School of Information Technologies and Engineering, ADA University

CSCI4734 – Machine Learning

Fall 2025

Assignment 1

Deadline: December 7, 2025, 23:59

This assignment has multiple components.

Linear Regression

For this part of the assignment, you will work with a collection of cars (*turboaz.csv*). Modify the provided Jupyter notebook file to complete the following tasks.

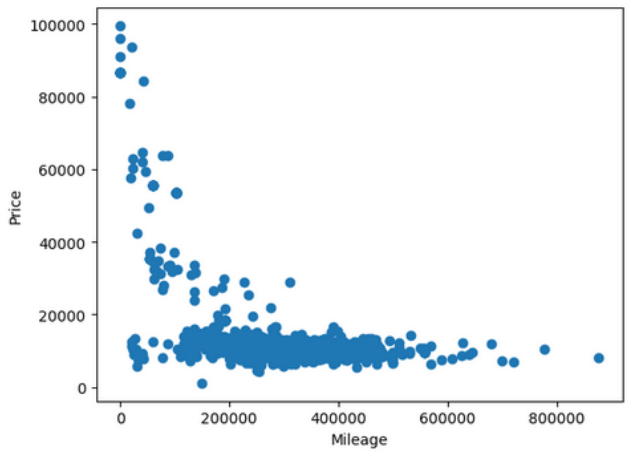
1. Data (5%)

Read the data from the file (*turboaz.csv*) using *pandas* and extract the following three columns for the task:

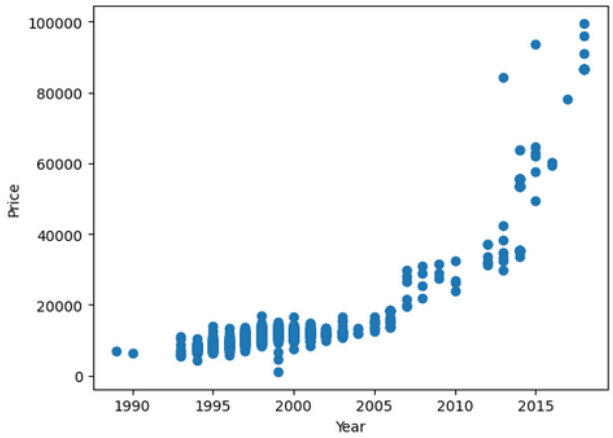
a) X_1 – *Yurush* (mileage), conversion to the numeric type may be needed
b) X_2 – *Buraxilish ili* (model year)
c) Y – *Qiymet* (price), normalizing to a single currency may be needed for different currencies (e.g., USD to AZN)

Using *matplotlib* library (*scatter*, *Axes3D*), provide the following visualizations.

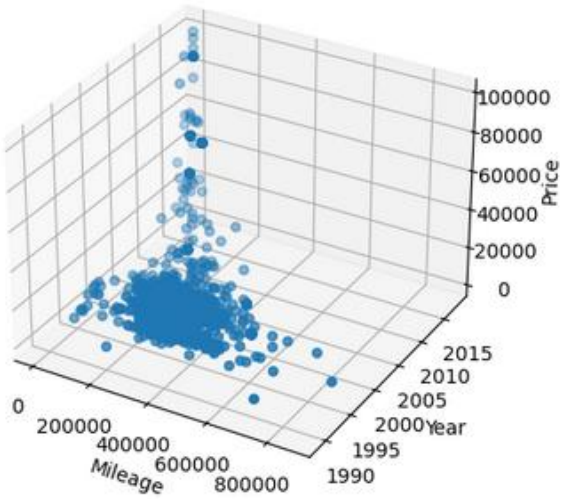
a) X_1 and Y



b) X_2 and Y



c) 3D plot of all three fields (X_1, X_2, Y)



Create copies of original features (will be needed later) and apply Z-score normalization to each feature $X_j, j \in \{1,2\}$ and target Y:

$$Z_{X_j} = \frac{X_j - \mu_{X_j}}{\sigma_{X_j}}$$

$$Z_Y = \frac{Y - \mu_Y}{\sigma_Y}$$

2. Linear regression from scratch (50%)

a) Implement the following hypothesis function (linear):

$$h_\Theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

b) Implement the following cost function:

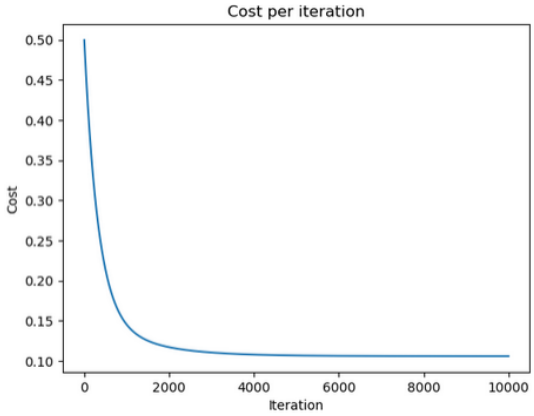
$$J(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\Theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

c) Implement gradient descent algorithm to minimize the cost function:
   - Create the data matrix with all three features.

