Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 170 (2020) 626–633

Procedia Computer Science

www.elsevier.com/locate/procedia

The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40), April 6 - 9, 2020, Warsaw, Poland

An Enhanced Ride Sharing Model Based on Human Characteristics and Machine Learning Recommender System

Govind Yatnalkar, Husnu S. Narman*, Haroon Malik

Marshall University, 1 John Marshall Dr, Huntington, WV 25755, USA

Abstract

Ride Sharing provides benefits like reducing traffic and pollution, but currently, the usage is significantly low due to social barriers, long rider waiting time, and unfair pricing models. Considering the aforementioned issues, we present an Enhanced Ride Sharing Model (ERSM) in which riders are matched based on a specific set of human characteristics using Machine Learning. After trip completion, we record the user feedback and compute two main characteristics that are most important to riders. The registered and the computed characteristics are fed to a classification module, which later predicts the two main characteristics for new riders. We have carried an extensive simulation with Google Map APIs and real-time New York City Cab data to measure the model performance. Our proposed model and obtained results will help service providers to increase the usage of Ride Sharing, and implicitly preserve natural resources plus improve environmental conditions.

Keywords: Ride Sharing, Characteristics, Machine Learning, Recommender System

1. Introduction

There are several issues in the transportation domain like vehicular pollution, traffic, accidents, and unavailability of parking spaces [1]. Ride Sharing is synonymous with Car-Pooling and is a practical solution if applied effectively [7, 11]. For example, if five users share a ride, they cut down the usage of four cars, which implies a reduction in fuel consumption and carbon emissions by almost 80% [1]. Additional advantages include splitting the stress and fares among riders and encouraging social interactions in a trip [17, 4]. Despite copious benefits, Ride Sharing is discouraged due to social barriers since riders do not possess any knowledge of other commuting riders [19].

Our *aim* in this paper is to deliver an Enhanced Ride Sharing Model that addresses the limitations in the transportation domain and current Ride Sharing models, which are discussed in the section of Literature Survey. The first

* Corresponding author: Husnu S. Narman
E-mail address: narman@marshall.edu

1877-0509
10.1016/j.procs.2020.03.135

task in our model is noting the five human characteristics on a scale of 1 to 5. The characteristics selected are often associated while traveling on a trip or while commuting in a group, and therefore are punctuality, chatty, safety, comfortability, and friendliness. Also, the task includes noting an approximate or estimated maximum waiting time a rider waits while picking other riders. If riders satisfy certain matching layer conditions, they are added to the trip itinerary. The system also consists of a newly designed feedback system where riders rate the driver and riders in terms of the five characteristics on a scale of 0 to 5. The feedback data-set is later used by the model to compute two classifiers for every user. The classifiers indicate what kind of characteristics a user expects in other users while commuting on a trip. In the end, every rider is associated with two main characteristics. The system tends to pair the riders having similar classifiers and if they are traveling on a similar trajectory. The registered characteristics and the computed classifiers are fed to a Machine Learning module to predict the characteristics of newly registering riders. Riders are recommended based on these predicted characteristics in future trips.

Our *key contributions* include: (i) Implementing the proposed Ride Sharing model using characteristics matching layer and Machine Learning recommender system (ii) Recording rider feedback and computing rider classifiers (iii) Training and testing the Machine Learning module for prediction accuracy (iv) Subjecting the model to an extensive simulation for evaluating the model performance. Our observations and results show that our model is feasible and can be deployed to increase the usage of Ride Sharing.

2. Literature Survey

2.1. Popular Ride Sharing Applications and Their Limitations

Our study began with an in-depth inspection of several famous Ride Sharing applications like UberPool, LyftLine, Juno, Curb, Wingz, Via, Flywheel, Zimride, and Waze [10, 15, 16, 6]. Some of the common limitations and reasons for disputes observed in all the applications are that drivers get to know the count of passengers at the pickup point [18], and in many trips, the vehicle is occupied with only one passenger, which is entirely against the essence of Ride Sharing [11, 18]. Additional limitations include users do not have basic details of other users they are traveling with, unfair pricing [2], and sudden addition of riders, which adds a significant amount of time in trip completion due to far locations [11, 2].

Noting the limitations in applications, we have designed our model considering most of the discovered limitations. While matching, we first perform the Exact match, which finds riders with exactly matching characteristics. If the pool is incomplete, we find riders with a little different or Closer characteristics. If the pool remains incomplete, we incorporate the current Uber or Lyft model of matching riders irrespective of characteristics [8]. Utilizing the three types of matching in the system ensures that we serve most of the broadcasting rider requests and completes the pool for a maximum number of trips. After having every passengers' details, we provide the trip itinerary to every rider, including the driver, before commencing the trip, which assists in reducing the social barriers among riders. The types of matching are illustrated in Table 1.

Table 1. Types of Rider Characteristics Matching.

Riders(%)	Chatty	Safety	Punctuality	Friendliness	Comfortability
Broadcasting Rider Characteristics	2	3	4	4	2
Search Riders having Exact Characteristics	2	3	4	4	2
Search Riders having Closer Characteristics (Little Different Characteristics)	3 (+1)	3	4	3 (-1)	2
Search Riders Irrespective of Characteristics or having Alternative Characteristics	4	1	5	5	5

2.2. Tracking Rider Characteristics

The designed model allows the riders to provide the feedback only to the users they have travelled before on trips. The method for tracking the data characteristic utilizes rider feedback records. For example, a user may constantly rate

with a high score of 4 or a low score of 0 to specific characteristic for several trips, implying users are less interested in that specific characteristic. Our motive is to find the characteristics the user is most interested in and recommend riders based on the tracked characteristics. The methodology selected for tracking rider characteristics is variance and is demonstrated with the help of lists, $L_1 = [1, 0, 5, 4, 0]$, $L_2 = [0, 0, 0, 0, 2]$, and $L_3 = [4, 4, 4, 4, 4]$. The total number of sample points in each list is given by N and the mean is denoted by x_i. The difference $x_{distance}$ of data-point x to the mean x_i is computed by $x - x_i$ [12]. The general definition of the variance is the average of squared differences from the mean [12]. Variance indicates the level of spread of each sample point in a data-set [12] and is given by Equation 1. The larger the variance of a data-set, the higher is the data-variety [12]. If the variance is applied to all three lists, the highest score is computed for L_1 as the data-variety in L_2 and L_3 is notably low [12]. If a similar methodology is applied for every characteristic feedback, the characteristic feedback with the highest variance is the characteristic the rider focusses the most.

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

2.3. Machine Learning Module Selection

After researching several methods for matching, we selected the Machine Learning Content-Based recommendation system. In this system, the features are converted to vectors and represented in d-dimensional space, where d is the number of features [3]. The angular distance or the cosine of the angle, θ between the vectors is calculated using the dot product equation [3, 13]. Vectors with the highest cosine values are deemed as the best match. We made a similar use where the features represented the registered rider characteristics, and riders with higher cosine similarities are paired up on a trip.

The classification module we selected for classifier prediction is the Support Vector Machines (SVM) because of SVM's Radial Bias Function (RBF) Kernel. The RBF kernel is a highly non-linear curve that is used for distinguishing classes [5]. SVM works on the principle of placing the line to the closest data-point with maximum distance. The placement of the line is changed through the regularization parameter, C, and the gamma parameter, γ [5]. The process of governing the curve placement is called Kernelization [5, 9]. The regularization allows a small error of fitting of a class, including fewer data-points of other classes and is best suited for imbalanced data-sets [5, 9].

3. Proposed Model

3.1. System Architecture

The key elements of the proposed architecture are the broadcasting rider request, the closest driver, matching layers, the feedback system, and the Machine Learning module. Figure 1 reflects the system architecture. The broadcasting rider request consists of rider source, destination, and the rider user-id. The registered five characteristics are retrieved from the data server and are referenced throughout the trip while searching for more riders using the user-id. For the best simulation practices, we have utilized the New York Cab location database [14]. The NYC Cab locations are divided into 263 zones that are useful to avoid a larger search. The closest available driver is retrieved from the same source broadcasting rider's source zone using the Google Map Distance Matrix API. The key element in driver association is noting the vehicle seating capacity.

The next step is the execution of the characteristics matching layer. All the broadcasting riders having Exact or Closer characteristics from the same source zone are retrieved. Riders go through the ML-based recommendation system and are later added to the final trip itinerary. The next step is of 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. The following step is the training and testing of the Machine Learning classification model. 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.

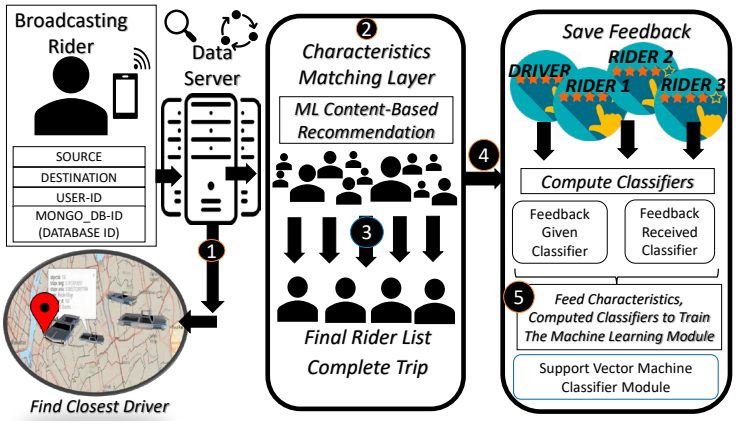

Fig. 1. The System Architecture.

3.2. Machine Learning Reccomendation System

Initially, rider characteritics are converted to a vector. For example, a broadcasting rider $rider_{br}$ with characteristics be chatty:3, safety:4, punctuality:3, friendliness:3 and comfortability:4 is represented as $char\_v_{br} = [3,4,3,3,4]$. In Figure 2, 'O' represents the origin and points 'B', '1', and '2' represent the points plotted by the vectors for broadcasting and other riders. The next step is to compute the angular distance between vectors or the cosine of angle θ_{ab} using the dot-product Equation 2.

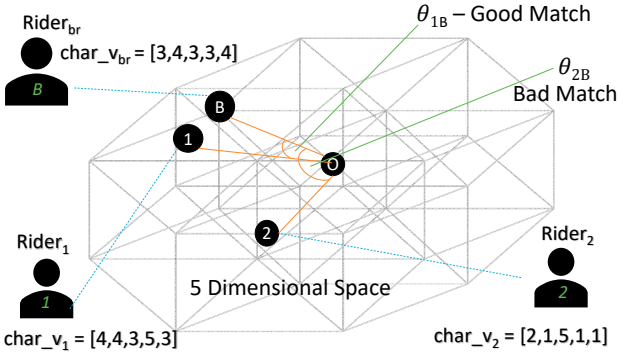

Fig. 2. Rider matching using Content-Based recommendation.

$$cos\theta_{ab} = \frac{(char\_v_a \cdot char\_v_b)}{||char\_v_a|| ||char\_v_b||} \tag{2}$$

The case of rider matching by Exact characteristics is when the θ_{ab} equal to 0 or $cos\theta_{ab}$ is 1 . Therefore, a greater cosine value means a greater match. In Figure 2 $char\_v_1$ or $Rider_1$ seems to be a better match than $char\_v_2$ due to a smaller angle θ_{1B}. In rider matching simulations, we accept riders only with a $cos\theta_{ab}$ value above 0.85 or 85% which we felt is enough percentage to indicate a good match.

