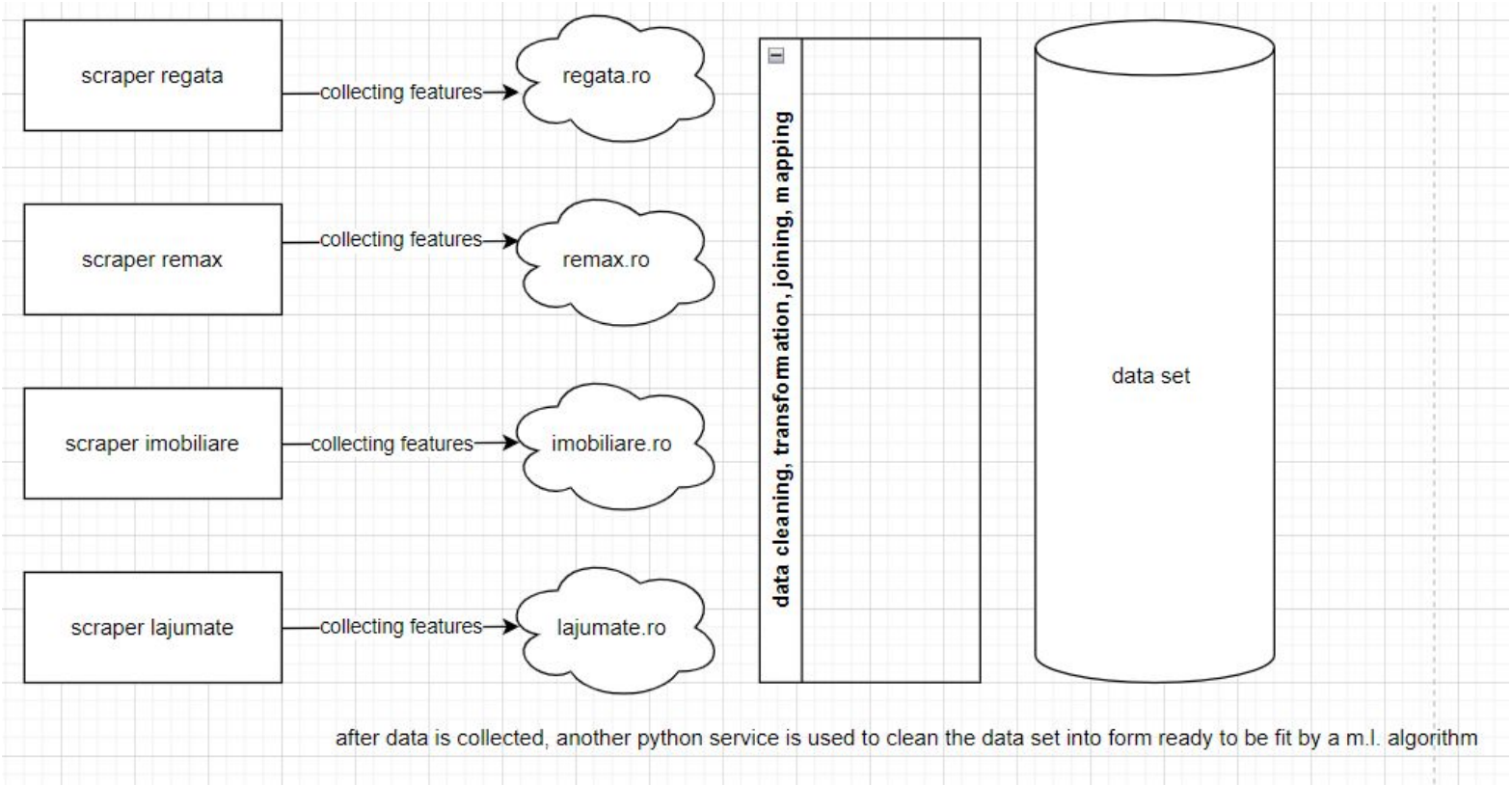
Real estate data collection, mining and learning

Fulea Andrei, 511
Burz Florin, 511

Dataset

- the dataset used is collected automatically from online sources with web scraping - ETL - .



Data cleaning

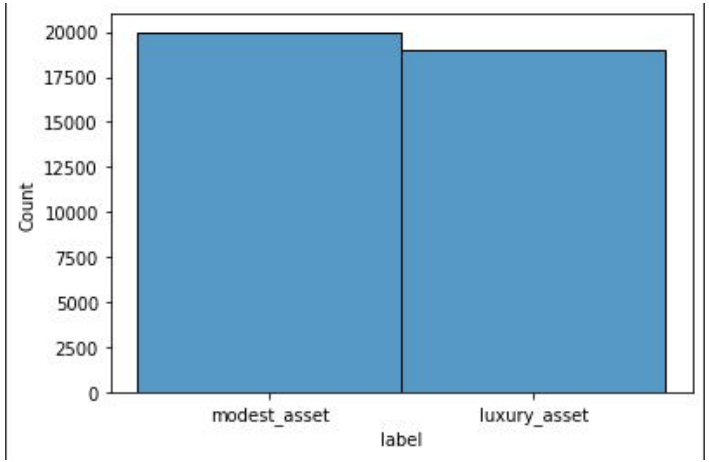
- in the initial form, the data is in object format read by pandas
- after the cleaning and transformation, we eliminate useless part of text and keep just relevant information + casting to data type