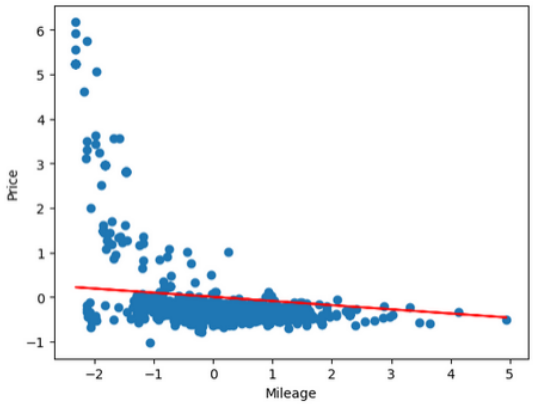
- Assign zero as initial values of $\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$.
- Learning rate: $\alpha = 0.001$, you can change it in different experiments.
- Number of iterations: 10000 or you can stop it when two sequential values are too close.
- Update parameter values using the update rule:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\Theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

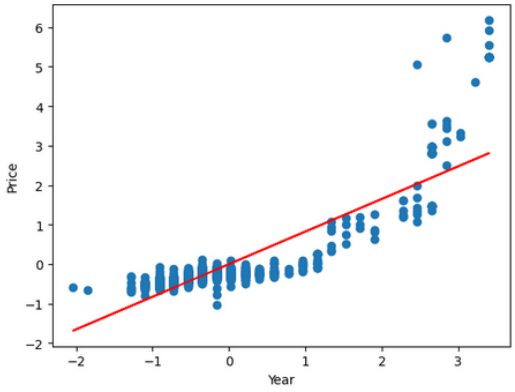
d) Plot the cost as a function of iterations:



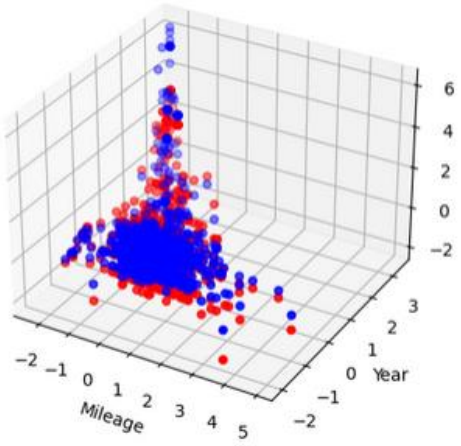
e) Plot points of X_1 and Y and draw a line of predictions made with optimal parameter values you got from the gradient descent.



f) Plot points of X_2 and Y and draw a line of predictions made with parameters you got from gradient descent.



g) Plot 3D graph of points of X_1, X_2, Y and the predicted Y using the same X_1 and X_2. It should look like this (blue points are true values, red points are predicted values):



h) Predict the prices for all cars and evaluate the model using *mean_squared_error* and *r2_score*.

i) Predict the prices for the given two cars that are not in the dataset:
Car 1 = {mileage: 240000, year: 2000, price: 11500}
Car 2 = {mileage: 415558, year: 1996, price: 8800}

3. Linear regression using library (10%)

Use *scikit-learn* library to perform linear regression on the unnormalized data. Complete 2(h) and 2(i) for the trained model.

4. Linear regression using normal equation (5%)

Implement normal equation in Python using *numpy* and apply it to solve the same regression problem.

Logistic Regression

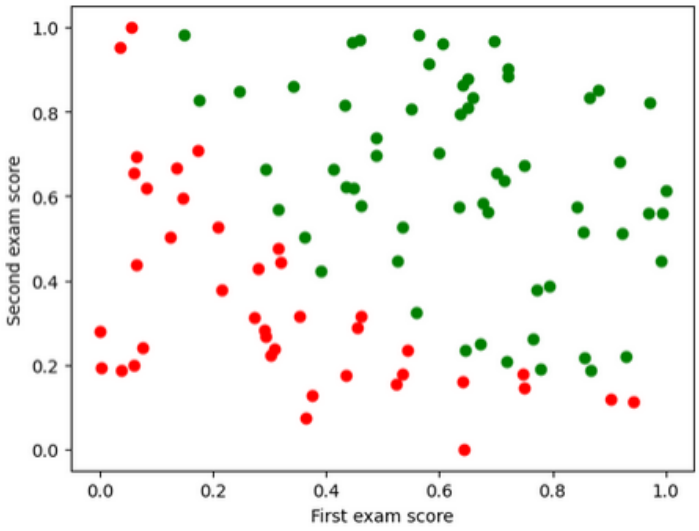
For this part of the assignment, you will work with students' exam results dataset (*exams.csv*). Modify the provided Jupyter notebook file to complete the following tasks.

