SBERBANK RUSSIAN HOUSING MARKET PRICE PREDICTION

Salmaan Pehlari (011409307) Shrey Bhatt (011489777) Maitray Shah (011433279)

Section 1 Introduction

- **Motivation:**
- Housing costs demand a significant investment from both consumers and developers. And when it comes to planning a budget—whether personal or corporate—the last thing anyone needs is uncertainty about one of their biggest expenses. Sberbank helps their customers by making predictions about realty prices so renters, developers, and lenders are more confident when they sign a lease or purchase a building.

- **Objective:**
- In this project, Sberbank is challenging us to develop algorithms which use a broad spectrum of features to predict realty prices. We have a rich dataset that includes housing data and macroeconomic patterns. An accurate forecasting model will allow Sberbank to provide more certainty to their customers in an uncertain economy.
- There are multiple factors due to which the prices of the houses are affected. There are multiple direct factors such as how big is the house in terms of area, how many rooms does it have, which floor is the house on, etc. These are the factors that are considered in a narrow perspective.
- Overall the housing prices are also affected by the economic conditions of a country. For example, if we think about "inflation", if the inflation is increasing then the commodities are getting expensive and people's purchasing power is decreasing but that doesn't necessarily mean lower purchases as the people believe in buying physical assets. If we consider interest rates - then high interest rates don't help when people are trying to buy houses.
- Also, these conditions don't take place directly it maybe a chain reaction of multiple events which may result in an outcome. The general population doesn't buy house immediately for most of them it is generally a long process, so the actual decision to buy a house maybe 1-2 months from buying it.
- How realty prices work also depends on country to country, for example in countries such as Russia where land is a scarcity the prices balloons up when a lot of people start buying properties for investments and which could have a further impact on inflation and interest rates as well.

- **Literature/Market Review:**
- Although the housing market is relatively stable in Russia, the country's volatile economy makes forecasting prices as a function of apartment characteristics a unique challenge. Complex interactions between housing features such as number of bedrooms and location are enough to make pricing predictions complicated. Adding an unstable economy to the mix means Sberbank and their customers need more than simple regression models in their arsenal.

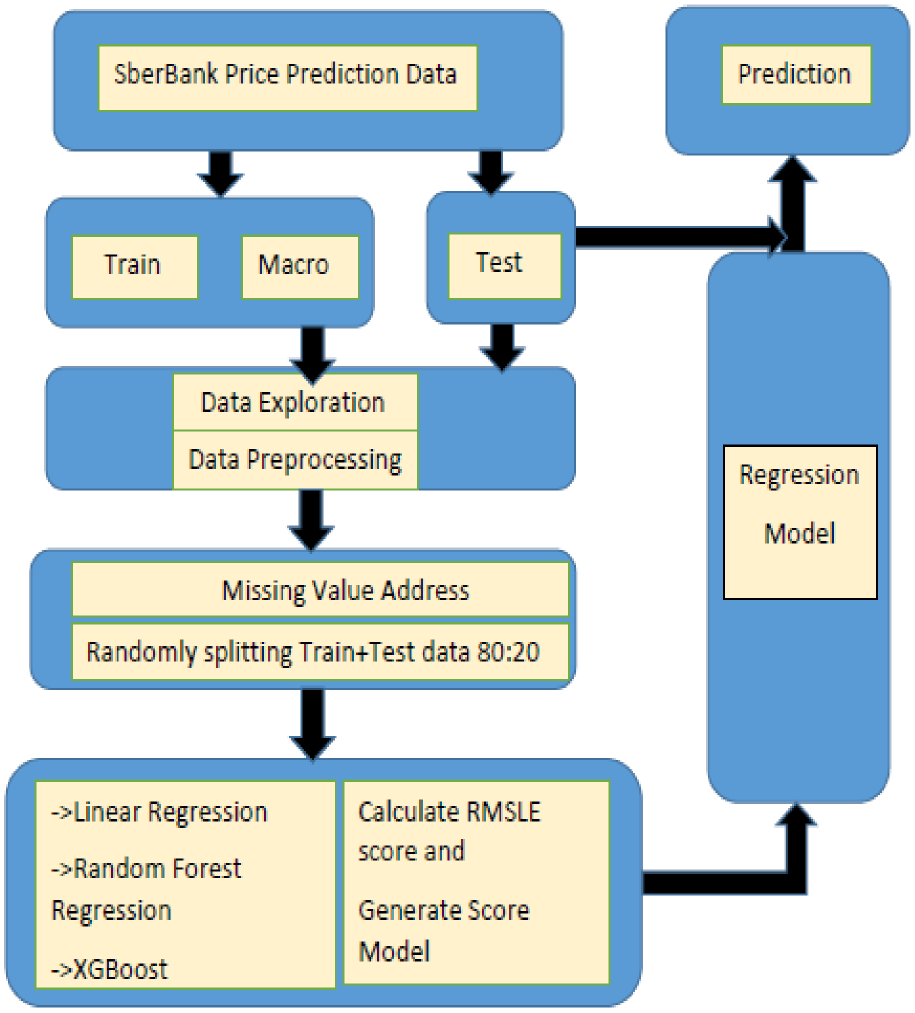
Section 2 System Design & Implementation details

