Process Models, Design Thinking, and Introduction

In this module you should have learned:

- An example student for this course is a data science practitioners that has some foundation and some expertise building ML models
- This course assumes some fundamental knowledge in linear algebra, probability, statistics, and Python programming
- In Python it is expected that you have some experience with NumPy, matplotlib, pandas and scikit-learn
- Watson Studio and Watson Knowledge Catalog make it easy to share code, data and collaborate on data science projects
- OSEMN is an example of a simple process model and CRISP-DM an example of a more complex process model
- The design thinking process can be naturally applied as a data science process model
- Two advantages of design thinking in data science is that it is applied outside of data science and it encourages the inclusion of domain experts and stakeholders
- All process models work to describe the business opportunity before anything else
- All process models encourage feedback loops for iterative improvement

Data Collection

In this module you should have learned:

- Stakeholder or domain expert opinion, feasibility and impact are three of the most important factors when prioritizing business opportunities
- The practice of articulating a business opportunity, with the data in mind, as a testable hypothesis helps keep the overall project linked to the business needs
- The notion of degree of belief is important when making statements both in science and in business. No statement has 100% degree of belief, it is some percentage of 100% that is a reflection of accumulated evidence
- The scientific method helps formalize a process for rationalizing business decisions through experimentation and evidence

Data Ingestion

In this module you should have learned:

The fundamental parts of a data ingestion pipeline are:

1. Gather all relevant data from the sources of provided data
2. Implement several checks for quality assurance
3. Take the initial steps towards automation of the ingestion pipeline

Data engineers and data scientists have the most overlap in job duties when the data is being collected and when models are deployed. This is particularly true when it comes to automation.

Data Analysis

Exploratory Data Analysis

The main goals of EDA are:

1. Provide summary level insight into a data set
2. Uncover underlying patterns and structure in the data
3. Identify outliers, missing data and class balance issues
4. Carry out quality control checks

The principal steps in the process of EDA are:

1. **Summarize the data** - Generally done using dataframes in R or Python
2. **Tell the Story** - Summarize the details of what connects the dataset to the business opportunity
3. **Deal with missing data** - Identify the strategy for dealing with missing data
4. **Investigate** - Using data visualization and hypothesis testing delve into the relationship between the dataset and the business opportunity
5. **Communicate** - Communicate the findings from the above steps

Data visualization

- Jupyter notebooks in combination with pandas and simple plots are the basis for modern EDA when using Python as a principal language

Advantages of Jupyter notebooks:

- They are portable: then can be used locally on private servers, public cloud, and as part of IBM Watson Studio
- They work with dozens of languages
- They mix markdown with executable code in a way that works naturally with storytelling and investigation
- matplotlib itself and its numerous derivative works like seaborn are the core of the Python data visualization landscape
- pandas and specifically the dataframe class works naturally with Jupyter, matplotlib and downstream modeling frameworks like sklearn

EDA and Data Visualization best practices

1. The majority of code for any data science project should be contained within text files. This is a software engineering best practice that ensures re-usability, allows for unit testing and works naturally with version control. >In Python the text files can be executable scripts, modules, a full Python package or some combination of these.

2. Keep a record of plots and visualization code that you create. It is difficult to remember all of the details of how visualizations were created. Extracting the visualization code to a specific place will ensure that similar plots for future projects will be quick to create.
3. Use you plots as a quality assurance tool. Given what you know about the data it can be useful to make an educated guess before you execute the cell or run the script. This habit is surprisingly useful for quality assurance of both data and code.

Missing values

- Dealing with missing data sits at the intersection of EDA and data ingestion in the AI enterprise workflow
- Ignoring missing data may have unintended consequences in terms of model performance that may not be easy to detect
- Removing either complete rows or columns in a feature matrix that contain missing values is called **complete case analysis**
- Complete case analysis, although commonly used, can lead to undesirable results—the extent to which depends on the category of missingness

The categories of missingness are:

- **Missing completely at random or MCAR**
- **Missing at random or MAR**
- **Missing not at random or MNAR**

- The best case scenario is that the data are MCAR. It should be noted that imputing values under the other two types of missingness can result in an increase in bias.
- In statistics the process of replacing missing data with substituted values is known as **imputation**.
- It is a common practice to perform multiple imputations.
- The practice of imputing missing values introduces uncertainty into the results of a data science project.
- One way to deal with that additional uncertainty is to try a range of different values for imputation and measure how the results vary between each set of imputations. This technique is known as **multiple imputation**.

