	**Proposed Enhancements**	20
6	**Conclusion**	21
7	**Bibliography**	21

1. Introduction:

1.1 Objective of the New System:

The objective of the system is to analyze and manage BMW sales data efficiently. By examining attributes like model, year, region, fuel type, engine size, and price, the system aims to provide valuable insights into customer preferences, sales performance, and market trends. This will help stakeholders in decision-making, demand forecasting, and sales strategy optimization.

1.2 Problem Definition:

Heart disease is a leading cause of death globally. Early detection and prediction can significantly increase survival rates. However, manual analysis of large medical datasets is time-consuming and prone to errors. This project addresses the need for an automated and reliable predictive system.

1.3 Core Components:

- **Data Preprocessing:** Cleaning, handling missing values, and standardizing formats.
- **Data Analysis:** Identifying patterns in sales volume, fuel types, and regions.
- **Visualization:** Charts/graphs (line charts, bar plots, heatmaps) to understand sales trends.
- **Classification:** Categorizing sales into "High", "Medium", or "Low" performance.
- **Decision Support:** Insights for pricing, inventory management, and future strategy.

1.4 Project Profile:

- **Domain:** Data Analytics / Business Intelligence
- **Dataset Used:** BMW Sales Data (20,999 records, 11 attributes)
- **Technology Stack:** Python (Pandas, Matplotlib, Scikit-learn)
- **End Users:** Sales managers, business analysts, marketing teams
- **Outcome:** Interactive dashboards and statistical models to support BMW's global sales strategy.

1.5 Assumptions and Constraints:

- **Assumptions:**
  - The dataset provided is accurate and representative of BMW sales.
  - Regional and yearly data cover the majority of market trends.
  - Sales classification labels (High, Medium, Low) are predefined and correct.
- **Constraints:**
  - Historical data may not reflect future trends accurately.
  - Dataset may not include external factors (e.g., economic conditions, competitor pricing).
  - Limited to the attributes provided (e.g., missing customer demographics).

1.6 Advantages and Limitations of the Proposed System:

- **Advantages:**
  - Provides clear insights into BMW sales patterns.
  - Helps identify high-demand models and regions.
  - Supports data-driven business decisions.
  - Reduces manual effort in analyzing large datasets.
- **Limitations:**
  - Predictive accuracy depends on dataset quality.
  - External factors (market trends, government policies) are not considered.
  - Only focused on BMW data, not benchmarked against competitors.

2. Requirement Determination & Analysis:

2.1 Requirement Determination:

The project requires a system to analyze **BMW Sales Data** and classify sales performance into categories (High / Medium / Low). The major requirements include:

- **Data preprocessing**: Cleaning, handling missing values, encoding categorical data, and scaling numerical features.
- **Model training**: Applying machine learning classifiers (KNN, Decision Tree, Random Forest, SVM, Naive Bayes).
- **Performance evaluation**: Using confusion matrices, accuracy, and classification reports.
- **Visualization**: Heatmaps, histograms, scatterplots, and pair plots to identify relationships among attributes.
- **Prediction**: Allowing input of a new sales record and predicting its sales classification.

2.2 Targeted Users:

The system is useful for:

- **Business Analysts** – to interpret sales patterns and suggest strategies.
- **Sales Managers** – to monitor regional performance and demand trends.
- **Marketing Teams** – to identify customer preferences (fuel type, model popularity).
- **Data Scientists / Students** – to explore machine learning applications in real-world business datasets.

2.3 Tool details (Python / PowerBI/ Tableau):

The project is implemented in **Python** (using Jupyter/Colab). However, the same dataset could also be analyzed in **Power BI** or **Tableau** for dashboard-style visualizations.

- **Python**: Used for data preprocessing, machine learning, and visualization.
- **Power BI / Tableau**: Can be used later for interactive dashboards and business reporting.

2.4 Library description (Details on various libraries / packages used):

Your code uses several key Python libraries:

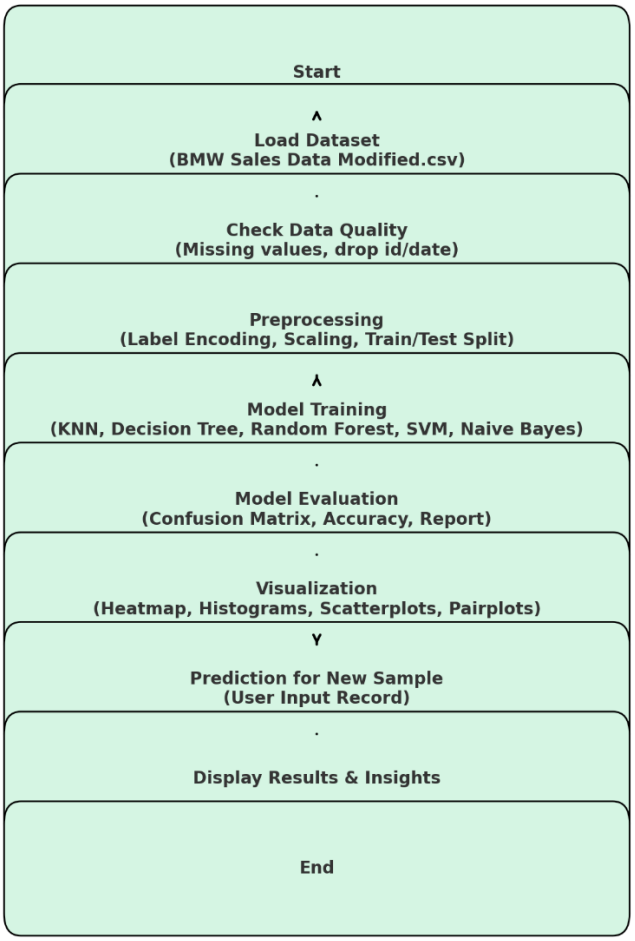
1. **pandas** – for data manipulation, cleaning, and analysis.
2. **numpy** – numerical operations, array handling.
3. **scikit-learn (sklearn)** –
    - train_test_split → splitting dataset.
    - LabelEncoder → converting categorical values to numeric codes.
    - StandardScaler → feature scaling.
    - Classification models:
        - KNN
        - Decision Tree Classifier
        - Random Forest Classifier
        - SVC (Support Vector Machine)
        - Naive Bayes
    - Accuracy Score, Confusion Matrix, Classification Report → model evaluation.
4. **matplotlib** – plotting graphs, confusion matrix heatmaps.
5. **seaborn** – advanced visualizations (heatmaps, histograms, pair plots).

