# HW2 Part 2

Hsin-Ping Hsu (G01167652)

1. **Breast Cancer Data:**

* Data preprocessing:

1. Fill missing value: use SimpleImputer library to fill the missing value by the “most frequent” strategy
2. Convert categorical variable into dummy (indicator) variables via panada.get\_dummies(). The new dataset is shown as below.

a. Decision Stump:

**Approach:**

1. Split data into training and testing datasets and conduct the cross validation (cv = 10) later.
2. Utilize sklearn.tree.DecisionTreeClassifier to build the decision tree. To create the decision stump, set the parameter: max\_depth = 1.
3. Use cross validation to obtain f1 score for each fold. And calculate the overall f1 scores.
4. Visualize the decision stump through export\_graphviz.

**Result:**

The root split the data into two leaf nodes by asking whether the value of deg-malig\_3 (X[31]) is greater or equal than 0.5 (Fig. 1). In other words, the root is asking whether the value of deg-malig equals to 3; therefore, if the value of deg-malig is equal to 1 or 2 in the original dataset, the answer is true; otherwise deg-malig = 3 (deg-malig\_3 = 1) and the answer is false.

|  |  |  |
| --- | --- | --- |
| Confusion Matrix | | |
|  | class 0 | class 1 |
| class 0 | 124 | 28 |
| class 1 | 27 | 35 |
| f1\_score | 0.81848185 | 0.56 |

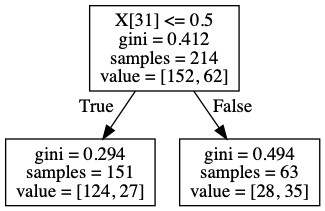
****

Fig. 1 Fig. 2

* What is causing most of the errors in the Decision Stump?

Decision stump is a biased model, it only classifies by one attribute; however, as we can see from the result when the deg-malig not equals to 3, the numbers of class 0 and class 1 are not significantly different ; therefore, if the model stop classifying by using other attributes, class 0 will be misclassified. On the other hand, the left child contains 43% of examples of class 1; therefore, the misclassification of class 1 is very high and the f1 score of class 1 is low.

b. Decision Tree Unpruned

**Approach:**

1. Split data into training and testing datasets and conduct the cross validation (cv = 10) later.
2. Utilize sklearn.tree.DecisionTreeClassifier to build the decision tree. To create the unpruned decision tree, I used the default parameters.
3. Use cross validation to obtain f1 score for each fold. And calculate the overall f1 scores.
4. Visualize the decision stump through export\_graphviz.

**Result:**

Fig.3 shows the unpruned decision tree, the depth of this tree is 15 and it split the nodes until there’s no more attributes or samples.

|  |  |  |
| --- | --- | --- |
| Confusion Matrix | | |
|  | class 0 | class 1 |
| class 0 | 36 | 4 |
| class 1 | 10 | 7 |
| f1\_score | 0.8372093 | 0.5 |