CASE STUDY: Data visualization

It can be easy to get lost in the details of the findings when communicating the finding from EDA to business stakeholders. Project planning and milestones are important so remember to talk about what you:

1. have done
2. are doing
3. and plan to do

- Remember that deliverables are generally a presentation or a report and they should use a portable format (e.g. PDF or HTML)
- Deliverables should should be concise and clear. Appendices are useful as supplemental materials to a deliverable and they help keep them free of unnecessary items.

- Visual summaries are a key component of EDA deliverables
- There is no single right way to communicate EDA, but a minimum bar is that the data summaries, key findings, investigative process, conclusions are made clear.

Data Investigation

TUTORIAL: IBM Watson Dashboard

With the analytics dashboard, you can

- build sophisticated visualizations of your analytics results
- communicate the insights that you've discovered in your data on the dashboard
- share the dashboard with others

The visualizations can tell the story of an investigative process or they can be made to summarize and communicate data in a way that is difficult to do with simple plots.

Hypothesis Testing

- Statistical inference and hypothesis testing can be used together to carry out investigations into the data
- When carrying out a hypothesis test, the central question, null hypothesis and alternative hypothesis should be stated **before** the data are collected
- Simulation based hypothesis testing like permutation tests provide a flexible alternative to more classical approaches
- The bootstrap can be used to quantify the uncertainty around a parameter estimate and the two combined can be used as an investigative tool
- Bayesian methods bring to the table a number of way to think differently about hypothesis testing. They generally require more time to implement, but the quantification of uncertainty can be useful when making important business decisions.
- The t-test is a simple way to care out 1 or 2 sample hypothesis tests.
- There are a number of variants on the t-test, but the unequal variances t-test is commonly used.
- The t-test and ANOVA (more than two groups) test whether group means have differences between each other

CASE STUDY: Multiple comparisons

- *p*-values themselves are not a source of ground truth, but they are nonetheless quite useful if used appropriately.
- There are a number of ways hack your way to significant results using *p*-values
- Running more than one hypothesis test, on the same data, results in the multiple comparisons problem.

- Multiple comparisons are an issue because there is an expected false positive rate for running one test, and if we run multiple tests say using different combinations of features this expected rate should be higher.
- The Bonferroni correction is commonly used to mitigate the multiple comparisons problem, but it is generally too conservative for large data sets.
- A number of other methods are available including the Benjamini/Hochberg correction that is based on the **false discovery rate**.
- Permutation experiments are offer an additional method to correct for multiple comparisons that require fewer assumptions.

Data transforms and feature engineering

Getting Started

The data-transformations part of the AI workflow is the first point in the overall process that explicitly encourages iteration. This stage encompasses all possible transformations of the data, such as dimensionality reduction, outlier detection, clustering and other forms of unsupervised learning. Combining these activities into a single process is done mainly because selecting a suitable transformation or tuning a given transformation takes the same form. That form builds on the three interfaces of scikit-learn and the container class pipelines.

Transformer interface

- Used to convert data from one form to another

Estimator interface

- Used to build and fit models

Predictor interface

- Used to making predictions

It is worth noting that these interfaces in combinations with pipelines have had such an impact on the data science workflow that Apache Spark now has similar ML pipelines.

Class imbalance, data bias

Imbalanced classes are common especially in specific application scenarios like fraud detection and and customer retention. The first guideline is to ensure that you do not use accuracy as the metric as the results can be misleading. Accuracy is the number of correct calls divided by all of the calls.

$$\text{accuracy} = \frac{tp+tn}{tp+fp+tn+fn}$$

If our positive class is only a small percentage of the overall data you can see that the model will be optimized for the negative class. The ability of a model to resolve true positives will not be well-represented in the metric because it will be overwhelmed by the influence of the majority class. Metrics based on precision and recall will be more specific to the problem because TN (true negatives) is not part of the numerator.

$$\text{precision} = \frac{tp}{tp+fp}$$

$$\text{precision} = \frac{tp}{tp+fn}$$

The most common approaches to address imbalanced classes are sampling based. Between over-sampling and under-sampling, under-sampling is conceptually simpler. Given a minority class or classes that are noticeably underrepresented, you may randomly drop some of those observations from the training data so that the proportions are more closely matched across classes. A major caveat to under-sampling is that we are not using all of the data.

