Data-driven framework to make prediction using error based clustering technique: Manual

Pragneshkumar R Rana, Krithika Narayanaswamy,
Sivaram Ambikasaran

June 2, 2021

Contents

List of Figures

Note: The proposed framework works appropriately with any data set having continuous dependent variable. As this framework came out while studying ignition delay, certain explanation is made by keeping ignition delay data as a central part (reference). For data-set other than fuel, please look at the '-o ′ flag.

1 Key-idea behind framework:

The goal of a framework is to make predictions. For that, machine learning concepts, multiple linear regression is used along with the concept of the recursive tree. To divide the data based on error, the tertiary tree is utilized.

The whole process is summarized below:

1. Fit regression plane on all the data using multiple regression.
2. Calculate the relative error of the actual and predicted value for all data.
3. Data points that have a relative error less than specified criteria will be combined in one bin called **center cluster**
4. Other data, which has a relative error more than the specified criterion, will be further bifurcated based on the sign of error difference of actual and predicted value.
5. Data points which have positive absolute error has lain on one side of the fitted plane, and other data points which have negative sign has lain on another side of the fitted plane.
6. Thus, three clusters will be obtained in which,

   Left cluster: Data points with the relative error of more than specified error and prediction error have a positive sign.

   Center Cluster: Data points that have a relative error less than specified criteria.

   Right Cluster: Data points with relative error more than specified error and prediction error have a negative sign.

The Center cluster gives a correlation for the data and will not be divided further. Whereas, Left and Right cluster may also give correlation if they satisfy the specified relative error criteria; otherwise, it will be divided recursively by following step:1-6 until specified relative error criteria are not fulfilled.

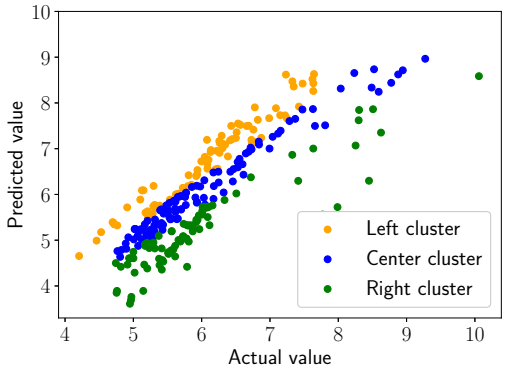
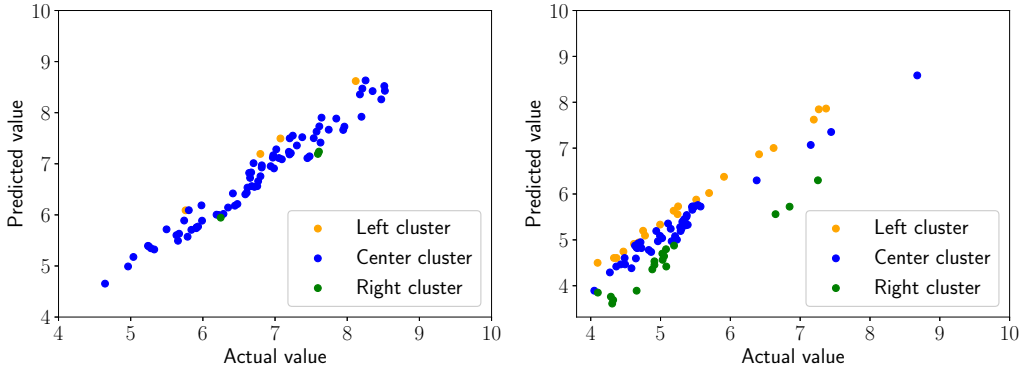


Figure 1: Division of cluster based on relative error in prediction and sign of prediction error (1^{st} level regression). Blue Data points satisfy 5% criterion of relative error. Orange and Green data points have higher error than the specified threshold value.



(a) Left Cluster Division (b) Right Cluster Division

Figure 2: Division of left and right cluster into sub-clusters (2^{nd} level regression). Blue data points satisfy 5% criterion of relative error. Orange and green data points have higher error than the specified threshold value.

