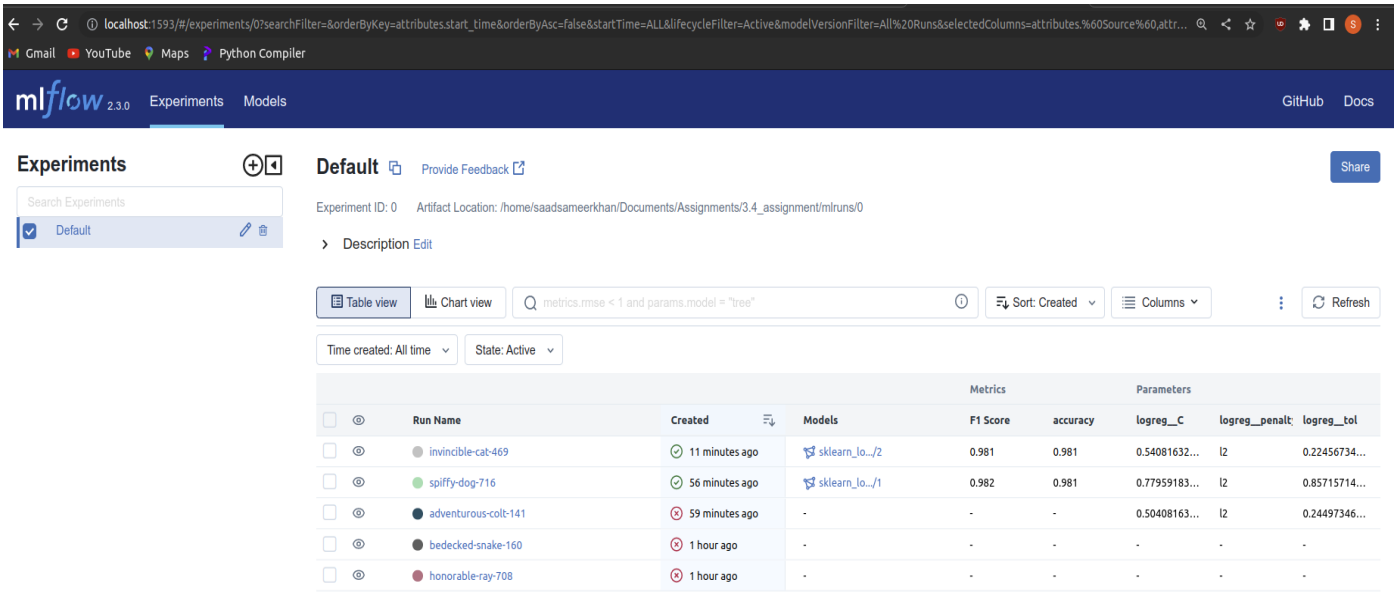
Graded Assignment 3.4(a) & 3.4(b)

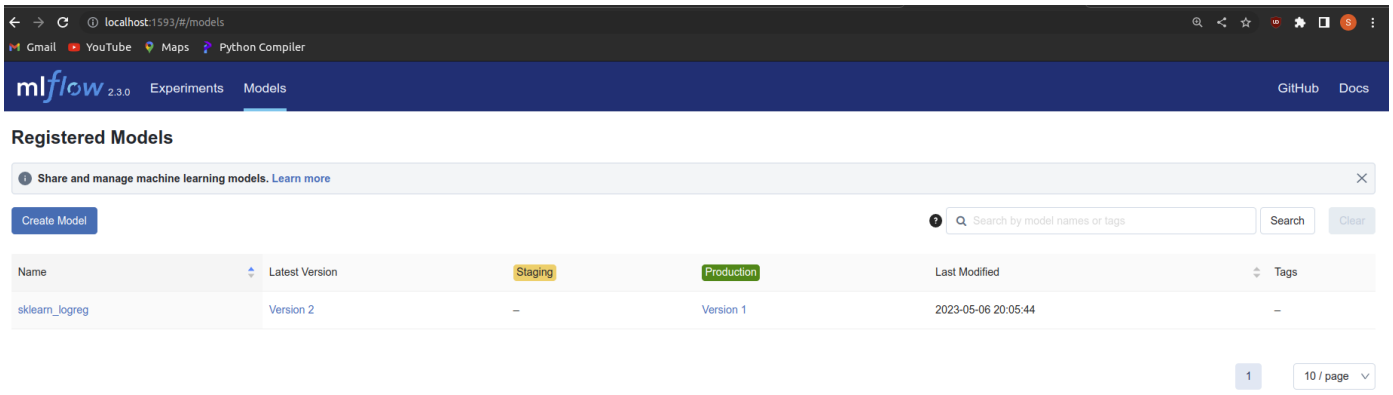
Name: Hamza Mohammad Asim

Employee#: 2303.KHI.DEG.014

Collaborated with: Saad Sameer Khan (2303.KHI.DEG.034)

Mlflow server (running at port 1593):





train.py file:

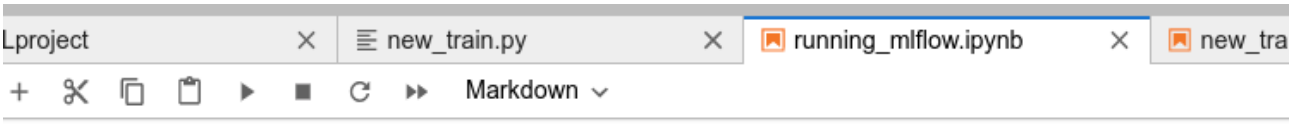
MLproject | new_train.py | running_mlflow.ipynb | new_train.ipynb

```python
import fire
import mlflow
import pandas as pd
import numpy as np
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score

#splitting our data
def split_data(x, y):
    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1522)
    y_train = y_train.values.ravel()
    return X_train, X_test, y_train, y_test


# setting a pipeline that scales and instantiates our model
def scale_pipeline(scaler_name, scaler, model_name, model):
    steps = [(scaler_name, scaler), (model_name, model)]
    pipeline_model = Pipeline(steps)
    return pipeline_model

# randomized_search looks for parameters that performs the best within a specified range
def randomized_search(pipe, x_train, y_train):
    #specifying the ranges with param_grid
    param_grid = {
              "logreg__penalty" : ["l2"],
              "logreg__tol" : np.linspace(0.0001, 1, 50),
              "logreg__C" : np.linspace(0.1, 1, 50)
             }
    randomized_search_cv = RandomizedSearchCV(pipe, param_grid, cv=10, n_iter=10)
    randomized_search_cv.fit(x_train, y_train)

    #selecting the best optimal parameters
    best_params = randomized_search_cv.best_params_

    #selecting the pipeline that performed the best with the most optimal parameters
    best_pipeline = randomized_search_cv.best_estimator_
    return best_pipeline, best_params


def track_with_mlflow(model, X_test, Y_test, mlflow, model_metadata):
```

MLproject | new_train.py | running_mlflow.ipynb | new_train.ipynb

```python
def track_with_mlflow(model, X_test, Y_test, mlflow, model_metadata):
    #logging the model parameters
    mlflow.log_params(model_metadata)

    y_pred = model.predict(X_test)

    #performing metrics
    accuracy = accuracy_score(Y_test, y_pred)
    f1 = f1_score(Y_test, y_pred, average="weighted")

    #logging the metrics
    mlflow.log_metric("accuracy", accuracy)
    mlflow.log_metric("F1 Score", f1)

    #logging our model
    mlflow.sklearn.log_model(model, "logreg", registered_model_name="sklearn_logreg")


def main(logreg_type: str, solver_name: str):
    wine = datasets.load_wine()
    X = pd.DataFrame(wine.data, columns = wine.feature_names)   # dataframe with all feature columns
    y = pd.DataFrame(wine.target, columns = ['encoded_class'])   # dataframe with target column

    #removing the last 2 columns so that we can make predictions on them later **THROUGH MLFLOW**
    X = X.iloc[:-2]
    y = y.iloc[:-2]

    X_train, X_test, y_train, y_test = split_data(X, y)

    with mlflow.start_run():
        #initializing the LogisticRegression model
        log_model = LogisticRegression(multi_class=logreg_type, solver=solver_name, random_state=1522)

        #setting up the pipeline
        pipeline = scale_pipeline("scaler", StandardScaler(), "logreg", log_model)

        #getting the best pipeline and the best parameters
        best_pipe, model_metadata = randomized_search(pipeline, X_train, y_train)

        #fitting data to the best pipeline
        best_pipe.fit(X_train, y_train)

        #tracking with mlflow
        track_with_mlflow(best_pipe, X_test, y_test, mlflow, model_metadata)
```

