Housing Price Prediction Model Development and Deployment in Cloud

Table of Contents

1) Introduction

In this project, a machine-learning model to predict housing prices in DFW metropolitan area was developed and deployed to the cloud. The idea was to use some key characteristics of house and neighborhood features for the prediction. No data was provided and it was the individual's responsibility to acquire data and perform extract, transform, and load processes. The main Python packages that were used in this project are 'Scikit-Learn', 'Pandas', 'googlemaps', and 'numpy'.

Objectives:

- Build a machine-learning model to predict house prices using a set of features
- Develop a flask-web app that utilizes the best-trained model to predict the price based on user data
- Deploy the flask-webapp to the cloud so that users can use it to fill in the data and obtain predicted prices real-time and also use REST API to obtain results real-time.

2) Model development using a machine learning technique

2.1. Data extraction from web scraping

Since no data were provided, house-related data were extracted from the trillow.com website using Python modules. Four different features 'size', 'bed', 'bath', 'address', and 'price' were scraped out. Altogether 709 records were scraped.

Using google APIs, a few more features as listed below were computed. These features represent the shortest distance to the places from a given house

* Distance to supermarket
* Distance to school
* Distance to park
* Distance to hospital

2.2. Data cleaning and sanitizing

Some unnecessary columns were removed, and datatype formats were corrected. Three more boolean features 'isPool', 'isBackyard', and 'isGarage' were added using some random rule-based thresholds on price. Since it was asked to add missing values, missing values were added randomly to all of the features. Finally, two different data imputation techniques for numerical and Boolean values were applied to impute the values

2.3. Data preprocessing

Two more features were created 'housequality' and 'conviniencefactor' using the ['size', 'bed', and 'bath] and all of the distance-based features, respectively. Since the dataset had outliers, the interquartile range 0.3 – 0.7 was used instead of the common range (0.25 – 0.75) for the dataset. The common range implementation still resulted in outliers so 0.3-0.7 was used. After outlier removal, only 350 rows of data remained. The boolean features were label encoded before further processing. Finally, two different data scaling techniques: minmax scaler (minmax) and standardization scaler (std) were applied separately to experiment with their impact on model performance.

