Mini Project

Xin Huang(xhuang2), Qi Pang(qpang)

1. Introduction:

In this project, we compare the accuracy of prediction for Naive Bayes, Neural Network and Logistic Regression on the 'Skin Segmentation Data Set' (classification) where we get from https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation#. The goal for our project is to get the best prediction algorithm for our dataset

a. Data:

The Skin Segmentation dataset is constructed over B, G, R color space. Skin and Nonskin dataset is generated using skin textures from face images of diversity of age, gender, and race people. The dataset has 3 features (Blue, Green, Red) over 245057 data points. All features are numerical (between 0 to 255). The binary target Skin represents by 1 with 50859 samples and Nonskin represents by 0 with 194198 samples.

b. Importance of the problem:

The result of this project provides us a way to find people face from pictures. We can also input some BGR data, and see if they can be human's face color.

2. Design of Experiments:

a. Selection of metric:

we use accuracy as our metric, because skin vs unskin rate for our dataset is 1/4. It is not highly unbalanced like 1/100. We got accuracies more than 90%, which means accuracy is still a good choice for metric.

Basically we calculate how many correct predictions do we have, then divided by number of test units, then times 100% which gives us accuracy rate.

Accuracy=(Number of correct predictions / Number of units in testset)*100%

b. Data splits:

We found 100,000 data points already big enough for accuracy. In order to improve the running speed, we just choose 100,000 data points. We shuffle the dataset and randomly pick 100,000 data points for training and testing, then randomly pick another 100,000 data points for hyperparameter validation by call python random_select.py. We used the output dataset "train_test.txt" for training and testing, and "validation.txt" for hyperparameter validation in this project.

For feature set, we also add a column of 1 as bias for classification.

For training and testing, we use K-fold cross validation. We Splitting the dataset to 10 disjoint sets, each set has 10000 samples. Also same strategy for validation.

c. Data normalization:

In order to quickly approach the gradient result, we choose to normalize our date.

Normalize function:

for each unit in dataset:

for each feature in unit:

feature = feature / the largest number in entire dataset

That makes every feature to become the number between 0 to 1

3. Model:

We obtain QQ and frequency plots for our three features by SPSS (showing blow). Those plots give us a hint that our features fit for Gaussian distribution. Also our targets are 0 and 1 which is binary. So we choose logistic regression, neural network and naive bayes.

Naive bayes:

probability calculated by Gaussian.

Logistic regression:

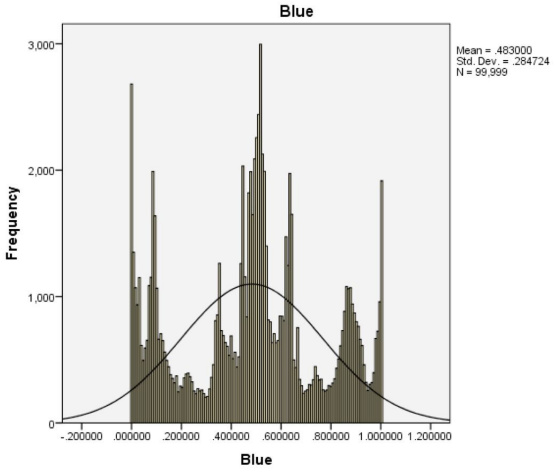
Using Batch gradient descent, set step size (eta) to 0.001, set tolerance to 0.00001.

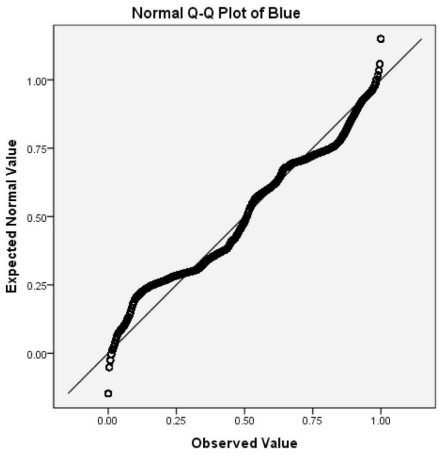
Neural Network:

