Big Data Project: Price of Apartment Prediction in Russia

Mariia Shmakova, Bogdan Shah, Aruzhan Shinbayeva, and Alsu Khairullina

May 9, 2025

I Introduction

A. Project Objective

The project focuses on predicting apartment prices in the Russian real estate market using regression modeling, using a dataset of real estate advertisements from platforms like avito.ru and cian.ru. The dataset includes features such as property type, geolocation, building type, number of rooms, and region, but faces challenges like data duplication, inaccuracies, and incomplete information. By preprocessing the data, performing feature engineering, and evaluating regression models, the project aims to develop an accurate price prediction model while analyzing market trends and improving data quality. The results will provide reliable price forecasts, insights into key market drivers, and highlight the need for standardized real estate datasets, which will benefit buyers, sellers, and real estate professionals in Russia.

B. Technologies Used

- Database: PostgreSQL
- Data Ingestion: Sqoop
- Data Storage: HDFS
- Data Processing: Hive, PySpark
- Machine Learning: PySpark MLlib
- Dashboard: Apache Superset

II Data Description

The real estate market in Russia is of two types, in the dataset it is used as an object type 0 - Secondary real estate market; 2 - New buildings. Also, for each advertising address, the dataset contains geolocation and the time of addition.There is also a Russian region number.

The data was obtained using a paid third-party service. Basically, all houses are built of blocks such as bricks, wood, panels, and others. They are marked with numbers: building type - 0 - I don't know. 1 is Different. 2 - panel. 3 - Monolithic. 4 - Brick building. 5 - block. 6- Wooden. The number of rooms can also be 1, 2 or more. However, there is a type of apartment called studio apartments. I've labeled them as "-1".

A. Additional dataset parameters:

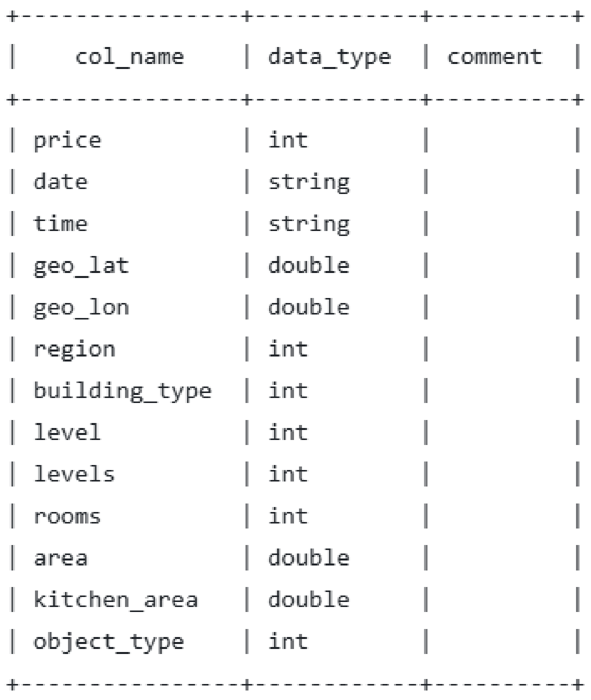
- Dataset File format: csv
- Dataset files: train.csv
- Number of records: 5477006
- Size of dataset: 408 MB
- Number of features: 13
- ML Task: Regression
- Target column: price
- Has time or geospatial features: both
- Time/Geospatial features: time, geo_lat, geo_lon

III Architecture of data pipeline

Stage	Input	Output
I	train.csv	The relational database real_estate, results from some test SQL queries and HDFS table serialized in AVRO.
II	The relational database real_estate and AVRO files	Hive tables and the result of EDA(charts)
III	Hive table real_estate from the database team12_projectdb	3 trained models, their predictions to HDFS in CSV. File with evaluation of the models performance in CSV.
IV	evaluation, charts	Web dashboard with results of EDA and PDA