- **Algorithms Selected:** Linear Regression, Random Forest Regression, XGBoost
- We started with the simplest approach of Linear Regression because it is always better to start with the simplest approach.

- After that we applied Logistic Regression but in that we got some errors due to floating point values.
- We did research on which other algorithms can be best suited for the price prediction and we came across Random Forest Regression. We applied that and that gave us better results than linear regression. Random forests will overfit unless we set ‘min_samples_leaf’ to a reasonable value, so we picked 50 relatively arbitrarily.
- We then learned that XGBoost will give the best performance for the price prediction for this data as it takes care of missing values and has cross validation inbuilt and applied that algorithm.
- We used all three algorithms with (macro+train) data and only train data without using macro to see the effects on the results.

- **Tools and Technologies Used:** python, jupyter notebook
  - Because we were familiar with python and its libraries and could code and debug efficiently in jupyter notebook.

- **System Architecture Design:**



- We had train data as well as macro data for analyzing and predicting the price_doc values.
- First, we did some data exploration so that we can get a better idea about the dataset.
- Then we pre-processed the data, removed columns having more of null values because those were not very helpful in making the right prediction, addressed the missing values by replacing them with mean/median values in some cases.
- Basically, we applied three algorithms for making our Regression Model;
  1) Linear Regression
  2) Random Forest Regression
  3) XGBoost Regression
- We calculated the RMSLE values for making our regression model.
- We applied our test data in the regression model to generate the scores of RMSLE values for the price_doc prediction in the test data.

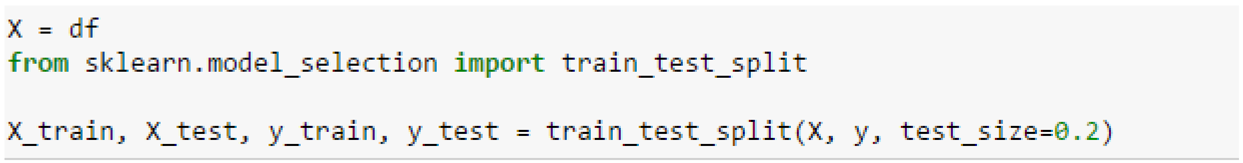
- **Use cases:**
  - Price Prediction
  - Market Rate Analysis
  - Budgeting and Planning
  - Mortgage Lending
  - Banking and Loans

Section 3 Experiments / Proof of concept evaluation

- **Dataset Used:** Sberbank Russian Housing Market
- **Source:** https://www.kaggle.com/c/sberbank-russian-housing-market/data
- **Size**: train.csv (45 MB), test.csv (11 MB), macro.csv (1.5 MB)
- **Dimensions of data**: train data (30471*292), test data (7662*291), macro (2484*100)

