TWEETS CHALLENGE REPORT

Team 3s

Table of Contents

Executive Summary

Company

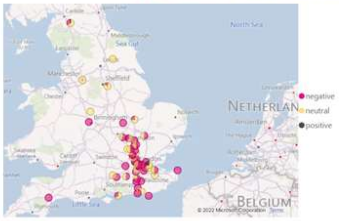
Thameslink is part of Govia Thameslink Railway, the largest train operating company in the UK. The company always looks to improve every aspect of customer service experience while also believes in doing business in a responsible way.

Problem

- Increase vehicle availability
- Improve customer satisfaction
- Optimize business operations and costs

Facts (original dataset)

16.92K	10.61K	9022
Total Number of Tweets	Negative Tweets	Tweets about delays



Solution

A dashboard connected to Twitter API and stream public Tweets in real-time

- Classify live tweets into negative/neutral/positive sentiment
- Classify tweets mentioned delays
- Provide insights and analytics over time

Team 3s

- Rashmi Carol D'souza - s0581358
- Ekin Pomay Polat - s0581497
- Quynh Pham - s0581655

Milestones

- **Project Starts** Oct 25, 2022
- 1 **Milestone 1** Target hierarchy, project roadmap
- 2 **Milestone 2** Sentiment analysis model, connection to Twitter API
- 3 **Milestone 3** Topic classification model, connection to database
- 4 **Milestone 4** Complete pipeline and dashboard
- **Project Ends** Jan 03, 2023

Project Process

Project Conception
Project Initiation
Product Conceptualization

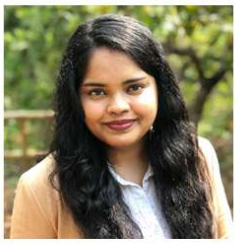
Project Execution & Performance Evaluation
Status & Tracking
Quality Deliverables
Risk Assessment & Solutions
Retrospectives

0% Start — 100%

Project Definition & Planning
Scope & Goal Setting
Target Hierarchy
Schedule & Roadmap
Document Setup

Project Close
Final Product Delivery
Technical Manual & Instruction
Project Report

Our Team



RASHMI D'SOUZA
Proxy PO / Developer

EKIN POMAY POLAT
Developer

QUYNH PHAM
Scrum Master / Developer

Work Breakdown

No	Topic	Rashmi Carol D'souza	Ekin Pomay Polat	Quynh Pham
1	Business Understanding	33%	33%	33%
2	Data Understanding	45%	45%	10%
3	Data Exploration	10%	80%	10%
4	Data Preparation	10%	10%	80%
5	Modelling and Evaluation for Sentiment Analysis	10%	10%	80%
6	Modelling and Evaluation for Delay Classification	10%	80%	10%
7	Building Pipeline	80%	10%	10%
8	Building Dashboard	10%	10%	80%
9	Final Presentation	33%	33%	33%
10	Report Writing	33%	33%	33%

Business Overview

Thameslink is part of Govia Thameslink Railway, the largest train operating company in the UK. The company always looks to improve every aspect of customer service experience while also believes in doing business in a responsible way – serving passengers and benefiting the communities around stations.

Stakeholders

- Thameslink
- Passengers
- Local communities
- Twitter
- Tweets users
- 3s team

As a company, Thameslink constantly strives for providing comfortable, reliable and seamless train service

- Improve the reliability and capacity of our trains
- Make staff more available on trains and in stations
- Use new technology to keep passengers informed
- Improve the on board and station environment

Opportunities & Challenges

What are the benefits of this project?

- **Increase vehicle availability**, which is the indicator showing the extent to which the vehicle fleet is available for revenue-earning work
  - Strive for having a system where all trains, infrastructure and technology run at highest capacity 24/7 without any failure
  - Fix any occurred problem as quick as possible & send vehicle back to service
- **Improve customer satisfaction** by improving company's reaction time to issues
- **Optimize business operations and costs**
  - By collecting & analyzing live tweets, company is quickly notified of problems in real-time
    - Reduce the need to send staff out to investigate any train issues or have a team done inspection on a regular basis
    - Low cost (Twitter is free)
    - Increase staff availability for other tasks

Challenges that might affect the performance of our models in this project

- Current situation between Twitter and Elon Musk
- Not enough data to train the machine learning model
- Some common challenges for NLP such as synonyms, phrases, irony, ambiguity etc.

Project Overview

Customer: Steven from Thameslink

Vision: A simple and efficient dashboard that is connected to Twitter API to stream public Tweets in real-time and provides brief analytics and assessment of current concerns on which further inquiry, solutions, or initiatives can be carried out

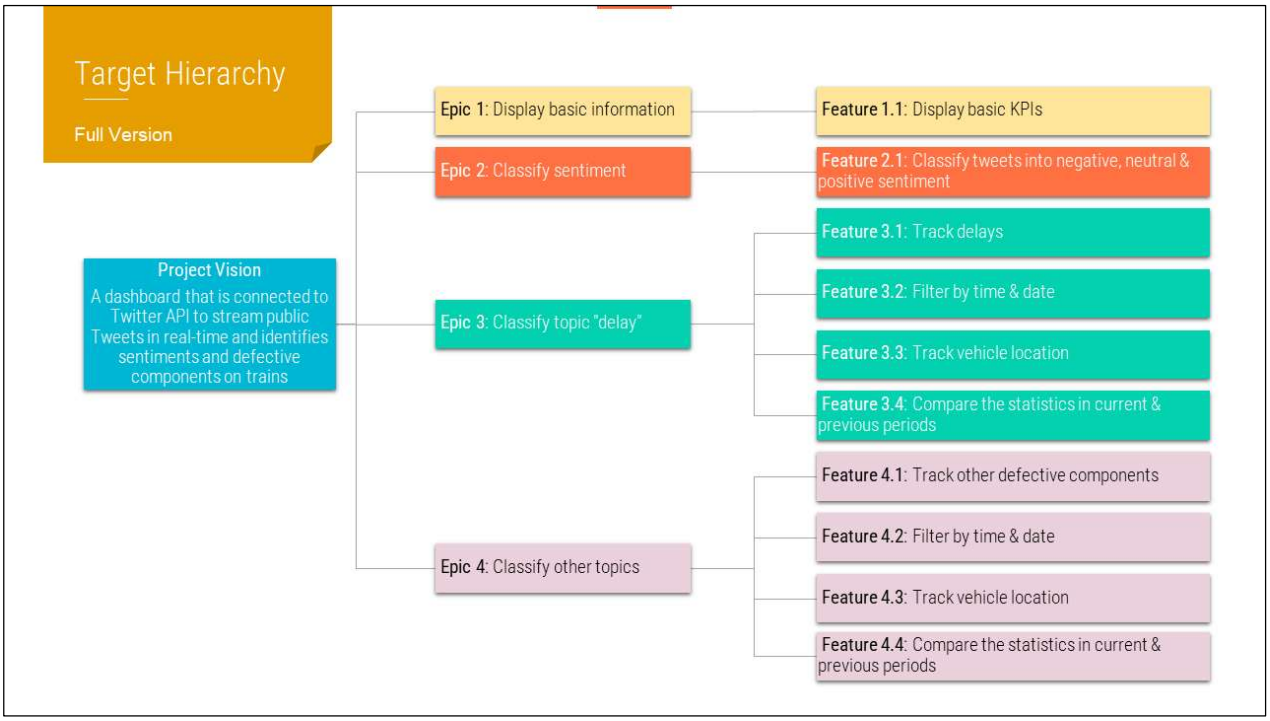
Project Goal:

- Classify live Tweets into negative/neutral/positive sentiment
- Classify Tweets mentioned delays
- Create a dashboard connected to Twitter API to stream public Tweets in real-time and provide statistical analytics over time to assist managers in making decisions

Project Duration: Oct 25, 2022 - Jan 3, 2023

Target Hierarchy

- Full Target Hierarchy
- Reduced Target Hierarchy with Priority
- Target Hierarchy at the beginning of the Project
- Final Target Hierarchy at the end of the Project
- Demonstration of each epic



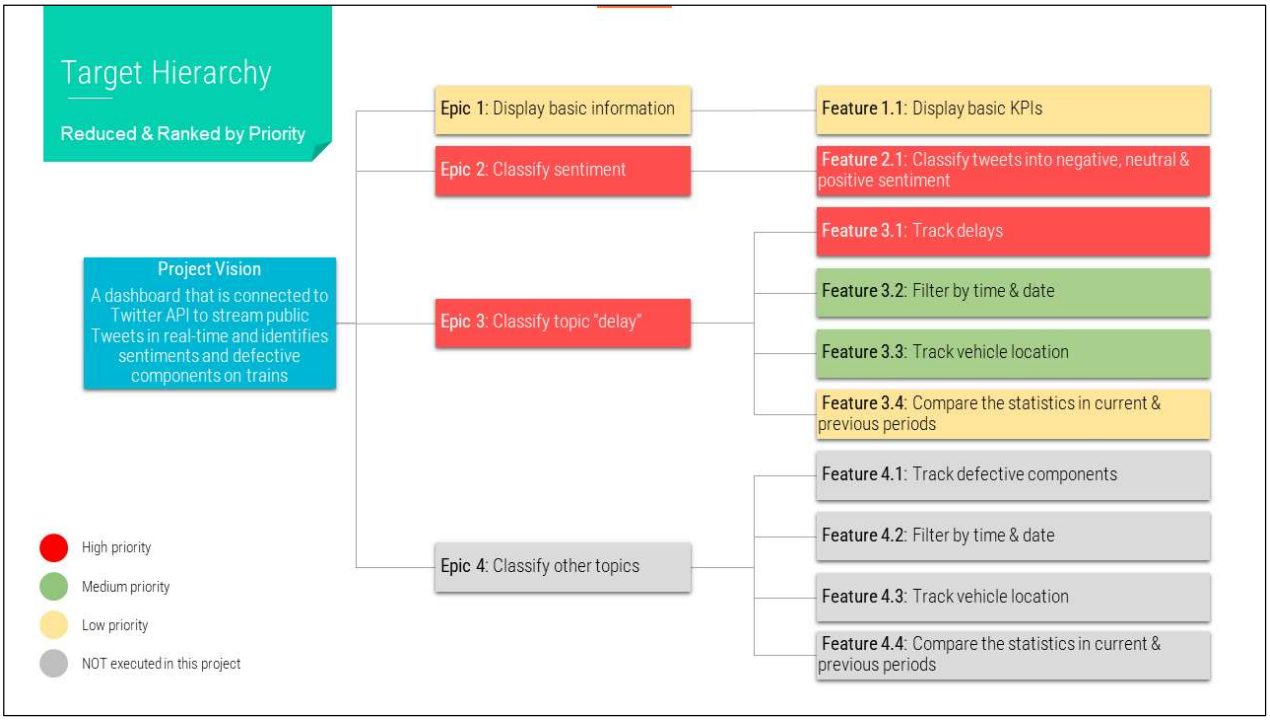
This is the full version of Target Hierarchy. After considering the project goal, we divide our project into 4 different epics.

Epic 1. Display Basic Information

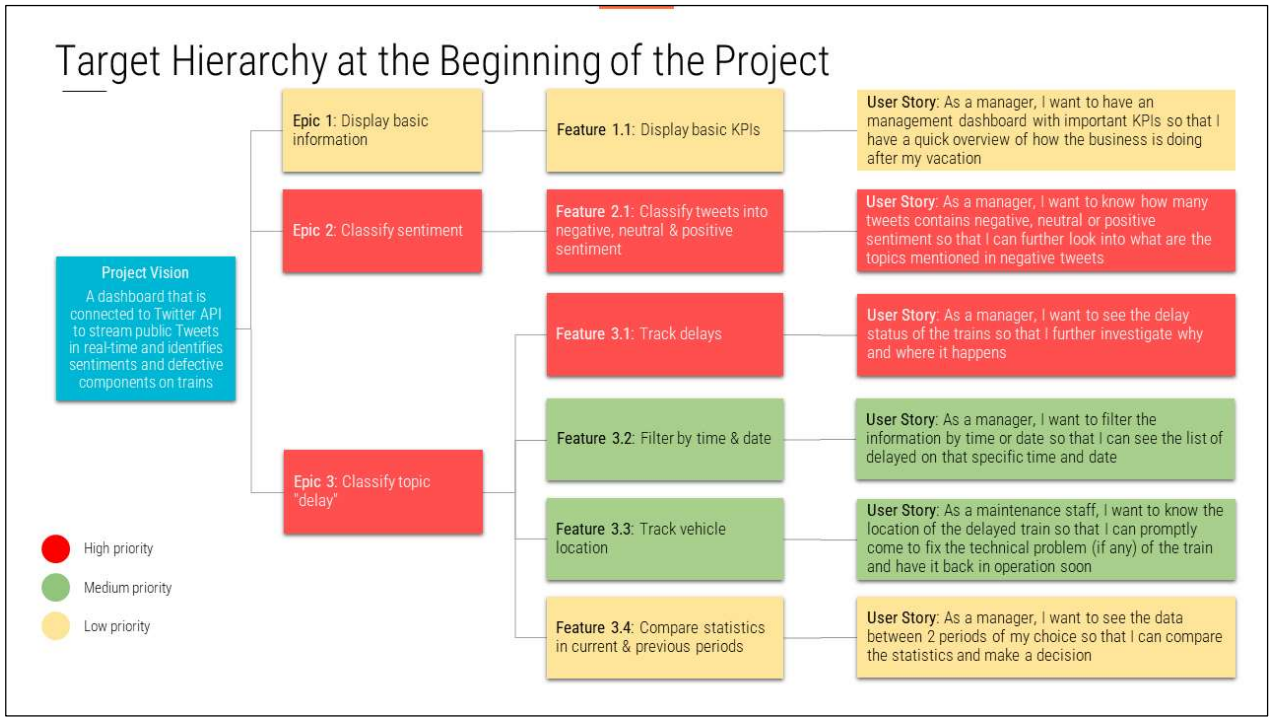
Epic 2. Classify Sentiment

Epic 3. Classify topic “delay”

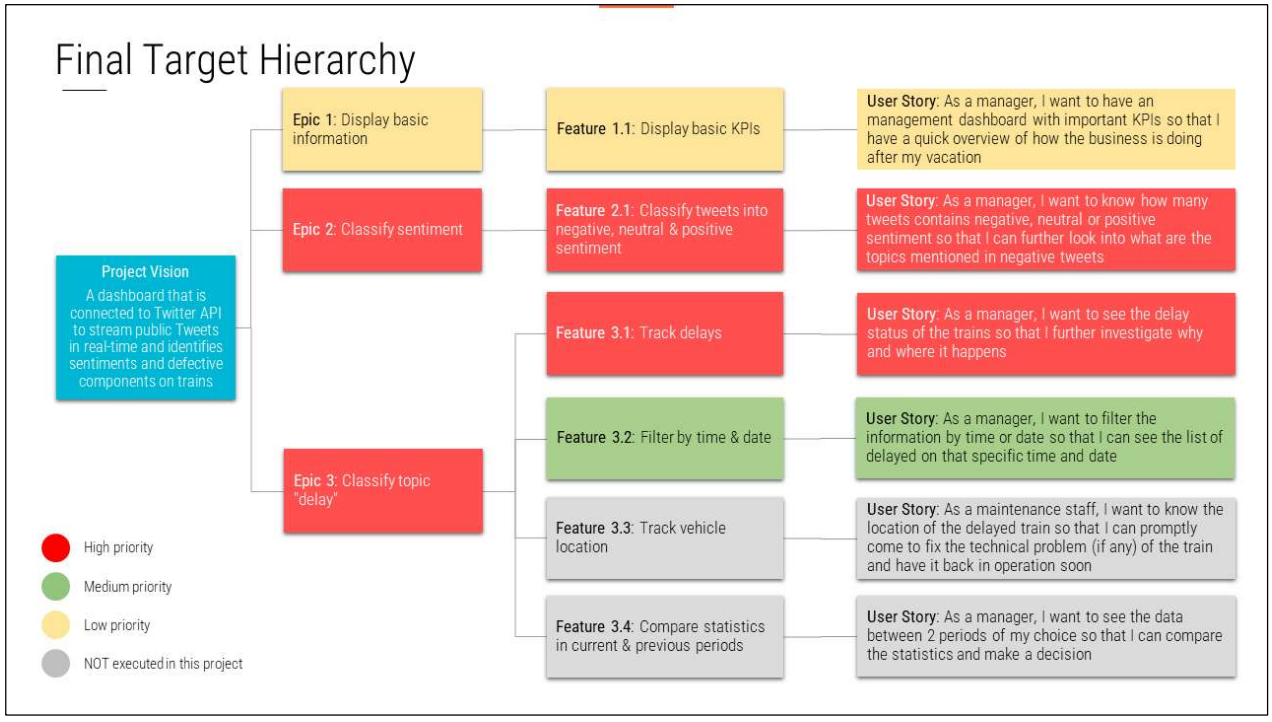
Epic 4. Classify other topics



Due to the time constraints and the lack of personnel in our team, we decide to prioritize our target hierarchy as high, medium and low. With the approval from the customer, Epic 4 *(Classifying other topics)* is excluded from the scope of this project and will not be executed.



This is our Target Hierarchy at the start of this project with user stories approved by our customer. In this project, Epic 2 *(Classify sentiment)* and Epic 3 *(Classify topic delay)* are the main focuses. The user stories of prioritized features are created from the management perspective.



However, as the project progresses, we decide to drop Feature 3.3 *(Track vehicle location)* and Feature 3.4 *(Compare statistics in current and previous periods)* due to the lack of GPS information and past data obtained from Twitter API.

EPIC 1

Display basic information

Feature 1.1

Display basic KPIs

User Story 1.1.1

As a manager, I want to have an management dashboard with important KPIs so that I have a quick overview of how the business is doing after my vacation

Acceptance Criteria

- Dashboard has a title, correct dimension, basic layout for every features
- Dashboard has Thameslink logo and relevant images
- Details include total number of delays, other topics and tweets

Definition of Done (DoD)

- Information display correctly on the dashboard
- Acceptance criteria is met
- Sign off by PO
- Approved by Client

EPIC 2

Classify sentiment

Feature 2.1

Classify tweets into negative, neutral & positive sentiment

User Story 2.1.1

As a manager, I want to know how many tweets contains negative, neutral or positive sentiment so that I can further look into what are the topics mentioned in negative tweets

Acceptance Criteria

- Pie chart of sentiment classification
- Display total numbers of negative, neutral, positive tweets
- Have general details (Thameslink logo and relevant images/buttons)

Definition of Done (DoD)

- Code builds with no error
- Testing is complete
- Information display correctly on the dashboard
- Acceptance criteria is met
- Sign off by PO
- Approved by Client

EPIC 3

Classify topic "delay"

Feature 3.1

Track delays

User Story 3.1.1

As a manager, I want to see the delay status of the trains so that I further investigate why and where it happens

Acceptance Criteria

- List of top 10 trains in delay
- Details include total number of tweets, location, time & date
- Have general details (Thameslink logo and relevant images/buttons)

Definition of Done (DoD)

- Code builds with no error
- Testing is complete
- Information display correctly on the dashboard
- Acceptance criteria is met
- Sign off by PO
- Approved by Client

Feature 3.2

Filter by time & date

User Story 3.2.1

As a manager, I want to filter the information by time or date so that I can see the list of delayed on that specific time and date

Acceptance Criteria

- Have a drop down filter box with parameters 'hour' and 'date'
- Automatically update list of delayed train when filter is applied
- Have general details (Thameslink logo and relevant images/buttons)

Definition of Done (DoD)

- Code builds with no error
- Testing is complete
- Information display correctly on the dashboard
- Acceptance criteria is met
- Sign off by PO
- Approved by Client

EPIC 3

Classify topic "delay"

Feature 3.3

Track vehicle location

User Story 3.3.1

As a maintenance staff, I want to know the location of the delayed train so that I can promptly come to fix the technical problem (if any) of the train and have it back in operation as quickly as possible

Acceptance Criteria

- Have a map of the delayed trains
- Have a list of the nearest station
- Have general details (Thameslink logo and relevant images/buttons)

Definition of Done (DoD)

- Code builds with no error
- Testing is complete
- Information display correctly on the dashboard
- Acceptance criteria is met
- Sign off by PO
- Approved by Client

Feature 3.4

Compare the statistics in current & previous periods

User Story 3.4.1

As a manager, I want to see the data between 2 periods of my choice so that I can compare the statistics and make a decision

Acceptance Criteria

- Able to filter by:
  - hour
  - week / month / quarter / year
- Automatically update lists of delayed train in:
  - current period
  - previous period
- Display relative graph/diagram
- Have general details (Thameslink logo and relevant images/buttons)

Definition of Done (DoD)

- Code builds with no error
- Testing is complete
- Information display correctly on the dashboard
- Acceptance criteria is met
- Sign off by PO
- Approved by Client

PAGE 15

Project Roadmap

Kick-off Meeting

1 **Sprint 1: Business Understanding**
Target hierarchy, project roadmap

2 **Sprint 2: Data Understanding**
Dashboard mockup, high-level pipeline, data exploration

3 **Sprint 3: Data Preparation**
Data preparation, system design and tools to build our application

4 **Sprint 4: Model Iteration 1**
Model for sentiment analysis (1st iteration), accessing Twitter API

5 **Sprint 5: Model Iteration 2 & Evaluation**
Model for sentiment analysis (2nd iteration), model evaluation for sentiment analysis, ververless database to store data from Twitter API

6 **Sprint 6: Model Iteration 3**
Model for delay classification, proof of concept for local pipeline

7 **Sprint 7: Model Evaluation**
Model evaluation for delay classification, progress on pipeline, final dashboard

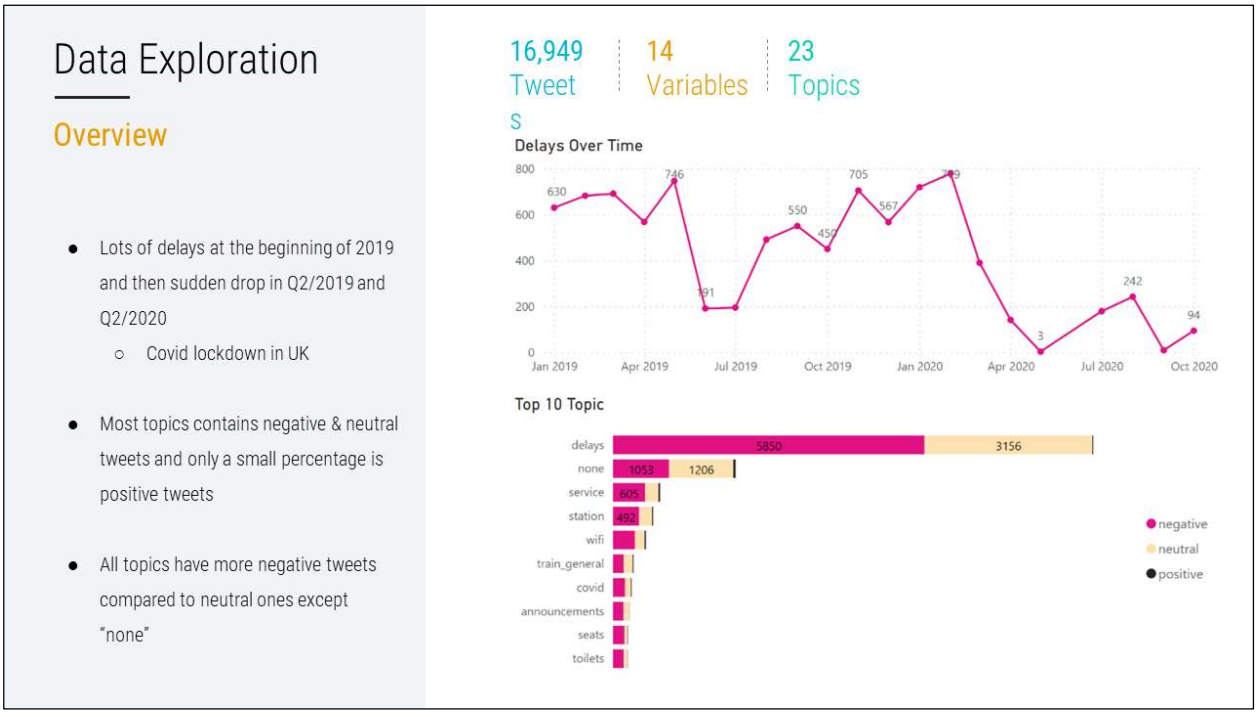
8 **Sprint 8: Deployment**
Moving to the cloud, dashboard update

9 **Sprint 9: Project Finalization**
Final presentation, demonstration of final dashboard

Project Ends

PAGE 18

Our project roadmap includes detailed goals and deliverables for each Sprint.



We explore the data based on our prioritized Epics.

