Data Analytics Pipeline with Lambda Architecture for Taxi Fare Prediction

Parth Nagori, Anirudha Tambolkar
Department of Computer Science
North Carolina State University
{pnagori, atambol}@ncsu.edu

Abstract

The industry today relies heavily on data analytics to make predictions. These predictions lead to successful business models that incentivise heavily from machine learning. Popular taxi services such as Uber and Lyft provide their users with a prediction of taxi fare before the customer is mapped to a driver. We try to provide a similar solution using the open dataset provided by the NYC Taxi and Limousine Commision (NYC-TLC). The intention is to process voluminous data in streams from NYC-TLC's public data repository and perform parallel feature engineering and deploy a prediction engine on top of it.

Keywords

Serverless Architecture, Data Pipelining, Batch processing, Machine Learning, Amazon Web Services, Data analytics, Scalable Architectures

Motivation

The NYC TLC generated huge amounts of local trip data everyday which can be mined to learn important traits about the trips being taken in the city. For eg. Taxi Fare Prediction, Surge prediction, determine hot spots, i.e. areas high in demand based on number of requests from a certain geographical location.

The main motivation in solving this problem is to be able to design a system which can deal with huge amounts of data generated everyday and still be able to maintain the efficiency in terms of processing it. Machine Learning models in general tend to discard the future repercussions on efficiency when data starts to grow in size. We want to tackle that problem from day one.

Overview

Our system will process the inflow of data in order of Gigabytes from various taxi trips in the most efficient manner possible. Data processing can be further specialized here as reading the data in parallel using a data pipeline, performing various data preprocessing tasks like data cleaning and featuring engineering in parallel, storing the preprocessed data in a Elasticsearch database for faster access and then training a machine learning model on top of it to perform fare prediction. The architecture diagram in Figure 1 provides a high level view of these steps and the tools used.

The solution would be built using various Amazon Web Services. We would start by first creating a Lambda mechanism to read large CSV files [1] in chunks and perform some sort of basic preprocessing. After this, these records would be streamed to EMR cluster where the data would be further processed. We would employ AWS Kinesis to stream the data to EMR. Next, we would create jobs for EMR to perform feature engineering in parallel. This task would be followed by storing the processed dataset on Elasticsearch. Once this step is completed, we can play around with multiple machine learning models using Sagemaker and try to find the best approach for prediction. At the end, we would provide an API to the user for making predictions on demand.

The major tasks here can be broken down into: 1) Efficiently performing read operations on billions of records in parallel stored on S3 in form of a CSV file. 2. Processing the data in order to make it ready for consumption by the machine learning model. This would involve performing data cleaning, feature engineering etc. 3) Storing the processed dataset to a fast retrieval database. 4) Training machine learning model and finding out the best prediction system through trial and error. 5) Creating an API that takes parameters such as start and ending location and time of travel and receive a prediction.

Verification of this data pipeline would depend on the accuracy of the machine learning model as well as the scalability of our data pipeline to large amounts of data. We focus on the latter which would be performed by replicating the data on to larger number of records and testing the breaking point of our pipeline.

Timelines

The 4 major milestones for this project are:

1) Breaking huge csv datasets into chunks for parallel processing. This would involve creating streams of data using kinesis for further feature engineering on a EMR cluster. We would employ the lambda architecture to perform several of these tasks. We target to finish this step by second week of October.

2) Storing the cleaned dataset from EMR on Elasticsearch for faster retrieval by the end of October

3) Training multiple machine learning models on the prepared data and then performing hyperparameter optimization using Sagemaker. We will accomplish this step by second week of November.

4) Provide an API for the user to provide custom input for fare prediction using the trained model. This would set for completion by November end.

