

# Airline Delay Analysis Using Deep Learning

Alina Gu, Jackie Kim, Jessica Wen, Omar Shatrat



# TABLE OF CONTENTS

- Abstract
- EDA
- Regression Results
- Classification Results
- Model Operations
- Conclusion
- Appendix



# Abstract/Project Objective

For our final project, our aim is to predict flight cancellation and delay times

- 1) Can we predict which flights will be canceled?**
- 2) For uncanceled flights, can we predict their delay departure time?**

Our approach incorporates popular regression and classification deep learning architectures.

The project could offer insights into patterns, allow airlines to plan accordingly ahead of time, potentially increase efficiency and customer satisfaction.





# Data Overview

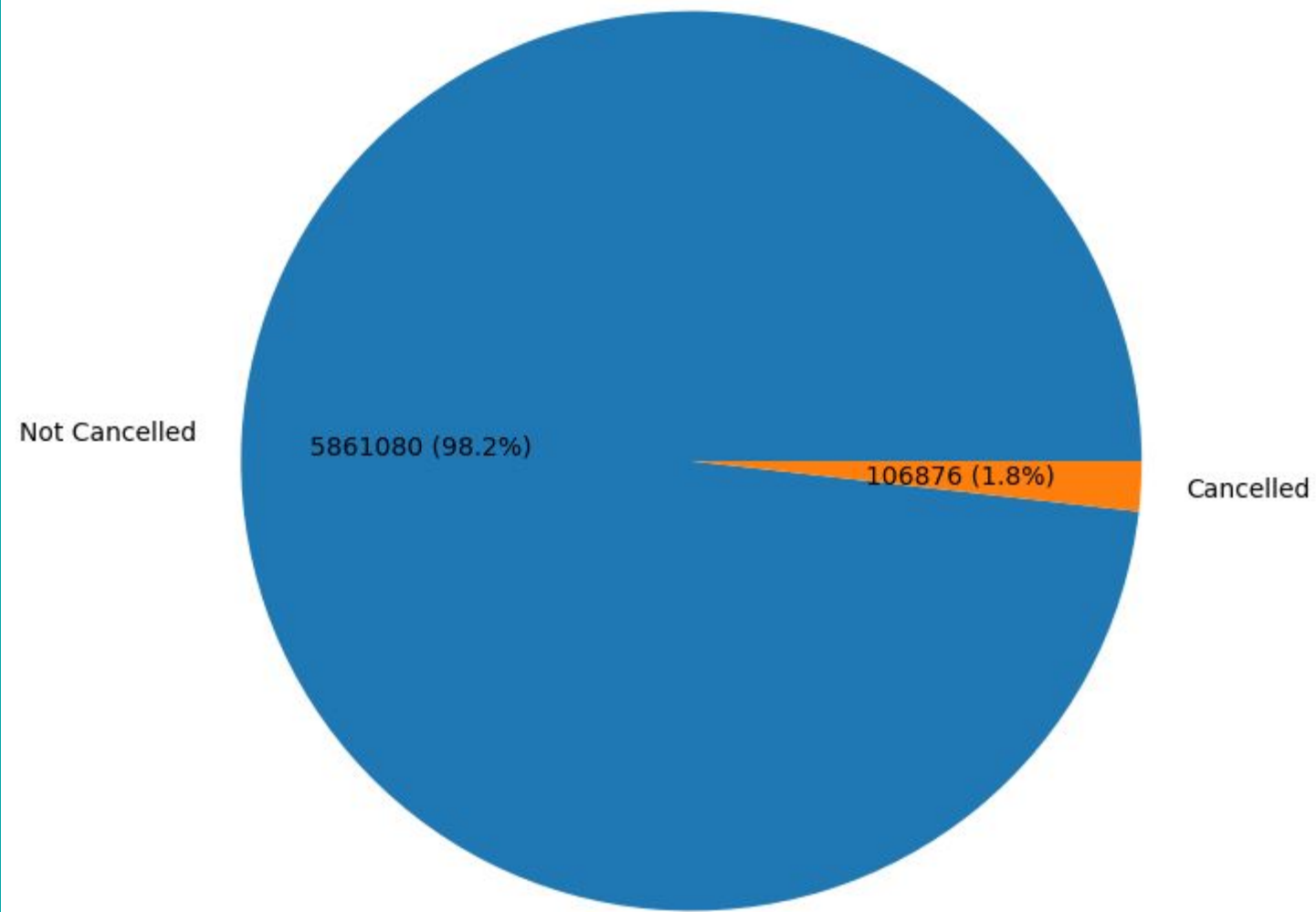
- About 6 million rows
- 24 columns
- No null value
- Highly non-linear
- 10 categorical variables: cancelled, airline, origin, destination
- 14 numerical variables: distance, delay minutes, weather (wind, temperature, precipitation, snow)



