Capstone Project

**Predicting Property Price in a specific Location Using**

**Machine Learning**

**Submitted By - Priyanka Verma**

**Course Name - Post Graduate program in AI and Data Science**

**Instructor’s Name - Miss Aishwarya**

**Date - 20/11/24**

**1. Introduction to the Project**

This capstone project aims to develop a predictive model for real estate prices in a specific geographic area by leveraging machine learning techniques. Given the increasing demand for accurate property price forecasting, this study seeks to provide potential buyers and real estate investors with actionable insights based on historical property data and various influencing factors.

The project involves several steps: Data Collection,Data Preprocessing,Exploratory Data Analysis, Model Selection,Model Evaluation etc. The outcome of this project is intended to illuminate the factors driving property values in the chosen location and to equip stakeholders with a robust tool for informed decision-making in property investments.

**2.Objectives of the Project**

**1.Data Collection:** Utilizing **datasets that include property features such as location, size, number of bedrooms, age of construction and local amenities, alongside historical price figures.**

**2.Data Preprocessing: Cleaning the data, handling missing values and transforming categorical variables to ensure they are suitable for analysis.**

**3. Exploratory Data Analysis (EDA) : Analyzing the dataset to identify pattern and correlation between property characteristics and prices, visualizing the data using tools like Matplotlib and seaborn.**

**4.Model Selection: Comparing several machine learning algorithms like Linear regression, Random Forest regressor, Gradient Boosting to determine the most effective model for price prediction.**

**5.Model Evaluation: Using performance metrics such as RMSE (Root mean square error) and r2 to assess model accuracy and reliability.**

**3. Flow Chart of Operations**

**Import libraries**

**Download dataset**

**Read the dataset using pandas**

**Observation of the dataset**

**Preprocessing of the dataset**

**Perform EDA on the dataset**

**Encode the categorical dataset**

**Standardization and PCA technique**

**Split dataset into train and test**

**Apply Algorithms for Model selection**

**Model evaluation**

**4. Python Codes:**

**Import libraries**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**# Display all the columns of the dataframe**

**pd.pandas.set\_option('display.max\_columns',None)**

**from sklearn.preprocessing import StandardScaler, LabelEncoder, OrdinalEncoder**

**from sklearn.impute import SimpleImputer**

**from sklearn.decomposition import PCA**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.metrics import r2\_score, mean\_squared\_error**

**from sklearn.ensemble import RandomForestRegressor**

**from sklearn.model\_selection import cross\_val\_score**

**from sklearn.linear\_model import LinearRegression**

**import warnings**

**warnings.filterwarnings('ignore')**

**Load dataset**

**df=pd.read\_csv("Property\_data.csv")**

**df.head()**

**Observations**

**df.shape**

**df.info()**

**df.describe()**

**Preprocessing**

**df = df.drop('PropertyID', axis=1)**

**# Identify numeric and categorical columns**

**numeric\_cols = df.select\_dtypes(include=['int64', 'float64']).columns**

**categorical\_cols = df.select\_dtypes(include=['object']).columns**

**# print list of Numerical columns name.**

**numeric\_cols**

**# print list of catgorical columns name.**

**categorical\_cols**

**# Print info about missing values**

**print("Missing values summary:")**

**print(df.isnull().sum()[df.isnull().sum() > 0])**

**-----------------------------------------------------------------------------------------------------------**

**# Define ordinal columns and their order to seprate them from nominal columns.**

**ordinal\_columns = {**

**'OverallQual': range(1, 11),**

**'OverallCond': range(1, 11),**

**'ExterQual': ['Po', 'Fa', 'TA', 'Gd', 'Ex'],**

**'ExterCond': ['Po', 'Fa', 'TA', 'Gd', 'Ex'],**

**'KitchenQual': ['Po', 'Fa', 'TA', 'Gd', 'Ex'],**

**'HeatingEfficiency': ['Po', 'Fa', 'TA', 'Gd', 'Ex'],**

**'BsmntFinish': ['None', 'Unf', 'LwQ', 'Rec', 'BLQ', 'ALQ', 'GLQ'],**

**'BasementQual': ['None', 'Po', 'Fa', 'TA', 'Gd', 'Ex'],**

**'BasementCond': ['None', 'Po', 'Fa', 'TA', 'Gd', 'Ex']**

**}**

**# Handle missing values by different techiques like fill with mean,mode,0,none as par needs.**

**df['PropertyFrontage'] = df['PropertyFrontage'].fillna(df['PropertyFrontage'].mean())**

**df['ExteriorCladdingArea'] = df['ExteriorCladdingArea'].fillna(0)**

**df['Alley'] = df['Alley'].fillna('None')**

**df['ExteriorCladdingType'] = df['ExteriorCladdingType'].fillna('None')**

**df['BsmntFinish'] = df['BsmntFinish'].fillna('None')**

**df['BsmntMaintenance'] = df['BsmntMaintenance'].fillna('None')**

**df['BsmntVisibility'] = df['BsmntVisibility'].fillna('None')**

**df['BsmntFinRat1'] = df['BsmntFinRat1'].fillna('None')**

**df['BsmntFinQual1'] = df['BsmntFinQual1'].fillna('None')**

**df['Electrical'] = df['Electrical'].fillna(df['Electrical'].mode()[0])**

**df['QualFireplace'] = df['QualFireplace'].fillna('None')**

**df['BasementType'] = df['BasementType'].fillna('None')**

**df['BasementYrBlt'] = df['BasementYrBlt'].fillna(0)**

**df['BasementFinish'] = df['BasementFinish'].fillna('None')**

**df['BasementQual'] = df['BasementQual'].fillna('None')**

**df['BasementCond'] = df['BasementCond'].fillna('None')**

**df['PoolQC'] = df['PoolQC'].fillna('None')**

**df['BoundaryFeatures'] = df['BoundaryFeatures'].fillna('None')**

**df['AddFeatures'] = df['AddFeatures'].fillna('None')**

**# To check all null values filled or not.**

**df.isnull().sum()**

**----------------------------------------------------------------------------------------------------------------------**

