TTC Bus Delay Report

Agam Sanghera, Ashita Diwan, Cheng Zhang, Yichun Liu

2024-12-07

Table of contents

Summary

This report examines the 2024 TTC bus delay dataset to forecast delay times based on factors such as route, incident type, location, and time. We aim to classify delays into short, medium, and long categories using a logistic regression model. This study acts as a foundation for the implementation of real-time prediction models that could aid the Toronto Transit Commission in resource allocation and improving schedule adherence.

Introduction

Public transportation systems, such as Toronto's TTC, are essential for facilitating commuter mobility. However, delays are unavoidable and can affect the efficiency of services. Anticipating these delays may enhance operational decision-making and increase commuter satisfaction. The objective of this analysis is to identify the primary factors contributing to delays and to accurately forecast the duration of these delays by utilizing route, incident types, location, and time-related features as predictors.

Data

The data for this analysis was sourced from the open.toronto.ca website, with a specific emphasis on the bus delay data for the year 2024. This dataset contains multiple incident reports, including information on route number, delay duration, incident type, and incident location. Raw data can be found here.

EDA Analysis

We will first conduct an EDA analysis on the data. The steps include:

1. **Loading and Preprocessing Data:** Handling missing values, converting timestamp data to day parts, and cleaning data fields irrelevant to our delay analysis.
2. **Visualization:** Analyze the distribution of delays, identify top routes and locations with frequent delay incidents, and visualize delays based on day and incident type.

```
DataTransformerRegistry.enable('vegafusion')
```

Loading and Fixing Data

Table 1: Snippet of TTC bus delay data

	Date	Route	Time	Day	Location	Incident	Min Delay	Min
0	2024-01-01	89	02:08	Monday	KEELE AND GLENLAKE	Vision	10	20
1	2024-01-01	39	02:30	Monday	FINCH STATION	General Delay	20	40
2	2024-01-01	300	03:13	Monday	BLOOR AND MANNING	General Delay	0	0
3	2024-01-01	65	03:23	Monday	PARLIAMENT AND BLOOR	Security	0	0
4	2024-01-01	113	03:37	Monday	MAIN STATION	Security	0	0

A snippet of the TTC bus delay data is shown in Table 1.

This dataset has 45300 rows and 10 columns.

The details of each column are shown in Table 2.

Now, let's split 'Date' column into 'Date_' and 'Month' (year is not needed since this is for 2024), and convert 'Time' into 'Hours' so it more useful in the analysis later on.

Table 2: Details of each column

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45300 entries, 0 to 45299
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       45300 non-null  datetime64[ns]
 1   Route      44861 non-null  object
 2   Time       45300 non-null  object
 3   Day        45300 non-null  object
 4   Location   45300 non-null  object
 5   Incident   45300 non-null  object
 6   Min Delay  45300 non-null  int64
 7   Min Gap    45300 non-null  int64
 8   Direction  38343 non-null  object
 9   Vehicle    45300 non-null  int64
dtypes: datetime64[ns](1), int64(3), object(6)
memory usage: 3.5+ MB
```

Preprocessing Data

Now that the data is loaded in, we want to make sure we don't perform our analysis will null values. So, we will go through and identify any columns that have big number of null values and determine if they are worth keeping for analysis or not.

