**CPSC 330 Midterm 2019W2 Crowd-Sourcing Solutions**

***Exam:***

[**https://github.com/UBC-CS/cpsc330/blob/master/exams/2019W2/midterm.ipynb**](https://github.com/UBC-CS/cpsc330/blob/master/exams/2019W2/midterm.ipynb)

***Rules:***

* Respect the others
* Write down your answer to the questions or make comments on the others’ answer
* Never delete the others’ answers
* Try to use a different color for your own answer

***Q1***

- The only thing i can think of, especially because it says "what did i do wrong in the above line?" is that they should have done df\_train.describe() instead of df.describe(). I notice that we've done that instead in our homework and lectures. (From Piazza: <https://piazza.com/class/ky0j51i4ud64t5?cid=128>)

I think it might be you have to use df\_train.describe() instead because, you would be breaking the golden rule by doing df.describe since you would be looking at the testing data.

You can definitely look at testing data if you want without breaking the golden rule. We look at all data when we decide how to preprocess it.

Facts mb

Though thinking about it some more maybe we’ve already done preprocessing at this point? So I do think you have a point.

Maybe we could say that we skipped the categorical data? Include = all would include the categorical features as well. The default option only shows Numeric Columns.

That’s a good point

**Are we meant to assume them splitting train/validate/test datasets based on year was correct because they said it was reasonable? They’ve said explicitly you need to split randomly**

***Q2***

- There would be no difference here because there are only two "categories", yes and no, OrdinalEncoder would try to give those two an inherent order of 0 and 1; ultimately producing the same result as OneHotEncoder, which also produced a column with 0 and 1.

correction: with drop='first" the OHE would drop one of the columns, but since this feature is binary it should not make a difference

It should make a different column name. That’s about it.

I think it’s different because for OrdinalEncoder, the user can specify what is 1 and what is 0, while it’s arbitrary for OHE. Aren’t we dropping the first column so we now wich one OHE is going to keep I think

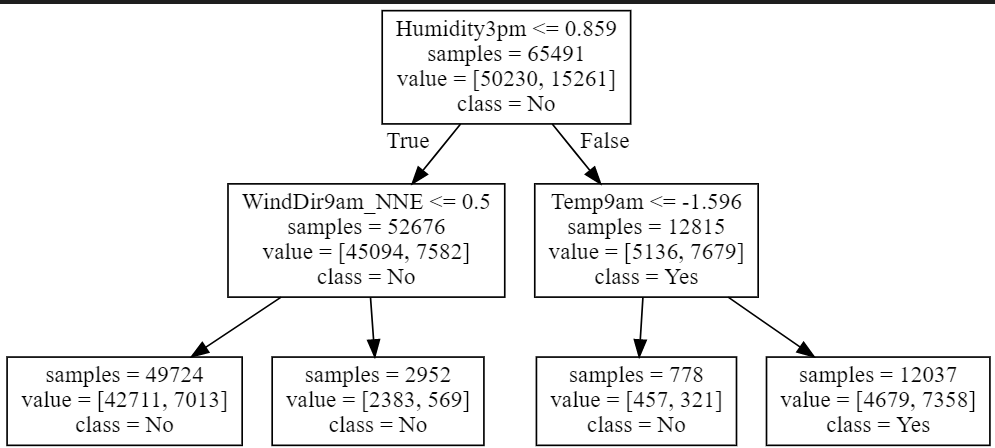
***Q3***

- The default scoring for DummyClassifier is accuracy. We see a very high accuracy score of about 80% on both the train and validation sets. This suggests that there is about a 80/20 class imbalance in our data set. If I had to guess, It’s predicting the most common target, which is NO rain tomorrow and getting it right about 80% of the time because it only rains on 20% of the days.

What can we say about the validation score being almost similar to the training score? Just that the model is underfitting since there is no gap between the two scores? - Yes I think so, and also the distribution is not that biased

***Q4***

*I see the image, it should go to the 1st grid: Humidity3pm <= 0.859 so we go to the left, WindDir9am is SE for our sample which means WindDir9am == 0 so we go to the left again, landing in the first leaf*

**

*The Image won’t load on GitHub, clone the repo, it should appear in your IDE.*

Can someone explain to me Nhow this came to be: ***WindDir9am is SE for our sample which means WindDir9am == 0***

^ replying to purple: WindDir9am\_NNE in one-hot-encoding essentially means [1 if WindDir9am == NNE, 0 otherwise]. Since the sample we have has WindDir9am = SE, its column of Winder9am\_NNE must be 0.

Thank you that finally makes sense!

