Arjun Anand Mallya 23110039

Venkatakrishnan E 23110357

Importing the Breast cancer Dataset

main.ipynb | workflow.py | serving.py | secret.txt | load_data.py | trainer.py

```python
import mlrun
from sklearn.datasets import load_breast_cancer
import pandas as pd

@mlrun.handler(outputs=["dataset", "label_column"])
def breast_cancer_loader(context, format="csv"):

    data = load_breast_cancer(as_frame=True)
    breast_cancer_df = data.frame
    breast_cancer_df['target'] = data.target

    context.logger.info('Saving breast cancer dataset to {}'.format(context.artifact_path))
    context.log_dataset('breast_cancer_dataset', df=breast_cancer_df, format=format, index=False)

    return breast_cancer_df, "target"

if __name__ == "__main__":
    with mlrun.get_or_create_ctx("breast_cancer_loader", upload_artifacts=True) as context:
        breast_cancer_loader(context, context.get_param("format", "csv"))
```

Splitting the data into train/test (Trainer.py)

The dataset was split into 90% train and 10% split. The dataset was loaded using ML Run and prepared for training. It trains a Random Forest classifier.

main.ipynb | workflow.py | serving.py | secret.txt | load_data.py | trainer.py

```python
import mlrun
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from mlrun.frameworks.sklearn import apply_mlrun

def train(
    dataset: mlrun.DataItem,
    label_column: str = 'target',
    n_estimators: int = 100,
    max_depth: int = None,
    model_name: str = "breast_cancer_classifier"
):
    df = dataset.as_df()
    X = df.drop(label_column, axis=1)
    y = df[label_column]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
    model = RandomForestClassifier(n_estimators=n_estimators, max_depth=max_depth, random_state=42)
    apply_mlrun(model=model, model_name=model_name, x_test=X_test, y_test=y_test)
    model.fit(X_train, y_train)
```

Serving.py

The Classifier Model inherits from **mlrun.serving.V2ModelServer.** Thus, we can load the model from storage and be able to predict according to requests.

```python
from cloudpickle import load
import numpy as np
from typing import List
import mlrun

class ClassifierModel(mlrun.serving.V2ModelServer):
    def load(self):

        model_file, extra_data = self.get_model('.pkl')
        self.model = load(open(model_file, 'rb'))

    def predict(self, body: dict) -> List:

        feats = np.asarray(body['inputs'])
        results: np.ndarray = self.model.predict(feats)
        return results.tolist()
```

Workflow.py

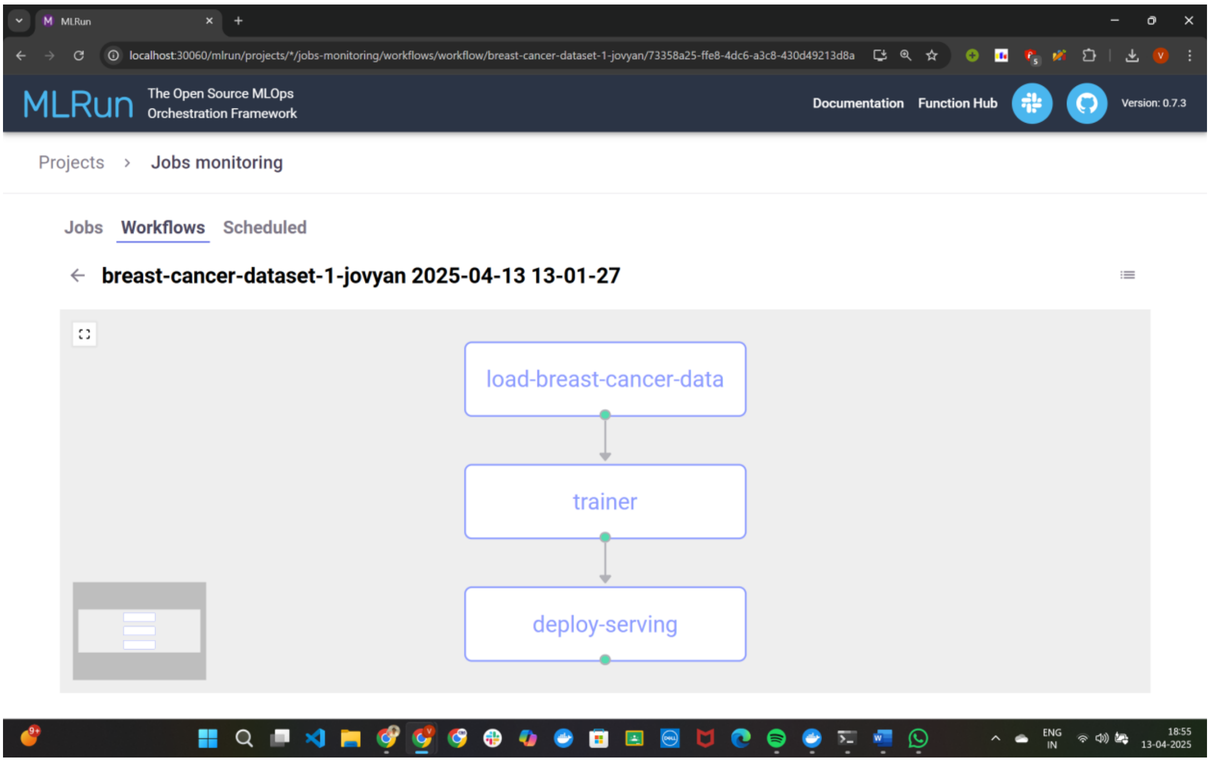
Data Ingestion:

The **@dsl.pipeline** decorator. For data ingestion, the **load-data** function from **load_data.py** is ran and the output is stored in **dataset.** The **trainer** function in **trainer.py** is used to experiment with different hyperparameters like number of estimators, and max depth. The **max_accuracy** selector chooses the model with the best accuracy, effectively performing a grid search over the hyperparameters. The output is stored in **model.** Next, the model is deployed using the ClassifierModel in **serving.py.**

main.ipynb | workflow.py | serving.py | project.yaml | secret.txt | load_data.py | trainer.py

```python
import mlrun
from kfp import dsl

@dsl.pipeline(name="breast-cancer-pipeline")
def pipeline(model_name="breast_cancer_classifier"):

    ingest = mlrun.run_function(
        "load-data",
        name="load-breast-cancer-data",
        params={"format": "pq", "model_name": model_name},
        outputs=["dataset"],
    )

    train = mlrun.run_function(
        "trainer",
        inputs={"dataset": ingest.outputs["dataset"]},
        hyperparams={
            "n_estimators": [10, 100, 200],
            "max_depth": [2, 5, 10]
        },
        selector="max.accuracy",
        outputs=["model"],
    )

    deploy = mlrun.deploy_function(
        "serving",
        models=[{"key": model_name, "model_path": train.outputs["model"], "class_name": "ClassifierModel"}],
        mock=False
    )


```

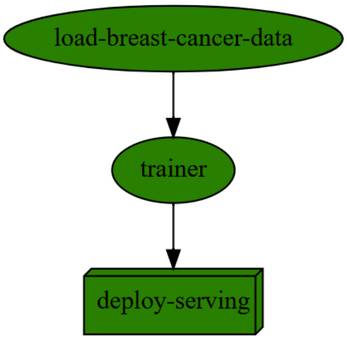
Workflow Graph



click here to view progress

Pipeline running (id=73358a25-ffe8-4dc6-a3c8-430d49213d8a), **click here** to view the details in MLRun UI