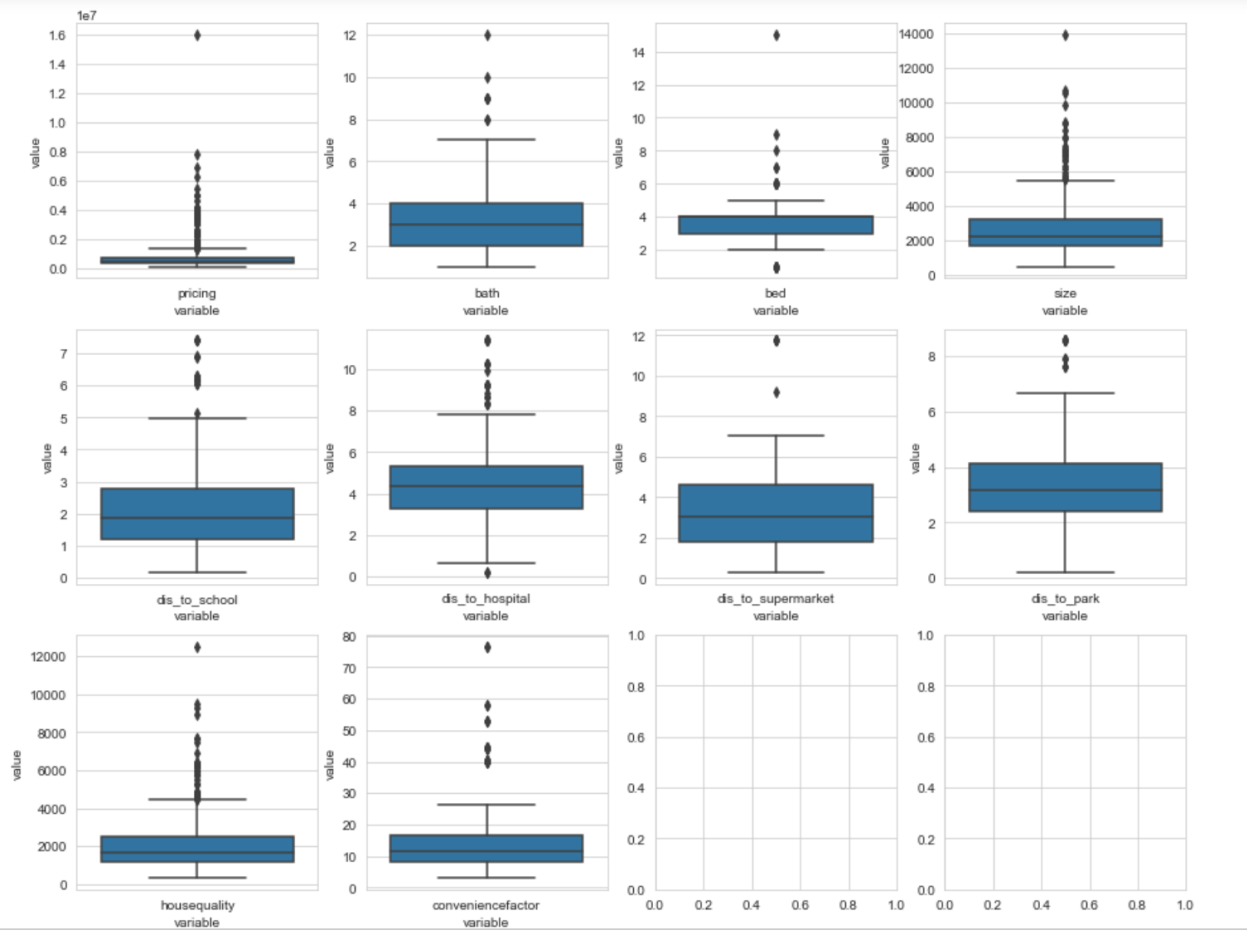


Fig.1. Features before outlier removal

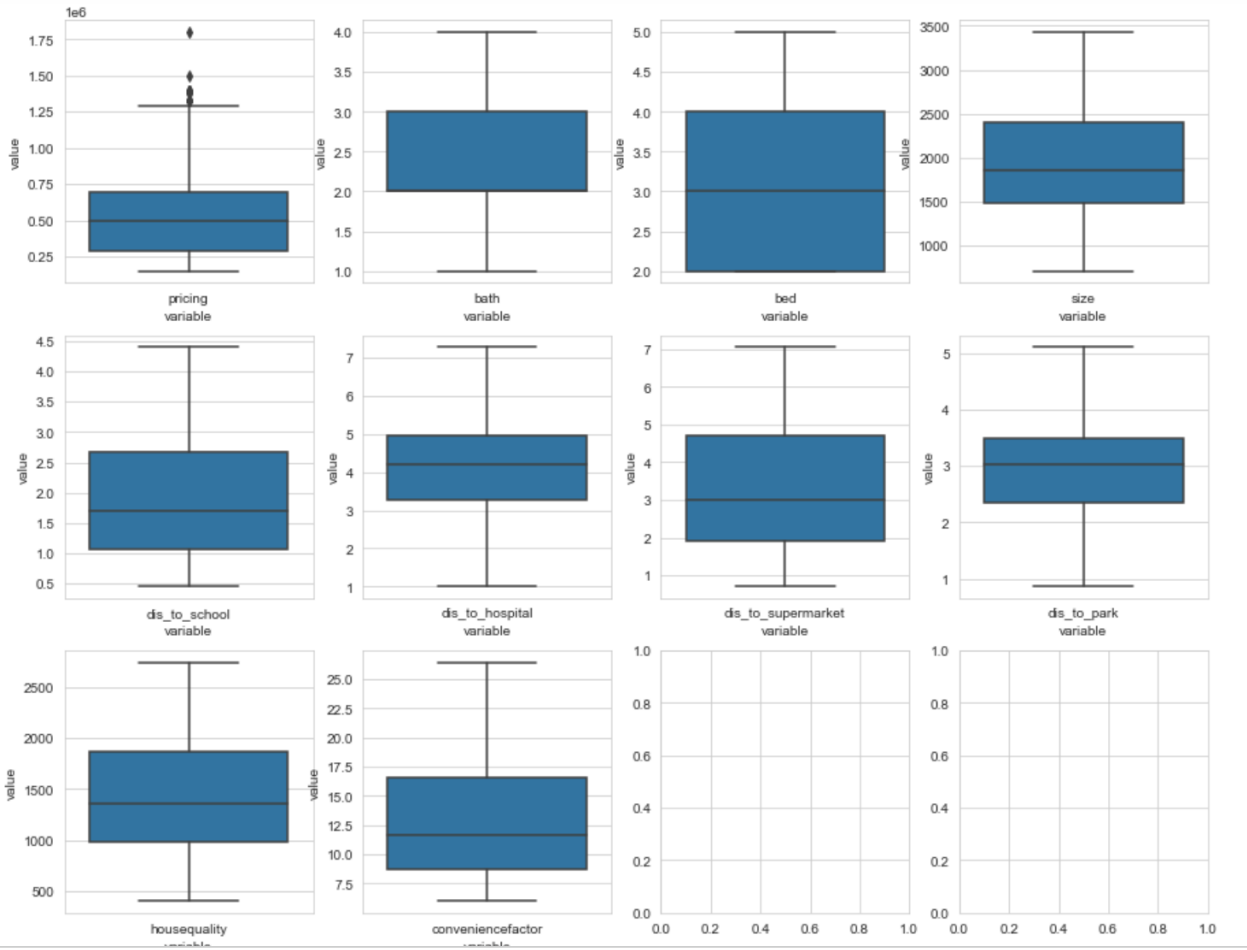


Fig 2. Features after outlier removal

2.4. Feature engineering

Altogether, there were 13 different features (10 numerical features & 3 boolean features)

pricing	bath	bed	size	dis_to_school	dis_to_hospital	dis_to_supermarket	dis_to_park	isPool	isBackyard	isGarage	housequality	conveniencefac
359000	0.333333	0.666667	0.450424	0.281726	0.377573	0.274882	0.165094	0	0	0	0.393654	0.426

The target variable was ‘pricing’. Two different feature selection (FS) schemes: mutual-information regression (mic) & extratreeclassifier (etc) were used to select features for experimentation. It was of interest to find the impact of FS schemes on the model performance. FS was performed on data scaled by two DS methods separately for experimentation. So, there were altogether 4 sets of features were selected

```
features selected with mic_minmax: ['size', 'dis_to_school', 'dis_to_hospital', 'dis_to_park', 'housequality']
features selected with etc_minmax: ['size', 'dis_to_school', 'dis_to_supermarket', 'dis_to_park', 'housequality']
features selected with mic_std: ['size', 'dis_to_hospital', 'dis_to_supermarket', 'dis_to_park', 'housequality']
features selected with etc_std: ['size', 'dis_to_school', 'dis_to_supermarket', 'housequality', 'conveniencefactor']
```

2.5. Hyperparameter tuning, training, and validation

The research method was used with xgboost as a regressor for hyperparameter tuning and model training. The tuning and training process was performed separately for six modeling schemes as listed below :

- All features from minmax scaled data
- Mic-selected features from minmax scaled data
- Etc-selected features from minmax scaled data
- All features from standardized data
- Mic-selected features from standardized scaled data
- Etc-selected features from standardized scaled data

For each modeling scheme, 140 combinations of parameters were trained to find the optimal set. Using the optimal combination, predictions on the test dataset were made and rmse and nrmse values were calculated

