Flight Delay Predictor

Machine Learning Project

Milestone 2

Mariam ElGhobary
CSCE Department
The American University In Cairo
m.elghobary@aucegypt.edu
SID: 900211608

Youssef Elhagg
CSCE Department
The American University In Cairo
yousseframi@aucegypt.edu
SID: 900211812

THE CHOSEN DATASET

The Flight Delay and Causes [1] dataset was the selected dataset for our model development. It is a combination of multiple datasets of US-based flights during the first half of 2019 making up a net total of 485K sample points and 29 columns. This dataset was selected due to it containing a variety of features posing significant impact on our designated label, departure delay. The dataset also had relatively less feature redundancy in comparison to other datasets. Moreover, the features are accompanied by a dictionary offering a detailed breakdown of what information each feature contains as well as their formats and units enhancing our understanding of each column which is integral when it comes to feature analysis. Plus, the dataset also had near-zero null/NaN values which would significantly facilitate our data cleaning and preprocessing phase.

FEATURE ANALYSIS

1. Feature Relevance and Removal:

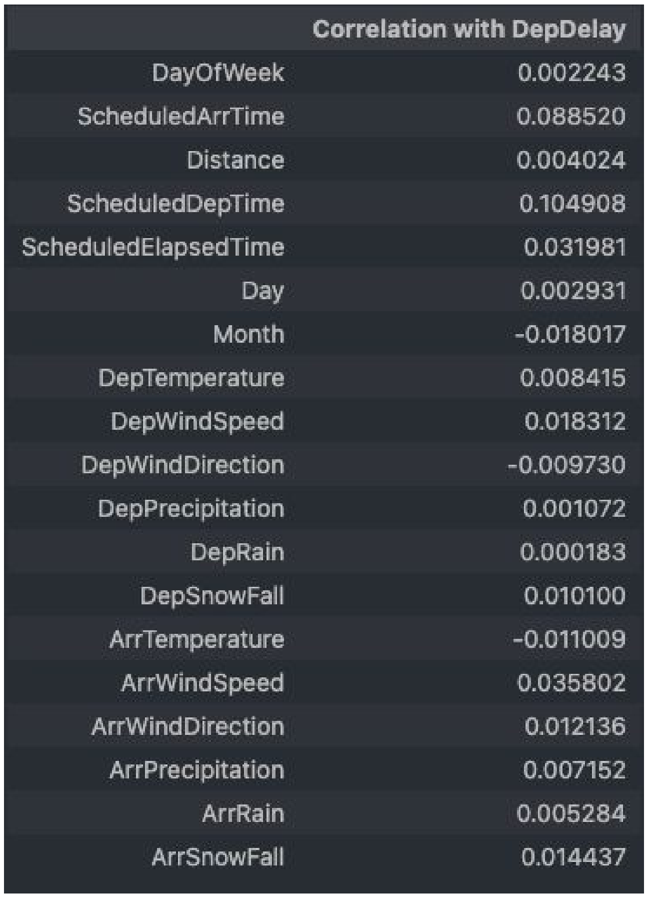
The first major step to be taken was the semantic analysis of the relevance of existing columns in the dataset with regards to their impact on predicting the label. This would enable us to shrink our initial number of columns to non-redundant influential features. For example, the **UniqueCarrier** and **Airline** features both represent the airline, but one uses the unique airline code and the other uses the full airline name. Hence **Airline** can be dropped for redundancy. Similarly, **Origin** and **Origin_Airport** both represent the origin airport name but one using the code and the other with the fully expanded name. And the same can be applied to **Dest** and **Dest_Airport**. Hence dropping **Origin_Airport** and **Dest_Airport** columns leaves us with the same info using the unique airport codes. We now move on to non-inputed features which are essentially all either part of the label or are measured in real life (so part of the predicted data itself) and hence were dropped. These include: **DepTime** and **ArrTime** (actual arrival and departure), as well as, **ActualElapsedTime** , **AirTime** , **TaxiIn**,**TaxiOut**,**CarrierDelay**,**NASDelay**, **WeatherDelay** , **SecurityDelay** and **LateAircraftDelay**. There were also some features that were simply impactless on the flight delay prediction such as **FlightNum** for example which was also dropped. There were some additional labels that simply did not relate to our problem specification or motivation of our project and so were dropped such as:**ArrDelay**, **Canceled**, **Cancellation_Code** and **Diverted**. It's important to note that the **Canceled, Cancellation_Code** and **Diverted** features were all set to false and so even if we sought to utilize them or predict them somehow that would not have been possible.

2. Missing Values:

One of the biggest strengths of our dataset is that it did not have problematic amounts of null values. In fact, our remaining columns post-relevance-analysis and column dropping had no null values whatsoever. However, as part of our feature creation/insertion process which will be discussed later, we needed to utilize the **Origin_Airport** and **Dest_Airport** columns before dropping them and those two had 2656 rows with NaN values. Since replacing the NaNs with the mean, 0 or most recurring value would only mess up, skew and tamper with the logic of our data and since the number of affected rows is relatively miniscule, we decided to cope by simply dropping those rows.

