Time: 25 minutes

Instructions:
- Do not plagiarize. Do not assist your classmates in plagiarism.
- The first question is a True/False question. A wrong answer will fetch you a **negative score of 1 point.**
- The first question is an Multiple Choice Question. There could be multiple correct answers and all need to be marked for full credit. A wrong (incorrect or missed) answer will fetch you a **negative score of 0.5 points.**
- You can leave any questions that you are not sure of.
- In the unlikely case a question is not clear, discuss it with an invigilating TA. Please ensure that you clearly include any assumptions you make, even after clarification from the invigilator.

1. (5 points) State True or False.

   (i) I could train a linear regression model on the $\log(odds)$ of the output variable y, and use the model as a binary classifier to decide whether $y = 1$ or $y = -1$.
   Solution: True

   (ii) The logistic regression model can be designed modeling the predicted output variable as a binomial random variable. Recall that a $Binomial(n, p) = \binom{n}{x} p^x (1-p)^{(n-x)}$, where $x = \{0, 1, 2, ..., n\}$ and n is a positive integer.
   Solution: True

   (iii) Logistic regression cannot be trained using gradient descent because the output variable is discrete and therefore non-differentiable.
   Solution: False

   (iv) Linear regression requires all the input variables to be continuous to ensure that the model remains differentiable and subsequently gradient descent could be applied as the training algorithm.
   Solution: False

   (v) Using a regularizer with linear regression can help mitigate overfitting to training data.
   Solution: True

2. (5 points) Select all that may apply.

   (i) I would resolve overfitting of my model by

   (A) Adding a regularizer
   (B) Training a more complex model (with more parameters)
   (C) Getting more data, both in numbers and diversity
   (D) All of the above

   Solution: A, C

   (ii) Cross-validation can be used for

   (A) Model Selection
   (B) Hyperparameter tuning (e.g., the regularization parameter λ in ridge regression)
   (C) Estimating true error (or test error, i.e., error on unseen data)
   (D) None of the above

   Solution: A, B, C

   (iii) If my student gets a training accuracy of 95% and a testing accuracy of 78% on an ML problem, you'll tell him that

   (A) His model is overfitting

(B) His model is underfitting

(C) His test set is not representative

(D) None of the above

Solution: A
You can only conclusively say that the model is overfitting because of the large difference between the training & test errors. We cannot comment on whether the data is representative or not.

(iv) If my student gets a training accuracy of 78% and a testing accuracy of 95% using an ML model he trained to convergence, you'll tell him that

(A) His model is overfitting

(B) His model is underfitting

(C) His test set is not representative

(D) None of the above

Solution: C
Since the test set has much better accuracy than the training set, it is evident that the test split is easier than the rest of the data, thus not representative of the data.
Since the training set performance is not decidedly poor (78% accuracy may be acceptable in many cases), this is not a typical case of underfitting.

(v) Select all the statements that are always true.

(A) The linear regression model assumes additive, zero-mean Gaussian noise

(B) Logistic regression is a non-linear model due to the use of the sigmoid function

(C) Training error is always lower than validation or test error

(D) k-fold cross-validation randomly selects samples to create a validation set in each of the k iterations.

Solution: A
Despite the use of the sigmoid function, logistic regression is a linear model due to the relationship between the parameters and the input variables, i.e., $w^\top x$ that models the $\log(odds)$ of the output variable.

Training error can be higher than test error if we have a poorly represented test set.

After an initial random shuffling of the data, the k-fold cross-validation method partitions the data into k disjoint sets, taking each of the sets as a validation set in each of the k iterations.