CS464 Project Final Report - Group 22

Project: Forecasting Second-Hand Car Market Prices

28.12.2022

Simge Çınar 21901465

Necati Furkan Çolak 21803512

Emre Can Şen 21902516

Serkan Uygun 21902916

1. Introduction

In today's industry, regression models are used in forecasting, time series, and finding the cause-and-effect relationship between variables [1]. In general terms, it's used to predict the output of the model. In that way, companies can manage their long-term investments and plans efficiently since they can predict long-term issues with regression analyses. Regression models are also used for detecting the actual prices of assets, these assets can be cars, houses, etc. As it is done in business, this project also focuses on regression models to predict second-hand car prices.

The machine learning methods for asset prediction can be divided into 4 groups. Firstly, linear regression is one of the most fundamental parts of asset prediction algorithms. This model takes related features like age, quality, brand, etc., and gives the price as an output. However, there are some problems related to linear regression, which is that linear regression always assumes that data is in the linear form [2]. However, this may not happen since, in real-life practices, price data can move in a way other than linear. In addition to linear regression, lasso, ridge, and elastic net regression are also common to predict prices more precisely. Secondly, regression with the random forest is used in the industry also. This regression type gives the same output for determined intervals. With this method, the price distribution of the dataset can be seen easily [3]. The problem with this method is allocating prices for each unique data point since algorithms work with intervals. The third approach is support vector regression, this algorithm is used for both classification and regression. SVR finds the points which have the least error rate and are within the decision boundary. The advantage of SVR is that it acknowledges the presence of non-linearity in the data and provides a proficient prediction model [4].

Last technique that we use is a neural network with ReLU. This technique may offer higher performance than the other three techniques. Neural networks take the parameters as input, then the input will be investigated in a hidden layer with the rectified linear unit method. Lastly, the price will be given as an output.

2. Data Analysis & Preprocessing

The original data has 8127 rows, 13 columns and there are 222 rows that contain NaN values. The cars are sold between 1983 and 2020 and the number of seats varies between 2 and 14. The minimum selling price is 29,999, maximum is 10,000,000 and the mean is 638,294.9739. The histogram of the selling price can be seen below.

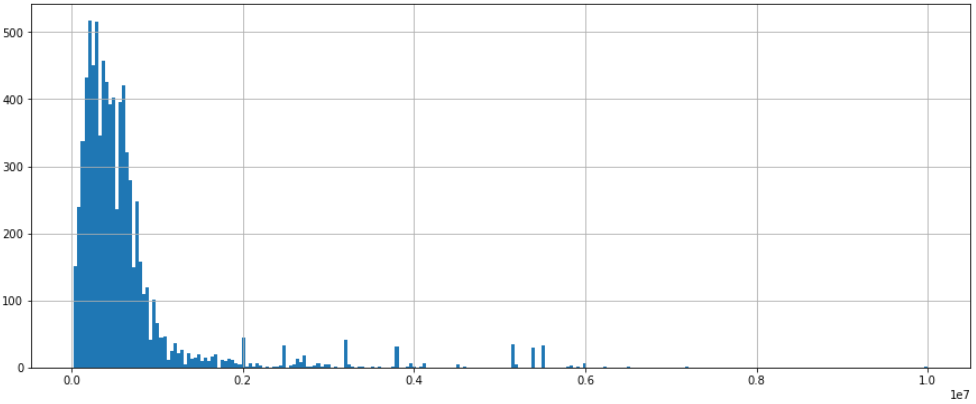

Figure 1: Histogram of the *selling_price* column

It can be observed that the selling price is right-skewed. There are only 2369 price values which are above the average. Boxplot of the selling price is as follows

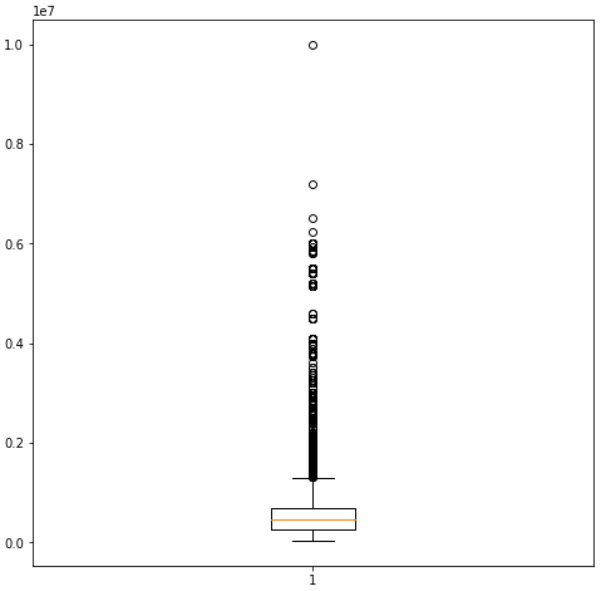

Figure 2: Boxplot of the *selling_price* column

It can be observed that there are a lot of outliers in the selling price column. To see which columns are correlated with the selling price, correlation matrix is examined. Correlation matrix for the numeric features can be seen below (rows that contain NaN values are extracted)

	year	selling_price	km_driven	mileage	engine	max_power	seats
year	1.00	0.41	-0.42	0.33	0.02	0.22	-0.01
selling_price	0.41	1.00	-0.23	-0.13	0.46	0.75	0.04
km_driven	-0.42	-0.23	1.00	-0.18	0.21	-0.04	0.23
mileage	0.33	-0.13	-0.18	1.00	-0.58	-0.37	-0.46
engine	0.02	0.46	0.21	-0.58	1.00	0.70	0.61
max_power	0.22	0.75	-0.04	-0.37	0.70	1.00	0.19
seats	-0.01	0.04	0.23	-0.46	0.61	0.19	1.00

Figure 3: Correlation matrix for numeric features

The table above illustrates that selling price has a positive correlation with max_power and negative correlation with km_driven. After examining data, the following preprocessing steps are applied.

- Non-numeric values are extracted from columns *engine*, *max_power* and *mileage*.
- *Torque* column is removed since it doesn't provide any information
- Brands of the cars are determined from the *name* column. The brands of the cars are added as a new column called "*brand"* and the "*name"* column is removed
- One Hot Encoding is applied to "*seller_type"*, "*transmission"*, "*owner"*, "*fuel"*, "*brand*" columns since they contain categorical values. After One Hot Encoding, the column number increased from 12 to 53.
- The rows that contain NaN values are removed.
- Data is splitted into training, validation and test sets.
- The columns apart from "*seller_price"* are standardized using StandardScaler
- PCA is applied with 95% explained variance.