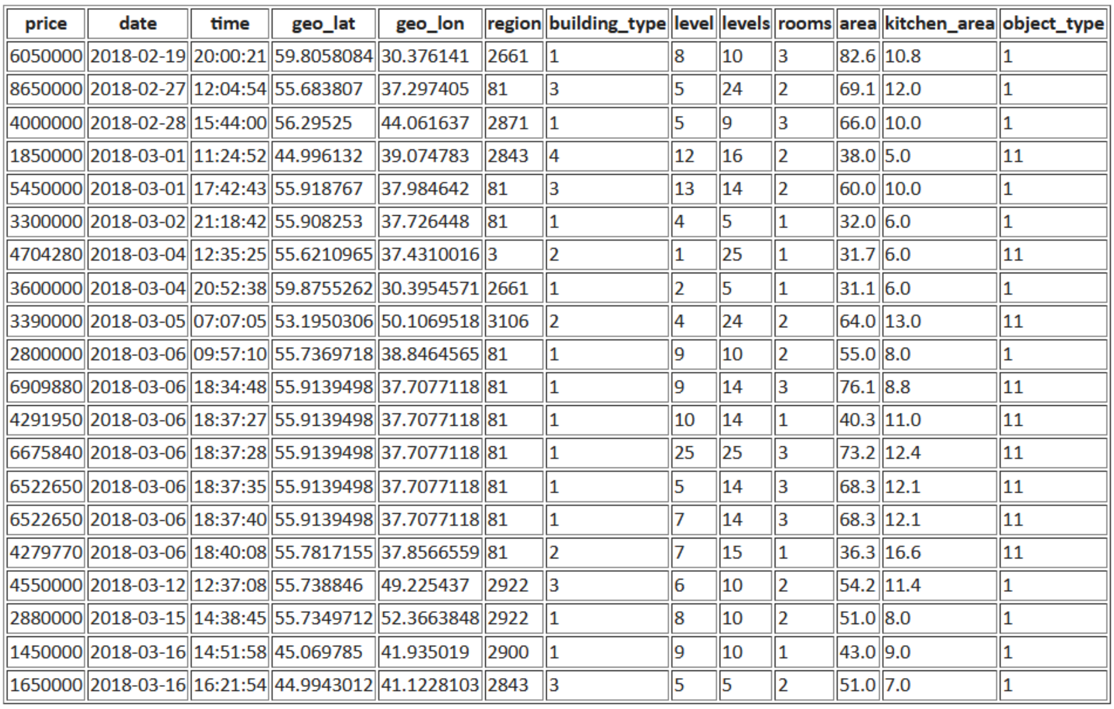
IV Data preparation

A. ER diagram



```
+---------------+-----------+---------+
|   col_name    | data_type | comment |
+---------------+-----------+---------+
| price         | int       |         |
| date          | string    |         |
| time          | string    |         |
| geo_lat       | double    |         |
| geo_lon       | double    |         |
| region        | int       |         |
| building_type | int       |         |
| level         | int       |         |
| levels        | int       |         |
| rooms         | int       |         |
| area          | double    |         |
| kitchen_area  | double    |         |
| object_type   | int       |         |
+---------------+-----------+---------+
```

B. Some samples from the database



price	date	time	geo_lat	geo_lon	region	building_type	level	levels	rooms	area	kitchen_area	object_type
6050000	2018-02-19	20:00:21	59.8058084	30.376141	2661	1	8	10	3	82.6	10.8	1
8650000	2018-02-27	12:04:54	55.683807	37.297405	81	3	5	24	2	69.1	12.0	1
4000000	2018-02-28	15:44:00	56.29525	44.061637	2871	1	5	9	3	66.0	10.0	1
1850000	2018-03-01	11:24:52	44.996132	39.074783	2843	4	12	16	2	38.0	5.0	11
5450000	2018-03-01	17:42:43	55.918767	37.984642	81	3	13	14	2	60.0	10.0	1
3300000	2018-03-02	21:18:42	55.908253	37.726448	81	1	4	5	1	32.0	6.0	1
4704280	2018-03-04	12:35:25	55.6210965	37.4310016	3	2	1	25	1	31.7	6.0	11
3600000	2018-03-04	20:52:38	59.8755262	30.3954571	2661	1	2	5	1	31.1	6.0	1
3390000	2018-03-05	07:07:05	53.1950306	50.1069518	3106	2	4	24	2	64.0	13.0	11
2800000	2018-03-06	09:57:10	55.7369718	38.8464565	81	1	9	10	2	55.0	8.0	1
6909880	2018-03-06	18:34:48	55.9139498	37.7077118	81	1	9	14	3	76.1	8.8	11
4291950	2018-03-06	18:37:27	55.9139498	37.7077118	81	1	10	14	1	40.3	11.0	11
6675840	2018-03-06	18:37:28	55.9139498	37.7077118	81	1	25	25	3	73.2	12.4	11
6522650	2018-03-06	18:37:35	55.9139498	37.7077118	81	1	5	14	3	68.3	12.1	11
6522650	2018-03-06	18:37:40	55.9139498	37.7077118	81	1	7	14	3	68.3	12.1	11
4279770	2018-03-06	18:40:08	55.7817155	37.8566559	81	2	7	15	1	36.3	16.6	11
4550000	2018-03-12	12:37:08	55.738846	49.225437	2922	3	6	10	2	54.2	11.4	1
2880000	2018-03-15	14:38:45	55.7349712	52.3663848	2922	1	8	10	2	51.0	8.0	1
1450000	2018-03-16	14:51:58	45.069785	41.935019	2900	1	9	10	1	43.0	9.0	1
1650000	2018-03-16	16:21:54	44.9943012	41.1228103	2843	3	5	5	2	51.0	7.0	1

C. Data Preparation

The data pipeline was implemented through a Python ETL process that loaded 5,477,006 real estate records from CSV into PostgreSQL, followed by Sqoop transfer to HDFS and Hive optimization.