3. System Design:

3.1 Flowchart / Algorithm with steps:

- **Algorithm:**
  Step 01:- Start
  Step 02:- Load Dataset (BMW Sales Data Modified.csv)
  Step 03:- Check Data Quality
  Step 04:- Preprocessing
  Step 05:- Model Training
  Step 06:- Model Evaluation
  Step 07:- Visualization
  Step 08:- Prediction for New Sample
  Step 09:- Display Results & Insights
  Step 10:- End

- **Flowchart:**



3.2 Dataset Design:

Dataset Characteristics (BMW Sales Data Modified.csv):

- **Size:** 20,999 rows × 11 columns
- **Attributes:**
    1. **Model** → BMW car model (e.g., 5 Series, i8, X3, 7 Series)
    2. **Year** → Manufacturing year
    3. **Region** → Geographic market (Asia, North America, Europe, etc.)
    4. **Color** → Vehicle Color
    5. **Fuel Type** → Petrol, Diesel, Hybrid, Electric
    6. **Transmission** → Manual / Automatic
    7. **Engine Size L** → Engine size in Liters
    8. **Mileage KM** → Mileage driven in Kilometers
    9. **Price USD** → Price of the vehicle in USD
    10. **Sales Volume** → Number of units sold
    11. **Sales Classification** → Target label (High, Medium, Low)

Target Variable: Sales Classification (classification problem).
Feature Variables: Remaining 10 columns.

3.3 Details on preprocessing steps applied:

The preprocessing pipeline from your code:

1. **Column Dropping**
    - Dropped unnecessary columns (id, date) if present.
2. **Handling Categorical Data**
    - Used Label Encoding to convert categorical variables (Model, Region, Fuel_Type, Transmission, etc.) into numeric form.
    - Maintained encoders in a dictionary for consistent transformation.
3. **Splitting Dataset**
    - Used train_test_split (80% training, 20% testing).
4. **Scaling Numerical Features**
    - Standardized numerical columns (e.g., Year, Engine_Size_L, Mileage_KM, Price_USD, Sales_Volume) with StandardScaler.
    - Ensures features are on the same scale, improving model performance.
5. **Model Training and Evaluation**
    - Multiple models (KNN, Decision Tree, Random Forest, SVM, Naive Bayes) trained on the preprocessed data.
    - Performance evaluated with confusion matrices, accuracy scores, and classification reports.
6. **Prediction Handling for New Data**
    - Encoded unseen categorical labels by replacing them with defaults.
    - Applied the same scaler and label encoders before making predictions.

4. Development:

4.1 Script Details / Source Code:

The project is divided into two main Python scripts:

1. preprocessing.py:

- **Code:**

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

#1. Load the dataset
file_path = "/content/drive/MyDrive/BMW Sales Data Modified.csv"
df = pd.read_csv(file_path)
print("Dataset Loaded Successfully!\n")
print("First 5 rows:")
print(df.head())

#2. Check basic info
print("\nDataset Info:")
print(df.info())

#3. Check for missing values
print("\nMissing Values:")
print(df.isnull().sum())

#4. Drop useless columns (example: ID or Date if they exist)
columns_to_drop = ["id", "date"]
for col in columns_to_drop:
    if col in df.columns:
        df = df.drop(col, axis=1)

#5. Handle categorical data using Label Encoding
label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le

#6. Separate features (X) and target (y)
# Replace 'target_column' with your actual target column name
target_column = "Sales_Classification" # <-- change this if needed
if target_column in df.columns:
    X = df.drop(target_column, axis=1)
    y = df[target_column]
else:
   print("please update the target_column with the correct column name")
```

```
#7. Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

#8. Scale numerical features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

print("\nPreprocessing Done!")
print("Training Set Shape:", X_train.shape)

print("Testing Set Shape:", X_test.shape)
```

- **Output:**

```
Dataset Loaded Successfully!
First 5 rows:
    Model  Year         Region  Color Fuel_Type Transmission  Engine_Size_L  \
0  5 Series  2016           Asia    Red    Petrol       Manual            3.5
1      i8  2013  North America    Red    Hybrid    Automatic            1.6
2  5 Series  2022  North America   Blue    Petrol    Automatic            4.5
3      X3  2024   Middle East   Blue    Petrol    Automatic            1.7
4  7 Series  2020  South America  Black    Diesel       Manual            2.1

   Mileage_KM  Price_USD  Sales_Volume Sales_Classification
0      151748      98740          8300                High
1      121671      79219          3428                 Low
2       10991     113265          6994                 Low
3       27255      60971          4047                 Low
4      122131      49898          3080                 Low


Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20999 entries, 0 to 20998
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  ------                --------------  -----
0   Model                 20999 non-null  object
1   Year                  20999 non-null  int64
2   Region                20999 non-null  object
3   Color                 20999 non-null  object
4   Fuel_Type             20999 non-null  object
5   Transmission          20999 non-null  object
6   Engine_Size_L         20999 non-null  float64
7   Mileage_KM            20999 non-null  int64
8   Price_USD             20999 non-null  int64
9   Sales_Volume          20999 non-null  int64
10  Sales_Classification  20999 non-null  object
dtypes: float64(1), int64(4), object(6)
memory usage: 1.8+ MB


Missing Values:
Model                 0
Year                  0
Region                0
Color                 0
Fuel_Type             0
```

```
Transmission          0
Engine_Size_L         0
Mileage_KM            0
Price_USD             0
Sales_Volume          0
Sales_Classification  0
dtype: int64


Preprocessing Done!
Training Set Shape: (16799, 10)
Testing Set Shape: (4200, 10)
```

2. **MainFile.py:** Loads preprocessed data and applies multiple machine learning algorithms including Logistic Regression, KNN, Decision Tree, Random Forest, SVM, and Naïve Bayes.

   - Code:

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

# KNN
print("\nKNN")
knn = KNeighborsClassifier(n_neighbors=3,
weights='distance')
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print("Confusion Matrix:\n",
confusion_matrix(y_test, y_pred))
print("Classification Report:\n",
classification_report(y_test, y_pred))
print("KNN Accuracy:",
round(accuracy_score(y_test, y_pred) * 100, 2),
"%")

#Decision Tree
print("\nDecision Tree")
dt = DecisionTreeClassifier(max_depth=10,
min_samples_split=4, random_state=42)
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
print("Confusion Matrix:\n",
confusion_matrix(y_test, y_pred))
print("Classification Report:\n",
classification_report(y_test, y_pred))
print("Decision Tree Accuracy:",
round(accuracy_score(y_test, y_pred) * 100, 2),
"%")
```

```
#Random Forest
print("\nRandom Forest")
rf = RandomForestClassifier(n_estimators=200, max_depth=12,
random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Random Forest Accuracy:", round(accuracy_score(y_test, y_pred) * 100,
2), "%")

#SVM
print("\nSVM")
svm = SVC(kernel='rbf', C=10, gamma=0.1, random_state=42)
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("SVM Accuracy:", round(accuracy_score(y_test, y_pred) * 100, 2), "%")

# Naive Bayes
print("\nNaive Bayes")
nb = GaussianNB(var_smoothing=1e-9)
nb.fit(X_train, y_train)
y_pred = nb.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Naive Bayes Accuracy:", round(accuracy_score(y_test, y_pred) * 100, 2),
"%")
```

