Tools for Data Science

Notes by José A. Espiño P. [1]

Summer Semester 2023–2024

[1]The content in these notes is sourced from what was covered in the course the document is named after. I claim no autorship over any of the contents herein.

Contents

1 Overview of Data Science Tools

Raw data needs to be processed for it to be useful. Normally, this involves it going through a series of steps:

1. Data Management: This refers to collecting and storing data in a secure and efficient manner. The data could come from a plethora of sources such as social media, sensors, direct input, and more. The most commonly used open–source data management tools are **relational databases** like MySQL and PostgreSQL, and **NoSQL databases** like MongoDB (stores data in a flexible JSON) and Cassandra. Relational databases are those that store data in tables, while NoSQL databases store data in a non–tabular format. There are also file–based tools like the Hadoop File System (HDFS) and Amazon S3.

2. Data Integration and Transformation: This involves Extracting, Transforming, and Loading data (ETL). Extracting refers to concatenating the data from different sources, transforming refers to cleaning and normalizing the data, and loading refers to storing the data in a database. The most widely–used tools to achieve this are Apache AirFlow, KubeFlow (which allows the execution of machine learning pipelines), Apache Kafka, Apache Nifi, Apache SparkQSL (which allows the execution of SQL queries on Spark), and NodeRED (which is resource–efficient and can be used to create IoT applications). Apache is a popular open–source software foundation that provides support for a wide range of software projects.

3. Data Visualization: This is the graphical representation of data and aids in understanding the data properly. Pixie Dust is a library that allows you to visualise data in Python and Jupyter Notebooks. Hue creates visualisations for large datasets in Apache Hadoop. Kibana is a tool that allows you to visualise data stored in Elasticsearch. Tableau is a popular data visualisation tool that allows you to create interactive dashboards. Lastly, Apache Superset is a data visualisation tool that allows you to create visualisations from a wide range of data sources.

4. Model Building: This step involves the creation of a machine learning model that will be trained on the data to make predictions on new, unseen data. IBM in particular aids

in this step through the IBM Watson Machine Learning service. As quoted directly from the course, " Watson Machine Learning, available as a service on the IBM Cloud platform, enables users to scale their training and deployment of machine learning models. It seamlessly supports popular frameworks like TensorFlow, PyTorch, and scikit-learn, and offers APIs for seamless integration with other applications."

5. Model Deployment: This step refers to the integration of a developed model into a production environment. Here, the machine learning model is made available to other pieces of software via APIs. An example of this are the SPSS Collaboration and Deployment Services. Apache PredictionIO, TensorFlow, and Seldon are tools that can be used to deploy machine learning models.

6. Model Monitoring and Assessment: Continuous evaluation of the model is necessary to ensure that it is still accurate, fair, and robust. Some of the most common metrics used to assess the performance of models are the F1 score, the confusion matrix, and the sum of squared errors. Tools such as Fiddler, ModelDB, and Prometheus can be used to this purpose.

In every step of the way, it is important for you to work on

- Data Asset Management (DAM): The proper storage and documentation of data from multiple sources. Good DAM platforms support collaboration, replication, backup, and access right managements.

- Code Asset Management: The usage of version control to ensure all the code is properly documented and stored. GitHub is a popular tool for this.

- Execution Environment: These are the libraries and tools needed to compile, verify, and execute code. When the data is too large for a single computer's storage, it is executed in cluster execution environments. The most popular of these is Apache Spark, which is known for its **linear scalability**, meaning that the time it takes to process data is proportional to the number of nodes in the cluster.