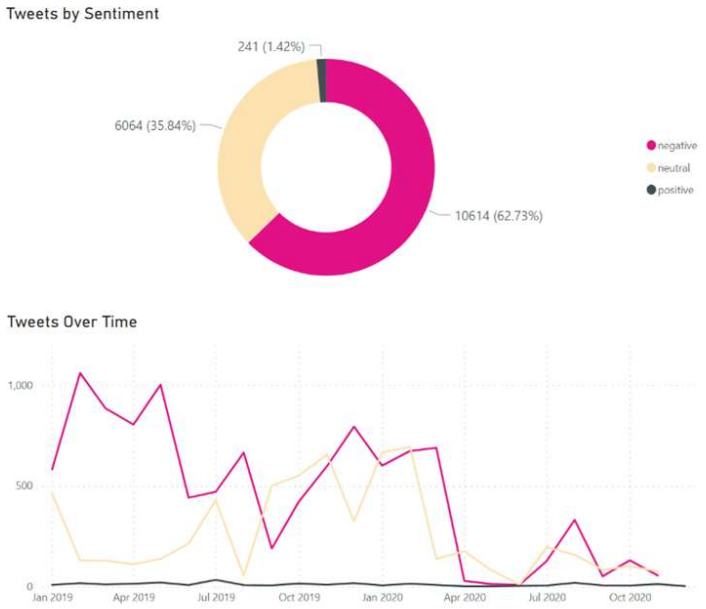
In the original dataset, there are 16,949 tweets with 14 variables and 23 different topics. When we examine the delays over time, it is observed that lots of delays are at the beginning of 2019. A Sudden drop is observed in both Q2 2019 and Q2 2020. The drop in Q2 2020 prompt us to take into consideration the Covid Lockdown in the UK.

Topics are arranged based on their frequency. It is clear that Topics mostly contain negative and neutral tweets and only a small percentage of positive tweets.

Data Exploration

Sentiment

- Imbalance sentiment
- Higher negative tweets in the first half year of 2019

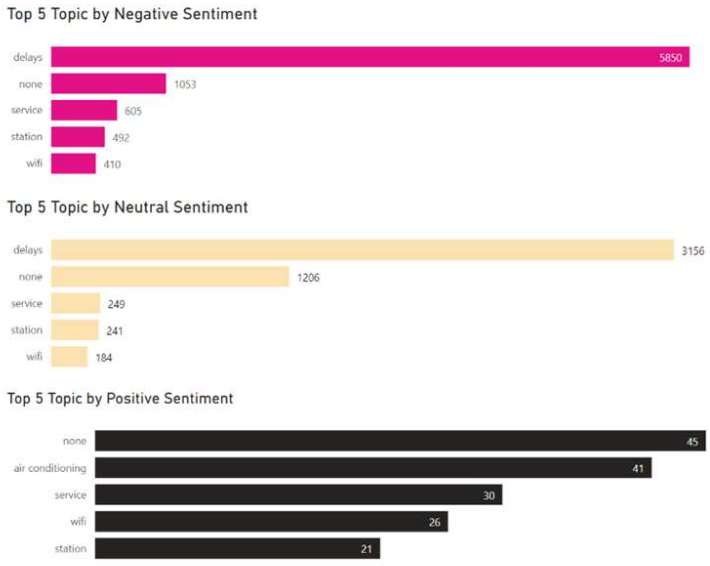


When we look at tweets by sentiment, we see an imbalance of sentiment and especially more negative tweets in the first half of 2019.

Data Exploration

Sentiment

- Explore topics by sentiment:
  - Tweets with **negative and neutral** sentiment has the **same topic**
  - Tweets with **positive** sentiment has somewhat **different topic**

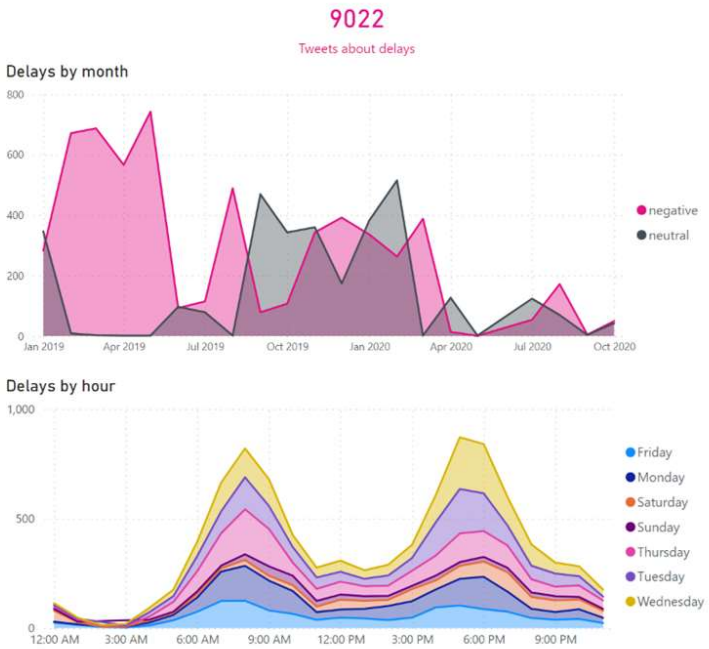


Exploration based on the sentiment reveals that the top 5 topics with negative and neutral sentiments are the same: delays, none, service, station, and Wi-Fi respectively. Tweets with positive sentiment have somewhat different topics. The most common topic after delay is “none”.

Data Exploration

Delays

- More delays in
  - winter
  - mornings and evenings
- Delays gets mostly negative & neutral tweets, some rare positive tweets which relates to customer services when delays happen

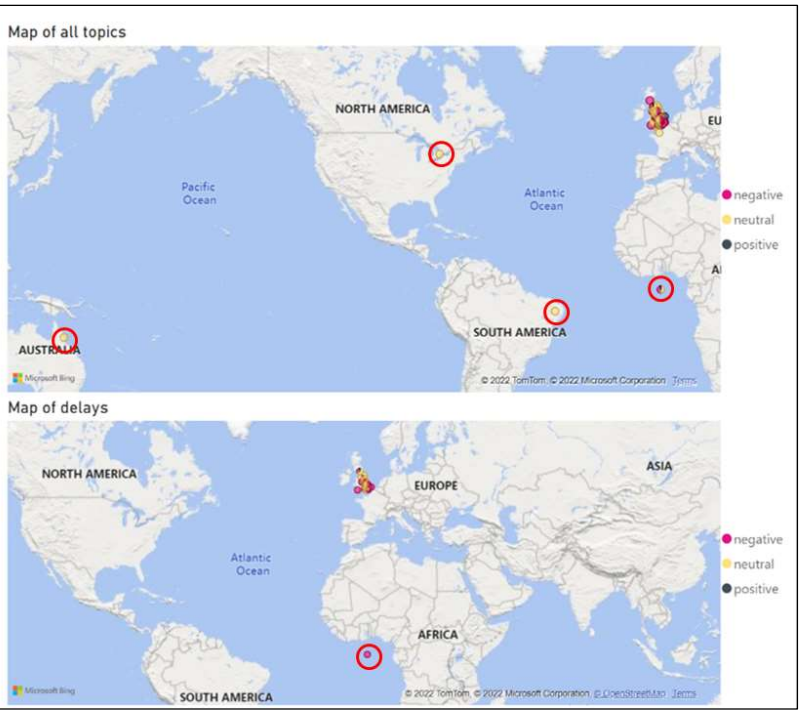


In terms of “delays”, we observe an increase in delays in winter, mornings, and evenings. The days with the highest "delays" in the mornings and evenings are Wednesdays and Tuesdays. When we look further, we observe that delays mostly receive negative and neutral tweets. There are a few positive tweets about delays, which refer to good customer service when delays occur.

Data Exploration

Delays

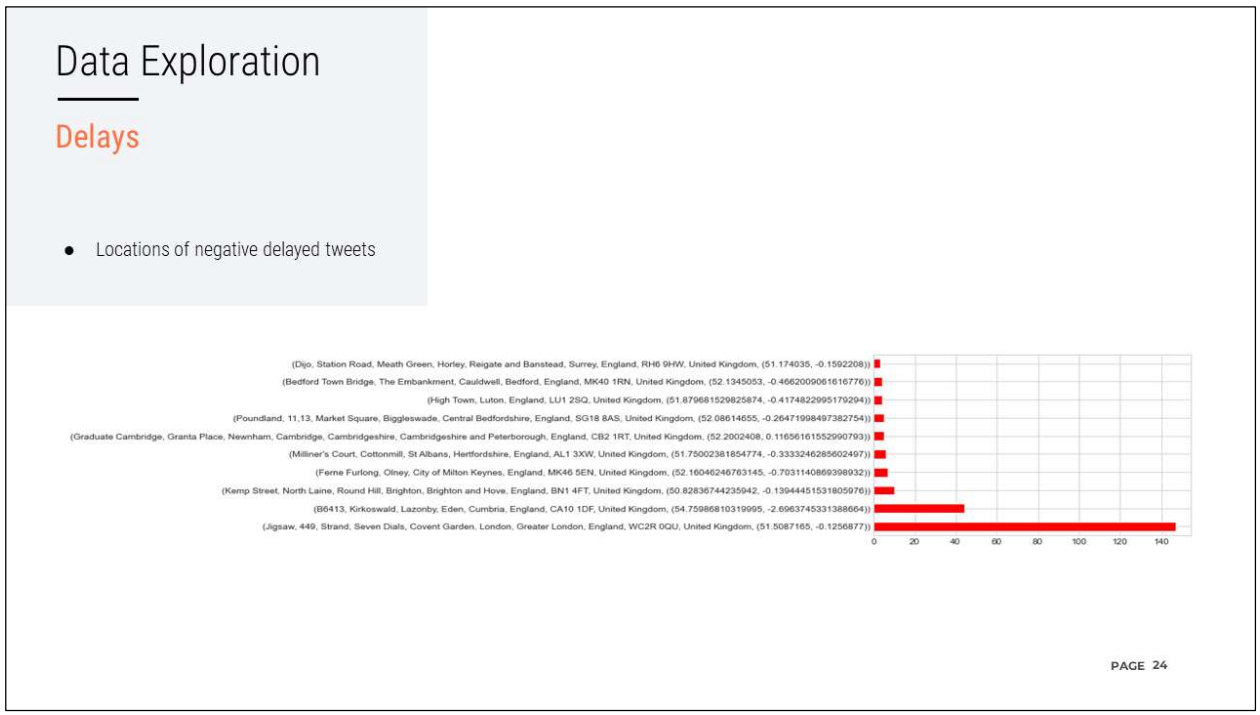
- Some GPS location is outside of UK and in other continent
- Delays **without GPS** info: **8493** tweets
  - 5564 negative tweets
  - 2915 neutral
  - 14 positive
- Delay **with GPS** info: **530** tweets (5%)
  - 287 negative
  - 241 neutral
  - 1 positive



When we check the map of all topics, we notice there are a few tweets coming from all over the world including North & South America, Australia and Africa. If we only look at the delays, we still see some data points from outside of the UK. We define those inexplicable GPS locations as outliers.

In terms of having GPS information, just 5% of "delays" tweets include the location point.

We explore the most tweeted locations in address detail based on their coordinates using “geopy” which is a Python client for several popular geocoding web services. The most tweeted location is **Jigsaw, 449, Strand, Seven Dials, Covent Garden, London, Greater London, England, WC2R 0QU, United Kingdom** with 247 delay tweets (149 negative, 90 neutral). We check manually and find out that the nearest station to this location is Charing Cross Railway that is a junction where 6 routes meet. The second most tweeted location is **B6143, Kirkoswald, Lazonby, Eden, Cumbria, England, CA 1DF, United Kingdom** with 131 delay tweets (45 negative, 85 neutral, 1 positive). The nearest stations are Lazonby & Kirkoswald.



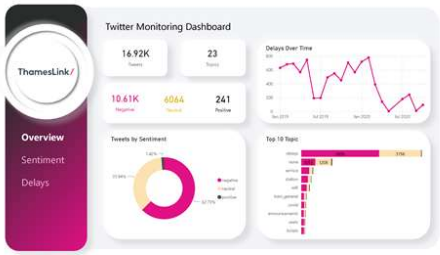
Here are the top 10 locations of negative delayed tweets can be found.

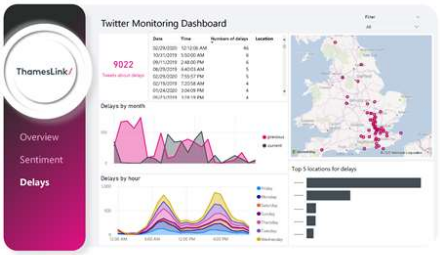
Summary of Data Exploration

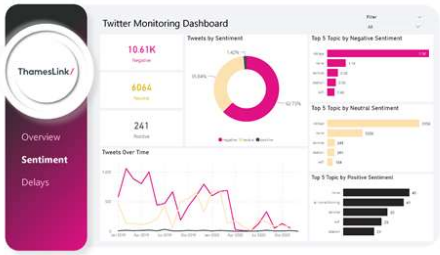
No	Problems	Possible Solutions	Risk Assessment
1	Missing data for GPS location		Might not get the GPS location from live tweets, which will affect how we identify where the delays happen and how to locate the nearest station
2	Wrong GPS location	Some GPS information shows the location in the middle of nowhere (ocean, other continent or countries) → flag as outliers & treat them later	
3	Imbalanced sentiment classes	Treatment for data imbalance before modeling phase (SMOTE, Random Undersampling, Random Oversampling…)	
4	Missing data in Q2 & Q4 of 2020 (June, November and December)		This might affect how we later build "Feature 3.4: Compare statistics in current & previous periods" in our dashboard
5	"None" is a big topic		This is the 2nd biggest topic (after delays) but it doesn't provide much insight into what are the real problems with the train services. It either isn't classified the topic correctly or needed to re-distributed to other topics.
6	Data is collected during Covid period		Consider the reliability (associate with Covid such as uncertainties, people emotional sensitivity, lockdown policy, etc) of this dataset to train our models, which will be used to make predictions in normal time
7	Some common challenges for NLP such as synonyms, phrases, irony, ambiguity…	New techniques (POS, BERT, word embedding, etc.) will help deal with this	

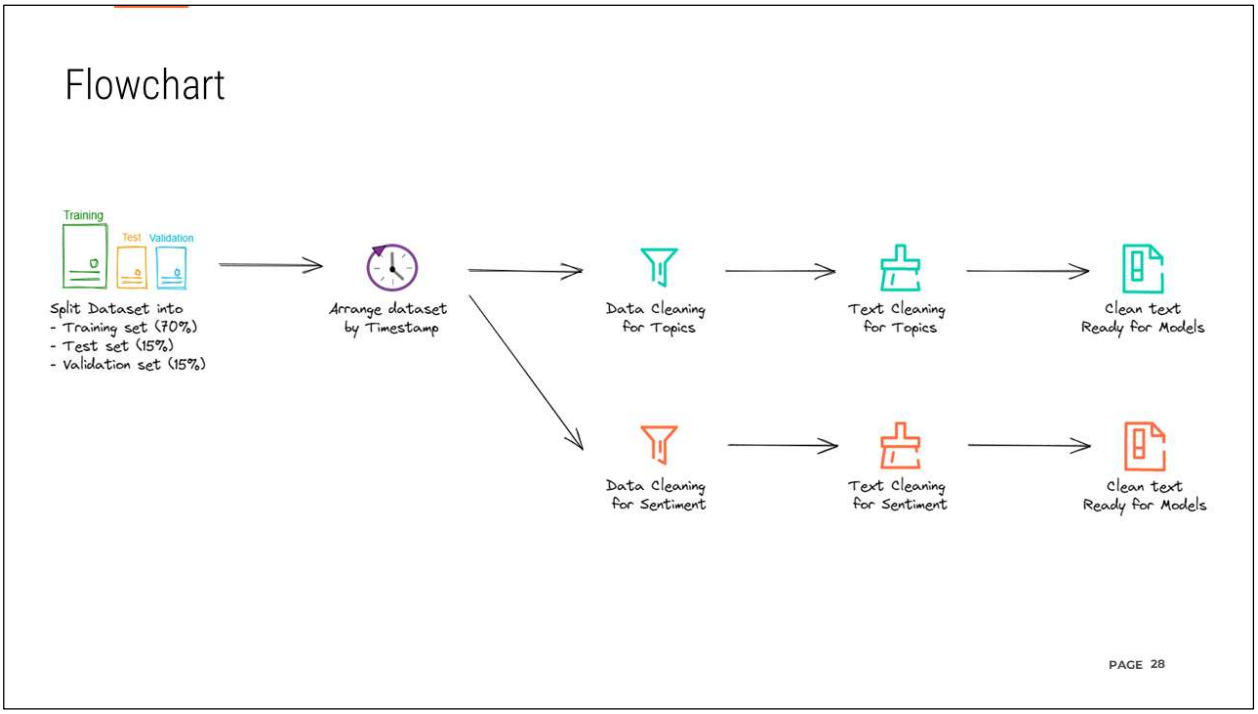
Dashboard Mockup

- Dashboard size 16:9 with title, Thameslink logo and color scheme:
  - white
  - dark grey (#444e55)
  - bright pink (#e21185)
- Side menu
- Basic layouts for all features









This is the flow chart for our data preparation process. The steps include:

1. Split data into training, test and validation sets with a ratio of 70% - 15% - 15% respectively
2. Sort data by timestamp so that all duplicates show up next to each other3. For this project, we are building two models - 1 model for sentiment analysis and 1 model for delay classification.
3. Both models require the same data preparation process including data cleaning and text cleaning before passing to the models.

Result - Splitting Data

1. **Split the dataset into:**
   - Training set (75%)
   - Test set (15%)
   - Validation set (15%)
   - Retain the same ratio of different categories (sentiment & topic) in these 3 subsets

```
X_main, X_test, y_main, y_test = train_test_split(X,y,test_size = 0.15, random_state=2, stratify = y) #split dataset into main & test set
X_train, X_val, y_train, y_val = train_test_split(X_main, y_main, test_size=0.15, random_state=2, stratify = y_main) #split main into train & validation sets
```

```
Length of training set:         12245
Length of test set:             2543
Length of validation set:       2161
```

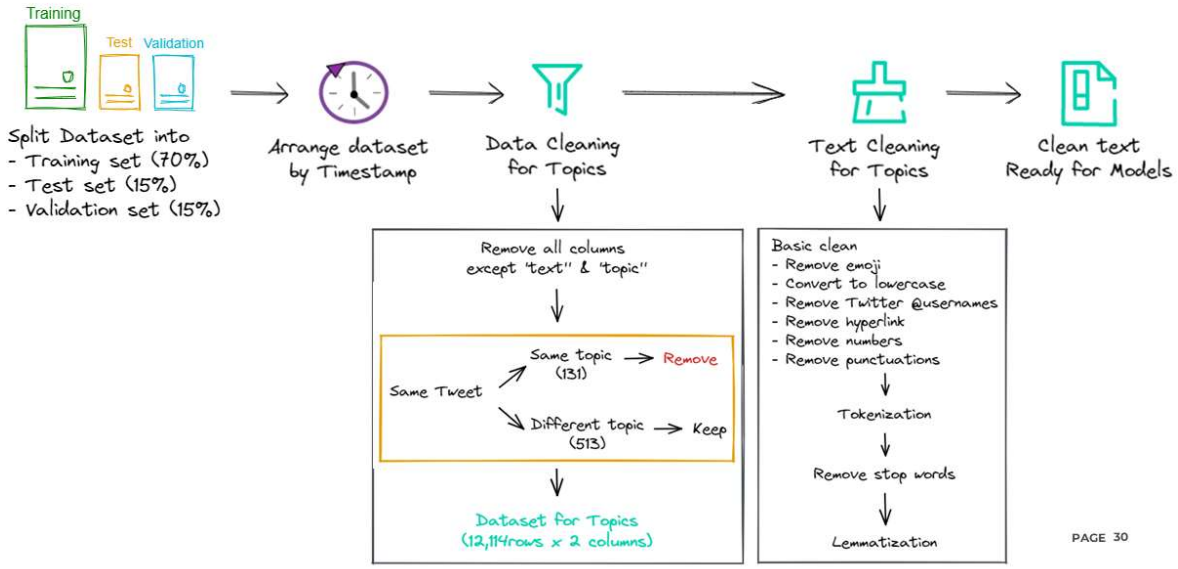
1. **Sort by timestamp**
   - Reason: easier to see the duplicates in next steps

	source_created_at	author_id	text	source	language	longitude	latitude	id	source_id	tweet_id	user_id	relevant	topic	ground_truth	sentiment
10484	2019-01-16 10:41:46	2819417231.0	@catherinerusse2 @TLRailUK The definitely know...	brandwatch	en-GB	NaN	NaN	e057f6d2-58a5-e832-6312-57eee9e29aa7	1.085487e+18	e057f6d2-58a5-e832-6312-57eee9e29aa7	Z003XDCS	True	delays	True	negative
13991	2019-01-16 11:09:15	3295163848.0	This morning, on a busy commuter train, a woma...	brandwatch	en-GB	NaN	NaN	8fc9f9e1-9dc4-90e7-2b7e-86f8752661ed	1.085494e+18	8fc9f9e1-9dc4-90e7-2b7e-86f8752661ed	Z003XDCS	True	none	True	neutral
11081	2019-01-16 11:58:45	447569320.0	@myubi @UlyssesGuybrush @delayrepayagent @TLRa...	brandwatch	en-GB	NaN	NaN	b815339a-9981-862c-a062-193cb7fd2d2c	1.085507e+18	b815339a-9981-862c-a062-193cb7fd2d2c	Z003XDCS	True	delays	True	neutral

PAGE 29

Here are the results of first 2 steps (split data and sort by timestamp) using Python script.

Flowchart for Data Preparation for Delay Classification

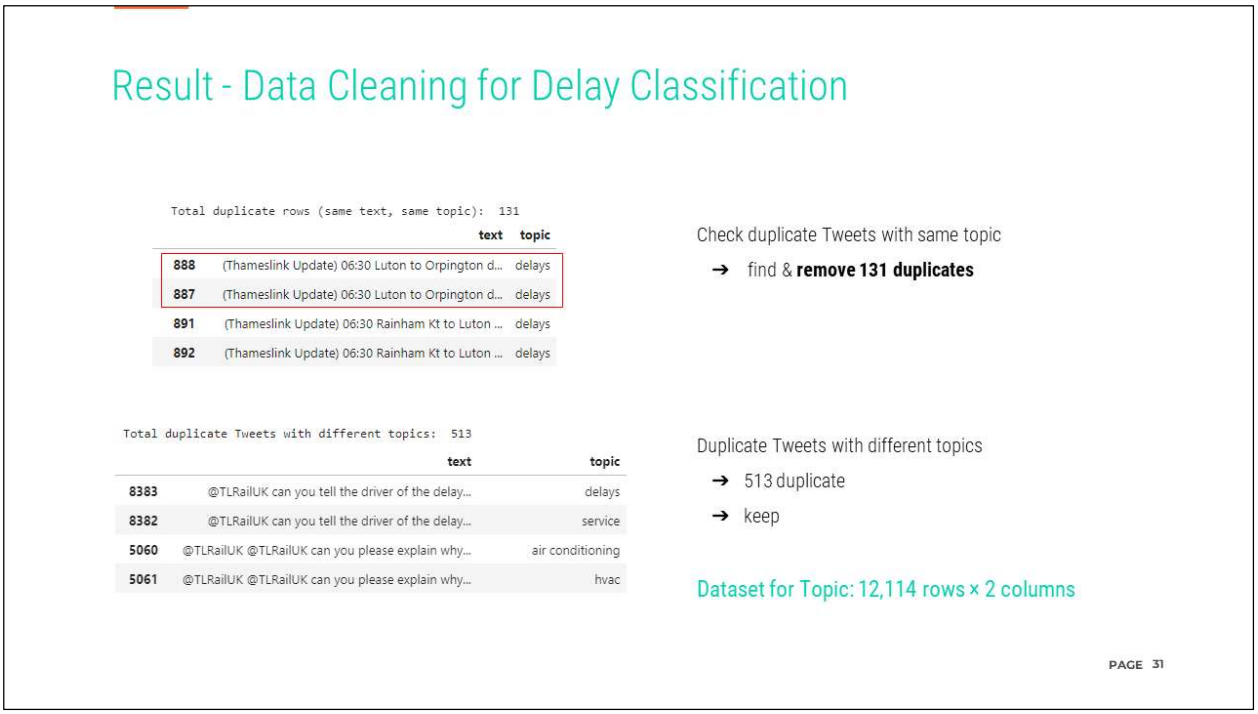


PAGE 30

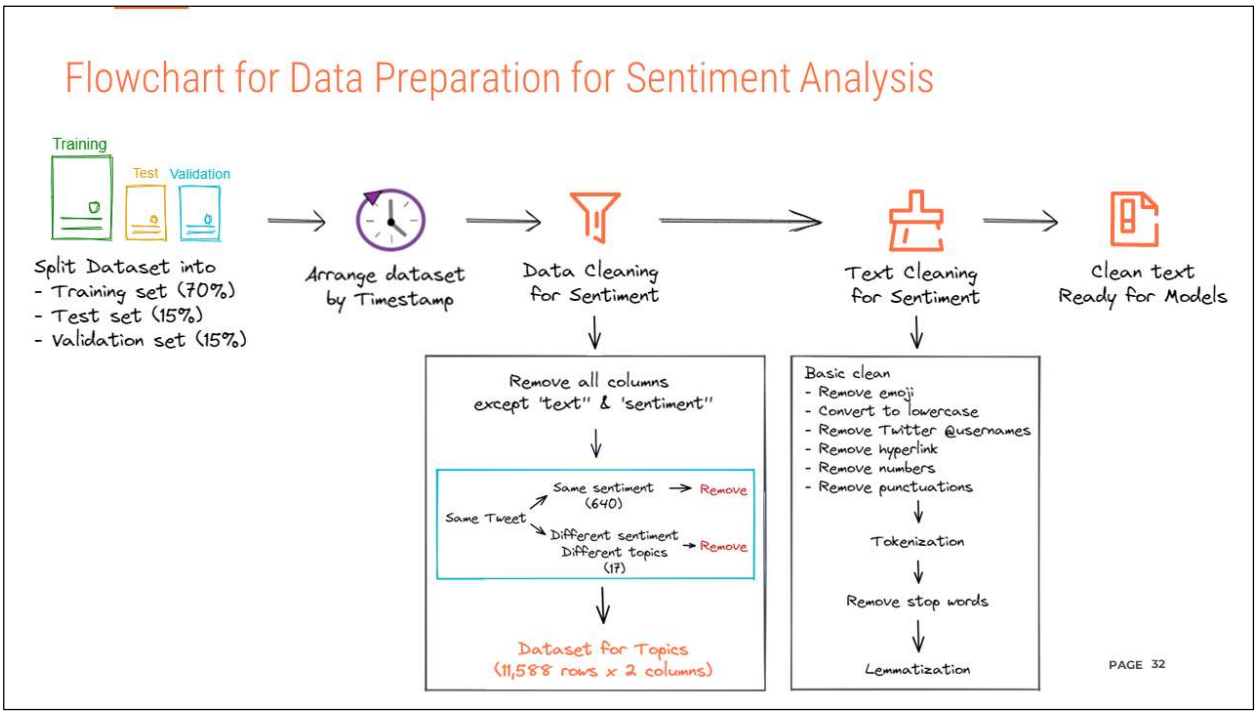
This is the flowchart for data preparation process for delay classification. After the first 2 steps, the next steps include:

1. Remove all columns except 'text' and 'topic' in the training set. The same step will be applied to test and validations sets at the time of modelling.
2. Remove 131 duplicates (i.e. rows with same tweet content and topic). There is other 513 tweets with same content and different topics but we decide to keep them. After removing duplicates, the training dataset now has 12,114 rows and 2 columns.
3. Clean text in tweet content. Steps include:
   a. Basic text cleaning includes removing emoji, hyperlink, Twitter username, numbers, punctuations, convert text to lowercase.

b. Tokenization - break raw text into words
c. Remove stop words
d. Lemmatization - strip words to their base roots
e. Done data preparation process. Clean texts are ready to be passed to models



Here are the results of finding and removing duplicates. After this, the training dataset now has 12,114 rows and 2 columns.

