

Predictive Analytics for Real Estate

Nest Analytics

2023S-T4_AIP_GROUP A

Loyalist College in Toronto

FINAL REPORT & PORTFOLIO GROUP A13

GROUP MEMBER DETAILS:

#	Student Name	Student ID	Roles & Responsibilities	Email
1	Husanpreet Kaur	500195671	Researcher, Testing & QA	husanpreetkaur2@loyalistcollege.com
2	Nanda Kishore Karicherla	500197946	Solution and design, Development	nandakaricherla@loyalistcollege.com
3	Tajdar Unnisa Begum	500201392	Project Manager, Implementation/Deployment, Documentation	tajdarbegum@loyalistcollege.com

REAL-ESTATE INDUSTRY

TITLE: A CASE STUDY ON PRICE PREDICTION OF HOUSES USING MACHINE LEARNING

PURPOSE

This project's purpose is to develop a predictive model that can accurately predict a house's price based on various features. The project aims to provide Natty City, a housing company, with a reliable tool that can help them make informed decisions about pricing their properties. As a data engineer the objective is to clean and transform the raw data provided by the client and work closely with the data science team to develop the predictive model.

VALUE

The project holds significant value for the real estate industry as it aims to provide a reliable tool for predicting house prices based on features. This predictive model can be used by housing companies like Natty City to optimize pricing strategies, enhance the customer experience, and gain a competitive edge in the market. Moreover, the project demonstrates the importance of data analytics in the real estate industry, highlighting the increasing demand for skilled professionals who can effectively analyze and interpret complex data.

ABSTRACT

The accurate prediction of house prices is of significant interest in real estate and financial sectors. This project employs the eXtreme Gradient Boosting (XGBoost) algorithm, a powerful machine learning technique, to predict house prices based on a comprehensive set of features. The study aims to demonstrate the effectiveness of XGBoost in modeling complex relationships within housing data and producing reliable price predictions.

The project begins by collecting a diverse dataset that includes various attributes such as land use, acreage, year built, property size, amenities, neighborhood characteristics, and economic indicators. Data preprocessing techniques are applied to handle missing values, outliers, and categorical variables. Feature engineering and selection are performed to extract meaningful features and create a robust input for the XGBoost model.

The XGBoost model is then trained using the prepared dataset and fine-tuned to the hyperparameters to approach a better prediction. Feature importance analysis using a correlation matrix is conducted to gain insights into the features that most significantly influence sale prices. The trained model is rigorously evaluated using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared to assess its predictive performance.

The results demonstrate that the XGBoost model effectively captures complex patterns in housing data and provides accurate house price predictions compared to other techniques. The analysis of feature importance shows interdependent factors, and this dependence was used in feature selection. This study showcases the effectiveness of the XGBoost algorithm in generating reliable and interpretable predictions, providing valuable insights for real estate investors.

INTRODUCTION

A piece of land's natural features, such as any trees, bodies of water, and permanently erected improvements like fences and structures, are all considered to be part of the real estate. People use real estate for a variety of purposes, including retail, offices, manufacturing, housing, ranching, farming, recreation, the church, and entertainment. Numerous interrelated factors, such as geography, population, management experience, managerial competence, governmental legislation, and tax policy, affect whether certain uses are successful or unsuccessful.



Various factors influence the real estate market, including supply and demand, economic conditions, interest rates, government policies, and local market trends. Predicting housing prices with complete certainty is challenging; multiple factors can provide insights into the market's direction. Here are a few critical considerations for predicting housing prices:

1. **Supply and demand:** The housing supply and demand could majorly affect price trends. When demand is more than supply, prices tend to increase, and vice versa.
2. **Economic factors:** GDP growth, employment rates, and household income levels can impact housing prices.
3. **Interest rates:** Mortgage interest rates are vital in the housing market. Lower interest rates stimulate demand and increase prices.
4. **Housing market trends:** Ongoing local market conditions and trends can affect housing prices.
5. **Government policies:** Government rules and policies can influence too. For example, homeownership tax incentives or changes in lending rules can affect demand, impacting prices.
6. **Comparable sales:** Comparing similar properties in the neighbourhood that have recently sold can indicate the price range for similar properties.

With the research outcomes relating to the dataset, we can convey that the following are typically considered when predicting house pricing:

1. **Land Use:** The designated land use of a property, such as a residential condo, single-family, vacant res land, or duplex, can impact its value. Different land uses have varying demand levels and tend to command higher prices in the housing market.
2. **Property Address:** Property location is an essential factor in determining its value. Desirable neighbourhoods, closeness to amenities, schools, transportation, and other factors can significantly influence housing prices.
3. **Sale Price:** The actual sale price of a property is a critical factor in predicting house pricing. Previous sale prices can indicate the value buyers are willing to pay for similar properties in the area.
4. **Acreage:** The property's land size can impact its value. Properties with ample land may have higher prices, especially in areas where land is scarce or in high demand.
5. **Land Value:** The assessed value of the land can provide pricing insights. Location, zoning regulations, development potential, and market demand can influence land value.
6. **Building Value:** The assessed value of the building, size, condition, quality of construction, and architectural features can influence its value.
7. **Finished Area:** The total finished area of the property is an important indicator. Larger, more spacious properties generally command higher prices.
8. **Year Built:** The year house was built can impact its value. Older properties may have distinctive architectural or historical value, while new ones may contain more modern features and amenities.
9. **Bedrooms, Full Bath, Half Bath:** The number of bathrooms and bedrooms is crucial. More bedrooms and baths are typically in higher demand and can fetch higher costs.

