Lecture 1: Intro to Data Science in Python

Areeb Gani, Michael Ilie, Vijay Shanmugam

Welcome!

ml.mbhs.edu

Outline

Topics

- Jupyter Notebooks
- Deepnote
- Data
- Pandas
- Linear Regression
- Scikit-Learn
- Statsmodels

Deepnote!

Jupyter Notebooks

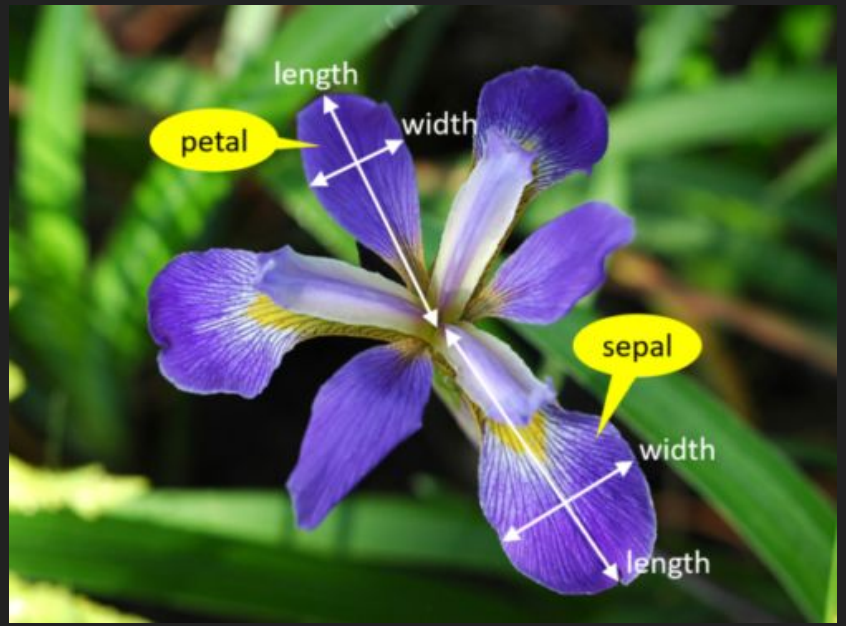
- Jupyter notebooks are the google docs of code
- Standard and widely used format:
  - Extensively used in ML world, (ie kaggle)
- Serves many purposes:
  - Can be a document for you to work in, with easy graphical control
  - Good format to present your work to others
  - Can be used interactively

Jupyter Notebooks

- Comprised of cells (blocks of code to be run)
- Run these blocks by pressing “shift +Enter” (or clicking “Run”)
- Supports Markdown (nice formatting language)
- Has magic functions which have special features

Data

- Iris dataset
  - Each entry contains petal length, petal width, sepal length, sepal width, and species



Pandas (overview)

- Library for manipulating and storing datasets
- Can be converted from files to Python/Numpy arrays, making it easy for machine learning algorithms

```
import pandas as pd
```

Pandas (basic functions)

- Data is represented in pandas “dataframe” (`pd.DataFrame`)
  - In the following slides, let `iris` be a variable holding a pandas DataFrame
- We can view first 5 rows of dataframe by calling `.head()`

	sepal_length float64	sepal_width float64	petal_length float64	petal_width float64	species object
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5	3.6	1.4	0.2	setosa

5 rows × 5 columns

```
display(iris)
```

	sepal_length float64 4.3 – 7.9	sepal_width float64 2.0 – 4.4	petal_length float64 1.0 – 6.9	petal_width float64 0.1 – 2.5	species object setosa ..... 33.3% versicolor 33.3% virginica 33.3%
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5	3.6	1.4	0.2	setosa
Expand rows 5 - 144					
145	6.7	3	5.2	2.3	virginica
146	6.3	2.5	5	1.9	virginica
147	6.5	3	5.2	2	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3	5.1	1.8	virginica

150 rows × 5 columns

Pandas (basic functions)

- We can access *shape* (rows, columns) of DataFrame using `.shape`
- We can access specific column using `[]`
  - `e.g. iris['sepal_length']`
  - `e.g. iris[['sepal_length', 'sepal_width']]`


```
iris['sepal_length']
```

```
0      5.1
1      4.9
2      4.7
3      4.6
4      5.0
      ...
145    6.7
146    6.3
147    6.5
148    6.2
149    5.9
Name: sepal_length, Length: 150, dtype: float64
```

Pandas (basic functions)

```
# 1. Column labels, and types of data in each column
iris.dtypes
```

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

```
# 2. Calculate the average petal length
iris['petal_length'].mean()
```

```
3.7580000000000005
```

```
# 3. Determine which iris species are in the dataset
iris['species'].unique()
```

```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

