CS-UY 4563: Machine Learning:

Final Project Written Report

Facial Emotion Classification

2:00PM Section

Dec 7, 2022

Professor Linda N.Sellie

Zijun Wang, Fei Xiong

1 Introduction

This project focuses on using machine learning models to predict the emotion of people. The dataset contains 10 features to predict face emotion. Original features are obtained by placing ten virtual markers named Action Units on subjects face and measuring the Euclidean distance between the center marker and 10 virtual markers.

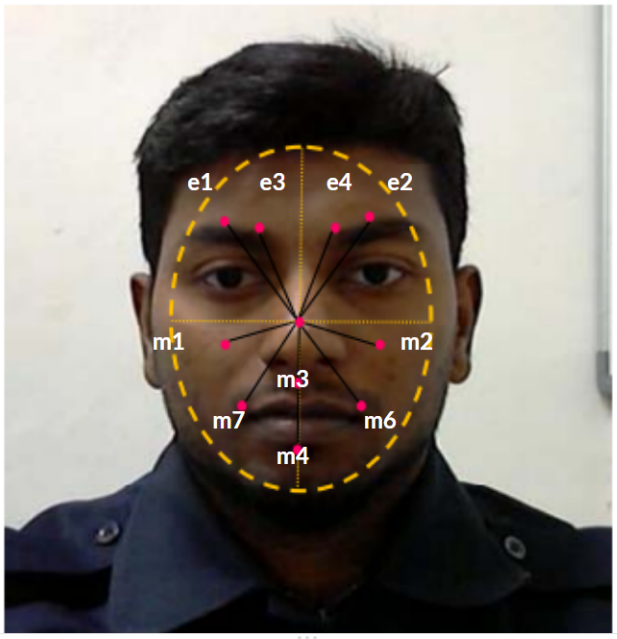
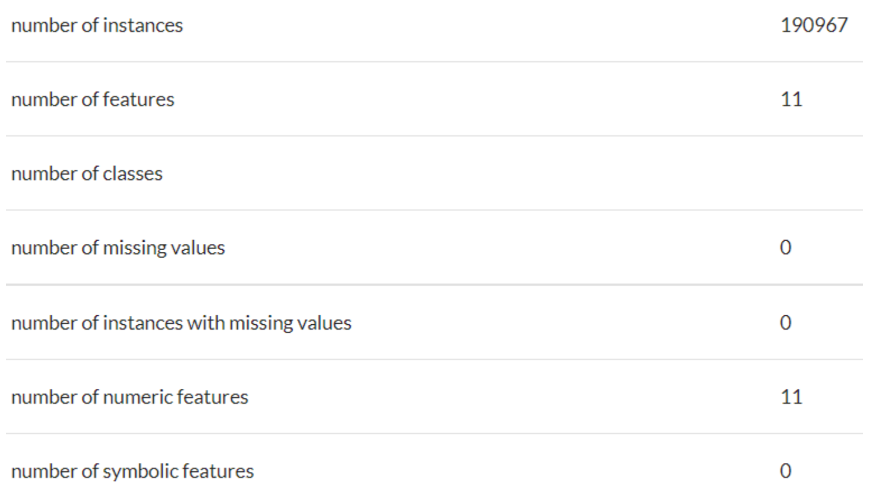


Figure 1: Virtual and Center markers

We used logistic regression, support vector machines and neural networks to classify emotions into six facial expressions: Anger, Disgust, Fear, Sad, Happy and Surprise, represented correspondingly as 0-5. All three models were simulated multiple times using different regularization techniques and feature transformation methods.

2 Preprocessing

The original dataset was obtained from openML datasets. The dataset itself is balanced. We decided to keep the original dataset and perform transformation such as scaling based on the needs of algorithm. We demonstrated to randomly selected 143225(75%) training examples and 47742(25%) testing examples from the original dataset.



number of instances	190967
number of features	11
number of classes	
number of missing values	0
number of instances with missing values	0
number of numeric features	11
number of symbolic features	0

Figure 2: Information of dataset

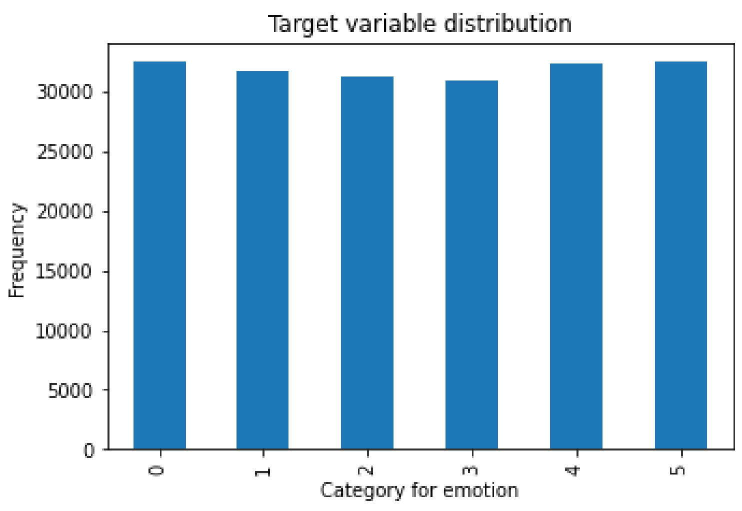


Figure 3: Distribution of the data points for each emotion

3 Logistic Regression

The first method we used to solve this facial emotion classification problem is logistic regression. At the very first, for experimental purposes, we normalized the features and set up the function sklearn.LogisticRegression() with no regularization and default parameters. With this setup, the training accuracy should be very high since we have no regularization on the weights which makes the model can overfit every data point in the training examples. However, the accuracy for training examples is only 0.49. Therefore, it shows that the logistic regression performs poorly on the data points even after scaling. The reason behind this may be the data points are not linearly separable. Therefore, we need to do some feature transformations. Due to the high dimensions of the feature space, it is hard to visualize the relation of the features. Thus, the feature transformations are more based on trial and error. There are total 10 features, for more comfort visualization, these are the distributions for the first 9 features.

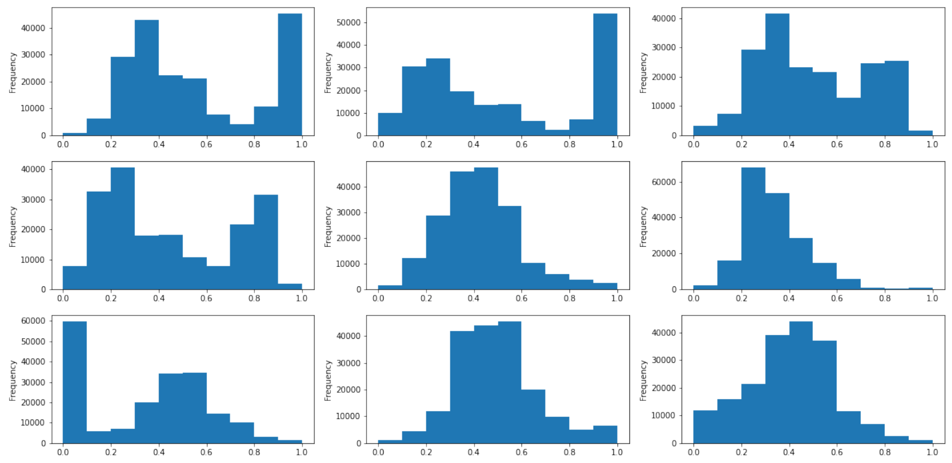


Figure 4: Features before transformation

We have tried square and square root transformations, all common feature scaling. Although the accuracy has raised from 0.49 to a maximum of 0.55, it is still not satisfactory.

Therefore, we have tried to add more features by multiplying and dividing several origin features together. What we do was multiply and divide from 2 to 6 number of features together as a new feature. Therefore, we got a data set containing 1264 features. After the standardized scaling, we got 0.83 prediction accuracy with no regularization. This is an acceptable result, but the time for training is too long due to the huge number of features. Therefore, we removed the features whose weights are less than 0.5, which are total of 714 features - a huge amount. In the end, we preserved a data set with 550 features which is still a big number but much better than the previous setup. The prediction accuracy only drops from 0.83 to 0.82 which is still acceptable. Therefore, the comparison of regularization was built upon this new data set. Since the solver "lbfgs" in the sklearn method only have l_2 penalty, we have drawn the graph of training error vs. testing error based on l_2 penalty on X, X^2, and $\sqrt{X}$

The prediction accuracy reaches the peak as $C = 10000$ in figure 5. The testing error also shows a similar trend since we don't really reach any overfit and underfit, the testing error would be very similar to the training error as shown in the graph.

