DATADUDE[SAAD KHAN]

MLOps

$Step-1$: **Install required packages**

here our required package is mlflow

```
!pip install mlflow
```

Show hidden output

```
!mlflow
```

```
Usage: mlflow [OPTIONS] COMMAND [ARGS]...

Options:
  --version  Show the version and exit.
  --help     Show this message and exit.

Commands:
  artifacts    Upload, list, and download artifacts from an MLflow...
  db           Commands for managing an MLflow tracking database.
  deployments  Deploy MLflow models to custom targets.
  doctor       Prints out useful information for debugging issues with MLflow.
  experiments  Manage experiments.
  gc           Permanently delete runs in the `deleted` lifecycle stage.
  models       Deploy MLflow models locally.
  recipes      Run MLflow Recipes and inspect recipe results.
  run          Run an MLflow project from the given URI.
  runs         Manage runs.
  sagemaker    Serve models on SageMaker.
  server       Run the MLflow tracking server.
```

```
!mlflow --version
```

```
mlflow, version 2.13.1
```

$Step-2$: **Import required packages**

- Pandas : to read and analysis of the data
- Numpy: For the numerical analysis/ some mathamatical formulae
- Sklearn: sikit-learn for ML development
- mlflow

```
import numpy as np
import pandas as pd
from sklearn.linear_model import ElasticNet  # ML model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
import mlflow
import mlflow.sklearn
```

$Step - 3$: **Read the data**

```
# Here we are taking wine quality dataset
# It is a regression data set
# This data set is available in kaggle as wela as in UCI website
data=pd.read_csv('/content/winequality_red.csv')
data.head()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	q
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	

```
#data.dropna
```

```
data.columns
```

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol', 'quality'],
      dtype='object')
```

```
data.shape
# we have 12 columns
# In that 11 columns are input columns ==================== > X
# 1 column: Quality is output column( Target features) ===== > y
# The total number of observations are 1599
```

```
(1599, 12)
```

```
data.info()
```

```
<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
```

```
data.isnull().sum()
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

$Step - 4$:

Develop ML model using MLflow

- we need to divide data into two parts train_data and test_data
- both train_data and test_data includes input columns(X) and output columns(y)
- Next we divide train_data to X_train and y_train
- Next we divide test_data to X_test and y_test
- Model will be developed on train data
- Model predictions happen on X_test data that is called y_predictions
- Finally we compare y_test with y_predictions

```
# Start our mlflow here
# Mlflow will create some default experiments, our models will deploy in that experiment

mlflow.set_experiment("/mlflow/naresh_it")
# all the logs
# all the deployments
# will save in this folder  "mlflow/naresh_it"
# artifacts
# content/mlruns/254997122522303723
```

```
2024/05/31 06:32:00 INFO mlflow.tracking.fluent: Experiment with name '/mlflow/naresh_it' does not exis
<Experiment: artifact_location='file:///content/mlruns/946479812595149752',
creation_time=1717137120631, experiment_id='946479812595149752', last_update_time=1717137120631,
lifecycle_stage='active', name='/mlflow/naresh_it', tags={}>
```

```
print(mlflow.set_experiment("/mlflow/saadkhan").experiment_id)
print(mlflow.set_experiment("/mlflow/saadkhan").name)
print(mlflow.set_experiment("/mlflow/saadkhan").lifecycle_stage)
print(mlflow.set_experiment("/mlflow/saadkhan").artifact_location)

# yaml files configuration files
```

```
2024/05/31 06:33:30 INFO mlflow.tracking.fluent: Experiment with name '/mlflow/saadkhan' does not exist
168934479190679116
/mlflow/saadkhan
```

```
        active
        file:///content/mlruns/168934479190679116
```

```
train,test=train_test_split(data,test_size=0.3,random_state=1234)

# test size=0.3 means train data is 70% and test data =30%
# random state gives random observations

train_x=train.drop(['quality'],axis=1)  # we are droping qulaity column
train_y= train[['quality']]
test_x=test.drop(['quality'],axis=1)
test_y=test[['quality']]
```