For example, after fitting the first regression plane, the actual and predicted value is shown in Fig. 1. The blue colour shows the data points which has a relative error of less than 5%. Green and orange coloured data points have a relative error of more than 5%. Apart from relative error, the sign of error difference also plays an essential role in clustering. Orange data point

has a positive error difference and green data points have negative error difference, which indicates that they lie in opposite side of fitted regression plane. In this way, three clusters are obtained. Further, orange and green data points will be divided into three clusters individually, following the same procedure explained above. The obtained result is shown in Fig. 2a and 2b.

Each center clusters gives one correlation for the prediction of ignition delay. Left and right clusters can also give correlation (generate cluster) if it satisfies the specified relative error criteria else, it will further divide into more parts/clusters until relative error criteria are not satisfied. For example distribution of data points in the different cluster is shown in Fig. 3. From figure, it is clear that the nodes highlighted with green borders generate final clusters.

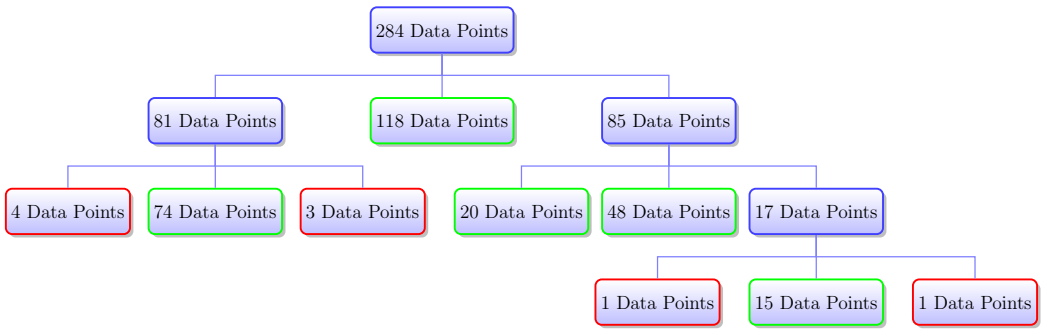


Figure 3: Division of 284 ignition delay data points into different clusters after applying recursive regression based clustering tree algorithm. Terminal nodes with the green border generate resultant prediction models, whereas terminal nodes with the red border have fewer data points than required to proceed further so, they were discarded.

Once all clusters are generated, identification of cluster for new/unknown data point is major issue to get suitable model. Generally, centroid of the cluster is utilized for identification of cluster. But, in this framework apart from centroid, different extreme point is utilized to make correct identification of cluster and avoid wrong assignment of cluster. The idea is illustrated in Fig. 4

To understand full algorithmic procedure, please go through the article.