```
Route         439
Day             0
Location        0
Incident        0
Min Delay       0
Min Gap         0
Direction    6957
Vehicle         0
Date_           0
Month           0
Hour            0
dtype: int64
```

	Route	Day	Location	Incident	Min Delay	Min Gap
1	39	Monday	FINCH STATION	General Delay	20	40
2	300	Monday	BLOOR AND MANNING	General Delay	0	0
4	113	Monday	MAIN STATION	Security	0	0
12	600	Monday	LAKESHORE AND THIRTY S	Cleaning - Unsanitary	25	50
13	300	Monday	KIPLING STATION	Emergency Services	0	0
23	41	Monday	KEELE AND IAN MACDONAL	Diversion	780	793
24	85	Monday	TORONTO ZOO	Security	20	40
27	NaN	Monday	EGLINTON DIVISION	Operations - Operator	0	0
28	12	Monday	KENNEDY STATION	Emergency Services	30	60
34	900	Monday	KIPLING STATION	Operations - Operator	14	28
35	66	Monday	OLD MILL STATION	Mechanical	25	50
38	87	Monday	EAST YORK ACRES	General Delay	52	62
41	96	Monday	HUMBERLINE LOOP	Security	10	20
49	60	Monday	PIONEER VILLAGE STATIO	Collision - TTC	10	20
56	106	Monday	SHEPPARD WEST STATION	Mechanical	12	24
57	61	Monday	AVENUE AND 401	Diversion	350	360
69	NaN	Monday	GUILDWOOD AND KINGSTON	Utilized Off Route	0	0
73	165	Monday	WESTON AND LANYARD	Diversion	10	49
99	11	Monday	BAYVIEW AND STEELES	Security	24	48
104	300	Tuesday	KENNEDY STATION	Security	0	0

Maybe deleting the rows isn't such a good idea because it makes up for a lot of overall data in the dataset. Instead, let's drop the column since we already have information about the route. For that reason, it is also okay to drop 'Vehicle' column as it is not needed for our goal of observing delays in busses.

	Route	Day	Location	Incident	Min Delay	Min Gap	Date_	I
0	89	Monday	KEELE AND GLENLAKE	Vision	10	20	2024-01-01	1
1	39	Monday	FINCH STATION	General Delay	20	40	2024-01-01	1
2	300	Monday	BLOOR AND MANNING	General Delay	0	0	2024-01-01	1
3	65	Monday	PARLIAMENT AND BLOOR	Security	0	0	2024-01-01	1
4	113	Monday	MAIN STATION	Security	0	0	2024-01-01	1

Now, let's look at the Route column and its NaNs. This column is particularly important for us for our goal.

	Route	Day	Location	Incident	Min Delay	Min Ga
27	NaN	Monday	EGLINTON DIVISION	Operations - Operator	0	0

	Route	Day	Location	Incident	Min Delay	Min Ga
69	NaN	Monday	GUILDWOOD AND KINGSTON	Utilized Off Route	0	0
213	NaN	Tuesday	QUEENSWAY GARAGE	Mechanical	0	0
248	NaN	Tuesday	3RD FLOOR GUNN BUILDIN	Operations - Operator	0	0
293	NaN	Wednesday	WILSON TRAINING OPERAT	Operations - Operator	0	0
350	NaN	Wednesday	RUSSELL CARHOUSE	Mechanical	0	0
602	NaN	Thursday	FINCH STATION	Operations - Operator	12	24
624	NaN	Thursday	ARROW GARAGE	Collision - TTC	0	0
638	NaN	Thursday	123 PARKWAY FOREST DRI	Utilized Off Route	0	0
659	NaN	Friday	MOUNT DENNIS DIVISION	Operations - Operator	0	0

```
Route          0
Day          363
Location     363
Incident     363
Min Delay    363
Min Gap      363
Direction     68
Vehicle      363
Date_        363
Month        363
Hour         363
dtype: int64
```

Since all the NaN routes have 0 delays, it is safe to drop these rows. We will also remove all other null values to ensure a clean dataset to work with

```
Route        0
Day          0
Location     0
Incident     0
Min Delay    0
Min Gap      0
Date_        0
Month        0
Hour         0
dtype: int64
```

	Route	Day	Location	Incident	Min Delay	Min Gap	Date_	
0	89	Monday	KEELE AND GLENLAKE	Vision	10	20	2024-01-01	
1	39	Monday	FINCH STATION	General Delay	20	40	2024-01-01	
2	300	Monday	BLOOR AND MANNING	General Delay	0	0	2024-01-01	
3	65	Monday	PARLIAMENT AND BLOOR	Security	0	0	2024-01-01	
4	113	Monday	MAIN STATION	Security	0	0	2024-01-01	

Visualizing

Now it's the fun part: visualizations!

Let's look at how the delays are distributed in this data

```
alt.Chart(...)
```

This plot shows that majority of the delays occur within 0 to 25 mins. This can indicate that most delays are short in duration.

Now let's take a look at top 20 routes with highest delay incidents

```
alt.Chart(...)
```

This plot shows that majority of the incidents are occuring on route 32 with close to 1,400 counts of delays, followed by route 32 and 36. As we move towards other routes, there seems to be a smooth decline in delay counts, showing a decreasing trend.

```
                    Location  Delay Count
3675          KENNEDY STATION         1263
3955          KIPLING STATION          820
2326         EGLINTON STATION          688
7713           WILSON STATION          628
2734            FINCH STATION          623
5544  PIONEER VILLAGE STATIO          575
5986  SCARBOROUGH CENTRE STA          431
7365           WARDEN STATION          419
8026       YORK MILLS STATION          377
1688        DON MILLS STATION          371
```

```
alt.Chart(...)
```

```
Delay Counts by Day of the Week
         Day  Delay Count
0     Friday         6541
5    Tuesday         6198
4   Thursday         6189
6  Wednesday         6038
2   Saturday         5662
1     Monday         5441
3     Sunday         4149
```

```
alt.Chart(...)
```

```
Delay Counts by Incident Type
                             Incident  Delay Count
6                          Mechanical        14846
7              Operations - Operator         7483
2                           Diversion         3125
9                            Security         2838
1                  Collision - TTC          2772
3                 Emergency Services         1978
10                Utilized Off Route         1953
0              Cleaning - Unsanitary         1764
11                             Vision         1372
4                     General Delay          1192
5                     Investigation           772
8   Road Blocked - NON-TTC Collision          123
```

