Machine Learning Assignment 1

Ting Chun Yeh 葉庭均

NM6124012

AI Robotics, National Cheng Kung University, Tainan, Taiwan

Abstract — **This report is all about Machine Learning Assignment 1. In the report I will introduce all the processes of how I deal with the data and how I build the models. After that, I will be doing some feature engineering work and use SHAP to see the feature importance. I will also create some new features. Finally, I will be doing K-Fold Cross Validation and compare the performance between the original results and the results of K-Fold Cross Validation.**

Keywords — Machine Learning, Assignment 1, Linear Classifier, K-NN Classifier, Naïve Decision Tree Classifier, Decision Tree with Pruning, K-Fold Cross Validation

Code — https://github.com/FierceTiffany/MachineLearning_Assignment1.git

I. Introduction

Different models need different kinds of data processing methods. Therefore, first of all, I will start with data preprocessing. Later on, I will introduce how I build the models. After that, finishing training and having predictions of the data, I will be doing some analysis, which is about feature engineering and feature importance. Last but certainly not least, there will be some k-fold cross validation work too at the very end. After finishing all the work, I will put all the performance results together and have a clear look to see which has the best performance and wins the comparison.
This report is all about the whole working process and the results of different kinds of models using different kinds of strategy. I will be talking about all the details later down in the report.

II. Data Preprocessing

First of all after loading the data is to check whether there is any NA in the dataset. Luckily, there isn't any NA value in the dataset,

A. For Linear Classifier, due to some of the features being categorical, i.e. "Yes" and "No" or "C1", "C2", …, "C22", I have to do some transformations to the data. For those columns that is named is_something, I wrote a function just to change "No" to 0 and "Yes" to 1, and for the other two which are also binary ("rear_brakes_type" and

"transmission_type") I just changed "Drum" and "Manual" to 0 and "Disc" and "Automatic" to 1.
As for those which have multiple categories, I use the function `pd.get_dummies()` to do one hot encoding for those columns.
After complete the transformation of those columns, I concat them back and download as csv "train_new.csv" and "test_new.csv"

B. Due to the huge amount of computation, causing not being able to finish the code in a short time. Therefore, I randomly sampled and cut down the data into only 1500 pieces of data, which contains 1000 pieces of data with label: "is_claim" == 0 and 500 pieces of data with label: "is_claim" == 1.

III. Models

According to the assignment question, we are assigned to build the following models and to compare the performances between them.

A. Linear Classifier
The linear classifier is built based on the concept of y = W*X + b. Given an initial W and b, then do the forward backward propagation. Train num_of_iteration times in order to update the W and b to make the model as precise as possible.

B. K-NN Classifier
K-NN, it counts the point x_i's distance with all the other x in the dataset first. Then, it counts in the k closest neighbor, which the most common label is. It uses voting to make the final decision of the classification.
There are three different versions of K-NN Classifiers. They are different because of the way they calculate distance. One uses the most common "Euclidean Distance", the other two use "Manhattan Distance" and "Chebyshev Distance". Besides the way of calculating the distance, there is nothing else different between them.
In the code, I only set k = 3.

C. Naïve Decision Tree Classifier
First of all, count gini impurity. Then, use entropy in place of gini in order to get information gain. By the information gain value we calculated, we can find the best place or I should say best node to split the data. After the first split, we will keep doing the same thing recursively until we reach the leaf of all the branches.

D. Decision Tree with Pruning
All the things done at first, meaning building the tree, is the same as the Naïve Decision Tree Classifier. After finishing building the tree, we are going to start pruning, which means we are going to cut some unnecessary branches

and keep the ones that really help with the performance. In the code we will see whether pruning helps improve the accuracy, or it's actually no difference between.

