PREDİCTİON OF SECOND-HAND CAR PRİCES

STUDENT'S:

Name – Lastname: Zeynep ÖZÇELİK

Student Number: 42388888986

Eskişehir

2023

```
[ ] import pandas as pd
    import numpy as np
    import seaborn as sns
    import matplotlib.pyplot as plt

    data = pd.read_csv('train.csv')
```

```
[ ] data.shape
```

```
(1000, 12)
```

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

	v.id	on road old	on road now	years	km	rating	condition	economy	top speed	hp	torque	current price
0	1	535651	798186	3	78945	1	2	14	177	73	123	351318.0
1	2	591911	861056	6	117220	5	9	9	148	74	95	285001.5
2	3	686990	770762	2	132538	2	8	15	181	53	97	215386.0
3	4	573999	722381	4	101065	4	3	11	197	54	116	244295.5
4	5	691388	811335	6	61559	3	9	12	160	53	105	531114.5

```
[ ] data.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
```

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

	v.id	on road old	on road now	years	km	rating	condition	economy	top speed	hp	torque	current price
count	1000.000000	1000.000000	1000.0000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.00000	1000.000000	1000.00000
mean	500.500000	601648.286000	799131.3970	4.561000	100274.430000	2.988000	5.592000	11.625000	166.89300	84.54600	103.423000	308520.24250
std	288.819436	58407.246204	57028.9502	1.719079	29150.463233	1.402791	2.824449	2.230549	19.28838	20.51694	21.058716	126073.25915
min	1.000000	500265.000000	700018.0000	2.000000	50324.000000	1.000000	1.000000	8.000000	135.00000	50.00000	68.000000	28226.50000
25%	250.750000	548860.500000	750997.7500	3.000000	74367.500000	2.000000	3.000000	10.000000	150.00000	67.00000	85.000000	206871.75000
50%	500.500000	601568.000000	798168.0000	5.000000	100139.500000	3.000000	6.000000	12.000000	166.00000	84.00000	104.000000	306717.75000
75%	750.250000	652267.250000	847563.2500	6.000000	125048.000000	4.000000	8.000000	13.000000	184.00000	102.00000	121.000000	414260.87500
max	1000.000000	699859.000000	899797.0000	7.000000	149902.000000	5.000000	10.000000	15.000000	200.00000	120.00000	140.000000	584267.50000

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

```
v.id             0
on road old      0
on road now      0
years            0
km               0
rating           0
condition        0
economy          0
top speed        0
hp               0
torque           0
current price    0
dtype: int64
```

```
[ ] data.isin(['?']).sum()
```

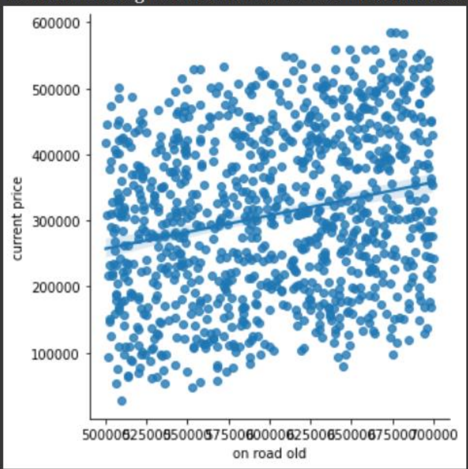
```
v.id             0
on road old      0
on road now      0
years            0
km               0
rating           0
condition        0
economy          0
top speed        0
hp               0
torque           0
current price    0
dtype: int64
```

```
[ ] data.duplicated().sum()
```

```
0
```

```
[ ] sns.lmplot(x="on road old", y="current price", data=data)
```

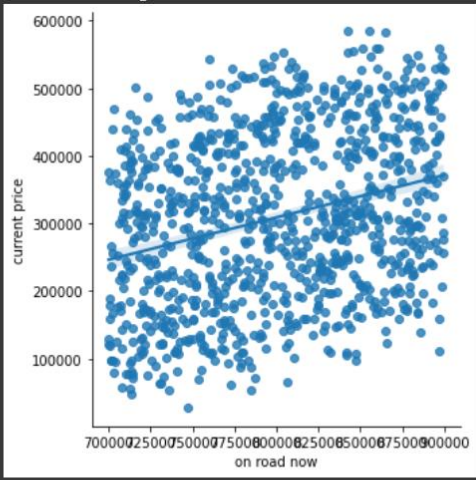
```
&lt;seaborn.axisgrid.FacetGrid at 0x7fd0a18b8700&gt;
```



