CPE393: MLOps

HOUSE RENT PREDICTION

group_name

Faculty of Engineering, King Mongkut's University of Technology Thonburi 2024

TEAM MEMBERS

Deployment

Jednipat Kemawat
64070501011

Web development

Chayarob Chantrapiwat
64070501015

Model Training

Techathat Sakulsak
64070501064

Preprocessing

Sarunyarat Wongsason
64070501086

Model Training

Worapol Khunaekanan
64070501097

INTRODUCTION

PROBLEM STATEMENT

Goal

- To predict rental prices for residential properties in six major Indian cities:
  - Mumbai, Chennai, Bangalore, Hyderabad, Delhi, and Kolkata.

Problem

- Manually estimating fair rental prices is difficult due to varying property features, market demand, and local economic factors.

Solution:

- Develop a user-friendly machine learning model that uses key property details to generate accurate rent predictions to help users and clients quickly and easily estimate a fair rental value without needing deep market knowledge.

MOTIVATION & IMPACT

- Tenants can make better choices by comparing predicted rents with listed prices.
- Landlords and agents can confidently set competitive prices based on data, reducing time to find tenants.
- Researchers and policymakers can analyze housing affordability and trends.

SCOPES

- **Data Collection:**
  - Acquiring relevant datasets, from Kaggle
- **Data Preprocessing:**
  - Cleaning and transforming raw data
- **Model Training & Evaluation:**
  - Applying machine learning algorithms and evaluating performance using standard metrics
- **Experiment Tracking:**
  - MLflow to log experiments
- **Model Deployment:**
  - Deploying the final trained model as a production-ready API