**EDA**

**# Univariate analysis**

**df.hist(figsize=(15,10),bins=30)**

**plt.tight\_layout()**

**plt.show()**

**------------------------------------------------------------------------------------------------------------------------**

**# Define the function to remove outliers**

**def remove\_outliers\_conservative(df, columns):**

**df\_clean = df.copy()**

**total\_rows = len(df)**

**for column in columns:**

**Q1 = df\_clean[column].quantile(0.25)**

**Q3 = df\_clean[column].quantile(0.75)**

**IQR = Q3 - Q1**

**lower\_bound = Q1 - 3 \* IQR**

**upper\_bound = Q3 + 3 \* IQR**

**df\_clean = df\_clean[(df\_clean[column] >= lower\_bound) & (df\_clean[column] <= upper\_bound)]**

**rows\_removed = total\_rows - len(df\_clean)**

**print(f"Total rows removed: {rows\_removed} ({(rows\_removed/total\_rows)\*100:.2f}% of data)")**

**return df\_clean**

**# Define important features**

**important\_features = [**

**'PropPrice', 'OverallQual', 'GrLivArea', 'YearBuilt',**

**'1stFlrSF', '2ndFlrSF', 'PropertyFrontage', 'PropertySize'**

**]**

**# Remove outliers**

**df\_cleaned = remove\_outliers\_conservative(df, important\_features)**

**# Display the shape of cleaned dataset**

**print("\**

**Shape of cleaned dataset:", df\_cleaned.shape)**

**# Create box plots for key features**

**plt.figure(figsize=(15, 10))**

**key\_features = ['PropPrice', 'OverallQual', 'GrLivArea', 'YearBuilt']**

**for i, feature in enumerate(key\_features, 1):**

**plt.subplot(2, 2, i)**

**plt.boxplot([df[feature], df\_cleaned[feature]], labels=['Before', 'After'])**

**plt.title(f'{feature} Distribution')**

**plt.xticks(rotation=45)**

**plt.tight\_layout()**

**plt.show()**

**# Save the cleaned dataset**

**df\_cleaned.to\_csv('cleaned\_property\_data.csv', index=False)**

**print("\**

**Cleaned dataset has been saved as 'cleaned\_property\_data.csv'")**

**# Display summary statistics**

**print("\**

**Summary statistics of key features after cleaning:")**

**print(df\_cleaned[key\_features].describe())**

**--------------------------------------------------------------------------**

**plt.figure(figsize=(10, 6))**

**sns.histplot(df['PropPrice'], bins=30, kde=True )**

**plt.title('Distribution of Property Prices')**

**plt.xlabel('Property Price')**

**plt.ylabel('Frequency')**

**plt.show()**

**# Bivariate analysis for the highlighted univariate trends of visualization**

**# Create new dataframe**

**property\_data = pd.DataFrame(df)**

**# Select key features for bivariate analysis**

**key\_features = ['OverallQual', 'GrLivArea', 'YearBuilt', '1stFlrSF', '2ndFlrSF']**

**# Create scatter plots**

**plt.figure(figsize=(15, 10))**

**for i, feature in enumerate(key\_features, 1):**

**plt.subplot(2, 3, i)**

**sns.scatterplot(data=property\_data, x=feature, y='PropPrice')**

**plt.title(f'{feature} vs PropPrice')**

**plt.xlabel(feature)**

**plt.ylabel('PropPrice')**

**plt.tight\_layout()**

**plt.show()**

**# Calculate and display correlation matrix**

**correlation\_matrix = property\_data[key\_features + ['PropPrice']].corr()**

**plt.figure(figsize=(10, 8))**

**sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', center=0)**

**plt.title('Correlation Heatmap')**

**plt.show()**

**# Additional analysis: Box plots for categorical variables vs price**

**plt.figure(figsize=(12, 6))**

**sns.boxplot(x='OverallQual', y='PropPrice', data=property\_data)**

**plt.title('Property Price Distribution by Overall Quality')**

**plt.show()**

**----------------------------------------------------------------------------------------------------**

**Multivariate analysis:**

**key\_features = ['PropPrice', 'OverallQual', 'GrLivArea', 'YearBuilt', 'OverallCond']**

**plt.figure(figsize=(12, 8))**

**sns.pairplot(property\_data[key\_features], diag\_kind='kde')**

**plt.tight\_layout()**

**plt.show()**

**# Create a heatmap of correlations for more features**

**extended\_features = ['PropPrice', 'OverallQual', 'GrLivArea', 'YearBuilt',**

**'OverallCond', '1stFlrSF', '2ndFlrSF', 'BsmtFullBath',**

**'BedroomUpLev', 'KitchenUpLev']**

**correlation\_matrix = property\_data[extended\_features].corr()**

**plt.figure(figsize=(12, 8))**

**sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', center=0)**

**plt.title('Extended Correlation Heatmap')**

**plt.show()**

**# Print summary of multivariate statistics**