running_mlflow.ipynb file



[44]:
```
!python -c "import sys; print(sys.executable)"
```

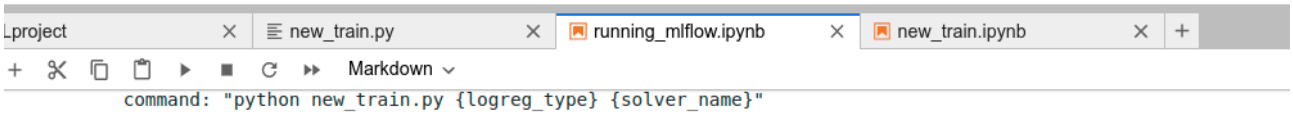
```
/home/saadsameerkhan/anaconda3/bin/python
```

Setting up mlflow server at port 1593

[45]:
```
%%bash --bg

mlflow server --host 0.0.0.0 \
    --port 1593 \
    --backend-store-uri sqlite:///mlflow.db \
    --default-artifact-root ./mlruns
```

[46]:
```
%cat MLproject
```

```
name: basic_mlflow

conda_env: conda.yaml

entry_points:
  main:
    # parameters is a key-value collection.
    parameters:
      solver_name:
        type: str
        default: "lbfgs"
      logreg_type:
        type: str
        default: "multinomial"
    command: "python new_train.py {logreg_type} {solver_name}"
```



Running MLproject file

[47]:
```
%%bash
source mlflow_env_vars.sh
mlflow run .
```

```
2023/05/06 20:05:41 INFO mlflow.utils.conda: Conda environment mlflow-dd0fbdd40ba98798131458f29496394bd1a3fb33
2023/05/06 20:05:41 INFO mlflow.projects.utils: === Created directory /tmp/tmpdnadj8kn for downloading remote
type 'path' ===
2023/05/06 20:05:41 INFO mlflow.projects.backend.local: === Running command 'source /home/saadsameerkhan/anaco
nda.sh && conda activate mlflow-dd0fbdd40ba98798131458f29496394bd1a3fb33 1>&2 && python new_train.py multinomi
d43e51bfbf4d3e87821c1c2fcbce7e' ===
/home/saadsameerkhan/anaconda3/envs/mlflow-dd0fbdd40ba98798131458f29496394bd1a3fb33/lib/python3.11/site-packag
py:33: UserWarning: Setuptools is replacing distutils.
  warnings.warn("Setuptools is replacing distutils.")
Registered model 'sklearn_logreg' already exists. Creating a new version of this model...
2023/05/06 20:05:44 INFO mlflow.tracking._model_registry.client: Waiting up to 300 seconds for model version t
e: sklearn_logreg, version 2
Created version '2' of model 'sklearn_logreg'.
2023/05/06 20:05:44 INFO mlflow.projects: === Run (ID 'a0d43e51bfbf4d3e87821c1c2fcbce7e') succeeded ===
```

