Real Estate Prediction in NYC

1st Ahmad Shah
School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, United States
sshah6@stevens.

2nd Alexandra Anthony
School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, United States
aanthony@stevens.edu

3rd Moyosola Omole
School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, United States
momole@stevens.edu

***Abstract*—Abstract:**
This study presents the development of a machine learning model aimed at predicting future property values utilizing datasets from NYC Open Data. Initial focus was placed on data pre-processing, where missing values were addressed through imputation techniques, and feature scaling was standardized to ensure the dataset was uniform. Cross-validation methods allowed for model performance assessments and an evaluation of selected classifiers. Experimentation with cross-validation techniques helped us to discover the most generalized model for our dataset. Using GridSearchCV, parameter combinations were explored to enhance model generalizability and accuracy. Evaluation metrics, including ROC curves for classification tasks and error metrics for regression models, were consistently reported throughout the project. Visual representation of repeated stratified k-fold cross-validation further developed model performance across multiple iterations.

I. Introduction

This paper presents an approach to developing a machine learning model aimed at predicting future property value using data sourced from the New York City (NYC) Open Data using practical implementations and analytical evaluations. The steps are covered beginning with data pre-processing, model comparison with various classifiers, cross-validation techniques, and model optimization. Each stage was designed to ensure the accuracy of the predictive model. The key steps involved in the process were data imputation, scaling, model selection, cross-validation, and optimization.

II. Data Pre-processing

A. Data Selection

Initially, we selected Property Valuation and Assessment Data [1] to base our machine learning algorithm on. However, after looking over the data, we quickly realized the dataset had not been updated since 2022. In order to obtain an accurate model, we added a second data set [2] with records up to and including 2024.The datasets had similar presentation and categorical variables as they were from the same source. After loading in both datasets, we began categorical analysis to choose and combine the values we want from data frame (df)1 (2010-2019) and df2 (2021-2025). We first modified the variable names making them all lowercase for consistency. All data from 2024-2025 was dropped in order to predict those values.

Then we ensured that the two pandas dataframes, ‘df2‘ and ‘df3‘, had compatible column names for integration.

```
# This is to combine the values we want from df1 (2010-2019) and df2 (2021-2025

# This is because d1 has the years formatted like this 2010/11 when we want it
df1['year'] = df1['year'].str.slice(0, 4)

# Drop rows where the year is 2024 or 2025
df2 = df2[~df2['year'].isin([2024, 2025])]


columns_mapping = {
    'boro': 'boro',              # d2:boro -> d3:boro
    'block': 'block',            # d2:block -> d3:block
    'lot': 'lot',                # d2:lot -> d3:lot
    'easement': 'easement',      # d2:easement -> d3:easement
    'year': 'year',              # d2:year -> d3:year
    'fintaxclass': 'taxclass',   # d2:FINTAXCLASS -> d3:taxclass
    'bldg_class': 'bldgcl',      # d2:bldg_class -> d3:bldgcl
    'owner': 'owner',            # d2:owner -> d3:owner
    'street_name': 'staddr',     # d2:street_name -> d3:staddr
    'lot_frt': 'ltfront',        # d2:LOT_FRT -> d3:LTFRONT
    'lot_dep': 'ltdepth',          # d2:LOT_DEP -> d3:ltdep
    'bld_story': 'stories',      # d2:bld_story -> d3:stories
    'pymktland': 'fullval',      # d2:PYMKTLAND -> d3:FULLVAL
    'pyactland': 'avland',       # d2:PYACTLAND -> d3:AVLAND
    'pyacttot': 'avtot'          # d2:PYACTTOT -> d3:AVTOT
}
```

Fig. 1. Code Utilized for Column Mapping.