3. Correlation with Label:

We used the **corrwith()** function to determine the correlation between all our numerical features with the label, **DepDelay**. A threshold of 0.8 was set so that any feature with a higher correlation with the label is to be dropped. This is a very crucial step as excessively highly correlated features have an "overpowered" influence on the label and may dismantle the entire machine learning aspect of the project. No features exceeded our threshold and so thankfully no feature-dropping occurred here.



	Correlation with DepDelay
DayOfWeek	0.002243
ScheduledArrTime	0.088520
Distance	0.004024
ScheduledDepTime	0.104908
ScheduledElapsedTime	0.031981
Day	0.002931
Month	-0.018017
DepTemperature	0.008415
DepWindSpeed	0.018312
DepWindDirection	-0.009730
DepPrecipitation	0.001072
DepRain	0.000183
DepSnowFall	0.010100
ArrTemperature	-0.011009
ArrWindSpeed	0.035802
ArrWindDirection	0.012136
ArrPrecipitation	0.007152
ArrRain	0.005284
ArrSnowFall	0.014437

4. **Statistical Distribution:**

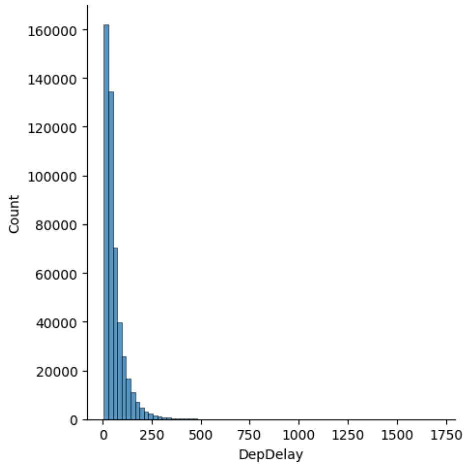
One of the pillars of preprocessing is determining the statistical distribution of numerical features. In order to figure out the distribution of each feature, we utilized a function to compute the best fitting distribution based on the sum squared error. The function, called **get_best_distribution()**, first gets all the numeric features of our dataset. It then constructs a dictionary having the column name as keys, and the best fitted distribution as the value. For each column in the numeric columns, it finds the best of these 3 distributions:

1. Lognormal
2. Normal
3. Uniform

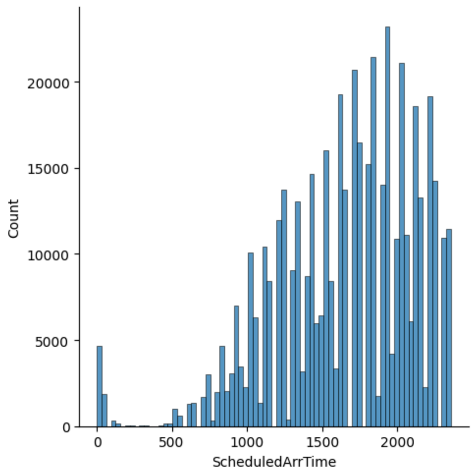
Finally, it stores the best fitting distribution for that column in the dictionary.

The library we used to compute the best fitting distribution is called **Fitter**.

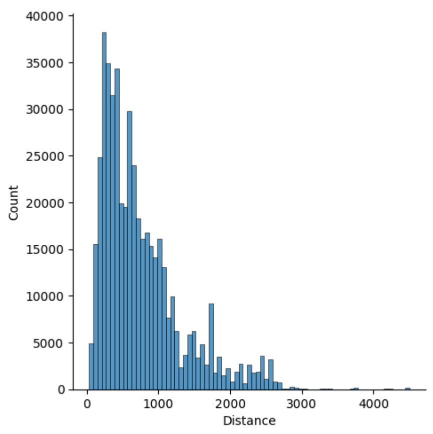
Here are the distributions of each column:



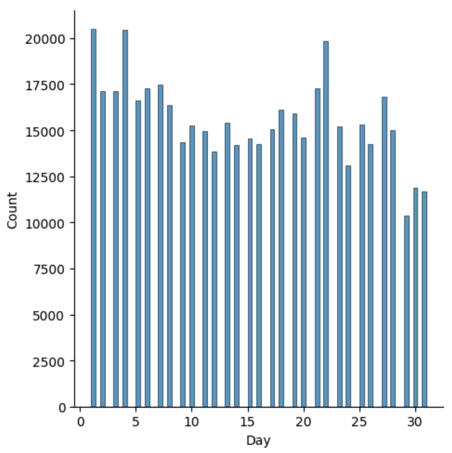
This is the distribution of the Label column. We can see that it's a lognormal distribution.



