Deep Learning Prerequisites Tutorials

CS 5102 Deep Learning

Agenda

Introduction to Python

Introduction to Data Analysis using NumPy and Pandas

Introduction to Data Visualization using Matplotlib and Seaborn

Data Types and Data Structures in Python

Data Types

- Numbers
- Strings

Data Structures

- Dictionaries
- Lists
- Tuples
- Sets
- NumPy Arrays
- Pandas Series and DataFrames

Numbers

```
In [1]: 1 + 1 # Addition
Out[1]: 2

In [2]: 2 * 2 # Multiplication
Out[2]: 4

In [3]: 100 / 2 # Division
Out[3]: 50.0

In [4]: 2 ** 6 # Exponent / Raising to a Power
Out[4]: 64

In [5]: 54 % 3 # Modulus or Remainder Operator
Out[5]: 0

In [6]: (34 + 2) * (7 - 8) # Working with Parenthesis
Out[6]: -36
```

Strings

```
In [7]: "Double Quotes"
Out[7]: 'Double Quotes'

In [8]: 'Single Quotes'
Out[8]: 'Single Quotes'

In [9]: x = 'Deep '

In [10]: y = "Learning"

In [11]: print(x + y)
Deep Learning

In [12]: code = 'CS 5102'

In [13]: course = 'Deep Learning'

In [14]: print('Course Code is: {one}, and Course Name is: {two}'.format(one = code, two = course))
Course Code is: CS 5102, and Course Name is: Deep Learning
```

Dictionaries

```
In [31]: d = {'key1':'item1','key2':'item2','key3':'item3'}

In [32]: d
Out[32]: {'key1': 'item1', 'key2': 'item2', 'key3': 'item3'}

In [33]: d.keys()
Out[33]: dict_keys(['key1', 'key2', 'key3'])

In [34]: d.values()
Out[34]: dict_values(['item1', 'item2', 'item3'])

In [35]: d.clear()

In [36]: d
Out[36]: {}
```

Lists

```
In [15]: [1, 2, 3]
Out[15]: [1, 2, 3]

In [16]: [1, 2, [3,4]]
Out[16]: [1, 2, [3, 4]]

In [17]: ['one', 2, "3"]
Out[17]: ['one', 2, '3']

In [18]: my_list = [1, 2, 3]

In [19]: my_list
Out[19]: [1, 2, 3]

In [20]: my_list = [1,2,3,[4, 5, 6, [7, 8]]]

In [21]: my_list
Out[21]: [1, 2, 3, [4, 5, 6, [7, 8]]]

In [22]: my_list[3][3][1]
Out[22]: 8
```

Lists

```
In [23]: my_list = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

In [24]: my_list[0]
Out[24]: 1

In [25]: my_list[1:]
Out[25]: [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]

In [26]: my_list[1:5]
Out[26]: [2, 3, 4, 5]

In [27]: my_list[:-5]
Out[27]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

In [28]: my_list[-8:]
Out[28]: [13, 14, 15, 16, 17, 18, 19, 20]

In [29]: my_list[0] = 1000

In [30]: my_list
Out[30]: [1000, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

Tuples

```
In [53]: t = (1, 2, 3, 4, 5, 6, 7, 8, 'nine', 'ten', '11')

In [54]: t
Out[54]: (1, 2, 3, 4, 5, 6, 7, 8, 'nine', 'ten', '11')

In [55]: t.index(4) # return first index of value. Raises ValueError if the value is not present.
Out[55]: 3

In [56]: t.count(3) # return number of occurrences of value
Out[56]: 1

In [57]: t[5]
Out[57]: 6

In [58]: t[5] = 11
Traceback (most recent call last):

  File "<ipython-input-58-f284916c681a>", line 1, in <module>
    t[5] = 11

TypeError: 'tuple' object does not support item assignment
```

Sets

```
In [59]: t = {1,2,3}

In [60]: t
Out[60]: {1, 2, 3}

In [61]: s = {3,4,5,5,5,5,5,5,5,5,5,5,5,5,5}

In [62]: s
Out[62]: {3, 4, 5}

In [63]: t.intersection(s)
Out[63]: {3}

In [64]: t.union(s)
Out[64]: {1, 2, 3, 4, 5}

In [65]: t.issubset(s)
Out[65]: False
```

Sets

```
In [67]: t[4]
Traceback (most recent call last):

  File "<ipython-input-67-5977d405d0f2>", line 1, in <module>
    t[4]

TypeError: 'set' object does not support indexing


In [68]:

In [68]: t[4] = 5
Traceback (most recent call last):

  File "<ipython-input-68-fa7207573daf>", line 1, in <module>
    t[4] = 5

TypeError: 'set' object does not support item assignment
```

Conditional Statements - if, elif, else

```
In [73]: x = 5

In [74]: y = 7

In [75]: if x < y:
    ...:     print('x is greater')
    ...: elif x > y:
    ...:     print('y is greater')
    ...: else:
    ...:     print('X is equal to y')
    ...:
x is greater

In [76]:
```

For Loop

```
In [76]: items = [1,2,3,4,5,6,7,8]

In [77]: for item in items:
    ...:     print(str(item ** item))
    ...:
1
4
27
256
3125
46656
823543
16777216
```

```
In [81]: for i in range(0, len(items)):
    ...:     print(i)
    ...:
0
1
2
3
4
5
6
7
```

While Loop

```
In [85]: value = 100

In [86]: while value >= 0:
    ...:     print(value)
    ...:     value = value - 20
    ...:
100
80
60
40
20
0

