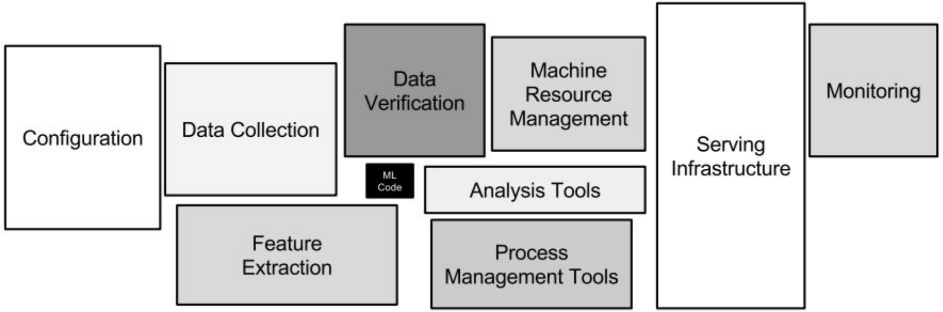
Machine Learning Bot for Financial Market

Submitted by

Wirachapong Suwanphibun
Sivakorn Lerttripinyo
Krittapasa Boontaveekul

Background

- Historical data in financial market is always collected.
- Collected historical prices can be used to predict a future trend.
- Machine Learning maybe able to predict by learning from data.
- Training Model is only a part of machine learning development project.



Background: Problem

- Reliable data source
- Experiment data, such as models, hyperparameters, should be recorded
- Deploying models to the real-world application
- Changing a model version
- Underlying infrastructures to implement a system

Background

- Objective
    - Design and implement a system going beyond the machine learning experiment.
    - Conduct an experiment on a price prediction model
        - Input
        - Output
        - Evaluation

Outline of this Presentation

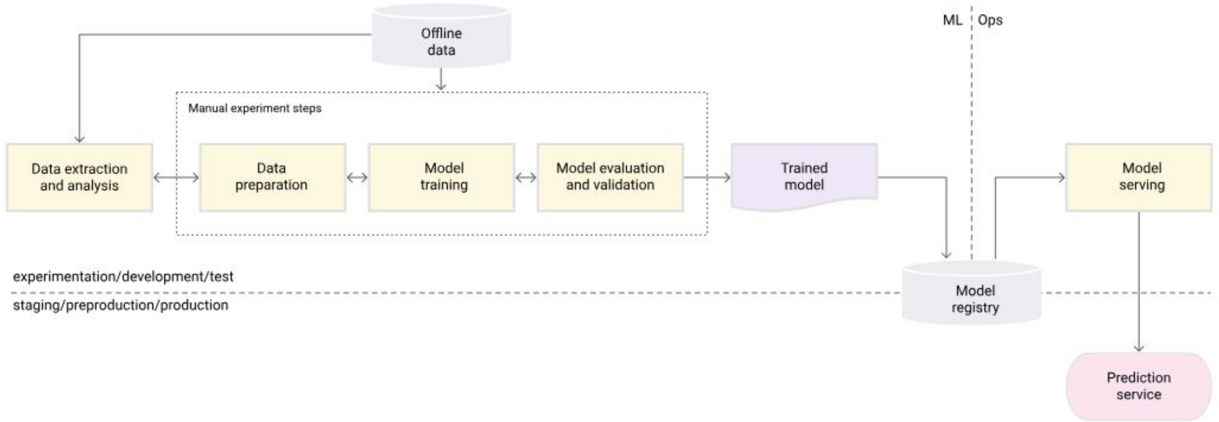
- System
  - Design
  - Experiment
  - Result and Discussion
- Model Development
  - Design
  - Experiment
  - Result and Discussion
- Conclusion
  - Assessment
  - Next Steps

Design Requirement

- Steps for implementing ML project
    - Data Extraction
    - Data Analysis
    - Data Preparation
    - Model Training
    - Model Evaluation
    - Model Serving
    - Model Monitoring

Design Requirement

- Applying MLOps: Use Maturity Model to evaluate
    - No standardized model, but there are some proposals by Google and Microsoft
    - Google Model: Level of Automation
    - Microsoft Model: Technical Capability
- This project starts from Google’s MLOps Level 0
    - All components are run manually.



Design Requirement

- Then, the system is improved to achieve most features from Microsoft’s MLOps Level 2
    - A data pipeline automatically gathers data
    - Experiment results are tracked
    - Both the training code and the resulting models are version-controlled
    - Implementing models are heavily dependent on data scientist expertise
    - Application code has unit tests
    - Basic integration tests exist for the model

Design Requirement

- Required Components
  - Source Code Repository
  - Model Training Infrastructure
  - Model Registry
  - ML Metadata Stores
  - Model Serving Component
  - Model Monitoring Component
  - Price Prediction Model: Be covered in the next section
    - One or more features as an input
    - A single value as an output
      - Price itself
      - Difference in price

Infrastructure Management

- AWS and DigitalOcean
  - Free Tier for AWS
  - Free 200 USD for DigitalOcean
- Infrastructure as Code (IaC) to manage infrastructures’ specification
  - Terraform
  - Backend on AWS S3 (State) and AWS DynamoDB (LockID)
    - Locking Management
    - State Management
    - Secret
  - Having additional infrastructures for managing the state is worthwhile, albeit troublesome.

```
resource "aws_security_group" "cap_aws_sg" {
  name = "postgresql-rds-sg"

  ingress {
    from_port = 5432
    to_port   = 5432
    # from_port   = 0
    # to_port     = 65535
    protocol    = "TCP"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port = 5432
    to_port   = 5432
    # from_port   = 0
    # to_port     = 65535
    protocol    = "TCP"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

terraform > 03-main.tf

```
provider "aws" {
  region = "ap-southeast-1"
}

resource "aws_security_group" "cap_aws_sg" {…
}

resource "aws_db_instance" "seniorproj_db" {
  identifier              = "maindb-seniorproj"
  engine                  = "postgres"
  engine_version          = "14.10"
  instance_class          = "db.t3.micro"
  db_name                 = "seniorproj_maindb"
  username                = var.db_username
  password                = var.db_password
  allocated_storage       = 20
  storage_type            = "gp2"
  publicly_accessible     = true
  backup_retention_period = 2
  skip_final_snapshot     = true
  vpc_security_group_ids  = [aws_security_group.cap_aws_sg.id]

  tags = {
    Name = "seniorproj_db"
  }
}
```

terraform apply
-var-file="02-value.tfvars"

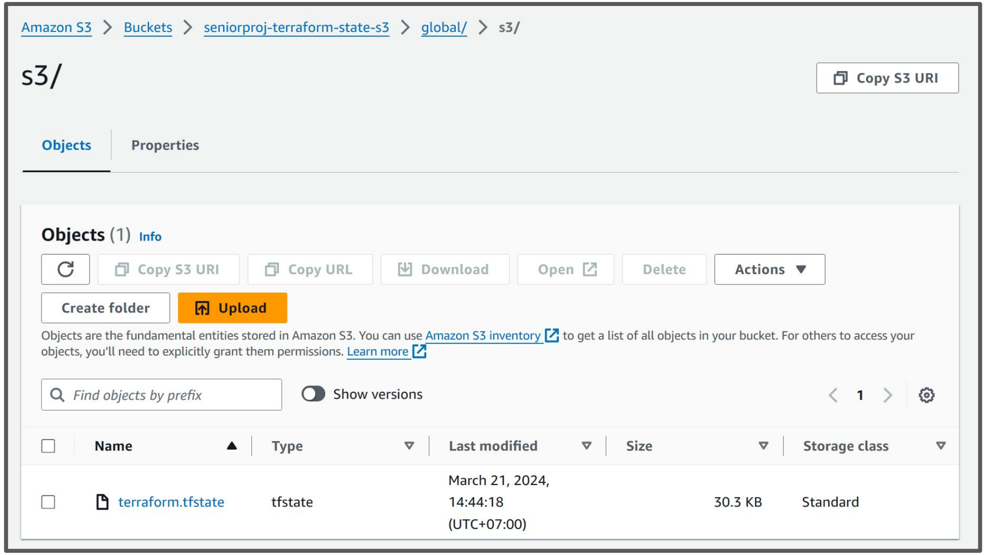
.tfvars is not stored in GitHub.

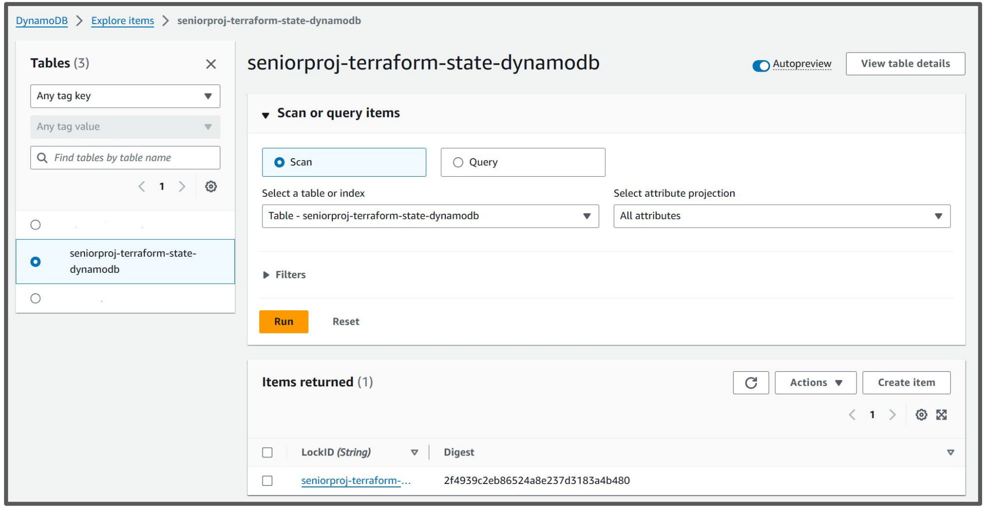
Terraform State File

terraform.json ×

C: > Users > super > AppData > Local > Temp > MicrosoftEdgeDownloads > be65621f-95c9-4b6a-bbf2-2c3ec07e

```
  "resources": [
    {
      "instances": [
        {
          "attributes": {
            "license_model": "postgresql-license",
            "listener_endpoint": [],
            "maintenance_window": "tue:16:16-tue:16:46",
            "manage_master_user_password": null,
            "master_user_secret": [],
            "master_user_secret_kms_key_id": null,
            "max_allocated_storage": 0,
            "monitoring_interval": 0,
            "monitoring_role_arn": "",
            "multi_az": false,
            "nchar_character_set_name": "",
            "network_type": "IPV4",
            "option_group_name": "default:postgres-14",
            "parameter_group_name": "default.postgres14",
            "password": "df[redacted]F",
            "performance_insights_enabled": false,
            "performance_insights_kms_key_id": "",
            "performance_insights_retention_period": 0,
            "port": 5432,
```


Amazon S3 > Buckets > seniorproj-terraform-state-s3 > global/ > s3/
s3/
Copy S3 URI
Objects
Properties
Objects (1) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Show versions
Name
Type
Last modified
Size
Storage class
terraform.tfstate
tfstate
March 21, 2024, 14:44:18 (UTC+07:00)
30.3 KB
Standard


DynamoDB > Explore items > seniorproj-terraform-state-dynamodb
Tables (3)
Any tag key
Any tag value
Find tables by table name
seniorproj-terraform-state-dynamodb
seniorproj-terraform-state-dynamodb
Autopreview
View table details
Scan or query items
Scan
Query
Select a table or index
Table - seniorproj-terraform-state-dynamodb
Select attribute projection
All attributes
Filters
Run
Reset
Items returned (1)
Actions
Create item
LockID (String)
Digest
seniorproj-terraform-...
2f4939c2eb86524a8e237d3183a4b480

01-variable.tf | 02-value.tfvars | pass.txt | 10-ld.tf | 09-digitalocean.tf | .gitignore

terraform > 09-digitalocean.tf

```
# https://docs.digitalocean.com/reference/api/api-try-it-now/

resource "digitalocean_app" "bento_app" {
  spec {
    name   = "bento-app-spec-terraform"
    region = "sgp"

    env {
      key   = "AWS_ACCESS_KEY_ID"
      value = var.s3_access
      type  = "SECRET"
    }
    env {
      key   = "AWS_SECRET_ACCESS_KEY"
      value = var.s3_secret_access
      type  = "SECRET"
    }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # digitalocean_app.bento_app will be updated in-place
  ~ resource "digitalocean_app" "bento_app" {
        id                    = "df08d3a3-1025-4eec-ab85-9860a6f21f32"
        # (7 unchanged attributes hidden)

