**MACHINE LEARNING**

**TASK 2 - REPORT**

01.06.2022

***PART A***

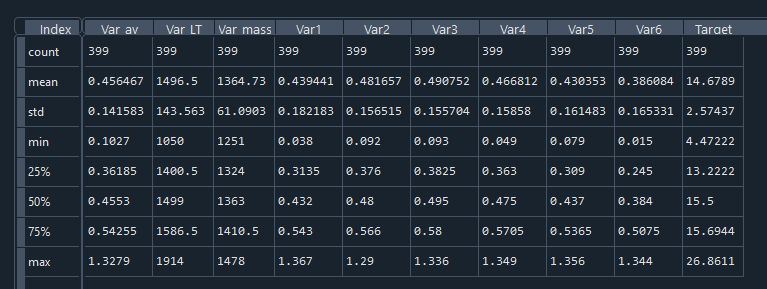
To begin with, let us discuss the task.

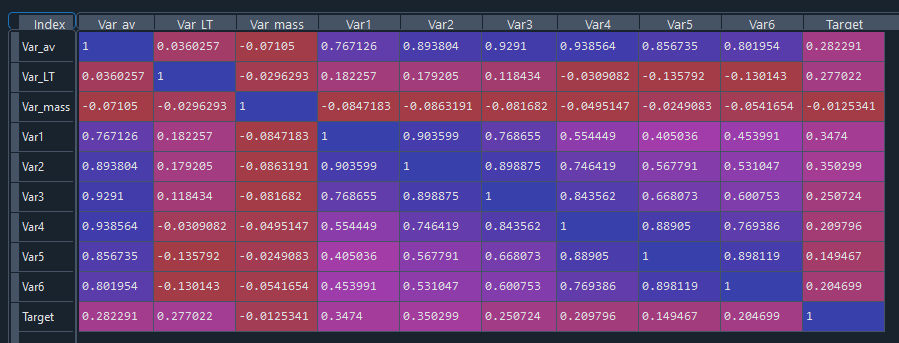
“The company Tajne-Przez-Poufne has commissioned you to predict a certain experimentally determined value of Target (as the project is top secret, the company has not told you the technical specifications of the projects carried out). Informally, the company only informed you that it is able to influence the value of the Target to some extent -- by modifying the parameters Var\_mass and Var\_LT, as well as changing the form of the experiment.”

Our main task will be to build a regressor and demonstrate how it works.

1.Basic understanding of the data:

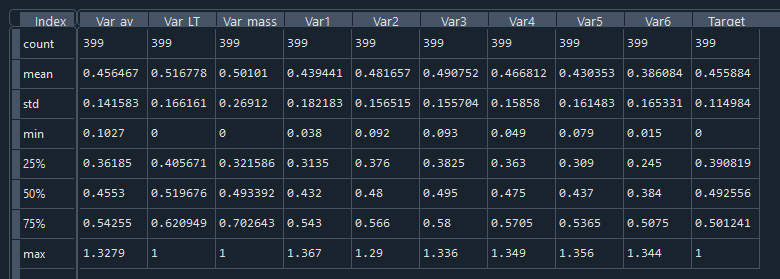
* It is probably some kind of physical experiment, a pendulum or something with movement (sine waves).
* Target value is probably some measure of the distance traveled by the object, or thrust.
* Most likely, the variables are environmental such as wind, temperature, humidity, coating, ball weight, cord length (on which the ball dangles), etc. The relationship between them is such that we have all the information that accompanied pushing the ball, apart from the force with which it was pushed. In addition, the company has influence over (so it probably manipulates in a controlled way) the variables Var\_mass and Var\_LT.



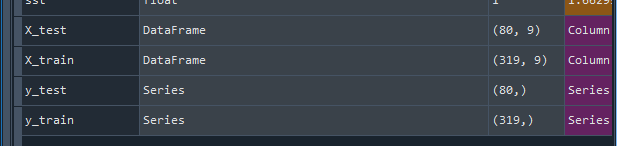


It is immediately noticeable that two columns need to be normalized. These will be “Var\_LT” and “Var\_mass”. Also, using correlation matrix we can agree that using all variables to predict “Target” will be understandable.

2.Data cleaning: There is no need for cleaning the data (with a high probability) because we have to deal with physical experience.  
3. Preparing data: We normlizied data



and split it into train and test sets.

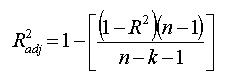


Later, when we decide to choose a method, we will also add a validation set to the training data to determine which parameters will work best for our model.

4.Testing various regressors: We will choose between implemented by hand Linear multiple regression (OLS), LinearRegression built in sklearn, OLS built in scipy and ANN built in keras/TensorFlow.