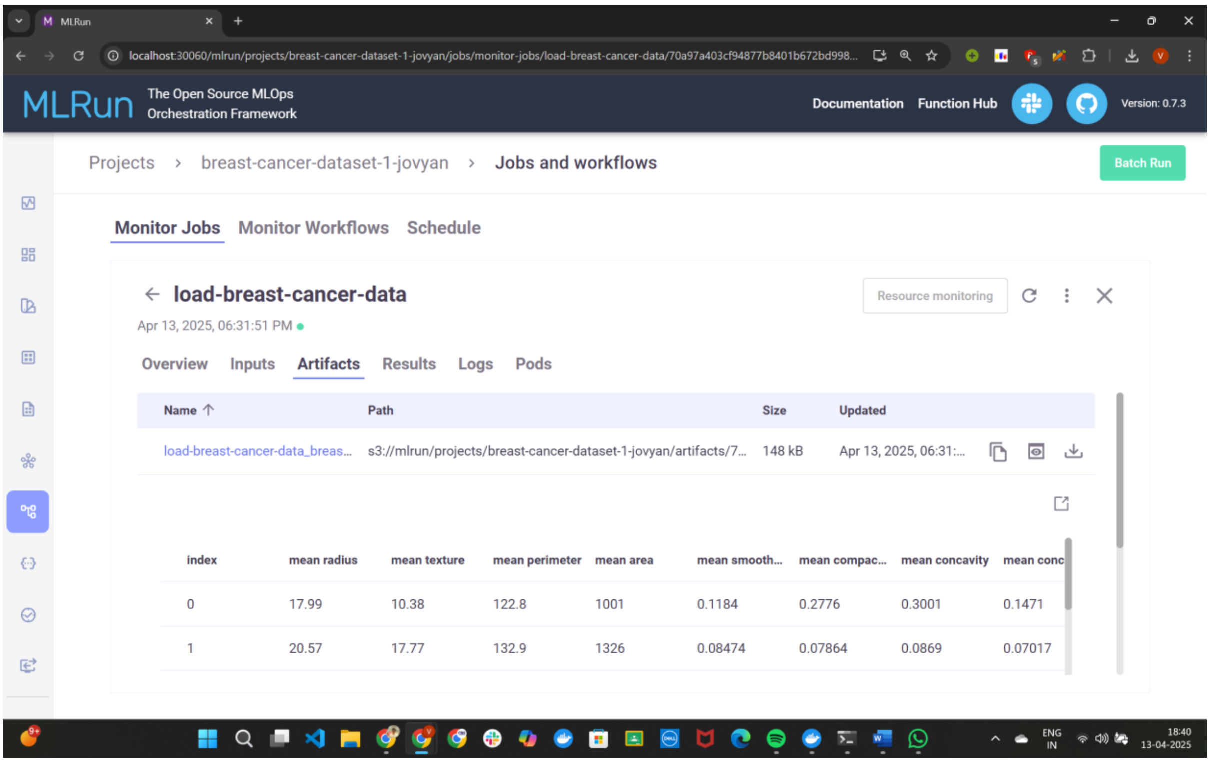
```
> 2025-04-13 13:01:28,393 [info] Started run workflow breast-cancer-dataset-1-jovyan with run id = '73358a25-ffe8-4dc6-a3c8-430d49213d8a' by kfp engine
> 2025-04-13 13:01:28,394 [info] Waiting for pipeline run completion: {"project":"breast-cancer-dataset-1-jovyan","run_id":"73358a25-ffe8-4dc6-a3c8-430d49213d8a"}
```