Checking MLmodel file of last model run

It gives info about the model, the input output data as well as some metadata

[48]:
```
%%bash
last_model_path=$(ls -tr mlruns/0/ | tail -1)
cat mlruns/0/$last_model_path/artifacts/logreg/MLmodel
```

```
artifact_path: logreg
flavors:
  python_function:
    env:
      conda: conda.yaml
      virtualenv: python_env.yaml
    loader_module: mlflow.sklearn
    model_path: model.pkl
    predict_fn: predict
    python_version: 3.11.3
  sklearn:
    code: null
    pickled_model: model.pkl
    serialization_format: cloudpickle
    sklearn_version: 1.2.2
mlflow_version: 2.3.1
model_uuid: 5d2f9556dcea4f46aa6caf6365acbfa3
run_id: a0d43e51bfbf4d3e87821c1c2fcbce7e
utc_time_created: '2023-05-06 15:05:43.389914'
```

Serving model (at port 1594)

Serving the model that is in production

so that we can give it our data

and it give us its predictions

[49]:
```
%%bash --bg
source mlflow_env_vars.sh
mlflow --version
mlflow models serve -m models:/sklearn_logreg/Production -p 1594 --env-manager=conda
```

Selecting the last 2 rows of the dataset

These rows were unseen by the model during training

[66]:
```
from sklearn import datasets
wine = datasets.load_wine()
X = pd.DataFrame(wine.data, columns = wine.feature_names)   # dataframe with all feature columns
y = pd.DataFrame(wine.target, columns = ['encoded_class'])   # dataframe with target column
X['target'] = y
test_df = X.iloc[-2:]
test_df
```

[66]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	

Making prediction

[68]:
```
sample2 = test_df.iloc[1].tolist()
sample2 = sample2[:-1]
sample2
```

[68]:
```
[14.13, 4.1, 2.74, 24.5, 96.0, 2.05, 0.76, 0.56, 1.35, 9.2, 0.61, 1.6, 560.0]
```

Giving these 2 samples to the model

[69]:
```

13.27,4.28,2.26,20.0,120.0,1.59,0.69,0.43,1.35,10.2,0.59,1.56,835.0],[13.17,2.59,2.37,20.0,120.0,1.65,0.68,0.53,1.46,9.3,0.6,1.62,840.
ata

"{\"inputs\": $data}" -H 'Content-Type: application/json' 127.0.0.1:1594/invocations
```

```
[[13.27,4.28,2.26,20.0,120.0,1.59,0.69,0.43,1.35,10.2,0.59,1.56,835.0],[13.17,2.59,2.37,20.0,120.0,1.65,0.68,0.53,1.46,9.3,0.6,1.62,840.
0]]
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   174  100    23  100   151  19557   125k --:--:-- --:--:-- --:--:--  169k
{"predictions": [2, 2]}
```

[70]:
```
%%bash
data='[[13.27,4.28,2.26,20.0,120.0,1.59,0.69,0.43,1.35,10.2,0.59,1.56,835.0],[13.17,2.59,2.37,20.0,120.0,1.65,0.68,0.53,1.46,9.3,0.6,
echo $data

curl -d "{\"instances\": $data}" -H 'Content-Type: application/json' 127.0.0.1:1594/invocations
```

```
[[13.27,4.28,2.26,20.0,120.0,1.59,0.69,0.43,1.35,10.2,0.59,1.56,835.0],[13.17,2.59,2.37,20.0,120.0,1.65,0.68,0.53,1.46,9.3,0.6,1.62,840.
0]]
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   177  100    23  100   154   9770  65420 --:--:-- --:--:-- --:--:-- 88500
{"predictions": [2, 2]}
```

The model has made correct predictions!

MLProject File:

MLproject | new_train.py | running_mlflow.ipynb

```
name: basic_mlflow

conda_env: conda.yaml

entry_points:
  main:
    # here we specify the parameters that the our main function takes
    parameters:
      solver_name:
        type: str
        default: "lbfgs"
      logreg_type:
        type: str
        default: "multinomial"
    command: "python new_train.py {logreg_type} {solver_name}"
```

mlflow_env_vars.sh file

MLproject | mlflow_env_vars.sh | new_tra

```
#!/bin/sh

export MLFLOW_CONDA_HOME=/home/saadsameerkhan/anaconda3/
export MLFLOW_TRACKING_URI="http://0.0.0.0:1593"
export MLFLOW_AR=./mlruns
```