Data Science Overview

expedia group™

Introduction

This document is an overview of the data science development process and is designed to inform the considerations and expectations you should have for data science projects.

Acknowledgments

This document is based on slide decks my colleague Brooke Cowan and I developed for a Machine Learning Basics class within Expedia. We thank our Expedia Data Science colleagues and all the students who have provided feedback to help improve the material

I would like to thank everyone who contributed to this document with content and thoughtful review comments. Feel free to reach out to me with questions, and your suggestions for additional material for subsequent versions of this document.

Thomas Crook
tcrook@expedia.com
Data Science Program Director

The Data Science Development and Refinement Process

There are notable differences between the typical data science development process and software engineering development processes. In our experience, data science projects tend to require more research, more time to develop data sources, more trialing of different algorithms and more iterations to refine the system prior to production testing. The payoff comes in more adaptive systems that support feedback loops and can serve a broad range of use cases.

The Data Science Workflow

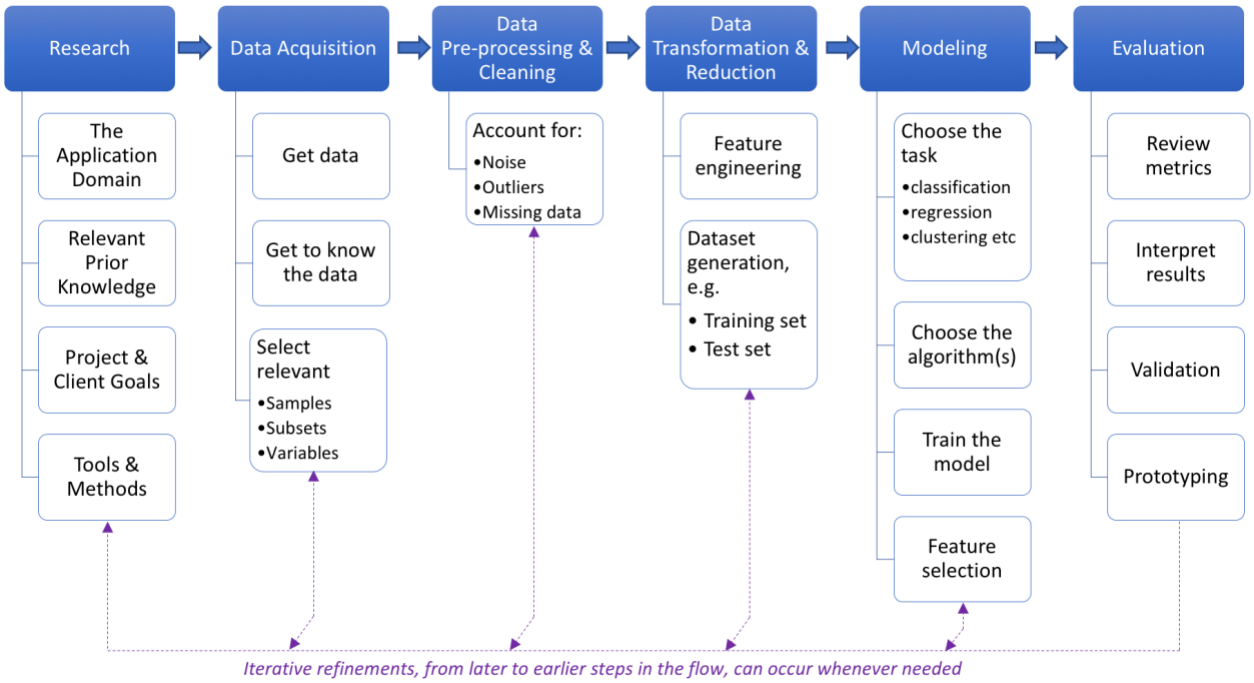

Figure 1: The Data Science Workflow[1]

Steps in the Data Science Workflow

1. Research

The first step in any new machine learning project is to understand the domain. Data scientists work with business experts, product managers, engineers and others to develop a baseline understanding of the business opportunity, risks and requirements.

Activities in this step may include:

- a review of academic literature to get a baseline knowledge on the topic
- enumerating what tools and techniques will help ensure success of the project

[1] Based on:

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, *17*(3), 37.

- identifying user needs, preferences and experiences, and
- documenting engineering performance requirements.

Data are the key (Data Science Workflow Steps 2-4)

The majority of the time spent developing machine learning applications is typically spent acquiring, processing and transforming data. A common estimate is that data scientists spend as much as 70% of their time on these steps.

