## ADS508 Data Science Design Document Team 4

**Authors:** Dingyi Duan, Stephen Kuc, Luis Perez

**Company Name:** Streamology

**Company Industry:** Video Streaming Service

**Company Size:** Small to Mid-Sized, 100-200 employees

**Datasets link**: <https://github.com/dingyiduan7/ADS-508/tree/main/Datasets>

### **Abstract:**

*A one or two sentence summary describing the problem the company is facing.*

In the past 5 years of rapid growth, Streamology has been experiencing 25% of customer turnover rate due to continuous rise of competitors like Netflix and Hulu, with personalized experiences, original content and other relevant shows and movies. Thus, Streamology has created a mission-critical goal to win back the users by first providing a reliable holistic view of its users for more user-centric experiences.

### **Problem Statement:**

*One or two paragraphs dedicated to explaining the problem at a high level and why it is worth addressing given the company history.*

Streamology does not currently have a holistic and accurate view of its customer base which has led to poor customer experiences and, ultimately, customer turnover, as its competitors are excelling in customizing customer experiences. Many customers have been recommended a show that the actual view time from the customers is much less than the play time which presents an issue that the users are not interested in the contents recommended by the algorithms.

The current algorithms have been getting low confidence scores for predicting users’ demographics and psychographics tags. Streamology has been tasked to tackle this challenge to improve the accuracy of the prediction in order to focus on understanding the psychological and demographic factors in order to provide recommendation systems and better investment strategies in terms of licensing and original programming.

### **Goals:**

*A list of the outcomes that will come as a result of this design. Defining goals helps others understand what success of this project will look like.*

* Understand the underperformed measure of the current model and identify the reason for loss of users.
* Build a model to accurately predict and classify customers by their demographics and psychographics traits, and use that information to implement the aforementioned strategies. Currently, the baseline accuracy is about 43%. We’d like to get the accuracy to be nearly doubled at 85%.
* Cut customer turnover rate by 50% by implementing stronger by creating better recommendation systems, marketing strategies, and adding relevant content

### **Non-Goals:**

*What are you intentionally not doing or solving with this project? Defining non-goals helps limit the scope of your project and prevents feature creep.*

We are not to change the attributes of our current data structures as well as how the data is acquired. Since it’s a small to mid-sized company, the budget for this project is on the low end and we do not want to predict anything else other than the demographics/ psychographics traits.

#### **Data Sources:**

*Describe where your source data will be coming from, and any risks you see given the data type, size, etc. Where will you store the data, how will it get there?*

The source data mainly comes from the backend on the server internally from Stream Co. Since the data is single-sourced, we are facing potential risks of impurity of the data and lack of diversity on the data. We can see that in these data we don’t have enough information on the diversity of customers such as age, race, gender and occupation, etc to form a complete demographics / psychographics mapping. Without having other complementary data from other sources, we are risking having an underfitting final predictive model(s).

The data will be stored in Amazon S3 buckets by manual uploading in this case(Ideally it would be written to AWS Kinesis for instant ingestion or use AWS Glue Crawler for continuous ingestion), and queried and processed in Amazon SageMaker AutoML.

### **Implementation:**

*Details on how the solution will be implemented. This should be the longest section of the document.*

#### **Data Exploration:**

*Detail what you will look for in the data during the exploration phase. Consider data quality, potential data bias, key fields, data types, etc.*

* *Where will you be storing your data, how will you get ingest it there?*

The original datasets (3-5) will be manually uploaded to a pre-made Amazon S3 bucket. Any successive files will be stored in S3 buckets made through SageMaker. Specific Athena tables were created for each dataset, assets","plays","users","demographics","psychographics", to gain insights and data exploration using Athena. After data exploration the five tables will be joined into one df\_psych master table using Sagemaker. The final table will contain only the features that can contribute to classifying the target variable. Although Athena can provide a platform to explore data, our team pivoted to data exploration using Sagemaker and Python as the team felt more comfortable using Python within Sagemaker for data exploration.

* *What tools will you use to ingest and explore the data?*

We will use SageMaker to pull the data from the s3 bucket. Using boto3, the AWS SDK for Python, we will use Python and its libraries like pandas, seaborn, matplotlib, as well as the Python client for Athena with PyAthena, to explore the dataset.

