Z:\COMUNICAZIONE\IDENTITA'_VISIVA\Loghi\Loghi Ateneo\eng_nero_uni1r.tif

Department of Information Engineering and Computer Science

Bachelor’s Degree in

Computer Science

final dissertation

Bike sharing usage forecast

*A neural network-based web application to predict bike usage in Washington D.C.*

|  |  |
| --- | --- |
| Supervisor  Maurizio Marchese, Ivan Koychev | Student  Luca Scotton |

Academic year 2017/2018

**Acknowledgements**

*Firstly, I would like to express my sincere gratitude to my advisor Prof. Ivan Koychev for the continuous support for my research.*

*To my Erasmus friends that bore me for these 5 months in Sofia.*

*Finally, thanks to my family for giving me the possibility to study what I love.*

**INDEX**

[**1** **Introduction** 5](#_Toc517605617)

[**2** **Preliminaries** 5](#_Toc517605618)

[**2.1** **The Dataset** 5](#_Toc517605619)

[**2.2** **Content** 5](#_Toc517605620)

[**2.3** **Data analysis** 6](#_Toc517605621)

[**3** **Method** 8](#_Toc517605622)

[**3.1** **Neural Network** 8](#_Toc517605623)

[**3.1.1** **Data preparation** 8](#_Toc517605624)

[**3.1.2** **Model** 9](#_Toc517605625)

[**3.1.3** **Training and Export** 10](#_Toc517605626)

[**3.2** **Web Application** 10](#_Toc517605627)

[**3.2.1** **RESTful API** 10](#_Toc517605628)

[**3.2.2** **Bicycle Up - Landing Page** 12](#_Toc517605629)

[**4** **Results** 13](#_Toc517605630)

[**5** **Conclusions** 14](#_Toc517605631)

[**6** **Bibliography** 14](#_Toc517605632)

**Abstract**

High fidelity predictions on the trend of bicycle’s usage on shared bike services is crucial to keep a quality service for the citizens.

This task explores a dataset of historical data from the bike sharing service in Washington D.C. and produces a neural network model capable of predict the intensity of usage of the service during a time frame from now to the following 5 days.

This regression model outperformed the other more traditional methods, obtaining sensible improvements and high-fidelity predictions.

The final product is a web app capable to provide accurate forecasts in a simple and intuitive way.

1. **Introduction**

Big data analysis is growing more and more thankfully to the increasing amount of data available for research. Lot of cities are moving to a more sustainable condition, for a better living of the citizens and to prevent global issues such air pollution or global warming.

Governments are investing money for new and more eco-friendly ways of transportation in their cities. Washington D.C. had made his choice and installed a shared bicycle system for a green and cheap mobility. This service offers challenging tasks for a data scientist and a lot of insights about society.

The purpose of this thesis is to build a web application capable of producing accurate predictions regarding the use of the bicycles by the citizens.

The starting point is a dataset with a 2 years history of usage of the service with information about weather and time. To get the most from these data I chose to use a neural network.

Indeed, nowadays these techniques are being used in a wide range of applications for instance natural language processing, computer vision or financial applications.

Many researchers showed the efficiency of neural networks in regression problems [1].

I built and trained a TensorFlow’s model using a feedforward deep neural network and a NodeJS server to handle the new data from the RESTful weather API, interface with the trained neural network and display the forecasts in a responsive web app.

This platform can produce valuable information useful for the management of the service to overcome problems regarding high demand of bicycles in the clue hours and optimizing the maintenance of the service.

My aim is to achieve a good compromise between model’s accuracy and computational complexity and demonstrate once again that these new technologies are able to produce compelling results.