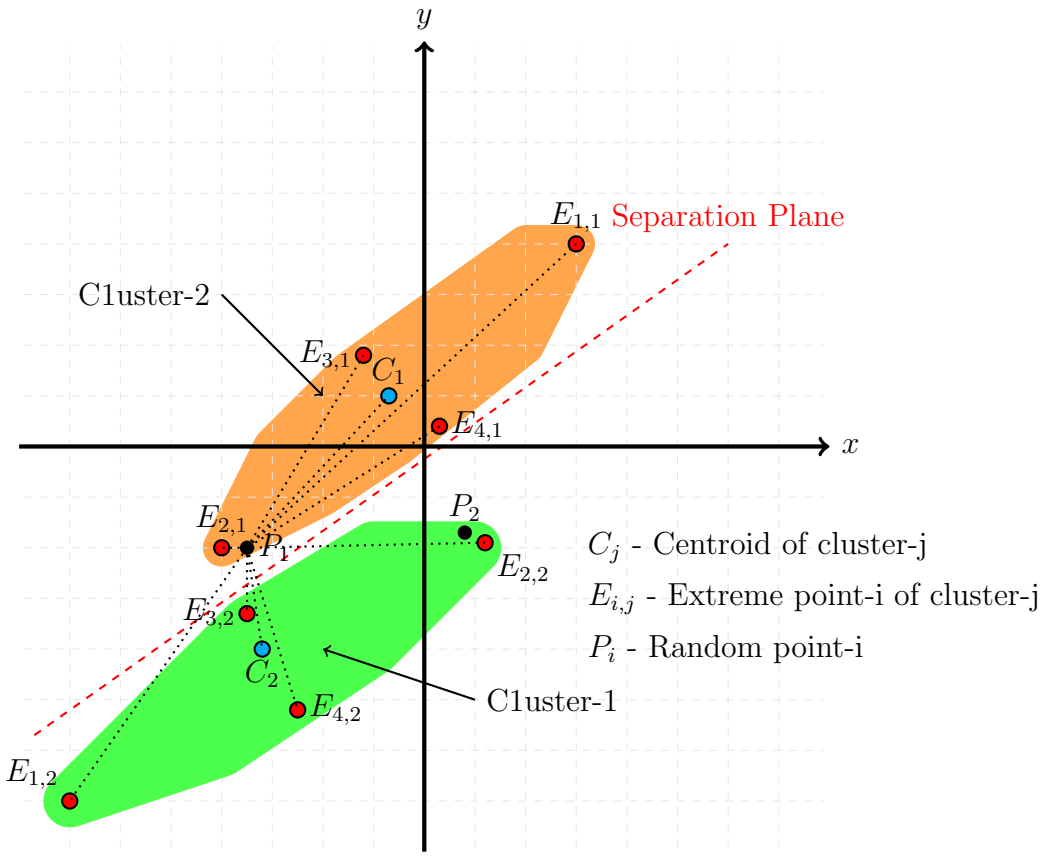
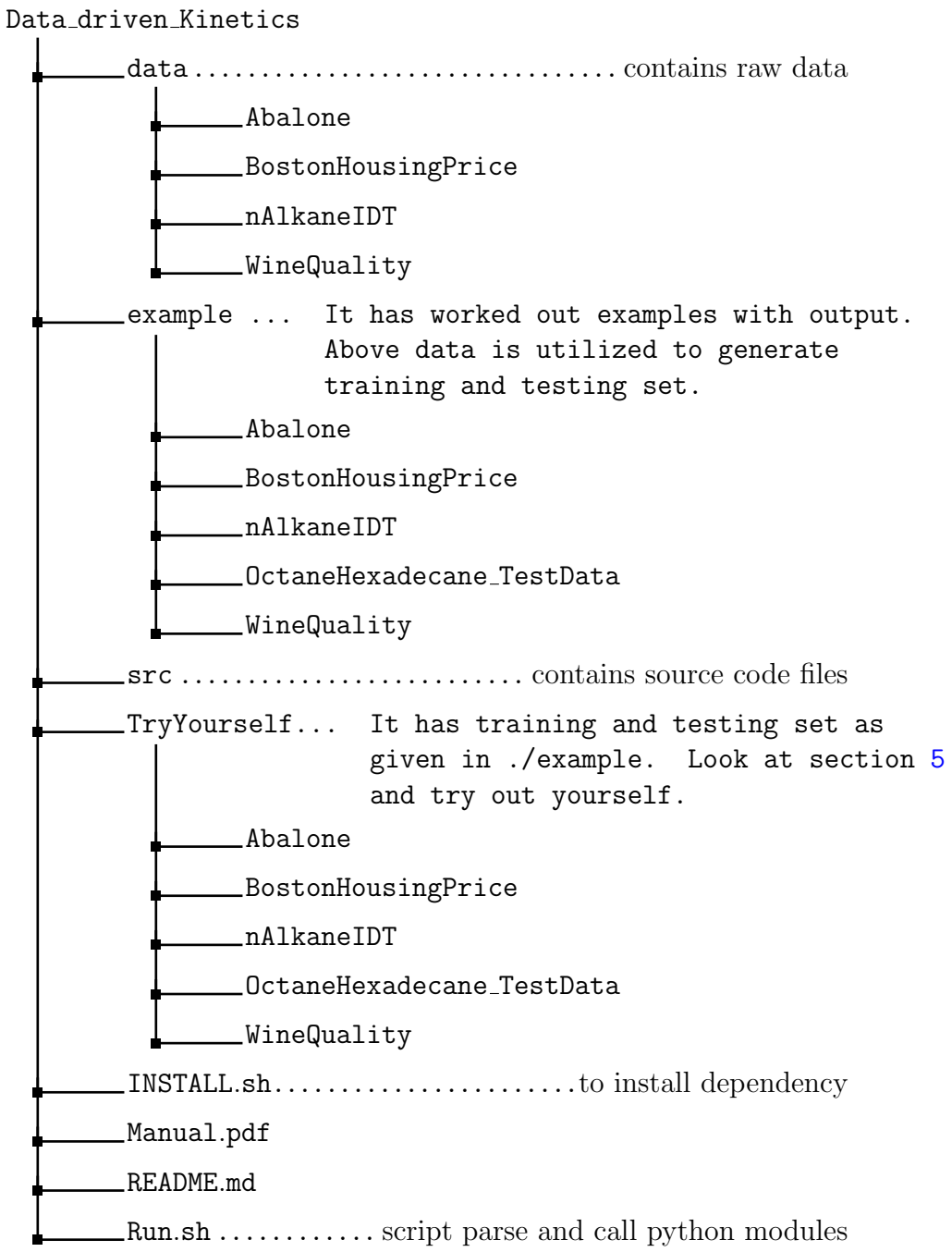


