Fa23: 1)if given a logistic curve, and probabilities, get the y value made at each x value, and multiply the y values to get your answer **(1*.9*.4) = 0.36** /2) gradient descent, do derivative of function, then plug into function **|**start - alpha(d/d(start))**|** ex: alpha = 0.5, starting at (0,0) and the derivative is 2x+1 == (0-0.5(2(0)+1) = 0 - 0.5 = 0.5), so now it's (0.5, 0).
NOTE when doing derivatives, chain rule, bring the exponent down, and do derivatives inside the parenthesis. / 3a) for soft margin svm and given W, b and X and y, do **|** w*x (w1*x1) + (w2*x2) - b **|** if > 0, it is +1, if less than 0, it is -1. / 3b) **large C value** separates data with minimal slack, **rbf kernel** can capture scattered patterns, **use a smaller C value** if there are outliers of colors in areas they don't belong, **RBF kernel with high gamma** handles scattered data across the map/ 4)design matrix is: $X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix}$ replace values with said values (**X**), XtX would have to be a transpose, so it must be the **row(1st matrix)*colum(2nd matrix)**,

y = y1, y2, y3/ w must be a **2x1** matrix/ that is a 3x1 matrix, can do process of elimination/ fw,b is **(w1*w2 + w1)**/ **intercept** is the fir**st element of w**/**fw,b** must be a number 5a) calculate **Minkowski distance** using given points **sqrt(|zi - xi|^2)** zi being z1, z2, etc/5b **3NN classify** get y values of closest distances, and choose most frequent /5c **3-NN regression** predict = y value of smallest distances, and do average of y values /**5d weighted 3-NN classify z** Sum inverse (1/x) of distances for each class, then assign z to the class with the highest number with its y'. Since the closest neighbor belongs to class 1 and has the highest weight z is classified as class 1.// **6a)** look at threshold, remove values below number//**6b**) **r_regression** features most linearly correlated with y, k = 2 means only two options //**6c**) f_regression is variance and correlation/ **7 t/f)** if two functions are same w/ a different constant, it is an alternative//gradient desc doesn't call to f// **hinge loss** = max(0,1- y(wx+b)) // **cannot** build a 3nn model without errors// **cannot** train an SVM, then remove support vectors//**SVM(supportVectorMachine) margin** = 2/||w|| (w = sqrt(x^2+y^2+z^2)// every decision tree and kNN regression function is a step **function**// gradient desc can fail if stuck in a **local min/max**// 8a) **min/max rescaling** = (x - min(x))/(max(x)- min(x)) // 8b) **OneHOtEncoder:** binary encoder, set 1 where values are true, columns are 'if' // **9a1) entropy** of node in bits by finding total(zeros)/sum(y) and total(ones)/ sum(y) and then doing H(S)=−(p0*log2(p0)+p1*log2(p1))//

[illegible]	(output)		
activity	swim	bike	run
swim	1	0	0
bike	0	1	0
run	0	0	1
bike	0	1	0

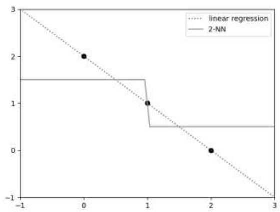
SP24: logistic model & estimate probability: 1/1+e^-(wx+b) // **t/f:** product of probabilities **can underflow** in fixed-precision computer arithmetic. **CANNOT OVERFLOW**/ **differentiating** a sum is easier than **differentiating** a product/ **cannot** use ln() to find a closed-form expression// ln() is strictly increasing, maximizing same as minimizing negative log // SVM classify x by wx+b. If >0, +1. If <0, -1// svm constraint holds if >1. Done by y(wx+b)//
f**eature, threshold**, find midpoints of all features for a threshold, then use **y column** to see class **0**, and class **1**. Then ones that fit threshold, do how many **count(zeros) / total(zeros)**, then add **count(ones)/total(zeros)**. We want the **most ones** displayed on one side of a threshold.
for a **decision tree**, use x1 values to calculate a midpoint, use a threshold for x1, do **average** of the fitted threshold, and MSE (x1 - avg)^2/n (n being number of how many fit the threshold), then do for x2. Once done, do (examples/total examples)*MSE + 2nd MSE threshold to get the answer/ **5**) **standardRescaling = ($x_{orignal}$** - mean)/stddev **6) cosine similarity distance = (z*x)/(||z||*||x||)** \\\\ ||z|| = sqrt($z1^2$+$z2^2$), once cosine sim distance is found, subtract/add to y value to solve. **7a) MSE** = 1/n * $(y_i - y_i^{hat})^2$, do **wx_1+b** and **wx_2+b**, then do ((**wx_1+b**)^2 + (**wx_2+b**)^2) / n to