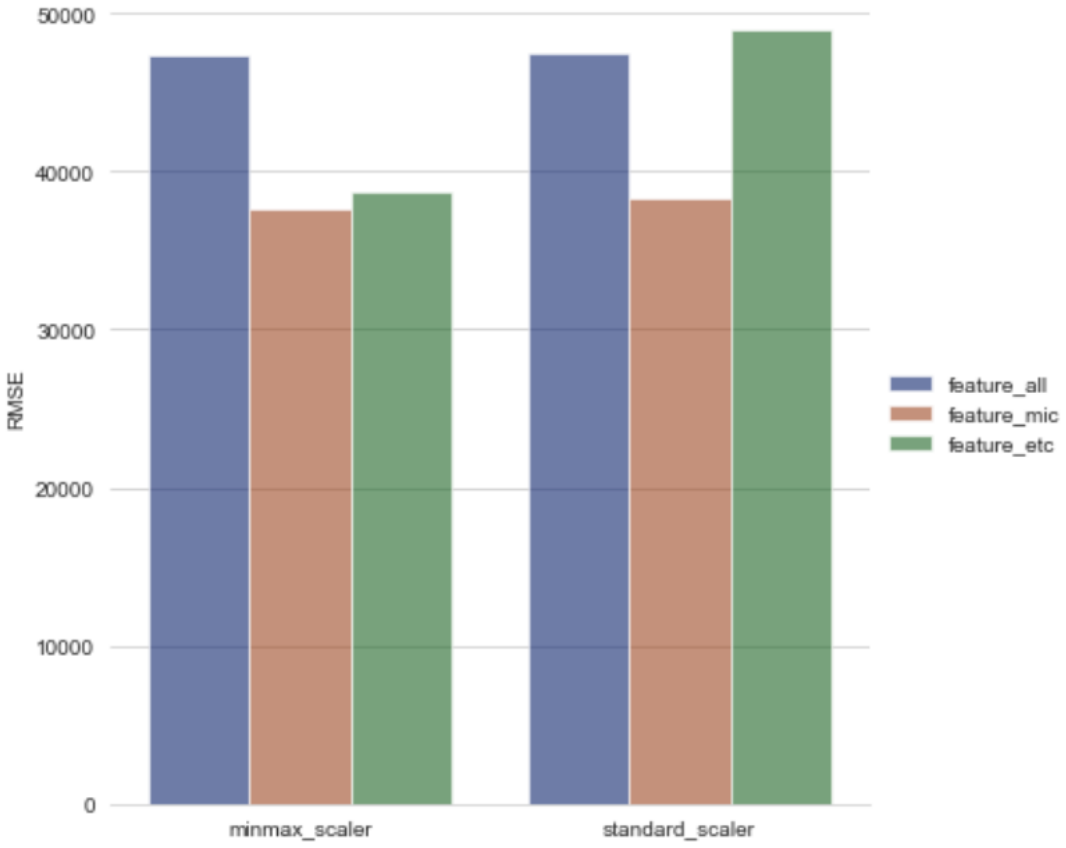


Fig 3. Barchart for RMSE values

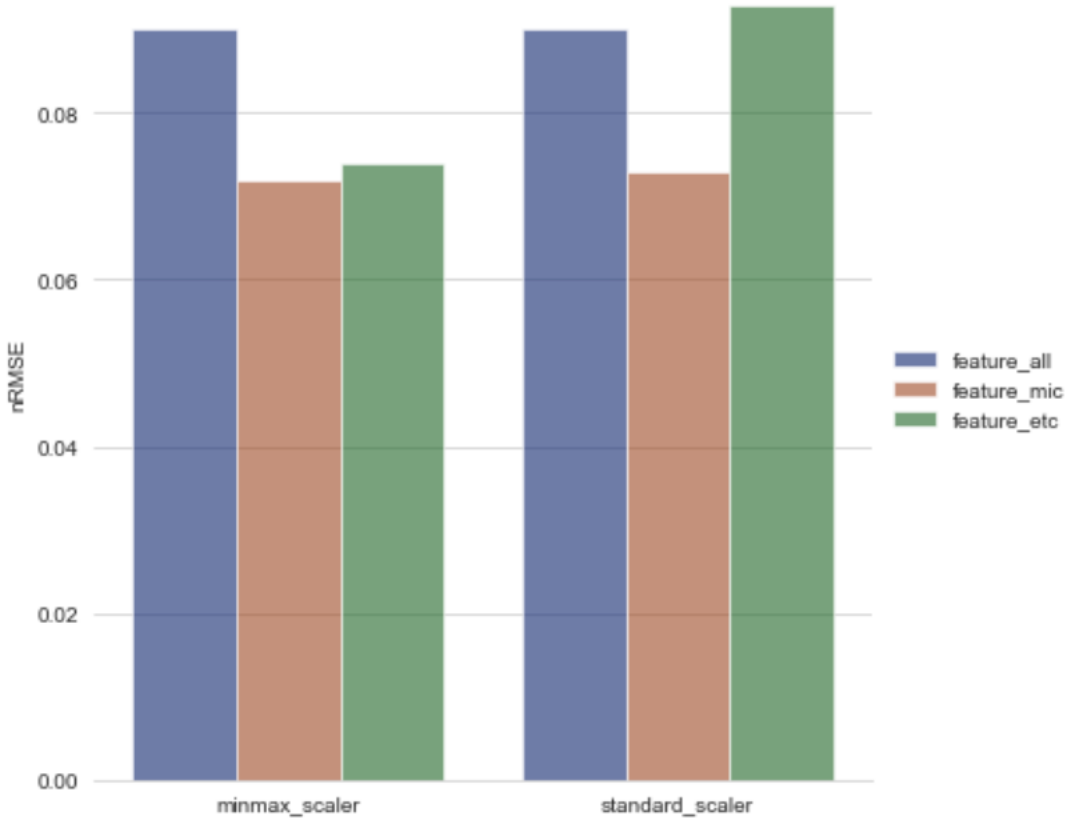


Fig 4. Barchart for nRMSE values

The best result was obtained with “Mic-selected features from minmax scaled data” scheme. The selected features were

```
['size', 'dis_to_school', 'dis_to_hospital', 'dis_to_park', 'housequality']
```

2.6. Insights gained

The results show that housing prices can be predicted using web-available data. The required features can mostly be web-scraped. Housing price is determined by not only the house features but also by the characteristics of the neighborhood. Out of the 5 best features, 3 features were related to neighborhood features. I think this is a great finding for real estate agents. It was also understood from the experimentation of different outlier thresholds that failing to remove outlier properly could result in the generalization of the model and thus inaccurate estimation of values. Also, it is found that it is important to test out different feature selection schemes as they can exert different impacts on model performance.

