US House Price Prediction

Table of Contents

This project contains:

About the data:

This dataset is from the Kaggle platform.
https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques

alt

The main motive of this project is to predict the price of a house information like its location, area, no. of rooms etc.Random Forest Classifier model will be implemented to predict the final price of each home.

```
# installing opendatasets library
!pip install opendatasets
```

```
Requirement already satisfied: opendatasets in /opt/conda/lib/python3.9/site-packages
(0.1.22)
Requirement already satisfied: click in /opt/conda/lib/python3.9/site-packages (from
opendatasets) (8.0.3)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.9/site-packages (from
opendatasets) (4.62.3)
Requirement already satisfied: kaggle in /opt/conda/lib/python3.9/site-packages (from
opendatasets) (1.5.12)
Requirement already satisfied: requests in /opt/conda/lib/python3.9/site-packages (from
kaggle->opendatasets) (2.26.0)
Requirement already satisfied: certifi in /opt/conda/lib/python3.9/site-packages (from
kaggle->opendatasets) (2021.10.8)
Requirement already satisfied: python-dateutil in /opt/conda/lib/python3.9/site-
packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: six>=1.10 in /opt/conda/lib/python3.9/site-packages
(from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: urllib3 in /opt/conda/lib/python3.9/site-packages (from
```

```
kaggle->opendatasets) (1.26.7)
Requirement already satisfied: python-slugify in /opt/conda/lib/python3.9/site-packages
(from kaggle->opendatasets) (7.0.0)
Requirement already satisfied: text-unidecode>=1.3 in /opt/conda/lib/python3.9/site-
packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.9/site-packages
(from requests->kaggle->opendatasets) (3.1)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/opt/conda/lib/python3.9/site-packages (from requests->kaggle->opendatasets) (2.0.0)
```

```
!pip install jovian
```

```
Requirement already satisfied: jovian in /opt/conda/lib/python3.9/site-packages
(0.2.45)
Requirement already satisfied: uuid in /opt/conda/lib/python3.9/site-packages (from
jovian) (1.30)
Requirement already satisfied: click in /opt/conda/lib/python3.9/site-packages (from
jovian) (8.0.3)
Requirement already satisfied: requests in /opt/conda/lib/python3.9/site-packages (from
jovian) (2.26.0)
Requirement already satisfied: pyyaml in /opt/conda/lib/python3.9/site-packages (from
jovian) (6.0)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.9/site-
packages (from requests->jovian) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.9/site-packages
(from requests->jovian) (3.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.9/site-
packages (from requests->jovian) (1.26.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/opt/conda/lib/python3.9/site-packages (from requests->jovian) (2.0.0)
```

```
# importing necessary libraries
import opendatasets as od
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib
import pandas as pd
import numpy as np
import jovian
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
%matplotlib inline
matplotlib.rcParams['figure.facecolor']='white'
```

```
data_url='https://www.kaggle.com/competitions/house-prices-advanced-regression-techniqu
```

```
od.download(data_url)
```

```
Skipping, found downloaded files in "./house-prices-advanced-regression-techniques"
(use force=True to force download)
```

```
data_dir='./house-prices-advanced-regression-techniques'
```

```
import os
```

```
os.listdir(data_dir)
```

```
['test.csv', 'train.csv', 'sample_submission.csv', 'data_description.txt']
```

```
test_csv=data_dir+'/test.csv'
train_csv=data_dir+'/train.csv' # The training data only will be used for EDA, performa
sample_submission=data_dir+'/sample_submission.csv'
data_description='/data_description.txt'
```

```
train_df=pd.read_csv(train_csv)
```

```
train_df
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolA
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	AllPub	...	
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	AllPub	...	
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl	AllPub	...	
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl	AllPub	...	
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	Lvl	AllPub	...	

1460 rows × 81 columns

```
test_df=pd.read_csv(test_csv)
test_df
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	Scree

