Data-driven framework to predict ignition delays: Manual

Pragneshkumar R Rana, Sivaram Ambikasaran,
Krithika Narayanaswamy

December 12, 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 explanations are 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 data-driven framework is designed to train the model and make prediction for ignition delay data. The idea behind the framework is to cluster the data based on similarity and generate regression models using multiple linear regression. Gaussian mixture modeling, tertiary tree clustering are implemented to generate cluster and Multiple regression method is used to generate regression models upon it.

1.1 Multiple regression:

The framework builds model by solving the below equation.

$$y_i = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n \tag{1}$$

1.2 Gaussian mixture modelling and regression:

The accurate models can be obtained by dividing the data into more than cluster and then generating regression models upon obtained clusters. GMM algorithm is implemented to generate clusters of data. The GMM algorithm is implemented using scikit-learn [1]. The identification of clusters is performed using the method mentioned in Error based tertiary tree model 1.3.

1.3 Spath's Algorithm

Spath algorithm train the model till accuracy is achieved or till maximum iteration performed, provided number of base models. It exchanges the data points among clusters. The full mentioned in this article [2].

1.4 Error based tertiary tree model and regression

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 clusters may also give correlation if they satisfy the specified relative error criteria; otherwise, they will be divided re-cursively by following step:1-6 until specified relative error criteria are not fulfilled.

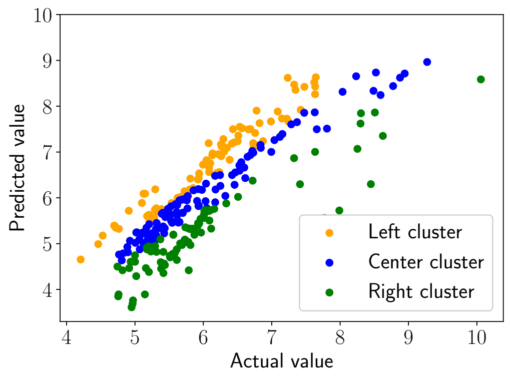
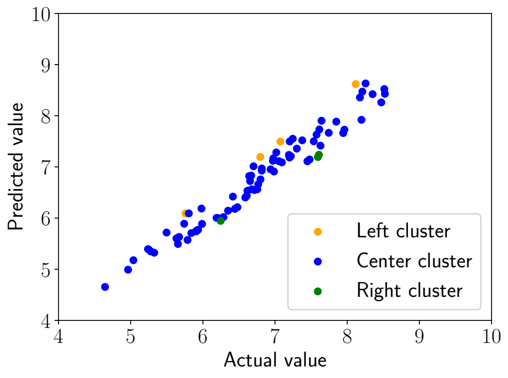
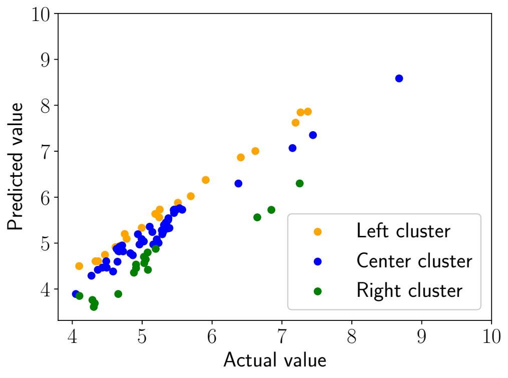


Figure 1: Division of cluster based on relative error in prediction and sign of prediction error (1^{st} level regression). Blue Data points satisfy 5% criterion of relative error. Orange and Green data points have higher error than the specified threshold value.



(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 values are 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 they satisfy the specified relative error criteria else, it will further divided 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.

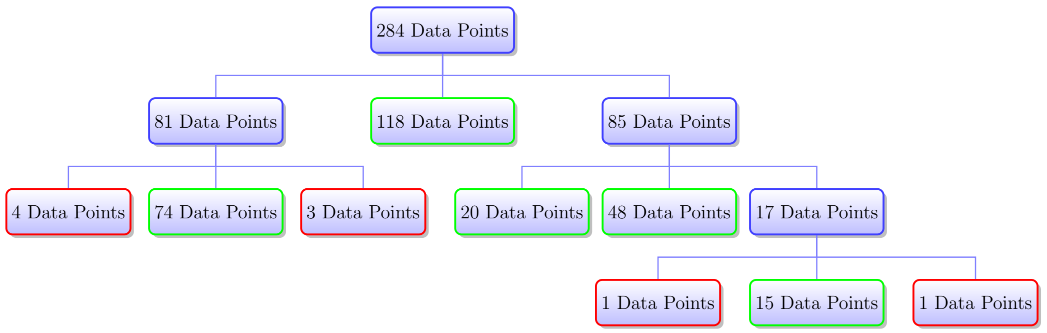


Figure 3: Division of 284 ignition delay data points into different clusters after applying recursive regression based clustering tree algorithm. Terminal nodes with the green border generate resultant prediction models, whereas terminal nodes with the red border have fewer data points than required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

Once all clusters are generated, identification of cluster for new/unknown data point is a 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 points are 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.

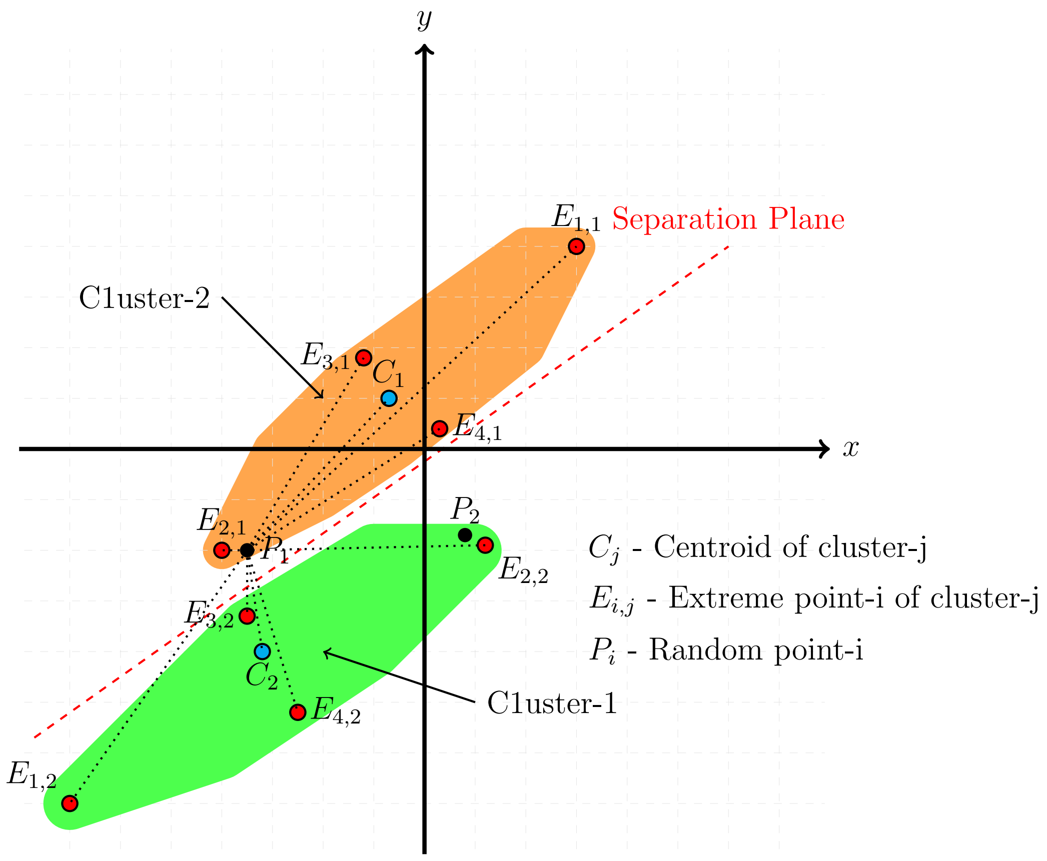


Figure 4: Use of extreme points along with centroids for correct assignment of cluster or model to the data point. Assignment of cluster only through the centroid is erratic which is illustrated by point P_1 and P_2. Point P_1 is part of cluster-1. But, if cluster assignment is done only using centroid 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:

```
Data_driven_Kinetics
├── data ............................. contains raw data
│   ├── Abalone
│   ├── BostonHousingPrice
│   ├── nAlkaneIDT
│   └── WineQuality
├── example ...  It has worked out examples with output.
│   │            Above data is utilized to generate
│   │            training and testing set.
│   ├── Abalone
│   ├── BostonHousingPrice
│   ├── nAlkaneIDT
│   ├── OctaneHexadecane_TestData
│   └── WineQuality
├── src ........................ contains source code files
├── TryYourself...  It has training and testing set as
│   │               given in ./example.  Look at section 5
│   │               and try out yourself.
│   ├── Abalone
│   ├── BostonHousingPrice
│   ├── nAlkaneIDT
│   ├── OctaneHexadecane_TestData
│   └── WineQuality
├── INSTALL.sh......................to install dependency
├── Manual.pdf
├── README.md
└── Run.sh ............ script parse and call python modules
```

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

- After downloading (cloning) the application, put it in the **./home** directory.
- Install all the dependencies. Dependencies 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
  ```

  Also, make **Run.sh** file executable by following command:

  ```
  chmod +x Run.sh
  ```

- After installing all the dependency, open **.bashrc** file and copy-paste following command at end of the file.
- If repo is cloned (copied) in any path then make appropriate changes in **.bashrc** file. Given below,

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

Change '/PathToDir/.../' with your folder path.

- If repo is cloned (copied) in **./home** then make given changes in **.bashrc** file.

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

- **IMPORTANT:** 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.

- Command to run the code:

  IDprediction --flag argument

4.1 Available options and flags:

4.1.1 --algo : *'Algorithm '* selection

- command :

```
IDprediction --algo multi
```

This command selects the algorithm.Currently 4 modules are implemented.

1. Multiple regression - **multi**
2. Guassian Mixture modelling with regression - **GMM**
3. Spath's Algorithm - **spath**
4. Error based tertiary tree - **tree**

4.1.2 --train : *'Train'* model on given IDT data

- command :

```
IDprediction --algo GMM --train alkane.csv
```

This command will train the model using passed file as argument provided algorithm for ignition delay data.

If you want to train algorithm with on any dataset other than fuel then 'feature_selection.py' file has to be provided with relevant feature columns.

4.1.3 --test : *'Test'* model on given IDT data

- command :

```
IDprediction --algo GMM --test testset.csv
```

This command will perform testing using trained model on passed data file as argument.

If you want to test algorithm with any dataset other than fuel then 'feature_selection.py' file has to be provided with relevant feature column.

4.1.4 --sfile : *'Source'* file to analyze the data

- command :

```
IDprediction --a pressure --sfile trainset.csv
```

This command will be used to pass source file using which clusters are generated. While analyzing cluster properties like temperature and pressure, source file can be passed through this command.

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

- command :

```
IDprediction --a manual --sfile trainset.csv
```

- To analyze range of properties for ignition delay. This module has five options,

   1. **manual** - to perform manual analysis to identify range of parameter
      Select fuel by giving corresponding index as input
      Input equivalence-ratio from available options
      Input pressure value from available options
      By default it will consider 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' then input the desired range else input:'n'

Once clusters are obtained after training the model, analysis can be performed over certain properties among cluster. Below commands are available to perform the task,

2. **pressure** - to analyze range of pressure covered by individual fuel comparing all the clusters.
3. **temperature** - to analyze range of temperature covered by individual fuel comparing all the clusters.
4. **parameter** - to analyze range of normal and transformed features separately for all the clusters.
5. **distribution** - to analyze fractional distribution of data comparing all the clusters.

- **Output :**
  - ./result/AnalysisResult/

    It will generate the plots based on the given choice of command. The plots can be found out in this directory.

4.1.6 --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 of Primary-Primary carbon bonds
    - **P_S** : Total number of Primary-Secondary carbon bonds
    - **P_T** : Total number of Primary-Tertiary carbon bonds
    - **P_Q** : Total number of Primary-Quaternary carbon bonds
    - **S_S** : Total number of Secondary-Secondary carbon bonds
    - **S_T** : Total number of Secondary-Tertiary carbon bonds
    - **S_Q** : Total number of Secondary-Quaternary carbon bonds
    - **T_T** : Total number of Tertiary-Tertiary carbon bonds
    - **T_Q** : Total number of Tertiary-Quaternary carbon bonds
    - **Q_Q** : Total number of Quaternary-Quaternary carbon bonds
    - **P_H** : Total number of Primary carbon - Hydrogen bonds
    - **S_H** : Total number of Secondary carbon - Hydrogen bonds
    - **T_H** : Total number of Tertiary carbon - Hydrogen bonds
    - **Fuel** : Fuel SMILES

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

4.1.7 --h : ***'Histogram'*** plots of parameters for each separate fuel

- 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.

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 regressors are less then specified value.

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

- command :

```
IDprediction --algo multi --train trainset.csv --r True --s 0.1
```

This Flag **has to be passed with True or False** .

True : Backward elimination will be activated

Flase : Backward elimination will be deactivated

- **Default : False**

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

- command :

```
IDprediction --algo multi --train trainset.csv --r True --s 0.1
```

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**

4.1.10 --c : *'Criteria'* for separation

- command :

```
IDprediction --algo tree --train trainset.csv --c 0.1
```

Flag has to be passed with error based clustering regression. Criteria value 0.1 indicates relative error of 10%. Criteria plays a 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**

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

- command (either one) :

```
IDprediction --algo tree --train trainset.csv --c 0.1 --l 10
IDprediction --algo tree --train trainset.csv --c 0.1 --l True
IDprediction --algo tree --train trainset.csv --c 0.1 --l False
```

  Flag has to be passed with multiple linear regression. If you pass 10 as value then number of reference points 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**

4.1.12 --algo multi : *'Multiple'* linear regression

- command :

```
IDprediction --algo multi --train trainset.csv --r True --s 0.1
```

  It will do multiple regressions 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-directories 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 gets 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}

4.1.13 --algo tree : *'Tree'* type error based clustering algorithm with regression(error based clustering)

- command (either one) :

```
IDprediction --algo tree --train trainset.csv --c 0.1 --l 20
```

  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. Each file contains average calculated value of features associated with cluster-data.
  - *./plots/*

    Directory contains several tree based plots which are 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/centroids/*

  It contains centroid location files for all the clusters.

– *./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}

4.1.14 --n : *'Number'* of clusters

- command :

```
IDprediction --algo spath --train trainset.csv --i 100 --n 3
IDprediction --algo GMM --train trainset.csv --i 100 --n 2-10
```

This Flag is used to define number of clusters to be created by algorithm. In case of GMM, range of cluster can be passed to get AIC, BIC plots.

- **Default : 2**

4.1.15 --i : *'Maximum Iterations'* in spath's algorithm

- command :

```
I IDprediction --algo spath --train trainset.csv --i 100 --n 3
```

This Flag is used to Maximum number of iterations in spath's algorithm.

- **Default : 2**

4.1.16 --algo spath : *'Spath's'* algorithm with regression(error based clustering)

- **DON'T FORGET TO PASS 'feature_selection.py' FILE AS IT DOES NOT CONTAIN 'constant' FEATURE.**
- command :

```
IDprediction --algo spath --train trainset.csv --i 100 --n 3
```

- **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)
    - **./cluster_ref** - reference points for identification of the cluster
    - **./centroid**- it contains files for final clusters. Each file contains average calculated value of features associated with cluster-data.
  - *./result/centroids/*

    It contains centroid location files for all the clusters.
  - *./result/coefficients/cluster_data/*

    It contains data of all the generated cluster data including intermittent.
  - *./result/coefficients/final_cluster/*

    It contains data of all the end clusters.

- *./result/coefficients/*

  stores coefficient obtained after multiple regression. **It has data of only final clusters.**

- *./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 .

4.1.17 --algo GMM : *'Gaussian mixture modelling'* algorithm with regression(error based clustering)

-
- command :

```
IDprediction --algo spath --train trainset.csv --i 100 --n 3
```

- **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)
    * **./cluster_ref** - reference points for identification of the cluster
    * **./centroid**- it contains files for final clusters. Each file contains average calculated value of features associated with cluster-data.
  - *./result/centroids/*

    It contains centroid location files for all the clusters.

– *./result/cluster_data/*

It contains data of all the generated cluster data including intermittent.

– *./result/final_cluster/*

It contains data of all the end clusters.

– *./result/coefficients/*

stores coefficient obtained after multiple regression. **It has data of only final clusters.**

– *./result/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/console_output/*

Output printed on console screen also get stored on the in the output_result.txt file

– *./result/cluster_data_before_regression/*

Data of all the cluster just before partition of the node.

– *./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}

4.1.18 --test : *'External'* Dataset used for prediction of ignition delay

- command [**works with any algo**] :

```
    IDprediction --algo tree --test 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 values), y_actual (actual ignition delay vales) and relative error between them. All files are cluster specific means data point is assigned to only one 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.

4.1.19 --k : To predict external dataset result of ignition delay for 'k' training modules (Modules are generated by different training sets) 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 '--test' flag +
- **Output :**
  - *./all_test_result_i*
    Result of all the test datasets will be store here.

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

- command :

```
    IDprediction --f testset.csv
```

- It will combine result of each cluster for every test case run. Further, it will store combined result of every test case and plot pdf result.
- **Output :**
  - *./all_test_result/result_i/merge_dataset.csv*

Combined result of all cluster for each individual test case.

- *./all_test_result/final.csv*
  Combined 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

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

- command :

```
IDprediction --p coefficient.csv
```

  While running the regression, data points will be split 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 exists.

- **Output:**
  - ./coefficient_histogram_plots
    it contains all histogram plots
  - ./result/
    it contains file output_result.txt which stores output printed on console screen.

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! then 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:

nAlkaneIDT

trainset.csv	training data of alkane
testset.csv	testing data of alkane
Run_commands.sh	shell script to run all command

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

```
IDprediction --algo GMM --train trainset.csv --c 0.1
```

- The output will be obtained as given in section 4.1.17
- Let's test the training model by running following command:

```
IDprediction --algo GMM --test testset.csv
```

- The output of test result will be same as given in section 4.1.18
- 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 a 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 --algo GMM --train trainset.csv --c 0.1
```

- The output will be obtained as given in section 4.1.17
- Let's test the training model by running following command:

```
IDprediction --algo GMM --test testset.csv
```

- The output of test result will be same as given in section 4.1.18

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[2] Helmuth Späth. Algorithm 39 clusterwise linear regression. *Computing*, 22(4):367–373, 1979.