

Airbnb listings price prediction

Introduction to Big Data

Team Members:

Evgenii Erokhin DS-01 e.erokhin@innopolis.university
Danil Davydov DS-01 d.davydov@innopolis.university
Ilia Mitrokhin DS-02 i.mitrokhin@innopolis.university
Egor Poliakov AI-01 e.poliakov@innopolis.university

Submission Date: 09/05/2025

Contents

| | | |
|-----------|--|-----------|
| 1 | Introduction | 2 |
| 2 | Business Objectives | 2 |
| 3 | Dataset Description and Data Characteristics | 2 |
| 3.1 | Missing Values Analysis | 3 |
| 3.2 | Data Suitability Assessment | 3 |
| 3.3 | Dataset Specifications | 3 |
| 4 | Architecture of Data Pipeline | 4 |
| 4.1 | Stage 1 | 4 |
| 4.2 | Stage 2 | 4 |
| 4.3 | Stage 3 | 11 |
| 4.4 | Stage 4 | 13 |
| 5 | Data Preparation | 13 |
| 5.1 | ER Diagram | 13 |
| 5.2 | Some Samples from the Database | 14 |
| 5.3 | Creating Hive Tables and Preparing the Data for Analysis | 16 |
| 6 | Data Analysis | 16 |
| 6.1 | Charts and explanation | 16 |
| 7 | ML Modeling | 22 |
| 8 | Data Presentation | 24 |
| 8.1 | The Description of the Dashboard | 24 |
| 9 | Conclusion | 25 |
| 10 | Reflections on Own Work | 25 |
| 10.1 | Challenges and Difficulties | 25 |
| 10.2 | Recommendations to us for future improvement | 26 |
| 10.3 | Table of Contributions of Each Team Member | 26 |

1 Introduction

Hosts on Airbnb often struggle to determine the optimal nightly rental price for their properties. Setting a price too high may lead to low occupancy, while pricing too low could result in lost revenue. Current pricing decisions are often based on intuition or manual comparisons with similar listings, which can be time-consuming and inaccurate.

To address this, we aim to develop a regression model that analyzes the provided data by the host—including location, property features, amenities, reviews—to recommend competitive and profitable rental prices. This solution will help hosts optimize their pricing strategy, improve occupancy rates, maximize earnings, and simplify the process of price setting.

2 Business Objectives

- **Primary Goal:** Develop a **data-driven regression model** to recommend optimal nightly rental prices for Airbnb hosts by analyzing:
 - Property attributes (type, amenities, capacity)
 - Geographic factors (country, city, neighborhood)
 - Host profile data (description completeness, ratings)
 - Market dynamics (competitor pricing, occupancy trends)
- **Key Outcomes:**
 - **Competitive Pricing:** Balance affordability for guests with revenue maximization for hosts.
 - **Occupancy Optimization:** Mitigate price-induced vacancy risks (e.g., overpricing) and revenue loss (e.g., underpricing).
 - **Process Efficiency:** Replace manual price-setting with automated, scalable recommendations.
- **Data Scope:** Analysis limited to even-row IDs (0, 2, 4, ...) from the Kaggle dataset to ensure non-redundancy with Team 24's work.

3 Dataset Description and Data Characteristics

- **Title:** Airbnb Listings Data (Cleaned)
- **Source:** Kaggle (User: joebeachcapital) <https://www.kaggle.com/datasets/joebeachcapital/airbnb>
- **Total Records:** 916,000 (our analysis uses even-row IDs: 0, 2, 4,... resulting in 458,000 entries)
- **Memory Usage:** 336.1+ MB
- **Data Types:**
 - Float64: 33 columns

- Int64: 1 column
- Object: 55 columns
- **Currency:** All prices converted from local currency to USD
- **Source:** Comprehensive data description <https://docs.google.com/spreadsheets/d/1iWCNJcSutYqpULSQH1NyGIInUvHg2BoUGoNRIGa6Szc4/edit?gid=1322284596#gid=1322284596>

3.1 Missing Values Analysis

Table 1: Top Variables with Missing Values

| Variable | Missing Count | Missing Percentage |
|----------------------|---------------|--------------------|
| Has Availability | 485,647 | 98.12% |
| Square Feet | 482,745 | 97.53% |
| License | 480,358 | 97.05% |
| Host Acceptance Rate | 452,696 | 91.46% |
| Monthly Price | 398,863 | 80.59% |

- **Complete Variables** (0% missing):
 - Host URL, Host ID, Experiences Offered
 - Scrape ID, ID

3.2 Data Suitability Assessment

Despite some missing values in less critical features, the dataset contains:

- Complete geographical and host identification data
- Robust pricing information (converted to USD)
- Sufficient property attributes for modeling
- **Conclusion:** The dataset is comprehensive and appropriate for achieving our business objectives of price prediction, with 458K clean records available for analysis.

3.3 Dataset Specifications

- **Records:** ~916,000 the total dataset our part is 458,000 entries (original dataset; our analysis uses even-row IDs: 0, 2, 4, ...)
- **Columns:** ~89 attributes (mix of numeric, categorical, and text)
- **Time Period:** Snapshot of active listings (exact year not specified; infer from host registration dates)

4 Architecture of Data Pipeline

4.1 Stage 1

- Download csv file from Yandex Disc
- Split on 4 tables: Hosts, Listings, Listing features, Review scores
- Upload to PostgreSQL, maintaining connection between tables via foreign key
- Import from PostgreSQL to HDFS via Sqoop using compression Snappy like AVRO files (We used AVRO since our data is row oriented and have many textual fields)

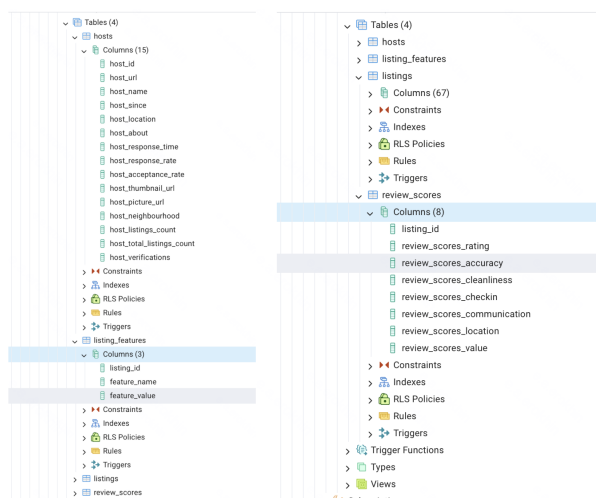


Figure 1: Stage 1 results

4.2 Stage 2

- Import our tables to Hive, created bucketed table hosts by host location, partitioned listings by country, bucketed reviews score clustered by listing ID
- Create six SQL queries for Dashboard charts
- Results of the queries are stored in csv file in output folder