We used a mapping dictionary, ‘columns_mapping‘1, which paired column names from ‘df2‘ with their corresponding names in ‘df3‘. It filtered and renamed the columns in ‘df2‘ based on the mappings, resulting in a modified dataframe named ‘df2_renamed‘. It combined the aligned data from ‘df2_renamed‘ with ‘df3‘ or updates ‘df3‘ with the changes, depending on the operation chosen. This process ensured that both datasets were aligned and ready for further analysis or processing.

df3

	boro	block	lot	easement	year	taxclass	bldgcl	owner	staddr	ltfront	ltdepth	stories	fullval	avland	avtot
0	3	2541	30	NaN	2021	NaN	C1	299 LLC	MCGUINNESS BOULEVARD	62.0	35.0	5.0	870000	39150	774450
1	3	2541	33	NaN	2021	NaN	C1	293 LLC	MCGUINNESS BOULEVARD	76.0	35.0	5.0	1060000	47700	1153080
2	3	2541	36	NaN	2021	NaN	C1	287A LLC	MCGUINNESS BOULEVARD	62.0	35.0	5.0	870000	39150	872640
3	3	2541	38	NaN	2021	NaN	B9	JUSZCZAK, ANTONINA	JAVA STREET	25.0	100.0	2.0	314000	18840	95580
4	3	2541	39	NaN	2021	NaN	B9	SAWICKI CZESLAW	JAVA STREET	25.0	100.0	3.0	298000	17880	116520
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9895550	5	7038	1059	NaN	2011	1A	R3	MARGUERITE FIORE/FAMI	131 GERVIL STREET	4.0	42.0	1.0	254970	2461	11322
9895551	5	7860	1001	NaN	2011	1A	R3	KORIK, ANTHONY	340 SPRAGUE AVENUE	22.0	27.0	1.0	212275	1866	10725
9895552	5	7858	36	NaN	2011	1	A5	MALDARELLI, LEONARD	709 ROCKAWAY STREET	25.0	100.0	2.0	369000	10318	18755
9895553	5	7091	19	NaN	2011	1	B9	JOHN J PERROTTA	1584 WOODROW ROAD	24.0	113.0	2.0	431000	15420	25860
9895554	5	7710	526	NaN	2011	1	A2	STEPHEN W REBRACCA	430 RICHARD AVENUE	60.0	100.0	1.5	405000	13480	24300

9895555 rows × 15 columns

Fig. 2. Head of df3 After df1 and df2 Concatenation.

Before continuing with data imputation, we decided to split the data by borough. Since each borough is likely to exhibit different growth characteristics, we decided to separate the dataset based on the ’boro’ column. By doing this, we discovered through the graph in Figure 3 that there was significantly more data for Brooklyn and Queens. Visualizing the comparisons between the number of data points before and after cleaning helps assess the impact of preprocessing operations. It provides a clear snapshot of data loss or

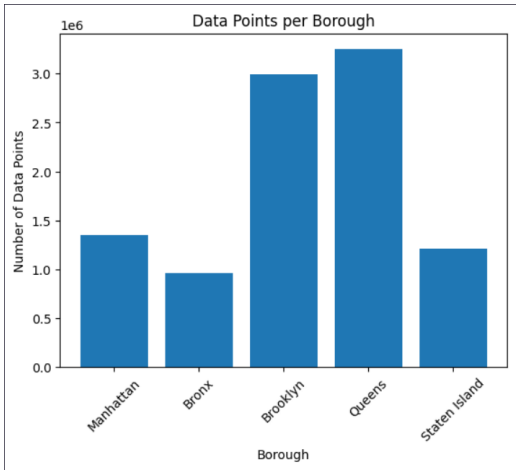

Fig. 3. Bar Graph of Number of Datapoints By Borough.

modification, aiding in understanding dataset reliability. These visuals enhance data quality assessment and development.

