Apartment rental offers in Germany

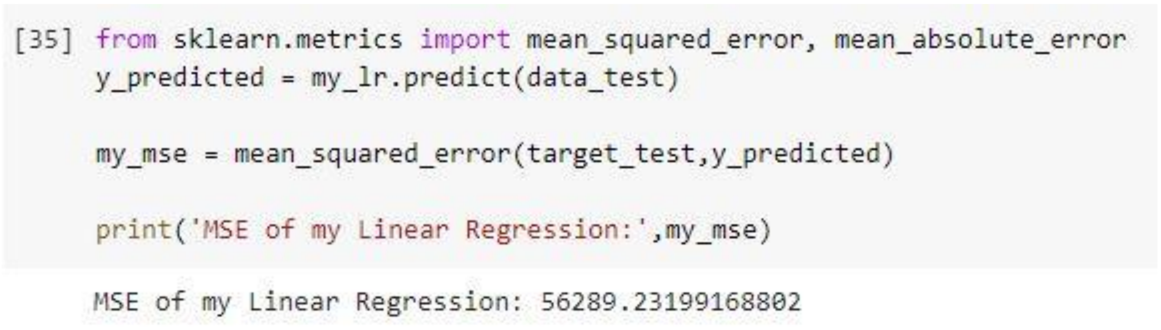
Saeid Cheshmi – 98222027

1. Data Exploration, Preprocessing, EDA

We did all these steps in last assignment, so we don’t mention them here again.

2. Implementing Linear Regression

We implemented linear regression with mean squared error as cost function and gradient descent as optimization algorithm. We will train our model with 3 specific features, ‘serviceCharge’, ‘heatingType’ and ‘telekomUploadSpeed’. Then, we will compare MSE of our model with sklearn one. With learning rate of 0.01 and 2500 iterations we got the following results:

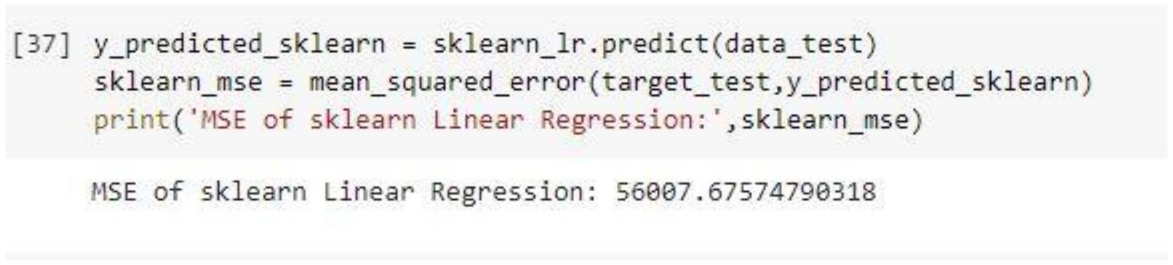


```
[35] from sklearn.metrics import mean_squared_error, mean_absolute_error
     y_predicted = my_lr.predict(data_test)

     my_mse = mean_squared_error(target_test,y_predicted)

     print('MSE of my Linear Regression:',my_mse)

     MSE of my Linear Regression: 56289.23199168802
```

Figure 1. mean squared error of trained model

3. Comparing with Sklearn

Now, we train a linear regression using sklearn library then compare it with my model! As figure 2 shows, we got almost the same MSE as above results on test set with sklearn model.

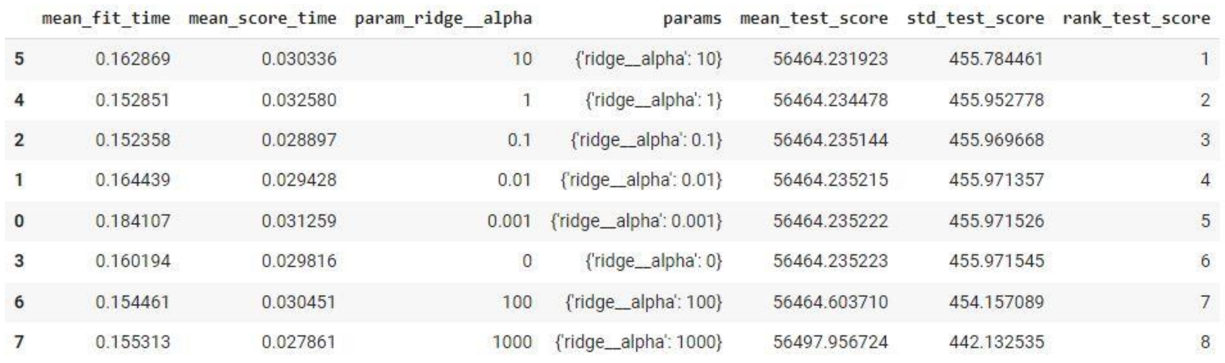


```
[37] y_predicted_sklearn = sklearn_lr.predict(data_test)
     sklearn_mse = mean_squared_error(target_test,y_predicted_sklearn)
     print('MSE of sklearn Linear Regression:',sklearn_mse)

     MSE of sklearn Linear Regression: 56007.67574790318
```

