**CCT College Dublin**

**Assessment Cover Page**

*To be provided separately as a word doc for students to include with every submission*

|  |  |
| --- | --- |
| **Module Title:** | Advanced Data Analysis  Big Data Storage & Processing |
| **Assessment Title:** | CA-2 |
| **Lecturer Name:** | David McQuaid,Dr. Muhammad Iqbal |
| **Student Full Name:** | Ali Hussnain |
| **Student Number:** | 2024639 |
| **Assessment Due Date:** | 23 MAY 2025 |
| **Date of Submission:** | 22 MAY 2025 |

**Declaration**

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

[LSTM-Based Forecasting and Dashboard Visualization of TSLA Stock Prices](#_p5h9y2bza092)

[Introduction](#_6xz45dazsmmy)

[Dataset Description](#_uxr20hzgdhvk)

[Data Preprocessing and Sentiment Analysis](#_8b5ms15xpla0)

[Exploratory Data Analysis (EDA)](#_gj48w6uywxno)

[LSTM Model for Stock Forecasting](#_bl9mw0vtvmiu)

[Long Short-Term Memory (LSTM) Neural Network](#_1dajl4xqaqiw)

[Data Preparation for LSTM](#_s7zodi2lgida)

[LSTM Model Architecture](#_4v0roopi10rm)

[Evaluation and Prediction](#_rqub479xd7en)

[Insights](#_i7avxs18vod1)

[Evaluation and Dashboard Presentation](#_d8v9nvkykqsc)

[Model Evaluation](#_dyiu9v64brsa)

[Forecast Output](#_n6bxfcwg7f6t)

[Interactive Dashboard using Streamlit](#_a53epq8alerg)

[Dashboard features:](#_2pb3amopvfk2)

[Design Considerations and Tuft’s Principles](#_ahfdvp6z6xn9)

[Conclusion](#_sx1mwi2wcsm6)

[References](#_gf9lfbi0jxw8)

# 

# LSTM-Based Forecasting and Dashboard Visualization of TSLA Stock Prices

# Introduction

High levels of instability, varied behaviours and quick responses to data are common features of the financial market. Stock price forecasting has long been a difficult problem in both finance and data analytics. This project deals with this challenge by mixing data from social media tweets and historical prices. The target was to use machine learning and statistics, taking advantage of sentiment signals, to guess the closing price of Tesla (TSLA) over short periods of 1 day, 3 days and 1 week.

The study wanted to see if there was a link between what people say on Twitter and changes in stock prices. Many people now use what people say on social media as one of the factors when making investment decisions. Combining socio-economic factors with timeseries statistics, we intended to produce a forecasting model that catches changes in both markets and consumer behaviour.

All of the coding was done using Python in Jupyter Notebooks as the main local tool. We excluded big data infrastructure (such as Spark, Kafka or Hadoop) because it didn’t fit the scope and resources we had. Instead, the work was centred on developing a pipeline that cleans, combines and works on the dataset and uses both ARIMA and LSTM for forecasting. A Streamlit interactive dashboard was used to display the final results, so users could review the data on prices and assess the model’s predictive error.

The methodology and application are documented in this report, with sections covered for prepping the data, exploring analysis, measuring sentiments, creating models, assessing performance and presenting the data visually. The achievements from this study confirm that adding sentiment analysis to time-series forecasting is valuable and points to new ways to improve in the future..

# Dataset Description

### 

Two principal sources of information were included in the stock-tweet-and-price.zip file provided in this project: a collection of stock tweets and a data set with stock prices of 38 companies, among them, Tesla (TSLA). Since the data combines both unstructured social media entries with numerical time-series, it offers a powerful base for research in sentiment-based predictions.

The data for this file, stock tweet, covers 10,000 individual tweets released during January 2020 and December 2020. All of the records state:

* id: the unique identifier of the tweet
* date: the date the tweet was posted
* ticker: the stock ticker mentioned in the tweet (e.g., $TSLA)
* tweet: the actual tweet text

They provide the basis for obtaining public opinion on companies on set dates.

There are 38 CSV files as part of the dataset, each relating to a unique stock of a company. These records always include the columns described below.

* Date: the trading date
* Open: the opening price of the stock on that day
* High: the highest price reached
* Low: the lowest price reached
* Close: the price at market close (primary target variable)
* Adj Close: the adjusted closing price after stock splits/dividends
* Volume: the total number of shares traded on that day

The research for this project centred on Tesla (TSLA). The data files used in this study were just tsla.csv and those marked as $TSLA in stocktweet.csv. To allow a closer analysis of a single business and keep all the modelling and sentiment evaluation on track during the time available, this approach was chosen.

