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

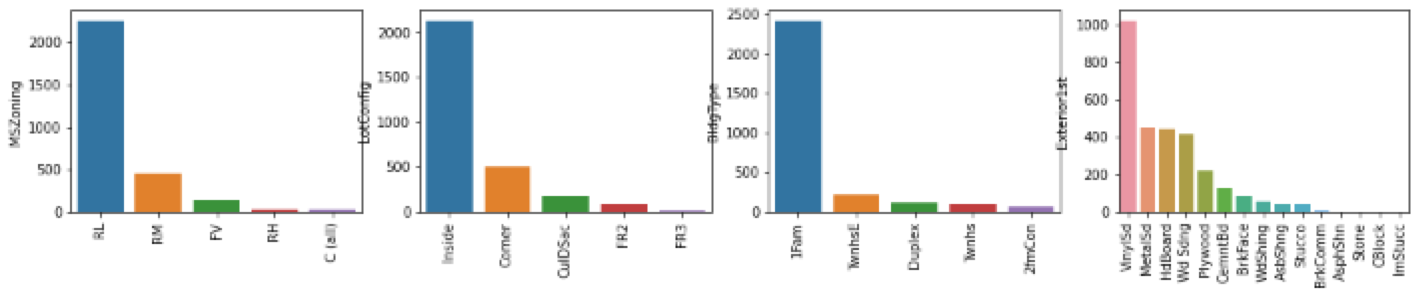
Training data is an extremely large dataset that is used to teach a Machine learning model. Training data is used to teach prediction models That use machine learning algorithms how to extract features that are Relevant to specific business goals.

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.