Term Project

DSCI-644 Fall 2020 | RIT | TEAM-B

Amazon Reviews - Building a Better Predictive Model

A Software Engineering for Data Science Project

11/18/2020

Contents

I. The Team

Alex Pataky

Heather Akers-Healy

Jacob Miller

Matt Agone

Susan George

II. Executive Summary

1. Purpose

The purpose of this project has been to follow a typical software life cycle model from beginning through end, while improving an existing machine learning model to classify Amazon reviews using sentiment analysis. OpenUp Agile methodology has been followed. The wrapping of the final optimized model in a user-friendly interface is discussed, to allow quick and easy positive/negative predictions to new review text.

2. Motivation

Customer opinion is vital in drawing useful insights, improving products and services, making informed decisions, and taking actionable steps. Sentiment analysis categorizes the unstructured data from customer perceptions driving branding and marketing. An automated solution for categorizing Amazon reviews by sentiment will allow greater flexibility for our customers in utilizing reviews, ultimately making it easier for consumers to glean product information without reading excessive amounts of reviews.

3. Goal

The goal of the project has been to improve the accuracy and reliability of the existing machine learning model by following a typical software life cycle process. The original model made predictions with an accuracy of ~60%, which is no better than random guessing when taking into account the spread of the data. The original model exclusively predicts a rating of 1.0, which accounts for 60% of the entire dataset, and therefore results in the 60% accuracy.

For our model, target labels have been transformed into binary (positive and negative) sentiment. We have improved the overall accuracy, and achieved good performance on precision, recall, F1 and AUC (area under the ROC curve).

III. Requirements

- An algorithm will consume text review data and return a predictive score for the numerical rating of that review, either positive (1) or negative (0)
- The model will be a classification algorithm
- Training Model Accuracy will exceed 60% to improve on the original model
- The algorithm will be accessed via API on a public website
- The User Interface for this API must have a text box where users can paste a review text and submit
- The API will return a score within 1 second after the review is submitted, or the page will update with a progress bar if the process is delayed
- The User Interface will make the predicted binary score available to the user

IV. Solution Architecture

For the term project, our solution architecture is limited to MS Azure ML Studio. We delivered our improved model and outputs via ML Studio directly.

In addition, we have drawn out what the architecture might look like for a production deployment, where we would have users interact with a simple UI attached to API endpoint. In the UI, users can input review text and get a "pos/neg" sentiment returned. We would have our reference data in Azure storage and use ML Studio to build the model. The model and API would be deployed using an AKS (Azure Kubernetes Service), since we want to have a real-time interaction and response.