3.3. Computation of Classifiers

The first classifier is called the Feedback-Given-Classifier and uses the first part of the feedback data which includes the ratings given by a rider to other riders. For example, let the feedback given by $Rider_1$ to other riders be as shown in Table 2. The data given by the rider is segregated characteristic wise and appended in new lists as follows: $chatty_{Rider1} = [0,0,1]$, $safety_{Rider1} = [2,3,5]$, $punctuality_{Rider1} = [1,0,0]$, $friendliness_{Rider1} = [4,4,4]$, and

Table 2. Feedback given by $Rider_1$ to riders $Rider_1$, $Rider_2$, and $Rider_3$.

Riders(%)	Chatty	Safety	Punctuality	Friendliness	Comfortability
$Rider_2$	0	2	1	4	0
$Rider_3$	0	3	0	4	0
$Rider_4$	1	5	0	4	0

$comfortability_{Rider1} = [0, 0, 0]$. 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 is in the safety rating. Thus the $Rider_1$ Feedback-Given-Classifier is the safety class. The computation is done using the variance equation. Total number of elements in a characteristic list is n_{char} or $data\_count_{char}$. The mean is denoted by x_{char_i} and variance equation is stated in Equation 3.

$$\sigma^2_{char} = \frac{\sum_{i=1}^{n_{char}} (x - x_{char\_i})^2}{data\_count_{char}} \tag{3}$$

The characteristic list with the highest variance is selected as this proves that the user is more diverse in rating the characteristic and therefore focusses more on the specific characteristic. After getting the first classifier, the system computes the Feedback-Received-Classifier using the second part of the feedback data-set, which is the feedback received by other riders to a rider.

Table 3. Feedback provided to $Rider_1$ by riders $Rider_2$, $Rider_3$, and $Rider_4$.

Riders(%)	Chatty	Safety	Punctuality	Friendliness	Comfortability
$Rider_2$	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

Let the feedback given to $Rider_1$ be as shown in the Table 3. Each element in the column has two values. The first value is the rating given by the rider for a specific characteristic, and the second value is the characteristic variance ($(\sigma_{i\_char})^2$) computed for the individual rider's characteristics. Every time a rider provides feedback, the feedback value is multiplied by the corresponding characteristic variance. To exemplify, $Rider_2$ variance for safety is 4.31, and the safety rating to $Rider_1$ is 2. The computed feedback to $Rider_1$ safety characteristic is $2 * 4.31$, which is 8.62. In the end, for every characteristic, all multiplications are added and compared to get the highest characteristic value, which forms the Feedback-Received-Classifier. In the same example, the classifier is chatty, as the value is highest, which is 39.26. After computing the two classifiers, every rider's search criteria is dynamically defined by the system. The scenario at this stage resembles a practical use case as riders classify other riders based on their past experiences, which assists in better and real-time recommendations to riders.

3.4. Machine Learning Model & Prediction

Table 4. Sample rows in Feedback-Given-Classifier ML-data or Feedback-Received-Classifier Data ML-data.

Chatty	Safety	Punctuality	Friendliness	Comfortability	Class_Given/ Class_Received
3	3	4	1	4	Comfortability
1	2	4	3	5	Chatty

We selected Support Vector Machine or SVMs as our Machine Learning prediction module. We have created two data-sets, namely Feedback-Given-Classifier ML-data and Feedback-Received-Classifier ML-data, for training the SVM. In both data-sets, the input fields are registered user characteristics, and the output is the computed classifier. Sample rows in both data-set are shown in Table 4. For two data-sets, we have two distinct SVM modules. The first SVM outputs Feedback-Given-Classifier and the other outputs Feedback-Received-Classifier. We have maintained the results in a document for every model training and testing events. We also tested the module with newly registered users. Both modules predict the two classifiers for the newly registered riders, and the system recommends riders based on the predicted classifiers.

3.5. Experimentations

A simulation is denoted by $\{U_i, RC_i\}$ where U_i denotes the registered waiting time of riders, and RC_i denotes the number of riders traversed for the i^{th} iteration. In simulations, for drafting the results, we selected the registered waiting time of riders from 10 to 30 minutes in multiples of 5. Also, the number of traversed riders is selected from 200 to 1000. The first simulation is represented by $S_1 = \{U_1, RC_1\} = \{10, 200\}$. For every next simulation, the U_i is kept the same, and the RC_i is increased by 200 until it reaches 1000. As the RC_i reaches 1000, it is reset to 200, and the U_i is increased by 5 until it reaches 30. Indeed, the n^{th} or the last recorded simulation is given by $S_n = \{30, 1000\}$. For every simulation S_i, we noted the RP_i, the total number of riders accepted, and the $trip\_count_i$, the total number of trips computed. The matching rate is the division of accepted riders and the total number of traversed riders and is given by MR_i in Equation 4. The matching rate provides an idea of how many riders are accepted out of a total traversed population. According to our expectations, the matching rate and the number of computed trips should keep increasing for consecutive simulations. Two variables, $closer_i$, and $aletnative_i$ track how many riders are accepted by the Exact and Closer match or by the Alternative matching type for every simulation. In the end, we added the tracked values from all simulations and save them in $match_{closer}$ and $match_{alternative}$ as stated in Equation 4.

$$MR_i = \frac{RP_i}{RC_i} \qquad match_{closer} = \sum_{S_i=1}^{n} closer_i \qquad match_{alternative} = \sum_{S_i=1}^{n} alternative_i \tag{4}$$

4. Model Evaluation & Results

At first, we evaluate the Machine Learning model by computing the F1 score, precision, and recall for the five classifiers. The F1 score, recall, precision, and confusion matrix provides a comparison between computed scores and predicted scores. A higher score means the predicted classes match the computed classes for the same set of inputs. Tables 5 and Table 6 presents the performance measures for both SVMs.

Table 5. Accuracy Measures for Feedback-Given-Classifier SVM.

Overall SVM Accuracy: 91.65% | Root Mean Square Error: 0.64					
Measurement(%)	Chatty	Safety	Punctuality	Friendliness	Comfortability
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

The confusion matrix provides a two-dimensional array, which states how much error the module makes in predictions. The matrix reflects how much the model predicted correctly, also called as the true positive scores. In the case of the confusion matrix, if the diagonal elements have the highest values, the prediction of the model is notably accurate. From Figure 3, we conclude that our true positive scores for both SVMs are high, and the model predicts accurately. From Tables 5 and 6, the precision and recall is above 85% which is a good measure for a Machine Learning classification model. We indeed got an overall accuracy of 90% for both SVMs. Also, we computed the Root Mean

Table 6. Accuracy Measures for Feedback-Received-Classifier SVM.

Overall SVM Accuracy: 91.33% | Root Mean Square Error: 0.42					
Measurement(%)	Chatty	Safety	Punctuality	Friendliness	Comfortability
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

