An interactive web tool for understanding the vehicle behaviour

FINAL PROJECT REPORT

HONGYANG LIU

Supervisors: Yuantao Fan , Reza Khoshkangini

May 26 , 2019

Abstract

The web interface is a platform for plotting the vehicle onboard sensor data streamss. This web would also assist data scientists or fleet operators in understanding the vehicle's operations and abnormal or normal behaviours through the system analysing tool. Specifically, we plot the sensor data using an analytical web tool which combines python web framework and the supervised learning algorithms.The interface has the functions which could be utilised for users to analyse the sensor data streams, e.g. they could use calendar or sliders to put dates, they could use the web tool to select dropdown menu and button to put different types of signals with time series and then explore the track on the GPS map. Besides, because of the use of the productive machine learning library and the Python framework named Dash, the analytical tool is beneficial for experts or students to detect outliers or unusual behaviours of the vehicle via sensor data such as engine speed, engine temperature. Therefore, the web interface could improve users' efficiency on helping them understand the vehicle's operations; detecting the normal or abnormal vehicle's behaviours; predicting the need for maintenance. We also find there are some limitations that we need to improve on this project.

Table of Contents

1. Introduction

The project is designed for helping data scientists or fleet operators to understand the behaviour of vehicles. The number of vehicle sensor data streams is massive. It is difficult for experts and data scientists to find an easy and fast way to understand the behaviour of the vehicle just through the data streams. To handle these problems and improve their efficiency, Our project makes the visualisation tool which plots the data streams on the interface. Besides, we also combine the web interface with supervised learning algorithms, e.g. KNN, SVM and Neural Network. The combination of two different fields would help data scientists or fleet operators spend less time and energy on understanding the behaviours of the vehicle. They could also select one part of the data to analyse or detect anomalies or faults.

1.1 Problem formulation

The general objective of this project is firstly to make an interactive web tool for data scientists or fleet operators. This tool would contribute to them more easily to understand the vehicle's normal behaviours, e.g. acceleration, deceleration, turning right or left. To understand these behaviours, they could explore the different types of bus signals like vehicle speed, selected gear and GPS, etc. However, it is difficult for users who are lack of professional knowledge about the vehicles to understand the behaviours of the vehicle, let alone find outliers or anomalies. Therefore, we make a visualisation tool so that it could assist them in understanding the vehicle operations. In addition, we combine the interface with supervised learning to do data analysis. It would provide support for data scientists or fleet operators and reduce the complexity of the data analysis.

The project still has its limitations:

One feature data In the visualisation tool, users could only plot one signal once time. The reason is that the onboard sensor are not simultaneously detecting the vehicle's operations and time series are different. Therefore, this means users could not plot the multi-feature data on the system. Actually, users could only visualise one feature data and label the only feature once time.

Sufficient data In the task of classifying, we mainly use binary classifier. It means we classify the data into two groups. In the project we would classify the data belongs to normal or abnormal. To achieve the goal, we need sufficient data in both groups. If we don't do like that, when we use the classification algorithms and cross-validation method, the algorithms would throw an error.

System response time If the number of input data is massive, the interface would need more time to handle this data to plot it. The tables show that the number of different features' signal data(Figure 1). We could see if users decide to plot the whole number of signal data once time, the system would crush because of the massive data. However, if users could filter the data and plot a limited number of them, this way would improve the system response speed.

Features	Number of data
Fuel Rate	8750663
Selected gear	8026487
Vehicle speed	8855860
Engine speed	8856561
Engine Fuel Temperature	8717588

Figure 1 the number of data

Exception and errors problems Because of the web system interface using Dash and Plotly. They are Python visualisation tool with no javascript. Consequently, it is difficult to remind users in the front-end if there are some errors happened.

1.2 Background Information

The project aims at helping data scientists or fleet operators to understand the vehicle's behaviour and analyse the sensor data to detect normal or abnormal behaviour when the vehicle runs on the road. The cumulative vehicle signal data streams could be used by companies to train the system model and find mechanical problems through data analysis. This work would make the needs for maintenance more exactly and improve the efficiency of the vehicle system. Making an interactive tool should have these functionality: comparing various signal features and locating the position of the vehicle; providing users fundamental data analysis tool; applying the model on other untrained data model to do prediction. Besides, the web interactive tool should be considered to be easy, reliable and simple for users.

1.3 Related work

The project is related to the visualisation tool and supervised learning to detect abnormal hehaviour. The visualisation tool is made of the python web framework. However, in this project, we customise the framework to meet our requirements that the tool could be used for plotting the sensor data streams; labelling the data according to users' needs and detecting vehicle's anomalies;

The related work regarding using the visualisation tool is Plotly and Dash. Jeff Lai(2017) used Plotly to visualise deep learning instead of matplotlib which is also a python plotting library. He found the tool was easy-use and interactive[1].

In data analysis and anomaly detection, Stacey Ronaghn(2018) wrote an article regarding a way to detect outliers or unusual behaviour via several machine learning algorithms. Her objective was to introduce the anomaly detection techniques which includes: The use-cases were detecting abnormal behaviour of equipment in a manufacturing plant using data such as temperature, pressure and humidity; The most common anomaly detection algorithms were for anomaly detection: K-Means, One-class Support Vector Machines, and Autoencoders[2].

Consequently, Their works illustrated that analytic interface could be combined with machine learning algorithms to make an engineering tool. This tool could be used for data visualisation and data analysis to achieve anomaly detection techniques.

2. Methods

The motivation of the project is that data scientists or fleet operators could see the results of the vehicle's operations using the data visualisation tool and assist them in understanding the behaviour of the vehicle. This work would also help them analyse the data to detect outliers or anomalies. There are two sections regarding the project work: The first section is the visualisation tool. It has four tags including data visualisation, manual labelling, modelling and evaluation and automatic labelling. The second section is concerning data analysis. In this project, we use python library: scikit-learn to achieve the goal. Scikit-learn is a machine learning library in python language. Because the visualisation tool is also a python web framework, we could combine them to build the web interface easily.

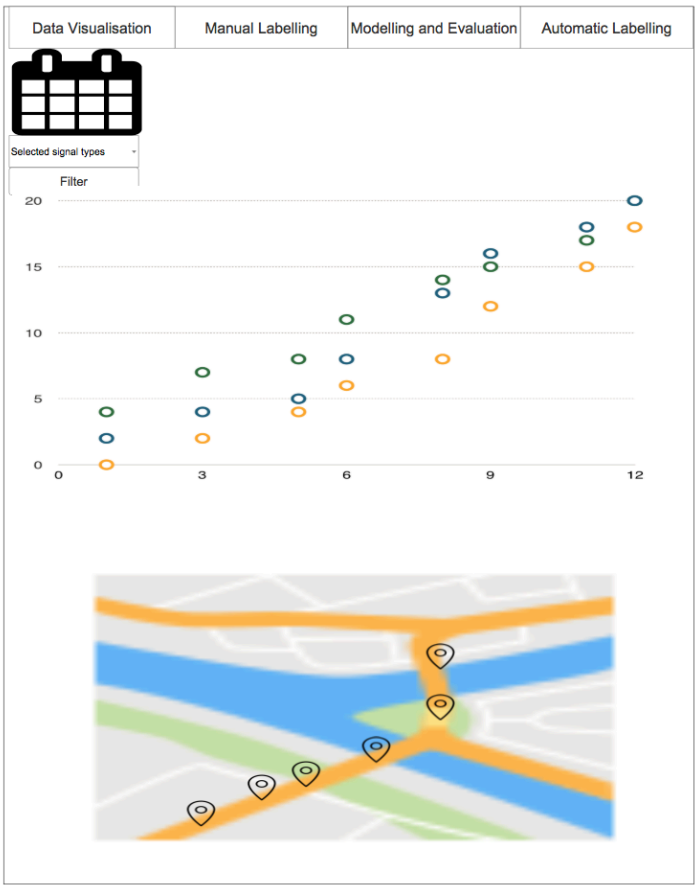
2.1 The web interface.

The web interface was designed to have several functionalities: visualising data, labelling data, evaluating the model and applying the algorithms. The figures are the prototype of the web interface. Every figure has the same tab to switch the different functionalities.

When users click the first tab: data visualisation, they could plot the sensor data streams in a certain duration selected by the calendar components. They could also choose the data signals

like engine speed or fuel temperature and filter the number of data through some components like calendar, dropdown list and slider bar. In figure 2, The main diagram would be plotted immediately via scatter plots. In the diagram, the x-axis should represent the time series. The y-axis should represent the sensor data value. The GPS sensors contain longitude value, latitude value and time series, it is different from other types of signal data which have two values. Therefore, we plotted the trace of the signal data on the map individually in the back-end.

The second tab is manual labelling. If users are likely to do data analysis. They could use the box select tool on the visualisation tag to select a part of segments of data scatters to label data on the labelling tab. Therefore, users select data on the data visualisation tab. The data would be shown through scatter plots on manual labelling tab(Figure 3). The manual labelling tab has two diagrams. One diagram is scatter plots and another one is data table. The scatter plots are used for shown the selected data from the data visualisation tab. The data table on the labelling tab has three columns: the first column named timestamp, the second column named data and the last one named label which should be input by users manually one by one. In addition, the data table should be flexible. This means users could add, delete, update the rows or columns of the table.



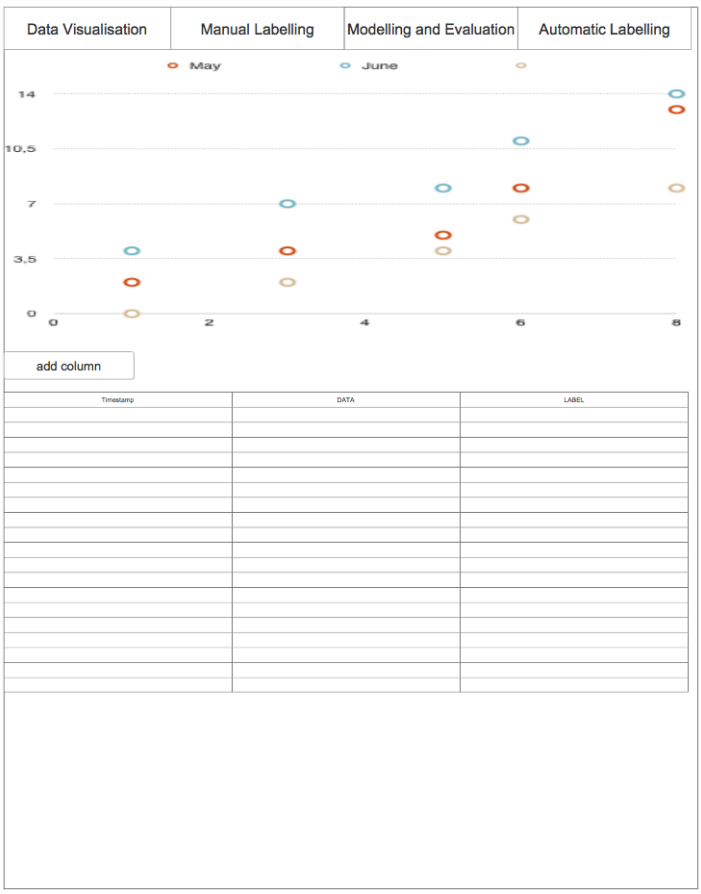


Figure 2 the prototype of data visualisation

Figure 3 the prototype of manual visualisation

The third tab called modelling and evaluation(Figure 4). It has two parts. The first part is to use machine learning classification algorithms like KNN, SVM and Decision Tree to classify the data from the data table on labelling tab. There are three columns on the data table. The data in the first two columns, timestamp and data, would be taken as training data while data in the last one column, label, would be taken as testing data. The back-end algorithms on the system would handle these data and give the evaluations of the performance of the algorithms.

