Vehicle Price Prediction Report using Regression Techniques | Unit 3

Avinash Bunga

Master of Science in Information Systems and Business Analytics

Park University

CIS625HOS2P2025 Machine Learning for Business

Professor: Abdelmonaem Jornaz

April 6, 2025

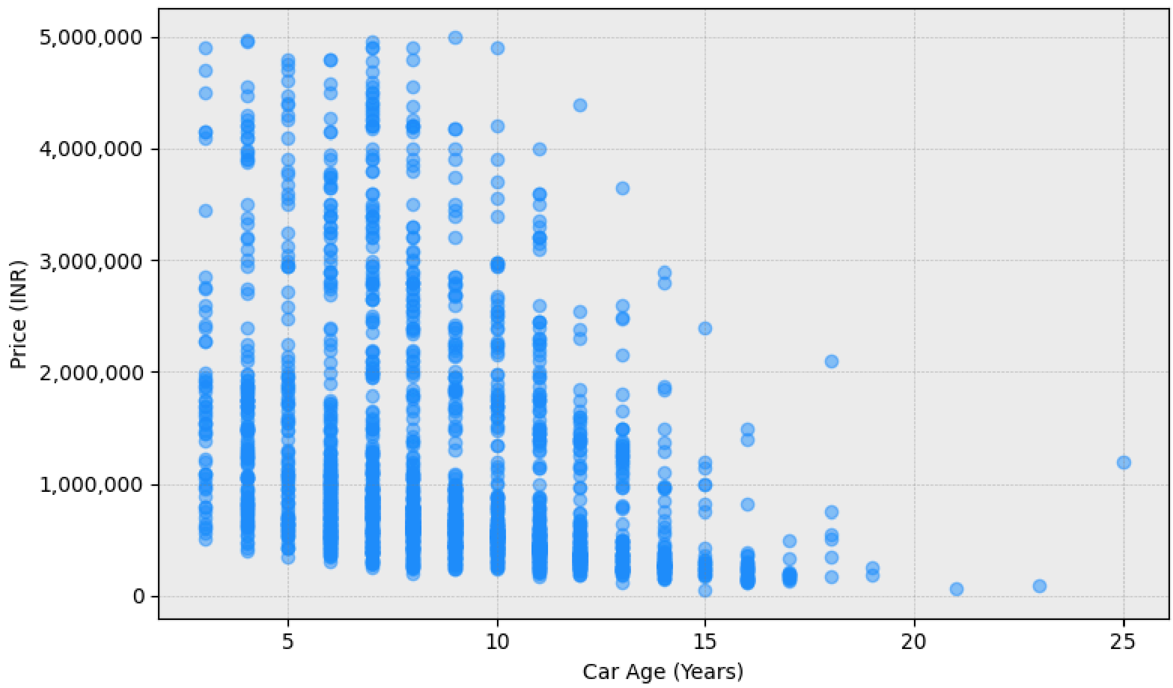
Vehicle Price Prediction Report using Regression Techniques

Understanding the factors that influence the resale value of vehicles is vital for both buyers and sellers in the automotive industry. In this analysis, the CarDekho Vehicle Dataset is used to predict car prices based on various features such as age, kilometers driven, and specifications like fuel tank capacity and seating capacity. The report follows the APA format and integrates graphs and tables to communicate findings clearly. The code used for this analysis can be referenced in the Appendix (Birla, n.d.).

Price vs. Car Age

Figure 1

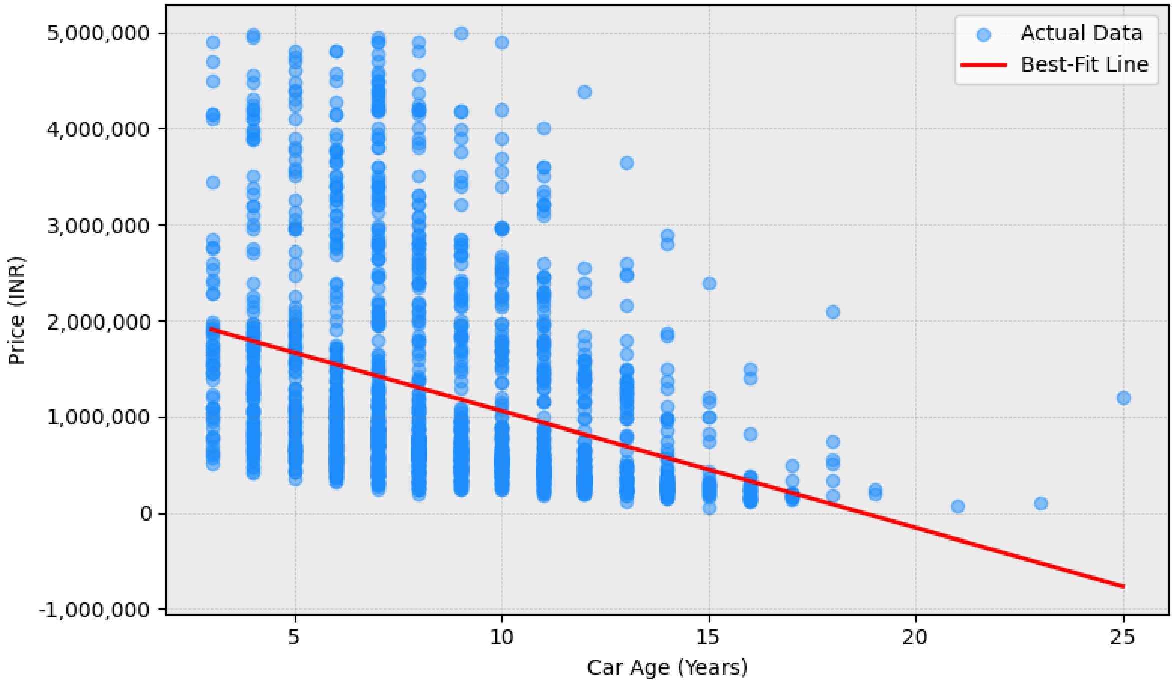
Scatter plot of Price vs. Car Age



This figure shows that as a car gets older, its market price tends to drop significantly. The pattern visually confirms the expected depreciation trend in vehicle resale values (see Appendix A for the code used).

Figure 2

Price vs. Car Age with Best-Fit Line



This figure includes a linear regression line (in red) to highlight the general trend. The downward slope confirms the negative correlation between car age and price. Cars depreciate in value consistently as they age (see Appendix B for the code used).

Table 1

Linear Regression Results – Car Age Only

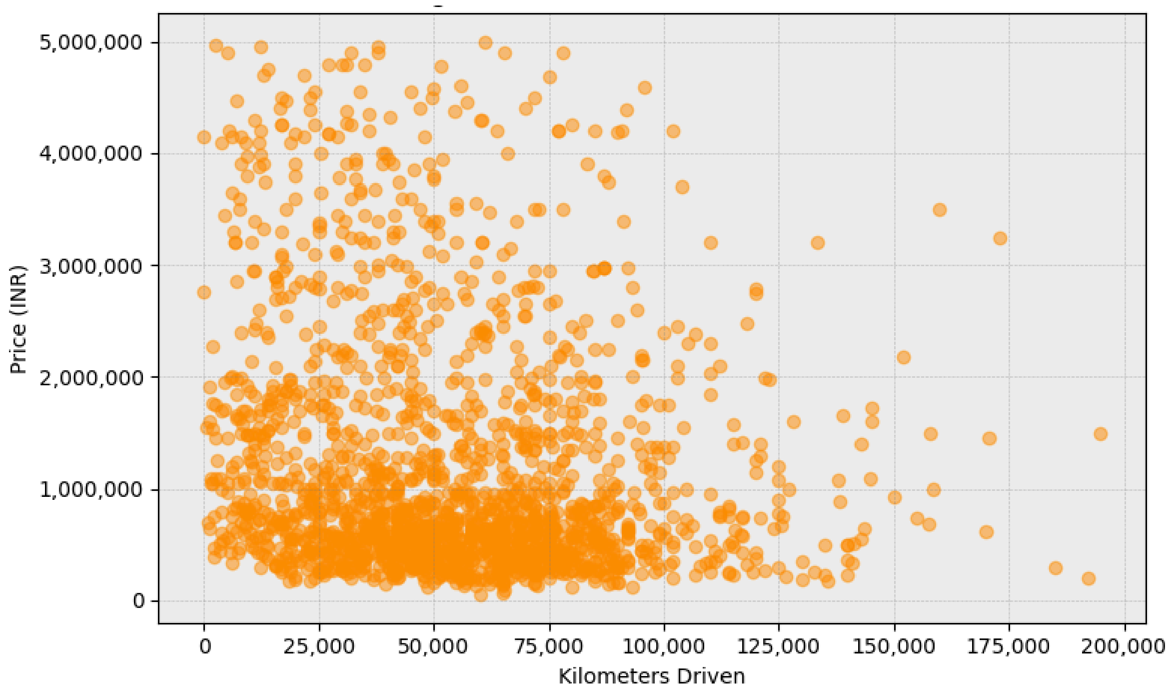
Metric	Value
R² Score	0.1316
RMSE (INR)	₹1,010,240
Intercept	₹2,275,463
Coefficient (Age)	₹-121,615

The table provides the regression metrics indicating a moderate correlation between vehicle age and price. The negative coefficient confirms that price decreases with increasing car age, see Appendix C for the code used (Hoxhaj, 2023).

