Streamlining Machine Learning Lifecycle to Improve Retail Sales Performance Using MLflow

Sagar Baronia, Shubham Goyanka, Abhishek Jani, Aditya Roy Choudhary, Shirish Shinde, Matthew A. Lanham

Purdue University, Department of Management, 403 W. State Street, West Lafayette, IN 47907

sbaronia@purdue.edu; sgoyanka@purdue.edu; jania@purdue.edu; roychoua@purdue.edu; shinde1@purdue.edu; lanhamm@purdue.edu

Abstract

Organizations leveraging machine learning seek to streamline their machine learning development lifecycle. Machine learning model development possesses additional new challenges as compared to the traditional software development lifecycle. These include but are not limited to tracking experiments, code & data versioning, reproducing results, model management, and deployment. This paper aims to describe the implementation of MLflow, an open-source platform to manage the ML lifecycle, including experimentation, reproducibility, deployment, and a central model registry. MLflow is a rapidly growing open-source community with contributions from industry leaders in data science and integration with extensive machine learning libraries, algorithms, and programming languages. This paper outlines how MLflow on Azure Databricks can streamline the process of building a recommender system to predict the user preference for a product or the likelihood of the user purchasing a product, given they are targeted with coupons in a promotional campaign; and finally, the entire machine learning pipeline integration with Flask using Rest API to serve the model on real-time and batch inferencing.

Keywords: MLOps, MLflow, Rest API, Azure Databricks, Recommender Systems

Streamlining Machine Learning Lifecycle to Improve Retail Sales Performance Using MLflow

Retail is a subset of business where a business sells a product/service to an individual. Retailers can analyze the big data generated through billions of retail transactions and easily influence the customers' buying decisions. The data generated in the retail industry is enormous, leading retailers to leverage big data analytics for price optimization, personalized marketing, and target promotions.

Coupons, discounts, and promotions are essential strategic instruments in the arsenal of retailers (Johnson et al., 2013). Retailers spend billions of dollars on promotions every year to acquire new customers or retain existing customers (Johnson et al., 2013). For example, Catalina Marketing, a distributor of promotions, retrieves about 250 million transactions per week across more than 21,000 grocery stores, analyzes over 100 million households to customize promotions, and earns $400 million in revenue (Johnson et al., 2013).

Understanding what factors affect coupon redemption is crucial to determine the success and opportunity to improve a promotional campaign. This paper discusses how our client can use millions of past transactions and coupon redemption data to improve promotional campaigns efficacy using machine learning techniques like recommender systems.

However, getting a model into the real-world concerns more than building it. Deployment of the model into production is essential to take full advantage of the produced machine learning model. According to a report, 2020 state of enterprise machine learning (Algorithmia, 2020), many organizations haven’t figured out a way to accomplish their machine learning goals because bridging the gap between machine learning model building and deployment is still a challenging task. Although machine learning budgets are on the rise, only 22 percent of companies that use

machine learning have successfully deployed a machine learning model into production. Therefore, there is a need to establish effective practices and processes on designing, building, and deploying machine learning models into production - MLOps. (Dr. Larysa et al., DDD Advisor).

This paper discusses building and streamlining an entire Machine Learning pipeline and deploying the end model using a flask container using MLOps principles. MLflow is used for all machine learning experiment tracking and deployment statistics.

ML FLOW for End-to-End system architecture

The following concepts implemented over ML Flow have been essential to achieving our project's goals.

- **Automatic retrain** – Based on model monitoring, if the accuracy/performance of the model starts decaying, the new data MLflow library should automatically retrain the model and evaluate the performance of the new model based on accuracy/recall/f1 score/precision.
- **Event-based alerts & notifications** – Suppose an old model's performance has decayed and the recent batch prediction accuracy comes below the user-defined threshold value. In that case, the MLflow library should provide a feature to alert the ML ops team to take appropriate actions.
- **Data Drift** – MLflow library should diagnose the significant change in pattern in data. Change in pattern or distribution could be in overall data or a specific feature/column.
- **Model Accuracy, health, and prediction stability** – Evaluation metrics, health, and prediction stability for the assessment of the model should be continuously monitored.
- **Detection of Model Bias** – With time on retraining the model with the new data, the model might create a bias towards data that MLflow should detect.