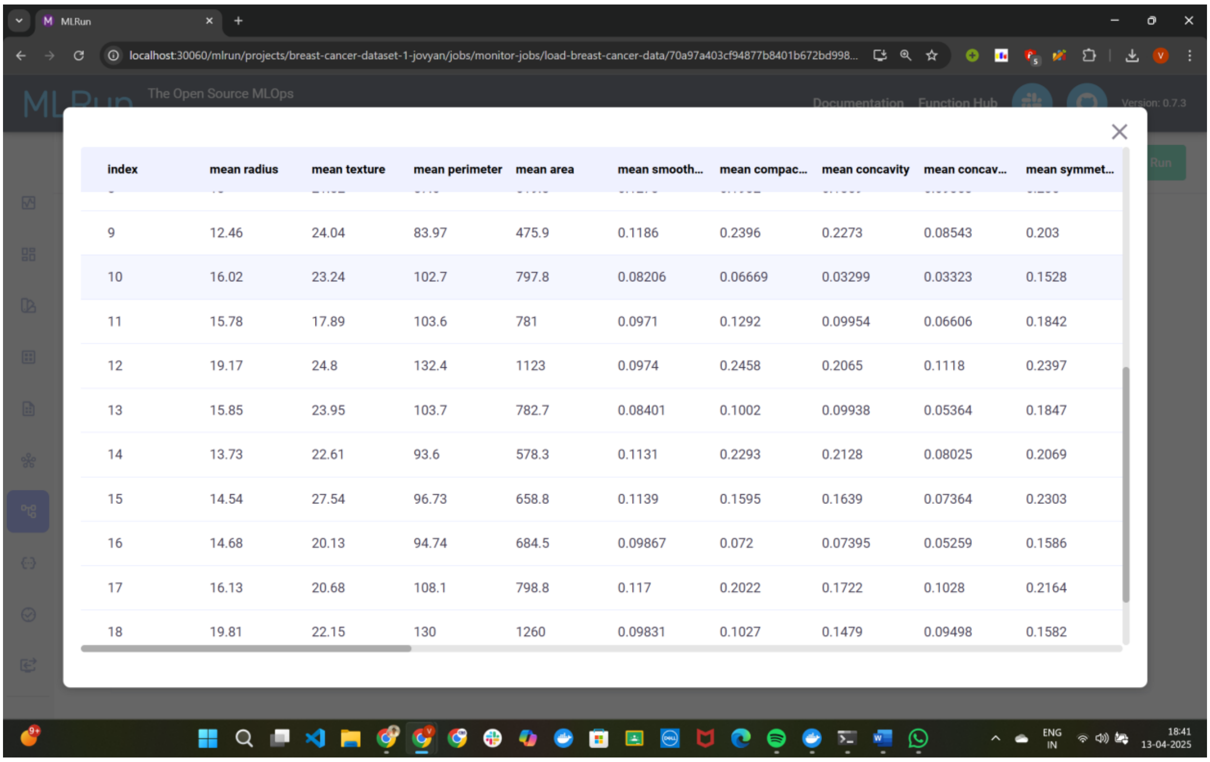


click the hyper links below to see detailed results

uid	start	state	kind	name	parameters	results
...94a426aa	Apr 13 13:02:12	completed	run	trainer		best_iteration=1 accuracy=0.9649122807017544 f1_score=0.975 precision_score=0.975 recall_score=0.975
...2bd998a9	Apr 13 13:01:48	completed	run	load-breast-cancer-data	format=pq model_name=breast-cancer-classifier	label_column=target