IV. Feature Engineering

A. Algorithm

1. Linear Classifier

```
Top 5 features:
age_of_policyholder
turning_radius
length
width
height
```

2. Decision Tree

```
Top 5 features:
policy_tenure
age_of_policyholder
age_of_car
population_density
height
```

B. SHAP

1. Linear Classifier

```
Accuracy = 60.0%
Important Features:
width
population_density
length
displacement
height
```

2. Decision Tree

```
Accuracy = 59.0%
Important Features:
policy_tenure
age_of_car
age_of_policyholder
cylinder
is_brake_assist
```

C. Design New Features

From the result of linear classifier, decision tree, and SHAP, we can tell that "age_of_policyholder" and "height" are two very important features. Therefore, we can use these two to create a new feature. First, I calculate the mean and the variance of "height" in order to normalize the feature. Secondly, I multiply all the values by 10 to increase the value (otherwise it will be a little bit too small compared to age_of_policyholder, and doing these steps are to make the power of the feature height comparable to age_of_policyholder). Finally, I subtract "age_of_policyholder" from "height", and finish creating the new feature.

The results are as follow:

1. Linear Classifier

```
Test Accuracy =  69.0
%
```

2. Decision Tree

```
Test Accuracy =
62.67 %
```

I would say that adding this feature does improve the performance compared to the performance when doing SHAP.

V. K-Fold Cross Validation

Setting K = 3, 5, and 10 as requested, and re-run the Problem 1 model code to see whether the performance is better than the original setting which test size is 0.2 and the data is split randomly.

Here, I just simply import the module k-fold to do the test. The model code is the same as Problem 1, is just that each time, the validation data will be different according to the k-fold split.

VI. *k* Classifiers in K-Fold Cross-Validation.

By Voting

```
Test Accuracy =  59.07 %
```

By Merge

```
Test Accuracy =  63.60 %
```

The complexity of *k* classifiers in k-fold cross-validation is generally higher than a single classifier, due to training overhead, computational cost, memory usage, and algorithm selection.

Training Overhead: In k-fold cross-validation, you train k different classifiers, each on a different subset of the data. This implies k times the training overhead compared to training a single classifier on the entire dataset.
Computational Cost: Running k-fold cross-validation involves training and evaluating the classifier k times (for each fold). This increases the computational cost compared to training and evaluating a single classifier.
Memory Usage: With k-fold cross-validation, you need to keep k different models in memory during the cross-validation process.
Algorithm Selection: If you're using different types of classifiers in each fold, the overall complexity depends on the complexity of each classifier. If the classifiers are complex or computationally expensive, the overall complexity increases.

As for the performance, it depends on the classifier. According to the results, merging/aggregating the predicted results from *k* classifiers in k- fold cross-validation is better than most of the models. This is true. However, there's still one model (or I should say two) which won the performance.

VII. Model Performance

Normal Situation

Model	Accuracy
Linear Classifier	61.67%
K-NN Classifier Euclidean Distance	61.67%
K-NN Classifier Manhattan Distance	61.33%
K-NN Classifier Chebyshev Distance	58.33%
Naïve Decision Tree Classifier	67.00%
Decision Tree with Pruning	**69.00%**

K-Fold Cross-Validation

A. Linear Classifier

K =3	K = 5	K = 10
65.00%	63.67%	66.00%
70.20%	67.33%	60.67%
64.80%	70.34%	66.00%
x	70.67%	68.67%
x	61.67%	72.67%

x	x	67.33%
x	x	73.33%
x	x	67.33%
x	x	56.01%
x	x	66.00%
66.67%	66.74%	66.40%

* Last Row: Average Accuracy

B. K-NN Classifier

Euclidean Distance		
K =3	K = 5	K = 10
57.71%	56.94%	59.38%
59.02%	57.06%	57.38%
57.58%	58.81%	57.63%
x	58.13%	56.38%
x	56.44%	60.88%
x	x	56.88%
x	x	58.13%
x	x	60.25%
x	x	55.63%
x	x	57.88%
58.10%	57.48%	58.04%

* Last Row: Average Accuracy

Manhattan Distance		
K =3	K = 5	K = 10
57.44%	56.94%	58.75%
59.51%	56.75%	57.00%
57.58%	59.75%	56.63%
x	58.06%	57.63%
x	55.94%	62.63%
x	x	57.50%
x	x	56.63%
x	x	59.00%
x	x	55.00%
x	x	57.63%
58.18%	57.49%	57.84%