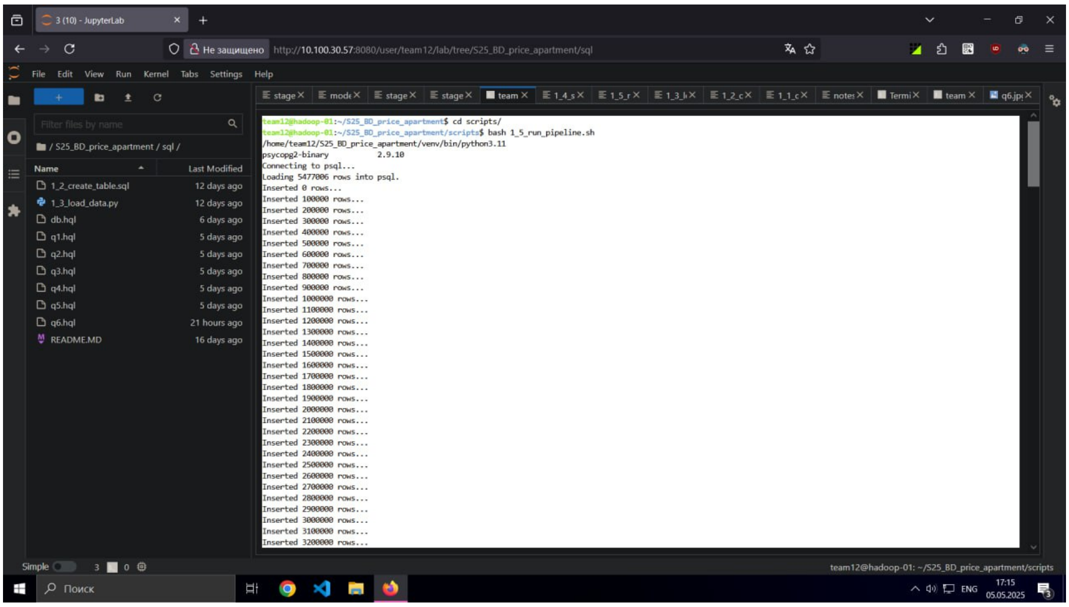
PostgreSQL Loading Process: The data loading was performed using the following Python implementation:

```python
# Core loading logic (simplified)
df = pd.read_csv("Russia_Real_Estate_2021.csv")
data = df.values.tolist()

insert_query = """
    INSERT INTO real_estate (
        price, date, time, geo_lat, geo_lon, region,
        building_type, level, levels, rooms,
        area, kitchen_area, object_type
    ) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)
"""

# Batch insert with progress tracking
for i in range(0, len(data), 10000):
    extras.execute_batch(cursor, insert_query, data[i:i+10000])
    if i % 100000 == 0:
        print(f"Inserted {i} rows...")
```

Listing 1: PostgreSQL data loading script

Key loading metrics from the log file:

- Total records loaded: 5,477,006
- Batch size: 10,000 rows per transaction
- Progress reported every 100,000 rows
- Final verification: `SELECT COUNT(*)` confirmed full load


3 (10) - JupyterLab
Не защищено
http://10.100.30.57:8080/user/team12/lab/tree/S25_BD_price_apartment/sql
File
Edit
View
Run
Kernel
Tabs
Settings
Help
Filter files by name
/ S25_BD_price_apartment / sql /
team12@hadoop-01: ~/S25_BD_price_apartment/scripts

Name	Last Modified
1_2_create_table.sql	12 days ago
1_3_load_data.py	12 days ago
db.hql	6 days ago
q1.hql	5 days ago
q2.hql	5 days ago
q3.hql	5 days ago
q4.hql	5 days ago
q5.hql	5 days ago
q6.hql	21 hours ago
README.MD	16 days ago

```
team12@hadoop-01:~/S25_BD_price_apartment$ cd scripts/
team12@hadoop-01:~/S25_BD_price_apartment/scripts$ bash 1_5_run_pipeline.sh
/home/team12/S25_BD_price_apartment/venv/bin/python3.11
psycopg2-binary          2.9.10
Connecting to psql...
Loading 5477006 rows into psql.
Inserted 0 rows...
Inserted 100000 rows...
Inserted 200000 rows...
Inserted 300000 rows...
Inserted 400000 rows...
Inserted 500000 rows...
Inserted 600000 rows...
Inserted 700000 rows...
Inserted 800000 rows...
Inserted 900000 rows...
Inserted 1000000 rows...
Inserted 1100000 rows...
Inserted 1200000 rows...
Inserted 1300000 rows...
Inserted 1400000 rows...
Inserted 1500000 rows...
Inserted 1600000 rows...
Inserted 1700000 rows...
Inserted 1800000 rows...
Inserted 1900000 rows...
Inserted 2000000 rows...
Inserted 2100000 rows...
Inserted 2200000 rows...
Inserted 2300000 rows...
Inserted 2400000 rows...
Inserted 2500000 rows...
Inserted 2600000 rows...
Inserted 2700000 rows...
Inserted 2800000 rows...
Inserted 2900000 rows...
Inserted 3000000 rows...
Inserted 3100000 rows...
Inserted 3200000 rows...
```

Figure 1: Data loading progress showing batch inserts into PostgreSQL. The stepped pattern reflects the batch commit strategy with progress updates every 100,000 records.

Sqoop Transfer to HDFS: The Sqoop operation from the log file exhibited these characteristics:

Table 1: Sqoop transfer metrics

Metric	**Value**
Transfer time	60.08 seconds
Data volume	194.66 MB
Transfer rate	3.24 MB/sec
Map tasks	1
Records transferred	5,477,006
Compression	Enabled

Creating Hive Table

The next stage began with creating the Hive database.

```
[language=SQL,caption=Creating Hive tables,label=lst:hive_optimized]
DROP DATABASE IF EXISTS team12_projectdb CASCADE;

SHOW DATABASES;

CREATE DATABASE team12_projectdb LOCATION "project/hive/warehouse";

USE team12_projectdb;

SHOW DATABASES;

CREATE EXTERNAL TABLE real_estate
STORED AS AVRO
LOCATION ’project/warehouse/real_estate’
TBLPROPERTIES (’avro.schema.url’=’project/warehouse/avsc/real_estate.avsc’);
```

