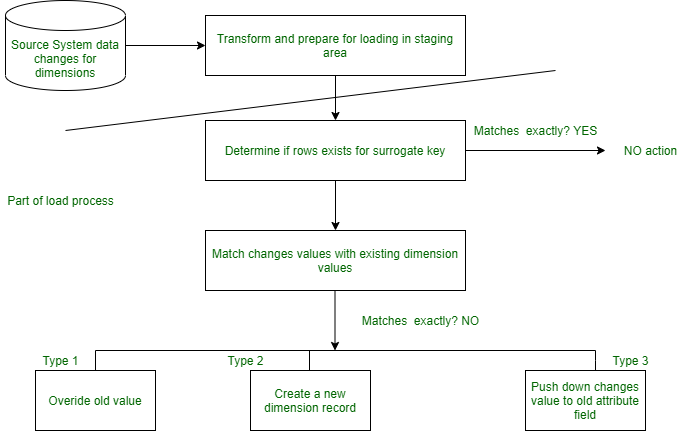
Predicting House Prices using Machine Learning

Overview:

The housing market is an important and complex sector that impacts people's lives in many ways. For many individuals and families, buying a house is one of the biggest investments they will make in their lifetime. Therefore, it is essential to accurately predict the prices of houses so that buyers and sellers can make informed decisions. This project aims to use machine learning techniques to predict house prices based on various features such as location, square footage, number of bedrooms and bathrooms, and other relevant factors.

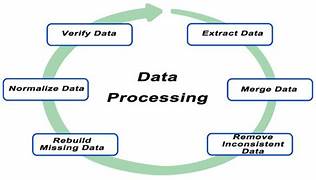
Data Loading:

Data loading is quite simply the process of packing up your data and moving it to a designated data warehouse. It is at the beginning of this transitory phase where you can begin planning a roadmap, outlining where you would like to move forward with your data and how you would like to use it. Data Loading is the ultimate step in the ETL process it also convert from one format to another format.



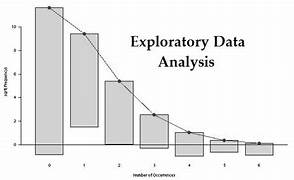
Data Preprocessing:

Data processing occurs when data is collected and translated into usable information. Usually performed by a data scientist or team of data scientists, it is important for data processing to be done correctly as not to negatively affect the end product, or data output.



Exploratory data Analysis:

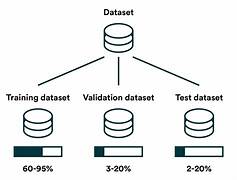
Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.



Data splitting:

Data splitting is when data is divided into two or more subsets. Typically, with a two-part split, one part is used to evaluate or test the data and the other to train the model.

Data splitting is an important aspect of data science, particularly for creating models based on data. This technique helps ensure the creation of data model and processes that use data models -- such as machine learning - are accurate.



**Code :**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

HouseDF = pd.read\_csv('USA\_Housing.csv')

HouseDF.head()

HouseDF=HouseDF.reset\_index()

HouseDF.head()

HouseDF.info()

HouseDF.describe()

HouseDF.columns

sns.pairplot(HouseDF)

sns.distplot(HouseDF['Price’])

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

X = HouseDF[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms','Avg. Area

Number of Bedrooms', 'Area Population']]

y = HouseDF['Price’]

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,test\_size=0.4,random\_state=101)

from sklearn.linear\_model import minmaxscaler

lm = minmaxscaler(feature\_range=(0,1))

lm.fit\_transform(X\_train,y\_train)

print(lm.intercept\_)

coeff\_df = pd.DataFrame(lm.coef\_,X.columns,columns=['Coefficient’])

coeff\_df

from keras.layersimport Dense,Dropout,LSTM

from keras.models import Sequential

model = Sequential()

model.add(LSTM(units = 50,activation = 'relu',return\_sequences = True,input\_shape =

(x\_train.shape[1], 1)))

model.add(Dropout(0.2))

model.add(LSTM(units = 60,activation = 'relu',return\_sequences = True))

model.add(Dropout(0.3))

model.add(LSTM(units = 80,activation = 'relu',return\_sequences = True))

model.add(Dropout(0.4))

model.add(LSTM(units = 120,activation = 'relu'))

model.add(Dropout(0.5))

model.add(Dense(units = 1))

model.compile(optimizer='adam', loss = 'mean\_squared\_error’)

model.fit(x\_train, y\_train,epochs=50)

print(lm.intercept\_)

coeff\_df = pd.DataFrame(lm.coef\_,X.columns,columns=['Coefficient’])

coeff\_df

predictions = lm.predict(X\_test)

scale\_factor = 1/0.02099517

y\_predicted = y\_predicted \* scale\_factory

y\_test = y\_test \* scale\_factor

plt.scatter(y\_test,predictions)

sns.distplot((y\_test-predictions),bins=50);

plt.figure(figsize=(12,6))

plt.plot(y\_test,'b',label = 'Original Price')

plt.plot(y\_predicted,'r',label = 'Predicted Price')

plt.xlabel('Time')

plt.ylabel('Price')

plt.legend()

plt.show()

from sklearn import metrics

print('MAE:', metrics.mean\_absolute\_error(y\_test, predictions))

print('MSE:', metrics.mean\_squared\_error(y\_test, predictions))

print('RMSE:', np.sqrt(metrics.mean\_squared\_error(y\_test, predictions)))

**Code** **Explanation :**

First we import a sample data from sklearn library , you can get different types of sample data from Kaggle. The data taken here is the data of various parameters and the house prices in a given city called boston in the year between 1970 to 2020.Then we check if our data has some null values i.e missing values. Since if the data is incomplete , then there will be error during processing state which may lead to loss of accuracy in predicting model.

After checking , if the data does not contains any missing value, the program will skip the dropping phase in data processing, where data is dropped to increase accuracy and fit missing values in a way so that it is suitable for modelling. Next we try to describe the data in such a way so that both people and machine find it easy to understand the given data . In order to do this we use the describe() function.Std means the standard value i.e the most common value in given set of data for a particular column. Min refers the least data value in each column. Max refers to the maximum data value in each column.

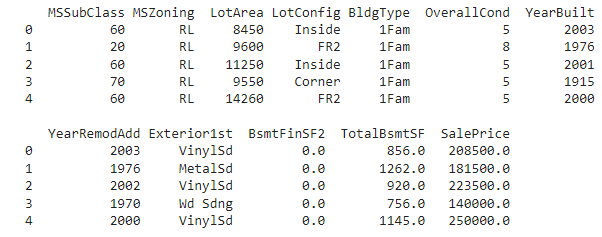
Then from keras layers import Dense, Dropout, LSTM and also import sequential model. Use Rectified Linear Unit (relu) for return sequences and activation. The scale factor is 1/0.02099517.Find the predicted and test results of house prices and multiply with the scale factor to find the actual predicted and test house prices.

Actual Predicted price=Predicted price\*Scale factor .

Actual Test price=Test price\*Scale factor

To compile this model it’s always best to use Adam optimizer and set the loss as required for the specific data. Then using matplotlib plot a graph for better understanding , taking x label as Time and y label as Price. Compare the test and predicted value to see the increase/decrease rate of values in each time of the year in a particular place. Based on this people will know when it’s best time to sell or buy a place in a given location.

Output:



Conclusion:

Thus the machine learning model to predict the house price based on given dataset is executed successfully using machine learning .This model further helps people understand whether this place is more suited for them based on heatmap correlation. It also helps people looking to sell a house at best time for greater profit. Any house price in any location can be predicted with minimum errorby giving appropriate dataset.