***Machine Learning Project-1 DURAN DUMAN 190101037***

**1. Data arrangement**

The data is arranged in two CSV files, **'trainDATA.csv'** and **'testDATA.csv'**, which are loaded into Pandas DataFrames for further processing.

The dataset contains information about used cars, including features like **year**, **km driven**, **fuel type**, **seller type**, **transmission**, and **owner**, also the **target variable** 'selling\_price'.

***The features I used for Linear Regression with Gradient Descent***

* year,
* km\_driven,
* fuel,
* seller type,
* transmission,
* owner.

***1.1 Encoding of Categorical Variables***

There are **categorical variables** at the data as fuel, seller\_type, transmission, owner.

**T**hese **Categorical variables** were **encoded to numerical values** using the following mapping dictionaries:

* **'fuel'**: {'Diesel': 1, 'Petrol': 2}
* **'seller\_type'**: {'Individual': 1, 'Dealer': 2, 'TrustmarkDealer': 3}
* **'transmission'**: {'Manual': 1, 'Automatic': 2}
* **'owner'**: {'FirstOwner': 1, 'SecondOwner': 2, 'ThirdOwner': 3,

'Fourth&AboveOwner': 4}

def encode\_categorical\_data(data, column\_name, encoding\_map):  
 data[column\_name] = data[column\_name].map(encoding\_map)  
  
fuel\_encoding = {'Diesel': 1, 'Petrol': 2}  
seller\_type\_encoding = {'Individual': 1, 'Dealer': 2, 'TrustmarkDealer': 3}  
transmission\_encoding = {'Manual': 1, 'Automatic': 2}  
owner\_encoding = {'FirstOwner': 1, 'SecondOwner': 2, 'ThirdOwner': 3, 'Fourth&AboveOwner': 4}  
  
encode\_categorical\_data(train\_data, 'fuel', fuel\_encoding)  
encode\_categorical\_data(train\_data, 'seller\_type', seller\_type\_encoding)  
encode\_categorical\_data(train\_data, 'transmission', transmission\_encoding)  
encode\_categorical\_data(train\_data, 'owner', owner\_encoding)  
  
encode\_categorical\_data(test\_data, 'fuel', fuel\_encoding)  
encode\_categorical\_data(test\_data, 'seller\_type', seller\_type\_encoding)  
encode\_categorical\_data(test\_data, 'transmission', transmission\_encoding)  
encode\_categorical\_data(test\_data, 'owner', owner\_encoding)

***1.2 Data Splitting***

The dataset was divided into training and test sets for machine learning.

For training data:

* Features (**'X\_train**') included columns: **'year'**, '**km\_driven**', **'fuel'**, **'seller\_type'**, **'transmission'**, and **'owner'**.
* The target variable (**'y\_train**') was **'selling\_price'**.

For test data:

* Features ('**X\_test'**) included columns: **'year'**, **'km\_driven'**, **'fuel'**, **'seller\_type'**, **'transmission'**, and **'owner'**.
* The target variable (**'y\_test'**) was **'selling\_price'**.

# Train verileri.  
X\_train = train\_data[['year', 'km\_driven', 'fuel', 'seller\_type', 'transmission', 'owner']].values  
y\_train = train\_data['selling\_price'].values  
# Test verileri.  
X\_test = test\_data[['year', 'km\_driven', 'fuel', 'seller\_type', 'transmission', 'owner']].values  
y\_test = test\_data['selling\_price'].values

***1.3 Feature Scaling***

**Min-max scaling** was applied to both the training and test datasets to normalize the feature values to the range [0, 1].

**X = (X – X.min) / (X.max- X.min) Y = (Y – Y.min) / (Y.max – Y.min)**

def min\_max\_scaling(data):  
 min\_vals = np.min(data,axis=0)  
 max\_vals = np.max(data,axis=0)  
 scaled\_data = (data - min\_vals) / (max\_vals - min\_vals)  
 return scaled\_data

***1.4 Learning Rates***

The learning rates that I used for the experiments,

* For the **first** and **second** experiments, I used **0.01** for two different iteration numbers.
* For the **third** and **fourth** experiments, I used **0.001** for two different iteration numbers.
* For the **fifth** experiment, I used **0.1**.

***1.5 For the estimations (Convergence Rule)***

Convergence Rule is based on the iteration numbers.

* For the **first** and **third** experiments **800** **iterations** applied and the results are received.
* For the **second** and **fourth** experiments **300 iterations** applied and the results are received.
* For the **fifth** experiment **100** iterations applied and the results are received.

