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 7,032 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. 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 **??**.

Tree	Accuracy	Precision	Recall	AUC
Basic (§**??**)	0.738	0.505	0.517	0.668
Adj. params (§**??**)	0.794	0.624	0.566	0.831
Ensemble (§**??**)	0.790	0.551	0.500	0.680

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.

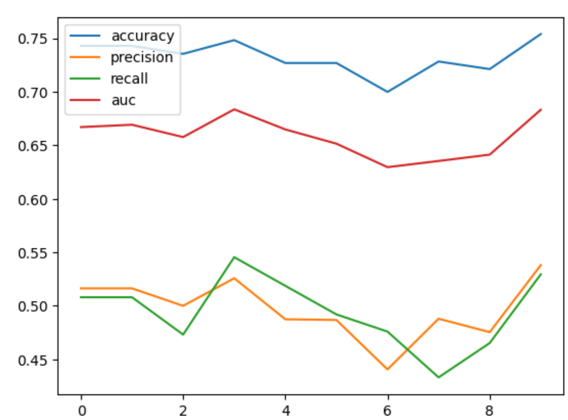


Figure 1. different metrics across the folds

Figure **??** shows the metrics across different folds during cross validation, and figure **??** shows the AUC curve. The metrics seem 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

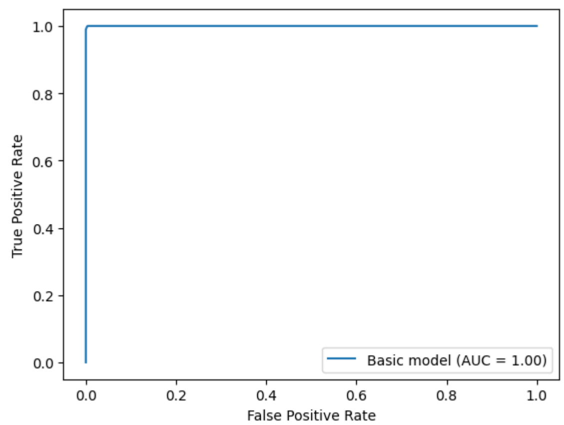


Figure 2. AUC curve for the basic tree

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 model to be built.

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 based on empirical testing. Even when we set lower / higher values, the best estimator found by GridSearchCV 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 as min_samples_split.

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

After deciding how to build the tree, the next step is to decide which metric GridSearchCV should use to decide the best estimator. We used maximizing accuracy, precision, recall and F1 as our different criterion, and summarized the results in table **??** and in figure **??**. As usual, each model was trained with `random_state = 2211`, and evaluated using 10 folds CV.

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

Out of the 4 criterion chosen, figure **??** shows that the models generally performed equally, so which out of the four models to choose will depend on the costs of misclassifying the customers, to be handled in section **??**. For now, we will pick the fourth model (F1 score as criteria) as the benchmark for this subsection, since it offers a balanced consideration of both precision and recall.

