Building a better predictive model for text review scores (...across 3 time zones)

Alex

Sprint A: Inception

Purpose Statement

Develop an improved algorithm that classifies Amazon reviews, predicting their sentiment rating at a rate that improves upon the existing accuracy of the model in place.

- re-design and re-train the model
- plan architecture for solution
- implement and run improved model
- compare accuracy to existing model
- iterate over changes
- document and communicate the process and results

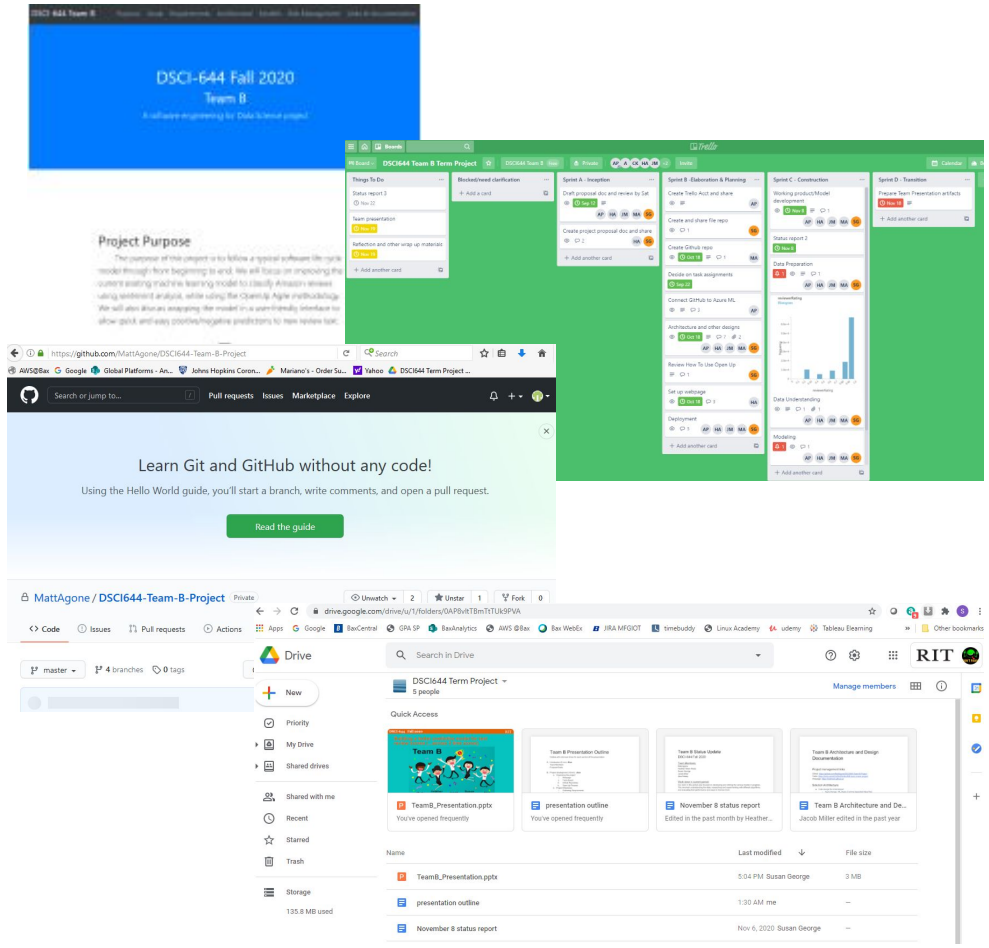
Alex

Sprint B: Elaboration and Planning

Project Development and Design

Project Development - Organization

- Webpage
  - https://hrahhrah.github.io/
- Open Up project management
- Trello
  - https://trello.com/b/UvRzi2Zz/dsci644-team-b-term-project
- Github
  - https://github.com/MattAgone/DSCI644-Team-B-Project
- Dev File Share
  - https://drive.google.com/drive/folders/0AP8vltTBmTtTUk9PV
- Communications
  - Zoom & Slack