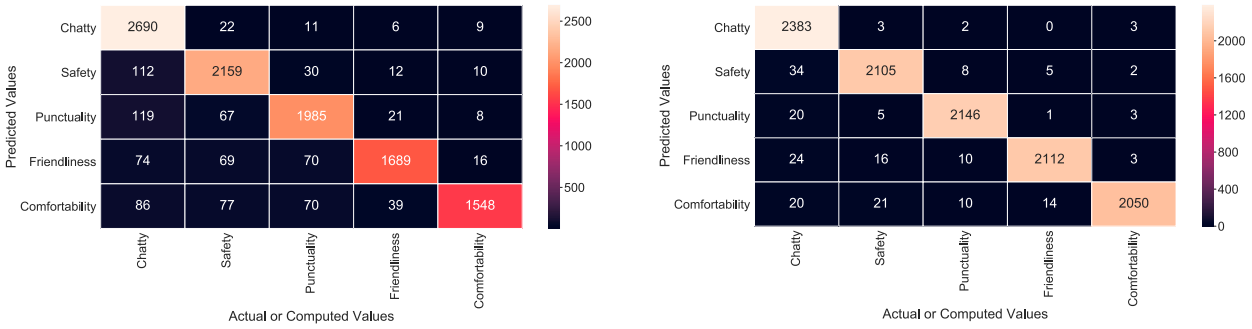

Fig. 3. Confusion Matrix for Feedback-Given-Classifier SVM (left) and Feedback-Received-Classifier (right) SVM.

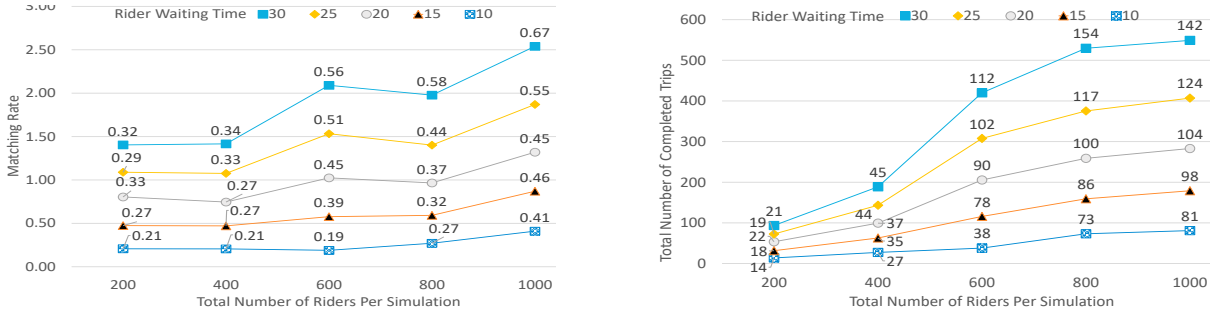

Fig. 4. Simulation matching rate (left) and total number of computed trips (right).

Square Error (RMSE) scores between the computed and predicted values, which provides the average of squared errors between every computed and predicted value. For both SVMs, we received a notably lower RMSE score, which contributes to model prediction quality.

Table 7. Observations from Simulations for Enhanced Ride Sharing Model Using Machine Learning.

Name of the Measure	Computed Values
Total number of trips computed	10921
Total trips computed with pool completion	10734 (98%)
Total number of riders traversed	90800
Percentage of riders accepted by Closer or Exact characteristics matching	99%
Average trip formation time (mins)	1.02

Furthermore, Table 7 provides model observations. We achieved a major goal of completing a maximum number of trips with pool completion. As shown in Figure 4, the matching rate, MR_i, and the number of computed trips

$trip\_count_i$ keeps increasing with the increasing number of traversed riders, RC_i. The greater the RC_i, there is more room for matching. Also, matches by Exact and Closer characteristics matching is much higher than the matches by Alternative type. The higher $match_{closer}$ than $match_{alternative}$ results in gaining one of the most important goals of the model, which is pairing most of the users with similar likings on the same trip, indirectly leading to a social and interactive journey.

5. Conclusion and Future Work

We implemented our designed and proposed model of Ride Sharing based on rider characteristics and Machine Learning Content-Based recommendation system. We subjected the model to an extensive simulation to test system performance. The matching rate and the number of completed trips continue rising with the progressing simulations or increasing number of traversed riders. Also, the SVM modules run with an accuracy of 90% and precisely predict classifiers for newly registering riders, which is crucial in providing better and real-time rider recommendations. Based on observations, the overall trip formation time rounds up to a minute. Indeed, we achieved our major goals of completing a maximum number of trips with pool completion and getting maximum rider matches by Exact and Closer characteristics matching.