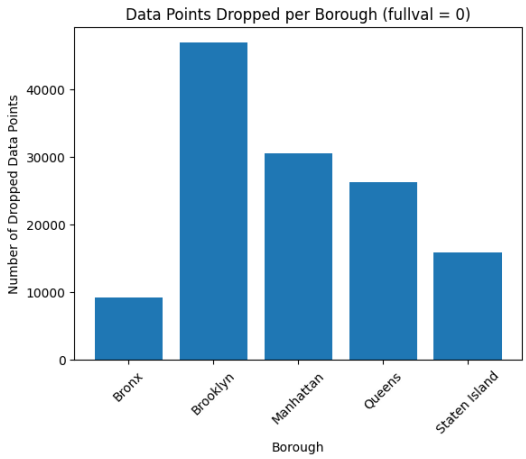

Fig. 4. Bar Graph of Number of Data points Removed By Borough.

As we interpret the graphs above, we can take some educated guesses about the real estate dynamics across the NYC boroughs. Since a "zero" valuation is given when a property does not meet the lending criteria or if further information is needed to assess the value [3]. Characteristics such as fire hazards, social issues due to location, or invasive plants can all trigger a "zero" valuation. So in the context of the boroughs there's a lot of information in the null data visualized in Figure 4. Areas with lower incomes might have more 'zero' valuations if they are associated with higher crime rates or poor infrastructure, which could affect the ability to sell the property. However, this does not appear to be directly supported by the data, as the borough with the fewest dropped data points is the Bronx, which is traditionally considered one of the less affluent boroughs. Still, boroughs like Brooklyn and Queens are significantly larger in size. More processing and analysis regarding the ratios between total data points and dropped data of each borough is needed to make more accurate assumptions. This analysis hints at a complex relationship between property valuations, lending practices, and the socioeconomic profiles of different boroughs. While we might guess that affluent areas could have more active real estate markets and stricter lending standards, the data also suggest that other factors such as building age, infrastructure, and risk could impact the real estate market value. Affluent areas may have less data due to the cost of living as well. The market may be more active in less affluent areas during socioeconomic distress. Further analysis will likely produce more clear insight.

B. Data Imputation

Data imputation is the process of replacing or excluding missing or null values in a dataset. In our algorithm, the SimpleImputer class from sklearn is used within a pipeline structure. We use the dropna() method to eliminate rows with missing values in specific columns. For instance, df combined.dropna(subset=[target], inplace=True) removes rows where the target variable is missing. We removed values that had missing values so as to not alter any trends. For example, if a lot is missing a full valuation, we would rather remove that data, as giving it a mean value for nearby lots would be nonsensical as the stories of the building may be different or it may have a vastly different front or back lot-tage. We also removed the outliers from the code as well using interquartile range (IQR) in the remove outliers() function to filter them out. We did this to remove abnormally high or low values such as the skyscrapers and penthouses. However not all were removed as we used IQR to remove the most abnormal, as we continue we will need to address separating the other skyscrapers

C. Scaling

We combined the steps into pipelines using sklearn's Pipeline and ColumnTransformer for handling both numerical and categorical data. For numerical features the pipeline includes an imputer (using the mean strategy) and a standard scalar, however we don't want to scale as we are trying to predict market value. We also had to process the text data into a form the computer could understand using sklearn preprocessing module OneHotEncoder. For categorical features an imputer (filling missing values with a placeholder) is used.

```
# Data Cleaning
df_combined.dropna(subset=[target], inplace=True)

# Convert categorical features to string
# categorical_features = ['boro', 'taxclass', 'bldgcl', 'staddr']  # Assuming 'staddr' is categorical
categorical_features = ['taxclass', 'bldgcl']
for col in categorical_features:
    df_combined[col] = df_combined[col].astype(str)

# Categorize features
numerical_features = [f for f in features_to_use if f not in categorical_features]

# Preprocessor for handling both categorical and numerical features
preprocessor = ColumnTransformer(
    transformers=[
        ('num', Pipeline([
            ('imputer', SimpleImputer(strategy='mean')),  # or 'median'
            ('scaler', StandardScaler())
        ]), numerical_features),
        ('cat', Pipeline([
            ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
            ('encoder', OneHotEncoder(handle_unknown='ignore'))
        ]), categorical_features)
    ]
)
```

Fig. 5. Column Transformer Set Up.

