Boulder BCycle Wait Time Estimation and Analysis

Niranjan Cholendiran, Santhosh Pattamudu Manoharan

1. Abstract

BCycle is an important public bike sharing service frequently used by Boulder's student population. However, because of its popularity, the service often faces a shortage of bikes, leading to users considering alternatives due to a lack of awareness regarding wait times. To address this issue, our work analyzes the usage patterns, identifies key contributing factors, and uses this knowledge to develop a machine learning model, that estimates the wait time with an overall accuracy of 66%, surging to 75% during peak periods when it is highly needed.

2. Introduction

BCycle is a public bicycle-sharing company that provides electric cycles for public use. It is in 47 locations within the United States. They provide a membership program for the public where one can sign up, pay upfront, and use the cycles for the allotted time. BCycle also provides a mobile application where the users can track the location of charging stations, availability of cycles, navigation between stations, and more.

Out of these 47 locations, Boulder is one of the most popular ones mainly because it is widely used by the students at the University of Colorado Boulder to commute between their houses and campus. This reflects BCycle’s decision to provide free membership for CU students.

Though BCycle’s mobile application has various features, it lacks a critical component: informing users about the wait time for the next available cycle when stations are out of cycles or have fewer than required. Given BCycle's popularity in Boulder, stations frequently run out of cycles, leaving without any knowledge of the potential wait times.

This is where our project comes to play in, where we have analyzed the usage pattern of cycles across different stations, identified the factors that influence them, and built a Machine Learning model that estimates the bicycle waiting period. This algorithm is inspired by various robust wait time prediction models available in the Data Science community [1] [2] [3]. The scope of this project is restricted to the 10 most popular stations in Boulder.

3. Data overview

General Bikeshare Feed Specification (GBFS) is a standard for open mobility data that was created in 2014 by Mitch Vars, in collaboration with public, private, and non-profitable mobility system owners and technology vendors without API Token. The vision of this program is to integrate the live status of the bikes and docks into other applications that will enhance bike usage. The Boulder BCycle is a proud supporter of this program, so were able to collect the live status of the Boulder BCycle from the 31st of Oct to Dec 7th of 2023.

The entire data flow architecture is presented in Fig 1. We have used GBFS to collect BCycle station status data, along with weather and temperature data using Open Mateo API. Additionally, we used the CU Class Search website to get information regarding the class schedules as all these features influence the cycle usage pattern.

GBFS API does not provide historical data rather it provides the status at the point when the API call is initiated. Hence, we set up an Amazon EC2 instance in which we scheduled a Python script that pulls data from all the mentioned sources continuously at a 3-minute frequency.

Subsequently, the collected data is stored in a Postgres server, from which the data is again pulled by a Python preprocessing pipeline, which cleans and preprocesses the data. The preprocessed data is again stored in the Postgres server which is used for Exploratory Data Analysis (EDA), Visualization, and modeling.