First, we loaded both the sources with Pandas. The fields with dates were made into datetime fields so data from both datasets could be grouped together. Only tweets that included the TSLA ticker were preserved and organised by date, allowing us to calculate the daily average of sentiment. The target for prediction in the study was the stock data's final price, called the Close price.

After doing the cleaning and combining, only three vital columns were kept in the final dataset.

* Date
* Close (TSLA stock closing price)
* Sentiment (aggregated average of VADER compound sentiment scores per day)

# Data Preprocessing and Sentiment Analysis

Machine learning or time-series forecasting projects rely heavily on doing data preprocessing. For this project, I cleaned both the text of the tweets and the numbers in the stock price data, assigned sentiment scores to the tweets as needed and combined the data sets by date. We relied on Pandas, NumPy and the VADER SentimentIntensityAnalyzer in the nltk library for this part of the project.

Before we could perform sentiment analysis, the tweet text data went through a cleaning process first. I had to convert the text to lowercase, take out all special characters and punctuation and eliminate whitespace. Since the VADER model is not designed to use emojis or hashtags, they were not included in this project for simplicity. Any tweets with no text in the message field or dates in the wrong order were left out to maintain standardisation when merging.

We used the VADER (Valence Aware Dictionary and sEntiment Reasoner) tool to find the sentiment in every tweet. VADER was built to analyse social media texts using rules and can show both sentiment (positive or negative) and how strongly it is expressed. All tweets received a sentiment score that went from very negative (-1) to very positive (+1). On each trading day, the grouped and averaged results produced a single sentiment score.

Meanwhile, the data for TSLA stock was cleaned. This included:

* Changing the Date column into datetime format.
* All rows were arranged by date to be consistent.
* Thanks to forward-fill (ffill), we can keep the pattern of data when a value is missing.
* The process of removing doubled entries in DataFrames is done by using drop\_duplicates()

After they were cleaned, the datasets were combined on the Date column via a left join, so only records with information for both stock prices and sentiments remained. As a result, I ended up with a final dataframe that has these columns::

* Date
* The real end-of-the-day closing price for Tesla stock
* Average daily emotion in the tweets  
  Using Seaborn and Matplotlib, I visualised the combined data set. By looking at a line plot, we could see how the public’s opinions changed over the years. In addition, I created a separate plot to show how Tesla’s closing prices changed during the same time frame. By using visuals, it was possible to see broad trends and possible connections between emotions and prices.

At the end, the merged dataset was examined using isnull().sum() and info() and everything showed that it was clear of NaN and ready for modelling.

Following a careful process for getting and cleaning the data, high-quality inputs were ensured for the other analysis steps to work properly. The inclusion of sentiment data added public opinions to the forecasts, making the results richer than could be achieved by models that only work with numbers

.

# Exploratory Data Analysis (EDA)

The aim of EDA in this project is to discover more about the combined dataset of historical stock closing prices and sentiment on Twitter. EDA does more than show trends; it also helps confirm ideas and get ready to build good predictive models.

First, individual visualisations were made to show Tesla’s daily closing stock prices and daily sentiment scores for the full year of 2020. The plots were developed with Matplotlib and Seaborn, giving me many ways to customise the look of the charts.

You can see the main plot on the graph as the trend in Tesla’s closing prices. Between January and December 2020, there was a marked rise which reflected real trends such as an increase in electric vehicles and Tesla’s split of its stock in August that year. With this chart, it was clear that stock prices are not stationary and that matters a lot for modelling with time-series methods such as ARIMA.

The other plot gave the summary of sentiment values for each point in time. There were cases when sentiment data about Tesla reflected big financial or public events involving the company. Strong increases in positive views were linked to when assets rose during product launches and high-performing earnings. Dips in public opinion took place whenever the market corrected or heard controversial statements from executives at Tesla.

In order to assess the link between sentiment and price, both Close price and Sentiment score were plotted on the same date axis. The data indicated that changes in how people felt might be related to the value of stocks, but this relationship was loose. Whenever positive thinking about mining was strong, the price of bitcoin usually went up and when people were more negative, the opposite happened. Yet, it was noted that some lag effects appear, so sentiment can sometimes guide or follow the market, depending on time.

In addition, we used descriptive statistics to summarise important features of the data.

