General Assembly notes

Important links:

Pandas: <http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.plot.html>

GitBash: <http://ss64.com/bash/>

Lecture #1

09.20.16

0. Make sure you can do the python exercises. Make sure you can do the basic and mid level python exercises fairly quickly

I. data science intro

examples of data science projects:

-facebook facial recognition

- netflix/amazon.spotify recommendations

- neural style

- etc

Data scientist attributes to help with success

-

II. Data science workflow

III. Course project

Lecture # 2

09.22.16

Not autosaved by atom -\_-

Think SMART

Lecture #3

09.27.16

Notes for myself, how do you change the directory that a

1. Pandas core command

df.xyz 🡪 this type of command structure is an attribute #example df.index

df.abc() 🡪 this type of command structure is a method #example df.tail()

1. Statistic refresher
   1. Mean, median, and mode
2. Basic stats
3. Correlation
4. Visualization and correlation w/ pandas codealong

Lecture #4 – Command Line and Version Control (gitbash and github)

Command line:

Order is command –options argument

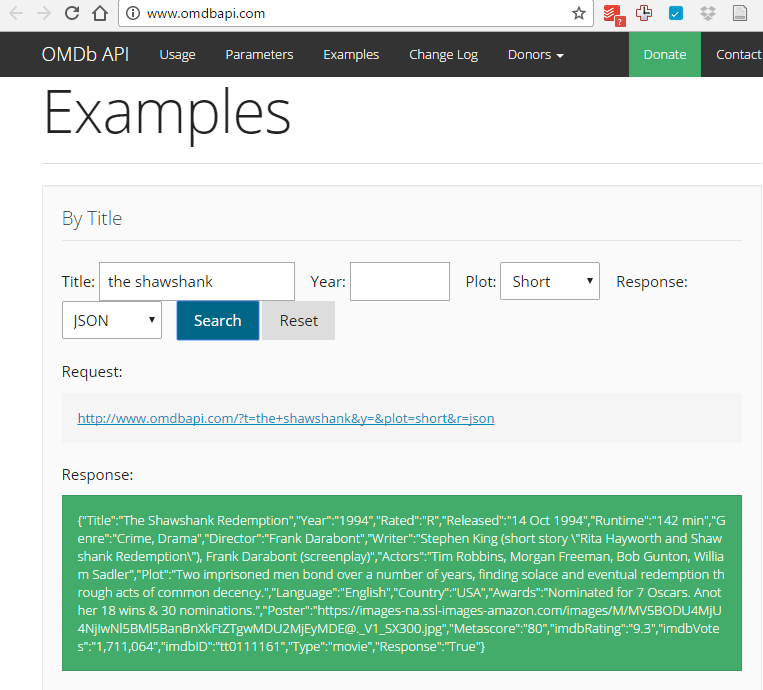
Ex: ls –l (-l means longer list)

* Git & Github
  + If you make a directory in Github with README.md (markdown language) Github will display that page.
  + SHARING - an image or file from Github you must hit ‘download’ or ‘raw’ makes sure the URL includes the word ‘raw’.
  + When creating a repository – Click the ‘initialize this repository with a README’ so you have a directory?? – not sure that’s why but I think that’s close
  + GitHub workflow diagram
    - Clone 🡪 clones github repository on your machine
    - ‘Push’ from computer to github and ‘pull’ github to computer syncs the repository
    - Push or Pull you must be in the .git directory?
      * Red – means local copy is different only
      * Green – means local copy is staged to push to github
  + Some common commands
    - Mkdir – make a new directory
    - Touch – make a new file
    - Pwd – prints working directory
    - Git status – shows what files to push/pull
    - Cd – change directory
    - Ls or dir – lists the directory ‘-l’ long ‘-a’ will show everything
    - Push commands:
      * git add –A 🡪stages files
      * git commit –m “note attached to commit” 🡪 commits file to local repository (NOTE: if you miss the –m you end up in VI….you’ll have to look up how to get out)
      * git push origin master 🡪 pushes file to github
* Markdown Language
  + Easy-to-read, east-to-write markup language
  + Usually rendered as html (always?)
  + Common commands
    - ## Header size 2
    - \*italics\* and \*\*bold\*\*
    - [link to GitHub](<https://github.com>)
    - \* bullet
    - ‘inline code’ and ‘’’code blocks’’’
    - Valid HTML can also be used in markdown
      * Ex: <br></br> to add a line break
* Things you should be able to do at the end of this class
  + Clone a github repository to your laptop
  + Sync your local files to your github repository
  + Use more advanced command line
* Intermediate command line

