Team B Architecture and Design Documentation

Project management links

Github: https://github.com/MattAgone/DSCI644-Team-B-Project
Trello: https://trello.com/b/UvRzi2Zz/dsci644-team-b-term-project
Webpage: https://hrahhrah.github.io/

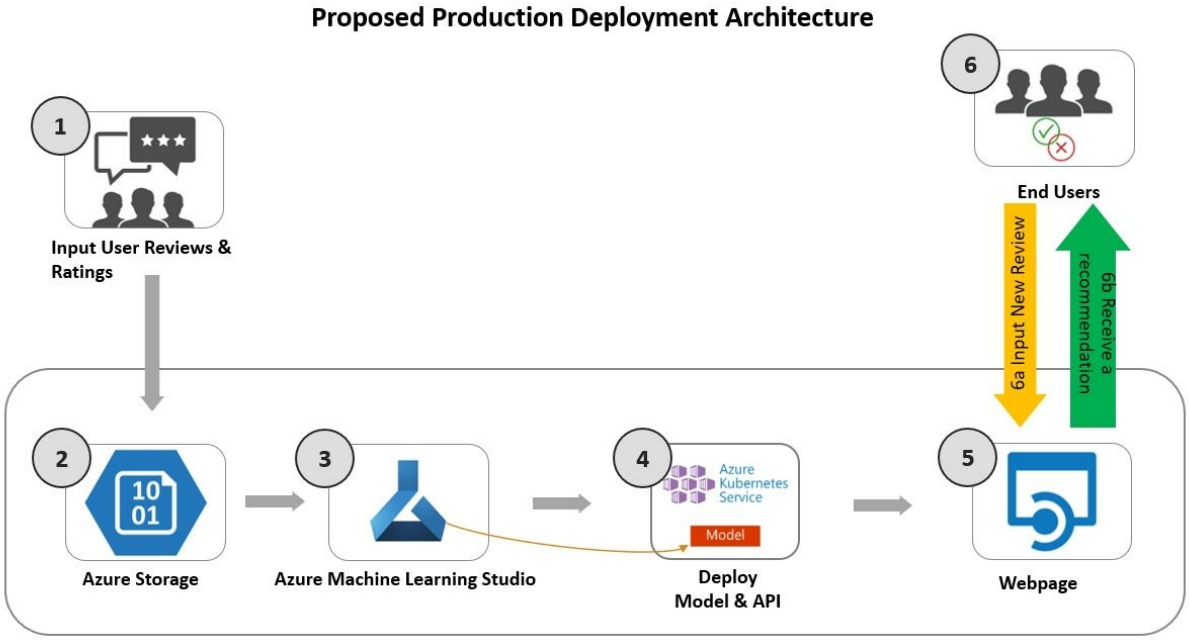
Solution Architecture

- Data storage for initial dataset
    - Azure Storage - ML Studio (Comma Separated Value File)

- Data is run through model hosted on Azure

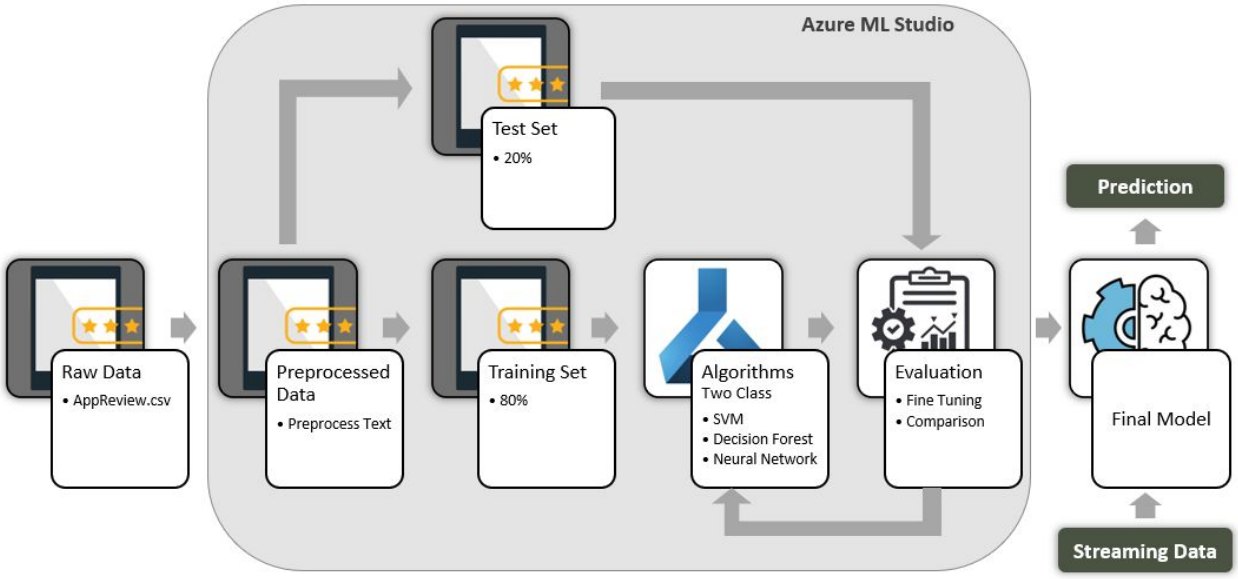
- User Interface
    - UI for this POC: outputs via Azure ML Studio
    - Proposed production UI: Simple UI attached to API endpoint where user can input review text and get a “pos/neg” sentiment returned

