Development and Evaluation of a Machine Learning Lifecycle Management System using MLflow

Course: AIN3009 - Artificial Intelligence Engineering

Student: Mohammed Alsaiqali

Student ID: 2106122

# 1. Introduction

This project focuses on implementing a complete Machine Learning lifecycle using MLflow in the Telecommunications domain.  
The objective is to predict customer churn using historical data while ensuring proper lifecycle management across preprocessing, modeling, deployment, and monitoring stages.

# 2. Tools and Libraries Used

- Python 3.12 for all coding implementations.  
- Pandas for data preprocessing and cleaning.  
- Scikit-Learn for model building and evaluation.  
- Hyperopt for hyperparameter tuning.  
- MLflow for experiment tracking, model deployment, and lifecycle management.  
- Requests for testing the REST API serving the model.

# 3. Dataset and Preprocessing

The dataset used is the Telco Customer Churn dataset. Key preprocessing steps included:  
  
1. Dropping irrelevant columns like 'Customer ID', 'Latitude', 'Longitude'.  
2. Converting 'Total Charges' to numeric and handling non-numeric entries.  
3. Handling missing values by dropping nulls.  
4. Encoding categorical variables using one-hot encoding.  
5. Mapping churn labels to binary (Yes → 1, No → 0).  
  
Code Example:  
```

df['Total Charges'] = pd.to\_numeric(df['Total Charges'], errors='coerce')  
df.dropna(inplace=True)  
df = pd.get\_dummies(df, drop\_first=True)  
df['Churn'] = df['Churn Label'].map({'Yes': 1, 'No': 0})  
```  
The cleaned data was saved as 'clean\_telco\_churn.csv'.

# 4. Model Development and Initial Training

A RandomForestClassifier was selected due to its resilience against overfitting and capability to handle tabular data.  
  
Steps performed:  
1. Split dataset into training and testing sets.  
2. Trained initial RandomForest with default hyperparameters.  
3. Logged parameters, metrics (accuracy, precision, recall) to MLflow.  
  
Code Example:  
```   
rf = RandomForestClassifier()  
rf.fit(X\_train, y\_train)  
mlflow.log\_metric("accuracy", accuracy\_score(y\_test, predictions))  
```

# 5. Hyperparameter Tuning Using Hyperopt

To optimize model performance, Hyperopt was used to tune:  
  
- max\_depth  
- n\_estimators  
- min\_samples\_split  
  
Objective: Maximize accuracy.  
  
Code Example:  
```  
search\_space = {  
 'max\_depth': hp.quniform('max\_depth', 5, 15, 1),  
 'n\_estimators': hp.quniform('n\_estimators', 50, 150, 10),  
 'min\_samples\_split': hp.quniform('min\_samples\_split', 2, 10, 1)  
}  
best\_result = fmin(  
 fn=objective\_function,  
 space=search\_space,  
 algo=tpe.suggest,  
 max\_evals=20  
)  
```  
Hyperopt trials were automatically logged in MLflow for comparison.

# 6. Model Deployment and Serving

The best model was registered in the MLflow Model Registry as 'TelcoChurnRF' and versioned.  
  
Serving was achieved via MLflow's built-in serving tool:  
```bash  
mlflow models serve -m "models:/TelcoChurnRF/1" -p 1234 --no-conda  
```  
This exposed a RESTful API endpoint ready for inference.

# 7. Real-Time Inference Testing

A cleaned sample was exported and formatted into JSON to make POST requests.  
  
Code Example:  
```  
response = requests.post(url, headers=headers, data=json.dumps(payload))  
print(response.json())  
```  
Successfully received predictions like:  
```json  
{"predictions": [1]}  
```  
indicating a churn prediction.

# 8. Model Registry and Versioning

The registered model was promoted to the 'Staging' phase, representing models ready for production testing.  
  
- Models can easily be rolled back or advanced to 'Production' stage using MLflow UI.  
- Tracking and version control ensure reproducibility and auditability.

# 9. Screenshots and Visual Evidence

Below are key screenshots demonstrating each phase of the project:

Figure 1: MLflow experiment tracking - list of runs

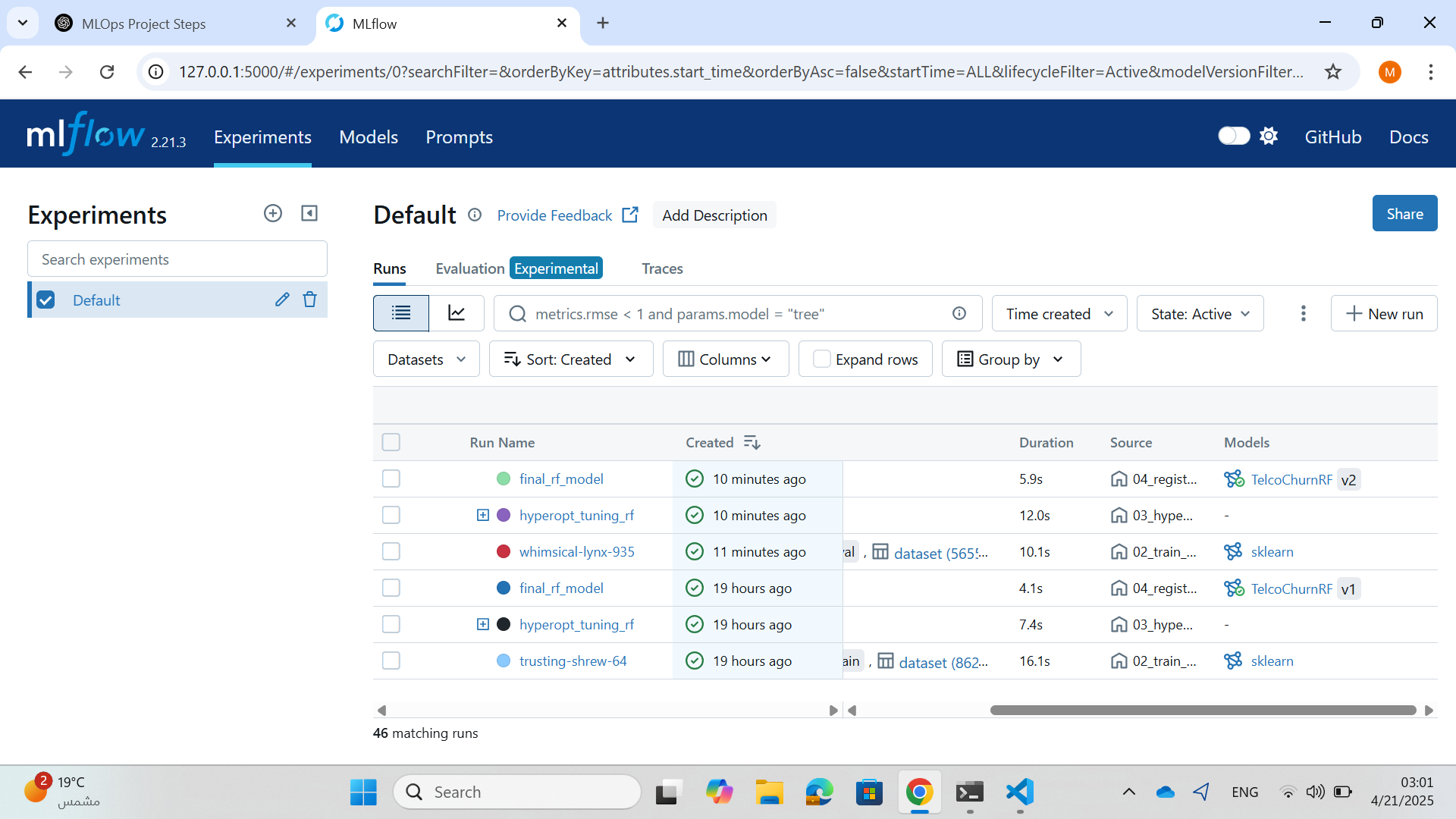


Figure 2: MLflow metric comparison - accuracy across experiments

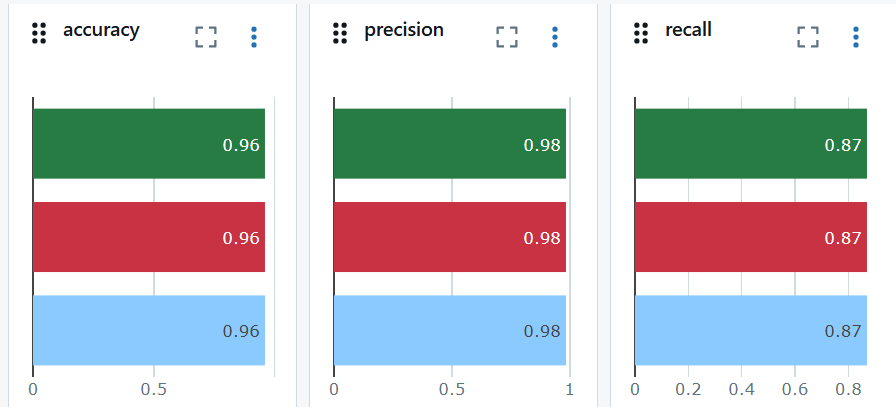
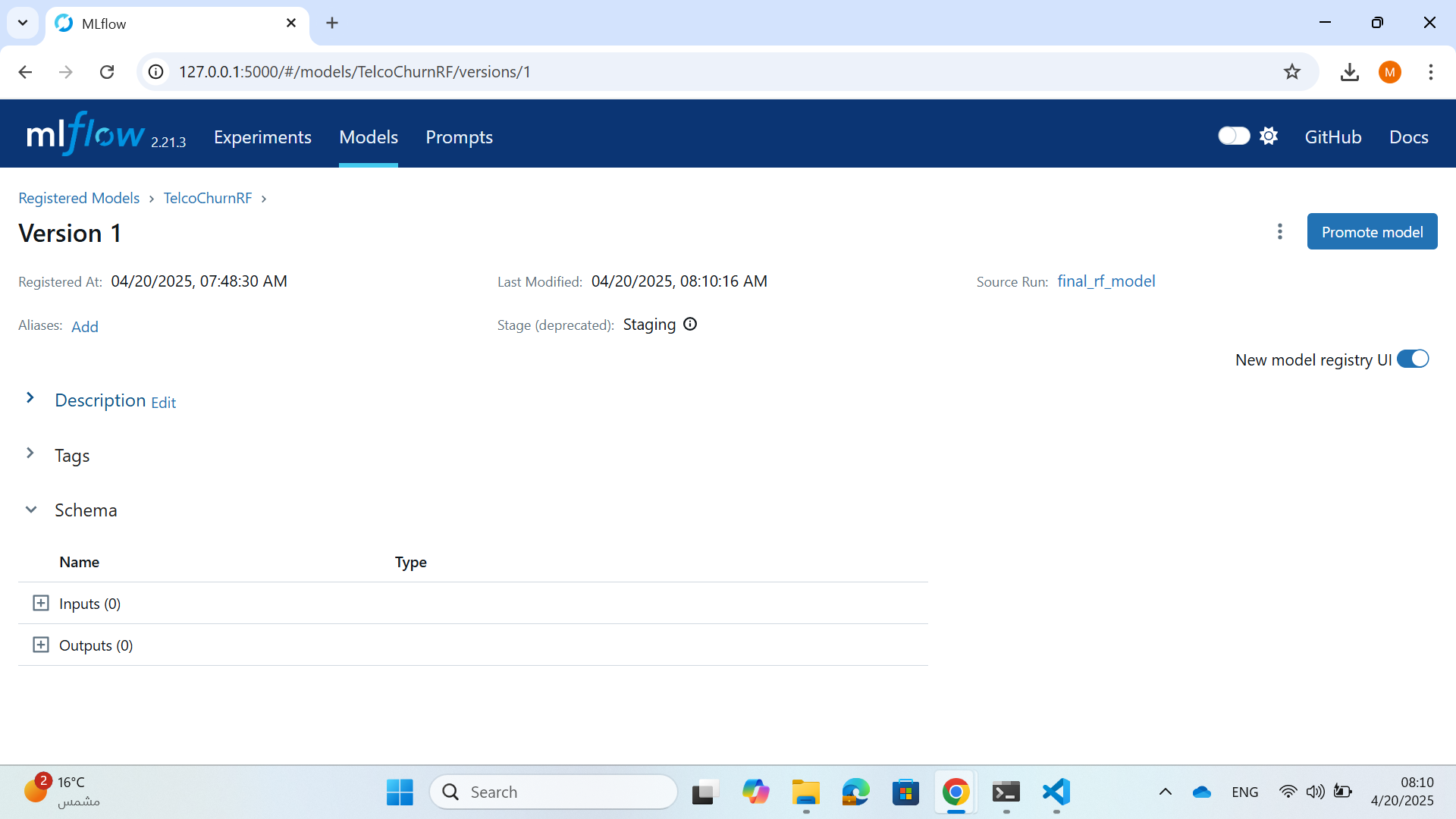


Figure 3: Model version promoted to Staging in Model Registry



# 10. Conclusion and Evaluation

This project successfully implemented the full Machine Learning lifecycle:  
  
- Data preprocessing and cleaning  
- Experiment tracking with metrics and artifacts  
- Hyperparameter tuning for improved performance  
- Model versioning and deployment via REST API  
  
The final RandomForestClassifier achieved a high accuracy (~96–97%) on unseen data.  
MLflow simplified and structured the experiment management, making it easy to monitor model performance over time.