JAPAN USED CAR PRICE PREDICTION

META SCIFOR TECHNOLOGIES

2024

Script. Sculpt. Socialize

Guided By:

Saurav Kumar

Submitted By:

Bhushan Khushal Nimje

MST02-0020

TABLE OF CONTENT

ABSTRACT

This project presents the development of an innovative Japan Used Car Price Prediction System. The primary objective is to explore the relationships between the features and provide insight into used car pricing as well as the impact of factors such as vehicle age, mileage, engine specifications and fuel type on pricing.

To achieve this, we utilized three machine learning models: Linear Regression, Decision Tree Regressor and Random Forest Regressor. These models were trained on a comprehensive dataset containing ten key features of each vehicle such as: Price, Mark (Brand), Model, Year of Registration, Mileage, Engine Capacity, Transmission Type, Wheel Drive Type (2wd, 4wd and awd), Hand Drive Configuration (left-hand or right-hand) and Fuel Type. The models were evaluated using metrics.

This Japan Used Car Price Prediction System exemplifies how machine learning models can be leveraged to enhance decision-making in the used car market. This project also highlights the potential for further improvements, such as incorporating personalized pricing recommendations and exploring more advanced machine learning models to improve prediction accuracy and market analysis.

This abstract encapsulates the project's scope, methodology and outcomes providing a comprehensive overview of the Japan Used Car Price Prediction System.

INTRODUCTION

In recent years, the global automobile market has seen a significant rise in the popularity of used car transaction, driven by affordability and a range of available options. Japan being one of the largest producers and exporters of cars, plays a crucial role in this trend.

Understanding the factors that influence the price of a used vehicle is critical for both buyers and sellers. Price determinants range from the car's brand and model to its condition, mileage, age, and additional features. Data-driven insights into these factors can empower stakeholders to make more informed decisions, enhancing the overall transparency and efficiency of the marketplace.

In this study, we aim to develop machine learning models to predict the price of used cars using a dataset. The dataset includes ten key features, such as the car's price, brand (Mark), model, registration year (Years), mileage, engine capacity, transmission type, drive type (2WD, 4WD, AWD), hand drive orientation (left or right), and fuel type. By analysing these features, the goal is to establish predictive models that can accurately estimate a car's price based on its characteristics.

We employ three widely used machine learning algorithms for this task: linear regression, decision tree regressor, and random forest regressor. Each of these algorithms offers unique strengths. Linear regression provides a straightforward approach to understanding the relationship between features and price, while decision trees offer more flexibility by capturing non-linear relationships. Random forest, an ensemble method, builds multiple decision trees and aggregates their predictions to enhance accuracy. By comparing the performance of these models, we aim to identify the most suitable algorithm for predicting car prices in this dynamic market.

TECHNOLOGIS USED

The technology used in this project consists a variety of tools essential for building a machine learning model. Below is a detailed overview of the technologies used:

Programming Language

Python: The primary programming language used for implementing the machine learning models and data manipulation. Python's simplicity and extensive library support make it ideal for machine learning projects.

Machine Learning Model

1. **Machine Learning Frameworks & Libraries:**
    a. **Scikit-learn:** A machine learning library used for implementing the Linear Regression, Decision Tree Regressor and Random Forest Regressor. It also offers various tools for model building, data preprocessing and evaluation.
    b. **Numpy:** A library for numerical computations used extensively for handling arrays, mathematical operations and data manipulation.
    c. **Pandas:** A data manipulation library used for loading, preprocessing and manipulating the dataset.
    d. **Matplotlib:** A data visualization library used for plotting graphs, visualizing the data distribution and comparing model performance.
2. **Data Handling and Manipulation:**
    a. **Loading and preprocessing the dataset:** Utilizing pandas for reading and cleaning the dataset, and Scikit-learn for scaling and splitting the data.
3. **Model Building:**
    a. **Linear Regression:** A regression algorithm implemented using Scikit-learn to model the relationship between the features and the target variable.
    b. **Decision Tree Regressor:** A tree algorithm which offers more flexibility by capturing non-linear relationships.
    c. **Random Forest Regressor:** An ensemble learning method implemented using Scikit-learn to build a robust predictive model by combining multiple decision trees.
    d. **Model Training:** Training the Linear Regression, Decision Tree Regressor and Random Forest Regressor models using the .fit method.
    e. **Hyperparameter Tuning:** Finding the best parameters for Linear Regression with LassoCV. Using GridSearchCV for Decision Tree Regressor and Random Forest Regressor.
4. **Model Evaluation:**
    a. **Evaluation Metrics:** Evaluating the model with metrics like R2 Score, Mean Absolute Error (MAE) and Mean Squared Error (MSE).
    b. **Visualization:** Plotting the models prediction against the actual values using Matplotlib to visualize the accuracy of the models.

