USC CSCI 467 Intro to Machine Learning	**Final Exam** December 7, 2023, 2:00-4:00pm	Fall 2023 Instructor: Robin Jia

Name: _____

USC e-mail: _____@usc.edu

Answer the questions in the spaces provided. **If you run out of space, continue your work on the last two pages, and indicate that your answer is there**. You may use the backs of pages for scratch work only. **Please use pen for ease of grading**. This exam has 7 questions, for a total of 150 points.

Question 1: K-Means and Global Optima (26 points)

Recall the k-means objective function is

$$L\left(z_{1:n}, \mu_{1:k}\right) = \sum_{i=1}^{n} \left\| x^{(i)} - \mu_{z_i} \right\|^2,$$

where $z_{1:n}$ are the cluster assignments, $\mu_{1:k}$ are the corresponding cluster centroids, and $x^{(i)}$ is the i-th input datapoint. Now consider the following dataset consisting of three datapoints in a 2-D plane: $\{(0,3),(0,0),(4,0)\}$.

(a) (6 points) Suppose you run k-means with $k = 2$ and have the initial cluster centers be $\mu_1 = (4,0)$ and $\mu_2 = (0,3)$. What will be the final cluster centers when k-means converges? Show your work and explain how you know that k-means has converged.

Solution: In the first round, we will group $(0,3)$ and $(0,0)$ together in cluster 2, and $(4,0)$ will be by itself in cluster 1. Thus, we'll get new cluster centers

$$\boxed{\mu_1 = (4,0), \mu_2 = (0,1.5)}.$$

This is a convergence point because $(4,0)$ is closer to μ_1 and the other two points are closer to μ_2. Thus the cluster assignments do not change, so k-means has converged.

(b) (6 points) Prove that the above initialization converges to the global minimum for $k = 2$. (Hint: Enumerate over all possible cluster assignments and compute the objective for each assignment.)

Solution: There are three possible groupings:

1. $(4,0)$ with $(0,0);(0,3)$
2. $(4,0)$ with $(0,3);(0,0)$
3. $(0,3)$ with $(0,0);(4,0)$

The objective for (1.) is: 8; for (2.) is: 25/2; for (3.) is: 9/2. Using our initialization in part (a) can lead us to group $(0,3)$ and $(0,0)$, and $(4,0)$ as the other group. This assignment has the minimum loss. Hence, it's a global minimum.

(c) (6 points) Suppose we pick two distinct datapoints (out of the three datapoints) uniformly at random as the cluster centers at initialization. Will k-means always converge to the global minimum on this dataset? Justify your answer.

Solution: There are three cases of the initialization choices:

1. Selecting $(4,0)$ and $(0,0)$
2. Selecting $(4,0)$ and $(0,3)$
3. Selecting $(0,3)$ and $(0,0)$

We have already studied case 2. For (1.), we will end up grouping $(0, 3)$ with $(0, 0)$. Hence, this is an optimal global assignment. For (3.). we will end up grouping $(4, 0)$ with $(0, 0)$. According to our discussion of (b), this is not an optimal assignment. So, convergence the global minimum is not guaranteed.

(d) For this question, you may **not** use k-means as an answer for either part.

i. (4 points) Name one machine learning method from class (not including k-means) that **is** guaranteed to converge to the global optimum of its objective function. Explain why it is guaranteed.

Solution: Any linear supervised learning method (linear regression, logistic regression, softmax regression, SVM) is guarnateed to converge because they solve a convex problem.
PCA is also guaranteed to achieve the minimum because we can convert the objective to a problem that has a closed-form solution based on eigenvectors of the covariance matrix.

ii. (4 points) Name one machine learning method from class (not including k-means) that **is not** guaranteed to converge to the global optimum of its objective function. Explain why it is not guaranteed.

Solution: Any neural network-based learning method is a possible answer, as they create non-convex learning problems. It is also valid to say that gradient descent is not guaranteed to converge in general.

Question 2: Transformers and Multi-Headed Attention (24 points)

Consider a Transformer model with a single 1-headed self-attention layer. As input, we pass in four words, each represented with an embedding of dimension 2. That is, the inputs are vectors $w_1, w_2, w_3, w_4 \in \mathbb{R}^2$.

