Syracuse University

Master of Science in Applied Data Science (ADS)

Portfolio Milestone- Project

Venkata Sharat Sripada
SUID# 436036419
vssripad@syr.edu

TABLE OF CONTENTS

1.0 Overview

I used to work at VMWare, Inc until recently, and a talk by Prof. Michael Jordan from Berkeley at one of VMware's R&D conferences in the Spring of 2019 inspired me to pursue the field of data-science and machine-learning. At the time I was applying machine-learning to niche problems in my role as a scale and performance Engineering Lead in the computer networking/virtualization technology, viz:

- sifting through hundreds of unit-tests, randomizing, and sequencing them in ways that would cover code paths eventually making software components resilient using
- training a Random-forest model on traffic patterns in a datacenter and aiding software-defined networks (SDNs) to enforce dropping anonymous traffic patterns
- using NLP and Apriori algorithms in large scale log aggregation endpoints to correlate events and auto-heal based on a remediation table lookup

However, in quest to gain deeper knowledge I had applied in the Fall of 2019 to the program of Applied Data-Science at Syracuse. More recently I transitioned to a new role at Amazon LLC as a System Development Engineer working on retail *search engine [1]*, where I intend to not just use knowledge garnered through this course but eventually cut into areas of research moving the needle of innovation, science, and Engineering.

1.1 Learning goals and objectives

While largely the goal was to explore where academia was meeting the industry, there was keen interest about the curriculum that led to the following goals:

- dealing with data-sets – how to collect or obtain data in the real-world, munge or cleanse it and stage it so models can be deployed and consumed in production environments
- learning to make predictions using the right machine-learning algorithms
- guidance by Professors at regular cadence with sync-sessions, evaluation of assignments and projects

- understand how data analysis or interpretation can be communicated back to business or stake holders via meaningful and succinct visualization and messaging

1.2 Course Structure

The course was split into two parts:

- Asynchronous module
    - Recorded videos presented by the Dean or Professor teaching the course
    - Questions pertaining to topics presented
    - Simulated lab exercises
    - Homework and Quizzes

- Synchronous module
    - 90-min live session delivered by the Professor
    - Breakout discussions
    - Online Exams

2.0 Program catalog – Outline

The Applied Data Science program comprised 11 courses (33 credits) in all and was organized as:

- Primary Core - 6 courses (18 credits)
- Secondary Core in *Language Analytics Track [2]* - 2 courses (6 credits)
- Elective Courses - 3 courses (9 credits)
- Exit requirement (1 credit)

Below are details of all completed courses (ranked chronologically):

Course	Description	Track	Professor	Term	Credits
MBC-638 - Data Anls & Decisn Making	Introduction to statistics to perform analysis and interpret results in a meaningful way. Understand value of data collection and analysis in acquiring knowledge and making decisions in today's business environment	Elective	Luz Flores Lee	Fall 2019	3
IST-687 - Introduction to Data Science	Explore key concepts related to data-science, statistics, information visualization and text mining using R	Primary Core	G. Krudys	Spring 2020	3
IST-659 - Data Admin Concepts & Db Mgmt	Data administration and concepts and skills – data analysis techniques, data modelling and schema design	Primary Core	Chad Harper	Spring 2020	3
IST-707 - Data Analytics	Focus on machine-learning model building and	Primary Core	Jeremy Bolton	Summer 2020	3

	optimization, real-world applications				
IST-652 - Scripting for Data Analysis	Skills of scripting (using Python) to solve problems of accessing and preparing data in a variety of formats – establishes skills essential to form data science pipelines	Elective	D. Landowski	Winter 2020	3
IST-718 - Big Data Analytics	Insights into logistic regression, decision trees and neural networks to make predictions – build Apache spark/python to build big data analytics pipelines	Primary Core	Jon Fox	Spring 2021	3
IST-664 - Natural Language Processing	Linguistic and computational aspect of natural language processing – used NLTK libraries in Python to solve various problems related to real-world applications of NLP	Secondary Core	Norma Polamino	Summer 2021	3
IST-736 - Text Mining	Advanced text mining algorithms for information extraction, text classification and clustering, opinion mining, and their applications in real-world problems	Secondary Core	Jeremy Bolton	Fall 2021	3
IST-719 - Information Visualization	Information visualization through the R programming language and Adobe illustrator - Data cleaning techniques, control of the R graphics environment, develop custom plots, visually explore data, use design concepts to visually communicate the story in the data, and discuss issues related to the ethics of data visualization	Elective	G. Krudys	Spring 2022	3
IST-772 - Quant Reasoning Data Science	Multiple strategies for inferential reasoning about quantitative data and methods for connecting data provenance to	Primary Core	Jason Anastasopoulos	Spring 2022	3

	substantive analytical conclusions				
SCM-651 - Business Analytics	Developing a portfolio of skills related to: Data collection – use tools like Google Analytics to collect/ organize data Data analysis: identify patterns in the data via visualization, statistical analysis, and data mining Strategy and decisions: develop alternative strategies based on the data Implementation: develop a plan of action to implement the business decisions	Primary Core	Don Harter	Summer 2022	3

3.0 Projects and key learnings from coursework

3.1 [IST-687] Introduction to Data Science

3.1.1 About the project/data

- **Title:** Analyze Airline Passenger data

3.1.2 Source code

https://github.com/sharatsv/MS-DataScience/tree/main/IST-687/Final-Project

3.1.3 Highlights

Analysis on data-set

- **Understanding variables:**
  - Satisfaction
  - Arrival & Departure Delay in minutes
  - Airline Names (to determine Airline popularity)
  - No. of other loyalty cards
  - Correlating Satisfaction, Popularity, Delays
- **Analysis from data-crunching:**
  - Delay (variables Arrival/Departure time)

  West Airways Inc. arrives on average 6.691395 minutes late

  FlyFast Airways Inc. arrives on average 42.83042 minutes late

  Southeast Airlines Co. arrives on average 19.20768 minutes late
  - Popular airliners (variable Airline Names)

  Cheapseats Airlines Inc. is the most used airline (20.06175 % of all passenger bookings)

  Cool&Young Airlines Inc. is the least used airline (0.9916159 % of all passenger bookings)

  Southeast Airlines Co. is at # 7 (7.373219 % of all passenger bookings)
  - Mean satisfaction (variable Satisfaction grouped per Airline)

  West Airways Inc. has highest average CSAT(3.486967)

  GoingNorth Airlines Inc. has lowest average CSAT(3.297194)

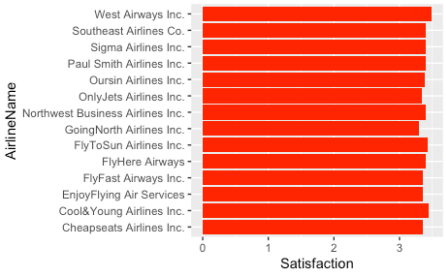
  Southeast Airlines Co. is at # 6 (average CSAT 3.396888)
- **Analysis from data-crunching (cont.):**
  - Applied Loyalty cards/discount coupons (variable No. of other Loyalty cards)

  Cheapseats Airlines Inc. has highest loyalty-cards offered/used (23209)

  Cool&Young Airlines Inc. has lowest loyalty-cards offered/used (1128)

  Southeast Airlines Co. is at # 7 (loyalty-cards offered/used 8439)

Descriptive Stats – popularity, satisfaction, loyalty program



Data Modeling Techniques and Predictive Analysis

Linear Regression

- Low R-square values indicates low correlation & we are therefore unable to reject the NULL hypotheses – 'No correlation exists between Airline Names and No. of other Loyalty Cards.
- Low R-square values/correlation co-efficient between several combination of variables
- predictive analysis using Linear regression, factoring in results from Parsimonius model:
- All variables shortlisted – *11.44%*
- Variables as a result of Parsimonius Model/Step function did not yield very useful prediction as well at *11.48%*

Support vector machines

- **KSVM:**

```
library(kernlab)
cutpoint2_3 <- floor((2 * length(randindex) /3))
trainData <- df_models[randindex[1:cutpoint2_3],]
testData <- df_models[randindex[(cutpoint2_3 + 1):length(randindex)],]

ksvmoutput <- ksvm(`Airline Name`~., data=trainData,
                   kernel="rbfdot", #kernel function that projects the low dimensional problem into higher dimensional space
                   kpar="automatic", #params used to control radial function kernel(rbfdot)
                   C=10, #C -> cost of constraints
                   cross=10, #use 10 fold cross-validation in this model
                   prob.model=TRUE)
```

Prediction accuracy - 11.74%

- **SVM:**

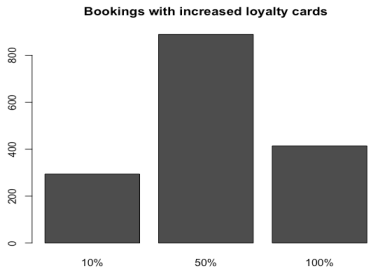
```
library(e1071)
svmoutput <- svm(`Airline Name`~., data=trainData,
                 kernel="linear", #kernel function that projects the low dimensional problem into higher dimensional space
                 cross=10, #use 10 fold cross-validation in this model
                 scale=FALSE)

svmpredict <- round(predict(svmoutput, test, type="response"))
svm_compTable <- data.frame(testData[,1], svmpredict)
colnames(svm_compTable) <- c('Test', 'Pred')

percentage_svm <- length(which(svm_compTable$Test ==
svm_compTable$Pred))/dim(svm_compTable)[1]
```

Prediction accuracy - 11.44%

Insights



Percentage increase in loyalty cards offered	Bookings per data set (in numbers)	Bookings per data set (in %)
10%	294	3.9%
50%	889	11.88%
100%	414	5.53%

As an experiment, 'No. of loyalty cards' variable was adjusted in the following manner for SouthEast Airlines Co.:

- If no cards were offered, offer one card at the least
- If cards were already offered then increase the number of loyalty cards by 10%, 50% and 100%

Conclusion

Clearly in my opinion variable 'No. of Loyalty cards' seems a mere distractor and cannot be used in any meaningful analysis. Also, that statement possibly holds some ground to the dataset in entirety. As a Data-Scientist or Business Analyst for Southeast Airlines Co. I would recommend getting more insightful data:

- Adding passenger first and last names can possibly help in determining how a passenger truly rated his/her experience when flying an airline provider and how that impacted flying again with the same airline in the future
- Specific satisfaction indices – related to delay and service

3.2 [IST-659] Data Admin Concepts & Db Mgmt

3.2.1 About the project/data

- Title: Server and network inventory management at scale

A production (aka prod.) environment typically comprises several servers and upstream switches (CLOS fabric comprising first-hop switches termed LEAF and a SPINE aggregate). Developers are building these prod. environments typically doing the following:

- Add servers to prod. to increase capacity
  (OR)
  Remove servers from prod. when servicing hardware faults or deprecating servers (due to end of life/expiry of service contracts)
- Configuring upstream switches to facilitate workloads or virtual machines (VM) on prod. servers to communicate with each other across the datacenter
- Setting up NFS mounts etc.
- Shuffling servers between prod. environments

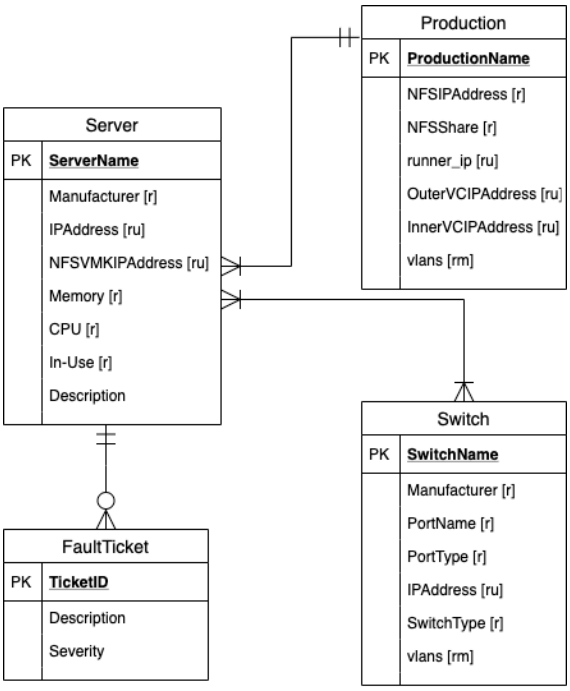
Clearly, this is a cumbersome process proving detrimental to Engineering productivity. The endeavor is to automate the above steps end-to-end offering simplistic workflows via a Graphical User interface (GUI) or Application Programmable Interface (API) and utilizing an RDBMS at the backend.