DATASET INFORMATION

The dataset used in this project plays a crucial role in building and validating the model. Below is the detailed overview of the dataset:

Dataset Size:

The dataset contains 2318 rows and 11 columns which represents various features of cars such as:

- **Price:** Tells the retail value of the car.
- **Mark (Brand):** Tells that the car belongs to which particular brand like Nissan, Toyota and more.
- **Model:** Tells the model's name of car like March, Lafista, Mira and more.
- **Year:** Tells the manufacturing year of the car.
- **Mileage:** Tells the mileage of the car.
- **Engine Capacity:** Give us the information about the power of the engine.
- **Transmission:** Tell us about the transmission type weather it is Automatic, Manual or CVT.
- **Drive:** Tells weather the car is 2wd, 4wd or awd.
- **Hand Drive:** Tells weather the car is Right Hand Drive, Left Hand Drive or Center Hand Drive.
- **Fuel:** Tells the type of fuel required to run the car like Petrol, Diesel, Hybrid, LPG, CNG.

```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2318 entries, 0 to 2317
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   id               2318 non-null   int64
 1   price            2318 non-null   int64
 2   mark             2318 non-null   object
 3   model            2318 non-null   object
 4   year             2318 non-null   int64
 5   mileage          2318 non-null   int64
 6   engine_capacity  2318 non-null   int64
 7   transmission     2318 non-null   object
 8   drive            2318 non-null   object
 9   hand_drive       2318 non-null   object
 10  fuel             2318 non-null   object
dtypes: int64(5), object(6)
memory usage: 199.3+ KB
None
```

METHODOLOGY

The methodology section outlines the systematic approach followed in developing the Japan Used Car Price Prediction System using the machine learning models. The key stage of this process includes Data Preprocessing, Model Selection, Training & Evaluation.

1. **Data Preprocessing:** Data Preprocessing is a crucial step in preparing the raw data for machine learning models. It involves cleaning, transforming and organizing the data to make it suitable for analysis and model building.
   - **Handling Missing Values:**
     - Checking the missing values with the help of .isnull().sum() to see if there are any null values present in the dataset. In this dataset there were no null values present.
     - If there were any missing values they were either be filled with mean or median values in numerical case but in categorical case it will be filled with mode() or removed.
   - **Outlier Detection:**
     - Anomalies in data, such as extremely high or low values that could skew the model, were identified and treated appropriately.
     - The methods used to treat the outliers are ZScore method and IQR (Inter Quantile Range) method.
   - **Checking Skewness:**
     - Checking the skewness with help of .skew() method and treating the skewness with Power Transformer.
   - **Feature Scaling:**
     - Standardizing the dataset with the help of Standard Scaler. There are two methods to standardize the data 1. Standard Scaler 2. Min Max Scaler.
   - **Checking Multicollinearity:**
     - Checking the relationship between the features to see if they are related to each other with the variance inflation factor.
2. **Model Selection:** Model selection involves choosing the right machine learning algorithm that would predict the target variable.
   - **Linear Regression:** It is a straight forward model that assumes a linear relationship between the independent variables (features like mileage, engine capacity, year, etc.) and the dependent variable (price). This simplicity makes it easy to interpret the coefficients of the model, helping you understand how much feature affects the price.
   - **Decision Tree Regressor:** Decision Tree Regressor can model non-linear relationship between the features and the target. It can naturally handle both numerical and categorical features without the need for extensive preprocessing like encoding.
   - **Random Forest Regressor:** Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions to produce a more accurate and stable result. Thiis ensemble approach reduces the risk of overfitting that single decision tree might suffer from, leading to better generalization on unseen data.

