Productionising Telco Churn Rate Prediction Model

Nathan Murstein, Hisham Salem, Dany Stefan, Jeewon Kim, Uzair Ahmad, Reza Soleimani, Danyal Hamid

Team Composition (github username)

1. **Nathan** Murstein (NathanMurstein) - Data Scientist
2. **Hisham** Salem (HishamSalem) - Data Analyst
3. **Reza** Soleimani (rsoleimani) - Lead Data Scientist
4. **Dany** Stefan (dany-stefan) - MLOps Engineer
5. **Danyal** Hamid (danyalhamid1996) - Machine Learning Engineer
6. **Jeewon** Kim (jeewonk) - Business Analyst
7. **Uzair** Ahmad (uzairahmadxy) - Project Manager

Github Repository - MMA4_ChurnPrediction

Context and Scope of the project

Goal: To productionize ML model to generate repeatable experiments and manage model cycle

What we've done

Initial ML modelling

- Data Collection
- Business Understanding
- EDA
- Feature Engineering
- Model Training
- Model Evaluation
- Causal Analysis

Now we target

Productionize ML model

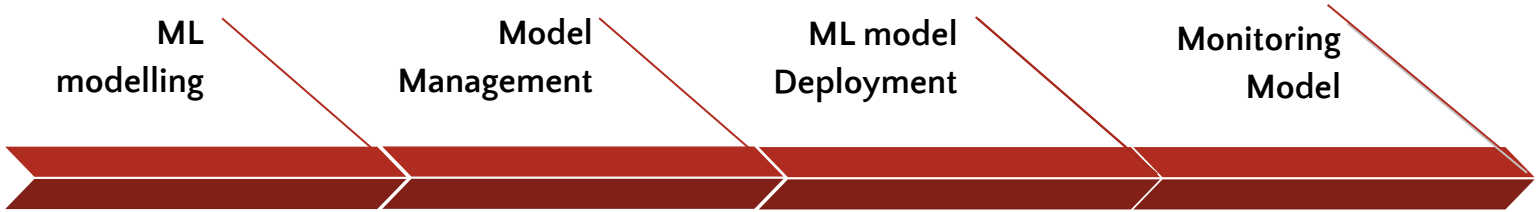
- Further optimisation of model with **Optuna**
- Manage model on **MLflow**
- Deploy model using **Databricks**

Operate ML model

- **Update** and **monitor** ML model

→ **Continuously reduce churn** among Telco's clients

Methodology Overview



Tool utilised	Pickle Optuna (for optimization)	MLflow tracking MLflows registry	Python API Databricks	Databricks

Expected outcome

Productionalised ML model in streamlined MLOps process

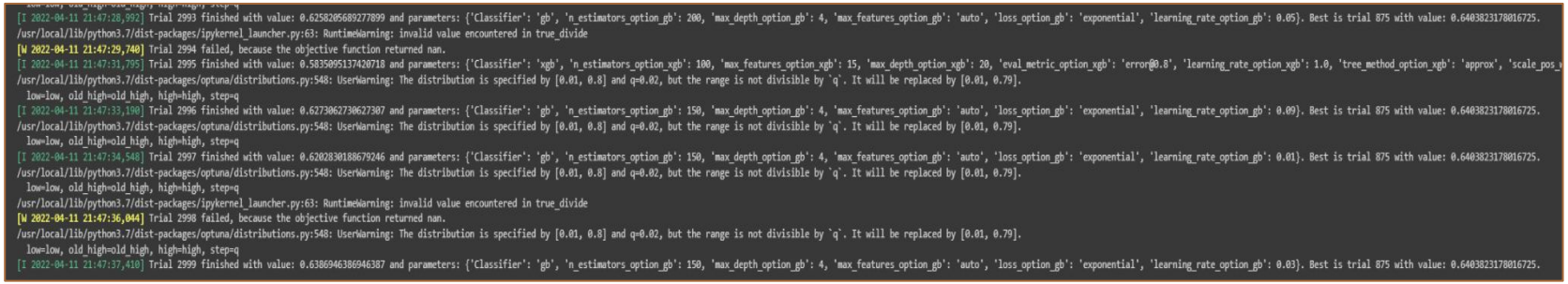
Modelling

Refactored Churn Predictions Modeling Optimization and Productionized Enterprise 1 code