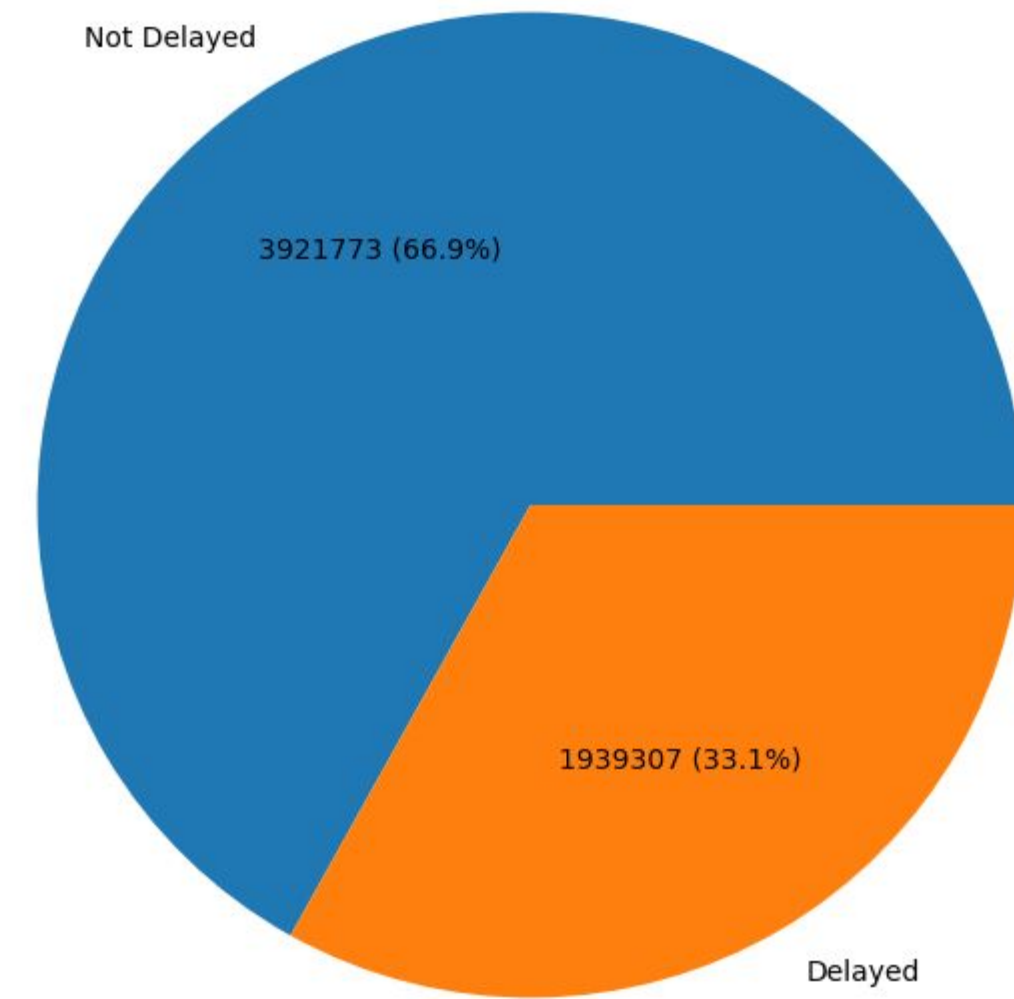
# Visualization



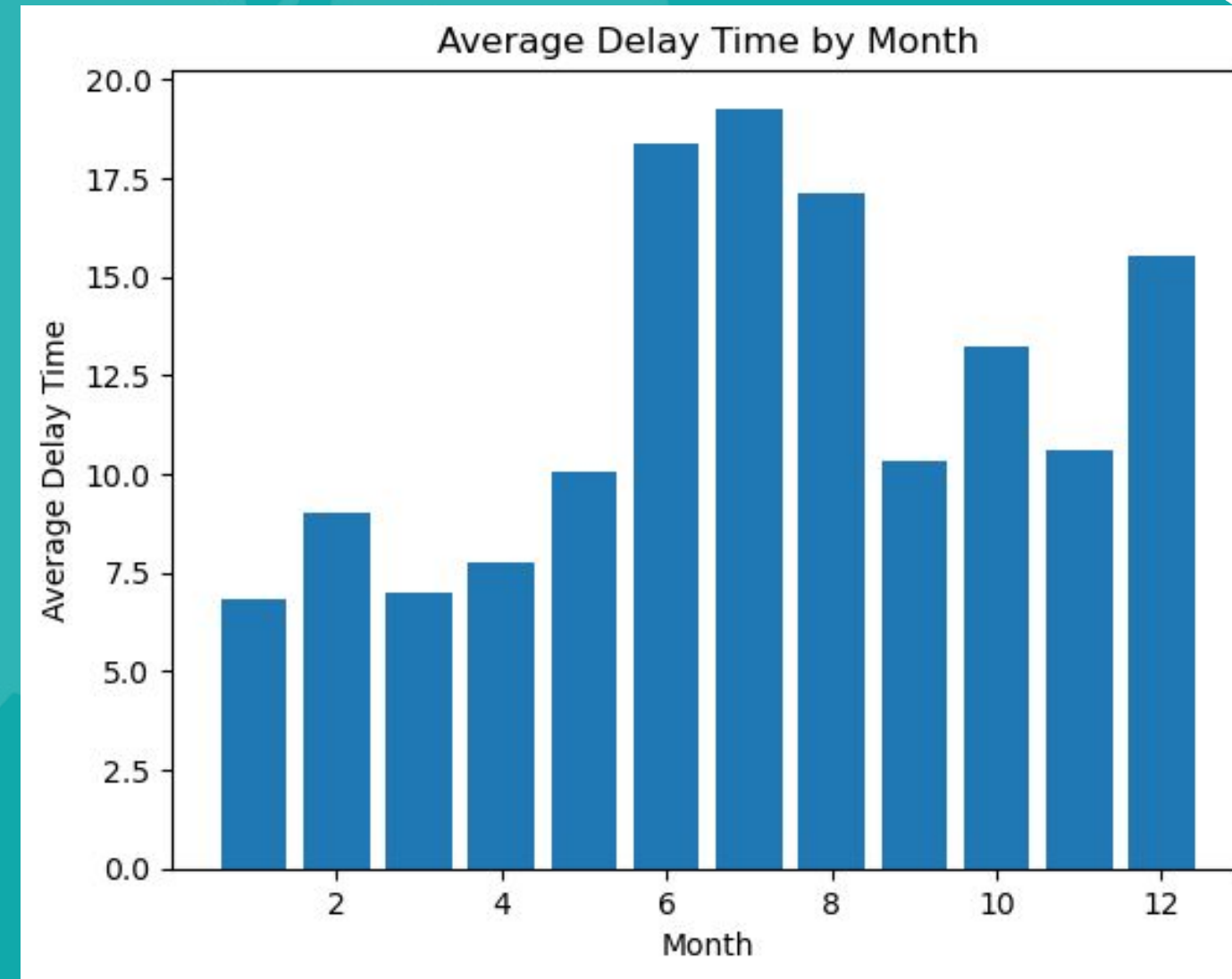
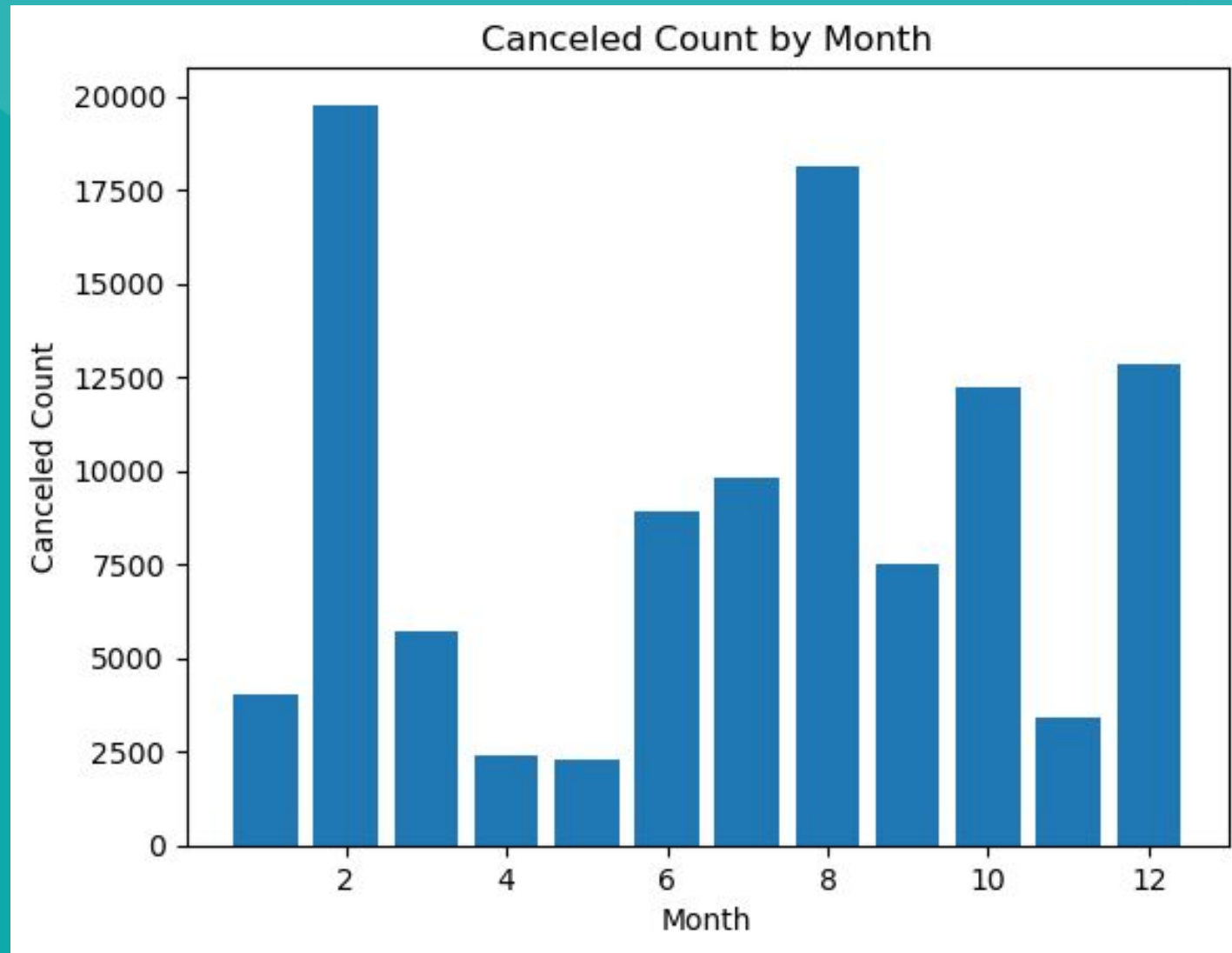
Proportion of Cancelled and Non-cancelled Flights