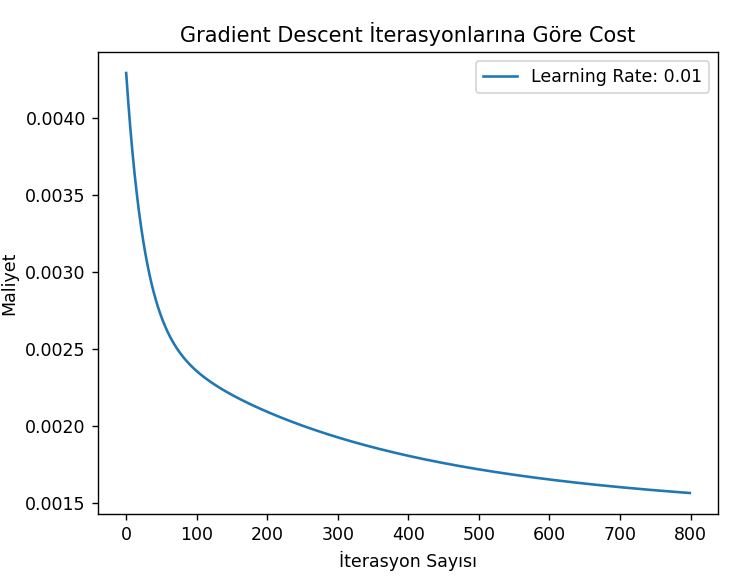
A plot was created to visualize the convergence of the cost function with the number of iterations.

**2 - Experiment Results**

**2.1 - For the first estimation, (alpha 0.01 , iterations 800)**

|  |  |
| --- | --- |
| **Alpha** : | **0.01** |
| **Number of iterations :** | **800** |
| **Initial Theta** **Value**: | [0,0,0,0,0,0,0,0] |
| **First Cost (error) Value**  of the “training dataset” : | 0.0042927576422540055 |
| **Last Cost Value (error)** of the “training dataset” : | 0.0015653437775611913 |
| **Final Theta :** | [ 0.03183822 0.04853344 -0.0026168 -0.0314302 0.02387306 0.06500636 -0.01443162] |
| **Error between “predicted y values” and “actual y values” on test Data**: (predicted y values AND actual y\_test values are scaled) | 0.17306532358236648 |

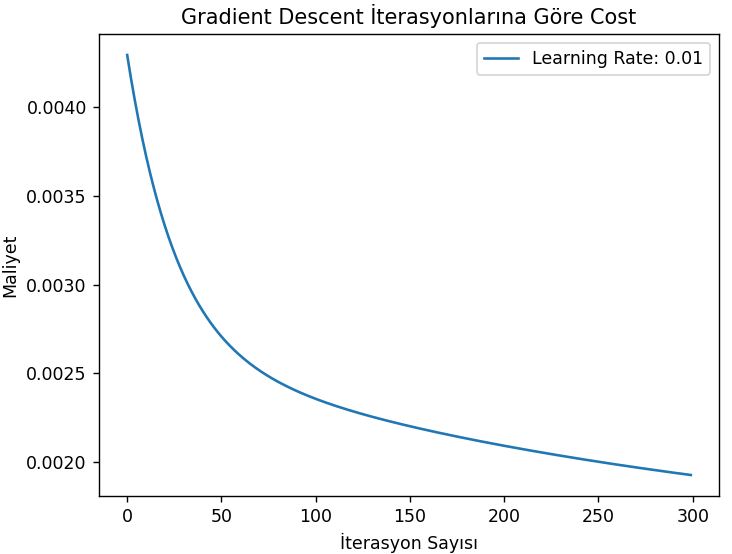
**Note : You can see the predictions in the “FirstEstimationIteration800.xlsx” file.**

****

**2.2 - For the second estimation, (alpha 0.01 , iterations 300)**

|  |  |
| --- | --- |
| **Alpha** : | **0.01** |
| **Number of iterations :** | **300** |
| **Initial Theta** **Value**: | [0,0,0,0,0,0,0,0] |
| **First Cost (error) Value**  of the “training dataset” : | 0.0042927576422540055 |
| **Last Cost Value (error)** of the “training dataset” : | 0.0019279424914852262 |
| **Final Theta :** | [ 0.0330776 0.0360186 0.00066683 -0.0119282 0.01475974 0.03351455 -0.00428797] |
| **Error between “predicted y values” and “actual y values” on test Data**: (predicted y values AND actual y\_test values are scaled) | 0.19821446087019273 |