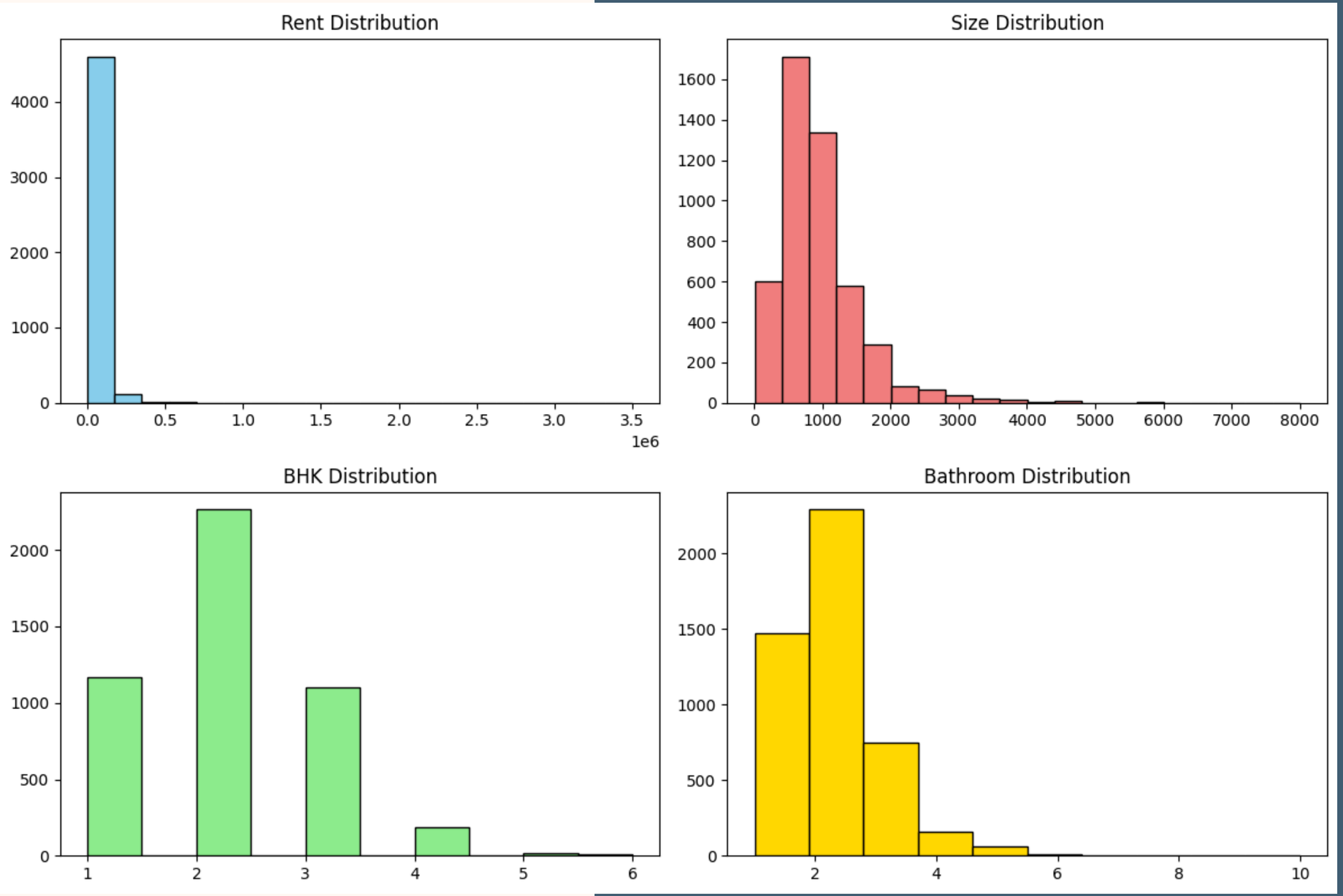
DATA COLLECTION

DATASET

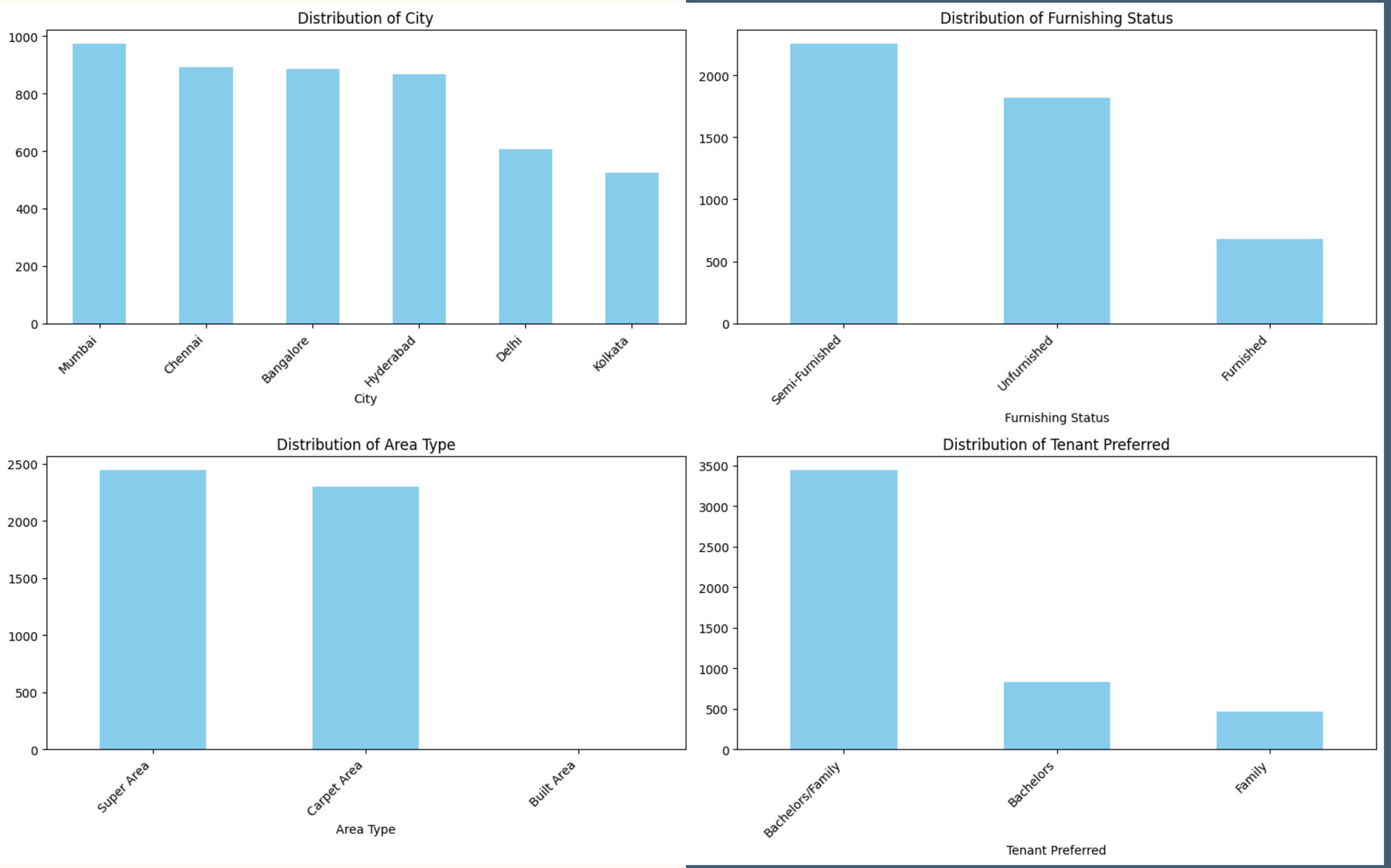
	Posted On	BHK	Rent	Size	Floor	Area Type	Area Locality	City	Furnishing Status	Tenant Preferred	Bathroom	Point of Contact
0	2022-05-18	2	10000	1100	Ground out of 2	Super Area	Bandel	Kolkata	Unfurnished	Bachelors/Family	2	Contact Owner
1	2022-05-13	2	20000	800	1 out of 3	Super Area	Phool Bagan, Kankurgachi	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
2	2022-05-16	2	17000	1000	1 out of 3	Super Area	Salt Lake City Sector 2	Kolkata	Semi-Furnished	Bachelors/Family	1	Contact Owner
3	2022-07-04	2	10000	800	1 out of 2	Super Area	Dumdum Park	Kolkata	Unfurnished	Bachelors/Family	1	Contact Owner
4	2022-05-09	2	7500	850	1 out of 2	Carpet Area	South Dum Dum	Kolkata	Unfurnished	Bachelors	1	Contact Owner

(4746, 12)

EDA


Rent Distribution
4000
3000
2000
1000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1e6
Size Distribution
1600
1400
1200
1000
800
600
400
200
0
0
1000
2000
3000
4000
5000
6000
7000
8000
BHK Distribution
2000
1500
1000
500
0
1
2
3
4
5
6
Bathroom Distribution
2000
1500
1000
500
0
2
4
6
8
10

EDA

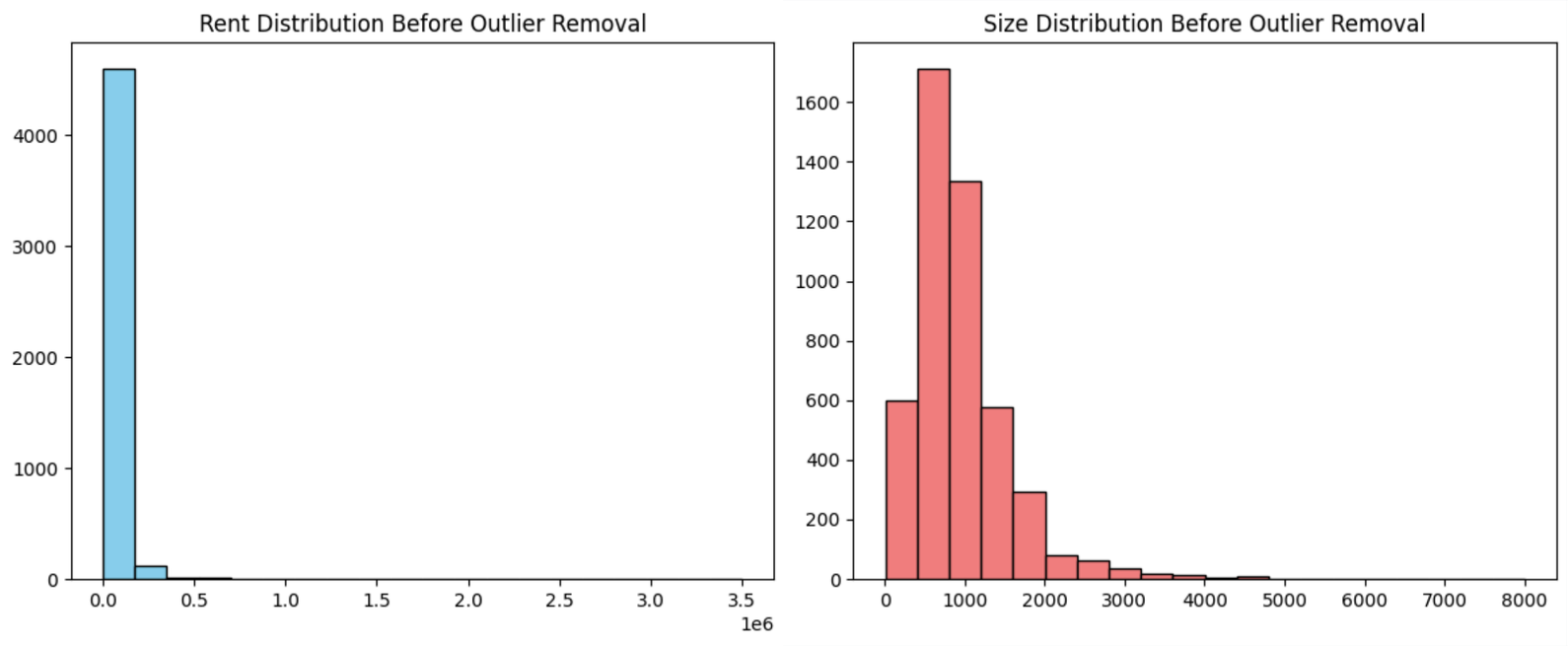

Distribution of City
1000
800
600
400
200
0
Mumbai
Chennai
Bangalore
Hyderabad
Delhi
Kolkata
City
Distribution of Furnishing Status
2000
1500
1000
500
0
Semi-Furnished
Unfurnished
Furnished
Furnishing Status
Distribution of Area Type
2500
2000
1500
1000
500
0
Super Area
Carpet Area
Built Area
Area Type
Distribution of Tenant Preferred
3500
3000
2500
2000
1500
1000
500
0
Bachelors/Family
Bachelors
Family
Tenant Preferred