Three standardization methods were used before applying PCA which are StandardizedScaler, MinMaxScaler and RobustScaler. RobustScaler is tried since it is a good method for the data which contains a lot of outliers. It gave the number of components as 13 for PCA with 95% explained variance. Since the original data has 13 features, we decided to eliminate RobustScaler since it might overfit the data. The graph for explained variance in RobustScaler can be seen below and code can be seen in the file 'RobustScaled_Model.ipynb'.

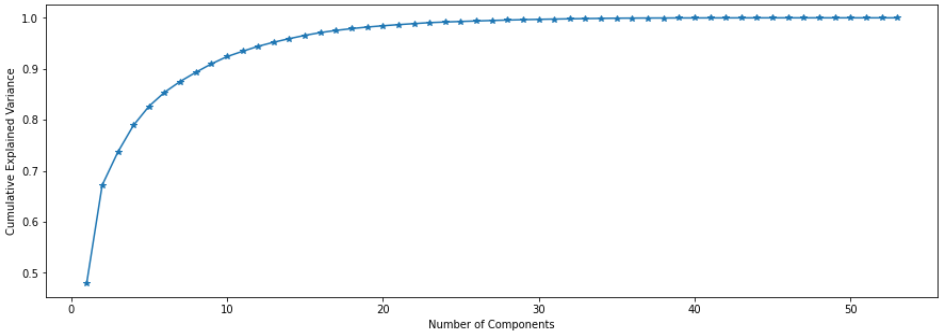

Figure 4: Cumulative explained variance with RobustScaler

The number of components for StandardizedScaler and MinMaxScaler are 39 and 19 respectively. It was observed that StandardizedScaler performed better on the validation data, while MinMaxScaler gave better results on test data for first four models. This situation might originate from how the data is splitted. The graphs for cumulative explained variance can be seen below for both standardization methods. Even though both standardization methods gave similar results, we choose standardization instead of normalization since it captures information about outliers [5]. The code of the MinMaxScaler can be seen in the file 'MinMaxScaled_Model.ipynb'

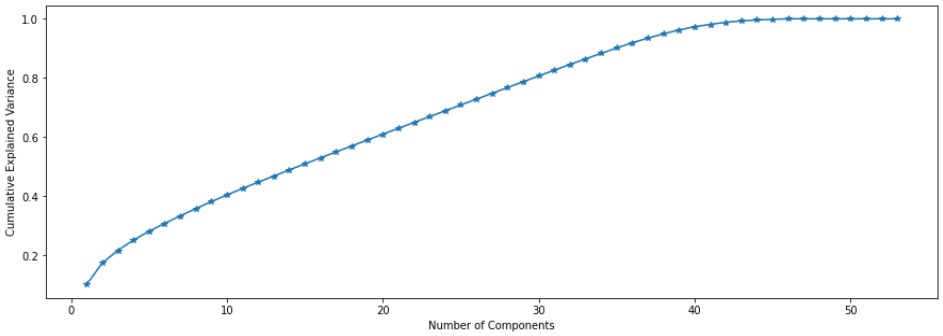

Figure 5: Cumulative explained variance with StandardizedScaler

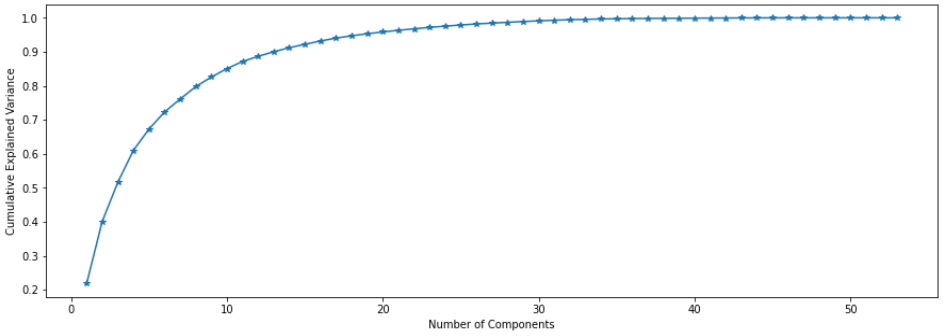

Figure 6: Cumulative explained variance with MinMaxScaler

3. Model Analysis & Output Performances

3.1. Linear Regression

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} (y_i - \sum_{j=0}^{p} w_j * x_{ij})^2 \quad \text{(Eqn. 1)}$$

Firstly, linear regression algorithm is utilized to generate the first model. Since linear regression models are commonly used in price prediction and are the simplest, this algorithm was chosen as the starting point. The above equation 1 is the cost function of linear regression, and the aim is to minimize this cost while mapping inputs with coefficients to outputs. Sklearn library is used for generating these coefficients so that the cost function is minimum.

The metric scores calculated from the validation data are as follows:

$$RMSE = 296,542.0347$$

$$MAPE = 0.5023$$

$$R^2\ Score = 0.8553$$

The metric scores calculated from the test data are as follows:

$$RMSE = 279,506.6463$$

$$MAPE = 0.4323$$

$$R^2\ Score = 0.8748$$

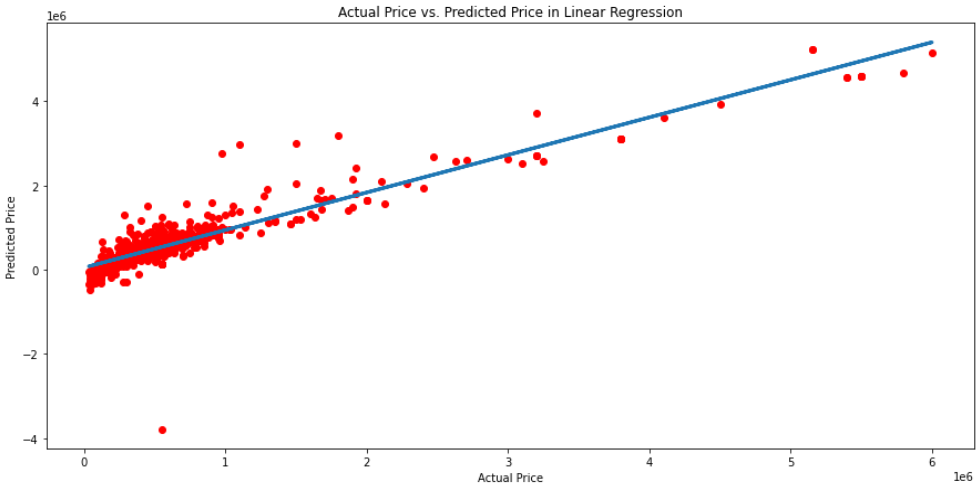

Figure 7: Predicted vs. Actual Prices: Linear Regression

It can be observed that there are negative values in the predicted prices. The number of negative price values are 60 and 98 in the validation and test data respectively. These negative values originated from the outliers in the selling price but overall it can be said that model is not bad by looking at the R^2 value which is 0.8553 on the validation data. 200 samples were taken from the test data and its actual and predicted price are illustrated to show how the predictions fit the model. The graph can be seen below