- **Preprocessing performed:**

  - Merged train data and macro on timestamp using pd.merge() function of pandas with parameter how='left' for prediction with macro+train data.
  - Now the train data has dimensions of 30471*391.
  - Extracted target column 'price_doc' from the train data and stored log of all price_doc values.
  - Removed columns having more than 20% missing values.
  - Removed id and price_doc columns from training data.
  - Now the combined data has dimensions of 30471*367.
  - Converted the timestamp values to numeric using pd.to_numeric() from pandas.
  - Now, from the combined data we only dummies object values using pd.get_dummies() .
  - Now the train data has dimensions of 30471*536.
  - Split training and testing data (log value of price_doc) with selecting test_size=0.2 in train_test_split function of sklearn.model_selection.
  - Addressed missing data with mean values in Imputer() and making pipeline with StandardScaler() and transforming the train data splitted in the train_test_split() operation as shown below:



```
X = df
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

- **Methodology followed:**

1) Data Exploration:

- We noticed that the data contained missing values which may be an issue, when we apply regression models.
- We were predicting price_doc column for the test dataset so first for understanding the house price changing pattern we plotted a histogram of the price_doc from the training data.
- Using the prices directly in linear regression gave us negative values in some cases. This was because linear regression doesn't respect the bounds of 0, and could predict negative values.
- Then we applied log function to price_doc to remove skewness and to have a uniform distribution.
- We have also listed out correlation of other features with price_doc to understand and explore the dataset and relations among features.
- We also checked for the null values in the data and we found that 2.05% of columns have more than 33% missing values.
- We tried to plot the number of rooms as it was the most important feature in determining the price of the house.
- Then we tried plotting the median of house prices and year-month values, there have been changes in median prices with respect to time but found nothing dramatic or a specific pattern.
- Then we tried plotting floor number for all the different houses and the relation between floor number and median price_doc so that we can get more idea of co-relation between floor numbers and price values
- The macro data consists different economic factors. These factors have a complex relationship with housing prices. For example, if we take the inflation - when inflation increases people have lesser money to spend. Also, if we see the feature salary growth which indicates how much more or less people are earning. These kinds of factors come into play when predicting the house prices.
- Then we tried showing the general trends for USD->RUB over time, the general trends for Consumer Price Index over time, the general trends for Salary Growth over time and the general trends for Mortgage Rate over time to understand the macro data

2) Evaluation Metric:

- Since the evaluation metric is RMSLE as it was suitable for this kind of dataset we used that as our internal evaluation metric
- RMSLE (Root Mean Squared Logarithmic Error) is defined as;

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

Where:

ϵ is the RMSLE value (score)

n is the total number of observations in the (public/private) data set,

pi is your prediction,

ai is the actual response for ii.

log(x) is the natural logarithm of x

We defined the same RMSLE function for using it as our internal evaluation metric

```python
def rmsle_exp(y_true_log, y_pred_log):
    y_true = np.exp(y_true_log)
    y_pred = np.exp(y_pred_log)
    return np.sqrt(np.mean(np.power(np.log(y_true + 1) - np.log(y_pred + 1), 2)))
```

3) Score Model:

- We have addressed missing data with mean values in Imputer() and making pipeline with StandardScaler() and transforming the train data split in the train_test_split() operation.

```python
from sklearn.preprocessing import Imputer, StandardScaler
from sklearn.pipeline import make_pipeline

# Make a pipeline that transforms X
pipe= make_pipeline(Imputer(strategy="median"), StandardScaler())
pipe.fit(X_train)
pipe.transform(X_train)
```

- Now we have generated our score model by defining the train_error and test_error as the RMSLE value by giving actual data in true values and test data in prediction values in the above RMSLE calculation function.