```
alt.Chart(...)
```

Linear Regression Model for Prediction

The EDA analysis was very informative in understanding the columns of interest for this project. The columns which seem to impact the Min Delay are Incident and Route, so this analysis will be focused around using these 2 columns to create a Logistic Regression model, to predict the expected delay given a route and incident. The delay output will be categorized into “Short”, “Medium” or “Long”. Cross-validation and randomized grid search were applied for hyperparameter tuning to enhance model performance.

First the required libraries are loaded to perrform the analysis.

To perform the analysis, the cleaned data from the EDA will be used.

	Route	Day	Location	Incident	Min Delay	Min Gap	Da
0	89	Monday	KEELE AND GLENLAKE	Vision	10	20	20
1	39	Monday	FINCH STATION	General Delay	20	40	20
2	300	Monday	BLOOR AND MANNING	General Delay	0	0	20
3	65	Monday	PARLIAMENT AND BLOOR	Security	0	0	20
4	113	Monday	MAIN STATION	Security	0	0	20
...	...	...	...	...	...	...	...
44856	63	Monday	KING AND DOWLING	Vision	10	20	20
44857	32	Monday	EGLINTON AND OAKWOOD	Vision	13	18	20
44858	63	Monday	OSSINGTON STATION	Vision	10	20	20
44859	31	Monday	COXWELL STATION	Emergency Services	17	34	20
44860	54	Monday	ROGUE HILL GO STATION	Vision	10	20	20

Seeing as the data has alot of extreme values, the outliers will be eliminated. These outliers are delays greater than 30 minutes and less than 1 minute.

Next, the `Min Delay` column needs to be pre processed as there are multiple unique values and predicting on such values is not efficient, thus the times are divided in 3 classes: * Short Delay: delays less than or equal to 10 minutes * Medium Delay: Delays of more than 10 minutes but less than or equal to 20 minutes * Long Delays: Delays greater than 20 minutes

	Route	Day	Location	Incident	Min Delay	Min G
0	89	Monday	KEELE AND GLENLAKE	Vision	Short Delay	20
1	39	Monday	FINCH STATION	General Delay	Medium Delay	40
2	320	Monday	YONGE AND QUEENSQUAY	Operations - Operator	Short Delay	16
3	171	Monday	MOUNT DENNIS GARAGE	General Delay	Medium Delay	20
4	12	Monday	VICTORIA PARK AND DANF	Emergency Services	Long Delay	42
...	...	...	...	...	...	...
35251	63	Monday	KING AND DOWLING	Vision	Short Delay	20
35252	32	Monday	EGLINTON AND OAKWOOD	Vision	Medium Delay	18
35253	63	Monday	OSSINGTON STATION	Vision	Short Delay	20
35254	31	Monday	COXWELL STATION	Emergency Services	Medium Delay	34
35255	54	Monday	ROGUE HILL GO STATION	Vision	Short Delay	20

```
ColumnTransformer(transformers=[('num', StandardScaler(), ['Hour', 'Month']),
                                ('cat', OneHotEncoder(handle_unknown='ignore'),
                                 ['Location', 'Route', 'Incident', 'Day'])])
```

```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num', StandardScaler(),
                                                  ['Hour', 'Month']),
                                                 ('cat',
                                                  OneHotEncoder(handle_unknown='ignore'),
                                                  ['Location', 'Route',
                                                   'Incident', 'Day'])])),
                ('model', LogisticRegression(max_iter=2000, random_state=123))])
```

Perform cross validation to see how the model performs

	fit_time	score_time	test_score	train_score
0	0.779215	0.012383	0.690480	0.772459
1	0.810716	0.012429	0.699876	0.766831
2	0.834245	0.012829	0.689949	0.771927
3	0.896829	0.012810	0.683389	0.774144
4	0.839267	0.012884	0.702482	0.767550

We then conduct hyperparameter tuning to optimize results, using a parameter grid of values of the hyperparameter C in the range 10^{-5} to 10^{10} and then using it in a Randomized Search.

```
Grid size: 15
```

```
RandomizedSearchCV(estimator=Pipeline(steps=[('preprocessor',
                                              ColumnTransformer(transformers=[('num',
                                                                               StandardScale
                                                                               ['Hour',
                                                                                'Month']),
                                                                              ('cat',
                                                                               OneHotEncoder
                                                                               ['Location',
                                                                                'Route',
                                                                                'Incident',
                                                                                'Day'])])),
                                             ('model',
                                              LogisticRegression(max_iter=2000,
                                                                 random_state=123))]),
                   n_iter=15, n_jobs=-1,
                   param_distributions={'model__C': [1e-05, 0.0001, 0.001, 0.01,
                                                     0.1, 1, 10, 100, 1000,
```