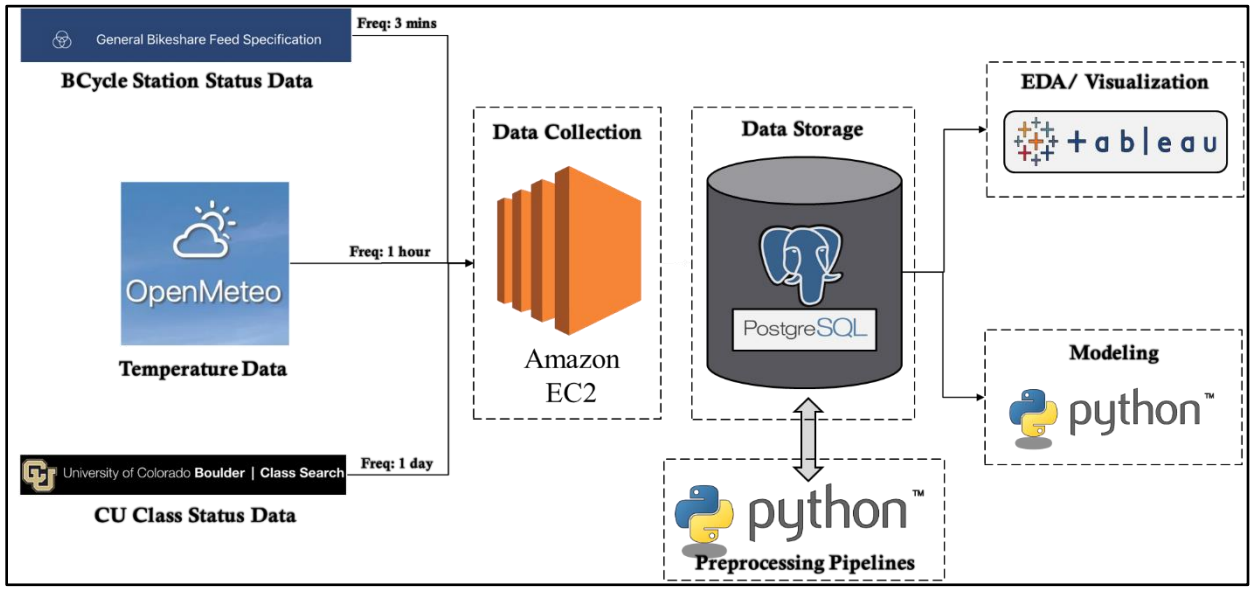

Fig 1: Data Flow Architecture

The upcoming section provides a detailed description of the data collected from different sources mentioned above.

3.1. Data Description:

Data Sources: GBFS

Column	Column Description
station_last_updated	Time is updated by the station to the GBFS server.
status_last_reported	Time reported by the station to the GBFS server
station_id	This specifies the station ID
station_name	This specifies the station name
station_address	This specifies the station address

station_longitude	This specifies the longitude
station_latitude	This specifies the latitude
station_is_returning	This specifies whether the bike is returned to its home station
station_is_renting	This specifies whether the station is renting or not
station_is_installed	This specifies whether the station is installed or not
station_type	This specifies the station type.
docks_available	This specifies the number of docks available
bikes_available	This specifies the number of bikes available
electic_bikes_available	This specifies the number of electric bikes available
smart_bikes_available	This specifies the number of smart bikes available
classic_bikes_available	This specifies the number of classical bikes available

Data Sources: Open Mateo

Column	Column Description
datetime_temperature	Timestamp
temperature_2m	Air temperature at 2 meters above the ground
precipitation_probability	Probability of precipitation with more than 0.1 mm of the preceding hour. Probability is based on ensemble weather models with 0.25° (~27 km) resolution. 30 different simulations are computed to better represent future weather conditions.
rain	Rain from large-scale weather systems of the preceding hour in millimeter
snowfall	Snowfall amount of the preceding hour in centimeters. For the water equivalent in millimeters, divide by 7. E.g. 7 cm snow = 10 mm precipitation water equivalent
visibility	Viewing distance in meters. Influenced by low clouds, humidity, and aerosols. Maximum visibility is approximately 24 km.

Data Sources: CU Class Search

Column	Column Description
cu_class_status	Details regarding whether CU classes are in session or not.

4. Statistical Modelling methods

4.1. Preprocessing.

Round off the timestamp from GBFS:
We are collecting data continuously through a Python script. The process starts with the code fetching the data from multiple APIs, combining them, and saving it in PostgreSQL. At times, due to data latency, this process might take some time and it affects the temporal sequence. For example, data that was supposed to be pulled at 13:00:00 will be pulled at 13:00:45. Hence, we rounded it off to the nearest ideal interval.

Handling Missing values
Due to some network errors, our script stopped working, which affected our data collection process. To handle these errors, we updated the missing values with the rows from the previous date.