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1}$$

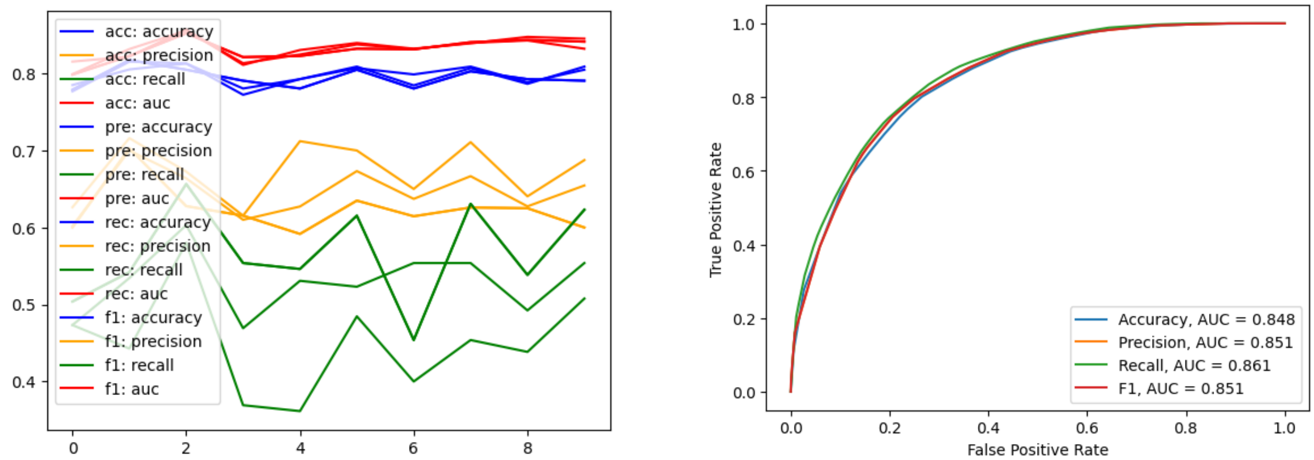
2.3. Ensemble trees

Recall in section **??**, 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 **??**.

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 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.



(a) the performance of the four models across folds

(b) AUC curve for the four models

Figure 3. Performance of the four models in §**??**

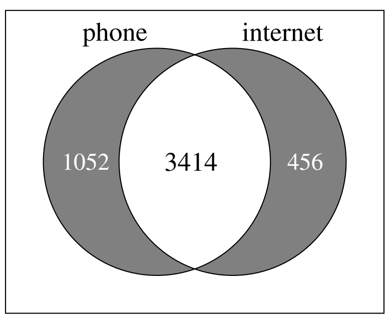


Figure 4. subscription service provides a natural split

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

We then used the following formula to convert the score back to the whole training set, to mimic the effect of setting the first split[1]:

$$\text{Score} = \frac{1052 \times P + 3414 \times P \cap I + 456 \times I}{4922} \tag{2}$$

[1]This method does not make too much sense for the AUC metric, but we tried it on the whole training set and the result was similar.

Tree	Accuracy	Precision	Recall	AUC
Ensemble	0.790	0.551	0.500	0.680

Table 4. Ensemble tree performance

where

- *Score* denotes the specific metric in question (accuracy, precision , recall, AUC),
- P denotes the metric obtained from the model trained using only phone subscribers,
- $P \cap I$ denotes the metric obtained from the model trained using phone and internet subscribers, and
- I denotes the metric obtained from the model trained using only internet subscribers

The results of the Ensemble model are summarized in table **??**.

2.4. Choosing the best model

Based on table **??**, we choose the adjusted parameters model in section **??** to be our most effective model.

Recall that before we started training the 3 different types of models, we split off 30% of the dataset to be used as our testing set. Now that we have chosen our final model, we can use the testing set to evaluate the effectiveness of our model.

The results on the testing set are summarized in table **??**. The results are similar, indicating that overfitting was unlikely.

Figure **??** shows the relative feature importance for the attributes that contributed to reducing entropy in the leaf

Tree	Accuracy	Precision	Recall	AUC
Adj. params (§2.2)	0.787	0.604	0.602	0.729

Table 5. model performance on testing set

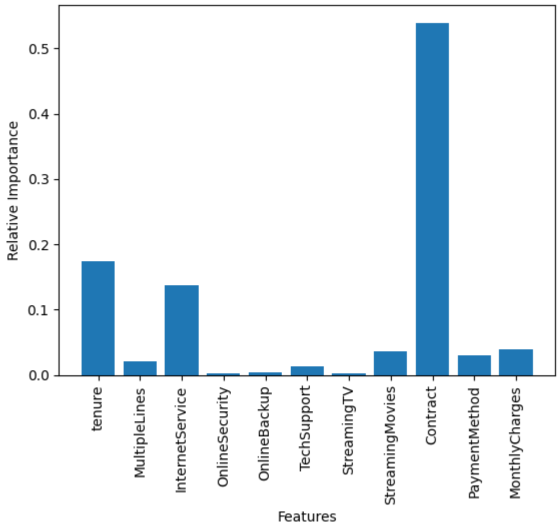


Figure 5. the relative feature importance for various attributes

Contract	Churn	Total	Churn Ratio
Monthly contract	1655	3875	42.7%
Yearly contract	166	1472	11.3%
Bi-yearly contract	48	1685	2.85%

Table 6. proportion of customers that churn, sorted by contract type

nodes. The factor that by far contributed the most was the attribute "contract", which had three possible values: (1) Month-to-month, (2) One year, and (3) Two year .

