What *is* the Best Classifier? What *is* the Best Regressor?

Yixuan Li Rui Li Yang Liu Hao Chen

Abstract

This report investigates two questions. First, for a given selection of data sets, can we say what is the 'best' classifier or the 'best' regressor in terms of good predictions? How much does the answer depend on the particular selection of data sets? How much does the answer depend on our computational constraints? We investigate these questions using data sets from the UCI repository. Second, we compare the interpretability of a decision tree classifier to that of a convolutional neural network. We compare the decision tree visualization to 'activation maximization', a technique to gain insight into the kinds of inputs that deep neural networks respond to. At last, we compare four different strategies for reducing the feature dimensionality and apply it to our experiments.

1. Introduction

As there are more and more data collecting from each domain, data analytics is becoming a significant part for studying the data. Supervised machine learning is encouraged by this need, as the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features [3]. More machine learning models and hyper-parameter searching approaches are introduced to solve diverse types of datasets and effectively tune the hyper-parameters which drive performance. As the saying goes, "Industry specializing in surgery", when doing machine learning projects, two of the most important things to be considered are hyper-parameters tuning and model interpreting. Though we have learned the principles and the characteristics of different models, it may still not be equivalent for us to deal with real world situations easily. So in this project, our goal is to dive into model selections and model performance on various datasets, and investigate the interpretability of decision tree classifier and convolutional neural network (convnet). Firstly, we use 8 classification models and 7 regression models to evaluate each of their performance on 10 different datasets to find out the overall best model and their sensitivity to certain types of data. Then, we train a decision tree and a convolutional neural network to do the classification based on CIFAR-10 dataset [4]. And we visualize the best decision tree structure, and we adapt back-propagation to modify the blank input image to make the activation of a certain class as big as possible. This is known as 'activation maximization'. By doing this, we can roughly know how decision tree is going to split to fit the input images, and what kind of input images would maximize activation for a certain class.

In the next section, we will start by explaining our methodology for data loading and data preprocessing techniques, then we move on to our hyper-parameter search strategies, training and testing. We then present our experimental analysis and results on classification and regression datasets as well as on CIFAR-10 datasets. In the novelty section, we will discuss the four strategies of dimensionality reduction. In section 3, we will draw our attention on experimental conclusion and potential implications.

2. Methodology & Experimental Results

There are two main types of supervised machine learning problems which we are going to discuss in this paper, classification and regression. On each data set, we will perform data preprocessing techniques, such as feature normalization. In the first and second parts, we perform a two-phase cross validation on different datasets during the period of hyper-parameter search. The performance of classifiers and regressors will be evaluated on multiple success metrics. In addition, we prefer to show the evaluation results horizontally (over datasets) and vertically (over models) so as to demonstrate the 'best' classifier or regressor. This will allow us to summarize performance of models by visualization. Furthermore, to gain more insight on why a particular model performs well on the data set, we choose a few datasets to invest on and purpose our investigations.

In the third part, we intend to evaluate two models, decision tree and convnet, for image classification on their interpretability as well as accuracy. Our approach is to visualize the model structure and interpret by using 'activation maximization' technique. Additionally, we trained 3 convnets in order to analyze our models and their activation images for each class label.

In the fourth part, we will discuss dimensionality reduc-

Model	Parameters	Raw
KNN	n_neighbours	5
KNN	weights	'uniform'
SVC	C	1.0
SVC	gamma	'auto'
SVC	kernel	'RBF'
DTC	criterion	'gini'
DTC	max_depth	None
RFC	criterion	'gini'
RFC	n_estimators	100
RFC	max_depth	None
ABC	n_estimators	50
ABC	learning_rate	1.0
LR	C	1.0
LR	max_iter	100
GNB	var_smoothing	1e09
NNC	hidden_layer_sizes	100
NNC	max_iter	200

Table 1. The Parameters we modified in classification models.

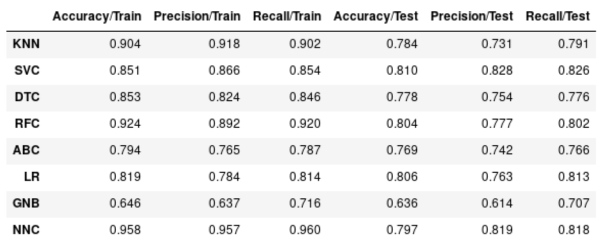
tion. For most datasets, input features are less than 200. But the situation changed for the 10th dataset Merck Molecular Activity Challenge. The number of input attributes is 5877 for the first sub dataset and 4306 for the second sub dataset. We use four different models such as PCA to reduce the feature dimensionality and compare the results individually. The specific results will be discussed later in the novelty component.