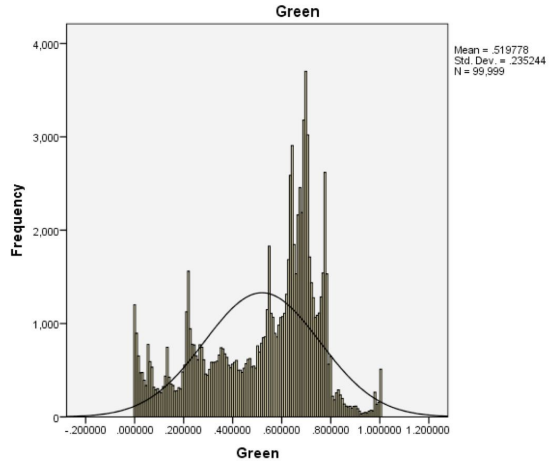
Using Stochastic gradient descent, with one hidden layer and 12 hidden units, default episode is 100, step size is 0.01 and sigmoid function for transfer.

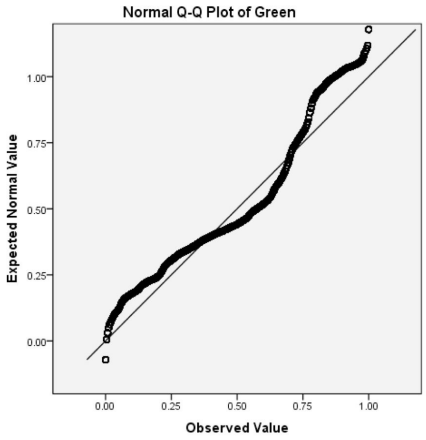
For each algorithm, we use 9 sets as training set and test on the other set. Running 10 times, make sure every set is tested as testing set. Then we obtain 10 models with 10 their accuracy.

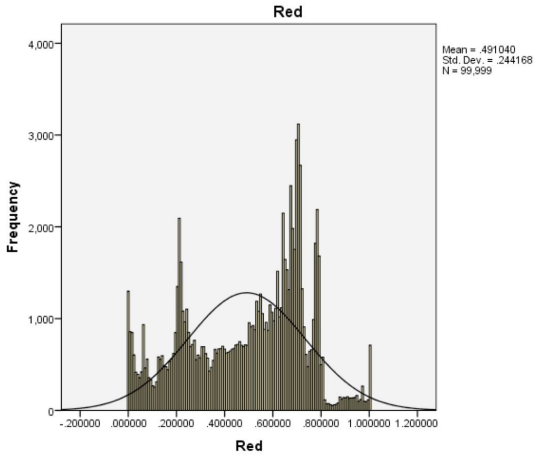
Plots:

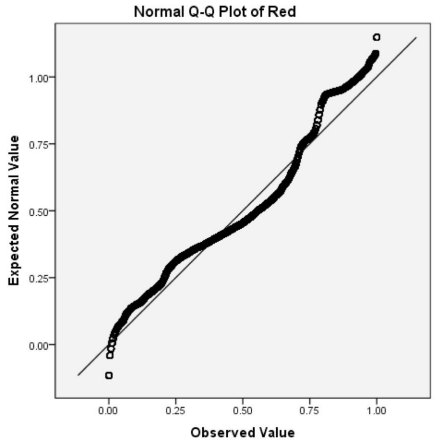












4. **Algorithm parameters:**

We ran dataset “validation.txt” and we used k-fold (followed the Model), totally 10 sets of data, we have 16 runs for Neural network and 17 runs for Logistic regression, we have two parameters “nh” and “step_size”.

“nh” is the number of hidden units in hidden layer for Neural Network. we have 9 different choices for “nh” which are: 2,4,6,8,10,12,14,16 and 32.