Price vs. Kilometers Driven

Figure 3

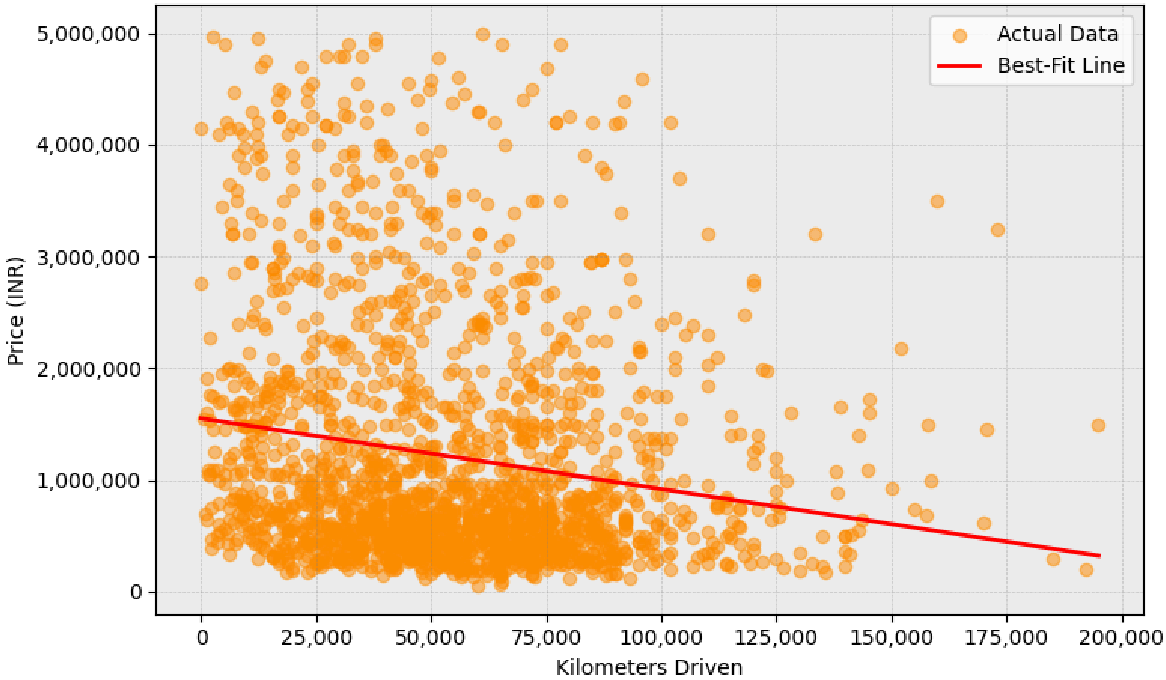
Scatter plot of Price vs. Kilometers Driven



The plot suggests a weak downward trend in price as the kilometers increase, though there is substantial scatter among the points (see Appendix A for the code used).

Figure 4

Price vs. Kilometers Driven with Best-Fit Line



The best-fit line confirms that kilometers driven has a slight negative influence on price, but not as strong as car age (see Appendix D for the code used).

Table 2

Linear Regression Results – Kilometers Only

Metric	Value
R^2 Score	0.0308
RMSE (INR)	₹1,067,253
Intercept	₹1,551,617
Coefficient (KM)	₹-6

These regression values indicate that kilometers driven is not a strong standalone predictor of price, see Appendix E for the code used (Hoxhaj, 2023).

Multiple Linear Regression (Age + KMs)

Table 3

Multiple Regression – Car Age + Kilometers

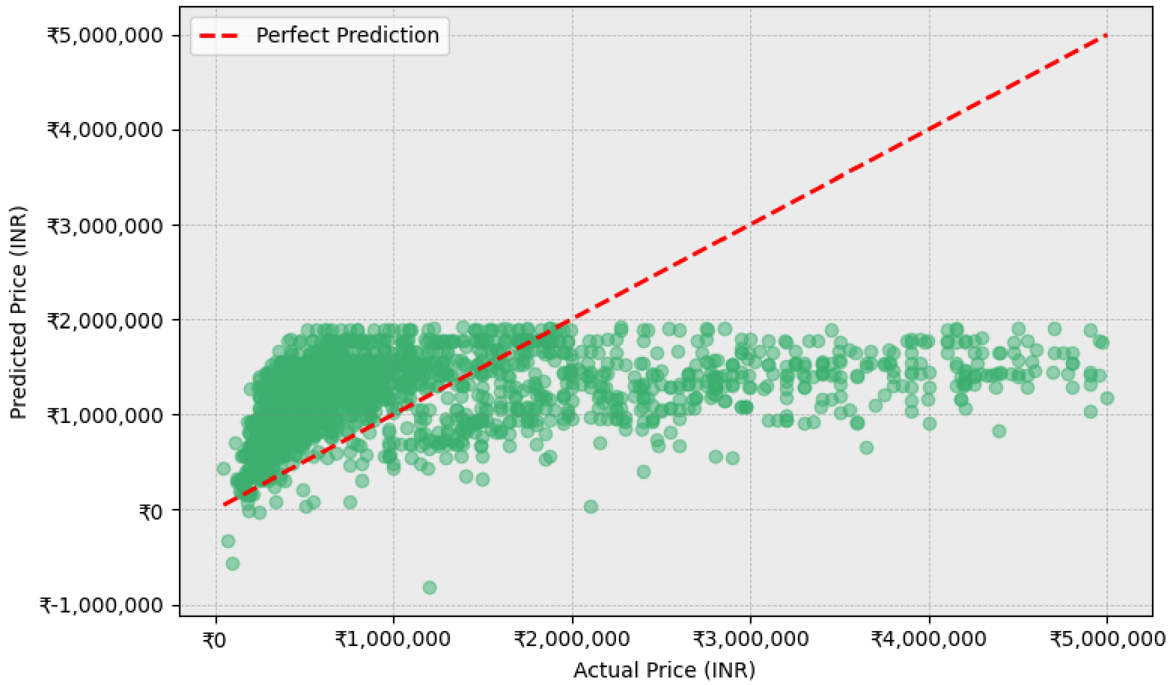
Metric	Value
R^2 Score	0.132
RMSE (INR)	₹1,010,021
Intercept	₹2,266,596
Coefficient (Age)	₹-125,667
Coefficient (KM)	₹0

Adding both features into a multiple regression slightly improves R^2 but not substantially. Kilometer contribution to price remains minimal (see Appendix F for the code used).