- **REST API Deployment** – Since the model will be present in the docker to call and predict the outcome for a batch or an individual record, we need to create a REST API.
- **Portable Prediction Server** –MLflow should coordinate among the champion and challengers model. Should be able to split the extensive load from only one champion model to highest performing challengers and subsequently to other challenger models.
- **A/B testing and Multi-Arm Bandit Testing** – MLflow should support A/B testing to evaluate the high-performing model, support the selection of appropriate confidence intervals, the sensitivity of the experiment, and group sampling among control and experiment segments.
- **Real-time prediction** – MLflow should support integration with Flask API to conduct real-time predictions and populate the output on the web page.
- **Batch prediction** – The library should support the prediction of a large amount of data in one go. This records input could be through Panda's data frame, JSON, or a CSV file.
- **Model Registry** – At any instance, if the model is improved through further training and validation over performance metrics, MLflow should assist and make the process of staging the model and eventually productionizing the model seamlessly.
- **Centralized Monitoring Dashboard** – MLflow library should track the notebooks, artifacts, model, and parameter values at the project and deployment version level.
- **Role-based access control** – MLflow should provide access to deploy models, read artifacts, re-train and evaluate new models only to users or groups of users authorized to take respective actions.

Recommender System

Apart from the end-to-end process flow set up on ML Flow, another significant component that proves essential to this project is the team's recommender system to address the problems stated above in the business objectives. Under the various scenarios of the data passed as input into the recommender system, we have developed three recommender models that will return the recommendations to the users.

The second type of model that we have built is Collaborative Filtering recommender systems. This predicts a person's likeliness for selecting a particular item or product from a set of catalogs by connecting the person's recorded interest with the recorded interest of other similar people who share similar features and characteristics based on the reviews, ratings, and feedback they have provided to the existing set of products. The image below highlights the basic differences between a collaborative filtering technique against a Content-Based Filtering Technique.

