Industrialization of a Data Science use-case using Open Source technologies

Jyotishka DAS
CentraleSupélec
jyotishka.das@student-cs.fr

Xinran YAO
CentraleSupélec
xinran.yao@student-cs.fr

Jiayi WU
CentraleSupélec
jiayi.wu@student-cs.fr

Priyam DASGUPTA
CentraleSupélec
priyam.dasgupta@student-cs.fr

Shamir MOHAMED
CentraleSupélec
shamir.mohamed@student-cs.fr

1 Abstract

Employee churn poses significant challenges to organizations in terms of productivity, knowledge loss, and increased recruitment costs. Predicting and mitigating employee churn has become a critical aspect of human resource management. In this study, we propose and evaluate a predictive model for employee churn using machine learning techniques. The objective of this study was to develop an accurate and robust employee churn prediction model that can assist organizations in identifying employees at risk of leaving. The model utilizes a comprehensive dataset comprising various employee attributes, such as demographic information, job-related factors, performance metrics, and engagement indicators. The IBM HR Analytics Employee Attrition Performance dataset was used for evaluating the model performance. Machine learning algorithms such as Random Forest and Logistic Regression were implemented. The performance of these classifiers was evaluated based on metrics such as accuracy, precision, recall, and F1-score. Additionally, feature importance analysis was conducted to identify the top influential factors contributing to employee churn which would enable organizations to focus their retention strategies on these critical areas.

The developed model can serve as a valuable tool for HR professionals and decision-makers in proactively managing employee retention efforts.However, the utility of the model is minimized unless it can be productionized in a large scale industrial setup. Industrialization involves deploying and scaling it to production environments, which requires reliable and efficient processes for managing and monitoring the ML workflow. These ML pipelines which were established using an open source MLOps platform were ensured to be reliable, efficient, easy to manage, and reusable to any project scenario, thereby cutting future RD costs and increasing efficiency.

2 Project Description

The primary objective of this project is to facilitate the industrialization of a specific use case by establishing a comprehensive machine learning (ML) pipeline. This entails undertaking various tasks, including the development of ML models, orchestration of the pipeline, and implementing model serving capabilities to cater to business requirements. The project will delve into the construction of the pipeline by leveraging a diverse range of cloud technologies, such as MLflow and Airflow, which offer powerful features for managing and deploying ML workflows.

The selected use case for this endeavor revolves around predicting employee churn rate, a crucial aspect that holds significant implications for the human resource department within the organization. By implementing advanced ML techniques, this project aims to generate valuable insights and forecasts, enabling the HR department to proactively address employee attrition and optimize workforce management strategies. Through the integration of open source technologies and the construction of the MLOps pipeline, the project seeks to enhance decision-making processes, foster efficiency, and ultimately contribute to the long-term success of the company.

An implementation of the project is available in the GitHub repository here [1].

3 Value proposition

The values created through this project is two-pronged, one dealing with the business values created through the use-case to correctly predict which employees are likely to resign in the future, and the engineering values created through the creation of a robust and scalable infrastructure that can be used to productionize any industrial level ML projects by the same organisation in the future.

[1]https://github.com/dasjyotishka/Implementing-an-Employee-Churn-Prediction-model-using-open-source-technologies

3.1 Business values

The business values created in the project are:

- **Talent Retention**:Retaining talented employees is crucial for business continuity and success. An employee churn prediction model enables companies to identify factors and patterns associated with employee attrition. By understanding these factors, organizations can take targeted actions to address the underlying issues, improve employee satisfaction, and increase retention rates.
- **Cost Savings**: High employee turnover can be costly for businesses due to recruitment, hiring, and so on. High employee turnover can be costly for businesses due to recruitment, hiring, and onboarding expenses. By accurately predicting employee churn, organizations can proactively identify individuals who are at risk of leaving and take preventive measures to retain them. This can result in significant cost savings by reducing the need for frequent hiring and training.
- **Organizational Stability**: High employee turnover can create instability within an organization, impacting team dynamics, knowledge transfer, and overall morale. By accurately predicting churn, companies can minimize these disruptions and foster a more stable work environment. This stability can contribute to improved collaboration, continuity in projects, and a positive work culture.
- **Enhanced Employee Engagement**: A churn prediction model can uncover insights into the factors that contribute to employee dissatisfaction and disengagement. Armed with this knowledge, companies can implement initiatives to improve employee satisfaction, work-life balance, and overall well-being. By addressing these issues, businesses can foster a more engaged workforce, leading to increased productivity, innovation, and loyalty.