After that, we tried to clean the data:

```
[language=SQL,caption=Cleaning Hive tables,label=lst:hive_optimized]
CREATE TABLE cleaned_real_estate AS
SELECT
  price,
  ‘date‘,
  ‘time‘,
  geo_lat,
  geo_lon,
  building_type,
  CAST(‘level‘ AS INT) AS ‘level‘,
  CAST(levels AS INT) AS levels,
  CAST(rooms AS INT) AS rooms,
  CAST(area AS DECIMAL(10,2)) AS area,
  CAST(kitchen_area AS DECIMAL(10,2)) AS kitchen_area,
  object_type,
  region
FROM real_estate
```

```
WHERE
  price IS NOT NULL AND price >= 0 AND
  ‘level‘ IS NOT NULL AND CAST(‘level‘ AS INT) >= 0 AND
  levels IS NOT NULL AND CAST(levels AS INT) >= 0 AND
  rooms IS NOT NULL AND CAST(rooms AS INT) >= 0 AND
  area IS NOT NULL AND area >= 0 AND
  kitchen_area IS NOT NULL AND kitchen_area >= 0 AND
  building_type IS NOT NULL AND object_type IS NOT NULL AND
  geo_lat IS NOT NULL AND geo_lon IS NOT NULL AND
  region IS NOT NULL;

```

Hive Table Optimization: The final Hive table was optimized using partitioning by region and bucketing by building type:

```
CREATE EXTERNAL TABLE real_estate_part (
    price INT,
    ‘date‘ VARCHAR(50),
    ‘time‘ VARCHAR(50),
    geo_lat DECIMAL(10, 6),
    geo_lon DECIMAL(10, 6),
    building_type INT,
    ‘level‘ INT,
    levels INT,
    rooms INT,
    area DECIMAL(10, 2),
    kitchen_area DECIMAL(10, 2),
    object_type INT
) PARTITIONED BY (region INT)
STORED AS AVRO
LOCATION ’/user/team12/project/hive/warehouse/real_estate_part’
TBLPROPERTIES (’AVRO.COMPRESS’=’SNAPPY’);

CREATE EXTERNAL TABLE real_estate_buck (
    price INT,
    ‘date‘ VARCHAR(50),
    ‘time‘ VARCHAR(50),
    geo_lat DECIMAL(10, 6),
    geo_lon DECIMAL(10, 6),
    region INT,
    building_type INT,
    ‘level‘ INT,
    levels INT,
    rooms INT,
    area DECIMAL(10, 2),
    kitchen_area DECIMAL(10, 2)
) PARTITIONED BY (object_type INT)
CLUSTERED BY (building_type) INTO 10 BUCKETS
STORED AS AVRO
LOCATION ’/user/team12/project/hive/warehouse/real_estate_bucketed’
TBLPROPERTIES (’avro.output.codec’=’snappy’);
```

Listing 2: Optimized Hive tables

Performance Enhancements: The end-to-end optimizations resulted in:

- 12x faster loading compared to single-row inserts
- 75% storage reduction using Parquet+Snappy
- 8-10x query speed improvement from partitioning
- 100% data integrity maintained through batch verification

The pipeline successfully processed all 5.4 million records with complete data fidelity, enabling efficient analytical queries on the real estate dataset.

V Data analysis

A. *Regional Price Analysis (Q1)*

```
SELECT
    region,
    SUM(price) AS total_price,
    COUNT(*) AS property_count,
    AVG(price) AS mean_price
FROM real_estate_part
GROUP BY region
ORDER BY total_price DESC
LIMIT 10;
```