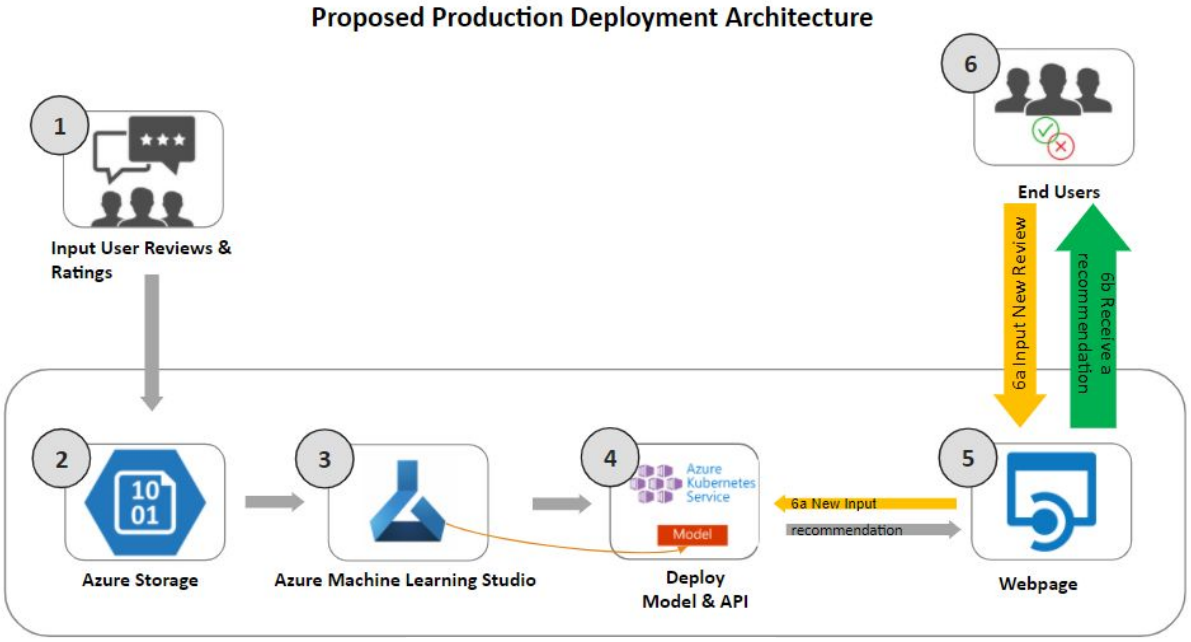


Fig.1 Solution Architecture

1. Data Storage for Initial Dataset

Azure Storage - ML Studio (Comma Separated Value File)

Data is run through model hosted on Azure

2. User Interface

UI for this Proof of Concept: outputs via Azure ML Studio

Proposed production UI: Simple UI attached to API endpoint where user can input review text and get a “pos/neg” sentiment returned

3. High Level Design and Component Interactions for the Proposed System

Architectures based on MS reference diagrams at:
https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/ai/real-time-recommendation
and
https://docs.microsoft.com/en-us/azure/architecture/example-scenario/ai/movie-recommendations.

Our model differs in that we are not using Databricks or Cosmos DB, because we are using the functionality within Azure ML Studio to handle any transformations.

