DEVELOPMENT TECHNIQUES
TEAM ASSIGNMENT

SEPTEMBER 2019

Master of Management in Artificial Intelligence 2020 , Smith School of Business

TEAM OSGOODE

Sneha Abraham	20197020	**Jerry (He) Liu**	20190907
Alexander Banh	06349542	**Darryl Lobo**	20192859
Yao Chen	20189695	**Blair Nicolle**	20194169
Mahesh Kumar	20193608	**Nayef Abou Tayoun**	20141011

EXECUTIVE SUMMARY

Establish a Centre of Excellence ("COE") centralizing specialized advanced analytics and related technical skills while serving other department's analytics requests.

The COE will adopt Microsoft Azure Machine Learning Studio ("AzureML") so that both advanced analytics and devops teams can operate using a common, well-integrated platform. AzureML has an easy-to-understand user interface and a very deep and rich feature set.

AzureML is part of the Azure ecosystem: its software is highly-integrated with Microsoft's enterprise solutions (e.g. Office, Visual Studio) *and* open-source systems *including* Tensorflow, R, and Python. Furthermore, Microsoft Azure has many *cloud analogs* to their traditional software making cloud adoption easier for the enterprise (e.g. Active Directory to Azure AD)

Azure's pay-as-you-go PaaS can lead to better ROI calculations. While AzureML offers free and inexpensive evaluation software sandboxes as the COE and company's needs grow Azure's standard pay-as-you-go rates will apply: those marginal costs resulting from activities in AzureML can be measured directly with respect to values derived from that business initiative.

Reserve use of Tensorflow[1] to specialized tasks lead by senior development staff and those who are fully familiar with Tensorflow and Python libraries (e.g. Keras). Use it for technical examinations into available parameterizations of Deep Learning or Neural Networks algorithms or when a lower-latency, lower-overhead solution is required by a production application.

Further, we provide two **Additional Recommendations** that may help guide a company defining roles and selecting skilled staff. It can be found as *Appendix 1: Additional Recommendations*.

[1] Full disclosure: Google's open-sourced Tensorflow represents a set of useful machine learning libraries with a specialization towards deep learning applications. Tensorflow is only one component of Google's Cloud Platform (GCP) which has a similar feature set to Azure and Amazon's AWS, albeit with each juggernaut's cloud platform sporting differently-named tools.

RESULTS OF OUR EVALUATION: Comparison of AzureML vs Tensorflow

We were tasked to compare our experiences and the performance of Azure ML Performance vs Tensorflow. We used a dataset consisting of 800 Keurig coffee machine reviews posted on Amazon. We assigned each review with a rating from "1: very unhappy" to "5: very happy".

We then ran a few **supervised learning algorithms** in several ways and with two different software tools so as to compare within a tool (MC classifiers) as well across tools (neural networks). We examined both confusion matrixes, performance tables (focusing on accuracy, though precision and recall were also available to us), and accuracy-versus-epoch graphs.

Table 1: Plan fo Performance Comparisons by Algorithm and by Vendor Product

Supervised Learning ML Algorithm	Azure ML	Tensorflow
Multi-class Decision Tree	**Yes** (Ngrams/TF vs Unigrams/TF-IDF)	--
Multi-class Logistic Regression	**Yes** (N-grams/TF vs Unigrams/TF-IDF)	--
Neural Network (several parameterizations)	**Yes**	**Yes** (Word Length vs #/Epochs)

Our individual and team results are provided, as requested, in *Appendixes 3 and 4.* There was some variability in our individual observations due to our codings -- but generally we observed:

Table 2: Experimental Observations on Predictive Performance

Topic	Observations
NLC had generally low accuracy	A relatively low accuracy in test set which we attributed to the relatively low number of records (both 100 and 800) relative to the average number of words in a review and the size of diversity of the words list.
We could've added to stop words list.	Common words such as "coffe[e]" or "cup" may've impeded the accuracy. See word clouds, below, comparing "V. Unhappy (1)" vs. "V. happy (5)."
Useful CM matrix interpretation	The confusion matrixes indicated algorithms *did* tend towards the correct *end* of the CM (predicting "negative" but not the *magnitude of negativity*).
AzureML vs TF	Overall accuracy for AzureML was better than that for Tensorflow.
AzML vs TF (NN)	The Neural network (NN) in Azure higher accuracy than Tensorflow's NN.
NN:Epochs;MaxW. NN vs D'Trees.	Increasing NN Epochs improved accuracy while reducing *max_words* reduced acc'cy. NN was more accurate than D'Trees for n-gram & unigram
Model Overfitting.	Some individuals observed that Tensorflow's NN overfit the data: “training accuracy” reached almost 100% for Darryl and Yao while "test accuracy" remained below ~40%. Model predicted only what we told it to remember.

Illustration 1: Word Clouds for Unhappy ("1") and Happy ("5") based on 800 records.

Interpretation: Useful sentiments (in yellow) were lost amongst generic words

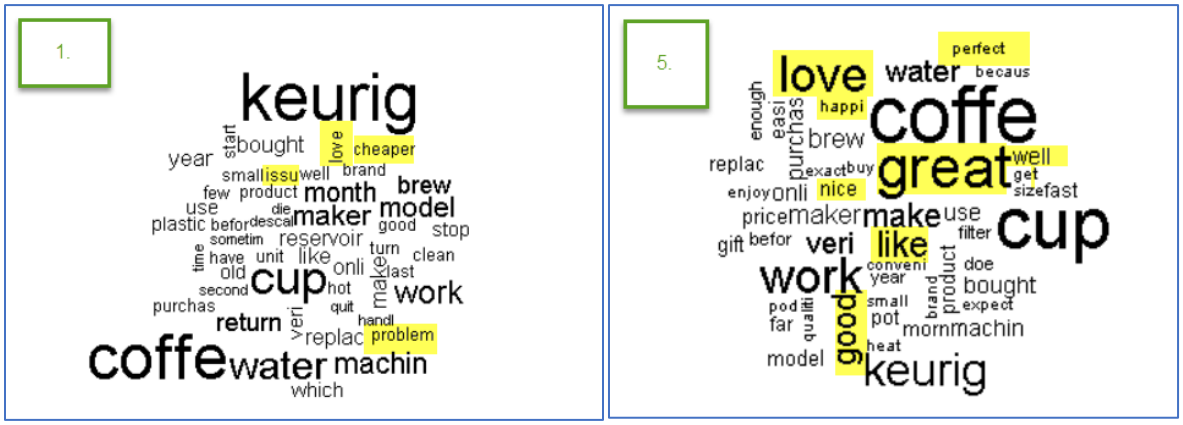


Illustration 2: Confusion Matrixes - samples

Interpretation: Relaxing accuracy analysis based on "Negativity vs Positivity" showed promise