In [87]: while True:
    ...:     print('infinity')
    ...:
```

Functions

```
In [89]: def take_square(value = 3):
    ...:     """
    ...:     This is a function which returns the square of a number
    ...:     """
    ...:     return value ** 2
    ...:

In [90]: print(take_square(7))
49

In [91]:
```

Data Analysis

Numpy Arrays essentially come in two flavors: vectors and matrices.

1. **Vectors** are strictly 1-D arrays and
2. **Matrices** are 2-D (but you should note a matrix can still have only one row or one column).

Creating Numpy Arrays from a List

```
In [91]: import numpy as np

In [92]: mylist = [1,2,3] # Creating a numpy array from a list

In [93]: arr = np.array(mylist)

In [94]: arr
Out[94]: array([1, 2, 3])

In [95]: my_matrix = [[1,2,3],[4,5,6],[7,8,9]]

In [96]: arr = np.array(my_matrix)

In [97]: arr
Out[97]:
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Creating Numpy Arrays using Built-in Methods

```
In [98]: np.arange(0,11,2)
Out[98]: array([ 0,  2,  4,  6,  8, 10])

In [99]: arr = np.zeros(3)

In [100]: arr
Out[100]: array([0., 0., 0.])

In [101]: np.zeros((5,5))
Out[101]:
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])

In [102]: np.ones((2,2))
Out[102]:
array([[1., 1.],
       [1., 1.]])
```

Creating Numpy Arrays using Built-in Methods

```
In [106]: np.linspace(0,10,3) # Return evenly spaced numbers over a specified interval.
Out[106]: array([ 0.,  5., 10.])

In [107]: np.eye(5)
Out[107]:
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])

In [108]: np.random.rand(5,5)
Out[108]:
array([[0.48921464, 0.31836927, 0.86728359, 0.93446508, 0.57683906],
       [0.08389133, 0.60626255, 0.58837947, 0.63650733, 0.38455093],
       [0.77710136, 0.9121635 , 0.43595676, 0.44678391, 0.01961414],
       [0.5382528 , 0.55208194, 0.4112583 , 0.0389055 , 0.62487892],
       [0.05494904, 0.11535842, 0.20499799, 0.04588116, 0.64492743]])
```

Reshaping an Array

```
In [109]: arr = np.arange(25)

In [110]: arr
Out[110]:
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24])

In [111]: arr.reshape(5,5)
Out[111]:
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

Numpy Array Operations

```
In [119]: arr.shape
Out[119]: (25,)

In [120]: arr.min()
Out[120]: 0

In [121]: arr.mean()
Out[121]: 12.0

In [122]: arr.max()
Out[122]: 24

In [123]: arr.dtype
Out[123]: dtype('int32')
```

Array Indexing

```
In [124]: arr[4]
Out[124]: 4

In [125]: arr[4:9]
Out[125]: array([4, 5, 6, 7, 8])

In [126]: arr[4:9] * 10
Out[126]: array([40, 50, 60, 70, 80])

In [127]: arr
Out[127]:
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24])

In [128]: arr[4:9] = 100

In [129]: arr
Out[129]:
array([  0,   1,   2,   3, 100, 100, 100, 100, 100,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24])
```

Array Operations using Indexing

```
In [133]: new_arr = np.array([a for a in arr * 10])

In [134]: new_arr
Out[134]:
array([   0,   10,   20,   30, 1000, 1000, 1000, 1000, 1000,   90,  100,
        110,  120,  130,  140,  150,  160,  170,  180,  190,  200,  210,
        220,  230,  240])
```

Array Operations using Indexing

```
In [135]: arr
Out[135]:
array([  0,   1,   2,   3, 100, 100, 100, 100, 100,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24])

In [136]: arr[:]
Out[136]:
array([  0,   1,   2,   3, 100, 100, 100, 100, 100,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24])

In [137]: arr[:5]
Out[137]: array([  0,   1,   2,   3, 100])

In [138]: arr[:-5]
Out[138]:
array([  0,   1,   2,   3, 100, 100, 100, 100, 100,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19])

In [139]: arr[-5:]
Out[139]: array([20, 21, 22, 23, 24])
```

2D Array Operations using Indexing

```
In [141]: arr = np.array(([5,10,15],[20,25,30],[35,40,45]))

In [142]: arr
Out[142]:
array([[ 5, 10, 15],
       [20, 25, 30],
       [35, 40, 45]])

In [143]: arr[1]
Out[143]: array([20, 25, 30])

In [144]: arr[1][1]
Out[144]: 25

In [145]: arr[1,2]
Out[145]: 30

In [146]: arr[:1,:2]
Out[146]: array([[ 5, 10]])
```

2D Array Operations

```
In [147]: arr * 10
Out[147]:
array([[ 50, 100, 150],
       [200, 250, 300],
       [350, 400, 450]])

In [148]: arr / 10
Out[148]:
array([[0.5, 1. , 1.5],
       [2. , 2.5, 3. ],
       [3.5, 4. , 4.5]])

In [149]: arr
Out[149]:
array([[ 5, 10, 15],
       [20, 25, 30],
       [35, 40, 45]])

In [150]: arr > 25
Out[150]:
array([[False, False, False],
       [False, False,  True],
       [ True,  True,  True]])
```

Array Operations

```
In [151]: np.sqrt(arr)
Out[151]:
array([[2.23606798, 3.16227766, 3.87298335],
       [4.47213595, 5.        , 5.47722558],
       [5.91607978, 6.32455532, 6.70820393]])