Figure 2. mean squared error of sklearn trained model

4. Ridge and Lasso

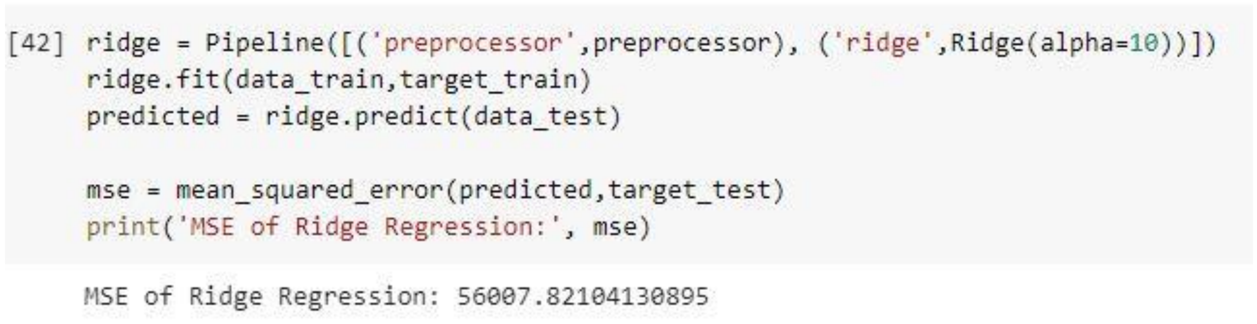
We use the same 3 selected features as above section and tune our hyperparameter alpha for ridge model. As the following figure shows, we got almost the same results with different alphas.



	mean_fit_time	mean_score_time	param_ridge__alpha	params	mean_test_score	std_test_score	rank_test_score
5	0.162869	0.030336	10	{'ridge__alpha': 10}	56464.231923	455.784461	1
4	0.152851	0.032580	1	{'ridge__alpha': 1}	56464.234478	455.952778	2
2	0.152358	0.028897	0.1	{'ridge__alpha': 0.1}	56464.235144	455.969668	3
1	0.164439	0.029428	0.01	{'ridge__alpha': 0.01}	56464.235215	455.971357	4
0	0.184107	0.031259	0.001	{'ridge__alpha': 0.001}	56464.235222	455.971526	5
3	0.160194	0.029816	0	{'ridge__alpha': 0}	56464.235223	455.971545	6
6	0.154461	0.030451	100	{'ridge__alpha': 100}	56464.603710	454.157089	7
7	0.155313	0.027861	1000	{'ridge__alpha': 1000}	56497.956724	442.132535	8

Figure 3. alpha tuning for ridge

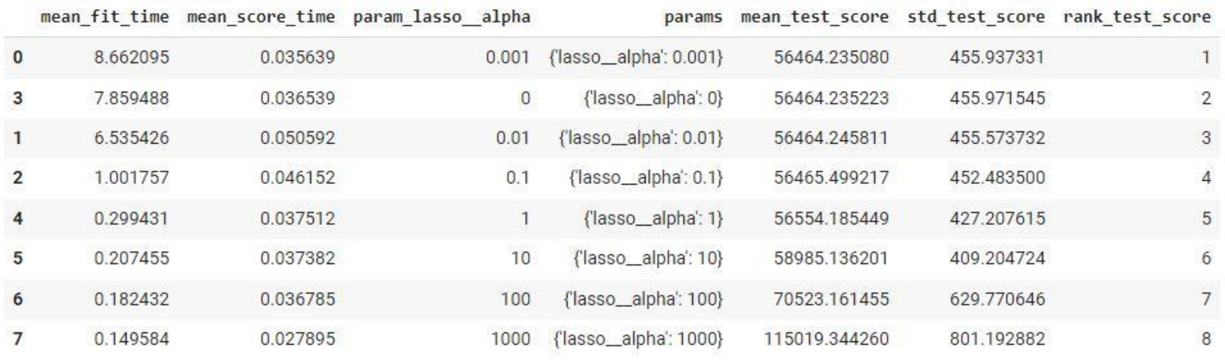
So, we train a model on train set and evaluate it on test set.



```
[42] ridge = Pipeline([('preprocessor',preprocessor), ('ridge',Ridge(alpha=10))])
     ridge.fit(data_train,target_train)
     predicted = ridge.predict(data_test)

     mse = mean_squared_error(predicted,target_test)
     print('MSE of Ridge Regression:', mse)

     MSE of Ridge Regression: 56007.82104130895
```