Listing 1: Q1

```

1  -- year when host register
2  USE team19_projectdb;
3
4  -- Drop existing table (if needed)
5  DROP TABLE IF EXISTS q1_results;
6
7  -- Create external table to store results
8  CREATE EXTERNAL TABLE q1_results(
9      host_since_year NUMERIC,
10     host_count BIGINT
11 )

```

```

12 ROW FORMAT DELIMITED
13 FIELDS TERMINATED BY ','
14 LOCATION 'project/hive/warehouse/q1';
15
16 -- Disable unique column names in output
17 SET hive.resultset.use.unique.column.names = false;
18
19 -- Insert data into q1_results
20 INSERT INTO q1_results
21 SELECT
22     YEAR(host_since) AS host_since_year, -- Extract year using
23     COUNT(host_id) AS host_count         -- Count hosts per
24     year
25 FROM hosts_bucketed
26 GROUP BY YEAR(host_since)               -- Group by the year
27     expression
28 ORDER BY YEAR(host_since);              -- Order by the year
29     expression
27
28 -- Verify results
29 SELECT * FROM q1_results;

```

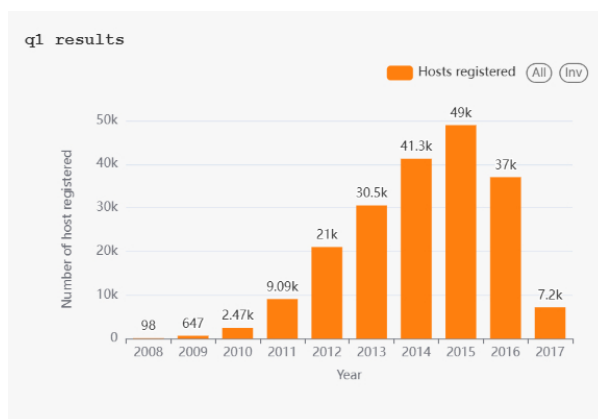


Figure 2: Q1 results

Listing 2: Q2

```

1 -- how much hosts has description
2 USE team19_projectdb;
3
4 -- Drop existing table (if needed)
5 DROP TABLE IF EXISTS q2_results;
6
7 -- Create external table to store results
8 CREATE EXTERNAL TABLE q2_results(
9     description_status STRING,
10    host_count BIGINT
11 )
12 ROW FORMAT DELIMITED

```

```

13  FIELDS TERMINATED BY ','
14  LOCATION 'project/hive/warehouse/q2';
15
16  -- Disable unique column names in output
17  SET hive.resultset.use.unique.column.names = false;
18
19
20  -- Insert data into q2_results
21  INSERT INTO q2_results
22
23  SELECT
24      description_status,
25      COUNT(host_id) AS host_count
26  FROM (
27      SELECT
28          host_id,
29          CASE
30              WHEN host_about IS NOT NULL AND TRIM(host_about)
31                  != '' THEN 'Has description'
32              ELSE 'No description'
33          END AS description_status
34      FROM hosts_bucketed
35  ) AS subquery
36  GROUP BY description_status;
37
38  -- Verify results
39  SELECT * FROM q2_results;

```

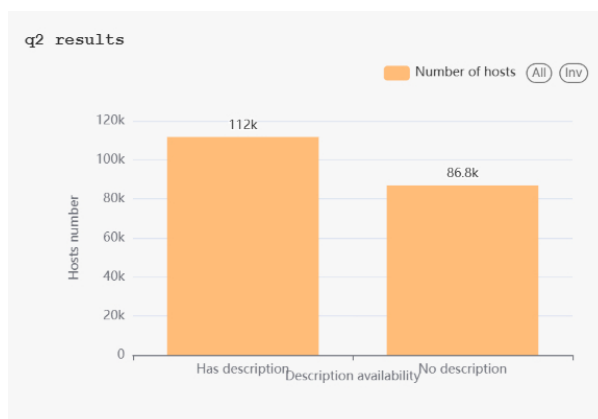


Figure 3: Q2 results

Listing 3: Q3

```

1  -- Top 15 country by avg price
2
3  USE team19_projectdb;
4
5  -- Drop existing table (if needed)
6  DROP TABLE IF EXISTS q3_results;
7

```

```

8  -- Create external table to store results
9  CREATE EXTERNAL TABLE q3_results(
10     country STRING,
11     avg_price FLOAT
12 )
13 ROW FORMAT DELIMITED
14 FIELDS TERMINATED BY ','
15 LOCATION 'project/hive/warehouse/q3';
16
17 -- Disable unique column names in output
18 SET hive.resultset.use.unique.column.names = false;
19
20
21 -- Insert data into q3_results
22 INSERT INTO q3_results
23
24 SELECT
25     country,
26     AVG(
27         price
28     ) AS avg_price
29 FROM listings_partitioned
30 WHERE price IS NOT NULL
31 GROUP BY country
32 HAVING
33     AVG(
34         price
35     ) IS NOT NULL
36 ORDER BY avg_price DESC
37 LIMIT 15;
38
39 -- Verify results
40 SELECT * FROM q3_results;