3.2.2 Source code

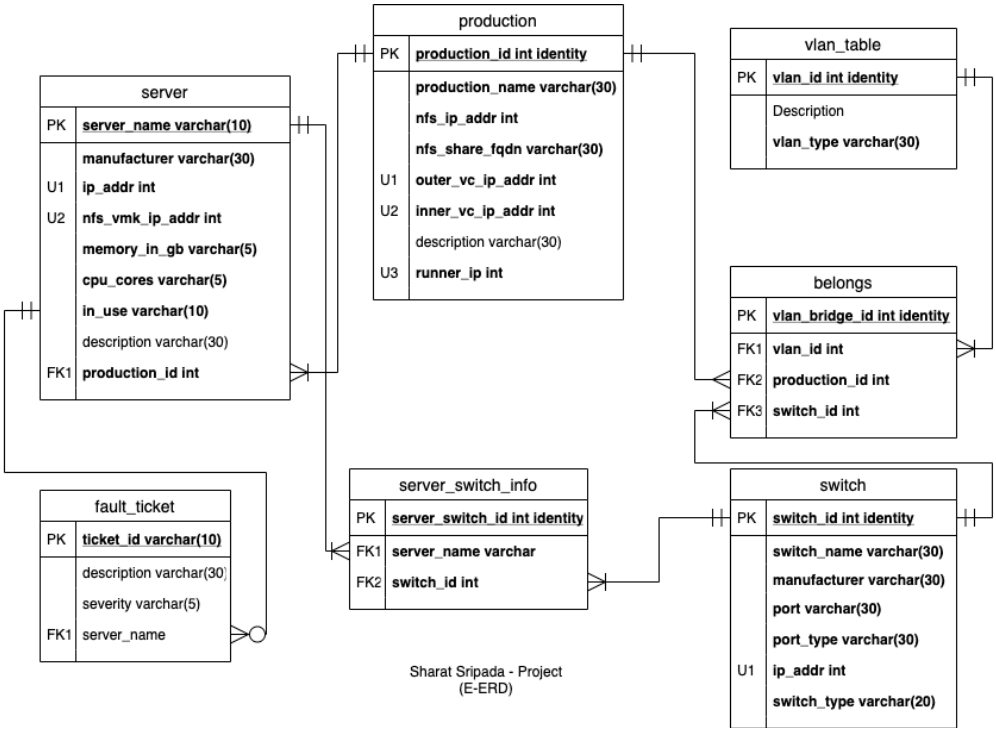
In Python-Flask for webserver and HTML/JavaScript for frontend/GUI:
https://github.com/sharatsv/MS-DataScience/tree/main/IST-659/Final-Project

3.2.3 Highlights

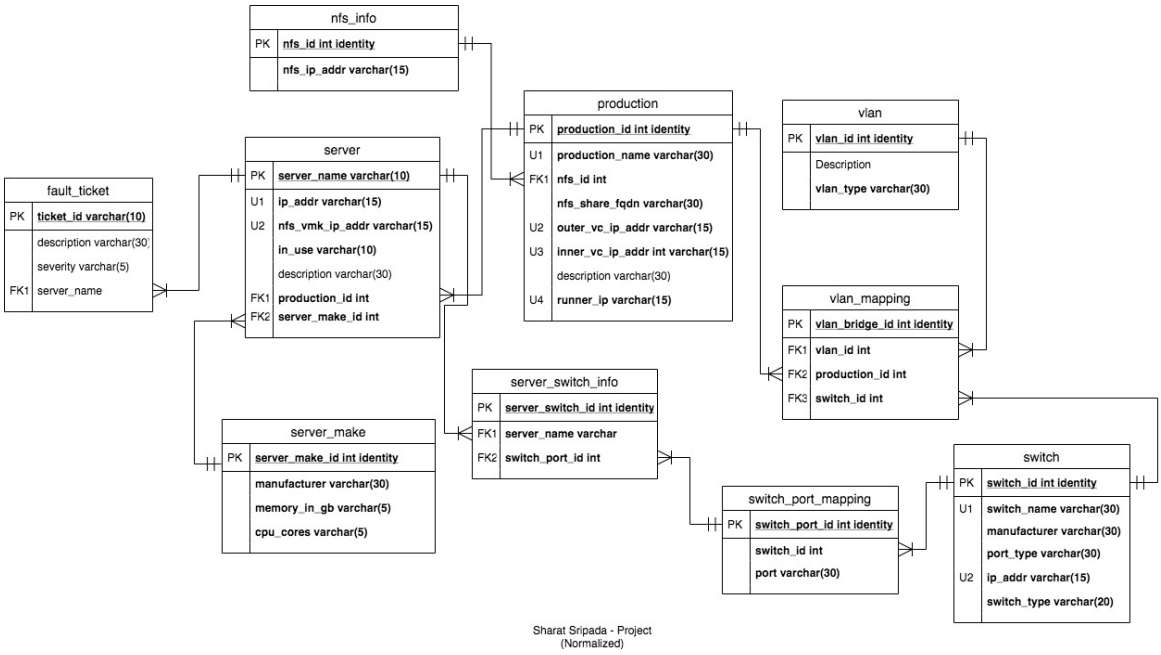
3.2.3.1 Entity Relation Diagram (ERD) and Normalization



Sharat Sripada - Project
(ERD)



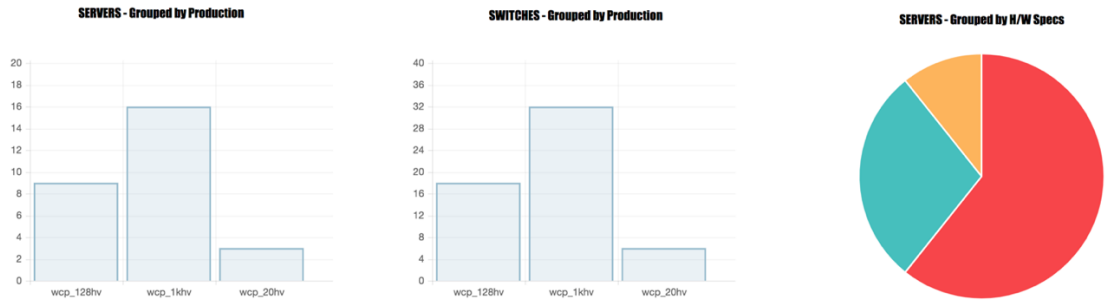
Sharat Sripada - Project
(E-ERD)



3.2.4 Conclusion/Summary

Achievements include:

- Conceptualization to realization using techniques imbibed via IST-659 in designing databases
    - All tables organically went from ERD through logical-modeling and normalization process before being 3NF compliant
- All tables have real data from THREE prod. environments which greatly establishes a workflow we have in place for consumption
- Setting in place a rich web application framework
- A fully functional dashboard showing resources consumed across environments:



3.3 [IST-707] Data Analytics

3.3.1 About the project/data

Google image retrieval using Landmarks data (Kaggle competition)

3.3.2 Source-code

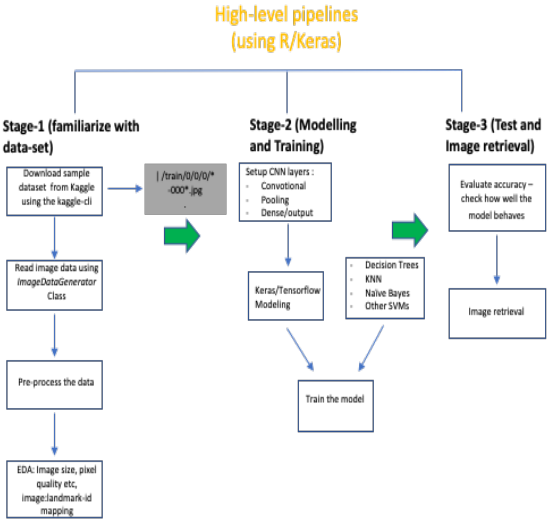
The code was entirely written in R using the Keras/Tensorflow libraries:
https://github.com/sharatsv/MS-DataScience/tree/main/IST-707/Final-Project

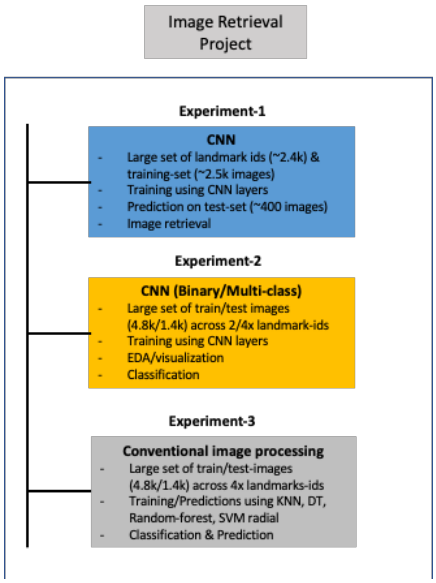
3.3.3 Highlights

3.3.3.1 Execution – Pipelines and experiments with CNNs/Traditional models

The landmark retrieval problem is a paradigm in itself. While there is a steep learning curve to image processing with understanding the architecture of Convolutional Neural Networks (CNN) there are deeper aspects to experimentation at scale. Given the problem complexity at the outset and the resource limitations (lack of compute resources or GPUs), the Team dissected the project into three parts:

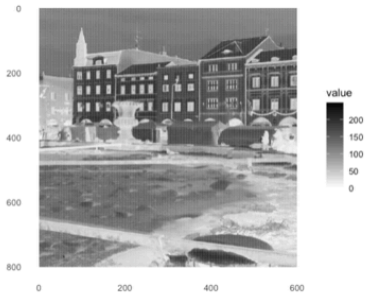
- Focusing on exploring CNNs, generators to sift through images at scale and experimentation with convnet parameters to improve prediction
- Deeper understanding of image processing at each layer and other aspects of CNNs while limiting the data-set classifications to binary or multi-class at most
- Exploring traditional classification and prediction algorithms for image processing

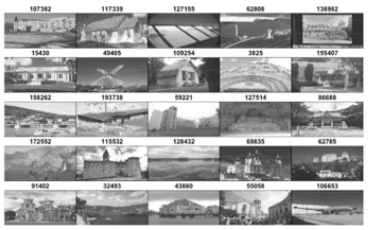




3.3.2.2 Experiments and Results

Experiment 1: Classification/Retrieval Using CNN





Stage-1: Familiarize with Data Set/EDA

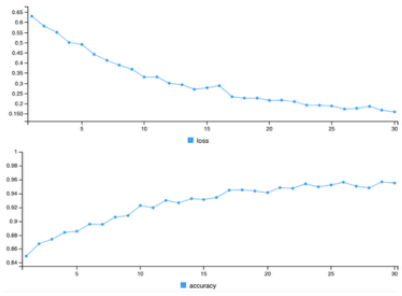
- Details: 600 x 800 images with pixels ranging between 0-255
- Inspection of images to see quality of images that will be fed to the layers within the Convolutional Neural Net
  - Based on the image classifiers (landmark-ids), images are tagged with corresponding landmark-ids

Stage-2: Modeling and Training (with Convolutional Neural Networks)

- The neural network was built using keras_model_sequential().
- The basic building block of the neural network is the Layer:
- Layer-1 & Layer-2: Convolution Layer
  - Activation Layer - Rectified Linear Units (ReLU)
- Layer-3: Pooling Layer
  - Activation Layer - Rectified Linear Units (ReLU)
- Layer-4: Dense Layer (Output)
  - Activation Layer - Softmax

IST707 Final Project: Image Classification Using Google Landmarks Data

Experiment 1: Classification/Retrieval Using CNN



Stage-3: Training results (loss vs accuracy)

> hist$metrics

$loss

[1] 0.6295260 0.5806534 0.5500867 0.5000029 0.4906446 0.4415369 0.4122235 0.3888409 0.3685994 0.3300786 0.3307345 0.2996668 0.2922836 0.2693858

[15] 0.2770554 0.2874595 0.2329875 0.2265687 0.2265248 0.2153458 0.2163346 0.2093395 0.1918329 0.1910077 0.1879119 0.1724818 0.1762784 0.1855961

[29] 0.1672001 0.1588635

$accuracy

[1] 0.8496459 0.8673535 0.8737926 0.8837733 0.8853831 0.8956858 0.8953638 0.9059884 0.9082421 0.9227302 0.9195106 0.9301353 0.9265937 0.9327109

[15] 0.9314413 0.9343207 0.9449453 0.9452672 0.9436575 0.9414037 0.9484868 0.9475209 0.9539601 0.9497746 0.9523503 0.9565358 0.9507405 0.9481648