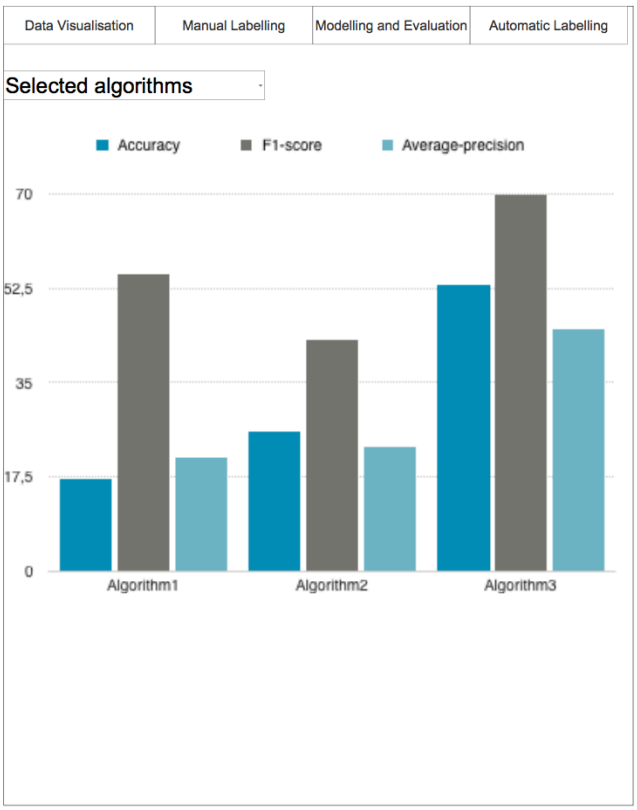


Figure 4 the prototype of modelling and evaluation

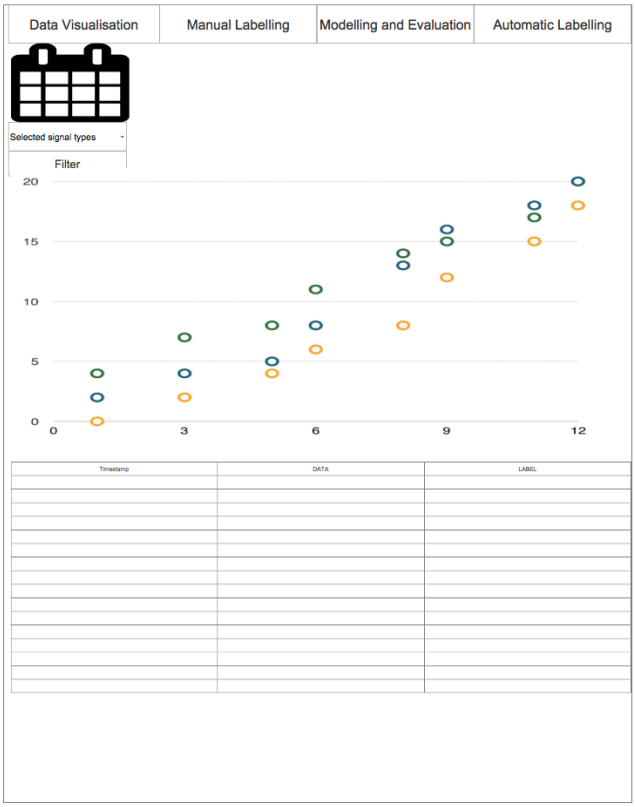


Figure 5 the prototype of automatic labelling

The fourth tab named automatic labelling(Figure 5). In this tab, users could select training data to do prediction. Users could apply the best performance algorithm on the training data. After training, the system would generate the results to plot on the table. Because of the results are numerical, these data would be plotted as scatter plots diagram: the x-axis value is the timestamp and the y-axis value is the prediction results. Moreover, the system could also calculate the proportion of the results. If the dots are anomalies, users could see the percentage they account for.

Visualisation tool: Dash. To implement the prototypes showed on figures, we use the visualisation tool, Dash, to achieve the functionalities above. Dash is a open source Python library for creative, web-based applications. Through a couple of simple patterns, Dash abstracts away all of the technologies and protocols that are required to build an interactive web-based application which could be shared through URL. The other reason to use it is that we decide to use scikit-learn which is a python machine library. Therefore, when we combine the Dash and the scikit-learn, the web system could be improved to achieve our objective.

2.2 Supervised learning

In this project, users should use the algorithms to help them detect normal or abnormal vehicle behaviour. In the beginning, we found that the most important part of solving a machine learning problem was to find a right estimator for the work. As there were different types of signals and operations labelled by users, users needed to choose the best estimators according to the problem. We learn from the official specification on scikit-learn to resolve the problem[3].

In this work, because we just need to apply the supervised learning to train the data and achieve our goals, we designed the data analysis part according to the flowchart below(Figure 6). The number of input data should be sufficient so that the system could train the data exactly and precisely to prevent underfitting, and then we would manually label the data. Next, we would apply the supervised learning classification algorithms to classify these data. After that, users could select algorithms to predict a category. The system would evaluate the results from the

measurement and validate machine learning model. Finally, the users could apply the best algorithms to whole features. The flowchart:

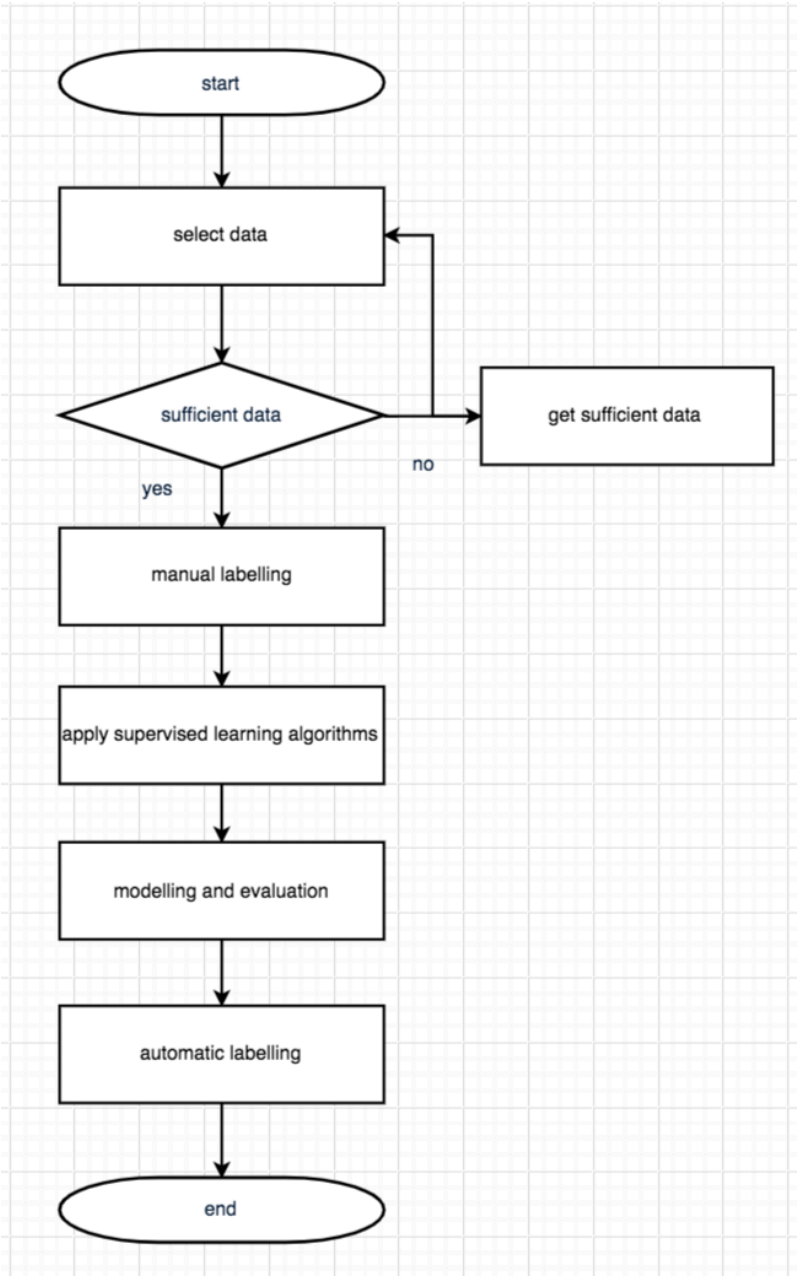


Figure 6 the flowchart of procedures

In data analysis, the algorithms would classify the data to find usual or unusual data according to the features selected by users. The project mainly uses supervised learning for classification and prediction.

To implement it, the basic classification function:

$$y = h_p(x)$$

•**Learning**: given a training set of labelled examples, e.g. $\{(x_1,y_1),(x_2,y_2),(x_3,y_3),(x_4,y_4) \dots, (x_n, y_n)\}$ and estimate to the parameters **p** of the prediction function **h**

•**Prediction & Application**:apply **h** to a never before seen test example **x** and output the predicted value $y = f(x)$

There are several classification in the project including: Nearest Neighbours, Decision Tree, Support Vector Machines and Neural Network, e.g.

K-nearest neighbours algorithm. The k-nearest neighbours algorithms is a type of instance-based learning or non-generalising learning. it does not attempt to construct a general internal model, but simply stores instances of the training data[4].

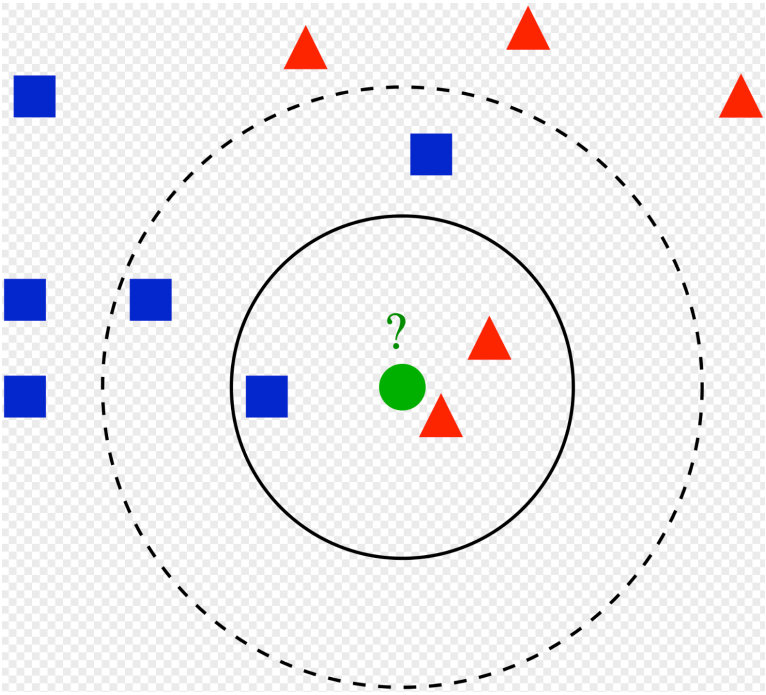


Figure 7 k-NN classification

Example of k-NN classification(Figure 7). The test sample (green dot) should be classified either to blue squares or to red triangles. If k = 3 (solid line circle) it is assigned to the red triangles because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the blue squares (3 squares vs. 2 triangles inside the outer circle)[5].

Decision tree algorithm. Decision trees are trees that classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can assume[6].

Support Vector Machine.A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane[7].

Neural Network. A neural network, when used for classification, is typically a collection of neuron-like processing units with weighted connections between the units. Back propagation is a neural network learning algorithm. Neural network learning is also referred to as connectionist learning due to the connections between units[6].

These whole algorithms would be applied on the project. However, users could choose one or many from these algorithms to they apply. After using algorithms to train data, we use three metrics to evaluate, e.g. accuracy, f1-score, average-precision. These three metrics would help the users to evaluate the performance of the algorithms. During modelling and evaluation, the results should be evaluated several times to calculate the mean value. In scikit-learn library, there is a method named cross validation to achieve the work.

Cross Validation Cross-Validation is the method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model[6].

To better understand the metric methods, the confusion matrix(Figure 8)and the equation was given[8].

	Negative (predicted)	Positive (predicted)
Negative (actual)	true negative	false positive
Positive (actual)	false negative	true positive

Figure 8 Confusion Matrix

equations:

$$accuracy = \frac{truepositives + truenegatives}{totalexamples}$$

$$precision = \frac{truepositives}{truepositives + falsepositives}$$

$$recall = \frac{truepositives}{truepositives + falsenegatioves}$$

$$F1Score = \frac{2 * precision * recall}{precision + recall}$$

Scikit-learn. In the supervised learning, we mainly use the python open source tool, scikit-learn, to accomplish the algorithm code part. Scikit-learn is a free software machine learning library for data mining and data analysis. Besides, because of written in Python language, it could be compatible with the front-end interface library: Dash. It also has the measurements of the algorithms, so it could be used for evaluating the performance of the system. These measurements would help the users to choose the best methods to apply in the labelling automatically tab on the interface. contribution

2.3 Code for interactive components

Because of the engineering work, we would introduce the core interactive components for the project. The core components were written by the Python web framework Plotly and Dash.

Firstly we use the callbacks to interactive between components and interface: Whenever an input component's property changes, callbacks would also be chained, allowing one update to trigger several updates across the app[9].

```
app.layout = html.Div([
    dcc.Input(id='my-id', value='initial value', type='text'),
    html.Div(id='my-div')
])


@app.callback(
    Output(component_id='my-div', component_property='children'),
    [Input(component_id='my-id', component_property='value')]
)
def update_output_div(input_value):
    return 'You\'ve entered "{}"'.format(input_value)
```

Figure 9 callbacks

Callback(Figure 9)

$id =' my - id'$ means the input component

$id =' my - div'$ means the output component

The data would be input through the input component and output results would be output in output component.

Data visualisation To design the interface named data visualisation, we would use these core components: date picker range(Figure 10), dropdown list(Figure 11), button(Figure 12), range slider(Figure 13), scatter plot(Figure 14), map(Figure 15)

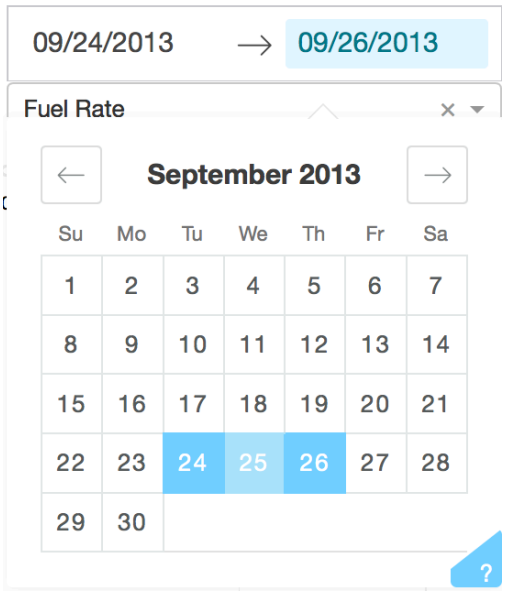


Figure 10 Date picker range

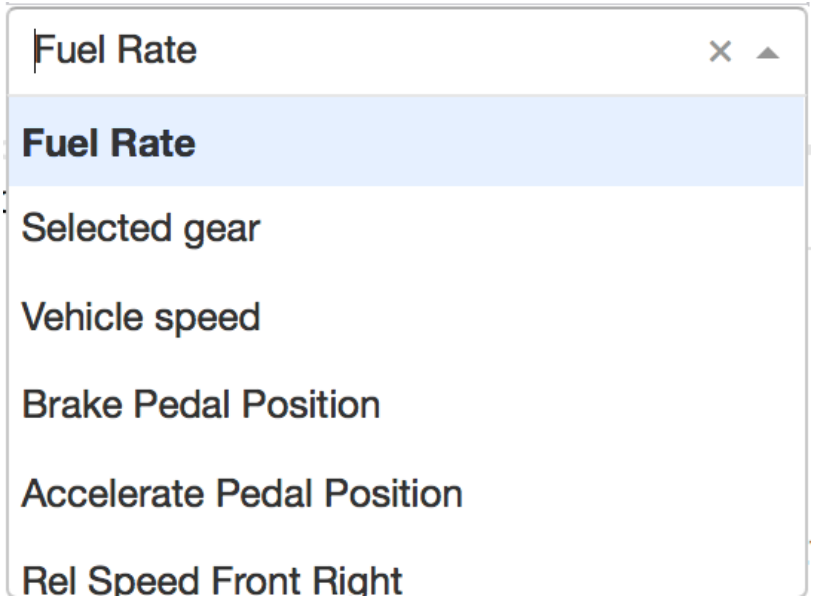


Figure 11 dropdown list

Figure 12 Button

Figure 13 range slider

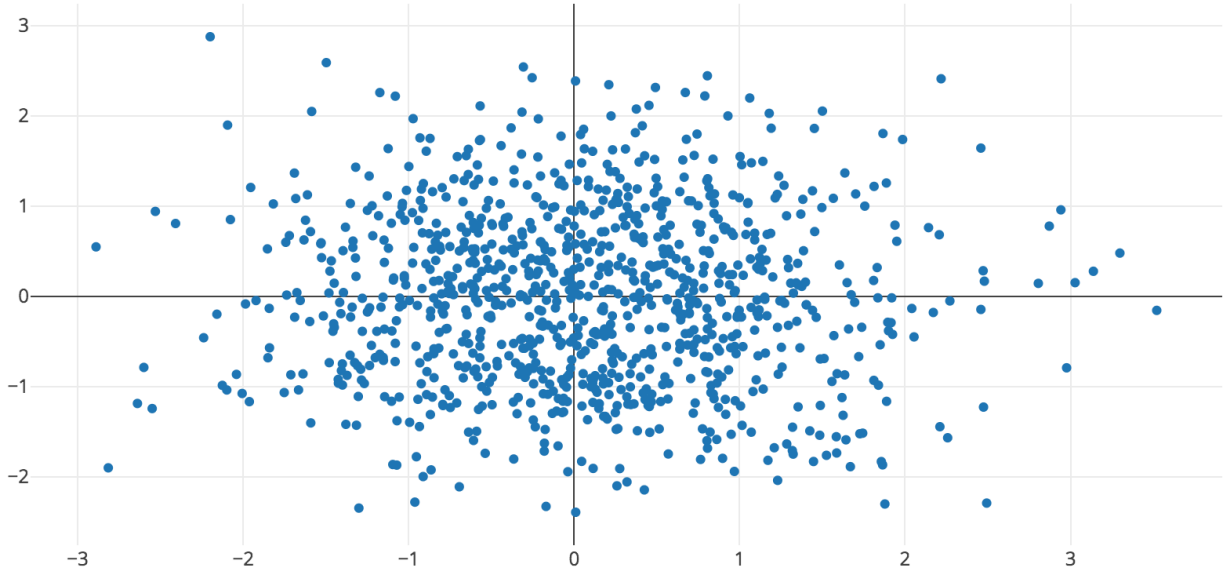


Figure 14 scatter plot

Figure 15 map

There are core Python code to introduce how to implement them.

Figure 16 shows how to implement the date picker range.

$id = 'my-date-picker-range'$ shows the id of the date picker range component.

set the start date and end date:

$start\_date = dt(2013,9,24)$

$end\_date = dt(2013,9,25)$

The date picker range component used for users to select the date range to put data.(Figure 16)

```
dcc.DatePickerRange(
    id='my-date-picker-range',
    min_date_allowed=dt(1995, 8, 5),
    max_date_allowed=dt(2017, 9, 19),
    initial_visible_month=dt(2013, 9, 24),
    start_date=dt(2013, 9, 24),
    end_date=dt(2013, 9, 25)
),
```

Figure 16 code for date picker range

In dropdown menu(Figure 17), We could see the different labels and defaulted value $'FueRate'$.