Over-sampling techniques come in several forms, from random or naive versions to classes of algorithms like the Synthetic Minority Oversampling Technique (SMOTE) [1] and the Adaptive Synthetic (ADASYN) [2] sampling method. There are a number of variants of these over-sampling algorithms that can be compared.

All of the sampling techniques that we discussed are implemented in the imbalanced-learn Python package. This package is convenient because it allows for the implementation of multiple sampling techniques as pipelines. Additionally, it interfaces with TensorFlow and Keras in a convenient way as well, which is important because neural networks are generally sensitive to class imbalance. Support Vector Machines (SVM) are an example of a machine learning algorithm that is less sensitive to imbalanced classes. SVMs can be tuned to accommodate situations with unbalanced class proportions making them a reasonable tool for outlier detection as well.

Dimensionality reduction

Applications of data science that often require dimensionality reduction for visualization or for modeling purposes are: image analysis, text analysis, signal processing, astronomy, and medicine. We discuss three main categories of dimensionality reduction techniques: matrix decomposition, manifold learning, and topic models. The techniques developed for topic models generally fall under one of the first two categories, but the application is natural language processing. The principal reasons for considering dimensionality reduction in your workflow are:

- visualization
- remove multicolinearity
- remove redundant features
- deal with the curse of dimensionality
- identify structure for supervised learning
- high-dimensional data

These materials review principal components analysis (PCA), Non-negative matrix factorization (NMF) and singular value decomposition (SVD) as examples of matrix decomposition algorithms. A major drawback to using PCA is that non-linear or curved surfaces tend to not be well-explained by the approach. An alternative approach uses manifold learning for dimensionality reduction. Specifically, we discuss the tSNE family of approaches.

We present NMF and Latent Dirichlet Allocation (LDA) as example methods to carry out topic modeling. The embedding approach tSNE is often used to visualize the results of topic model representations in lower dimensional space to both tune the model as well as gather insight into the data. The package pyLDAvis is specifically purposed with visualizing the results of these models.

- sklearn.decomposition
- sklearn.manifold

CASE STUDY: topic modeling

Topic modeling is a commonly used form of dimensionality reduction. When we use visualization tools to explore the results of topic modeling we do so to identify features that are relevant to the domain. These insights can be transformed into new features that may be used directly or appended to the feature matrix. We use topic modeling in this case study to enable domain specific feature engineering.

---

[1]: Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[2]: Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. Adasyn: adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, volume, 1322–1328. June 2008. doi:10.1109/IJCNN.2008.4633969.

Pattern recognition and data mining best practices

TUTORIAL: IBM ai360

The AI Fairness 360 toolkit is an extensible open-source library containing techniques developed by the research community to help detect and mitigate bias in machine learning models throughout the AI application lifecycle. The AI Fairness 360 Python package includes metrics to test for biases and algorithms to mitigate bias.

A bias detection and/or mitigation tool needs to be tailored to the particular bias of interest. More specifically, it needs to know the attribute or attributes, called *protected attributes*, that are of interest: *race* is one example of a protected attributei, another is *age*.

The tutorial shows how to employ the toolkit to compute fairness metrics and mitigate the bias using a feature transformation.

Handling outliers

Outlier detection algorithms:

Isolation forest

- One efficient way of performing outlier detection in high-dimensional datasets is to use random forests. This method *isolates* observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

Elliptic envelope

- One common way of performing outlier detection is to assume that data are Gaussian distributed. From this assumption, we can try to define the general shape of the data, which leads to a threshold-approach to calling an observation an outlier.

OneClassSVM

- Use a single class version of a support vector machine to identify an optimal hyperplane that can be used to call an observation an outlier.

For high-dimensional data it is sometimes necessary to pipe the data through a dimension reduction algorithm before applying the outlier detection algorithm. In addition to how you scale the data, there is the choice of dimension reduction algorithm and the choice of outlier detection algorithms. Moreover, there are generally parameters that modify these outlier detection algorithms, such as an assumed level of contamination. Given the number of tunable components, grid-searching and iteratively comparing pipelines to decide on a suitable outlier detection pipeline can be time consuming.

Unsupervised learning

Machine learning models that are used to predict the values of response variables generally fall into the domain of supervised learning models. By contrast, unsupervised learning deals with the discovery of patterns derived from the feature matrix itself. We have already seen how useful derived patterns are in the example of visualizing high-dimensional data and outliers. The domain of unsupervised learning has many families of models. Clustering analysis is one of the main subject areas of unsupervised learning and it is used to partition the observations in a feature matrix into groups. Some models use probabilistic or 'soft' assignments.

