Lab Experiment 1

MTech in Applied AI
Deployments of ML Models

by

Sanjeev Kumar Pandey[*]
Faculty: Dr Ajay

Submitted December 29, 2024

[*] Student ID: 31050, Enrolment No:MT23AAI001, Email: `pandey.sanjeev@yahoo.com`

Table of Contents

1 Question : Hyperparameter Tuning and MLflow Integration

Objective: Showcase your skills in optimizing model performance through hyperparameter tuning and tracking experiments using MLflow.

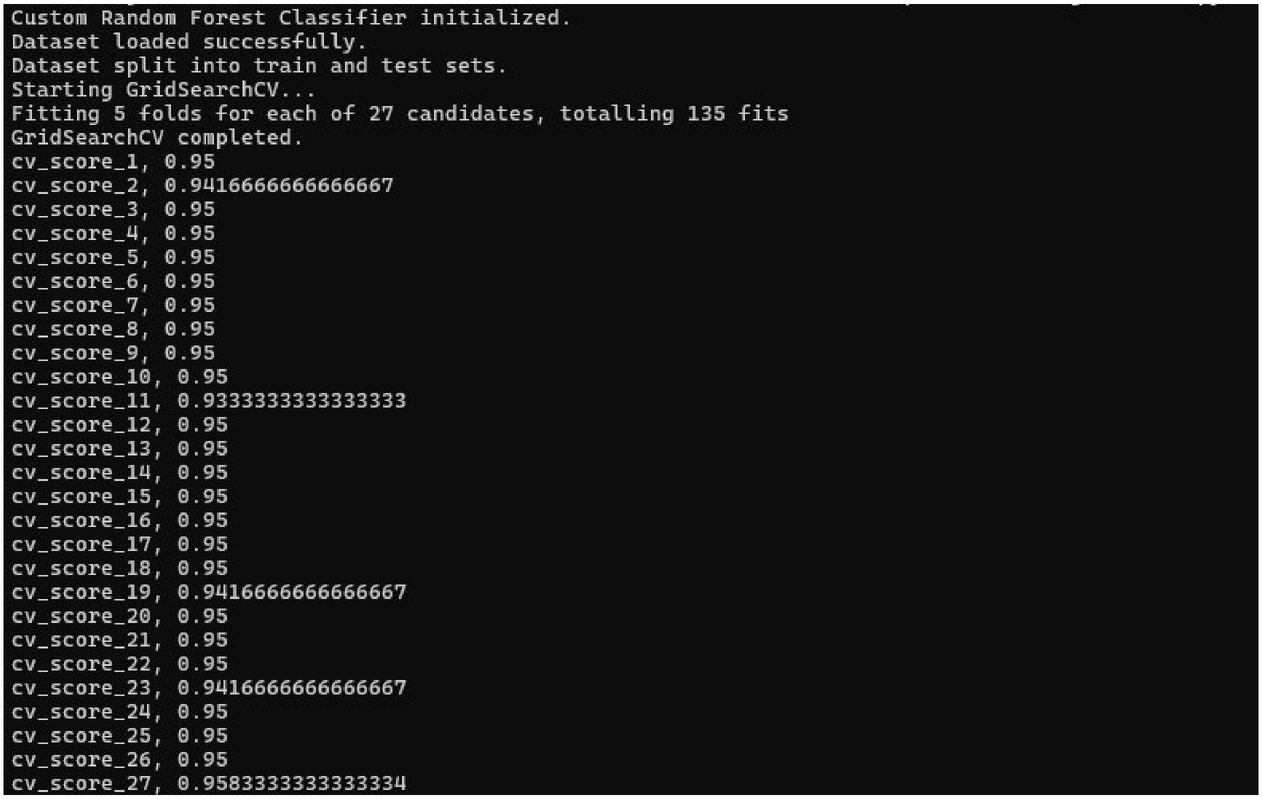
1.1 Code

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pandas as pd
import mlflow.sklearn
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report


class GridSearchCVImpl:
# Step 0: Initialize
    def __init__(self):
    # a) Features and Targets
        self.X = None
        self.y = None

    # b) Define train and test
        self.X_train = self.X_test = self.y_train = self.y_test = None
        self.y_pred = None

    # c) Define the Random Forest Classifier
        self.mrfc = RandomForestClassifier()
        print("Custom Random Forest Classifier initialized.")

    # d) 1. Parameter Grid Definition:
        self.param_grid = {
            ’n_estimators’: [50, 100, 150],
            ’max_depth’: [None, 10, 20],
            ’min_samples_split’: [2, 5, 10]
        }

    # e) define grid_search
        self.grid_search = None

# Step 1: Load the dataset
    def load_dataset(self):
        try:
            my_df = pd.read_csv("Iris.csv")
            mapping = {’Iris-setosa’: 0.0, ’Iris-versicolor’: 1.0, ’Iris-virginica’: 2.0}
            my_df[’Species’] = my_df[’Species’].map(mapping).astype(float)

            # X is feature so dropping the Y from the dataframe
            # ensure to take only 4 features as model is accepting only 4 features
            self.X = my_df[[’SepalLengthCm’, ’SepalWidthCm’, ’PetalLengthCm’, ’PetalWidthCm’]]
            self.y = my_df[’Species’]
            print("Dataset loaded successfully.")
        except Exception as e:
            print(f"Error loading dataset: {e}")
            exit()

# Step 2: Split the dataset
    def split_dataset(self):
        try:
            X = self.X.values
            y = self.y.values
            self.X_train, self.X_test, self.y_train, self.y_test = (
                train_test_split(X, y, test_size=0.2, random_state=42)
            )
            print("Dataset split into train and test sets.")
        except Exception as e:
            print(f"Error splitting dataset: {e}")
            exit()

# Step 3: Set up and run GridSearchCV
    def execute_gridsearchcv(self):
        try:
            print("Starting GridSearchCV...")
            self.grid_search = GridSearchCV(
                estimator=self.mrfc,
                param_grid=self.param_grid,
                scoring=’accuracy’,
                cv=5,  # 5-fold cross-validation    2. Grid Search Implementation:
                n_jobs=-1,  # Use all CPU cores
                verbose=1   # Display progress
            )
            self.grid_search.fit(self.X_train, self.y_train)
            print("GridSearchCV completed.")
        except Exception as e:
            print(f"Error during GridSearchCV: {e}")
            exit()

# Step 4: Get the best hyperparameters and evaluate
    def eval_and_report(self):
        try:
            print(f"Best Hyperparameters:{self.grid_search.best_params_}")
            print(f"Best Cross-Validation Score:{self.grid_search.best_score_}")

            # Evaluate on the test set
            best_model = self.grid_search.best_estimator_
            self.y_pred = best_model.predict(self.X_test)

            print(f"Actual : Predicted")
            for i in range(len(self.y_test)):
                print(f"{self.y_test[i]}     : {self.y_pred[i]}")
            accuracy = accuracy_score(self.y_test, self.y_pred)
            print("Test Set Accuracy:", accuracy)

            # Print classification report
            print("Classification Report:")
            print(classification_report(self.y_test, self.y_pred))
        except Exception as e:
            print(f"Error during evaluation: {e}")
            exit()


# Set up experiment tracking
mlflow.set_experiment(’RandomForest_Hyperparameter_Tuning’)

# Start an MLflow run
with mlflow.start_run():
    gscvi = GridSearchCVImpl()
    gscvi.load_dataset()
    gscvi.split_dataset()

    # Log hyperparameter grid
    mlflow.log_params({
        "n_estimators_range": gscvi.param_grid[’n_estimators’],
        "max_depth_range": gscvi.param_grid[’max_depth’],
        "min_samples_split_range": gscvi.param_grid[’min_samples_split’]
    })

    # Execute GridSearchCV
```

```python
    gscvi.execute_gridsearchcv()

    # Log the best hyperparameters and the best score
    mlflow.log_params(gscvi.grid_search.best_params_)
    mlflow.log_metric("best_cv_accuracy", gscvi.grid_search.best_score_)

    # Log cross-validation scores
    for i, score in enumerate(gscvi.grid_search.cv_results_['mean_test_score']):
        mlflow.log_metric(f'cv_score_{i + 1}', score)
        print(f"cv_score_{i + 1}, {score}")

    # Log the best model
    input_example = gscvi.X_train[0].reshape(1, -1)  # Example input for reproducibility
    mlflow.sklearn.log_model(gscvi.grid_search.best_estimator_, "best_random_forest_model",
                             input_example=input_example)

    # Log test accuracy
    gscvi.eval_and_report()
    test_accuracy = accuracy_score(gscvi.y_test, gscvi.y_pred)
    mlflow.log_metric("test_accuracy", test_accuracy)
```

Listing 1: Code

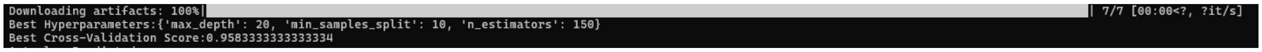
1.2 Output



