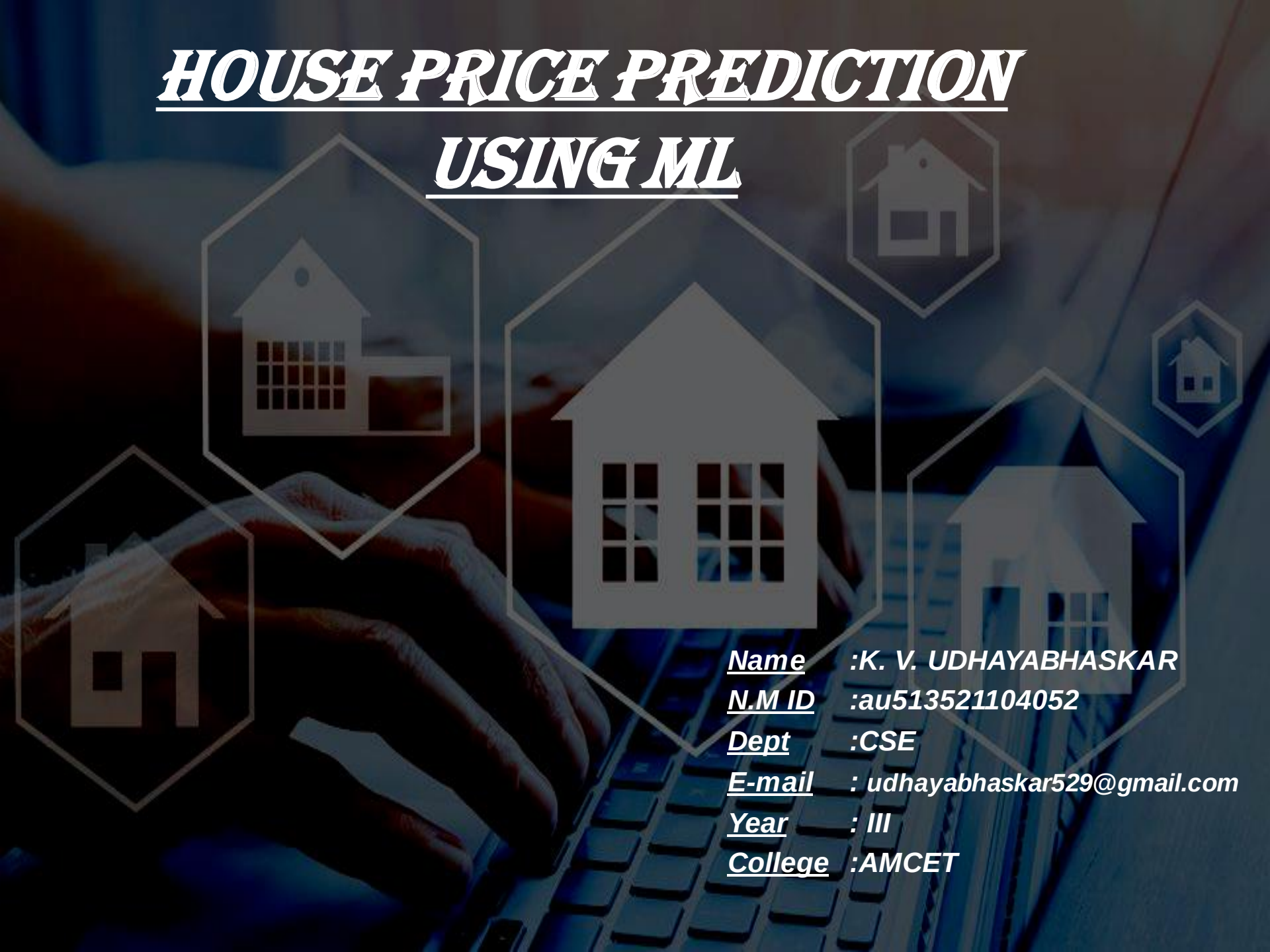
HOUSE PRICE PREDICTION USING ML



Name *:K. V. UDHAYABHASKAR*

N.M ID *:au513521104052*

Dept *:CSE*

E-mail *: udhayabhaskar529@gmail.com*

Year *: III*

College *:AMCET*

Introduction

- ***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.