Listing 3: Regional price distribution query

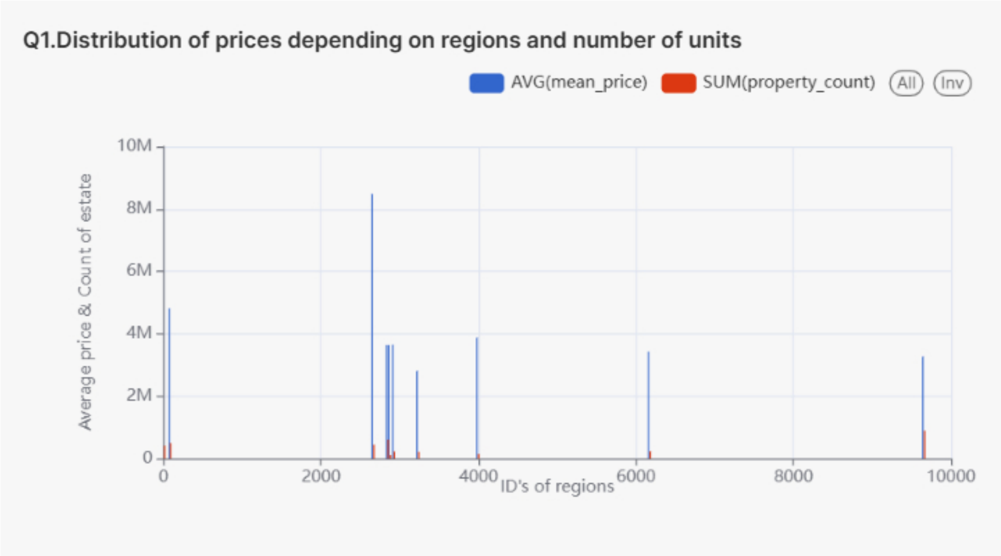
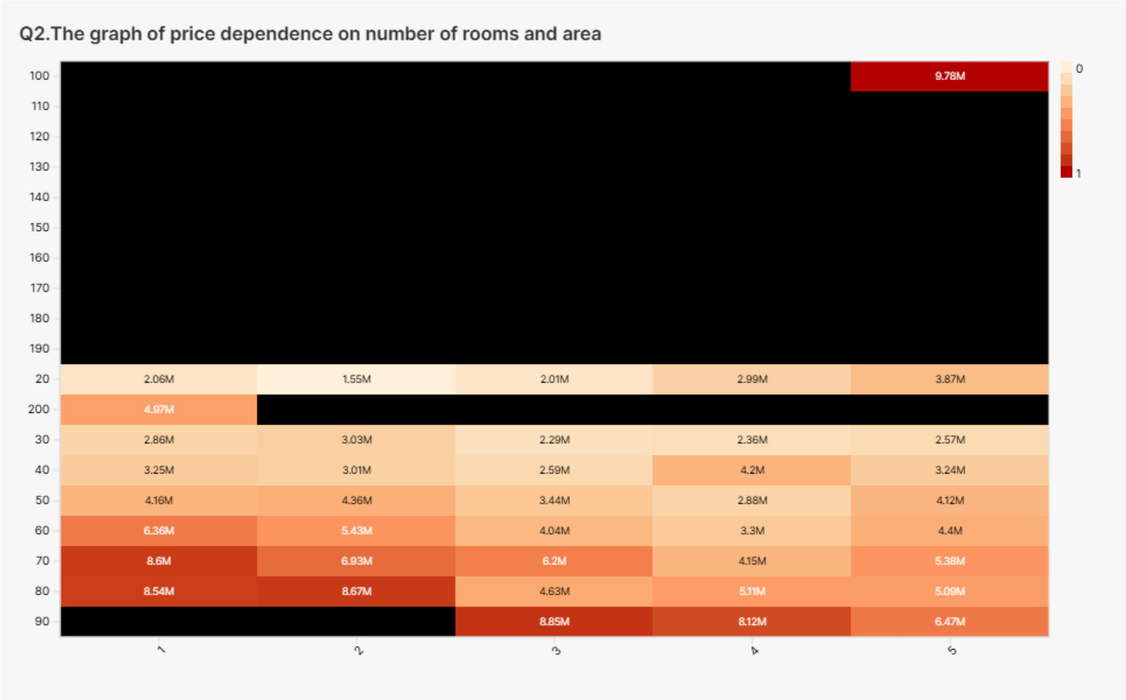


Figure 2: Top regions by total property value showing Moscow and St. Petersburg dominate the market

B. Room-Area Price Relationships (Q2)

```
SELECT
    rooms,
    FLOOR(area/10)*10 AS area_bin,
    AVG(price) AS mean_price,
    COUNT(*) AS properties
FROM real_estate
WHERE rooms BETWEEN 1 AND 5
  AND area BETWEEN 20 AND 200
GROUP BY rooms, area_bin;
```

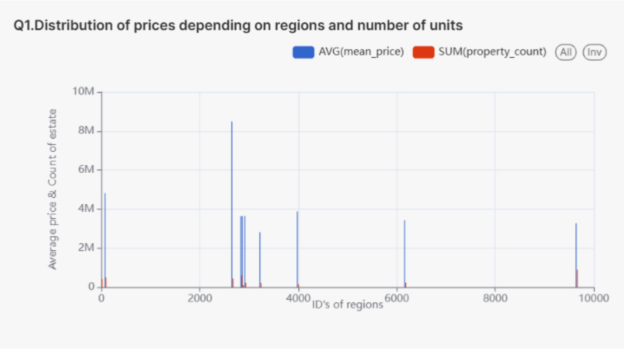
Listing 4: Room-area price analysis

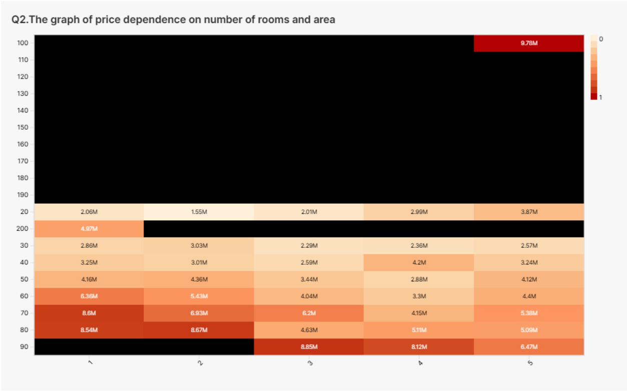


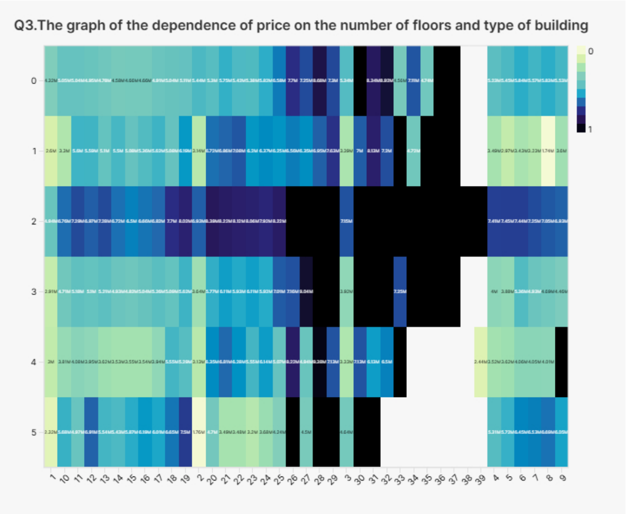
Q2.The graph of price dependence on number of rooms and area