Land Use	Property Address	Suite/ RESIDENTIAL CONDO#	Sale Date	Sale Price	Legal Reference	Sold As Vacant	Multiple Parcels Involved in Sale	Owner Name	Address ...	Building Value	Total Value
RESIDENTIAL CONDO	1208 3RD AVE S	8	2013-01-24	132000	20130128-0008725	No	No	NaN	NaN ...	NaN	NaN
SINGLE FAMILY	1802 STEWART PL	NaN	2013-01-11	191500	20130118-0006337	No	No	STINSON, LAURA M.	1802 STEWART PL ...	134400.0	168300.0
SINGLE FAMILY	2761 ROSEDALE PL	NaN	2013-01-18	202000	20130124-0008033	No	No	NUNES, JARED R.	2761 ROSEDALE PL ...	157800.0	191800.0
			Finished Area	Foundation Type	Year Built	Exterior Wall	Grade	Bedrooms	Full Bath	Half Bath	
			NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
			1149.00000	PT BSMT	1941.0	BRICK	C	2.0	1.0	0.0	
			2090.82495	SLAB	2000.0	BRICK/FRAME	C	3.0	2.0	1.0	

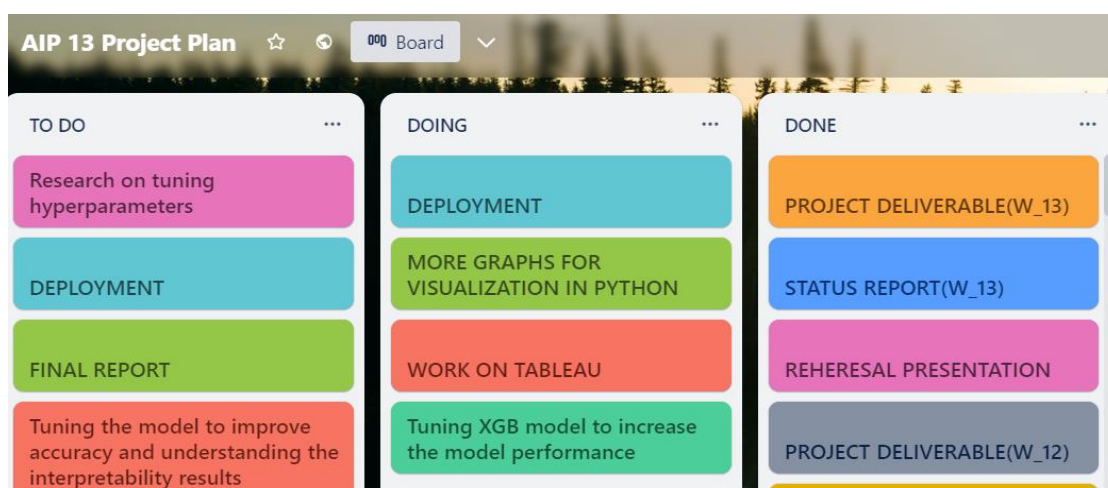
While other features may provide additional context, these features are commonly used to predict house pricing in the current real estate market.

PROJECT MANAGEMENT

Platform: The project makes use of the Python programming language and a few data analysis tools. Jupyter Notebook and Google collab is used to set up the development environment, and GitHub repository is used to maintain version control. We have used Microsoft Teams and outlook for sharing our research outcome, resources, and various files.

Action plan: The team worked together to specify the project's objectives, goals, and deliverables. The processes necessary to clean and transform the raw data to generate the prediction model, and assess its effectiveness were outlined in a plan of action.

Timeline: The project timelines were properly scheduled by dividing the tasks appropriately to each of the team members and then created a Trello board track the project workflow and share individual collaborations. The board was categorized based on various tasks that were involved in the project using different labels, which helped to review and adjust the timeline. [See Appendix]



Hours of Labor: The team utilized AIP class hours, spread across researching on real estate, exploring data and model building.

Cost: The project utilized existing resources on google, streamlit and no additional costs were incurred.

SOLUTION FLOW

The below mentioned steps were followed for solving the problem and come to a better conclusion:

- 1. Requirement Finalization:** The team conducted thorough discussions to understand their specific requirements and expectations for the predictive model of Natty City data set.
- 2. Approach:** The team has decided to follow a supervised machine learning approach using a regression algorithm to predict house prices based on the provided features. The data will be divided into training and testing sets to evaluate the model's performance.
- 3. Data Cleaning and Transformation:** The team has performed data cleaning tasks, handling missing values, and inconsistent data. Feature engineering techniques were employed to extract relevant information from the raw data and select influencing features for the model building.

4. **Model Building:** The team has worked together to develop and fine-tune a predictive model using appropriate regression algorithms. The model was trained on the training data and validated using the testing data.
5. **Evaluation:** The model's performance was evaluated using various metrics such as mean absolute error (MAE), root mean squared error (RMSE), and R-squared score. The team has iteratively refined the model based on the evaluation results.
6. **Interpretability:** Interpreted the model performance to understand and explain how the model makes its predictions or decisions. LIME (Local Interpretable Model-agnostic Explanations) library was employed to accomplish this task.
7. **Deployment:** The team has successfully deployed the model and make it available for use in a production environment, where it can process real-world data and generate predictions.
8. **User interface (UI):** Designed a real-time user interface and integrated the model into a webpage for the customers to interact with.

RESOURCES AND METHODOLOGY

Resource Selection: The team selected Python as the best programming language for data analysis and machine learning activities because of its large library and ecosystem support. The chosen libraries, including pandas and scikit-learn, offered effective tools for model creation and data manipulation.

Resource Quality Considerations: The team will be using established documentation and user reviews to ensure the credibility and quality of the resources it chooses. To maintain the readability and quality of the code, best practices and coding standards will be adhered to.

