IE7275 Data Mining in Engineering

Project Report

Spring Semester 2025

Group Number – 15

Yaswanth Kumar Reddy Gujjula (002415723)

Nishchay Linge Gowda (002309438)

Yash Harale (002080348)

Chirag Verma (002308890)

Submission Date: 20th April 2025

Problem Statement:

The automotive industry constantly needs accurate predictions and insights to enhance business decisions. This project aims to develop a predictive model that estimates the resale value of vehicles based on various features such as age, mileage, and horsepower. Understanding these factors' influence on pricing can help stakeholders make informed decisions about buying, selling, and maintaining inventory. The project explores comprehensive data exploration, rigorous outlier detection, and advanced modeling techniques to achieve a reliable model that stakeholders can use to predict prices effectively and manage their assets more efficiently.

Contents:

Data Exploration:

```python
import pandas as pd

resale_df=pd.read_csv('ToyotaCorolla.csv')
resale_df.head()
```

Python

Price	Age_08_04	Mfg_Month	Mfg_Year	KM	Fuel_Type	HP	Met_Color	...	Central_Lock	Powered_Windows	Power_Steering	Radio	Mistlamps	Sport_Model	Backseat_Divider	Metallic_Rim	Radio_cassette	Tow_Bar
13500	23	10	2002	46986	Diesel	90	1	...	1	1	1	0	0	0	1	0	0	0
13750	23	10	2002	72937	Diesel	90	1	...	1	0	1	0	0	0	1	0	0	0
13950	24	9	2002	41711	Diesel	90	1	...	0	0	1	0	0	0	1	0	0	0
14950	26	7	2002	48000	Diesel	90	0	...	0	0	1	0	0	0	1	0	0	0
13750	30	3	2002	38500	Diesel	90	0	...	1	1	1	0	1	0	1	0	0	0

```python
resale_df.columns
```

Python

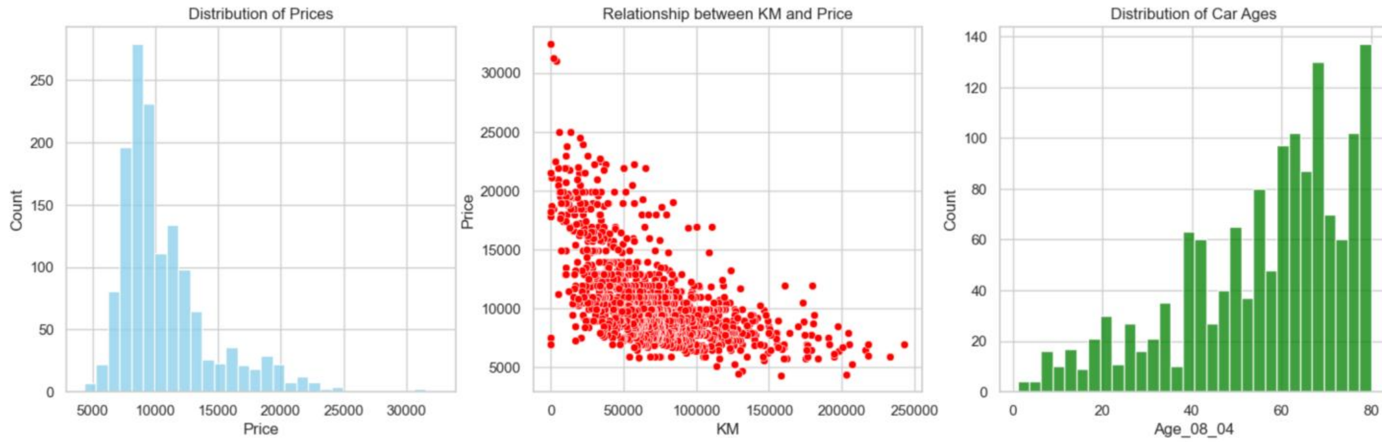
```
Index(['Id', 'Model', 'Price', 'Age_08_04', 'Mfg_Month', 'Mfg_Year', 'KM',
       'Fuel_Type', 'HP', 'Met_Color', 'Automatic', 'cc', 'Doors', 'Cylinders',
       'Gears', 'Quarterly_Tax', 'Weight', 'Mfr_Guarantee', 'BOVAG_Guarantee',
       'Guarantee_Period', 'ABS', 'Airbag_1', 'Airbag_2', 'Airco',
       'Automatic_airco', 'Boardcomputer', 'CD_Player', 'Central_Lock',
       'Powered_Windows', 'Power_Steering', 'Radio', 'Mistlamps',
       'Sport_Model', 'Backseat_Divider', 'Metallic_Rim', 'Radio_cassette',
       'Tow_Bar'],
      dtype='object')
```

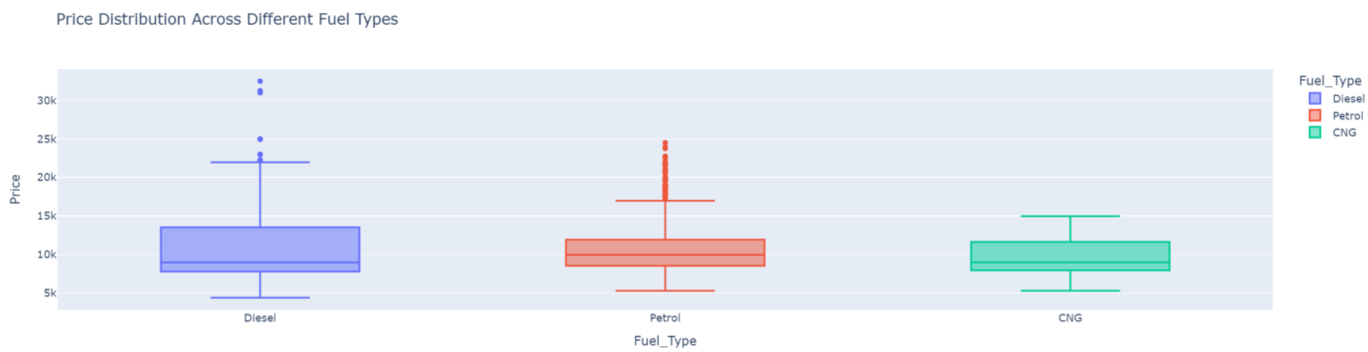
```python
descriptive_stats = resale_df.describe()
descriptive_stats
```