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	Scree
0	1461	20	RH	80.0	11622	Pave	NaN	Reg	Lvl	AllPub	...	
1	1462	20	RL	81.0	14267	Pave	NaN	IR1	Lvl	AllPub	...	
2	1463	60	RL	74.0	13830	Pave	NaN	IR1	Lvl	AllPub	...	
3	1464	60	RL	78.0	9978	Pave	NaN	IR1	Lvl	AllPub	...	
4	1465	120	RL	43.0	5005	Pave	NaN	IR1	HLS	AllPub	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1454	2915	160	RM	21.0	1936	Pave	NaN	Reg	Lvl	AllPub	...	
1455	2916	160	RM	21.0	1894	Pave	NaN	Reg	Lvl	AllPub	...	
1456	2917	20	RL	160.0	20000	Pave	NaN	Reg	Lvl	AllPub	...	
1457	2918	85	RL	62.0	10441	Pave	NaN	Reg	Lvl	AllPub	...	
1458	2919	60	RL	74.0	9627	Pave	NaN	Reg	Lvl	AllPub	...	

1459 rows × 80 columns

```
sample_submission_df=pd.read_csv(sample_submission)
sample_submission_df
```

	Id	SalePrice
0	1461	169277.052498
1	1462	187758.393989
2	1463	183583.683570
3	1464	179317.477511
4	1465	150730.079977
...	...	...
1454	2915	167081.220949
1455	2916	164788.778231
1456	2917	219222.423400
1457	2918	184924.279659
1458	2919	187741.866657

1459 rows × 2 columns

```
jovian.commit()
```

```
[jovian] Updating notebook "kavinm642/house-prices-prediction" on https://jovian.com
[jovian] Committed successfully! https://jovian.com/kavinm642/house-prices-prediction
```

'https://jovian.com/kavinm642/house-prices-prediction'

Exploratory Data Analysis

```
#checking the no.of rows and columns
n_rows=1460
```

```
n_cols=81
print('The dataset contains {} rows and {} columns.'.format(n_rows,n_cols))
```

