MATH 390.4 / 650.2 Spring 2018 Homework #4t

Professor Adam Kapelner

Tuesday 8th May, 2018

Problem 1

These are questions about Silver's book, chapters ... For all parts in this question, answer using notation from class (i.e. $t, f, g, h^*, \delta, \epsilon, e, t, z_1, \ldots, z_t, \mathbb{D}, \mathcal{H}, \mathcal{A}, \mathcal{X}, \mathcal{Y}, X, y, n, p, x_{\cdot 1}, \ldots, x_{\cdot p}$, $x_{1\cdot}, \ldots, x_{n\cdot}$, etc. and also we now have $f_{pr}, h^*_{pr}, g_{pr}, p_{th}$, etc from probabilistic classification as well as different types of validation schemes).

(a) [easy] What algorithm that we studied in class is PECOTA most similar to?

PECOTA is most similar to the k-nearest neighbor algorithm we learned in class. PECOTA works by using baseball players various statistical attributes to cluster players that are more similar. The power of PECOTA lies in the fact that it is able to predict the performance of early career players by finding the player in the training set that is most similar to the player.

(b) [easy] Is baseball performance as a function of age a linear model? Discuss.

No. One of the points that Nate Silver discusses is that in analyzing the data he used to build PECOTA, he noticed that a player's overall performance peaked between ages 25-30, so the function would better resemble a concave parabola than a straight line

(c) [harder] How can baseball scouts do better than a prediction system like PECOTA?

Baseball scouts have a larger selection of Xs. They have better access to early career players and are able to use non-quantifiable characteristics of the player's performance. For example, one of the things Nate Silver says scouts like to look for is how disciplined the player is during training. For example, Pedroia, who was forecasted to be a good performer, tended to be very focused during practices, to the point of blowing off Silver's questions because he didn't want to lose focus on the game. Scouts also can measure more detailed data like the player's speed, agility, and speed of their pitches which is information not available to PECOTA.

(d) [harder] Why hasn't anyone (at the time of the writing of Silver's book) taken advantage of Pitch f/x data to predict future success?

There is not yet still enough training data from Pitch f/x in order to build an accurate model. Pitch f/x systems started being used around 2006, whereas baseball as a sport

is more than a hundred years old. There is much more data available on player's other statistics, such as height, weight, batting averages, home runs, and other variables going back decades than there is Pitch f/x data. As more Pitch f/x dataset increases, it is possible that more accurate models can be built using it.

(e) [difficult] Chapter 4 is all about predicting weather. Broadly speaking, what is the problem with weather predictions? Make sure you use the framework and notation from class. This is not an easy question and we will discuss in class. Do your best.

"Chaos Theory" is a major underlying cause limiting the accuracy of weather models. Minute changes in the starting state of the models can amplify and cause drastic changes in the final results. Because of this, forecasters have to be very careful in how they process their data as minute rounding errors can produce different results, in addition even small bugs in the program can cause large divergences in the results

(f) [easy] Why does the weatherman lie about the chance of rain? And where should you go if you want honest forecasts?

Weathermen lie about the chance of rain, because the audience get upset when it rains when forecasts call for no precipitation, which might result in loss of viewership. False negatives are more expensive than false positives (people are not as upset if it didnâĂŹt rain when forecasts said it would) therefore in order to minimize their risk, TV forecasters tend to adjust their models to err on the side of false positive rather than false negatives. Institutions like the national weather service are more interested in more empirical, non-consumer oriented weather prediction, so they are a better source of accurate weather forecasts.

(g) [difficult] Chapter 5 is all about predicting earthquakes. Broadly speaking, what is the problem with earthquake predictions? It is *not* the same as the problem of predicting weather. Read page 162 a few times. Make sure you use the framework and notation from class.

Unlike weather, what causes earthquakes is still a mystery. This is because earthquakes happen deep underground away from scientist's prying eyes, making them harder to study. The error due to ignorance is very large because we still don't have a good set of X's that explain earthquakes.

(h) [easy] Silver has quite a whimsical explanation of overfitting on page 163 but it is really educational! What is the nonsense predictor in the model he describes?