  - ***Data Cleaning***
  - ***Data visualization***
  - ***Data Transformation***
  - ***Data Reduction***

Libraries

- *House price prediction using machine learning(Linear regression model) We have used Kaggle kc_house_data dataset. Import the required libraries and modules, including pandas for data manipulation, scikit-learn for machine learning algorithms, and Linear regression for the linear regression model.*

```
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
```

Data Cleaning

- *Data cleaning can be applied to filling in missing values, remove noise, resolving inconsistencies, identifying and removing outliers in the data.*
- *Data integration merges data from multiple sources into a coherent data store, such as a data warehouse.*
- *Data transformations, such as normalization, may be applied. For example, normalization may improve the accuracy and efficiency of mining algorithms involving distance measurements.*
- *Data reduction can reduce the data size by eliminating redundant features, or clustering, for instance.*
- *Reference: Data Mining:Concepts and Techniques Second Edition, Jiawei Han, Micheline Kamber.*

Find the missing percentage of each columns in training set

```
def find_missing_percent(data):
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_v
       al])),ignore_index=True)
    return miss_df
miss_df = find_missing_percent(train)
display(miss_df[miss_df['PercentMissing']>0.0])
print("\n")
print(f"Number of columns with missing
       values:{str(miss_df[miss_df['PercentMissing']>0.0].shape[0])}")
```

Column	Name	TotalMissingVals	PercentMissing
3	LotFrontage	259.0	17.74
6	Alley	1369.0	93.77
25	MasVnrType	8.0	0.55
26	MasVnrArea	8.0	0.55
30	BsmtQual	37.0	2.53
31	BsmtCond	37.0	2.53
32	BsmtExposure	38.0	2.60
33	BsmtFinType1	37.0	2.53
35	BsmtFinType2	38.0	2.60
42	Electrical	1.0	0.07
57	FireplaceQu	690.0	47.26
58	GarageType	81.0	5.55
59	GarageYrBlt	81.0	5.55
60	GarageFinish	81.0	5.55
63	GarageQual	81.0	5.55
64	GarageCond	81.0	5.55
72	PoolQC	1453.0	99.52
73	Fence	1179.0	80.75
74	MiscFeature	1406.0	96.30

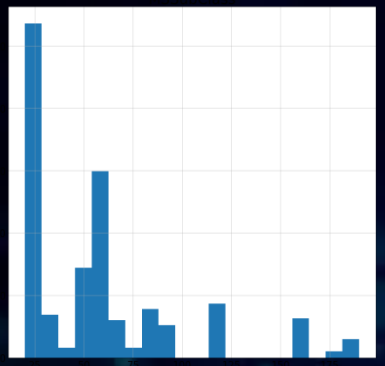
Number of columns with missing values:19

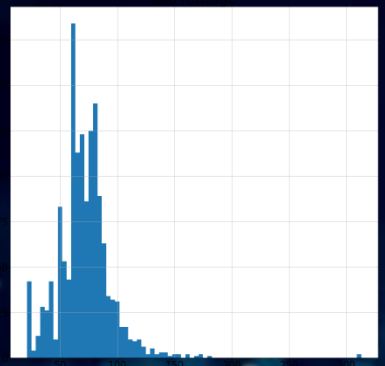
Data visualization

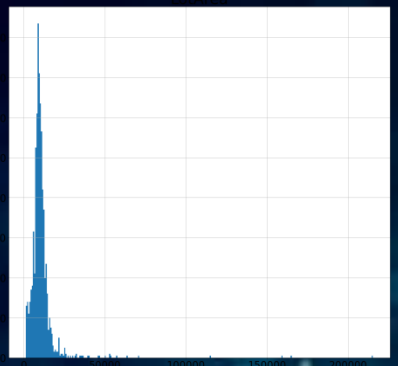
```
def plot_histogram(train, col1, col2, cols_list, last_one =False):
    Freedman-Diaconis Rule:
  Freedman-Diaconis Rule is a rule to find the optimal number of bins.
  Bin width: (2 * IQR)/(N^1/3)
  N - Size of the data
  Number of bins : (Range/ bin-width)
  Sturges Rule:
  Sturges Rule is a rule to find the optimal number of bins.
  Bin width: (Range/ bin-width)
  N - Size of the data
  Number of bins : ceil(log2(N))+1
if(col1 in cols_list):
    freq1, bin_edges1 = np.histogram(train[col1],bins='sturges')
  else:
    freq1, bin_edges1 = np.histogram(train[col1],bins='fd')
  if(col2 in cols_list):
    freq2, bin_edges2 = np.histogram(train[col2],bins='sturges')
  else:
    freq2, bin_edges2 = np.histogram(train[col2],bins='fd')
if(last_one!=True):
    plt.figure(figsize=(45,18))
    ax1 = plt.subplot(1,2,1)
    ax1.set_title(col1,fontsize=45)
    ax1.set_xlabel(col1,fontsize=40)
    ax1.set_ylabel('Frequency',fontsize=40)
    train[col1].hist(bins=bin_edges1,ax = ax1, xlabelsize=30, ylabelsize=30)
```