```
Custom Random Forest Classifier initialized.
Dataset loaded successfully.
Dataset split into train and test sets.
Starting GridSearchCV...
Fitting 5 folds for each of 27 candidates, totalling 135 fits
GridSearchCV completed.
cv_score_1, 0.95
cv_score_2, 0.9416666666666667
cv_score_3, 0.95
cv_score_4, 0.95
cv_score_5, 0.95
cv_score_6, 0.95
cv_score_7, 0.95
cv_score_8, 0.95
cv_score_9, 0.95
cv_score_10, 0.95
cv_score_11, 0.9333333333333333
cv_score_12, 0.95
cv_score_13, 0.95
cv_score_14, 0.95
cv_score_15, 0.95
cv_score_16, 0.95
cv_score_17, 0.95
cv_score_18, 0.95
cv_score_19, 0.9416666666666667
cv_score_20, 0.95
cv_score_21, 0.95
cv_score_22, 0.95
cv_score_23, 0.9416666666666667
cv_score_24, 0.95
cv_score_25, 0.95
cv_score_26, 0.95
cv_score_27, 0.9583333333333334
```

Figure 1: op-1

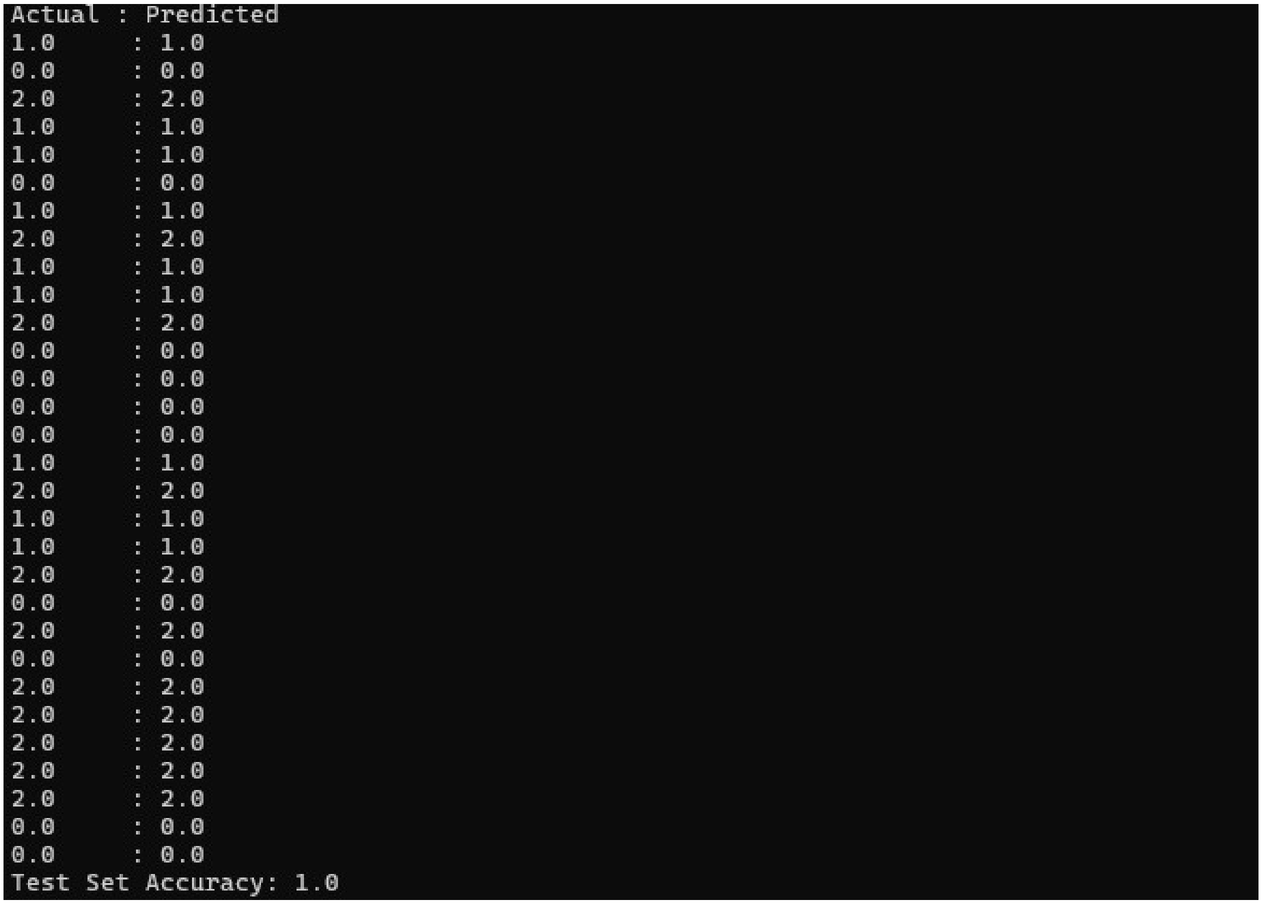
1.3 Output



```
Downloading artifacts: 100%|████████████████████████████████████████| 7/7 [00:00<?, ?it/s]
Best Hyperparameters:{'max_depth': 20, 'min_samples_split': 10, 'n_estimators': 150}
Best Cross-Validation Score:0.9583333333333334
```

Figure 2: op-2

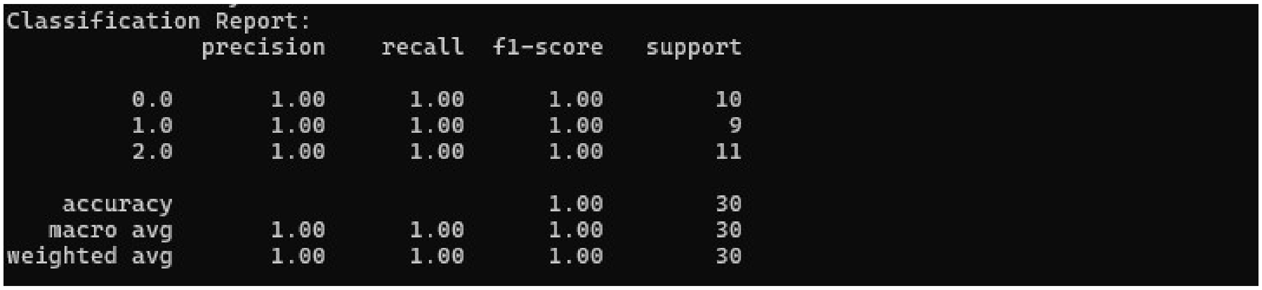
1.4 Output



```
Actual : Predicted
1.0     : 1.0
0.0     : 0.0
2.0     : 2.0
1.0     : 1.0
1.0     : 1.0
0.0     : 0.0
1.0     : 1.0
2.0     : 2.0
1.0     : 1.0
1.0     : 1.0
2.0     : 2.0
0.0     : 0.0
0.0     : 0.0
0.0     : 0.0
0.0     : 0.0
1.0     : 1.0
2.0     : 2.0
1.0     : 1.0
1.0     : 1.0
2.0     : 2.0
0.0     : 0.0
2.0     : 2.0
0.0     : 0.0
2.0     : 2.0
2.0     : 2.0
2.0     : 2.0
2.0     : 2.0
2.0     : 2.0
0.0     : 0.0
0.0     : 0.0
Test Set Accuracy: 1.0
```

Figure 3: op-3

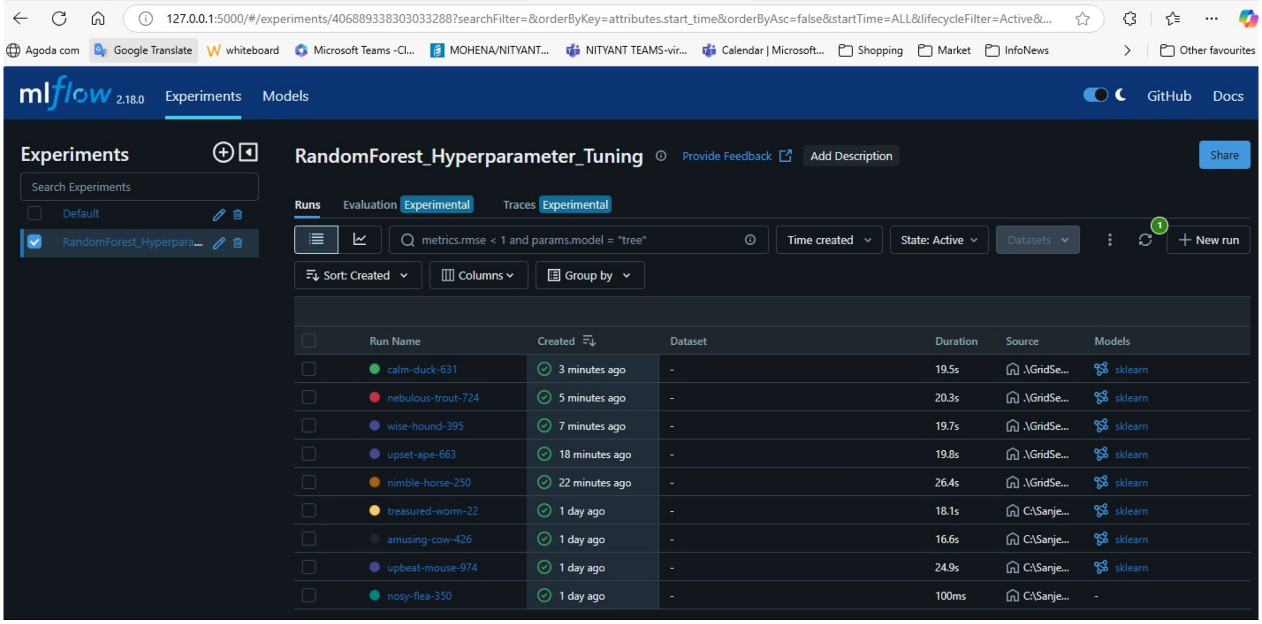
1.5 Output



```
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        10
         1.0       1.00      1.00      1.00         9
         2.0       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

