Question 2a Recall the optimal value of θ should minimize our loss function. One way we've approached solving for θ is by taking the derivative of our loss function with respect to θ, like we did in HW5.

In the space below, use LaTeX to write/compute the following values: * $R(\mathbf{x}, \mathbf{y}, \theta_1, \theta_2)$: our loss function, the empirical risk/mean squared error * $\frac{\partial R}{\partial \theta_1}$: the partial derivative of R with respect to θ_1 * $\frac{\partial R}{\partial \theta_2}$: the partial derivative of R with respect to θ_2

Recall that $R(\mathbf{x}, \mathbf{y}, \theta_1, \theta_2) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2$

$R(\mathbf{x}, \mathbf{y}, \theta_1, \theta_2) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{y}_i - \hat{\mathbf{y}}_i)^2$ $\hat{\mathbf{y}} = \theta_1 x + sin(\theta_2 x)$ $R(\mathbf{x}, \mathbf{y}, \theta_1, \theta_2) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{y}_i - (\theta_1 x + sin(\theta_2 x)))^2$ $\frac{\partial R}{\partial \theta_1} = \frac{1}{n} \sum_{i=1}^{n} 2(-x_i)(\mathbf{y}_i - \theta_1 x - sin(\theta_2 x))$ $\frac{\partial R}{\partial \theta_2} = \frac{1}{n} \sum_{i=1}^{n} 2(\mathbf{y}_i - \theta_1 x - sin(\theta_2 x)) * (-x_i cos(\theta_2 x_i))$

In 1-2 sentences, describe what you notice about the path that theta takes with a static learning rate vs. a decaying learning rate. In your answer, refer to either pair of plots above (the 3d plot or the contour plot).

The path that theta takes with a static learning rate tends to jump back and forth between the optimal value of theta until it converges towards it (going above, going below, etc.) as seen in the Gradient Descent with Static Learning Rate Contour Plot. This varies from the Gradient Descent with Decay Learning Rate because with a decaying learning rate, theta makes an initial leap from the initial guess and slowly converges and follows a smooth path towards the optimal value of theta, instead of jumping back and forth.

0.0.1 Question 4b

Is this model reasonable? Why or why not?

The model is not reasonable because it suggests that a team that scored 0 or more points will have at least a 0.5 probability of winning. This is problematic because even teams that score 0 points will have a 50% chance of winning, and this would result in every single team having a probability of at least 0.5 of winning.

0.0.2 Question 4c

Try playing around with other theta values. You should observe that the models are all pretty bad, no matter what θ you pick. Explain why below.

All the observations perform poorly because we predict the probability of winning based on the number of points scored, which is always greater than or equal to zero. Plugging in zero into the sigmoid function, we get $\frac{1}{1+e^0} = 0.5$*. So, no matter what theta we choose, every team will always have a probability of at least 0.5*

0.0.3 Question 5b

Using the plot above, try adjusting θ_2 (only). Describe how changing θ_2 affects the prediction curve. Provide your description in the cell below.

Adjusting θ_2 *allows us to vertically shift the prediction curve up or down. Increasing* θ_2 *allows us to move the graph up, and decreasing lowers it.*

0.0.4 Question 7c

Look at the coefficients in `theta_19_hat` and identify which of the parameters have the biggest effect on the prediction. For this, you might find useful_numeric_fields.columns useful. Which attributes have the biggest positive effect on a team's success? The biggest negative effects? Do the results surprise you?

The attributes with the biggest positive effect on team success is the BIAS and FG3_PCT. The attributes with the biggest negative effect on team success if FG_PCT and PTS. The results for the attributes that contribute the most to negative effects surprised me because as a basketball fan myself, I thought high field goal percentage and points would always contribute to helping a team win a game and not losing one. What was also surprising for me was the bias was the most impactful in terms of positive effect on a team's success. I thought the bias would not contribute much, but it ended up having the most assigned weight out of all the useful numeric fields.

To double-check your work, the cell below will rerun all of the autograder tests.

```
In [65]: grader.check_all()
```

```
Out[65]: q1:

             All tests passed!


         q2b:


             0 of 1 tests passed


             Tests failed:

                     ./tests/q2b.py

         Test result:
         Trying:
             abs(sin_MSE([0, np.pi]) - 19.49000412080223) <= 1e-5
         Expecting:
             True
         **********************************************************************
         Line 1, in ./tests/q2b.py 0
         Failed example:
             abs(sin_MSE([0, np.pi]) - 19.49000412080223) <= 1e-5
         Expected:
             True
         Got:
```

```
    False




q3a:

    All tests passed!


q3b:

    All tests passed!


q4a:

    All tests passed!


q5a:

    All tests passed!


q5c:

    All tests passed!


q6a:

    All tests passed!


q6b:

    All tests passed!


q6c:

    All tests passed!


q6d:

    All tests passed!


q7a:
    All tests passed!


q7b:

    All tests passed!
```

0.1 Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zipfile for you to submit. **Please save before exporting!**

```
In [66]: # Save your notebook first, then run this cell to export your submission.
         grader.export()
```

```
<IPython.core.display.HTML object>
```