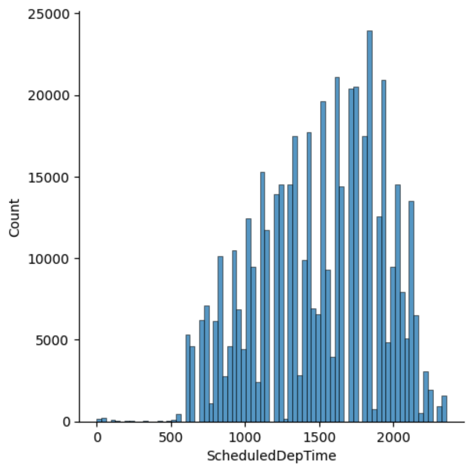
This is the distribution of the Scheduled Arrival Time column. We can see that it's a normal distribution.



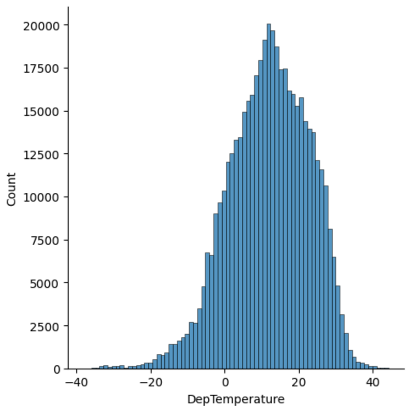
This is the distribution of the Distance column. We can see that it's a lognormal distribution.



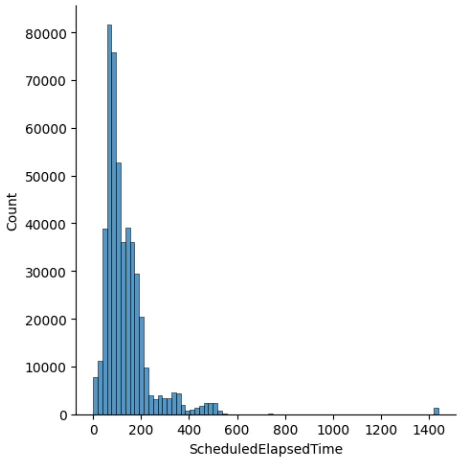
This is the distribution of the Day column. We can see that it's a uniform distribution.



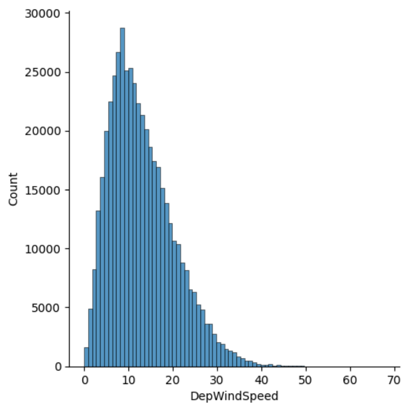
This is the distribution of the Scheduled Departure Time column. We can see that it's a normal distribution.



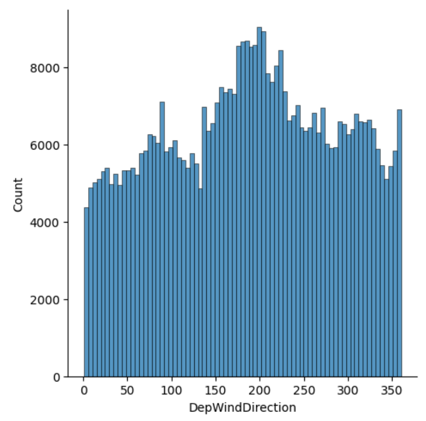
This is the distribution of the Departure Temperature column. We can see that it's a uniform distribution.



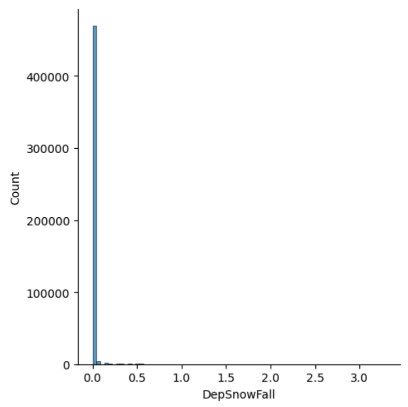
This is the distribution of the Scheduled Elapsed Time column. We can see that it's a normal distribution.



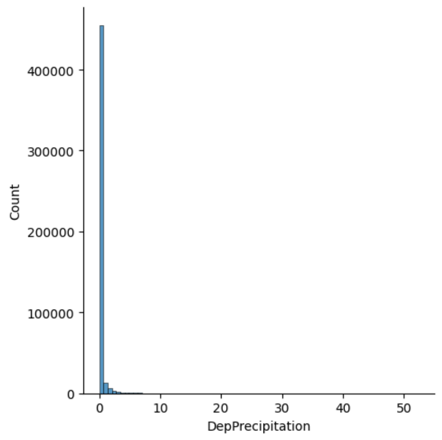
This is the distribution of the Departure Wind Speed column. We can see that it's a lognormal distribution.