Creating Synthetic Features

We have created some synthetic columns for our dataset to increase the relevance and utility of our analysis.

Station Capacity - This feature specifies the capacity of the particular station. It is the sum of bikes available and docks available.

New cycles and docks: This indicates the number of new bikes and docks available. It is calculated by subtracting the bike available from the previous interval.

Bike wait time - This indicates the waiting duration for the new bike in the top 10 popular stations.

From analyzing the dataset, we have found that these top 10 stations have higher variance despite having fewer data points.

4.2. Exploratory Data Analysis

We will cover some of the crucial Exploratory Data Analysis that focuses on decoding the BCycle usage patterns across Boulder. The insights that are covered in this section are later used in building the classification model.

Popular BCycle Stations

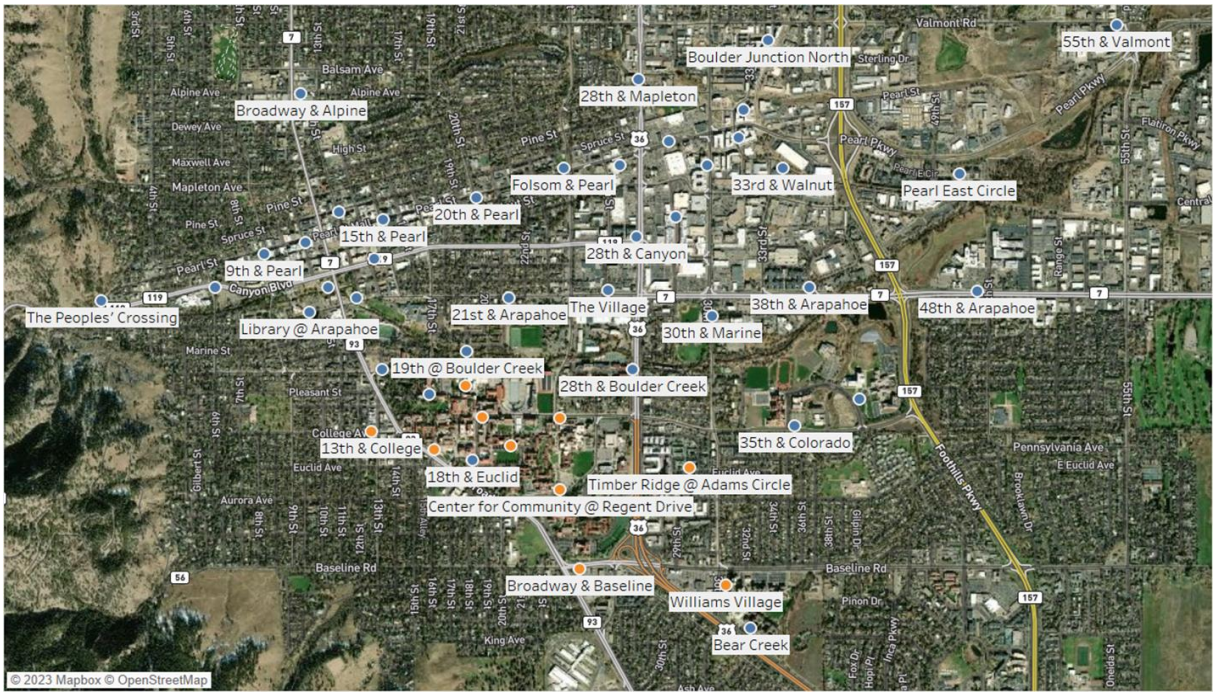

Fig 2: List of BCycle station locations in Boulder locations in Boulder

There is a total of 54 BCycle stations located in Boulder (Fig 2) covering the entire city. However, not every station is frequently used. Since BCycle is popular in Boulder among the students of CU as they get free membership, there are a handful of stations that are frequently used.

