AWS data pipeline – Used Car price Analysis

Introduction

This project implements a comprehensive data pipeline leveraging AWS cloud services and PySpark to analyze and predict used car prices from a large-scale dataset. The pipeline orchestrates multiple components, starting with raw data storage in S3, followed by data processing using Jupyter Notebooks for cleaning and feature engineering. The processed data then flows through parallel paths - one for SQL analysis using Python scripts, and another for machine learning model development using Amazon SageMaker AutoML. The pipeline culminates with AWS QuickSight for data visualization and insights delivery to end users. This architecture ensures efficient data processing, scalable machine learning operations, and interactive visualization capabilities, all while maintaining data integrity and processing efficiency throughout the workflow. The implementation demonstrates the effective use of cloud-native services for handling complex data processing and machine learning tasks in a production environment.

Dataset overview

The dataset in question is a comprehensive collection of used car listings, encompassing a wide range of vehicle attributes and market information. With a size of 100 MB, it provides a rich source of data for analyzing the used car market and developing predictive models. The dataset captures various aspects of each vehicle, including basic information like make and model, technical specifications, aesthetic features, geographic details, and pricing information. This dataset serves as the foundation for building a machine learning model to predict the prices of used cars. It offers a diverse set of features that can potentially influence a vehicle's value, making it ideal for exploring complex relationships between car attributes and market prices. The inclusion of both structured data (such as numeric and categorical fields) and unstructured data (like descriptions) allows for the application of various data analysis and machine learning techniques, enabling a comprehensive approach to price prediction in the used car market.

Project Objective

This project implements a comprehensive big data pipeline leveraging AWS cloud services and PySpark to analyze and predict used car prices from a large-scale dataset. The pipeline orchestrates multiple components to efficiently process, analyze, and visualize data from raw ingestion to actionable insights.

Methodology

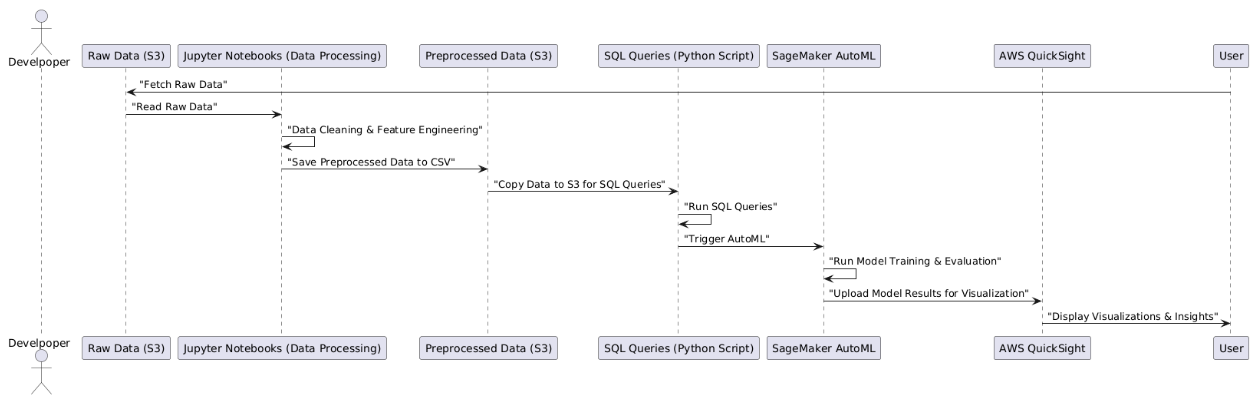


Fig 1- Architecture Sequence diagram of the Data pipeline

<u>Environment Setup:</u>

I have used the boto3 library of python provided by AWS to interact with the AWS services programmatically.

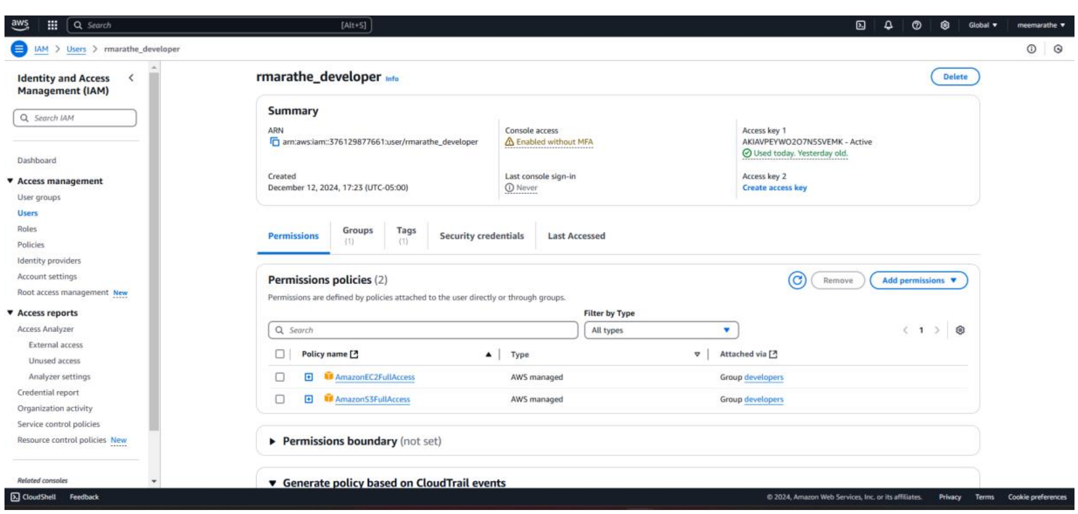


Fig 2 – Setting up of permission for user to access S3 and EC2 instance.

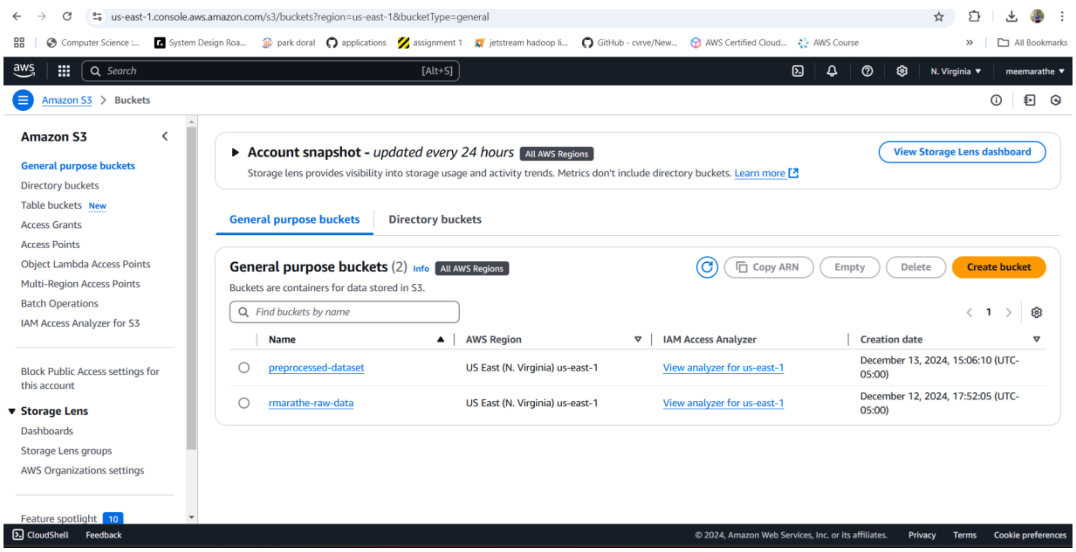


Fig 3 – The uploaded csv files are present in the S3 buckets seen above.

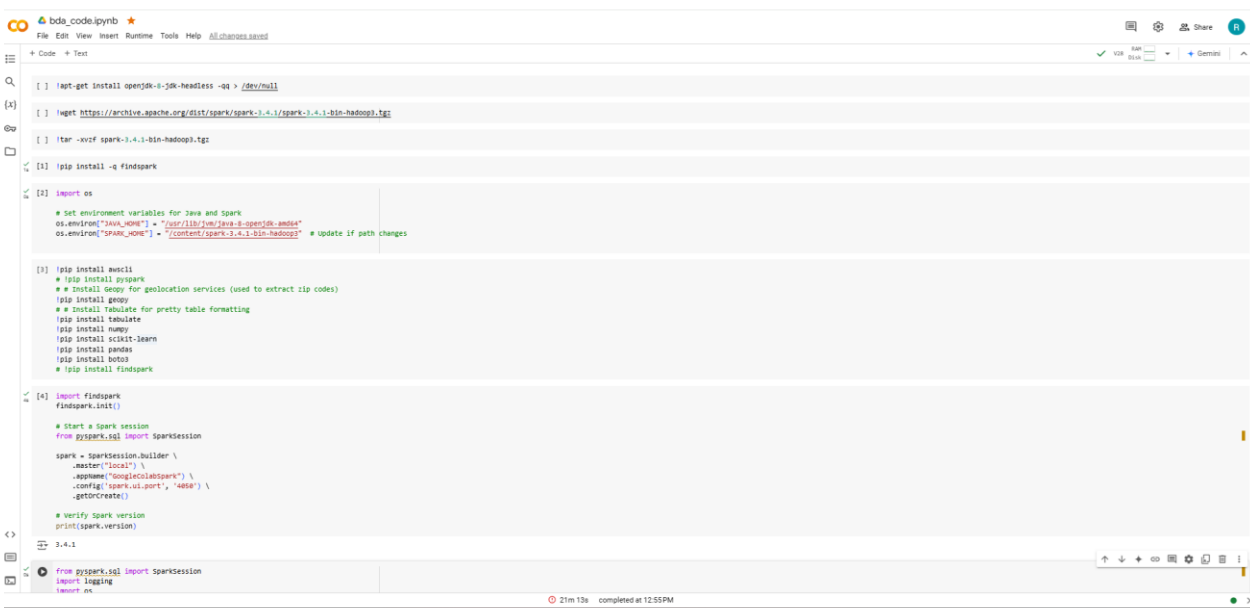


Fig 4 – Installing the libs and downloading the python, pyspark, hadoop packages to match the right version.