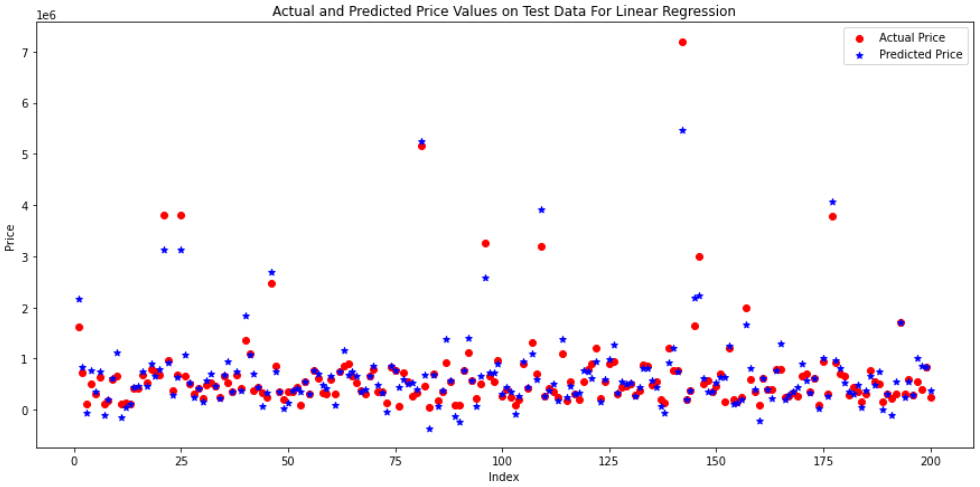

Figure 8: Scatter plot of predicted and. actual prices: Linear Regression (random 200 samples)

3.2. Ridge Regression

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}(y_i - \sum_{j=0}^{p} w_j * x_{ij})^2 + \lambda \sum_{j=0}^{p} w_j^2 \quad \text{(Eqn. 2)}$$

In order to have a more uniform model distribution, ridge regression algorithm is applied. This method works best when most of the variables in the model are useful. The penalty term in the equation shrinks all of the parameters with their correlated variables. So all of the variables contribute similarly to the outcome. Sklearn library was used for generating this model. To find the optimal value of alpha to use on the model, the RidgeCV module was used. This method iterates over the given range with a step size of 0.01 and determines the best alpha value to use in the model. Our alpha value was 0.99.

The metric scores calculated from the validation data are as follows:

$$RMSE = 296,533.3638$$

$$MAPE = 0.5022$$

$$R^2\ Score = 0.8553$$

The metric scores calculated from the test data are as follows:

$$RMSE = 279,502.7138$$

$$MAPE = 0.4322$$

$$R^2\ Score = 0.8748$$

These scores on the validation data are interpreted as a good result for a non-complex ridge regression model. The score of this model is better than linear regression.

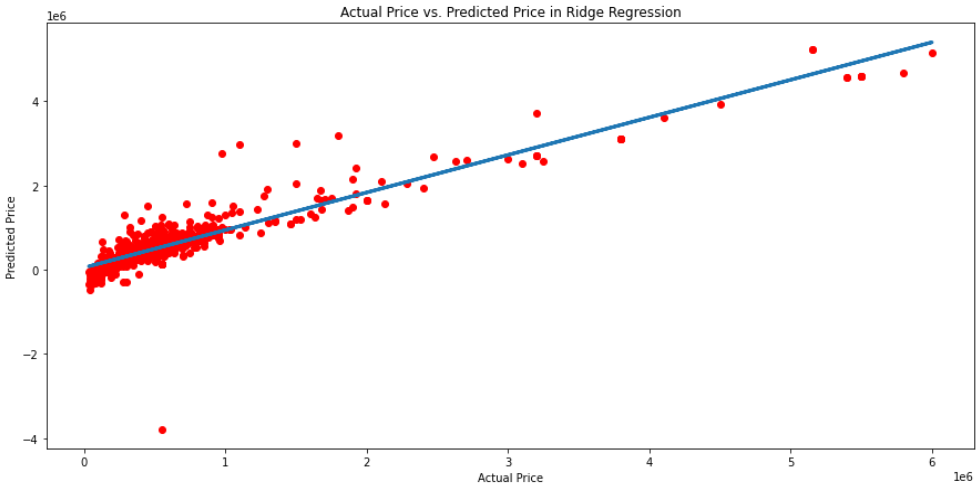

Figure 9: Predicted vs. Actual Prices: Ridge Regression

From the above graph, it can be seen there are still negative price values in the predictions. 200 samples were taken from the test data and its actual and predicted price are illustrated to show how the predictions fit the model. The graph can be seen below

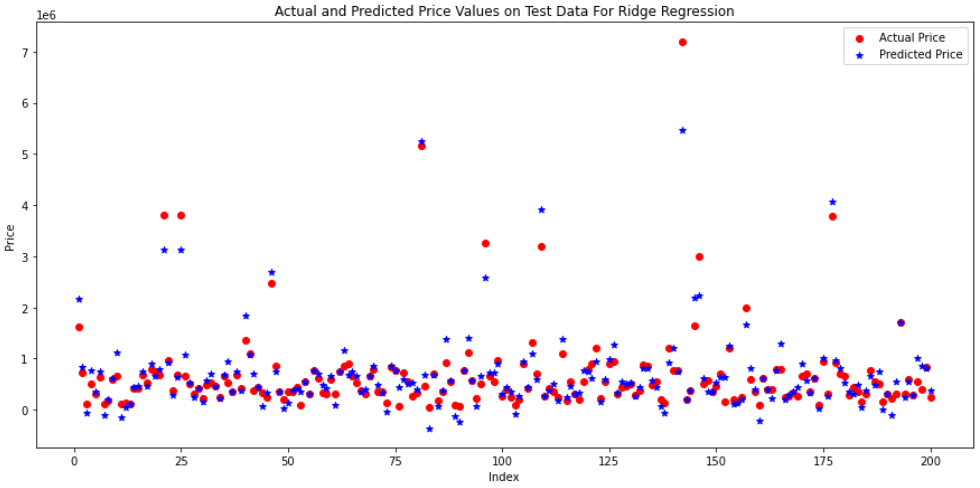

Figure 10: Predicted vs. Actual Prices: Ridge Regression (random 200 samples)

3.3. Lasso Regression

$$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{M}(y_i - \sum_{j=0}^{p} w_j * x_{ij})^2 + \lambda \sum_{j=0}^{p} |w_j| \quad \text{(Eqn. 3)}$$

In order to have a more uniform model distribution and to also eliminate some classes completely, the lasso regression algorithm is applied. This method works best when there are a lot of useless variables in the model. Lasso regression penalty term helps the function pick one of the correlated terms and eliminate the others. So the useless variables are dropped, and only the useful variables contribute to the output. Implementation of this model was done with the Sklearn library. To find the optimal value of alpha to use on the model, the LassoCV module was used. This method iterates over the given range with a step size of 0.01 and determines the best alpha value to use in the model. Our alpha value was 0.99.

