Lab-3_Knowing_data

July 15, 2020

Descriptive Statistics in Python Python Descriptive Statistics process describes the basic features of data in a study. It delivers summaries on the sample and the measures and does not use the data to learn about the population it represents. Under descriptive statistics, fall two sets of properties- central tendency and dispersion. Python Central tendency characterizes one central value for the entire distribution. Measures under this include mean, median, and mode. Python Dispersion is the term for a practice that characterizes how apart the members of the distribution are from the center and from each other. Variance/Standard Deviation is one such measure of variability.

1. Descriptive Statistics – Central Tendency in Python Now let's take a look at all the functions Python caters to us to calculate the central tendency for a distribution.

Let us now understand the functions under Descriptive Statistics in Python Pandas. The following table list down the important functions Sr.No. Function Description 1 count() Number of non-null observations 2 sum() Sum of values 3 mean() Mean of Values 4 median() Median of Values 5 mode() Mode of values 6 std() Standard Deviation of the Values 7 min() Minimum Value 8 max() Maximum Value 9 abs() Absolute Value 10 prod() Product of Values 11 cumsum() Cumulative Sum 12 cumprod() Cumulative Product

Functions like sum(), cumsum() work with both numeric and character (or) string data elements without any error. Though in practice, character aggregations are never used generally, these functions do not throw any exception.

Functions like abs(), cumprod() throw exception when the DataFrame contains character or string data because such operations cannot be performed.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
houseprice="house_price.csv"
df = pd.read_csv(houseprice)
```

In [3]:
```
df.head()
```

Out[3]:
```
   LotFrontage  LotArea  OverallQual  OverallCond  YearBuilt  YearRemodAdd  \
0         65.0     8450            7            5       2003          2003
1         80.0     9600            6            8       1976          1976
2         68.0    11250            7            5       2001          2002
3         60.0     9550            7            5       1915          1970
```

```
4        84.0    14260            8            5       2000          2000

   MasVnrArea  BsmtFinSF1  BsmtFinSF2  BsmtUnfSF    ...      GarageArea  \
0       196.0         706           0        150    ...             548
1         0.0         978           0        284    ...             460
2       162.0         486           0        434    ...             608
3         0.0         216           0        540    ...             642
4       350.0         655           0        490    ...             836

   WoodDeckSF  OpenPorchSF  EnclosedPorch  3SsnPorch  ScreenPorch  PoolArea  \
0           0           61              0          0            0         0
1         298            0              0          0            0         0
2           0           42              0          0            0         0
3           0           35            272          0            0         0
4         192           84              0          0            0         0

   MiscVal  YrSold  SalePrice
0        0    2008     208500
1        0    2007     181500
2        0    2008     223500
3        0    2006     140000
4        0    2008     250000

[5 rows x 35 columns]
```

In [4]:
```
df.shape
```

Out[4]:
```
(1379, 35)
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1379 entries, 0 to 1378
Data columns (total 35 columns):
LotFrontage      1379 non-null float64
LotArea          1379 non-null int64
OverallQual      1379 non-null int64
OverallCond      1379 non-null int64
YearBuilt        1379 non-null int64
YearRemodAdd     1379 non-null int64
MasVnrArea       1379 non-null float64
BsmtFinSF1       1379 non-null int64
BsmtFinSF2       1379 non-null int64
BsmtUnfSF        1379 non-null int64
TotalBsmtSF      1379 non-null int64
1stFlrSF         1379 non-null int64
2ndFlrSF         1379 non-null int64
LowQualFinSF     1379 non-null int64
GrLivArea        1379 non-null int64
```