Data Ingestion and preprocessing

I have used Google collab to run the data ingestion from AWS S3 using boto3 package and preprocessing using pyspark.

[**Challenge** faced when tried with EC2 due to resource limitation for free tier as the dataset was large]

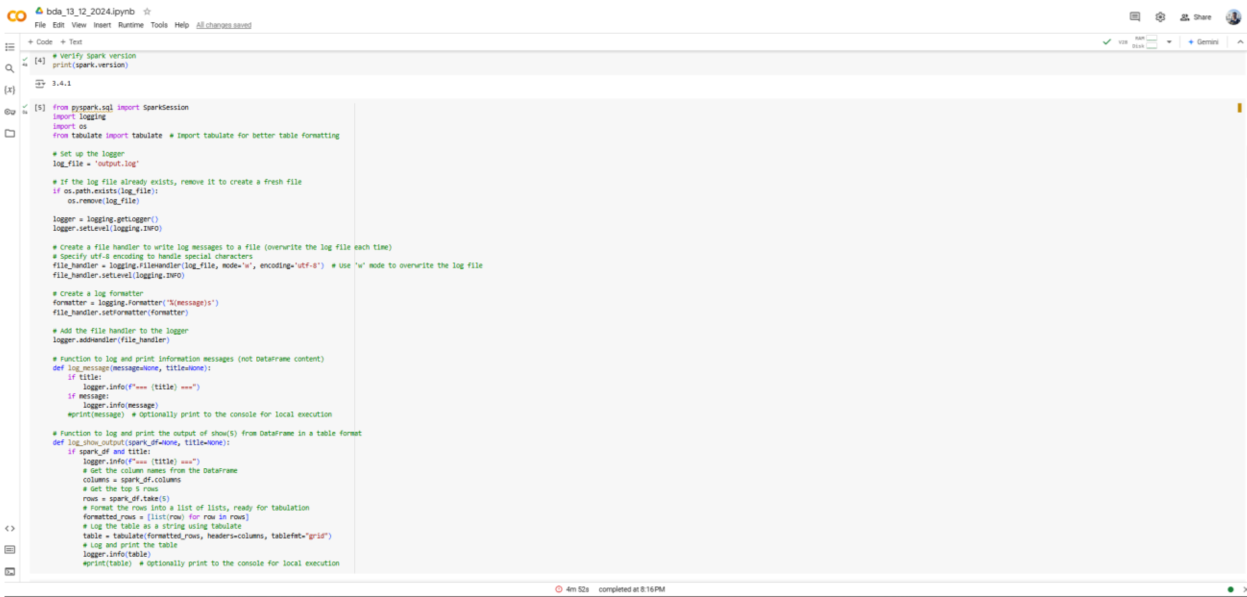


Fig 5- Logging enabled for debugging purposes – logs are dumped to a file

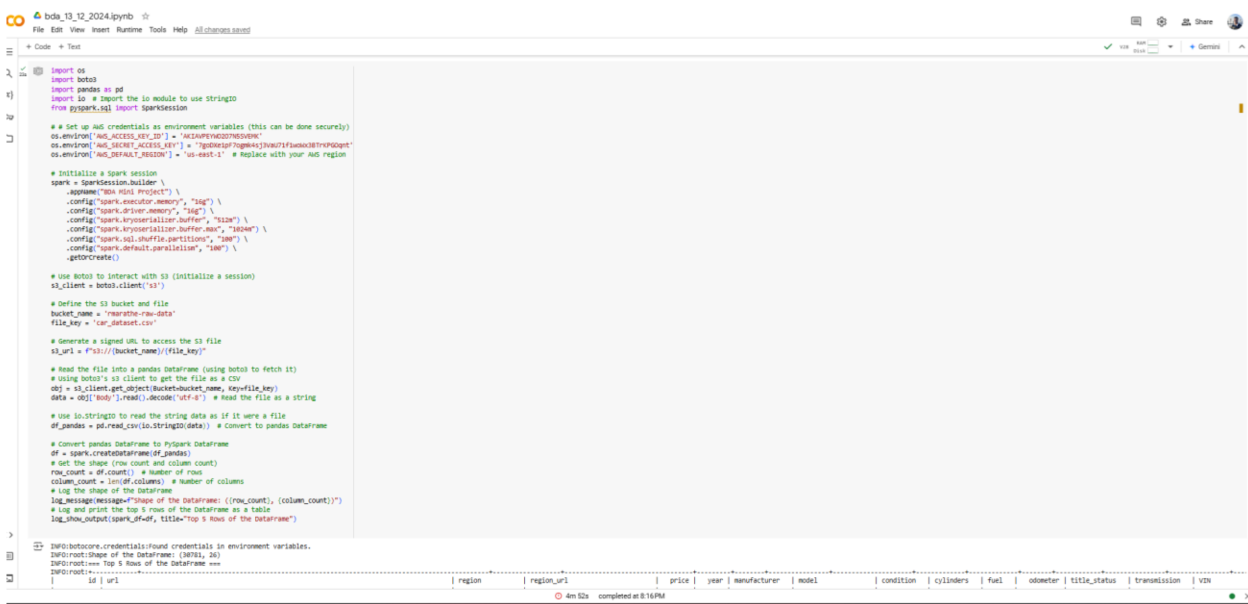


Fig 6 – Setting up the spark session to interact with dataset in AWS S3 bucket and reading raw file

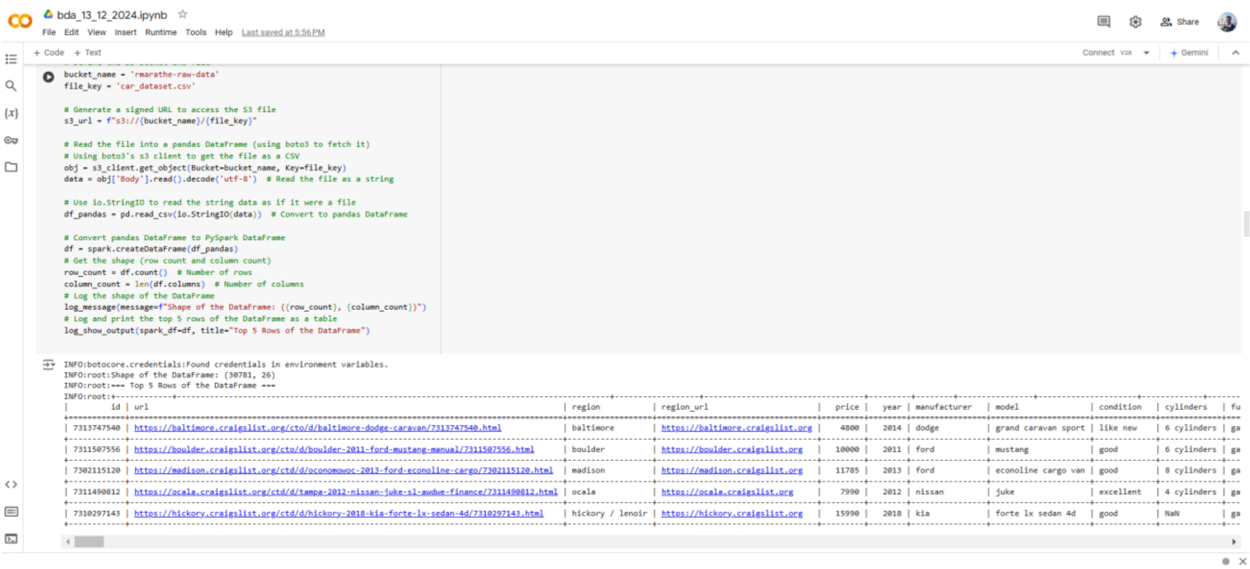


Fig 7 – Data ingested successfully and displayed in the terminal.

