**House Price Prediction**

**Import all the Libraries**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import os

**Adjust the Display settings**

pd.set\_option('display.width', 2000)

**Read the Data Set**

df1= pd.read\_csv("C:\\Noble\\Training\\Top Mentor\\Training\\Data Set\\House Price.csv")

print (df1)

**Read Dependent variable**

DV = input ('Enter the dependent variable')

position\_DV = df1.columns.get\_loc(DV)

print (position\_DV)

**Data Analysis commands**

print (len(df1))

print (df1.count())

print (df1.size)

print (df1.head())

print (df1.isnull().any)

print (df1.isnull().sum()) # Sum of Null Records

**Generate the values in the required format**

FileNameDesc = pd.DataFrame(columns=['ColumnName', 'MissingCount','MissingPercentage','UniqueValues'])

for col in list(df1.columns.values):

sum\_missing = df1[col].isnull().sum()

percent\_missing = (sum\_missing/len(df1)) \*100

uniq\_count =(df1.groupby ([col])[col].count()).count()

FileNameDesc = FileNameDesc.append ({'ColumnName': col,'MissingCount':sum\_missing,'MissingPercentage': percent\_missing,

'UniqueValues': uniq\_count} ,ignore\_index=True )

**Print the result Set**

print (FileNameDesc)

**Display datatype of each columns**

FileColType = (df1.dtypes).reset\_index()

print ( FileColType)

**Merge Data Type column and Data Value column**

FileNameDesc = FileNameDesc.merge(FileColType, left\_on ='ColumnName',right\_on ='index',how ='inner')

print (FileNameDesc)

**Delete Duplicate Index Column**

del FileNameDesc['index']

**Rename the column Name**

FileNameDesc = FileNameDesc.rename(index=str, columns = {0:"DataType"})

print (FileNameDesc)

**Delete Alley column since more Nulls Values present there**

df1.drop(['Alley'], axis = 1, inplace = True)

print (df1)

**Function to Remove outliers**

def remove\_outlier(df\_in, col\_name):

q1 = df\_in[col\_name].quantile(0.25)

q3 = df\_in[col\_name].quantile(0.75)

iqr = q3-q1

fence\_low = q1 - 1.5\*iqr

fence\_high = q3 + 1.5\*iqr

df\_out = df\_in.loc[(df\_in[col\_name]> fence\_low) & (df\_in[col\_name] < fence\_high)]

return df\_out

**Get int or Float columns to pass to Function to Remove outliers**

df\_col=FileNameDesc[(FileNameDesc.DataType=='int64') | (FileNameDesc.DataType =='float64')]

print (df\_col)

**Call function to remove outliers.**

for col\_na in df\_col['ColumnName']:

if col\_na != DV:

df1= remove\_outlier(df1, col\_na)

**Print the list of Missing Values**

print (df1.isnull().sum())

**Missing value treatment for GarageType column** . Convert all missing values Attchd since this is the maximum count

df1.groupby('GarageType').agg({'GarageType': np.size})

df1["GarageType"]=df1["GarageType"].fillna('Attchd')

**Convert GarageType column** to category from object and label encoding

df1["GarageType"] = df1["GarageType"].astype('category')

df1["GarageType"] = df1["GarageType"].cat.codes

**Display count by GarageType**

df1.groupby('GarageType').agg({'GarageType': np.size})

**Display the Data Frame**

print (df1)

**Print Data type of each column**

FileColType = (df1.dtypes)

print ( FileColType)

**Mean strategy to populate the quantitative variable**

df1["LotFrontage"].fillna(df1["LotFrontage"].mean(), inplace=True)

**Median strategy to populate the quantitative variable**

df1["MasVnrArea"].fillna(df1["MasVnrArea"].median(), inplace=True)

**One hot encoding ( Drop first column – Dummy Variable Trap)**

df1 = pd.get\_dummies(df1, columns=['GarageType'], drop\_first=True)

print (df1)

***Drop first column – Dummy Variable Trap***

*Y = m1x1 +m2x2+m3x3+c*

*Predict Salary of employee*

*X1 = Experience*

*X 2 = Age*

*X3 = Gender (Male / Female)*

*Salary = m1 X age +m2 X Exp + m3 X Gender (Male /Female ) +c*

*Salary = m1 X age +m2 X Exp + m3 X Gender (Male /Female ) +c*

*Expand*

*Salary = m1 X age +m2 X Exp + m3 X Male + m4 X Female +c*

*Salary Male*

*Salary = m1 X age +m2 X Exp + m3 X Male +c*

*Example*

*15000 = 23 2 M*

*Salary Male = 23 X m1 + 2 X m2 + m2 X M +C*

*14000 = 23 2 F*

*Salary Female = 23 X m1 + 2 X m2 + C*

*This is known as degree of freedom*

***Garage type all be zero for the 6th record***

**Export the data frame to CSV file to check one hot encoding result**

df1.to\_csv('C:\\Noble\\Training\\DS Temporary Files\\Sample1.csv')

os.getcwd()

