Prediction of Second-Hand Car Prices

Task

The current price variable is the target variable of our dataset, while the other variables such as the maximum speed, age, and torque of a car are the feature variables that determine the price of a car. because, as the mileage of a car increases, its price decreases.

In [2]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import  mean_absolute_error
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
import matplotlib.pyplot as plt

a = pd.read_csv("train.csv")
```

Analysis of Variables

"Rating", "v.id", and "condition" variables are categorical variables. For the "Rating" variable, 1 represents the lowest degree, and 5 represents the best score. The "condition" variable represents the values given to the car in the best condition, and it has an ordinal structure. The "Economy" variable is could be a categorical variable that takes values between 8 and 15. All these variables have been transformed into qualitative values like the other variables in the dataset.

Since we have not found any missing values when checking the variables in the dataset, we do not need to perform any operations such as deleting observations or assigning a value to missing values. Lastly we have 12 column 1000 observation.

In [3]:
```
print(a.describe())
```

```
             v.id   on road old  on road now        years            km  \
count  1000.000000   1000.000000   1000.0000  1000.000000   1000.000000
mean    500.500000 601648.286000 799131.3970     4.561000 100274.430000
std     288.819436  58407.246204  57028.9502     1.719079  29150.463233
min       1.000000 500265.000000 700018.0000     2.000000  50324.000000
25%     250.750000 548860.500000 750997.7500     3.000000  74367.500000
50%     500.500000 601568.000000 798168.0000     5.000000 100139.500000
75%     750.250000 652267.250000 847563.2500     6.000000 125048.000000
max    1000.000000 699859.000000 899797.0000     7.000000 149902.000000

           rating    condition      economy  top speed          hp  \
count  1000.000000  1000.000000  1000.000000  1000.00000  1000.00000
mean      2.988000     5.592000    11.625000   166.89300    84.54600
std       1.402791     2.824449     2.230549    19.28838    20.51694
min       1.000000     1.000000     8.000000   135.00000    50.00000
25%       2.000000     3.000000    10.000000   150.00000    67.00000
50%       3.000000     6.000000    12.000000   166.00000    84.00000
75%       4.000000     8.000000    13.000000   184.00000   102.00000
max       5.000000    10.000000    15.000000   200.00000   120.00000

           torque  current price
count  1000.000000     1000.00000
mean    103.423000   308520.24250
std      21.058716   126073.25915
min      68.000000    28226.50000
25%      85.000000   206871.75000
50%     104.000000   306717.75000
75%     121.000000   414260.87500
max     140.000000   584267.50000
```

In [9]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   v.id           1000 non-null   int64
 1   on road old    1000 non-null   int64
 2   on road now    1000 non-null   int64
 3   years          1000 non-null   int64
 4   km             1000 non-null   int64
 5   rating         1000 non-null   int64
 6   condition      1000 non-null   int64
 7   economy        1000 non-null   int64
 8   top speed      1000 non-null   int64
 9   hp             1000 non-null   int64
 10  torque         1000 non-null   int64
 11  current price  1000 non-null   float64
dtypes: float64(1), int64(11)
memory usage: 93.9 KB
```

Removing some variables from the dataset

When the p-value is greater than 0.05, the independent variable is not statistically significant in the model, and its effect may have arisen by chance. When looking at the "P>|t|" column, I noticed that the values for v.id, rating, economy, top speed, hp, and torque were greater than 0.05, indicating that these variables were not statistically significant for our model. Therefore, I decided to remove these variables.

In [5]:
```
y = a['current price']
x = a.drop(['current price'], axis=1)

x = x.assign(const = 1)
model = sm.OLS(y,x)
result = model.fit()
print(result.summary())
predictions = result.predict(x)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          current price   R-squared:                       0.995
Model:                            OLS   Adj. R-squared:                  0.995
Method:                 Least Squares   F-statistic:                 1.883e+04
Date:                Sat, 25 Mar 2023   Prob (F-statistic):               0.00
Time:                        15:30:35   Log-Likelihood:                -10488.
No. Observations:                1000   AIC:                         2.100e+04
Df Residuals:                     988   BIC:                         2.106e+04
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
v.id           0.6938      0.961      0.722      0.471      -1.193       2.580
on road old    0.5058      0.005    106.402      0.000       0.497       0.515
on road now    0.5003      0.005    102.907      0.000       0.491       0.510
years      -1618.4429    161.459    -10.024      0.000   -1935.285   -1301.601
km            -3.9964      0.010   -419.067      0.000      -4.015      -3.978
rating       233.1715    197.887      1.178      0.239    -155.155     621.498
condition   4631.3151     98.644     46.950      0.000    4437.740    4824.890
economy       58.1130    124.892      0.465      0.642    -186.971     303.197
top speed    -14.6198     14.416     -1.014      0.311     -42.910      13.671
hp            20.4506     13.573      1.507      0.132      -6.185      47.086
torque        -1.7369     13.160     -0.132      0.895     -27.562      24.088
const      -1.427e+04   6078.759     -2.347      0.019   -2.62e+04   -2340.057
==============================================================================
Omnibus:                       82.921   Durbin-Watson:                   2.038
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               95.942
Skew:                           0.734   Prob(JB):                     1.47e-21
Kurtosis:                       2.615   Cond. No.                     2.22e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctl
y specified.
[2] The condition number is large, 2.22e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Trianing a Linear Regression Model

In [6]:
```
a = pd.read_csv("train.csv")
y = a['current price']
x = a.drop(['current price','v.id','rating','economy','top speed','hp','torque']
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_
model = LinearRegression()
result = model.fit(x,y)
y_pred = model.predict(X_test)