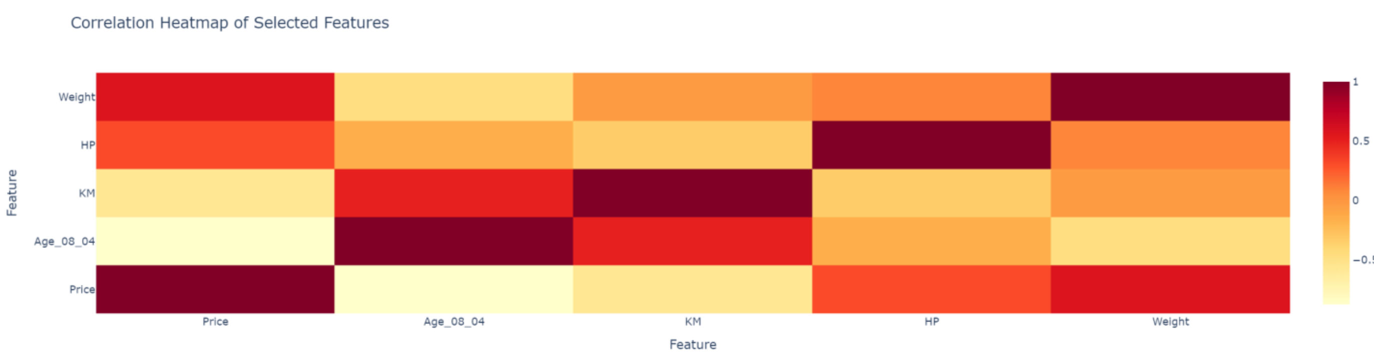
Python

	Id	Price	Age_08_04	Mfg_Month	Mfg_Year	KM	HP	Met_Color	Automatic	cc	...	Central_Lock	Powered_Windows	Power_Steering	Radio	Mistlamps
count	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000	1436.00000	...	1436.000000	1436.000000	1436.000000	1436.000000	1436.000000
mean	721.555014	10730.824513	55.947075	5.548747	1999.625348	68533.259749	101.502089	0.674791	0.055710	1576.85585	...	0.580084	0.561978	0.977716	0.146240	0.256964
std	416.476890	3626.964585	18.599988	3.354085	1.540722	37506.448872	14.981080	0.468616	0.229441	424.38677	...	0.493717	0.496317	0.147657	0.353469	0.437111
min	1.000000	4350.000000	1.000000	1.000000	1998.000000	1.000000	69.000000	0.000000	0.000000	1300.00000	...	0.000000	0.000000	0.000000	0.000000	0.000000
25%	361.750000	8450.000000	44.000000	3.000000	1998.000000	43000.000000	90.000000	0.000000	0.000000	1400.00000	...	0.000000	0.000000	1.000000	0.000000	0.000000
50%	721.500000	9900.000000	61.000000	5.000000	1999.000000	63389.500000	110.000000	1.000000	0.000000	1600.00000	...	1.000000	1.000000	1.000000	0.000000	0.000000
75%	1081.250000	11950.000000	70.000000	8.000000	2001.000000	87020.750000	110.000000	1.000000	0.000000	1600.00000	...	1.000000	1.000000	1.000000	0.000000	1.000000
max	1442.000000	32500.000000	80.000000	12.000000	2004.000000	243000.000000	192.000000	1.000000	1.000000	16000.00000	...	1.000000	1.000000	1.000000	1.000000	1.000000

Visualization:







Data Processing:

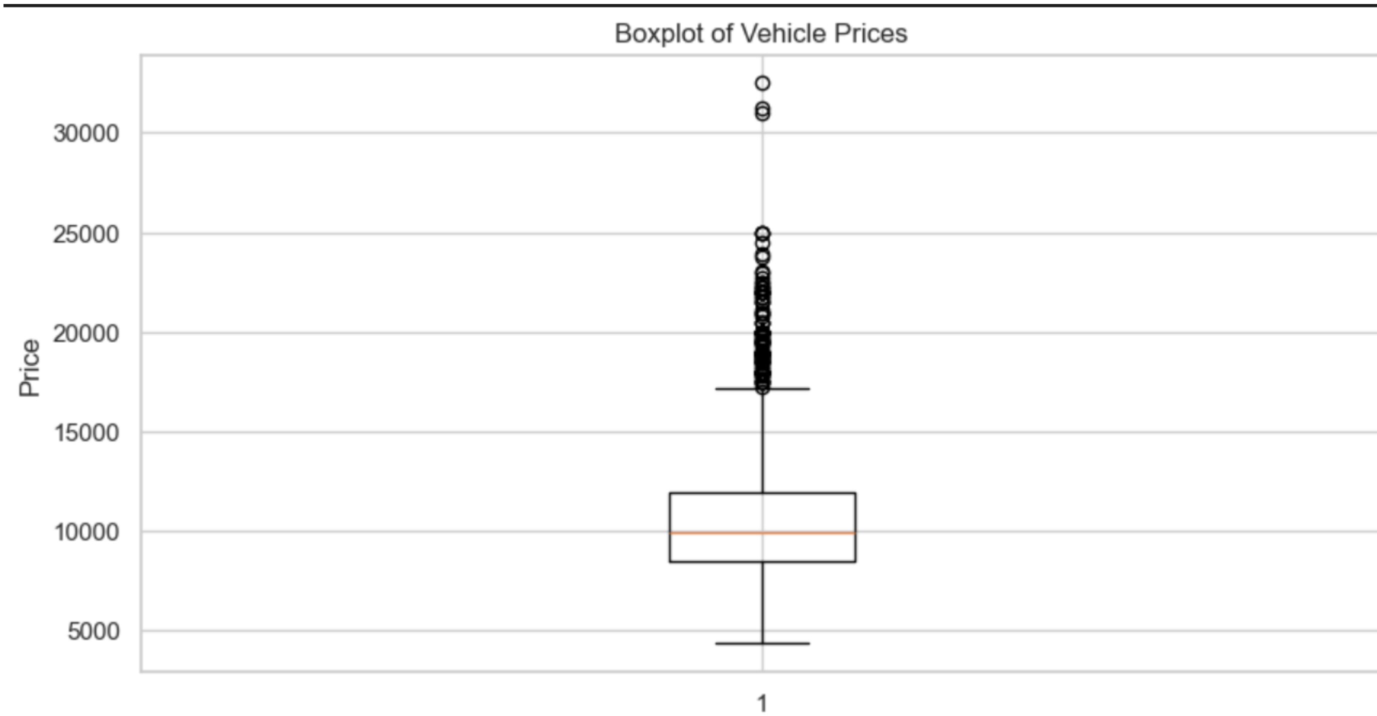
```
#Features that we are using for the prediction
current_year = 2024
current_month = 4

# Calculate Age in months
resale_df['Age_in_months'] = (current_year - resale_df['Mfg_Year']) * 12 + (current_month - resale_df['Mfg_Month'])

# Original feature list
requested_features = ['Price', 'Age_in_months', 'KM', 'Fuel_Type', 'HP', 'Met_Color', 'Automatic', 'CC', 'Doors', 'Quarterly_Tax', 'Weight']

# Filter to only use columns that exist in the DataFrame
features = [feature for feature in requested_features if feature in resale_df.columns]

# Now select those columns
resale_selected_df = resale_df[features]

# Display the selected features
print(resale_selected_df.head())
```