Alex

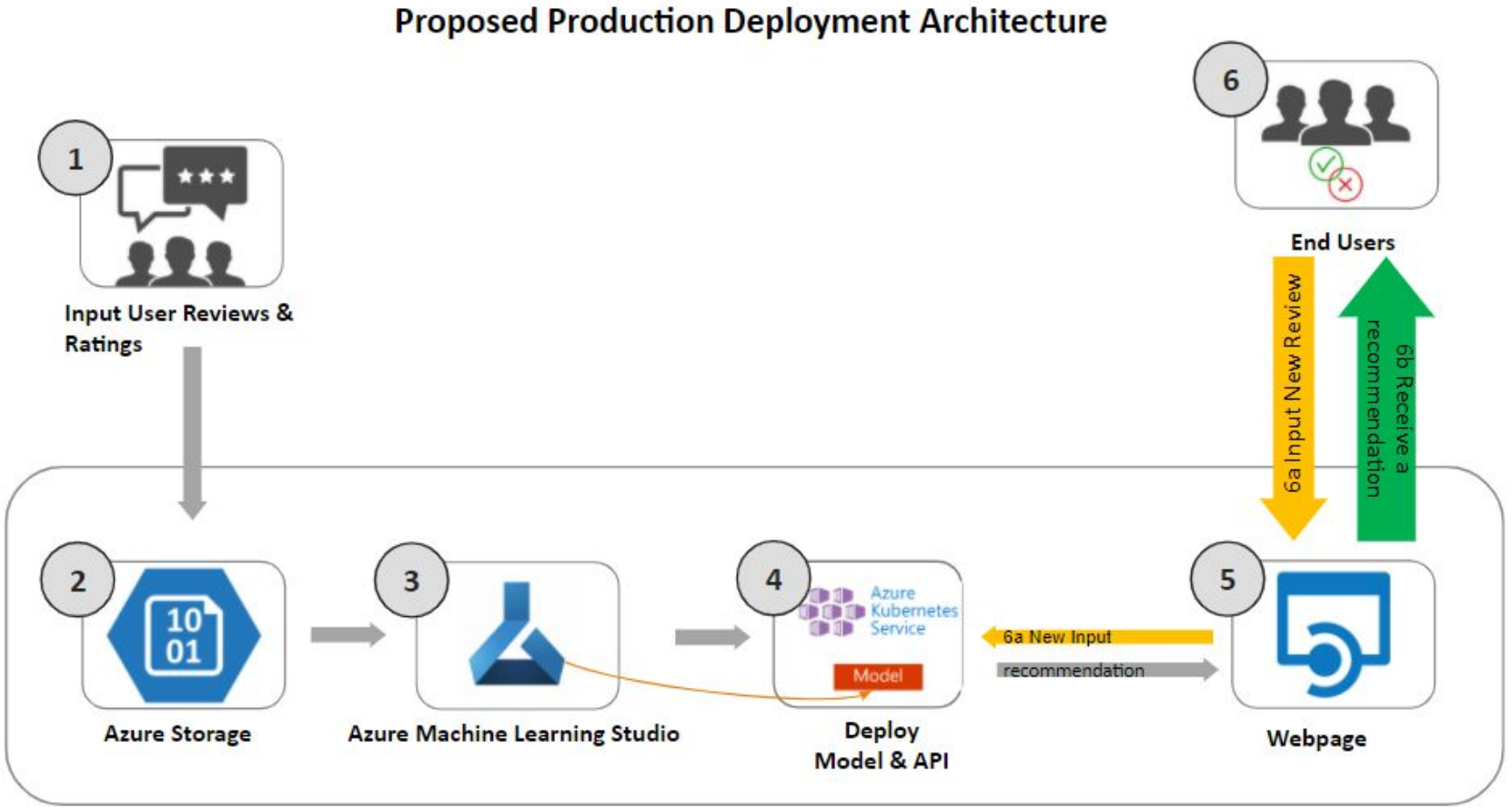
Project Development

Gathering Requirements

- Improve existing accuracy (60%)
- Use training data inputs
- Sentiment Analysis
- Provides positive negative class as output
- Fast, reliable and easy to use web-based tool

Alex

Architecture Design



Alex

Sprint C: Construction
Predictive Model Development

Data Understanding

Word Cloud -

Scrubbed review text

Matt


work
doesnt
good
please
time
update
cant
phone
just
working
screen
like
use
great
get
now
fix
dont
android
crashes
will
can
even
works
one
using


great
good
love
works
like
best
one
use
awesome
just
nice
can
work
perfect
better
thanks
simple
easy
excellent
really
phone
android
apps