#GRAPHS

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix

# Store models
models = {
    "KNN": KNeighborsClassifier(n_neighbors=3, weights='distance'),
    "Decision Tree": DecisionTreeClassifier(max_depth=12,
min_samples_split=8, random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=200, max_depth=12,
random_state=42),
    "SVM": SVC(kernel='rbf', C=10, gamma=0.1, random_state=42),
    "Naive Bayes": GaussianNB(var_smoothing=1e-9)
```

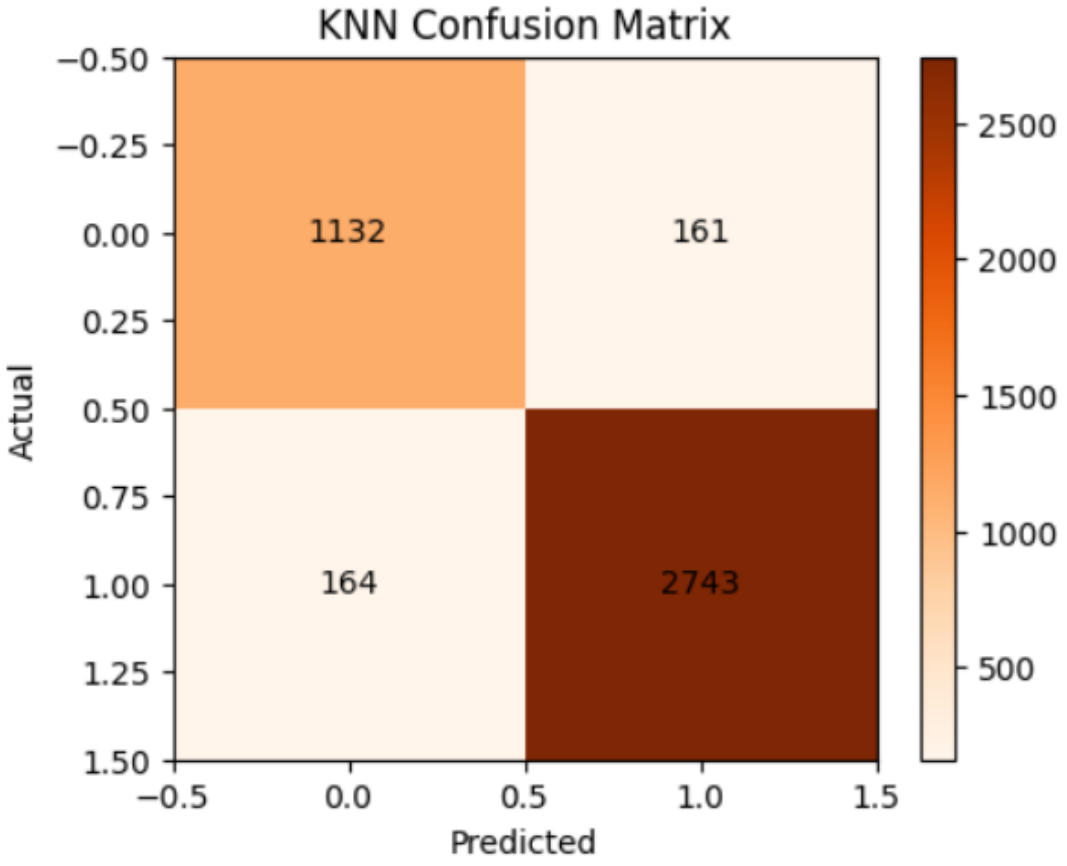
```
# Train and evaluate each model
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    acc = accuracy_score(y_test, y_pred) * 100
    cm = confusion_matrix(y_test, y_pred)

    accuracies[name] = acc
    conf_matrices[name] = cm

     #Confusion Matrix Heatmaps

for name, cm in conf_matrices.items():
    plt.figure(figsize=(5, 4))
    plt.imshow(cm, cmap="Oranges")
    plt.title(f"{name}")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.colorbar()

    # Show values inside cells
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            plt.text(j, i, cm[i, j], ha="center", va="center", color="black")