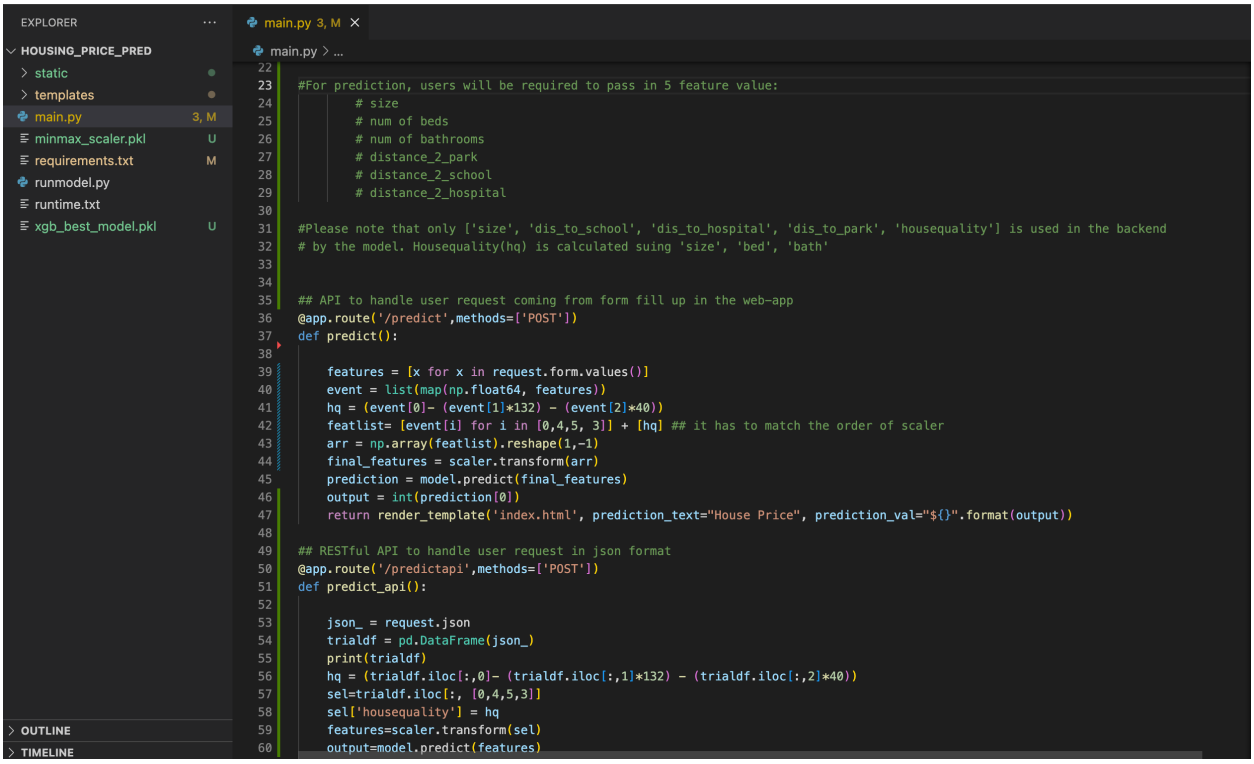
2.7 Challenges faced

The main challenge faced was during the web-scaping phase. Zillow had API limits which limited the ability to grab the data. Trillow was relatively easy to scrape into. Also, since the dataset had mixed types of variables (numerical and Boolean), methods had to be designed to handle them separately during imputation, scaling, and outlier removal.

3) Flask-web app development

Flask web app was developed using the Python-based flask framework. Two different APIs (predict, and predictable) were created.

predict: to predict over the data coming in from the form fill up in the webapp
predictapi: to predict over the json data coming as REST API hit.



4) Flask-web app deployment to the cloud

AWS t2.micro free instance was used for the virtual server to run the Flask app.

```
ubuntu@ip-172-31-85-122:~/housing_price_pred$ python3 main.py
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:347: InconsistentVersionWarning: Trying to unpickle estimator MinMaxScaler from version 0.24.2 when u
 code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
 * Serving Flask app 'main'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8080
 * Running on http://172.31.85.122:8080
```

Gunicorn production server was used to communicate with Flask though WSGI protocol. Nginx was used for reverse proxy and load balancing purpose.