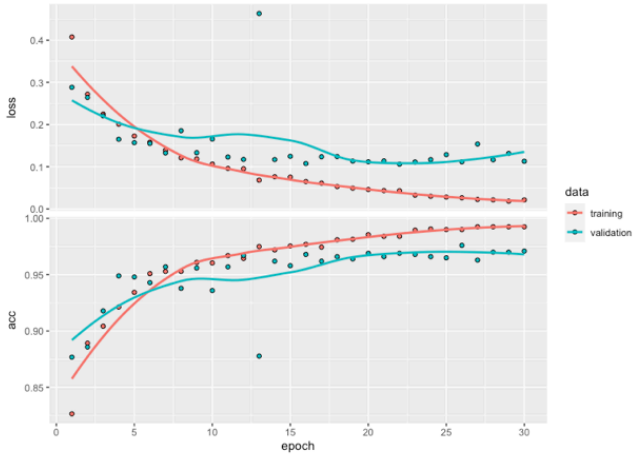
[29] 0.9568577 0.9552479

IST707 Final Project: Image Classification Using Google Landmarks Data

Experiment 2: Binary/Multi-Class Classifiers (CNN)

After rounds of trial and errors with hyparameter-tuning, three models produced good results:

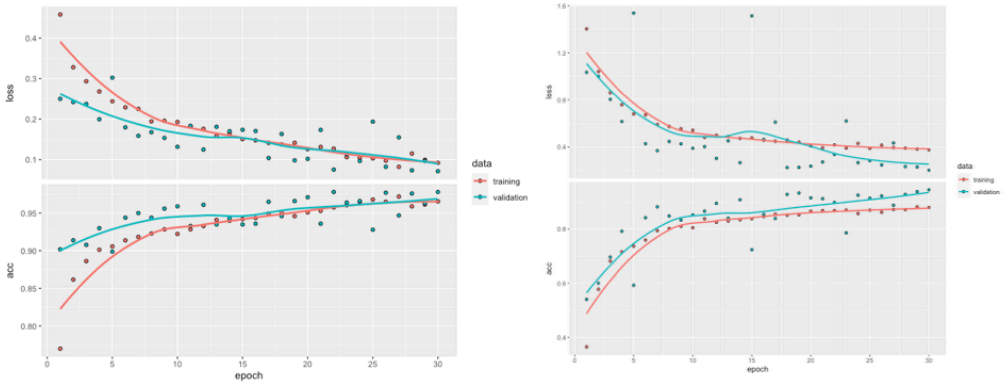
- Model 1: rmsprop optimizer, learning rate 0.0001



IST707 Final Project: Image Classification Using Google Landmarks Data

Experiment 2: Binary/Multi-Class Classifiers (CNN)

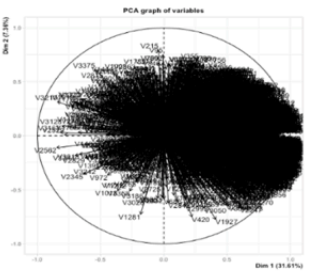
- Model 2: rmsprop optimizer, learning rate 0.0001, data augmentation, a dropout layer at rate 0.5
- Model 3: rmsprop optimizer, learning rate 0.01, data augmentation, a dropout layer at rate: 0.5



IST707 Final Project: Image Classification Using Google Landmarks Data

Experiment 3: Traditional Classifiers

Dimensionality Reduction - PCA




```
'data.frame':   3431 obs. of  31 variables:
 $ label: Factor w/ 4 levels "204097","83144",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ dim1 : num  0.498 0.405 0.439 -0.101 0.397 ...
 $ dim2 : num  0.5206 0.4451 0.4835 0.0515 -0.2893 ...
 $ dim3 : num  0.1175 0.0531 0.11 0.1251 0.0692 ...
 $ dim4 : num  -0.1703 -0.136 -0.3632 -0.0815 -0.111 ...
 $ dim5 : num  -0.1458 0.194 0.162 -0.1708 -0.0188 ...
 $ dim6 : num  0.00143 0.29715 0.03987 0.08689 -0.17395 ...
 $ dim7 : num  -0.00803 0.1158 0.01046 0.05978 0.05938 ...
 $ dim8 : num  -0.0786 0.1147 0.0587 -0.0334 0.1909 ...
 $ dim9 : num  0.0854 0.1102 0.0522 0.0485 0.2051 ...
 $ dim10: num  0.11772 -0.0291 -0.13328 -0.00389 -0.16107 ...
 $ dim11: num  0.04401 0.07451 0.11536 0.22196 -0.00746 ...
 $ dim12: num  0.1305 0.015 0.1299 -0.2429 -0.0477 ...
 $ dim13: num  -0.0374 0.1755 -0.074 -0.0136 -0.1614 ...
 $ dim14: num  0.1316 -0.0492 0.0588 0.0797 0.0578 ...
 $ dim15: num  -0.1597 -0.1268 0.1008 0.1131 0.0941 ...
 $ dim16: num  -0.1065 0.0538 -0.0588 -0.0163 -0.0296 ...
 $ dim17: num  -0.1192 -0.0693 -0.0661 0.0429 -0.0925 ...
 $ dim18: num  0.01149 -0.04418 0.00861 -0.02725 0.01518 ...
 $ dim19: num  0.08038 0.08427 0.04674 0.00787 0.00388 ...
 $ dim20: num  0.0233 -0.0492 -0.1205 0.1527 0.1339 ...
 $ dim21: num  0.057 -0.0136 -0.1807 0.0302 0.0396 ...
 $ dim22: num  0.07742 -0.00802 -0.06741 0.1737 -0.12538 ...
 $ dim23: num  -0.109027 0.007008 0.000556 -0.057942 -0.034231 ...
 $ dim24: num  -0.05323 0.00737 0.00135 0.01436 -0.05806 ...
 $ dim25: num  0.0815 -0.0887 0.0303 0.1225 -0.0028 ...
 $ dim26: num  -0.0851 0.0693 0.0442 0.0431 0.0718 ...
 $ dim27: num  -0.0214 -0.0216 0.0364 0.1089 -0.041 ...
 $ dim28: num  0.0478 -0.0038 -0.0992 -0.0125 0.0536 ...
 $ dim29: num  -0.04607 0.09187 -0.02084 -0.03893 -0.03676 ...
 $ dim30: num  0.04878 0.00316 0.0223 0.043 -0.06424 ...
```

	label	dim1	dim2	dim3	dim4	dim5	dim6	dim7	dim8
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
V1	20409	0.4980725060	0.5206232041	1.175278e-01	-0.1702753975	-1.458471e-01	1.426727e-03	-8.028745e-03	-7.861914e-02
V2	20409	0.4054563151	0.4450773505	5.306249e-02	-0.1360230989	1.939511e-01	2.971534e-01	1.157985e-01	1.146755e-01
V3	20409	0.4388928700	0.4835451748	1.100448e-01	-0.3631578905	1.419521e-01	3.987432e-02	1.046112e-02	5.869128e-02
V4	20409	-0.1013120185	0.0515269744	1.250884e-01	-0.0815308451	-1.707650e-01	8.688634e-02	5.977568e-02	-3.340610e-02
V5	20409	0.3971858616	-0.2893145030	6.924303e-02	-0.1109672718	-1.880515e-02	-1.739462e-01	5.938137e-02	1.908783e-01
V6	20409	0.8592059236	-0.0861166247	-4.482786e-02	0.0918064183	-1.247434e-01	-9.935780e-02	3.225483e-02	3.639416e-02
V7	20409	0.7511464199	0.1569206105	1.077312e-01	0.1490037261	2.742168e-02	-1.185176e-01	-6.624512e-02	-8.404849e-02
V8	20409	0.8200655822	0.1717857897	-7.891303e-02	0.1313414223	-1.340032e-01	-1.075936e-03	-1.573946e-01	-3.801776e-02
V9	20409	0.0598521621	0.3529782871	8.229825e-02	-0.1635255209	-5.137578e-02	1.981870e-01	1.148669e-01	2.769956e-01
V10	20409	0.4757691138	-0.1801867133	4.194072e-02	-0.3336278878	2.168655e-02	4.691868e-02	-1.807515e-02	-2.050473e-01

1–10 of 3,431 rows | 1–10 of 31 columns

Previous 1 2 3 4 5 6 ... 100 Next

Accuracy Comparison for All Models

Model	Accuracy (4 labels)	Accuracy (2 labels)
Decision Tree	53.71%	70%
Random Forest	63.54%	83.70%
KNN	56.73% (k=5)	80.34% (k=17)
SVM Radial	61.43%	78.86%
CNN	93.1%	97.7%

3.3.4 Conclusion/Summary

The findings or summary from the study of experiments in this project are as follows:

- For image classification, convolutional neural networks give much higher accuracy in making predictions than traditional classification algorithms, as shown in the table below.
- CNNs can even work well with very small sample sets.
- When dealing with large image data sets, like Google Landmark data, CNNs need much computing power to carry out their “learning”.
- Keras library with TensorFlow backend provides an easy-to-use framework that allows fast prototyping.
- Hyperparameter-tuning of CNNs can get very complex and time-consuming.

3.4 [IST-718] Big Data Analytics

3.4.1 About the project/data

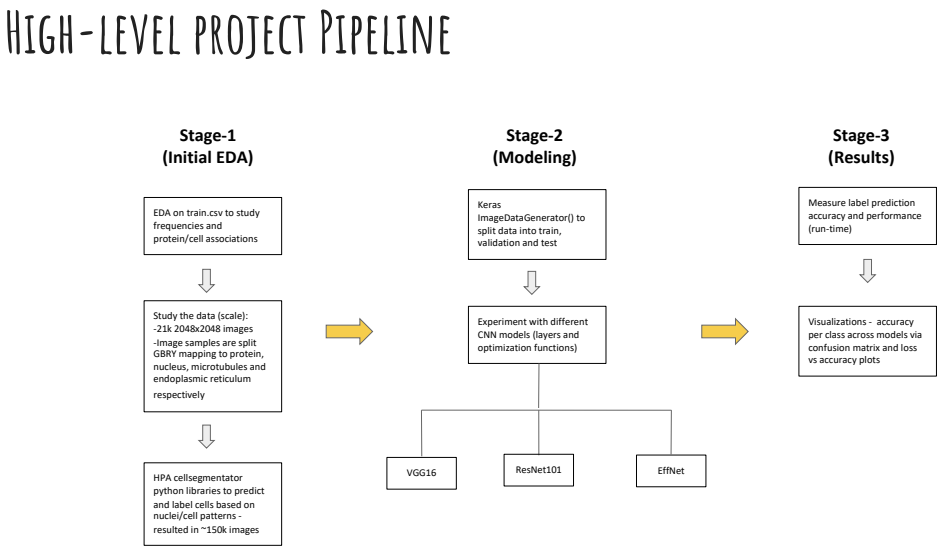
Human protein Atlas Single cell classifier (Kaggle competition)

3.4.2 Source-code

https://github.com/sharatsv/MS-DataScience/tree/main/IST-718/Final-Project
https://github.com/srihari-busam/hpa-deep-learning

3.4.3 Highlights

3.4.3.1 Execution – Pipelines and experiments with CNNs

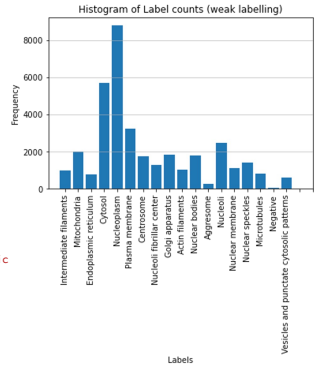


Exploratory Data Analysis (on weak labels/train.csv)

- Initial EDA was on train.csv comprising weak labels linked to each image
- Sample of transformed data shown below where an image may have upto 5 labels

  NOTE: 19 is a custom label showing no data
- Histogram showing the distribution of various proteins/other cell components across 21807 images

```
Classes of proteins
    0: 'Nucleoplasm',
    1: 'Nuclear membrane',
    2: 'Nucleoli',
    3: 'Nucleoli fibrillar center',
    4: 'Nuclear speckles',
    5: 'Nuclear bodies',
    6: 'Endoplasmic reticulum',
    7: 'Golgi apparatus',
    8: 'Intermediate filaments',
    9: 'Actin filaments',
    10: 'Microtubules',
    11: 'Mitotic spindle',
    12: 'Centrosome',
    13: 'Plasma membrane',
    14: 'Mitochondria',
    15: 'Aggresome',
    16: 'Cytosol',
    17: 'Vesicles and punctate cytosolic
patterns',
    18: 'Negative'
```