Characteristic fit, nonsense predictor pg 163

(i) [easy] John von Neumann was credited with saying that "with four parameters I can fit an elephant and with five I can make him wiggle his trunk". What did he mean by that and what is the message to you, the budding data scientist?

It is a message about overfitting. Even though using more predictors can make more detailed fits, and with 5 variables we can fit the elephant down to the shape of his

trunk, the next elephant we look at might be wriggling their trunk differently. Is how an elephant wriggles their trunk part of what makes an elephant an elephant anyway? It is likely that the essence of an elephant can be described in other terms and fitting the elephant's wriggle would be like fitting the noise in the data.

(j) [difficult] Chapter 6 is all about predicting unemployment, an index of macroeconomic performance of a country. Broadly speaking, what is the problem with unemployment predictions? It is *not* the same as the problem of predicting weather or earthquakes. Make sure you use the framework and notation from class.

There is inherent economic biases in predicting the economy. Economists are pressured in making definitive point-predictions and probabilistic predictions are few and far-between.

(k) [E.C.] Many times in this chapter Silver says something on the order of "you need to have theories about how things function in order to make good predictions." Do you agree? Discuss.

I agree. Being familiar with the system you are trying to model will allow you to build better models. A person with domain knowledge about the system they are trying to model will have better knowledge about the underlying causes that govern the system, and will therefore be better able to be selective and pick appropiate set of Xs. A layman with little or no domain knowledge might either miss one crucial predictor that would be immediately be immediately recognized by an expert, or be nondescriminate and incorporate many Xs into their model and therefore be in danger of overfitting, fitting the noise, and mistaking correlation where there is none.

Problem 2

This question is about validation for the supervised learning problem with one fixed $\mathbb{D}$.

(a) [easy] For one fixed $\mathcal{H}$ and $\mathcal{A}$ (i.e. one model), write below the steps to do a simple validation and include the final step which is shipping the final g.

Inputs:
$\mathcal{H}$
$\mathcal{A}$
$\mathcal{D}$
p : size of the split as a pecent value (ie 80)
function simpleValidate($\mathcal{H}$, $\mathcal{A}$, $\mathcal{D}$, p) :
let $\mathcal{D}_{\text{train}}$ = A subset of $\mathcal{D}$ such that it contains p percent of $\mathcal{D}$
let $\mathcal{D}_{\text{test}}$ = the remaining 1-p percent of $\mathcal{D}$
fit $g = \mathcal{A}(H, D_{\text{train}})$
test g using $\mathcal{D}_{\text{test}}$
return g

(b) [easy] For one fixed $\mathcal{H}$ and $\mathcal{A}$ (i.e. one model), write below the steps to do a K-fold cross validation and include the final step which is shipping the final g.

Inputs:
$\mathcal{H}$
$\mathcal{A}$
$\mathcal{D}$
K : number of folds of $\mathcal{D}$
function simpleCrossValidate($\mathcal{H}$, $\mathcal{A}$, $\mathcal{D}$, K) :
1: partition $\mathcal{D}$ into K pieces, labeled $\mathcal{D}_1, D_2 ... D_K$
2: let $\mathcal{G} = []$
3: let $cost = 0$
4: let $k = 1$
5: $\mathcal{D}_{\text{test}}$ =D$_k$
6: $\mathcal{D}_{\text{train}}$ = all $\mathcal{D}_1, D_2, ..., D_K$ except $\mathcal{D}_k$
7: fit $g_k = \mathcal{A}(H, D_{\text{train}})$
8: test g_k using $\mathcal{D}_{\text{test}}$
9: add test cost to $cost$
10: add g_k to $\mathcal{G}$
11: increment k by 1
12: repeat steps 5-11 until $k > K$
13: g_{final} = average of all $g's$ in $\mathcal{G}$
14: return g_{final}

(c) [harder] For one fixed $\mathcal{H}$ and $\mathcal{A}$ (i.e. one model), write below the steps to do a bootstrap validation and include the final step which is shipping the final g.

