Predicting used car prices using regression methods

1. Introduction

Used cars are being sold more than new cars mainly because of their low price and how a new car on the other hand loses a lot of its value during its lifetime. Driving a brand-new car off the dealership lot will decrease its value approximately by 10%.[1] The price of a used car fluctuates a lot depending on the characteristic and condition of the car. Machine learning methods would assist us in predicting the prices of used cars that are for sale in the future, with understanding the data of used car prices.

The report will consist of 6 sections. Section 2 will explain the details of the problem, label, feature and the source of the dataset used in this report. Section 3 will include information about the utilization of the method. In section 4 the results will be shown and then put into words in section 5. The sources will be listed in section 6.

2. Problem Formulation

The dataset used to process is a collection of information about used cars such as name and model of the car, price, year, seller type, etc. The dataset includes details of 301 used cars collected by Nehal Birla from different websites, the dataset was found in Kaggle [2].

The goal of this ML problem is to predict the selling price of used cars based on various factors and features. This can help buyers and sellers estimate a fair market value for a used car, leading to more informed decisions. The features of this problem are going to be name of the cars, type of sellers, model years, type of transmission, model of the cars, kilometers the cars have been driven, type of fuel, brand of the cars. These features are vital to determining the correct price for a used car. The label values are going to future market prices of the used cars. There are categorical data types such as transmission, seller and fuel type.

3. Methods

3.1 Dataset

This dataset has 301 datapoints and the dataset includes all the details mentioned above. All the datapoints will be used, except irrelevant information such as the name of the car Understanding specific patterns can enhance the accuracy of predicting future market prices compared to relying solely on annual averages that tend to capture broader market trends. This approach enables more precise predictions when working with this dataset, as it hones in on distinct patterns rather than providing overly generalized insights. There are all types of values in this dataset. There are integers, floats and strings.

3.2 Linear regression model

Starting with a Linear regression model, since it is simple and easily interpretable as it provides a clear presentation of the correlation between the independent variables (features) and the target variable (used car price). However, linear regression model may not hold for all variables. Used car prices can be influenced by complex, nonlinear interactions between factors.

3.3 Decision tree model

Decision tree models offer unique advantages when it comes to predicting future used car prices. Notably, their strong suit lies in their interpretability. Decision trees provide a straightforward and intuitive representation of the decision-making process, making it easy to comprehend the factors that contribute to the predicted car price. In the context of used cars, where consumers and dealers often need to justify or explain price estimates, this interpretability is a highly valuable feature. It ensures transparency and trust in the pricing model's decision logic.

Furthermore, decision trees are particularly adept at capturing non-linear relationships in the data. Used car prices are influenced by a multitude of variables, and these relationships can be intricate and non-linear. Features like mileage or model year may exert exponential or non-linear effects on car prices. Decision trees can effectively model such intricate patterns, providing more accurate predictions compared to linear models. Their flexibility to handle non-linearity allows them to better approximate the complexities inherent in used car pricing.

3.4 Loss functions

Mean Absolute Error (MAE) and Mean Squared Error (MSE) were appropriate evaluation metrics for this problem because they effectively quantify the prediction accuracy and the magnitude of errors in the context of predicting used car prices. MAE provides a straightforward measure of the average absolute difference between predicted and actual prices, making it easy to interpret in terms of currency. MSE, on the other hand, gives higher weight to larger errors and is sensitive to outliers, which can be essential when assessing the impact of significant price deviations. These metrics are particularly relevant in a real-world scenario where accurate pricing predictions are vital for both buyers and sellers, and understanding the magnitude and distribution of errors is crucial for informed decision-making.

3.5 Training, validation, test

Not having tested yet which format of data splitting would be the best nor divided the data yet, I decided to go with the 70-15-15 split ratio, in this manner, we ensure that the model is robust, not overfit to the training data, and capable of providing accurate predictions for a wide range of used cars. It also allows us to assess the model's performance, adjust as needed, and have confidence in its ability to predict used car prices accurately when deployed in a real-world scenario. The data is randomly shuffled to ensure that it's not sorted in any

particular order that might introduce bias. This is particularly important if the data is collected in chronological order, as it prevents temporal bias.