```

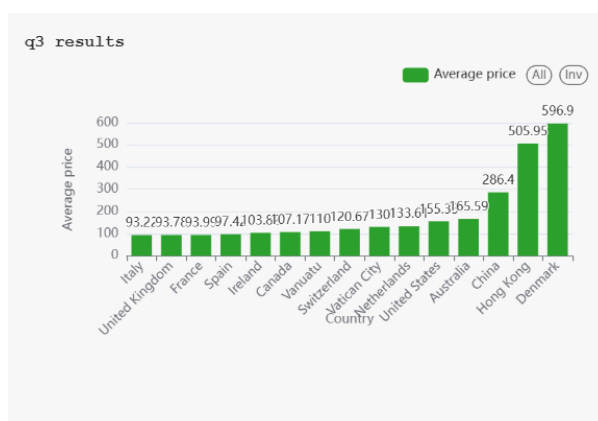


Figure 4: Q3 results

Listing 4: Q4

```

1  -- Top 10 city by most frequent

```



```

2
3 USE team19_projectdb;
4
5 -- Drop existing table (if needed)
6 DROP TABLE IF EXISTS q4_results;
7 -- dfs -rm -r /user/team19/project/hive/warehouse/q4;
8 -- Create external table to store results
9 CREATE EXTERNAL TABLE q4_results(
10     city STRING,
11     count_city BIGINT
12 )
13 ROW FORMAT DELIMITED
14 FIELDS TERMINATED BY ','
15 LOCATION 'project/hive/warehouse/q4';
16
17 -- Disable unique column names in output
18 SET hive.resultset.use.unique.column.names = false;
19
20 -- Insert data into q4_results
21 INSERT INTO q4_results
22
23 select city, count(id) as count_city from
24     listings_partitioned
25 group by city
26 order by count_city desc
27 Limit 10;
28
29 -- Verify results
30 SELECT * FROM q4_results;

```

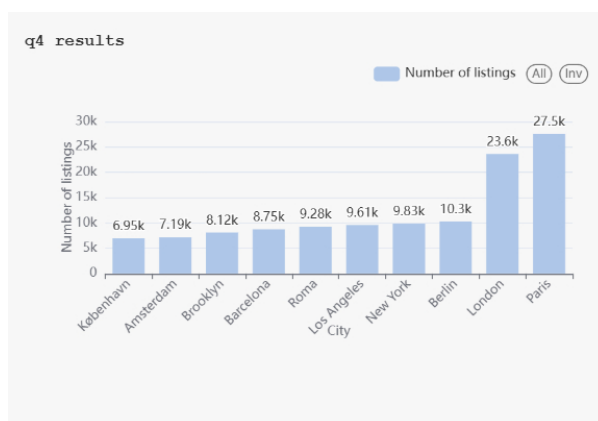


Figure 5: Q4 results

Listing 5: Q5

```

1 -- avg price by score
2
3 USE team19_projectdb;
4
5 -- Drop existing table (if needed)

```

```

6 DROP TABLE IF EXISTS q5_results;
7 -- dfs -rm -r /user/team19/project/hive/warehouse/q5;
8 -- Create external table to store results
9 CREATE EXTERNAL TABLE q5_results(
10     rating_group NUMERIC,
11     avg_price FLOAT
12 )
13 ROW FORMAT DELIMITED
14 FIELDS TERMINATED BY ','
15 LOCATION 'project/hive/warehouse/q5';
16
17 -- Disable unique column names in output
18 SET hive.resultset.use.unique.column.names = false;
19
20 -- Insert data into q5_results
21 INSERT INTO q5_results
22
23
24 SELECT
25     rating_group,
26     CAST(
27         AVG(cleaned_price) * 100 AS INT
28     ) / 100.0 AS avg_price
29 FROM (
30     SELECT
31         rs.review_scores_rating,
32         CASE
33             WHEN rs.review_scores_rating = 100 THEN 10
34             ELSE FLOOR(rs.review_scores_rating / 10) + 1
35         END AS rating_group,
36         l.price AS cleaned_price
37     FROM
38         listings_partitioned l
39     JOIN
40         review_scores_bucketed rs ON l.id = rs.listing_id
41     WHERE
42         rs.review_scores_rating BETWEEN 1 AND 100
43         AND l.price IS NOT NULL
44 ) AS prepared_data
45 GROUP BY rating_group
46 ORDER BY rating_group;
47
48
49 -- Verify results
50 SELECT * FROM q5_results;

```

Listing 6: Q6

```

1 -- avg price by property_type
2
3 USE team19_projectdb;
4

```

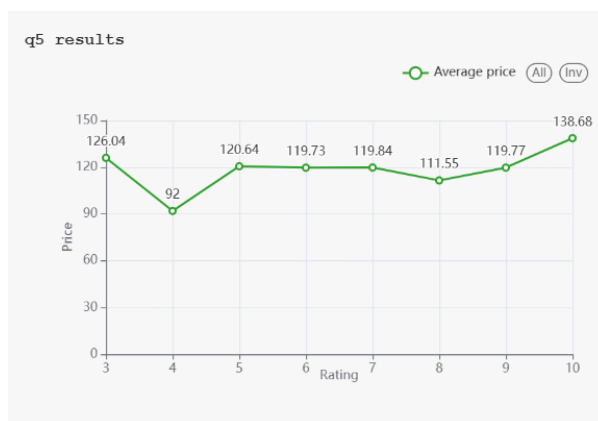


Figure 6: Q5 results

```

5  -- Drop existing table (if needed)
6  DROP TABLE IF EXISTS q6_results;
7
8  -- Create external table to store results
9  CREATE EXTERNAL TABLE q6_results(
10     property_type STRING,
11     avg_price FLOAT
12 )
13 ROW FORMAT DELIMITED
14 FIELDS TERMINATED BY ','
15 LOCATION 'project/hive/warehouse/q6';
16
17 -- Disable unique column names in output
18 SET hive.resultset.use.unique.column.names = false;
19
20
21 -- Insert data into q6_results
22 INSERT INTO q6_results
23
24 SELECT
25     property_type,
26     AVG(price) AS avg_price
27 FROM listings_partitioned
28 WHERE price IS NOT NULL
29 GROUP BY property_type
30 HAVING
31     AVG(price) IS NOT NULL
32 ORDER BY avg_price DESC;
33
34 -- Verify results
35 SELECT * FROM q6_results;

