Winter Quarter 2019 – UCSB Physics 129L
Homework 9 – not for credit

This optional exercise requires you to figure out quite a few things for yourself. A solution (note necessarily **the** solution) is available on the website.
Warning: I had some difficulties making my code work on the rpi due to the obsolete software installation. The first problem was that I could not read the "pickle" data file that had been prepared on a different machine. I fixed that by also providing an equivalent csv file, as discussed below. I then found an incompatibility between libraries that you may or may not also encounter, depending on what method you use to tackle the problem. The error message that I encountered and that I could not get around was:
`ValueError: the 'dtype' parameter is not supported in the pandas implementation of sum()`
The bottom line is that if you have a better installation of python on your own computer you should use it instead of the rpi. I am sorry about that, but this is real life!

The goal of the exercise is to take some features of a set of preselected simulated events from LHC collisions and use a multivariate machine learning algorithm to separate "signal" events from "background" events. The particular signal that you are looking for is not important for this exercise – you can ask me separately about that if you are interested. The starting point is either
`/home/pi/physrpi/campagnari/python/data_fourtop.pkl` or
`/home/pi/physrpi/campagnari/python/data_fourtop.csv`.

These are equivalent pickle or csv files containing the same pandas data frame. On my MAC I can read both, but on the rpi I can only read the csv file. Figure 1 is a screenshot of a jupyter notebook session that shows you how to read the file and also shows you what the data frame (partially) looks like.

You can think of a pandas data frame as sort of an excel spreadsheet. Each row in the dataframe corresponds to one simulated pre-selected event, and each event (or row) is characterized by a number of variables. The first column ("signal") tells you whether the event is signal (=1) or background (=0). The second column ("weight") is the relative weight of each event, i.e., the relative probability of the event (a) being produced in a proton-proton collision, and (b) passing the pre-selection requirements of a (very!) realistic

In [2]:
```
# read the pickle file or the csv file
# df = pd.read_pickle("data_fourtop.pkl")
df_bad = pd.read_csv("data_fourtop.csv")
# if we are reading csv, we need to drop one column
df = df_bad.drop(['Unnamed: 0'], axis=1)
df
```

Out[2]:

	signal	weight	htb	nbtags	njets	nleps	ptj1	ptj7	ptl1	ptl2	ptl3	q1
0	1.0	0.000364	206.779850	2.0	7.0	2.0	373.429020	46.658970	137.676560	67.604260	-1.000000	-1.0
1	0.0	0.004883	35.586070	1.0	3.0	2.0	188.072630	0.000000	121.428140	53.178535	-1.000000	-1.0
2	0.0	0.003294	489.509370	2.0	4.0	3.0	408.042100	0.000000	208.297500	41.227450	156.992800	-1.0
3	0.0	0.004319	85.794560	1.0	3.0	2.0	276.633600	0.000000	87.203590	49.023970	-1.000000	-1.0
4	0.0	0.002928	296.612200	2.0	4.0	2.0	242.500760	0.000000	123.092630	41.000244	-1.000000	1.0
5	0.0	0.001323	96.686806	1.0	3.0	2.0	163.566330	0.000000	34.729990	26.165741	-1.000000	1.0

Figure 1: Screen shot of a jupyter notebook session for reading the pandas data frame.

LHC data analysis. The other ten columns give measured characteristics of the event. Again, what they really represent is not so important, but I can tell you if you are interested.

You should plot histograms of these ten quantities for both signal and background, keeping track of the weights. You will find that signal and background are a little different in every variable. Some variables are better than other at distinguishing signal from background.

We will next combine these ten variables into a single variable that maximizes the separation between signal and background. You can do this by trying different classification algorithms, with different settings. There are several such algorithms available. Some examples are here:
https://tinyurl.com/y69ot733
but there are more on the market. The solution that I posted is based on a Boosted Decision Tree using the `AdaBoost` algorithm.

Some general notes

- Classifiers have several knobs you can turn. Experiment until you find something you like.
- You should divide your sample into a training and testing sample. The training sample should be used to train the classifier. The testing sample should be used to test how well the separation really works. This is because the classifier may be 'tuning' itself on statistical fluctuations, so it is important to test on a statistically indepenedent sample.
- To see how well you are doing, you should construct a "receiver oper-

ating curve" (ROC) and calculate the "area under the curve" (AUC). See https://tinyurl.com/y8d8l9uz.

- If you google around a little bit you will find python functions that calculate and plot ROCs and AUCs with minimal work on your part. Or you can look at my solution and copy it...