Methods of Implementation: The project will be using a collaborative and iterative process. To encourage a productive workplace, routine code reviews, pair programming, and knowledge exchange sessions will be held. Git version control will make it possible to integrate code quickly and effectively.

Analytics Techniques: To obtain insights into the data, spot trends, and spot abnormalities, exploratory data analysis (EDA) methods will be used. The prediction model's significant characteristics will be derived using feature extraction and selection techniques.

NATTY CITY DATASET

Price prediction is an essential task in the real estate industry, letting companies make good property pricing decisions. Various techniques and strategies must be employed to predict prices accurately. In this report, we will discuss the techniques required for price prediction in the real estate industry.

I. DATA CLEANING AND TRANSFORMATION

The dataset for house price prediction in Natty city includes 56,636 housing records with 25 features. However, many records have missing data for some features, and only 24,000 records have complete data to study. To address this issue, we explored various methods to impute the missing data and augment the sample size for improved model accuracy.

- 1. Missing Data:** Missing data is defined as the values of data that are blank or not present (N/A) for some variables in the given dataset. Below is a sample of the missing data from the given housing data:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
84	VACANT RE 7204 SMOKEY HILL RD	04/03/13	50000	20130114	No	Yes																					
85	SINGLE FAM 112 FOLEY CT	24/01/13	339900	20130125	No																						
86	SINGLE FAM 408 EASTBORDR	24/01/13	131350	20130125	No																						
87	SINGLE FAM 3202 PARK AVE	14/01/13	285000	20130117	No																						
88	DUPLEX 4306 DAKOTA AVE	11/01/13	450000	20130114	No																						
89	VACANT RE 804 LENA ST	15/01/13	16000	20130118	No																						
90	VACANT RE 729 25TH AVE N	16/01/13	17000	20130118	No																						
91	VACANT RE 732 25TH AVE N	16/01/13	17000	20130118	No																						
92	VACANT RE 2320 BATAVIA ST	15/01/13	27000	20130118	No																						
93	VACANT RE 2318 BATAVIA ST	15/01/13	27000	20130118	No																						
94	DUPLEX 410 37TH AVE N	23/01/13	65500	20130123	No																						
95	VACANT RE 2822 GEORGIA AVE	15/01/13	35000	20130115	No																						
96	SINGLE FAM 333 CHAMBERLAIN ST	24/01/13	130000	20130125	No																						
97	RESIDENTIAL 900 19TH	214	25/01/13	118000	20130129	No																					
98	RESIDENTIAL 900 20TH	802	07/01/13	900000	20130110	No																					
99	RESIDENTIAL 900 20TH	1205	18/01/13	390000	20130131	No																					
100	CONDO 1510 DEM	506	16/01/13	227000	20130118	No																					
101	RESIDENTIAL 6880 CHAM B-3	17/01/13	49500	20130122	No																						
102	RESIDENTIAL 6880 CHAM H-1	11/01/13	40000	20130115	No																						
103	SINGLE FAM 5518 BORN AVE CIR	16/01/13	194000	20130125	No																						
104	SINGLE FAM 4907 IRVING AVE	08/01/13	307500	20130110	No																						

Missing Values

Missing values is a common challenge in real estate data. Below are the techniques that can be implemented to handle missing values:

- Deleting:** If missing values are few and randomly distributed, deleting the corresponding rows or columns can be an option.

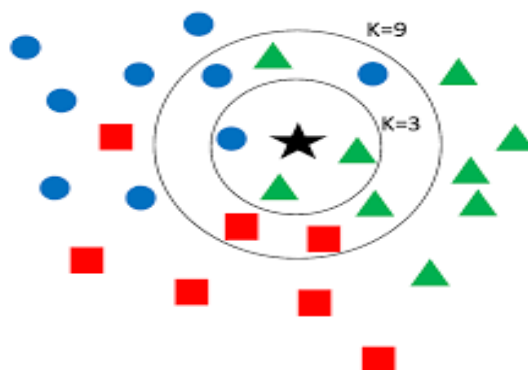
```
df = df.dropna(axis='rows')
df
```

	Land Use	Sale Price	Sold As Vacant	Multiple Parcels Involved in Sale	Acreage	Tax District	Neighborhood	Land Value	Building Value	Total Value	Finished Area	Year Built
1	SINGLE FAMILY	191500	No	No	0.17	URBAN SERVICES DISTRICT	3127.0	32000.0	134400.0	168300.0	1149.00000	1941.0
2	SINGLE FAMILY	202000	No	No	0.11	CITY OF BERRY HILL	9126.0	34000.0	157800.0	191800.0	2090.82495	2000.0
3	SINGLE FAMILY	32000	No	No	0.17	URBAN SERVICES DISTRICT	3130.0	25000.0	243700.0	268700.0	2145.60001	1948.0

- b. Imputation:** Techniques such as mean, median, mode imputation, or more advanced methods like regression imputation or multiple imputation can be used to estimate missing values.

```
df[categorical_features] = df[categorical_features].fillna(df[categorical_features].mode().iloc[0])
```

- c. Advanced techniques:** Use of more sophisticated methods like K-nearest neighbors (KNN) imputation or matrix factorization. For the given dataset we implemented KNNImputer to fill in the null values.