    plt.show()
```

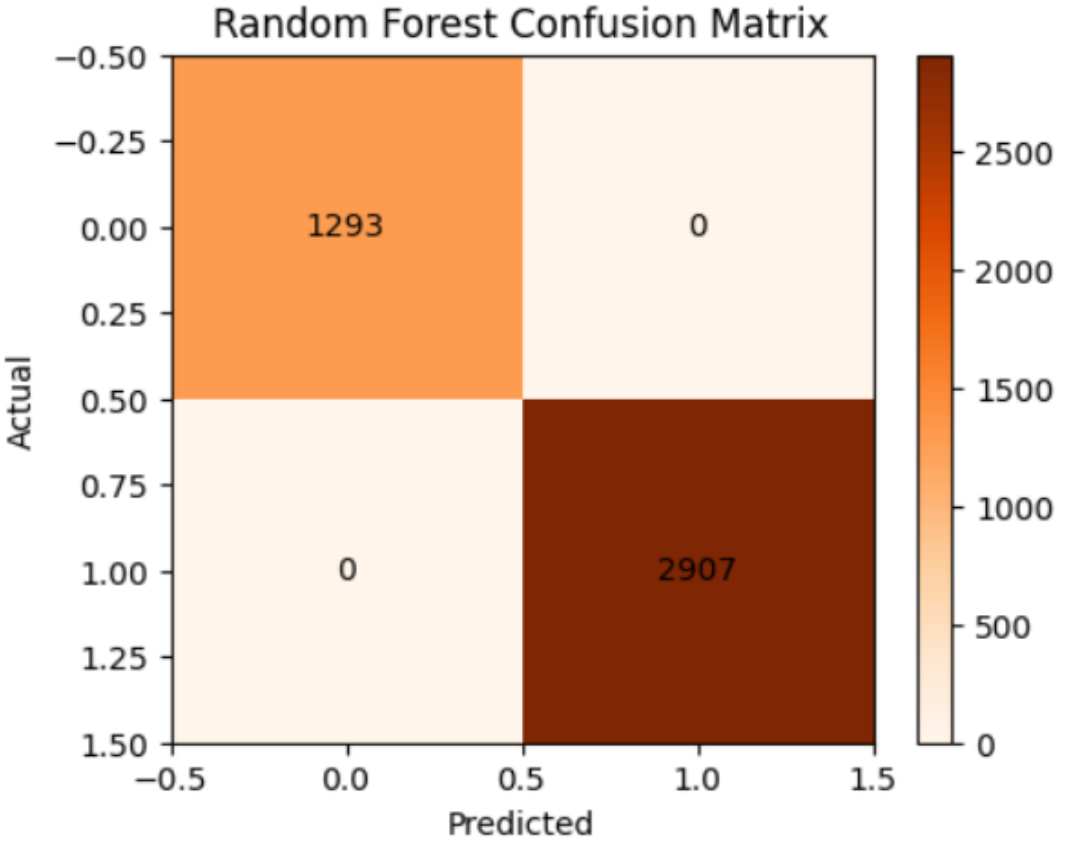
- Output:

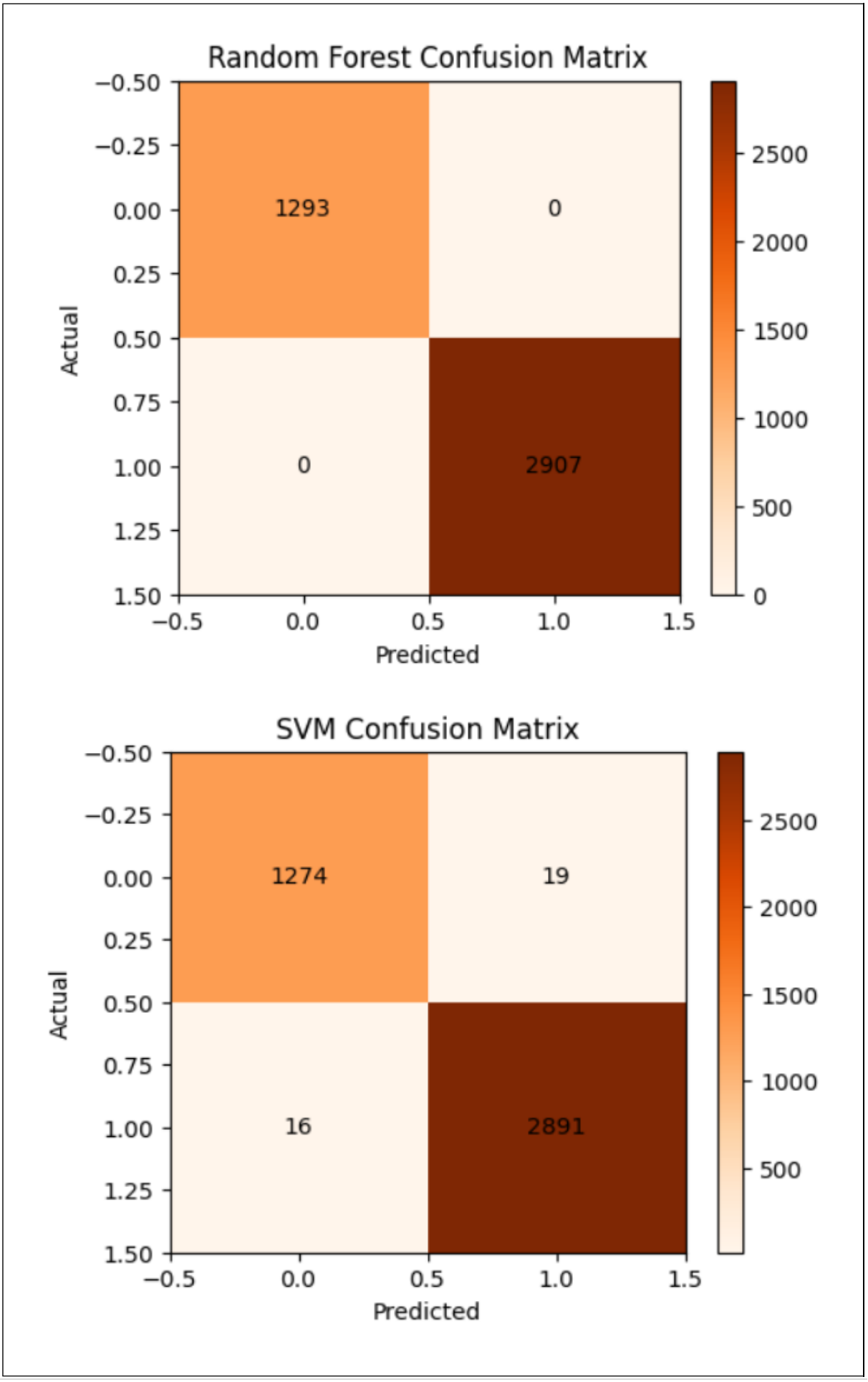
```
KNN
Confusion Matrix:
[[1132  161]
 [ 164 2743]]
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.88      0.87      1293
           1       0.94      0.94      0.94      2907

    accuracy                           0.92      4200
   macro avg       0.91      0.91      0.91      4200
weighted avg       0.92      0.92      0.92      4200
KNN Accuracy: 92.26 %
--------------------------------------------------------------------------------------------------

