Project
Status Report 2
11/7/2020

Team Members: Adarsh Balakrishnan, Daniel Gilroy, Brandon Vidro

The method we chose:

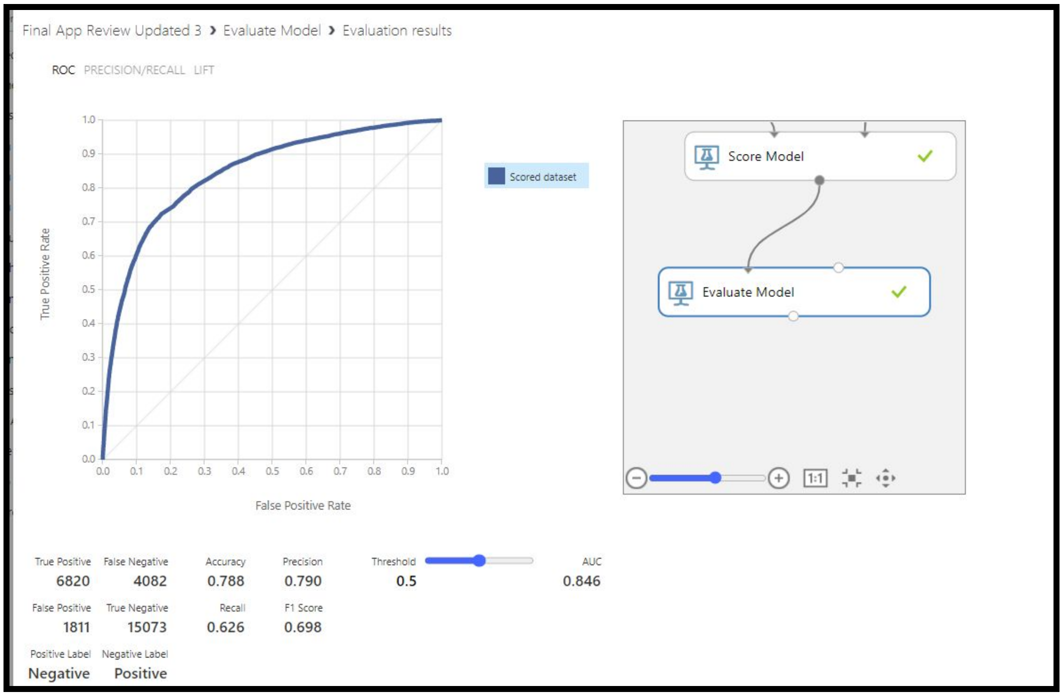
1) Grouped reviews into 2 categories (Positive and Negative) - any values below .5 were assumed Negative and any values about .5 were assumed Positive
2) Clean Data -
    a) Remove Stop Words
3) Final application allows a user to enter a review as user input and the model will predict whether that review is Positive or Negative.

We developed the model using Microsoft Azure ML Studio and tracked our overall progress of the project using Trello. Other technologies utilized include PyCharm, Jupyter Notebooks, Python (Execute Custom Python Scripts inside of Azure), Git

Trello board: https://trello.com/b/4XceHD7e/term-project
Git Repository: https://github.com/bwvidro/dsci644_team_d

**Model**: The initial model used a neural network that would give about 60% accuracy based on the reviews. The dataset was split and 75% of the dataset was used for training and 25% was used for testing.