      ~ spec {
          ~ features = [
```

```
Error: Error acquiring the state lock

Error message: operation error DynamoDB: PutItem, https response error StatusCode: 400, RequestID:
FS2R7PP1VTQMKC2MFU4KD0MH1VVV4KQNSO5AEMVJF66Q9ASUAAJG, ConditionalCheckFailedException: The conditional request failed
Lock Info:
  ID:        aaa27854-b9ad-80ca-4c97-a1a3bbcffbfe
  Path:      seniorproj-terraform-state-s3/global/s3/terraform.tfstate
  Operation: OperationTypeApply
  Who:       CAQ_DESKTOP\super@CaQ_Desktop
  Version:   1.3.9
  Created:   2024-05-08 14:25:17.2242123 +0000 UTC
  Info:


Terraform acquires a state lock to protect the state from being written
```

Database Design

- Raw Data
  - Data from BinanceAPI
  - Relational DB
    - Apply constraint easily
  - PK: (time, currency)
  - All in one table
- Derived Data
  - One currency and one set of indicators has a dedicated table.
  - Store data after performing ETL from a data pipeline
  - Relational DB
    - Apply constraint easily
  - PK: (time, currency)
- PostgreSQL on AWS RDS

Database Design: Raw-data Database

Query Query History Data Output Messages Notifications

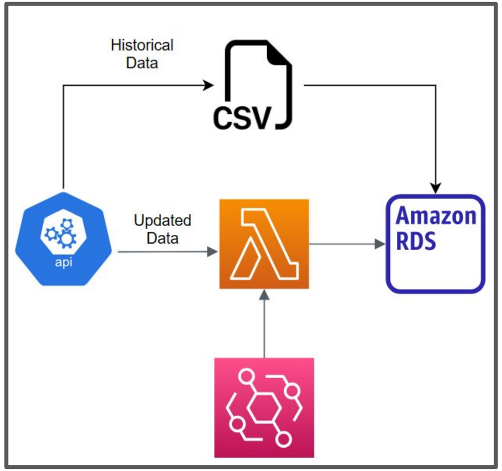
	currency [PK] text	time [PK] timestamp without time zone	open double precision	close double precision	low double precision	high double precision	volume double precision
1	BTCUSDT	2024-03-28 11:59:00	70637.99	70647.18	70616.41	70647.18	14.91807
2	BTCUSDT	2024-03-28 11:58:00	70673.05	70673.06	70626.33	70638	11.27007
3	BTCUSDT	2024-03-28 11:57:00	70702.01	70702.01	70673.05	70673.06	19.92578
4	BTCUSDT	2024-03-28 11:56:00	70747.01	70747.02	70700	70702	28.8656
5	BTCUSDT	2024-03-28 11:55:00	70682	70748	70682	70747.02	12.802
6	BTCUSDT	2024-03-28 11:54:00	70660	70692.06	70659.99	70682.01	22.7333
7	BTCUSDT	2024-03-28 11:53:00	70674.96	70681.99	70647.06	70660	14.35116
8	BTCUSDT	2024-03-28 11:52:00	70715.05	70715.05	70658.92	70674.96	16.10953
9	BTCUSDT	2024-03-28 11:51:00	70700	70727.93	70670.16	70715.04	28.70524
10	BTCUSDT	2024-03-28 11:50:00	70665.99	70720	70665.99	70700	28.57304
11	BTCUSDT	2024-03-28 11:49:00	70685.93	70685.94	70665.99	70666	14.85146
12	BTCUSDT	2024-03-28 11:48:00	70669.99	70701.1	70665.99	70685.94	19.95825

Database Design: Derived-data Database

- crypto_ind_adausdt
- crypto_ind_ethusdt
- crypto_ind_one

	time [PK] timestamp without time zone	currency [PK] text	close_minmax_scale double precision	close double precision	ma25_99h double precision	ma7_25h double precision	ma7_25d double precision	ma25_99h_scale double precision	ma7_25h_scale double precision	ma7_25d_scale double precision
1	2024-05-13 23:59:00	ETHUSDT	0.2227577896	2951.05	-2.862040404	6.6394285714	-122.3024571429	-0.000969838	0.002249853	-0.0414437089
2	2024-05-13 23:58:00	ETHUSDT	0.2212467244	2950.26	-3.1062505051	6.3083428571	-122.4149142857	-0.0010528735	0.0021382329	-0.0414929241
3	2024-05-13 23:57:00	ETHUSDT	0.2212467244	2950.26	-2.9413373737	6.0204571429	-122.7217142857	-0.0009969756	0.0020406531	-0.0415969149
4	2024-05-13 23:56:00	ETHUSDT	0.2192192192	2949.2	-2.8162666667	5.9367428571	-122.2489714286	-0.0009549256	0.0020130011	-0.0414515704
5	2024-05-13 23:55:00	ETHUSDT	0.2192000918	2949.19	-2.923830303	6.0408	-122.7245142857	-0.0009914011	0.0020482912	-0.0416129562
6	2024-05-13 23:54:00	ETHUSDT	0.2189705629	2949.07	-3.1964363636	5.8499428571	-122.0528571429	-0.0010838794	0.0019836568	-0.0413868973
7	2024-05-13 23:53:00	ETHUSDT	0.2176890266	2948.4	-3.1386909091	6.2338857143	-122.2893142857	-0.0010645404	0.0021143284	-0.0414765006
8	2024-05-13 23:52:00	ETHUSDT	0.2170004399	2948.04	-3.2212323232	6.0037142857	-122.7628	-0.0010926691	0.0020365105	-0.0416421758
9	2024-05-13 23:51:00	ETHUSDT	0.2175933896	2948.35	-3.5102626263	6.8582857143	-122.6173714286	-0.0011905855	0.0023261437	-0.041588472

Data Ingestion - Design



- Historical Data
    - Insert manually
        - Explore and Clean Data first
        - One-time Operation + Large data size
            - Bulk Insertion is more efficient
        - Find Pattern
            - Create Automation for further new-generated data
- New-Generated Data
    - Insert automatically
    - AWS Lambda + AWS EventBridge Rule

Data Ingestion - Usage

- 1440 rows per day per currency (1440 minutes)
- Limitation
    - Limit 1000 rows per query for BinanceAPI
        - Solution: 2 queries per day
            - 0:00 and 12:00 UTC+0
            - 720 rows per query
            - Can increase frequency(e.g. 3 per day) to update data more real-time
    - No Available Libraries in Lambda’s Python Runtime
        - Pack libraries as .zip and include as layers
        - Libraries must be downloaded in the same Python version as Lambda Runtime Env

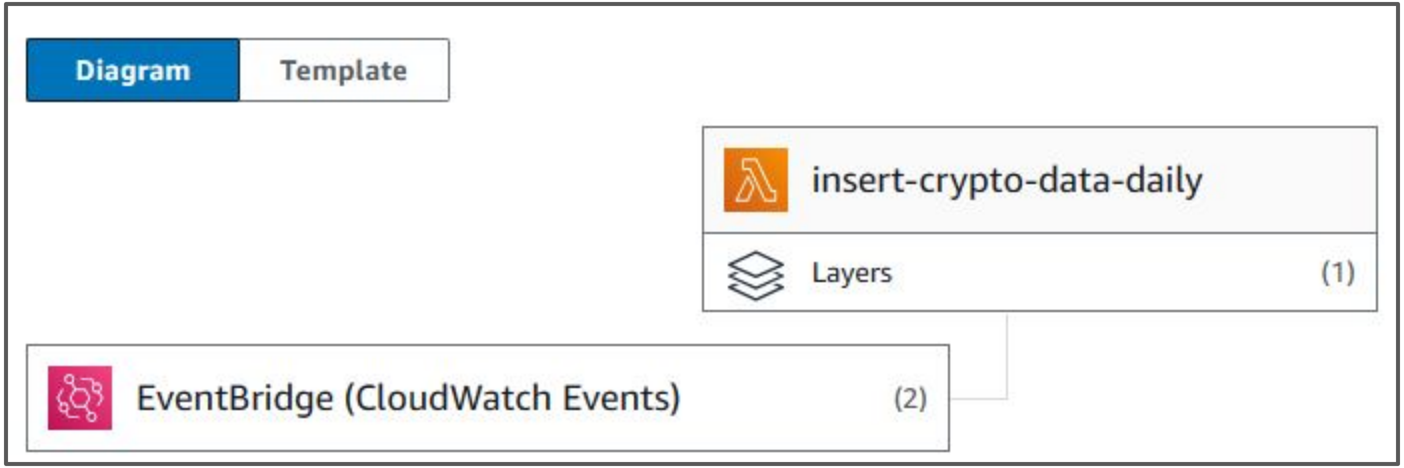


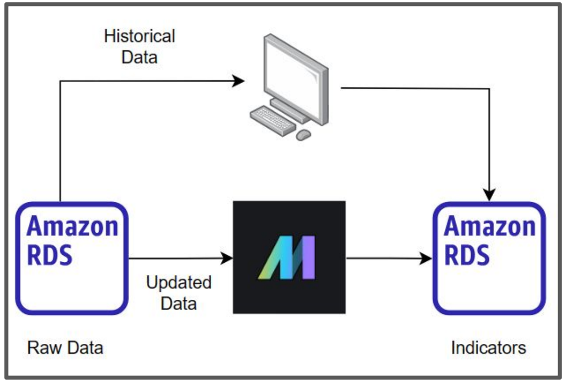
Diagram
Template
insert-crypto-data-daily
Layers
(1)
EventBridge (CloudWatch Events)
(2)

AWS Lambda Limitation

- Tough to debug and test
    - Not suitable for implementing a complex operation.
- Tough to manage dependencies
    - Although having layers, this method is time-consuming and error-prone
    - Layer having size > 50 MB must be separated to two smaller layers (AWS Limitation)
- More appropriate tool (e.g. data pipeline tool) should be used

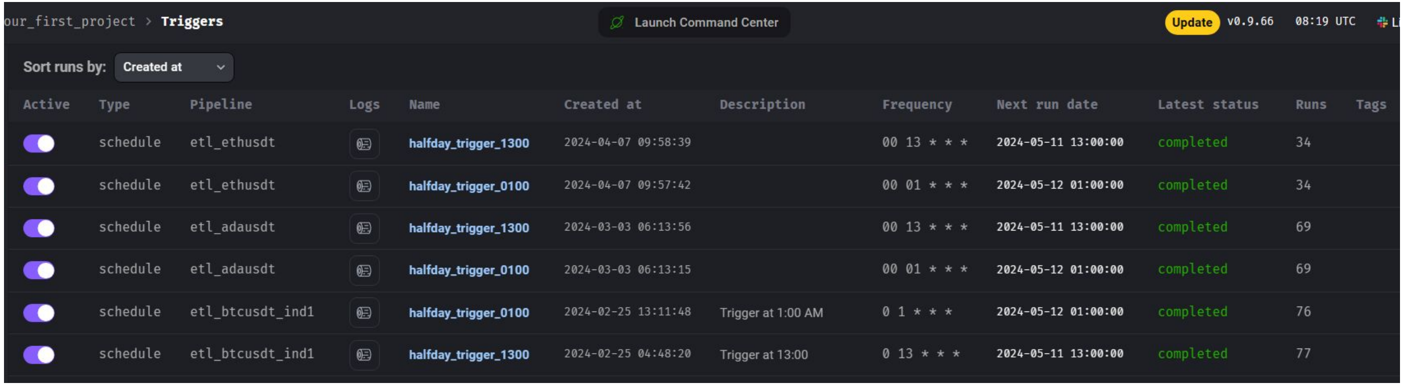
Data Transformation - Design

- Historical Data
  - Insert Manually
  - One-time Operation + Large amount of data
    - Not worthwhile to automate this
- New-generated Data
  - Automated ETL
  - Mage.ai (Data Pipeline Tool) is applied.
    - Apache Airflow requires at least 4GB RAM
    - Has less functions but more than enough for creating data pipelines



Data Transformation - Usage

- Extract data from raw-data database
- ETL is firstly implemented in Google Colab
  - Pandas is enough for approx 3M rows
  - Test with **Historical Data**
- Then, it is adapted as a data pipeline in Mage.ai
  - Directed Acyclic Graph (DAG)