2. Data Acquisition and Familiarization

Next in the data science workflow is to assess what data are already available and what new sources of data might be required. If new data are required, we may need to work with engineers, product specialists and others to specify and operationalize the data sources.

Once we have some data in hand, we need to assess it. Are there missing data? Is there a lot of variance in the data? What does the distribution of values look like? Does it follow a normal (i.e., bell-shaped) distribution, or is it bi-modal, or something else? Is it long-tailed? fat-tailed? etc. We can often answer some of these questions at a glance by visualizing the data. As humans we can easily perceive patterns in visualizations that we would easily miss in reviewing tables of numbers.

An important step we will almost always want to do is visualize the data.

> **"***Data visualization is a general term that describes any effort to help people understand the significance of data by placing it in a visual context. Patterns, trends and correlations that might go undetected in text-based data can be exposed and recognized easier with data visualization software.*" [2]

If the data do not meet our requirements, we will need to transform it or develop additional data sources.

3. Data Pre-processing and Cleaning

Regardless of our data source, we almost always need to do some pre-processing on it.

We might consider removing *outliers*, as in these two examples:

- *The Royal Penthouse Suite* in the Hotel President Wilson in Geneva goes for more than $80,000 per night. If we're building a hotel recommendation model for families, and we want to incorporate hotel room pricing into the model, we might decide to exclude rooms like this, whose pricing is far outside the norm.
- Our data might include a few invalid prices that we want to exclude. For example, a hotel room price that was intended to be entered in Korean *won* was instead entered in US dollars and the room price is set at $541,150 per night instead of the intended US$ equivalent of $500.

We need to deal with *missing data*. Let's say that we want to build a model that predicts whether hotels in different locations are family friendly or not, and that we needed to supplement our own data with third party data for destinations where we don't have much inventory. Unfortunately, we end up

[2] https://searchbusinessanalytics.techtarget.com/definition/data-visualization

missing room prices for 10% of the hotels. If we anticipate that we might use analyses techniques that are not robust to missing data we will need to mitigate the impact of the missing values. We could throw out all hotel records with missing prices, or we could use an algorithm that imputes the missing prices from the other data we have available.

4. Data Transformation and Reduction

To build all but the simplest models, we will need to *transform* the data in various ways. *Feature engineering* is a fundamental activity in machine learning. In their simplest form, features are just individual fields in our input data. Let's say we want to predict the price we should bid in a metasearch bidding auction. Our experience shows that we get better performing models when we include several transformations of historical prices. We start with the simple data field feature that holds yesterday's price and then create a new feature consisting of the average price over the past week (a transformation of the price field). Then we add a feature of the price over the past 30 days (another transformation), and so on. The process of developing and validating these features is called feature engineering.

We might also derive or *extract* a new feature from existing data, or combinations of existing and external data. For example, given dates and northern or southern hemisphere locations of transactions, we can extract a new *season* feature telling us whether a transaction occurred for a destination's winter, spring, summer or autumn.

As we iteratively refine our model, we'll likely want to do some *feature selection*. We may have hundreds of available features, but only enough computer time and memory to use a subset of them when building our models. We'll try many different subsets of features and probably engage in additional feature engineering to find the mix of features that works best, given our needs and constraints.

Partitioning Data

Also fundamental to machine learning is the unbiased partitioning of data into separate datasets for use in the model development process. Before we can explain the purpose of partitioning data we need to take brief but important detour to explain the basic types of machine learning.

The Basic Types of Machine Learning: Supervised, Unsupervised and Reinforcement

Machine learning algorithms broadly fall into three classes: supervised learning, unsupervised learning and reinforcement learning.

In *supervised learning*, we understand the expected output of the algorithm before building the model and we build the model, in part, by labeling some data. We use this labeled data to train the model what we expect it to output at runtime. For example, if we want to build a model to classify photos as being of humans or cats, we would acquire several photos of the two image classes and label them for use as training data, like this:

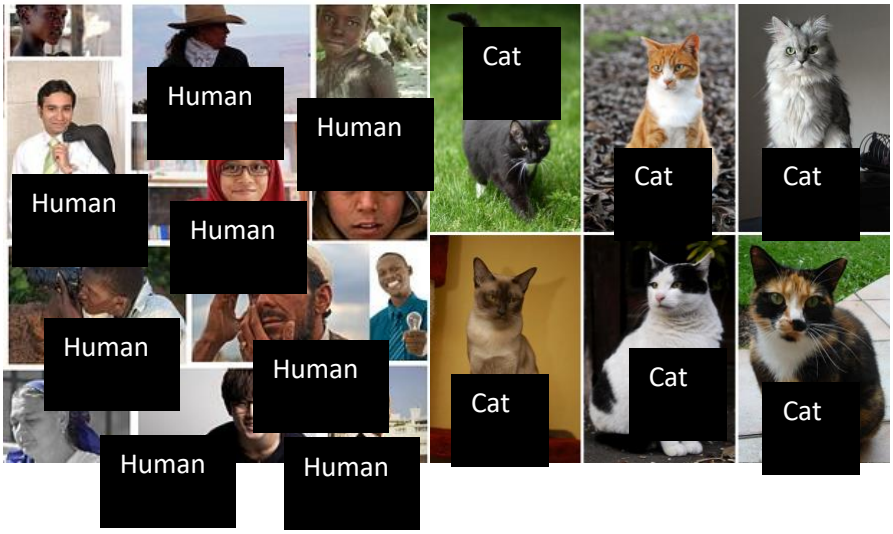

Figure 2: Training dataset with labeled human and cat photos

In *unsupervised learning* the algorithm acts on the data without labeling information. For example, an algorithm can group similar images by detecting common patterns in photos. In a notable 2012 exercise, Google used an unsupervised neural network algorithm to recognize unlabeled thumbnail images of cats and humans with 70% accuracy.

Reinforcement learning algorithms learn by using feedback from previous actions to alter their future actions. In one example, a reinforcement learning algorithm learned to outperform humans in playing 1970's-era video games by iteratively updating its actions in response to in-game outcomes.[3] Unfortunately, developing reinforcement learning applications is very difficult because "real-world reinforcement learning problems have incredibly complicated state and/or action spaces."[4]

Datasets for Model Building

Now that we understand the basic types of machine learning algorithms, we're ready to discuss the types of datasets we may need to partition our data into for model building: *training datasets*, *test datasets* and *validation datasets*. It is imperative that these datasets be kept separate and only be used for their intended purpose. A surefire way to build an unacceptable model is to train it on some or all of the data that is subsequently used to test the model.

We use *training datasets* to *train* or *fit* supervised learning models. For example, say you formulate a hypothesis that hotels with more reviews yield more Raw Margin Dollars for Expedia. You can train the model by making a training set of hotel data and plotting the number of reviews each hotel has against the RMD for that hotel (the blue circles in Figure 3). Using a simple linear regression algorithm, we find the equation for the line through the blue circles that minimizes the collective vertical distance between the circles and the line. This is represented by the line and constitutes our predictive model. We can use the model to predict the output RMD for any input number of reviews.

[3] https://medium.freecodecamp.org/explained-simply-how-deepmind-taught-ai-to-play-video-games-9eb5f38c89ee
[4] http://pemami4911.github.io/blog/2018/01/17/how-hard-is-rl.html

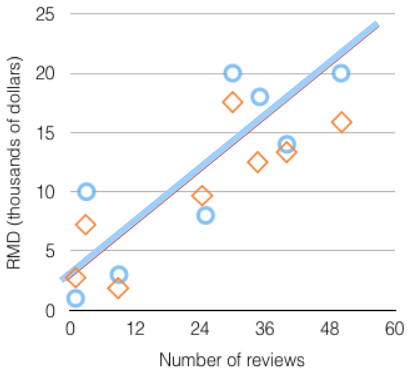

Figure 3: Hypothetical Model of RMD by Number of Hotel Reviews

After training the model we use a *test dataset*, shown by the orange diamonds on the graph, to assess the performance of the model. We do this by calculating the differences between the predicted RMD values (the points that make up the line) and the actual test data (orange diamond) RMD values.

To explain the use of validation data, we'll set linear regression aside and introduce an unsupervised learning algorithm known as k-means. k-means is one of many clustering algorithms. We use clustering algorithms to find previously unknown groups in our data. The K in k-means refers to the number of groups returned by the algorithm and you decide what value of K to use each time you build your model. We refer to input parameters like K, which are not part of our datasets but which we use when creating our models, as *hyperparameters.* Figure 4 shows the effects of setting K to either 2 (the points enclosed by dotted blue lines) or 3 (the points enclosed by solid red lines). We use validation data to assess the performance of our model each time we change hyperparameters.