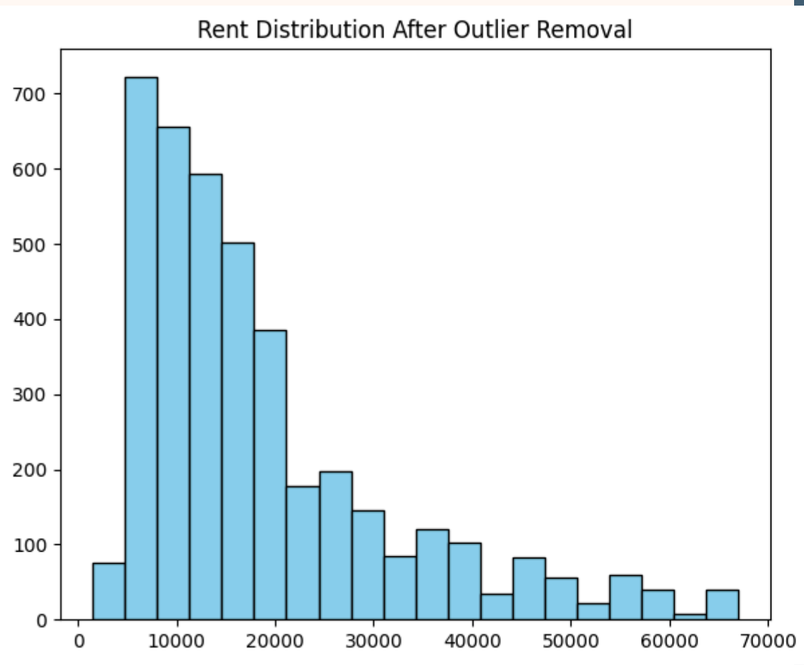
DATA CLEANING

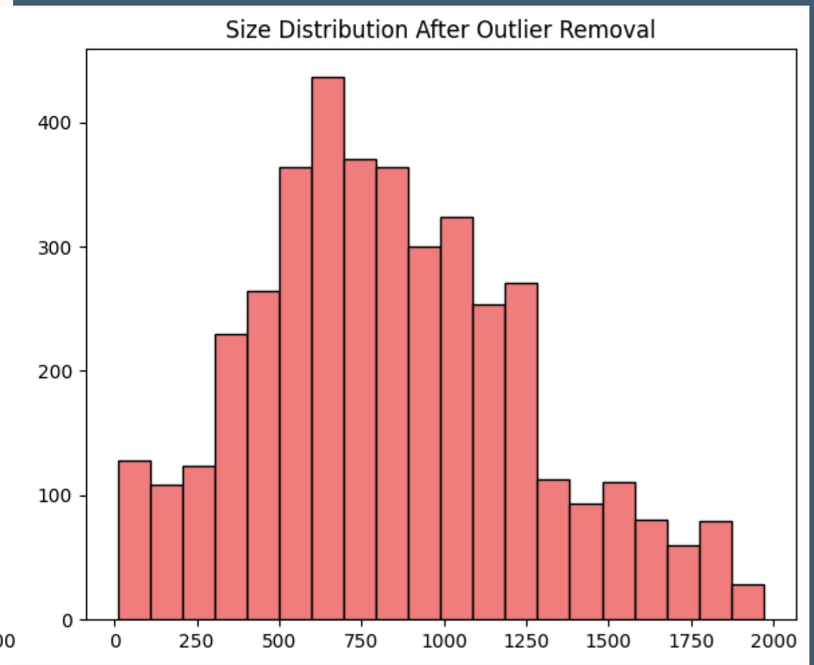
- OUTLIER REMOVAL -

```python
def remove_outliers_iqr(data, column, threshold=1.5):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - threshold * IQR
    upper_bound = Q3 + threshold * IQR
    return data[(data[column] >= lower_bound) & (data[column] <= upper_bound)]
```

OUTLIER REMOVAL







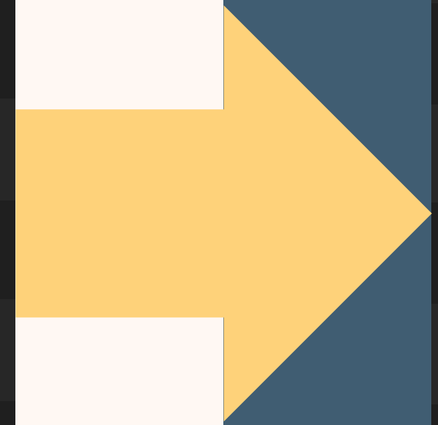
DATA CLEANING

- EXTRACT FLOOR INFORMATION -

```
def extract_floor_info(self,df):
        pattern = r'(?P<CurrentFloor>\w+)\s*out\s*of\s*(?P<TotalFloors>\d+)'
        df[['CurrentFloor', 'TotalFloors']] = df['Floor'].str.extract(pattern)
        floor_replacements = {
            'Ground': 0,
            'Basement': -1
        }

        df['CurrentFloor'] = df['CurrentFloor'].fillna(df['Floor'])
        df['TotalFloors'] = df['TotalFloors'].fillna(df['Floor'])

        df['CurrentFloor'] = df['CurrentFloor'].replace(floor_replacements)
        df['TotalFloors'] = df['TotalFloors'].replace(floor_replacements)

        df['CurrentFloor'] = pd.to_numeric(df['CurrentFloor'], errors='coerce').astype('Int64')
        df['TotalFloors'] = pd.to_numeric(df['TotalFloors'], errors='coerce').astype('Int64')

        df = df.drop(columns=['Floor'])
        return df
```

EXTRACT FLOOR INFORMATION

	Floor
0	Ground out of 2
1	1 out of 3
2	1 out of 3
3	1 out of 2
4	1 out of 2
5	Ground out of 1
6	Ground out of 4
7	1 out of 2
8	1 out of 2
9	1 out of 3



	CurrentFloor	TotalFloors
0	0	2
1	1	3
2	1	3
3	1	2
4	1	2
5	0	1
6	0	4
7	1	2
8	1	2
9	1	3

DATA CLEANING

- DROPPING UNNECESSARY COLUMNS -

‘Posted On’

This column represents the date a property listing was posted. It does not influence the intrinsic rental value of a property and may introduce noise or unnecessary temporal bias into the model.

