IA2

Yongsung Cho

Computer Science, Oregon State University

AI 534: Machine Learning

Prof. Fern

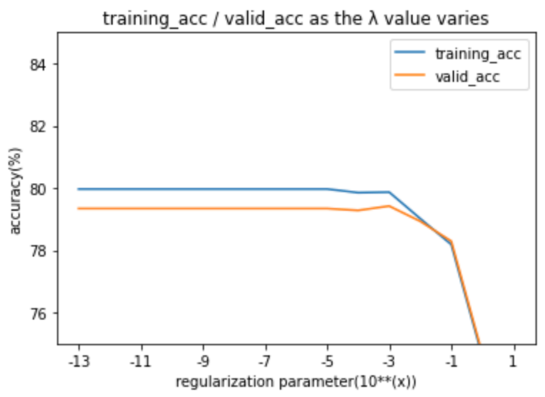
Due Oct 29, 2021

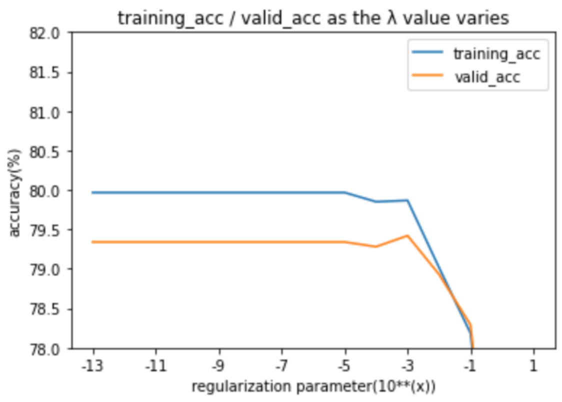
Preprocessing

Most features were one-hot encoding, so no other preprocessing was required. However, normalization was needed because Age, Annual_Premium, and Vintage are numeric numbers. I did normalization and training in the same way that we did in IA1.

Part 1 Logistic regression with L2 (Ridge) regularization

(a) Plot the training accuracy and validation accuracy of the learned model as the λ value varies.

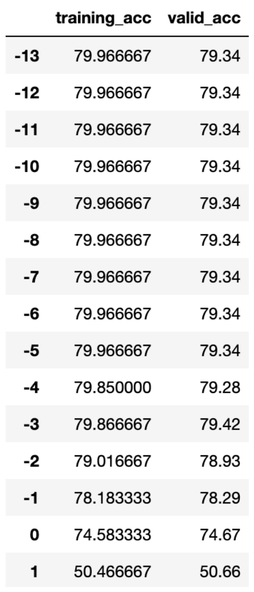




The two plots shown above are the same graph. I adjusted the range to ensure that validation accuracy and training accuracy are higher in each normalization parameter. I trained parameters with 10^2 and 10^3, but it was excluded from the graph because it did not converge.

Question: what trend do you observe for the training accuracy as we increase λ? Why is this the case? What trend do you observe for the validation accuracy? What is the best λ value based on the validation accuracy?

Training accuracy is almost unchanged when λ is very small (10^{-13} ~ 10^{-4}), because it is rarely applied to regularization. However, the λ increases slightly when it is 10^{-3}, and since then, the training accuracy and validation accuracy tend to decrease due to too large λ. If it was too large, the regularization would be too excessive, which would have affected the overall accuracy. Validation accuracy appears to be similar to training accuracy. Like training accuracy, when it is a very small regularization parameter, it peaks at 10^{-3} and tends to decrease gradually when there is little change.



	training_acc	valid_acc
-13	79.966667	79.34
-12	79.966667	79.34
-11	79.966667	79.34
-10	79.966667	79.34
-9	79.966667	79.34
-8	79.966667	79.34
-7	79.966667	79.34
-6	79.966667	79.34
-5	79.966667	79.34
-4	79.850000	79.28
-3	79.866667	79.42
-2	79.016667	78.93
-1	78.183333	78.29
0	74.583333	74.67
1	50.466667	50.66

This table represents training accuracy and validation accuracy for each parameter. When the corresponding to this table, the parameter is 10^{-3}, the best accuracy is shown.

(b) Consider the best λ∗ selected in (a), a value λ− that is smaller than λ∗, and a λ+ that is bigger than λ∗. Report for each of three λ values, the resulting model's top 5 features with the largest weight magnitude |wj|.

best 10^{-3}

1: Previously_Insured
2: Vehicle_Damage
3: dummy
4: Policy_Sales_Channel_160
5: Region_Code_3
6: Policy_Sales_Channel_26

best+ (larger) 10^{-2}
1: Vehicle_Damage
2: Previously_Insured
3: dummy
4: Policy_Sales_Channel_26
5: Policy_Sales_Channel_152
6: Policy_Sales_Channel_160

best_ (smaller) 10^{-4}
1:Previously_Insured
2:Vehicle_Damage
3:dummy
4:Region_Code_3

