Python For Machine Learning

Tutorials 01
Using Pandas

Outlines

Part One:

- Introduction To Pandas
- Basic Data Analysis

Part Two:

- Feature Representation
- Data Wrangling

Part One

1. Introduction to Pandas
2. Basic Data Analysis

Put Into Practice

The best method to use this tutorial is to apply the code samples in slides while you are studying the tutorial.

Open your favorite python editor and download the iris dataset from the course github repo and follow the tutorial step by step

The dataset is under ***code/pyhton_tutorials/01/***

Introduction To Pandas

- Pandas is a high level data manipulation tool.
- In pandas, we store data in a so-called dataframe.
- A dataframe is a table:
  - The rows represent different entries or observations n
  - The columns represent different properties, and are identified by their column labels

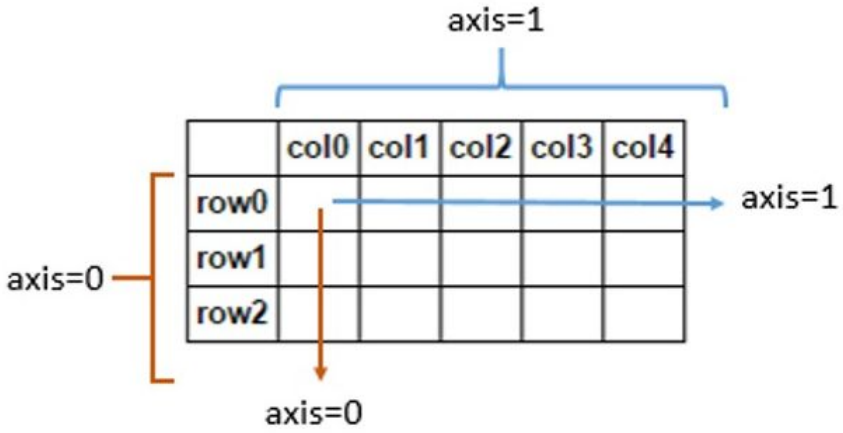
Introduction To Pandas

What can we do with Pandas?

- Read|Load data from different data sources (e.g. CSV, SQL, XML, JSON, Web, etc)
- Manipulate date (e.g. select subset, drop subset, edit and update samples, etc)
- Statistical analysis
- Feature representation (e.g. encode categorical features)
- Basic date pre-processing and wrangling

Pandas Data Frame Layout

In Pandas, *axes* refers to the two-dimensional, matrix-like shape of your dataframe. Samples span horizontal rows and are stacked vertically on top of one another by index (axis=0). Features are vertical spans that are stacked horizontally next to each other by columns (axis=1):



Using Pandas: Import Pandas

To use pandas module, use the python **import** statement to import pandas in your python script.

```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

We used the **as** to create an alias while importing a module. It means giving a different name (user-defined) to a module while importing it. In this case we use **pd** as an alias to padas

Using Pandas: Loading Data

As we mentioned before we can load data from different source the following examples show how we can load data from XML file, JSON file, CSV file, or even an HTML table from a web page on web server or URL

```
xls_dataframe  = pd.read_excel('dataset.xlsx', 'Sheet1', na_values=['NA', '?'])
json_dataframe = pd.read_json('dataset.json', orient='columns')
csv_dataframe  = pd.read_csv('dataset.csv', sep=',')
table_dataframe= pd.read_html('http://somewhere.com/iris/table.html')[0]
```

As we can see we do that using several built-in I/O methods in pandas, such as pd.read_csv()

Using Pandas: Loading Data

We can even load data from SQL database. However, in this case pandas will need help from other SQL module to be able to understand SQL tables.

```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///:memory:')

sql_dataframe  = pd.read_sql_table('table_name', engine, columns=['ColA', 'ColB'])
```

Usually, it is better to export your SQL tables into csv files and work with the csv files.

Using Pandas: Loading Iris Dataset

Let us load the Iris data set from an csv file and check the number of samples in the dataset.