```
BsmtFullBath     1379 non-null int64
BsmtHalfBath     1379 non-null int64
FullBath         1379 non-null int64
HalfBath         1379 non-null int64
BedroomAbvGr     1379 non-null int64
KitchenAbvGr     1379 non-null int64
TotRmsAbvGrd     1379 non-null int64
Fireplaces       1379 non-null int64
GarageYrBlt      1379 non-null float64
GarageCars       1379 non-null int64
GarageArea       1379 non-null int64
WoodDeckSF       1379 non-null int64
OpenPorchSF      1379 non-null int64
EnclosedPorch    1379 non-null int64
3SsnPorch        1379 non-null int64
ScreenPorch      1379 non-null int64
PoolArea         1379 non-null int64
MiscVal          1379 non-null int64
YrSold           1379 non-null int64
SalePrice        1379 non-null int64
dtypes: float64(3), int64(32)
memory usage: 377.1 KB
```

```
In [6]: df.describe(include='all')
```

```
Out[6]:        LotFrontage        LotArea  OverallQual  OverallCond    YearBuilt  \
        count  1379.000000    1379.000000  1379.000000  1379.000000  1379.000000
        mean     57.766497   10695.812183     6.187092     5.577955  1972.958666
        std      35.038221   10214.702133     1.345780     1.081031    29.379883
        min       0.000000    1300.000000     2.000000     2.000000  1880.000000
        25%      41.500000    7741.000000     5.000000     5.000000  1955.000000
        50%      64.000000    9591.000000     6.000000     5.000000  1976.000000
        75%      79.000000   11708.500000     7.000000     6.000000  2001.000000
        max     313.000000  215245.000000    10.000000     9.000000  2010.000000

               YearRemodAdd   MasVnrArea   BsmtFinSF1   BsmtFinSF2    BsmtUnfSF  \
        count   1379.000000  1379.000000  1379.000000  1379.000000  1379.000000
        mean    1985.435098   108.364757   455.578680    48.102248   570.765047
        std       20.444852   184.195220   459.691379   164.324665   443.677845
        min     1950.000000     0.000000     0.000000     0.000000     0.000000
        25%     1968.000000     0.000000     0.000000     0.000000   228.000000
        50%     1994.000000     0.000000   400.000000     0.000000   476.000000
        75%     2004.000000   170.500000   732.000000     0.000000   811.000000
        max     2010.000000  1600.000000  5644.000000  1474.000000  2336.000000

                   ...       GarageArea   WoodDeckSF  OpenPorchSF  EnclosedPorch  \
        count      ...      1379.000000  1379.000000  1379.000000    1379.000000
```

```
mean      ...     500.762146   97.456853   47.276287    21.039159
std       ...     185.680520  126.699192   65.210465    60.535107
min       ...     160.000000    0.000000    0.000000     0.000000
25%       ...     380.000000    0.000000    0.000000     0.000000
50%       ...     484.000000    0.000000   27.000000     0.000000
75%       ...     580.000000  171.000000   69.500000     0.000000
max       ...    1418.000000  857.000000  547.000000   552.000000

         3SsnPorch  ScreenPorch     PoolArea       MiscVal       YrSold  \
count  1379.000000  1379.000000  1379.000000   1379.000000  1379.000000
mean      3.609862    15.945613     2.920957     42.889050  2007.812183
std      30.154682    57.249593    41.335545    501.613931     1.330221
min       0.000000     0.000000     0.000000      0.000000  2006.000000
25%       0.000000     0.000000     0.000000      0.000000  2007.000000
50%       0.000000     0.000000     0.000000      0.000000  2008.000000
75%       0.000000     0.000000     0.000000      0.000000  2009.000000
max     508.000000   480.000000   738.000000  15500.000000  2010.000000

          SalePrice
count    1379.00000
mean   185479.51124
std     79023.89060
min     35311.00000
25%    134000.00000
50%    167500.00000
75%    217750.00000
max    755000.00000

