CSCE 633: Machine Learning

SAMPLE EXAM # 1

Fall 2023

Total Time: 75 minutes

Name: _____

UID: _____

Question	*Point*	*Grade*
1	*25*	
2	*25*	
3	*20*	
4	*30*	
Total	*100*	

People Sitting In Your Cluster: **VERSION A**

1. (25 points) Concepts.

For each of the following questions, please provide your answer and an explanation/justification. Please answer the following questions

a) (5 points) Recall that the error in each iteration m of AdaBoost is calculated as:

$$err_m = \frac{\sum_{i=1}^{N} w_i \; I\,(y_i \neq f_m(x_i))}{\sum_{p=1}^{P} w_p}$$

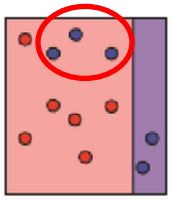
When this boosting process begins (iteration m = 1), describe the differences (if any) to the standard misclassification rate

$$err_m = \sum_{i=1}^{N} \quad I\,(y_i \neq f_m(x_i))$$

The error is exactly the same on the first iteration because everything is weighted equally as 1/N – be careful typo in the formula p=1 to P instead of I = 1 to N – no one asked about this in the test

+ 3 if they know that sample weights change
+ 2 if they only know the initial weight is 1/N
(only 3 points if they talk about how adaboost weighs differently missing the first round explanation)

b) (5 points) In the following iteration of Boosting



Please describe which points will increase in weight and which will stay the same or decrease in weight (depending on the boosting algorithm used). Additionally, tell us why learning these new weights should be done slowly.

Those three increase (3 points)

Weights shifting too fast may overfit to outliers (2 points)

c) (5 points) Describe how regularization modifies loss functions.

LOSS + lamda * regularization – increases penalty which shrinks weights to make model smaller/more efficient.
Define regularization (2)
Explain why and how regularization term changes the loss function (3)

d) (5 points) Please explain the differences between Ridge regression and Lasso regularization and how they impact a logistic regression model.

L1 Regularization, also called a lasso regression, adds the "absolute value of magnitude" of the coefficient as a penalty term to the loss function. L2 Regularization, also called a ridge regression, adds the "squared magnitude" of the coefficient as the penalty term to the loss function. lasso actually drives coefficients to 0/feature selects, while ridge shrinks but does not eliminate

+ 3 both regularize
+ 2 LASSO feature selects whereas ridge just shrinks but doesn't set to 0

e) (5 points) Please describe, in brief, how Bagging, Random Forest, and XGBoost train decision tree models, highlighting the key differences.

+3 descriptions (1 for Bagging, 1 for Random Forest and 1 for Boosting)
+ 2 differences between random/bagging for Bagging + RF vs. sequential building for XGBoost and when you care about overfitting

2. (25 points) Trees

Given the weather conditions, we want to predict if a person is going to go for a run or not. The data that we have collected are the following:

Sample	Features		Outcome: Go for Run?
	Forecast	Temperature	
1	Sunny	Cool	Yes
2	Sunny	Hot	Yes
3	Overcast	Cool	Yes
4	Rain	Cool	No
5	Rain	Hot	Yes
6	Overcast	Hot	No
7	Rain	Cool	No

a) (15 points) Draw a depth = 2 tree. This means splitting the tree once on one variable and once on the other variable, using the Gini Index, which is defined as follows:

$Gini = \sum_{k=1}^{K} \widehat{p_{mk}} (1 - \widehat{p_{mk}})$, where K is the number of classes, and $\widehat{p_{mk}}$ represents the proportion of the training observations in the mth region that are from the kth class.

Show the calculations at each step. You can round/approximate.

- Gini Index for Forecast:

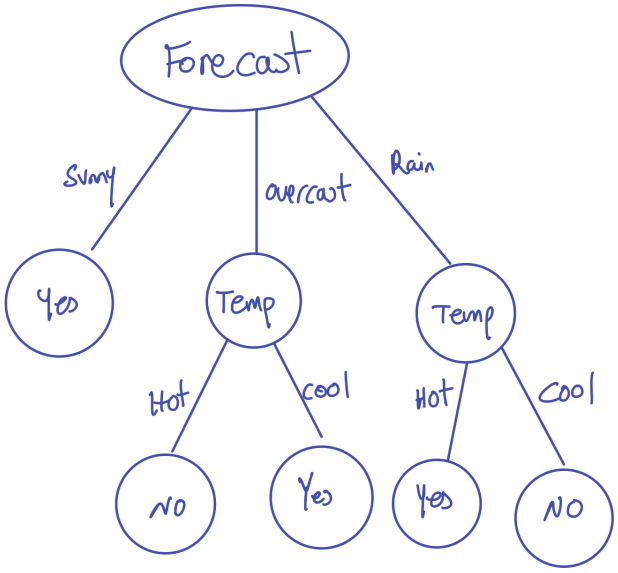
Sunny	Yes	2	$\frac{2}{2}(1-\frac{2}{2})=0$
Sunny	No	0	$\frac{0}{2}(1-\frac{0}{2})=0$
Overcast	Yes	1	$1/2 \cdot 1/2 = 1/4$
Overcast	No	1	$1/2 \cdot 1/2 = 1/4$
Rain	Yes	1	$1/3 \cdot 2/3 = \frac{2}{9}$
Rain	No	2	$2/3 \cdot 1/3 = \frac{2}{9}$