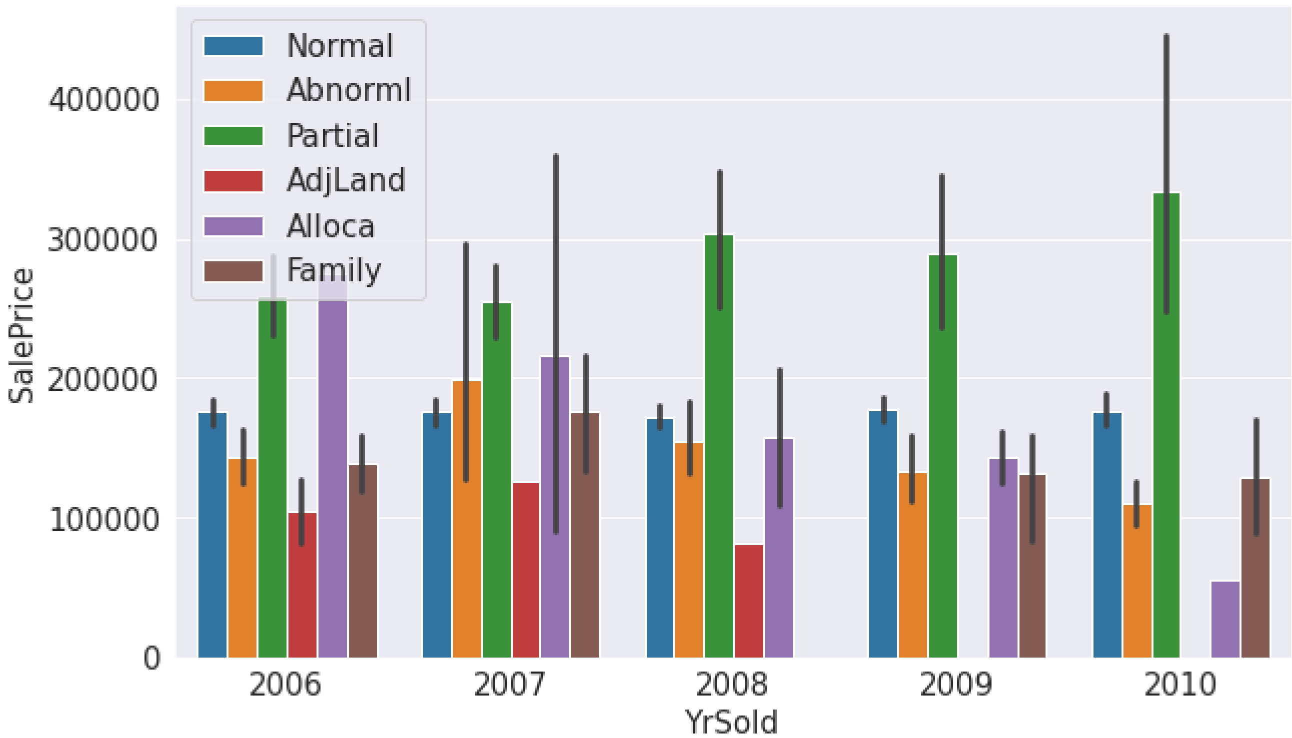
```
The dataset contains 1460 rows and 81 columns.
```

```
!pip install plotly
```

```
Requirement already satisfied: plotly in /opt/conda/lib/python3.9/site-packages
(5.13.0)
Requirement already satisfied: tenacity>=6.2.0 in /opt/conda/lib/python3.9/site-
packages (from plotly) (8.1.0)
```

```
# For visualizing the data
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib
%matplotlib inline
sns.set_style('darkgrid')
matplotlib.rcParams['font.size']=15
matplotlib.rcParams['figure.figsize']=(10,6)
```

```
# year sold and sale price
sns.barplot(data=train_df,x='YrSold',y='SalePrice',hue='SaleCondition')
plt.legend(loc='upper left')
plt.show()
```




```
#comparing the amenities of Garage in a house according to its price
fig=px.scatter(train_df,x='GarageYrBlt',y='GrLivArea',color='GarageType',opacity=1.0,hc
```

```
fig.update_traces(marker_size=5)
fig.show()
```

```
train_df.BsmtFinSF1.corr(train_df.BsmtFinSF2)
```

```
-0.050117400047150915
```

```
train_df.LotFrontage.corr(train_df.LotArea)
```

```
0.4260950187718078
```

```
jovian.commit()
```

```
[jovian] Updating notebook "kavinm642/house-prices-prediction" on https://jovian.com
[jovian] Committed successfully! https://jovian.com/kavinm642/house-prices-prediction
```

```
'https://jovian.com/kavinm642/house-prices-prediction'
```

Preparing the dataset for training

```
train_df
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolA
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolA
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	AllPub	...	
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	AllPub	...	
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl	AllPub	...	
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl	AllPub	...	
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	Lvl	AllPub	...	

1460 rows × 81 columns

`input_col` which contains the data that can be used as an `input` to train the model.

`target_col` which contains the data that can be used as a `target` to train the model.

```
input_col=['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street', 'Alley', 'LotS
```

```
target_col='SalePrice'
```

```
print(list(input_col))
```

```
['MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street', 'Alley', 'LotShape',
'LandContour', 'Utilities', 'LotConfig', 'LandSlope', 'Neighborhood', 'Condition1',
'Condition2', 'BldgType', 'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt',
'YearRemodAdd', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond',
'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2', 'BsmtFinSF2',
'BsmtUnfSF', 'TotalBsmtSF', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical',
'1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath',
'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual', 'TotRmsAbvGrd',
'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType', 'GarageYrBlt', 'GarageFinish',
'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond', 'PavedDrive', 'WoodDeckSF',
'OpenPorchSF', 'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType', 'SaleCondition']
```