4. Workflow

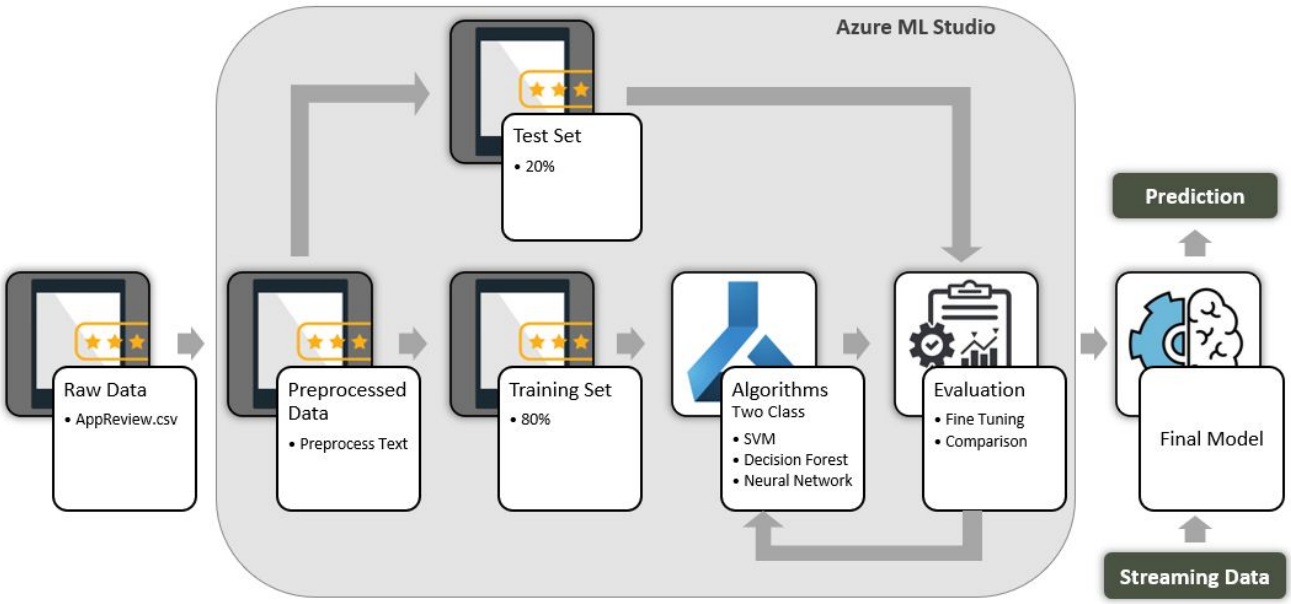


Fig.2 Workflow

5. Version Control

GitHub branching and merging is leveraged for the development collaboration and version control of the project website.

6. Deployment

Using Azure we will deploy the model via a web service. The web service provides an API endpoint which requires the endpoint in the request header and the review text in the request body. The response will include the predicted score for that review as a binary response, 1 or 0.

Since this model will be publicly available, neither tokens nor keys will be required to access the API from the UI.

V. Design

1. Data Processing

Targets were converted into binary outcomes. Original data contained six possible ratings (0-5 stars, which were translated to ratings of 0.0, 0.2, 0.4, 0.6, 0.8, 1.0 to appropriately feed into the algorithm), some of which were infrequent or non-existent, so a threshold was set at 0.7: scores greater than 0.7 were given a positive sentiment while scores less than 0.7 were given a negative sentiment. This skewed threshold was selected because the scores were largely skewed towards the higher ratings (1.0, 0.8).

An SQL transformation was applied to create the threshold for a positive or negative review.

The review text was preprocessed to force the text into a more digestible format for model training. Stop words and extraneous text such as email addresses introduce noise to a text mining process. The comment data was cleansed before the model was trained.

The input text is fed through the Azure "Extract Key Phrases" module so that the model will focus on the more meaningful text within the review data.

Latent Dirichlet Allocation is applied to split the input review data into similar groups, to aid in categorizing the text as a predictor of a rating.

2. Training the Model

A model was tested using Two-class Support Vector Machine. This algorithm was chosen because it is a convenient and well-researched algorithm for predicting binary outcomes. Another model was tested with Logistic Regression, which appeared to offer superior accuracy for the binary-formatted data.

Two-class Decision Jungle also looked promising as a high performance classifier and is the algorithm chosen for the final model.

3. Testing and Performance

Our initial iterations of binary classification models, as tested by running on Azure, had more success than the initially delivered model. Some models were getting a low number of accurate negative predictions, whereas the original model only predicted ratings in the highest category.

Various model assessment measures were used to aid in the final model selection. This includes the AUC, F1, Precision and Recall scores related to the ROC curve, as well as the model's general accuracy.

The final model utilizes Synthetic Minority Over-sampling Technique (SMOTE) to rebalance the classes by synthetically generating additional samples of the negative reviews. This was key to the high improvement of performance in the final model.

4. User Interface and Deployment

Design and Test Interface:

The UI in use is Azure ML Studio directly.

Intended Production Use:

The model would be deployed on the web using Azure Kubernetes Service (AKS) with a user-friendly API on top. AKS is a good choice for real-time inference executions, which will be ideal in a production scenario.

UI functionality:

The user enters a text review, and our model returns the binary prediction.