$\sum P_{mk}(1-P_{mk}) = 0 + 1/2 + 4/9 = 17/18$

- Gini Index for Temperature:

cool	Yes	2	$2/3 \cdot 1/3 = 2/9$
cool	No	1	$1/3 \cdot 2/3 = 2/9$
Hot	Yes	2	$2/4(1-2/4) = 1/4$
Hot	No	2	$2/4(1-2/4) = 1/4$

$\sum P_{mk}(1-P_{mk}) = 2/9 + 2/9 + 1/4 + 1/4 = 4/9 + 1/2 = 17/1$



17/18 for both features - we should have an explanation of what tree we put but basically note it is okay either way

Gini index calculation = 7 points
Tree based on the previous step calculation = 8 points

b) (5 points) Describe the procedure for picking a classification in leaf nodes that are not completely a single class

+4 If a node is not pure, we can do majority voting
+1 point if they talk about what happens if perfectly split 50/50

c) (5 points) Now evaluate the following test set, please provide the accuracy in the following test set.

Sample	**Features**		**Outcome: Go for Run?**
	Forecast	**Temperature**	
8	**Sunny**	**Cool**	**Yes**
9	**Sunny**	**Hot**	**Yes**
10	**Overcast**	**Cool**	**No**

Version A:

Sample 8: prediction: yes GT: Yes
Sample 9: prediction: yes GT:Yes
Sample 10: prediction: yes GT:No
Accuracy = 2/3

Version B:

Sample 8: prediction: Yes, GT: Yes Sample 9: prediction: yes GT: No, Sample 10: Prediction: yes GT: no

Accuracy = 1/3

Version C:

Sample 8: prediction: Yes, GT: No Sample 9: prediction: yes GT: Yes, Sample 10: Prediction: yes GT: no

Accuracy = 1/3

Version D:

Sample 8: prediction: Yes, GT: Yes Sample 9: prediction: yes GT: No, Sample 10: Prediction: yes GT: Yes

Accuracy = 2/3

+1 for Each prediction
+2 for Accuracy

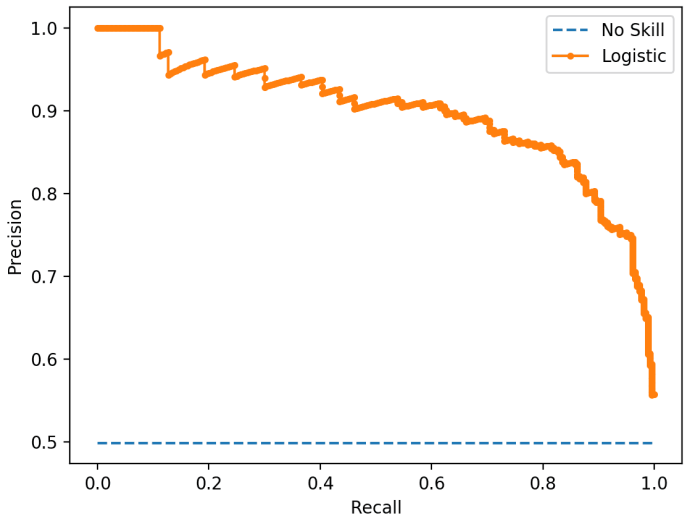
3. (20 points) Calculations

For the following parts of this question, recall that

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

Like an ROC curve (which uses False Positive Rate on the x axis and true positive rate on the y axis), a method by which we evaluate model performance is by the area under the curve (AUC) of the precision recall curve (PRC), illustrated here:



a) (10 points) Please calculate the precision and recall with thresholds p=0.45 and p=0.55. (You can approximate the fraction values to 2 decimal points). Use these values to draw a Precision and Recall Curve