The metric scores calculated from the validation data are as follows:

$$RMSE = 296,541.2904$$

$$MAPE = 0.5023$$

$$R^2\ Score = 0.8553$$

The metric scores calculated from the test data are as follows:

$$RMSE = 279,506.3466$$

$$MAPE = 0.4323$$

$$R^2\ Score = 0.8748$$

These scores on the validation data are acceptable parameters for a non-complex lasso regression model. Lasso regression gave better results compared to linear regression and slightly worse results compared to ridge regression

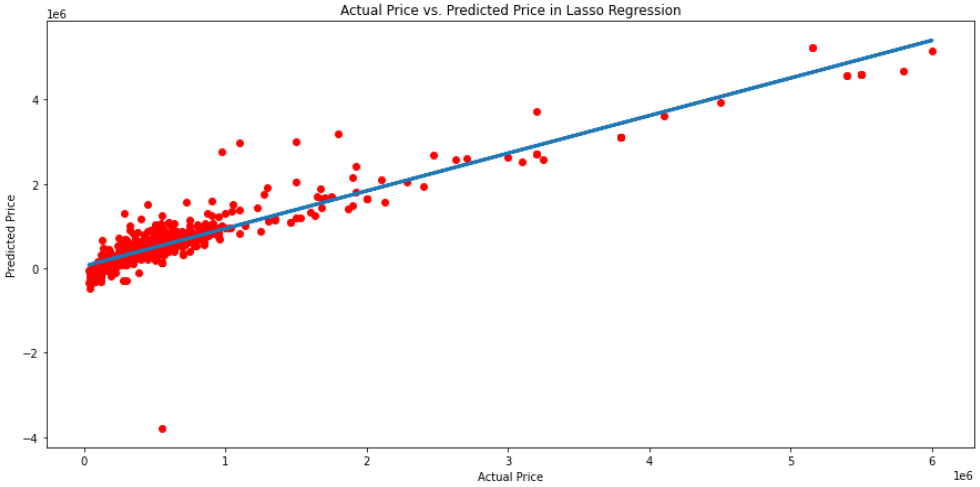

Figure 11: Predicted vs. Actual Prices: Lasso Regression

From the above graph, it can be seen there are still negative price values in the predictions. 200 samples were taken from the test data and its actual and predicted price are illustrated to show how the predictions fit the model. The graph can be seen below

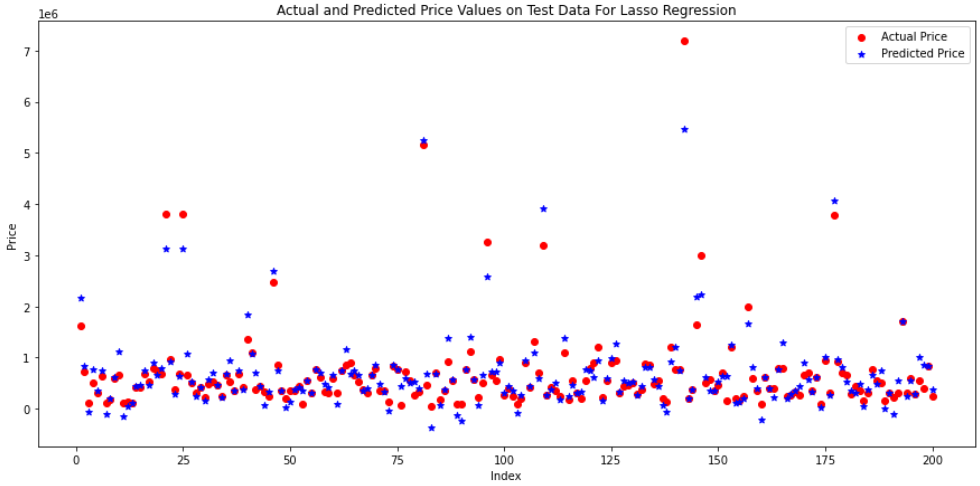

Figure 12: Predicted vs. Actual Prices: Lasso Regression (random 200 samples)

3.4. ElasticNET Regression

It was decided that ridge regression should be used when most of the variables in the model were useful and lasso regression when there were a lot of useless variables. However, when the variable counts start to ramp up, it becomes harder to ascertain which variables are useful and which are useless. So elastic net regression is used in order to combine the best parts of lasso and ridge regressions [6]. It combines ridge and lasso regression penalty terms and balances them optimally. If the variable is useless, the lasso term will be high, and the ridge term will be low. If it is useful, the lasso term will be low, and the ridge term will be high. Or the relation will be a combination of two parameters, allowing for fine-tuning of variables. The best combination is found by cross-validation. Sklearn library was used for this model. To find the optimal value of alpha to use on the model, the ElasticNetCV module was used. This method iterates over the given range with a step size of 0.01 and determines the best alpha value to use in the model. Our alpha value was 0.99.

The metric scores calculated from the validation data are as follows:

$$RMSE = 293,483.0294$$

$$MAPE = 0.4653$$

$$R^2\ Score = 0.8583$$

The metric scores calculated from the test data are as follows:

$$RMSE = 280,214.2995$$

$$MAPE = 0.4072$$

$$R^2\ Score = 0.8742$$

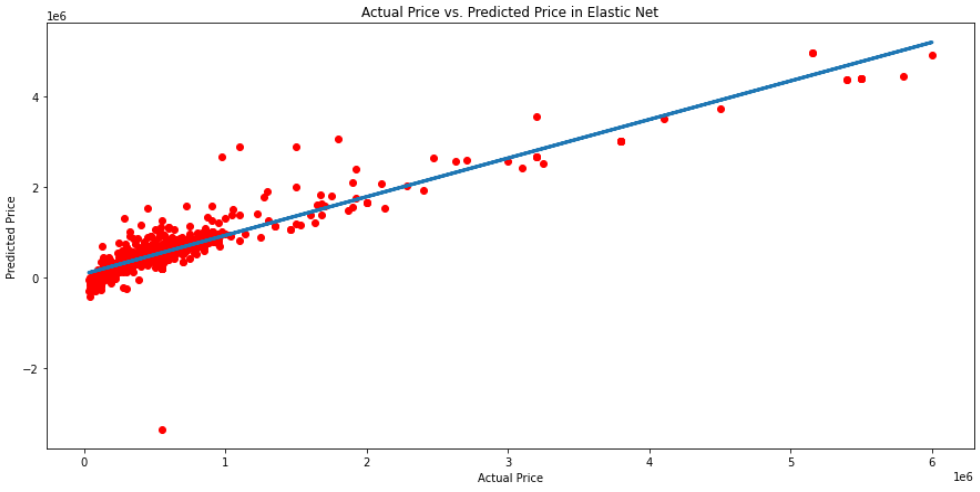

Figure 13: Predicted vs. Actual Prices: ElasticNet

From the above graph, it can be seen there are still negative price values in the predictions. 200 samples were taken from the test data and its actual and predicted price are illustrated to show how the predictions fit the model. The graph can be seen below

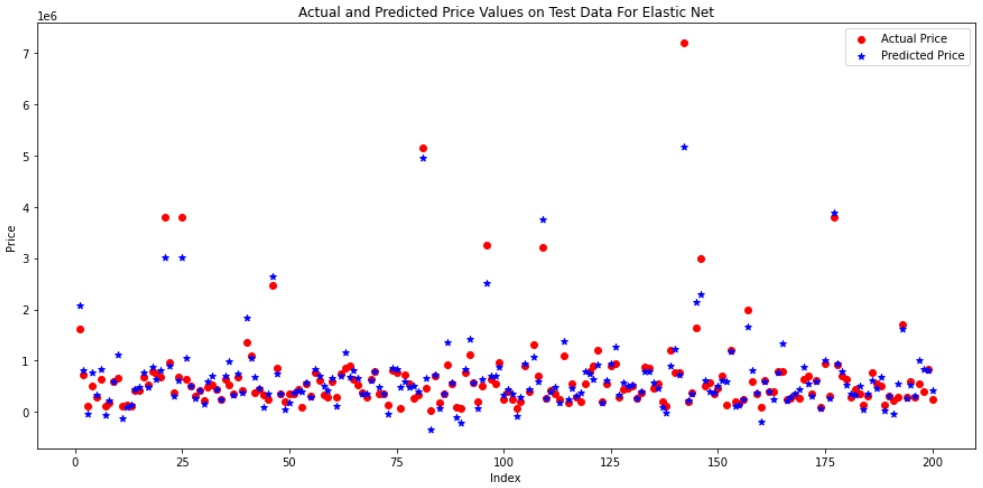

Figure 14: Predicted vs. Actual Prices: Elastic Net (random 200 samples)

Overall, it can be said that elastic net gives better results than linear, ridge and lasso regression.

3.5. Random Forest Regression

Random forest regression algorithm can be applied to both classification and regression problems. The main purpose of the algorithm is creating different decision trees to predict the label in our case it's price of the car. The reason why we use random forest regression is to prevent overfitting over the model since decision tree algorithms captures most of the variance. Therefore, the biggest problem of the classical decision tree algorithms is that models overfit too much [7]. However, random forest regression has a capability to reduce variance unlike other decision tree algorithms. To construct our model, we used sklearn library. In the graph below, the chosen 200 samples were taken from the data, to make graph more clear.

The metric scores calculated from the validation data are as follows:

$$RMSE = 158,744.1925$$

$$MAPE = 0.1811$$

$$R^2\ Score = 0.9581$$

The metric scores calculated from the test data are as follows:

$$RMSE = 214,339.4650$$

$$MAPE = 0.1698$$

$$R^2\ Score = 0.9146$$

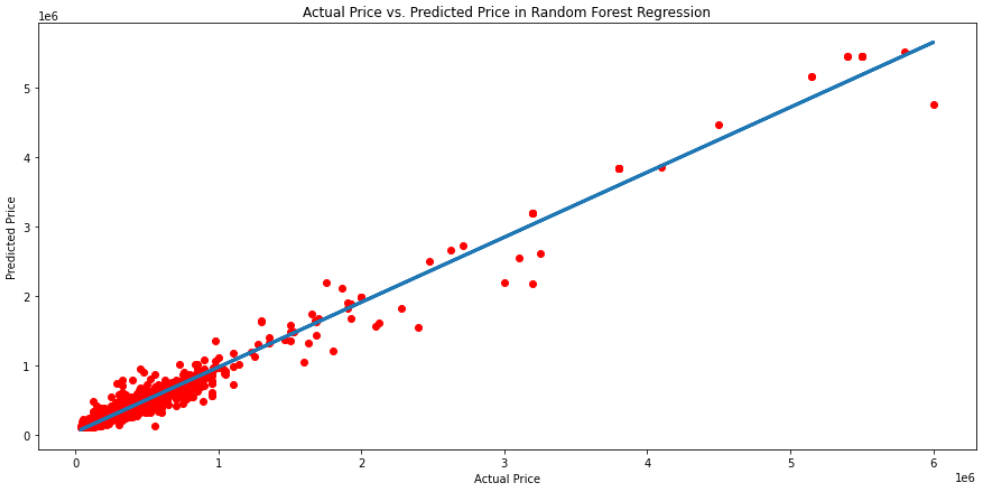

Figure 15: Predicted vs. Actual Prices: Random Forest Regression

200 samples were taken from the test data and its actual and predicted price are illustrated to show how the predictions fit the model. The graph can be seen below

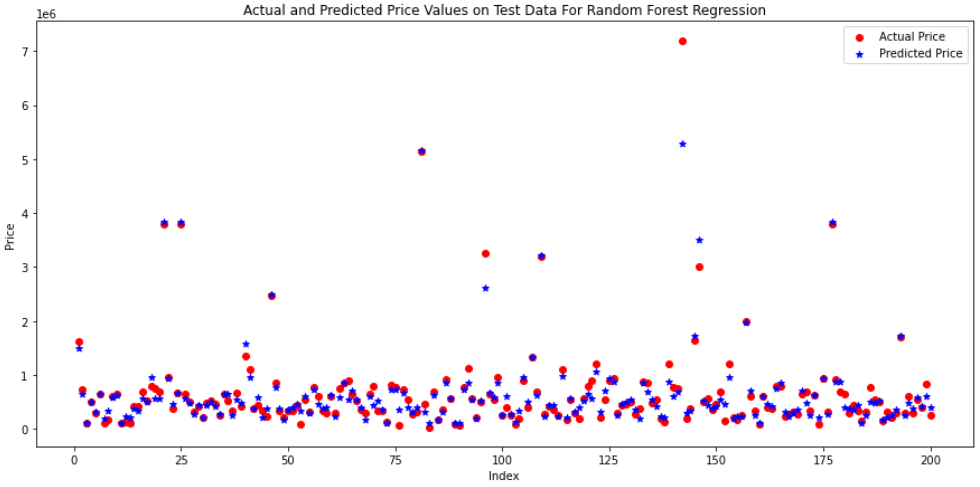

Figure 16: Predicted vs. Actual Prices: Random Forest Regression (random 200 samples)

It can be seen that the model could not capture some outliers. The related figures ,Figure 15 & Figure 16, can be found above. Random Forest Regression gives better results than linear, lasso, ridge and elastic net regression.