5. Risk Management

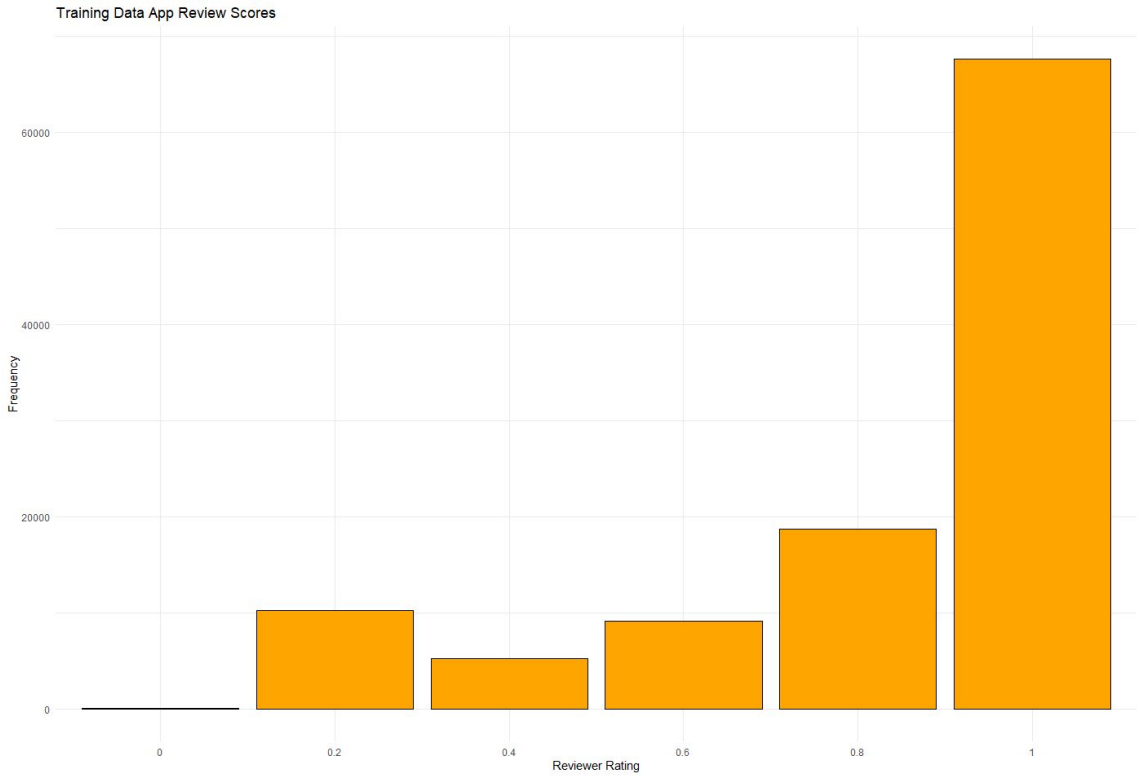
Risk Number	Risk	Likelihood (0-1)	Impact (1-10)	Risk Score	Mitigation
1	Model process failure	0.1	10	1	MS Azure ML has high availability, the likelihood of a system failure is low and any downtime would be minimal.
2	Model scoring failure	0.2	8	1.6	Depending on the algorithm, some models are more likely to degrade over time than others. In addition, some algorithms do not provide details about the score, and incorrect predictions may be difficult to explain. This risk will be taken into account during the model selection and evaluation steps.
3	Data input (user error)	0.6	4	2.4	Users inputting non-text or non review data will be scored unless error handling is built into the model. This risk is not captured in the requirements and any documentation created or user manuals should explain how to use the product.

VI. Model Development

1. Data Exploration

The raw dataset contains an imbalance of targets. Perfect review scores represent 60% of the total dataset, and targets in the top two buckets that are transformed into our positive label represent nearly 80% of the data.

Score distribution:



Conducting sentiment analysis confirms that the imbalance exists in the review text as well. Positive words represent a 4:1 majority after removing stop words.

Negative Words	Positive Words	Overall Sentiment
49,711	197,334	147,623

Word	Sentiment
love	positive
well	positive
worth	positive
super	positive
easy	positive
miss	negative
swipe	negative
issue	negative
awful	negative
bug	negative
ruined	negative
free	positive
like	positive
awful	negative
bugs	negative

Word clouds created on our two label categories validate that the vocabulary used most frequently in the review text is indeed different. There is some crossover where positive words are used in negative reviews, as expected, but generally the word clouds are in line with expectations.

