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.

Analytics and ML pipeline:

Analytics: S3 -> EMR->S3 (45mins)

Machine Learning: S3-> SageMaker -> S3. (45 mins)

Note:

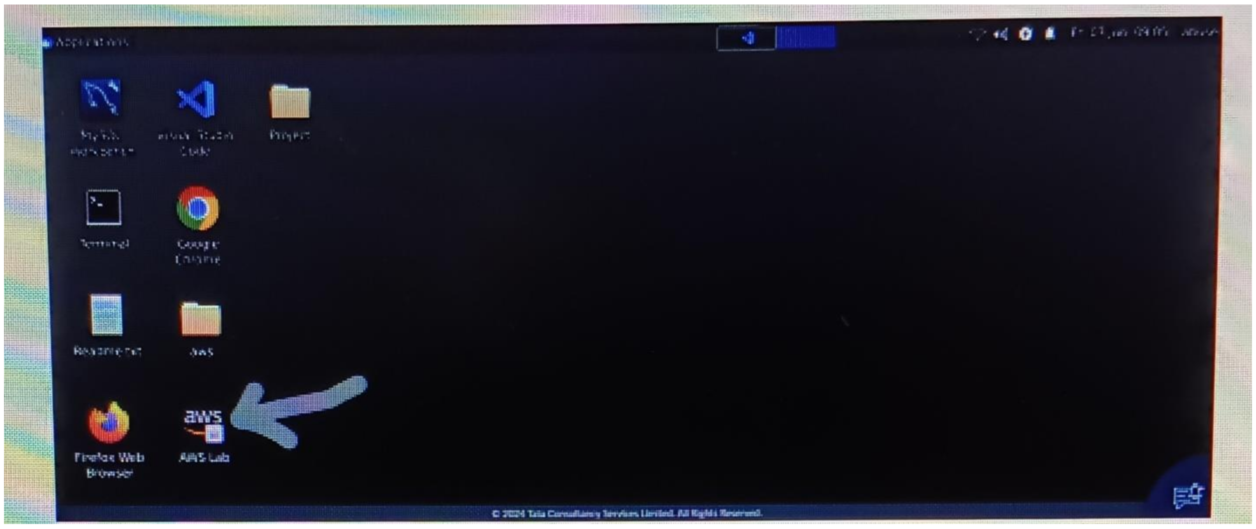
- Don't worry about the CSV files present in the local folder, only use the CSV files from the s3 bucket itself for the tasks outlined in this challenge
- You can do any of the parts first, **both parts (analytics and machine learning) are independent**.

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!

IMPORTANT NOTE: Use **US East (N. Virginia) us-east-1** Region only.

Note: Please follow the naming conventions which are given in the problem statement.

How to login into AWS Account


Terminal
Google Chrome
AWS
Firefox Web Browser
AWS Lab
© 2024 Tata Consultancy Services Limited. All Rights Reserved.

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.

Analytics

Note: in this challenge you will be using EMR cluster. Before proceeding with the challenge, create 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.

EMR Cluster

EMR cluster creation:

Follow the configurations 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: Keep the default sizes.

Networking: Default

Steps: Leave as it is.

Cluster termination and node replacement: Termination Option - Manually terminate the cluster.

Bootstrap actions: Default

Cluster Logs: Default

Tags: Default

Software Settings: Default

Security configuration and EC2 key pair: Create a keypair named "**emr_spark**"

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

And download this inside location ~/home/labuser/Desktop/Project/kickoffs-cloud_analytics_andai-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 buckets 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.

Open the challenge folder which is present in the Project folder in vs code, then if you are using python complete the code in the **challenge.py** template. If you are using scala, complete the code in **challenge.scala** file.

Problem Statement

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 locaiton inside the inputfile folder.
- 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 operations in the clean data function

The following are the parameters:

- Output of read_data function-**input_df**

1. Drop the null values if it is having in any of the columns.
2. Drop duplicate rows
3. 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 operations in the “S3 load_data” function

The following are the parameters:

- 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_path param.

• Output files should be a single partition CSV file with header.

Task4:

result_1

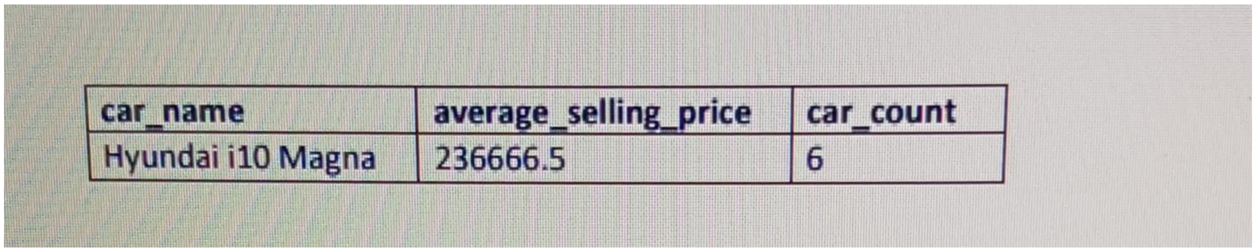
Complete the following operations in the result_1 function

The following are the parameters:

Output of clean data function-**input_df**

- Group the data by “**car_name”**.
- Calculate the average selling price for each brand and store it in the new column **average_selling_price.**
- Calculate the count of cars for each brand and store it in the new column **car_count**.
- Filter the grouped data to include only brands with **more than 2 cars.**
- 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	average_selling_price	car_count
Hyundai i10 Magna	236666.5	6