We then tried to find out which contract type of customers were most likely to churn. The results are shown in table **??**.

This result makes sense and matches our intuition - if customers signed monthly contracts, they can easily cancel their Telco subscription and switch to another service provider at the end of every month. On the other hand, the longer the contract, the less likely the customers would cancel their contract, as cancelling a longer contract requires a larger penalty for breaching the contract, and once they miss their contract cancellation window they will be automatically renewed for another yearly or bi-yearly period.

Regularization	Accuracy	Precision	Recall	AUC
None (§**??**)	0.803	0.654	0.544	0.841
L_1 (§**??**)	0.803	0.654	0.543	0.841

Table 7. logistic regression model performances

3. Logistic Regression *(Task 1, Question 2)*

We build two logistic regression models, one with no regularization and one using L_1 regularization. To ensure fair a comparison, both models were

- trained with the dataset normalized,
- trained with `random_state = 2211`,
- trained until they converged (tolerance $< 10^{-4}$) and
- fit with an intercept.

We report the accuracy, precision, recall and AUC score in table **??**.

3.1. No regularization

The model's parameters are shown in table **??**.

The loss across folds is shown in figure **??**, and the AUC curve is shown in figure **??**.

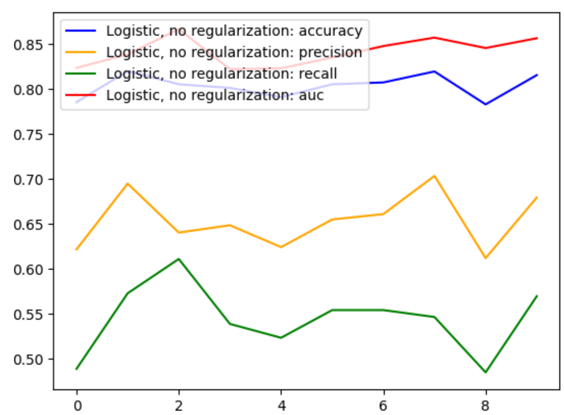


Figure 6. model performance across different folds

When we compare it with the best decision tree model built in section **??**, we find that the logistic regression model performed marginally better. Besides having a slightly lower recall, the accuracy, precision and AUC score were slightly higher.

However, as we know from section **??**, only 11 out of the 18 attributes contributed to reducing the impurity in the leaf nodes. When we take a look at the parameters for the model (table **??**), we observe that the top two factors with the greatest contribution were the same (tenure and contract),

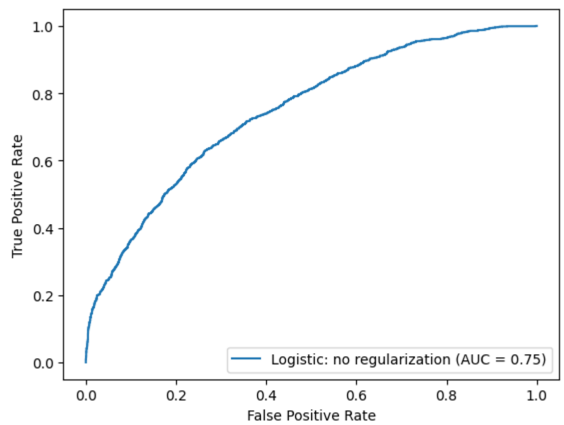


Figure 7. ROC curve with no regularization

Coefficient		**Coefficient**	
gender	0.0183	OnlineBackup	-0.0449
SeniorCitizen	0.078	DeviceProtection	-0.0121
Partner	0.0496	TechSupport	-0.32
Dependents	-0.0999	StreamingTV	0.209
tenure	-0.844	StreamingMovies	0.298
PhoneService	-0.223	Contract	-0.526
MultipleLines	0.158	PaperlessBilling	0.169
InternetService	0.76	PaymentMethod	-0.197
OnlineSecurity	-0.212	MonthlyCharges	-0.0192
Intercept	-1.64		

Table 8. logistic regression (§**??**) model parameters

but the logistic regression model without regularization was not able to drop any parameters. We expect that using regularization would help us improve our model performance.

