Machine Learning - Bicycle-sharing Prediction

Weiming Long, Ruining Zhang*
*George Washington University, longweiming@.gwu.edu, ruining0916@gwu.edu

***Abstract*—As bicycle-sharing service is growing rapidly in last few years, there's still some problems need to solve. The optimal distribution of bicycles in different area and time is difficult to predict. However, as more bicycle-sharing data become available and machine learning become more reachable, we can now fully take use of data and predict the usage of bicycles at different occasions to provide a better user experiences for all users.**

Keywords*—Machine Learning, bicycle-sharing, weather, holiday, transport;

I. Bicycle-sharing Prediction

A bicycle-sharing is a service in which bicycles are made available for shared use to individuals on a short term basis for a price or free. Many bike share systems allow people to borrow a bike from a "dock" and return it at another dock belonging to the same system. Docks are special bike racks that lock the bike, and only release it by computer control. The user enters payment information, and the computer unlocks a bike. The user returns the bike by placing it in the dock, which locks it in place.

In this project, we are going apply different machine learning algorithm on a London bike sharing dataset [1] and intend to predict the total count of used sharing bike in one day our the input features are temperature, humidity, wind speed, weather and holiday and so on.

Bicycle-sharing in London has been growing rapidly in last few years. The number of cyclists on London roads has more than doubled in the past decade. London's public bike-sharing scheme, Santander Cycles, is available 24/7, 365 days a year. There are more than 750 docking stations and 11,500 bikes in circulation across London to help you get around quickly and easily. Santander Cycles bikes are available for hire at the docking-station terminal with a bank card or contactless payment card. Cycling in London is a good ways of exploring the city and hanging out.

The dataset we are using here is from mainly three sources. One is London government that released open public cycling data. Second is the weather data from freemeteo.com that would help our prediction. The usage of bicycle are closely related to the current weather. The usage would be drastically higher when it's sunny outside then raining or snowing. The third dataset is the official holidays from London government.

In terms of data preprocessing, we drop some features which are not actually relevant to the bike sharing count like timestamp firstly. Then in order to apply classification algorithm, we divide dataset to several category according the bike sharing count number. Specifically, dataset is separated into five class, which are 0 to 1000, 1000 to 2000, 2000 to3000, 3000 to 4000 and over 4000.As for the numeric features, we normalize them into 0-1 range.

We are looking for neural network model to make a classification as neural network is one of the most popular training models. The implementation of neural network bases on Back-Propagation algorithm.As we define this problem as a multi-class classification problem, we choose softmax-cross-entropy function as the activation function of output layer. And also, to avoid over-fitting in training process, we define a 'dropout' layer to deactivate some nodes randomly.

For the evaluation part we are going to use cross validation to check its performance, which is a technique that involves partitioning the original observation dataset into a training set, used to train the model, and an independent set used to evaluate the analysis. We will use confusion matrix to check false positive and false positive to find out the problem of our model and then tune it. We are also looking into the accuracy, precision, and recall for each changes. There are methods like logarithmic loss and area under curve.

We are looking to have a better understanding of bicycle-sharing service usage across different time and weather in London. There are situations that there are no bicycles available on the street when there's a high amount of usage demand. There are also time that there are a lot of bicycles not used outside which becomes a waste of resources. The distribution of bicycles across the city needs to be calculated precisely. There can be a high demand of bicycles in tourist attractions during holiday season as well as peak time in office concentrated areas. When the weather is severe outside like raining or snowing, the usage of bicycles can drop dramatically. Building up our machine learning model can predict the usage and let the bicycle-sharing company react to the high demand as well as low demand from time to time.

Reference

[1] Hristo Mavrodiev (2019,10) London bike sharing dataset-Historical data for bike sharing in London 'Powered by TfL Open Data' https://www.kaggle.com/hmavrodiev/london-bike-sharing-dataset

II. Introduction

The bicycle-sharing service has been growing rapidly in modern cities as the Internet connection and smartphones have become much more popular in our daily life. We can easily take out our smartphones, open the bike APP, and scan the QR code to use the bike. It's not only environmental friendly but also a good exercise or leisure for everyone. However, there are many challenges for the sharing services to run smoothly. The service provider has to consider about demand forecasting and bike distribution in different area and time. Those challenges bring machine learning problems to us. With given weather, date, time and season, how are we going to predict the amount of bike usage. Here we are using techniques from reinforced

learning which is the area have been put in a huge amount of development. It can be seen from automated chat bots and game bots, where a move will be taken by the model and depending upon the result of that move next step will be taken and the previous move will be learned by the model and it will use that knowledge in next attempts. Neural networks are built using smaller units called perceptrons that work like neurons in our brain. Data is fed into a network of nodes or perceptrons. Each node takes in the input data and decide how to categorize that data.

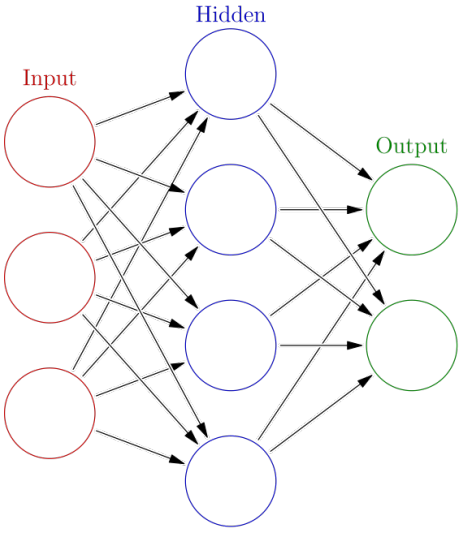

Fig. 1: Neural network model

We will multiply the data coming into perceptrons with values called weights. For example, the perceptron having two inputs will have two individual weights that can be adjusted individually. The weights will start as random values and will be adjusted based on errors in categorization that the previous iteration made. This is the training process of the neural network. Higher weight suggests the neural network consider that input to be more important.

Our model will be similar to Backpropagation Algorithm that consist two important processes which are forward pass and backward pass. In the forward pass, we will multiply weights with input data and adding those products together with bias and passing the output to the next neuron. In backward process, we will calculate the error and then back-propagate the error to update the weights.

III. Problem Statement

Many cities have realized the negative effects from the increasing number of vehicles on the roads such as congestion, greenhouse gas emission and air pollution. Governments are promoting bike sharing systems to prevent traffic jams and further air pollution. It's been shown as energy-efficient and healthy option for commuters or travellers in cities across the world. Data shows the bike sharing system has been introduced in over 1,139 cities and 50 countries. According to the National Association of City Transportation Officials in the U.S, in 2016 alone there were over 28 million bike trips, an increase of 25 percent compared to 2015. This increased

Fig. 2: Bike sharing system in London

usage of bikes led many cities to either expand their existing system or launch a new one. However, merely expanding the number of bikes are not optimal. The demand of bikes are unbalanced in space and time. Bike stations can be full or empty during the day. It would be so frustrated when returning a bike and find out the station is full or when renting a bike and the station is empty. Those problems significantly harm the reliability of bike sharing systems with less users and fail to ease the traffic jams and environmental problems. The essential part of rebalancing the bike sharing system is to predict the bike counts at space and time accurately that the imbalance can be discovered in advance and make the bike distribution accordingly. Many factors can have effects on bike usage such as weather, time, and date.

IV. Related Work

There have been different researches using different methods to predict bike counts at stations such as regression, count models, clustering, and exploring algorithms, etc. These methods use many input variables such as weather and time information. The Deep Learning Network model have been used for similar prediction[1].For example, there's an approach proposed artificial immune system predication framework, combined with back-propagation neural networks to achieve numerical prediction, is named the AIS-ANN. The components of the AIS are cells, antibodies and antigens; the operation of the first two components can eliminate antigens in the human body. A set of ANN sub-models generated by partial features with samples of training data are forming a set of cells. The cells' combinatorial optimization and affinity between antigens and antibodies are key points for numerical prediction[2]. Another interesting way is using app popularity prediction and competitive analysis. It uses Popularity-based Hidden Markov Model (PHMM) to model the popularity information of mobile apps, which can learn the sequences of heterogeneous popularity observations from mobile Apps. Competitive analysis involves the early identification of potential risks and opportunities by gathering and analyzing information about the environment, which is important to help managers making strategic decisions for an enter- prise[3]

References

[1] Chandrasegar Thirumalai. Bike sharing prediction using deep neural network. *International Journal of Informatics Visualization*, 3.

[2] Pei-Chann Chang. Bike sharing demand prediction using artificial immune system and artificial neural network. *Soft Comput*, 213(6334):613–626, 2017.

[3] Bin Guo Yi Ouyang. Competitivebike: Competitive analysis and popularity prediction of bike-sharing apps using multi-source data. *IEEE*, 18(8):1760–1773, 2019.

V. Dataset

The dataset we are using is the bike sharing data in both London

"timestamp" - timestamp field for grouping the data

"cnt" - the count of a new bike shares

"t1" - real temperature in C

"t2" - temperature in C "feels like"

"hum" - humidity in percentage

"wind_speed" - wind speed in km/h

"weather_code" - category of the weather

"is_holiday" - boolean field - 1 holiday / 0 non holiday

"is_weekend" - boolean field - 1 if the day is weekend

"season" - category field meteorological seasons: 0-spring ; 1-summer; 2-fall; 3-winter.

"weather_code" category description:

1 = Clear ; mostly clear but have some values with haze/fog/patches of fog/ fog in vicinity

2 = scattered clouds / few clouds

3 = Broken clouds

4 = Cloudy

7 = Rain/ light Rain shower/ Light rain

10 = rain with thunderstorm

26 = snowfall

94 = Freezing Fog

timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
2015-01-04 00:00:00	182	3.0	2.0	93.0	6.0	3.0	0.0	1.0	3.0
2015-01-04 01:00:00	138	3.0	2.5	93.0	5.0	1.0	0.0	1.0	3.0
2015-01-04 02:00:00	134	2.5	2.5	96.5	0.0	1.0	0.0	1.0	3.0
2015-01-04 03:00:00	72	2.0	2.0	100.0	0.0	1.0	0.0	1.0	3.0
2015-01-04 04:00:00	47	2.0	0.0	93.0	6.5	1.0	0.0	1.0	3.0
2015-01-04 05:00:00	46	2.0	2.0	93.0	4.0	1.0	0.0	1.0	3.0
2015-01-04 06:00:00	51	1.0	-1.0	100.0	7.0	4.0	0.0	1.0	3.0
2015-01-04 07:00:00	75	1.0	-1.0	100.0	7.0	4.0	0.0	1.0	3.0
2015-01-04 08:00:00	131	1.5	-1.0	96.5	8.0	4.0	0.0	1.0	3.0
2015-01-04 09:00:00	301	2.0	-0.5	100.0	9.0	3.0	0.0	1.0	3.0
2015-01-04 10:00:00	528	3.0	-0.5	93.0	12.0	3.0	0.0	1.0	3.0
2015-01-04 11:00:00	727	2.0	-1.5	100.0	12.0	3.0	0.0	1.0	3.0
2015-01-04 12:00:00	862	2.0	-1.5	96.5	13.0	4.0	0.0	1.0	3.0
2015-01-04 13:00:00	916	3.0	-0.5	87.0	15.0	3.0	0.0	1.0	3.0
2015-01-04 14:00:00	1039	2.5	0.0	90.0	8.0	3.0	0.0	1.0	3.0
2015-01-04 15:00:00	869	2.0	-1.5	93.0	11.0	3.0	0.0	1.0	3.0
2015-01-04 16:00:00	737	3.0	0.0	93.0	12.0	3.0	0.0	1.0	3.0
2015-01-04 17:00:00	594	3.0	0.0	93.0	11.0	3.0	0.0	1.0	3.0
2015-01-04 18:00:00	522	3.0	1.5	93.0	6.5	3.0	0.0	1.0	3.0
2015-01-04 19:00:00	379	3.0	1.0	93.0	7.0	3.0	0.0	1.0	3.0

Fig. 3: The format of dataset

VI. Approach

Here we are taking use of neural network model and Backpropagation Algorithm (BP) with Mini-Batch Gradient Decent (MBGD) and momentum. The BP algorithm can efficiently computes the gradient of the loss function with respect to the weights of the network for a single input-output example. This enables it feasible to use gradient methods for training multilayer networks, updating weights to minimize loss.

In general, Backpropagation computes the gradient in weight space of a feedforward neural network, with respect to a loss function. During model training, the input-output pair is fixed, while the weights vary, and the network ends with the softmax-cross-entropy function. The weights will be set randomly at first and then the neuron learns from training examples. A loss function is used for measuring the discrepancy between the target output and the computed output. Now if the relation is plotted between the network's output y on the horizontal axis and the error E on the vertical axis, the result is a parabola. The minimum of the parabola corresponds to the output y which minimizes the error E. For a single training case, the minimum also touches the horizontal axis, which means the error will be zero and the network can produce an output y that exactly matches the target output t. Therefore, the problem of mapping inputs to outputs can be reduced to an optimization problem of finding a function that will produce the minimal error. One commonly used algorithm to find the set of weights that minimizes the error is gradient descent. Backpropagation is then used to calculate the steepest descent direction in an efficient way.

The gradient descent method involves calculating the derivative of the loss function with respect to the weights of the network. The derivative of the output of neuron with respect to its input is simply the partial derivative of the activation function. There are also drawbacks like gradient descent with backpropagation is not guaranteed to find the global minimum of the error function, but only a local minimum; also, it has trouble crossing plateaus in the error function landscape.

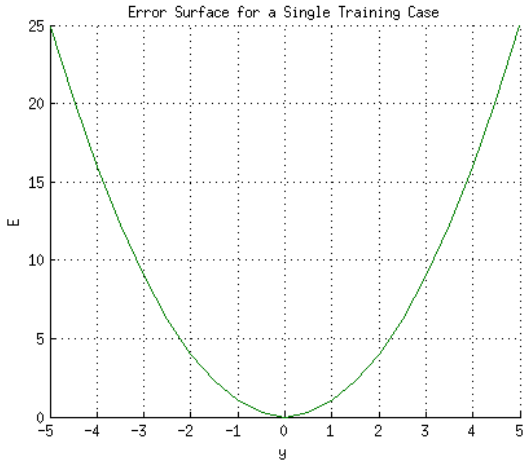

Fig. 4: Error surface of a linear neuron for a single training case

In order to improve generalization performance and reduce over-fitting , we implement an dropout layer among hidden

$$\delta_i^{(L)} = -(y_i - a_i^{(L)}) f'(z_i^{(L)}) \quad \text{(BP-1)}$$

$$\delta_i^{(l)} = \left(\sum_{j=1}^{n_{l+1}} \delta_j^{(l+1)} w_{ji}^{(l+1)} \right) f'(z_i^{(l)}) \quad \text{(BP-2)}$$

$$\frac{\partial E}{\partial w_{ij}^{(l)}} = \delta_i^{(l)} a_j^{(l-1)} \quad \text{(BP-3)}$$

$$\frac{\partial E}{\partial b_i^{(l)}} = \delta_i^{(l)} \quad \text{(BP-4)}$$

Fig. 5: BP Algorithm equation overview

layers which randomly deactivated some neurons and using learning curve to find a better dropout rate in this model.

VII. Experiments

In order to make training the neural network easier, we'll standardize each of the continuous variables. The scaled variables will have mean of zero and standard deviation of 1. The scaling factors are saved so we can go backward when we train the neural network. We tried both three-layered neural networks and four-layered neural networks models. There are hidden layers in the middle and output layer on the right. The hidden layers will apply ReLU function for activations. The model will work through each layer to calculate the output for each neuron. All of the outputs from one layer will become inputs to the neurons on the next layer. This process is called forward propagation. The weights are used to propagate signals forward from the input to the output layer. We also use those weights to also propagate error backward from the output back into the network to update our weights. This process is the back propagation.

Another experiment that we tried is using the dropout for neural network. When a large neural networks trained on relatively small datasets, there can be over-fitting happen. Dropout is a regularization method that some number of layer outputs are randomly ignored during training. This has the effect of making the layer looked like and treated like a layer with a different number of neurons to the prior layer. The effect is that the network becomes less sensitive to the specific weights of neurons. This in turn results in a network that is capable of better generalization and is less likely to overfit the training data.

VIII. Results

A. Subsection

Firstly, we used two-layered neural network and run 20 epoches to calculate the accuaracy.

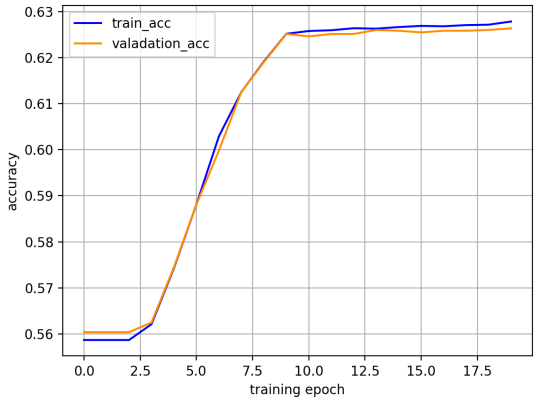

Fig. 6: Accuracy after each epoch

Later we add dropout method for our model. We can tell from the difference that the performance of the model has increased and become more consistent. Both training accuracy and validation accuracy are increasing as the dropout rate is increasing. Here the dropout methods is having a obvious positive effect on our model's prediction. The validation accuracy increased about 3 percent by increasing the dropout rate.