Note. For model validation, we will use measures such as:

* Adjusted R^2 whis is a corrected goodness-of-fit (model accuracy) measure for linear models.



* MAE (mean absolute error)



* MAPE (mean absolute percentage error)



* MSE (mean square error)

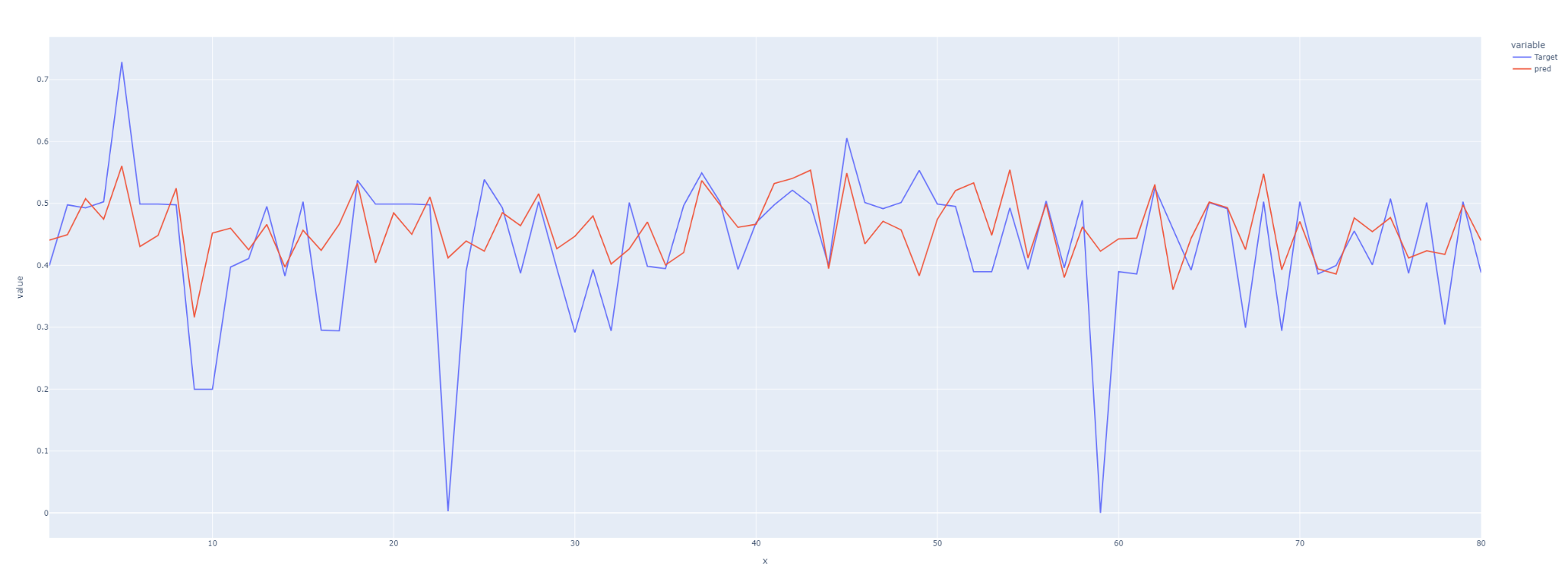


* AIC (Aike information criterion) which is an (best) estimator of prediction error and thereby relative quality of statistical models for a given set of data.



We will focus primarily on the measurement of AIC (the smaller the better).