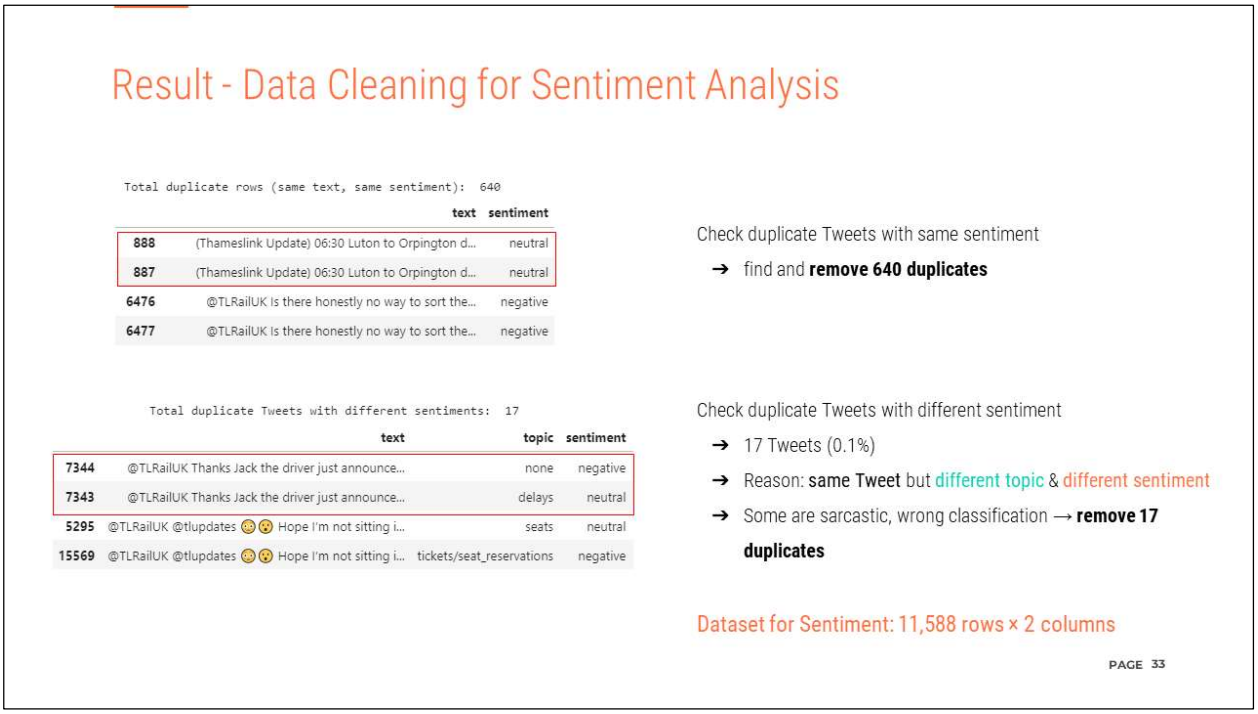


This is the flowchart for data preparation process for sentiment analysis with the similar steps as follows:

1. Remove all columns except ‘text’ and ‘sentiment’ in the training set. The same step will be applied to test and validations sets at the time of modelling.
2. Remove 640 duplicates (i.e. rows with same tweet content and sentiment). There are other 17 tweets with same content and different sentiments in different topics. Since the number was

small (0.1%) and most of them seemed to be sarcastic in nature and wrongly classified, we decide to remove them as well. After this, the training dataset now has 11,588 rows and 2 columns.

3. (Same) Clean text in tweet content. Steps include
   a. Basic text cleaning includes removing emoji, hyperlink, Twitter username, numbers, punctuations, convert text to lowercase.
   b. Tokenization - break raw text into words
   c. Remove stop words
   d. Lemmatization - strip words to their base roots
   e. Done data preparation process. Clean texts are ready to be passed to models



Here are the results of finding and removing duplicates. After this, the training dataset now has 11,588 rows and 2 columns.

Summary of Data Preparation

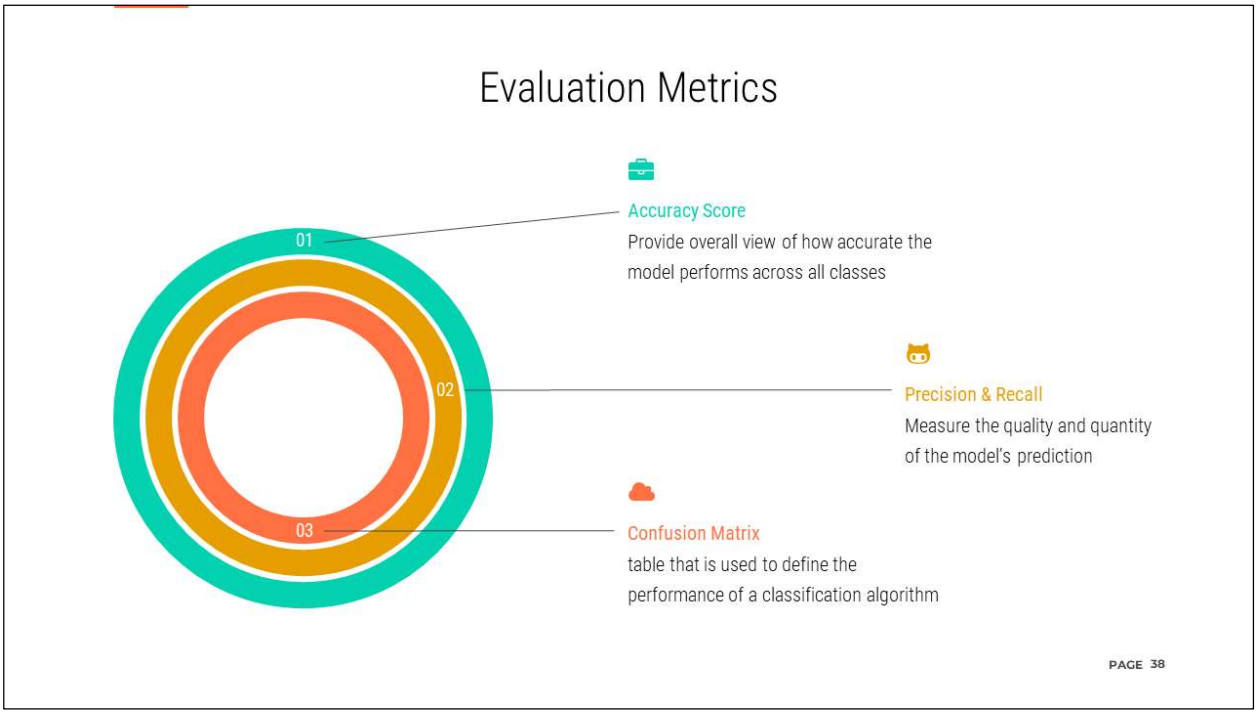
No	Problems	Possible Solutions	Risk Assessment
1	Text cleaning - Removing numbers		Number already removed from the texts but we might reconsider this step for later when building the ML models for topic classification
2	We preferred Lemmatization over Stemming	• Lemmatization is slower than Stemming • Lemmatization decreases noise and has higher accuracy than Stemming	
3	Some terms associated with Thameslink such as thameslink, TLUpdates, gtrailuk, Govia Thameslink Railway, TLRailUK, etc	These terms is useful in harvesting the Tweets from Twitter API but won't give much meaningful to build the ML models ➔ add to the stop word list to be removed during cleaning process	
4	Removing punctuation causes incorrect forms for words such as • didn't → didnt • I'm → im		❖ **for negative word (didn't):** might be useful later in the modelling phase for sentiment ➢ keep them in the dataset for now ➢ go back and further clean if needed ❖ **for other words (I'm)**: become typos and remains in the dataset → create noise → further clean needed

PAGE 35

Summary of Data Preparation

No	Problems	Possible Solutions	Risk Assessment
5	Hashtags (#graffiti, #work, #friday, etc) are used to filter data quickly on Twitter		Removing hashtags (#) only and leave the text in the dataset ➔ might not be meaningful or helpful in building models
6	Typos such as • 10mins • #Thameslink#Efficientcy → thameslinkefficientcy • delay/on-time → delayontime		Typos remains in the dataset → create noise → further clean needed
7	"none" is the 2nd biggest topic		"none" is a big topic but doesn't give much meaningful insights ➔ Search for solutions to re-distribute tweets in "none" to other topics
8	Imbalance proportion of values in each categories (sentiment & topics)	Treatment for data imbalance before modeling phase (SMOTE, ROS, RUS, etc)	The model would learn from the majority class
9	Wrong GPS & missing GPS data	GPS information is irrelevant to build ML model for sentiment & topic ➔ remove during cleaning process	

PAGE 36



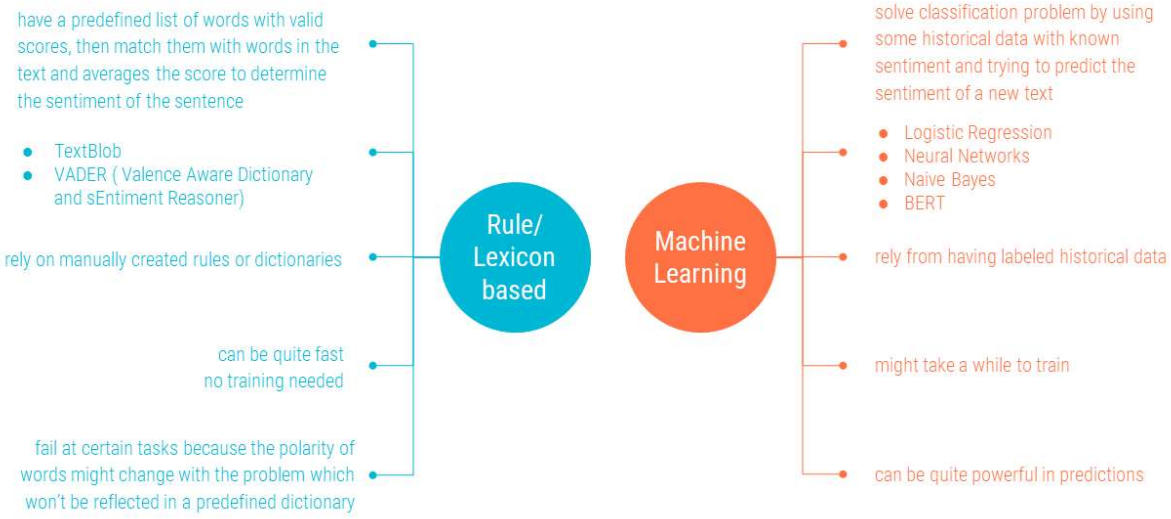
We use 3 different metrics to evaluate and assess the performance of these methods.

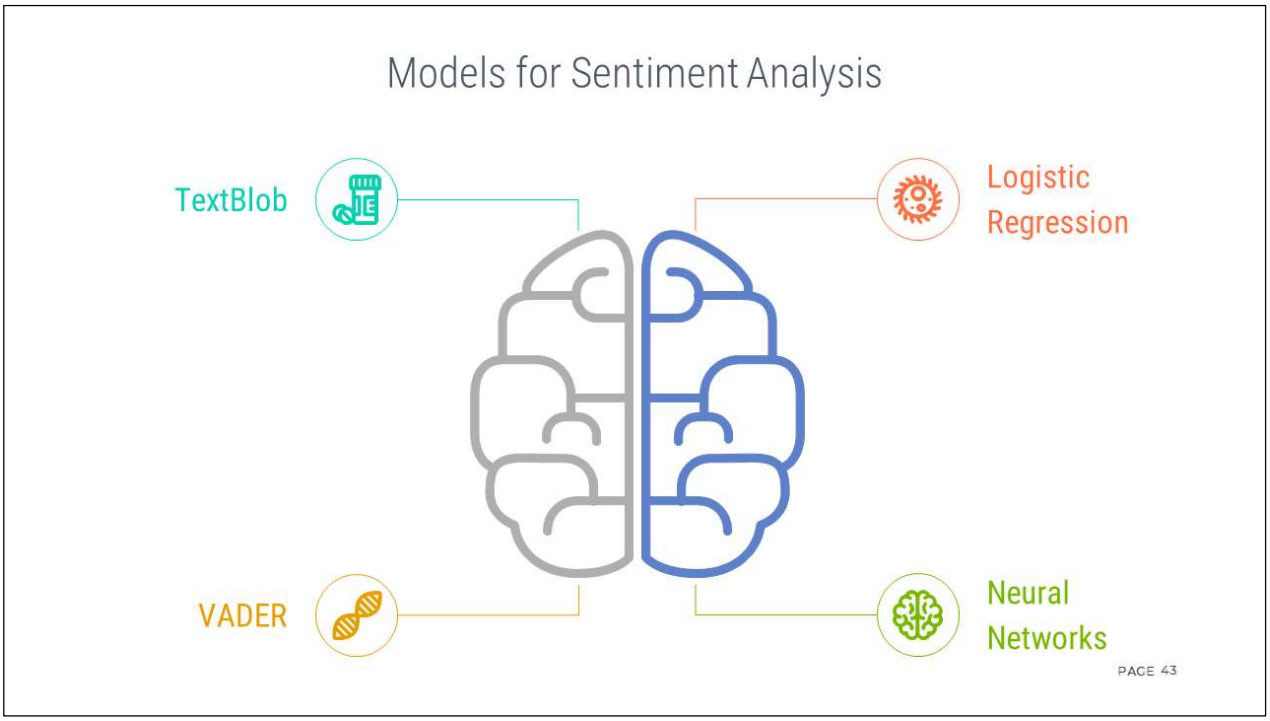
1. Accuracy: provide the overall view of how accurate the model performs across all classes
2. Precision & Recall: measure the quality and quantity of the model's prediction. Precision measures the reliability of our model in classifying the positive samples while recall measures the model's ability to detect positive samples.
3. Confusion Matrix: a table used to define the performance of a classification algorithm

Sentiment Analysis

- Types of Algorithms for Sentiment Analysis
- Models for Sentiment Analysis
  - TextBlob
  - VADER
  - Logistic Regression
  - Neural Networks
- Model Evaluation
  - Model Comparison
  - Final Model for Sentiment Analysis

Type of Algorithms for Sentiment Analysis





For sentiment analysis, we will try 4 different methods:

- TextBlob and VADER from lexicon-based
- Logistic Regression from machine learning
- Neural Networks from deep learning

Lexicon Based Approach

TEXTBLOB

- TextBlob is a python library for text analytics and natural language processing operations
  - input: a single tweet
  - output: polarity (score between -1 to 1)
    - -1: negative
    - 0: neutral
    - 1: positive
- The baseline to compare between different methods
  → **apply on test set only**

```
Test set has  2363 rows x 2 columns with
negative    1464
neutral      865
positive      34
Name: sentiment, dtype: int64
```

text	sentiment	polarity	TextBlob Analysis
we have been advised by our colleagues at ne...	neutral	0.014583	positive
aiming like chucking a hot dog up a toilet r...	negative	0.108333	positive
long time since ive had to get a train late f...	negative	-0.275000	negative
can i ask for your views please mr brake on ...	neutral	0.258333	positive
how is it even possible for you to compound t...	negative	-0.100000	negative

```
TextBlob Accuracy: 38.04 %
----------------------------------------
Confusion Matrix using TextBlob

          neutral  negative  positive
neutral       228       294       343
negative      305       647       512
positive        8         2        24
```

PAGE 41

The results of using TextBlob is on the right.

- The test set has 2,636 rows with different sentiment classes.
- DataFrame after running TextBlob in test set. It gives the polarity and corresponding sentiment
- TextBlob accuracy is 38.04% - our baseline

Lexicon Based Approach

VADER (Valence Aware Dictionary and sEntiment Reasoner)

- VADER is a module in nltk.sentiment Python library. It not only tells the lexicon is positive, negative, or neutral but also determine the overall sentiment of a text.
  - input: a single tweet
  - output: 'neg', 'neu', 'pos', and 'compound'
    - neg: negative
    - neu: neutral
    - pos: positive
    - compound: an indispensable score that is calculated by normalizing the other 3 scores (neg, neu, pos) between -1 and +1.
- **apply on test set only**

text	sentiment	neg	neu	pos	compound	VADER Analysis
advised colleague network rail point failure s...	neutral	0.356	0.644	0.000	-0.7351	negative
aiming like chucking hot dog toilet roll tube ...	negative	0.000	0.762	0.238	0.3612	positive
long time ive train late filthy got seat look ...	negative	0.168	0.670	0.162	-0.0258	negative
ask view mr brake 1m fine given railway death ...	neutral	0.157	0.726	0.117	-0.4588	negative
possible compound today delay having train pla...	negative	0.103	0.897	0.000	-0.3182	negative

```
TextBlob Accuracy: 38.04 %
-----------------------------------
Confusion Matrix using TextBlob

          neutral  negative  positive
neutral       228       294       343
negative      305       647       512
positive        8         2        24
```

```
VADER Accuracy: 43.80 %
-----------------------------------
Confusion Matrix for VADER

          neutral  negative  positive
neutral       126       461       278
negative      164       881       419
positive        3         3        28
```

PAGE 42

The result of using VADER is on the right.

- DataFrame after running VADER in test set. It gives 4 keys 'neg', 'neu', 'pos', 'compound' and corresponding sentiment
- VADER accuracy is 43.80%, which is a little bit better than TextBlob at 38.04%

Machine Learning Approach - Logistic Regression

Original Flowchart

PAGE 43

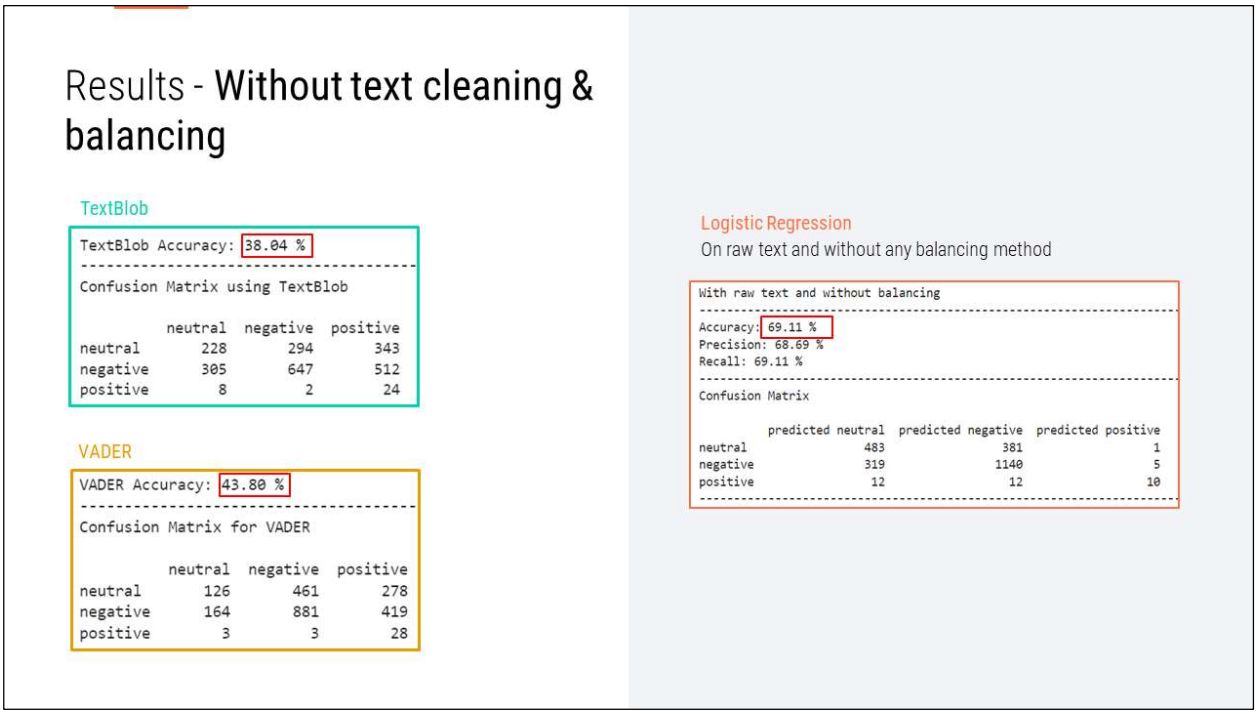
Third method is Logistic Regression. Here is the original flowchart of this method.

- Training set goes through the following steps:
  - Text cleaning: 2 options: basic clean or full clean
  - Vectorization: using CountVectorizer provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each

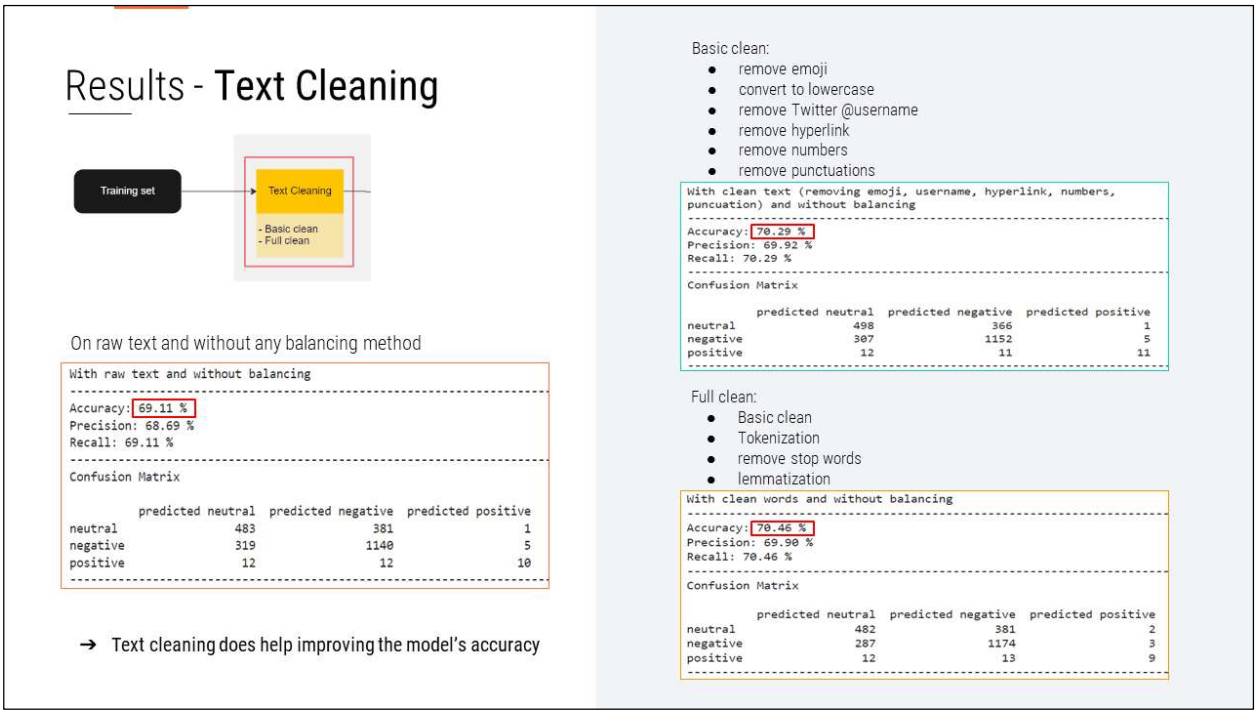
word that occurs in the entire text. Because machine only understand numbers, we have transformation text into numbers before passing them to the model.

