**Session-19**

**Machine Learning-1**

**Problem Statement 1:** What are the three stages to build the hypotheses or model in machine learning?

**Answer:** There are 3 stages to build the hypotheses or model in machine learning.

1. **Model Building:**

The goal of this phase is to align as a team on the key problem the model solves, the objective function and the potential inputs to the model.

 **Align on the problem.** Machine learning needs to be used to solve a real business problem. Make sure all the stakeholders on our team and in the company agree on the problem we’re solving and how we’ll use the solution.

 **Choose an objective function.** Based on the problem, decide what the goal of the model should be. Is there an objective function the model is trying to predict? Is there some measure of “truth” we’re trying to get to that we can verify against “ground truth” data, e.g. home prices, stock price changes etc.? Alternatively, are we just trying to find patterns in data? For example, cluster images into groups that have something in common?

 **Define quality metrics.** How would we measure the model’s quality? It is sometimes difficult to foresee what acceptable quality is without actually seeing the results, but a directional idea of the goal is helpful.

 **Brainstorm potential inputs.** Our goal is to decide what data could help we solve the problem / make decisions. The most helpful question to ask is: “How would an expert in the space approach this problem?” Think what would be the variables / pieces of data that person would base a solution on. Every factor that may affect human judgement should be tested — at this stage go as broad as possible. Understanding the key factors may require problem business space knowledge, which is one of the reasons it’s important for business / product people to be heavily involved at this stage. The data team will have to translate these potential inputs into model features. Please note that in order to turn inputs into features additional processing may be required — more on that next.

**Data Preparation**

The goal of this phase is to collect raw data and get it into a form that can be plugged as an input into our prototype model. We may need to perform complex transformations on the data to achieve that. For example, suppose one of our features is consumer sentiment about a brand: We first need to find relevant sources where consumers talk about the brand. If the brand name includes commonly used words (e.g. “Apple”), we need to separate the brand chatter from the general chatter (about the fruit) and run it through a sentiment analysis model, and all that before we can begin to build our prototype. Not all features are this complex to build, but some may require significant work.

Let’s look at this phase in more detail:

* **Collect data for your prototype in the fastest way possible.** First, identify our missing data. In some cases we may have to break down the necessary inputs to get to the “building blocks” level of raw data that is more easily available, or to data that is a close proxy to what we need and is easier to get. Once identified, figure out the quickest, easiest way to get our data. Non-scalable methods such as a quick manual download, writing a rudimentary scraper or buying a sample of data even if a little expensive may be the most practical approach. Investing too much in scaling our data acquisition at this stage usually doesn’t make sense, since we don’t yet know how useful the data would be, what format would be best etc. Business people should be involved — they can help brainstorm ways to find data that is not readily available or simply get it for the team (the relevant business functions to involve depend on the data needs and the org structure — partnerships, business development or marketing may be helpful here). Note that in the case of a supervised learning algorithm, we need data not just for the model features; we need “ground truth” data points for our model’s objective function in order to train and then verify and test our model. Back to the home prices example — in order to build a model that predict home prices, we need to show it some homes with prices!
* **Data cleanup and normalization.** At this stage the responsibility largely moves to our data science / engineering team. There is significant work involved in translating ideas and raw data sets into actual model inputs. Data sets need to be sanity checked and cleaned up to avoid using bad data, irrelevant outliers etc. Data may need to be transformed into a different scale in order to make it easier to work with or align with other data sets. Especially when dealing with text and images, pre-processing the data to extract the relevant information is usually required. For example, plugging too many large images into a model results in an enormous amount of information that may not be feasible to process, so we may need to downgrade the quality, work with a portion of the image or use only the outlines of objects. In the case of text, we may need to detect the entities that are relevant to us in the text before we decide to include it, perform sentiment analysis, find common [n-grams](https://en.wikipedia.org/wiki/N-gram) (frequently used sequences of a certain number of words) or perform a variety of other transformations. These are usually supported by existing libraries and don’t require our team to reinvent the wheel, but they take time.

