

House-Price- Prediction Using Machine Learning



House-Price- Prediction Using Machine Learning

A PROJECT REPORT

Submitted by

GUNASEKARAN M (812921104008)

LOGESWARAN O (812921104015)

THIRISHA E (812921104047)

THULASI P (812921104048)

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE AND ENGINEERING

Abstract

Predicting house prices is a crucial task in the real estate industry, offering valuable insights for buyers, sellers, and investors.

Machine learning techniques have become a powerful tool for this purpose. This abstract provides an overview of a typical approach to house price prediction using machine learning.

The process begins with data collection, where historical housing prices and relevant property features are gathered. Data preprocessing follows, involving cleaning, encoding, and normalization. Feature selection and engineering help in improving the model's accuracy. The dataset is then divided into training and testing sets, and an appropriate regression algorithm is selected for model training.

Keywords:-

Python , Computer Vision , Neural Networks , Deep Learning , Image , Price Detection , House-Price-Prediction

TABLE OF CONTENT

CHAPTER NO	TITLE
	Abstract
1	Introduction
	Data Processing, Data Collection, Data Cleaning
	Feature Selection and Engineering, Data Transformation
	Data Splitting, Data Visualization, Data Quality Check
	Save Processed Data
2	Flow chart
3	PROGRAM
4	OUTPUT
5	Conclusion

Abstract :

Predicting house prices is a crucial task in the real estate industry, offering valuable insights for buyers, sellers, and investors. Machine learning techniques have become a powerful tool for this purpose. This abstract provides an overview of a typical approach to house price prediction using machine learning.

The process begins with data collection, where historical housing prices and relevant property features are gathered. Data preprocessing follows, involving cleaning, encoding, and normalization. Feature selection and engineering help in improving the model's accuracy. The dataset is then divided into training and testing sets, and an appropriate regression algorithm is selected for model training.

Evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean

Squared Error (RMSE) are employed to assess the model's performance. Hyperparameter tuning is conducted to optimize the model, and once satisfactory results are achieved, the model is deployed in a production environment.

Continual monitoring and maintenance are crucial, as house prices can change over time. Ethical considerations and compliance with regulations, if applicable, are essential throughout the process. This abstract highlights the key steps and considerations involved in house price prediction using machine learning, emphasizing the significance of data quality, model selection, and ongoing model management for accurate and reliable predictions in the dynamic real estate market.

Introduction :

The prediction of house prices is a fundamental and practical challenge within the real estate industry. Accurate price estimations are vital for buyers, sellers, and investors to make informed decisions. Traditionally, real estate professionals rely on market knowledge and expertise to gauge property values. However, in the era of big data and advanced technology, machine learning has emerged as a powerful tool for predicting house prices with a high degree of precision.

This introduction sets the stage for the discussion of house price prediction using machine learning. It outlines the significance of the task, the role of data, and the promise of machine learning in providing robust, data-driven insights.

Data Processing :

Data processing is a critical step in the house price prediction using machine learning. It involves several tasks aimed at preparing the raw data for model training and analysis. Here are the key data processing steps.

Data Collection :

Gather a comprehensive dataset that includes historical house prices and relevant features. Sources may include real estate websites, government records, or data providers.

Data Cleaning :

Address missing values: Identify and handle missing data points in the dataset. Options include imputation, deletion, or using domain-specific knowledge to fill in missing values.

Handle outliers: Detect and manage outliers in the data. Outliers can significantly affect model performance, so you may choose to remove them or transform them.

Feature Selection and Engineering :

Feature selection: Identify and select the most relevant features that are likely to impact house prices. This step reduces dimensionality and enhances model efficiency.

Feature engineering: Create new features that may improve the model's performance. For example, you could calculate the total area of a property by combining the dimensions of its rooms.

Data Transformation :

Categorical data encoding: Convert categorical variables (e.g., property type, location) into a numerical format.

Common techniques include one-hot encoding or label encoding.