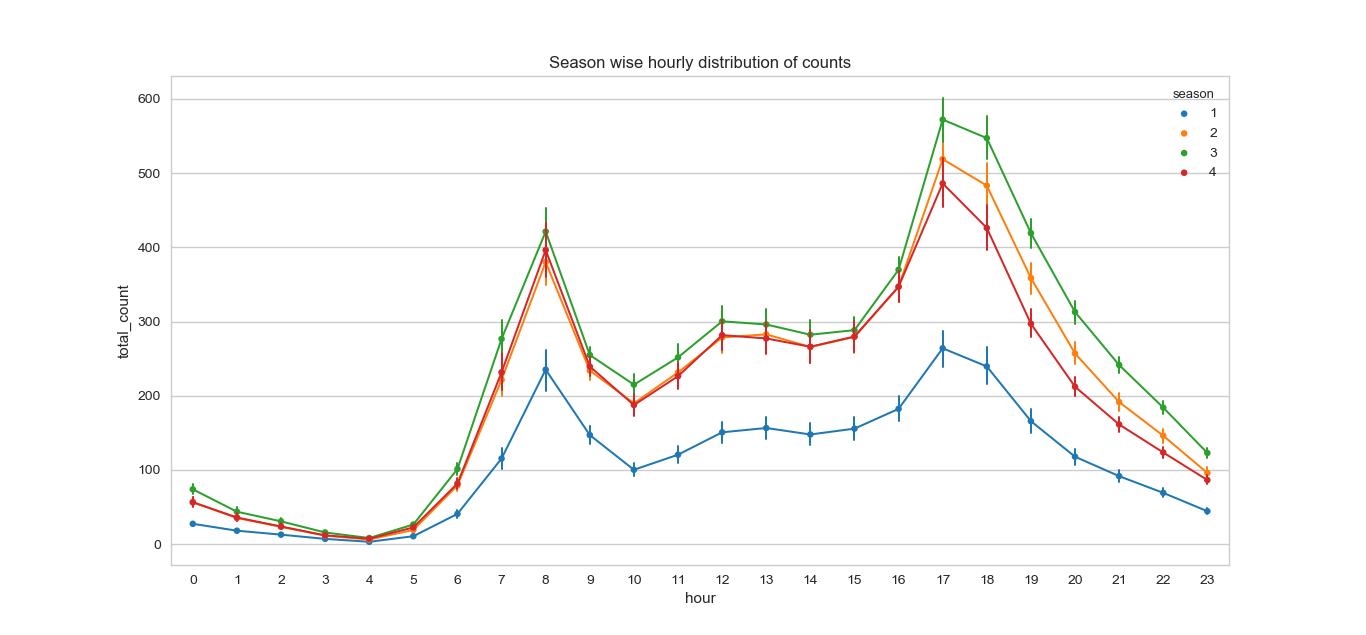
1. **Preliminaries**

## **The Dataset**

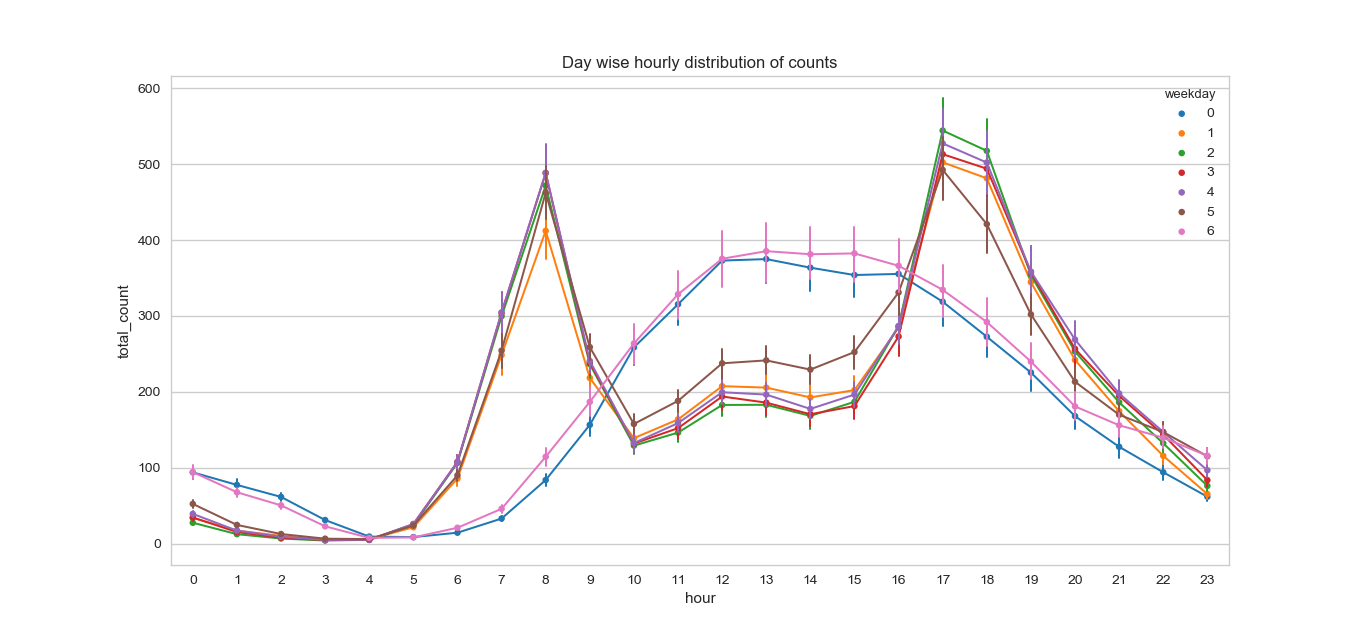
The dataset [2] contains 17379 hourly and daily count of rental bikes between years 2011 and 2012 in Capital bikeshare system in Washington DC with the corresponding weather and seasonal information.

* 1. **Content**
* **instant:** Record index
* **dteday:** Date
* **season:** Season (1: Winter, 2: Spring, 3: Summer, 4: Fall)
* **yr:** Year (0: 2011, 1:2012)
* **mnth:** Month (1 to 12)
* **hr:** Hour (0 to 23)
* **holiday:** weather day is holiday or not
* **weekday:** Day of the week
* **workingday:** If day is neither weekend nor holiday is 1, otherwise is 0.
* **weathersit:** (extracted from Freemeteo)
* 1: Clear, Few clouds, Partly cloudy, Partly cloudy
* 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
* 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
* 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
* **temp:** Normalized temperature in Celsius. The values are derived via (t-t\_min)/(t\_max-t\_min), t\_min=-8, t\_max=+39 (only in hourly scale)
* **atemp:** Normalized feeling temperature in Celsius. The values are derived via (t-t\_min)/(t\_max-t\_min), t\_min=-16, t\_max=+50 (only in hourly scale)
* **hum:** Normalized humidity. The values are divided to 100 (max)
* **windspeed:** Normalized wind speed km/h. The values are divided to 67 (max)
* **casual:** count of casual users
* **registered:** count of registered users
* **cnt:** count of total rental bikes including both casual and registered
  1. **Data analysis**

To better understand the nature of the task I extracted insights form the dataset.

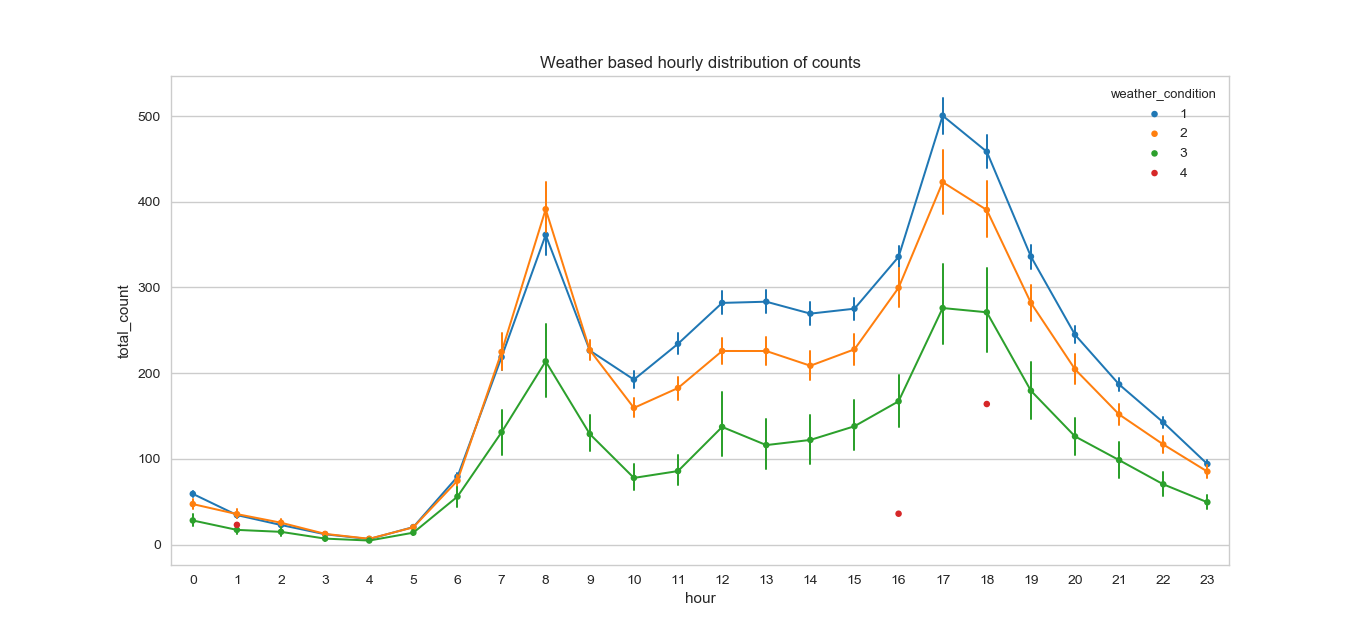


Where: 1: Winter, 2: Spring, 3: Summer, 4: Fall



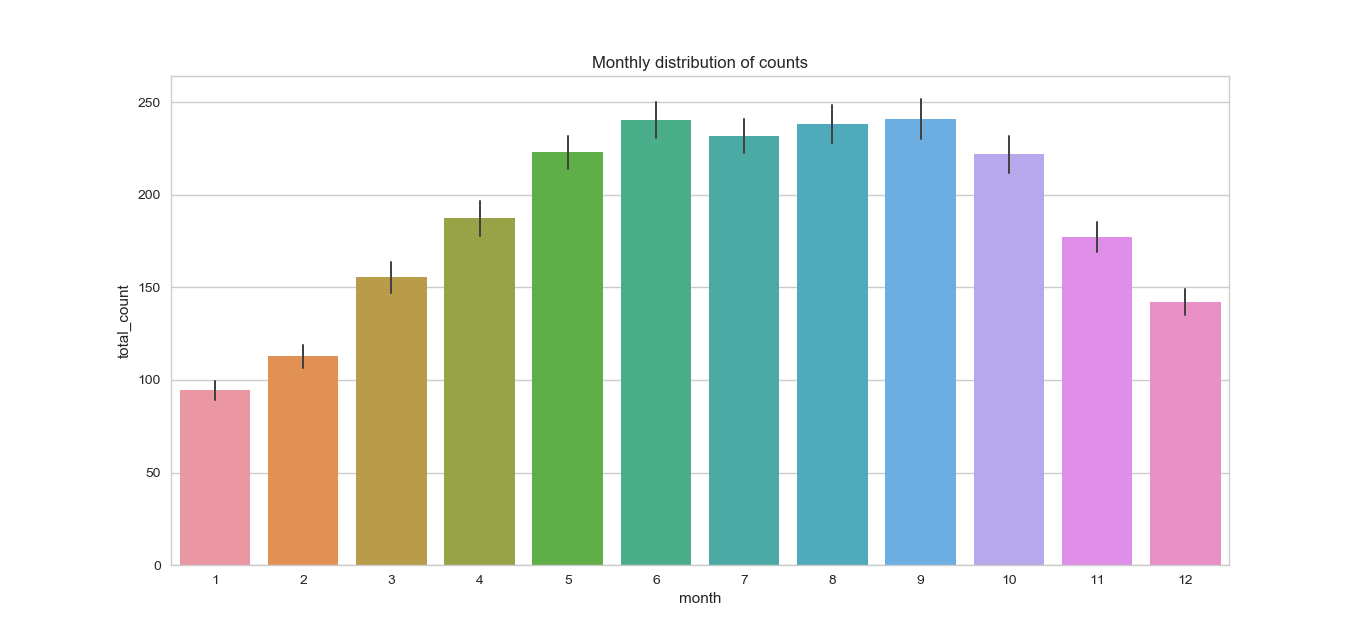
Where: 0: Sunday, 1: Monday, 2: Tuesday, 3: Wednesday, 4: Thursday, 5: Friday, 6: Saturday

It’s clear that the usage of bicycles has a seamless evolution during the working days but on the weekend the behaviour of the customers has a drastic change.



Where: 1: Clear, 2: Cloudy, 3: Rainy, 4: Snow (more details in section 2.2)

Not only the type of the day has an influence but also the weather conditions.