	ID	Label-1	Label-2	Label-3	Label-4	Label-5
1	5c27f04c-bb99-11e8-b2b9-ac1f6b6435d0	8	5	0	19	19
2	5fb643ee-bb99-11e8-b2b9-ac1f6b6435d0	14	0	19	19	19
3	60b57878-bb99-11e8-b2b9-ac1f6b6435d0	6	1	19	19	19
4	5c1a898e-bb99-11e8-b2b9-ac1f6b6435d0	16	10	19	19	19
5	5b931256-bb99-11e8-b2b9-ac1f6b6435d0	14	0	19	19	19

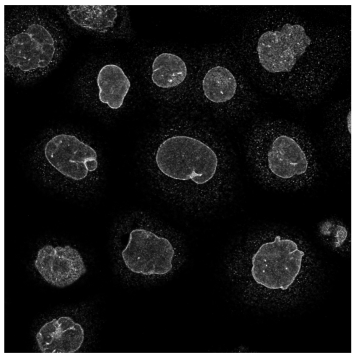
Exploratory Data Analysis (on weak labels/train.csv)

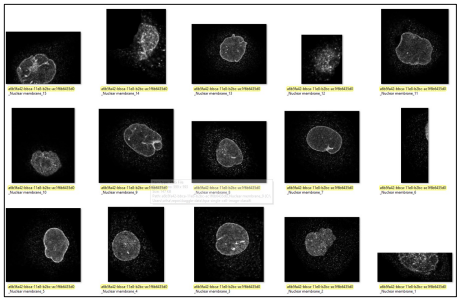
- Finally, we were interested in understanding proteins or other cell components that occurred frequently using the Apriori algorithm/Associated Rule Mining technique
- The table alongside shows relationships - highlighted ones relatively with high confidence while others in 25-30%

	Association	Support	Confidence	Lift
0	Microtubules -> Mitotic spindle	0.001009	0.282051	7.518839
1	Nucleoplasm -> Cytosol -> Mitotic spindle	0.000780	0.566667	2.173568
2	Nucleoplasm -> Mitochondria -> Vesicles and pu...	0.000642	0.875000	2.168950
3	Nuclear bodies -> Nucleoplasm -> Vesicles and ...	0.000459	0.833333	2.065666
4	Microtubules -> Mitotic spindle -> None	0.001009	0.282051	7.518839
5	Nucleoli -> Mitochondria -> Plasma membrane	0.000917	0.259740	2.310851
6	Nucleoli -> Cytosol -> Endoplasmic reticulum	0.001513	0.268293	2.386940
7	Nucleoplasm -> Cytosol -> Mitotic spindle -> None	0.000780	0.566667	2.174715
8	Nucleoplasm -> Mitochondria -> None -> Vesicle...	0.000642	0.875000	2.169443
9	Nucleoli -> Nucleoplasm -> Cytosol -> Intermed...	0.000780	0.265625	2.403410
10	Nuclear bodies -> Nucleoplasm -> Vesicles and ...	0.000459	0.833333	2.066136
11	Nucleoli -> Mitochondria -> Plasma membrane ->...	0.000917	0.259740	2.310851
12	Nucleoli -> Endoplasmic reticulum -> Cytosol -...	0.001513	0.268293	2.386940
13	Intermediate filaments -> Cytosol -> Nucleopla...	0.000780	0.265625	2.405406

Solving the data challenge

- The idea was to convert weak labels to concrete labeled data
- Approach
  - Identified images that only has one class. There are about **10,412** images with single cell type only
  - Used the previous competition cell segmentation model to separate them into individual cell images(python magic used to isolate the images)
  - This approach resulted in **151,707** strong labeled samples
  - It took **~2 days** to convert the data!!





Imageid shown: a6b5fa42-bbca-11e8-b2bc-ac1f6b6435d0_**Nuclear membrane**

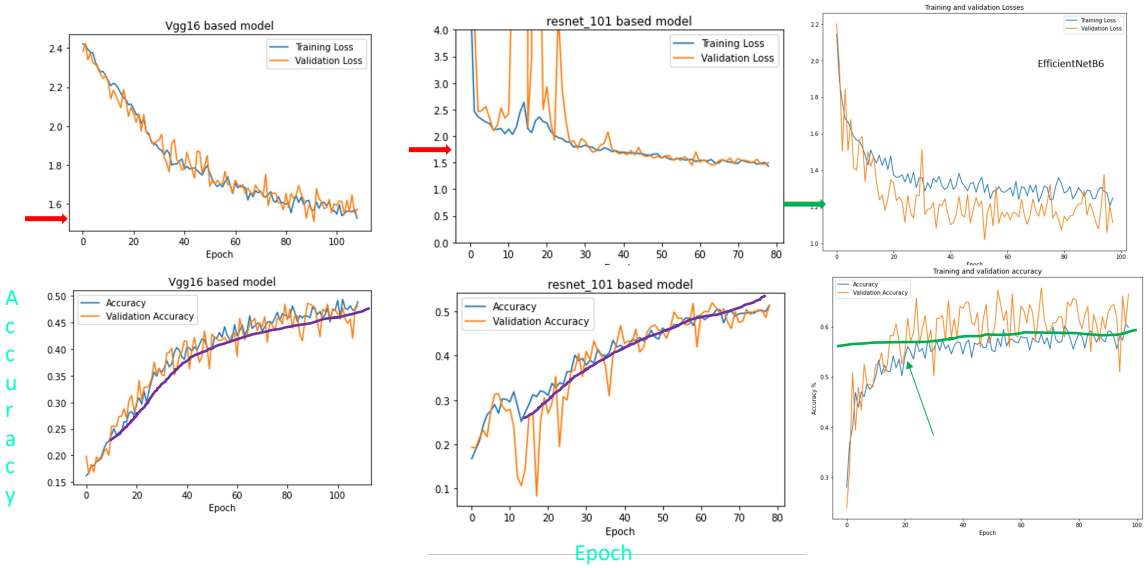
Data cleanup & scoping for modeling

- After conversion removed all files <4kb and >1MB
  - Reasoning:
    - Small size - may not have enough features
    - Large size - compression may introduce artifacts
- Scoped to 12 classes only
  - When the experiments started Kaggle has about ~40% as the top. We want to hit atleast 50% accuracy and one way we thought is to have enough samples to train.
  - Chosen classes with more than 1k samples
  - "Negative" class images are removed as these could be any cell image and could be a noise

ConvNet models - experiments & results

Model based on	Input Details	Trainable params	Optimizer	Training Time	Training Accuracy	Validation Accuracy
VGG16 16 layers Weights: None Weights locked : NO	Res:512 X 512 Batchsize: 32 Steps per epoch:100 Validation steps: 20	14,714,688	Adam (lr=0.0001)	~5hrs	**46.59%**	**48.59%**
Resnet101 101 layers Weights: imagenet Weights locked : NO	Res:512 X 512 Batchsize: 24 Steps per epoch:120 Validation steps: 30	48,844,300	NAdam (lr=0.0001)	~5hrs	**49.78%**	**51.94%**
EfficientNetB6 Weights: imagenet Weights locked : NO	Res:512 X 512 Batchsize: 7 Steps per epoch:150 Validation steps: 10	40,763,364	Nadam (lr=0.0001) (tried Ada, rmsprop,SGD)	~8hrs	**70.08%**	**67.14%**

Loss & Accuracy



3.4.4 Conclusion/Summary

When we started the project, the leaderboard on the Kaggle competition was at **~44%** accuracy and beating this number became our immediate goal. As a team, we scoped the problem from 19 to 12 class detection but was able to achieve ***~67%*** *accuracy for the 12 classes* identified.

There is still room to improve models and experiment more. However, we conclude at this point that we met our accuracy goal for the project.

3.5 [IST-652] Scripting for Data Analysis

3.5.1 About the project/data

Securing enterprise/cloud data can be realized via what is commonly known as network firewall policies or rules

- In traditional network architecture, perimeter firewalls can achieve this but with the advent of software defined networks (SDN), firewalls are de-centralized and distributed

The goal at large was to gather flow data in the form of logs (un-structured data) across the distributed plane and build an analytics/recommendation system based on machine-learning

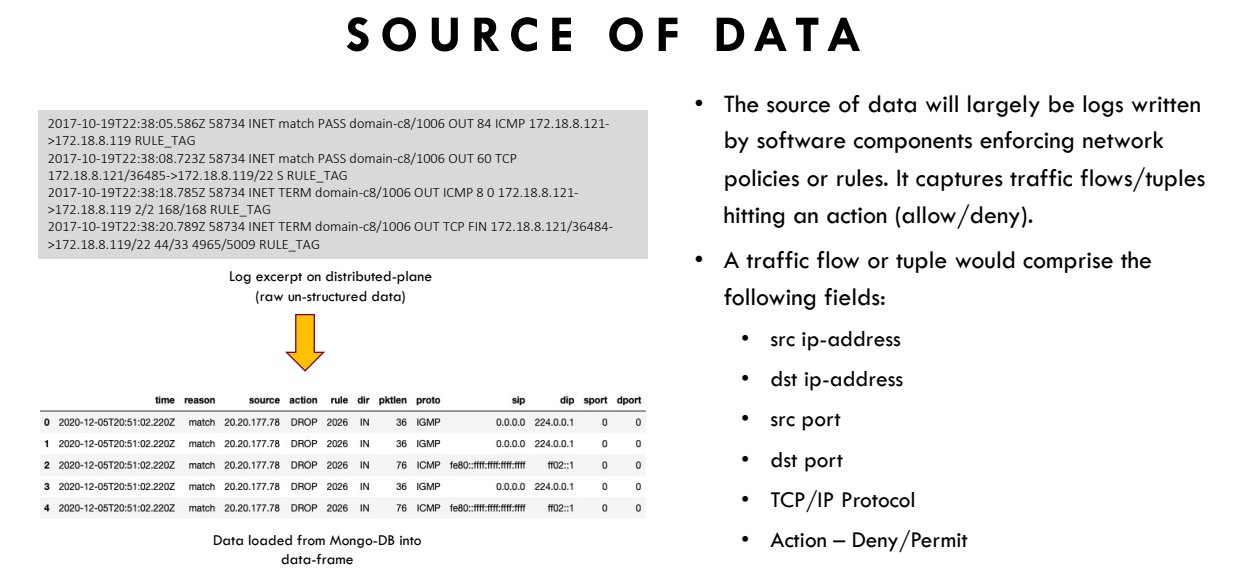


Fig: Translate raw files comprising data-frame for analysis

3.5.2 Source-code

https://github.com/sharatsv/MS-DataScience/tree/main/IST-652/Final-Project

3.5.3 Highlights

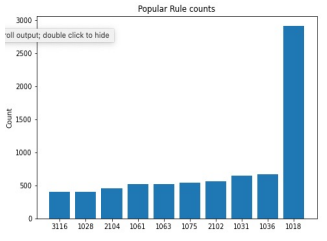
RESEARCH QUESTIONS

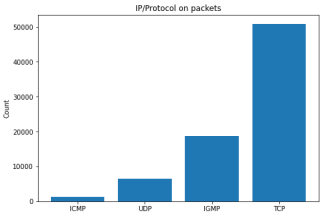
Workflow

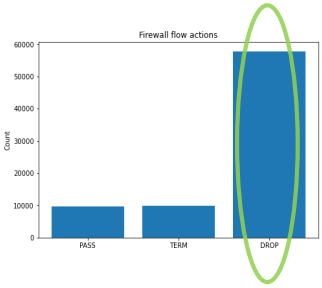
1. Traffic patterns or flows across the datacenter
2. Analyze and visualize traffic patterns – get counts of flows across TCP, UDP and ICMP protocols
3. Provide rule metrics across all network policies
4. Plot time-series graphs:
   - For high hitting flows (allow/deny) identify spikes, saw-tooth behavior or sustained/repeated patterns etc.
5. Finally, based on traffic patterns make recommendations to add/remove/update network policies or rules. Explore machine-learning algorithms to achieve this.

