**Machine Learning Notes**

***Namrata Acharya & Neil Gupta***

**Pandas**

* pandas is a column-oriented data analysis API
* tool for handling and analyzing input data

Basic Concepts:

Importing pandas:

from \_\_future\_\_ import print\_function

import pandas as pd

pd.\_\_version\_\_

* Primary data structures
  + Data Frame - a relational data table, with rows and named columns, made of series
  + Series - single column of data
* Creating Series: pd.Series(['San Francisco', 'San Jose', 'Sacramento'])
* Creating Data frame: pd.DataFrame({ 'City name': city\_names, 'Population': population })
* Can load files and read data
* Can use graphing to view data

Accessing & Manipulating Data

* cities = pd.DataFrame({ 'City name': city\_names, 'Population': population })

print(type(cities['City name']))

cities['City name']

* NumPy used for scientific computing
* import numpy as np

np.log(population)

* Creates a new Series applied to each value:
* population.apply(lambda val: val > 1000000)
* Modifying data frames:

cities['Area square miles'] = pd.Series([46.87, 176.53, 97.92])

cities['Population density'] = cities['Population'] / cities['Area square miles']

cities

* Creating a new Series based on previous information like above or as in Example 1
* Reorders data frame: DataFrame.reindex([num1,num2,num3,...])
* Missing indices makes it easier to reindex using an internal list

<https://dev.to/rodmsmendes/create-your-first-machine-learning-model-in-5-minutes-with-google-colab-4o6h>

**Applying Machine Learning:**

0:

import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix

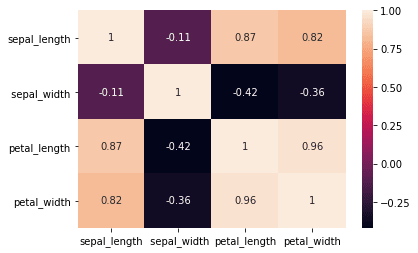
from joblib import dump, load

1. Load the data

cols = ['sepal\_length', ' sepal\_width', 'petal\_length', 'petal\_width', 'class']

df = pd.read\_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data', names=cols)

df.head()

1. Explore and visualize
   1. Exploratory data analysis
   2. The describe() method prints statistics of the dataset, like mean, standard deviation
   3. df.describe()
   4. Data visualization
      1. sns.pairplot(df, hue='class');
   5. Heat map allows you to visualize correlation matrix of data
      1. sns.heatmap(df.corr(), annot=True)
      2. 
2. Preprocess the data
   1. Datasets from databases must be prepped to be analyzed by the algorithm
   2. dataset needs some kind of preparation or preprocessing before being used as input to a machine learning algorithm
   3. If algorithm does not process string values, convert to integers
      1. df['class\_encod'] = df['class'].apply(lambda x: 0 if x == 'Iris-setosa' else 1 if x == 'Iris-versicolor' else 2)
      2. df['class\_encod'].unique()
3. Select an algorithm and train the model
   1. split dataframe into input attributes and target attributes
      1. y = df[['class\_encod']] # target attributes

X = df.iloc[:, 0:4] # input attributes

X.head()

* 1. Split data by rows
     1. train\_test\_split() will split the X and y dataframes in training data and test data
     2. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3,

random\_state=0, stratify=y)

np.shape(y\_train)

* 1. use the datasets X\_train and y\_train to build a KNN classifier, using the KNeighborsClassifier class provided by scikit-learn
     1. m = KNeighborsClassifier()

m.fit(X\_train, np.ravel(y\_train))

* 1. use the predict() method to calculate the predicted category of an instance
     1. predict the class of the first 10 lines of the X\_test dataset:
     2. Returns array with estimated categories
     3. m.predict(X\_test.iloc[0:10])
  2. score() and confusion\_matrix() measure the performance of model
     1. m.score(X\_test, y\_test)
     2. confusion\_matrix(y\_test, m.predict(X\_test))

1. Save the model for future test
   1. Dump saves model to a file

dump(m, 'iris-classifier.dmp')

ic = load('iris-classifier.dmp')

confusion\_matrix(y\_test, ic.predict(X\_test))

* Can be reproduced for many data sets

**Introduction to TensorFlow:**

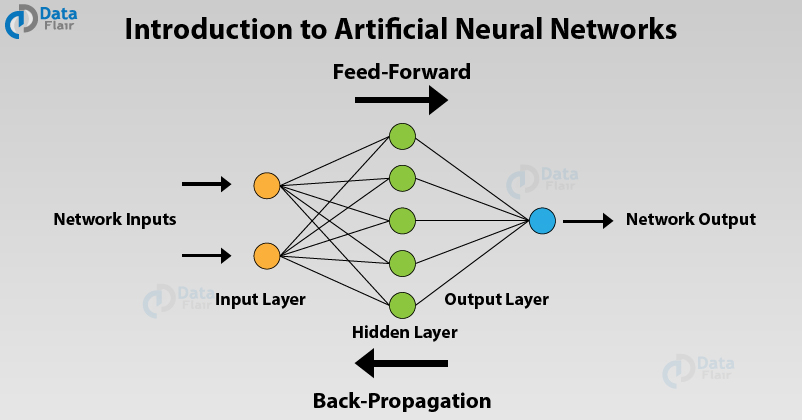
* TensorFlow is a great piece of software and currently the leading deep learning and
* neural network computation framework.
* It is based on a C++ low level backend but is usually controlled via Python (there is also

a neat TensorFlow library for R, maintained by RStudio).

* TensorFlow operates on a graph representation of the underlying computational task. This approach allows the user to specify mathematical operations as elements in a graph of data, variables and operators. Since neural networks are actually graphs of data and mathematical operations,
* TensorFlow is just perfect for neural networks and deep learning.

**Neural Networks and Relation to Machine Learning:**

* Neural networks utilize numerous examples known as training examples. By doing this, we can create a system which can learn from those training examples.
* An example is classifying handwritten digits
* Some machine learning algorithms are trained off of thousands, millions or even billions of training examples.



**Perceptrons:**

* A perceptron is an algorithm for supervised learning of binary classifiers. More common
* A perceptron takes several binary inputs, x1,x2,…, and produces a single binary output
* Terms called *weights →*  real numbers expressing the importance of the respective inputs to the output.
* Perceptrons are a device that makes decisions by weighing up evidence.

**Sigmoid neurons (Training the system using bias)**

* The computer needs to learn how to make distinctions between small changes in the input. (for example, the diff between a ‘8’ and a ‘9’)
* A network that doesn’t work can be fixed by making a small change in the weights and biases.
  + By adjusting the network over and over, the machine is learning
* Artificial neurons called *sigmoid neurons* are modified so small changes in their *weights* cause only a small change in their output. That will allow a network of sigmoid neurons to learn.