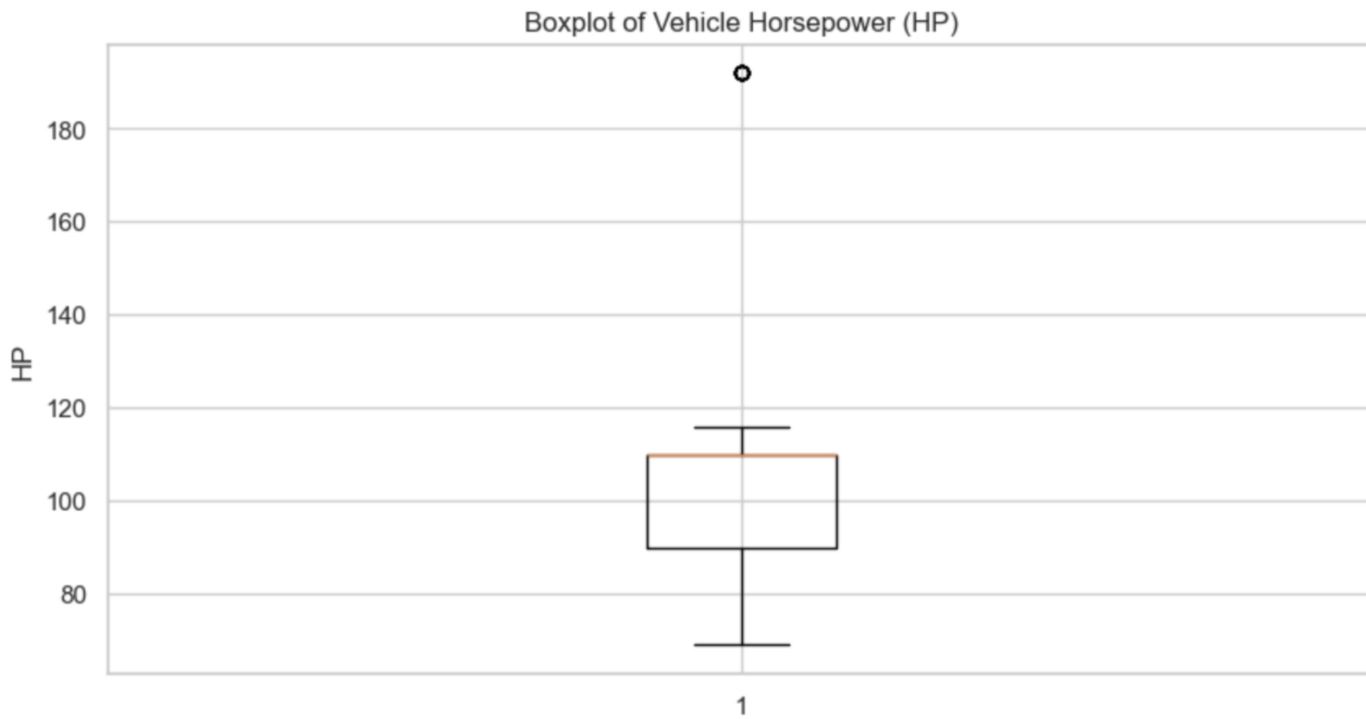
```
   Price  Age_in_months     KM Fuel_Type  HP  Met_Color  Automatic  Doors  \
0  13500            258  46986    Diesel  90          1          0      3
1  13750            258  72937    Diesel  90          1          0      3
2  13950            259  41711    Diesel  90          1          0      3
3  14950            261  48000    Diesel  90          0          0      3
4  13750            265  38500    Diesel  90          0          0      3

   Quarterly_Tax  Weight
0            210    1165
1            210    1165
2            210    1165
3            210    1165
4            210    1170
```