**Linear multiple regression (OLS by hand)**



SSE: 0.766200725939506

SSR: 0.24685110459317428

SST: 1.0130518305326803

R^2: 0.2436707551906556

R^2 adjusted: 0.15845055859241952

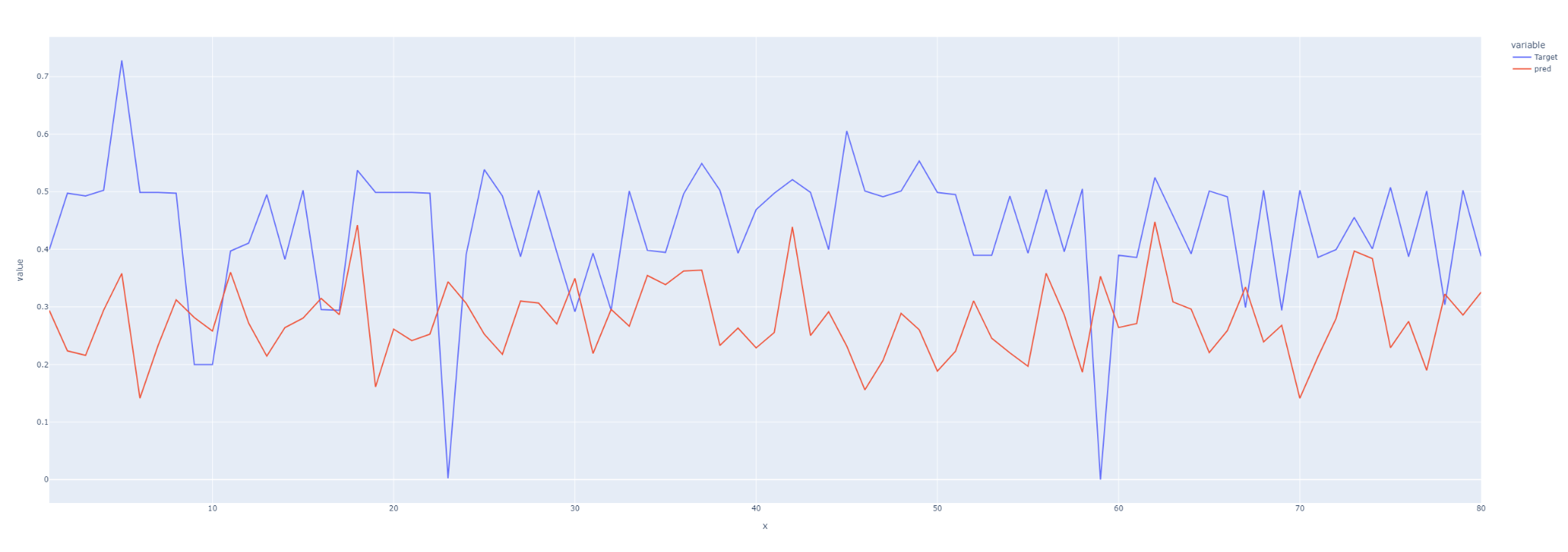
MAE: 0.06356744865735994

MAPE: 6.876637311090468

MSE: 0.009577509074243824

AIC: -351.8670187148473

**LinearRegression from sklearn**



SSE: 3.5544536881930853

SSR: 2.3435568519814076

SST: 5.898010540174493

R^2: 0.3973470098125788

R^2 adjusted: 0.32944244753793983

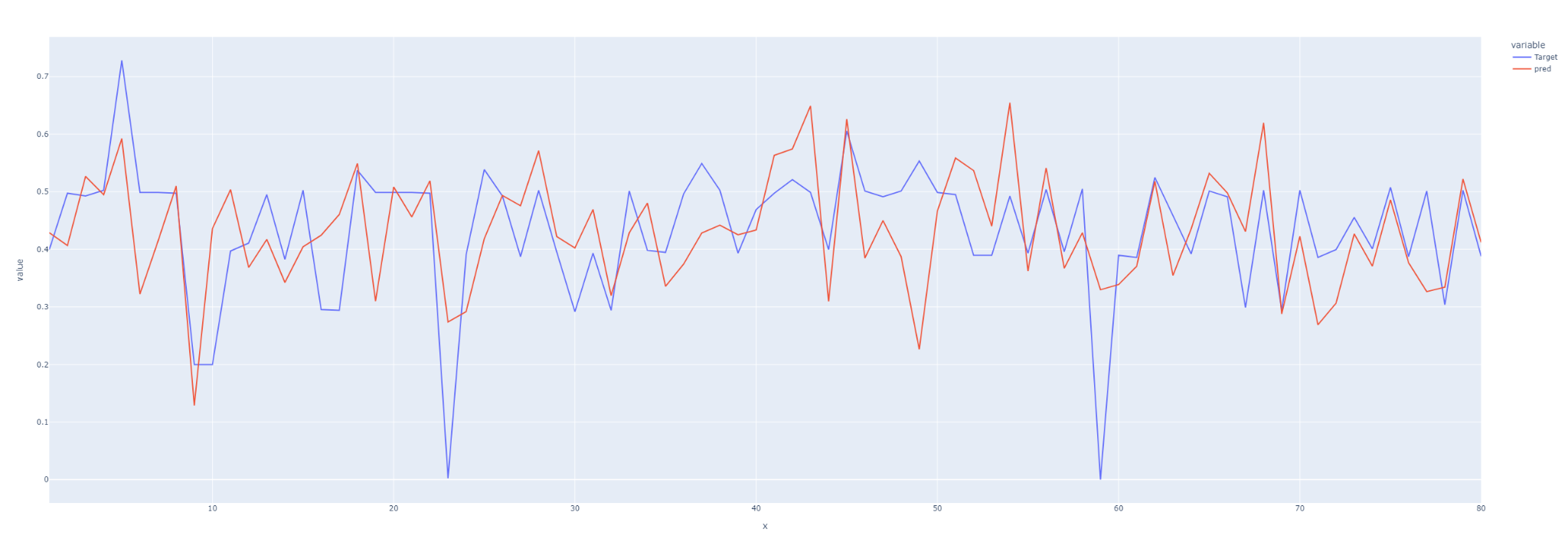
MAE: 0.18261900890501587

MAPE: 5.992398594734831

MSE: 0.04443067110241356

AIC: -229.10602059480465