Tool Utilized: Optuna

Output:

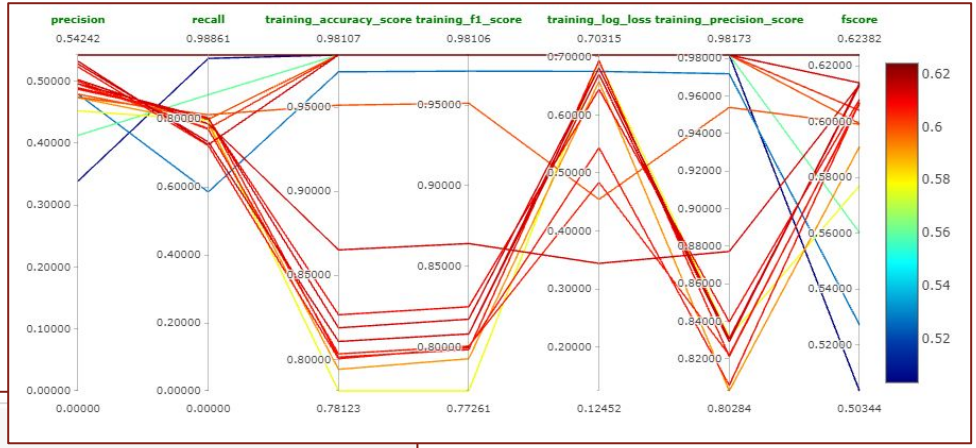
- Champion Model Pickle
- Optimal Model Threshold Pickle



Model Management

Tool utilised: MLflow Tracking

- Keep track of model parameters and metrics
- Find best model by comparing the metrics of different models in the history
- Best **F-score** in this case is **0.62**



Cmd 65

```
# register model to ML FLow registry

# this requires a database to hold the registry so it doesn't work locally, but it should work on Databricks
# if this doesn't work, try removing the '/gbmodel' from the first parameter of register_model

best_run_id = runs.loc[0]['run_id']
model_name = 'best_model'

best_model = mlflow.register_model(
    f'runs:/{best_run_id}/gb_model',
    model_name
    )

```

```
Registered model 'best_model' already exists. Creating a new version of this model...
2022/04/18 00:18:10 INFO mlflow.tracking._model_registry.client: Waiting up to 300 seconds for model version to finish creation.                    Model name: best_model, version 8
Created version '8' of model 'best_model'.
```

Command took 6.44 seconds -- by dany.stefan@mail.mcgill.ca at 4/17/2022, 8:16:00 PM on INSY_695_cluster

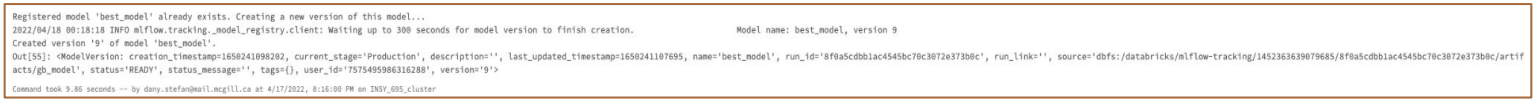
MLFlow Registry

- Save best model versions in registry
- When model is updated/optimized, archive old model and update new model stage to ‘production’
- Fetch model from registry and serve