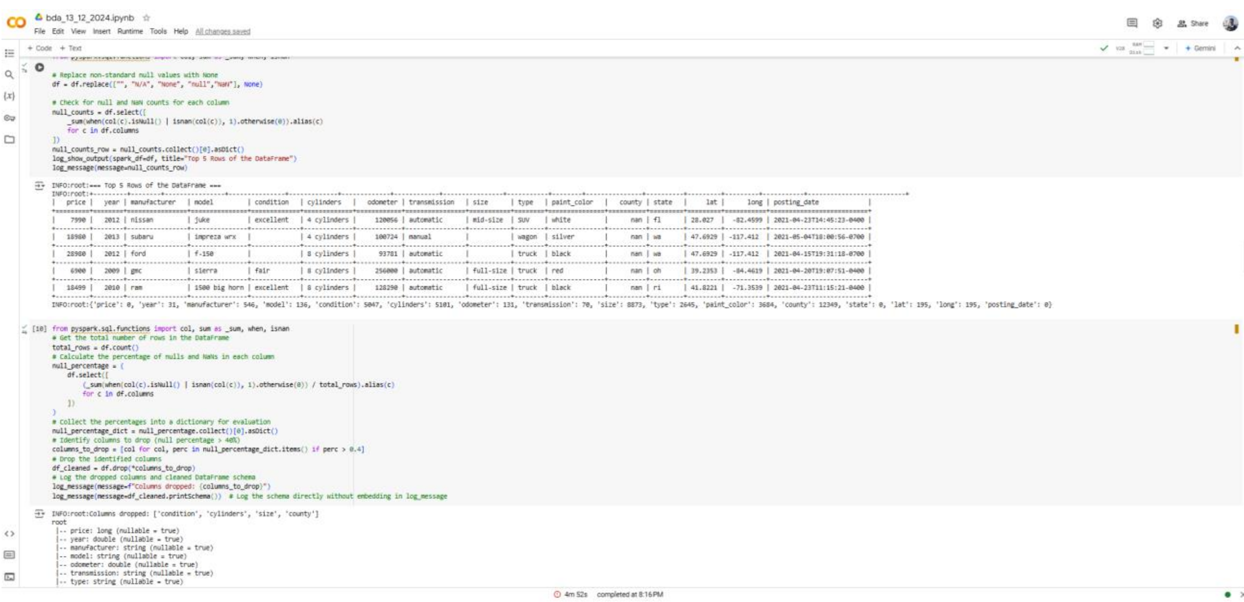


Fig 8 – code to check the nulls in all the columns.

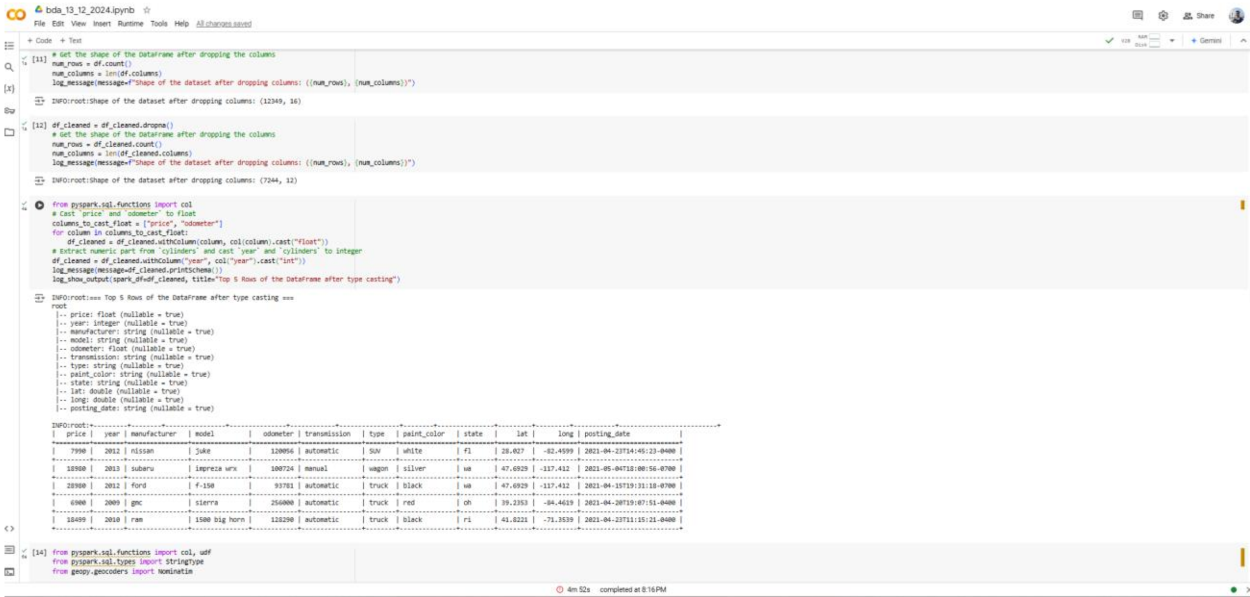


Fig 9 – rows with nulls dropped and data type casting performed

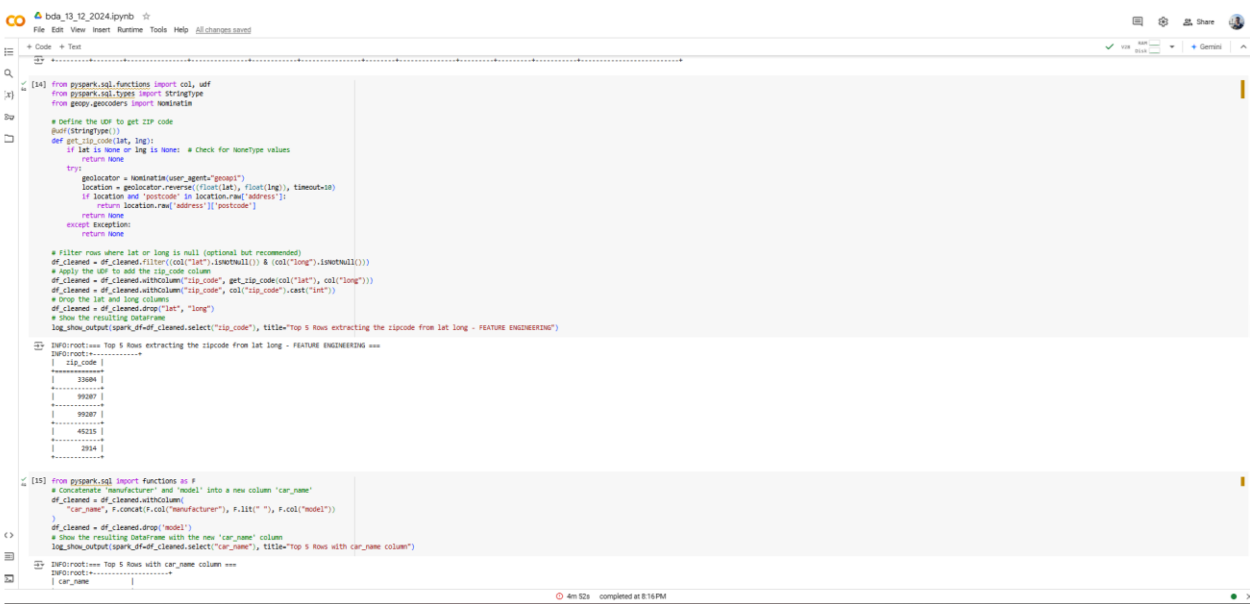


Fig 10 – Feature engineering – using lat and long along with geopy, extracted the zip codes.

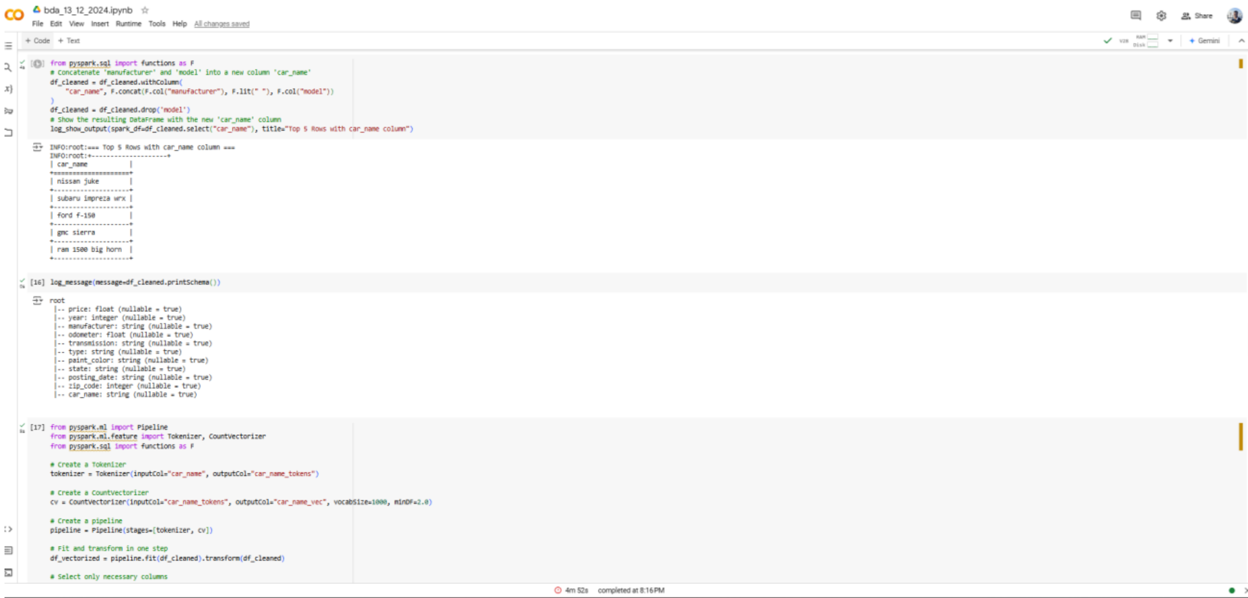


Fig 11 – Feature Engineering - Merged the columns manufacturer and model into new column car_name.