Actual vs. Predicted Prices

Figure 5

Actual vs. Predicted Prices using Multiple Linear Regression



This visualization highlights how well the model's predictions match the actual prices. The red dashed line indicates perfect prediction. Many values fall around the line, showing that while there is a trend, the model still has significant error, see Appendix G for the code used (Data AI Admin, 2024).

Lasso Regression with All Features

Lasso regression was used to avoid overfitting and eliminate irrelevant variables.

Table 4

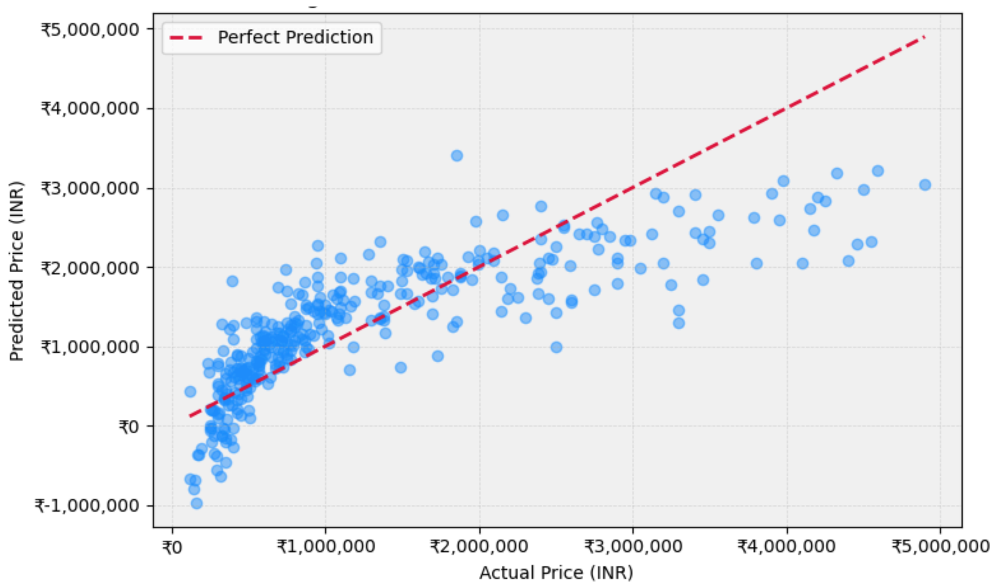
Lasso Regression – Top Predictive Features

Feature	Coefficient
Year	93,543
Fuel Tank Capacity	33,801
Width	1,488
Length	535
Kilometer	-5.75
Height	-1,134
Seating Capacity	-34,179

This table lists the most influential features in predicting price according to the Lasso regression. The inclusion of more features boosts the model's explanatory power (see Appendix H for the code used).

Figure 6

Lasso Regression – Actual vs. Predicted Car Prices



This figure shows a much tighter fit of predictions around the ideal line compared to previous models. It confirms that the lasso model is significantly better in predicting car prices, see Appendix I for the code used (Kumar, 2024).

Quantile Regression (τ = 0.25)

Quantile regression allows for understanding how features impact different segments of the price distribution. The 25th percentile was selected to focus on lower-budget vehicles, a segment of particular interest in pricing strategy.