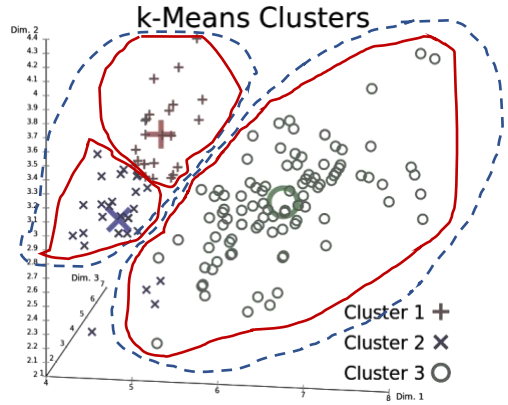

Figure 4: Illustration of k-means clustering

5. Modeling

We use machine learning systems for different tasks. Three important classes of tasks are *regression*, *clustering* and *classification*. We already discussed regression and clustering in the previous section so we'll move straight into classification now.

Classification tasks are used to predict which class an entity belongs to. In contrast to clustering, where we don't know how many groups we will find beforehand, we start with a known number of classes. An example of classification is to determine whether Points of Interest (POI) we import from Google-supplied data are already present in our Gaia geography database. An unknown POI can fall into one of two classes, *Present-in-Gaia* or *Not-present-in-Gaia*.

Given a task, we need to choose the best algorithm for that task. Several factors determine what algorithm will work best for us, including the characteristics of our data, the cost and ease of developing models that use the algorithm, the experience and expertise of our available data scientists, the cost to deploy the model and more. We use many types of machine learning algorithms at Expedia. We have already discussed linear regression and k-means clustering. Now we'll cover some other popular algorithms.

Decision Trees

You are probably already familiar with decision trees in their simplest form. Figure 5 illustrates a simple decision tree model that can be used to classify whether a passenger on the Titanic died or survived given their sex, age and number of family members accompanying them. In addition to classification, decision trees can be used for regression, in which case the leaf nodes would hold continuous values like prices rather than classifications like "died" and "survived."

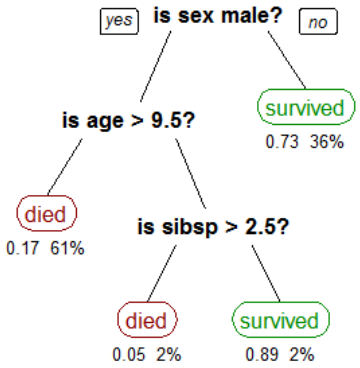

Figure 5: A decision tree that predicts whether a Titanic passenger was a survivor or died — Wikimedia commons image

Ensemble decision tree models are popular in real-world machine learning applications. "Ensemble methods [combine] several decision trees to produce better predictive performance than utilizing a single decision tree." [5] The two most common ensemble methods are *boosting* and *bagging*. Other than mentioning that we use boosting quite a lot at Expedia, we won't go into the details here, referring you instead to this article for a relatively simple introduction.

Naive Bayes Classifiers

Bayes classifiers predict the probability that an object belongs to given class. You may recall Bayes' theorem from your mathematics or statistics class:

[5] https://towardsdatascience.com/decision-tree-ensembles-bagging-and-boosting-266a8ba60fd9

"Bayes' theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event."[6]

In the case of a classifier, the event is that of belonging to a certain class.

They are called naive classifiers because

"…naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features."[7]

Despite their simplicity and "naiveté" in assuming independence between features, naive Bayes classifiers can work quite well in practice. They are used in applications such as sentiment analysis (is a hotel review positive or negative?) and classifying email as spam or not.

Logistic Regression

Like naive Bayes, logistic regression is a relatively simple algorithm that performs well for classification tasks. "Logistic regression uses an equation as the representation [of the model], very much like linear regression.[8] "

We can write the equation that represents the fitted model like this:

$$y = f(x) = \sigma(w_1 x_1 + \cdots + w_d x_d + b)$$

The function σ transforms the sum of the terms inside the parentheses into the range from 0 to 1, enabling us to interpret it as a probability. For example, if the value is 0.74 we can interpret that there is a 74% chance that *y* is a member of the class of interest. The nice thing about the logistic regression representation is that the model is very easy to interpret. The x's in the equation are the model features and the w's are the weights that express the relative contribution of each feature to the overall score. Figure 6 shows the first few features and weights from a model that predicts whether a movie will generate a high amount of revenue, sorted by weight. (We believe that *duration* is the highest weighted feature here because feature films, which make a lot of money, are lengthy compared to movies as a whole. In contrast documentaries are relatively short and don't make nearly so much money at the box office.)

