Sta-370 Intro to ML (Practice Exam)

Your Name: _____

Directions

- Answer each question using *no more than the specified number of sentences*, and do not attempt to avoid these guidelines using run-on sentences. Answers that are unnecessarily verbose may be penalized.
- Do not include superfluous information in your answers, you may be penalized if you make an inaccurate statement even if you go on to provide a correct answer. Your answers should be clear, concise, and include only what is needed to answer the question that was asked.

Formulas

- **True Positive Rate (TPR)** - $\frac{\text{True Positives}}{\text{Total Positives}}$
- **Sensitivity** and **Recall** - TPR
- **Specificity** - $1 - \text{FPR}$
- **Precision** - $\frac{\text{True Positives}}{\text{Total Predicted Positives}}$
- **Balanced Accuracy** - $\frac{\text{TPR}+\text{TNR}}{2}$
- **F1 Score** - $\frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}$

Section 1

Part A: In a few sentences, explain your understanding of the *bias-variance trade off*. More specifically, why do methods with lower bias often exhibit higher variance? How does one go about balancing these when developing a machine learning model?

- Bias is caused by a model lacking the structural complexity to detect patterns in the training data, so lowering bias will increase the propensity for the model to pick up on chance artifacts in the training data that look like real patterns (overfitting). We can balance the bias-variance by using performance on out-of-sample data, such as a validation set or using cross-validation.

Part B: Consider the application of 5-fold cross-validation to evaluate the performance of the decision tree classification algorithm with maximum depth of 4 using a training set consisting of $n = 100$ observations.

- i: How many times must a decision tree be trained during this evaluation process?
  - 5 trees
- ii: How many data-points are used to train each decision tree involved in the evaluation process?
  - 80 observations (4/5 of the data)
- iii: How many times is each data-point used as part of the training set within the evaluation process?
  - 4 times

Part C: Consider a machine learning application that seeks to classify emails as either "spam" or "not spam". On the test set, 980 of 1000 "not spam" emails were correctly classified but only 120 of 300 "spam" emails were correctly classified.

- i: Create a confusion matrix summarizing the performance of this classifier. Consider the outcome "spam" to be the positive class.
  - Actual Spam: 120, 180 (predicted spam, non-spam, respectively)
  - Actual Non-Spam: 20, 980 (predicted spam, non-spam, respectively)
- ii: What is the *classification accuracy* of the classifier?
  - $1100/1300 = 0.846$
- iii: What is the *F1 score* of the classifier?
  - Precision $= 120/140 = 0.857$, Recall $= 120/300 = 0.4$, so F1 $= \frac{2 \cdot 0.857 \cdot 0.4}{0.857 + 0.4} = 0.545$
- iv: Which metric, classification accuracy or F1 score, provides a more useful assessment of how effective this classifier is? Briefly explain.
  - F1 is more useful due to the large class imbalance, which makes the accuracy seem high even though the model isn't performing well with the spam class.
- v: What would need to change about the application or the nature of the observed data in order for the other metric (whichever you did not select in part iv) to become the more useful quantity? Briefly explain.
  - Collecting a balanced dataset (equal amounts of spam and non-spam) would make accuracy more useful, as this would allow for the benefits of accuracy (easy interpretation) without needing to worry about high baseline performance due to a class imbalance.

Part D: Clearly mark each of the following statements as either TRUE or FALSE. If a statement is false, briefly explain why it is incorrect.