3.2 Engineering values

The enginnering values created in the project are:

- **Reproducibility**: The entire model deployment process was made reproducible. This means that the deployment pipeline should be automated, version-controlled, and easily repeatable. MLFlow can help achieve reproducibility by tracking model versions, experiment parameters, and code versions, hence cutting costs for any organizations.
- **Scalability**: The system was designed to handle a growing number of users and increasing data volumes. Airflow, a workflow management tool, was used to orchestrate the deployment pipeline and schedule model training and retraining tasks. It provides a scalable and distributed architecture that can handle large-scale deployments. Therefore, one-time investment in a scalable infrastructure would lead to the company getting long tern returns when the company grows.
- **Monitoring and Logging**: We have implemented robust monitoring and logging mechanisms to track the performance of the deployed model and detect any issues or anomalies. MLFlow was used to log model metrics and track model performance over time, thus reducing manual interveneces and reducing operational costs for the company.
- **Continuous Integration and Continuous Deployment (CI/CD)**: We have implemented CI/CD practices to ensure that changes to the model or deployment pipeline are tested, validated, and deployed automatically. This involves using tools like Git for version control, setting up automated tests, and deploying new models in a controlled manner. Airflow was used to trigger deployment tasks automatically based on predefined schedules or events.
- **Security and Access Control**: We have established appropriate security measures to protect sensitive data and control access to the deployed system through access controls at the API level. It was important to follow security best practices to prevent unauthorized access or data breaches.
- **Accessibility**: Since the model is hosted in the FlaskAPI and can be accessed using an API, non-tech teams such as HR can leverage the model's predictions without needing technical understandings.
- **Integrations with other systems**: Through the API which was developed, predictions can be integrated into existing software system (eg, HRIS) for comprehensive analysis
- **Real-time analysis**: Through the API which was developed, employee churn can be reduced by responding to changes in employee behavior on the go and take prompt engagement

4 Related Work

Modern churn models frequently draw their foundation from machine learning, more specifically from binary classification methods. There are several of these algorithms; therefore, it is important to test which one works best in each circumstance.

For the sake of benchmarking our model with the latest literatures available, we performed a comparative analysis of various works that have been developed on the prediction of employee attrition using ML techniques. A comparison of existing literature on models used to predict employee churn is shown in Table 1.

Table 1. Comparison of existing literature on models used to predict employee churn

Year	Model	Authors	Accuracy
2022	Support vector machines (SVMs)	Naz et al. [1]	0.775
	Decision Tree		0.975
	Naïve Bayes		0.975
	Random Decision Forest		0.98
	Logistic Regression		0.765
2021	Logistic Regression	Srivastava and Eachempati [2]	0.88
	Gradient Boosting Algorithm		0.852
2021	SVMs	Bandyopadhyay and Jadhav [3]	0.604
	Naive Bayesian		0.688
	Random Forest		0.708
2020	Naive Bayesian	Ayşe Bat [4]	0.966
	Decision tree		0.887
	Random Forest Classifier		0.86
2017	Naive Bayesian Classifier	Sisodia et al. [5]	0.7918
	Decision Tree classifier		0.9768
	Random Forest		0.9897
	K-nearest neighbor		0.9600
2016	SVMs	Punnoose and Ajit [6]	0.84
	Random Forest		0.53
	SVM (RBF kernel)		0.68
2013	C5 Decision tree	Alao et al. [7]	0.74
	REPTree		0.62
	CART		0.64
2006	Deep Neural Networks	Saradhi et al. [8]	0.912
	Random Forest		0.823
	Naive Bayesian		0.55
2018	XGBoost	Alamsyah et al. [9]	0.88
	Random Forest (Depth controlled)		0.79
	Random Forest		0.975

5 Methodology

The core-methodology for developing a model for predicting the employee attrition results consisted of selecting a proper dataset and then pre-process it to make it suitable to use it for training a ML model. Then, the performance indices of various models on the same dataset was monitored, and the model that gave the best performance indices was selected for continuing with industrialising it with scalable MLOps pipeline.