2.1. Classification Experiments

There are 10 different datasets for the classification phase. They are Diabetic Retinopathy (life), Default of credit card clients (business), Breast Cancer Wisconsin (life), Statlog [Australian credit approval] (finance), Statlog [German credit data] (finance), Steel Plates Faults (physical), Adult (Social), Yeast (life), Thoracic Surgery Data (life), Seismic-Bumps. Generally speaking, 3 are from financial and business area and 4 come from life area. Since all of them are famous datasets in the machine learning domain, the results of our experiment can be persuasive.

There are 8 different classifiers in this stage we need to train for every single dataset. The specific hyper-parameters for each model in the cross validation are shown in Table 1,

Two-phase cross validation. In order to find the best estimator for every model, we decided to use 3-fold GridSearchCV class from sklearn [5]. Then we use it fitting the training set for the first time. It help us find the optimal order of magnitude for each hyper-parameter. After we got the best order of magnitude, we generated different normal distributions around the best order of magnitudes that we got from the grid search phase. After that, we passed those



	Accuracy/Train	Precision/Train	Recall/Train	Accuracy/Test	Precision/Test	Recall/Test
KNN	0.904	0.918	0.902	0.784	0.731	0.791
SVC	0.851	0.866	0.854	0.810	0.828	0.826
DTC	0.853	0.824	0.846	0.778	0.754	0.776
RFC	0.924	0.892	0.920	0.804	0.777	0.802
ABC	0.794	0.765	0.787	0.769	0.742	0.766
LR	0.819	0.784	0.814	0.806	0.763	0.813
GNB	0.646	0.637	0.716	0.636	0.614	0.707
NNC	0.958	0.957	0.960	0.797	0.819	0.818

Figure 1. Average results for classifiers.

normal distributions to a new RandomizedSearchCV object and fit it with the training data for another time. Then if it met our expectations, we can find the best hyper-parameters for particular model.

When we finished the two-phase hyper-parameter search, we got all the trained models and result tables for 10 datasets. For some datasets, the execution time of an individual classifier exceeds three minutes. We will consider that the classifier failed for that dataset. Next, we are going to specify the three different criteria, Accuracy, Precision, and Recall(TPR). Before getting into the details of each dataset, we generated the averaged result for all the dataset with no weight. The result table is shown in Figure 1. From the averaged result, we learned that the three most accurate classifiers for the training set are neural network classifier(NNC), Random forest classifier(RFC) and k-nearest neighbours classifier(KNN) with 95.8%, 92.4% and 89.9% respectively. For the test set, accuracy overall decreased rapidly, it declined to around 80% for most classifiers. Gaussian Naive Bayes(GNB) got the worst result on average with 63.6% on test set. Since the result barely showed the degree of decrease between the training and testing sets, we generated another ratio figure to show the relationship between the training and test set. As we can see in Figure 2. Even though KNN, RFC and NNC performed pretty robust on the training set, their decline on the test set are top 3 as well. So generally speaking, GNB can be more robust for through different data sets. KNN, NNC and RFC could be overfitted in our tuning process.

From the perspective of precision and recall, the situation did not change too much. Even though the score of both criteria are totally different. Recall are more significant on health and life area since it is usually more important to recognize a person with potential to have a disease rather than recognizing a healthy person to be a patient. Whereas that precision is more focusing on the predicted positive whether they are true positive. Most of the results for the recall and precision followed the shape of accuracy. But interestingly, we found that the recall of logistic regression barely changed throughout all the 10 datasets output by the models we generated.

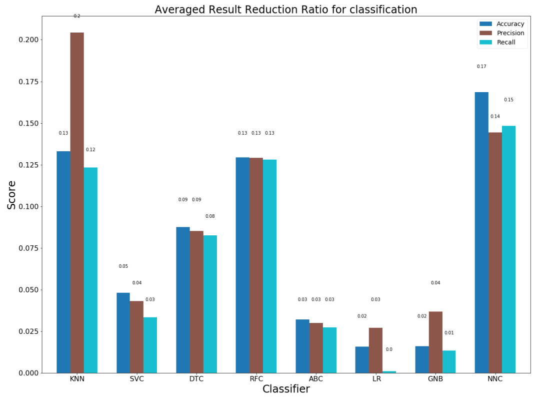


Figure 2. Averaged Result Reduction Ratio for classifiers.