3

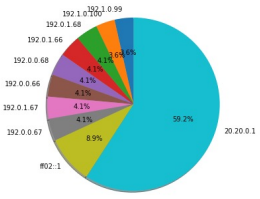
ANALYTICS-DASHBOARD



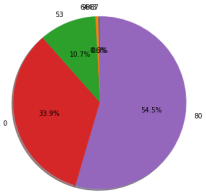




Zoom in on DROP flows



Drop flows categorized by source



Drop flows categorized by L4 ports

5

Conclusion/Summary

3.5.3.1 Recommendations

Decision-Tree (used sklearn libraries) to train, test and predict outcomes given a flow tuple:

- Data-split: 80% train, 20% test
- Result: Accuracy of 78%

```
# Step-5:
# Check the accuracy
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

# Conclusion:
# Using Decision-Trees (GINI method), given certain packet/flow attributes we
# predict with upto 78% accuracy if Firewall action is DROP, PASS or TERM
```

```
Accuracy: 0.7804326767839845
```

3.5.4 Conclusion and future work

- Improve accuracy of prediction with usage of Random Forests (ensemble methods) or other kernel SVM techniques
- Explore application of deep-learning methods
    - Research if database signatures of commonly known threat/attack vectors can be maintained in an MNIST-like datastore
- Build a simple utility – given a packet tuple (s.ip, d.ip, ports, protocol) predict Firewall action based on flows extracted

3.6 [IST-736] Text Mining

3.6.1 About the project/data

- **Title:** Marvel Search Engine
- **Data source:** The data collected for this project came from multiple sources. Naturally, there were inconsistent file formats when pulling data from multiple sources. This piece had to be resolved before any of the rest of the project moved forward to standardize all the scripts into one model-friendly format.

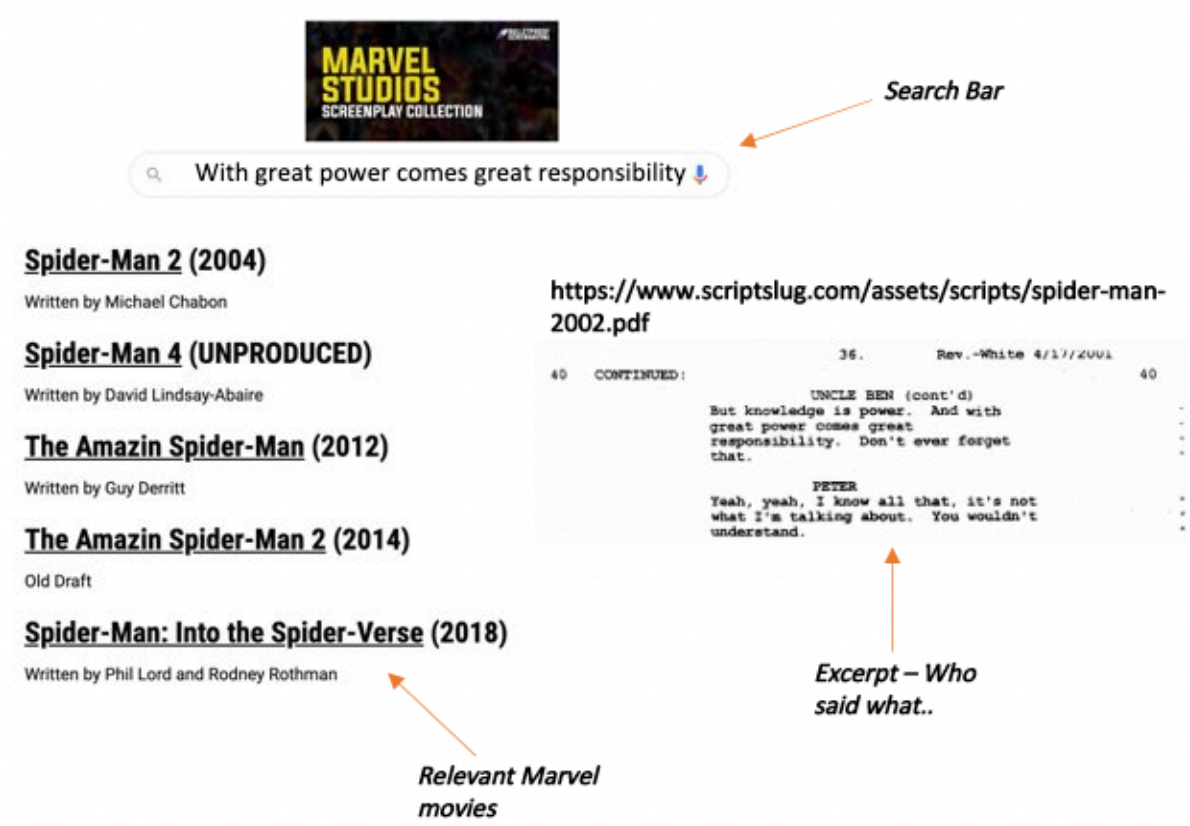
When scripts are published, they are more likely to come as PDF's, but there already existed a public repository [Marvel-Dialog-NLP] with the PDF's converted into a text format for 70% or 16 out of 23 of the movie scripts. The remaining scripts were taken directly as PDFs from a movie script website [MCU-Script-PDF] which had to go through additional processing steps as the other movies.

3.6.2 Source-code

https://github.com/sharatsv/MS-DataScience/tree/main/IST-736/Final-Project
https://github.com/ramosem97/mcu_marvel_search_engine

3.6.3 Highlights

3.6.3.1 Prototype/High-level design



3.6.3.2 Models and results

3.6.3.2.1 BM25

Implementing BM-25 is incredibly simple owing to availability of library *rank-bm25.* Once the data was sufficiently translated and cleaned, it is aggregated into a data-frame as below:

	Movie	Script
0	spider-man-4.pdf	(CONTINUED) Co another. building swing cost...
1	.DS_Store	(CONTINUED) Co another. building swing cost...
2	spider-man-2.pdf	him. know find might feel seemed MAY (...
3	incredible-hulk.pdf	event.) (MORE suppressed ust maybe we've ev...
4	spider-man-2002.pdf	Ww, 1 @ , - enone -_~ 7. ny ~ _ Lo bank those...

Text in the Script column is passed through the tokenize function in *SpaCy.* The resulting data structure of list of lists is passed to bm-25 to extract two results namely:

- Given a search query locate the movie using the *bm-25.get_scores()* method

```
import re

bm25 = BM25Okapi(tok_text)

tokenized_query = proc_query.lower().split(" ")
import time

print('Final query: %s' % proc_query)

# Retrieve most relevent movie to query
scores = bm25.get_scores(tokenized_query)
max_score_index = list(scores).index(max(scores))

movie = re.sub(r'\..*$', '', df_movies_pkl.iloc[max_score_index][0])

print('Most relevant movie to query: %s' % movie)
```

```
Final query: great power comes great responsibility
Most relevant movie to query: spider-man-2
```

- Given a search query locate relevant texts comprising excerpts or parts of speech using the *bm-25.get_top_n()* method

```
# Retrieve search like index - Top N documents that match
# the quote

start_time = time.time()
results = bm25.get_top_n(tokenized_query, df_movies_pkl.Script.values, n=3)
end_time = time.time()

print('Searching %d scripts took %s seconds' %(len(scores), (end_time-start_time)))

for i in results:
      print(i)
```

```
Lincoln fountain. | building  man.  Lincoln brother    accent) CABBIE CABBIES.  CABBIE lean
s couple  talking cab,   ExT STREET 64 64 shadow barely night,  disappears  turn. Grocer
Robber   sharply Robber's  |! Stunned, guz tne ia hand. x's aro' wraps  yanks   around web~
-strand ! THWIP |   shoot turns, hin, Robber bat, baseball money m, Cal out, chases GROCER
suddenly ien carrying  other.   ~e / ( gur one  ROBBER hand,  sack  deli, Korean  EXT DELI
DAY 63 63   heart city.  heads train    Manhattan  background  EXT, MANHATTAN DAY A63 A63 c
ollege lap.  textbooks Peter  pile   skyline,  train,  staring sits INT. DAY TRAIN 62 62  R
OARS . tunnel  train EXT L62 MANHATTAN L.62 DAY TO: DISSOLVE ROAR.  HEAR GREAT become  appr
oach .  great power comes responsibility, great wath er...   BEN (V.0.) UNCLE C) CONTINUED:
K62 K62 .-Tan Rev 54. 4/12/2001  dS te oar =  ne, -. wT. 4s < Zab, x ie ye ws 233 apy Aly U
ES soos , Seber) ee Loe, Ts tat ng Sate wo - ORS Ee yg ge. ye Slee , : . a. 7 foal oe . moo
```

3.6.3.2.2 Google USE

In the trials performed, the Google USE performed better overall. The USE could produce more precise results than the BM25 did when searching for various phrases. This is because the BM25 relies on a bag of words approach, which means that if the exact words input in the search bar does not match, there will not be a precise result. With USE being able to compare phrases and words semantically, this approach provided the best approach for the backend for the search engine. USE was able to see that although the exact words or phrases were not within the different documents, USE could match the closely related semantic versions of what was the individual was searching. While BM25 does pull results quickly, some of these results may not make sense in terms of what it is that the individual is searching for.

3.6.3.2.3 Dashboard

Below is the dashboard or the face of the search engine, where given a query search string it pulls up relevant movies, relevant characters and relevant monologues or dialogues with highest relevant displayed ranked on top

Search Here:

Character Dies | Search

Relevant Movies

Movie	Release Year
Iron Man	2008
Avengers: Endgame	2019
Thor: The Dark World	2013
Iron Man 3	2013
Avengers: Age of Ultron	2015

Relevant Characters

Character	Movie Apperances
MR. HARRINGTON	['Spider-Man: Far From Home', 'Spider-Man: Homecoming']
HELA	['Thor: Ragnarok']
HELMUT ZEMO	['Captain America: Civil War']
DR. ARNIM ZOLA	['Captain America: The First Avenger', 'Captain America: The Winter Soldier']
SCOTT LANG	['Ant-Man', 'Ant-Man and the Wasp', 'Avengers: Endgame', 'Captain America: Civil War']

Relevant Movie Lines

Character	Line	Movie	Relevance
DRAX	Die, blanket of death!	Avengers: Infinity War	0.296999990940094
IVAN VANKO	You come from a family of thieves and butchers. And now, like all guilty men, you try to rewrite your own history. And you forget all the lives the Stark family has destroyed.	Iron Man 2	0.257999986410141
GAMORA	Everything will die.	Guardians of the Galaxy	0.257999986410141
THOR	You know, I'm 1,500 years old. I've killed twice as many enemies as that, and every one would have rather killed me, but none succeeded. I'm only alive because fate wants me alive. Thanos is the latest in a long line of bastards and he will be the latest to feel my vengeance. Fate wills it so.	Avengers: Infinity War	0.2460000067949295
DOCTOR STRANGE	The patient's not dead, but he's dying. Do you still want to harvest his organs?	Doctor Strange	0.23899999260902405
HANK PYM	We lost her in a plane crash.	Ant-Man	0.23800000548362732
T'CHALLA	In my culture death is not the end. It's more of a . . . stepping-off point. You reach out with both hands and Bast and Sekhmet, they lead you into the green veldt where . . . you can run forever.	Captain America: Civil War	0.23600000143051147
MR. HARRINGTON	Turns out, she ran off with a guy in her hiking group. We had a fake funeral for her and everything. Well, the funeral was real because I thought she was really dead. Wanna see the video?	Spider-Man: Far From Home	0.23499999940395355
ERIK SELVIG	In some cases.	Thor	0.22800000001192093
SIF	No! I will die a warrior's death. Stories will be told of this day--	Thor	0.22499999403953552

3.6.4 Conclusion and future work

After running experiments, Google USE produced better results. USE was not only able to pull references to the search quicker but was also able to match the search to the different scripts semantically. Matching semantically is something that BM25 continually struggled with, seeing as it is based on a "bag of

words" model. With this being the case, the BM25 would occasionally produce odd results when trying to query things not explicitly mentioned within the documents themselves. It is not to say that when provided a phrase or words within the scripts, BM25 could not perform the proper functions to display what it is the individual is searching for, but that the inconsistency leads to USE being the better of the two. The semantic matching elevates USE to outperform BM25 in this particular use case.

The whole reason an individual might be searching for different things within the Marvel Cinematic Universe (MCU) is that they are trying to understand the context of what was happening and catch up without viewing the previous films. With this being why this search engine was created, it might be hard for the person to know a particular phrase or word that might have been used in a previous film. Thus, semantic matching is crucial to the searches populating results that make sense to the individual understanding of the context and possible story elements moving forward into phase four.

3.7 [IST-664] Natural Language Processing

3.6.1 About the project/data

- **Title:** Detection of SPAM in email
- **Data source:** Detect spam emails based on a dataset produced from Enron's public email corpus.

3.6.2 Source code

https://github.com/sharatsv/MS-DataScience/tree/main/IST-664/Final-Project

3.6.3 Highlights

3.6.3.1 Model and results

Historically, although other algorithms like Support vector machines and Boosting are known to be top-performing, we will use Naïve Bayes as it is vastly popular in commercial and open-source category spam filters.

NLTK readily offers the Naïve Bayes classification method, and we will see in the following sections how it can be used to solve the problem and study its performance.

