Gofickle : A University Discovery Platform

1. Introduction

While applying for universities, a lot of students face difficulty in shortlisting and searching for those that correctly fit to their needs; it is a hassle given the myriad of information. To address this issue, we have created a web application called Gofickle, which uses AWS services. It takes an aggregation of 2100 university which contains numerous programs that a user can select from and analyze according to their profile to know if they are eligible for that specific program; this is done using machine learning. We are also analyzing user click stream data to understand the user preferences.

2. Implementation

The user can access the system with a simple Login/Signup page. We allow users to login/sign up with Facebook, Google, or our own custom login/signup. When the user signs up we get the username, gre/toefl, gpa, name, and address. With the given information we create a session of the user. We are also maintaining the user data in a tab called My Profile. The user can also choose to alter their information.

Once the user is authenticated, they are directed towards their dashboard called My University. In the dashboard, the user can shortlist programs of different university based on their liking and maintain a status of the program - Accepted or Interested.

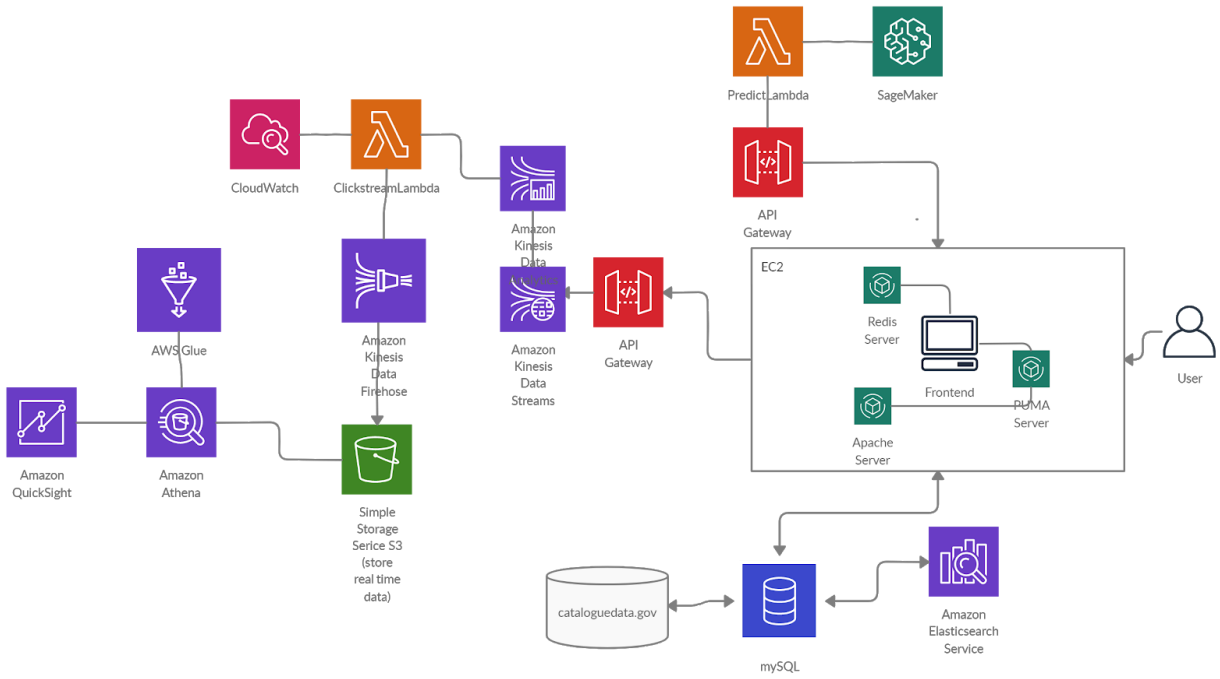
Predict feature : Here the user can predict the reachability of a university based on their gpa and gre. We are using sagemaker to analyze the data received from the user and based on that we are predicting if the university is “Reachable” or “Not Reachable“ given the users profile.

Search Tab: Here we have provided the user with an ability to search for colleges based on the Major. For example, the user can search for “law” in the major search field. The result would be a list of programs that offer different set of law programs at different universities.

FeedBack : Here the user can send feedback to the developers.

DropDown: Here we have provided the user to see the privacy policy of the website. Also, this dropdown option provides the Logout option.

3. Architecture



3a. Front End

Frontend of the project is built with Html, Css, Javascript and Jquery. For the better look and feel of the project we used bootstrap framework. The frontend gives the user interface to interact with the application dynamically. The application is connected to the MySql with the elasticsearch in backend.
The user can search, shortlist and manage university based on his preferences. The searching we used, uses the elasticsearch which provides a distributed, multitenant-capable full-text

search engine with an HTTP web interface and schema-free JSON documents. We have a huge dataset of nearly 2100 universities scrapped from the "cataloguedata.gov" with the help of ruby library "nokogiri" and hit the external apis with "Faraday" library.

