Advanced Machine Learning Homework One

Matthew Ferguson

September 16th, 2022

Requirements

Develop models to predict HAS_DIV from the census supplement dataset:

- Convert adjusted gross income, age, and weeks worked to binary indicators, where values above the mean are 1 and values below the mean are 0.
- Use entropy-based information gain to determine which of the binary predictors is the best to use for the top node in a decision tree to predict HAS_DIV.
- Create a set of decision tree classifiers using binary features. Iterate through max depth and information gain algorithms. Print out the confusion matrices and cross-validation scores of the models. Discuss.
- Create a set of decision tree classifiers using continuous features. Iterate through max depth and information gain algorithms. Print out the confusion matrices and cross-validation scores of the models. Discuss.
- Create a set of random forest classifiers using continuous features. Iterate through arguments of your choice. Print out the confusion matrices and cross-validation scores of the models. Discuss.

Part 1 Information Gain First Level

- Entropy of HAS_DIV:

$$-\frac{32\,955}{100\,000}\log_2\left(\frac{32\,955}{100\,000}\right)-\frac{67\,045}{100\,000}\log_2\left(\frac{67\,045}{100\,000}\right)$$

= 0.9144 bits

- Remainder of AGI_BIN:

$$\frac{27\,039}{100\,000}\left(-\frac{11\,764}{27\,039}\log_2\left(\frac{11\,764}{27\,039}\right)-\frac{15\,275}{27\,039}\log_2\left(\frac{15\,275}{27\,039}\right)\right)+\frac{72\,961}{100\,000}\left(-\frac{21\,191}{72\,961}\log_2\left(\frac{21\,191}{72\,961}\right)-\frac{51\,770}{72\,961}\log_2\left(\frac{51\,770}{72\,961}\right)\right)$$

= 0.901 bits (IG = 0.0134)

- Remainder of A_AGE_BIN:

$$\frac{48\,814}{100\,000}\left(-\frac{17\,225}{48\,814}\log_2\left(\frac{17\,225}{48\,814}\right)-\frac{31\,589}{48\,814}\log_2\left(\frac{31\,589}{48\,814}\right)\right)+\frac{51\,186}{100\,000}\left(-\frac{15\,730}{51\,186}\log_2\left(\frac{15\,730}{51\,186}\right)-\frac{35\,456}{51\,186}\log_2\left(\frac{35\,456}{51\,186}\right)\right)$$

= 0.9128 bits (IG=0.0016)

- Remainder of WKSWORK_BIN:

$$\frac{45\,267}{100\,000}\left(-\frac{16\,330}{45\,267}\log_2\left(\frac{16\,330}{45\,267}\right)-\frac{28\,937}{45\,267}\log_2\left(\frac{28\,937}{45\,267}\right)\right)+\frac{54\,733}{100\,000}\left(-\frac{16\,625}{54\,733}\log_2\left(\frac{16\,625}{54\,733}\right)-\frac{38\,108}{54\,733}\log_2\left(\frac{38\,108}{54\,733}\right)\right)$$

= 0.9118 bits (IG=0.0026)

- Remainder of A_SEX:

$$\frac{49\,116}{100\,000}\left(-\frac{16\,744}{49\,116}\log_2\left(\frac{16\,744}{49\,116}\right)-\frac{32\,372}{49\,116}\log_2\left(\frac{32\,372}{49\,116}\right)\right)+\frac{50\,884}{100\,000}\left(-\frac{16\,211}{50\,884}\log_2\left(\frac{16\,211}{50\,884}\right)-\frac{34\,633}{50\,884}\log_2\left(\frac{34\,633}{50\,884}\right)\right)$$

=0.9144 bits (IG=0)

- Information Gain:

$0.9144 - 0.901 = 0.0134$ bits for AGI_BIN
$0.9144 - 0.9128 = 0.0016$ bits for A_AGE_BIN
$0.9144 - 0.9118 = 0.0026$ bits for WKSWORK_BIN
$0.9144 - 0.9144 = 0$ bits for A_SEX