5:Driving_License
6:Policy_Sales_Channel_26

=============Additional check====================
best_ _ / best++

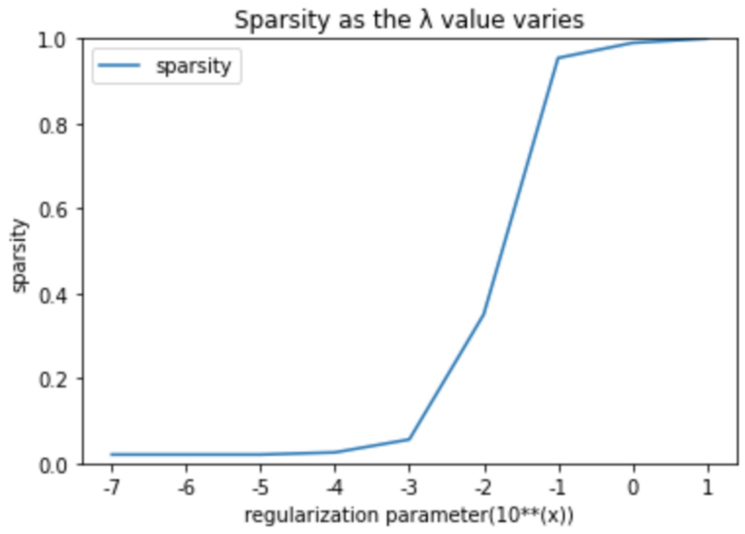
10^{-1}
1:Vehicle_Damage
2:Previously_Insured
3:dummy
4:Policy_Sales_Channel_152
5:Vehicle_Age_1
6:Vehicle_Age_0

10^{-5}
1:Previously_Insured
2:Vehicle_Damage
3:dummy
4:Region_Code_3
5:Region_Code_34
6:Policy_Sales_Channel_157

Question: Do you see differences in the selected top features with different λ values? What is your explanation for this behavior?

There is no significant change in the top three features; only the order of Vehicle_Damage and Previously_Insured occasionally changes. In all λ, third place is a dummy, but this feature is not meaningful, so it is excluded from the explanation. The lower three features change slightly depending on the size of the λ. Among them, the two channels have a more significant effect on the response determination than other features, as the Policy_Sales_Channel_26 and Policy_Sales_Channel_160 are frequently seen. The similarities of the top five features suggest that Accuracy could be the reason to stay at 79%.

(c) For different values of λ, compute the sparsity of the model as the number of weights that equal zero and plot it against λ.



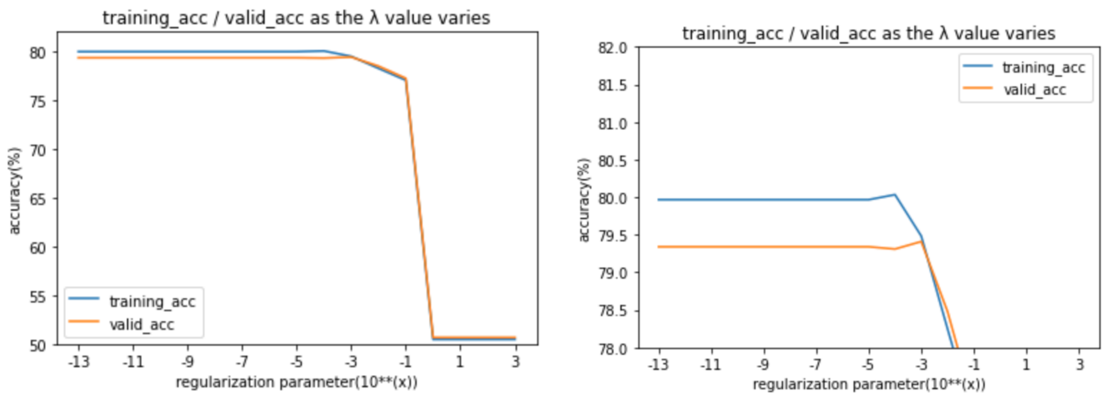
In L2 regularization, the threshold was set to 0.1 because the feature was not converged to zero.

Question: What trend do you observe for the sparsity of the model as we change λ? If we further increase λ, what do you expect? Why?

As a result, the sparsity shows a small sparsity in a small regularization parameter: there are few weight vector elements close to zero.However, as the parameter increases, the sparsity increases. And the more λ increases, the sparsity increases.