3. **Model Training & Evaluation:** After selecting the model the next step is training them on the dataset and evaluating their performance.
   - **Training:**
     - The dataset was split int training and testing sets (70:30). The models were trained on the training data, where the algorithms learned the patterns in the input features and their relationship to the target variable.
   - **Evaluation Metrics:**
     - The models were evaluated using the metrics such as R-squared method, Mean Absolute Error (MAE), Mean Squared Error (MSE) to measure the accuracy of the model.
4. **Hyperparameter Tuning:** To optimize the model performance, hyperparameter tuning was performed.
   - **LassoCV:** It simplifies the model by selecting key features, reduce overfitting, handles multicollinearity and ensures optimized performance via cross-validation.
   - **RidgeCV:** It minimizes overfitting by penalizing large coefficients, handles multicollinearity and optimizes regularization strength through cross-validation.
   - **Grid Search CV:** It systematically explores a range of hyperparameters, identifying the best combination for model performance. It ensures optimal tuning, enhancing accuracy and robustness in predicting used car price.

CODE SNIPPET

Installing Dependencies

```
[1]: import pandas as pd
     import numpy as np

     import seaborn as sns
     import matplotlib.pyplot as plt
     %matplotlib inline

     import warnings
     warnings.filterwarnings('ignore')
```

Data Loading

```
[2]: #Lets load the dataset
     data = pd.read_csv("Japan_used_cars_datasets.csv")
```

```
[3]: # Check the shape of the dataset to understand its dimensions
     print(data.shape)
```

```
(2318, 11)
```

```
[4]: data.head()
```

[4]:

	id	price	mark	model	year	mileage	engine_capacity	transmission	drive	hand_drive	fuel
0	0	80	nissan	march	2003	80000	1240	at	2wd	rhd	gasoline
1	1	110	nissan	march	2010	53000	1200	at	2wd	rhd	gasoline
2	2	165	nissan	lafesta	2005	47690	2000	at	2wd	rhd	gasoline
3	3	190	toyota	avensis	2008	130661	1990	at	2wd	rhd	gasoline
4	4	190	daihatsu	mira	2006	66300	660	at	2wd	rhd	gasoline

Data Preprocessing

```
# Droping unwanted columns
data = data.drop(['id'], axis=1)
```

```
data
```

	price	mark	model	year	mileage	engine_capacity	transmission	drive	hand_drive	fuel
0	80	nissan	march	2003	80000	1240	at	2wd	rhd	gasoline
1	110	nissan	march	2010	53000	1200	at	2wd	rhd	gasoline
2	165	nissan	lafesta	2005	47690	2000	at	2wd	rhd	gasoline
3	190	toyota	avensis	2008	130661	1990	at	2wd	rhd	gasoline
4	190	daihatsu	mira	2006	66300	660	at	2wd	rhd	gasoline
...	...	...	...	...	...	...	...	...	...	...
2313	1400	toyota	vitz	2009	121000	996	at	2wd	rhd	gasoline
2314	1400	toyota	estima	2003	101000	3000	at	2wd	rhd	gasoline
2315	1400	subaru	r2	2005	101000	660	cvt	2wd	rhd	gasoline
2316	1400	honda	z	2000	170000	660	at	4wd	rhd	gasoline
2317	1400	toyota	estima t	2005	72320	3000	at	2wd	rhd	gasoline

2318 rows × 10 columns

```
#Lets Check for Outliers
#Select all numerical columns for ploting Distplot and Box plot
numeric = ['int64']

newdf = data.select_dtypes(include=numeric)
newdf
```

	price	year	mileage	engine_capacity
0	80	2003	80000	1240
1	110	2010	53000	1200
2	165	2005	47690	2000
3	190	2008	130661	1990
4	190	2006	66300	660
...	...	...	...	...
2313	1400	2009	121000	996
2314	1400	2003	101000	3000
2315	1400	2005	101000	660
2316	1400	2000	170000	660
2317	1400	2005	72320	3000

2318 rows × 4 columns

```
# Encode categorical columns using Label Encoding
from sklearn.preprocessing import LabelEncoder
label_encoders = {}
for column in df_zscore.select_dtypes(include=['object']).columns:
    label_encoders[column] = LabelEncoder()
    df_zscore[column] = label_encoders[column].fit_transform(df_zscore[column])
```

```
df_zscore
```

	price	mark	model	year	mileage	engine_capacity	transmission	drive	hand_drive	fuel
1	110	17	133	2010	53000	1200	0	0	1	2
2	165	17	124	2005	47690	2000	0	0	1	2
3	190	23	28	2008	130661	1990	0	0	1	2
4	190	4	143	2006	66300	660	0	0	1	2
5	190	4	143	2004	81400	660	0	0	1	2
...	...	...	...	...	...	...	...	...	...	...
2313	1400	23	230	2009	121000	996	0	0	1	2
2314	1400	23	91	2003	101000	3000	0	0	1	2
2315	1400	21	183	2005	101000	660	1	0	1	2
2316	1400	7	243	2000	170000	660	0	1	1	2
2317	1400	23	92	2005	72320	3000	0	0	1	2

