IMMO-ELIZA

Houses and apartments price predictions

HAMERS Robin - May 2025

Cleaning the data

-> preprocessing.py

- Function that read csv, split into test and train
- Transform data into numeric, delete NaN rows for price
- Get coordinates from Giraffe csv and do KNN with price per squared meter using KDE (Kernel density estimation)
- Standardize for some models, not for others

Multiple models tests with PyCaret

-> predicting_pyCaret.ipynb

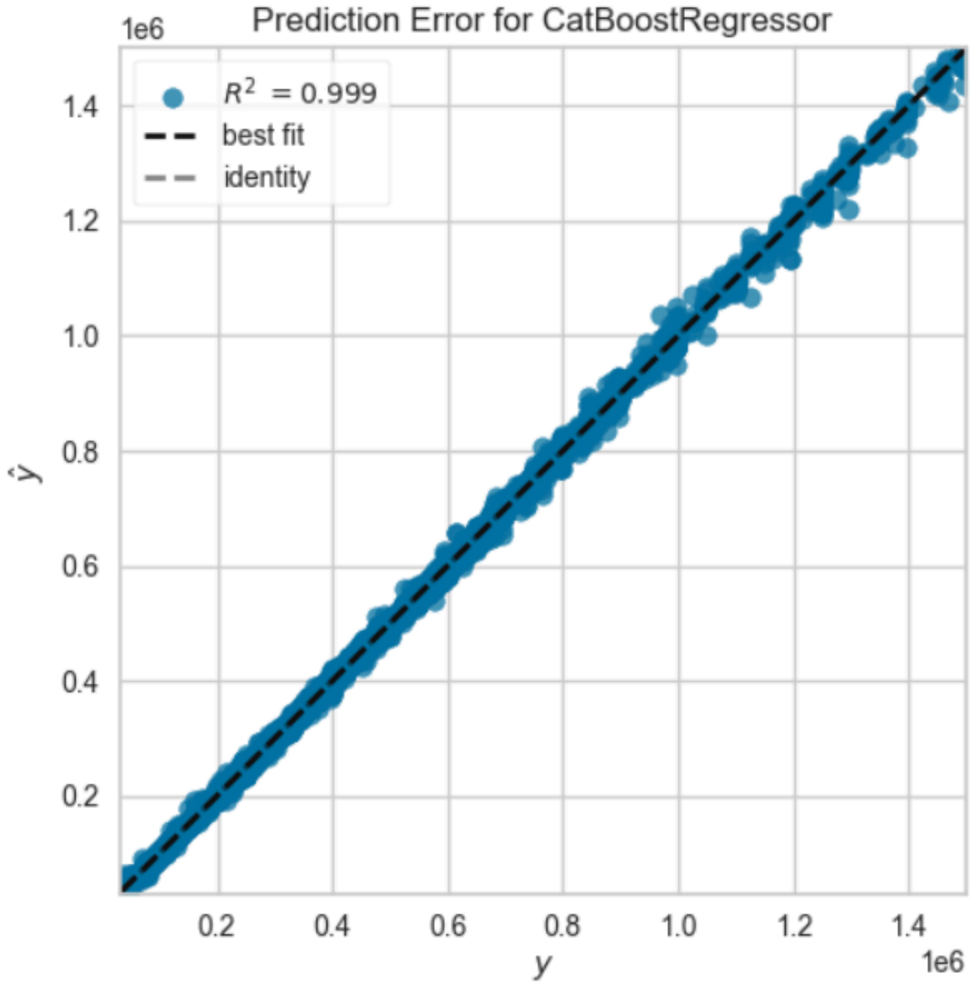
	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
catboost	CatBoost Regressor	4775.6990	180016426.7849	13112.8603	0.9967	0.0316	0.0134	2.1850
rf	Random Forest Regressor	4008.0710	266503083.6191	16099.8211	0.9952	0.0313	0.0094	3.9690
lightgbm	Light Gradient Boosting Machine	7550.3758	270023389.0976	16187.5978	0.9951	0.0378	0.0211	0.4720
xgboost	Extreme Gradient Boosting	9395.1763	424599001.6000	20454.4348	0.9923	0.0425	0.0248	0.1660
dt	Decision Tree Regressor	8076.8284	607804511.6229	24282.0575	0.9889	0.0484	0.0191	0.1980
gbr	Gradient Boosting Regressor	14288.4043	626103849.0153	24939.3546	0.9886	0.0632	0.0415	2.1180
et	Extra Trees Regressor	10490.8564	914213308.2311	30087.6606	0.9835	0.0584	0.0262	5.0740
ada	AdaBoost Regressor	133330.3221	21968275787.1072	148082.4866	0.6016	0.4884	0.5555	1.7040
knn	K Neighbors Regressor	138264.4688	41008485990.4000	202439.4359	0.2575	0.4824	0.4260	0.1400
omp	Orthogonal Matching Pursuit	135198.3704	45572201697.5597	210617.7813	0.1707	0.4657	0.4464	0.0340
huber	Huber Regressor	150161.6780	52756016112.3357	229103.8738	0.0433	0.5279	0.4452	0.1420
dummy	Dummy Regressor	168205.3484	55234320793.6000	234988.5766	-0.0003	0.5856	0.5977	0.0440
en	Elastic Net	94802.5463	80137555954.7649	217720.5340	-0.4452	0.3373	0.2903	0.9840
par	Passive Aggressive Regressor	240461.2502	96264693140.5654	307950.2743	-0.7423	0.9170	0.7583	0.0800
llar	Lasso Least Angle Regression	87543.6574	197104288040.3220	257882.0782	-2.5466	0.3633	0.2702	0.0450
lr	Linear Regression	86265.9512	242218398017.0526	271907.3796	-3.3577	0.3649	0.2688	0.4360
lasso	Lasso Regression	86176.8060	244333385537.6194	272681.3296	-3.3957	0.3695	0.2678	0.8500
ridge	Ridge Regression	86218.1542	245555191083.0296	273189.3282	-3.4177	0.3708	0.2676	0.0300
br	Bayesian Ridge	86207.6695	245919321765.1690	273306.1586	-3.4242	0.3682	0.2675	0.1600
lar	Least Angle Regression	24806134264.6628	4085843970755744633978888.0000	207107850871.9962	-7762702017749.1396	8.5963	111148.0465	0.0590