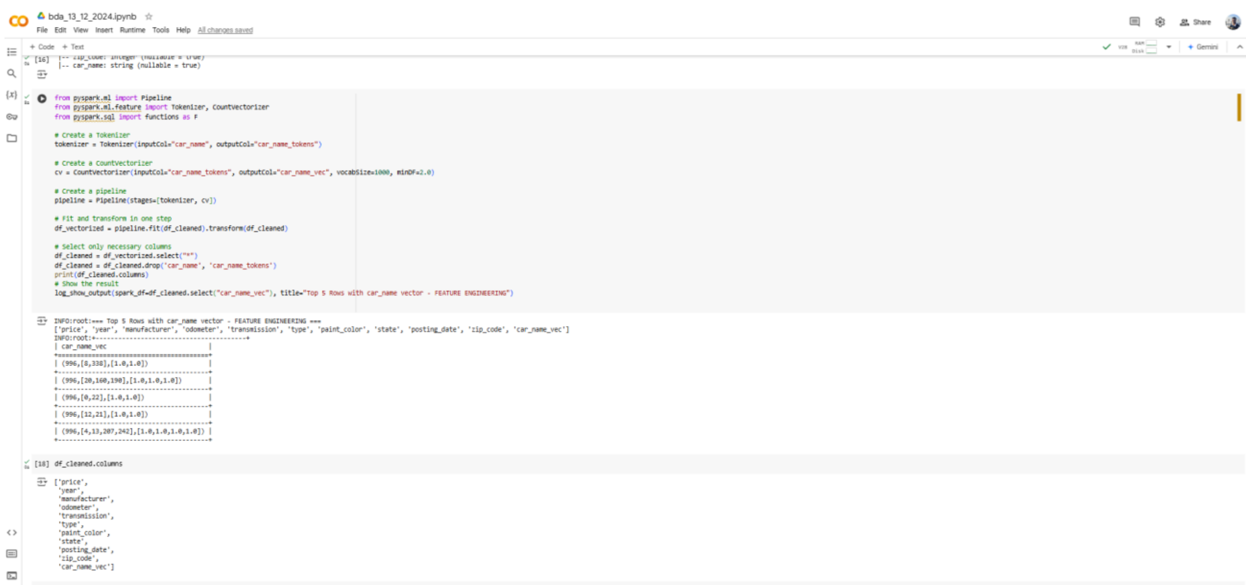


Fig 12 – Performing tokenization, vectorization using HashingTF on the car_name column.

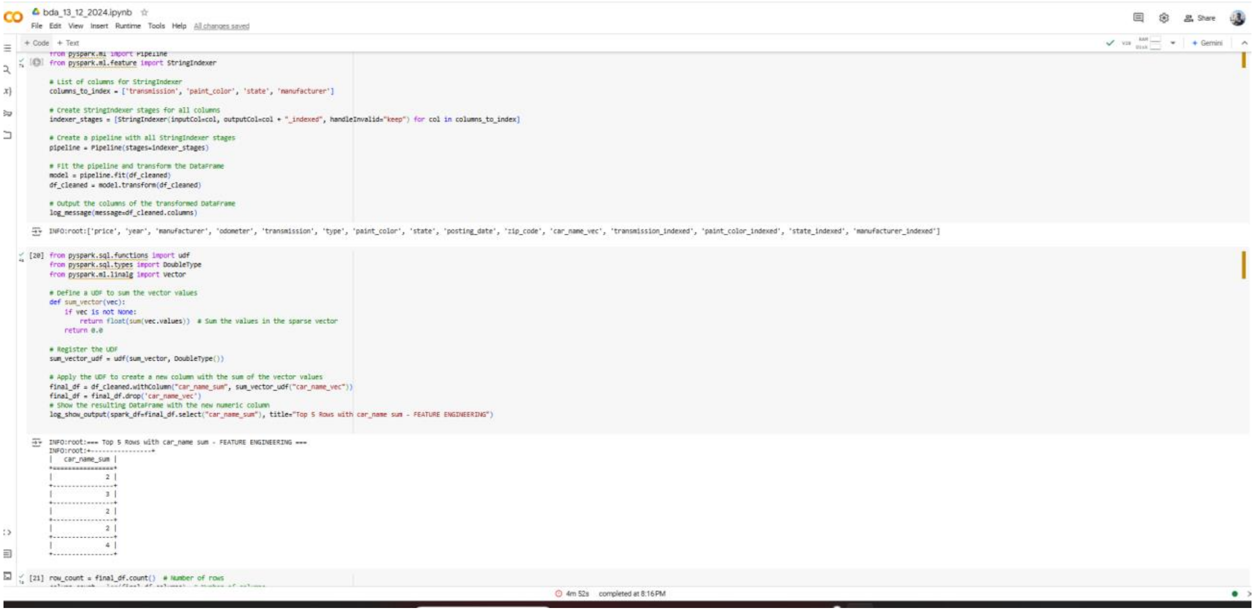


Fig 13 – Handling the categorical columns with string indexer.

Fig 14 – Data aggregation to gain some insights

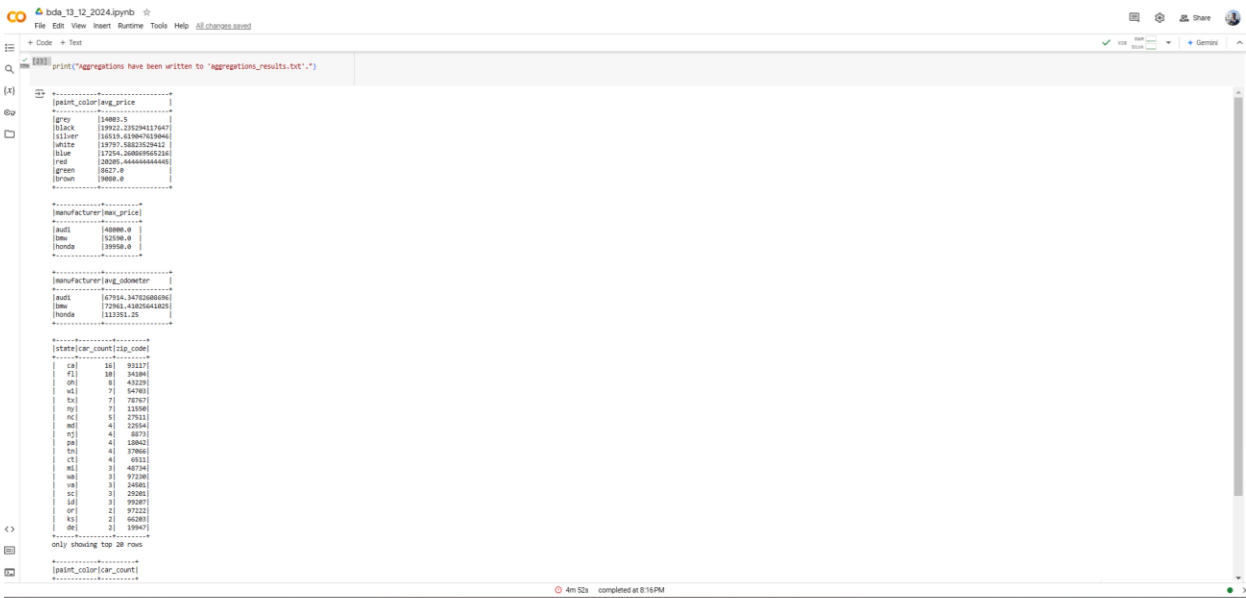


Fig 15 – Output of the data aggregation.

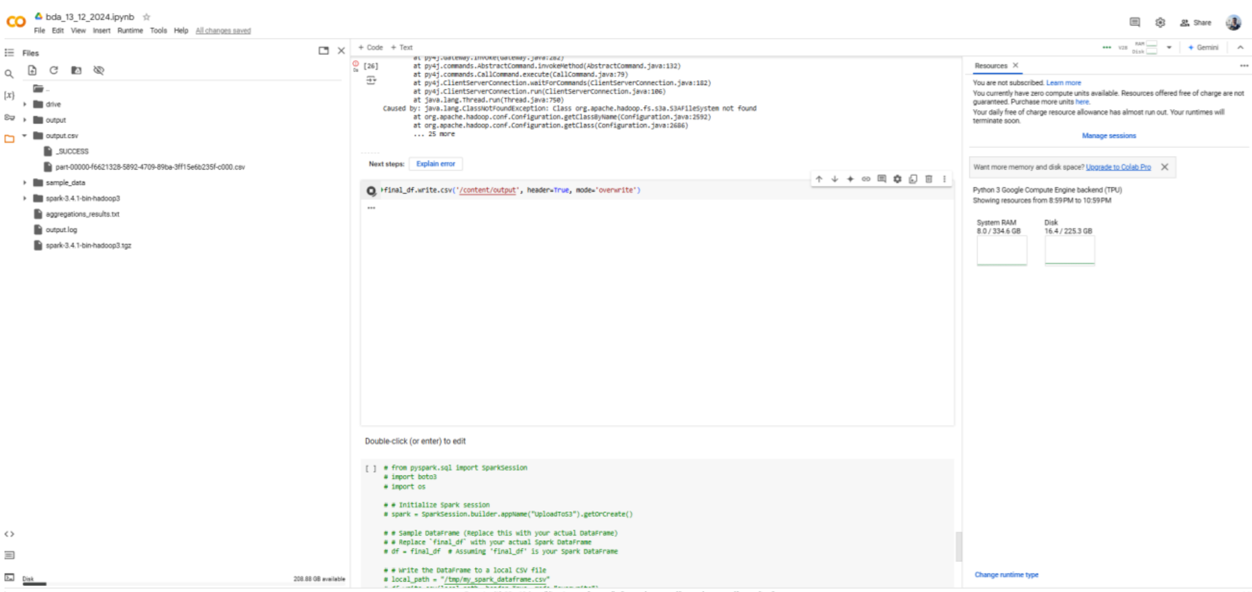


Fig 16 – Dumping the cleaned dataset to a csv file as seen on the right side [part file]