  - Use the pipelines to transform **new-generated data**
- Schedules are configured to automatically trigger the pipeline to run

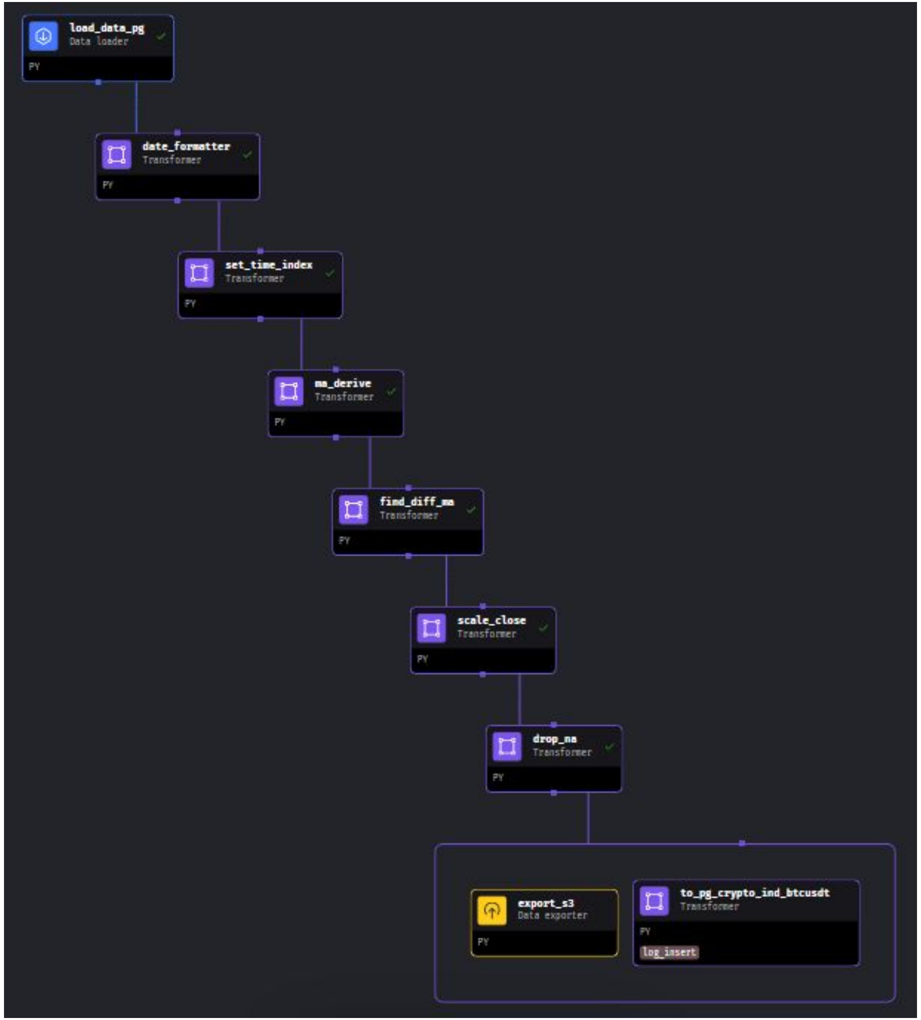

our_first_project > Triggers

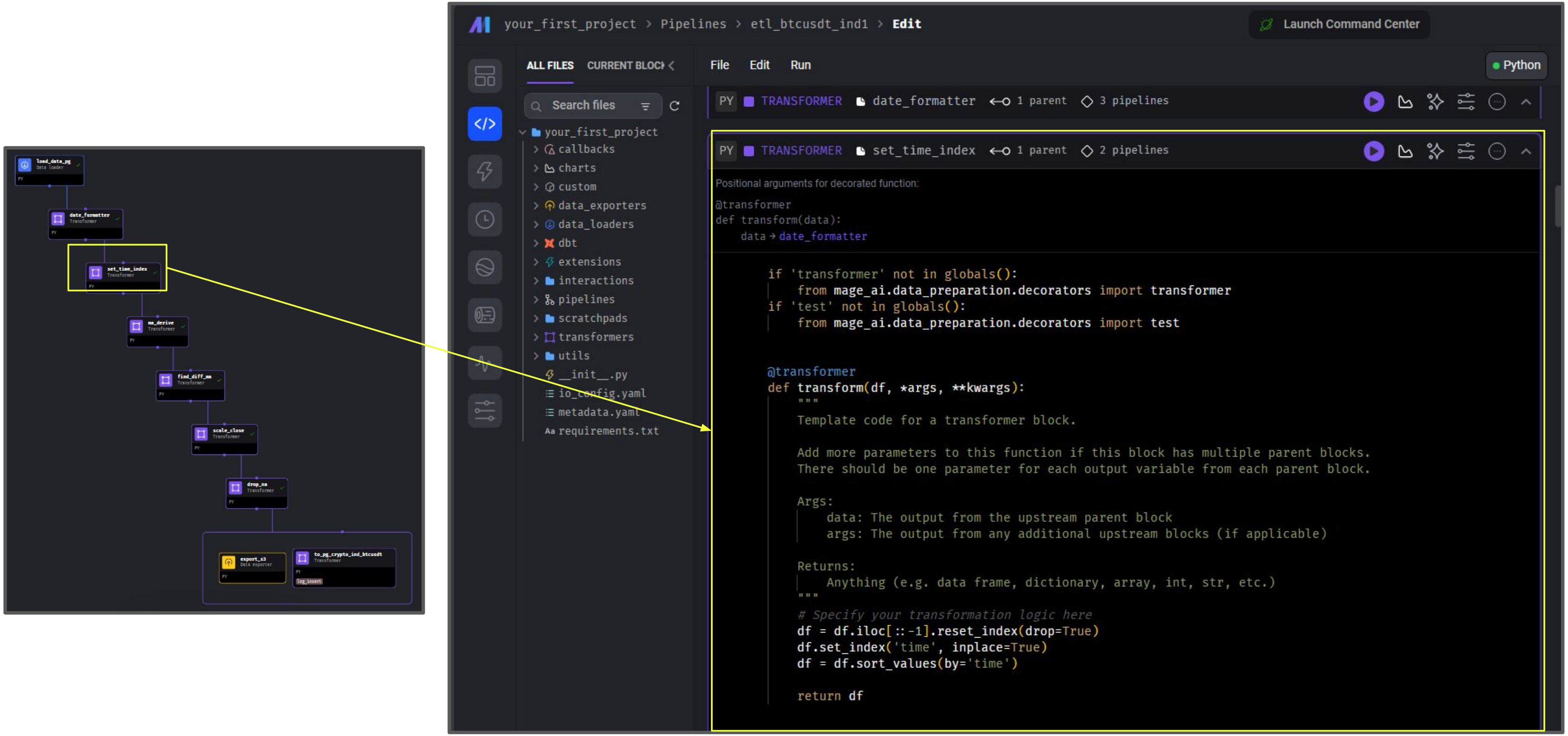
Launch Command Center

Update v0.9.66 08:19 UTC

Sort runs by: Created at

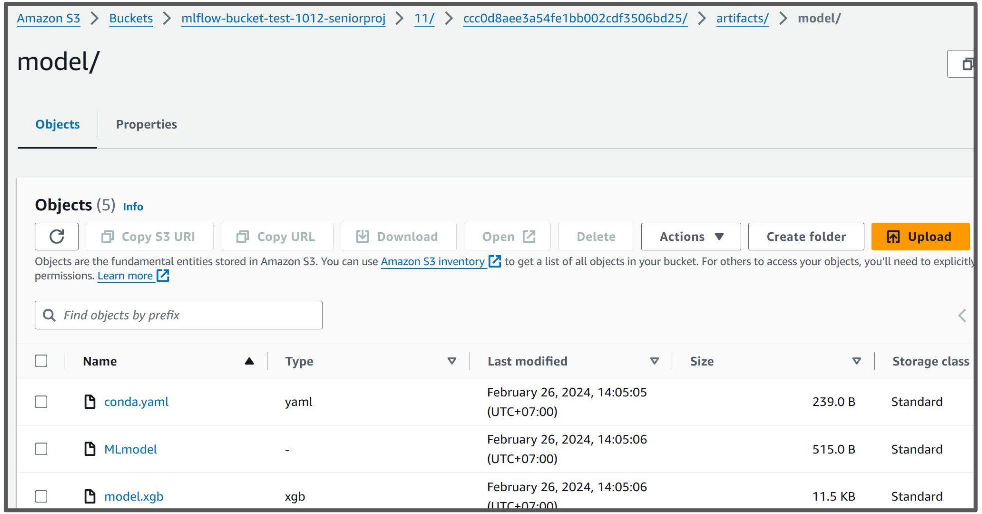
Active	Type	Pipeline	Logs	Name	Created at	Description	Frequency	Next run date	Latest status	Runs	Tags
	schedule	etl_ethusdt		halfday_trigger_1300	2024-04-07 09:58:39		00 13 * * *	2024-05-11 13:00:00	completed	34	
	schedule	etl_ethusdt		halfday_trigger_0100	2024-04-07 09:57:42		00 01 * * *	2024-05-12 01:00:00	completed	34	
	schedule	etl_adausdt		halfday_trigger_1300	2024-03-03 06:13:56		00 13 * * *	2024-05-11 13:00:00	completed	69	
	schedule	etl_adausdt		halfday_trigger_0100	2024-03-03 06:13:15		00 01 * * *	2024-05-12 01:00:00	completed	69	
	schedule	etl_btcusdt_ind1		halfday_trigger_0100	2024-02-25 13:11:48	Trigger at 1:00 AM	0 1 * * *	2024-05-12 01:00:00	completed	76	
	schedule	etl_btcusdt_ind1		halfday_trigger_1300	2024-02-25 04:48:20	Trigger at 13:00	0 13 * * *	2024-05-11 13:00:00	completed	77	


load_data_pg
Data loader
PY
date_formatter
Transformer
PY
set_time_index
Transformer
PY
ma_derive
Transformer
PY
find_diff_ma
Transformer
PY
scale_close
Transformer
PY
drop_na
Transformer
PY
export_s3
Data exporter
PY
to_pg_crypto_ind_btcusdt
Transformer
PY
log_insert

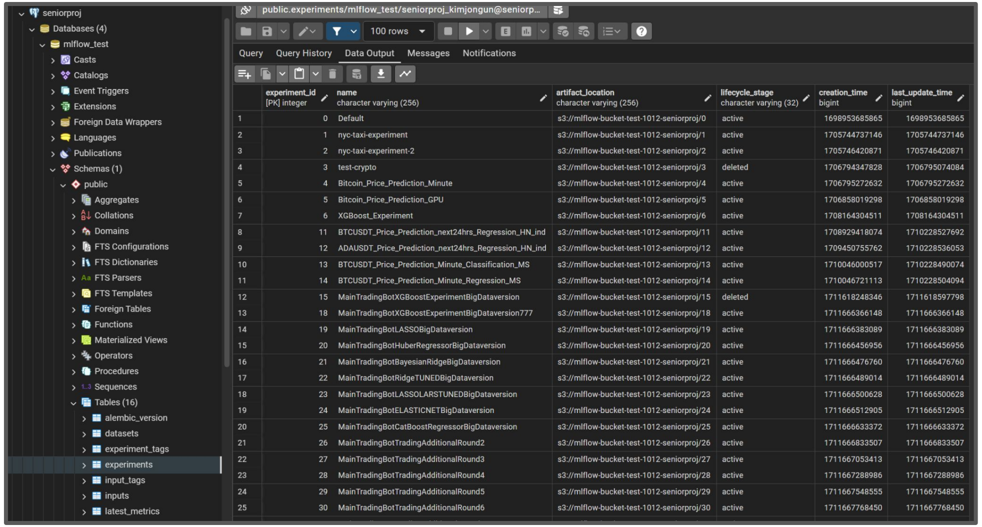

load_data_pg
date_formatter
set_time_index
ma_derive
find_diff_ma
scale_close
drop_na
export_s3
to_pg_crypto_ind_btcusdt
your_first_project > Pipelines > etl_btcusdt_ind1 > Edit
Launch Command Center
ALL FILES
CURRENT BLOCK
File
Edit
Run
Python
PY TRANSFORMER date_formatter 1 parent 3 pipelines
PY TRANSFORMER set_time_index 1 parent 2 pipelines
Search files
your_first_project
callbacks
charts
custom
data_exporters
data_loaders
dbt
extensions
interactions
pipelines
scratchpads
transformers
utils
__init__.py
io_config.yaml
metadata.yaml
requirements.txt
Positional arguments for decorated function:
@transformer
def transform(data):
data → date_formatter
if 'transformer' not in globals():
from mage_ai.data_preparation.decorators import transformer
if 'test' not in globals():
from mage_ai.data_preparation.decorators import test
@transformer
def transform(df, *args, **kwargs):
"""
Template code for a transformer block.
Add more parameters to this function if this block has multiple parent blocks.
There should be one parameter for each output variable from each parent block.
Args:
data: The output from the upstream parent block
args: The output from any additional upstream blocks (if applicable)
Returns:
Anything (e.g. data frame, dictionary, array, int, str, etc.)
"""
# Specify your transformation logic here
df = df.iloc[::-1].reset_index(drop=True)
df.set_index('time', inplace=True)
df = df.sort_values(by='time')
return df

Training Management System

- Track experiments, Record relevant information, and Be a centralized source for distributing models
- MLFlow: Hosted on EC2
  - Metadata: PostgreSQL on AWS RDS
  - Artifact and Object: AWS S3 Bucket
  - Prevent data loss from the instance failure
  - IAM is required for accessing data in DB and S3
    - “aws configure” to enter access and secret access key
    - Attach “Roles” directly to EC2
- Training code is modified to utilize MLFlow to track the experiment.



Name	Type	Last modified	Size	Storage class
conda.yaml	yaml	February 26, 2024, 14:05:05 (UTC+07:00)	239.0 B	Standard
MLmodel	-	February 26, 2024, 14:05:06 (UTC+07:00)	515.0 B	Standard
model.xgb	xgb	February 26, 2024, 14:05:06 (UTC+07:00)	11.5 KB	Standard

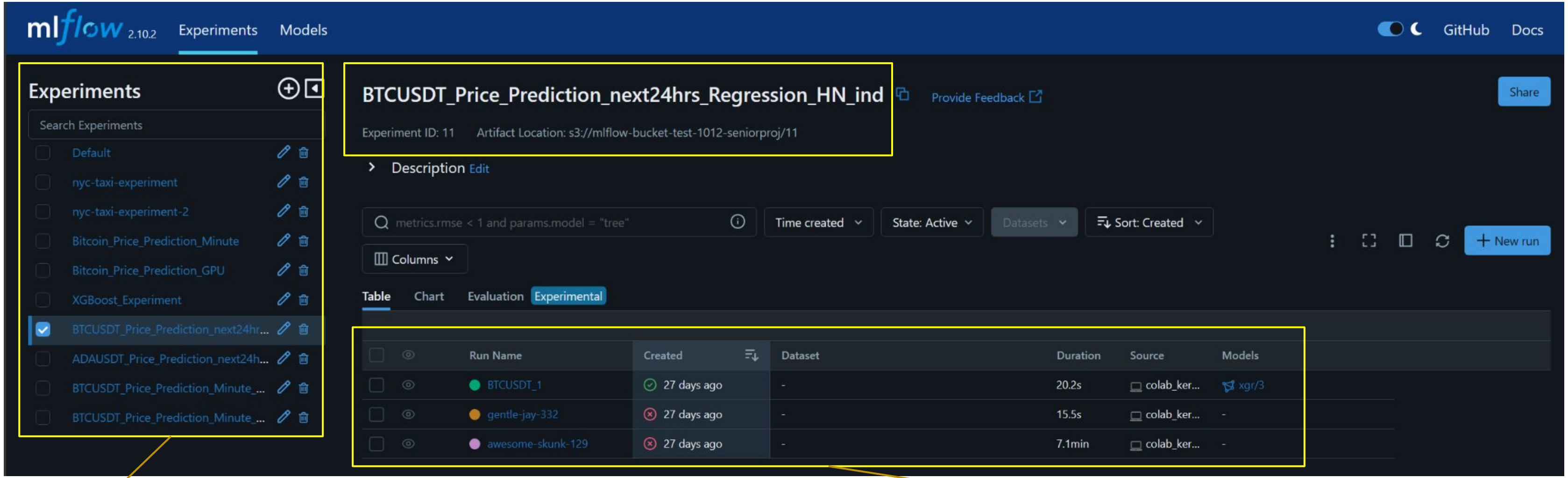


	experiment_id [PK] integer	name character varying (256)	artifact_location character varying (256)	lifecycle_stage character varying (32)	creation_time bigint	last_update_time bigint
1	0	Default	s3://mlflow-bucket-test-1012-seniorproj/0	active	1698953685865	1698953685865
2	1	nyc-taxi-experiment	s3://mlflow-bucket-test-1012-seniorproj/1	active	1705744737146	1705744737146
3	2	nyc-taxi-experiment-2	s3://mlflow-bucket-test-1012-seniorproj/2	active	1705746420871	1705746420871
4	3	test-crypto	s3://mlflow-bucket-test-1012-seniorproj/3	deleted	1706794347828	1706795074084
5	4	Bitcoin_Price_Prediction_Minute	s3://mlflow-bucket-test-1012-seniorproj/4	active	1706795272632	1706795272632
6	5	Bitcoin_Price_Prediction_GPU	s3://mlflow-bucket-test-1012-seniorproj/5	active	1706858019298	1706858019298
7	6	XGBoost_Experiment	s3://mlflow-bucket-test-1012-seniorproj/6	active	1708164304511	1708164304511
8	11	BTCUSDT_Price_Prediction_next24hrs_Regression_HN_ind	s3://mlflow-bucket-test-1012-seniorproj/11	active	1708929418074	1710228527692
9	12	ADAUSDT_Price_Prediction_next24hrs_Regression_HN_ind	s3://mlflow-bucket-test-1012-seniorproj/12	active	1709450755762	1710228536053
10	13	BTCUSDT_Price_Prediction_Minute_Classification_MS	s3://mlflow-bucket-test-1012-seniorproj/13	active	1710046000517	1710228490074
11	14	BTCUSDT_Price_Prediction_Minute_Regression_MS	s3://mlflow-bucket-test-1012-seniorproj/14	active	1710046721113	1710228504094
12	15	MainTradingBotXGBoostExperimentBigDataversion	s3://mlflow-bucket-test-1012-seniorproj/15	deleted	1711618248346	1711618597798
13	18	MainTradingBotXGBoostExperimentBigDataversion777	s3://mlflow-bucket-test-1012-seniorproj/18	active	1711666366148	1711666366148
14	19	MainTradingBotLASSOBigDataversion	s3://mlflow-bucket-test-1012-seniorproj/19	active	1711666383089	1711666383089
15	20	MainTradingBotHuberRegressorBigDataversion	s3://mlflow-bucket-test-1012-seniorproj/20	active	1711666456956	1711666456956
16	21	MainTradingBotBayesianRidgeBigDataversion	s3://mlflow-bucket-test-1012-seniorproj/21	active	1711666476760	1711666476760
17	22	MainTradingBotRidgeTUNEDBigDataversion	s3://mlflow-bucket-test-1012-seniorproj/22	active	1711666489014	1711666489014