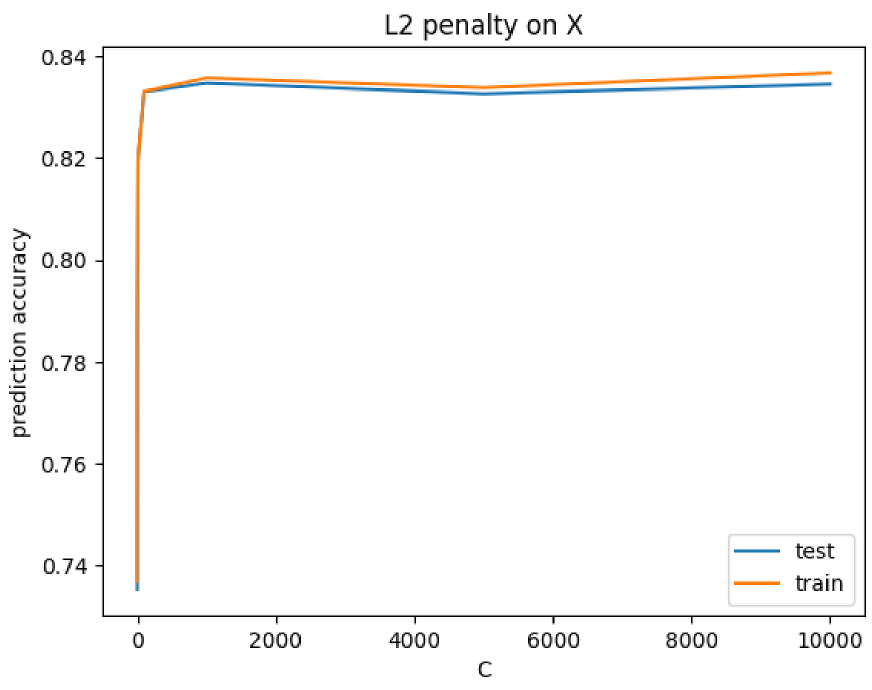


Figure 5: L2 penalty on data set

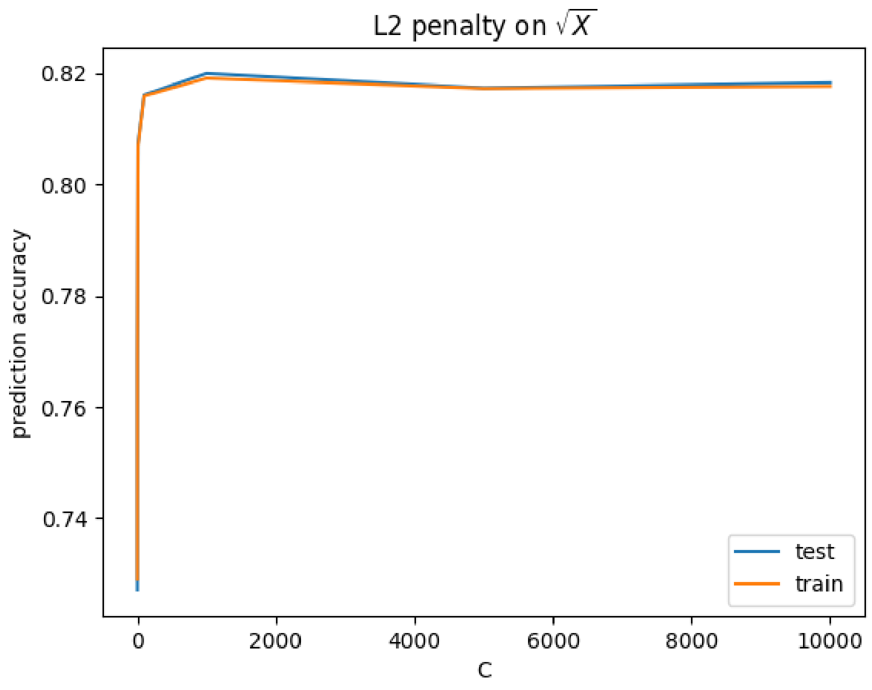


Figure 6: L2 penalty on the square root of the data set

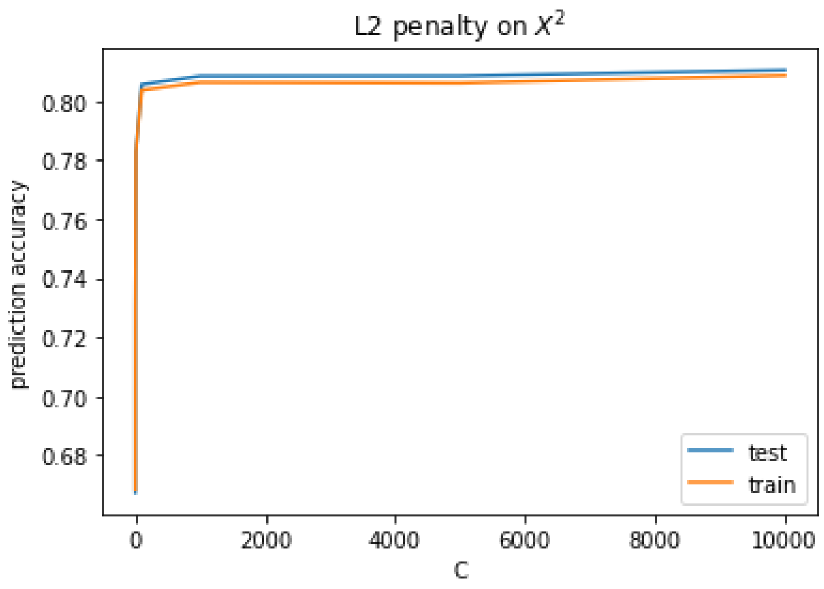
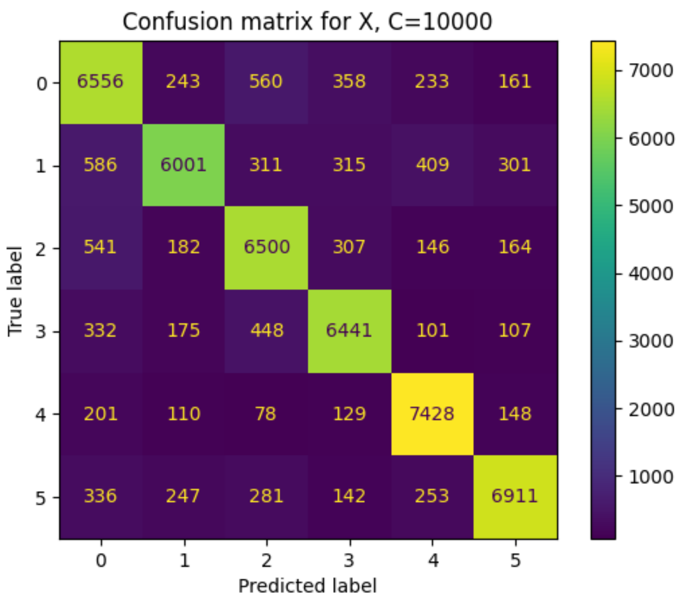


Figure 7: L2 penalty on the square of the data set

Based on those experiments, we draw the confusion matrix based on the best testing error based on the $C = 10000$ L2 penalty on X with standardized scaling.



	0	1	2	3	4	5
0	6556	243	560	358	233	161
1	586	6001	311	315	409	301
2	541	182	6500	307	146	164
3	332	175	448	6441	101	107
4	201	110	78	129	7428	148
5	336	247	281	142	253	6911

Figure 8: Confusion Matrix

As a white box model, one advantage of logistic regression is we can check the weights for every feature in each class. To detect anger, the feature e_2/e_3 and e_1/e_2 have the most weights, which means maybe when people are angry their eyebrows move the most. For disgust, the e_1/e_2 and e_3/e_4 have the most weights which are also the eyebrows. For fear, $m_1 \times m_6$ performs most, and it is no longer related that much to the top side of the face but to the bottom side. For sad, the $m_3 \times m_4 \times m_6$ stands out, which is pretty intuitive and correct since sad is related very much to the movement of the mouth. For Happy, the most weighted feature is $m_2 \times m_3$, which is counterintuitive since it is not related most to the corner of the mouth. For the surprise, the single feature m_2 stands out which is pretty surprising since maybe that means we can see the left side of people's faces to find out whether they are surprised.