The single headed self-attention layer is parameterized by matrices $K, Q, V \in \mathbb{R}^{2\times 2}$. For each w_i, the key, query, and value vectors are defined as $k_i = Kw_i$, $q_i = Qw_i$, and $v_i = Vw_i$, respectively. For simplicity, we provide the following values:

$$q_1 = \begin{bmatrix}3\\7\end{bmatrix} \quad q_2 = \begin{bmatrix}1\\0\end{bmatrix} \quad q_3 = \begin{bmatrix}0\\2\end{bmatrix} \quad q_4 = \begin{bmatrix}2\\5\end{bmatrix}$$
$$k_1 = \begin{bmatrix}1\\12\end{bmatrix} \quad k_2 = \begin{bmatrix}1\\12\end{bmatrix} \quad k_3 = \begin{bmatrix}1\\12\end{bmatrix} \quad k_4 = \begin{bmatrix}1\\26\end{bmatrix}$$
$$v_1 = \begin{bmatrix}12\\6\end{bmatrix} \quad v_2 = \begin{bmatrix}9\\3\end{bmatrix} \quad v_3 = \begin{bmatrix}3\\6\end{bmatrix} \quad v_4 = \begin{bmatrix}4\\1\end{bmatrix}$$

(Note: The values here are created for ease of calculation. Technically, these values are not possible, since k_1, k_2, k_3 are identical but q_1, q_2, q_3 are not.)

(a) (6 points) Assume this transformer is an encoder. Calculate the output of self-attention for w_2. Note that your answer should be a vector $\in \mathbb{R}^2$.

Solution: We note that $q_2 k_i$, for all i from 1 to 4, equals 1. Thus, the self attention will assign equal weighting to all the v_i vectors. Thus, the output of the self attention for q_2 is thus the sum of each of the v_i vectors divided by 4. The answer is thus $\begin{pmatrix}7\\4\end{pmatrix}$.

(b) (6 points) Now assume this transformer is a decoder. Calculate the output of self-attention for w_3. Again, note that your answer should be a vector $\in \mathbb{R}^2$.

Solution: It is important to note that since this is a decoder, we are masking out all future tokens during calculation. Thus, the self attention for w_3 does not attend to w_4.
We note that $q_3 k_i$, for all i from 1 to 3, equals 24. Thus, the self attention will assign equal weighting to the first three v_i vectors. Thus, the output of the self attention for q_3 is thus the sum of v_1, v_2, v_3 vectors divided by 3. The answer is thus $\begin{pmatrix}8\\5\end{pmatrix}$.
When using a decoder, you can attend to past words and the current word, just not future words.

(c) (4 points) During lecture, we saw that the entire self attention mechanism can be represented in matrix form. Why is it important that we are able to do this?

Solution: Hardware are often optimized for matrix calculations (think numpy), which allow for parallelization of computation.

(d) (4 points) Ryan is training a Transformer language model, but it is taking a long time. He notices that the multi-headed attention layers are slow, so he decides to remove all the

multi-headed attention layers from the model. Will the model still perform similarly as before, or will it perform much worse? Explain why.

Solution: It will perform much worse. The multi-headed attention layer is the only part of the Transformer where information from other tokens can be accessed. Without this, the model will be unable to look at anything besides the current word when predicting the next word.

(e) (4 points) Transformer models commonly have multiple layers of multi-headed attention, such as having 12 layers with 12 heads each. An alternative would be to have a single layer with 144 heads, thus having the same total number of attention heads. Give one reason why using 12 layers and 12 heads is generally preferable.

Solution: In general, the advantage of having multiple layers is that later layers can build more complex features that depend on earlier features. If you have a single layer with many heads, no head can use the result of another head to build more complex features.

Question 3: Linear Models and PCA (23 points)

Vishesh is training a linear model on medical data. Each example represents information about a patient collected at a doctor's visit, including their height, medical history, blood test results, etc. The classifier is trying to predict whether the patient required any type of surgery in the 5 years following their visit. He has n training examples, and each training input $x^{(i)}$ is a vector $\in \mathbb{R}^d$, where d is much larger than n.

(a) (2 points) Is Vishesh solving a regression, binary classification, or multi-class classification problem?

Solution: This is binary classification.

(b) (3 points) Explain why having d much larger than n could lead to problems in this scenario.

Solution: Having $d >> n$ makes it very likely that the model will overfit. There is not enough information in the n examples to learn weights for all d features well.