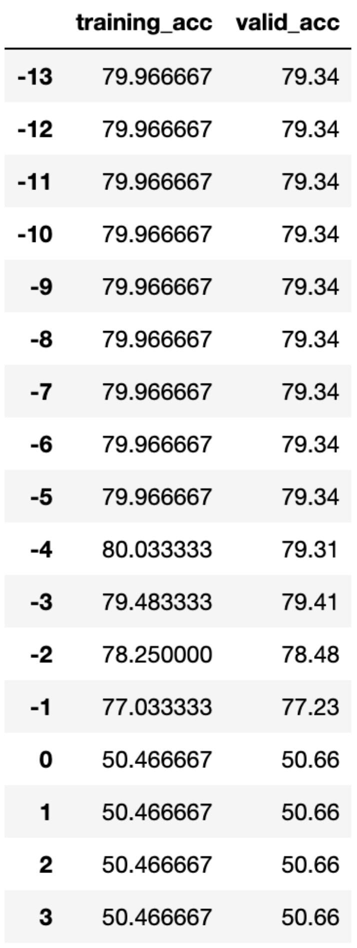
Part 2 Logistic Regression with L1 (Lasso) regularization

(a)



Question: what trend do you observe for the training accuracy as we increase λ? Why is this the case? What trend do you observe for the validation accuracy? What is the best λ value based on the validation accuracy?

Similar to L2 regularization, in the small λ, there is little change in the increase of λ, but when λ is 10^{-4} and 10^{-3}, it increases slightly, and the accuracy decreases since then. Validation accuracy also shows a graph of a similar form to training accuracy.



	training_acc	valid_acc
-13	79.966667	79.34
-12	79.966667	79.34
-11	79.966667	79.34
-10	79.966667	79.34
-9	79.966667	79.34
-8	79.966667	79.34
-7	79.966667	79.34
-6	79.966667	79.34
-5	79.966667	79.34
-4	80.033333	79.31
-3	79.483333	79.41
-2	78.250000	78.48
-1	77.033333	77.23
0	50.466667	50.66
1	50.466667	50.66
2	50.466667	50.66
3	50.466667	50.66

Here is the training accuracy and validation accuracy by regularization parameter. According to validation accuracy, 10^{-3} shows best performance with 79.34% accuracy.

(b) Report for each of three λ values, the resulting model's top 5 features with the largest weight magnitude |wj|.

# best 10^{-3}

# 1:Previously_Insured
# 2:Vehicle_Damage
# 3:dummy
# 4:Policy_Sales_Channel_160
# 5:Region_Code_3
# 6:Policy_Sales_Channel_157

# best+ (larger) 10^{-2}

# 1:Previously_Insured
# 2:Vehicle_Damage
# 3:dummy
# 4:Policy_Sales_Channel_152
# 5:Vehicle_Age_1
# 6:Region_Code_28

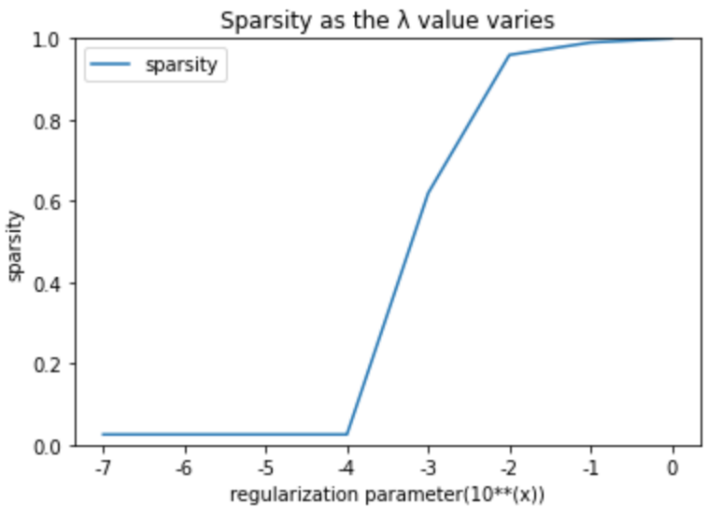
# best_ (smaller) 10^{-4}

# 1:Previously_Insured
# 2:Vehicle_Damage
# 3:dummy
# 4:Region_Code_3
# 5:Region_Code_34
# 6:Policy_Sales_Channel_157

Question: Do you see differences in the selected top features with different λ values? What is your explanation for this behavior?

The top three features did not change as λ changed. Like the one shown in L2 regularization, the top three features were Previously_Insured, Vehicle_Damage, and Dummy. However, the bottom three features changed slightly as the λ changed. In the Larger parameter, all three of the lower features were changed. But in the smaller parameter, the weight of the features is similar to the best parameter. In this behavior, Previously_Insured, Vehicle_Damage plays a significant role in determining the response.

(c) For different values of λ, compute the 'sparsity' of the model as the number of weights that equal zero and plot it against λ.



Question: What trend do you observe for the sparsity of the model as we change λ? If we further increase λ, what do you expect? Is this trend different from what you observed in 1(c)? Provide your explanation for your observation.

As λ increases, sparsity also increases; if λ increases, sparsity is expected to increase even more. These trends are similar to what they have seen in 1(c). However, in L1 regularization, the sparsity increases rapidly when the λ is 10^{-4} or more, which is faster than the L2 regularization. In addition, the threshold was applied to L2 regularization because the weight did not converge to zero. But the weight with less influence was zero in L1 regularization. So, I don't need to set a threshold in L1 regularization.