**Overall results of current model:**



Final App Review Updated 3 › Evaluate Model › Evaluation results

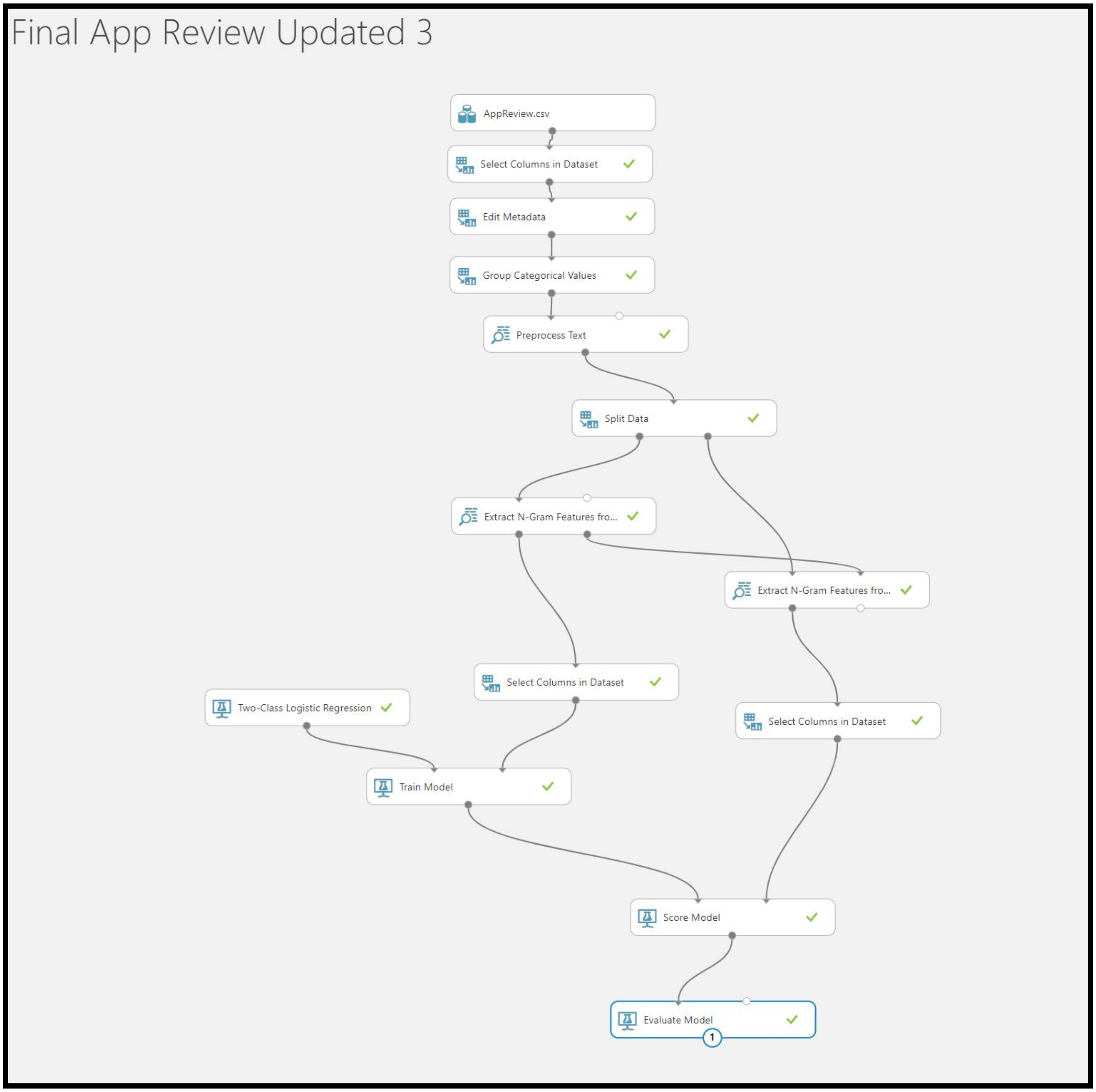
True Positive 6820
False Negative 4082
Accuracy 0.788
Precision 0.790
Threshold 0.5
AUC 0.846