3.2. Lasso regularization

We varied the regularization parameter c from 0.005 to 3, and computed the 10 fold cross validation scores for accuracy, precision, recall and AUC. The graph is shown in figure **??**.

The results shows that the accuracy, precision, recall and AUC are relatively fixed for $c > 5 \times 10^{-2}$. Based on this finding, we set $c = 0.05$ and trained the model. The parameters estimated are shown in table **??**.

We see that the parameters such as "gender", "partner", "OnlineBackup" and "DeviceProtection" are dropped by the logistic regressor. This makes sense, because we would not expect somebody's gender or marital status to contribute to the cancellation their contract.

As for "MonthlyCharges", the reason the regressor drops this parameter is not too apparent, and requires further investigation. We investigate by considering the following two

Coefficient		**Coefficient**	
gender**	0	OnlineBackup**	0
SeniorCitizen*	0.077	DeviceProtection**	0
Partner**	0	TechSupport*	-0.24
Dependents*	-0.0585	StreamingTV*	0.142
tenure	-0.749	StreamingMovies*	0.227
PhoneService*	-0.113	Contract	-0.489
MultipleLines*	0.0583	PaperlessBilling	0.15
InternetService	0.694	PaymentMethod	-0.19
OnlineSecurity*	-0.146	MonthlyCharges***	0
Intercept	-1.52		

Table 9. parameters with L_1 regularization (§**??**), $c = 0.05$.
* indicates parameters dropped when $c = 0.005$
** indicates parameters dropped when $c = 0.05$
*** indicates parameters dropped when $c = 0.5$

questions:

1. Why is "MonthlyCharges" dropped in logistic regression, but not in our decision tree?
2. Why do both logistic regression, and decision tree decide to keep the attribute "Contract"?

To investigate the first question, we plot "MonthlyCharges" against our target variable "Churn". The graph is shown in figure **??**. From figure **??**, we see that there is no clear-cut relationship between "MonthlyCharges" and "Churn", or put simply we cannot use a function of the form

$$\text{Churn} = \frac{1}{1 + e^{-(c_1 \times \text{MonthlyCharge} + c_2)}} \tag{3}$$

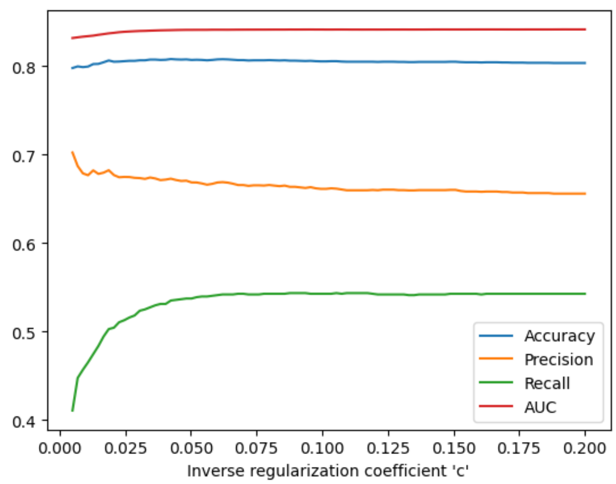
to separate them. For the attribute "Contract" though, table **??** shows that there is a clear-cut relationship between "Contract" and "Churn": as the contract time increases, the probability of churning reduces.

This raises the question: why are decision trees able to make use of "MonthlyCharges", when there is no clear-cut relationship? The reason is that an attribute can be used multiple times within a tree. For example, we are able to build a tree that achieves the following:

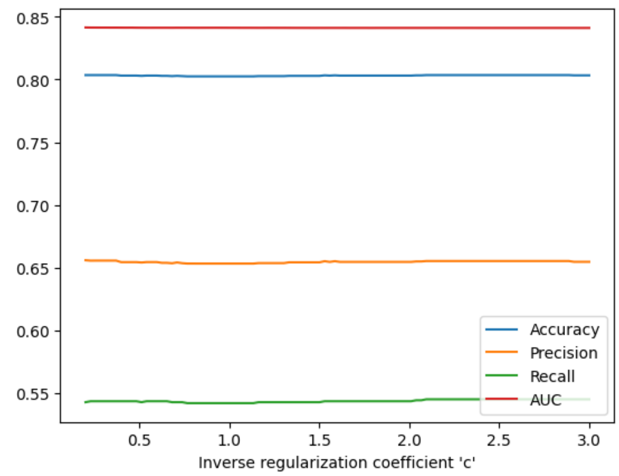
$$\text{Churn} = \begin{cases} 0 & \text{if MonthlyCharges } \in (0, 50) \cup (100, \infty) \\ 1 & \text{if MonthlyCharges } \in [50, 100] \end{cases} \tag{4}$$

as shown in figure **??**. However, due to the functional form of a logistic regression model, we are only able to split a continuous attribute into two disjoint *intervals*. We are unable to split a continuous attribute as follows:

$$\text{Class } 0 \Leftrightarrow \text{MonthlyCharges} \in (0, 50) \cup (100, \infty) \tag{5}$$
$$\text{Class } 1 \Leftrightarrow \text{MonthlyCharges} \in [50, 100] \tag{6}$$



(a) the effect of regularization for $c \in (0.005, 0.2)$



(b) the effect of regularization for $c \in (0.2, 3)$

Figure 8. effect of varying the regularization strength on performance metrics

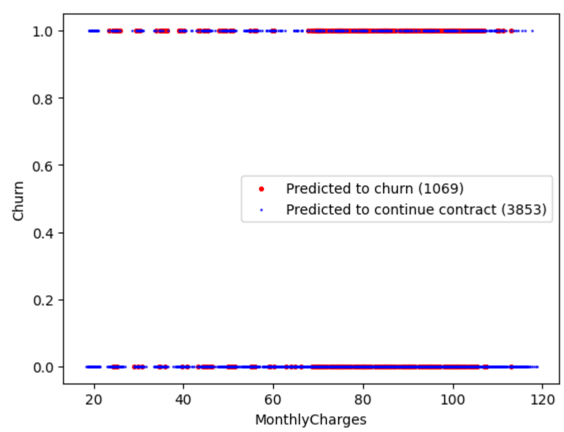


Figure 9. MonthlyCharges plotted against Churn

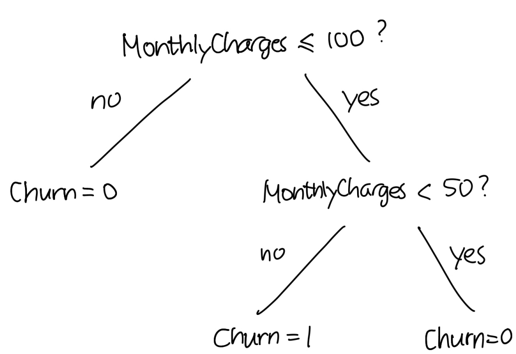


Figure 10. A tree that partitions a continuous attribute into multiple disconnected sets

Regularization	Accuracy	Precision	Recall	AUC
None (§**??**)	0.790	0.629	0.532	0.845
L_1 (§**??**)	0.794	0.638	0.541	0.845

Table 10. Evaluating the models in §**??** and §**??** on the testing set

since $(0, 50) \cup (100, \infty)$ is not an interval. This explains why the "good" features identified by the two models are different.

3.3. Evaluating the model

For good measure, we evaluate the models on the 30% testing set we split off from the start. The performace metrics are given in table **??**.

The models achieve similar scores on the testing set as on the training set, which means the likelihood of overfitting is low. Also, after dropping some attributes in the L_1 regularization model, we find that the results do not differ much from the model without regularization. Out of 2,110 samples, 2,104 of them had the same predicted class.

This makes sense, because L_1 regularization helped us drop the variables that did not contribute to our prediction. Since the original model in section **??** did not overfit, we should expect both models to perform similarly.

4. Incorporating Misclassification Costs *(Task 2, Question 3)*

In reality, wrong predictions may lead to classification costs. Figure **??** shows the confusion matrix for the adjusted parameters decision tree (§**??**), and figure **??** shows the confusion matrix for the L_1 logistic model (§**??**).