Appendix 1: ADDITIONAL RECOMMENDATIONS

1. Develop a Business Needs Matrix

While performing its initial evaluation of AzureML, it is recommend that the **COE should survey internal business departments to form a business needs matrix** of ML-related business initiatives. It may contain and distinguish simple, low-risk projects and complex, high-risk projects (e.g. development of an advanced chat bot or reinforcement learning system). It can be used to help categorize work in the COE's pipeline as well as a handrail when hiring.

The COE can **select one or more low-risk pilot projects** to initially evaluate useability, costs, and technical aspects of any ML platform, such as AzureML. *Table 3*, below, lists some low-risk initiatives along with some suggested KPIs and typical underlying algorithms. In *Table 4*, common algorithm types are used to roughly compare AzureML and Tensorflow's strengths.

Table 3: A Business Needs Matrix with some lower-risk pilot projects shown

Business Need/Initiative	KPI's to Watch	ML Algorithm Type(s)
Product, Call Centre, or Service Sentiment Analysis	Revenue-related KPIs, or, Call-Center KPIs	Classification Algorithms; Neural Networks
Customer Chatbots (either pre-trained and/or custom NLP-trained data)	Customer Satisfaction and Call-center KPIs	Natural Language Processing; Neural Networks
Recommender Systems (for cross-selling services)	Sales Conversions	Clustering & Association
Customer and Market Segmentation Analysis	Revenue Growth; Customer Acquisition Cost %Δ's	Clustering & Association

Table 4: Performance Comparison of AzureML vs Tensorflow, by Algorithm Type

ML Algorithm Type(s)	Who performs 'best' in this area: AzureML, or Tensorflow, or "Both"?
Classification Algorithms	Both
Clustering & Association Algorithms	Both
Natural Language Processing: Custom Models	Both
Natural Language Processing:Pre-Trained Models	Azure (Pre-trained Models)
Image Classification (Convolutional NNs)	Both
Facial Recognition	Azure (Face API)
Video signal processing (Recurrent NNs)	Tensorflow
Speech Recognition (Recurrent NNs)	Tensorflow
Deep Learning (self-training computers)	Both

2. Productionization (Operationalization) Considerations

While a sufficiently-technical data scientist could *theoretically* be a "one-person" ML-based technology department, it is more likely that several roles will be needed within a COE. A product like Microsoft's AzureML contains a deep set of features which, significantly, can perform tasks assigned to different users who could all be using the same application. *Table 5* lists off roles that a COE could include with the latter roles seconded from the IT department.

Table 5: Some job roles that could perform tasks for the COE using AzureML

Role	**Responsibilities (in AzureML-Azure environment)**
Data Analyst/Researcher	Research, discover, acquire, and/or mine existing datasets
Data Engineer	Data Preparation; Data Transformations; Scripting
Data Scientist	Math/Statistical, Modelling, and Algorithmic Expertise
Analytics Manager	Controls access between Business and Analytics Team; Decides on highest value among many business requests.
System Architect and/or IT Development Manager	Sets IT standards related to Production-bound changes. E.g. may mandate any or all aspects like: Continuous Delivery (CD); or, *Docker* Containers; or, Kubernetes orchestration; or, microservices to be Azure-discoverable.
Full-Stack Developers and DevOps Team Members	Takes a tested algorithm or model and converts to a consumable, production-ready module that is integrateable with existing IT systems and aligned with IT standards.

Regardless of the software adopted, productionization aspects are critical to the success of the integration within the company's current IT ecosystem. In short, once a model is developed by the Data Analytics team, it is turned over to IT & DevOps for productionization using the Azure Machine Learning Services suite. This segues to system integration design topics.

For example, Azure also offers two services for exposed ML microservices: RRS (Response-Request Service) for low-latency, near "real time" services and BES (Batch Execution Service) for calls to model best handled via a batch process, perhaps due to the size of the dataset.

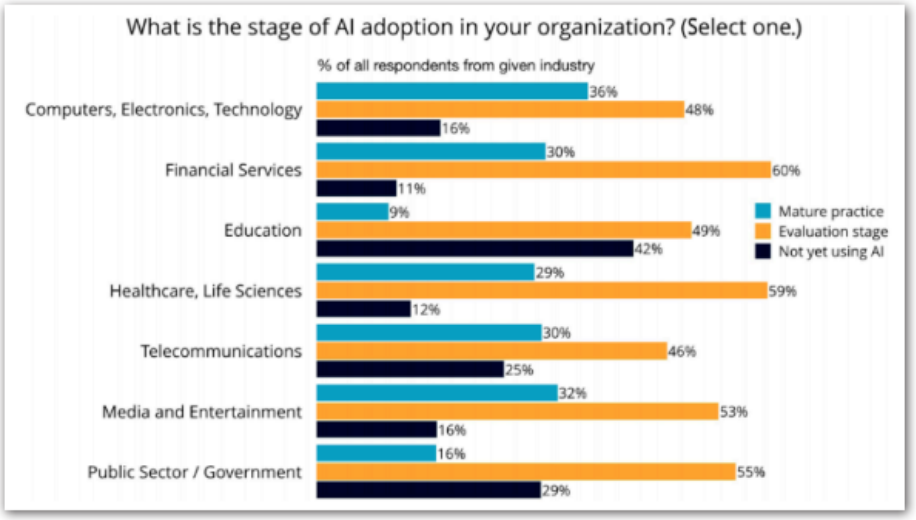
To over-simplify, design inputs may be based on **how formal an ML initiative must be:**

Repeatability/Formality	**Business Example**
One-time ML/Data Analysis	Measure upticks in customer activity after a campaign.
Reusable ML Analyses	Reporting activities using ML for monthly board meeting.
POC or Minimum Viable Product	ML-based POC built with potential future formalization.
Full-blown, architected ML idea	Exposing new ML microservices to user's web devices.