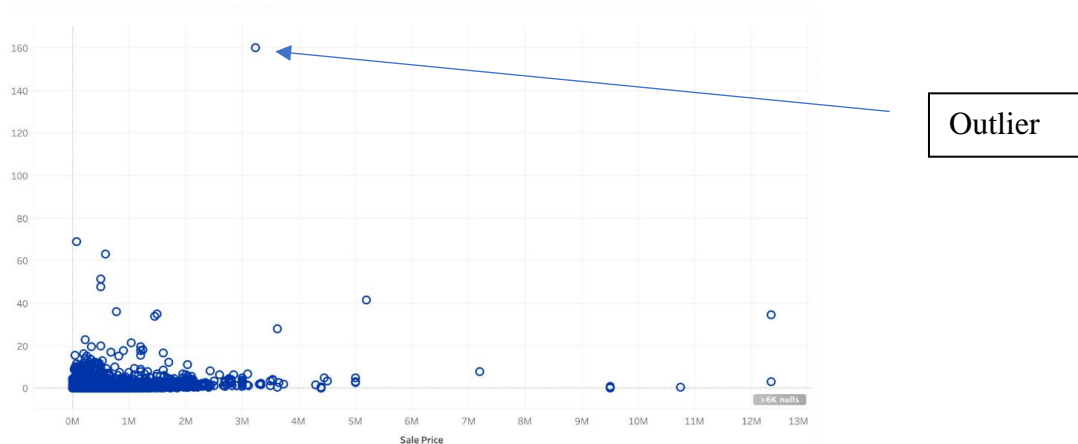


KNNImputer technique by scikit-learn is a widely used method to impute missing values into a dataset. It is widely being used as a replacement for conventional imputation techniques.

```
imputer = KNNImputer(n_neighbors=3)
imputed = imputer.fit_transform(df[numerical_features])
imputed
```

After imputing the null values using KNN Imputer, the total number of records were increased to 56636 samples with 19 features (8 categorical and 11 numerical).

- 2. Outliers:** Some values in the dataset are much greater than others. These factors introduce bias into the statistical results, which are often called outliers. It is better to get rid of these outliers before building the model.

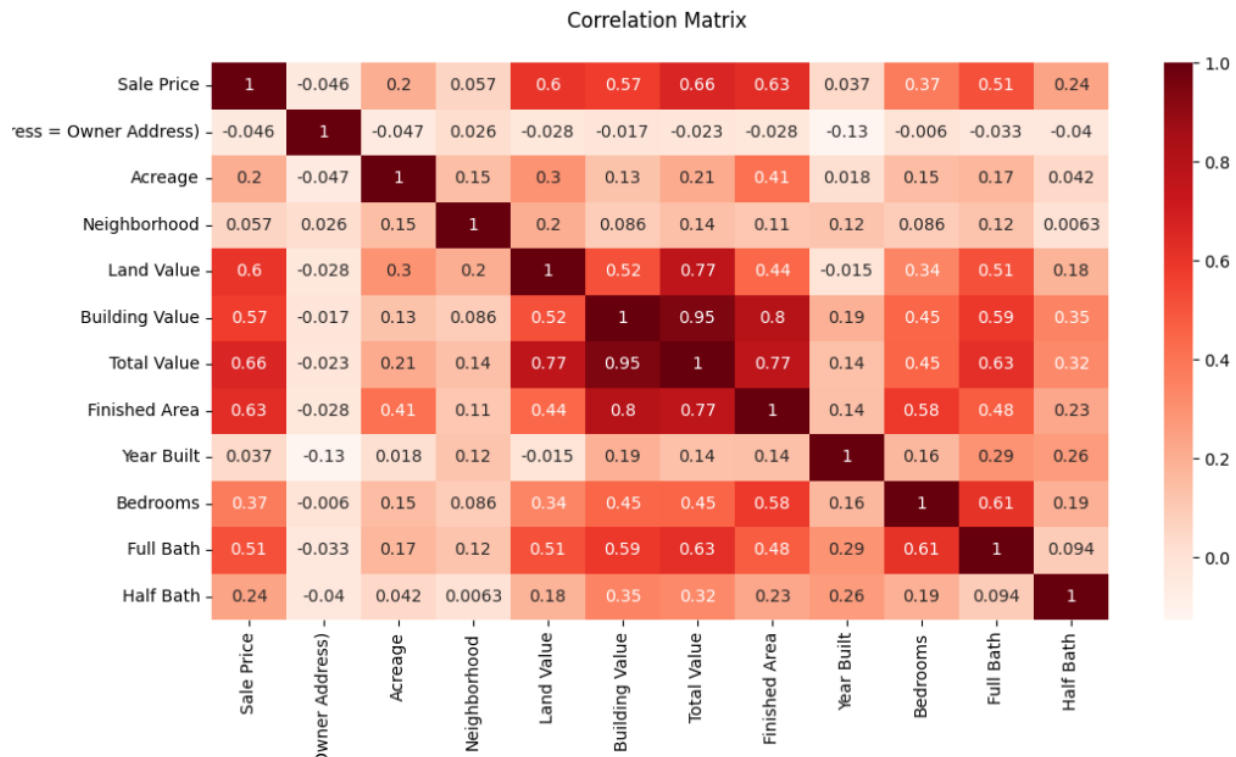


3. **Categorical Data:** Categorical data is non-numerical information that is divided into groups. As its name suggests, categorical data describes categories or groups. To address the categorical features, techniques such as imputation with mode, label encoding, one-hot encoding, etc., can be implemented.
4. **Date, Address, and Name Data:** Date, address, and name data require special treatment while building the model for price prediction.
 - Date data - Extracting features like a day of the week, month, or year and incorporating time-related patterns.
 - Address data - Extracting location features like zip code or neighborhood and incorporating spatial patterns.
 - Name data - Extracting relevant information from names, such as property owner demographics or property characteristics.