5.1 Dataset

The IBM HR Analytics Employee Attrition Performance dataset was used which contains employee data for 1,470 employees with various information about the employees. It was used to predict when employees are going to quit by understanding the main drivers of employee churn. There are 35 factors : *Age, Attrition, Business Travel, Daily Rate, Department, Distance From Home, Education, Education Field, Employee Count, Employee Number, Environment Satisfaction, Gender, Hourly Rate, Job Involvement, Job Level, Job Role, Job Satisfaction, Marital Status, Monthly Income, Monthly Rate, Number of Companies Worked, Is Over 18?, Is OverTime?, Percent Salary Hike, Performance Rating, Relationship Satisfaction, Standard Hours, Stock Option Level, Total Working Years, Training Times Last Year, Work-Life Balance, Years At Company, Years In Current Role, Years Since Last Promotion, Years With Current Manager.*

These attributes represent various factors related to employees in the dataset, such as their age, job details, work environment, and personal information. They play a crucial role in analyzing and predicting employee attrition rates and understanding the factors that contribute to employee satisfaction and engagement.

As stated on the IBM website "This is a fictional data set created by IBM data scientists". Its main purpose was to demonstrate the IBMWatson Analytics tool for employee attrition.

5.2 Data Preprocessing

The analysis begins with data preprocessing, which involved cleaning and formatting the dataset and converting the categorical variables into dummy variables. Machine Learning algorithms can typically only have numerical values as their predictor variables. Hence Label Encoding becomes necessary as they encode categorical labels with numerical values. To avoid introducing feature importance for categorical features with large numbers of unique values, Label Encoding and One-Hot Encoding were used.

The dataset used in this study has no missing values. In HR Analytics, it is unlikely for employee data to feature a large ratio of missing values since HR Departments typically have all personal and employment data on-file. However, the type of documentation data is being kept in (i.e. whether it is paper-based, Excel spreadsheets, databases, etc) has a massive impact on the accuracy and the ease of access to the HR data.

Feature scaling is an essential step in data preprocessing when developing predictive models. It involves transforming the numerical features in the dataset to a common scale or range. The main purpose of feature scaling is to ensure that all features contribute equally to the model training process, preventing any single feature from dominating the model's learning process due to differences in their original scales.

When features have significantly different scales, it becomes challenging to compare the importance or weight of each feature. Scaling the features to a common range enables fair comparison and interpretation of their contributions to the model's predictions.In this case the MinMaxScaler was used to essentially shrink the range of values such that it is now between 0 and 5.

5.3 Model Selection and Experimental Evaluation

In this study, we employed 10-fold cross-validation to evaluate the performance of the various models used for employee churn prediction. Cross-validation is a robust technique commonly used in machine learning to estimate the model's performance on unseen data and assess its generalization capabilities. The 10-fold cross-validation procedure involves splitting the dataset into ten equal-sized subsets or "folds." The models are then trained and evaluated ten times, each time using a different fold as the validation set and the remaining nine folds as the training set. This process ensures that each data point is used for both training and validation, providing a comprehensive assessment of the model's performance across different subsets of the data.

By performing cross-validation, we mitigate the risk of overfitting or underfitting the model to a specific subset of the data. It helps us understand how the model generalizes to unseen instances and provides a more robust estimate of its performance.

The average performance metrics obtained from the 10-fold cross-validation provide a more reliable estimate of the model's performance than a single train-test split. Additionally, the standard deviation of the performance metrics across the folds provides insights into the consistency and stability of the model's predictions.

By utilizing 10-fold cross-validation, we ensure a rigorous evaluation of the models, capturing their performance across multiple subsets of the data. This approach enables us to make informed comparisons between different classifiers and select the model that exhibits the best overall performance in predicting employee churn.

A number of algorithms were tested and their performances were compared. The accuracy for each algorithm is shown below :

Algorithm	Accuracy
Random Forest	**85.12**
K - Nearest Neighbour	84.67
Support Vector Machines	84.30
Decision Tree Classifier	80.31
Logistic Regression	76.51
Gaussian Naïve Bayes Classifier	66.33

It was observed that the Random Forest Classifier provided the highest accuracy of 85.12. Since it aggregates predictions from multiple decision trees, the impact of individual noisy or outlier instances is reduced. The ensemble nature of Random Forest helps to smooth out the effects of individual data points, resulting in more accurate predictions.

Additionally, Random Forest can handle imbalanced datasets effectively. It assigns weights to each class during the training process, giving equal importance to both the majority and minority classes. This prevents the model from being biased towards the dominant class and allows it to learn meaningful patterns from the minority class, which is crucial in employee churn prediction where churned employees are often in the minority.