```
csv_dataframe  = pd.read_csv('../input/iris/Iris.csv', sep=',')

print('number of samples is {0}'.format(len(csv_dataframe)))
```

OUTPUT

```
Time  Line #  Log Message
1.4s       0  number of samples is 150
```

Using Pandas: A Quick Peek

To check the data features, you can print the feature names (column headers | name) using the columns property of the dataframe

```
print(csv_dataframe.columns.values)
```

OUTPUT

```
Time  Line #  Log Message
1.4s       0  ['Id' 'SepalLengthCm' 'SepalWidthCm' 'PetalLengthCm' 'PetalWidthCm'
               'Species']
```

Using Pandas: A Quick Peek

To check the data features, you can print the feature names (column headers|name) using the **columns.values** property of the dataframe

```
print(csv_dataframe.columns.values)
```

OUTPUT

```
Time  Line #  Log Message
1.4s       0  ['Id' 'SepalLengthCm' 'SepalWidthCm' 'PetalLengthCm' 'PetalWidthCm'
               'Species']
```

Using Pandas: A Quick Peek

To check the types of the features, you can print the feature names (column headers| name) and types using the **dtypes** property of the dataframe

```
print(csv_dataframe.dtypes)
```

OUTPUT

```
Time  Line #  Log Message
1.5s       0  Id                int64
              SepalLengthCm    float64
              SepalWidthCm     float64
              PetalLengthCm    float64
              PetalWidthCm     float64
              Species           object
              dtype: object
```

Using Pandas: A Quick Peek

To check I view the data of the ***top n*** samples, we use the ***head(n)*** method to print the top n samples.

```
print(csv_dataframe.head(5))
```

OUTPUT

```
Time  Line #  Log Message
1.2s       0
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Using Pandas: A Quick Peek

To check I view the data of the **last n** samples, we use the **tail(n)** method to print the top n samples.

```
print(csv_dataframe.tail(5))
```

OUTPUT

```
Time  Line #  Log Message
3.6s       0
           Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
145  146            6.7           3.0           5.2           2.3
146  147            6.3           2.5           5.0           1.9
147  148            6.5           3.0           5.2           2.0
148  149            6.2           3.4           5.4           2.3
149  150            5.9           3.0           5.1           1.8

             Species
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica
```

Using Pandas: Statistical Summary

To see a descriptive statistical summary of your we can use **describe()** method

```
print(csv_dataframe.describe())
```

OUTPUT

```
Time  Line #  Log Message
1.4s       0
                Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count   150.000000     150.000000    150.000000     150.000000    150.000000
mean     75.500000       5.843333      3.054000       3.758667      1.198667
std      43.445368       0.828066      0.433594       1.764420      0.763161
min       1.000000       4.300000      2.000000       1.000000      0.100000
25%      38.250000       5.100000      2.800000       1.600000      0.300000
50%      75.500000       5.800000      3.000000       4.350000      1.300000
75%     112.750000       6.400000      3.300000       5.100000      1.800000
max     150.000000       7.900000      4.400000       6.900000      2.500000
```

Using Pandas: Selecting Subsets

It is important to learn how you can use pandas to select subsets of your data with different filters (e.g. select n number of samples, or m number of features or both, select samples with specific feature values)

This is important as part of preparing the training and test data, or manual pre-processing, etc

Pandas in general provide many powerful methods to do that.

