**Simple Linear Regression**

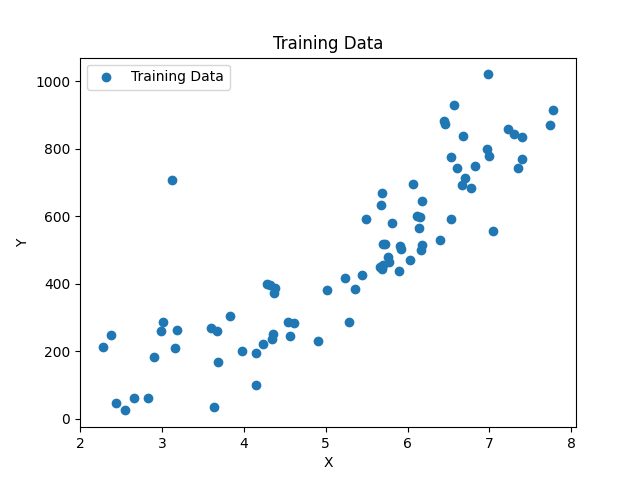
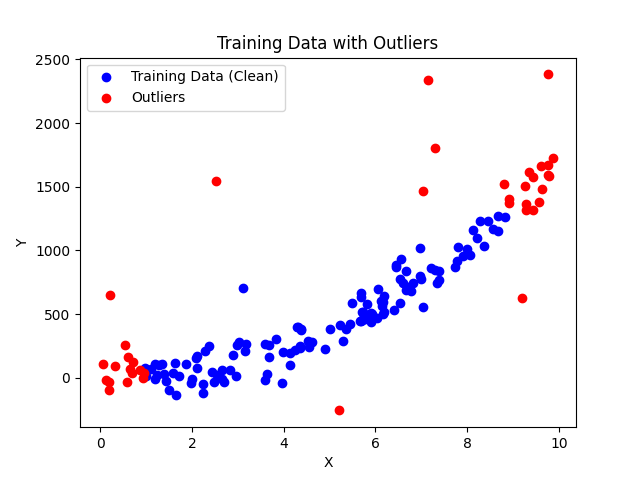
SV42/2020 Danilo Babić, SV61/2020 Jelena Miković

# Problem

The problem given to us was the prediction of variable Y based on the given X, which is a simple linear regression commonly used in statistic. In this report, the focus is on the implementation of a stochastic gradient descent (SGD) and minimizing the Root Mean Squared Error (RMSE).

# Data

The dataset used in this project comprises observations of predictor variables (X) and the corresponding target variable (Y). The training data-set had only 161 pairs, as shown on the right picture, and by removing the outliers that number dropped to around 130. Most of the data were positive numbers and clustered in a nice curved, which helped while training the model.



# Outliers

Outliers are observed data points that are far from the least-squares line. They have large errors, where the error or residual is not very close to the best-fit line. In this project, outliers are detected using the Z-score method, which measures the deviation of each data point from the mean in terms of standard deviations. Data points with Z-scores exceeding a predefined threshold are considered outliers and are subsequently removed from the dataset, shown on the picture above.

# Batch Gradient Descent (BGD)

Batch Gradient Descent (BGD) is a popular optimization algorithm used in machine learning and specifically in the context of training models, particularly in supervised learning scenarios. It is a first-order optimization algorithm aimed at minimizing a cost function (also known as loss function) that measures the difference between the model's predictions and the actual target values in the training data.

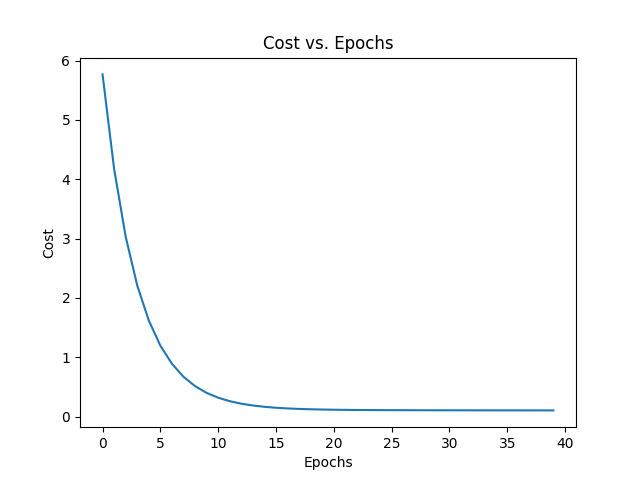
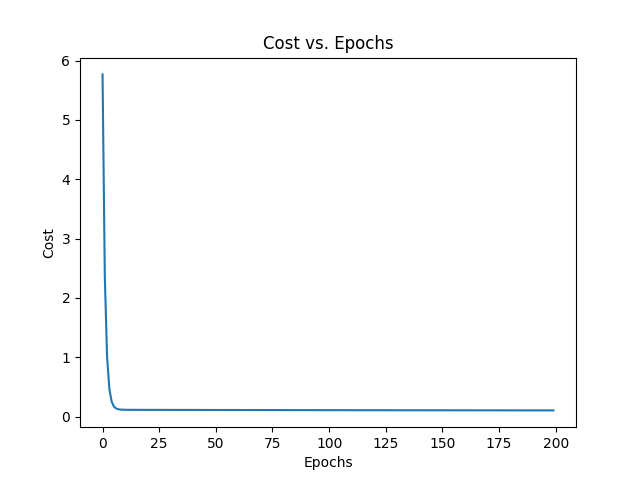
The gradient update step is performed using the entire dataset (x) in each iteration. This is indicative of batch gradient descent. In batch gradient descent, the gradients are computed using the entire dataset, as opposed to stochastic gradient descent where gradients are computed using individual samples, or mini-batch gradient descent where gradients are computed using a subset (mini-batch) of the dataset.