Figure 4: Use of extreme points along with centroids for correct assignment of cluster or model to the data point. Assignment of cluster only through the centroid is erratic which is illustrated by point P_1 and P_2. Point P_1 is part of cluster-1. But, if cluster is assignment done only using centroid then point P_1 will be assigned to cluster-2 as $||P_1 - C_2||_2 < ||P_1 - C_1||_2$. To rectify it, Euclidean distances should be calculated from all the points $E_{i,j}$ and C_j. Out of which, the point with the least distance is prominent for assignment of a correct cluster. Here, the point $E_{2,1}$ is nearest to the point P_1 so, cluster-1 should be assigned to P_1 as $E_{2,1}$ is part of it.

2 Directory structure:



3 Setting up for the first time [Linux OS]:

- After downloading the application, put it in the *./home* directory.
- Install all the dependency. Dependency can be directly installed using INSTALL.sh file. They are mentioned below:
  - numpy
  - matplotlib
  - seaborn
  - scikit
  - statmodel
  - rdkit
  - latexpdf - Install manually by following instruction

  To install all dependency run the following command:
  - chmod +x INSTALL.sh
  - ./INSTALL.sh
- After installing all the dependency, open **.bashrc** file and copy-paste following command at end of the file.

```
##For DATA DRIVEN FRAMEWORK
export IDCODE="~/Data_driven_Kinetic/"
export PATH=$PATH:$IDCODE
alias IDprediction="pwd ~/Data_driven_Kinetics/filelocation.txt && Run.sh"
```

- After saving the code, source the changes by typing following command.

```
source .bashrc
```

- All set! - Run the program following instruction given in section-4

4 Commands to run the program:

After following the procedure given in section-3, it is convenient to use the program from any directory. let's consider filename- **alkane.csv** for explanation of all commands.

- Command to run the code:

  IDprediction -flag File-Name

4.1 Available options and flags:

4.1.1 -a : *'Analyze'* the data-set by selecting range of parameters

- command : **IDprediction -a alkane.csv**

- To analyze and find out data-points having properties in certain range to analyze ignition delay for specific fuel.
  - select fuel by giving corresponding index as input
  - Input equivalence-ratio from available options
  - Input pressure value from available options
  - By default it will considers all the data points having same fuel structure and same equivalence ration within ± 0.5 atm. If you want to change the range pressure input:'y' and input the desired range else input:'n'

- **Output :**
  - ./result/data_analysis/

    It will generate the plot of τ vs 1000/T, which includes data points having selected ϕ, specified range of pressure for selected fuel.

4.1.2 -b : *'Bond'* types in given fuel

– command : **IDprediction -b CCCC**

To find out type of chemical bonds available in different type of carbon and other atoms.
Note: Works for n-alkanes and branched alkanes.

– **Output :**

* ./result/Bond_details/

It will generate **SMILE_result.csv** file which contains available bond details for given fuel. Detail in the file includes,

· **Primary_C** : Total number of primary carbons
· **Secondary_C** : Total number of secondary carbons
· **Tertiary_C** : Total number of tertiary carbons
· **Quaternary_C** : Total number of Quaternary carbons
· **Other_Atom** : Total number of atoms other than carbon
· **P_P** : Total number Primary-Primary carbon bonds
· **P_S** : Total number Primary-Secondary carbon bonds
· **P_T** : Total number Primary-Tertiary carbon bonds
· **P_Q** : Total number Primary-Quaternary carbon bonds
· **S_S** : Total number Secondary-Secondary carbon bonds
· **S_T** : Total number Secondary-Tertiary carbon bonds
· **S_Q** : Total number Secondary-Quaternary carbon bonds
· **T_T** : Total number Tertiary-Tertiary carbon bonds
· **T_Q** : Total number Tertiary-Quaternary carbon bonds
· **Q_Q** : Total number Quaternary-Quaternary carbon bonds
· **P_H** : Total number Primary carbon - Hydrogen bonds
· **S_H** : Total number Secondary carbon - Hydrogen bonds
· **T_H** : Total number Tertiary carbon - Hydrogen bonds
· **Fuel** : Fuel SMILE

Note : If file already exits then result will be appended to old output file.