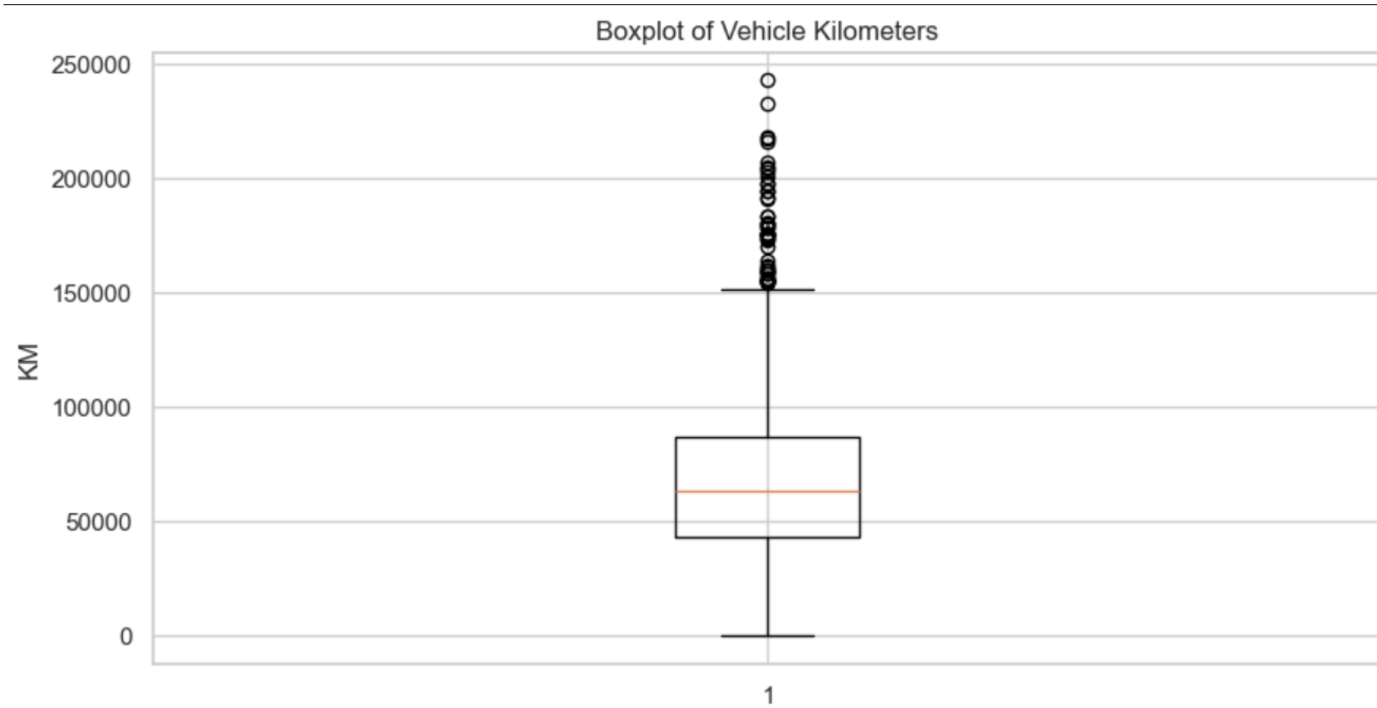
```
statistical_summary = resale_prepared_df.describe()
# Focusing on 'Price', 'Age', 'KM' (mileage), and 'HP' (Horse Power) as they are likely to affect the resale value
relevant_columns = ['Price', 'Age_in_months', 'KM', 'HP']

Q1 = resale_prepared_df[relevant_columns].quantile(0.25)
Q3 = resale_prepared_df[relevant_columns].quantile(0.75)
IQR = Q3 - Q1

outliers = ((resale_prepared_df[relevant_columns] < (Q1 - 1.5 * IQR)) |
            (resale_prepared_df[relevant_columns] > (Q3 + 1.5 * IQR))).sum()
statistical_summary, outliers
```

```
(              Price  Age_in_months             KM           HP    Met_Color  \
 count   1436.000000    1436.000000    1436.000000  1436.000000  1436.000000
 mean   10730.824513     290.947075   68533.259749   101.502089     0.674791
 std     3626.964585      18.599988   37506.448872    14.981080     0.468616
 min     4350.000000     236.000000       1.000000    69.000000     0.000000
 25%     8450.000000     279.000000   43000.000000    90.000000     0.000000
 50%     9900.000000     296.000000   63389.500000   110.000000     1.000000
 75%    11950.000000     305.000000   87020.750000   110.000000     1.000000
 max    32500.000000     315.000000  243000.000000   192.000000     1.000000

          Automatic        Doors  Quarterly_Tax      Weight
 count  1436.000000  1436.000000    1436.000000  1436.00000
 mean      0.055710     4.033426      87.122563  1072.45961
 std       0.229441     0.952677      41.128611    52.64112
 min       0.000000     2.000000      19.000000  1000.00000
 25%       0.000000     3.000000      69.000000  1040.00000
 50%       0.000000     4.000000      85.000000  1070.00000
 75%       0.000000     5.000000      85.000000  1085.00000
 max       1.000000     5.000000     283.000000  1615.00000  ,
 Price            110
 Age_in_months      7
 KM                49
 HP                11
 dtype: int64)
```

