**Machine Learning**

1. Learn Python

Download Jupyter Notebook at : <https://jupyter.org/>

* **NB**: It is important to get familiar with Jupyter Notebook from the start as this will be used mainly in machine learning **!!!**

Get familiar with python at:

* <https://www.python.org/>
* <https://www.codecademy.com/learn/paths/computer-science>

Useful videos for introduction to python, pandas and machine learning:

* <https://www.youtube.com/channel/UCnVzApLJE2ljPZSeQylSEyg>

**A. Important Libraries**

Make sure you understand the first three libraries before starting SciKit-learn.

1. **Pandas**

For dealing with input and output files in the form of tables, csv, excel etc. For a detailed step by step explanation of Pandas:

* [**http://pandas.pydata.org/pandas-docs/stable/reference/index.html**](http://pandas.pydata.org/pandas-docs/stable/reference/index.html)
* Always start with: **import pandas as pd** :this will import the Pandas library in the name pd

Below is a list of important commands in Pandas

|  |  |
| --- | --- |
| **Command** | **Description** |
| fid = open(‘path.txt’, ‘w’) | Opening a text file and saving it in the name of your choice, here it is saved as fid  w for writing to file and r for reading file |
| fid.write(‘string’) | Writing to a text file |
| fid.close() | Close file |
| fid.readline() | Reads from text file |
| pd.read\_csv(‘path’,parameters) | For reading comma separated values (csv) file into a data frame  Parameters: <http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html#pandas.read_csv>  Important arguments   * sep = ‘\t’ or sep = ‘|’ if in table * header = none * names = list (arraylike) * skip\_blank\_lines =True * index\_cols = False (forces pandas to not use the first column as index) * usercols=[0,n] (reads columns 0-4) * nrows=3, reads the first 3 rows |
| pd.read\_fwf(‘path’,parameters) | For reading a table fixed with formatted lines into a data frame  Parameters: <http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_fwf.html#pandas.read_fwf> |
| read\_excel(r’io[,sheet\_name,header,names])  ExcelFile.parse([sheet\_name,header,names,..])  ExcelWriter(path[,engine,date\_format,mode = ‘w/a’…]) | Read an Excel file into a pandas dataframe.  <http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_excel.html#pandas.read_excel>  Must import numpy as np to deal with arrays  Parameters   * io is the path * Strings are used for sheet names. Integers are used in zero-indexed sheet positions. E.g. [0,1,”Sheet5”] loads first, second and sheet named “Sheet5” * Names   Parse specified sheets into dataframe. Equivalent to read\_excel  Class for writing dataframe objects into excel sheets, default is to use xlwt for xls,openpyxl for xlsx   * Date\_format for date objects written into excel files ‘YYYY-MM-DD’ * Mode: {‘w’ or ‘a’} default ‘w’ , file mode to use, write or append   E.g.  Default usage  **with** ExcelWriter(‘path\_to\_file.xlsx’, date\_format='YYYY-MM-DD'  ) **as** writer:  df.to\_excel(writer)  To write to separate sheets in a single line:  **with** ExcelWriter('path\_to\_file.xlsx') **as** writer:  df1.to\_excel(writer, sheet\_  name='Sheet1')  df2.to\_excel(writer, sheet\_name='Sheet2') |
| Other Input/output commands found at  <http://pandas.pydata.org/pandas-docs/stable/reference/io.html> | |
| dataframe = pd.read\_csv(‘path’,parameters) | Saves the data into data frame |
| dataframe.head()  dataframe.tail() | Gets the first 5 rows  Gets last 5 rows  Can specify number of rows in the () |
| dataframe[‘header’]  dataframe.header | Gives you a series of the data under the specific header  Similar but cannot be used if header has similar name as a built-in function |
| dataframe[‘new\_header’] = dataframe.header1 + ‘,’ + dataframe.header2 | Combines two series separated by a comma |
| dataframe.describe()  dataframe.describe(include=[‘object’]) | If you have numeric data, it will give you the statistics  If not, it will just describe count, frequency and other properties  In argument you can have include = [‘Header’] /[‘type’] |
| dataframe.dtypes | Gives you the type of data in each column |
| dataframe.size | Gives you the size of the data (row, column) |
| dataframe.rename(columns = {‘header’ : ‘new\_header’}, inplace = True) | Renaming columns |
| dataframe.columns  OR  dataframe\_cols = [list]  dataframe.columns = dataframe\_cols | Gives you names of the columns  Replaces previous columns, useful if you change a lot of column names |
