Machine Learning

ML - Understanding Data with Statistics

Computer & Health Informatics Department

Introduction

- In the previous chapter, we discussed how we can upload CSV data into our ML project, but it would be good to understand the data before uploading it.
- We can understand the data by two ways, with statistics and with visualization.
- In this chapter, with the help of following Python recipes, we are going to understand ML data with statistics.

Looking at Raw Data

- The very first recipe is for looking at your raw data. It is important to look at raw data because the insight we will get after looking at raw data will boost our chances to better pre-processing as well as handling of data for ML projects.
- Following is a Python script implemented by using head() function of Pandas DataFrame on Pima Indians diabetes dataset to look at the first 50 rows to get better understanding of it −

Example

```
from pandas import read_csv
path = r"C:\pima-indians-diabetes.csv"
headernames = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = read_csv(path, names=headernames)
print(data.head(50))
```

Checking Dimensions of Data

- It is always a good practice to know how much data, in terms of rows and columns, we are having for our ML project. The reasons behind are −
  - Suppose if we have too many rows and columns then it would take long time to run the algorithm and train the model.
  - Suppose if we have too less rows and columns then it we would not have enough data to well train the model.
- Following is a Python script implemented by printing the shape property on Pandas Data Frame. We are going to implement it on iris data set for getting the total number of rows and columns in it.

Example

```
from pandas import read_csv
path = r"C:\iris.csv"
data = read_csv(path)
print(data.shape)
```

Output

```
(150, 4)
```

We can easily observe from the output that iris data set, we are going to use, is having 150 rows and 4 columns.

Getting Each Attribute's Data Type

- It is another good practice to know data type of each attribute. The reason behind is that, as per to the requirement, sometimes we may need to convert one data type to another.
- For example, we may need to convert string into floating point or int for representing categorial or ordinal values.
- We can have an idea about the attribute's data type by looking at the raw data, but another way is to use *dtypes* property of Pandas DataFrame.

Getting Each Attribute's Data Type

- With the help of *dtypes* property we can categorize each attributes data type.
- It can be understood with the help of following Python script −

Example

```
from pandas import read_csv
path = r"C:\iris.csv"
data = read_csv(path)
print(data.dtypes)
```

Getting Each Attribute's Data Type

Output

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
dtype: object
```

- From the above output, we can easily get the datatypes of each attribute.

Statistical Summary of Data

- We have discussed Python recipe to get the shape i.e. number of rows and columns, of data but many times we need to review the summaries out of that shape of data.
- It can be done with the help of *describe()* function of Pandas DataFrame that further provide the following 8 statistical properties of each & every data attribute −

Statistical Summary of Data

- Count
- Mean
- Standard Deviation
- Minimum Value
- Maximum value
- 25%
- Median i.e. 50%
- 75%

Example

```
from pandas import read_csv
from pandas import set_option
path = r"C:\pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = read_csv(path, names=names)
set_option('display.width', 100)
set_option('precision', 2)
print(data.shape)
print(data.describe())
```

Output

```
(768, 9)
       preg     plas     pres     skin     test     mass     pedi     age      class
count  768.00   768.00   768.00   768.00   768.00   768.00   768.00   768.00   768.00
mean     3.85   120.89    69.11    20.54    79.80    31.99     0.47    33.24     0.35
std      3.37    31.97    19.36    15.95   115.24     7.88     0.33    11.76     0.48
min      0.00     0.00     0.00     0.00     0.00     0.00     0.08    21.00     0.00
25%      1.00    99.00    62.00     0.00     0.00    27.30     0.24    24.00     0.00
50%      3.00   117.00    72.00    23.00    30.50    32.00     0.37    29.00     0.00
75%      6.00   140.25    80.00    32.00   127.25    36.60     0.63    41.00     1.00
max     17.00   199.00   122.00    99.00   846.00    67.10     2.42    81.00     1.00
```

From the above output, we can observe the statistical summary of the data of Pima Indian Diabetes dataset along with shape of data.

Reviewing Class Distribution

- Class distribution statistics is useful in classification problems where we need to know the balance of class values. It is important to know class value distribution because if we have highly imbalanced class distribution i.e. one class is having lots more observations than other class, then it may need special handling at data preparation stage of our ML project. We can easily get class distribution in Python with the help of Pandas DataFrame.

Example

```
from pandas import read_csv
path = r"C:\pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = read_csv(path, names=names)
count_class = data.groupby('class').size()
print(count_class)
```

Output

```
Class
0 500
1 268
dtype: int64
```

From the above output, it can be clearly seen that the number of observations with class 0 are almost double than number of observations with class 1.

Reviewing Correlation between Attributes

- The relationship between two variables is called correlation. In statistics, the most common method for calculating correlation is Pearson's Correlation Coefficient. It can have three values as follows :
- **Coefficient value = 1**

  It represents full **positive** correlation between variables.
- **Coefficient value = -1**

  It represents full **negative** correlation between variables.
- **Coefficient value = 0**

  It represents **no** correlation at all between variables.

Reviewing Correlation between Attributes

- It is always good for us to review the pairwise correlations of the attributes in our dataset before using it into ML project because some machine learning algorithms such as linear regression and logistic regression will perform poorly if we have highly correlated attributes.
- In Python, we can easily calculate a correlation matrix of dataset attributes with the help of *corr()* function on Pandas DataFrame.

Example

```
from pandas import read_csv
from pandas import set_option
path = r"C:\pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = read_csv(path, names=names)
set_option('display.width', 100)
set_option('precision', 2)
correlations = data.corr(method='pearson')
print(correlations)
```

Output

```
preg   plas   pres   skin   test    mass    pedi    age    class
preg   1.00   0.13   0.14  -0.08   -0.07    0.02   -0.03   0.54    0.22
plas   0.13   1.00   0.15   0.06    0.33    0.22    0.14   0.26    0.47
pres   0.14   0.15   1.00   0.21    0.09    0.28    0.04   0.24    0.07
skin  -0.08   0.06   0.21   1.00    0.44    0.39    0.18  -0.11    0.07
test  -0.07   0.33   0.09   0.44    1.00    0.20    0.19  -0.04    0.13
mass   0.02   0.22   0.28   0.39    0.20    1.00    0.14   0.04    0.29
pedi  -0.03   0.14   0.04   0.18    0.19    0.14    1.00   0.03    0.17
age    0.54   0.26   0.24  -0.11   -0.04    0.04    0.03   1.00    0.24
class  0.22   0.47   0.07   0.07    0.13    0.29    0.17   0.24    1.00
```

The matrix in above output gives the correlation between all the pairs of the attribute in dataset.

Reviewing Skew of Attribute Distribution

- Skewness may be defined as the distribution that is assumed to be Gaussian but appears distorted or shifted in one direction or another, or either to the left or right.
- Reviewing the skewness of attributes is one of the important tasks due to following reasons :
- Presence of skewness in data requires the correction at data preparation stage so that we can get more accuracy from our model.

Reviewing Skew of Attribute Distribution

- Most of the ML algorithms assumes that data has a Gaussian distribution i.e. either normal of bell curved data.
- In Python, we can easily calculate the skew of each attribute by using **skew()** function on Pandas DataFrame.

Example

```
from pandas import read_csv
path = r"C:\pima-indians-diabetes.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = read_csv(path, names=names)
print(data.skew())
```

Output

```
preg   0.90
plas   0.17
pres  -1.84
skin   0.11
test   2.27
mass  -0.43
pedi   1.92
age    1.13
class  0.64
dtype: float64
```

From the above output, positive or negative skew can be observed. If the value is closer to zero, then it shows less skew.

Thank you