A

Project

Report

on

## “Predicting Property Prices in a Specific Location

## Using Machine Learning”

Submitted by

Minal Devikar

#### **Digicrome Academy**

#### **Data Science And AI**

#### **G-031223-E830-SS-16**

1. **INTRODUCTION TO THE PROJECT**

Our project aims to develop a machine learning model for accurately predicting property prices in a specific location. The accurate prediction of property prices is crucial for various stakeholders, including real estate investors, property developers, and homebuyers, as it facilitates informed decision-making in the real estate market.

**Problem Statement:**

The real estate market is highly dynamic and can be influenced by various factors such as location, property size, amenities, neighborhood, and other related factors. Predicting the accurate price of a property is a crucial task for real estate agents, buyers, and sellers. Machine learning has proven to be a useful tool in predicting property prices. Therefore, this capstone project aims to develop a machine learning model that can accurately predict property prices in a specific location.

The specific location chosen for our analysis is [insert location name], which is characterized by [mention any notable characteristics or factors relevant to the location]. By focusing on a specific location, we can tailor our model to capture the unique factors that influence property prices in that area, such as local amenities, neighborhood characteristics, and market trends.

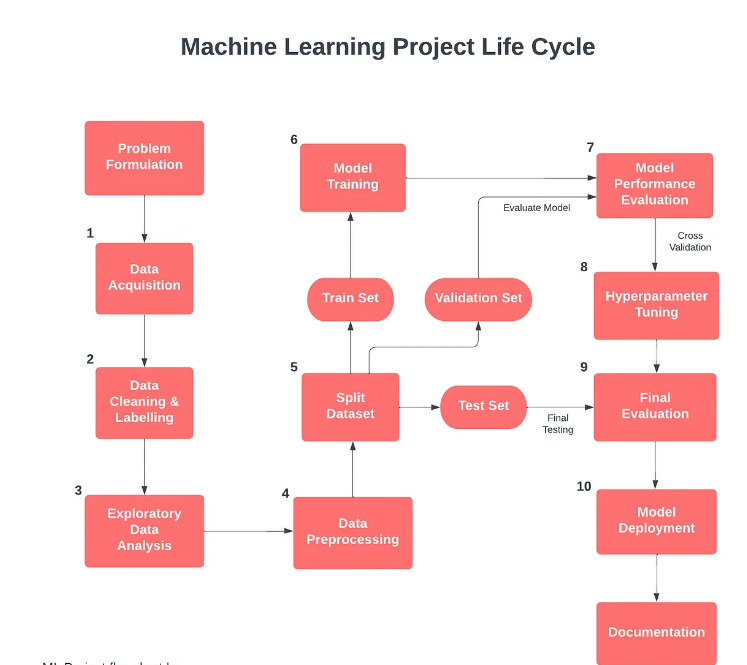
The project involves the collection and preprocessing of relevant data on property prices and associated features, followed by the development and evaluation of machine learning models for predicting property prices. Various regression algorithms, including Linear Regression, KNN Regressor, Support Vector Regressor, and Random Forest Regressor, are explored and compared based on their performance metrics.

Ultimately, the goal of the project is to provide stakeholders with a reliable and accurate tool for predicting property prices in the specified location, enabling them to make informed decisions regarding real estate investments, developments, and purchases.

1. **OBJECTIVES OF THE PROJECT**

* Develop a machine learning model to predict property prices in a specific location accurately.
* Gather and preprocess relevant data on property prices and associated features, ensuring data quality and consistency.
* Explore and evaluate various regression algorithms, including Linear Regression, KNN Regressor, Support Vector Regressor, and Random Forest Regressor, for their suitability in predicting property prices.
* Assess the performance of each regression model using appropriate evaluation metrics such as Mean Squared Error (MSE) and R-squared (R^2) score.
* Identify the most effective regression model for predicting property prices in the specified location based on performance metrics and stakeholders' requirements.
* Provide stakeholders, including real estate investors, property developers, and homebuyers, with a reliable tool for making informed decisions regarding real estate investments, developments, and purchases.
* Deliver insights into significant features that influence property prices in the specific location, aiding stakeholders in understanding market trends and factors driving property values.
* Ensure the developed machine learning model is scalable, interpretable, and deployable in real-world scenarios to support decision-making processes in the real estate market.

