ECS171 Winter 2024
Midterm Study Guide

Cheat sheet

Sigmoid and derivative of sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\sigma'(x) = \frac{d}{dx}\sigma(x) = \sigma(x)(1 - \sigma(x))$$

ReLU and Derivative for ReLU:

$$f(x) = max(0, x) = \begin{cases} x & if\ x > 0, \\ 0 & otherwise. \end{cases}$$

$$f'(x) = \begin{cases} 1 & if\ x > 0, \\ 0 & otherwise, \end{cases}$$

Gradient descent weight update:

$$W^{t+1} = W^t - \lambda \cdot \nabla J(W^t)$$

Errors:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Variance:

$$Var = \sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}$$

Chain Rule:

$$if\ h(x) = f\big(g(x)\big)$$

$$h'(x) = f'(g(x)) \cdot g'(x)$$

Performance Metrics:

$$Precision = \frac{TP}{TP + FP}$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Recall\ (True\ Positive\ Rate) = \frac{TP}{TP + FN}$$

$$(True\ Negative\ Rate)\ TNR = \frac{TN}{TN + FP}$$

$$F_1 score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

$$(False\ Positive\ Rate)\ FPR = \frac{FP}{TN + FP}$$

Q1:
In machine learning, what is the dropout technique, and how does it help prevent overfitting in neural networks?

Answer:
Dropout is a regularization technique in neural networks where during training, a random set of neurons are "dropped out" or turned off with a certain probability. This helps to prevent overfitting by making the network more robust and less reliant on specific neurons.

For example, let's consider a neural network with dropout applied to a hidden layer. If the dropout rate is set to 0.5, during training, for each forward and backward pass, half of the neurons in that layer will be randomly deactivated. This means the network has to learn to be effective even when only a fraction of its neurons are active. This prevents the network from becoming overly specialized in recognizing patterns in the training data and helps it generalize better to unseen data.

Q2:
Given the hours of exercising per week measured in hours (x_1) and time from last Covid-19 infection measured in weeks(x_2), the model predicts the probability that the person will be re-infected with Covid-19 in the next 5 months. The model follows the Linear Regression $\hat{y} = 0.4 - 0.05x_1 + 0.07x_2$, for each of the data pairs in the training and testing sets, do the following:

(a) Compute the predicted output for the given regression model for each set of input
(b) Compute the bias (SSE) for each set of inputs
(c) Compute the variance.
(d) Is the model overfit, underfit, or a good fit? Justify your answer using the following metrics for the base case:
SSE_train = 0.050
SSE_test = 0.049
Variance = 0.01

Where needed, round your answer to 4 d.p.

Training set:

x_1	x_2	y	$\hat{y}$	$(y - \hat{y})^2$
0.0	4.0	0.70		
3.0	10.0	0.95		
2.0	3.0	0.60		
5.0	1.0	0.15		
8.0	5.5	0.25		

12.0	7.5	0.23		
10.0	4.0	0.20		
3.0	2.0	0.30		

Testing set:

x_1	x_2	y	$\hat{y}$	$(y-\hat{y})^2$
1.0	9.0	0.95		
9.0	6.0	0.20		
7.0	3.0	0.25		
5.0	5.0	0.45		

Answer:

Training set:

x_1	x_2	y	$\hat{y}$	$(y-\hat{y})^2$
0.0	4.0	0.70	0.68	0.0004
3.0	10.0	0.95	0.95	0
2.0	3.0	0.60	0.51	0.0081
5.0	1.0	0.15	0.22	0.0049
8.0	5.5	0.25	0.385	0.0182
12.0	7.5	0.23	0.325	0.0009
10.0	4.0	0.20	0.18	0.0004
3.0	2.0	0.30	0.39	0.0081

Testing set:

x_1	x_2	y	$\hat{y}$	$(y-\hat{y})^2$
1.0	9.0	0.95	0.98	0.0009
9.0	6.0	0.20	0.37	0.0289
7.0	3.0	0.25	0.26	0.0001
5.0	5.0	0.45	0.50	0.0025

SSE_train = sum of all $(y-\hat{y})^2$ = 0.041
SSE_test = sum of all $(y-\hat{y})^2$ = 0.0324
Variance = 0.041 - 0.0324 = 0.0086
Therefore the model is better fit compared to the base case.