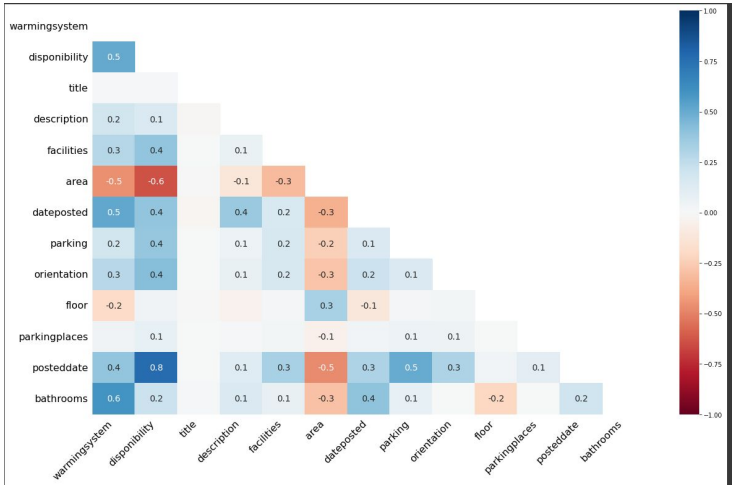
price	33.000 €
assetstate	Bună
warmingsystem	Calorifere, Termoficare
disponibility	Imediat
colector	remax
title	Garsoniera vanzare in bloc de apartamente Bucu...
description	\r\n\r\nPROPRIETATEA ESTE GREVATA DE SARCINA U...
facilities	NaN
compartimentation	Decomandat
rooms	1 camera
yearconstruction	1985.0
confort	1
area	Sectia Politie 20
pagenumber	1.0
dateposted	Acum o zi
town	bucuresti-ilfov
parking	NaN
assettype	apartamente
transactiontype	vanzare
orientation	vedere stradala
neighborhood	NaN
balcony	1 balcon
furnished	nespecificat
floor	1
parkingplaces	2.0
posteddate	23 August 2021
link	https://www.remax.ro/anunt/75391/garsoniera-de...
squaremetres	36.62 mp
bathrooms	1.0

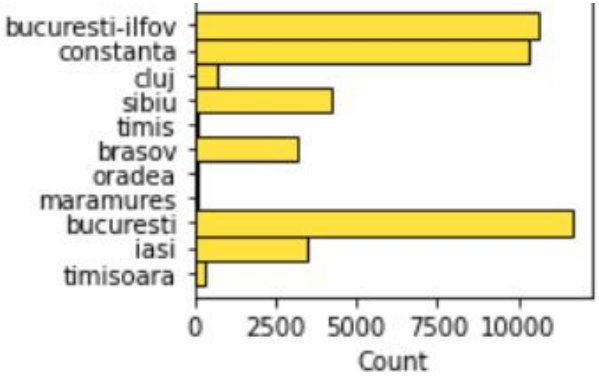
price	33000
assetstate	2
warmingsystem	Calorifere, Termoficare
disponibility	Imediat
colector	4
title	Garsoniera vanzare in bloc de apartamente Bucu...
description	\r\n\r\nPROPRIETATEA ESTE GREVATA DE SARCINA U...
facilities	NaN
compartimentation	7
rooms	1
yearconstruction	1985
confort	2
area	Sectia Politie 20
pagenumber	1
dateposted	Acum o zi
town	bucuresti-ilfov
parking	NaN
assettype	apartamente
transactiontype	vanzare
orientation	vedere stradala
neighborhood	NaN
balcony	1
furnished	4
floor	1
parkingplaces	2.0
posteddate	23 August 2021
link	https://www.remax.ro/anunt/75391/garsoniera-de...
squaremetres	36
bathrooms	1.0