III. Model Comparisons

A. Training and Evaluation

After preprocessing the data, different machine learning models or classifiers were trained and compared to determine which one performs best for the given task. Upon evaluating various regression models including Linear Regression, Random Forest Regressor, and XGBRegressor, we aimed to select the model that best predicts property values in NYC. The provided code snippet executed these models and assessed their performance metrics such as Mean Squared Error (MSE) and R-squared. The MSE quantifies the average squared difference between predicted and actual property values, providing an accuracy measurement. R-squared, on the other hand, represents the proportion of the variance in the dependent variable that is predictable from the independent variables. After thorough evaluation, the XGBRegressor model was the best choice, with a lower MSE compared to other models. This indicates that the XGBRegressor model more accurately predicts property values. Additionally, the high R-squared value further illustrates the model's ability to explain variance in property values. We created a scatter plot comparing the actual target values against the predicted values. This allowed us to visually assess how closely the predictions align with the actual data points. We can also include a line of best fit to show the general trend.

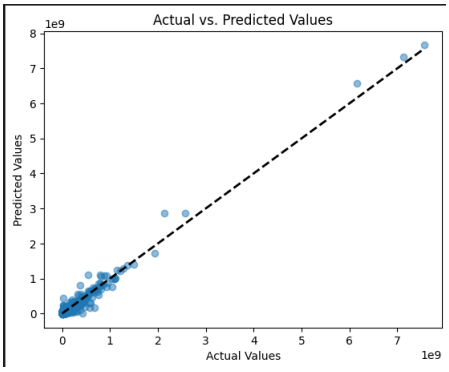

Fig. 6. Scatter Plot of Actual vs Predicted Property Values

The scatter plot in Figure 6 shows that the predicted property values align with the actual values, since the points cluster around the diagonal line. Still, there are a few outliers at the top right of the graph, indicating where the model's predictions deviated from the actual values. These outliers may represent properties with unique characteristics that are not captured by the model. The scatter plot provides a visual of the model's performance and its ability to accurately predict property values in NYC.

B. Choosing A Model

We chose to test three different regression models; RandomForestRegressor, LinearRegression, and XGBRegressor due to their modeling approaches. RandomForestRegressor can handle both classification and regression, and it can capture non-linear relationships in the data. LinearRegression, was chosen as a baseline as it assumes a linear relationship between input features and the target variable. Finally, XGBRegressor, part of gradient boosting models, was included for its ability to handle complex non-linear relationships. Following the evaluation described and visualised previously; XGBRegressor was the best choice, demonstrating the highest accuracy among the models we tested.

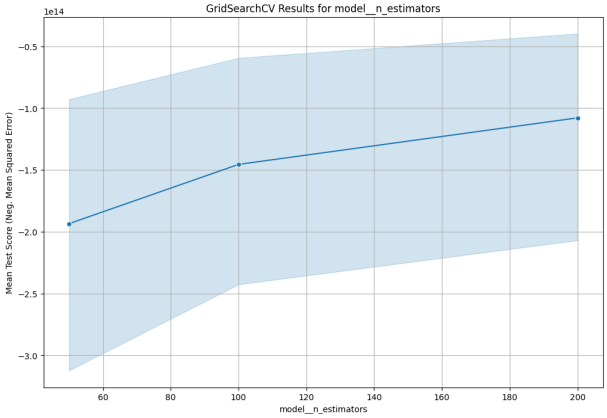

Fig. 7. Gridsearch results for model_n_estimators

As the number of estimators increases (from 50 to 200) as seen in Figure 7, there is a decreasing MSE, indicating improved model performance with more estimators. This improvement in accuracy suggests that increasing the complexity of the model by adding more estimators leads to better predictive power, as the model can capture more intricate patterns in the data. Still, over-fitting needs to be considered as it occurs when the model becomes too complex and starts capturing noise in the training data, leading to poor generalization. The graph in Figure 8 reveals a discernible trend