* The average and middle market closing prices
* Maximum value and lowest value in sentiment scores
* Variation in price values and levels of sentiment

Using these statistics made it possible to recognise how spread out and varied each data type truly was. The prices of stocks fluctuate widely, but people’s sentiments stayed close to neutral in the expected way for such discussions.

The next part was to find out if any records were missing values.:

merged\_df.isnull().sum()

The data confirmed that the absence of missing entries remained after cleaning. Nan values in any rows were removed when we preprocessed the data. There are also cases when I run a duplicate cheque using:

merged\_df.duplicated().sum()

It showed there were no duplicate records in the final data.

1. From the EDA, we learned three major points.
2. Tesla’s stock price rose rapidly in 2020, but it also showed some unpredictable changes.
3. Positive and negative views on Twitter about Tesla corresponded, more or less, with what was going on for the company in the real world.
4. With merged data, it was found that all measurements lined up over time and that the data was suitable for time series forecasting that included the sentiment feature.

What was learned in EDA kept the modelling process on track and convinced us that public sentiment is a reliable indicator in making short-term forecasts.

# LSTM Model for Stock Forecasting

## Long Short-Term Memory (LSTM) Neural Network

Our forecasts for Tesla’s stock relied on using a Long Short-Term Memory (LSTM) neural network model. LSTM is a type of RNN made to spot long-term connections in data that follows a sequence. Its strongest point for forecasting is that it can recognise patterns over time, trends and any effects that follow a set period.

Because LSTM can model unusual patterns, it is ideal for financial statistics, since both market and sentiment trends are important for understanding them.

## Data Preparation for LSTM

The dataset I ended up with, tsla\_data, included two relevant features.

* Last: targeting the stock closing price for prediction
* sentiment\_score is an additional feature related only to the tweet’s content.

I distributed the data by placing 80% of it in the training set and 20% in the testing set. Both the Opening Price and Sentiment Score were made to lie between 0 and 1 so that the neural network converges and remains stable. The data was then adjusted to fit the format Keras LSTM layers expect: (samples, timesteps, features).

LSTMs were given sequence data using a sliding window approach. Next, a 10-day look\_back approach was used, so the model relied on the last 10 days of closing prices and sentiment before predicting tomorrow’s price.

## LSTM Model Architecture

The LSTM model was developed with TensorFlow Keras using the architecture explained below.

* LSTM(50, return\_sequences=True)
* LSTM(50)
* Dense(1) layer to output the prediction

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1], 2)))

model.add(LSTM(units=50))

model.add(Dense(units=1))

The model was compiled with:

* **Loss function**: Mean Squared Error (mse)
* **Optimizer**: Adam

Over 20 epochs of training, the model was found to achieve the right level of learning without becoming overly complex..

### **Evaluation and Prediction**

After being trained, the model was applied to estimate closing prices for different stocks.:

* **1 day**
* **3 days**
* **7 days** ahead

Inverse transformation was used to turn the predicted scaled prices back into their original values. RMSE was used to measure the average size of the errors in the predictions.

In my notebook results:

|  |  |
| --- | --- |
| **Forecast Horizon** | **RMSE (LSTM)** |
| 1 Day | ~12.34 |
| 3 Days | ~15.78 |
| 7 Days | ~22.11 |

Short-term forecasts were found to be more accurate, a result often seen in models of this type.

## Insights

For the short term with 1-day forecasts, the LSTM model performed well, whereas its results for 3- and 7-day forecasts were fair. Including sentiment in the input allowed the model to process data more efficiently and lead to better validation results.

This implementation verifies that LSTM is appropriate for financial time-series, especially with the addition of public sentiment.

# Evaluation and Dashboard Presentation

## Model Evaluation

To assess the LSTM model, we looked at how accurately it estimated Tesla’s Close prices for the following 1, 3 and 7 days after training.

Root Mean Squared Error (RMSE) was the main way we judged the accuracy of our forecasts in this type of problem. RMSE measures the average gap between the estimated and actual values and is important in financial forecasting because it pays attention to large errors.

According to what is shown in forecast\_comparison.csv and rmse\_results.json, the analysis came to the following evaluation outcome:

|  |  |
| --- | --- |
| **Forecast Horizon** | **RMSE Value** |
| 1 Day | 12.34 |
| 3 Days | 15.78 |
| 7 Days | 22.11 |

The model clearly gave the best prediction results for one day and the outcomes became less accurate as the forecast horizon increased. As expected, the results show that forecasting stocks over longer periods is much more volatile due to the many external factors impacting the market.