Figure 4: op-4

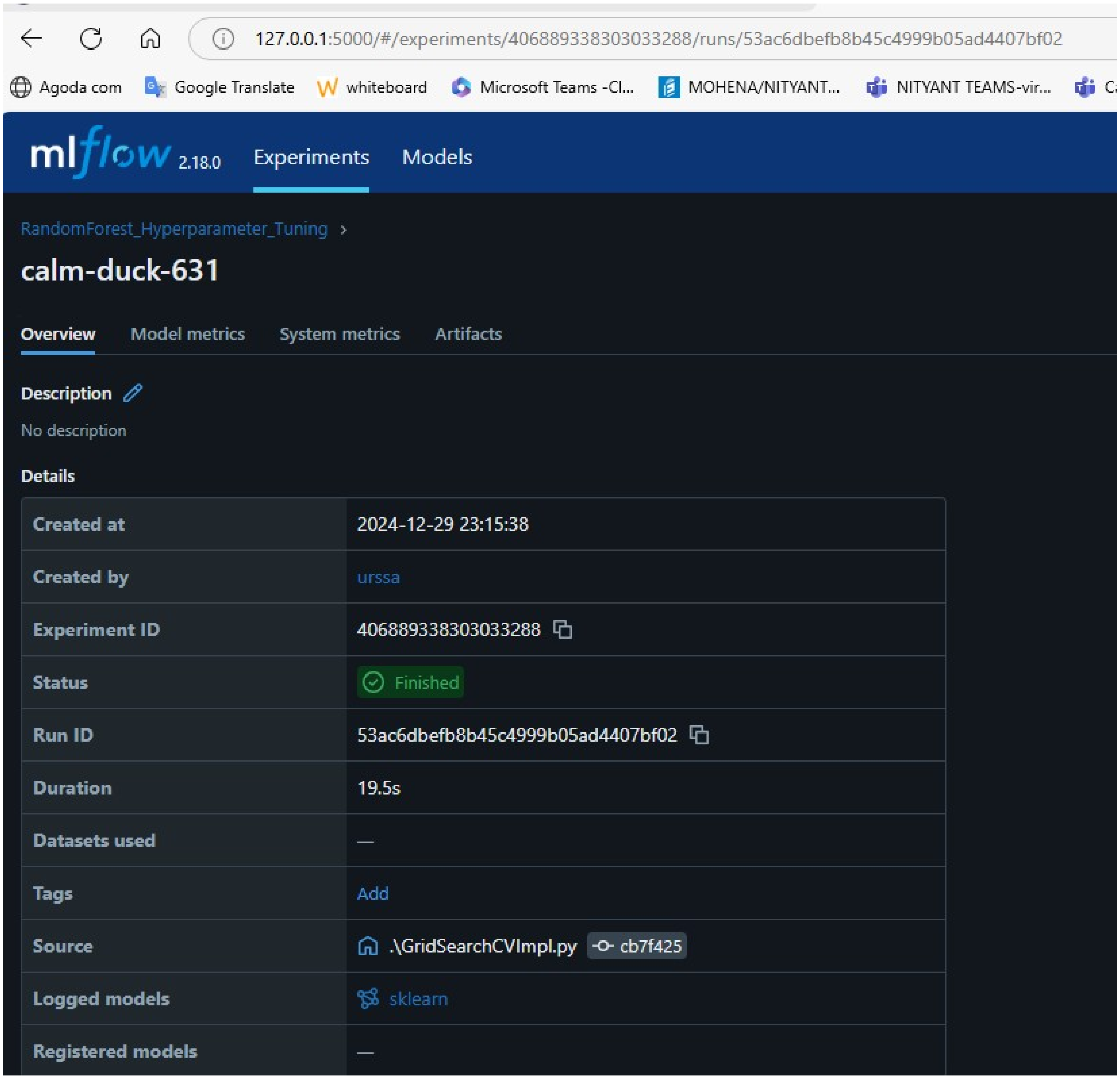
1.6 Output



□	Run Name	Created	Dataset	Duration	Source	Models
□	calm-duck-631	3 minutes ago	-	19.5s	.\GridSe...	sklearn
□	nebulous-trout-724	5 minutes ago	-	20.3s	.\GridSe...	sklearn
□	wise-hound-395	7 minutes ago	-	19.7s	.\GridSe...	sklearn
□	upset-ape-663	18 minutes ago	-	19.8s	.\GridSe...	sklearn
□	nimble-horse-250	22 minutes ago	-	26.4s	.\GridSe...	sklearn
□	treasured-worm-22	1 day ago	-	18.1s	C:\Sanje...	sklearn
□	amusing-cow-426	1 day ago	-	16.6s	C:\Sanje...	sklearn
□	upbeat-mouse-974	1 day ago	-	24.9s	C:\Sanje...	sklearn
□	nosy-flea-350	1 day ago	-	100ms	C:\Sanje...	-

Figure 5: op-5

1.7 Output



calm-duck-631

Description

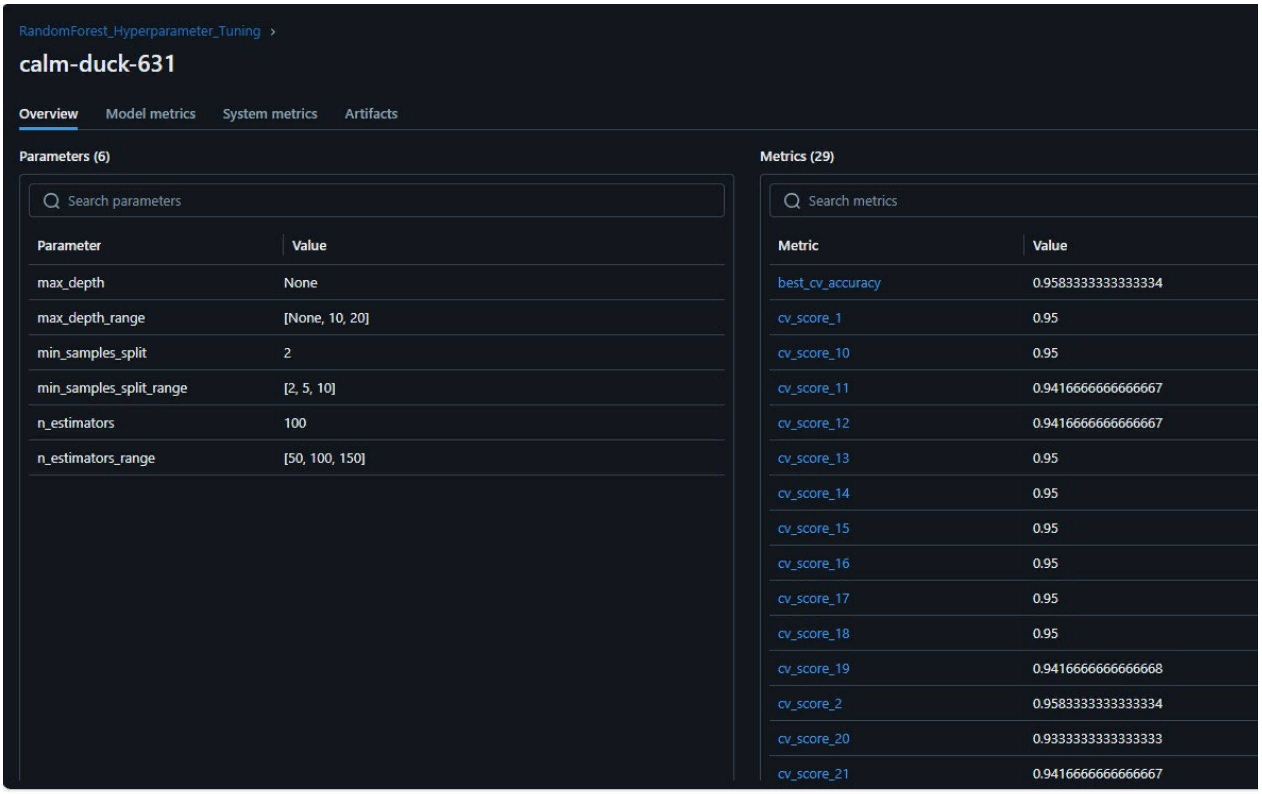
No description

Details

Created at	2024-12-29 23:15:38
Created by	urssa
Experiment ID	406889338303033288
Status	Finished
Run ID	53ac6dbefb8b45c4999b05ad4407bf02
Duration	19.5s
Datasets used	—
Tags	Add
Source	.\GridSearchCVImpl.py cb7f425
Logged models	sklearn
Registered models	—

Figure 6: op-6

1.8 Output



Parameters (6)