1. Data (5%)

Read the data from the file (*exams.csv*) using *pandas*. It contains three columns. First two columns (exam_1, exam_2) are exam scores of a student, and the third value indicates whether the student has passed the course or not (1 or 0).

It is recommended to scale the data at this stage. You will see that gradient descent algorithm performs much better when the data is normalized. Use min-max scaling.

Plot a graph of the first and second exam scores and second exam scores. Color-code the positive class as green and negative class as red.



2. Logistic regression from scratch (15%)

a) Implement the following hypothesis function (sigmoid):

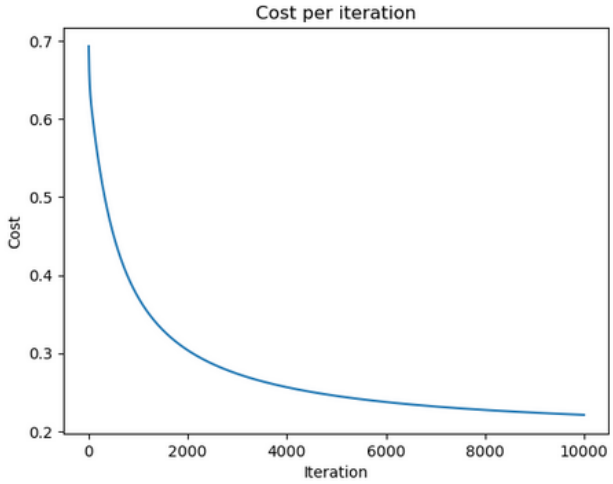
$$h_\Theta(x) = \sigma(\Theta^T x) = \frac{1}{1 + e^{-\Theta^T x}}$$

b) Implement the following cost function:

$$J(\Theta) = -\frac{1}{m}\left(\sum_{i=1}^{m} y^{(i)} \log h_\Theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\Theta\left(x^{(i)}\right)\right)\right)$$

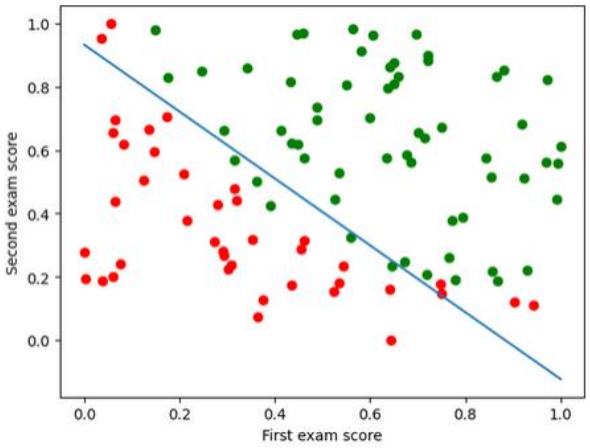
c) Use the gradient descent function implemented for the previous task (linear regression) to find the optimal parameter values. Use 0.1 for learning rate and run it for 10000 iterations. You should be able to change training steps and number of iterations through the variables (or input into the function). In addition, you should save the cost value at each iteration for plotting.

d) Plot the cost as a function of iterations:



e) Plot points of the first exam score and the second exam score. Admitted student points should be green and failed student points should be red. (same as

Visualization part). And plot the decision boundary using the parameters found by gradient descent on the same graph:



f) Predict the classes for all students with threshold 0.5 and evaluate the model using *accuracy_score*.
g) Predict the probability of passing the course for the given two students that are not in the dataset:
Student 1 = {1st exam score: 55, 2nd exam score: 70}
Student 2 = {1st exam score: 40, 2nd exam score: 60}

3. Logistic regression using library (10%)

Use *scikit-learn* library to perform logistic regression on the unnormalized data. Complete 2(f) and 2(g) for the trained model.

Submission

Please follow the instructions below when you make your submissions to the Blackboard System:

- The language of choice for assignments is Python. Consider using Jupyter Lab for your own convenience.
- Submit your solution as a **single** Jupyter notebook (.ipynb) file named according to the **template** (all capital):
CSCI4734_2025F_CRN_A1_FIRSTNAME_LASTNAME
- Do **not** submit your work in a compressed (archive) format such as .zip or .rar.
- You may submit your assignment up to **three times** before the deadline.
- Only your **latest submission** will be graded.
- A **25% penalty** will be applied for submissions up to **one day late**. Submissions received **after one day** will **not be accepted**.
- Please refer to the relevant section of the **course syllabus** for guidance on the **use of AI tools**.
- In addition, students whose submissions raise any **questions or concerns** may be asked to **provide explanations** of their work.