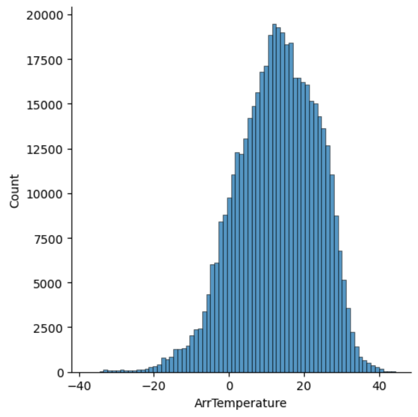
This is the distribution of the Departure Wind Direction column. We can see that it's a uniform distribution.



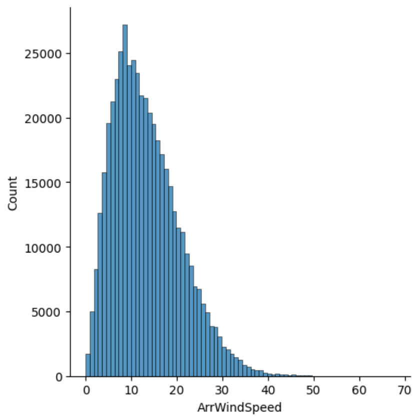
This is the distribution of the Departure SnowFall column. We can see that it's a lognormal distribution.



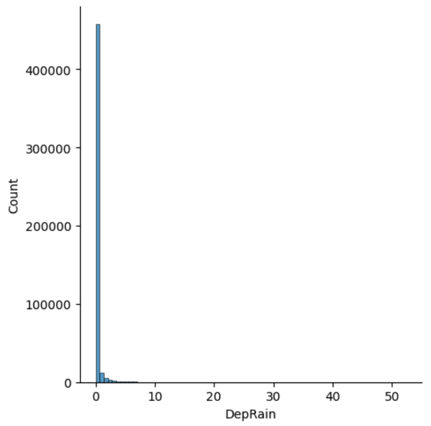
This is the distribution of the Departure Precipitation column. We can see that it's a normal distribution.



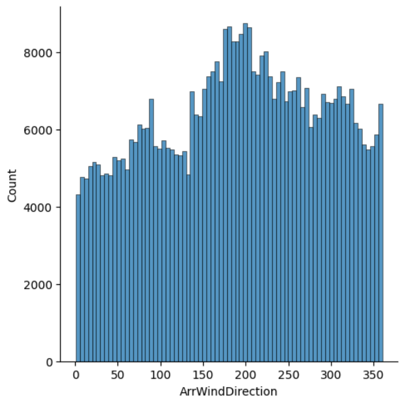
This is the distribution of the Arrival Temperature column. We can see that it's a uniform distribution.



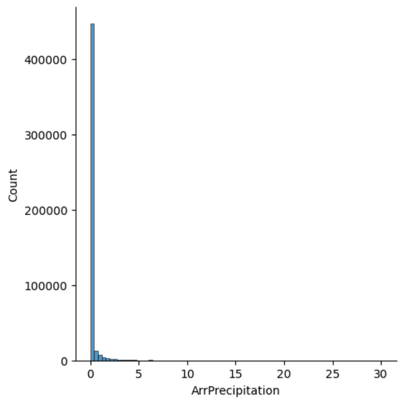
This is the distribution of the Arrival Wind Speed column. We can see that it's a lognormal distribution.



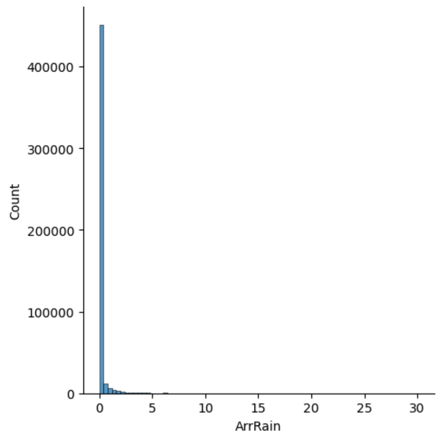
This is the distribution of the Departure Rain column. We can see that it's a normal distribution.



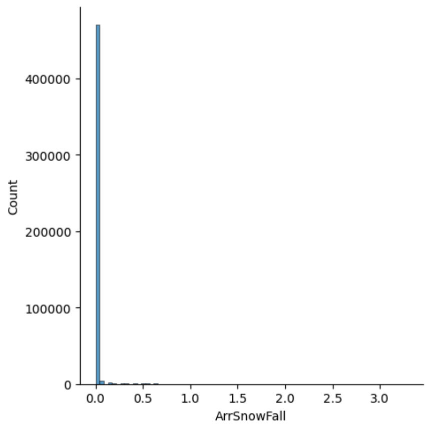
This is the distribution of the Arrival Wind Direction column. We can see that it's a uniform distribution.



This is the distribution of the Arrival Precipitation column. We can see that it's a normal distribution.



This is the distribution of the Arrival Rain column. We can see that it's a normal distribution.



This is the distribution of the Arrival SnowFall column. We can see that it's a lognormal distribution.

CLEANING AND PREPROCESSING

1. **Feature Insertion and Adjustment:**

After analyzing our features and dropping columns that needed to be dropped it became clear to us that some additional work needed to be done. Firstly, to make proper use of the **Date** feature, it was broken down to two features, the day of month and the month, as they are both strongly impacting the label. Meanwhile, the year was disregarded as all data points were in 2019.