AGI_BIN is the feature with the highest information gain (IG = 0.0134 bits)

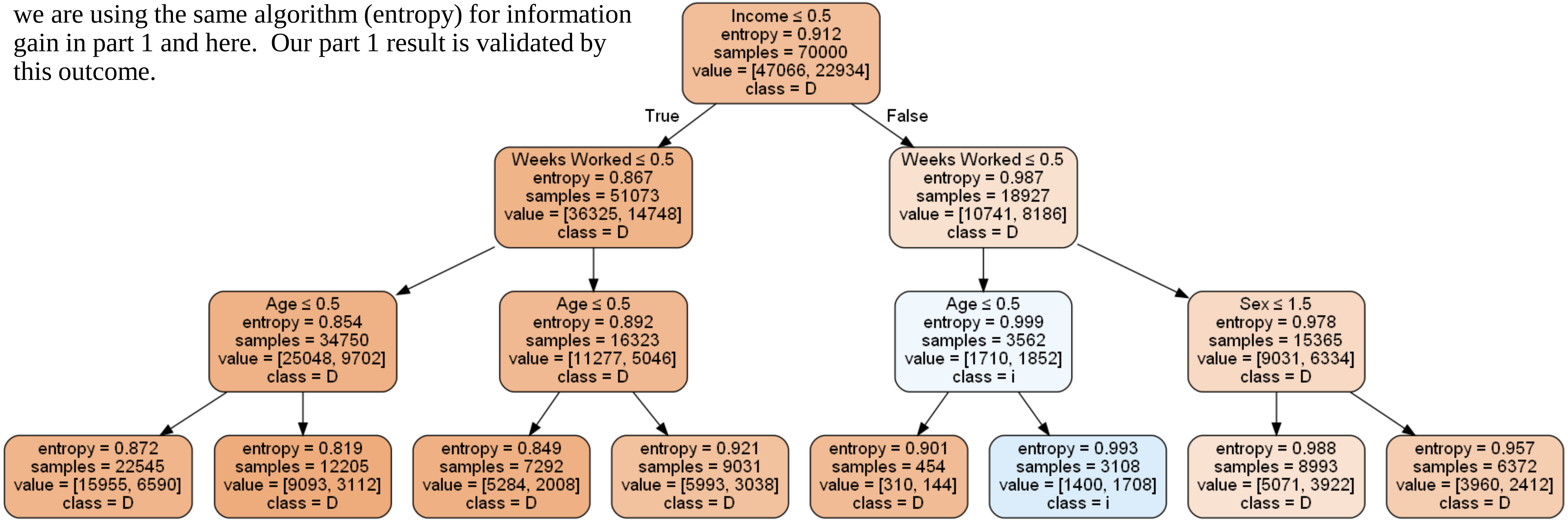
Code Review

Here we see the developed python code for Parts 2-4. We make a pandas dataframe by reading in the census data. The required binary features were calculated in excel and written to the census supplement before creating a dataframe. We normalize the dataframe using min max normalization. We extract the target and features (both binary and continuous) and store them in memory. We iterate through maximum depths of 3, 4, 5, and 10. We iterate through information gain algorithms of gini and entropy. Inside of our iterations we declare a decision tree classifier object and then fit it to the binary features and target. We obtain cross validation scores, have the model make a prediction on the test set, create a confusion matrix, and then print the results. We repeat the same modelling process for the continuous features and target. These decision tree models are exported to png using graph viz for the case of depth = 3 and information gain algorithm = entropy. Finally, we create a set of random forest models using continuous features. We iterate through the same depth values and information gain algorithms as before. In addition, we iterate through number of estimators/trees of 10, 100, and 1000.

```python
#AML Homework 1
import pandas as pd
from sklearn import tree, ensemble, model_selection, metrics
import pydot

df=pd.read_excel(r'C:\Data\Census_Supplement.xlsx')
df=(df-df.min())/(df.max()-df.min())

features_cont=df[['AGI','A_AGE','WKSWORK', 'A_SEX']].values.tolist()
features_bin=df[['AGI_BIN','A_AGE_BIN','WKSWORK_BIN', 'A_SEX']].values.tolist()
features='features_bin','features_cont'
target=df['HAS_DIV'].values.tolist()

fnames=['Income','Age','Weeks Worked', 'Sex']
tnames='Dividend'

depths=[10,5,4,3]
criteria = 'gini', 'entropy'

for crit in criteria:
    for depth in depths:

        (x_train_bin, x_test_bin, y_train_bin, y_test_bin) = model_selection.train_test_split(features_bin, target, test_size=0.3, random_state=22222)
        clf_bin=tree.DecisionTreeClassifier(max_depth=depth, criterion=crit)
        bin_tree=clf_bin.fit(x_train_bin,y_train_bin)
        acc_1=model_selection.cross_val_score(bin_tree, x_test_bin, y_test_bin, cv=5)