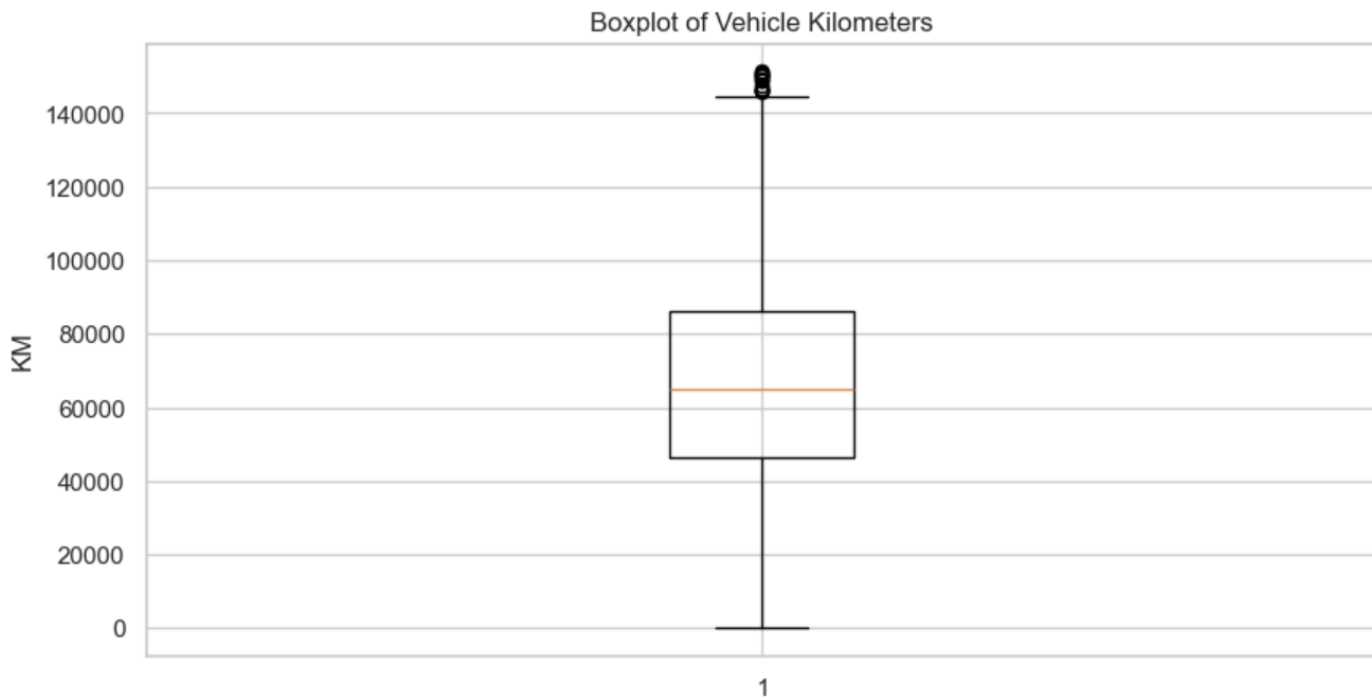
We conducted a preliminary analysis of the numerical features in the dataset to identify potential outliers. We calculated the descriptive statistics, including mean, standard deviation, minimum, 25th percentile (Q1), median (50th percentile), 75th percentile (Q3), and maximum values, for the features 'Price', 'Age_in_months', 'KM' (mileage), and 'HP' (Horsepower). Next, we calculated the Interquartile Range (IQR) for each of these features, which is the range between the first and third quartiles (Q1 and Q3), to determine the spread of the middle 50% of the data. Using the IQR, we identified potential outliers as values that fall below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR for each feature. The count of outliers for each feature was computed and presented alongside the descriptive statistics. This analysis helps in understanding the distribution of the numerical features and detecting any extreme values that may require further investigation or treatment.

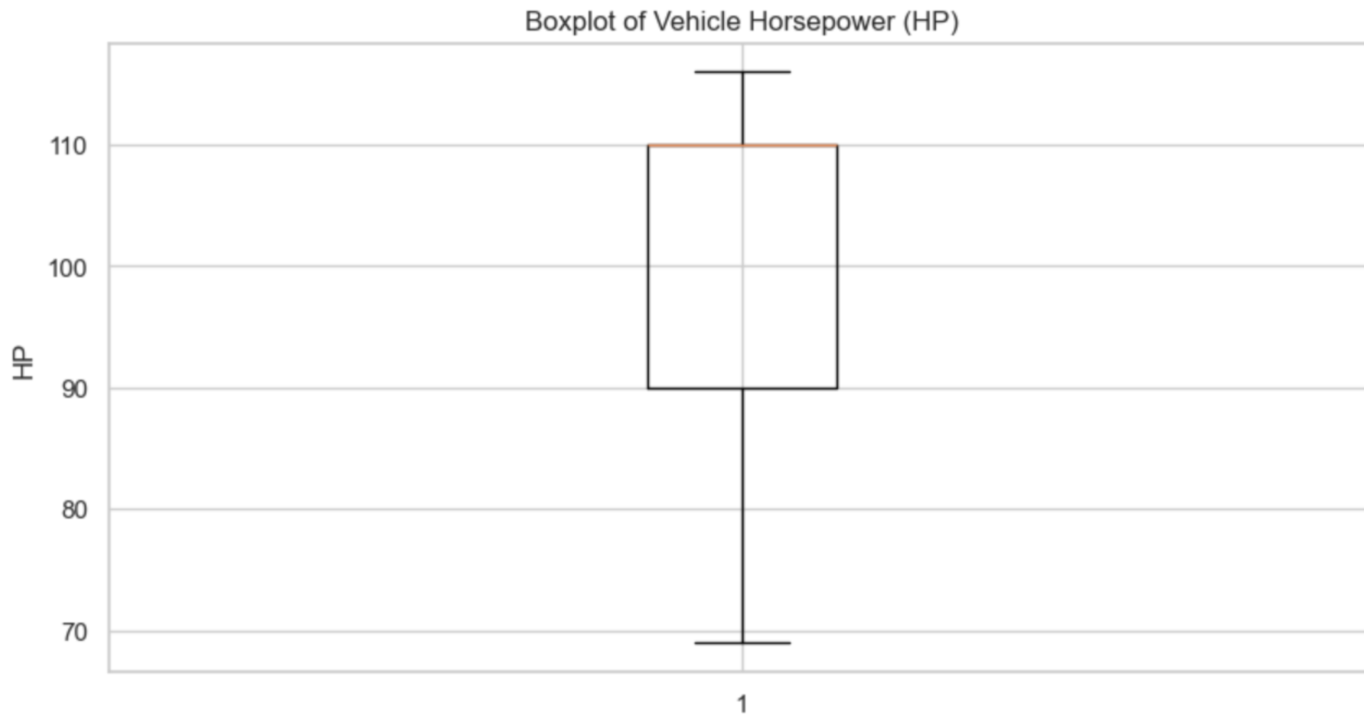






Here, we created boxplots to visually inspect potential outliers for the numerical features 'Price', 'KM' (kilometers), and 'HP' (horsepower) in the dataset resale_prepared_df. Boxplots are effective tools for identifying outliers as they display the distribution of data, including outliers, quartiles, and the median. By plotting these boxplots, we aimed to gain insights into the distribution of these variables and identify any extreme values that may indicate outliers. This visual analysis complements the earlier statistical summary and provides a clearer understanding of the data distribution, aiding in the detection of potential outliers.