We discuss some of the most common clustering algorithms and mixture models.

Clustering algorithms

- *k*-means
- Hierarchical clustering
- Affinity Propagation
- Spectral Clustering

Mixture modeling

- Gaussian Mixture Models
- Dirichlet Process Gaussian Mixture Models

There are a number of other clustering algorithms available

Model selection

Silhouette Coefficient

- PROS: very flexible and can be visualized through a silhouette plot CONS: generally higher for convex clusters than other clusters such as density based clusters

Davies-Bouldin Index

- PROS: The computation of Davies-Bouldin is simpler than that of Silhouette scores. CONS: generally higher for convex clusters than other clusters, such as density based cluster. It is also limited to the distance metrics in Euclidean space.

Calinski-Harabasz Index

- PROS: The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster CONS: generally higher for convex clusters than other clusters such as density based clusters

CASE STUDY: clustering

In this case study we begin to see the utility of iterative comparison across transformation pipelines. The three central questions to keep in mind are:

- Which preprocessing processing pipeline is the best?
- Which transformation pipeline is the best?
- Which model is the best?

The choice of model is the question we will dive deeply into during the next course in this series, but the first two are very relevant to this course. Similar to the first case study one aspect of this exercise is to create new features with unsupervised learning. Here we use all of clusters, but a domain expert or say someone from marketing might want to weight certain profiles differently than others.

Model Evaluation and Performance Metrics

Evaluation metrics

Supervised learning is the focus of this course, but it should be seen in the context of the other learning fields.

- supervised learning
- unsupervised learning
- semi-supervised learning
- reinforcement learning

As a reminder the type of supervised learning depends on the data type of the target. The *supervised learning* problem is referred to as either

Regression (when Y is real-valued)

- e.g., if you are predicting price, demand, or number of subscriptions.

or

Classification (when Y is categorical)

- e.g., if you are prediction fraud or churn.

Regression

The two following metrics are the most commonly used.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

The root mean square error (RMSE) can be calculated in several ways and it is equivalent to the sample standard deviation of the differences. The mean absolute error (MAE) is another commonly used metric in regression problems. A major advantage of RMSE and MAE is that the values are interpreted in the same units as the original data. MAE is the average of the absolute difference between the predicted values and observed value. Unlike RMSE all of the individual scores are weighted equally during the averaging. **The squaring of the term in RMSE results in a higher penalty on larger differences when compared to MAE**.

Classification

Most classification metrics start from a confusion matrix.

	Predicted False ($\hat{Y} = 0$)	**Predicted True** ($\hat{Y} = 1$)
True ($Y = 0$)	True Negatives (TN)	False Positive (FP)
True ($Y = 1$)	False Negatives (FN)	True Positives (TP)

$\text{accuracy} = \frac{tp+tn}{tp+fp+tn+fn}$: overall proportion correct

$\text{precision} = \frac{tp}{tp+fp}$: proportion called true that are correct

$\text{recall} = \frac{tp}{tp+fn}$: proportion of true that are called correctly

The F1_score is the harmonic mean of precision and recall. There are different variants of the F1_score, notably the F_β score.

$$\text{F1\_score} = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}}$$

There are also several ways to average the F1_score when working in multi-class applications (e.g. weighted, micro, macro). The *average* parameter can change significantly the behavior and performance of your model especially when the classes are imbalanced.

Multi-class and multi-label metrics

Multiclass classification

- The retention example from above is an example of *multi-class classification* because there are more than two classes. Multiclass classification works under the assumption that each sample is assigned to only a single label.

Multilabel classification

- Multilabel classification uses a set of possible labels and allows multiple labels to be assigned to each sample. The labels are are not mutually exclusive.