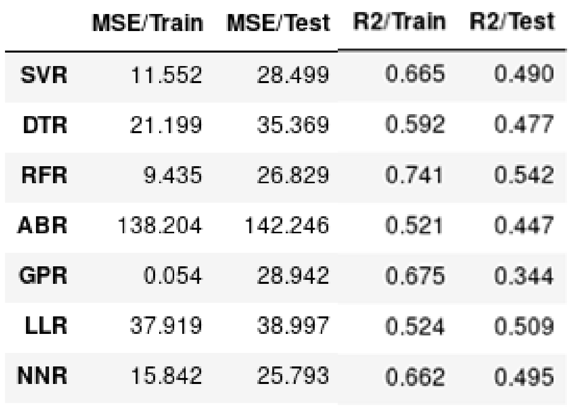
For data set "Statlog_German_credit_data", this data set has it own cost matrix which is

$$\begin{bmatrix} & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 5 & 0 \end{bmatrix}, (1 = \text{Good}, 2 = \text{Bad})$$

Each row indicates the actual labels, and each column indicates our prediction. It means that it is much worse to predict a bad one to be a good one. Instead of using the accuracy score for model selection. Here we change the criteria to the dot production of the cost matrix shown above and the confusion matrix. And the strategy is to find the model with the minimum of this production which could maximize the likely-hood function of the loss function. The truly loss is shown in Table above. In order to achieve this goal. We need to change the parameter of cross validation scoring by using our own loss function or score function. Sklearn is open sourced and really powerful and flexible on changing different score strategy based on our experiments.

The data set "Yeast" gave us the worst result on average, the reason behind it is more or less due to three main reasons. The first reason may be the dataset contains 10 labels and they are highly imbalanced. The most frequent label 'CYT' in the training set takes 311 of 991 spaces, while the least frequent label 'ERL' only shows up once in the training set. The second reason is that there are 2 columns of data holds a fixed number for most of the time. Even if it changes, the value does not have a huge drop or jump. For example, the value of class 'ERL' maintain 0.5 at 984 cases and only get a 1 value for a 9 times, which may confuse the classifiers, thus causing a bad classification performance. Moreover, the dataset is kind of small, for it only contains 1400 rows of data, which can easily over-fits.

While data set "Breast Cancer Wisconsin" has the best score on average. The results between training and test set are rarely changed and pretty high. We considered the reason why the result is so good mainly due to two reasons. Firstly, it is a binary classification which is much easier compared with multi-label classification like the "Yeast" data set. Secondly, the variance of the features are pretty high which means that it is so rich that it could contain much more information than others. It is well prepared and organized.



	MSE/Train	MSE/Test	R2/Train	R2/Test
SVR	11.552	28.499	0.665	0.490
DTR	21.199	35.369	0.592	0.477
RFR	9.435	26.829	0.741	0.542
ABR	138.204	142.246	0.521	0.447
GPR	0.054	28.942	0.675	0.344
LLR	37.919	38.997	0.524	0.509
NNR	15.842	25.793	0.662	0.495

Figure 3. Regression Average Results

Model	Parameters	Raw
SVR	C	1.0
SVR	gamma	'auto'
SVR	kernel	'rbf'
DTR	max_depth	None
DTR	min_samples_leaf	1
RFR	n_estimators	10
RFR	max_depth	None
ABR	n_estimators	50
ABR	learning_rate	1.0
GPR	alpha	1e-10
GPR	kernel	1**2*RBF(length_scale=1)
LLS	alpha	1.0
LLS	max_iter	None
LLS	solver	'auto'
NNR	hidden_layer_sizes	100
NNR	max_iter	200

Table 2. The Parameters we modified in regression models.