Lecture #5 – Fetching Data

10.04.16

* Fetching data
  + Json – format is in key value pairs (ex {“firstName” : “Name”}
  + API – allows people to interact with the structures of an application to get, put, delete or update data
    - RESTful APIs include a base URL and collection
    - A media type (JSON)
    - Operations (GET, PUT, POST, DELETE) using http requests
    - API FORMAT: GET <https://api.instagram.com/v1/users/search/?q=andy>
      * GET = Operation command = GET
      * Collection = ‘users’ in this example
      * Query string = ‘?q=andy’ in this example
      * List of api wrappers are posted on github: <https://github.com/realpython/list-of-python-api-wrappers>
    - Web Scraping – make sure you don’t pull the data too quickly (once per second is pretty safe)
    - API for pubmed : <http://bcb.io/2010/01/02/automated-retrieval-of-expression-data-with-python-and-r/>
    - OMDb example:



* + - Requests: HTTP for Humans: <http://docs.python-requests.org/en/master/>
    - In a dictionary a key is unique
* Key concepts

Lecture #6 – Intro to regression

10.06.16

* Linear Regression NOTEBOOK: “06\_linear\_regression”
  + Create LR model
    - Run .head AND .dtypes before analyzing data so you know what you’re looking at.
    - Do visualizations so you know the likely best fir model
    - Kaggle model: <https://www.kaggle.com/c/bike-sharing-demand/data>
    - Example of making a column an index at import: bikes = pd.read\_csv(url, index\_col='datetime', parse\_dates=True)
    - For regression, x and y len must be equal (aka complete data set)

STEPS for creating a model:

1. Create X and y

# create X and y (x is capitalized because it is the matrix or the column factors)

feature\_cols = ['temp']

X = bikes[feature\_cols]

y = bikes.total

X.shape

1. Choose & import our estimator – using the scikit learn library ‘estimator’ is specific
   * + - * Import linearRegression as the ‘estimator’

# import, instantiate, fit

from sklearn.linear\_model import LinearRegression

1. Assign estimator into a variable

linreg = LinearRegression()

1. Fit the model with the training data:

linreg.fit(X, y)

1. Calculated error to determine ‘goodness’ of the model
2. Outputted results/predictions (two ways)

linreg.intercept\_ + linreg.coef\_\*21

array([ 198.62756307])

linreg.predict(21)

array([ 198.62756307])

1. MAKE SURE THAT YOU ARE COMPARING TO THE ACTUAL MEAN, this should be true
   1. Null predictions; category – mode; response – mean
   * Working with multiple features
   * Choosing between models
   * Creating features
   * Comparing LR to other models

Lecture # 7 – K – Nearest Neighbors

10.11.16

Knn & SciKit learn

* Supervised – you’re teaching an algorithm to determine the difference between certain categories. Example teaching an algorithm to determine what coin is being inserted into a vending machine based on size and weight.
* Unsupervised – user data and you’re trying to categorize people after they use a website or product and creating a model based on the type of person because the prediction breaks people into segments

NBA data

Lecture # 8 – Basic Model Evaluation

10.13.16

* Discuss the bias/variance reading
* Exploring the bias – variance tradeoff
* Model evaluation using train/test split
* Reproducibility

Lecture # 9 – Logistic Regression

10.18.16

NOTE: to see seaborn documentation look at github NOT STANFORD

* Logistic regression
  + Benefits using the logistic regression
    - Predicted probability of the model being one of two variables
    - The probability equation uses an exponential of the linear equation which forces the prbabilites to have a limit at 0 and 1 in an s-shaped curve.
  + Logistic regression can be used for categories that have more than 2 different values (continuous), you have to break the variables into subsets.