$id = 'my-dropdown'$ shows the id of the dropdown menu.

```
html.Div([
    dcc.Dropdown(
        id='my-dropdown',
        options=[
            {'label': 'Fuel Rate', 'value': 'Fuel Rate'},
            {'label': 'Selected gear', 'value': 'Selected gear'},
            {'label': 'Vehicle speed', 'value': 'Vehicle speed'},
            {'label': 'Brake Pedal Position', 'value': 'Brake Pedal Position'},
            {'label': 'Accelerate Pedal Position', 'value': 'Accelerate Pedal Position'},
            {'label': 'Rel Speed Front Right', 'value': 'Rel Speed Front Right'},
            {'label': 'Rel Speed Front Left', 'value': 'Rel Speed Front Left'},
            {'label': 'Engine Fuel Temperature', 'value': 'Engine Fuel Temperature'},
            {'label': 'Engine Speed', 'value': 'Engine Speed'}

        ],
        value='Fuel Rate'
    )

], style={'width': 318, 'height': 50}),
```

Figure 17 code for dropdown list

In submit button, we just need to claim the button id $'FilterButton'$ and reset the n_clicks equal to 0 to record the click times.

$id =' submit - button'$ represents the submit button(Figure 18).

```
html.Button('Filter Button', id='submit-button', n_clicks=0),
```

Figure 18 code for button

In range slider, the $'min'$ means the minimal value. The $'max'$ means the maximal value. We use marks to represent the value(Figure 19).

```
dcc.RangeSlider(
    id='application-my-range-slider',
    min=0,
    max=240,
    step=1,
    value=[50, 60],
    marks={
        0: {'label': '0', 'style': {'color': '#000000'}},
        10: {'label': '1/24', 'style': {'color': '#000000'}},
        20: {'label': '2/24', 'style': {'color': '#000000'}},
        30: {'label': '3/24', 'style': {'color': '#000000'}},
        40: {'label': '4/24', 'style': {'color': '#000000'}},
        50: {'label': '5/24', 'style': {'color': '#000000'}},
        60: {'label': '6/24', 'style': {'color': '#000000'}},
        70: {'label': '7/24', 'style': {'color': '#000000'}},
        80: {'label': '8/24', 'style': {'color': '#000000'}},
        90: {'label': '9/24', 'style': {'color': '#000000'}},
        100: {'label': '10/24', 'style': {'color': '#000000'}},
        110: {'label': '11/24', 'style': {'color': '#000000'}},
        120: {'label': '12/24', 'style': {'color': '#000000'}},
        130: {'label': '13/24', 'style': {'color': '#000000'}},
        140: {'label': '14/24', 'style': {'color': '#000000'}},
        150: {'label': '15/24', 'style': {'color': '#000000'}},
        160: {'label': '16/24', 'style': {'color': '#000000'}},
        170: {'label': '17/24', 'style': {'color': '#000000'}},
        180: {'label': '18/24', 'style': {'color': '#000000'}},
        190: {'label': '19/24', 'style': {'color': '#000000'}},
        200: {'label': '20/24', 'style': {'color': '#000000'}},
        210: {'label': '21/24', 'style': {'color': '#000000'}},
        220: {'label': '22/24', 'style': {'color': '#000000'}},
        230: {'label': '23/24', 'style': {'color': '#000000'}},
        240: {'label': '1', 'style': {'color': '#000000'}}
    }
),
```

Figure 19 range slider

The scatter plot has $x-axis$ and $y-axis$. The data type is tuple in Python. The number of $x-axis$ equals to the number of $y-axis$(Figure 20).

```
data = []
trace_close = go.Scatter(
    x=xaxis,
    y=yaxis,
    mode='markers',
    marker=dict(
        color='rgb(255, 0, 0)',
        size=15
    )
)

data.append(trace_close)

layout = {"title": 'selected scatter',
          "clickmode": 'event+select'
          }

return {
    'data': data,
    'layout': layout
}
```

Figure 20 code for scatter plot

Next, the map need *mapbox* access token to show on the web interface. We apply for the token on the website named *mapbox*. In the code(Figure 21), the input data has four variables: *'title'*, '*londgitude', 'latitude'* and *'currentTimeStamp'*. To plot the position of the vehicle. We need its GPS signal including longitude and latitude. Therefore, we assign the two variables to *'lat'* and *'lon'*.

```
def map(title, longgitude, latitude, currentTimeStamp):
    data = [
        go.Scattermapbox(
            lat=longgitude,
            lon=latitude,
            mode='markers',
            text=currentTimeStamp,
            marker=dict(
                color='rgb(255, 0, 0)',
                size=9
            ),
        )
    ]

    layout = go.Layout(
        title=title,
        autosize=True,
        hovermode='closest',
        mapbox=dict(
            bearing=0,
            accesstoken=mapbox_access_token,
            center=dict(
                lat=38.92,
                lon=-77.07
            ),
            pitch=0,
            zoom=0

        ),
    )

    return {
        'data': data,
        'layout': layout
    }
```

Figure 21 code for map

Manual labelling Next, we design manual labelling. The manual labelling has two core components: one component is the plot scatter and another is a data table(Figure 22). The code in plot scatter is the same as what has introduced before while the data table is different. The data table is editable. It just likes spreadsheet and can be used as input for controlling models with a variable number of inputs.

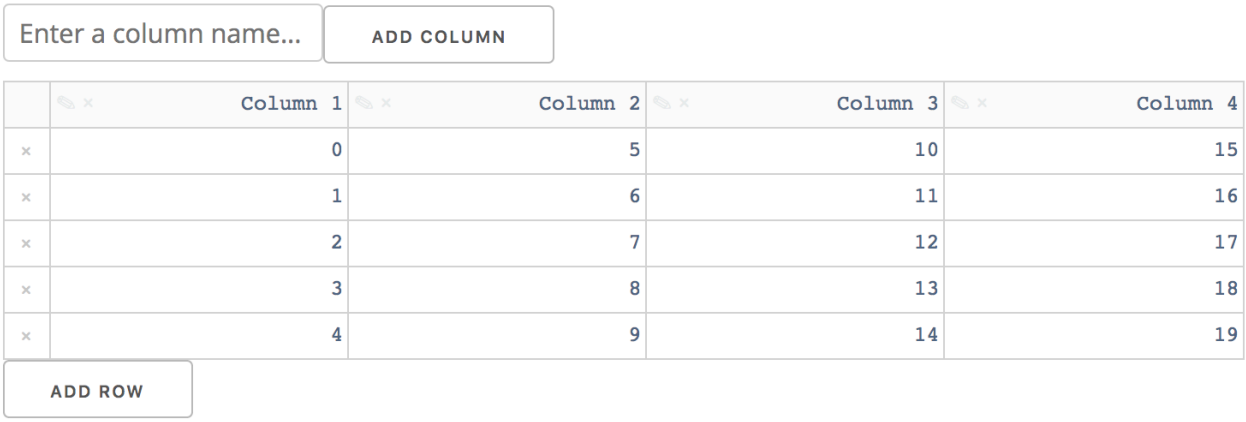


Figure 22 data table

In figure 23, we could see the code for the data table. The columns have two default names. Hence, the columns, Timestamp and Data, are defaulted. Users could add another column. Actually, they could even edit or delete data on the table.

```
dash_table.DataTable(
    id='adding-rows-table',
    columns=[{
        'name': 'Timestamp',
        'id': 'Timestamp',

    }] + [{'name': 'Data', 'id': 'Data'}],

    data=[{'input-data': i} for i in range(50)],

    editable=True,
    row_deletable=True
),
```

Figure 23 code for data table

Modelling and evaluation In modelling and evaluation, there are two core components: multi-value dropdown and bar chart. The multi-value has the parameter $'multi = True'$ (Figure 25)which is differs from former dropdown list(Figure 17). Therefore, The multi-value dropdown could have several values simultaneously.

Figure 24 multi-value dropdown

```
dcc.Dropdown(
    id='algorithm-dropdown',
    options=[
        {'label': 'KNN', 'value': 'KNN'},
        {'label': 'DT', 'value': 'DT'},
        {'label': 'NW', 'value': 'NW'},
        {'label': 'SVM', 'value': 'SVM'}

    ],
    value=['KNN', 'DT'],
    multi=True

),
```

Figure 25 code for multi-value dropdown

The bar chart shows that the measurement of the algorithms. We need several traces to show the metrics. The bar chart is considered to show the measurement value. This figure is beneficial to users to compare the values(Figure 26).

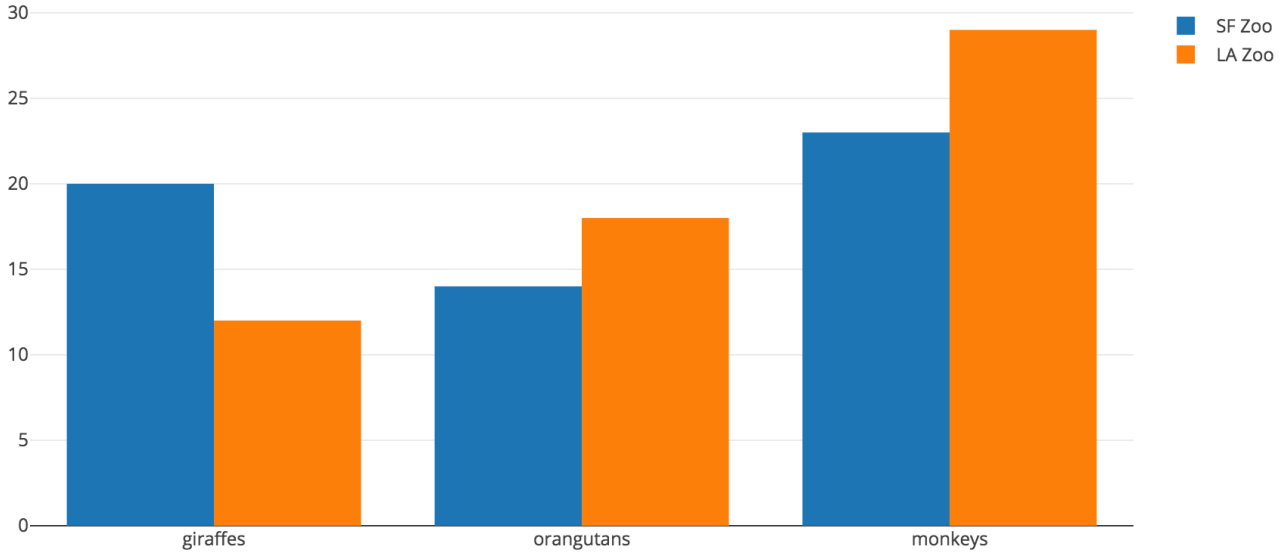


Figure 26 bar chart

Figure 27 shows the code how to implement the bar chart. We need change the *'x'* and *'y'* to show the x-axis and y-axis of the bar chart.

```
trace1 = go.Bar(
    x=Algorithm_name,
    y=algorithm_accuarcy,
    text=algorithm_accuarcy,
    name="accuarcy",
    marker=dict(

        line=dict(
            color='rgb(49,130,189)',
            width=1.5),
    )
)
```

Figure 27 code for bar chart

Automatic labelling In automatic labelling, the components are the same as the comments above. The purpose of the automatic labelling is to apply the algorithms selected by users. The output results would be shown on the table(Figure 28).

```
trace = go.Table(
    header=dict(values=list(['Timestamp','Data', 'label_results']),
                fill=dict(color='#C2D4FF'),
                align=['left'] * 5),
    cells=dict(values=[xaxis,inputData,prediciton_y],
               fill=dict(color='#F5F8FF'),
               align=['left'] * 5))

data = [trace]
```

Figure 28 code for table

2.4 Code for supervised learning algorithms

In data analysis part, we need combine the code with the web interface. Actually, the way to implement the supervised learning algorithms shown on Figure 29. We need use the functions in scikit-learn to train the data, e.g.