***Q5***

- From the strong graph we can tell its most likely Histogram 2 due to the clear separation between the Yes and No classes and the clear split at 0.859.

***Q6***

- I would not change C. Our model has a good level of complexity and is doing the best it can. It is not over fit as the train score is not much larger than the validation score and its not underfoot as the error in both scores is quite low. Therefore, we are near the sweet spot.

* I think that c could be increased as the train score is still less than the validation but once it crosses then it is overfitted, I would try a couple more values of C to see if it starts to overfit and pick the largest C when the train’s accuracy is greater than the other accuracy.

***^ why not increase C***

***+1***

***+1***

***+ 3***

***increase++***

I would increase C. The difference between the train and validation scores is pretty low; we may be underfitting the data. Try a few more values, stop when it starts overfitting.

Max iterations is also really low (relative to some other examples we’ve done), so I think it’s definitely possible that we’re underfitting

***Q7***

- True. Evaporation has a positive coefficient and is therefore contributing positively to the model. Meaning the higher the evaporation value and more likely it is for the sample to be predicted as Yes.

***Q8***

- False. WNW has a higher coefficient of 0.15 vs 0.03 of W and is therefore contributing more.

***Q9***

- I believe the 8 would decrease as the model would weigh the minority class of Yes rain tomorrow more and predict more yeses, therefore naturally, some of that 8 would shift to the right (yes prediction).

In this dataset, “Yes” is the minority class; therefore by using class\_weight=’balanced’ we will increase “Yes”’s importance. An increased importance implies our model is more likely to predict “yes” - therefore the bottom left will decrease.

***Q10***

- Looking at the FP and FN on both the train and validation confusion matrix we can see a huge jump in number on the validation set. This is a clear example of over-fitting. The error is much larger in the latter than the former.

We could also mention that recall and precision dropped significantly.

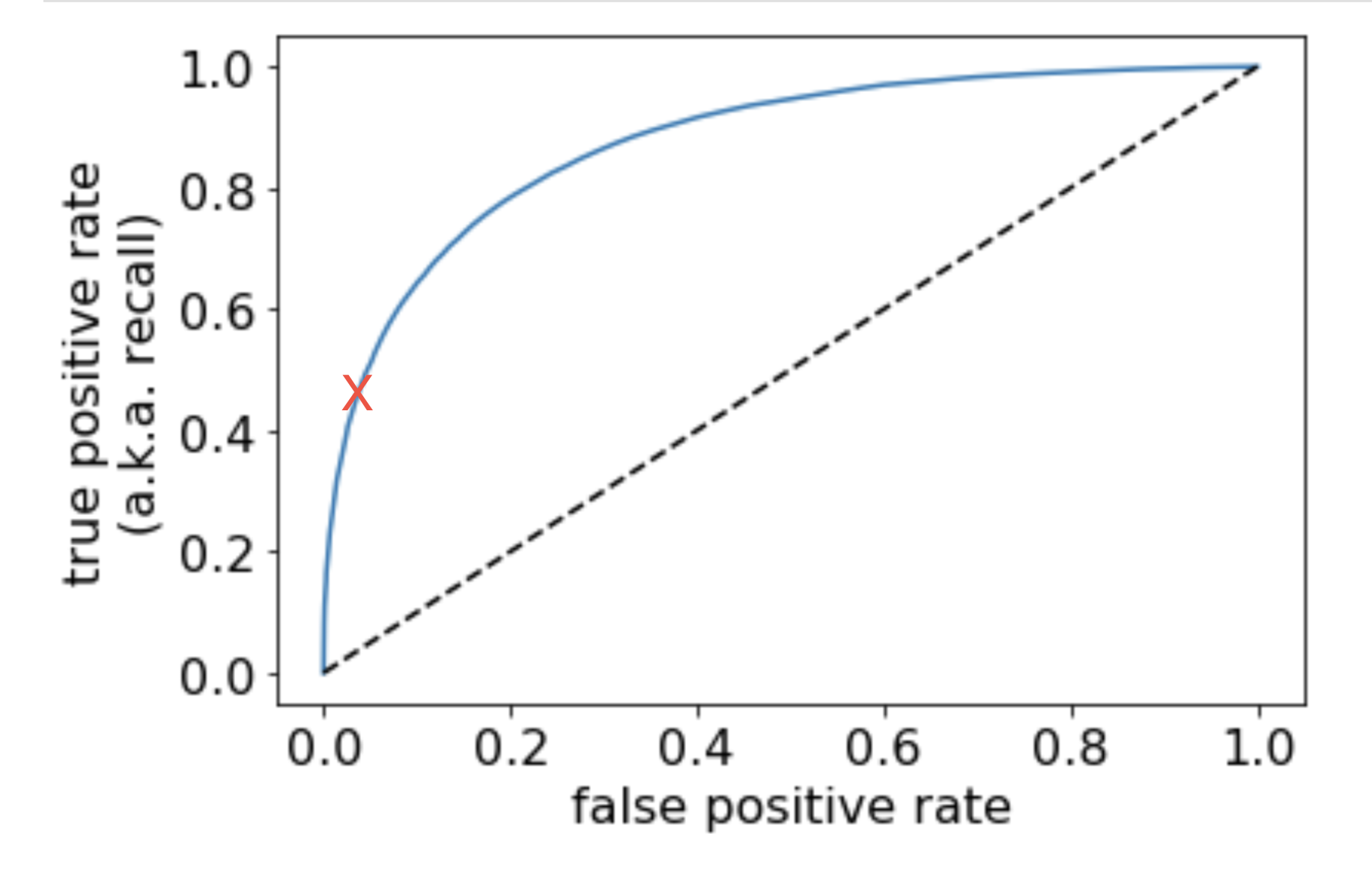
Technically those are the same thing (more false negatives and false positives imply recall and precision dropped) but it’s probably good to use those terms