get MSE // **7b)** to find **best fit line**, do mu_1 = mean(x_1+x_2) and mu_2 = mean(y_1+y_2), then do ((x_1-mu_1)(y_1-mu_2)+(x_2-mu_1)(y_2-mu_2)) / ((x_1-mu_1)(y_1-mu_1))^2 to get w, to get b, do **(y_2-mu_2) - w(x_1-mu_1) // logistic regression is** exponential, no need for additional function/ **linear regression model IS** sensitive to the signs of label// weighted kNN are inversely proportional to distance// **SGD** can use more iterations with a smaller learning rate α than **GD//Lasso regression** tends to set most coefficients to zero.)

FA24: gradient desc can fail if alpha is too big; will have + and - values in each iteration/ gradient desc can fail by descending without bound//**clf.SVM)** if n**onlinear and not comple**x. Choose **kernel = rbf** and **low gamma, less complex** // Two clouds, linearly separable, few outliers, choose **linear kernel**; with low **C** value, low c = more outliers // Random mixed points inside disk (nonlinear), choose **rbf kernel**, and **high gamma**, makes shape tight

SP23: P(y=1|x) = 1/(1+e$^{-(wx+b)}$) // Gradient desc can fail to converge on **convex** function if stuck in local min// decisiontreeclassifiers are blocky, KNN is **blocky** and has turns, RBF is **curvy** //

I(0) = − log2(zerocount/total) //



<u>FINAL</u>

<u>K-means on data</u> - {x} = {1,2,3} c1 = 2, c2 = 3, do |x-c1| and |x-c2|, whichever is lower, assign to c value. Once done, do the average of each c (c1 = 1.5, c2 = 3). If k = 3. Choose three numbers in {x} unless given. If 2nd iteration, use first iterations found.

<u>Grid vs random search</u> - grid needs to be fine and goes through individual values in a range, Takes a lot of time also needs specific options to run well. - random search samples values at the mean, and samples from range rather than individual, more random.

<u>T/F</u> - If 0 < d < p < D, PCA on **p** to **d** is same as **D** to **d** | if x,y = -1, (wx+b) >0 is FALSE | threshold is required to predict yhat for lr | same outcome when MSE and logarithm of MSE is minimized | y wx + b = −1, 0, 1, w is normal to all 3 functions | Ridge regression fixes overfiting of training data | **bagging** and **random forest** methods use resampling with replacement from training examples | **Boosting** builds models sequentially and fixes issues as it builds | **k-NN** may benefit from standardizing features | **Decision trees** don't benefit from standardizing features. | standardizing x in **linear regression** does not benefit | PCA uses greatest variability in the data.

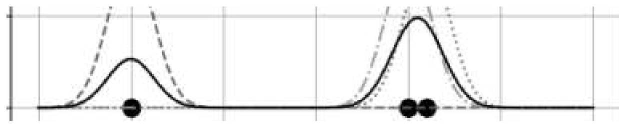
<u>Yes / No</u> - **methods used for feature selection/extraction**: Lasso regression, Correlation, Kernel-trick, Principal component analysis, Permutation feature importance | **methods not used**: Kernel Trick, Kernel Density Estimation, Gradient Descent, Ridge Regression

<u>Gaussian Curves -</u> Look at the graph, and at each point. Get the y values at each point for each color. Get the average of each value on each point, then get the average of the values.

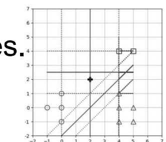
- (1.6+0+0)/3 at 1 - At 4, (1.6+1.4+0)/3 - At 4.2. (1.4+1.6+0)/3 - (1+1+1.6)/3 = 1.2

if asked for another point, find the values for each color at that point, then do the average to get the answer.

- For density function: the height of the bell curve is at stddev, or where the graph changes shape. Group the bell curves together

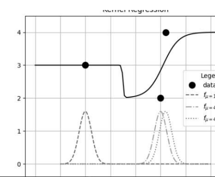


Gradient Boosting - Do the average of y values. Dont worry about what X is, because its not considered. Otherwise, it is where the predicted y value is.
To train a 2nd model, do y values - average of y values/predicted values.
Binary Classifier V >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>.