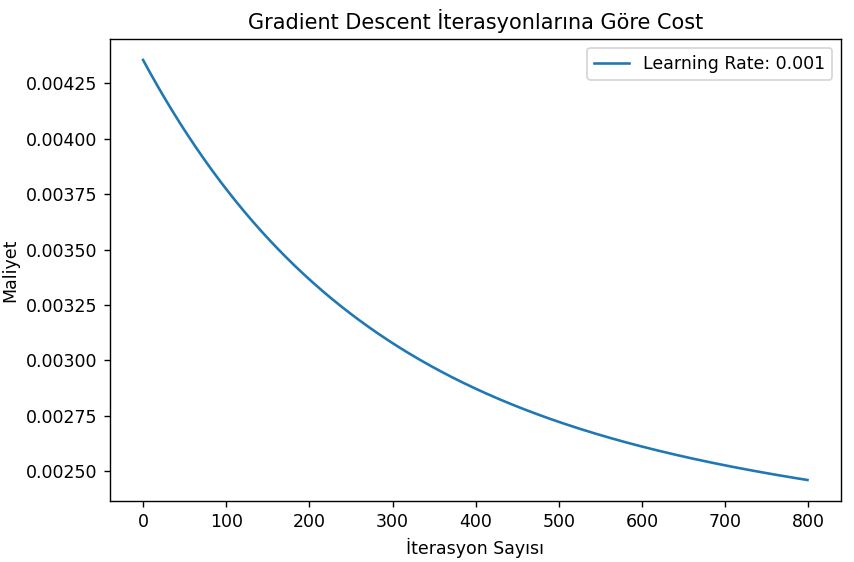
**Note : You can see the predictions in the “SecondEstimationIteration300.xlsx” file.**



**2.3 - For the third estimation, (alpha 0.001 , iterations 800)**

|  |  |
| --- | --- |
| **Alpha** : | **0.001** |
| **Number of iterations :** | **800** |
| **Initial Theta** **Value**: | [0,0,0,0,0,0,0,0] |
| **First Cost (error) Value**  of the “training dataset” : | 0.0043548197967425446 |
| **Last Cost Value (error)** of the “training dataset” : | 0.002461160490981018 |
| **Final Theta :** | [0.02491935 0.02210344 0.00143487 0.00377415 0.00673795 0.01184285 0.0009344 ] |
| **Error between “predicted y values” and “actual y values” on test Data**: (predicted y values AND actual y\_test values are scaled) | 0.23984881542651937 |

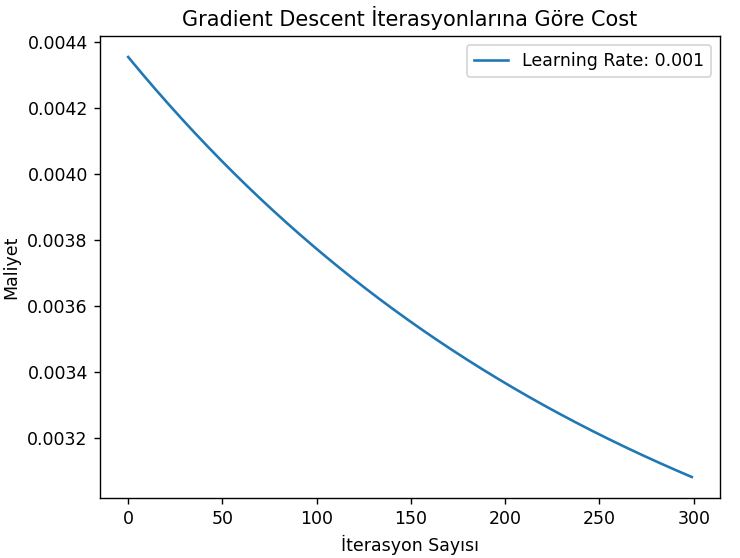
**Note : You can see the predictions in the “ThirdEstimationIteration800.xlsx” file.**

****

**2.4 - For the fourth estimation , (alpha 0.001 , iterations 300)**

|  |  |
| --- | --- |
| **Alpha** : | **0.001** |
| **Number of iterations :** | **300** |
| **Initial Theta** **Value**: | [0,0,0,0,0,0,0,0] |
| **First Cost (error) Value**  of the “training dataset” : | 0.0043548197967425446 |
| **Last Cost Value (error)** of the “training dataset” : | 0.0030806933756540944 |
| **Final Theta :** | [0.01372672 0.01163978 0.00087754 0.00370831 0.00321371 0.0050478 0.00096925] |
| **Error between “predicted y values” and “actual y values” on test Data**: (predicted y values AND actual y\_test values are scaled) | 0.2973649225201767 |

**Note : You can see the predictions in the “FourthEstimationIteration300.xlsx” file.**

****

**2.5 - For the fifth Estimation, (alpha 0.1 , iterations 100)**

|  |  |
| --- | --- |
| **Alpha** : | **0.1** |
| **Number of iterations :** | **100** |
| **Initial Theta** **Value**: | [0,0,0,0,0,0,0,0] |
| **First Cost (error) Value**  of the “training dataset” : | 0.0037288831114208197 |
| **Last Cost Value (error)** of the “training dataset” : | 0.0015108905706266587 |
| **Final Theta :** | [ 0.03044906 0.0518642 -0.00393246 -0.03467824 0.02587505 0.07356426 -0.01737949] |
| **Error between “predicted y values” and “actual y values” on test Data**: (predicted y values AND actual y\_test values are scaled) | 0.17011829993149022 |

**Note : You can see the predictions in the “FifthEstimationIteration100.xlsx” file.**