**Keep the Sale price as last column**

df=df1['SalePrice']

df1.drop(['SalePrice'], axis = 1, inplace = True)

df1=pd.concat([df1,df],axis=1)

print (df1)

**Commands to check Data**

len(df1)

df1.head()

df1.isnull().any()

df1.isnull().sum()

**Create a pair plot – Check the last row to see the linear trend with variables , 7 variables has linear relation**

sns.pairplot(df1)

**Create Heat Map \_ Darker the colour \_ ve co relation lighter the colour – Positive co relation**

from matplotlib.pylab import rcParams

rcParams['figure.figsize'] = 20,10

df1.corr()

sns.heatmap(df1.corr(),annot=True)

**Start with all variables then continue with , backward elimination based on p – value ie P > 0.05**

**Split the data between X and Y**

x = df1.iloc[:,1:13]

y = df1.iloc[:,13:]

**Train Test Split**

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.2,random\_state = 100)

**Create the Linear Model**

from sklearn.linear\_model import LinearRegression

lm = LinearRegression()

lm = lm.fit(x\_train,y\_train)

**Display the coefficient**

lm.coef\_

**Display the coefficient as a Table**

coefficients = pd.concat([pd.DataFrame(x\_train.columns),pd.DataFrame(np.transpose(lm.coef\_))], axis = 1)

print (coefficients)

**Display Intercept**

lm.intercept\_

**Prediction**

y\_pred = lm.predict(x\_test)

**Display the Error / Differences**

y\_error = y\_test - y\_pred

y\_error

**Accuracy**

from sklearn.metrics import r2\_score

r2\_score(y\_test,y\_pred)

**Cross Validation**

from sklearn.model\_selection import cross\_val\_score

cvm= cross\_val\_score(lm, x, y, cv=10)

pd.DataFrame(cvm)

**Cross Validation Mean and Standard Deviation**

print ('Mean', cvm.mean())

print ('STD',cvm.std())

**Stats Model**

import statsmodels.api as sma

x\_train = sma.add\_constant(x\_train)

## let's add an intercept (beta\_0) to our model

x\_test = sma.add\_constant(x\_test)

**Stats Model OLS (Ordinary Least Square for Accuracy )**

import statsmodels.api as sm

lm2 = sm.OLS(y\_train,x\_train).fit()

lm2.summary()

**Variance Inflation Factor (VIF )**

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

[{x\_train.columns[j]:variance\_inflation\_factor(x\_train.values, j) for j in range(x\_train.shape[1])}]

*VIF used to find multicollinearity*

*VIF = 1/ (1- R2)*

*VIF exclude y and create model with x\_test*

*X1 = number of rooms*

*X2 = Area of the house*

*X1= m2x2+ c Assume R2 = 0.9*

*VIF = 1/(1-0.9) = 10*

*X2 = m1x1+c Assume R2 = 0.95*

*VIF = 1/(1-0.95) = 20*

*General Assumption if VIF > 5 , there is a multicollinearity exists*

**Function to Eliminate by using VIF**

def calculate\_vif(x):

thresh = 5.0

output = pd.DataFrame()

k = x.shape[1]

vif = [variance\_inflation\_factor(x.values, j) for j in range(x.shape[1])]

for i in range(1,k):

print("Iteration no: ", i)

print(vif, '\n')

a = np.argmax(vif)

print("Max VIF is for variable no:",a)

print("Column Name:",x.columns[a], '\n')

if vif[a] <= thresh :

break

if i == 1 :

x= x.drop(x.columns[a], axis = 1)

vif = [variance\_inflation\_factor(x.values, j) for j in range(x.shape[1])]

elif i > 1 :

x = x.drop(x.columns[a],axis = 1)

vif = [variance\_inflation\_factor(x.values, j) for j in range(x.shape[1])]

return(x)

**Call Function**

train\_out = calculate\_vif(x\_train)

**Print the Result After VIF**

train\_out.head()

**Create the Model after VIF elimination, with (train\_out) new set of variables**

from sklearn.linear\_model import LinearRegression

lm = LinearRegression()

lm = lm.fit(train\_out,y\_train)

**Check Model Accuracy with OLS Method**

import statsmodels.api as sm

lm2 = sm.OLS(y\_train,train\_out).fit()

lm2.summary()