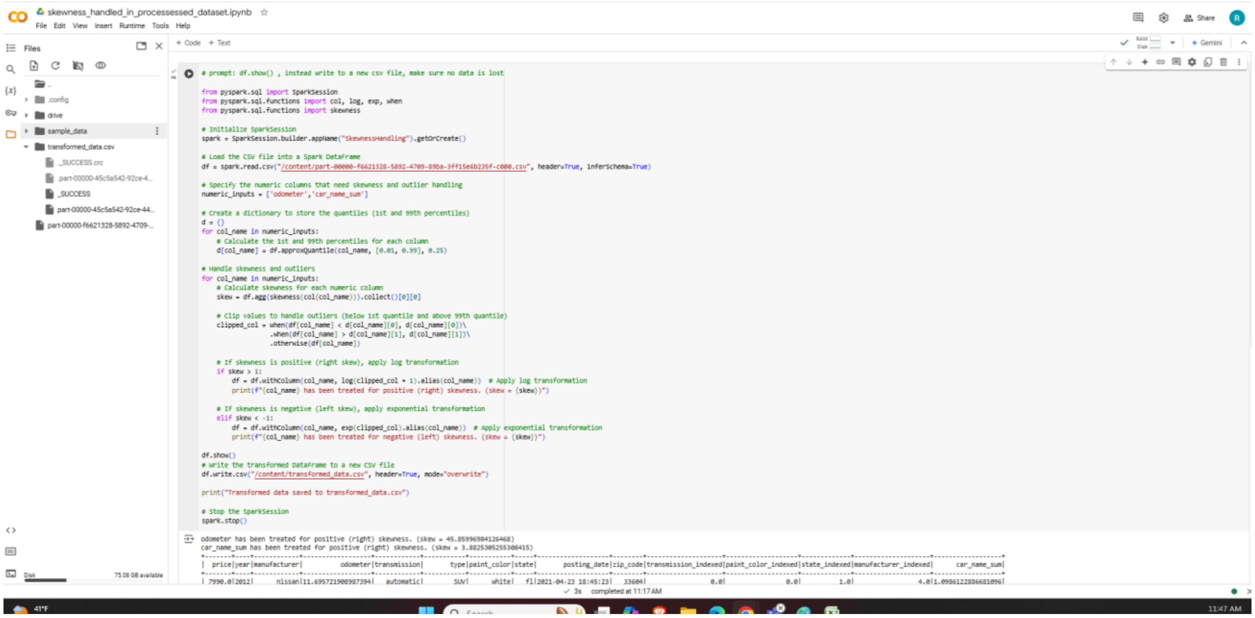


Fig 16 – Reused the preprocessed csv file to further process and remove skewness by applying log and exp transform

Pyspark SQL queries

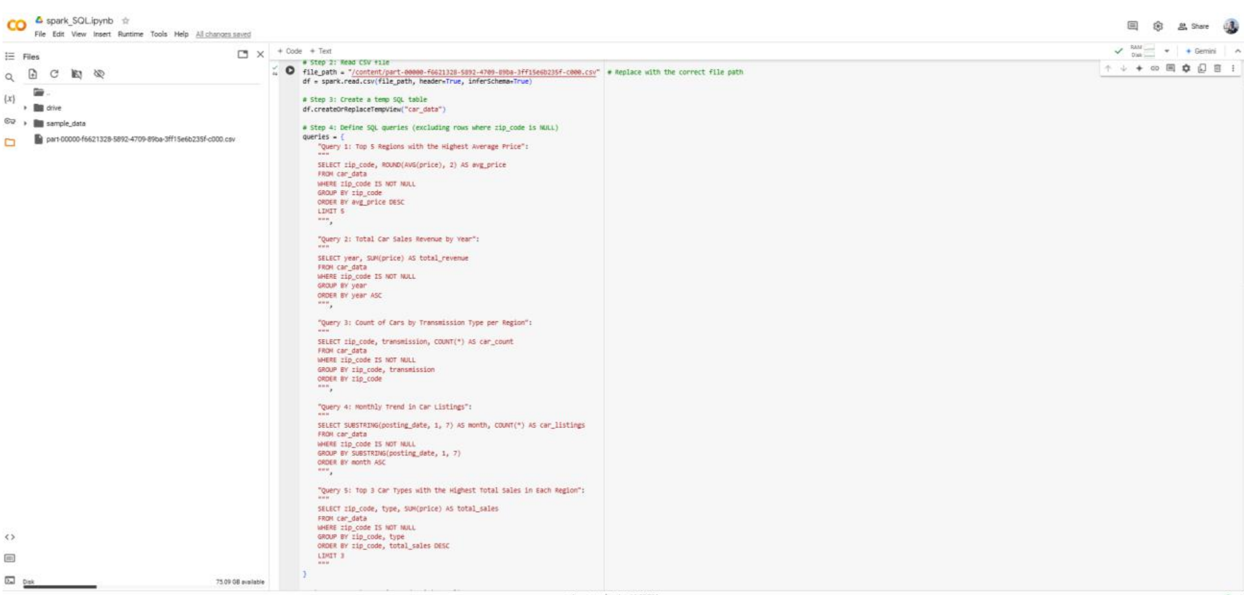
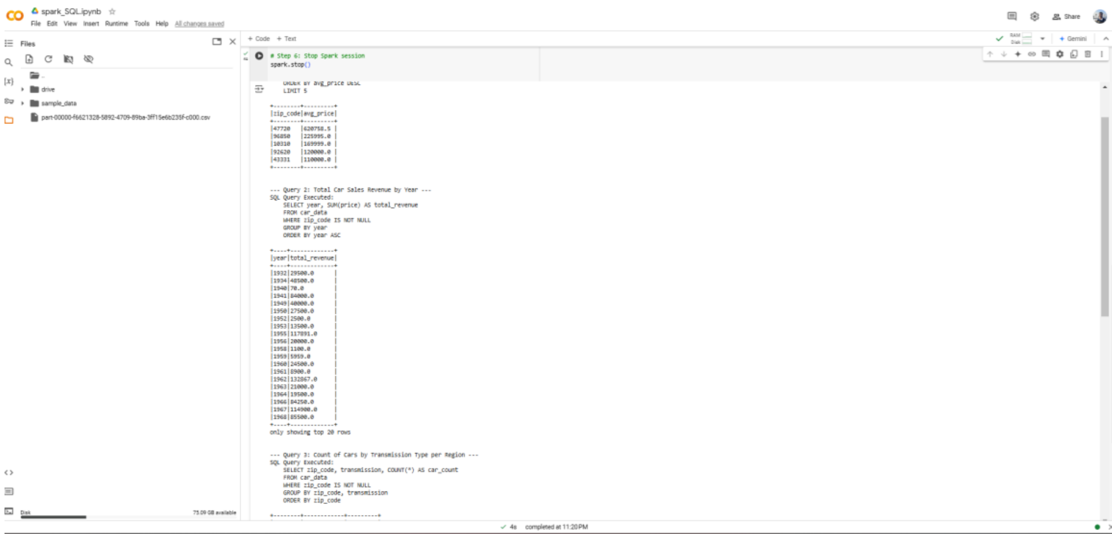


Fig 1 – Ingested the clean dataset csv file into spark dataframe and performed data analysis using spark SQL.