(c) (4 points) Vishesh knows that PCA can be used to address the issue of d being much larger than n. Explain how PCA could be helpful.

Solution: Vishesh could run PCA on the dataset first and extract the top k principal components for some $k < n$. This would reduce the dimensionality of the examples from d to k, so there would no longer be the issue of having too many features relative to the number of examples.

(d) (4 points) If Vishesh were to use PCA to address this issue, this would introduce an additional hyperparameter for his classification problem. What is this hyperparameter and how should Vishesh choose a good value for it?

Solution: The new hyperparameter is the number of principal components to use k. He should choose this by trying different values of k and evaluating the resulting model on a development set.

(e) (6 points) An alternative strategy would be to use L_1 regularization instead of PCA. L_1 regularization will cause many of the classifier's weights to be 0, which essentially means those features get ignored. Give one reason for why L_1 regularization might perform better than using PCA.

(Hint: Imagine that Vishesh is performing multiple tasks with the same dataset, e.g., he could either try to predict whether the patient will require surgery in five years, or whether they will have another doctor visit within the next year. Think about what L_1 regularization would do in these two scenarios compared with what PCA would do.)

Solution: The key difference is that PCA only looks at the x's and not the y's. Therefore, for a fixed value of k, PCA will always return the same k features. However, L_1 regularization is applied during supervised learning, so depending on the classification

task, it could choose a different subset of features for one task vs. another task. This means it can be more customized towards whatever the model is trying to predict.

(f) (4 points) A local hospital has heard about Vishesh's model and wants to use it to decide which patients should get priority when scheduling doctor's visits. Vishesh is concerned that the model could be unfair to members of different racial groups. Describe one evaluation metric that Vishesh could use to measure the fairness of the model. (You don't need to remember the exact name of the metric. If you do give the name, you should still explain what it does.)

Solution: Any of the allocative harm-based fairness metrics discussed in class is an appropriate answer. This includes:

- Checking if race is independent of the model prediction
- Checking if the model's precision is the same for all racial groups
- Checking if the model's recall is the same for all racial groups

Question 4: Explaining Q-Learning (18 points)

In class, we saw the following update rule for Tabular Q-Learning:

$$\hat{Q}(s,a) \leftarrow (1-\eta)\hat{Q}(s,a) + \eta(r + \gamma \hat{V}(s')).$$

(a) (4 points) What is the difference between choosing a large value of η and a small value of η?

Solution: A large value of η is a larger learning rate, and it means that the current observations are given more weight when updating the Q-value. A small value of η means we rely more on the previous observations rather than the current one.

(b) (6 points) In class, we discussed two quantities, $\hat{V}(s')$ and $V_{\text{OPT}}(s')$. Explain (1) the difference between these two quantities **and** (2) why the Q-learning update rule uses $\hat{V}(s')$ instead of $V_{\text{OPT}}(s')$.

Solution:

1. The difference is that $\hat{V}(s')$ is defined in terms of our model's predictions of the Q-value, whereas $V_{\text{OPT}}(s')$ is defined in terms of the true Q-value and true optimal policy.
2. We cannot use $V_{\text{OPT}}(s')$ during Q-learning because we can only compute it if we actually knew the exact transitions and rewards of the MDP, but these are not known during reinforcement learning.

(c) (4 points) In class, we learned about something called ϵ-Greedy. What is the purpose of ϵ-Greedy? What would go wrong if we did not use ϵ-Greedy when running Q-learning?

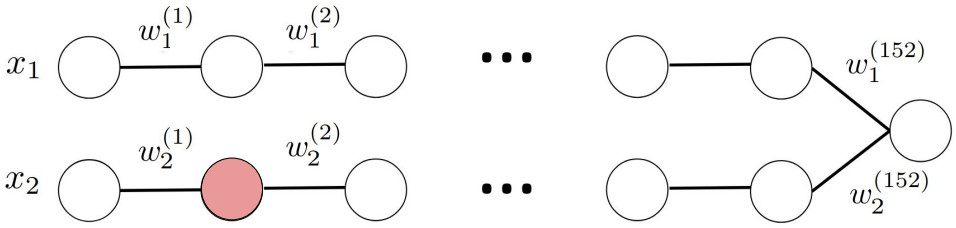
Solution: The purpose is to encourage exploration by occasionally choosing random actions. Without ϵ-Greedy, the agent could rule out good actions based on a single bad experience, and get stuck with a suboptimal policy.

