**COLLEGE CODE: 6208**

**NAME : MEENATCHI K**

**IBM REG NO : au620821106055**

**PROJECT NAME: PREDICTION HOUSE PRICE USING MACHINE LEARNING**

**Project Title**: House Price Predictor

**Phase 4**: Development Part 2

**Topic :** Continue building the house price prediction model by feature engineering, model training, and evaluation.



**Overview of the process**:

The following is an overview of the process of building a house price prediction model by feature selection, model training, and evaluation:

**Prepare the data**

This includes cleaning the data, removing outliers, and handling missing values.

**Perform feature selection:**

This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.

**Train the model:**

There are many different machine learning algorithms that can be used for house price prediction. Some popular choices include linear regression, random forests, and gradient boosting machines.

**Evaluate the model:**

This can be done by calculating the mean squared error (MSE) or the root mean squared error (RMSE) of the model's predictions on the held-out test set.

**Deploy the model:**

Once the model has been evaluated and found to be performing well, it can be deployed to production so that it can be used to predict the house prices of new houses.

**PROCEDURE:**

**Feature selection:**

**Identify the target variable:**

This is the variable that you want to predict, such as house price.

**Explore the data:**

This will help you to understand the relationships between the different features and the target variable. You can use data visualization and correlation analysis to identify featuresthat are highly correlated with the target variable.

**Remove redundant features:**

If two features are highly correlated with each other, then you can remove one of the features, as they are likely to contain redundant information.

**Remove irrelevant features:**

If a feature is not correlated with the target variable, then you can remove it, as it is unlikely to be useful for prediction.

**Feature Selection:**

We are selecting numerical features which have more than 0.50 or less than -0.50 correlation rate based on Pearson Correlation Method—which is the default value of parameter "method" in corr() function. As for selecting

categorical features, I selected the categorical values which I believe have significant effect on the target variable such as Heating and MSZoning.

In [1]:

important\_num\_cols=list(df.corr()["SalePrice"][(df.corr()["SalePrice"]>0.5 0) | (df.corr()["SalePrice"]<-0.50)].index)

cat\_cols=["MSZoning","Utilities","BldgType","Heating","KitchenQual"," SaleCondition","LandSlope"]

important\_cols = important\_num\_cols + cat\_cols

df = df[important\_cols]

Checking for the missing values

In [2]:

print("Missing Values by Column")

print("-"\*30)

print(df.isna().sum())

print("-"\*30)

print("TOTAL MISSING VALUES:",df.isna().sum().sum())

Missing Values by Column

------------------------------

OverallQual 0

YearBuilt 0

YearRemodAdd 0

TotalBsmtSF 0

1stFlrSF 0

GrLivArea 0

FullBath 0

TotRmsAbvGrd 0

GarageCars 0

GarageArea 0

SalePrice 0

MSZoning 0

Utilities 0

BldgType 0

Heating 0

KitchenQual 0

SaleCondition 0

LandSlope 0

dtype: int64

------------------------------

TOTAL MISSING VALUES: 0

**Model training:**

**Choose a machine learning algorithm.** There are a number of different machine learning algorithms that can be used for house price prediction, such as linear regression, ridge regression, lasso regression, decision trees, and random forests are Covered above.

Machine Learning Models:

In [3]:

Models=pd.DataFrame(columns=["Model","MAE","MSE","RMSE","R2 S core","RMSE (Cross-Validation)"])

**Linear Regression:**

In [4]:

lin\_reg = LinearRegression()

lin\_reg.fit(X\_train, y\_train)

predictions = lin\_reg.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(lin\_reg)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "LinearRegression","MAE": mae, "MSE": mse, "RM SE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross \_val}models =models.append(new\_ignore\_index=True)

Out[4]:

MAE: 23567.890565943395

MSE: 1414931404.6297863

RMSE: 37615.57396384889

R2 Score: 0.8155317822983865

------------------------------

RMSE Cross-Validation: 36326.451444669496

**Ridge Regression:**

In [5]:

ridge=Ridge()ridge.fit(X\_train,y\_train)predictions=ridge.predict

(X\_test) mae, mse, rmse, r\_squared =evaluation(y\_test,predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(ridge)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "Ridge","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)":

rmse\_cross\_val}model s = models.append(new\_row, ignore\_index=True)

Out[5]:

MAE: 23435.50371200822

MSE: 1404264216.8595588

RMSE: 37473.513537691644

R2 Score: 0.8169224907874508

------------------------------

RMSE Cross-Validation: 35887.852791598336

**Lasso Regression:**

In [6]:

lasso = Lasso()lasso.fit(X\_train, y\_train)prediction

lasso.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(lasso)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "Lasso","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}model s = models.append(new\_row, ignore\_index=True)

Out[6]:

MAE: 23560.45808027236

MSE: 1414337628.502095

RMSE: 37607.680445649596

R2 Score: 0.815609194407292

------------------------------

RMSE Cross-Validation: 35922.76936876075

**Elastic Net:**

In [7]:

elastic\_net = ElasticNet()elastic\_net.fit(X\_train, y\_train)predictions = elasti c\_net.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(elastic\_net)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "ElasticNet","MAE": mae, "MSE": mse, "RMSE": r mse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val} models = models.append(new\_row, ignore\_index=True)

Out[7]:

MAE: 23792.743784996732

MSE: 1718445790.1371393

RMSE: 41454.14080809225

R2 Score: 0.775961837382229

------------------------------

RMSE Cross-Validation: 38449.00864609558

**Support Vector Machines:**

In [8]:

svr = SVR(C=100000)svr.fit(X\_train, y\_train)predictions = svr.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(svr)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "SVR","MAE": mae, "MSE": mse, "RMSE": rmse, " R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

Out[9]:

MAE: 17843.16228084976

MSE: 1132136370.3413317

RMSE: 33647.234215330864

R2 Score: 0.852400492526574

------------------------------

RMSE Cross-Validation: 30745.475239075837

**Random Forest Regressor:**

In [9]:

random\_forest=RandomForestRegressor(n\_estimators=100)random\_forest.fit(X\_train,y\_train)prediction=random\_forest.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(random\_forest)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "RandomForestRegressor","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (CrossValidation)”:rmse\_cross\_val}modelsmodels.append(new\_row, ignore\_index=True)

Out[9]:

MAE: 18115.11067351598

MSE: 1004422414.0219476

RMSE: 31692.623968708358

R2 Score: 0.869050886899595

------------------------------

RMSE Cross-Validation: 31138.863315259332

**XGBoost Regressor:**

In [10]:

xgb = XGBRegressor(n\_estimators=1000,learning\_rate=0.01)xgb.fit(X\_train, y\_train)predictions = xgb.predict(X\_test)

mae, mse, rmse, r\_squared = evaluation(y\_test, predictions)

print("MAE:", mae)

print("MSE:", mse)

print("RMSE:", rmse)

print("R2 Score:", r\_squared)

print("-"\*30)rmse\_cross\_val = rmse\_cv(xgb)

print("RMSE Cross-Validation:", rmse\_cross\_val)

new\_row = {"Model": "XGBRegressor","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r\_squared, "RMSE (Cross-Validation)": rmse\_cross\_val}models = models.append(new\_row, ignore\_index=True)

Out[10]:

MAE: 17439.918396832192

MSE: 716579004.5214689

RMSE: 26768.993341578403

R2 Score: 0.9065777666861116

------------------------------

RMSE Cross-Validation: 29698.84961808251

**Model training:**

* Model training is the process of teaching a machine learning model to predict house prices. It involves feeding the model historical data on house prices and features, such as square footage, number of bedrooms, and location. The model then learns the relationships between these features and house prices.
* Once the model is trained, it can be used to predict house prices for new data. For example, you could use the model to predict the price of a house that you are interested in buying.

**Prepare the data:**

This involves cleaning the data, removing any errors or inconsistencies, and transforming the data into a format that is compatible with the machine learning algorithm that you will be using.