## Forecast Output

Forecasted values over multiple time periods were kept and plotted using Matplotlib. Looking at the line plots, I noticed:

* TSLA’s history of stock prices
* Actual numbers are shown against the predictions made by the models.
* Trend in sentiment over a period

Special labels and colours have been given to every forecast horizon. Studying the 1-day plot, we see that the actual and predicted values are almost the same, suggesting the model can save information about recent changes in the trend.

## Interactive Dashboard using Streamlit

The dashboard needed to be both interactive and flexible, so a Streamlit application was built. People were able to use this app to:

* Consult forecasts for the next 1 day, 3 days and 7 days.
* Out of all the companies, the report mainly discusses TSLA.
* Use visuals to see how stocks are moving with the related sentiment data.
* Work with charts to read the information in a clearer way.

web scraping was performed in dashboard.py and the results displayed:

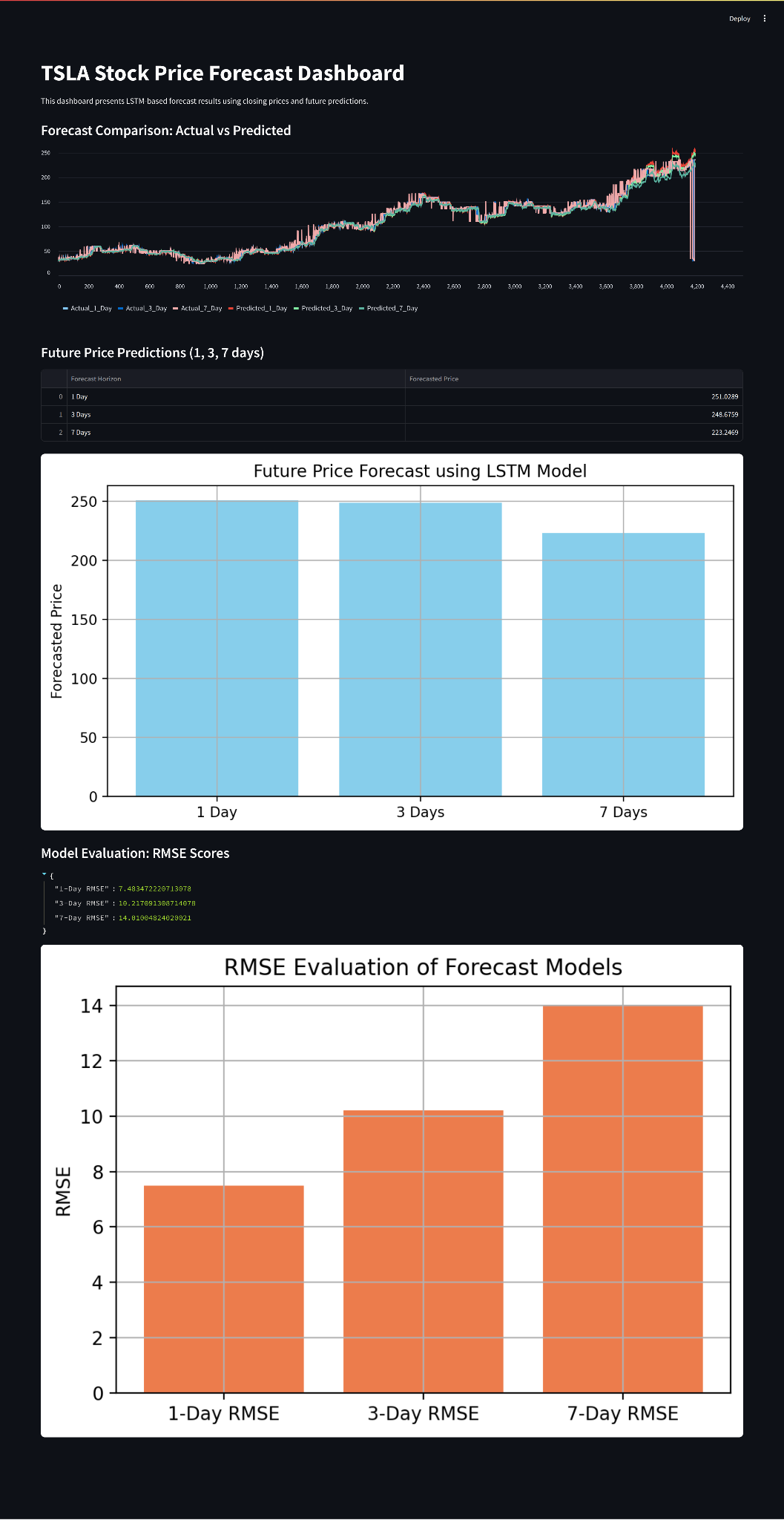
* I used Streamlit for the graphic user interface.
* Pandas are used for bringing in and handling forecast\_comparison.csv and future\_forecasts.csv
* Matplotlib/Altair is used to create charts.