```python
def score_model(model, pipe):
    train_error = rmsle_exp(y_train, model.predict(pipe.transform(X_train)))
    test_error = rmsle_exp(y_test, model.predict(pipe.transform(X_test)))
    return train_error, test_error
```

4) Linear Regression:

- We started solving the price prediction problem using the simplest approach of linear regression because it is always better to start with the simplest approach and trying improving the performance after the baseline.
- We fitted the transformed data of X_train with the pipe with y-train for generating out Linear Regression Model.
- We used our Linear Regression Model and pipe in score model parameters to get the RMSLE value in train and test data.
- In Linear Regression, we got RMSLE score in the range of 0.50
- We also had some issues with linear regression returning very small values sometimes which when exponentiated gave a very large value i.e. infinite.

5) Random Forest Regression:

- As we go on solving the prediction problem with approaches that can give better results we applied RandomForestRegressor from sklearn.ensemble we used number of trees in the forest (n_estimators) as 50.
- Random forests will overfit unless we set 'min_samples_leaf' to a reasonable value, so we picked 50 relatively arbitrarily.
- We fitted the transformed data of X_train with the pipe with y-train for generating out Random Forest Regression Model.
- We used our Random Forest Regression Model and piped in score model parameters to get the RMSLE value in train and test data.
- In Random Forest Regression, we got RMSLE score in the range of 0.45
- So, Random Forest Regressor improved the performance than the Linear Regression.

6) XGBoost Regression:

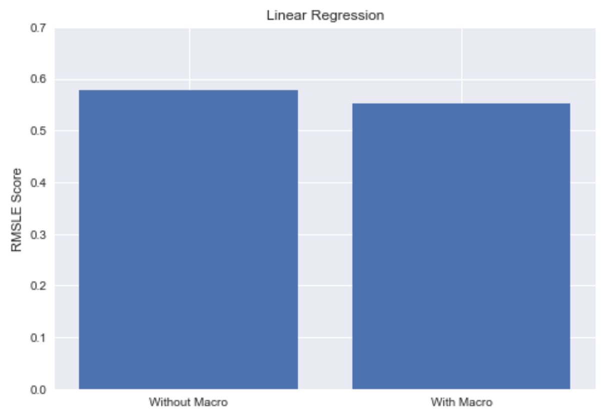
- XGBoost (eXtreme Gradient Boosting) is an advanced implementation of gradient boosting algorithm.
- XGBoost would be a good regression model to be used because - it is extremely fast; it can handle the missing values in our dataset and it has built-in cross validation. So, these are the things that are extremely important which is handled by this library.
- It's a highly sophisticated algorithm, powerful enough to deal with all sorts of irregularities of data like missing values.

- **Comparison of Evaluation graphs:**

1) Linear Regression:

We didn’t get much difference in RMSLE score with using macro and without using macro.

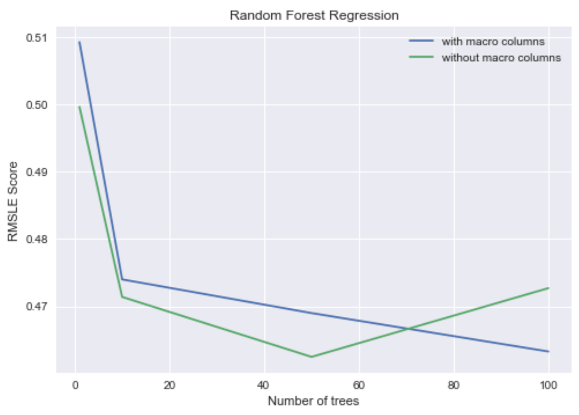
Basically, Linear Regression provide baseline score (0.50 RMSLE Score) in solving this problem.