‘Area Locality’

Although it specifies the exact neighborhood, **it contains over 2000 unique values,** making it too granular and potentially leading to overfitting. Instead, we rely on broader location features (like city) that generalize better across the dataset.

FEATURE ENGINEERING

Categorical Feature

Transformed using one-hot encoding, converting each category into binary columns (0 or 1). This allows machine learning models to process categorical data effectively while preserving the underlying information.

Numerical Features

Scaled using StandardScaler to standardize the data (mean = 0, variance = 1). This ensures balanced feature contribution and improves model performance.

MODEL TRAINING

MODELS

RIDGE REGRESSION

RANDOM FOREST

NEURAL NETWORK

TRACKING

mlflow

Run Name	Created	Dataset	Duration	Source	Models
Test	38 minutes ago	-		C:\Users...	-
Ridge	57 minutes ago	dataset (1d6e4ed6) Train	12.1min	C:\Users...	Ridge v1 +1
RandomForest	1 hour ago	dataset (1d6e4ed6) Train	5.9min	C:\Users...	Random Forest v1 +1
NeuralNetwork	1 day ago	-	18.1min	C:\Users...	-
NeuralNetwork27	1 day ago	dataset (1d6e4ed6) Train	13.7s	C:\Users...	tensorflow
NeuralNetwork26	1 day ago	dataset (1d6e4ed6) Train	19.4s	C:\Users...	Neural Network v1
NeuralNetwork25	1 day ago	dataset (1d6e4ed6) Train	1.0min	C:\Users...	tensorflow
NeuralNetwork24	1 day ago	dataset (1d6e4ed6) Train	12.2s	C:\Users...	tensorflow
NeuralNetwork23	1 day ago	dataset (1d6e4ed6) Train	21.3s	C:\Users...	tensorflow
NeuralNetwork22	1 day ago	dataset (1d6e4ed6) Train	1.0min	C:\Users...	tensorflow
NeuralNetwork21	1 day ago	dataset (1d6e4ed6) Train	12.5s	C:\Users...	tensorflow
NeuralNetwork20	1 day ago	dataset (1d6e4ed6) Train	20.6s	C:\Users...	tensorflow
NeuralNetwork19	1 day ago	dataset (1d6e4ed6) Train	54.7s	C:\Users...	tensorflow

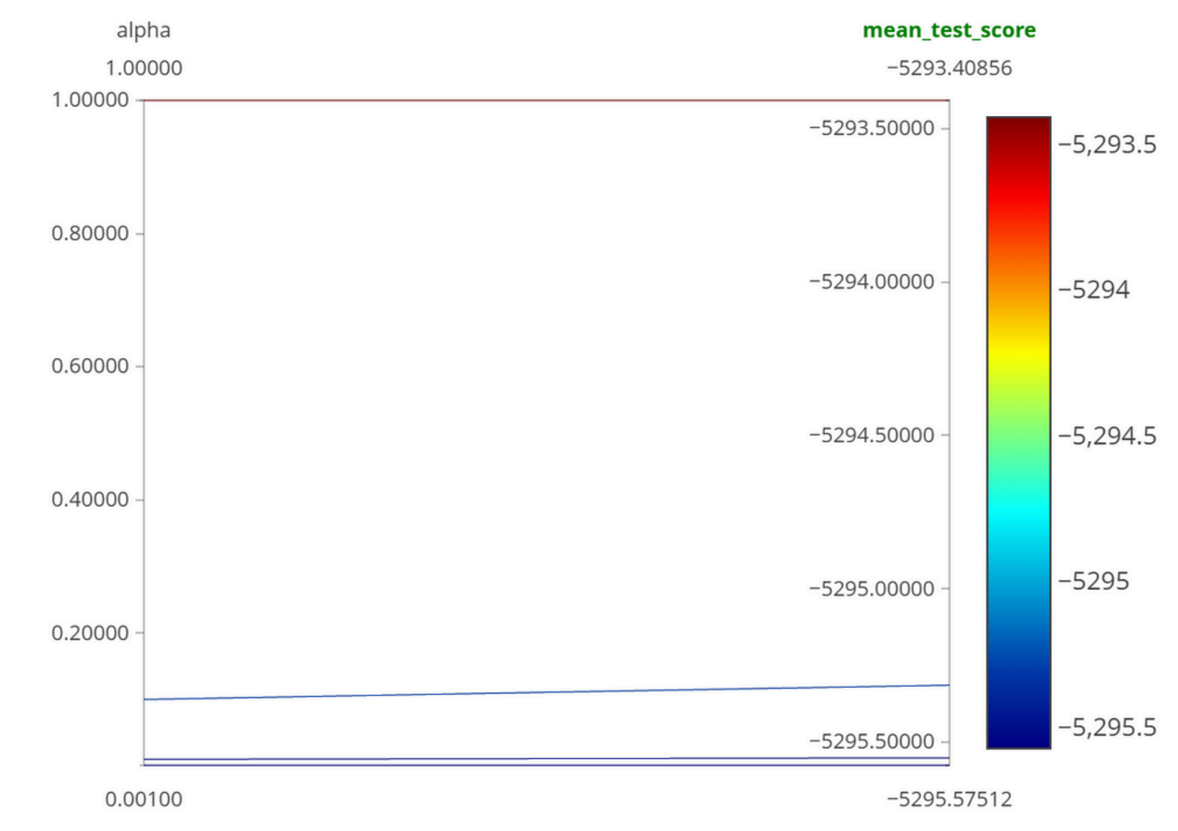
43 matching runs

HYPERPARAMETER TUNING

RIDGE REGRESSION

- GridSearchCV
- Hyperparameter search space:
  - **alpha**: 0.001, 0.01, 0.1, 1.0
- Use neg_mean_absolute_error as metric
- Log all 4 models into MLflow