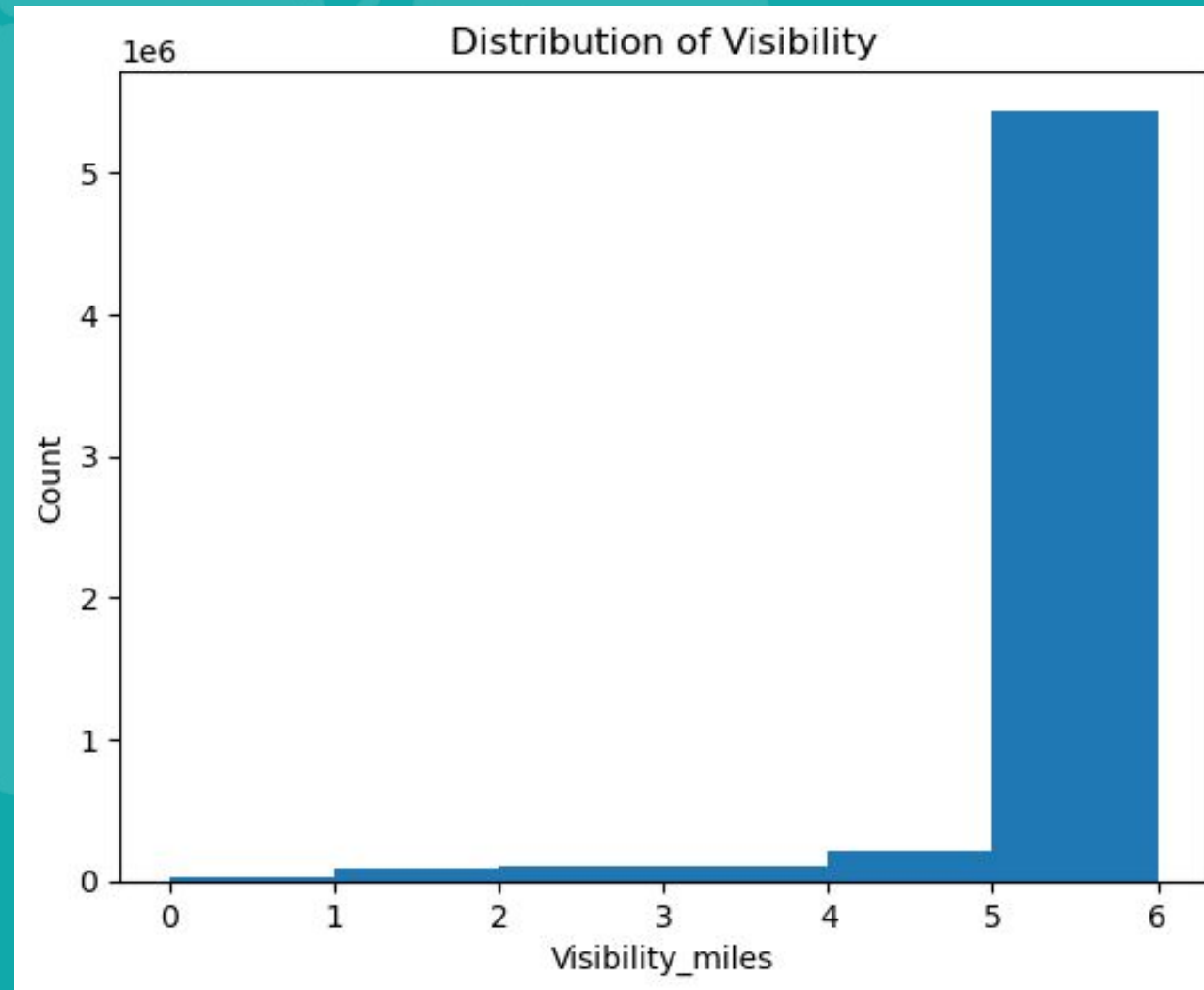
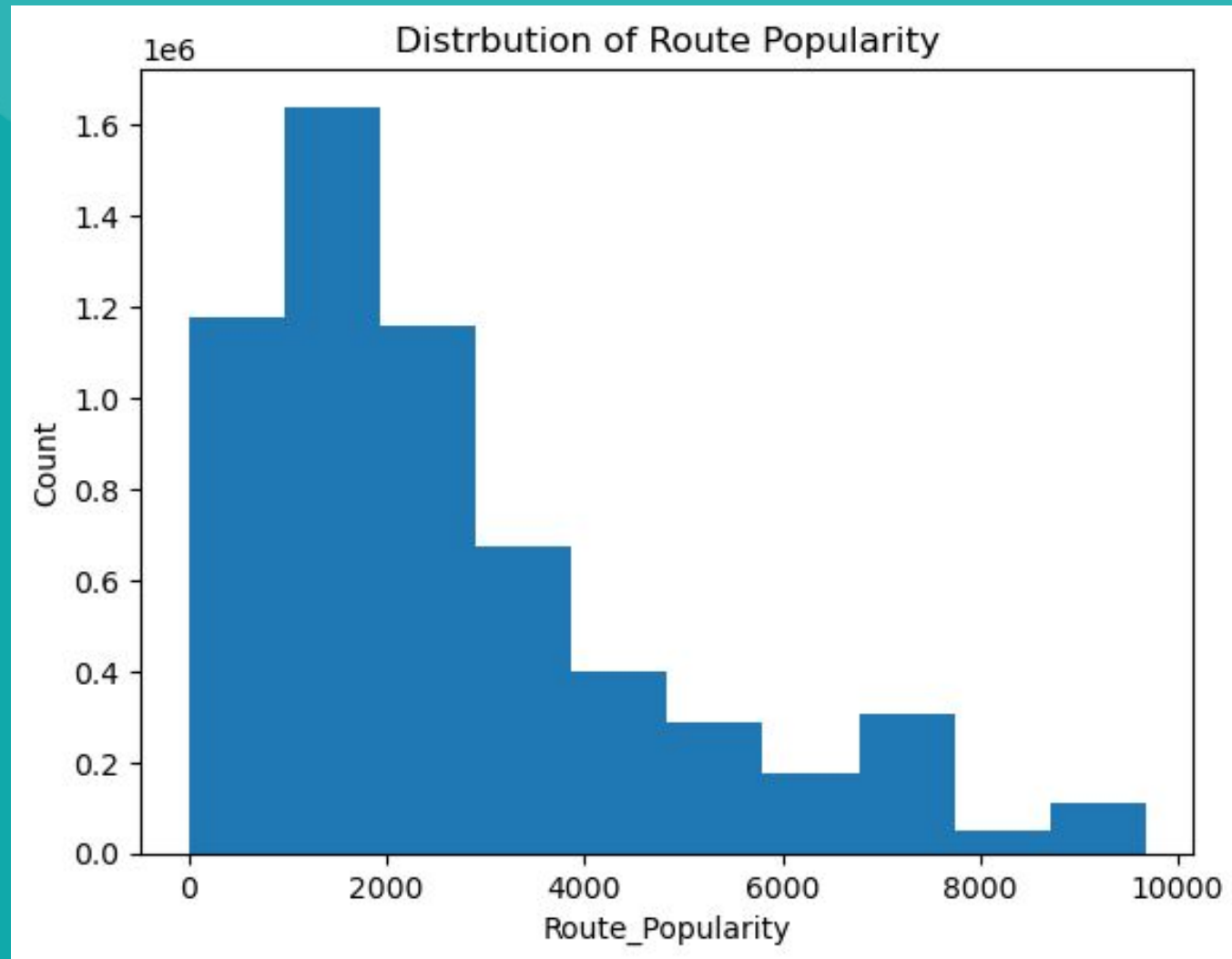
Proportion of Delayed and Non-delayed Flights



# Visualization

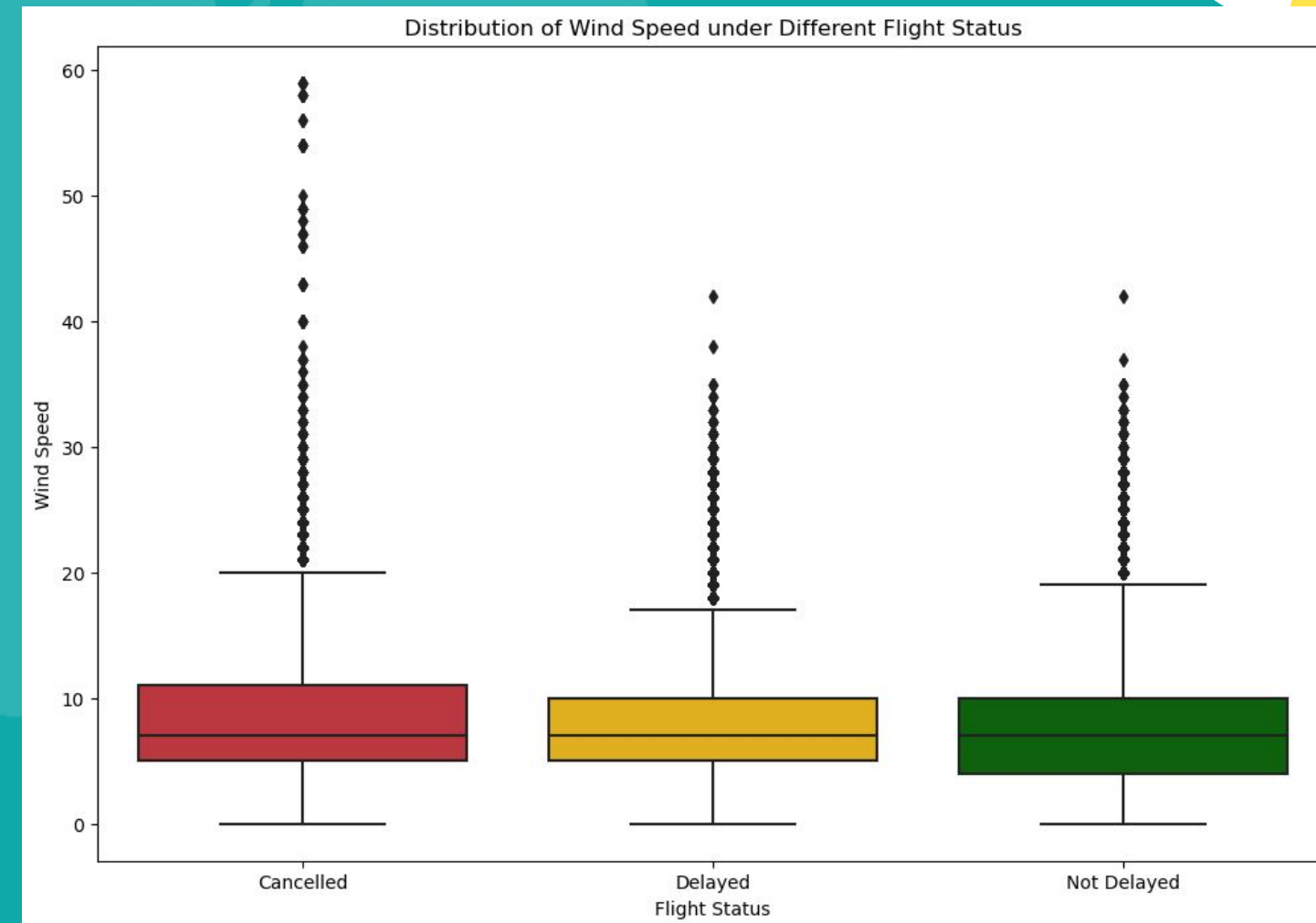
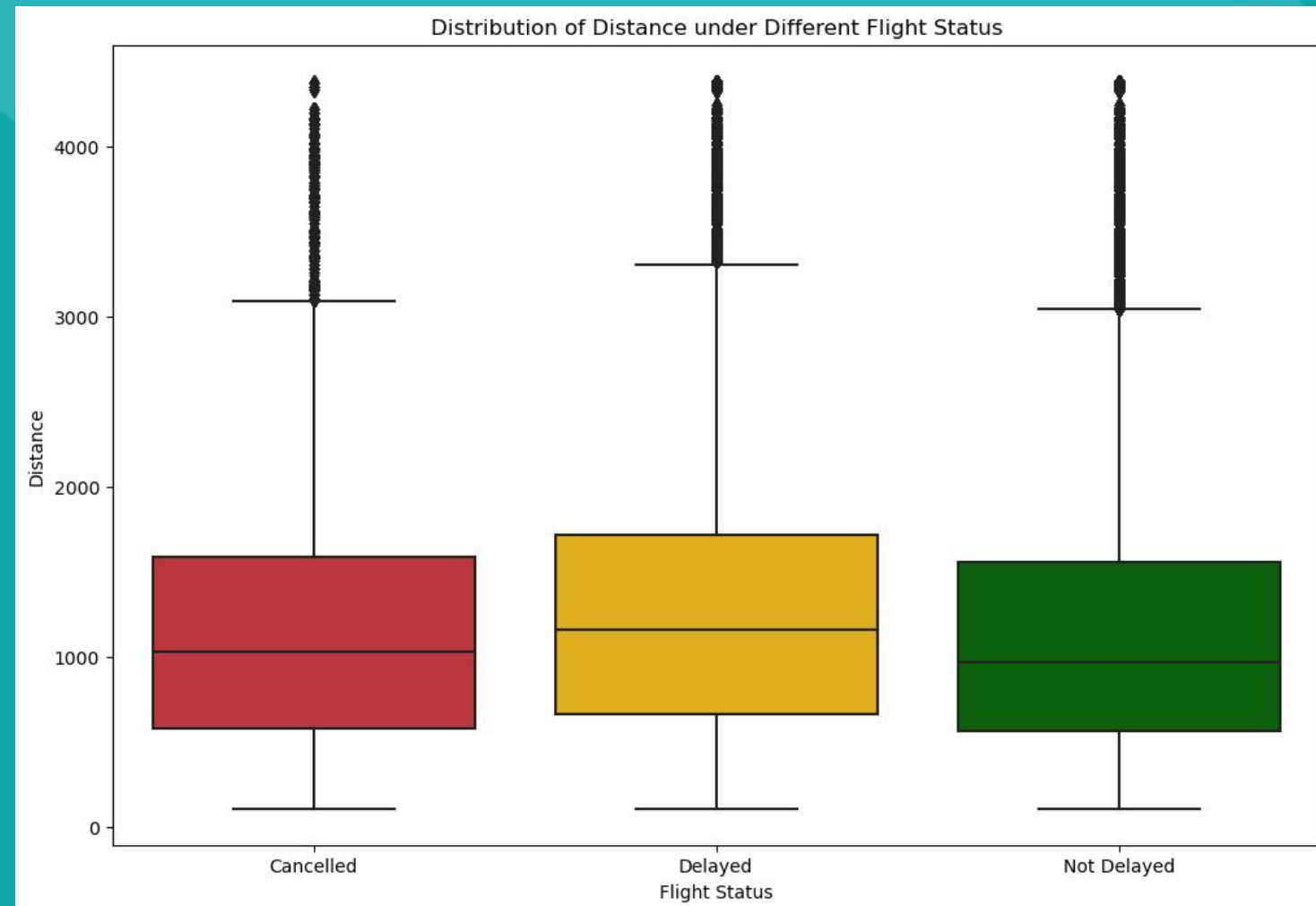


