Classifying Blocks of Page Layout of a Document

Sai Krishna Karthikeya Dulla Harshit Shrivastava

Abstract

In our project, we decided to do classification, for which we chose a dataset from University of California at Irvine machine learning repository. We took the dataset called page-blocks, which contains page layouts of a document created from segmentation process. After visualizing the data, we first ran Naive Bayes Classification algorithm to classify data. We noticed that the accuracy is not good. We then classified it again using Decision Tree algorithm. In this report, we discuss about the structure of the dataset, it's visualization, classification algorithms and contrast their outputs. The report ends with a brief section about the future work that is possible on this dataset.

Introduction

Classification is the task of assigning objects or observations into one or more categories. Interesting and useful applications of classification are in spam detection, categorization of cells as malignant or benign, classifying galaxies based on shapes, classifying documents based on their content, etc. The input data generally used for classification task if a collection of records in a data structure usually knows as the data matrix.

A row of a data matrix contains two elements. First one is the data vector where each value corresponds to a single feature of the data, and the second element is the class-label or the target value that we want to predict.

We are particularly interested in text classification. Examples of text classification include, but are not limited to:-

1. Assigning subject categories, topics or genres to documents
2. Spam detection
3. Authorship identification
4. Age/gender identification
5. Language identification
6. Sentiment analysis

In a general text classification, the input is a document d and a fixed set of classes C = {c1, c2, c3, c4, ... , cn} and the output is a class c $\in$ C. Possible machine learning rules that can be used to classify are hand coded rules or supervised machine learning. While the accuracy of hand-coded rules can be very high, they are expensive to build and maintain. Hence supervised machine learning is preferred, wherein a model is generated by training a set of hand-labelled

documents. This model is tested on a test set, which predicts the class of each document (observation).

In this project, we explore two supervised machine learning techniques, decision trees and naive bayes for their applications in text classification.

Problem statement

The problem consists in classifying all the blocks of the page layout of a document that has been detected by a segmentation process. This is an essential step in document analysis in order to separate text from graphic areas.

Dataset

Source and description

We chose a dataset called page-blocks from the University of Irvine's Machine Learning repository (https://archive.ics.uci.edu/ml/datasets/Page+Blocks+Classification). The dataset consists of a 5473 examples coming out of 54 distinct documents. Each observation in the dataset (a row in data matrix) concerns one block. All attributes in the dataset are numeric. The dataset is in a format readable by C4.5.

Attributes

There are 10 attributes in the dataset, viz.:-

- height: integer. | Height of the block.
- length: integer. | Length of the block.
- area: integer. | Area of the block (height * length);
- eccen: continuous. | Eccentricity of the block (length / height);
- p_black: continuous. | Percentage of black pixels within the block (blackpix / area);
- p_and: continuous. | Percentage of black pixels after the application of the Run Length Smoothing Algorithm (RLSA) (blackand / area);
- mean_tr: continuous. | Mean number of white-black transitions (blackpix / wb_trans);
- blackpix: integer. | Total number of black pixels in the original bitmap of the block.
- blackand: integer. | Total number of black pixels in the bitmap of the block after the RLSA.
- wb_trans: integer. | Number of white-black transitions in the original bitmap of the block.

Target variable (classes)

There are 5 target variables or classes:-

- Text (1)
- horizontal line (2)
- Picture (3)
- vertical line (4)
- graphic (5)

Class distribution

Class	Frequency	Percent	Valid Percent	Cum Percent
text	4913	89.8	89.8	89.8
horiz. line	329	6.0	6.0	95.8
graphic	28	.5	.5	96.3
vert. line	88	1.6	1.6	97.9
picture	115	2.1	2.1	100.0
	-------	-------	-------	
TOTAL	5473	100.0	100.0	

Summary statistics

Variable	**Mean**	**Std Dev**	**Minimum**	**Maximum**	**Correlation**
HEIGHT	10.47	18.96	1	804	.3510
LENGTH	89.57	114.72	1	553	.0045
AREA	1198.41	4849.38	7	143993	.2343
ECCEN	13.75	30.70	.007	537.00	.0992
P_BLACK	.37	.18	.052	1.00	.2130
P_AND	.79	.17	.062	1.00	-.1771
MEAN_TR	6.22	69.08	1.00	4955.00	.0723
BLACKPIX	365.93	1270.33	7	33017	.1656