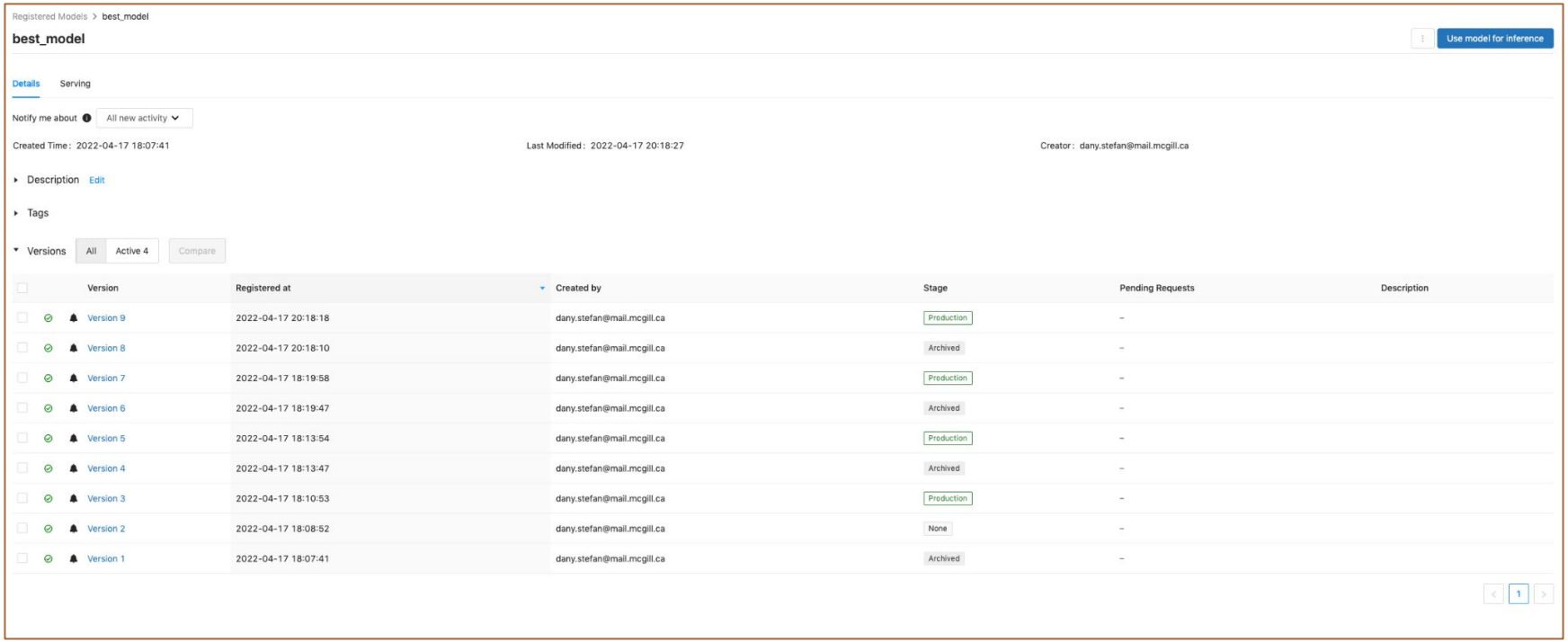
```
new_best_model = mlflow.register_model(f"runs:/{best_run.run_id}/gb_model", model_name)

# Archive the old model version
client.transition_model_version_stage(
  name=model_name,
  version=model_version.version,
  stage="Archived"
)

# Promote the new model version to Production
client.transition_model_version_stage(
  name=model_name,
  version=new_best_model.version,
  stage="Production"
)
```

Continuous Deployment

- Serving the registered models on Databricks
  - Via: the browser, Curl, Python API -> exposes the model through an endpoint that stays on the whole time & returns JSON output



Model Monitoring



Experiment ID: 1452363639079685

▸ Description Edit

Refresh | Compare | Delete | Download CSV | ↓ Start Time | All time | Columns | Only show diffe

