ISOM 3360 Assignment 2

Joshua Chang, Fung Ho Yin Matthew, Chan Shun Hang
{jchangad, hymfung, shchanax}@connect.ust.hk

1. Exploring and preprocessing the dataset

Before we build any models, we first need to explore and reformat the dataset. The dataframe consisted of 7032 Telco customers, and had 19 columns to each customer, with the last column (churn) being our target variable. This means the dataset had shape (7032, 19).

We checked and confirmed that the dataset did not contain any `na` values, and used `np.unique()` to find out the possible values to each column. After exploring the dataset, we are able to group the columns of the dataframe into three main categories:

- General information

  These columns contained basic information that every customer had, for example their gender, whether they have a partner, their monthly charges and their contract length.

- Phone service related information

  These columns had information on whether the customers were subscribed to their phone service, and whether they had multiple lines with Telco.

- Internet service related information

  These columns had information on whether the customers were subscribed to their internet service, had online security, online backup and so on.

Since most columns contained only a few possible values, we converted the dataframe into a NumPy array for easier modelling later on. The detailed conversion rules can be found in the python notebook.

2. Building decision trees *(Task 1, Question 1)*

We considered 3 different methods of building decision trees in total. Each model was trained and validated using 10-fold cross validation on 70% of the whole dataset. To ensure our trees are reproducible for fair comparison between different trees, we fixed "`random_state`" as "2211", Evaluation of the trees were done using **accuracy**, **precision**, **recall**, and **AUC** as metrics, where each metric was taken as the simple average over all 10 folds. The summary of our models can be found in table 1.

Tree	Accuracy	Precision	Recall	AUC
Basic (§2.1)	0.738	0.505	0.517	0.668
Adj. params (§2.2)	0.794	0.624	0.566	0.831
Ensemble (§2.3)	0.0	0.0	0.0	0.0

Table 1. Comparison between trees

2.1. The basic tree

The first and most basic model we built was a basic decision tree with all the features used for training, and all the default parameters `sklearn` provides. This means using "entropy" to decide which feature to split, splitting with the best feature, and allowing the tree to grow until all leaves were pure or contained less than 2 samples. During each split, the model considers *all* attributes.

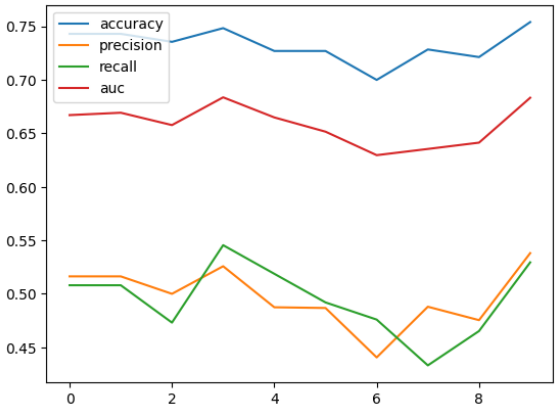


Figure 1. different metrics across the folds

Figure 1 shows the metrics across different folds during cross validation, and figure 2 shows the AUC curve. The metrics seems normal, however the AUC curve is too good to be true. Indeed, upon inspection we found that

```
model.predict_proba(X)
>>> array([[0., 1.], [0., 1.],
..., [1., 0.]])
```

which indicates that the model predicts the probabilities for

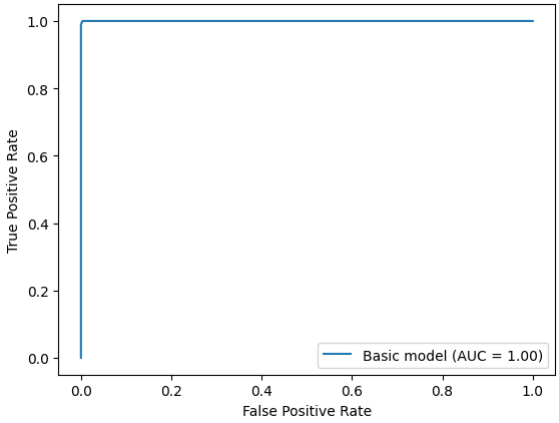


Figure 2. AUC curve for the basic tree

Criteria	Accuracy	Precision	Recall	AUC
Accuracy	0.799	0.646	0.529	0.831
Precision	0.794	0.624	0.566	0.831
Recall	0.796	0.673	0.451	0.834
F1	0.794	0.624	0.566	0.831

Table 2. The best trees based on different criterion

Tree	Accuracy	Precision	Recall	AUC
Phone & Internet	0.753	0.623	0.610	0.716
Phone only	0.921	0.333	0.190	0.582
Internet only	0.766	0.520	0.394	0.639

Table 3. Comparison between ensemble trees

each sample with 100% certainty. In fact, when comparing it with the ground truth, out of 4,992 samples it correctly classifies 4,889 of them. This means adjusting the threshold won't affect the prediction results, which explains the absurd AUC curve.