Part 3 Kaggle competition

For this part, you need to describe the methods you used and their performance on the public leaderboard. How do you handle the data usage? Do you treat the features differently from what was used for part I and II? What do you think is limiting the performance you can get on this dataset? The amount of data? the availability of features? the complexity of the algorithm? ...

Regularization parameter: 10^{-3}
Learning rate: 10^{-1}
(Based on Part 1 & 2)

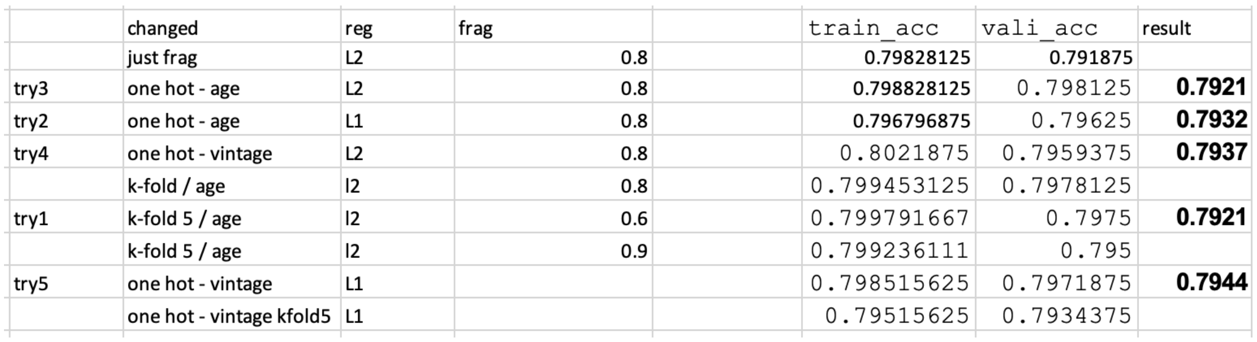
First, I thought the number of training data was too small. Validation data had 10,000 rows, but training data was only 6,000. Therefore, I have divided the training and validation sets with 16000 rows by combining training and validation data.

Second, I trained the age and vintage features into one-hot encoding. "Age" divided into ten and created columns, and the "Vintage" changed column for each value. However, the disadvantage of this method is that the number of features increases, and it takes a long time to calculate. When I did one-hot encode, "Vintage," the number of features increased to 492, so it took a long time to train.

Third, I tried to apply a k-fold test/validation dataset. I randomly separated 16000 rows into the test set and validation set. And then, I tried five times randomly separate and trained weight vector. However, the result shows that it doesn't work efficiently. Training accuracy and validation accuracy didn't increase significantly.

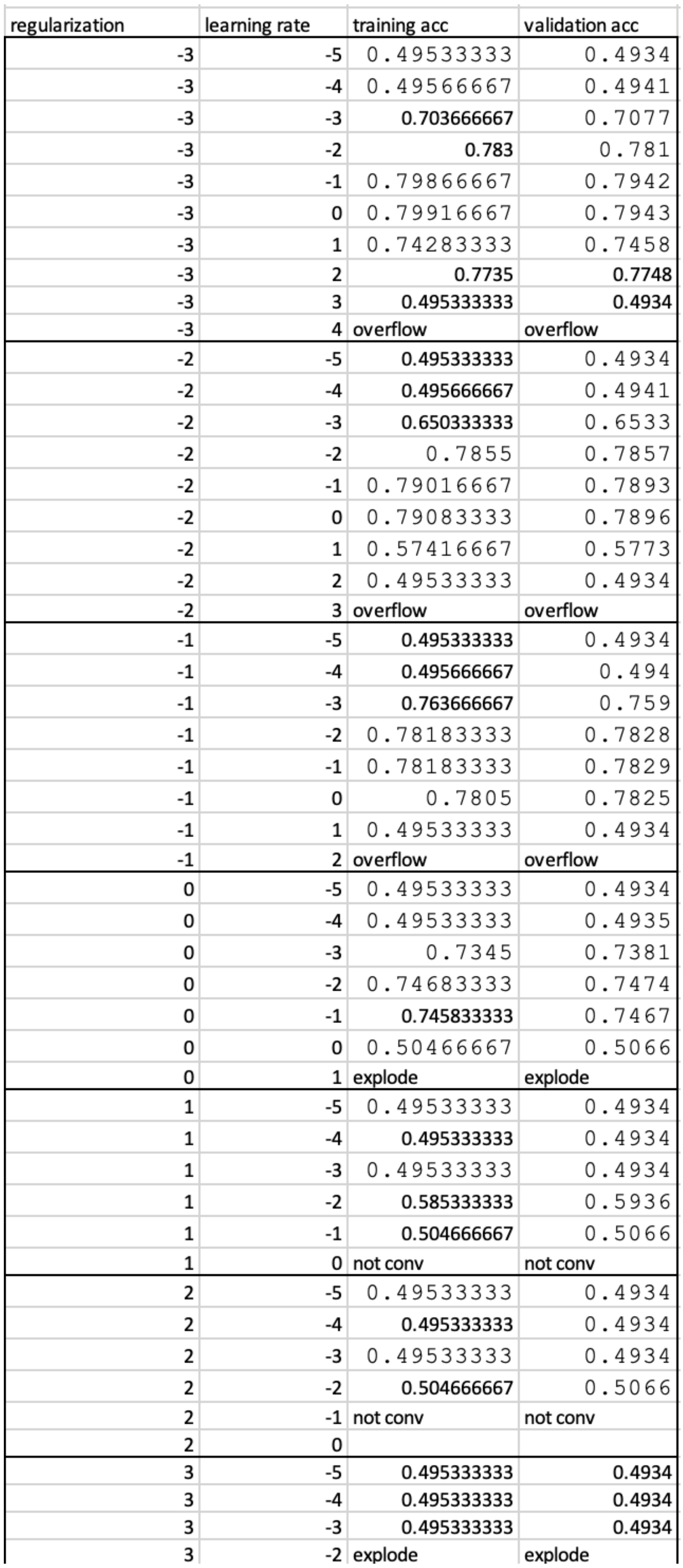
Despite the three new attempts, the accuracy does not increase significantly, so I think that the dataset does not contain meaningful data. Or, there is a possibility that the accuracy does not rise above the limit due to the algorithm's limit. According to the Additional part, L2 regularization showed higher accuracy than L1 regularization result. However, as a result of testing in Kaggle, sometimes L1 regularization shows higher accuracy than L2 regularization.

