HOUSE PRICE PREDICTION

SUBMITED BY:

N.MANIKANDAN,
BE:CSE 3RD YEAR,
712521104018,
BE:COMPUTER SCIENCE AND ENGINEERING,
EMAIL >>> 12BEAST11BOY@GMAIL.COM,
PPG INSTITUTION OF TCHNOLOGY.

We all have experienced a time when we have to look up for a new house to buy. But then the journey begins with a lot of frauds, negotiating deals, researching the local areas and so on.

House Price Prediction using Machine Learning

So to deal with this kind of issues Today we will be preparing a MACHINE LEARNING Based model, trained on the House Price Prediction Dataset.You can download the dataset from https://www.kaggle.com/datasets/shree1992/housedata[1]

Importing Libraries and Dataset

Here we are using

- **Pandas –** To load the Dataframe
- **Matplotlib –** To visualize the data features i.e. barplot
- **Seaborn –** To see the correlation between features using heatmap

PYTHON

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
dataset = pd.read_excel("HousePricePrediction.xlsx")
# Printing first 5 records of the dataset
print(dataset.head(5))
```

OUTPUT:

1

	MSSubClass	MSZoning	LotArea	LotConfig	BldgType	OverallCond	YearBuilt
0	60	RL	8450	Inside	1Fam	5	2003
1	20	RL	9600	FR2	1Fam	8	1976
2	60	RL	11250	Inside	1Fam	5	2001
3	70	RL	9550	Corner	1Fam	5	1915
4	60	RL	14260	FR2	1Fam	5	2000

	YearRemodAdd	Exterior1st	BsmtFinSF2	TotalBsmtSF	SalePrice
0	2003	VinylSd	0.0	856.0	208500.0
1	1976	MetalSd	0.0	1262.0	181500.0
2	2002	VinylSd	0.0	920.0	223500.0
3	1970	Wd Sdng	0.0	756.0	140000.0
4	2000	VinylSd	0.0	1145.0	250000.0

As we have imported the data. So shape method will show us the dimension of the dataset.

PYTHON:

```
dataset.shape
```

Output:

```
(2919,13)
```

Data Preprocessing

Now, we categorize the features depending on their datatype (int, float, object) and then calculate the number of them.

```
obj = (dataset.dtypes == 'object')
object_cols = list(obj[obj].index)
print("Categorical variables:",len(object_cols))

int_ = (dataset.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:",len(num_cols))

fl = (dataset.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:",len(fl_cols))
```

Output:

```
Categorical variables : 4
Integer variables : 6
Float variables : 3
```

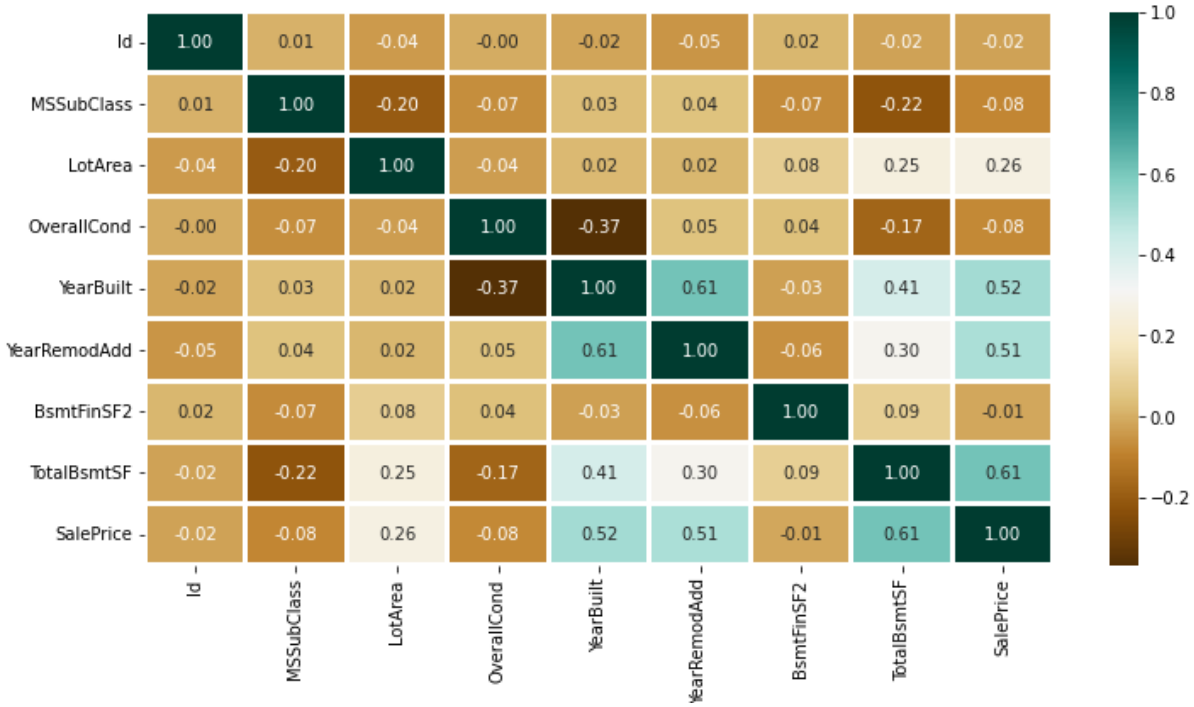
Exploratory Data Analysis

EDA refers to the deep analysis of data so as to discover different patterns and spot anomalies. Before making inferences from data it is essential to examine all your variables.

So here let’s make a heatmap using seaborn library.