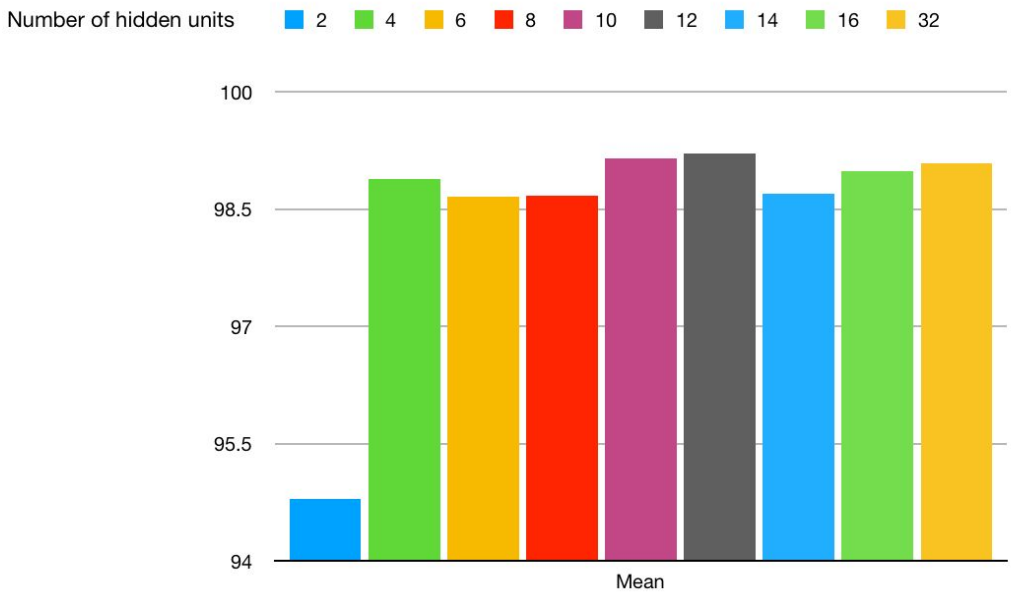
“step_size” is the step size for batch gradient descent when we use logistic regression. we have 9 different choices for step_size which are: 0.01, 0.001, 0.0001, 0.0005, 0.00001, 0.00005, 0.000001, 0.000005 and 0.0000001.

Finally we did not set any hyperparameter for Naive bayes.

For each “nh” and “step_size” we got the results below.