II. DATA PREPROCESSING

1. Correlation Matrix:

Most of these features are categorical which has less effect on the sale price of the house. So, the first step is to identify the features that most influence the sale price. For this purpose, we use the Correlation Heatmap to check the correlation between features of the dataset. It helps to find out which of the features are dependent on each other and their dependence can be used to select appropriate features.



From the above correlation matrix, it is evident that ‘Total Value’ of the house has much influence on the sale price compared to other features and there is a negative influence with respect to ‘Property Address = Owner Address’. A negative correlation between variables indicates that as one increases, the other tends to decrease. In the context of the correlation between ‘Sale Price’ and the relation ‘Property Address = Owner Address’, a negative correlation implies an intriguing but possibly counterintuitive relationship.

When the property address and owner address are the same, it may indicate that the owner lives in the property. This could result in a lower sale price due to sentimental or personal value attached to the property. Conversely, when Property Address differs from Owner Address, it may indicate the property is an investment or rental unit. The owner may be more motivated to sell at a higher price resulting in a higher sale price.

It is also important to note that the correlation between variables does not have a direct impact on the sale price. The negative correlation in the correlation matrix suggests a statistical relationship between the variables, but it may not have a direct cause in the difference in sale prices. Other factors may affect the price, and further analysis would be needed to establish a strong relationship or better understand the influence on sale price.

2. One-hot Encoding:

One-hot encoding is a technique used in machine learning to convert categorical variables into a binary matrix representation, which is ideal for algorithms that require numerical inputs. This technique is often used when dealing with categorical features that have no ordinal relationship between categories. Categorical features are encoded using the OneHotEncoder library from scikit-learn. It encodes the categorical features and concatenates them with the numerical features to form a dataframe. The parameters have been fine-tuned to get a maximum number of features to be considered for modeling.

```
OH_encoder = OneHotEncoder(sparse=False, handle_unknown='ignore', min_frequency=200)
OH_cols = OH_encoder.fit_transform(df[categorical_features])
feature_names = OH_encoder.get_feature_names_out(input_features=categorical_features)
OH_df = pd.DataFrame(OH_cols, columns=feature_names, index=df.index)
df3 = df.drop(categorical_features, axis=1)
df4 = pd.concat([df3, OH_df], axis=1)
```

After fixing the one-hot encoded features by dropping unnecessary columns, the output was increased to 41 features, and these can be adjusted by changing the parameters based on model requirements.

```
df4.columns
Index(['Sale Price', 'Acreage', 'Neighborhood', 'Land Value', 'Building Value',
      'Total Value', 'Finished Area', 'Year Built', 'Bedrooms', 'Full Bath',
      'Half Bath', 'Land Use DUPLEX', 'Land Use RESIDENTIAL CONDO',
      'Land Use SINGLE FAMILY', 'Land Use VACANT RESIDENTIAL LAND',
      'Land Use_ZERO LOT LINE', 'Land Use_infrequent_sklearn',
      'Sold As Vacant_No', 'Sold As Vacant_Yes',
      'Multiple Parcels Involved in Sale_No',
      'Multiple Parcels Involved in Sale_Yes',
      'Is (Property Address = Owner Address)_False',
      'Is (Property Address = Owner Address)_True',
      'Tax District_CITY OF BELLE MEADE', 'Tax District_CITY OF FOREST HILLS',
      'Tax District_CITY OF GOODLETTSVILLE', 'Tax District_CITY OF OAK HILL',
      'Tax District_GENERAL SERVICES DISTRICT',
      'Tax District_URBAN SERVICES DISTRICT',
      'Tax District_infrequent_sklearn', 'Foundation Type_CRAWL',
      'Foundation Type_FULL BSMT', 'Foundation Type_PT BSMT',
      'Foundation Type_SLAB', 'Foundation Type_infrequent_sklearn',
      'Exterior Wall_BRICK', 'Exterior Wall_BRICK/FRAME',
      'Exterior Wall_FRAME', 'Exterior Wall_STONE',
      'Exterior Wall_infrequent_sklearn', 'Grade_A', 'Grade_B',
      'Grade_C', 'Grade_D', 'Grade_X', 'Grade_infrequent_sklearn'],
      dtype='object')
```

III. MODEL BUILDING

In machine learning, the "train-test split" is a fundamental technique used to evaluate the performance of a model on unseen data. It involves splitting your dataset into two subsets: one for training the model and another for testing its performance. The goal of this splitting is to assess how well the model generalizes to new data.

After generating and transforming the features, the dataset should be split into train-test subsets with a ratio of 80/20, which means 80% of the data will be used for training and 20% will be used for testing the model performance.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Modeling is the part where the data subsets will be used, and the quality of data will be tested. Our problem involves using regressor models from scikit-learn for prediction. Training data is used for training the model, and testing data could be used for testing the model. We will train multiple models to find the optimal one, check their train-test accuracy, and select the best-performing model. The following models are used:

1. Linear Regression
2. Polynomial Regression
3. Ridge Regression
4. Lasso Regression
5. Gradient Boosting Regressor
6. XGBoost Regressor

Out of the above regression models, it is observed that XGBoost (Extreme Gradient Boosting) technique effectively captures complex patterns in housing data and provides accurate house price predictions compared to other techniques. The results of the model building are shown below:

```
xgb = XGBRegressor(n_estimators=2500, learning_rate=0.1, n_jobs=5, reg_alpha=5)
xgb.fit(X_train_scaled, y_train)
```

XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=2500, n_jobs=5, num_parallel_tree=None,
              predictor=None, random_state=None, ...)
```

IV. MODEL EVALUATION

Model evaluation is a crucial step in machine learning to assess the performance and generalization capabilities of the trained model. It helps understand how well the model will likely perform on new, unseen data.

There are multiple parameters for evaluating a model based on the label of the dataset. Categorical and regression labels have different evaluation methods. Following methods were used for evaluation with regression:

1. **R-squared (R^2 Score):** Also known as Coefficient of Determination, it measures the amount of predictable variance in the target variable based on the input features.
2. **Root Mean Square Error (RMSE):** The mean squared error (MSE) is the average of the squared differences between predicted and actual values. The root mean squared error (RMSE) is the square root of MSE and provides a more interpretable scale.
3. **Mean Absolute Error (MAE):** It is the average of absolute differences between predicted and actual values.

The results for the XGBoost model were as follows:

```
print("XGB R^2 Score: ", xgb.score(X_train_scaled, y_train))
print("XGB Test R^2 Score: ", xgb.score(X_test_scaled, y_test))
```

XGB R^2 Score: 0.7663655635822292
XGB Test R^2 Score: 0.7815666996066185

```
y_pred_xgb = xgb.predict(X_test_scaled)
```

```
mae_xgb = mean_absolute_error(y_test, y_pred_xgb)
print('XGB Regression MAE: ', mae_xgb)

rmse_xgb = np.sqrt(mean_squared_error(y_test, y_pred_xgb))
print('XGB Regression RMSE: ', rmse_xgb)
```

XGB Regression MAE: 44436.364605728515
XGB Regression RMSE: 390869.9182212071

XG Boosting is the highest performing model amongst all the models used with an accuracy of 76% on the training dataset and 78% on the testing dataset. Also, the metrics to evaluate the performance can be utilized to tune the hyperparameters of the model and get better results.

Tuning:

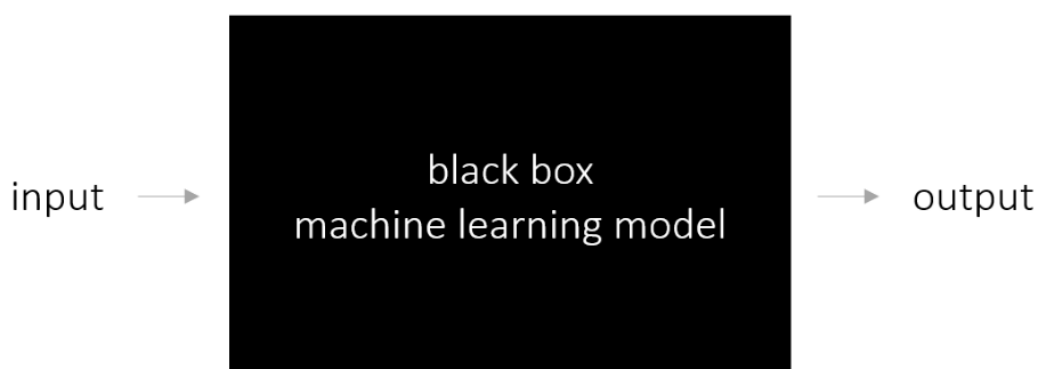
After increasing the number of samples and features, the model accuracy was increased by a little and the underfitting issue was resolved. Below are the results of tuning for XG boost model.

```
print("XGB R^2 Score: ", xgb.score(X_train_scaled, y_train))
print("XGB Test R^2 Score: ", xgb.score(X_test_scaled, y_test))
```

XGB R^2 Score: 0.8178764203186268
XGB Test R^2 Score: 0.8178101238001536

V. INTERPRETABILITY WITH LIME

In the context of machine learning, interpretability refers to the ability to understand and explain how a model makes predictions or decisions. It's about analyzing why a model produces a certain output based on a set of input features. When working with complicated models, such as deep neural networks or ensemble approaches, which are often referred to as "black boxes" because of their complex internal workings, interpretability becomes especially crucial.



Why Lime?

Black-box versions are no longer fashionable. Creating beautiful models these days is simple, but what's going on inside? That is what Explainable AI and LIME are attempting to discover.

Understanding why the model generates the predictions it does is critical for fine-tuning. Consider this: How can you enhance anything if you don't know what's going on inside?

Lime Interpretability

LIME is an abbreviation for Local Interpretable Model-agnostic Explanations. The project aims to illustrate what machine learning models do (source). LIME presently offers explanations for tabular models, text classifiers, and picture classifiers.

The primary goal of LIME is to provide insight into why a specific instance was classified or forecasted in a particular way by the model. This aids in comprehending the model's behavior and finding the features that significantly impact its predictions. LIME works by locally approximating the decision boundary of the black-box model around the instance of interest with a simpler, interpretable model, such as linear regression or decision trees.

The following steps are involved while using LIME:

1. **Choose the Instance:** Select the precise data point or instance you wish the model prediction to be interpreted.
2. **Create Perturbations:** LIME creates perturbations to the selected instance by randomly picking data points nearby.
3. **Get Perturbed Instance Model Predictions:** Run these perturbed cases through the black-box model and gather the model's predictions.

4. **Fit an Interpretable Model:** LIME fits a simplified interpretable model to the altered data, with the feature values as input and the model's prediction as output.
5. **Analyze the Interpretable Model's Coefficients or Rules:** Analyse the interpretable model's coefficients or rules to learn how different features contribute to the prediction. Positive and negative coefficients represent the impact of the attribute on increasing or decreasing the prediction, respectively.

Python's Lime library implements 'LIME' and provides a variety of classes and methods for performing these tasks. You can use 'lime.lime_tabular.LimeTabularExplainer' for tabular data and 'lime.lime_text.LimeTextExplainer' for text data, among other things.

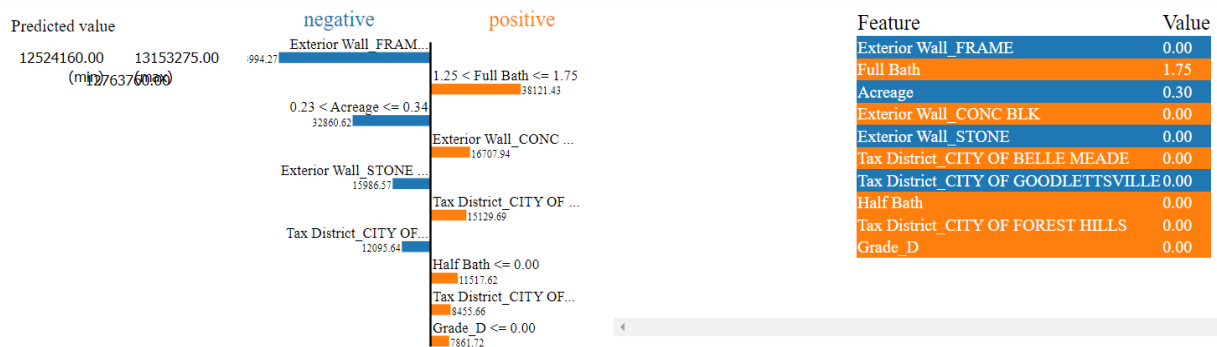
```
import lime
import lime.lime_tabular
```