Model Performance

How well a model performs can be decomposed as bias, variance and noise. Given a true model and infinite data we would be able to calibrate a model to reduce both bias and variance. These conditions are not realistic so there is a trade-off between the error that is related to model assumptions (bias) and the error that arises due to fluctuations in the training data (variance)

When a model exhibits high bias

- **Add more features** such as engineered features or those derived from additional data
- **Use a more sophisticated model**
- **Decrease regularization** by tuning the regularization hyperparameter

When a model exhibits high variance

- **Use fewer features** techniques like variance thresholding, ANOVA (SelectKBest), manifold learning,and matrix decomposition can be used.
- **Use a simpler model** because the model may be overly complex for a relatively simple signal in the data
- **Use more training samples**
- **Increase regularization** by tuning the regularization hyperparameter

Cross-validation along with train-test splits are critical tools to help ensure our models are well calibrated for unseen data. Cross-validation is often performed along with a grid-search over different configurations of hyperparameters. There are several variants of grid-searching including a non-exhaustive version that randomly searched. Model evaluation plots like the *learning plot* can help you understand the type of error associated with model performance and they can help you determine if you need more training data.

Linear models

All of these models make use of the following function:

$$\hat{y}(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1, + \ldots + w_p x_p,$$

The target y could be a column vector or a matrix in the multivariate case. There are p features and p coefficients. The intercept is written here as w_0. If $p > 1$ then we are under the category of multiple linear regression a very common variant in the data science application space. The w_i:'s are parameters or *weights*.

Many of the linear models that are commonly used in data science like linear regression, the t-test and ANOVA are examples of the general linear model. If we relax the assumption that residuals can only be normally distributed and we introduce the concept of a link function then we extend into generalized linear models, of which logistic regression is the best example. One extension further from GLMs brings us into the family of models known as generalized linear mixed models (GLMM). GLMMs contain some of the most flexible and useful linear models available with the best example being multilevel models.

With each extension comes the need for more sophisticated model inference methods. For example, GLMMs generally require Bayesian inference methods like MCMC sampling, while simple linear regression can be carried out with ordinary least squares approaches.

Gradient decent can also be used for inference for several methods including support vector machines and logistic regression. It is a powerful and flexible way to carry out inference on linear models and the results can compares favorably to even more sophisticated models.

Linear models also have a number of extensions including kernels and splines that enable non-linear functions. The level of model interpretation that is available with linear models and ease of implementation make them a safe choice for a baseline model. A baseline model is the one that you default to if a more sophisticated model cannot be shown to have superior performance.

TUTORIAL: Watson NLU

The Watson NLU service has the following features:

Categories: Categorize your content using a five-level classification hierarchy.

Concepts: Identify high-level concepts that aren't necessarily directly referenced in the text.

Emotions: Analyze emotion conveyed by specific target phrases or by the document as a whole.

Entities: Find people, places, events, and other types of entities mentioned in your content.

Keywords: Search your content for relevant keywords.

Metadata: For HTML and URL input, get the author of the webpage, the page title, and the publication date.

Relations: Recognize when two entities are related, and identify the type of relation.

Semantic Role: Parse sentences into subject-action-object form, and identify entities and keywords that are subjects or objects of an action.

Sentiment: Analyze the sentiment toward specific target phrases and the sentiment of the document as a whole.

Custom Models: Identify custom entities and relations unique to your domain with Watson Knowledge Studio.

The service can use as input both URLs and text in the form of strings. Change the target_url below to see how the output changes. The service is used in a similar way to the other Watson services, like text classification and speech to text. Also it is good to keep in mind that Python is just one of several SDK APIs than can be used for NLU and other services.

CASE STUDY: connecting evaluation metrics and business metrics

Natural language processing, or NLP, is a subfield of linguistics, computer science, information engineering, and artificial intelligence. In the last ten years there has been an explosion of data, technology and deep-learning techniques that has advanced NLP applications.

Two prominent examples are speech recognition and conversational agents. Both of which are commonly used in business applications. With so many rapidly expanding areas of NLP, the ability to effectively use the available tools has become increasingly important. As you process the text within a corpus into a matrix that can be consumed by machine learning algorithms there are several questions to stop and ask yourself,

because each domain has nuances that require the treatment of text to be customized. For example, a word or n-gram that appears in high frequency in one corpus may not be important, but in another it may hold importance. The phrase “we will” is probably of little importance in most corpora, but if we analyzed a corpus of political speeches it may be relevant.

- Do I mask pronouns?
- Which stop words do I include?
- Which stemmer/lemmatizer is best?
- Which n-grams do I include?
- Do I filter based on frequency min an max?
- TF or TFIDF or word embeddings?

There are non-frequency based representations of text that have proven useful in NLP with the word2vec group of models being among the most widely used. The model takes a corpus of text and produces a word embedding, which is essentially a projection into a vector space that generally has a few hundred dimensions.