Matt

Negative Reviews

Positive Reviews

Matt

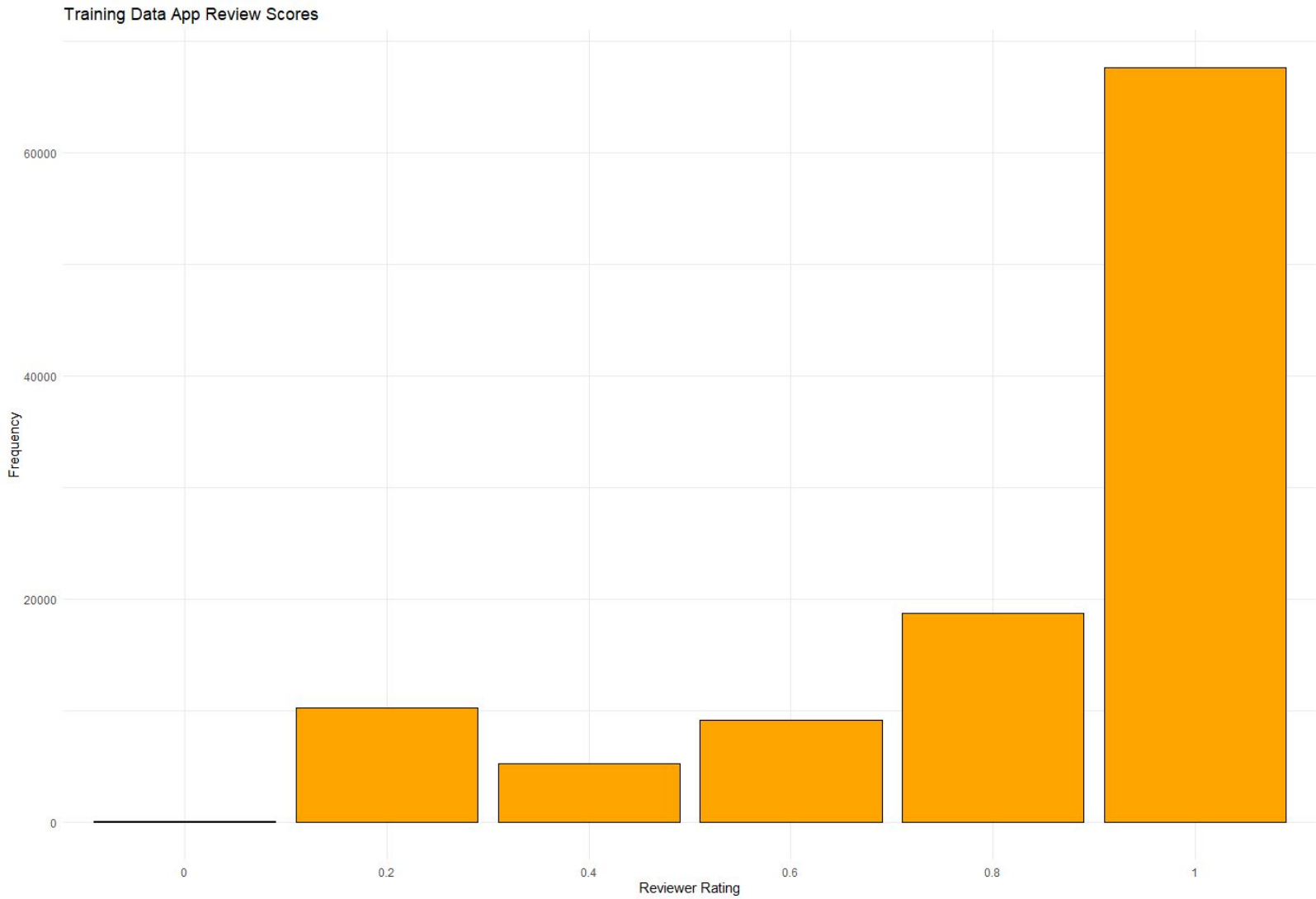
Sentiment Analysis - Full Dataset

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

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

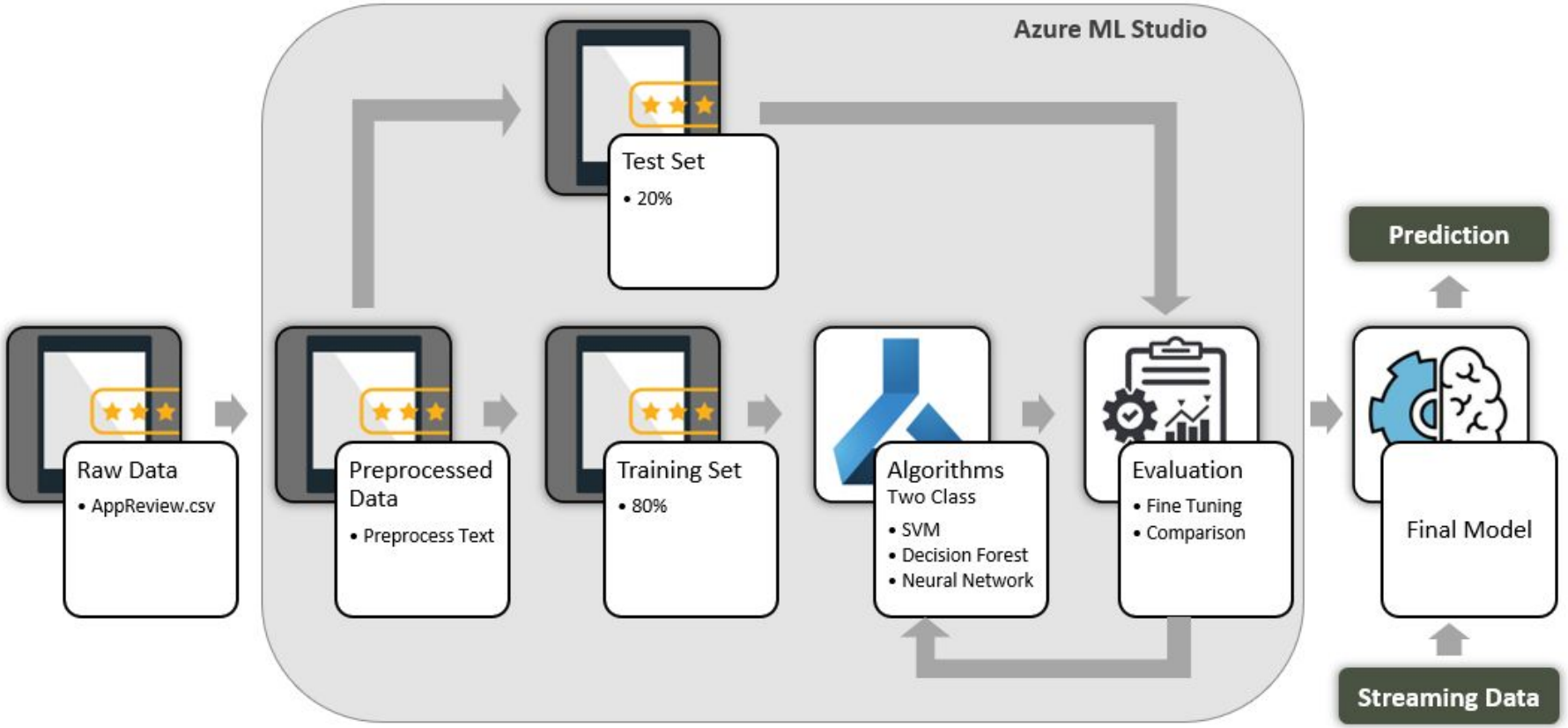
Matt

Score Distribution



Matt

Building a Predictive Model



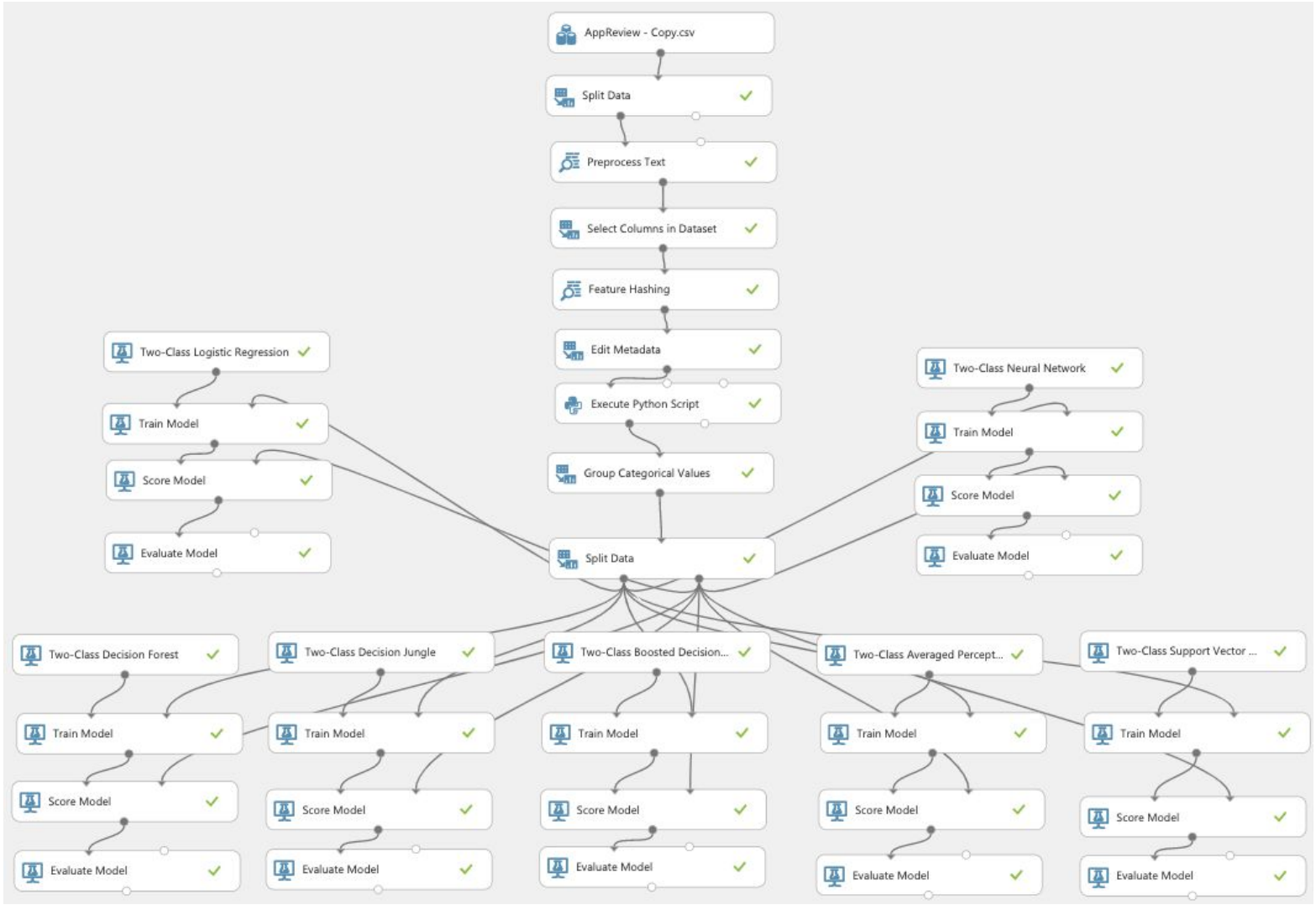
Text Preprocessing

- **Understand data**
- **Stop words removed (“a”, “is”, “it”, “the”, etc.)**
- **Lemmatization (“are” → “be”, “running” → “run”)**
- **Lower-case**
- **Remove numbers, special characters, email addresses**
- **Etc.**