18	23	MainTradingBotLASSOLARSTUNEDBigDataversion	s3://mlflow-bucket-test-1012-seniorproj/23	active	1711666500628	1711666500628
19	24	MainTradingBotELASTICNETBigDataversion	s3://mlflow-bucket-test-1012-seniorproj/24	active	1711666512905	1711666512905
20	25	MainTradingBotCatBoostRegressorBigDataversion	s3://mlflow-bucket-test-1012-seniorproj/25	active	1711666633372	1711666633372
21	26	MainTradingBotTradingAdditionalRound2	s3://mlflow-bucket-test-1012-seniorproj/26	active	1711666833507	1711666833507
22	27	MainTradingBotTradingAdditionalRound3	s3://mlflow-bucket-test-1012-seniorproj/27	active	1711667053413	1711667053413
23	28	MainTradingBotTradingAdditionalRound4	s3://mlflow-bucket-test-1012-seniorproj/28	active	1711667288986	1711667288986
24	29	MainTradingBotTradingAdditionalRound5	s3://mlflow-bucket-test-1012-seniorproj/29	active	1711667548555	1711667548555
25	30	MainTradingBotTradingAdditionalRound6	s3://mlflow-bucket-test-1012-seniorproj/30	active	1711667768450	1711667768450

Experiment Name

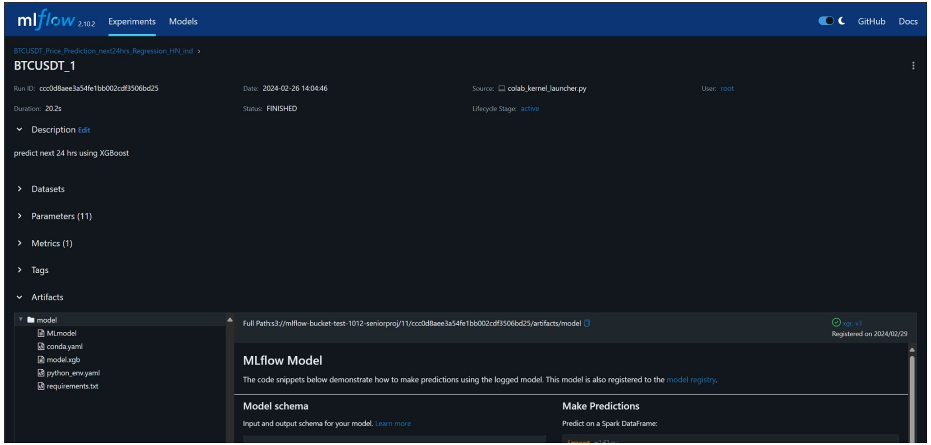
```
TRACKING_SERVER_HOST = "..."
mlflow.set_tracking_uri(f"http://{TRACKING_SERVER_HOST}:<port>")
mlflow.set_experiment("...")
with mlflow.start_run(run_name="..."):
    params = {
            "colsample_bytree": 0.3,
            "learning_rate": 0.1,
            <more params can be added>
        }
    mlflow.log_params(params)
    xg_reg = xgb.XGBRegressor(**params)
    xg_reg.fit(X_train, y_train)
    y_test_pred = xg_reg.predict(X_test)
    test_mse = mean_squared_error(y_test, y_test_pred)
    mlflow.log_metric("test_mse", test_mse)
    mlflow.xgboost.log_model(xg_reg, "model")
```

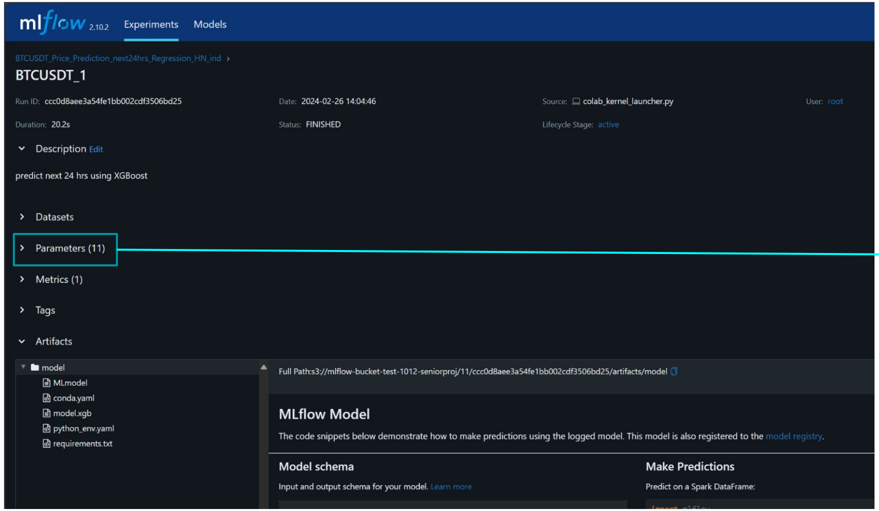
Original Training Code

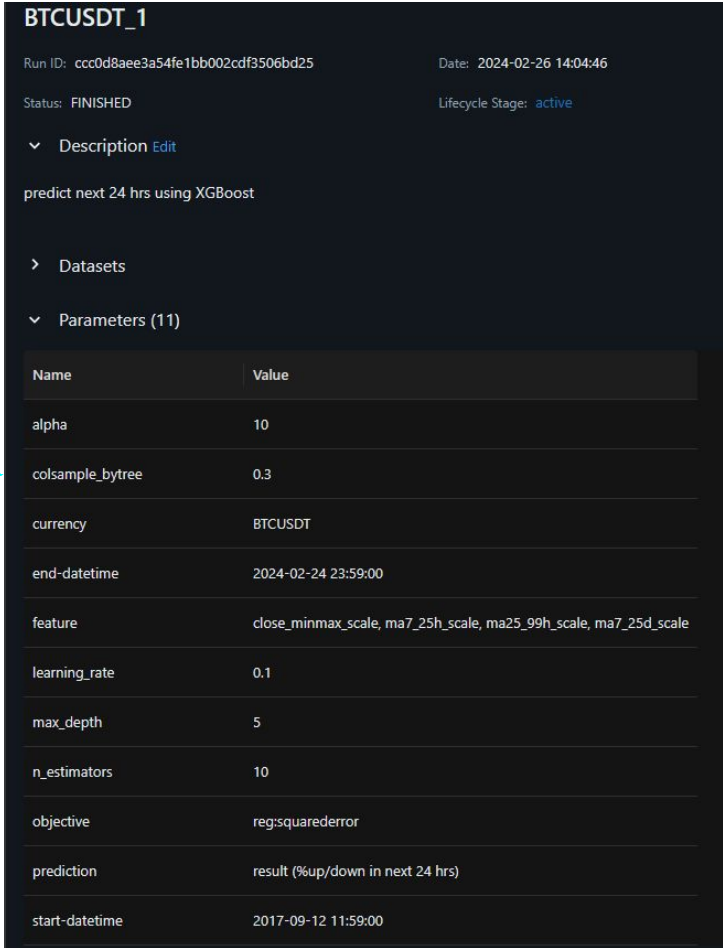


Experiment Names

Run names

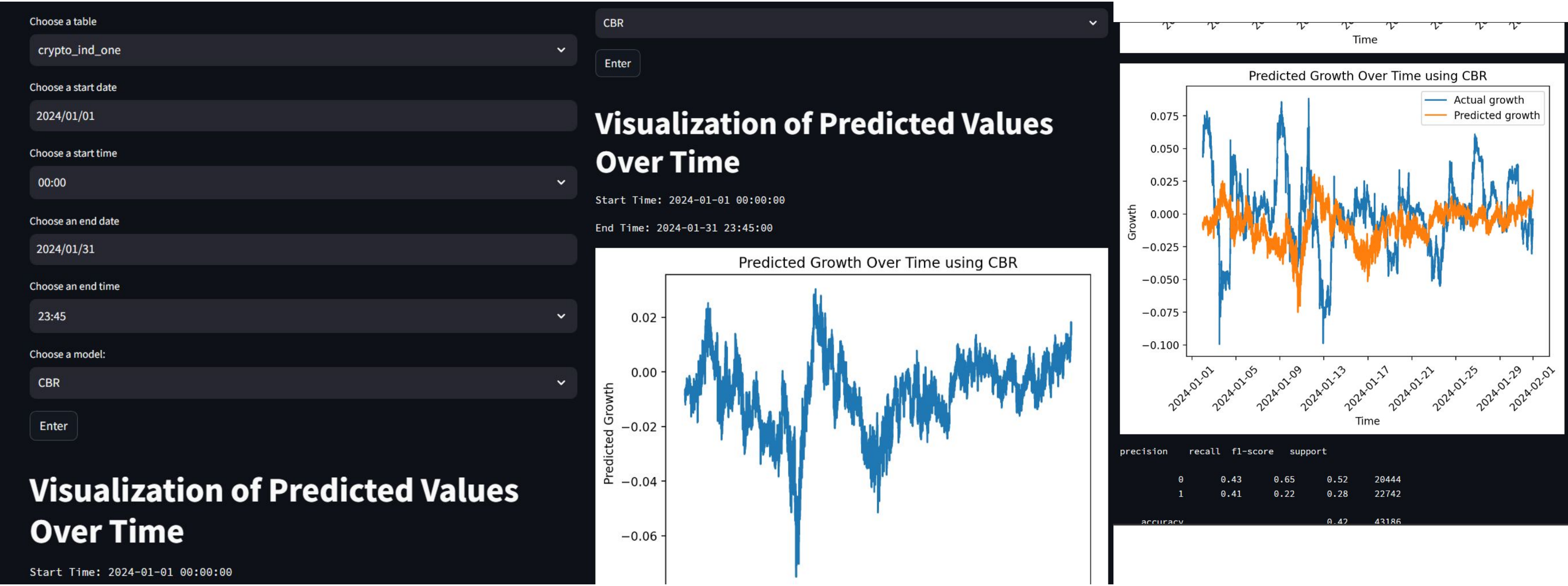



mlflow 2.10.2
Experiments
Models
BTCUSDT_Price_Prediction_next24hrs_Regression_HN_ind ›
BTCUSDT_1
Run ID: ccc0d8aee3a54fe1bb002cdf3506bd25
Date: 2024-02-26 14:04:46
Source: colab_kernel_launcher.py
User: root
Duration: 20.2s
Status: FINISHED
Lifecycle Stage: active
Description Edit
predict next 24 hrs using XGBoost
Datasets
Parameters (11)
Metrics (1)
Tags
Artifacts
model
MLmodel
conda.yaml
model.xgb
python_env.yaml
requirements.txt
Full Path:s3://mlflow-bucket-test-1012-seniorproj/11/ccc0d8aee3a54fe1bb002cdf3506bd25/artifacts/model
MLflow Model
The code snippets below demonstrate how to make predictions using the logged model. This model is also registered to the model registry.
Model schema
Input and output schema for your model. Learn more
Make Predictions
Predict on a Spark DataFrame:


BTCUSDT_1
Run ID: ccc0d8aee3a54fe1bb002cdf3506bd25
Date: 2024-02-26 14:04:46
Status: FINISHED
Lifecycle Stage: active
Description Edit
predict next 24 hrs using XGBoost
Datasets
Parameters (11)
Name
Value
alpha
10
colsample_bytree
0.3
currency
BTCUSDT
end-datetime
2024-02-24 23:59:00
feature
close_minmax_scale, ma7_25h_scale, ma25_99h_scale, ma7_25d_scale
learning_rate
0.1
max_depth
5
n_estimators
10
objective
reg:squarederror
prediction
result (%up/down in next 24 hrs)
start-datetime
2017-09-12 11:59:00

Model Monitoring

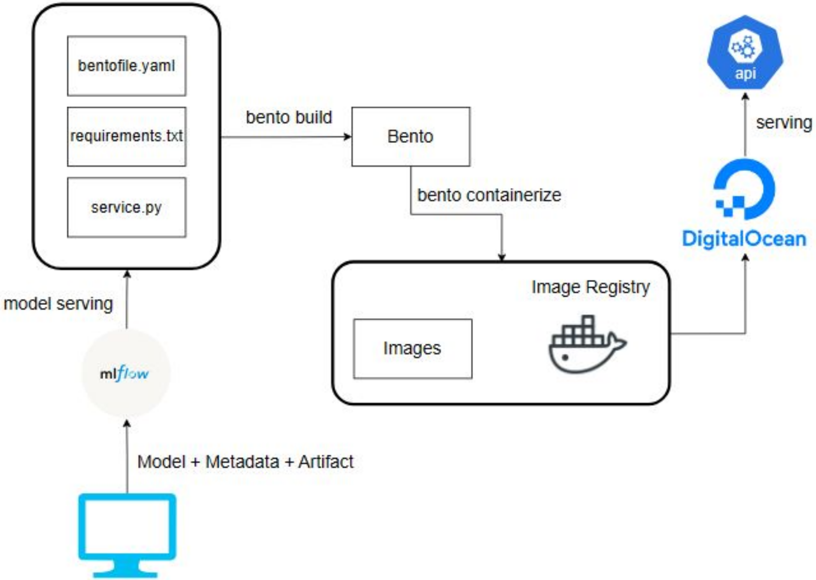
- Continuously monitor ML model
- Jupyter Notebook is inconvenient for opening and sharing
- Webpage is better
  - Easier to share a line and open in any supported browser
  - Streamlit
    - Python-based: Easy to integrate with Matplotlib

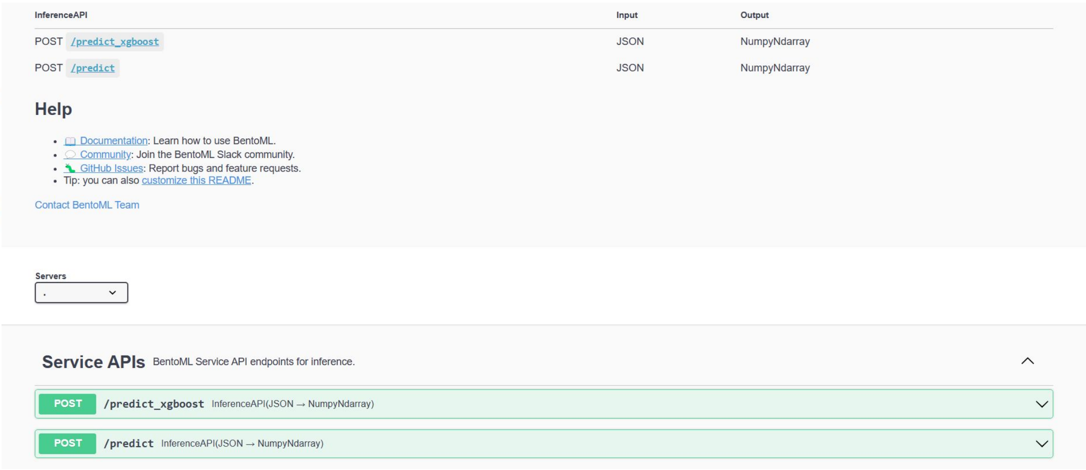


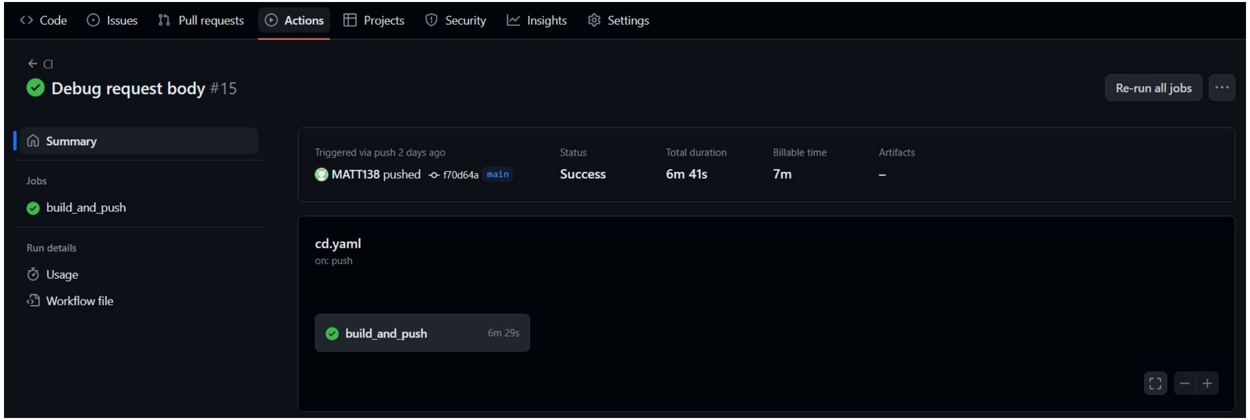
Model results will be discussed later in the next section.

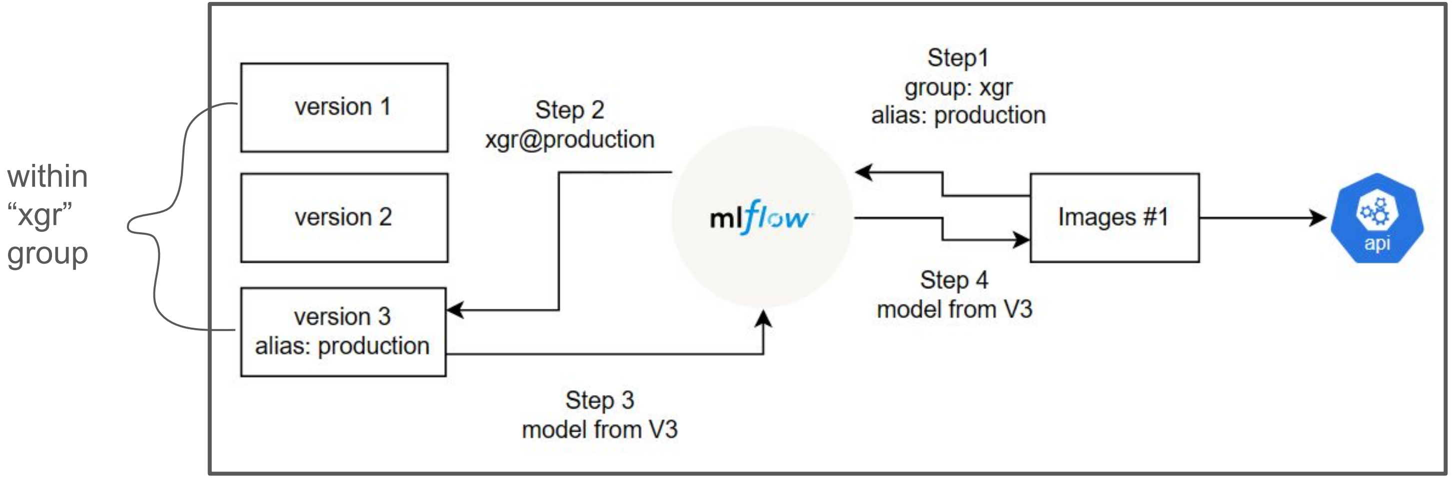
Model Building and Serving

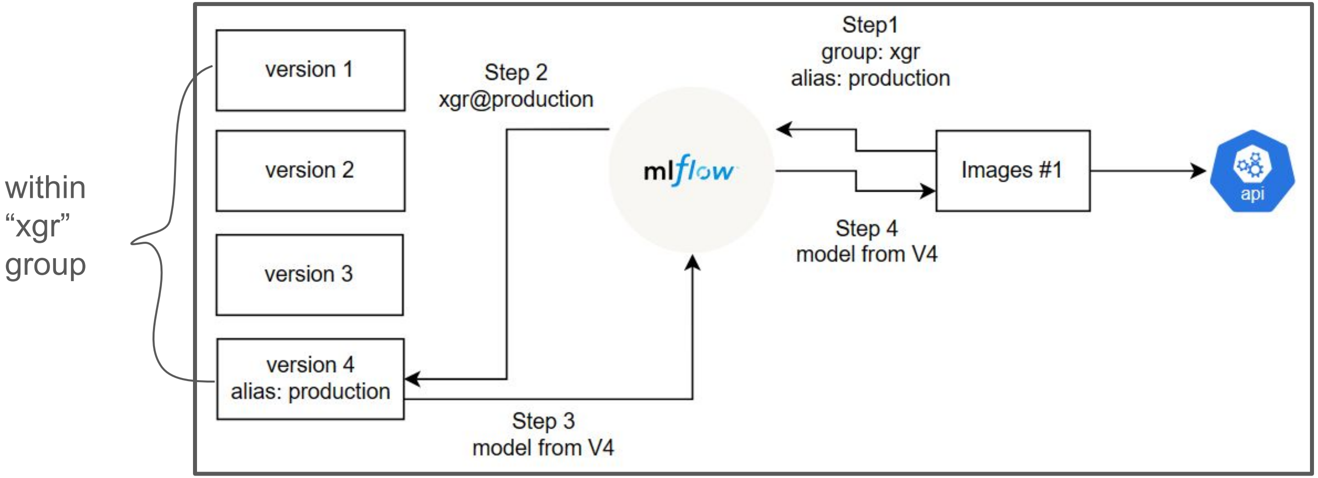
- Bridging between the model experiment and the real-world application
- Building an API as an image and Deploying the image


bentofile.yaml
requirements.txt
service.py
bento build
Bento
bento containerize
Image Registry
Images
serving
api
DigitalOcean
model serving
mlflow
Model + Metadata + Artifact


InferenceAPI
Input
Output
POST /predict_xgboost JSON NumpyNdarray
POST /predict JSON NumpyNdarray
Help
Documentation: Learn how to use BentoML.
Community: Join the BentoML Slack community.
GitHub Issues: Report bugs and feature requests.
Tip: you can also customize this README.
Contact BentoML Team
Servers
Service APIs BentoML Service API endpoints for inference.
POST /predict_xgboost InferenceAPI(JSON → NumpyNdarray)
POST /predict InferenceAPI(JSON → NumpyNdarray)


Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
CI
Debug request body #15
Re-run all jobs
Summary
Jobs
build_and_push
Run details
Usage
Workflow file
Triggered via push 2 days ago
MATT138 pushed f70d64a main
Status
Success
Total duration
6m 41s
Billable time
7m
Artifacts
–
cd.yaml
on: push
build_and_push 6m 29s


within
“xgr”
group
version 1
version 2
version 3
alias: production
Step 2
xgr@production
Step1
group: xgr
alias: production
Images #1
api
Step 4
model from V3
Step 3
model from V3


within
“xgr”
group
version 1
version 2
version 3
version 4
alias: production
Step 2
xgr@production
Step1
group: xgr
alias: production
Images #1
api
Step 4
model from V4
Step 3
model from V4

Experiment Setup

- A software testing concept *is adapted* to test this system.
    - A unit test tests whether a particular system functions properly.
    - A component test will test whether an interaction between two systems work properly.
    - A system test is applied to verify that all systems can work together properly.
        - An integration test is skipped as our system has a small scale, and the borderline between the system and integration test is blurred

Experiment Setup: Infrastructure List

- AWS
    - S3 Bucket
    - Lambda (Python 3.9 as Runtime Environment) with AWS EventBridge Rules
    - EC2: t2.micro, 1vCPU, RAM 1GB
    - RDS PostgreSQL: t3.micro, 2vCPU, RAM 1 GB, gp2-20GiB Storage
    - DynamoDB: Pay per Request
- DigitalOcean
    - Droplet: RAM 4GB, 2vCPU, 80GB Storage
    - App Platform: RAM 2GB, 1vCPU