BLACKAND	741.11	1881.50	7	46133	.1565
WB_TRANS	106.66	167.31	1	3212	.0337

Visualization

The data can be visualized as below using a pairs plot:-

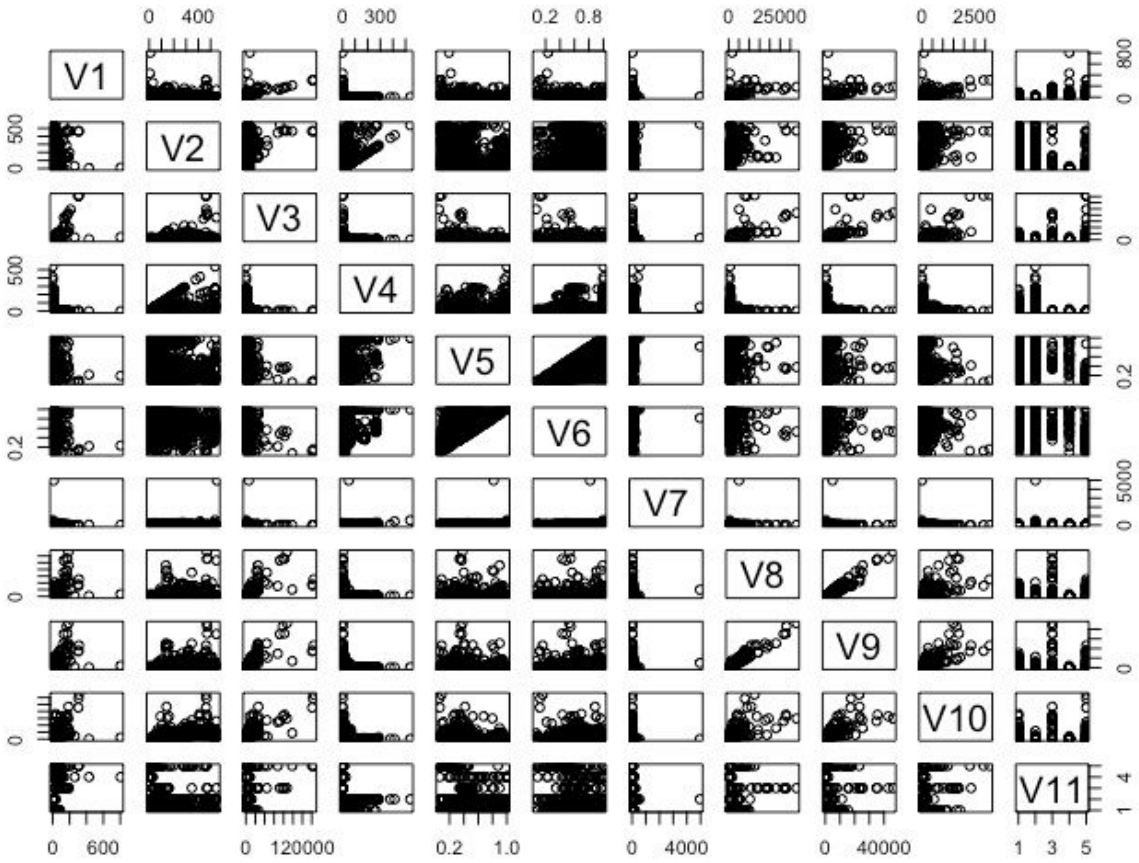


Figure 1. Pairs plot of the dataset

Correlated Data

From the pairs plot it is evident that columns “blackpix (V8)” and “blackand (V9)” are linearly correlated, and they have a correlation coefficient of 0.957. Including both the features would

be adding repetitive information which could affect the model. So, we planned to remove feature blackand.

Design / Methodology

Imbalanced Class Distribution

The class distribution is very skewed in this dataset. The class distribution section aboves gives the percentage of records belonging to each class in the dataset, where we can observe that class '1' has 89.8% of data records, class '2' has 6% of data records and the rest of the classes have very low percentage of occupancy in the data. There are many options to try to overcome this issue, but we chose the below three

1. Stratified Sampling
2. Resampling Data (Oversampling or Undersampling)
3. Trying different Classification Algorithms

Classification Techniques

We thought of implementing the Naive Bayes algorithm to classify this dataset, implementation of which will be explained briefly in later sections. The reason to choose naive bayes is to form a baseline to compare the learning performance, and also it does not need a lot of data to perform well.

As the class distribution is unbalanced, we cannot rely on classification accuracy as a parameter for performance estimation, because even classifying all the records as '1' would give 89.8% accuracy. So we considered to include confusion matrix also as the parameter for performance estimation, so that we can know how the classes with less occupancy are classified.