General Model Development

- **Binary classification**
  - Positive/negative sentiment
  - Ratings: 0.0, 0.2, 0.4, 0.6, 0.8, 1.0
  - Cutoff +/- 0.7
- **Still heavy skew towards “Positive” sentiment**
- **50% train/test split**
- **Elected not to use validation set**

MS Azure ML Studio - Multiple Models Tested



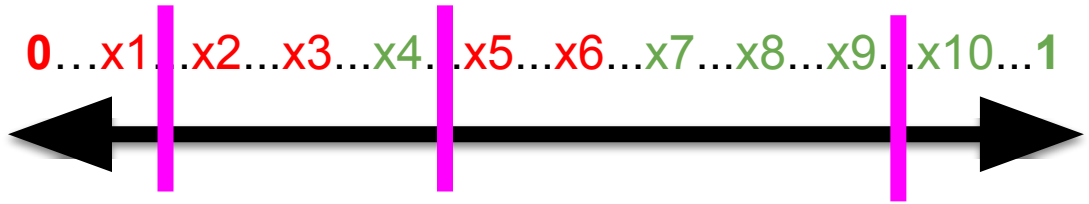
JM

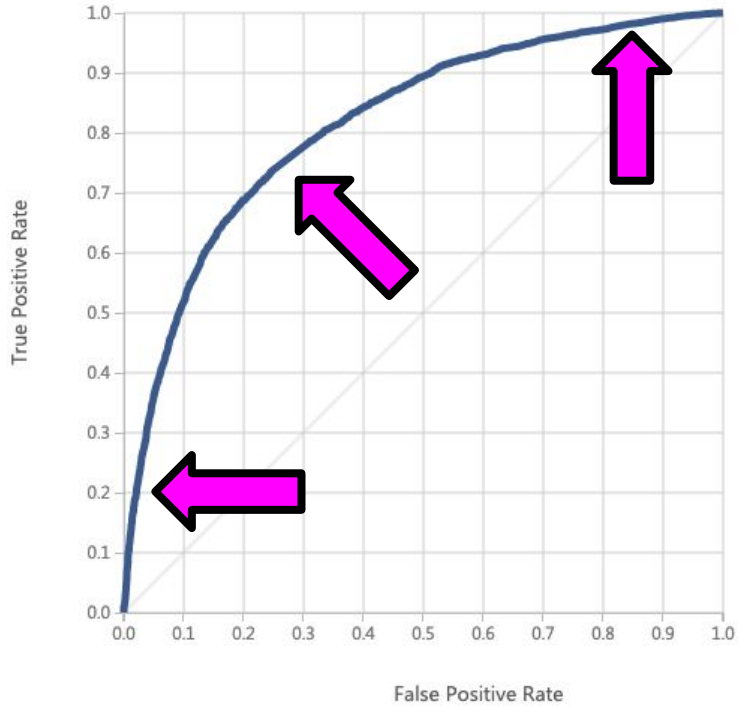
Binary Classification Assessment

- **More than just accuracy**
  - If 75% of data is positive, just guess “positive”
  - 75% accurate model! Hooray!
- **ROC Curve**
- **Precision/Recall Tradeoff**
- **F1-Score**