    - Container Registry: 5GB Storage

Experimental Method: Unit Test

- IaC
    - The scripts shall provision defined infrastructures with a defined specification.
    - The state after running the script shall be updated and stored in S3 bucket.
- Data Ingestion
    - Raw-data database shall be created.
    - Historical data shall be able to be inserted manually to the raw-data database.
    - New data shall be automatically retrieved from an external API and stored in the raw-data database at the time specified by triggers.
    - No duplicated data shall exist in the raw-data database.

Experimental Method: Unit Test

- Data Transformation
  - Derived-data database shall be created.
  - Historical data shall be able to be derived and inserted into the derived-data database manually.
  - The data pipeline shall be triggered at a specified time.
  - The data pipeline shall successfully retrieve data from a defined data source, derive new indicators, and store them in the derived-data database.
- Training Management
  - The system shall be able to create several experiments.
  - Each experiment shall be able to contain several runs.
  - Each run shall contain logged data, such as metrics and hyperparameters, and model.
  - The AWS access key and secret access key shall be required to log the experiment result from the local to the system.

Experimental Method: Unit Test

- Model Monitoring
  - The dashboard hosted on the website shall be able to visualize the model metrics on the selected currency and date range.
  - For this test, the hard-coded model and dataset can be used.
- Model Building and Serving
  - After pushing the code into the main branch, the GitHub shall be automatically activated, and the new built image shall be automatically deployed through the API.

Experimental Method: Component Test

- **The data ingestion system** shall always ingest new data and insert into the raw-data database. **The data transformation system** shall be able to set the raw-data database as the data source, derive new indicators, and insert them into the derived-data database.

- **The model building and serving system** shall be able to retrieve the model from **the training management system** and use it for serving. If there is a change in the model version for serving, the new version shall be able to be deployed and applied without rebuilding the image.

Experimental Method: Component and System Test

- **The model monitoring system** shall be able to retrieve data from **derived-data database** and the model from **the training management system.** They shall be used to generate a metric and graph to show the model performance.

- System test
  - All components shall be able to work together properly.
  - The overall system shall be run and periodically check whether there is any error occur

Non-Functional Requirements Testing

- Operational Requirements
  - The system shall provide a result of at least 99% uptime.
  - The system databases shall have 99% uptime.
  - The system shall backup all databases to prevent data loss.
- Performance Requirements
  - The system shall store daily data every day.
  - The system shall provide a response within 5 seconds after making a request.
- Security and Privacy
  - The system shall scrape data from a legal source only.
  - The system shall allow only team members to access databases.

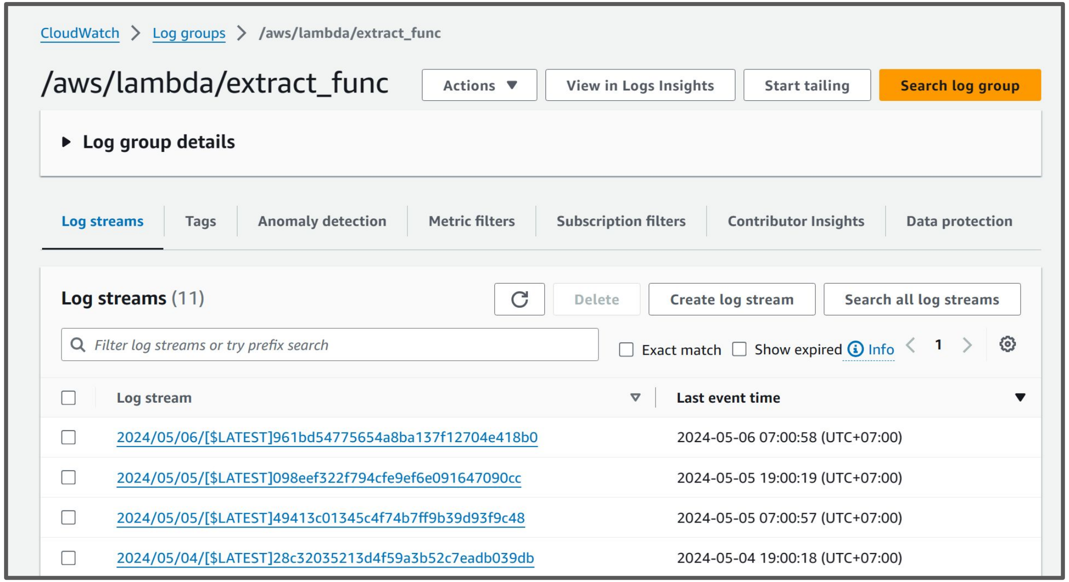
Test Result

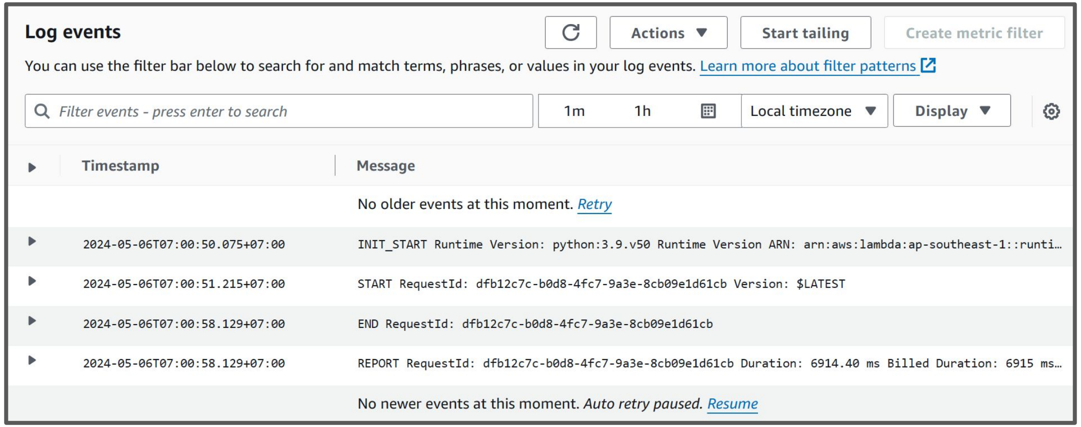
Topic	Result (Pass/Partial Pass/Fail)
Infrastructure as Code	Pass
Data ingestion system	Pass
Data transformation system	Pass
Training management system	Pass
Model monitoring system	Pass
Model building and serving system	Pass

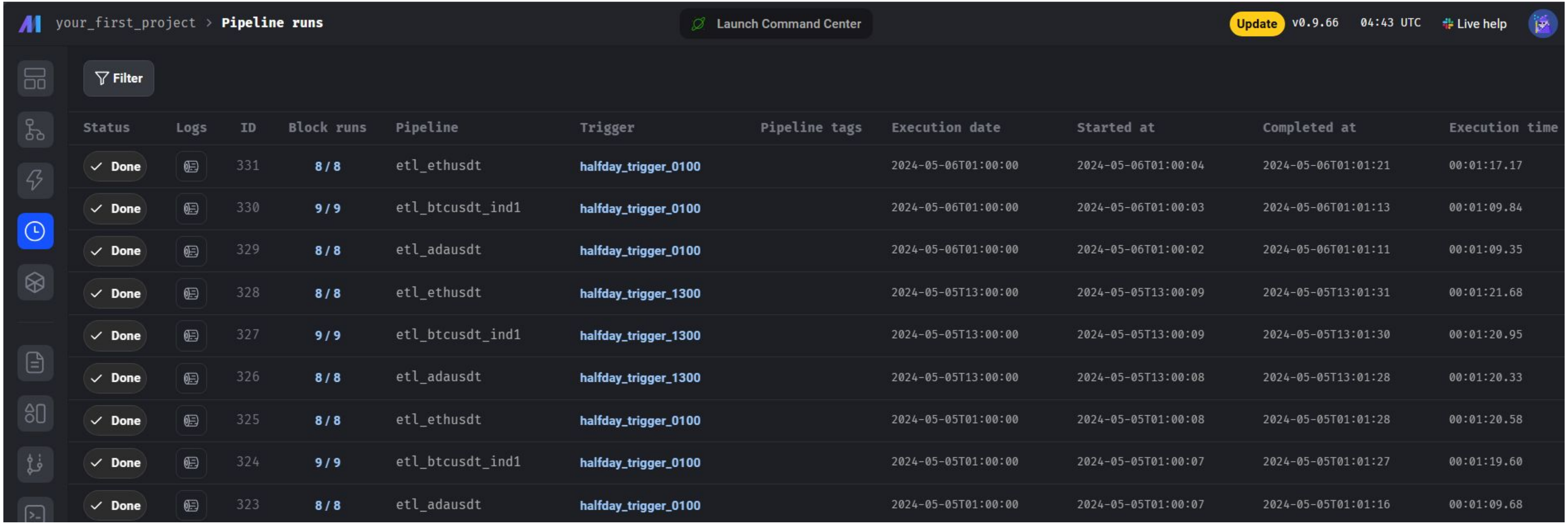
Table 2: Unit Test Result

Topic	Result (Pass/Partial Pass/Fail)
Data Ingestion + Data Transformation	Pass
Building and Serving + Management System	Pass
Model monitoring + Management System	Pass

Table 3: Component Test Result


CloudWatch > Log groups > /aws/lambda/extract_func
/aws/lambda/extract_func
Actions
View in Logs Insights
Start tailing
Search log group
Log group details
Log streams
Tags
Anomaly detection
Metric filters
Subscription filters
Contributor Insights
Data protection
Log streams (11)
Delete
Create log stream
Search all log streams
Filter log streams or try prefix search
Exact match
Show expired
Info
1
Log stream
Last event time
2024/05/06/[$LATEST]961bd54775654a8ba137f12704e418b0
2024-05-06 07:00:58 (UTC+07:00)
2024/05/05/[$LATEST]098eef322f794cfe9ef6e091647090cc
2024-05-05 19:00:19 (UTC+07:00)
2024/05/05/[$LATEST]49413c01345c4f74b7ff9b39d93f9c48
2024-05-05 07:00:57 (UTC+07:00)
2024/05/04/[$LATEST]28c32035213d4f59a3b52c7eadb039db
2024-05-04 19:00:18 (UTC+07:00)


Log events
Actions
Start tailing
Create metric filter
You can use the filter bar below to search for and match terms, phrases, or values in your log events. Learn more about filter patterns
Filter events - press enter to search
1m
1h
Local timezone
Display
Timestamp
Message
No older events at this moment. Retry
2024-05-06T07:00:50.075+07:00
INIT_START Runtime Version: python:3.9.v50 Runtime Version ARN: arn:aws:lambda:ap-southeast-1::runti…
2024-05-06T07:00:51.215+07:00
START RequestId: dfb12c7c-b0d8-4fc7-9a3e-8cb09e1d61cb Version: $LATEST
2024-05-06T07:00:58.129+07:00
END RequestId: dfb12c7c-b0d8-4fc7-9a3e-8cb09e1d61cb
2024-05-06T07:00:58.129+07:00
REPORT RequestId: dfb12c7c-b0d8-4fc7-9a3e-8cb09e1d61cb Duration: 6914.40 ms Billed Duration: 6915 ms…
No newer events at this moment. Auto retry paused. Resume



Status	Logs	ID	Block runs	Pipeline	Trigger	Pipeline tags	Execution date	Started at	Completed at	Execution time
Done		331	8 / 8	etl_ethusdt	halfday_trigger_0100		2024-05-06T01:00:00	2024-05-06T01:00:04	2024-05-06T01:01:21	00:01:17.17
Done		330	9 / 9	etl_btcusdt_ind1	halfday_trigger_0100		2024-05-06T01:00:00	2024-05-06T01:00:03	2024-05-06T01:01:13	00:01:09.84