* *Detail what you will look for in the data during the exploration phase. Consider data quality, potential data bias, key fields, data types, etc.*

As we’ve learned, garbage in means garbage out. Thus, data quality is important for our model’s performance. We will first need to understand data type, size, quantity, then check for any anomalies in the datasets such as missing values, nulls, and outliers. Any findings in the data exploration phase will be recorded either in the GIthub repository or in a team document where findings are discussed in weekly debriefs.

We are to combine the individual files into one or two master files by merging different dataframes; there may be many outstanding values from that action. We will be using various visualization tools and libraries to gain insights from the data and help us learn any potential bias. The main primary key in this case would be user\_id which we will use to merge/join the tables together with and we’ll connect the assets.csv with the rest data through assets\_id. Some of our concerns include duplicated user\_ids and features with extreme skewness which we are expecting to discover.

After joining the files, the first step in our data exploration was to look for records that had no genre and remove those records. For the studios that are missing source language, there can be no language, one language or two languages. To avoid confusion, we will remove records with missing studio\_id and source\_language. After analyzing the distribution of season\_id and series\_id we concluded that the two columns are derived from showtype.

Therefore, season\_id and series\_id were removed. Additionally any level\_3 missing data was salvaged by replacing null with data from the level\_2. The data also had outliers, we dealt with those by comparing z-cores as a cutoff. As well as removing columns like ‘iflix’ that had no explanation on what the data in the column entailed.The column platform\_type was generalized into groups (mobile\_phones, web\_based, and home\_tv) to decrease the number of categories the column contained.

Features that were deemed non-contributing to classifying the target variable were removed from the final file. The correlation between independent variables yielded that we needed to remove running\_minutes from the dataframe due to high correlation with showtype. Additionally confidence\_score was removed from the dataset as it was a score provided by the initial data with no explanation of how they achieved it.

* Implement your data ingestion (if done via code) and data exploration using SageMaker studio notebook. Store your code in a github repository and provide a link to the code used for ingestion and exploration.

<https://github.com/dingyiduan7/ADS-508/tree/main/SageMaker_Code>

#### **Data Preparation:**

*How will you transform the raw data so that it is ready for training?*

* *Data Scrubbing: What data cleansing techniques will you apply?*

Using boxplots, histograms and descriptive statistics, we will manually check for missing values, outliers and other anomalies. These will also further inform us of distributions that were seen in the data exploration portion.

For certain nulls, like genres and studio\_id, which are impossible to either impute or interpret, we will delete these records, especially as these make up a very small portion of the overall data.

For outliers in running\_minutes and minutes\_viewed, we will utilize a z-score of 3 as the cutoff to remove the high outliers those features contain.

For low values in minutes\_viewed, which could be anomalous data, we won’t get rid of them, however, we will create another column stating if the play was longer than 2 minutes.

For the level\_2 trait ‘iflix Viewing Behavior’, this does not relate to the rest of the traits as it is a created feature that iflix already segmented their customers in. We will not include this in our modeling, so we will remove it.

Then, utilizing a correlation matrix, we will identify the features that may have high correlation with each other to avoid multicollinearity concerns from the model.