2.2. Regression Experiments

We train our regressors based on 10 different datasets : Wine Quality (Business), Communities and Crime (Social), QSAR aquatic toxicity (Physical), Parkinson Speech (Life), Facebook metrics (Business), Bike Sharing (Social), Student Performance (Social), Concrete Compressive Strength (Physical), SGEMM GPU kernel performance (Computer), Merck Molecular Activity Challenge (Life). The area of datasets for regression covered every aspect of daily life and the distribution is averaged. Machine learning has been applied in various fields around us.

We train 7 different regressors on every dataset. They are support vector regressor, decision tree regressor, random forest regressor, ada boost regressor, gaussian naive bayes regressor, linear regressor (Ridge from sklearn) and neural network regressor. The two-phase hyper-parameter search is identical to the classification phase. The specific hyper-parameters for each regression model in the cross validation are shown below in Table 2.

There is a special 'Merck Molecular Activity Challenge' dataset in the regression dataset, By following the requirements, after we extracted the data. We found the data is a sparse matrix. Most of the values in that input matrix are zeros, we tried to train regressors based on the raw data but never succeed. So we decided to use special strategy to reduce the dimension of the data. We then choose to deep dive this part as one of our novelty approach. The details of the approach is going to be discussed in section 2.4. As we transformed the raw data into lower dimensional data. We need to use the cost matrix given by the organization. The formulas are shown below:

$$R^2 = \frac{1}{2}\sum_{s=1}^{2} r_s^2$$

$$r_s^2 = \frac{[\sum_{i=1}^{N_s}(x_i - \bar{x})(y_i - \bar{y})]^2}{\sum_{i=1}^{N_s}(x_i - \bar{x})^2 \sum_{i=1}^{N_s}(y_i - \bar{y})^2}$$

The 'Merck challenge' from kaggle are formed by two datasets **ACT2** and **ACT4** by using a different evaluation matrix which are shown above. Here, we are going to make the average of squared pearson correlation as big as possible. [**?**]

After we found the optimal hyper-parameters for each model, we output all the results with respect to different dataset by using the criteria Mean Squared Error (MSE) and R^2 (coefficient of determination) Score. R^2 score is a comparable simple, seemingly intuitive metric that measures how well the linear model fits a set of observations. But only using R^2 score to evaluate models is not enough, We also need MSE to give us general feelings of how far the predictions are from their corresponding targets respectively.

We can easily find that adaboost get the biggest MSE overall which is much larger than other models. It did not perform well generally on the whole data set. Looking at MSE is not enough, by comparing, we also found that the relationship between targets and predictions for adaboost regressor can performed similar to all the other regressors. By comparing the results, we can From the Figure 4, we

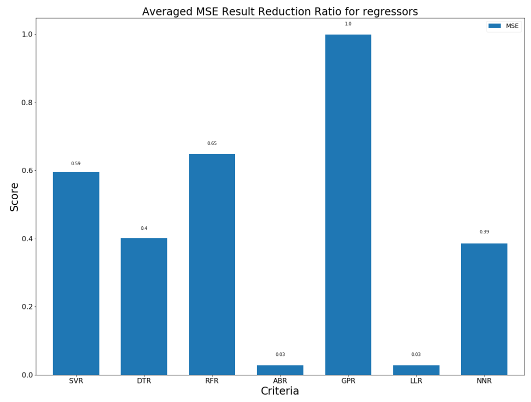


Figure 4. Averaged MSE Result Reduction Ratio for regressors

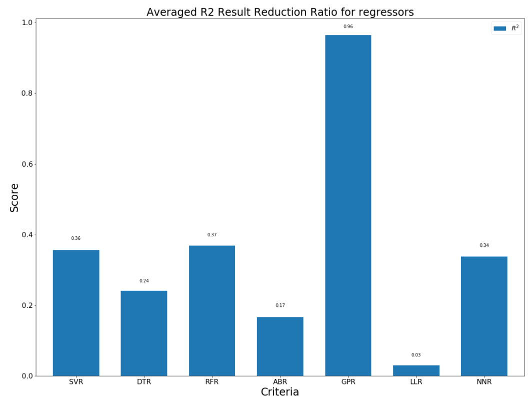


Figure 5. Averaged R2 Result Reduction Ratio for regressors

can see that the performance of Gaussian process model decreased a lot. But Adaboost seemed to be relatively stable. When inspecting Figure 5, Gaussian process fell down rapidly whereas Linear least Regressor is most stable.