2. Data Preprocessing

- Targets were converted into binary outcomes. Original data contained six possible ratings (0-5 stars), some of which were infrequent or non-existent, so a threshold was set at 0.7: scores

greater than 0.7 were given a positive sentiment while scores less than 0.7 were given a negative sentiment.

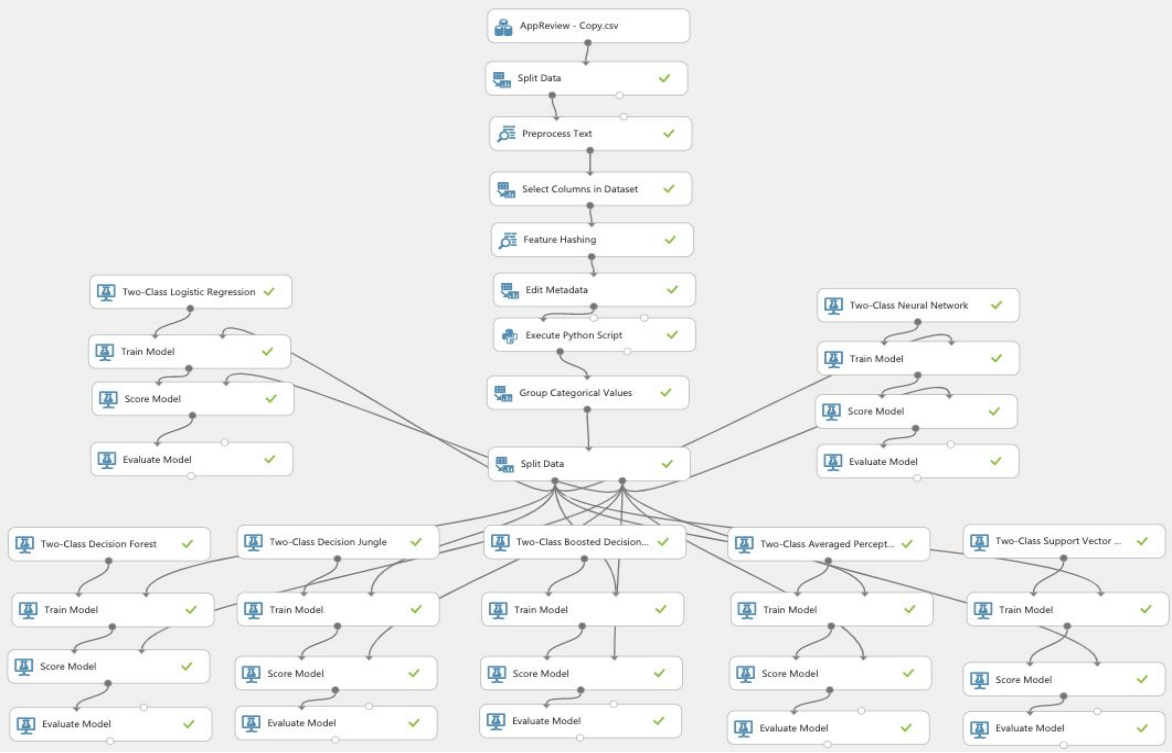
- This skewed threshold was selected because the scores were largely skewed towards the higher ratings (1.0, 0.8).
- A SQL transformation was applied to create the threshold for a positive or negative review.

- The review text was preprocessed to force the text into a more digestible format for model training. Stop words and extraneous text such as email addresses introduce noise to a text mining process. The comment data was cleansed before the model was trained.
- The input text is fed through the Azure "Extract Key Phrases" module so that the model will focus on the more meaningful text within the review data
- Latent Dirichlet Allocation is applied to split the input review data into similar groups, to aid in categorizing the text as a predictor of a rating

3. Models Tested

Various models were researched and tested through Azure. One model utilized feature hashing and text preprocessing. Two other models in development used Latent Dirichlet Allocation. The final model combines text preprocessing, Extracting Key Phrases From Text, Latent Dirichlet Allocation and SMOTE with a Two-class Decision Jungle algorithm.

