omkargaikwad23 updates History

1 contributor

1683 lines (1683 sloc) | 55.7 KB

31126 Assignment-01

Data Wrangling, I Perform the following operations using Python on any open-source dataset (e.g., data.csv)

1. Import all the required Python Libraries.
2. Locate an open-source data from the web (e.g. https://www.kaggle.com). Provide a clear description of the data and its source (i.e., URL of the web site).
3. Load the Dataset into pandas' data frame.
4. Data Preprocessing: check for missing values in the data using pandas insult (), describe () function to get some initial statistics. Provide variable descriptions. Types of variables etc. Check the dimensions of the data frame.
5. Data Formatting and Data Normalization: Summarize the types of variables by checking the data types (i.e., character, numeric, integer, factor, and logical) of the variables in the data set. If variables are not in the correct data type, apply proper type conversions.
6. Turn categorical variables into quantitative variables in Python. In addition to the codes and outputs, explain every operation that you do in the above steps and explain everything that you do to import/read/scrape the data set.

1. Import all the required Python Libraries.

In [4]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
```

2. URL and description

URL: https://www.kaggle.com/anthonypino/melbourne-housing-market

Description: The dataset contains several attributes of the houses in Melbourne along with their prices. This dataset is made public by its owners. It contains numerous attributes that can affect the prices of the houses/apartments. Some of the features like no. of rooms, landsize area have clear effect on the price while some of the features are hard to examine by mere observation.

3. Load the Dataset into pandas' data frame.

In [2]:
```
df = pd.read_csv('melb_data.csv')
```

4. Generate descriptive statistics, check missing values

In [4]:
```
df.describe()
```

Out[4]:

	Rooms	Price	Distance	Postcode	Bedroom2	Bathroom	
count	13580.000000	1.358000e+04	13580.000000	13580.000000	13580.000000	13580.000000	1351
mean	2.937997	1.075684e+06	10.137776	3105.301915	2.914728	1.534242	
std	0.955748	6.393107e+05	5.868725	90.676964	0.965921	0.691712	
min	1.000000	8.500000e+04	0.000000	3000.000000	0.000000	0.000000	
25%	2.000000	6.500000e+05	6.100000	3044.000000	2.000000	1.000000	
50%	3.000000	9.030000e+05	9.200000	3084.000000	3.000000	1.000000	
75%	3.000000	1.330000e+06	13.000000	3148.000000	3.000000	2.000000	

max	10.000000	9.000000e+06	48.100000	3977.000000	20.000000	8.000000	1(

> **Return a tuple representing the dimensionality of the DataFrame.**

In [5]:
```
df.shape
```

Out[5]:
```
(13580, 21)
```

> **Read the headers**

In [6]:
```
df.columns
```

Out[6]:
```
Index(['Suburb', 'Address', 'Rooms', 'Type', 'Price', 'Method', 'SellerG',
       'Date', 'Distance', 'Postcode', 'Bedroom2', 'Bathroom', 'Car',
       'Landsize', 'BuildingArea', 'YearBuilt', 'CouncilArea', 'Lattitude',
       'Longtitude', 'Regionname', 'Propertycount'],
      dtype='object')
```

> **Read each column**

In [7]:
```
df[['Suburb', 'Address', 'Rooms']]
```

Out[7]:

	Suburb	Address	Rooms
0	Abbotsford	85 Turner St	2
1	Abbotsford	25 Bloomburg St	2
2	Abbotsford	5 Charles St	3
3	Abbotsford	40 Federation La	3
4	Abbotsford	55a Park St	4
...	...	...	...
13575	Wheelers Hill	12 Strada Cr	4
13576	Williamstown	77 Merrett Dr	3
13577	Williamstown	83 Power St	3
13578	Williamstown	96 Verdon St	4
13579	Yarraville	6 Agnes St	4

13580 rows × 3 columns

> **Read each row**

In [40]:
```
#for index, row in df.iterrows():
#    print(index, row['Name'])
df.loc[df['Suburb'] == "Brighton"]
```

Out[40]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Post(
1040	Brighton	802 Hampton St	3	h	1550000.0	VB	Buxton	3/09/2016	11.2	31

1041	Brighton	18 Rooding St	3	h	1635000.0	S	Buxton	3/09/2016	11.2	31
1042	Brighton	152A Cochrane St	4	h	1830000.0	S	Hodges	3/12/2016	11.2	31
1043	Brighton	17 McCallum St	4	h	3695000.0	S	Marshall	3/12/2016	11.2	31
1044	Brighton	2/26 Pearson St	2	u	536000.0	SP	Hodges	3/12/2016	11.2	31
...	...	...	...	...	...	...	...	...	...	
12915	Brighton	2A Hillcrest Av	4	h	1850000.0	VB	Jellis	19/08/2017	10.5	31
12916	Brighton	34 Moffat St	4	h	2770000.0	S	Nick	19/08/2017	10.5	31
13366	Brighton	3 Adamson St	4	h	3200000.0	VB	Marshall	26/08/2017	10.5	31
13367	Brighton	53 Elwood St	4	h	2830000.0	S	Nick	26/08/2017	10.5	31
13368	Brighton	44 Meek St	4	h	2260000.0	S	Nick	26/08/2017	10.5	31

186 rows × 21 columns

> **Read a specific location (R, C)**

In [44]:
```
print(df.iloc[2,1])
```