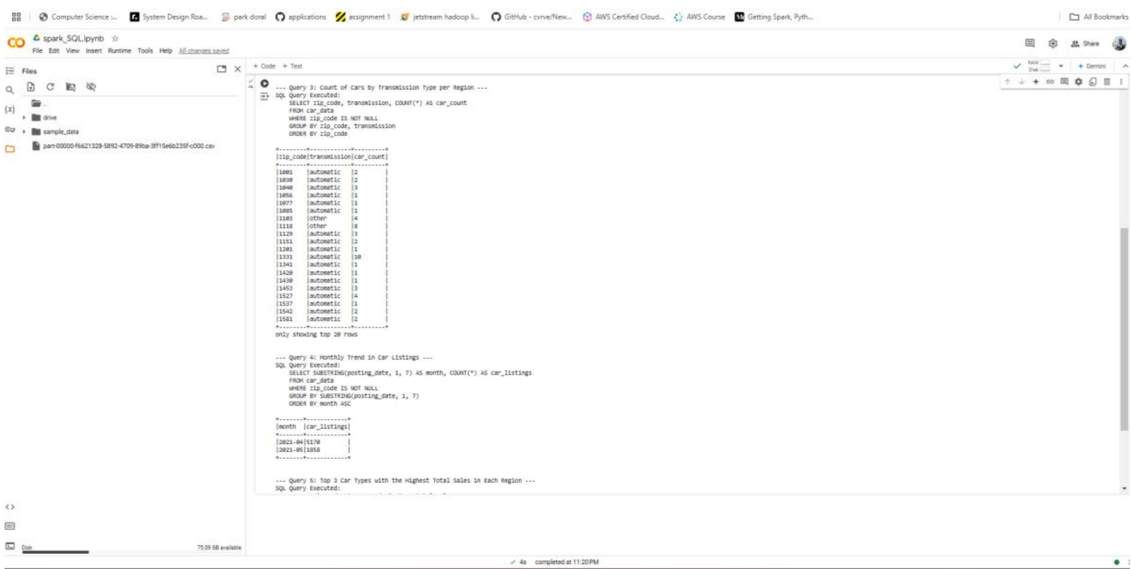


Fig 2 – The outputs of the SQL queries

<u>AWS SageMaker Autopilot</u>

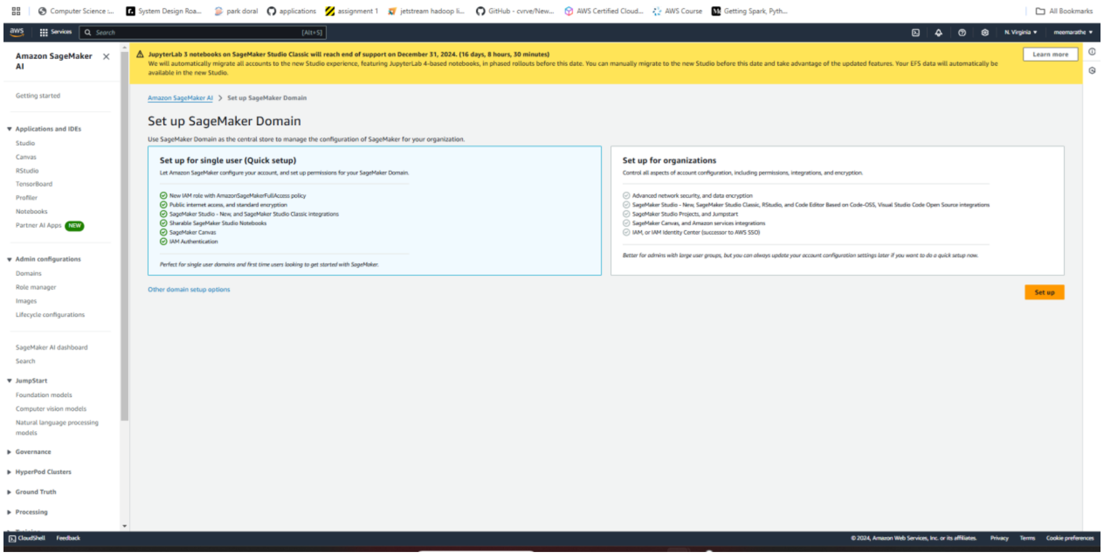


Fig 1 – Domain creation for Sage maker – up and running

We use this domain to launch the Studio where we can run the experiment Autopilot – AutoML for modeling and evaluation.

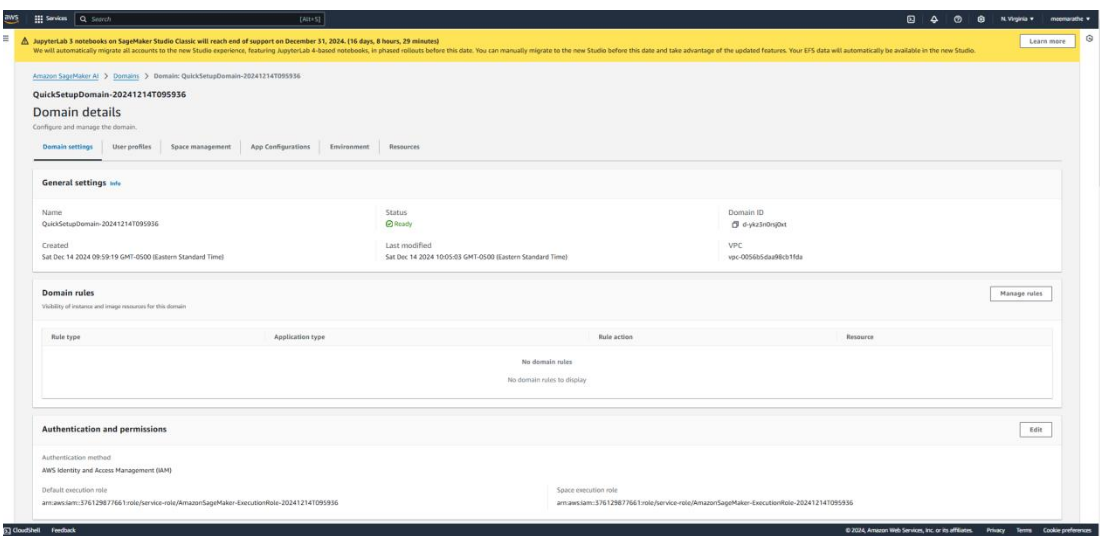


Fig 2 – Sagemaker domain setup success.

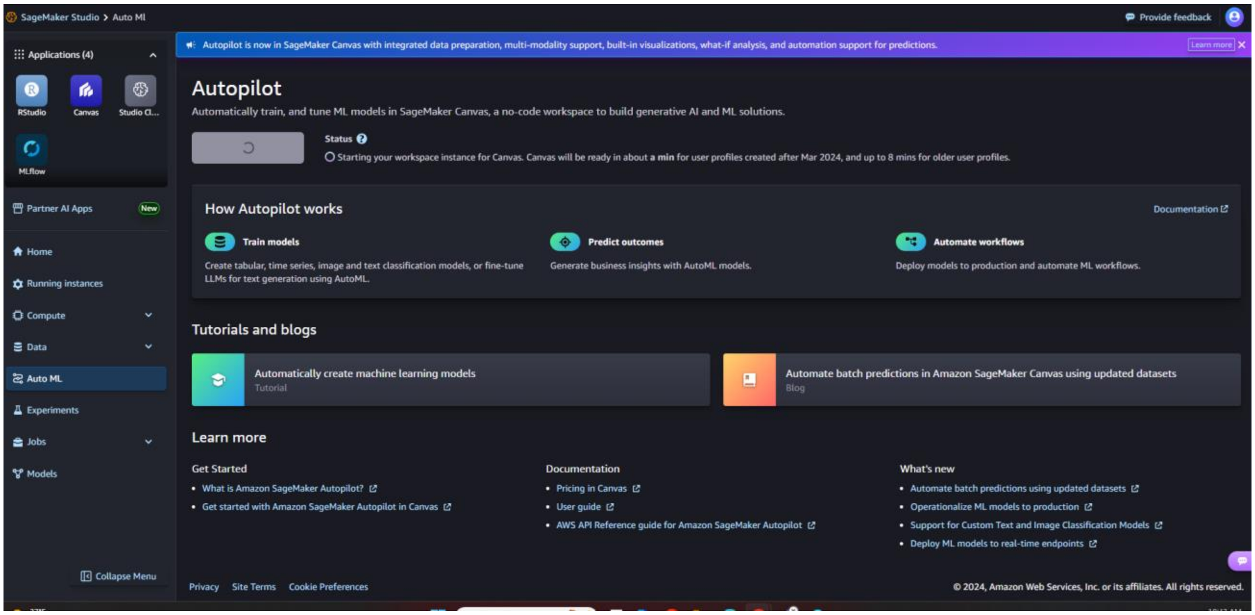


Fig 3 – AWS Sage maker Autopilot running to launch the canvas.