# MAE (Mean Absolute Error) hesaplama
```

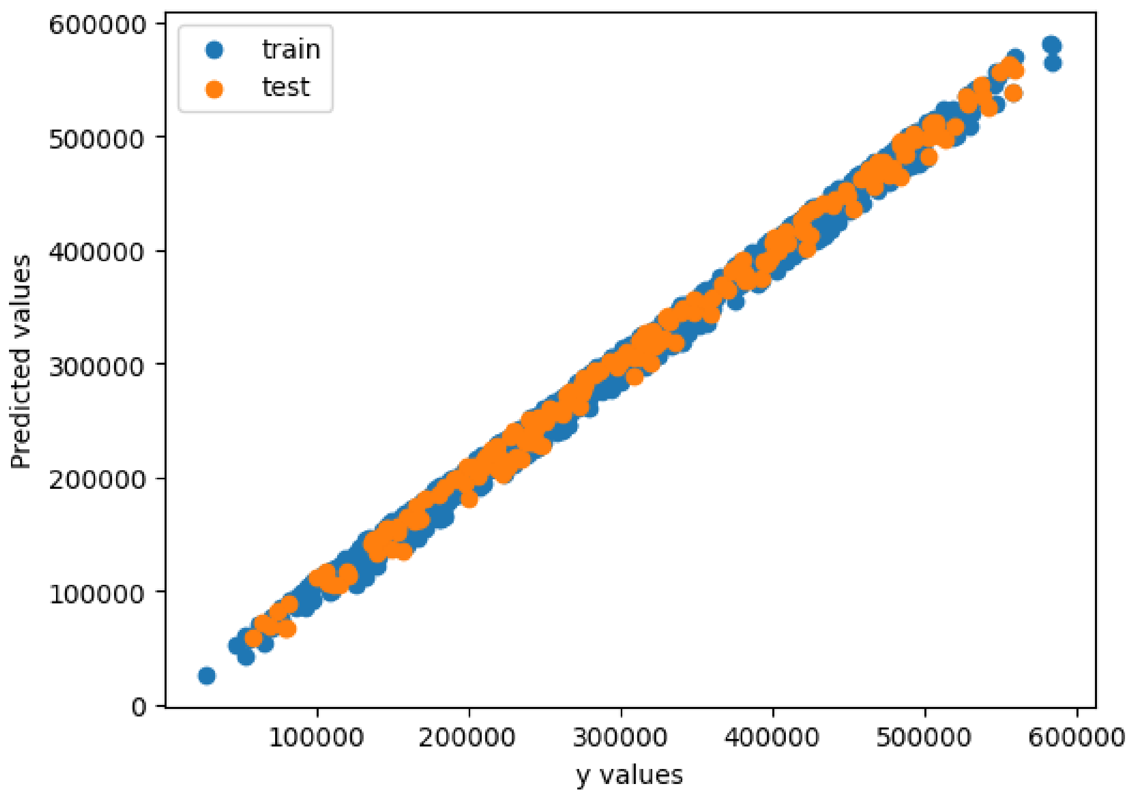
```
mae_train= mean_absolute_error(y_train, model.predict(X_train))
mae_test = mean_absolute_error(y_test, y_pred)

# Model skorunu hesaplama
score = model.score(X_test, y_test)

# Sonuçları yazdırma
print("MAE train:", mae_train)
print("MAE test:", mae_test)
print("Model Score:", score)
```

```
MAE train: 7262.713456482463
MAE test: 7351.841992887447
Model Score: 0.9953913395133458
```

In [7]:
```
plt.scatter(y_train, model.predict(X_train),label='train')
plt.scatter(y_test, y_pred  ,label='test')
plt.legend()
plt.xlabel("y values")
plt.ylabel("Predicted values")
plt.show()
```



In [8]:
```
a['current price']
```

Out[8]:
```
0      351318.0
1      285001.5
2      215386.0
3      244295.5
4      531114.5
         ...
995    190744.0
996    419748.0
997    405871.0
998     74398.0
999    414938.5
Name: current price, Length: 1000, dtype: float64
```

Determination of Performance Criteria

RMSE is a more accurate measure than MAE, as it takes into account the spread of the data. Therefore, RMSE may be a more suitable metric, especially in cases where there are large differences between the data or when there are outliers. However, since we did not observe such a difference in our data based on the graph, I did not see any problem using MAE as a performance metric.

Underfitting and Overfitting Control

when we look at current price vaiable and mae value and we compare them we conclude that mae value is not a big value for underfitting so we can say that there is not underfiting in our model.

When we compare the MAE values for the training and testing sets, we notice that train MAE smaller than test mae so we can say that there is a overfitting. Now, let's also look at the graph of y and predicted values to make sure.