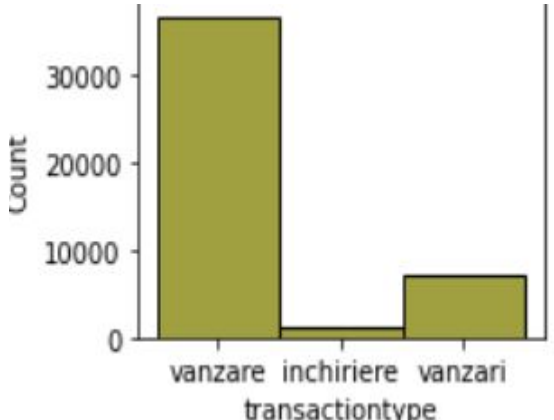
EDA - p1

- static variables



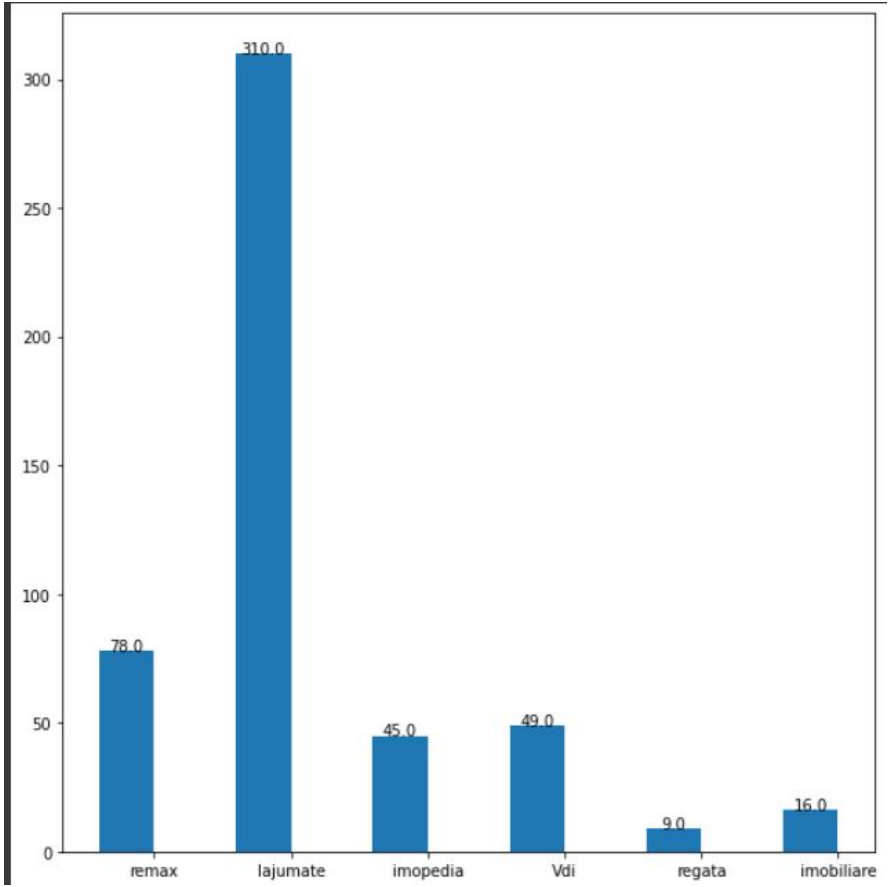


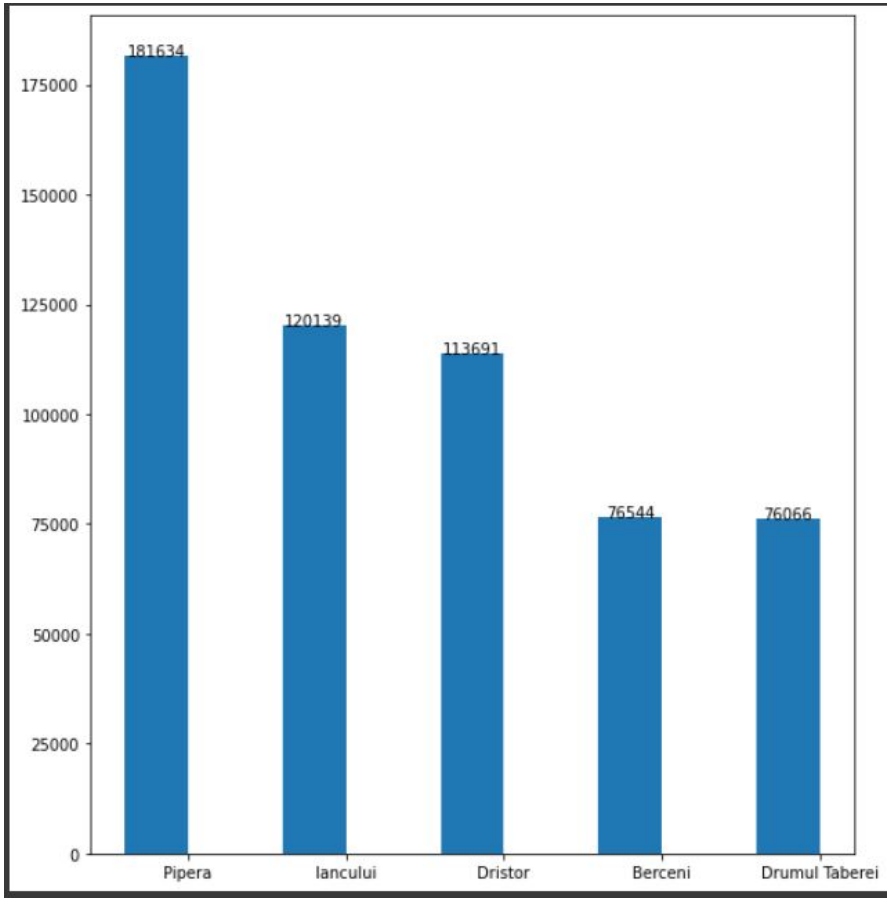




EDA - p2

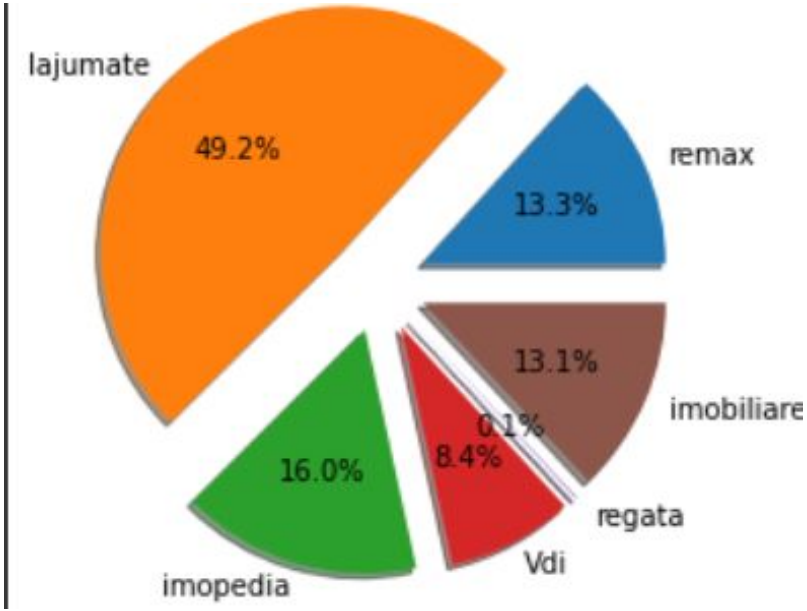
Distribution of medium price with areas and max page collector from each website