When we jumped into each dataset, we found that some models of dataset with bad relatively terrible MSE score did not really performed that bad, i.e. ABR gave a huge MSE in "Bike Sharing" dataset which is around 100. But actually the R^2 score performed not bad compared with other models.

In evaluation of the regression dataset "student performance", we found its average performance on R2_score is the worst among all the regression datasets, which may be caused by the following reasons. Firstly, over half of the features in the dataset is of category type (17/32), and there are 8 features with only two values of 'yes' and 'no', which may not deliver the real character of the data. Secondly, because only the student-performance dataset is used in the regression, the amount of data is relatively small, which will also cause some regression models like GPR to get a bad performance.

2.3. Interpretability Experiments

In this section, we will use two different models to process CIFAR-10 dataset, decision tree classifier and convolutional neural network (CNN) classifier. After training the two models, we will show and compare the interpretability by visualizing them through different angle.

Before training decision tree classifier, we downloaded the dataset and combined the training batches into one giant Numpy array with $shape = (50000, 3072)$. For preparing the dataset for CNN classifier, we use *torchvision.datasets.CIFAR10* method whereas we set *download=False* since we have downloaded.

After the download step, we normalize the training and testing data to $[0, 1]$ for decision tree classifier, and $[-1, 1]$ for CNN classifier respectively. For CNN classifier, we also performed Random Erasing trying to achieve higher accuracy.

2.3.1 Decision Tree Classifier

The first model we trained is decision tree classifier. We use GridSearch to do hyper-parameter search for parameters *max_depth* and *criterion*, and we limit the tree depth up to 16 while training for completing training in reasonable time. The best decision tree classifier achieves the test accuracy of 30.26% on CIFAR-10 with $max\_depth = 8$ and $criterion = gini$. We need to set $average$ to "micro" or "macro" to get the recall score in multi-class classification, where $average = micro$ is calculated by counting the total true positives, false negatives and false positives, and $average = macro$ is calculated by counting each class's unweighted mean. When the dataset is balanced, $average = micro$ and $average = macro$ should give the same recall score, which is applied on CIFAR-10 dataset. The decision tree classifier gives recall of 0.350 on $50,000$ training images and 0.303 on $10,000$ test images.

2.3.2 Convolutional Neural Network Classifier

The second model trained on CIFAR-10 is convolutional neural network classifier (for the purpose of notation, we will call it CNN-1). In order to make trade-off between the network depth, test accuracy, iteration efficiency and network simplicity, we choose to use multiple VGG Blocks [2] as our reference, and we apply batch normalization after each convolutional layer, and apply dropout after MaxPooling layer in order to generalize well. Following the VGG blocks, we add two fully-connected layers, which is the similar strategy used in classic deep neural network such as AlexNet, LeNet. CNN-1 did get better test accuracy (77.30% after 30 epochs) than our best decision tree classifier (30.26%).

Model	Epoch #	Train Loss	Test Accuracy
base-CNN	10	1.020	56.55%
	20	0.238	64.35%
CNN-1	10	1.057	64.90%
	20	0.566	74.57%
	30	0.403	77.30%

Table 3. Base-CNN and CNN-1.

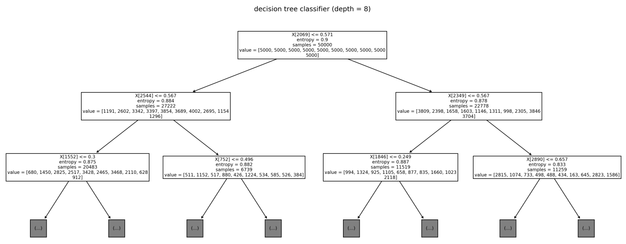

decision tree classifier (depth = 8)
X[2069] <= 0.571
entropy = 0.9
samples = 50000
value = [5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000, 5000
5000]
X[2544] <= 0.567
entropy = 0.884
samples = 27222
value = [1191, 2602, 3342, 3397, 3854, 3689, 4002, 2695, 1154
1296]
X[2349] <= 0.567
entropy = 0.878
samples = 22778
value = [3809, 2398, 1658, 1603, 1146, 1311, 998, 2305, 3846
3704]
X[1552] <= 0.3
entropy = 0.875
samples = 20483
value = [680, 1450, 2825, 2517, 3428, 2465, 3468, 2110, 628
912]
X[752] <= 0.496
entropy = 0.882
samples = 6739
value = [511, 1152, 517, 880, 426, 1224, 534, 585, 526, 384]
X[1846] <= 0.249
entropy = 0.887
samples = 11519
value = [994, 1324, 925, 1105, 658, 877, 835, 1660, 1023
2118]
X[2890] <= 0.657
entropy = 0.833
samples = 11259
value = [2815, 1074, 733, 498, 488, 434, 163, 645, 2823, 1586]
(...)
(...)
(...)
(...)
(...)
(...)
(...)
(...)