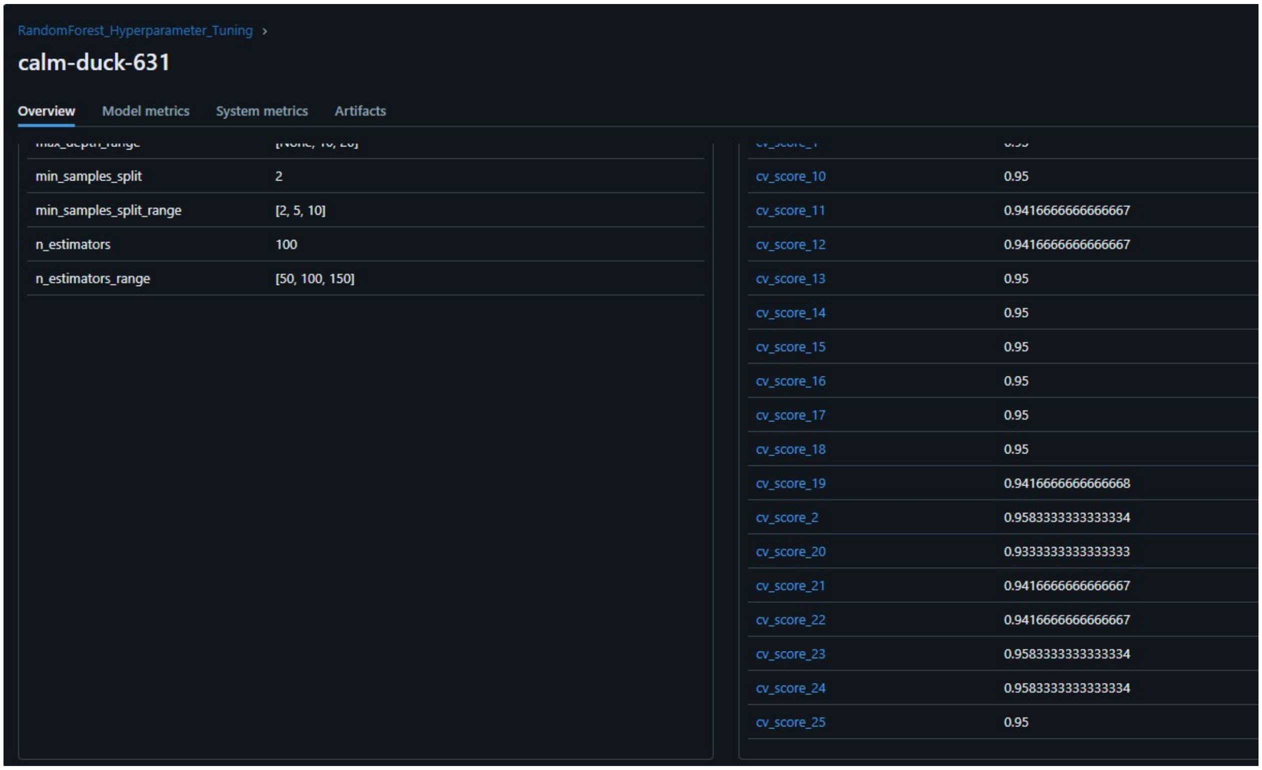
Parameter	Value
max_depth	None
max_depth_range	[None, 10, 20]
min_samples_split	2
min_samples_split_range	[2, 5, 10]
n_estimators	100
n_estimators_range	[50, 100, 150]

Metrics (29)

Metric	Value
best_cv_accuracy	0.9583333333333334
cv_score_1	0.95
cv_score_10	0.95
cv_score_11	0.9416666666666667
cv_score_12	0.9416666666666667
cv_score_13	0.95
cv_score_14	0.95
cv_score_15	0.95
cv_score_16	0.95
cv_score_17	0.95
cv_score_18	0.95
cv_score_19	0.9416666666666668
cv_score_2	0.9583333333333334
cv_score_20	0.9333333333333333
cv_score_21	0.9416666666666667

Figure 7: op-7

1.9 Output



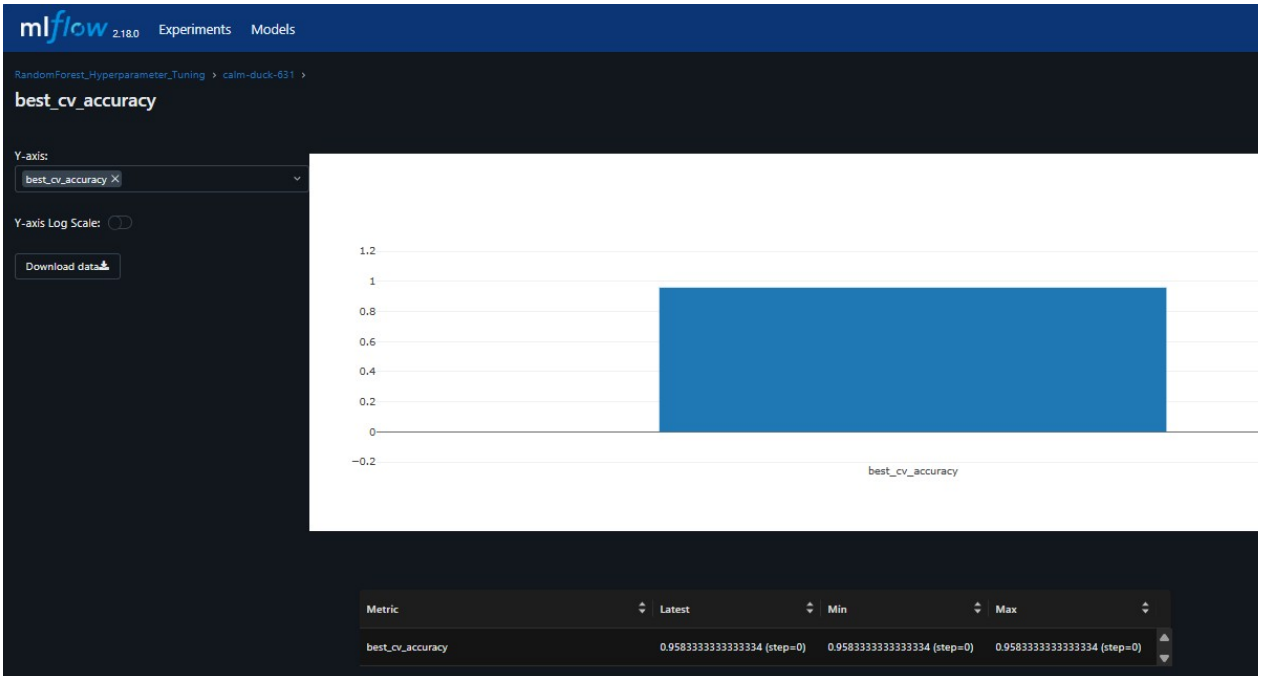
calm-duck-631

max_depth_range	[None, 10, 20]
min_samples_split	2
min_samples_split_range	[2, 5, 10]
n_estimators	100
n_estimators_range	[50, 100, 150]

cv_score_1	0.95
cv_score_10	0.95
cv_score_11	0.9416666666666667
cv_score_12	0.9416666666666667
cv_score_13	0.95
cv_score_14	0.95
cv_score_15	0.95
cv_score_16	0.95
cv_score_17	0.95
cv_score_18	0.95
cv_score_19	0.9416666666666668
cv_score_2	0.9583333333333334
cv_score_20	0.9333333333333333
cv_score_21	0.9416666666666667
cv_score_22	0.9416666666666667
cv_score_23	0.9583333333333334
cv_score_24	0.9583333333333334
cv_score_25	0.95

Figure 8: op-8

1.10 Output

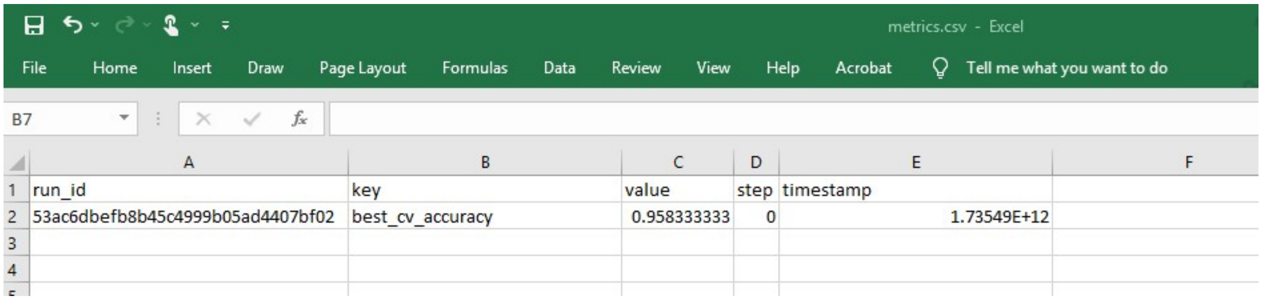


best_cv_accuracy

Metric	Latest	Min	Max
best_cv_accuracy	0.9583333333333334 (step=0)	0.9583333333333334 (step=0)	0.9583333333333334 (step=0)

Figure 9: op-9

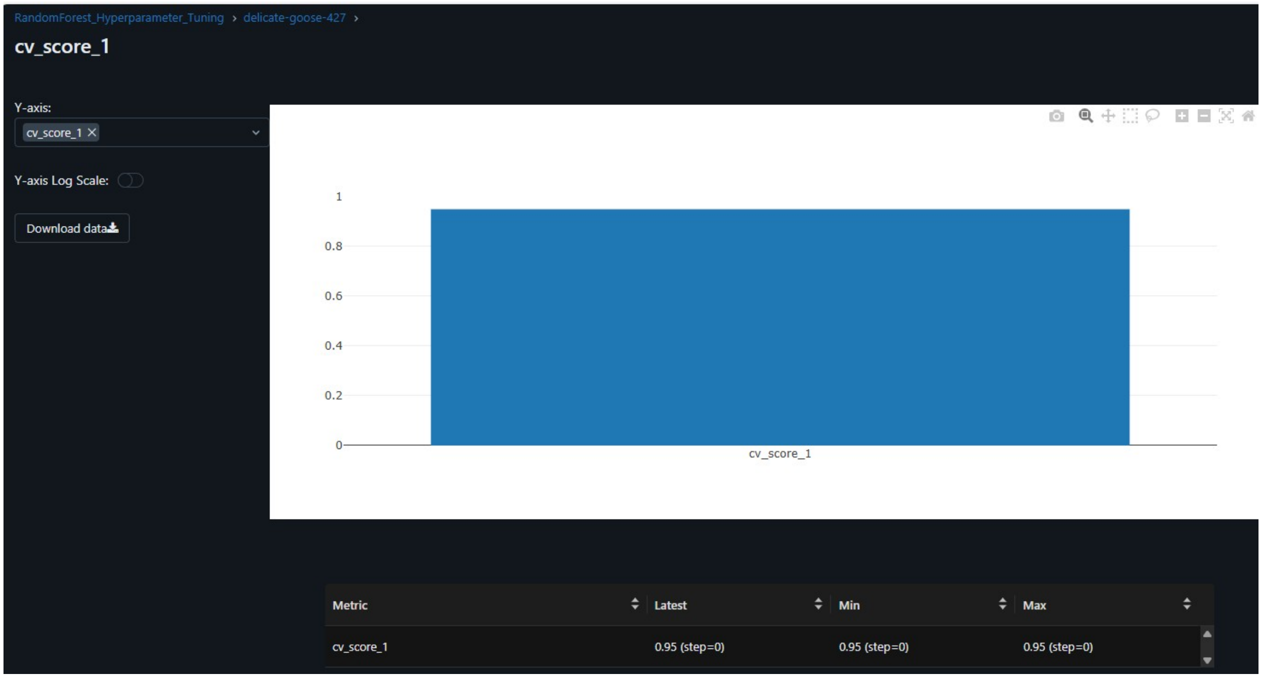
1.11 Output



	A	B	C	D	E	F
1	run_id	key	value	step	timestamp	
2	53ac6dbefb8b45c4999b05ad4407bf02	best_cv_accuracy	0.958333333	0	1.73549E+12	
3						
4						