Inputs:
$\mathcal{H}$
$\mathcal{A}$
$\mathcal{D}$ with dimension n x (p+1)

function simpleBootstrapValidate($\mathcal{H}$, $\mathcal{A}$, $\mathcal{D}$) :
1: let $\mathcal{D}_{\text{train}}$ = a random sample with replacement of $\mathcal{D}$ (with size n)
2: let $\mathcal{D}_{\text{test}}$ = the remaining samples in $\mathcal{D}$ not in $\mathcal{D}_{\text{train}}$
3: fit $g = \mathcal{A}(H, D_{\text{train}})$
4: test g using $\mathcal{D}_{\text{test}}$
5: return g

(d) [harder] For one fixed $\mathcal{H}_1, \ldots \mathcal{H}_M$ and $\mathcal{A}$ (i.e. M different models), write below the steps to do a simple validation and include the final step which is shipping the final g.

Inputs:
$\mathcal{H}$ (multiple H's)
$\mathcal{A}$
$\mathcal{D}$

p : size of the split as a pecent value (ie 80)
function simpleValidate($\mathcal{H}$, $\mathcal{A}$, $\mathcal{D}$, p) :
1: let $min\_cost = 100$
2: let $g_{\text{final}} = []$
3: let $i = 1$
4: let $\mathcal{D}_{\text{train}}$ = A subset of $\mathcal{D}$ such that it contains p percent of $\mathcal{D}$
5: let $\mathcal{D}_{\text{test}}$ = the remaining 1-p percent of $\mathcal{D}$
6: fit $g = \mathcal{A}(H_i, D_{\text{train}})$
7: let $cost$ = test of g using $\mathcal{D}_{\text{test}}$
8: if $cost < min\_cost$: $min\_cost = cost$, $g_{\text{final}} = g$
9: increment i by 1
10: repeat steps 4-9 until $i > M$
11: return g_{final}

(e) [difficult] For one fixed $\mathcal{H}_1, \ldots \mathcal{H}_M$ and $\mathcal{A}$ (i.e. M different models), write below the steps to do a K-fold cross validation and include the final step which is shipping the final g. This is not an easy problem! There are a lot of steps and a lot to keep track of...

Inputs:
$\mathcal{H}$
$\mathcal{A}$
$\mathcal{D}$
K : number of folds of $\mathcal{D}$
function ComplexCrossValidate($\mathcal{H}$, $\mathcal{A}$, $\mathcal{D}$, K) :
1: let $min\_cost = 100$
2: let $g_{\text{final}} = []$
3: let $i = 1$ (iterating through all the models)
4: partition $\mathcal{D}$ into K pieces, labeled $\mathcal{D}_1, D_2 ... D_K$
5: let $\mathcal{G} = []$
6: let $avg\_cost = 0$
7: let $k = 1$ (iterate through folds)
8: $\mathcal{D}_{\text{test}}$ =D$_k$
9: $\mathcal{D}_{\text{train}}$ = all $\mathcal{D}_1, D_2, ..., D_K$ except $\mathcal{D}_k$
10: fit $g_k = \mathcal{A}(H_i, D_{\text{train}})$
11: $cost$ = test g_k using $\mathcal{D}_{\text{test}}$
12: add $cost$ to avg_cost
13: add g_k to $\mathcal{G}$
14: increment k by 1
15: repeat steps 5-11 until $k > K$

16: compute $cost\_avg = cost\_avg/K$
17: g_{final} = average of all $g's$ in $\mathcal{G}$
18: if $avg\_cost < min\_cost$: $min\_cost = avg\_cost$, $g_{\text{final}} = g$
19: increment i by 1
20: repeat steps 5-11 until $i > M$
21: return g_{final}

Problem 3

This question is about ridge regression — an alternative to OLS.

(a) [harder] Imagine we are in the "Luis situation" where we have $\boldsymbol{X}$ with dimension $n \times (p+1)$ but $p+1 > n$ and we still want to do OLS. Why would the OLS solution we found previously break down in this case?