**Split the data into training and test sets:**

The training set will be used to train the model, and the test set will be used to evaluate the performance of the model on unseen data.

**Choose a machine learning algorithm:**

There are a number of different machine learning algorithms that can be used for house price prediction, such as linear regression, ridge regression, lasso regression, decision trees, and random forests.

**Tune the hyperparameters of the algorithm:**

The hyperparameters of a machine learning algorithm are parameters that control the learning process. It is important to tune the hyperparameters of the algorithm to optimize its performance.

**Train the model on the training set:**

This involves feeding the training data to the model and allowing it to learn the relationships between the features and house prices.

**Evaluate the model on the test set:**

This involves feeding the test data to the model and measuring how well it predicts the house prices.

**Model evaluation:**

* Model evaluation is the process of assessing the performance of a machine learning model on unseen data. This is important to ensure that the model will generalize well to new data.
* There are a number of different metrics that can be used to evaluate the performance of a house price prediction model. Some of the most common metrics include:
* **Mean squared error (MSE):** This metric measures the average squared difference between the predicted and actual house prices.
* **Root mean squared error (RMSE):** This metric is the square root of the MSE.
* **Mean absolute error (MAE**): This metric measures the average absolute difference between the predicted and actual house prices.
* **R-squared:** This metric measures how well the model explains the variation in the actual house price.
* In addition to these metrics, it is also important to consider the following factors when evaluating a house price prediction model:
* **Bias:** Bias is the tendency of a model to consistently over- or underestimate house prices.
* **Variance**: Variance is the measure of how much the predictions of a model vary around the true house prices.
* **Interpretability**: Interpretability is the ability to understand how the model makes its predictions. This is important for house price prediction models, as it allows users to understand the factors that influence the predicted house prices.

**Model Comparison:**

🡪(The less the Root Mean Squared Error (RMSE), The better the model).Model compression refers to the process of reducing the size of a machine learning model while maintaining or minimizing its performance.

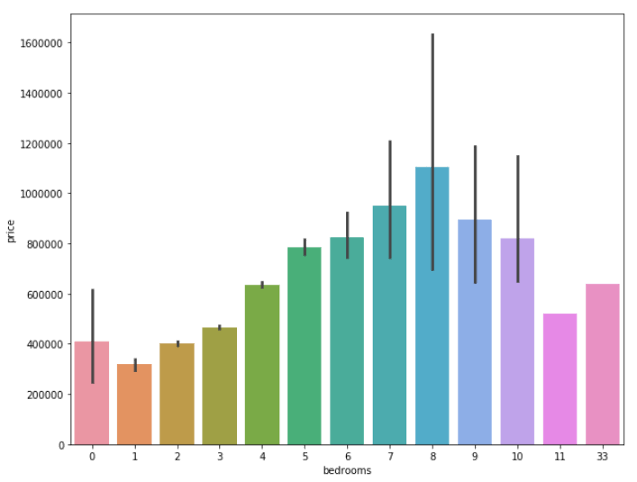
🡪 This can be important for various reasons, including deploying models on resource-constrained devices, reducing memory and storage requirements, and improving model inference speed. Here are some techniques for model compression:

**Quantization:** Convert model weights and activations from high-precision floating-point numbers to lower-precision integers. For example, you can use quantization-aware training to train models for inference with reduced precision

**Low-Rank Factorization:** Decompose weight matrices into low-rank factors, reducing the number of parameters and the computational cost of inference.

**Model Quantization:** Reduce the number of unique operations in a model’s architecture, which can help optimize inference on hardware accelerators.

**Model Quantization with Compression Libraries:** Use specialized libraries like TensorFlow Lite or ONNX Runtime to quantize and compress models for deployment on mobile or edge devices.



**Feature Engineering:**

Feature engineering is a crucial aspect of building a house price prediction model using machine learning. It involves creating new features, transforming existing ones, and selecting the most relevant variables to improve the model's predictive power. Here are some feature engineering ideas for house price prediction:

**1.Total Area Features:**

Combine individual room areas to create features like "Total Living Area," "Total Bedroom Area," or "Total Bathroom Area." These can be significant predictors of house price.