Using Pandas: Selecting Subsets

To copy a subset of data samples into a new dataframe we need to work with the row index. In general there are two way to do that:

New_dataframe = main_dataframe[start_index : end_index]

```
# copy the samples from index 10 to 15 into a new dataframe
samples_subset = csv_dataframe[10:15]
print (samples_subset)
```

Using Pandas: Selecting Subsets

```
# copy the samples from index 10 to 15 into a new dataframe
samples_subset = csv_dataframe[10:15]
print (samples_subset)
```

OUTPUT

```
Time Line #  Log Message
1.6s       0
             Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
10  11            5.4           3.7           1.5            0.2  Iris-setosa
11  12            4.8           3.4           1.6            0.2  Iris-setosa
12  13            4.8           3.0           1.4            0.1  Iris-setosa
13  14            4.3           3.0           1.1            0.1  Iris-setosa
14  15            5.8           4.0           1.2            0.2  Iris-setosa
```

Using Pandas: Selecting Subsets

To select a subset of features (e.g. columns in the dataframe) we use the column index to do that. Pandas provide several options to access features and columns either by feature name, index or property name.

```
# copy the samples from index 10 to 15 into a new dataframe
samples_subset = csv_dataframe[10:15]

# differnt methods to select create  new data frames with selected colums or features

print(samples_subset.Species)
print(samples_subset['Species'])
print(samples_subset[['Species']])
print(samples_subset.loc[:, 'Species'])
print(samples_subset.loc[:, ['Species']])
print(samples_subset.iloc[:, 0])
print(samples_subset.iloc[:, [0]])
print(samples_subset.ix[:, 0])
```

Using Pandas: Selecting Subsets

Note: Pandas documentation recommends you use either .loc[], .iloc[], or .ix[] data access methods, which are more optimized. The .loc[] method selects by column label, .iloc[] selects by column index, and .ix[] can be used whenever you want to use a hybrid approach of either.

some of the methods take in a *list* of parameters, e.g.: df[[Species]], df.loc[:, [Species]], and df.iloc[:, [0]]. By passing in a list of parameters, you can select more than one column to slice.

Using Pandas: Selecting Subsets

Let us say we want to create a new data frame by selecting only the sample from 5 to 10 and only the 'SepalLengthCm', 'PetalLengthCm', 'Species' features

```
print(csv_dataframe.loc[5:10, ['SepalLengthCm','PetalLengthCm','Species']])
```

OUTPUT

```
ine #  Log Message
    0       SepalLengthCm  PetalLengthCm      Species
       5              5.4              1.7  Iris-setosa
       6              4.6              1.4  Iris-setosa
       7              5.0              1.5  Iris-setosa
       8              4.4              1.4  Iris-setosa
       9              4.9              1.5  Iris-setosa
       10             5.4              1.5  Iris-setosa
```

Using Pandas: Selecting Subsets with Filters

Let us say for example we want to select samples where the 'SepalLengthCm' greater than 5.0 and the 'PetalLengthCm' equal 1.5

```
SepalLengthCm_Encoding = csv_dataframe[(csv_dataframe['SepalLengthCm'] >= 5.0) & (csv_dataframe['PetalLengthCm']==1.5)]

print(SepalLengthCm_Encoding)
```

OUTPUT

```
Time Line #  Log Message
1.5s       0
        Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
7    8             5.0           3.4            1.5           0.2  Iris-setosa
10   11            5.4           3.7            1.5           0.2  Iris-setosa
15   16            5.7           4.4            1.5           0.4  Iris-setosa
19   20            5.1           3.8            1.5           0.3  Iris-setosa
21   22            5.1           3.7            1.5           0.4  Iris-setosa
27   28            5.2           3.5            1.5           0.2  Iris-setosa
31   32            5.4           3.4            1.5           0.4  Iris-setosa
32   33            5.2           4.1            1.5           0.1  Iris-setosa
39   40            5.1           3.4            1.5           0.2  Iris-setosa
48   49            5.3           3.7            1.5           0.2  Iris-setosa
```

Additional Pandas Resources

Question about this Tutorials : See me at the Lab in ELW B325

Pandas Tutorial:

http://pandas.pydata.org/pandas-docs/stable/tutorials.html

Pandas Cookbook:

http://pandas.pydata.org/pandas-docs/stable/cookbook.html