2) Random Forest Regression:

We got better RMSLE score in Random Forest Regression than Linear Regression

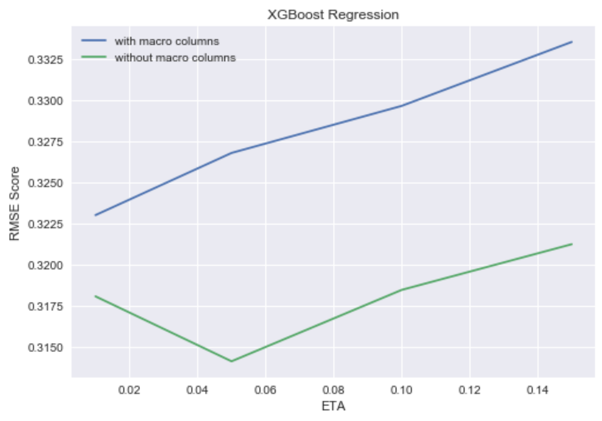
The RMSLE error reduced around 5% with random forest regression.



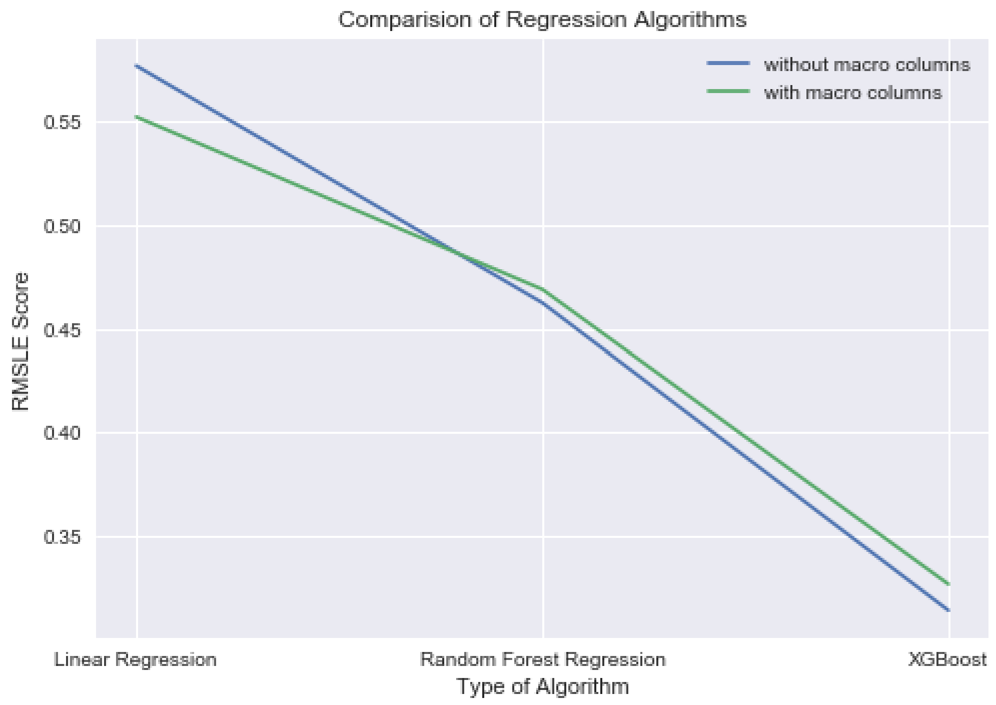
3) XGBoost Regression:

XGBoost is the algorithm in which we got the best performance due to in-built missing value handling and cross validation support.

We got RMSE score in the range of 0.31 – 0.32 using XGBoost.



4) Comparison of all three Algorithms:



- **Analysis of results:**
- With the help of results and the evaluation graphs we can say that the Linear Regression can provide baseline for solving this kind of prediction problems.
- Random Forest Regression, on top of the linear regression can perform slightly better than Linear Regression.
- XGBoost (eXtreme Gradient Boost) performs way better than the Linear Regression and the Random Forest Regression in both the cases i.e with macro and without macro.

Section 4 Discussion & Conclusions

- **Decisions made:**
- The data had many NaN values as well as missing values. So, preprocessing of the data was important. We decided to remove the columns having more than 33% missing values and we tried to dummy object values to change the categorical columns into dummy columns to remove string values.
- We selected the logarithm of the price_doc values from the train data b to remove skewness and to have a uniform distribution.
- We started with basic approach of Linear Regression and we got 50% error rate with overflow in some of the cases leading to infinite error values.

- Linear Regression was giving negative price prediction values initially, but then we took the logarithm of the price_doc values which solved the problem.
- Then we applied Random Forest Regression with n_estimators ranging from 1 to 100. Random forests will overfit unless we set 'min_samples_leaf' to 50 relatively arbitrarily.
- We learn that XGBoost can give even better performance than Linear Regression and Random Forest Regression.

- **Difficulties faced:**
- Linear Regression was giving negative price prediction values by directly taking price values from training data (without taking log).
- The data had many NaN values as well as missing values. So, preprocessing of the data was important.
- Got the infinite error because of overflow in calculating logarithm in some of the cases.

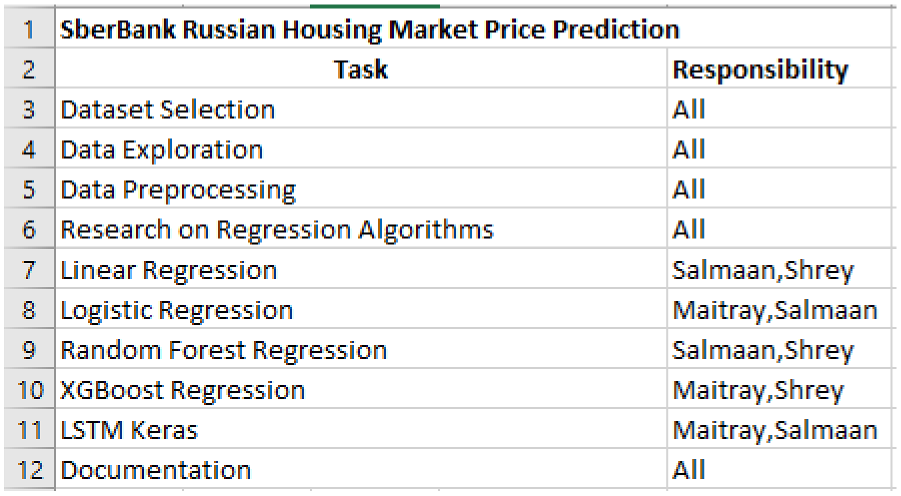
- **Things that worked:**