```
5 Charles St
```

In [3]:
```
df.isnull().sum()
```

Out[3]:
```
Suburb             0
Address            0
Rooms              0
Type               0
Price              0
Method             0
SellerG            0
Date               0
Distance           0
Postcode           0
Bedroom2           0
Bathroom           0
Car               62
Landsize           0
BuildingArea    6450
YearBuilt       5375
CouncilArea     1369
Lattitude          0
Longtitude         0
Regionname         0
Propertycount      0
dtype: int64
```

Filling missing values in dataset

In [16]:
```
df.fillna(0)

# filling a missing value with previous ones
df.fillna(method = 'pad')

# drop a columns which have at least 1 missing values
# df.dropna(axis = 1)

# drop any rows that have missing data.
# df.dropna(how="any")
```

Out[16]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	..
0	0	12794	2	0	1480000.0	1	23	45	2.5	3067.0	..
1	0	5943	2	0	1035000.0	1	23	47	2.5	3067.0	..
2	0	9814	3	0	1465000.0	3	23	48	2.5	3067.0	..
3	0	9004	3	0	850000.0	0	23	48	2.5	3067.0	..
4	0	10589	4	0	1600000.0	4	155	49	2.5	3067.0	..
...	...	...	...	...	...	...	...	...	...	...	..
13575	302	1991	4	0	1245000.0	1	16	33	16.7	3150.0	..
13576	305	12234	3	0	1031000.0	3	251	33	6.8	3016.0	..
13577	305	12745	3	0	1170000.0	1	194	33	6.8	3016.0	..
13578	305	13311	4	0	2500000.0	0	222	33	6.8	3016.0	..
13579	313	10776	4	0	1285000.0	3	239	33	6.3	3013.0	..

13580 rows × 21 columns

5. Data types in data frame

In [45]:
```
df.dtypes
```

Out[45]:
```
Suburb           object
Address          object
Rooms             int64
Type             object
Price           float64
Method           object
SellerG          object
Date             object
Distance        float64
Postcode        float64
Bedroom2        float64
Bathroom        float64
Car             float64
Landsize        float64
BuildingArea    float64
YearBuilt       float64
CouncilArea      object
Lattitude       float64
Longtitude      float64
Regionname       object
Propertycount   float64
dtype: object
```

6. Conversion of dtypes

In [15]:

```
df1 = df.copy()
dfn = df1.convert_dtypes()
dfn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13580 entries, 0 to 13579
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         13580 non-null  Int32
 1   Address        13580 non-null  Int32
 2   Rooms          13580 non-null  Int64
 3   Type           13580 non-null  Int32
 4   Price          13580 non-null  Int64
 5   Method         13580 non-null  Int32
 6   SellerG        13580 non-null  Int32
 7   Date           13580 non-null  Int32
 8   Distance       13580 non-null  Float64
 9   Postcode       13580 non-null  Int64
 10  Bedroom2       13580 non-null  Int64
 11  Bathroom       13580 non-null  Int64
 12  Car            13518 non-null  Int64
 13  Landsize       13580 non-null  Int64
 14  BuildingArea   7130 non-null   Float64
 15  YearBuilt      8205 non-null   Int64
 16  CouncilArea    12211 non-null  string
 17  Lattitude      13580 non-null  Float64
 18  Longtitude     13580 non-null  Float64
 19  Regionname     13580 non-null  Int32
 20  Propertycount  13580 non-null  Int64
dtypes: Float64(4), Int32(7), Int64(9), string(1)
memory usage: 2.1 MB
```

In [13]:

```
def encode_features(df):
    features = ['Suburb', 'Address', 'Type', 'Method', 'SellerG', 'Date', 'Re
    for feature in features:
        le = LabelEncoder()
        le = le.fit(df[feature])
        df[feature] = le.transform(df[feature])
    return df

df1 = encode_features(df)
df1.head()
```

Out[13]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Ba
0	0	12794	2	0	1480000.0	1	23	45	2.5	3067.0	...	
1	0	5943	2	0	1035000.0	1	23	47	2.5	3067.0	...	
2	0	9814	3	0	1465000.0	3	23	48	2.5	3067.0	...	
3	0	9004	3	0	850000.0	0	23	48	2.5	3067.0		