In [152]: np.exp(arr)
Out[152]:
array([[1.48413159e+02, 2.20264658e+04, 3.26901737e+06],
       [4.85165195e+08, 7.20048993e+10, 1.06864746e+13],
       [1.58601345e+15, 2.35385267e+17, 3.49342711e+19]])

In [153]: np.sin(arr)
Out[153]:
array([[-0.95892427, -0.54402111,  0.65028784],
       [ 0.91294525, -0.13235175, -0.98803162],
       [-0.42818267,  0.74511316,  0.85090352]])
```

Array Operations

```
In [154]: np.cos(arr)
Out[154]:
array([[ 0.28366219, -0.83907153, -0.75968791],
       [ 0.40808206,  0.99120281,  0.15425145],
       [-0.90369221, -0.66693806,  0.52532199]])

In [155]: np.log(arr)
Out[155]:
array([[1.60943791, 2.30258509, 2.7080502 ],
       [2.99573227, 3.21887582, 3.40119738],
       [3.55534806, 3.68887945, 3.80666249]])

In [156]: np.ceil(np.log(arr))
Out[156]:
array([[2., 3., 3.],
       [3., 4., 4.],
       [4., 4., 4.]])
```

Data Analysis

Pandas comes in two flavours, Series and DataFrames.

Series

```
In [171]: import numpy as np

In [172]: import pandas as pd

In [173]: my_list = [1,2,3,4,5,6,7,8]

In [174]: my_arr = np.array(my_list)

In [175]: my_series = pd.Series(my_arr)

In [176]: my_arr
Out[176]: array([1, 2, 3, 4, 5, 6, 7, 8])

In [177]: my_series
Out[177]:
0    1
1    2
2    3
3    4
4    5
5    6
6    7
7    8
dtype: int32
```

Series

```
In [179]: labels = ['a','b','c','d','e','f','g','h']

In [180]: my_series = pd.Series(data=my_arr, index=labels)

In [181]: my_series
Out[181]:
a    1
b    2
c    3
d    4
e    5
f    6
g    7
h    8
dtype: int32
```

Series

```
In [184]: my_series.isnull()
Out[184]:
a    False
b    False
c    False
d    False
e    False
f    False
g    False
h    False
dtype: bool

In [185]: my_series.dropna()
Out[185]:
a    1
b    2
c    3
d    4
e    5
f    6
g    7
h    8
dtype: int32
```

DataFrames

```
In [204]: import pandas as pd

In [205]: import numpy as np

In [206]: from numpy.random import randn

In [207]: index1 = ['a','b','c','d']

In [208]: index2 = ['e','f','g','h']

In [209]: df = pd.DataFrame(randn(4,4), index = index1, columns = index2)

In [210]: df
Out[210]:
          e         f         g         h
a  0.716012 -0.458829  0.024247 -1.178026
b  1.284239 -0.202902  0.424135 -1.926622
c  0.764377  0.758341 -0.600992 -0.405615
d -0.002138 -0.373414 -0.233751 -0.096857
```

Pandas - Working with CSV

```
In [163]: import pandas as pd

In [164]: train = pd.read_csv('G:/Titanic/train.csv')

In [165]: train.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

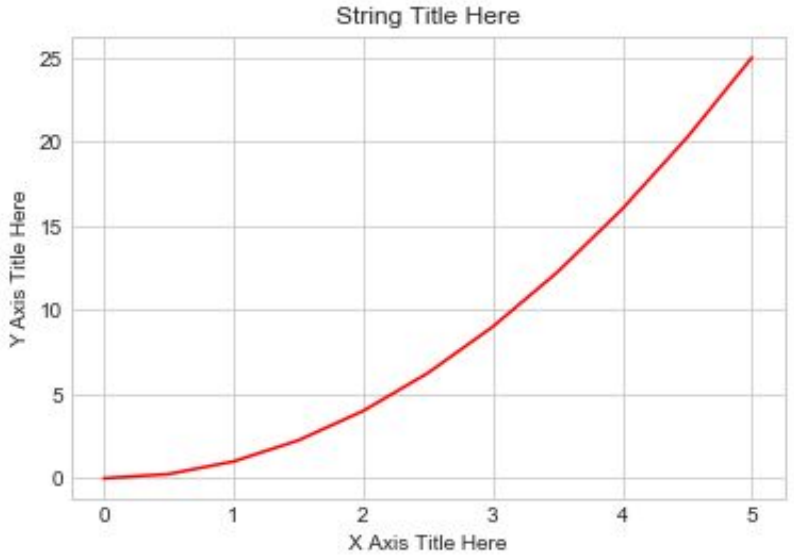
Data Visualization

A brief introduction to Data Visualization using Matplotlib and Seaborn Libraries.