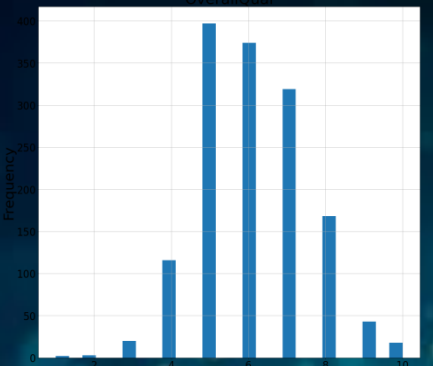
```
    else:
        plt.figure(figsize=(20,10))
        ax1 = plt.subplot(1,2,1)
        ax1.set_title(col1,fontsize=25)
        ax1.set_xlabel(col1,fontsize=20)
        ax1.set_ylabel('Frequency',fontsize=20)
        train[col1].hist(bins=bin_edges1,ax  = ax1, xlabelsize=15, ylabelsize=15)

    if(last_one != True):
        ax2 = plt.subplot(1,2,2)
        ax2.set_title(col2,fontsize=45)
        ax2.set_xlabel(col2,fontsize=40)
        ax2.set_ylabel('Frequency',fontsize=40)
        train[col2].hist(bins=bin_edges2,  ax = ax2, xlabelsize=30, ylabelsize=30)
cols_list = ['LowQualFinSF','BsmtFinSF2','BsmtHalfBath','KitchenAbvGr',
            'EnclosedPorch','3SsnPorch','ScreenPorch','PoolArea','MiscVal']

# Except ID
hist_cols = numeric_cols[1:]
for i in range(0,len(hist_cols),2):
    if(i == len(hist_cols)-1):
        plot_histogram(train,hist_cols[i],hist_cols[i],cols_list,True)
    else:
        plot_histogram(train,hist_cols[i],hist_cols[i+1],cols_list)
```