```

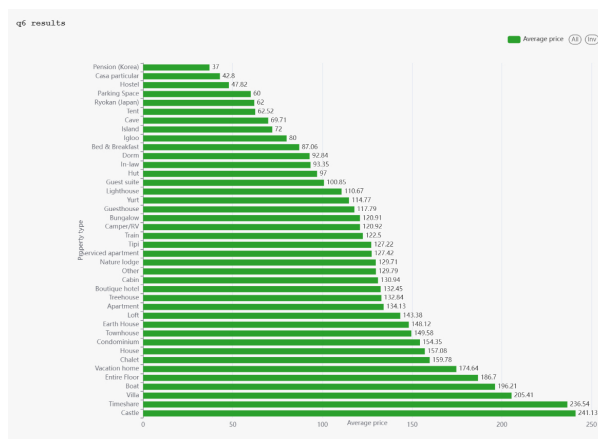


Figure 7: Q6 results

4.3 Stage 3

- Connect to hive and read hive tables
- Join hive tables to 1 dataframe
- Select relevant features and delete null values
- Extract datetime from string field
- Encode datetime data using sin and cos transformation
- Initialize tokenizer Word2Vec indexers encoders and assembler
- Initialize pipeline with stages from previous step
- Transform data using pipeline we built
- Create vector indexer and parse date through that
- Split data to the train and test

Train test split

We utilized 60/40 train/test split

Train: 90550

Test: 60490

Figure 8: Train/test split results

- Reduce amount of partitions
- Build, train, predict and evaluate linear regression model

- Hyper - parameter optimization for linear regression model

The results of hyper-parameter optimization

results of hyper-parameter optimization linear regression

| reg_param | elastic_net_param | rmse |
|-----------|-------------------|-------------------|
| 0.01 | 0.2 | 74.21428413776074 |
| 0.01 | 0.6 | 74.14980552889402 |
| 0.1 | 0.2 | 73.92970762445209 |
| 0.1 | 0.6 | 73.51338533201208 |

Linear model with reg_param = 0.1 and elastic_net_param = 0.6 having rmse-73.82 is the best linear model

Figure 9: Results of hyper-parameter optimization linear regression

- Save best linear regression model

Evaluation result of linear regression

| model | rmse | r2 |
|------------------|-------------------|--------------------|
| linear_reg_tuned | 73.80396417408669 | 0.7295295887924427 |

Figure 10: Best linear regression model

- Build, train, predict and evaluate Gradient boosting tree model
- Hyper - parameter optimization for Gradient boosting tree model

results of hyper-parameter optimization GBT regression

| max_depth | min_instances_per_node | rmse |
|-----------|------------------------|-------------------|
| 3 | 1 | 77.1218302181103 |
| 3 | 2 | 77.1218302181103 |
| 5 | 1 | 70.36428485919534 |
| 5 | 2 | 70.49007973590999 |

GBT model with max_depth = 5 and min_instances_per_node = 1 having rmse-71.03 is the best GBT model

Figure 11: Results of hyper-parameter optimization GBT regression


- Save the best gradient-boosting tree model



| model | rmse | r2 |
|----------|-------------------|-------------------|
| GBT_tune | 70.77984052530067 | 0.751240576223164 |

Figure 12: Best Gradient boosting tree model

- Compare the best models



| model | rmse | r2 |
|--|-------------------|-------------------|
| GBMRegressor(learning_rate=0.01, max_depth=3, min_samples_split=10, subsample=0.5) | 70.807817128845 | 0.74243097028207 |
| GBMRegressor(learning_rate=0.01, max_depth=3, min_samples_split=10, subsample=0.5) | 70.77984052530067 | 0.751240576223164 |

Figure 13: Comparison best models

4.4 Stage 4

- Import results of stage 3 into hive
- Creating pretty custom dashboard with data description, charts, key findings from stage 2 and results of stage 3
- <http://hadoop-03.uni.innopolis.ru:8808/superset/dashboard/142/>

5 Data Preparation

5.1 ER Diagram

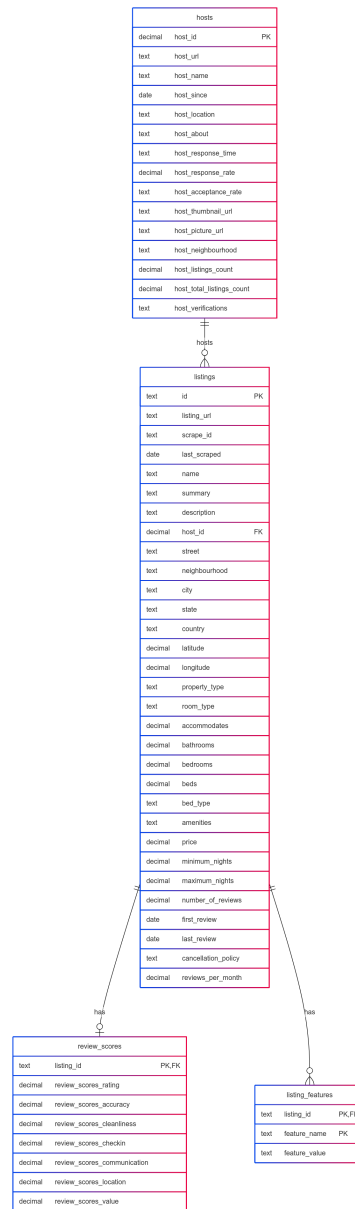
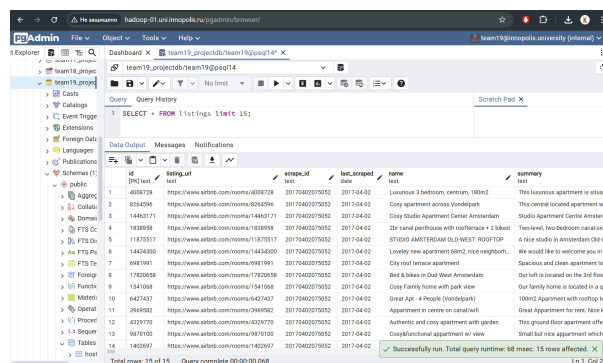


Figure 14: ER Diagram

5.2 Some Samples from the Database

The screenshot shows the DBeaver IDE interface. The top toolbar includes icons for Explorer, Dashboard, and Query History. The main window displays a SQL query: `SELECT * FROM hosts limit 15;`. The results are shown in a table with columns: host_id, host_name, host_since, host_location, and host_about. The table contains 15 rows of data, including host details like ID, name, location, and a brief description.