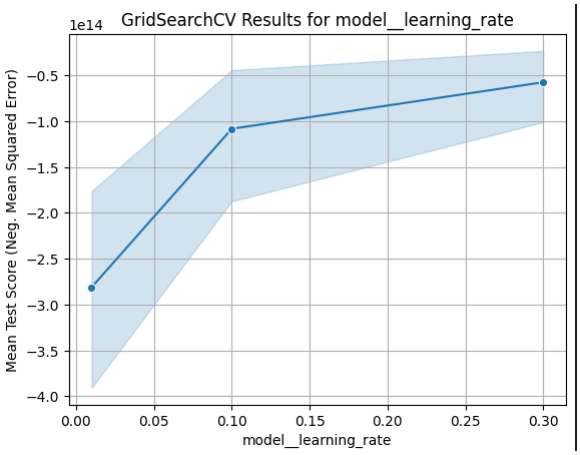

Fig. 8.

where lower MSE values are associated with specific learning rates, indicating improved model performance. This suggests that certain learning rates are more effective in minimizing prediction errors and enhancing the model's predictive power.

The choice of learning rate is crucial in gradient boosting algorithms like XGBoost, as it determines the step size during the optimization process. A smaller learning rate allows for finer adjustments to the model parameters, potentially leading to better convergence and lower MSE. Conversely, a larger learning rate may result in faster convergence, but it could also lead to overshooting optimal parameter values and higher MSE. The graph in Figure 8 reveals a trend where lower MSE values are associated with specific learning rates, indicating improved model performance.

IV. Cross-Validation

A. K-Fold Cross-Validation Execution

For K-Fold Cross-Validation, we used the RepeatedKFold function from the scikit-learn library. The dataset was split into 5 folds, with each fold repeated 3 times, using n_splits=5 and n_repeats=3. We created a Linear Regression model within a pipeline, incorporating preprocessing steps, such as imputation and scaling as a backup. The model was trained on the training data (X_train, y_train) and evaluated on the test data (X_test, y_test) for each iteration. MSE was calculated for each fold, resulting in a Mean MSE of 5.23e+14, with a Standard Deviation of 9.99e+13.The high mean MSE depicts the model's predictions deviated significantly from the actual values. The large standard deviation suggests variability in the model across different folds, displaying sensitivity to training and test data splits. The linear regression model, is not performing well on this dataset. The mean squared error and the high standard deviation suggests inconsistencies across different folds. This might mean that the model is either too simple for the data or that the features being used are not informative enough to predict the target variable. Based on further evaluation of the XGBRegressor model, we infer that the Linear Regression model is in fact to simple for the data we are presenting and can be filtered out as a regression choice.

B. K-Fold Cross-Validation Execution

For Grid Search Cross-Validation, we used the GridSearchCV function from scikit-learn.We defined a parameter grid to search across, specifying different parameters combinations such as learning_rate, max_depth, and n_estimators. The Grid Search used 5-fold cross-validation (cv=5), with MSE as the evaluation metric (scoring='neg_mean_squared_error'). The model was fitted on the entire dataset (X, y), and the best combination of parameters was based on the lowest MSE. The best parameters for the XGBRegressor model are shown in Table I.

TABLE I
BEST PARAMETERS FOR XGBREGRESSOR

Parameter	Value
model_learning_rate	0.3
model_max_depth	9
model_n_estimators	200

TABLE II
MODEL EVALUATION METRICS

Model	MSE	R-squared	Std. Dev.
LinearRegression	5.23×10^{14}	null	9.99×10^{13}
RandomForestRegression	null	null	null
XGBRegression	1.37×10^{13}	0.978	null

The XGBRegressor model had an MSE of 1.37e+13 and an R-squared value of 0.978. The lower MSE obtained with the XGBRegressor means it preforms better compared to the linear regression model. Additionally, the runtime of the RandomForestRegressor model was unfeasible, which led to its exclusion from further analysis. All of the data is specified in Table II. We attempted to perform Leave-One-Out Cross-Validation (LOOCV) to evaluate the Linear Regression model's performance. However, LOOCV had a very slow execution speed, making it unusable for this project. LOOCV provides unbiased estimates of model performance, yet, its runtime increases with larger datasets, making it unrealistic for real-world applications such as this.