Scaling and normalization: Rescale numerical features to bring them to a common scale (e.g., between 0 and 1) to ensure that no feature dominates others during modeling.

Logarithmic transformation: Apply logarithmic transformations to features with skewed distributions to make them more normally distributed.

Data Splitting :

Divide the preprocessed dataset into training and testing sets. A common split ratio is 80% for training and 20% for testing. This separation ensures that the model is evaluated on unseen data to assess its generalization capability.

Data Visualization (Optional but helpful) :

Create visualizations such as histograms, scatter plots, or correlation matrices to gain insights into the data and relationships between variables. This step can be particularly valuable for understanding feature importance.

Data Quality Check :

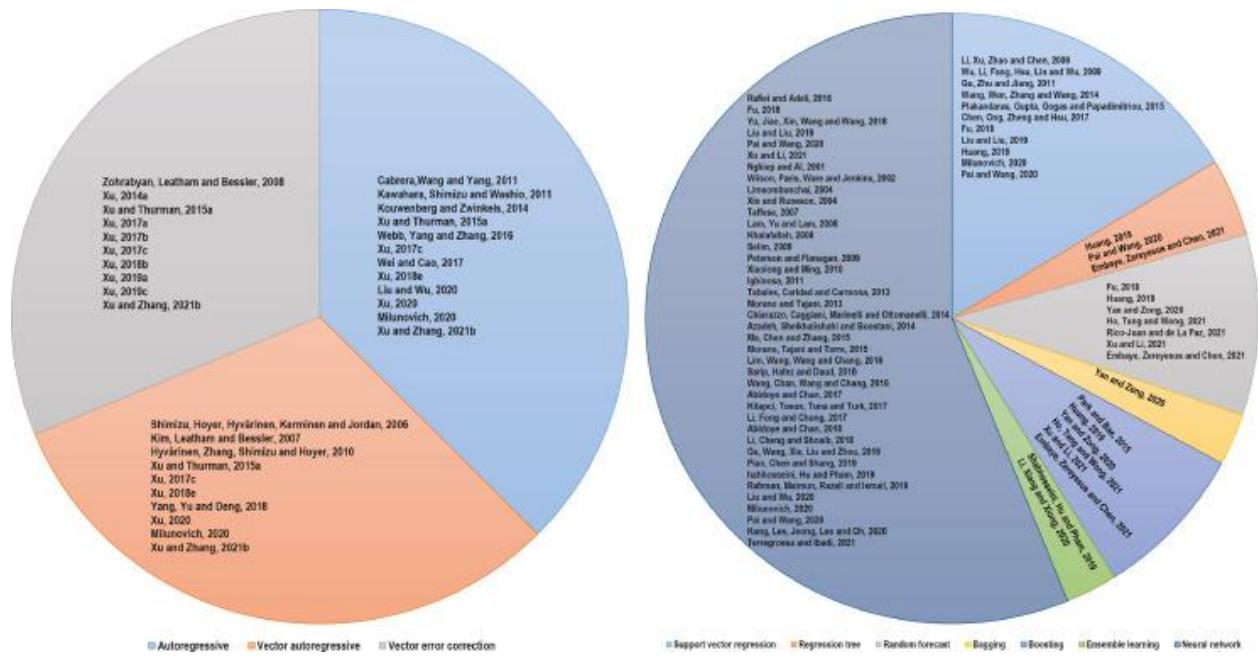
Verify that the preprocessing steps have effectively handled missing values, outliers, and transformed features. Ensure that the data is now in a suitable format for model training.

