Unit 3.4A & 3.4B Graded Assignment

Submitted By:

1. **Ali Nasir (2303.KHI.DEG.012)**
2. **Saif ur Rehman (2303.KHI.DEG.007)**

Solution:

→ We have imported pandas and also check its version.

→ Setting up ML flow tracking system setting the ehost p[ort at 0.0.0.0and server at port 5000.
→ The the function of these codes is to they configure the server to use SQLite as the backbend store for tracking experiments and storing metadata about runs, and to store artifacts (such as models or data) in a local directory.

--backend-store-uri sqlite:///mlflow.db \
--default-artifact-root ./mlruns

MLFlow lab

```
[20]: import pandas as pd
```

```
[21]: pd.__version__
```

```
[21]: '2.0.1'
```

Setting up MLFlow tracking server

We also specify artifact root and backend store URI. This makes it possible to store models.

After running this command tracking server will be accessible at `localhost:5000`

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

mlflow server --host 0.0.0.0 \
    --port 5000 \
    --backend-store-uri sqlite:///mlflow.db \
    --default-artifact-root ./mlruns
```

→ We checking the content in Mlproject

```
[6]: %cat MLproject
```

```
name: basic_mlflow

# this file is used to configure Python package dependencies.
# it uses Anaconda, but it can be also alternatively configured to use pip.
conda_env: conda.yaml

# entry points can be ran using `mlflow run <project_name> -e <entry_point_name>
entry_points:
  download_data:
    # you can run any command using MLFlow
    command: "bash download_data.sh"
  # MLproject file has to have main entry_point. It can be toggled without using -e option.
  main:
    # parameters is a key-value collection.
    parameters:
      file_name:
        type: str
        default: "winequality_red.csv"
      max_k:
        type: int
        default: 10
    command: "python train.py {file_name} {max_k}"
```

First we need to download data. We will use weather data from previous machine learning tutorial.

→ Now we are running setting up our mlflow_env_vars environment which is setting up the conda environment and tracking uri, in download_data we are unzipping the wine quality downloaded data.

```
%%bash
source mlflow_env_vars.sh
mlflow run .  -e download_data
```

```
2023/05/05 08:37:02 INFO mlflow.utils.conda: Conda environment mlflow-08f859f1899eef32a9a81e70d4539ba67c0fa5a2 already exists.
2023/05/05 08:37:02 INFO mlflow.projects.utils: === Created directory /tmp/tmpuk52n3ms for downloading remote URIs passed to arguments of type 'path' ===
2023/05/05 08:37:02 INFO mlflow.projects.backend.local: === Running command 'source /home/ali/anaconda3/bin/../etc/profile.d/conda.sh && conda activate mlflow-08f859f1899eef32a9a81e70d4539ba67c0fa5a2 1>&2
&& bash download_data.sh' in run with ID 'eb004bc21f3841b5a410b573764eae3f' ===
Archive:  archive.zip
2023/05/05 08:37:02 INFO mlflow.projects: === Run (ID 'eb004bc21f3841b5a410b573764eae3f') succeeded ===
```

```
#!/bin/sh

export MLFLOW_CONDA_HOME=/home/ali/anaconda3/
export MLFLOW_TRACKING_URI="http://0.0.0.0:5000"
export MLFLOW_AR=./mlruns
```

```
unzip -n archive.zip
```

→ In code snipping below we have implemented train.py exracting data and passing through the supervised machine learning algorithm tracking the accuracy of the model for max_ k 10 times.

```
import fire
import mlflow
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier


def preprocess_data(df: pd.DataFrame):

    return df


# we use scikit-learn pipeline to package standarization into single object with model
def setup_decision_tree_pipeline(max_depth):
    dtc = DecisionTreeClassifier(max_depth=max_depth)
    pipe = make_pipeline(StandardScaler(), dtc)
    return pipe

def split_data(df):
    X = df.drop('quality',axis = 1)
    y = df['quality']
    X_train, X_test, Y_train, Y_test = train_test_split(X,y,test_size=0.2, random_state=42)
    return X_train, X_test, Y_train, Y_test


def track_with_mlflow(model, X_test, Y_test, mlflow, model_metadata):
    mlflow.log_params(model_metadata)
    mlflow.log_metric("accuracy", model.score(X_test, Y_test))
    mlflow.sklearn.log_model(model, "decision_tree_model", registered_model_name="sklearn_decision_tree")


def main(file_name: str, max_k: int):
    df = preprocess_data(pd.read_csv(file_name))

    X_train, X_test, Y_train, Y_test = split_data(df)
    # let's check some other k
    max_depth_list = range(1, max_k)

    for depth in max_depth_list:
        with mlflow.start_run():
            dt_pipe = setup_decision_tree_pipeline(depth)
            dt_pipe.fit(X_train, Y_train)
            model_metadata = {"max_depth": depth}
            track_with_mlflow(dt_pipe, X_test, Y_test, mlflow, model_metadata)

if __name__ == "__main__":
    fire.Fire(main)
```

→ The Ml flow model I serving at port 5010 we have loaded the data and testing our predictions

We'll load data that we can feed into prediction server.