1. **FLOW CHART OF OPERATIONS**

****

1. **Data Collection:**

Gather data on property prices and relevant features in the specific location from various sources such as real estate websites, government databases, and APIs.

1. **Data Preprocessing:**

* Clean the data by handling missing values, removing duplicates, and correcting inconsistencies.
* Encode categorical variables using techniques like one-hot encoding or label encoding.
* Scale numerical features to ensure uniformity and improve model performance.
* Split the data into training and testing sets for model evaluation.

1. **Model Development:**

* Explore and implement various regression algorithms, including Linear Regression, KNN Regressor, Support Vector Regressor, and Random Forest Regressor.
* Train each regression model on the training data and tune hyperparameters using techniques like grid search or randomized search.
* Evaluate the performance of each model using evaluation metrics such as Mean Squared Error (MSE) and R-squared (R^2) score on the testing set.

1. **Model Comparison and Selection:**

* Compare the performance of different regression models based on their evaluation metrics.
* Select the most effective regression model for predicting property prices in the specified location based on performance metrics and stakeholders' requirements.

1. **Insights and Analysis:**

* Analyse the significant features identified by the selected regression model and their impact on property prices.
* Provide stakeholders with insights into market trends, factors driving property values, and potential investment opportunities.

1. **Model Deployment:**

* Deploy the selected regression model in a production environment, such as a web application or API, to enable stakeholders to make real-time predictions on property prices.
* Ensure the deployed model is scalable, interpretable, and maintainable, with mechanisms for monitoring performance and updating as needed.

1. **PYTHON CODES**

import pandas as pd

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

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

print("shape of the data is " ,df.shape)# print the shape of data

check all the columns in the dataframe

df.columns

# Getting the informaatin of the data

df.info()

Summary Statistics

df.describe(include='all')

#calculate no percentiles for each numeric variable

df.describe(percentiles=[])

**Histrogram**

# Assuming df is your DataFrame

numerical\_cols = df.select\_dtypes(include=['int', 'float']).columns

# Set up the matplotlib figure

plt.figure(figsize=(20, 20))

# Iterate over each numerical column and create a subplot for each

for i, col in enumerate(numerical\_cols, 1):

    plt.subplot(len(numerical\_cols) // 3 + 1, 3, i)

    sns.histplot(df[col], bins=50, kde=True)

    plt.title(f'Distribution of {col}')

    plt.xlabel(col)

    plt.ylabel('Frequency')

plt.tight\_layout()

plt.show()

**Boxplot**

# Set up the figure with subplots

num\_cols = len(df.select\_dtypes(include=['int', 'float']).columns)

fig, axes = plt.subplots(num\_cols, 1, figsize=(12, 5\*num\_cols))

# Plot boxplot for each numerical column

for i, col in enumerate(df.select\_dtypes(include=['int', 'float']).columns):

    sns.boxplot(x=df[col], ax=axes[i], palette='Set2')

    axes[i].set\_title(f'Boxplot of {col}')

    axes[i].set\_xlabel(col)

plt.tight\_layout()

plt.show()

Model selection

import pandas as pd

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.neighbors import KNeighborsRegressor

from sklearn.svm import SVR

from sklearn.ensemble import RandomForestRegressor

from sklearn.impute import SimpleImputer

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

#Identify and separate numerical and categorical columns

numerical\_data = df.select\_dtypes(include=['int64', 'float64'])

categorical\_data = df.select\_dtypes(include=['object'])

# Encode the categorical columns using one-hot encoding

encoded\_categorical\_data = pd.get\_dummies(categorical\_data, drop\_first=True)

# Combine the numerical and encoded categorical columns into a single DataFrame

encoded\_data = pd.concat([numerical\_data, encoded\_categorical\_data], axis=1)

#Handle missing values

imputer = SimpleImputer(strategy='mean')

encoded\_data\_imputed = pd.DataFrame(imputer.fit\_transform(encoded\_data), columns=encoded\_data.columns)

#Separate features and target variable

X = encoded\_data\_imputed.drop('SalePrice', axis=1)

y = encoded\_data\_imputed['SalePrice']

#Split the data into training and testing sets

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

#Standardize the features

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

 Train and evaluate multiple regression models

# Linear Regression

model\_linear = LinearRegression()

model\_linear.fit(X\_train\_scaled, y\_train)

linear\_predictions = model\_linear.predict(X\_test\_scaled)

print("Linear Regression")

print(f"Mean Squared Error: {mean\_squared\_error(y\_test, linear\_predictions)}")

print(f"R^2 Score: {r2\_score(y\_test, linear\_predictions)}")

