ChBE 6745/4745 - Data Analytics for Chemical Engineers
Semester Project
---Improved Model

11/05/2020

Po Hsien Hsu
Liyun Ren
Ziheng Shen
Jason Yao
Christian Herridge

Several modifications have been made on top of the baseline model we have.

1 Modification of Feature Extraction Based on Physical Meaning

A modification has been made to the code extracting feature matrix in "Data Preparation" module. The changes include:
(1) Corrected a mistake in function that calculates the ignition temperature.
(2) Baseline correction function now only incorporates flattening and smoothing the curve, the minor peaks are no longer removed. This change is trying to keep the baseline correction results as close as the results calculated by commercialized software.

2 Principal Component Analysis (PCA) for new feature extraction

In the previous data treatment processes, the feature matrix was composed of 5 features extracted based on physical meanings. However, it's worth exploring the prediction power between those manually extracted features and data-driven features, especially for the spectra-like data we are dealing with. Therefore, Principal component analysis (PCA), in turn, has been performed and several principal components were extracted from the raw MCC data (i.e. temperature – Heat release rate data) with and without baseline correction. The extracted principal components, as well as the manually extracted features, will all be considered in the prediction.

The HRR values (y values) of all datapoints in each MCC curve, not surprisingly, were treated as input features of PCA as it is HRR values that determine the shape of curves. It's worth noting that the temperature values (x values) of each observation are not aligned to each other. To improve the accuracy of PCA, a new grid of temperature values was created and HRR values of all observations were predicted using a 1-D linear interpolation based on the new temperature grid. As a result, 10 PCs were extracted out of 1399 features. As shown in **Figure 1**, the first 6 PCs contributes almost 90% of the total variance. The 10 PCs were reconstructed to the original coordinate, which is demonstrated in **Figure 2**. Based on the figures, there was no big difference between PCs from raw data and corrected data.

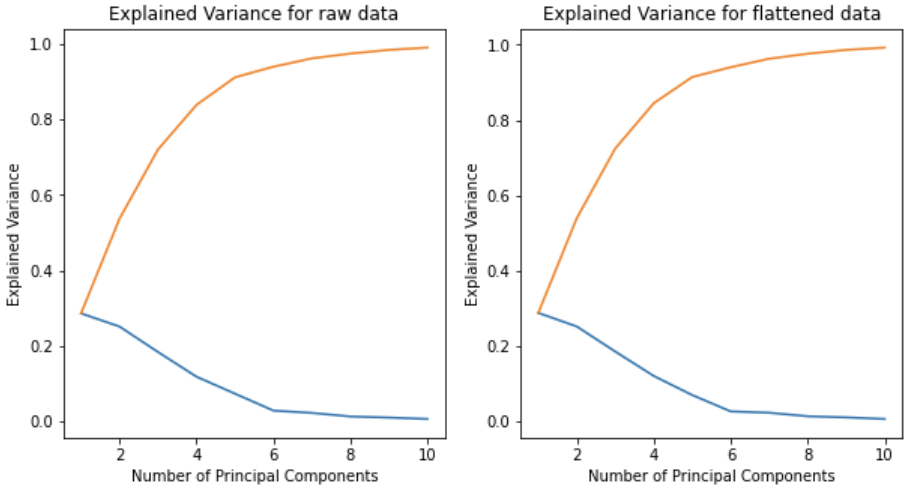

Figure 1 Explain variance and cumulative variance of the 10 principal components calculated from (a) original MCC data without any correction (b) baseline corrected MCC data

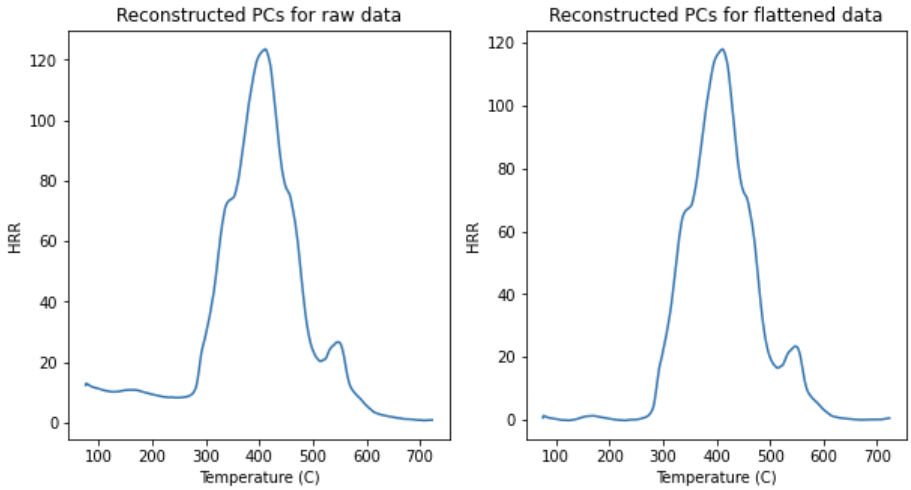

Figure 2 Reconstructed curve based on PCs for (a) original MCC data without any correction (b) baseline corrected MCC data