In conclusion, the best result I got is try5 which consists of 492 weight vectors with one-hot encoded age and vintage columns and trained with L1 regularization.

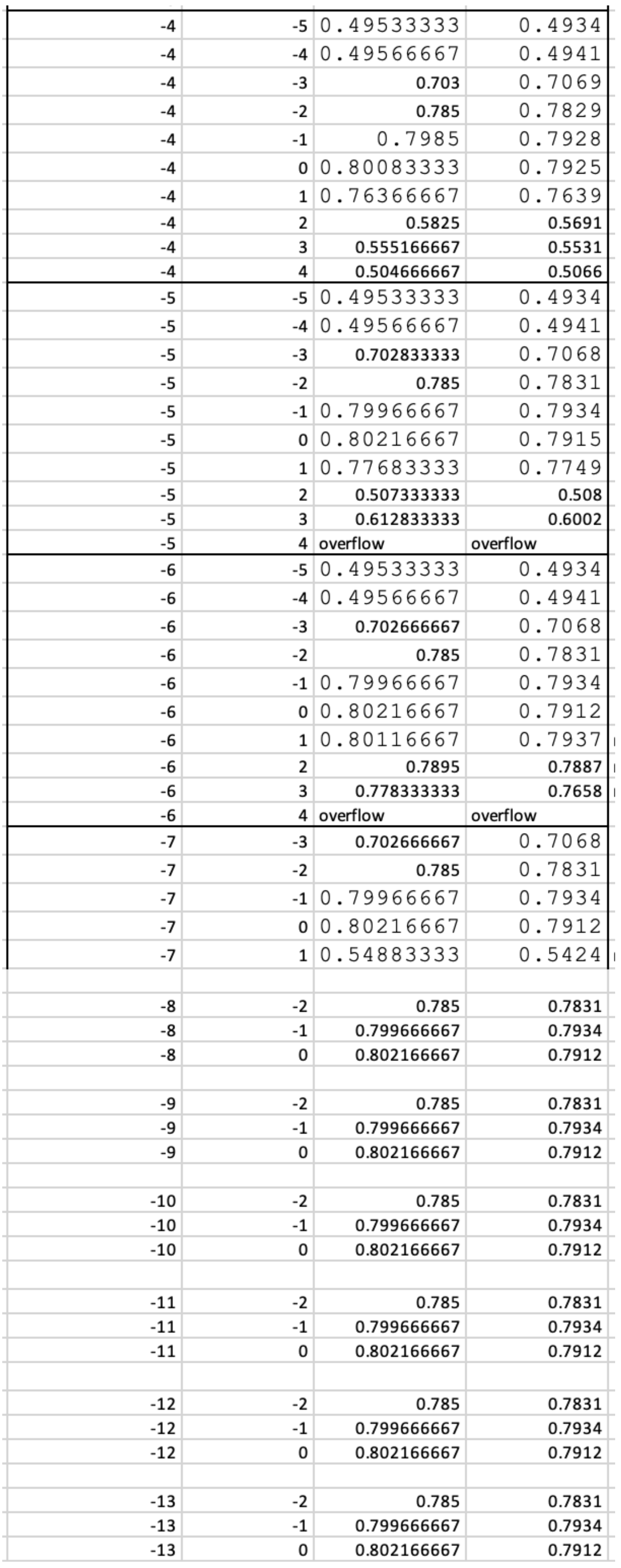


	changed	reg	frag		train_acc	vali_acc	result
	just frag	L2	0.8		0.79828125	0.791875	
try3	one hot - age	L2	0.8		0.798828125	0.798125	**0.7921**
try2	one hot - age	L1	0.8		0.796796875	0.79625	**0.7932**
try4	one hot - vintage	L2	0.8		0.8021875	0.7959375	**0.7937**
	k-fold / age	l2	0.8		0.799453125	0.7978125	
try1	k-fold 5 / age	l2	0.6		0.799791667	0.7975	**0.7921**
	k-fold 5 / age	l2	0.9		0.799236111	0.795	
try5	one hot - vintage	L1			0.798515625	0.7971875	**0.7944**
	one hot - vintage kfold5	L1			0.79515625	0.7934375	

Additional Part # / L2 regularization result

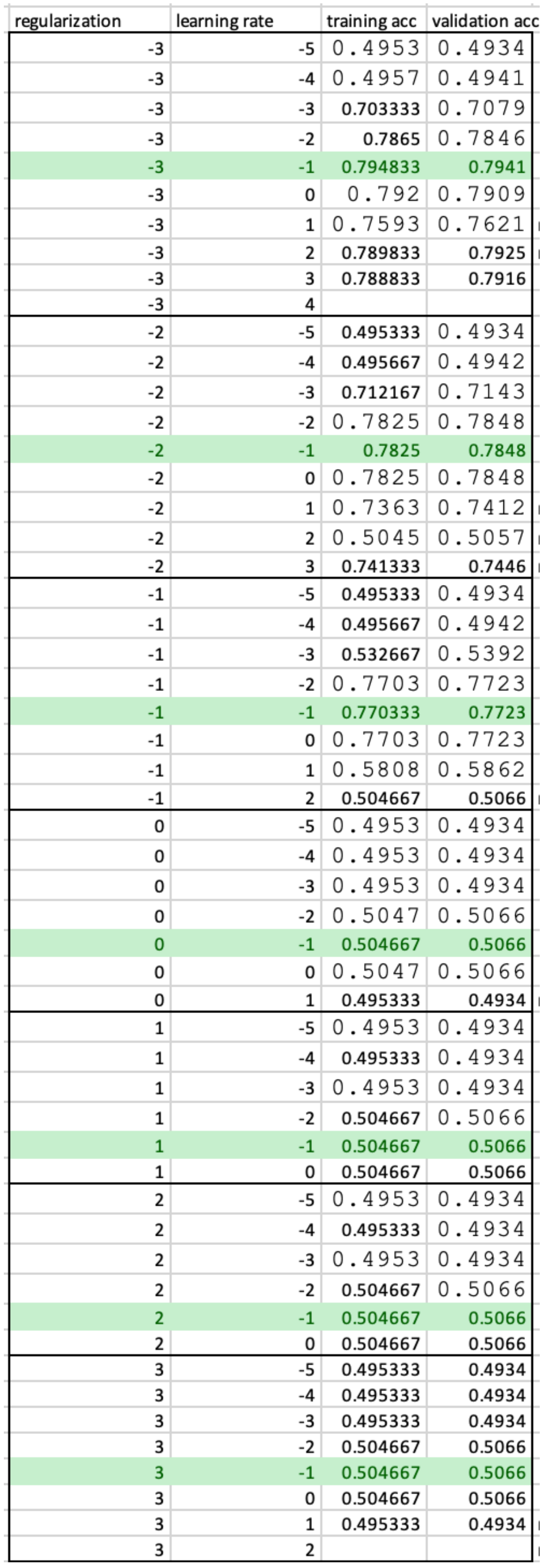


regularization	learning rate	training acc	validation acc
-3	-5	0.49533333	0.4934
-3	-4	0.49566667	0.4941
-3	-3	0.703666667	0.7077
-3	-2	0.783	0.781
-3	-1	0.79866667	0.7942
-3	0	0.79916667	0.7943