Showing 87 matching runs

								Metrics			Parameters			Tags	
	↓ Start Time	Duration	Run Name	User	Source	Version	Models	recall	training_accura	training_f1_scc	algorithm	base_score	booster	estimator_class	estimator_nam
	29 minutes ago	2.7s	-	dany.stefan...	Productior	-	lightgbm	-	-	-	-	-	-	-	-
	30 minutes ago	5.0s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.783	-	-	-	-	-	-	-
	30 minutes ago	4.9s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.765	-	-	-	-	-	-	-
	30 minutes ago	5.0s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.821	-	-	-	-	-	-	-
	30 minutes ago	4.8s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.703	-	-	-	-	-	-	-
	30 minutes ago	4.9s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.69	-	-	-	-	-	-	-
	30 minutes ago	4.8s	-	dany.stefan...	Productior	-	lightgbm, 1 more	1	-	-	-	-	-	-	-
	30 minutes ago	4.8s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.765	-	-	-	-	-	-	-
	30 minutes ago	4.7s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.765	-	-	-	-	-	-	-
	30 minutes ago	4.8s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.765	-	-	-	-	-	-	-
	30 minutes ago	11.2s	-	dany.stefan...	Productior	-	sklearn, 1 more	0.791	0.951	0.951	-	-	-	sklearn.ens...	GradientBo...
	31 minutes ago	4.9s	-	dany.stefan...	Productior	-	xgboost, 1 more	0.979	-	-	-	None	None	-	-
	31 minutes ago	4.7s	-	dany.stefan...	Productior	-	xgboost, 1 more	0.979	-	-	-	None	None	-	-
	31 minutes ago	5.1s	-	dany.stefan...	Productior	-	sklearn, 1 more	0.797	0.783	0.775	-	-	-	sklearn.ens...	GradientBo...
	31 minutes ago	9.5s	-	dany.stefan...	Productior	-	best_model/9, 1 more	0	0.969	0.969	-	-	-	sklearn.ens...	GradientBo...
	31 minutes ago	5.0s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.799	-	-	-	-	-	-	-
	31 minutes ago	4.9s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.872	-	-	-	-	-	-	-
	31 minutes ago	5.1s	-	dany.stefan...	Productior	-	sklearn, 1 more	0.805	0.793	0.791	-	-	-	sklearn.ens...	GradientBo...
	31 minutes ago	4.9s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.783	-	-	-	-	-	-	-
	31 minutes ago	5.1s	-	dany.stefan...	Productior	-	xgboost, 1 more	0.872	-	-	-	None	None	-	-
	31 minutes ago	6.6s	-	dany.stefan...	Productior	-	sklearn, 1 more	0.797	0.787	0.783	-	-	-	sklearn.ens...	GradientBo...
	2 hours ago	4.1s	-	dany.stefan...	Productior	-	sklearn	-	0.824	0.823	-	-	-	sklearn.ens...	GradientBo...
	2 hours ago	7.1s	-	dany.stefan...	Productior	-	sklearn, 1 more	0.722	0.875	0.874	-	-	-	sklearn.ens...	GradientBo...
	2 hours ago	6.3s	-	dany.stefan...	Productior	-	sklearn, 1 more	0.749	0.824	0.823	-	-	-	sklearn.ens...	GradientBo...
	2 hours ago	6.5s	-	dany.stefan...	Productior	-	sklearn, 1 more	0.749	0.824	0.823	-	-	-	sklearn.ens...	GradientBo...
	2 hours ago	5.0s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.794	-	-	-	-	-	-	-
	2 hours ago	5.1s	-	dany.stefan...	Productior	-	sklearn, 1 more	0.757	0.811	0.809	-	-	-	sklearn.ens...	GradientBo...
	2 hours ago	5.1s	-	dany.stefan...	Productior	-	sklearn, 1 more	0.714	0.825	0.823	-	-	-	sklearn.ens...	GradientBo...
	2 hours ago	5.2s	-	dany.stefan...	Productior	-	lightgbm, 1 more	0.757	-	-	-	-	-	-	-

Next Steps

Our Solution

- Deploy state of the art models
- Model experimentation & data-logging with MLFlow
- Deployment on Databricks
- Communication with REST

Extensions

- Security with Secure Endpoint (Azure Private Link)
- Model Bias & Drift detection
  - A/B Testing
  - Drift Detection
    - KL Divergence, KS Tests
    - Population Stability Index
- Non-production (Exploratory):
  - Model Explainability with SHAP
  - Causal Analysis (from previous project)
  - Optimization & Cost-saving Reporting (from previous project)
- Alternate Deployment
  - Docker & Fast API

Thanks!

Any questions ?

You can find us at

- Github Repository - MMA4_ChurnPrediction