Best performing model is **alpha: 1.0**



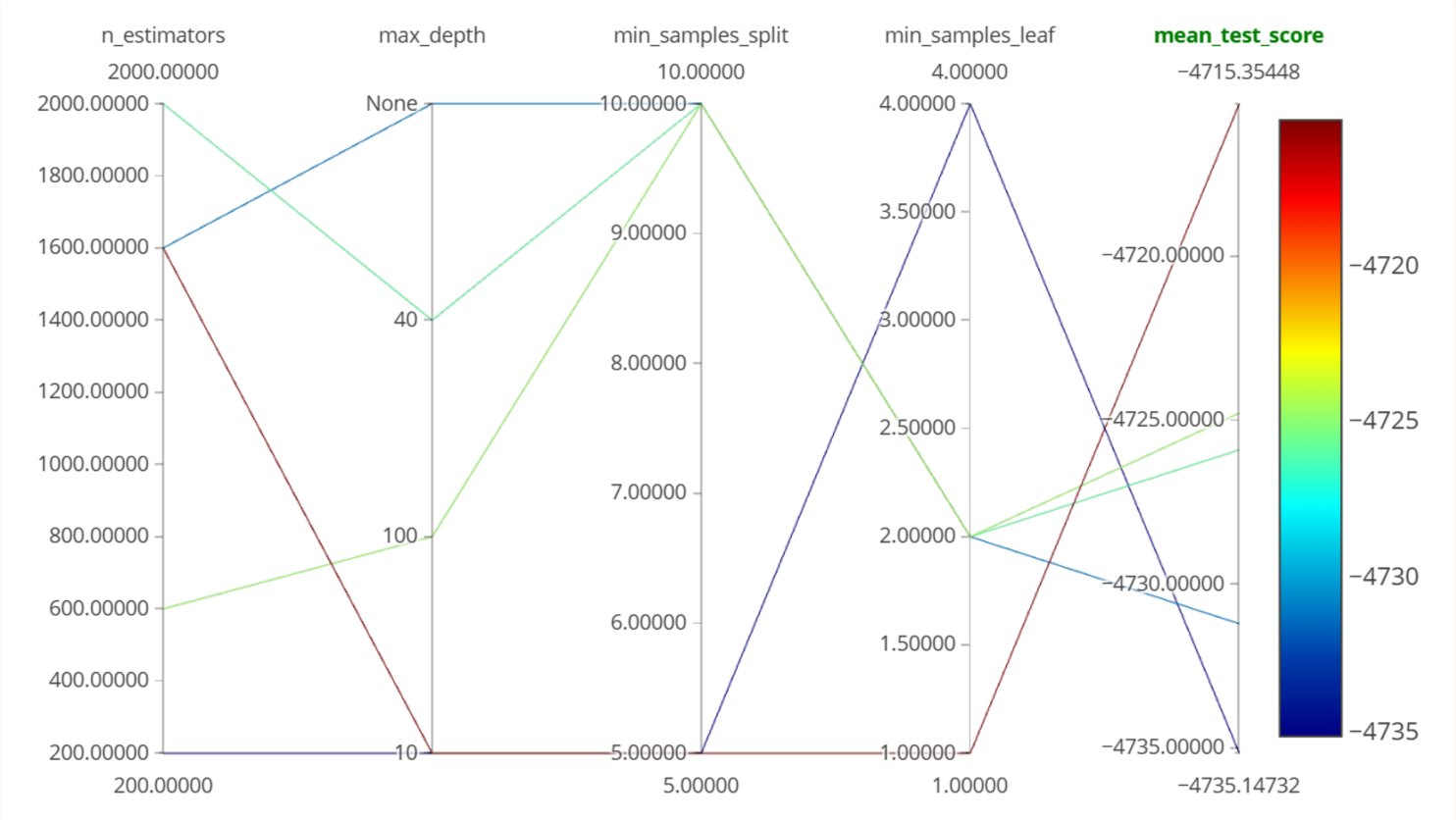
HYPERPARAMETER TUNING

RANDOM FOREST

- RandomSearchCV
- Hyperparameter search space:
  - **n_estimators**: 200 - 2000 (200 step)
  - **max_depth**: 10 - 110 (10 step), None
  - **min_samples_split**: 2, 5, 10
  - **min_samples_leaf**: 1, 2, 4
- Use neg_mean_absolute_error as metric
- Log best 5 models into MLflow

Best performing model is

n_estimators: 1600
max_depth: None
min_samples_split: 10
min_samples_leaf: 2



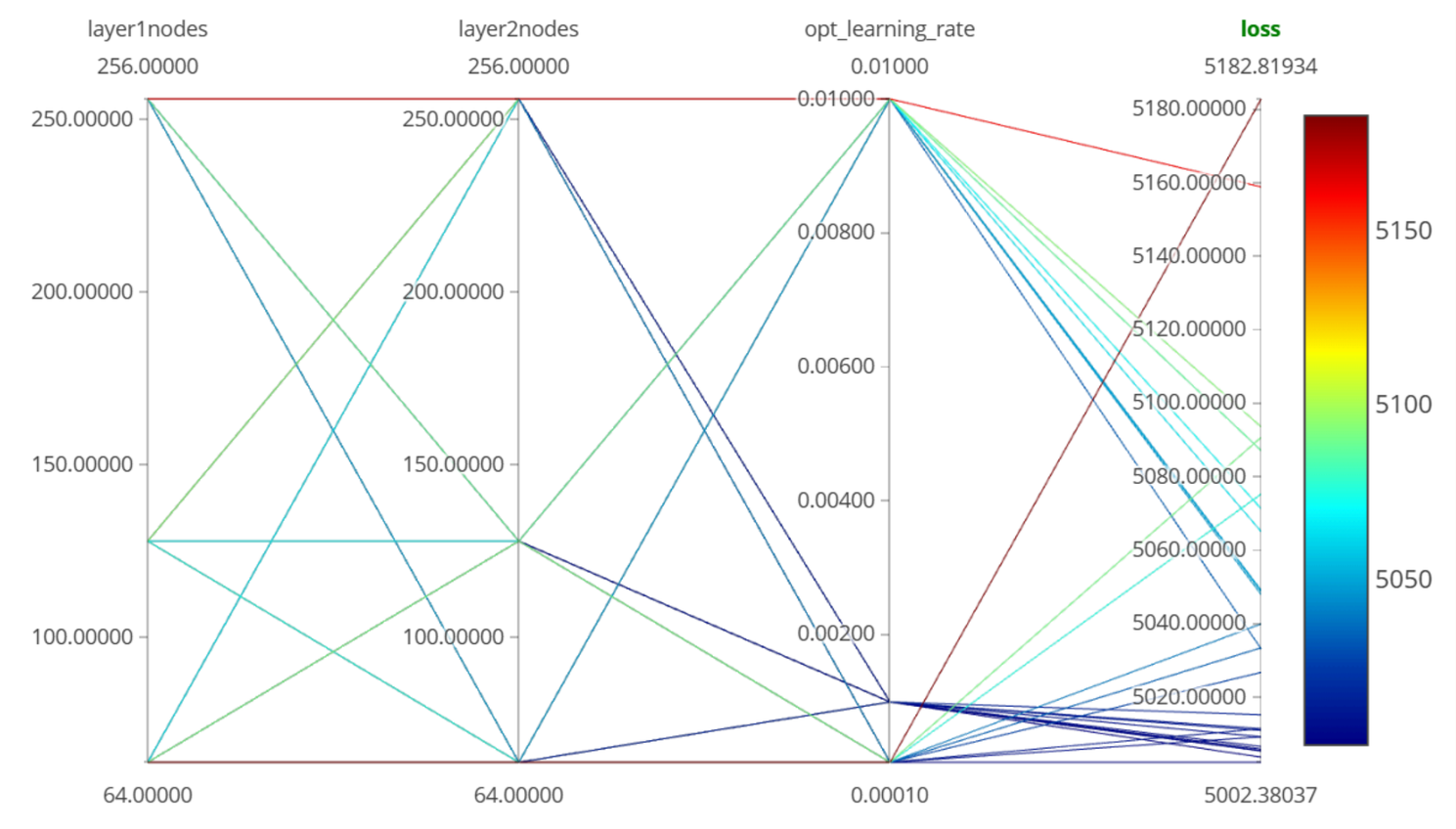
HYPERPARAMETER TUNING

NEURAL NETWORK

- 2-layer Dense Neural Network
- Hyperparameter search space:
  - **layer1_node**: 64, 128, 256
  - **layer2_node**: 64, 128, 256
  - **learning_rate**: 0.0001, 0.001, 0.01
- Use loss as metric
- Log all 27 models into MLflow