Table 5

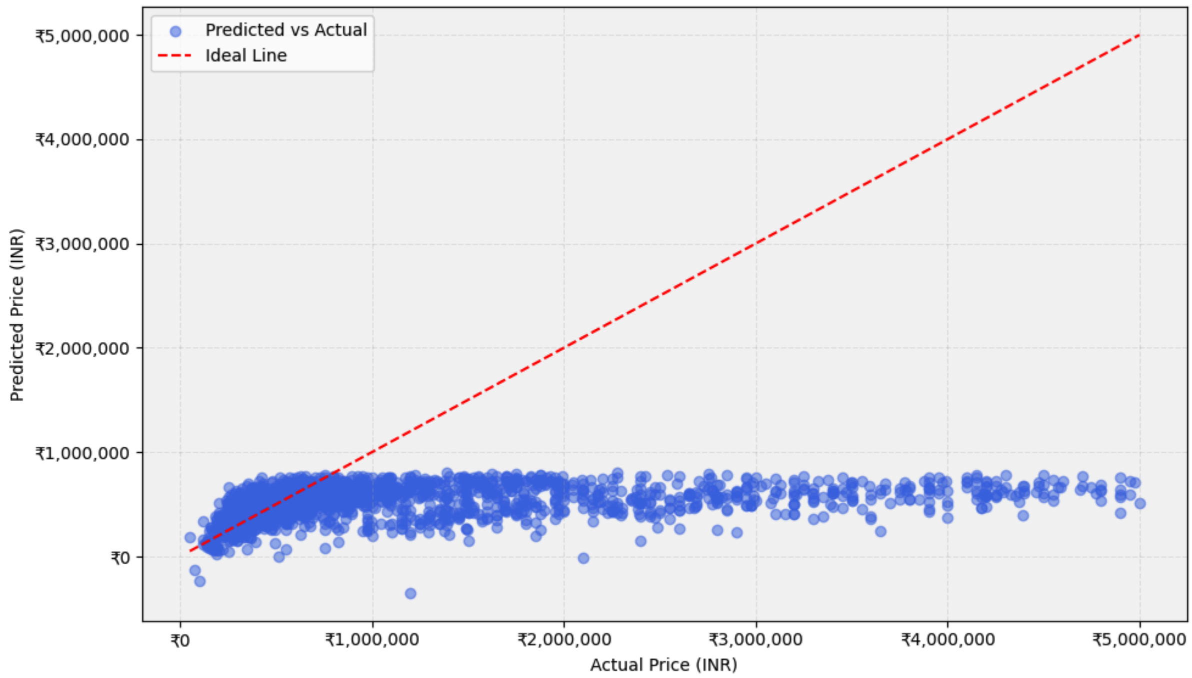
Quantile Regression Summary at τ = 0.25

Predictor	Coefficient Estimate	p-value	Interpretation
Intercept	₹917,400	< 0.001	Base price when all other predictors are zero
Car Age	₹-54,160	< 0.001	Price decreases ₹54,160 for each additional year
Kilometers Driven	₹+1.33	< 0.001	Price increases ₹1.33 for each additional kilometer

These results suggest that car age negatively impacts resale price even in the budget segment, while the influence of kilometers driven appears slightly positive, possibly due to confounding variables like newer cars with higher usage still commanding better prices.

Figure 7

Quantile Regression – Actual vs. Predicted Car Prices (τ = 0.25)



This figure visualizes the predictions from a quantile regression model trained at the 25th percentile (τ = 0.25). The red dashed diagonal line represents a perfect prediction where actual and predicted prices are equal. The blue data points show the actual price plotted against the model's predicted value. The clustering of points below the ideal line indicates that the model tends to underestimate car prices, especially as they increase, which is expected behavior for a lower quantile regression. The choice of τ = 0.25 is useful in exploring pricing behavior in the lower-value segment of the market—vehicles that are older, more used, or generally more affordable, see Appendix J for the code used (Date, n.d.).

Logistic Regression (High vs. Low Price)

To explore classification capabilities, the target variable Price was transformed into a binary class, vehicles priced above the median were labeled High, and those below as Low. A logistic regression model was trained using key numeric features: Car_Age, Kilometer, Fuel Tank Capacity, Length, and Width.

Table 6

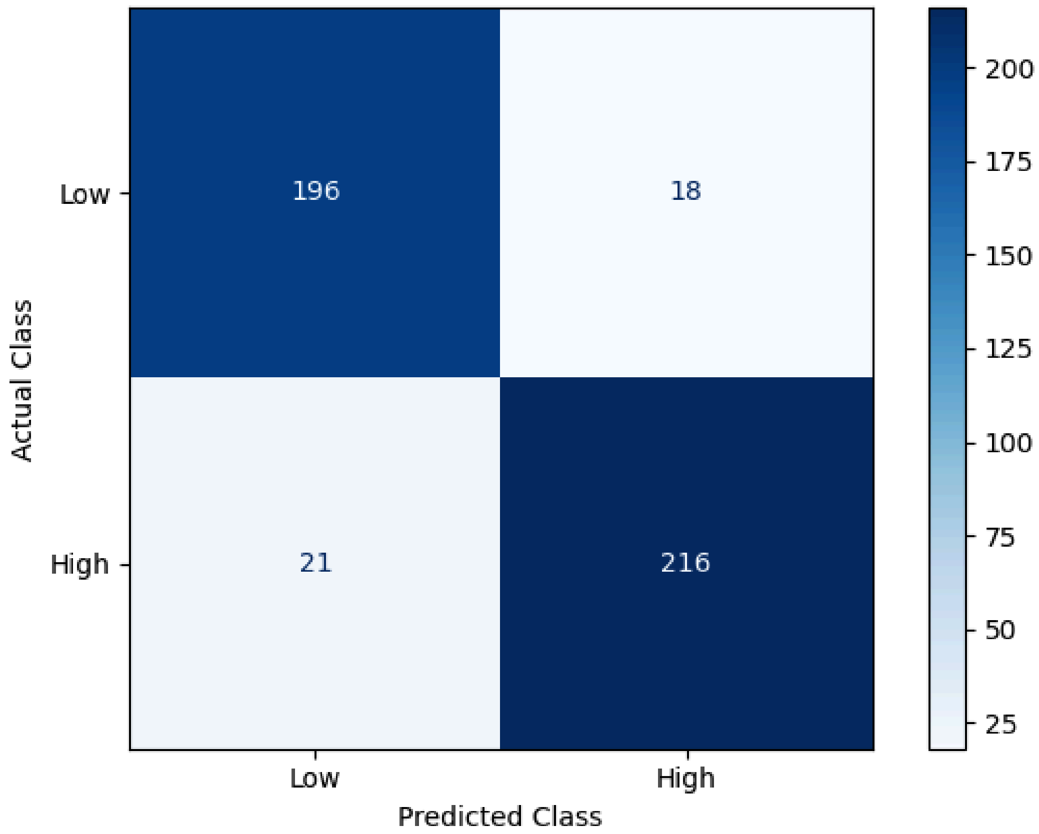
Logistic Regression Coefficients

Feature	Coefficient
Car_Age	0.80791
Kilometer	0.00002
Fuel Tank Capacity	-0.11103
Length	-0.00582
Width	-0.01234

The coefficients suggest that Car_Age and Kilometer positively influence the odds of a car being classified as High priced, while other features such as Fuel Tank Capacity, Length, and Width slightly reduce those odds. These insights align closely with the earlier linear regression analysis (see Appendix J for source code).