Feature Weights

Feature	Weight
duration	4.19936
cast_total_facebook_likes	2.25612
director_facebook_likes	2.00444
Bias	-1.9251
actor_1_facebook_likes	1.61026
actor_2_facebook_likes	1.50208
actor_3_facebook_likes	1.37835
Biography	1.02381
Fantasy	0.871521
Sci-Fi	0.860016
Comedy	-0.825805
student(40)	0.773526
Adventure	0.771341

Figure 6: Movie popularity model features and weights

[6] https://en.wikipedia.org/wiki/Bayes%27_theorem
[7] https://en.wikipedia.org/wiki/Naive_Bayes_classifier
[8] https://machinelearningmastery.com/logistic-regression-for-machine-learning/

Neural networks & Deep learning

The concept of artificial neural networks (ANNs) has been around for a long time in computer science terms—some of the underlying theories were developed as far back as the 1940's. However, artificial intelligence research mostly went in other directions. It wasn't until the 2000's that advances in algorithms and computer hardware made the implementation of useful large-scale networks practical. In 2006, Geoffrey Hinton of the University of Toronto and his colleagues published important work that kicked off a "deep learning" neural network renaissance.[9] This Toronto Star newspaper article[10] provides some very readable background and history.

Recently, the deep reinforcement learning system AlphaGo won acclaim for beating the best human players in the world in the game of Go.[11] As we noted earlier, reinforcement learning is difficult to implement for general use. We don't expect to see it widely used in commercial applications in the near future.

Deep learning is realized using multi-layered neural networks with visible input and output layers and multiple "hidden" layers. Referring to Figure 7, we see that there can be many connections between network nodes in different layers. Neural networks work by associating weights with the connections and incorporating an "activation" function in each node whose output depends on the weights into the node. A complex network may have hundreds or thousands of nodes and the connections and weights in the network constitute a pattern which reacts to the data at the input to produce its output.

ANNs can perform many tasks. They have enabled breakthroughs in fields such as image recognition. (Think back to our example in Figure 2 where an unsupervised neural network learned to classify images with humans and cats.) We use ANNs at Expedia for tasks such as ranking the quality of hotel images and bidding on metasearch.

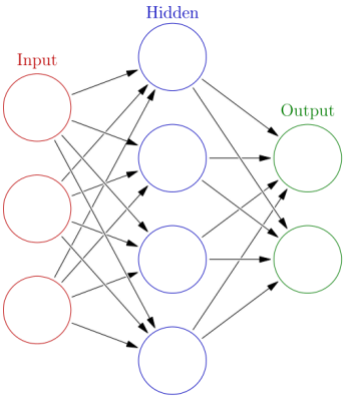

Figure 7: Artificial neural network with input, hidden and output layers. So-called "deep networks" can have many hidden layers.
- Credit: Glosser.CA: CC BY-SA 3.0

6. Model Evaluation

Once we've created a machine learning model we need to assess its performance and suitability for the use case it was developed for.

Model performance methods depend, of course on the model type. Here, we review some key metrics for some selected model types:

Classification Task Performance Metrics

The following metrics can be used to gauge the performance of classifiers regardless of the algorithm we used.

[9] http://www.cs.toronto.edu/~hinton/absps/fastnc.pdf
[10] https://www.thestar.com/news/world/2015/04/17/how-a-toronto-professors-research-revolutionized-artificial-intelligence.html
[11] https://en.wikipedia.org/wiki/AlphaGo

- **Accuracy** measures the goodness of the model as the proportion of true results to total cases: (true positives + true negatives) / (true positives + true negatives + false positives + false negatives)
- **Precision** is the proportion of true positive results divided by all positive results: (true positives) / (true positives + false positives)
- **Recall** is the number of correct results returned by the model / all positives in the test data. (true positives) / (true positives + false negatives)

See this article for a more comprehensive, easy-to-understand treatment of classification metrics.[12]

Regression Model Performance Metrics

The following excerpt covers the most basic regression model performance metrics:

> A well-fitting regression model results in predicted values close to the observed data values. The mean model, which uses the mean for every predicted value, generally would be used if there were no informative predictor variables. The fit of a proposed regression model should therefore be better than the fit of the mean model.
>
> Three statistics are used in Ordinary Least Squares (OLS) regression to evaluate model fit: R-squared, the overall F-test, and the Root Mean Square Error (RMSE). All three are based on two sums of squares: Sum of Squares Total (SST) and Sum of Squares Error (SSE). SST measures how far the data are from the mean and SSE measures how far the data are from the model's predicted values. Different combinations of these two values provide different information about how the regression model compares to the mean model.