Figure 10: op-10

1.12 Output



RandomForest_Hyperparameter_Tuning › delicate-goose-427 ›

cv_score_1

Y-axis: cv_score_1

Y-axis Log Scale: ○

Metric	Latest	Min	Max
cv_score_1	0.95 (step=0)	0.95 (step=0)	0.95 (step=0)

Figure 11: op-11

1.13 Output

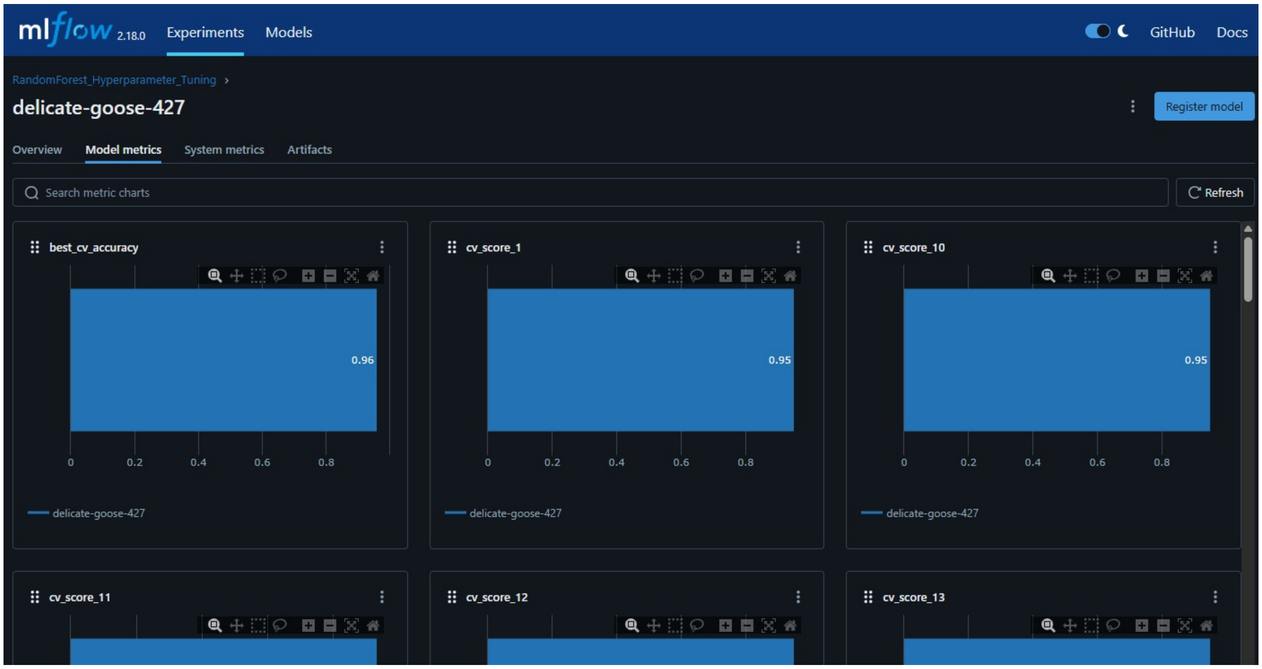
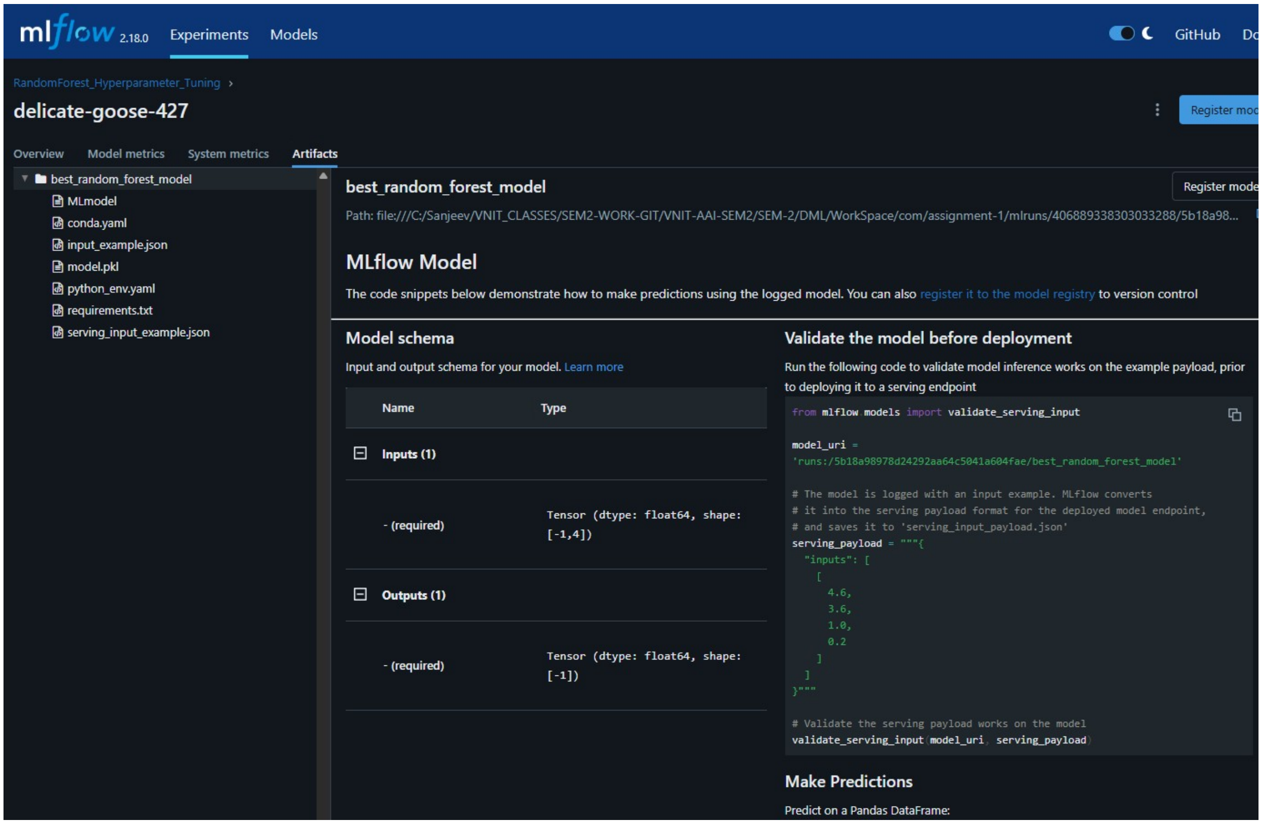


Figure 12: op-12

1.14 Output



RandomForest_Hyperparameter_Tuning ›

delicate-goose-427

- best_random_forest_model
  - MLmodel
  - conda.yaml
  - input_example.json
  - model.pkl
  - python_env.yaml
  - requirements.txt
  - serving_input_example.json

best_random_forest_model

Path: file:///C:/Sanjeev/VNIT_CLASSES/SEM2-WORK-GIT/VNIT-AAI-SEM2/SEM-2/DML/WorkSpace/com/assignment-1/mlruns/406889338303033288/5b18a98...