# Visualization



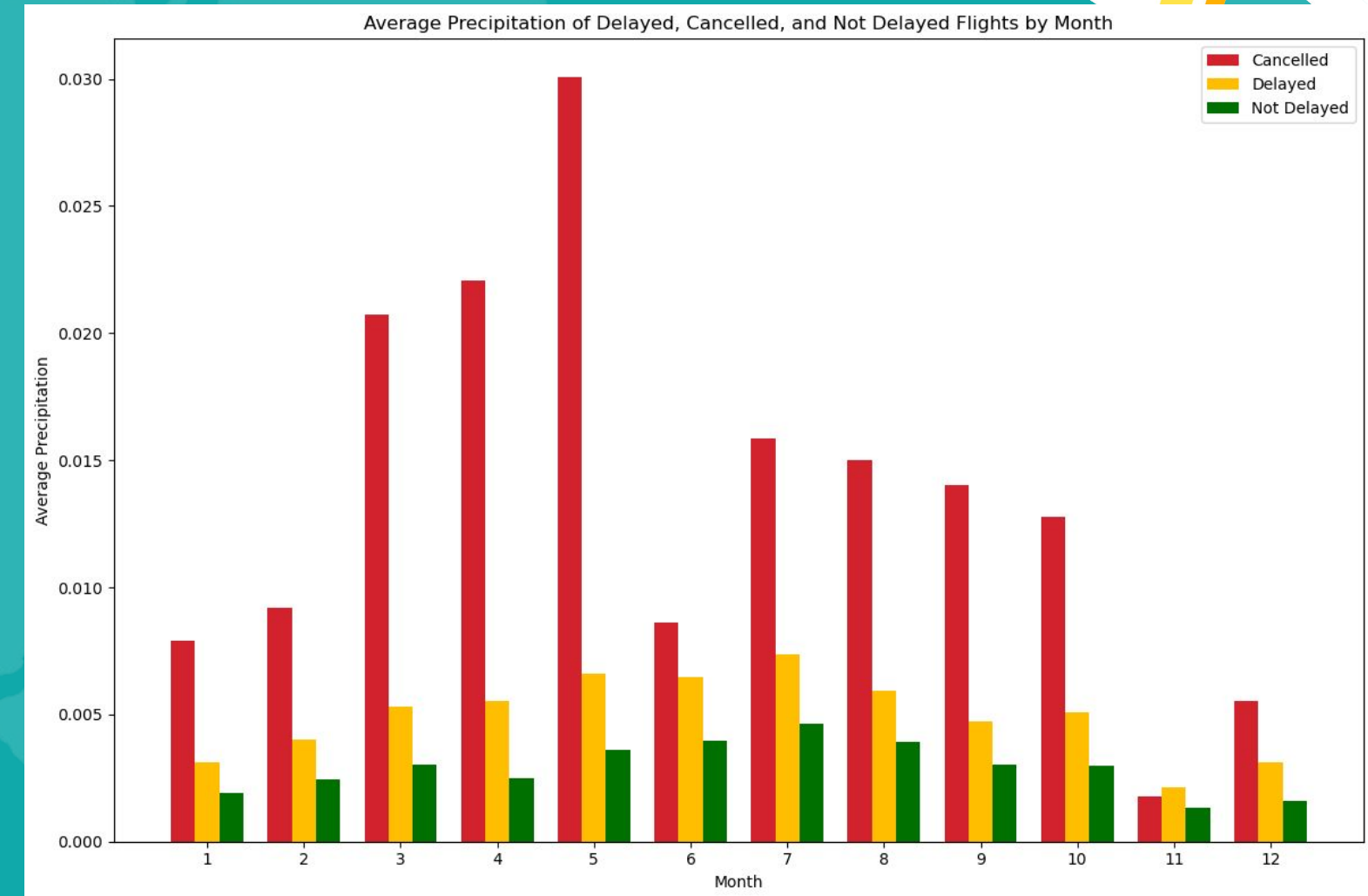
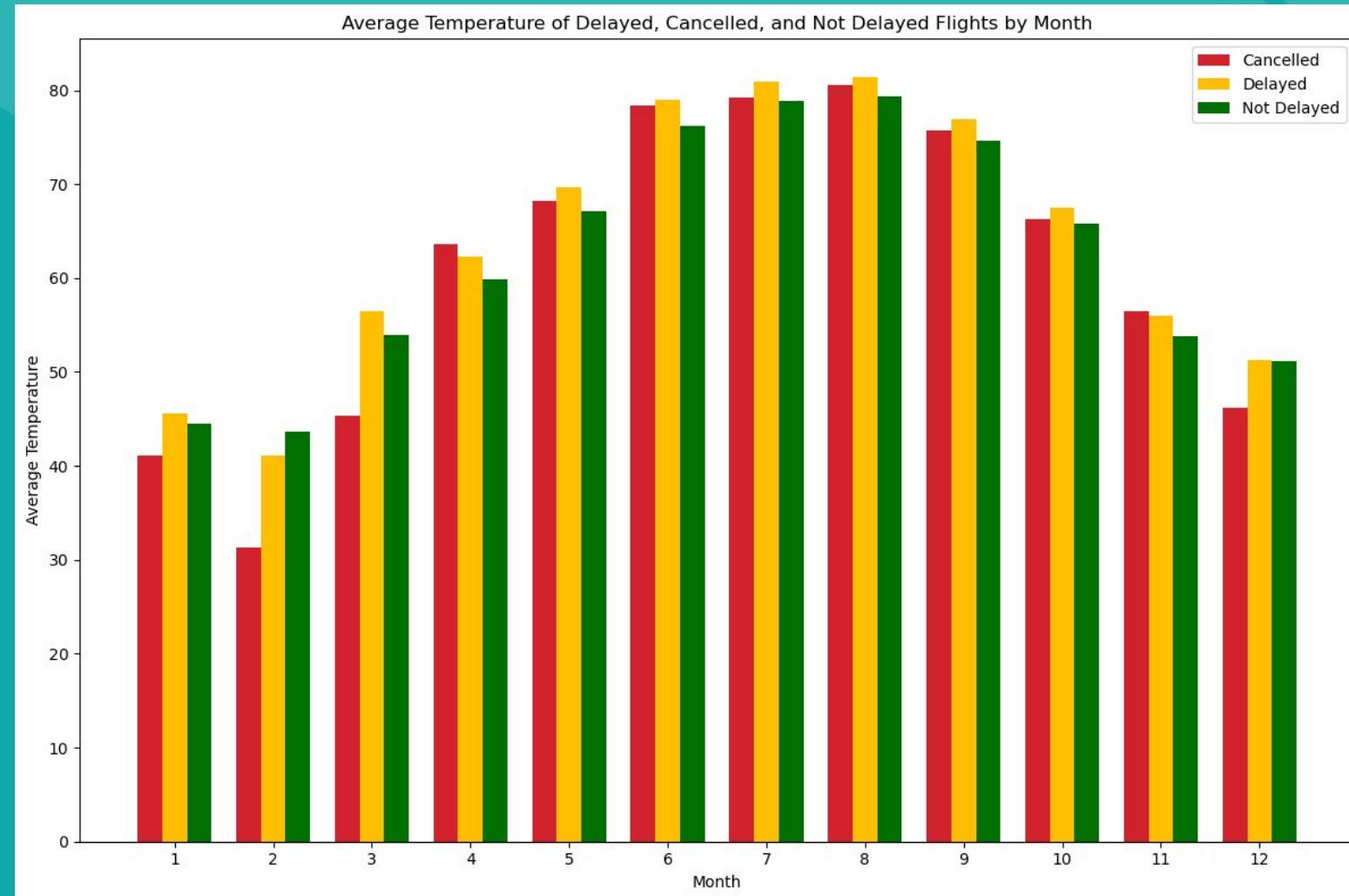
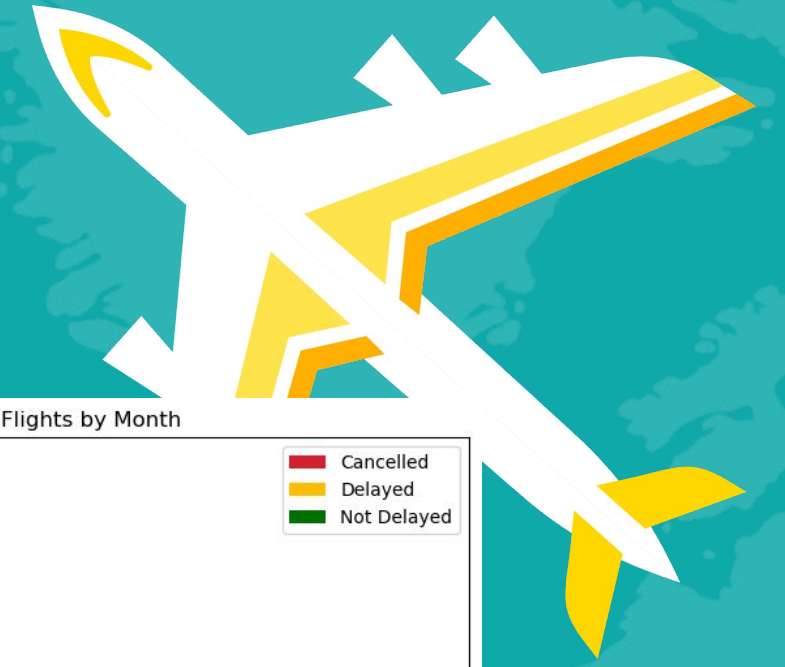