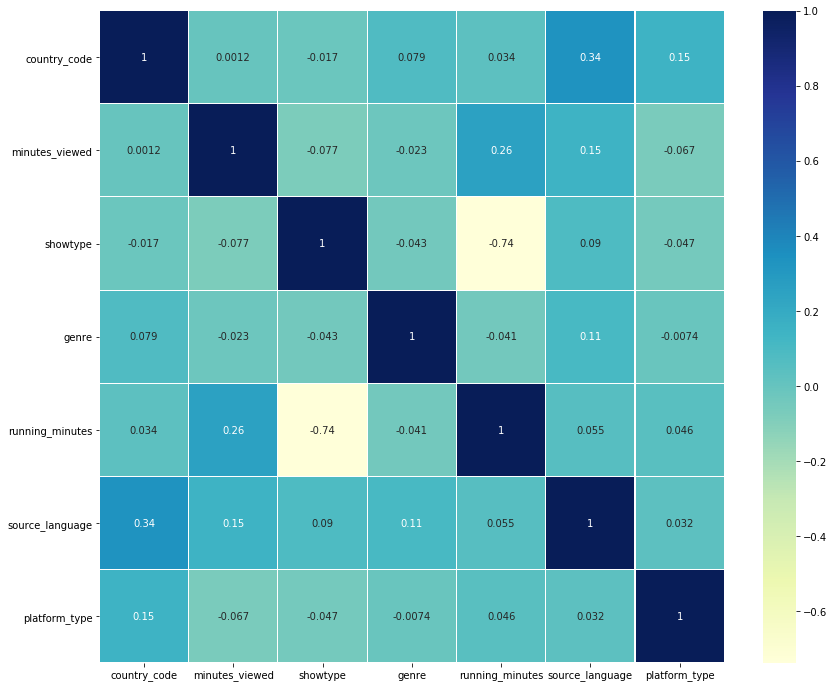
For confidence score, we want to ensure we’re modeling off of the best data available. We will use its 70th percentile as the cutoff score - keeping the top 30% of confidence scores in the data set (confidence\_score > ~.86).

* *Feature Selection: Which fields from your data will you use/exclude?*

We will remove features that don’t contribute to our prediction or simply being meaningless such as: user\_id (used for merging the datasets), level\_1 trait (since we work on ‘Psychographics”, all level\_1 traits are the same), assets\_id (used for merging the datasets), studio\_id (identifier for studio with hundreds of classes, does not contribute to users’ behavior), platform\_x (duplicates from platform\_y), running\_minutes (has high correlation with showtype, directly reflects that running\_minutes > 85 min would be ‘Movies’ and < 85 min would be ‘TV shows’), confidence\_score (used to filter out the bad records and no more need afterwards) and level\_3 traits (since we are focused on level\_2 traits, and level\_3 traits are sub-groups of level\_2.

Figure 1 displays the correlations between all of our current features up to this point in the project. We can see that running\_minutes has a correlation coefficient of -0.74 with showtype which means that these two variables are linearly correlated with each other. Since ‘showtype’ is an important feature that we want to keep, we will remove running\_minutes as described in “Feature Selection”.

**Figure 1.** *Correlation Matrix of Features*



* *Feature Creation: Which fields will be combined, or bucketed?*

Platforms will be bucketed and combined into platform types. Since we have 13 classes: 'android','iOS', 'web-embed', 'webOS', 'web', 'web-pwa', 'android-tv', 'Panasonic', 'Roku', 'Samsung Tizen', 'googlecast', 'Vewd', 'Samsung Orsay', platforms like iOS and Android will be combined into mobile\_devices, ones like web and web-embed will be combined to web-based, and the rest, which are tv-based devices like Samsung Tizen or googlecast, will be combined into home-tv.

Other features we’re creating, some mentioned earlier in this document, are if users viewed an asset for over 2 minutes. We believe that 2 minutes is about the threshold for if a user commits to watching a certain asset or not, and that could be valuable in attaining accuracy in predicting their trait.

* *Feature Transformation:* What other transformations (such as one hot encoding, etc) will you apply to your data?

When we look at our level\_2 and level\_3 traits, we see some level\_3 traits have missing values and invalid string values like \"\". To predict level\_3 traits we will need to hot code those values to something we can comprehend. So we will manually change all invalid level\_3 traits to their level\_2 names.

Our original dataset has over 300,000 records and has a range of confidence score of 0.1 to 1. To ensure our data quality, we will manually crop out our data and only keep the records with confidence\_score 0.86 and above (70th percentile).

For minutes\_viewed, while we will get rid of high outliers (perhaps for people who leave their services running without actually watching), we want to ensure that we know what records were people actually watched the show, rather than hitting play and leaving. Thus, we created a feature for if it was viewed for greater than 2 minutes. 2 minutes is the 25th percentile, as well, for minutes\_viewed, and from a non-technical standpoint, it is around when somebody may commit to actually watching a show.

