CE 264 Problem Set 6: Machine Learning

Jiajian Lu (3033084290)
Kun Qian (3033030782)
Franklin Zhao (3033030808)

26 Apr. 2018

Problem 1

1)

The plot of accuracy vs. training size is shown as follow:

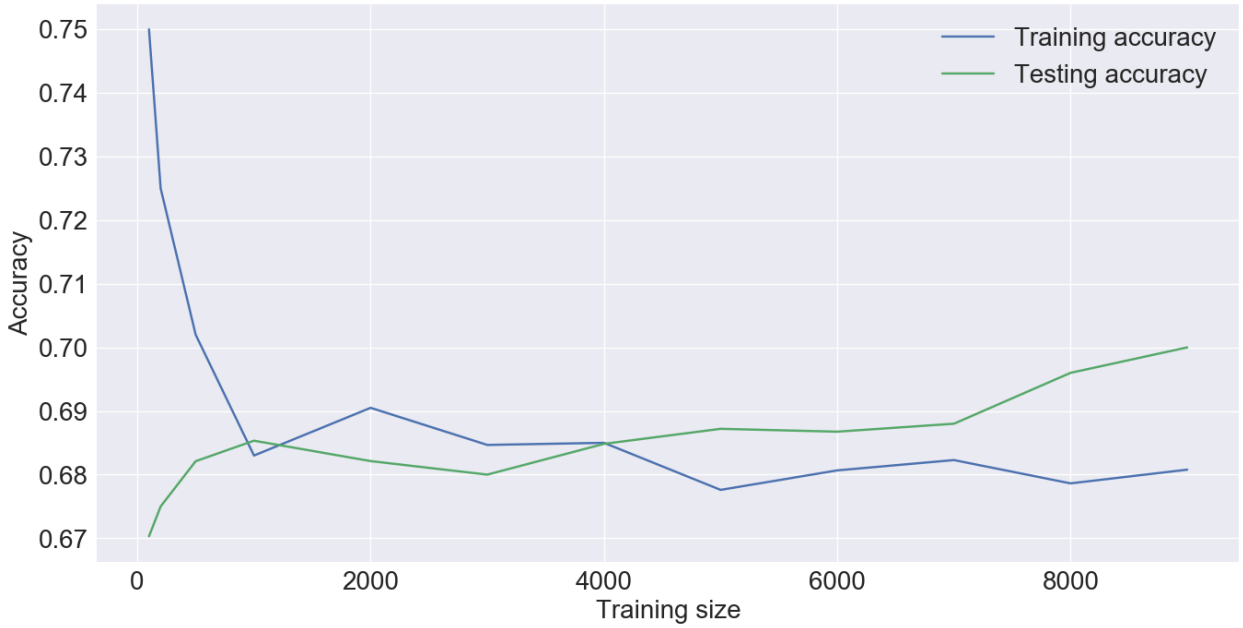

Fig. 1: Accuracy vs. training size

The training accuracy decreases from 0.75 to 0.702 and the testing accuracy increases from 0.67 to 0.68 when the training size increases from 100 to 500. The reason is that when the training size is small, the model can easily fit the training data but hard to predict the test set.

As the training size increases, training accuracy oscillates and continues to decrease and testing accuracy increases. The reason is that the model captures more information from the data and generalizes better but not flexible enough to fit all the training data.

2)

The plot of average log-likelihood vs. training size is shown as follow:

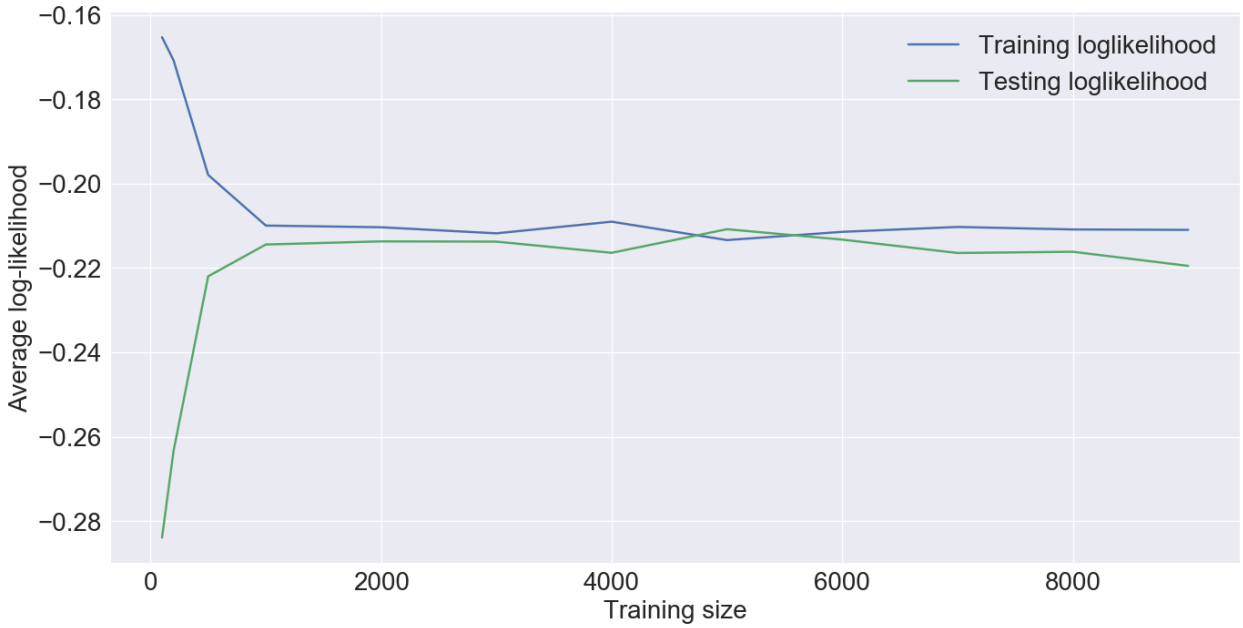

Fig. 2: Average log-likelihood vs. training size

3)

The average training log-likelihood and the average testing log-likelihood of the model have the same trend with the training accuracy and testing accuracy. The reason is that average log-likelihood is the criteria for MNL to fit the model while accuracy is the criteria for prediction.

4)

Because the log-likelihood is:

$$LL = \sum_{n \in N} \sum_{k \in C_n} I(y_n = k) ln(p(y_n = k)) \tag{1}$$

where N is the total number of respondents, C_n is the choice set of person n and y_n is the actual choice for person n.

As the training size increases, LL would definitely decrease because LL is confounded by training size. When using the average log-likelihood, the direct effect of training size is eliminated.

5)

Discuss the trends we see in the accuracy and average LL as training size grows. Why does the mean trend behave this way? Why does the standard deviation behave this way?

The mean trend for the training accuracy is high and for the testing accuracy is low when the training size is small. Then they converge as the training size increases. The reason is that when the training size is small, the model can easily fit all of them but hard to predict. As the training size grows, the model captures more information from the data and generalizes better but not flexible enough to fit more data which leads to the convergence of the testing accuracy and training accuracy. The mean trend for average LL behaves in this way for the same reason.

The standard deviation for training accuracy decreases as training size grows because the accuracy follows normal distribution $N\left(\mu, \frac{\sigma}{n}\right)$ as training size increases and the standard deviation will decrease. On the other hand, when the training size increases, the testing size decreases which will results in the increase of standard deviation for testing accuracy. The standard deviation for average LL behaves in this way for the same reason.