- Linear regression gave the baseline for solving the problem
- Random Forest Regression worked better by selecting 50 trees in the forest with min_samples_leaf' =50
- XGBoost performed best and was extremely fast as it can handle the missing values in our dataset and it has built-in cross validation.

- **Things that didn't work well:**

- The actual value of price_doc didn't work well as it gave negative predictions.
- Sometimes the Linear Regression generates very small prediction value which when exponentiated gives infinite value so RMSLE will be infinite in that case.
- We also tried Logistic Regression but it didn't work due to floating point values.
- We tried LSTM but it also didn't work well for this data due to data processing issue and time constraints.

- **Conclusion:**
- Firstly, we tried to explore the dataset and tried to establish relations between features
- We learned and implemented different regression algorithms after performing the necessary preprocessing required like Linear Regression, Random Forest Regression, Logistic Regression, and ultimately XGBoost with and without using the macro data.
- Through this we conclude that XGBoost was the best algorithm to predict price values in this kind of dataset and it was very fast and efficient along with the advantages of built-in cross validation as well as built-in missing values handling.

Section 5 Project Plan / Task Distribution



1	**SberBank Russian Housing Market Price Prediction**	
2	**Task**	**Responsibility**
3	Dataset Selection	All
4	Data Exploration	All
5	Data Preprocessing	All
6	Research on Regression Algorithms	All
7	Linear Regression	Salmaan,Shrey
8	Logistic Regression	Maitray,Salmaan
9	Random Forest Regression	Salmaan,Shrey
10	XGBoost Regression	Maitray,Shrey
11	LSTM Keras	Maitray,Salmaan
12	Documentation	All

References

1) https://www.kaggle.com/c/sberbank-russian-housing-market
2) http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
3) http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
4) https://github.com/dmlc/xgboost
5) http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
6) http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Imputer.html
7) http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
8) http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
9) http://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html
10) https://keras.io/layers/recurrent/
11) https://www.kaggle.com/wiki/RootMeanSquaredLogarithmicError
12) http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html