Additional resources

- scikit-learn tutorial working with text

Building Machine Learning and Deep Learning Models

Tree-based methods

Tree-based methods represent an important middle ground between interpretable and black-box types of models. For some business questions like those that deal with marketing and sales data understanding ‘how’ a model makes a decision can contribute to a deployed model’s usefulness.

Decision trees are sometimes used by themselves as a classification or regression model. They are also an important base model for random forests and boosting, two commonly used ensemble models. Not long after the introduction of the bootstrap it was applied to to ensembles of decision trees, which is known as bootstrap aggregating or bagging. Although bagging is often discussed in the context of decision trees, it can be used with any base model.

If we consider a only a subset of features as we grow our decision trees during bootstrap aggregation then we arrive at the random forest model. The trees in boosting are correlated and random forests seeks to mitigate this by averaging slightly different, but fully grown trees. It does this through random subsetting of the features as the branches are grown in the tree. This in turn helps reduce the overall model variance.

The decision tree base models in random forests individually have low bias and high variance. When boosting is carried out with ensembles of trees they tent to be high bias and low variance. The individual trees are also grown sequentially focusing on the bias component of error. Adaboost is perhaps the most widely used boosting algorithm, but there are several others like XGBoost that tend to perform well. Boosting is often referred to as gradient boosting, because it uses gradient decent to carry out inference on the model.

Ensemble learning through model averaging, Bayesian model averaging, model stacking, majority vote and other methods are among the best performing methods known in machine learning. Large ensembles with sophisticated component models tend to have practical limitations like time to train and time to predict, but many of these limitations will be overcome as computers become more powerful, making this an important area to keep in touch with.

- scikit-learn ensemble methods

TUTORIAL: Watson visual recognition

Some of the main features offered by the Watson visual recognition service are:

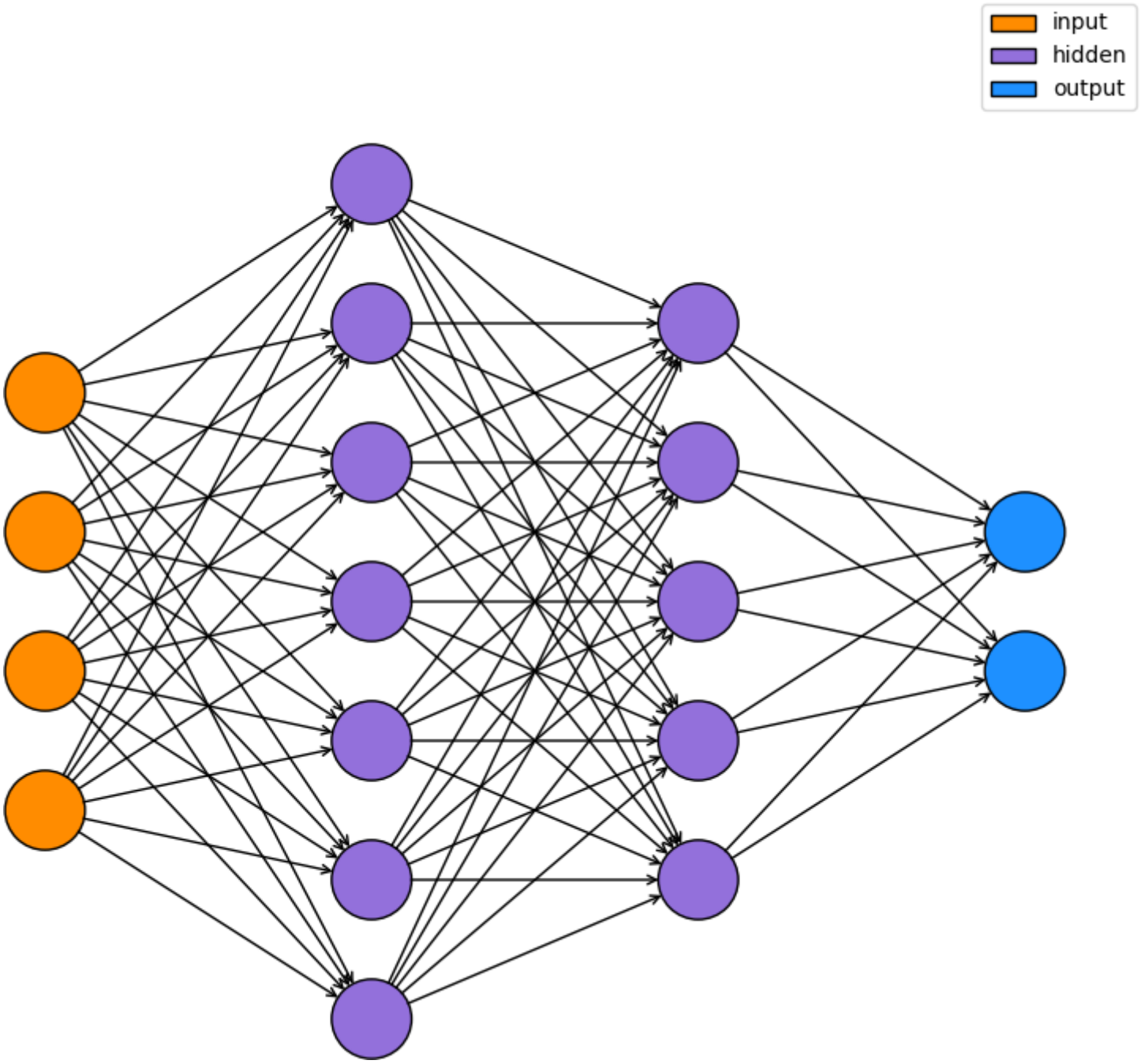
- Train a custom model for visual inspection
- Bring AI into iOS Apps with Core ML
- Classify your images with no additional training