# Visualization





# Visualization





# Modeling Part I: Regression: Prediction for delay minutes

# DL Models for Regression



**LSTM**



**Transformer**



**CNN + Attention + LSTM**



# Data Preprocessing

## Objective:

- Predict the delay minutes for flights using historical flight data.

## Data Characteristics:

- **Categorical Features:**
  - Airline, Departure Time Block, Department Type.
- **Numerical Features:**
  - Quarter, Month, Day of the Week, Department Elevation, Route Popularity, Distance, Weather Conditions (Wind, Visibility, Temperature, Precipitation, Snow).

## Preprocessing Steps:

- One-hot encoding for categorical variables.
- Normalization of numerical features using MinMaxScaler.
- Log transformation of the target variable 'DepDelayMinutes' to handle skewness and outliers.



# LSTM Model for Flight Delay Prediction



## Model Overview:

- Employs LSTM to effectively process sequential data with long-term dependencies.

## Pros:

- Handles long data sequences well
- Avoids the vanishing gradient problem.

## Cons:

- **High Computational Costs:** making it slower and more expensive to train.
- **Complex Tuning Required:** Demands careful tuning and expertise to optimize and prevent overfitting.



# LSTM Model Architecture

[Input Layer]



[LSTM Layer]



→ [Hidden State]

→ [Cell State]



[Output of Last Time Step]



[Linear Layer]



[Output] → Predicted Value



# Transformer Model for Flight Delay Prediction



## Model overview:

- Uses Transformer Encoder with self-attention.

## Pros:

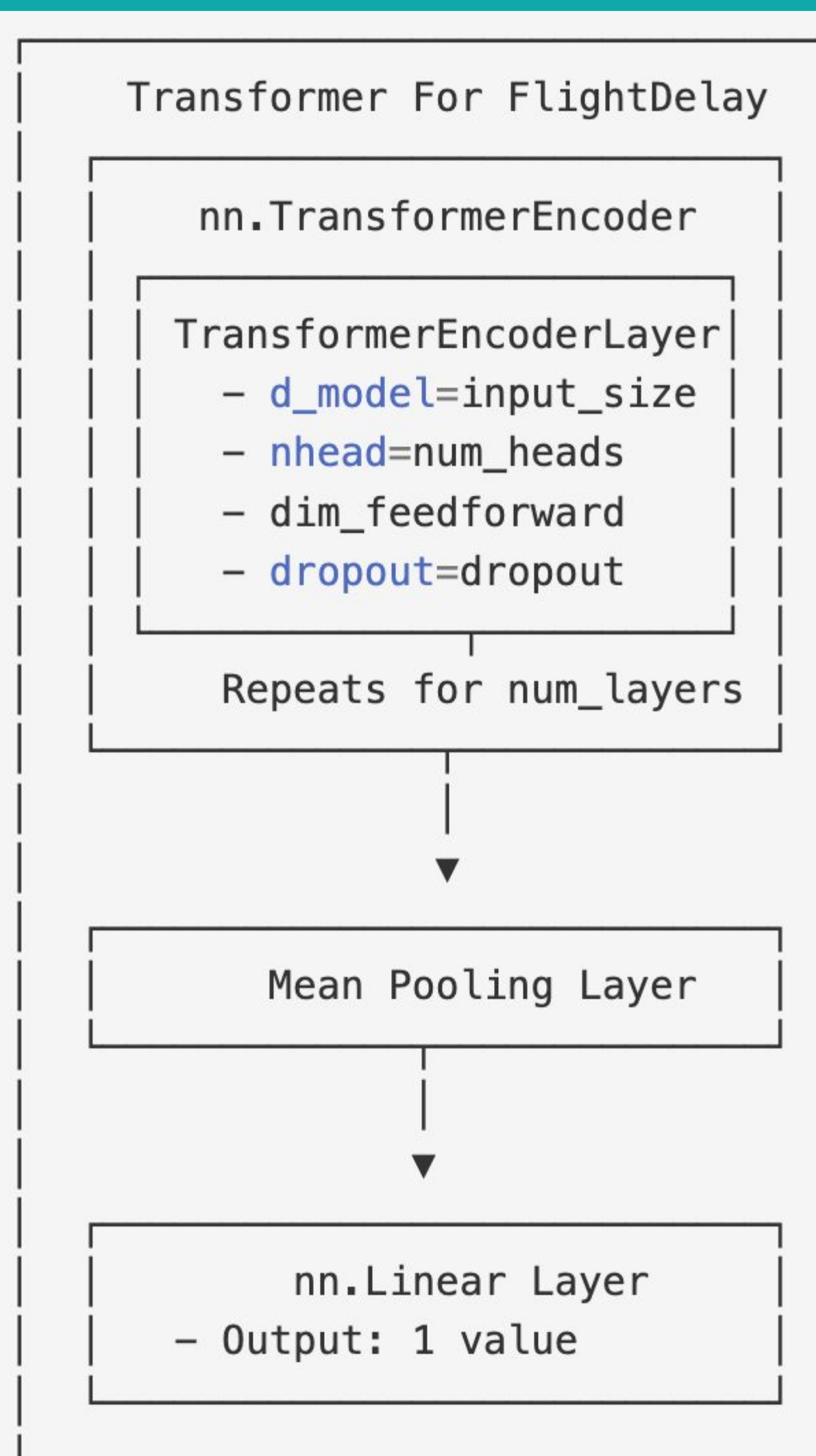
- Efficient parallel processing.
- Effective with long sequences due to self-attention.

## Cons:

- High computational demands.
- Complexity can lead to tuning challenges and overfitting.



# Transformer Model Architecture



# Reference to Paper: CNN + Attention + LSTM



## Citation of Model:

- Referenced Model: SimAM-CNN-MLSTM from "Flight Delay Propagation Prediction Based on Deep Learning".
- Source: Mathematics 2023, 11(3), 494.

## Why We Refer to This Model:

- Proven Accuracy: Achieved 91.36% accuracy, demonstrating effectiveness in handling complex data.
- Technique Integration: Combines CNNs, MLSTMs, and SimAM for enhanced spatial-temporal analysis.

## Model Influence:

- Informs our approach to optimizing predictive accuracy and adapting advanced deep learning techniques.



