Optimization and hyperparameter values

In order to find the optimal number of iterations for the Gradient Descent algorithm we used various. All results are demonstrated below. While all were useful and assisted us to choose the most optimal values, the most helpful one was to create a grid and perform grid search. The grid creation function is an iterative function that performs n*m*k calculations, where n = data split options (in our code – 10:90, 25:75, 30:70 and 50:50) to train and validation date out of the total train dataset, m = number of iterations options (in our code – 10, 100, 200, 300, 400, 600, 800, 1,000, 1,100, 1,500 and 10,000) and k = number of different learning rates (in our code – 0.1, 0.3, 0.7, 0.01, 0.03, 0.07, 0.001, 0.003, 0.007 and 0.0001), calculating the model's accuracy when re-training the model using each set of hyperparameters. To determine those inspected values, we performed some manual tests before to find a range of reasonable values, while starting with edge-values – very big and very small for each parameter and adapting.

After inspecting the results, we found the optimal values in terms of accuracy (both train and validation data accuracy to avoid overfitting to the train data) and performance cost.

Those values are:

Low resource usage:

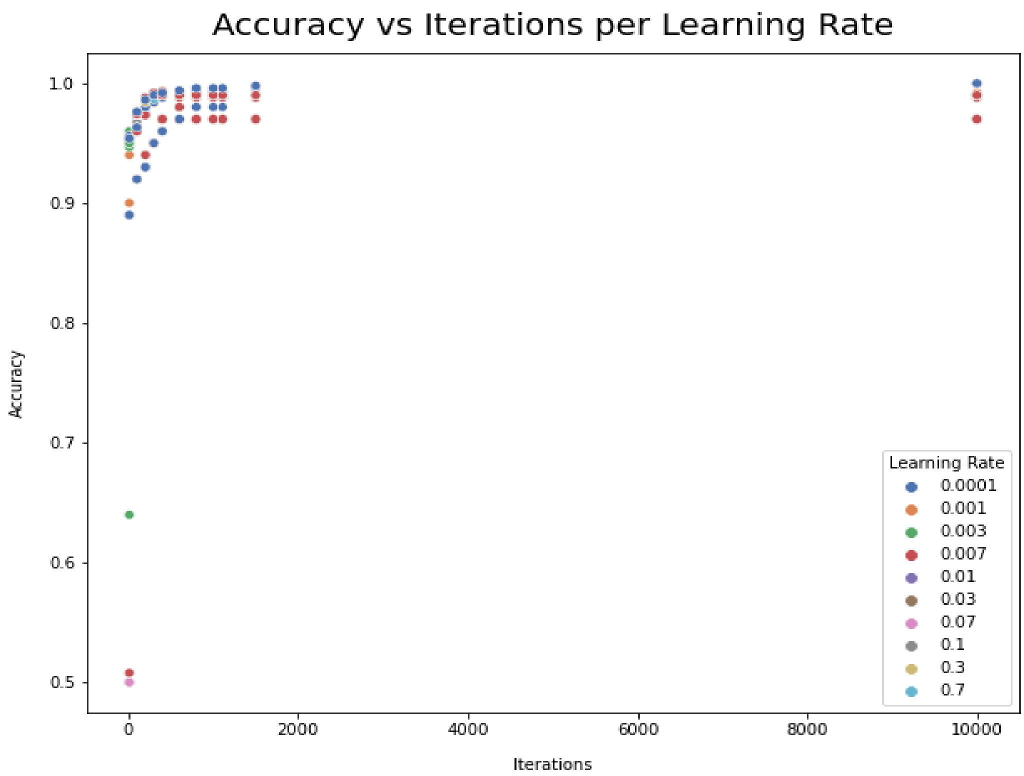
Hyperparameter	**Description**	**value**
M	Gradient Descent iterations	400
α	Learning rate	0.1
Data split	Validation:Test	25:75

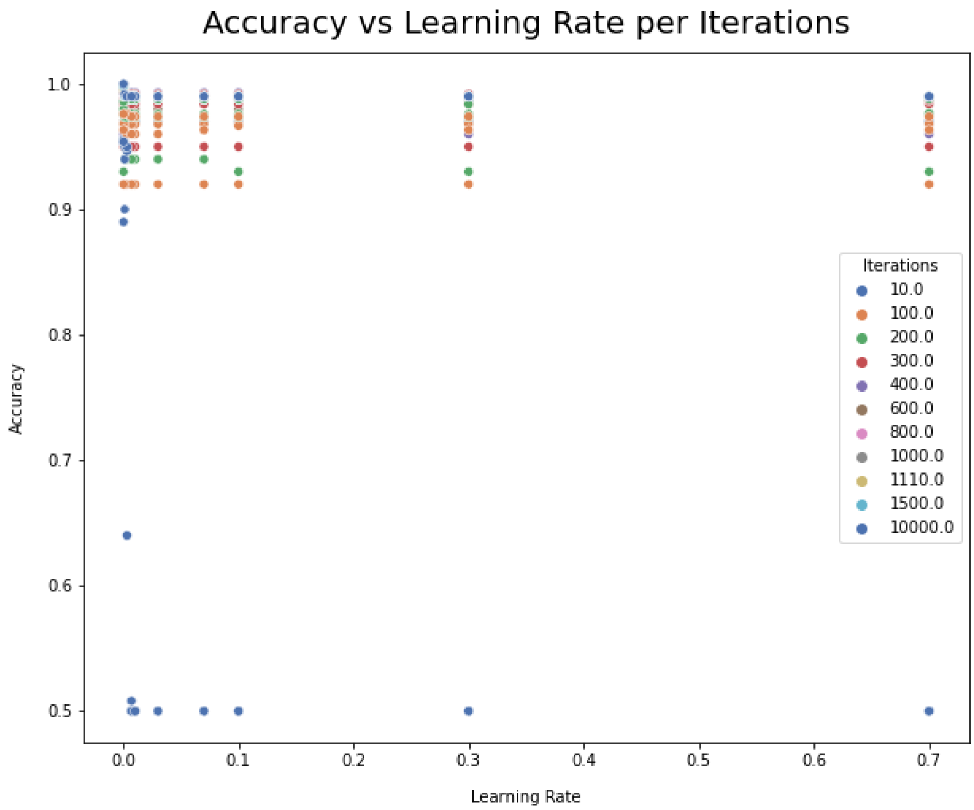
High performance:

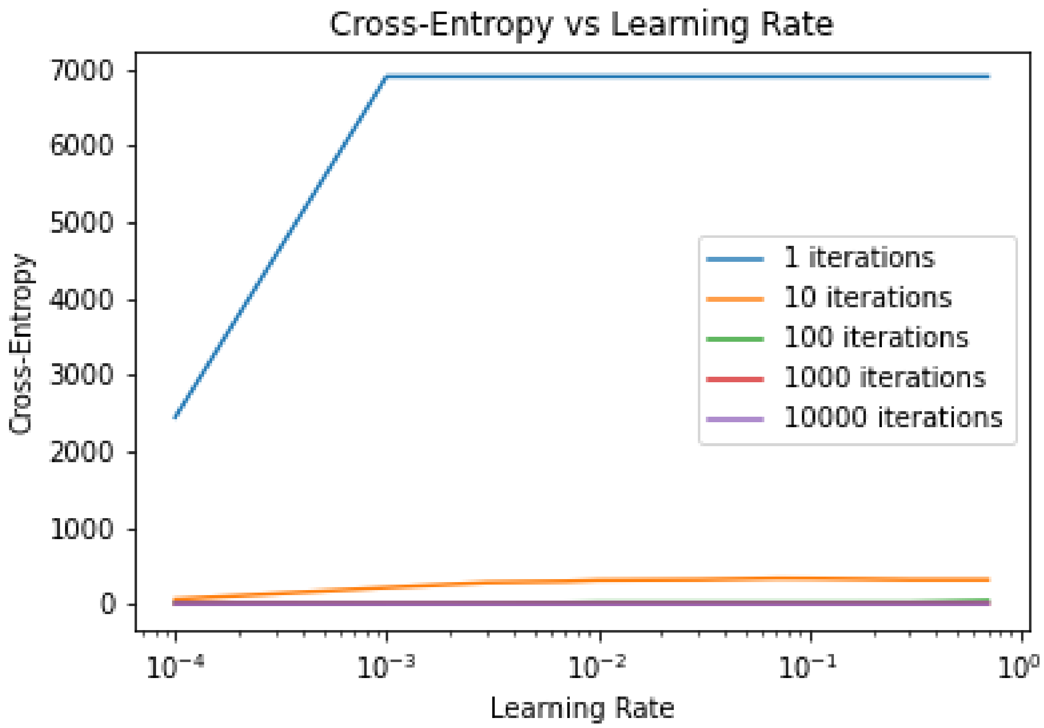
Hyperparameter	**Description**	**value**
M	Gradient Descent iterations	10000
α	Learning rate	0.0001
Data split	Validation:Test	all possible

After inspecting all the possible combinations, we can determine that besides several edge-cases, the change of learning rates or number of iterations provide very similar values for accuracy and cross entropy.

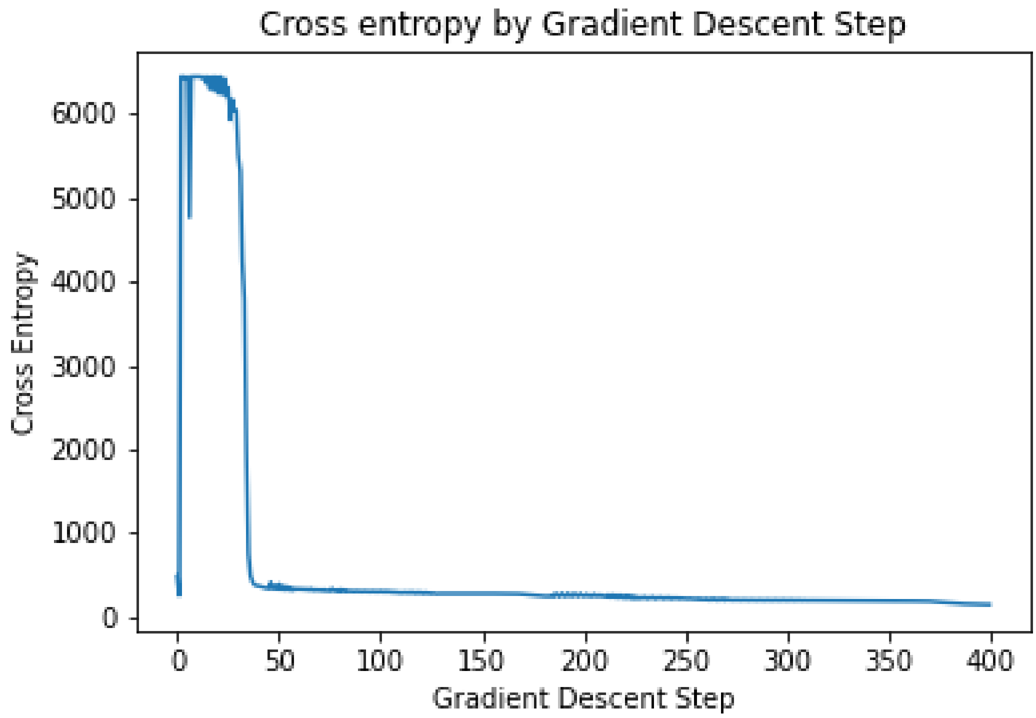
Other values, such as more iterations (10,000) gave slightly better accuracy – perfect 100% though it contains a risk for overfitting and performs many more iterations that with a bigger dataset might cost too much time and money. Below are the results:


Accuracy vs Iterations per Learning Rate
1.0
0.9
0.8
0.7
0.6
0.5
Accuracy
0
2000
4000
6000
8000
10000
Iterations
Learning Rate
0.0001
0.001
0.003
0.007
0.01
0.03
0.07
0.1
0.3
0.7


Accuracy vs Learning Rate per Iterations
1.0
0.9
0.8
0.7
0.6
0.5
Accuracy
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Learning Rate
Iterations
10.0
100.0
200.0
300.0
400.0
600.0
800.0
1000.0
1110.0
1500.0
10000.0



The Cross Entropy by Gradient Descent step using the optimal hyperparameters values:



More detailed view of the above, separated by each value of learning rate:

The grid (Here are the best accuracy rates and the worst ones, the complete grid is joined in an external file – 'grid.xlsx'):

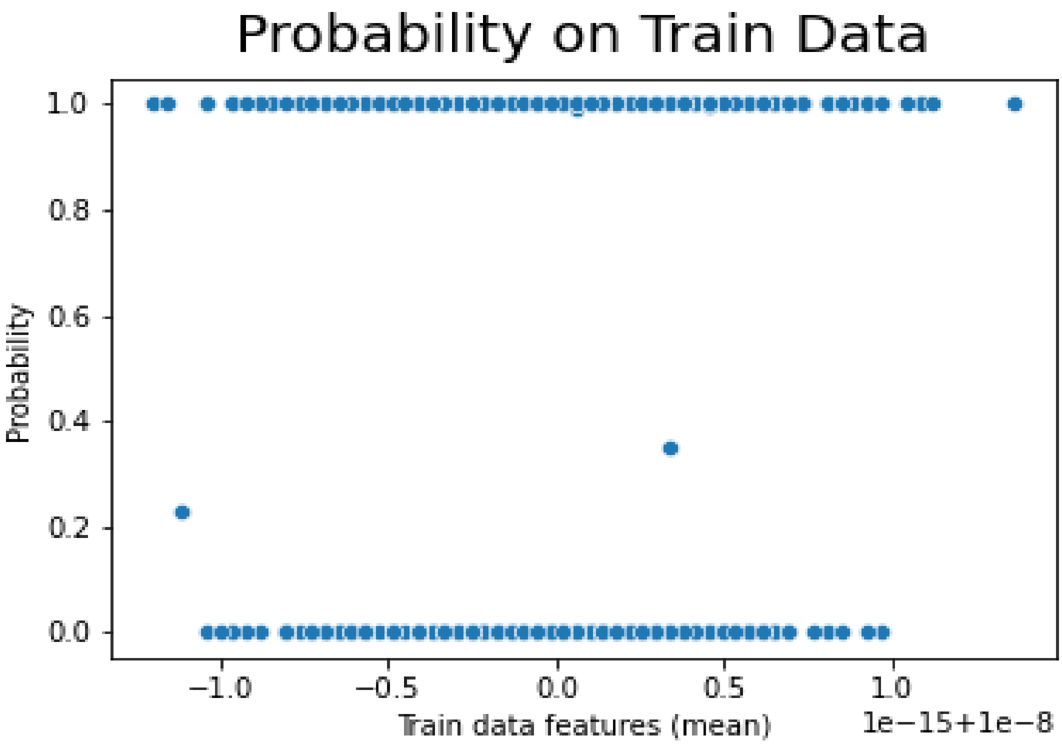
Best accuracy:

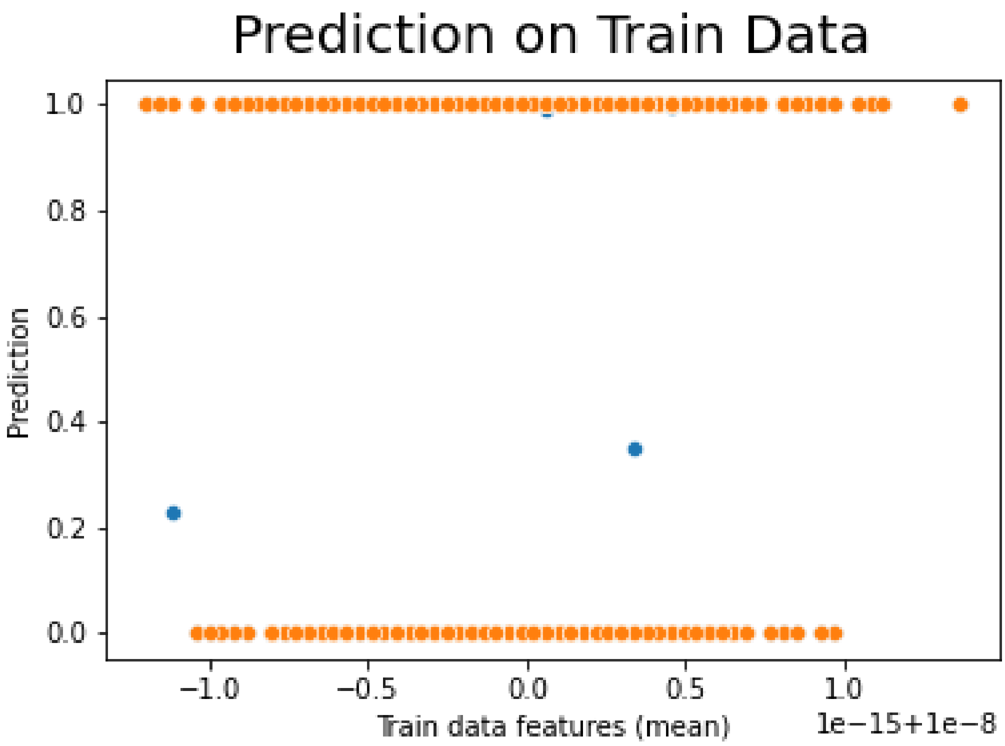
Accuracy	Error	Iterations	Learning Rate	Data Validation Percent
1	0	10000	0.0001	10
1	0	10000	0.0001	25
1	0	10000	0.0001	30
1	0	10000	0.0001	50
0.998	0.002	1500	0.0001	50
0.996667	0.003333	1500	0.0001	30
0.996	0.004	1500	0.0001	25
0.996	0.004	800	0.0001	50
0.996	0.004	1000	0.0001	50
0.996	0.004	1110	0.0001	50
0.994	0.006	600	0.0001	50
0.993333	0.006667	400	0.1	30
0.993333	0.006667	400	0.01	30
0.993333	0.006667	400	0.03	30
0.993333	0.006667	400	0.07	30
0.993333	0.006667	400	0.001	30
0.993333	0.006667	400	0.003	30
0.993333	0.006667	400	0.007	30
0.993333	0.006667	600	0.0001	30
0.993333	0.006667	800	0.0001	30
0.993333	0.006667	1000	0.0001	30
0.993333	0.006667	1110	0.0001	30
0.992	0.008	400	0.1	25
0.992	0.008	400	0.3	25
0.992	0.008	400	0.01	25
0.992	0.008	400	0.03	25
0.992	0.008	400	0.07	25
0.992	0.008	400	0.001	25
0.992	0.008	400	0.003	25
0.992	0.008	400	0.007	25

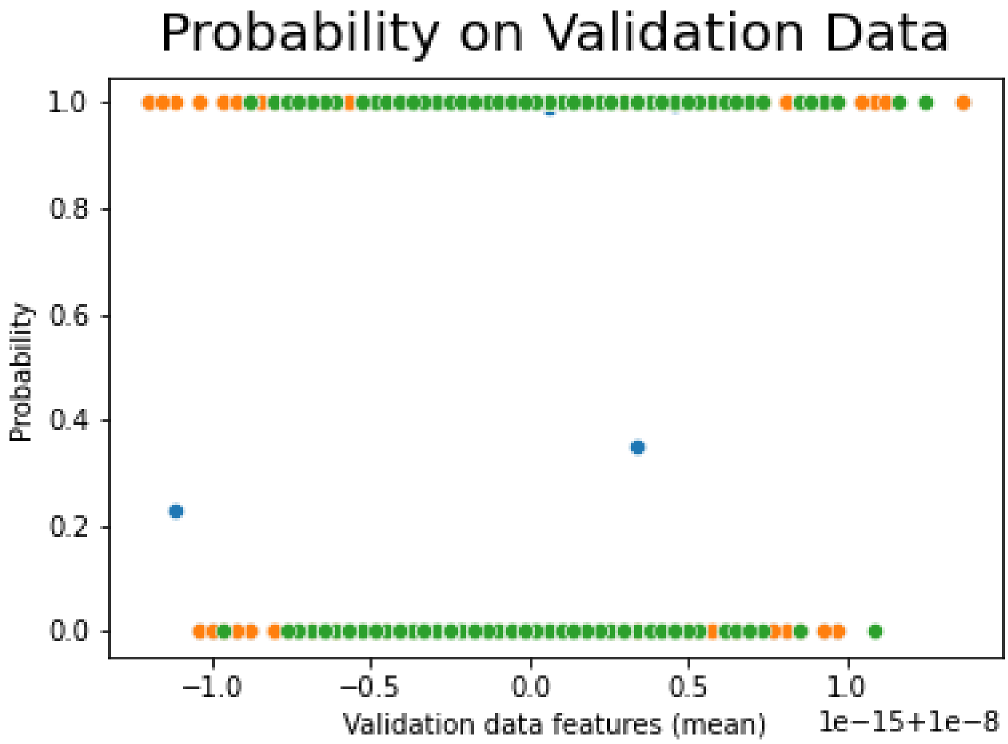
Worst accuracy:

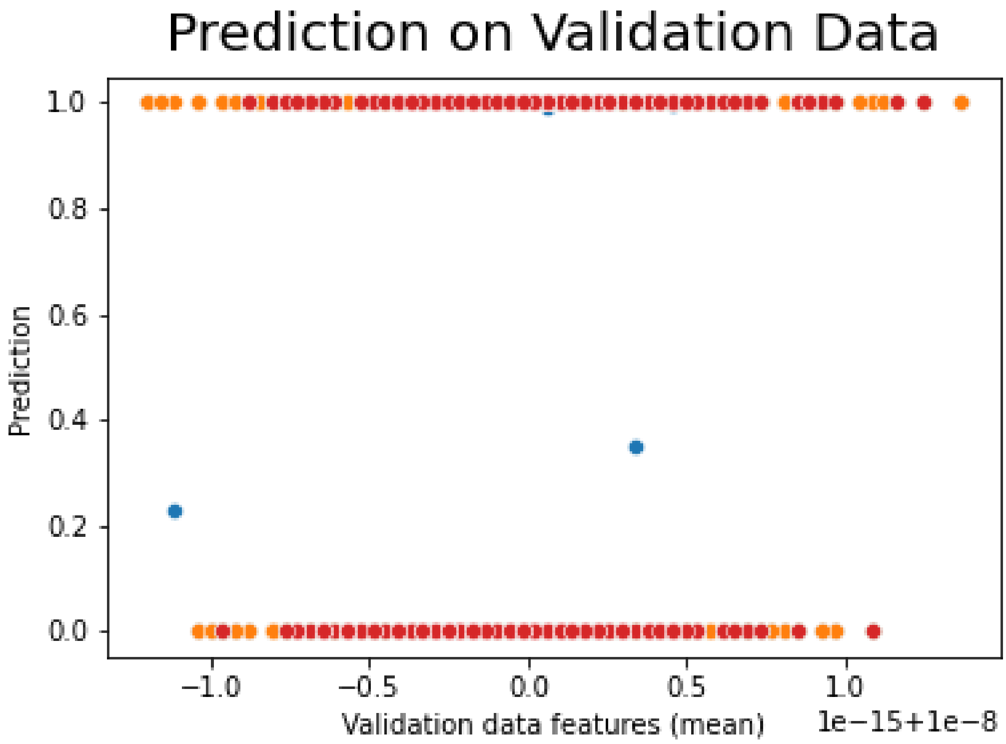
Accuracy	Error	Iterations	Learning Rate	Data Validation Percent
0.89	0.11	10	0.0001	10
0.64	0.36	10	0.003	10
0.508	0.492	10	0.007	50
0.5	0.5	10	0.1	10
0.5	0.5	10	0.3	10
0.5	0.5	10	0.7	10
0.5	0.5	10	0.01	10
0.5	0.5	10	0.03	10
0.5	0.5	10	0.07	10
0.5	0.5	10	0.007	10
0.5	0.5	10	0.1	25
0.5	0.5	10	0.3	25
0.5	0.5	10	0.7	25
0.5	0.5	10	0.01	25
0.5	0.5	10	0.03	25
0.5	0.5	10	0.07	25
0.5	0.5	10	0.007	25
0.5	0.5	10	0.1	30
0.5	0.5	10	0.3	30
0.5	0.5	10	0.7	30
0.5	0.5	10	0.01	30
0.5	0.5	10	0.03	30
0.5	0.5	10	0.07	30
0.5	0.5	10	0.007	30
0.5	0.5	10	0.1	50
0.5	0.5	10	0.3	50
0.5	0.5	10	0.7	50
0.5	0.5	10	0.01	50
0.5	0.5	10	0.03	50
0.5	0.5	10	0.07	50

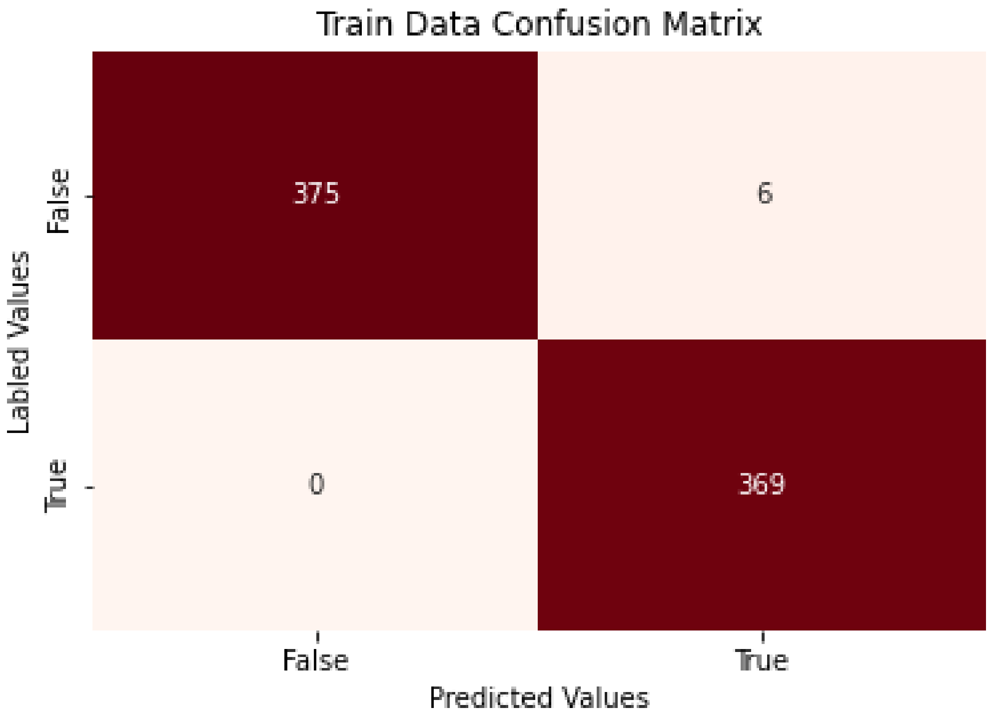
We can see that the model gives almost obsolete predictions and probabilities for the given train and validation data:

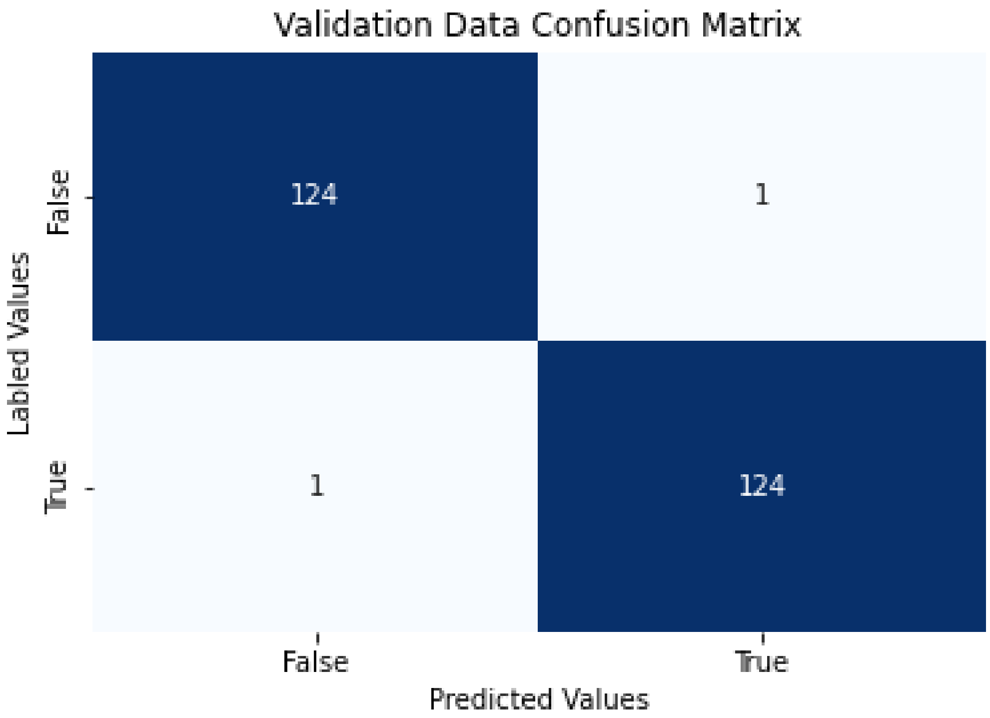


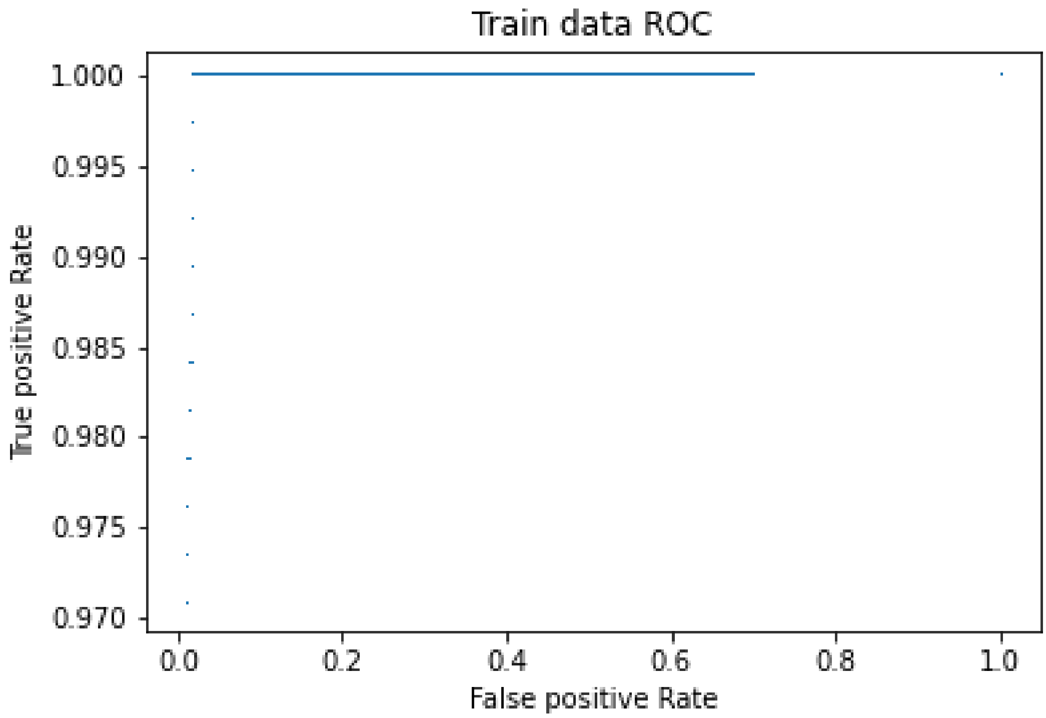


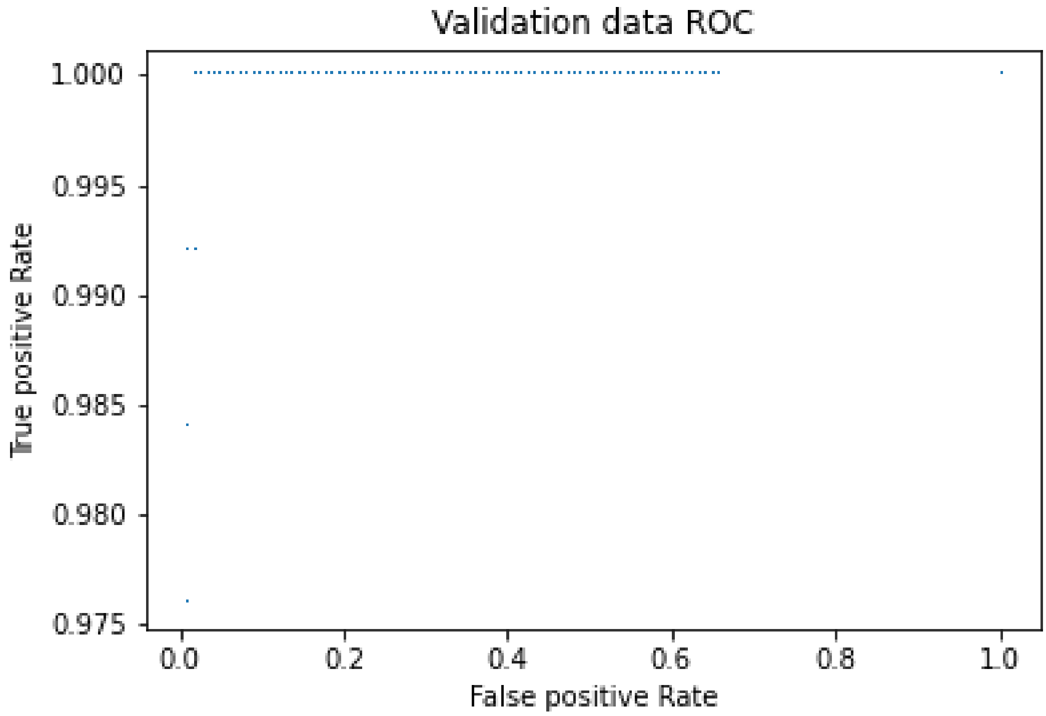

Probability on Validation Data
Probability
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0
Validation data features (mean)
1e−15+1e−8


Prediction on Validation Data
Prediction
1.0
0.8
0.6
0.4
0.2
0.0
−1.0
−0.5
0.0
0.5
1.0
Validation data features (mean)
1e−15+1e−8


Train Data Confusion Matrix
375
6
0
369
False
True
False
True
Labled Values
Predicted Values


Validation Data Confusion Matrix
124
1
1
124
False
True
False
True
Labled Values
Predicted Values


Train data ROC
True positive Rate
False positive Rate
1.000
0.995
0.990
0.985
0.980
0.975
0.970
0.0
0.2
0.4
0.6
0.8
1.0


Validation data ROC
True positive Rate
False positive Rate
1.000
0.995
0.990
0.985
0.980
0.975
0.0
0.2
0.4
0.6
0.8
1.0

Different cross-validation result:

Folds	Iterations	Learning Rate	Accuracy Average
4	1110	0.01	0.992
10	10000	0.01	0.992
4	1110	0.07	0.992
10	1000	0.1	0.992
10	1500	0.1	0.992
4	10000	0.3	0.992
10	10000	0.3	0.992
10	1500	0.7	0.992
10	1000	0.007	0.991
4	1000	0.1	0.99
4	800	0.3	0.99
10	800	0.3	0.99
4	1110	0.3	0.99
4	1500	0.3	0.99
10	1110	0.7	0.99
4	10000	0.7	0.99
4	10000	0.007	0.989
10	800	0.01	0.989
10	600	0.07	0.989
4	800	0.07	0.989
10	800	0.07	0.989
10	600	0.1	0.989
4	600	0.7	0.989
4	800	0.7	0.989
10	1000	0.01	0.988
10	1110	0.01	0.988
10	300	0.03	0.988
4	800	0.03	0.988
10	800	0.03	0.988
4	400	0.07	0.988

Row Labels	Sum of Predicted
500.jpg	0
501.jpg	0
502.jpg	1
503.jpg	1