Real Estate Price Prediction

Web Scrapping:

For data sourcing, I scrapped the data from **realtor** website, focusing on listings in New York state.
Libraries used:

- Requests
- Json
- Pandas

For realtor, the website was not statically embedded in the html code, so i rendered by sending a POST request to the API of realtor.com.The responses returned in json formatted string. My objective was to fake the POST request ton pretend I was the website requesting data from website.

json_data = r.json()

The json_data variable contains list of the real estate listings and I will use it to extract information from it. I am looping over each item to create a feature dictionary so I can append to list and after create a pandas data frame.

My data frame has originally 18 columns consisting of numerical as well as categorical features.

I scrapped the data from 206 different pages with real estate data from the city of New York by iterating over each of the 206 pages (by sending a request for each page with customised request payload).

There are three dict values that needed to be adjusted for request. The offset key is number that gets increased by 42 for each page.

```
def send_request(page_number: int, offset_parameter: int):
    ...
    json_body["variables"]["offset"]= offset_parameter
    ...
```

After running the entire script I got a data set containing 8652 rows and 179 columns.

Data Cleaning:

After saving the file in csv format, I loaded it in another ipynb file. The goal now is to clean the data, handling na values and feature selection (basic). For for this I used pandas. I started with dropping columns that I had analysed not to be relevant(either too many missing values or they were highly correlated with each other causing the problem of multicollinearity).

I handled the na values with different strategies for each column I was left with after dropping.

- I handled ,missing county and postal code values by replacing na values with similar values where either my postal code or county values were given.
- I handled the year_built missing values by taking average of years for a specific postal code (groupby).
- There was only one missing value for price, so I dropped it.

Similarly, I handled other missing values. Before I saved the data cleaned file, I was left with 8565 entries and 55 columns.

Data Preprocessing:

Since I was going to use two models, I preprocessed the data accordingly.

For Linear Regression:

- I selected specific columns ('beds', 'baths', 'postal_code', 'price', 'sqft') from the DataFrame and stored them in a new DataFrame called 'correlated_features'. I also filtered out rows where the 'price' was less than $200,000.
- Deleted the independent variables that were highly correlated to each other to avoid multicollinearity.
- I was left with three most important variables for my dependent variable due to their high correlation with the dependent variable.
- I calculated the correlation matrix for the selected features and visualized it using a heatmap.
- I defined a function called 'handle_skewness' to handle skewed numerical features by applying the Yeo-Johnson transformation. This function also handles zero and negative values by adding a constant.
- I looped through the numerical features in 'correlated_features' and applied the 'handle_skewness' function to each feature, printing the lambda value used for the transformation.
- After transforming the skewed features, I scaled the numerical features using the StandardScaler from scikit-learn.

- I visualized the distributions of the transformed features using histograms with kernel density estimation (KDE) plots.
- I created scatter plots to visualize the relationships between each feature and the target variable ('price').
- I performed data normalization and standardization on the feature matrix (X) by subtracting the mean and dividing by the standard deviation.
- I split the data into training and test sets using the train_test_split function from scikit-learn.

For Random Forest:

- Dropped columns with object type due to them being less important. These mostly consisted of location and I already had postal codes that was actually correlated to the price movement.
- Removed the prices for rentals (because this model is focused only on buying houses).
- I plotted a histogram of the house prices along with a kernel density estimate to visualize the distribution.
- To address the skewness in the price distribution, I applied a log transformation to the 'price' column.
- I filled in any missing values in the numerical features with the mean value.
- I identified the features with high skewness (absolute value greater than 0.75) and applied a log transformation to those skewed features.
- I separated the features (X) and target variable (y) from the dataframe.
- I created scatter plots to visualize the relationship between selected features (beds, baths, postal code, year built) and the target variable (price).
- I generated box plots for all the features to identify potential outliers.
- I calculated the correlation between features and the target variable, identifying highly correlated features.
- I created a heatmap to visualize the correlation between the highly correlated features.
- I set up a column transformer to standardize and normalize the numerical features.

Model Training and Evaluation Metrics

For Linear Regression

Model Training and Tuning:

- The feature matrix (X) and target variable (y) were separated from the preprocessed DataFrame.
- The data was split into training and test sets using train_test_split from scikit-learn, with a test set size of 20% and a random state of 42.
- The training and test feature matrices were standardized using the StandardScaler from scikit-learn.
- A constant term (column of ones) was added to the standardized training and test feature matrices.
- A Linear Regression model was instantiated using the LinearRegression class from scikit-learn.
- The Linear Regression model was fitted to the standardized training feature matrix and target variable.

Evaluation and Performance Metrics:

- Predictions were made on the test set using the fitted Linear Regression model.
- The R-squared (coefficient of determination) was calculated on the test set using the score method of the Linear Regression model, which gave a value of 0.464. This indicates that the model can explain about 46.4% of the variance in the target variable.

Additional evaluation metrics were calculated, including:

- Mean Absolute Error (MAE): 0.6015235323378247
- Mean Squared Error (MSE): 0.5691010730260931
- Root Mean Squared Error (RMSE): 0.7543878796919349

A scatter plot was created to visualise the actual and predicted prices, with a line plot showing the predicted prices.

Analysis of the Graph:

- The scatter plot shows the actual prices (blue dots) and the predicted prices (red line) plotted against the correlated features.
- There is a significant spread of actual prices around the predicted price line, indicating that the model's predictions have a considerable amount of error.
- The predicted price line appears to follow a somewhat linear trend, but there are several instances where the predicted prices deviate substantially from the actual prices.
- The R-squared value of 0.464 and the relatively high values of the error metrics (MAE, MSE, RMSE) suggest that the Linear Regression model may not be the best

fit for this dataset, and further tuning or exploration of other models might be necessary to improve the predictions.