There is a positive correlation between on road old and current price.

```
[ ] sns.lmplot(x="on road now", y="current price", data=data)
```

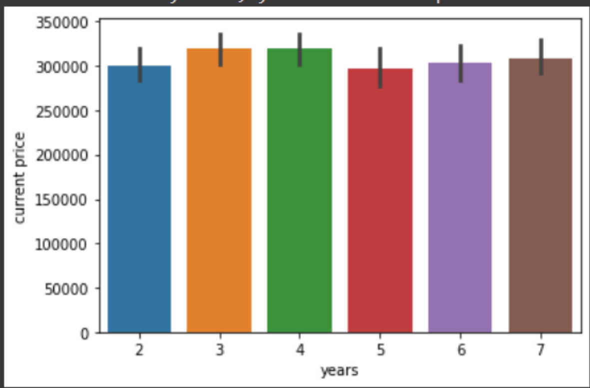
<seaborn.axisgrid.FacetGrid at 0x7fd09871bac0>



There is a positive correlation between on road now and current price.

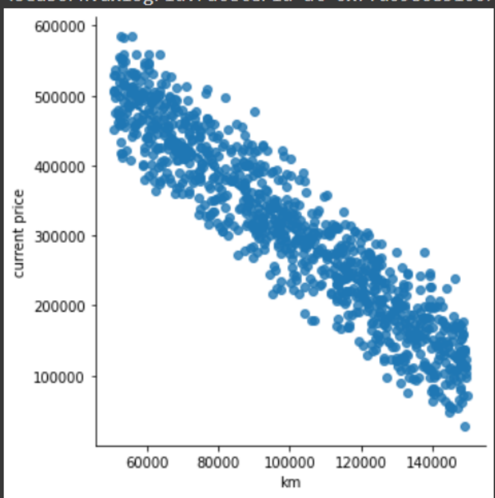
```
[ ] sns.barplot(x = data["years"], y = data["current price"])
```

<Axes: xlabel='years', ylabel='current price'>




```
[ ] sns.lmplot(x="km", y="current price", data=data)
```

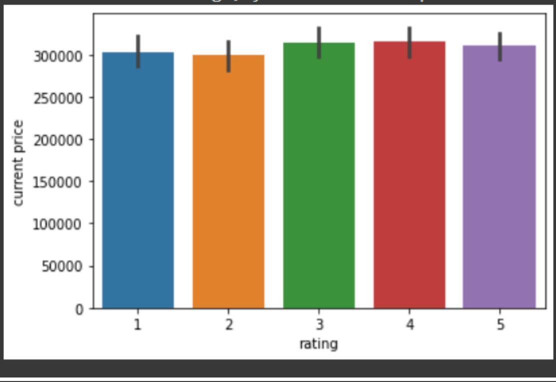
<seaborn.axisgrid.FacetGrid at 0x7fd098683100>



There is a negative correlation between km and current price.

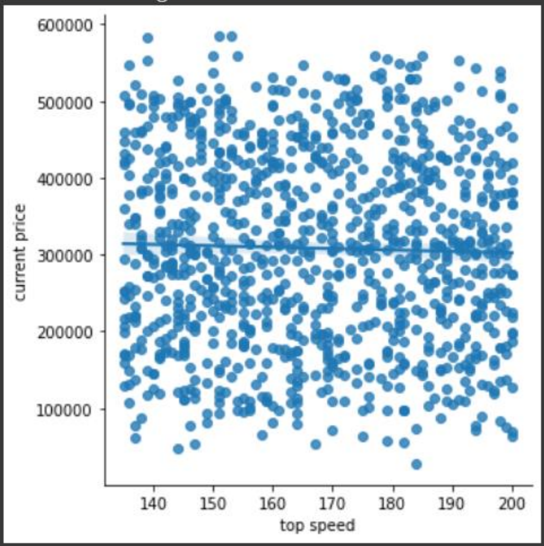
```
[ ] sns.barplot(x = data["rating"], y = data["current price"])
```

```
<Axes: xlabel='rating', ylabel='current price'>
```