In order to do OLS what we do is find the orthogonal projection of $\boldsymbol{Y}$ onto the columnspace of $\boldsymbol{X}$ we use the $\hat{H}$ matrix, $\hat{Y} = \hat{H}Y = X(X^TX)^{-1}X^TY$

This is a valid only only if the matrix (X^TX) is invertible.

When $\boldsymbol{X}$ has dimensions $n \times (p+1)$, and $p+1 < n$ it has rank $p+1$ and the square matrix (X^TX) also has rank $p+1$ and dimension $p+1 \times p+1$ and therefore invertible.

When $p+1 > n$, the dimension is $p+1 \times p+1$ but the rank is now n, which is less than $p+1$. The matrix is not full rank and therefore not invertible, so we cant use the OLS method.

(b) [harder] We will embark now to provide a solution for this case. The solution will also give nice results for other situations besides the Luis situation as well. First, assume λ is a positive constant and demonstrate that the expression $\lambda\,||\boldsymbol{w}||^2 = \boldsymbol{w}^\top(\lambda\boldsymbol{I})\boldsymbol{w}$ i.e. it can be expressed as a quadratic form where $\lambda\boldsymbol{I}$ is the determining matrix. We will call this term $\lambda\,||\boldsymbol{w}||^2$ the "ridge penalty".

$\lambda\,||\boldsymbol{w}||^2$
$\lambda\boldsymbol{w}^\top\boldsymbol{w}$
$\boldsymbol{w}^\top(\lambda\boldsymbol{I})\boldsymbol{w}$
$\boldsymbol{w}^\top(\lambda\boldsymbol{I})\boldsymbol{w}$
$\boldsymbol{w}^\top(\lambda\boldsymbol{I})\boldsymbol{w}$

(c) [easy] Write the $\mathcal{H}$ for OLS below where there parameter is the $\boldsymbol{w}$ vector. $\boldsymbol{w} \in$?

(d) [easy] Write the error objective function that OLS minimizes using vectors, then expand the terms similar to the previous homework assignment.

$w = argmin(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^\top$ ($\boldsymbol{y}$- $\boldsymbol{X}\boldsymbol{w}$)

(e) [easy] Now add the ridge penalty $\lambda ||\boldsymbol{w}||^2$ to the expanded form you just found and write it below. We will term this two-part error function the "ridge objective".

$w = argmin(\boldsymbol{y}\text{- }\boldsymbol{X}\boldsymbol{w})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \lambda ||\boldsymbol{w}||^2$

(f) [easy] Note that the ridge objective looks a bit like the hinge loss we spoke about when we were learning about support vector machines. There are two pieces of this error function in counterbalance. When this is minimized, describe conceptually what is going on.

The λ constant is added into the hat matrix, it places a lower limit on the orthogonalization of $\boldsymbol{y}$ in order to prevent overfitting.

(g) [harder] Now, the ridge penalty term as a quadratic form can be combined with the last term in the least squares error from OLS. Do this, then use the rules of vector derivatives we learned to take $d/d\boldsymbol{w}$ and write the answer below.

(h) [easy] Now set that derivative equal to zero. What matrix needs to be invertible to solve?

(i) [difficult] There's a theorem that says *positive definite* matrices are invertible. A matrix is said to be positive definite if every quadratic form is positive for all vectors i.e. if $\forall \boldsymbol{z} \neq \boldsymbol{0} \quad \boldsymbol{z}^\top A \boldsymbol{z} > 0$ then A is positive definite. Prove this matrix from the previous question is positive definite.

(j) [easy] Now that it's positive definite (and thus invertible), solve for the $\boldsymbol{w}$ that is the argmin of the ridge objective, call it $\boldsymbol{b}_{ridge}$. Note that this is called the "ridge estimator" and computing it is called "ridge regression" and it was invented by Hoerl and Kennard in 1970.

(k) [easy] Did we just figure out a way out of Luis's situation? Explain.

(l) [harder] It turns out in the Luis situation, many of the values of the entries of $\boldsymbol{b}_{ridge}$ are close to 0. Why should that be? Can you explain now conceptually how ridge regression works?