Best performing model is

layer1_node: 256
layer2_node: 256
learning_rate: 0.001



EVALUATION

METRICS

- Mean Average Error

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

- Mean Sqaured Error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- Coefficient of determination (R^2)

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

EVALUATION

		Mean Squared Error (MSE)	Mean Average Error (MAE)	R-squared (R^2)
TRAIN	Ridge Regression	54334118.84	5256.04	0.69
	Random Forest	**19469060.27**	**3247.56**	**0.89**
	Neural Network	57514526.23	5102.12	0.67
TEST	Ridge Regression	51513004.86	5244.22	0.71
	Random Forest	**41514612.62**	**4452.25**	**0.77**
	Neural Network	53446249.89	5070.60	0.70

BEST MODEL SELECTION

RIDGE REGRESSION

RANDOM FOREST

NEURAL NETWORK

DEPLOYMENT

API SERVICE

POST
/predict

BODY


```
{
   "BHK": 1,
   "Size": 1000,
   "CurrentFloor": -1,
   "TotalFloors": 2,
   "Area Type": "Super Area",
   "City": "Mumbai",
   "Furnishing Status": "Semi-Furnished",
   "Tenant Preferred": "Family",
   "Bathroom": 1,
   "Point of Contact": "Contact Agent"
}
```


RESPONSE

HTTP 200 OK


```
{
   "prediction": {
      "prediction": 38245.0
   },
   "success": true
}
```

API SERVICE

POST
/predict

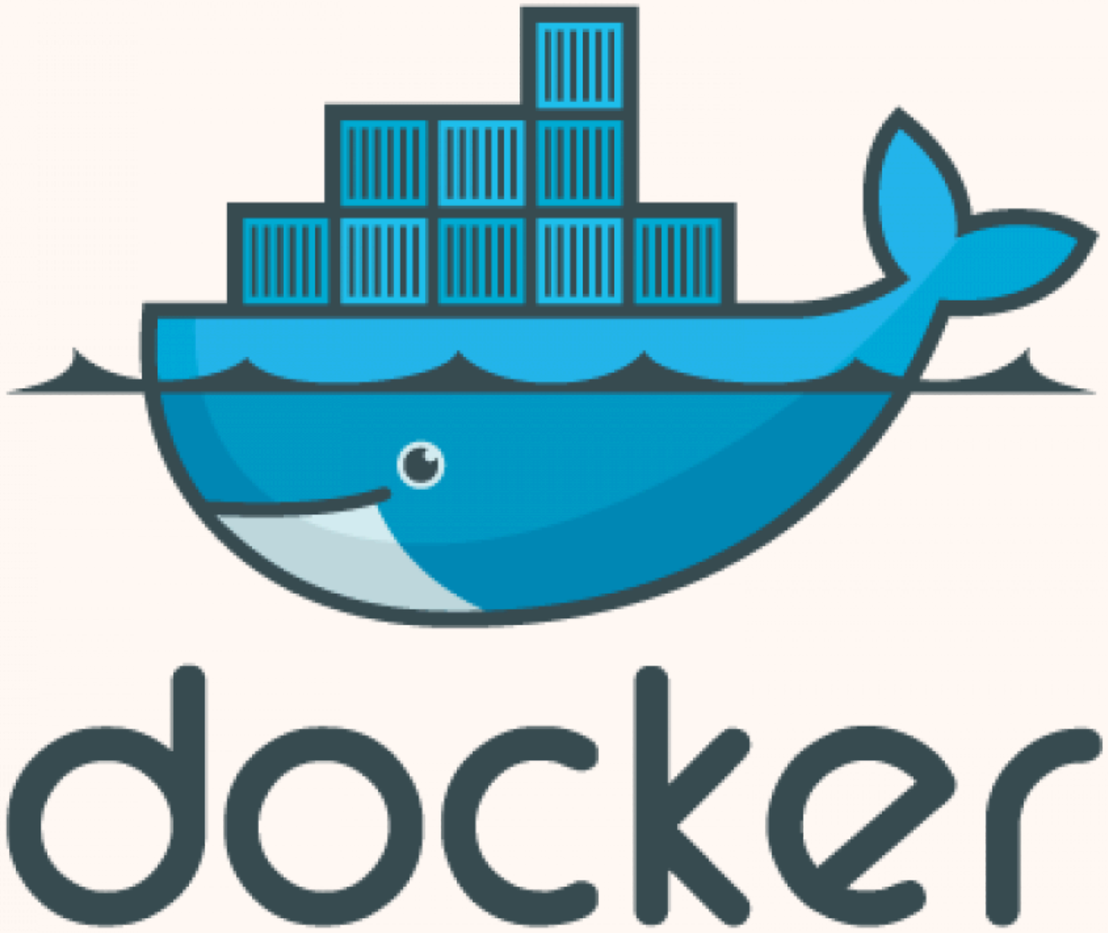
BODY


```
{
  "BHK": 1,
  "Size": 1000,
  "CurrentFloor": -1,
  "TotalFloors": 2,
  "Area Type": "Super Area",
  "City": "Mumbai",
  "Furnishing Status": "Semi-Furnished",
  "Tenant Preferred": "Family",
  "Bathroom": 1,
  "Point of Contact": "Contact Agent"
}
```