# CNN + attention + LSTM Flight Delay Prediction



## Model Overview:

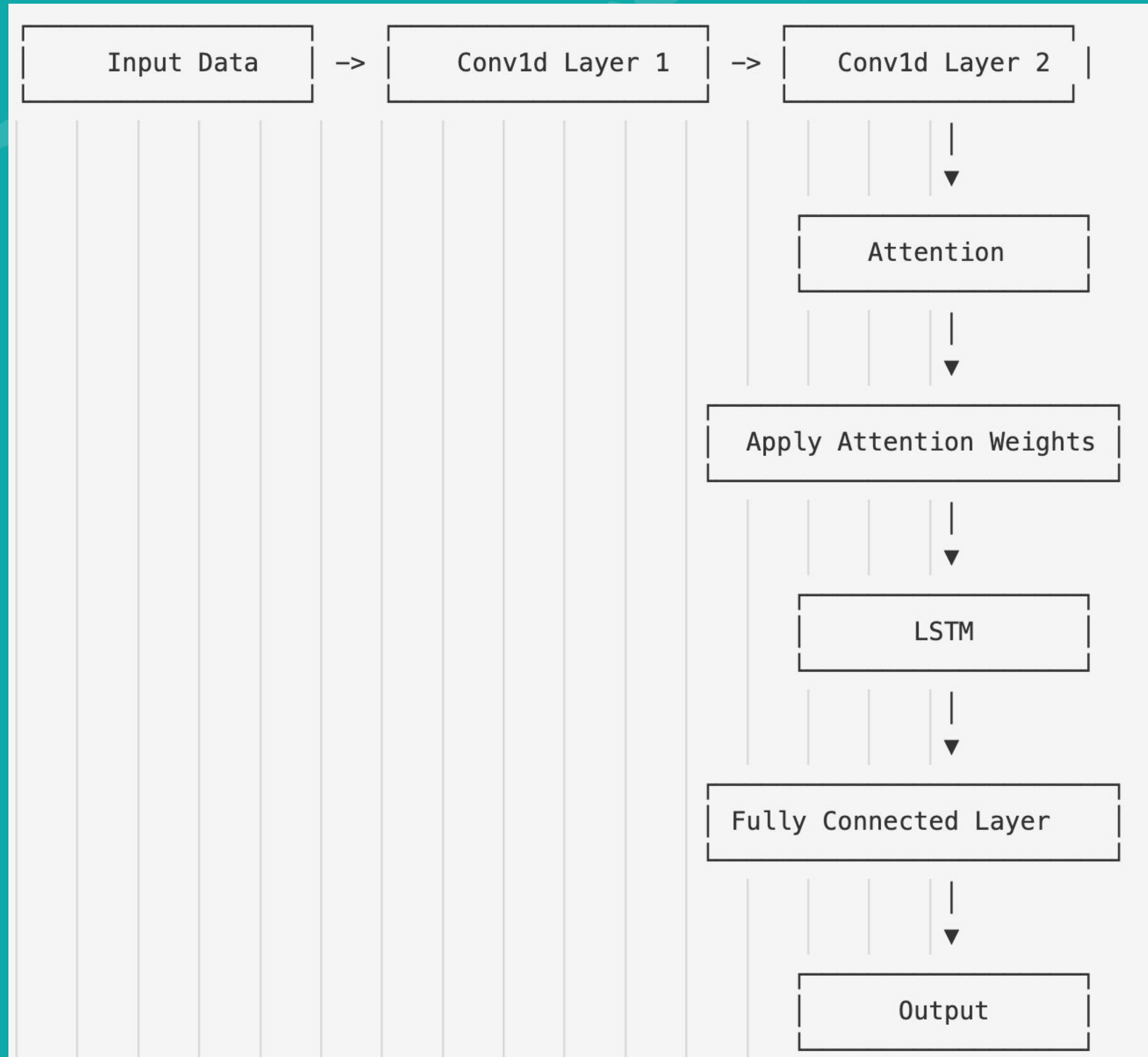
- Combines CNNs, attention, and LSTM to effectively process sequential data, capturing both spatial and temporal dynamics.

## Pros:

- **Enhanced Feature Analysis:** Leverages CNN for spatial details and attention for prioritizing key features, boosting analytical accuracy.
- **Temporal Precision:** LSTM effectively manages long-term dependencies, crucial for sequence prediction.

## Cons:

- **Resource Intensive:** Requires high computational power due to the complex architecture.
- **Tuning Complexity:** The integration of multiple advanced components leads to challenges in model tuning and potential overfitting.



# CNN + attention + LSTM Model Architecture

# Model Comparison



Model	RMSE
Transformer	1.4185
CNN + Attention + LSTM	1.4121
LSTM	1.4176

- **Minimal RMSE Differences:** All models show highly competitive performance with nearly identical RMSE values.
- **Choice Flexibility:** Model selection can focus on computational efficiency and ease of use due to similar accuracies.



# Modeling Part II: Classification: Prediction for cancellation



# DL Models for Classification



Simple RNN



Hybrid LSTM/GRU



Stacked LSTM



SDA-LSTM

# Data Preprocessing

## Objective:

- Predict flight cancellations using historical flight data.

## Data Characteristics:

- **Categorical Features:**
  - Airline, Departure Time Block, Department Type.
- **Numerical Features:**
  - Quarter, Month, Day of the Week, Department Elevation, Route Popularity, Distance, Weather Conditions (Wind, Visibility, Temperature, Precipitation, Snow).

## Preprocessing Steps:

- Applied Downsampling due to imbalance cancellation dataset
- One-hot encoding for categorical variables.
- Standardization with the first three models.
- Normalization of numerical features for SDA-LSTM





# Simple RNN Model for Flight Cancellation



## Model Overview:

- Employs RNN to effectively process variable-length sequential data.

## Pros:

- Processes data sequentially
- Avoids the vanishing gradient problem.

## Cons:

- High Computational Costs: making it slower and more expensive to train.
- Complex Tuning Required: Demands careful tuning and expertise to optimize and prevent overfitting.



# SimpleRNN Model Architecture

```
+-----+  
|           Input Layer           |  
+-----+
```

```
|  
v
```

```
+-----+  
| SimpleRNN (50) - Return Sequences: True |  
+-----+
```

```
|  
v
```

```
+-----+  
| Dropout (0.2) |  
+-----+
```

```
|  
v
```

```
+-----+  
| SimpleRNN (50) - Return Sequences: False |  
+-----+
```

```
|  
v
```

```
+-----+  
| Dropout (0.2) |  
+-----+
```

```
|  
v
```

```
+-----+  
| Dense (1) |  
| Activation: Sigmoid |  
+-----+
```

# Hybrid LSTM/GRU



## Model Overview:

- Employs ensemble of architectures to effectively process variable-length sequential data with long-term dependencies.

## Pros:

- Processes data sequentially
- Combines two popular techniques together
- Avoids the vanishing gradient problem.

## Cons:

- High Computational Costs: having to train 2 models instead of 1.
- Complex Tuning Required: Demands careful tuning and expertise to optimize and prevent overfitting.



```

      Input Layer
      /           \
    LSTM (100) -----> [Return Sequences: True]
      /           \
    Dropout (0.3)   Dropout (0.3)
      /           \
  BatchNormalization BatchNormalization
      /           \
      |             GRU (100) -----> [Return Sequences: True]
      |             /           \
      |           Dropout (0.3)   Dropout (0.3)
      |             /           \
      |         BatchNormalization BatchNormalization
      |             /           \
    LSTM (50) -----> [Return Sequences: False]
      /           \
    Dropout (0.3)   Dropout (0.3)
      /           \
  BatchNormalization BatchNormalization
      /           \
    Dense (50) -----> [Activation: ReLU]
      /           \
    Dropout (0.3)   Dropout (0.3)
      /           \
    Dense (1) -----> [Activation: Sigmoid]

```



# LSTM/GRU Architecture

# Stacked LSTM



## Model Overview:

- Employs LSTM to effectively process sequential data with long-term dependencies.

## Pros:

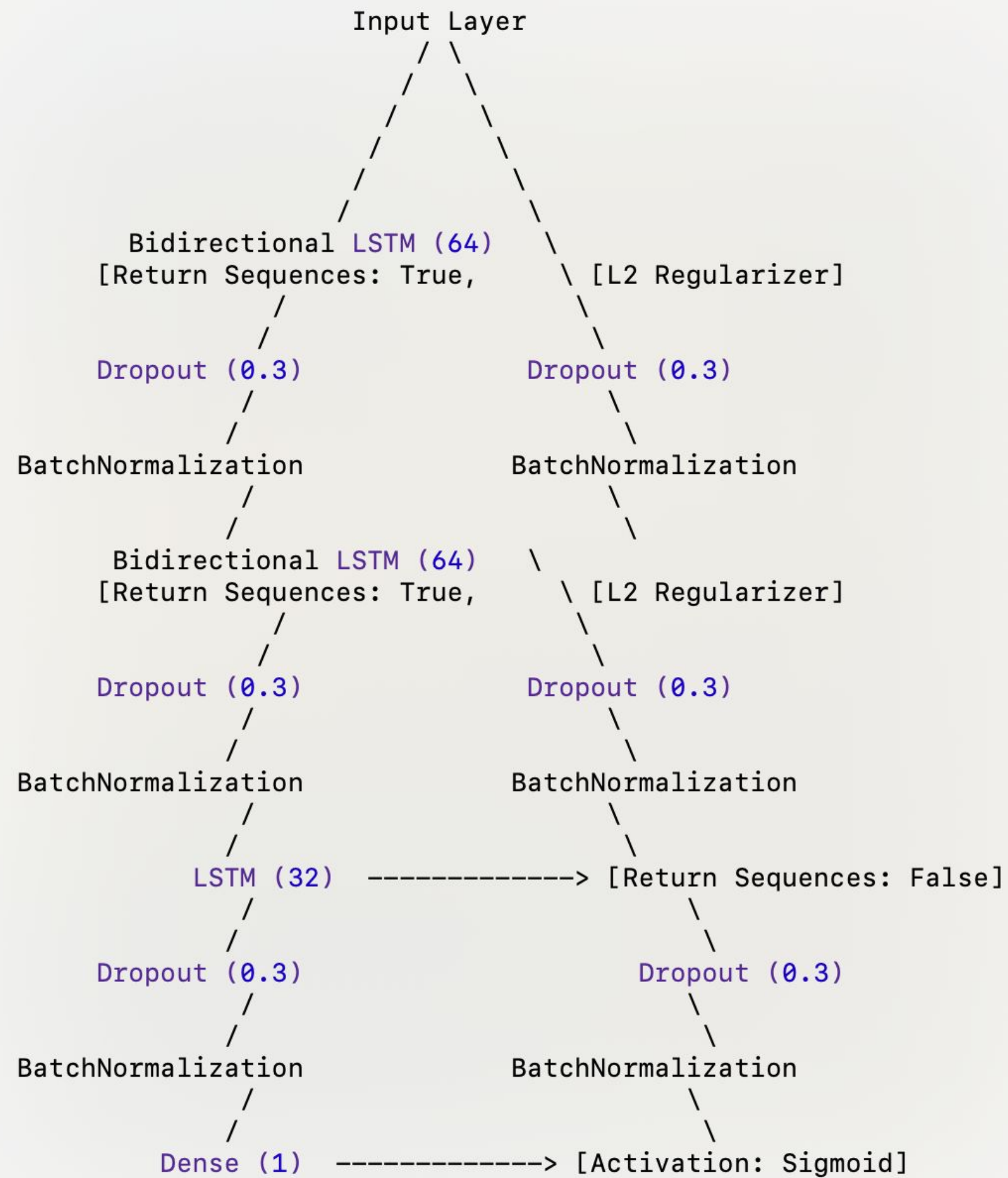
- Handles long data sequences well
- Avoids the vanishing gradient problem.