Our model could also identify the **top five factors** affecting employee attrition. These top five factors are **Overtime, Monthly income, Age, Total working years** and **Daily rate**.

5.4 Creating scalable pipelines using MLOps

MLOps, which stands for Machine Learning Operations, refers to the practices and techniques used to streamline and operationalize machine learning models in production environments that focus on the challenges of deploying, monitoring, and maintaining machine learning systems. It helps establish a robust and efficient workflow that encompasses the entire lifecycle of a machine learning project.

Secondly, MLOps focuses on building scalable and automated pipelines for data preprocessing, model training, and evaluation. These pipelines enable efficient data ingestion, feature engineering, model training, hyperparameter tuning, and evaluation. Automation helps reduce manual errors, enhances productivity, and enables rapid experimentation and iteration. Another critical aspect of MLOps is model deployment and serving. It involves packaging trained models into deployable formats, setting up scalable and robust infrastructure for serving predictions, and monitoring the model's performance in real-world scenarios. In this specific use case, MLOps technologies like Airflow and MLflow were used to provide a framework for managing the entire ML workflow, from data preparation to model training, evaluation, and deployment. An entire overview of the scalable MLOps pipelines that was created for this project is shown in Figure 1. The red-dotted lines show the pipelines which were not yet created for this project but can be taken up in the future for more versatility.

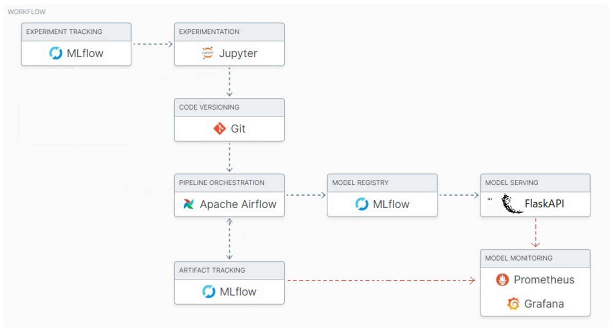


Figure 1. An entire overview of the scalable MLOps pipelines that was created for this project

- **Airflow**: Airflow is an open-source platform for orchestrating and managing complex workflows or pipelines.

It provides a robust framework that allows users to define, schedule, and monitor tasks within a workflow, enabling efficient and reliable data processing and analysis.
Pipeline orchestration is the process of managing and coordinating complex workflows or pipelines in a systematic and automated manner. It involves defining tasks, their dependencies, and scheduling their execution to achieve efficient data processing and analysis. In this case, we are utilizing pipeline orchestration to streamline the workflow of our data processing and modeling tasks.
One of the main reasons for choosing Airflow as our pipeline orchestration tool is its flexibility and scalability. Airflow utilizes a directed acyclic graph (DAG) structure to represent workflows, allowing users to define tasks and their dependencies in a highly configurable and intuitive manner. This flexibility enables us to design and customize our workflows to suit our specific business requirements.

- **MLflow**: MLflow is an open-source platform for managing the complete lifecycle of machine learning models, which provides functionalities for tracking, packaging, and deploying models, making it easier to serve models in production environments. With MLflow, models can be easily saved, uploaded to a server, and served through REST API endpoints. This simplifies the process of exposing machine learning models as web services, allowing easy integration with other applications or systems.

5.4.1 Environment Setup in a Linux environment.

Given that cloud technologies are primarily tailored for Linux systems, we have opted to employ the Windows Subsystem for Linux (WSL) as a practical solution for our workflow. By utilizing WSL, we benefit from simplified operations and enhanced compatibility with the cloud technologies we intend to leverage in our project. This choice ensures a seamless integration of our workflows with the desired cloud technologies, offering a smoother and more efficient working environment.

After installing and enabling WSL, we proceeded to select and install a Linux distribution of our choice, opting for Ubuntu. Subsequently, we launched the Linux distribution and executed essential updates and package installations such as Python, pip, and venv to ensure a robust foundation. To facilitate Airflow's functionality, we installed the necessary dependencies according to the instructions from the Airflow official documentation and created a virtual environment to encapsulate Airflow's Python packages.

The virtual environment is used in order to ensure that the required Python packages and their dependencies are isolated and don't conflict with other packages installed on our systems, and to make sure we can easily reproduce the exact package versions used in this Airflow project so that it can be easily shared or deployed on different systems.

