**Custom model tracking with MLflow and remote mySQL docker database**

**Setting up MySQL database:**

**run docker desktop and open wsl linux on windows**

**powershell > wsl**

* docker pull mysql/mysql-server:latest
* docker run --name=mysql1 -p 3306:3306 -p 33060:33060 -d mysql/mysql-server:latest
* We map the ports 3306 and 33060 to the docker container so that we can access the database outside of the docker container.
* **docker ps**

, let’s configure the password for the root user. To do that we will need the automatically generated password for root.

> docker logs mysql1 2>&1 | grep GENERATED

**>** docker exec -it mysql1 mysql -u root -p

Mysql will ask for the password, you must enter the password generated by mysql which we saw earlier.

After this, we can change the password for the root user. Replace the string **‘password’** with the actual password that you want to set. To make the access available for the root user from outside the container, we will update the host value from ‘localhost’ to ‘%’.

> alter user 'root'@'localhost' identified by 'password';

>update mysql.user set host = ‘%’ where user=’root’;

>SELECT host, user FROM mysql.user;

*>* create database mlflow;

add new user account

>CREATE USER ‘senol’@'localhost' IDENTIFIED BY 'password';

> GRANT ALL PRIVILEGES ON \*.\* TO ‘senol’@'localhost'

-> WITH GRANT OPTION;

> CREATE USER 'senol'@'%' IDENTIFIED BY 'password';

> GRANT ALL PRIVILEGES ON \*.\* TO 'senol'@'%'

-> WITH GRANT OPTION;

*>quit;*

**Now open conda shell on windows and run mlflow ui with parameters below:**

* **mlflow ui --backend-store-uri "mysql://senol:password@localhost:3306/mlflow"**

**then run below code in pycharm and refresh mlflow ui from web browser to see the new runs.**

from sys import version\_info  
import cloudpickle  
import pandas as pd  
  
import mlflow.pyfunc  
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer  
  
#import os  
#print("ENV:",os.getenv('MLFLOW\_TRACKING\_URI'))  
  
#  
# Good and readable paper from the authors of this package  
# http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf  
#  
#remote\_server\_uri = "http://localhost:5000" # set to your server URI. Local web server. Artifacts are in local mlruns.  
#mlflow.set\_tracking\_uri(remote\_server\_uri)  
  
user = 'senol'  
password = 'password'  
hostname = 'localhost'  
port = 3306  
database = 'mlflow'  
uri = f'mysql://{user}:{password}@{hostname}:{port}/{database}'  
print(uri)  
mlflow.set\_tracking\_uri(uri)  
  
mlflow.set\_experiment("/SERVE\_CUSTOM\_NODEL\_vader")  
  
INPUT\_TEXTS = [{'text': "This is a bad movie. You don't want to see it! :-)"},  
 {'text': "Ricky Gervais is smart, witty, and creative!!!!!! :D"},  
 {'text': "LOL, this guy fell off a chair while sleeping and snoring in a meeting"},  
 {'text': "Men shoots himself while trying to steal a dog, OMG"},  
 {'text': "Yay!! Another good phone interview. I nailed it!!"},  
 {'text': "This is INSANE! I can't believe it. How could you do such a horrible thing?"}]  
  
PYTHON\_VERSION = "{major}.{minor}.{micro}".format(major=version\_info.major,  
 minor=version\_info.minor,  
 micro=version\_info.micro)  
def score\_model(loaded\_model):  
 # Use inference to predict output from the customized PyFunc model  
 for i, text in enumerate(INPUT\_TEXTS):  
 text = INPUT\_TEXTS[i]['text']  
 m\_input = pd.DataFrame([text])  
 scores = loaded\_model.predict(m\_input)  
 print(f"<{text}> -- {str(scores[0])}")  
  
# Define a class and extend from PythonModel  
class SocialMediaAnalyserModel(mlflow.pyfunc.PythonModel):  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 # embed your vader model instance  
 self.\_analyser = SentimentIntensityAnalyzer()  
  
 # preprocess the input with prediction from the vader sentiment model  
 def \_score(self, txt):  
 prediction\_scores = self.\_analyser.polarity\_scores(txt)  
 return prediction\_scores  
  
 def predict(self, context, model\_input):  
  
 # Apply the preprocess function from the vader model to score  
 model\_output = model\_input.apply(lambda col: self.\_score(col))  
 return model\_output  
  
  
model\_path = "vader\_model"  
reg\_model\_name = "NewPyFuncVaderSentimentAnalysis"  
vader\_model = SocialMediaAnalyserModel()  
  
  
  
# Save the conda environment for this model.  
conda\_env = {  
 'channels': ['defaults', 'conda-forge'],  
 'dependencies': [  
 'python={}'.format(PYTHON\_VERSION),  
 'pip'],  
 'pip': [  
 'mlflow',  
 'cloudpickle=={}'.format(cloudpickle.\_\_version\_\_),  
 'vaderSentiment==3.3.2'  
 ],  
 'name': 'mlflow-env'  
}  
  
# Save the model  
with mlflow.start\_run(run\_name="Vader\_Sentiment\_Analysis") as run:  
 model\_path = f"{model\_path}-{run.info.run\_uuid}"  
 mlflow.log\_param("algorithm", "VADER")  
 mlflow.log\_param("total\_sentiments", len(INPUT\_TEXTS))  
 #mlflow.pyfunc.save\_model(path=model\_path, python\_model=vader\_model, conda\_env=None)  
  
 # Use the saved model path to log and register into the model registry  
 mlflow.pyfunc.log\_model(artifact\_path=model\_path,  
 python\_model=vader\_model,  
 registered\_model\_name=reg\_model\_name,  
 conda\_env=conda\_env)  
  
# Load the model from the model registry and score  
model\_uri = f"models:/{reg\_model\_name}/1"  
print("model uri",model\_uri)  
loaded\_model = mlflow.pyfunc.load\_model(model\_uri)  
score\_model(loaded\_model)