

**Birzeit University**

**Faculty of Engineering and Technology**

**Department of Computer Systems Engineering**

**MACHINE LEARNING AND DATA SCIENCE**

**ENCS5341**

**EXP #5 Filters**

**Report #2**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Prepared by:**

Aws Shaheen 1212585

Ghassan Qandeel 1212397

**Instructor:** Dr. Yazan Abu Farha

**Section:** 1

**Date:** 28-11-2024

# Abstract

This assignment aims to build a series of regression models using a dataset, evaluate and compare their performance, and apply various techniques to improve model accuracy and prevent overfitting. It focuses on both linear and nonlinear registration model, feature selection methods and regularization techniques will be used also followed by hyperparameter tuning, to select the optimal model.

**Table of Contents**

[Abstract I](#_Toc183717346)

[1. Data set description 1](#_Toc183717347)

[2. Data preprocessing 1](#_Toc183717348)

[3. Building Regression Models: 2](#_Toc183717349)

[3.1. Linear models: 2](#_Toc183717350)

[3.1.1. Linear Regression: 2](#_Toc183717351)

[3.1.2. Lasso regression: 2](#_Toc183717352)

[3.1.3. Ridge Regression: 3](#_Toc183717353)

[3.2. Closed Form Solution: 3](#_Toc183717354)

[3.3. Nonlinear Models: 4](#_Toc183717355)

[3.3.1. Polynomial Regression: 4](#_Toc183717356)

[3.3.2. Radial Basis Function (RBF): 4](#_Toc183717357)

[4. . Model Selection Using Validation Set: 5](#_Toc183717358)

[5. Feature Selection with Forward Selection: 5](#_Toc183717359)

[6. Hyperparameter Tuning with Grid Search: 6](#_Toc183717360)

[References 7](#_Toc183717361)

**Table of Figures**

[Figure 1:missing values summary before cleaning 1](#_Toc183716735)

[Figure 2:missing values summary after cleaning 1](#_Toc183716736)

[Figure 3:linear regression performance 2](#_Toc183716737)

[Figure 4:Lasso for alpha 0.1 2](#_Toc183716738)

[Figure 5:Lasso for alpha 1 2](#_Toc183716739)

[Figure 6:Lasso for alpha 0.01 2](#_Toc183716740)

[Figure 7:Lasso for alpha 100 2](#_Toc183716741)

[Figure 8Lasso for alpha 10 2](#_Toc183716742)

[Figure 9:Ridge for alpha 1 3](#_Toc183716743)

[Figure 10:Ridge for alpha 0.1 3](#_Toc183716744)

[Figure 11:Ridge for alpha 0.01 3](#_Toc183716745)

[Figure 12:Ridge for alpha 100 3](#_Toc183716746)

[Figure 13:Ridge for alpha 10 3](#_Toc183716747)

[Figure 14:optimal alpha for lasso and ridge 3](#_Toc183716748)

[Figure 15:closed form solution performance 3](#_Toc183716749)

[Figure 16:Polynomial Regression performance 4](#_Toc183716750)

[Figure 17:Radial Basis Function performance 4](#_Toc183716751)

[Figure 18:best model performance 5](#_Toc183716752)

[Figure 19:features selection 5](#_Toc183716753)

# Data set description

The dataset refers to YallaMotors website [1], it includes around 6,750 rows and 9 columns, contains 9 features (car name, price, engine\_capacity, cylinder, horse\_power, top\_speed,

seats, brand, country). The main objective of this dataset is to predict car prices, making it ideal for developing regression models to understand the relationship between various features, the target variable is car price.

# Data preprocessing

first step we have standardized all currencies to USD for the target variable “Price”, then we cleaned the dataset by handling the missing values, encoding categorial features and standardizing numerical features.

To convert the price to USD we have used Mocki.io [2] to generate Json file contains the exchange price from each currency to USD [3].

For data cleaning we have cleaned each column separated each null values or wrong values that does not makes sense in the column it was replaced by Nan from the numpy library for the following features: seats, top\_speed, horse\_power, cylinder, engine\_capacity.

Then for the price column each row was checked if it has one of the currencies from the Json file [4] and for the wrong and null data, it was replaced with Nan from numpy library then the null values were taken as the test dataset.

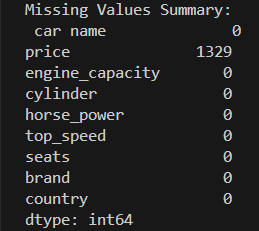
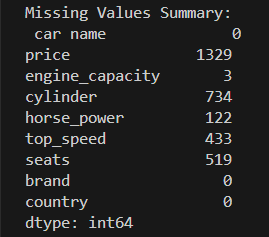
 

Figure :missing values summary before cleaning

Figure :missing values summary after cleaning

Then the data was split for three data sets, training, validation, test.