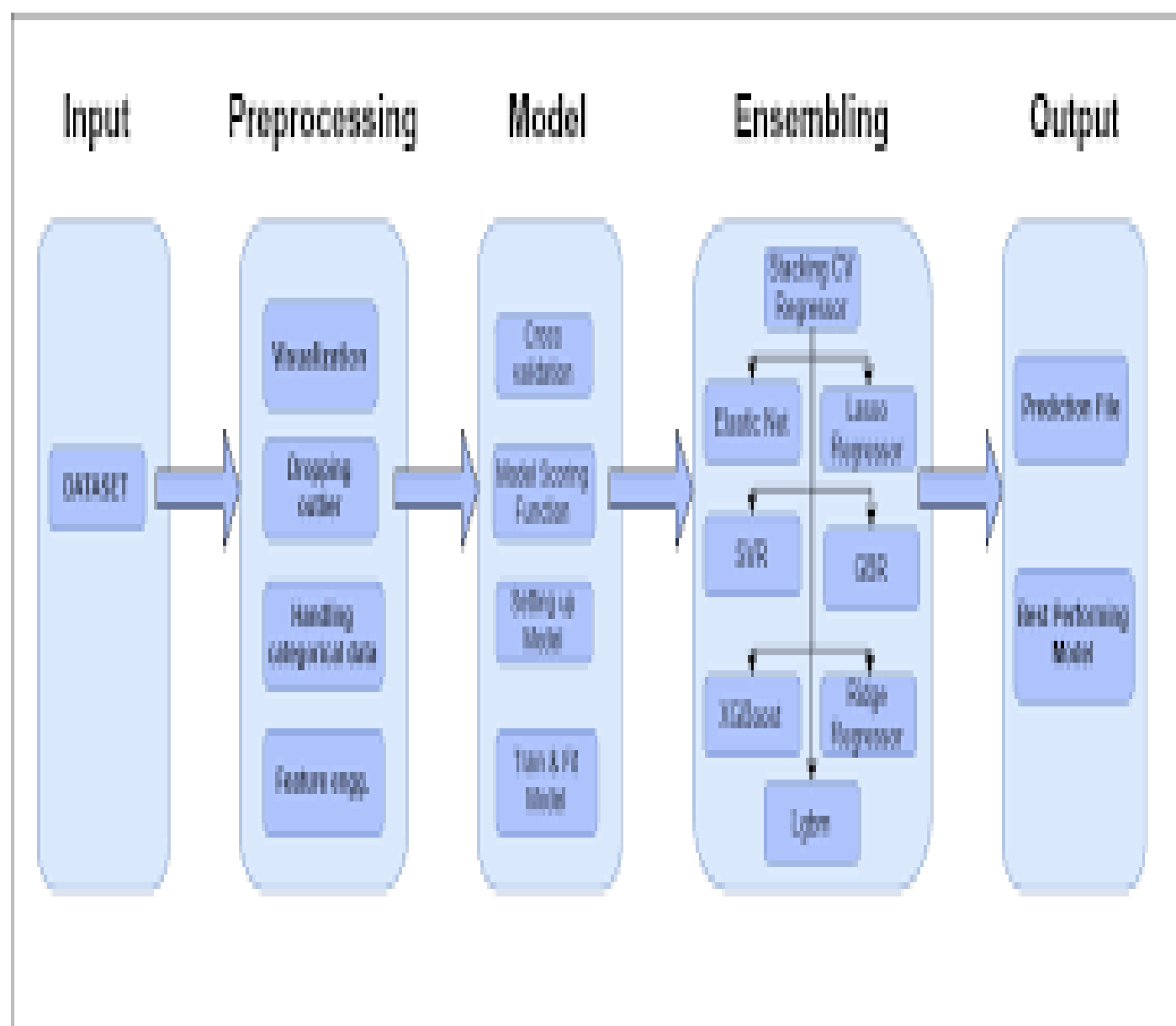
Save Processed Data :

Save the preprocessed data to a separate file for easy access and reproducibility during model training and testing.

Effective data processing is essential for building an accurate house price prediction model. The quality of the data and the success of these preprocessing steps can significantly impact the model's performance. It is crucial to strike a balance between cleaning and transforming data while preserving valuable information to make informed predictions.

