

ML flow	ML flow tracking	MLflow projects	MLflow models	Model registry
---------	------------------	-----------------	---------------	----------------

MLflow demo

Introducti on to ML flow	
ML flow demo	
Azure ML Demo - training the model	<pre> #!/usr/bin/env python # coding: utf-8 # In[1]: #install the libraries import pandas as pd import seaborn as sns import matplotlib.pyplot as plt #Get the data from sklearn.datasets import load_breast_cancer #Load this data to cancer cancer=load_breast_cancer() #this data set is presented in dictionary form cancer.keys() #print features name cancer['feature_names'] #setup dataframe df_feat=pd.DataFrame(cancer['data'],columns=cancer['feature_names']) df_feat.info() #Print target cancer['target'] #Create dataframe for target df_target=pd.DataFrame(cancer['target'],columns=['cancer']) display(df_target) #convert the values of dataframe to array import numpy as np np.ravel(df_target) df_feat.head() #now standarizing the model from sklearn.preprocessing import StandardScaler scaler=StandardScaler() #Fit the model scaler.fit(df_feat) #Transform the dataset scaled_features=scaler.transform(df_feat) </pre>

```

df_feat_scaled=pd.DataFrame(scaled_features,columns=df_feat.columns)
df_feat_scaled.head()
#Split the data
import numpy as np
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    scaled_features, np.ravel(df_target), test_size=0.20,
    random_state=105)

from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=1) #this is k value,it mean it
will check 1 nearest data
knn.fit(X_train,y_train)
#Check the prediction
y_pred=knn.predict(X_test)

#Find matcis
def get_metrics(y_true, y_pred):
    from sklearn.metrics import
    accuracy_score,precision_score,recall_score,log_loss
    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    return {'accuracy': round(acc, 2), 'precision': round(prec, 2),
'recall': round(recall, 2)}

# In[24]:

experiment_name = "exp_1"
#Here am getting metrics in run_metrics
run_metrics = get_metrics(y_test, y_pred)
print(run_metrics)

```



The screenshot shows a Jupyter Notebook interface with the following code in the input cell:

```

In [4]: #Print target
cancer['target']

#Create dataframe for target
df_target=pd.DataFrame(cancer['target'],columns=['cancer'])
display(df_target)

```

The output of the code is a DataFrame named 'cancer' with 569 rows and 1 column. The first few rows are shown in the output:

	cancer
0	0
1	0
2	0
3	0
4	0
...	...
564	0
565	0
566	0
567	0
568	1

At the bottom of the output, it says "569 rows x 1 columns".

jupyter Untitled14 Last Checkpoint: Yesterday at 11:37 AM (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3.8

In [4]: #Print target
cancer['target']

#Create dataframe for target
df_target=pd.DataFrame(cancer['target'],columns=['cancer'])
display(df_target)

cancer	
0	0
1	0
2	0
3	0
4	0
...	...
564	0
565	0
566	0
567	0
568	1

569 rows x 1 columns

Run This
pip

pip install mlflow

pip show mlflow

In [12]: pip show mlflow

Name: mlflowNote: you may need to restart the kernel to use updated packages.

Version: 2.9.2
Summary: MLflow: A Platform for ML Development and Productionization
Home-page: <https://mlflow.org/>
Author: Databricks
Author-email:
License: Apache License 2.0
Location: C:\Users\FV565YA\AppData\Local\anaconda3\Lib\site-packages
Requires: alembic, click, cloudpickle, databricks-cli, docker, entrypoints, Flask, gitpython, importlib-metadata, J
own, matplotlib, numpy, packaging, pandas, protobuf, pyarrow, pytz, pyyaml, querystring-parser, requests, scikit-
sqlalchemy, sqlparse, waitress
Required-by:

In [13]: import mlflow

#Set the experiment
mlflow.set_experiment(experiment_name)

#Start the mlflow
with mlflow.start_run():
 #Check if metrics present in this run_metrics (if yes log that
 for metric in run_metrics:
 mlflow.log_metric(metric, run_metrics[metric])