2262 rows × 10 columns

```
# Check for skewness and apply transformations if needed
x.skew()
```

```
mark               -0.891209
model              -0.227651
year               -0.143756
mileage             0.601181
engine_capacity     0.607818
transmission        4.212776
drive               3.333863
hand_drive        -12.165646
fuel                2.400797
dtype: float64
```

```
from sklearn.preprocessing import PowerTransformer
pt = PowerTransformer()
```

```
pt.fit_transform(x)
```

```
array([[-0.32907868, -0.08466299,  1.16159072, ..., -0.29491934,
         0.08170415,  0.02486769],
       [-0.32907868, -0.22222394, -0.36152605, ..., -0.29491934,
         0.08170415,  0.02486769],
       [ 0.84574664, -1.73171183,  0.53867105, ..., -0.29491934,
         0.08170415,  0.02486769],
       ...,
       [ 0.41806198,  0.67231296, -0.36152605, ..., -0.29491934,
         0.08170415,  0.02486769],
       [-1.55933022,  1.56794121, -1.77497058, ...,  3.3907572 ,
         0.08170415,  0.02486769],
       [ 0.84574664, -0.71542592, -0.36152605, ..., -0.29491934,
         0.08170415,  0.02486769]])
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaler = scaler.fit_transform(x)
```

```
x_scaler
```

```
array([[-0.10981305, -0.098294  ,  1.15067023, ..., -0.29092489,
         0.08170415,  0.02420349],
       [-0.10981305, -0.23649894, -0.33869579, ..., -0.29092489,
         0.08170415,  0.02420349],
       [ 0.78854905, -1.71068492,  0.55492382, ..., -0.29092489,
         0.08170415,  0.02420349],
       ...,
       [ 0.48909502,  0.6695112 , -0.33869579, ..., -0.29092489,
         0.08170415,  0.02420349],
       [-1.60708322,  1.59087744, -1.82806181, ...,  3.24709712,
         0.08170415,  0.02420349],
       [ 0.78854905, -0.72789427, -0.33869579, ..., -0.29092489,
         0.08170415,  0.02420349]])
```

```
#Lets check VIF
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif['Score'] = [variance_inflation_factor(x_scaler,i) for i in range(x_scaler.shape[1])]
vif['features'] = x.columns
vif
```

	Score	features
0	1.183166	mark
1	1.188354	model
2	1.195452	year
3	1.073293	mileage
4	1.179190	engine_capacity
5	1.144746	transmission
6	1.130224	drive
7	1.312991	hand_drive
8	1.295940	fuel

In our model there is no multicollinearty problem.

Model Preparation

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, accuracy_score
```

Splitting the data into train and test

```
x_train, x_test, y_train, y_test = train_test_split(x_scaler, y, test_size=0.3, random_state=517)
```

Linear Regression

```
lr = LinearRegression()
lr.fit(x_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred = lr.predict(x_test)
print(f'Linear Regression R^2 Score: {r2_score(y_test, y_pred)}')
```

```
Linear Regression R^2 Score: 0.11480732471976263
```

```python
# Accuracy score
print("Model Training Score : ",lr.score(x_train,y_train))
print("Model Testing Score : ",lr.score(x_test,y_test))
```

```
Model Training Score :  0.06189263299183312
Model Testing Score :  0.11480732471976263
```

```python
#MAE
mean_absolute_error(y_test,y_pred)
```

```
217.15085350044944
```

```python
#MSE
mean_squared_error(y_test,y_pred)
```

```
68058.18985548979
```

```python
# Cross-Validation
cv_scores = cross_val_score(lr, x_train, y_train, cv=5)
print(f'Cross-Validation Scores: {cv_scores}')
```

```
Cross-Validation Scores: [0.04627349 0.06961918 0.0119702  0.03949414 0.07860815]
```

Decision Tree Regressor