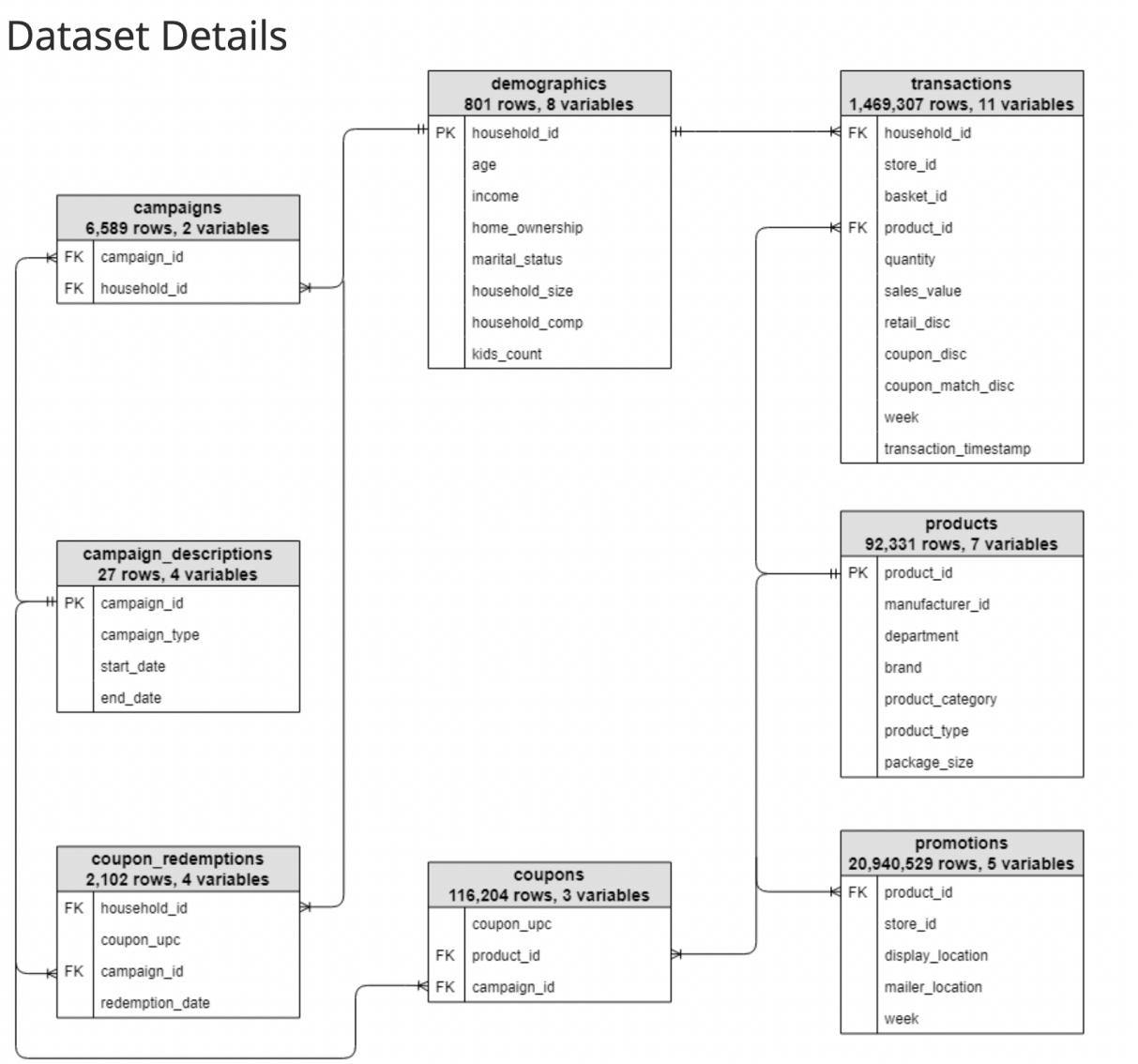
Another kind of recommender system that has been explored to find the solution to our business objectives is using a knowledge-based recommender system that uses knowledge about users and products to pursue a knowledge-based approach to generating a recommendation, reasoning about what products meet the user's requirements. Such a recommender system can prove to be highly sensitive as it uses personalized details about the users to recommend products that they might find helpful. Such details can be captured by carrying out a survey to capture details about the users or gather publicly available demographic data from the census records, as these details can be used to inform choices and generate personalized recommendations.

Data Description

Dataset is by The Complete Journey, which consists of an open-source retail chain dataset shared for the purposes of academic research as it enables us to study the effect of direct marketing

on customers. The Dataset characterizes household-level transactions over one year from a group of 2,469 households who are frequent shoppers at a grocery store. There are eight built-in datasets which consist of the data belonging to the following entities.

- campaigns: Contains identifying information for the marketing campaigns each household participated in
- campaign_descriptions: Contains campaign metadata, specifically the time a campaign was active
- coupons: coupon metadata (UPC code, campaign, etc.)
- coupon_redemptions: Contains individual coupon redemptions (household, day, UPC code, campaign) that can be used to measure campaign efficacy and coupon usage on specific products
- demographics: Contains household demographic data (age, income, family size, etc.). Due to nature of the data, the demographic information is not available for all households.
- products: Contains metadata regarding the products purchased (brand, description, etc.)
- promotions: Contains product placement information (e.g., mailer page placement and in-store display placement) corresponding to campaigns
- transactions: Contains transaction-level product purchases by households (what you would normally see on a receipt)