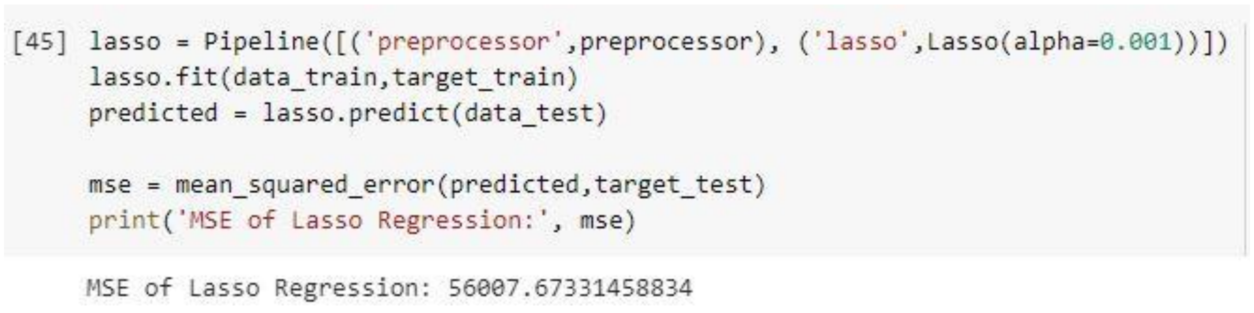
Figure 4. MSE of Ridge

Now, we tune alpha parameter for lasso regression. Then, evaluate model with best alpha on test set.



	mean_fit_time	mean_score_time	param_lasso__alpha	params	mean_test_score	std_test_score	rank_test_score
0	8.662095	0.035639	0.001	{'lasso__alpha': 0.001}	56464.235080	455.937331	1
3	7.859488	0.036539	0	{'lasso__alpha': 0}	56464.235223	455.971545	2
1	6.535426	0.050592	0.01	{'lasso__alpha': 0.01}	56464.245811	455.573732	3
2	1.001757	0.046152	0.1	{'lasso__alpha': 0.1}	56465.499217	452.483500	4
4	0.299431	0.037512	1	{'lasso__alpha': 1}	56554.185449	427.207615	5
5	0.207455	0.037382	10	{'lasso__alpha': 10}	58985.136201	409.204724	6
6	0.182432	0.036785	100	{'lasso__alpha': 100}	70523.161455	629.770646	7
7	0.149584	0.027895	1000	{'lasso__alpha': 1000}	115019.344260	801.192882	8

Figure 5. alpha tuning for lasso



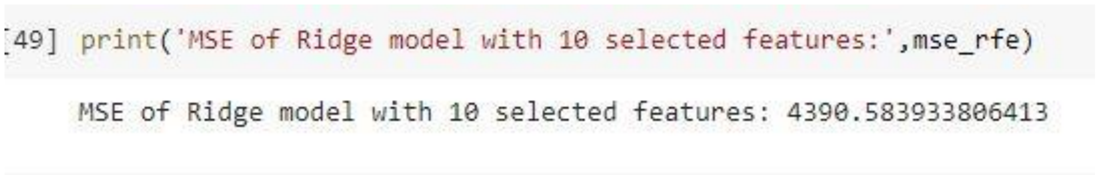
```
[45] lasso = Pipeline([('preprocessor',preprocessor), ('lasso',Lasso(alpha=0.001))])
     lasso.fit(data_train,target_train)
     predicted = lasso.predict(data_test)

     mse = mean_squared_error(predicted,target_test)
     print('MSE of Lasso Regression:', mse)

     MSE of Lasso Regression: 56007.67331458834
```