RESPONSE

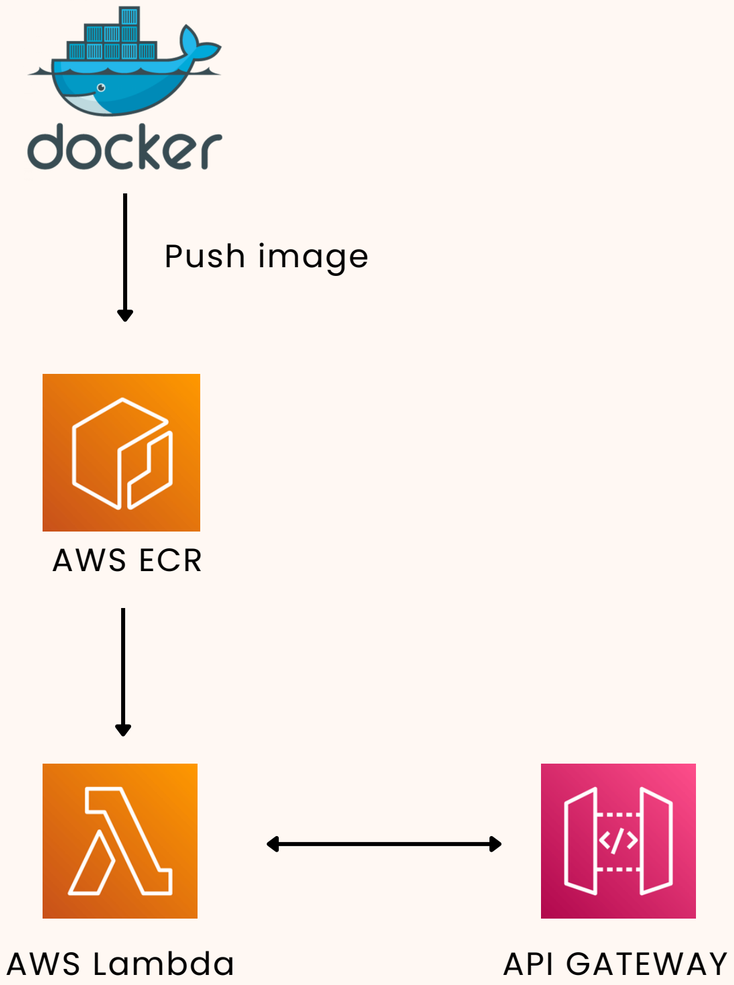
HTTP 400 BAD REQUEST


```
{

  "error": "Missing required field: Point of Contact"

}
```



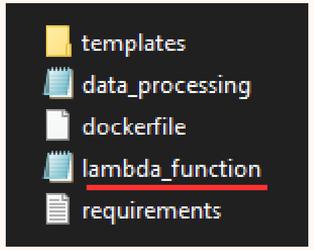
LOCAL DEPLOYMENT

- **Clone the repository:**
  - git clone https://github.com/WorapolKhu/CPE393-group_name
- **Build the Docker image:**
  - docker build -t rent-prediction .
- **Run the Docker container:**
  - docker run -p 5000:5000 rent-prediction
- **Access the application at:**
  - http://localhost:5000



AWS DEPLOYMENT

- Build & Push Docker Image
  - Upload Project ZIP to Cloud Shell
  - Tag and **Push** to **Amazon ECR**

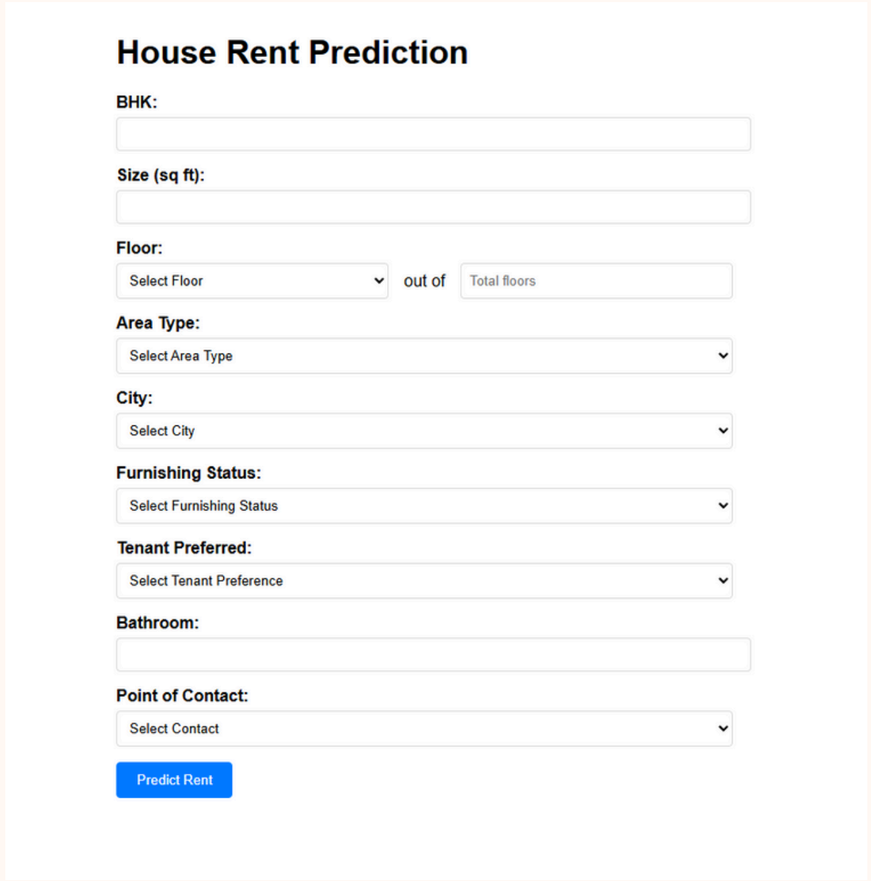


- Create AWS Lambda (Using Container Image)
  - Set handler: **lambda_function**.lambda_handler

- Create and Configure API Gateway
  - Create **REST API**
  - **Define Routes**
    - POST /predict → Model prediction endpoint

WEB-BASED API USING FLASK

Live Endpoint: https://wt9vo8xuke.execute-api.ap-southeast-1.amazonaws.com/dev



CHALLENGES AND POTENTIAL SOLUTIONS

DATA QUALITY AND INCONSISTENCIES

Missing & inconsistent data

- Data cleaning
- Imputation
- Encoding (OneHotEncoding)
- Normalized numeric data (MinMaxScaler)

LIMITED DATASET SIZE (~4,700 SAMPLES)

Risk of overfitting, reduced generalization

- Feature engineering
- Selection of Random Forest (performs well on smaller datasets)

CONTAINERIZATION AND CLOUD INTEGRATION

Dependency management, Lambda cold starts, API Gateway setup

- Multi-stage Dockerfile
- Lightweight libraries
- Leveraging Amazon ECR/Lambda container support
- API Gateway proxy integration

REPRODUCIBILITY ACROSS ENVIRONMENT

Inconsistencies between local and cloud execution

- Docker containers
- Saved preprocessing artifacts
- Used requirements.txt & virtual environments to ensure dependency consistency