```
len(input_col)
```

```
79
```

```
print(target_col)
```

SalePrice

```
input_df=train_df[input_col].copy()
```

```
target_df=train_df[target_col].copy()
```

```
input_df
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	S
0	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	Inside	...	
1	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	FR2	...	
2	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	Inside	...	
3	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	Corner	...	
4	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	FR2	...	
...	...	...	...	...	...	...	...	...	...	...	...	
1455	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	AllPub	Inside	...	
1456	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	AllPub	Inside	...	
1457	70	RL	66.0	9042	Pave	NaN	Reg	Lvl	AllPub	Inside	...	
1458	20	RL	68.0	9717	Pave	NaN	Reg	Lvl	AllPub	Inside	...	
1459	20	RL	75.0	9937	Pave	NaN	Reg	Lvl	AllPub	Inside	...	

1460 rows × 79 columns

```
target_df
```

```
0       208500
1       181500
2       223500
3       140000
4       250000
         ...
1455    175000
1456    210000
1457    266500
1458    142125
1459    147500
Name: SalePrice, Length: 1460, dtype: int64
```

```
jovian.commit()
```

```
[jovian] Updating notebook "kavinm642/house-prices-prediction" on https://jovian.com
[jovian] Committed successfully! https://jovian.com/kavinm642/house-prices-prediction
```

```
'https://jovian.com/kavinm642/house-prices-prediction'
```

Identifying Numeric and Categoric data

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1460 non-null   int64
 1   MSSubClass     1460 non-null   int64
 2   MSZoning       1460 non-null   object
 3   LotFrontage    1201 non-null   float64
 4   LotArea        1460 non-null   int64
 5   Street         1460 non-null   object
 6   Alley          91 non-null     object
 7   LotShape       1460 non-null   object
 8   LandContour    1460 non-null   object
 9   Utilities      1460 non-null   object
 10  LotConfig      1460 non-null   object
 11  LandSlope      1460 non-null   object
 12  Neighborhood   1460 non-null   object
 13  Condition1     1460 non-null   object
 14  Condition2     1460 non-null   object
 15  BldgType       1460 non-null   object
 16  HouseStyle     1460 non-null   object
 17  OverallQual    1460 non-null   int64
 18  OverallCond    1460 non-null   int64
 19  YearBuilt      1460 non-null   int64
 20  YearRemodAdd   1460 non-null   int64
 21  RoofStyle      1460 non-null   object
 22  RoofMatl       1460 non-null   object
 23  Exterior1st    1460 non-null   object
 24  Exterior2nd    1460 non-null   object
 25  MasVnrType     1452 non-null   object
 26  MasVnrArea     1452 non-null   float64
 27  ExterQual      1460 non-null   object
 28  ExterCond      1460 non-null   object
 29  Foundation     1460 non-null   object
 30  BsmtQual       1423 non-null   object
 31  BsmtCond       1423 non-null   object
 32  BsmtExposure   1422 non-null   object
 33  BsmtFinType1   1423 non-null   object
 34  BsmtFinSF1     1460 non-null   int64
 35  BsmtFinType2   1422 non-null   object
```