Q3:
Find the coefficients of a polynomial with degree 2 which gives the lowest mean square error.
y_predicted = ax^2 + bx + c
x= [1,2,3]
y_actual = [4, 13, 20]
Coefficient Set 1: a= 1, b=2, c=3
Coefficient Set 2: a=3, b=5, c=2
Coefficient Set 3: a=1, b=1, c=5

Answer:
y_predicted = ax^2 + bx + c
x= [1,2,3]
y_actual = [4, 13, 20]

Calculate the y_predicted for all 3 values of X for each coefficient set.

For Coefficient set 1(a= 1, b=2, c=3):
Y_predicted when x=1 is: 1*(1*1) + 2*1 + 3 = 6
Y_predicted when x=2 is: 1*(2*2) + 2*2 + 3 = 11
Y_predicted when x=3 is: 1*(3*3) + 2*3 + 3 = 18

MSE for coefficient set 1 = ((4-6)**2 + (13-11)**2 + (20-18)**2)/3= 4

For Coefficient set 2(a=3, b=5, c=2):
Y_predicted when x=1 is: 3*(1*1) + 5*1 + 2 = 10

Y_predicted when x=2 is: 3*(2*2) + 5*2 + 2 = 24
Y_predicted when x=3 is: 3*(3*3) + 5*3 + 2 = 44

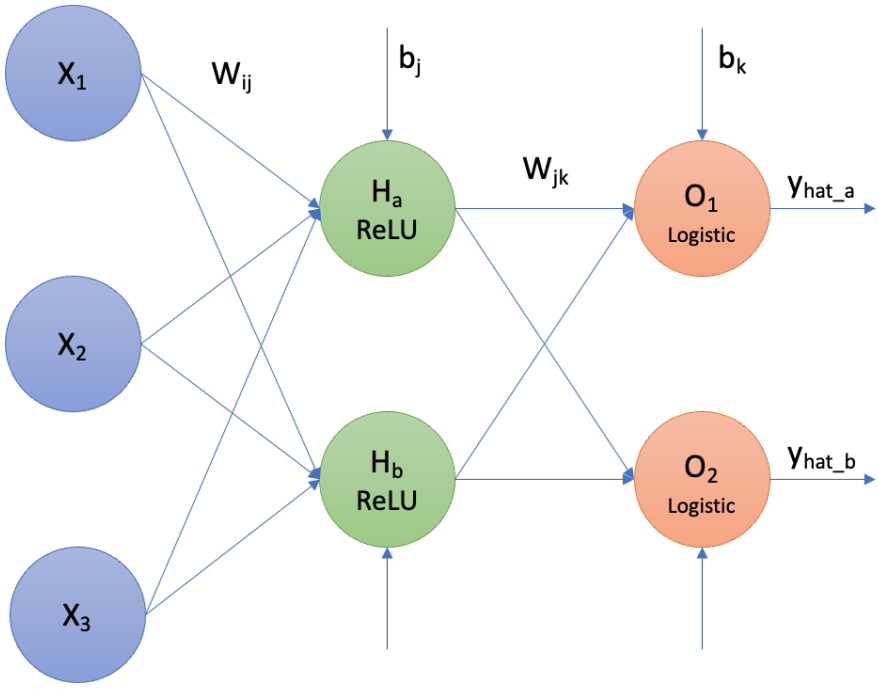
MSE for coefficient set 2 = ((4-10)**2 + (13-24)**2 + (20-44)**2)/3
= (36 + 121 + 484) / 3
= 213.66

For Coefficient set 3(a=1, b=1, c=5):
Y_predicted when x=1 is: 1*(1*1) + 1*1 + 5 = 7
Y_predicted when x=2 is: 1*(2*2) + 1*2 + 5 = 11
Y_predicted when x=3 is: 1*(3*3) + 1*3 + 5 = 17

MSE for coefficient set 2 = ((4-7)**2 + (13-11)**2 + (20-17)**2)/3
= (9 + 4 + 9) / 3
= 7.33

As we can see the lowest MSE is achieved for coefficient set 1. Therefore that is the best prediction of coefficients.

Use the graph below to answer Q4-6:



Q4:
If we designed this neural network with ReLU add after each hidden neural as activation function, and logistic(sigmoid) add in the end before we get the predicted value(y_hat), use the table below to calculate that which class does x1 = 7, x2 = 10, and x3 = 9 belongs. (Assume that as the result of one output has a value over a threshold τ = 0.9, it will be classified into that class.)