Advantages of logistic regression:

* Highly interpretable (if you remember how)
* Model training and prediction are fast
* No tuning is required (excluding regularization)
* Features don't need scaling
* Can perform well with a small number of observations
* Outputs well-calibrated predicted probabilities

Disadvantages of logistic regression:

* Presumes a linear relationship between the features and the log-odds of the response
* Works only on datasets with binary responses unless extended by breaking larger numbers of responses into multiple binary problems or Multinomial logistic regression is used
* Performance is (generally) not competitive with the best supervised learning methods
* Can't automatically learn feature interactions
* Confusion matrix

Lecture # 10 – Advanced Model Evaluation

10.20.16

* Data Preparation
* Visualizing the pf of a binary classifier
* ROC and AUC

Lecture # 11 – Standardization (z-value scaling) and clustering

10.25.16

* Standardize feature values
  + KNN is susceptible to scaling so standardizing the KNN is effected by scale
* Cluster using K-means
  + How to find the number of clusters (K)
    - You can look at goodness of fit for different k values
    - Use a method : farthest point, k-means++ (probably the best default)
    - For this case we don’t have a y value because this is unsupervised learning.
* Compare ‘how good’ the clustering models are
  + As the number of clusters gets closer to the number of observations the silhouette coeffeicient will continue to rise. PICK a k-mean that corresponds to a peak.

Lecture #12 - Lectures

10.27.16

* Opening
* Pipeline

Lecture # 13 Natural Language Processing

11.01.16

* Natural language processing
  + Test train split is an efficient way to evaluate a model. The correct way to evaluate a model is with cross-validation .
* h
  + a
    - a

Lecture # 13 Natural Language Processing and dimensionality reduction

11.03.16

* Natural language processing
  + Tokenize: Pulls out all the unique words for the text
  + countvectorizer
  + Ngrams: 1, 2 - can count 1 word and 2 word Test train split is an efficient way to evaluate a model. The correct way to evaluate a model is with cross-validation.
  + Limit the number of features (took the top words)
  + Stop word removal
  + Select a case (lower or upper)
  + Finding the root word (lemmatization and stemming)
  + Sentiment analysis
* Tf-IDF uses more of a ration instead of straight word count
* Textblob
  + Quick tools to pull out whole sentences
  + **Sentiment analysis** was part of textblob
  + Spelling correction
  + Give it text and it will tell you what language it is

Lecture # 15 Decision Trees

11.08.16

* Decision trees
  + In the car example, the null model would be the average cost of all of the cars (6571)
  + “Gini feature” is how the model determines the decision tree

Lecture #16 Random Forest

11.10.16

* Random forest is a poplar method for producing and accurate model
* At the end of running a random forest (even with using an out of bag error) you will need to cross validate your model.
* Decision trees
  + In the car example, the null model would be the average cost of all of the cars (6571)
  + “Gini feature” is how the model determines the decision tree
  + Performance is competitive with the best supervised learning methods
  + Provides a more reliable estimate of feature importance
  + Allows you to estimate out-of-sample error without using train/test split or cross-validation

Lecture # 17 Time series data I

* Random forest review
  + In the car example, the null model would be the average cost of all of the cars (6571)

“Gini feature

Lecture # 18 Time series data II

* Time series modeling
  + For time series modeling, the training set must be chronologically before the testing set.
  + Models may fail for non-stationary data sets
    - Stationary time series, may have a cyclical element but stays flat
    - Random walk – no cyclical trend or overall trend
    - Non-stationary - can have a cyclical period by overall trends up or down
  + De-trending (the average stays the same over time) can remove the increase/decrease trend will allow us to predict future sale.
* Autocorrelated model
  + AR(1) – the autoregressive model and how it auto correlates one time period back (one week, one month, one second, etc)
  + AR(2) – looks at 1 and 2 back
* Moving average models
  + This model is useful for handling specific of abrupt changes in the system
  + AR models slowly incorporate changes in the system by combining previous values; MA models use prior errors to quickly incorporate changes.

Lecture #19 Bonus stuff

* Multi user git repository
* Command line revisited
* Additional methods