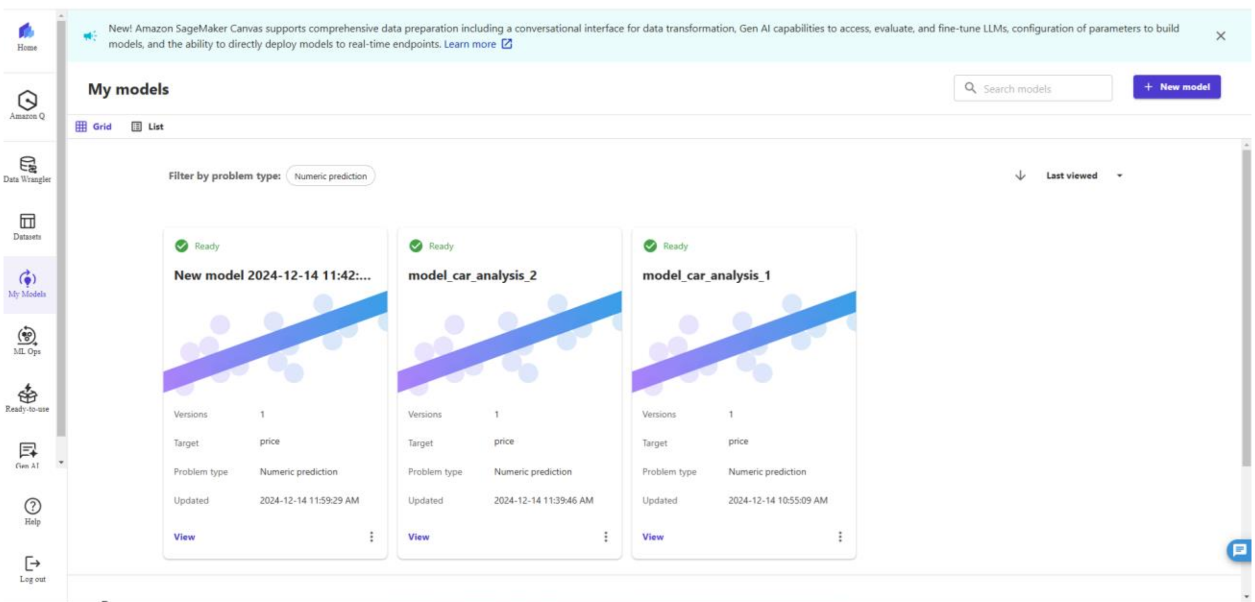
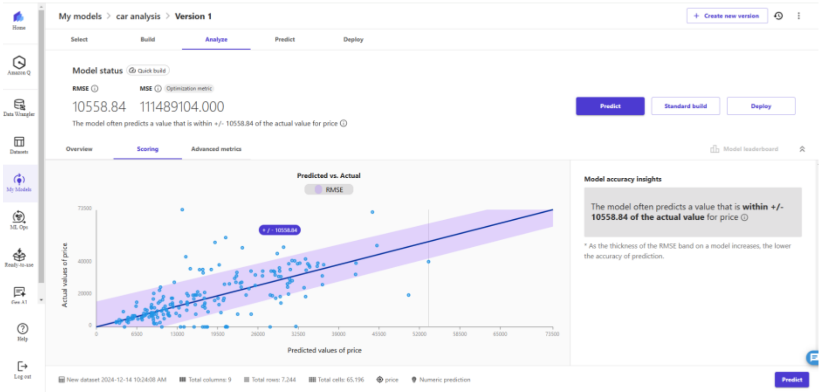
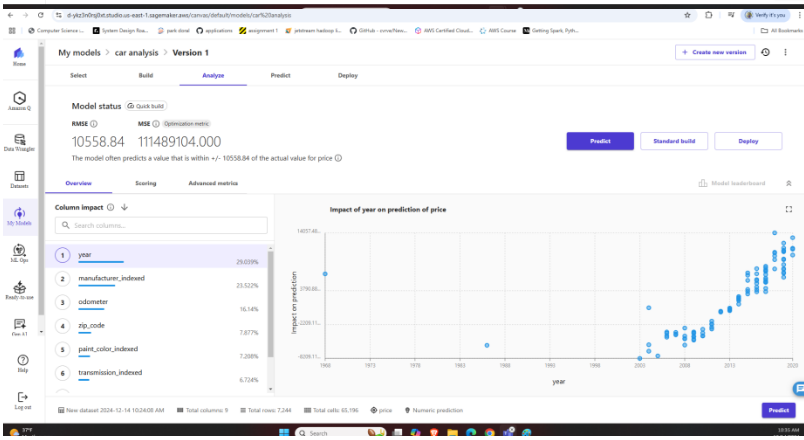
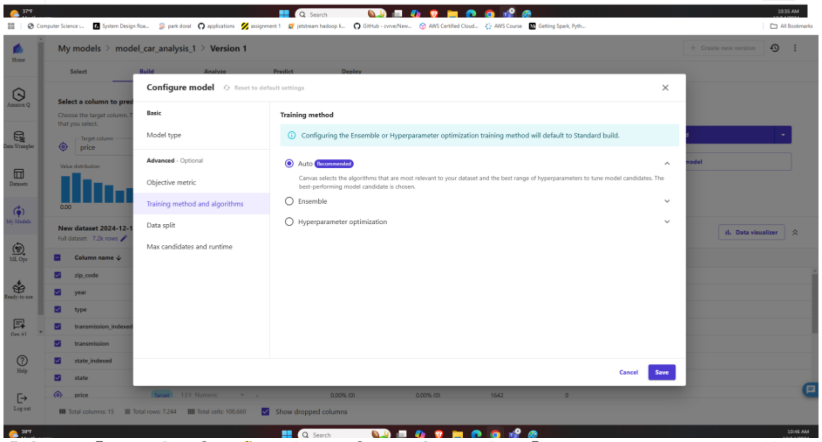
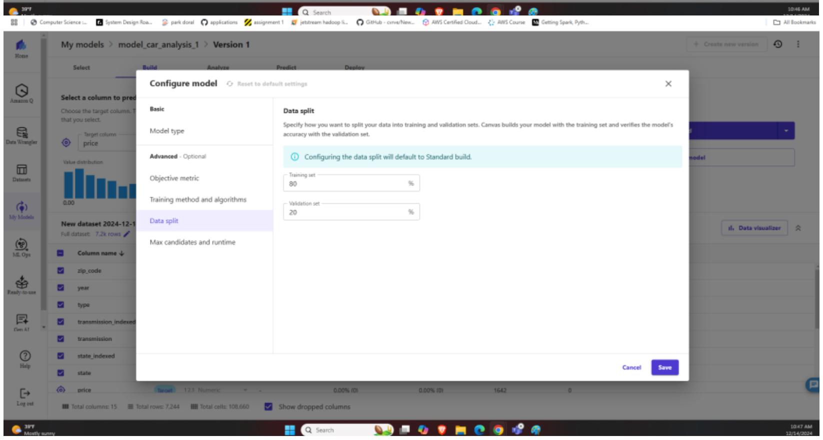


Fig 4 – Created and experimented 3 models [1. With only numeric cols, 2. Raw set with selecting transform operation in AutoML 3. Tree based models on cleaned Dataset]


My models > car analysis > Version 1
Create new version
Select
Build
Analyze
Predict
Deploy
Model status
Quick build
RMSE
MSE
Optimization metric
10558.84
111489104.000
The model often predicts a value that is within +/- 10558.84 of the actual value for price
Predict
Standard build
Deploy
Overview
Scoring
Advanced metrics
Model leaderboard
Predicted vs. Actual
RMSE
+/- 10558.84
Actual values of price
Predicted values of price
Model accuracy insights
The model often predicts a value that is within +/- 10558.84 of the actual value for price
* As the thickness of the RMSE band on a model increases, the lower the accuracy of prediction.
New dataset 2024-12-14 10:24:08 AM
Total columns: 9
Total rows: 7,244
Total cells: 65,196
price
Numeric prediction
Predict


My models > car analysis > Version 1
Create new version
Select
Build
Analyze
Predict
Deploy
Model status
Quick build
RMSE
MSE
Optimization metric
10558.84
111489104.000
The model often predicts a value that is within +/- 10558.84 of the actual value for price
Predict
Standard build
Deploy
Overview
Scoring
Advanced metrics
Model leaderboard
Column impact
Search columns...
1 year 29.039%
2 manufacturer_indexed 23.322%
3 odometer 16.14%
4 zip_code 7.877%
5 paint_color_indexed 7.208%
6 transmission_indexed 6.724%
Impact of year on prediction of price
Impact on prediction
year
New dataset 2024-12-14 10:24:08 AM
Total columns: 9
Total rows: 7,244
Total cells: 65,196
price
Numeric prediction
Predict


My models > model_car_analysis_1 > Version 1
Configure model
Reset to default settings
Basic
Model type
Advanced - Optional
Objective metric
Training method and algorithms
Data split
Max candidates and runtime
Training method
Configuring the Ensemble or Hyperparameter optimization training method will default to Standard build.
Auto
Recommended
Canvas selects the algorithms that are most relevant to your dataset and the best range of hyperparameters to tune model candidates. The best-performing model candidate is chosen.
Ensemble
Hyperparameter optimization
Cancel
Save


My models > model_car_analysis_1 > Version 1
Configure model
Reset to default settings
Basic
Model type
Advanced - Optional
Objective metric
Training method and algorithms
Data split
Max candidates and runtime
Data split
Specify how you want to split your data into training and validation sets. Canvas builds your model with the training set and verifies the model's accuracy with the validation set.
Configuring the data split will default to Standard build.
Training set
80
%
Validation set
20
%
Cancel
Save

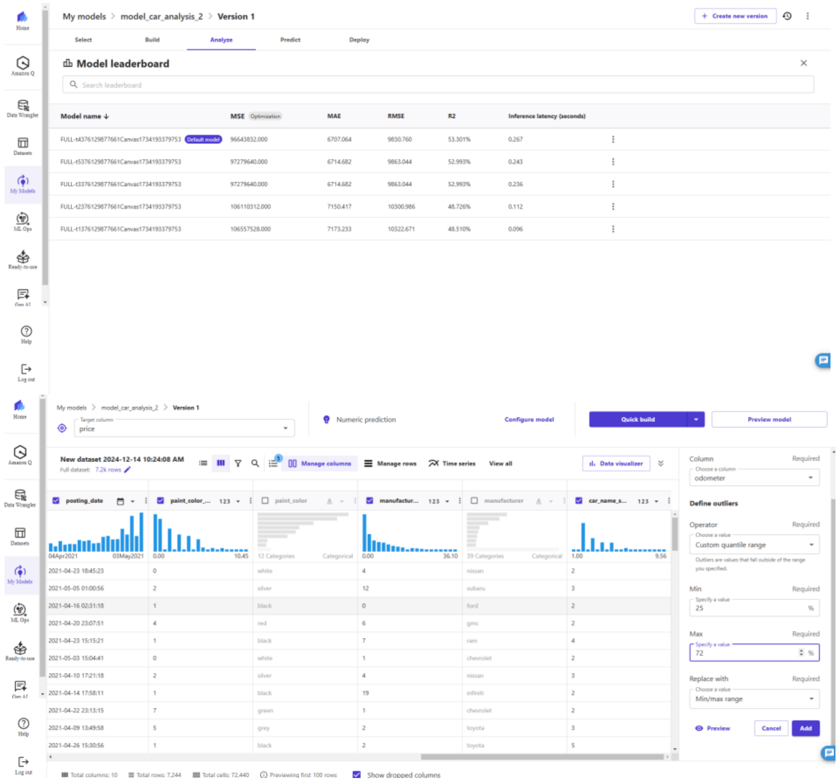


Fig 5 : The figures correspond to config of ML model, running the AutoML tool, Leaderboard for multiple models trained, output metrics of the model for evaluation and prediction of new data – unlabeled. [Regression task to predict price of used cars]

Model insights :

The machine learning model for used car price prediction demonstrates the following performance metrics:

- **RMSE (Root Mean Square Error)**: 10558.84
- **MSE (Mean Square Error)**: 111489104.000

The scatter plot of predicted vs. actual values reveals several key findings:

- The model predicts values within +/- $10,558.84 of the actual car prices
- There is a positive linear correlation between predicted and actual values

- The prediction accuracy appears to decrease as prices increase, shown by the widening purple band
- The model shows better performance in the lower price range (0-30,000) with tighter clustering around the diagonal line

The width of the RMSE band indicates moderate model performance.