5.4.2 Pipeline Construction using Airflow.

After setting up the environment, we constructed the Airflow pipeline from a Python notebook that specifically designed to address the task of predicting employee attrition.

- **Creating DAG** The DAG(Directed Acyclic Graph) is the fundamental structure in Airflow that represents the workflow or pipeline. It defines the tasks and their dependencies, allowing for the scheduling and execution of the tasks in a specific order. Here we create a new DAG named "employee_attrition_dag_2", which is the unique identifier for the DAG specified by the **dag_id** parameter.

- **Creating Tasks** The pipeline comprises several interconnected tasks that operate sequentially to accomplish the overall objective. The tasks are defined using the PythonOperator and DummyOperator classes from Airflow. The PythonOperator is used to execute Python functions as tasks, while the DummyOperator is a simple operator used as a placeholder or marker within the pipeline. The tasks are represented by the following functions that we wrapped from the original codes so that each function can represent a distinct step or operation within the pipeline, allowing for better organization and reusability of code.

  - **read-data** the pipeline reads the employee attrition data from a specified file path and performs initial data analysis. The data is loaded into a DataFrame, and relevant insights about the dataset's shape and columns are extracted. This task serves as the starting point for subsequent data preprocessing and modeling steps.
  - **prep-encoding** focuses on preparing the data for modeling. It applies label encoding to columns with two or fewer unique values, transforming them into numeric representations. Additionally, categorical variables are converted into dummy variables, enhancing their usability in subsequent modeling stages. The modified Data
    Frame is stored in the XCom storage, ensuring data continuity across tasks.
  - **prep-scale** Here, the numerical features of the DataFrame are scaled using the MinMaxScaler from the sklearn library. By rescaling the features to a specified range (0 to 5 in this case), the task aims to achieve consistent scaling across different variables, ensuring that

their magnitudes do not disproportionately impact the modeling process. The modified DataFrame is again stored in the XCom storage for further processing.

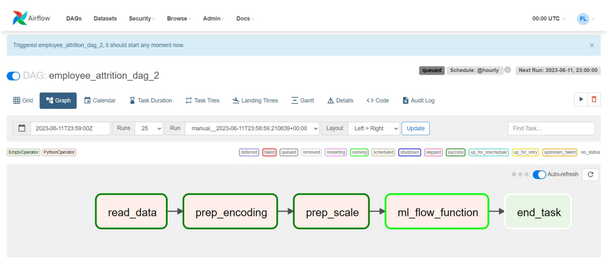
– **ml-flow-function** Once the data is preprocessed and scaled, the pipeline moves to the "ml-flow-function" task, which entails the machine learning modeling stage. In this task, a Random Forest Classifier is trained using the preprocessed data. The model's performance is evaluated by making predictions on a holdout dataset, and metrics such as accuracy and F1 score are calculated. Importantly, the task leverages the MLflow library to log these metrics, allowing for easy tracking and comparison of different model iterations. Furthermore, the trained model itself is logged as an artifact, making it readily accessible for deployment or future reference.

Finally, the pipeline concludes with an "end-task" that serves as a placeholder, denoting the completion of the pipeline execution.

- **Define task dependencies** Task dependencies define the order in which tasks should be executed and allow for the coordination and synchronization of tasks within the pipeline.In our project, the task dependencies are defined using the » operator, which signifies that one task is dependent on the completion of another task. A screenshot showing the task dependencies of the various tasks created for this project is shown in Figure 2. A green border around the task shows that the task has been completed and a red border around the task shows that the task has failed to be executed. Thus, any business user can go to the Airflow server page, search for the DAG, and trigger it to be executed without a single piece of code. The Airflow interface not only enables any user to check the status of every sub-task involved in the project, but he can also monitor the status and execution time of different tasks going on in parallel, thus providing an easy interface for monotoring different ML processes going on in a large-scale industrial production environment.

  – **read-data-task** » **prep-encoding-task**: This dependency indicates that the **prep-encoding-task** should be executed only after the **read-data-task** is completed. The output of the **read-data-task** is stored in the XCom storage, and the **prep-encoding-task** retrieves this data as input.
  – **prep-encoding-task** » **prep-scale-task**: The **prep-scale-task** is dependent on the completion of the **prep-encoding-task**. It retrieves the modified DataFrame from the XCom storage, which was stored by the **prep-encoding-task**, and performs further data scaling operations.
  – **prep-scale-task** » **ml-flow-function-task**: The **ml-flow-function-task** depends on the completion of the **prep-scale-task**. It retrieves the scaled DataFrame from the XCom storage, which was stored by the **prep-scale-task**, and proceeds with the machine learning modeling stage.
  – **ml-flow-function-task** » **end-task**: The **end-task** has a dependency on the completion of the **ml-flow-function-task**. It serves as the final task in the pipeline, indicating the completion of the pipeline execution.