| dataframe.columns = dataframe.columns.str.replace(‘ ’,’\_’) | Replaces column heading spaces with \_ |
| dataframe.drop('Header1', axis = 1, inplace = True).header()  OR  Dataframe.drop([‘header1’,’header2’], axis =1, inplace = True) | Removing first column from a pandas dataframe, axis tells us that we want to drop column . header is for visualization  Axis = 0 is the row axis ( here we use the index)  Deletes multiple columns |
| dataframe.header1.sort\_values()  OR  dataframe[‘header1’].sort\_values() | Sorts a series into alphabetical order or increase in numbers  If you have a combination of integers and words, first it sorts numbers then words in alphabetical order  Does not change original order of data, you are shown the sorted data, but this does not change the original data source |
| dataframe.sort\_values(‘header’) | Sorts a dataframe not a series. Can still see the entire dataframe with ‘header’ sorted |
| is\_long = dataframe.header1>= 200  OR  dataframe[(dataframe.header1 >=200) | (dataframe.header2 == ‘string’)]  etc. | Sorts data by choosing dataframe >= 200s e.g. movie duration  Sorts by choosing data in header1 >= 200 or header2==’string’  Etc. |
| dataframe.mean() OR  OR dataframe.mean(axis=0)  OR dataframe.mean(axis=index)  OR dataframe.header.mean()  dataframe.mean(1) OR  dataframe.mean(axis=columns) | Column means  Row means |
| Dataframe.header.str.contains(‘string’) | To check if column contains a specific word, may want to filter the dataframe |
| dataframe[‘header']= dataframe.header.astype(float) | Overwrites the column under header with a float type  Useful when doing mathematical operations on int data |
| groupby | E.g.  drinks.groupby(‘continent’).beer\_servings.mean()  This will analyze the mean of beer servings by continents  If beer servings is not specified then it will output the mean of all columns |
| agg | To output many arithmetic calculations at once  E.g.  drinks[drinks.continent ==’Europe’].beer\_servings.agg([‘count’, ‘min’,’max’,’mean’])  This will give us the ,count,min,max,mean of beer servings in Europe |
| %matplotlib inline | To use plotting library |
| Plot(kind = ‘ ‘) | E.g.  drinks.groupby('continent').mean().plot(kind='bar')  will output a bar graph of the means of each column grouped by continents  kind = bar,line,hist |
| value\_counts() | Gives us the count of an attribute |
| unique()  nunique() | Unique values in the series  To get the number of unique items in series |
| pd.crosstab() | Can pass it two different series, allows us to get a tally account of the dataframe that belongs to combination of the two series |
| NaN | Means there’s a missing value |
| isnull()  notnull() | Gives TRUE where there is data missing  Gives TRUE where there is data |
| dataframe.isnull().sum() | To sum up the number of missing values in each column |
| dataframe.dropna(how=’any’) | Drop a row if any of its values are missing  Can add another argument subset=[‘header1’,’hearder2’]  Which drops a row if either content in header1 or header2 are missing  Can use dataframe.dropna(how=’any’).size() to see the shape |
| dataframe.dropna(how =’any’) | Only drop a row if all its values are missing |
| fillna  dataframe[‘Header1’].fillna(value = ‘Various’.inplace=True) | Replaces the empty slots with the word various |
| Dataframe.loc[rows,columns]  dataframe.loc[rows,] | Gives you output multiple row/column selection  Loc is inclusive on both sides  All columns  Eg, rows = [0,1,2] or 0:2  Columns = ‘headings’ |

Online data to use as practice can be found at:

<https://bit.ly/chipordes>

<https://bit.ly/imbratings>

<https://bit.ly/uforeports>

<https://bit.ly/drinksbycountry>

1. **Numpy**

For managing multi- dimensional arrays

* <https://www.numpy.org>
* Import numpy as np
* For machine learning data has to be in arrays

Important functions

1. **Matplotlib**

This library is for data visualization

* <https://matplotlib.org/tutorials/index.html#introductory>
* Import matplotlib.pyplot as plt
* Import seaborn as sns : python library for statistical data visualization built on top of Matplotlib

Examples of functions:

* plt.plot(x,y,label = ‘linear/cubic/quadratic’)
* plt.xlabel(‘..’)
* plt.title(‘,,’)
* plt.legend()
* plt.show()

1. **Scikit-learn**

This is for machine learning algorithms based on python.