After applying these transformations and other clean-up methods, there remain five categorical predictor variables. We will one-hot encode each of these utilizing the get\_dummies module from pandas. These variables are source\_language, genre, platform\_type, country\_code and showtype.

To be able to utilize level\_2 traits as the target variable with built-in ML algorithms in Sagemaker, like XGBoost which requires all numerical values, we will use the LabelEncoder module from sklearn to assign them dummy variables as well as remove the headers from the datasets.

* *How will you balance your data set?*

Our target variable (level\_2 or level\_3 traits) has a vast range of sample sizes which can be from a maximum of 11000+ to a minimum of 1. We would delete the records with too few sample sizes and perform oversampling for the rest. This way we can work with a relatively large sample size and avoid overfit.

* *How will you split your dataset?*

The dataset will be split into 3 parts: train\_df, validation\_df and test\_df in a proportion of 90%, 5% and 5%.

**Model Training:**

*How will you train your model, what tools will you use?*

* *Will you be using SageMaker Jumpstart, built-in-algorithms, bring-your-own-script or bring-your-own container?*

We will be using either SageMaker Jumpstart or built-in-algorithms since they are less coding intensive and more user friendly.

* *Which algorithm will you be using?*

We plan on using XGBoost for multiclass classification as XGboost is a highly flexible and versatile tool that excels with these types of problems. It can, of course, be trained on AWS, and it allocates optimal usage of memory resources and computing time (Reinstein, 2017).

Since our goal is multi-class prediction/ classification, there are a few options in SageMaker built-in algorithms: [Factorization Machines Algorithm](https://docs.aws.amazon.com/sagemaker/latest/dg/fact-machines.html), [K-Nearest Neighbors (k-NN) Algorithm](https://docs.aws.amazon.com/sagemaker/latest/dg/k-nearest-neighbors.html), [Linear Learner Algorithm](https://docs.aws.amazon.com/sagemaker/latest/dg/linear-learner.html), and [XGBoost Algorithm](https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost.html). Among which, the Factorization Machines Algorithm targets either binary classification or regression problems while k-NN is an unsupervised machine learning algorithm which does not fit for our target. Thus, we are simply choosing XGBoost due to its easy-to-find tutorials and documentation that help us proceed and troubleshoot. We then will use the Linear Learner Algorithm if more time is allowed as a comparison.

* *Which parameters will you be passing?*

We will be using hyperparameters below:

* Num\_class =8: Since our objective is *multi:softmax* and our target variable (level\_2 traits) has 8 classes.
* Max\_depth = 5: We want to keep our model relatively less complex and avoid overfit, so we choose the depth of the tree as 5.
* Eta = 0.2: Used to aim for a more conservative boosting process.
* Gamma = 4: Same with eta = 0.2, aim for a conservative algorithm use.
* Min\_child\_weight = 6: We want the building process to not go for further partitioning in order to maintain a low cost, in which we want to make the tree partition step result in a leaf node with the sum of instance weight going below the limit of 6.
* Subsample = 0.7: Allow XGBoost to randomly select 70% of the data instances to avoid overfitting. This can be adjusted down to 50% if we find our model has an overfitting issue.
* Objective = ‘multi:softmax’: Since we are to predict level\_2 traits, we use multiclass classification as our objective. This also requires us to set a num\_class.
* Num\_round = 20: We set the training to run 20 times for shorter and quicker learning purposes.
* *Which instance size/count will you be using?*

For instance size we are using “1” for the low cost and type of 'ml.m5.large' the learner lab doesn’t allow 2xlarge sizes while retaining the capability of handling large data.

* *How will you evaluate your model?*

It is common and suggested that we can first make predictions of a given size of samples from our test data set to gain a glance of our model’s performance. Then we want to use a confusion matrix to check for our classifier’s performance with true positives, true negatives, false positives and false negatives counts.Then we will use accuracy scores to determine if our model has achieved our business goal (>85% accuracy). Lastly, it may be a good idea to check for system metrics such as CPU, GPU and memory utilization using CloudWatch to ensure that our training job is realistically related to the capacity of our imagined company. However, we should keep in mind that multi-class prediction is different from a binary classification and since we are using SageMaker’s built-in algorithm, the traditional scikit-learn functions may not be compatible with the environment.