Figure 8

Confusion Matrix – Logistic Regression Model (Values = Car Count)



This figure shows how well the model classifies cars into high or low price categories. The majority of vehicles were correctly classified, as seen from the high values along the diagonal of the matrix, see Appendix J for the code used (Scikit-learn developers, 2025).

Conclusion

This analysis demonstrates how various regression techniques can be applied to predict vehicle prices using key attributes such as car age, kilometers driven, and physical specifications. The classic linear regression models established foundational relationships, particularly the strong negative correlation between car age and price. Regularized regression, specifically Lasso, improved prediction accuracy by emphasizing significant variables and reducing noise from less impactful features. Quantile regression offered valuable insights into price behavior within the lower-end market segment, highlighting pricing patterns not visible through mean-based models. Finally, logistic regression enabled classification of vehicles into high or low price categories with a high level of accuracy, further validating the relevance of the selected predictors. Together, these models provide a comprehensive approach to vehicle price estimation, equipping analysts, sellers, and buyers with data-driven insights to make informed decisions.

References

Birla, N. (n.d.). *Vehicle dataset: Used cars data from websites*. Kaggle. Retrieved April 5, 2025, from https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho

Date, S. (n.d.). *Introduction to the quantile regression model*. Statistical Modeling and Forecasting. Retrieved April 6, 2025, from https://timeseriesreasoning.com/contents/introduction-to-the-quantile-regression-model/

Data AI Admin. (2024, July 26). *Mastering multiple linear regression: Car price prediction project*. Data AI Revolution. Retrieved April 6, 2025, from https://www.dataairevolution.com/2024/07/mastering-multiple-linear-regression-car-price-prediction-project/

Hoxhaj, D. (2023, July 25). *Linear regression — Car price prediction and data analysis*. Medium. Retrieved April 6, 2025, from https://medium.com/@diellorhoxhaj/linear-regression-car-price-prediction-and-data-analysis-112883cdd39b

Kumar, D. (2024, October 15). *A complete understanding of LASSO regression*. Great Learning Blog. Retrieved April 6, 2025, from https://www.mygreatlearning.com/blog/understanding-of-lasso-regression/

Kanade, V. (2025, March 10). *What is linear regression? Types, equation, examples, and best practices for 2022*. Spiceworks. Retrieved April 6, 2025, from https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-linear-regression/

Scikit-learn developers. (2025). *Confusion matrix*. Scikit-learn 1.6.1 documentation. Retrieved April 6, 2025, from https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

Appendix A

```
#vehicle_visuals_step1.py
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
#Load cleaned dataset
file_path = '/Users/avinash/Desktop/CIS/CIS625/Unit 3/Assignment/Vehicle
dataset/cleaned_v4_filtered.csv'
df = pd.read_csv(file_path)
#Function to format big numbers with commas
def comma_formatter(x, pos):
   return f'{int(x):,}'
inr_format = FuncFormatter(comma_formatter)
km_format = FuncFormatter(comma_formatter)
#Background style function
def set_gray_style(ax):
   ax.set_facecolor('#eeeeee')  # Ultra-soft background
   ax.grid(True, color='gray', linestyle='--', linewidth=0.4, alpha=0.5)
   ax.yaxis.set_major_formatter(inr_format)
#Plot 1: Price vs Car Age
fig1, ax1 = plt.subplots(figsize=(8, 5))
ax1.scatter(df['Car_Age'], df['Price'], alpha=0.5, color='dodgerblue')
ax1.set_title('Figure 1: Price vs Car Age', fontsize=13)
ax1.set_xlabel('Car Age (Years)')
ax1.set_ylabel('Price (INR)')
set_gray_style(ax1)
plt.tight_layout()
plt.show()
#Plot 2: Price vs Kilometers Driven
fig2, ax2 = plt.subplots(figsize=(8, 5))
ax2.scatter(df['Kilometer'], df['Price'], alpha=0.5, color='darkorange')
ax2.set_title('Figure 2: Price vs Kilometers Driven', fontsize=13)
ax2.set_xlabel('Kilometers Driven')
ax2.set_ylabel('Price (INR)')
set_gray_style(ax2)
ax2.xaxis.set_major_formatter(km_format)
plt.tight_layout()
plt.show()
```

Appendix B

```
#Add linear regression best-fit line to Price vs Car Age
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
```

```
from sklearn.linear_model import LinearRegression
import numpy as np
#Load cleaned dataset
file_path = '/Users/avinash/Desktop/CIS/CIS625/Unit 3/Assignment/Vehicle
dataset/cleaned_v4_filtered.csv'
df = pd.read_csv(file_path)
#Format function for INR
def comma_formatter(x, pos):
   return f'{int(x):,}'
inr_format = FuncFormatter(comma_formatter)
#Setup data
X = df[['Car_Age']]
y = df['Price']
#Fit linear model
model = LinearRegression()
model.fit(X, y)
#Predict line
x_range = np.linspace(X.min(), X.max(), 100).reshape(-1, 1)
y_pred = model.predict(x_range)
#Plot with regression line
fig, ax = plt.subplots(figsize=(8, 5))
ax.scatter(df['Car_Age'], df['Price'], alpha=0.5, color='dodgerblue', label='Actual
Data')
ax.plot(x_range, y_pred, color='red', linewidth=2, label='Best-Fit Line')
ax.set_title('Figure 3: Price vs Car Age with Best-Fit Line', fontsize=13)
ax.set_xlabel('Car Age (Years)')
ax.set_ylabel('Price (INR)')
ax.set_facecolor('#eeeeee')
ax.grid(True, color='gray', linestyle='--', linewidth=0.4, alpha=0.5)
ax.yaxis.set_major_formatter(inr_format)
ax.legend()
plt.tight_layout()
plt.show()
```