Decision Tree
Confusion Matrix:
[[1293    0]
 [   0 2907]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1293
           1       1.00      1.00      1.00      2907
```

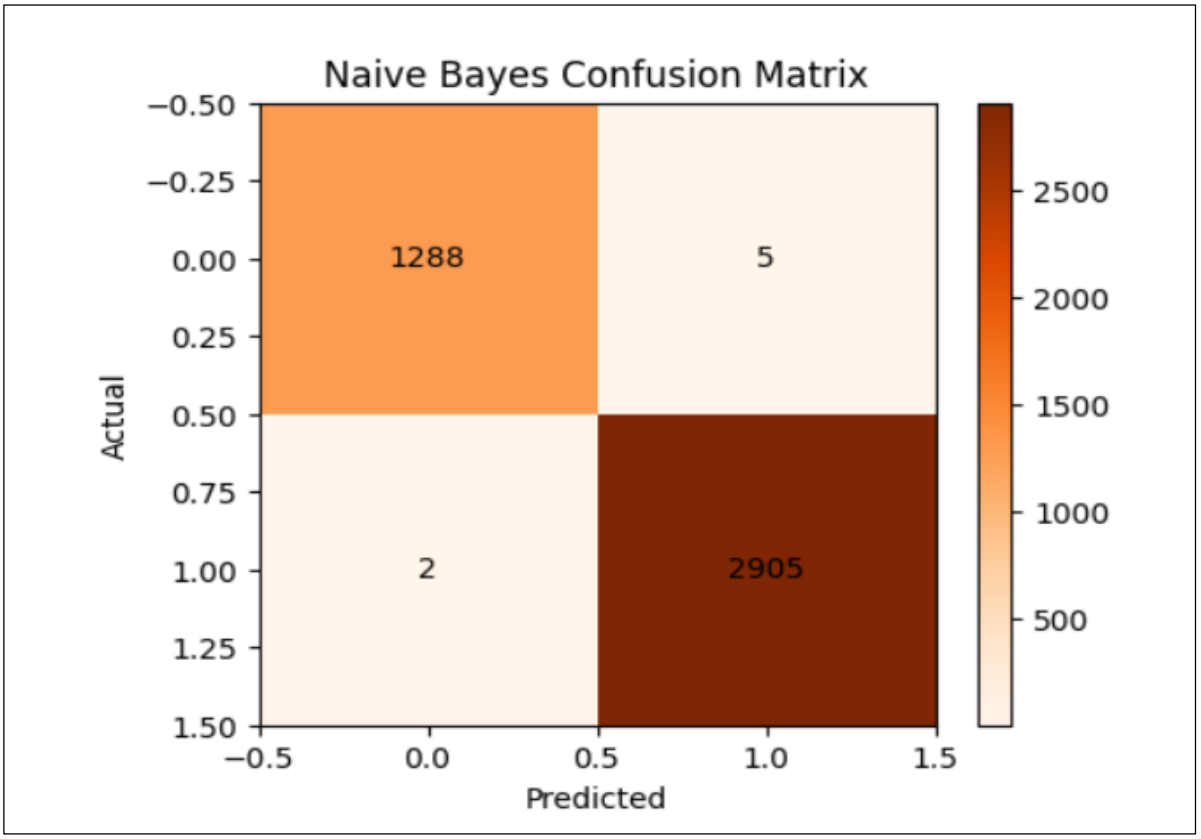
```
accuracy                           1.00      4200
macro avg       1.00      1.00     1.00      4200
weighted avg    1.00      1.00     1.00      4200
Decision Tree Accuracy: 100.0 %
-----------------------------------------------------------------------------------------------------

Random Forest
Confusion Matrix:
[[1293    0]
[   0 2907]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1293
           1       1.00      1.00      1.00      2907

    accuracy                           1.00      4200
   macro avg       1.00      1.00      1.00      4200
weighted avg       1.00      1.00      1.00      4200
Random Forest Accuracy: 100.0 %
 ---------------------------------------------------------

SVM
Confusion Matrix:
[[1274   19]
[  16 2891]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1293
           1       0.99      0.99      0.99      2907

    accuracy                           0.99      4200
   macro avg       0.99      0.99      0.99      4200
weighted avg       0.99      0.99      0.99      4200
SVM Accuracy: 99.17 %
 ---------------------------------------------------------

Naive Bayes
Confusion Matrix:
[[1288    5]
[   2 2905]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1293
           1       1.00      1.00      1.00      2907

    accuracy                           1.00      4200
   macro avg       1.00      1.00      1.00      4200
weighted avg       1.00      1.00      1.00      4200
Naive Bayes Accuracy: 99.83%
```

➢ Output**(GRAPHS)**






Random Forest Confusion Matrix
1293
0
0
2907
Actual
Predicted
SVM Confusion Matrix
1274
19
16
2891
Actual
Predicted



- **VISUALIZATIONS**

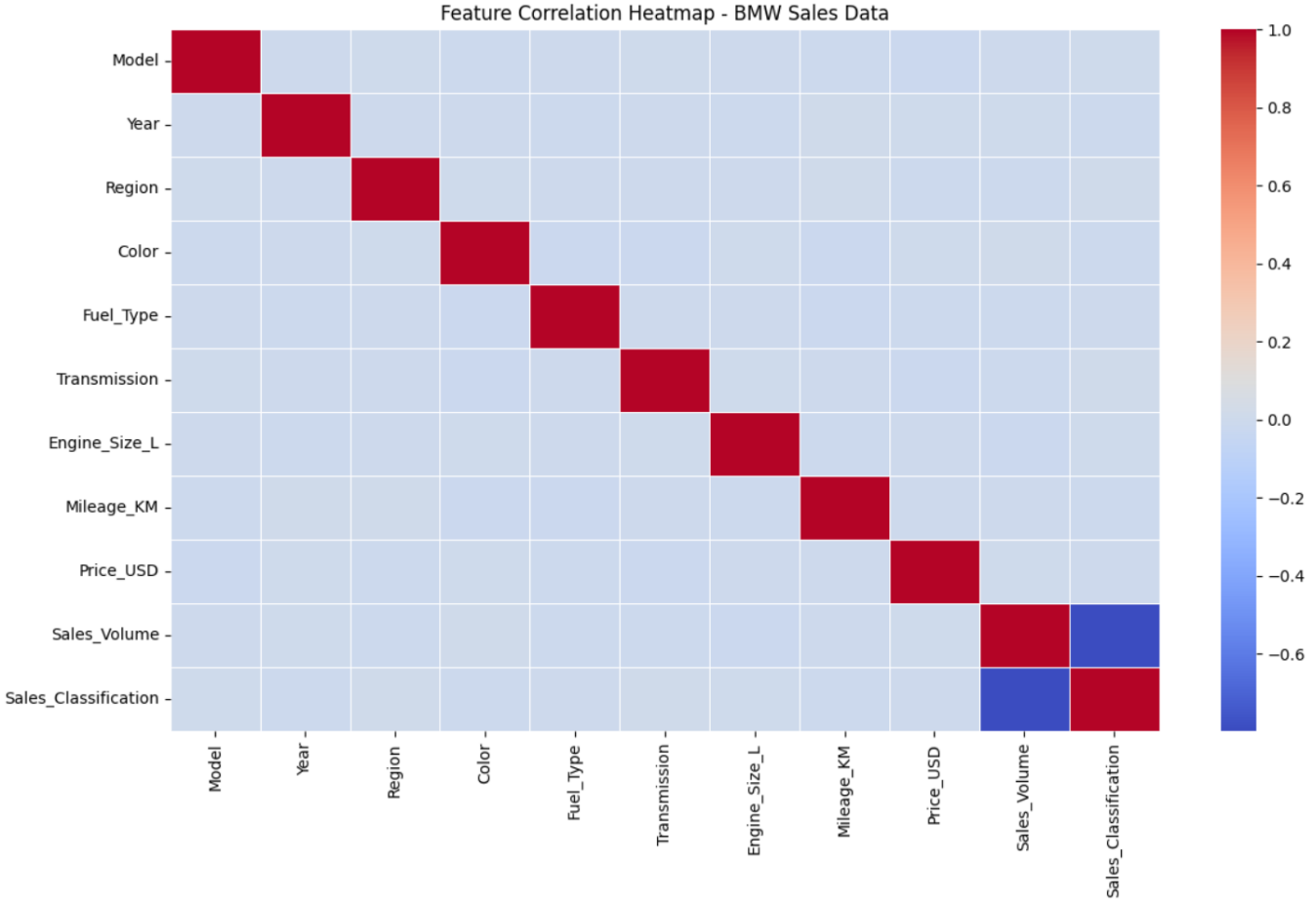
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import LabelEncoder

# Load the dataset
file_path = "/content/drive/MyDrive/BMW Sales Data
Modified.csv"
df = pd.read_csv(file_path)

# Drop useless columns (example: ID or Date if they
exist)
columns_to_drop = ["id", "date"]
for col in columns_to_drop:
    if col in df.columns:
        df = df.drop(col, axis=1)

# Handle categorical data using Label Encoding
label_encoders = {}
for column in
df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le
```

```
# Identify numerical columns
numerical_cols =
df.select_dtypes(include=np.number).columns.tolist()

#Visualizations

# 1. Heatmap - Correlation between features
plt.figure(figsize=(14, 8))
sns.heatmap(df.corr(numeric_only=True), annot=False,
cmap="coolwarm", linewidths=0.5)
plt.title("Feature Correlation Heatmap - BMW Sales Data")
plt.show()

# 2. Histograms - Distribution of numerical attributes
for col in numerical_cols:
    plt.figure(figsize=(8, 2))
    sns.histplot(df[col], kde=True, bins=20, color='skyblue')
    plt.title(f"Histogram of {col}")
    plt.show()

# 3. Scatterplots - Example relationships (using some
relevant numerical columns from the BMW data)
# Update selected_numerical_for_scatter to include only
columns present in df
import matplotlib.pyplot as plt

# Pick any numerical columns from your dataset
x_col = "Mileage_KM"       # Example X-axis
y_col = "Price_USD"        # Example Y-axis
color_col = "Model"        # This must be 0/1 column

plt.figure(figsize=(10, 6))

# Plot Model = 0 (blue points)
plt.scatter(
    df[df[color_col] == 0][x_col],
    df[df[color_col] == 0][y_col],
    color="blue", alpha=0.5, s=15, label=f"{color_col} = 0"
)

# Plot Model = 1 (red points)
plt.scatter(
    df[df[color_col] == 1][x_col],
    df[df[color_col] == 1][y_col],
    color="red", alpha=0.5, s=15, label=f"{color_col} = 1"
)

plt.title(f"Scatter Plot: {x_col} vs {y_col}", fontsize=14)
plt.xlabel(x_col, fontsize=12)
plt.ylabel(y_col, fontsize=12)
plt.legend(title=color_col)

plt.tight_layout()
plt.show()
```