Matplotlib

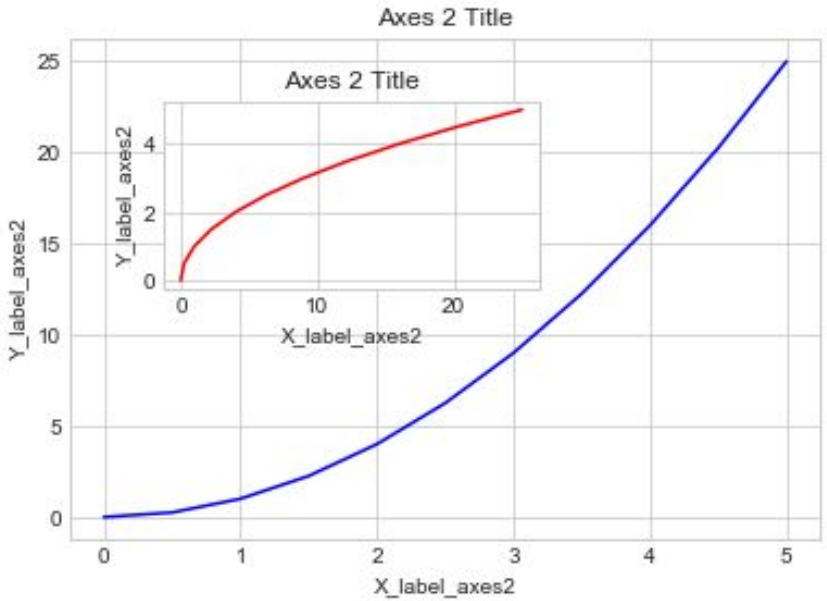
```
In [230]: import numpy as np
     ...: import matplotlib.pyplot as plt
     ...: x = np.linspace(0, 5, 11)
     ...: y = x ** 2
     ...:

In [231]: plt.plot(x, y, 'r') # 'r' is the color red
     ...: plt.xlabel('X Axis Title Here')
     ...: plt.ylabel('Y Axis Title Here')
     ...: plt.title('String Title Here')
     ...: plt.show()
     ...:
```



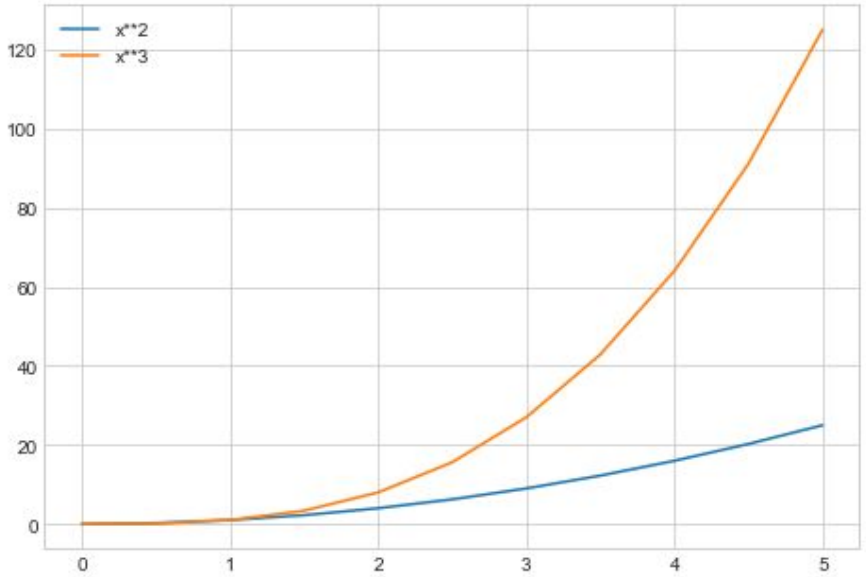
Matplotlib

```
In [232]: # Creates blank canvas
     ...: fig = plt.figure()
     ...:
     ...: axes1 = fig.add_axes([0.1, 0.1, 0.8, 0.8]) # main axes
     ...: axes2 = fig.add_axes([0.2, 0.5, 0.4, 0.3]) # inset axes
     ...:
     ...: # Larger Figure Axes 1
     ...: axes1.plot(x, y, 'b')
     ...: axes1.set_xlabel('X_label_axes2')
     ...: axes1.set_ylabel('Y_label_axes2')
     ...: axes1.set_title('Axes 2 Title')
     ...:
     ...: # Insert Figure Axes 2
     ...: axes2.plot(y, x, 'r')
     ...: axes2.set_xlabel('X_label_axes2')
     ...: axes2.set_ylabel('Y_label_axes2')
     ...: axes2.set_title('Axes 2 Title');
     ...:
```



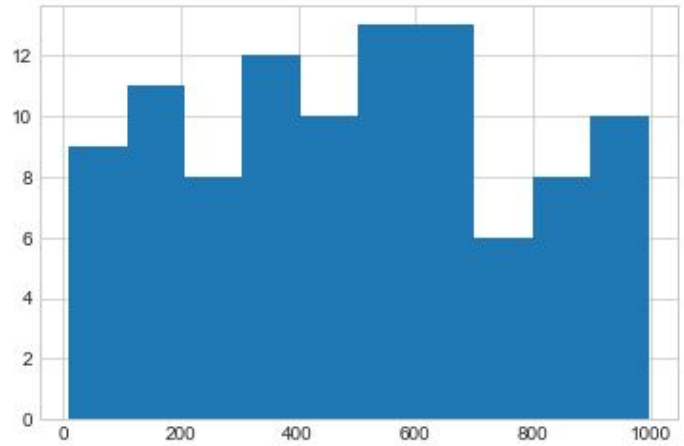
Matplotlib

```
In [233]: fig = plt.figure()
     ...:
     ...: ax = fig.add_axes([0,0,1,1])
     ...:
     ...: ax.plot(x, x**2, label="x**2")
     ...: ax.plot(x, x**3, label="x**3")
     ...: ax.legend()
     ...:
Out[233]: <matplotlib.legend.Legend at 0x1969c6f3c50>
```