```python
from sklearn.tree import DecisionTreeRegressor
```

```python
decision_tree = DecisionTreeRegressor(random_state=42)
```

```python
decision_tree.fit(x_train,y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

```python
y_pred = decision_tree.predict(x_test)
print(f'Decision Tree Regressor R^2 Score: {r2_score(y_test, y_pred)}')
```

```
Decision Tree Regressor R^2 Score: -0.3508244103302631
```

```python
# Calculate evaluation metrics for Decision Tree Regressor
mae_dt = mean_absolute_error(y_test, y_pred)
mse_dt = mean_squared_error(y_test, y_pred)
print(f'mae_dt: {mae_dt}')
print(f'mse_dt: {mse_dt}')
```

```
mae_dt: 248.00294550810014
mse_dt: 103858.36524300442
```

Random Forest Regressor

```python
from sklearn.ensemble import RandomForestRegressor
```

```python
rf_model = RandomForestRegressor(n_estimators=10, random_state=42)
```

```python
rf_model.fit(x_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor(n_estimators=10, random_state=42)
```

```python
y_pred = decision_tree.predict(x_test)
print(f'Random Forest Regressor R^2 Score: {r2_score(y_test, y_pred)}')
```

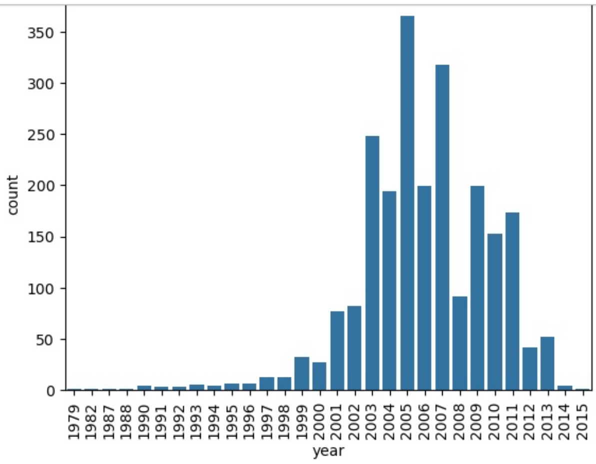
```
Random Forest Regressor R^2 Score: 0.1398395630021131
```

```python
# Evaluate the model
mae_rf = mean_absolute_error(y_test, y_pred)
mse_rf = mean_squared_error(y_test, y_pred)
```

```python
print(f'mae_rf: {mae_rf}')
print(f'mse_rf: {mse_rf}')
```

```
mae_rf: 214.95123934891967
mse_rf: 66133.58194457513
```