- 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.

- Workflow Diagram



- Deploy in Azure
    - 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.

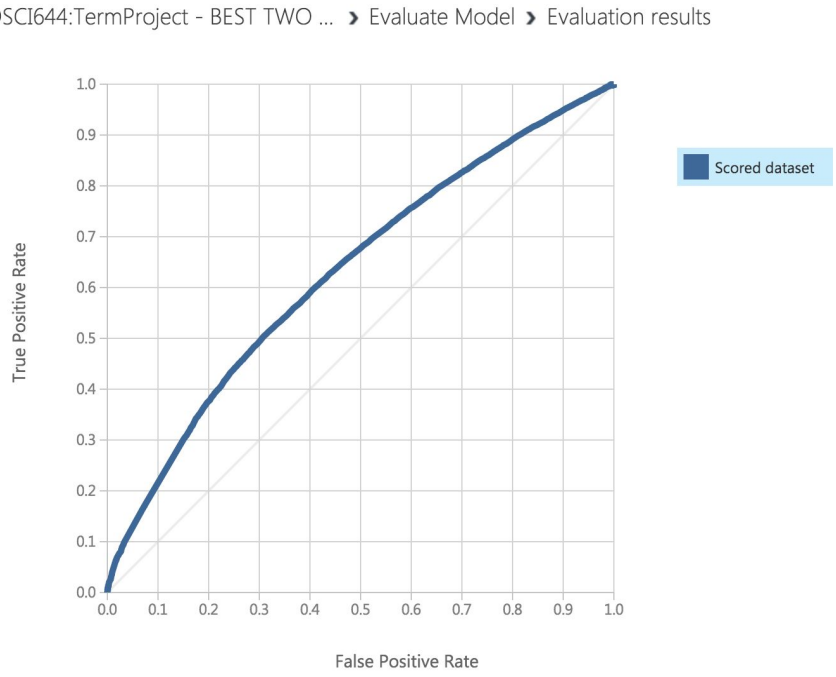
Model Design

- Data Processing
    - 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
    - **Our data processing methods may change as we test and refine various binary outcome models**

- Training the Model
    - A model was trained using Two Class Support Vector. This algorithm was chosen because it is a convenient and well-researched algorithm for predicting binary outcomes.
    - Another model has been tested with logistic regression, which appears to offer superior accuracy for the binary-formatted data.
    - Two-class Decision Jungle also looks promising as a high performance classifier (see https://gallery.azure.ai/Experiment/b2bfde196e604c0aa2f7cba916fc45c8).
    - **As with the data processing step, the training algorithm may change as we test the performance of various models**

- Testing and performance
    - Our current binary classification model as tested by running it on Azure has more success than the initial model.
    - We plan to improve the accuracy beyond the original 60%.

- We are getting some accurate negative predictions, whereas the original model only predicted ratings in the highest category.
- Various model assessment measures will be used to aid in the final model selection. This will include the AUC, F1, Precision and Recall scores related to the ROC curve, as well as the model's general accuracy.



User Interface and Deployment

Design and test interface

- In the design and test phase we're in, the UI in use is Azure ML Studio directly.

Intended Production Use

- The model would be delivered to end users through a UI on top of Azure API
- Microsoft has good documentation on building an interface on top of your ML models at: https://docs.microsoft.com/en-us/azure/machine-learning/how-to-deploy-and-where?tabs=azcli. We would use this as a guide.
- The chosen deployment would use Azure Kubernetes Service (AKS). That is a good choice for real-time inference executions. The reason for this is in a production scenario, we want a user to be able to present a text review and get a recommendation in real-time. Given that, a batch approach wouldn't meet our needs.
- UI functionality:
    - The user enters a text review, and our model returns the binary prediction