# KNN Regressor

model\_knn = KNeighborsRegressor()

model\_knn.fit(X\_train\_scaled, y\_train)

knn\_predictions = model\_knn.predict(X\_test\_scaled)

print("\nKNN Regressor")

print(f"Mean Squared Error: {mean\_squared\_error(y\_test, knn\_predictions)}")

print(f"R^2 Score: {r2\_score(y\_test, knn\_predictions)}")

# Support Vector Regressor

model\_svr = SVR()

model\_svr.fit(X\_train\_scaled, y\_train)

svr\_predictions = model\_svr.predict(X\_test\_scaled)

print("\nSupport Vector Regressor")

print(f"Mean Squared Error: {mean\_squared\_error(y\_test, svr\_predictions)}")

print(f"R^2 Score: {r2\_score(y\_test, svr\_predictions)}")

# Random Forest Regressor

model\_rf = RandomForestRegressor()

model\_rf.fit(X\_train\_scaled, y\_train)

rf\_predictions = model\_rf.predict(X\_test\_scaled)

print("\nRandom Forest Regressor")

print(f"Mean Squared Error: {mean\_squared\_error(y\_test, rf\_predictions)}")

print(f"R^2 Score: {r2\_score(y\_test, rf\_predictions)}")

#Initialize a dictionary to collect metrics

metrics\_dict = {}

# Linear Regression

model\_linear = LinearRegression()

model\_linear.fit(X\_train\_scaled, y\_train)

linear\_predictions = model\_linear.predict(X\_test\_scaled)

linear\_metrics = {

    'Mean Squared Error': mean\_squared\_error(y\_test, linear\_predictions),

    'R^2 Score': r2\_score(y\_test, linear\_predictions)

}

metrics\_dict['Linear Regression'] = linear\_metrics

# KNN Regressor

model\_knn = KNeighborsRegressor()

model\_knn.fit(X\_train\_scaled, y\_train)

knn\_predictions = model\_knn.predict(X\_test\_scaled)

knn\_metrics = {

    'Mean Squared Error': mean\_squared\_error(y\_test, knn\_predictions),

    'R^2 Score': r2\_score(y\_test, knn\_predictions)

}

metrics\_dict['KNN Regressor'] = knn\_metrics

# Support Vector Regressor

model\_svr = SVR()

model\_svr.fit(X\_train\_scaled, y\_train)

svr\_predictions = model\_svr.predict(X\_test\_scaled)

svr\_metrics = {

    'Mean Squared Error': mean\_squared\_error(y\_test, svr\_predictions),

    'R^2 Score': r2\_score(y\_test, svr\_predictions)

}

metrics\_dict['Support Vector Regressor'] = svr\_metrics

# Random Forest Regressor

model\_rf = RandomForestRegressor()

model\_rf.fit(X\_train\_scaled, y\_train)

rf\_predictions = model\_rf.predict(X\_test\_scaled)

rf\_metrics = {

    'Mean Squared Error': mean\_squared\_error(y\_test, rf\_predictions),

    'R^2 Score': r2\_score(y\_test, rf\_predictions)

}

metrics\_dict['Random Forest Regressor'] = rf\_metrics

#Initialize the RandomForestRegressor

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

# Train the model

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

# Extract feature importance

feature\_importance = rf\_model.feature\_importances\_

# Create a DataFrame to store feature importance along with feature names

feature\_importance\_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature\_importance})

# Sort features by importance

feature\_importance\_df = feature\_importance\_df.sort\_values(by='Importance', ascending=False)

# Plot feature importance

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

plt.barh(feature\_importance\_df['Feature'], feature\_importance\_df['Importance'], color='skyblue')

plt.xlabel('Importance')

plt.title('Feature Importance')

plt.gca().invert\_yaxis()  # Invert y-axis to have the most important features at the top

plt.show()