        y_pred_bin=bin_tree.predict(x_test_bin)
        conf_1=metrics.confusion_matrix(y_test_bin, y_pred_bin)
        print(crit, depth, 'binary', acc_1)

        if depth==3 and crit=='entropy':
            dot_data=tree.export_graphviz(bin_tree, out_file=None, feature_names=fnames, class_names=tnames, filled=True, rounded=True, special_characters=True)
            (graph,) =pydot.graph_from_dot_data(dot_data)
            graph.write_png(r'C:\Users\Matt\Desktop\Dividend_Tree_Binary.png')

        (x_train_cont, x_test_cont, y_train_cont, y_test_cont) = model_selection.train_test_split(features_cont, target, test_size=0.3, random_state=22222)
        clf_cont=tree.DecisionTreeClassifier(max_depth=depth, criterion=crit)
        cont_tree=clf_cont.fit(x_train_cont,y_train_cont)
        acc_2=model_selection.cross_val_score(cont_tree,x_test_cont, y_test_cont, cv=5)
        y_pred_cont=cont_tree.predict(x_test_cont)
        conf_2=metrics.confusion_matrix(y_test_cont,y_pred_cont)
        print(crit, depth, 'continuous', acc_2)

        if depth==3 and crit=='entropy':
            dot_data=tree.export_graphviz(cont_tree, out_file=None, feature_names=fnames, class_names=tnames, filled=True, rounded=True, special_characters=True)
            (graph,) =pydot.graph_from_dot_data(dot_data)
            graph.write_png(r'C:\Users\Matt\Desktop\Dividend_Tree.png')


estimators=[10,100,1000]
for crit in criteria:
    for estimator in estimators:
        for depth in depths:
            (x_train, x_test, y_train, y_test) = model_selection.train_test_split(features_cont, target, test_size=0.3, random_state=22222)
            rf=ensemble.RandomForestClassifier(max_depth=depth, criterion=crit, n_estimators=estimator)
            forest=rf.fit(x_train,y_train)
            acc_3=model_selection.cross_val_score(forest,x_test,y_test,cv=5)
            y_pred=forest.predict(x_test)
            conf_3=metrics.confusion_matrix(y_test,y_pred)
            print(crit, depth, estimator,acc_3)
```

Part 2 Binary Feature Decision Tree

We can see that the top-level decision in our binary feature tree is operating on Income (AGI_BIN). We expect this as we are using the same algorithm (entropy) for information gain in part 1 and here. Our part 1 result is validated by this outcome.