(d) (4 points) Your friend knows that the choice of ϵ affects the final result of Q-learning, but is confused about how this is possible because ϵ does not show up in the update rule for Tabular Q-learning. Explain which parts of the Q-learning update rule would be directly affected when ϵ is changed, and why.

Solution: The use of ϵ-Greedy affects what states s and actions a are tried. In other words, ϵ affects the data that is used to perform the Q-learning updates.

Question 5: Backpropagation (25 points)

Consider the 152-layer neural network in the figure below. Throughout the question, we assume that the training example is (x, y) where $x = (x_1, x_2) \in \mathbb{R}^2$ and $y \in \mathbb{R}$ (all activations and gradients are with respect to this training example). The network has no bias terms, and it is fully-connected only for the final output node (i.e. the input x_1 is not connected to the red node in the next layer and so on, except for the last output node).



The network has a total of $152 \times 2 = 304$ parameters, namely $w_1^{(1)}, \ldots, w_1^{(152)}$ and $w_2^{(1)}, \ldots, w_2^{(152)}$. It has two neurons in each layer, except for the final output layer. The network computes a prediction as follows:

- In the first layer, each neuron $o_i^{(1)}$ is computed by first computing $a_i^{(1)} = w_i^{(1)} x_i$, and then applying the activation function h: $o_i^{(1)} = h(a_i^{(1)})$. For example, the red node takes in x_2, computes $a_2^{(1)} = w_2^{(1)} x_2$, and outputs $o_2^{(1)} = h(a_2^{(1)})$.
- Each subsequent layer before the last layer works the same way, except it takes the previous layer's output as input. So, $o_i^{(j)}$ is computed by first computing $a_i^{(j)} = w_i^{(j)} o_i^{(j-1)}$, then outputting $o_i^{(j)} = h(a_i^{(j)})$.
- The final layer takes in the outputs of the penultimate layer, $o_1^{(151)}$ and $o_2^{(151)}$, and outputs $o^{(152)} = w_1^{(152)} o_1^{(151)} + w_2^{(152)} o_2^{(151)}$.

The activation function $h(a)$ is the same for every neuron.

The network is trained on the standard squared loss, so that $L(w) = \frac{1}{2}(o^{(152)} - y)^2$ for the example (x, y), where w is the vector of all the weights in the network.

Note: For parts (a), (b), and (c), you can use the notation $h'(a)$ to represent the derivative of the activation function $h(\cdot)$ with respect to some input a in your answer.

(a) (3 points) Derive the expression for $\frac{\partial L(w)}{\partial o^{(152)}}$.

Solution:

$$\frac{\partial L(w)}{\partial o^{(152)}} = \frac{\partial}{\partial o^{(152)}} \left(\frac{1}{2} (o^{(152)} - y)^2 \right) = (o^{(152)} - y)$$

(b) (4 points) Derive the expression for $\frac{\partial L(w)}{\partial w_1^{(152)}}$.

Solution: We can apply the chain rule to compute that:

$$\frac{\partial L(w)}{\partial w_1^{(152)}} = \frac{\partial L(w)}{\partial o^{(152)}} \cdot \frac{\partial o^{(152)}}{\partial w_1^{(152)}} = (o^{(152)} - y) \cdot o_1^{(151)}.$$

(c) (10 points) Derive the expression for $\frac{\partial L(w)}{\partial w_1^{(1)}}$.

(Hint: Use the chain rule, for example, $\frac{\partial o_1^{(j)}}{\partial o_1^{(j-1)}} = \frac{\partial o_1^{(j)}}{\partial a_1^{(j)}} \frac{\partial a_1^{(j)}}{\partial o_1^{(j-1)}}$.)

Solution: First, let's compute:

$$\begin{aligned}\frac{\partial L(w)}{\partial o_1^{(1)}} &= \frac{\partial L(w)}{\partial o^{(152)}} \cdot \frac{\partial o^{(152)}}{\partial o_1^{(151)}} \cdot \frac{\partial o_1^{(151)}}{\partial o_1^{(150)}} \cdots \frac{\partial o_1^{(2)}}{\partial o_1^{(1)}} \\ &= \frac{\partial L(w)}{\partial o^{(152)}} \cdot \frac{\partial o^{(152)}}{\partial o_1^{(151)}} \cdot \prod_{j=1}^{150} \frac{\partial o_1^{(j+1)}}{\partial o_1^{(j)}}\end{aligned}$$

The first term in this product was already computed in part (a).
The second term we can compute directly:

$$\frac{\partial o^{(152)}}{\partial o_1^{(151)}} = w_1^{(152)}.$$

Now we compute the third term. For each $j = 1, \cdots, 150$, we know that $o_1^{(j+1)} = h(a_1^{(j+1)})$ and $a_1^{(j+1)} = w_1^{(j+1)} o_1^{(j)}$, so

$$\frac{\partial o_1^{(j+1)}}{\partial o_1^{(j)}} = \frac{\partial o_1^{(j+1)}}{\partial a_1^{(j+1)}} \cdot \frac{\partial a_1^{(j+1)}}{\partial o_1^{(j)}} = h'(a_1^{(j+1)}) \cdot w_1^{(j+1)}$$

for $l = j, \cdots, 150$. Putting all this together, we get:

$$\frac{\partial L(w)}{\partial o_1^{(1)}} = (o^{(152)} - y) \cdot w_1^{(152)} \cdot \prod_{j=1}^{150} h'(a_1^{j+1}) w_1^{(j+1)}.$$

Finally, $w_1^{(1)}$ affects $o_1^{(1)}$ (and does not have any direct effect on any other nodes in the computation graph), so we have:

$$\frac{\partial L(w)}{\partial w_1^{(1)}} = \frac{\partial L(w)}{\partial o_1^{(1)}} \cdot \frac{\partial o_1^{(1)}}{\partial w_1^{(1)}} = \frac{\partial L(w)}{\partial o_1^{(1)}} \cdot \frac{\partial o_1^{(1)}}{\partial a_1^{(1)}} \cdot \frac{\partial a_1^{(1)}}{\partial w_1^{(1)}} = \frac{\partial L(w)}{\partial o_1^{(1)}} \cdot h'(a_1^{(1)}) \cdot x_1$$

Putting everything together, this means our final answer is:

$$\begin{aligned}\frac{\partial L(w)}{\partial w_1^{(1)}} &= (o^{(152)} - y) \cdot h'(a_1^{(1)}) \cdot x_1 \cdot w_1^{(152)} \cdot \prod_{j=1}^{150} h'(a_1^{(j+1)}) w_1^{(j+1)} \\ &= (o^{(152)} - y) \cdot x_1 \cdot \left(\prod_{j=1}^{151} h'(a_1^{(j)}) \right) \cdot \left(\prod_{j=2}^{152} w_1^{(j)} \right).\end{aligned}$$

Forgetting an entire step is a bigger math mistake and should be penalized by at least -1.

(d) (4 points) Bill decides to use the sigmoid activation function $h(a) = \frac{1}{1+e^{-a}}$ for all nodes in this neural network. He notices that when he is trying to train this model, $\frac{\partial L(w)}{\partial w_1^{(1)}}$ is very close to 0. Using your expression from the previous part, explain why this is the case. *(Hint: The gradient of $h(a)$ is $h'(a) = h(a)(1 - h(a))$. You may use the fact that for any real number a, the quantity $h(a)(1 - h(a))$ is always between 0 and 0.25.)*

> **Solution:** When $h(a)$ is the sigmoid activation, $h'(a) \in [0, 0.25]$. This means that the derivative from the previous part contains the product of 151 $h'(a)$ values. Since each of these is at most 0.25, the product of all these values will become very close to 0.
>
> If they have an off-by-one error in part (c) in terms of how many copies of $h'(a)$ get multiplied together, that should not get penalized here.

(e) In class, we learned about a phenomenon where the gradient can often become zero when the computation graph is very long.

i. (2 points) Give the name for this phenomenon.

> **Solution:** This is the vanishing gradient problem.

ii. (2 points) Name one strategy that can be used to combat this phenomenon. (Your answer does not need to apply to the neural network in this question.)

> **Solution:** Acceptable strategies include:
>
> - Using gated connections, such as in a GRU or LSTM.
> - Using residual connections
> - Using ReLU instead of sigmoid.

Question 6: Short Response (14 points)

Answer the following questions and **explain your reasoning fully**. You may also draw explanatory diagrams when appropriate.