ROC Curve

- **Receiver Operating Characteristic**
  - True Positive Rate (TPR) = TP / (TP + FN)
  - False Positive Rate (FPR) = FP / (FP + TN)
- **Raise or lower threshold**
  - Lower: increase positive classification
  - Raise: decrease positive classification





Precision-Recall Tradeoff

- **Precision: TP / (TP + FP)**
  - Correct making prediction about data point
- **Recall (aka TPR): TP / (TP + FN)**
  - Correct compared to all in class
- **Do we want more precision, ensuring each prediction is correct?**
- **Or do we want more recall, capturing more of the correct data points?**

F-1 Score

$$F1 = 2 * \frac{(Precision * Recall)}{(Precision + Recall)}$$

- 0 to 1 score (harmonic mean)
- Balance between precision and recall

True Positive	False Negative	Accuracy	Precision	Threshold	AUC
2914	2499	0.752	0.756	0.5	0.807
False Positive	True Negative	Recall	F1 Score		
941	7539	0.538	0.629		

JM

References for Model Development slides

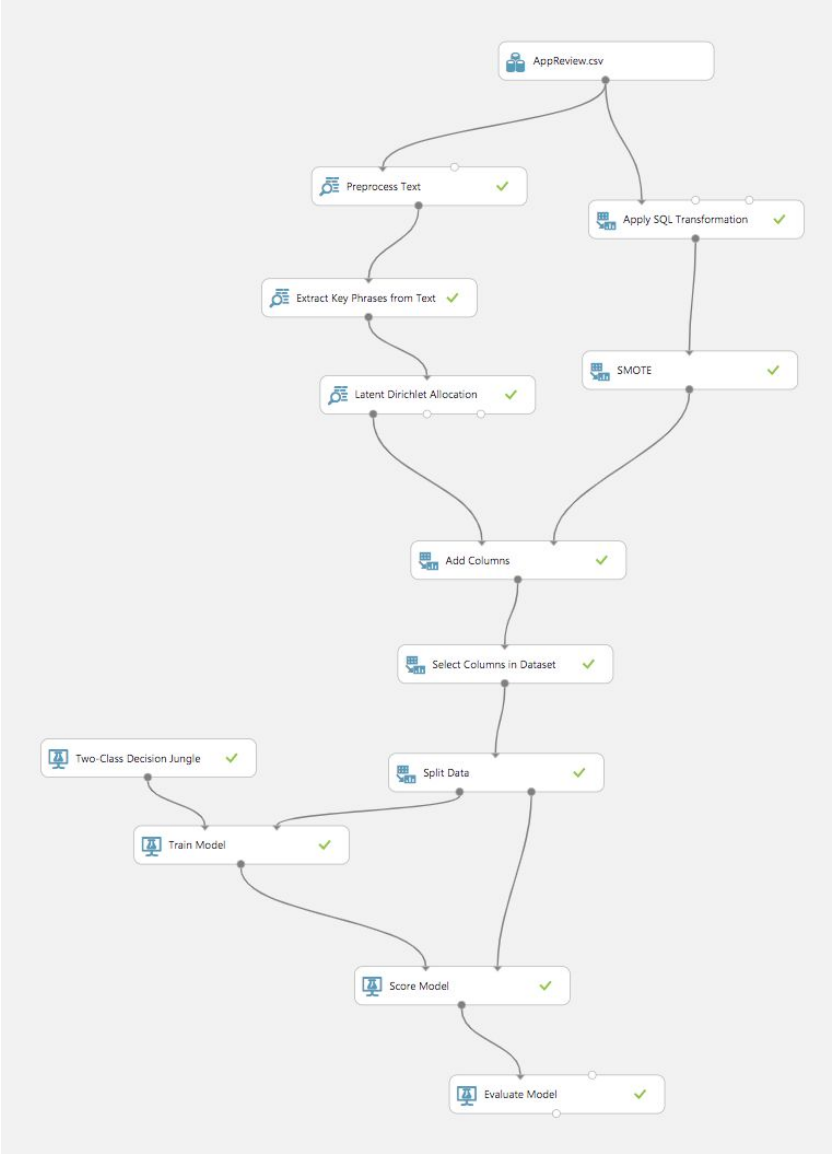
- **Machine Learning Mastery**
  - machinelearningmastery.com
- **Google Developers ML Crash Course**
  - developers.google.com/machine-learning/crash-course

Sprint C: Construction

Final Model

Final model modules

- SQL Transformation
- Preprocess Text
- Extracting Key Phrases From Text
- Latent Dirichlet Allocation
- Two-class Decision Jungle
- Breakthrough: SMOTE



Heather

Decision Jungle

high performance extension of the decision forest algorithm

- lower memory footprint and better generalization performance than a decision tree
- integrated feature selection and classification, resilient in the presence of noisy features [1]
- outperformed other binary classifiers in a test by the Azure team [2], and worked well on our data in combination with the key phrases and Latent Dirichlet Allocation modules

1. descriptions from: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-decision-jungle
2. https://gallery.azure.ai/Experiment/b2bfde196e604c0aa2f7cba916fc45c8