We defined a function find_outliers_iqr() to identify outlier indices based on the Interquartile Range (IQR) for a given DataFrame column. We then used this function to find outliers for the numerical features 'Price', 'KM', and 'HP' in the resale_prepared_df DataFrame. Next, we combined all outlier indices across these features and removed the corresponding rows from the DataFrame to create a filtered DataFrame df_filtered. Finally, we generated boxplots for 'Price', 'KM', and 'HP' in the filtered DataFrame to visually inspect the distribution of these variables after removing outliers. This approach allows us to handle outliers more systematically by using the IQR method and provides a clearer representation of the data distribution without the influence of extreme values.

Model Exploration and Selection:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.metrics import r2_score


categorical_cols = resale_df.select_dtypes(include=['object']).columns
print("Categorical columns:", categorical_cols)

resale_df_encoded = pd.get_dummies(resale_df, columns=categorical_cols)

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

X = resale_df_encoded.drop('Price', axis=1)
y = resale_df_encoded['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#Linear Regression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lin = lin_reg.predict(X_test)
mse_lin = mean_squared_error(y_test, y_pred_lin)
mae_lin_scaled = mean_absolute_error(y_test, y_pred_lin)
r2_lin = r2_score(y_test, y_pred_lin)

print("Linear Regression MSE with encoded data:", mse_lin)
print("Linear Regression MAE with scaled data:", mae_lin_scaled)
print("R-squared for Linear Regression:", r2_lin)
```

```
Categorical columns: Index(['Model', 'Fuel_Type'], dtype='object')
Linear Regression MSE with encoded data: 2.1400181960636756e+16
Linear Regression MAE with scaled data: 35242241.486192495
R-squared for Linear Regression: -1603876235.918666
```

```python
# Random Forest Regressor
rf_reg = RandomForestRegressor(n_estimators=100, random_state=42)
rf_reg.fit(X_train, y_train)
y_pred_rf = rf_reg.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
mae_ridge = mean_absolute_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

print("Random Forest MSE:", mse_rf)
print("Random Forest MAE:", mae_ridge)
print("R-squared for Random Forest:", r2_rf)
```

```
Random Forest MSE: 916826.2125684028
Random Forest MAE: 746.4905208333334
R-squared for Random Forest: 0.9312867629617646
```

```python
# Gradient Boosting Regressor
gb_reg = GradientBoostingRegressor(n_estimators=100, random_state=42)
gb_reg.fit(X_train, y_train)
y_pred_gb = gb_reg.predict(X_test)
mse_gb = mean_squared_error(y_test, y_pred_gb)
mae_gb = mean_absolute_error(y_test, y_pred_gb)
r2_gb = r2_score(y_test, y_pred_gb)

print("Gradient Boosting MSE:", mse_gb)
print("Gradient Boosting MAE:", mae_gb)
print("R-squared for Gradient Boosting:", r2_gb)
```

```
Gradient Boosting MSE: 874503.278718984
Gradient Boosting MAE: 719.1782331220406
R-squared for Gradient Boosting: 0.9344587335554082
```

```python
rmse_gb = np.sqrt(mse_gb)
print("Gradient Boosting RMSE:", rmse_gb)
```