-3	1	0.74283333	0.7458
-3	2	0.7735	0.7748
-3	3	0.495333333	0.4934
-3	4	overflow	overflow
-2	-5	0.495333333	0.4934
-2	-4	0.495666667	0.4941
-2	-3	0.650333333	0.6533
-2	-2	0.7855	0.7857
-2	-1	0.79016667	0.7893
-2	0	0.79083333	0.7896
-2	1	0.57416667	0.5773
-2	2	0.49533333	0.4934
-2	3	overflow	overflow
-1	-5	0.495333333	0.4934
-1	-4	0.495666667	0.494
-1	-3	0.763666667	0.759
-1	-2	0.78183333	0.7828
-1	-1	0.78183333	0.7829
-1	0	0.7805	0.7825
-1	1	0.49533333	0.4934
-1	2	overflow	overflow
0	-5	0.49533333	0.4934
0	-4	0.49533333	0.4935
0	-3	0.7345	0.7381
0	-2	0.74683333	0.7474
0	-1	0.745833333	0.7467
0	0	0.50466667	0.5066
0	1	explode	explode
1	-5	0.49533333	0.4934
1	-4	0.495333333	0.4934
1	-3	0.49533333	0.4934
1	-2	0.585333333	0.5936
1	-1	0.504666667	0.5066
1	0	not conv	not conv
2	-5	0.49533333	0.4934
2	-4	0.495333333	0.4934
2	-3	0.49533333	0.4934
2	-2	0.504666667	0.5066
2	-1	not conv	not conv
2	0		
3	-5	0.495333333	0.4934
3	-4	0.495333333	0.4934
3	-3	0.495333333	0.4934
3	-2	explode	explode