## Cons:

- **High Computational Costs:** making it slower and more expensive to train.
- **Complex Tuning Required:** Demands careful tuning and expertise to optimize and prevent overfitting.



# Stacked LSTM Architecture





# SDA-LSTM



## Model Overview:

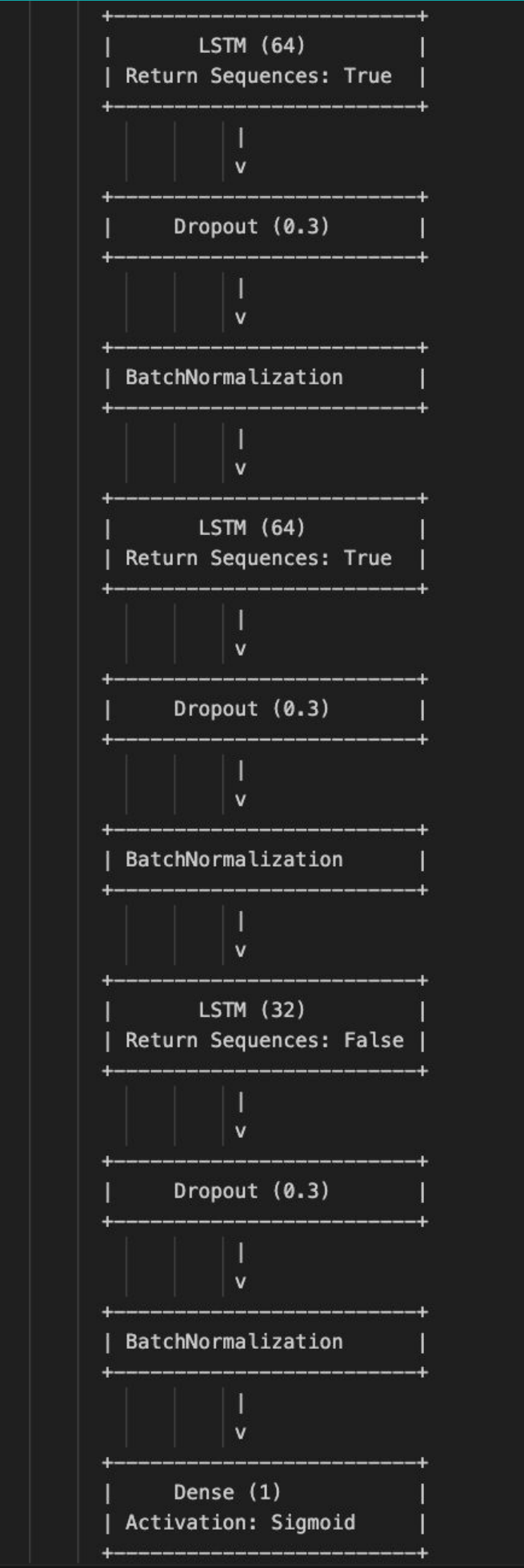
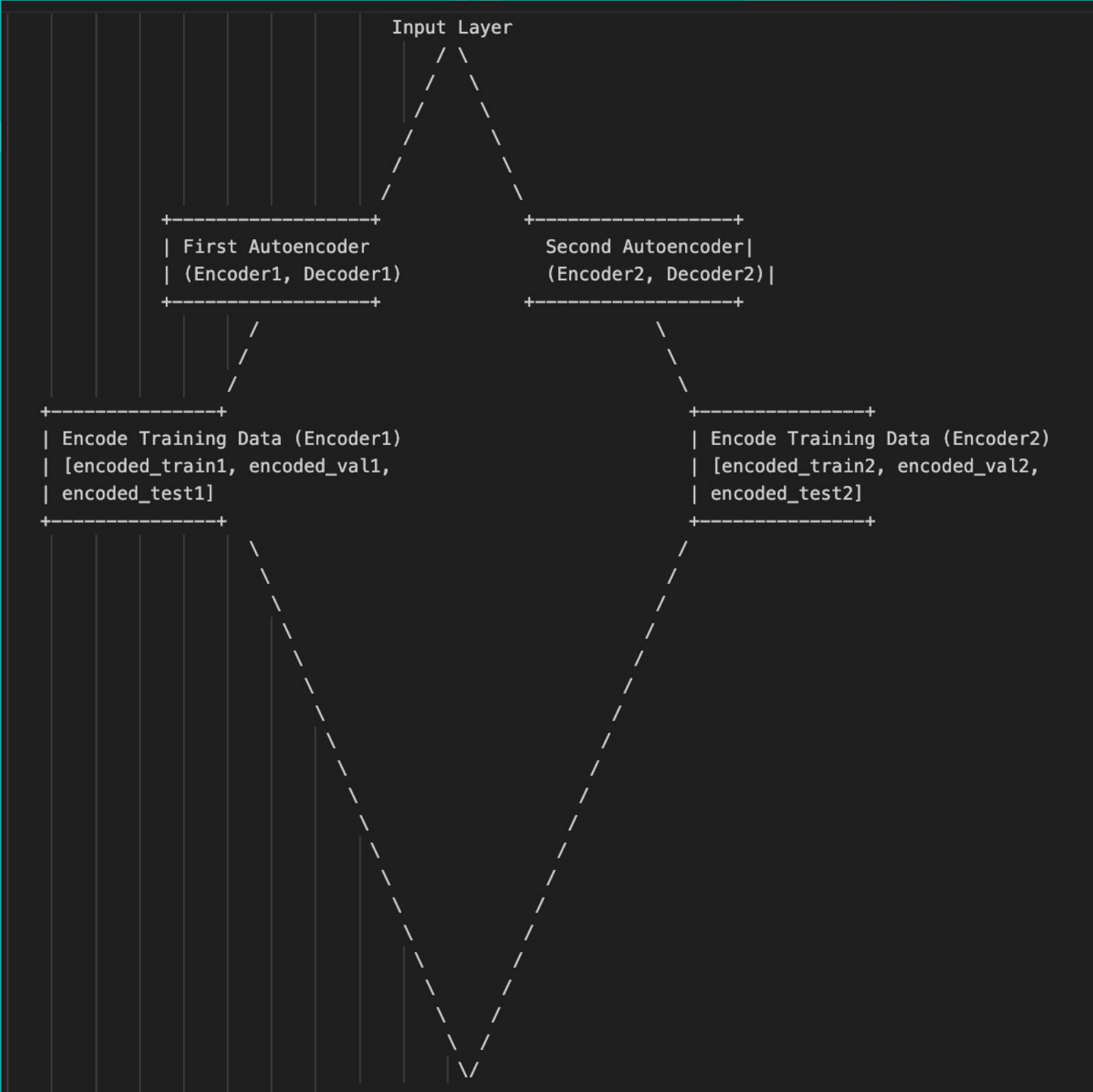
- Adjustment of: SDA-LM (Cannot implement due to data overflow issue)
- Employs 2 autoencoders to reduce dimensionality and noise of data before feeding into LSTM

## Pros:

- Handles long data sequences well
- Utilizes modern encoder methods
- Ideal of flight problem

## Cons:

- High Computational Costs: making it slower and more expensive to train.
- Complex Tuning Required: Demands careful tuning and expertise to optimize and prevent overfitting.



# SDA-LSTM Architecture

# Model Comparison



Model	Accuracy	Precision	Recall	F1
Simple RNN	.75	.73	.78	.76
LSTM/GRU Ensemble	.76	.77	.76	.76
Stacked LSTM	.69	.76	.73	.71
SDA-LSTM	1.0	1.0	1.0	1.0



- **Deployment:**
  - Containerize the best model using Docker.
  - Deploy the container on a Kubernetes cluster / ECS.
  - Expose the model via a RESTful API.
- **Monitoring:**
  - Set up a solution for real-time monitoring.
  - Implement logging.
- **Maintenance:**
  - Schedule monthly retraining using new data collected via an automated pipeline.
  - Implement a CI/CD pipeline to automate testing and deployment of the trained model.
  - Perform regular hyperparameter tuning to account for data drift.
- **Feedback and Improvement:**
  - Collect and analyze user feedback.
  - Continue to study model errors.
  - Update the model and redeploy as needed based on findings.