```
Gradient Boosting RMSE: 935.148800308798
```

Both Random Forest and Gradient Boosting models perform well, but the Gradient Boosting model edges out slightly better across all three key metrics: it has a lower MSE and MAE, and a slightly higher R-

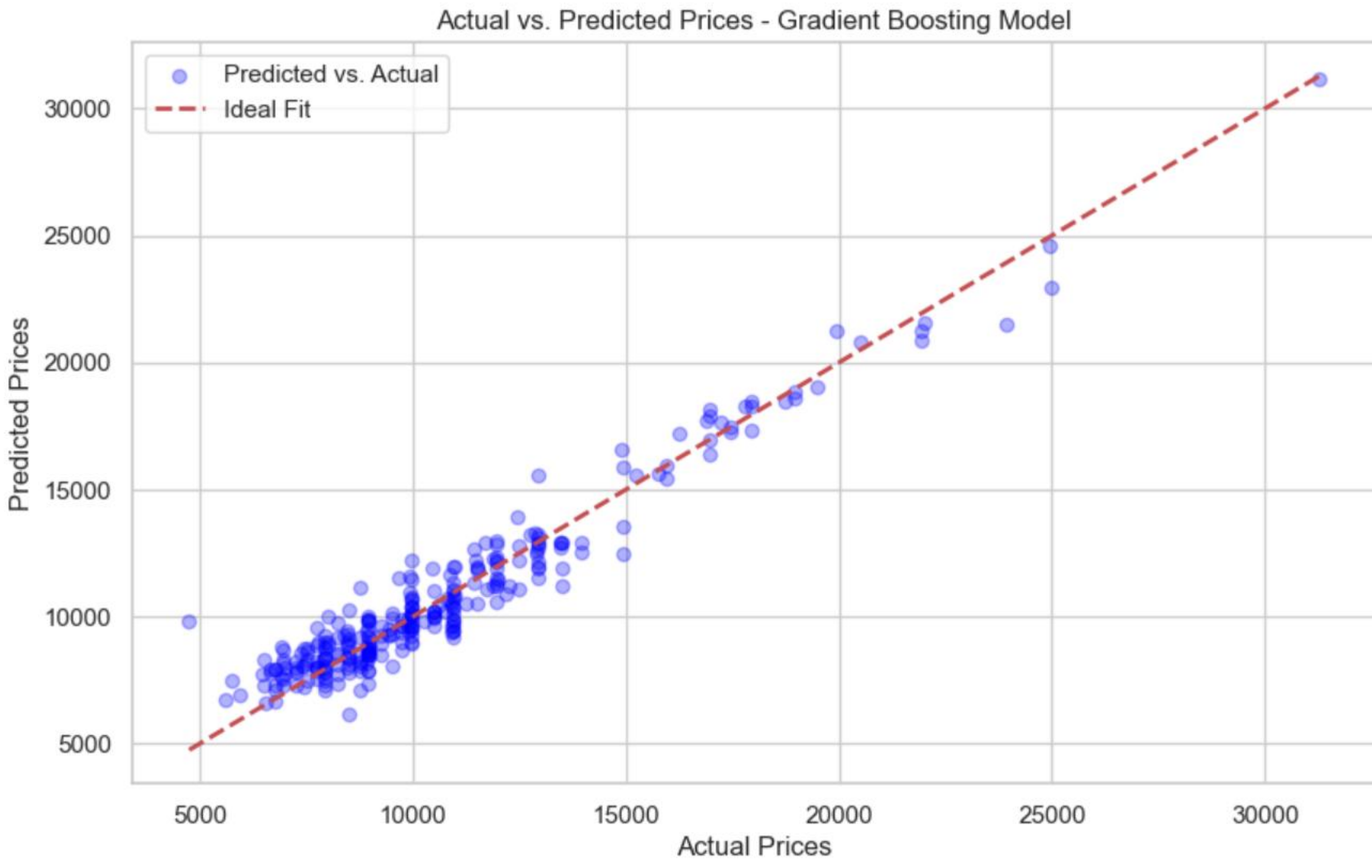
squared value. The higher R-squared value close to 1 indicates that the model explains a very high proportion of the variance in the dataset, which is desirable.

Therefore, The Gradient Boosting Regressor would be the best choice among the three models.

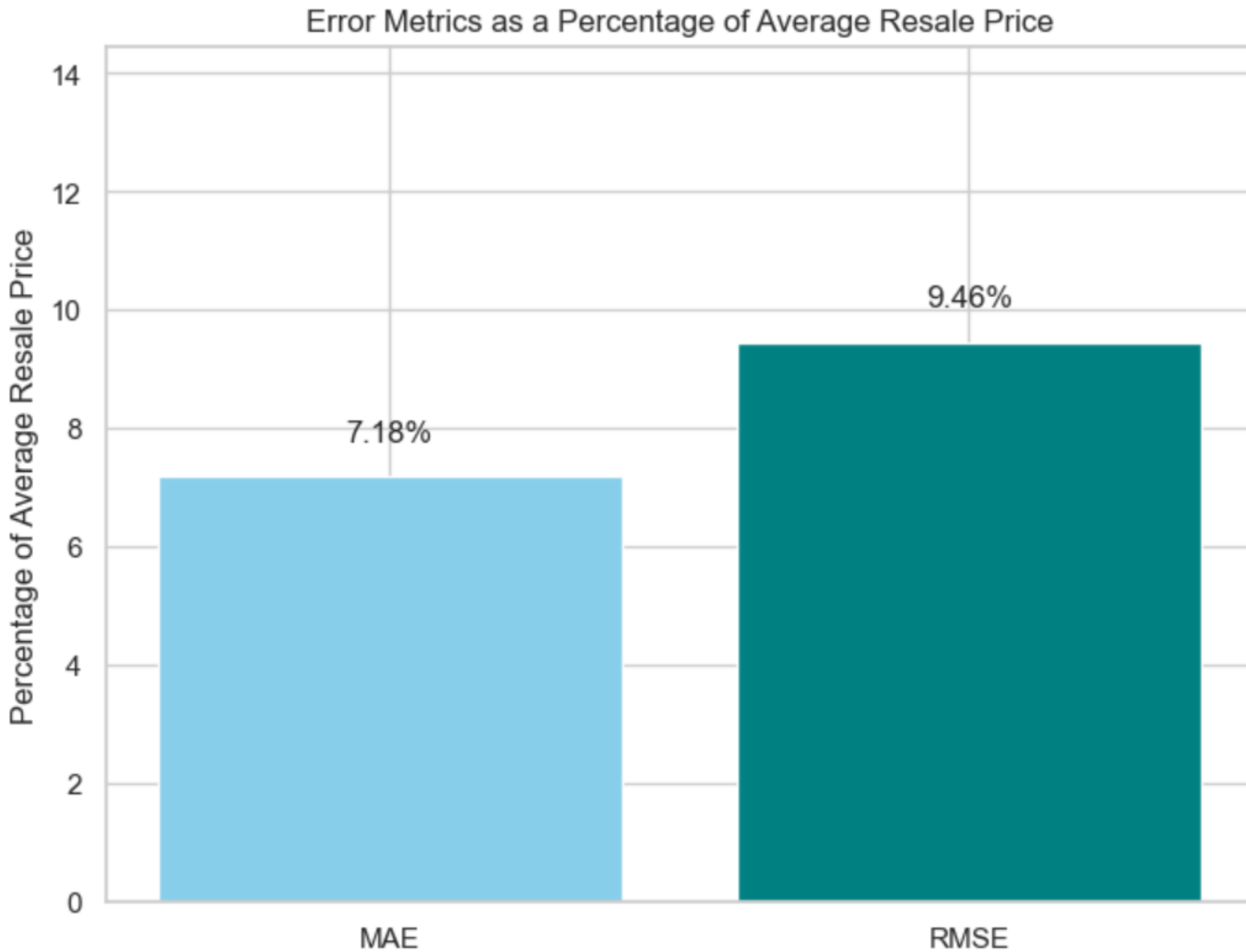
```
average_price = df_filtered['Price'].mean()
print(f"Average Reselling Price : {average_price}")

efficiency_percentage = r2_gb * 100
print(f"Prediction Efficiency in %: {efficiency_percentage}")
```