We then noticed that the range of values for the **CRSElapsedTime** (expected elapsed time) was quite peculiar as it contained a small number of illogical values such as negative time elapsed and also values significantly inconsistent with expected arrival and departure time. Thus, we recomputed this erroneous column by subtracting expected arrival and departure times. This of course was done after we confirmed through the dictionary and through the majority of the rows of this column that this is the correct calculation.

Later on, we felt as though we could further breakdown the root causes of departure delays, and we deduced that the removed output **WeatherDelay** can have corresponding input features which we can utilize to predict the weather delay component of departure delay. Hence we utilized an API by Google [3] which takes in the airport name and returns its latitude and longitude. The latitude and longitude were then utilized alongside another API [2] which uses them and the date to return various weather condition metrics. The ones we selected were: Temperature, Rain, Snowfall, Precipitation, Wind Direction and Wind Speed due to their relevance to flight delays. This process was carried out for every data point on both the arrival and departure airports.

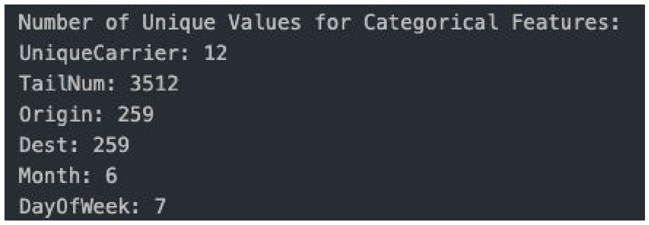
2. **One Hot Encoding Categorical Features:**

When it came to preprocessing our categorical/nominal features, we decided that one-hot-encoding would be the best approach over frequency or hash encoding as it is simple to implement yet collision-free. This was done via the **OneHotEncoder()** and **fit_transform()** functions to create the binary vector representing each one-hot-encoded categorical value and then applying it fully to each row in the designated columns. All of the one-hot-encoded columns were then concatenated with our dataset and their initial columns dropped.

Features such as **UniqueCarrier**, **Origin** and **Dest** were one-hot-encoded as they are simply categorical codes. The **TailNum** attribute was also one-hot-encoded. Initially, we were planning on removing the suffix and prefix from each tail number, however after conducting some research we discovered that they are needed alongside the actual numeric digits and that the numeric digits on their own are in fact deemed as meaningless when it comes to uniquely identifying the aircraft. But since all our flights are US-based all tail numbers had the same prefix and so it was removed for redundancy. The numerical digits alongside the suffix were kept intact and were one-hot-encoded.

After breaking up the **Date** column we put a lot of thought into the best way of which we can preprocess any date-related columns. We concluded that despite features like the **Month** and the **DayOfWeek** being represented by integer digits, they are not actually numerical data. In other words, each number resembles a **Month** or a **DayOfWeek** and is a discrete value with no numerical significance in their ascending/descending order. 1 simply resembles January in the Month column or Monday in the Day of the Week column and since flight traffic associated with each month or day of the week is very significant such features need to be one-hot-encoded.

Below we can see the number of unique values in each of the one-hot-encoded columns. Hence we expect the sum of all of these unique values to be the additional number of columns in our preprocessed dataset (factoring out the dropping of the original 6 columns after OHE).



```
Number of Unique Values for Categorical Features:
UniqueCarrier: 12
TailNum: 3512
Origin: 259
Dest: 259
Month: 6
DayOfWeek: 7
```

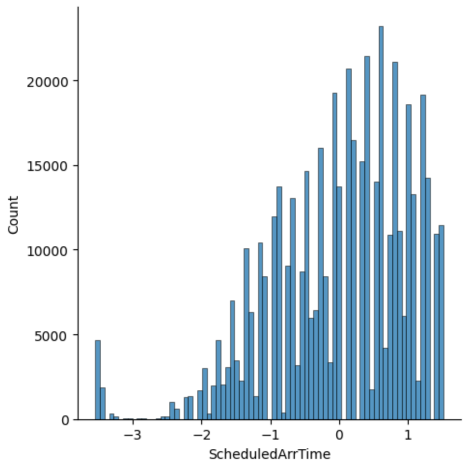
3. **Scaling Numerical Features**

After creating the columns' distribution dictionary, we normalize the columns based on the best normalization method for its best fitted distribution. We decided to use the 3 normalization methods we studied which are:

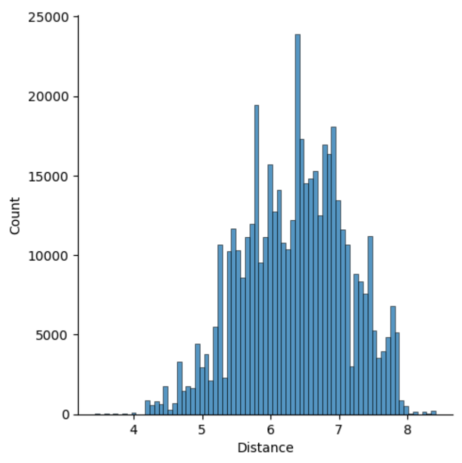
1. Z-Scaling (normal distribution)
2. Min-Max Scaling (uniform distribution)
3. Log (lognormal distribution)

The libraries we used for the Z-Scaling and Min-Max Scaling are both from **sklearn.preprocessing**, which are **StandardScaler** and **MinMaxScaler** respectively. For the Log normalization, we used the built in **numpy.log()** function.