```
# 4. Summary of the data
iris.describe()
```

	sepal_length float64	sepal_width float64	petal_length float64	petal_width float64
count	150	150	150	150
mean	5.843333333333334	3.0573333333333337	3.7580000000000005	1.1993333333333336
std	0.828066127977863	0.4358662849366982	1.7652982332594662	0.7622376689603465
min	4.3	2	1	0.1
25%	5.1	2.8	1.6	0.3
50%	5.8	3	4.35	1.3
75%	6.4	3.3	5.1	1.8
max	7.9	4.4	6.9	2.5

Pandas (indexing)

```
virginica = iris[iris['species'] == 'virginica']
display(virginica)
```

	sepal_length float64 4.9 – 7.9	sepal_width float64 2.2 – 3.8	petal_length float64 4.5 – 6.9	petal_width float64 1.4 – 2.5	species object virginica . 100%
100	6.3	3.3	6	2.5	virginica
101	5.8	2.7	5.1	1.9	virginica
102	7.1	3	5.9	2.1	virginica
103	6.3	2.9	5.6	1.8	virginica
104	6.5	3	5.8	2.2	virginica
Expand rows 5 - 44					
145	6.7	3	5.2	2.3	virginica
146	6.3	2.5	5	1.9	virginica
147	6.5	3	5.2	2	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3	5.1	1.8	virginica

Pandas (creation)

- We can also create our own DataFrame

```
column_labels = ['A', 'B']

column_entries = [
    [1, 2],
    [4, 5],
    [7, 8]
]

pd.DataFrame(column_entries, columns=column_labels)
```

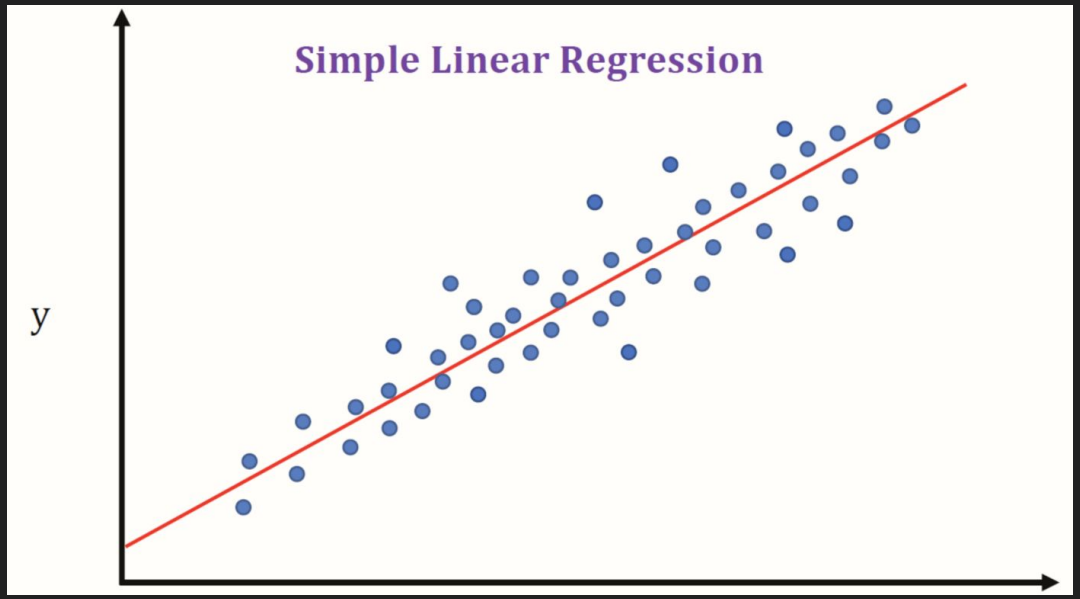
	A int64	B int64
0	1	2
1	4	5
2	7	8

Numpy

- Numerical library for Python
- Enables us to perform mathematical operations, vector and matrix operations, random distributions, etc.
- Heavily used under-the-hood in many ML libraries
- Written in C so it is very fast

Linear Regression

- Linear Regression is trying to predict a linear relationship between data points
  - Image below is 2D, but we can generalize to n-dimensions
- Will cover math and methods behind it next week



Scikit Learn

- Scikit-learn is one of the most popular machine learning libraries in Python
- Easy-to-use, but still very powerful - not as complicated as TF/PyTorch
- Models are pre-programmed in classes

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(x,y)
```

Statsmodels

- Statsmodels is a data science library that uses statistical methods
- Similar interface to Scikit-learn, although can be more complex with detailed metrics - similar to R

```
import statsmodels.api as sm
lm      = sm.OLS(y,x)
results = lm.fit()
```

Join Our Groups

- Sign up for Discord (https://discord.gg/3Z5YuPqt)
- Join Deepnote (https://deepnote.com/join-team?token=af3af0284bc8497)
- Fill out our form (https://forms.gle/Fr31aFLWx8cHdtTY8)
  - Join mailing list + Github organization

- Next week: Linear Regression + Logistic Regression