Part 2 Discussion and Results

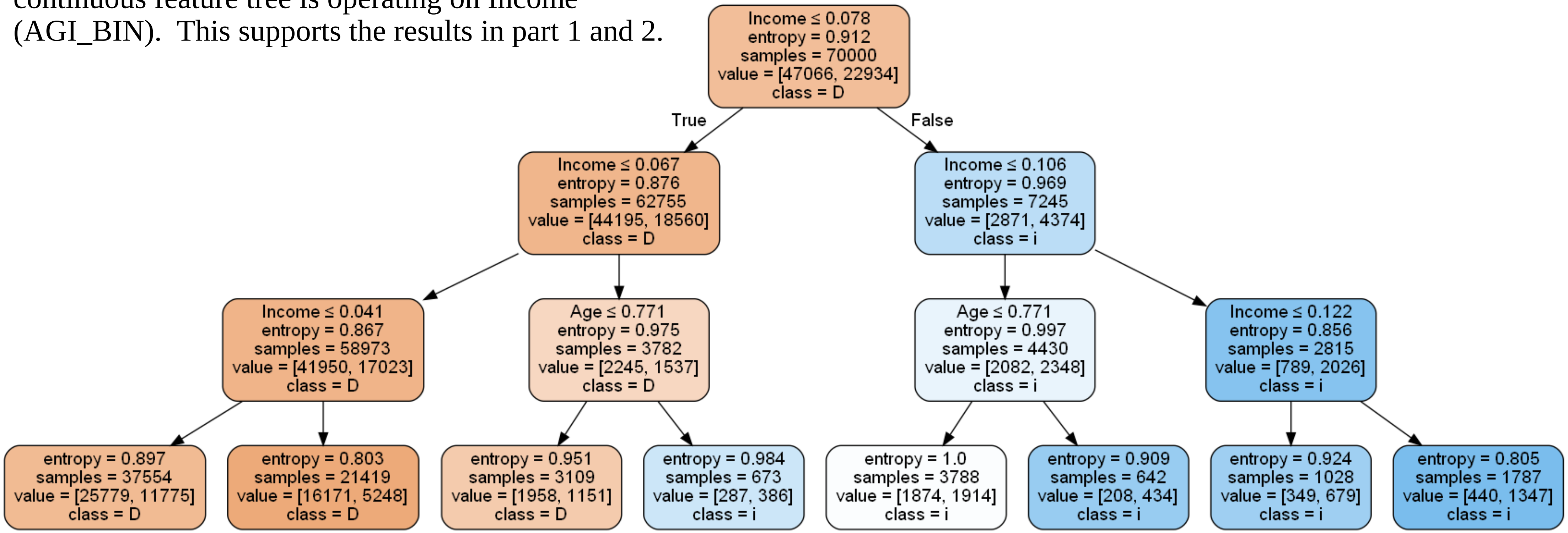
See the top table for cross-validation accuracies and the bottom table for the confusion matrices for our binary feature decision trees. Overall, we see that the highest average cross-validation classification accuracy obtained was 67.6% for the both gini and entropy criterion at a depth of 3. It appears that for our data set, additional depth does not help the model discriminate better. Information gain algorithms of gini and entropy had zero difference in resulting average accuracies. We see in our binary decision tree that the models perform almost the same across all parameters. There are only small changes in accuracies across parameters. We were initially concerned that a mistake had been made so we added depth parameters of 1 and 2 in order to validate that the model does perform worse when given the fewest possible decisions. The information gain calculations in part 1 show that the selected features are low information gain for the target. They have a high remainder. We suspect the reason being that the target appears in similar proportion across subpopulations of feature=0 and feature=1. No matter how many ways we cut the data the incidence of has div 0/1 is similar proportionally in feature splits, that is why information gain is so low. I think decisions beyond a certain point fail to add anything to discrimination when working with binary variables. It appears to me that you only really can make n=feature count meaningful number of decisions with binary data. If all four features are set equal to 1 in a tree, and the resulting subpopulation is 66% positive then you would have to make a low-confidence positive output. Note the bias in our confusion matrices towards false negatives which we explore in greater detail in part 4.