In the initial implementation of naive bayes algorithm:

(i) We took random sampling of data and divided them to training and test sets, the classification error for this algorithm is coming around 10% -14% and in which the algorithm has a very poor performance in classifying the low occupancy classes 3, 4 and 5 in the test set.

(ii) In the later attempt we tried to use stratified sampling, though these modification reduced the classification error to 8%, the confusion matrix implied that the algorithm improved in classifying the classes 1 and 2, but it still has a poor performance with rest of the classes.

(iii) Then we used oversampling of data to get enough examples of classes 3, 4, 5 to train, but this turned out to be increasing error rate.

The next attempt was to trying different classification techniques. We considered implementing the decision tree classifier as its splitting rule looks at the class variable used in creation of trees and can force all the classes to be addressed [5]. This gave us an accuracy around 95.26% on the test set and the classification of low occupancy classes is also good, which will be discussed in detail in next sections.

Implementation

The algorithms were implemented in R language.

Naive Bayes Classification

1. The data set is read from the UCI dataset as a matrix into R
2. The feature blackand is removed from the matrix as it is highly correlating with feature blackpix
3. For stratified sampling, divide the data into individual matrices for each class. Sample each class respective data and divide them into training and test sets with 4:1 ratio. This ensures that each class is picked up proportionally in training and test sets
4. Training sets of all classes were clubbed to form a unified training set. Repeat the same for test sets as well
5. For the Resampling, we opted to oversample the data, i.e. repeating the data samples of classes with low occupancy
6. Use the variable "case" present in data pre-processing section of R-code to check different data processing methods and classification model on that respective data
    - case = 1 , gives classification model on random sampling of original dataset
    - case = 2 , gives classification model on stratified sampling of original dataset
    - case = 3 , gives classification model on oversampled dataset
7. After the data preparation, the data is passed to classification_error_naive_bayes() function to get the predicted class vector, and the classification error. Both training and test data are passed to this method to observe how the model is working with test data
8. Confusion matrix is built for both training and test data to check the model performance while classifying the classes with low occupancy in data

Decision Tree Classification

1. The dataset is read from the UCI dataset as a matrix into R
2. It is divided into 2 parts for training and testing with 80% observations going into the training set and 20% observations going into the test set
3. Tree was constructed using rpart and plotted

4. It was tested to see if it is overfitting (complexity parameter, cp, was printed), but it did not overfit, hence no pruning was required. Data points for this are provided in the subsequent section
5. The model constructed from the rpart was testing against the test set
6. The accuracy was very good
7. Confusion matrix was plotted

Evaluation / Findings

Separate test and train sets were constructed from the original dataset for both the algorithms. The models generated from the Decision Tree algorithm and Naive Bayes Classification algorithm were used to classify the data in test set.

Naive Bayes Classification

We constructed the Naive Bayes Classifier model to run on both the test as well as the train datasets, as the error rate was 7.35%% and 6.85% for train and test on a random experiment.

The distribution of classes in the test and train datasets is:-

```
> #class distribution of data
> table(train.data[,10])

   1    2    3    4    5
3937  255   25   68   93
> table(test.data[,10])

  1   2   3   4   5
976  74   3  20  22
```

The error rate of the model on both the test and train datasets and the confusion matrix :-

```
> #error rates on training and test sets
> train.error
[1] 0.07354957
> test.error
[1] 0.06849315
>
> #confusion matrices on both training and test sets
> confusion_matrix_train = conf_mat(yhat.train, train.data[,10])
> confusion_matrix_test = conf_mat(yhat.test, test.data[,10])
> confusion_matrix_train
     [,1] [,2] [,3] [,4] [,5]
[1,] 3774   22    5   91   45
[2,]   69  162    0   21    3
[3,]    8    0   17    0    0
[4,]    1    1    1   64    1
[5,]   45    1    5    3   39
> confusion_matrix_test
     [,1] [,2] [,3] [,4] [,5]
[1,]  869   41    8   35   23
[2,]   63    3    0    7    1
[3,]    2    0    1    0    0
[4,]   16    1    0    2    1
[5,]   18    1    0    3    0
>
```