MLflow Model

The code snippets below demonstrate how to make predictions using the logged model. You can also register it to the model registry to version control

Model schema

Input and output schema for your model. Learn more

Name	Type
⊟ Inputs (1)	
- (required)	Tensor (dtype: float64, shape: [-1,4])
⊟ Outputs (1)	
- (required)	Tensor (dtype: float64, shape: [-1])

Validate the model before deployment

Run the following code to validate model inference works on the example payload, prior to deploying it to a serving endpoint

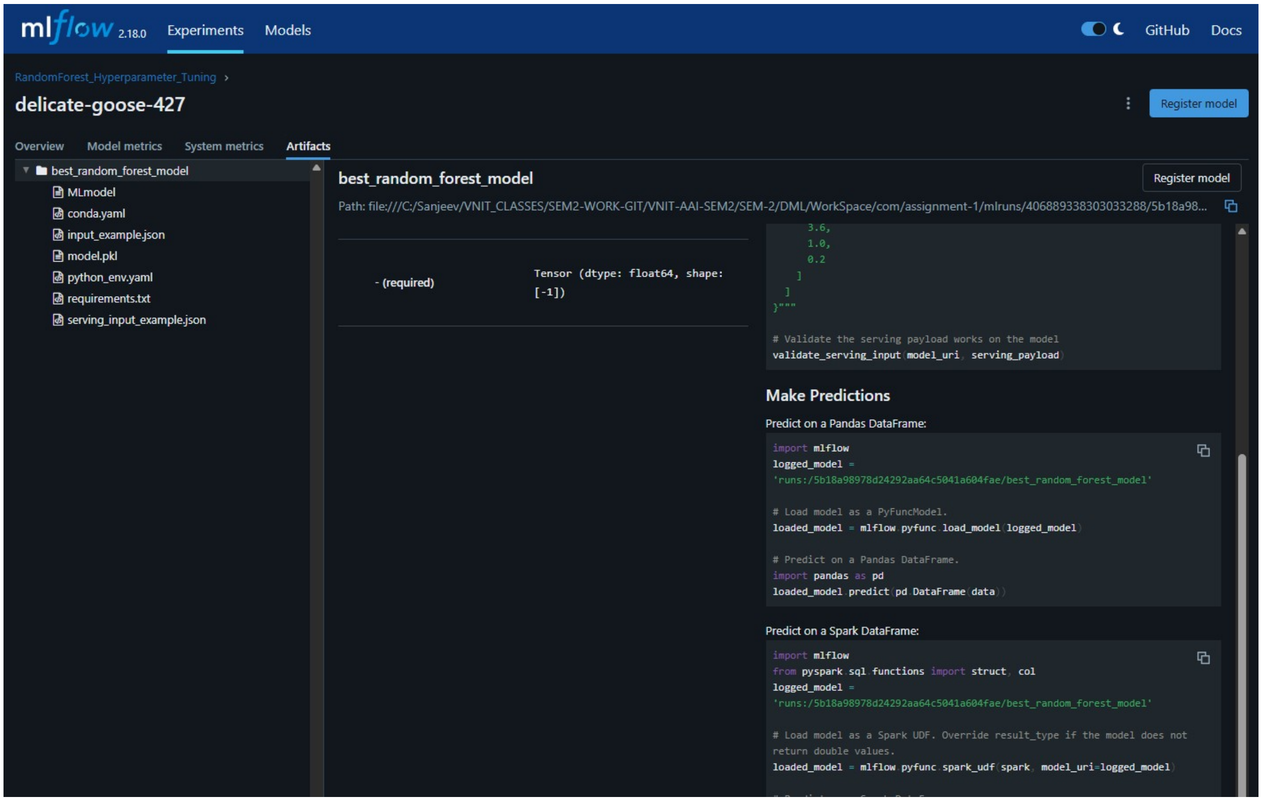
```
from mlflow.models import validate_serving_input

model_uri =
'runs:/5b18a98978d24292aa64c5041a604fae/best_random_forest_model'

# The model is logged with an input example. MLflow converts
# it into the serving payload format for the deployed model endpoint,
# and saves it to 'serving_input_payload.json'
serving_payload = """{
  "inputs": [
    [
      4.6,
      3.6,
      1.0,
      0.2
    ]
  ]
}"""

# Validate the serving payload works on the model
validate_serving_input(model_uri, serving_payload)
```

Make Predictions

Predict on a Pandas DataFrame:

Figure 13: op-13

1.15 Output



delicate-goose-427

- best_random_forest_model
  - MLmodel
  - conda.yaml
  - input_example.json
  - model.pkl
  - python_env.yaml
  - requirements.txt
  - serving_input_example.json

best_random_forest_model

Path: file:///C:/Sanjeev/VNIT_CLASSES/SEM2-WORK-GIT/VNIT-AAI-SEM2/SEM-2/DML/WorkSpace/com/assignment-1/mlruns/406889338303033288/5b18a98...

- (required)	Tensor (dtype: float64, shape: [-1])

```
      3.6,
      1.0,
      0.2
    ]
  ]
}"""

# Validate the serving payload works on the model
validate_serving_input(model_uri, serving_payload)
```

Make Predictions

Predict on a Pandas DataFrame:

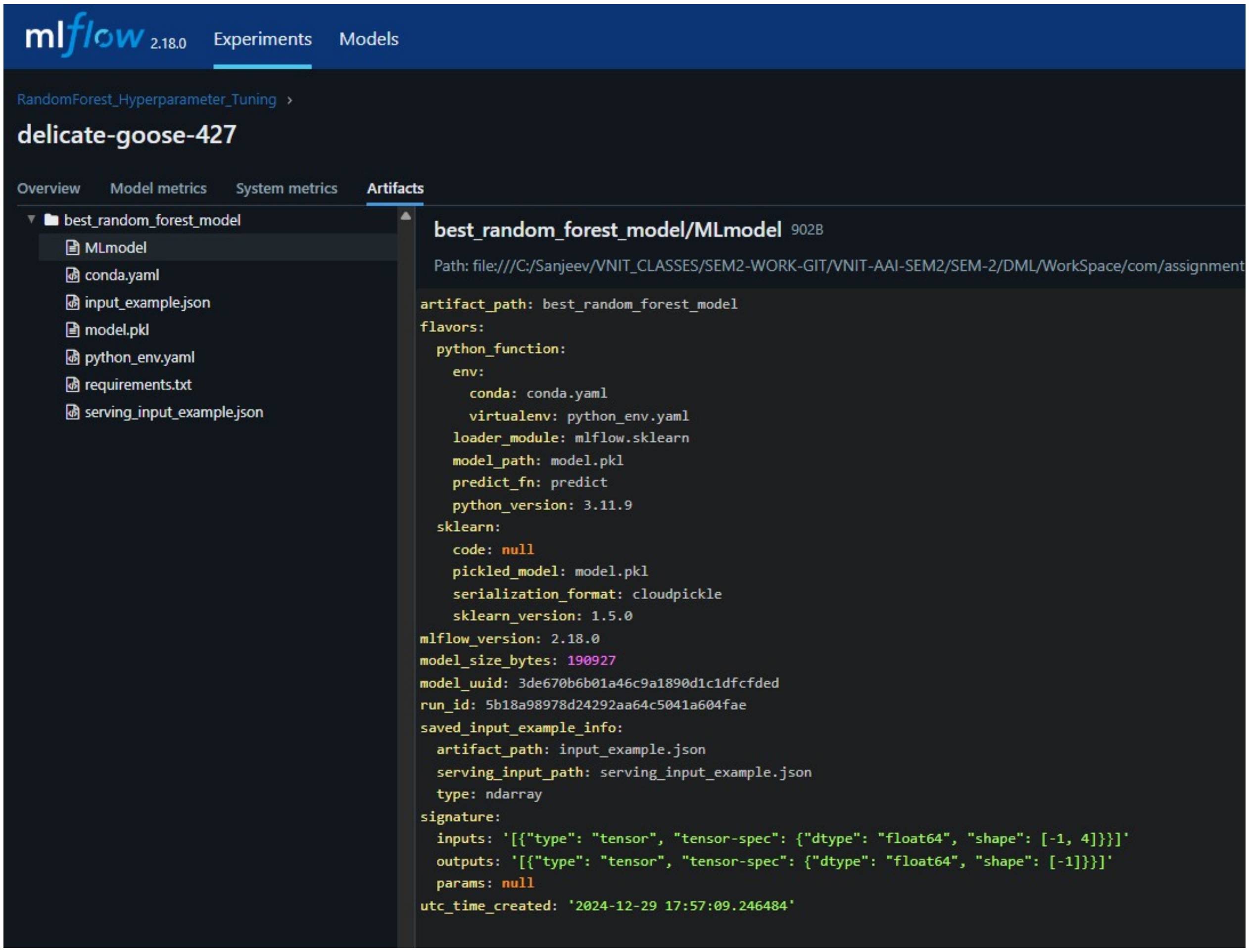
```
import mlflow
logged_model =
'runs:/5b18a98978d24292aa64c5041a604fae/best_random_forest_model'

# Load model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

# Predict on a Pandas DataFrame.
import pandas as pd
loaded_model.predict(pd.DataFrame(data))
```

Predict on a Spark DataFrame:

```
import mlflow
from pyspark.sql.functions import struct, col
logged_model =
'runs:/5b18a98978d24292aa64c5041a604fae/best_random_forest_model'

# Load model as a Spark UDF. Override result_type if the model does not
return double values.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model)
```