**OLS from scipy.stats**

SSE: 0.8795504191504383

SSR: 0.7834472242201771

SST: 1.6629976433706153

R^2: 0.4711054326164058

R^2 adjusted: 0.4115116785450148

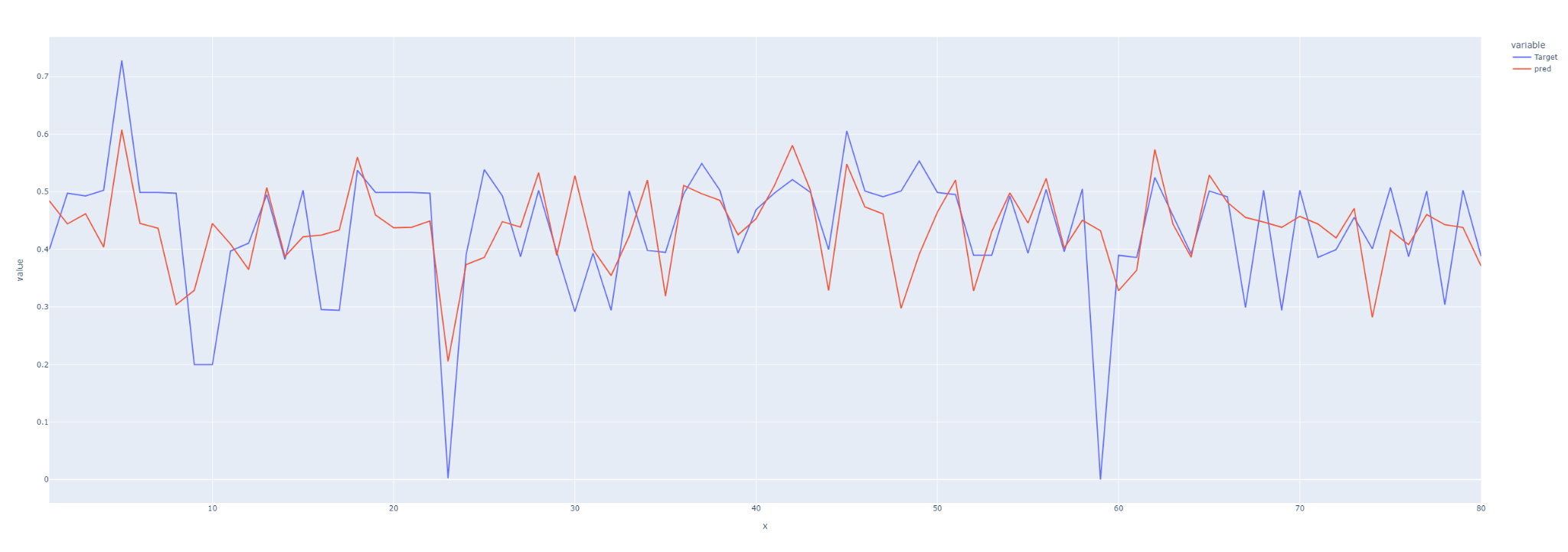
MAE: 0.07953817141185435

MAPE: 5.253673764715712

MSE: 0.010994380239380477

AIC: -340.8296819248169

**ANN regression**



SSE: 0.7636233026232983

SSR: 0.4051587973855373

SST: 1.1687821000088356

R^2: 0.34665041275227815

R^2 adjusted: 0.2730335578511265

MAE: 0.06830103593410526

MAPE: 14.370759963989258

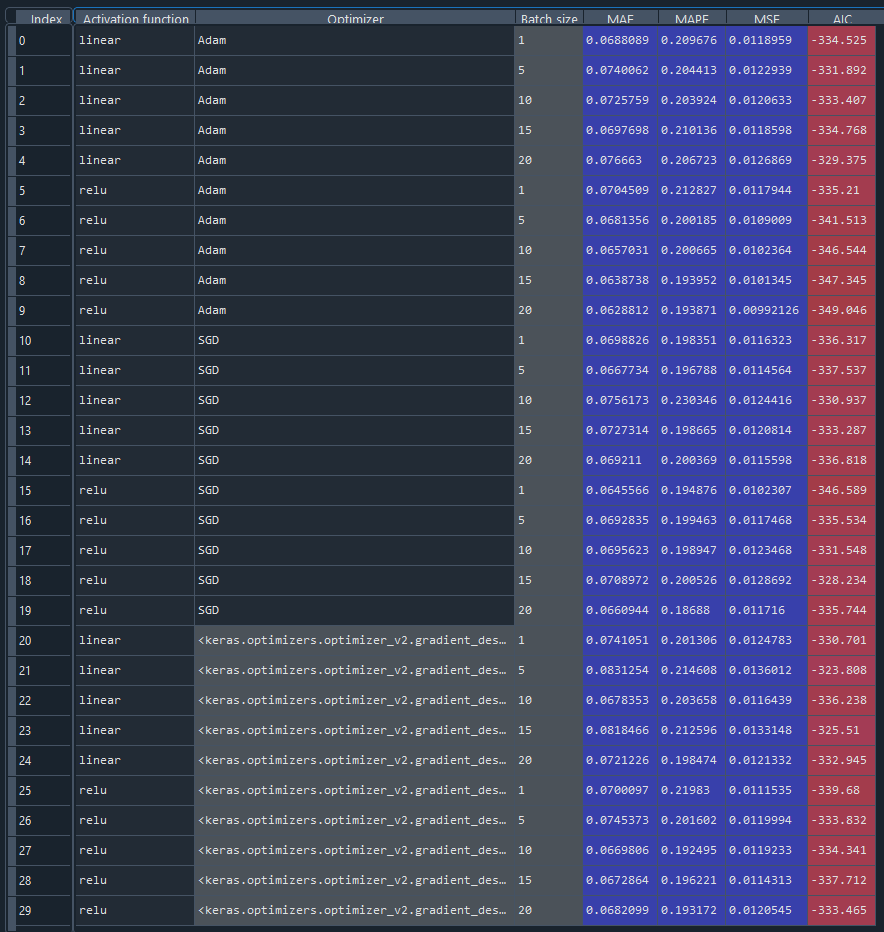
MSE: 0.009545291282791227

AIC: -352.1365844381472