Heather

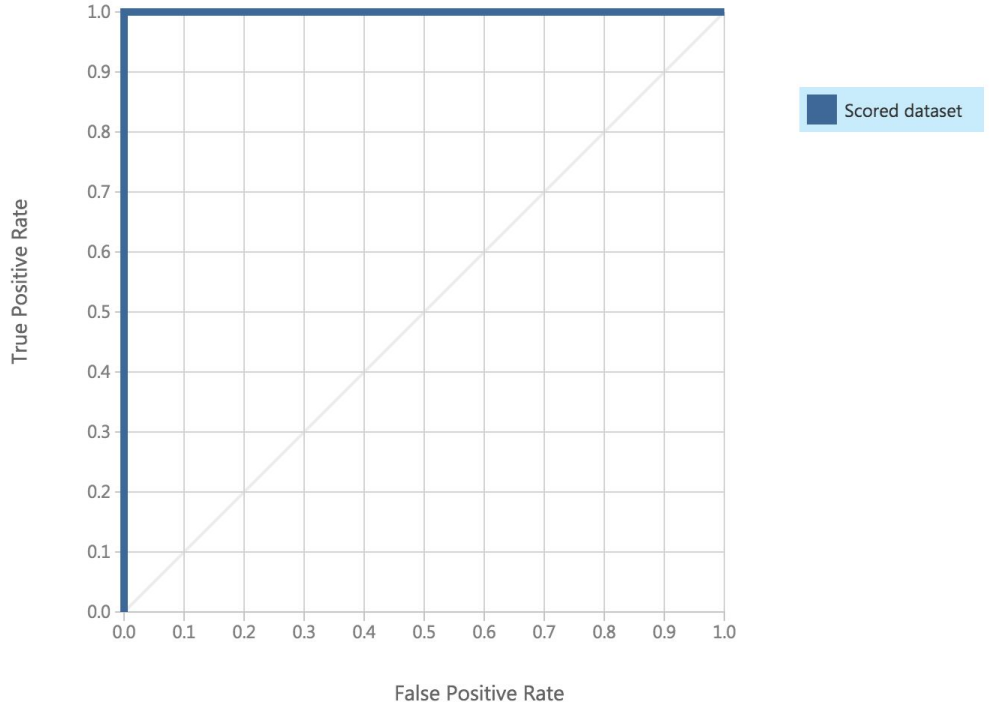
SMOTE

- **Despite skewed cutoff, still led to unbalanced classes**
- **Synthetic Minority Over-sampling Technique**
  - First described 2011: Chawla, Bowyer, Hall, Kegelmeyer
  - Under-represented classes do not present enough training examples
- **Synthetically generates samples in minority class**
- **Massive improvement**

Heather

Final model metrics

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



True Positive	False Negative	Accuracy	Precision	Threshold	AUC
37254	5932	0.926	1.000	0.5	1.000

False Positive	True Negative	Recall	F1 Score
0	37156	0.863	0.926

Positive Label	Negative Label
1	0

Heather

Improvement over previous models

- original model was only around 60% accurate, not much better than guessing
- original model predicted only ratings of 1.0
- our previous team model iterations were between 70-77% accurate, lower metrics across the board
- previous team models also had far fewer negative guesses, some of them still guessing zero negatives because of the low number of negative examples

Heather

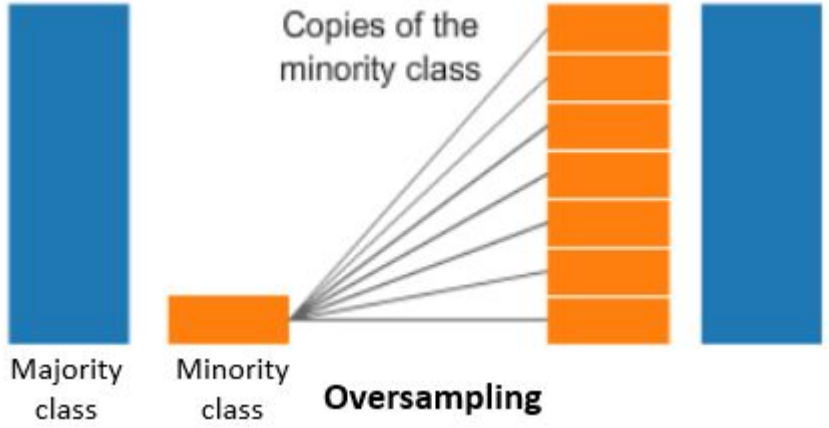
Sprint C: Construction
Supporting Research

Oversampling, LDA

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.[1]

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.[2]

- Oversampling to reduce class distribution, and improve confusion matrix in Amazon dataset
- LDA topic modelling and sentiment analysis on a comparable volume of tweets to explore hidden topics