TABLE III
BEST PARAMETERS FOR XGBCLASSIFIER

Parameter	Value
model_learning_rate	0.3
model_max_depth	6
model_n_estimators	200

XGBClassifier first sets up and runs a grid search on an XGBClassifier to predict whether the 'fullval' value of properties exceeds 300,000, using features like block number, tax class, building class, lot front, lot depth, number of stories, and year. This is a classification problem where outcomes are categorical (1 if above 300,000, 0 otherwise). The data undergoes preprocessing where numerical features are scaled and categorical features are one-hot encoded. The model is optimized using GridSearchCV for hyperparameters like the number of estimators, learning rate, and maximum depth, aiming for the highest accuracy, as depicted in Table III.

In contrast, XGBRegressor uses a process for feature engineering and prediction, which is used for regression problems. Depicted in Table I, the 'fullval' itself is predicted as a continuous variable rather than categorizing its value. The data preprocessing steps are similar, involving scaling and encoding. The model, however, uses the XGBRegressor with settings for regression, including an objective function tailored for squared errors. The performance of the regression model is evaluated using the MSE and R-squared value, which measure prediction accuracy and the proportion of variance in the dependent variable that is predictable from the independent variables, respectively. The key difference between these approaches lies in the nature of the problem and the model. XGBClassifier is used for classification tasks where the output is categorical. XGBRegressor is used for regression tasks

TABLE IV
CLASSIFICATION REPORT

Category	Precision	Recall	F1-Score	Support
0	0.83	0.89	0.86	187148
1	0.87	0.81	0.84	177045
Macro Avg	0.85	0.85	0.85	364193
Weighted Avg	0.85	0.85	0.85	364193
Overall Accuracy		0.8509		

where the output is a continuous value.

V. OPTIMIZATION

A. Optimization with Grid-Search

Once a model is chosen based on the comparison and cross-validation results, optimization involves fine-tuning the model's hyperparameters or adjusting its configuration to improve performance. The MSE is 13679515562653.578 and the R-squared is 0.9777 after Grid Search. The most-probable explanation the mean square error is excessively high with a good R-squared value is the nature of our dataset. Since we are dealing with very high property values, it contributes to the high MSE. Furthermore, it is likely the model is having difficulties with small subsets of data. For example, small residential building data is being referenced against New York's skyscrapers, which may explain this unusual distribution. Thus we can infer that the model performs well for the majority of our dataset. Since it performs poorly on small subsets of data, further processing to separate outliers such as skyscrapers may aid in reducing the MSE.

B. Further Analysis Based on Visualizations

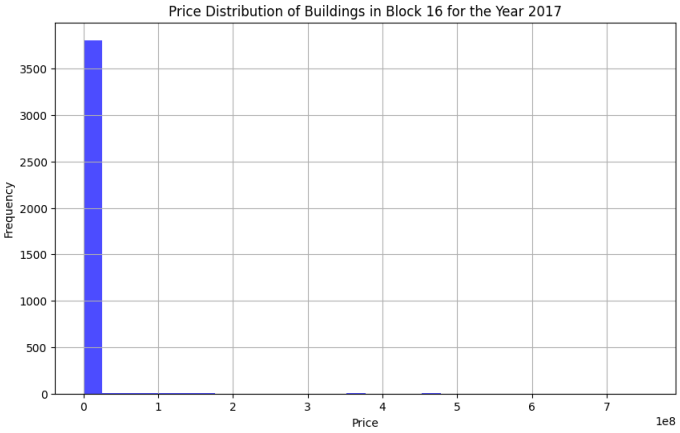

Fig. 9. 2017 Price Distribution of Buildings in Block 16 Before Outlier Removal.

