In [68]:

```
import pandas as pd
import numpy as np
import warnings
```

In [69]:

```
df_train = pd.read_csv('../datasets/house train.csv')
```

In [70]:

```
df_train
```

Out[70]:

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

1460 rows × 81 columns

In [71]:

```
df_train.columns
```

Out[71]:

```
Index(['Id', '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', 'SalePrice'],
      dtype='object')
```

In [72]:

```
df_train.info()
```

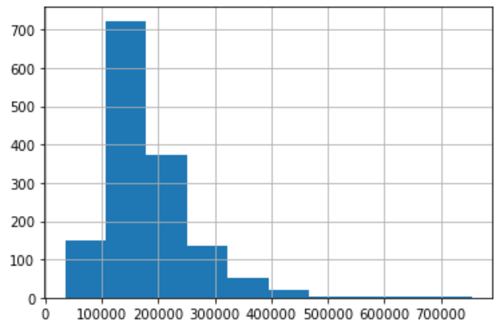
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             1460 non-null   int64  
 1   MSSubClass     1460 non-null   int64  
 2   MSZoning       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 
```

In [73]:

```
df_train['SalePrice'].hist()
```

Out[73]:

```
<AxesSubplot:>
```

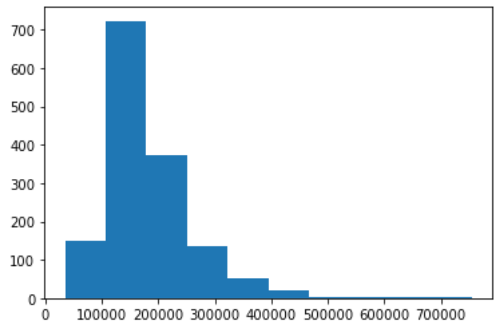


In [74]:

```
plt.hist(df_train['SalePrice'])
```

Out[74]:

```
(array([148., 723., 373., 135.,  51.,  19.,   4.,   3.,   2.,   2.]),
 array([ 34900., 106910., 178920., 250930., 322940., 394950., 466960.,
        538970., 610980., 682990., 755000.]),
 <BarContainer object of 10 artists>)
```



In [75]:

```
df_train['SalePrice'].describe()
```

Out[75]:

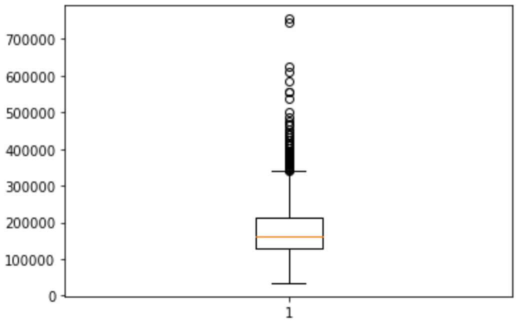
```
count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

In [76]:

```
plt.boxplot(df_train['SalePrice'])
```

Out[76]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x134d6a9a880>,
  <matplotlib.lines.Line2D at 0x134d6a9abe0>],
 'caps': [<matplotlib.lines.Line2D at 0x134d6a9af40>,
  <matplotlib.lines.Line2D at 0x134d6aa42e0>],
 'boxes': [<matplotlib.lines.Line2D at 0x134d6a9a520>],
 'medians': [<matplotlib.lines.Line2D at 0x134d6aa4670>],
 'fliers': [<matplotlib.lines.Line2D at 0x134d6aa49d0>],
 'means': []}
```



In [77]:

```
z_scores = (df_train['SalePrice']-df_train['SalePrice'].mean())/df_train['SalePrice'].std()
df_train = df_train[((z_scores >-3) & (z_scores <=3))]
```

In [78]:

```
z_scores
```

Out[78]:

```
0       0.347154
1       0.007286
2       0.535970
3      -0.515105
4       0.869545
          ...
1455   -0.074534
1456    0.366036
1457    1.077242
1458   -0.488356
1459   -0.420697
Name: SalePrice, Length: 1460, dtype: float64
```

In [79]:

```
df_train.head()
```

Out[79]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	0
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0

5 rows × 81 columns

In [80]:

```
df_train['Alley'].value_counts()
```

Out[80]:

```
Grvl    50
Pave    41
Name: Alley, dtype: int64
```

In [81]:

```
list1=['Alley','BsmtQual','BsmtCond', 'BsmtExposure', 'BsmtFinType1','BsmtFinType2','FireplaceQu','GarageType',
       'GarageFinish','GarageQual','GarageCond','Fence','MiscFeature','PoolQC']
```

In [82]:

```
dict1 = {}
for i in df_train.columns.drop('SalePrice'):
    if i not in list1:
        dict1[i] = ['', '#N/A', '#N/A N/A', '#NA', '-1.#IND', '-1.#QNAN', '-NaN', '-nan',
                    '1.#IND', '1.#QNAN', '<NA>', 'N/A', 'NA', 'NULL', 'NaN', 'n/a','nan', 'null']
    else:
        dict1[i] = [', '#N/A', '#N/A N/A', '#NA', '-1.#IND', '-1.#QNAN', '-NaN', '-nan',
                    '1.#IND', '1.#QNAN', '<NA>', 'N/A', 'NULL', 'NaN', 'n/a','nan', 'null']
len(dict1)
```

Out[82]:

```
80
```

In [83]:

```
df_train = pd.read_csv('../datasets/house train.csv',keep_default_na=False,na_values=dict1)

df_test = pd.read_csv('../datasets/house test.csv',keep_default_na=False,na_values=dict1)
```

In [84]:

```
df_train.isna().sum()[df_train.isna().sum()>0]
```

Out[84]:

```
LotFrontage    259
MasVnrType       8
MasVnrArea       8
Electrical       1
GarageYrBlt     81
dtype: int64
```

In [85]:

```
df_test.isna().sum()[df_test.isna().sum()>0]
```

Out[85]:

```
MSZoning          4
LotFrontage     227
Utilities         2
Exterior1st       1
Exterior2nd       1
MasVnrType       16
MasVnrArea       15
BsmtFinSF1        1
BsmtFinSF2        1
BsmtUnfSF         1
TotalBsmtSF       1
BsmtFullBath      2
BsmtHalfBath      2
KitchenQual       1
Functional        2
GarageYrBlt      78
GarageCars        1
GarageArea        1
SaleType          1
dtype: int64
```

In [86]:

```
columns_cat = list(df_train.select_dtypes(include='object').columns)
columns_num = list(df_train.select_dtypes(exclude='object').columns)
columns_num.remove('Id')
columns_num.remove('SalePrice')
```

In [87]:

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

In [88]:

```
imputer_num = SimpleImputer(strategy='median')
imputer_cat = SimpleImputer(strategy='most_frequent')

imputer_num.fit(df_train[columns_num])
imputer_cat.fit(df_train[columns_cat])

df_train[columns_num] = imputer_num.transform(df_train[columns_num])
df_train[columns_cat] = imputer_cat.transform(df_train[columns_cat])

df_test[columns_num] = imputer_num.transform(df_test[columns_num])
df_test[columns_cat] = imputer_cat.transform(df_test[columns_cat])
```

In [89]:

```
df_train.isna().sum()[df_train.isna().sum()>0]
```

Out[89]:

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

In [90]:

```
df_test.isna().sum()[df_test.isna().sum()>0]
```

Out[90]:

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

In [91]:

```
from sklearn.preprocessing import MinMaxScaler
```

In [92]:

```
scaler = MinMaxScaler()
scaler.fit(df_train[columns_num])
df_train[columns_num] = scaler.transform(df_train[columns_num])
df_test[columns_num] = scaler.transform(df_test[columns_num])
```

In [93]:

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

In [94]:

```
ohe = OneHotEncoder(handle_unknown='ignore')
ohe.fit(df_train[columns_cat])
df_train[ohe.get_feature_names()] = ohe.transform(df_train[columns_cat]).toarray()
df_test[ohe.get_feature_names()] = ohe.transform(df_test[columns_cat]).toarray()
```

In [95]:

```
df_train[ohe.get_feature_names()]
```

Out[95]:

	x0_C (all)	x0_FV	x0_RH	x0_RL	x0_RM	x1_Grvl	x1_Pave	x2_Grvl	x2_NA	x2_Pave	...	x41_ConLw	x41_New	x41_Oth	x41_WD	x42_Abnorml	x42_
0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	
1	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	
2	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	
3	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	1.0	
4	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1455	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	
1456	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	
1457	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	
1458	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	
1459	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0	

1460 rows × 266 columns

In [96]:

```
len(ohe.get_feature_names())
```

Out[96]:

```
266
```

In [97]:

```
ohe.transform(df_train[columns_cat]).toarray()
```

Out[97]:

```
array([[0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       ...,
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 1., 0.]])
```

In [98]:

```
df_train.shape,df_test.shape
```

Out[98]:

```
((1460, 347), (1459, 346))
```

In [99]:

```
column_list = columns_num + list(ohe.get_feature_names()) + ['SalePrice']

corr_values = df_train[column_list].corr()['SalePrice']
corr_values

selected_col = list((corr_values[(corr_values > 0.1) | (corr_values <-0.1)]).index)
selected_col.remove('SalePrice')
```

In [100]:

```
X = df_train[selected_col]
y = df_train['SalePrice']
```

In [101]:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X,y)
yp = model.predict(df_test[selected_col])
df_test['SalePrice'] = yp
df_test[['Id','SalePrice']].to_csv('finalhp1.csv',index = False)
model.score(X,y)
```

Out[101]:

```
0.8886568944585275
```

In [102]:

```
from sklearn.neighbors import KNeighborsRegressor
modelKNN = KNeighborsRegressor(n_neighbors=3)
modelKNN.fit(df_train[columns_num + list(ohe.get_feature_names())],df_train['SalePrice'])
print(modelKNN.score(df_train[columns_num + list(ohe.get_feature_names())],df_train['SalePrice']))
yp = modelKNN.predict(df_test[columns_num + list(ohe.get_feature_names())])
df_test['SalePrice'] = yp
df_test[['Id','SalePrice']].to_csv('subKNN3.csv',index=False)
```

```
0.8672683240814711
```

In [104]:

```
from sklearn.ensemble import RandomForestRegressor

params = {'n_estimators':[20,30,40,50],'max_depth':[2,3,4,5,6],'min_samples_leaf':[3,4,5,6]}
gridCV = GridSearchCV(RandomForestRegressor(random_state=51),
                     param_grid=params,cv=5,verbose=0,scoring='accuracy')

gridCV.fit(df_train[selected_col],df_train['SalePrice'])
```

```
ValueError: Classification metrics can't handle a mix of multiclass and continuous targets

  warnings.warn(
C:\Users\Rishabh jain\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:683: UserWarning: Scoring faile
d. The score on this train-test partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\Rishabh jain\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 674, in _score
    scores = scorer(estimator, X_test, y_test)
  File "C:\Users\Rishabh jain\anaconda3\lib\site-packages\sklearn\metrics\_scorer.py", line 199, in __call__
    return self._score(partial(_cached_call, None), estimator, X, y_true,
  File "C:\Users\Rishabh jain\anaconda3\lib\site-packages\sklearn\metrics\_scorer.py", line 242, in _score
    return self._sign * self._score_func(y_true, y_pred,
  File "C:\Users\Rishabh jain\anaconda3\lib\site-packages\sklearn\utils\validation.py", line 63, in inner_f
    return f(*args, **kwargs)
  File "C:\Users\Rishabh jain\anaconda3\lib\site-packages\sklearn\metrics\_classification.py", line 202, in accuracy_sco
re
    y_type, y_true, y_pred = _check_targets(y_true, y_pred)
  File "C:\Users\Rishabh jain\anaconda3\lib\site-packages\sklearn\metrics\_classification.py", line 92, in _check_target
s
    raise ValueError("Classification metrics can't handle a mix of {0} "
```

In [105]:

```
from sklearn.model_selection import GridSearchCV,RandomizedSearchCV
gridCV.best_params_

modelRF = RandomForestRegressor(max_depth=12, min_samples_leaf=2,random_state=95)

modelRF.fit(df_train[selected_col],df_train['SalePrice'])
print(modelRF.score(df_train[selected_col],df_train['SalePrice']))

yp = modelRF.predict(df_test[selected_col])

df_test['SalePrice'] = yp
df_test[['Id','SalePrice']].to_csv('modelRF3.csv',index=False)
```

```
0.9724212494478934
```

In [106]:

```
from xgboost import XGBRegressor
modelXG = XGBRegressor()
modelXG.fit(df_train[selected_col],df_train['SalePrice'])

yp = modelXG.predict(df_test[selected_col])

df_test['SalePrice'] = yp
df_test[['Id','SalePrice']].to_csv('modelXG2.csv',index=False)

modelXG.score(df_train[selected_col],df_train['SalePrice'])
```

Out[106]:

```
0.9997100838688698
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [48]:

In [49]:

In [50]:

In [ ]: