Gold BlogMachine Learning Model Development and Model Operations: Principles and Practices

The ML model management and the delivery of highly performing model is as important as the initial build of the model by choosing right dataset. The concepts around model retraining, model versioning, model deployment and model monitoring are the basis for machine learning operations (MLOps) that helps the data science teams deliver highly performing models.

1. Introduction

The use of Machine Leaning (ML) has increased substantially in enterprise data analytics scenarios to extract valuable insights from the business data. Hence, it is very important to have an ecosystem to build, test, deploy, and maintain the enterprise grade machine learning models in production environments. The ML model development involves data acquisition from multiple trusted sources, data processing to make suitable for building the model, choose algorithm to build the model, build model, compute performance metrics and choose best performing model. The model maintenance plays critical role once the model is deployed into production. The maintenance of machine learning model includes keeping the model up to date and relevant in tune with the source data changes as there is a risk of model becoming outdated in course of time. Also, the configuration management of ML model play an important role in model management as the number of models grow. This article focuses on principles and industry standard practices, including the tools and technologies used for ML model development, deployment, and maintenance in an enterprise environment.

2. Model Development

Machine Learning (ML) Model Lifecycle refers to the process that covers right from source data identification to model development, model deployment and model maintenance. At high level, the entire activities fall under two broad categories, such as ML Model Development and ML Model Operations.

Scale Data Science by Switching to KNIME

Machine Learning (ML) model development includes a series of steps as mentioned in the Fig. 1.

Figure

Fig 1: Machine Learning (ML) Model Development Lifecyle

The ML model development lifecycle steps can be broadly classified as – data exploration, model building, model hyperparameters tuning and model selection with optimum performance.

Exploratory data analysis is an important step that starts once business hypothesis is ready. This step takes 40-50% of total project time as the model outcome depends on the quality of input data being fed to train the model. Exploratory data analysis involves data attributes identification, data preprocessing and feature engineering. Attributes' identification involves identification of predictor/features variables (inputs) and target/class variable (output), along its data types (string or numeric or datetime) and classification of features into categorical and continuous variables that helps in applying appropriate treatment to be given to the variable by the algorithm while building the model. Data pre-processing involves identification of missing values and outliers and fill these

gaps by computing mean or median for quantitative attributes and mode for qualitative attributes of data to improve the predictive power of model. The outliers cause increased mean and standard deviation, that can be eliminated by taking natural log value which reduces the variation caused by extreme values.

Feature Engineering is the next most important step in exploratory data analysis where the raw dataset is processed to convert data types of string or datetime or numeric ones to numeric vectors for an ML algorithm to understand and build efficient predictive model. Also, the labelled categorical data (e.g., color red, green, blue; performance → poor, fair, good, very good, excellent; risk → low, medium, high; status → started, in-progress, on-hold, closed; gender → male/female; is_loan_approved/is_claim_fraudulent → yes/no) cannot be understood by an ML algorithm in its true context, hence it is required to be converted into numeric data. Feature Encoding is the widely used technique to transform the categorical data into continuous (numerical) values, e.g., Encode color as 1,2,3; performance as 1,2,3,4,5; risk as 1,2,3; status as 1,2,3,4; gender / is_loan_approved / is_claim_fraudulent as 0/1 etc. It is recommended to use label encoding in case categorical variables have no ordered relationship (e.g., color and status), ordinal encoding in case categorical variables have an ordered relationship (e.g., performance, risk) and one hot encoding in case categorical variable data is binary in nature (e.g., gender, is_loan_approved, is_claim_fraudulent). A set of libraries are available in R or Python to implement these encoding methods. In some cases, a set of dummy variables or derived variables are created, especially in handling 'date' data types. Once the categorical text data is converted into numeric data, the data is ready to be fed to the model; As a last step, it is required to choose the appropriate features that help in improving the accuracy of model, by using the techniques such as Univariate Selection (statistical measure), Feature Importance (model property) and Correlation Matrix (identifies which features are most related to the target variable). These methods detect Collinearity between two variables where they are highly correlated and contain similar information about the variance within a given dataset. One may find the Variance Inflation Factor (VIF) useful to detect Multicollinearity where highly correlated three or more variables are included in the model. The key points in exploratory data analysis are represented in Fig.2, as shown below