While Naïve Bayes would largely help us with the classification, we take two fundamental NLP approaches:

1. Unigram feature-set - In this approach, we derive a feature set comprising single words ordered by frequency and pick the top 2000 words.

```
# We'll find the 2000 most common words and use them as an
# important feature of the whole corpus
def unigram_freq(docs):
    all_words = []
    # Write a regex to pull only the word portion & leave
    # out any punctuation marks etc.
    for (word_tokens, category) in docs:
        for word in word_tokens:
            # Not writing a regex here since we want to know if
            # random patterns or words would cause an email to be spam
            all_words.append(word)
    top_words = nltk.FreqDist(all_words)
    most_common_words = top_words.most_common(2000)
    word_features = [word for (word,count) in most_common_words]
    return all_words, word_features

# uni_features now has the top-2000 most common words
# across the entire spam and ham data
all_words, uni_features = unigram_freq(spam_n_ham)

print(all_words[:5], uni_features[:5])
```

```
['remote', 'www', 'back', 'sport', 'special'] ['please', '2000', 'subject', 'enron', 'thanks']
```

Sample of unigram feature-sets is:
`Email classified` **`spam`** `has the following words:` **`['back', 'www', 'order', 'receive', 'special', 'control', 'allow', 'events']`**

2. Bi-gram feature-set - In this approach, we will group two-words at a time using the *NLTK collocations.BigramAssocMeasures()* and score frequencies using the *score_ngrams()* functionality.

```
# Let's try the same with bi-grams to see if we get better
# accuracy.

from nltk.collocations import *

def bigram_freq(all_words):
    #creating bigrams features for the corpus and applying cleaning steps
    bigram_measures = nltk.collocations.BigramAssocMeasures()
    finder = BigramCollocationFinder.from_words(all_words)
    scored = finder.score_ngrams(bigram_measures.raw_freq)

    #extracting clean bigrams (no frequency information)
    bigram_features = [bigram for (bigram, count) in scored[:2000]]

    return bigram_features

```

Sample of bi-gram feature-sets is:
`Email classified` **`spam`** `has the following words:` **`["contains(('receive', 'control'))", "contains(('allow', 'receive'))"]`**

Results as evaluation measures

- RECALL

  Definition: The percentage of actual yes answers that are right
  Or
  Recall = TP / (TP + FP)
- PRECISION

  Definition: The percentage of predicted yes answers that are right
  Or
  Precision = TP / (TP + FN)
- F-measure

Recall and precision combined into an average or harmonic mean
Or
F-measure = 2 * (Recall * Precision) / (Recall + Precision)

- ACCURACY

  Percentage of correct Yes and No out all the text examples
  Or
  Accuracy = TP + TN / (TP + FP + FN + TN)

where TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative

These definitions are formulated into Python code in rows 18-22 and encompassed with the *eval_measures()* function as shown below:

```
1  # Utilizing a function from labs let's now obtain the eval_measures
2  def eval_measures(gold, predicted):
3      # get a list of labels
4      labels = list(set(gold))
5      # these lists have values for each label
6      recall_list = []
7      precision_list = []
8      F1_list = []
9      for lab in labels:
10         # for each label, compare gold and predicted lists and compute values
11         TP = FP = FN = TN = 0
12         for i, val in enumerate(gold):
13             if val == lab and predicted[i] == lab:  TP += 1
14             if val == lab and predicted[i] != lab:  FN += 1
15             if val != lab and predicted[i] == lab:  FP += 1
16             if val != lab and predicted[i] != lab:  TN += 1
17         # use these to compute recall, precision, F1
18         recall = TP / (TP + FP)
19         precision = TP / (TP + FN)
20         recall_list.append(recall)
21         precision_list.append(precision)
22         F1_list.append( 2 * (recall * precision) / (recall + precision))
23
24     # the evaluation measures in a table with one row per label
25     print('\tPrecision\tRecall\t\tF1')
26     # print measures for each label
27     for i, lab in enumerate(labels):
28         print(lab, '\t', "{:10.3f}".format(precision_list[i]), \
29           "{:10.3f}".format(recall_list[i]), "{:10.3f}".format(F1_list[i]))
```

Using the classifier from the unigram and bi-gram classification task earlier we will put together two lists – actu al and predicted. We will then pass the lists to the *eval_measures()* function to get a summary of the results.

```
31 # Eval measures for uni-gram classifier
32 eval_measures(actual, predicted)
```

	Precision	Recall	F1
spam	0.993	0.882	0.935
ham	0.945	0.997	0.971

```
1  # Eval measures for bi-gram classifier
2  eval_measures(actual, predicted)
```

	Precision	Recall	F1
spam	0.993	0.682	0.809
ham	0.809	0.997	0.893

Interpreting evaluation measures

- Precision for bi-gram classification shows poor-performance for ham classification emails.
  - Precision for spam emails is comparable at 99.3%.
- Recall for bi-gram classification also shows poor-performance for spam (that is, the percent of predicti on spam vs actual is only at 68%).
- F-measure is showing an overall improved performance with unigrams.

3.6.4 Conclusion and future work

In this case-study, spam and ham emails were classified using a Naïve Bayes classification algorithm. Feature-sets were mostly based on unigrams and bi-grams derived from the email corpus.

The model was trained and tested across datasets derived from cross-validation (k-folds/k=5) and yielded moderately poorer performance or accuracy for the bi-gram feature-set at 83.8% (compared to 96% with unigrams).

This is strangely an outlier in comparison with some of the experiments with books from Glutenberg corpus in other home-work assignments where we always recorded bi-grams to show improved performance in comparison with unigrams. Further analysis with SPAM email classifier using evaluation measurement techniques also showed particularly poor predictions to classify ham (non-spam) emails.

To improve the accuracy of the model, whitepaper '*Spam Filtering with Naïve Bayes – Which Naïve Bayes?*' seems to provide some answers. While the paper broadly acknowledges the popularity of the Naïve Bayes algorithms for SPAM classification it brings to light some possible nuances with the algorithm – and choices between the following:

- Multi-variate Bernoulli NB
- Multinominal NB, TF attributes
- Multinominal NB, Boolean attributes
- Multi-variate Gauss NB
- Flexible Bayes

Per the results in the whitepaper the Multinominal NB, Boolean spam and ham recall is at >97%. As part of some of the future work I intend to explore some of these variants.

3.8 [IST-719] Information Visualization

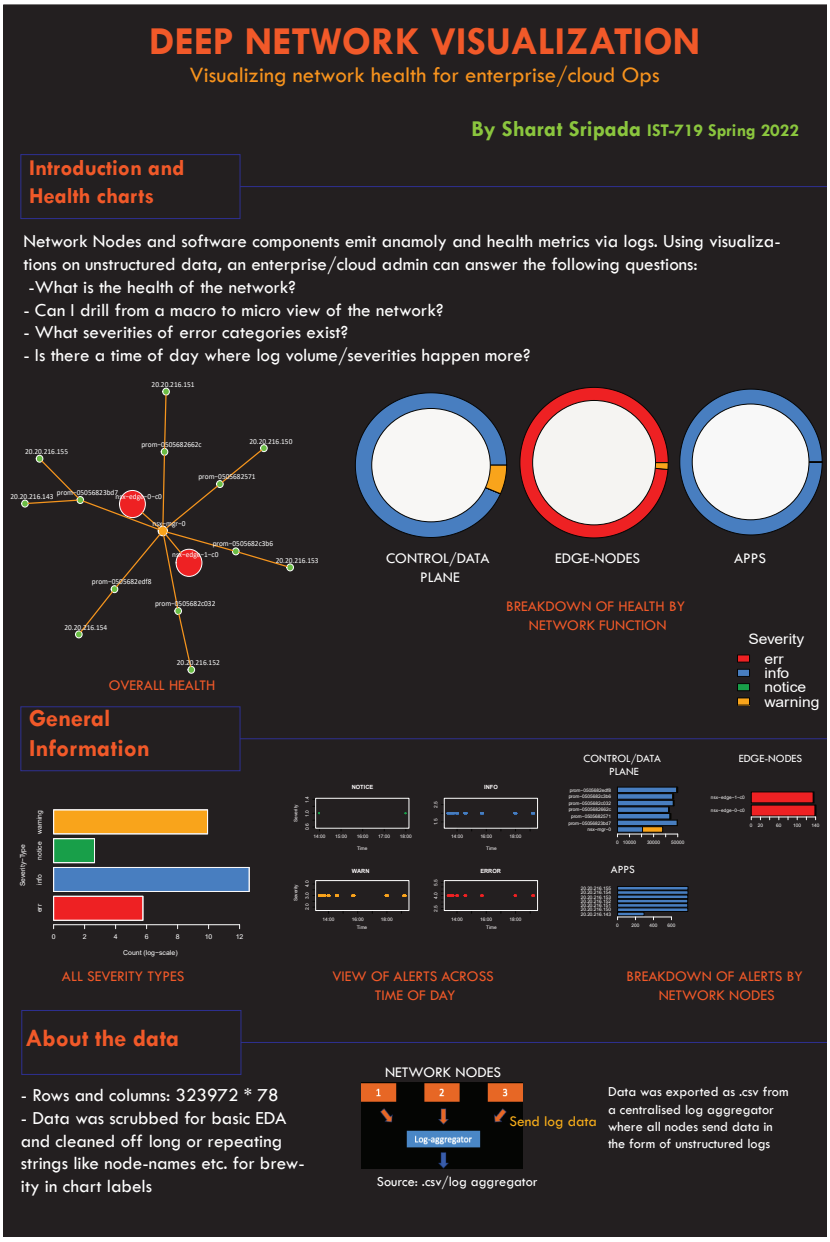
3.6.1 About the project/data

- **Title:** Deep Network Visualization

The project was executed through the delivery of a poster which is supposed to effectively convey details from data-source to the problem being solved using information visualization techniques taught during the course. The final poster is a product of Microsoft Adobe Illustrator with R churning out visualizations (from raw data).

3.6.2 Source code

https://github.com/sharatsv/MS-DataScience/tree/main/IST-719/Final-Poster-Sharat-Sripada


DEEP NETWORK VISUALIZATION
Visualizing network health for enterprise/cloud Ops
By Sharat Sripada IST-719 Spring 2022
Introduction and Health charts
Network Nodes and software components emit anamoly and health metrics via logs. Using visualizations on unstructured data, an enterprise/cloud admin can answer the following questions:
-What is the health of the network?
- Can I drill from a macro to micro view of the network?
- What severities of error categories exist?
- Is there a time of day where log volume/severities happen more?
OVERALL HEALTH
CONTROL/DATA PLANE
EDGE-NODES
APPS
BREAKDOWN OF HEALTH BY NETWORK FUNCTION
Severity
err
info
notice
warning
General Information
ALL SEVERITY TYPES
VIEW OF ALERTS ACROSS TIME OF DAY
BREAKDOWN OF ALERTS BY NETWORK NODES
About the data
- Rows and columns: 323972 * 78
- Data was scrubbed for basic EDA and cleaned off long or repeating strings like node-names etc. for brewity in chart labels
NETWORK NODES
Send log data
Log-aggregator
Source: .csv/log aggregator
Data was exported as .csv from a centralised log aggregator where all nodes send data in the form of unstructured logs

3.9 [MBC-638] Data Anls & Decisn Making

3.9.1 About the project

- **Title:** Improve turn-around time & time-to-resolution for customer found defects (CFDs).

3.9.2 Source Code

https://github.com/sharatsv/MS-DataScience/tree/main/MBC-638/Final-Project

3.9.3 Highlights

Define – Problem Statement, Impact & Goals

Improve turn-around time & time-to-resolution for customer found defects (CFDs).

- **Impact**
  - Impact to Customers (Consumers) – Critical services and applications can be impacted depending on the severity of a CFD. Severity is categorized P1-P4 (P1 being most severe) and correspondingly have a spiraling effects on their growth and revenue.
  - Impact to R&D (Producers) - Engaging multi-tier support/R&D escalation groups, development and test teams investigating, root-causing and fixing CFDs impacts productivity & delivery of critical milestones.

- **Goals**
  - ~10% reduction in turn-around time/mean time to defect resolution (in terms of hours)
  - ~5% lower incoming CFD rate – bracing some customer practices into our existing

- **Team**

Vice-President - Quality Engineering, Customer Advocacy Specialist & Sharat Sripada

NOTE – Cannot callout a dollar value in Impact/Savings/Goals since it is protected by NDA.