**print("\**

**Multivariate Summary Statistics:")**

**print(property\_data[extended\_features].describe())**

**--------------------------------------------------------------------------------------------------------------**

**# Encode ordinal variables**

**ordinal\_encoder = OrdinalEncoder()**

**for col, categories in ordinal\_columns.items():**

**if col in df.columns:**

**df[col] = pd.Categorical(df[col], categories=categories, ordered=True)**

**df[col] = df[col].cat.codes**

**---------------------------------------------------------**

**# Get remaining categorical columns (nominal)**

**nominal\_cols = df.select\_dtypes(include=['object']).columns**

**# Encode nominal variables using LabelEncoder**

**label\_encoder = LabelEncoder()**

**for col in nominal\_cols:**

**df[col] = label\_encoder.fit\_transform(df[col].astype(str))**

**# Scale numerical features**

**scaler = StandardScaler()**

**numeric\_cols = df.select\_dtypes(include=['int64', 'float64']).columns**

**df[numeric\_cols] = scaler.fit\_transform(df[numeric\_cols])**

**# Perform PCA for feature reduction**

**X = df.drop('PropPrice', axis=1)**

**y = df['PropPrice']**

**pca = PCA(n\_components=0.95) # Keep 95% of variance**

**X\_pca = pca.fit\_transform(X)**

**print("Original number of features:", X.shape[1])**

**print("Number of features after PCA:", X\_pca.shape[1])**

**print("Explained variance ratio:", sum(pca.explained\_variance\_ratio\_))**

**# Calculate average price by neighborhood and overall quality**

**neighborhood\_quality\_price=df.groupby(['Neighborhood', 'OverallQual'])['PropPrice'].mean().reset\_index()**

**# Sort by average price**

**neighborhood\_quality\_price=neighborhood\_quality\_price.sort\_values('PropPrice', ascending=False)**

**# Display top 15 combinations**

**print("Top 15 Neighborhood-Quality combinations by average price:")**

**print(neighborhood\_quality\_price.head(15))**

**# Create a heatmap**

**pivot\_table = neighborhood\_quality\_price.pivot\_table(**

**values='PropPrice',**

**index='Neighborhood',**

**columns='OverallQual'**

**)**

**plt.figure(figsize=(12, 8))**

**sns.heatmap(pivot\_table, cmap='YlOrRd', annot=True, fmt='.2f')**

**plt.title('Property Prices by Neighborhood and Overall Quality')**

**plt.tight\_layout()**

**plt.show()**

**# Box plot of prices by neighborhood**

**plt.figure(figsize=(15, 6))**

**sns.boxplot(x='Neighborhood', y='PropPrice', data=df)**

**plt.xticks(rotation=45, ha='right')**

**plt.title('Property Price Distribution by Neighborhood')**

**plt.tight\_layout()**

**plt.show()**

**-------------------------------------------------------------------------------------------------------**

**from sklearn.ensemble import RandomForestRegressor**

**from sklearn.model\_selection import cross\_val\_score**

**# Split the data**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_pca, y, test\_size=0.2, random\_state=42)**

**-------------------------------------------------------------------------**

**X\_train**

**----------------------------------------------------------------------------------------------------------------**

**y\_train**

**----------------------------------------------------------------------------------------------------------**

**# Train Random Forest model**

**rf\_model = RandomForestRegressor(n\_estimators=100, random\_state=42)**

**rf\_model.fit(X\_train, y\_train)**

**# Make predictions**

**y\_pred = rf\_model.predict(X\_test)**

**# Calculate metrics**

**r2 = r2\_score(y\_test, y\_pred)**

**rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))**

**------------------------------------------------------------------------------------------**

**# Perform cross-validation**

**cv\_scores = cross\_val\_score(rf\_model, X\_pca, y, cv=5, scoring='r2')**

**print("Model Performance Metrics:")**

**print("R2 Score:", r2)**

**print("RMSE:", rmse)**

**print("\**

**Cross-validation R2 scores:", cv\_scores)**

**print("Mean CV R2:", cv\_scores.mean())**

**print("CV R2 Standard deviation:", cv\_scores.std())**

**# Plot actual vs predicted values**

**plt.figure(figsize=(10, 6))**

**plt.scatter(y\_test, y\_pred, alpha=0.5)**

**plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--', lw=2)**

**plt.xlabel('Actual Price')**

**plt.ylabel('Predicted Price')**

**plt.title('Actual vs Predicted Property Prices')**

