CST3130 Coursework 2 – Data Visualisation Website Final Report

# Website Description

# This website is for viewing the historical mean monthly temperature of five locations in the UK over the last hundred years or so and displaying the results of machine learning predictions of the future temperatures. Due to limitations in my source data, the predicted temperatures are for this current year 2021 as the source seems to only update year by year.

It also shows the result of sentiment analysis of Twitter tweets related to the chosen locations although due to the nature of Twitter the contents of the tweets are likely to be marginally related to the topic and have no effect on the weather.

The machine learning takes a set of data of temperatures and the date of the first piece of data and uses it to create a model for predicting future data. I then use an endpoint to query it for predictions and insert the predictions into graphs.

The sentiment analysis lambda function is set to trigger whenever one or more tweets are inserted into a DynamoDB table. The tweets’ data is accessed and the text of the tweet is sent through AWS Comprehend and the result stored in another table.

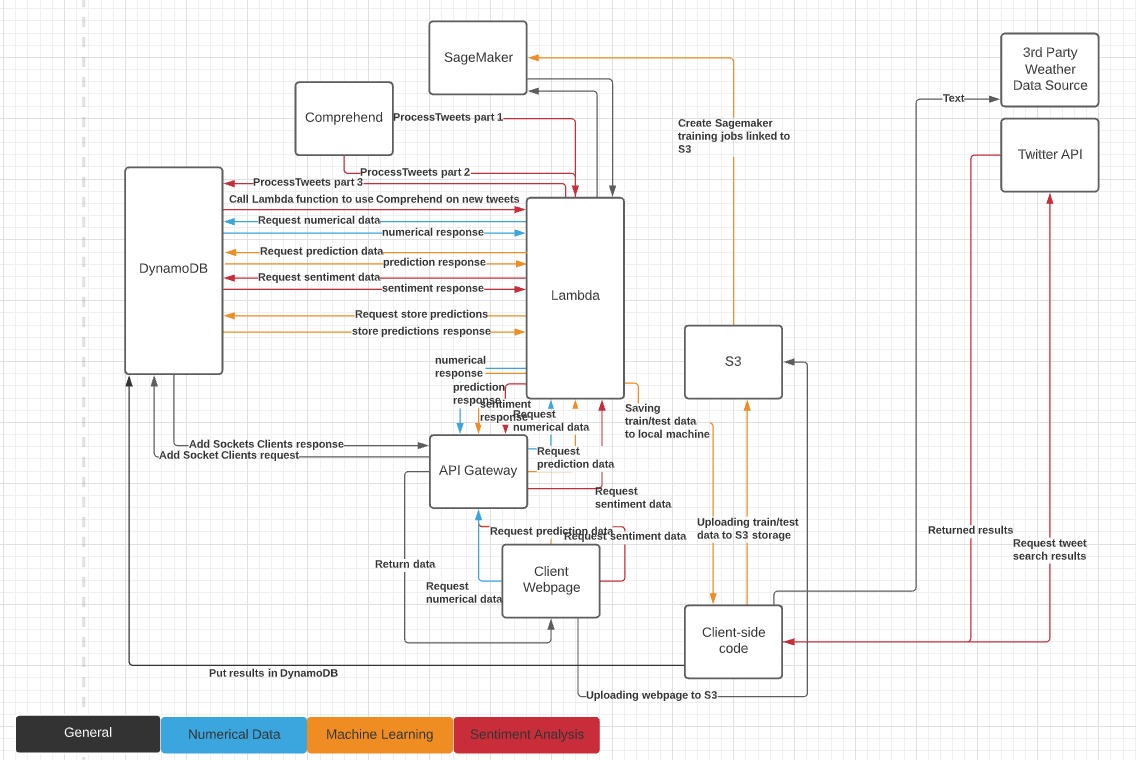
The website relies on several tables in DynamoDB

* SageMakerResults for storing the predictions made by SageMaker so the website doesn’t have to requery the endpoint for each new client (and so I can delete the endpoints for reduced costs)
* SentimentResults for storing the sentiment results made by Comprehend
* TweetData to store the responses from the Twitter API
* WeatherData to store the data from the original source in a searchable format
* WebSocketsClients to keep track of the Web Socket connection ID of connected clients

