Wings1 T15 Cloud Analytics and AI

Challenge Overview:

This challenge comprises two distinct parts: Analytics and Machine Learning. You will work with CSV datasets stored in an S3 bucket named "car-data" with a unique prefix followed by a random number. Within this bucket, there are two essential folders:

1. Inputfile: This folder contains the car_data.csv dataset, which you will utilize for the Analytics part. Using car_data.csv, you will be cleaning the data and loading the data to s3. Also you will be performing various analytics tasks based on the cleaned data.
2. Car_cleaned_data: This folder contains the car_cleaned_data.csv dataset, designated for the Machine Learning tasks.
3. Analytics and ML pipeline:
4. Analytics: S3 -> EMR -> S3 (45mins)
5. Machine Learning: S3 -> SageMaker -> s3. (45 mins)

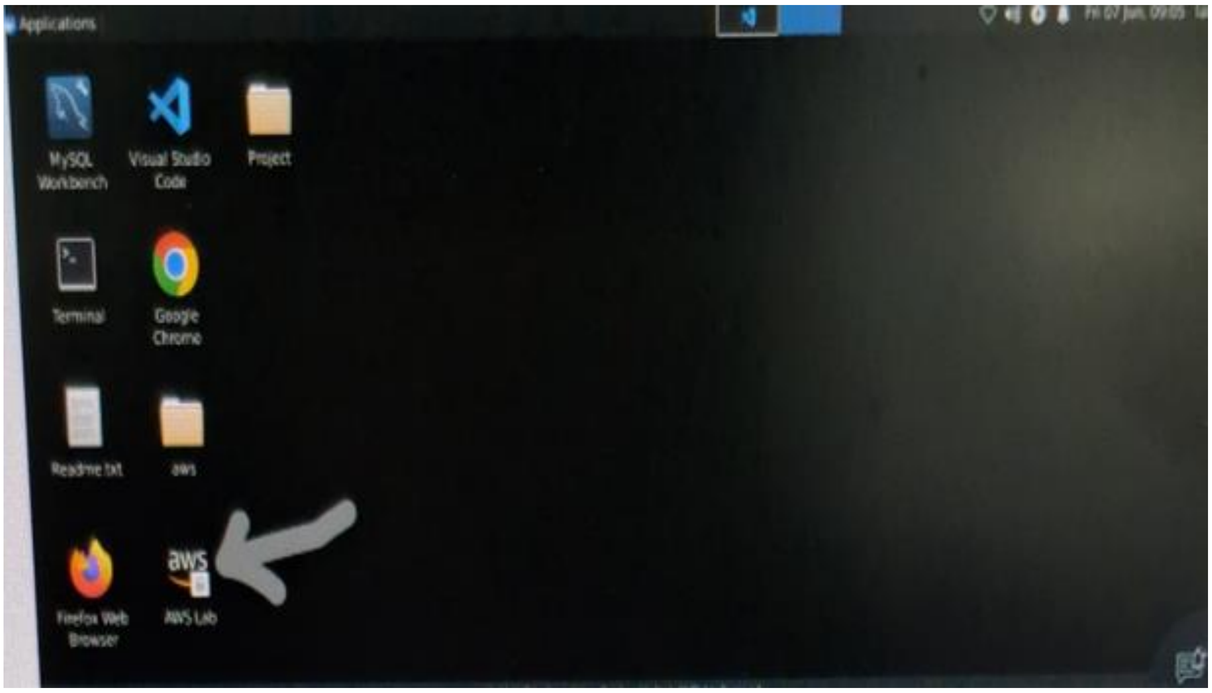
Note:

- Don't worry about the csv files present in the local folder, only use the csv file from the s3 bucket itself for the tasks outlined in this challenge.
- You can do any of the parts first, both parts(analytics and machine learning) are independent.

Your mission is to tackle each part systematically, ensuring efficient data processing and preparation for both analytics and Machine Learning endeavors. Let's dive into the specifics of each task!