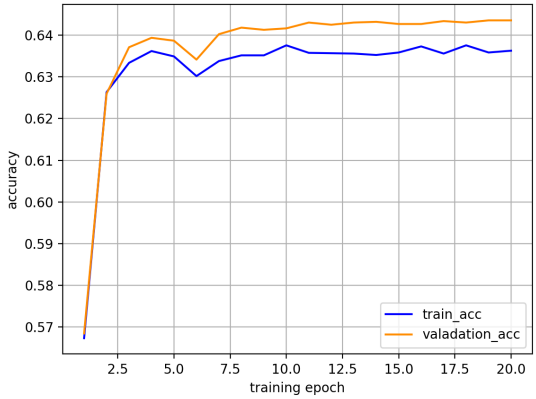

Fig. 7: Accuracy after each epoch

Then change the dropout rate gradually from 0.02 to 0.30 and one more layer to see the change in accuracy.

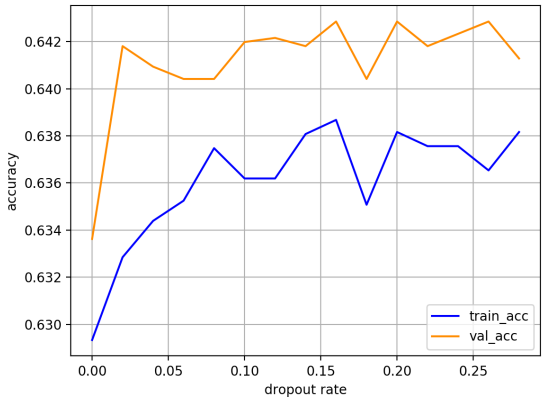

Fig. 8: Accuracy with respect to dropout rate

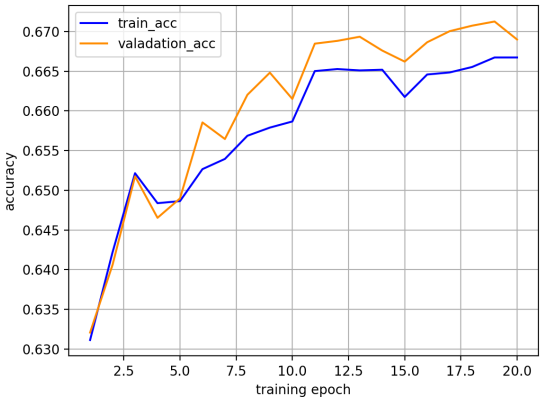

Fig. 11: Accuracy after each epoch

```
At epoch 1
Training loss at epoch 1 is 642.76721171107
Training accuracy at epoch 1 is 0.5590909090909091
Validation accuracy at epoch 1 is 0.5609756097560976
At epoch 2
Training loss at epoch 2 is 620.6904199288964
Training accuracy at epoch 2 is 0.6144939965694682
Validation accuracy at epoch 2 is 0.610801393728223
At epoch 3
Training loss at epoch 3 is 602.8256756113336
Training accuracy at epoch 3 is 0.6209262435677531
Validation accuracy at epoch 3 is 0.6195121951219512
At epoch 4
Training loss at epoch 4 is 591.10909616477725
Training accuracy at epoch 4 is 0.6298456260720412
Validation accuracy at epoch 4 is 0.6275261324041812
At epoch 5
Training loss at epoch 5 is 584.3652453319817
Training accuracy at epoch 5 is 0.6331046312178388
Validation accuracy at epoch 5 is 0.6327526132404181
At epoch 6
Training loss at epoch 6 is 580.0173411559664
Training accuracy at epoch 6 is 0.6347341337907375
Validation accuracy at epoch 6 is 0.6376306620209059
At epoch 7
Training loss at epoch 7 is 577.4103021914773
Training accuracy at epoch 7 is 0.6349056603773585
Validation accuracy at epoch 7 is 0.6383275261324042
At epoch 8
Training loss at epoch 8 is 575.3243941139854
Training accuracy at epoch 8 is 0.6349056603773585
Validation accuracy at epoch 8 is 0.6393728222996515
At epoch 9
Training loss at epoch 9 is 574.0379977467958
Training accuracy at epoch 9 is 0.635934819897084
Validation accuracy at epoch 9 is 0.6405923344947735
At epoch 10
Training loss at epoch 10 is 572.3969066099227
Training accuracy at epoch 10 is 0.6355917667238422
Validation accuracy at epoch 10 is 0.6400696864111498
```

Fig. 9: Epoch 1-10

```
At epoch 11
Training loss at epoch 11 is 5406.2734713231675
Training accuracy at epoch 11 is 0.6570375450025716
Validation accuracy at epoch 11 is 0.663418029933867
At epoch 12
Training loss at epoch 12 is 5402.673222027453
Training accuracy at epoch 12 is 0.6575518601062917
Validation accuracy at epoch 12 is 0.6646362687086669
At epoch 13
Training loss at epoch 13 is 5401.04553220385
Training accuracy at epoch 13 is 0.6579804560260586
Validation accuracy at epoch 13 is 0.6618517229376958
At epoch 14
Training loss at epoch 14 is 5399.128770076376
Training accuracy at epoch 14 is 0.6576375792902451
Validation accuracy at epoch 14 is 0.6623738252697529
At epoch 15
Training loss at epoch 15 is 5395.384484671136
Training accuracy at epoch 15 is 0.6582376135779188
Validation accuracy at epoch 15 is 0.6646362687086669
At epoch 16
Training loss at epoch 16 is 5393.235260532059
Training accuracy at epoch 16 is 0.6572089833704783
Validation accuracy at epoch 16 is 0.6630699617124957
At epoch 17
Training loss at epoch 17 is 5392.6029096666925
Training accuracy at epoch 17 is 0.6572089833704783
Validation accuracy at epoch 17 is 0.6627218934911243
At epoch 18
Training loss at epoch 18 is 5390.426622997871
Training accuracy at epoch 18 is 0.6569518258186182
Validation accuracy at epoch 18 is 0.6616776888270101
At epoch 19
Training loss at epoch 19 is 5390.081564117898
Training accuracy at epoch 19 is 0.6562660723469913
Validation accuracy at epoch 19 is 0.6627218934911243
At epoch 20
Training loss at epoch 20 is 5387.481054669438
Training accuracy at epoch 20 is 0.6583233327618722
Validation accuracy at epoch 20 is 0.6641141663766098
```

Fig. 10: Epoch 11-20

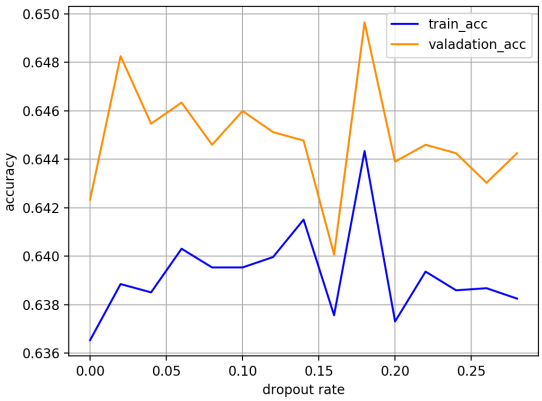

Fig. 12: Accuracy with respect to dropout rate

The accuracy has increased to more than 67 percent which is a notable improvement. But somehow the effects from dropout rate are getting more unpredictable.

IX. Discussion

This bicycle-sharing dataset is hard to predict because of the unpredictability and randomness of bike usage. The attribute are related to the result but not too closely related. There's still bike usage during extreme weather like heavy rain or snow in some urgent situation. The attribute of holiday can be more specific instead of only yes or no. In some holidays, a lot of people might be travelling outside and having less bike users in the city and thus less bike usage. However, in some holidays a lot of locals are still staying in the city or still have to go to work while a lot of travellers are coming to the city. In this situation, the usage of bike will not decrease but instead increase. Both commuters and travellers visiting will bring up the bike usage. There can be new attributes such as how many of bike users are frequent users and how many are not. This can help a lot that some people ride the bike regularly for commuting or exercising. Non-frequent users might ride a bike randomly once a month or three times a week at different time.

X. Conclusion

In this paper, we are focusing on predicting the bike-sharing usage using attributes from outside environment. We standardized the data first for better training. Then we use the data to train our multi-layer BP Neural network model with both forward pass and backward pass to update the weights. The dropout method are also used in our model to randomly drop neurons to improve our model and prevent overfitting. The BP neural network model we are working on is performing better than general neural network model. However, it can still be improved using more machine learning techniques and more related attributes in dataset. The performance has been increasing as we add more layers and dropout technique. We are getting pretty close results for predicting the usage of bike-sharing services as the service provider can make decisions for bike distribution based on our prediction thus provide better service to those who need.