**plt.tight\_layout()**

**plt.show()**

**# Feature importance analysis**

**feature\_importance = pd.DataFrame({**

**'feature': range(X\_pca.shape[1]),**

**'importance': rf\_model.feature\_importances\_**

**})**

**feature\_importance=feature\_importance.sort\_values('importance', ascending=False).head(10)**

**plt.figure(figsize=(10, 6))**

**plt.bar(range(10), feature\_importance['importance'])**

**plt.xlabel('PCA Components')**

**plt.ylabel('Importance')**

**plt.title('Top 10 Important PCA Components')**

**plt.tight\_layout()**

**plt.show()**

**from sklearn.linear\_model import LinearRegression**

**lr = LinearRegression()**

**lr.fit(X\_train,y\_train)**

**y\_pred = lr.predict(X\_test)**

**print("intercept:",lr.intercept\_)**

**print("Slope :",lr.coef\_[0])**

**----------------------------------------------------------------------------------------**

**# Testing score of linear regression**

**Model\_accuracy = lr.score(X\_test,y\_test)\*100**

**Model\_accuracy**

**# Training score of linear regression**

**Model\_accuracy = lr.score(X\_train,y\_train)\*100**

**Model\_accuracy**

**----------------------------------------------------------------------------------------**

**from sklearn.metrics import r2\_score,mean\_squared\_error**

**mse = mean\_squared\_error(y\_test,y\_pred)**

**mse**

**-----------------------------------------------------------------------------------------------**

**# Calculate metrics**

**r2 = r2\_score(y\_test, y\_pred)**

**rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))**

**-----------------------------------------------------------------------------------------------**

**# Perform cross-validation**

**cv\_scores = cross\_val\_score(lr, X\_pca, y, cv=5, scoring='r2')**

**print("Model Performance Metrics:")**

**print("R2 Score:", r2)**

**print("RMSE:", rmse)**

**print("\Cross-validation R2 scores:", cv\_scores)**

**print("Mean CV R2:", cv\_scores.mean())**

**print("CV R2 Standard deviation:", cv\_scores.std())**

**# Plot actual vs predicted values**

**plt.figure(figsize=(10, 6))**

**plt.scatter(y\_test, y\_pred, alpha=0.5)**

**plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--', lw=2)**

**plt.xlabel('Actual Price')**

**plt.ylabel('Predicted Price')**

**plt.title('Actual vs Predicted Property Prices')**

**plt.tight\_layout()**

**plt.show()**

**-----------------------------------------------------------------------------------------------**

**from xgboost import XGBRegressor**

**xgb=XGBRegressor(objective='reg:squarederror', random\_state=42)**

**xgb.fit(X\_train,y\_train)**

**--------------------------------------------------------------------------------------------------------------**

**# Make predictions**

**y\_pred = xgb.predict(X\_test)**

**# Evaluate the model's performance**

**mse = mean\_squared\_error(y\_test, y\_pred)**

**r2 = r2\_score(y\_test, y\_pred)**

**print(f'Mean Squared Error: {mse:.2f}')**

**print(f'R-squared: {r2:.2f}')**

**-------------------------------------------------------------**

**# Plotting actual vs predicted prices**

**plt.figure(figsize=(10, 6))**

**plt.scatter(y\_test, y\_pred, alpha=0.6)**

**plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--') # 45-degree line**

**plt.xlabel('Actual Prices')**

**plt.ylabel('Predicted Prices')**

**plt.title('Actual vs. Predicted Prices')**

**plt.show()**

**---------------------------------------------------------------------------------------------------------------**

**Hyperparameter Tuning(optional)**

**# To improve the performance of our model, consider using techniques like Grid Search or Random Search to find the best hyperparameters.**

**from sklearn.model\_selection import GridSearchCV**

**param\_grid = {**

**'n\_estimators': [100, 200],**

**'max\_depth': [3, 5, 7],**

**'learning\_rate': [0.01, 0.1, 0.2]**

**}**

**grid\_search = GridSearchCV(estimator=XGBRegressor(objective='reg:squarederror'),**

**param\_grid=param\_grid, scoring='neg\_mean\_squared\_error', cv=3,verbose=1)**

**# Fit the grid search**

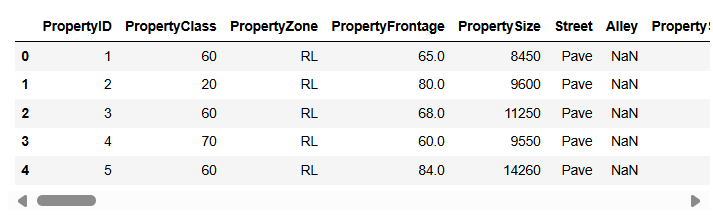
**grid\_search.fit(X\_train, y\_train)**

**# Output the best parameters found**

**print("Best parameters found: ", grid\_search.best\_params\_)**

**5. Screenshots of the Outputs**

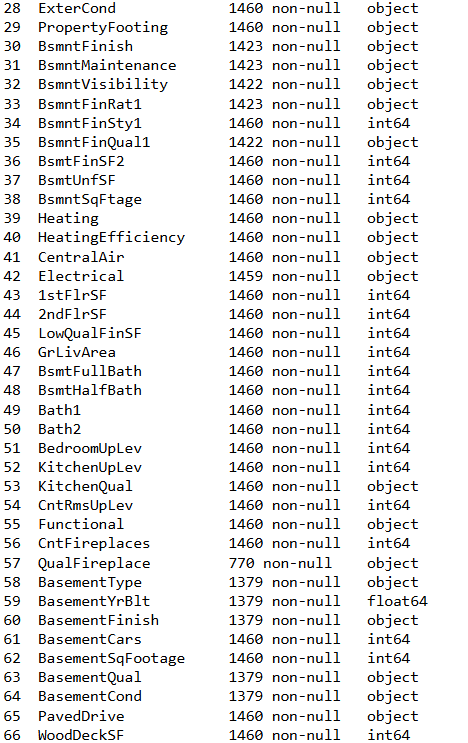
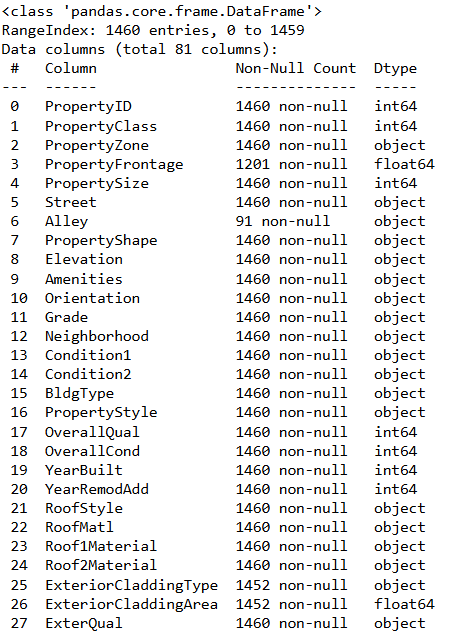
**Output - 1**