Visualizing data offers valuable insights into our dataset, providing a visual that eases understanding ans guides decisions on how to proceed with data manipulation. In this context, the bar plot in Figure V-B illustrates the distribution of building valuations within block 16 in 2017. It illustrates a concentration of values within the range of 100,000 to 800,000, suggesting a common valuation pattern among properties in the block. However, the plot also reveals outliers at both ends of the spectrum, with some properties exhibiting high values nearing 700,000,000 and others showing remarkably low values approaching zero. These outliers, may represent unique cases or anomalies influenced by factors such as property condition, location-specific characteristics, or the type of building represented. Contextualizing these findings within real estate in New York City depicts the complexity of property valuation across blocks. It's important to mention that this is only for one block of one year. New York has hundreds of blocks and the price valuation differs by year, so the trends are exacerbated when you consider the vastness of our dataset. Visualizing these trends underlines the need for data analysis and interpretation for informed decision-making within real estate.

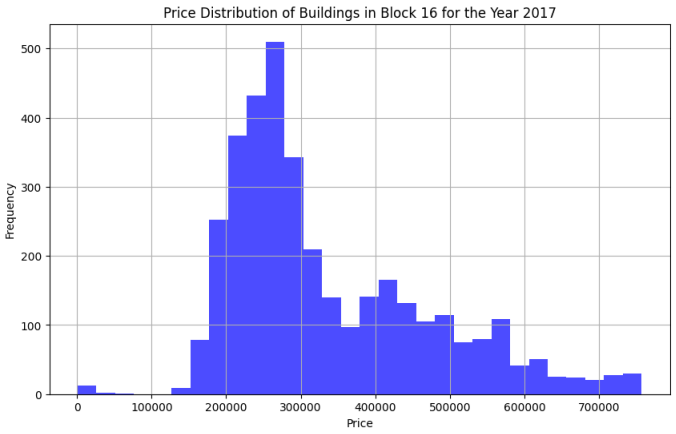

Fig. 10. 2017 Price Distribution of Buildings in Block 16 After Outlier Removal

The data in figure V-B is right skewed and is normally distributed. It depicts the data after outlier removal which provides us with a more detailed image of our dataset. Using the data after outlier removal can help improve our algorithm by lowering the MSE.

C. User Interactivity

The data preparation for the heat map was altered to provide a more relevant visual. Our approach revolved around cleaning and preparing a dataset of property values, which involves removing incomplete records and eliminating statistical outliers based on the IQR to address extreme value discrepancies. We applied a logarithmic transformation to the 'FULLVAL' property value column to normalize the data and mitigate skew, although we opted to utilize the original 'FULLVAL' for model training and predictions.

Our predictive model, built with an XGBRegressor, is trained using block, tax class, building class, lot frontage, lot depth, stories, and the year. This setup is encapsulated within a Scikit-learn pipeline configured to manage preprocessing for both numerical and categorical data automatically. Key preprocessing steps include imputation for missing values and

scaling/encoding to ensure data is appropriately formatted for modeling.

The interactive component of the system allows users to input a specific year to generate predictions, dynamically adjusting the dataset to reflect this temporal aspect. This functionality is particularly useful for analyzing trends or making future projections based on past and current data. After the model makes predictions for the specified year, these values are visualized on a geographic heatmap V-C.

