**Naan Mudhalvan**

**IBM project**

**Artificial Intelligence**

**Topic- House Price Prediction using Machine**

**Learning in python(Phase 4)**

**TOPIC:**

* Importing Libraries and Dataset
* Data Preprocessing
* Exploratory Data Analysis
* Model Building
* Model Evaluation

The dataset contains **13 features** :

|  |  |  |
| --- | --- | --- |
| **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 |

## **Importing Libraries and Dataset :**

Here we are using

* [**Pandas**](https://www.geeksforgeeks.org/python-pandas-dataframe/) **–** To load the Dataframe
* [**Matplotlib**](https://www.geeksforgeeks.org/matplotlib-tutorial/) **–** To visualize the data features i.e. barplot
* [**Seaborn**](https://www.geeksforgeeks.org/introduction-to-seaborn-python/) **–** To see the correlation between features using heatmap

DATA PREPROCESSING :

Data preprocessing is a predominant step in machine learning to yield highly accurate and insightful results. Greater the quality of data, greater is the reliance on the produced results. **Incomplete, noisy, and inconsistent data** are the properties of large real-world datasets. Data preprocessing helps in increasing the quality of data by filling in missing incomplete data, smoothing noise and resolving inconsistencies.

# Data Cleaning :

def find\_missing\_percent(data):  
 *"""*  
 *Returns dataframe containing the total missing values and percentage of total*  
 *missing values of a column.*  
 *"""*  
 miss\_df = pd.DataFrame({'ColumnName':[],'TotalMissingVals':[],'PercentMissing':[]})  
 for col **in** data.columns:  
 sum\_miss\_val = data[col].isnull().sum()  
 percent\_miss\_val = round((sum\_miss\_val/data.shape[0])\*100,2)  
 miss\_df = miss\_df.append(dict(zip(miss\_df.columns,[col,sum\_miss\_val,percent\_miss\_val])),ignore\_index=True)  
 return miss\_df

# IMPORT THE REQUIRED LIBRARIES

In [1]:

import warnings  
warnings.filterwarnings('ignore')  
import numpy as np   
import pandas as pd   
import matplotlib.pyplot as plt  
from operator import itemgetter  
from sklearn.experimental import enable\_iterative\_imputer   
from sklearn.impute import IterativeImputer  
from sklearn.preprocessing import OrdinalEncoder  
from category\_encoders.target\_encoder import TargetEncoder  
from sklearn.preprocessing import StandardScaler  
from sklearn.ensemble import (GradientBoostingRegressor, GradientBoostingClassifier)  
import xgboost

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial step in the process of house price prediction using machine learning in Python. EDA involves analyzing and visualizing the data to identify patterns, trends, and anomalies.

# Data Modeling :

def fit\_model(x\_train,y\_train, model):  
 *"""*  
 *Fits x\_train to y\_train for the given*  
 *model.*  
 *"""*  
 model.fit(x\_train,y\_train)  
 return model  
  
'''Xtreme Gradient Boosting Regressor'''  
model = xgboost.XGBRegressor(objective="reg:squarederror", random\_state=42)  
model = fit\_model(x\_train,y\_train, model)  
'''Predict the outcomes'''  
predictions = model.predict(test)

### **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.

|  |
| --- |
| # 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 |

OUTPUT:

0.893643437976127