Algorithm	Depth	CV 1	CV 2	CV 3	CV 4	CV 5	Average
gini	3	0.675	0.679	0.6762	0.6717	0.6792	0.6762
entropy	3	0.675	0.679	0.6762	0.6717	0.6792	0.6762
gini	10	0.6752	0.679	0.6752	0.6717	0.677	0.6756
gini	5	0.6752	0.679	0.6752	0.6717	0.677	0.6756
gini	4	0.6752	0.679	0.6752	0.6717	0.677	0.6756
entropy	10	0.6752	0.679	0.6752	0.6717	0.677	0.6756
entropy	5	0.6752	0.679	0.6752	0.6717	0.677	0.6756
entropy	4	0.6752	0.679	0.6752	0.6717	0.677	0.6756
gini	2	0.6718	0.677	0.6742	0.6688	0.6765	0.6737
entropy	2	0.6718	0.677	0.6742	0.6688	0.6765	0.6737
gini	1	0.6723	0.6723	0.6723	0.6723	0.6722	0.6723
entropy	1	0.6723	0.6723	0.6723	0.6723	0.6722	0.6723

Algorithm	Depth	Model Negative	Model Positive	
gini	10	19553	616	**Target Negative**
		9098	733	**Target Positive**
gini	5	19553	616	**Target Negative**
		9098	733	**Target Positive**
gini	4	19553	616	**Target Negative**
		9098	733	**Target Positive**
gini	3	19553	616	**Target Negative**
		9098	733	**Target Positive**
entropy	10	19553	616	**Target Negative**
		9098	733	**Target Positive**
entropy	5	19553	616	**Target Negative**
		9098	733	**Target Positive**
entropy	4	19553	616	**Target Negative**
		9098	733	**Target Positive**
entropy	3	19553	616	**Target Negative**
		9098	733	**Target Positive**

Part 3 Continuous Feature Decision Tree

We can see that the top-level decision on our continuous feature tree is operating on Income (AGI_BIN). This supports the results in part 1 and 2.



Part 3 Discussion and Results

See the top table for cross-validation accuracies and the bottom table for the confusion matrices of our continuous feature decision trees. Our top model is 1.72% more accurate when using continuous features over binary features. This is small but significant. We see that the highest average cross-validation classification accuracy obtained was 69.34% for the entropy criterion at a depth of 4. Gini performed virtually the same at a depth of 4 obtaining an average cross-validation accuracy of 69.33%. As Dr. Jones states the simplest possible model is best. I think a depth of 4 allows the model to evaluate each feature once before making a prediction in any given path. Additional depth does not help the model discriminate better it appears. There was more difference in average cross-validation accuracy across parameters for our continuous tree, but the overall difference is still small. There was a 0.54% difference in the maximum and minimum model average accuracy for the continuous trees, and 0.06% for the binary trees (when only considering required depths 3, 4, 5 and 10). We conclude there are minor or insignificant changes across the given depth and information gain parameters for this dataset. Our continuous decision tree performs better than the binary decision tree, and perhaps this is because continuous features have better information gain. We are not constrained to ask questions about the mean of our distribution. Say if the top 20% of income earners have dividends 80% of the time, this would be a more meaningful way of splitting which can occur using continuous over binary data. Note the bias in our confusion matrices towards false negatives which we explore in greater detail in part 4.

Algorithm	Depth	CV 1	CV 2	CV 3	CV 4	CV 5	Average
entropy	4	0.6932	0.6945	0.6895	0.6948	0.6948	0.6934
gini	4	0.6928	0.6945	0.6895	0.6948	0.6948	0.6933
gini	3	0.6898	0.6930	0.6928	0.6917	0.6937	0.6922
entropy	3	0.6898	0.6930	0.6928	0.6917	0.6937	0.6922
entropy	5	0.6923	0.6918	0.6885	0.6928	0.6948	0.6921
gini	5	0.6923	0.6920	0.6885	0.6938	0.6932	0.6920
entropy	10	0.6910	0.6868	0.6902	0.6858	0.6910	0.6890
gini	10	0.6887	0.6878	0.6888	0.6865	0.6887	0.6881