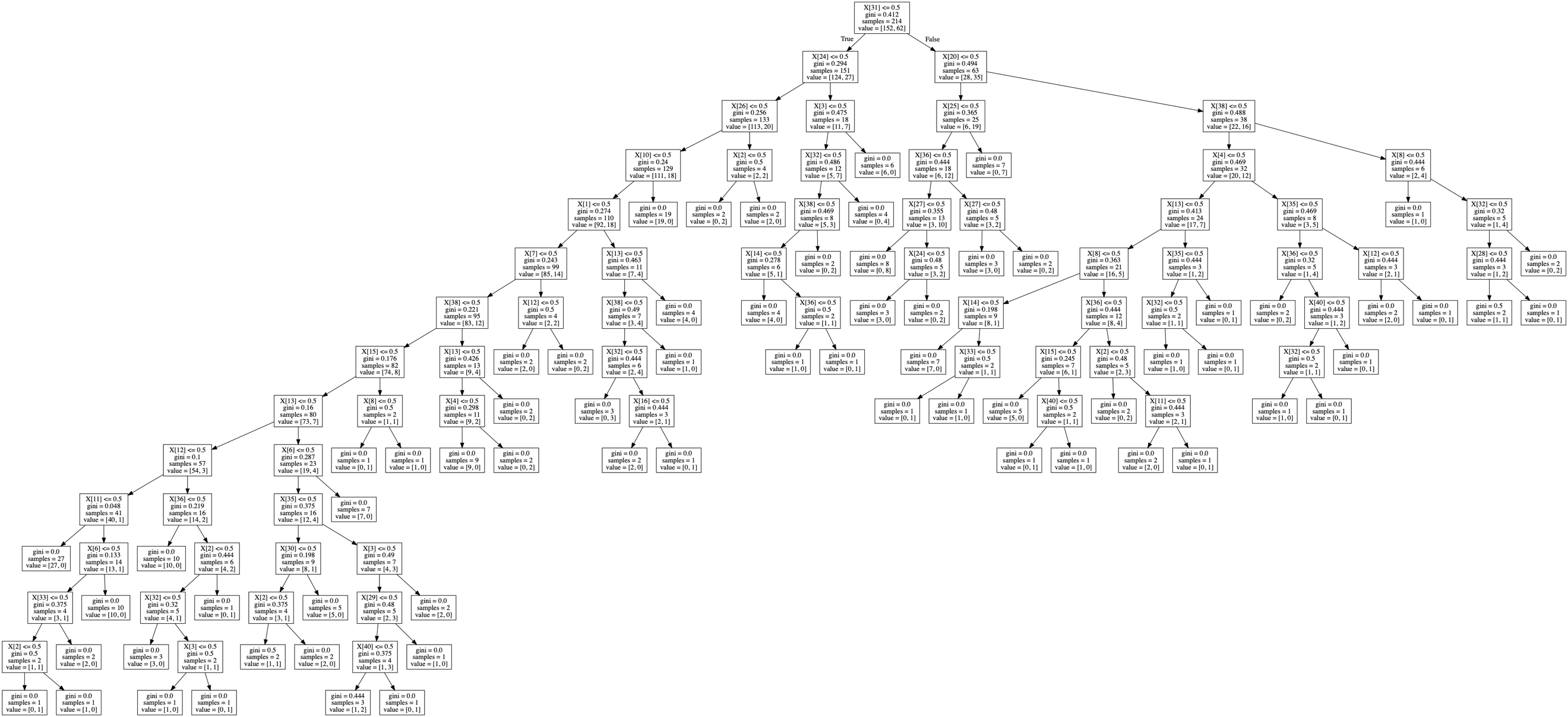
****

Fig. 3 Fig. 4

* What is causing most of the errors in the Unpruned Decision Tree?

The complexity of the unpruned decision tree (Fig. 3) is relatively high; therefore, this is a model with high variance. Hence, if the training dataset is different, the decision tree might be different, which might cause the errors in the model.

c. DT Pruned

**Approach:**

1. Split data into training and testing datasets and conduct the cross validation (cv = 10) later.
2. Utilize sklearn.tree.DecisionTreeClassifier to build the decision tree
3. Use randomized search cv to find the best parameters for pruned decision tree.
4. Use cross validation to obtain f1 score for each fold. And calculate the overall f1 scores.
5. Visualize the decision stump through export\_graphviz.

**Result:**

Fig.3 shows the pruned decision tree that is the same as the decision stump. The depth of the pruned decision tree is 1, and the root is the deg-malig\_3 attribute, which also means that rules of the pruned decision tree are the same as the decision stump.

|  |  |  |
| --- | --- | --- |
| Confusion Matrix | | |
|  | class 0 | class 1 |
| class 0 | 124 | 28 |
| class 1 | 27 | 35 |
| f1\_score | 0.81848185 | 0.56 |

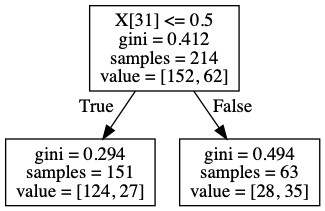
****

Fig. 5 Fig. 6

* What happened when you allowed the tree to get pruned?

As shown in the result, the pruned decision tree is the same as the decision dump. This might because of the small number of attributes and examples in the dataset.

d. K-Nearest neighbors

**Approach:**

1. Split data into training and testing datasets and conduct the cross validation (cv = 10) later
2. Utilize sklearn.neighbors. KNeighborsClassifier to build the K-Nearest neighbors model.
3. Use cross validation to obtain f1 score for each fold. And calculate the overall f1 scores

**Result:**

The confusion matrix (Fig. 7) shows that the K-Nearest neighbors has relatively low fi scores of class 1. This probably caused by the imbalanced data. Since the number of class 1 is much lesser than the number of class 0; therefore, while performing the K-Nearest neighbors, the probability of the neighbors being class 0 is relatively higher than the probability of the neighbors being 1.

|  |  |  |
| --- | --- | --- |
| Confusion Matrix | | |
|  | class 0 | class 1 |
| class 0 | 146 | 6 |
| class 1 | 50 | 12 |
| f1\_score | 0.83908046 | 0.3 |

Fig.7

1. **Churning data:**

For this problem, use the churning data. Observe the “defector” attribute (last one): what do you notice?

The number of class 0 is 8704 and the number of class 1 is 984. This dataset is imbalanced.

Before starting to build the models, I split the original data set into training and testing (10%).

1. Using a decision tree implementation and cross-validation produce a classification model for this data. What is noticeable in the data? Is this model good?

The confusion matrix of this model is shown as Fig. 8. The matrix shows that the f1 score of class 0 is relatively higher than class because of the imbalanced data set. The model can classify class 0 correctly mainly because the probability of class 0 is much higher than class 1. Therefore, this model is not good for imbalanced data.

|  |  |  |
| --- | --- | --- |
| Confusion Matrix | | |
|  | class 0 | class 1 |
| class 0 | 873 | 9 |
| class 1 | 72 | 15 |
| f1\_score | 0.95566502 | 0.27027027 |

Fig. 8.