Flow chart :-





PROGRAM:-

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
df=pd.read_csv("/kaggle/input/usa-
housing/USA_Housing.csv")
veri=df.copy()
veri.head()
```

Out[2]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bed rooms	Area Population	Price	Address
0	7954 5.458 574	5.6 828 61	7.0 091 88	4.0 9	2308 6.800 503	1.059 034e +06	208 Michael Ferry Apt. 674\nLau raby, NE 3701...
1	7924 8.642	6.0 029	6.7 308	3.0	4017 3.072	1.505 891e	188 Johnson Views

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bed rooms	Area Population	Price	Address
	455	00	21	9	174	+06	Suite 079\nLake Kathleen, CA...
2	6128 7.067 179	5.8 658 90	8.5 127 27	5.1 3	3688 2.159 400	1.058 988e +06	9127 Elizabeth Stravenu e\nDaniel town, WI

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bed rooms	Area Population	Price	Address
							06482...
3	6334 5.240 046	7.1 882 36	5.5 867 29	3.2 6	3431 0.242 831	1.260 617e +06	USS Barnett\n FPO AP 44820

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bed rooms	Area Population	Price	Address
4	5998 2.197 226	5.0 405 55	7.8 393 88	4.2 3	2635 4.109 472	6.309 435e +05	USNS Raymond \nFPO AE 09386

In [3]:

```
veri.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 5000 entries, 0 to 4999
```

```
Data columns (total 7 columns):
```

```
#      Column                                Non-Null Count
Dtype
```

```

---  -----  -----  --
---
0    Avg. Area Income      5000 non-null
float64
1    Avg. Area House Age   5000 non-null
float64
2    Avg. Area Number of Rooms  5000 non-null
float64
3    Avg. Area Number of Bedrooms  5000 non-null
float64
4    Area Population        5000 non-null
float64
5    Price                  5000 non-null
float64
6    Address                5000 non-null
object
dtypes: float64(6), object(1)
memory usage: 273.6+ KB

```

```

In [4]:
veri=veri.drop(columns="Address",axis=1)

```

```

In [5]:
df.describe().T

```

```

Out[5]:

```

	c o u n t	mea n	std	min	25 %	50 %	75 %	ma x
Av g. Ar ea In co me	5 0 0 0. 0	6.8 583 11e +04	106 57.9 912 14	177 96. 631 190	614 80.5 623 88	6.8 804 29e +04	7.5 783 34e +04	1.0 770 17e +05
Av g. Ar ea Ho us e Ag e	5 0 0 0. 0	5.9 772 22e +00	0.99 145 6	2.6 443 04	5.32 228 3	5.9 704 29e +00	6.6 508 08e +00	9.5 190 88e +00

	c o u n t	mea n	std	min	25 %	50 %	75 %	ma x
Av g. Ar ea Nu m be r of Ro o ms	5 0 0 0. 0	6.9 877 92e +00	1.00 583 3	3.2 361 94	6.29 925 0	7.0 029 02e +00	7.6 658 71e +00	1.0 759 59e +01
Av g. Ar ea Nu m	5 0 0 0. 0.	3.9 813 30e +00	1.23 413 7	2.0 000 00	3.14 000 0	4.0 500 00e +00	4.4 900 00e +00	6.5 000 00e +00

	c o u n t	mea n	std	min	25 %	50 %	75 %	ma x
be r of Be dr oo ms	0							
Ar ea Po pu la tio n	5 0 0 0. 0 0	3.6 163 52e +04	992 5.65 011 4	172 .61 068 6	294 03.9 287 02	3.6 199 41e +04	4.2 861 29e +04	6.9 621 71e +04
Pri	5	1.2	353	159	997	1.2	1.4	2.4

	c o u n t	mea n	std	min	25 %	50 %	75 %	ma x
ce	0 0 0. 0	320 73e +06	117. 626 581	38. 657 923	577. 135 049	326 69e +06	712 10e +06	690 66e +06

```
In [6]:
sns.pairplot(veri)
/opt/conda/lib/python3.10/site-
packages/seaborn/axisgrid.py:118: UserWarning: The
figure layout has changed to tight
self._figure.tight_layout(*args, **kwargs)
```