- Balancing: because our dataset is quite imbalance with most of them is negative tweets, we need to re-balance it before modelling to reduce the biased prediction towards the negative class. We use some balancing methods such as random over sampling, random under stamping, etc and also class-weight that is built-in with Logistic Regression model.
- Train and evaluate model

- Test set goes through the same text cleaning process, vectorization and into the trained model which will give us the sentiment score.



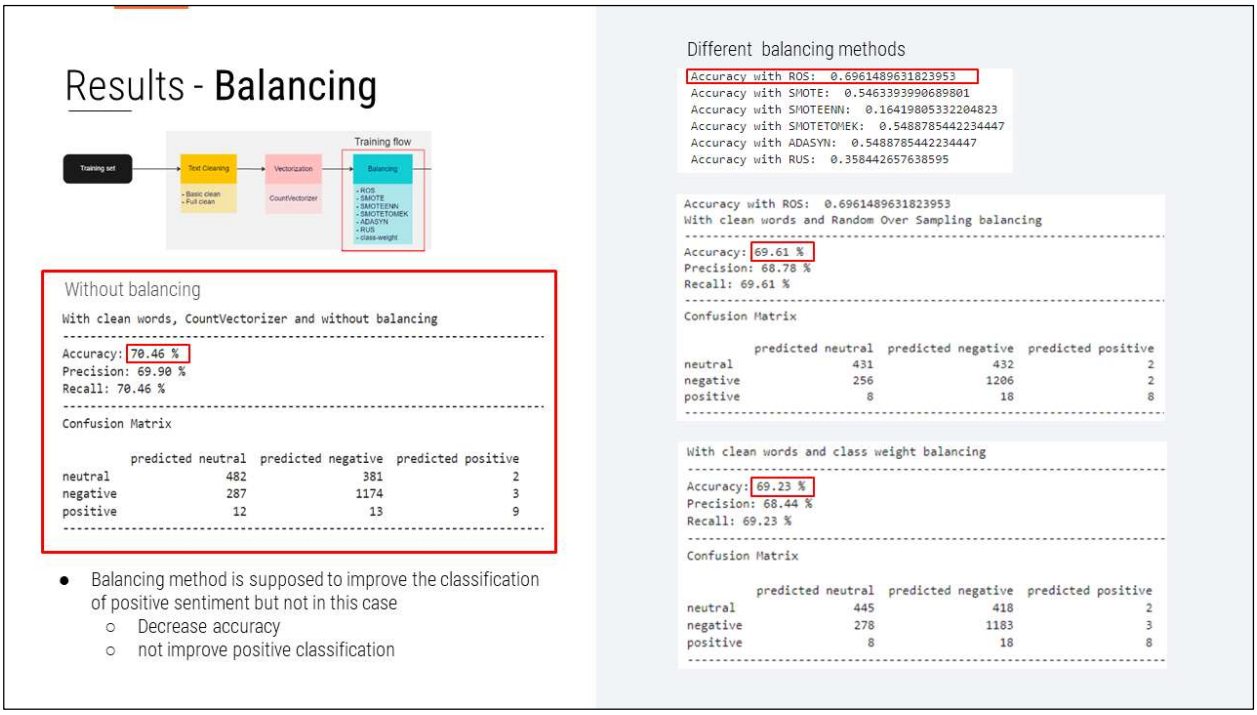
Here are the results of using Logistic Regression model. Even with raw text (i.e. no text cleaning) and without any balancing method, it still gives a better result than either TextBlob or VADER.



Next, we see how different text cleaning processes affect the model's performance. For text cleaning, there are 2 options:

- Basic clean, which removes emoji, convert text to lowercase, etc. With basic clean, the accuracy is 70.29%.
- Full text cleaning, which cleans the text further with tokenization, stop words removal and lemmatization. As expected, the accuracy increases a bit to 70.46%

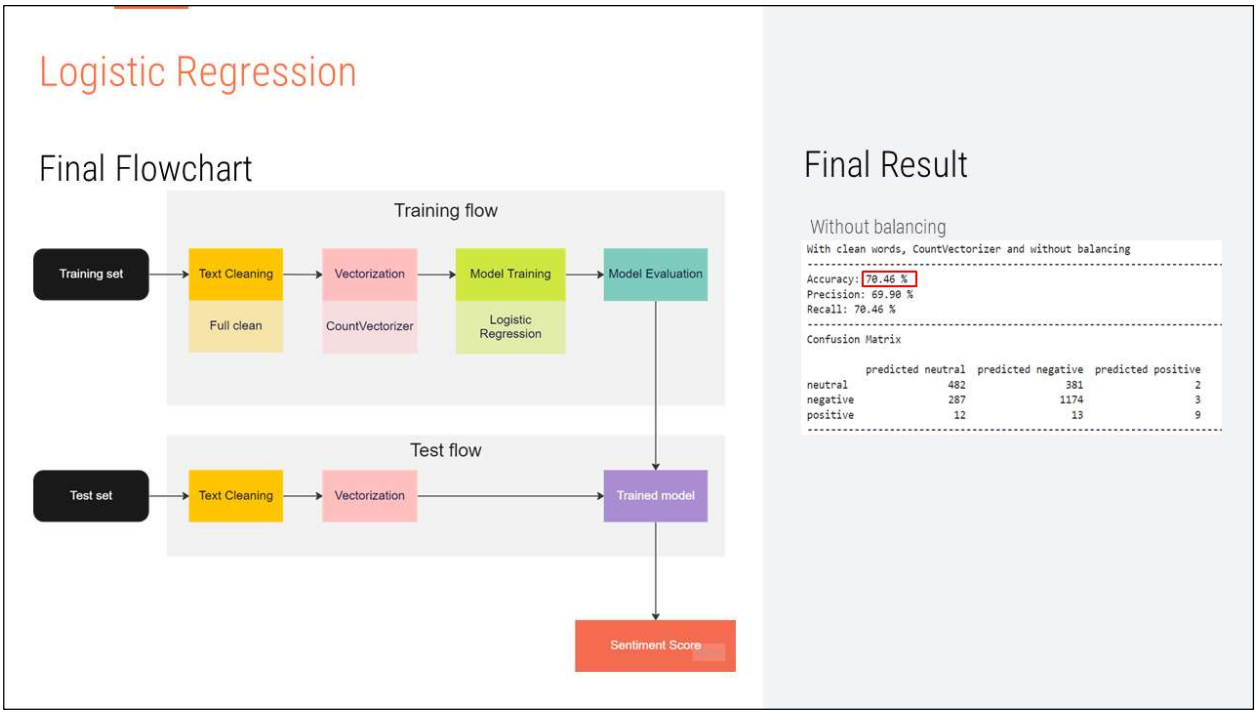
Compare the results without text cleaning (on the left) and with text cleaning (on the right), the accuracy improves a little. Therefore, text cleaning help improving the model accuracy.

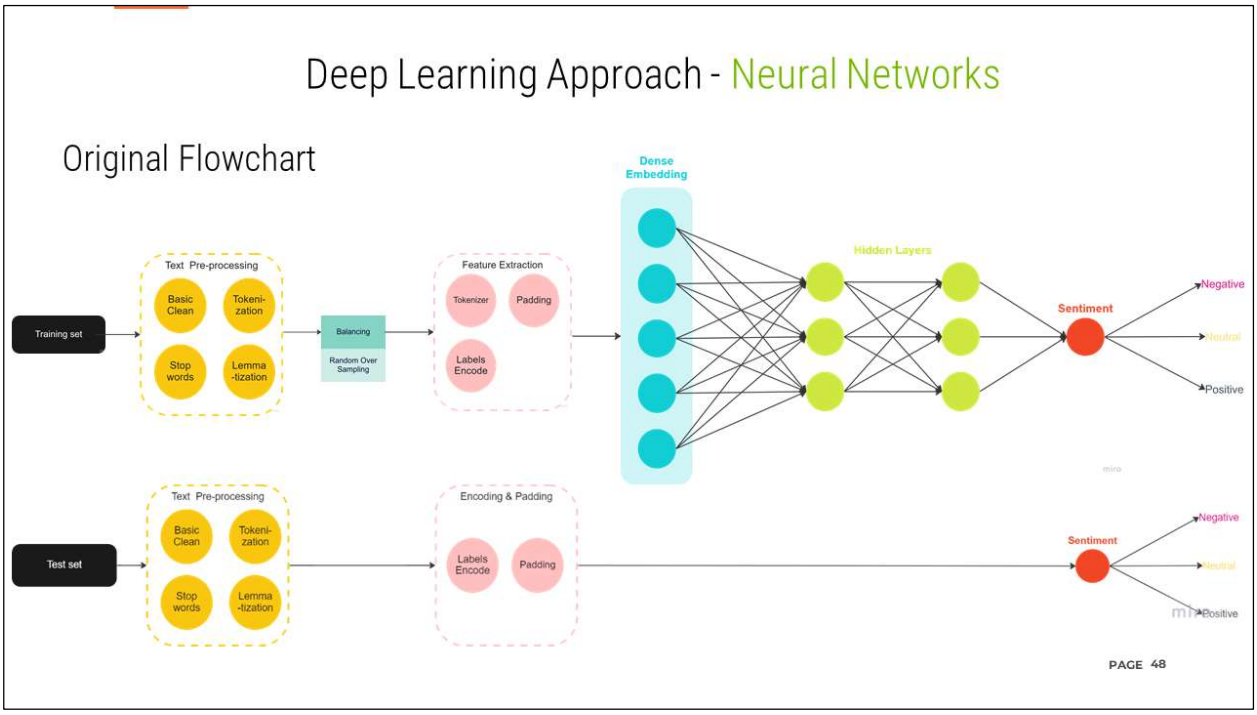


Next, we see how different balancing methods affect the model's performance. For balancing, there are 2 options:

- 6 different balancing methods: Random oversampling, SMOTE, SMOTEENN, SMOTETOMEK, ADASYN, Random undersampling. Out of these methods, Random oversampling yields the best result with accuracy of 69.61%.
- class-weight balancing within Logistic Regression: class_weight is a dictionary that determine the weight of each class and apply those weights to the calculation during the model training. This gives a slightly worse result than Random oversampling with accuracy of 69.23%.

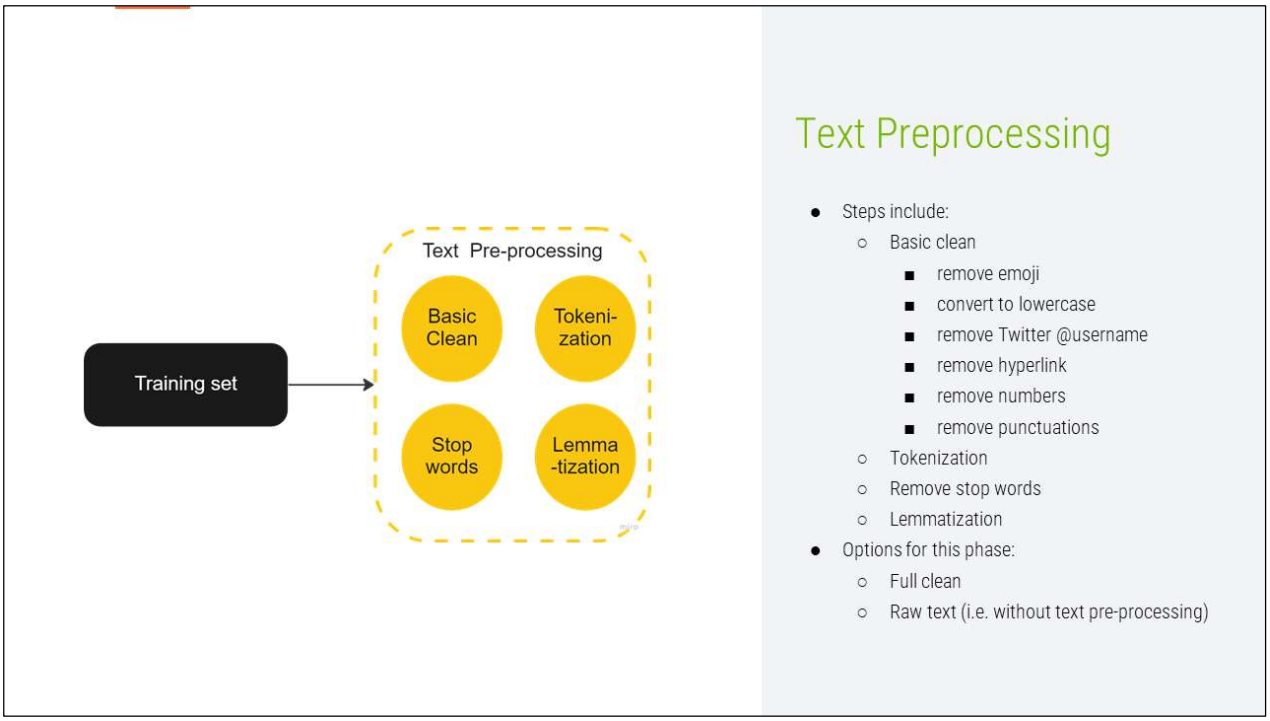
Balancing is supposed to help improve the accuracy and positive classification. However, the model with both balancing options (on the right) performs worse than it without any balancing (on the left - 70.46% accuracy). Thus, it's best to not apply any balancing to Logistic Regression model



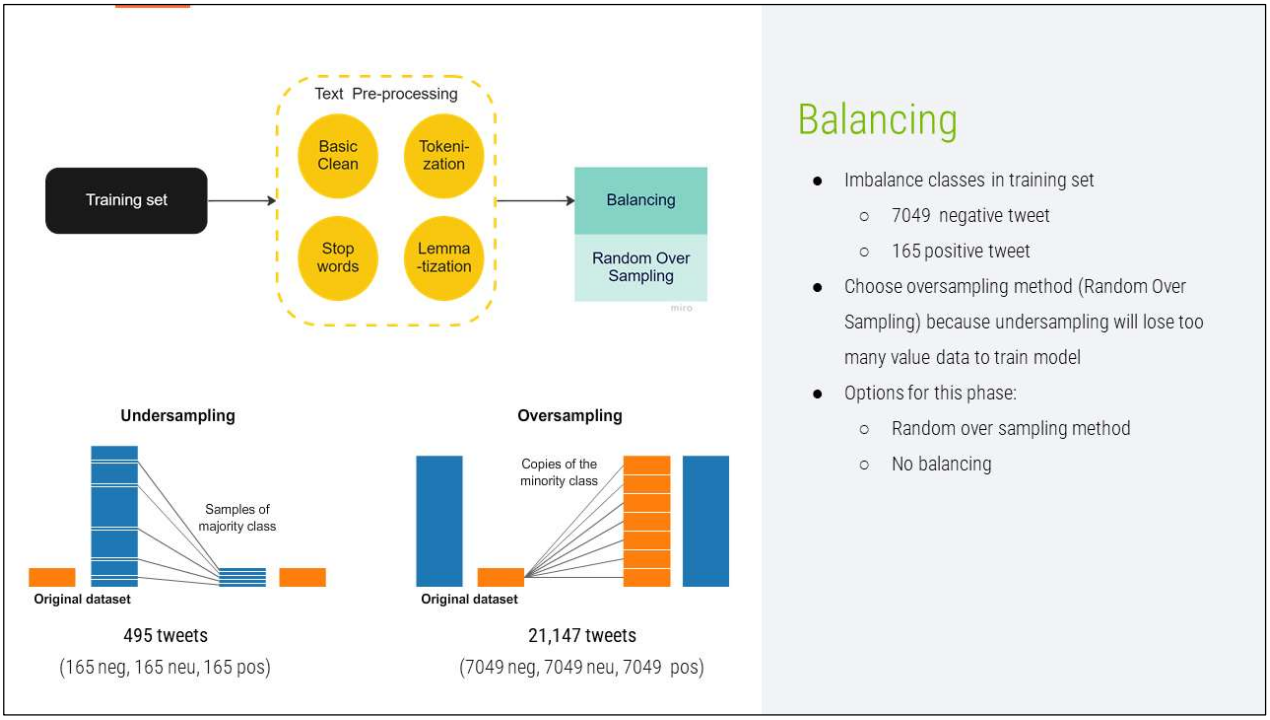


Final method is Neural Networks. Here is the original flowchart of this method.

- Training set goes through the following steps:
  - Text cleaning: all cleaning steps like in Logistic Regression
  - Balancing: Random oversampling method.
  - Feature Extractions: tokenizer, padding and label encoding
  - Train Neural Networks model
- Test set goes through the same text cleaning process, padding, encoding and into the trained model which will give us the sentiment score.

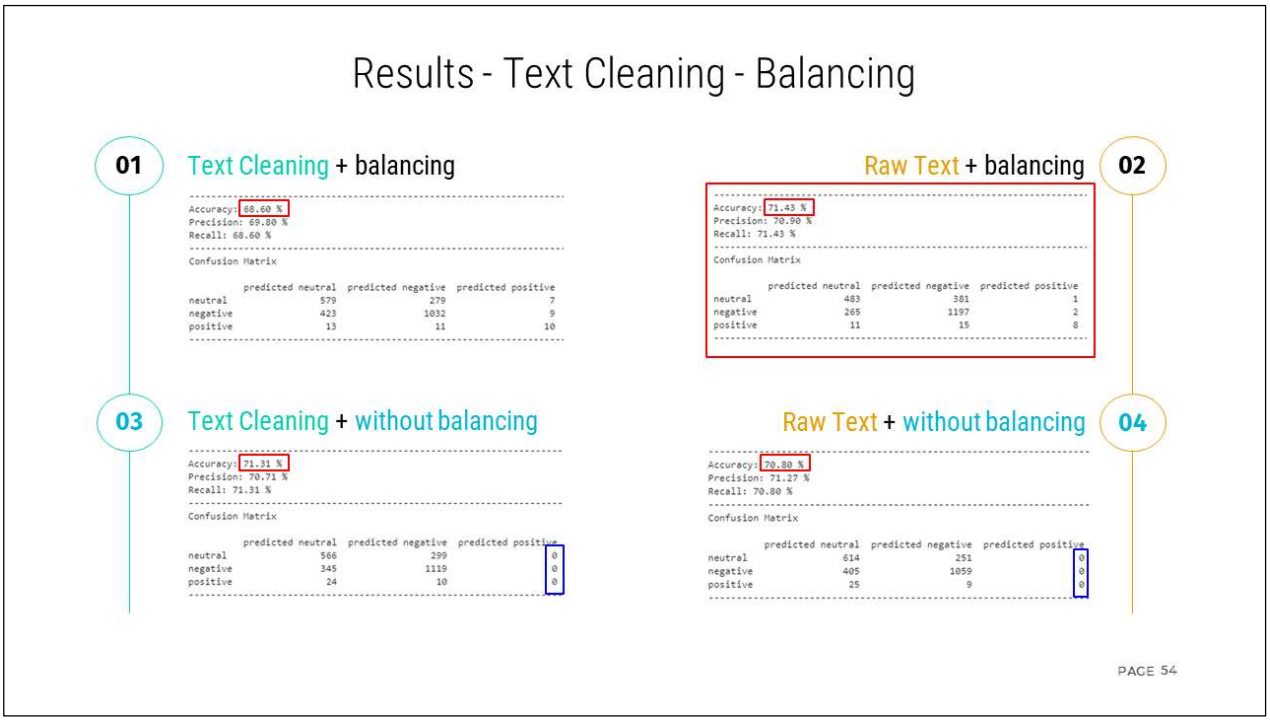


For text cleaning, we apply the same steps as in Logistic Regression. Because text cleaning helps improve the accuracy of Logistic Regression model, we also want to see how the Neural Networks model will perform, with and without text cleaning process.



Because our dataset is quite imbalance with most of them is negative tweets, we need to re-balance it before modelling to reduce the biased prediction towards the negative class.

- 2 main balancing methods:
  - Oversampling: duplicate or create new synthetic examples in the minority class → dataset of 21,147 tweets → more data to train
  - Undersampling: delete or merge examples in the majority class → dataset of 495 tweets → less data to train
  - Choose oversampling method (Random oversampling) because undersampling loses too many valued data
- Because balancing reduces the accuracy of Logistic Regression model, we also want to see how the Neural Networks model will perform, with and without balancing.

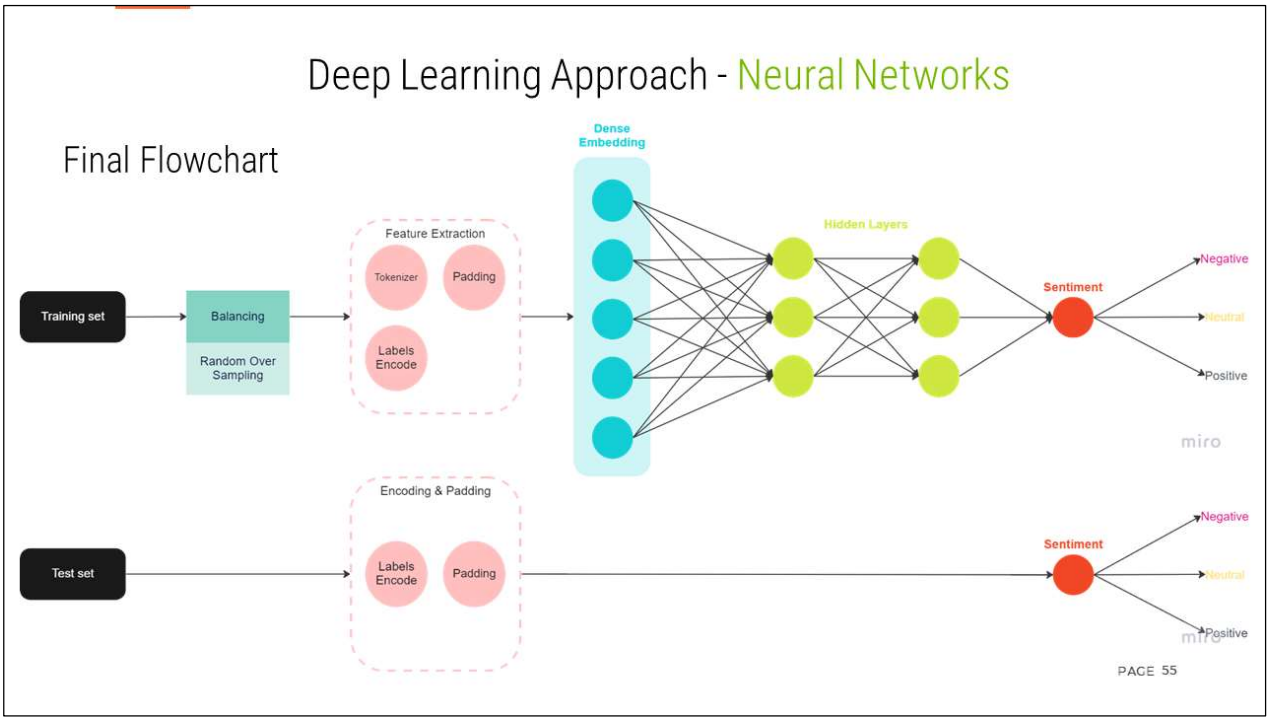


Here are the results of Neural Networks model

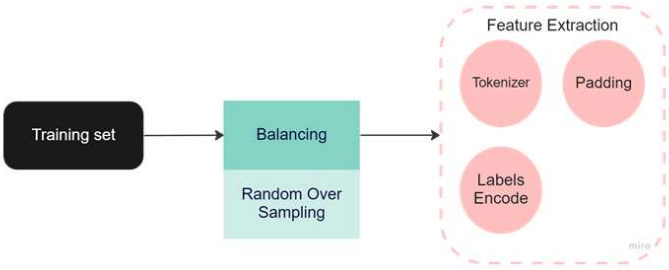
- with and without text cleaning and
- with and without balancing

The results of all 4 options are quite close to each other. However, looking at the 2 results of NN model without the balancing method (bottom left and right), it's clear that the model can't predict any positive tweet at all. As opposed to Logistic Regression where balancing actually reduces the model's accuracy, Neural Networks model does better with balancing, therefore, we will apply balancing to Neural Networks model

Moreover, contrast to Logistic Regression where the model does better with text cleaning, Neural Networks model does better without any text cleaning. As text cleaning takes up computing power and doesn't improve the model's performance, we will remove the text cleaning process from our flowchart.