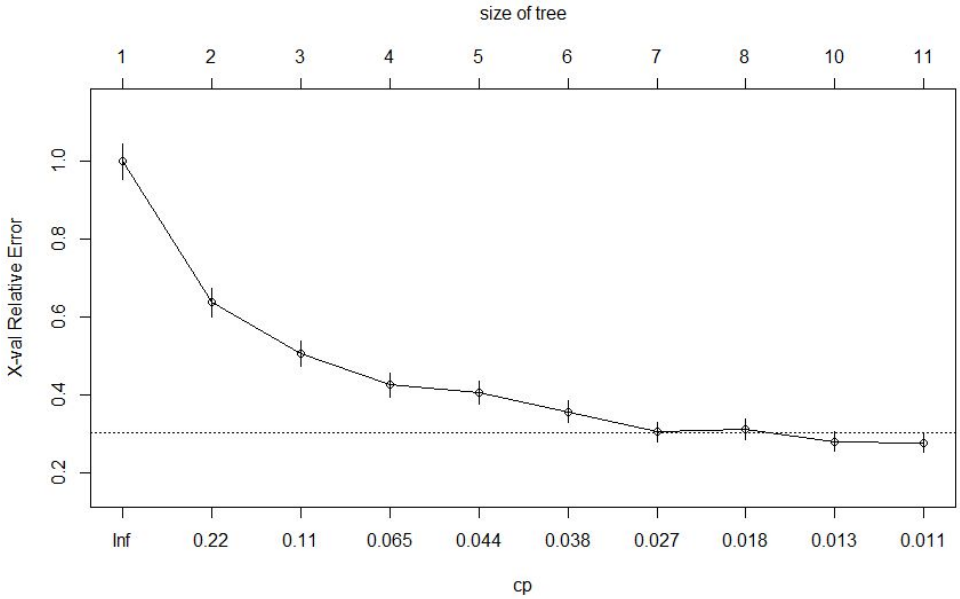
We can see that though the classification accuracy is 6.8% for test set, the confusion matrix shows that classification of classes 2, 3, 4, 5 is very poor.

Decision Tree

The complexity parameter printed for the decision looked like below

```
> printcp(model)

Classification tree:
rpart(formula = trainset$V11 ~ ., data = as.data.frame(trainset),
    method = "class", minbucket = 5)

Variables actually used in tree construction:
[1] V1 V2 V4 V5 V7

Root node error: 439/4378 = 0.10027

n= 4378

         CP nsplit rel error  xerror     xstd
1  0.362187      0   1.00000 1.00000 0.045271
2  0.132118      1   0.63781 0.63781 0.036878
3  0.084282      2   0.50569 0.50569 0.033068
4  0.050114      3   0.42141 0.42597 0.030477
5  0.038724      4   0.37130 0.40547 0.029767
6  0.036446      5   0.33257 0.35763 0.028026
7  0.020501      6   0.29613 0.30524 0.025962
8  0.015945      7   0.27563 0.31207 0.026242
9  0.011390      9   0.24374 0.28018 0.024906
10 0.010000     10   0.23235 0.27790 0.024807
> plotcp(model)
```