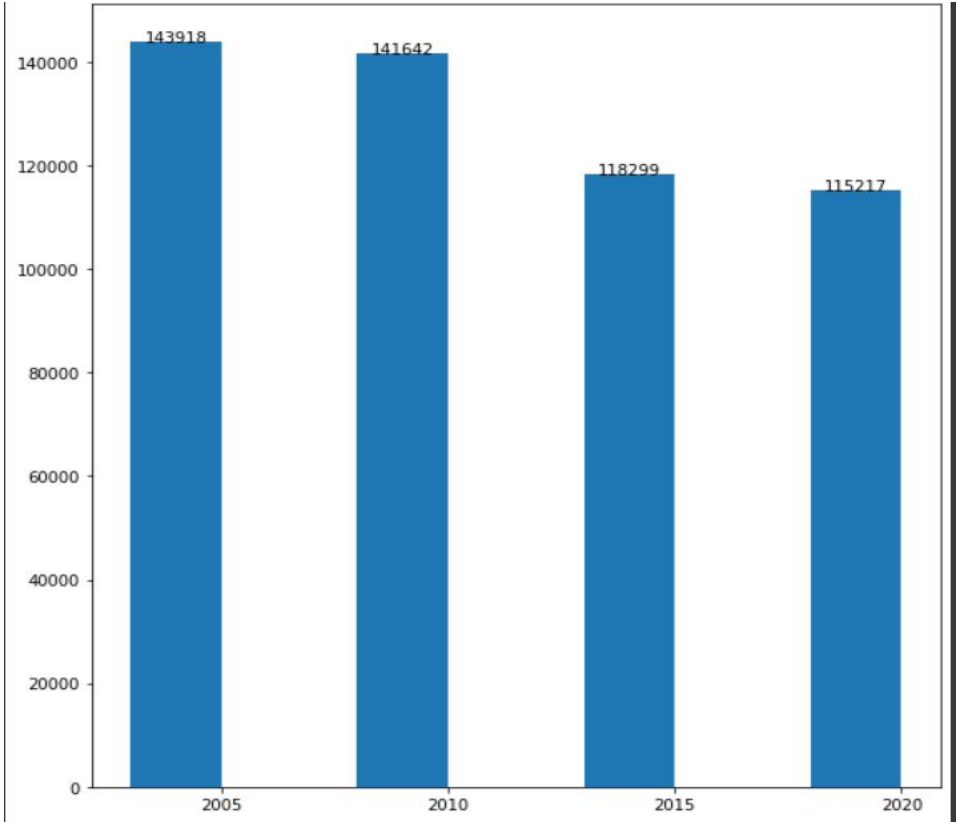




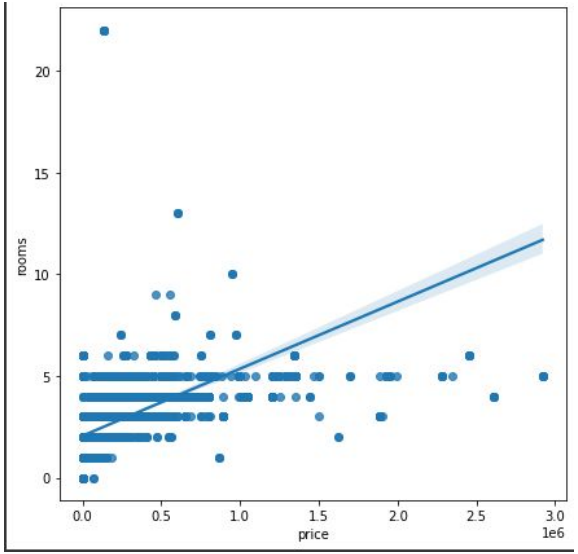
EDA - p3

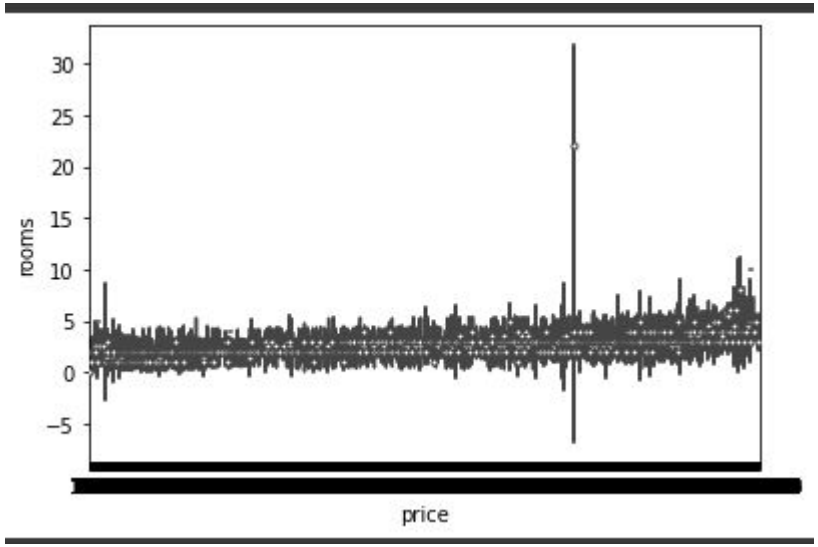
Distribution of medium price with year of construction and number of samples from each website





EDA - p4
Distributions, variance, observations





Regression problem - price prediction

- the task is to predict the price of an asset
- We construct the X and Y, y being df[‘price’] and X

```
x = national_real_estate_data_CLEANED[['rooms','yearconstruction','confort','furnished','squaremetres','compartimentation']]
```

- Data was splitted with 0.3 testing having the following shape:

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((31510, 6), (13505, 6), (31510,), (13505,))
```

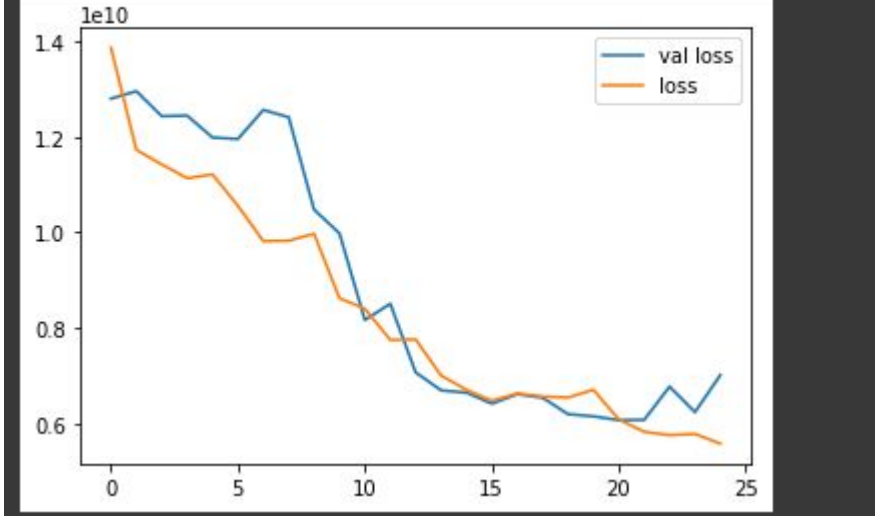
- initial results:

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Elastic Net Regression	45909.494407	1.393692e+10	118054.748756	19.226309	-0.197257
1	Artficial Neural Network	77615.584913	2.288715e+10	151284.983071	-32.646150	0.000000
2	Polynomail Regression	47169.806022	1.249788e+10	111793.897005	27.566545	0.000000
3	Robust Regression	49702.800140	3.629103e+11	602420.376733	-2003.305290	-606.203894
4	Ridge Regression	46001.150420	1.392967e+10	118024.028941	19.268341	-0.121264

Experiments

- standardization applied
- we can observe that neural net presents a better r2 square, so we modified the architecture by adding neurons on layers to improve ANN model

R2 Square 0.5787609557024341



Layer (type)	Output Shape	Param #
dense_36 (Dense)	(None, 7)	56
dense_37 (Dense)	(None, 32)	256
dropout_16 (Dropout)	(None, 32)	0
dense_38 (Dense)	(None, 128)	4224
dropout_17 (Dropout)	(None, 128)	0
dense_39 (Dense)	(None, 256)	33024
dropout_18 (Dropout)	(None, 256)	0
dense_40 (Dense)	(None, 512)	131584
dropout_19 (Dropout)	(None, 512)	0
dense_41 (Dense)	(None, 256)	131328
dense_42 (Dense)	(None, 128)	32896
dense_43 (Dense)	(None, 32)	4128
dense_44 (Dense)	(None, 1)	33

E1 - Extending X with townId

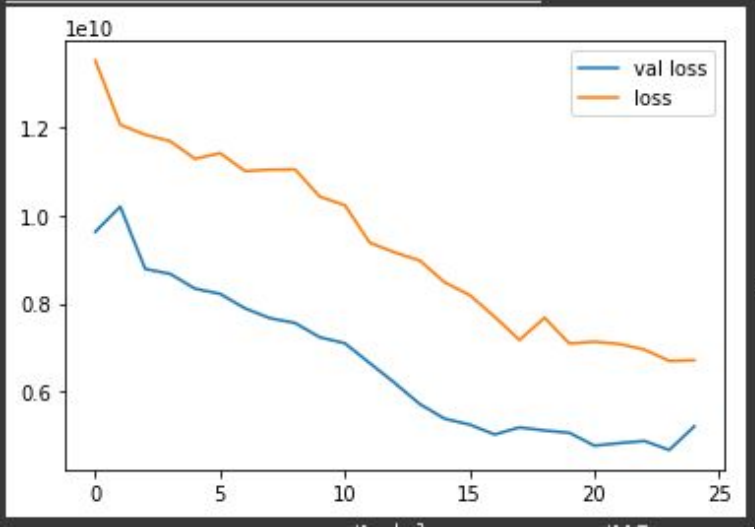
	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Elastic Net Regression	45187.434903	1.049403e+10	1.024404e+05	19.500061	-0.450123
1	Artficial Neural Network	35287.994277	5.751872e+09	7.584109e+04	55.877250	0.000000
2	Polynomail Regression	45800.998701	9.307971e+09	9.647783e+04	28.598320	0.000000
3	Robust Regression	64494.858070	2.881867e+12	1.697606e+06	-22006.877258	-529.215522
4	Ridge Regression	45289.856102	1.049735e+10	1.024566e+05	19.474547	-0.403236

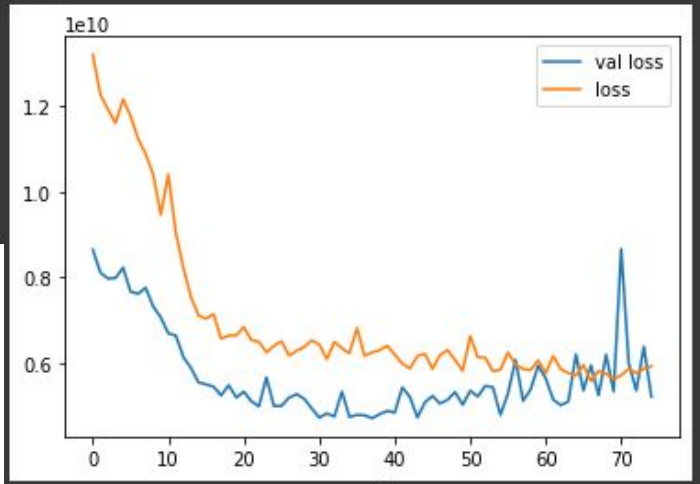
- we added 1 more feature, named townID. Starting from town as a string, we mapping using a world cities dataset that string with an ID
- because now the data are in range apart values, we transform X train and X test in standardized data
- changing standardization method from preprocessing.scale() into standardscaler

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Elastic Net Regression	44625.216482	8.600103e+09	9.273674e+04	18.766281	-0.450123