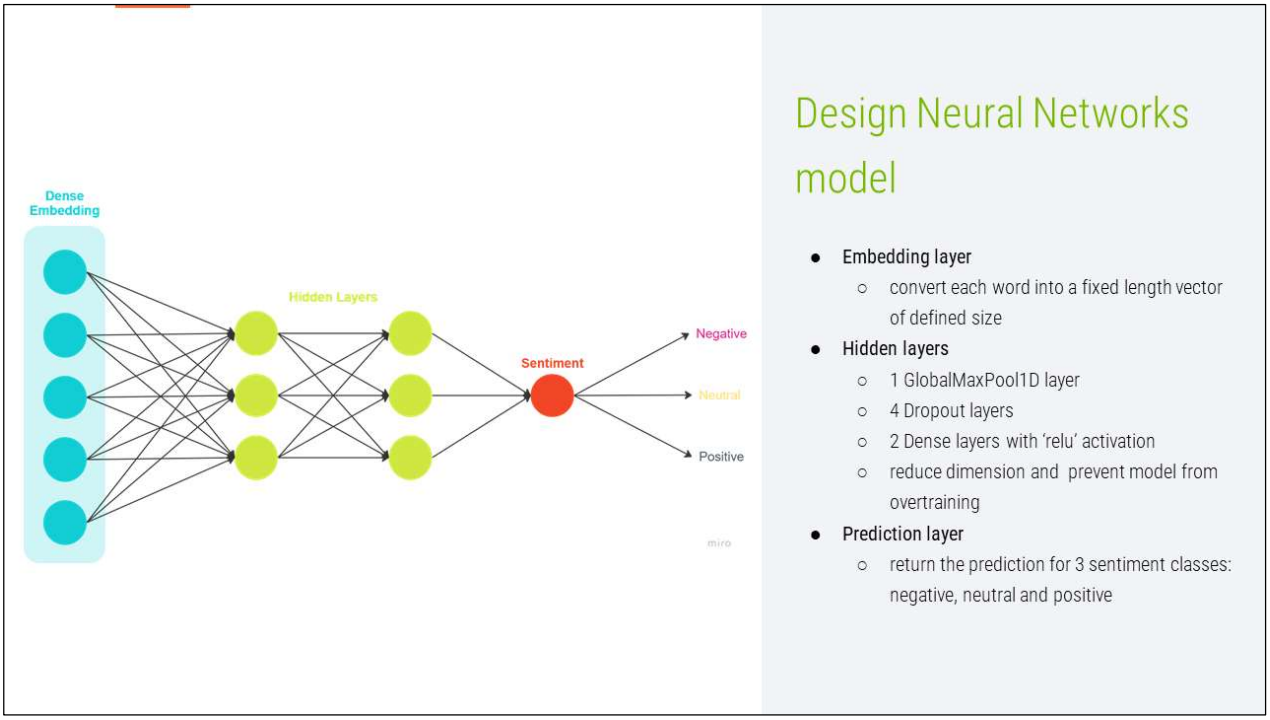


This is our final flowchart after removing the text cleaning process.



Feature Extraction

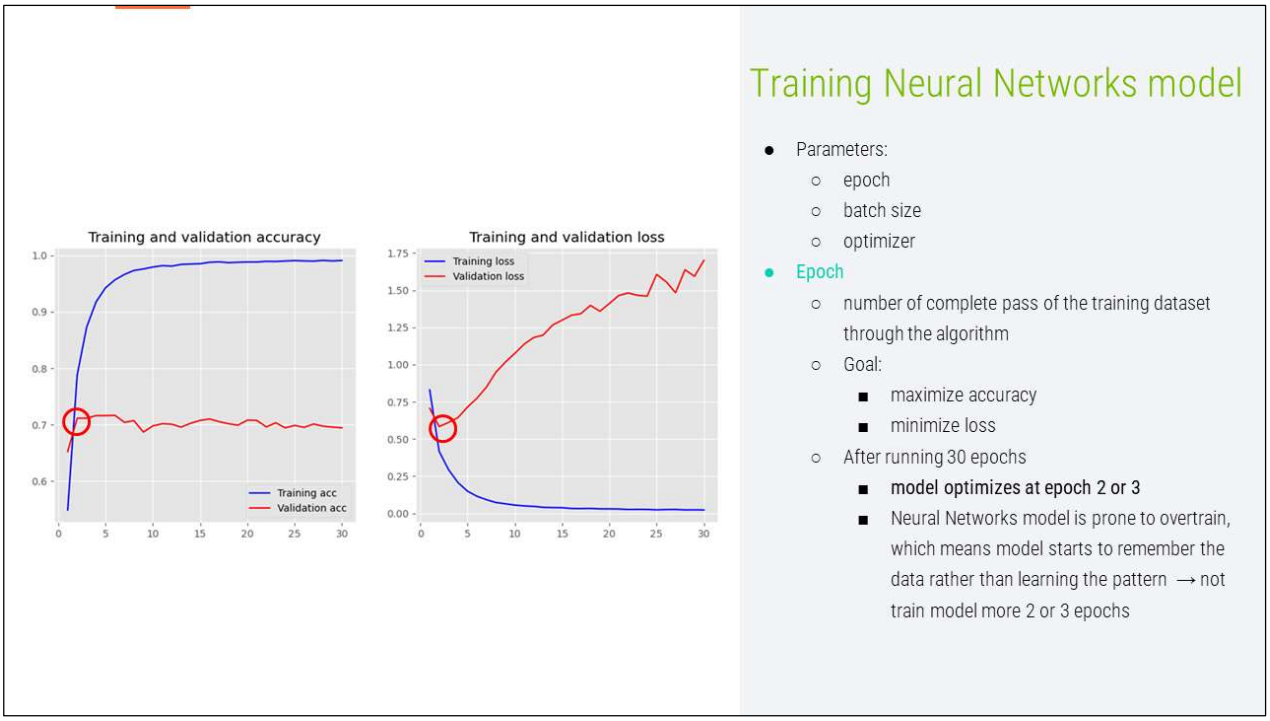
- Tokenizer
  - transform text into numeric tensors by going through all tweets and index each word in the tweet
  - if a word occurs multiple times within the tweet, it **has that token in a vector multiple times** and **the order of the words is important**
- Padding
  - purpose: make all tweets into the same length by
    - find the longest tweet in the dataset and set it to be the length of the input
    - insert paddings to fill in empty spaces for shorter tweets to make them into equal size
- Labels Encoding
  - transform sentiment into numbers before feeding into the model

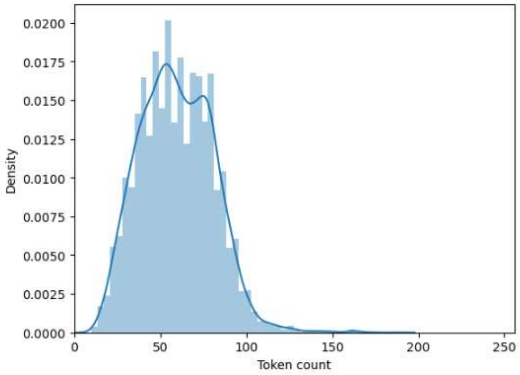


Next is designing the Neural Networks model with 3 main parts:

- Input layer (Embedding): convert each word into a fixed length vector of defined size.
- Hidden layers: reduce the dimension and prevent model from overtraining
- Output layer (Prediction): return the prediction for sentiment

There are different parameters for each of these layers that we run many trials to achieve the best result.





Training Neural Networks model

- Batch size:
  - number of samples that will be propagated through the network
  - most of our tweest are in the range of 128 → 128 will be our starting batch size
  - can be adjusted up or down to achieve the best results

With sgd

```
Accuracy: 1.44 %
Precision: 0.02 %
Recall: 1.44 %
Confusion Matrix

          predicted neutral  predicted negative  predicted positive
neutral                   0                   0                 865
negative                  0                   0                1464
positive                  0                   0                  34
```

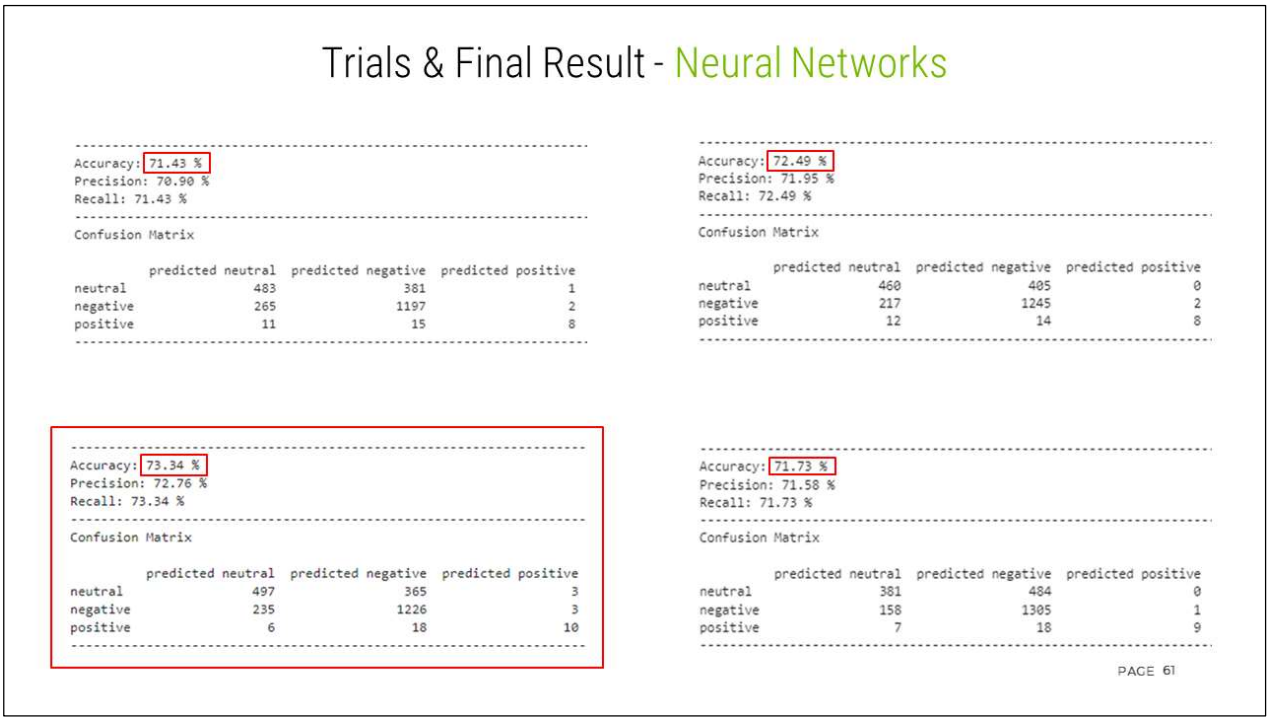
With adam

```
Accuracy: 71.43 %
Precision: 70.90 %
Recall: 71.43 %
Confusion Matrix

          predicted neutral  predicted negative  predicted positive
neutral                 483                 381                   1
negative                265                1197                   2
positive                 11                  15                   8
```

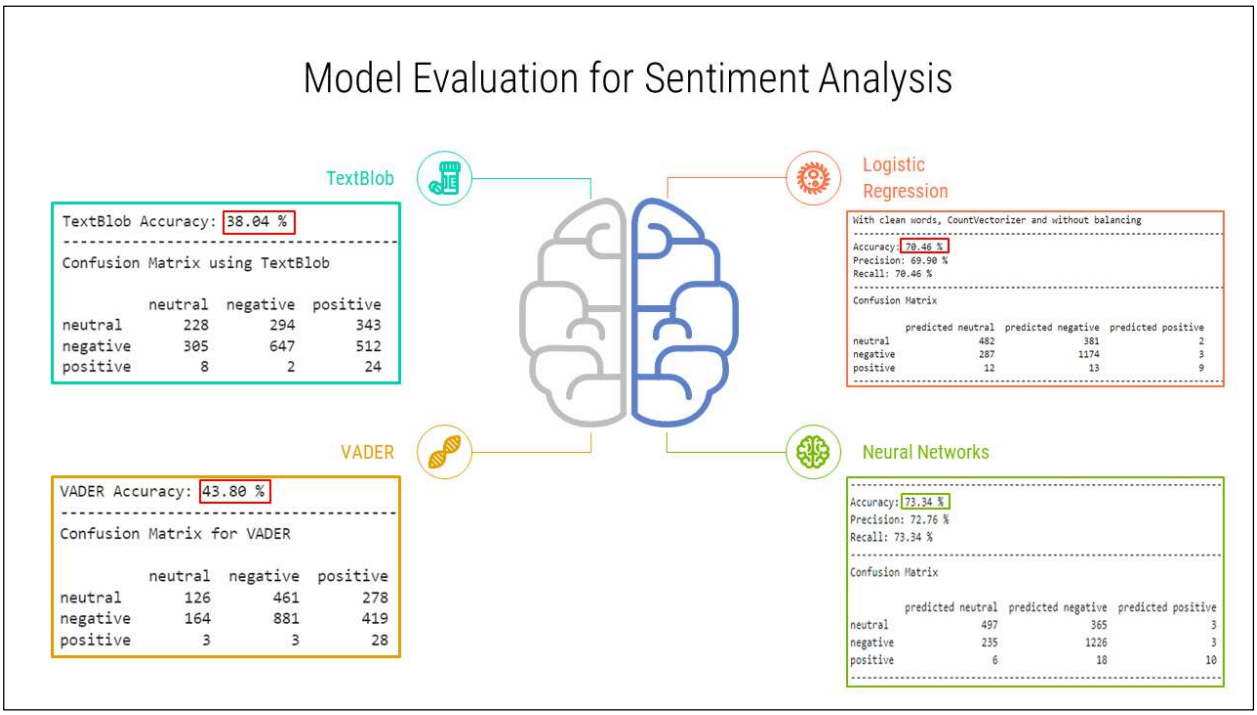
Training Neural Networks model

- Optimizer:
  - a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rate
    - → reduce loss
    - → improve accuracy
  - optimizer:
    - adam
    - sgd
    - "adam" optimizer outperforms "sgd" optimizer → choose adam



With Neural Networks, there are so many possibilities of changing the parameters to tune the model for better results. Here are some snapshots of our best results with the highest accuracy of around 72-73%.

In the bottom left corner is our best result so far 73.34% accuracy (using 2 epochs, batch size 160, optimizer adam, embedding dimension 500, 2 Dense layers with relu activation size 256 and 512, etc). And this is our final model for NN.



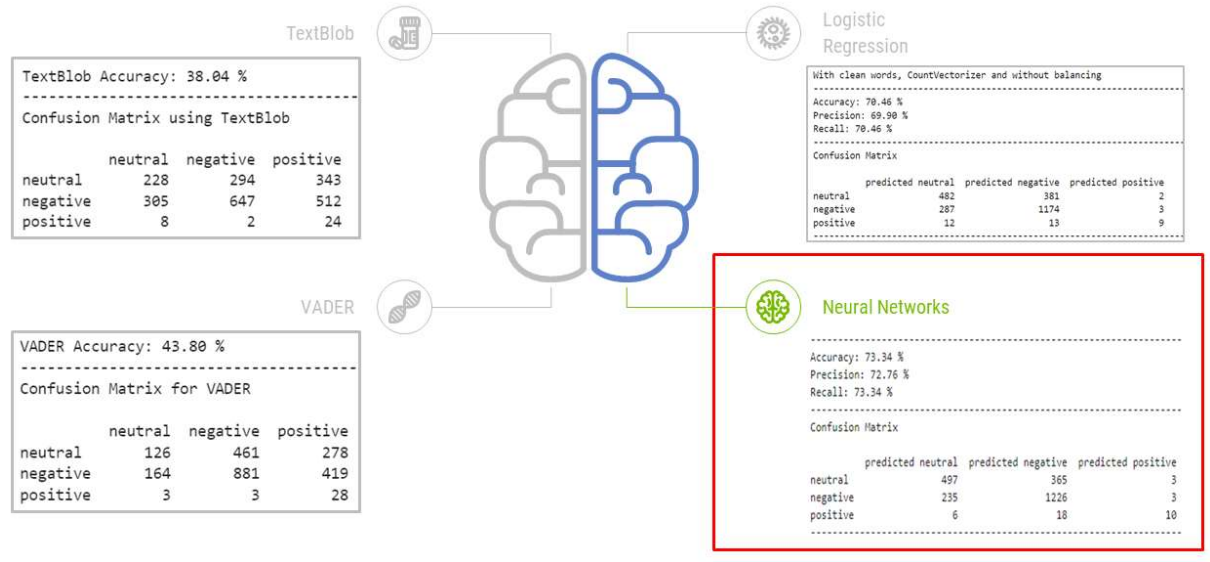
Machine learning and deep learning approach give better results than lexicon-based approach

Model Comparison

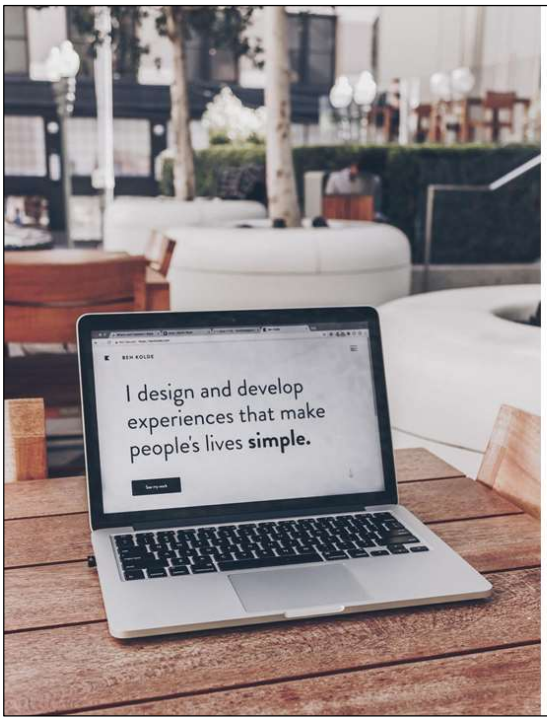
Criteria	Logistic Regression	Neural Networks
Text Cleaning	• improve accuracy	• reduce accuracy
Balancing	• reduce accuracy	• improve accuracy
Pros	• simple & easier to understand • less time & computing power to train the model • easier to fine tune	• give slightly better accuracy (73.34%)
Cons	• give slightly worse accuracy (70.46%)	• prone to overtrain • complex & harder to understand • require more time & computing power to train the model • more complicated and harder to fine tune

PAGE 53

Final Model for Sentiment Analysis



After evaluating and comparing the results of all models, we decide to go with Neural Networks model for sentiment analysis

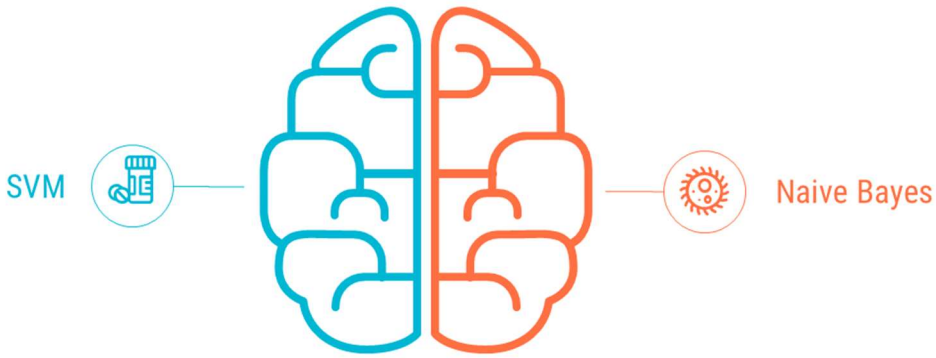


Delay Classification

- Models for Delay Classification
  - SVM
  - Naive Bayes
- Model Evaluation
  - Model Comparison
  - Final Model for Delay Classification

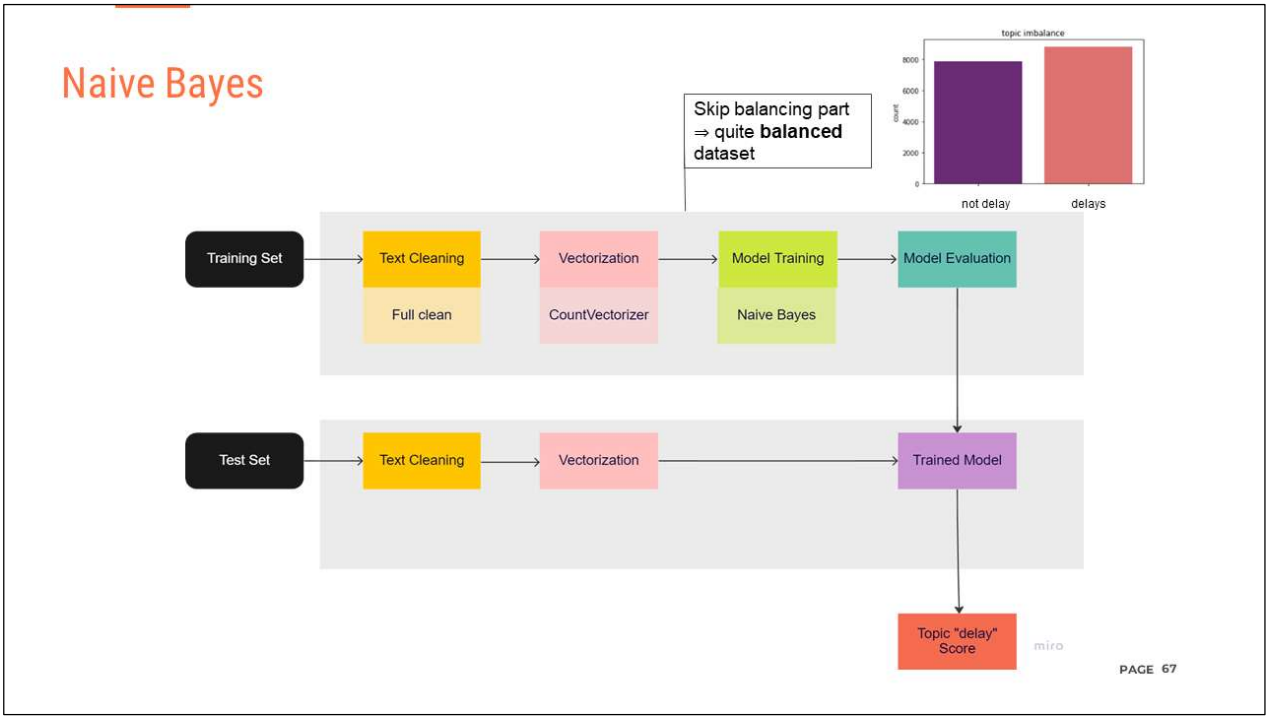
PAGE 65

Models for Delay Classification



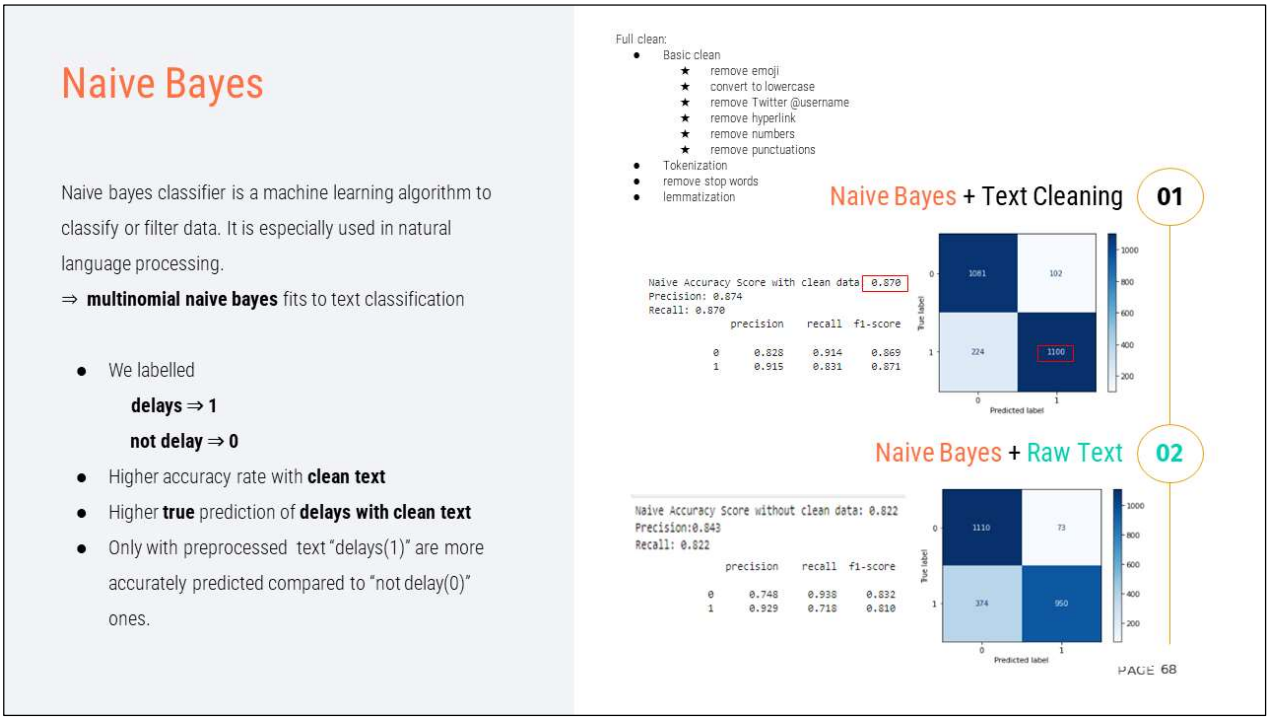
PAGE 66

We tried 2 different machine learning models for the delay classification. One is Support Vector Machine and the other is Naive Bayes. Both models are commonly used in NLP classification tasks.



