DM ASSIGNMENT Data Preprocessing Techniques & Data Classification (in Python)

28.05.2021

—

Thomas Jose
Roll No: 094
RegNo:20218096
Class: CS-B

Name: Thomas Jose
Roll no: 94
Reg no: 20218096

DM Assignment
Data Processing and Data classification
(in python)

A vital component of data science is cleaning the data and getting it ready for predictive modeling. The most common problem related to data cleaning is coping with the missing data, duplicate values, transforming data and visualizing data

Data Set

We will be using a dataset of prices of houses in a city containing 5000 observations and 7 columns

1. Average Area Income
2. Avg House Age
3. Avg Area Number of Rooms
4. Avg Number Of Bedrooms
5. Area Population
6. Price
7. Address

We will be predicting the prices of houses using linear regression model below.

Load the required libraries and the dataset. Also display the details of the dataset along with the entire dataset.

Import Libraries

In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
```

Importing Data and Checking out.

In [3]:
```
HouseDF = pd.read_csv('USA_Housing.csv'
                      )
```

In [4]:
```
HouseDF.head()
```

Out[4]:

	AvgAreaIncome	AvgAreaHouseAge	AvgAreaNumberofRooms	AvgAreaNumberofBedrooms	AreaPopulation	Price	Address
0	79545.45857	5.682861	7.009188	4.09	23086.80050	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.64245	6.002900	6.730821	3.09	40173.07217	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.06718	5.865890	8.512727	5.13	36882.15940	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.24005	7.188236	5.586729	3.26	34310.24283	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.19723	5.040555	7.839388	4.23	26354.10947	6.309435e+05	USNS Raymond\nFPO AE 09386

In [5]:
```
HouseDF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   AvgAreaIncome           4993 non-null   float64
 1   AvgAreaHouseAge         4996 non-null   float64
 2   AvgAreaNumberofRooms    5000 non-null   float64
 3   AvgAreaNumberofBedrooms 4997 non-null   float64
 4   AreaPopulation          4999 non-null   float64
 5   Price                   4999 non-null   float64
 6   Address                 5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 254.0+ KB
```

Dropped duplicate values

```
In [8]: HouseDF = HouseDF.drop_duplicates()
```

Replacing missing values with Mean and mode

```
In [9]: print("Number of missing values in each header:")
        print(HouseDF.isnull().sum())
```

```
Number of missing values in each header:
AvgAreaIncome               7
AvgAreaHouseAge             4
AvgAreaNumberofRooms        0
AvgAreaNumberofBedrooms     3
AreaPopulation              1
Price                       1
Address                     0
dtype: int64
```

```
In [10]: HouseDF.AvgAreaIncome.fillna(HouseDF.AvgAreaIncome.mean(), inplace = True)
         print(HouseDF.isnull().sum())
```

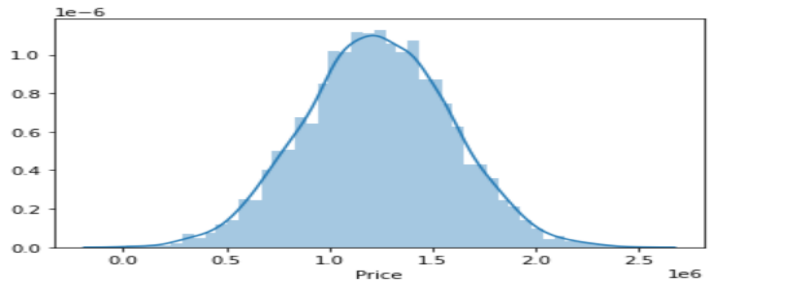
```
In [14]: HouseDF.AvgAreaHouseAge.fillna(HouseDF.AvgAreaHouseAge.mean(), inplace = True)
         HouseDF.AvgAreaNumberofBedrooms.fillna(HouseDF.AvgAreaNumberofBedrooms.mode()[0], inplace = True)
         HouseDF.AreaPopulation.fillna(HouseDF.AreaPopulation.mean(), inplace = True)
         HouseDF.Price.fillna(HouseDF.Price.mean(), inplace = True)
         print(HouseDF.isnull().sum())
```

```
AvgAreaIncome               0
AvgAreaHouseAge             0
AvgAreaNumberofRooms        0
AvgAreaNumberofBedrooms     0
AreaPopulation              0
Price                       0
Address                     0
dtype: int64
```

Data visualization

```
sns.distplot(HouseDF['Price'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x9d3e838>
```



Linear Regression Model

Linear regression attempts to **model** the relationship between two variables by fitting a **linear** equation to observed data. A **linear regression** line has an equation of the form Y = a + bX, where X is the explanatory variable and Y is the dependent variable.

```
In [19]: X = HouseDF[['AvgAreaIncome', 'AvgAreaHouseAge', 'AvgAreaNumberofRooms',
                      'AvgAreaNumberofBedrooms', 'AreaPopulation']]

         y = HouseDF['Price']
```

Split Data into Train, Test

```
In [20]: from sklearn.model_selection import train_test_split

In [21]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

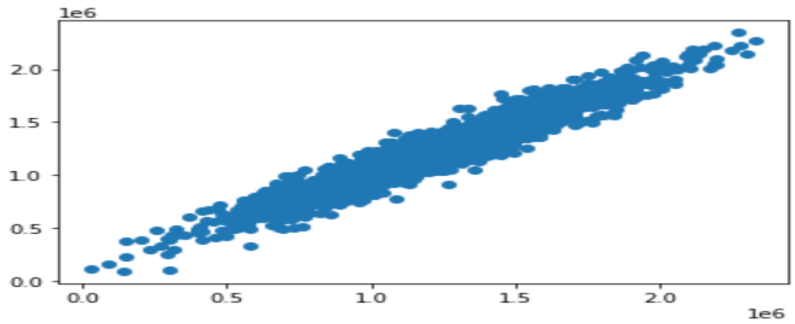
We split our dataset 60:40 for training and testing respectively.

Training The Model

```
In [22]: from sklearn.linear_model import LinearRegression

In [23]: lm = LinearRegression()

In [24]: lm.fit(X_train,y_train)
Out[24]: LinearRegression()
```

Predicting

```
In [27]: predictions = lm.predict(X_test)

In [28]: plt.scatter(y_test,predictions)
Out[28]: <matplotlib.collections.PathCollection at 0xad4d5b0>
```



In the above scatter plot, we see data is in line shape, which means our model has done good predictions.

Regression model evaluation metrics

In [59]:
```
from sklearn import metrics
```

In [60]:
```
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)))
```

```
MAE: 82612.0604625824
MSE: 10552164034.478632
RMSE: 102723.72673573828
```

Conclusion

Successfully completes data preprocessing and prediction using linear regression model on house price dataset.