**2.Ratio Features:**

Create features that represent ratios, such as the "Bedroom to Bathroom Ratio" or "Living Area to Lot Area Ratio." These ratios may capture the property's layout and functionality.

**3.Age of the Property:**

Calculate the age of the property by subtracting the construction year from the current year. Newer properties might have higher values.

**4.Neighborhood Statistics:**

Aggregate neighborhood-level statistics, such as the average income, crime rate, school ratings, or proximity to amenities, and use these as features.

**5.Distance to Key Locations:**

Calculate distances from the property to essential places like schools, parks, shopping centers, or public transportation hubs. Closer proximity to such amenities can affect the price.

**6.Categorical Encodings:**

Use techniques like one-hot encoding, label encoding, or target encoding for categorical variables, such as property type, heating system, or garage type.

**7.Seasonal Features:**

Create features indicating the season during which the house was sold. Seasonality can influence property demand and prices.

**8.Historical Data:**

Incorporate historical data on house prices and local real estate market trends. This can help the model account for cyclical patterns.

**9.Exterior Features:**

Develop features related to the property's exterior, such as the presence of a swimming pool, patio, or garden. These features can be valuable for determining a property's appeal.

**10.Quality Scores:**

Create a combined quality score by aggregating the quality ratings of various components of the property, such as kitchen quality, bathroom quality, and overall house quality.

**11.Logarithmic Transformations**:

Apply logarithmic transformations to features like "Lot Area" or "Number of Bedrooms" to make their distributions more normal.

**12.Interaction Features:**

Create interaction terms by multiplying or dividing relevant features. For example, "Number of Bathrooms" multiplied by "Total Living Area" can represent the total bathroom area.

**13.Missing Value Indicators:**

Create binary indicators for missing values in the dataset. The presence of missing data can be an informative feature.

**Various feature to perform model training:**



**Use a variety of feature engineering techniques:**

Feature engineering is the process of transforming raw data into features that are more informative and predictive for machine learning models. By using a variety of feature engineering techniques, you can create a set of features that will help your model to predict house prices more accurately.

**Use cross-validation:**

Cross-validation is a technique for evaluating the performance of a machine learning model on unseen data. It is important to use crossvalidation to evaluate the performance of your model during the training process. This will help you to avoid overfitting and to ensure that your model will generalize well to new data.

**Use ensemble methods:**

Ensemble methods are machine learning methods that combine the predictions of multiple models to produce a more accurate prediction. Ensemble methods can often achieve better performance than individual machine learning models.

**Use a holdout test set:**

A holdout test set is a set of data that is not used to train or evaluate the model during the training process. This data is used to evaluate the performance of the model on unseen data after the training process is complete.

**Compare the model to a baseline:**

A baseline is a simple model that is used to compare the performance of your model to. For example, you could use the mean house price as a baseline.

**Analyze the model's predictions:**

Once you have evaluated the performance of the model, you can analyze the model's predictions to identify any patterns or biases. This will help you to understand the strengths and weaknesses of the model and to improve it.

**Conclusion:**

* In the quest to build an accurate and reliable house price prediction model, we have embarked on a journey that encompasses critical phases, from feature selection to model training and evaluation. Each of these stages plays indispensable role in crafting a model that can provide meaningful insights and estimates for one of the most significant financial decisions individuals and businesses make—real estate transactions.
* Finally, model evaluation is the litmus test for our predictive prowess. Using metrics like Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, and R-squared, we've quantified the model's performance. This phase provides us with the confidence to trust the

model's predictions and assess its ability to adapt to unseen data.

* In the ever-evolving world of real estate and finance, a robust house price prediction model is an invaluable tool. It aids buyers, sellers, and investors in making informed decisions, mitigating risks, and seizing opportunities. As more data becomes available and market dynamics change, the model can be retrained and refined to maintain its accuracy.