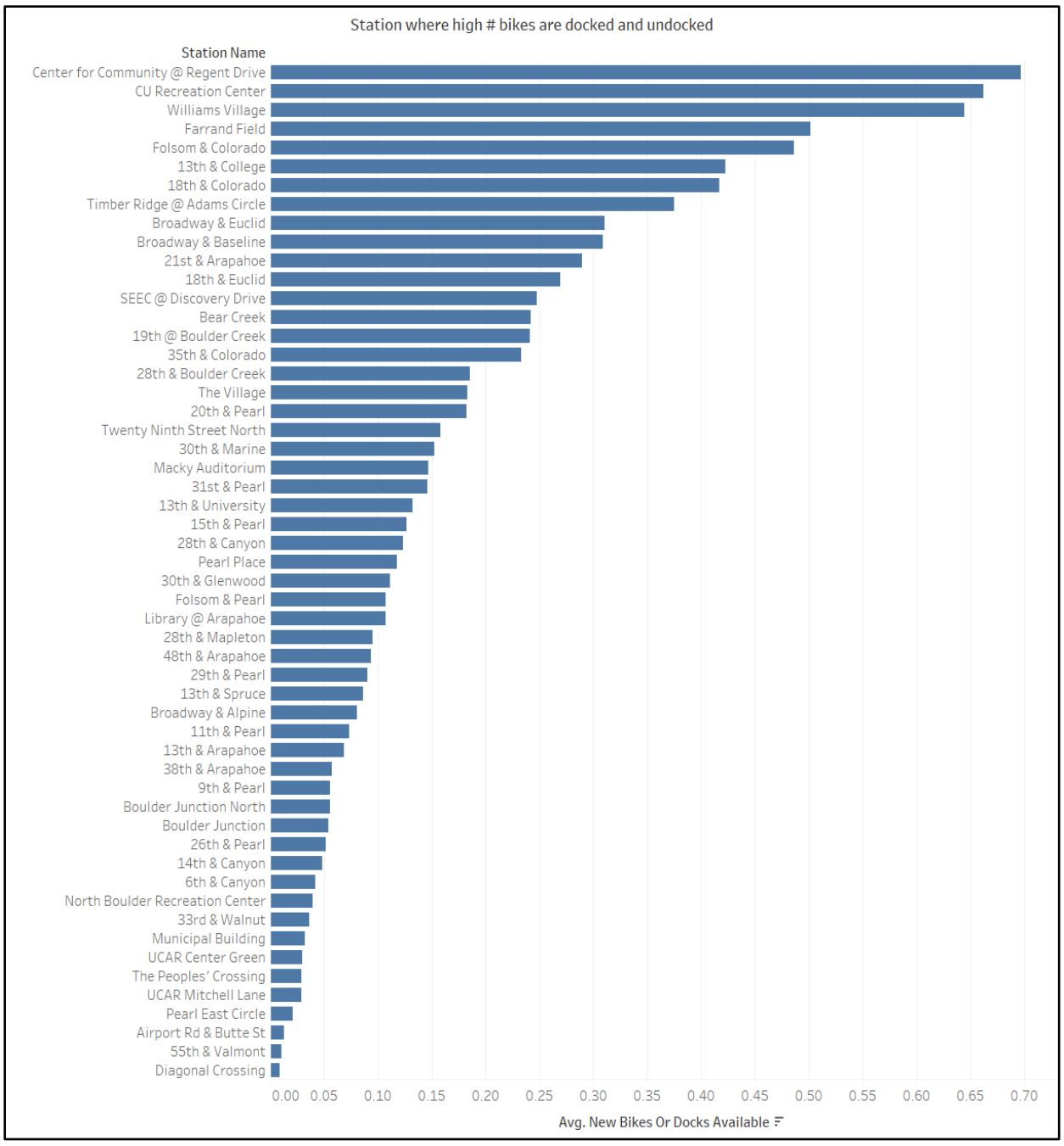

Fig 3: Stations sorted based on popularity.