3 Generative method to expand the size of feature matrix

In the FR labels, there are samples where testing results are uncertain, which are labeled as "FR_label=2". During the FR testing, the results indicate these materials show borderline performances, which means these materials sometimes passes FR testing and sometimes fails. To refine the supervised models performed previously, the generative models are also explored in the improved model part. Generative models are unsupervised since they do not require labels for the output data. Mixture models in general don't require knowing which subpopulation a data point belongs to, allowing the model to learn the subpopulations automatically. Gaussian mixture models (GMM) was selected for improved modeling. Bayesian Information Criterion (BIC) are used to determine the best approximations in GMM. In GMM, the optimal number of components is 6 as demonstrated in **Figure 3**. It turned out that the number did vary from 5-7 due to the sparse input data. **Figure 4** shows the pair-plot generated in terms of FR labels. The distribution of the original data is well preserved even with the addition of 100 points. "FR_label=2" class was dropped because it was too sparse to begin with (2 observations).

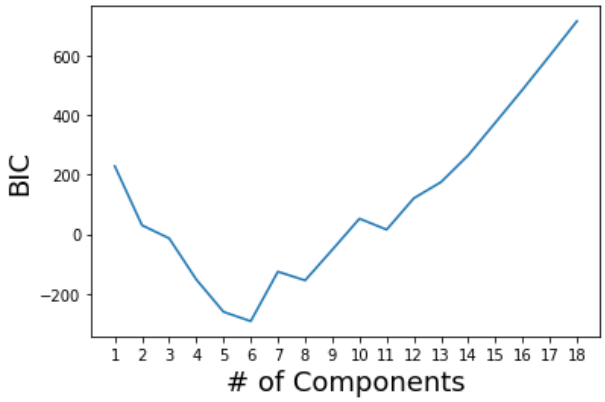

Figure 3 Optimization for number of components based on BIC

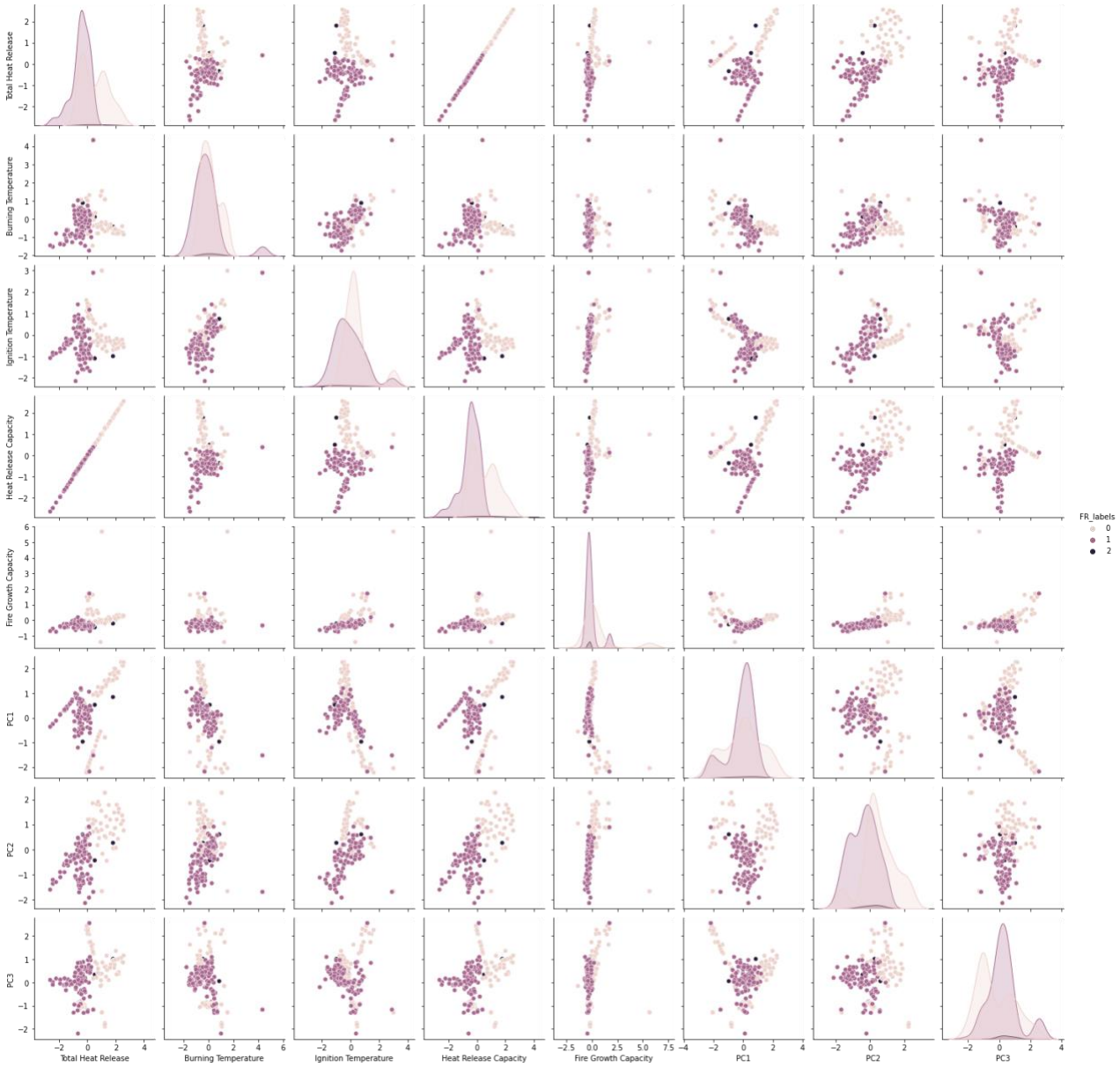

Figure 4 Pair plot of feature matrix with GMM generated data

4 New prediction models in addition to baseline model

4.1 Baseline model

The support vector classifier was the original tool we used for our baseline model, and is a powerful tool commonly used in classification problems. We chose to vary the hyperparameters which varies regularization strength (Cs) as well as the kernel coefficient for our radial basis function kernel used in the algorithm (gamma).

4.2 Alternative models

For our alternative classifier models for performance comparison, we attempted to fit our data using a decision tree classifier, a random forest ensemble classifier, and a k-nearest neighbors classifier. All our models are written in a well commented standard format in order to enable ease of reading and review.

We sent each model through a GridSearchCV over relevant hyperparameters which we think will increase likelihood of successful performance.

4.2.1 Decision Tree Classifier

The decision tree classifier attempts to learn decision rules in order to predict values for the variables targeted. For this model, we chose to vary the maximum depth of the tree (max_depth) as well as the function which measures the quality of each split (criterion). The maximum depth parameter was tailored specifically to the size of our dataset.

4.2.2 Random Forest Ensemble Classifier

The random forest ensemble classifier is a collection of decision trees which has good predictive ability as well as control against overfitting the data. For this model, we have chosen four hyperparameters for our parameter grid: the number of trees in the forest (n_estimators), the function which measures the quality of the splitting (criterion), and maximum depth of each tree (max_depth), and the number of features to look for when determining the best split (max_features).