```
                                                  10000, 100000, 1000000,
                                                  10000000, 100000000,
                                                  1000000000]},
                   random_state=123, return_train_score=True)
```

The selected model has the best train score with the smallest difference in train and validation scores, and indicating less overfitting. Thus the value of 0.1 is the value chosen for the final model.

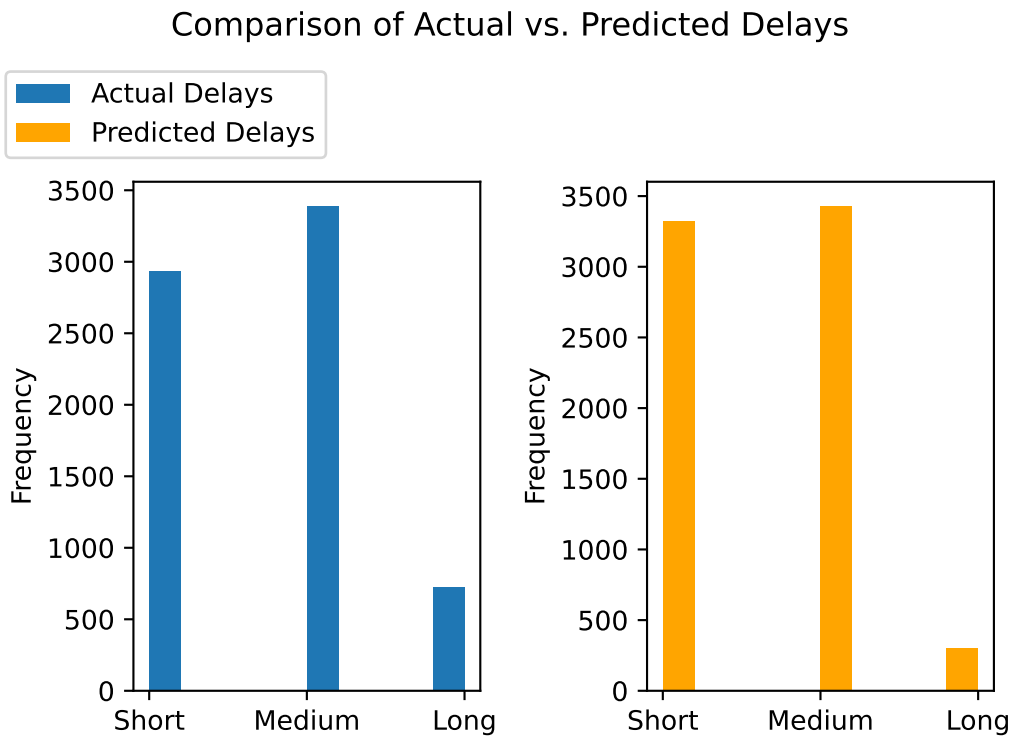
	params	mean_train_score	mean_test_score
4	{'model__C': 0.1}	0.717478	0.693342

Create optimized pipiline with updated C

```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num', StandardScaler(),
                                                  ['Hour', 'Month']),
                                                 ('cat',
                                                  OneHotEncoder(handle_unknown='ignore'),
                                                  ['Location', 'Route',
                                                   'Incident', 'Day'])])),
                ('model',
                 LogisticRegression(C=0.1, max_iter=2000, random_state=123))])
```

Making predictions on the optimized model

Create visualization for visually comparing the model performance.



Discussion of Results

The EDA analysis of the TTC bus delay data uncovers several key insights. The distribution of delays is significantly biased towards shorter durations, with the majority of delays lasting less than 20 minutes. This indicates that although delays occur frequently, they are typically short and manageable.

Kennedy Station stands out as the site with the highest number of delays, suggesting that either infrastructural limitations or a high volume of passengers are contributing to these frequent interruptions. The analysis of delay counts by day of the week indicates that Tuesday through Friday are the days with the highest delays, suggesting a potential correlation with weekday commuter traffic.

Mechanical issues are the primary cause of delays, comprising a substantial portion, followed by operator-related operations and diversions. This finding indicates potential areas for intervention, such as improved maintenance or optimized scheduling, to mitigate delay incidents.

The results of the logistic regression model show moderate effectiveness in predicting delay durations. Predicted frequencies for short and medium delays correspond with actual data; however, the model underpredicts long delays, highlighting the complexity of accurately capturing extended durations and their contributing factors. We could explore more advanced

predictive models to improve accuracy. Furthermore, more data integration such as weather conditions could enhance model performance.

Data Validation

```
Error: File is either missing or not in CSV format.
```

The warning suggests that the pd.read_csv() method couldn't infer a uniform date format for the Date column. This happens when the column contains inconsistent date formats or unparseable values. This was fixed when we used parse_dates=['Date'] in ttc = pd.read_csv('data/ttc-bus-delay-data-2024.csv', parse_dates=['Date'])