We can easily see that linear regression (sklearn) is significantly underestimated. Multiple regression (by hand) behaves decently. Therefore, the choice will be between the OLS model and the ANN model. Although OLS performs similarly, the ANN is slightly better at capturing outliers and has overall better performance.

5. Training ANN: We will now test different activation functions, optimizer, number of epochs and batch sizes, for our model, to achieve the best possible results.

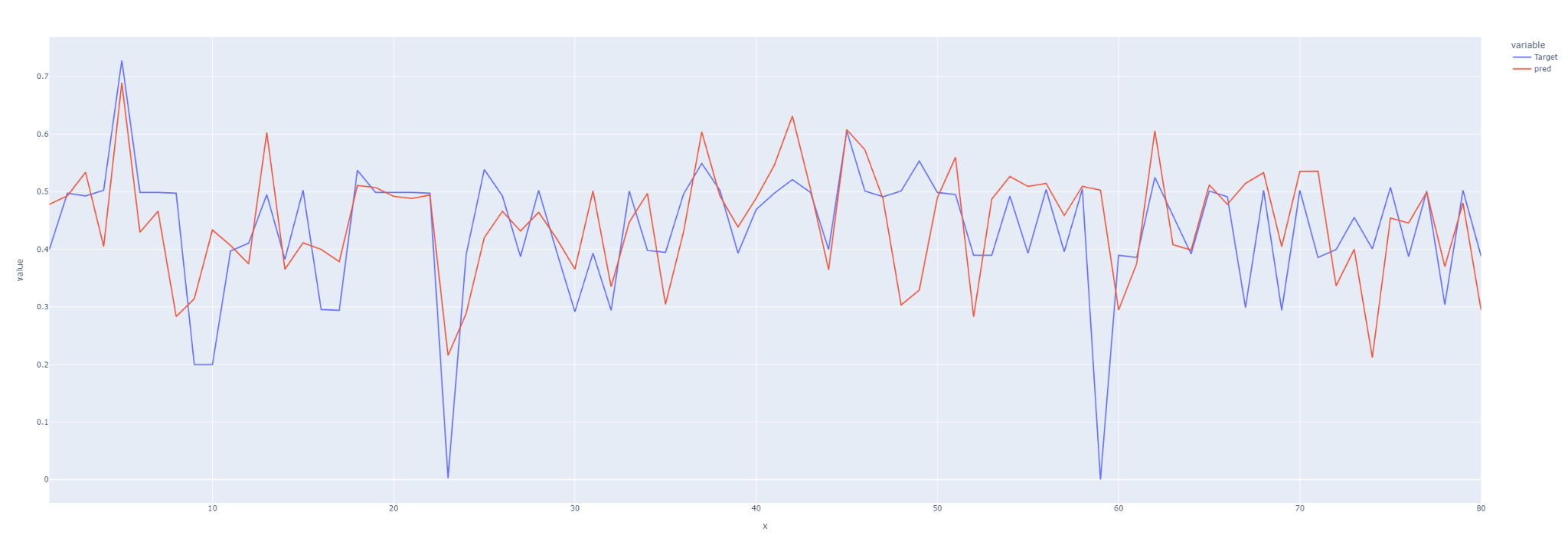
First, we will further split the data into training, validation and test sets. Now, based on the training and validation sets, with a fixed number of epochs = 100, we will compare performance using measures.