Import note: Use US East (N.Virginia) us east 1 Region only

<u>How to login into AWS Account</u>



Click on the above mentioned icon on your desktop to be redirected to the AWS login page in the Firefox web browser. Here, you will find the username and password for the AWS access page. Click on the ‘Access Lab’ button to be redirected to a new page. Enter the credentials provided on the previous page

<u>Analytics</u>

Note: In this challenge you will be using EMR cluster. Before proceeding wit the challenge. Crete an EMR cluster. It will take 8 to 10 minutes to create the cluster. While it was creating the cluster. You can read the problem statement and write the code in the template given in the project folder.

<u>EMR Cluster</u>

EMR cluster creation:

Follow the configuration below for creating the cluster.

Name: spark_cluster

EMR version: 7.1.0

Application Bundle: select the custom aws

- Hadoop 3.3.6
- Hive 3.1.3
- Livy 0.8.0
- Spark 3.5.0

Cluster Configuration:

- Uniform instance Groups
- Keep only primary node, remove the other two node by clicking Remove Instance group. Primary node EC2 instance type should be m4.large

EBS root Volume: Kee the default sizes.

Networking: Default

Steps: Leaving as it is.

Cluster termination and node replacement: Terminate option – Manually terminate the cluster.

Bootstrap actions: Default

Cluster Logs: Default

Tags: Default

Software Settings: Default

Security configuration and EC2 key pair: Create a key pair named “emr_spark”

- Key pair type – RSA
- Private key file format -.pem format

And download this inside location `/home/labuser/Desktop/Project/Kickoofs-cloud_analytics_and_ai-wingsT15-car_data/emr_spark.pem

Identity and Access Management (IAM) roles:

Amazon EMR service role- click on create new service role

EC2 instance profile for Amazon EMR – create an instance profile

S3 bucket access – All S3 bucket in this account with read and write access

Custom automatic scaling role: Default

Leave the other options as it is and click on create cluster (it will take 6 to 8 mins to create the cluster). Meanwhile you can read the problem statement and start coding.

Templates:

You can solve this analytics part using either pyspark or scala. You are provided with python template in challenge.py file and challenge.scala file. You need to complete the code in the template using pyspark or scala and push the file to EMR using SSH and do a spark-submit or sbt run. Otherwise, you can use the EMR add step option to run the file.

<u>Problem Statement</u>

The following functions are given in the python and scala templates.

Task1:

Read_data

Complete the following operations in the read_data function

The following are the parameters:

Spark session – spark

Mention the bucket name inside the bucket_name variable

The dataset will be in the s3 location inside the input file foldet.

Read the CSV file into a dataframe. Make sure to give only header as true

In the challenge file, the return statement is already defined, and you need to replace the df with your final output dataframe.

Task2:

Clean_data

Complete the following operatins n the clean_data function

The following are the parameters:

Output of read_data function – input_df

Drop the null values if it is having in any of the columns.

Drop the null values if it is having in any of the colums

Drop duplicate rows

In the challenge file, the return statement is already defined, and you need to replace the df with your final output dataframe.

Task3:

S3_load_data

Complete the following operation in the “S3_load_data” function

The following are the paameters:

Final dataframe to load the data: data (the final dataframe of the following functions)

File name for the results: file_name

Mention the bucket name inside the bucket_name variable.

Write a code the store the outputs to the respective locations using the output_oath param.

Output files should be a single partition csv file with header.

Output of clean_data function – input_df

Add a new column price_per_km calculated as selling_price/km_driven

Filter the data to include only rows where price_per_km is less than 10.

In the challenge file, the return statement is already defined, and you need to replace the df with your final output dataframe.

Sample Output

Car_name	Year	Selling_price	Km_driven	Fuel	Seller_type	Transmission	Owner	Price_er_km
Maruti Alto LX	2008	65000	140000	Petrol	Individual	Manual	First Owner	1.0

Note:

The column names (column names are case-senstive) and order should be same as given in the sample output for each task.

Important

You have been given two ways to run the spark code i.e. either inside the emr cluster or by using step function

Jump to that heading to complet the task.