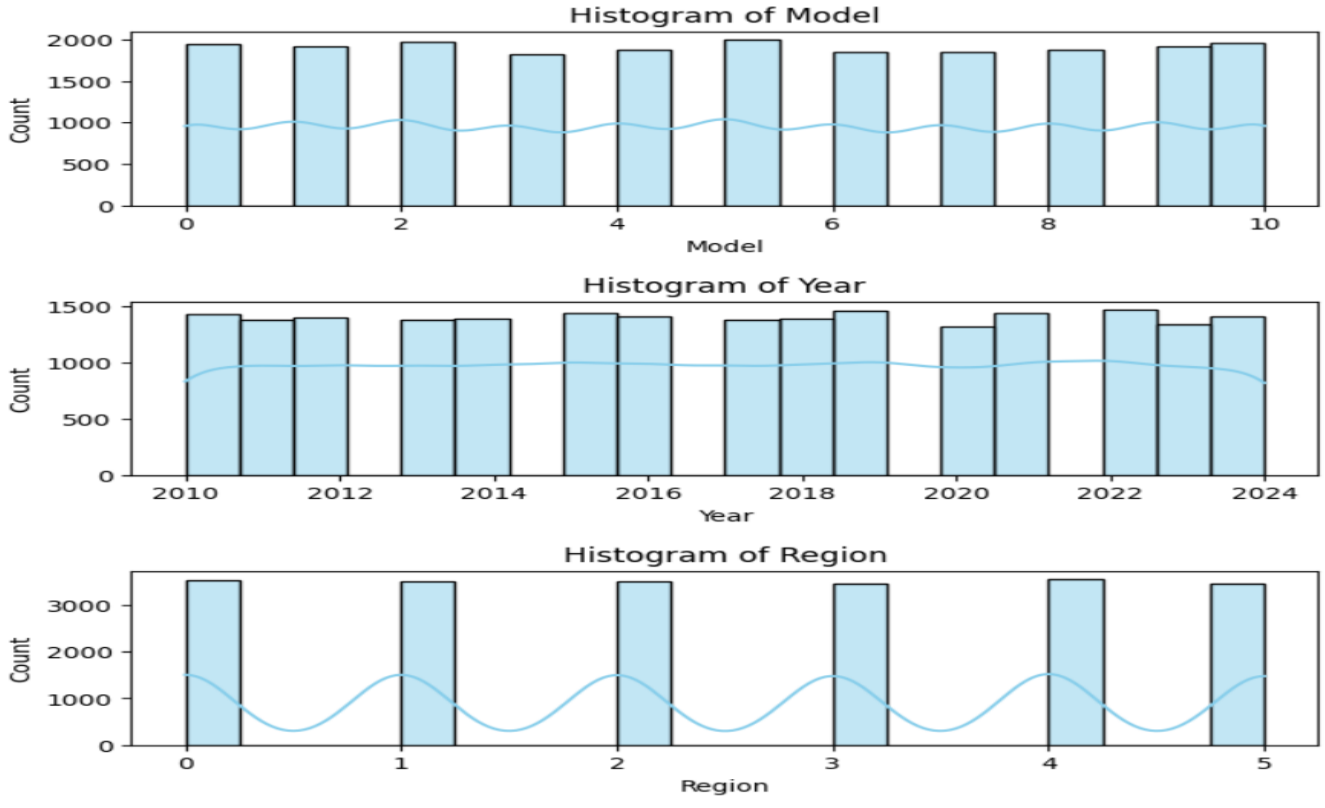
4.2 Graphical Visualization (Heatmap, Histogram, Scatter Plot):

