July 2nd, 2021

**Data Science Fundamentals with Python & SQL:**

Week 1: Introduction:

* Visual programming available in many tools and helps non-programmers enter field of data science
* Open source and commercial tools are options for data scientist have various strengths and weaknesses
* Cloud Computing can be used to help speed up data scientist work

Languages of Data Science:

* Python is an open source and R is a free software
* Open source and free software ref er to same set of licenses, support collaboration (in many cases but not all)
* Open Source Initiative (OSI) champions open source while Free Software Foundation defines free software
* Open source is more business focused while Free Software is focused on a set of values
* R has array-oriented syntax making it easier to translate from math to code, good for statistical knowledge, can conduct complex exploratory data analysis
* SQL (Structure Query Language), limited to querying and managing data, simple and powerful
  + Useful in handling structured data
  + Relational Database: formed by collection of two-dimension tables (excel spreadsheet)
  + Access data directly, without copying beforehand
  + Interpreter between you and the database, if you learn SQL and learn it with one database you can apply to other databases
  + SQL databases: MySQL, PostgreSQL
* C++ is an extension of C, C with classes, TensorFlow is built with C++

Categories of Data Science Tools:

* Data Integration and Transformation: is referred to as ETL, extract, transform, load, process of retrieving data from remote data management systems
* Data visualization part of initial process and part of final deliverable
* Model Building: process of creating machine learning or deep learning model using an appropriate algorithm with a lot of data
* Model Deployment make machine learning or deep learning model available to third-party applications
* Model monitoring and assessment ensures continuous performance and quality check
* Code asset management allows versioning and collaborative features to facilitate teamwork
* Data asset management brings same versioning and collaborative components to data, replication, backup and access right management
* Development environment (Integrated development environments), tools that allow scientist to implement, execute, test and deploy work
* Execution environment are tools where data preprocessing, model training and deployment take place

Open Source Tools for Data Science:

* Open Source Data Management tools are relational databases: MySQL, PostgreSQL
* Data Management Tools: NoSQL databases such as MongoDB Apache CouchDB and Apache Cassandra
* Data Management Tools: File based tools such as Hadoop File System or Cloud File Systems like Ceph
* Data Management Tools: Elastic search used for storing text data and creating a search index for fast document retrieval
* Data Integration and Transformation Tools: Apache Air Flow, KubeFlow, Apache Kafka, Apache Nifi, Apache SparkSQL, NodeRED. HUE which can create visualizations from SQL queries
* Apache superset is a data exploration and visualization web application
* To turn machine learning model consumable by other developers must turn it into an API (application programming interface)
* Model Deployment: you can use ApachePredictionIO (which only supports Apache Spark ML), Seldon (supports nearly every framework), MLeap, TensorFlow (can serve any of its models using TensorFlow Service)
* Model Monitoring Tools: ModelDB machine model metadatabase where information about models are stored and can be queried, supports Apache Spark ML Pipelines, Prometheus
* IBM AI Fairness 360 can help to model bias against protected groups like gender or race in machine learning models
* IBM Adversarial Robustness 360 Toolbox can be used to detect vulnerability to adversarial attacks and help make the model more robust from manipulated data
* IBM AI Explainability 350 Toolkit makes machine learning process more understandable by finding similar examples within a dataset that can be presented to a user for manual comparison
* Code Asset Management (version control), GIT is the standard, multiple services support Git, most promiment beng GitHub, which provides hosting for software development version management, GitLab (fully open source that you can host and manage yourself) or Bitbucket
* Data asset management, data governance or data lineage, crucial part of enterprise grade data science, versioned and annotated with metadata (data that provides information about other data)
  + Apache Atlas, ODPi Egeria, Kylo are tools that supports this task

Open Source Tools for Data Science Part 2:

* Jupyter is one of the most popular development environments that data scientists are using, supports hundred different programming languages through kernels, different interactive interpreters for different programming languages
* Able to unify documentation, code, output from code, shell commands and visualizations into single document, next generation of Jupyter Notebooks is JupyterLab which is able to open different types of files
* RStudio, exclusively runs R and is tightly integrated to provide optimal experience, unifies programing, execution, debugging, remote data access, data exploration and visualization into one tool, Spyder tries to mimic RStudio to Python World
* Cluster Execution environments come in when your data doesn’t fit into a single computer’s storage or main memory capacity
  + Cluster computing framework Apache spark which is known of linear scalability, doubling the number of servers in a cluster you’ll also roughly double its performance
  + Apache Flink was created as well but the difference is Apache Spark is a batch data processing engine capable of processing huge amounts of data file by file whereas Apache Flink is a stream processing image, main focus on processing real time data streams
  + Ray is also big in the data science execution environments, clear focus on large scale deep learning model training
  + Fully integrated and visual open-source tools are KNIME, Orange,