Good luck and have fun. Come see me and show me what you have, and let's see if you can do better than I did. Figures 2, 3, and 4, summarize my results

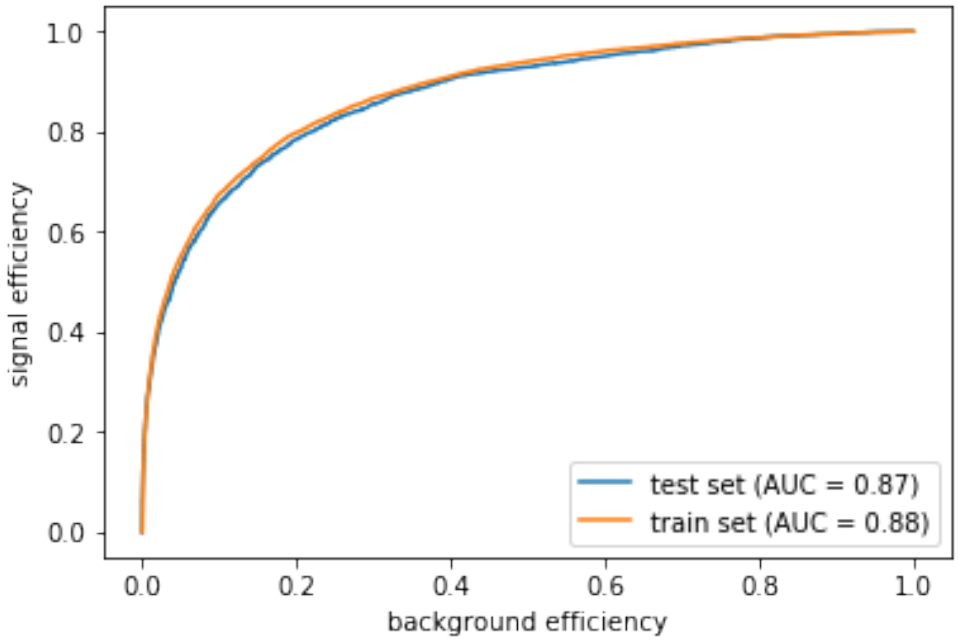

Figure 2: ROC curve and AUC.

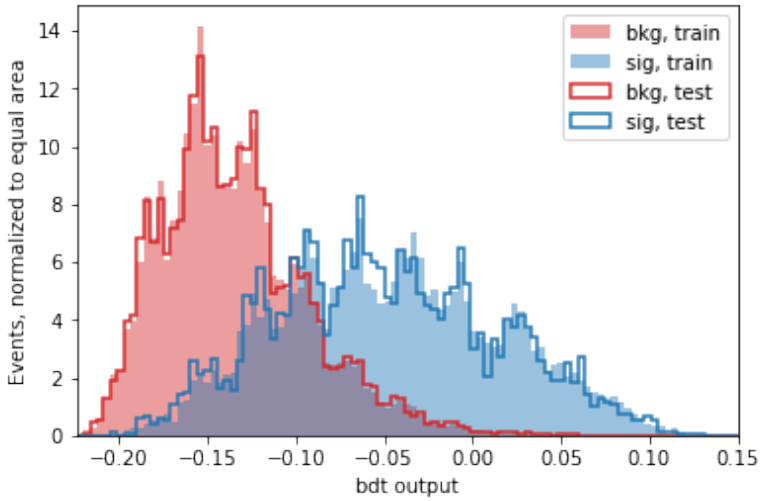

Figure 3: BDT output distributions for signal and background, normalized to the same area.

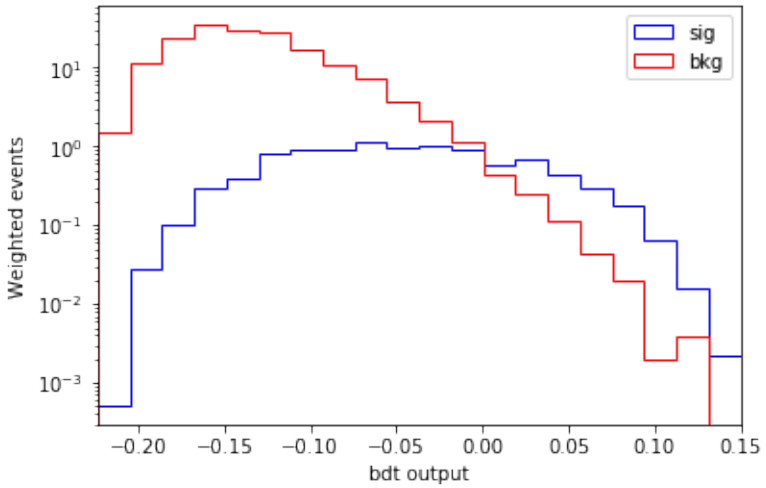

Figure 4: BDT output distributions for signal and background, normalized by the weights.