**B. Important modules**

Also go through these modules

1. **Random** : to generate random numbers

* import random
* random.random() : random float between 0-1
* random.randint(start,stop) : random integer between start-stop
* random.randrange(start,stop,step): randomly selected element in this range
* random.seed(): initialize the random number generator
* random.choice(seq) : return element from a sequence
* random.sample(population,k) : return a *k* length list of unique elements chosen from the population sequence or set.
* random.gauss(mu, sigma) : Gaussian distribution. mu is the mean, and sigma is the standard deviation.
* random.lognormvariate(mu, sigma): Log normal distribution
* random.normalvariate(mu,sigma) : normal distribution
* Gaussian distribution. mu is the mean, and sigma is the standard deviation.

1. **Statistics**

* Import statistics
* Create a sample list, eg a = [2,3,4,55,2,1,]
* Statistics.mean(a)
* Median(), median\_low(), median\_high()
* Mode(), pstdev(), pvariance(), stdev(), variance()

1. **Math** : pi,sqrt,log etc
2. **Datetime**

* Import datetime
* Datetime.datetime.now() : date as a list (YYYY,M,DD,MM,SS)
* Datetime.datetime.now().strftime(“%F”) : YYYY-MM-DD

1. **Csv** : comma separated values

**SciKit -Learn**

Scikit-learn is used to provide simple and efficient tools for data mining and analysis.

**Recommended Book**: An Introduction to Statistical Learning with Applications in R, Springer

**Machine learning repository**: 474 data sets for practice <http://archive.ics.uci.edu/ml/datasets.php>

**Uses**

* Classification – Identification to which category an object belongs.
* Regression – Predicting a continuous- valued attribute associated with an object.
* Clustering - Automatic grouping of similar objects into sets.
* Dimensionality reduction – Reducing the number of random variables to consider
* Model Selection – Comparing, validating and choosing parameters and models

**SCIKIT-LEARN**

1. Supervised Learning

Definition : Making predictions using data

1. First, train a machine learning model using labeled data( input , outcome)
2. Then make predictions on new data for which the label is unknown

Terminology

* Each **row** is an observation/sample/example/instance/record
* Each **column** is a feature/predictor/attribute/independent variable
* Each value we are predicting is the response/ target/outcome/dependent variable
* **Classification is supervised learning** in which the response is categorical e.g. predicting a species or if email is spam or ham
* **Regression is** **supervised learning** in which the response is ordered and continuous e.g. price of a house, height of a person

Requirements

We must make sure that the features and response are in the form that sci-kit learn expects

1. **Features and response** should be passed into the machine learning model as **separate objects**

e.g iris.data

iris.target

1. Regardless of if the problem is numeric or classification, the **response object should always be numeric**
2. Features and response should be **numpy arrays**
3. Features (2D,row,coulumn) and response (single dimension, same as row/observation) should have **specific shapes**

Then;

Store the independent variable in **X** (capitalized as it’s a matrix) and the dependent variable in ***y*** (vector)

Prediction;unknown input has similar data as known input, for visualization we can use KNN(below) or matplotlib

Models may not give the same answer so do evaluate which is best ;

**Machine Learn Model Evaluation Procedure**

**Evaluation Procedure 1**: Train and test on the entire dataset, test the model on the same dataset and evaluate how well we did by comparing the predicted response values with the true response values

e.g

* from sklearn.datasets import load\_iris
* iris = load\_iris()
* X = iris.data
* y = iris.target

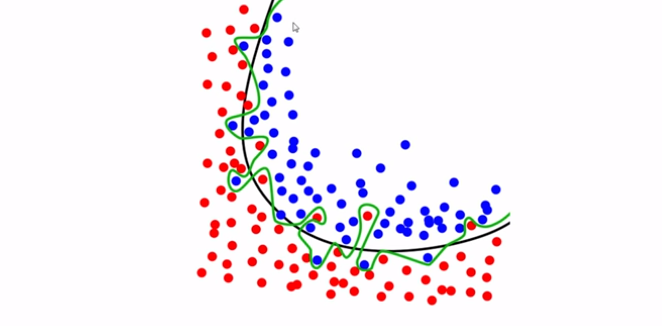
Then test models, store predicted response values and make a classification accuracy using metrics, i.e.