First method for delay classification is Naive Bayes. Here is the flowchart:

- Training set goes through the following steps:
  - Text cleaning: full cleaning
  - Vectorization: using CountVectorizer provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. Because machine only understand numbers, we have transformation text into numbers before passing them to the model.
  - Balancing: We skip the balancing part because when we check the count of delay and non-delay, we see that the result is quite balanced.
  - Train and evaluate model
- Test set goes through the same text cleaning process, vectorization and into the trained model which will give us the sentiment score.



Naive Bayes (Multinomial Naive Bayes)

- Applied GRID SEARCH CV,
  Cross Validation (5-folds) with Grid Search CV

	alpha
Multinomial Naive Bayes	[0.00001, 0.0001, 0.001, 0.1, 1, 10, 100,1000]

Best Estimator `{'alpha': 10}` `MultinomialNB(alpha=10)`

	precision	recall	f1-score
0	0.828	0.914	0.869
1	0.915	0.831	0.871
accuracy			0.870

Increased accuracy

	precision	recall	f1-score
0	0.856	0.901	0.878
1	0.907	0.865	0.886
accuracy			0.882

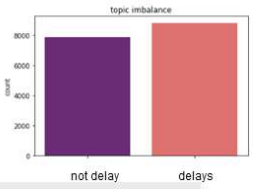
In Multinomial Naive Bayes, the alpha parameter is a hyperparameter; i.e. a parameter that controls the form of the model itself. In most cases, the best way to determine optimal values for hyperparameters is through a grid search over possible parameter values and using cross validation to evaluate the performance of the model as well as control overfitting.

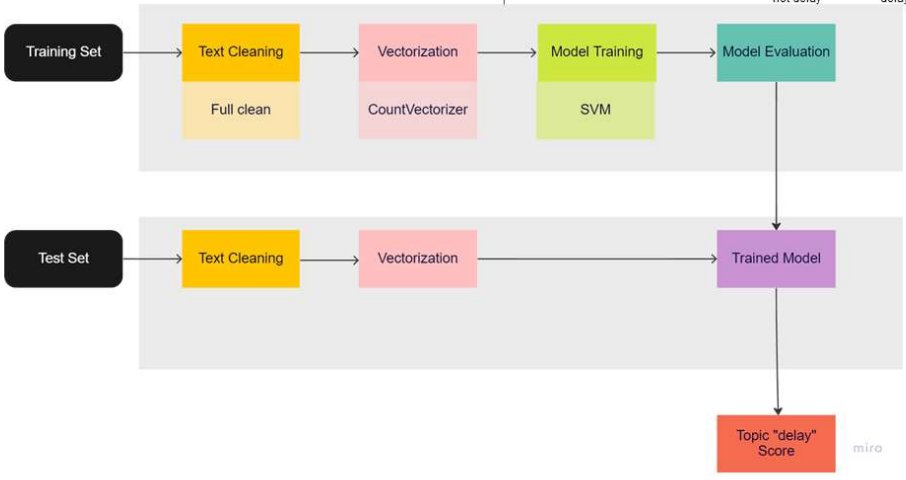
For our model we apply Grid Search cv, with the alpha parameters as mentioned in the table with a 5-fold Cross Validation.

The Best Estimator so far is where alpha is 10 and it increases our initial accuracy and f1-score for both predicting the delays and not delays.

Support Vector Machine (SVM)

Skipped balancing part ⇒ quite **balanced** dataset





miro

PAGE 70

The same flowchart of Naive Bayes model is applied to Support Vector Machine model.

Support Vector Machine (SVM)

Support Vector Machines is generally considered to be a classification approach

Kernel: **linear**

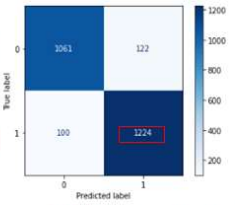
- We labelled
  - **delays ⇒ 1**
  - **not delay ⇒ 0**
- Higher accuracy rate with **clean data**
- Higher **true** prediction of **delays with clean data**
- In both Cases "delays(1)" are more accurately predicted compared to "not delay(0)" ones

Full clean:

- Basic clean
  - ★ remove emoji
  - ★ convert to lowercase
  - ★ remove Twitter @username
  - ★ remove hyperlink
  - ★ remove numbers
  - ★ remove punctuations
- Tokenization
- remove stop words
- lemmatization

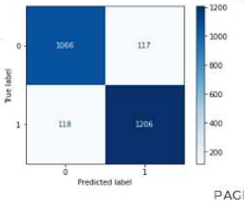
SVM + Text Cleaning **01**

```
SVM Accuracy Score with clean text: 0.919
Precision: 0.919
Recall: 0.919
              precision    recall  f1-score
           0      0.919     0.910     0.914
           1      0.920     0.928     0.924
```



SVM + Raw Text **02**

```
SVM Accuracy Score: 0.906
Precision: 0.906
Recall: 0.906
              precision    recall  f1-score
           0      0.900     0.901     0.901
           1      0.912     0.911     0.911
```



SUPPORT VECTOR MACHINE (SVM)

- Applied GRID SEARCH CV

  Disadvantages: Very long run time. ⇒ Trying combinations with other kernels did not work.

  The fastest kernel is Radial Basis Function with the following C and Gamma with 5-fold Cross Validation

  ⇒ but it did not increase accuracy.

KERNEL	C	GAMMA	CV
Radial Basis Function (RBF)	0.0001, 0.001, 0.1, 1, 10, 100, 1000	[1, 0.1, 0.01, 0.001, 0.0001	5

kernel: linear C:1.0

```
              precision    recall  f1-score
           0      0.919     0.910     0.914
           1      0.920     0.928     0.924
    accuracy                          0.919
```

Remained same

Best Estimator {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}

```
              precision    recall  f1-score
           0      0.919     0.910     0.914
           1      0.920     0.928     0.924
    accuracy                          0.919
```

Support Vector Machine also has some hyper-parameters like C or gamma values and kernel. However, finding optimal hyper-parameters is a very hard task to solve for this model.

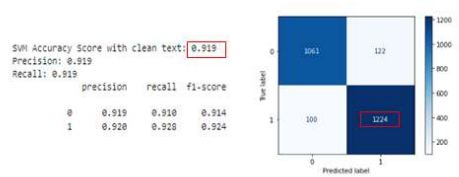
We need to try all combinations and see what parameters work best. The main idea behind it is to create a grid of hyper-parameters and just try all of their combinations. Here, we use the Grid Search CV method, based on all possible kernels such as Linear, poly, sigmoid and Radial Basis Function with 5-fold Cross Validation. However, we realize that it takes very long run and this option doesn't work very well. We try many options to make it run, however the fastest and only working kernel was (RBF).

We also use other hyperparameters as can be seen in the table such as C and Gamma. The best estimators we have so far after GridSearch is C:1000, gamma:0.0001 and kernel 'rbf'.

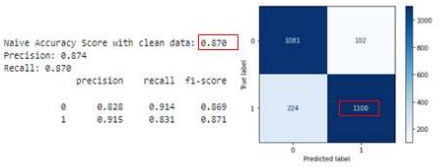
However, it does not affect our first model's performance at the end which was linear kernel and C=1.

Model Evaluation for Delay Classification

SVM + Text Cleaning



Naive Bayes + Text Cleaning

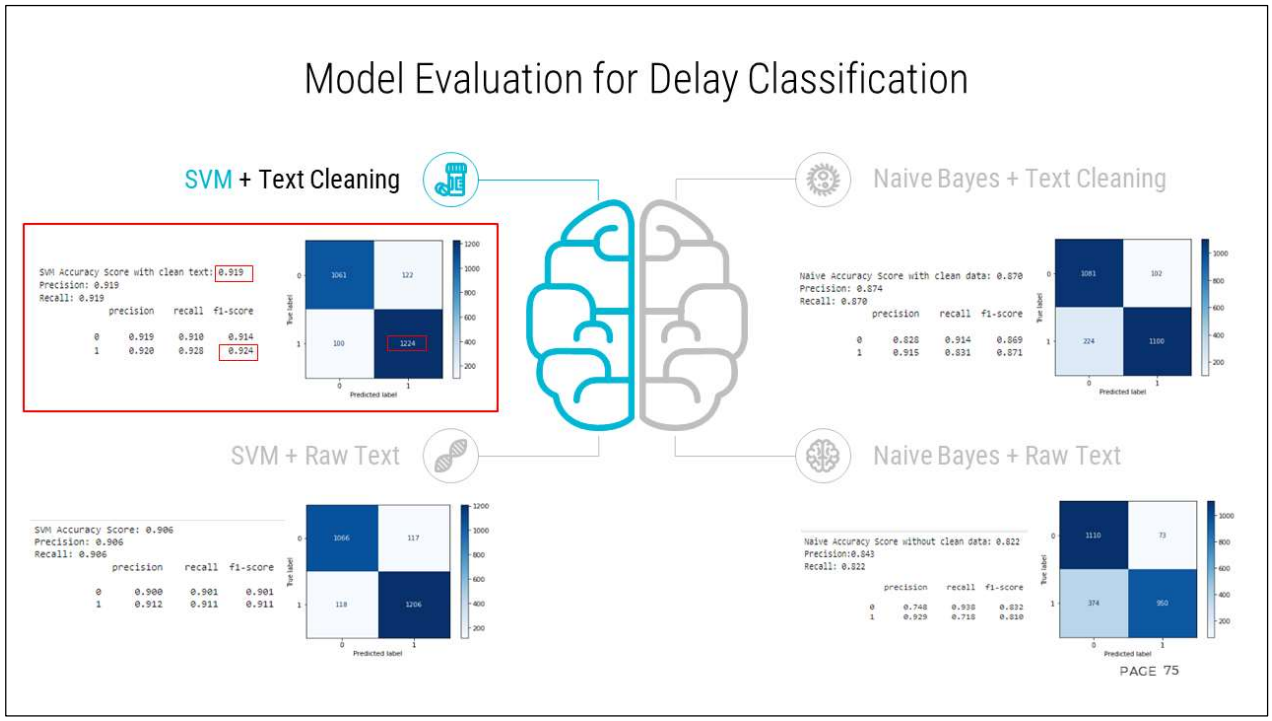


SVM + Raw Text

Naive Bayes + Raw Text

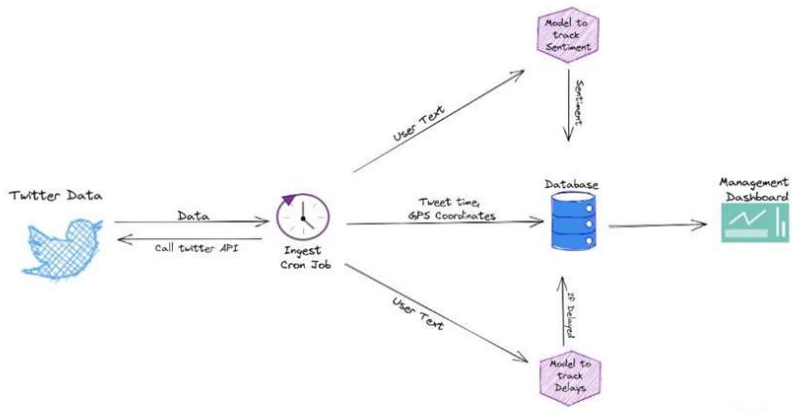
Model Comparison

Criteria	Support Vector Machine	Naive Bayes
Text Cleaning	• improve accuracy	• reduce accuracy
Pros	• gives better accuracy (91.9%) • simple & easy to understand • fast computing power to train the model in our case	• simple & easy to understand • fast computing power to train the model • easy to implement and evaluate.Therefore, it does not require an iterative process
Cons	• Choosing a good kernel is not easy • The SVM hyperparameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact	• give slightly worse accuracy (87%)



The best result out of these options is Support Vector Machine with the clean text.

High-level Pipeline



DATA FLOW

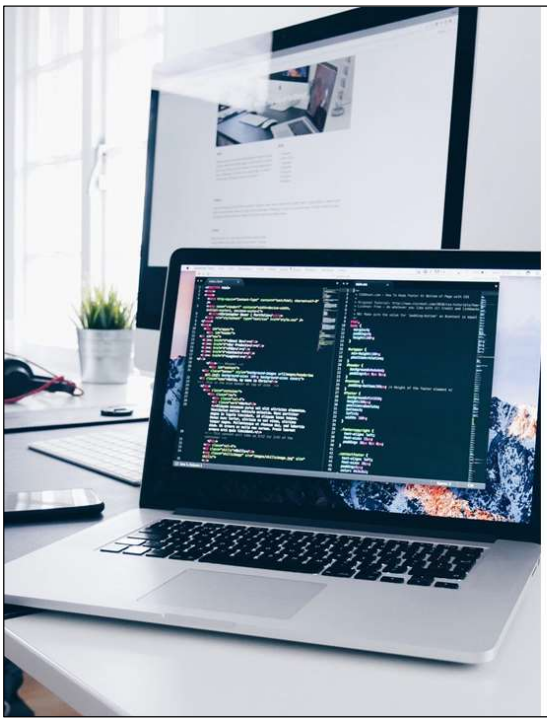
1. **Data Ingestion**
   Cron job or some time function to call the Twitter API at regular intervals

1. **Clean and process data**
   Some of the data will be formatted and relevant information will be saved to the database such as timestamp, GPS coordinates, etc.
   Some of the data will be processed using machine learning models.

1. **Save data in the database**
   Thereafter, the result from ML models will be saved to the database.

1. **Visualize data**
   The database will be connected to a monitoring dashboard which will update automatically every time the database is updated.

*The ML models used in this process will go through all steps of data exploration, data cleaning, modelling and evaluation in addition to deployment. We are yet evaluate if we will need a server framework to connect all parts of our pipeline.



System Design

1. Constraints
2. Data structure and capacity
3. Components of solution design
4. Performance, maintenance & cost optimisation
5. Stages of Development
   a. Make it work - Local (Sprint 4-7)
   b. Make it resilient (Sprint 8-9)
   c. Make it scalable (out of scope)

PAGE 78

Constraints

- **Low cost**
  Our first constraint is cost. Since we do not have a budget for this project, we will look for free alternatives.
- **Low Maintenance**
  Secondly, we want to focus on the data and modelling aspect in this project and look for suitable solutions that are low maintenance and do not require any upkeep from our team.
- **Non-critical workload**
  We understand that our work is non-critical to the business process. While looking for solutions that do not have downtime, we will prioritize cost and maintenance aspects for our solutions.
- **Storage**
  Finally, depending on data retention period, we will consider the storage capacity for our solution as well.

Data Structure and Capacity

Field	Type	Size (UTF-8 encoding / Length	Example
Created At	Timestamp	24 bytes	2020-09-18 21:56:20.798000
Coordinates	POINT / Spatial Data	16 bytes	
Text	String	280 chats / 280 bytes	'@DSisourath The Thameslink core between London St Pancras and London Blackfriars in rush hours only but the in cab rubbish rarely works'
Sentiments	String	50	Positive, Negative, Neutral
Components	String	50	Delays, others
Tweet ID	uuid	20 bytes	1227640996038684673

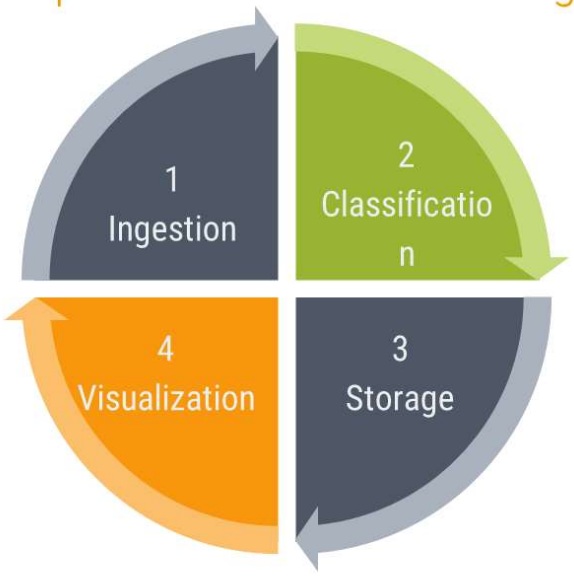
Total size: 1.24 KB

This is what our data looks like once the processing is complete. We have defined the data types for optimizing the storage.

Our current dataset has 16K tweets. Each tweet on the upper size is 2 KBs in size.

If we have a retention rate of 5 years and assume 500 tweets/hour, we anticipate the need of about 43.8 GB of data storage.

Components of Solution Design



PAGE 80

1. Data Ingestion

Data Ingestion

Local
Local script executed on demand that inserts tweets into database
Con - Needs to be executed manually

Cron
Cron job to pull Twitter on intervals
Con - Since it is a local solution, any system failure can affect the job execution

Server Application
HTTP Server Application that pulls twitter on time based intervals
It can be hosted either locally or on the cloud.
Pro - Have some prior experience
Con - No need for such infrastructure with a dedicated instance on which server application is running for our product. Expensive.

Serverless Function
Serverless functions pulls Twitter data on time based intervals
Pro - Cost efficient & low maintenance
Options: Azure Serverless, AWS Lambda

Selected Option

2. Model

Model

Server Application
Pre-trained model on server application
Pro - Have some prior experience building such solution
Dedicated instance is not needed and can be expensive

Model as a Service
Pre-packaged model as a service
Ex: Amazon Sage Maker
Con - Very expensive considering our cost constraint

Serverless Function
Serverless functions execute ML models when triggered
Pro - Cost Efficient & low maintenance
Options: Azure Serverless, AWS Lambda

Selected Option

PAGE 81

3. Database

Database

On Premise
Example: Mongo DB
Pro - Free
Con - System failure can lead to data loss

On the Cloud
Example: Mongo DB
Pro - Data will be highly available
Con - Expensive to host data

Serverless
Example: Dynamo DB, AWS RDS, Azure SQL Database
Pro - Highly available data and lower cost

Selected Option

4. Visualization

Visualization

Grafana
Pro - Easy to create, explore & share dashboard with your team and work well with time series data
Con - No prior experience

Looker Studio
Pro - Easier to create reports understand familiarity with Google Suite
Con - No prior experience

Power BI
Pro - Re-use the mockup Dashboard
Con - Unaware if dashboard can be shared

Selected Option

PAGE 82

Operational Excellence

Performance & Cost Optimisation

- Azure Timer Trigger, serverless functions and managed databases

Since our work is non-critical to the business operation, we will use Azure Timer Trigger, serverless functions and managed databases, all of which will help reduce operational and maintenance costs while allowing more focus to optimize the Machine Learning models and report services.

DB Cleanup (Out of scope)

- After every ingestion, run a function to delete data that is older than RETENTION_PERIOD
  - which will prove costly as we will double the size of our calls to functions
- Run a function to delete data that is older than RETENTION_PERIOD once per year
  - it'll be fewer calls but we risk storing data for longer than needed which could be significant if there are many tweets in that year.

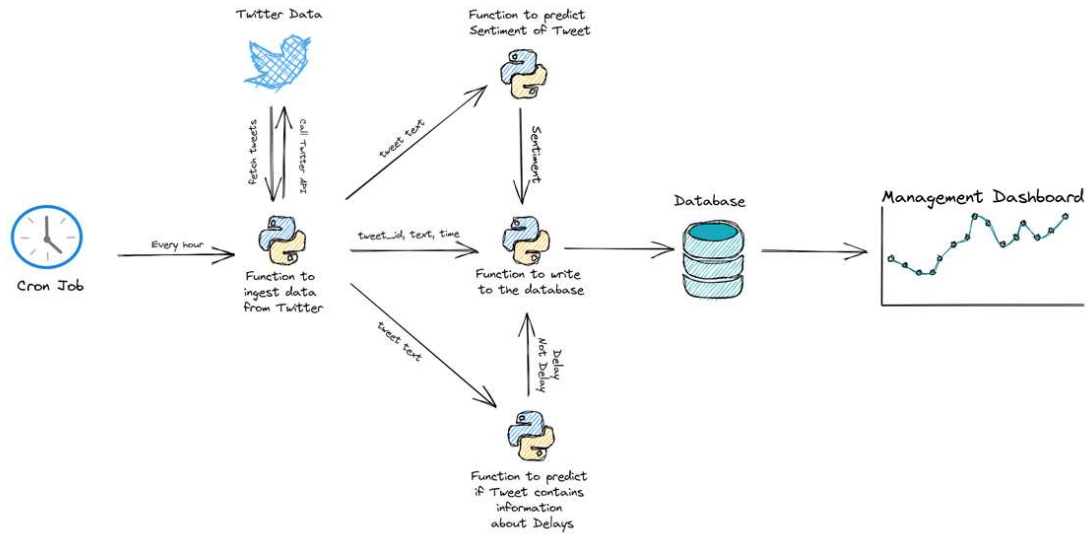
Monitoring (Out of scope)

- Monitoring service that tracks failures or errors in Serverless Functions. Options include:
  - Azure Monitor
  - AWS Cloudwatch

PAGE 83

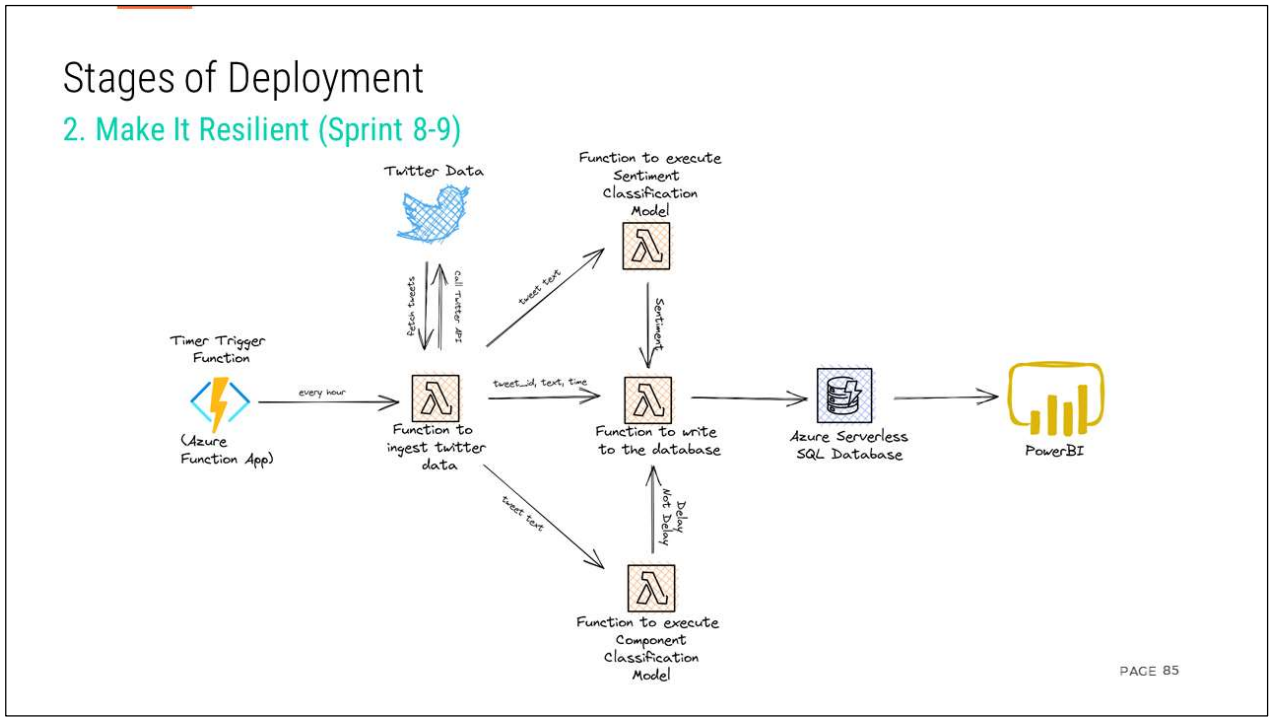
Stages of Deployment

1. Make It Work - Local (Sprint 4-7)



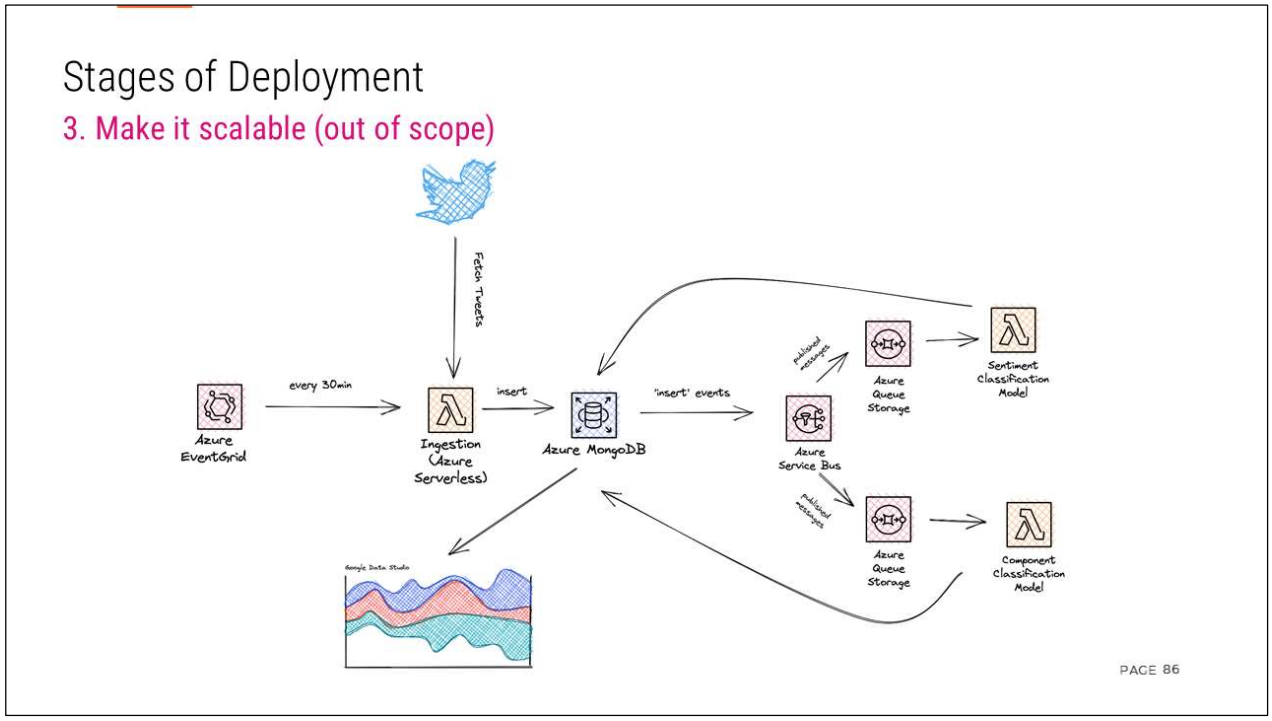
Our first milestone is to make it work locally.