The tutorial used the Watson Visual Recognition service to classify images with built-in and a custom classifiers. The matplotlib library was used to display images. The *classify* capability can be used for local files or it can be pointed to a URL. As with the other services that we have seen the information is transmitted using JSON and there are multiple SDKs that enable access from different environments.

- Watson Visual Recognition docs
- Watson Visual Recognition overview
- Watson Visual Recognition best practices

Neural networks

A multilayer perceptron (MLP) MLPs are sometimes referred to as vanilla neural networks. The number of hidden layers in a MLP and the size (number of notes in each) are configurable parameters that you will need to keep in mind when building neural networks.



MLP networks are comprised of an **input layer**, one or more **hidden layers**, and an **output layer**. Each neuron in the input layer corresponds to a feature and the information propagates from the input layer to the output layer. This is done using a weighted linear summation and a non-linear **activation function**. This left to right movement of information is known as **feed-forward**.

To train most neural networks an important algorithm called backpropagation is used. The algorithms is used to compute the gradient, while another algorithm, such as stochastic gradient descent, carries out the rest of learning process.

Notable types of artificial neural networks:

Multi-layer perceptron

- Feed-forward neural network used for supervised learning. Often uses multiple layers and non-linear activation functions. Can be applied to most supervised learning tasks.

Autoencoders

- It is a feed forward neural network that can be used to predict the feature matrix. Autoencoders may be thought of as being a special case of feedforward network used for unsupervised learning tasks. Example applications include dimension reduction and de-noising.

Convolutional neural network

- A convolutional neural network (CNN, or ConvNet) uses image filters or kernels to learn patterns in a feature matrix. Commonly used to detect patterns in images and video.

Recurrent neural network

- In RNNs connections between nodes can be cyclical, giving the network memory. Used for sequences: handwriting, speech recognition, time series. Once commonly used RNN architecture is the long short-term memory.

CASE STUDY: Tensorflow

The case study demonstrated how to build, iterate on and compare several versions of a simple convolutional neural network (CNN). Among other use cases CNNs have been applied to:

- image classification,
- image segmentation
- image retrieval
- Object detection

The Keras Sequential API was used during the case study to facilitate iteration. Some of the key features from the interface were:

- *model.summary()* is a function is important for understanding the shape of tensors as they are passed between layers
- *models.Sequential()* is an object that provides an intuitive and readable way to build the network
- *model.compile* is needed before training the model
- *model.fit* is used for model training and it works like a scikit-learn estimator

There are several options when it comes to saving and loading TensorFlow models. Saved models can be deployed using with TFLite, TensorFlow.js, TensorFlow Serving, or TensorFlow Hub.

Deploying Models

Data at scale

When it comes to preparing a model for deployment there is a guiding set of steps that may be useful.