Figure 14: op-14

1.16 Output

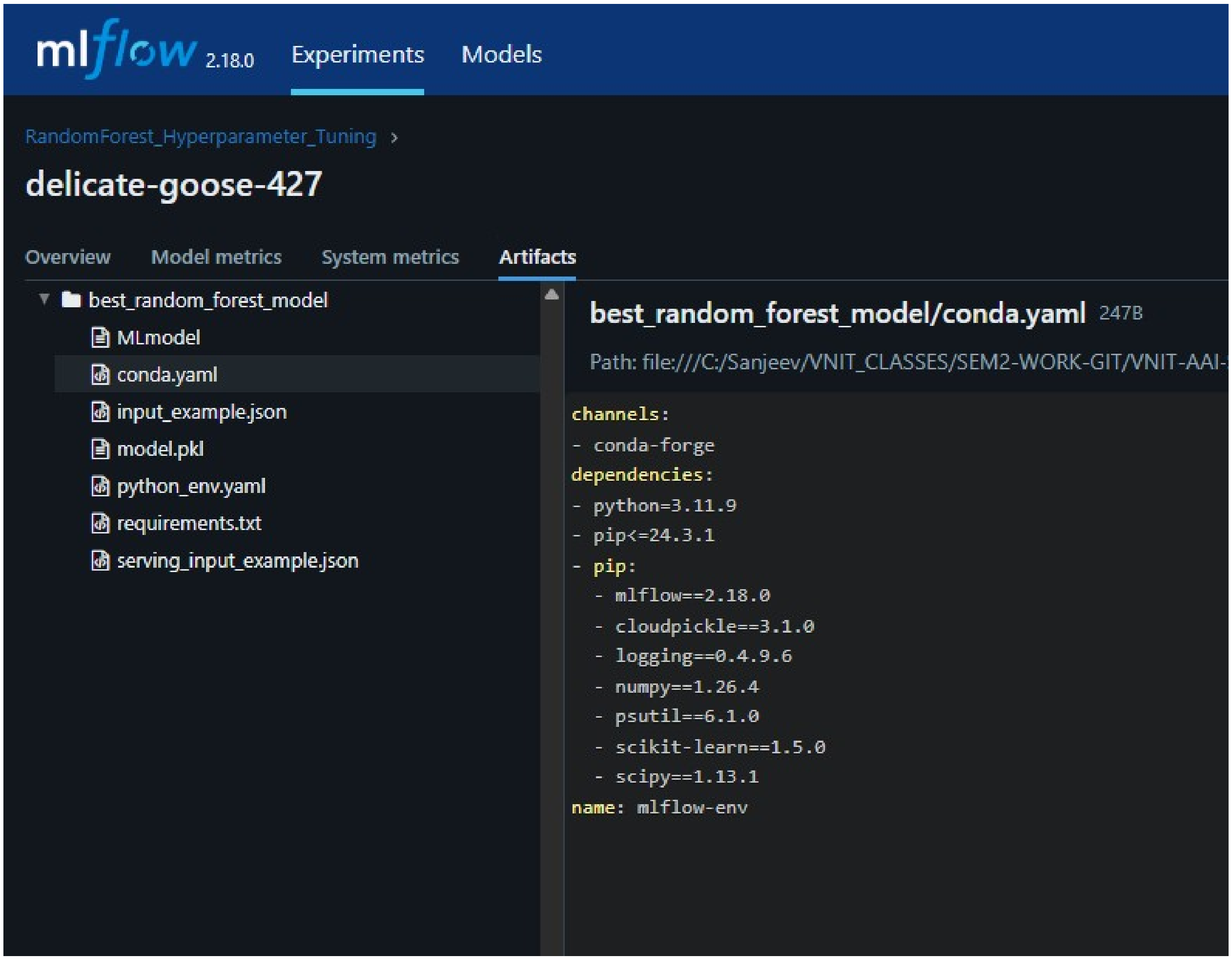


Path: file:///C:/Sanjeev/VNIT_CLASSES/SEM2-WORK-GIT/VNIT-AAI-SEM2/SEM-2/DML/WorkSpace/com/assignment

```
artifact_path: best_random_forest_model
flavors:
  python_function:
    env:
      conda: conda.yaml
      virtualenv: python_env.yaml
    loader_module: mlflow.sklearn
    model_path: model.pkl
    predict_fn: predict
    python_version: 3.11.9
  sklearn:
    code: null
    pickled_model: model.pkl
    serialization_format: cloudpickle
    sklearn_version: 1.5.0
mlflow_version: 2.18.0
model_size_bytes: 190927
model_uuid: 3de670b6b01a46c9a1890d1c1dfcfded
run_id: 5b18a98978d24292aa64c5041a604fae
saved_input_example_info:
  artifact_path: input_example.json
  serving_input_path: serving_input_example.json
  type: ndarray
signature:
  inputs: '[{"type": "tensor", "tensor-spec": {"dtype": "float64", "shape": [-1, 4]}}]'
  outputs: '[{"type": "tensor", "tensor-spec": {"dtype": "float64", "shape": [-1]}}]'
  params: null
utc_time_created: '2024-12-29 17:57:09.246484'
```

Figure 15: op-15

1.17 Output

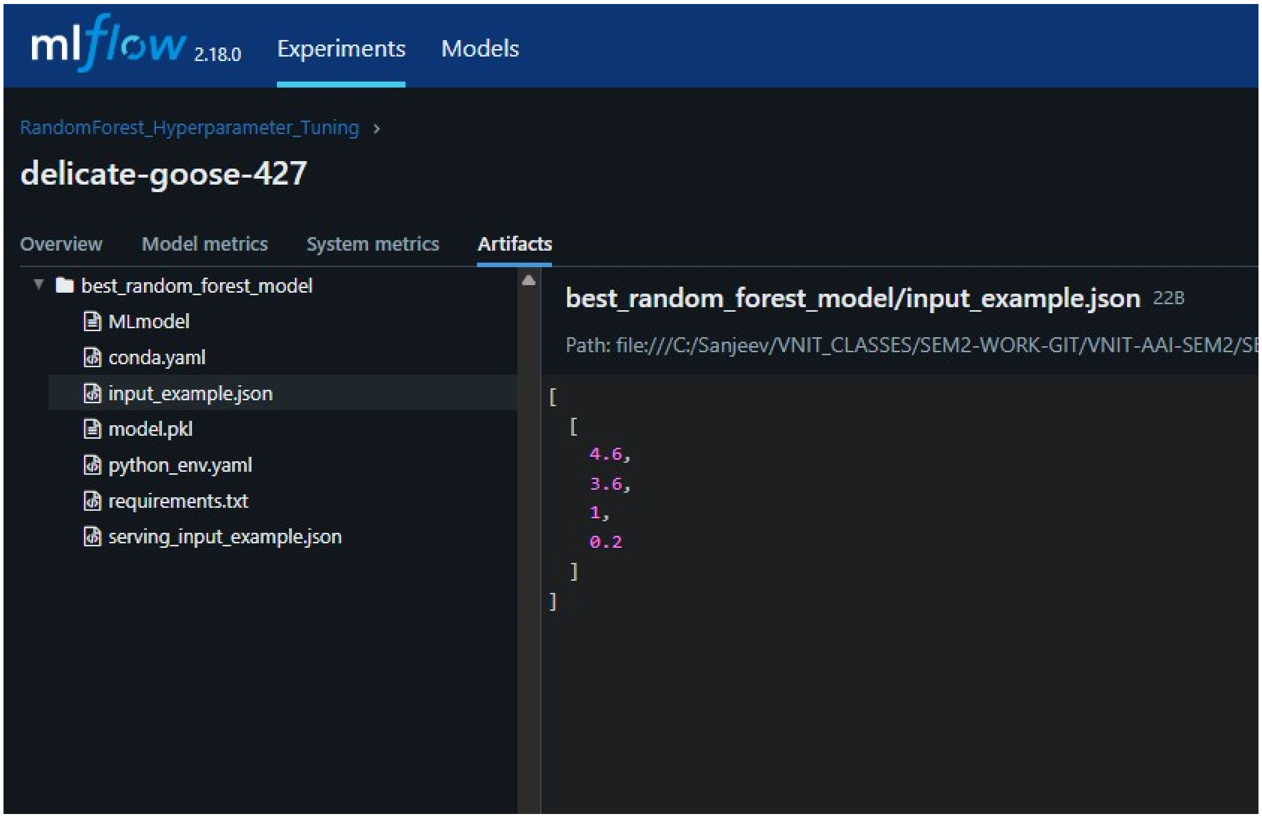


Path: file:///C:/Sanjeev/VNIT_CLASSES/SEM2-WORK-GIT/VNIT-AAI-

```
channels:
- conda-forge
dependencies:
- python=3.11.9
- pip<=24.3.1
- pip:
  - mlflow==2.18.0
  - cloudpickle==3.1.0
  - logging==0.4.9.6
  - numpy==1.26.4
  - psutil==6.1.0
  - scikit-learn==1.5.0
  - scipy==1.13.1
name: mlflow-env
```

Figure 16: op-16

1.18 Output



RandomForest_Hyperparameter_Tuning ›

delicate-goose-427

- best_random_forest_model
  - MLmodel
  - conda.yaml
  - input_example.json
  - model.pkl
  - python_env.yaml
  - requirements.txt
  - serving_input_example.json