4. Results

4.1 Linear regression model

The Linear Regression model, when applied to the used car price prediction task, exhibits the following performance on both the validation and test sets:

	Validation set	Test set
Mean Absolute Error (MAE)	1.742	1.218
Mean Squared Error (MSE)	14.598	3.683
R2 Score	0.631	0.812

The Linear Regression model shows decent predictive performance, but it has limitations in capturing complex, non-linear relationships between the car's features and its selling price. The R2 score of 0.812 on the test set indicates that the model explains a substantial portion of the variance but may not capture all the nuances in the data. The model's test set MAE and MSE are also relatively high, suggesting some degree of error in price prediction.

4.2 Decision tree model

In contrast, the Decision Tree model provides the following performance metrics on both the validation and test sets:

	Validation set	Test set
Mean Absolute Error (MAE)	0.876	0.624
Mean Squared Error (MSE)	2.674	1.301
R2 Score	0.933	0.933

The Decision Tree model excels in capturing the complex relationships in the used car price dataset. It displays significantly lower MAE and MSE, implying more accurate price predictions. The R2 score, reaching 0.933, is close to its maximum value of 1, indicating an excellent fit to both the validation and test data. This suggests that the Decision Tree model is better suited for this specific used car price prediction task.

4.3 Comparison between the models

The Decision Tree model consistently outperforms the Linear Regression model on both the validation and test sets in terms of MAE, MSE, and R2 score.

The Decision Tree model demonstrates a superior ability to capture the complex relationships between the car's features and selling price, which is often the case in the used car market.

The test error of the Decision Tree model (MAE: 0.624, MSE: 1.301, R2 score: 0.933) indicates its strong performance on unseen data.

5. Conclusion

In this report, we addressed the problem of predicting used car prices based on various features and attributes of the cars. We used a dataset containing information about 301 used cars to develop and evaluate machine learning models for price prediction. Two main models were applied: Linear Regression and Decision Tree Regression.

In conclusion, while the Decision Tree model demonstrated remarkable performance in predicting used car prices, there are several avenues for improvement. Data quality remains a pivotal factor, as ensuring the dataset's completeness and accuracy can significantly enhance prediction outcomes. Furthermore, feature engineering, encompassing more informative attributes and considering additional factors like regional variations or vehicle history, may lead to more precise predictions. The application of ensemble models, hyperparameter tuning, and outlier handling also holds potential for enhancing model accuracy. Incorporating temporal data, when available, to account for trends and seasonality could be beneficial. Finally, understanding customer preferences and their influence on pricing is an area of exploration that could further refine the model's predictive capabilities. Addressing these aspects would contribute to more robust and accurate predictions in the field of used car pricing.

6. References

[1] https://www.ramseysolutions.com/saving/car-depreciation.

[2] https://www.kaggle.com/datasets/nehalbirla/vehicle-dataset-from-cardekho

MLproject

October 11, 2023

[48]:
```
import numpy as np                    # import numpy package under shorthand "np"
import pandas as pd                   # import pandas package under shorthand␣
 ↪"pd"
import matplotlib.pyplot as plt
from nose.tools import assert_equal
from numpy.testing import assert_array_equal
from matplotlib import style
%matplotlib inline
%config Completer.use_jedi = False  # enable code auto-completion
import seaborn as sns
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures, StandardScaler    #␣
 ↪function to generate polynomial and interaction features
from sklearn.linear_model import LinearRegression    # classes providing Linear␣
 ↪Regression with ordinary squared error loss and Huber loss, respectively
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error ␣
 ↪# function to calculate mean squared error
```

[49]:
```
# read the data from the file "autos.csv" and store
# it in the dataframe "data"

data = pd.read_csv('car data.csv')

data.head(5)
```