3.6. Support Vector Regression

Even though support vectors are generally used in classification problems, it can also be used in regression by finding the best line which is a hyperplane that has the maximum number of points [8]. Support vector regression is good at capturing the nonlinearity in the data.

The metric scores calculated from the validation data are as follows:

$$RMSE = 107,728.4356$$

$$MAPE = 0.1580$$

$$R^2\ Score = 0.9809$$

The metric scores calculated from the test data are as follows:

$$RMSE = 215,669.4807$$

$$MAPE = 0.1499$$

$$R^2\ Score = 0.9255$$

Support Vector Regression gives much better results compared to other regression models and its R^2 value is really high on the validation set.

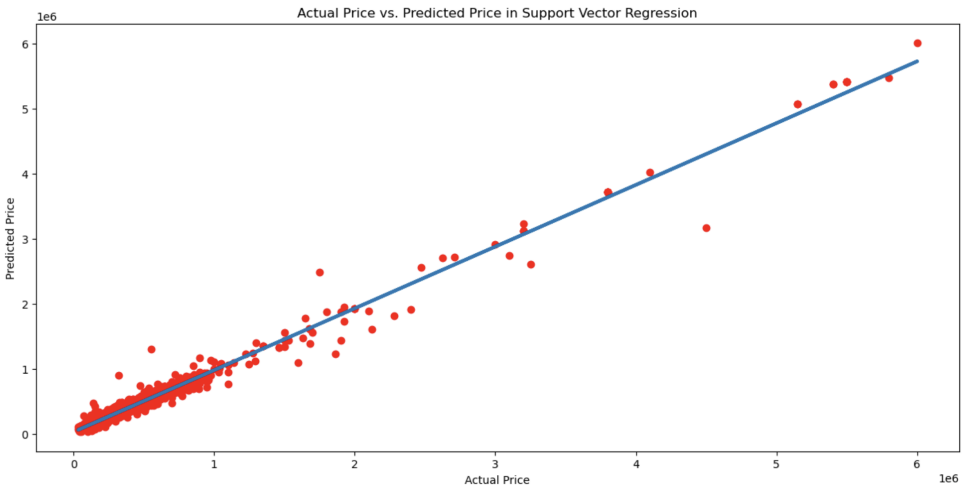

Figure 17: Predicted vs. Actual Prices: Support Vector Regression

200 random points are selected from the test data to show how the predictions match the actual prices. It can be seen that the model could not capture some outliers.

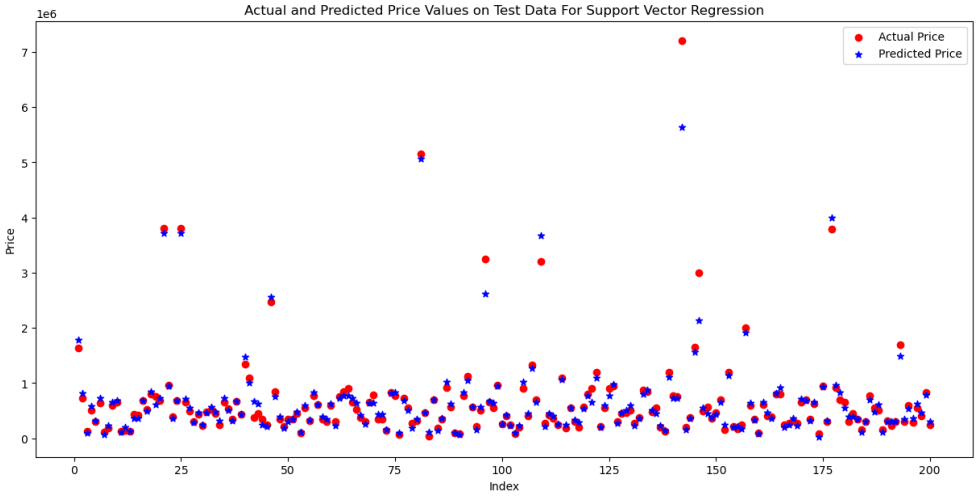

Figure 18: Predicted vs. Actual Prices: Support Vector Regression (random 200 samples)

3.7. Neural Networks

To determine the initial neural network structure, papers and web articles with similar topics and datasets to our case were researched and examined [9], [10]. Some common practices from these findings were extracted and were used as a starting point. It was seen that either two or three hidden layers were sufficient for price prediction algorithms with similar dimensions as our dataset. Also, the hidden layer sizes were chosen in the form of 2^n in these articles, and this number was between the size of input and output layers [11]. Initially, two hidden layers with sizes chosen from the combination of {2, 4, 8, 16, 32, 64} were experimented with. Adam optimizer was used, and these first models were trained with 150 epochs. Here are the best results for two hidden layers:

Hidden Layer Sizes	**Val R^2**	**Test R^2**
[16, 8]	0.9297	0.9124
[32, 16]	0.9386	0.9196