Question 8

- Kernel regression - get the y values of each bell curve and add it to a group (x, y, z). Add them together, and divide the group by the sum of them (xyz)/(x+y+z). Then get the y value of dataset, and multiply it by the divided group and add (x/xyz * a) + (y/xyz * b) + (z/xyz * c)
- Y = f(x) for x = 1 - get the y value of each color, and multiply them by the y dataset, and add them together to get the answer / 8b is easy, just look at y value of x = 1
- Estimated regression function - Blue is more important than orange, Hitting the edge is not realistic, make sure the dots are being touched



Question9 - model assessment

- Y is real answer, yhat is predicted, if yhat = y, is correct (if It is a TN or TP. but if y is 0 and yhat is 1, is FP, viceversa Is a FN. (1 = postive). Remember recall and precision formula

Predicted ŷ	0	1
Actual y = 0	TN	FP
Actual y = 1	FN	TP

- Calculating AUC - 1st(FP = 0, TP = 0.5) or 2nd(FP = 0.5, TP = 1). For 1st(0, 0.5) 0.5 - 0 = 0.5. Next do 0.5 * (2nd(FP) + 2nd(TP)) do it again, but do 0.5 * 2nd(TP + 2nd(TP)). Add 0.5 * (2nd(FP) + 2nd(TP)) + 0.5 * 2nd(TP + 2nd(TP)) to get answer.
- If TPR = 0, AUC is 0 because it is predicting wrong all the time
- low FN and high TP, **Recall** / high TP and low FP, **precision**. THINK LOGICALLY FOR THESE QUESTIONS

RANDOM STUFF

Precision - proportion of relevant documents (TP/ (TP + TN))
Recall - proportion of relevant documents returned to relevant documents available (TP/(TP+FN))
False negative def - predicts a negative outcome that is positive, cons - missed opportunity for correction, and bad for healthcare, leads to not detecting disease
False positive def - incorrectly predicts a positive that is negative, cons - unnecessary testing, leads to making right things wrong, false alarms and wrongful accusations
Random forest - chooses random subsets for trees based on depth, trees and features
Gradient boosting - builds trees sequentially vs independently (random forest)
Imbalance datasets solutions - oversample, undersample, make synthetic examples, train_test_split

- Combine models

    - Average predictions, stacking (uses more info and uses different models for information)

Unsupervised learning - works with no y value,

- Clustering
    - Chooses examples and has a centroid, labels each example closest to centroid, until a convergence is found
- DBSCAN
    - Puts x in cluster if too close to other points with threshold epsilon and n, the number of values that can be in a cluster. Neighbors of clusters are clusters
    - Good for noisy data, weird shaped data, outliers of DB scan have no neighbors
- Dimension reality
    - Maps x into vector with less features to reduce noise and to be able to visualize it
    - PCA benefits
        - Saves memory, space and computation time due to compressing information, can visualize data up to [0, 10]^D

Homeworks

HW4 - Training Data: Linear regression, because it has the lowest MSE, meaning it fits the data better

Test Data: Ridge, because it reduces over/under fitting on unknown data and reduces noise

Feature Selection: Lasso, because many of the w coefficients are close to 0, so lasso would already make those 0s irrelevant

Review bs

Question 5

- Std is where the curve changes
- KERNEL DENSITY ESTIMATION
- Zeros at the curve, via the color
    - Look at the color
    - (1.6+0+0)/3 at 1
    - At 4, (1.6+1.4+0)/3
    - At 4.2. (1.4+1.6+0)/3
    - (1+1+1.6)/3 = 1.2

Question 9

- ???

Question 7 (weird graph one)

- Hard margin means linearly separable
- Blue is +1, red and green is -1
- For each of the 1 vs rest classifier, find the rest of the signed distance of boundaries
- Questions
    - For blue, make diagonal SVM???
        -
    -

Question 8c

- Three weights have to sum to one
    - ½ ½ 0
    - Got by adding (0 + 1.5+1.5)/sum of 3 values
- 8b is easy, just look at y value of x = 1
- 8c
    - Blue is more important than orange
    - Hitting the edge is not realistic, make sure the dots are being touched

Question 6

- What does x = 2 do???
    - Starts with a constant model, which is the average of y
    - Does nothing??? Just a constant model. IT DOES NOTHING.