Machine Learning workflow using Docker

Document type :  **POC**

Document version : **V1**

Date created : **2020**

Created by : **Vimal Dharmalingam**

Table of Contents

[1. Problem Statement: 3](#_Toc51580283)

[2. ML Solutions 3](#_Toc51580284)

[3. ML with Docker – Architect Design 4](#_Toc51580285)

[4. S3 source data structure 4](#_Toc51580287)

[4.1 Datalake Folder structure for Machine Learning 4](#_Toc51580288)

[5. Training Data integration 5](#_Toc51580289)

[6. EDA - Exploratory Data Analysis. 5](#_Toc51580290)

[6.1 Training File metadata 6](#_Toc51580291)

[6.2 Analysis metrics 6](#_Toc51580292)

[7. Model Training 6](#_Toc51580293)

[7.1 Data Preprocessing 8](#_Toc51580294)

[7.2 Model Training & Evaluation 8](#_Toc51580295)

[8. Save the Model and Metadata 9](#_Toc51580296)

[9. Inference Prediction using Docker containers 11](#_Toc51580297)

[10. Conclusion 15](#_Toc51580298)

# Problem Statement:

The objective is to build the end to end Machine Learning flow g and provide the insights in the source data using predictive modelling, Natural Language capabilities for consistent evaluation of business and to improve the customer experience.

# ML Solutions

To solve the above statement, below are the options.

1. ML flow with Apache Airflow
2. **ML flow with Docker Containers**
3. ML Flow with SageMaker.

Each method has their own pros and cons and it depends on the use case and Tools/software capabilities in Swire Shipping.

One of the primary advantages of Docker is: **build once run it anywhere,** all the required packages to run the application can be containerized and run it in dev/testing/production environment.

# 3. ML with Docker – Architect Design

# Diagram Description automatically generated

# 4. S3 source data structure

To start the workflow, created the specific folder structure in **dev-bucket** bucket

for ML in **Non-Prod Data lake** server.

### 4.1 Datalake Folder structure for Machine Learning

Since its in initial stage , all the required folders are created in a single bucket, it might be in different location in future , for example: Analytical result store will be stored somewhere in the processed data location where all the other processed data is getting stored.

**datalake-dev-bucket**

/ **ml**

/ analytical\_result\_store

/ metadata

/ training\_data

/ training\_models

/ prediction-data

/ prediction-data-backup

/ prediction-models

Under the Metadata folder below are the detailed structure to store the metadata for all the models and use cases.

**/ metadata**

/classification

/ titanic(problem\_statement\_name)

/ model\_metadata

/ trainingfile\_metadata

/ eda\_results

/ cv

/ regression

/ nlp

/ time\_series

# 5. Training Data integration

In this Proof of concept, **Titanic public dataset** is used to get the flow and manually uploaded in the training data folder.

In real time the model training data will comes from the multiple source buckets and it depends on the use case.

Before to start the ML flow the training data file should be aggregated from multiple tables and the process should be automated.

# 6. EDA - Exploratory Data Analysis.

To get the insights and to understand the data, EDA process will provide the complete insight about the training data. So that the performance of the model will be enhanced by better feature engineering process.

Once the training data is analyzed, the metadata for training file and the analysis metrics will be stored in specific S3 folders.

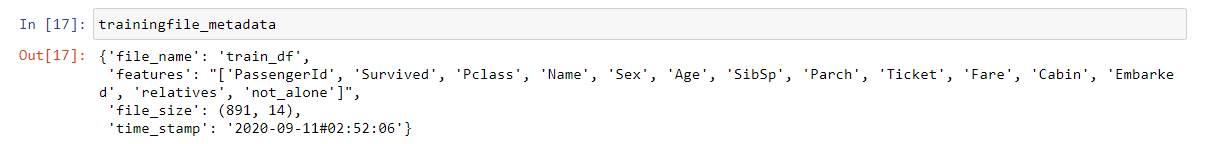
## Training File metadata

**Location**: datalake-dev-bucket/ml/metadata/classification/titanic/trainingfile\_metadata"

**Info\_available:**

1. file name
2. features
3. file size
4. time stamp

The stored file is in json format:



## Analysis metrics

The analysis metrics which produced by data analysis will be created as the PNG file to store the metrics in the specific folders.

Location: devbucket/ml/metadata/classification/titanic/eda\_results

# 7. Model Training

Below are the instance details, and the instance is considered as Analytical server. Instance can be connected through Bastion host for Model training and EDA processes.

Text

Description automatically generatedText

Description automatically generated

Diagram

Description automatically generated

## 7.1 Data Preprocessing

Once the data is analyzed, based on the data quality, need to handle the missing values, feature selection, data imbalance, data normalization and other data cleansing tasks. After the data cleansing, vectors will pass it to model development.