1	Artficial Neural Network	31793.649879	4.561492e+09	6.753882e+04	56.913656	0.000000
2	Polynomail Regression	43758.945013	8.198176e+09	9.054378e+04	22.562752	0.000000
3	Robust Regression	79593.785742	1.912251e+12	1.382842e+06	-17962.490676	-434.650979
4	Ridge Regression	44745.877849	8.609961e+09	9.278988e+04	18.673165	-0.403236

E2 - Changing optimizer with RMSprop and running with 75 epochs

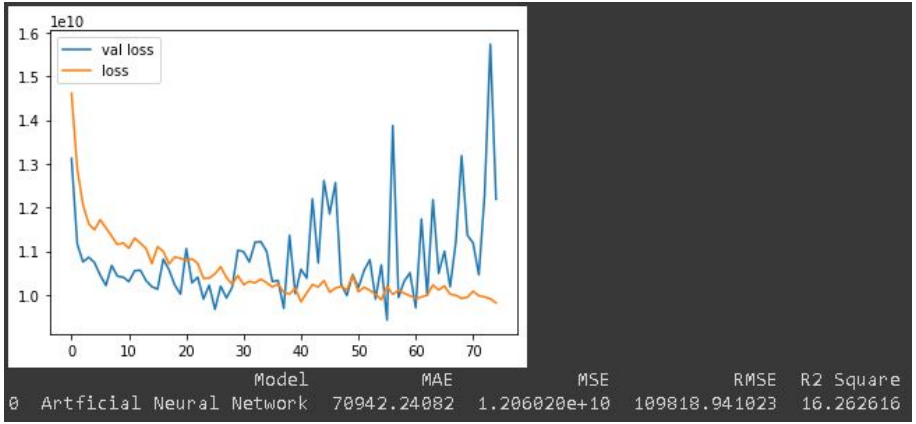
```
R2 Square 0.5515624738354833
```



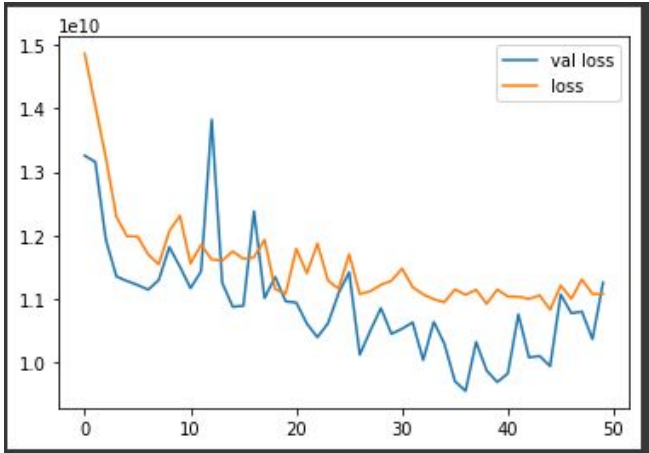


	Model	MAE	MSE	RMSE	R2 Squar
0	Artficial Neural Network	47397.820393	5.663762e+09	75257.967071	62.08082

E3 - Changing standardization method in MinMax and adding more dropout layers

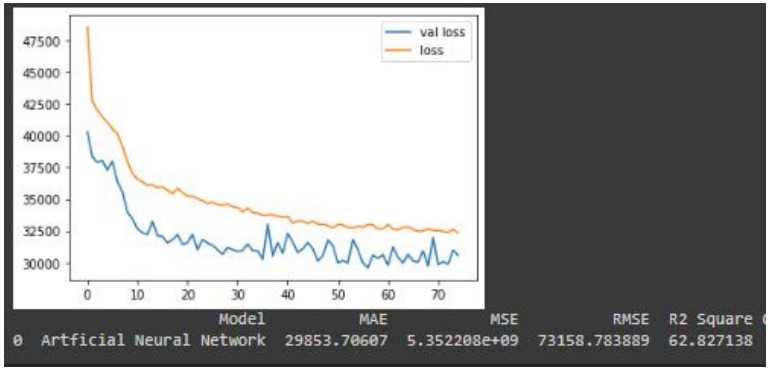


	Model	MAE	MSE	RMSE	R2 Square
0	Artficial Neural Network	70942.24082	1.206020e+10	109818.941023	16.262616

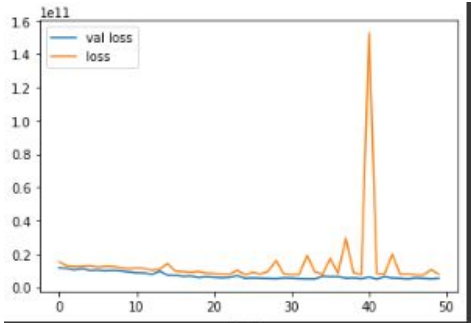


	Model	MAE	MSE	RMSE	R2 Square
0	Artficial Neural Network	56129.824112	1.146757e+10	107086.752031	20.762355

E4 - Adding more layers and neurons - 50 epochs - 100 epochs
Changing loss function with huber



	Model	MAE	MSE	RMSE	R2 Square
0	Artficial Neural Network	29853.70607	5.352208e+09	73158.783889	62.827138

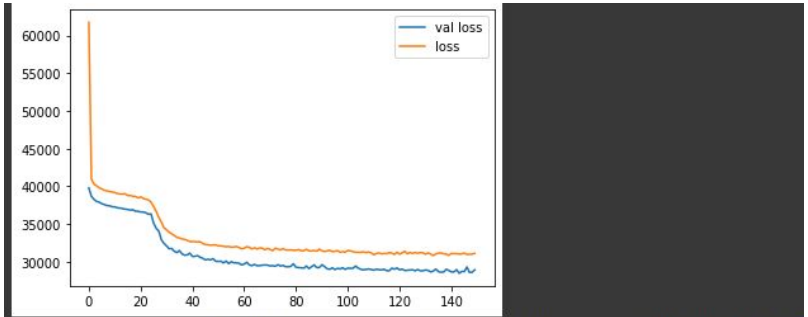


	Model	MAE	MSE	RMSE	R2 Square
0	Artficial Neural Network	35822.900261	5.512887e+09	74248.815019	61.858036

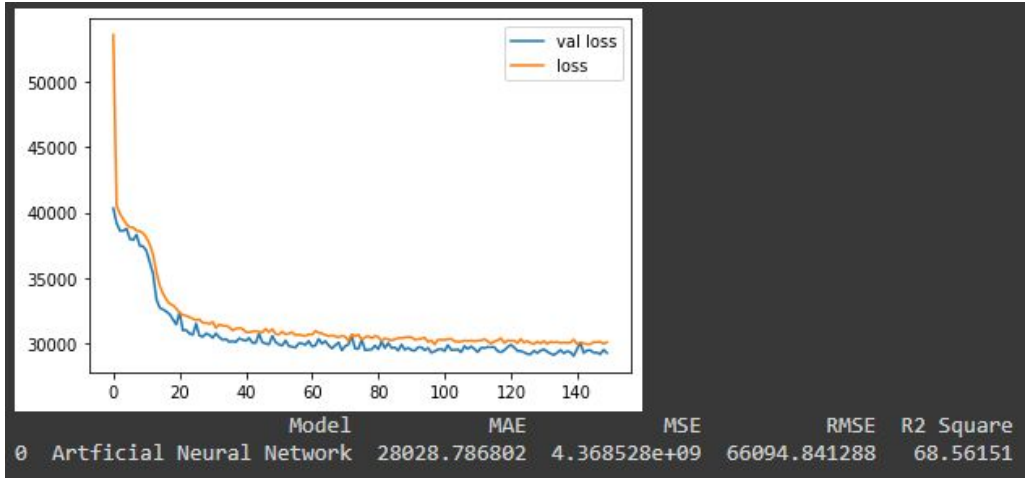
dense_39 (Dense)	(None, 7)	56
dense_40 (Dense)	(None, 64)	512
dropout_31 (Dropout)	(None, 64)	0
dense_41 (Dense)	(None, 128)	8320
dropout_32 (Dropout)	(None, 128)	0
dense_42 (Dense)	(None, 256)	33024
dropout_33 (Dropout)	(None, 256)	0
dense_43 (Dense)	(None, 512)	131584
dropout_34 (Dropout)	(None, 512)	0
dense_44 (Dense)	(None, 1024)	525312
dropout_35 (Dropout)	(None, 1024)	0
dense_45 (Dense)	(None, 2048)	2099200
dropout_36 (Dropout)	(None, 2048)	0
dense_46 (Dense)	(None, 512)	1049088
dropout_37 (Dropout)	(None, 512)	0
dense_47 (Dense)	(None, 256)	131328
dropout_38 (Dropout)	(None, 256)	0
dense_48 (Dense)	(None, 128)	32896
dropout_39 (Dropout)	(None, 128)	0
dense_49 (Dense)	(None, 64)	8256
dropout_40 (Dropout)	(None, 64)	0
dense_50 (Dense)	(None, 1)	65

E5 - Extending X with more features - NaN filling method - bsize 128 - 200 epochs

