COMP 264 - CLOUD MACHINE LEARNING

LAB 02

DINH HOANG VIET PHUONG – 301123263

**REPORT**

Introduction

In this lab assignment, I engaged with Amazon Web Services (AWS) to build, train, and deploy a machine learning model. My objective was to use Amazon SageMaker to predict income levels based on the adult census dataset, focusing on the classification of individuals earning over $50,000 annually.

Efficiency and cost-effectiveness were paramount. I navigated through SageMaker's capabilities while carefully managing resources to keep the costs within the class average. I also delved into the XGBoost algorithm, utilized within SageMaker, and explored the SHAP method for interpreting model predictions.

Accompanying this report is a video where I explain each step of the process, demonstrating the setup, training, and evaluation phases within SageMaker, and the cleanup process that followed. This comprehensive approach showcases not only the technical proficiency in handling AWS services but also an awareness of cost management strategies essential in cloud computing.

4. Create an Amazon SageMaker Notebook Instance

A screenshot of a computer

Description automatically generated

6.

A screenshot of a computer code

Description automatically generated

7.

A screenshot of a computer

Description automatically generated

10.

A screen shot of a computer code

Description automatically generated

15.

A screenshot of a computer

Description automatically generated

16.

A screenshot of a computer

Description automatically generated

17.

CloudWatch is used here for monitoring the logs that these training jobs generate.

A screenshot of a web page

Description automatically generated

18.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

20.

A screenshot of a computer

Description automatically generated

22.

XGBoost is a decision-tree-based ensemble machine learning algorithm that uses a gradient boosting framework. In machine learning, boosting is a sequential technique that works on the principle of an ensemble. It combines a set of weak learners and delivers improved prediction accuracy.

Here's an overview of the steps that XGBoost takes:

* Model Initialization: XGBoost starts with an initial estimate which could be the mean of the labels for regression tasks or the log odds for a classification task.
* Additive Training (Boosting): Trees are added one at a time, and each tree is trained to correct the errors made by the previous ones.
* Gradient Computation: The gradient of the loss function is computed to understand the direction in which the next tree should be built to minimize the loss function.
* Tree Building: A decision tree is added that predicts the gradient. Instead of predicting the actual target value, it tries to predict the negative gradient of the loss function, which is analogous to the direction of steepest descent in numerical optimization.
* Pruning: Trees are pruned using the depth-first approach. Unlike GBM where tree pruning stops once a negative loss is encountered, XGBoost grows the full tree and then prunes it back.
* Regularization: It includes both L1 (Lasso regression) and L2 (Ridge regression) regularization to prevent overfitting.
* Cross-Validation: XGBoost allows for cross-validation at each iteration of the boosting process, making it easy to get the most optimal model.

AWS Implementation:

* When you use XGBoost on AWS SageMaker, the service manages much of the complexity behind the scenes:
* Algorithm Implementation: SageMaker implements a highly optimized version of XGBoost that can run in a distributed fashion across multiple compute instances.
* Automatic Resource Management: SageMaker handles resource management and infrastructure provisioning, scaling the underlying hardware to meet the demands of the training job.
* Data Input: SageMaker's XGBoost can pull training data from Amazon S3, and it supports CSV and LibSVM formats. SageMaker also handles shuffling and splitting the data into training and validation sets if required.
* Hyperparameter Tuning: SageMaker provides hyperparameter optimization (HPO) to automatically find the best version of a model by running many training jobs with different hyperparameter combinations and finding the combination that results in the best model performance.
* Distributed Training: The AWS implementation is designed for distributed training, which allows for training on larger datasets more quickly by utilizing multiple compute instances.
* Built-in Metrics: SageMaker provides a variety of built-in metrics for monitoring model training, such as area under the curve (AUC), accuracy, and log-loss.
* Logging and Monitoring: Integration with Amazon CloudWatch allows for real-time logging and monitoring of training jobs, giving visibility into the performance and any potential issues.
* Deployment: Once trained, models can be easily deployed to SageMaker's managed hosting services for making real-time predictions with low latencies, or for batch predictions against larger datasets stored in S3.
* Security: All data transferred to and from SageMaker is encrypted to ensure security, and SageMaker also supports encryption at rest.
* Ecosystem Integration: The trained models can be easily integrated with other AWS services for data processing workflows, such as AWS Lambda, Amazon API Gateway, and AWS Step Functions.

23.

SHAP (SHapley Additive exPlanations) is a game theory approach to explain the output of any machine learning model. It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions. Shapley values, a concept from cooperative game theory, were introduced by Lloyd Shapley in 1953. They are used to fairly distribute the payout (or gain or cost) among the contributors (or players) in a coalition game, where the contributions of each player may vary. The key idea is that each player should receive a payout corresponding to their marginal contribution to the total payout. In the context of machine learning, each feature of the input data is considered a "player" in a game where the prediction is the "payout". Shapley values are used to attribute the contribution of each feature to the difference between the actual model output and some baseline output.

SHAP in Machine Learning:

SHAP values interpret the impact of having a certain value for a given feature in comparison to the prediction we'd make if that feature took some baseline value. Here’s how it is generally used in practice:

* Model Agnostic: SHAP can be applied to any machine learning model, including but not limited to decision tree models, ensemble models, deep learning, and linear models.
* Local Explanations: SHAP explains the prediction of an instance by computing the contribution of each feature to the prediction. This allows for detailed, instance-specific explanations which are particularly useful in understanding complex models.
* Global Interpretability: While SHAP provides local explanations for individual predictions, these can also be aggregated to provide a global view of feature importance across the entire dataset.
* Consistency: If a model change such that it relies more on a certain feature, then the SHAP value attributed to that feature can only increase, which aligns with our intuitive understanding of feature importance.
* Human-Readable Visualizations: SHAP comes with visualization tools that make it easier to interpret the explanations. These include summary plots, dependence plots, and force plots.

Using SHAP:

To use SHAP, we typically follow these steps:

* Choose a Model: Train a machine learning model using your chosen algorithm.
* Select a Prediction: Choose one or more specific predictions to explain or decide to examine the model's behavior more broadly.
* Compute SHAP Values: Use the SHAP library to compute SHAP values for the selected predictions. This involves fitting a model-agnostic explainer object to the trained model, which can compute the SHAP values for any given instance.
* Interpret Results: Analyze the SHAP values to determine the impact of each feature. Positive SHAP values indicate a feature pushing the prediction higher, while negative values mean it's pushing the prediction lower.
* Visualize: Use SHAP's visualization tools to display the results in an understandable way, aiding in the interpretation and communication of the model's decision-making process.

Site note:

I’ve deleted everything on AWS, nothing left, nothing is running as well

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated