Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-020-02848-5

ORIGINAL RESEARCH

An enhanced ride sharing model based on human characteristics, machine learning recommender system, and user threshold time

Husnu S. Narman[1] · Haroon Malik[1] · Govind Yatnalkar[1]

Received: 15 July 2020 / Accepted: 5 November 2020

Abstract
The ubiquitous availability of the Internet and advanced computing systems has resulted in the rapid development of smart cities. From connected devices to live vehicle tracking, technology is taking the field of transportation to a new level. An essential part of the transportation domain in mart cities is to share vehicles. Sharing vehicles is an impeccable solution to issues like vehicle congestion, pollution, and the rapid consumption of fuel. Even though carpooling has several benefits, currently, the usage is significantly low due to social barriers, long rider waiting time, and unfair pricing models. Considering these issues, we have designed an enhanced vehicle-sharing model with two matching layers. The first layer matches riders based on similar characteristics, and the second layer provides matching options to riders and drivers to restrict the waiting time by using personalized threshold time. At the end of trips, feedback is collected from users according to five characteristics. Then, the two main characteristics that are the most important to riders are determined based on the collected feedback. The characteristics and classifiers are fed to our machine-learning classification module. For new users, the module predicts riders' characteristics, which allows riders to be matched to riders with similar characteristics. We have carried out an extensive simulation and measured the efficiency of the matching model while comparing the results with and without machine learning algorithms. In the simulation, we have used real-time New York City Cab traffic data with real-traffic conditions by using Google Maps APIs. Results indicate that the proposed model is feasible and efficient as the number of riders increases while maintaining threshold time for riders. Our proposed model and obtained results will help service providers to increase the usage of carpooling, and implicitly preserve natural resources and improve environmental conditions.

Keywords Carpooling · Machine learning · Characteristics · User feedback system · User threshold time (UTT) · Recommendation systems

1 Introduction

Vehicle sharing is the process of completing a journey by following a particular trajectory based on multiple rider locations. The field of smart transportation is becoming more developed, accessible, and is indeed a positive solution to several issues in the transportation domain (Carrese et al. 2017; Mallus et al. 2017; Xu and Zhou 2020; Streitz 2019). One of the primary issues in this domain is the pollution resulting due to emissions by many vehicles (De Lira et al. 2018). It has been shown that there is a relation between the number of vehicles and the population (Wang 2019). As the population increases, the number of vehicles and emissions increases, which results in global warming (Wang 2019; Apte et al. 2017). The increase in the number of vehicles also leads to traffic congestion, car accidents, and a minor but key issue, such as reducing parking spaces (Carrese et al. 2017; Apte et al. 2017).

Ride-sharing is a practical solution for the previously mentioned problems if applied effectively (Huang et al. 2014). For example, if five users who have self-purchased vehicles decide to carpool, they cut down the usage of four cars. This implies reducing the fuel usage and carbon footprint by almost 80% and an overall reduction in traffic (Apte et al. 2017; Contreras and Paz 2018). Additional advantages

✉ Husnu S. Narman
narman@marshall.edu

Haroon Malik
malikh@marshall.edu

Govind Yatnalkar
yatnalkar@marshall.edu

[1] One John Marshall Dr, WAEC 3107, 25755 Huntington, WV, USA

Published online: 28 January 2021

can be decreasing the stress, fatigue, and fares among riders, increasing parking spots; and encouraging social interactions with others during the journey (Carrese et al. 2017; Teubner and Flath 2015; Duan et al. 2018; Contreras and Paz 2018).

Despite copious benefits, ride-sharing is discouraged due to social barriers since riders do not possess any knowledge of other riders they will be commuting with (Wang et al. 2019; Boldrini et al. 2016; Kim 2012). An additional issue is the sudden elongation of trip time due to the unexpected addition of riders (Wang et al. 2019). This ramifies in rider frustrations, disputes and even results in avoiding the usage of carpooling (Cramer and Krueger 2016; Zhao and Su 2019).

Our *aim* in this paper is to implement an enhanced vehicle sharing model that addresses the mentioned issues related to unknown characteristics of riders and the sudden elongation of the trip time.

The *main objective* of the proposed research works is to enhance vehicle sharing models by using feedback and the User Tolerated Time or User Threshold Time (UTT). A generic flow of our designed system is reflected in Fig. 1. The system's first task is rider registration or noting the basic profile data with five human characteristics on a scale of 1–5. These characteristics are safety, punctuality, chattiness, comfortability, and friendliness. Then, UTT is used to create an agreement on waiting time between riders. UTT is the time in minutes that riders are willing to spend picking up other riders during the trip. For example, assume that a driver and the driver's first passenger know that reaching from the source of the first passenger to destination 25 min, and both driver and the first passenger agree on 15 min UTT. In this scenario, the driver can spend at most extra 15 min to pick up a second or more passengers as long as the driver does not violate other passengers' agreed UTT time. Thus, the first passenger should arrive at the destination at most 40 min. The amount of time preferred can vary, but we have taken UTT as a scale of 10–30 and in multiples of 5 in this

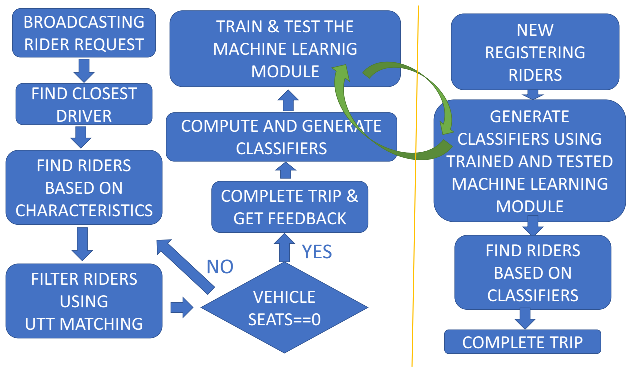

BROADCASTING RIDER REQUEST
FIND CLOSEST DRIVER
FIND RIDERS BASED ON CHARACTERISTICS
FILTER RIDERS USING UTT MATCHING
TRAIN & TEST THE MACHINE LEARNIG MODULE
COMPUTE AND GENERATE CLASSIFIERS
COMPLETE TRIP & GET FEEDBACK
YES
NO
VEHICLE SEATS==0
NEW REGISTERING RIDERS
GENERATE CLASSIFIERS USING TRAINED AND TESTED MACHINE LEARNING MODULE
FIND RIDERS BASED ON CLASSIFIERS
COMPLETE TRIP

Fig. 1 A generic flow of rider matching layers with the role of the machine learning model

paper. Riders are only accepted if they are at a traveling time less than or equal to registered UTT. Therefore, riders are first searched with similar characteristics and then filtered based on UTT. If riders satisfy the matching layer conditions, they are added to the trip itinerary. This point marks the completion of trip formation and is followed by our newly orchestrated novel feedback system where riders rate the driver as well as other riders in the trip. Feedback comprises rating the five characteristics on a scale of 0–5. This feedback is an important data-set as we use this data to compute two classifiers for every user. These classifiers are later utilized to provide better matches and train the machine learning module.

There are two classifiers; "Feedback-Given-Classifier" and "Feedback-Received-Classifier." Feedback-Given-Classifier is derived based on the feedback the rider gives to other riders, while Feedback-Received-Classifier is derived on the feedback data-set the rider gets from other riders. These classifiers are used to determine the characteristics of riders based on the five characteristics. In the end, every rider is associated with two main characteristics of riders. Riders with similar characteristics, but especially these two main characteristics, are recommended to each other for future trips. In order to predict the main characteristics of riders, machine learning algorithms are used. Machine learning is a captivating technology where a system learns and trains based on an existing data-set and predicts outputs for new input data (Campana et al. 2016; Depari et al. 2019). In our case, we are using the Support Vector Machines for the classification. After appropriate training and testing, the module predicts the characteristics of registering riders. Riders are recommended based on these predicted characteristics.

The *key contributions* of the paper are: (i) Implementation of vehicle sharing model, (ii) Performing the matching by using characteristics of riders and then filtering riders based on UTT, (iii) Predicting characteristics for newly registering riders based on "Feedback-Given-Classifier" and "Feedback-Received-Classifier", and (iv) Evaluating the proposed model with real data to analyze the performance of the model. The observations and results show that the proposed model is feasible and can be deployed to increase the usage of vehicle sharing usage.

The rest of the paper is organized as follows: The previous works are described in Sect. 2. In Sect. 3, we discuss the problem statement and the system architecture. The proposed model and simulations are explained in Sect. 4. Section 5 has our results and observations, and Sect. 6 has the concluding remarks and a plan to improve the proposed model.