Define – Operational Definitions

Defects

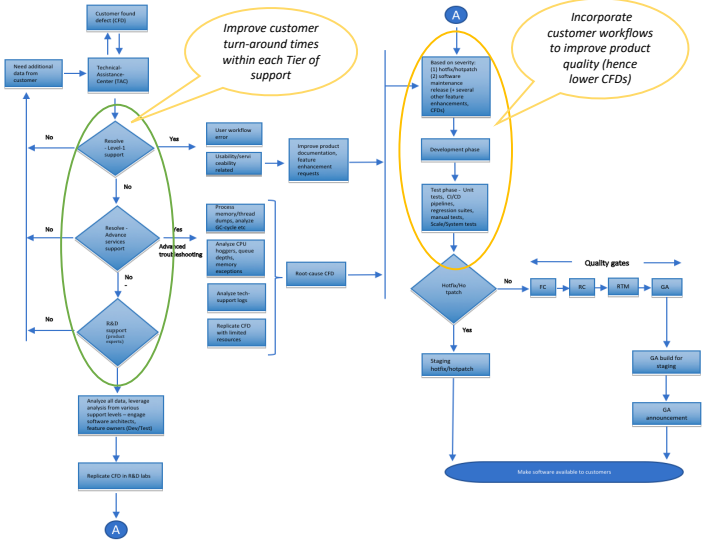
Based on the goals, we were able to identify the following indices and determine some pilot values for each of the categories:

- Turn-around time top contributors:
  - Define problem > 24 hours
  - Upload logs > 8 hours
  - Tag the right category/component > 120 hours
  - To close defect after RCA > 400 hours
  - Overall time to closure > 600 hours
- CFD incoming rate > 5 per day

Definitions

- Define problem: Time a support Engineer takes to understand and define/create a defect in the Bug Tracking Tool (also explains steps he did to triage or remediate a problem)
- Upload logs: Time a support Engineer takes to collect diagnostic information/logs from customer production & upload it to the defect in the Bug Tracking Tool
- Tag the right category/component: Time a defect is un-attended to since it landed in a wrong component
- To close defect after RCA: Time a defect is left open after an RCA for completion of other book-keeping processes

Define Phase – Process Map



Identified broadly two areas of process improvement:

1. Improve turn-around times within various Tiers of support

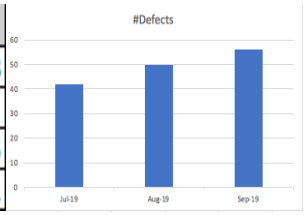
2. Enhance certain R&D practices to improve overall product quality (reviews, incorporating customer workflows, automation etc).

NOTE – Pilot for (2) of the process improvement is planned in Phase-2, Q1 FY-2020. We will merely, plot/use some meaningful data but do not strive to prove any benefits.

Measure Phase – Data collection

	Tier-1 technical-support				Core Engineering/Escalation support				Overall turn-around time			
Issue	In-Time	Out-Time	Time-Taken (day:hrs:mins)	Time-Taken (normalize to hours)	In-Time	Out-Time	Time-Taken (day:hrs:mins)	Time-Taken (normalize to hours)	Open	Close	Time-Taken (day:hrs:mins)	Time-Taken (normalize to hours)
1	7/1/19 5:54 AM	7/1/19 8:46 AM	0:2:51	2.85	7/2/19 9:22 AM	7/3/19 10:51 PM	1:13:28	37.47	7/1/19 5:54 AM	7/4/19 6:36 AM	3:0:42	72.70
2	7/2/19 8:08 AM	7/2/19 8:08 AM	0:0:0	0.00	7/2/19 11:52 AM	9/3/19 11:07 AM	62:23:15	1511.25	7/2/19 8:08 AM	9/20/19 1:39 PM	80:5:31	1925.52
3	7/2/19 11:55 PM	7/3/19 5:00 AM	0:5:5	5.08	7/3/19 11:06 AM	7/26/19 11:21 AM	23:0:14	552.58	7/2/19 11:55 PM	8/19/19 9:19 AM	47:9:24	1137.40
4	7/3/19 11:55 AM	7/3/19 2:16 PM	0:2:21	2.35	7/5/19 1:41 AM	7/10/19 10:57 AM	5:9:15	129.63	7/3/19 11:55 AM	8/15/19 9:14 PM	43:9:18	1041.30
5	7/3/19 2:23 PM	7/4/19 8:16 AM	0:17:53	17.88	7/4/19 2:14 AM	7/4/19 7:37 PM	0:17:23	17.38	7/3/19 2:23 PM	7/4/19 7:37 PM	1:5:14	29.23
6	7/4/19 6:58 AM	7/5/19 12:32 AM	0:17:34	17.57	7/5/19 3:40 AM	7/6/19 8:59 PM	1:17:19	41.32	7/4/19 6:58 AM	7/23/19 2:04 PM	19:7:6	463.10
7	7/4/19 7:01 AM	7/4/19 8:56 AM	0:1:54	1.90	7/5/19 6:56 AM	9/4/19 12:00 AM	60:17:3	1457.13	7/4/19 7:01 AM	9/5/19 4:13 AM	62:21:12	1509.20
8	7/5/19 12:09 PM	7/5/19 2:30 PM	0:2:21	2.35	7/5/19 2:30 PM	7/5/19 10:00 PM	0:7:29	7.48	7/5/19 12:09 PM	7/5/19 10:00 PM	0:9:50	9.83
9	7/5/19 3:36 PM	7/8/19 8:24 AM	2:16:48	64.80	7/8/19 2:52 PM	7/17/19 9:52 AM	8:17:59	209.98	7/5/19 3:36 PM	7/19/19 1:33 AM	13:9:57	321.95
10	7/6/19 12:27 PM	7/6/19 12:34 PM	0:0:6	6.00	7/7/19 10:54 PM	7/8/19 9:06 PM	0:22:12	22.20	7/6/19 12:27 PM	7/10/19 8:53 PM	4:8:26	104.43
11	7/18/19 5:50 AM	7/19/19 1:08 AM	0:19:17	19.28	7/19/19 1:20 AM	7/24/19 3:37 PM	5:14:16	134.27	7/18/19 5:50 AM	8/1/19 8:45 PM	14:14:55	350.92
12	8/2/19 4:31 AM	8/2/19 4:49 AM	0:0:18	18.00	8/7/19 6:19 AM	9/2/19 3:31 PM	26:9:12	633.20	8/2/19 4:31 AM	9/12/19 5:28 AM	41:0:57	984.95
13	8/2/19 9:37 PM	8/2/19 9:44 PM	0:0:6	6.00	8/2/19 11:58 PM	8/27/19 4:54 PM	24:16:56	592.93	8/2/19 9:37 PM	8/27/19 9:45 PM	25:0:7	600.12
14	8/6/19 2:36 AM	8/6/19 2:59 AM	0:0:23	23.00	8/6/19 11:35 AM	9/13/19 9:57 AM	37:22:22	910.37	8/6/19 2:36 AM	9/19/19 6:11 AM	44:3:35	1059.58
15	8/6/19 8:06 PM	8/6/19 9:06 PM	0:1:0	1.00	8/13/19 4:06 AM	9/11/19 11:43 PM	29:19:36	715.60	8/6/19 8:06 PM	9/23/19 8:10 AM	47:12:3	1140.05
16	8/7/19 6:49 AM	8/8/19 11:19 PM	1:16:29	40.48	8/8/19 11:54 PM	8/10/19 4:45 PM	10:16:51	256.85	8/7/19 6:49 AM	8/19/19 4:45 PM	12:9:55	297.92
17	8/8/19 5:24 PM	8/13/19 1:53 PM	4:20:28	116.47	8/14/19 2:31 PM	8/14/19 8:03 PM	0:5:31	5.52	8/8/19 5:24 PM	8/14/19 8:03 PM	6:2:38	146.63
18	8/10/19 8:03 AM	8/12/19 2:44 AM	1:18:41	42.68	8/12/19 2:45 AM	9/3/19 8:48 AM	22:6:2	534.03	8/10/19 8:03 AM	9/3/19 8:48 AM	24:0:44	576.73
19	8/10/19 9:43 AM	8/12/19 12:17 PM	2:2:34	50.57	8/12/19 8:40 PM	8/14/19 11:35 AM	1:14:54	38.90	8/10/19 9:43 AM	8/22/19 12:10 PM	12:2:27	290.45
20	8/10/19 11:06 AM	8/13/19 9:52 AM	2:22:46	70.77	8/15/19 10:29 PM	8/20/19 1:21 PM	4:14:51	110.85	8/10/19 11:06 AM	8/20/19 1:21 PM	10:2:14	242.23
21	9/2/19 3:10 AM	9/6/19 7:15 AM	4:4:4	100.07	9/6/19 12:00 AM	9/13/19 11:47 AM	7:11:47	179.78	9/2/19 3:10 AM	9/18/19 3:52 AM	16:0:41	384.68
22	9/2/19 10:13 AM	9/3/19 8:17 AM	0:22:3	22.05	9/4/19 12:39 PM	11/7/19 9:04 AM	63:20:24	1532.40	9/2/19 10:13 AM	11/7/19 9:04 AM	65:22:51	1582.85
23	8/15/19 9:46 AM	8/15/19 9:52 AM	0:0:6	6.00	8/15/19 9:38 PM	8/29/19 3:08 PM	13:17:30	329.50	8/15/19 9:46 AM	8/30/19 12:10 AM	14:14:24	350.40
24	9/5/19 5:05 AM	9/5/19 7:30 AM	0:2:24	2.40	9/5/19 11:00 AM	9/30/19 2:39 PM	25:3:38	603.63	9/5/19 5:05 AM	9/30/19 9:36 PM	25:16:30	616.50
25	9/10/19 2:47 AM	9/10/19 10:59 AM	0:8:12	8.20	9/11/19 3:12 AM	9/11/19 3:54 AM	0:0:41	0.68	9/10/19 2:47 AM	9/10/19 10:59 AM	0:8:12	8.20
26	9/10/19 11:36 PM	9/10/19 11:36 PM	0:0:0	0.00	9/11/19 12:31 AM	9/11/19 12:32 AM	0:0:0	0.00	9/10/19 11:36 PM	9/11/19 9:55 PM	0:22:18	22.30
27	9/11/19 5:53 AM	9/11/19 5:53 AM	0:0:0	0.00	9/16/19 12:51 AM	9/23/19 3:12 AM	7:2:21	170.35	9/11/19 5:53 AM	9/23/19 3:12 AM	11:21:19	285.32
28	9/11/19 9:31 AM	9/12/19 5:21 AM	0:19:50	19.83	9/12/19 10:48 AM	10/28/19 2:19 PM	46:3:30	1107.50	9/11/19 9:31 AM	10/28/19 2:19 PM	47:4:47	1132.78
29	9/11/19 4:06 PM	9/11/19 6:43 PM	0:2:36	2.60	9/11/19 11:08 PM	10/14/19 12:15 PM	32:13:6	781.10	9/11/19 4:06 PM	10/17/19 9:48 PM	36:5:41	869.68
30	9/12/19 6:17 PM	9/12/19 6:23 PM	0:0:5	5.00	9/12/19 6:26 PM	10/9/19 2:11 PM	26:19:44	643.73	9/12/19 6:17 PM	10/14/19 5:05 AM	31:10:48	754.80

Time	#Defects	Avg	Median	SD	Range
Jul-19	42	661.47	392.53	693.07	1915.68
Aug-19	50	568.96	463.83	368.88	993.42
Sep-19	56	600.75	500.59	498.72	1574.65
Overall	148	610.39	423.89	519.63	1917.31



- In an attempt to study trends we sampled data of incoming defects during the period of Jul-Sept 19 and plotted a baseline (10x defects from each month)
- We broadly measured time-taken for defects across Tier-1 technical-support, Core Engineering/Escalation support & the Overall turn-around times* for defects (which includes an RCA & release)
- Since all measurements are in terms of 'hours' this data will be classified as ***CONTINUOS***

* Overall turn-around time is not additive of the time defects spend in Tier-1/Escalation support (it involves additional R&D practices).

Measure phase – SQL & Measurement Error

Sigma Quality Level (SQL)	
Overall turn-around time	Defect incoming rate
• Defect opportunities that can cause the overall turn-around time to increase = 5 • Defects sample set (Jul-Sept 2019) = 30 • Overall defect opportunities = 5 * 30 = 150 • Actual defects = 46 • Defect per opportunity rate = 46/150 * 100 = 30.67 • Defects per million opportunities = 306667 **SQL = 2.0**	• Defect opportunities = 1 • Actual defect threshold = 5 • Total possible defects per day = 5 • Total actual defects (averaging data = 148/92 days) = 1.608 • Defect per opportunity rate = 1.608/5 * 100 = 32.16 • Defects per million opportunities = 32173.91 **SQL = 2.0**

Issue	1st	2nd	Did you agree
1	good	good	yes
2	good	good	yes
3	good	good	yes
4	good	good	yes
5	good	good	yes
12	good	bad	no
13	good	good	yes
14	good	good	yes
15	good	good	yes
16	good	good	yes
21	bad	good	no
22	good	good	yes
23	good	good	yes
24	good	good	yes
25	good	good	yes
Totals			
Percent good	93.33333333	93.33333333	
Percent bad	6.666666667	6.666666667	
Percent agreed			0.866666667
P observed	0.866666667		
P chance	0.124444444		
K (repeatability)	0.847715736		

- **Measurement Error**: There are potential measurement errors since this involves reading through several updates on defects & using discretion (based on certain deterministic criteria) to tabulate data.