Problem 2

1) k-Nearest Neighbors

For kNN, we tuned the hyperparameter K to see the performance of the model and find the best value. The result is shown as follow:

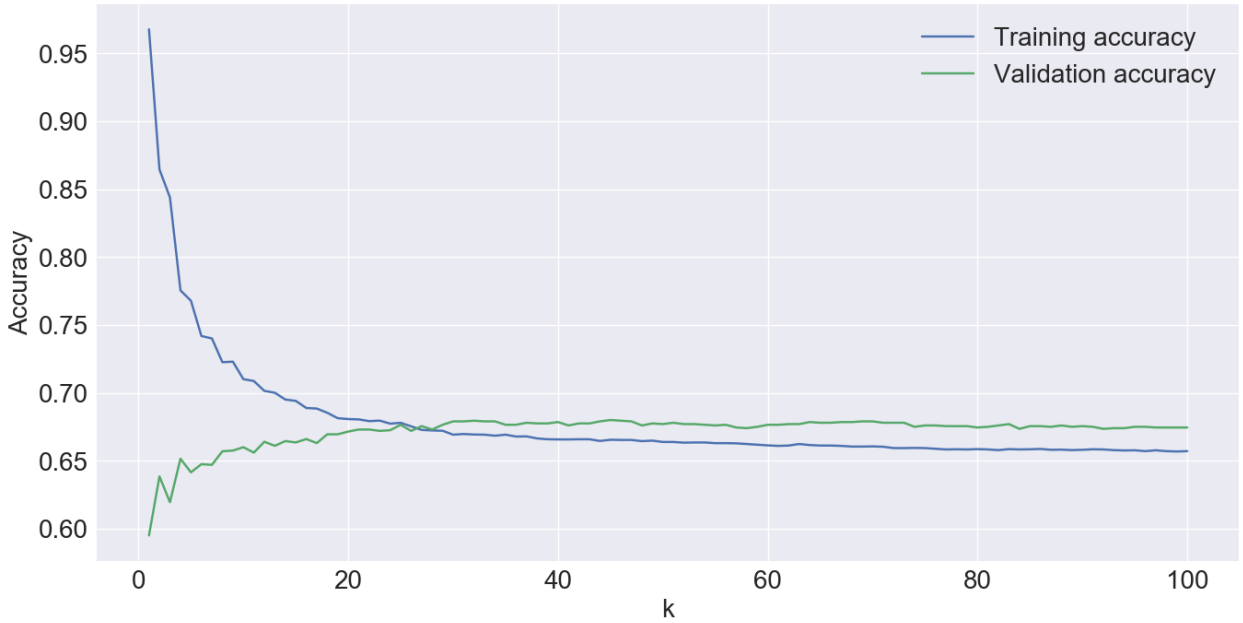

Fig. 3: K vs. accuracy in kNN

From the plot we notice that as k goes up, training accuracy decreases and validation accuracy increases since the model would be overfitting when k is not large enough. However, when k is large enough, increasing k would not make significant change but only increase the computation complexity. In our experiment, the best k in terms of accuracy is 45.

2) Decision Tree

For decision tree, we tuned 2 hyperparameters: one is minimum number of samples split an internal node, the other is the depth. The results are shown as follows:

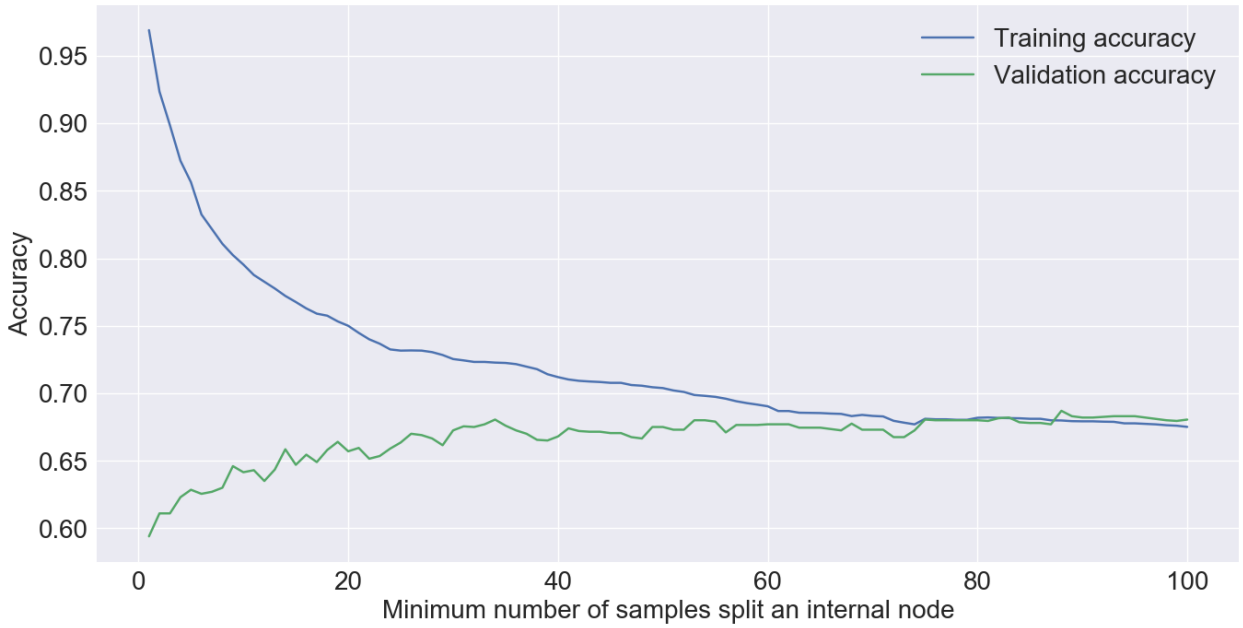

Fig. 4: Minimum number of samples vs. accuracy in decision tree

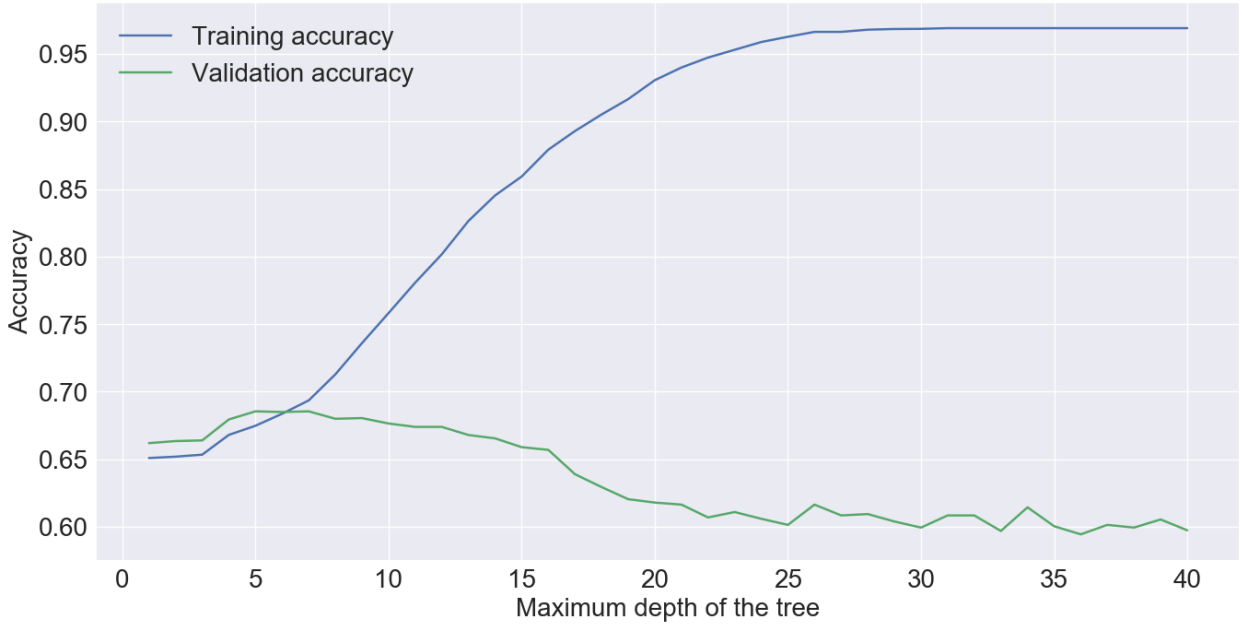

Fig. 5: Maximum depth vs. accuracy in decision tree

For minimum number of samples, it performs similar to k in kNN model as the model would be overfitting when it is small. For maximum depth, there is a bias-variance trade-off when tunning this hyperparameter. Small depth results in underfitting while two much depth results in overfitting. The best number of samples and depth in our model are 88 and 5, respectively.