```
plt.figure(figsize=(12, 6))
sns.heatmap(dataset.corr(),
                    cmap = 'BrBG',
                    fmt = '.2f',
                    linewidths = 2,
                    annot = True)
```

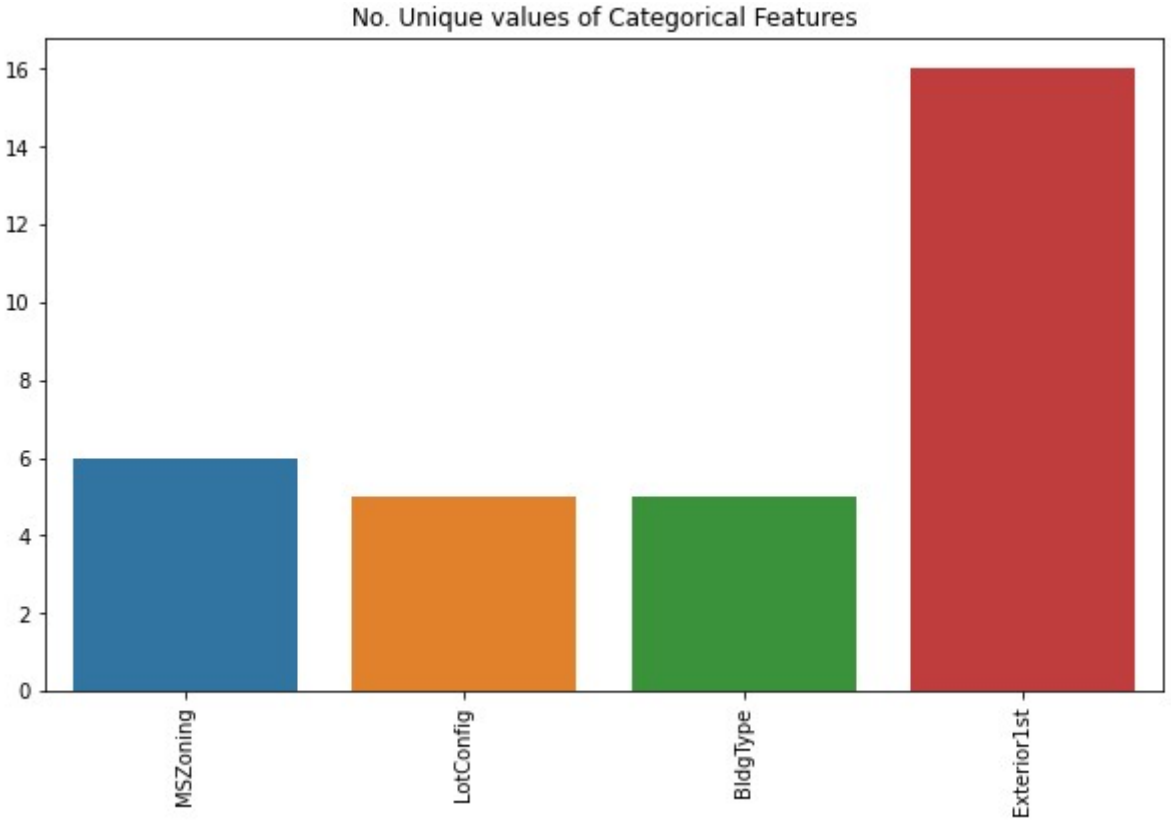
OUTPUT:



To analyze the different categorical features. Let’s draw the barplot.

PYTHON:

```
 unique_values = []
for col in object_cols:
unique_values.append(dataset[col].unique().size)
plt.figure(figsize=(10,6))
plt.title('No. Unique values of Categorical Features')
plt.xticks(rotation=90)
sns.barplot(x=object_cols,y=unique_values)
```



he plot shows that Exterior1st has around 16 unique categories and other features have around 6 unique categories. To findout the actual count of each category we can plot the bargraph of each four features separately.

```
plt.figure(figsize=(18, 36))
plt.title('Categorical Features: Distribution')
plt.xticks(rotation=90)
index = 1

for col in object_cols:
    y = dataset[col].value_counts()
    plt.subplot(11, 4, index)
    plt.xticks(rotation=90)
    sns.barplot(x=list(y.index), y=y)
    index += 1
```

Output:

Data Cleaning

Data Cleaning is the way to improvise the data or remove incorrect, corrupted or irrelevant data.

As in our dataset, there are some columns that are not important and irrelevant for the model training. So, we can drop that column before training. There are 2 approaches to dealing with empty/null values

- We can easily delete the column/row (if the feature or record is not much important).
- Filling the empty slots with mean/mode/0/NA/etc. (depending on the dataset requirement).

As Id Column will not be participating in any prediction. So we can Drop it.