```
train.shape,test.shape
```

```
((1119, 12), (480, 12))
```

```
train_x.shape,test_x.shape
```

```
((1119, 11), (480, 11))
```

```
train_y.shape,test_y.shape
```

```
((1119, 1), (480, 1))
```

```
from mlflow.entities.model_registry import registered_model
#from atexit import register
def train_model(alpha,l1_ratio):
    #==============Develop train test data=====================================
    train,test=train_test_split(data,
                                test_size=0.3,
                                random_state=1234)
    train_x=train.drop(['quality'],axis=1)  # we are droping qulaity column
    train_y= train[['quality']]
    test_x=test.drop(['quality'],axis=1)
    test_y=test[['quality']]

    #====== Now initiate MLflow run===================================
    with mlflow.start_run(experiment_id=168934479190679116,
                          run_name='regression',
                          description='Performing regression model'):

        #========== Model building=================================
        lr=ElasticNet(alpha=alpha,l1_ratio=l1_ratio)
        lr.fit(train_x,train_y)

        #========== Model prediction=================================
        predicted_data= lr.predict(test_x)

        # ========= Model evaluation=================================
        rmse=np.sqrt(mean_squared_error(test_y,predicted_data))
        mae=mean_absolute_error(test_y,predicted_data)
        r2=r2_score(test_y,predicted_data)

        print("rmse:",rmse)
        print("mae:",mae)
        print("r2_score:",r2)

        #=============Log the metrics, parameters,and model=======================
        mlflow.log_param("alpha",alpha)
        mlflow.log_param("l1_ratio",l1_ratio)

        mlflow.log_metric("RMSE",rmse)
        mlflow.log_metric("MAE",mae)
        mlflow.log_metric("R2",r2)

        mlflow.sklearn.log_model(lr,"model",registered_model_name="ElasticNet")
```

```
train_model(0.3,0.4)

# In the name of Elastic model
# we ran two times:  a new version
```

```
rmse: 0.6699096451486062
mae: 0.5221158983216444
r2_score: 0.21304605273127153
/usr/local/lib/python3.10/dist-packages/_distutils_hack/__init__.py:33: UserWarning: Setuptools is repl
  warnings.warn("Setuptools is replacing distutils.")
Successfully registered model 'ElasticNet'.
Created version '1' of model 'ElasticNet'.
```

$Step-5$: **Connect with MLflow UI**

```
# Now we need to see all artifacts in mlflow UI
# For this will use  ngrok

!pip install pyngrok
```

```
Collecting pyngrok
  Downloading pyngrok-7.1.6-py3-none-any.whl (22 kB)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.10/dist-packages (from pyngrok) (6
Installing collected packages: pyngrok
Successfully installed pyngrok-7.1.6
```

```
from pyngrok import ngrok
ngrok_tunnel= ngrok.connect(addr='5000',proto="http")
print("Tracking uri:",ngrok_tunnel.public_url)
```

```
Tracking uri: https://6347-104-198-179-157.ngrok-free.app
```

```
!mlflow ui
# trigger this first
# then open url
```

```
[2024-05-31 07:02:52 +0000] [8875] [INFO] Starting gunicorn 22.0.0
[2024-05-31 07:02:52 +0000] [8875] [INFO] Listening at: http://127.0.0.1:5000 (8875)
[2024-05-31 07:02:52 +0000] [8875] [INFO] Using worker: sync
[2024-05-31 07:02:52 +0000] [8876] [INFO] Booting worker with pid: 8876
[2024-05-31 07:02:52 +0000] [8877] [INFO] Booting worker with pid: 8877
[2024-05-31 07:02:52 +0000] [8878] [INFO] Booting worker with pid: 8878
[2024-05-31 07:02:52 +0000] [8879] [INFO] Booting worker with pid: 8879
```