#send the model also

Anaconda Prompt - mlflow ui

```
(base) C:\Users\FV565YA>mlflow ui  
INFO:waitress:Serving on http://127.0.0.1:5000
```

```
import mlflow  
#Set the experiment  
mlflow.set_experiment(experiment_name)  
#Start the mlflow  
with mlflow.start_run():  
    #Check if metrics present in this run_metrics if yes log that  
    for metric in run_metrics:  
        mlflow.log_metric(metric, run_metrics[metric])  
  
    #send the model also  
    mlflow.sklearn.log_model(knn, "model")  
  
print('It is logged to Experiment - %s' %( experiment_name))
```

own, matplotlib, numpy, packaging, pandas, protoout, pyarrow, sqlalchemy, sqlparse, waitress
Required-by:

```
In [12]: import mlflow

#Set the experiment
mlflow.set_experiment(experiment_name)

#Start the mlflow
with mlflow.start_run():
    #Check if metrics present in this run_metrics if yes log
    for metric in run_metrics:
        mlflow.log_metric(metric, run_metrics[metric])

    #send the model also
    mlflow.sklearn.log_model(knn, "model")

print('It is logged to Experiment - %s' %( experiment_name))

It is logged to Experiment - exp_1
```

The screenshot displays the MLflow web interface. At the top, there's a navigation bar with 'mlflow 3.0.2', 'Experiments', and 'Models'. The 'Experiments' section is active, showing a list of experiments on the left. 'exp_1' is selected. The main area shows details for 'exp_1', including its ID (534482564028160034) and artifact location. Below this, there's a search bar and filters for 'State: Active' and 'Sort: Created'. A table lists runs with columns for Run Name, Created, and Deleted. The runs listed are 'furry-gndi-996', 'loki-pug-367', 'clean-bear-381', and 'furry-lab-367'. Below the table, the 'Artifacts' section is expanded, showing a list of artifacts: 'MLmodel', 'conda.yaml', 'model.pkl', 'python_env.yaml', and 'requirements.txt'. The 'MLflow Model' section provides code snippets for making predictions using the logged model, both for Spark DataFrames and Pandas DataFrames.

Experiments

Search Experiments

Default

exp_1

exp_1 Provide Feedback

Experiment ID: 534482564028160034 Artifact Location: file:///C:/Users/FV562WA/mruns/534482564028160034

Description Edit

metrics: none = 1 and parent: model = "none"

Time created: -

State: Active Sort: Created Columns

Table Chart Evaluation Experimental

Run Name	Created	Deleted
furry-gndi-996	1 minute ago	-
loki-pug-367	10 minutes ago	-
clean-bear-381	11 minutes ago	-
furry-lab-367	6 hours ago	-

Artifacts

model

Full Path: file:///C:/Users/FV562WA/mruns/534482564028160034/3944aeb96466433e8ef4d8d993ede98d3/artifacts/

MLflow Model

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

Model schema

Input and output schema for your model. Learn more

Name	Type
------	------

No schema. See MLflow docs for how to include input and output schema with your model.

Make Predictions

Predict on a Spark DataFrame:

```
import mlflow
from pyspark.sql.functions import struct, col
logged_model = 'runs:/534482564028160034/3944aeb96466433e8ef4d8d993ede98d3/model'

# Load model as a Spark UDF. Override result_type if the model
# returns double values.
loaded_model = mlflow.pyfunc.spark_udf(spark, model_uri=logged_model,
                                       result_type='double')

# Predict on a Spark DataFrame.
df_withColumn("predictions", loaded_model(struct(col, df)))

# Predict on a Pandas DataFrame.
```

	<pre> import mlflow logged_model = 'runs:/420e1098ea034b2fb182467ea5dd7a93/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(X_test)) # In[19]: </pre>
Introducti on to Azure ML	
Azure ML Demo- Deploying the model It has 3 Parts	<<Classification Model Deployment using Azure ML designer.docx>>