Predicted Probability	0	5	10	14	24	50	53	57	75	100
Ground Truth	**No**	**No**	**No**	**Yes**	**No**	**Yes**	**Yes**	**No**	**Yes**	**Yes**
p = 0.45	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes
p = 0.55	No	No	No	No	No	No	No	Yes	Yes	Yes
P = 0.25	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes
P = 0.65	No	No	No	No	No	No	No	No	Yes	Yes
P = 0.35	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes
P = 0.6	No	No	No	No	No	No	No	No	Yes	Yes
P = 0.15	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes
P = 0.85	No	No	No	No	No	No	No	No	No	Yes

+2 points Predictions for each threshold
+2 points Precision and Recall for each threshold
+2 for Curve

Precision = TP / (TP + FP)
Recall = TP / (TP + FN)

Version A

P = 0.45: TP = 4 FP = 1 TN = 4 FN = 1
P = 0.55: TP = 2 FP = 1 TN = 4 FN = 3

P = 0.45: Precision = 4/(4+1) = 0.8 Recall = 4/(4+1)=0.8
P = 0.55: Precision = 2/(2+1) = 0.67 Recall = 2/(2+3)=0.4

Version B

P = 0.25: TP = 4 FP = 1 TN = 4 FN = 1
P = 0.65: TP = 2 FP = 0 TN =5 FN = 3

P = 0.25: Precision = 4/(4+1) = 0.8 Recall = 4/(4+1)=0.8
P = 0.65: Precision = 2/(2+0) = 1 Recall = 2/(2+3)=0.4

Version C

P = 0.35: TP = 4 FP = 1 TN = 4 FN = 1
P = 0.6: TP = 2 FP = 0 TN =5 FN = 3

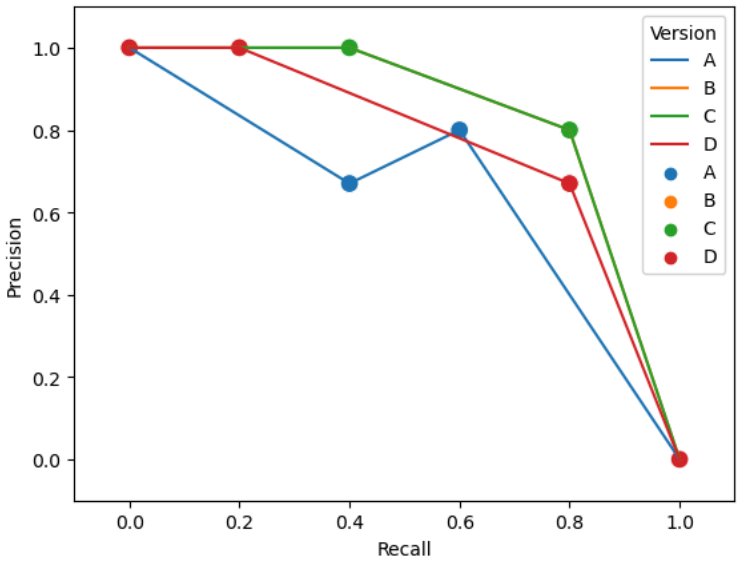
P = 0.35: Precision = 4/(4+1) = 0.8 Recall = 4/(4+1)=0.8
P = 0.6: Precision = 2/(2+0) = 1 Recall = 2/(2+3)=0.4

Version D

P = 0.15: TP = 4 FP = 2 TN = 3 FN = 1
P = 0.85: TP = 1 FP = 0 TN =5 FN = 4

P = 0.15: Precision = 4/(4+2) = 0.67 Recall = 4/(4+1)=0.8
P = 0.85: Precision = 1/(1+0) = 1 Recall = 1/(1+4)=0.2

B and C are the same.



(10 points) Assume you have collected data for a group of students with variables X1 = Amount of Hunger, X2 = Money Saved, and outcome Y = whether they ate at a particular restaurant. You fit a logistic regression model and produce estimated coefficients $\beta_0 = -6$, $\beta_1 = 0.05$ and $\beta_2 = 1$.

Given the probability of an event is calculated as

$$p(y|x) = \frac{e^{\beta 0 + \beta 1\ x_1 + \beta 2\, x_2}}{1 + e^{\beta 0 + \beta 1\ x_1 + \beta 2\, x_2}}$$

Which can also be calculated as

$$\log\left(\frac{p(y|x)}{1 - p(y|x)}\right) = \beta 0 + \beta 1\ x_1 + \beta 2\, x_2$$

If I have 3.5 money saved, how much hunger is necessary to choose to eat at a restaurant with probability of 50%?