Multiple models tests with PyCaret

-> predicting_pyCaret.ipynb

- Got CatBoost as a best model
- On Test dataset :

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	CatBoost Regressor	4562.3667	124927658.2891	11177.1042	0.9978	0.0279	0.0130



Linear Regression with NumPy

-> linearRegression_scratch.ipynb

- Added a column filled with 1 for prices in the prediction matrix
- Find beta with normal equation (X = X_train) :
- beta = (X(transposed)*X)^-1 * (X(transposed)*y) #with * as dot product
- Than make predictions with X_test
- preds = X_test * beta #with * as dot product
- Bad results for now :

```
MAE = 4378597.040455299, MAPE = 1518.5471878028306, RMSE = 4971328.562809775, r_square = -308.1116948889128
```

NeuralNetwork - PyTorch

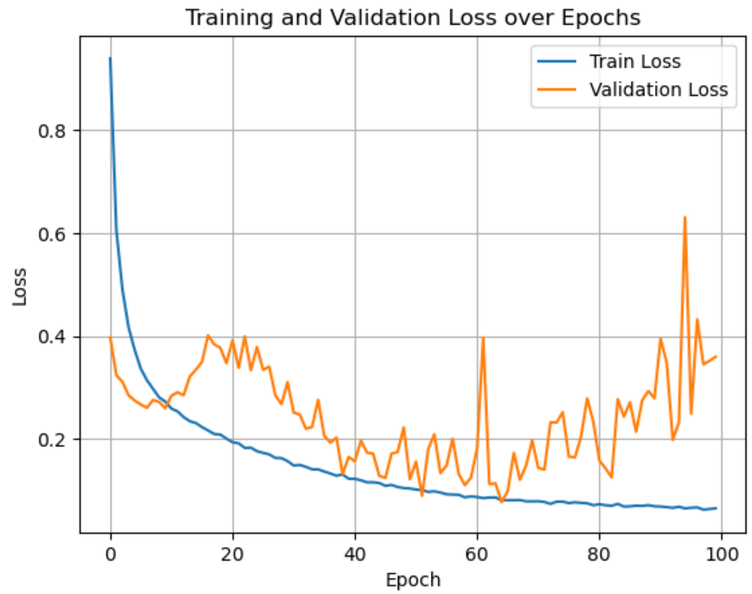
-> predicting_NN.ipynb

- Fully connected feedForward neural network
- Input Layer : Linear(93->512), BatchNorm1d, ReLu, Dropout
- HiddenLayer 1 : Linear(512->256), BatchNorm1d, ReLu, Dropout
- HiddenLayer 2 : Linear(256->128), BatchNorm1d, ReLu, Dropout
- Output layer : Linear(128->1)