```
Average Reselling Price : 10047.586530931872
Prediction Efficiency in %: 93.44587335554083
```

Model Performance Evaluation:

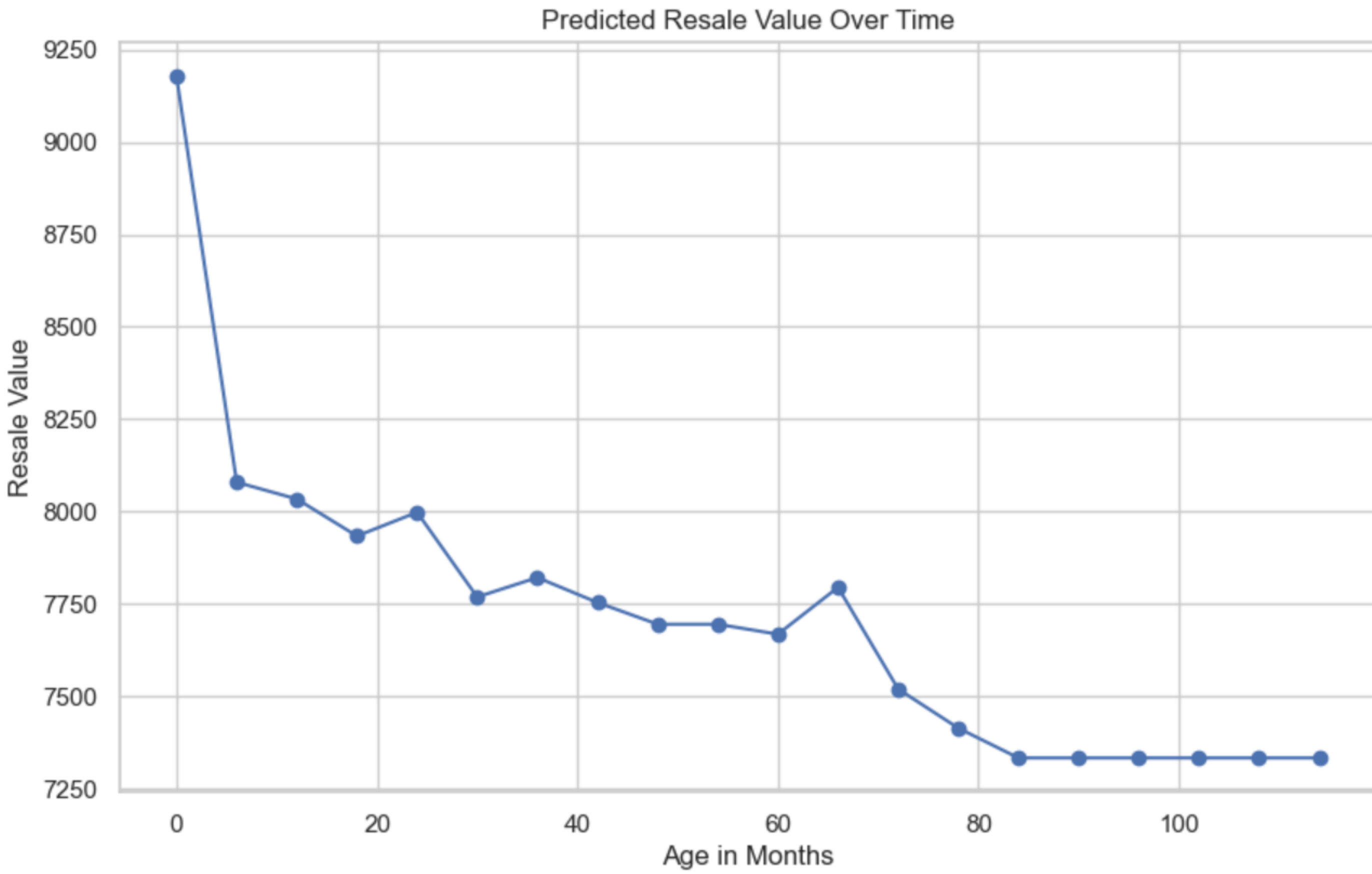

Actual vs. Predicted Prices - Gradient Boosting Model
Predicted vs. Actual
Ideal Fit
Predicted Prices
Actual Prices
5000
10000
15000
20000
25000
30000


Residuals vs. Predicted Prices - Gradient Boosting Model
Residuals
Zero Error
Residuals
Predicted Prices
2000
1000
0
−1000
−2000
−3000
−4000
−5000
5000
10000
15000
20000
25000
30000



In the business contexts, a prediction error margin of less than 10% is quite good, especially if prices vary significantly. In this case, in the used car market, where prices can vary widely based on the factors such as make, model, year, condition, and mileage, an average prediction error of about 7-10% is acceptable.

Resale Time-Series Analysis of a Sample Car Type Petrol:



Conclusion:

In this project, we applied various data mining techniques to analyze and extract meaningful insights from the given dataset. The process involved comprehensive data preprocessing, exploratory data analysis, and the application of machine learning models such as Decision Trees, Random Forest, and K-Nearest Neighbors. These models were evaluated based on accuracy, precision, recall, and F1-score to determine their effectiveness.

Our analysis revealed key patterns and trends within the data that can support informed decision-making. Among the models used, Gradient

Boosting Regressor performed the best, indicating its robustness in handling the classification task. Through feature importance analysis, we identified the most influential variables contributing to the predictions, which can guide future strategic actions.

Overall, this project demonstrates the power of data mining techniques in uncovering hidden patterns and optimizing predictive performance. It highlights the value of clean data, proper feature selection, and model tuning in building efficient analytical models. This approach can be extended to similar domains for better data-driven decisions.