Susan

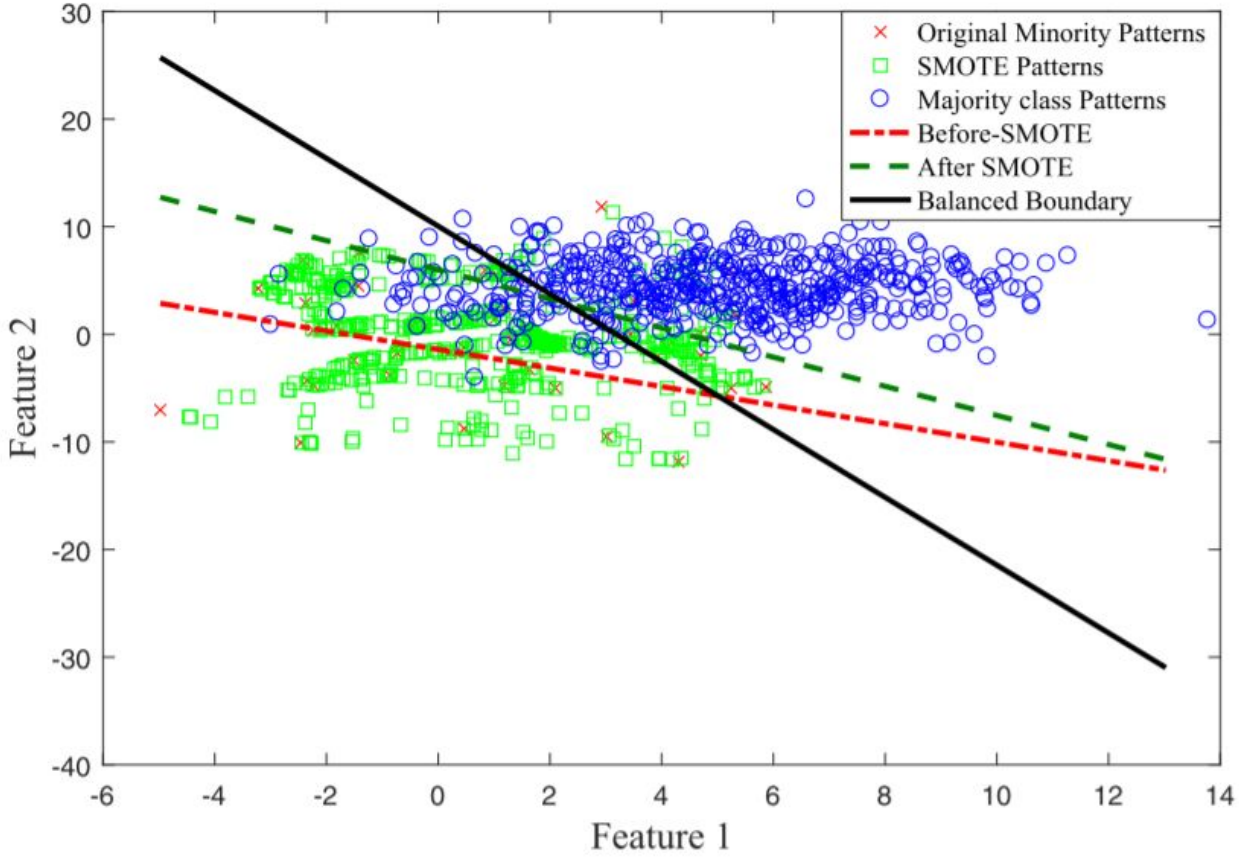
Using SMOTE to Improve Classification

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.[3]

- A comprehensive analysis of Synthetic Minority Over-sampling TEchnique (SMOTE)
- Analysis of the impact of SMOTE on classification performance
- Effect of different factors on the performance of SMOTE
- The superiority of over-sampling approach over under-sampling for the LDA classifier
- Comparison of oversampling methods

Using SMOTE to Improve Classification

- Leveraged to rebalance the positive and negative classes
- Accuracy has improved as the number of minority examples is higher
- Not effective for high dimensional data (curse of dimensionality), which is not a disadvantage in our case

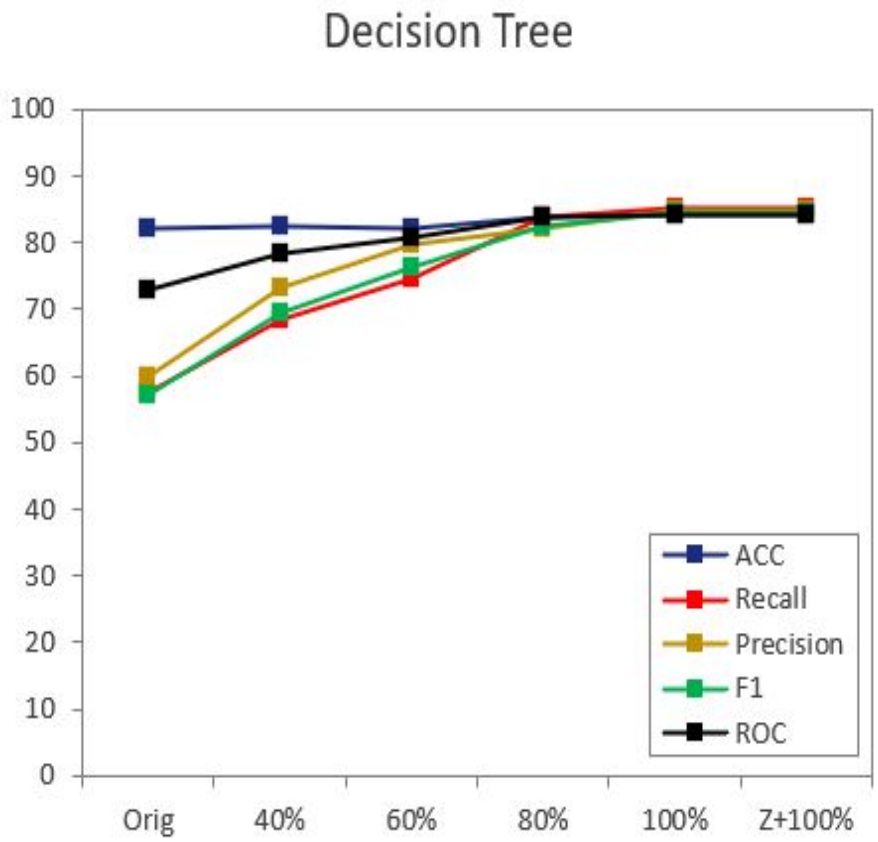


Susan

Performance Comparison

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.[4]

- Classification Algorithms compared: KNN, Decision Tree, Naive Bayes, Logistic Regression, SVM, ANN
- Evaluation metrics: Accuracy, Recall, Precision, F1-score, ROC
- Decision tree algorithm works with the highest performance
- SMOTE and normalization has significantly improved performance
- Performance has improved with the SMOTE resampling ratio size