Done		329	8 / 8	etl_adausdt	halfday_trigger_0100		2024-05-06T01:00:00	2024-05-06T01:00:02	2024-05-06T01:01:11	00:01:09.35
Done		328	8 / 8	etl_ethusdt	halfday_trigger_1300		2024-05-05T13:00:00	2024-05-05T13:00:09	2024-05-05T13:01:31	00:01:21.68
Done		327	9 / 9	etl_btcusdt_ind1	halfday_trigger_1300		2024-05-05T13:00:00	2024-05-05T13:00:09	2024-05-05T13:01:30	00:01:20.95
Done		326	8 / 8	etl_adausdt	halfday_trigger_1300		2024-05-05T13:00:00	2024-05-05T13:00:08	2024-05-05T13:01:28	00:01:20.33
Done		325	8 / 8	etl_ethusdt	halfday_trigger_0100		2024-05-05T01:00:00	2024-05-05T01:00:08	2024-05-05T01:01:28	00:01:20.58
Done		324	9 / 9	etl_btcusdt_ind1	halfday_trigger_0100		2024-05-05T01:00:00	2024-05-05T01:00:07	2024-05-05T01:01:27	00:01:19.60
Done		323	8 / 8	etl_adausdt	halfday_trigger_0100		2024-05-05T01:00:00	2024-05-05T01:00:07	2024-05-05T01:01:16	00:01:09.68

Test Result

Topic	Result (Pass/Partial Pass/Fail)
Operational	Pass
Performance	Partial Pass
Security and Privacy	Pass

Table 4: Non-functional Requirements Test

- Some operations provide a response using more than five seconds.
  - The involvement of retrieving large amount of data.

Discussion

- The system has delivered an acceptable result.
    - It contains the essential components for creating a simple end-to-end machine learning project that includes from the data system to serving system.
- IaC
    - Handle the unstandardized infrastructure problem.
    - Define infrastructures' specification including in this project and manages the provisioning work.
- Data Ingestion and Transformation
    - Handle the data quality and availability problem
    - clean, collect, transform, and centralize data
        - The most important ingredient for working with the machine learning model.

Discussion

- Training Management System
  - Handle the experiment information recording and the model version distribution problem
  - Store several versions of model, along with metadata
  - Centralized source for serving the model
- Model Monitoring System
  - Handle the unobserved model degradation problem
  - Dashboard on the website to display graphs and metrics
- Model Building and Serving system
  - Handle the model deployment problem
  - API for serving models
    - Ex. in this project: API is used by the model monitoring system

Discussion

- Each component can tackle various problems of the machine learning system development.
    - When connected together as a bot, it can work without the significant error.
- This bot can demonstrate the simple end-to-end machine learning system
    - It satisfies major functional and non-functional requirements
    - This project has effectively solved the identified issues
- Our project assessment will be covered later in the final section of this presentation.

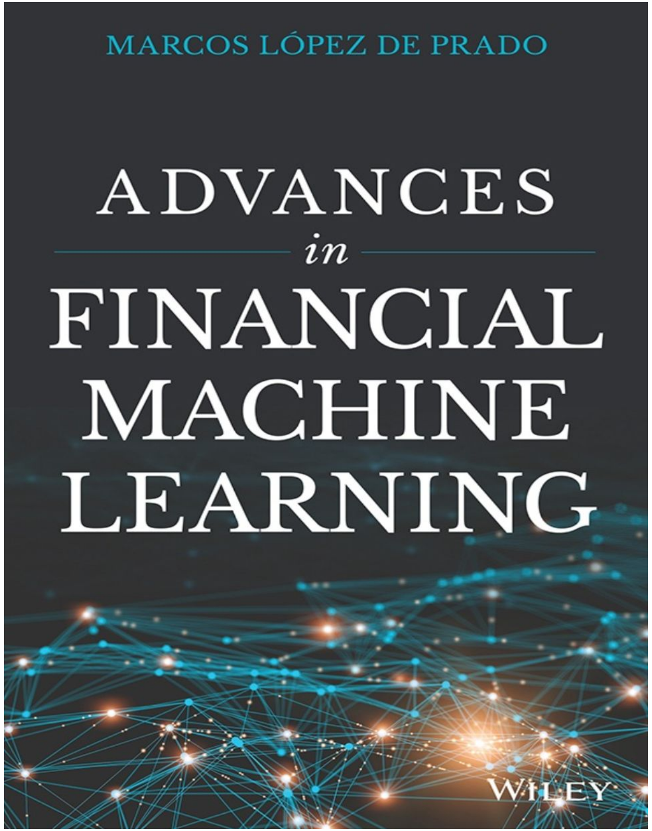
Discussion: Design Requirement Revisited

- Required Components
  - Source Code Repository: **GitHub**
  - Model Training Infrastructure: **Local Computer and Google Colab**
  - Model Registry: **MLFlow**
  - ML Metadata Stores: **PostgreSQL and S3**
  - Model Serving Component: **API by BentoML and DigitalOcean App Platform**
  - Model Monitoring Component: **Streamlit**
  - Price Prediction Model
    - One or more features as an input
    - A single value as an output
      - Price itself
      - Difference in price

Model Development

Research

- Papers and Textbooks
- Other experienced teams



Summarize

- It will be a gigantic task to actually implement a model that outperform the market by focusing only on one specific token due to our model being OVERFIT.

Summarize

- Let’s imagine that there is an actual trend that could really provide profits to users. In order to gain the highest amount of profit, users have to buy low and sell high according to those trends.
- However, there are so many competitors within the field who will be using statistical and quant approaches to detect the trends.
- Therefore, there are a significant chances that different players could end up detecting the same price trend. Once the trends are detected, some users will frontrun others to sell first. Resulting in the trend to always be changing.

Summarize

- This means the trend of cryptocurrency prices might still be close to before. However, the change in trend could always be big enough to make the trained model perform worse than just simply holding tokens.

Our decision

- That’s why we shifted our attention toward building models to gain the most gains from shifting between tokens instead of focusing only on one token.

- For example, allocating our fund in day1 to token A and shift our fund in the next day to other tokens once the token A is predicted to have its value reduced.

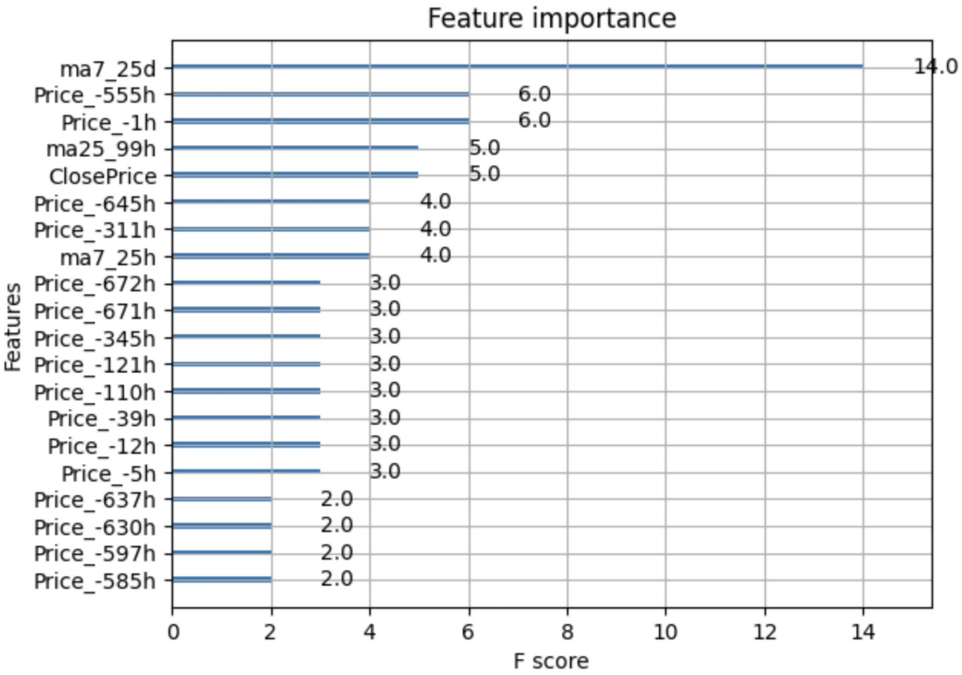
Preprocess

The indicators we experimented on include but are not limited to the followings:

- Close price (normalized)
- ma7h-ma25h
- ma25h-99h
- ma7d-ma25d
- ma25d-ma99d
- RSI
- MACD
- Every Hour of Historical Price (normalized)
    - Using data from the previous 1,2,3,4,..., 672 hours
- Bollinger Bands

Preprocess

Then, we plot out the feature significance of those indicators by training them with some of our models.



Preprocess

We summarize that the indicators that contribute the most to the model performances are the followings

- Close price (normalized)
- ma7h-ma25h
- ma25h-99h
- ma7d-ma25d
- Every 24 Hour Data of Historical Price (normalized)
  - Using data from the previous 1,25,49,73,..., 635, 672 hours

Model Selection

- We listed out some of the models that are convenient to be implemented. With more time provided, our team could experiment on more models and might receive better results

XGBoost
LassoLars
Bayesian Ridge
Ridge
Lasso
Elastic Net
Huber
CatBoost

Hypertuning models

- We adopted Optuna in hypertuning all our models.
- We set mean square error (MSE) as the objective function for training.
    - The fewer, the better

