CA04 – Ensemble Models

1. Data Source and Contents

Same as CA03. However, being repeated here.

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

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.

You do not need to do DQA, as you had already done so in CA03 for the same data. However, you need to do the same data cleaning and transformations here again to be able to run the ML Model. You can re-use the code from CA03.

2. Finding Optimal Value of a key Ensemble Method Hyper-parameter

For Ensemble Models, one of the key hyper-parameter is number of "estimators". You are required to find its best value by creating the following line graphs:

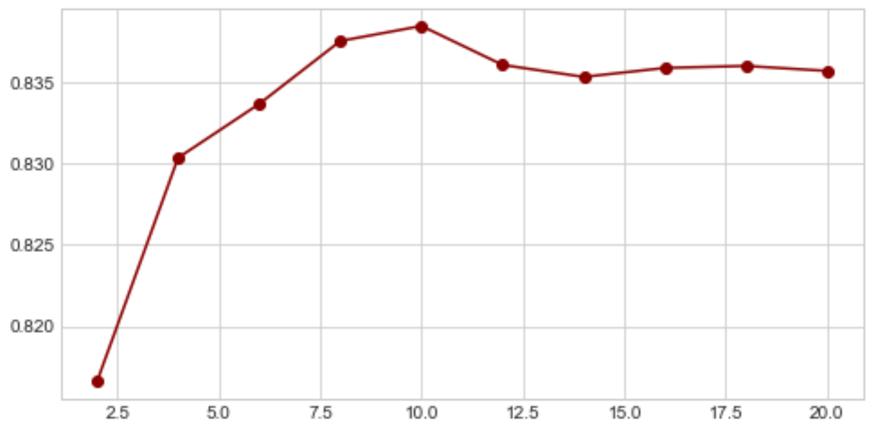
- Accuracy Vs. n_estimators
- AUC Vs. n_estimators

Following is an example code of finding the optimal value of "Accuracy" Vs. "Maximum Depth" for DecisionTreeClassifier algorithm. Review the following code snippet to understand how the optimal value of this hyper-parameter (Max Depth = 10) is found by plotting a graph.

```
In [17]: 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)

         plt.figure(figsize=(8,4))
         pd.Series(results, max_depth_options).plot(color="darkred",marker="o")

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x20703c75630>
```



3. Building a Random Forest Model

Using Notebook, and the same data source from CA03, train a Random Forest Model. Using similar approach of Section 2 above, plot a graph of Accuracy vs. n_estimator and AUC Vs. n_estimator. Use n_estimator values as [50,100,150,200,250,300,350,400,450,500]. Keep all other hyperparameter values at default.

Answer the following questions for Random Forest model and each algorithm in 4:

1. Write your observations about the Classifier's behavior with respect to the number of estimators
2. Is there an optimal value of the estimator within the given range?

4. Building AdaBoost, Gradient Boost, and XGB.

Repeat the process of Section 3 above for AdaBoost, Gradient Boost, and XGB Classifiers.

5. Compare Performance

For the best values of Accuracy and AUC for four models (Random Forest, AdaBoost, Gradient Boost, XGB) in the previous steps, fill up the following table:

	Random Forest	AdaBoost	Gradient Boost	XGB
Accuracy				
AUC				

6. Deliverables

Your assignment outputs will have the following components:

(1) Fully functional Notebook, Data, Readme file in a single folder at GitHub
(2) Fully functional and fully executed Notebook at BrightSpace

*** Useful Material ***

Here is a great list of all Model Evaluation Functions in SKlearn library grouped by the Model Type. You can keep this as a "cheat sheet" for your future project / assignment use.

https://scikit-learn.org/stable/modules/model_evaluation.html