```
36  BsmtFinSF2     1460 non-null   int64
37  BsmtUnfSF      1460 non-null   int64
38  TotalBsmtSF    1460 non-null   int64
39  Heating        1460 non-null   object
40  HeatingQC      1460 non-null   object
41  CentralAir     1460 non-null   object
42  Electrical     1459 non-null   object
43  1stFlrSF       1460 non-null   int64
44  2ndFlrSF       1460 non-null   int64
45  LowQualFinSF   1460 non-null   int64
46  GrLivArea      1460 non-null   int64
47  BsmtFullBath   1460 non-null   int64
48  BsmtHalfBath   1460 non-null   int64
49  FullBath       1460 non-null   int64
50  HalfBath       1460 non-null   int64
51  BedroomAbvGr   1460 non-null   int64
52  KitchenAbvGr   1460 non-null   int64
53  KitchenQual    1460 non-null   object
54  TotRmsAbvGrd   1460 non-null   int64
55  Functional     1460 non-null   object
56  Fireplaces     1460 non-null   int64
57  FireplaceQu    770 non-null    object
58  GarageType     1379 non-null   object
59  GarageYrBlt    1379 non-null   float64
60  GarageFinish   1379 non-null   object
61  GarageCars     1460 non-null   int64
62  GarageArea     1460 non-null   int64
63  GarageQual     1379 non-null   object
64  GarageCond     1379 non-null   object
65  PavedDrive     1460 non-null   object
66  WoodDeckSF     1460 non-null   int64
67  OpenPorchSF    1460 non-null   int64
68  EnclosedPorch  1460 non-null   int64
69  3SsnPorch      1460 non-null   int64
70  ScreenPorch    1460 non-null   int64
71  PoolArea       1460 non-null   int64
72  PoolQC         7 non-null      object
73  Fence          281 non-null    object
74  MiscFeature    54 non-null     object
75  MiscVal        1460 non-null   int64
76  MoSold         1460 non-null   int64
77  YrSold         1460 non-null   int64
78  SaleType       1460 non-null   object
```

```
 79  SaleCondition  1460 non-null   object
 80  SalePrice      1460 non-null   int64
dtypes: float64(3), int64(35), object(43)
memory usage: 924.0+ KB
```

```
numeric_col=input_df.select_dtypes(include=['int64','float64']).columns.tolist()
```

```
categoric_col=input_df.select_dtypes(include=['object']).columns.tolist()
```

```
print(list(numeric_col))
```

```
['MSSubClass', 'LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt',
'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF',
'1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath',
'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces',
'GarageYrBlt', 'GarageCars', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF',
'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold']
```

```
print(list(categoric_col))
```

```
['MSZoning', 'Street', 'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle',
'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'ExterQual',
'ExterCond', 'Foundation', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1',
'BsmtFinType2', 'Heating', 'HeatingQC', 'CentralAir', 'Electrical', 'KitchenQual',
'Functional', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond',
'PavedDrive', 'PoolQC', 'Fence', 'MiscFeature', 'SaleType', 'SaleCondition']
```

Impute Numerical Data

```
missing_count=input_df[numeric_col].isna().sum().sort_values(ascending=False)
missing_count[missing_count>0]
```

```
LotFrontage    259
GarageYrBlt     81
MasVnrArea       8
dtype: int64
```

Importing `Simple Imputer`

```
from sklearn.impute import SimpleImputer
```

```
imputer=SimpleImputer(strategy='mean')
```

```
imputer.fit(input_df[numeric_col])
```

```
SimpleImputer()
```

```
input_df[numeric_col]=imputer.transform(input_df[numeric_col])
```

```
missing_count=input_df[numeric_col].isna().sum().sort_values(ascending=False)
missing_count[missing_count>0]
```

```
Series([], dtype: int64)
```

```
jovian.commit()
```

```
[jovian] Updating notebook "kavinm642/house-prices-prediction" on https://jovian.com
[jovian] Committed successfully! https://jovian.com/kavinm642/house-prices-prediction
```

'https://jovian.com/kavinm642/house-prices-prediction'

Scaling Numerical Values

```
input_df[numeric_col].describe().loc[['min','max']]
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinS
min	20.0	21.0	1300.0	1.0	1.0	1872.0	1950.0	0.0	(
max	190.0	313.0	215245.0	10.0	9.0	2010.0	2010.0	1600.0	5644

2 rows × 36 columns

Importing `MinMaxScaler`

```
scaler=MinMaxScaler()
```

```
scaler.fit(input_df[numeric_col])
```

```
MinMaxScaler()
```

```
input_df[numeric_col]=scaler.transform(input_df[numeric_col])
```

```
input_df[numeric_col].describe().loc[['min','max']]
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.(
max	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.(

2 rows × 36 columns

Here the values turns into boolean expressions after scaling.

Encoding Categoric Values