Task5:

result_2

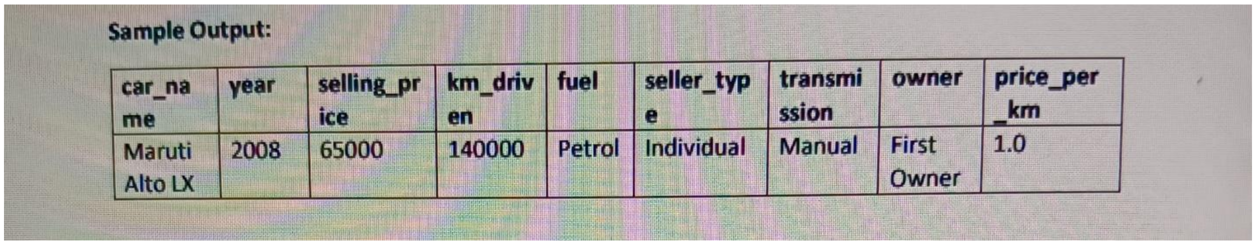
Complete the following operations in the **result_2** function

The following are the parameters:

- 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:



Sample Output:

car_name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	price_per_km
Maruti Alto LX	2008	65000	140000	Petrol	Individual	Manual	First Owner	1.0

Note:

The column names (column names are case-sensitive) 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 emr step function.

- Jump to that heading to complete the task.

Inside the EMR Operations

Steps to be followed:

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/
2. In the navigation pane, choose **Instances** which is created by emr.
3. Select the instance and go to the security section and add the **ssh port** to the ec2 created by emr.
4. Select the instance and choose **Connect.**
5. Choose the **EC2 Instance Connect tab**.
6. For Connection type, choose **Connect using EC2 Instance Connect.**
7. For **Username**, should be root.
8. Choose **Connect** to open a terminal window.
9. 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 your 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 **/home/hadoop/setup** dir by "**bash setup.sh**".

[NOTE: If the sbt or pyspark is not working then run the command **"source ~/.bashrc"]**

- 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

EMR Step Function

Follow these steps:

For **Pyspark**:

- Push the **challenge.py** file which is present inside the **python** directory into the s3 location.

For **Scala**:

- First enter into the scala dir
- 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.
- Scroll to the **Steps** section and expand it, then choose **Add step**.
- In the **Add Step** dialog:

  • 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_file_path>"**

• For **Action on failure**, accept the default option **(Continue)**.

• Choose **Add**. The step appears in the console with a status of Pending.

Machine Learning: Predicting Car Selling Prices

In this problem, you will build a machine learning model to predict the selling prices 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 model should achieve an R^2 score significantly higher than the baseline.

Dataset Description:

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:

Task 1: Launching an Amazon SageMaker Instance

Objective: Launch an Amazon SageMaker notebook instance to develop and deploy machine learning models.

Instructions:

1. Access Amazon SageMaker:

   -Access the AWS Management Console.

   -Navigate to the SageMaker service under the 'Machine Learning' category.

2. 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 permissions to access S3 buckets by cnewly creating an appropriate role.
   - Create the notebook instance.

3. Access the notebook:

   - Wait for the notebook instance status to change to 'InService'.

-Open the Jupyter dashboard from the SageMaker console to begin your machine learning project.

Perform the following ML Tasks:

Task 1: 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**: Bucket name 'car-dataXYZXYZ (XYZXYZ can be any random integers) & Folder 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 inbuilt functions to get insights on the dataset.

Hints:

1. Sample S3 URI – "s3://bucket_name/folder_name/file_name.csv"

Task 2: 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' from the dataset

Hints:

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

Task 3: Define Features and Target Variable

Instructions:

1. Define the features “X’ by dropping the target variable (selling price) from the dataset
2. Define the target 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 trasformed data back to ‘X’ & ‘V.
3. Use StandardScaler to scale the numerical features.
4. Use OneHotEncoder to encode the categorical features, ensuring that unknown categories are handled.

Hints:

1. Use ‘np.log1p() 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 ‘Column Transformer pipeline to preprocess the numerical and categorical features.

2. Create a pipeline that includes the preprocessors ie “StandardScaler & “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_list2)

])

Task 6: Build a Model Pipeline and Split the Data

Instructions:

1. Create a Sklearn model pipeline with the preprocessor pipeline and ‘Random ForestRegressor’ model with a random state set to ‘8’. Store it under the name ‘model’.
2. 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:

Sample pipeline:

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

Task 7: Hyperparameter Tuning

Instructions:

1. Perform hyperparameter tuning using GridSearchCV to find the best parameters for the Random ForestRegressor. The parameter grid is provided in the sample notebook and in the hint section.
2. Build the Grid Search named grid search with the generated model pipeline, parameter grid, cross- validation generator set to 'S', scoring set to '12' and number of jobs set to "1".

Hints:

1. Parameter grid:

Param_grid = {

'regressor_n_estimators': [100, 200, 300),

'regressor_max_depth': [None, 10, 20, 30],

'regressor_min_samples_split': [2, 5, 10)}

2. Define a parameter grid as given above with different values for in_estimators, max_depth, and min_samples_split.

Task 8: Train the Model.

Instructions:

1. Fit the GridSearchCV on the training data to find the best model *(Note: Fitting process may take a few minutes to complete the execution, please wait until the process is finished)*
2. Extract the best model from the grid search results. Store the best model under variable for example 'best model'.

Hints:

1. Use 'grid_search.best_estimator_' to get the best model.

Task 9: Make Predictions

Instructions:

1. Use the best model to make predictions on the test set and store it in variable called 'y_pred".
2. Transform the predictions ie 'y_test' and 'y_pred' back to the original scale.

Hints:

1. Use np.expm1() to reverse the log transformation.

Task 10: 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:

1. Use 'grid_search.best_params_' to get the best hyperparameter.

Task 11: Saving the Model to AWS S3

Instructions:

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 53 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.TemporaryFile()."
2. Use joblib.dump() for saving the model.

3. We can push objects into $3 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.