We will have a local cron job that will call a python script. Part of the script will call the Twitter API for the required data which will be preliminarily cleaned. The script will also run both models on the user text and all this data will be written to the database. The dashboard will be auto-updated from the database.



The next milestone is to make it resilient.

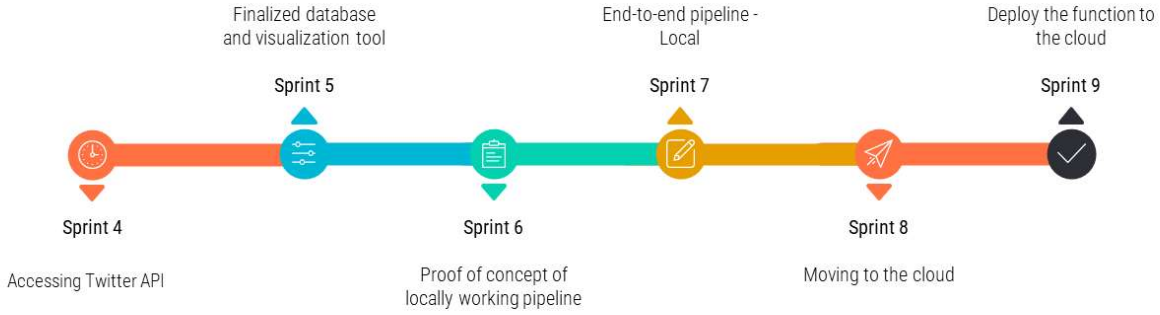
Here, we will have a Timer Trigger to call a serverless function every 60 mins. A part of this function will ingest data by calling the Twitter API. Another part of the function will trigger the model functions, which will classify if there are delays mentioned in the tweets and the sentiment of the tweets. All data will then be written to the database. Finally, the database will be connected to the dashboard and auto-updated every run.

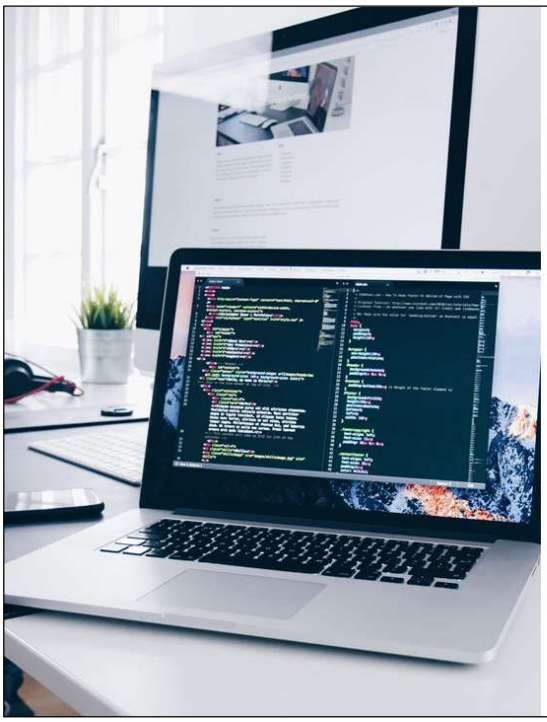


The next milestone is to make it scalable – this, however, is out of scope for the duration of our project.

The entire system is the same as the previous one. However, in this iteration, we will decouple our models from the ingestion system. This will be beneficial when we want to patch or make changes to our models as we avoid down time. In case our model fails, we will also retain data in queues.

Execution

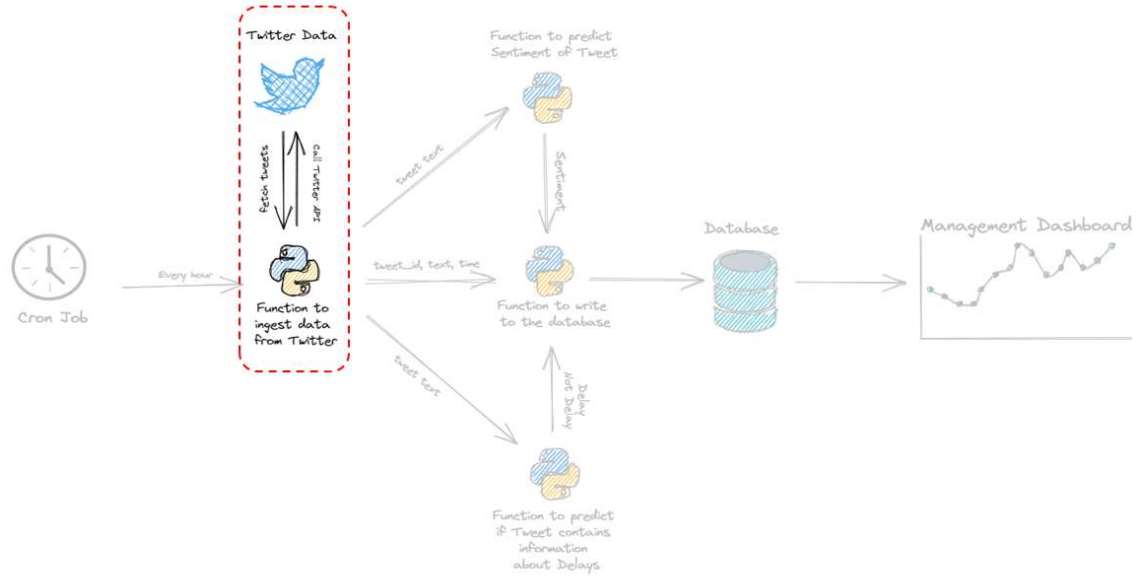




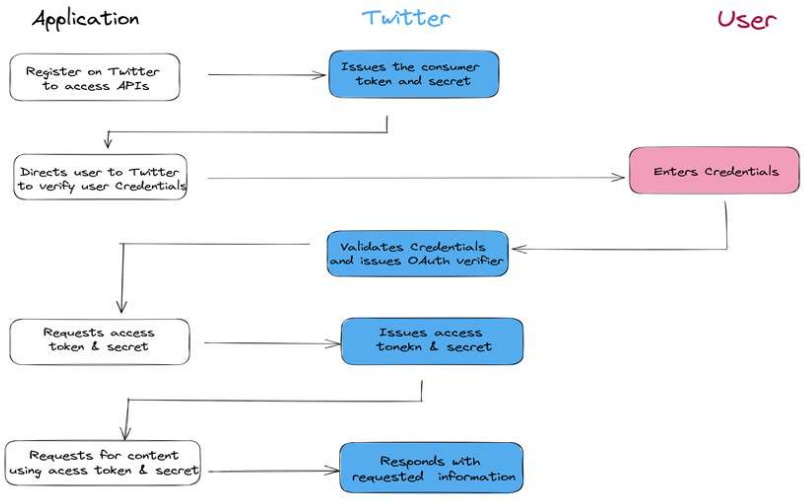
Sprint 4 - Accessing Twitter API

1. Part of pipeline built in Sprint 4
2. Accessing Twitter API
3. Steps
4. Result of accessing Twitter API
5. Impediment to cron job execution on local machine

1. Part of Pipeline built in Sprint 4



2. Accessing Twitter API



Accessing Twitter API has 3 components:

- Application we are building
- Twitter and
- Us as user

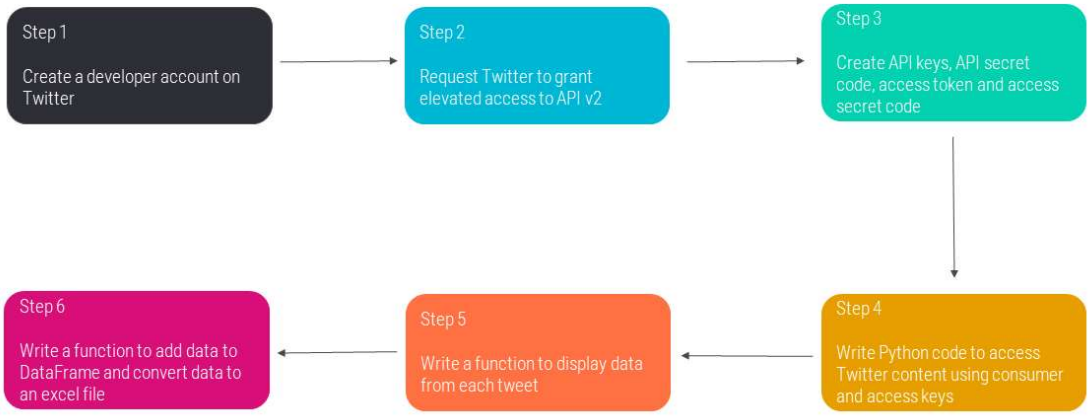
To access Twitter API, we first register the application on Twitter.Twitter then issues a consumer token and consumer secret code.

Then, user is redirected to Twitter to verify their credentials. Once user enters login information and address correctly, Twitter will validate the credentials and grant an OAuth verifier.

Using this we can request access for token and secret code. If everything is in order, Twitter will issue access token and secret code.

Thereafter, we can use the consumer, access token and secret code to request content from Twitter, which will respond with the information.

3. Steps



PAGE 91

4. Result of accessing Twitter API

	Text	Location	Place
	London ok with Thameslink, Jubilee Line, Overground, Docklands LR, Thameslink 2, Crossrail etc? https://t.co/StONsrhCrt		53.666112], [-1.5528479, 53.666112], [-1.5528479, 53.698671], [-1.5957296, 53.698671]]]), attributes={})
511	@guardian HS2 10% of transport budget, x2 spent on 4,000 miles new roads #RIS2. Scotland & north of England getting same capacity to London as Brussels, Paris & Brighton essential to level up. London ok with Thameslink, Jubilee Line, Overground, Docklands LR, Thameslink 2, Crossrail etc?	Benevento, Campania	Place(_api=<tweepy.api.API object at 0x000001615AF5A280>, id='4efd6fc8c50fae33', url='https://api.twitter.com/1.1/geo/id/4efd6fc8c50fae33.json', place_type='admin', name='Yorkshire and The Humber', full_name='Yorkshire and The Humber, England', country_code='GB', country='United Kingdom', contained_within=[], bounding_box=BoundingBox(_api=<tweepy.api.API object at 0x000001615AF5A280>, type='Polygon', coordinates=[[[-2.56475248726412, 53.3015341502953], [0.149787002205902, 53.3015341502953], [0.149787002205902, 54.5621550270294], [-2.56475248726412, 54.5621550270294]]]), attributes={})
512	(Thameslink Update) 19:25 Brighton to Bedford due 21:51 - 19:25 Brighton to Bedford due 21:51 will be started from St Albans City.	London, England	
513	@jamezymaryan @StevenageFC @officialKLtown Not affected on Thameslink apparently	United Kingdom	
514	In a UK-first, accessibility advisors at @BrightonHoveBus and Govia Thameslink Railway (GTR) have worked together with adults with a learning disability to support them in making an integrated journey by both bus and train.	Edinburgh	
515	@neilbays I'm surprised they ran in only a couple of mins ahead of the Thameslink service, thought there was a time interval? Remember first year of water jetting and a 465 going skating through BEX by about a mile following it.	Kent	
516	I'm now able to get the Elizabeth line to and from work, which has the unexpected benefit of actually getting a seat on Thameslink	Croydon, London	Place(_api=<tweepy.api.API object at 0x000001615AF5A280>, id='3eb2c704fe8a50cb', url='https://api.twitter.com/1.1/geo/id/3eb2c704fe8a50cb.json', place_type='city', name='City of London', full_name='City of London, London', country_code='GB', country='United Kingdom', contained_within=[], bounding_box=BoundingBox(_api=<tweepy.api.API object at 0x000001615AF5A280>, type='Polygon', coordinates=[[[-0.112442, 51.5068], [-0.0733794, 51.5068], [-0.0733794, 51.522161], [-0.112442, 51.522161]]]), attributes={})
517	@JordWright93 @TLRailUK This page has a useful diagram part way down,	Hatfield, Herts	

PAGE 92

5. Impediment to cron job execution on local machine

Inability to debug cron job on local machine

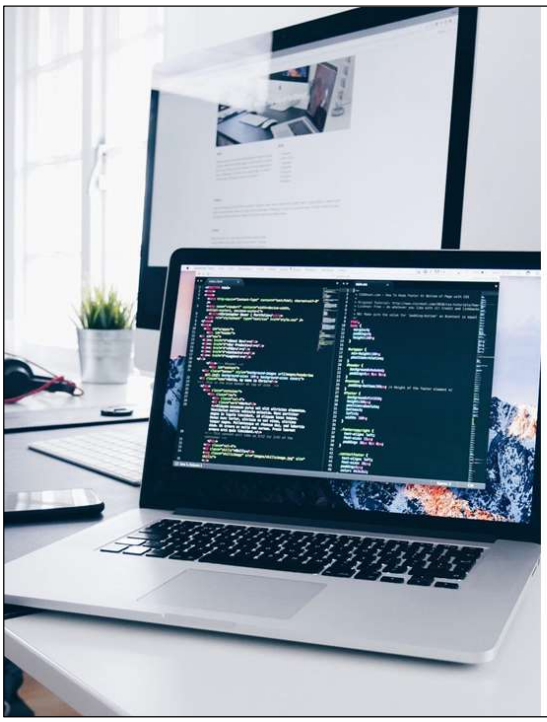
```
rashmi@DESKTOP-4R71HAB:/mnt/e/Data Lab/Final tweets$ sudo service cron start
 * Starting periodic command scheduler cron
rashmi@DESKTOP-4R71HAB:/mnt/e/Data Lab/Final tweets$ service cron status
 * cron is running
```

As part of our deliverables, we started working on a cron job set on our local machine to pull data from Twitter API on a daily basis.

We started with trying this approach on powershell which was taking too long to debug. So, we installed WSL, which is Windows subsystem for Linux. We, then, were able to run the cron as seen in the above image. The cron job was successfully scheduled but didn't execute. We tried to fix it by setting the right permissions, verifying file paths, etc. but it seemed like it was a WSL specific problem.

Since we had spent a lot of time trying to make it work on our local machine for a task that can easily be done manually and our end goal was to move to the cloud anyways, we would not pursue this and try to get this functionality to work using a serverless function.

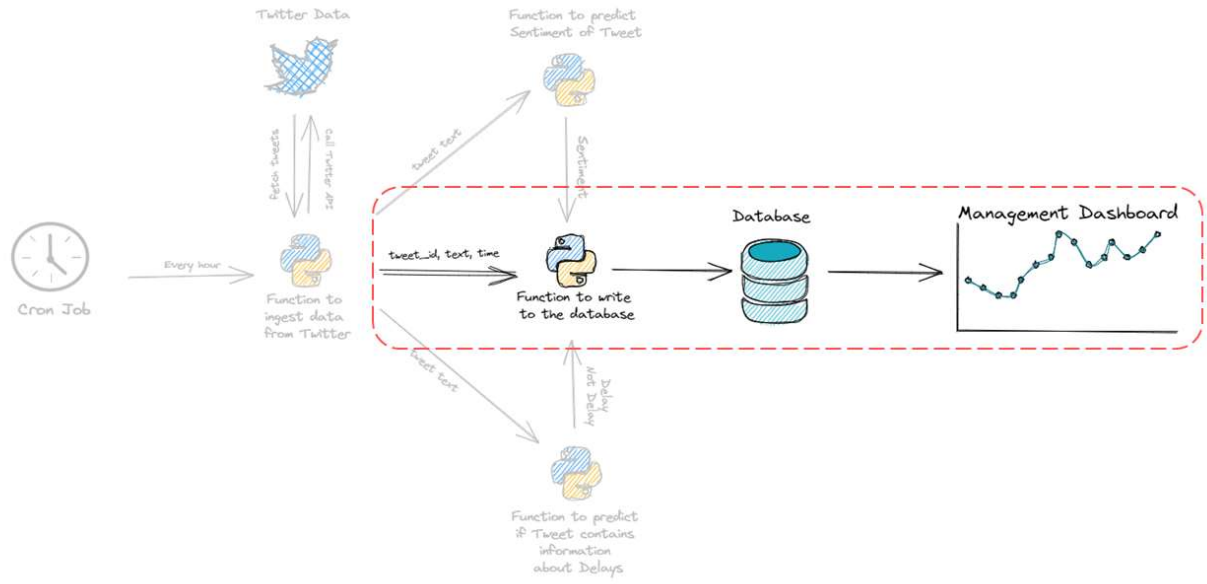
The risk with this approach is that we do not have any prior experience in a serverless environment. So, we will mitigate this risk by gaining knowledge from online tutorials and in the worst case scenario that it also does not work, we will pull the data from Twitter using a function that runs on intervals.



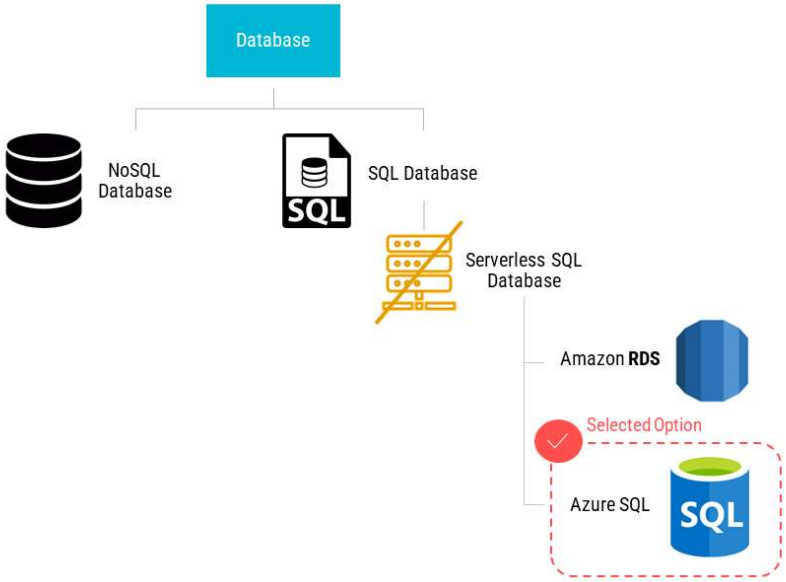
Sprint 5 - Finalized database and visualization tool

1. Part of pipeline built in Sprint 5
2. Finalized database and visualization tool

1. Part of pipeline built in Sprint 5



2. Finalized database and visualization tool - Database

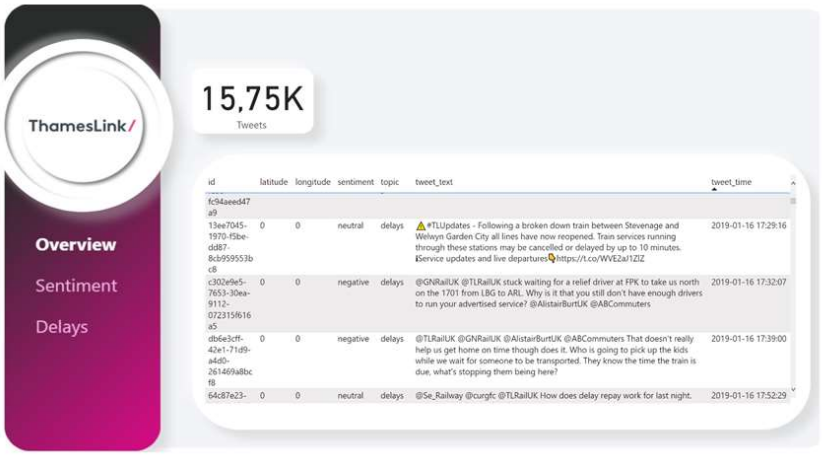


Since we were not clear on the type of database structure we would need for this project, we started considering a NoSQL database, like MongoDB or Cassandra. However, as the project progresses, it becomes clearer the structure we will need and we would be able to operate within the constraints of schemas of relational databases. We also have to consider that not all visualization tools support NoSQL databases. Hence, we opt to build a SQL database. As we do not want to maintain and manage our database, we opt for a serverless SQL database which would also provide us with cost optimization.

The 2 options that we consider while keeping in mind our constraints are Amazon RDS and Azure SQL. According to our research, Azure is more user-friendly platform. Also, for the choice of database, it is important to consider the visualization tool we will use. Since one of our choices was Power BI, connecting Azure to another Microsoft product is also easier. Hence, our final serverless database will be built using Azure SQL database services.

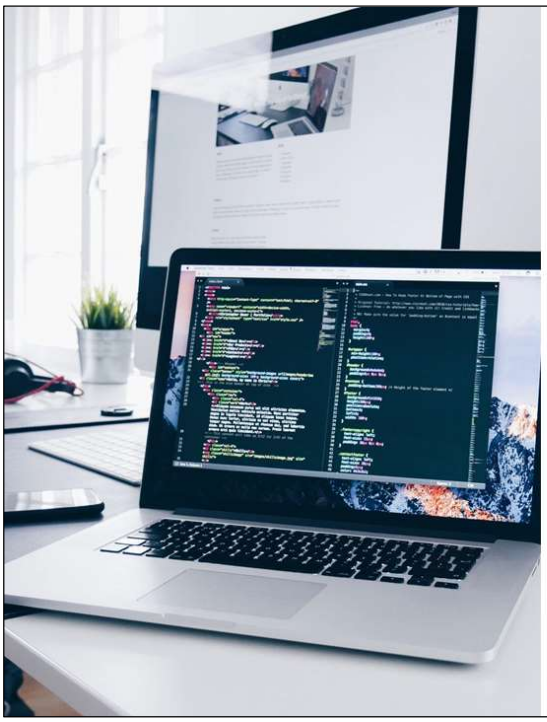
2. Finalized database and visualization tool - Visualization

Power BI



For visualization, we weighed various options. But since our mockup dashboard was built using Power BI, we want to continue using the same tool in order to not redo the entire dashboard. We connect the existing dashboard to our database and re-adjust our landing page to get the result above.