Inside the EMR Operations

Step to be followed:

Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

In the navigation pane, choose instance which is created by emr.

Select the instance and go to the securith sectin and add the ssh port to the ec2 created by emr.

Select the instance and choose connect.

Choose the EC2 instance connect tab.

For connection type, choose connect using EC2 instance connect

For username, should be root.

Choose connect to open a terminal window

It will open a new browser tab with the emr cluster

Now run the emr_copy.py file using python emr_copy.py in vscode inside the /home/labuser/Desktop/Project/kickoffs-cloud_analytics_and_ai-wingsT15-car_data/, it will copy you updated file to the emr cluster in the location /home/Hadoop/.

[Note: Make sure the keypair is located correct /home/labuser/Desktop/Project/kickoffs-cloud_analytics_and_ai-wingsT15-car_data/emr_spark.pem.]

Go to the /home/hadoop/ directory you will find the files which you have uploaded.

Run the setup.sh file inside the /hom/hadoop/setup dir by “bash set up.sh”.

After successfully setup of spark you can submit the spark application.

For Pyspark:

spark-submit challenge.py

For Scala:

Enter inside the scala directory

sbt run

<u>EMR Step Function</u>

Follow the steps

For pyspark:

Pus the challenge.py file which is present inside the python directory into the s3 locaiton

For Scala:

First enter into the scala fir

Enter the command “sbt_package” it will give the challenge file as a jar file inside the target/scala*/directory.

Push the jar file into the s3 location

Now open the Amazon EMR console at https://console.aws.amazon.com/emr.

In the cluster list, select your cluster

Sroll to the steps section and expand it, then choose add step.

In the Add step sialog

For step type choose Custom JAR

For name type any name

For JAR S3 location – “command-runner.jar”

For Arguments

Mention this command

For scala – “spark-submit -class main <s3_file-path>

For pyspark – “spark-submit <s3_fiel_path>*

For action on failure, accept the default option(continue).

Choose Add. The step appears in the console with a status of pending.

<u>Machine Learing: Predicting car selling prices</u>

In this problem, you will build a machine learning model to predict the selling price of used cars. You are given a dataset with various features related to the cars, such as the car’s) age, mileage, fuel type, etc. Your goal is to preprocess the data. Select appropriate features and train a model to make accurate price predictions. The final mode should achieve an R2 score significantly higher than the baseline.

Dataset descrition:

The dataset contains the following columns:

Car_name: Name of the car.

Year: Year the car was manufactured.

Selling_price: Price at which the car was sold (target variable).

Km_driven: Kilometers driven by the car.

Fuel: Type of fuel used by the car.

Seller_type: Type of seller(individual/dealer)

Transmission: Type of transmission (manual/automatic).

Owner: Number of previous owners.

Perform the following Cloud Driven Task:

Task1: Launch an Amazon SageMaker Instance

Objective: Lanch an Amazon SageMaker notebook instance to develop and deploy machine

Learning models.

Instuctions:

Access Amazon Sagemaker

- Access the AWS Management Console.
- Navigate to the SageMaker service under the “Machine Learning” category.

Create a New Notebook instance:

- Create a new notebook instance under the name ‘car-prediction-notebook’.
- Choose the instance type ‘ml.t3.medium’.
- Configure the necessary permission to access s3 buckets by newly creating an appropriate role.
- Create the notebook instance.

Access the note book:

- Wait for the notebook instance status to change to ‘inservice’.
- Open the Jupyter dashboard from the SageMaker consol to begin your machine learning project.

<u>Perform the following ML Tasks:</u>

Task1: Load and Explore the Dataset

Instructions:

1. Import AWS SDK ‘boto3’ and other necessary libraries such as Numpy, pandas & Sklearn on your notebook
2. Load the dataset from S3 bucket. Build the S3 path for the dataset ’car_cleaned_data.csv’using string formatting to concatenate the bucket name. folder name (if any) and file key i.e the name of the dataset. Note: Bucker name- ‘car-dataXYZXYZ’ (XYZXYZ can be any random integers) & Folde name – ‘car_cleaned_data’.
3. Load and store the dataset using pandas DataFrames.
4. Print the first few rows of the dataset to understand its structure and use inbuild functions to get insights on the dataset.