Algorithm	Depth	Model Negative	Model Positive	
gini	10	18898	1081	**Target Negative**
		8204	1817	**Target Positive**
gini	5	18855	1124	**Target Negative**
		8059	1962	**Target Positive**
gini	4	19025	954	**Target Negative**
		8211	1810	**Target Positive**
gini	3	18624	1355	**Target Negative**
		7897	2124	**Target Positive**
entropy	10	18871	1108	**Target Negative**
		8146	1875	**Target Positive**
entropy	5	18843	1136	**Target Negative**
		8045	1976	**Target Positive**
entropy	4	19025	954	**Target Negative**
		8211	1810	**Target Positive**
entropy	3	18624	1355	**Target Negative**
		7897	2124	**Target Positive**

Part 4 Results

Random Forest Cross Validation Scores

Algorithm	Depth	Estimators	CV 1	CV 2	CV 3	CV 4	CV 5	Average
gini	5	10	0.6930	0.6960	0.6940	0.6958	0.6953	0.6948
gini	4	10	0.6930	0.6970	0.6937	0.6953	0.6947	0.6947
gini	5	100	0.6922	0.6960	0.6920	0.6975	0.6955	0.6946
entropy	5	1000	0.6935	0.6960	0.6908	0.6973	0.6950	0.6945
entropy	5	10	0.6930	0.6960	0.6915	0.6978	0.6938	0.6944
entropy	5	100	0.6927	0.6968	0.6915	0.6955	0.6952	0.6943
gini	4	1000	0.6925	0.6963	0.6908	0.6967	0.6952	0.6943
entropy	4	1000	0.6928	0.6963	0.6910	0.6963	0.6950	0.6943
gini	5	1000	0.6928	0.6962	0.6908	0.6973	0.6942	0.6943
gini	4	100	0.6932	0.6965	0.6908	0.6965	0.6942	0.6942
entropy	3	10	0.6910	0.6950	0.6933	0.6950	0.6963	0.6941
entropy	4	100	0.6930	0.6968	0.6903	0.6955	0.6947	0.6941
gini	3	100	0.6928	0.6963	0.6902	0.6955	0.6938	0.6937
entropy	3	1000	0.6920	0.6963	0.6907	0.6955	0.6935	0.6936
gini	3	1000	0.6918	0.6960	0.6905	0.6953	0.6938	0.6935
gini	3	10	0.6922	0.6963	0.6898	0.6945	0.6938	0.6933
entropy	10	1000	0.6895	0.6943	0.6918	0.6972	0.6938	0.6933
entropy	4	10	0.6917	0.6947	0.6905	0.6950	0.6943	0.6932
entropy	3	100	0.6922	0.6953	0.6898	0.6957	0.6928	0.6932
gini	10	100	0.6895	0.6948	0.6915	0.6958	0.6917	0.6927
entropy	10	100	0.6893	0.6958	0.6913	0.6932	0.6932	0.6926
gini	10	1000	0.6887	0.6940	0.6908	0.6950	0.6923	0.6922
entropy	10	10	0.6902	0.6917	0.6883	0.6933	0.6917	0.6910
gini	10	10	0.6887	0.6908	0.6898	0.6925	0.6930	0.6910

