In [ ]:

```
#### HOUSE PRICE PREDICTION
```

In [ ]:

```
a = 10+10-10
b = 20

 = 30


40+30

50
```

In [ ]:

```
speed , time ....distance



1,2,3....78,79,80
0
1
2
...

543
654
```

STEP 1

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import seaborn as sns

pd.set_option("display.max_rows", None, "display.max_columns", None)

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [2]:

```
df = pd.read_csv('house-price-data.csv')
```

In [3]:

```
df['SaleCondition'].value_counts()
```

Out[3]:

```
Normal     1198
Partial     125
Abnorml     101
Family       20
Alloca       12
AdjLand       4
Name: SaleCondition, dtype: int64
```

In [ ]:

```
ab , ad , al , f , n ,p
```

In [5]:

```
from sklearn.preprocessing import LabelEncoder
```

In [13]:

```
label = LabelEncoder()

x =label.fit_transform(df['SaleCondition'])

xy = pd.get_dummies(df['SaleCondition'])
```

In [14]:

```
xy
```

Out[14]:

	Abnorml	AdjLand	Alloca	Family	Normal	Partial
0	0	0	0	0	1	0
1	0	0	0	0	1	0
2	0	0	0	0	1	0
3	1	0	0	0	0	0
4	0	0	0	0	1	0
5	0	0	0	0	1	0
6	0	0	0	0	1	0
7	0	0	0	0	1	0
8	1	0	0	0	0	0
9	0	0	0	0	1	0

In [ ]:

```
6 ....categories = 0 - 5
```

In [ ]:

```
df['SaleCondition'].value_counts()
```

In [15]:

```
df['SaleCondition']
```

Out[15]:

```
0         Normal
1         Normal
2         Normal
3        Abnorml
4         Normal
5         Normal
6         Normal
7         Normal
8        Abnorml
9         Normal
10        Normal
11       Partial
12        Normal
13       Partial
14        Normal
15        Normal
16        Normal
17        Normal
```

In [ ]:

```
for i in x:
    print(i)
```

In [ ]:

```
df.head(2)
```

In [ ]:

```
df.tail(2)
```

In [ ]:

```
df.shape
```

In [ ]:

```
df['MSZoning'].value_counts()
```

In [ ]:

```
df.isnull().sum()
```

In [ ]:

```
#df.info
```

In [ ]:

```
#df.describe
```

In [ ]:

```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

In [ ]:

```
#df['LotFrontage']
```

In [ ]:

```
df['LotFrontage'].mean()
```

In [ ]:

```

```

STEP 2

In [ ]:

```
df['LotFrontage'] = df['LotFrontage'].fillna(df['LotFrontage'].mean())
```

In [ ]:

```

```

In [ ]:

```
df.drop(['Alley'],axis=1,inplace=True)
```

In [ ]:

```
df.head(2)
```

In [ ]:

```
df.shape
```

In [ ]:

```
1,1,1,1,1,1,2,2,2,3,3
```

In [ ]:

```
df['BsmtCond'].value_counts()
```

In [ ]:

```
df['BsmtCond'].mode()[0]
```

In [ ]:

```
df['BsmtCond'] = df['BsmtCond'].fillna(df['BsmtCond'].mode()[0])
df['BsmtQual'] = df['BsmtQual'].fillna(df['BsmtQual'].mode()[0])
df['FireplaceQu'] = df['FireplaceQu'].fillna(df['FireplaceQu'].mode()[0])
df['GarageType'] = df['GarageType'].fillna(df['GarageType'].mode()[0])
```

In [ ]:

```
df.drop(['GarageYrBlt'],axis=1,inplace=True)
```

In [ ]:

```
df['GarageFinish'] = df['GarageFinish'].fillna(df['GarageFinish'].mode()[0])
df['GarageQual'] = df['GarageQual'].fillna(df['GarageQual'].mode()[0])
df['GarageCond'] = df['GarageCond'].fillna(df['GarageCond'].mode()[0])
#df['GarageType'] = df['GarageType'].fillna(df['GarageType'].mode()[0])
```

In [ ]:

```
df.drop(['PoolQC','Fence','MiscFeature'],axis=1,inplace=True)
```

In [ ]:

```
df.drop(['Id'],axis=1,inplace=True)
```

In [ ]:

```

```

In [ ]:

```
df.shape
```

In [ ]:

```
#df.columns
```

In [ ]:

```
df.isnull().sum()
```

In [ ]:

```
df['MasVnrType'] = df['MasVnrType'].fillna(df['MasVnrType'].mode()[0])
df['MasVnrArea'] = df['MasVnrArea'].fillna(df['MasVnrArea'].mode()[0])
```

In [ ]:

```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='coolwarm')
```

In [ ]:

```
#sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

In [ ]:

```
df['BsmtExposure'] = df['BsmtExposure'].fillna(df['BsmtExposure'].mode()[0])
```

In [ ]:

```
df['BsmtFinType1'] = df['BsmtFinType1'].fillna(df['BsmtFinType1'].mode()[0])
```

In [ ]:

```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

In [ ]:

```
df.dropna(inplace=True)
```

In [ ]:

```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

In [ ]:

```
df.head(2)
```

In [ ]:

```
df.shape
```

In [ ]:

```
#df.isnull().sum()
```

In [ ]:

```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='coolwarm')
```

In [ ]:

```
col = df.columns
```

In [ ]:

```
col
```

In [ ]:

```
len(col)
```

In [ ]:

```
numeric = list(df.dtypes[df.dtypes!='object'].index)
```

In [ ]:

```
numeric
```

In [ ]:

```
len(numeric)
```

In [ ]:

```
#df['YrSold']
```

In [ ]:

```
df1 = df
```

In [ ]:

```
x = 5
```

In [ ]:

```
df.head(2)
```

In [ ]:

```

```

In [ ]:

```
train = df[numeric]
```

In [ ]:

```

```

In [ ]:

```
train.head(2)
```

In [ ]:

```
type(train)
```

In [ ]:

```
#train_scaled
```

In [ ]:

```
train.head(2)
```

In [ ]:

```

```

In [ ]:

```
train.shape
```

In [ ]:

```
train.isnull().sum()
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
#xtest
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
train.head(2)
```

In [ ]:

```
#train.iloc[row_starting_index : row_ending_index , col_starting_index : col_ending_
```

In [ ]:

```
price = train.iloc[:,35:36] #OUTPUT

#price = train['SalePrice']
price.head(2)
```

In [ ]:

```
train.head(2)
```

In [ ]:

```
train = train.iloc[:,:-1]  #INPUT
train.head(2)
```

In [ ]:

```
100% ....80% training 20% test
```

In [ ]:

```
input = 100
output = 100


training_input = 80    #xtrain
training_output = 80   #ytrain

test_input = 20    #xtest
testing_output = 20  #ytest



#random state

100 ....

80 training data ...randomly selected data out of 100
20 testing data


80 ... model accuracy = 85
20 ...  model accuracy = 75
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
xtrain,xtest,ytrain,ytest = train_test_split(train,
                                             price,
                                             test_size=0.2,
                                             random_state=40)
```