```
def objective_lars(trial):
    alpha = trial.suggest_float('alpha', 1e-6, 1.0)
    model = LassoLars(alpha=alpha)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    return mean_squared_error(y_test, y_pred)
```

Model Training

- We trained our model by having the training process recorded on MLFlow.

```
with mlflow.start_run(experiment_id=experiment_id) as run:
    best_model = Ridge(**study.best_params)
    best_model.fit(X_train, y_train)
    y_pred = best_model.predict(X_test)
```

Model Evaluation

Model Evaluation

- Evaluation Datasets
    - 20% of Historical price data of 194 currencies
    - Recent price data of BTCUSDT spanning from January 1st to January 31st 2024
- Evaluation Metrics
    - Pool Value
    - Classification Report
    - Actual vs Predicted Value Plot

Pool Value

- Uniquely designed metric that we designed ourselves
- Capable of telling whether the model can outperform the market.
- The logic underlying it is provided as following:
  - We divide our money into 24 parts.
  - Every start of an hour, we will predict which tokens will perform the best on the next 24 hours and we will allocate our money to it.
  - Total pool value represent the percent gains across all the pools.

```
pools = np.array([1.0] * 24)
pool_index = 0
for value1,value2 in zip(filtered_y_pred_data,filtered_y_test_data):
    # print(index,value)
    if value1 > 0.001:
        pools[pool_index] *= (1+value2)
        pool_index = (pool_index + 1) % 24  # Increment the pool index and loop back after the 24th