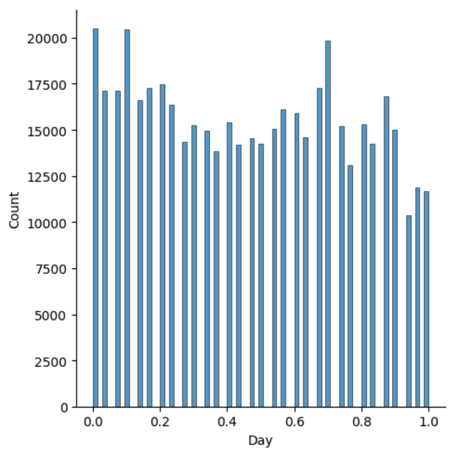
Here are the distribution plots after normalization:



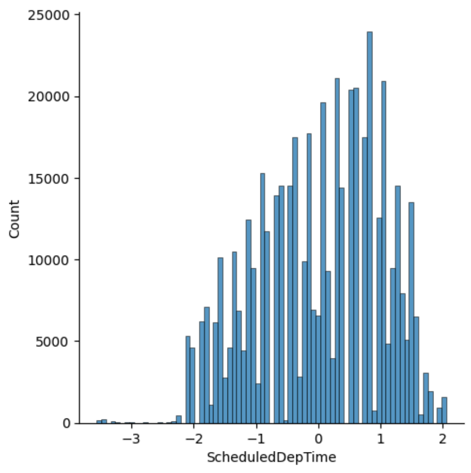
This is the distribution of the Scheduled Arrival Time column, which was normalized using Z-Scaling.



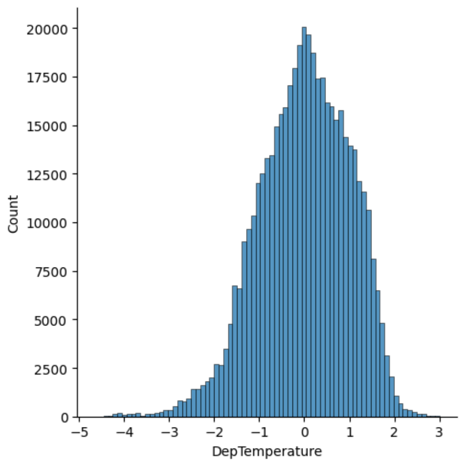
This is the distribution of the Distance column, which was normalized using Log scaling.



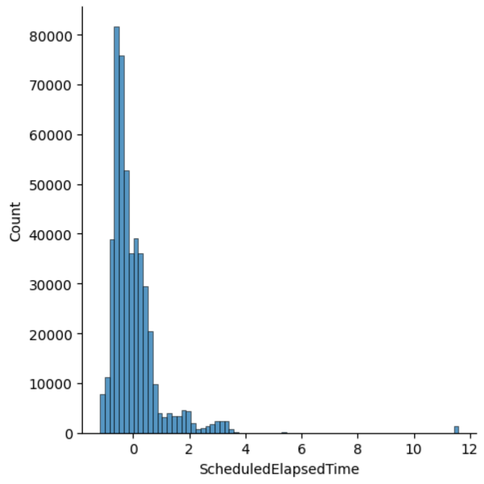
This is the distribution of the Day column, which was normalized using Min-Max Scaling.



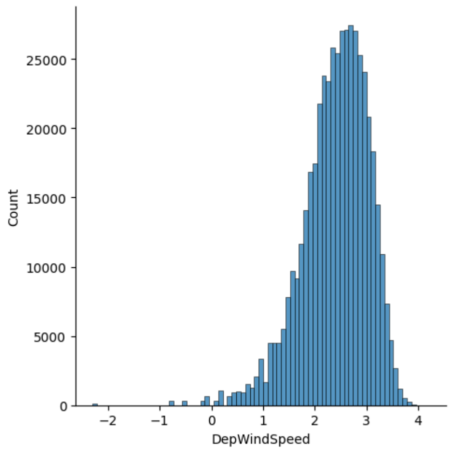
This is the distribution of the Scheduled Departure Time column, which was normalized using Z-Scaling.



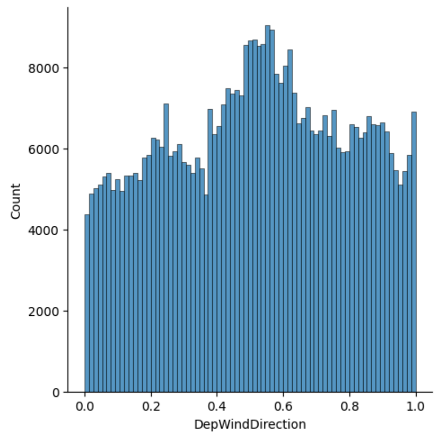
This is the distribution of the Departure Temperature column, which was normalized using Z-Scaling.