(m) [easy] Find $\hat{\boldsymbol{y}}$ as a function of $\boldsymbol{y}$ using $\boldsymbol{b}_{ridge}$. Is $\hat{\boldsymbol{y}}$ an orthogonal projection of $\boldsymbol{y}$ onto the column space of $\boldsymbol{X}$?

(n) [E.C.] Show that this $\hat{\boldsymbol{y}}$ is an orthogonal projection of $\boldsymbol{y}$ onto the column space of some matrix $\boldsymbol{X}_{ridge}$ (which is not $\boldsymbol{X}$!) and explain how to construct $\boldsymbol{X}_{ridge}$ on a separate page.

(o) [easy] Is the $\mathcal{H}$ for OLS the same as the $\mathcal{H}$ for ridge regression? Yes/no.
Is the $\mathcal{A}$ for OLS the same as the $\mathcal{A}$ for ridge regression? Yes/no.

(p) [harder] What is a good way to pick the value of λ, the hyperparameter of the $\mathcal{A}$ = ridge?

(q) [easy] In classification via $\mathcal{A}$ = support vector machines with hinge loss, how should we pick the value of λ? Hint: same as previous question!

(r) [E.C.] Besides the Luis situation, in what other situations will ridge regression save the day?

(s) [difficult] The ridge penalty is beautiful because you were able to take the derivative and get an analytical solution. Consider the following algorithm:

$$\boldsymbol{b}_{lasso} = \underset{\boldsymbol{w} \in \mathbb{R}^{p+1}}{\arg\min} \left\{ (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \lambda ||\boldsymbol{w}||^1 \right\}$$

This penalty is called the "lasso penalty" and it is different from the ridge penalty in that it is not the norm of $\boldsymbol{w}$ squared but just the norm of $\boldsymbol{w}$. It turns out this algorithm (even though it has no closed form analytic solution and must be solved numerically a la the SVM) is very useful! In "lasso regression" the values of $\boldsymbol{b}_{lasso}$ are not shrunk *towards* 0 they are harshly punished *directly to* 0! How do you think lasso regression would be useful in data science? Feel free to look at the Internet and write a few sentences below.

(t) [easy] Is the $\mathcal{H}$ for OLS the same as the $\mathcal{H}$ for lasso regression? Yes/no.
Is the $\mathcal{A}$ for OLS the same as the $\mathcal{A}$ for lasso regression? Yes/no.

Problem 4

These are questions about non-parametric regression.

(a) [easy] In problem 1, we talked about schemes to validate algorithms which tried M different prespecified models. Where did these models come from?

(b) [harder] What is the weakness in using M pre-specified models?

(c) [difficult] Explain the steps clearly in forward stepwise linear regression.

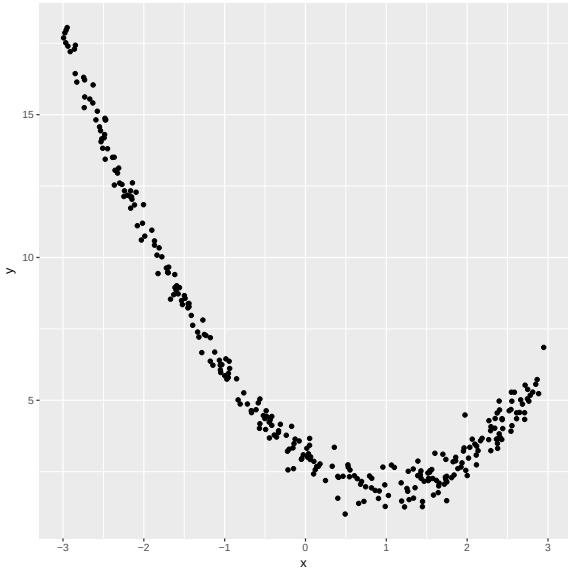
(d) [difficult] Explain the steps clearly in *backwards* stepwise linear regression.

(e) [harder] What is the weakness(es) in this stepwise procedure?

(f) [easy] Define "non-parametric regression". What problem(s) does it solve? What are its goals? Discuss.