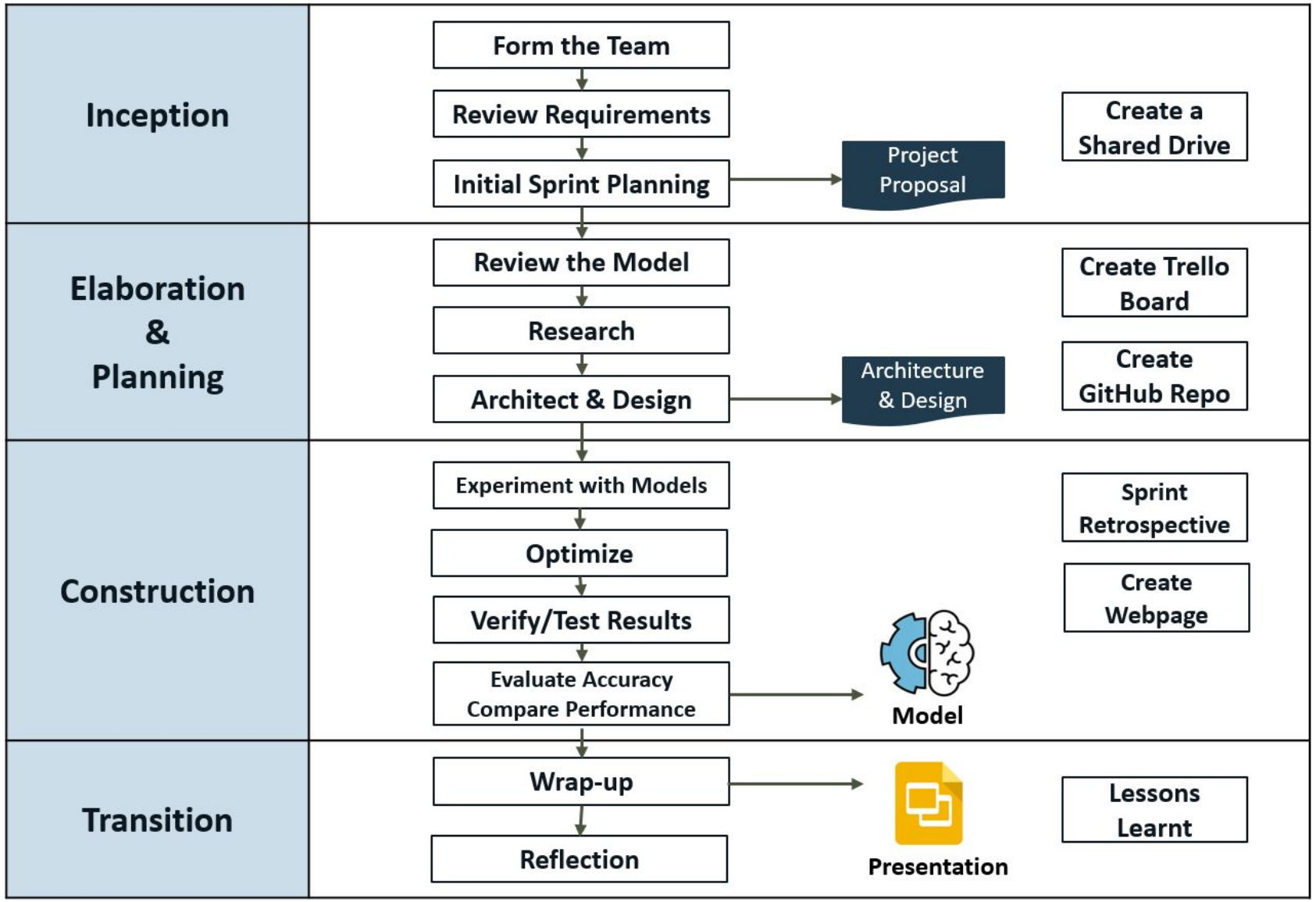
Susan

Sprint D: Transition

Final Reflections

Task Flow

- OpenUP process
- Trello to keep track
- Formal - Bi-weekly meetings
- Informal / ad-hoc - Slack Channel
- Google drive for collaboration



Susan

Tool Reflections

Tool	Pros	Cons
Azure ML Studio	• GUI minimizes need to learn/understand line-level code • Speeds up development and deployment • solution is readable by non-coders • Good integration with Azure toolsets • PaaS solution requires no platform support • pure web has no install requirements • Makes barrier to entry lower - so more people can try to get insights without needing as much training	• Whereas Python can be used on any cloud platform, Azure ML locks you into MS Azure only • Mixed solution if advanced code is necessary • Obfuscates code - makes detailed explanations of “why” harder • Makes barrier to entry lower - thus allowing/incenting people who don’t understand data science to perform DS tasks - and potentially not get right/best results • Tied to MS-only cloud
Other GUI ML tools (Knime, Alteryx, SPSS Modeler, etc.)	• GUI minimizes need to learn/understand line-level code • Speeds up development and deployment • solution is readable by non-coders • PaaS solutions (where appl) require no platform support • Makes barrier to entry lower - so more people can try to get insights without needing as much training • Many are not cloud provider locked	• Some have same cloud provider lock as Azure ML • If advanced code is neccessary, still need to code, thereby creating a mixed solution • Obfuscates code - makes detailed explanations of “why” harder • Makes barrier to entry lower - thus allowing/incenting people who don’t understand data science to perform DS tasks - and potentially not get right/best results
Pure code (python, R, etc.)	• most flexible • most advanced • robust community knowledge and market prescence	• requires not just DS knowledge, but programming knowledge • everything is code == nothing is AS fast to develop • only coders can understand the tech solution
Auto ML/AI tools (i.e. AWS Lex, IBM Watson Auto AI, Google Auto ML, …)	• no coding, no development whatsoever • all answers are automated • …”easy!”…	• least ability to offer input to answers • least ability to see underlying code • least trust by seasoned data scientists

Alex

Model Reflections

- How to process text
- Binary vs. multi-class classification
- 2-Class Decision Jungle
- SMOTE
- Discussion about which metrics to use and why
- Easy to iterate in ML Azure Studio