+8 Setting up the formula correctly
+2 arithmetic

Version A:
Log (0.5 / (1-0.5)) = Log(1) = 0

0 = -6 + 0.05*X1 + 1*3.5 => 0.05X1 = 2.5 => X1 = 2.5/0.05 = 50

Version B:

Log(0.75(1-0.75)) = Log(3)
Log(3) = -6 +0.05X1+4.5 => 0.05X1 = log(3)+1.5 => X1 = 20(log(3)+1.5)

Version C:

0 = -4+0.25X1+3.5 => 0.25X1 = 0.5 => X1 = 0.5*4 = 2

Version D:

Log(0.8/(1-0.8))=Log(0.8/0.2)=Log(4)

Log(4) = -6 + 0.05X1 + 2.5 => 0.05X1 = log(4)+3.5 => X1 = 20(Log(4)+3.5)

4. (30 Points) Debugging. Assume you have the following implementation of Linear Regression for a Regression purpose and Logistic Regression for a classification purpose, and you want to evaluate the model performance using AUROC value for the Logistic Regression and root-mean-square error (RMSE) for the Linear Regression model. However, when testing these models, they do not seem to work correctly. (Hint: there are 10 corrections that need to be made)

Please identify the sources of possible errors in the code and provide fixes for them.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix

class LinearRegression_Local:
    def __init__(self, learning_rate=0.00001, iterations=30):
        self.learning_rate = learning_rate
        self.iterations    = iterations
    def fit(self, X, Y):
        self.m, self.n = X.shape
        self.W = np.zeros(self.n)
        self.b = 0
        self.X = X
        for i in range(self.iterations) :
            self.update_weights()
    def update_weights(self):
        Y_pred = self.predict(self.X)
        dW = - (2 * (self.X.T).dot(self.Y - Y_pred))
        db = - 2 * np.sum(self.Y - Y_pred)
        self.W = self.W - dW
        self.b = self.b - db
    def predict(self, X):
        return X.dot(self.W) + self.b

def model_pred_continous_rmse(X_train, y_train, X_test, y_test):
    Linear = LinearRegression_Local(0.0001, 30)
    Linear.fit(X_train, y_train)
    y_pred = Linear.predict(X_train)
    rmse = np.mean((y_pred-y_test)**2)
    return rmse

def model_pred_binary_auroc(X_train, y_train, X_test, y_test):
    logistic = LogisticRegression().fit(X_train, y_train)
    y_pred = logistic.predict(X_train)
    tn, fp, fn, tp = confusion_matrix(y_test, y_pred)
    fpr = fp/(fp+fn)
    fnr = fn/(fn+tp)
    plt.plot(fpr, fnr)
    plt.title("ROC Curve")
    plt.show()
    return roc_auc_score(y_test, y_pred)
```

This Page Intentionally Left Blank

No penalties for incorrect corrections unless there is an egregious mistake being made

+3 points for each bug: 2 points To find error 1 point for the solution

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

class LinearRegression_Local:
    def __init__(self, learning_rate=0.00001, iterations=30):
        self.learning_rate = learning_rate
        self.iterations    = iterations
    def fit(self, X, Y):
        self.m, self.n = X.shape
        self.W = np.zeros(self.n)
        self.b = 0
        self.X = X
        for i in range(self.iterations) :
            self.update_weights()
    def update_weights(self):
        Y_pred = self.predict(self.X)
        dW = - (2 * (self.X.T).dot(self.Y - Y_pred)) # need to divid by self.m  (1)
        db = - 2 * np.sum(self.Y - Y_pred) # need to divid by self.m  (2)
        self.W = self.W - dW  # need to multiply the derivative by self.learning_rate
(3)
        self.b = self.b - db # need to multiply the derivative by
self.learning_rate  (4)
    def predict(self, X):
        return X.dot(self.W) + self.b

def model_pred_continous_rmse(X_train, y_train, X_test, y_test):
    Linear = LinearRegression_Local(0.0001, 30)
    Linear.fit(X_train, y_train)
    y_pred = Linear.predict(X_train) # need to be X_test (5)
    rmse = np.mean((y_pred-y_test)**2) #correct np.sqrt(np.mean((y_pred-y_test)**2))
(6)
    return rmse

def model_pred_binary_aucroc(X_train, y_train, X_test, y_test):
    logistic = LogisticRegression().fit(X_train, y_train)
    y_pred = logistic.predict(X_train) # need to use predict_proba to make predictions
on x_test + need to use X_test instead of X_train (7, 8)
    tn, fp, fn, tp = confusion_matrix(y_test, y_pred)
    fpr = fp/(fp+fn) #correct: fp/(fp+tn) (9)
    fnr = fn/(fn+tp)
    plt.plot(fpr, fnr, label='ROC Curve') #correct: fpr, tpr (10)
    plt.legend()
    plt.show()
    return roc_auc_score(y_test, y_pred)
```