1. **Method**

The project consists in two core parts that interact between each other:

* 1. **Neural Network**

The kernel of the whole project is the neural network model. The model is written in Python using the Tensorflow library over Keras.

### **Data preparation**

First step is importing the dataset by using the ad-hoc function provided by Pandas library.

The second step is to perform feature selection. This process removes extra parameters from the dataset and sometimes adds some new ones. The result will be a meaningful dataset that better represents the problem and facilitates the training of the neural network.

In this case is necessary to drop some extra features present in the dataset: ‘instant’, ‘yr’, ‘atemp’, ‘dteday’, ‘casual’ and ‘registered’

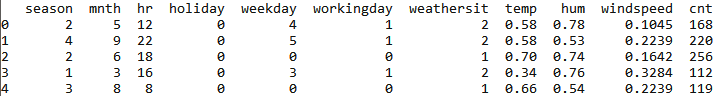


Figure 1 - Extract from the cleaned dataset

It’s common in many datasets to have impurities on the entries.

A quick check of average, minimum and maximum helps to identify the outliers if present.

It’s clear from *Figure 1* that the attributes have different ranges of values. This is not yet suitable for machine learning. A process of normalization is required.

Before applying the transformation to the dataframe it is mandatory to remove any missing value. Since the dataset is chronologically ordered, it’s possible to interpolate missing values without losing quality. Pandas library provides the *interpolate()* function that takes care of it.

The *MinMaxScaler(feature\_range=(-1, 1))* is able to perform the scaling process by applying the function:

*MinMaxScaler* object keeps track of the *min()* and the *max()* of every attribute in order to provide the inverse transformation if required.

Tests showed that the scaling between -1 and 1 is more efficient than the common scaling between 0 and 1.

It is important to notice that ‘cnt’ is willingly excluded from the process because it represents the value to predict.

It’s necessary to export the scaler to keep the scaling factors saved. If not, it’s not possible to import the model from other files and perform a prediction.

The next step is shuffling the entries of the dataset. This process is extremely important to get the best generalization of the problem therefore best performance of the neural network. The reason why I do this is to overcome the exclusion form the training set of rare entries.

It is possible to proceed to the creation of the training and testing sets. The number of entries is over 17000 so it possible to perform a split 90% for train set 10% test set. For each set it is mandatory to separate the class from the attributes. The resulting arrays are 4: X\_train, y\_train, X\_test, y\_test

### **Model**

My task is to predict the total amount of bicycles in use. This is known as non-linear regression problem.

To achieve the best results in this type of scenarios I’m using is a fully connected feedforward deep neural network.

The structure of the network involves a *Sequencial()* wrapper for a 8 layers deep neural network:

* Input layer: Dense layer, 64 neurons
* Dense layer 128 neurons
* Dense layer 128 neurons
* Dense layer 128 neurons
* Dense layer 128 neurons
* Dense layer 64 neurons
* Dense layer 32 neurons
* Dense layer 16 neurons
* Output layer: Dense layer 1 neuron

All the layers have ReLU as activation function.



Figure 2 - Rectified linear unit

It’s a linear function that performs well in deep neural network because prevents the problem of the vanishing gradient (typical of Sigmoid activation function) and have a lower impact on performance due to its simplicity.

The last layer instead has the linear activation function. It’s mandatory to perform the prediction.

### **Training and Export**

Once the model is built it must be compiled. The loss function used on this task is the most popular for regression problems: Mean Squared Error.

The optimizer is RMSprop with its adaptive learning method.

Finally, it’s now possible to train the neural network.

I provided a number of 100 epochs and a batch size of 64, it’s a good compromise between training speed and accuracy of the fitted model.

For further usages of the same model it’s recommended to export the structure and the neurons’ weights of the trained model. This step is essential because it allows a fast deploy of the neural network in business applications.

* 1. **Web Application**

The web application has the purpose to offer both a RESTful API service and a landing page that displays the forecasts.

The web server is implemented in NodeJS, a run-time server environment in JavaScript, with the inclusion of the Express library.

### **RESTful API**

The prediction core functionalities can be queried by the path /predict of the web server. The server accepts POST type requests only containing the json structure in the body.