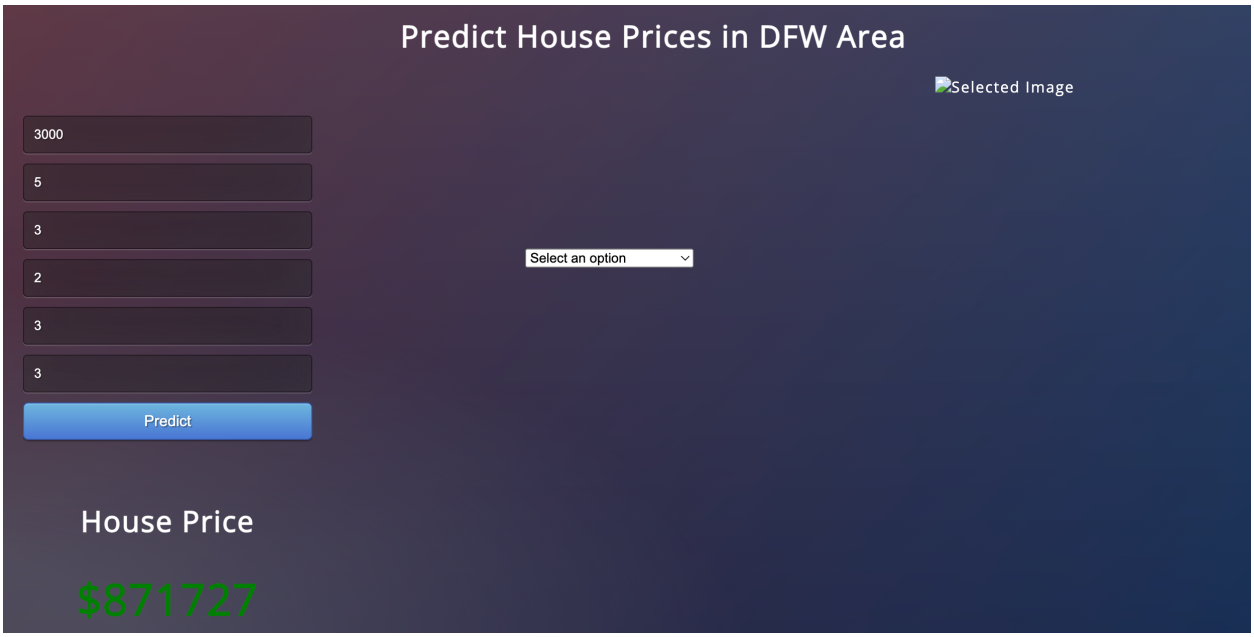
```
ubuntu@ip-172-31-85-122:~/housing_price_pred$ gunicorn -b 0.0.0.0:8000 main:app
[2023-08-04 05:17:33 +0000] [7124] [INFO] Starting gunicorn 21.2.0
[2023-08-04 05:17:33 +0000] [7124] [INFO] Listening at: http://0.0.0.0:8000 (7124)
[2023-08-04 05:17:33 +0000] [7124] [INFO] Using worker: sync
[2023-08-04 05:17:33 +0000] [7125] [INFO] Booting worker with pid: 7125
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:347: InconsistentVersionWarning: Trying to unpickle estimator MinMaxScaler from version 0.2
 code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
  warnings.warn(
^C[2023-08-04 05:26:17 +0000] [7124] [INFO] Handling signal: int
[2023-08-04 05:26:17 +0000] [7125] [INFO] Worker exiting (pid: 7125)
[2023-08-04 05:26:17 +0000] [7124] [INFO] Shutting down: Master
```

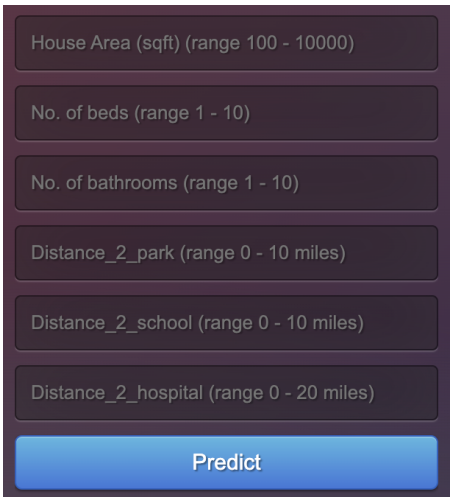
5) Demonstration of prediction on a new set of data using GUI interface

!!! Please excuse the rudimentary layout and design !!!

