House prices – advanced regression techniques.

The goal of the project is to build a regression model, that can predict apartment prices in Almaty based on input data. For this purpose, we need firstly collect the data, preprocess and try some regression models.

Web-Scraping

As part of the project, 40,000 apartment sale advertisement were scraped from the krisha.kz website. For these purposes, the "Parser" class was written. Parser has 2 main methods:

1. crawl() – this method returns Generator, which provides in a loop url address of a single advertisement from krisha.kz corresponding to search criteria. The criteria are: city Almaty, type apartment;
2. scrape() – this method retrieves web page of provided url from crawl generator and transforms the content of the page into beautifulsoup4 structure. After that the method uses some help functions to parse the content of bs4 structure.

There are following features for each apartment:

1. rooms\_count – number of rooms in apartment (int)
2. mortgaged – is apartment mortgaged or not (bool)
3. building\_type – type of the house (кирпичный, панельный, иное, монолитный)
4. build\_year - year of construction of the house (int)
5. floor (int)
6. max\_floor – maximal floor in the building (int)
7. general\_area – general area of the apartment (float)
8. kitchen\_area – area of the kitchen (float)
9. living\_area – living space (float)
10. private\_hostel - is it in the hostel (bool)
11. price – price of apartment
12. residential\_complex - builder's name, if given
13. address: microdistinct, distinct, street, house\_number
14. images\_count – number of images provided by apartment’s owner
15. condition – good / average / needs repair / free layout / etc
16. many true/false columns like: telephone, internet, plastic\_windows, non angular / angular, rooms isolated/not isolated, etc
17. text – description of the advertisement
18. etc

Totally there are 56 distinct features, which were transformed after preprocessing into 474-dimensional vector.

To test the whole parsing process TestView class was written. This class tests separately crawl() and scrape() methods of the parser. Results of scraping were asserted to some truly pre scraped results, to ensure, that Parser works without errors. However, there were some mistakes. For example, I forgot to store url address of scraped advertisement and couldn’t therefore refer to source page.

The parser is executed from main.py script. The script parses the data in batches with size of 2000 advertisements. After parsing of a single batch, we ensure that scraped data is in true format, converting each column explicitly using numpy and save obtained batch to df\_{batch\_number}.csv file. After all data for given search request was parsed, we concatenate batches into single df.csv file.

Data wrangling and regression models.

For training and preprocessing NN, Preprocessor, Visualizer classes were written. The pipeline is executed from main.ipynb notebook.

Data preprocessing

Before training we need to resolve some problems:

1. Missing data. It was decided that if more than 15% of the data is missing, the column will be dropped. For less than 20% rows were dropped.
2. Data normalization. To resolve skewness of numerical values “Box cox transformation” was applied.
3. Missing values in category columns were replaced with string ’None’.
4. Category columns were encoded into Boolean columns using get\_dummies() method of pandas library. For example, district\_alatauskij, district\_Almalinskij, etc. They were also transliterated from Cyrillic to Latin alphabet, because some models didn’t accept Cyrillic column names.
5. Text column was transformed to length of text column.
6. Rows with price less than 100000 tenge were dropped.
7. Rows with false scraped building types were dropped.
8. After preprocessing the data was splited into x\_train, y\_train and x\_test, y\_test datasets in 80:20 ratio.

Following regression models were trained to predict price:

1. Kernel Ridge Regression (KRR)
2. Elastic Net
3. Lasso Regression
4. Gradient Boosting Regression
5. XGBoost
6. LightBoost
7. Averaged: KRR+Elastic Net + Lasso + Gradient Boosting

Each model was initialized before training in *\_\_init\_\_ ()* function of *NN* class. After data preprocessing, rmsle\_cv\_score(model\_name) and r2\_fit\_and\_score(model\_name) methods were called, which evaluate particular model on following validation metrics:

1. score from sklearn;
2. Mean Absolute Error from sklearn;
3. Mean Squared Error from sklearn;
4. Cross validation (negative mean squared error) + KFold from sklearn;

*r2\_fit\_and\_score()* method returns *r2\_score, mean\_absolute\_error, mean\_squared\_error* scores as well as *y\_pred* – prediction of the model for given test set. *rmsle\_cv\_score()* returns mean negative squared error and standard deviation across each training fold. The methods use predefined in each model *predict()* and *fit()* methods to train and evaluate results. *fit()* function accepts x\_train dataframe and it’s label Series. *predict()* method predicts on x\_test dataframe. After that a metric evaluates the results in comparison to *y\_test* label Series.

After prediction, to obtain the denormalized price values *reverse()* function was applied. The function turns back operations of logarithmic function and Box Cox transformation.

**Link to video recordings (P.S. better download the video, because google drive somehow compresses the video to low quality 360p in web player):**

<https://drive.google.com/file/d/16hj0fjnhLddyTXRLaI4kMJrVE5zjXEmW/view?usp=sharing>

**Link to dataset:**

<https://drive.google.com/file/d/1lCO3z-iplg-Pf_w3HXP8jm3yvHzQAzEI/view?usp=sharing>