The json query structure has to be as the following:

1. {
2. "list": [{
3. "dt": 1529668800,
4. "main": {
5. "temp": 294.53,
6. "humidity": 89,
7. },
8. "weather": [{
9. "main": "Rain",
10. "description": "light rain",
11. }],
12. "wind": {
13. "speed": 4.92,
14. "deg": 110.504
15. },
16. "dt\_txt": "2018-06-22 12:00:00"
17. }, {
18. "dt": 1529679600,
19. "main": {
20. "temp": 295.56,
21. "humidity": 88,
22. },
23. "weather": [{
24. "main": "Rain",
25. "description": "light rain",
26. }],
27. "wind": {
28. "speed": 5.91,
29. "deg": 100.502
30. },
31. "dt\_txt": "2018-06-22 15:00:00"
32. }]
33. }

The ‘list’ field contains all the predictions to be made. For each entry it’s mandatory to specify all the information about time and weather conditions as described in the example.

The web server will send as response a json list:

1. {
2. "pred": [123, 248]
3. }

The array ‘pred’ contains the predictions for each of the entries provided as input keeping the same order.

In order to produce predictions, the back-end side parse the json and print to a csv file all the clean entries submitted.

Successively, the python script to make the actual prediction is called as child process of the server passing the file name of the dataset created before.

The script parses the dataset row by row and extract the time features from the datestamp. As in the description of the source dataset, it’s mandatory to apply pre-normalization to the other attributes such temperature, humidity and wind speed.

The script also parses the nominal form of the weather conditions and substitutes it with the appropriate class. The algorithm that is applied is a hierarchical set of rules based on the content of the string.

As soon as all the features are collected it’s necessary to import the scaler exported during the initial training step. The scaler applies the same transformation applied in the training set to be coherent.

Then the trained model is imported and compiled. Through the function *predict()* the dataset is processed from the neural network and the array of predicted values is given as output and printed.

The server perceives the end of the child process and catches the output of the printing function.

To conclude the process the array of results is inserted in the output json structure and sent back as response to the call.

### **Bicycle Up - Landing Page**

Bicycle Up is the single page web application that allows the user to see and predict the number of bicycles in use in Washington D.C. up to 5 days from the current moment.

The design of the page is minimalistic to enhance the user experience. The tools implied in the creation are the Html 5 standard and Bootstrap 4 as css framework.

The page supplies dynamic content through external Ajax GET request to the RESTful API of openweathermap.com [3]. A json containing the weather conditions and the forecasts are sent back and the content of the page is updated with the latest conditions.

Afterwards, a POST call with the previous json object attached is sent to our server on the /predict path to get the expected bicycle usage. For further details see section 3.2.1.

When the prediction has been retrieved as json object, the content of the page is updated by displaying the current number and the interactive chart of the following 5 days.

The chart is generated through the Chart.js component. The labels are obtained from the first json object and the values are from the second one.