Unsure about this one - the latter example is on the validation set, which we would expect to see a drop in the accuracy compared to the training set. I agree that having a low rate of FP/FN in the training set implies an overfit, but I am not sure about what we can say about the validation that would indicate an overfit.

***Q11***

- By calculating the TPR (5021/(5021+5649)= 0.4705716963

We marked this as the 0.5 threshold which is the default used by the confusion matrix above.

******

***~~I don't think the above is correct, TPR = 5021/(5021 + 1573) = 0.76144~~***

*This formula you used from the lecture has a typo. Green is correct*

Could you explain why you calculated the TPR? +1

I think i get it, the TPR rate is for when the threshold is 0.5 (default val in the confusion matrix). So, by calculating the TPR, we can tell where the threshold (0.5) is located.

But in the lecture example, wasnt the 0.5 threshold located at TPR 0.6? (lec 9 page 15). ooo sry i get it now thanks!

It’s different for each data set.

https://stackoverflow.com/questions/32627926/scikit-changing-the-threshold-to-create-multiple-confusion-matrixes

“the default threshold is at 50%”

***Q12***

Don’t really understand. But I think histogram 1 is LR.

Hist 1: lr

Hist 2: rf

Hist 3: dc

I think Hist 1 is LR because Hist 2 is too perfect in separation b/w “Yes” and “No” (score for LR should be way higher if it is Hist 2), and Hist 3 is a crazy distribution.

**My answer below is incorrect;**

**ran the file and the answers were**

**Histogram 1 LR**

**Histogram 2 RF**

**HIstogram 3 DC**

**(see Red’s explanation below)**

**-**

Histogram 3 is the dummy classifier; it’s apparent because of the distribution - all share the same probability.

Histogram 2’s distribution suggests the model is making predictions based on the given probability. All examples with less than 0.5 predicted probability are classified as “No” and all examples with more than 0.5 predicted probability are classified as “Yes”. This is logistic regression. **BUT I’M NOT SURE**

^ Random forest is essentially multiple decision trees.

^ And each tree works with only a random set of features, and has different slightly different data (bootstrapping).

^ That makes sense to me, but I’m still confused because logistic regression models make their decision based on the threshold right? Like if it’s less than 0.5, then it will classify as the negative class and vice versa.

I’m going to try running it; one sec!

I think what is being plotted are the probabilities of each of the categories being labeled for all the data points. So, better classifiers have a better split. Since rf has the highest score, it is the best and hence predicts the correct probability the most amount of times. Hence the clean split.

If an incorrect prediction is made, the model.predict\_proba(X\_train\_neg)[:,1][index] will return a lower probability for the correct output. Hence there is overlap in the models besides random forests.

^^ You were totally right. I just ran the code. Thanks for the explanation!

I think more separated histograms imply more accuracy, so I believe histogram 3 is the dummy, histogram 1 is the linear regressor and histogram 2 is the random forest

***Q13***

*2\*5\*2\*5 = 100 Models*

2 for n\_estimators, five for max\_depth, two for max\_features, five for cross\_validation?

Yes

👍

***Q14***

- Using a small training set and optimizing the random\_set has created a false high accuracy score. You are only scoring well due to limited samples in the test set (using only 100) and by finding a randomization order that happens to score well. You should instead use the full train set and optimize the real hyper-parameters instead to get a better model. (optimization bias)