****

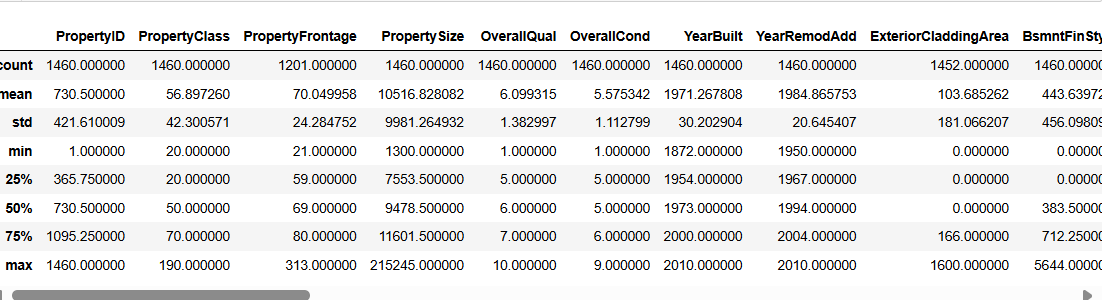
**Output – 2**

****

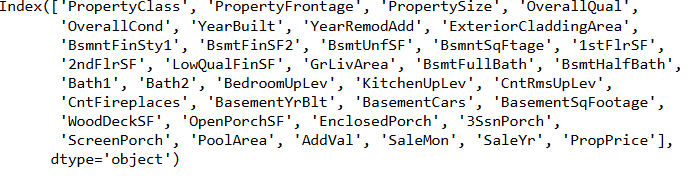
**Output -3**

****

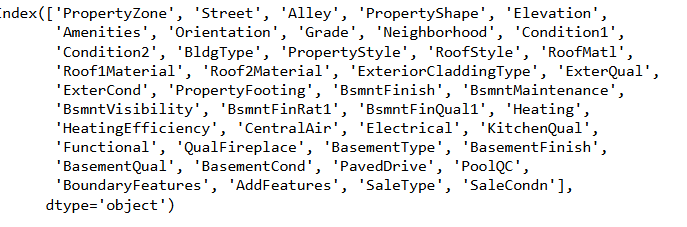
**Output – 5**

****

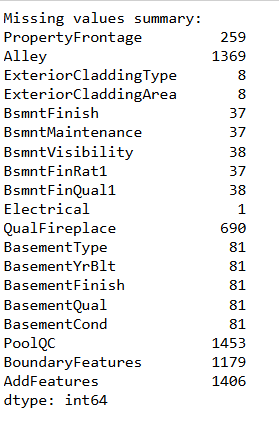
**Output - 8**

****

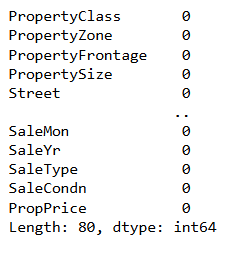
**Output – 9**

****

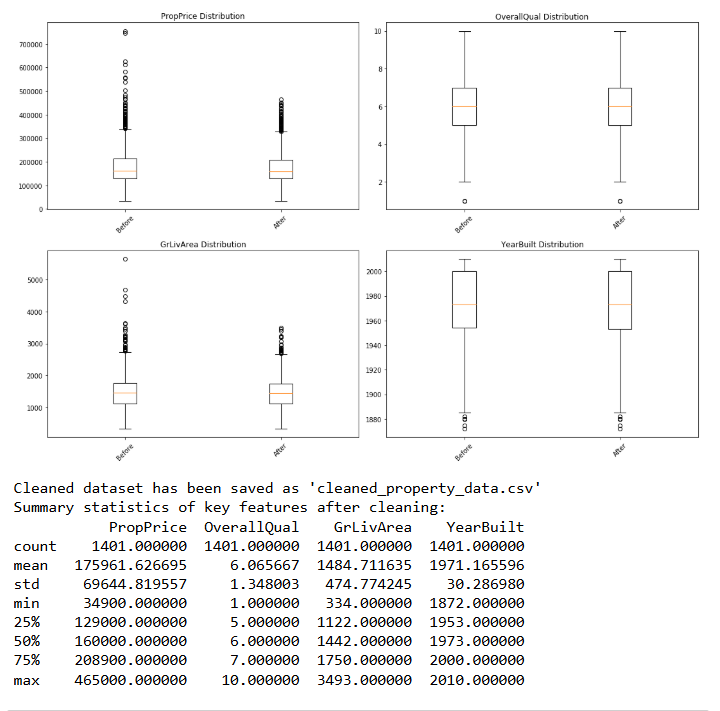
**Output - 10**



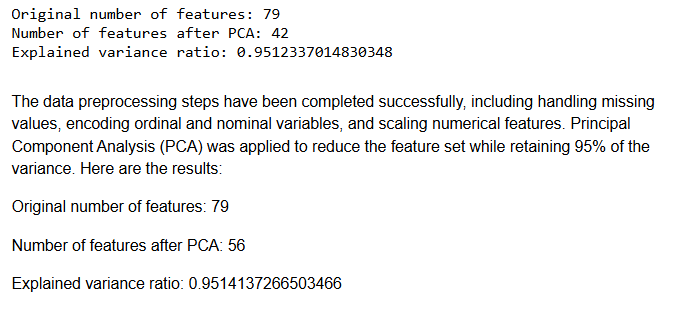
**Output - 12**



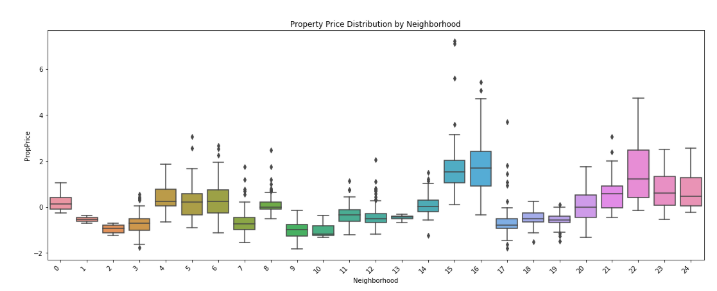
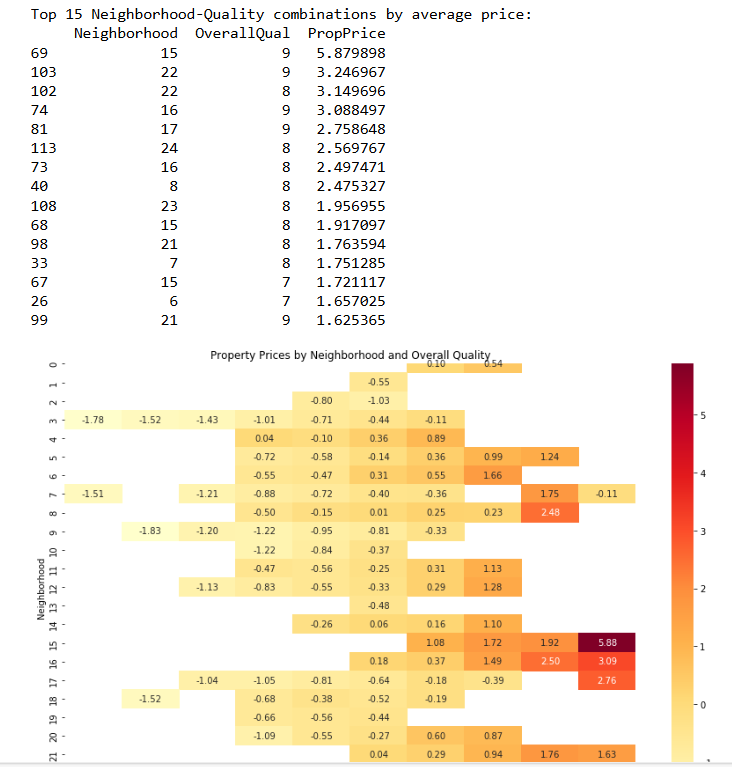
**Output – 15**