1. **Model Testing:**

The goal of this stage is to get to a prototype of a model, test it and iterate on it until you get to a model that gives good enough results to be ready for production.

* **Build prototype.** Once the data is in good shape, the data science team can start working on the actual model. Keep in mind that there’s a lot of art in the science at this stage. It involves a lot of experimentation and discovery — selecting the most relevant features, testing multiple algorithms etc. It’s not always a straightforward execution task, and therefore the timeline of getting to a production-ready model can be very unpredictable. There are cases where the first algorithm tested gives great results, and cases where nothing you try works well.
* **Validate and test prototype.**At this stage our data scientists will perform actions that ensure the final model is as good as it can be. They’ll assess model performance based on the predefined quality metrics, compare the performance of various algorithms they tried, tune any parameters that affect model performance and eventually test the performance of the final model. In the case of supervised learning they’ll need to determine whether the predictions of the model when compared to the ground truth data are good enough for our purposes. In the case of unsupervised learning, there are various techniques to assess performance, depending on the problem. That said, there are many problems where just eyeballing the results helps a lot. In the case of clustering for example, we may be able to easily plot the objects we cluster across multiple dimensions, or even consume objects that are a form of media to see if the clustering seems intuitively reasonable. If our algorithm is tagging documents with keywords, do the keywords make sense? Are there glaring gaps where the tagging fails or important use cases are missing? This doesn’t replace the more scientific methods, but in practice helps to quickly identify opportunities for improvement. That’s also an area where another pair of eyes helps, so make sure to not just leave it to your data science team.
* **Iterate.** At this point we need to decide with our team whether further iterations are necessary. How does the model perform vs. our expectations? Does it perform well enough to constitute a significant improvement over the current state of our business? Are there areas where it is particularly weak? Is a greater number of data points required? Can we think of additional features that will improve performance? Are there alternative data sources that would improve the quality of inputs to the model? Etc. Some additional brainstorming is often required here.

1. **Applying the model:**

We get to this stage when we decide that our prototype model works well enough to address our business problem and can be launched in production. Note that we need to figure out which dimensions we want to scale our model on first if we’re not ready to commit to full productization. Say our product is a movie recommendation tool: We may want to only open access to a handful of users but provide a complete experience for each user, in which case our model needs to rank every movie in our database by relevance to each of the users. That’s a different set of scaling requirements than say providing recommendations only for action movies, but opening up access to all users.

Now let’s discuss the more technical aspects of productizing a model:

* **Increase data coverage.**In many cases we prototype our model based on a more limited set of data than we would actually use in production. For example, we prototype the model on a certain segment of customers and then need to broaden it to our entire customer base.
* **Scale data collection.** Once we verified which data is useful for the model, we need to build a scalable way to gather and ingest data. In the prototyping phase it was fine to gather data manually and in an ad-hoc fashion, but for production we want to automate that as much as possible.
* **Refresh data.**Create a mechanism that refreshes the data over time — either updates existing values or adds new information. Unless for some reason we don’t need to keep historical data, our system needs to have a way to store growing quantities of data over time.
* **Scale models.** There is both a data science and an engineering aspect to this. From a data science perspective, if we changed the underlying data, e.g. expanded the number of customer segments we include, we need to retrain and retest our models. A model that works well on a certain data set won’t always work on a broader or otherwise different data set. Architecturally, the model needs to be able to scale to run more frequently on growing quantities of data. In the movie recommendations example that would likely be more users, more movies and more information about each user’s preferences over time.
* **Check for outliers.** While the model as a whole may scale very well, there may be small but important populations that the model doesn’t work well for. For example, our movie recommendations may work very well for users on average, but for parents we’ll show mostly kids movies because they choose movies for the kids from their account. This is a product design problem — we need to separate the recommendations for the parent from the recommendations for their kids in the product, but this is not something the model will just tell us.