Table 1: R^2 Values for Different Hidden Layer Sizes with 2 Hidden Layers

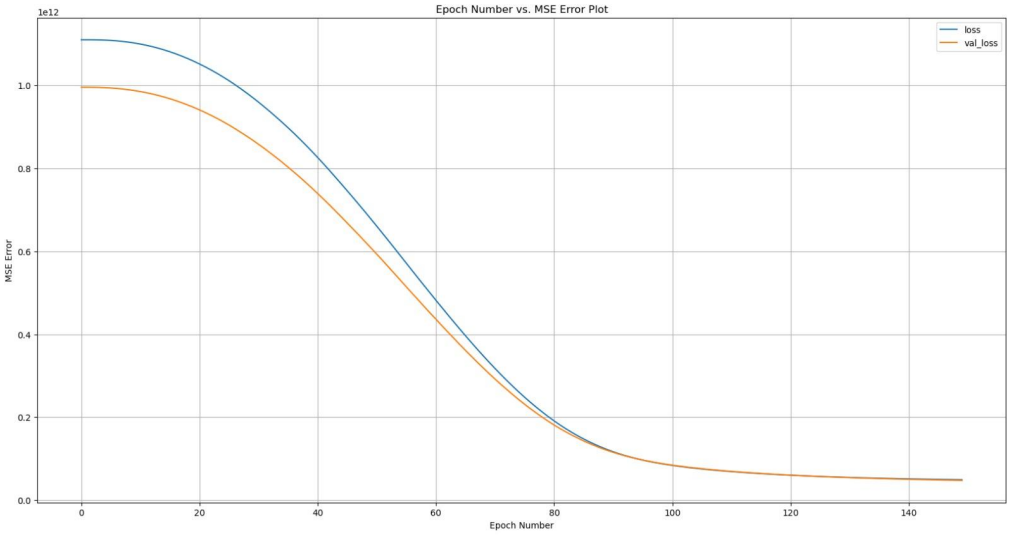

Figure 19: Epoch Number vs. MSE Error: [16, 8] Neural Network

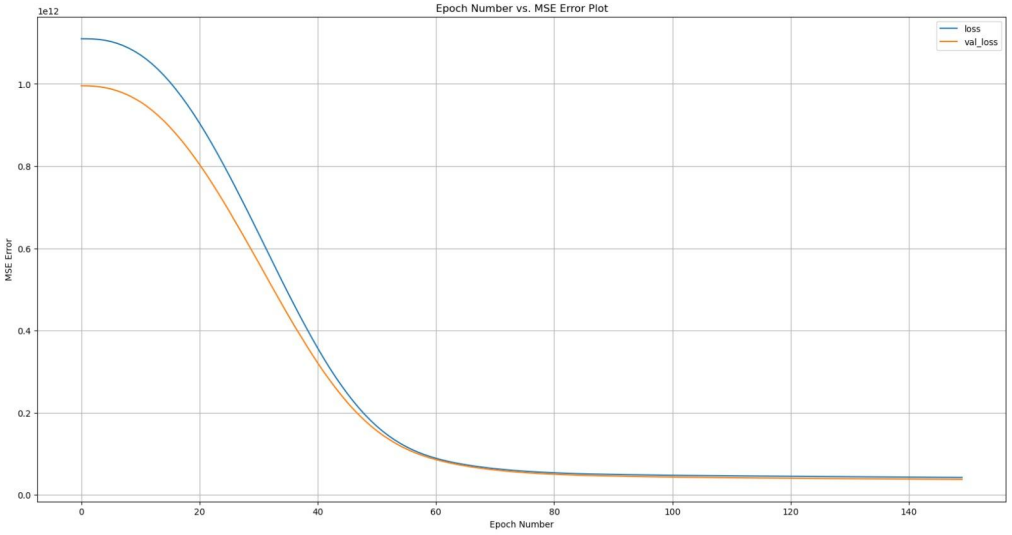

Figure 20: Epoch Number vs. MSE Error: [32, 16] Neural Network

Then the model was trained with three and four hidden layers, and their sizes were chosen from the combination of these numbers {2, 4, 8, 16, 32, 64, 128)}

Hidden Layer Sizes	**Val R^2**	**Test R^2**
[32, 16, 8]	0.9827	0.9689
[64, 32, 8]	0.9832	0.9747
[128, 64, 32, 16]	0.9954	0.9698

Table 2: R^2 Values for Different Hidden Layer Sizes with 3, 4 Hidden Layers

The best-performing neuron sizes were chosen as 32 for the first hidden layer, 16 for the second hidden layer and 8 for the third. As the next step, the learning rate and batch sizes were determined. For this step, the previously mentioned sources were taken as a starting point and experimenting from that point, the best result was found. Here are the error metrics for differing learning rates and batch sizes, 300 epochs were used for training:

Learning Rate	**Val R^2**	**Test R^2**
0.001	0.9827	0.9689
0.005	0.9935	0.9313
0.007	0.9975	0.9371
0.008	0.9953	0.9774
0.009	0.9953	0.9501
0.01	0.9955	0.9570

Table 3: R^2 Values for Different Learning Rates

We can see from the preceding figure that the model performs the best when the learning rate is chosen as 0.008. To experiment with the batch sizes, we have set the learning rate of the model as such. The model runs with the default batch size of 32. Therefore, we have tried the models with the batch sizes set as 16 and 64. The performance metrics of the trained model can be seen below:

Batch Size	**16**	**32**	**64**
Val R^2	0.9928	0.9953	0.9876
Test R^2	0.9465	0.9774	0.9482

Table 4: R^2 Values for Different Batch Sizes

Hence the best outcome was achieved with parameters as the following:

- Amount of Hidden Layers: 3
- Hidden Layer Sizes: [32, 16, 8]
- Learning Rate: 0.008
- Batch Size: 32

Furthermore, to ensure that our best model does not overfit, we have checked the Root Mean Squared Error on the train data and compared it to the validation data. We have seen that the RMSE when the model predicts with the train data does not exceed the RMSE when the model predicts the validation data. Therefore, we can say that the model does not overfit. The final performance metrics of the trained model can be seen below.

Metrics	**RMSE**	**MAPE**	R^2
Train	30,647.7551	0.0417	0.9986
Validation	71,474.5870	0.1291	0.9953
Test	13,1911.1992	0.0552	0.9774

Table 5: Performance metrics

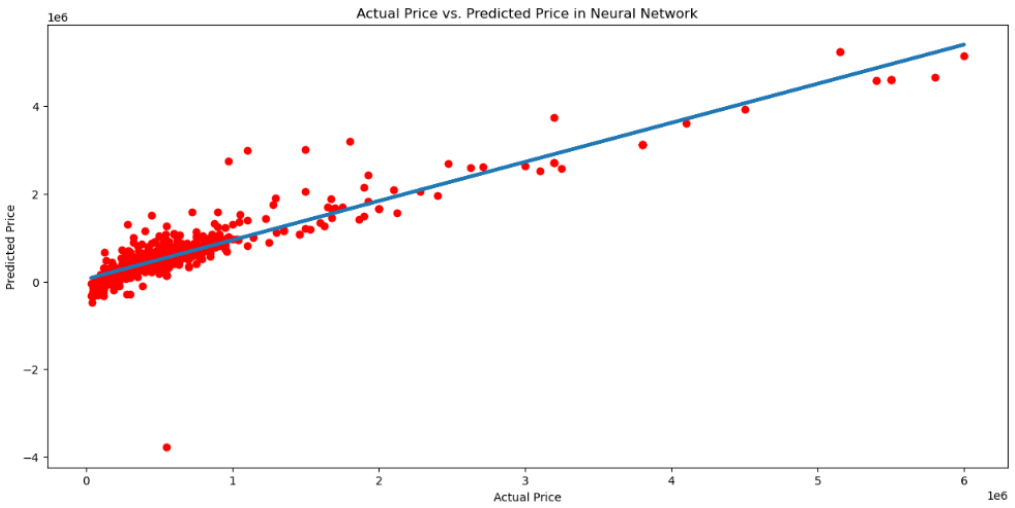

Figure 21: Actual Price vs Predicted Price in Neural Network

200 random points are selected from the test data to show how the predictions match the actual prices.