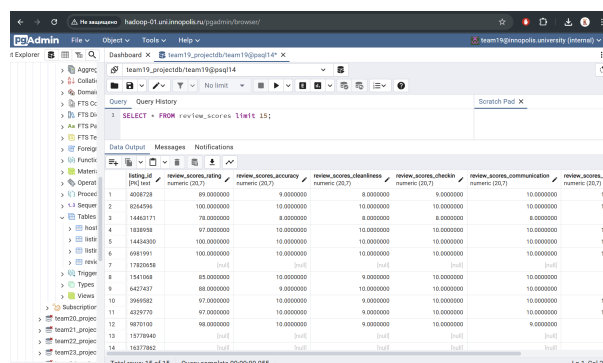
Figure 15: Hosts



The screenshot shows the DBeaver IDE interface. The top toolbar includes icons for Explorer, Dashboard, and Query History. The main window displays a SQL query: `SELECT * FROM listings limit 15;`. The results are shown in a table with columns: id, listing_url, scope_id, host_id, name, and summary. The table contains 15 rows of data, including listing details like ID, URL, location, and a brief description.

| id | listing_url | scope_id | host_id | name | summary |
|----|-------------|---------------------------------------|----------|---|--------------------------------------|
| 1 | 4008728 | https://www.airbnb.com/rooms/4008728 | 20786453 | Luxurious 3 bedrooms, centrum, 180m2 | This luxurious apartment is situated |
| 2 | 8284594 | https://www.airbnb.com/rooms/8284594 | 20786453 | Cozy apartment across Vondelpark | This central located apartment is |
| 3 | 14483771 | https://www.airbnb.com/rooms/14483771 | 20786453 | Cozy Studio Apartment Center Amsterdam | Studio Apartment Center Amsterdam |
| 4 | 1838958 | https://www.airbnb.com/rooms/1838958 | 20786453 | Stylish penthouse with terrace + 23kcal | Two level, two-bedroom canal-side |
| 5 | 11875517 | https://www.airbnb.com/rooms/11875517 | 20786453 | STUDIO AMSTERDAM OLD-WEST: ROOFTOP | A nice studio in Amsterdam Old West |
| 6 | 1442430 | https://www.airbnb.com/rooms/1442430 | 20786453 | Lovely new apartment 10min. nice neighborhood | We would like to welcome you in a |
| 7 | 6981991 | https://www.airbnb.com/rooms/6981991 | 20786453 | City and terrace apartment | Spacious and clean apartment for |
| 8 | 17820658 | https://www.airbnb.com/rooms/17820658 | 20786453 | Bed & blues in Old West Amsterdam | Our loft is located on the 3rd floor |
| 9 | 1541068 | https://www.airbnb.com/rooms/1541068 | 20786453 | Cozy Family home with park view | Our family home is located in a st |
| 10 | 6427427 | https://www.airbnb.com/rooms/6427427 | 20786453 | Great Apt. - 4 People (Vondelpark) | 100m2 Apartment with rooftop to |
| 11 | 3949582 | https://www.airbnb.com/rooms/3949582 | 20786453 | Apartment in center on canal-side | Great Apartment for rent. Near H |
| 12 | 4329770 | https://www.airbnb.com/rooms/4329770 | 20786453 | Authentic and cozy apartment with garden | This ground floor apartment offer |
| 13 | 9870100 | https://www.airbnb.com/rooms/9870100 | 20786453 | Cozy functional apartment w view | Small but nice apartment which |
| 14 | 1420697 | https://www.airbnb.com/rooms/1420697 | 20786453 | | |

Figure 16: Listings



The screenshot shows the DBeaver IDE interface. The top toolbar includes icons for Explorer, Dashboard, and Query History. The main window displays a SQL query: `SELECT * FROM review_scores limit 15;`. The results are shown in a table with columns: listing_id, review_scores, review_scores_accuracy, review_scores_cleanliness, review_scores_checkin, review_scores_communication, and review_scores_location. The table contains 15 rows of data, including review scores for various listings.

| listing_id | review_scores | review_scores_accuracy | review_scores_cleanliness | review_scores_checkin | review_scores_communication | review_scores_location |
|------------|---------------|------------------------|---------------------------|-----------------------|-----------------------------|------------------------|
| 1 | 4008728 | 89.00000000 | 9.00000000 | 9.00000000 | 10.00000000 | 10.00000000 |
| 2 | 8284594 | 100.00000000 | 10.00000000 | 10.00000000 | 10.00000000 | 10.00000000 |
| 3 | 14483771 | 78.00000000 | 8.00000000 | 8.00000000 | 8.00000000 | 8.00000000 |
| 4 | 1838958 | 97.00000000 | 10.00000000 | 10.00000000 | 10.00000000 | 10.00000000 |
| 5 | 1442430 | 100.00000000 | 10.00000000 | 10.00000000 | 10.00000000 | 10.00000000 |
| 6 | 6981991 | 100.00000000 | 10.00000000 | 10.00000000 | 10.00000000 | 10.00000000 |
| 7 | 17820658 | [null] | [null] | [null] | [null] | [null] |
| 8 | 1541068 | 85.00000000 | 10.00000000 | 9.00000000 | 10.00000000 | 10.00000000 |
| 9 | 6427427 | 88.00000000 | 9.00000000 | 10.00000000 | 10.00000000 | 10.00000000 |
| 10 | 3949582 | 97.00000000 | 10.00000000 | 9.00000000 | 10.00000000 | 10.00000000 |
| 11 | 4329770 | 97.00000000 | 10.00000000 | 9.00000000 | 10.00000000 | 10.00000000 |
| 12 | 9870100 | 98.00000000 | 10.00000000 | 9.00000000 | 10.00000000 | 9.00000000 |
| 13 | 14708079 | [null] | [null] | [null] | [null] | [null] |
| 14 | 1637382 | [null] | [null] | [null] | [null] | [null] |

Figure 17: Review scores

| listing_id | listing_name | listing_value |
|------------|------------------------|---------------|
| 1 | Host Has Profile Pic | true |
| 2 | Host Identity Verified | true |
| 3 | Is Location Exact | true |
| 4 | Host Has Profile Pic | true |
| 5 | Host Identity Verified | true |
| 6 | Is Location Exact | true |
| 7 | Host Has Profile Pic | true |
| 8 | Host Has Profile Pic | true |
| 9 | Host Identity Verified | true |
| 10 | Is Location Exact | true |
| 11 | Host Has Profile Pic | true |
| 12 | Host Identity Verified | true |
| 13 | Host Has Profile Pic | true |
| 14 | Host Identity Verified | true |

