Shahid Beheshti University

Faculty of Mathematical Science

Department of Computer Science

Fundamentals of Machine Learning– Spring 2022 – Assignment 2

By:

Arman Davoodi

Abstract

This assignment is consisted of two parts. The first part concentrates on different methods of feature selection as well as some feature engineering approaches. The second part is about implementing simple linear regression using the numpy package.

Dataset

The datasets referred in this report are obtained from the Kaggle and are the same datasets used in the previous assignment. The link for the dataset used in the first part is provided here and this is the link to the dataset used in the second part. More details about the datasets, the preprocessing done on them and their hyper parameter tuning is provided in previous assignment reports as most of preprocesses and tuning approaches are the same.

Part One

In this part, three methods of feature selection are used on the dataset. Approaches are forward selection, backward selection and PCA respectively. The forward selection uses a threshold of 0.01 as a stopping criterion while backward selection and PCA have been used to reduce the features to the number features provided by the forward selection which is 5. Also, to compare different feature sets in forward and backward selection algorithms, logistic regression is used and then the AUC of using the model on the test dataset is computed.

Forward Selection

In this implementation, the train and test datasets in addition to the model are passed as arguments to the algorithm. Moreover, this method has three hyper parameters which are, min_features, max_features and threshold. The min_features parameter is the number of minimum features that the algorithm must return and is 0 by default. The max_features parameter is the maximum number of features that is returned by the algorithm and is the same as the total number of features by default. The threshold parameter is a positive float number and when the increase in AUC in the algorithm is less than this number and the number of selected features is at least equal to the min_features, the algorithm stops and returns the selected features. The default value for threshold is 0.01 in this implementation. In the end, this algorithm with default parameters and logistic regression as its model, returned five features which are ram, battery_power, px_width, px_height and mobile_wt respectively.

Backward Selection

In this implementation, the train and test datasets in addition to the model are passed as arguments to the algorithm. Moreover, this method has three hyper parameters which are, min_features, max_features and threshold which are the same as their counterpart for forward selection. However, the default value for threshold is 0 in this implementation. Since the forward selection gave us 5 features, the max_features parameter is set to 5. The result feature set was consisting of battery_power, mobile_wt, px_height, ram and sc_h.

PCA

Another, approach to feature selection in this assignment is by using PCA with n_components of 5.

Logistic Regression and Comparing different Feature Selection Approaches

The first step here was to tune the hyper parameters for the model and to choose which scaling should be applied to the dataset if any. The best results were achieved by not scaling the dataset at all and using the sklearn default parameters. The results for using this tuned model on different feature sets provided by different approaches is illustrated in the three tables below.

Table 1: Forward Selection

	Precision	Recall	F-Score
Class 0	0.99	1	0.9950
Class 1	1	0.9899	0.9949
Total	0.9949	0.9949	0.9949

Table 2: Backward Selection

	Precision	Recall	F-Score
Class 0	0.9840	0.9343	0.9585
Class 1	0.9375	0.9848	0.9606
Total	0.9596	0.9596	0.9596

Table 3: PCA

	Precision	Recall	F-Score
Class 0	0.99	1	0.9950
Class 1	1	0.9899	0.9949
Total	0.9949	0.9949	0.9949

As it can be seen, forward selection and PCA gave us the same results while backward selection performed a bit weaker in this example.

SVM and Feature Engineering

In this part, four different feature engineering approaches were taken in order to increase the model's performance.

The first approach was to bin the battery power. The binning is done in a way that all records with battery_power of lower than 1000 were labeled 0 while those with battery_power of at least 1000 and lower than 1500 were put in the second bin(were labeled 1) and the rest were put in the last bin which was 2.

Another important issue in most datasets is handling categorical features. One the most common used approaches for this is using one hot encoding. The two reasons this approach is mostly used with categorical features are that firstly, most models cannot handle categorical features and we need to convert them to numerical features and second, label encoding a feature is the same as assuming that the categories are in some order and the differences between the first category and the second category, is the same as the difference between the second and the third category. Due to these facts, when our data is categorical and not ordinal we have to use one hot encoding. In this dataset however, since the only categorical features are either binary or ordinal, this method is of little use.