Some of the preliminary binary classification model iterations can be seen below. Azure ML Studio allowed for quick and easy iterating so we were able to quickly decide on one algorithm family (2-Class Decision Jungle) and focus on optimization.



4. Final Model

As stated previously, we researched and experimented with various two class algorithms and data processing steps such as feature hashing and text preprocessing. After extensive testing and evaluation, we chose a model using the **Decision Jungle** algorithm with an **SQL transformation** to split the reviews into binary classes, and the text preprocessing, **Extracting Key Phrases From Text**, **Latent Dirichlet Allocation** and **SMOTE** modules from Azure.

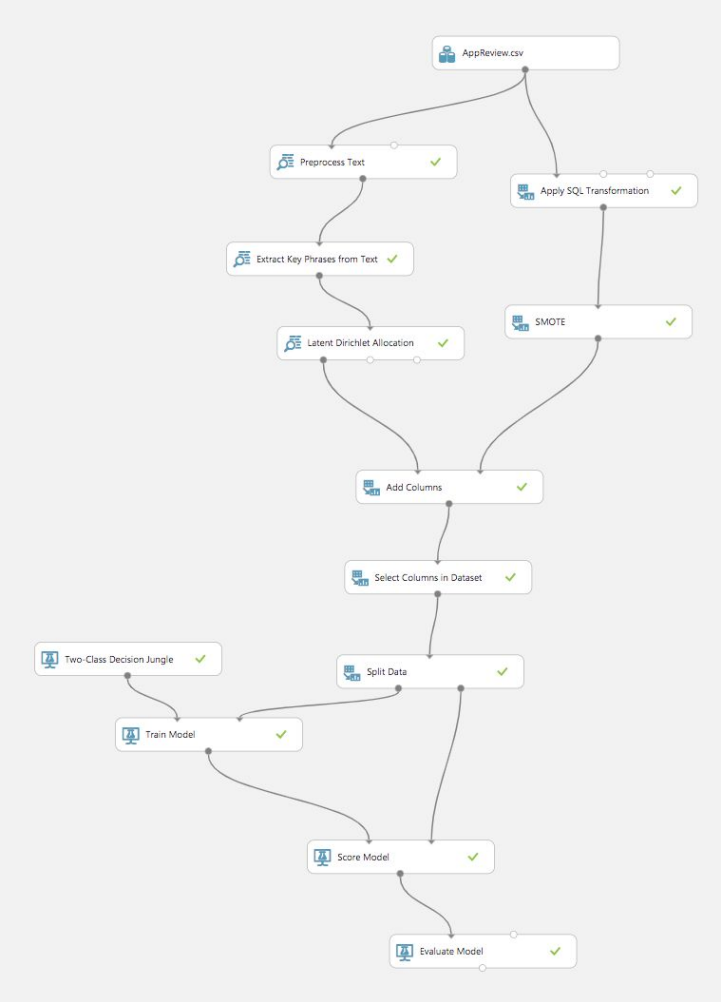
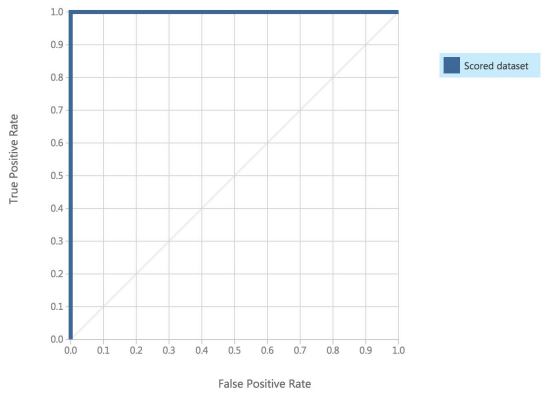


Fig.3 Final Model from Azure ML Studio

5. Final Model Metrics

At a .5 threshold, the final model achieves accuracy of .926, Precision and AUC of 1.00, Recall of .863, and F1 of .926.