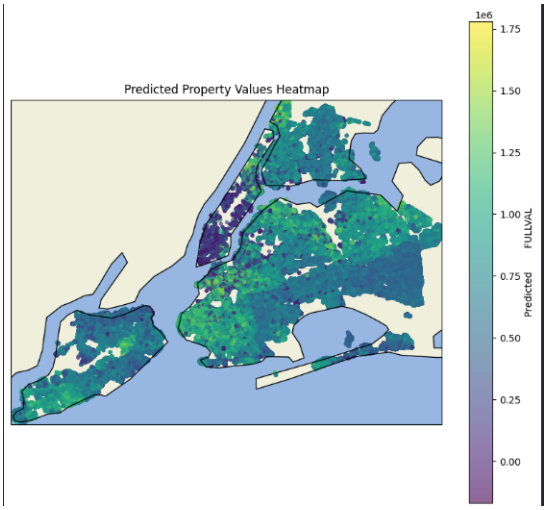

Fig. 11. Heat Map of Predicted FullVal for 2025

For the visualization, we utilize Cartopy within Matplotlib to overlay the predicted property values onto a map. The data points are represented in a color gradient—ranging from blue to red—to depict the spectrum of property values from low to high. This not only makes the data easier to interpret visually but also highlights geographic patterns and anomalies in property valuation. The color mapping and geographic detailing provide a rich, intuitive understanding of spatial variations in property values, making it a powerful tool for stakeholders needing to make informed decisions based on property market dynamics.

```
# Ask user for input and output cost prediction

# Function to handle "N/A" input and convert it to None (which pandas interprets as np.nan)
def get_input(prompt, input_type=str):
    user_input = input(prompt)
    if user_input.lower() == 'n/a':
        return None
    return input_type(user_input)

block =   get_input("Enter the block number (or N/A if unknown): ", int)
taxclass =get_input("Enter the tax class (or N/A if unknown): ")
bldgcl =  get_input("Enter the building class (or N/A if unknown): ")
ltfront = get_input("Enter the lot frontage (or N/A if unknown): ", float)
ltdepth = get_input("Enter the lot depth (or N/A if unknown): ", float)
stories = get_input("Enter the number of stories (or N/A if unknown): ", int)
```

Fig. 12. Code to take User Input

The model can also take input for block, tax class, building class, lot frontage, lot depth, stories, and the year and provide a prediction for a specific property. This system utilized the predictive model described in depth in sections II, III, and IV.

Conclusion

From completing the task of designing our own machine-learning pipeline with cross-validation and optimization, we were able to discover more regarding the intricacies of our dataset. Beginning with data pre-processing, we addressed missing values through imputation and ensured uniformity in feature scaling to prepare the dataset for analysis. The implementation of cross-validation techniques then enabled us to assess model performance, allowing for a more accurate evaluation of our chosen classifiers. Meeting project requirements, we experimented with different cross-validation approaches, identifying the most generalized model for our dataset. Employing GridSearchCV, we systematically explored parameter combinations, striving to improve the model's generalizability and predictive accuracy. Throughout the project, we reported evaluation metrics, including ROC curves for classification tasks and error metrics for regression models. Being able to visualize the repeated stratified k-fold cross-validation furthered our understanding of model performance across multiple iterations. We also aim to incorporate an AI model to update with real time events and produce a better human-like interface. Our model initially predicted values within a few hundred-thousand dollars of the market value, but it gives an exact number. Have added a classification technique that takes a categorical approach instead to better evaluate our model's metrics. Overall, we hope to continue our development of a model for predicting property value that can be implemented for any location.

References

[1] D. of F. (2020, May 26). Property valuation and assessment data: NYC open data.Property Valuation and Assessment Data—NYC Open Data. Retrieved April 4, 2024,from https://data.cityofnewyork.us/City-Government/Property-Valuation-and-Assessment-Data/yjxr-fw8i/about_data

[2] Finance, D. of. (2024, January 23). Property valuation and assessment data tax classes 1,2,3,4: NYC open data. Property Valuation and Assessment Data Tax Classes 1,2,3,4 — NYC Open Data. Retrieved April 4, 2024, from https://data.cityofnewyork.us/City-Government/Property-Valuation-and-Assessment-Data-Tax-Classes/8y4t-faws/about_data

[3] Eswpad. (2024, January 12). Understanding "Zero" and "Nil" Valuation with e.surv Surveyors. e.surv Chartered Surveyors. https://www.esurv.co.uk/home-owners/what-is-a-zero-valuation