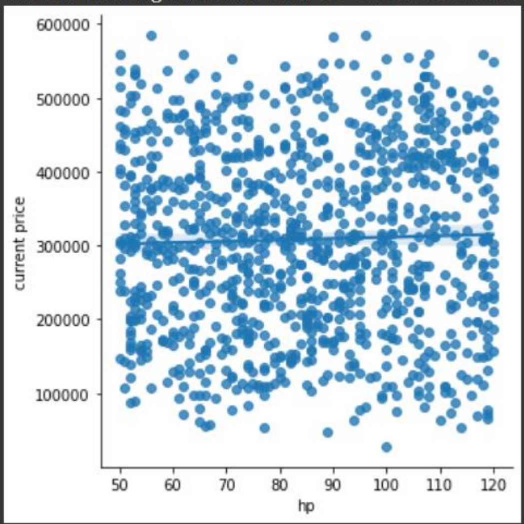

```
[ ] sns.lmplot(x="top speed", y="current price", data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fd09859a040>
```



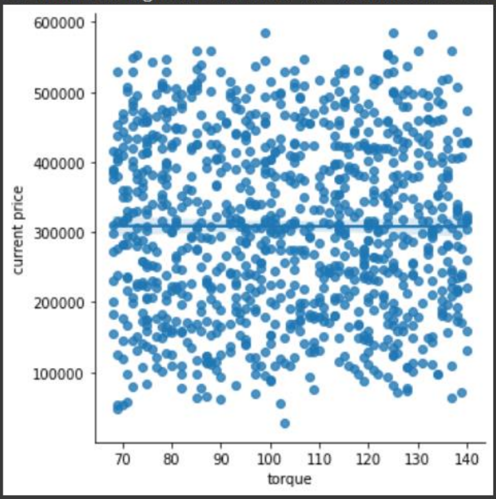

```
[ ] sns.lmplot(x="hp", y="current price", data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fd0984b02b0>
```



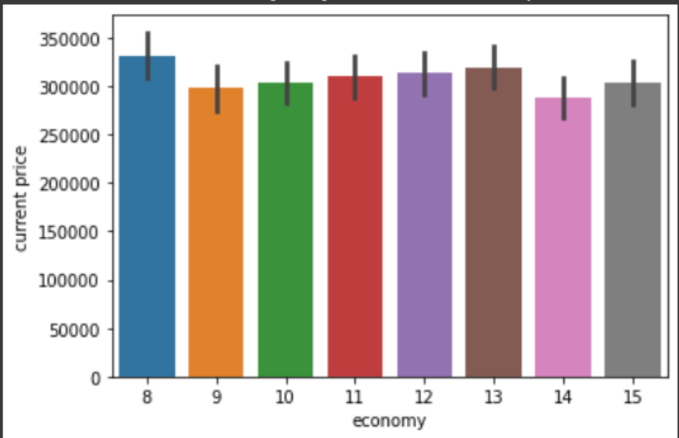
```
[ ] sns.lmplot(x="torque", y="current price", data=data)
```

<seaborn.axisgrid.FacetGrid at 0x7fd098543730>



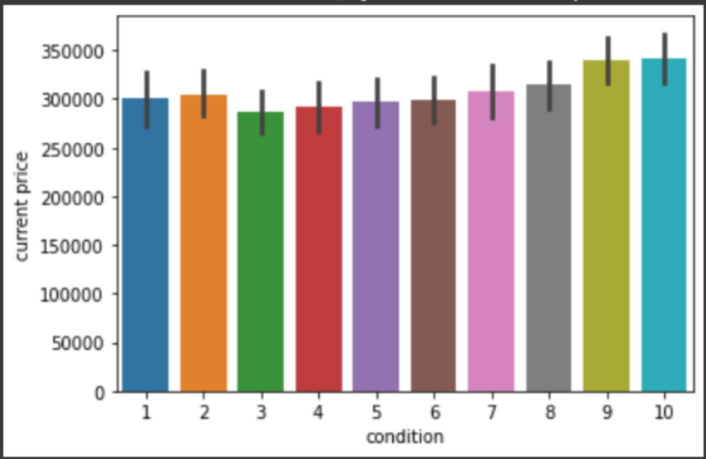

```
[ ] sns.barplot(x = data["economy"], y = data["current price"])
```