2 Related work

In this section, we explain the current applications and previous studies on vehicle sharing. We first explain the current applications of vehicle sharing. Then, we discuss methodologies for classifier computations. We also explore different machine learning classification models.

2.1 Previous vehicle sharing applications and their limitations

We performed a deep inspection of the current popular ride-sharing applications like UberPool, LyftLine, Juno, Curb, Wingz, Via, Flywheel, Zimride, and Waze (Li et al. 2016; Rodriguez 2019; Shaheen and Cohen 2019; He et al. 2018). Studying the applications provided a basic idea of carpooling. Uber, Lyft, Wingz, Via are vehicle-sharing applications that allow any person to be a rider or driver (Li et al. 2016; Rodriguez 2019). Role restrictions are observed in Juno, Gett, and Curb as they are taxi based ride-sharing services (Wang 2019).

The strong point of most of the applications is the usage of carpooling with modern technologies like IoT and Cloud Computing, which facilitates greater availability, scalability, load-balancing of requests, and compelling notification abilities (Mallus et al. 2017; Huang et al. 2014; He et al. 2014). Some of the common limitations and reasons for disputes observed in all the applications can be listed as follows: (i) Drivers get to know the count of passengers at the pickup point (Zhao and Su 2019). (ii) In most of the trips, the vehicle has only one passenger leaving the pool incomplete (Mallus et al. 2017; Wang 2019). (iii) Passengers do not have basic details of other passengers they are traveling with. (iv) Unfair pricing is used for customers (Cramer and Krueger 2016). (v) The sudden addition of riders whose destination is too far, which adds a significant amount of time to trip completion. All these limitations show that the addition of a rider should be based on traveling time and not distance (Mallus et al. 2017; Cramer and Krueger 2016; Boldrini et al. 2016).

Noting the limitations in applications, we have designed our model to eliminate most of the problems. We first perform the exact match, then perform a "closer" match if the exact match does not exist, and then a match irrespective of characteristics if the first two are not available. The types of matching are portrayed in Fig. 2. The exact match finds the riders with exactly matching characteristics. If the pool is incomplete, we find riders with little different or "closer" characteristics. The process of matching is described in Sect. 4. If the pool is still incomplete, we incorporate the current Uber model of matching for riders irrespective of characteristics, i.e., matching riders based on the closest distance (Uber, 2019). Initially, we search riders from the same vicinity. If the rider list is exhausted, we increase the radius of our search and traverse through more riders. This methodology ensures we capture most of the broadcasting riders and completes the pool for a maximum number of trips. The rider with the mark 'B' in Fig. 2 is the broadcasting rider.

After having every passengers' details, we provide the trip itinerary to every rider, including the driver, before commencing the trip. With the UTT matching layer and shortest path multi-source and multi-destination model, we ensure the accepted riders in the trip are not at a location above the restricted traveling time. We will be implementing a sophisticated pricing model in the future, but our current pricing model includes billing every user for the initially planned trip agreement between the driver and the passenger, and not for the entire trip. We deployed our Python code in the Google Cloud and used the Atlas MongoDB Cloud database to exploit the benefits of Cloud Computing (He et al. 2014).

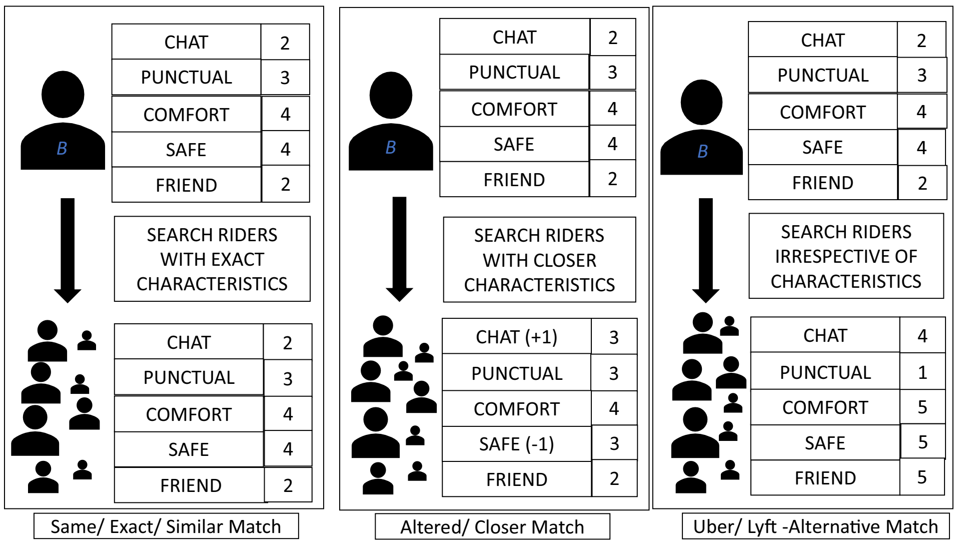


Fig. 2 Types of Rider Matching

By researching limitations, we learned it is necessary to reach user expectations and improvise the user experience as much as possible (Boldrini et al. 2016; Wang et al. 2019).

2.2 Tracking rider characteristics

One of our system's main motivations is to track the characteristic the rider focuses on the most while rating other riders. The method for tracking the data characteristic depends on our feedback records. A user may provide ratings to a few characteristics for most of the trips. It may also be the case that the user rates some characteristics with high scores like 4 or low scores like 0 to a specific characteristic for several trips, implying users are less interested in that characteristic. We target to find the characteristics the user is most interested in and recommend riders based on the tracked characteristics.

Through our research work, we found statistical methods like finding the mean and range of the numbers, standard deviation, and variance (Huang and Peng 2018; Fang et al. 2018; Ahn and Fessler 2003). The methodology we selected can be explained by considering the example of lists L_1, L_2 and L_3.

$$L_1 = [1, 0, 5, 4, 0]$$
$$L_2 = [0, 0, 0, 0, 2]$$
$$L_3 = [4, 4, 4, 4, 4]$$

N gives the total number of sample points in a list. The mean of the sample set is denoted by x_i. For calculating standard deviation and variance, we need the mean x_i and the range of the sample set. Standard deviation calculates how normally distributed the data is while variance computes how far or spread each sample point is from the mean (Ahn and Fessler 2003). The distance of data-point to the mean or the level of spread of a specific sample point, x is computed by using Eq. (1) (Fang et al. 2018; Ahn and Fessler 2003).

$$x_{distance} = x - x_i \tag{1}$$

We exactly found what we are looking for in the variance, σ^2 as stated in Eq. (2). The general definition of variance is the average of squared differences from the mean(Fang et al. 2018; Ahn and Fessler 2003). The squared differences are squaring the$x_distance$ as it may result in negative values because of sample points being less than mean, x_i (Ahn and Fessler 2003). Variance provides the spread of data within a specific range, and the larger the spread, the larger is the data variety (Huang and Peng 2018; Fang et al. 2018; Ahn and Fessler 2003).

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x - x_i)^2}{N} \tag{2}$$

If the variance is applied to all three lists, it is the highest at L_1. The spread of sample data in L_2 and L_3 around the mean is low, or albeit, the data variety is low (Ahn and Fessler 2003). If a similar methodology is applied for every characteristic feedback a rider rates, a score with the highest variance is the characteristic the rider focuses the most. After performing several experiments and proof of concepts with our feedback data, we employed variance in classifiers' computation.

2.3 Machine learning module

The closer matching of riders consisted of manually altering characteristics of broadcasting riders like adding or subtracting by 1 and then re-searching riders. Indeed, we wanted to automate this process. For finding a solution, we turned our research towards machine learning and stumbled upon the machine learning content-based recommendation system (Luo et al. 2018). In this system, the features are converted to vectors and represented in d-dimensional space, where d is the number of features (Luo et al. 2018; Dehak et al. 2010). The angular distance or the cosine of the angle, θ between the vectors is calculated using the dot product equation (Dehak et al. 2010; Nguyen and Bai 2010). Vectors with the highest cosine values are closer and are deemed as the best match (Luo et al. 2018; Dehak et al. 2010; Nguyen and Bai 2010). We made a similar use where the features represented the registered rider characteristics with UTT, and riders with higher cosine similarities are paired up on a trip.

Additionally, if we wanted to make use of machine learning for predictions, there is room for a little error due to the presence of an imbalanced training feedback data-set (Tang et al. 2008). The inputs may be repeated for different outputs in case of an imbalanced data-set (Sáez et al. 2016). Another goal of machine learning in our system is to predict classifiers for newly registering riders after the training and testing of our machine learning module is complete.

In our pilot study, we tested many classification models like Logistic Classification, KNN Classifiers, Naive Bayes Multinomial model, Random Forests model, Neural Networks, and Support Vector Machines (SVM) (Liang et al. 2018; Ye et al. 2018; Wang et al. 2019; Shahane et al. 2019; Jiang et al. 2019). Out of all classification models, SVM is found to be most feasible because of the Radial Bias Function (RBF) Kernel (Han et al. 2012). The RBF kernel is a highly non-linear curve that is used for distinguishing classes (Jiang et al. 2019; Ma et al. 2017; Han et al. 2012). SVMs work on the principle of placing the line or curve to the closest data point with maximum distance (Jiang et al. 2019). The classification is changed through the regularization parameter, C, and the gamma parameter, γ (Han et al. 2012). This process of governing the curve placement is called Kernelization (Ma et al. 2017; Han et al. 2012; Jiang et al. 2018). The regularization allows a class to include fewer data points of other classes and is best suited for imbalanced data-sets (Tang et al. 2008; Han et al. 2012; Jiang et al. 2018).

3 System architecture

In this section, we start with the problem statement and conclude by a brief description of the architecture.

3.1 Problem statement

The increased number of vehicles has led to significant problems like global warming, rapid consumption of fuel, and crucial traffic issues (Carrese et al. 2017; Mallus et al. 2017; Apte et al. 2017). With humans, this also affects other living beings on our planet (Swami 2018). To overcome these problems, we provide a novel platform for vehicle sharing that includes rider matching based on human characteristics and restricted travel time. The model also uses machine learning to predict better rider recommendations and tune up the system efficiency. We hope to cope with the stated issues and tend to improve the weather conditions in the future by providing social vehicle sharing.

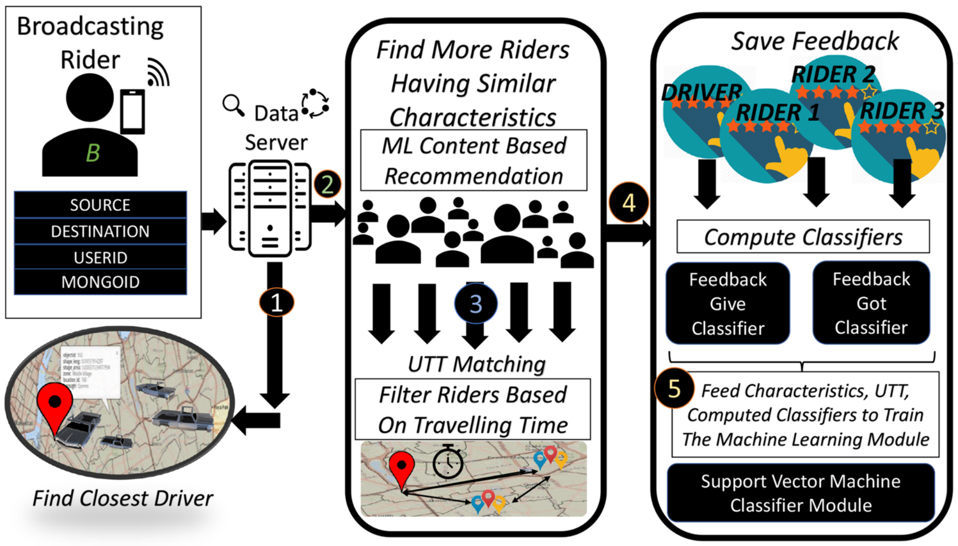
3.2 Architecture

The key parts of the architecture are the broadcasting rider request, the driver, matching layers, the feedback system, and the machine learning module. Figure 3 reflects the system architecture. The broadcasting rider request consists of rider source, destination, and the rider user-id. Using the user-id, the registered UTT and five characteristics are retrieved from the data server and are referenced throughout the trip while searching for more riders.

For best simulation practices, we made the use of the New York Cab location database (NYC 2019). The NYC Cab locations are divided into 263 zones. These zones are highly useful to avoid a bigger search, i.e., searching the entire New York State for exact or closer riders. Using the broadcasting user source zone, the closest available driver from the same zone is retrieved. This marks the completion of Step 1 in the architecture.

After the driver is allocated to a trip, rider matching commences, divided into two steps. Step 2 is the characteristics matching layer. In Step 2, all the broadcasting riders from the same source zone are retrieved with exactly matching or closer characteristics. After a rider is matched and accepted, the traveling time between the broadcasting and accepted rider locations is checked in Step 3 or the UTT matching layer. If the riders are accepted in both layers, they are added to the trip. Step 2 and 3 continue until the vehicle seats are filled, or no riders are left to traverse. If the seating capacity is reached, the trip's pool completion status is assigned as 'Yes' else it is labeled as 'No'.

Step 4 is saving the feedback and computing two classifiers for every user. Riders are classified into two classes, and these classes are referred for rider recommendations in future trips. Step 5 is training and testing the machine learning model. The input to the machine learning model is the registered characteristics and UTT, and the outputs are the classifiers. For newly registering riders, the machine learning module predicts classifiers and provides better rider recommendations. The phase of machine learning is the final step of our architecture.


Broadcasting Rider
B
SOURCE
DESTINATION
USERID
MONGOID
Data Server
1
2
Find Closest Driver
Find More Riders Having Similar Characteristics
ML Content Based Recommendation
3
UTT Matching
Filter Riders Based On Travelling Time
4
Save Feedback
DRIVER
RIDER 1
RIDER 2
RIDER 3
Compute Classifiers
Feedback Give Classifier
Feedback Got Classifier
5
Feed Characteristics, UTT, Computed Classifiers to Train The Machine Learning Module
Support Vector Machine Classifier Module
Fig. 3 The System Architecture

4 Proposed model

This section focuses on our contributions and implementations. We sub-categorize this Section as 4.1 Driver Search, 4.2 Characteristics & UTT Matching Layers, 4.3 Computation of Classifiers, 4.4 Machine Learning Model & Prediction, and 4.5 Experimentation.

4.1 Driver search

Along with the user-id, source, and destination, the broadcasting rider request also possesses the source and destination zones. Based on the source zone, all the available drivers are retrieved. The traveling time including real-time traffic is obtained using the Google Map API, and the closest driver is selected. An important step in the driver search is noting the vehicle seating capacity. The complete driver search using Google Map API is shown in Fig. 4.

4.2 Characteristics and UTT matching layers

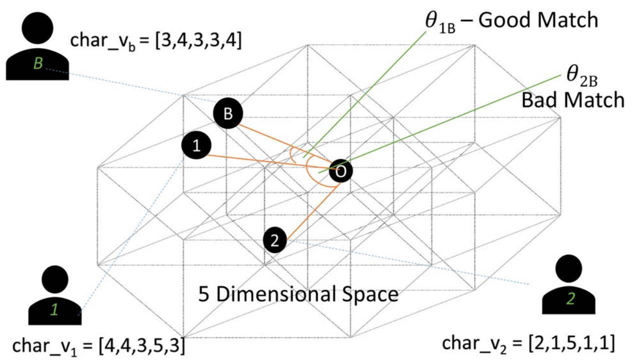
The characteristics and UTT layers are the most vital parts of our system. Again, based on the broadcasting rider's source zone, riders with exactly matching characteristics are fetched first. The odds of finding an exact match in the same zone are low because two riders at the same time should start and reach the same or nearby sources and destinations. Instead, there are many chances that riders may have little different characteristics and are heading on the same trajectory. For these purposes, we used the concept of machine learning content-based recommendations. Initially, for every $rider_a$, his/her registered characteristics are converted to a vector, $char\_v_a$ as shown in Eq. (3).

$$char\_v_a = [chat_a, safe_a, punctual_a, friend_a, comfort_a] \tag{3}$$

For example, let broadcasting rider, $rider_b$, characteristics be chatty:3, safety: 4, punctuality:3, friendliness:3, and comfortability:4. The broadcasting rider vector representation, $char\_v_b$ is given by Eq. (4).

$$char\_v_b = [3, 4, 3, 3, 4] \tag{4}$$

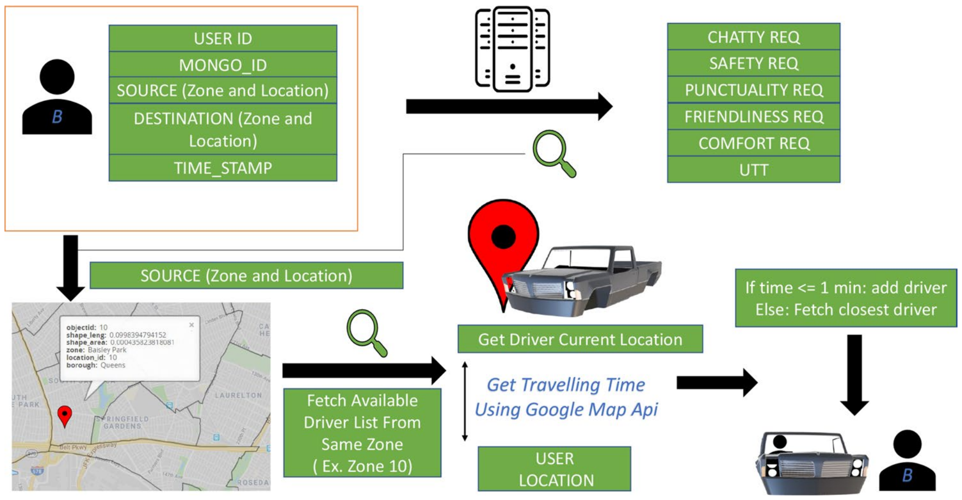
As there are 5 features or characteristics, the vectors are represented in a 5d or a 5-dimensional hypercube, as shown in Fig. 5. 'O' represents the origin and 'B', '1', and '2' represent the points plotted by the vectors for broadcasting rider and other riders to be matched. For measuring the level of match, the angular distance between two vectors or the cosine of angle θ_{ab} is calculated using the Eq. (5). The cosine of θ_{ab} is equal to the dot product of vector divided by the product of the vector magnitude.



B
$char\_v_b = [3,4,3,3,4]$
θ_{1B} – Good Match
θ_{2B}
Bad Match
B
1
O
2
5 Dimensional Space
1
$char\_v_1 = [4,4,3,5,3]$
2
$char\_v_2 = [2,1,5,1,1]$

Fig. 5 Rider matching using content-based recommendation

Fig. 4 Fetching the closest driver



B
USER ID
MONGO_ID
SOURCE (Zone and Location)
DESTINATION (Zone and Location)
TIME_STAMP
CHATTY REQ
SAFETY REQ
PUNCTUALITY REQ
FRIENDLINESS REQ
COMFORT REQ
UTT
SOURCE (Zone and Location)
Get Driver Current Location
If time <= 1 min: add driver
Else: Fetch closest driver
Fetch Available Driver List From Same Zone (Ex. Zone 10)
Get Travelling Time Using Google Map Api
USER LOCATION
B

objectid: 10
shape_leng: 0.0998394794152
shape_area: 0.000435823818081
zone: Baisley Park
location_id: 10
borough: Queens

$$cos\theta_{ab} = \frac{(char\_v_a \cdot char\_v_b)}{\|char\_v_a\|\|char\_v_b\|} \tag{5}$$

For θ_{ab} equal to 0, $cos\theta_{ab}$ is 1. This is the case of rider matching by "exact" characteristics. If there are other riders with the same characteristics as the broadcasting rider, it is a 100% match. Hence, a greater cosine value means a greater match. In Fig. 5 $char\_v_1$ seems to be a better match than $char\_v_2$ due to a smaller angle θ_{1B}. Therefore, in this example, $Rider_1$ will be selected, and $Rider_2$ will be directed to other broadcasting requests. In simulations, we accept riders only with a match of greater than 85% or if the $cos\theta_{ab}$ is above 0.85.

Accepted riders are sent for the UTT check, the second matching layer, as shown in Fig. 6. At first, the traveling time between broadcasting and accepted rider's source locations is noted using the Google Map Distance Matrix API. It is checked if the traveling time is equal to or less than the trip UTT. If the condition is satisfied, the traveling time between broadcasting and accepted rider's destination locations is checked if it is equal or less than the UTT. If both conditions satisfy, the accepted rider is added onto the trip. This process continues until the riders reach the vehicle's seating capacity, or no riders are left for matching.

We have maintained a threshold for trip formation time of 2 minutes, which assures the time for trip formations is not time-consuming. If the rider list is exhausted, we continue the same process with other zones instead of rider source zones. The main motive of the matching layers is subjecting users to commute with people having similar characteristics and complete the journey in limited time constraints.

4.3 Computation of classifiers

The phase of classifiers computation takes place after saving the rider feedback and dividing the feedback data into two parts. The first classifier is called the "Feedback-Given-Classifier" and uses the first part of the feedback data, which includes the ratings given by each rider to other riders. The purpose of this classifier is to track the characteristic the rider most focuses on while giving feedback. The process can be explained with an example of the feedback table created using the feedback given by $Rider_1$ to other riders, as shown in Table 1.

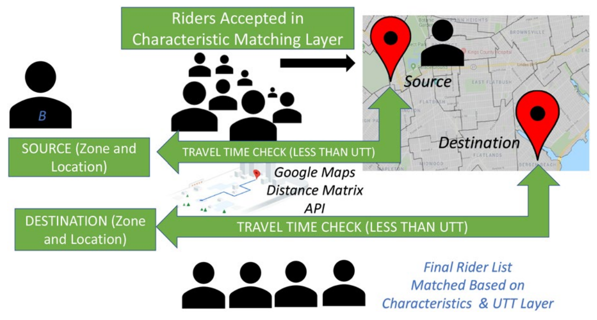

Riders Accepted in Characteristic Matching Layer
Source
B
SOURCE (Zone and Location)
TRAVEL TIME CHECK (LESS THAN UTT)
Destination
Google Maps Distance Matrix API
DESTINATION (Zone and Location)
TRAVEL TIME CHECK (LESS THAN UTT)
Final Rider List Matched Based on Characteristics & UTT Layer

Fig. 6 User Threshold Time (UTT) matching layer

The data given by the rider is segregated characteristic wise and appended in new lists. The following lists are created based on the feedback given by $Rider_1$.

$$\begin{aligned} chat_{Rider1} &= [0, 0, 1] \\ safe_{Rider1} &= [2, 3, 5] \\ punctual_{Rider1} &= [1, 0, 0] \\ friend_{Rider1} &= [4, 4, 4] \\ comfort_{Rider1} &= [0, 0, 0] \end{aligned}$$

The observation made from the five lists is that $Rider_1$ may continue to give a friend rating of 4 or comfort or a punctual rating of 0 in future trips. The only data variety observed was in the safety rating. Thus, the $Rider_1$ "Feedback-Given-Classifier" is the safety class. This computation is done using the variance equation. The higher the spread of the data around the mean of a characteristic list, the higher is the variance of that list. Total number of elements in a characteristic list is n_{char} or $data\_count_{char}$. The mean is denoted by x_{char_i}, and the squared differences are calculated by Eq. (6).

$$x_{char\_distance} = (x - x_{char\_i})^2 \tag{6}$$

The selected characteristic variance is denoted by σ^2_{char} and is represented in Eq. (7). The characteristic list with the highest variance is selected as this proves the user is more diverse in rating the characteristic and therefore focuses more on the specific characteristic.

$$\sigma^2_{char} = \frac{\sum_{i=1}^{n_{char}} (x - x_{char\_i})^2}{data\_count_{char}} \tag{7}$$

After calculating variance and first classifier, the second classifier or the "Feedback-Received-Classifier" is computed. The second classifier uses the second part of feedback databases, which is the feedback received by other riders to a rider.

Let the feedback given to $Rider_1$ be as shown in the Table 2. Each element in the column has two values. The first value is the feedback given by a $User_i$ for a specific characteristic,

Table 1 Feedback given by $Rider_1$ to other riders

Riders	Chat	Safe	Punctual	Friend	Comfort
$Rider_2$	0	2	1	4	0
$Rider_3$	0	3	0	4	0
$Rider_4$	1	5	0	4	0

Table 2 Feedback given to $Rider_1$ by other riders which are the product of feedback given value and the respective $Rider_i$ characteristic variance

Riders	Chat	Safe	Punctual	Friend	Comfort
$Rider_2$	4*0.32	2*4.31	0*2.10	2*0.1	4*1.73
$Rider_3$	3*3.45	1*0.15	1*0.55	0*5.72	3*3.34
$Rider_4$	3*9.21	0*3.21	3*0.02	0*0.21	0*1.32
$\sum$ Total	39.26	8.77	0.61	0.2	16.92

and the second value is the characteristic variance ($(\sigma_{i\_char})^2$) computed for the $User_i$ characteristics. Every time a rider provides feedback, the feedback value is multiplied by their characteristic variance. To exemplify, $Rider_2$ variance for safety is 4.31, and the safety rating is 2. The feedback to $Rider_1$ for safety is the product of both these values. This is done for every characteristic and computed for every rider.

In the end, for every characteristic, all the performed multiplications are added and compared. The characteristic value with the highest score is the "Feedback-Received-Classifier". In the following example, the classifier is Chatty, as the value is highest, which is 39.26. After computing the two classifiers, every rider's search criteria are dynamically changed. This is a practical scenario as riders will classify other riders based on their experiences, which assists in better and real-time recommendations to riders.

4.4 Machine learning model and prediction

In fact, we selected Support Vector Machine or SVMs for our machine learning module. We have created two data-sets, namely, "Feedback Given Classifier data-set" and "Feedback Received Classifier data-set". In both data-sets, the input fields are registered user characteristics, registered UTT, and the output is computed classifier. Sample rows in data-sets are shown in Tables 3 and 4.

For both tables, every row contains registered characteristics, UTT, and computed classifier. The inputs or the features selected for the classification model are the characteristics and UTT, and the labeled classes or output is the classifier. We used SVMs for both data-sets and maintained the results in a document every time we trained and tested the SVM. Hence, for two data-sets, we have two distinct SVM modules. One of the SVM outputs, "Feedback-Given-Classifier" and the other outputs "Feedback-Received-Classifier". The working on the SVM modules is reflected in Fig. 7.

We trained both SVMs and tested with the newly registered riders. At first, the newly registered riders provide the characteristics and UTT at registration. These entities are sent as input to both SVMs. Both modules predict classifiers for the newly registered riders. Also, we allowed the riders to complete their first trip and rate other riders for better accuracy. Based on their first trip feedback data, we computed the variance for the new riders. We then re-trained the machine learning module with variance as an additional input feature. In fact, the higher features are for an SVM; the higher is the accuracy of the module. With variance, a new edge is provided to SVM for classifying and plotting the data points into labeled classes. With variance as an additional feature, we got better measurements and higher model accuracy, as showcased in the Results section.

4.5 Experimentation

A simulation model is denoted by Eq. (8) where U_i shows the User Threshold Time, and RC_i shows the number of riders traversed for the i^{th} iteration.

$$S_i = \{U_i, RC_i\} \tag{8}$$

At the beginning of every simulation, a broadcasting rider with UTT equal to U_i is selected. For the 1^{st} iteration, U_1 is taken as 10 minutes and RC_1 as 200. The trip formation is commenced. If the trip is formed within 20 riders, a new trip is commenced, and this process continues till RC_1 reaches

Table 3 Sample rows in Feedback Given Classifier data-set with classifier at the end

Feedback given classifier data-set						
Chat	Safe	Punctual	Friend	Comfort	UTT	Class_Given
3	3	4	1	4	20	Comfortability
1	2	4	3	5	10	Chatty

Table 4 Sample rows in Feedback Received Classifier data-set with classifier at the end

Feedback received classifier data-set						
Chat	Safe	Punctual	Friend	Comfort	UTT	Class_Received
4	2	2	3	1	10	Punctuality
5	4	4	1	4	25	Safety

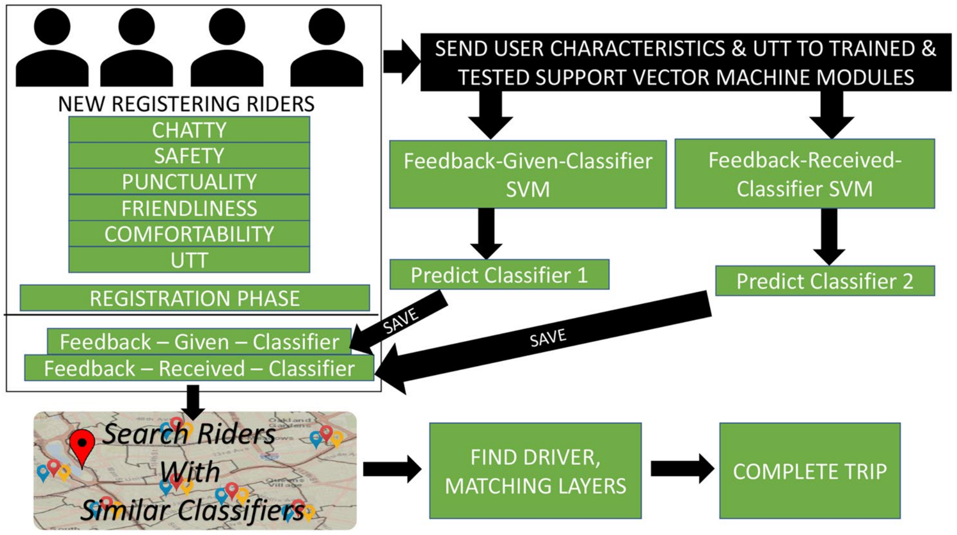

NEW REGISTERING RIDERS
CHATTY
SAFETY
PUNCTUALITY
FRIENDLINESS
COMFORTABILITY
UTT
REGISTRATION PHASE
Feedback – Given – Classifier
Feedback – Received – Classifier
SEND USER CHARACTERISTICS & UTT TO TRAINED & TESTED SUPPORT VECTOR MACHINE MODULES
Feedback-Given-Classifier SVM
Feedback-Received-Classifier SVM
Predict Classifier 1
Predict Classifier 2
SAVE
SAVE
Search Riders With Similar Classifiers
FIND DRIVER, MATCHING LAYERS
COMPLETE TRIP

Fig. 7 Working of Support Vector Machines for predicting classifiers

Table 5 The variables included in keeping a track of resultant values in every simulation S_i

Variable	Description
U_i	Trip User Threshold Time for a simulation S_i
RC_i	Total number of riders traversed
RP_i	Total number of riders accepted
T_i	Total time consumed for completion of a simulation S_i
$trip\_count_i$	Total number of trips computed
MR_i	Matching rate of a simulation S_i

200. The following parameters represent the first simulation model.

$$S_1 = \{U_1, RC_1\} = \{10, 200\}.$$

For every next simulation, the U_i is kept the same, and the RC_i is increased by 200 until it reaches 1000. Hence the next iteration would be $S_2 = \{U_2, RC_2\} = \{10, 400\}$ and followed by simulations till $S_5 = \{U_5, RC_5\} = \{10, 1000\}$. As the RC_i reaches 1000, it is reset to 200, and the U_i is increased by 5, i.e., 15. Hence, the next simulation is denoted by $S_6 = \{U_6, RC_6\} = \{15, 200\}$ and again followed by simulations till $S_{10} = \{U_{10}, RC_{10}\} = \{15, 1000\}$. These simulations are performed until the U_i reaches 30. Indeed, the n^{th} or the last recorded simulation is given by the following simulation parameters.

$$S_n = \{30, 1000\}.$$

For every simulation S_i, we noted RP_i, the total number of riders accepted, T_i, the total time required for completing the simulation, and $trip\_count_i$, the total number of trips computed. The descriptions of these variables are also mentioned in Table 5. We observed, in some cases, the RP_i is 35 and RP_{i+1} is 18. This was a randomness factor introduced due to uneven acceptance of the riders at the UTT matching layer. To reduce this randomness, the same simulation with and without the machine learning model was performed at least ten times. The randomness was reduced by the average of records and provided the scores and system's measurement accurately.

An important measure in the system efficiency is the matching rate. The matching rate is defined by Eq. (9). It is the division of accepted riders and the total number of traversed riders. The matching rate provides an idea of how many riders are accepted out of a stated population. According to our expectations, the matching rate should keep increasing for consecutive simulations.

$$MR_i = \frac{RP_i}{RC_i} \tag{9}$$

Two variables, $closer_i$ and $aletnative_i$ track how many riders are accepted based on exact or closer match, and how many are accepted based on the alternative match. In the end, we add up and save them for checking how riders are matched by type. Eqs. (10) and (11 represent the summed up variable count for all simulations. n marks the total number of simulations and $match_{closer}$ and $match_{alternative}$ reflect the number of matching by type, i.e., if the users were matched closely or irrespective of characteristics.

$$match_{closer} = \sum_{S_i=1}^{n} closer_i \tag{10}$$

$$match_{alternative} = \sum_{S_i=1}^{n} alternative_i \tag{11}$$

The next section of Results describes the contributions of matching rate, number of trips, and time required for trip formation towards the overall system efficiency.

5 Model evaluation and results

At first, we evaluate the machine learning model. As the data is imbalanced, we note the F1 score, precision, and recall for the 5 classes. Initially, we got an overall accuracy of 62% but with the inclusion of variance, we got an accuracy of 90% for both SVMs. The F1 score, recall, precision and confusion matrix tests the comparison between computation and predicted scores. A higher score means the predicted classes match the computed classes for the same set of inputs. The tables for accuracy measures can be stated in Tables 6 and 7.

The confusion matrix provides a two-dimensional result stating how much error was possessed while making predictions. The matrix shows how much the model predicted for each class correctly, also called as the true positive score. If the true positive score is higher for each class, the model behaves as expected. From Figs. 10 and 11, we conclude our true positive scores are high and the model works as expected.

For both SVMs, we started training the machine learning model by 5000 records. We increased the training by 3000 records for the consecutive accuracy test. We meddled with regularization for SVMs for getting maximum accuracy. From Tables 6 and 7, the precision and recall is above 85% which is a good measure for a machine learning model. We ceased further training after 27000 records and testing after 12000 records when we received an overall accuracy of 90% for both the SVM models. Indeed, the prediction is good, and it predicts classifiers based on feedback-records.

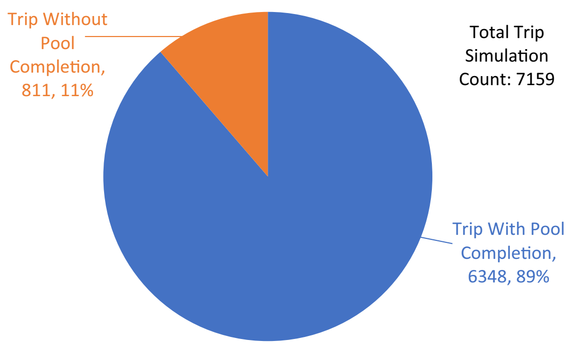
Furthermore, we would like to present some observations, as stated in the following Table 8. We define Phase 1 as the model without machine learning, where the matching was based on exact, closer, and alternative characteristics. Also, in Phase 1, we manually altered the characteristics in "closer" matching. In Phase 2, we made use of variance and recommender systems for matching and classifier computation. Also, in Phase 2, we made use of SVM to predict classifiers and get better rider recommendations.

Machine learning has a powerful impact on our system. From the observations in Table 8, Phase 2 gave better results. Even though we traversed through fewer riders, we computed a higher number of trips, T_i with pool completion. This is one of the major goals achieved in both phases, where we complete a maximum number of trips with pool completion, and the result is portrayed in Fig. 8.

Our system's efficiency is also measured with the average time for simulation completion, T_i, matching rate, MR_i, and the number of trips computed, $trip\_count_i$ for every simulation event, S_i. In Phase 2, there is a high increase in T_i as compared to Phase 1. The average simulation time is reflected in Fig. 12. The system proves to be efficient if it follows the condition that for an increase in T_i, the values of MR_i and $trip\_count_i$ should also increase. If MR_i and $trip\_count_i$ decrease with the increasing T_i, the system fails to be efficient. As shown in Fig. 13 MR_i increases in consecutive simulations. Also, from Fig. 14, $trip\_count_i$ keeps rising with the increasing T_i for every simulation S_i.

In both phases, results reach our expectations and achieve the best with machine learning. The matching rate, MR_i and the number of computed trips $trip\_count_i$ keep increasing due to the increasing number of traversed riders, RC_i and increasing UTT, U_i. The greater the RC_i, there is more room for matching. Also, with increased U_i, riders at a little farther traveling distances are accepted. Hence, the matching rate and the number of trips depend on the number of riders and the UTT. It can be stated from the results, MR_i and $trip\_count_i$ is directly proportional to RC_i and U_i.

A major change observed was in matching by characteristics type. In Phase 1, we got less count for matching based on exact or closer characteristics. The number of matches by the type of matching is traced by $match_{closer}$ and $match_{alternative}$. In Phase 2, $match_{closer}$ is much higher than the $match_{alternative}$. The reason for a higher match is the replacement of manual



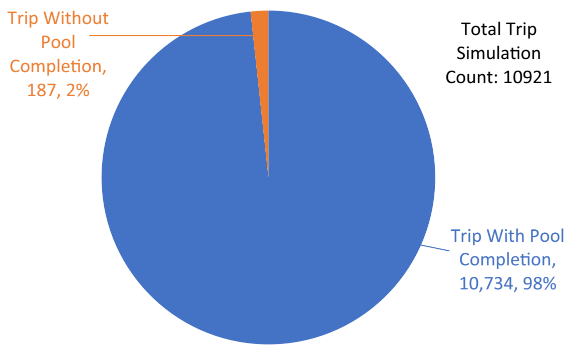


Fig. 8 Classification of trips with pool completion in Phase 1 (left) and Phase 2 (right)

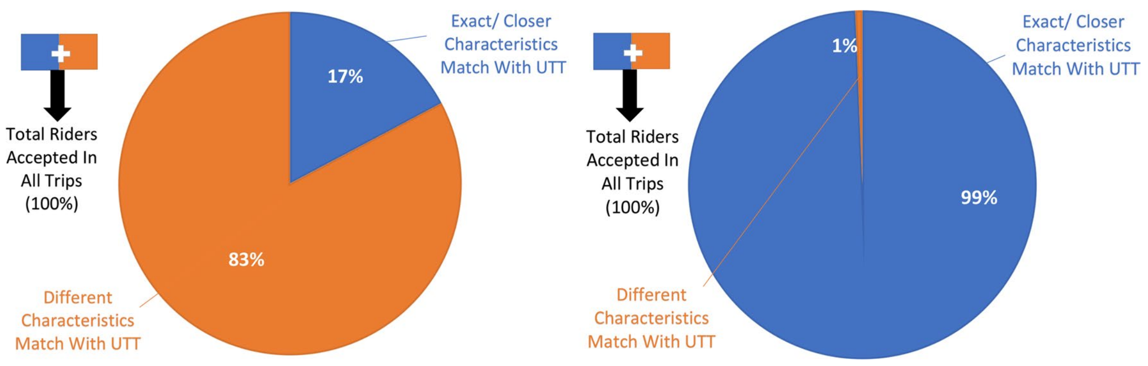


Fig. 9 Percentage of characteristic matching by type with the absence (left) and presence (right) of machine learning

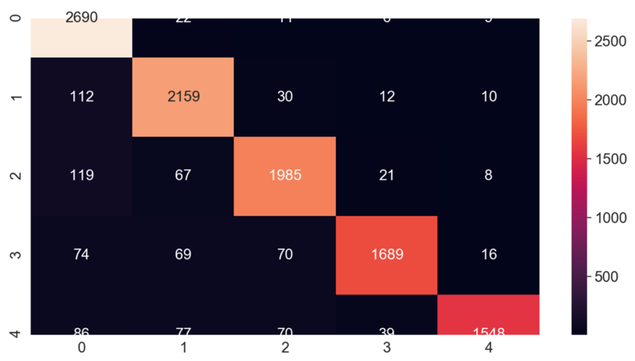


Fig. 10 Confusion Matrix for "Feedback-Give-Classifier" Support Vector Machine

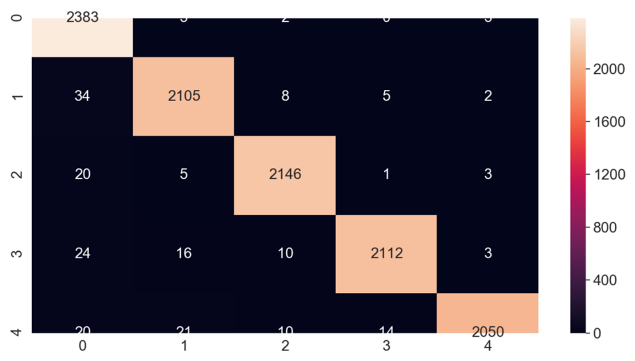


Fig. 11 Confusion Matrix for "Feedback-Received-Classifier" Support Vector Machine

altering of characteristics with the machine learning recommendation systems. This was the second major goal of the system where most of the users with similar liking are added on to a trip and are reflected in Fig. 9.

Overall, with the use of a machine learning algorithm, we were able to tune up the system efficiency significantly. We achieved a higher matching rate with a higher number of computed trips with pool completion in Phase 2. Also, as seen in the Observation Table 8, in both phases, the average trip formation time rounds up to a minute. We conclude this section by summarizing the key points as the achievements of our goals of maximum trips computed with pool completion and maximum matches observed for exact and closer characteristics matching.

6 Conclusion and future work

We implemented our designed model of vehicle sharing based on rider characteristics and user threshold time. To test the system efficiency, we subjected the model to an extensive simulation. Indeed, we achieved our major goals of completing a maximum number of trips with pool completion and getting maximum exact and closer characteristic match count in the characteristics matching layer.

Our results also prove that with the proposed technique, there is no decrement effect even increment on the matching rate and the total number of trips. The matching rate and the number of completed trips are an important measure from the perspective of system efficiency and is directly proportional to the number of riders traversed plus trip user threshold time. Our models run with an accuracy of 90% and predict classifiers for new registering riders, which is crucial in providing better rider recommendations.

Our future work includes building an Android or Web application for providing a UI for users. Additionally, we may implement a driver feature that allows drivers to switch the role between a rider and a driver. Also, riders may be allowed to select other riders as "favorites" and will be recommended if they are broadcasting at the same time on a similar commuting trajectory.

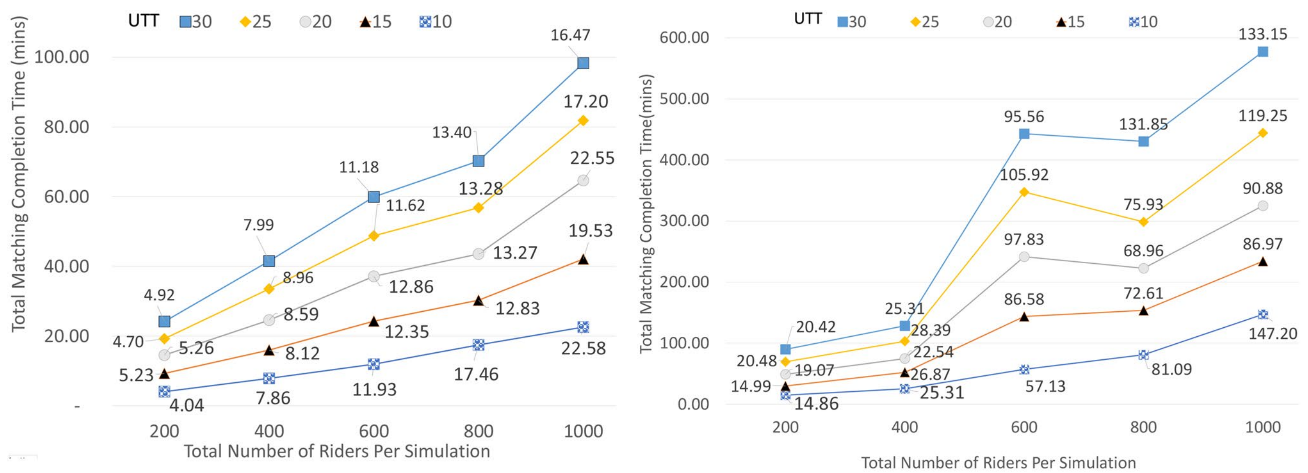


Fig. 12 Average simulation time in Phase 1 (left) and Phase 2 (right)

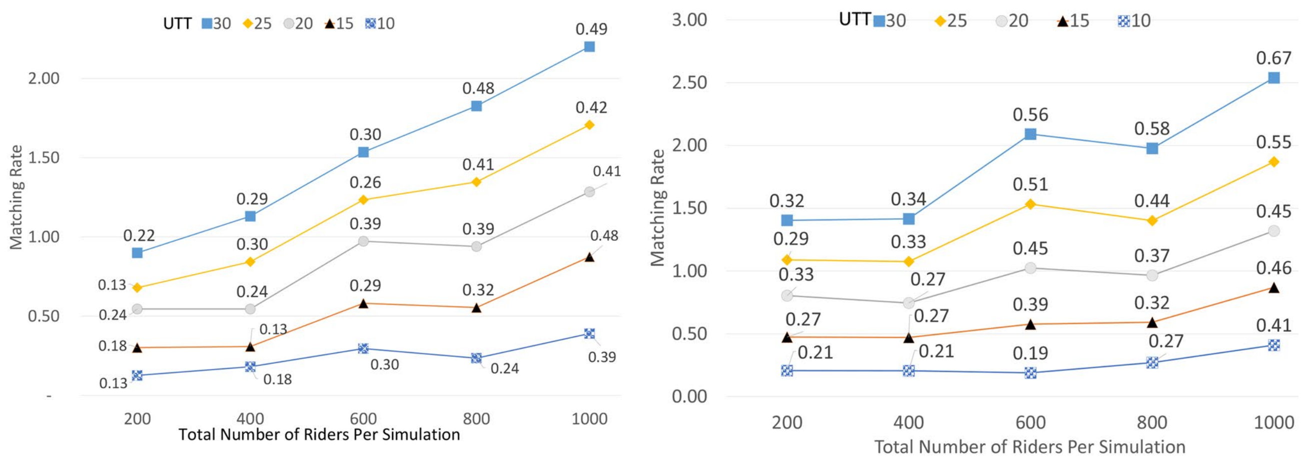


Fig. 13 Average matching rate achieved in Phase 1 (left) and Phase 2 (right)

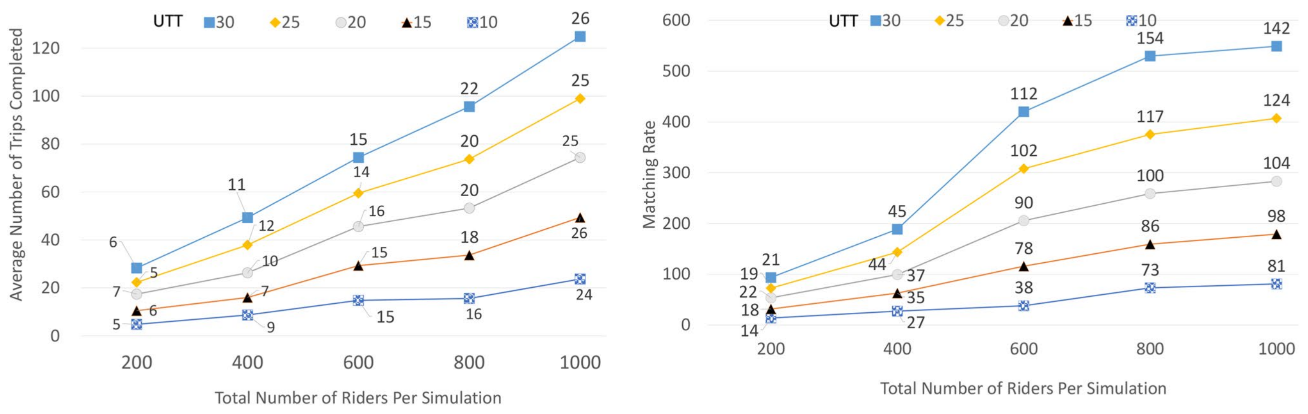


Fig. 14 Number of computed trips in Phase 1 (left) and Phase 2 (right)

Table 6 Accuracy Measures for "Feedback-Given-Classifier" Support Vector Machine

Overall SVM accuracy: 91.65%					
Root mean square error: 0.64					
Accuracy measure class wise					
Measurement(%)	Chat	Safe	Punctual	Friend	Comfort
F1 Score	92.34	91.65	91.07	91.92	90.90
Precision	87.04	90.40	91.97	95.84	97.35
Recall	98.31	92.94	90.20	88.32	85.25

Table 7 Accuracy Measures for "Feedback-Received-Classifier" Support Vector Machine

Overall SVM accuracy: 91.33%					
Root mean square error: 0.42					
Accuracy measure class wise					
Measurement(%)	Chat	Safe	Punctual	Friend	Comfort
F1 Score	87.85	89.02	90.63	93.22	93.21
Precision	86.13	87.52	92.58	91.97	95.48
Recall	89.21	88.82	89.67	94.49	96.96

Table 8 Observations for experimentations with and without machine learning

Observations	Phase1	Phase2
Total number of trips computed	7159	10921
Total trips computed with pool completion	6348	10734
Total number of riders traversed	276400	90800
Average trip formation time (mins)	0.80	1.02

Acknowledgements The previous incomplete versions of this paper were published in IEEE HONET-ICT (Yatnalkar and Narman 2019) and EDI40 (Yatnalkar et al. 2020).