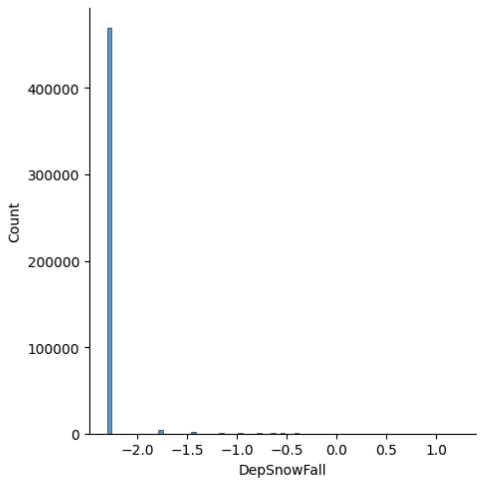
This is the distribution of the Scheduled Elapsed Time column, which was normalized using Z-Scaling.



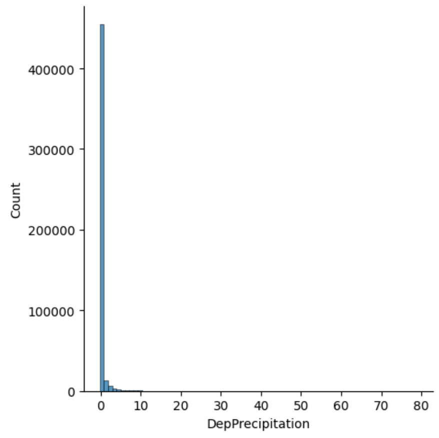
This is the distribution of the Departure Wind Speed column, which was normalized using Log scaling.



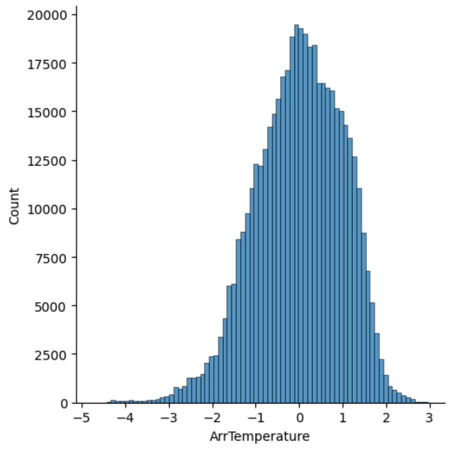
This is the distribution of the Departure Wind Direction column, which was normalized using Min-Max Scaling.



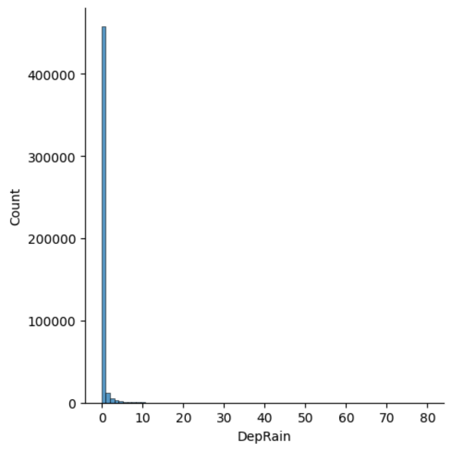
This is the distribution of the Departure SnowFall column, which was normalized using Log scaling.



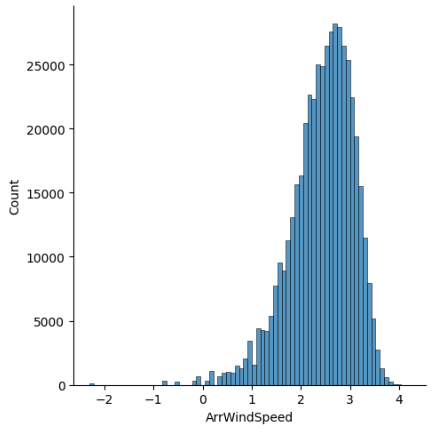
This is the distribution of the Departure Precipitation column, which was normalized using Z-Scaling.



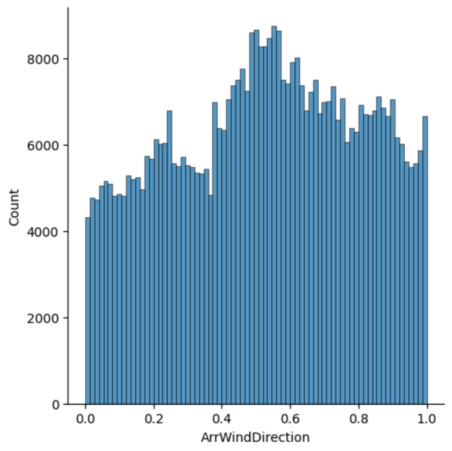
This is the distribution of the Arrival Temperature column, which was normalized using Z-Scaling.