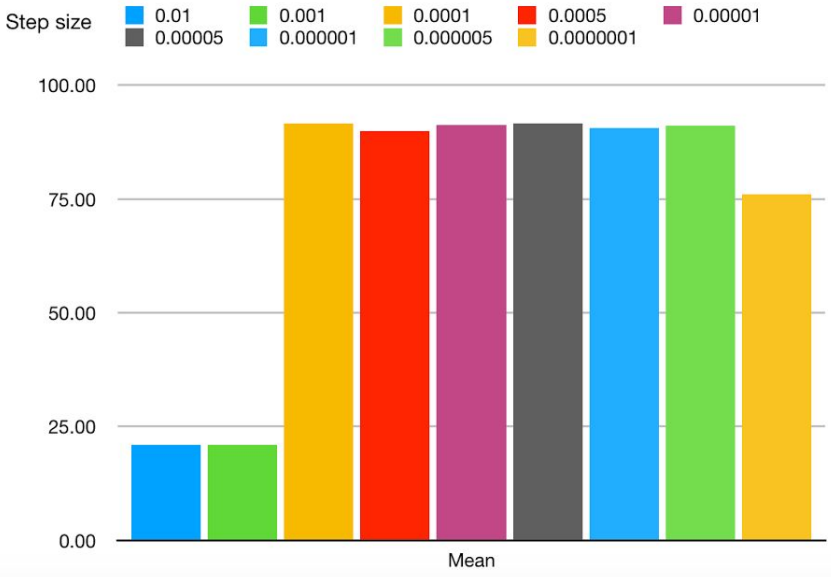
Neural network parameters

Nth run / Number of hidden units	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Mean	Standard deviation
2	94.3	94.7	94.4	95.4	95.1	95.2	94.3	94.7	94.4	95.4	95.1	95.2	94.6	93.8	95.4	94.7	94.79375	0.46566
4	98.6	99.2	98.8	99.1	99.2	98.8	98.6	99.2	98.8	99.1	99.2	98.8	98.9	97.8	99.2	98.8	98.88125	0.35039
6	98.9	98.9	98.2	98.7	99	98.8	98.6	99.2	98.8	99.1	99.2	98.8	98.9	98.3	98.4	97.9	98.65625	0.33160
8	98.8	99	98.1	98.4	99.1	98.5	98.8	99	98.1	98.4	99.1	98.5	98.5	98.8	98.9	98.8	98.67500	0.31524
10	99.3	99.2	98.6	99.2	99.4	99	99.3	99.2	98.6	99.2	99.4	99	98.9	99.5	99.5	99.2	99.15625	0.26685
12	99.2	99.3	98.8	99.3	99.4	99	99.2	99.3	98.8	99.3	99.4	99	99.3	99.5	99.3	99.3	99.21250	0.19961
14	99	99	98,4	98.5	99.1	98.6	99	99	98.4	98.5	99.1	98.6	98.4	98.6	98.8	98.2	98.70000	0.28723
16	99.4	99.2	98.6	98.7	99.2	98.8	99.4	99.2	98.6	98.7	99.2	98.8	98.9	98.9	99.3	98.9	98.98750	0.27128
32	99.3	99.3	98.6	98.8	99.3	99	99.3	99.3	98.6	98.8	99.3	99	99	99.4	99.4	99	99.08750	0.26663



Logistic regression parameters

Nth run / Step size	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Mean	Standard deviation
0.01	20.87	20.9	20.53	20.57	21.85	21.35	20.66	21.24	20.89	19.72	20.87	20.9	20.53	20.57	21.85	21.35	20.66	20.90	0.50
0.001	20.87	20.9	20.53	20.57	21.85	21.35	20.66	21.24	20.89	19.72	20.87	20.9	20.53	20.57	21.85	21.35	20.66	20.90	0.50
0.0001	92.08	91.66	91.77	91.71	91.76	91.76	91.16	91.14	91.66	91.41	92.08	91.66	91.77	91.71	91.76	91.76	91.16	91.65	0.27
0.0005	91.85	83.69	88.6	93.29	92.59	88.69	92.17	87.06	92.65	88.3	91.85	83.69	88.6	93.29	92.59	88.69	92.17	89.98	3.03
0.00001	91.64	91.28	91.58	91.39	91.44	91.3	90.8	90.72	91.35	91.12	91.64	91.28	91.58	91.39	91.44	91.3	90.8	91.30	0.28
0.00005	91.93	91.55	91.69	91.63	91.63	91.11	90.98	91.57	91.34	91.93	91.55	91.69	91.63	91.63	91.65	91.11	91.65	91.55	0.26
0.000001	90.87	90.59	90.89	90.6	90.71	90.52	89.87	90.11	90.74	90.53	90.87	90.59	90.89	80.6	90.71	90.52	98.87	90.56	0.31
0.000005	91.54	91.16	91.46	91.31	91.24	91.12	90.66	90.61	91.17	90.98	91.54	91.16	91.46	91.31	91.24	91.12	90.66	91.16	0.28
0.0000001	76.3	75.84	86.52	76.29	75.27	75.86	76.07	75.97	76.33	77.03	76.3	75.84	76.52	76.29	75.27	75.86	76.07	76.10	0.42



Best perimeters:

For neural network we choose 12 as number of hidden units for hidden layers
For logistic regression we choose 0.0001 as step size when we do batch gradient descent

5. Statistical Significance Test on different Algorithms:

We used two-sided t-test for statistical significance test, we import python spicy for t-test. To prevent that in case the variances of the populations are not equal, we actually used Welch's t-test by setting 'equal_var' to 'False'.