References

Ahn S, Fessler JA (2003) Standard errors of mean, variance, and standard deviation estimators. The University of Michigan, EECS Department, pp 1–2

Apte JS, Messier KP, Gani S, Brauer M, Kirchstetter TW, Lunden MM, Marshall JD, Portier CJ, Vermeulen RC, Hamburg SP (2017) High-resolution air pollution mapping with google street view cars: exploiting big data. Enviro Sci Technol 51(12):6999–7008

Boldrini C, Bruno R. and Conti M. (2016). Characterising demand and usage patterns in a large station-based car sharing system. In: IEEE conference on computer communications workshops (INFOCOM WKSHPS)', pp. 572–577

Campana MG, Delmastro F. and Bruno R. (2016). A machine-learned ranking algorithm for dynamic and personalised car pooling services. In: IEEE 19th international conference on intelligent transportation systems (ITSC)', pp. 1856–1862

Carrese S, Giacchetti T, Patella SM. and Petrelli M. (2017) . Real time ridesharing: Understanding user behavior and policies impact: Carpooling service case study in Lazio Region, Italy. In: IEEE 5th international conference on models and technologies for intelligent transportation systems (MT-ITS)', pp. 721–726

Contreras SD, Paz A (2018) 'The effects of ride-hailing companies on the taxicab industry in Las Vegas. Nevada', Elsevier Transportation Research Part A: Policy and Practice 115:63–70

Cramer J, Krueger AB (2016) Disruptive change in the taxi business: The case of Uber. Am Econ Rev 106(5):177–82

De Lira VM, Perego R, Renso C, Rinzivillo S, Times VC (2018) Boosting ride sharing with alternative destinations. IEEE Tran Intell Trans Syst 19(7):2290–2300

Dehak N, Dehak R, Glass JR, Reynolds DA, Kenny P. et al. (2010). Cosine similarity scoring without score normalization techniques, *in* 'Odyssey', p. 15

Depari A, Ferrari P, Flammini A, Rinaldi S. and Sisinni, E. (2019). Lightweight machine learning-based approach for supervision of fitness workout. InL: IEEE Sensors Applications Symposium (SAS), pp. 1–6

Duan Y, Mosharraf T, Wu J. and Zheng H. (2018). Optimizing carpool scheduling algorithm through partition merging. In: IEEE International conference on communications (ICC), pp. 1–6

Fang X, Hodge B-M, Bai L, Cui H, Li F (2018) Mean-variance optimization-based energy storage scheduling considering day-ahead and real-time lmp uncertainties. IEEE Trans Power Syst 33(6):7292–7295

Han S, Qubo C. and Meng H. (2012). Parameter selection in SVM with RBF kernel function. In: IEEE World Automation Congress, pp. 1–4

He W, Yan G, Da Xu L (2014) Developing vehicular data cloud services in the IoT environment. IEEE Trans Ind Inform 10(2):1587–1595

He Y, Ni J, Wang X, Niu B, Li F, Shen X (2018) Privacy-preserving partner selection for ride-sharing services. IEEE Trans Vehicular Technol 67(7):5994–6005

Huang S-C, Jiau M-K, Lin C-H (2014) A genetic-algorithm-based approach to solve carpool service problems in cloud computing. IEEE Trans Intell Trans Syst 16(1):352–364

Huang X. and Peng H. (2018). Efficient mobility-on-demand system with ride-sharing. In: IEEE 21st international conference on intelligent transportation systems (ITSC), pp. 3633–3638

Jiang S, Chen W, Li Z, Yu H (2019) Short-term demand prediction method for online car-hailing services based on a least squares support vector machine. IEEE Access 7:11882–11891

Jiang S, , B. (2018). Kernel support vector machines and convolutional neural networks. In: IEEE Digital Image Computing: Techniques and Applications (DICTA), pp. 1–7

Kim E (2012) Enjoy the silence: commuters are 'nonsocial' for good reason. [Online Accessed November 20, 2020]. https://theconversation.com/enjoy-the-silence-commuters-are-nonsocial-for-good-reason-8698

Li Z, Hong Y. and Zhang Z. (2016). An empirical analysis of on-demand ride sharing and traffic congestion. In: International Conference on Information Systems

Liang L, Ye H, Li GY (2018) Toward intelligent vehicular networks: a machine learning framework. IEEE Internet of Things 6(1):124–135

Luo C, Zhan J, Xue X, Wang L, Ren R. and Yang Q. (2018). Cosine normalization: Using cosine similarity instead of dot product in neural networks. In: Springer International Conference on Artificial Neural Networks, pp. 382–391

Ma L, Fu T, Blaschke T, Li M, Tiede D, Zhou Z, Ma X, Chen D (2017) Evaluation of feature selection methods for object-based land cover mapping of unmanned aerial vehicle imagery using random forest and support vector machine classifiers. Multidisciplinary Dig Publ Inst ISPRS Int J Geo-Inform 6(2):51

Mallus M, Colistra G, Atzori L, Murroni M. and Pilloni V. (2017). A persuasive real-time carpooling service in a smart city: A case-study to measure the advantages in urban area. In: IEEE 20th conference on innovations in clouds, internet and networks (ICIN). pp. 300–307

Nguyen HV. and Bai L. (2010). Cosine similarity metric learning for face verification. In: Springer Asian Conference on Computer Vision pp. 709–720

NYC (2019) NYC open data. [Online Accessed November 1, 2019]. https://data.cityofnewyork.us/Transportation/NYC-Taxi-Zones/d3c5-ddgc

Rodriguez G (2019) Autonomous vehicles and unmanned aerial systems: data collection and liability [leading edge]. IEEE Technol Soc Mag 38(3):14–16

Sáez JA, Krawczyk B, Woźniak M (2016) Analyzing the oversampling of different classes and types of examples in multi-class imbalanced datasets. Elsevier Pattern Recognit 57:164–178

Shahane N, Kaul M. and Zheng Y. (2019). Exploratory analysis of chicago taxi rides. In: ACM Proceedings of the 20th annual SIG conference on information technology education. pp. 158–158

Shaheen S, Cohen A (2019) Shared ride services in North America: definitions, impacts, and the future of pooling. Taylor Francis Trans Rev 39(4):427–442

Streitz N (2019) Beyond 'smart-only' cities: redefining the 'smart-everything' paradigm. J Ambient Intell Human Comput 10(2):791–812

Swami A (2018) Impact of automobile induced air pollution on road side vegetation: a review. ESSENCE Int J Environ Rehabilit Conservation IX 1:101–116

Tang Y, Zhang Y-Q, Chawla NV, Krasser S (2008) 'Svms modeling for highly imbalanced classification', IEEE Transactions on Systems, Man, and Cybernetics. Part B (Cybernetics) 39(1):281–288

Teubner T, Flath CM (2015) The economics of multi-hop ride sharing. Springer Busin Inform Syst Eng 57(5):311–324

Uber, (2019) How does Uber match riders with drivers?. [Online Accessed November 1, 2019]. https://marketplace.uber.com/matching

Wang L, Geng X, Ma X, Zhang D, Yang Q (2019) Ridesharing car detection by transfer learning. Elsevier Artificial Intell 273:1–18

Wang X (2019) Preparing the public transportation workforce for the new mobility world, in 'Empowering the New Mobility Workforce'. Elsevier, Amsterdam, pp 221–243

Wang Y, Gu J, Wang S, Wang J (2019) Understanding consumers' willingness to use ride-sharing services: the roles of perceived value and perceived risk. Elsevier Trans Res Part C 105:504–519

Xu Z. and Zhou Q. (2020). 'Special issue on multi-modal information learning and analytics for smart city. *Journal of Ambient Intelligence and Humanized Computing* **11**

Yatnalkar G, Narman HS, Malik H (2020) An enhanced ride sharing model based on human characteristics and machine learning recommender system. Proc Comput Sci 170:626–633 The 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops

Yatnalkar GP. and Narman HS. (2019). A matching model for vehicle sharing based on user characteristics and tolerated-time. In: IEEE 16th international conference on smart cities: improving quality of life using ICT IoT and AI (HONET-ICT), pp. 143–147

Ye H, Liang L, Li GY, Kim J, Lu L, Wu M (2018) Machine learning for vehicular networks: recent advances and application examples. IEEE Vehicular Technol Mag 13(2):94–101

Zhao XY. and Su Q. (2019). Existing issues of ride sharing company operation and sharing economy in China: Uber case analysis. In: 5th annual international conference on management, economics and social development (ICMESD)