```
# 2hBhb2tVKvPLkF24ACLI7ULAnkT_77nT4dmo24Hu28Tpf4U3h
```

Your Authtoken

This is your personal Authtoken. Use this to authenticate the ngrok agent that you downloaded.

2KVWElPAg1tzrBVnxmFFYa

Copy

```
from pyngrok import ngrok
ngrok.kill()
```

```
from pyngrok import ngrok
ngrok.kill()  # It will kill the already established tunnels

auth_code='2hBhb2tVKvPLkF24ACLI7ULAnkT_77nT4dmo24Hu28Tpf4U3h'
ngrok.set_auth_token(auth_code)
```

```
ngrok_tunnel= ngrok.connect(addr='5000',proto="http")
print("Tracking uri:",ngrok_tunnel.public_url)
```

```
Tracking uri: https://c984-104-198-179-157.ngrok-free.app
```

```
import mlflow
logged_model = 'runs:/874b3cc09f94469ca678ca2b2e1a9cce/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(test_x))
```

```
array([5.50691077, 5.44702698, 5.76445222, 5.76641743, 5.90243809,
       5.5036374 , 5.75608784, 5.98582184, 5.57053167, 6.26511971,
       5.22755313, 5.3581384 , 5.81797447, 5.55800702, 5.62260861,
       5.36544904, 5.15382329, 5.42999068, 5.85164569, 5.13261257,
       5.03540007, 5.72708555, 5.35291168, 5.64415582, 5.7580662 ,
       5.85814378, 5.42198793, 5.28166568, 5.71111299, 5.58721121,
       5.25290353, 5.92462832, 5.21442612, 5.67868597, 5.4801912 ,
       5.39120765, 6.04083875, 5.51315581, 5.50042925, 5.89968015,
       5.25243326, 5.79612922, 5.70267694, 5.45013977, 5.51339529,
       5.3077769 , 4.97863579, 5.83555803, 5.74596448, 5.45110355,
       5.25886646, 5.61039293, 6.02969701, 5.79597746, 5.67296378,
       5.5123182 , 5.93171896, 5.42669798, 5.94282161, 5.44801926,
       5.71459517, 5.1808671 , 5.28114072, 5.5429531 , 5.26551196,
       5.88447329, 5.55443771, 5.97316062, 5.35080663, 5.84528777,
       5.89718998, 5.81779455, 5.37387673, 6.01998578, 5.28166568,
       5.51403228, 6.18341298, 5.81794479, 5.10793968, 5.34207852,
       5.59026392, 5.39471391, 5.79926458, 5.8076285 , 5.49263202,
       5.50055218, 5.82206368, 5.3803558 , 5.81862131, 5.79942125,
       6.05472666, 5.37269552, 5.72174227, 5.25222964, 5.75247885,
       5.91022743, 5.49189252, 5.94950145, 5.98857939, 6.18341298,
       5.53708144, 5.636761  , 5.01851843, 5.29515728, 5.48753784,
       5.25268925, 5.78367181, 5.58044844, 5.66096618, 5.61312218,
       5.80907603, 5.77078119, 6.01264085, 5.63920128, 5.63059886,
       5.47949903, 5.64201431, 5.84371394, 6.15414105, 5.21608741,
       5.79580917, 5.76300936, 5.50938079, 6.10075123, 5.86594003,
       5.1428468 , 5.51716487, 6.07635402, 5.87624244, 5.80113831,
       5.16930615, 5.08011379, 5.8916408 , 5.4112736 , 5.45862412,
       6.08061452, 5.59631898, 5.55997807, 5.38701611, 5.43958577,
       5.64974278, 5.56378488, 5.41116483, 5.23976572, 5.52808489,
       5.57650796, 5.52824455, 5.93242719, 6.05354735, 5.47920024,
       5.29330666, 5.68703838, 6.02564868, 6.12644877, 5.78018938,
       5.71631876, 5.7039909 , 5.76976621, 6.12717709, 5.75268326,
```