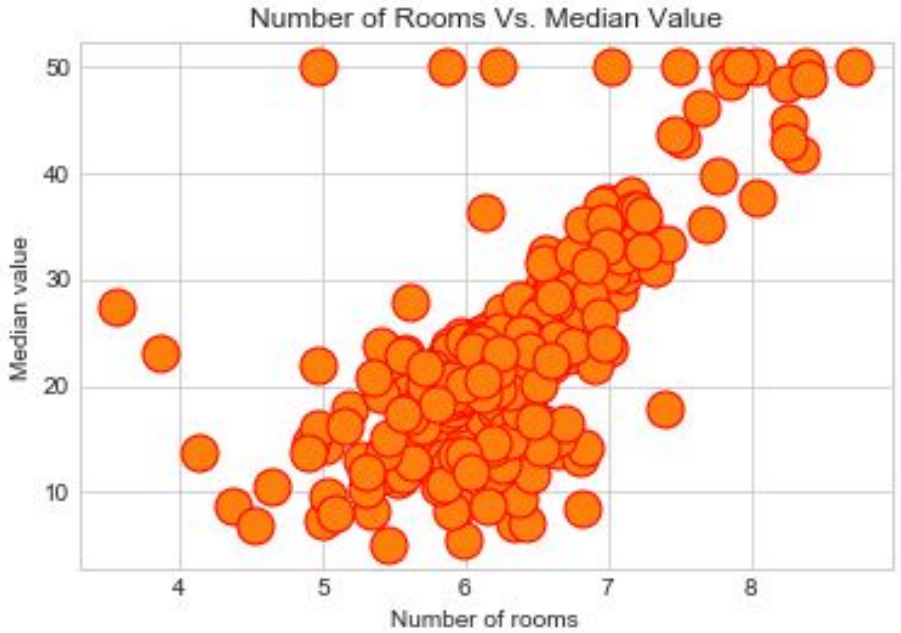
Matplotlib

```
In [234]: from random import sample
     ...: data = sample(range(1, 1000), 100)
     ...: plt.hist(data)
     ...:
Out[234]:
(array([ 9., 11.,  8., 12., 10., 13., 13.,  6.,  8., 10.]),
 array([ 10. , 108.6, 207.2, 305.8, 404.4, 503. , 601.6, 700.2, 798.8,
        897.4, 996. ]),
 <a list of 10 Patch objects>)
```



Matplotlib

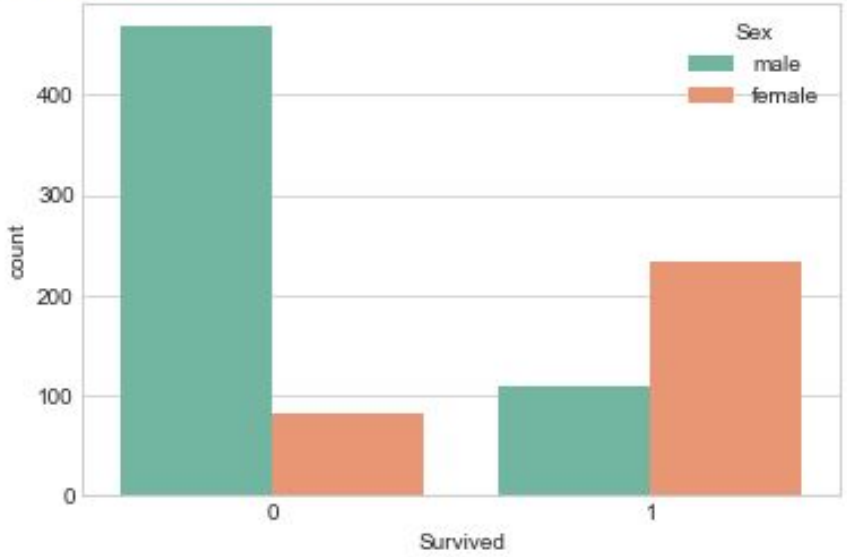
```
In [238]: df = pd.read_csv('G:/Boston Housing/train.csv')
     ...:
     ...: # medv: median value of owner-occupied homes in \$1000s.
     ...: # rm: average number of rooms per dwelling.
     ...: houses = plt.plot(df['rm'], df['medv'], 'ro')
     ...:
     ...: plt.title('Number of Rooms Vs. Median Value')
     ...: plt.xlabel('Number of rooms')
     ...: plt.ylabel('Median value')
     ...:
     ...: plt.setp(houses, markersize=15)
     ...: plt.setp(houses, markerfacecolor='C1')
     ...:
     ...: plt.show()
```



Seaborn

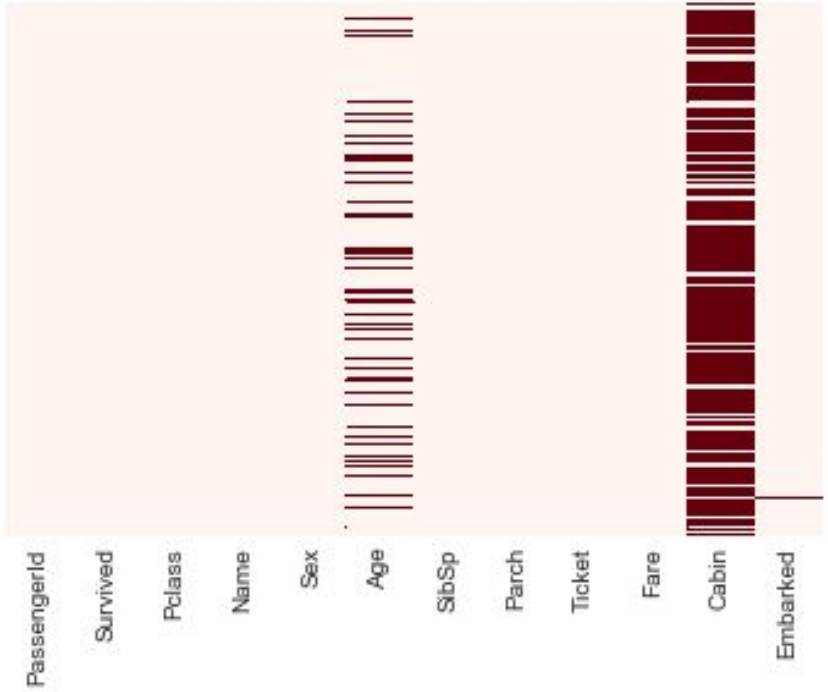
```
In [220]: import seaborn as sns

In [221]: # Checking how many survived vs. how many did not with respect to gender.
     ...: sns.set_style('whitegrid')
     ...: sns.countplot(x='Survived',hue='Sex',data=train,palette='Set2')
     ...:
Out[221]: <matplotlib.axes._subplots.AxesSubplot at 0x1969d29f6d8>
```