# Building Regression Models:

## Linear models:

### Linear Regression:

The linear regression was done using the built I function for the linear regression, then to test our model performance it was passed through three performance metrics and the results shown in the figure below:

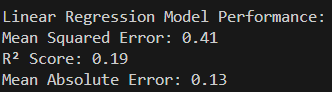
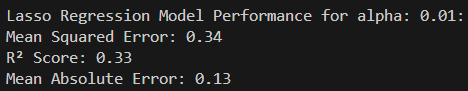
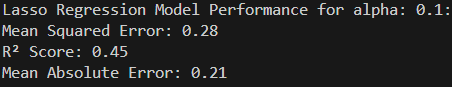


Figure :linear regression performance

### Lasso regression:

The Lasso regression was done using the built it function for Lasso for different alphas,”0.01,0.1,1,10,100”, and passed through three performance metrics and the results shown in the figures below:

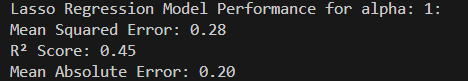


Figure :Lasso for alpha 0.1

Figure :Lasso for alpha 1

Figure :Lasso for alpha 0.01

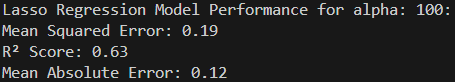
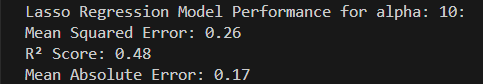


Figure :Lasso for alpha 100

Figure Lasso for alpha 10

### Ridge Regression:

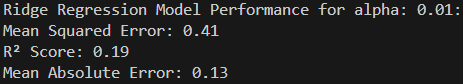
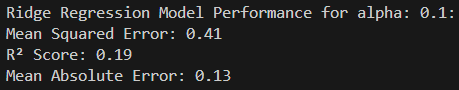
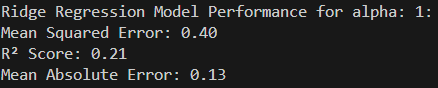
The ridge regression was done using the built it function for Lasso for different alphas,”0.01,0.1,1,10,100”, and passed through three performance metrics and the results shown in the figures below:

Figure :Ridge for alpha 1

Figure :Ridge for alpha 0.1

Figure :Ridge for alpha 0.01

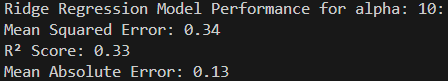
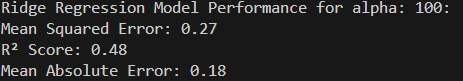


Figure :Ridge for alpha 100

Figure :Ridge for alpha 10

Then by using the belt in function for grid search to find the optimal alpha for the lasso and ridge regression models for lasso it was 0.01, for ridge it was 10 as shown the output in the figure below:



Figure :optimal alpha for lasso and ridge

## Closed Form Solution:

The closed form aims to find the weight vector W without using gradient descent, it was done using the closed form equation:

y

By using “np.linalg.pinv” from NumPy library to make sure of the numerical stability.

Then the error was evaluated using three performance metrics, mean square error, R2, mean Absolute Error the results in the figure below:

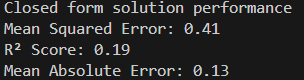


Figure :closed form solution performance

## Nonlinear Models:

### Polynomial Regression:

It extends linear regression by adding polynomial features to the input data, allowing the model to fit non-linear relationships.

We chose the polynomial degree to be “2”

poly\_regression learns the coefficients of the polynomial regression equation:

Then the prediction for the validation labels based on the polynomial features of the validation set, after this the metric performance MSE, R2, MAE and the results in the figure below:

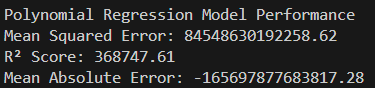


Figure :Polynomial Regression performance

### Radial Basis Function (RBF):

Using the standard scaler built in function to standardize the features “mean = 0, standard deviation = 1” to ensure all features contribute equally to the model.

The Kernal RBF to standardize non linear functions c is the Regularization parameter, and gamma defines the influence of a single training example, the labels prediction for the validation set, finally the performance and the results shown in the figure below:

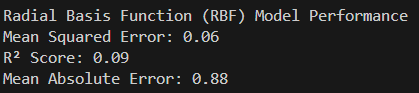


Figure :Radial Basis Function performance

# . Model Selection Using Validation Set:

Int this part, the aim was to find the best model depending on its performance, from each model of the previous models the best model is the model with highest R2 and lowest MSE, it was the radial basis function, the result shown in the figure below:

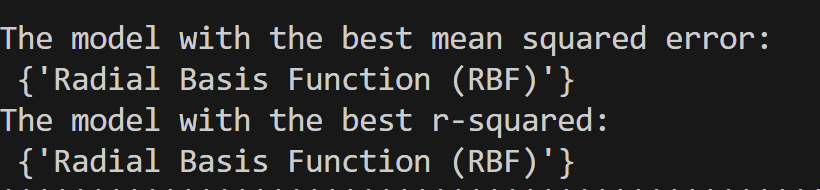


Figure :best model performance

# Feature Selection with Forward Selection:

This was done using the forward selection method, by selecting different features, and find the mean square error for each of them, after selecting different features when the there is no enhancement we stop selecting:

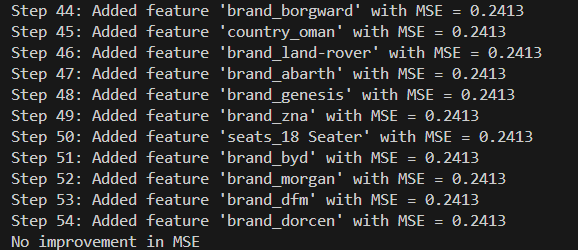


Figure :features selection

As shown in the previous figure after 54 features the improvements stopped.

# Hyperparameter Tuning with Grid Search:

This part was done before and the results are shown in the previous fiquers , figure 14.

# References

[1]: <https://www.kaggle.com/datasets/ahmedwaelnasef/cars-dataset/data>

[2]: <https://mocki.io/>

[3]: <https://mocki.io/v1/6b55e2fc-4bfd-4e13-9589-30636717e6ce>

[4]: <https://mocki.io/v1/24326926-978b-4c04-a7f8-d79022e96d6f>