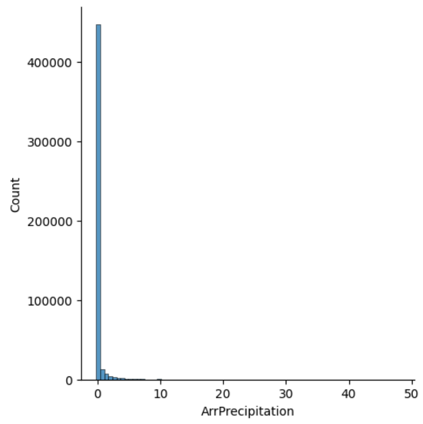
This is the distribution of the Departure Rain column, which was normalized using Z-Scaling.



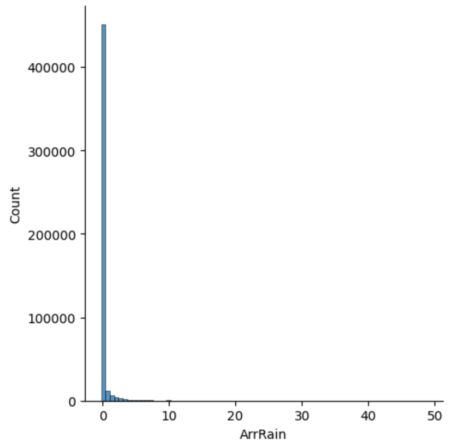
This is the distribution of the Arrival Wind Speed column, which was normalized using Log scaling.



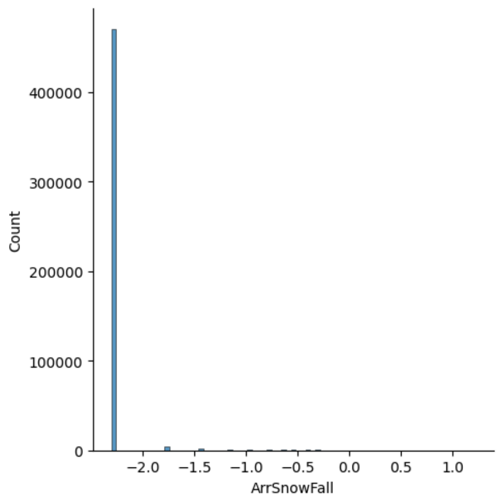
This is the distribution of the Arrival Wind Direction column, which was normalized using Min-Max Scaling.



This is the distribution of the Arrival Precipitation column, which was normalized using Z-Scaling.



This is the distribution of the Arrival Rain column, which was normalized using Z-Scaling.



This is the distribution of the Arrival SnowFall column, which was normalized using Log scaling.

FINAL LIST OF FEATURES

Our finalized and cleansed dataset consists of 481895 rows and 4073 columns which aligns with the unique values shown in the OHE section. The finalized features and label are as follows:

1. **DayOfWeek**
   Resembles day of week by digits where 1 is Monday (OHE)
2. **ScheduledArrTime**
   Expected time of Arrival
3. **UniqueCarrier**
   Airline Carrier Code (OHE)
4. **TailNum**
   Aircraft Tail Number (OHE)
5. **Origin**
   Origin Airport Code (OHE)
6. **Dest**
   Destination Airport Code (OHE)
7. **Distance**
   Distance between Airports in miles
8. **ScheduledDepTime**
   Scheduled departure time
9. **ScheduledElapsedTime**
   Scheduled elapsed time
10. **Day**
    Day of month
11. **Month**
    Months from 1 to 6 (OHE)
12. **DepTemperature**
    Temperature at Departure coordinates
13. **DepWindSpeed**
    Wind speed at Departure coordinates
14. **DepWindDirection**

Wind Direction at Departure coordinates

15. **DepPrecipitation**
    Precipitation at Departure coordinates
16. **DepRain**
    Rain at Departure coordinates
17. **DepSnowFall**
    Snowfall at Departure coordinates
18. **ArrTemperature**
    Temperature at Arrival coordinates
19. **ArrWindSpeed**
    Wind speed at Arrival coordinates
20. **ArrWindDirection**
    Wind direction at Arrival coordinates
21. **ArrPrecipitation**
    Precipitation at Arrival coordinates
22. **ArrRain**
    Rain at Arrival coordinates
23. **ArrSnowFall**
    Snow fall at Arrival coordinates
24. **DepDelay**
    Departure Delay in minutes (label)

REFERENCES

[1] Trivedi, P. 2022. Flight Delay and Causes. Kaggle dataset. Available at:https://www.kaggle.com/datasets/undersc0re/flight-delay-and-causes

[2] Open-Meteo.com. 2024. Historical Weather API. Open-Meteo.com. Available at: https://open-meteo.com/en/docs/historical-weather-api/

[3] Google Developers. 2024. Geocoding request and response | Geocoding API. Google Developers. Available at: https://developers.google.com/maps/documentation/geocoding/requests-geocoding