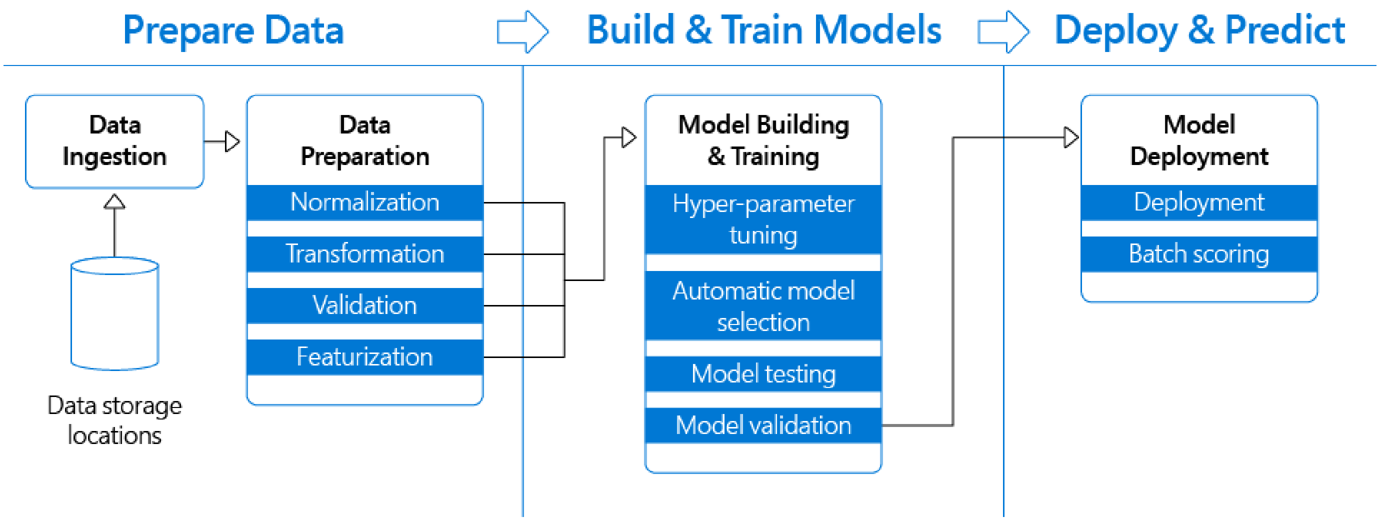
End to End Pipeline in Machine Learning



This **end to end pipeline can be divided into a few steps** for better understanding, and those are:

Understanding the problem statement

Acquiring the required data

Understanding the data

Cleaning the data

Selecting the best model for training

Fine-tuning the hyperparameters

Presenting the results

Deploying and maintaining the system

Understanding the problem statement

In order to build a good solution, one needs to understand the problem statement very clearly. You will most probably end up building and training a Machine Learning model but real-life application areas need much more than just the models. The model's output should be matched with what exactly is needed by the end-user.

The important reason for this step is to exactly understand what needs to be done and exactly what kind of solution is needed. This is where the main brainstorming part is done for how the problem statement must be approached.

Acquiring the required data

Once you have understood the problem statement clearly and have decided to move forward with a Machine Learning approach to solve the problem, you should start searching for relevant data. Data is the most important ingredient of any Machine Learning project so you must carefully find and select the quality data only. The final performance of the ML models depends on the data that was used while

Understanding the data

It is a very important aspect of the ML solution to be able to understand the data that you are working with.

This enables us to choose which algorithms or model architectures are better suited for the project.

Before starting to look at the data in detail, it is a good practice to first split the dataset into train and test sets. This keeps the test set untouched and hence decreases the chances of overfitting to the test set. By doing this, you are eliminating the data snooping bias from the model.

There are **various ways of splitting the datasets** into **these train and test sets. One of these is splitting it with a hardcoded percentage value**. **90% train and 10% test** is a common value in most of the cases.

After the splitting, you will have to visualize the train set in-depth to understand the data.

Finding the **correlation between two attributes in the dataset** is helpful to understand which attributes relate more to the required attribute.

Cleaning the data

In this step, you prepare the data for the Machine Learning project. It is the most time consuming and important step of the entire pipeline.

The performance of the model majorly depends on how well you prepare the data. Usually, **it is a good practice to write functions for this purpose as it will allow you to use those functions whenever needed and the same functions** can be used in the production line to prepare the new data for predictions.

One of the most encountered problems in **real data is the missing values for a few entries in the dataset.**

There are a few ways of handling it.

You can **directly delete the entire attribute but this is not very good for the model**.

You can get rid of the row which has one missing value. **Another way which is mostly used is to set the missing value to some other value like zero or the arithmetic mean of the entire column if it is a numeric value**.

For **categorical values, it is better to represent them by numbers and encoding them into a one-hot encoding so that it is easier for the model to work on it**. Scikit-Learn also provides the OneHotEncoder class so that we can easily convert categorical values into one-hot vectors

Another thing that you have to look after is the **feature scaling. There might be some attributes whose value ranges are very drastic. So it is better to scale them to a standard scale so that the model can easily work with those values and perform better.**

Selecting the best model for training

After completing all the **data cleaning and feature engineering**, the next step becomes quite easy.

Now, all you have to do is train some promising models on the data and find out the model that gives the best predictions.

There are a few ways that help us select the best model.

Linear regression and Linear classifier. Despite an apparent simplicity, they are very useful on a huge amount of features where better algorithms suffer from overfitting.

Logistic regression is the simplest non-linear classifier with a linear combination of parameters and nonlinear function (sigmoid) for binary classification.

Decision trees are often similar to people's decision process and is easy to interpret. But they are most often used in compositions such as Random Forest or Gradient boosting.

K-means is more primal, but a very easy to understand the algorithm, that can be perfect as a baseline in a variety of problems.

PCA is a great choice to reduce the dimensionality of your feature space with minimum loss of information.

Neural Networks are a new era of machine learning algorithms and can be applied for many tasks, but their training needs huge computational complexity.

The **first step here is to train a few models and test them on the validation set. You should not use the test set here as it will lead to overfitting on the test set and eventually the model will have a very low regularization.** From those models, the model with good training accuracy and validation accuracy should be chosen most of the time. It may also depend on the use case as some tasks require different configurations than others.

As we have already cleaned up the data and the preprocessing functions are ready, it is very easy to train different models in three to four lines of code using some frameworks like Scikit-Learn or Keras. In Scikit-Learn we also have an option of cross-validation which helps a lot to find good hyperparameters for models like decision trees.

Fine-tuning the hyperparameters

After having a few models shortlisted there comes a need for fine-tuning the hyperparameters to unleash their true potential.

There are many ways to achieve this too. One of which is that you can manually change the hyperparameters and train the models again and again till you get a satisfactory result.

Here you can clearly see the problem that you cannot possibly check out as many combinations as an automated task would. So here comes in some good methods to automate this stuff.

Grid Search is a wonderful feature provided by Scikit-Learn in the form of a class **GridSearchCV where it does the cross-validation on its own and finds out the perfect hyperparameter values for better results.** All we have to do is mention which hyperparameters it has to experiment with. It is a simple but very powerful feature.

Randomized search is another approach that can be used for a similar purpose. Grid Search works well when there is a small space of hyperparameters to be experimented with but when there's a large number of hyperparameters, it is better to use the RandomizedSearchCV. It tries random hyperparameters and comes up with the best values it has seen throughout.

Last but not least, is the approach of Ensemble Learning. Here we can use multiple models to give their respective predictions and at last, we can choose the final prediction as to the average of all.

Presenting the results

Once the best model is selected and the evaluation is done, there is a need to properly display the results. Visualization is the key to making better Machine Learning projects as it is all about data and understanding the patterns behind it. The raw numeric results can sound good to people already familiar with this domain but it is very important to visualize it on graphs and charts as it makes the project appealing and everyone can get a clear picture of what actually is happening in our solution.

Deploying and maintaining the system

There are **two types of Machine Learning models that can be deployed**:

An online model and an offline model.

The online model is the one that keeps learning from the data that it is receiving in real-time.

Offline models do not learn from new samples and have to be updated and maintained properly if there is a change in the kind of data received by it.

So there needs to be proper maintenance for both types of models.

While deploying Machine learning models, they need to be wrapped in a platform for the users to have ease in interacting with them.

The options are wide, we can wrap it in a web app, android app, Restful API, and many more.

Basic knowledge of building such apps or APIs is a huge plus point.

You should be able to deploy NodeJS or Python apps on cloud services like Google Cloud Platforms, Amazon Web Services, or Microsoft Azure.

If you are not comfortable with some frameworks like Django or Flask, you can try out Streamlit which allows you to deploy a python code in the form of a web app in just a few lines of additional code. There are various such libraries and frameworks which can be explored.