Commercial Tools for Data Science:

* Data Management Tools: Oracle Database, Microsoft SQL Server, or IBM Db2
* Data Integration and Transformation Tools: Informatica Powercenter and IBM InfoSphere DataStage are the leaders, followed by products from SAP, Oracle, SAS, Talend, and Microsoft. These tools support design and deployment of ETL data-processing pipelines through a graphical interface
  + One tool from Watson Studio Desktop includes component called Data Refinery enabling defining and execution of data integration processes in a spreadsheet style
* To create Machine Learning Models with commercial tools, use data mining product including SPSS Modeler (one version offered in Watson Studio Desktop) and SAS Enterprise Miner
* Model deployment: usually tightly integrated in model building process
* Model monitoring is a new discipline and no relevant commercial tools available, open source is first choice
* Code Asset Management also uses open-source tools, Git via Github
* Data Asset Management, is provided by vendors such as Enterprise Data Governance and IBM creating functions such as data dictionary which facilitates discovery of data assets
* Watson Studio and H2O Driverless AI fully integrated development environment for data scientists usually consumed through the cloud combining Jupyter Notebook with graphical tools to maximize data scientists performance

Cloud Based Tools for Data Science:

* Cloud products are a newer species, follow the trend of having multiple tasks integrated in tools.
* Tools that cover complete development life cycle for all data science include: Watson Studio together with Watson OpenScale, Microsoft Azure Machine Learning, H2O Driverless (operations and maintenance not done by cloud provider so it is not platform or software as a service)
* Data Management: Amazon Web Services DynamoDB, Cloudant
* Data integration Tools: Cloud Data integration and IBM Data Refinery (part of IBM Watson Studio)
* Data Visualiztion: DataMeer, IBM Cogns Business intelligence Suite, IBM Data Refinery can also be used
* Model Building: Watson Machine Learning, Google AI Platform Training
* Model deployment usually tightly integrated to model building process

Libraries for Data Science:

* Libraries are collection of functions and methods that enable you to perform a wide variety of actions without writing the code
  + Libraries built in modules providing different functionalities that you can use directly are also called frameworks
* Python Libraries: Pandas (offers data structures and tools for effective data cleaning, manipulation, and analysis, mainly works with data frame (table with columns and rows for easy indexing)
  + NumPy libraries based on arrays allowing you to apply mathematical functions to these arrays, Pandas is built on NumPy
  + You can create graphs, charts and maps with both
  + Matplotlib package is great for data visualization for making graphs and plots
  + Seaborn, based on matplotlib, easy to generate plots like heat maps, time series and violin plots
  + For machine learning: Scikit-learn library contains statistical modelling including regression, classification, clustering and others built on NumPy, SciPy and matplotlib
    - High level machine learning only needing to specify parameter types you would like to use
  + For deep learning: Keras enables you to build models quickly and simply, function using graphics processing units (GPU) but for many deep learning cases a lower level environment is required
    - Tensor flow is a low level framework used in large scale production of deep learning models
    - Pytorch, use for experimentation making it simple for researchers to test ideas
    - Apache Spark general purpose cluster computing framework that enables you to process data using complete clusters
      * Processes data in parallel using multiple computers simultaneously
* Scala: predominantly used in data engineering but sometimes used in data science
  + Spark Vegas, scala library for statistical data visualizations
  + Can work with data files as well as Spark DataFrames
  + Deep learning: BigDL
* R has built in functionality for machine learning and data visualization but there are also several complementary libraries
  + Ggplot2 popular library for data visualization in R

API: Application Program Interfaces:

* An API lets two pieces of software talk to each other
  + API refers to the interface for example when you have data that you would like to be processed using the Pandas library
    - You can use you the pandas API to process the data by communicating with its software components (many of which aren’t even written in Python, written in NumPy)
    - There can be a separate API for different languages
* Rest (Representative, state, transfer) APIs, another popular type of API, enable you to communicate using the internet, taking advantage of storage, greater data access, AI algorithms and many other resources
  + You can code can be referred to as client, web service as a resource, client finds service through endpoint, client sends request to resource and response is sent to client
  + HTTP methods are way of transmitting data through the internet, we tell REST APIs what to do by sending a request
  + HTTP message will usually contain a JSON file (instructions for the operation that we would like the service to perform)
  + Webservice will return a response through an HTTP message where information is usually return using a JSON file

Data Sets – Powering Data Science:

* data set is a structured collection of data that could be text, number or media
  + could be tabular such as CSV
  + Hierarchical or network data structure used to represent relationships between data, organized in tree-like structure, whereas network data might be stored as a graph
  + Could be raw data files such as images or audio
    - MNIST dataset is popular for data science, contains images of handwritten digits, commonly used to train image processing systems
* Sites where you can obtain data include:
  + Datacatalogs.org
  + UN, European Union,
  + Kaggle – popular data science online community, where you can find and contribute data sets that might help you find the ones that have particular value for you
* open data distribution and use may be restricted as defined by licensing terms
* Linux foundation created the Community Data License Agreement (CDLA), CDLA sharing license you permission to use and modify data but if shared or published must be under the same license terms as original data
* CDLA Permissive license grants you to permission use and modify data as well, however you are not required to share changes to data, don’t need to publish it under the specific license

Data Asset Exchange (DAX):

* IBM created the Data Asset eXchange (“DAX”), to provide data sets that are both high quality and have clearly defined license and usage terms, you can obtain data from IBM Research projects
* Model Asset Exchange provides tutorials (notebooks) on how to perform analysis
* Regularly you would download compressed data set archive from cloud storage, explore data set using Jupyter Notebooks, review data set metadata such as format, licensing terms and size

Machine Learning Models:

* Traditional data analysis approaches, such as personal manually inspecting data or specialized computer program that automates human analysis is limited to the amount of data to be analyzed and complexity of problem
* Machine learning uses algorithm -models- to identify patterns in data
  + the process in which a model learns patterns from data is called “model training”
  + when model is presented with new data it will try to make predictions or decision based on patterns it has learned from past data
  + Machine learning model divided into three classes: supervised learning, unsupervised learning and reinforcement learning
  + Supervised learning: human provides input data and correct outputs and model tries to identify relationships and dependences between input data and correct outputs
    - Good for regression and classification problems
  + Unsupervised learning: data is not labelled by human, model must analyze data and try to identify patterns and structure within data based only on characteristics of data itself
    - Clustering and anomaly detection are two examples of this learning style of the machine model
    - Clustering: used to divide each reach of a data set into one of small number of similar groups
    - For example clustering models could be providing purchase recommendations based on past shopping behaviour and content of a shopping basket
    - Anomaly detection identifies outliers in a data set such as fraudulent credit card transactions or suspicious online log in attempts
* Reinforcement learning: based on way human beings and other organisms learns, learns best set of actions to take given current environment in order to get most reward over time
  + Model used to beat human players in go, chess and popular strategy video games
* Deep learning: specialized type of machine learning tries to emulate the way the human brain solves a wide range of problems, use to analyze natural language, both spoken and text, images, audio and video, to forecast time series data and much more
  + Requires large sets of labeled data to train model, computer intensive and usually requires special purpose hardware to achieve acceptable training times
  + Implemented using popular frameworks such as TensorFlow, PyTorch, Keras
  + Deep learning frameworks typically provide Python API, C++, Javascript
  + You can download pre-trained state of the art from repositories known as model “zoos”
* Enabling an application to identify objects in images by training a deep learning model:
  + Collect and prepare data that will be used to train model, in order to train a model to detect object in images, raw training data must be labelled, for example drawing bounding boxes around objects and labelling them
  + Next, build model from scratch or select an existing model that might be well suited for the task from public or private resource
  + Train model on your prepared data, during training, model learns from the labeled data how to identify objects that are depicted in an image
  + once training commenced, analyze training results and repeat process until trained model performance meets your requirement
  + when satisfied, deploy model to make it available to your applications

Model Asset Exchange (MAX):

* Free open source repository for ready-to-use and customizable deep learning microservices
* You can find models for a variety of domains including image, audio, video and natural language analysis

Week 2: Introduction to Jupyter Notebook:

* Browser based application that allows you to create and share documents that contain code, equations, visualizations, narrative text links
* When ode is run, it generates outputs including plots and tables within the notebook file
* Jupyter can be exported as a pdf or HTML file that can be shared with anyone
* Meant for Julia, Python and R but can support many other languages
* Jupyter lab allows you to access multiple notebook files as well as other code and data files, text editors, terminals, compatible with file formats such as CSV, JSON, PDF
* Can be installed via the command line using the pip install function or downloaded locally to your laptop through the Anaconda platform
* Notebook kernel is computational engine that executes code contained in a Notebook file
  + Jupyter kernels for many other languages exist, but languages may need to be installed through command line interface
* Two process model with kernel and client, client is interface offering user ability to send code to a kernel, kernel will execute code and returns result to client for display
  + the client is the browser
  + when the notebook is saved, it is sent from browser to Notebook server which saves the notebook file on a disk as a JSON file with a (.ipynb) extension
  + to convert files into other formats, Jupyter uses a tool called NB convert, which modifies the notebook by a preprocessor, an exporter converts the notebook to the new file format and a post processor will work on the file produced by exporting it

Introduction to R and RStudio:

* R is a statistical programming language, powerful tool for data processing, manipulation, statistical interference, data analysis, machine learning algorithms
* R supports importing data from different sources such as flat files, databases, the web, statistical software like SPDD and STATA
* Rstudio is a popular environment for running R language
* Popular libraries include dplyri (manipulating data), stringer for manipulating strings, ggplot for visualizing data, caret for machine learning

Plotting within RStudio

* Ggplot used for data visualization such as histograms, bar charts, scatterplots, you can add layers and components on a single visualization
  + it can handle complex requests to add layers into plots by tweaking functions and arguments
  + read ggplot2 by using library(ggplot2)
  + GGally extends ggplot2 by adding several functions to reduce complexity of combining geometric objects with transformed data
* Plotly is an R package can be used to create web-based data visualizations that can be displayed or saved as individual HTML Files
* Lattice (high-level) data visualization tool that is used to implement complex, multi-variable data sets
* Leaflets is useful in creating interactive plots
* To install use the install dot packages and package name command (install.packages(“package name”))
* example of how to use the ggplot function, geom\_point() specifies that you want to have a scatter plot

Useful Functions to use in RStudio:

* make note that functions in R Studio are case sensitive
* use the function library (package) - to load in a package that is part of the standard (base) packages such as ggplot2 or Ggally
* data(data\_set\_name) – to load in data set
* View(data\_set\_name) – for another tab to open in the editor interface displaying the data
* unique(data\_set\_name$data\_column) – to see how many difference species there are present in data set
* install.packages(“Package Name”, repos = “https:// ….”, type = “source”) – run this in the console to install a package
* head(data\_set, #) – to display the first # of rows in a data set
* $data\_set\_name to dislay information about the variables in the Help tab (bottom right)
* Ggplot(aes(x=x\_column\_name, y= y\_column\_name), data = data\_set\_name) + geom\_point – scatter plot
* + ggtitle (“title\_name”) – to add title but must add to command above
* +labs(x=”xlabel”, y=”ylabel”) – to above to add x and y labels
* + geom\_boxplot() – for boxplot
* +geom\_histogram(binwidth = #) – to create histogram, don’t need yaxis

Overview of Git/Github:

* Popular for version control of source code files and projects and collaborating with others
* Version control systems allows you to keep track of changes to documents, makes it easy for you to recover older versions of your document if you make a mistake
* Git is free open source software distributed under the GNU General Public license
* GitHub is popular web-hosted interface by using your command line interface
* SSH protocol is method for secure remote login from one computer to another
* Repository contains project folders that are set up for version control
* Fork is a copy of the repository
* Pull request is way to request that someone reviews and approves your changes before they become final
* Working directory contains files and subdirectories on your computer that are associated with a GIT repository
* You only need to create a repository once either locally and then push to GitHub or by cloning an existing repository by using the command “init”

Git Commands:

* Init – clones an existing repository
* Add – moves changes from working directory to staging area
* Status – allows you to see state of your working directory and the stages snapshot of your changes
* Commit – takes your stages snapshot of changes and commits them to the project
* Reset – undoes changes you’ve made to the files In your working directory
* Log – enables you to browse previous changes to a project
* Branch – lets you create an isolated environment within your respository to make changes
* Checkout – lets you see and change existing branches
* Merge – lets you put everything back together again