### ***Measuring Impact:***

*List at least two specific metrics you expect to change with this project. These should tie back to the goals above.*

* From a business perspective, customer turnover and customer satisfaction. These metrics presumably have some correlation between each other, so Streamology wants to ensure that current customers are happy and stay and ,in turn, they will not turnover. So, cutting customer turnover by 50% is the goal, and the leading indicator would be customer satisfaction - which we’d like to increase by 25%.
* In order to increase customer satisfaction, reduce customer turnover rate and draw more customers in. We are expecting to achieve that by increasing the “confidence score” of our prediction. Since we do not know how the confidence score was calculated in the original datasets, we will use “accuracy” or “F1” score as our measuring metric to justify our model’s performance.

### **Security Checklist, Privacy and Other Risks:**

* *Will this store or process* [*PHI*](https://www.hhs.gov/answers/hipaa/what-is-phi/index.html) *data?*

No.

* *Will this store or process* [*PII*](https://www.dhs.gov/privacy-training/what-personally-identifiable-information#:~:text=%E2%80%9D%2C%20or%20PII%3A-,DHS%20defines%20PII%20as%20any%20information%20that%20permits%20the%20identity,U.S.%2C%20or%20employee%20or%20contractor) *data?*

No.

* *Will user behavior be tracked and stored?*

Yes, with consent.

* *Will this store or process credit card data?*

No

* *What S3 buckets will this application read or write to?*
* Initial datasets: s3://ads508-team4-raw
* Master datasets: s3://ads508-team4-master
* For built-in algorithm training, we have a bucket as s3://ads508-team4-split in which we have 3 subfolders as /train, /validation and /test for XGBoost algorithm to read from for model training.
* Then we have a bucket as s3://ads508-team4-xgboost with a subfolder called /models that contains our training job outputs.
* Optional: We also have a bucket called ads508-team4-jumpstart which contains 3 subfolders that are the same as s3://ads508-team4-split with all files named uniformly as “data.csv”. It is used by Amazon SageMaker Jumpstart to read training and validation files.
* *What data bias should be considered?*

We should be considering:

* False Causality: Since we have over 20 features in the datasets, we must perform additional research when it comes to feature engineering to decide what features to be used for modeling. As correlation does not imply causation, it is important to think deeply about the problem and to learn if things correlated with each other do cause one another or not.
* Confirmation Bias: As we are to predict traits for the customers who watch different shows, we may possess some pre-existing opinions on what features may be useful; However, to avoid encountering such bias we will need to take an objective approach to analyze the features and pick the most appropriate ones.
* Sunk cost fallacy: This project will require a lot of attention from the data team and results are critical. We must avoid spending more money if the long term effects do not pan out as expected, and we must be ready to pivot if our initial goals are not going as planned.
* Availability bias: The project has the potential to be biased based on personal taste in streaming content. To avoid this we must refrain from trying to find the connections we think we know. Also, the data has been collected just from this streaming company, so we must be aware of the potentially limited scope the data may provide.
* *Are there any ethical concerns with the data or business problem that should be addressed?*
* We need to make sure to check for data purity, “Garbage In Garbage Out”. If the datasets possess low quality data, we’ll need to clean them to make sure that our decisions are not going to be affected by faulty data.
* We need to be cautious with and respect the customer’s privacy since many of these features are private information and we should never leak our customer’s personal information.
* We must respect the intellectual properties from other resources and only use available public data when necessary.
* We also need to pay attention to any subjective biases mentioned before and not let our pre-existing knowledge overtake the big picture. Always keep an open mind and be sure to treat all data equally.

### **Future Enhancements:**

*Provide at least 3 ways you would improve your model pipeline if you had more time/ additional resources?*

**Adding new features:** One area that is intriguing for us to understand is how much a user may have watched a particular asset (TV show, movie, etc.). Utilizing the features minutes\_viewed and running\_minutes, we created a new feature pct\_watched to see how much they watched. Interestingly, its distribution was highly right skewed. After eliminating outliers, it was apparent there was a bimodal distribution with many either watching near 0% or near 100% of a given asset. Some watched over 100% as well. Thus, we bucketed these into four categories: 0 to 10 percent, 11 to 90 percent, 90 to 100 percent, and over 100 percent.