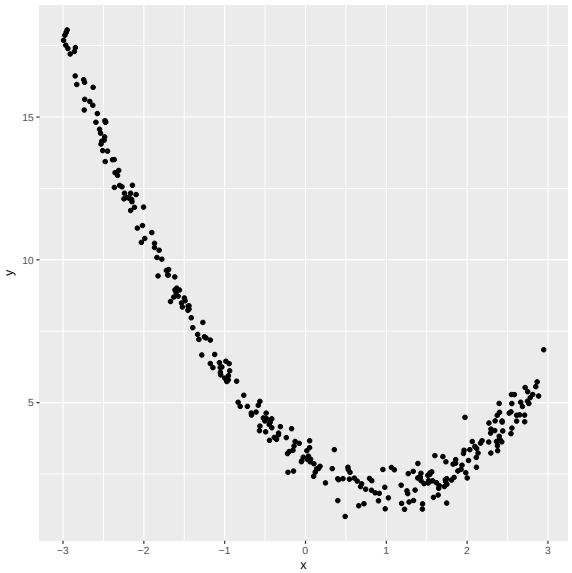
(g) [harder] Provide the steps for the regression tree (the one algorithm we discussed in class) below.

(h) [easy] Consider the following data

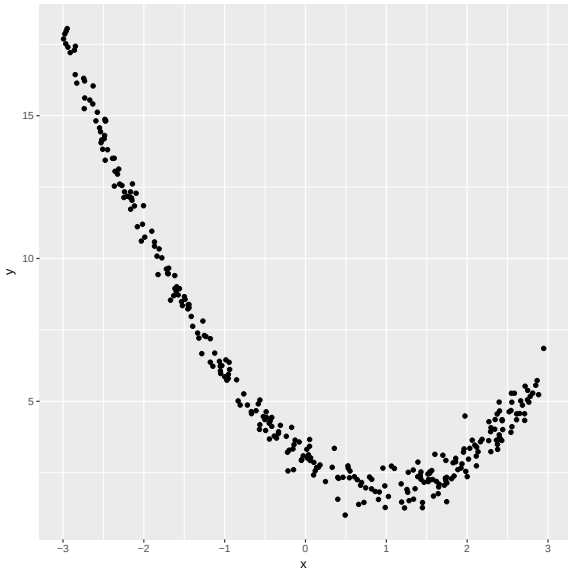


Create a tree with maximum depth 1 (i.e one split at the root node) and plot g above.

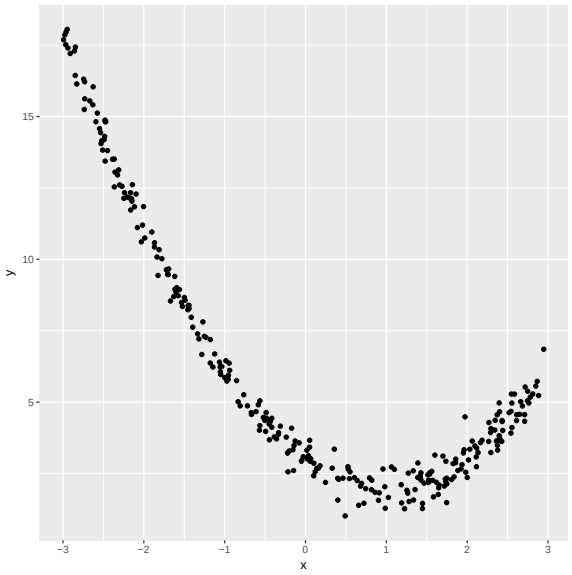
(i) [easy] Now add a second split to the tree and plot g below.



(j) [easy] Now add a third split to the tree and plot g below.