W_1a = 0.4	W_a1=0.2	b_a=0.7
W_1b =-0.7	W_a2=0.5	b_b=0.4
W_2a=0.6	W_b1=-0.1	b_1=0.9
W_2b=-0.5	W_b2=0.6	b_2=-0.8
W_3a=0.3		
W_3b=0.7		

Answer:
H_a = 7*0.4+10*0.6+9*0.3+0.7=12.2
H_b = 7*-0.7+10*-0.5+9*0.7+0.4=-3.2
O_1 = max(0,12.2)*0.2+max(0,-3.2)*-0.1+0.9=3.34
Y_hata = σ(O_1) = 1/(1+e^(-3.34)) = 0.9658
O_2 = max(0,12.2)*0.5+max(0,-3.2)*0.6+(-0.8)=5.3
Y_hatb = σ(O_2) = 1/(1+e^(-5.3)) = 0.9950
Therefore, this data point should belong to both category a and b.

Q5:
if the data points mentioned in Q7 have y_a= 1 and y_b = 0, update weight w_a1, w_a2, w_1a, and w_3a, consider the learning rate η =0.2. Assume we use SSE for the loss of this question.
Answer:

∂error/∂w_a1 = ∂error/∂y_hata*∂y_hata/∂O_1*∂O_1/∂w_a1
=-(y_a-y_hata)*y_hata*(1-y_hata)*max(0,H_a)
= -(1-0.9658)*0.9658*(1-0.9658)*max(0,12.2)
= -0.01378
W_a1new = 0.2-0.2*(-0.01378) =0.2027

∂error/∂w_a2 = ∂error/∂y_hatb*∂y_hatb/∂O_2*∂O_2/∂w_a2
=-(y_b-y_hatb)*y_hatb*(1-y_hatb)*max(0,H_a)
= -(0-0.9950)*0.9950*(1-0.9950)*max(0,12.2)

= 0.06039

W_a2new = 0.5-0.2*0.06039 =0.4879

∂error/∂w_1a = ∂error/∂ha_out*∂ha_out/∂h_a*∂h_a/∂w_1a

=(-(y_a-y_hata) * y_hata * (1-y_hata) * w_a1 + (-(y_b-y_hatb) * y_hatb * (1-y_hatb)*w_a2)*1*X1

= (-(1-0.9658)*0.9658*(1-0.9658)*0.2+-(0-0.9950)*0.9950*(1-0.9950)*0.5) *1*7

= 0.01574

W_1anew = 0.4-0.2*0.01574 =0.3969

∂error/∂w_3a = ∂error/∂ha_out*∂ha_out/∂h_a*∂h_a/∂w_3a

=(-(y_a-y_hata) * y_hata * (1-y_hata) * w_a1 + -(y_b-y_hatb) * y_hatb * (1-y_hatb)*w_a2)*1*X3

= (-(1-0.9658)*0.9658*(1-0.9658)*0.2+-(0-0.9950)*0.9950*(1-0.9950)*0.5) *1*9

= 0.02024

W_3anew = 0.3-0.2*0.02024 =0.2959

Q6:

Similarly, update all the biases.

Answer:

∂error/∂b_1 = ∂error/∂y_hata*∂y_hata/∂O_1*∂O_1/∂b_1

= -(1-0.9658)*0.9658*(1-0.9658)*1

= - 0.00112

B_1new = 0.9 - 0.2*(-0.00112) = 0.9002

∂error/∂b_2 = ∂error/∂y_hatb*∂y_hatb/∂O_2*∂O_2/∂b_2

= -(0-0.9950)*0.9950*(1-0.9950)*1

= 0.00496

B_2new = -0.8 - 0.2*(0.00496) = -0.8010

∂error/∂b_a = ∂error/∂ha_out*∂ha_out/∂h_a*∂h_a/∂b_a

= (-(1-0.9658)*0.9658*(1-0.9658)*0.2+-(0-0.9950)*0.9950*(1-0.9950)*0.5)*1*1

= 0.00225

B_anew = 0.7 - 0.2*(0.00225) = 0.6996

$\partial error/\partial b\_b = \partial error/\partial hb\_out * \partial hb\_out/\partial h\_b * \partial h\_b/\partial b\_b$

$= (-(1-0.9658)*0.9658*(1-0.9658)*(-0.1)+-(0-0.9950)*0.9950*(1-0.9950)*0.6)*0*1$

$= 0$

$B\_bnew = 0.4 - 0.2*(0) = 0.4$

Q7:

In logistic regression, what is the hypothesis function and what does the predicted output of the hypothesis function represent, given an input data point x(1)?