For more information see the complete article.[13]

Cluster Model Performance Metrics

The very nature of the task of finding clusters in data is somewhat arbitrary. The number and shape of clusters found depends on our choices while carrying out the analysis. We should validate that the clusters map well to real-world groups represented by the data and try to strike a balance between generalization afforded by fewer clusters against the accuracy of having lots of clusters. These references provide more guidance on evaluating the output of cluster models:

- Determining the number of clusters in a data set[14]
- scikit-learn Clustering performance evaluation[15]

Real-world Test and Learn

Once we have verified that our models meet standard development performance evaluation criteria, we need to test them in real-world conditions using our standard A/B and multivariate testing processes. It

[12] https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c
[13] https://www.theanalysisfactor.com/assessing-the-fit-of-regression-models/
[14] https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set
[15] http://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation

is quite possible to build a model that looks great on standard performance measures using our test and validation data and have it perform poorly in a real-world A/B test.

Iterations in the Data Science Workflow

As denoted in Figure 1, iterations occur in the Data Science Workflow whenever needed. Figure 8 illustrates how we incorporate data loops into production ML systems.

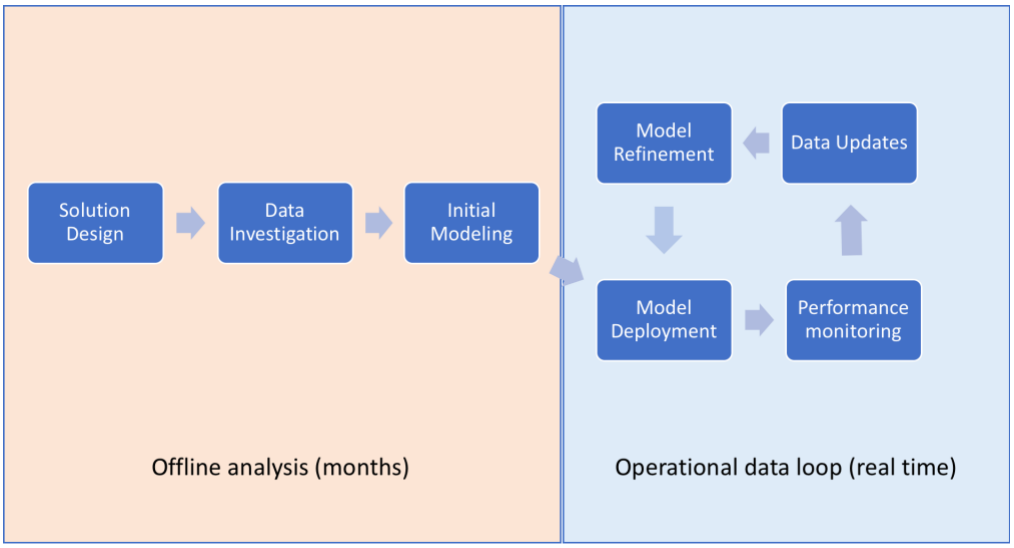

Figure 8: Data Loops and Model Refinement

Typical Machine Learning Project Time and Resource Requirements

In our experience, teams and managers who are new to machine learning tend to underestimate the initial effort needed to develop an ML-based Minimum Viable Product. Table 1 will help inform their expectations. As noted previously, the project phases that include developing the initial data sources and machine learning features tend to be the most time consuming. In our experience those activities can consumer as much as 70% of the pre-production schedule.

Also of note are the high skill and experience levels of key staff involved in delivering successful ML projects. Most of the data scientists have PhD level or equivalent educations and the Leads have PhDs plus years of industry experience.

Activity	Time	Role	Skill Level
Research: - Translating business problem into ML requirements. - Planning the initial ML approach	Weeks	Sr. DS Lead	High
Data Acquisition & Pre-processing	Months	DS 2 / Data Engineer	Medium
Data Transformation & Feature engineering	Weeks / Months	DS 2	Medium / high
Modeling	Weeks	Sr. DS Lead	High
Model Evaluation	Weeks	DS 2	Medium
Performance monitoring	Ongoing	DS 2 / Analytics	Medium
Model refinement	Ongoing	DS2 / Sr. DS	High

Table 1: Typical ML Project Time and Resource Requirements