```
explainer = lime.lime_tabular.LimeTabularExplainer(X_train.to_numpy(), feature_names=X_train.columns, class_names=['Sale price'], mode='regression')
```

LIME is a robust model interpretability and transparency tool that helps data scientists, machine learning practitioners, and researchers get deeper insights into complicated models' decision-making processes.

Using the Lime library to interpret how the features are affecting the model prediction, we considered 5 instances of the dataset to explain the model, with a sample of 10 features and 10000 records.

```
instance = X_test.values[0]
exp = explainer.explain_instance(instance, xgb.predict, num_features=10, num_samples=10000)
exp.show_in_notebook(show_table=True)
```



```
exp.as_list()
```

```
[('0.00 < Land Use_RESIDENTIAL CONDO <= 1.00', 2166501.2356927674),
 ('Bedrooms > 3.33', 114812.87673028081),
 ('0.22 < Acreage <= 0.31', -65971.29963120173),
 ('0.00 < Half Bath <= 0.67', 44664.43574838465),
 ('Exterior Wall_FRAME <= 0.00', -31244.921679834755)]
```

Using the above explanation for various instances, the company can iterate the values of the influencing features based on customer requirements and market demands, to get a higher sale price.

VI. USER INTERFACE

User friendly web application is designed to enhance company's experience in property evaluation and showcase the results to any user. With our intuitive user interface, company can effortlessly input key house features such as the number of rooms, area in square feet, full bathrooms, and half bathrooms. Our advanced machine learning model, powered by Flask, delivers predictions, helping to make informed property valuation decisions.

But it didn't stop there, we consider the importance of understanding the "why" behind predictions, that's why we have integrated interpretability into our application. Our platform goes beyond the prediction itself to provide feature importance interpretations. Now, you can gain valuable insights into which specific features influence the predicted house price the most. This transparency empowers a company to analyze and refine its strategies for property assessment and pricing.

House Price Prediction

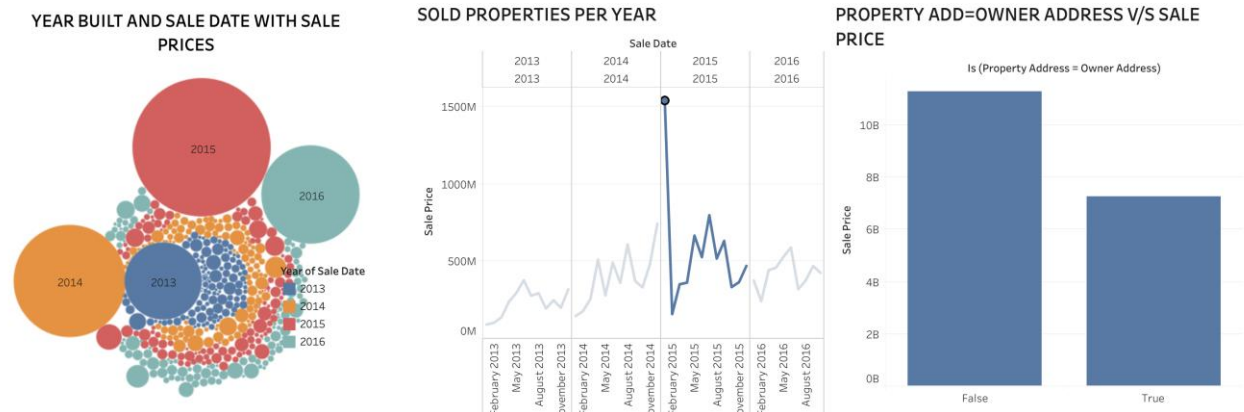
Number of Rooms:

Area (in square feet):

Number of Full Bathrooms:

Number of Half Bathrooms:

Tableau Dashboard



Additionally, our HTML-based interface ensures a visually appealing and cohesive experience, making the process of interacting with the application intuitive and efficient. We've combined the power of Flask for backend processing, interpretability, and HTML for user interface design, all with the goal of delivering a holistic, user-centric experience. UI incorporates a Tableau Dashboard, which offers data visualization and insights related to the house price prediction.

Comprehensive summary of both the Python code and the provided HTML template, covering all the key components and functionalities

VII. PYTHON CODE

1. Importing Libraries:

The code imports the required libraries, including Flask for building web applications and joblib for model serialization.

2. Setting up Flask:

A unique template folder is created for the Flask application to render the user interface.

3. Loading the Model:

`joblib.dump(xgb, 'xgb.pkl')` is used to download trained models and then we load the model using `joblib`, a pre-trained machine learning model (XGBoost) is loaded from the file 'xgb.pkl'.

4. Data Preprocessing:

The `preprocess_input` function is defined to handle user input (rooms, area, full bath, half bath) for prediction.

5. Get_feature_interpretations:

This function pulls feature importance or justifications from the loaded model.

6. Routes:

The following routes are listed,