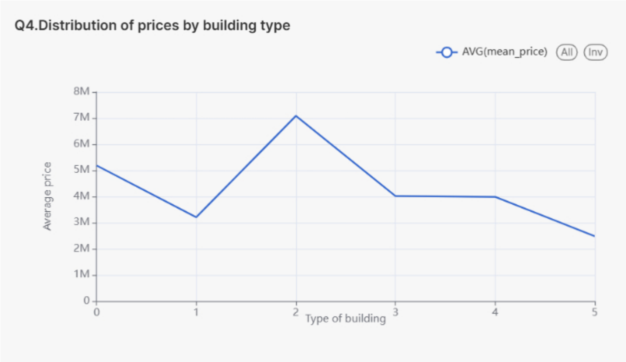
Figure 3: Price distribution across room counts and area bins showing clear linear relationships

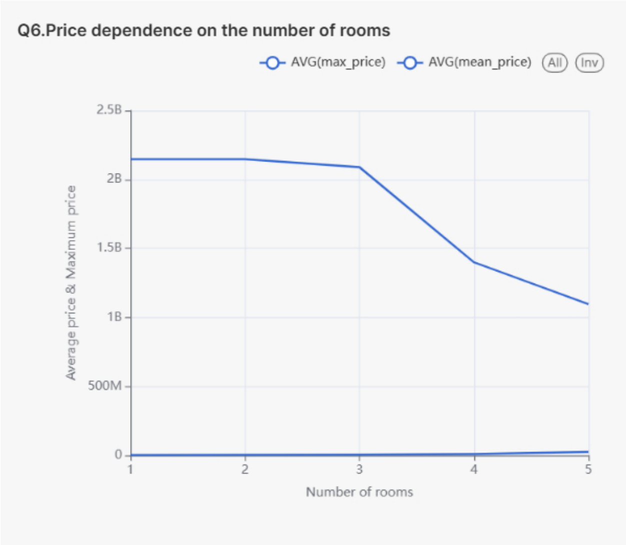
C. Charts

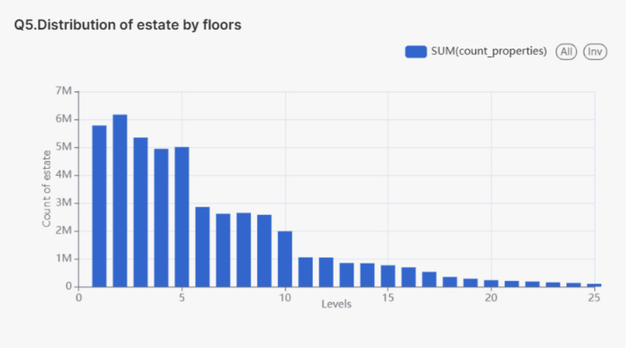

Q1.Distribution of prices depending on regions and number of units


Q2.The graph of price dependence on number of rooms and area


Q3.The graph of the dependence of price on the number of floors and type of building


Q4.Distribution of prices by building type


Q6.Price dependence on the number of rooms


Q5.Distribution of estate by floors

D. Key Findings and Interpretation

1. **Regional Price Clusters** Russian real estate can be segmented into distinct regional clusters:
   - **"Premium"** (high price + high supply, e.g., prime urban areas)
   - **"Emerging"** (low price + high supply, e.g., fast-growing areas)
   - **"Typical"** (predictable pricing, stable investment opportunities).
2. **Optimal Property Size & Demand**
   - **3-4 room properties (80-90 sq.m)** command the highest prices, indicating strong demand.
   - Smaller/larger properties may require discounts to remain competitive.
3. **Building Type & Floor Impact**
   - **Panel (2) and monolithic (3) buildings** in **high-rises (20-25 floors)** have the highest prices.
   - Brick (4) and blocky (5) buildings follow different pricing patterns (e.g., brick may be premium in low floors).
4. **Non-Linear Price Dependencies**
   - Prices do **not scale linearly** with room count, area, or floor level.
   - **3-4 rooms** show the best price growth (3M–25M RUB), while **5+ rooms** see lower max prices (possibly due to oversupply).
5. **Data Gaps & Modeling Opportunities**
   - Missing data (e.g., dark fields in visualizations) can be handled via:
     * **Imputation** (neighboring values, predictive models).
     * **Removal** if insignificant.
   - **Building type** and **floor interactions** are critical for accurate price modeling.