We have three groups of t-test, and alpha is 0.01

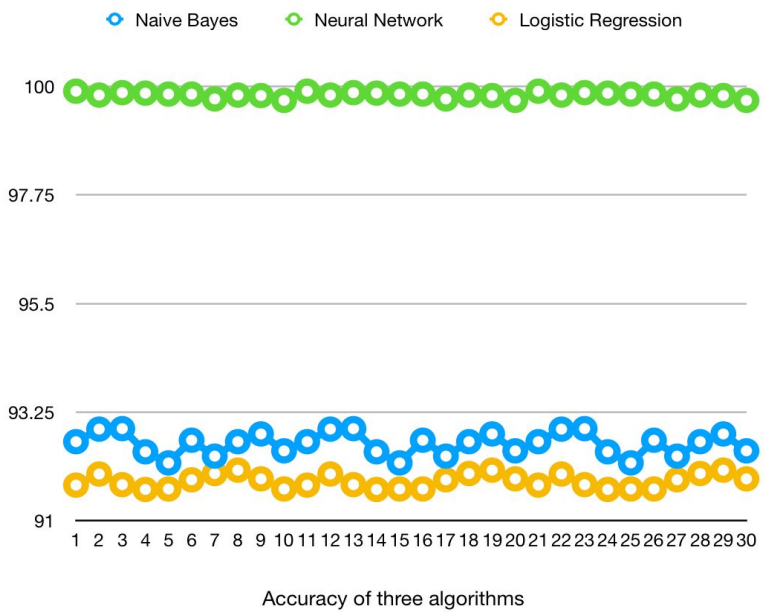
For each algorithm we ran 30 times with dataset “train_test.txt” followed k-fold and our model design) also set parameter to the best parameter that we got from part 4. The error table and t-test shows below:

Accuracy for 3 algorithms

Nth run / Algorithms	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Naive Bayes	92.64	92.9	92.91	92.43	92.2	92.67	92.34	92.64	92.8	92.45	92.64	92.9	92.91	92.43	92.2
Neural Network	99.9	99.82	99.87	99.86	99.84	99.84	99.74	99.82	99.81	99.71	99.9	99.82	99.87	99.86	99.84
Logistic Regression	91.74	91.97	91.75	91.65	91.66	91.85	91.98	92.05	91.87	91.66	91.74	91.97	91.75	91.65	91.66
Nth run / Algorithms	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**
Naive Bayes	92.67	92.34	92.64	92.8	92.45	92.64	92.9	92.91	92.43	92.2	92.67	92.34	92.64	92.8	92.45
Neural Network	99.84	99.74	99.82	99.81	99.71	99.9	99.82	99.87	99.86	99.84	99.84	99.74	99.82	99.81	99.71
Logistic Regression	91.66	91.85	91.98	92.05	91.87	91.74	91.97	91.75	91.65	91.66	91.66	91.85	91.98	92.05	91.87

Mean, standard deviation and confidence interval

	Mean	Standard deviation	Confidence interval	Confidence level
Naive Bayes	92.598	0.2304	[92.0044,93.1915]	99%
Neural Network	99.821	0.0556	[98.6778,98.9642]	99%
Logistic Regression	91.818	0.1423	[91.4514,92.1846]	99%



H0: Errors for Neural Network and Naive bayes are the same

Ha: Errors for Neural Network and Naive bayes are different

T-value is 166.915514, P-value is 0.000000, which leads us to reject h0

H0: Errors for Neural Network and Logistic regression are the same

Ha: Errors for Neural Network and Logistic regression are different

T-value is 286.902377, P-value is 0.000000, which leads us to reject h0

H0: Errors for Naive bayes and Logistic regression are the same

Ha: Errors for Naive bayes and Logistic regression are different

T-value is 15.775931, P-value is 0.000000 which leads us to reject h0

T-test

	T-value	P-value	Statistical significance level
Neural Network vs Naive Bayes	166.915514	0	0.01
Neural Network vs Logistic regression	286.902377	0	0.01
Naive Bayes vs Logistic regression	15.775931	0	0.01

From three groups of t-test, “Accuracy for 3 algorithms” table and plot tells us that Neural network has a better performance than Naive bayes and logistic regression, and also Naive bayes has a better performance than logistic regression for our dataset

6. Conclusion:

After running three algorithms with the best parameters from part 4, Welch’s t-test lead three rejections of our null hypothesis. So there are difference among those three algorithms.

According to part 5 Statistical Significance Test on different Algorithms, we believe that Neural Network has the best performance. Because neural network generate new features in hidden layer. So it will have more features for predicting the targets which means hidden layer with 12 features is perfect fitting our dataset. (compare neural network and logistic, logistic is a little bit under fit).

7. Github repository:

https://github.com/panchyo0/466project

8. References

https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation#

https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_ind.html