```
D[['townID','rooms','yearconstruction','confort','furnished','squaremetres','compartimentation','balcony','parkingplaces','bathrooms']]
```



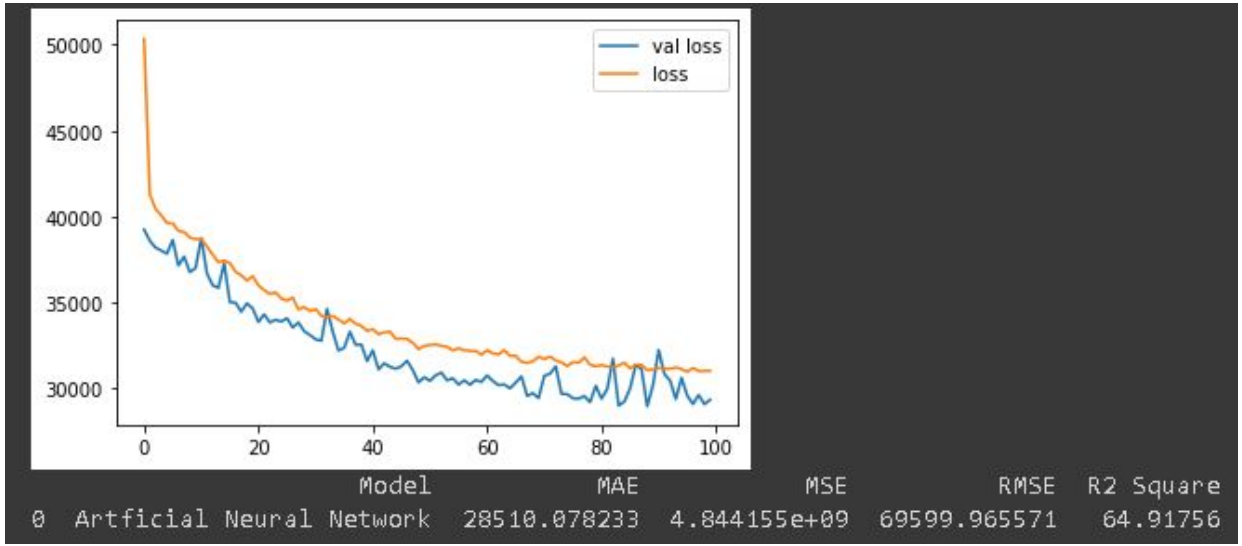
	Model	MAE	MSE	RMSE	R2 Square
0	Artficial Neural Network	29132.095374	4.774814e+09	69100.029956	66.563337



	Model	MAE	MSE	RMSE	R2 Square
0	Artficial Neural Network	28028.786802	4.368528e+09	66094.841288	68.56151

- using ffill() method for filling nan values - stands for 'forward fill' and will propagate last valid observation forward

It doesn't seem like a big difference between bfill and ffill for this problem



	Model	MAE	MSE	RMSE	R2 Square
0	Artficial Neural Network	28510.078233	4.844155e+09	69599.965571	64.91756

- using bfill() method for filling nan values - stand for backward fill' and will propagate last valid observation forward

Adding random forest regressor and gridsearch

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Elastic Net Regression	44625.216482	8.600103e+09	9.273674e+04	18.766281	-0.450123
1	Artficial Neural Network	31793.649879	4.561492e+09	6.753882e+04	56.913656	0.000000
2	Polynomail Regression	43758.945013	8.198176e+09	9.054378e+04	22.562752	0.000000
3	Robust Regression	79593.785742	1.912251e+12	1.382842e+06	-17962.490676	-434.650979
4	Ridge Regression	44745.877849	8.609961e+09	9.278988e+04	18.673165	-0.403236
5	Random Forest Regressor	15280.705051	1.460016e+09	3.821015e+04	86.209175	0.000000