Appendix 2: Industry trends in AI/ML adoption, by Industry and by Vendor.

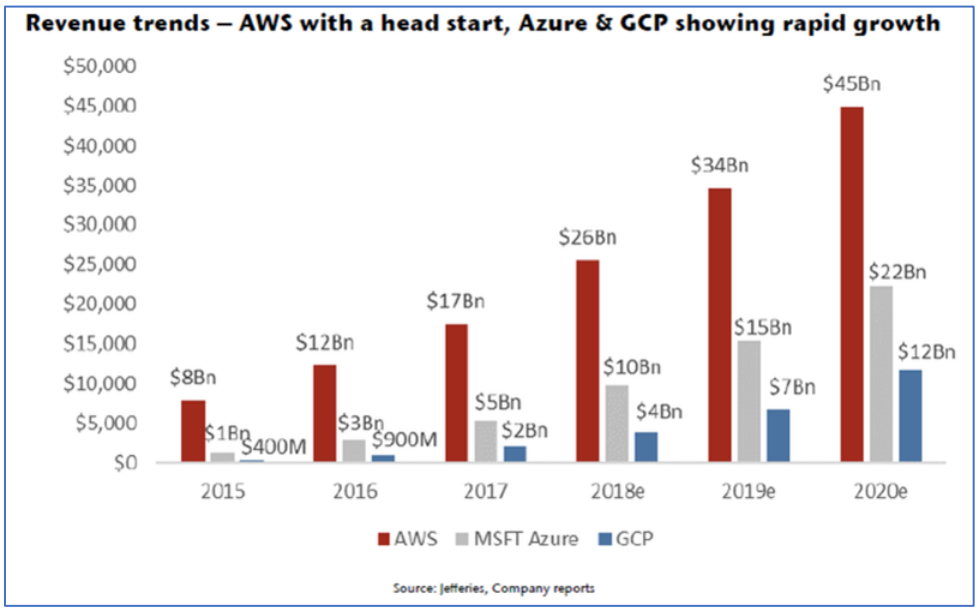
While researching AzureML and Tensorflow, we found interesting research that indicated that AI/ML adoption is not endemic and Azure is more popular than Google's Cloud Platform.

Snapshot 1.1: Most firms are just starting to evaluate AI/ML.



Source: (ZDNet)

Snapshot 1.2: Microsoft Azure leads Google Cloud Platform (GCP) but Amazon AWS is #1.



Source: (ZDNet and Jeffries Co.)

Appendix 3 - Sources and References

ZDNet and Jeffries Co. (n.d.). *ZDNet Top Cloud Providers 2018*. Retrieved from ZD Net: https://www.zdnet.com/article/top-cloud-providers-2018-how-aws-microsoft-google-ibm-oracle-alibaba-stack-up/

ZDNet. (n.d.). *https://www.zdnet.com/article/top-cloud-providers-2018-how-aws-microsoft-google-ibm-oracle-alibaba-stack-up/*.

Also, we researched the products on numerous websites and sub-pages:

"Microsoft Azure ML" and "Microsoft Azure" solutions

https://docs.microsoft.com/en-us/azure/machine-learning/

https://azure.microsoft.com/en-in/solutions/

"Azure Machine Learning: Analytics & The Power of Cloud Machine Learning"

https://www.youtube.com/watch?v=z-lsheCYtug#action=share

"Tensorfow Blog"

https://medium.com/tensorflow/

"Cloud Wars: Amazon AWS vs Microsoft Azure vs Google Cloud Platform"

https://www.youtube.com/watch?v=342KEaxFVjM

APPENDIX 3 - Individual Results (Summarized)

Team Member	Product	Algorithm	N-grams, TF Overall Accuracy	Unigrams,TF-IDF Overall Accuracy	Confusion Matrix Generated	Neural Network; Acc'y vs Epoch	Analysis & Commentary
Sneha	AzureML	MC Decision Forest	0.428571	0.428571	Yes		For multi-class decision forest, both n-grams and unigrams perform similarly
Sneha	AzureML	MC Neural Network	0.428571	0.357143	Yes		For Multi-class neural network, n grams performed better than unigrams
Sneha	AzureML	AzureML Summary	--	--	--		Summary: Multi-class Decision Forest and Multi-class neural network performed same for n grams Multi-class Decision Forest performed slightly better for unigrams than Multi-class neural network.
			MaxLength	**Epochs**	**CM Gen'rtd?**	**Test Set (Green) Accuracy**	
Sneha	Tensorflow	Neural Network	500	10	--	27%-50% & non-convergent	
Sneha	Tensorflow	Neural Network	1000	10	--	18%-42% & non-convergent	
Sneha	Tensorflow	Neural Network	1000	100	--	22%, convergent by 50 epochs	
Sneha	Tensorflow	Neural	500	100	--	20%-35%, non-	

		Network				convergent	
Sneha	Tensorflow	Tensorflow Summary					Increasing the number of epochs, improves the convergence of both training and testing set However, increasing the maximum length does not change the convergence of the training set However, for the training set we don't need to increase the epochs to more than 50 epochs
Mahesh	AzureML	MC Logistic Regression	0.462185	0.453782	Yes	--	
Mahesh	AzureML	MC Decision Forest	0.453782	0.411765	Yes	--	
Mahesh	AzureML	MC Neural Network	0.470588	0.436975	Yes	--	
Mahesh	AzureML	Azure Summary					No appreciable difference between algorithms (accuracy 41%-47%)
Nayef	AzureML	MC Decision Forest	0.071429	0.357143	Yes	--	
Nayef	AzureML	MC Neural Network	0.214286	0.357143	Yes	--	