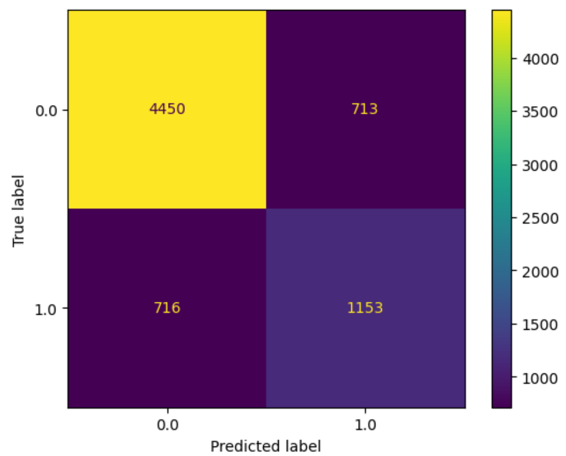


Figure 11. confusion matrix for the adjusted parameters tree (§??)

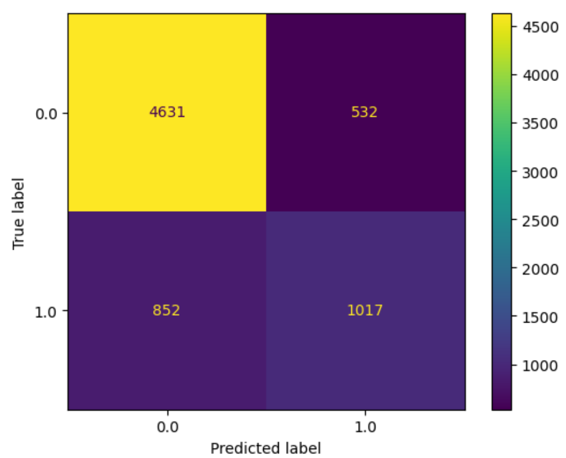


Figure 12. confusion matrix for the L_1 logistic model (§??)

Cost	Predicted (+)	Predicted (-)
Actual (+)	0	CV-205
Actual (-)	205	0

Table 11. cost matrix

Given the cost matrix in table **??**, the costs of misclassification for the decision tree model is shown in table **??**. The costs of misclassification for the logistic regression model is shown in table **??**.

Using the default threshold, the decision tree offers a lower overall cost. If we were allowed to data-mine the optimal thresholds, figure **??** shows the effect of adjusting the thresholds on the total cost. Table **??** shows the optimal thresholds that minimizes the total cost for each model. The minimum total cost generated by either model is therefore $481,506.

Cost	Predicted (+)	Predicted (-)
Actual (+)	0	$ 439,881
Actual (-)	$ 146,165	0
Total cost	$ 586,046	

Table 12. cost matrix for decision tree

Cost	Predicted (+)	Predicted (-)
Actual (+)	0	$ 496,748
Actual (-)	$ 109,060	0
Total cost	$ 605,808	

Table 13. cost matrix for logistic regression

Model	Optimal θ	Total cost
Decision Tree	0.203	$ 487,892
Logistic Regression	0.265	$ 481,506

Table 14. the total cost for optimal thresholds

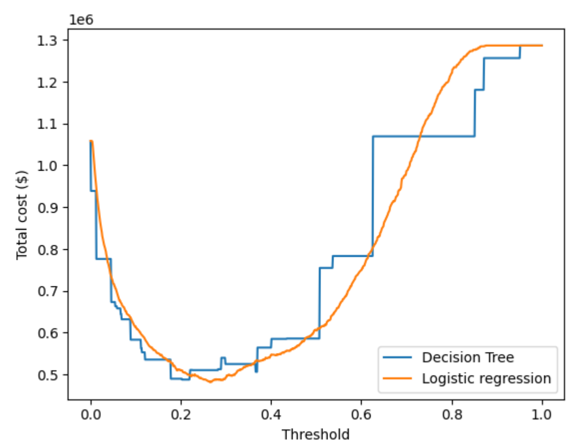


Figure 13. the effect of changing the decision threshold against the total cost

To test if our model performs better than the baseline model, we constructed two baseline models, one where everyone is predicted to churn, and one where everyone continues their contract. The results are summarized in table **??**. This coincides with setting the decision threshold as $\theta = 0$ and $\theta = 1$, as shown in figure **??**.