It also uses 7 Lambda functions to perform tasks

* processTweets fetches the tweetID, location, tweetText and timestamp from new tweets in TweetData, runs the text through Comprehend and inserts the results into SentimentResults
* DataupdateClients fetches all active IDs from WebSocketsClients and for each ID found, sends them an array of *n* recent numerical, prediction and sentiment data in JSON format from WeatherData, SageMakerResults and SentimentResults respectively
* DatarequestData fetches the latest *n* entries from WeatherData, SageMakerResults or SentimentResults depending on what was requested. It then sends the data to the client in JSON format The client is coded to request all 3 in sequence, the only reason it’s split up is that all together the data is too much to send through the Web Socket all at once.
* DataConnect and DataDefault are used for the connect and default routes of the API for the website in API Gateway
* FetchPredictions opens files of test data, formats them into the input format for SageMaker, then inserts the results from SageMaker into SageMakerResults
* MachineLearning fetches the numerical data from WeatherData, reformats them into the format needed for SageMaker and prints them to the console for saving and manual upload to a S3 bucket.

# Architecture Relationship Diagram (Figure 1)



# Front End Screenshots

Quick foreword before the screenshots: Yes, I know the graphs are tiny, the borders around them for the labels are much bigger than they appear to be. If I increase the width to 700 from 600 the legends for the pie charts and half the legends for the graphs get cropped off.

I could make them bigger by rearranging the layout but then it no longer matches the proposal design.