Figure 18: Listing features

5.3 Creating Hive Tables and Preparing the Data for Analysis

Import our tables to Hive, created bucketed table hosts by host location, partitioned listings by country, bucketed reviews score clustered by listing ID

6 Data Analysis

We made analysis, plot charts and explain them from business side

6.1 Charts and explanation

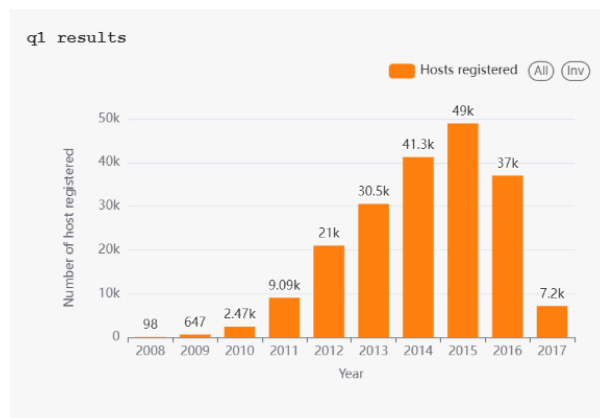


Figure 19: Q1 results

- **What and Why to Investigate**

This query examines when hosts registered on the platform by year. The insights worth investigating include:

- Annual registration trends to track platform growth over time
- Peak registration years to identify successful growth periods
- Slowing registration periods to spot potential market challenges

- **Key Insights**

Based on the host registration data from 2008-2018:

- **Growth Pattern:** Minimal registrations in 2008-2010, followed by rapid growth peaking in 2015 (~49,000 new hosts)
 - **Peak and Decline:** Significant drop after 2015, returning to pre-growth levels by 2018
 - **Market Lifecycle:** Shows typical business cycle (early adoption → explosive growth → market maturation)
 - **Strategic Implications:** Need to focus on host retention and investigate causes of the 2015 surge and decline
 - **Further Analysis Needed:** Examine 2015-2016 events that reversed growth momentum
- **Why This Matters**
Understanding these patterns helps:
 - Assess market maturity
 - Develop targeted strategies for different host cohorts
 - Investigate post-2015 market challenges

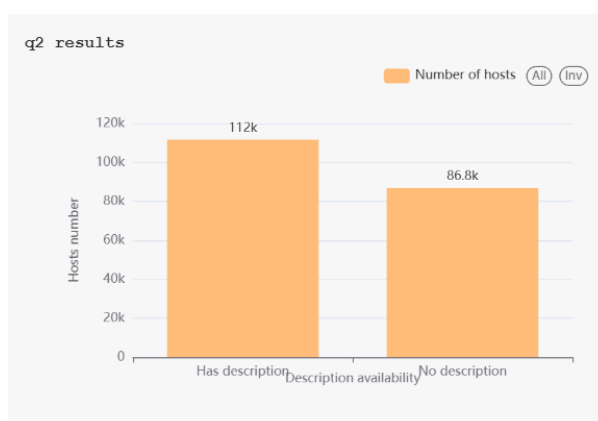


Figure 20: Q2 results

- **Analysis of Host Profile Descriptions**

This analysis examines whether hosts have provided a description in their profiles (in the `host_about` field). The analysis reveals current host engagement in fulfilling profile information and supports future platform development strategies.

- **Key Insights**

- Has description: ~111,000 hosts (57%)
- No description: ~85,000 hosts (43%)

The majority of hosts complete their profiles with a description, but a significant portion (over 40%) do not.

• Findings

- Majority Have Descriptions: 57% of hosts complete this profile element, suggesting most understand the value of personal descriptions.
- Significant Gap: The 43% without descriptions represents a substantial opportunity for platform improvement.
- Engagement Indicator: The balanced distribution suggests a key differentiator between engaged and casual hosts.
- Trust Factor: Descriptions build trust; increasing completion rates could boost conversions.
- Strategic Opportunity: Targeting the 85,000 hosts without descriptions could improve platform quality with focused effort.
- Future Opportunities: Fulfilling the "About" section enables targeted offers and revenue growth.

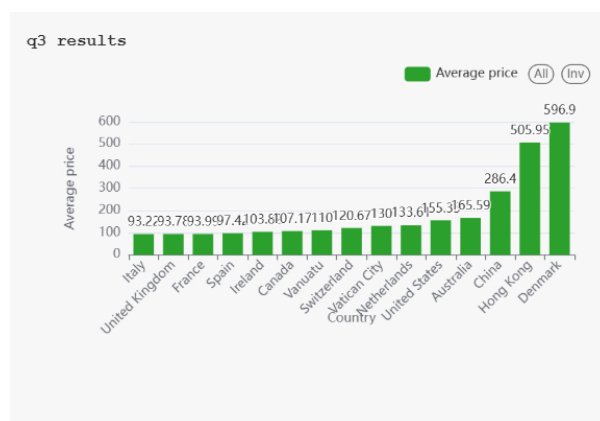


Figure 21: Q3 results

• What and Why to Investigate

This analysis examines the average price of listings across different countries. Understanding price variations by location provides insights into:

- Market positioning
- Pricing strategies
- Destination popularity

These factors are essential for platform optimization and business planning.

• Key Insights

Based on the chart showing countries by average price:

- **Price Tier Clustering:** Most premium countries cluster between \$500-\$600, suggesting a luxury accommodation price ceiling
- **Premium Destinations:** Highest prices in Denmark and Hong Kong (luxury/tourist markets with high costs of living)

- **Minimal Variation:** European countries show consistent premium pricing (\$100-\$150 range difference)
- **Location-Specific Patterns:** Significant jumps between price tiers indicate distinct regional market conditions
- **Strategic Opportunities:** Clear price tiers suggest potential for country-specific dynamic pricing strategies

- **Strategic Implications**

- Potential to adjust commission structures by country tier
- Opportunity to develop targeted marketing for under-priced markets
- Basis for localized host pricing recommendations

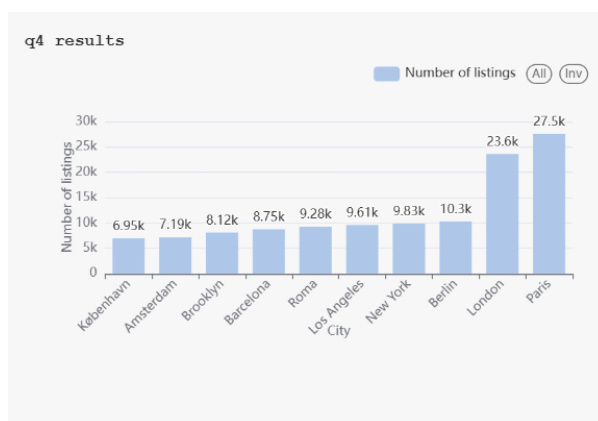


Figure 22: Q4 results

- **What and Why to Investigate**

This analysis examines the distribution of listings across different cities to understand:

- Market penetration patterns
- Platform growth geography
- Strategic focus areas

These insights are crucial for resource allocation and expansion planning.