****

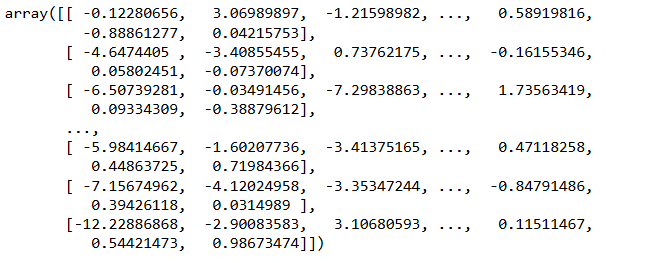
**Output – 26**

****

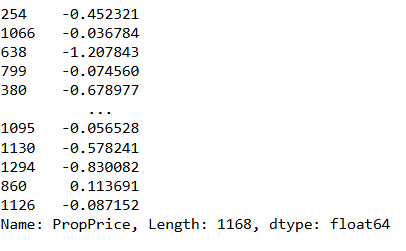
**Output- 27**

****

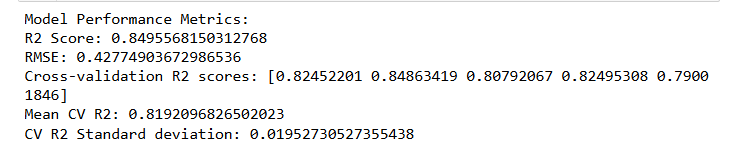
**Output- 29**

****

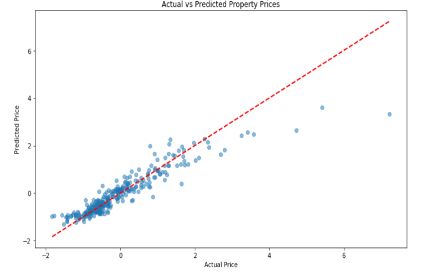
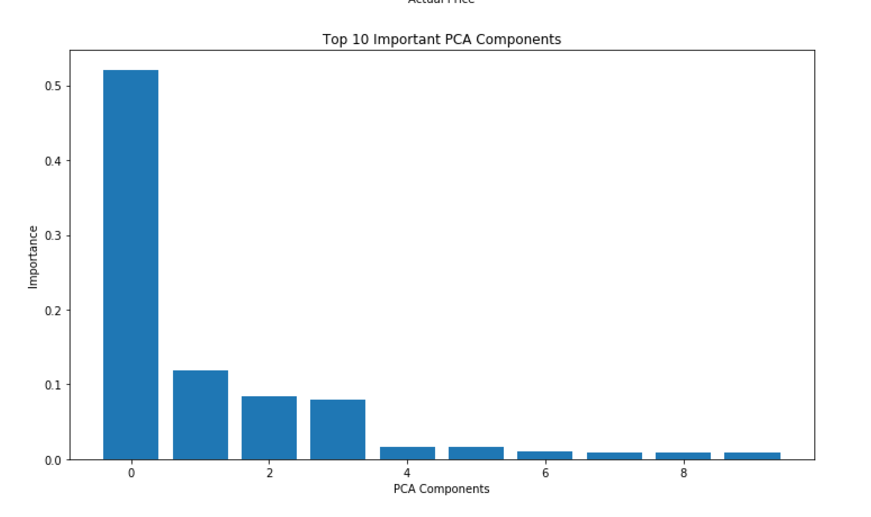
**Output – 30**