The model also had a depth of 34 with 1,467 leaves. The reason the model overfits the dataset is because by default, `sklearn` will continue splitting until each leaf is pure or contains 2 or less samples, so these are the first parameters we are going to change in our next iteration.

2.2. A tree with adjusted parameters

The next model we will build aims to solve the issue above. Specifically, we will adjust the following parameters, and find the best model by GridSearchCV using 10 folds.

- `max_depth`

  Search range is `np.arange(5, 35, 5)`.

  We chose this range because the overfitted model in section 2.1 had a depth of 34, so setting `max_depth` to 30 should be sufficient.

- `min_samples_split`

  Search range is `np.arange(10, 110, 10)`.

  We chose this range because based on empirical testing. Even when we set lower / higher values, the best estimator found by GridSearchCV still mostly took on values in this range.

- `min_samples_leaf`

  Search range is `np.arange(10, 110, 10)`.

  We chose this range based on the same empirical testing reason.

After deciding how to build the tree, the next step is to decide which metric to use for GridSearchCV to decide the best estimator. We tried using different criterion, and the results are summarized in table 2 and in figure 3. As usual, each model was trained with `random_state = 2211`, and evaluated using 10 folds CV.

Our empirical testing shows that the regardless of the criterion chosen, the model's performance is already a significant improvement compared to the basic model in section 2.1.

Out of the 4 criterion chosen, figure 3 shows that the models generally performed equally, so which model to choose out of the four models will depend on the costs of misclassifying the customers, to be handled in the later sections. For now, we will pick the fourth model (F1 score as criteria) as the benchmark for this section.

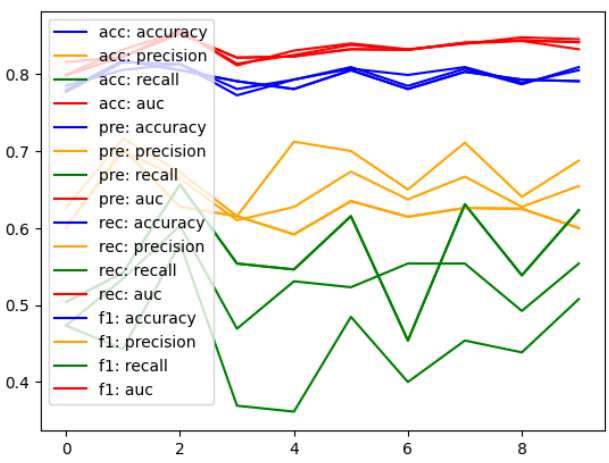
2.3. Ensemble trees

Recall in section 1, we discussed that the customers could be broadly grouped into two categories, those with phone service and those with internet service.

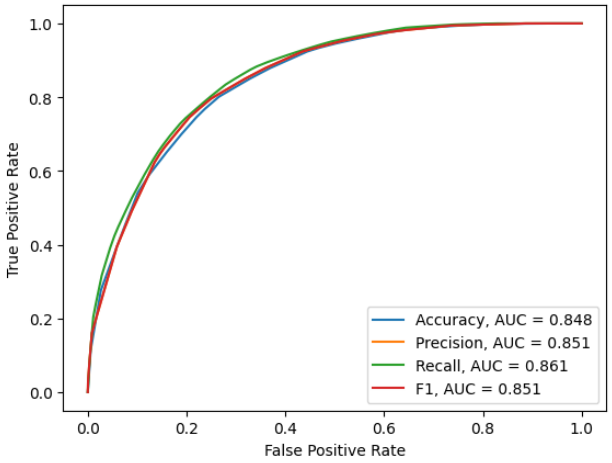
It follows from intuition that these two types of customers would have different reasons for continuing or discontinuing their contracts, so one idea to build the decision tree would be to first split the customers into three categories: (1) those subscribed to phone service only, (2) those subscribed to internet service only, and (3) those subscribed to both services.

Unfortunately, scikit learn does not have an easy way to specify the first split, so we have to do it manually. We built three models, corresponding to the three types of customers mentioned in the paragraph above. The results are summarized in table 3.

Note that since the sample size for internet only users is 456, when we did 10 fold CV we encountered a division by 0 error due to the small fold size. To make the comparison



(a) the performance of the four models across folds



(b) AUC curve for the four models

Figure 3. Performance of the four models in §2.2

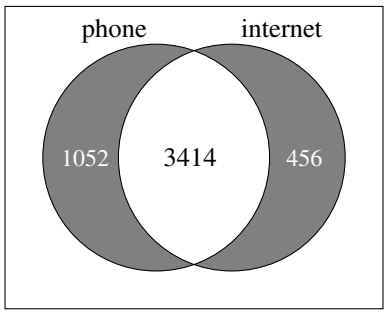


Figure 4. subscription service provides a natural split

fair, we did not use CV to evaluate the models. Instead, we split off another 30% from the training set, and used that to validate our models.