- **Key Insights**

Based on the top 10 cities by listing count:

- **Market Concentration:**
 - * Paris (~27,000 listings)
 - * London (~23,000 listings)

show dominant market positions
- **Tier Structure:**

- * *Dominant markets*: 20,000+ listings (London, Paris)
- * *Mid-tier markets*: 8,000-10,000 listings (New York, Roma, Berlin)
- * *Emerging markets*: <8,000 listings (København, others)
- **Geographic Diversity**: Top cities span Europe and North America
- **Urban Focus**: All leading cities are major metropolitan areas
- **Growth Potential**: Significant expansion opportunities in mid-tier markets
- **Strategic Implications**
 - Prioritize support for dominant markets to maintain leadership
 - Develop targeted growth programs for mid-tier cities
 - Explore urban expansion models for new markets

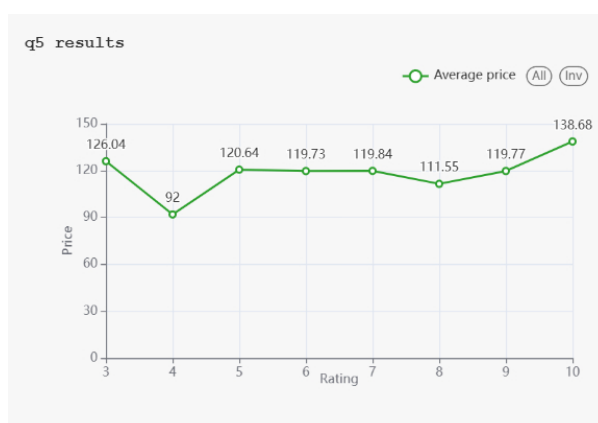


Figure 23: Q5 results

- **What and Why to Investigate**

This analysis examines the relationship between listing ratings and average prices to understand:

- Value perception dynamics
- Pricing optimization opportunities
- Quality-price tradeoffs

These insights help maximize revenue while maintaining guest satisfaction.

- **Key Insights**

Based on average price by rating group (3-10 scale):

- **Non-Linear Relationship**: U-shaped curve reveals complex value perception
- **Price Dip**: Lowest prices at ~\$90 for 4-star ratings (value segment/quality concerns)
- **Premium Pricing**: 9-10 star ratings command ~\$140 (price premium for excellence)

- **Quality Threshold:** Significant price increase after 4-star threshold
- **Mid-Range Stability:** 5-8 star ratings maintain stable ~\$120 pricing (market standard)
- **Strategic Implications**
 - Encourage hosts to improve ratings beyond 4-star threshold
 - Highlight premium pricing potential for high-rated listings
 - Develop pricing guidance based on rating brackets
 - Investigate causes of 4-star price depression

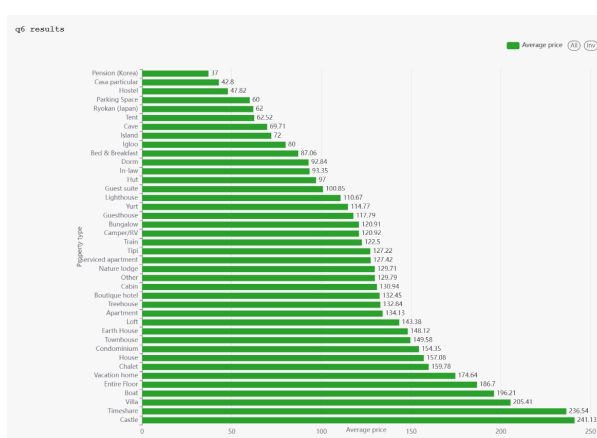


Figure 24: Q6 results

- **What and Why to Investigate**

This analysis examines price variations across property types to reveal:

- Market segmentation patterns
- Pricing strategy opportunities
- Consumer preference trends

These insights inform platform strategy and host pricing guidance.

- **Key Insights**

Average prices by property type show:

- **Extreme Price Range:** 6x difference from budget (\$40) to luxury (\$240)
- **Luxury Premium:** Castles command \$200+ nightly rates
- **Clear Market Tiers:**
 - * *Budget* (\$40-\$70): Pensions, basic units
 - * *Standard* (\$80-\$120): Apartments, guest suites
 - * *Premium* (\$130-\$180): Treehouses, unique stays
 - * *Luxury* (\$190+): Castles, exclusive properties

- **Uniqueness Premium:** Distinctive properties earn 2-3x standard rates
- **Smooth Price Gradient:** Gradual increases between most categories
- **Strategic Implications**
 - Develop tier-specific marketing campaigns
 - Create pricing benchmarks for each property category
 - Highlight ROI potential for unique property investments
 - Educate hosts about category-based price expectations

7 ML Modeling

- **Data Pipeline Setup**
 - Established connection to Hive and loaded required tables
 - Performed table joins to create a unified dataset
 - Selected relevant features and handled null values
 - Processed datetime fields from string format
 - Applied cyclical encoding (sin/cos) to datetime features
- **Feature Engineering**
 - Initialized text processing with Word2Vec tokenizer
 - Configured indexers and encoders for categorical features
 - Built feature assembler for model input
 - Created and executed a comprehensive ML pipeline
 - Implemented vector indexing for optimized processing
- **Model Development**
 - Split data into training (60%) and test (40%) sets

Train test split

```
We utilized 60/40 train/test split
Train: 90550
Test: 60490
```