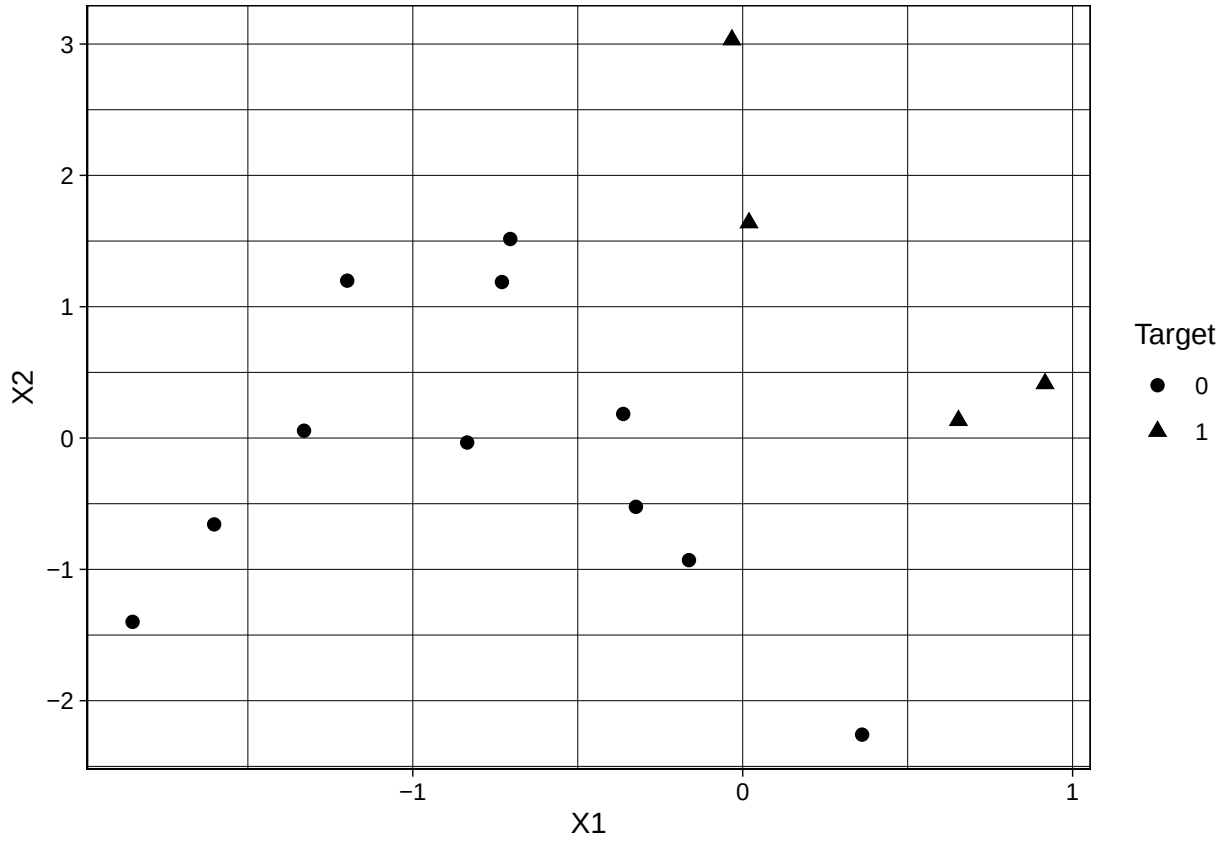
- i: Stochastic gradient descent exhibits noisy convergence because it uses a single training observation for each parameter update.
  - TRUE
- ii: The neurons in the hidden layers of a neural network require an activation function that introduces non-linearity, otherwise the depth of the network would be meaningless.
  - TRUE

- iii: In k-nearest neighbors, uniform weighting allows neighbors that are closer in distance to a new data-point to have a larger influence on the algorithm's prediction than neighbors that are further away from the new data-point.
  - FALSE - uniform weighting treats all neighbors equally regardless of their distance.
- iv: In k-nearest neighbors, using more neighbors will decrease the bias of the algorithm's predictions.
  - FALSE - increasing k increases bias due to more neighbors contributing to predictions (smoother prediction line, less variation)
- v: While the random forest algorithm benefits from bagging, or bootstrap aggregation, this is not an essential component of the algorithm.
  - FALSE - bootstrapping is one of the defining components of the random forest algorithm, it will not work without bootstrapping as all the base learners will use the exact same training data and thus not vary from each other.
- vi: An essential step in setting up a gradient descent algorithm is differentiating the cost function with respect to the model's unknown parameters.
  - TRUE

Section 2

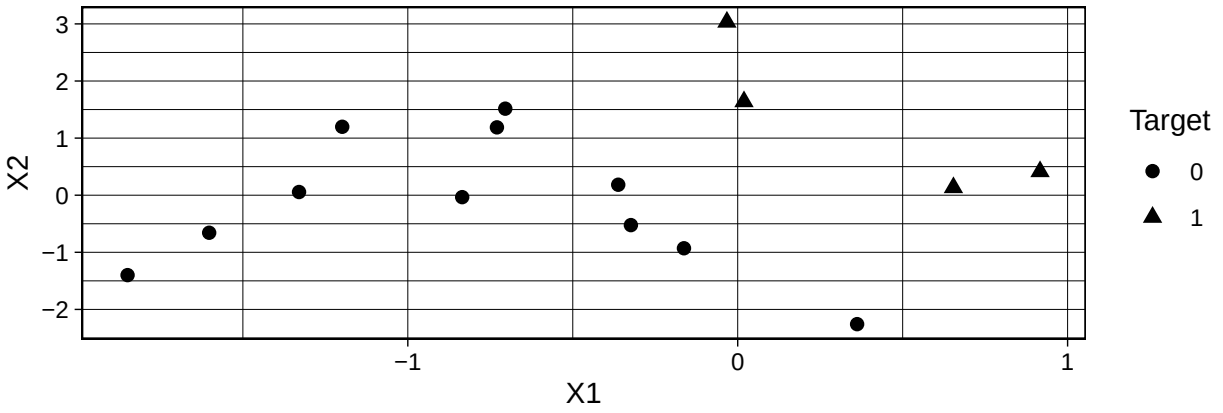
Part A: Consider the training data shown below and the goal of correctly classifying "Target" as either 1 or 0. On the given graph

- i: Consider a KNN model using $k = 3$ neighbors and uniform weighting. What is this model's predicted class and predicted probability for a new observation with X1 = 0, X2 = 2?
  - Class = 1, Prob = 2/3
- ii: Consider altering the KNN model from i to use *distance weighting*. What impact would this have on the model's predicted class and predicted probability for the new observation with X1 = 0, X2 = 2?
  - Class is unchanged, predicted prob increases since the closest neighbors were class 1
- iii: On the provided graph, sketch the splitting rules of a decision tree classifier with a maximum depth of 2. Your split locations don't need to be perfect, but they should clearly illustrate that you understand this method.
  - Split 1 around X1 = -0.1, Split 2 in the right partition around X2 = 0
- iv: Consider *decreasing* the maximum depth of the decision tree classifier to a value of 1. What consequences does this change have on the *bias* and *variance* of the classification method?
  - Increase bias and decrease variance.