**Parameters:**

* Learning Rate: Determines the step size of each parameter update during optimization. It controls the speed of convergence and stability of the algorithm.
* Epochs: Defines the number of iterations over the entire training dataset during the optimization process.

**Cost Function:**

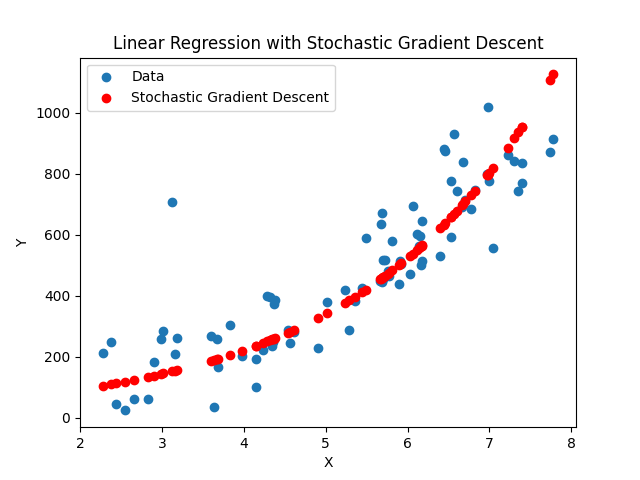
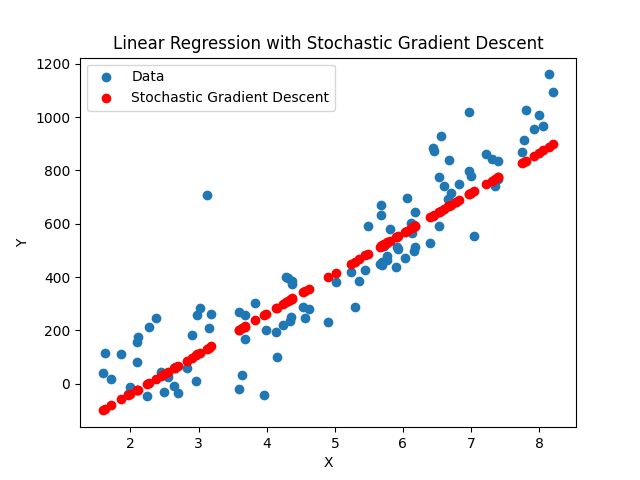
The cost function used in this project is the Root Mean Squared Error (RMSE), which measures the average squared difference between the predicted and actual values. The goal of the SGD algorithm is to minimize this cost function by adjusting the parameters of the linear regression model. On the first picture the it’s shown for learning rate of 0.5 and 40 epoches, the second one shows learning rate of 0.1 and 200 epochs. The first one has given a much smaller RMSE.



# Log Transformation

Logarithmically transforming variables in a regression model is a very common way to handle situations where a non-linear relationship exists between the independent and dependent variables, but they are also a convenient means of transforming a highly skewed variable into one that is more approximately normal. The second one is the reason we used this transformation in this project, helping us achive better results.

On the right we have a linear regression with Batch Gradient Descent and normalization, giving us a straight line as close as possible to the real values. On the left we have the same algorith and normalization, but we also use logarithm transformation, and the line we get follows the data more accuratly.



Bat

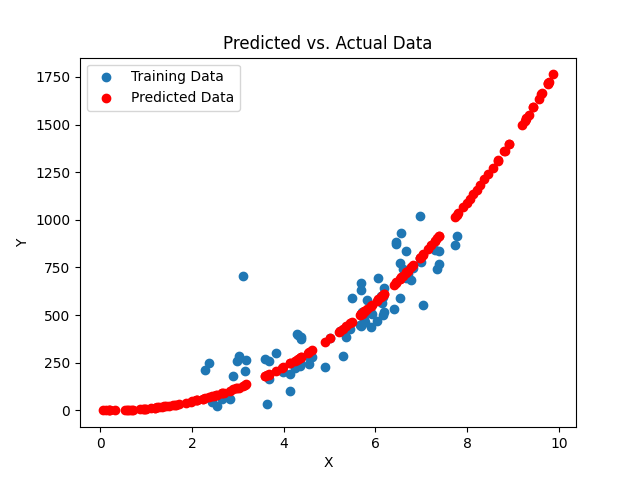
Bat

# RMSE and Results Analysis

Root Mean Squared Error (RMSE) is a commonly used metric to evaluate the performance of regression models. It measures the average magnitude of the errors between the predicted and actual values, with lower values indicating better model performance.

The final score we got on the testing data was around 82, beating the algorithm that we previously used that has normalization and logarithm transformation by more then 200. Stohastic Gradient Descent (SGD) and Normal Equation (NE) gave slightly worse results on a same data-set shown on a table below.

|  |  |  |
| --- | --- | --- |
| BGD | SGD | NE |
| ~ 246 | ~ 252 | ~ 258 |



# References

* <https://learn.saylor.org/mod/book/view.php?id=55086&chapterid=40788>
* <https://medium.com/@amannagrawall002/batch-vs-stochastic-vs-mini-batch-gradient-descent-techniques-7dfe6f963a6f>
* <https://kenbenoit.net/assets/courses/me104/logmodels2.pdf>
* <https://en.wikipedia.org/wiki/Root-mean-square_deviation>

Note: ChatGPT helped format some of the sentances above, we chose everything we wanted to say, he just made it sound better. ☺