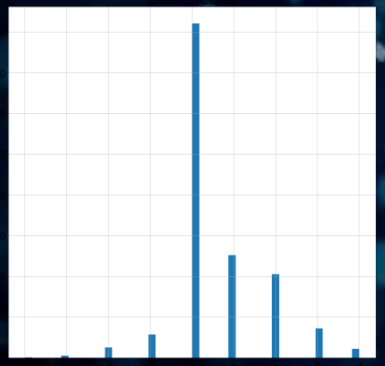
Output

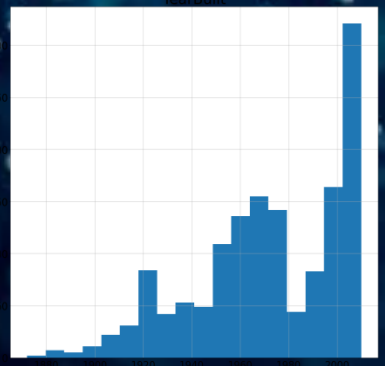

MSSubClass

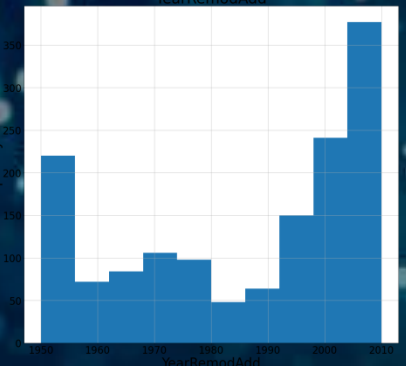

LotFrontage
Frequency

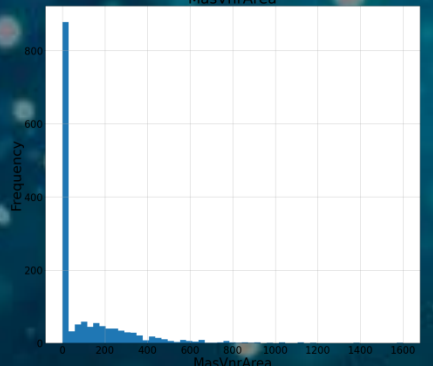

LotArea
Frequency

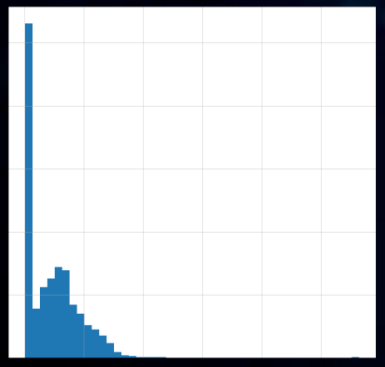

OverallQual
Frequency

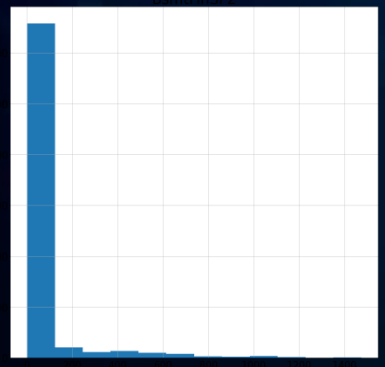

OverallCond

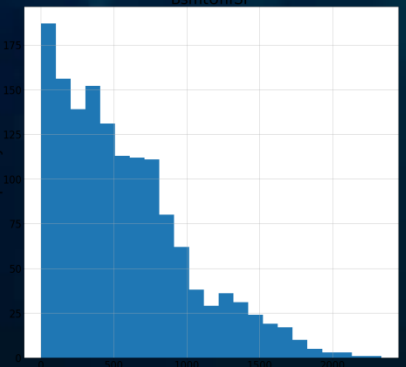

YearBuilt
Frequency

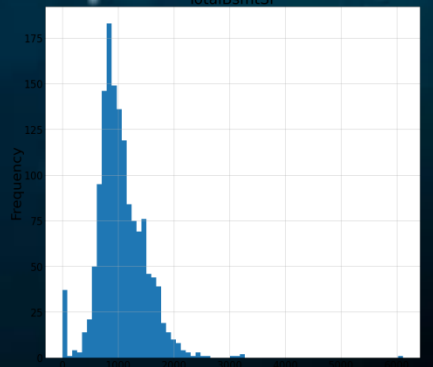

YearRemodAdd
Frequency


MasVnrArea
Frequency



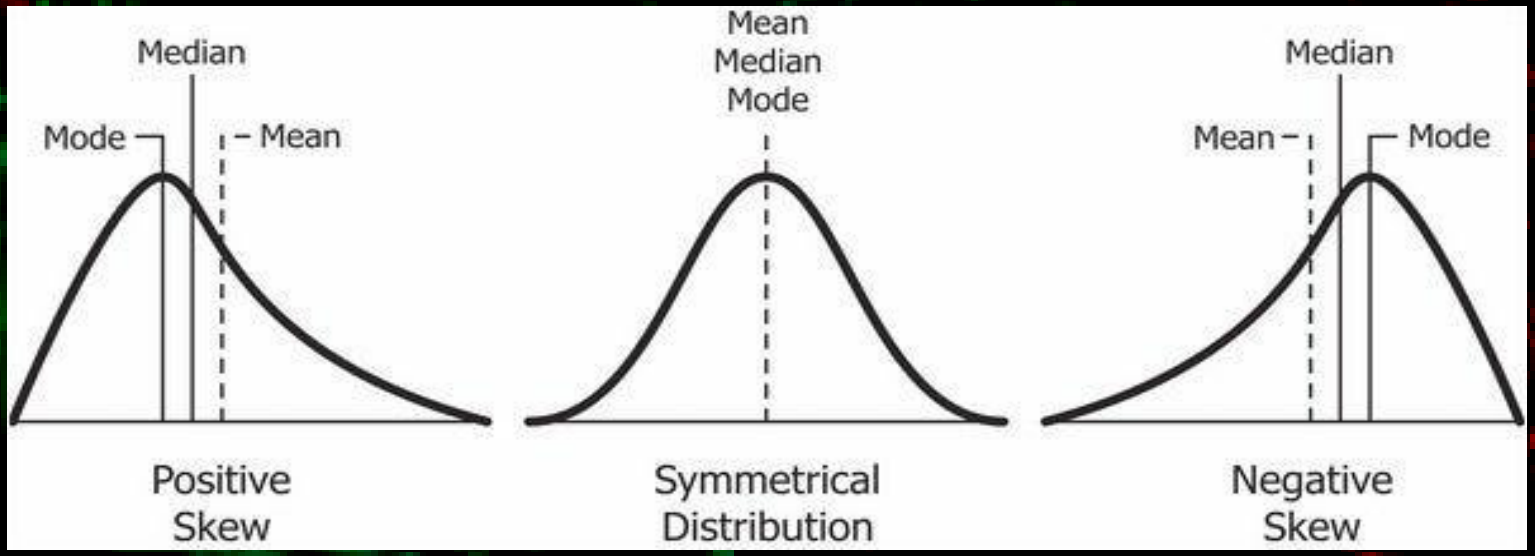

BsmtFinSF2


BsmtUnfSF
Frequency


TotalBsmtSF
Frequency

Data Transformation

- ***Humans can look at photos, listen to audio, and read text without needing to consciously think about individual pixels, waveforms, or words. Computers, on the other hand, are not so good at understanding this type of data. In order to work with data on a computer, it must be transformed into a format that the machine can understand. This process is known as data transformation.***



```
def find_skewness(train, numeric_cols):
    """
    Calculate the skewness of the columns and segregate the positive
    and negative skewed data.
    """
    skew_dict = {}
    for col in numeric_cols:
        skew_dict[col] = train[col].skew()
skew_dict = dict(sorted(skew_dict.items(),key=itemgetter(1)))
    positive_skew_dict = {k:v for (k,v) in skew_dict.items() if v>0}
    negative_skew_dict = {k:v for (k,v) in skew_dict.items() if v<0}
    return skew_dict, positive_skew_dict, negative_skew_dict
def add_constant(data, highly_pos_skewed):
   C = 1
    for col in highly_pos_skewed.keys():
        if(col != 'SalePrice'):
            if(len(data[data[col] == 0]) > 0):
                data[col] = data[col] + C
    return data
def log_transform(data, highly_pos_skewed):
     for col in highly_pos_skewed.keys():
        if(col != 'SalePrice'):
            data[col] = np.log10(data[col])
    return data
def sqrt_transform(data, moderately_pos_skewed):
      for col in moderately_pos_skewed.keys():
        if(col != 'SalePrice'):
            data[col] = np.sqrt(data[col])
    return data
```

```
def reflect_sqrt_transform(data, moderately_neg_skewed):
    """
    Reflection and log transformation of highly negatively skewed
    columns.
    """
    for col in moderately_neg_skewed.keys():
        if(col != 'SalePrice'):
            K = max(data[col]) + 1
            data[col] = np.sqrt(K - data[col])
    return data
"""def find_skewness(train, numeric_cols):
    """
    Calculate the skewness  of the columns and segregate the positive