•$KNN = KNeighborsClassifier(n\_neighbors = 2)$

The function used for calling the KNN algorithm in the machine learning library.

•$KNN.fit(X, integer\_encoded\_label)$

'X' is training data and integer_encoded_label is labelled target data. The function is used for fit the training data and target data.

•$KNN.predict(prediction)$

Apply the algorithm KNN to never before seen test example 'prediction' and output the predicted value.

```
def applyKNNAlgorithm(prediction, X, integer_encoded_label):
    KNN = KNeighborsClassifier(n_neighbors=2)
    KNN.fit(X, integer_encoded_label)
    return KNN.predict(prediction)


def applyDTAlgorithm(prediction, X, integer_encoded_label):
    DT = tree.DecisionTreeClassifier()
    DT.fit(X, integer_encoded_label)
    return DT.predict(prediction)


def applySVMAlgorithm(prediction, X, integer_encoded_label):
    SVM = SVC()
    SVM.fit(X, integer_encoded_label)
    return SVM.predict(prediction)


def applyNWAlgorithm(prediction, X, integer_encoded_label):
    NW = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1)
    NW.fit(X, integer_encoded_label)
    return NW.predict(prediction)
```

Figure 29 the implement of algorithms

3. Results

The results of the project would be shown through web interface which has four tabs. In figure 30, the first tab called data visualisation, it would be used for users to see the signal data streams on scatter plot and the map.

The mechanism is that users could filter the data streams using some components like dropdown menu, slider and the calendar. In figure 30, we put the data like this way:e.g. use the calendar to put the data from *'09/24/2013'* to *'09/25/2013'*; Select dropdown menu to put the signal type *'Engine Speed'*; Use sliders to put the range of data. Finally, after users click the submit button, the scatter plot would show the data immediately.

The dropdown list offers users the ability that they could change signal data or time duration on the calendar. They could also use map to plot the vehicle track with timestamp, longitude and latitude.

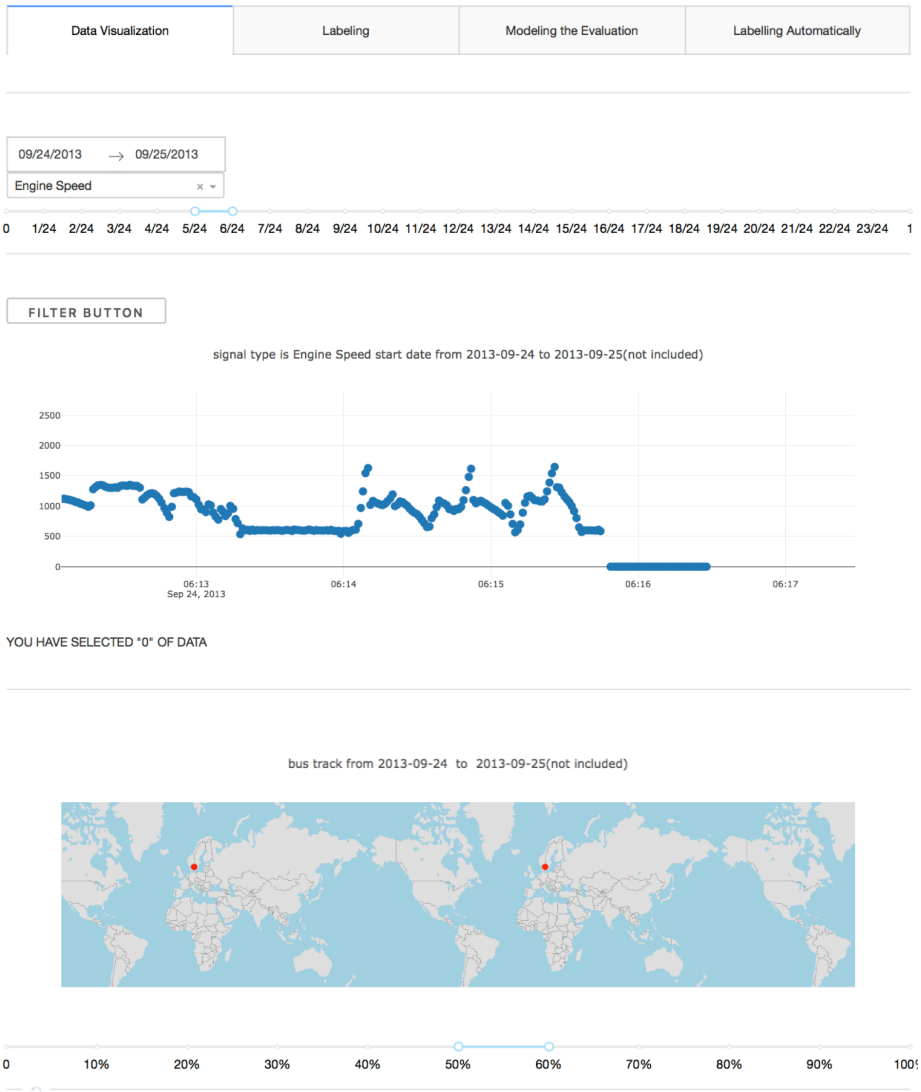


Figure 30 data visualisation

Example use-case 1

Find information through data visualisation.Now we zoom up the scatter plot(Figure 31), we could find there are several abnormal speed dots which surrounded by red circle. Then we zoom up again, we could find the vehicle is accelerating between 06:05:30 and 06:05:45. This means the serval abnormal speed dots caused by the vehicle's acceleration.

In the same time(time stamp), we could find there are also several intersections on the road(Figure 33). This means every the abnormal speed red circle is related to the intersection. Therefore, we could give an assumption: the number of the vehicle's acceleration is related with the number of intersections on the route.

If the assumption is right, we could count the number of intersections on the road and count the number of acceleration to compare them. If the number of acceleration is much larger than what we count from the map, the reason may be traffic congestion or road repair or the problem of the vehicle. Consequently, we could find more information through comparison.