Architecture

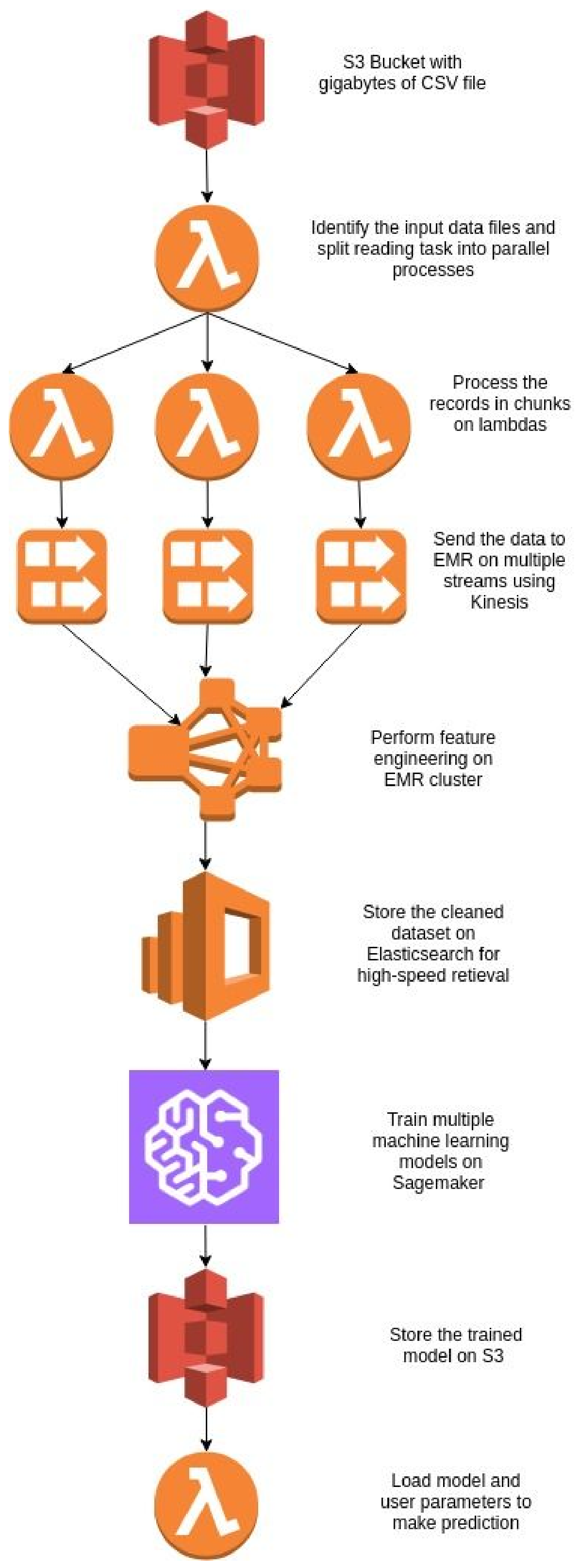
S3 Bucket with gigabytes of CSV file
Identify the input data files and split reading task into parallel processes
Process the records in chunks on lambdas
Send the data to EMR on multiple streams using Kinesis
Perform feature engineering on EMR cluster
Store the cleaned dataset on Elasticsearch for high-speed retieval
Train multiple machine learning models on Sagemaker
Store the trained model on S3
Load model and user parameters to make prediction

Figure 1: Architecture diagram of the data pipeline

Background

Data pipelines are not a new concept and they have been in place since the early 2010s. We have studied some cases where Amazon provides an architecture for tackling such problems using their popular services. [2] explains in detail how a model data pipeline should look like.

Resources

The resources needed to accomplish this project are:

1) Public dataset hosted ny NYC-TLC on S3, 2) AWS Kinesis, 3) AWS EMR, 4) AWS Elasticsearch, 5) AWS Sagemaker, 6) AWS Lambda, 7) Pandas, 8) Scikit-learn, 9) AWS S3 and 10) Python

References

[1] Public NYC TLC data repository http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

[2] Building data pipeline using Amazon Web Services https://aws.amazon.com/blogs/big-data/build-a-real-time-stream-processing-pipeline-with-apache-flink-on-aws/