All above mentioned stations are sorted based on the usage frequency in Fig 3. The usage frequency is determined by the average number of bikes that are docked and undocked in a day. This metric highlights the frequency of cycle usage.

For our analysis, we have selected the top 10 popular stations as they run out of cycles more often than others and our project's impact is more significant in these areas. The locations of these selected 10 stations have been presented in Fig 4. Anyone familiar with the Boulder map can easily realize that all these stations are located near the hotspots of the CU campus. Notably, stations near Norlin Library, Recreational Center,

Engineering Building, Center for Community, Williams Village Campus, and others rank highest on the list.

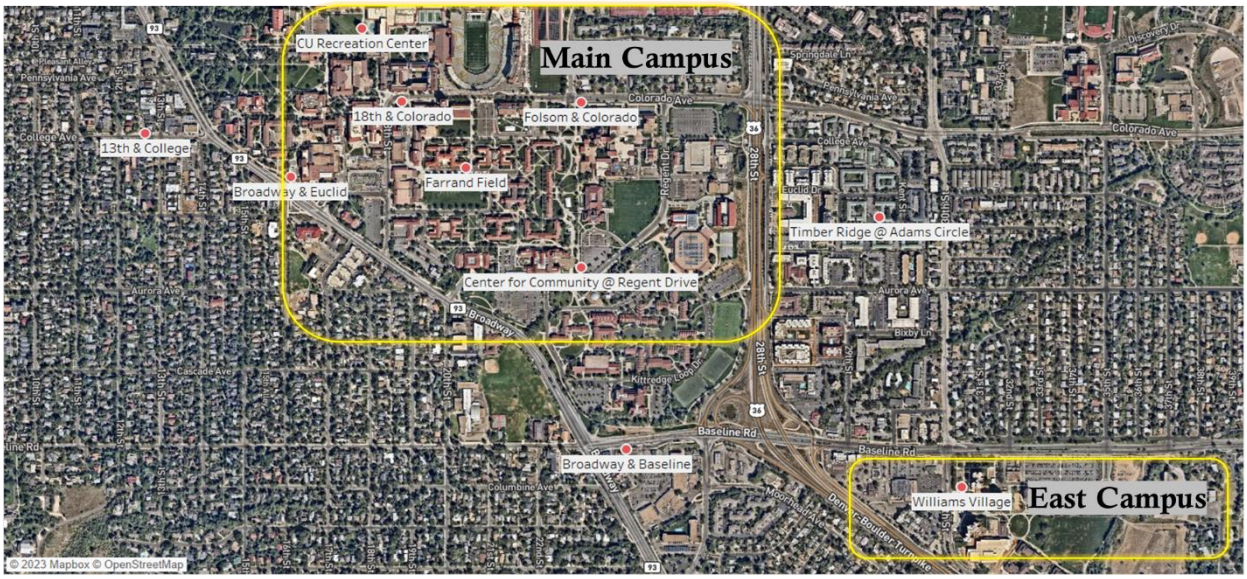

Fig 4: Location of popular BCycle stations in Boulder

Busiest Time of The Day

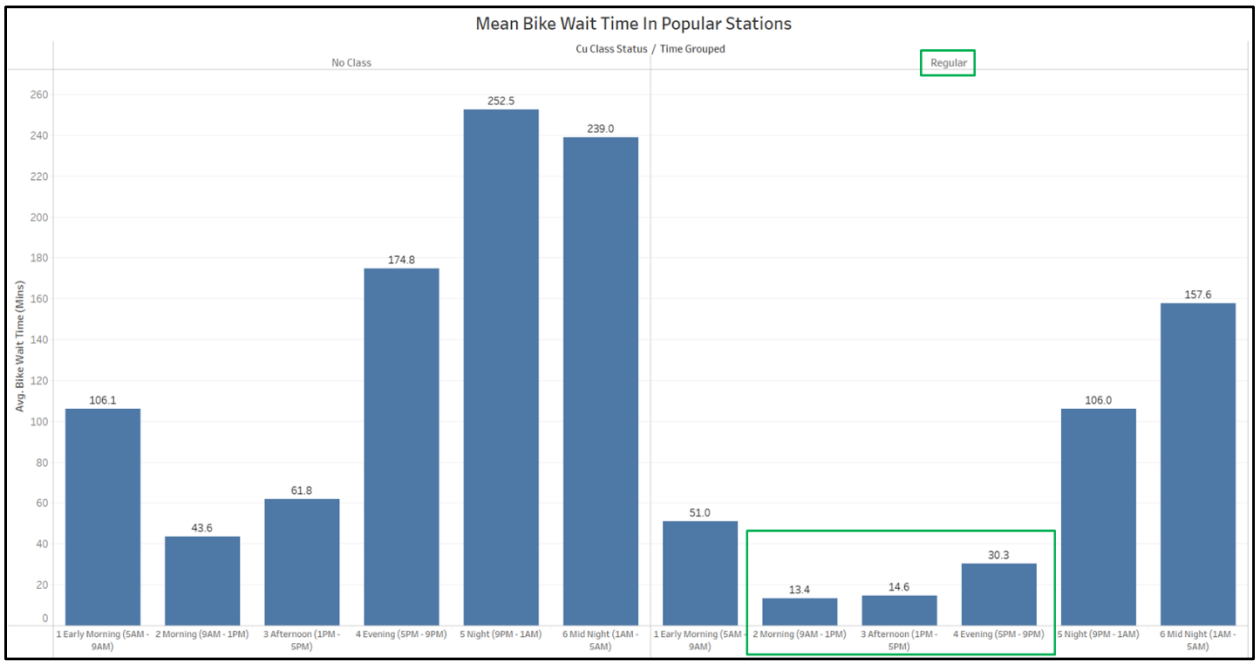

Fig 5: Average Cycle Wait Time Segmented Across Various Times of The Day

BCycle is available for public use around the clock, and it is widely used throughout the day. However, we focused on finding the busiest period of the day across the 10 most popular stations. This analysis proved valuable in allowing us to segment and compare our results specifically during the busiest period along with the overall usage throughout the day.

Fig 4 conveys the average cycle wait time divided along different times of the day (from Early Morning to Mid Night) and CU class schedule categories (Regular classes and No Class days).