Random Forest Confusion Matrices

Algorithm	Depth	Estimators	Model Negative	Model Positive	
gini	10	10	18813	1166	**Target Negative**
			8018	2003	**Target Positive**
gini	5	10	19020	959	**Target Negative**
			8182	1839	**Target Positive**
gini	4	10	19002	977	**Target Negative**
			8204	1817	**Target Positive**
gini	3	10	19165	814	**Target Negative**
			8385	1636	**Target Positive**
gini	10	100	18866	1113	**Target Negative**
			8060	1961	**Target Positive**
gini	5	100	19030	949	**Target Negative**
			8183	1838	**Target Positive**
gini	4	100	19012	967	**Target Negative**
			8180	1841	**Target Positive**
gini	3	100	18992	987	**Target Negative**
			8211	1810	**Target Positive**
gini	10	1000	18879	1100	**Target Negative**
			8055	1966	**Target Positive**
gini	5	1000	19075	904	**Target Negative**
			8226	1795	**Target Positive**
gini	4	1000	19043	936	**Target Negative**
			8214	1807	**Target Positive**
gini	3	1000	19019	960	**Target Negative**
			8225	1796	**Target Positive**
entropy	10	10	18818	1161	**Target Negative**
			8019	2002	**Target Positive**
entropy	5	10	18931	1048	**Target Negative**
			8106	1915	**Target Positive**
entropy	4	10	19009	970	**Target Negative**
			8171	1850	**Target Positive**
entropy	3	10	19305	674	**Target Negative**
			8540	1481	**Target Positive**
entropy	10	100	18884	1095	**Target Negative**
			8057	1964	**Target Positive**
entropy	5	100	19075	904	**Target Negative**
			8251	1770	**Target Positive**
entropy	4	100	19059	920	**Target Negative**
			8242	1779	**Target Positive**
entropy	3	100	18983	996	**Target Negative**
			8197	1824	**Target Positive**
entropy	10	1000	18882	1097	**Target Negative**
			8042	1979	**Target Positive**
entropy	5	1000	19049	930	**Target Negative**
			8204	1817	**Target Positive**
entropy	4	1000	19006	973	**Target Negative**
			8190	1831	**Target Positive**
entropy	3	1000	18994	985	**Target Negative**
			8203	1818	**Target Positive**

Part 4 Discussion

The highest accuracy we achieved using random forest models is 69.48%. This is 0.14% higher than our continuous decision tree models' maximum average cross-validation accuracy. We iterate through depths of 3, 4, 5, and 10 and information gain algorithms of gini and entropy. We also now iterate through n_estimators values of 10, 100, and 1000. N_estimators is the number of decision trees within the random forest. We attempted to compute random forest models with 10,000 estimators but they took more than an hour of runtime to compute and so we cut this parameter argument. Surprisingly, the random forest model with the best performance had only 10 estimators, a depth of 5, and used gini information gain. Again, simple models appear to perform quite well. Bootstrapping was left to its default value of true when creating our results. When experimenting we discovered turning bootstrapping off led to lower accuracy for our random forest models. We see a bias across all models towards false negatives from our confusion matrices. Perhaps the models see someone of a particular demographic, they have a high income, high age, and work frequently. The model then infers they should have a high chance of receiving stock dividends. When, many people who have high income, busy careers, and are advancing in years fail to invest in their own retirement. An economic phenomena I have read about states that as one's income grows inexorably their spending grows. The data present appears to not allow us to peer far enough to see if an individual has good financial habits! Another alternative explanation may be the proportion of investors portfolio's that yield dividends is statistically small, I know I don't go out of my way to purchase dividend yielding stocks, I usually go for index funds. Perhaps our census data only counts dividends received as income. I believe my dividends are automatically re-invested and I would have no resulting tax burden. It is an interesting problem.

References

When writing my decision trees to png file as a graph I referenced Dr. Creed Jones supplied code.

```
# call this function like this:
    writegraphtofile(clf, dataset.features, ("0", "1"), dataset.basedir+"conttree.png")

def writegraphtofile(clf, featurenames, classnames, pathname):
    dot_data = tree.export_graphviz(clf, out_file=None,
                                    feature_names=featurenames, impurity=True,
                                    class_names=classnames, filled=True,
                                    rounded=True, special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data)
    colors = ('lightblue', 'lightgreen')
    edges = collections.defaultdict(list)
    for edge in graph.get_edge_list():
        edges[edge.get_source()].append(int(edge.get_destination()))
    for edge in edges:
        edges[edge].sort()
        for i in range(2):
            dest = graph.get_node(str(edges[edge][i]))[0]
            dest.set_fillcolor(colors[i])
    graph.write_png(pathname)
```