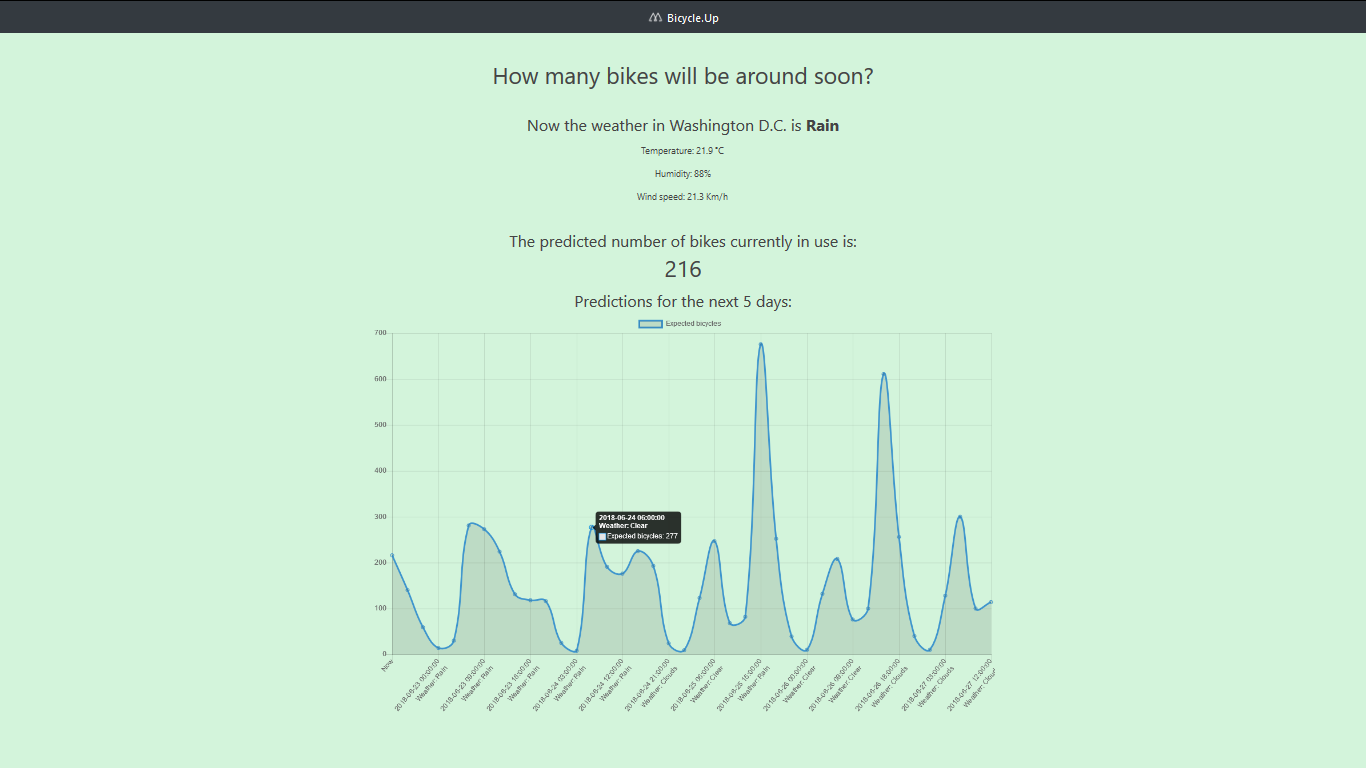


Figure 3 - Screenshot of the inteface

1. **Results**

The training of the model has gone smoothly. The loss function decreased until reaching an asymptote. On the other hand, the accuracy is growing following a logarithmic scale up until 0.024. Further improvements would be more time consuming.

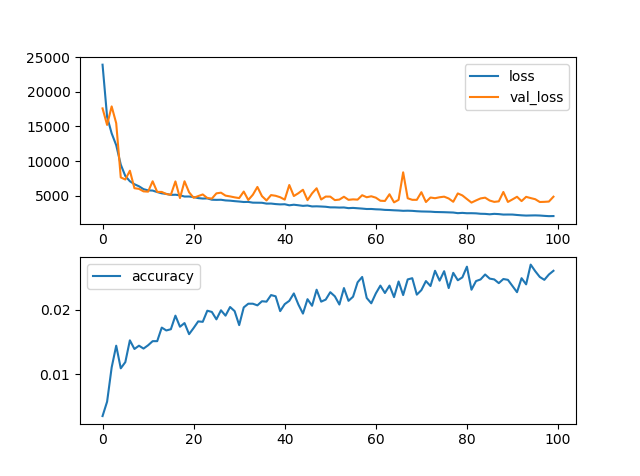


Figure 4 - Metrics of network's training

Subsequently the evaluation of the model with the test set shows a good fidelity of the model. Table 1 describes extra error metrics that keep consistency with the obtained results.