```
Fitting 2 folds for each of 288 candidates, totalling 576 fits
{'bootstrap': True, 'max_depth': 110, 'max_features': 3, 'min_samples_leaf': 3, 'min_samples_split': 8, 'n_estimators': 100}
None
```

```
param_grid = {
'bootstrap': [True],
'max_depth': [80, 90, 100, 110],
'max_features': [2, 3],
'min_samples_leaf': [3, 4, 5],
'min_samples_split': [8, 10, 12],
'n_estimators': [100, 200, 300, 1000]
}
```

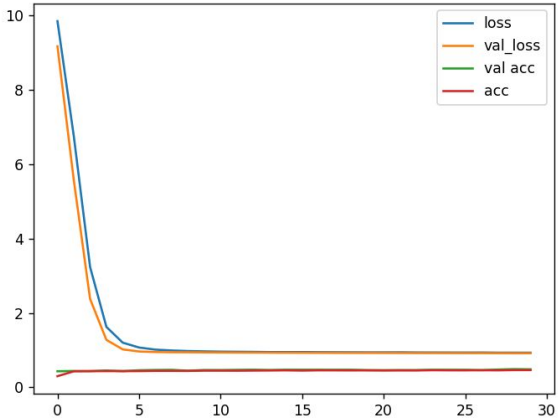
Classification problem for assets

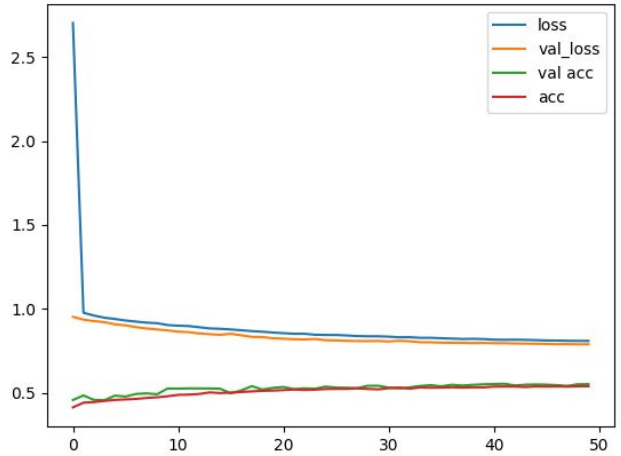
- using a labelling mechanism for samples, we experimented some supervised learning algorithms and a dense layers network architecture
- feature engineering on description and analyzing key words
- larger description are usually referring to luxury asset
- applying cosine similitudine, hardcoded keywords and variables

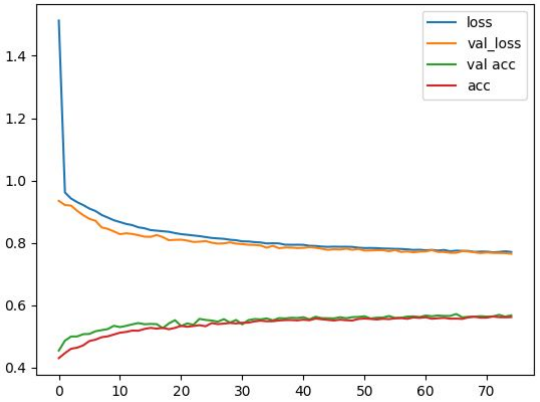
```
tags_luxury_assets = ['spatioasa','mobilata','nou','investitie','bloc nou','centru','utilata','modern','calitate']
tags_nonluxury_assets = ['margine','nemobilat']
```

```
    for tag in tags_luxury_assets:
      clf = CleaningDataSet(self.data_frame)
      tag_ = clf.text_to_vector(tag,0)
      desc_ = clf.text_to_vector(desc,1)
      cosine = clf.get_cosine(tag_, desc_)
      if cosine > 0.0:
        contor += 1

    num_feature = dataset_labeled.loc[dataset_labeled['description'] == desc, 'price'].iloc[0]

    if (contor >= len(tags_luxury_assets)/2) and (len(word_tokenize(desc)) > 150): #and (num_feature > 80000):
      dataset_labeled.loc[dataset_labeled['description'] == desc, 'label'] = "luxury_asset"
elif pd.isna(desc) == True:
      dataset_labeled.loc[dataset_labeled['description'] == desc, 'label'] = "nan"
```

Initial results. Adjusting lr 1e-5 and 1e-4 - adding more layers - decrease batch size








```
Epoch 11/35
943/943 [==============================] - 23s 25ms/step - loss: nan - accuracy: 0.4257 - val_loss: nan - val_accuracy: 0.4176
Epoch 12/35
943/943 [==============================] - 23s 25ms/step - loss: nan - accuracy: 0.4257 - val_loss: nan - val_accuracy: 0.4176
Epoch 13/35
943/943 [==============================] - 23s 24ms/step - loss: nan - accuracy: 0.4257 - val_loss: nan - val_accuracy: 0.4176
Epoch 14/35
943/943 [==============================] - 22s 23ms/step - loss: nan - accuracy: 0.4257 - val_loss: nan - val_accuracy: 0.4176
Epoch 15/35
943/943 [==============================] - 22s 23ms/step - loss: nan - accuracy: 0.4257 - val_loss: nan - val_accuracy: 0.4176
Epoch 16/35
943/943 [==============================] - 22s 23ms/step - loss: nan - accuracy: 0.4257 - val_loss: nan - val_accuracy: 0.4176
Epoch 17/35
943/943 [==============================] - 22s 23ms/step - loss: nan - accuracy: 0.4257 - val_loss: nan - val_accuracy: 0.4176
Epoch 18/35
943/943 [==============================] - 24s 25ms/step - loss: nan - accuracy: 0.4257 - val_loss: nan - val_accuracy: 0.4176
Epoch 19/35
943/943 [==============================] - 23s 25ms/step - loss: nan - accuracy: 0.4257 - val_loss: nan - val_accuracy: 0.4176
Epoch 20/35
943/943 [==============================] - 23s 24ms/step - loss: nan - accuracy: 0.4257 - val_loss: nan - val_accuracy: 0.4176
Epoch 21/35
```

Sklearn models

	precision	recall	f1-score	support
0	0.53	0.63	0.58	5747
1	0.62	0.53	0.57	6021
2	0.62	0.53	0.57	1737
accuracy			0.57	13505
macro avg	0.59	0.56	0.57	13505
weighted avg	0.58	0.57	0.57	13505

0.5731951129211403

	precision	recall	f1-score	support
0	0.53	0.67	0.59	5747
1	0.59	0.47	0.52	6021
2	0.68	0.57	0.62	1737
accuracy			0.57	13505
macro avg	0.60	0.57	0.58	13505
weighted avg	0.57	0.57	0.56	13505

- MlpClassifier and Knn

	precision	recall	f1-score	support
0	0.58	0.55	0.57	5767
1	0.59	0.64	0.62	5948
2	0.72	0.68	0.70	1790
accuracy			0.61	13505
macro avg	0.63	0.62	0.63	13505
weighted avg	0.61	0.61	0.60	13505

- random forest classifier with 750 estimators

Conclusion

For regression problem, we have the following :

- Standard Scaler was better than MinMaxScaler
- Bigger batch size was better than lower batch size
- Huber loss function was pretty better than MSE
- ffill was better than bfill
- rmsprop was better than adam

For both problems, filling nan values with previous entry imprease the accuracy and r2 square with at least 8% with the best architecture.

For classification problem, we have the following :

- Min Max Scaler was better than Standard Scaler
- Smaller batch size was better than higher batch size
- ffill was better than bfill