[49]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

```
3      Dealer       Manual      0
4      Dealer       Manual      0
```

[50]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       301 non-null    object
 1   Year           301 non-null    int64
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64
 5   Fuel_Type      301 non-null    object
 6   Seller_Type    301 non-null    object
 7   Transmission   301 non-null    object
 8   Owner          301 non-null    int64
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

[51]:
```
data.isnull().sum()
```

[51]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

[52]:
```
data.describe()
```

[52]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.644115	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

[53]:
```
data.columns
```

[53]:
```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

[54]:
```
fuel_type = data['Fuel_Type']
seller_type = data['Seller_Type']
transmission_type = data['Transmission']
selling_price = data['Selling_Price']
```

[55]:
```
#manual encoding
data.replace({'Fuel_Type':{'Petrol':0, 'Diesel':1, 'CNG':2}}, inplace=True)
#one hot encoding
data = pd.get_dummies(data, columns=['Seller_Type', 'Transmission'],␣
 ↪drop_first=True)
```

[56]:
```
X = data.drop(['Car_Name','Selling_Price'], axis=1)
y = data['Selling_Price']

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.5,␣
 ↪random_state=42)

X_val, X_test, y_val, y_test = train_test_split(X_val, y_val, test_size=0.5,␣
 ↪random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
X_test = scaler.transform(X_test)

linear_model = LinearRegression()
linear_model.fit(X_train, y_train)

decision_tree_model = DecisionTreeRegressor(random_state=42)
decision_tree_model.fit(X_train, y_train)

pred_val_linear = linear_model.predict(X_val)
pred_val_tree = decision_tree_model.predict(X_val)

pred_test_linear = linear_model.predict(X_test)
pred_test_tree = decision_tree_model.predict(X_test)

linear_mae_val = mean_absolute_error(y_val, pred_val_linear)
linear_mse_val = mean_squared_error(y_val, pred_val_linear)
linear_r2_val = r2_score(y_val, pred_val_linear)
```

```
tree_mae_val = mean_absolute_error(y_val, pred_val_tree)
tree_mse_val = mean_squared_error(y_val, pred_val_tree)
tree_r2_val = r2_score(y_val, pred_val_tree)

linear_mae_test = mean_absolute_error(y_test, pred_test_linear)
linear_mse_test = mean_squared_error(y_test, pred_test_linear)
linear_r2_test = r2_score(y_test, pred_test_linear)

tree_mae_test = mean_absolute_error(y_test, pred_test_tree)
tree_mse_test = mean_squared_error(y_test, pred_test_tree)
tree_r2_test = r2_score(y_test, pred_test_tree)

print("Linear Regression Validation Set Metrics:")
print("MAE: ", linear_mae_val)
print("MSE: ", linear_mse_val)
print("R2 score: ", linear_r2_val)

print("Decision Tree Validation Set Metrics:")
print("MAE: ", tree_mae_val)
print("MSE: ", tree_mse_val)
print("R2 score: ", tree_r2_val)

print("Linear Regression Test Set Metrics:")
print("MAE: ", linear_mae_test)
print("MSE: ", linear_mse_test)
print("R2 score: ", linear_r2_test)

print("Decision Tree Test Set Metrics:")
print("MAE: ", tree_mae_test)
print("MSE: ", tree_mse_test)
print("R2 score: ", tree_r2_test)
```

```
Linear Regression Validation Set Metrics:
MAE:  1.742460869634859
MSE:  14.598076215476528
R2 score:  0.630525233132561
Decision Tree Validation Set Metrics:
MAE:  0.8764000000000003
MSE:  2.6469213333333332
R2 score:  0.9330068819949738
Linear Regression Test Set Metrics:
MAE:  1.2180690193977715
MSE:  3.6833127014913516
R2 score:  0.8116668616807556
Decision Tree Test Set Metrics:
MAE:  0.624078947368421
MSE:  1.3009960526315787
```

```
R2 score:  0.9334781786423274
```