To conclude, the data-driven solution *significantly* reduces the total cost borne by Telco for misclassifying its customers. Even without setting data-mining the optimal decision threshold θ, using the default value of $\theta = 0.5$ will still save the company more than half the possible losses from misclassification.

Model	Total cost
Baseline: Predict all +	\$ 1,058,415
Baseline: Predict all -	\$ 1,286,425

Table 15. Baseline model's performance

Cost	Condition 0	Condition 1
Actual (+)	0	CV-205
Actual (-)	205	0

Table 16. new cost matrix

Model	Condition 0	Condition 1	Total Cost
Decision Tree	\$ 228,575	\$ 505,292	\$ 733,867
Logistic Regression	\$ 200,900	\$ 394,508	\$ 595,408

Table 17. new cost matrix, incorporating expected value

5. Providing Retention Offers *(Task 2, Question 4)*

Under the modified policy for giving out retention packages, we will suffer a loss under the following two conditions:

- **Condition 0**: The model predicts the customer will cancel the contract and the expected value is positive, so we send out a retention package, but it turns out the customer was originally not planning to cancel the contract. In this case, we lose the cost of the retention package (\$205) per customer in this category.
- **Condition 1**: The model predicts the customer will not cancel, or the expected value is negative, so we do not send out a retention package. But in reality, the customer does end up terminating their contract, and by not sending a retention package we lost this customer. In this case, we lose the contract value (CV) of this customer but save \$205 for the retention package, netting us a loss of \$CV-205.

Under this framework, the new cost matrix is shown in table **??**.

Figure **??** shows the confusion matrix for the decision tree model, and figure **??** shows the confusion matrix for the logistic regression model. Notice that compared with section **??**, the number of those in the column `predicted = 1` has reduced. This makes sense, because we now have an additional requirement (the expected value has to be positive) to give out the retention packages.

Setting the decision threshold as $\theta = 0.5$, the total costs for the models are given in table **??**.

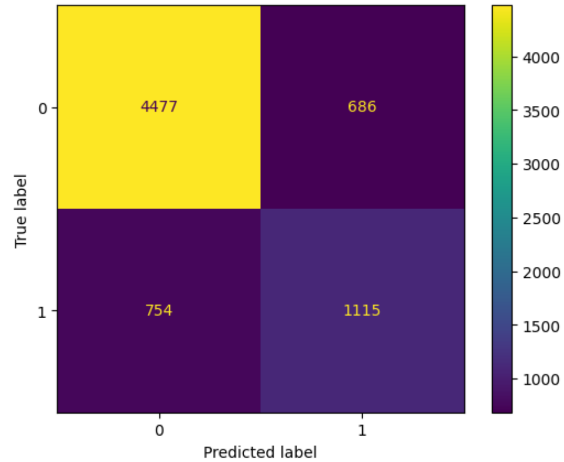


Figure 14. confusion matrix for the adjusted parameters tree (§**??**), incorporating expected value

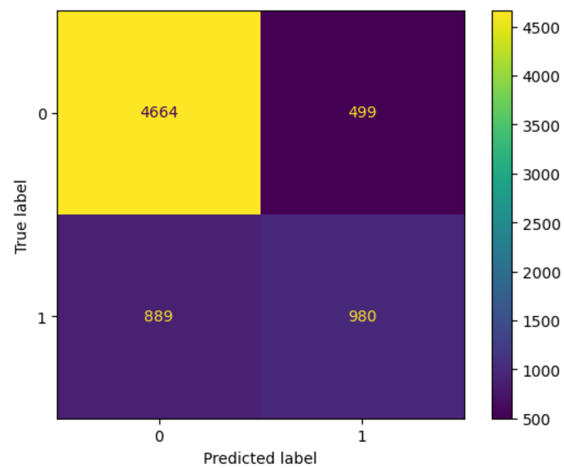


Figure 15. confusion matrix for the logistic regression model (§**??**), incorporating expected value

Similarly, we can adjust the decision threshold θ to find the optimal threshold that minimizes the total cost. The graph of θ against total cost can be found in figure .