Our pick will be activation function = “relu” with optimizer = ‘Adam’ and batch size = 5.

Therefore, using the training set, we will train our model and visualize the results from the test data.

Note. The model's overtraining leveling function (EarlyStopping() ) pauses no further than the first 10 epochs. Hence, we will teach our model in no more than 250 epochs.



SSE: 0.8741802567836199

SSR: 0.7356550403703723

SST: 1.6098352971539922

R^2: 0.4569753450374257

R^2 adjusted: 0.3957894684219243

MAE: 0.07219897865941312

MAPE: 10.65102481842041

MSE: 0.010927253209795249

AIC: -341.31962526449774

Summary: In our deliberations, the ANN model proved to be the most effective. Not far behind it is the OLS model from the scipy package. For small amounts of data, the OLS model will be more convenient, while for increasing amounts of data, and thus increasing amounts of information to learn the model, it will be more optimal to reach for ANN. In the case of a hand-written function, i.e. multiple regression, it behaves decently. As for the built-in linear regression function from the sklearn package, it is far behind.

Another linear regression model that we applied to the data is Elastic Net but it was not as efficient as previous models. This is a combination of a Ridge and Lasso regressions, because it uses penalties from both these models. In our model there are multiple attributes which are highly correlated to each other. If we applied Lasso regression, the model would choose just one of the correlated attributes but in the Elastic Net model all of them will be included.

Before we apply the Elastic Net model, we split our dataset to training and test sets and we also standardize our data using StandardScaller(). Then we use ElasticNet() function to create our regresor and predict values for our testing data.

We will select different values of parameters ‘alpha’ and ‘l1\_ratio’ which control ridge and lasso regressions penalties. We would like to see which combination of these parameters will be the most effective. Measures that we will use to define the effectiveness of the model will be:

* standard deviation of residuals,
* convergence coefficient.

| **alpha** | **l1\_ratio** | **standard deviation of residuals** | **convergence coefficient** |
| --- | --- | --- | --- |
| default (1) | default (0,5) | 2,5661 | 0,9350 |
| 0,2 | 0 | 2,5961 | 0,7119 |
| 0,2 | 0,2 | 2,5982 | 0,7225 |
| 0,2 | 0,6 | 2,5829 | 0,7502 |
| 0,2 | 1 | 2,5738 | 0,7755 |
| 0,5 | 0 | 2,5958 | 0,7564 |
| 0,5 | 0,2 | 2,5772 | 0,7907 |
| 0,5 | 0,6 | 2,5581 | 0,8535 |
| 0,5 | 1 | 2,5781 | 0,9016 |
| 0,8 | 0 | 2,5931 | 0,7880 |
| 0,8 | 0,2 | 2,5622 | 0,8416 |
| 0,8 | 0,6 | 2,5655 | 0,9217 |
| 0,8 | 1 | 2,5625 | 0,9738 |

The best two settings are the following:

* alpha = 0,2, l1\_ratio = 0 - Ridge regression penalty
* alpha = 0,8, l1\_ratio = 0,2

but the differences in measures of effectiveness are really small, thus we can conclude that the Elastic Net regression model can be applied with any combination of parameters.

Coefficients that we obtained respectively are:

* 0.21581793, 0.37503251, -0.00282672, 0.46498216, 0.39332233, -0.22264118, 0.13735007, 0.01602662, and we have a warning that objective did not converge,
* 0.14532329, 0.19363816, 0, 0.29939877, 0.26303077, 0, 0.04655672, 0

***PART B***

In the second part of this task, data which will be classified describe telephones’ parameters, such as battery power, clock speed, five g, etc. These telephones are being sold in Anonidas’ shop. He would like to have a classifier which will help him define which price range the telephone belongs to.

We will use a random forest classifier to build such a model. This classifier is based on decision trees. It generates uncorrelated decision trees which are really diverse. It is ensured by bagging and future randomness.

Bagging (Bootstrap Aggregation) means that each individual tree randomly selects a sample from the dataset with replacement. It means that if we have a dataset of size n, then the training set for an individual tree is also of size n but instead of the original data we have a sample selected with replacement, so some of the records can be repeated in the training set.

Whereas future randomness means that there is only a random subset of features available from which a condition to split a node is chosen. In a standard decision tree we can choose the most separating future from all of them, but in this model we can choose only from the random subset of all futures.

Now, we are going to describe how a random forest classifier predicts the class for a specific observation. For all trees in the random forest, the class is being predicted as in the standard decision tree. Then, we choose the model’s predicted class as the class which appeared in trees’ predictions most often.