****

**Output – 35**

****

**Output – 36**

****

**Output – 38,39,40,41,43**

****

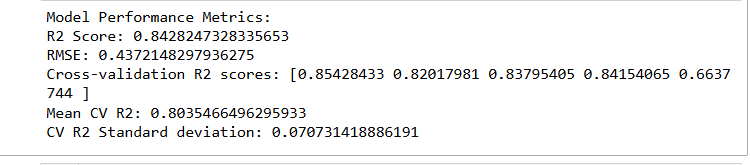
****

****

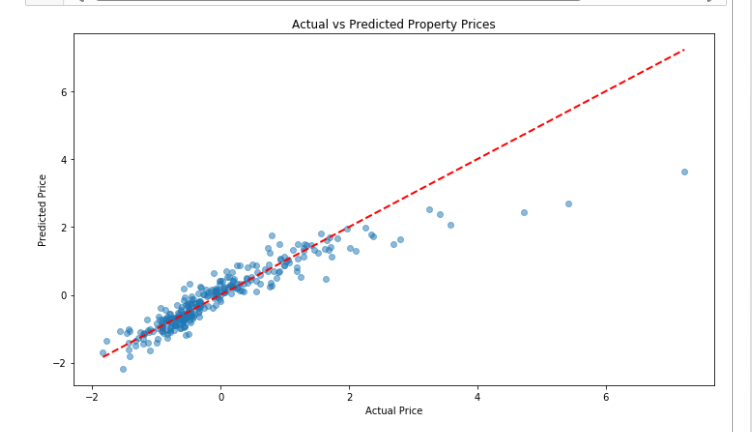
****

****

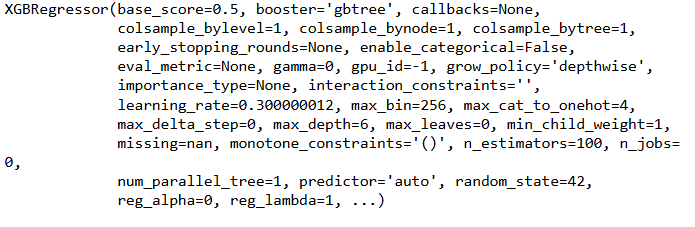
**Output- # Perform cross-validation**

****

**Output - # Plot actual vs predicted values**

****

**Output – 3rd Algorithm**

****

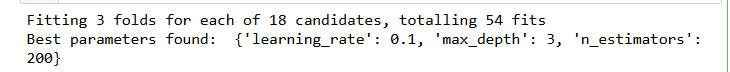
**Output - # Evaluate the model's performance**

****

**Output-# Plotting actual vs predicted prices**

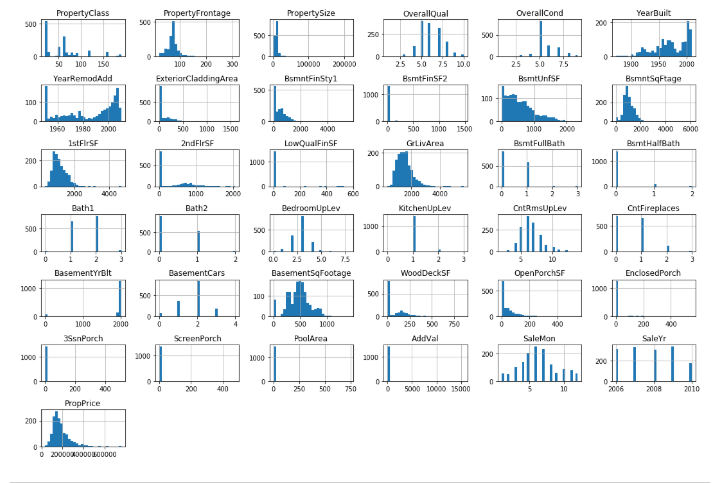
****

**Output - # To improve the performance of our model, consider using techniques like Grid Search or Random Search to find the best hyperparameters.**

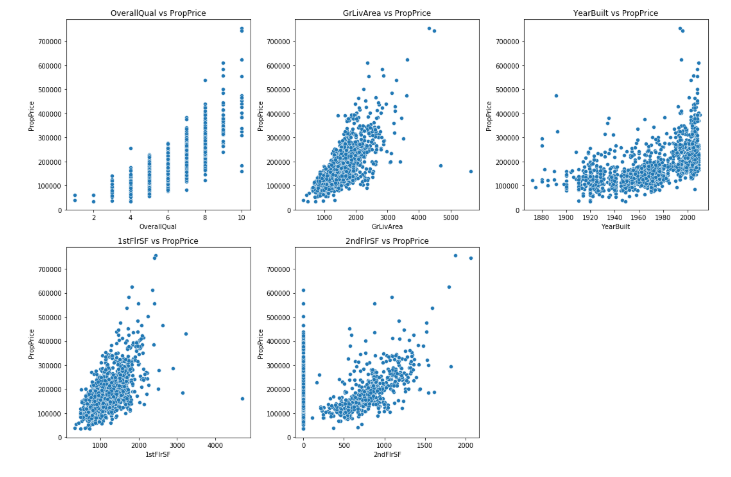


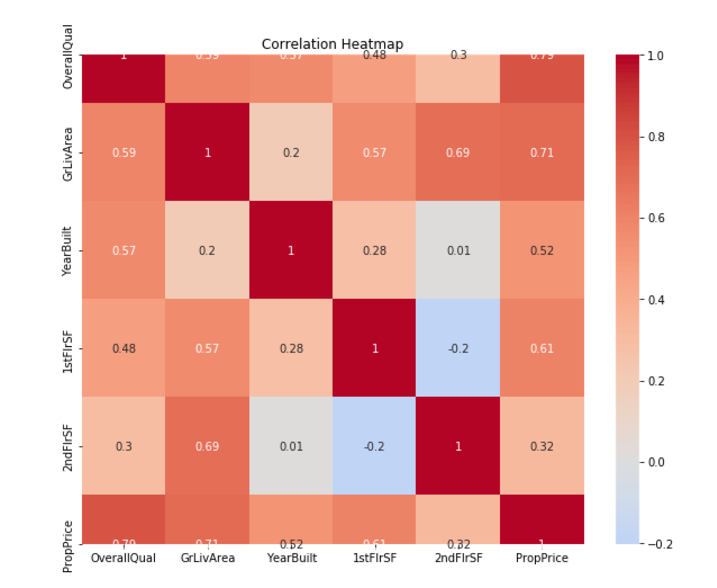
**6. Report on EDA (include pictuers of the graphs)**