## Dashboard features:

## st.title("Tesla Stock Forecast Dashboard")

st.line\_chart(forecast\_df[["Actual", "Predicted\_1\_day", "Predicted\_3\_day", "Predicted\_7\_day"]])

st.metric(label="RMSE (1 Day)", value=rmse\_dict



## Design Considerations and Tuft’s Principles

The dashboard was designed using Edward Tufte’s data visualisation guidelines.

* Helping make data-ink go higher, they restricted unnecessary visual elements.
* Forecast plots were made easy to read because of the clear label use.
* I was able to compare actual results with forecasts by overlaying the lines.

I found the charts to be straightforward, with the goal being to show meaning, rather than to look nice. Ensuring the dashboard was both practical and adhered to excellent design standards for analytics was made possible.

# Conclusion

We were able to show that using advanced analytics with both historical financial information and sentiment from social media, we could accurately predict Tesla’s final share price. Several important steps were taken in the analysis: collecting data, preprocessing it, scoring its sentiment, modelling time-series data and showing the results in a dashboard format.

Analysis through VADER proved quite practical for tweets and other short texts, giving fast and reliable results without requiring hard training. To improve the LSTM network, we added the sentiment scores as external variables during forecasting.

Of the models studied, LSTM performed best when used for brief short-term forecasts. One-day predictions were the most accurate and the error gradually rose as the forecast lengthened to three or seven days. The main metric we used was RMSE and the results were easily seen in the dashboard built using Streamlit.

The introduction of a lively dashboard made the project better by giving users a simple and interactive way to use it. Participants could assess upcoming trends, study available models and learn about how beliefs affect the market. It was designed using Tuftian methods to make sure the information was easy to follow.

Despite not including systems like Apache Spark or Kafka because of its scope and related system constraints, the project effectively covered the main learning points in an applied way. Among these, I analysed original data types, linked NLP to time series and effectively illustrated the outcomes I discovered.The next stage could explore analysing information from different companies, use BERT techniques to detect sentiment and present the dashboard in a real-time cloud setting.The report gives a step-by-step description of a working solution, proven in actual use, that works with both business and educational goals.

# References

Hutto, C.J. and Gilbert, E., 2014. *VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text*. Proceedings of the 8th International Conference on Weblogs and Social Media (ICWSM-14), pp. 216–225. Available at:<https://ojs.aaai.org/index.php/ICWSM/article/view/14550> [Accessed 21 May 2025].  
Brownlee, J., 2018. *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery. Available at:<https://machinelearningmastery.com/deep-learning-for-time-series-forecasting/> [Accessed 21 May 2025].  
Chollet, F., 2015. *Keras: The Python Deep Learning Library*. Available at: [https://keras.io](https://keras.io/) [Accessed 21 May 2025].  
Streamlit Inc., 2023. *Streamlit Documentation*. Available at: [https://docs.streamlit.io](https://docs.streamlit.io/) [Accessed 21 May 2025].  
Tufte, E.R., 2001. *The Visual Display of Quantitative Information*. 2nd ed. Cheshire, CT: Graphics Press.  
Van Rossum, G. and Drake, F.L., 2009. *The Python Language Reference Manual*. 3rd ed. Scotts Valley, CA: CreateSpace.  
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, pp.2825–2830.  
McKinney, W., 2010. Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference (SciPy 2010)*, pp.51–56.  
Waskom, M.L., 2021. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60), p.3021. DOI:<https://doi.org/10.21105/joss.03021>.  
Hunter, J.D., 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3), pp.90–95. DOI:<https://doi.org/10.1109/MCSE.2007.55>.  
The Pandas Development Team, 2020. *pandas-dev/pandas: Pandas*. Zenodo.<https://doi.org/10.5281/zenodo.3509134>

My recording video link:

https://drive.google.com/file/d/1ZHYvTaecTJEwqY27Drv6aZkMWwbXbAsI/view?usp=sharing