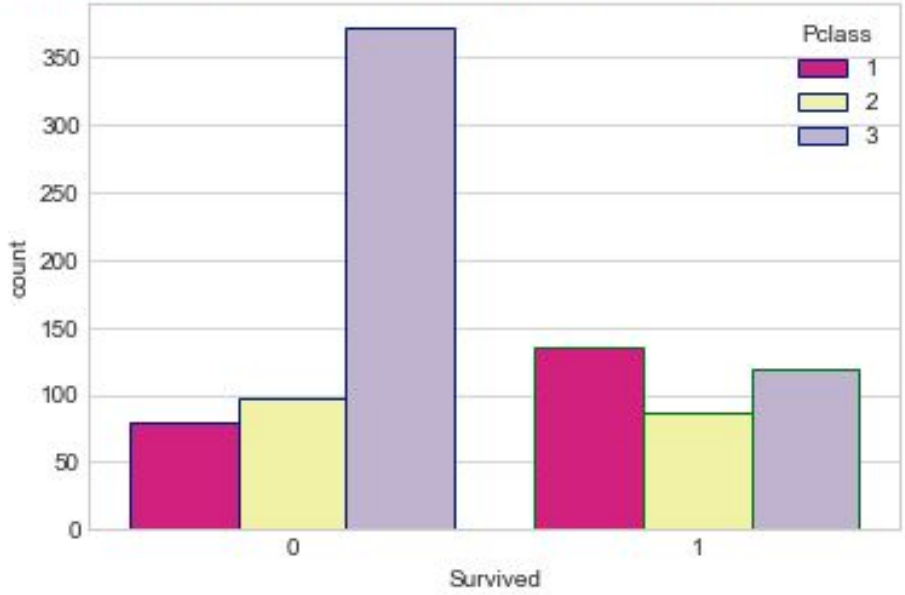
Seaborn

```
In [222]: # The following Heatmap will reveal the missing values.
     ...: # White lines indicate the missing values.
     ...: sns.heatmap(train.isnull(), yticklabels=False, cbar=False, cmap="Reds")
Out[222]: <matplotlib.axes._subplots.AxesSubplot at 0x1969d3ab9e8>
```



Seaborn

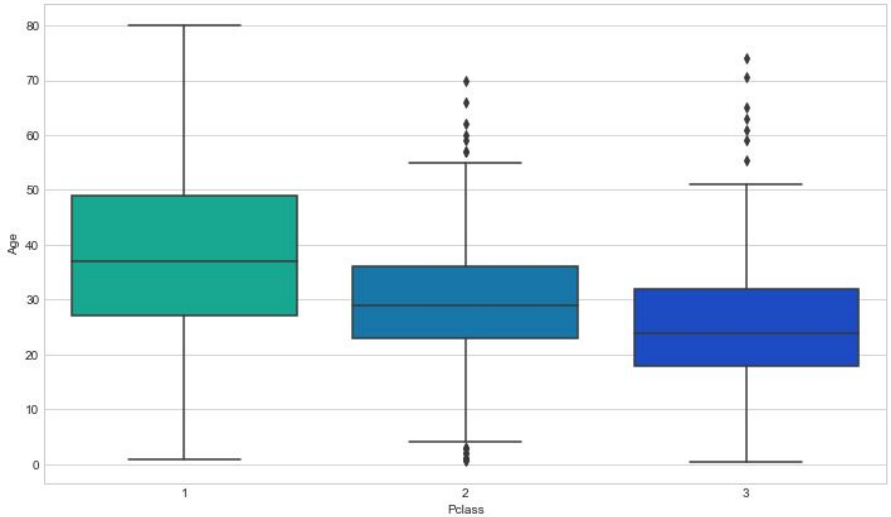
```
In [225]: # Checking how many survived vs. how many did not with respect to class.
     ...: sns.set_style('whitegrid')
     ...: sns.countplot(x='Survived',hue='Pclass',data=train,palette='Accent_r',
     ...:               edgecolor=sns.color_palette("dark", 3))
     ...:
Out[225]: <matplotlib.axes._subplots.AxesSubplot at 0x1969d54b710>
```