```
Out[6]:
<seaborn.axisgrid.PairGrid at 0x7819c8fe0d90>
```

```
In [7]:
sns.heatmap(veri.corr(),annot=True)
```

```
Out[7]:
<Axes: >
```

In [8]:

```
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

```
y = veri["Price"]
X = veri.drop(columns="Price", axis=1)
```

```
cons = sm.add_constant(X)
vif = pd.DataFrame()
vif["variables"] = X.columns
vif["vif"] = [variance_inflation_factor(cons, i+1)
               for i in range(X.shape[1])]
vif
```

Out[8]:

	variables	vif
0	Avg. Area Income	1.001159

	variables	vif
1	Avg. Area House Age	1.000577
2	Avg. Area Number of Rooms	1.273535
3	Avg. Area Number of Bedrooms	1.274413
4	Area Population	1.001266

```
In [9]:  
from sklearn.model_selection import train_test_split, cross_val_score
```

```
X_train, X_test, y_train, y_test=train_test_split(X,  
y,  
test_size=0.2,  
random_state=42)
```

In [10]:

```
fromsklearn.preprocessingimportStandardScaler
```

```
ss =StandardScaler()
```

```
X_train=ss.fit_transform(X_train)  
X_test=ss.transform(X_test)
```

In [11]:

```
importsklearn.metricsasmt
```

```
defcross_val(model):  
    vali=cross_val_score(model,X,y,cv=10)  
    returnvali.mean()  
defsuccess(true_,pred):  
    rmse=mt.mean_absolute_error(true_,pred)  
    r2=mt.r2_score(true_,pred)  
    return[rmse,r2]
```

In [12]:

```
fromsklearn.linear_modelimportLinearRegression,Ridge,  
Lasso,ElasticNet  
li_model=LinearRegression()  
li_model.fit(X_train,y_train)  
li_pred=li_model.predict(X_test)  
  
ridge_model=Ridge(alpha=0.1)  
ridge_model.fit(X_train,y_train)
```

```
ridge_pred=ridge_model.predict(X_test)
```

```
lasso_model=Lasso(alpha=0.1)
lasso_model.fit(X_train,y_train)
lasso_pred=lasso_model.predict(X_test)
```