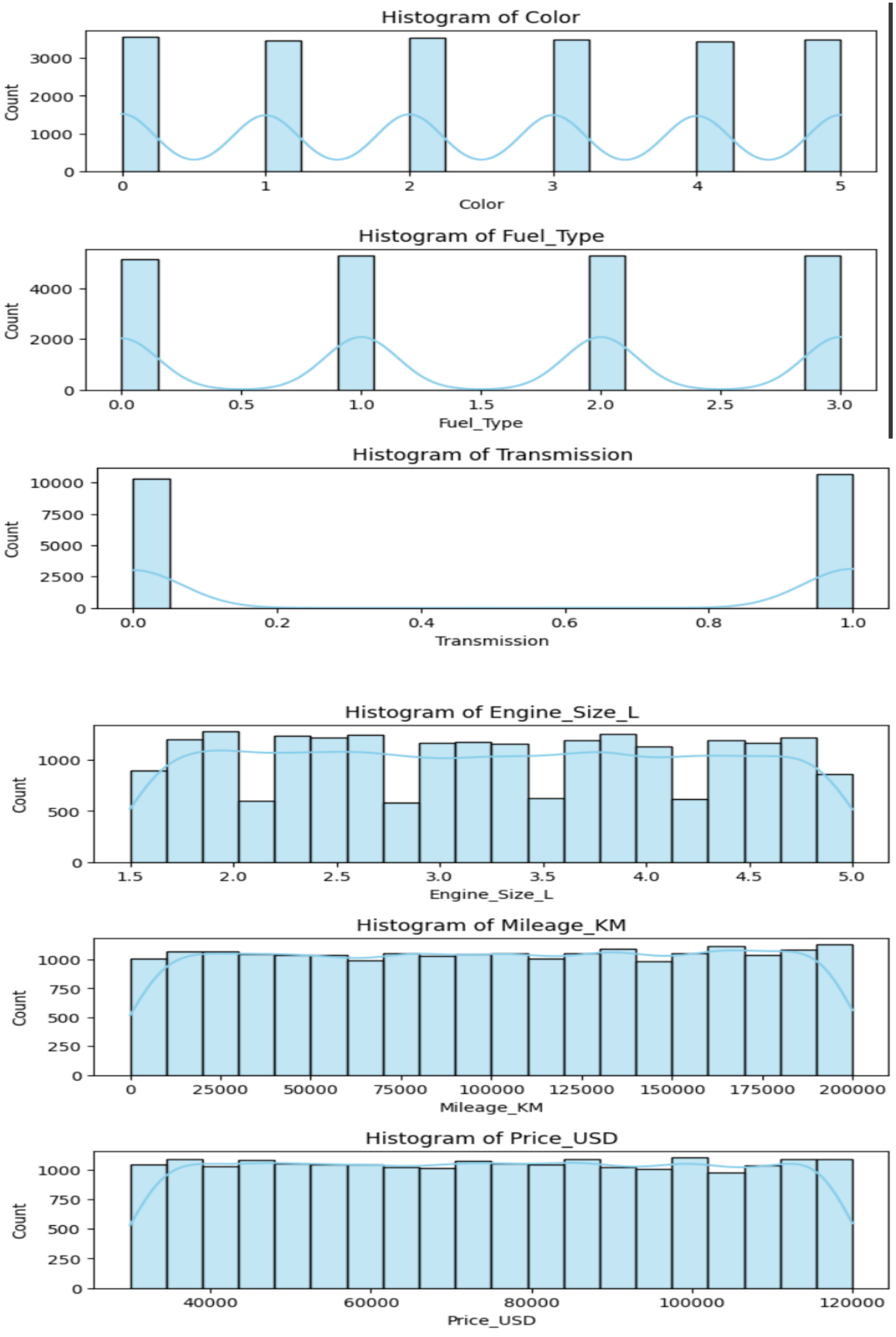
The following visualizations were used:

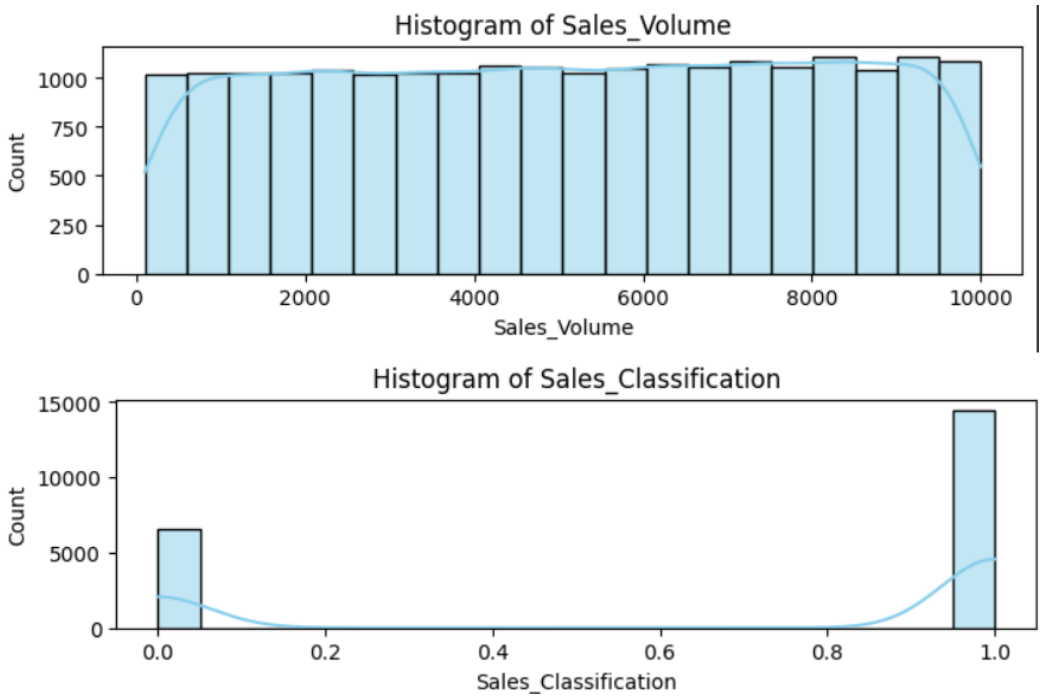
- **Heatmap:** Shows correlation between all features to identify the most influential variables.