False Positive 1811
True Negative 15073
Recall 0.626
F1 Score 0.698

Positive Label Negative
Negative Label Positive

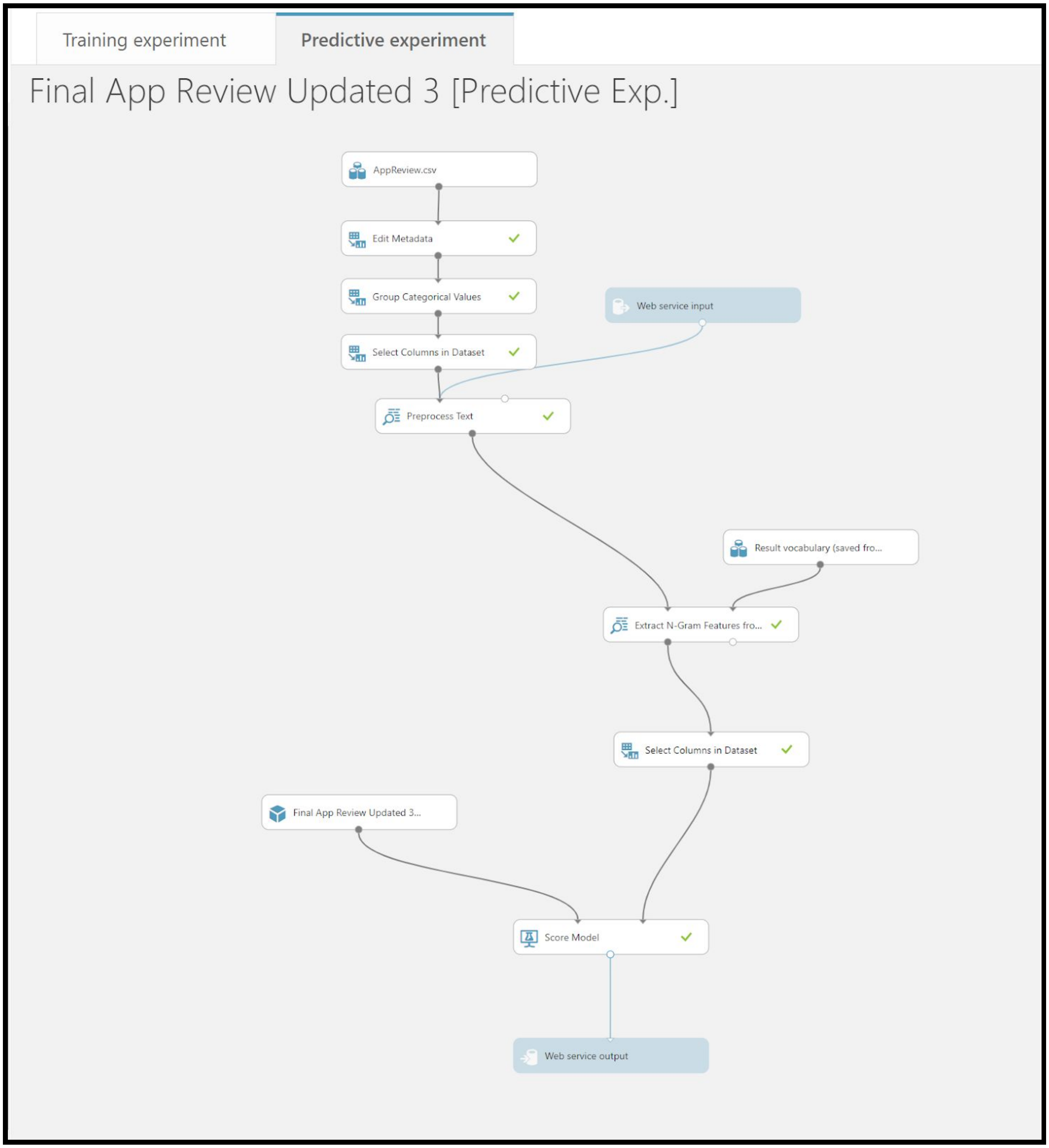
**Model Training Structure:**

Note the vocabulary was saved to a dataset which was used in the subsequent prediction


Final App Review Updated 3
AppReview.csv
Select Columns in Dataset ✓
Edit Metadata ✓
Group Categorical Values ✓
Preprocess Text ✓
Split Data ✓
Extract N-Gram Features fro... ✓
Extract N-Gram Features fro... ✓
Select Columns in Dataset ✓
Two-Class Logistic Regression ✓
Select Columns in Dataset ✓
Train Model ✓
Score Model ✓
Evaluate Model ✓
1