The frontend is deployed on EC2 instance and leverage the benefit of scalability as the application grows in size. We used redis server for the background jobs like when user input his details and scan through the search we send data as a background job to the sagemaker with the help of redis. To connect the frontend with the external network we used apache server. We also used AWS api "apigclient" to hit the api when user tries to find out that the selected university is reachable or not reachable.

3b. API Gateway

We are using the API gateway to pass data from the FrontEnd to our Backend, where we are doing some processing and sending the processed data back to our frontend.

- SageMaker Api:

We are taking data from the frontend predict tab: student gpa, gre, toefl, and aspired college(csv format). We are passing this data from frontend to a lambda via an ApiGateway. For this API we are using a POST method. Data is received over the lambda side and is then passed onto sagemaker. The processed data from sagemaker is passed to lambda which is then passing the data to our Frontend via a post method in our API gateway.

- Data Analytics API:

We are passing user clickstream data via this API. So when a user clicks on any university, that triggers our API gateway and we pass that specific program of that university to our lambda function which passes the data into Kinesis firehose for processing.

FrontEnd APIs: We have other APIs as well that we have covered in the frontend section (3b). This includes communicating with mysql server.

3c. Big Data Analysis

We want to visualize the amount of clicks for each university. This is helpful for us to analyze and learn which universities are popular choices for students, and we can use this analytics help give program suggestions for other users.

To set up this visualization, we first obtain data from user's click activities in the frontend. When the user clicks on a particular university/program, information is sent from the frontend, through the API Gateway, and into Kinesis Streams where clickstream data is created, and then through Kinesis Data Analytics, where sessions

are created to separate the clickstream data into different sessions depending on user behavior. This data is then passed to the ClickstreamLambda function, where the data is processed with Amazon Firehose and stored in an S3 bucket.

We use AWS Glue's crawler to crawl out the clicks data in the S3 bucket and send it to Athena in the form of a table. We can then query the tables to create desired formats for our tables; for example, we can do a SQL query to format the data to contain only clicks that were made within a certain time period (e.g. January-June).

With the tables being constructed in Amazon Athena, we can finally use Quicksight to visualize the data in the form of graphs.

3d. AWS Sagemaker:

We trained a Machine Learning model using SageMaker to predict whether a student will get admitted into a university program. Specifically, we get the student's GRE score, TOEFL score, GPA, and the name of the university. This information is used as input attributes to train the ML model. The output of the model is the confidence score with which the student will get admitted into the university.

The range of this confidence score is between zero and one. The higher the confidence score, the more likely the student can get into the program. Based on the confidence score, we output text that says if the student is likely to get into a university.

Training specifications:
60% of data - Training
20% of data - Validation(to tune the hyperparameters)
20% of data - Testing

ML algorithm used: XGBoost, a gradient boosting library which can be used for classification problems (binary classification in our case). It uses an ensemble of weaker prediction models to make a stronger prediction.

Training time:
The XGBoost model took around 1 minute to train on a ml.m4.xlarge instance.

Endpoint:

After training, the model was deployed and hosted as an endpoint. So when a student logs into the gofickle application and enters his/her details like GRE score, TOEFL etc, the data hits the endpoint and the confidence score predicted by the ML model is returned. The main advantage of hosting the trained model as an endpoint is to serve predictions for students in real-time.

Test Accuracy: The model was tested with the test data and it gave an accuracy of 74.3%

4. Future Work

In this project we are storing the users' click data and processing it using Athena. In the future we plan on making our recommendation engine more robust. To do so, we are using the clickstream data as a means of training our university recommendation model (sagemaker). This would tailor the recommendation system (sagemaker) based on the users' requirement and return more accurate results. We also plan to improve the recommendation system(sagemaker) with other classification techniques that could increase the accuracy.
We plan on deploying the application on mobile(android/IOS) so it's easier for people to browse it on their phone, thereby attracting more users to try out the application. We are also planning to add a comment thread and a user-based ranking system; these features can scale the system to provide users with responses for questions about the program and a program specific rating.

5. Conclusion

To summarize, we have made a web application that allows users to quickly browse a list of university masters programs, and also add parameters to reduce the list size. On top of that, we are able to analyze the clicks made by users that browse in our page. Once they have decided on a masters program, they can use our prediction model to see how likely it is for them to get into the program.

This application helps students easily see a list of university programs that match their interests. What is left is for them to do more research to narrow the list even more, before they apply to the programs.