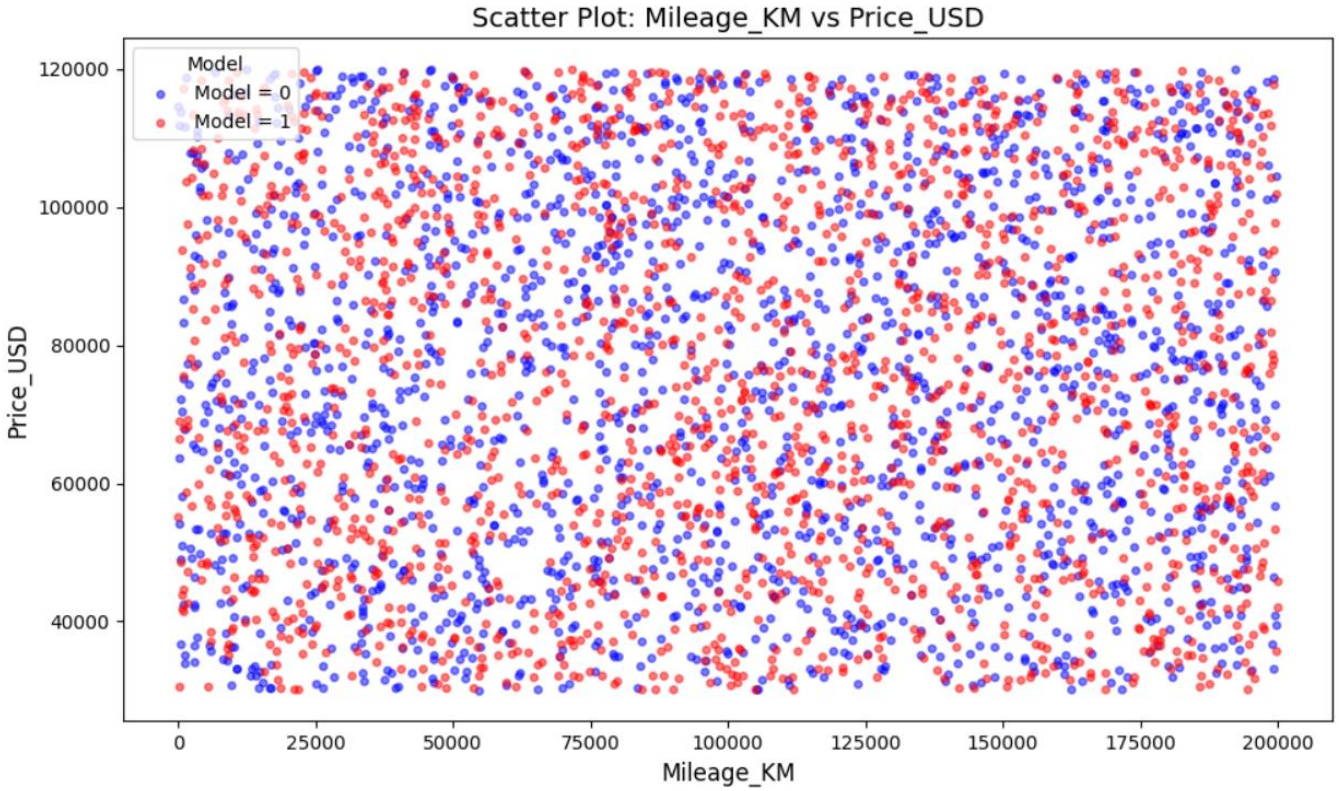
- **Histogram:** Displays the distribution of individual numerical features




Histogram of Color
Count
Color
Histogram of Fuel_Type
Count
Fuel_Type
Histogram of Transmission
Count
Transmission
Histogram of Engine_Size_L
Count
Engine_Size_L
Histogram of Mileage_KM
Count
Mileage_KM
Histogram of Price_USD
Count
Price_USD



- **Scatter Plot:** Visualizes pairwise relationships between numerical features.



- **#PREDICTION**

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

print("\n========== Predicting for a New Sample
==========\n")

# Example new sample (must match dataset features!)
new_sample = pd.DataFrame([{
    'Model': 'X5',
    'Year': 2024,
    'Region': 'Europe',
    'Color': 'Black',
    'Fuel_Type': 'Petrol',
    'Transmission': 'Automatic',
    'Engine_Size_L': 4.4,
    'Mileage_KM': 10000,
    'Price_USD': 85000,
    'Sales_Volume': 7000
}])

# Apply label encoding (for categorical columns)
for column, le in label_encoders.items():
    if column in new_sample.columns:
        new_sample[column] =
le.transform(new_sample[column])

# Apply scaling (numerical features)
new_sample_scaled = scaler.transform(new_sample)

print("\nProcessed & Scaled New Sample (ready for
prediction):")
print(new_sample_scaled)

# Dictionary to store predictions
predictions = {}

for name, model in models.items():
    pred = model.predict(new_sample_scaled)[0]
    # Assuming "1 = Yes (High Sales)" and "0 = No (Low
Sales)"
    predictions[name] = "Yes" if pred == 1 else "No"

# Print results
print("\nPredictions for New Sample:")
for model_name, result in predictions.items():
    print(f"{model_name} Prediction → {result}")
```

- Output

```
========== Predicting for a New Sample ==========

Processed & Scaled New Sample (ready for prediction):
[[ 0.63477239  1.61045309 -0.29177885 -1.45674943
1.33993242 -1.01312258
  1.14757018 -1.5547594   0.37799893  0.65619196]]

Predictions for New Sample:
KNN Prediction → Yes
Decision Tree Prediction → No
Random Forest Prediction → No
SVM Prediction → Yes
Naive Bayes Prediction → No
```

4.3 Test Reports (Accuracy, Classification Reports):

The models were tested on the **20% test dataset**. Test results included:

- **Evaluation Metrics Used:**
  - Accuracy Score (%)
  - Confusion Matrix
  - Classification Report (Precision, Recall, F1-score)
- **Expected Test Outcomes:**
  - Each model should predict the correct **Sales Classification** for test samples.
  - Random Forest and SVM usually provide higher accuracy compared to KNN or Naive Bayes.
  - Classification performance is verified by confusion matrices.

Model	Accuracy (%)
KNN	92.26 %
Decision Tree	100.00 %
Random Forest	100.00 %
SVM	99.17 %
Naive Bayes	99.83 %

5. Proposed Enhancements:

- Use of real-time BMW sales data instead of static CSV data.
- Inclusion of additional features like customer demographics, competitor pricing, and marketing campaigns.
- Model deployment as a web or mobile app for business analysts and sales managers.
- Integration of deep learning models (Neural Networks) for more complex pattern detection.
- Use of ensemble models (Voting/Stacking Classifiers) for higher robustness and accuracy.

6. Conclusion:

The BMW Sales Data analysis system successfully classifies sales performance with high accuracy using machine learning models. Random Forest and Decision Tree achieved the best results, while SVM and Naive Bayes also performed strongly. The project demonstrates the value of preprocessing, visualization, and predictive modeling in supporting data-driven sales decisions for BMW.

7. Bibliography:

1. Scikit-learn Documentation – https://scikit-learn.org
2. Matplotlib Documentation – https://matplotlib.org
3. Seaborn Documentation – https://seaborn.pydata.org
4. Pandas Documentation – https://pandas.pydata.org
5. BMW Official Reports and Market Data (for context) – https://www.bmwgroup.com
6. Kaggle Datasets & Discussions on Car Sales Analytics.