4.1.3 -h : *'Histogram'* plots of parameters for each fuel individually

- command : **IDprediction -h alkane.csv**

  It will generate the histogram plots of parameters associated with ignition delay, separately for all the fuels.

- **Output :**
  - ./result/Fuel_Parameter_Histogram/

    Directory contains, folders named by fuel smiles. Each folder contains histogram plot of parameters associated with ignition delay. These plots are useful for visualization and analysis of the parameters.

4.1.4 -c : *'Criteria'* for separation

- command : **IDprediction -c 0.05 -m alkane.csv**

  Flag has to be passed with multiple linear regression or error based clustering regression. Criteria value 0.05 indicates relative error of 5%. Criteria plays key role in creation of clusters. Use of criteria is to filter out the data points which has relative error less than specified value to create a cluster.

- **Default : 0.05**

4.1.5 -l : *'Limiting'* the reference point

- command : **IDprediction -c 0.05 -l (10/True/False) -m alkane.csv**

  Flag has to be passed with multiple linear regression. If you pass 10 as value then number of reference point will be limited by 10 times the feature columns

  Number of reference points for each cluster will be,

  # (any number) : # times total number of features

  True : All the points will be used as reference points.

  False : It will use centroid and one farthest point

- **Default : False**

4.1.6 -r : *'Remove'* feature by back-elimination

– command : **IDprediction -c 0.05 -r True -s 0.05 -m alkane.csv**

This Flag **has to be passed with True or False** .
True : Backward elimination will be activated
Flase : Backward elimination will be deactivated

– **Default : False**

4.1.7 -s : *'Significance Level'* for p-value or Confidence zone criteria

– command : **IDprediction -c 0.05 -r True -s 0.05 -m alkane.csv**

Value passed with this flag is useful for backward elimination. Significance level is used for rejection/acceptance of null hypothesis. Low significance value is generally passed as it indicates higher evidence is needed for rejection of null hypothesis.

– **Default : 0.05**

Null hypothesis :

For multiple linear regression, null hypothesis defined as, there is no statistical relationship between independent and dependent variables or coefficient associated with dependent variable is zero.

Backward elimination :

In this procedure, if any feature has p-value more than specified or default value 0.05 (if -r True and -s not passed) then that feature will be eliminated. Result will be obtained running regression again. Same procedure will be repeated till p-value associated with all the regressor are less then specified value.

4.1.8 -m : *'Multiple'* linear regression of data

– command : **IDprediction -c 0.05 -r False -m alkane.csv**
or **IDprediction -m alkane.csv**

It will do multiple regression on whole data-set and will **generate output as root of the tree.** Details of other flag is mentioned above. If others flags are not passed then it will take default value as mentioned.

– **Output :**

* *./object_file/* - useful for prediction
  This directory has three more sub-directory inside it; **./regressor -** object file of trained model. **./x_names** - has feature names of trained model. **./scalar**- object file which has function to scale data (NOT UTILIZED).

* *./result/coefficients/Node_type/*
  stores coefficient obtained after multiple regression

* *./result/console_output/Node_type/*
  Output printed on console screen also get stored in the **output_result.txt** file

* *./result/Node_type/ID_comparison_i/*
  Directory contains **./ID_comparison_i** folder, which contains file named, **ID_comparison_test_i.csv** which has Ignition delay comparison detail for testing and likewise similar detail of training data is in **ID_comparison_train_i.csv** file. Ignition Delay comparison file contains y_act(Actual Ignition Delay value), y_predicted(Predicted Ignition Delay value), and relative error between those two values.

* *./result/vif/Node_type/*

  VIF(Variation Inflation Factor) - it is useful to check the multi-collinearity of the features
  VIF = 1 means feature has no multi-collinearity
  VIF = 1-5 means feature has moderate correlation
  VIF > 5 means poorly identify the coefficient
  ./vif_i.csv contains features and associated VIF values.

  source:
  Multicollinearity
  VIF

  Note: i here indicated index
  Node Type includes : {root,left_node, center_node, right_node}

4.1.9 -t : *'Tree'* type regression based clustering algorithm (error based clustering)

– command : **IDprediction -c 0.05 -r False -t alkane.csv**
or **IDprediction -c 0.05 -t alkane.csv**

The whole idea behind error besed clustering is explained in section-1.

– **Output :**

* *./object_file/* -useful for prediction

  This directory has Four more sub-directory inside it;

  · **./regressor -** object file of trained model.
  · **./x_names** - feature names of trained model.
  · **./scalar** - object file which is used to scale the data (NOT UTILIZED)
  · **./centroid**- it contains files for final clusters (highlighted nodes as in fig-**??**). Each file contains average calculated value of features associated with cluster-data.

* *./plots/*

  Directory contains several tree based plots which helpful for visualization. Along with that it has sub directory named

./cluster_plot_y which includes comparative plots of y_{actual} vs $y_{predicted}$ for every clusters as shown in fig-1;
./plots/ directory has six plots. All plots are generated with the help of latex.

- **ChildLabel** plot is shows index associated with clusters.
- **Datasize** plot shows number of data points in each cluster.
- **Training** plot shows R2 value of training set in each cluster.
- **Testing** plot shows R2 value of testing set in each cluster.
- **MaxRelError** plot shows maximum relative error of training set for each cluster.
- **coefficient** plot shows coefficient obtained after regression for each cluster node.

./plots/cluster_plot_y/- it contains plots as in fig-1. Plots are useful to understand role of relative error and absolute error in clustering. Data points with same colour are in one cluster.

* *./result/coefficients/Node_type/*
  stores coefficient obtained after multiple regression

* *./result/cluster_data_before_regression/Node_type*
  Directory contains cluster specific data-file of **useful clusters -** means data which is utilized for generation of cluster.

* *./result/Tree_coefficient_result/Node_type/*
  stores coefficient obtained after multiple regression which is same as copy of ./result/coeffcients/. Stored seperatly as it will not mix its result with Multiple regression result. **It has data of only final clusters.**

* *./result/final_cluster/Node_type*
  contains data file of seperated clusters for all the nodes

* *./result/console_output/Node_type/*
  Output printed on console screen also get stored on the in the output_result.txt file

* *./result/Node_type/ID_comparison/*
  Directory contains ./ID_comparison_i folder, which has Ignition delay comparision files training and testing. Ignition delay comparison file contains y_act(Actual Ignition delay value), y_predicted(Predicted Ignition Delay value), and relative error .

* *./result/vif/Node_type/*
  VIF(Variation Inflation Factor) - To check the multi-collinearity of the features
  VIF = 1 means feature has no multi-collinearity
  VIF = 1-5 means feature has moderate correlation
  VIF > 5 means poorly identify the coefficient
  ./vif_i.csv contains features and associated VIF values.
  source: Multicollinearity, VIF

  Note: i here indicated index
  Node Type includes : {root,left_node, center_node, right_node}

4.1.10 -e : *'External'* Dataset used for prediction of ignition delay

– command : **IDprediction -e testset.csv**

testset.csv file may contain all the features except ignition delay values. The goal of the process is to predict ignition delay. Procedure behind code is briefly mentioned below:,

1. Load all the centroid files
2. Calculate the distance of all the data points from all the centroids and assign the centroid to each data point by least distance.
3. By assigned cluster, load the respective regressor object of that cluster and predict the ignition delay and process the result.

– **Output :**

* *./external_test_result/console_output/*

Directory contains **output_result.txt** which contains output printed on console screen.

* *./external_test_result/Ignition_delay_comparison/*
  ID_comparison_external_cluster_{Node_index}_{Node_type}.csv file which includes y_predicted - Predicted Ignition delay value, y_actual - actual ignition delay vales and relative error between them. All files are cluster specific means data point in the assigned to one specific cluster.

* *./external_test_result/classified_data*
  Directory contains several data files. Each file is associated with one cluster and it contains independent variable values,class of data points, predicted and actual ignition delay values.

* *./external_test_result/prediction_comparison_plots/*
  It contains Predicted Ignition Delay vs Actual Ignition Delay plot. Plots are generated using on data points assigned to specific cluster. In short, each plot is related to each cluster.