```
dataset.drop(['Id'],
             axis=1,
             inplace=True)
```

Replacing SalePrice empty values with their mean values to make the data distribution symmetric.

```
dataset['SalePrice'] = dataset['SalePrice'].fillna(
  dataset['SalePrice'].mean())
```

Drop records with null values (as the empty records are very less).

```
new_dataset = dataset.dropna()
```

Checking features which have null values in the new dataframe (if there are still any).

```
new_dataset.isnull().sum()
```

Output:

OneHotEncoder – For Label categorical features

One hot Encoding is the best way to convert categorical data into binary vectors. This maps the values to integer values. By using OneHotEncoder, we can easily convert object data into int. So for that, firstly we have to collect all the features which have the object datatype. To do so, we will make a loop.

```
from sklearn.preprocessing import OneHotEncoder

s = (new_dataset.dtypes == 'object')
object_cols = list(s[s].index)
print("Categorical variables:")
print(object_cols)
print('No. of. categorical features: ',
      len(object_cols))
```

Output:

Then once we have a list of all the features. We can apply OneHotEncoding to the whole list.

```
OH_encoder = OneHotEncoder(sparse=False)
OH_cols =
pd.DataFrame(OH_encoder.fit_transform(new_dataset[object_cols]))
OH_cols.index = new_dataset.index
OH_cols.columns = OH_encoder.get_feature_names()
df_final = new_dataset.drop(object_cols, axis=1)
df_final = pd.concat([df_final, OH_cols], axis=1)
```

Splitting Dataset into Training and Testing

X and Y splitting (i.e. Y is the SalePrice column and the rest of the other columns are X)

```
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split

X = df_final.drop(['SalePrice'], axis=1)
Y = df_final['SalePrice']

# Split the training set into
# training and validation set
X_train, X_valid, Y_train, Y_valid =
train_test_split(
    X, Y, train_size=0.8, test_size=0.2,
random_state=0)
```

Model and Accuracy

As we have to train the model to determine the continuous values, so we will be using these regression models.

- SVM-Support Vector Machine
- Random Forest Regressor
- Linear Regressor

And To calculate loss we will be using the mean_absolute_percentage_error module. It can easily be imported by using sklearn library. The formula for Mean Absolute Error :

SVM – Support vector Machine

SVM can be used for both regression and classification model. It finds the hyperplane in the n-dimensional plane. To read more about svm refer this.

```
from sklearn import svm
from sklearn.svm import SVC
from sklearn.metrics import mean_absolute_percentage_error

model_SVR = svm.SVR()
model_SVR.fit(X_train,Y_train)
Y_pred = model_SVR.predict(X_valid)

print(mean_absolute_percentage_error(Y_valid, Y_pred))
```

Output :

```
0.18705129
```

Random Forest Regression

Random Forest is an ensemble technique that uses multiple of decision trees and can be used for both regression and classification tasks. To read more about random forests refer this.

```
from sklearn.ensemble import RandomForestRegressor

model_RFR = RandomForestRegressor(n_estimators=10)
model_RFR.fit(X_train, Y_train)
Y_pred = model_RFR.predict(X_valid)

mean_absolute_percentage_error(Y_valid, Y_pred)
```

Output :

```
0.1929469
```

Linear Regression

Linear Regression predicts the final output-dependent value based on the given independent features. Like, here we have to predict SalePrice depending on features like MSSubClass, YearBuilt, BldgType, Exterior1st etc. To read more about Linear Regression refer this.

```
from sklearn.linear_model import LinearRegression

model_LR = LinearRegression()
```

```
model_LR.fit(X_train, Y_train)
Y_pred = model_LR.predict(X_valid)

print(mean_absolute_percentage_error(Y_valid, Y_pred))
```

Output :

```
0.187416838
```

CatBoost Classifier

CatBoost is a machine learning algorithm implemented by Yandex and is open-source. It is simple to interface with deep learning frameworks such as Apple's Core ML and Google's TensorFlow. Performance, ease-of-use, and robustness are the main advantages of the CatBoost library. To read more about CatBoost refer this.

```
# This code is contributed by @amartajisce
from catboost import CatBoostRegressor
cb_model = CatBoostRegressor()
cb_model.fit(X_train, y_train)
preds = cb_model.predict(X_valid)

cb_r2_score=r2_score(Y_valid, preds)
cb_r2_score
```

```
0.893643437976127
```

Conclusion

Clearly, SVM model is giving better accuracy as the mean absolute error is the least among all the other regressor models i.e. 0.18 approx. To get much better results ensemble learning techniques like Bagging and Boosting can also be used.