```
input_df[categoric_col].nunique().sort_values(ascending=False)
```

```
Neighborhood     25
Exterior2nd      16
Exterior1st      15
SaleType          9
Condition1        9
Condition2        8
HouseStyle        8
RoofMatl          8
Functional        7
BsmtFinType2      6
Heating           6
RoofStyle         6
SaleCondition     6
BsmtFinType1      6
GarageType        6
Foundation        6
Electrical        5
FireplaceQu       5
HeatingQC         5
GarageQual        5
GarageCond        5
MSZoning          5
LotConfig         5
ExterCond         5
BldgType          5
BsmtExposure      4
MiscFeature       4
Fence             4
LotShape          4
LandContour       4
BsmtCond          4
KitchenQual       4
MasVnrType        4
ExterQual         4
BsmtQual          4
LandSlope         3
GarageFinish      3
PavedDrive        3
PoolQC            3
Utilities         2
CentralAir        2
Street            2
Alley             2
dtype: int64
```

Importing `OneHotEncoder`

```python
from sklearn.preprocessing import OneHotEncoder
```

```python
encoder=OneHotEncoder(sparse=False,handle_unknown='ignore')
```

```python
encoder.fit(input_df[categoric_col])
```

```
OneHotEncoder(handle_unknown='ignore', sparse=False)
```

```python
encoded_col=list(encoder.get_feature_names_out(categoric_col))
len(encoded_col)
```

```
268
```

```python
input_df[encoded_col]=encoder.transform(input_df[categoric_col])
```

```
/opt/conda/lib/python3.9/site-packages/pandas/core/frame.py:3678: PerformanceWarning:

DataFrame is highly fragmented.  This is usually the result of calling `frame.insert`
many times, which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead.  To get a de-fragmented frame, use `newframe = frame.copy()`
```

```python
input_df
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...
0	0.235294	RL	0.150685	0.033420	Pave	NaN	Reg	Lvl	AllPub	Inside	...
1	0.000000	RL	0.202055	0.038795	Pave	NaN	Reg	Lvl	AllPub	FR2	...
2	0.235294	RL	0.160959	0.046507	Pave	NaN	IR1	Lvl	AllPub	Inside	...
3	0.294118	RL	0.133562	0.038561	Pave	NaN	IR1	Lvl	AllPub	Corner	...
4	0.235294	RL	0.215753	0.060576	Pave	NaN	IR1	Lvl	AllPub	FR2	...
...	...	...	...	...	...	...	...	...	...	...	...
1455	0.235294	RL	0.140411	0.030929	Pave	NaN	Reg	Lvl	AllPub	Inside	...
1456	0.000000	RL	0.219178	0.055505	Pave	NaN	Reg	Lvl	AllPub	Inside	...
1457	0.294118	RL	0.154110	0.036187	Pave	NaN	Reg	Lvl	AllPub	Inside	...
1458	0.000000	RL	0.160959	0.039342	Pave	NaN	Reg	Lvl	AllPub	Inside	...
1459	0.000000	RL	0.184932	0.040370	Pave	NaN	Reg	Lvl	AllPub	Inside	...

1460 rows × 347 columns

```python
jovian.commit()
```

```
[jovian] Updating notebook "kavinm642/house-prices-prediction" on https://jovian.com
[jovian] Committed successfully! https://jovian.com/kavinm642/house-prices-prediction
```

```
'https://jovian.com/kavinm642/house-prices-prediction'
```