**Problem Statement 2:** What is the standard approach to supervised learning?

**Answer:** The standard approach to supervised learning is to split the set of example into the training set and the test and also

1. Determine the type of training examples. Before doing anything else, the engineer should decide what kind of data is to be used as an example.

2. Gather a training set. The training set needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.

3. Determine the input feature representation of the learned function. The accuracy of the learned function depends strongly on how the input object is represented. Typically, the input object is transformed into a feature vector, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.

4. Determine the structure of the learned function and corresponding learning algorithm. For example, the engineer may choose to use support vector machines or decision trees.

5. Complete the design. Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a validation set) of the training set, or via cross-validation.

6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.

**Problem Statement 3:** What is Training set and Test set?

**Answer:**

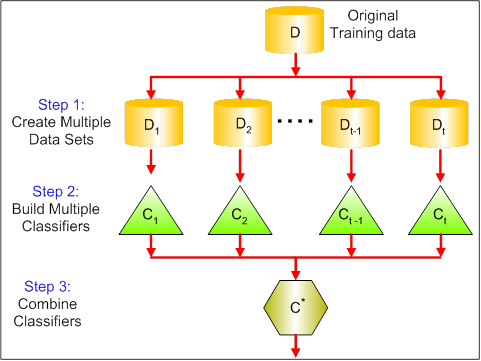
**Training set:** In machine learning, a training set is a dataset used to train a model. In training the model, specific features are picked out from the training set. These features are then incorporated into the model. Thereby, if the training set is labelled correctly, the model should be able to learn something from these features. Here a set of data is used to discover the potentially predictive relationship known as ‘Training Set’. Training set is an examples given to the learner.

**Test Set:** The test set is a dataset used to measure how well the model performs at making predictions on that test set. If the prediction scores for the test set are unreasonable, we’ll need to make some adjustments to our model and try again. While Test set is used to test the accuracy of the hypotheses generated by the learner, and it is the set of example held back from the learner.

**Problem Statement 4:** What is the general principle of an ensemble method and what is Bagging and Boosting in ensemble method?

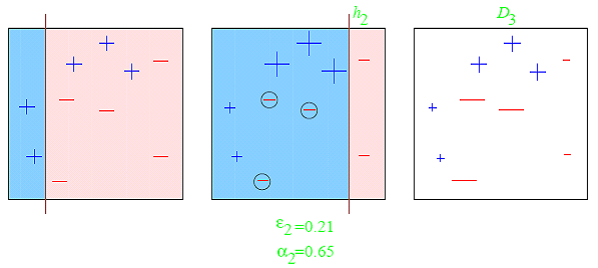
**Answer:** Ensemble model combines multiple ‘individual’ (diverse) models together and delivers superior prediction power. The general principle of an ensemble method is to combine the predictions of several models built with a given learning algorithm in order to improve robustness over a single model.

* **Bagging:** Bagging (Bootstrap Aggregating) is an ensemble method. First, we create random samples of the training data set (sub sets of training data set). Then, we build a classifier for each sample. Finally, results of these multiple classifiers are combined using average or majority voting. Bagging helps to reduce the variance error.



* **Boosting:**

**Boosting provides sequential learning of the predictors. The first predictor is learned on the whole data set, while the following are learnt on the training set based on the performance of the previous one. It starts by classifying original data set and giving equal weights to each observation. If classes are predicted incorrectly using the first learner, then it gives higher weight to the missed classified observation. Being an iterative process, it continues to add classifier learner until a limit is reached in the number of models or accuracy. Boosting has shown better predictive accuracy than bagging, but it also tend to over-fit the training data as well.**



**Problem Statement 5:** How can you avoid overfitting?

**Answer:** We can avoid overfitting by following below best practices.

1. Cross-validation: Use initial training data to generate multiple mini train-test splits. Use these splits to tune the model.

2. Train with more data.

3. Remove features.

4. Early stopping: Early stopping refers stopping the training process before the learner passes that point.

5. Regularization: Regularization refers to a broad range of techniques for artificially forcing the model to be simpler.

6. Ensembling.