For Random Forest

Visualisations:

- Scatter plots were created to visualize the relationship between selected features (beds, baths, postal code, year built) and the target variable (price).
- Box plots were generated for all features to identify potential outliers.
- A heatmap was created to visualize the correlation between the highly correlated features and the target variable.
- A bar plot was used to visualize the feature importances of the trained Random Forest Regressor model.
- Residual plots and actual vs. predicted values plots were created to evaluate the model's performance visually.

Actual vs. Predicted Values Plot:

- The plot shows the actual target values on the x-axis and the predicted values from the model on the y-axis.
- The points are clustered around the diagonal line, indicating a decent fit between actual and predicted values.
- However, there is some deviation from the diagonal line, suggesting some errors in the predictions.
- The spread of points increases as the values get larger, indicating potentially higher errors for higher-priced houses.

Residual Plot:

- This plot shows the residuals (difference between actual and predicted values) on the y-axis against the predicted values on the x-axis.
- The residuals are scattered around the horizontal zero line, which is desirable.
- However, the spread of residuals appears to increase for higher predicted values, indicating potential heteroscedasticity (non-constant variance of residuals).
- There are some large positive and negative residuals, suggesting that the model may have difficulty accurately predicting certain houses.

Random Forest Feature Importance:

- This bar plot shows the relative importance of each feature in the Random Forest Regressor model.
- The most important features are 'sqft', 'postal_code', 'year_built', and 'community'.
- Features like 'dual_master_bedroom' and 'dog_rated' appear to have relatively low importance.
- This visualization is helping to identify the most influential features for predicting house prices.

Correlation Heatmap:

- This heatmap shows the correlation between the target variable ('price') and selected features ('baths', 'sqft').
- 'price' has a strong positive correlation with 'sqft' and a moderate positive correlation with 'baths'.
- This aligns with the intuition that larger houses (higher sqft) and more bathrooms tend to be more expensive.

Box Plots of Features:

- These box plots show the distribution and potential outliers for each feature.
- The feature 'postal_code' appears to have several outliers.
- Most other features do not seem to have significant outliers based on these box plots.

Model Training and Tuning:

- A Random Forest Regressor model was chosen by me for the regression task.
- A grid of hyperparameters (n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, bootstrap) was defined for the Random Forest Regressor.
- Randomised search cross-validation was performed to find the best hyperparameters for the Random Forest Regressor.
- The best hyperparameters found were: n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features='sqrt', max_depth=40, bootstrap=False.
- The Random Forest Regressor model was trained with the best hyperparameters on the training data.

Evaluation and Performance Metrics:

- Root Mean Squared Error (RMSE) was calculated on the test set, and the reported value is 0.42.
- R-squared (R^2) was calculated on the test set, and the reported value is 0.81.
- The mean of the original target variable was reported as 13.54.

Root Mean Squared Error (RMSE): 0.42

- The RMSE is a measure of the average magnitude of the errors between the predicted and actual values.
- In this case, an RMSE of 0.42 indicates that, on average, the model's predictions deviate from the actual house prices by approximately 0.42 units (log-transformed).
- A lower RMSE value is better, as it implies smaller errors and a better fit between predicted and actual values.

R-squared (R^2): 0.81

- A value of 0.81 suggests that approximately 81% of the variance in house prices is explained by the features used in the Random Forest model.
- R-squared values range from 0 to 1, with higher values indicating a better fit of the model to the data.
- An R-squared of 0.81 is a good fit, suggesting that the model captures a significant portion of the variability in house prices.

Additional Observations:

- The "price" feature appears to be highly correlated with features like "sqft", "baths", and "postal_code" based on the correlation heatmap.
- The box plots show some features with potential outliers that may need further investigation or treatment.
- The residual plot suggests some heteroscedasticity in the residuals, which could be an area for improvement in the model.
- The actual vs. predicted values plot shows a reasonably good fit, with most points clustered around the diagonal line, indicating decent predictive performance.

Since Random Forest has much better performance, I have decided it as my main model.

Business Implications:

- The model's reasonably good performance (RMSE of 0.42 and R-squared of 0.81) suggests that it can be a useful tool for predicting house prices, which can aid in various business decisions.
- Real estate agencies or property investment firms can leverage this model to estimate property values more accurately, enabling better pricing strategies and investment decisions.
- Mortgage lenders and financial institutions can use the model to assess the collateral value of properties more reliably, facilitating risk management and loan approval processes.

Utilising the Model for Profitability and Operational Efficiency:

- Integrate the model into real estate listing platforms or property valuation tools to provide data-driven price estimates for sellers and buyers, enhancing transparency and trust in the market.

- Automate the property valuation process for large portfolios or frequent assessments, reducing manual effort and increasing operational efficiency.
- Combine the model's predictions with other data sources (e.g., market trends, neighbourhood information) to create comprehensive property valuation reports, offering value-added services to clients.
- Identify undervalued or overvalued properties based on the model's predictions, enabling strategic investment decisions and potential arbitrage opportunities.
- Analyse the feature importances to understand the key drivers of property values, informing marketing strategies and targeted property improvements for higher returns.

Recommendations for Future Work:

- Advanced techniques, such as ensemble modeling or neural networks,for better model's predictive accuracy.
- Develop a user-friendly interface or web application to streamline the deployment and usage of the model for various stakeholders.
- Continuously monitor and update the model as new data becomes available, ensuring its relevance and accuracy over time.
- Expand the model to encompass additional use cases, such as rental property valuations or commercial real estate assessments.