Training and Validation Set

```
x_train,x_test,y_train,y_test=train_test_split(input_df[numeric_col+encoded_col],target
```

```
x_train
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFir
1023	0.588235	0.075342	0.008797	0.666667	0.500	0.963768	0.933333	0.008750	0.002
810	0.000000	0.195205	0.041319	0.555556	0.625	0.739130	0.816667	0.061875	0.117
1384	0.176471	0.133562	0.036271	0.555556	0.500	0.485507	0.000000	0.000000	0.036
626	0.000000	0.167979	0.051611	0.444444	0.500	0.637681	0.466667	0.000000	0.000
813	0.000000	0.184932	0.039496	0.555556	0.625	0.623188	0.133333	0.151875	0.107
...	...	...	...	...	...	...	...	...	
1095	0.000000	0.195205	0.037472	0.555556	0.500	0.971014	0.933333	0.000000	0.004
1130	0.176471	0.150685	0.030400	0.333333	0.250	0.405797	0.000000	0.000000	0.110
1294	0.000000	0.133562	0.032120	0.444444	0.750	0.601449	0.666667	0.000000	0.029
860	0.176471	0.116438	0.029643	0.666667	0.875	0.333333	0.800000	0.000000	0.000
1126	0.588235	0.109589	0.011143	0.666667	0.500	0.978261	0.950000	0.081250	0.000

1095 rows × 304 columns

```
x_test
```

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFir
892	0.000000	0.167808	0.033252	0.555556	0.875	0.659420	0.883333	0.000000	0.117
1105	0.235294	0.263699	0.051209	0.777778	0.500	0.884058	0.750000	0.226250	0.182
413	0.058824	0.119863	0.035804	0.444444	0.625	0.398551	0.000000	0.000000	0.000
522	0.176471	0.099315	0.017294	0.555556	0.750	0.543478	0.000000	0.000000	0.070
1036	0.000000	0.232877	0.054210	0.888889	0.500	0.978261	0.966667	0.043750	0.181
...	...	...	...	...	...	...	...	...	
988	0.235294	0.167979	0.050228	0.555556	0.625	0.753623	0.433333	0.186250	0.027
243	0.823529	0.184932	0.044226	0.555556	0.625	0.782609	0.500000	0.000000	0.000
1342	0.235294	0.167979	0.037743	0.777778	0.500	0.942029	0.866667	0.093125	0.000
1057	0.235294	0.167979	0.133955	0.666667	0.625	0.884058	0.733333	0.000000	0.105
1418	0.000000	0.171233	0.036944	0.444444	0.500	0.659420	0.216667	0.000000	0.004

365 rows × 304 columns

```
y_train
```

```
1023    191000
810     181000
1384    105000
```

```
626      139900
813      157900
          ...
1095     176432
1130     135000
1294     115000
860      189950
1126     174000
Name: SalePrice, Length: 1095, dtype: int64
```

```
y_test
```

```
892      154500
1105     325000
413      115000
522      159000
1036     315500
          ...
988      195000
243      120000
1342     228500
1057     248000
1418     124000
Name: SalePrice, Length: 365, dtype: int64
```

Random Forest Classifier

```
random=RandomForestClassifier(n_jobs=-1,random_state=42,n_estimators=10)
```

```
random.fit(x_train,y_train)
```

```
RandomForestClassifier(n_estimators=10, n_jobs=-1, random_state=42)
```

```
y_pred=random.predict(x_test)
```

Making predictions and evaluating the model

The model's performance will be evaluated using **RMSE (Root Mean Squared Error)** loss function.
It is one of the main performance indicator for a regression model. It measures the average difference between values predicted by a model and the actual values. It provides an estimation of how well the model is able to predict the target value (accuracy).

```
from sklearn.metrics import mean_squared_error
```

```
train_pred=random.predict(x_train)
```

```
train_rmse=random.score(x_train,y_train)
```

```
train_rmse #train attributes(train_inputs and train targets)
```

```
0.9963470319634703
```

```
print('The RMSE loss for the training set is $ {}.'.format(train_rmse))
```

```
The RMSE loss for the training set is $ 0.9963470319634703.
```

```
val_pred=y_test
```

```
val_rmse=random.score(x_test,y_test)
```

```
val_rmse #target [validation] attributes (val_inputs and val_targets)
```

```
0.005479452054794521
```

```
print('The RMSE loss for the validation set is $ {}.'.format(val_rmse))
```

```
The RMSE loss for the validation set is $ 0.005479452054794521.
```

```
jovian.commit()
```

```
[jovian] Updating notebook "kavinm642/house-prices-prediction" on https://jovian.com
[jovian] Committed successfully! https://jovian.com/kavinm642/house-prices-prediction
```