Nayef	AzureML	Azure Summary					In both classifier methods (Multiclass neural network and decision forest) , Unigram shows better accuracy than Ngram. For N-gram, It seems that Multiclass neural network has a better accuracy than the decision forest. For n-gram, it didn't show any difference in accuracy using both classifier methods (Multiclass neural network and decision forest)
			MaxLength	**Epochs**	**CM Gen'rtd?**	**Test Set (Green) Accuracy**	
Nayef	Tensorflow	Neural Network	500	10	--	35%-17%, non-convergent	
Nayef	Tensorflow	Neural Network	1000	10	--	16%-33%, non-convergent	
Nayef	Tensorflow	Neural Network	1000	100	--	30%, convergent after 30 epochs	
Nayef	Tensorflow	Neural Network	500	100	--	30%, convergent after 50 epochs	
Nayef	Tensorflow	Tensorflow Summary					It can be observed that the convergence for both lines (red and green) gets better when the number of epochs increases from 10 to 100. From above plots, It can be concluded that the number of epochs can be reduced down to 50 instead of 100 and still get the convergence for both lines. Also , increasing the maximum length

							from 500 to 1000 doesn't improve the convergence of both lines.
			N-grams, TF	**Unigrams, TF-IDF**	**Confusion Matrix**		
			Overall Accuracy	**Overall Accuracy**	**Gen'rtd?**		
Jerry	AzureML	MC Decision Forest	0.2	0.2	Y		Bias towards predicting "Gibberish" category.
Jerry	AzureML	MC Neural Network	0.1	0	Y		Strong bias towards predicting "Gibberish."
Jerry	AzureML	Azure Summary					N-gram performed much better than Unigram. The interesting part is that under the multiclass neural network, it predicts all the comments to be Gibberish due to the limited dataset. Comparing to TensorFlow, TensorFlow has a better accuracy rate.
			MaxLength	**Epochs**	**CM Gen'rtd?**	**Test Set (Green) Accuracy**	
Jerry	Tensorflow		500	10	--	41%-78%, non-convergent	
Jerry	Tensorflow		500	100	--	53%-43%, semi-convergent	
Jerry	Tensorflow		1000	10	--	50%, convergent	

Jerry	Tensorflow		1000	100	--	44%-59%, semi-convergent	
Jerry	Tensorflow	Tensorflow Summary					Q: How does the performance differ between 10 vs 100 epochs? A: There is signification running time changing when I change the epochs number from 10 to 100 since the application needs to enhance the algorithm ten times more. Moreover, it's more stable when I change the epochs from 10 to 100. However, since our dataset is not big enough, we could get a higher accuracy even we used 70% on training side. The data point from the training set couldn't 100% cover all the features in the validation side.
Jerry	Tensorflow	Tensorflow Summary					Q: B) Between using 500 vs 1000 as the MAX_LENGTH? A: From the following images, the bigger Max_length has a better accuracy rate. The application has more chance to capture the special feature for each category in the training dataset.
			N-grams, TF	**Unigrams, TF-IDF**	**Confusion Matrix**		
			Overall Accuracy	**Overall Accuracy**	**Gen'rtd?**		

Blair	AzureML	MC Decision Forest	0.714286	0.714286	Y		N-grams had strong bias towards category "Neutral". Unigrams had bias towards "Unhappy" or "Neutral" (but never protected 'Happy') Note: I didnt use a 5-value label (class variable) but a 3-value label. I used 3 because I figured a review would either lead to: revenue loss (ie product return), no revenue loss (includes gibberish), or future revenue gain (ie buying more).
Blair	AzureML	MC Neural Network	0.714286	0.714286			Strong bias towards "Neutral". (It only predicted Neutral)
Blair	AzureML	Azure Summary					The metrics tables for the Multi-Class (MC) Decision Forest are confusing to interpret: The overall accuracy between both the n-gram and unigram sets were the EXACT SAME. And also were the EXACT SAME when compared with the MC Neural Network's metrics data. The sample size is small (n=100) so this may be as a direct result of that. However, the confusion matrixes are (slightly) different between the MC Decision Forest and MC Neural Network. Both are disappointing, showing that the algorithm slots everything into the "W" category (middle category of the three: Q = Loss of Revenue, W = No Loss of nor Promise of Future Revenue, E =

							Promise of Future Revenue) The filesizes are quite different between the N-gram (1.2MB) and Unigram (10KB) which is surprising: very curious.
			MaxLength	**Epochs**	**CM Gen'rtd?**	**Test Set (Green) Accuracy**	
Blair	Tensorflow	Neural Network	500	10	--	65%-50%, non-convergent	The Neural Network in Tensorflow performed better than the Neural Network in AzureML
			N-grams, TF	**Unigrams, TF-IDF**	**Confusion Matrix**		
			Overall Accuracy	**Overall Accuracy**	**Gen'rtd?**		
Yao	AzureML	MC Decision Forest	0.428571	0.428571	--		

Yao	AzureML	Azure Summary					N-gram vs Unigram: As can be seen from the result below, Unigram is a bit better than N-gram in this case. However, both of them were not very accurate, which due to the limited volume of the training dataset.
			MaxLength	**Epochs**	**CM Gen'rtd?**	**Neural Network; Acc vs Epoch**	
Yao	Tensorflow	Neural Network	500	10	--	32%, convergent	
Yao	Tensorflow	Neural Network	500	100	--	40%, convergent	
Yao	Tensorflow	Neural Network	1000	10	--	33%, convergent	
Yao	Tensorflow	Neural Network	1000	100	--	33%, convergent	
Yao	Tensorflow	Tensorflow Summary					Tensor Flow: I tried to evaluate the result by changing the Max_length between 500 and 1000, also Epoch between 10 and 100. Epoch = 100 is apparently a very big number to set for in this case, since it's even a bigger number than the training data (around 70). The performance will stop improving once the epoch increased to a certain threshold, which can also be seen from below 2) and 4). For Max_length, giving the dataset we have and the functions we used for this case, it's hard to tell which one is

							performing better. Giving the fact that below examples show either very fluctuate curves or very flat curves which both did not show improvement as the training going.
Yao	Tensorflow	AzureML vs Tensorflow					Conclusion: For further evaluation of the deep learning results, we definitely need way more data than this sample data. Otherwise, it's very hard to give suggestions on which platform to use or what kind of advantage each platform has. Or, the evaluation will not be accurate based on lack of input data.