Before we use the random forest classifier we need to preprocess our data. We standardize all the futures, then all of them will have 0 mean and variation of 1. Because no data is missing in this dataset, we do not have to apply any methods of data completion.

We visualize the result of prediction using a confusion matrix C, which shows in which class the observation was predicted to be and which class it really belongs to. On the place ci,j there is a number of observations that are known to be in class i but were predicted to be in class j. We also use an accuracy score to see how many observations were classified correctly.

In this case only 51 observations were classified incorrectly and 349 were classified to the proper class. The accuracy score is 0.8725 which is quite high and we conclude that this method of classification is effective.

We will apply different values of some parameters to test which combination would be the best classifier for this data. ‘n\_estimator’ parameter determines how many random trees will be generated and ‘min\_samples\_split’ defines how many samples have to be in the node to be split it:

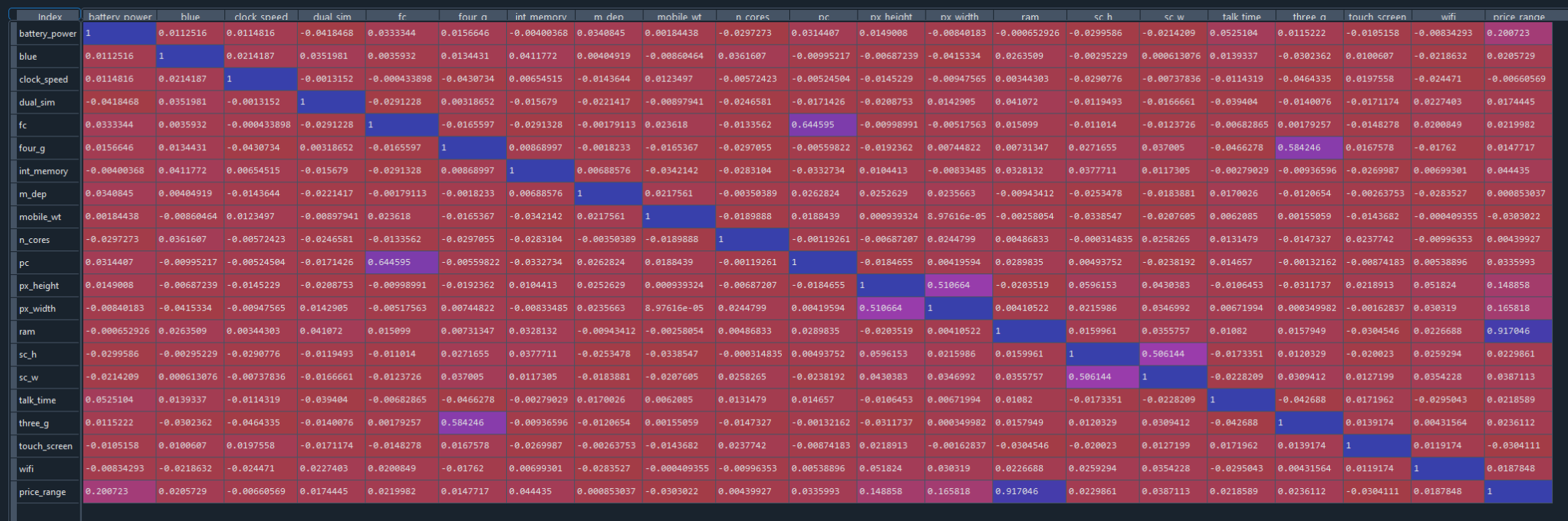
| **n\_estimators** | **min\_samples\_split** | **number of samples classified correctly** | **accuracy score** |
| --- | --- | --- | --- |
| 30 | 2 | 349 | 0,8725 |
| 30 | 5 | 345 | 0,8625 |
| 30 | 10 | 350 | 0,8750 |
| 60 | 2 | 348 | 0,8700 |
| 60 | 5 | 348 | 0,8700 |
| 60 | 10 | 347 | 0,8675 |
| 100 | 2 | 349 | 0,8725 |
| 100 | 5 | 354 | 0,8850 |
| 100 | 10 | 354 | 0,8850 |

We can clearly conclude that the best parameters’ settings are the following:

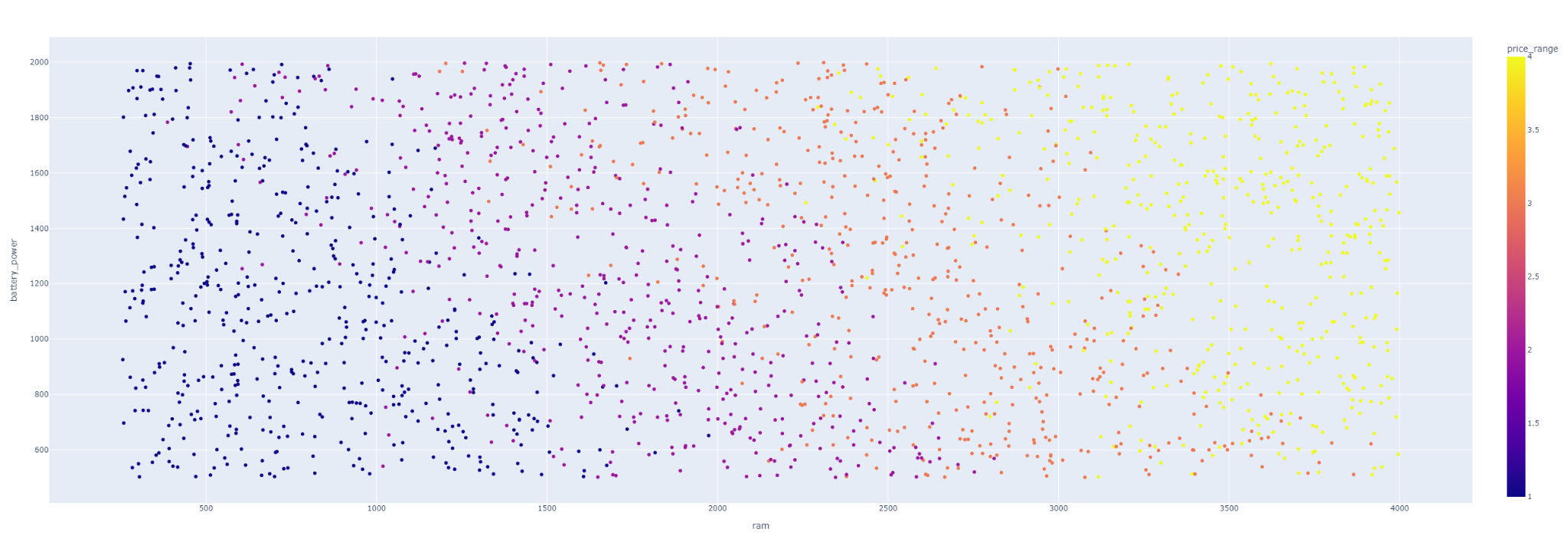
* n\_estimators = 100, min\_samples\_split = 5,
* n\_estimators = 100, min\_samples\_split = 10,

because they result in the best accuracy score and the largest number of samples which are classified correctly.

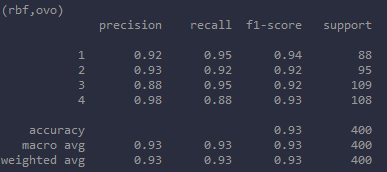
Now lets take a look at the Support Vector Machine classification method. We will use build in method from the sklearn package. Before we begin, we will take quick peek a correlation matrix:

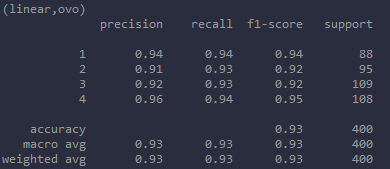
It is immediately apparent that RAM is extremely important for classification. Additionally we can add: battery\_power, px\_height and px\_width.

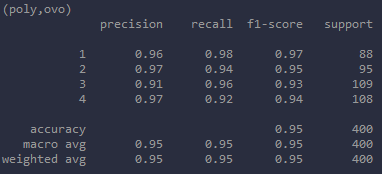
Plotting RAM against battery power, clearly show possibility of linear separation i.e.



Moving on to the selection of parameters, we will pick between: radial basis function (rbf), linear function (linear) and polynomial function (poly) as a kernel in our clasificator and “one-versus-one” method. Lets compare classification reports for these functions.



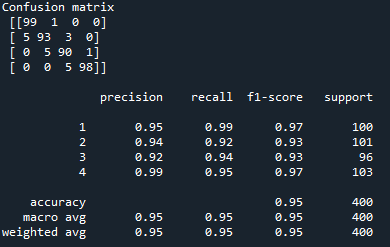




All three of them give decent results. So we need to ask ourselves a question. On which measure should we focus most? Well, we want to minimize losses and maximize profits by allocating phones to the right price shelves. Therefore, we will mainly minimize recalls for categories 3 and 4, as it can be more costly to undercut them, with accuracy in mind.

In this case, the polynomial kernel works best.

Lets run our learned classifier on test data.



In summary, a very high recall was achieved for categories 3 and 4, with errors more in favor of the seller (single inflated categories) and accuracy at 95%. Thus, it opens up the possibility of possible promotions or sales in the absence of interest without incurring losses.