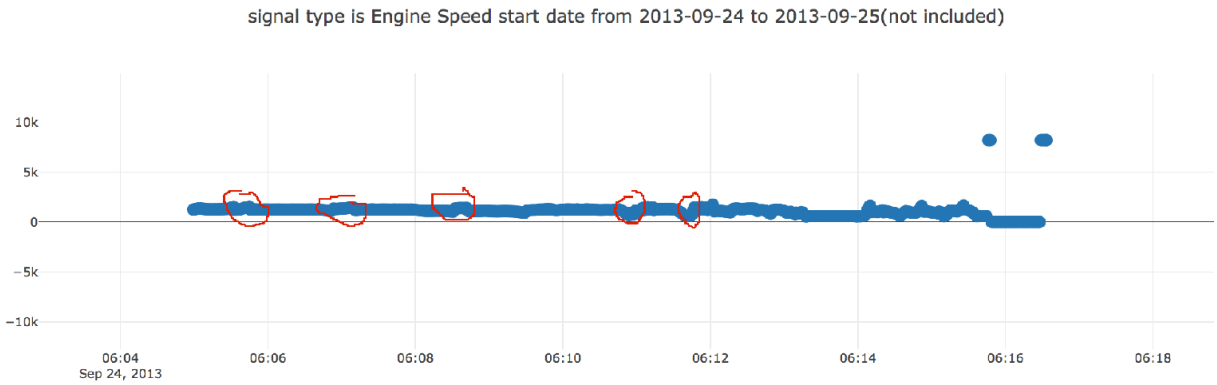


Figure 31 Engine Speed-1

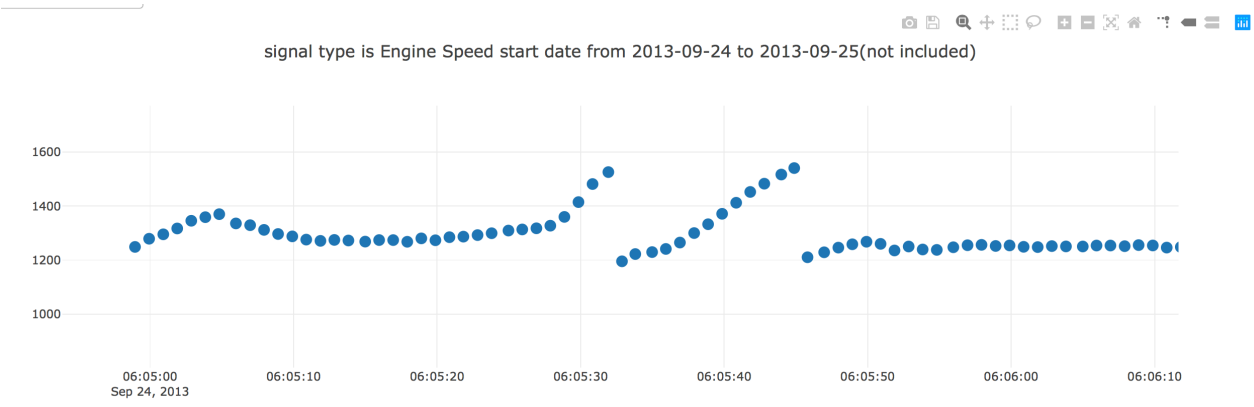


Figure 32 Engine Speed-2

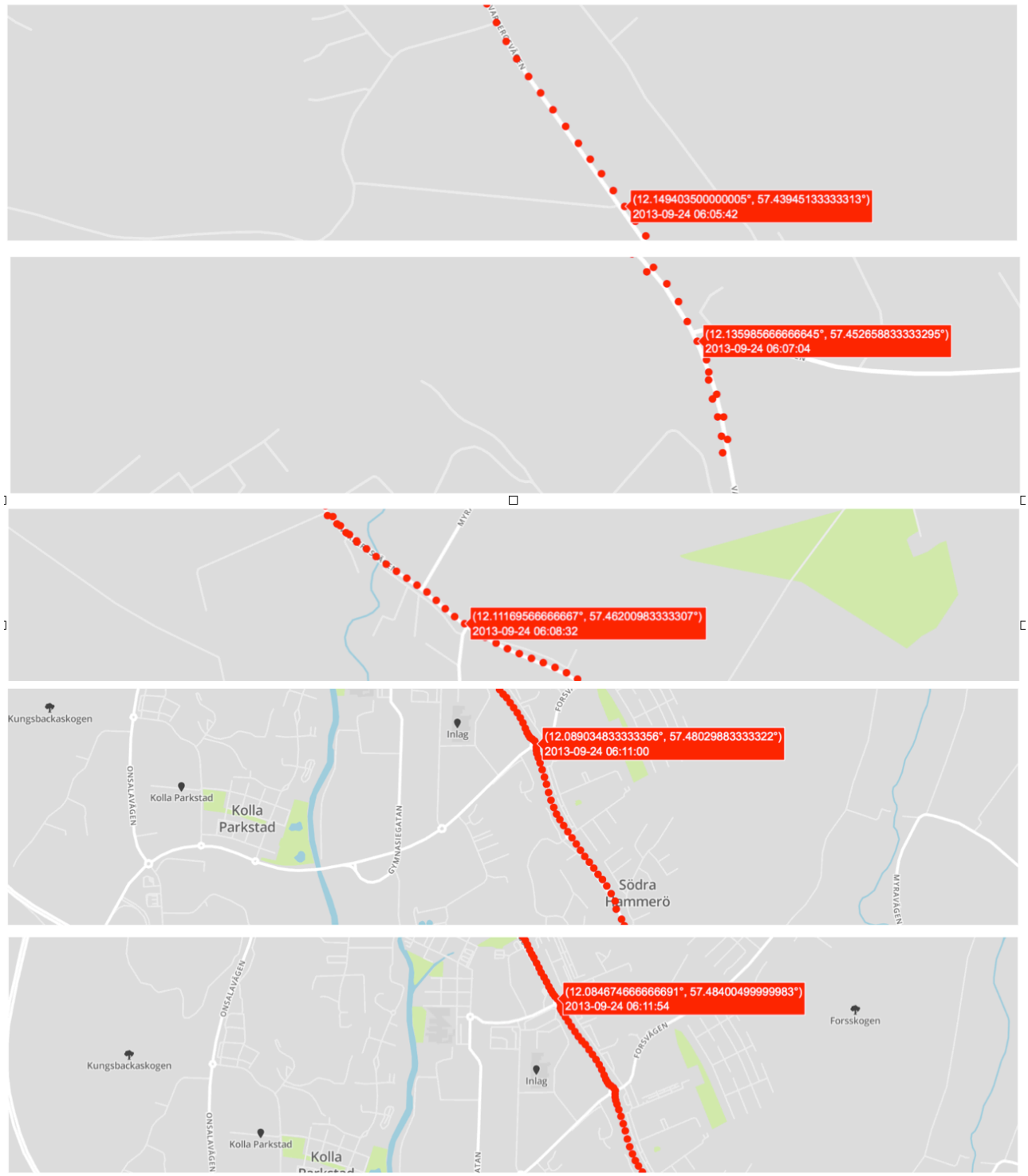


Figure 33 map

Example use-case 2

Explore abnormal data streams and predict to unused data. Another example is that we could detect the outliers from the data streams. For example, in figure 34, the scatter whose time is 06:15:48 and value is 215. It is different from others because other data value is much smaller that it. We could take it as unusual data. Now we could use the next part of the web system to do data analysis and anomaly detection.

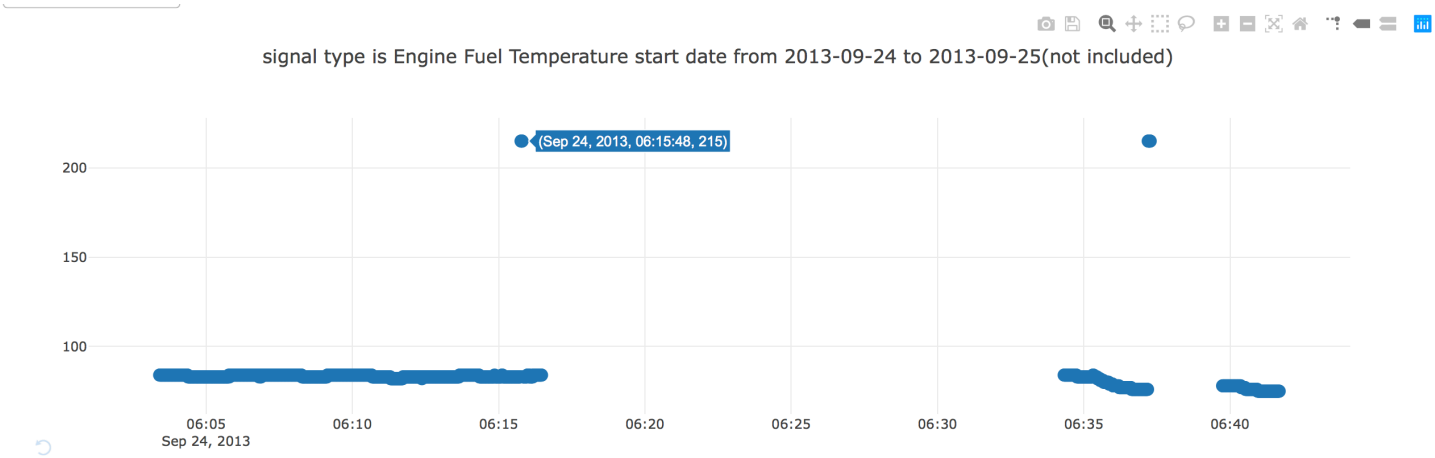


Figure 34 Fuel Temperature

In the beginning, we use the selection tool, box select or lasso select, to select the data on scatter plot. For example, in Figure 35, we use the box select to select a part of ‘Fuel Temperature' data. These data streams would be plotted on the manual labelling tab(Figure 36).

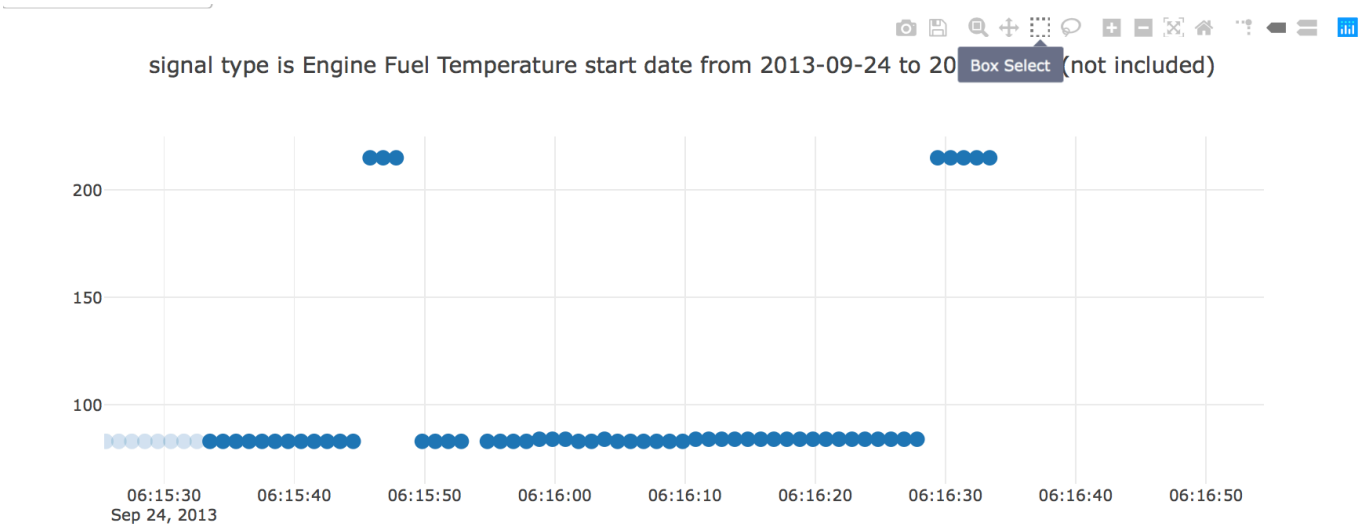


Figure 35 Fuel Temperature

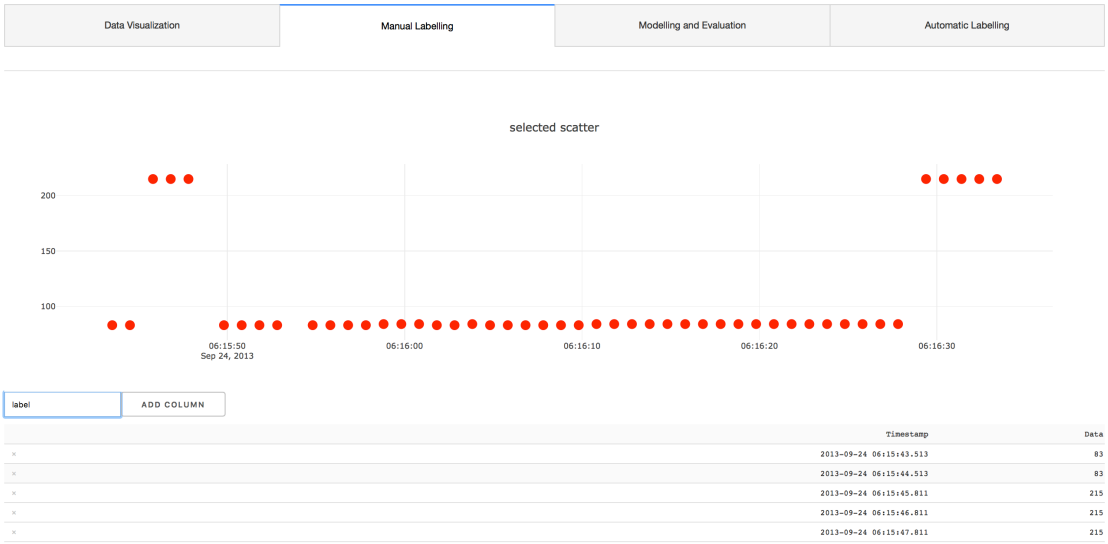


Figure 36 Manual Labelling

Users could classify the data and input the labelling value on the *'label'* columns. The label value is binary. We put *'0'* to represent normal data and the *'1'* to represent abnormal data on data table. We need manually label the selected data one by one until all data have been labelled(Figure 37).

label ADD COLUMN

	Timestamp	Data	label
×	2013-09-24 06:15:28.515	83	0
×	2013-09-24 06:15:29.515	83	0
×	2013-09-24 06:15:30.514	83	0
×	2013-09-24 06:15:31.514	83	0
×	2013-09-24 06:15:32.514	83	0
×	2013-09-24 06:15:33.514	83	0
×	2013-09-24 06:15:34.514	83	0
×	2013-09-24 06:15:35.514	83	0
×	2013-09-24 06:15:36.514	83	0
×	2013-09-24 06:15:37.514	83	0
×	2013-09-24 06:15:38.514	83	0
×	2013-09-24 06:15:39.514	83	0
×	2013-09-24 06:15:40.514	83	0
×	2013-09-24 06:15:41.514	83	0
×	2013-09-24 06:15:42.513	83	0
×	2013-09-24 06:15:43.513	83	0
×	2013-09-24 06:15:44.513	83	0
×	2013-09-24 06:15:45.811	215	1
×	2013-09-24 06:15:46.811	215	1
×	2013-09-24 06:15:47.811	215	1
×	2013-09-24 06:15:49.811	83	0
×	2013-09-24 06:15:50.811	83	0
×	2013-09-24 06:15:51.811	83	0

Figure 37 data table

The third part is modelling evaluation. Users should select the dropdown menu to put algorithms, e.g.We apply algorithms like kNN, Decision Tree on the selected data and submit the button.