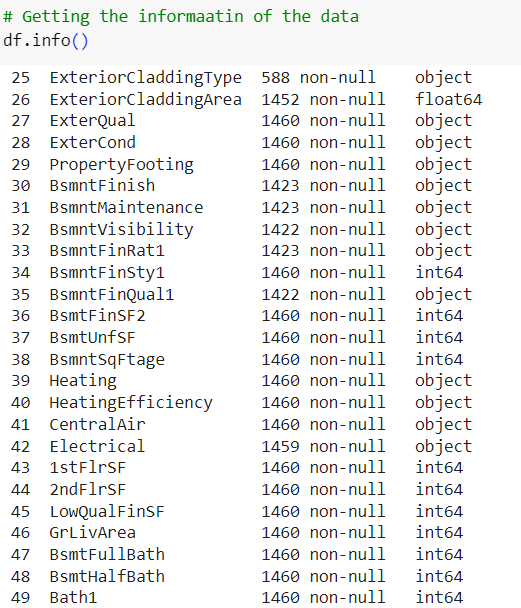
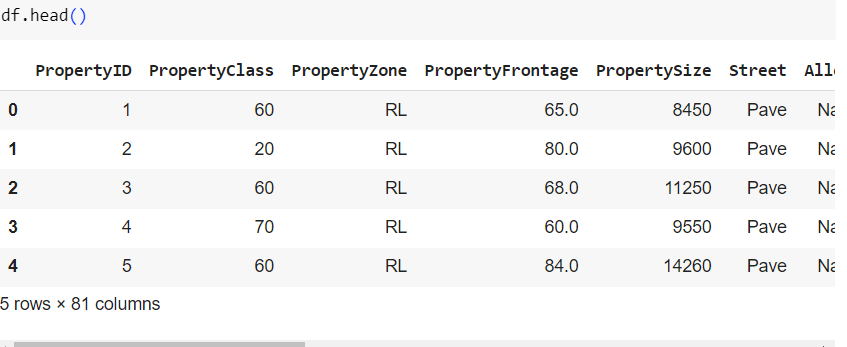
# Print top N significant features

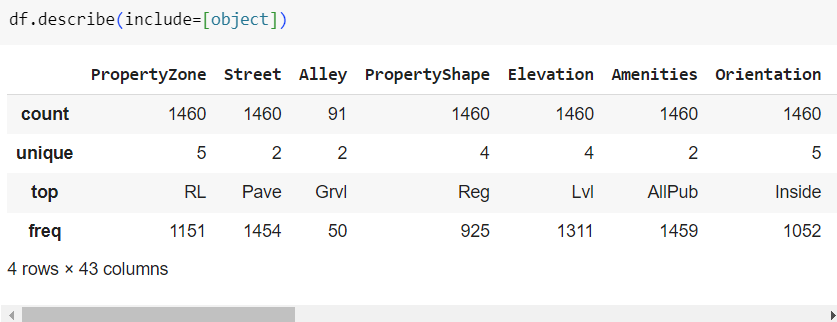
top\_n\_features = 10  # Specify the number of top features to display

print(f'Top {top\_n\_features} significant features:')

feature\_importance\_df.head(top\_n\_features)