  By defining these task dependencies, Airflow ensures that the tasks are executed in the specified order, with each task relying on the successful completion of its preceding task. This enables the pipeline to follow a well-defined sequence, ensuring that data flows seamlessly from one task to another and that each task operates on the correct input data.



Triggered employee_attrition_dag_2, it should start any moment now.

DAG: employee_attrition_dag_2

queued Schedule: @hourly Next Run: 2023-06-11, 23:00:00

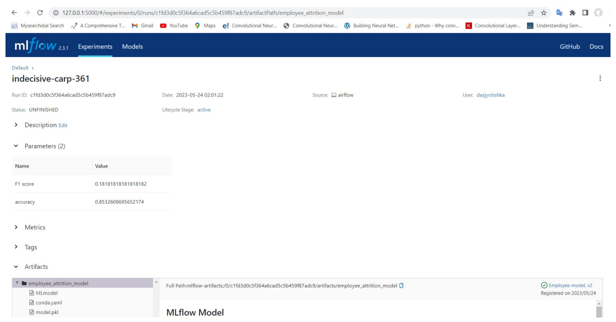
2023-06-11T23:59:00Z Runs 25 Run manual__2023-06-11T23:58:59.210639+00:00 Layout Left > Right

Figure 2. The dependencies and statuses of the Airflow Directed Acyclic Graph (DAG) for the project

- **Task Scheduling** Final step is to set the start_date and schedule_interval attributes of the DAG to determine when the pipeline should begin and the frequency at which it should run.
  – **start_date**: The **start_date** parameter indicates the date and time from which the DAG should start its execution. In the code, it is set to May 9, 2023, using the **datetime** module.
  – **schedule_interval**: The **schedule_interval** parameter determines the frequency at which the DAG should run. In this case, the "@hourly" value indicates that the DAG should be executed every hour.
  – **catchup**: The **catchup** parameter determines whether the DAG should catch up on missed execution dates or only run tasks based on the schedule going forward. In the code, it is set to **False**, indicating that the DAG should not catch up on missed runs.

5.4.3 Model Tracking and Registry using MLflow.

Once the model training step in the Airflow pipeline is completed, the trained model needs to be saved in a deployable format. This can be done by utilizing MLflow's model logging functionality. Within the pipeline, after the model training task, the trained model can be saved using the mlflow.sklearn.log-model() function provided by MLflow. This function saves the model along with its associated metadata and dependencies. After that, we register the model using MLflow Model Registry, which assigns a version number and creates a model entry in the registry. This step helps us to keep track of different versions of our model. A screenshot of the MLFlow server showing the model getting registered along with its run-id and performance indices is shown in Figure 3.



127.0.0.1:5000/#/experiments/0/runs/c1fd3d0c5f364a6cad5c5b459f87adc9/artifactPath/employee_attrition_model

indecisive-carp-361

Run ID: c1fd3d0c5f364a6cad5c5b459f87adc9 Date: 2023-05-24 02:01:22 Source: airflow User: dasjyotishka

Status: UNFINISHED Lifecycle Stage: active

Parameters (2)

Name	Value
F1 score	0.18181818181818182
accuracy	0.8532608695652174

employee_attrition_model

Full Path:mlflow-artifacts:/0/c1fd3d0c5f364a6cad5c5b459f87adc9/artifacts/employee_attrition_model

Employee-model, v2
Registered on 2023/05/24

MLflow Model

Figure 3. A screenshot of the MLFlow server showing the model getting registered along with its run-id and performance indices

5.5 Model Serving using FlaskAPI

API (Application Programming Interface) is a set of protocols and tools that allow different software applications to communicate and interact with each other. An API allows external applications or systems to interact with your model, sending input data and receiving the model's predictions as output. In this use case, we will need to create an API so that the model can be accessible by HR department for real-time predictions.