4.2.3 k-nearest Neighbors Classifier

The k-nearest neighbors classifier implements learning based on the k nearest neighbors to each point and is a flexible tool for classification. We vary hyperparameter which decides how many of the closest neighbors are used in the algorithm (n_neighbors).

4.3 Comparison of model performances

Table 1 Classification Metrics and Scores for Current Models

Model	Dataset	Accuracy	Precision	Recall
SVC (Baseline model)	Validation	0.40	0.44	0.28
	Full	0.73	0.72	0.61
SVC (with GMM)	Validation	0.82	0.54	0.57
	Full	0.91	0.61	0.62
	Original	0.83	0.56	0.60
Decision Tree	Validation	0.74	0.74	0.74
	Full	0.89	0.93	0.82
	Original	0.88	0.93	0.82
Random Forest	Validation	0.72	0.51	0.49
	Full	0.93	0.62	0.63
	Original	0.83	0.56	0.60
KNN	Validation	0.82	0.55	0.57
	Full	0.88	0.59	0.60
	Original	0.80	0.54	0.58

The generative model was successful in improving the results of our model's performance. As seen in **Table 1** above, the accuracy, precision, and recall classifier metrics were scoring poorly before we applied the method to generate new data. Comparing the SVC performance before and after GMM the performance of our model has improved, specifically in the accuracy metric. When comparing the SVC model to the alternative approaches mentioned above, there was little or no improvement in metric scores between most of them. The decision tree classifier was the only model that stands above the others in terms of performance, with an accuracy of 0.88, a precision of 0.93, and a recall of 0.82 when predicting the original dataset.

5 Problems and paths forward

5.1 Forward selection to determine most predictive PCs

Due to time limitation, we only extracted the 10 PCs and combined the first three with the manually extracted feature matrix. Forward selection will be performed to determine 3-5 PCs that contribute most to the prediction. It will be calculated after data generation (i.e. with GMM/KDE method).

5.2 Comparison between PC features and physical features

In the current codes, a feature matrix incorporating 5 physical features and 3 PCs with GMM was used as input for the prediction models. As a comparison, a feature matrix with solely physical features and one with solely PCs will be expanded by GMM (KDE), following by prediction tests with the same models. It could give us insights on the performance between data-driven features and manually extracted features.

5.3 Complete Kernel Density Estimation (KDE) method

In addition to GMM, Kernel Density Estimation (KDE) method is under development as an alternative generative method to expand our original dataset. The new dataset will be used as input for the prediction models and the results will be compared with those obtained from GMM model.

5.4 Effect of train-test split on prediction performance

It turned out that the prediction metrics vary depending on the randomness of train-test split. The accuracy scores fluctuated around 0.2. It would probably still be due to the small size of our dataset. A larger generated dataset, for example, a size of 2000, might be used to test the stability of train-test split.