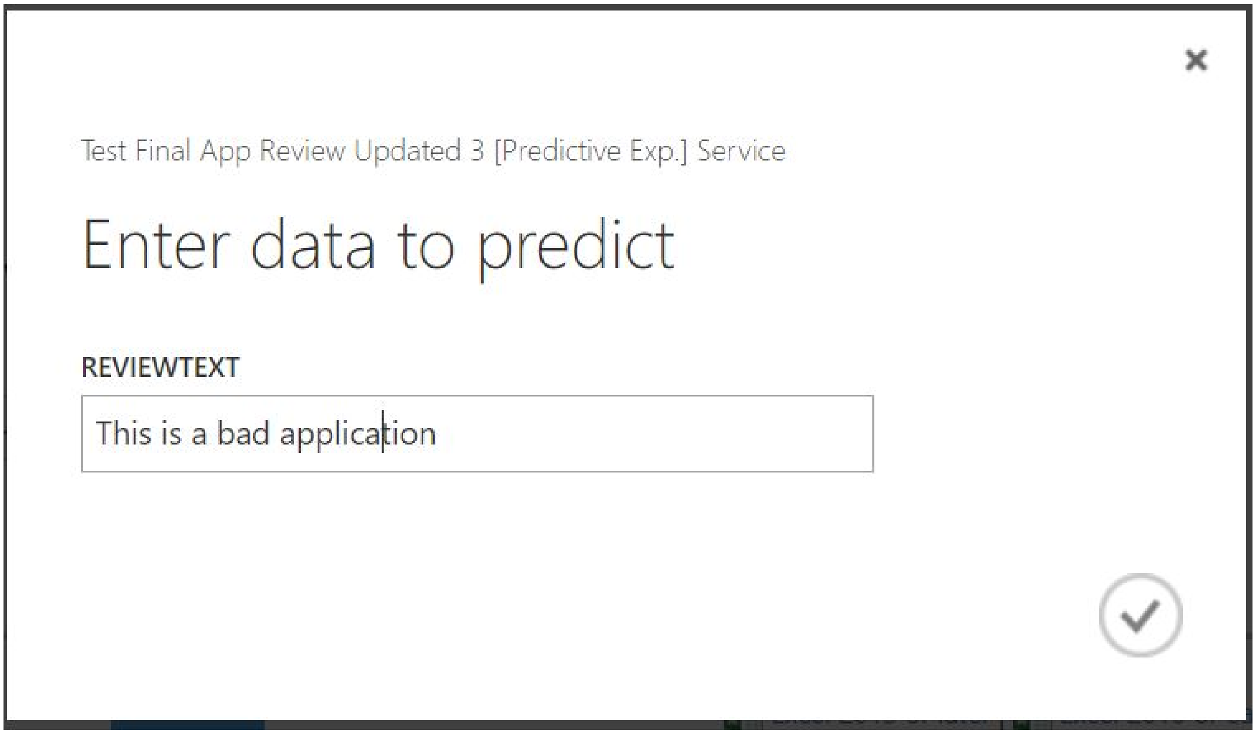
**Model Predictive Structure:**

Simple API for prediction, input is review text only, output is prediction of review


Training experiment
Predictive experiment
Final App Review Updated 3 [Predictive Exp.]
AppReview.csv
Edit Metadata
Group Categorical Values
Web service input
Select Columns in Dataset
Preprocess Text
Result vocabulary (saved fro...
Extract N-Gram Features fro...
Select Columns in Dataset
Final App Review Updated 3...
Score Model
Web service output

**Final solution interface for client consumption:**

**Input**:



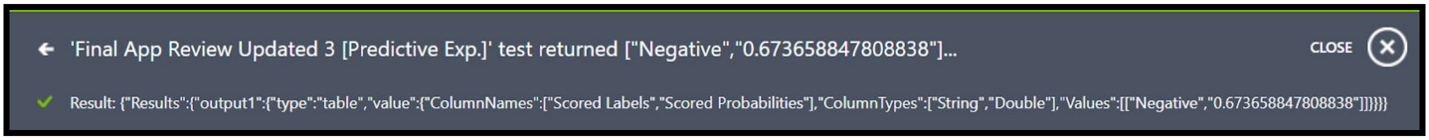
Test Final App Review Updated 3 [Predictive Exp.] Service

Enter data to predict

REVIEWTEXT

This is a bad application

**Output:**



'Final App Review Updated 3 [Predictive Exp.]' test returned ["Negative","0.673658847808838"]...

✓ Result: {"Results":{"output1":{"type":"table","value":{"ColumnNames":["Scored Labels","Scored Probabilities"],"ColumnTypes":["String","Double"],"Values":[["Negative","0.673658847808838"]]}}}}

**Details of response API in Json - showing 67% change of the review being negative:**

```
Result: {
   "Results": {
      "output1": {
         "type": "table",
         "value": {
            "ColumnNames": ["Scored Labels", "Scored Probabilities"],
            "ColumnTypes": ["String", "Double"],
            "Values": [["Negative", "0.673658847808838"]]
         }
      }
   }
}
```

**Details of sprint work**

**Initial Investigations:**

1) Cleaning data
    - a) Removed stop words
    - b) Cleaned out non-UTF8 encoded data
        - i) **This had an adverse effect on the outcome, and not implemented**
    - c) Remove non-letters -
        - i) review_text = re.sub("[^a-zA-Z]"," ", review_text)
    - d) Convert words to lower-case and split them
    - e) Filter out non-english words
        - **i) This had an adverse effect on the outcome, and not implemented**
2) Sampling data
    - a) After reviewing the data and noticing it had unbalanced classification. The distribution was negatively skewed meaning most of the reviews were positive (skewed to the right).
    - b) We adjusted sampling to ensure all classes were equally represented, using sampling with replacement if necessary.
        - **i) This turned out to be better handled in the modelling tool selected instead of adjusting initial sample**
3) Word vectorization
    - a) Went from Latent Dirichlet Allocation to Tf-Idf vectorization of words
        - i) Tf-Idf tried various ranges, 1000 words seemed to be enough
4) Modeling
    - a) Attempted the following
        - i) Default neural network
        - ii) Logistic regression
        - iii) SVM
        - iv) Random Forest
    - **b) Logistic regression was selected based on best learning for the data**

**Configuration Management:**

1. Initial Research and development
    - a. Performed on various individual developers machines
        - i. Pycharm
        - ii. Jupyter
    - b. Code versioning tracked and shared via git
2. Initial implementation of solutions
    - a. Azure ML
        - i. Each developer leveraged their own workspace for prototyping
        - ii. Sharing of work done via experiments and coping experiments to each others workspaces
    - b. Versioning done via naming conventions