**Data Transformations:** Within the current dataset, we are predicting a customer trait for each time a user hits play on an asset. While this can be effective to find all the possible traits a user can have, it may be worth creating a larger ‘master’ dataset that has one single record per user along with their viewing habits, demographic information, and more. This would be tough to do without having as many columns as there are assets (which would be hundreds of features), but there would be a lot of opportunities for feature engineering to limit them. One way could be to group assets by genre and use that to predict which level\_3 trait that pertains to the genre the user likes. To take it a step further, perhaps we could cluster assets into groups, using variables like runtime\_minutes, genre, country\_code and others, to create groups that we then use as a feature in our final model.

After creating this new master dataset, we could use it to predict their traits, but also, we could create clusters of our customer base to better understand how to market to customers, if we should innovate the product towards each of those groups and what they like, and use this to further create a great customer experience. The better the experience we can provide, the more likely the customers will stay.

**Additional Model:** As discussed in the “Model Training” section, our next approach for model selection would be training the SageMaker built-in Linear Learner algorithm as a comparison to the XGBoost that we used.

Linear Learner algorithm, if used in a multi-class classification model, behaves much similarly to the XGBoost:

* It requires a data matrix, with rows representing the observations, and columns representing the dimensions of the features.
* The target variable needs to be numerically labeled with num\_class -1 amount of labels.
* It also requires an additional column that contains the labels that match the data points along with specified input and output file path (S3 bucket in our case).

The Linear Learner algorithm also uses similar hyperparameters and provides us with the best solution from a validation set while we can simultaneously explore different training objectives (2019).

**Pipeline Improvement:** To improve the model pipeline, we would want to deploy our model as a real time API to provide low-latency predictions on single prediction requests through HTTPS, because we want to provide the end users instant response of content recommendations based on a single click.One of the options available is to use the REST API protocol through SageMaker Endpoints to deploy, scale and compare our models (Fregly & Barth, 2021).

Once the model(s) has been deployed to the endpoint, we can complete the inference pipeline and perform the following:

* Track the model deployment in Experiment and analyze the Experiment Lineage;
* Auto-scale the endpoint using Amazon Cloudwatch;
* Perform A/B testing to compare our XGBoost and Linear Learner models;
* Setting up data capture to the endpoint;
* Monitor model performance and data quality.

**Optimization:** Additionally, if given more time and additional resources, we would explore how to reduce costs and improve performance by utilizing Sagemaker Step Catching. Our team would set up a cache configuration on each step, which would enable us to avoid the re-execution of steps that have not changed. Benefits of step caching include saving time and resources when Sagemaker detects that the raw dataset and processing parameters have not changed allowing Sagemaker to use the cache and speed the pipeline. Step catching can be a factor that will enhance user experience, especially when show recommendations become a greater focus for the company. The content available to customers will keep growing, but only new releases will need to be evaluated, steps will be cached saving resources and time. Overall, it’s a great tool that will further reduce the response time of content recommendations based on a single click.

Another thing to keep in mind, is ensuring the longevity and cost effectiveness of the project as a priority. Using Spot Instances, via SageMaker’s built-in capabilities, is one way to accomplish that goal. Setting up Spot Instance checkpointing, along with adding in stopping rules, would help save on resources and mitigate costs. Once training job predictions plateau and no longer improve, stopping rules will ensure there aren’t any dead costs (Fregly & Barth, 2021).

**References:**

Reinstein, I. (2017. October, 3). XGBoost, a Top Machine Learning Method on Kaggle, Explained. *KD Nuggets.* <https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html>

Mishra, A. (2019). *Machine learning in the AWS cloud: Add intelligence to applications with Amazon Sagemaker and Amazon Rekognition*. Amazon. Retrieved April 6, 2022, from <https://docs.aws.amazon.com/sagemaker/latest/dg/linear-learner.html>

Fregly, C., & Barth, A. (2021). *Data Science on Aws: Implementing end-to-end, continuous Ai and Machine Learning Pipelines*. O'Reilly.