LESSONS LEARNED

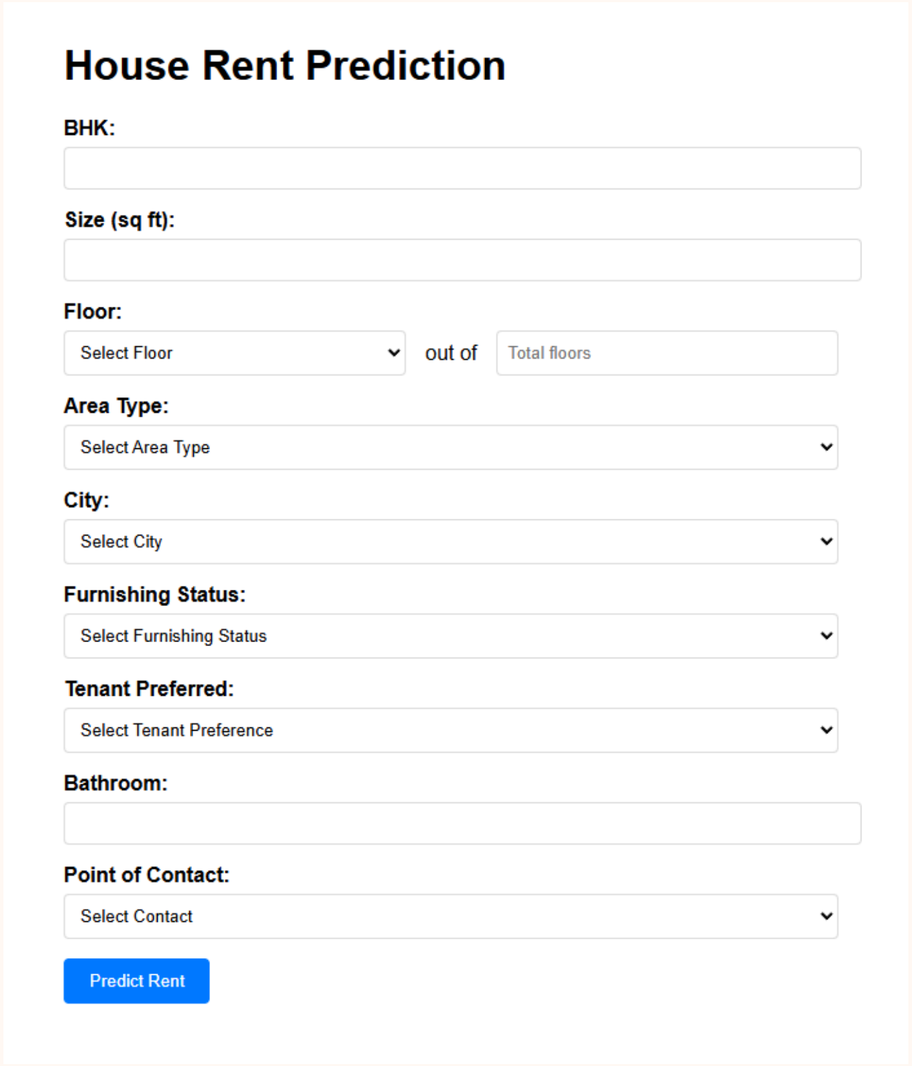
- Start Simple Model
- Data Quality is Critical
- Ensure Reproducibility
- Structured Collaboration

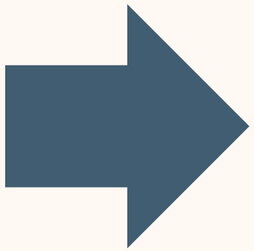
FUTURE WORK

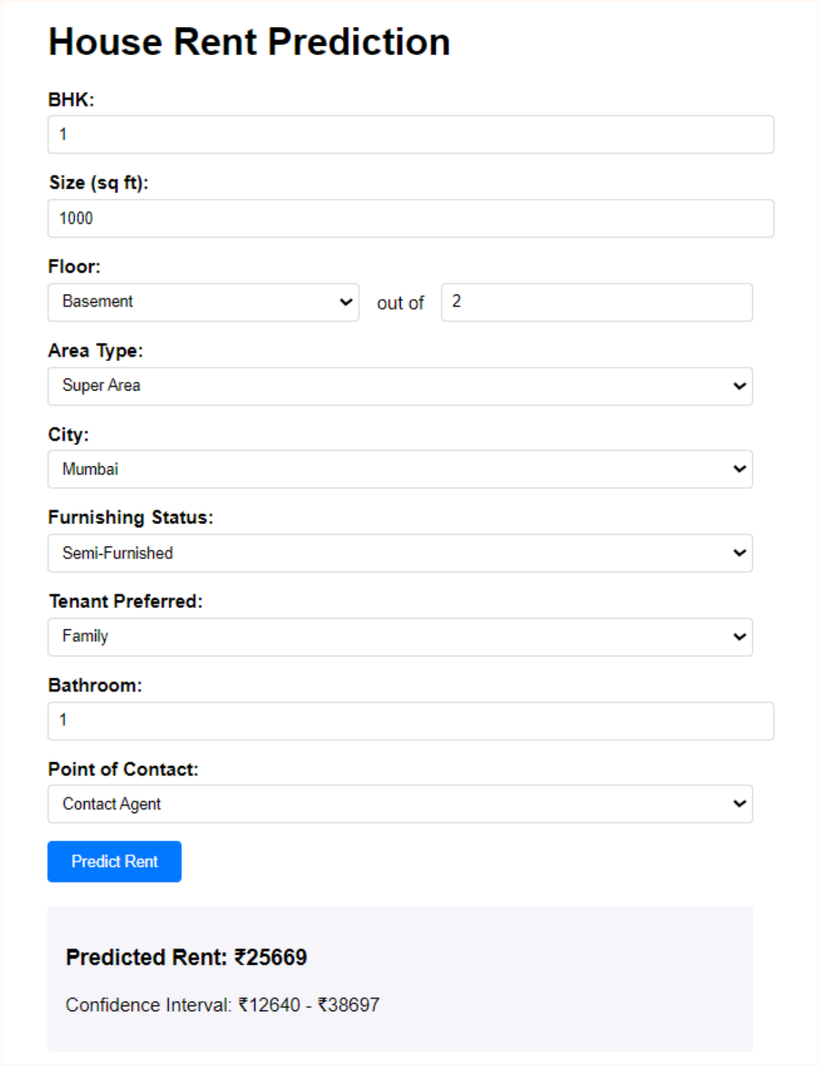
Dataset Expansion
Augmenting the dataset or integrating external data sources. This could potentially improve model performance and generalizability further.

DEMO

Live Endpoint: https://wt9vo8xuke.execute-api.ap-southeast-1.amazonaws.com/dev







REFERENCES

- Amazon ECR | Docker Container Registry | Amazon Web Services. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/ecr/
- AWS. (2019). AWS Lambda – Serverless Compute. Amazon Web Services, Inc. https://aws.amazon.com/lambda/
- Banerjee, S. (2022, August 20). House Rent Prediction Dataset. Kaggle. https://www.kaggle.com/datasets/iamsouravbanerjee/house-rent-prediction-dataset
- GDPR Implementation. (2020). Magicbricks.com. https://www.magicbricks.com/
- GitHub. (2025). GitHub. https://github.com/
- Keras. (2019). Home - Keras Documentation. Keras.io. https://keras.io/
- MLflow | MLflow. (2025). Mlflow.org. https://www.mlflow.org/
- Scikit-learn. (2019). scikit-learn: Machine learning in Python — scikit-learn 0.20.3 documentation. Scikit-Learn.org. https://scikit-learn.org/stable/index.html
- TensorFlow. (2019). TensorFlow. Google. https://www.tensorflow.org/

CODE REPOSITORY

https://github.com/WorapolKhu/CPE393-group_name.git

Q & A