Tase2: Feature Engineering

Instructions:

1. Create a new feature ‘car_age’ which represents the age of the car in years. Add the feature as a column under the name ‘car_age’ in the dataframe.
2. Drop the columns ‘car_name’ and ‘year’ for the dataset

Hints:

The car’s age can be calculated as 2024- years.

Task 3: Define Features and Target Varaible

Instructions:

Define the features ‘x’ by dropping the target variable (selling_price) from the dataset.

Define the targer variable ‘Y’ as the selling_price.

Task 4: Preprocess the Data

Instructions:

1. Identify numerical and categorical features present in “X” and “Y”. Store it in a list for building a pipeline.
2. Apply log transformation to the km_driven and selling_price columns to reduce skewness. Replace the transformed data back to “X
 & “Y”.
3. Use StandaradScaler to scale the numerical features.
4. Use OneHotEncoder to encode the categorical features, ensuring that unknown categories are handled.

Hints:

1. Use ‘np.loglp()’ for log transformation
2. Example numerical feature list: “numerical_features = [‘xy’,’yz’]”
3. Example categorical feature list: “categorical_features = [‘xy’,’yz’,’zz’]”

Task 5: Build a Transformer Pipeline

Instructions:

1. Create a ColumnTransformer pipeline to preprocess the numerical and categorical features.
2. Create a pipeline that includes the preprocessors i.e ‘StandardScler’ & ‘OneHotEncoder’, Store the transformer pipeline in a variable, for example ‘preprocessor’. We will utilize this for building the model pipeline in the upcoming steps.

Hints:

1. Sample Pipeline:
```
Preprocessor = ColumnTransformer(transformers=[
                    (‘xy’,scaler(),feature_list1),
                    (‘yz’,encoder(), feature_lis2)
                    ])
```

Task 6: Build a Model Pipelne and Split the Data

Instructions:

Create a Sklearn model pipeline with the preprocessor pipeline and ‘RandomForestRegressor’ model with a random state set to ‘8’. Store it under the name ‘model’.

Split the dataset into training and test sets using ‘train_test_split’ with a test size of 20% and random state set to ‘8.

Hints:

1 Sample pipeline:

```
Model =  Pipeline(steps=[(‘preprocessor’,preprocessor),
                         (‘model’,ModelFunc[arg=value))
                         ])
```

Task7: Hyperparameter Tuning

Instructions:

Perform hyperparameter tuning using GridSearchCV to find the best parameters for the RandomForestRegressor. The parameter grid is provided in the sample notebook and in the hint section.

Build the Grid Search named ‘grid_search’ with the generated model pipeline parameter grid, cross-validation generator set to ‘5’, scoring set to ‘r2’ and number of jobs set to ‘1’.

Hints:

Parameter grid:

Param_grid ={

‘regressor__n_estimators’:[100,200,300],

‘regressor__max_depth’:[None,10,20,30],

}

Task10: Evaluate the Model

Instructions:

1. Calculate the Mean Absolute Error(MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 score for the model.
2. Print the evaluation metrics and the best hyperparameters found.

Hints:

Use ‘grid_search.best_params_’ to get the best hyperparameter.

Task 11: Saving the Model to AWS S3

Instrucitons:

1. Serialize the trained GridsearchCV model using ‘joblib’.
2. Initialize the S3 client using the ‘boto3’ library.
3. Save the serialized model to a temporary file using ‘tempfile’.
4. Upload the model file to the specified S3 bucket named ‘car-data’
5. Ensure the model is saved as ‘grid_search.pkl’ in the S3 bucket.

Hints:

1. Temporary files in python can be managed using ‘tempfile.TemproraryFile().’
2. Use ‘Joblib.dump() for saving the model.
3. We can push objects into S3 using .put_object(…) method with necessary parameters available under boto3.

Once you complete the challenge, make sure all the files are saved in the specified directory and click on SUBMIT.