PAGE 97

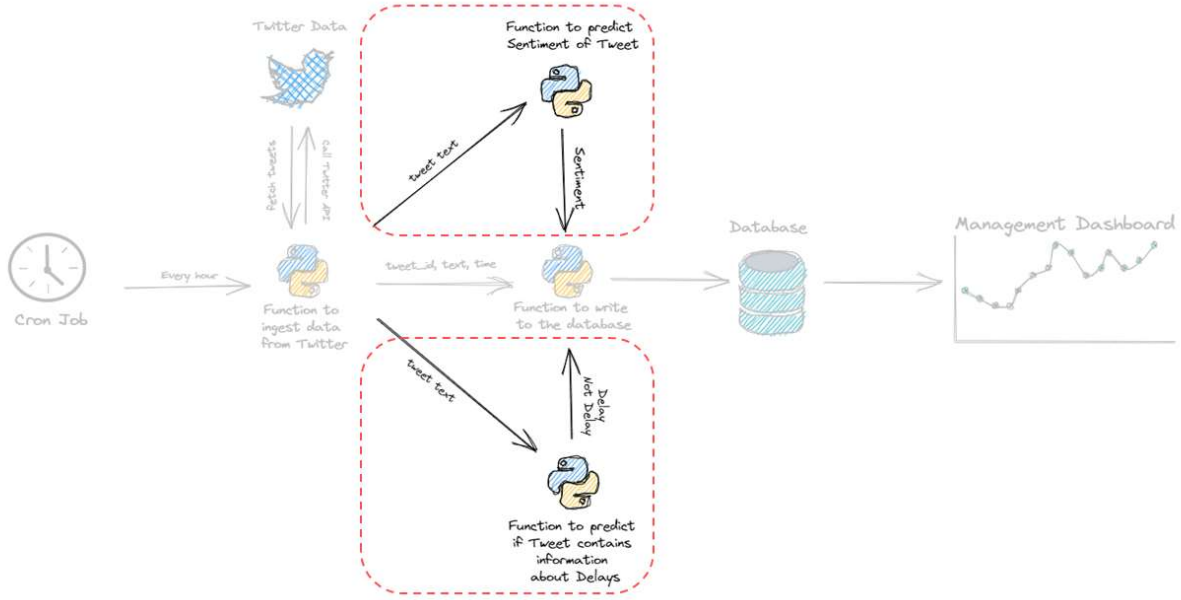


Sprint 6 - Proof of Concept of locally working pipeline

1. Part of pipeline built in Sprint 6
2. Outputs of our local scripts
3. Challenges & solutions

PAGE 98

1. Part of pipeline built in Sprint 6



2. Outputs of our local scripts

We pulled 10 tweets from on December 06, 2022 and predicted the sentiment and topics.

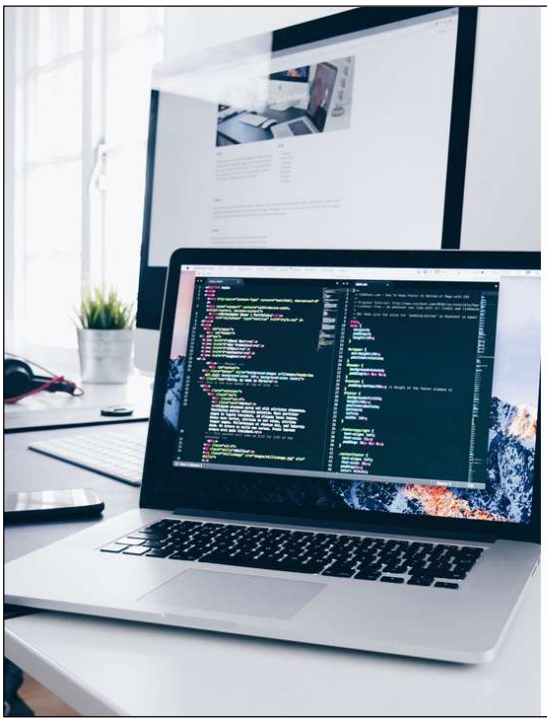
Here is an example of what our output DataFrame looks like.

	id	time	location	text	Sentiment	Topic
0	1600052125047717888	2022-12-06 08:58:48+00:00		!!!!CODE RED!!!!\n\nIntel says Thameslink @TLR...	negative	Not Delay
1	1600051954486583296	2022-12-06 08:58:08+00:00		@BBCr4today @TimesRadio @RMTunion Can somebody...	negative	Not Delay
2	1600049823436062721	2022-12-06 08:49:40+00:00		Replacement buses will run between Clapham Jun...	neutral	Not Delay
3	1600049412100673536	2022-12-06 08:48:01+00:00	Tayside	I've tagged Thameslink @TLRailUK multiple time...	neutral	Not Delay
4	1600049205522808834	2022-12-06 08:47:12+00:00		@realdanielloman @RMTunion They operate withou...	neutral	Delay
5	1600047670009135105	2022-12-06 08:41:06+00:00		!!!!CODE RED!!!!\n\nIntel says Thameslink @TLR...	negative	Not Delay
6	1600046325566828544	2022-12-06 08:35:46+00:00	Mill Hill, Barnet, London, NW7	I yearn for the days Thameslink was called the...	negative	Not Delay
7	1600045454699286528	2022-12-06 08:32:18+00:00		@vampywitchy @SCynic1 @GillianLazarus I have n...	negative	Not Delay
8	1600044826283712518	2022-12-06 08:29:48+00:00	Shropshire	!!!!CODE RED!!!!\n\nIntel says Thameslink @TLR...	negative	Not Delay
9	1600044644410105856	2022-12-06 08:29:05+00:00		!!!!CODE RED!!!!\n\nIntel says Thameslink @TLR...	negative	Not Delay

3. Challenges & Solutions

No.	Problems	Possible Solutions	Risk Assessment
1	The final sentiment analysis model uses tokenization that is built using the training data	1. Find a replacement to tokenization method 2. Deploy training data along with model (Not ideal)	→ Deploying data may not work with the serverless functions solutions and we may need to switch to a server framework
2	The final sentiment analysis model uses encoding and decoding the target variable that is built using the training data	1. Find a replacement to encoding method 2. Deploy training data along with model (Not ideal)	→ Deploying data may not work with the serverless functions solutions and we may need to switch to a server framework
3	The best delay classification model uses CountVectorizer that is built using the training data	1. Find a replacement to CountVectorizer method 2. Deploy training data along with model (Not ideal)	→ Deploying data may not work with the serverless functions solutions and we may need to switch to a server framework
4	Missing GPS coordination for most tweets	1. Extract latitude and longitude wherever available 2. For places that have latitude and longitude missing, convert the location provided in the tweets to latitude and longitude 3. Remove location from the scope of the dashboard	The tweets' locations may not be the same as users' locations

PAGE 101

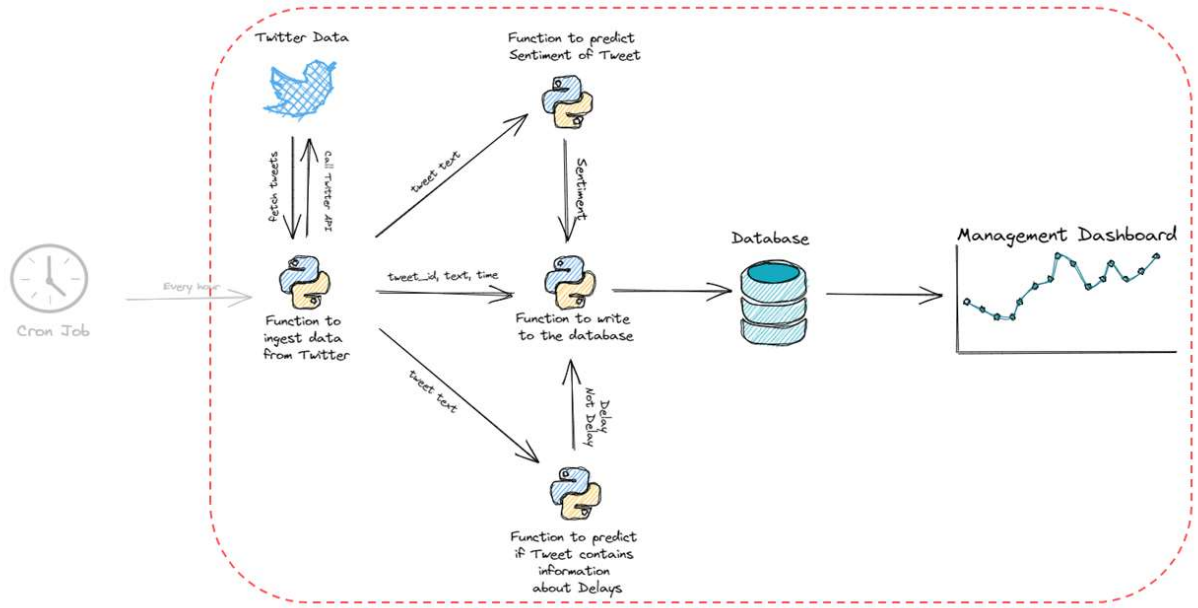


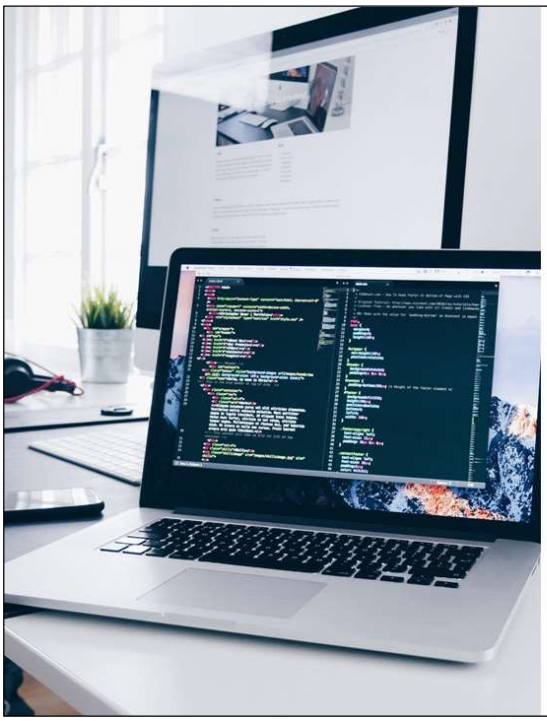
Sprint 7 - End to End Pipeline - Local

Part of pipeline built in Sprint 7

PAGE 102

Part of pipeline built in Sprint 7



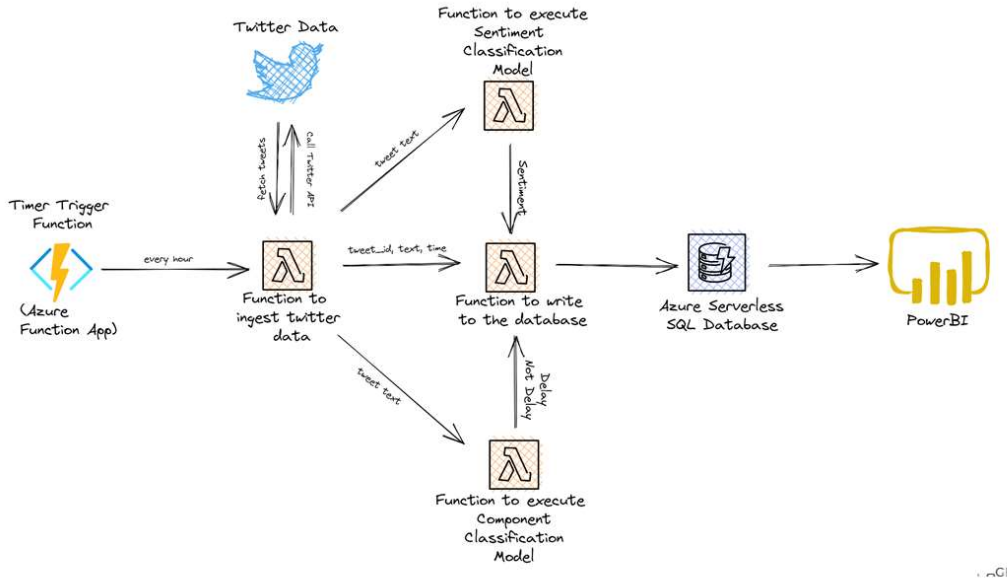


Sprint 8 - Moving to the cloud

1. Final system design for the cloud
2. Azure functions
    - Options
    - Phases
3. Progress
4. Challenges & solutions

PAGE 104

1. Final system design for the cloud



2. Azure Functions - Options

Advantages

- Serverless Solution
- Write less code
- Maintain less infrastructure
- Cloud provides up-to-date resources needs to keep the application running
- Scales up and down automatically
- Pay for what you use

Options

Azure functions deployed from local version

Azure functions in Azure Portal

2. Azure Functions - Phases

- VS Code and Azure Functions extension
- **Language**: Python
- **Trigger**:
  - **TimerTrigger**
  - CosmosDBTrigger
  - BlobTrigger
  - QueueTrigger
  - EventGridTrigger
  - EventHubTrigger
  - ServiceBusTopicTrigger

Phases

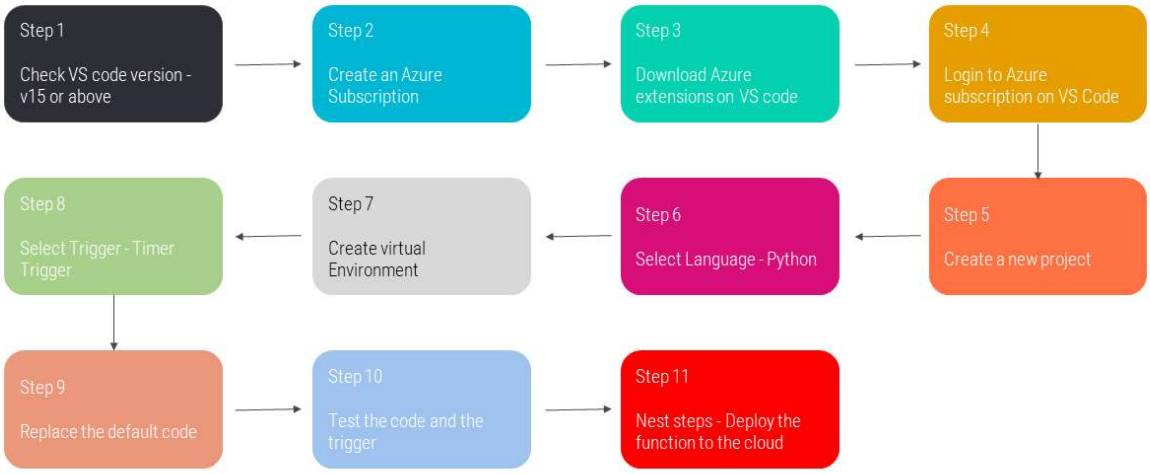
Phase 1 - Make the function work locally using VS code

Executed in this Sprint

Phase 2 - Deploy the function to the cloud

PAGE 107

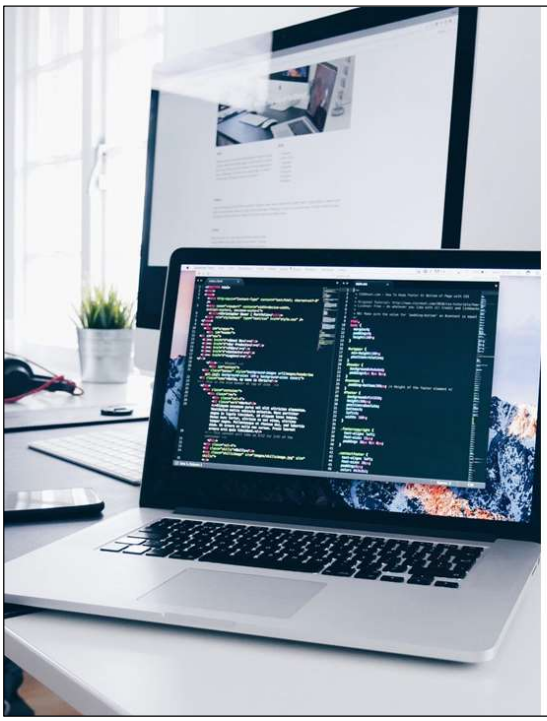
3. Progress so far



PAGE 108

4. Challenges & Solutions

No	Problems	Possible Solutions	Risk Assessment
1	A single function execution has a maximum of 5 minutes by default to execute. If the function is running longer than the maximum timeout, then Azure functions runtime can end the process at any point after the maximum timeout has been reached.	1. Run the function in smaller batches	➔ We may lose some data if the task is dropped, but given the size of tweets we have observed over the last 2 weeks, the risk is low
2	Our Azure subscription is a free student account with limited credit. We may run out of credits	1. Shut down the function when we run out of credits	➔ We will run the function till the report submission data, and as per cost forecast by Azure we will not run of credits.
3	Vendor-lock is the biggest drawback of this function. It is very difficult to run code deployed in Azure function outside Azure environment.		➔ We do not anticipate going into production and hence we can re-create the project with a different stack.
4	Upload limit on files over 100 MBs. Our sentiment analysis model is over 107 MBs.	1. Save the model in a blog storage 2. Move to a server framework	➔ As per our research, we can move the larger files to a blog storage option offered by Azure.

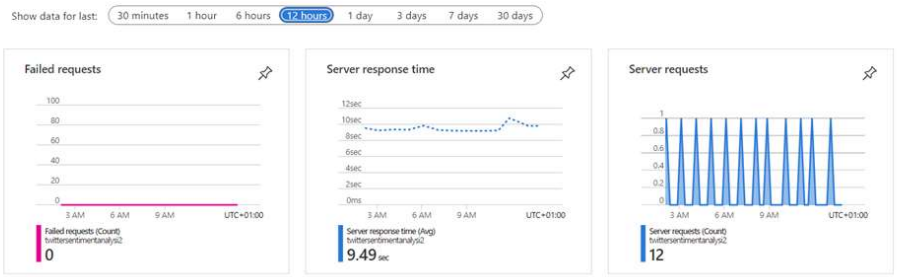


Sprint 9 - Deploy the function to the cloud

1. Phase 2 - Deploy function to the cloud & monitor the cloud system
2. Final dashboard
3. Future development

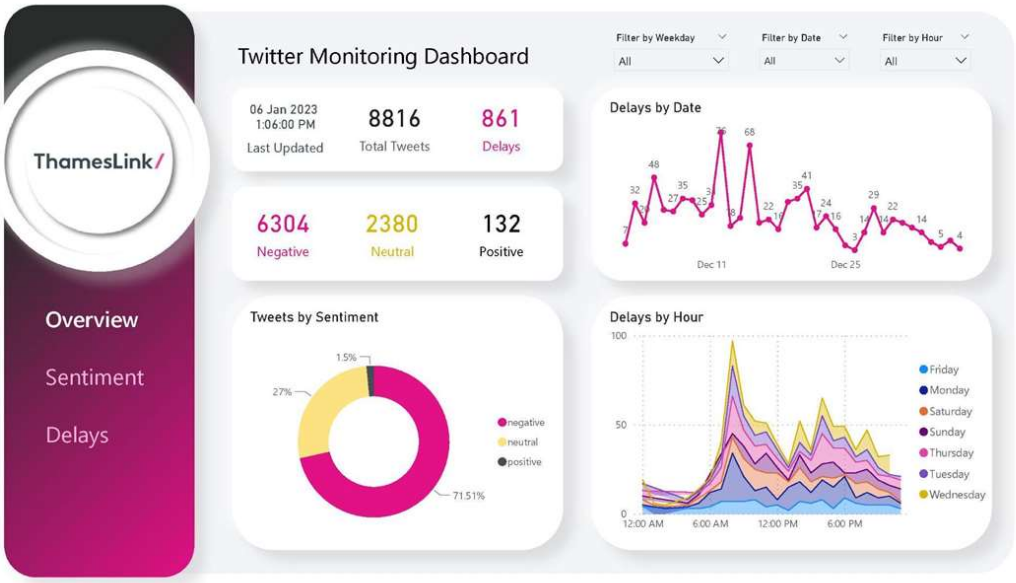
1. Phase 2 - Deploy Function to the Cloud & Monitor the Cloud System

Snapshot of Azure platform interface to monitor the function calls, server response time and failed requests

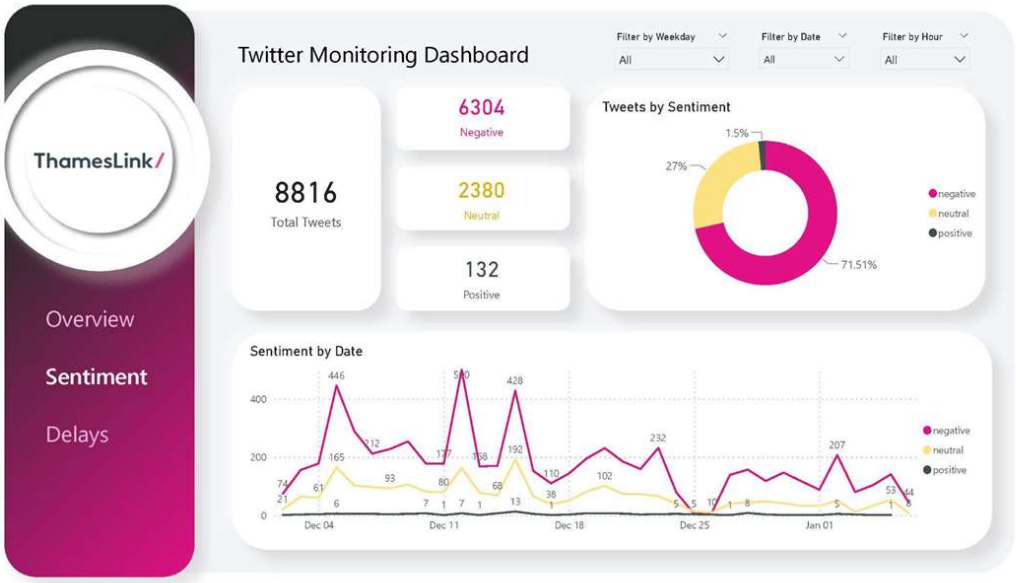


Snapshot of Azure Platform interface to see the Cost of running the function app for the last 30 days

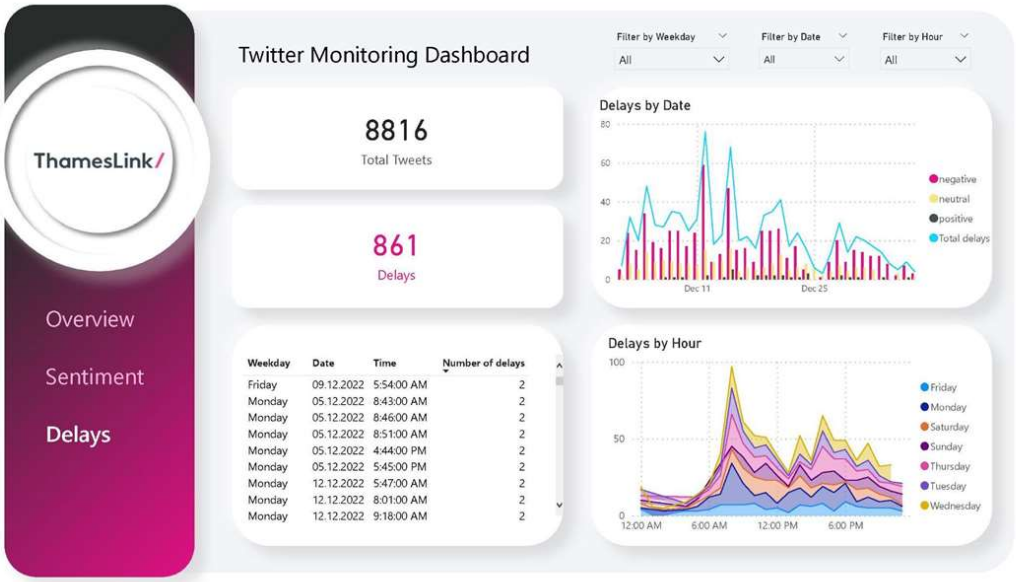
2. Final Dashboard



2. Final Dashboard



2. Final Dashboard



3. Future Developments

- Continue to develop Feature 3.4 *(Compare statistics in current & previous periods)* with statistics such as MoM, QoQ and YoY once enough data is collected
- Continue to develop Feature 3.3 *(track vehicle location)*
- Continue to develop Epic 4 *(Classify other topics)* and its features
- Improve the results of Epic 2 *(Classify sentiment)* and Epic 3 *(Classify topic "delay")* by
    - Make the dataset larger and re-train models on the new dataset
    - Try other models for sentiment analysis such as BERT
    - Try other models for delay classification such as BERTopic
- Make the pipeline scalable
- Implement monitoring of deployed models