Figure 6. MSE of Lasso

5. EXTRA

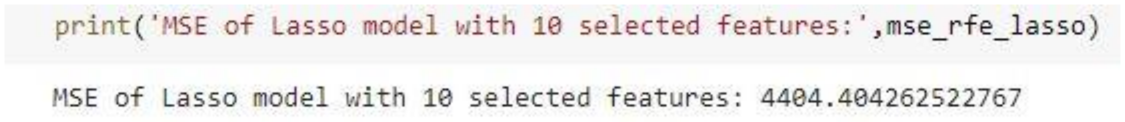
5.1. Feature Selection with RFE method

We use all of features and give them to RFE method to select top 10 features of it. We use ridge and lasso for this purpose. In below figures, you can see the results.



```
[49] print('MSE of Ridge model with 10 selected features:',mse_rfe)

     MSE of Ridge model with 10 selected features: 4390.583933806413
```

Figure 7. MSE of ridge



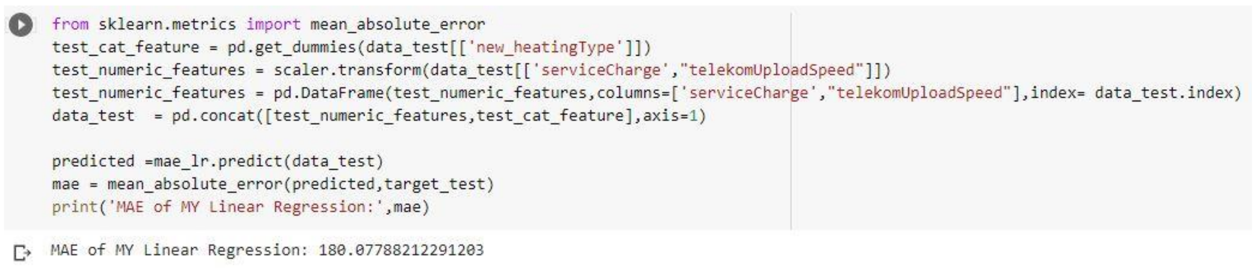
```
print('MSE of Lasso model with 10 selected features:',mse_rfe_lasso)

MSE of Lasso model with 10 selected features: 4404.404262522767
```

Figure 8. MSE of Lasso

5.2. Linear Regression with mean absolute error

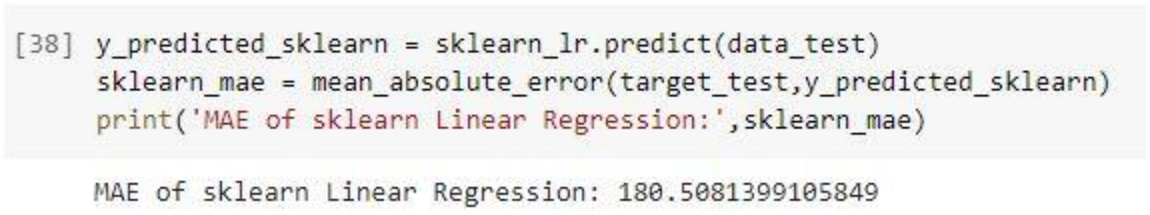
We implemented another linear regression model with mean absolute error as cost function. We used the same 3 selected features as before and at the end we compare our model with sklearn one.

With learning rate of 1 and 5000 iterations we got following results:



```
from sklearn.metrics import mean_absolute_error
test_cat_feature = pd.get_dummies(data_test[['new_heatingType']])
test_numeric_features = scaler.transform(data_test[['serviceCharge',"telekomUploadSpeed"]])
test_numeric_features = pd.DataFrame(test_numeric_features,columns=['serviceCharge',"telekomUploadSpeed"],index= data_test.index)
data_test  = pd.concat([test_numeric_features,test_cat_feature],axis=1)

predicted =mae_lr.predict(data_test)
mae = mean_absolute_error(predicted,target_test)
print('MAE of MY Linear Regression:',mae)

MAE of MY Linear Regression: 180.07788212291203
```

Figure 9. MAE of my linear regression



```
[38] y_predicted_sklearn = sklearn_lr.predict(data_test)
     sklearn_mae = mean_absolute_error(target_test,y_predicted_sklearn)
     print('MAE of sklearn Linear Regression:',sklearn_mae)

     MAE of sklearn Linear Regression: 180.5081399105849
```

Figure 10. MAE of sklearn linear regression

6. References

- Grokking Machine Learning, by Luis Serrano
- Feature Selection Techniques in Machine Learning[https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/]