3) Random Forest

For random forest, we also tuned 2 hyperparameters: one is the number of trees, the other is the depth of the trees. The results are shown as follows:

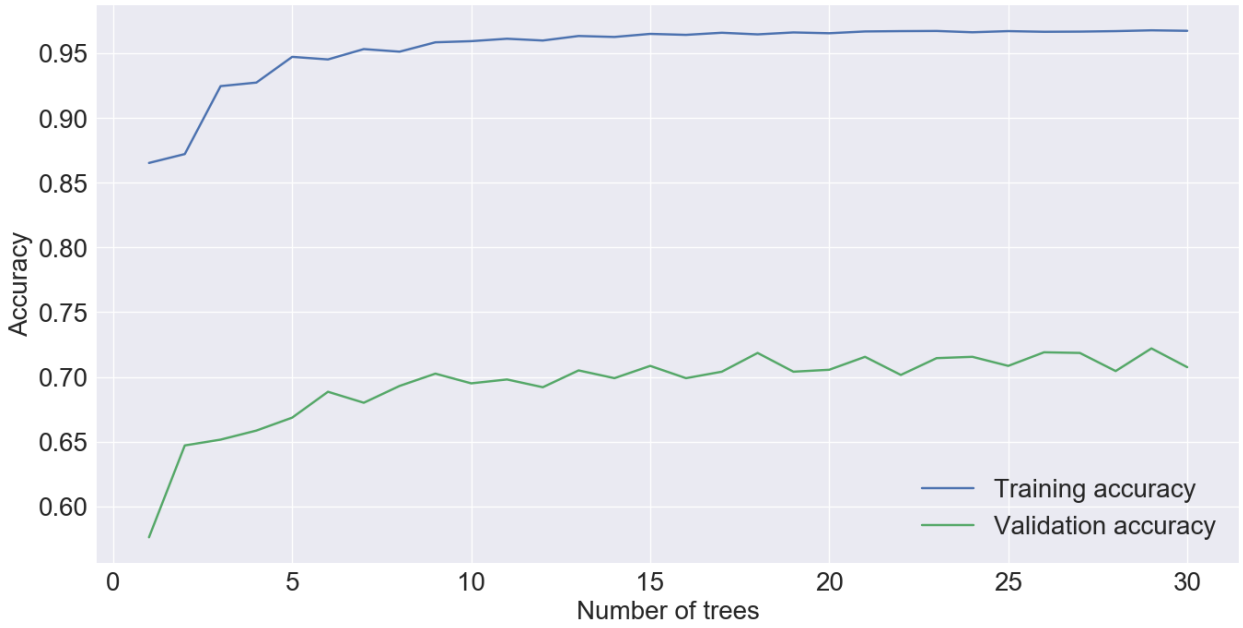

Fig. 6: Number of trees vs. accuracy in random forest

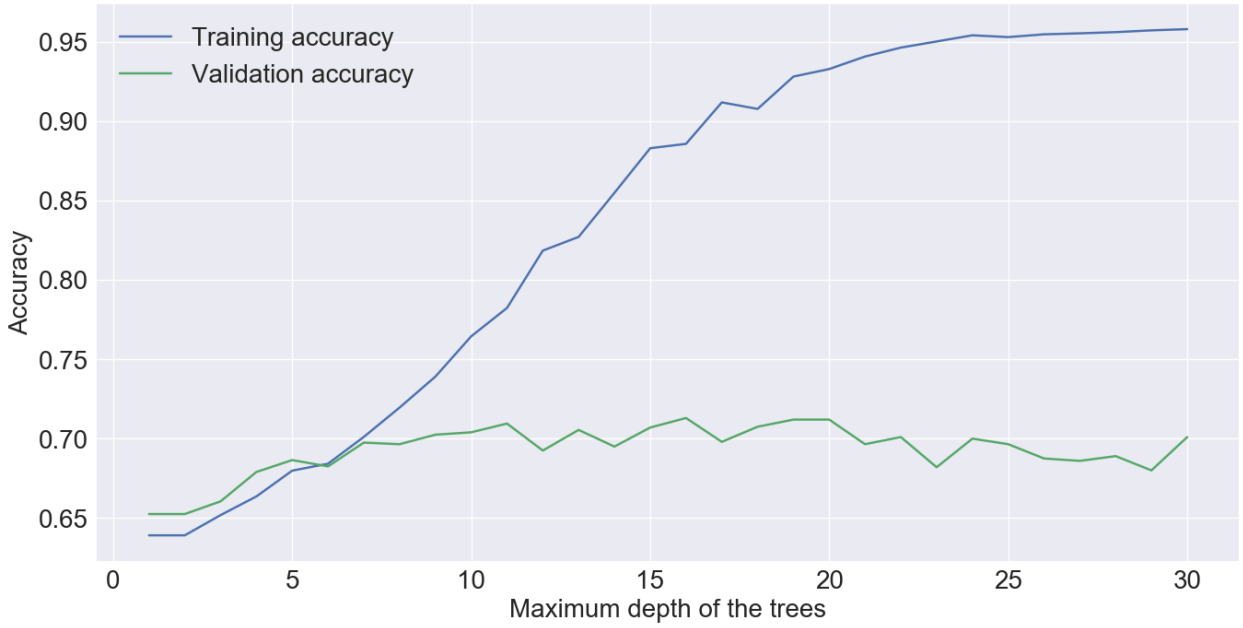

Fig. 7: Maximum depth vs. accuracy in random forest

For number of trees, when it is small, adding the number would significantly improve the model performance, and the performance would not improve much when the hyperparameter reaches some point. The depth is roughly the same as the corresponding hyperparameter in decision tree model. The selected best hyperparameters in this model are 27 and 16.

4) Support Vector Machine

For SVM, we only tuned the regularization term C. The result is shown as follow:

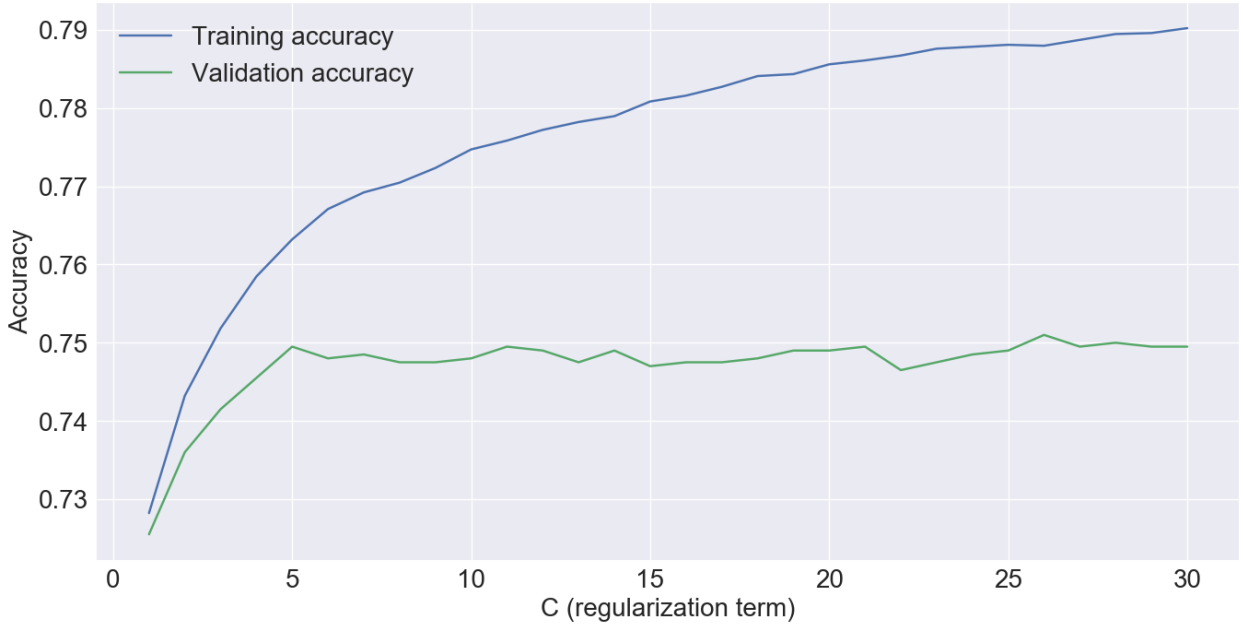

Fig. 8: C vs. accuracy in random forest

From the result we notice that increasing C would increase the accuracy since the margin is decreasing. However, too large C would result in overfitting and make the model very sensitive. The best value of C in our model is 25.

5) Highest accuracy of our models

All the algorithms we used with their highest accuracy are shown in the following table:

Table 1: Models and their highest accuracy

Model	**Highest accuracy**
MNL	0.7
kNN	0.68
Decision Tree	0.687
Random Forest	0.7235
SVM	0.51

6) Discussion

Comparing to what we have been doing in the class, what we are doing now is just a single validation for each hyperparameter while in the class we did cross-validation. They are similar since both of them are validation process, which split the data into training set and validation set. However, k-folds cross validation requires split the data into k subsets and each of them can be the validation set in certain step. Hence, the whole data can be evaluated for the model in order to tune the hyperparameters. We would notice the overfitting issue in the plots if we did the cross validation. However, if we only did the single validation like we did in this problem set, some overfitting issues might not be noticed.

Problem 3

1)

The missing set of variables are variables related to the availability of different methods. The first tree misses this variable. (According to the clarification, we know that the second tree used all the parameters to train but didnt show the cost in the tree.) These two plots are very different because they depend on different variables (one of them doesnt consider the availability of different methods), so they get very different results. This shows that the missing variable is a very useful one because according to the children nodes of the missing variables, they are divided explicitly into two parts with similar amount. And the further classes turn into pure classes. Its easy to understand the power of the indicator-availability. Because on a node to decide if a choice is available, we will filter away those unavailable, thus get a relatively purer classification.

2)

Random forest method is expected to perform better under this case. For bagging, sometimes random sampling is not random enough thus will make the tree look similar. Also, stronger predictors will always split the tree in the same way. If the trees look the same, taking average will not reduce the variance. With random forest, we can randomly sample from the features so that will decorrelate trees, as a result strong predictors will not be selected every time and will get different split.

3)

The meaning of this vector: Because there are 6 choices for people to choose from, the vector indicates the number of people choosing different cases. And **yes**, we can see that on each leaf, there is a number indicating the corresponding sample amount n_i and the total number of samples are N. We can calculate out the probability of distribution through $f_i = \frac{n_i}{N}$.

Problem 4

With the MNL method, we calculate the probability distribution of outcomes along 100 individuals:

$$[p(y_n = car), p(y_n = transit)] = \left[\frac{e}{e+1}, \frac{1}{e+1}\right] \tag{2}$$

Then we will predict the sample has $\frac{e}{e+1}$ portion choosing car and $\frac{1}{e+1}$ portion choosing transit when they are uniformly drawn from the population.

With machine learning method, for every person, we calculate his/her choice probability:

$$[p(y_n = car), p(y_n = transit)] = \left[\frac{e}{e+1}, \frac{1}{e+1}\right] \tag{3}$$

Then we will predict this person choose car since $p(y_n = car)$ is the highest. Therefore, for a sample of 100 individual, we all predict them choosing car with machine learning method.