Another important consideration when using a model, is to know the distribution of data and the behavior of the model on that distribution. As most of our models assume the data has normal distribution, it is necessary to change the distribution of our data. An example is when the distribution of our data is right skewed. In this case a popular approach is to use log transform on the dataset to change the distribution of our dataset. For this dataset, this transform is used on fc and px_height features which had skewed distributions.

Also, one way to increase the performance of the model is to add other features by combining primary features such as adding an area feature by using height and weight which was done for this dataset. However, sometimes this can lead to significant decrease in the models performance as the new feature may be too different compared with primary features. To avoid this sometimes we also need to standardize our dataset after adding the new feature.

The result of applying these methods on the dataset is illustrated in tables below.

Table 4: SVM without Feature Engineering

	Precision	Recall	F-Score
Class 0	0.975	0.9848	0.9799
Class 1	0.9847	0.9747	0.9797
Total	0.9798	0.9798	0.9798

Table 5: SVM with binning

	Precision	Recall	F-Score
Class 0	0.925	0.9343	0.9296
Class 1	0.9337	0.9242	0.9289
Total	0.9293	0.9293	0.9293

Table 6: SVM with One-Hot Encoding

	Precision	Recall	F-Score
Class 0	0.975	0.9848	0.9799
Class 1	0.9847	0.9747	0.9797
Total	0.9798	0.9798	0.9798

Table 7: SVM with Log Transform

	Precision	Recall	F-Score
Class 0	0.9648	0.9697	0.9672
Class 1	0.9695	0.9646	0.9671
Total	0.9671	0.9671	0.9671

Table 8: SVM with adding px_area and Standardizing

	Precision	Recall	F-Score
Class 0	0.9397	0.9444	0.9421
Class 1	0.9442	0.9394	0.9418
Total	0.9419	0.9419	0.9419

Table 9: SVM with all methods applied

	Precision	Recall	F-Score
Class 0	0.9579	0.9191	0.9381
Class 1	0.9223	0.9596	0.9406
Total	0.9394	0.9394	0.9394

As it can be seen, these methods mostly lowered the models performance in this specific example. But this does not mean that they are useless.

Resampling Methods

There are two main resampling methods called, bootstrapping and cross validation.

In bootstrapping we obtain distinct datasets by repeatedly sampling observations from the original dataset with replacement. And each of the obtained bootstrap datasets are created by sampling with replacement and is the same size as our original dataset.

In cross-validation, we divide our dataset into K roughly equal sized parts and then we iterate our parts K times, each time using 1 part as test and K-1 parts for train. Another method is 5 x 2 cross validation in which we do 5 cross validations with K equal to 2 and then we use the average of the results as the final result. This method can be used to obtain an approximate of the errors made by models and their variances. These data can be used for statistical tests and comparing the performance of different models.

Generally, cross validation is mostly used as mean to test the performance of our model while bootstrapping is mostly used to ensemble our data for models like random forest.

Part Two

In this part, simple linear regression is implemented using the numpy library. In this part only features, serviceCharge, heatingType and telekomUploadSpeed were used to train our models. The target of the training was the totalCost. Preprocesses done on the dataset are almost the same as the ones done in the previous assignment. Next, criterion functions for MSE and absolute error were implemented and used by the implemented model. The results were compared with the sklearn linear regression model and sklearn lasso and ridge regression models.

Implemented Linear Regression

The model takes 6 parameters when initialized which are the criterion function, gradient function, initial theta, learning rate, number of iterations and a parameter for printing the loss in some epochs.

By default, the criterion is MSE, learning rate is 0.01, initial theta is a vector of ones, and the number of iterations is 100. The model itself has two public functions fit and predict, and also a private function called forward.

In the predict function, a column of ones is inserted at the beginning of our predictors matrix as the bias term and then the dot product of the obtained matrix and theta which is the weight vector.

Just like the predict function, the fit function will insert a column of ones at the beginig of our predictor matrix and the uses the forward, and gradient functions in a loop to train the model and tune the theta vector.

The forward function does the same job as predict function except that it does not add any column at the start of the input matrix due to it being already added in the fit function.

Results

At the end, all of the models were trained on the same dataset and then they were tested on the same records. The RMSE of the predictions made by each model on the test dataset are demonstrated in *table 10*. In addition, to visualize the results for each model, the hexagonal plot of true values vs predicted values are plotted and provided bellow. For the implemented MSE model, 1000 was the number of iterations and learning rate was set as 0.00001. For the implemented absolute error however, learning rate was 0.01 but the number of iterations were 1000 like the MSE model.

Table 10

	Implemented Model(MSE)	Implemented Model(Absolute Error)	Sklearn (MSE)	Sklearn (Lasso)	Sklearn (Ridge)
RMSE	250.750	251.301	223.212	223.329	223.212

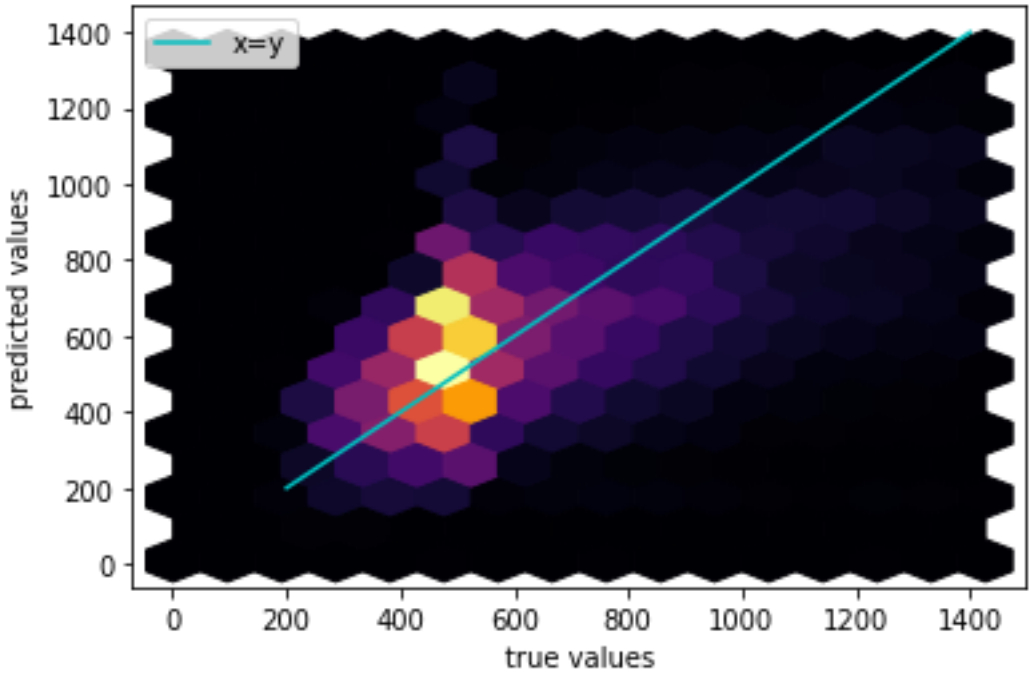