Among all the choices for creating API, MLflow has its inbuilt REST API, which is a specific type of API provided by it. It uses HTTP methods such as GET, POST, PUT, and DELETE to perform different operations on MLflow resources. After logged and saved the model, when we access the MLflow server, the endpoints of MLflow REST API are automatically generated by MLflow based on the MLflow server configuration.Once the MLflow server is up and running, we can make a POST request to the appropriate MLflow server endpoint from the Linux command window. Now the model can be served through the MLflow REST API, specifying the new input data for prediction.

However, the MLflow API itself is not designed to receive HTTP requests directly from external clients like Postman. It is primarily used as a backend service for tracking and managing machine learning experiments.

To leverage the MLflow API for making predictions with a CSV file, we would typically build a separate application or script that interacts with the MLflow server. This application would handle the HTTP requests from external clients, load the model saved in the MLflow server return the predictions as a response to the web-client. After constructing the pipeline in Airflow and training the machine learning model, and registering the model in MLFlow, the next step is to set up model serving using FlaskAPI.

Flask is a popular web framework in Python that is commonly used for building web applications, including model serving applications. Through a Flask application, a client such as HR can send requests to our model and get the output in return. The logic flow to achieve this is as follows: Flask Application -> MLFlow server -> MLFlow API called internally -> Results obtained -> Response delivered to Flask Application.

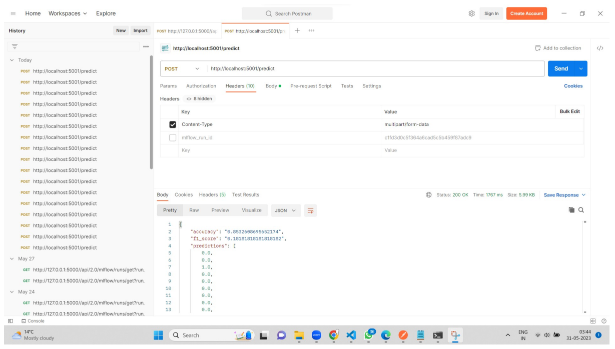
The process begins when a client sends a request to the Flask application, which containing test csv data. Upon receiving the request, the Flask application interacts with the MLflow server. The MLflow server is responsible for managing the trained models and their associated artifacts.

Within the Flask application, an internal call to the MLflow API is made to fetch the desired model. Then the MLflow server processes the API request and retrieves the requested model along with its associated artifacts.

Once the MLflow server has obtained the model, it performs the necessary computations or predictions on the input data from the Flask application. The resulting output or prediction is then returned as a response.

To allow interaction for clients, we defined a Flask application that handles the POST request at the '/predict' route. The make_predictions() function is responsible for reading the CSV file, performing data preprocessing and model prediction using MLflow, and returning the predictions as a JSON response. The handle_predict() function is the entry point for the POST request and calls make_predictions() to process the request.

Postman is an API Platform for developers to design, build, test and iterate their APIs. Test results as obtained from the response from the Flask app when a request is send through API via Postman web client is shown in Figure 4. Since everything is perfect, in the response field, we can see a binary value corresponding to the attrition prediction results of every employees, along with the accuracy and F1 score of the model fetched from MLFlow registry.



```
POST http://localhost:5001/predict
GET http://127.0.0.1:5000//api/2.0/mlflow/runs/get?run_
```

POST http://localhost:5001/predict

	Key	Value
✓	Content-Type	multipart/form-data
	mlflow_run_id	c1fd3d0c5f364a6cad5c5b459f87adc9
	Key	Value

Status: 200 OK Time: 1767 ms Size: 5.99 KB

```
{
    "accuracy": "0.8532608695652174",
    "f1_score": "0.18181818181818182",
    "predictions": [
        0.0,
        0.0,
        1.0,
        0.0,
        0.0,
        0.0,
        0.0,
        0.0,
        0.0,
```

Figure 4. Response structure for the employee attrition model fetched from the FlaskAPI using a Postman web-client for verification

6 Business Recommendations

We recommend the stakeholders to strategize on the two-pronged value-creation methods employed in this project through the business aspects and engineering aspects to create value in their organisations in the long run.