<Axes: xlabel='economy', ylabel='current price'>



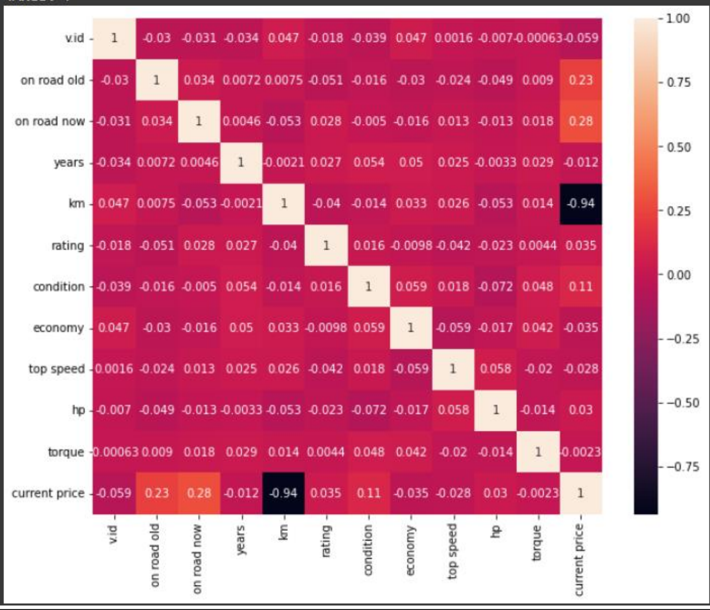

```
[ ] sns.barplot(x = data["condition"], y = data["current price"])
```

<Axes: xlabel='condition', ylabel='current price'>



```python
data.corr()
fig,ax=plt.subplots(figsize=(10,8))
sns.heatmap(data.corr(),annot=True)
```

<Axes: >



According to this correlation map, we can say that on road old, on road now and km are correlate with current price.

MACHINE LEARNING

Splitting

```python
y = data['current price'].values
x = data.drop('current price', axis=1).values
```

```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3, random_state = 42)
```

Scaling

```python
from sklearn.preprocessing import StandardScaler
standard = StandardScaler()
```

```python
x_train = standard.fit_transform(x_train)
```

```python
x_test = standard.fit_transform(x_test)
```

Lineer Regression

```
[ ] from sklearn.linear_model import LinearRegression
    linear_regression = LinearRegression()
    linear_regression.fit(x_train,y_train)
    y_pred_lin_reg = linear_regression.predict(x_test)
```

```
[ ] print("Score of the train set",linear_regression.score(x_train,y_train))
    print("Score of the test set",linear_regression.score(x_test,y_test))
```

```
Score of the train set 0.9953172816182181
Score of the test set 0.9940917055243087
```

```
[ ] from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_squared_log_error, r2_score

    print('R Square Score for Linear Regression : ', r2_score(y_test, y_pred_lin_reg))
    print("Mean squared error of the test set",mean_squared_error(y_test, y_pred_lin_reg))
    print("Root mean squared error of the test set", np.sqrt(mean_squared_error(y_test, y_pred_lin_reg)))
    print("Mean absolute error of the test set",mean_absolute_error(y_test, y_pred_lin_reg))
```

```
R Square Score for Linear Regression :  0.9940917055243087
Mean squared error of the test set 91621196.10605285
Root mean squared error of the test set 9571.89616042991
Mean absolute error of the test set 8314.408659638546
```

This problem is a regression problem so we can evoluate the model performance with MSE(Mean Squared Error) or R^2 score.

Train set score and test set score are similar. Because of this we can say there isn't overfitting or underfitting.