4.1.11 -k : To predict external dataset result of ignition delay for 'k' training modules (Modules are generated by different training set) and store all test prediction result in different directory [effect of sampling])

– command : **IDprediction -k testset.csv**

– ALL PROCEDURE AND OUTPUT WILL BE SAME AS OBTAINED IN '-e' flag +

– **Output :**

* *./all_test_result_i*
  Result of all the test dataset will be store here.

4.1.12 -f : Probability density *'function'* plot of testing result (when we want to plot pdf of test-result obtain by running code on same data but with different train-test set to see [effect of sampling])

- command : **IDprediction -f testset.csv**

- It will combine result of each cluster for every test case run. Further, it will store combine result of every test case and plot pdf result.
- **Output :**
  - *./all_test_result/result_i/merge_dataset.csv*
    Combine result of all cluster for each individual test case.
  - *./all_test_result/final.csv*
    Combine result of every test-case.
  - *./error_prediction_plots*
    It has pdf of errors for different data points
  - *./prediction_plots*
    Generates pdf plots of prediction for different data points

4.1.13 -p : *'Plot'* frequency of the parameters

- command : **IDprediction -p coefficient.csv**

  While running the regression, data points will be splitted randomly. Due to split and change in the data points, obtained coefficient will get affected. To visualize variations in coefficient and find out average coefficient value this command is utilized. Gives result in the form of plot.

  Use this command in the directory where coefficient file exist.
- **Output:**
  - ./coefficient_histogram_plots
    it contains all histogram plots

* ./result/
  it contains file output_result.txt which stores output printed on console screen.

4.1.14 -o : *'Other'* dataset than fuel

– command : **IDprediction -c 0.05 -l 10 -o anyFile.csv**

If you want to **run tree based algorithm on any dataset** other than fuel then this command it really useful. To run file extra 'feature_selection.py' file has to be provided with relevant feature column. Example of this file is given in ./Standard_data directory.

Don't forget to make changes in 'feature_selection.py file':

* Change feature values relevant to column
* In column selection method change the provide the column name with 'Constant'

– **Output:**

* It will be same as in case of **'-t'** flag

5 Examples:

If you have directly jumped to this section then first follow instruction given in section 4 and configure your set-up. If done! Go ahead and try out yourself.

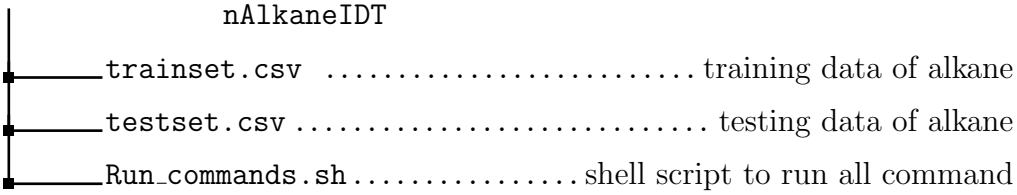
– **Currently you should be in './Data_driven_Kinetics'**

5.1 Example:1 - nAlkaneIDT

– Run the following commands to work with n-alkanes data:

```
cd TryYourself/nAlkaneIDT/
```

Directory has:



– To train the model (10% accuracy and using tree type regression based clustering algorithm) using n-alkanes training data, run the following command:

```
IDprediction -c 0.1 -t trainset.csv
```

– The output will be obtains as given in section 4.1.9

– Let's test the training model by running following command:

```
IDprediction -e testset.csv
```

– The output of test result will be same as given in section 4.1.10

– Below commands can be used to get results.

```
chmod +x Run_commands.sh
source ./Run_commands.sh
```

– **Change in feature selection.**

Similar procedure can be used with './OctaneHexadecane_TestData'.

Make changes in feature_selection and column_selection methods.

or

'feature_selection.py' file with appropriate changes.

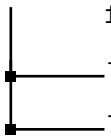
(To understand this let's consider different dataset other than fuel)

5.2 Example:2 - WineQuality

– Run the following commands to work with any dataset:

```
cd TryYourself/WineQuality/
```

Directory has:



`feature_selection.py` ..... edit this file for feature selection

`trainset.csv` ..... training data of alkane

`testset.csv` ..... testing data of alkane

– To train the model (10% accuracy and using tree type regression based clustering algorithm) using training data, run the following command:

```
IDprediction -c 0.1 -o trainset.csv
```

– The output will be obtains as given in section 4.1.9

– Let's test the training model by running following command:

```
IDprediction -e testset.csv
```

– The output of test result will be same as given in section 4.1.10