- Make it work
- Make it better
- Then make it faster

Before moving into a high-performance computing (HPC) environment like Apache Spark there are some optimizations that might improve performance of models once deployed. In some cases the optimizations are enough to avoid the HPC environment entirely. Some of the important Python packages for code optimization are:

Multiprocessing - This is a package in the standard Python library and it supports spawning processes (for each core) using an API similar to the threading module. The multiprocessing package offers both local and remote concurrency

Threading - Another package in the standard library that allows separate flows flow of execution at a lower level than multiprocessing.

Subprocessing - Module allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes. You may run **and control** non-Python processes like Bash or R with the subprocessing module.

mpi4py - MPI for Python provides bindings of the Message Passing Interface (MPI) standard for the Python programming language, allowing any Python program to exploit multiple processors.

ipyparallel - Parallel computing tools for use with Jupyter notebooks and IPython. Can be used with mpi4py.

Cython - An optimizing static compiler for both the Python programming language and the extended Cython programming language It is generally used to write C extensions for slow portions of code.

PyCUDA. - Python package that allows parallel computing on GPUs via CUDA (Compute Unified Device Architecture)

Supercomputers and parallel computing can help with model training, prediction and other related tasks, but it is worth noting that there are two laws that constrain the maximum speed-up of computing: Amdahl's law and Gustafson's law.

Dealing with data at scale, which is closely related to both code optimization and parallel computing. Apache Spark, is an example of a cluster computing framework that enables to enable parallel computing.

If we talk about scale in the context of a program or model we may be referring to any of the following questions. Let the word **service** in this context be both the deployed model and the infrastructure.

- Does my service train in a reasonable amount of time given a lot more data?
- Does my service predict in a reasonable amount of time given a lot more data?
- Is my service ready to support additional request load?

Docker and containers

Technologies that are commonly used in an environment with numerous Docker containers.

Docker Compose - Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services.

Kubernetes - An open-source system for automating deployment, scaling, and management of containerized applications.

Jenkins - Tool to help automate the model deployment process. Often used for Continuous Integration/Continuous Delivery (CI/CD).

Ansible - A tool for automation to the provision of the target environment and to then deploy the application

Watson Machine Learning Tutorial

Watson Machine Learning (WML) is an IBM service that makes deploying a model for prediction and/or training relatively easy. The Watson Machine Learning Python client was used in this tutorial to connect to the service. You may train, test and deploy your models as APIs for application development, then share the models with colleagues. In this tutorial you saw how you could match your local environment to the requirements of the available runtime environments in WML. You also have the option of iterating on models in Watson Studio and then using nearly the same code those same models could be deployed.

Deploying Models using Spark

Spark Machine Learning

There are two APIs for Spark MLlib. The RDD-based API and the dataframe based API, which is often referred to as Spark ML. Each has its own documentation.

- Spark MLlib docs
- Spark ML docs

Spark MLlib has a suite of available tools for unsupervised learning—namely dimension reduction and clustering. For clustering K-means and Gaussian Mixture Models (GMMs) are the main tools. Latent Dirichlet Allocation (LDA) is available as a tool for clustering over documents of natural language.

- Spark clustering documentation

Spark MLlib has a number of available supervised learning algorithms that is classification and regression. Many of the commonly used algorithms have been implemented including: random forests, gradient boosted trees, linear support vector machines and even basic multilayer perceptrons.

- Spark MLlib - supervised learning

Spark Recommenders

The majority of modern recommender systems embrace either a collaborative filtering or a content-based approach. A number of other approaches and hybrids exists making some implemented systems difficult to categorize.

Collaborative filtering - Based off of a ratings matrix collaborative filtering is a family of methods that infers a subset of users that have behavior similar to a particular user. The items preferred by these users are combined and filtered to create a ranked list of recommended items.

Content-based filtering - Predictions are made based on the properties/characteristics of an item. User behavior is not considered.

Matrix factorization is a class of collaborative filtering algorithms used in recommender systems. Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of lower dimensionality matrices.

There are several Python packages available to help create recommenders including surprise. Because scale with respect to *prediction* is often a concern for recommender systems many production environments use the implementation found in Spark MLlib. The Spark collaborative filtering implementation uses Alternating least Squares.

CASE STUDY: model deployment

We used a Docker image to create a local Spark environment. In this environment a recommender systems was created using Spark MLlib's collaborative filtering implementation. The model itself was tuned, by modifying hyperparameters and by toggling between implicit and explicit versions of the underlying algorithm. spark-submit was used to simulate a deployed model.