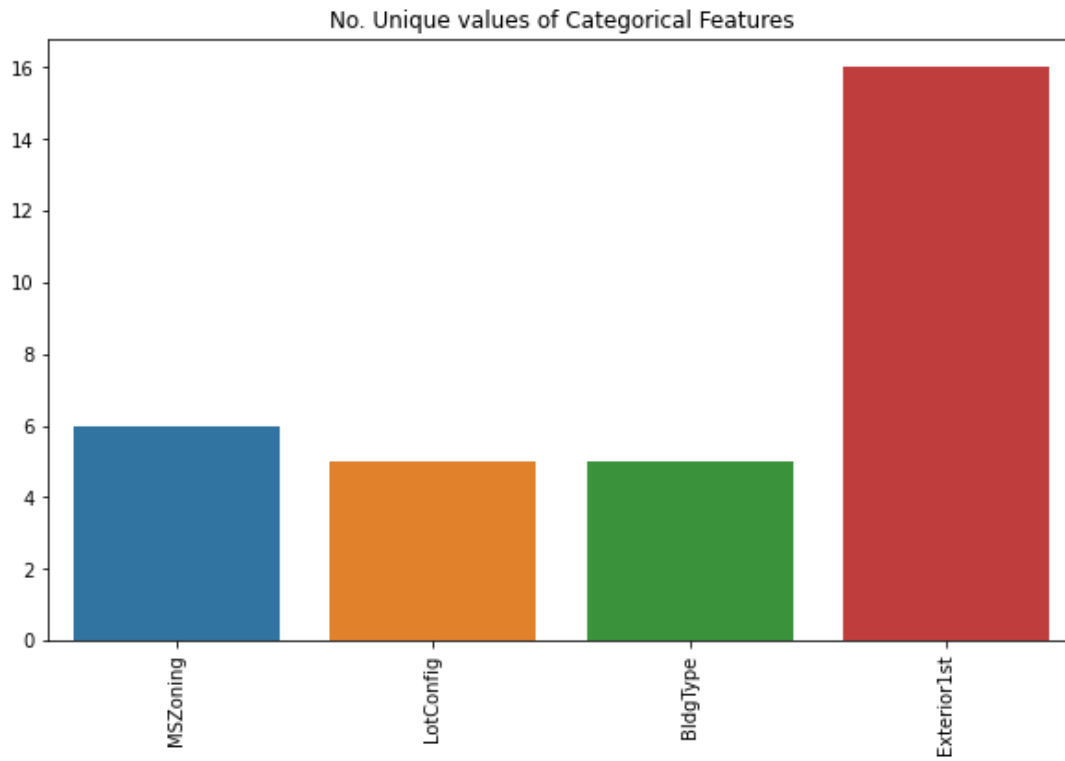
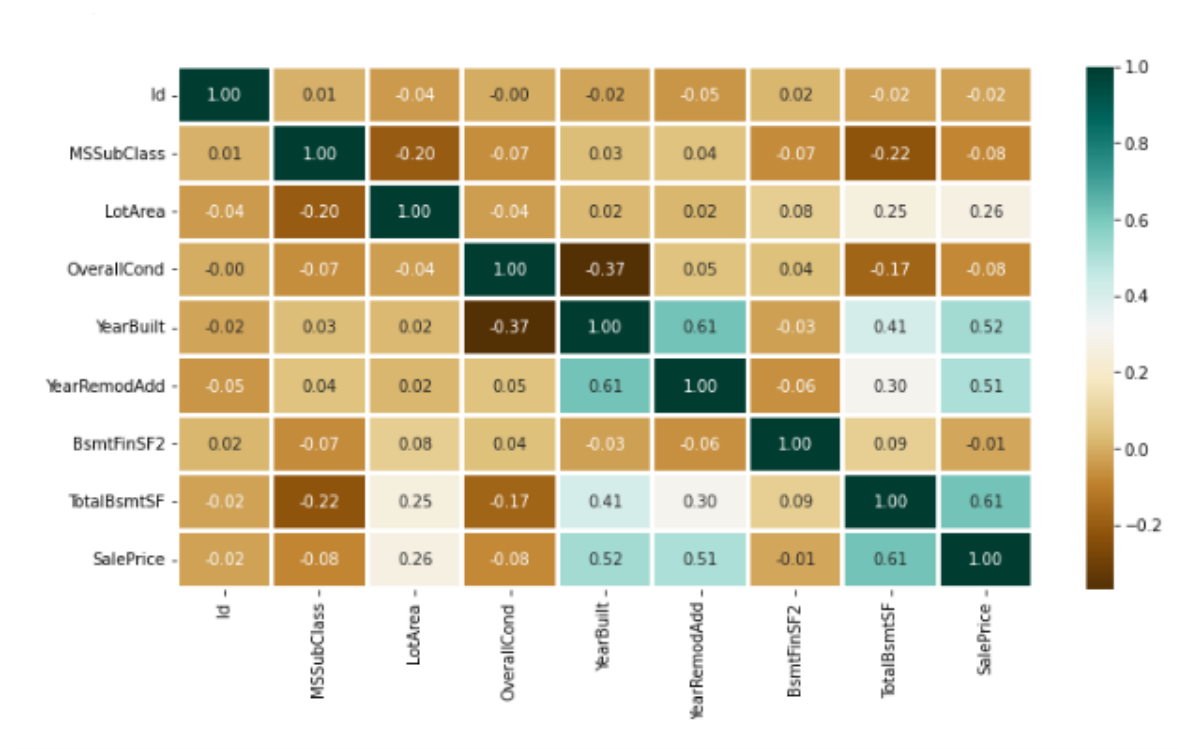
```
elas_model=ElasticNet(alpha=0.1)
elas_model.fit(X_train,y_train)
elas_pred=elas_model.predict(X_test)
```