```
'https://jovian.com/kavinm642/house-prices-prediction'
```

Making Predictions

```
def predict_input(single_input):
    input_df = pd.DataFrame([single_input])
    input_df[numeric_col] = imputer.transform(input_df[numeric_col])
    input_df[numeric_col] = scaler.transform(input_df[numeric_col])
    input_df[encoded_col] = encoder.transform(input_df[categoric_col].values)
    X_input = input_df[numeric_col + encoded_col]
    return random.predict(X_input)[0]
```

```
sample_input = { 'MSSubClass': 20, 'MSZoning': 'RL', 'LotFrontage': 77.0, 'LotArea': 93
 'Street': 'Pave', 'Alley': None, 'LotShape': 'IR1', 'LandContour': 'Lvl', 'Utilities':
 'LotConfig': 'Inside', 'LandSlope': 'Gtl', 'Neighborhood': 'NAmes', 'Condition1': 'Nor
 'BldgType': '1Fam', 'HouseStyle': '1Story', 'OverallQual': 4, 'OverallCond': 5, 'YearB
 'YearRemodAdd': 1959, 'RoofStyle': 'Gable', 'RoofMatl': 'CompShg', 'Exterior1st': 'Ply
 'Exterior2nd': 'Plywood', 'MasVnrType': 'None','MasVnrArea': 0.0,'ExterQual': 'TA','Ex
 'Foundation': 'CBlock','BsmtQual': 'TA','BsmtCond': 'TA','BsmtExposure': 'No','BsmtFin
 'BsmtFinSF1': 569,'BsmtFinType2': 'Unf','BsmtFinSF2': 0,'BsmtUnfSF': 381,
 'TotalBsmtSF': 950,'Heating': 'GasA','HeatingQC': 'Fa','CentralAir': 'Y','Electrical':
 '2ndFlrSF': 0, 'LowQualFinSF': 0, 'GrLivArea': 1225, 'BsmtFullBath': 1, 'BsmtHalfBath'
 'HalfBath': 1, 'BedroomAbvGr': 3, 'KitchenAbvGr': 1,'KitchenQual': 'TA','TotRmsAbvGrd'
```

```
'Fireplaces': 0,'FireplaceQu': np.nan,'GarageType': np.nan,'GarageYrBlt': np.nan,'Gara
'GarageArea': 0,'GarageQual': np.nan,'GarageCond': np.nan,'PavedDrive': 'Y', 'WoodDeck
'EnclosedPorch': 0,'3SsnPorch': 0, 'ScreenPorch': 0, 'PoolArea': 0, 'PoolQC': np.nan,
'MiscVal': 400, 'MoSold': 1, 'YrSold': 2010, 'SaleType': 'WD', 'SaleCondition': 'Norma
```

```
predicted_price=predict_input(sample_input)
```

```
/opt/conda/lib/python3.9/site-packages/sklearn/base.py:445: UserWarning:

X does not have valid feature names, but OneHotEncoder was fitted with feature names

/opt/conda/lib/python3.9/site-packages/pandas/core/frame.py:3678: PerformanceWarning:

DataFrame is highly fragmented.  This is usually the result of calling `frame.insert`
many times, which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead.  To get a de-fragmented frame, use `newframe = frame.copy()`
```

```
print('The predicted sale price of the house is ${}.'.format(predicted_price))
```

```
The predicted sale price of the house is $52500.
```

Conclusion

Thus,the final price of a house is **$52500**. However the cost will vary according to the changes committing in `sample_input`

```
jovian.commit()
```

```
[jovian] Updating notebook "kavinm642/house-prices-prediction" on https://jovian.com
[jovian] Committed successfully! https://jovian.com/kavinm642/house-prices-prediction
```

```
'https://jovian.com/kavinm642/house-prices-prediction'
```