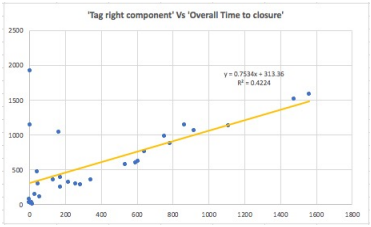
I randomly picked some defects between two separate dates & calculated the Kappa index for repeatability as 0.8477 -> 0.85 (which meets the criteria for measurement of >= 0.85)

NOTE – The sample set was 50% of the original set (this is a manual laborious process & hence the smaller comparison set)

Analyze phase – Scatter-plots and Correlation

Issue	Define Problem (hours)	Upload Logs (hours)	Tag right component/category (hours)	Time taken to close bug after RCA (hours)	Overall Time to closure (hours)
1	2.28	0.57	0.60	35.23	72.70
2	0.00	0.00	3.73	414.27	1925.52
3	4.07	1.02	6.10	584.82	1137.40
4	1.88	0.47	164.68	911.68	1041.30
5	14.31	3.58	11.35	11.85	29.23
6	14.05	3.51	44.45	421.78	463.10
7	1.52	0.38	1479.05	52.08	1509.20
8	1.88	0.47	7.48	2.35	9.83
9	51.84	12.96	216.45	111.97	321.95
10	4.80	1.20	56.53	82.23	104.43
11	15.43	3.86	134.48	216.65	350.92
12	14.40	3.60	754.70	351.75	984.95
13	4.80	1.20	595.17	7.18	600.12
14	18.40	4.60	918.97	149.22	1059.58
15	0.80	0.20	866.60	424.45	1140.05
16	32.39	8.10	257.43	41.07	297.92
17	93.17	23.29	30.17	141.12	146.63
18	34.15	8.54	534.05	42.70	576.73
19	40.45	10.11	47.28	251.55	290.45
20	56.61	14.15	171.47	131.38	242.23
21	80.05	20.01	172.53	204.90	384.68
22	17.64	4.41	1560.78	50.45	1582.85
23	4.80	1.20	341.25	20.90	350.40
24	1.92	0.48	607.13	12.87	616.50
25	6.56	1.64	16.90	7.52	8.20
26	0.00	0.00	0.92	22.30	22.30
27	0.00	0.00	285.32	114.97	285.32
28	15.87	3.97	1112.95	25.28	1132.78
29	2.08	0.52	785.52	88.58	869.68
30	4.00	1.00	643.80	111.07	754.80

Correlation co-efficient calculations b/n X's & Overall Time to closure		
X's	Overall Time to closure (R)	Overall Time to closure (R^2)
Define problem	-0.275883463	0.076111685
Upload logs	-0.275883463	0.076111685
Tag right component/category	0.649959013	0.422446719
Time taken to close bug after RCA	0.409240798	0.167478031



When gathering data & tabulating top contributors adding delay to the overall resolution of a defect, I wanted to understand their correlation.

Plotting the correlation co-efficient:

1. The data beside indicates the r-value highest for 'Tag right component/category' (vs 'Overall Time to closure')

2. Hence we plot the corresponding scatter plot & derive the equation from Linear Regression:

$$y = 0.7534x + 313.3$$

3. Also of significance was 'Time taken to close bug after RCA'

NOTE - Extreme corner table from Slide-6/Define Phase – Data collection

Improve phase - SQL & Hypothesis Test

Sigma Quality Level (SQL)	
Overall turn-around time	Defect incoming rate
• Defect opportunities that can cause the overall turn-around time to increase = 5 • Defects sample set (Oct-Dec 2019) = 25 • Overall defect opportunities = 5 * 25 = 125 • Actual defects = 37 • Defect per opportunity rate = 37/125 * 100 = 29.6 • Defects per million opportunities = 296000 **SQL = 2.0** *No change from baseline ☹*	Did not have a software GA release in the last couple of months. SQL remained at 2.0 as well

Overall Time to closure of defects		
	Before	After
Mean	610.39	596.28
Sample	30	25
SD	519.63	549.56
Zvalue	0.097	
Pvalue	0.922582939	
Alpha	0.1	

Hypothesis Test: Since there was marginal improvement in the mean across data-sets before & after the process improvement, I ran the following test:

- Continuous -> Two-Sample -> Upper/Right-tail test

$H_0: \mu_1 \leq \mu_2$

$H_a: \mu_1 > \mu_2$

- Since p-value >> alpha at 0.1, 0.05 or 0.01 we fail to Reject the Null hypothesis & therefore state the new mean is not statistically different from the previous!

3.9.2.1 Summary/Process Flow Diagram (DMAIC)

Process Improvement Project – Improve turn-around time & time-to-resolution for customer found defects

Process Owner: Sharat Sripada

Define – 9/25 → Measure – 10/10 → Analyze – 11/1 → Improve – 12/1 → Control (ongoing)

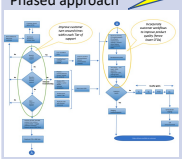
Problem Statement:

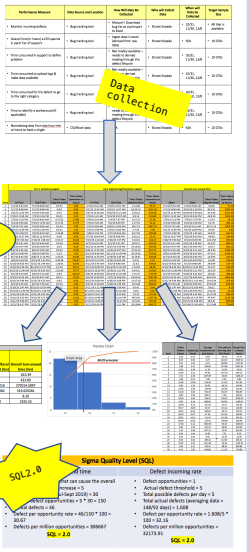
Improve support, Engineering turn-around time for customer found defects(CFDs) + improve overall quality of the product to lower incoming rate of defects

Goals:

- ~10% reduction in turn-around time/mean time to defect resolution (in terms of hours)
- ~5% lower incoming CFD rate – bracing some customer practices into our existing

Phased approach

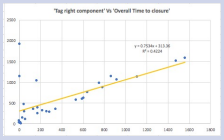


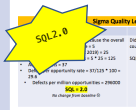


Correlation co-efficient calculations b/n X's & Overall Time to closure

X's	Overall Time to closure (R)	Overall Time to closure (R^2)
Define problem	-0.275883463	0.076111685
Upload logs	-0.275883463	0.076111685
Tag right component/category	0.649959013	0.422446719
Time taken to close bug after RCA	0.409240798	0.167478031

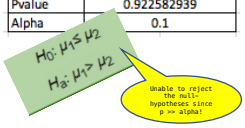
- High correlation co-efficient between 'Tag right component/category' and 'Overall Time to closure of defects' among all known inputs/Xs.
- Scatter plot below shows the correlation between the variables the corresponding equation from linear regression is shown:

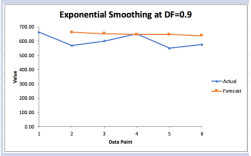




- SQL remained at 2.0 after the process change which proves no significant benefits were gained

Overall Time to closure of defects	Before	After
Mean	610.39	596.28
Sample	30	25
SD	519.63	549.56
Zvalue	0.097	
Pvalue	0.922582939	
Alpha	0.1	

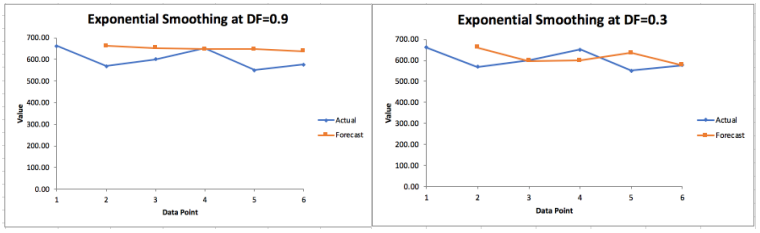




- Pilot for Phase-1 of the project (lowering overall CFD turn-around time) shows lower forecast for Jan-2020 which is encouraging.
- In pipeline for Q1 FY 2020, pilot for Phase-2 which also aims to reduce the overall count of CFDs.

3.9.3 Conclusion and future work

Control phase – Time series, Next-steps & conclusion



Time series analysis:

- As a final step, we predict the mean time for CFD resolution in Jan-2020 using an exponential smoothing time series method shown alongside.
- The forecast itself is encouraging, showing the mean time to resolution of a CFD:
  - DF=0.3 -> 576
  - DF=0.9 -> 637

Initial Goal of 10% reduction in time to resolution of defects is met!

Next steps & conclusion

- This is a pilot but just the sheer volume of data & all the variables associated with a complex eco-system are overwhelming – hence we would spend some effort in automating the data-collection process itself
- Continue driving process improvements & make necessary changes to get meaningful results

Goal - Streaming an efficient pipeline for flow of defects with the aim itself being to bring the rate of defects down.

4.0 Summary

Overall, the program has been of great value addition to my career and following are some of the key takeaways:

- While I had experience writing code in Python adopting it for data-science, information visualization, machine-learning and deep-learning problems using libraries like NumPy, Pandas, Matplotlib, Seaborn, NLTK, Sci-kit learning, SciPy, TensorFlow, Keras/PyTorch was truly fascinating
    - Learning the R programming language to solve problems in this realm has been another key facet to the course. I can fully translate code in Python to R or vice-verse
- Exposure to Kaggle competitions, datasets and how to compete on leaderboard
- Deep knowledge into statistics (descriptive and inferential)
- Process improvements using DMAIC
- Natural language processing (NLP) techniques like LSA, LDA, TF-IDF, Markov Models, Tokenizers, Analyzers, POS-tagging etc.
    - Building the Marvel search engine using BM-25 and USE as an extension to NLP in the text mining domain was riveting
    - Natural language understanding using sentiment analysis, custom analyzers, entity analysis, word embedding etc.
- Machine-learning: Supervised and Unsupervised learning techniques
    - Unsupervised – Dimensionality reduction using PCA, K-means clustering and Hierarchical clustering
    - Supervised – SVM, Decision Trees, Regression, Logistic Regression, Naïve Bayes classifiers, KNN

4.1 References

4.1.1 Textbooks and Readings

- **Discovering Statistics** by Daniel T. Larose - 3rd edition
- **Understanding Variation** - The Key to Managing Chaos, 2nd edition By Donald J. Wheeler; SPC Press
- Hoffer, J. A, Ramesh, V., & Topi, H. (2016). **Modern database management** (12th ed.). New
- York, NY: Pearson.
- **Introduction to Data Science** (2017), by Jeffrey S. Saltz & Jeffrey M. Stanton.
- Pang-Ning Tan, Michael Steinbach, and Vipin Kumar (2005) **Introduction to Data Mining**.
- Tom Mitchell (1997) **Machine Learning**
- Brett Lantz (2015) **Machine Learning with R** (second edition).
- Stanton (2017), **Reasoning with Data: An Introduction to Traditional and Bayesian Statistics Using R**
- Bird, S., Klein, E., & Loper, E. **Natural language processing** with Python
- Jurafsky, D., & Martin, J. H. **Speech, and language processing** (3rd ed. draft).
- Pang-Ning Tan, Michael Steinbach, and Vipin Kumar (2005) **Introduction to Data Mining**
- Yau, N. (2011). Visualize this: **The Flowing Data guide to design, visualization, and statistics**. Wiley Publishing.
- Yau, N. (2013). **Data points: Visualization that means something**. Wiley Publishing.
- **Python for Everybody**: https://www.py4e.com/book (Python version)
- Miller, Thomas W., **Modeling Techniques in Predictive Analytics with Python and R**, Pearson, 2015.
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville, **Deep Learning (DL)**, MIT Press, 2016
- James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani, **An Introduction to Statistical Learning with Applications** in R, Springer, 2013
- [Marvel-Dialog-NLP] https://github.com/prestondunton/marvel-dialogue-nlp/tree/master/data

- [MCU-Script-PDF] https://bulletproofscreenwriting.tv/marvel-studios-screenplays-download/
- [1] Search Engine drives retail site amazon.com where given a query, the backend sifts through billions of products and ranks by relevance items customers are likely to buy
- [2] Aligning to a competency related to language analytics and had courses - Natural Language processing (NLP) and Text Mining

4.1.2 Source Repository

https://github.com/sharatsv/MS-DataScience

4.1.3 Resume

https://www.linkedin.com/in/sharat-s-6a14385/