1. Using undersampling, oversampling, and Smote (separately) balance the data and reapply the decision tree software (pruned) to the data. What do you notice? Report your findings.

* Under sampling:

The number of class 0 and class 1 both become 886 after applying imblearn.under\_sampling .RandomUnderSampler library to balance the data. Fig. 9 shows the result of classification by applying pruned decision tree, I observed that the f1 score of class 1 is increased after under sampling. However, the f1 score of class 0 decreased after under sampling. Since the majority of class will be under sampled, one of the possible reasons that leads to this result might be the reduction samples of class 0 and the increment of class 1 samples provides a better prediction of class 1.

|  |  |  |
| --- | --- | --- |
| Confusion Matrix | | |
|  | class 0 | class 1 |
| class 0 | 657 | 213 |
| class 1 | 3 | 95 |
| f1\_score | 0.85882353 | 0.4679803 |

Fig. 9

* Over sampling:

The number of class 0 and class 1 both become 7834 after applying imblearn.over\_sampling .RandomOverSampler library to balance the data. Fig. 10 shows the result of classification by applying pruned decision tree, I observed that the f1 score of class 1 is higher than the one with imbalanced data. In addition, compare it with the model that built with under sampler, the misclassification of class 0 is lower (673 > 657); this might because the dataset still remains large enough amount of class 0 to train.

|  |  |  |
| --- | --- | --- |
| Confusion Matrix | | |
|  | class 0 | class 1 |
| class 0 | 673 | 197 |
| class 1 | 3 | 95 |
| f1\_score | 0.85882353 | 0.48717949 |

Fig. 10

* Combination – SMOTETomek sampling:

The SMOTETomek combine over-sampling and under-sampling using SMOTE and Tomek link.

The number of class 0 and class 1 both become 7759 after applying the imblearn.combine.SMOTETomek library to balance the data. Fig. 11 shows the result of classification by applying pruned decision tree. As we can see from the result, the misclassification of class 0 is relatively decreased compare with other models, especially with the model that built with under-sampling. The correctness of class 1 isn’t improved a lot, but the f1 score of class 1 is relatively higher because of the lower misclassification of class 0.

|  |  |  |
| --- | --- | --- |
| Confusion Matrix | | |
|  | class 0 | class 1 |
| class 0 | 722 | 148 |
| class 1 | 6 | 92 |
| f1\_score | 0.90362954 | 0.5443787 |

Fig. 11

* Conclusion:

The original dataset is imbalanced; thus, the f1 score of class 1 would be low without utilizing sampling techniques. Furthermore, the technique of SMOTETomek sampling might be the better sampling technique for this dataset because the f1 score of class 1 is relatively higher, and the f1 score of class 0 is also above 90%.

1. **train\_mnist\_clean\_bestdataset:**

I split the dataset into training and testing (20%) before following performances. And use 20% of the remaining dataset for cross validation.

1. Assess the “usefulness” of the features using the attribute evaluator Information Gain. How does this “attribute evaluator” work? Remove all the attributes that have 0

I converted the provided dataset into the original dataset (containing usefulness features), then used the info\_gain.py library to calculate the information gains for each attribute. The result shows that those attributes whose indices are greater than 524, the information gain is 0; therefore, I removed the attributes whose indices are greater than 524. The size of the training set is 1600 \* 524.

Is it safe to perform this operation? Save the reduced dataset.

In my opinion it is safe to perform this operation for this dataset because the information gain of each attribute is the main measurement among classes and attributes, which means that it measures how much information does an attribute tell us about the class. Therefore, if the information gain of an attribute is 0, then this attribute probably does not impact the classification much.

On the other hand, I observed that those attributes get removed whose pixels contain value of 0, the weakest intensities, which may because those pixels on images is blank or do not contain any handwriting. Therefore, the blank areas have nothing to do with the handwriting classification.

1. Perform PCA on the dataset and retain 90% of the variance. Save the reduced dataset.

After performing the PCA technique, the size of the training set is reduced to 1600 \* 126.

**Result:**

I performed the gird search by using the tool called grid.py from libsvm to obtain the best parameters for building the classifiers. The best parameters for c and g are 8 and 0. 001953125.

Fig.12 shows the results of classifiers using different data sets. The accuracies between these smodels do not have significant different; thus, reducing dimension and removing low information gain attributes are good for classification.

|  |  |  |
| --- | --- | --- |
| **Data set (classifier)** | **Training size** | **accuracy** |
| Original | 1600 \* 783 (excluding the class attribute) | 89.5 |
| Information gain | 1600 \* 524 | 90.25 |
| PCA | 1600 \* 126 | 90.0 |

Fig. 12

**References:**

1. <https://scikit-learn.org/stable/modules/generated/sklearn.tree.export_graphviz.html>
2. <https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html#imblearn-over-sampling-smote>
3. <https://github.com/cjlin1/libsvm/tree/master/tools>
4. <https://github.com/cjlin1/libsvm/tree/master/python>
5. <https://pypi.org/project/info-gain/>
6. <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html>
7. <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html>
8. <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html>
9. <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.get_params>