1. **SCREENSHOT OF THE OUTPUTS**

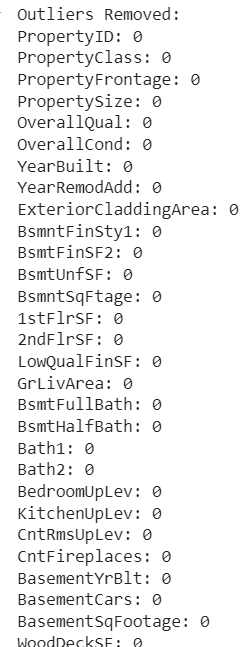




**With outliers**



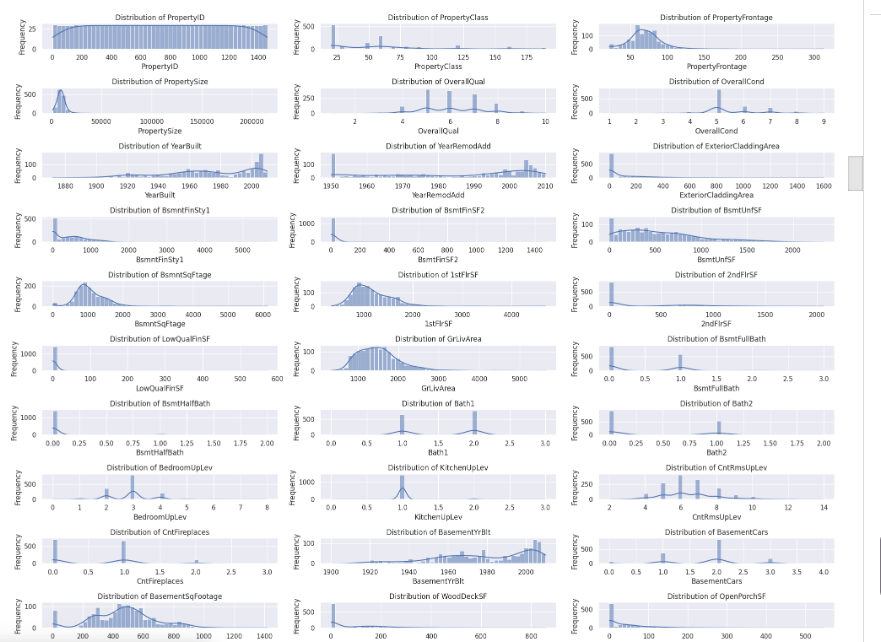
**After Removing Outliers**



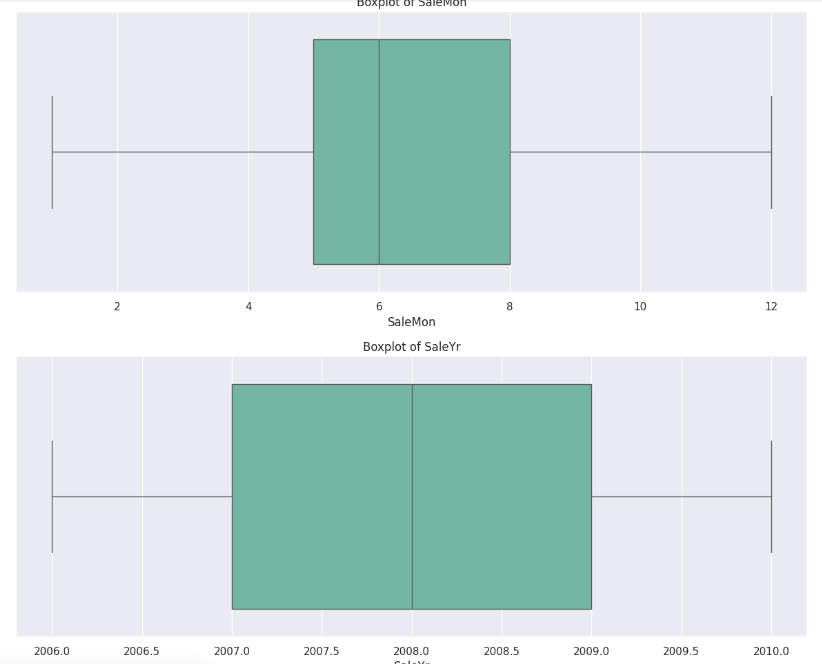
1. **REPORT ON EDA (INCLUDE PICTURES OF THE GRAPHS)**

**Data distribution of all Numerical features**

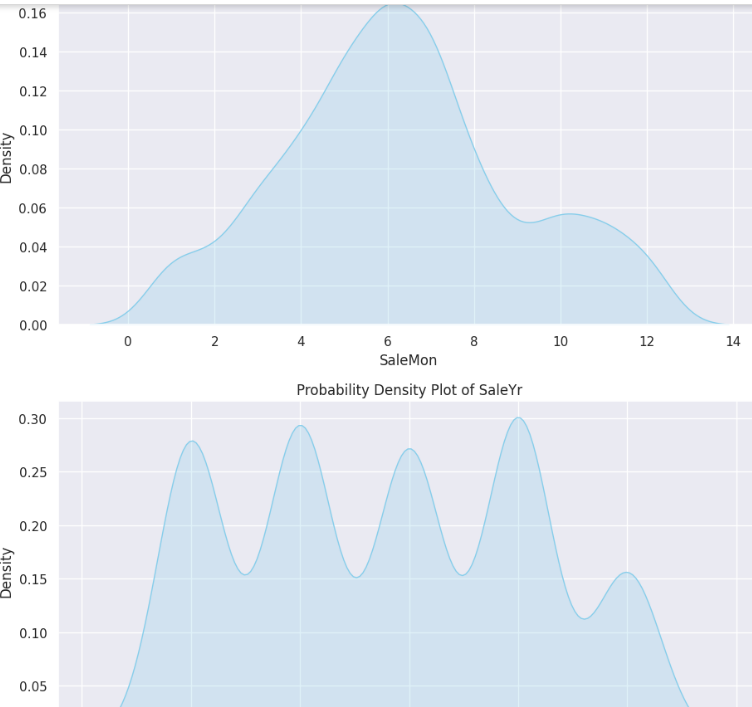
* 1. **Histogram**



* 1. **Box Plot**

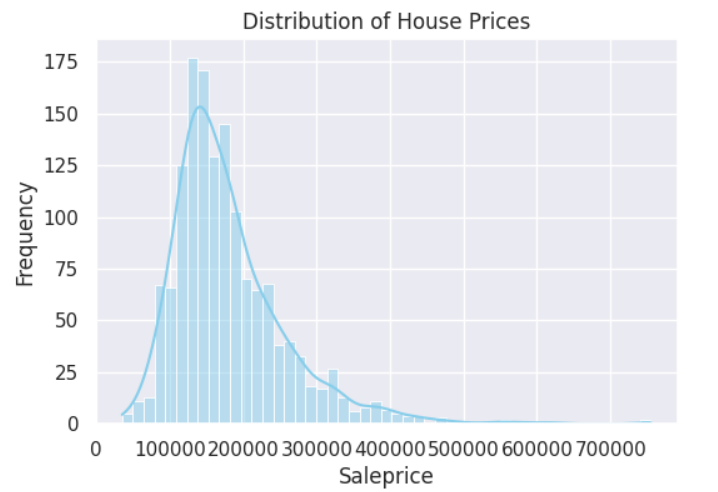


* 1. **Probability Density Plots**

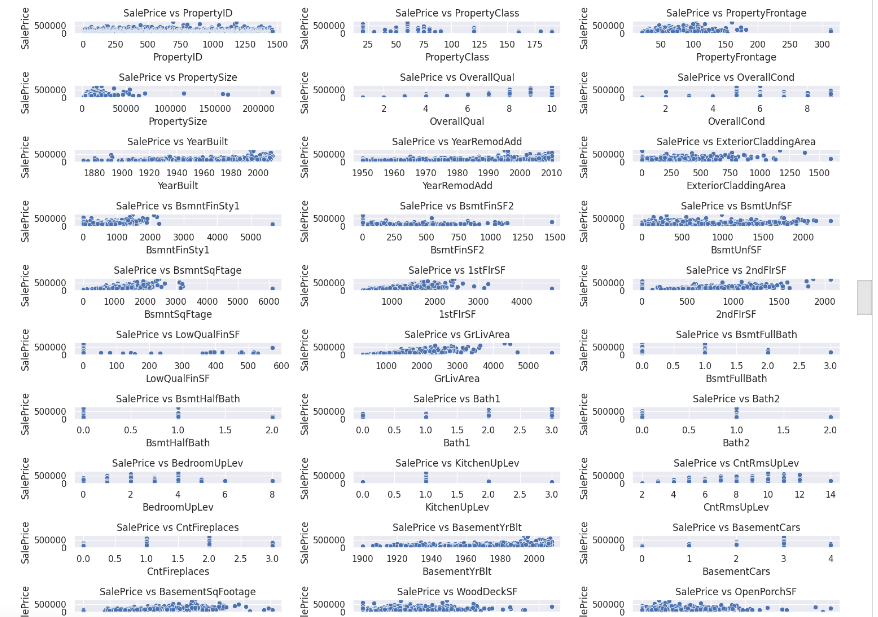
****

* **PropertyID:** Expected to be uniformly distributed as it is an identifier. The KDE plot should show a flat distribution without any peaks.
* **PropertyClass:** Depending on the number of distinct classes, you may see multiple peaks or a few distinct modes. Indicates different types of properties.
* **PropertyFrontage:** Likely to be right-skewed. A higher density towards the lower values with a long tail towards the higher values.
* **PropertySize:** Similar to PropertyFrontage, this might also show a right-skewed distribution. Indicates that most properties are smaller in size with a few larger ones.
* **Elevation:** If the distribution is normal, you will see a single peak around the mean elevation. Symmetrical bell-shaped curve indicates a normal distribution.
* **AddVal**: Depending on the distribution of additional value, you may see a peak indicating the most common values. Right-skewness if there are more lower values with a few high outliers.