Darryl	Azure	AzureML Summary					- As seen below the ngram vs unigram for the decision forest classier. The n grams did marginally better. - Overall accuracy was very bad because of low amounts of data - Ngram is when you calculate the probability of a word based on previous words. (gives the words some "context" - Ran quicker than NN - NN was more accurate in both N gram and unigram than the decision tree. Still bad overall accuracy. - Overall accuracy for azure better than overall accuracy for neural net in tensor - Tensor data got very over fit where "training accuracy" got over fit reaching almost 100%, but validation accuracy remained in 0.3 to 0.4 as opposed - Neural network in Azure did better than tensor in overall accuracy. - Aazure used 100 iterations/epochs and 100 hidden nodes one layer tensor flow used one layer with 32 nodes - More nodes means higher complexity and can result in better accuracy at the cost of possible over fitting
Darryl	Azure	AzureML	0.285714	0.214286	Y		N-grams biased towards (category 4): Satisfied; Unigrams biased towards both Very unsatisfied(1) and Satisfied(4). Runtime was 23 seconds.

Darryl	Azure	AzureML	0.5	0.428571	Y		N-grams biased towards category (4). Runtime 48 seconds
			MaxLength	**Epochs**	**CM Gen'rtd?**	**Test Set (Green) Accuracy**	
Darryl	Tensorflow	Tensorflow Summary					-performance did not seem to significantly change between max_length -explanation too little data -raising epochs means it converged to an optimal weight allocation on the neural nets, but did not need as many epochs as it flat lined around 20 epochs (guessed from x axis of the 100 epochs, would ideally run the notebook with 20 epochs and graph it out)
Darryl	Tensorflow	Neural Network	500	10	Yes	22%, convergent	
Darryl	Tensorflow	Neural Network	500	100	Yes	20%-30%, semi-convergent	Strong bias towards either categories (1) or (4).
Darryl	Tensorflow	Neural Network	1000	10	Yes	40%-28%, semi-convergent	
Darryl	Tensorflow	Neural Network	1000	100	Yes	38%, convergent	Strong bias towards either categories (1) or (4).

SELECTED** SAMPLE ILLUSTRATIONS

To save space; all information from individual results is captured in summary table above.