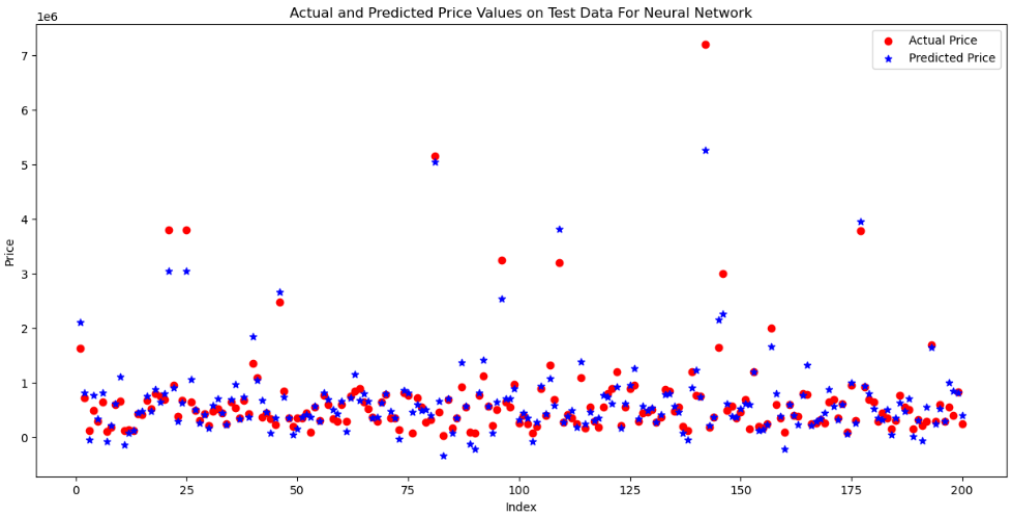

Figure 22: Actual Price vs Predicted Price in Neural Network (random 200 samples)

4. Conclusion

It was seen that linear regression was able to deal with the majority of the data points with a decent amount of accuracy. However, some outliers in the dataset did not fit the linear case and caused significant error factors. Therefore, some other methods were implemented for better results while dealing with these outliers. After the ridge and lasso regression models were applied to the dataset, it was seen that the introduced penalty terms were not able to diminish the effect of these outlier points as these points were comparably grand in magnitude to other points. The ElasticNet approach picked the best values of both the ridge and lasso penalty terms, but the improvement was minimal. Thus, it was evident that more complex models were needed for better results. Random forest regression is another way to predict prices. In this regression type, we do not observe any negative prices unlike linear, ridge, lasso and Elastic Net. Moreover, there were not any negative price values in the support vector regression either. Both random forest regression and support vector regression were more successful capturing the outliers in the data compared to other regression methods.

Finally, a neural network structure with three hidden layers and hyperparameters that were determined via experimentation and online research was implemented. The best model was achieved with the trade-off of a more complex model and more runtimes. The neural network was able to deal with both the outliers and general cases. Overall it can be said that the neural network gave the best R^2 in both validation data and test data.

5. Work Package and Distribution of Tasks

Data preprocessing, PCA and linear regression were done by Simge Çınar and Necati Furkan Çolak. Emre Can Şen and Serkan Uygun did the ridge, lasso regression and elastic net. All group members worked on the first progress report. After getting feedback from the TA, support vector regression was assigned to Simge, random forest regression to Necati, and the neural network to Serkan & Emre Can. After each member implemented their models, everyone handled their part in the slides and in the final report.

6. References

[1] Beers, Brian. "What Is Regression? Definition, Calculation, and Example." *Investopedia*, Investopedia, 9 Sept. 2022, https://www.investopedia.com/terms/r/regression.asp.

[2] Singh, Naman. "Advantages and Disadvantages of Linear Regression." *OpenGenus IQ: Computing Expertise & Legacy*, OpenGenus IQ: Computing Expertise & Legacy, 27 July 2020, https://iq.opengenus.org/advantages-and-disadvantages-of-linear-regression/.

[3] Reader, The Click. "Random Forest Regression Explained with Implementation in Python." *Medium*, Medium, 2 Dec. 2021, https://medium.com/@theclickreader/random-forest-regression-explained-with-implementation-in-python-3dad88caf165. [Accessed: 27-Dec-2022].

[4] Sethi, Alakh. "Support Vector Regression in Machine Learning." *Analytics Vidhya*, 2 Dec. 2022, https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/. [Accessed: 27-Dec-2022].

[5] Kumar, Ajitech, "MinMaxScaler vs StandardScaler – Python Examples" *Data Analytics*, 27-July-2020. [Online]. Available: https://vitalflux.com/minmaxscaler-standardscaler-python-examples/. [Accessed: 27-Dec-2022].

[6] Kargın, Kerem. "ElasticNet Regression Fundamentals and Modeling in Python." *Medium*, MLearning.ai, 2 May 2021, https://medium.com/mlearning-ai/elasticnet-regression-fundamentals-and-modeling-in-python-8668f3c2e39e.

[7] Bakshi, Chaya. "Random Forest Regression." *Medium*, 9 June 2020, https://levelup.gitconnected.com/random-forest-regression-209c0f354c84. [Accessed: 27-Dec-2022].

[8] Raj, Ashwin, "Unlocking the True Power of Support Vector Regression" *Medium*, 3-Oct-2020. [Online]. Available:

https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0. [Accessed: 27-Dec-2022].

[9] K. Kumari, “Car price prediction - machine learning vs Deep Learning,” *Analytics Vidhya*, 25-Jul-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/07/car-price-prediction-machine-learning-vs-deep-learning/#h2_5. [Accessed: 26-Dec-2022].

[10] Lamwilton, “Most concise neural network car price prediction,” *Kaggle*, 04-Aug-2020. [Online]. Available: https://www.kaggle.com/code/lamwilton/most-concise-neural-network-car-price-prediction. [Accessed: 26-Dec-2022].

[11] “How to choose the number of hidden layers and nodes in a feedforward neural network?,” *Cross Validated*, 01-Aug-1957. [Online]. Available: https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw. [Accessed: 26-Dec-2022].