* Last Row: Average Accuracy

Chebyshev Distance		
K =3	K = 5	K = 10
57.14%	57.06%	59.00%
58.53%	56.69%	55.75%
58.55%	59.75%	57.75%
x	56.94%	55.50%
x	57.63%	60.25%
x	x	58.25%
x	x	55.75%
x	x	59.88%
x	x	57.13%
x	x	59.00%
58.07%	57.61%	57.83%

* Last Row: Average Accuracy

C. Naïve Decision Tree Classifier

K =3	K = 5	K = 10
62.00%	69.00%	68.00%
61.00%	70.00%	66.00%
59.00%	67.00%	73.00%
x	70.00%	59.00%

x	67.00%	70.00%
x	x	69.00%
x	x	68.00%
x	x	67.00%
x	x	68.00%
x	x	66.00%
60.67%	68.60%	67.40%

* Last Row: Average Accuracy

D. Decision Tree with Pruning

K =3	K = 5	K = 10
65.00%	86.00%	86.00%
69.00%	89.00%	89.00%
64.00%	88.00%	90.00%
x	91.00%	86.00%
x	81.00%	83.00%
x	x	89.00%
x	x	91.00%
x	x	89.00%
x	x	85.00%
x	x	87.00%
66.00%	87.00%	87.50%

* Last Row: Average Accuracy

Model	Problem 1	K = 5
Linear Classifier	61.67%	**66.74%**
K-NN Classifier Euclidean Distance	**61.67%**	57.48%
K-NN Classifier Manhattan Distance	**61.33%**	57.49%
K-NN Classifier Chebyshev Distance	**58.33%**	57.61%
Naïve Decision Tree Classifier	67.00%	**68.60%**
Decision Tree with Pruning	69.00%	**87.00%**

Most of the time if the models are classifiers, we try to tell each model's performance by comparing their evaluation, inference result, which is normally accuracy. In this way, we will kind of know whether the performance of one model is really better than another one.

Comparing the results in 5-fold cross-validation and the results of problem 1, except k-nn classifiers, all the average accuracy of the classifiers in 5-fold cross-validation is higher than the accuracy of problem 1. I think it is because of the condition of the data and the ability of the model. First, this data is quite an imbalanced dataset, even though I didn't put all the data into the model. Nonetheless, when the dataset is imbalanced, the way we split the data will be influential to the result of accuracy. This is because if we didn't try to let the model learn all kind of conditions of the data, which is we need to try to generalize the model, else the model will only know a side of the data in which if the test data is of a totally different condition, then the model cannot make perfect prediction.
As for the model side, I didn't mean that the model's ability is bad or so. What I am trying to say is, not every kind of dataset

fits every kind of model. Sometimes, a certain model is just not suitable for that specific dataset. Therefore, I wouldn't say that model A is "REALLY" better under a certain condition than model B. However, I can say model A is better than model B under this kind of condition.
As for which model is "REALLY" better, for this dataset I will say that 5-fold cross-validation models generally perform better. More specifically, the decision tree classifier with pruning using 5-fold cross-validation has "REALLY" the best performance of all, for its accuracy is really high for others to compare with.

VIII. Conclusion

Overall, no matter in problem 1 or in k-fold cross-validation, the decision tree classifier with pruning has the best performance of all. The biggest problem of the data I don't think is the imbalancing of the data, but the features of the data we use. Take the decision tree classifier and the decision tree classifier with pruning as example, it kind of prove that sometimes we do not need to put all the features of the data into consideration to get the best performance, what we need to do is select the important features and use them wisely in the right model.

References

1. Naïve Decision Tree Classifier / Decision Tree with Pruning: https://github.com/anshul1004/DecisionTree
2. SHAP: https://shap.readthedocs.io/en/latest/ https://shap.readthedocs.io/en/latest/example_notebooks/tabular_examples/model_agnostic/Iris%20classification%20with%20scikit-learn.html
3. https://stackoverflow.com/questions/66079043/split-dataset-without-using-scikit-learn-train-test-split