Appendix C

```
#Regression evaluation metrics for Car Age vs Price
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
#Load cleaned dataset
file_path = '/Users/avinash/Desktop/CIS/CIS625/Unit 3/Assignment/Vehicle
dataset/cleaned_v4_filtered.csv'
df = pd.read_csv(file_path)
#Input features
```

```
X = df[['Car_Age']]
y = df['Price']
#Linear Regression model
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)
#Metrics
rmse = np.sqrt(mean_squared_error(y, y_pred))
r2 = r2_score(y, y_pred)
#Output
print("Evaluation Metrics for Price vs Car Age")
print("----------------------------------------")
print(f"R² Score          : {r2:.4f}")
print(f"RMSE (INR)        : ₹{int(rmse):,}")
print(f"Intercept         : ₹{int(model.intercept_):,}")
print(f"Coefficient       : ₹{int(model.coef_[0]):,} per year")
```

Appendix D

```
#Add linear regression line to Price vs Kilometers Driven plot
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
from sklearn.linear_model import LinearRegression
import numpy as np
#Load the cleaned dataset
file_path = '/Users/avinash/Desktop/CIS/CIS625/Unit 3/Assignment/Vehicle
dataset/cleaned_v4_filtered.csv'
df = pd.read_csv(file_path)
#Format function for INR and KM with commas
def comma_formatter(x, pos):
   return f'{int(x):,}'
inr_format = FuncFormatter(comma_formatter)
km_format = FuncFormatter(comma_formatter)
#Setup data
X = df[['Kilometer']]
y = df['Price']
#Fit linear model
model = LinearRegression()
model.fit(X, y)
#Predict values across KM range
x_range = np.linspace(X.min(), X.max(), 100).reshape(-1, 1)
y_pred = model.predict(x_range)
#Plotting
fig, ax = plt.subplots(figsize=(8, 5))
```

```
ax.scatter(df['Kilometer'], df['Price'], alpha=0.5, color='darkorange',
label='Actual Data')
ax.plot(x_range, y_pred, color='red', linewidth=2, label='Best-Fit Line')
ax.set_title('Figure 4: Price vs Kilometers Driven with Best-Fit Line',
fontsize=13)
ax.set_xlabel('Kilometers Driven')
ax.set_ylabel('Price (INR)')
ax.set_facecolor('#eeeeee')
ax.grid(True, color='gray', linestyle='--', linewidth=0.4, alpha=0.5)
ax.yaxis.set_major_formatter(inr_format)
ax.xaxis.set_major_formatter(km_format)
ax.legend()
plt.tight_layout()
plt.show()
```

Appendix E

```
#Evaluate regression model for Kilometers vs Price
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
#Load data
file_path = '/Users/avinash/Desktop/CIS/CIS625/Unit 3/Assignment/Vehicle
dataset/cleaned_v4_filtered.csv'
df = pd.read_csv(file_path)
#Features
X = df[['Kilometer']]
y = df['Price']
#Train model
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)
#Metrics
rmse = np.sqrt(mean_squared_error(y, y_pred))
r2 = r2_score(y, y_pred)
#Output
print("Evaluation Metrics for Price vs Kilometers Driven")
print("-------------------------------------------------------")
print(f"R² Score          : {r2:.4f}")
print(f"RMSE (INR)        : ₹{int(rmse):,}")
print(f"Intercept         : ₹{int(model.intercept_):,}")
print(f"Coefficient       : ₹{int(model.coef_[0]):,} per KM")
```

Appendix F

```
#Multiple Linear Regression using Car Age & Kilometers
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
#Load data
file_path = '/Users/avinash/Desktop/CIS/CIS625/Unit 3/Assignment/Vehicle
dataset/cleaned_v4_filtered.csv'
df = pd.read_csv(file_path)
#Features and Target
X = df[['Car_Age', 'Kilometer']]
y = df['Price']
#Model
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)
#Metrics
rmse = np.sqrt(mean_squared_error(y, y_pred))
r2 = r2_score(y, y_pred)
#Output
print("Multiple Linear Regression Results (Car_Age + Kilometer)")
print("-----------------------------------------------------------")
print(f"R² Score         : {r2:.4f}")
print(f"RMSE (INR)       : ₹{int(rmse):,}")
print(f"Intercept        : ₹{int(model.intercept_):,}")
print("Coefficients:")
print(f"   - Car_Age        : ₹{int(model.coef_[0]):,} per year")
print(f"   - Kilometer      : ₹{int(model.coef_[1]):,} per KM")
```