Illustration: Sneha's Accuracy vs Epoch Graph: MaxLength: 1000 and Epochs: 100

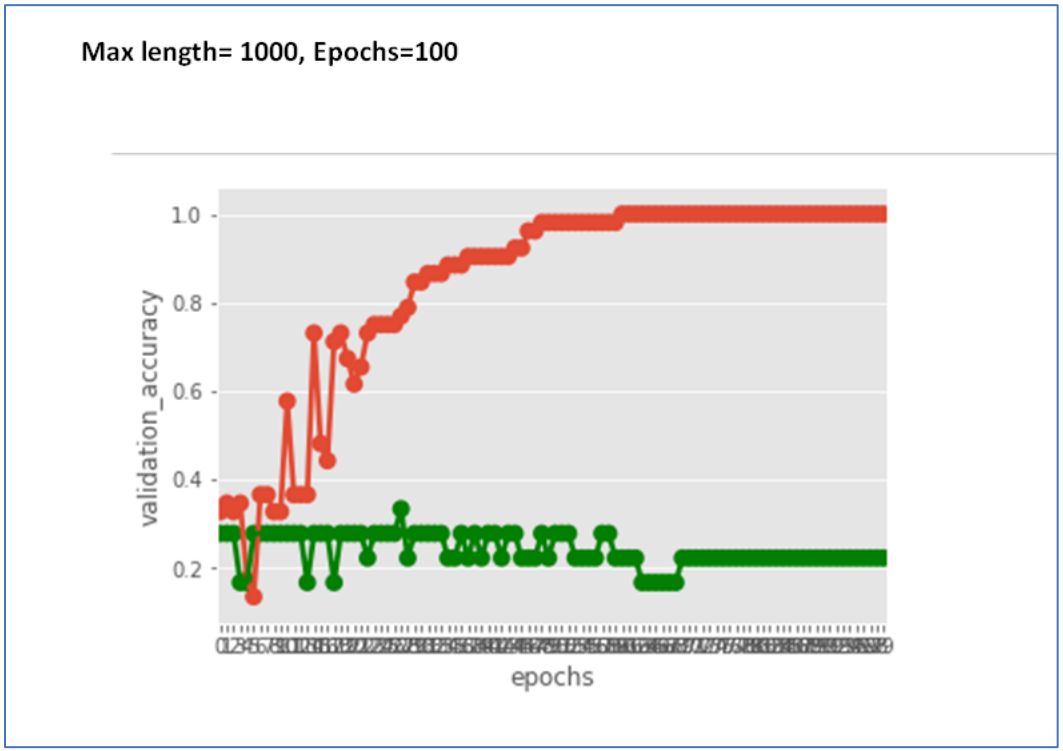


Illustration: Sneha's Accuracy vs Epoch Graph: MaxLength: 500 and Epochs: 100

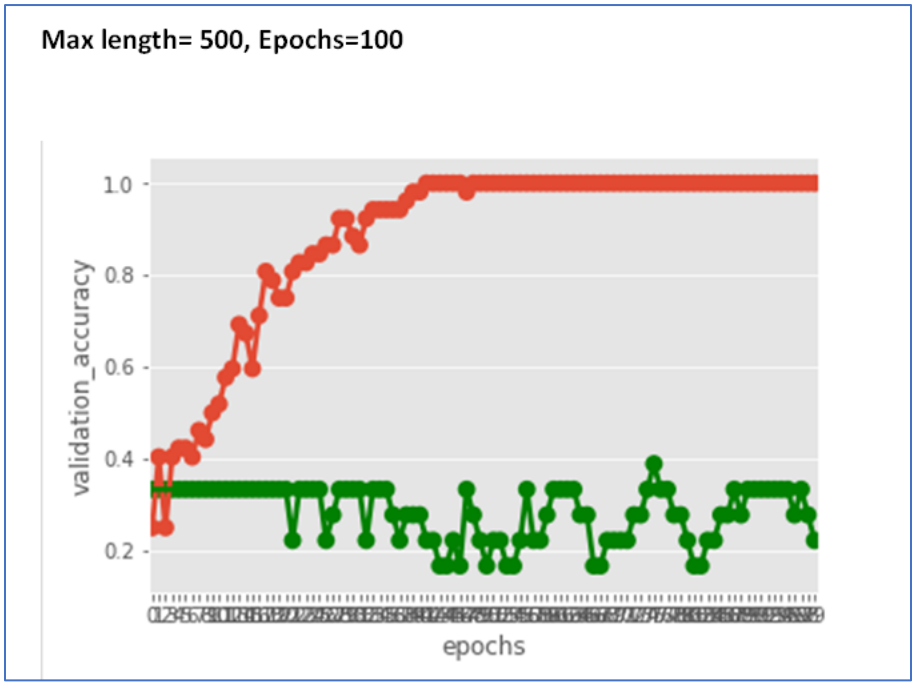


Illustration: Mahesh's Confusion Matrix and AzureML's Prediction Performance Tables: MC Decision Forest

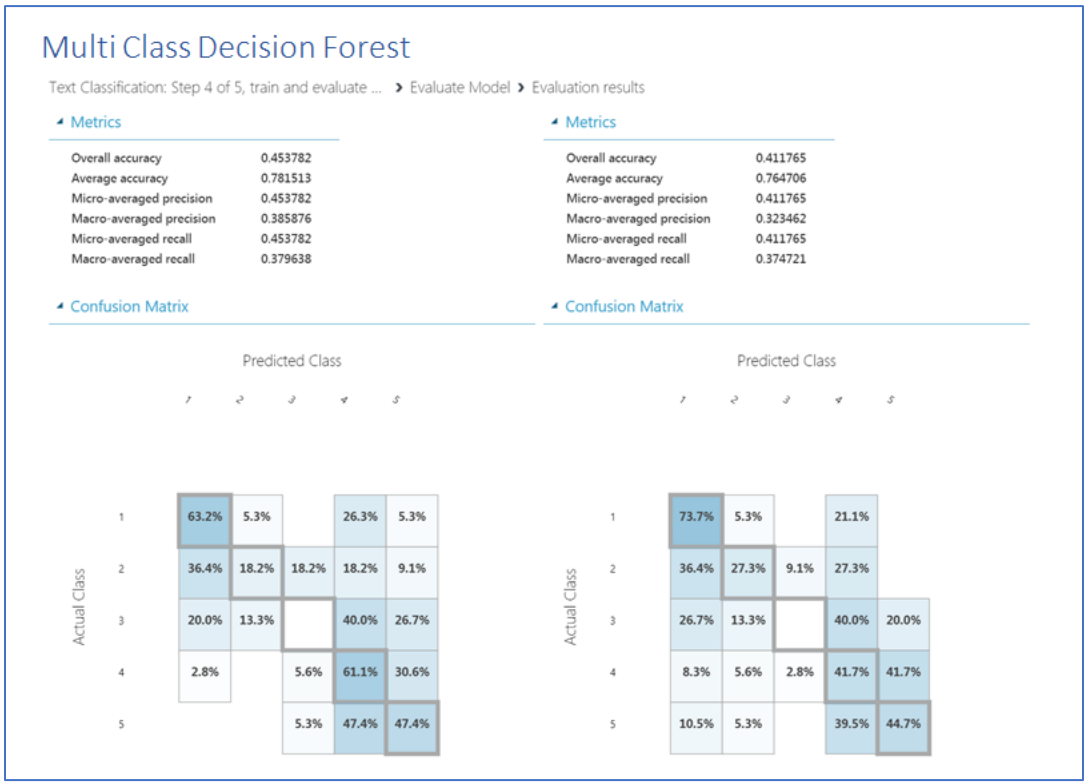
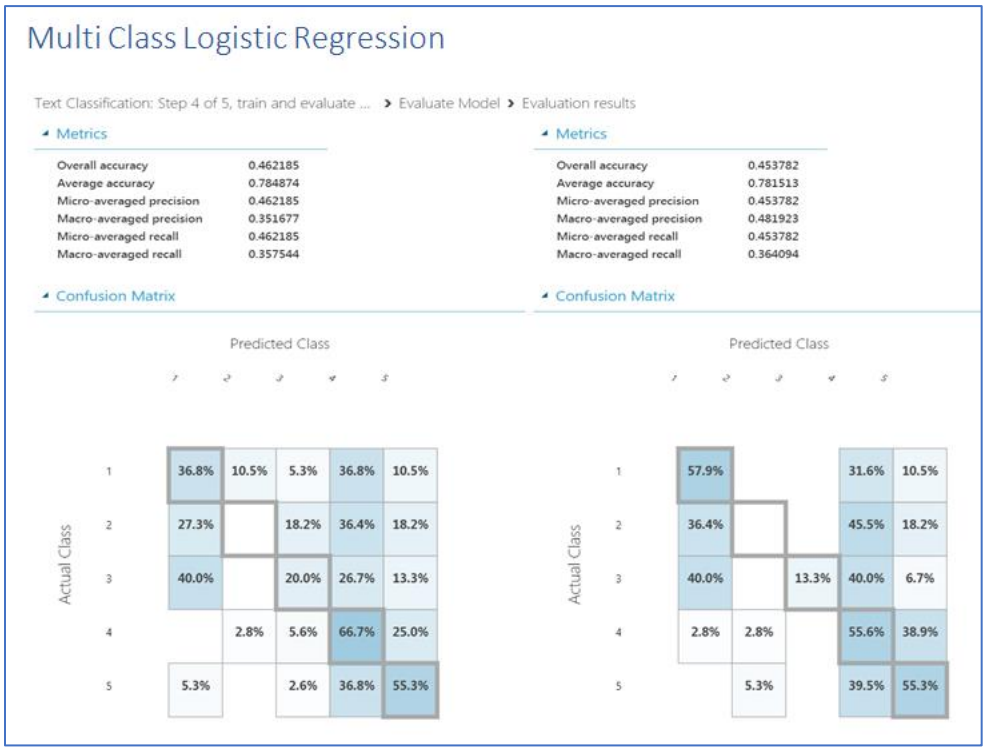