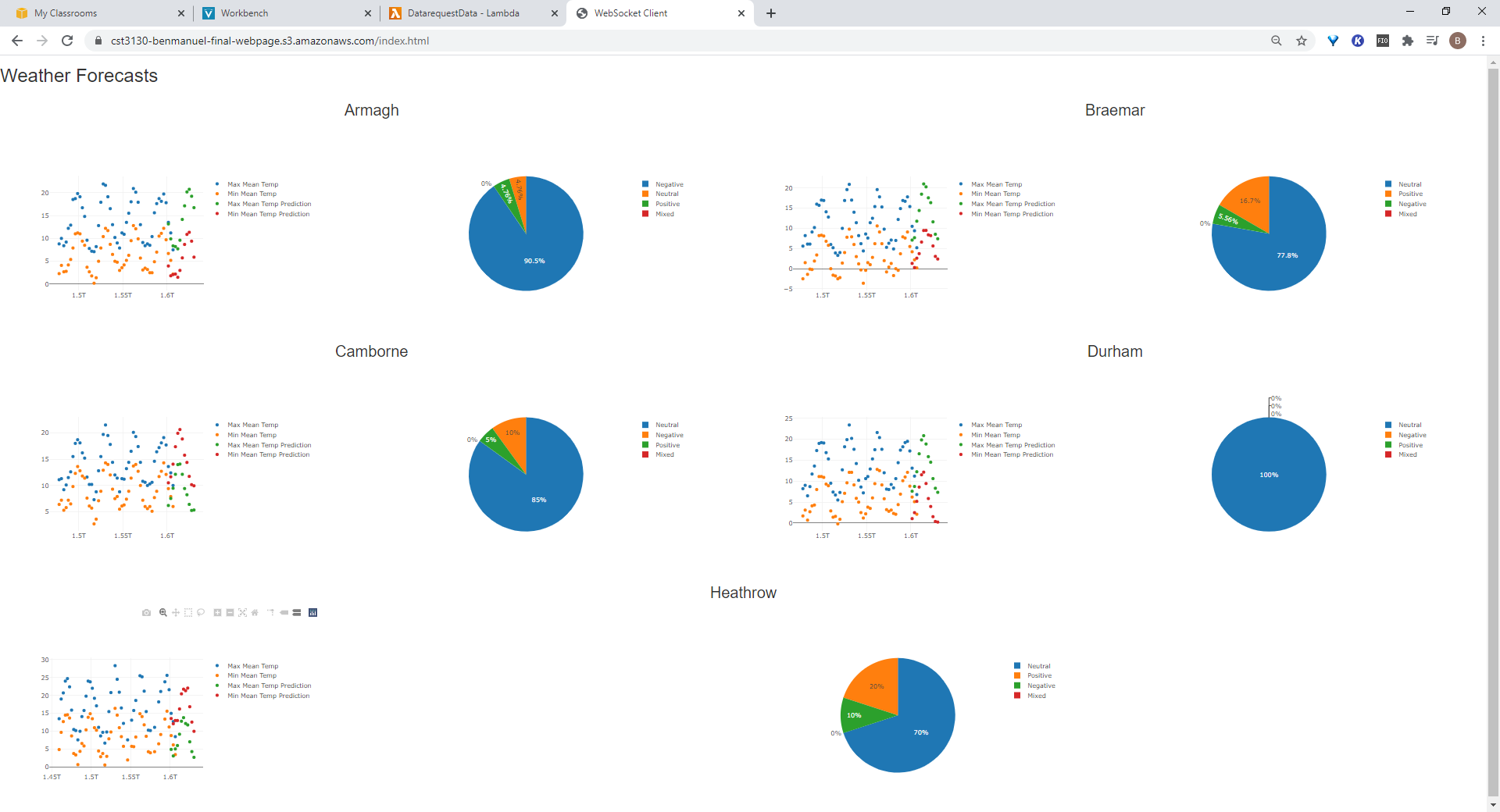
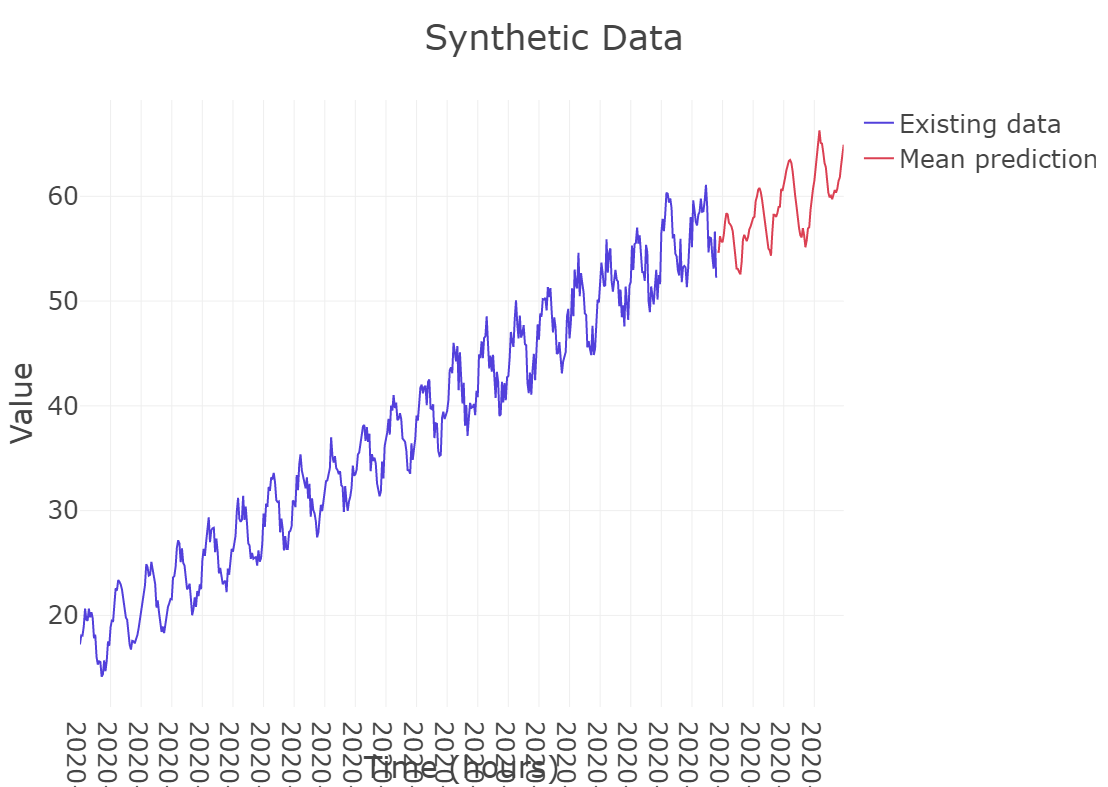


Figure 2. Screenshot of front end hosted on Amazon Web Services S3 Storage.

# Synthetic Data Visualisation

Figure 3. Synthetic data graph hosted by Plotly using data stored on Amazon Web Services.