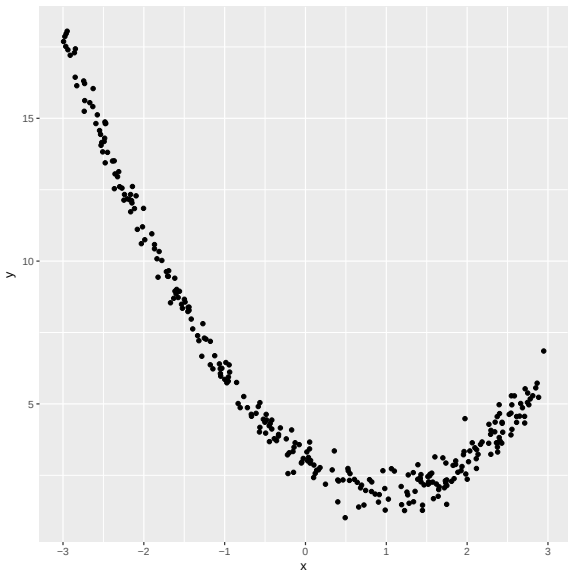


(k) [easy] Now add a fourth split to the tree and plot g below.



(l) [easy] Draw a tree diagram of g below indicating which nodes are the root, inner nodes and leaves. Indicate split rules and leaf values clearly.

(m) [easy] Plot g below for the mature tree with the default $N_0 =$ `nodesize` hyperparameter.



(n) [easy] If $N_0 = 1$, what would likely go wrong?

(o) [easy] How should you pick the $N_0 =$ `nodesize` hyperparameter in practice?

Problem 5

These are questions about classification trees.

(a) [easy] How are classification trees different than regression trees?

(b) [harder] What are the steps in the classification tree algorithm?

Problem 6

These are questions about measuring performance of a classifier.

(a) [easy] What is a confusion table?

Consider the following in-sample confusion table where "> 50K" is the positive class:

```
          y_hats_train
y_train  <=50K  >50K
  <=50K   3475   262
  >50K     471   792
```

(b) [easy] Calculate the following: n (sample size) =

FP (false positives) =

TP (true positives) =

FN (false negatives) =

TN (true negatives) =

$\#P$ (number positive) =

$\#N$ (number negative) =

$\#PP$ (number predicted positive) =

$\#PN$ (number predicted negative) =

$\#P/n$ (prevalence / marginal rate / base rate) =

$(FP + FN)/n$ (misclassification error) =

$(TP + TN)/n$ (accuracy) =

$TP/\#PP$ (precision) =

$TP/\#P$ (recall, sensitivity, true positive rate, TPR) =

$2/(\text{recall}^{-1} + \text{precision}^{-1})$ (F1 score) =

$FP/\#PP$ (false discovery rate, FDR) =

$FP/\#N$ (false positive rate, FPR) =

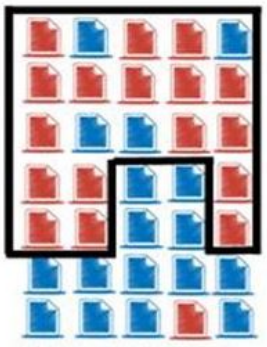
$FN/\#PN$ (false omission rate, FOR) =

$FN/\#P$ (false negative rate, FNR) =

(c) [easy] Why is FPR also called the "false alarm rate"?

(d) [easy] Why is FNR also called the "miss rate"?

(e) [easy] Below let the red icons be the positive class and the blue icons be the negative class.



The icons included inside the black border are those that have $\hat{y} = 1$. Compute both precision and recall.

(f) [harder] There is always a tradeoff of FP vs FN. However, in some situations, you will look at FPR vs. FNR. Describe such a classification scenario. It does not have to be this income amount classification problem, it can be any problem you can think of.

(g) [harder] There is always a tradeoff of FP vs FN. However, in some situations, you will look at FDR vs. FOR. Describe such a classification scenario. It does not have to be this income amount classification problem, it can be any problem you can think of.

(h) [harder] There is always a tradeoff of FP vs FN. However, in some situations, you will look at precision vs. recall. Describe such a classification scenario. It does not have to be this income amount classification problem, it can be any problem you can think of.

(i) [harder] There is always a tradeoff of FP vs FN. However, in some situations, you will look only at an overall metric such as accuracy (or $F1$). Describe such a classification scenario. It does not have to be this income amount classification problem, it can be any problem you can think of.