```
# Step 1: Import Dataset
df = pd.read_csv("winequality_red.csv")
df.tail()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

```
# Step 2: Explore the dataset
print(df.shape)  # Check number of rows and columns

print(df.info())  # Display data types of each column

print(df.head())  # Show the first few rows of the dataset
# Step 3: Summary statistics
print(df.describe())
```

```
(1599, 12)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
%%bash
data='[[7.4,0.70,0.00,1.9,0.076,1.0,4.0,0.9978,3.51,0.56,9.4], [11.2,0.28,0.56,1.9,0.075,17.0,60.0,0.9980,3.16,0.58,9.8]]'
echo $data

curl -d "{\"inputs\": $data}" -H 'Content-Type: application/json' 127.0.0.1:5010/invocations
```

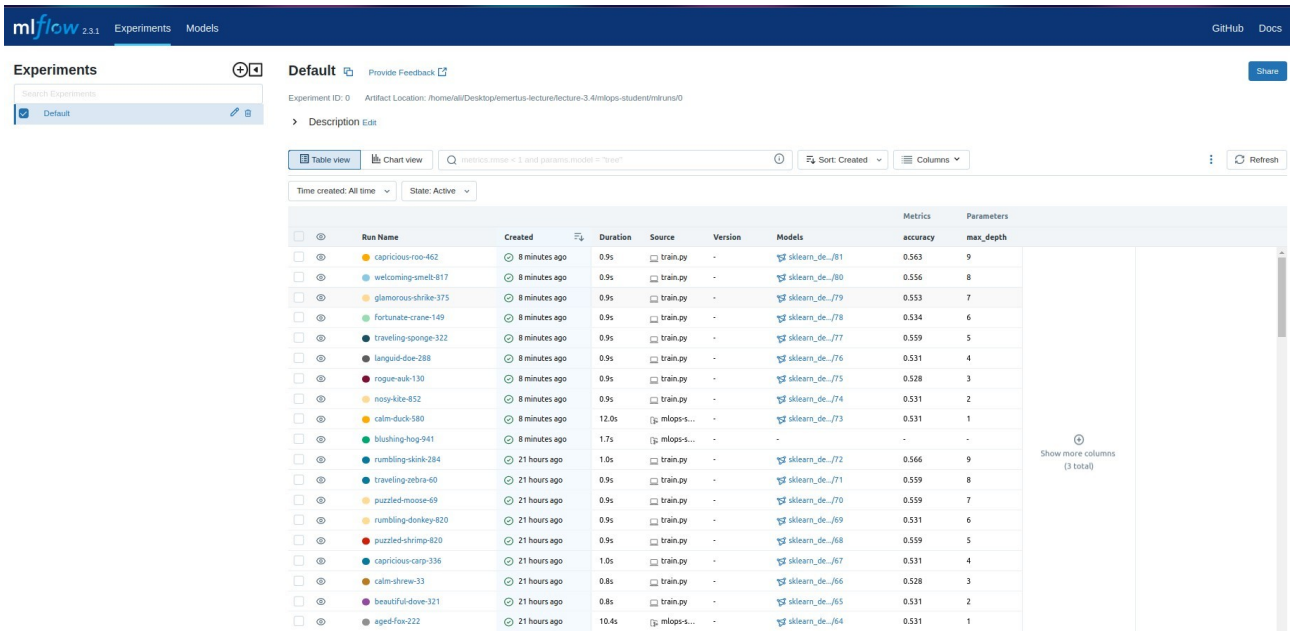
```
[[7.4,0.70,0.00,1.9,0.076,1.0,4.0,0.9978,3.51,0.56,9.4], [11.2,0.28,0.56,1.9,0.075,17.0,60.0,0.9980,3.16,0.58,9.8]]
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   150  100    23  100   127   6161  34020 --:--:-- --:--:-- --:--:-- 50000
{"predictions": [5, 5]}
```