Figure 1: Implemented Model with MSE

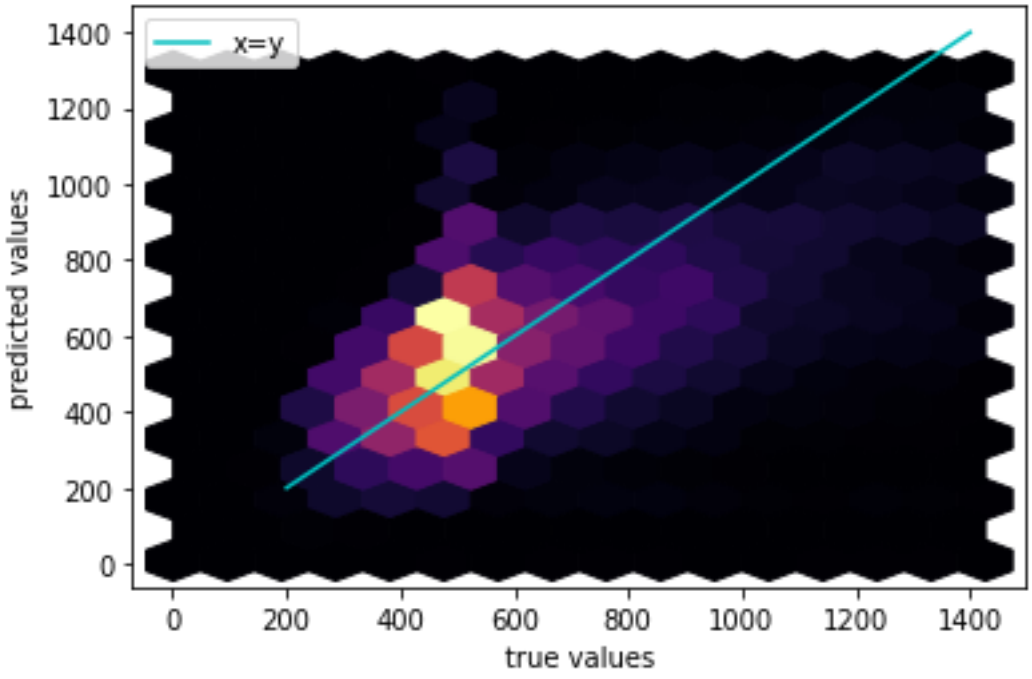

Figure 2: Implemented Model with Absolute Error

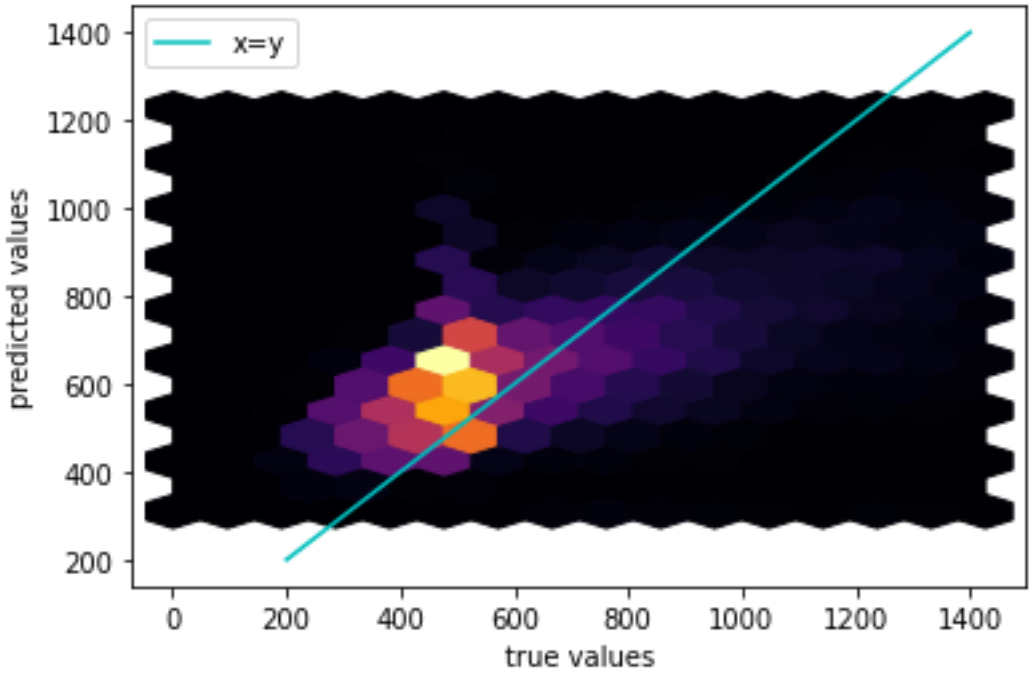

Figure 3: Sklearn Model with MSE

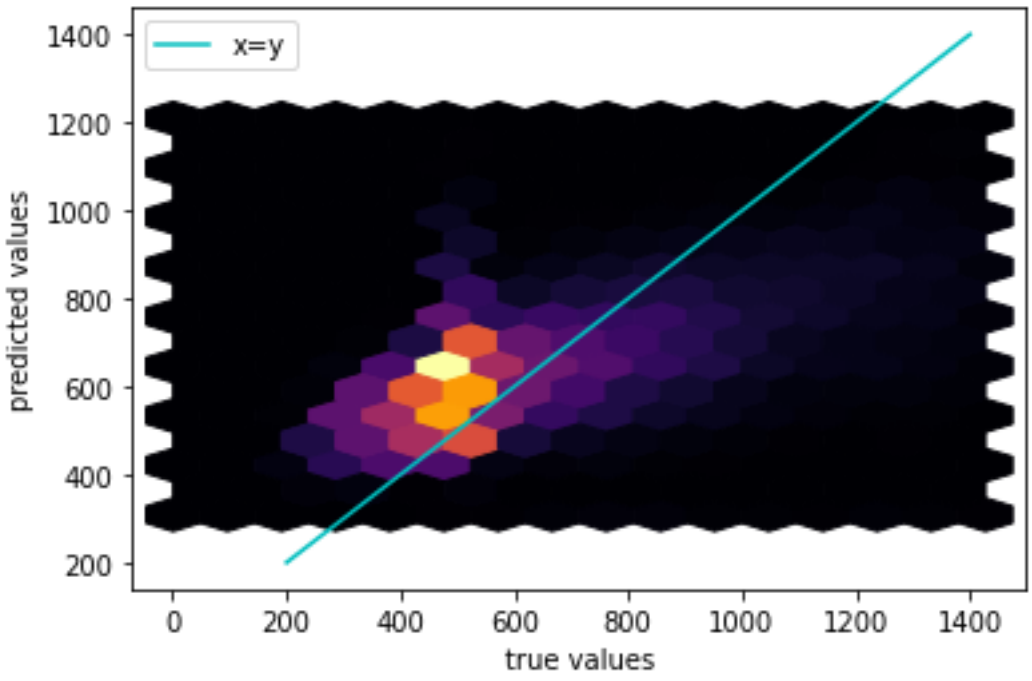

Figure 4: Sklearn Lasso Regression Model

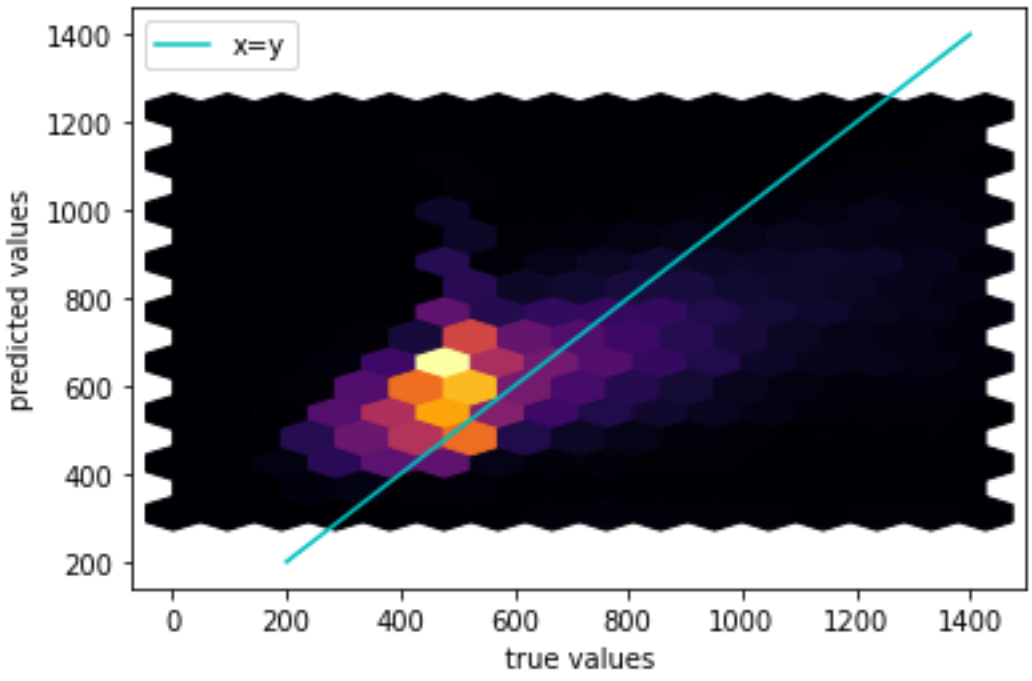

Figure 5: Sklearn Ridge Regression Model