Max Kimball

MSDS434

Final Documentation

Crypto Sentiment Analysis Project

For my final project, I decided to create a crypto sentiment analysis web app which would predict sentiment based on Telegram messages and then forecast the average sentiment over 7 days. The sentiment labels were positive, negative, neutral or mixed. I used AWS tools such as Lambda/Comprehend, S3, RedShift, SageMaker, and EC2. I used S3 to store raw JSON data from a Telegram dataset from Kaggle. I then used Lambda to clean and preprocess the data. Since the dataset was very large, I had to split the dataset into 125 files in order to process it in Lambda. I used AWS Comprehend within Lambda to gather sentiment for each message. I also used S3 to store the training and test data.

After cleaning and processing the data using AWS Comprehend, I used Redshift to analyze the sentiment dataset and created a SQL query to analyze the average sentiment for each day over 3 months. I then built a train sentiment model to analyze sentiment and built a docker container to host the app on an EC2 instance. The main API is defined in app.py which uses /predict to analyze the sentiment and /forecast\_chart/{days} to forecast future sentiment over a few days. After I got the main app working, I created a monitoring service using Prometheus and Grafana for charting. I created a production and test environment utilizing GitHub actions.

One advantage of this project is that I used docker which made deployment a lot easier once I got it working. I hosted the web app on an EC2 instance instead of Elastic Beanstalk and most of my data was stored on S3 which kept costs down. I used Lambda and RedShift serverless, so I didn’t have to maintain my own infrastructure. I was able to use GitHub actions for CI/CD deployment and also created a dev branch which made it easier to test and deploy. The GitHub actions workflow would automatically deploy my docker images on my test and production environments.

For this project, there were some disadvantages and many areas of improvement. Since I had limited time, I decided to just use a Telegram crypto sentiment dataset instead of live API data. This dataset only had 3 months of data from 2021 which made it difficult to train a model and was the biggest limitation. I initially tried getting API data from Twitter, but it was extremely limited and would have to pay monthly for API access. The second option was to scrape data instead, but I ran into a lot of problems When I used SageMaker AutoML and RedShift ML, it incurred a lot of costs just to get it set up. This made it difficult to test models since it was so expensive. I had to train a lot of the data locally first to test.

During the weekly discussions, I noticed a lot of other students ran into the same problems as me. Some examples were not having enough volume or computing power on the EC2 instance to train model and incurring a lot of costs using AutoML. Based on feedback from peers, I learned to add more volume to my EC2 instance and I gained tips on keeping costs down with RedShift, SageMaker, and AutoML and deleting everything after. I also learned the difference versions of SageMaker and how to set up IAM properly in order to get it to work.

If I had more time, I would have looked into other API’s such as Reddit or done web scraping based on news data. I would have done more feature engineering and combined live sentiment data with live price data and used Apache Kafka for streaming. Deep learning models could have also been used such as BERT for forecasting sentiment and I would have done A/B testing on the models.