Our future work includes building a full-fledged Android or Web application as well as a sophisticated pricing model for users. Also, riders may be allowed to add riders as "Favourites," and the system will recommend the added riders if they are broadcasting at the same time on a similar commuting trajectory.

References

[1] Apte, J.S., Messier, K.P., Gani, S., Brauer, M., Kirchstetter, T.W., Lunden, M.M., Marshall, J.D., Portier, C.J., Vermeulen, R.C., Hamburg, S.P., 2017. High-resolution air pollution mapping with Google street view cars: exploiting big data. ACS Environmental Science & Technology 51, 6999–7008.

[2] Cramer, J., Krueger, A.B., 2016. Disruptive change in the taxi business: The case of Uber. American Economic Review 106, 177–82.

[3] Dehak, N., Dehak, R., Glass, J.R., Reynolds, D.A., Kenny, P., et al., 2010. Cosine similarity scoring without score normalization techniques., in: Odyssey, p. 15.

[4] Duan, Y., Mosharraf, T., Wu, J., Zheng, H., 2018. Optimizing carpool scheduling algorithm through partition merging, in: IEEE International Conference on Communications (ICC), pp. 1–6.

[5] Han, S., Qubo, C., Meng, H., 2012. Parameter selection in SVM with RBF kernel function, in: IEEE World Automation Congress, pp. 1–4.

[6] He, Y., Ni, J., Wang, X., Niu, B., Li, F., Shen, X., 2018. Privacy-preserving partner selection for ride-sharing services. IEEE Transactions on Vehicular Technology 67, 5994–6005.

[7] Huang, S.C., Jiau, M.K., Lin, C.H., 2014. A genetic-algorithm-based approach to solve carpool service problems in cloud computing. IEEE Transactions on intelligent transportation systems 16, 352–364.

[8] Inc., U.T., 2019. How does Uber match riders with drivers? URL: https://marketplace.uber.com/matching. [Accessed November 1, 2019].

[9] Jiang, S., Hartley, R., Fernando, B., 2018. Kernel support vector machines and convolutional neural networks, in: IEEE Digital Image Computing: Techniques and Applications (DICTA), pp. 1–7.

[10] Li, Z., Hong, Y., Zhang, Z., 2016. An empirical analysis of on-demand ride sharing and traffic congestion, in: AIS International Conference on Information Systems.

[11] Mallus, M., Colistra, G., Atzori, L., Murroni, M., Pilloni, V., 2017. A persuasive real-time carpooling service in a smart city: A case-study to measure the advantages in urban area, in: 20th IEEE Conference on Innovations in Clouds, Internet and Networks (ICIN), pp. 300–307.

[12] Math, Science, . What is variance in statistics? learn the variance formula and calculating statistical variance! URL: https://www.youtube.com/watch?v=s0b9b_AtwDg.

[13] Nguyen, H.V., Bai, L., 2010. Cosine similarity metric learning for face verification, in: Springer Asian conference on computer vision, pp. 709–720.

[14] NYC, 2019. NYC open data. URL: https://data.cityofnewyork.us/Transportation/NYC-Taxi-Zones/d3c5-ddgc.

[15] Rodriguez, G., 2019. Autonomous vehicles and unmanned aerial systems: Data collection and liability [leading edge]. IEEE Technology and Society Magazine 38, 14–16.

[16] Shaheen, S., Cohen, A., 2019. Shared ride services in North America: definitions, impacts, and the future of pooling. Transport Reviews 39, 427–442.

[17] Teubner, T., Flath, C.M., 2015. The economics of multi-hop ride sharing. Springer Business & Information Systems Engineering 57, 311–324.

[18] Wang, X., 2019. Preparing the public transportation workforce for the new mobility world, in: Elsevier Empowering the New Mobility Workforce, pp. 221–243.

[19] 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 Transportation Research Part C: Emerging Technologies 105, 504–519.