Figure 6. Decision Tree Classifier top 2 layers.

For the purpose of comparing the generalization of our CNN-1, we trained another CNN (for notation, we will call it base-CNN) with only one VGG block without doing batch normalization, followed by a dropout layer with smaller probability of zeroing out hidden neurons, and there are two fully-connected layers in the end as CNN-1 does. In total, we trained the CNN-1 in 30 epochs and the base-CNN in 20 epochs. The performance of both classifiers is shown below in Table 3.

The Table 3 shows that after 20 epochs, the base-CNN performs better than CNN-1 on training set while much worse on testing set. It generally means that the base-CNN overfits and CNN-1 classifier generalizes well.

2.3.3 Interpret the Decision Tree and CNN classifiers

For the interpretability of a decision tree classifier We can clearly see the threshold of each node and the number of training images being split into each node, as shown in Figure 6, by plotting the top 2 layers of the decision tree structure. This gives us the idea that decision tree classifier finishes classification based on certain threshold in certain position of data.

After generating the top 2 layers of decision tree structure, this give us the idea of how the decision tree finishes the classification based on certain threshold value, through overlaying its layer, it connects the threshold determination in nodes from layer thus get a connection between pixel features between layers to do the classification. But this prin-

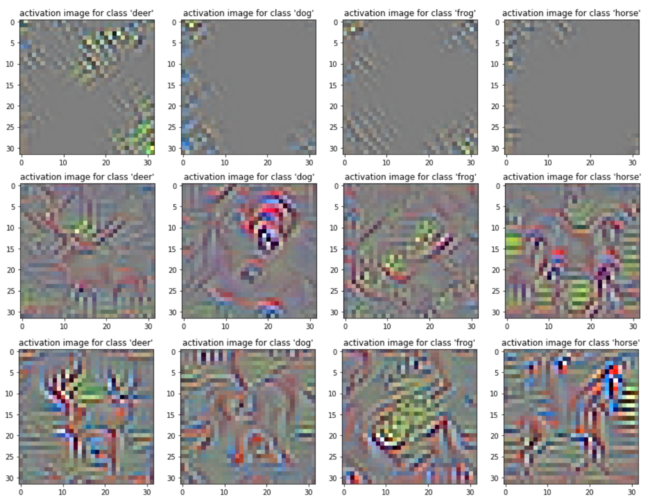

activation image for class 'deer'
activation image for class 'dog'
activation image for class 'frog'
activation image for class 'horse'

Figure 7. Base-CNN (1st row), CNN-1 (2nd row), CNN-2 (3rd row) activation map for classes.

ciple may not hold when the dataset itself is complicated, because the cifar10 dataset often contains images with the same class label, but from a different distance or angle, which may affect the image character in certain pixel, and may cause the decision tree model to make wrong decisions.

For the interpretability of CNN classifier We can gain intuition about the network by finding the optimal stimulus for each unit by performing gradient descent in image space to maximize the unit's activation. [6]. This is known as "activation maximization", aiming to maximize the activation of a certain neuron.

The Figures 7, below shows the activation map for class 'deer'. The model CNN-2 is the best model over our convolutional neural network experiments achieving the highest test accuracy of 81.40% in 16 epochs. CNN-2 has the exactly same architecture as CNN-1, but the only difference is that CNN-2 was not trained on Randomly Erased images.

As shown in Figures, Base-CNN's activation images for classes are vague than the other two. We can see the patterns for classes in CNN-1's activation images even if it is blurrier than CNN-2's, and CNN-2's activation images are best among the three classifiers.

In summary, decision tree classifier is a more naive approach to solving CIFAR10 dataset image classification problem. It is less interpretable and less accurate. Convolutional Neural Networks have excellent performance at the task of CIFAR10 image classification problem. Moreover, it is more interpretable by showing the activation image for each class in input layer. Similarly, we can understand the operations of a convnet by interpreting the feature activity in intermediate layers [6].