<u>AWS QuickSight</u>

Fig 1- creation of the dashboard and data plots.

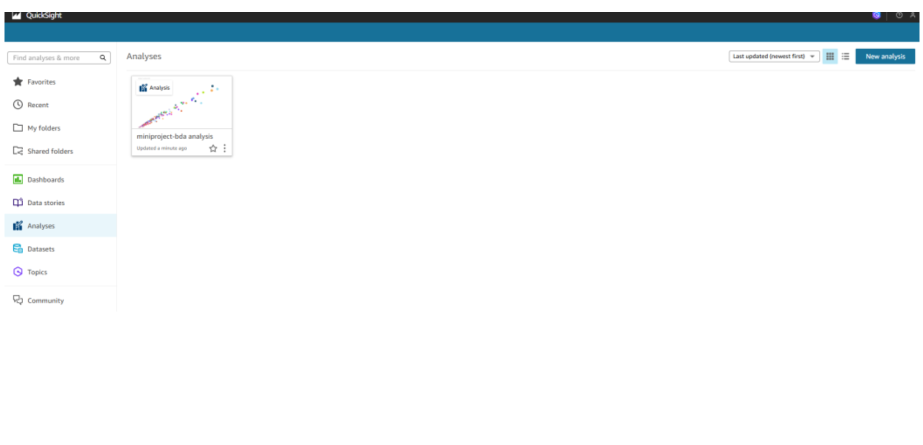


Fig 2 – Landing page of AWS Quicksight, Dashboard for analysis is displayed.

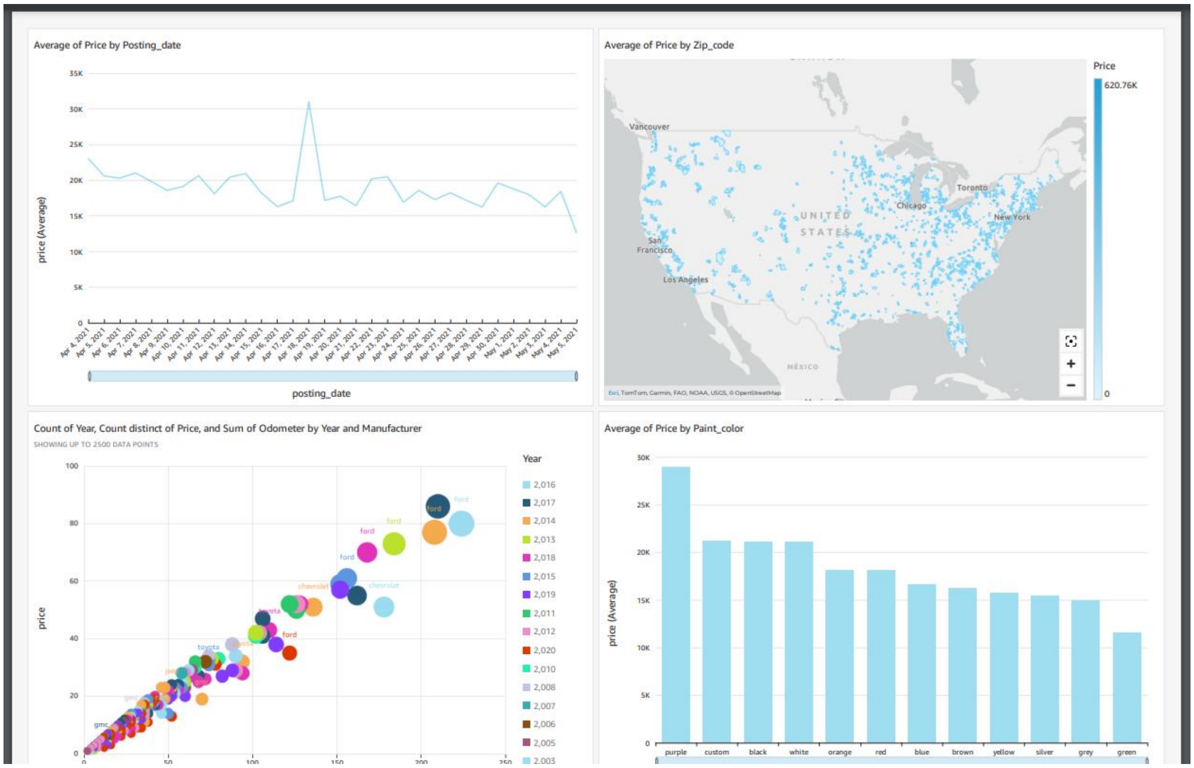


Fig 3 – The dashboard displaying trends with respect to the used car prices.

1. **Price Trend Over Time**: The line graph shows average prices fluctuating between $15,000-$20,000 throughout the posting period, with a notable spike reaching approximately $30,000 at one point..

2. **Geographic Distribution**: The map visualization reveals a higher concentration of car listings in the eastern United States, particularly along the coast. The pricing intensity (shown in blue) varies across regions, with some areas showing higher average prices than others.

3. **Manufacturer Analysis**: The bubble chart shows a positive correlation between year and price across different manufacturers. More recent years (2015-2017) tend to have larger bubbles, indicating higher prices. Ford appears to be one of the dominant manufacturers.

4. **Paint Color Impact**: The bar chart indicates that purple cars command the highest average price (around $28,000), followed by custom colors. Green vehicles show the lowest average price (approximately $12,000). Most common colors like black, white, and silver fall in the middle range between $15,000-$20,000

Summary of Issues and Resolutions for S3 setup and reading data

1. **S3 Bucket Access (400 Error)**
    - **Issue**: Incorrect AWS credentials caused a Bad Request error.
    - **Resolution**: Updated and verified AWS credentials, ensuring compatibility with PySpark.
2. **Bucket Permissions**
    - **Issue**: Public access was blocked, and the EC2 instance lacked sufficient permissions.
    - **Resolution**: Updated the S3 bucket policy and IAM role, granting the required permissions.
3. **PySpark Configuration on EC2 (t2.micro)**
    - **Issue**: Spark configurations exceeded the resource limitations of the t2.micro instance.
    - **Resolution**: Adjusted Spark memory and partition settings to fit the instance's capacity.
4. **Large Dataset Processing**
    - **Issue**: EC2's limited resources caused hangs during Spark transformations on a large dataset.
    - **Resolution**: Migrated preprocessing to Google Colab, which handled the dataset effectively.
5. **Downloading Preprocessed Data**
    - **Issue**: Version conflicts with Hadoop, Python, and PySpark hindered CSV file downloads.
    - **Resolution**: Installed compatible versions of dependencies and successfully saved data to S3.

6. **Accessing S3 Dataset**: Encountered issues with AWS credentials and permissions while accessing the S3 bucket. Resolved by ensuring IAM roles had s3:GetObject permission and setting credentials correctly in Spark configuration.

7. Configuring Spark Session: Faced errors with Spark session initialization due to syntax mistakes and missing dependencies. Corrected the code structure and added required JAR packages for S3 access.

8. Feature Engineering with Geo-Data: Needed to map latitude and longitude to postal codes for analysis. Addressed this by integrating geocoding libraries like Geopy for Python to convert coordinates to ZIP codes.

Conclusion

The project implements a comprehensive data pipeline for analyzing and predicting used car prices using AWS services. The pipeline begins with raw data stored in S3, which is then processed using Jupyter Notebooks for cleaning and feature engineering. The preprocessed data is saved back to S3 and used for SQL queries via Python scripts and machine learning model training through SageMaker AutoML. Finally, AWS QuickSight is utilized to create visualizations and deliver insights to end users.The resulting analysis reveals several key insights about the used car market. The average price of cars fluctuates over time, with a notable spike observed in the time series. Geographic distribution of listings shows a concentration in the eastern United States. The data also indicates a correlation between car age and price across different manufacturers, with newer models generally commanding higher prices. Additionally, the analysis reveals that car color impacts pricing, with purple and custom-colored vehicles fetching higher average prices compared to more common colors. The machine learning model developed for price prediction achieves an RMSE of 10558.84, indicating that predictions are typically within this range of the actual prices.

References:

1. **AWS Services Documentation**

- Amazon S3: https://docs.aws.amazon.com/s3/
- Amazon SageMaker AutoML:
  - https://docs.aws.amazon.com/sagemaker/latest/dg/autopilot-automate-model-development.html
  - https://www.youtube.com/watch?v=Atak2tU1iHY
- AWS QuickSight:
  - https://docs.aws.amazon.com/quicksight/
  - https://www.youtube.com/watch?v=rxyLC247h6E
- Amazon Athena: https://docs.aws.amazon.com/athena/
- Jupyter Notebooks on EC2: https://docs.aws.amazon.com/dlami/latest/devguide/setup-jupyter.html

2. GitHub Repository

- AWS Sample Notebooks: https://github.com/aws/amazon-sagemaker-examples

3. AWS Case Studies

- AWS Machine Learning Case Studies: https://aws.amazon.com/machine-learning/customers/