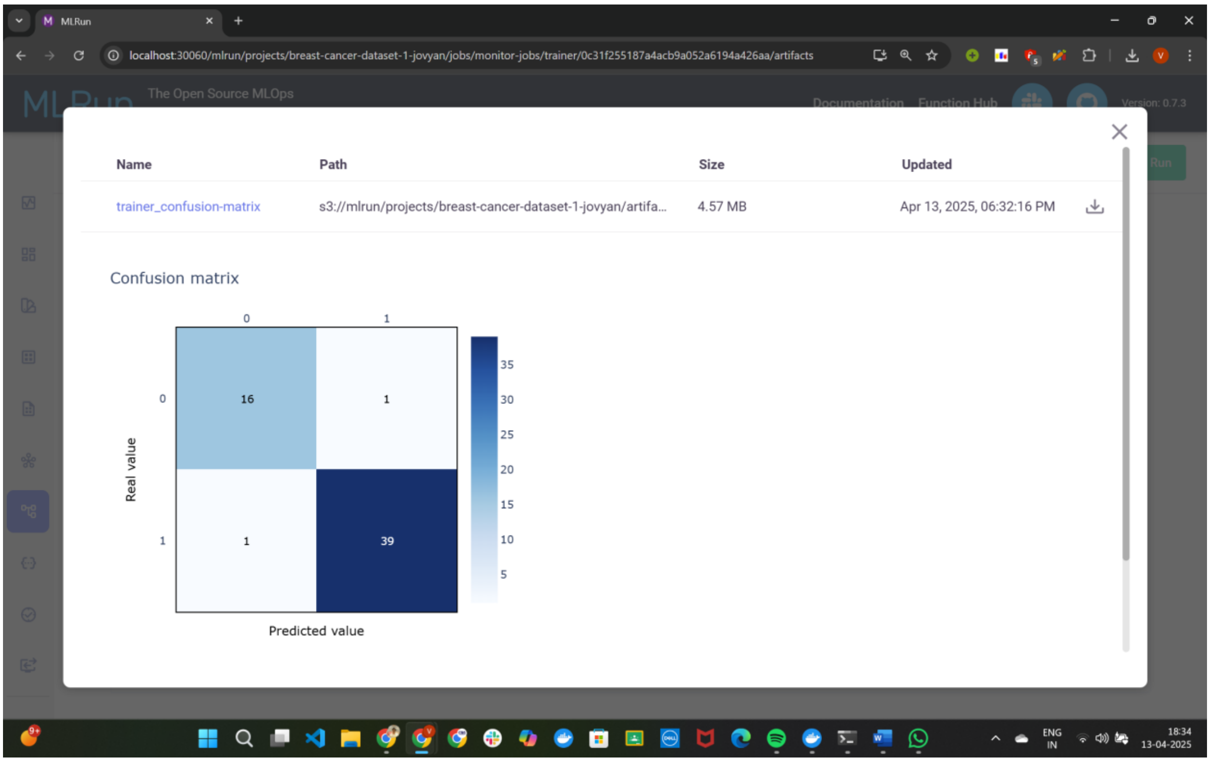
Data_prep artifact



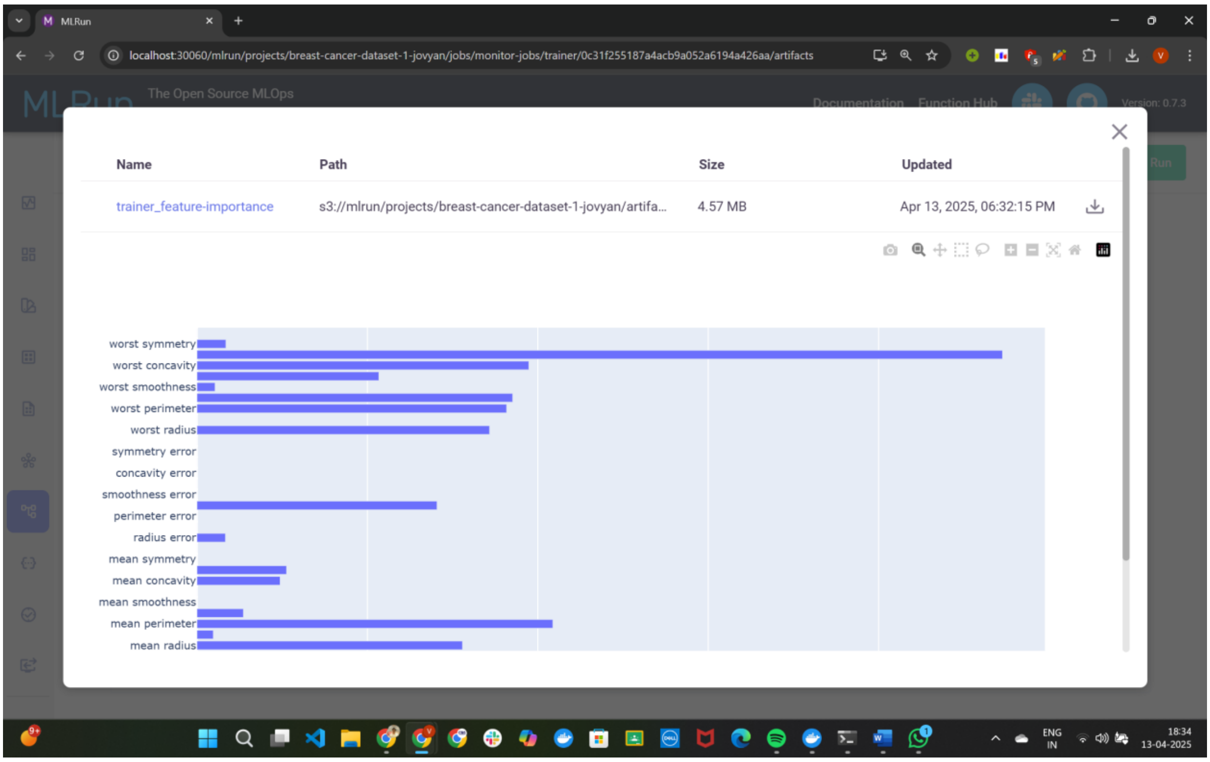


index	mean radius	mean texture	mean perimeter	mean area	mean smooth...	mean compac...	mean concavity	mean concav...	mean symmet...
9	12.46	24.04	83.97	475.9	0.1186	0.2396	0.2273	0.08543	0.203
10	16.02	23.24	102.7	797.8	0.08206	0.06669	0.03299	0.03323	0.1528
11	15.78	17.89	103.6	781	0.0971	0.1292	0.09954	0.06606	0.1842
12	19.17	24.8	132.4	1123	0.0974	0.2458	0.2065	0.1118	0.2397
13	15.85	23.95	103.7	782.7	0.08401	0.1002	0.09938	0.05364	0.1847
14	13.73	22.61	93.6	578.3	0.1131	0.2293	0.2128	0.08025	0.2069
15	14.54	27.54	96.73	658.8	0.1139	0.1595	0.1639	0.07364	0.2303
16	14.68	20.13	94.74	684.5	0.09867	0.072	0.07395	0.05259	0.1586
17	16.13	20.68	108.1	798.8	0.117	0.2022	0.1722	0.1028	0.2164
18	19.81	22.15	130	1260	0.09831	0.1027	0.1479	0.09498	0.1582

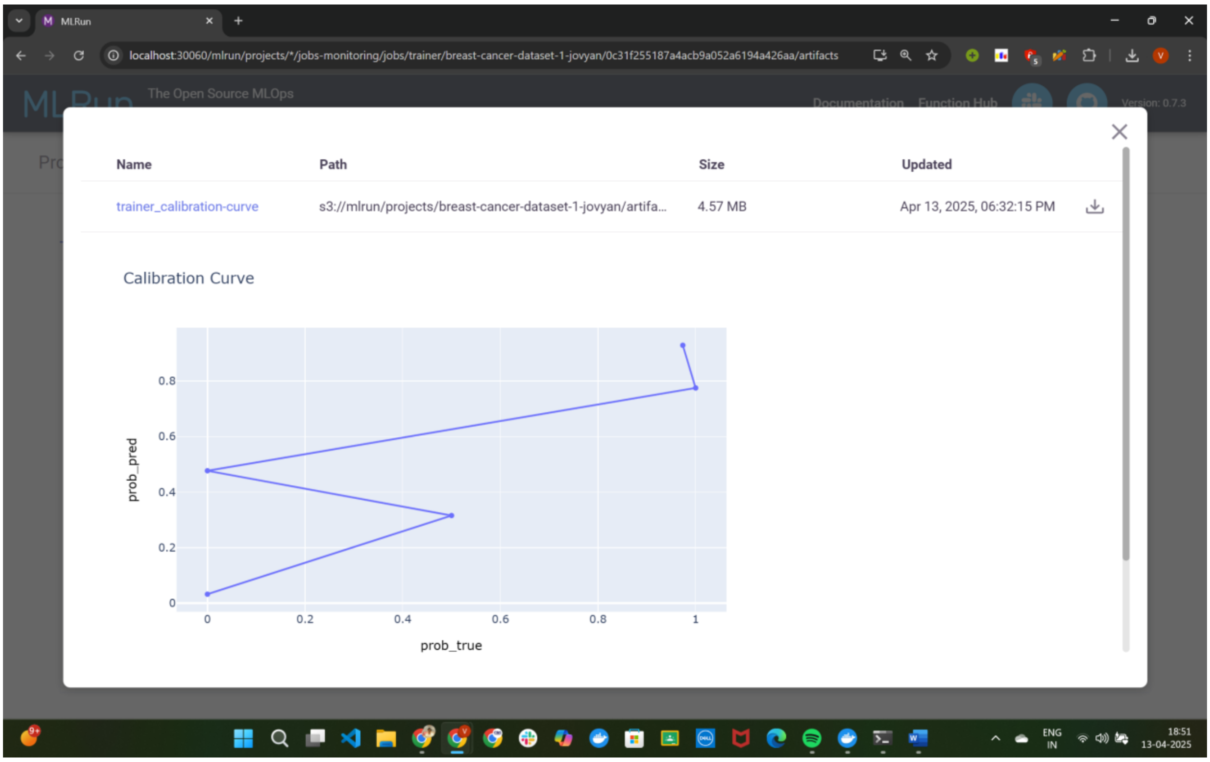
Confusion Matrix



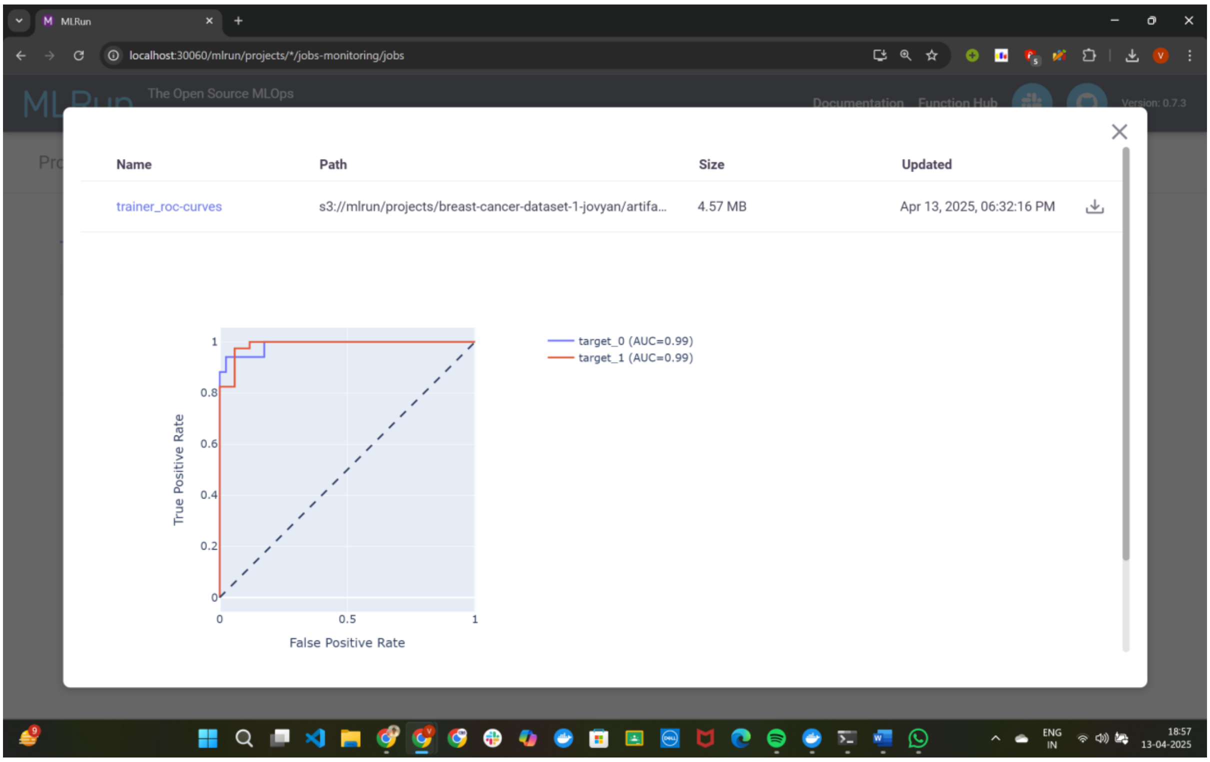
Feature Selection Artifact



Calibration Curve



Trainer – ROC Curves



Project.yaml

main.ipynb | workflow.py | serving.py | secret.txt | project.yaml | load_data.py | trainer.py


```yaml
metadata:
  name: breast-cancer-dataset-1-jovyan
spec:
  functions:
  - url: load_data.py
    name: load-data
    kind: job
    image: mlrun/mlrun
  - url: trainer.py
    name: trainer
    kind: job
    image: mlrun/mlrun
    handler: train
  - url: serving.py
    name: serving
    kind: serving
    image: mlrun/mlrun
  origin_url: git://Venkat-2341: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@github.com/Venkat-2341/lab9.git#refs/heads/master
  conda: ''
  source: git://Venkat-2341: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@github.com/Venkat-2341/lab9.git#refs/heads/master
  desired_state: online
kind: project
```