The wait times have a considerable variance, and the time of the day has a high influence on it. From Fig 5, it is evident that the busiest period occurs between morning to evening, particularly during regular CU classes. This insight aligns logically because we are focusing on the station located near the CU campus and hence the stations are busy when the classes were happening.

4.3. Feature Engineering:

Normalization

Some columns have low and high values, which will affect the model due to its magnitude. So, here we have solved this with Minmax Normalization, which brings all the values within 0 to 1. The three columns are as follows:

- temperature_2m
- precipitation_probability
- visibility

Bucketing the wait time

We have categorized the wait times into buckets to establish unique output classes.

Wait Time	Category
Less than 20 minutes	Very Low
Between 20 and 40 minutes	Low
Between 40 and 60 minutes	High
More than 60 minutes	Very High

Label Encoding

Since many machine learning algorithms work effectively with numerical data, we have converted all the categorical features into numerical form using LabelEncoder(). This helps the models to extract the frequency patterns. The Three columns which we have converted using LabelEncoder() are as follows:

Label Encoded Features
station_id
cu_class_status
day_of_week_rnd

4.4. Modelling

Our data is more imbalanced and has more features to process. We need a model that can be robust to fetch temporal patterns from these complex datasets and provide accurate results [4] [5]. Moreover, with only one month of data in hand completing this classification task is only possible by using Random Forest, XGBoost, and SVM based on researching their advantages and performance in the classification task.

Train test split

The dataset contains the GBFS data collected for November and the first seven days of December. We have collected the weather from an open-source API for the same period. Now for training, we have utilized the November month data and we have tested the model's performance with December month's data.

This temporal separation for training and testing the model allows us to see the predictive capability on unseen periods. We have utilized this method rather than the conventional 80:20 split as these splits will get more accurate results.

Random Forest

This is an ensemble model, which works by having multiple decision trees that have a random sample of features pushed into them. This idea of having individual decision trees helps in the reduction of overfitting of the complex relationships in the model. Furthermore, the final decision for a classification task is by voting towards the majority.

Hyper Parameter Tuning

The best parameters for the Random Forest model are found by the GridSearchCV techniques which search for the best parameters from the grid of parameters with multiple combinations by applying cross-validation to each combination. The best parameters for the Random Forest Model are as follows.

Parameter	Value
bootstrap	True
max_depth	10
max_features	4
min_samples_split	10
n_estimators	200

XGBoost

XGBoost is an acronym for extreme Gradient Boosting ML algorithm. It is similar to a Random Forest which contains multiple decision forests with a boosting technique that sequentially has those individual trees. Indeed, the model successive model corrects the errors by the former. In the end, it employs regularization techniques to avoid the overfitting on unseen data.

Hyper Parameter Tuning

The best parameters for the XGBoost model are found by the GridSearchCV techniques which search the best parameters from the grid of parameters with multiple combinations by applying cross-validation to each combination. The best parameters for the Random Forest Model are as follows.

Parameter	Value
max_depth	3
min_child_weight	5
n_estimators	100

SVM

It works by finding the best hyperplane which separates both the classes in the high-dimensional space. It

has three different kernels: Linear, Polynomial, and Radial Basis Functions. SVM has regularization parameters that prevent the model from overfitting.

Hyper Parameter Tuning

The best parameters for the SVM model are found by the GridSearchCV techniques which searches for the best parameters from the grid of parameters with multiple combinations by applying cross-validation to each combination. The best parameters for the Random Forest Model are as follows.

Parameter	Value
C	0.1
gamma	1
kernel	RBF

5. Result and Conclusion

After performing hyperparameter tuning and cross-validation using the mentioned three classification models, we achieved a maximum accuracy of 66% using the Random Forest Classifier on testing with the December data.

In our use case, we are particularly interested in the model's performance during the busiest times of the day. Hence, accuracy was checked for the busiest time, between morning to evening during regular class hours as we discussed in the Exploratory Data Analysis section, which was significantly high as 75%. The reason for this increase is that the variance of wait times was low during the busiest times and hence the model was able to capture the trends accurately.

Moreover, we also noticed that the wait time variance was different among different stations, hence we also split the 75% accuracy by stations which is displayed in Fig 6.

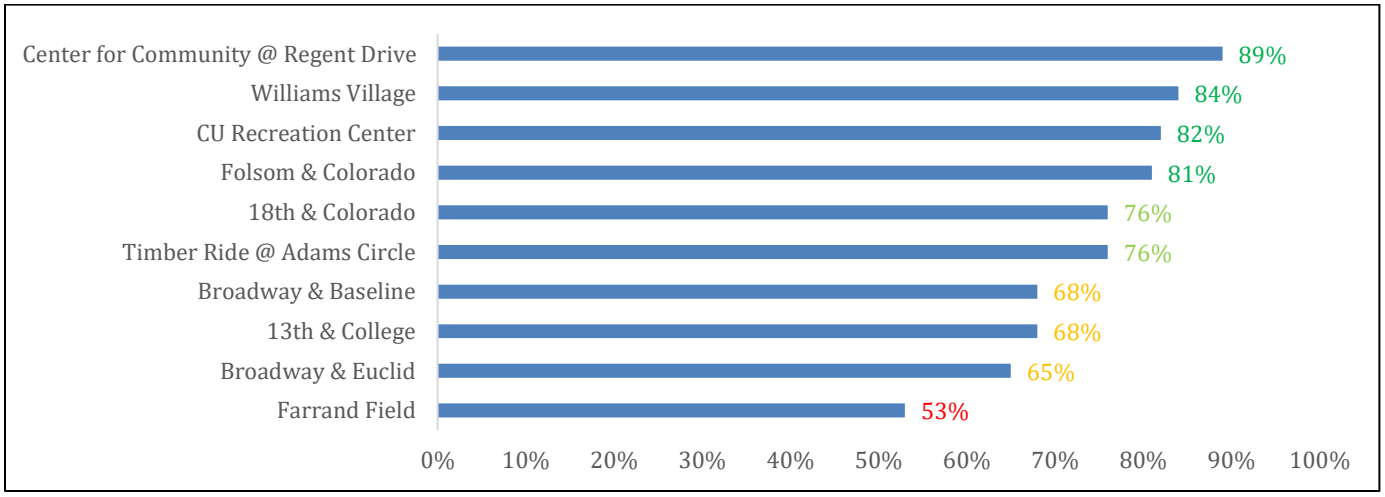

Fig 6: Model accuracy split by stations.

Notice that few stations outperformed with an accuracy of more than 80%, with the highest in “Center for Community” with 89%. This indicates the volatility of wait times across stations and the need for more data for the model to learn these hidden patterns.

However, the accuracy could be used to measure the overall performance of the model, we are keen to measure the performance of the “Very Low” category as it is crucial in our use case. The below table identifies the precision, accuracy, and F1 score of the “Very Low” category.

Class	Evaluation Metrics		
	Precision	Recall	F1 Score
Very Low	0.66	0.79	0.72
Low	0.19	0.05	0.07
High	0.25	0.01	0.02
Very High	0.58	0.80	0.67

From this, we conclude that our model can be used for predicting wait times at the top 10 popular stations, and it accurately predicts the “Very Low” category compared to other categories, particularly during the busiest time of the day.

6. Future Scope

Currently, we predict the likelihood of getting a new bike in under 20 minutes or higher but with more historical data we can try to estimate the exact wait time using the transformers which offer the potential for regression using complex temporal patterns such as seasonality and weather conditions and other factors contributing the model's accuracy. Moreover, we are trying to increase our dataset, so that we can develop a system to predict the exact wait time and facilitate this feature for all 54 stations in Boulder.

7. References

[1] Rastpour, A., & McGregor, C. (2022). Predicting Patient Wait Times by Using Highly De-identified Data in Mental Health Care: Enhanced Machine Learning Approach. JMIR Mental Health, 9(8). https://doi.org/10.2196/38428

[2] Erjie Ang, Sara Kwasnick, Mohsen Bayati, Erica L. Plambeck, Michael Aratow (2015) Accurate Emergency Department Wait Time Prediction. Manufacturing & Service Operations Management 18(1):141-156.https://doi.org/10.1287/msom.2015.0560

[3] Sun, Y., Teow, K. L., Heng, B. H., Ooi, C. K., & Tay, S. Y. (2012). Real-Time Prediction of Waiting Time in the Emergency Department, Using Quantile Regression. Annals of Emergency Medicine, 60(3), 299-308. https://doi.org/10.1016/j.annemergmed.2012.03.011

[4] Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for Multi-Class Classification: An Overview. ArXiv. /abs/2008.05756

[5] Har-Peled, S., Roth, D., Zimak, D. (2002). Constraint Classification: A New Approach to Multiclass Classification. In: Cesa-Bianchi, N., Numao, M., Reischuk, R. (eds) Algorithmic Learning Theory. ALT 2002. Lecture Notes in Computer Science(), vol 2533. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-36169-3_29

YouTube Presentation Link: https://youtu.be/QEBMTk4OsAE?si=6ka1qolqECUJ4IHm