- Development Environment: Also known as IDEs (Integrated Development Environments), they are a workspace and tools to implement, execute, test, and deploy source code. Some common IDEs are IBM Watson Studio and RStudio. Jupyter is the most famous IDE in data science; it supports several different programming languages through *kernels*, encapsulating the execution environment for each language, and it unifies documentation, code, and visualisations in a single document. JupyterLab is an extension of Jupyter that allows you to work with multiple documents, terminals, and notebooks in a single interface. Apache Zepplin is quite similar to Jupyter Notebook, but instead of requiring external libraries for data visualisation, it achieves this through built-in interpreters. R Studio is an IDE for the R programming language. It unifies programming, execution, debugging, remote data access, data exploration, and visualisation into one tool. It is a popular choice for data scientists who work with R. Spyder is an attempt to replicate R Studio's functionality for Python. It is a powerful IDE that supports data visualisation, debugging, and code execution.

2 Languages of Data Science

Choosing what language to use to work on a particular issue is not a trivial task. This mostly depends on the task in particular and what you are comfortable with. In this section, we will cover some popular languages and their advantages/disadvantages.

- Python:
  Python is the most widely used programming language in Data Science. Python has a simple and readable syntax, plenty of documentation, and robust libraries, such as Pandas, NumPy, SciPy, TensorFlow, Keras, PyTorch, and many others. It is also a general–purpose language, meaning that it can be used for a wide range of tasks. It is a high–language language.

- R:
  As opposed to Python, which is open–source, R is free software. This means that it is free to use, modify, and distribute — free software is more focused on a set of principles to adhere to than business. R is particularly favoured by statisticians, mathematicians, and data miners to develop statistical software. R has an array–oriented programming style, which makes it intuitive to translate from math to code. It is also a high–level language. R has the largest repository of statistical knowledge and has plenty of packages that specialise in data visualisation, data manipulation, and statistical analysis. In addition to that, R has the capacity to perform common mathematical operations like matrix multiplication seamlessly. In general, it is a language that has stronger object–oriented programming facilities.

- SQL:
  SQL usually stands from other languages in that it is a non–procedural language, meaning its scope is limited to querying and managing data. It is older than Python and R by about 20 years and still used to this day because of its simplicity and power. SQL is useful in managing structured data, data that incorporates relations among entities and variables. Relational databases are the most common type of structured data; these are formed by collections of two–dimensional tables, where each has a fixed number of columns and a variable number of rows.
  The SQL language is subdivided into several language elements, such as clauses, expressions, predicates, queries, and statements. When performing operations with SQL, data is accessed directly instead of being copied and then used, which speeds up the processing significantly. It basically acts like an interpreter between you and the database. Many SQL databases, such as MySQL and PostgreSQL, are open–source and free to use.

- Java:
  General–purpose object–oriented language that has had huge adoption in the industry. It is designed to be fast and portable, in part thanks to the fact its applications are compiled to bytecode and run on Java Virtual Machine as opposed to the hardware directly.

- Scala:
  General–purpose programming languages with support for functional programming. It is inter–operable with Java because it also runs in the Java Virtual Machine. It is a statically typed language, meaning that the type of a variable is known at compile time. This allows for faster execution and better error checking. In Data Science, the most popular Scala program is Apache Spark, which is a distributed computing system that allows you to process large datasets in parallel.

- Julia:
  Julia is a general–purpose language, but it is particularly well–suited for numerical and scientific computing. It has a rich set of libraries for linear algebra, signal processing, and data visualisation. Julia is also designed to be easy to use, with a simple syntax that is similar to Python and MATLAB. It is also designed to be fast, with a just–in–time compiler that generates efficient machine code. Julia is a relatively new language, but it is gaining popularity in the scientific computing community.

3 Packages, APIs, Data Sets, and Models

Libraries are a collection of functions and methods specialised in a particular domain or for a particular purpose. Scientific computing libraries are commonly referred to as frameworks because they provide a structure for developing software. Some frequently used ones in Python are:

- Pandas: Offers data structures and tools for effective data cleaning, manipulation, and analysis. Its primary instrument is the data frame, a two–dimensional table consisting of columns and rows.

- NumPy: Provides support for large, multi–dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

- Matplotlib: A 2D plotting library that produces high–quality figures in a variety of formats and interactive environments.

- Seaborn: A Python data visualisation library based on Matplotlib that provides a high–level interface for drawing attractive and informative statistical graphics.