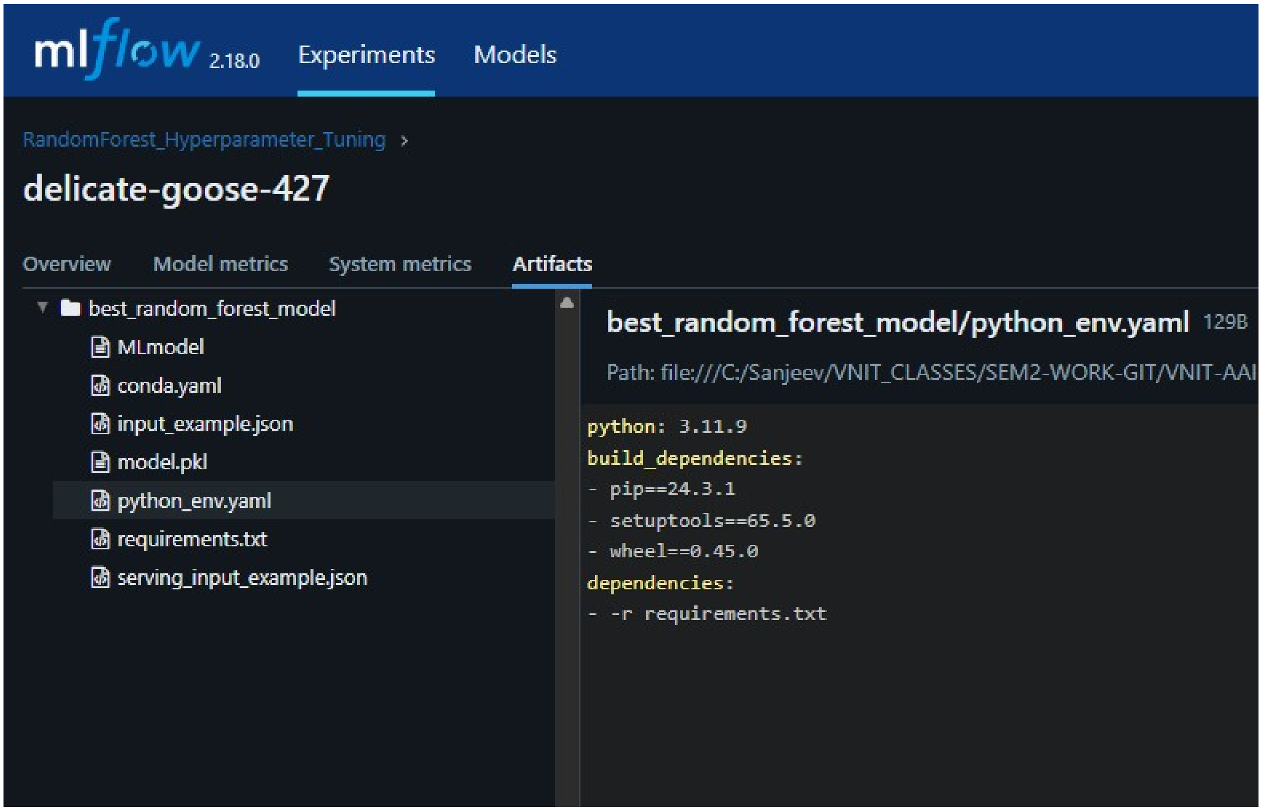
best_random_forest_model/input_example.json 22B

Path: file:///C:/Sanjeev/VNIT_CLASSES/SEM2-WORK-GIT/VNIT-AAI-SEM2/SE

```
[
  [
    4.6,
    3.6,
    1,
    0.2
  ]
]
```

Figure 17: op-17

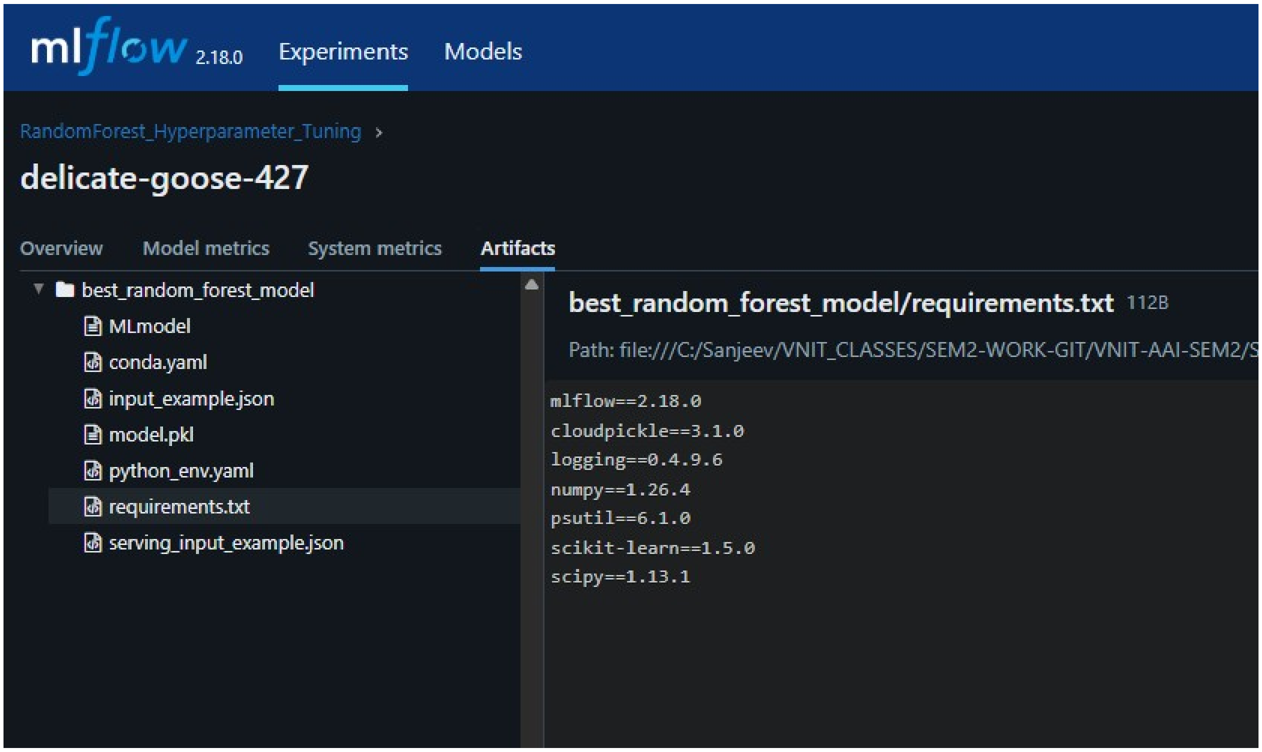
1.19 Output



```
python: 3.11.9
build_dependencies:
- pip==24.3.1
- setuptools==65.5.0
- wheel==0.45.0
dependencies:
- -r requirements.txt
```

Figure 18: op-18

1.20 Output

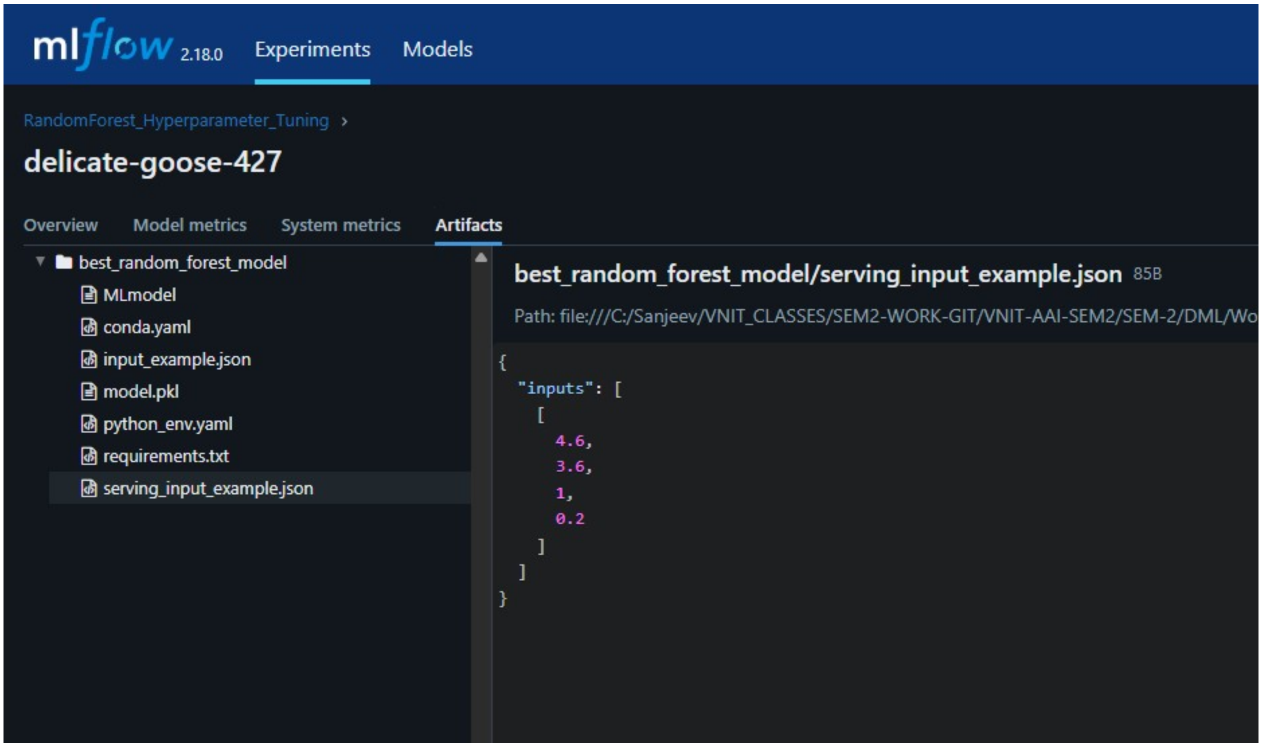


Path: file:///C:/Sanjeev/VNIT_CLASSES/SEM2-WORK-GIT/VNIT-AAI-SEM2/S

```
mlflow==2.18.0
cloudpickle==3.1.0
logging==0.4.9.6
numpy==1.26.4
psutil==6.1.0
scikit-learn==1.5.0
scipy==1.13.1
```

Figure 19: op-19

1.21 Output


mlflow 2.18.0
Experiments
Models
RandomForest_Hyperparameter_Tuning ›
delicate-goose-427
Overview
Model metrics
System metrics
Artifacts
best_random_forest_model
MLmodel
conda.yaml
input_example.json
model.pkl
python_env.yaml
requirements.txt
serving_input_example.json
best_random_forest_model/serving_input_example.json 85B
Path: file:///C:/Sanjeev/VNIT_CLASSES/SEM2-WORK-GIT/VNIT-AAI-SEM2/SEM-2/DML/Wo

```
{
  "inputs": [
    [
      4.6,
      3.6,
      1,
      0.2
    ]
  ]
}
```

Figure 20: op-20