## 7.2 Model Training & Evaluation

Based on the problem statement, ML models will be prioritizing and trained with multiple models to get the better accuracy. In this case, it’s a classification model and tried with Logistic regression, Random forest, Naïve bay’s models and based on grid search results- random forest ensemble model generates the better result.

After the model training the best model will be stored in the specified S3 bucket.

Depends on the data imbalance and model type Evaluation metrics will differ. Since it’s a classification model, Confusion matrix is the primary metric, with that I have evaluated the model using other metrics like precision, recall, F1-score, accuracy etc.

Once the model is evaluated – evaluation metrics will get stored in the specified s3 folders

# Save the Model and Metadata

Diagram

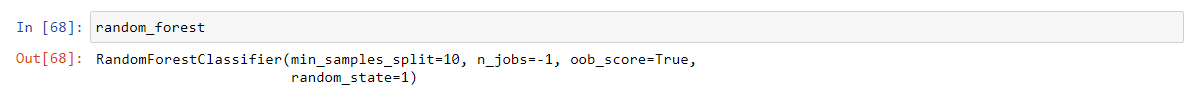
Description automatically generated

While model is training and tuning the model with different hyperparameters, trained model and their parameters are to be stored if required. The naming for the model and metadata file should follow the below format to get the details clearly in future.

***Model location:*** *dev-bucket/ml/training\_models/*

***Model file name: 'V3\_titanic\_model\_rf\_'+time+'.pkl' (file name should starts with version Id)***

Only model will get stored in training model storage as a pickle file



***Metadata file location:*** *dev-bucket/ml/metadata/classification/titanic/model\_metadata*

***Metadata file name: 'V3\_Rf\_'+time+'.txt' (file\_name should starts with same version Id which model is trained)***

**Metadata Parameters:**

1. Trained model parameters
2. Model input features
3. Specified evaluation metrics with time stamp

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

A picture containing graphical user interface

Description automatically generated

**Note:** *Since the trained model and their hyperparameters are getting stored in different folders using the version Tags (ex:V1) and file name , using version Id model and metadata can be mapped.*

The below are the processes need to be finished after completion of the model training.

# Inference Prediction using Docker containers

After model training and stored in the certain locations, Now the inference data will get predicted using the trained model.

The below figure will show the prediction process.

Diagram

Description automatically generated

***Trained\_model:*** *"V3\_titanic\_model\_rf\_2020-09-10#07:42:37.pkl"*

***Prediction\_model:*** *titanic\_prediction\_model.pkl*

***Prediction file name:*** *test.csv (in this case)*

Above is the permanent prediction file name for this particular use. So that no need to change the name in inference file. And to map the prediction file and the updated trained model, both are Tagged while updating the model.

Below is the Linux instance used for Docker.

Text

Description automatically generated

Check the Docker availability and version of the docker container



Using Winscp move the required folder and other files to Linux instance and check the folder structure as same as the source.

A screenshot of a computer

Description automatically generated

Build a Docker image using below docker command and docker built the image successfully.

Text

Description automatically generated

Check all the docker images available in docker.

A screenshot of a computer

Description automatically generated with medium confidence

Now run the container using docker image which is build already and ready for running.

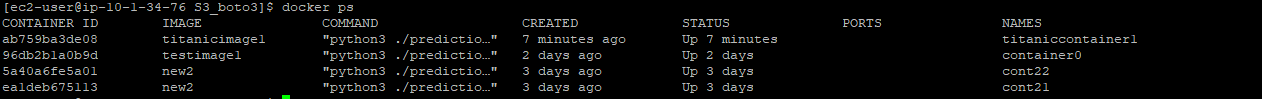
Text

Description automatically generated

Text

Description automatically generated

“Docker ps” is the command to see the running containers.



The Docker container will run always and check the file availability for each 5 seconds (can change based on requirement) once the prediction file is pushed to “prediction-data” folder in s3. Docker container will pickup the file and predict.

Once the prediction is done automated code will backup the file to “prediction-data-backup” folder and delete the file in prediction data location. So that, it will eliminate the prediction of the same file again and again.

To predict the batch inference, trained model from prediction storage, inference data from prediction data location and prediction code should be required.

In prediction code, All the preprocessing steps which involved in training process should be included.

In Airflow server, connection is made between the S3 location and Airflow to read write the data from S3.

The results which produced by the prediction model is getting stored in the Analytical result store folder in S3.

# 10. Conclusion

Once the model is developed and tested, the code will be committed to AWS code commit, (datalake\_ml). Below are the primary python files should get committed with Code commit.

**Folder/File Name:**

1. titanic\_eda.ipynb

2. titanic\_training.ipynb

3. Folder: S3\_boto3 (for prediction)

From analytical results store, the data can be consumed directly by user and visualize the data in PowerBI.