Seaborn

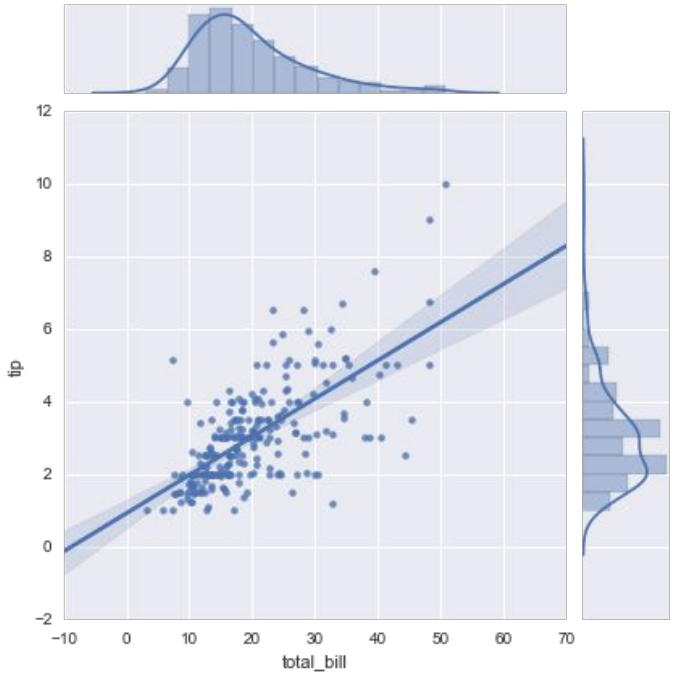
```
In [227]: import matplotlib.pyplot as plt

In [228]: # Checking the age groups of the people within each class.
     ...: # Grouped into classes
     ...: plt.figure(figsize=(12, 7))
     ...: sns.boxplot(x='Pclass',y='Age',data=train,palette='winter_r')
     ...:
Out[228]: <matplotlib.axes._subplots.AxesSubplot at 0x1969d4a3ba8>
```



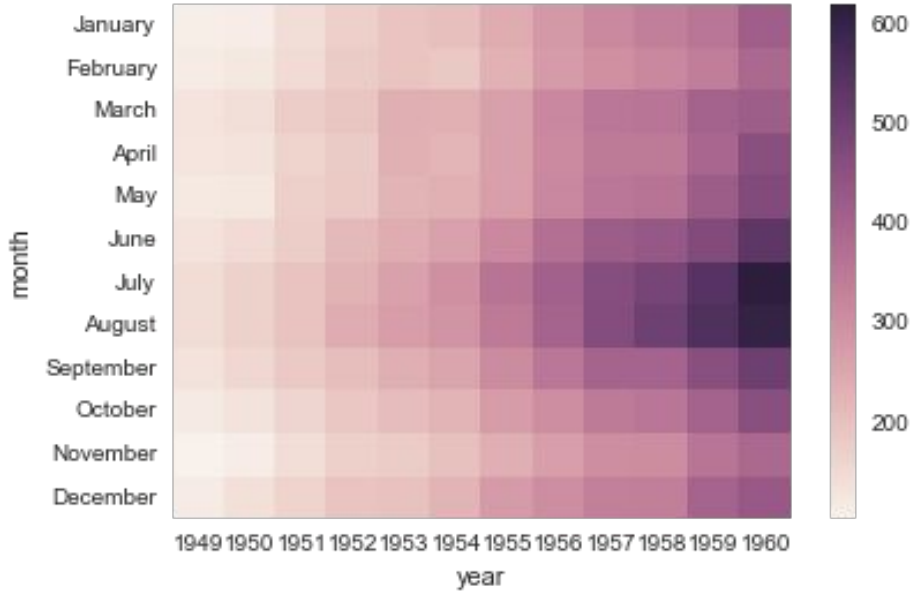
Seaborn

```
In [235]: g = sns.JointGrid(x="total_bill", y="tip", data=tips)
     ...: g = g.plot(sns.regplot, sns.distplot)
     ...:
```



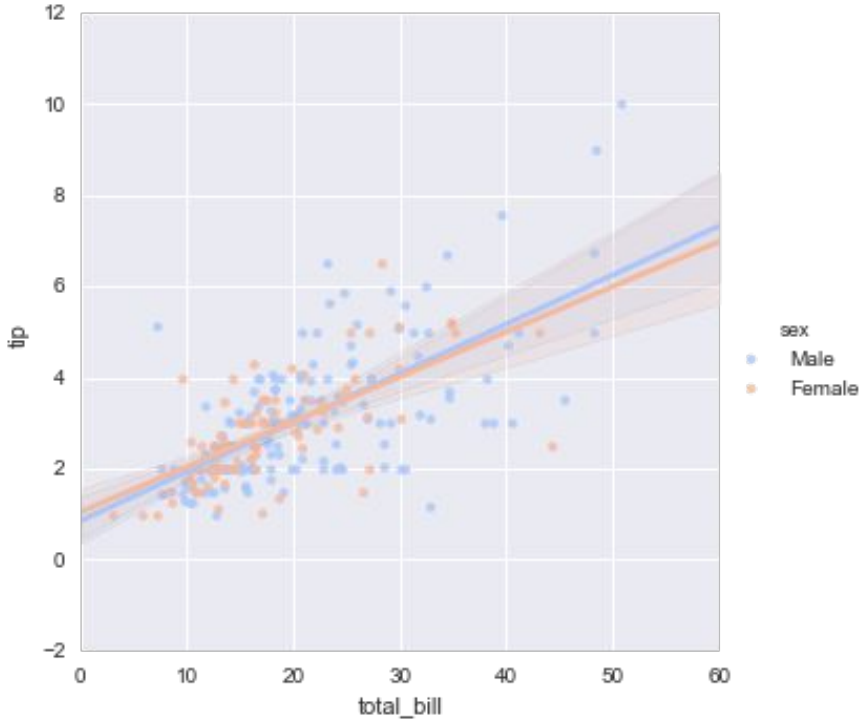
Seaborn

```
In [236]: pvflights = flights.pivot_table(values='passengers',index='month',columns='year')
     ...: sns.heatmap(pvflights)
     ...:
```