[8 rows x 35 columns]
```

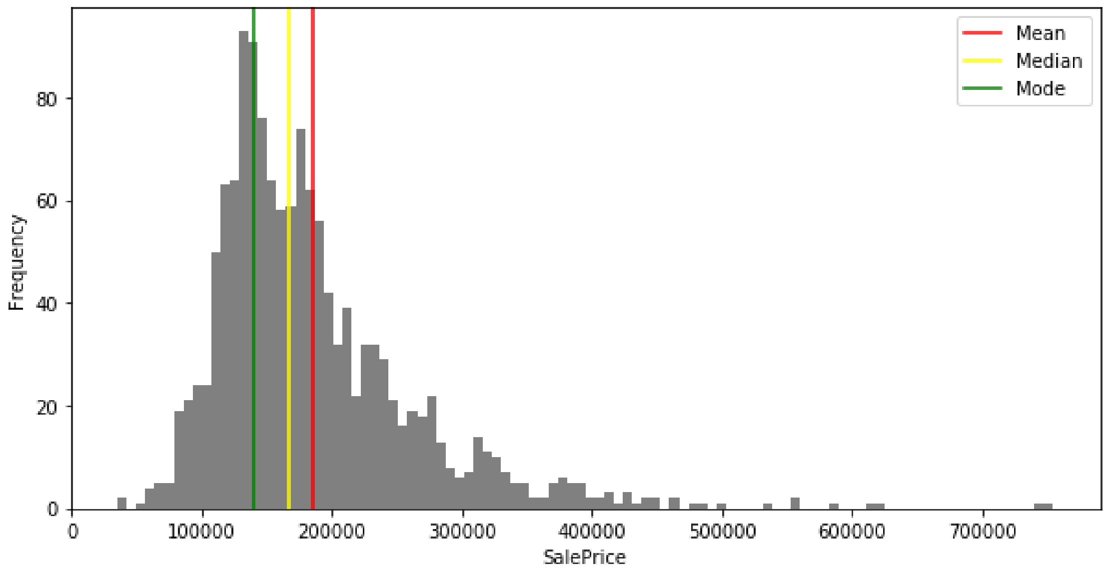
Measures of central tendency: This measure tries to describe the entire dataset with a single value or metric which represents the middle or center of distribution. It is also known as measure of center or central location. These measures include:

Mean: Computed by taking the sum of all the values in the dataset divided by the total number of values. Median: It's the value which lies in the middle of the dataset when arranged in ascending or descending order. Mode: It is the most occurring value in the dataset or the value which occurs very frequently.

```
In [7]: saleprice = df['SalePrice']

        mean=saleprice.mean()
        median=saleprice.median()
        mode=saleprice.mode()

        print('Mean: ',mean,'\nMedian: ',median,'\nMode: ',mode[0])
        plt.figure(figsize=(10,5))
        plt.hist(saleprice,bins=100,color='grey')
        plt.axvline(mean,color='red',label='Mean')
```

```
plt.axvline(median,color='yellow',label='Median')
plt.axvline(mode[0],color='green',label='Mode')
plt.xlabel('SalePrice')
plt.ylabel('Frequency')
plt.legend()
plt.show()
```

```
Mean:  185479.511240029
Median:  167500.0
Mode:  140000
```



In [8]:
```
saleprice.cumsum().head()
```

Out[8]:
```
0      208500
1      390000
2      613500
3      753500
4     1003500
Name: SalePrice, dtype: int64
```

1.1 The problem of outliers Remember that the mean is calculated by summing up all the values we want and dividing by the number of items, while the median is found by simply rearranging items. If we have outliers in our data, items that are much higher or lower than the other values, it can have an adverse effect on the mean. That is to say, the mean is not robust to outliers. The median, not having to look at outliers, is robust to them. Let's have a look at the maximum and minimum prices that we see in our data.

In [9]:
```
saleprice.min() #maximum value of salePrice
```

```
Out[9]: 35311
```

```
In [10]: saleprice.max() #minimum value of salePrice
```

```
Out[10]: 755000
```

We now know that outliers are present in our data. Outliers can represent interesting events or errors in our data collection, so it's important to be able to recognize when they're present in the data. The comparison of median and mode is just one of many ways to detect the presence of outliers, though visualization is usually a quicker way to detect them.

2. Python Descriptive Statistics – Dispersion in Python Measures of spread The measures of spread (also known as dispersion) answer the question, "How much does my data vary?" There are few things in the world that stay the same everytime we observe it. We all know someone who has lamented a slight change in body weight that is due to natural fluctuation rather than outright weight gain. This variability makes the world fuzzy and uncertain, so it's useful to have metrics that summarize this "fuzziness."

Range and interquartile range