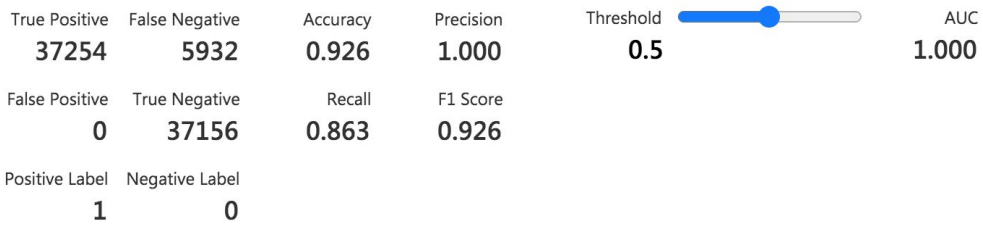


Fig.4 Final Model Metrics

6. Model Building Tools

All models were developed and tested in Microsoft Azure Machine Learning Studio. Additional research was done to explore and evaluate machine learning methods and algorithms.

VII. Application of Software Engineering

1. Process

Open Unified Process (OpenUP) was used for the software engineering lifecycle of the project. The project progressed in bi-weekly sprints through the major phases of Inception, Elaboration, Construction and Transition. The major artifacts are the Project Proposal (with detailed requirements), Architecture & Design, the Final Model, Project Presentation and the Project Report.