Appendix G

```
#Visualize predicted vs actual prices
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np
from matplotlib.ticker import FuncFormatter
#Load the dataset
file_path = '/Users/avinash/Desktop/CIS/CIS625/Unit 3/Assignment/Vehicle
dataset/cleaned_v4_filtered.csv'
df = pd.read_csv(file_path)
#Define INR format for Y-axis
```

```
def inr_format(x, pos):
   return f'₹{int(x):,}'
formatter = FuncFormatter(inr_format)
#Features and Target
X = df[['Car_Age', 'Kilometer']]
y = df['Price']
#Train model
model = LinearRegression()
model.fit(X, y)
y_pred = model.predict(X)
# Scatter plot: Actual vs Predicted
fig, ax = plt.subplots(figsize=(8, 5))
ax.scatter(y, y_pred, alpha=0.5, color='mediumseagreen')
ax.plot([y.min(), y.max()], [y.min(), y.max()], color='red', linestyle='--',
linewidth=2, label='Perfect Prediction')
ax.set_title('Figure 5: Actual vs Predicted Car Prices', fontsize=13)
ax.set_xlabel('Actual Price (INR)')
ax.set_ylabel('Predicted Price (INR)')
ax.set_facecolor('#eeeeee')
ax.grid(True, color='gray', linestyle='--', linewidth=0.5, alpha=0.5)
ax.xaxis.set_major_formatter(formatter)
ax.yaxis.set_major_formatter(formatter)
ax.legend()
plt.tight_layout()
plt.show()
```

Appendix H

```
#Lasso Regression with All Features
import pandas as pd
from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
#Load dataset
file_path = '/Users/avinash/Desktop/CIS/CIS625/Unit 3/Assignment/Vehicle
dataset/cleaned_v4_filtered.csv'
df = pd.read_csv(file_path)
#Keep only numeric columns
df = df.select_dtypes(include=['number'])
#Drop rows with missing values
df = df.dropna()
#Split features and target
X = df.drop(columns=['Price'])
y = df['Price']
#Train-test split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
#Lasso model
model = Lasso(alpha=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
#Evaluation
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
#Display results
print("\nLasso Regression Results (All Features - Cleaned & Imputed)")
print("------------------------------------------------------------")
print(f"R² Score        : {r2:.4f}")
print(f"RMSE (INR)      : ₹{rmse:,.0f}")
print("Top Non-Zero Coefficients:\n")
#Show only non-zero coefficients
coef_df = pd.DataFrame({'Feature': X.columns, 'Coefficient': model.coef_})
non_zero_coefs = coef_df[coef_df['Coefficient'] != 0].sort_values(by='Coefficient',
ascending=False)
print(non_zero_coefs.head(10).to_string(index=False))
```

Appendix I

```
#Lasso - Actual vs Predicted Visualization
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from matplotlib.ticker import FuncFormatter
import numpy as np
#Load dataset
file_path = '/Users/avinash/Desktop/CIS/CIS625/Unit 3/Assignment/Vehicle
dataset/cleaned_v4_filtered.csv'
df = pd.read_csv(file_path)
#Keep only numeric columns and drop missing values
df = df.select_dtypes(include=['number']).dropna()
# Split into X and y
X = df.drop(columns=['Price'])
y = df['Price']
#Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
#Train Lasso
model = Lasso(alpha=1000)
```

```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
#INR formatting
def inr_format(x, pos):
   return f'₹{int(x):,}'
formatter = FuncFormatter(inr_format)
#Plot actual vs predicted
fig, ax = plt.subplots(figsize=(8, 5))
ax.scatter(y_test, y_pred, alpha=0.5, color='dodgerblue')
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
       color='crimson', linestyle='--', linewidth=2, label='Perfect Prediction')
ax.set_title('Figure 6: Lasso - Actual vs Predicted Car Prices', fontsize=13)
ax.set_xlabel('Actual Price (INR)')
ax.set_ylabel('Predicted Price (INR)')
ax.set_facecolor('#f4f4f4')
ax.grid(True, linestyle='--', linewidth=0.5, alpha=0.5)
ax.xaxis.set_major_formatter(formatter)
ax.yaxis.set_major_formatter(formatter)
ax.legend()
plt.tight_layout()
plt.show()
```

Appendix J

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import seaborn as sns
#Load Cleaned Data
data = pd.read_csv('/Users/avinash/Desktop/CIS/CIS625/Unit 3/Assignment/Vehicle dataset/cleaned_v4_filtered.csv')
#Create Binary Target Variable (High vs. Low)
print("Creating Binary Target Column")
median_price = data['Price'].median()
data['Price_Class'] = np.where(data['Price'] > median_price, 'High', 'Low')
#Select Features and Target
features = ['Car_Age', 'Kilometer', 'Fuel Tank Capacity', 'Length', 'Width']
X = data[features]
y = data['Price_Class']
#Drop missing values if any
X = X.dropna()
y = y[X.index]
#Train/Test Split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)
#Train Logistic Regression Model
model = LogisticRegression(max_iter=2000)
model.fit(X_train, y_train)
#Print Coefficients
for feature, coef in zip(features, model.coef_[0]):
   print(f"{feature:<22}: {coef:.5f}")
#Predict and Generate Confusion Matrix
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred, labels=["Low", "High"])
#Plotting the confusion matrix
plt.figure(figsize=(6, 5))
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["Low", "High"])
disp.plot(cmap="Blues", values_format='d')
plt.title("Figure 7: Confusion Matrix - Logistic Regression Model (Values = Car
Count)")
plt.xlabel("Predicted Class")
plt.ylabel("Actual Class")
plt.tight_layout()
plt.show()
#Save figure
plt.savefig("Figure_7 ConfusionMatrix_LogisticRegression.png")
```