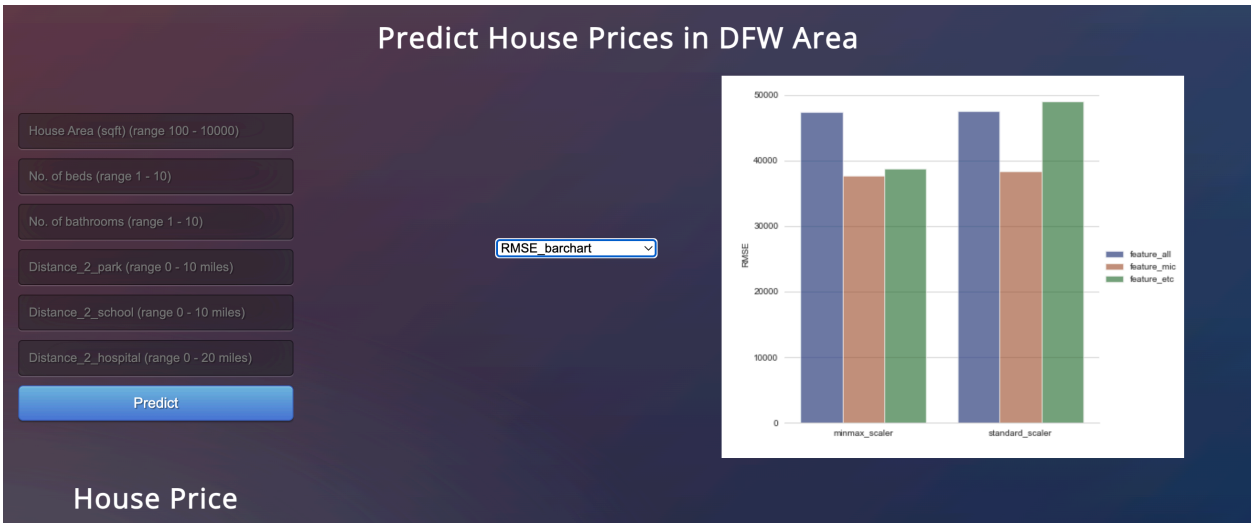
http://ec2-3-83-68-187.compute-1.amazonaws.com/

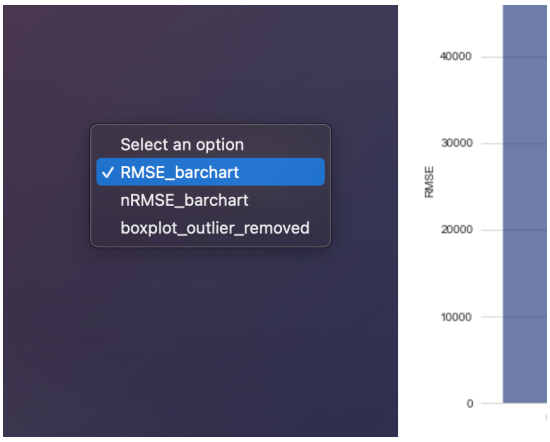
(Also, didn't get chance for DNS management)





Users can also visualize the accuracy metrics obtained during experimentation and features distribution after outlier removal





6) Demonstration of prediction on a new set of data using RESTful API

```
(base) bishwasapkota@Bishwas-MacBook-Air Housing_price % curl -X POST http://ec2-3-83-68-187.compute-1.amazonaws.com/predictapi -H "Content-Type: application/json" -d '[
{"size": 3000,
"bed": 5,
"bath": 3,
"distance_to_park": 2,
"distance_to_school": 3,
"distance_to_hospital":3}
]'
{"Predicted house price":"[871727.8]"}
```

API: http://ec2-3-83-68-187.compute-1.amazonaws.com/predictapi

Data:

```
'[
{"size": 3000,
"bed": 5,
"bath": 3,
"distance_to_park": 2,
"distance_to_school": 3,
"distance_to_hospital":3}
```

```
      ]'
```

Full command to test:

```
curl -X POST http://ec2-3-83-68-187.compute-1.amazonaws.com/predictapi
-H "Content-Type: application/json" -d '[
{"size": 3000,
"bed": 5,
"bath": 3,
"distance_to_park": 2,
"distance_to_school": 3,
"distance_to_hospital":3}
]'
```