Illustration: Mahesh's Confusion Matrix and AzureML's Prediction Performance Tables: MC Logistic Regression



Appendix 4: Team (Consolidated Dataset) Results and Interpretation

Dataset: Keurig Coffee Machine Reviews (various, from Amazon website)

Number of records: 800.

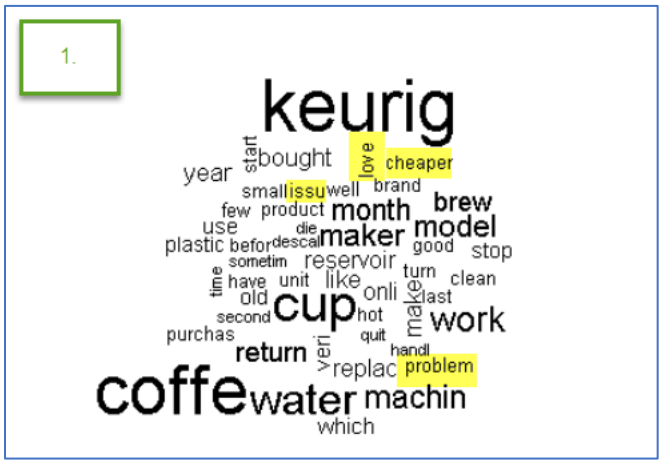
Rating Scale: 1 - Very unhappy, ... , 3 Neutral, ... , 5 Very happy.

Insights:

Word clouds from bag of words show that there isn't clearly defined correlation with corresponding sentiments with exception of some of the key words in label #5 which had "love" and "great" as keywords.

Also, words like "coffee", "cup", "water", and "Keurig" are seen dominantly in all clouds but don't provide meaningful insight towards sentiment. We could've considered adding these to our stop words list so that they would not confuse the algorithm.

Word clouds



3.


water
coffe
purchas
work
disappoint
becaus
cup
problem
reservoir
brand
quit
onli
keurig
model
machin
product
day
which
old
maker
make
befor
fill
die
filter
fewthrough
review
bought
horribl
these
year
month
veri
last
sever
return
money
plastic
use
amazon
buy
less
morn
brew
replac
like
tri
anoth
stop

4.


water
coffe
cup
use
model
great
far
replac
button
clean
which
maker
star
stop
onli
well
option
year
fill
reall
love
mug
size
need
fine
easi
work
unit
machin
befor
second
hot
brew
conveni
few
like
set
veri
brand
pod
month
buy
filter
purchas
tri
product
good
reservoir
keurig
make

5.


perfect
love
water
becaus
enough
happi
coffe
easi
purchas
brew
great
well
replac
exact
buy
get
enjoy
onli
nice
sizefast
price
maker
make
use
cup
gift
befor
veri
like
filter
work
conveni
doe
year
product
bought
pod
small
brand
expect
far
pot
mornmachin
good
heat
model
keurig

Insight:

In step 4.4 ("Tune the Hyper parameters") Blair opted to use the more familiar F-Score (rather than AUC) for measuring performance of the classification. Regardless, it did not improve the predictions of course. Aside: the F-Score punishes incorrect classifications and rewards correct classifications in seeking highest accuracy with a minimum of erroneous classifications.

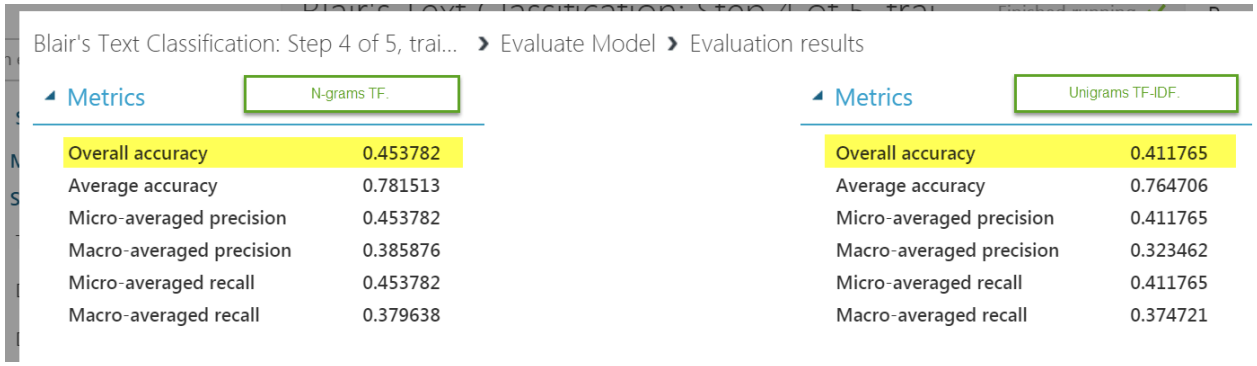
Results:

Using a Multi-class Decision Forest, first....