(a) (4 points) Wenyang has a CNN model for image classification that uses multiple conv-relu-pool blocks, followed a single feedforward layer. She decides to change all the max pooling layers in her network from 2×2 to 4×4, while the number of layers, type of layers, kernel sizes, and number of input/output channels will stay the same. The size of the input images also does not change. Will the number of parameters in her model have to increase, decrease, or stay the same?

Solution: The total number of parameters will decrease. A 4×4 max pooling layer reduces the size of the input by a larger factor than the 2×2 max pooling layer. Thus, the input to the feedforward layer will be smaller, and thus require fewer parameters.

This is close but not completely correct. Other wrong answers should receive lower marks. 3/4 for saying the CNN layers will have fewer parameters—the problem says that the kernel sizes stay the same.

(b) (4 points) Suppose we want to run linear regression on the following data points. Is there a unique solution for w that minimizes the linear regression loss function?

x_1	x_2	y
1	2	5
3	6	1
-2	-4	0

Solution: Answer: No. One feature is linearly dependent on another feature, as $x_2 = 2x_1$ for all data points. This means that $X^\top X$ is not invertible and thus there is no unique solution.

(c) (6 points) Lorena is training a supervised learning model and keeps collecting more and more training data. As the amount of training data increases,

- The variance of the model (**gets larger/gets smaller/stays the same**).
- The bias of the model (**gets larger/gets smaller/stays the same**).

Circle one choice for each bullet point and explain your answers.

Solution: Our model will have have a **lower** variance but **the same** bias.

As we increase the amount of training data, the learning algorithm will get closer and closer to finding the best possible model within the model family, since it will overfit less and less. This means the variance decreases.

The bias in a model is determined by the complexity of the model family (we are restricted to a set of possible functions we can learn). The amount of training data doesn't change the model's complexity; it only helps the model to estimate its parameters within its complexity constraints relatively more accurately, so we still have the same bias.

Question 7: Multiple Choice (20 points)

In the following questions, circle the correct answer(s). There is no need to explain your answer.

(a) (2 points) **True** or **False**: The sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ from $\mathbb{R} \to \mathbb{R}$ is a convex function.

Solution: False. It is not convex; however, log-sigmoid is convex.

(b) (2 points) **True** or **False**: Decision trees do not suffer from overfitting problems because all x's assigned to the same leaf node receive the exact same prediction.

Solution: False. Decision trees can overfit the training data just like other methods.

(c) (2 points) **True** or **False**: One difference between k-means and GMM is that k-means uses hard assignments whereas GMM uses soft assignments.

Solution: True. k-means assigns each point to a single cluster, whereas GMM infers a distribution over clusters for each point.

(d) (2 points) **True** or **False**: The SVM algorithm minimizes L_2-regularized hinge loss.

Solution: True. SVM minimizes the sum of hinge loss on the training data and an L_2 regularization term.

(e) (2 points) **True** or **False**: Adding Dropout is a viable way to regularize a Transformer model.

Solution: True. Transformers include feedforward layers, where Dropout can be applied.

(f) (2 points) **True** or **False**: In a Markov Decision Process, if an agent visits the same state twice and takes the same action each time, the probability of transitioning to an end state is guaranteed to be the same for both times.

Solution: True. This is a consequence of the Markov property.

(g) (2 points) Which of the following statements is true about the total number of parameters learned by word2vec?

A. It is equal to the dimension of one word vector.
B. It is proportional to the size of the training dataset.
C. It is proportional to the size of the vocabulary.
D. word2vec does not learn any parameters.

Solution: C is correct, since we learn one word vector for each word in the vocabulary.

(h) (3 points) Which of the following statements about PCA are true? Choose all that apply.

A. PCA involves randomly initializing a guess for the best dimension.

B. PCA computes a non-linear transformation of the original dataset.

C. PCA requires computing eigenvalues and eigenvectors of the covariance matrix.

D. PCA can be used to visualize data.

Solution: The answer is CD.

(i) (3 points) Which of the following are best practices for responsible machine learning? Choose all that apply.

A. It is important to identify the right metrics to quantify the performance of the ML model (such as accuracy, precision, recall, performance on sub-groups, etc.)

B. The model may latch onto spurious correlations to make their predictions, so should be tested extensively in use cases that directly impact many people.

C. The model should be tested on diverse data that reflects the use cases in deployment.

D. The model should not be monitored or updated after deployment.

Solution: The answer is ABC.

[This page provides extra space for answers]

[This page provides extra space for answers]