NeuralNetwork - PyTorch

-> predicting_NN.ipynb

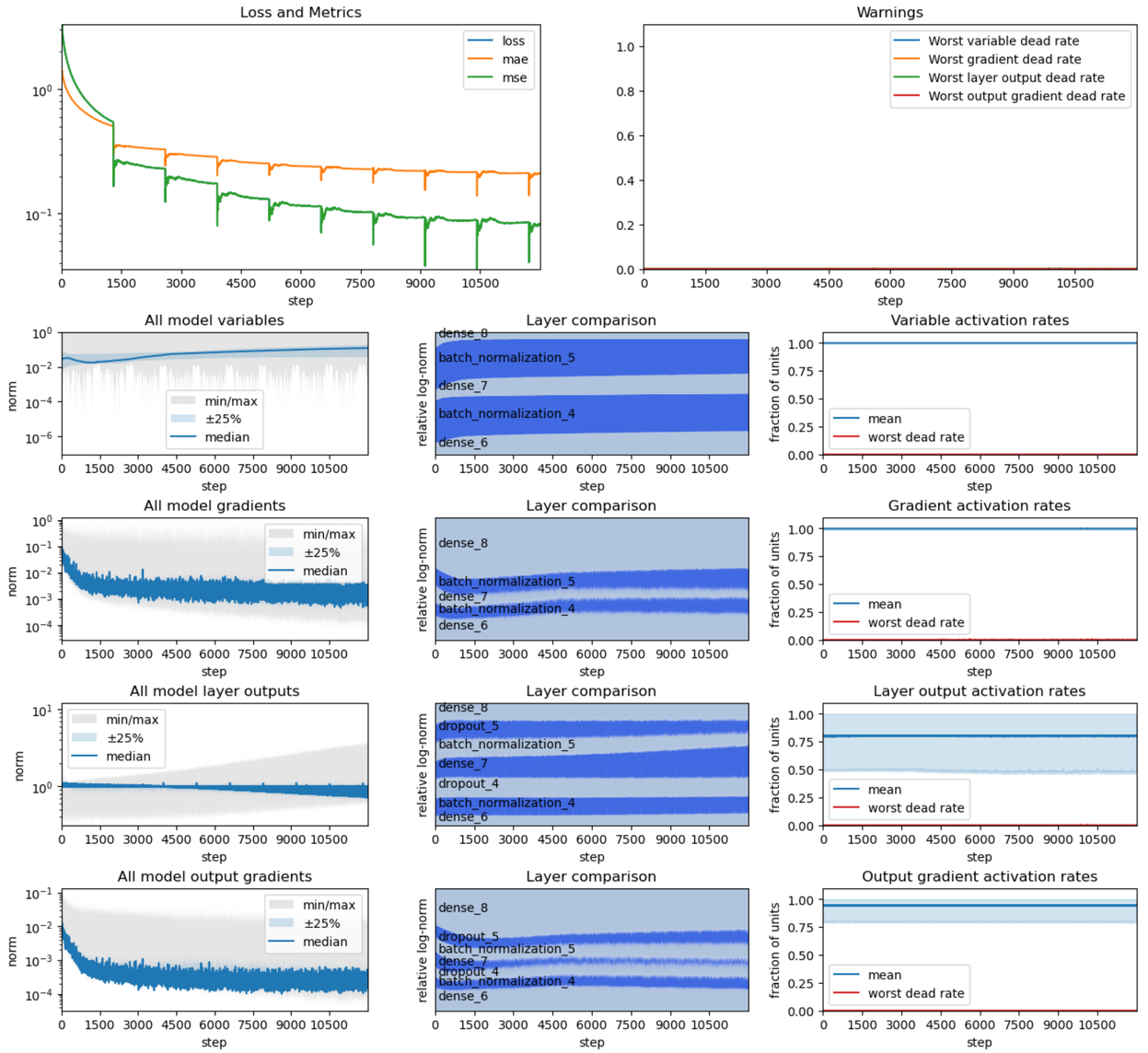
- Train this on 200 epochs with early stopping
- When 35 epochs are less good than previous one, it stops and keep the last good state
-
```
Train : R2 = 0.2998, MAE = 17843.6152, MSE = 24208351232.0000, RMSE = 155590.3281, MAPE = 5.7689%
Test : R2 = 0.6426, MAE = 20137.0430, MSE = 12437147648.0000, RMSE = 111521.9609, MAPE = 6.5109%
```



NeuralNetwork - Keras

-> predicting_NN_Keras.ipynb

- This is just to show how the model looks like after training
- Top left : spikes comes from validation at regular intervals
- Top right no dead units, good
- Than we can visualize Variables, Gradients, Outputs and Outputs Gradients



Conclusion

- The neural network is the best I tested
- With MAE = 20137.04 on test data
- Save it and keep it for later use

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 512]          48,128
       BatchNorm1d-2                  [-1, 512]           1,024
              ReLU-3                  [-1, 512]               0
           Dropout-4                  [-1, 512]               0
            Linear-5                  [-1, 256]         131,328
       BatchNorm1d-6                  [-1, 256]             512
              ReLU-7                  [-1, 256]               0
           Dropout-8                  [-1, 256]               0
            Linear-9                  [-1, 128]          32,896
      BatchNorm1d-10                  [-1, 128]             256
             ReLU-11                  [-1, 128]               0
          Dropout-12                  [-1, 128]               0
           Linear-13                    [-1, 1]             129
================================================================
Total params: 214,273
Trainable params: 214,273
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.03
Params size (MB): 0.82
Estimated Total Size (MB): 0.85
----------------------------------------------------------------
```