- Scikit–learn: A machine learning library that provides simple and efficient tools for data mining and data analysis. It is built on NumPy, SciPy, and Matplotlib.

- Keras: A high–level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It is designed to enable fast experimentation with deep neural networks. It also allows you to use the GPU for faster computation.

- TensorFlow: An open–source machine learning library developed by Google. It is designed to be flexible and scalable, allowing you to build and train deep learning models on large datasets. TensorFlow is widely used in research and industry for tasks such as image recognition, natural language processing, and reinforcement learning.

- Apache Spark: A distributed computing system that allows you to process large datasets in parallel. It is designed to be fast and fault–tolerant, making it ideal for big data applications. Spark provides a high–level API in Python, Java, and Scala, as well as built–in libraries for machine learning, graph processing, and stream processing.

There are also pretty useful libraries for other languages, such as bigDL (machine learning) and ggplot2(data visualisation) for R, and Apache Flink (stream processing) and Apache Mahout (machine learning) for Java.
An **Application Programming Interface** (API) is a set of rules and protocols that allow different software applications to communicate with each other. For example, the OpenAI API allows you to access the OpenAI GPT–3 model through a simple function call, without having to worry about the underlying implementation. APIs are commonly used in web development to allow different services to interact with each other.
Datasets are structured collections of data. Tabular datasets are those in which the data is structured in rows and columns, like a spreadsheet. Hierarchical data structures are generally used to represent relationships between data.

4 Jupyter Notebooks and JupyterLab

Jupyter Notebook is a browser–based application that allows you to create and share documents containing code, equations, visualisations, comments, and more. It is a powerful tool, since it allows you to record and reproduce experiments in an easy–to–share format. On the other hand, JupyterLab is an extension of Jupyter Notebook that allows you to work with multiple notebooks, terminals, text editors, and mutliple other components. The two of them are open source and support a plethora of programming languages, although originally they were devised for JUlia, PYThon, and R (Jupyter).
A kernel is a program responsible for executing the code in the notebook and returning the results. The kernel is specific to each programming language and are also where the memory is stored — upon shutting the kernel, the memory is lost.
The notebook itself is just a representation of your code, metadata, contents, and output stored as a JSON file with a ˙ipynb extension. The Jupyter architecture uses the NB convert tool to conver files to other formats, such as HTML or PDF.
Jupyter Notebooks can be run locally or in the cloud. The most popular cloud service for running Jupyter Notebooks is Google Colab, which is a free service that allows you to run Jupyter Notebooks in the cloud. Google Colab provides free access to GPUs and TPUs, which can be used to accelerate the training of machine learning models.

5 IBM Watson Studio

Watson Studio is an integrated environment for data scientists, developers, and domain experts to collaboratively and easily work with data and use that data to build, train, and deploy models at scale. It is a cloud–based platform that provides tools for data preparation, model

building, and model deployment. Watson Studio supports a wide range of programming languages, including Python, R, and Scala. It also provides a visual interface for building and training machine learning models. Watson Studio is built on top of IBM Cloud, which provides a secure and scalable infrastructure for running machine learning workloads.
Watson Studio provides a wide range of tools for data science, including:

- Data Refinery: A tool for cleaning and transforming data. It provides a visual interface for performing common data preparation tasks, such as removing missing values, normalizing data, and encoding categorical variables.
- AutoAI: A tool for building machine learning models automatically. It uses automated machine learning techniques to build and evaluate a wide range of models, and provides recommendations for the best model to use.
- Modeler Flows: A tool for building and training machine learning models. It provides a visual interface for building complex machine learning pipelines, and supports a wide range of algorithms and techniques.
- Jupyter Notebooks: A tool for building and training machine learning models using Python. It provides an interactive environment for writing and executing code, and supports a wide range of libraries and frameworks.
- Watson Machine Learning: A tool for deploying machine learning models. It provides a secure and scalable infrastructure for running machine learning workloads, and supports a wide range of deployment options, including on–premises and in the cloud.