Figure

Fig 2: Exploratory Data Analysis

Building an ML Model requires splitting of data into two sets, such as 'training set' and 'testing set' in the ratio of 80:20 or 70:30; A set of supervised (for labelled data) and unsupervised (for unlabeled data) algorithms are available to choose from depending on the nature of input data and business outcome to predict. In case of labelled data, it is recommended to choose logistic regression algorithm if the outcome to predict is of binary (0/1) in nature; choose decision tree classifier (or) Random Forest® classifier (or) KNN if the outcome to predict is of multi-class (1/2/3/...) in nature; and choose linear regression algorithm (e.g., decision tree regressor, random forest regressor) if the outcome to predict is continuous in nature. The clustering (Unsupervised) algorithm is preferred to analyze unlabeled / unstructured (text) data (e.g., k-means clustering). Artificial Neural Network (ANN) algorithms are suggested to analyze other unstructured (image/voice) data types, such as Convolutional Neural Networks (CNN) for image recognition and Recurrent Neural Networks (RNN) for voice recognition and Natural Language Processing (NLP). The model is built using training dataset and make prediction using test dataset. Use of deep learning (neural networks) models is preferred over regression models (ML models) for better performance as these models introduce extra layer of non-linearity with the introduction of Activation Function (AF). The algorithm selection under various scenarios has been represented in Fig.3, as shown below.

Figure

Fig 3: Choose Right Algorithm

Computation of Model Performance is next logical step to choose the right model. The model performance metrics will decide on final model selection, that include computation of accuracy, precision, recall, F1 score (weighted average of precision and recall) , along with confusion matrix for classification models and co-efficient of determination for regression models. It is not recommended to use accuracy as a measure to determine the performance of classification models that are trained with imbalanced/skewed datasets, rather precision and recall are recommended to be computed to choose right classification model.

Model Hyperparameters Tuning is highly recommended step in the process, continue till the model performance reach around 80%-85%. For example, the Random Forest algorithm takes maximum depth, maximum number of features, number of trees etc., as hyperparameters which can be intuitively tuned for improving model accuracy. Similarly, Neural Networks algorithm takes number of layers, batch size, number of epochs, number of samples etc. It is recommended to use Grid-search method to find the optimal hyperparameters of a model which results in the most 'accurate' predictions. Besides this, it is also recommended to perform cross-validation as sometimes improvement in model accuracy may be due to overfitting (too sensitive model, unable to generalize) or underfitting (very generalized model) of model, using k-fold cross validation technique. To avoid overfitting, increase training data sample size that introduces more patterns or, reduce number of features that avoids complexity or, perform data regularization data using Ridge and Lasso regularization methods that reduces error/penalty. Similarly, to avoid underfitting, increase model complexity such as moving from linear to non-linear or adding more hidden layers (epochs) to neural network or add more features that introduce hidden patterns. However, adding more data volume does not solve the problem of underfitting, rather it hampers the model performance.

Finally. Choose the model with optimum performance.

ML Model Development Best Practices: The recommended ML Model Development Best Practices are (1) Perform clear hypothesis for the identified business problem before attributes identification itself; (2) Build the model using a basic algorithm first, like a logistic regression or decision tree and compile performance metrics, that gives enough confidence about relevance of data before going for a fancier algorithms, like neural networks; (3) Keep intermediate checkpoints in building the model to keep track of its model hyperparameters and its associated performance metrics that gives an ability to train the model incrementally and make good judgement when it comes to performance vs training time; (4) Use real-world production data for training the model to improve correctness of predictions.

Model Operations:

Machine Leaning (ML) Model Operations refers to implementation of processes to maintain the ML models in production environments. The common challenge encountered in a typical enterprise

scenario is that the ML models worked in lab environment will remain stay at the proof-of-concept stage in many cases. If the model is rolled out into production, it becomes stale due to frequent source data changes that requires rebuilding of model. As the models are retrained multiple times, it is required to keep track for model performance and corresponding features and hyperparameters that are used for retraining the model. To perform all these operations, there should be a well-defined reproducible process in-place to implement the end-to-end machine learning operations (MLOps) that keeps the model current and accurate in production environment. The ML model operations lifecycle process is as shown in Fig. 4 that covers entire process of model development to model deployment to model performance monitoring in a seamless manner.

Figure

Fig 4: Machine Learning (ML) Model Operations Lifecyle

The model retraining is very important at regular intervals, say fortnightly or monthly or quarterly or on-demand basis as it is very likely that the underlying source data will change over a period in the real-world scenario. A cron job is scheduled to retrain the model at the predefined intervals, or as and when the source data is changed, or as and when the model performance is degraded. There could be some changes to the model code due to hyperparameter tuning during model re-training.it is required to automate these model operations by having an automated model pipeline.

The typical automated model pipeline in enterprise production environments include 3 types of stores, such as feature store, metadata store and model registry. The feature store contains data extracted from various source systems and transformed into the features as required by the model. The ML pipeline takes the data in batches from the feature store to train the model. The metadata store is a centralized model tracking (bookkeeping) system, maintained at the enterprise level, contains the model metadata at each stage of pipeline. The model metadata store facilitates the model stage transition, say from staging to production to archived. The model training is performed in one environment and deployment in other environments where the model inference will be performed just by specifying the remote model file path. The model metadata store is used for model experiments tracking and compare model experiments w.r.t. its performance. The model metadata includes training data set version, links to training runs and experiments. The model registry contains data of all trained models such as trained model version, model training date, model training metrics, hyperparameters used for training the model, predicted outcomes, and diagnostic charts (confusion matrix, ROC curves). The appropriate model will be picked from the model registry based on the intended target user's requirement. The model metadata store & model registry can be implemented (custom) using a light-weight database, such as SQLite and a set of python functions; or a set of open-source/proprietary products, such as MLflow (community edition/managed service from Databricks). This tool provides an ability to manage models using a GUI or a set of APIs.

Model Deployment: The models can be deployed in production environments to perform prediction either in batch inference mode or on-line inference mode. The batch inference can be achieved by scheduling as a job to run at a time interval and send the results via email to the intended users. The on-line inference can be achieved by exposing the model as a web service using frameworks such as python flask library or streamlit library to develop interactive web applications and invoke the model using its HTTP endpoint.

Model Packaging/Distribution: The trained ML model is serialized into various formats to be able to distribute the model for deployment into TEST/PROD environments. The most common formats are pickle (for machine learning or deep learning models developed in python), ONNX (Open Neural Network Exchange format, for deep learning models), and PMML(Predictive Model Markup Language XML-based format, specifically for models developed using logistic regression and neural networks). The model can be packaged as a .jar file or as a docker container image and deployed as a prediction service into production.

Model Containerization can be achieved by building a docker image, bundled with training and inference code, along with the necessary training and testing data and the model file for future predictions. Once the docker file is created bundled with necessary ML model, a CI/CD pipeline can be built using a tool, such as Jenkins. This docker container image can be exposed as a REST API, so that any external stakeholders can consume this ML model, either from on-premises or public cloud (in case of high compute requirements for building a deep learning model). In case of packaging and managing multiple docker containers, Kubeflow, the ML toolkit for Kubernetes can be used.

Model Performance Monitoring is an important activity where the predicted outcome (e.g., predicted sale price of an item) vs. Actual value (actual sale price) is continuously monitored. And it is recommended to understand the end users' reaction to the final predictions. In some scenarios, it is recommended to keep old model and new model running in-parallel to understand the variation in performance in both the models (model validation). It is mandatory to address performance degradation that arise due to 'data drift' (underlying source data pattern changes due to seasonality and trend, e.g., year-end sales data (or) addition of new categories (or) one particular attribute is not being generated by the upstream systems) and 'model drift' (statistical properties of target variable changes, e.g., definition of a fraudulent transaction itself changes). The most accurate way to measure the model drift is by measuring the F1 Score that combines the precision and the recall of a classifier into a single metric by taking their harmonic mean. The model will be retrained as an when the model drift (F1 Score) falls below certain threshold or at regular intervals (batch mode) or train the model as soon as the data is available (online training). It is very important to collect model logs and prediction logs by using popular logging tools such as elasticstack, and fluentd.

Model version management is a mandatory activity in ML model management as the model will be retrained at regular intervals due to changes in the underlying source data or as demanded by audit and compliance purposes. The source data, the model training scripts, model experiment, and the trained model are versioned together in the code repository. There are open-source tools available, such as Data Version Control (DVC) or AWS CodeCommit that uses Git as underlying code repository for model version management purposes.

ML Project Development Roles: There are various roles involved in implementing ML model development and operations. Data Engineer analyzes the business data from various sources and ensures right and up-to-date data is accessible at the required granularity and quality in cost effective way. Data Scientists look at the data, performs data pre-processing and feature

engineering, model building and choose right model that best fits the predictive/prescriptive requirements of business. Usually, Data Scientists take hypothesis-based approach to choose the model that fits the requirements. ML Engineer makes sure that the built model is giving the results reliably and assess the need for re-training the model by monitoring its outcome. Web App Developer builds the required presentation layer component to invoke and present the results to the intended stakeholders. The ML projects also require DevOps engineers to enable continuous delivery and data visualization experts to produce fancy dashboards.

ML Model Deployment Best Practices: The recommended model deployment best practices are (1) Automate the steps required for ML model development and deployment, by leveraging DevOps tools that gives some extra time for model retraining; (2) Perform continuous model testing, performance monitoring and retraining after its production deployment to keep the model relevant/current as the source data changes to predict the desired outcome(s); (3) Implement logging while exposing ML models as APIs, that includes capturing input features/model output (to track model drift), application context (for debugging production errors), model version (if multiple re-trained models are deployed in production); (4) Manage all the model metadata in a single reposition.