# Sum the values of all pools

total_pool_value = np.sum(pools)/24
```

Pool Value

- Higher pool value indicates better performance
  - Pool value is not ROI due to how some data from the test cases are from the same time and we can’t use our fund in different places at the same time.
- Pool value of a model surpasses that of the control groups(the pool value from just holding the tokens in all scenarios), indicating that some of our models are able to outperform the market.

Models	Pool Value
XGBoost	1.226×10^{49}
LassoLars	1.216×10^{-6}
Bayesian Ridge	1.661×10^{34}
Ridge	2.693×10^{36}
Lasso	1.516×10^{-6}
Elastic Net	1.516×10^{-6}
Huber	1.216×10^{30}
CatBoost	4.046×10^{125}
Control group	1.144×10^{-3}

Classification Report

- Evaluated using recent price data of BTCUSDT
- Positive values are classified as Class 1.
- Negative values are classified as Class 0.
- Class 1 precision indicates model accuracy in buying decisions.
- Class 1 recall indicates missed buying opportunities.
- Models can be grouped into 2 groups based on their performance.

Classification Report

- Group 1
  - Class 1 recall of 1 and class 0 recall of 0.
  - Indicates that the model only predict upward trends; therefore, these models should not be used.
  - The model in this group, including Lasso, LassoLars, and Elastic Net, displays similar performance.

Classification report for ElasticNet model

	precision	recall	f1-score	support
0	0.00	0.00	0.00	20458
1	0.53	1.00	0.69	22742
accuracy			0.53	43200
macro avg	0.26	0.50	0.34	43200
weighted avg	0.28	0.53	0.36	43200

Classification report for Lasso model

	precision	recall	f1-score	support
0	0.00	0.00	0.00	20458
1	0.53	1.00	0.69	22742
accuracy			0.53	43200
macro avg	0.26	0.50	0.34	43200
weighted avg	0.28	0.53	0.36	43200

Classification Report

- Group 2
  - The model in this group includes XGBoost, Bayesian Ridge, Ridge, Huber, and CatBoost.
  - Huber is the only model showing higher than 50% class 1 precision during the evaluation period.
  - Class 0 recall is significantly higher than the class 1 recall, which may implies the tendency to predict negative values for the model in this group.

Classification report for Huber model

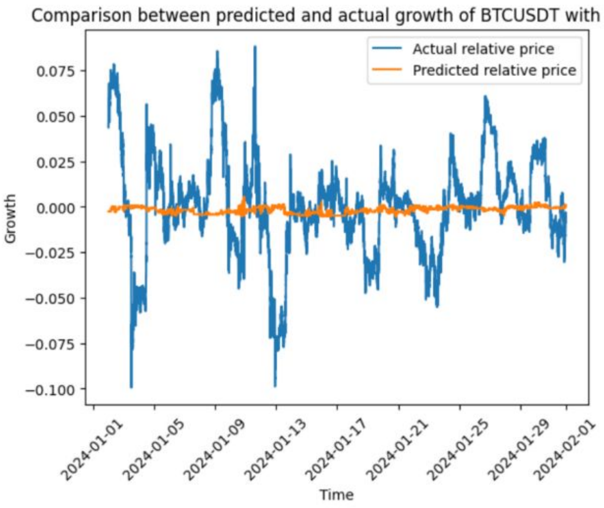
	precision	recall	f1-score	support
0	0.47	0.70	0.57	20458
1	0.53	0.30	0.38	22742
accuracy			0.49	43200
macro avg	0.50	0.50	0.47	43200
weighted avg	0.50	0.49	0.47	43200

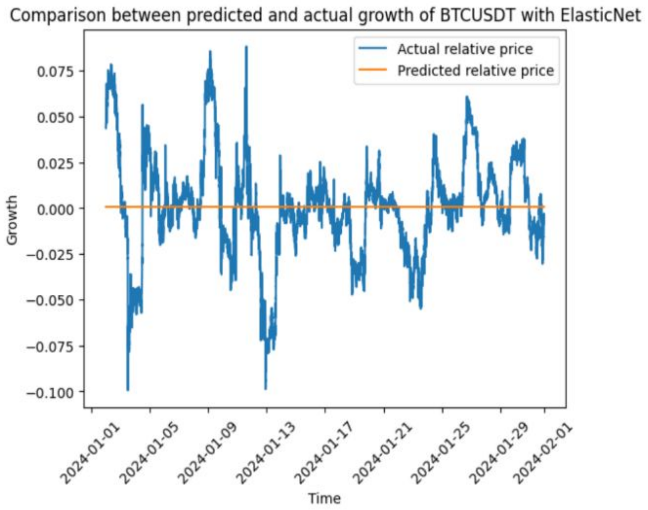
Actual vs Predicted Value Plot

- Evaluated using recent price data of BTCUSDT.
- The actual relative price and the predicted value are plotted in the same axis.
- Visualize if the model can capture the price trend.
- Requires human observation and judgement.
- Models can be grouped into 2 groups.

Actual vs Predecited Value Plot

- Group 1
  - The model in this group includes Lasso, LassoLars, Elastic Net, and XGBoost.
  - Grouping is similar to that of the classification report metric.
  - Fail to capture the price trend.
  - Most predicted values hovering around zero.



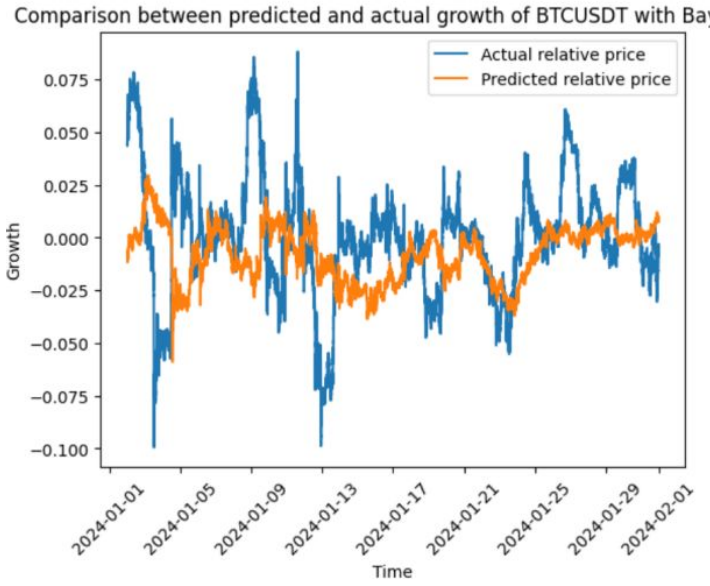


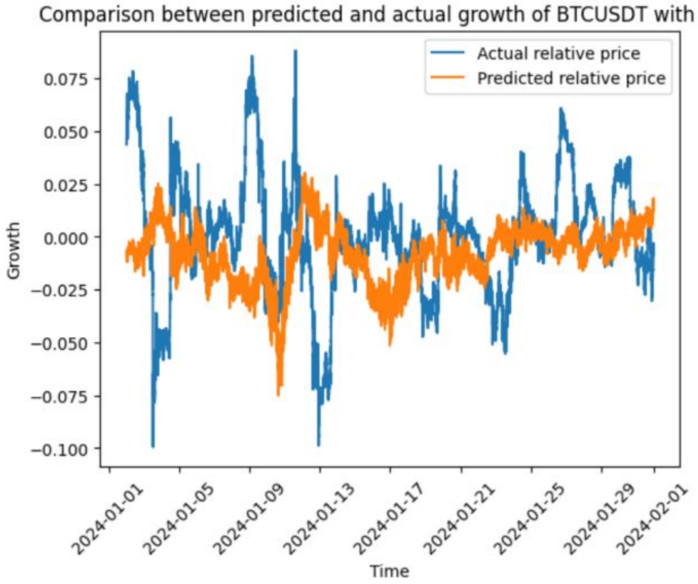
Actual vs Predecited Value Plot

- Group 2
  - The model in this group includes Bayesian Ridge, Ridge, Huber, and CatBoost.
  - Able to partially capture the price trend
  - With some limitations
    - Tendency to predict negative values aligning with speculation made when observing classification reports
    - Narrower ranges of predicted outputs
    - Some delay in the prediction trend compared to the actual trend

Actual vs Preditced Value Plot

- Group 2
    - Bayesian Ridge, Ridge, and Huber predict very similar trends, while CatBoost predict a slightly different trend.





Discussion

- When market conditions resemble those of the train dataset
  - CatBoost emerges as the best-performing model.
- With current market conditions
  - Huber emerges as the best-performing model.
- There is a performance degradation problem on most of the models.
  - Class 1 precision falls below 50% for many models.
  - Hard to gain profits with this precision.
- The Huber model is either more tolerant to performance degradation or coincidentally performs well in the current market condition.

Discussion

- Overall model performance can be further improved.
    - While the models are able to partially capture the underlying trend, they struggle to predict accurate values.
    - There is a tendency to predict values in a certain class.
- The selected model should be able to perform acceptably in the current market condition.
    - Class 1 precision is at 53%.
    - The model can make accurate buying decisions when the model predicts positive outcomes.

Discussion

- Reasons that factors to the underperformance of the model are speculated as follows
  - Using several currencies: Although the approach is taken to reduce the chance of overfitting, it may come with trade off as each currency may have different natures.
  - Constantly changing of market conditions: This is evident by the degradation in performance of the models.
  - Cryptocurrency is a highly voltaire market: Simple models may not be able to fully capture this voltaire nature.
  - MSE might not be the most appropriate metric to be used for the optimation: Other metrics should also be considered.

Conclusion

Assessment: System

- Successfully implement a minimum viable product(mvp) of the E2E ML system
- Weakness
    - Data Ingestion and Transformation can be indeed implemented on Mage.ai
        - However, also want to demonstrate serverless method
    - Use multiple cloud providers
        - Cannot fully integrate AWS features to DigitalOcean resources
    - Alias, allowing immediate model changing, requires the model to be downloaded every time.
        - Consume a large amount of memory
        - Still be a problem of our APIs
        - Maybe because we still don’t fully understand how BentoML works
            - Can’t properly optimized

Next Step: System

- Implement an automated model training pipeline and refine other components to fully fulfill Microsoft’s MLOps Level 2
- Find a method to improve the manual method to handle the historical data.
- Store timezone information along with datetime
    - May have a problem with daylight saving time
- A method of always downloading the model should be refined
- Grafana for model and API monitoring instead of using Streamlit

Assessment: Model

- There are room for further improvement as evident by the evaluation result using the recent data
- The best performing model should outperform the market in the current condition.
- In the lab environment, several models have successfully outperformed the market.

Next Step: Model

- Several potential methods for improving model performance and robustness are proposed as follows:
  - Probabilistic forecasting models, including AR, ARMA, and ARIMA, can be alternative solutions.
  - Deep Neural Networks (DNNs) are renowned for their ability to capture highly complex relationships.
  - The problem can be framed as a binary classification problem, opening up other objective functions for model optimization.

Q & A