The two methods are roughly similar with only marginal differences in results. The overall accuracy is unsurprisingly tepid - not sterling, but also not terrible. Both algorithms couldn't quite distinguish 1 from 2 or 2 from 1 but it generally did a decent job at determining bad comments from good comments.

Middle-of-the-road comments were never correctly classified - the algorithm would tend to want to allocate them to either the "bad sentiment" or "positive sentiment" and not simply as an "ambiguous" or "neutral" comment, which is relatively reasonable from a human perspective as people could be most likely expected to post a comment in situations where they actually had something to say, one way or the other.

Illustration: Performance tables and Confusion Matrixes in AzureML, upon 800 records of data



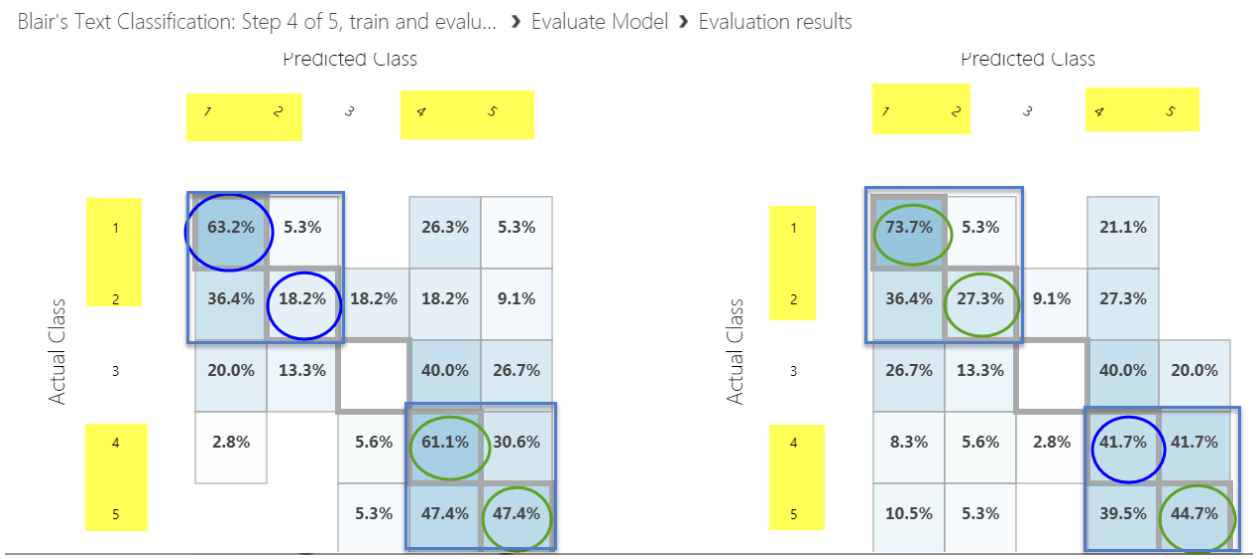
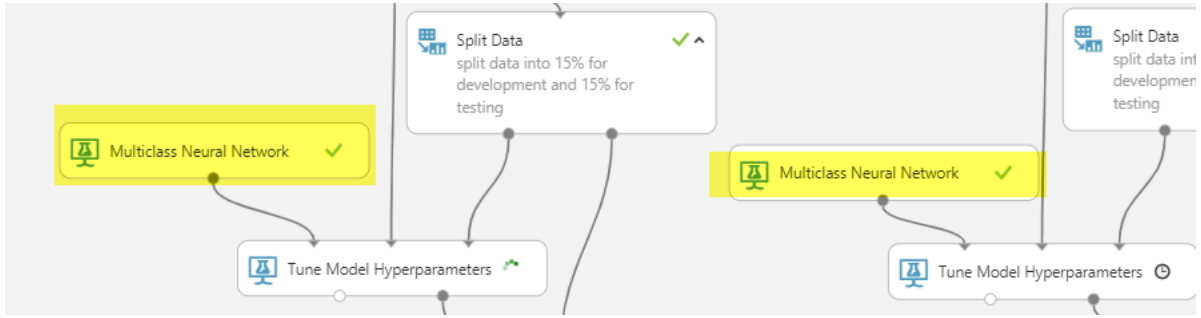


Illustration: Going back and trying things with a Multiclass Neural Network, rather than a Multiclass Decision Forest, in AzureML:



...Though the results of this change were substantially similar to the earlier results, which was not surprising.

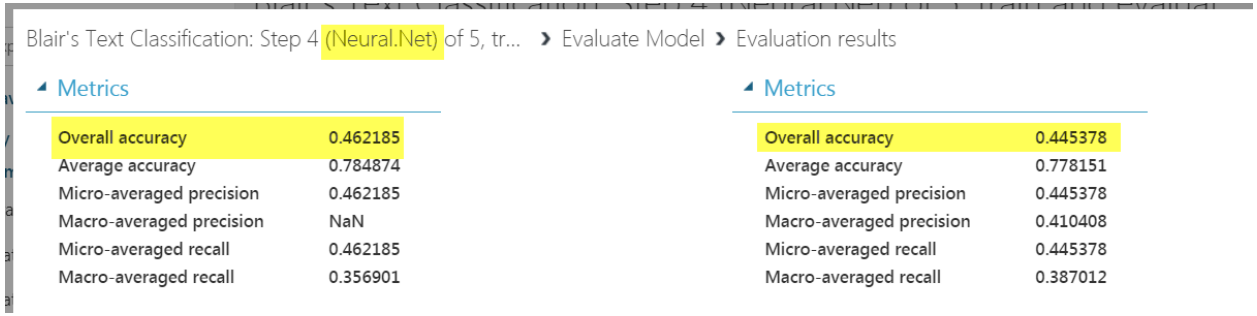


Illustration: The NN's CM shows *more* polarization towards very negative' sentiment for most kinds of negative and neutral reviews.