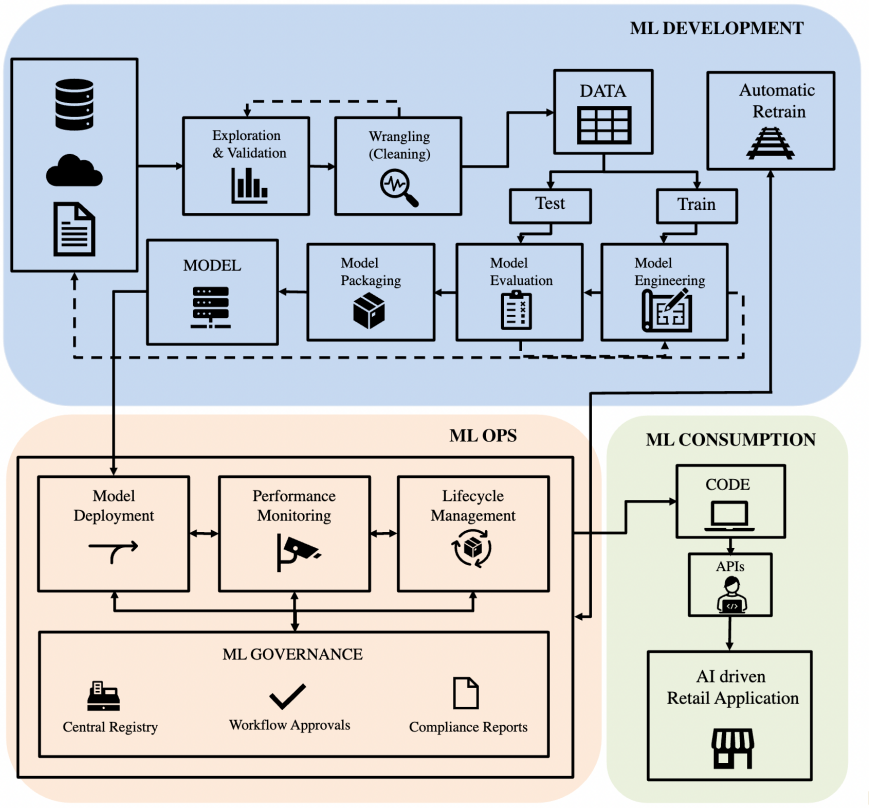


Method

The developed model that predicts the coupon code for an individual customer is based on several features like gender, demography, income bracket, number of members in the family, number of kids in the family, previous transactions with the company, and other relevant features. After developing the model, we deployed it to a flask container and used an open-source library, MLflow, for the entire end to end deployment of the project. This ensured that the accuracy of

predictions is tracked regularly, and when performance begins to lag either due to concept drift or data drift, our model can be retrained from the recent data. Also, one of the project's primary goals is to evaluate the use of MLflow.

The diagram below entails a brief overview of the process flow followed by the team to accomplish the tasks and goals of our project objectives.



The entire architecture can be divided into three parts: ML Development, MLOps, ML Consumption. ML Development includes Data Engineering and Model Engineering.

Data Engineering is the initial step in data science workflow is to acquire and prepare the data to be analyzed. It involves, data ingestion from open-source dataset in form of csv, followed by data exploration, validation using tools like statistical analysis (center tendency, distributions, relationships) and graphical plots (histograms, scatter plots, box plots). Furthermore, Data wrangling is performed to reformat attributes, remove outliers by custom value capping and

dropping null values. Finally, data is split into training, validation, and test datasets in 60:20:20 ratio for training and evaluation of machine learning model.

Model Engineering involves writing and executing machine learning algorithms to obtain an ML model. The collaborative filtering model is trained on training set, hyperparameter tuning is performed and evaluated on validation set using root mean squared error (rmse). Model Acceptance Test is performed on best evaluated model using the hold test dataset. Finally, the model is packaged into MLflow spark model to be consumed by the business applications.

Once we trained a machine learning model, we deploy it as part of a business application such as a mobile or desktop application. The final stage of the ML workflow is the integration of the previously engineered ML model into existing software. Packaged Model is deployed and served on Azure Databricks. Business application built on flask is used to make api calls to databricks server to get inference and log performance metrics.