The classification algorithms would classify them as normal or abnormal data. To evaluate the performance of the algorithms. Each algorithm in the bar chart(Figure 38) has three metrics.:Accuracy, F1 score and average-precision. The higher the score, the better it performs. We could easily find the kNN algorithm is best one for this demo.

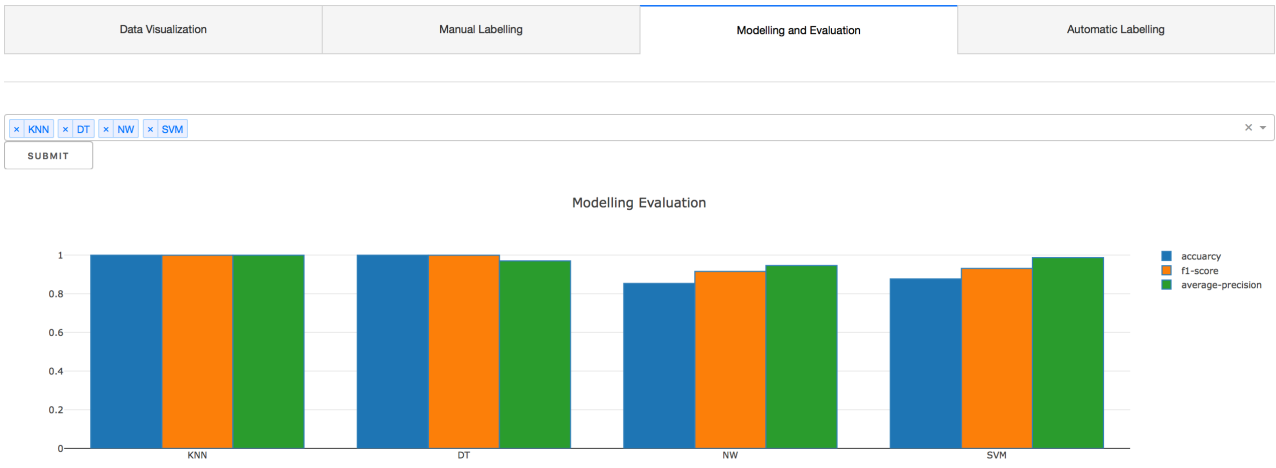


Figure 38 modelling and evaluation

The last step(Figure 39) is about automatic labelling. Users could use the filter button to select one part of the data to do prediction. After that, we select the dropdown menu to put the best performing algorithms and then submit the button. The system would apply the same pattern to unused data to predict it.

The results would be shown on the prediction table(Figure 40). As the label results are numerical, we could plot the results on another scatter plot. The results also show the percentage of the normal or abnormal data occupied in the total number. This is helpful for users to do data analysis.

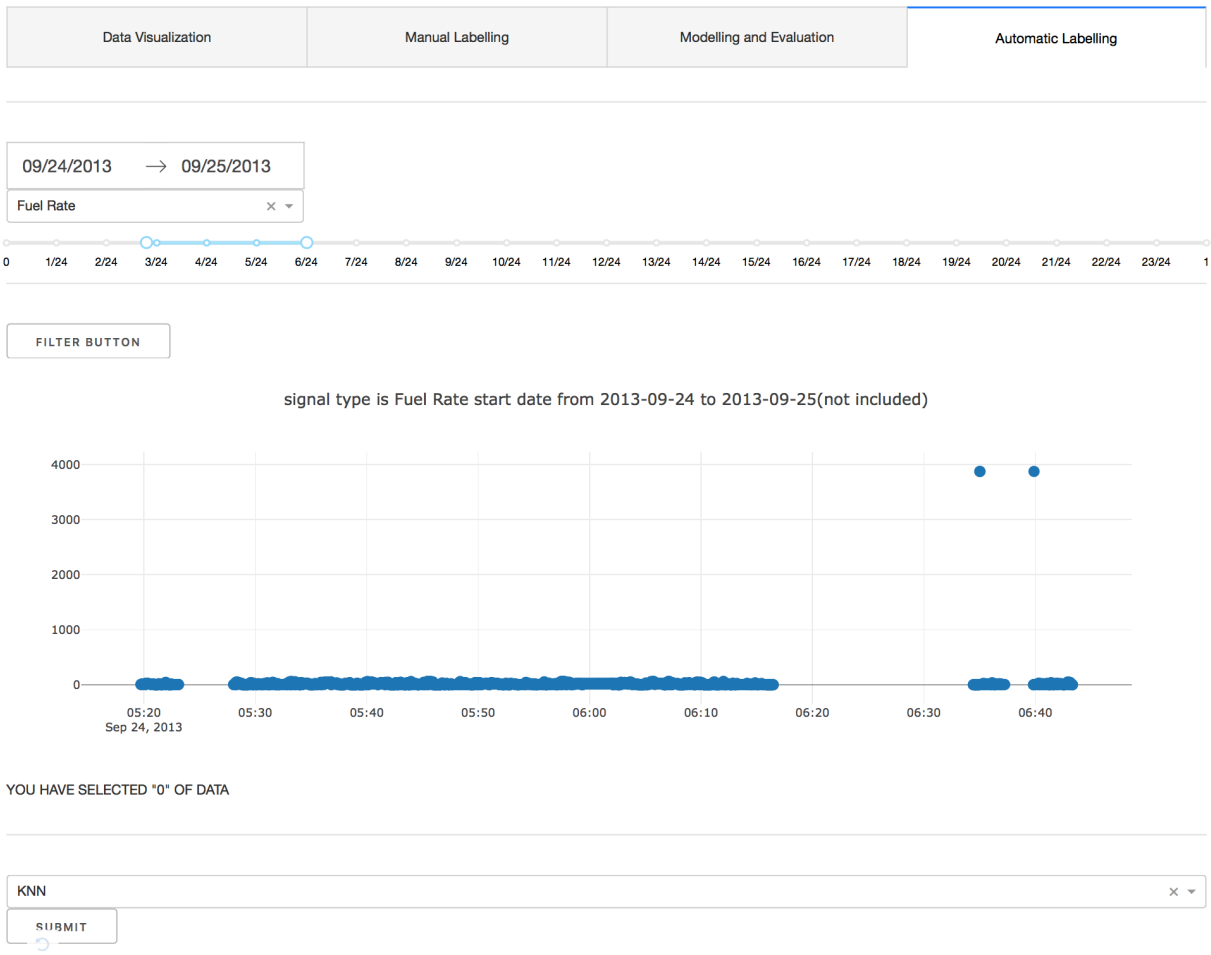


Figure 39 automatical labelling-1

prediction

Timestamp	Data	label_results
2013-09-24 06:03:24.58	84	0
2013-09-24 06:03:25.58	84	0
2013-09-24 06:03:26.58	84	0
2013-09-24 06:03:27.58	84	0
2013-09-24 06:03:28.58	84	0
2013-09-24 06:03:29.58	84	0
2013-09-24 06:03:30.58	84	0
2013-09-24 06:03:31.58	84	0
2013-09-24 06:03:32.579	84	0
2013-09-24 06:03:33.579	84	0
2013-09-24 06:03:34.579	84	0
2013-09-24 06:03:35.579	84	0

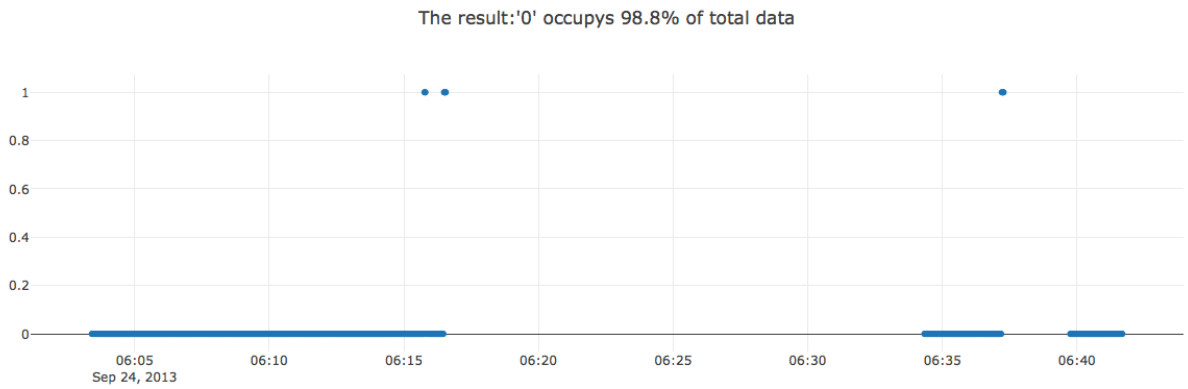


Figure 40 automatical labelling-2

4.Evaluation

4.1 Evaluation on the web interface.

In this project, we successfully achieve the objective to do a visualisation tool for data scientists or fleet operators. We compare our engineering work with older version. We evaluate the system to optimise its performance. The system's performance was measured through these two measurements: response time, functionality..

Response time:
During the process of making a website, I meet the problem of how to improve the speed of website delay, as the number of data is massive. In the original version of the web interface, there are two tabs: bus information and map. We select calendar and dropdown menu to put data, but the data is also massive. So this way makes the system poor response time. We need less time to respond to users. To accelerate the speed, we use the strategy
that using hardware memory to save response time. We sorted the data into CSV file format instead of database[10](Figure 41).

```
1  sqlite3 test.db
2  .output test.csv
3  select * from Data
4  .output stdout
```

Figure 41 sqlite3 code for file format

speeding time

In addition, we also use the interactive slider to control the number of the input data. This method would also save the plotting time. The two ways help the system reduce the response time.

