SUMMER 2019 CPT REPORT

LSTM BASED FORECASTING ENGINE

UNI: AV2845

NAME: AISHWARYA VEDANTARAMANUJAM SRINIVASAN

COMPANY: QUANTIPHI INC

LOCATION: MARLBOROUGH, MA

SUPERVISOR: VANSHAJ HANDOO

Vanshaj Handoo

INTRODUCTION:

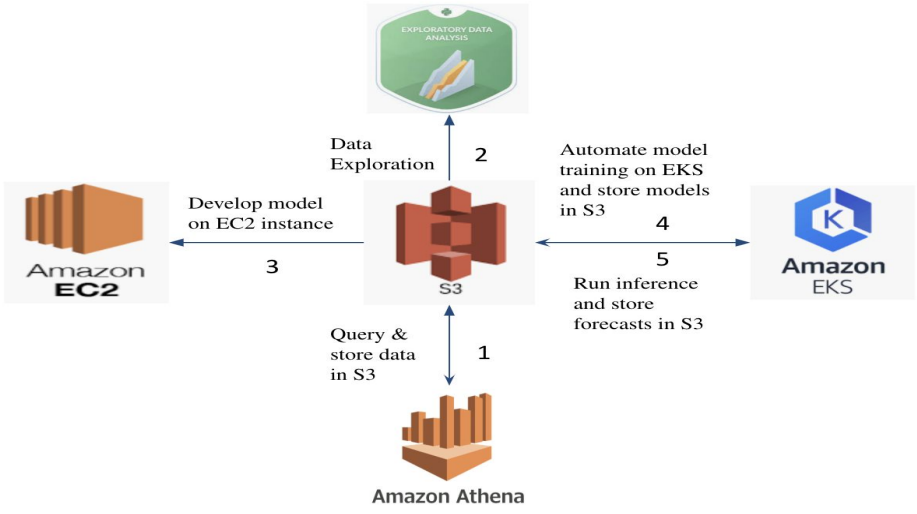
My project with Quantiphi for summer was to develop an end-to-end solution to forecast ingredient demands for a popular restaurant chain in America. One of the major challenges of any restaurant would be to determine beforehand, what quantity of each food item on the menu would be needed on a particular day. Over the summer, I designed and implemented the forecasting engine for the restaurant chain on Amazon Web Services (AWS).

PROJECT OBJECTIVE:

Quantiphi Inc. already has a solution in place to forecast item level demands for all food items in the menu and for multiple store locations. Having known the recipe for each food item in the menu, ingredient level forecasts can be derived mathematically from item level forecasts. However, the downside of this is that the errors in item level forecast negatively affect ingredient forecast too. To overcome this, a completely new LSTM (Long Short Term Memory) based model is proposed and results are evaluated.

PROCESS FLOW:

The figure below represents the overall flow of the solution.



- Historical item level data (available for 5 years - 01/01/2013 to 10/27/2018) along with recipe data is queried from S3 using Athena and the results are stored back in S3. This data serves as our train input for ingredient forecast.
- The resulting table consists of fields - ingredient id, store id, date and ingredient quantity.
- The train data is read from S3 and model building happens in a GPU powered EC2 instance. Data pre-processing and hyperparameter tuning are done in the EC2 instance and final best performing model is obtained.
- After finalizing the LSTM model, a single click, scalable model training solution to be deployed in production is designed on Amazon EKS.
- The resulting model objects are stored in S3 and are to be used to draw inferences - forecast the future.

LSTM MODEL DESIGN:

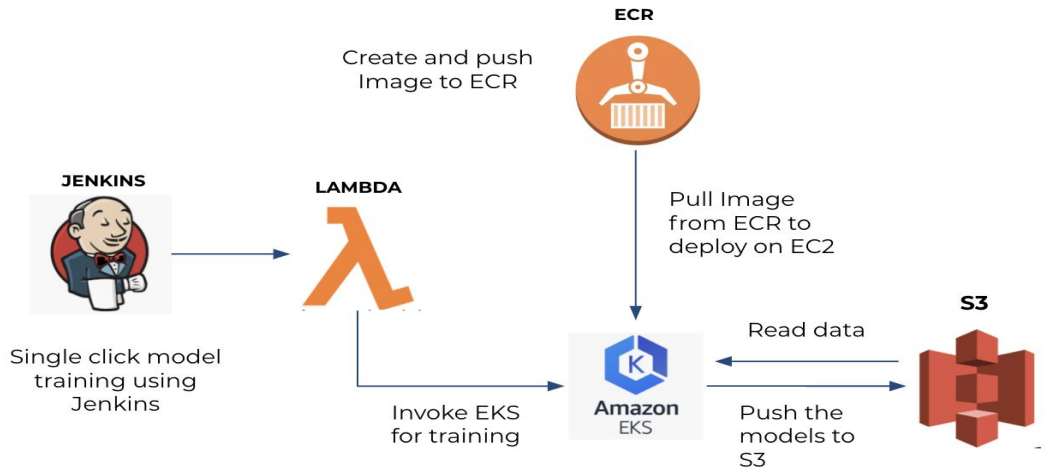
LSTM is a type of RNN model that can learn long term dependencies. It can connect past information with current context. LSTMs are generally well suited for sentence completion tasks. They also perform well on time series forecasting problems. The following steps were carried out in designing the model.

- Convert the historical time series into multiple samples by a certain window size. The window size is the addition of - lookback and horizon.
- Lookback - the number of time steps before the step “t” (eg., a lookback of 5 is from t-5 to t). Horizon - the number of time steps ahead of “t”. (eg., a horizon of 5 is from t to t+5).
- The horizon is fixed to the number of days ahead the client wants the forecast.
- Parameters to be tuned are - lookback, robust scaler quartile range, number of hidden layers, number of neurons and feature engineering.
- MAPE - Mean Absolute Percentage Error is chosen as a measure to assess the model.

- The model was tuned one by one for each of the above parameters over a broad range. At every stage, the value that gave the lowest MAPE was selected. Eventually, the best model obtained was with
    - Lookback - 15
    - Layers - 1
    - Neurons - 200
    - Robust scaler quartile range - (15, 85)
    - Features - Black_Friday_Check, around_christmas, around_thanksgiving, federal_holiday and one_day_prior_christmas_and_newyear
- The model was tested for different time periods such as - Nov to Jan and Mar to Jun.
- It performed well in varied test periods with **as low a MAPE as 11.05%,** as compared to the baseline ARIMA model whose MAPE is 16.3%.

AMAZON EKS DEPLOYMENT:

When it comes to production, it is vital to not have hassles and should be able to quickly move models to deployment. Amazon EKS enables customers easily deploy models as docker images on containers. EKS ensures high availability of containers and manages the infrastructure.



- The above figure illustrates model deployment flow in a detailed manner.

- Model training code is packaged and pushed into Amazon Elastic Container Registry (ECR) from where the Elastic Kubernetes Service (EKS) pulls images for deployment.
- The proposed design is to have one container per pod, and one pod per store.
- The above design enables us run models for the stores that are desired.
- However the GPU consumption in this design is **37%.**
- The resulting models are stored in S3 and are pulled by EKS to run inference for future days.

QUALITATIVE COMPARISON BETWEEN SAGEMAKER AND EKS:

For deployment, both Amazon Sagemaker and EKS were considered and evaluated. Sagemaker provides a fully managed infrastructure for model training and also has the provision to auto scale instances based on workload. On the other hand, EKS works as a cluster and ensures high availability of nodes. It provides failover mechanism for pods and also implements autoscaling of instances. EKS also allows users to set memory and compute limits to containers. Also, spot EC2 instances are supported by EKS unlike Sagemaker. With respect to cost and reliability, EKS is a better choice.

LIMITATIONS AND FUTURE WORK:

The forecast is done per store, per ingredient which accounts to numerous models. The same model design may not be applicable to all store locations and all ingredients. Ideal scenario is to have different model architectures for different type of stores and ingredients. A robust technique that can classify a new store into a specific category based on a variety of characteristics should be developed. In this current solution, there are chances that this model may not perform very well on certain stores and ingredients. With respect to resource utilization, there is room for improvement wherein ways to improve available GPU shall be explored. Currently, there is no support to allocate a fraction of GPU to one training job.

CONCLUSION:

This internship is my first experience handling a real world data science problem. It gave me a perspective of how to strike a trade off between model robustness and business needs. The most important thing I learned was to treat deployment as important as model building. Unless we are able to move the models in production, we are not there !