A basic recommendation model for a new user whose details have not yet been captured by the system to address the cold start problem, a basic recommendation model that returns the details about the most popular products will be returned, based on the frequency of the products being purchased across the retail chain. Simple filtering techniques attempt to present users with items of interest based on a set of pre-defined criteria.

Models

The three models that were deployed in the project are Content-based recommender, Knowledge-based recommender, and item-item Collaborative filtering using implicit feedback. The Knowledge based recommender system was mainly used to address the cold start problem for any new user coming into the retail chain ecosystem or a new product being added into the

inventory. This was modelled by taking into consideration the metadata of the user or the product and then running a multiclass classification to determine the user and product similarity and after the baseline of the models were calibrated with the newly inducted items, the products could then become a part of the collaborative filtering models.

An item-to-item collaborative filtering was performed using the Alternating Least Square (ALS) methodology to find the similarities between the items. The basics of matrix factorization are the cornerstone in achieving the similarity scores using ALS. The matrix factorization takes a large matrix and factors it into smaller representations. The product of these two or more smaller matrices results into the original one. The recommendation engine works upon these factorized matrices by gaining insights from the latent or hidden features using implicit feedback to calculate a similarity measure between a set of items and returns the score. An API call is made to the model with a household_id , product_id pair passed into the model for prediction. The deployed model then calculates the weighted likelihood of purchase for each of the pair and returns the list of score. Recommendations for products are made to the user or the household_id logged into the application based on the weighted score in descending order.

Results

The recommendation system that was developed was trained with multiple parameters to find the best score and the result that would best improve the efficiency of the recommender engine. The following table describes the list of parameters which were considered and tried to determine the best set of results.

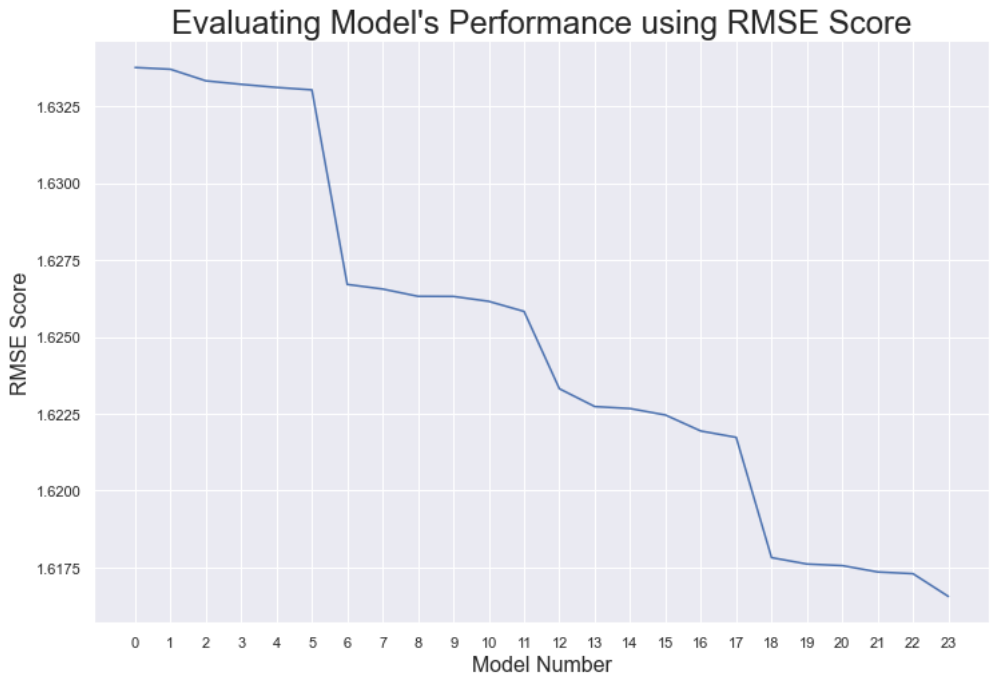
	Latent_Factors	Regularization_Rate	Iterations	Alpha	RMSE
0	40	0.2	100	15	1.616560
1	40	0.1	100	15	1.617299
2	40	0.3	100	15	1.617355
3	40	0.2	50	15	1.617561
4	40	0.1	50	15	1.617612
5	40	0.3	50	15	1.617821
6	40	0.1	100	25	1.621731
7	40	0.3	100	25	1.621938
8	40	0.1	50	25	1.622458
9	40	0.2	100	25	1.622668
10	40	0.2	50	25	1.622733
11	40	0.3	50	25	1.623312
12	20	0.1	100	15	1.625827
13	20	0.2	100	15	1.626155
14	20	0.2	50	15	1.626314
15	20	0.3	100	15	1.626318
16	20	0.1	50	15	1.626555
17	20	0.3	50	15	1.626704
18	20	0.1	100	25	1.633030
19	20	0.1	50	25	1.633109
20	20	0.2	100	25	1.633211
21	20	0.3	100	25	1.633325
22	20	0.2	50	25	1.633700
23	20	0.3	50	25	1.633758