Phase	Tasks and Artifacts

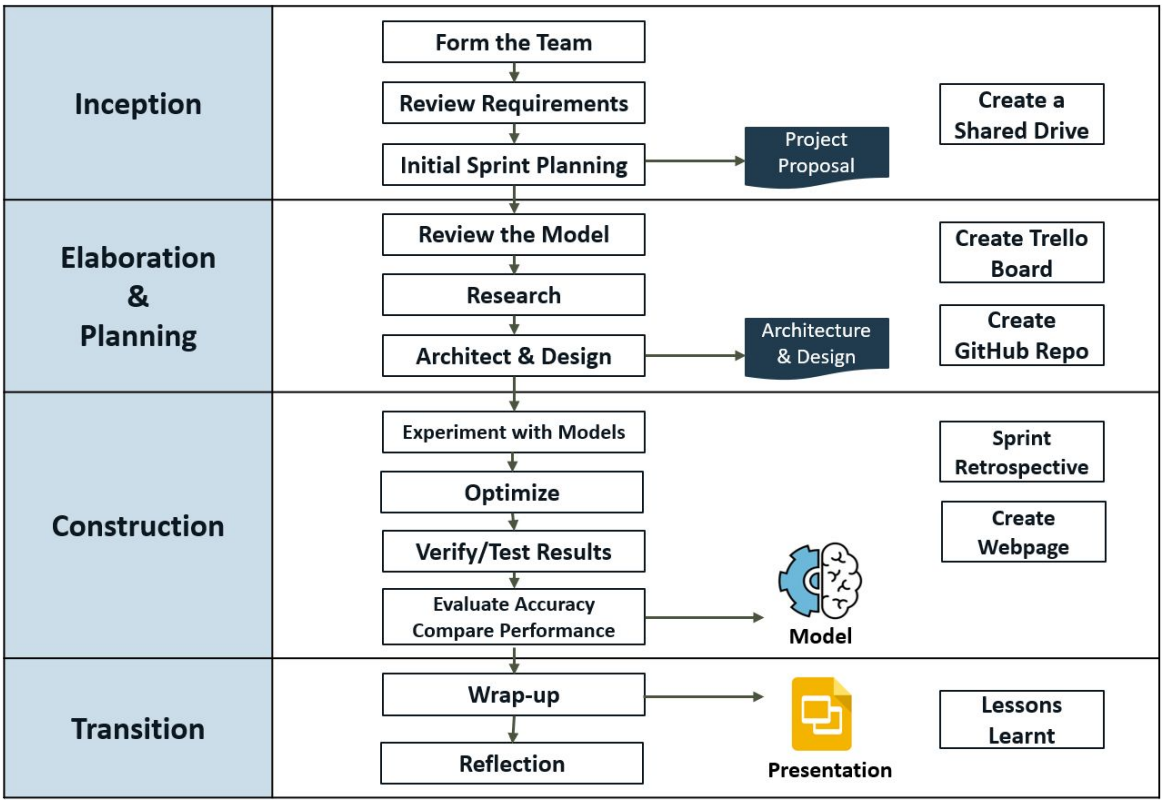


Fig.5 Task Flow

2. Software Engineering and Project Management Tools

Trello (https://trello.com/b/UvRzi2Zz/dsci644-team-b-term-project)

Google Drive (https://drive.google.com/drive/u/1/folders/0AP8vltTBmTtTUk9PVA)

Slack (https://app.slack.com/client/TP0FX0QAU/C019MGABM6H)

Webex/Zoom

Microsoft Azure ML Studio

Github (https://github.com/MattAgone/DSCI644-Team-B-Project/)

Project Website (https://hrahhrah.github.io/)

VIII. Conclusion

Team B has successfully completed the project. A new classification algorithm has been constructed and the software design for the implementation is in place. The requirements have been satisfied per the customers and outstanding risks have been documented and mitigated. The team used software engineering principles such as OpenUp to organize the work, and software such as GitHub and Trello to manage the project.

As students, this project provided a good opportunity to create and manage a data science project through the software life cycle. The GitHub repositories and Trello project management boards to organize the work with a website to communicate externally created an environment where collaboration was a necessity. There were less opportunities for members to work independently and stitch the project together this way, which is a positive thing. Microsoft ML Studio as well as a poorly performing model also allowed us to learn a new tool and experiment with various concepts without getting totally mired in the potential pitfalls of model tuning. Working together while spread out across different states and time zones was another opportunity to learn remote collaboration, a particularly useful skill in 2020.

IX. References

1. Mukherjee, A., Mukhopadhyay, S., Panigrahi, P. and Goswami, S. (2019). 'Utilization of Oversampling for multiclass sentiment analysis on Amazon Review Dataset,' IEEE 10th International Conference on Awareness Science and Technology (iCAST), Morioka, Japan. pp. 1-6, doi: 10.1109/ICAwST.2019.8923260.
2. Yang, S., Zhang, H. (2018). 'Text Mining of Twitter Data Using a Latent Dirichlet Allocation Topic Model and Sentiment Analysis'. World Academy of Science, Engineering and Technology, Open Science Index 139, International Journal of Computer and Information Engineering, 12(7), 525 - 529.
3. Elreedy, D., Atiya, A. (2019). 'A Comprehensive Analysis of Synthetic Minority Oversampling Technique (SMOTE) for handling class imbalance', Information Sciences, Volume 505, 2019, https://doi.org/10.1016/j.ins.2019.07.070.
4. Mohammed, A., Hassan, M., Kadir, D. (2020). 'Improving Classification Performance for a Novel Imbalanced Medical Dataset using SMOTE Method'. International Journal of Advanced Trends in Computer Science and Engineering. 9. 3161-3172. 10.30534/ijatcse/2020/104932020.
5. Microsoft. (2019, May 06). ML Studio (classic): Two-Class Decision Jungle - Azure. Retrieved November 19, 2020, from https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-decision-jungle
6. AzureML Team for Microsoft. (2014, September 02). Compare Binary Classifiers. Retrieved November 19, 2020, from https://gallery.azure.ai/Experiment/b2bfde196e604c0aa2f7cba916fc45c8