|  |  |
| --- | --- |
|  | P(happy) |
| Happy? = yes | 1/2 |
| Happy? = no | 1/2 |

|  |  |  |
| --- | --- | --- |
|  | year = freshman | grade = A |
| Happy? = yes | 1/3 | 2/3 |
| Happy? = no | 1/2 | 1/3 |

y\_samp = random()

year\_samp = random()

grade\_samp = random()

If y\_samp > P(Happy? = No):

Happy? = Yes

Else:

Happy? = No

If year\_samp > P(freshman | Happy?):

Year = senior

else:

Year = freshman

If grade\_samp > P(A | Happy?):

Grade = C

else:

Grade = A

return (Grade, Year, Happy?)

We have

So,

With

So

=

Compared to the standard ridge regression, we have the coefficient to consider. Because we have scaled x, our w will also scale since the summation part of the equation is minimized. This is because y will stay the same even with the other scaling. Therefore, the main difference with the rescaling transformation is the term which deals with the rescaling.

Preprocessing:

I made a couple changes to the data during the preprocessing stage. I eliminated columns that had zero variance, like the first date columns that were all the same (Year/Day/Hour). I tried to normalize the columns that had extreme values. For example, the entering\_host column had some extreme values along with a lot of missing values. So, I first filled the missing values with mean of the column values that were present, and I then normalized the entire column. Therefore, I dealt with columns with zero variance, the missing values, and extreme values.

Models:

I tried a couple of different models before coming to XGBoost. I initially tried random forests because I thought it would deal well with the large number of variables. I tested different parameters using MSE as validation, but I was not getting any good results. I decided to move on and try an out-of-the-box method: elastic net. I honestly have never used this before, but it popped up on the sklearn website. I gave it a couple tries, but the MSE was way too high. I finally came to my final model of XGBoost. Since this model performs well on Kaggle competitions, I thought I should use it. I tested parameters and tried to run a random grid, but my computer is too slow/I had other finals I had to run stuff for. This ultimately became my best model, which achieved my highest score.

Validation:

I used MSE as my main source of validation. I split the training data and ran different tests on each part of the training data. This gave me a better feel of how the models would perform. I used a random grid that split the data up many different times and tested different combinations of parameters on these different splits.

Committed Model:

When I first started using XGBoost, I achieved a good score on Kaggle, but I realized that I didn’t use the data that had the preprocessing done to it. It was my highest score overall, so I used that as one of my submissions. I was a little skeptical about the situation, since I didn’t touch the data, so I used the highest score XGBoost with preprocessing that I had as my second submission. Therefore, if there were any large changes to the private leaderboard, I would have that aspect covered.

Other Notable Procedures:

Nothing else, pretty much all I did was described above.