Answer:

The hypothesis function is a sigmoid function of the linear combination of weights and input variables:

$z(i)j = \Sigma wjxj(i) = w^T x(i)$

probability density function : $g(x(i); w) = 1/(1+ e^{(-w^T x(i))})$

The predicted output is the integral of the above probability density function for the interval $[x(1)- \epsilon , x(1)+ \epsilon]$ and represents the probability that a given input x(1) belongs to a category (such as probability of going for a hike).

Q8:

Joshua claims that in Machine Learning, most of the data are used in testing because accuracy in predicting unseen data is more important than the accuracy of the seen data in the training set. Do you agree with this claim? Justify your answer.

Answer:

Disagree because if the model doesn't work well with the current training set, there's no guarantee that it will work even better for the unseen testing set.

Q9:

Can OLS method be used to train a logistic regression model as a common practice? Explain your answer.

Answer:

Ordinary Least Squares (OLS) is a commonly used method for training linear regression models, where the goal is to predict a continuous numerical output. However, OLS is not typically used to train logistic regression models, which are used for binary or multi-class classification tasks. Here's why OLS is not a common practice for training logistic regression models:

Different Objectives:

OLS is designed to minimize the sum of squared differences between the predicted and actual numerical target values, which works well for regression problems.

Logistic regression, on the other hand, aims to model the probability of a binary outcome or a categorical outcome in multiple classes. It uses the logistic (sigmoid) function to map input features to a probability value between 0 and 1.
Different Error Metrics:

OLS minimizes the mean squared error (MSE) or similar metrics that are appropriate for regression tasks.
Logistic regression uses a different loss function, typically the logistic loss (or log loss), which is appropriate for classification problems.
Non-linearity:

Logistic regression models the relationship between the features and the binary outcome using a sigmoid (logistic) function, which introduces non-linearity into the model. This non-linearity is crucial for capturing the probability distribution and decision boundary for classification tasks. OLS, which is designed for linear regression, assumes a linear relationship between the features and the target variable, which is not suitable for capturing the non-linear nature of classification problems.
To train a logistic regression model, you typically use techniques like maximum likelihood estimation or gradient-based optimization methods (e.g., gradient descent) to find the optimal coefficients that maximize the likelihood of the observed class labels given the input features.

In summary, OLS is not a common practice for training logistic regression models because they have different objectives, error metrics, and modeling requirements. Logistic regression is specifically designed for classification tasks, and it employs a different mathematical approach to achieve this goal.

Q10:
Show mathematically how to obtain the weight of a linear regression model with attribute X using the OLS method.

Answer:
In linear regression, the goal is to find the weight (coefficients) of the model that minimizes the sum of squared differences between the predicted values and the actual target values. This can be done using the Ordinary Least Squares (OLS) method. Here's the mathematical expression for obtaining the weight of a linear regression model with attribute X using OLS:

Assuming you have a dataset with N data points, where each data point has a feature X and a target variable y, the linear regression model is represented as:

$y=b0 + b1*X$

where:
y is the target variable for a data point.

X is the feature or attribute.

b0 is the intercept (bias) term.

b1 is the weight or coefficient associated with the feature X.

The OLS method aims to find the values of b0 and b1 that minimize the sum of squared differences between the predicted values and the actual target values for all N data points. This is expressed as the loss function:
To find the optimal values of b0 and b1 , you differentiate the loss function with respect to b0 and b1 and set the derivatives equal to zero to solve for the optimal coefficients. Here are the partial derivatives:

Loss function: $L(b_0, b_1) = \sum_{i=1}^{N} (y_i - (b_0 + b_1 * X_i))^2$

Partial derivative with respect to b0
Solving this equation for b0 will give you the optimal intercept term.