An input form allowing visitors to submit property features is displayed at the root URL ('/').

The '/predict' route manages form submission, input data preprocessing, model predictions, and feature interpretation retrieval.

Requests for the favicon are handled by the route '/favicon.ico'.

7. Error handling:

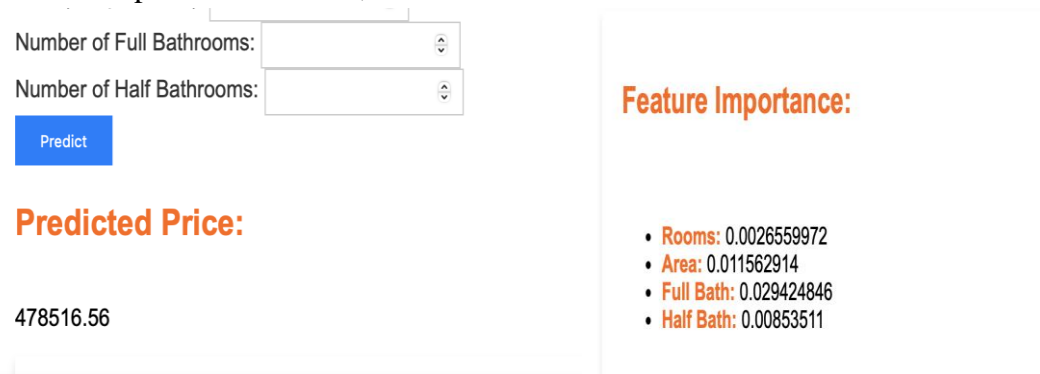
When a form is submitted, the application handles exceptions and sends back JSON answers with error details.

8. Launching the Program:

The script launches the Flask application, specifies the host and port, and enables debugging if it is launched as the primary program.

9. HTML Template:

The layout consists of a container with three main sections: the user input form, the feature importance interpretations section, and the Tableau Dashboard.



The screenshot shows a web application interface for house price prediction. On the left, there is a form with two input fields: 'Number of Full Bathrooms:' and 'Number of Half Bathrooms:'. Below these fields is a blue 'Predict' button. To the right of the form, the 'Predicted Price:' is displayed as '478516.56'. On the far right, under the heading 'Feature Importance:', there is a list of features and their importance scores: Rooms (0.0026559972), Area (0.011562914), Full Bath (0.029424846), and Half Bath (0.00853511).

Result of house price prediction and interpretability

In summary, our innovative approach combines user-friendliness, predictive accuracy, and feature interpretability, all wrapped in a HTML interface powered by Flask. We're excited to present a solution that not only predicts house prices but also provides the insights needed for a company to analyze, improve, and excel in the property market.

VIII. REFERENCES

- <https://www.kaggle.com/code/yasserh/housing-price-prediction-best-ml-algorithms>
- <https://www.investopedia.com/articles/mortgages-real-estate/11/factors-affecting-real-estate-market.asp>
- <https://wowa.ca/reports/canada-housing-market>
- <https://www.graana.com/blog/factors-affecting-real-estate-market/>
- <https://www.kaggle.com/code/chanakyavivekkapoor/house-price-prediction>
- <https://www.geeksforgeeks.org/house-price-prediction-using-machine-learning-in-python/>
- <https://stackoverflow.com/questions/58756515/onehotencoder-object-has-no-attribute-get-feature-names>
- <https://towardsdatascience.com/support-vector-regression-svr-one-of-the-most-flexible-yet-robust-prediction-algorithms-4d25fbdaca60>
- <https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e>
- <https://www.kaggle.com/code/gaurav896/nashville-housing-initial-analysis>
- <https://www.kaggle.com/code/freeknowledge/house-price-prediction-pipeline-one-hot-encoding>
- <https://medium.com/@vivekpadia70/understanding-how-airbnb-uses-machine-learning-to-solve-house-price-prediction-problem-afd5c6c9ec32>
- <https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/>
- <https://www.kaggle.com/code/phomolo/house-price-prediction-linear-lasso-ridge>
- <https://www.kaggle.com/code/prashant111/explain-your-model-predictions-with-lime>
- <https://sarit-r.medium.com/typeerror-slice-none-none-none-0-is-an-invalid-key-1b4d1eecdc46>
- <https://betterdatascience.com/impute-missing-data-with-python-and-knn/>
- <https://machinelearningmastery.com/knn-imputation-for-missing-values-in-machine-learning/>
- <https://www.w3schools.com/html/>
- <https://pythonbasics.org/what-is-flask-python/>

IX. APPENDIX

Document Name	Attachment/ Location saved
Project tracker	Trello Board - AIP A13 Project Progress
Code for Model Building	Project Group A13.ipynb - Google Collab
GitHub Repository	GitHub - nanda1296/Predictive-Analytics-for-Real-Estate
Tableau Dashboard	Tableau - Dashboard
User Interface	Demo video of UI
Code for UI	User interface code files
Web Page	WIX Real Estate Prediction