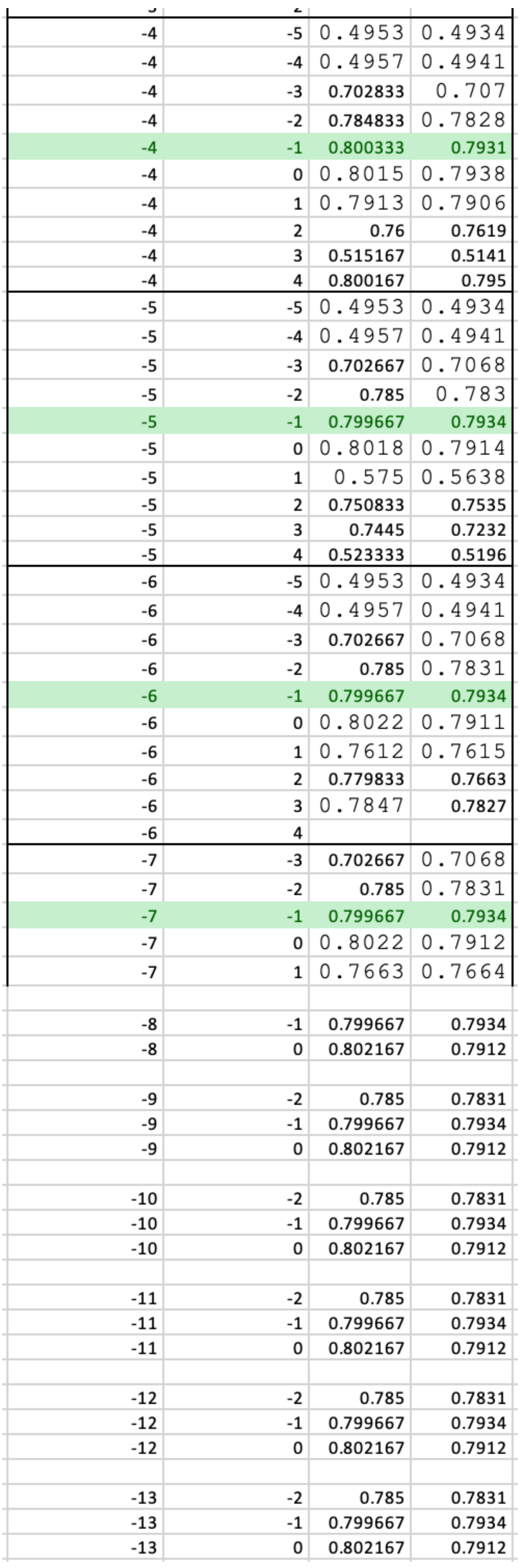


-4	-5	0.49533333	0.4934
-4	-4	0.49566667	0.4941
-4	-3	0.703	0.7069
-4	-2	0.785	0.7829
-4	-1	0.7985	0.7928
-4	0	0.80083333	0.7925
-4	1	0.76366667	0.7639
-4	2	0.5825	0.5691
-4	3	0.555166667	0.5531
-4	4	0.504666667	0.5066
-5	-5	0.49533333	0.4934
-5	-4	0.49566667	0.4941
-5	-3	0.702833333	0.7068
-5	-2	0.785	0.7831
-5	-1	0.79966667	0.7934
-5	0	0.80216667	0.7915
-5	1	0.77683333	0.7749
-5	2	0.507333333	0.508
-5	3	0.612833333	0.6002
-5	4	overflow	overflow
-6	-5	0.49533333	0.4934
-6	-4	0.49566667	0.4941
-6	-3	0.702666667	0.7068
-6	-2	0.785	0.7831
-6	-1	0.79966667	0.7934
-6	0	0.80216667	0.7912
-6	1	0.80116667	0.7937
-6	2	0.7895	0.7887
-6	3	0.778333333	0.7658
-6	4	overflow	overflow
-7	-3	0.702666667	0.7068
-7	-2	0.785	0.7831
-7	-1	0.79966667	0.7934
-7	0	0.80216667	0.7912
-7	1	0.54883333	0.5424
-8	-2	0.785	0.7831
-8	-1	0.799666667	0.7934
-8	0	0.802166667	0.7912
-9	-2	0.785	0.7831
-9	-1	0.799666667	0.7934
-9	0	0.802166667	0.7912
-10	-2	0.785	0.7831
-10	-1	0.799666667	0.7934
-10	0	0.802166667	0.7912
-11	-2	0.785	0.7831
-11	-1	0.799666667	0.7934
-11	0	0.802166667	0.7912
-12	-2	0.785	0.7831
-12	-1	0.799666667	0.7934
-12	0	0.802166667	0.7912
-13	-2	0.785	0.7831
-13	-1	0.799666667	0.7934
-13	0	0.802166667	0.7912