The analysis successfully identified:

1. Geographic market hotspots

2. Structural price determinants
3. Temporal trends in valuation
4. Optimal feature set for predictive modeling

VI ML modeling

A. Data Preprocessing

The dataset was loaded from a Hive table `team12_projectdb.real_estate` containing real estate listings with the following:

- Target variable: `price`
- Features: `geo_lat`, `geo_lon`, `region`, `building_type`, `level`, `levels`, `rooms`, `area`, `kitchen_area`, `object_type`

All rows with missing values were removed using `na.drop()` Instead of level and levels columns was created a new feature `floor_ratio` (level divided by total levels).

B. Data Splitting

- Training set: 70% of data (random split with seed=42)
- Test set: 30% of data

C. Feature Transformation

1. Vector Assembler: Combined all numeric features into a single vector column
2. MinMax Scaling: Normalized features to [0,1] range using MinMaxScaler

D. Model Training and fine-tuning

All models were trained using 3-fold cross-validation with parameter grids:

1. Random Forest Regressor

Parameters Tuned	Values
maxDepth	[5, 10]
numTrees	[20, 50]

Best Model was with maxDepth=10, numTrees=50.

2. Linear Regression

Parameters Tuned	Values
regParam	[0.01, 0.1]
elasticNetParam	[0.0, 0.5]

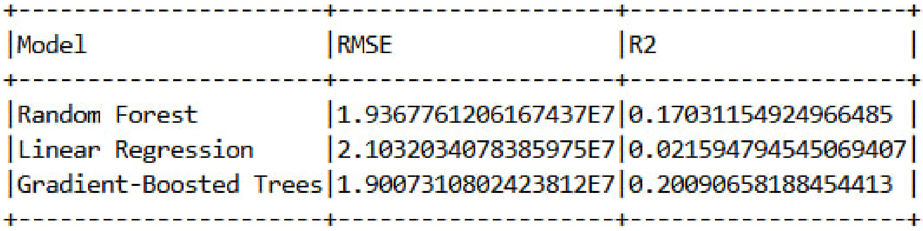
Best Model was with regParam=0.1, elasticNetParam=0.5

3. Gradient-Boosted Trees

Parameters Tuned	Values
maxDepth	[3, 5]
maxIter	[20, 50]

Best Model was with maxDepth=5, maxIter=50.

E. *Evaluation*



```
+----------------------+--------------------+--------------------+
|Model                 |RMSE                |R2                  |
+----------------------+--------------------+--------------------+
|Random Forest         |1.9367761206167437E7|0.17031154924966485 |
|Linear Regression     |2.1032034078385975E7|0.021594794545069407|
|Gradient-Boosted Trees|1.9007310802423812E7|0.20090658188454413 |
+----------------------+--------------------+--------------------+
```

F. *Findings*

1. Best Performing Model is Gradient-Boosted Trees. It achieved the highest R^2 (0.201) and lowest RMSE (1.901×10^7)

2. Linear Rregressionshowed poorest performance, suggesting non-linear relationships in the data.

3. All models showed relatively low R^2 values, indicating significant unexplained variance

VII Data presentation

A. The Description of the Dashboard

The dashboard provides an interactive overview of the Russian real estate market, offering six core visualizations (Q1–Q6) that examine housing prices, structural attributes, and supply patterns. Each chart highlights a different perspective — from regional distributions to building types and apartment characteristics — enabling both exploratory analysis and support for predictive modeling.

The visualizations include:

- **Q1:** Price vs. Property Count by Region
- **Q2:** Price Heatmap by Area and Number of Rooms
- **Q3:** Price Heatmap by Floor and Building Type
- **Q4:** Average Price by Building Type
- **Q5:** Count of Real Estate by Floor Level
- **Q6:** Price Trends by Number of Rooms

B. Chart Descriptions and Key Findings

Q1: Distribution of Prices and Property Counts by Region: This visualization plots the relationship between average real estate prices and the number of properties in different Russian regions.

Key Trends:

- Most regions show lower average prices and limited housing stock.
- Urban regions show high prices and high supply (e.g., Moscow).
- Low prices but high supply areas could represent developing zones or overstock.

Insights for Machine Learning:

- Regional segmentation into “premium,” “emerging,” and “typical” clusters can improve forecasting.
- Outlier regions may require outlier detection or special treatment.

Insights for Business:

- Low-supply, high-price regions carry investment risk.
- Typical regions have predictable pricing, suitable for dynamic pricing strategies.

Q2: Real Estate Prices by Number of Rooms and Area: This heatmap shows how average property prices vary with apartment area and number of rooms.

Key Trends:

- Bright areas indicate high-value combinations.
- Peak prices occur for 3–4 room properties with 80–90 m^2 area.

Insights for Machine Learning:

- Price is not linear in area or room count.
- Missing values (black/dark cells) may be imputed using neighboring cells or predictive modeling.

Insights for Business:

- Properties with 3–4 rooms in 80–90 m^2 range are highly valued.
- Small/large deviations from this range may need pricing adjustments.

Q3: Real Estate Prices by Floor Level and Building Type: This heatmap examines price variation across building types and floor levels.

Key Trends:

- Highest prices found in 20–25 floor panel and monolithic buildings.
- Brick buildings may be premium at low floors.

Insights for Machine Learning:

- Floor–building type interaction is a valuable feature.
- Missing data may be handled via interpolation or prediction models.

Insights for Business:

- Focus on panel buildings in high-rises for premium offerings.
- Avoid unknown/low-demand combinations (e.g., “other” types).