Figure 25: Train/test split results

- Optimized partition count for efficient YARN processing
- Deployed Spark ML application on YARN cluster:

```

DRIVER_MEMORY="4g"
EXECUTOR_MEMORY="4g"
EXECUTOR_CORES=4
NUM_EXECUTORS=4

spark-submit \
  --master yarn \
  --driver-memory $DRIVER_MEMORY \
  --executor-memory $EXECUTOR_MEMORY \
  --executor-cores $EXECUTOR_CORES \
  --num-executors $NUM_EXECUTORS \
  --conf spark.yarn.appMasterEnv.PYSPARK_PYTHON=$PYSPARK_PYTHON \
  --conf spark.executorEnv.PYSPARK_PYTHON=$PYSPARK_PYTHON \
  scripts/model.py

```

Figure 26: YARN usage

- * Configured with 4 executor nodes
- * Allocated 4GB memory per executor
- * Managed resources through YARN ResourceManager

• Linear Regression Implementation

- Developed and trained baseline linear regression model
- Conducted hyperparameter tuning using grid search

The results of hyper-parameter optimization

results of hyper-parameter optimization linear regression

| reg_param | elastic_net_param | rmse |
|-----------|-------------------|-------------------|
| 0.01 | 0.2 | 74.21428413776074 |
| 0.01 | 0.6 | 74.14980552889402 |
| 0.1 | 0.2 | 73.92970762445209 |
| 0.1 | 0.6 | 73.51338533201208 |

Linear model with reg_param = 0.1 and elastic_net_param = 0.6 having rmse=73.82 is the best linear model

Figure 27: Results of hyper-parameter optimization (Linear Regression)

- Persisted best-performing linear regression model

Evaluation result of linear regression

| model | rmse | r2 |
|------------------|-------------------|--------------------|
| linear_reg_tuned | 73.80396417408669 | 0.7295295887924427 |

Figure 28: Best linear regression model

• Gradient Boosted Trees Implementation

- Built and evaluated GBT regression model
- Optimized through extensive parameter tuning

results of hyper-parameter optimization GBT regression

| max_depth | min_instances_per_node | rmse |
|-----------|------------------------|-------------------|
| 3 | 1 | 77.1218302181103 |
| 3 | 2 | 77.1218302181103 |
| 5 | 1 | 70.36428485919534 |
| 5 | 2 | 70.49007973590999 |

GBT model with max_depth = 5 and min_instances_per_node = 1 having rmse=71.03 is the best GBT model

Figure 29: Results of hyper-parameter optimization (GBT)

- Saved optimal GBT model configuration

Evaluation result of GBT

| model | rmse | r2 |
|----------|-------------------|-------------------|
| GBT_tune | 70.77984052530067 | 0.751240576223164 |

Figure 30: Best Gradient boosting tree model

• Model Comparison

- Evaluated both models on test dataset
- Compared performance metrics (RMSE, R-squared)

Comparison table

| model | rmse | r2 |
|----------|-------------------|-------------------|
| GBT_tune | 70.77984052530067 | 0.751240576223164 |

Figure 31: Comparison of best performing models

- Selected best-performing model for deployment

8 Data Presentation

8.1 The Description of the Dashboard

We have 3 tabs all tabs are changed by styled css for pretty view

- The description of initial data
- EDA - Charts and key insights
 - Distribution of hosts registered on platform by year
 - Do hosts have provided a description in their profiles
 - Distribution of average price of listings across different countries
 - Distribution of listings across different cities
 - Relationship between listing ratings and average price
 - Relationship between property types and their average prices
- PDA - Modeling results

9 Conclusion

Conclusion: Airbnb Price Prediction System

Our team has successfully developed a comprehensive Airbnb price prediction system that leverages both traditional data processing techniques and modern big data technologies. This project demonstrates the effective application of a diverse technology stack to solve a real-world problem in the hospitality and short-term rental market.

Business Impact

Our price prediction system provides significant value to various stakeholders:

- **Property Hosts:** Can optimize their pricing strategies based on market trends and property attributes
- **Travelers:** Gain insights into fair pricing across different locations and property types
- **Market Analysts:** Can identify underserved markets and pricing anomalies
- **Platform Operators:** Can use predictions to suggest optimal pricing to maximize bookings

Final Assessment

This project effectively demonstrates how traditional tools like Python, Bash, and PostgreSQL can be integrated with modern big data technologies such as HDFS, YARN, Spark, Hive, and TEZ to solve complex problems like Airbnb price prediction. The combination of relational database design, distributed computing, and interactive visualization has resulted in a robust system that delivers actionable insights for the short-term rental market.

Our team has not only delivered a functional price prediction system but has also gained valuable experience in applying these diverse technologies in a cohesive manner. The skills and knowledge acquired through this project provide a strong foundation for future data-intensive applications across various domains.

10 Reflections on Own Work

10.1 Challenges and Difficulties

- Huge dataset with a lot of text features which affect data loading and data preparation
- Problem with view in Superset (Sometimes double charts in tabs, hard to delete)
- Problem with passing python environment via spark submit

10.2 Recommendations to us for future improvement

- More communication between team members
- Expand feature engineering
- Use other ml models to improve the results of prediction

10.3 Table of Contributions of Each Team Member

Table 2: Project Tasks Distribution

| Project Tasks | Task description | Ilia Mitrokhin | Evgenii Erokhin | Danil Davydov | Egor Poliakov | deliverables | Average hours spent |
|----------------------------------|---|----------------|-----------------|---------------|---------------|---|---------------------|
| Data extraction and ingestion | Collect the data from the kaggle, extract even rows for the project and Spit dataset in 4 tables load to PostgreSQL | 20% | 20% | 45% | 15% | data.csv, 4 tables | 20 |
| Exploratory data analysis | Perform initial analysis to identify patterns, correlations, and insights from the data | 20% | 15% | 20% | 45% | 6 HQL scripts, charts | 10 |
| Model development and evaluation | Create and train machine learning models based on the prepared data | 50% | 20% | 20% | 10% | 2 trained model, hyper-parameters search result, evaluation metrics | 30 |
| Dashboard creation | Build an interactive visualization dashboard to present the results | 20% | 45% | 25% | 10% | Dashboard via Superset | 17 |
| Final report | Write and compile the final project report with methodology and findings | 30% | 30% | 30% | 10% | final report | 12 |
| Presentation | Prepare and deliver the project presentation | 25% | 35% | 25% | 15% | presentation | 3 |