```
In [11]: print ('Range =',saleprice.max()-saleprice.min())

Range = 719689
```

```
In [ ]: We found a range of 719689, but what does that mean precisely?
        When we look at our various measures, it is important to keep all of this information i
        Our median price was 167500. The range is two orders of magnitude higher than our media
        Perhaps if we had another house data set, we could compare the ranges of these two data
        Otherwise, the range alone isnt super helpful.
        More often, well want to see how much our data varies from the typical value.
        This summary falls under the jurisdiction of standard deviation and variance.

        Standard deviation
        The standard deviation is also a measure of the spread of your observations, but is a s
        data deviates from a typical data point. That is to say, the standard deviation summari
        from the mean.
```

```
In [12]: #variance
         saleprice.var()
```

```
Out[12]: 6244775285.521461
```

```
In [13]: from math import sqrt
         #standard deviation
         std = sqrt(saleprice.var())
         std
```

```
Out[13]: 79023.89059975129
```

The larger the standard deviation, the more spread out the data is around the mean and vice-versa. We will see that variance is closely related to standard deviation.

Variance and standard deviation are almost the exact same thing! Variance is just the square of the standard deviation. Likewise, variance and standard deviation represent the same thing — a measure of spread — but it's worth noting that the units are different. Whatever units your data are in, standard deviation will be the same, and variation will be in that units-squared. A question that many statistics starters ask is, "But why do we square the deviation? Won't the absolute value get rid of pesky negatives in the sum?" While avoiding negative values in the sum is a reason for the squaring operation, it's not the only one.

Like the mean, variance and standard deviation are affected by outliers. Many times, outliers are also points of interest in our data set, so squaring the difference from the mean allows us to point out this significance. If your are familiar with calculus, you'll see that having an exponential term allows us to find our where the point of minimum deviation is. More often than not, any statistical analyses you do will require just the mean and standard deviation, but the variance still has significance in other academic areas. The measures of central tendency and spread allow us to summarize key aspects of our data set, and we can build on these summaries to glean more insights from our data.

Quartile/ Percentile: makes it easy to work with data which is not symmetrically distributed and has outliers. Mean, Median and mode is the numerical summary of the entire dataset which is symmetrically distributed whereas quartiles divide our dataset into four equally sized groups based on five number summary: Minimum, first quartile, median, third quartile and maximum. The box in the box plot represents the 50 percent of the data values known as interquartile range (IQR). IQR indicates the variability in the set of values. Large IQR means a large spread in values. Small IQR indicates most of the values fall near the center of data. Box plot shows minimum and maximum values through the whiskers which extends both the sides and also outlier points which extends beyond the whiskers.

Interquartile Range (IQR): It's the difference between the third quartile and the first quartile. 50% of the population data lies here.

```
In [19]: #50th percentile i.e median(q2)
         df['SalePrice'].quantile(0.5)

Out[19]: 167500.0

In [20]: #75th percentile
         q3 = df['SalePrice'].quantile(0.75)
         q3

Out[20]: 217750.0

In [22]: #25th percentile
         q1 = df['SalePrice'].quantile(0.25)
         q1