    and negative skewed data.
    """
    skew_dict = {}
    for col in numeric_cols:
        skew_dict[col] = train[col].skew()

    skew_dict = dict(sorted(skew_dict.items(),key=itemgetter(1)))
    positive_skew_dict = {k:v for (k,v) in skew_dict.items() if v>0}
    negative_skew_dict = {k:v for (k,v) in skew_dict.items() if v<0}
    return skew_dict, positive_skew_dict, negative_skew_dict

def add_constant(data, highly_pos_skewed):
    """
    Look for zeros in the columns. If zeros are present then the log(0) would result in -infinity.
    So before transforming it we need to add it with some constant.
    """
    C = 1
    for col in highly_pos_skewed.keys():
        if(col != 'SalePrice'):
```

```
            if(len(data[data[col] == 0]) > 0):
                data[col] = data[col] + C
        return data

    def log_transform(data, highly_pos_skewed):
        """
        Log transformation of highly positively skewed columns.
        """
        for col in highly_pos_skewed.keys():
            if(col != 'SalePrice'):
                data[col] = np.log10(data[col])
        return data

    def sqrt_transform(data, moderately_pos_skewed):
        """
        Square root transformation of moderately skewed columns.
        """
        for col in moderately_pos_skewed.keys():
            if(col != 'SalePrice'):
                data[col] = np.sqrt(data[col])
        return data

    def reflect_sqrt_transform(data, moderately_neg_skewed):
        """
        Reflection and log transformation of highly negatively skewed
        columns.
        """
        for col in moderately_neg_skewed.keys():
            if(col != 'SalePrice'):
                K = max(data[col]) + 1
                data[col] = np.sqrt(K - data[col])
        return data
```

Data Modeling:

- *A data model is an abstract model that organizes elements of data and standardizes how they relate to one another and to the properties of real-world entities. For instance, a data model may specify that the data element representing a car should be composed of several other elements which, in turn, represent the color and size of the car and define its owner.*

```
Fit XGBoost Regressor model to the preprocessed data.
def fit_model(x_train,y_train, model):
    Fits x_train to y_train for the given
    model.
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)
submission = pd.read_csv('../input/house-prices-advanced-regression-techniques/sample_submission.csv')
submission['SalePrice'] = predictions
submission.to_csv('submission.csv',index=False)
```


THANK YOU