Functionality

In the original version of the web tool, there are only two tabs. The first tab is *'bus information'* and the second tab named *'map'*(Figure 42). The functionality is limited. Because it could only offer users the analytic information through the comparison. However, in the later version, we improve it with supervised learning algorithm to achieve our goal. In this aspect, it would be more helpful for users. Because they could apply the the pattern to whole data. So the system has more functions:labelling, modelling and application. These three functionalities would be helpful for their analysis work.

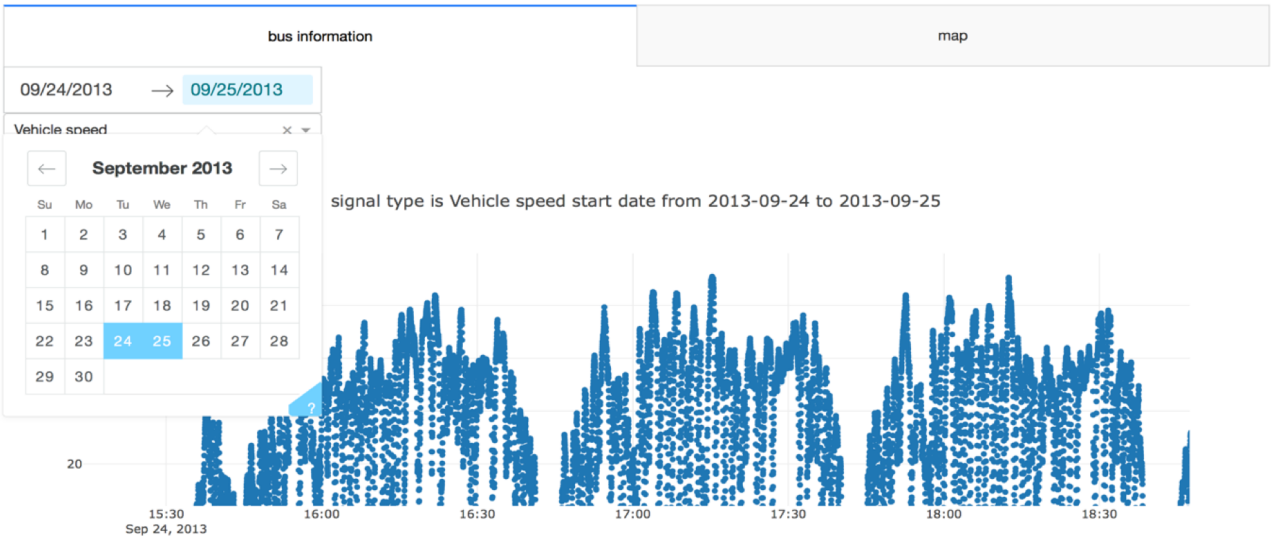


Figure 42 the original version

4.2 Evaluation on algorithms

To evaluate the performance of the algorithms, we use different metrics to compare the performance. In the older version, we just use one measurement: accuracy. However, accuracy is not the best measure for assessing classification classes. Because it may be a poor measurement for imbalance data. Then we add another two metrics: F1-score and average-precision. The precision helps when the costs of false positives are high. The F1-score combines precision and recall.

We should also notice that supervised learning algorithms has different sets of parameters. This different sets of parameters according to different mathematical formulations. To evaluate the performance of the algorithms, we need also find the best sets of parameters.

5. Conclusion

5.1 Summary

In this project. We finished the web interface and combine it with supervised learning algorithms. The visualisation tool is beneficial to data scientists or fleet operators. This tool may help them in anomaly detection techniques. It could be used for detecting abnormal behaviour of the vehicle using sensor data streams like fuel temperature or engine speed. In data visualisation tab, users could compare different signal types of data including acceleration or deceleration. They could also label these data to classify them or apply the pattern to other data to do prediction.
In the supervised learning part, I combined machine learning knowledge with an interactive website to analyse the data and understand the behaviour of the vehicle. The combination of different tools would assist users more easily to understand the vehicle's operation and predict the demand for maintenance.

5.2 Further work

We need also further work to improve the performance of the project:

•**Survey user interface** To survey the web interface, we could send questionnaires to users. Firstly we need let users understand the web interface tool and then we collect opinions from them. Next, we analyse the opinions or preferences to evaluate it is suitable for the project or not. Finally, we could change the user interface to make it more easy to use.

For example. we could change the name of the tab. So it would make users more easily to understand the behaviour of the vehicle; We could also adjust the way the interface organised so that users could compare the data with the track in the map without switch the tab again.

•**Add classification algorithms** In future work, we could improve the system on the algorithms. Supervised learning is easy to understand for freshmen. However, it could only solve part of the outliers or anomalies. We could improve the system using different classification algorithms including supervised learning or unsupervised learning or semi-supervised learning and so on. After evaluating the performance of the algorithms, we could choose the best algorithm according to the type of behaviour. Then, the project would be more helpful for users to understand the behaviour of the vehicle and address the problems.

6. Time Plan

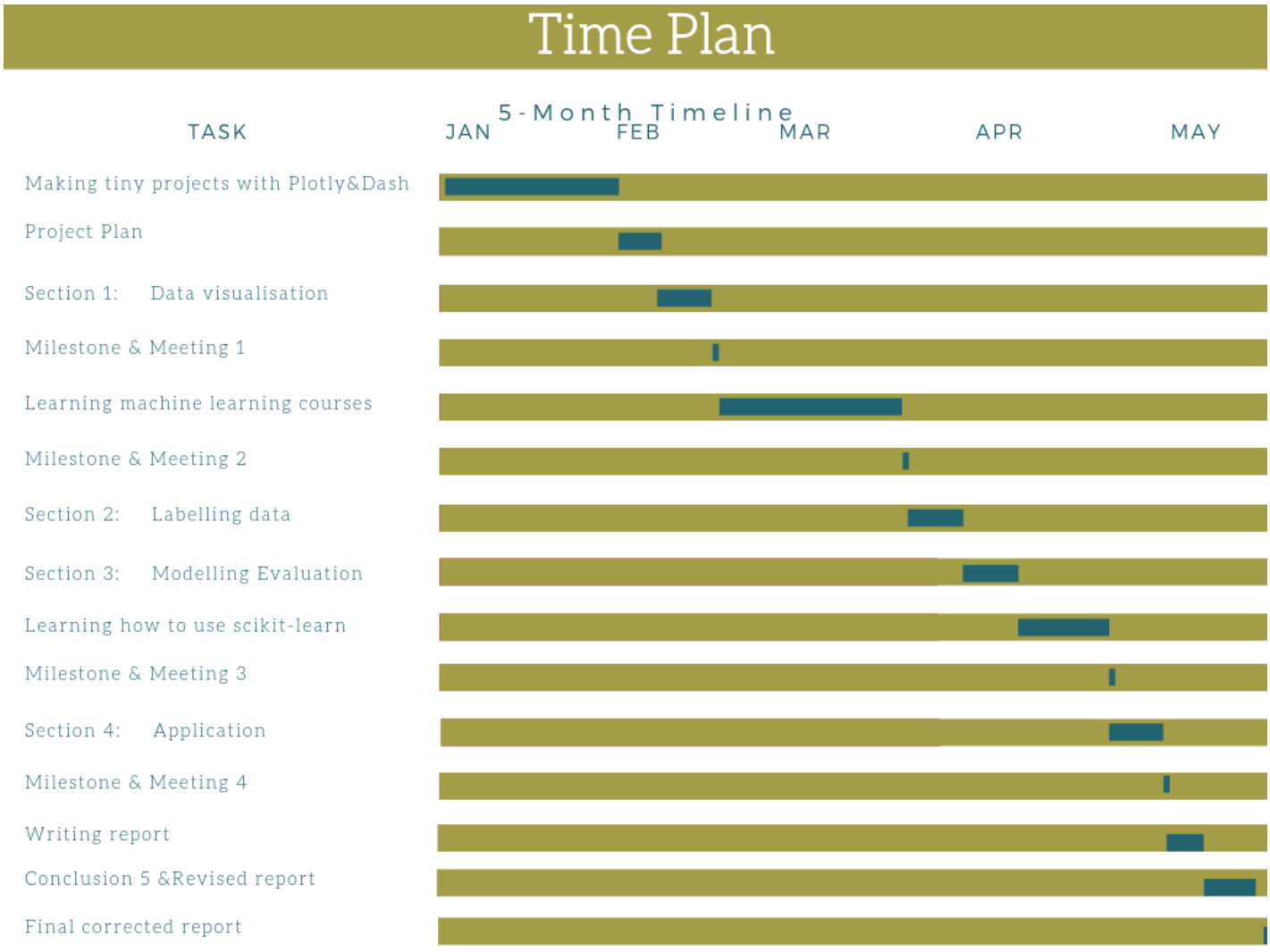


Figure 43 Time plan

7. References

[1] Jeff Lai (2017)Visualizing *Deep Learning with Plotly.* Available at: https://medium.com/@jefflai108/visualizing-deep-learning-with-plotly-8f2500e24b27 (Accessed at:15 May 2019)

[2] Stacey Ronaghan. (2018) *Machine Learning Trying to detect outliers or unusual behaviour.* Available at : https://medium.com/@srnghn/machine-learning-trying-to-detect-outliers-or-unusual-behavior-2d9f364334f9 (Accessed:15 May 2019)

[3] Scikit-learn 0.21.2. (2019) *Choosing the right estimator* Available at: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html (Accessed:15 May 2019)

[4] Scikit-learn 0.21.2. (2019) *Nearest Neighbors* Available at: https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html (Accessed:15 May 2019)

[5] Wikipedia (2019) *k-nearest neighbors algorithm.* Available at: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm (Accessed at:25 May 2019)

[6] Galathiya, A. S., A. P. Ganatra, and C. K. Bhensdadia. "Improved decision tree induction algorithm with feature selection, cross validation, model complexity and reduced error pruning." *International Journal of Computer Science and Information Technologies* 3.2 (2012): 3427-3431.

[7] Savan Patel. (2017) *SVM (Support Vector Machine)—Theory.* Available at: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72 (Accessed:25 May 2019)

[8] Wiki. *A.I. Wiki.* Available at: https://skymind.ai/wiki/accuracy-precision-recall-f1 (Accessed at:25 May 2019)

[9] Dash.(2019). *Dash User Guide* Available at: https://dash.plot.ly (Accessed:15 April 2019)

[10] pylduck(2017). *SQLite*表导出*csv*文件 Available at:
https://blog.csdn.net/pylduck/article/details/78250732
(Accessed:15 March 2019)