**Introduction**

House price prediction can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house. There are three factors that influence the price of a house which include physical conditions, concept and location. Here we have to study about Selection and training of the dataset.

A property’s value is important in real estate transactions. Housing price trends are not only the concern of buyers and sellers, but it also indicates the current economic situation. Therefore, it is important to predict housing prices without bias to help both the buyers and sellers make their decisions. This project development may help to predict the house price.

|  |  |  |
| --- | --- | --- |
| **1** | Id | To count the records. |
| **2** | MSSubClass | Identifies the type of dwelling involved in the sale. |
| **3** | MSZoning | Identifies the general zoning classification of the sale. |
| **4** | LotArea | Lot size in square feet. |
| **5** | LotConfig | Configuration of the lot |
| **6** | BldgType | Type of dwelling |
| **7** | OverallCond | Rates the overall condition of the house |
| **8** | YearBuilt | Original construction year |
| **9** | YearRemodAdd | Remodel date (same as construction date if no remodeling or additions). |
| **10** | Exterior1st | Exterior covering on house |
| **11** | BsmtFinSF2 | Type 2 finished square feet. |
| **12** | TotalBsmtSF | Total square feet of basement area |
| **13** | SalePrice | To be predicted |

FEATURE SELECTION :

Code:

*# Import starting libraries*

import numpy as np

import pandas as pd

import seaborn as sns

import

*# Separate temporal features*

feature\_with\_year = []

for feature **in** X\_train.columns:

if "Yr" **in** feature **or** "Year" **in** feature:

feature\_with\_year.append(feature)

*# Separate numerical and categorial features*

categorical\_features = []

numerical\_features = []

discrete\_features = []

continuous\_features = []

for feature **in** X\_train.columns:

if X\_train[feature].dtypes == "O":

categorical\_features.append(feature)

else:

numerical\_features.append(feature)

if len(X\_train[feature].unique()) <= 20 **and** feature **not** **in** feature\_with\_year:

discrete\_features.append(feature)

else:

continuous\_features.append(feature)

*# Separate numerical and categorial features*

categorical\_features = []

numerical\_features = []discrete\_features = []

continuous\_features = []

for feature **in** X\_train.columns:

if X\_train[feature].dtypes == "O":

categorical\_features.append(feature)

else:

numerical\_features.append(feature)

if len(X\_train[feature].unique()) <= 20 **and** feature **not** **in** feature\_with\_yerr:

discrete\_features.append(feature)

else:

continuous\_features.append(feature)

OUTPUT:

Numerical Features ['Id', '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', 'SalePrice']

Discrete Features ['MSSubClass', 'OverallQual', 'OverallCond', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageCars', '3SsnPorch', 'PoolArea', 'MoSold']

Continuous Features ['Id', 'LotFrontage', 'LotArea', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea', 'GarageYrBlt', 'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', 'ScreenPorch', 'MiscVal', 'YrSold', 'SalePrice']

CODE:

*# Mutual information on Numerical Input*

from sklearn.feature\_selection import mutual\_info\_regression

y\_train = X\_train['SalePrice']

final\_columns = discrete\_features

for i **in** continuous\_features:

if i **not** **in** feature\_with\_year:

final\_columns.append(i)

print(final\_columns)

mi\_scores = mutual\_info\_regression(X\_train[final\_columns], y\_train)

mi\_scores = pd.Series(mi\_scores, name="MI Scores", index=final\_columns)

mi\_scores = mi\_scores.sort\_values(ascending=False)

mi\_scores

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

OUTPUT:

SalePrice 5.453403

OverallQual 0.543599

GrLivArea 0.420799

GarageCars 0.353652

GarageArea 0.333527

TotalBsmtSF 0.323180

1stFlrSF 0.265578

FullBath 0.258839

MSSubClass 0.246294

2ndFlrSF 0.183458

LotFrontage 0.181488

TotRmsAbvGrd 0.174985

LotArea 0.168340

Fireplaces 0.162931

OpenPorchSF 0.138124

BsmtFinSF1 0.131653

BsmtUnfSF 0.116379

WoodDeckSF 0.089007

HalfBath 0.075320

OverallCond 0.075303

BedroomAbvGr 0.062036

MasVnrArea 0.043270

BsmtFullBath 0.036779

ScreenPorch 0.021023

LowQualFinSF 0.019126

EnclosedPorch 0.016887

BsmtFinSF2 0.008468

KitchenAbvGr 0.002491

PoolArea 0.002446

BsmtHalfBath 0.001572

MoSold 0.000000

3SsnPorch 0.000000

Id 0.000000

MiscVal 0.000000

Name: MI Scores, dtype: float

TRAINING AND TESTING OF DATA

Training data is an extremely large dataset that is used to teach a

Machine learning model. Training data is used to teach prediction models

That use machine learning algorithms how to extract features that are

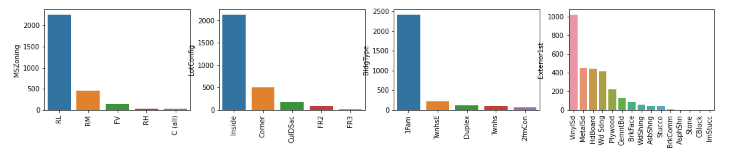
Relevant to specific business goals.

The test data set used to provide an unbiased evaluation of a final model fit on the training data set. If the data in the test data set has never been used in training , the test data set is also called a holdout dataset.

|  |
| --- |
| 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) |

|  |
| --- |
| 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:**



**Conclusion:**

This House price prediction project help us to predict the price of the house and detecting the quality of the house. By including some features we have able to measure the price approximately not be the decimal categorization.