Out[22]: 134000.0

In [23]: #interquartile range
         IQR = q3  - q1
         IQR
```

Out[23]: 83750.0

Measures to describe shape of distribution: Measures of center and spread tells us just about the central values and spread. How do we describe the shape of the distribution? Histogram will give us a general idea, but two numerical measures of shape will help us with the precise evaluation of the shape of the distribution.

Skewness is the asymmetry in the distribution because of which the curve appears distorted or skewed either to left or right of the normal distribution in a dataset. In other words skewness is the extent to which a distribution differs from a normal distribution.

Skewed distributions can be: Positively skewed: Most frequent values are low and tail is towards high values. Negatively skewed: Most frequent values are high and tail is towards low values.

Measures of central tendency can also be used to detect skewness in a distribution. Central tendency values will not be the same if the distribution is skewed. If Mode< Median< Mean then the distribution is positively skewed. If Mode> Median> Mean then the distribution is negatively skewed.

In [14]:
```
#skewness
saleprice.skew()
```

Out[14]: 1.935362098363132

In [ ]:
```
Kurtosis is the measure of the combined weight of the tails of the distribution relati
When a normal distribution is represented via histogram, it shows a bell curve with + 
However, when kurtosis is present then the tails further extend farther than the + and
bell-curved distribution. Kurtosis makes the data to look flatter (or less flat as com
The standard normal distribution has kurtosis of 3.
```

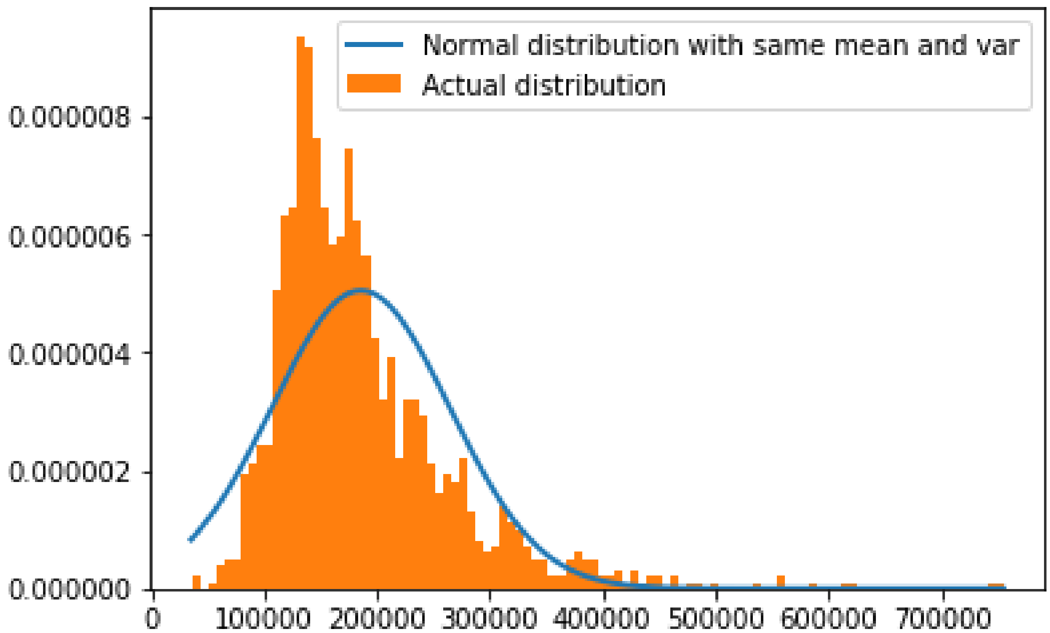
In [16]:
```
#kutosis
saleprice.kurt()
```

Out[16]: 6.735649337267559

In [ ]:
```
Types of Kurtosis
When the distribution of the data is similar to the normal distribution or the kurtosi
it is called as Mesokurtic distribution.

Any distribution which has kurtosis more than Normal distribution (K>3), it is called 
This type distribution has positive kurtosis.

Distribution which has kurtosis less than Normal distribution (K < 3), it is called as
    This type distribution has negative kurtosis.
```

In [18]:
```
import scipy.stats as stats
#convert pandas DataFrame object to numpy array and sort
h = np.asarray(df['SalePrice'])
h = sorted(h)

#use the scipy stats module to fit a normal distirbution with same mean and standard 
```

```
fit = stats.norm.pdf(h, np.mean(h), np.std(h))

#plot both series on the histogram
plt.plot(h,fit,'-',linewidth = 2,label="Normal distribution with same mean and var")
plt.hist(h,normed=True,bins = 100,label="Actual distribution")
plt.legend()
plt.show()
```



We can see int the above graph that it is positively skewed with skewness score 1.93 and also has positive kurtosis(k=6.735)