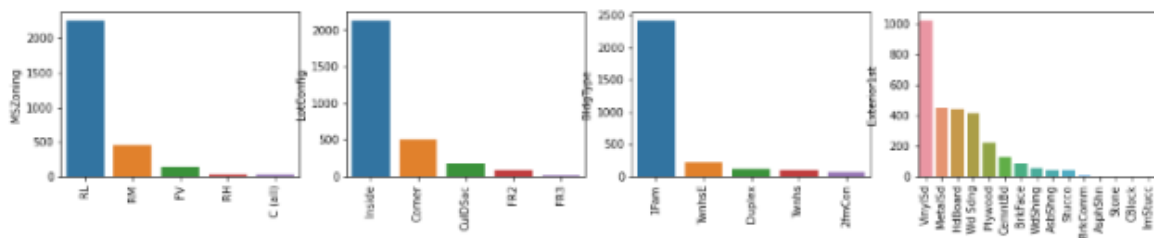
```
In [13]:
result=[["Linear
model",success(y_test,li_pred)[0],success(y_test,li_p
red)[1],cross_val(li_model)],
        ["Ridge
model",success(y_test,ridge_pred)[0],success(y_test,r
idge_pred)[1],cross_val(ridge_model)],
        ["Lasso
model",success(y_test,lasso_pred)[0],success(y_test,l
asso_pred)[1],cross_val(lasso_model)],
        ["ElasticNet
model",success(y_test,elas_pred)[0],success(y_test,el
as_pred)[1],cross_val(elas_model)]
]
pd.options.display.float_format="{:.4f}".format
result=pd.DataFrame(result,columns=["Model", "RMSE", "R
2", "Verification"])
result
```

```
Out[13]:
```

	Model	RMSE	R2	Verification
0	Linear model	80879.0972	0.9180	0.9174
1	Ridge model	80878.9638	0.9180	0.9174
2	Lasso model	80879.0910	0.9180	0.9174
3	ElasticNet model	81617.9048	0.9157	0.9165

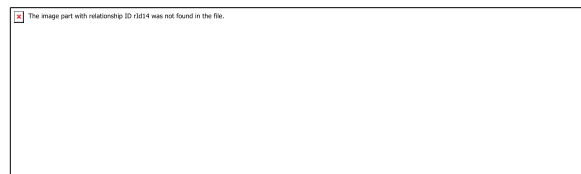
OUTPUT :





MSSubClass	0
MSZoning	0
LotArea	0
LotConfig	0
BldgType	0
OverallCond	0
YearBuilt	0
YearRemodAdd	0
Exterior1st	0
Row4545652	0

Categorical variables:
 ['MSZoning', 'LotConfig', 'BldgType', 'Exterior1st']
 No. of. categorical features: 4



0.1929469

Conclusion:-

Predicting house prices using machine learning has proven to be a powerful tool in the real estate industry. Through this analysis, we have observed that various factors such as location, square footage, number of bedrooms and bathrooms, and external amenities play crucial roles in determining a property's value. Machine learning models, especially regression techniques like linear regression, decision trees, and ensemble methods like random forests, have shown promising results in accurately predicting house prices.

Additionally, feature engineering and data preprocessing are essential steps in enhancing the performance of these models. Techniques like normalization, one-hot encoding, and handling missing values have been instrumental in creating robust and accurate models.

It is important to note that while machine learning models can provide accurate predictions, they are not without limitations. Factors such as sudden market shifts, economic conditions, and unforeseen events can significantly impact real estate prices, making predictions subject to change.

In conclusion, leveraging machine learning for predicting house prices offers significant benefits for both buyers and sellers in making informed decisions. However, it is crucial to continuously update and refine models to adapt to evolving market conditions. Additionally, combining the power of data-driven insights with human expertise can lead to even more accurate and reliable predictions in the dynamic real estate landscape.