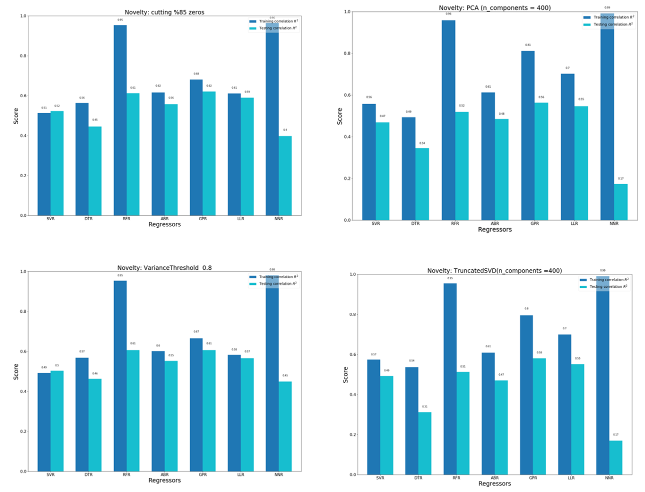


Figure 8. Novelty,Cutting 85% (upper-left image), Standard PCA (upper-right image), VarianceThreshold (lower-left image), TruncatedSVD (lower-right image))

2.4. Novelty

In the novelty part, we intend to discuss the feature dimensionality reduction strategies. Since we faced such problem that the amount of features in certain datasets maybe too large, we will try to resolve this problem in this section.

We found the data size of Merck Molecular Activity challenge data is much larger, the documents occupy over 100 mb spaces and the data is really sparse distributed. Whenever we need to load the data, we have to reload it which would reduce our productivity. So we use load_txt to load it and then save it to .npz files by using savez_compressed function from numpy which can decrease the data to around 2.5 mb which is one thirty of the original dataset size. The number of input attributes is 5877 for the first sub data set and 4306 for the second sub data set. We use four different models such as PCA [1] and strategies discussed later to reduce the feature dimensionality and compare the results individually. The specific results are shown in Figure 8.

In this case, we are going to use four different strategies to reduce the dimensionality of the raw data, which are:

1. Delete the columns with attributes that zeros occupied more than 85% of the values in that column.

2. By using sklearn.feature_selection.VarianceThreshold and set the threshold to 0.8 to split out attributes of low variance.

3. By using sklearn.decomposition.PCA, we set the n_components to 400 which can convert the original

matrix to a 400-columns matrix by linear transformation.

4. By using sklearn.decomposition.TruncatedSVD, we set $n\_components = 400$ as well. This data set has a slightly better performance compared with standard PCA against sparse matrix.

We only reduce the dimensionality of the raw data and keep other operations unchanged. Here we choose ACT2 as our input data and feed through all 7 kinds of regressors.

By observing the output histogram. We find that columns remove (strategy 1 and 2) has a better performance against the test set. PCA family has a strong prediction on training set. It seemed like this is due to overfitting of the training especially for NNR and RFR. So in this case, we determined that for this particular data set, VarianceThreshold performed more stable.

3. Conclusions

Aftering getting all the results of experiments, we start to think, which model has the best overall performance? which model should we use upon a new unseen dataset? and which model has the most considerable performance comparing to its training time?

We can reach the conclusion that, overall, in classification models, KNN, RFC and NNC perform the best on the training accuracy, while these classifiers are not stable compared with logistic regression. AdaBoost classifier tends to train in longer time and may not finish in 3 minutes in our experiments. In regression models, Gaussian Process and Random Forests are more likely to be overfitted. By contrast, linear least squares and neural network regressor stay stable on both training set and testing set. Based on our experiment, regressors are generally trained in longer than classifiers, and the results could vary a lot on dataset.

For image recognition problems, convolutional neural network are more interpretable and advantageous over decision tree classifier. However, for a particular network, we may fall into local minimum during the training. We solved this issue by reducing the learning rate.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010. 6

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 5

[3] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*, page 3–24, NLD, 2007. IOS Press. 1

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 1

[5] 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

[6] Fergus R Zeiler M.D. (2014) visualizing and understanding convolutional networks. 8689 LNCS:818–833, 2014. 6