Due data: 10/18/2023, end of the day. **Please submit an .ipynb file via Canvas.**

Instructions:

1) The .ipynb file shall include not only the **source code**, but also necessary **plots/figures** and **discussions** which include your *observations*, *thoughts* and *insights*.
2) Please avoid using a single big block of code for everything then plotting all figures altogether. Instead, use a small bock of code for each sub-task which is followed by its plots and discussions. This will make your homework more readable.
3) Please follow common software engineering practices, e.g., by including sufficient **comments** to functions, important statements, etc.

Programming Problem:

In this programming problem, you will get familiar with building a decision tree, using cross validation to prune a tree, evaluating the tree performance, and interpreting the result.

Potential packages to use and short tutorials:
(1)http://scikit-learn.org/stable/modules/tree.html
(2)http://chrisstrelioff.ws/sandbox/2015/06/25/decision_trees_in_python_again_cross_validation.html

```
from sklearn import tree # tree library
tree.DecisionTreeClassifier() # for classification tree
tree.DecisionTreeRegressor() # for regression tree
# X: design matrix; Y: labels
fit(X, Y) # fit a tree
predict(X) # make prediction on test data
tree.export_graphviz(model) # visualize  tree
from sklearn.model_selection import KFold # K-fold cross validation
```

```
from sklearn.grid_search import GridSearchCV
In python, you may have to do gridsearch and cross validation using
```

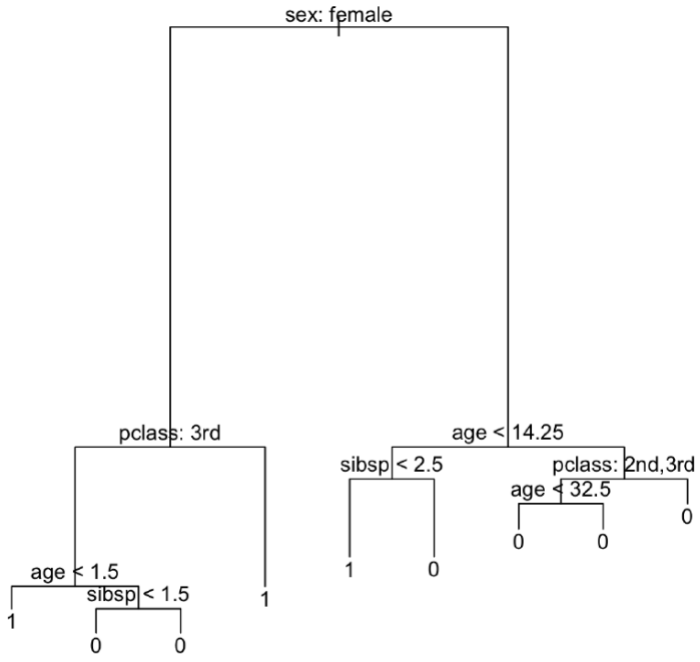
```
GridSearchCV() to choose the best parameters. Try use different values for
"max_leaf_nodes": [None, 1,2,3,4,5,6,7,8,9], (see reference 2).
```

classification tree

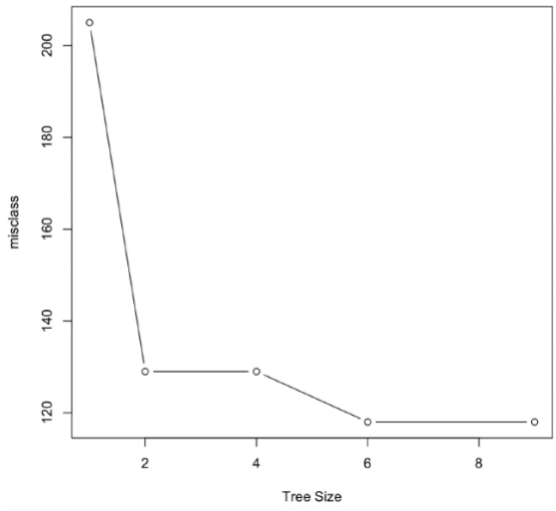
Use the titanic.csv dataset included in the assignment.

Step 1: Read in Titanic.csv and observe a few samples, some features are categorical and others are numerical. Take a random 70% samples for training and the rest 30% for test.

Step 2: Fit a decision tree model using independent variables 'pclass + sex + age + sibsp' and dependent variable 'survived'. Plot the full tree. Make sure 'survived' is a qualitative variable taking 1 (yes) or 0 (no) in your code. You may see a tree similar to (not necessarily the exact same as) this one:



Step 3: Use cross-validation to find the best parameter to prune the tree. You should be able to plot a graph with the 'tree size' as the x-axis and 'number of misclassification' as the Y-axis. You may have a plot similar to (not necessarily the exact same as) below:



Step 4: Find the tree size that yields a minimum number of misclassifications. Choose the optimal tree size to prune the tree and plot the pruned tree (which shall be smaller than the tree you obtained in Step 2). Report the accuracy of pruned tree on the test set for the following:

percent survivors correctly predicted (on test set)
percent fatalities correctly predicted (on test set)

Step 5: Use the *RandomForestClassifier()* function to train a random forest using the optimal tree size you found in Step 4. You can set n_estimators as 50. Report the accuracy of random forest on the test set for the following:

percent survivors correctly predicted (on test set)
percent fatalities correctly predicted (on test set)

Check whether there is improvement as compared to a single tree obtained in Step 4. If not, please discuss the potential reasons.