# Conclusion

## Best Regression Model:

- CNN + Attention + LSTM

## Best Classifier:


- SDA-LSTM: Offers state of the art performance for predicting flight cancellations. Blew all other models out of the water.

See appendix for more information

# Appendix



# Flight delay prediction based on deep learning and Levenberg-Marquart algorithm

Maryam Farshchian Yazdi, Seyed Reza Kamel , Seyyed Javad Mahdavi Chabok & Maryam Kheirabadi

*Journal of Big Data* **7**, Article number: 106 (2020) | [Cite this article](#)

**40k** Accesses | **35** Citations | [Metrics](#)

## Abstract

Flight delay is inevitable and it plays an important role in both profits and loss of the airlines. An accurate estimation of flight delay is critical for airlines because the results can be applied to increase customer satisfaction and incomes of airline agencies. There have been many researches on modeling and predicting flight delays, where most of them have been trying to predict the delay through extracting important characteristics and most related features. However, most of the proposed methods are not accurate enough because of massive volume data, dependencies and extreme number of parameters. This paper proposes a model for predicting flight delay based on Deep Learning (DL). DL is one of the newest methods employed in solving problems with high level of complexity and massive amount of data. Moreover, DL is capable to automatically extract the important features from data. Furthermore, due to the fact that most of flight delay data are noisy, a technique based on stack denoising autoencoder is designed and added to the proposed model. Also, Levenberg-Marquart algorithm is applied to find weight and bias proper values, and finally the output has been optimized to produce high accurate results. In order to study effect of stack denoising autoencoder and LM algorithm on the model structure, two other structures are also designed. First structure is based on autoencoder and LM algorithm (SAE-LM), and the second structure is based on denoising autoencoder only (SDA). To investigate the three models, we apply the proposed model on U.S flight dataset that it is imbalanced dataset. In order to create balance dataset, undersampling method are used. We measured precision, accuracy, sensitivity, recall and F-measure of the three models on two cases. Accuracy of the proposed prediction model analyzed and compared to previous prediction method. results of three models on both imbalanced and balanced datasets shows that precision, accuracy, sensitivity, recall and F-measure of SDA-LM model with imbalanced and balanced dataset is improvement than SAE-LM and SDA models. The results also show that accuracy of the proposed model in forecasting flight delay on imbalanced and balanced dataset respectively has greater than previous model called RNN.

## Reference:

<https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00380-z>



# Levenberg–Marquardt algorithm

 14 languages 

Article Talk

Read Edit View history Tools 

From Wikipedia, the free encyclopedia

In **mathematics** and computing, the **Levenberg–Marquardt algorithm** (**LMA** or just **LM**), also known as the **damped least-squares** (**DLS**) method, is used to solve **non-linear least squares** problems. These minimization problems arise especially in **least squares curve fitting**. The LMA interpolates between the **Gauss–Newton algorithm** (GNA) and the method of **gradient descent**. The LMA is more **robust** than the GNA, which means that in many cases it finds a solution even if it starts very far off the final minimum. For well-behaved functions and reasonable starting parameters, the LMA tends to be slower than the GNA. LMA can also be viewed as **Gauss–Newton** using a **trust region** approach.

The algorithm was first published in 1944 by **Kenneth Levenberg**,<sup>[1]</sup> while working at the **Frankford Army Arsenal**. It was rediscovered in 1963 by **Donald Marquardt**,<sup>[2]</sup> who worked as a **statistician** at **DuPont**, and independently by Girard,<sup>[3]</sup> Wynne<sup>[4]</sup> and Morrison.<sup>[5]</sup>

The LMA is used in many software applications for solving generic curve-fitting problems. By using the Gauss–Newton algorithm it often converges faster than first-order methods.<sup>[6]</sup> However, like other iterative optimization algorithms, the LMA finds only a **local minimum**, which is not necessarily the **global minimum**.

## The problem [ edit ]

The primary application of the Levenberg–Marquardt algorithm is in the least-squares curve fitting problem: given a set of *m* empirical pairs *(x<sub>i</sub>, y<sub>i</sub>)* of independent and dependent variables, find the parameters *β* of the model curve *f* (*x*, *β*) so that the sum of the squares of the deviations *S* (*β*) is minimized:

$$\hat{\boldsymbol{\beta}} \in \operatorname{argmin}_{\boldsymbol{\beta}} S\left(\boldsymbol{\beta}\right) \equiv \operatorname{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^m \left[y_i - f\left(x_i, \boldsymbol{\beta}\right)\right]^2, \text{ which is assumed to be non-empty.}$$

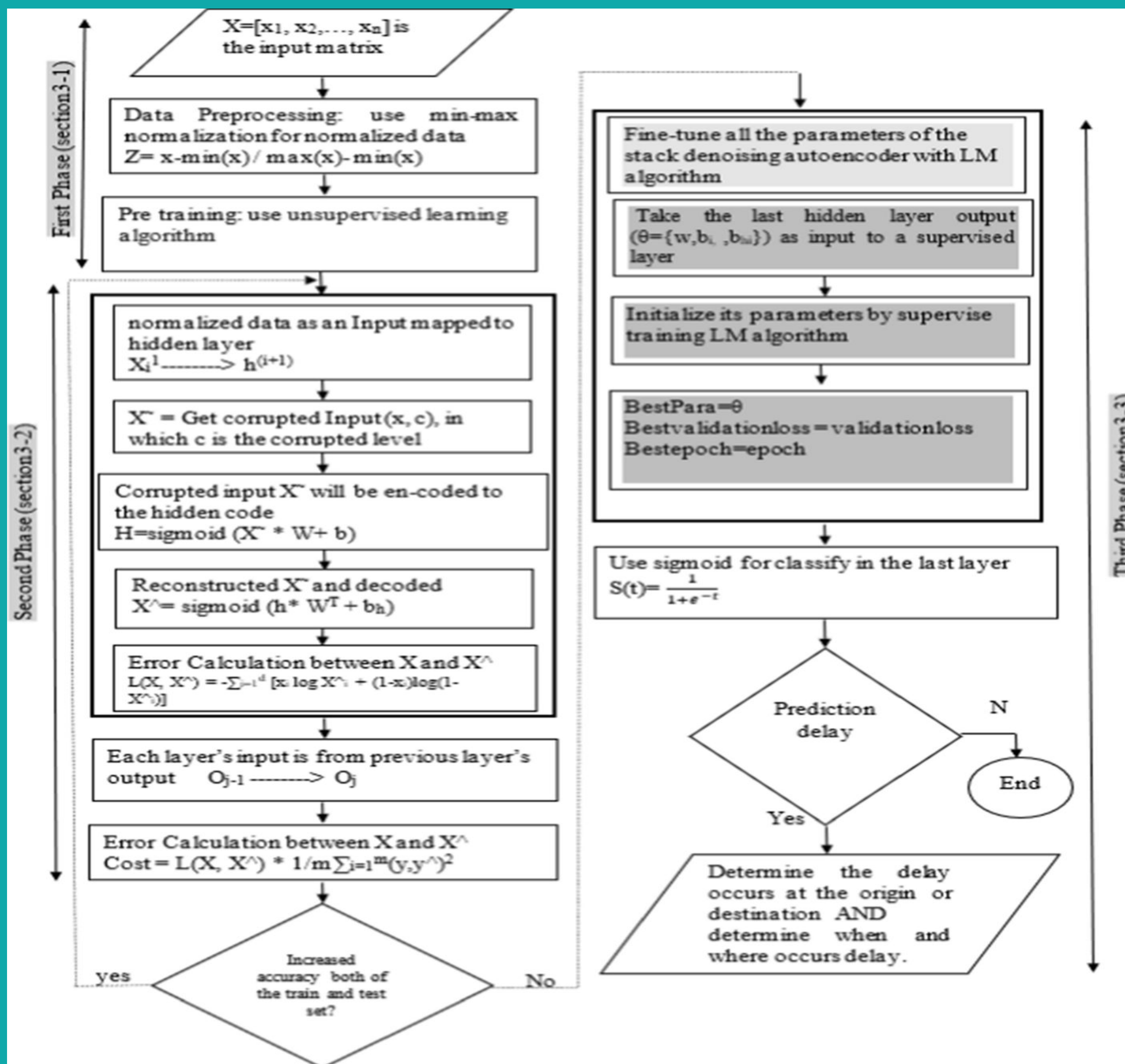
## The solution [ edit ]

Like other numeric minimization algorithms, the Levenberg–Marquardt algorithm is an **iterative** procedure. To start a minimization, the user has to provide an initial guess for the parameter vector *β*. In cases with only one minimum, an uninformed standard guess like *β*<sup>T</sup> = ( 1, 1, . . . , 1 ) will work fine; in cases with **multiple minima**, the algorithm converges to the global minimum only if the initial guess is already somewhat close to the final solution.

In each iteration step, the parameter vector *β* is replaced by a new estimate *β* + *δ*. To determine *δ*, the function *f* (*x<sub>i</sub>*, *β* + *δ*) is approximated by its

## Reference:

https://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt\_algorithm



Technical Image of Layer step for SDA-LM had we successfully implemented.