**Univariate Analysis**

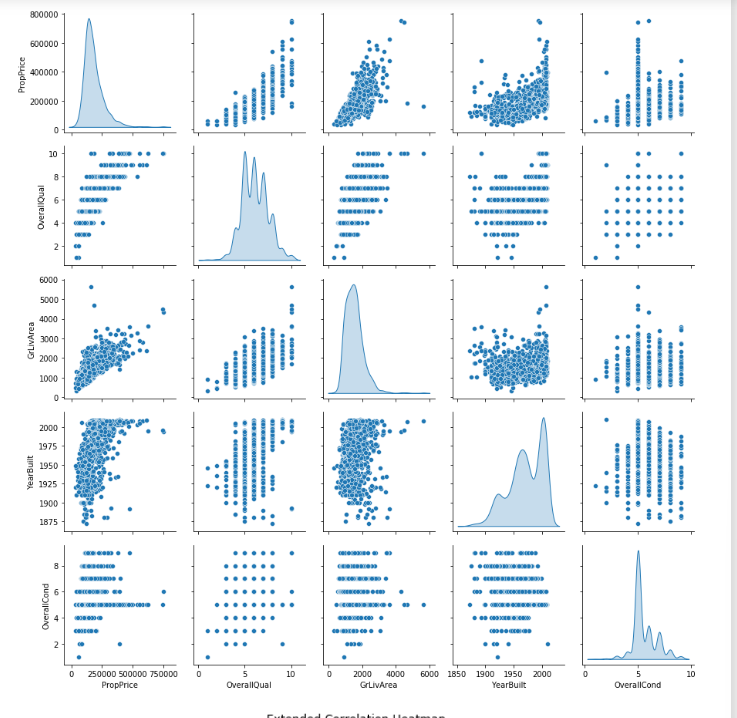
****

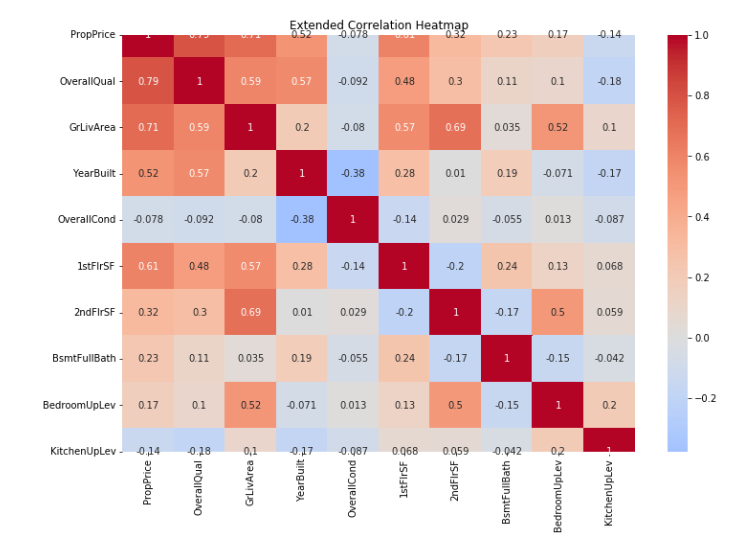
**Bivariate Analysis**

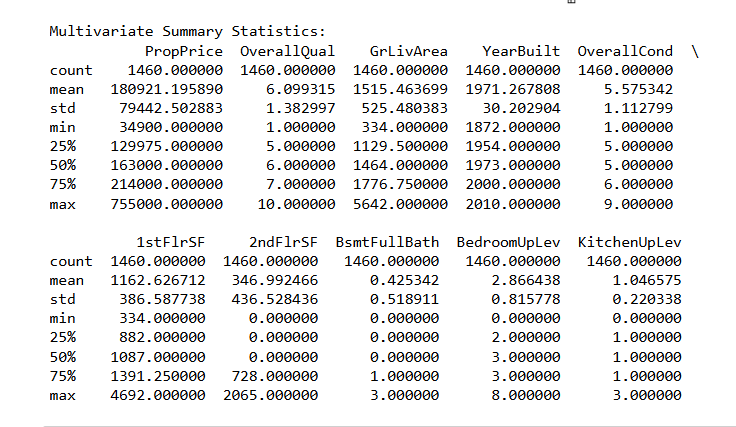
****

****

**Multivariate Analysis**

****

****

****

**Report on EDA - During the univariate analysis I have got some features of the data are showing the positive swing towards through the histplots like 'PropertyPrice','OverallQual','GrLivArea','YearBuilt','1stFlrSF', 2ndFlrSF',Property frontage, Property size. Specially I saw highest frequency of property price between 1 lac and 2 lac dollars. After that I use that feature in bivariate analysis here I saw the relation between each feature to property price through scatter plots and correlation by heat map to prove the above graphs. Then Multivariate analysis show all features again separately and to check extra insights I use some extra features corelation through heatmap like Bamentfullbath,BedroomUpLevel,KitchenUpLevel include in extended correlation heatmap but they are not very important as previous selected features that can play an important role of price prediction .But location wise trends are not showing any trends perfectly.**

**7. Learning Outcomes**

**In this project I have got related specific knowledge, skills, or competencies that i expect to gain or demonstrate through our project. Here are some potential learning outcomes for a project:**

1. **Understanding of data preprocessing like how to clean and how to prepare real estate data for machine learning models.**
2. **EDA give visualizing data and finding patterns or correlation in property price data which will help in understanding trends and information.**
3. **Model selection and Evaluation: here learn about different machine learning algorithms suitable for regression tasks (e.g., Linear Regression, Random Forest, Gradient Boosting) and how to select the best model based on performance metrics (e.g., RMSE, R-squared).**
4. **Hyperparameter tuning: how to optimize model performance through hyperparameter tuning techniques such as grid search .**
5. **Understanding of Overfitting and Underfitting: Now I become familiar with the concepts of overfitting and underfitting, and learn strategies to mitigate these issues (e.g., cross-validation, regularization).**
6. **Interpretability of Models: I learn how to interpret the results of your model to derive insights into the factors affecting property prices, which is crucial for stakeholders who need to understand the model’s predictions.**

**Conclusion : This Project is very helpful for real estate, agent, buyers and sellers to understand Property Business. On the basis of the Machine Learning tools, techniques and prediction they can improve their future property price plans. They can take decision to give hike to property prices or not. So, in this scenario this project is very fruitful for our users.**

9. Citations : Books and websites used for research

Geeks for geeks, YouTube

[www.medium.com](http://www.medium.com)

2. Linear Regression, Decision Trees, Random Fore.
3. **Model Selection and Evaluation**: You will learn about different m88achine learning algorithms suitable for regression tasks (e.g., Linear Regression, Decision Trees, Random Forest, Gradient Boosting) and how to select the best model based on performance metrics (e.g., RMSE, R-squared).

.