Additional Part # / L1 regularization result



regularization	learning rate	training acc	validation acc
-3	-5	0.4953	0.4934
-3	-4	0.4957	0.4941
-3	-3	0.703333	0.7079
-3	-2	0.7865	0.7846
-3	-1	0.794833	0.7941
-3	0	0.792	0.7909
-3	1	0.7593	0.7621
-3	2	0.789833	0.7925
-3	3	0.788833	0.7916
-3	4		
-2	-5	0.495333	0.4934
-2	-4	0.495667	0.4942
-2	-3	0.712167	0.7143
-2	-2	0.7825	0.7848
-2	-1	0.7825	0.7848
-2	0	0.7825	0.7848
-2	1	0.7363	0.7412
-2	2	0.5045	0.5057
-2	3	0.741333	0.7446
-1	-5	0.495333	0.4934
-1	-4	0.495667	0.4942
-1	-3	0.532667	0.5392
-1	-2	0.7703	0.7723
-1	-1	0.770333	0.7723
-1	0	0.7703	0.7723
-1	1	0.5808	0.5862
-1	2	0.504667	0.5066
0	-5	0.4953	0.4934
0	-4	0.4953	0.4934
0	-3	0.4953	0.4934
0	-2	0.5047	0.5066
0	-1	0.504667	0.5066
0	0	0.5047	0.5066
0	1	0.495333	0.4934
1	-5	0.4953	0.4934
1	-4	0.495333	0.4934
1	-3	0.4953	0.4934
1	-2	0.504667	0.5066
1	-1	0.504667	0.5066
1	0	0.504667	0.5066
2	-5	0.4953	0.4934
2	-4	0.495333	0.4934
2	-3	0.4953	0.4934
2	-2	0.504667	0.5066
2	-1	0.504667	0.5066
2	0	0.504667	0.5066
3	-5	0.495333	0.4934
3	-4	0.495333	0.4934
3	-3	0.495333	0.4934
3	-2	0.504667	0.5066
3	-1	0.504667	0.5066
3	0	0.504667	0.5066
3	1	0.495333	0.4934
3	2		



-4	-5	0.4953	0.4934
-4	-4	0.4957	0.4941
-4	-3	0.702833	0.707
-4	-2	0.784833	0.7828
-4	-1	0.800333	0.7931
-4	0	0.8015	0.7938
-4	1	0.7913	0.7906
-4	2	0.76	0.7619
-4	3	0.515167	0.5141
-4	4	0.800167	0.795
-5	-5	0.4953	0.4934
-5	-4	0.4957	0.4941
-5	-3	0.702667	0.7068
-5	-2	0.785	0.783
-5	-1	0.799667	0.7934
-5	0	0.8018	0.7914
-5	1	0.575	0.5638
-5	2	0.750833	0.7535
-5	3	0.7445	0.7232
-5	4	0.523333	0.5196
-6	-5	0.4953	0.4934
-6	-4	0.4957	0.4941
-6	-3	0.702667	0.7068
-6	-2	0.785	0.7831
-6	-1	0.799667	0.7934
-6	0	0.8022	0.7911
-6	1	0.7612	0.7615
-6	2	0.779833	0.7663
-6	3	0.7847	0.7827
-6	4		
-7	-3	0.702667	0.7068
-7	-2	0.785	0.7831
-7	-1	0.799667	0.7934
-7	0	0.8022	0.7912
-7	1	0.7663	0.7664
-8	-1	0.799667	0.7934
-8	0	0.802167	0.7912
-9	-2	0.785	0.7831
-9	-1	0.799667	0.7934
-9	0	0.802167	0.7912
-10	-2	0.785	0.7831
-10	-1	0.799667	0.7934
-10	0	0.802167	0.7912
-11	-2	0.785	0.7831
-11	-1	0.799667	0.7934
-11	0	0.802167	0.7912
-12	-2	0.785	0.7831
-12	-1	0.799667	0.7934
-12	0	0.802167	0.7912
-13	-2	0.785	0.7831
-13	-1	0.799667	0.7934
-13	0	0.802167	0.7912