Partial derivative with respect to b_1
Solving this equation for b1 will give you the optimal weight or coefficient associated with the feature X

The solutions for b0 and b1 that make the derivatives equal to zero are the values that minimize the loss function and provide the best-fitting linear regression model for the given data.
Finally:

$$w = X^T Y \, (X^T X)^{-1} = \frac{X^T Y}{X^T X}$$

Q11:
What is the number of neurons in the input layer of an ANN if the number of attributes in the dataset is 3?

Answer:
In an Artificial Neural Network (ANN), the number of neurons in the input layer is determined by the number of attributes or features in your dataset. Each attribute corresponds to one neuron in the input layer. So, if your dataset has 3 attributes, you should have 3 neurons in the input layer. Each neuron in the input layer takes one of the features as input and passes it to the subsequent layers in the neural network.

Q12:
Classify the feature based on the weight and threshold given below:

Feature 1	**Classification**
0.5	
0.7	
1.1	
1.5	
1.3	

The weight is as follows:
Weight combination 1: w1 = 0.5.
Threshold: t = 0.6
Use the sigmoid function to compute the probability.

Answer:

Feature 1	**Classification**
0.5	0
0.7	0
1.1	1
1.5	1
1.3	1

The weighted sum for Weight combination 1 is:
weighted sum = w1 * x
Where w1 = 0.5.
Using the sigmoid function:

sigmoid(x) = 1 / (1 + exp(-x))

We can compute the weighted sum and probability of classification as follows:

Observation 1:
weighted sum = w1 * x1 = 0.5 * 0.5 = 0.25
sigmoid(weighted sum) = 1 / (1 + exp(-0.25)) = 0.562

Observation 2:
weighted sum = w1 * x1 = 0.5 * 0.7 = 0.35
sigmoid(weighted sum) = 1 / (1 + exp(-0.35)) = 0.588

Observation 3:
weighted sum = w1 * x1 = 0.5 * 1.1 = 0.55
sigmoid(weighted sum) = 1 / (1 + exp(-0.55)) = 0.634

Observation 4:
weighted sum = w1 * x1 = 0.5 * 1.5 = 0.75
sigmoid(weighted sum) = 1 / (1 + exp(-0.75)) = 0.679

Observation 5:
weighted sum = w1 * x1 = 0.5 1.3 = 0.65
sigmoid(weighted sum) = 1 / (1 + exp(-0.65)) = 0.658

If the sigmoid(weighted sum) >= t, classify as 1 (positive)
If the sigmoid(weighted sum) < t, classify as 0 (negative)

Q13:
Given a dataset D that has 6 data points shown in the table below, you are going to develop a least mean square regression model in the form of $\hat{y} = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2$ where the weights $W = [w_0, w_1, w_2]$ makes the model have minimum error.

x_1	x_2	y
4	1	2
2	8	-14
1	0	1
3	2	-1
1	4	-7

6	7	-8

Use the gradient descent method to update the weights W of the regression model. The current weights are $W^t = [0.1, 0.2, 0.3]$, learning right $\lambda = 0.02$.
Include all your calculations for one round of weight updates. Round to 2 decimal places.
Hint: use the formula:

$$\frac{\partial J}{\partial w_j} = -\sum_{i=1}^{n} (y_i - \hat{y}_i) x_{ij}$$

to calculate the gradient of the cost function and to update the weights use the formula:

$$W^{t+1} = W^t - \lambda \cdot \nabla J(W^t)$$

Answer:

$\hat{y} = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 = [1.2, 2.9, 0.3, 1.3, 1.5, 3.4]$

$\frac{\partial J}{\partial w_0} = -\sum_{i=1}^{n} (y_i - \hat{y}_i) = 37.60$

$$\frac{\partial J}{\partial w_1} = -\sum_{i=1}^{n} (y_i - \hat{y}_i) \cdot x_{i1} = 113.70$$

$$\frac{\partial J}{\partial w_2} = -\sum_{i=1}^{n} (y_i - \hat{y}_i) \cdot x_{i2} = 252.80$$

$$\nabla J(W^t) = \left[\frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}\right] = [37.60, 113.70, 252.80]$$

$$W^{t+1} = W^t - \lambda \cdot \nabla J(W^t) = [-0.65, -2.07, -4.76]$$

Q14:
What is the difference between Batch GD (Gradient Descent) and Stochastic GD? Why is Stochastic more widely used compared to Batch GD and Newton's method?
Answer:
BGD uses the entire dataset to compute gradients in each iteration.
SGD uses one random data point in each iteration.
BGD is computationally expensive, especially for large datasets.
SGD is more efficient as it processes only one data point at a time.
Newton's method can be very computationally expensive due to Hessian computations.
Therefore, SGD is more widely used than BGD and Newton's method because it strikes a balance between efficiency and effectiveness.