After performing the hyperparameter tuning we were able to get RMSE score of 1.61 for our best model. We performed tuning for various ALS hyperparameters such as latent factor (20,40), regularization rate (0.1,0.2,0.3), n_iteration (50,100) and alpha(15,25). The model with best performance had parameters listed below:latent_factors = 40

regularization_rates = 0.2

n_iterations = 100

alpha_vals = 15



After performing grid-search CV and capturing the best performing parameters the recommendation engine showed improvements in item-item filtering.

Conclusions

Streamlining the development of a recommender system using a hybrid collaborative filtering technique to improve retail sales performance using MLflow is feasible. Out of the many functionalities evaluated, role-based access controls, real-time prediction, batch inferences, and model registry are the top strength of MLflow.

Furthermore, many functionalities like Model Bias detection, Data drift, Concept Drift, and Resource Monitoring can be seamlessly integrated with ML flow using open-source libraries and tools like why logs, shap, Azure monitor, delta lake.

References

1. *Three tiers of non-Customers* (n.d.) from https://www.blueoceanstrategy.com/tools/three-tiers-of-noncustomers/
2. Johnson, J., Tellis, G. J., & Ip, E. H. (2013). *To whom, when, and how much to discount? A constrained optimization of customized temporal discounts.* Journal of Retailing, *89*(4), 361-373.
3. Dr. Larysa Visengeriyeva, Anja Kammer, Isabel Bär, Alexander Kniesz, and Michael Plöd (DDD Advisor). *Why you Might Want to use Machine Learning.* https://ml-ops.org/content/motivation
4. Algorithmia (2020). *2020 state of enterprise machine learning.* https://info.algorithmia.com/hubfs/2019/Whitepapers/The-State-of-Enterprise-ML-2020/Algorithmia_2020_State_of_Enterprise_ML.pdf
5. Wang, T. H., Hu, X., Jin, H., Song, Q., Han, X., & Liu, Z. (2020, September). *AutoRec: An Automated Recommender System.* In Fourteenth ACM Conference on Recommender Systems (pp. 582-584).
6. *MLflow model documentation* (n.d.) from https://mlflow.org/docs/latest/models.html#model-api
7. *The Complete Journey User Guide* (2016) from https://bradleyboehmke.github.io/completejourney/articles/completejourney.html
8. Guéhéneuc, Y. G., & Mustapha, R. (2007). *A simple recommender system for design patterns.* Proceedings of the 1st EuroPLoP Focus Group on Pattern Repositories.

9. Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000, December). *Explaining collaborative filtering recommendations*. In Proceedings of the 2000 ACM conference on Computer supported cooperative work (pp. 241-250).
10. Doshi, S (2019) *Brief on Recommender Systems* https://towardsdatascience.com/brief-on-recommender-systems-b86a1068a4dd
11. Burke, R. (2000). Knowledge-based recommender systems. Encyclopedia of library and information systems, 69(Supplement 32), 175-186