Our model could also identify the top five factors affecting employee attrition. These top five factors, along with our proposal for the organisation to mitigate the risks of attrition in the future are as follows:

- **Overtime**: Implementing policies that limit excessive overtime and providing adequate support to employees can help reduce burnout and increase job satisfaction.
- **Monthly income**: The level of monthly income has a direct impact on attrition. It is crucial for companies to ensure that their compensation packages are competitive and aligned with industry standards.
- **Age**: Older workers generally exhibit higher levels of retention due to their tendency to remain with the same organization for longer periods. Factors such as job stability, experience, and a sense of loyalty often contribute to their reduced attrition rates.
- **Total working years**:Creating long-term career development plans and providing growth opportunities for employees, mentorship programs, training initiatives, and promotions based on performance can encourage employees to stay committed to the organization.
- **Daily Rate**: Regular benchmarking exercises, salary adjustments, and rewards for high-performing employees can help in reducing attrition related to inadequate pay.

A Technical Appendix

The steps folowed to create all the scalable pipelines used in the project are mentioned below:

1. **Download Windows Subsystem for Linux (WSL)**: Follow the steps as given in the URL https://www.freecodecamp.org/news/how-to-install-wsl2-windows-subsystem-for-linux-2-on-windows-10/
2. **Install Anaconda in the Linux subsystem**: Follow the steps as given in the URL https://www.how2shout.com/how-to/install-anaconda-wsl-windows-10-ubuntu-linux-app.html
3. **Install other packages**: Follow the steps as given in the URL https://learn.microsoft.com/en-us/windows/python/web-frameworks
4. **Install MLFlow on WSL**: Follow the steps as given in the URL https://www.adaltas.com/en/2020/03/23/mlflow-open-source-ml-platform-tutorial/
5. **Install Airflow on WSL**: Follow the steps as given in the URL https://www.freecodecamp.org/news/install-apache-airflow-on-windows-without-docker/
6. **Execute the DAG to train and save the model**: Copy the main DAG code "implementing-employee-churn-model.py" to the DAG folder of the Airflow directory and start the Airflow server on http://localhost:8080/home. Now go to the Airflow UI of the server, and execute the DAG. The saved model would be registered in the MLFlow registry. The model can be registered through MLFlow UI on http://127.0.0.1:5000/.
7. **Serve the model through Flask API**: Execute the python code "python_script_Flask_Testing.py" to serve the model through Flask API. To check whether the API is working, Postman can be used. In Postman, create a POST request on the address http://localhost:5001/predict. Under Headers, create a field "Content-Type" with value "multipart/form-data". Now create another field called "mlflow_run_id" with the value of the run-id of the saved MLFlow model that we want to use. Under body, in the src field, attach the csv file of the test-dataset whose output we want to see. Click on the "Send" button. If everything is perfect, then in the response field, we

can see a binary value corresponding to the attrition prediction results of every employees, along with the accuracy and F1 score of the model fetched from MLFlow registry

B Bibliography

[1] Naz, K., Siddiqui, I.F., Koo, J., Khan, M.A. and Qureshi, N.M.F., 2022. Predictive Modeling of Employee Churn Analysis for IoT-Enabled Software Industry. Applied Sciences, 12(20), p.10495.

[2] Srivastava, P.R. and Eachempati, P., 2021. Intelligent employee retention system for attrition rate analysis and churn prediction: An ensemble machine learning and multi-criteria decision-making approach. Journal of Global Information Management (JGIM), 29(6), pp.1-29.

[3] Jadhav, A., 2021. Churn prediction of employees using machine learning techniques. Tehnički glasnik, 15(1), pp.51-59.

[4] Ayşe Bat, 2020. Employee Churn Analysis. Medium. Accessed on 12/06/2023

[5] Sisodia, D.S., Vishwakarma, S. and Pujahari, A., 2017, November. Evaluation of machine learning models for employee churn prediction. In 2017 international conference on inventive computing and informatics (icici) (pp. 1016-1020). IEEE.

[6] JAjit, P., 2016. Prediction of employee turnover in organizations using machine learning algorithms. algorithms, 4(5), p.C5.

[7] Alao, D.A.B.A. and Adeyemo, A.B., 2013. Analyzing employee attrition using decision tree algorithms. Computing, Information Systems, Development Informatics and Allied Research Journal, 4(1), pp.17-28.

[8] Saradhi, V.V. and Palshikar, G.K., 2011. Employee churn prediction. Expert Systems with Applications, 38(3), pp.1999-2006.

[9] Alamsyah, A. and Salma, N., 2018, August. A comparative study of employee churn prediction model. In 2018 4th International Conference on Science and Technology (ICST) (pp. 1-4). IEEE.