Q15:
In batch gradient descent, if the number of batches is equal to the number of observations in the training dataset, the gradient descent approach is the same as “Stochastic gradient descent ”.

Answer: True

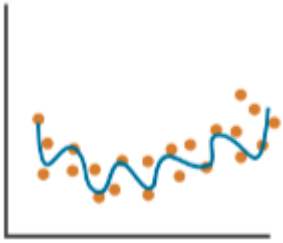
Q16:
In the context of training artificial neural networks, which of the following best describes the role of gradient descent?
a) It's a type of activation function applied to the neurons.
b) It's the process of adding layers to the neural network.
c) It's an optimization algorithm used to minimize the error by adjusting the weights.
d) It's a method to regularize the network and prevent overfitting.

Answer: c) It's an optimization algorithm used to minimize the error by adjusting the weights.

Q17:
Which of the following statements about L1 and L2 regularization is correct?
A. L1 regularization adds the squared value of the loss to the weight update
B. L2 regularization adds the absolute value of the weight to the loss function
C. L1 regularization adds the absolute value of the weight to the loss function
D. L2 regularization adds the squared value of the loss to the weight update

Answer: C

Q18:
Ture or False: This graph on the right is an example of overfitting



Answer: True

Q19:
Ture or False: Newton's method is a method that completely outperforms gradient descent in any setting because it can always find the minimum loss function value with fewer weight updates.
Answer: False

Q20:
Which of the following is the correct formula to find the weight of a linear regression model with the OLS method.
A. $X^T Y (X^T X)^{-1}$
B. $Y^T Y (X^T X)^{-1}$
C. $X^T X (X^T X)^{-1}$
D. $X^T X (X^T Y)^{-1}$

Answer: A

Q21:
A company conducted a study to analyze the performance of two machine learning models, Model A and Model B, on a dataset of 500 instances. The confusion matrices for both models are as follows:

Model A:	Predicted Positive	Predicted Negative
Actual Positive	50	10
Actual Negative	20	420

Model B:	Predicted Positive	Predicted Negative
Actual Positive	60	40
Actual Negative	30	370

Using these confusion matrices, calculate and compare the following performance metrics for Model A and Model B: Accuracy, Precision for the positive class, Recall (Sensitivity) for the positive class, F1 score for the positive class.

Based on these metrics, determine which model (A or B) performed better on the dataset

Answer:

Accuracy:

For Model A: (TP + TN) / (TP + TN + FP + FN) = (50 + 420) / (50 + 10 + 20 + 420) = 470 / 500 = 0.94 or 94%.

For Model B: (TP + TN) / (TP + TN + FP + FN) = (60 + 370) / (60 + 40 + 30 + 370) = 430 / 500 = 0.86 or 86%.

Precision for the positive class:

For Model A: TP / (TP + FP) = 50 / (50 + 20) = 50 / 70 = 0.7143 or 71.43%.

For Model B: TP / (TP + FP) = 60 / (60 + 30) = 60 / 90 = 0.6667 or 66.67%.

Recall (Sensitivity) for the positive class:

For Model A: TP / (TP + FN) = 50 / (50 + 10) = 50 / 60 = 0.8333 or 83.33%.

For Model B: TP / (TP + FN) = 60 / (60 + 40) = 60 / 100 = 0.6 or 60%.

F1 score for the positive class:

For Model A: 2 * (precision * recall) / (precision + recall) = 2 * (0.8333 * 0.7143) / (0.8333 + 0.7143) = 0.7692 or 76.92%.

For Model B: 2 * (precision * recall) / (precision + recall) = 2 * (0.6 * 0.6667) / (0.6 + 0.6667) = 0.6327 or 63.27%.

Based on these metrics, we can conclude that Model A performed better than Model B on the dataset. Model A achieved higher accuracy, precision, recall, and F1 score for the positive class compared to Model B.