* from sklearn import metrics
* print metrics. accuracy\_score(y,y\_pred)

But training and testing on the same data creates overly complex models that overfit the training data , models overfit

E.g KNN K=1

Black line has learned the signal and green has learned the noise



**Evaluation Procedure 2:** Train/test split

1. We first split the dataset into two pieces; a training set and a testing set
2. Then train the model on the training set
3. And test the model on the testing set, and evaluate how well we did

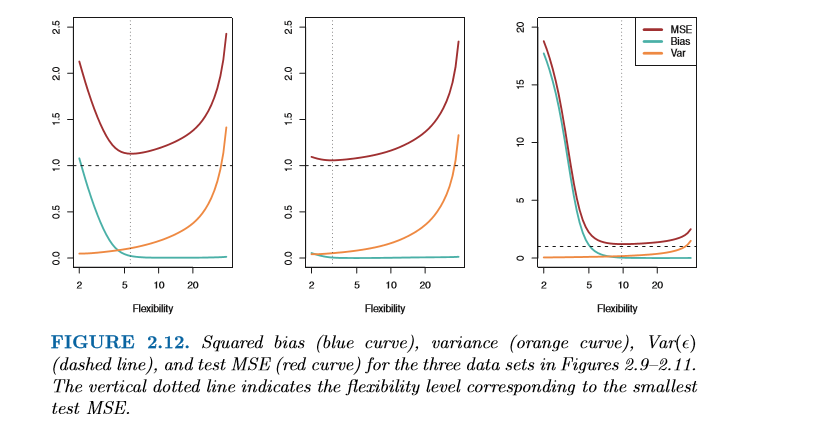
* from sklearn.cross\_validation import train\_test\_split
* X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.4)

The Bias- Variance Trade-Off

Variance is the amount by which the predicted value would change if we estimated it using a different training data set. Since the training data are used to fit the statistical learning method, different training data sets will result in a different prediction. But ideally the estimate should not vary much between training sets. Large variance indicates large changes in the predicted values. More flexible statistical methods have a higher variance because it follows data closely.

Bias refers to the error that’s introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model. For example, linear regression assumes there’s a linear relationship between Y and X variables. It is unlikely that any real-life problem truly has such a simple linear relationship, and so performing linear regression will undoubtedly result in some bias in the estimate of f.

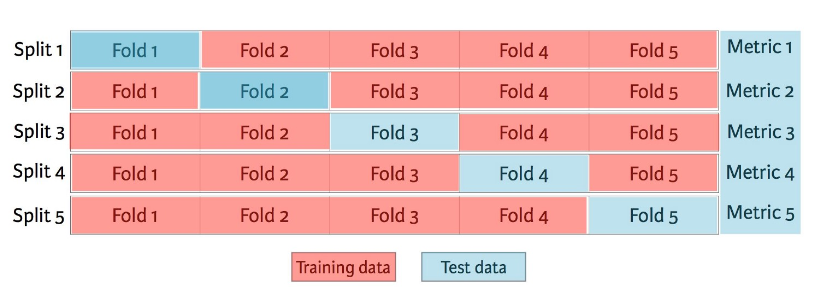
Generally, more ﬂexible methods result in less bias. As a general rule, as we use more ﬂexible methods, the variance will increase, and the bias will decrease. The relative rate of change of these two quantities determines whether the test MSE increases or decreases. As we increase the ﬂexibility of a class of methods, the bias tends to initially decrease faster than the variance increases. Consequently, the expected test MSE declines. However, at some point increasing ﬂexibility has little impact on the bias but starts to signiﬁcantly increase the variance. When this happens the test MSE increases



**K-Fold Cross Validation**

Cross-validation is when the dataset is randomly split up into ‘k’ groups. One of the groups is used as the test set and the rest are used as the training set. The model is trained on the training set and scored on the test set. Then the process is repeated until each unique group as been used as the test set.

For example, for 5-fold cross validation, the dataset would be split into 5 groups, and the model would be trained and tested 5 separate times so each group would get a chance to be the test set



The machine learning pipeline has the following steps:

1. Preparing data
2. Creating training/testing sets
3. Instantiating the classifier/regression model
4. Training the classifier/regression model
5. Making predictions
6. Evaluating performance
7. Tweaking parameters.