The complexity parameter was also plotted and it looked like this:-



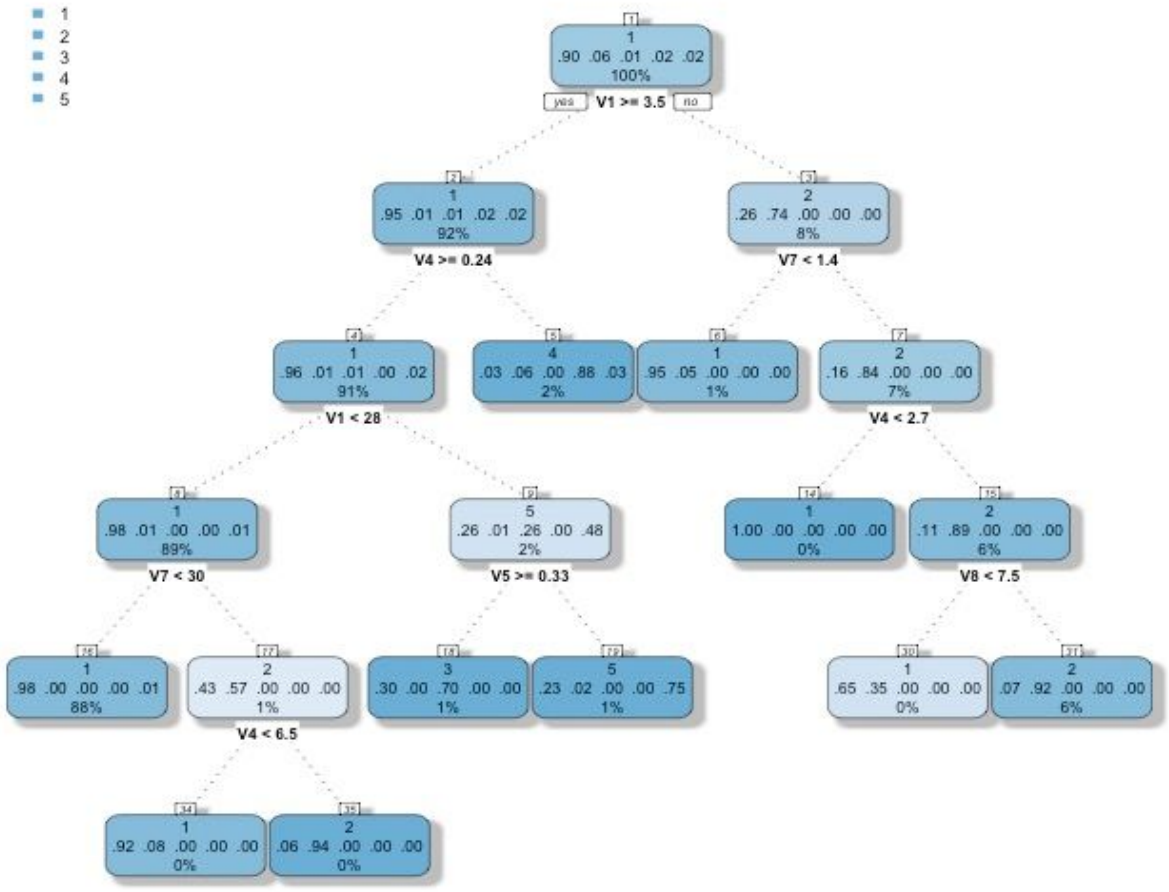
The above plot explains that how the relative error rate decreases with the size of the tree. From the above plot, it is also evident that the tree is not overfitting, as the relative error rate keeps decreasing with the size of the tree.

Further, the accuracy achieved on a random experiment (it will change every time as we sample the distribution before creating train and test datasets) was 95.26%.

The confusion matrix for this experiment looked like this:-

```
> class.pred

     1   2   3   4   5
1 960  10   8   3  15
2   9  49   0   0   0
3   0   0   1   0   0
4   1   2   0  19   0
5   4   0   0   0  14
>
```

From the confusion matrix it is evident that classification on 2, 3, 4, 5 classes has improved a lot when compared to naive-bayes algorithm.



Discussion

Both Naive Bayes and Decision tree have their own merits and demerits. In Naive Bayes, we need to train the classifier by hand, while in Decision Tree the classifier picks the best attributes by looking at the table. Hence, there will be situations where a very objective decision needs to be taken and Naive Bayes will perform worse than the decision tree, as in our case above. However, if the decision needs to be taken on some subjective situation where a lot of variable needs to be looked at, Naive Bayes will perform well and decision tree might perform poorly, eg. predicting the outcome of a poker game, as the decision tree might prune the branches that lead to some outcomes and obviously when the model is tested, observations leading to those classes will be misclassified.

In our case the Naive Bayes classifier tried to learn the model on the class that has greater distribution in the data, which resulted in a poor classification for the remaining classes available. Whereas Decision Tree took all the classes into consideration to generate the splitting rules, which eventually lead the model to address all the classes in a split. This not only resulted

in a good accuracy of the classification but also a better performance on the classes that have least occupancy in the dataset.

Conclusion

We concluded from our experiments that for this dataset, Decision Tree algorithm gave much better classification accuracy than the Naive Bayes Classification algorithm.

Future Work

In future we can implement Boosting algorithms on the dataset to see if it increases accuracy of classification. We can also implement a multi-class SVM on this dataset and evaluate it against other algorithms.

We can also apply techniques to resolve the issue of imbalance in the class distributions and then run all the algorithms again to see how much it affects the classification.

References

Following online and offline resources were referenced:-

1. B565 (Data Mining) class notes
2. Bishop, Christopher (2007): Pattern Recognition and Machine Learning, Springer
3. https://en.wikipedia.org/wiki/Document_classification
4. Pang-Ning Tan, Michael Steinback, Vipin Kumar (2007): Introduction to Data Mining, Pearson
5. http://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
6. R. Longadge, S. S. Dongre, and L. Malik, "Class Imbalance Problem in Data Mining: Review," International Journal 0/ Computer Science and Network (IJCSN), vol. 2, 2013.
7. P. Foster, “Machine learning from imbalanced data sets 101.” Proceedings of the AAAI 2000 workshop on imbalanced data sets, 2000, pp. 1–3