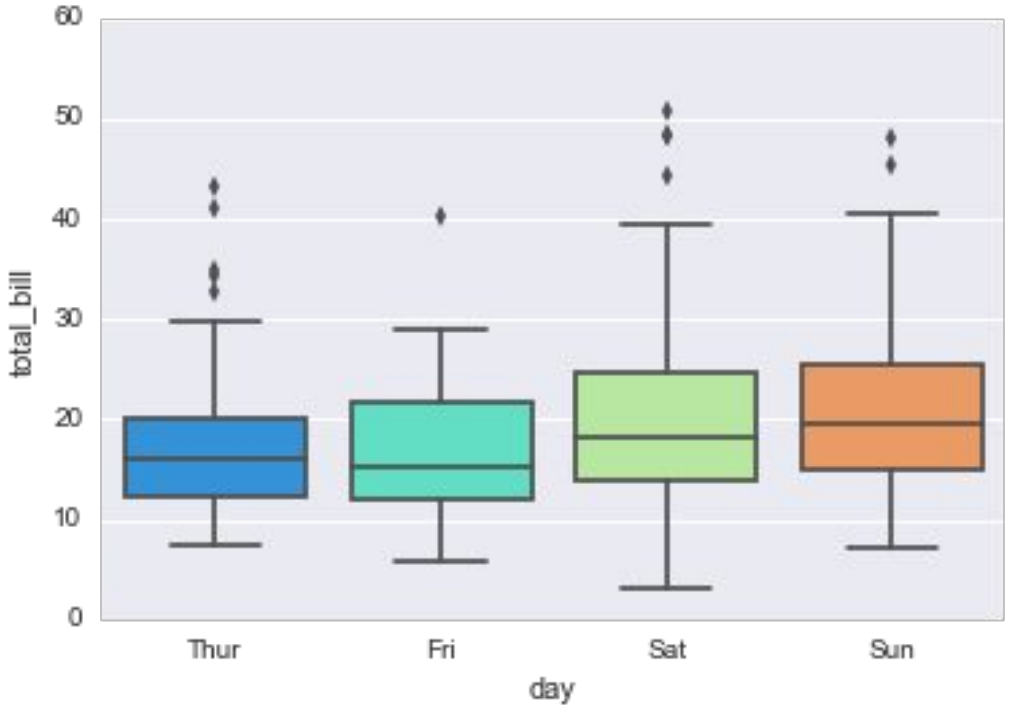
Seaborn

sns.lmplot(x='total_bill',y='tip',data=tips,hue='sex',palette='coolwarm')



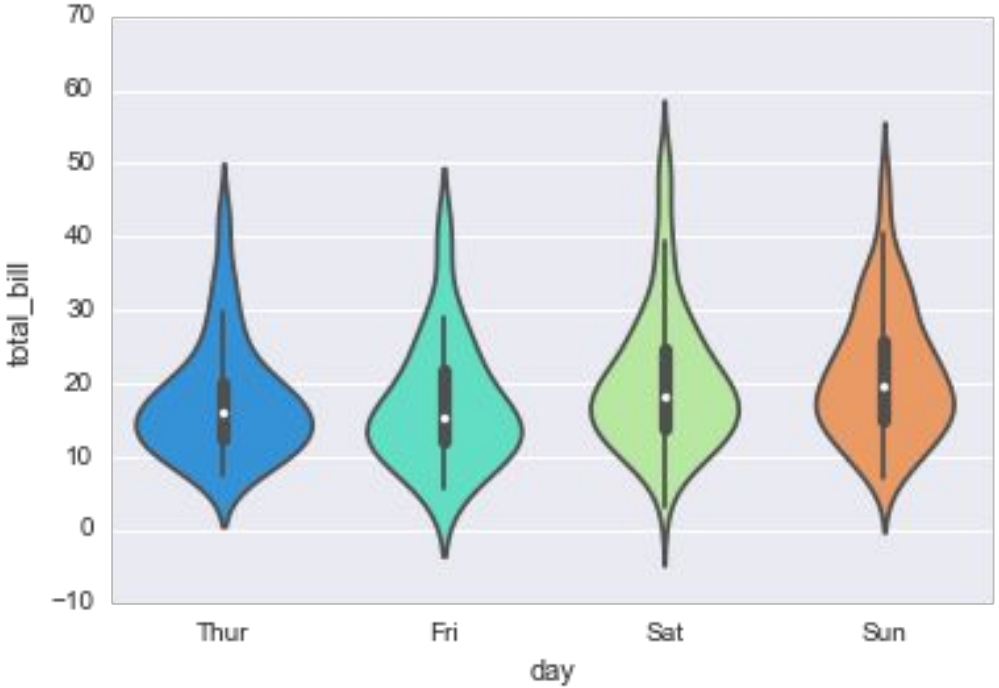
Seaborn

sns.boxplot(x="day", y="total_bill", data=tips,palette='rainbow')



Seaborn

sns.violinplot(x="day", y="total_bill", data=tips,palette='rainbow')



End of the First Workshop.

Contact: taz.taimur@gmail.com

Workshop Content can be found on
https://github.com/taimurzahid/FASTDeepLearning