DATA SCIENCE INTERNSHIP

Session 5 & 6

HDLC TECHNOLOGIES

What is Pandas?

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Why Use Pandas?

- Pandas allows us to analyze big data and make conclusions based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.
- Relevant data is very important in data science.

What Can Pandas Do?

- Pandas gives you answers about the data. Like:
- Is there a correlation between two or more columns?
- What is average value?
- Max value?
- Min value?
- Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called *cleaning* the data.

Installation of Pandas

- pip install pandas

Pandas Series

- ```
  import pandas as pd
  a = [1, 7, 2]
  myvar = pd.Series(a)
  print(myvar)
  ```

- ```
  import pandas as pd
  a = [1, 7, 2]
  myvar = pd.Series(a, index = ["x", "y", "z"])
  print(myvar)
  ```

Pandas Series

```
import pandas as pd

calories =
{"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories)

print(myvar)
```

Data frames

- A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.
- ```
  import pandas as pd

  data = {
    "calories": [420, 380, 390],
    "duration": [50, 40, 45]
  }

  #load data into a DataFrame object:
  df = pd.DataFrame(data)

  print(df)
  ```
- `print(df.loc[0])`
- `print(df.loc[[0, 1]])`
  ```

Read CSV Files

- A simple way to store big data sets is to use CSV files (comma separated files).
- CSV files contains plain text and is a well know format that can be read by everyone including Pandas.

Read CSV

- ```
  import pandas as pd

  df = pd.read_csv('data.csv’)
  ```

- ```
  print(df)

  print(df.to_string())
  ```

Read JSON

- Big data sets are often stored, or extracted as JSON.
- JSON is plain text, but has the format of an object, and is well known in the world of programming, including Pandas.

Read JSON

```
import pandas as pd

df = pd.read_json('data.json')

print(df.to_string())
```

Dictionary as JSON

- **JSON = Python Dictionary**
- JSON objects have the same format as Python dictionaries.
- If your JSON code is not in a file, but in a Python Dictionary, you can load it into a DataFrame directly

Analyzing data

- `import pandas as pd`
- `df = pd.read_csv('data.csv')`
- `print(df.head(10))`
- `print(df.tail())`
- `print(df.info())`

Data Cleaning

- Data cleaning means fixing bad data in your data set.
- Bad data could be:
- Empty cells
- Data in wrong format
- Wrong data
- Duplicates

Sample dataset

- The data set contains some empty cells ("Date" in row 22, and "Calories" in row 18 and 28).
- The data set contains wrong format ("Date" in row 26).
- The data set contains wrong data ("Duration" in row 7).
- The data set contains duplicates (row 11 and 12).

Empty cells

- One way to deal with empty cells is to remove rows that contain empty cells.
- This is usually OK, since data sets can be very big, and removing a few rows will not have a big impact on the result.

Remove rows

```
import pandas as pd

df = pd.read_csv('data.csv')

new_df = df.dropna()

print(new_df.to_string())
```

inplace

```
import pandas as pd

df = pd.read_csv('data.csv')

df.dropna(inplace = True)

print(df.to_string())
```

Replace Empty values

```
import pandas as pd

df = pd.read_csv('data.csv')

df.fillna(130, inplace = True)
```

Replace Using Mean, Median, or Mode

```
import pandas as pd

df = pd.read_csv('data.csv')

x = df["Calories"].mean()

df["Calories"].fillna(x, inplace = True)
```

Convert to correct format

- `import pandas as pd`
- `df = pd.read_csv('data.csv')`
- `df['Date'] = pd.to_datetime(df['Date'])`
- `print(df.to_string())`

Fixing wrong data

- `df.loc[7, 'Duration'] = 45`

```
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.loc[x, "Duration"] = 120
```

```
for x in df.index:
  if df.loc[x, "Duration"] > 120:
    df.drop(x, inplace = True)
```

Removing duplicate

- `print(df.duplicated())`

- `df.drop_duplicates(inplace = True)`

Pandas - Data Correlations

- A great aspect of the Pandas module is the corr() method
- The corr() method calculates the relationship between each column in your data set.

```
            Duration     Pulse  Maxpulse  Calories
Duration    1.000000 -0.155408  0.009403  0.922721
Pulse      -0.155408  1.000000  0.786535  0.025120
Maxpulse    0.009403  0.786535  1.000000  0.203814
Calories    0.922721  0.025120  0.203814  1.000000
```

Corr() method

- The number varies from -1 to 1.
- 1 means that there is a 1 to 1 relationship (a perfect correlation), and for this data set, each time a value went up in the first column, the other one went up as well.
- 0.9 is also a good relationship, and if you increase one value, the other will probably increase as well.
- -0.9 would be just as good relationship as 0.9, but if you increase one value, the other will probably go down.
- 0.2 means NOT a good relationship, meaning that if one value goes up does not mean that the other will.

- Perfect Correlation:
- "Duration" and "Duration" got the number 1.000
- Good Correlation:
- "Duration" and "Calories" got a 0.922
- Bad Correlation:
- Duration" and "Maxpulse" got a 0.009403

Pandas plotting

- Pandas uses the plot() method to create diagrams.
- Pyplot, a submodule of the Matplotlib library to visualize the diagram on the screen.

plot

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv')

df.plot()

plt.show()
```

Scatter Plot

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv')

df.plot(kind = 'scatter', x = 'Duration', y
= 'Calories')

plt.show()
```

Histogram

- `df["Duration"].plot(kind = 'hist')`