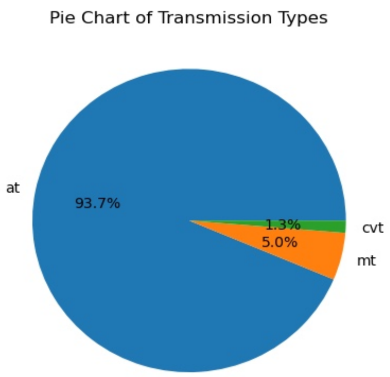
DATA VISUALIZATION

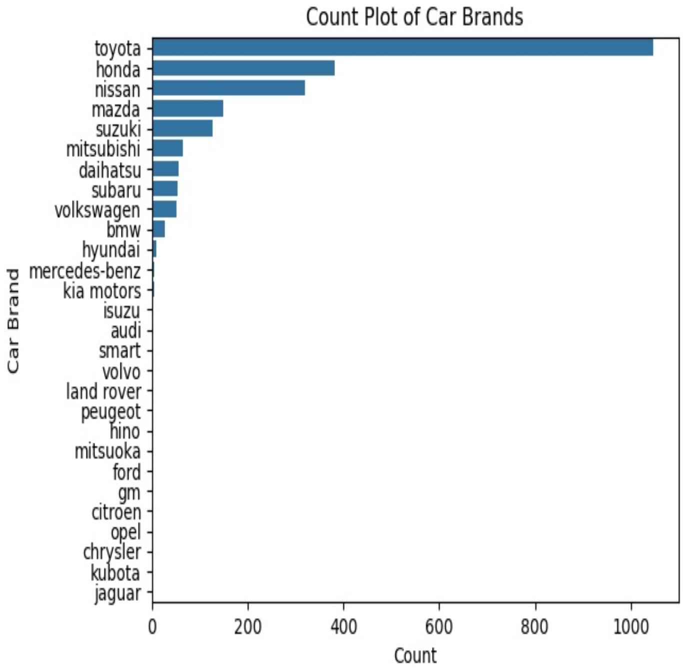


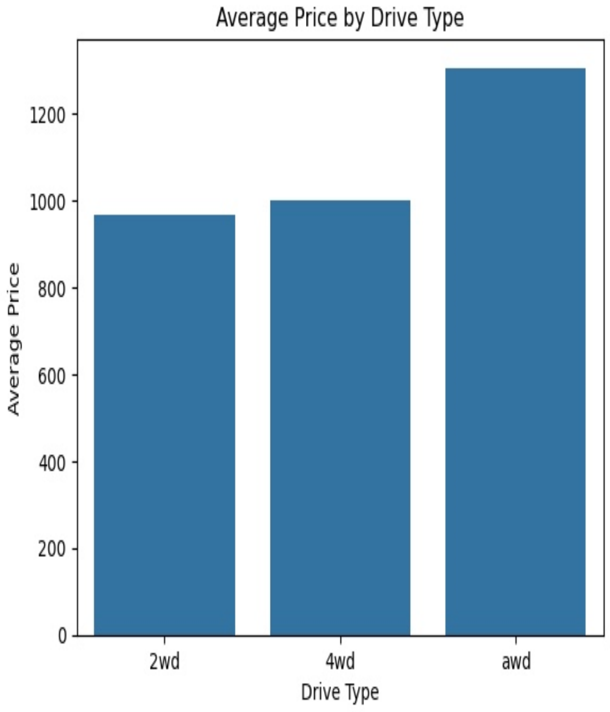

```
[9]:  year
      2005    365
      2007    318
      2003    248
      2006    199
      2009    199
      2004    194
      2011    173
      2010    153
      2008     91
      2002     82
      2001     77
      2013     52
      2012     42
      1999     32
      2000     27
      1998     13
      1997     13
      1996      6
      1995      6
      1993      5
      1994      4
      2014      4
      1990      4
      1991      3
      1992      3
      1979      1
      1982      1
      2015      1
      1987      1
      1988      1
      Name: count, dtype: int64
```

Here we can see that in 2005 most of the car sold.

```
#Lets transmission type
data['transmission'].value_counts().plot.pie(autopct='%1.1f%%')
plt.title('Pie Chart of Transmission Types')
plt.ylabel('')
plt.show()
```







Here we can see that awd cars tend to be priced higher on average, possibly due to their better performance in certain conditions.

RESULT

The result section presents the findings from the model evaluation, the performance of the predictive models.

Model Performance

1. **Linear Regression:**
    a. **Mean Absolute Error (MAE):** The MAE for the Linear Regression model was 217.15.
    b. **Mean Squared Error (MSE):** The MSE for Linear Regression model was 68058.18.
    c. **R-Squared Score:** The R-Squared score for Linear Regression model was 0.11480.
2. **Decision Tree Regressor:**
    a. **Mean Absolute Error (MAE):** The MAE for Decision Tree Regressor model was 214.95.
    b. **Mean Squared Error (MSE):** The MSE for Decision Tree Regressor model was 66133.58.
    c. **R-Squared Score:** The R-Squared score for Decision Tree Regressor model was 0.1398.
3. **Random Forest Regressor:**
    a. **Mean Absolute Error (MAE):** The MAE for Random Forest Regressor was 194.600.
    b. **Mean Squared Error (MSE):** The MSE for Random Forest Regressor was 57192.57.
    c. **R-Squared Score:** The R-Squared score for Random Forest Regressor was 0.2561.

Comparison of Models:

Comparing the performance of Linear Regression, Decision Tree Regressor and Random Forest Regressor it was observed that Random Forest Regressor has a significantly lower training MSE 57192.57 indicating a better fit to the training data and achieves a better R-Squared score on training data 0.2561 followed by Decision Tree Regressor, Linear Regression.

CONCLUSION

This project successfully predicts the price of used car with the Random Forest Regressor model with the help of various input features like brand, model, engine capacity, mileage, etc. This predictive capability helps the user to see if the car is fit in their budget and helps to negotiate with the salesman.