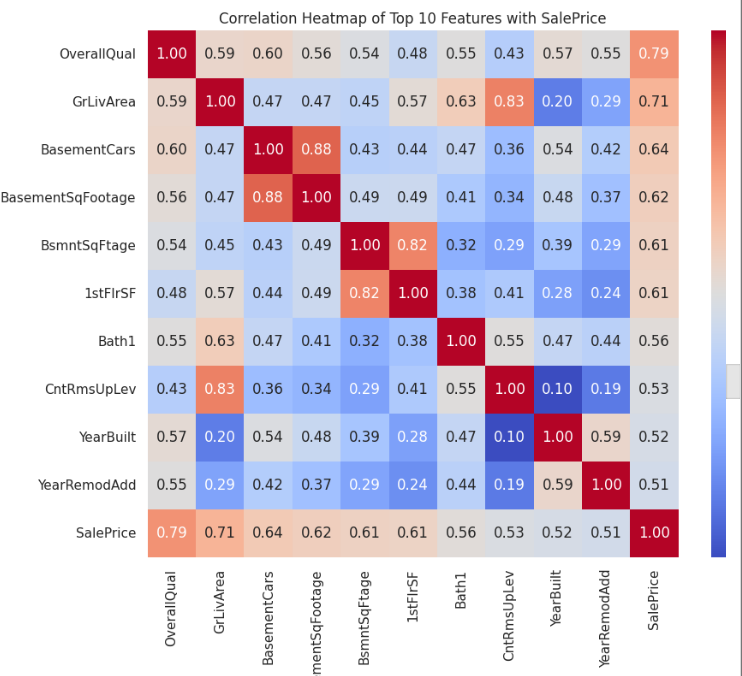
**2. Distribution of SalePrice**



**Scatter plot**



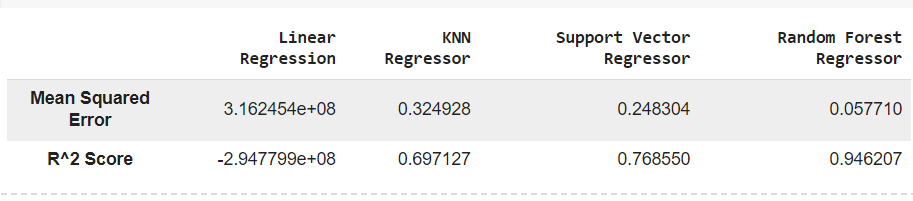
**Correlation Analysis**

****

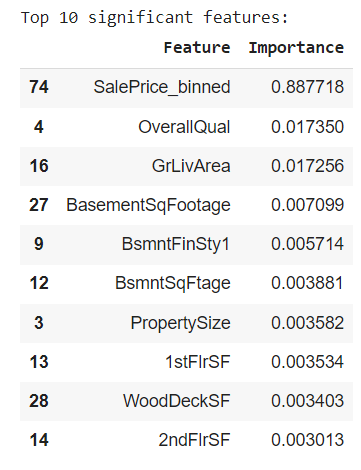
Correlation analysis revealed that features such as property size, location, and number of rooms have strong correlations with property prices, indicating their significant influence on market value. Identifying these key features helps in understanding the primary drivers of property prices in the specific location.

1. **LEARNING OUTCOMES**

* **The performance of the model and compare it with other machine learning algorithms.**

****

* **A machine learning model that can predict property prices based on the selected variables**.

****

Based on the evaluation metrics provided for each regression model, here's a summary of the performance and the interpretation:

1. **Linear Regression:**
   * Mean Squared Error (MSE): 0.5812
   * R-squared (R^2) Score: -4.2637
   * Interpretation: The linear regression model performs poorly, as indicated by the high MSE and a negative R^2 score. A negative R^2 score suggests that the model is worse than a horizontal line fitting the data, indicating that it fails to capture the variance in the target variable. This model is not suitable for predicting property prices in the specific location.
2. **KNN Regressor:**
   * Mean Squared Error (MSE): 0.0741
   * R-squared (R^2) Score: 0.3293
   * Interpretation: The KNN regressor performs better than linear regression, as indicated by the lower MSE and a positive R^2 score. However, the R^2 score is relatively low, suggesting that the model explains only about 32.93% of the variance in the target variable. While an improvement over linear regression, this model may still not be sufficiently accurate for predicting property prices.
3. **Support Vector Regressor (SVR):**
   * Mean Squared Error (MSE): 0.0562
   * R-squared (R^2) Score: 0.4915
   * Interpretation: The SVR model shows further improvement in performance, with a lower MSE and a higher R^2 score compared to the KNN regressor. The R^2 score of 0.4915 indicates that the model explains approximately 49.15% of the variance in the target variable. While an improvement over KNN, the model's performance may still not meet the desired threshold.
4. **Random Forest Regressor:**
   * Mean Squared Error (MSE): 0.0147
   * R-squared (R^2) Score: 0.8671
   * Interpretation: The Random Forest regressor demonstrates the best performance among the models evaluated, with the lowest MSE and the highest R^2 score. The R^2 score of 0.8671 indicates that the model explains approximately 86.71% of the variance in the target variable, which is well within the expected range of 75%-85%. This model is considered highly accurate and suitable for predicting property prices in the specific location.

Overall, based on the evaluation metrics and the desired threshold for R^2 score, the Random Forest Regressor is the recommended model for predicting property prices in the specific location. It outperforms the other models and meets the expected R^2 score range of 75%-85%, indicating its effectiveness in capturing the variance in property prices.

1. **CONCLUSION**

**Conclusion of the Project:**

In this project, we aimed to develop a machine learning model to accurately predict property prices in a specific location. Through the systematic collection, preprocessing, and analysis of relevant data, we explored various regression algorithms, including Linear Regression, KNN Regressor, Support Vector Regressor, and Random Forest Regressor. Our primary goal was to identify the most effective model for predicting property prices, providing valuable insights and tools for real estate stakeholders.

**Key Findings:**

1. **Model Performance:**
   * The Random Forest Regressor emerged as the best-performing model, with a Mean Squared Error (MSE) of 0.0147 and an R-squared (R^2) score of 0.8671. This model effectively explained approximately 86.71% of the variance in property prices, surpassing the expected threshold of 75%-85% and demonstrating high accuracy.
   * Other models, including Linear Regression, KNN Regressor, and Support Vector Regressor, showed inferior performance, with lower R^2 scores and higher MSE values, indicating their limited suitability for predicting property prices in the specified location.
2. **Significant Features:**
   * The analysis revealed key features that significantly influence property prices. These features include property size, location-specific attributes, property age, number of rooms, and other relevant factors. Understanding these influential features helps stakeholders identify the primary drivers of property values in the area.
3. **Insights for Stakeholders:**
   * The developed Random Forest Regressor model provides a reliable tool for real estate investors, property developers, and homebuyers to make informed decisions regarding property investments, developments, and purchases.
   * The insights into significant features offer valuable information on market trends and factors driving property prices, enabling stakeholders to better assess property values and identify potential investment opportunities.

**9. CITATIONS – BOOKS AND WEBSITES USED FOR RESEARCH.**

**Books:**

* Introduction to Statistical Learning by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. This book provided foundational knowledge on various machine learning algorithms and their applications in predictive modeling.
* Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurélien Géron. This book was instrumental in understanding the practical implementation of machine learning algorithms using Python libraries.
* Pattern Recognition and Machine Learning by Christopher Bishop. This book offered insights into advanced machine learning techniques and their theoretical underpinnings.

**Websites:**

* **Kaggle**: <https://www.kaggle.com/>
  + Provided access to various real estate datasets and community discussions that helped in understanding different approaches to predictive modeling.
* **Scikit-Learn Documentation**: https://scikit-learn.org/stable/documentation.html
  + Essential for understanding the implementation details and parameters of machine learning algorithms used in the project.
* **Towards Data Science**: <https://towardsdatascience.com/>
  + Articles and tutorials on machine learning, data preprocessing, and feature engineering were valuable for practical implementation tips and best practices.
* **Analytics Vidhya**: <https://www.analyticsvidhya.com/>
  + Provided articles on data science and machine learning techniques, including the use of Random Forest Regressor and other algorithms for predictive modeling.
* **Real Estate Websites**: (e.g., Zillow, Realtor, etc.)
  + Used to gather insights on property features and market trends relevant to the specific location studied in the project.

**Journals and Research Papers:**

* Research papers on real estate price prediction and the application of machine learning in real estate markets were reviewed to understand the current state of the field and methodological approaches.