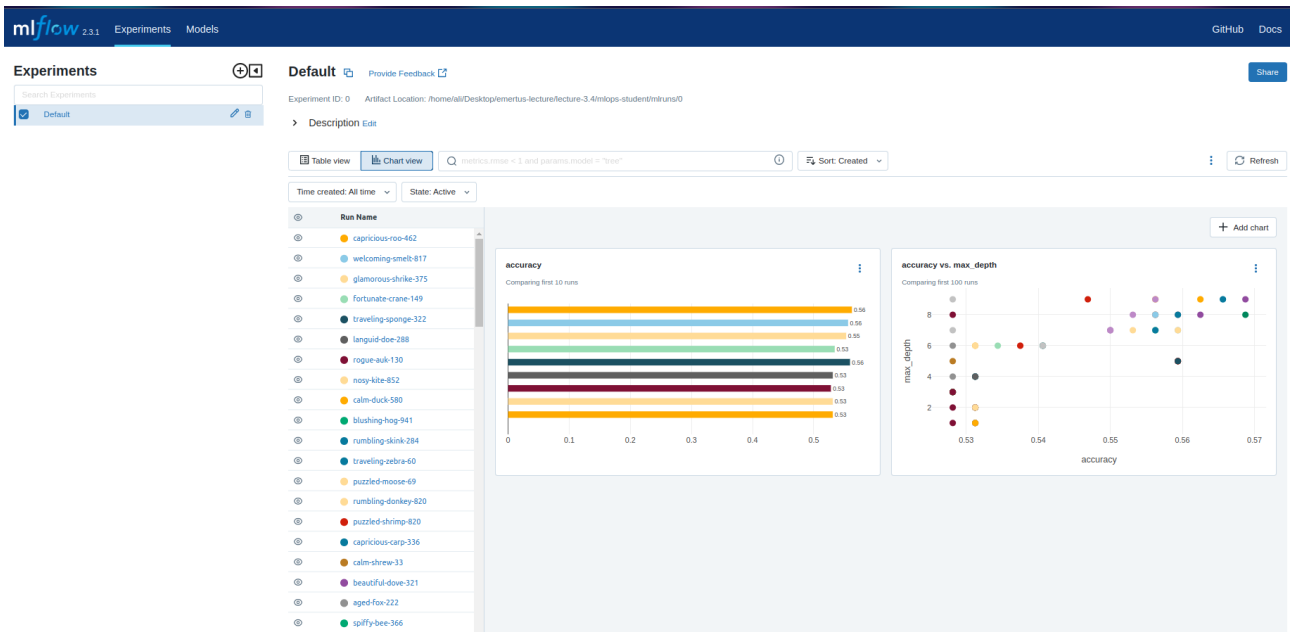
```
%%bash
data='[[6.0,0.310,0.47,3.6,0.067,18.0,42.0,0.99549,3.39,0.66,11.0], [5.9,0.645,0.12,2.0,0.075,32.0,44.0,0.99547,3.57,0.71,10.2]]'
echo $data

curl -d "{\"inputs\": $data}" -H 'Content-Type: application/json' 127.0.0.1:5010/invocations
```

```
[[6.0,0.310,0.47,3.6,0.067,18.0,42.0,0.99549,3.39,0.66,11.0], [5.9,0.645,0.12,2.0,0.075,32.0,44.0,0.99547,3.57,0.71,10.2]]
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   157  100    23  100   134  15394  89692 --:--:-- --:--:-- --:--:--  153k
{"predictions": [6, 5]}
```

→ This the the dashboard of ML flow we are getting the accuracy and the graphs for it as well.




mlflow 2.3.1
Experiments
Models
GitHub
Docs
Experiments
Search Experiments
Default
Default
Provide Feedback
Share
Experiment ID: 0
Artifact Location: /home/ali/Desktop/emertus-lecture/lecture-3.4/mlops-student/mlruns/0
Description Edit
Table view
Chart view
metrics.rmse < 1 and params.model = "tree"
Sort: Created
Refresh
Time created: All time
State: Active
Run Name
capricious-roo-462
welcoming-smelt-817
glamorous-shrike-375
fortunate-crane-149
traveling-sponge-322
languid-doe-288
rogue-auk-130
nosy-kite-852
calm-duck-580
blushing-hog-941
rumbling-skink-284
traveling-zebra-60
puzzled-moose-69
rumbling-donkey-820
puzzled-shrimp-820
capricious-carp-336
calm-shrew-33
beautiful-dove-321
aged-fox-222
spiffy-bee-366
Add chart
accuracy
Comparing first 10 runs
0.56
0.56
0.55
0.53
0.56
0.53
0.53
0.53
0.53
0
0.1
0.2
0.3
0.4
0.5
accuracy vs. max_depth
Comparing first 100 runs
max_depth
8
6
4
2
0.53
0.54
0.55
0.56
0.57
accuracy