CA03 – Decision Tree Algorithm

1. Data Source and Contents

The dataset is obtained from the Census Bureau and represents salaries of people along with seven demographic variables. The following is a description of our dataset:

- **Number of target classes:** 2 ('>50K' and '<=50K') [Labels: 1, 0]
- **Number of attributes (Columns):** 7
- **Number of instances (Rows):** 48,842

Data Source:

Use the following exact "path" in your code as the data source:

"https://github.com/ArinB/MSBA-CA-03-Decision-Trees/blob/master/census_data.csv?raw=true"

Training and Test Data: There is a column indicating the rows to be used as "Training Data" and "Testing Data". You can programmatically create your Training and Testing datasets as separate dataframes in your code based on this column value.

Hint: You should separate out the Training and Testing data AFTER you have completed your Data Quality Report, data cleaning, transformations, and feature selections (if any).

Note that the "continuous" data columns have been "transformed" into certain "data groups" or "data blocks". This technique is called "binning" and by this process you can transform "continuous" data to "discrete categories", because for certain applications like this "discrete categories" makes more sense that the exact value in the continuous scale. It's also called "discretization" of data. **Investigate the data and answer the following question in your assignment** submission. (Probably, you will be able to answer these questions better AFTER you have completed the rest parts of the assignment.)

Q.1 Why does it makes sense to discretize columns for this problem?

Q.2 What might be the issues (if any) if we DID NOT discretize the columns.

2. Data Quality Analysis (DQA)

Do all of these inside your Notebook:

- Perform a Data Quality Analysis to find missing values, outliers, NaNs etc.
- Display descriptive statistics of each column
- Create a Data Quality Report
- Perform necessary data cleansing and transformation based on your observations from the data quality analysis

3. Build Decision Tree Classifier Models

Definition: Given a data of attributes together with its classes, a decision tree produces a sequence of rules that can be used to classify the data.

Advantages: Decision Tree is simple to understand and visualise, requires little data preparation, and can handle both numerical and categorical data.

Disadvantages: Decision tree can create complex trees that do not generalize well, and decision trees can be unstable because small variations in the data might result in a completely different tree being generated.

Use "DecisionTreeClassifier" algorithm from scikit learn. Find details of sklearn tree algorithm below. Scitkit Learn implements an optimized version of CART algorithm and can be used for binary class as well as multi-class classifications. It can be used for classification, as well as regression. Study the following link thoroughly, including Section 1.10.5 (Tips on Practical Use).

https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart

Syntax to use the classifier is shown below:

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(max_depth=10, random_state=101,
                               max_features = None, min_samples_leaf = 15)
dtree.fit(x_train, y_train)
y_pred=dtree.predict(x_test)
```

random_state: This is a number you choose arbitrarily. It's also called "Random Seed". If you use a number for this parameter (any number), it ensures that if you run the program multiple times, it will generate the same randomness. Hence, the solution becomes more "reproduceable".

4. Evaluate Decision Tree Performance

Calculate and display the following. Do all of these inside your Notebook.

- Confusion Matrix (TP, TN, FP, FN … etc.)
- Accuracy, Precision, Recall, F1 Score

5. Tune Decision Tree Performance

Learn about all hyper-parameters and methods of Scikit Learn DecisionTreeClassifier algorithm at:

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

Try varying FOUR of the hyperparameters manually, as per the following table, and train / score your model for each set of these hyperparameters. Record your Tree's performance with respect to each of these sets of hyperparameters in the Model Performance section of the following table.

Four Hyperparameters to vary:

- Split Criteria – 'Entropy' or 'Gini Impurity'
- Maximum Features – The number of features to consider when looking for the best split. If float, then max_features is a fraction and max(1, int(max_features * n_features_in_)) features are considered at each split.
- Minimum Sample Leaf – Minimum of samples in a leaf node to stop further splitting (becomes a leaf node)
- Maximum Depth – Maximum depth of the tree allowed

Q.3 Decision Tree Hyper-parameter variation vs. performance

Here your goal is to determine the hyper-parameter values for the "best-performing" tree with respect to "accuracy"

You will vary FOUR hyperparameters, one at a time, as in the following table:

Hyperparameter Variations			
Runs in Loop			
Run 1	**Run 2**	**Run 3**	**Run 4**
Split Criteria (Entropy or Gini)	Minimum Sample Leaf	Maximum Feature	Maximum Depth
Entropy	*5*	*auto'*	*2*
Gini Impurity	*10*	*None*	*4*
	15	*0.3*	*6*
	20	*0.4*	*8*
	25	*0.5*	*10*
	30	*0.6*	*12*
	35	*0.7*	*14*
	40	*0.8*	*16*

For Run 1, vary Run 1 columns only and keep other columns to "default". For Run 2, use the BEST hyper-parameter value of Split Criteria from Run 1, For Run 3 - use the BEST hyper-parameter values of Split Criteria and Minimum Sample Leaf from Run 1&2, For Run 4 use the Best values of Split Crteria, Minimum Sample Leaf, Maximum Feature from previous runs. Judge the BEST hyper-parameter values with respect to "Accuracy".

For each run, your will "fill up" a table with all performance parameters as follows. So, there will be 4 tables like this for 4 runs, based on the number of rows of the hyper-parameters. For the Split Criteria, obviously you will have two rows only.

Maximum Depth	Accuracy	Recall	Precision	F1 Score
2				
4				
6				
8				
10				
12				
14				
16				

To determine the BEST hyper-parameter value, create line-graphs with the hyper-parameter as x-axis and accuracy as y-axis. A sample is provided below for a graph of Minimum Sample Leaf Vs. Accuracy. You don't need graph Split Criteria Vs. Accuracy. So you will have 3 graphs altogether.

You can use the model training and testing in a loop for each of the hyper-parameters as the code sample below. You can modify the code to capture each of the performance parameters and display them in the table format inside your notebook like the above figure.

```
results = []
max_depth_options = [2,4,6,8,10,12,14,16,18,20]
for trees in max_depth_options:
    model = DecisionTreeClassifier(max_depth=trees, random_state=101)
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    accuracy = np.mean(y_test==y_pred)
    results.append(accuracy)
```

6. Visualize Your Best Decision Tree using GraphViz

Get the detail of how to do this from the following link:

https://medium.com/@rnbrown/creating-and-visualizing-decision-trees-with-python-f8e8fa394176

Once you have the best hyper-parameters from 4 runs, you will know the hyper-parameter combination of your BEST-performing Tree with respect to "accuracy". Use these set of hyper-parameters to build the BEST Tree, record it's performance parameter (all of them) and then "visualize" this best tree.

Hint: Make sure you have the "graphviz" package latest version compatible with your Python version installed. You are free to use any other Visualization package as well!

7. Conclusion

Explain your observations from the above performance tuning effort.

Q.4 How long was your total run time to train the best model?

Q.5 Did you find the BEST TREE?

Q.6 Write your observations from the visualization of the best tree

Q.7 Will this Tree "overfit"? (Hint: Is this tree "fully grown")

8. Prediction using your "trained" Decision Tree Model

Based on the Performance Tuning effort in the previous section, pick your BEST PERFORMING TREE. Now make prediction of a "new" individual's Income Category (<=50K, or >50K) with the following information. Do this in your Notebook.

- Hours Worked per Week = 48
- Occupation Category = Mid - Low
- Marriage Status & Relationships = High
- Capital Gain = Yes
- Race-Sex Group = Mid
- Number of Years of Education = 12
- Education Category = High
- Work Class = Income
- Age = 58

Hint: You have to create a single record dataframe with the above values with the EXACT same dataframe structure you have used for "training". Then use the "model.predict()" object using this single-record dataframe. Also, note that you need to do exact same "binning" and "encoding" of the values above as per the original data file before you can use your dataframe for prediction.

Q.8 What is the probability that your prediction for this person is correct?

Hint: Research and find out "probability score" variable outputted when you use the "model.predict()" object.

9. Deliverables

Your assignment outputs will have the following components:

(1) Fully functional Notebook, Data, Readme file in a single folder at GitHub (this is for sharing with your peer for peer review)
(2) Upload your fully executed notebook at BrightSpace
(3) All questions should be answered with the corresponding Question Numbers in Marked-down Cells in the Notebook.