In [ ]:

```
print(len(xtrain), len(ytrain),len(xtest),len(ytest))
```

In [ ]:

```
xtrain.head(10)
```

In [ ]:

```
#ytrain
```

In [ ]:

```
#xtest
```

In [ ]:

```
#ytest
```

In [ ]:

```
print(len(xtrain), len(ytrain),len(xtest),len(ytest))
```

In [ ]:

```
ytrain.head(2)
```

In [ ]:

```
len(xtrain)
```

In [ ]:

```
xtrain.head(2)
```

In [ ]:

```
xtrain.head()
```

STEP 3

In [ ]:

```
# PREPROCESSING ##################
```

In [ ]:

```
from sklearn.preprocessing import StandardScaler
```

In [ ]:

```
scaler = StandardScaler()
```

In [ ]:

```
###INPUT DATA TRANSFORMATION
```

In [ ]:

```
xtrain_scaled = scaler.fit_transform(xtrain)
```

In [ ]:

```
xtrain_scaled
```

In [ ]:

```
#xtrain_scaled
```

In [ ]:

```
xtest_scaled = scaler.transform(xtest)
```

In [ ]:

```
#ytest
```

In [ ]:

```
### FIT TRANSFORM - Training Data
### TRANSFORM - Testing Data
```

In [ ]:

```
regressor = LinearRegression()
```

In [ ]:

```
regressor.fit(xtrain_scaled,ytrain)
```

In [ ]:

```
 regressor.score(xtrain_scaled,ytrain)
```

In [ ]:

```
 regressor.score(xtest_scaled,ytest)
```

In [ ]:

```
reg = regressor.predict(xtest_scaled)
```

In [ ]:

```
reg.shape
```

In [ ]:

```
ytest_li = []
for i in ytest.SalePrice:
    ytest_li.append(i)
```

In [ ]:

```
for i in range(len(reg)):
    print(reg[i] , ytest_li[i])
```

In [ ]:

```

```

In [ ]:

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

In [ ]:

```
ytest_scaled = scaler.fit_transform(ytest)
```

In [ ]:

```
reg_scaled = scaler.transform(reg)
```

In [ ]:

```
reg_scaled
```

In [ ]:

```
#mean_squared_error(ytest,reg) #mean_squared_error
```

In [ ]:

```
mean_squared_error(ytest_scaled,reg_scaled) #mean_squared_error
```

In [ ]:

```
mean_squared_error(ytest_scaled,reg_scaled, squared=False) #root_mean_squared_error
```

In [ ]:

```
mean_absolute_error(ytest_scaled,reg_scaled) #mean_absolute_error
```

In [ ]:

```

```

In [ ]:

```
######Test Set
```

In [ ]:

```
test = pd.read_csv('test.csv')
```

In [ ]:

```
test.head(2)
```

In [ ]:

```
test.shape
```

In [ ]:

```

```

In [ ]:

```
test = pd.read_csv('test.csv')

#test.info()

#test.isnull().sum()
test['MSZoning'].value_counts()


test['LotFrontage'] = test['LotFrontage'].fillna(test['LotFrontage'].mean())

test.drop(['Alley'],axis=1,inplace=True)

test['MSZoning'] = test['MSZoning'].fillna(test['MSZoning'].mode()[0])

test['BsmtCond'] = test['BsmtCond'].fillna(test['BsmtCond'].mode()[0])
test['BsmtQual'] = test['BsmtQual'].fillna(test['BsmtQual'].mode()[0])
test['FireplaceQu'] = test['FireplaceQu'].fillna(test['FireplaceQu'].mode()[0])
test['GarageType'] = test['GarageType'].fillna(test['GarageType'].mode()[0])

test.drop(['GarageYrBlt'],axis=1,inplace=True)

test['GarageFinish'] = test['GarageFinish'].fillna(test['GarageFinish'].mode()[0])
test['GarageQual'] = test['GarageQual'].fillna(test['GarageQual'].mode()[0])
test['GarageCond'] = test['GarageCond'].fillna(test['GarageCond'].mode()[0])
test['GarageType'] = test['GarageType'].fillna(test['GarageType'].mode()[0])

test.drop(['PoolQC','Fence','MiscFeature'],axis=1,inplace=True)

test.drop(['Id'],axis=1,inplace=True)

#test.shape

#test.columns

test['MasVnrType'] = test['MasVnrType'].fillna(test['MasVnrType'].mode()[0])
test['MasVnrArea'] = test['MasVnrArea'].fillna(test['MasVnrArea'].mode()[0])

test['BsmtExposure'] = test['BsmtExposure'].fillna(test['BsmtExposure'].mode()[0])
test['BsmtFinType2'] = test['BsmtFinType2'].fillna(test['BsmtFinType2'].mode()[0])


test.dropna(inplace=True)

#test.shape









#test.shape

#test.isnull().sum()

test['MasVnrType'] = test['MasVnrType'].fillna(test['MasVnrType'].mode()[0])
test['MasVnrArea'] = test['MasVnrArea'].fillna(test['MasVnrArea'].mode()[0])

test['BsmtExposure'] = test['BsmtExposure'].fillna(test['BsmtExposure'].mode()[0])
```

```
#test.drop(['Id'],axis=1,inplace=True)

test['BsmtExposure'] = test['BsmtExposure'].fillna(test['BsmtExposure'].mode()[0])


numeric = list(test.dtypes[test.dtypes!='object'].index)

test = test[numeric]

#test.shape
```

In [ ]:

```
test.head(2)
```

In [ ]:

```

```

In [ ]:

```
###################
```

In [ ]:

```

```

In [ ]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [ ]:

```
regr =RandomForestRegressor()
```

In [ ]:

```
regr.fit(xtrain_scaled,ytrain)
```

In [ ]:

```
regr.score(xtrain_scaled,ytrain)
```

In [ ]:

```
regr.score(xtest_scaled,ytest)
```

In [ ]:

```
predicted_value = regr.predict(xtest_scaled)
```

In [ ]:

```
predicted_value.shape
```

In [ ]:

```
#predicted_value
```

In [ ]:

```
ytest_li = []
for i in ytest.SalePrice:
    ytest_li.append(i)
```

In [ ]:

```
for i in range(len(predicted_value)):
    #print(predicted_value[i] , ytest_li[i])
    pass
```

In [ ]:

```
ytest_scaled2 = scaler.fit_transform(ytest)
```

In [ ]:

```
#ytest_scaled2
```

In [ ]:

```
reg_scaled2 = scaler.transform(predicted_value.reshape(-1,1))
```

In [ ]:

```
#reg_scaled2
```

In [ ]:

```
#reg_scaled2 = scaler.transform(reg_scaled2)
```

In [ ]:

```
mean_squared_error(ytest_scaled2,reg_scaled2)
```

In [ ]:

```
mean_squared_error(ytest_scaled2,reg_scaled2, squared = False) #RMSE
```

In [ ]:

```
mean_absolute_error(ytest_scaled2,reg_scaled2)
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
##########################################################
```

In [ ]:

```
##########################################################
```

In [ ]:

```
#predicted_value.to_list()
```

In [ ]:

```

```

In [ ]:

```
#pre = predicted_value.tolist()

#reg_scaled3 = scaler.transform(pre)
```

In [ ]:

```
#reg_scaled2
```

In [ ]:

```
predicted_value = reg.predict(xtest_scaled2)
```

In [ ]:

```
pre = predicted_value.tolist()
```

In [ ]:

```
predicted_value.shape
```

In [ ]:

```
li =[]
li.append(pre)
```

In [ ]:

```
#li
```

In [ ]:

```
predicted_value_scaled1 = scaler.transform(li)
```

In [ ]:

```
ytest_scaled = scaler.fit_transform(ytest)
```

In [ ]:

```
predicted_value_scaled = scaler.transform(predicted_value.tolist())
```

In [ ]:

```
result
```

In [ ]:

```
ytest_n
```

In [ ]:

```
mse(ytest_n,result.tolist())
```

In [ ]:

```
test['Price'] = result
```

In [ ]:

```
test.head(2)
```

In [ ]:

```
len(result)
```

In [ ]:

```
#result.ravel()
```

In [ ]:

```
len(ytest)
```

In [ ]:

```
#ytest
```

In [ ]:

```
#result.ravel()
```

In [ ]:

```
predicted_value.shape
```

In [ ]:

```
ytest.shape
```

In [ ]:

```
result = pd.DataFrame({"Actual price":ytest_li,"Predicted price":predicted_value})
```

In [ ]:

```
#result
```

In [ ]:

```
result.to_csv("house-price.csv",index=False) #to save result in csv file
```

In [ ]:

```
x ={result}
```

In [ ]:

```
#ytest
```

In [ ]:

```

```