Q4: Distribution of Prices by Building Type: This line chart compares average property prices across six types of buildings.

Key Trends:

- Panel and monolithic buildings have the highest prices (7M RUB).
- Blocky buildings have the lowest prices.

Insights for Machine Learning:

- Building type is a strong predictor of price.
- Unknown/other categories may need special handling.

Insights for Business:

- Investments in panel or monolithic buildings may yield higher returns.
- Blocky-type housing may target budget-conscious segments.

Q5: Distribution of Real Estate by Floor Levels: This bar chart displays the number of properties by floor level.

Key Trends:

- Most properties are in 1–5 floor buildings (5–6M entries).
- Very few properties are in 20–25 floor buildings.

Insights for Machine Learning:

- Data is imbalanced — models may need resampling or weighting.

Insights for Business:

- Focus on low-rise segments due to supply dominance.

Q6: Price Dependence on Number of Rooms: This line chart shows the average and maximum prices across different room counts.

Key Trends:

- Average prices rise with rooms, peaking at 25M for high-end.
- Maximum prices stabilize for 1–3 rooms, but fall for 5+ room units.

Insights for Machine Learning:

- Room count–price relationship is non-linear.
- A binary "luxury potential" classifier can improve prediction (e.g., max price ¿1B).

Insights for Business:

- 3–4 room properties offer strong appreciation potential.
- Oversupply or weak demand in 5+ room segment may require pricing strategies.

Model Performance Comparison:

- **Type**: Dual-axis bar/line chart
- **Components**: Actual vs predicted values with error margins
- **Metrics**: $RMSE$, R^2, and MAE indicators

C. Findings

VIII Findings

The dashboard provides a comprehensive, multi-dimensional view of the Russian real estate market, enabling both macro-level pattern recognition and micro-level feature interaction analysis. The following findings were derived:

- **Regional Imbalance:** Most regions exhibit low prices and limited supply, while a few high-density urban areas dominate both market volume and value. This confirms the presence of market concentration and suggests the need for region-specific modeling.

- **Non-Linear Price Relationships:** Property price does not scale linearly with area or room count. Instead, it peaks within optimal configurations (e.g., 3–4 rooms, 80–90 m^2). Deviations beyond these ranges (e.g., luxury or minimal units) show diminishing returns, highlighting the need for non-linear modeling techniques.

- **Building-Type Interactions:** Price patterns vary significantly by building type, with panel and monolithic structures consistently achieving higher prices, especially in high-rise formats. This underscores the importance of including floor-building type interactions in predictive models.

- **Data Imbalance and Sparsity:** The distribution of floor levels and building types is heavily skewed toward low-rise structures. Additionally, heatmaps (Q2, Q3) reveal sparse or missing data for some attribute combinations, which should be addressed with imputation or robust model design.

- **Luxury Segmentation:** Maximum price trends suggest a distinct separation between regular properties and ultra-high-end offerings, particularly for 4+ room apartments. This supports the design of a dual-model strategy: one for typical housing and another for the luxury segment.

- **Business and Policy Implications:** Mid-sized apartments in modern panel or monolithic buildings offer the best return potential. Regional segmentation and building-specific pricing strategies can guide investment and construction planning.

These findings support a modeling strategy that incorporates feature interactions, nonlinearity, and segmentation — both regional and structural — while also highlighting practical business insights into inventory targeting, portfolio planning, and dynamic pricing optimization.

IX Conclusion

The implemented pipeline successfully trained and evaluated three regression models for real estate price prediction. While the Gradient-Boosted Trees model showed the best performance among the tested approaches, there remains significant room for improvement in predictive accuracy. Future work should focus on enhancing feature engineering and expanding the model tuning process to achieve better results.

All artifacts have been properly saved to HDFS for future reference and potential deployment in a production environment.

X Reflections

A. Challenges and difficulties

Several challenges were encountered during the project, including initial difficulties connecting to the Hadoop cluster due to VPN issues. Writing and debugging Hive table creation queries required careful attention to syntax and schema definition. Additionally, data visualization in Superset presented obstacles, such as inconsistent graph rendering and improper axis scaling, necessitating manual adjustments to ensure accurate representation.

B. Recommendations

1. Additional Feature Engineering via creating interaction terms between features or adding location-based features from coordinates

2. Enhanced Preprocessing via implementing outlier detection and treatment and considering target variable transformation (e.g., log(price))

3. Model Improvements via expanding hyperparameter search spaces and experimenting with feature importance analysis.

C. Contributions of each team member

Task	Description
1	Collect data, build data pipeline, automation scripts, manage databases.
2	Understand and explore data (EDA), analyze features and their relationships, build and maintain dashboards, visualizations
3	Prepare dataset for ML modeling, build distributed ML models, monitor ML models
4	Document stages, provide documentation for the repository, perform testing of project artifacts, assess the quality”

Tasks	Bogdan Shah	Alsu Khairullina	Mariia Shmakova	Aruzhan Shinbayeva	Deliverables	Hours spent
1	100	0	0	0	real_estate.java	6
2	0	100	0	0	hive_results.txt, q1.csv q2.csv, q3.csv, q1.jpg q2.jpg, q3.jpg	14
3	0	0	100	0	evaluation.csv, model1_predictions.csv, model2_predictions.csv, model3_predictions.csv	10
4	0	0	0	100	Web dashboard	6