Part B: Consider gradient boosting using decision tree base learners with a maximum depth of 1.

- i: Sketch decision boundary of the first base learner on the first plot given below.
  - Single split around X1 = -0.1
- ii: For the purposes of this question, define the residual to be the true target (0 or 1) minus the model's estimate of $P(y_i = 1)$. Sketch the residuals after the first round of boosting (versus the more important predictor for second boosting round), then sketch the model from the second round of boosting on the second plot given below.
  - X2 is the more important predictor. Residuals are all zeros to the left of X1 = -0.1 (a total of 10 points ranging from X2 = -1.5 to X = 3), the residuals are 0.2 for 4 points (all with X2 > 0) and 0.8 for 1 point to the right of X1 = -0.2 (largest value of X2). Thus, a satisfactory answer will find a split that either accommodates the 4 residuals of 0.2 or the single residual of 0.8.
- iii: Briefly explain how the model will be trained in the third round of boosting. That is, what is the model? What are the training data?
  - A decision tree is fit to the residuals that remain after the second boosting iteration, with the training data being the original X values and the outcome being those residuals.
- iv: Consider the base learners from the three boosting iterations described so far. Indicate how the final model is formed from the prediction functions found during these iterations if $\alpha = 0.05$ is used as the learning rate.
  - $\hat{f}(x) = f_1(x) + 0.05f_2(x) + 0.05f_3(x)$
- v: In your own words, explain the process of *early stopping* when training a gradient boosted classifier. Your explanation should be 2-3 sentences.
  - Early stopping monitors performance on a validation set as additional trees are added. Training stops once a set number of training iterations have passed since the last decrease in validation error



Section 3

Consider the application of simple neural network consisting of one hidden layer with only one neuron and the sigmoid activation function. The input data set will involve 2 predictors. Furthermore, you may assume the outcome variable is numeric and the network is trained using the squared error cost function: Cost $= \frac{1}{n}(\mathbf{y} - \hat{\mathbf{y}})^T(\mathbf{y} - \hat{\mathbf{y}})$

Part A: Create a sketch of this network's architecture. In total, how many weight parameters are present in this network?

- 3 weights (2 in the first layer going to the neuron, 1 from the neuron going to the output)

Part B: Assume that biases are used in both the hidden layer and the final layer. How many bias parameters are present in this network?

- 2 biases (1 in the first layer, 1 going to the output)

Part C: Under the simplifying assumption that we'll be training this network using batches of size $n = 1$ (ie: a single 2-dimensional vector $\mathbf{x}_i$ containing the values of each predictor for the i^{th} sample), provide a chain rule expression for each weight and bias parameter in terms of partial derivatives. You do not need to solve for each partial derivative. For example, terms of the form $\frac{\partial a}{\partial b}$ can remain in your answer.

$$\frac{\partial Cost}{\partial \mathbf{w}^{(1)}} = \frac{\partial Cost}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial \mathbf{w}^{(1)}}$$

$$\frac{\partial Cost}{\partial w^{(2)}} = \frac{\partial Cost}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w^{(2)}}$$

$$\frac{\partial Cost}{\partial b^{(1)}} = \frac{\partial Cost}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a} \frac{\partial a}{\partial z} \frac{\partial z}{\partial b^{(1)}}$$

$$\frac{\partial Cost}{\partial b^{(2)}} = \frac{\partial Cost}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b^{(2)}}$$

Part D: Annotate the chain rule expressions you provided in Part C to include the dimensions of every partial derivative.

- Since $\hat{y}$ is a scalar, everything is a scalar except for partial derivatives involving $\mathbf{w}^{(1)}$, which are 1×2

Part E: Briefly explain how the gradient calculated in Part C is used to update the weight parameters of the network during training.

- Gradient descent uses iterative updates of the form: $\theta_{new} = \theta_{old} - \eta \cdot \text{Gradient}(\theta_{old})$; moving in the negative gradient is expected to change the parameter to a value producing a lower cost.

Part F: If the gradient component for a particular weight parameter evaluates to a negative value at the current iteration of gradient descent, how would you expect the cost function to change if the value of that weight parameter is *decreased*?

- A negative gradient indicates that increasing the weight would decrease the cost. Therefore, decreasing the weight would generally cause the cost function to increase.