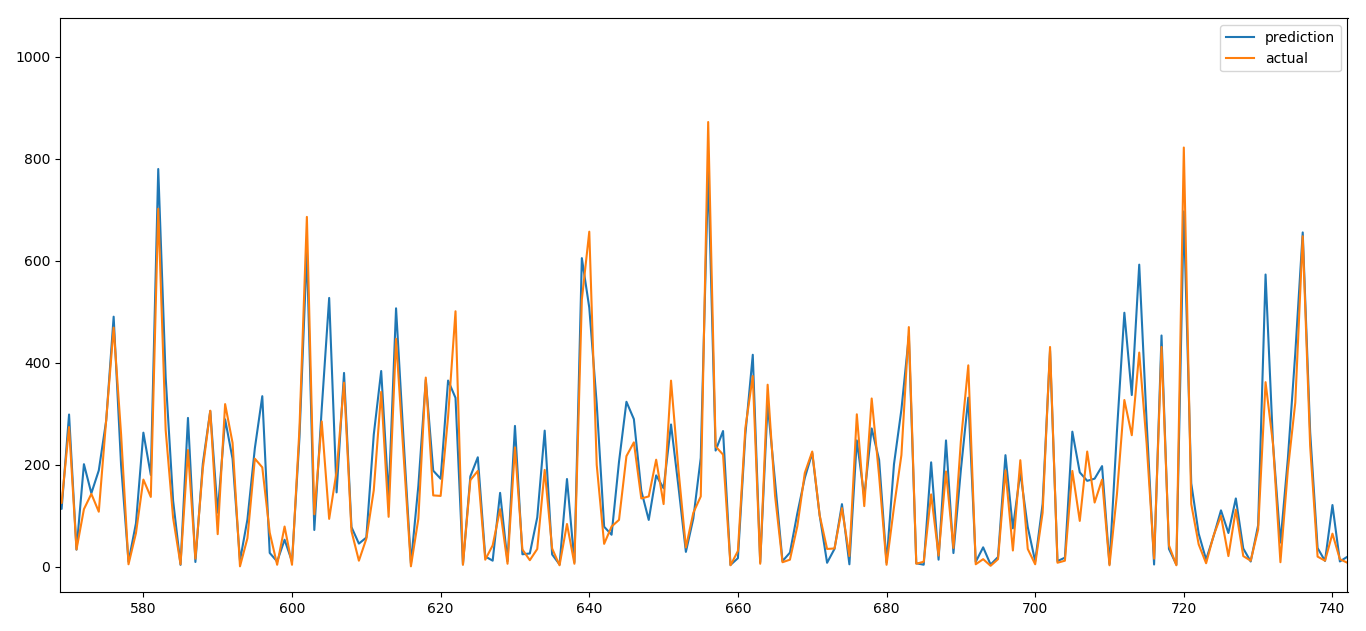


Figure 5 - Test set: predicted and expected values

Table 1- Comparison of error scores of different algorithms (lower is better)

|  |  |  |
| --- | --- | --- |
| Algorithm | Mean Absolute Error | Root Mean Squared Error |
| REPtree | 57.644 (+32.06%) | 86.933 (+24.83%) |
| M5 Rules | 49.779 (+14.04%) | 74.972 (+7.65%) |
| M5 Tree Model | 47.582 (+9.01%) | 71.755 (+3.03%) |
| My Neural Network | **43.651** | **69.643** |

Further tests have been performed on the same dataset with other machine learning techniques such decision trees and rule-based models. The scores are listed in Table 1. The neural network model I set-up outperformed better than the other machine learning techniques for regression on the same dataset such. The deltas between the scores are quite significant especially in the absolute error, indeed it varies of a 9% less on the second-best model.

1. **Conclusions**

Form the results I got, applying neural network has been effective to tackle the problem. The significant improvements made compared to the other algorithms state that my solution can be a valuable resource to deal also with regression problems similar to this. In general, this web app is a success under many aspects.

First, it offers in depth predictions in a valid response rate, quick enough to take actions in a possible future emergency scenario (i.e. low quantity of bicycles available).

Secondly, it remarks the flexibility and ease of interaction of two of the most trending programming languages of the last years.

This work gave me the possibility to test my knowledge in the big data field. I mastered several skills on data processing and data analysis. I consider helpful diving into the web app part to learn how these two trending branches of computer science can be interconnected in a single project. Moreover, this task leads me to produce factual results with a meaningful use in a business environment.

This work can also be a starting point for many other applications in which having an accurate view on what is going on in the future is crucial. Some examples can range from the monitoring of air pollution to the analysis of bus/metro/highway accesses.

# **Bibliography**

|  |  |
| --- | --- |
| [1] | D. Specht, «A general regression neural network,» *https://ieeexplore.ieee.org/document/97934/,* 1991. |
| [2] | M. Kaghazgarian, «Bike Sharing Dataset,» 2018. [Online]. Available: https://www.kaggle.com/marklvl/bike-sharing-dataset. |
| [3] | openweathermap.org, «Weather API,» [Online]. Available: https://openweathermap.org/api. |