4 Support Vector Machine

The second model we experimented with on our data set is the support vector machine using the sklearn.svm.SVC(). In our hypothesis, SVM should perform better than logistic regression since it can handle the nonlinear data without adding more features manually. Therefore, for the experiment of using SVM on our data set, we would use square root transformation, square transformation, and no transformation with the standardized scaler for the "RBF" kernel. We would experiment with how this setup performs on L2 penalty with $C = [5, 10, 15, 20, 25]$. The result is pretty surprising since the training accuracy does not necessarily increase as C increases. We have tried to modify every hyperparameter in the sklearn function like gamma and decision function shape, but this phenomenon always appears. Also, the ”linear”, ”poly”, and ”sigmoid” kernels perform poorly as we expected, so the model would be trained using ”RBF” kernel.

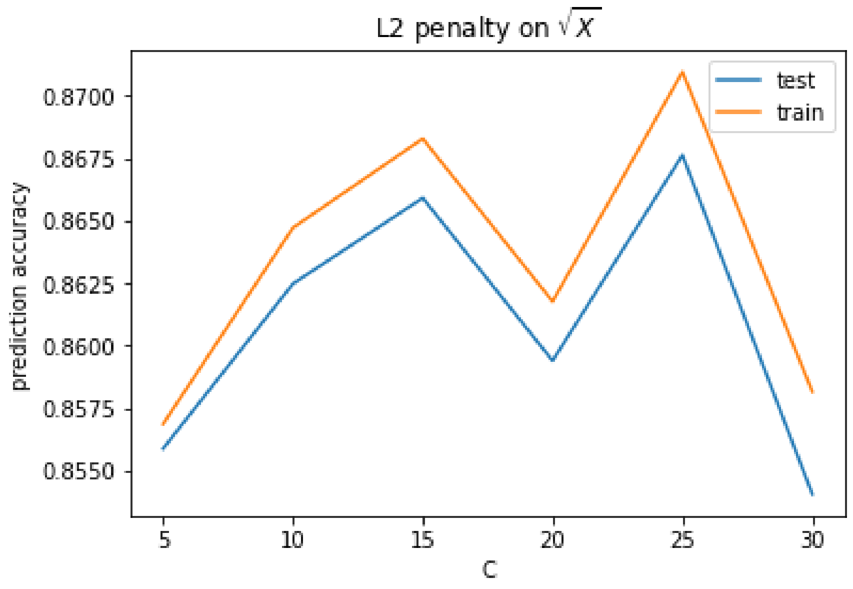


Figure 9: L2 penalty on the square root of the data set with RBF kernel

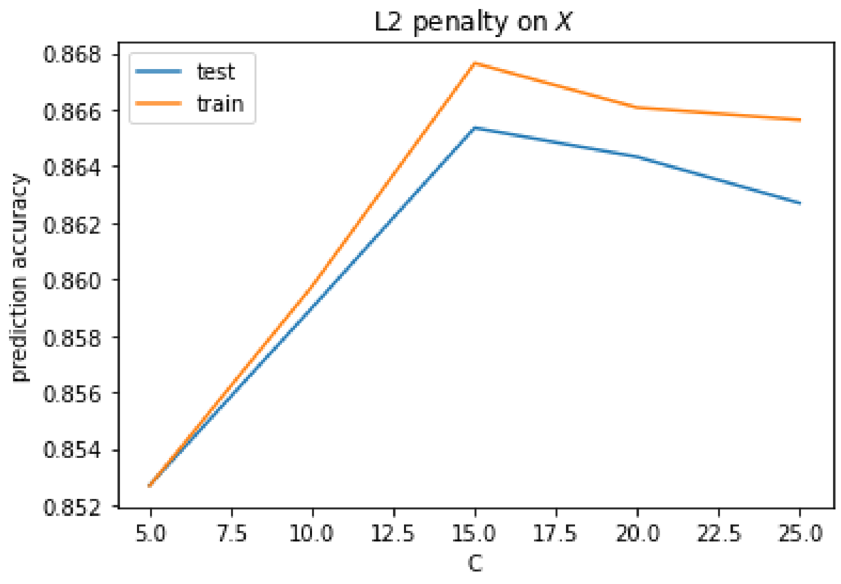


Figure 10: L2 penalty on the data set with RBF kernel

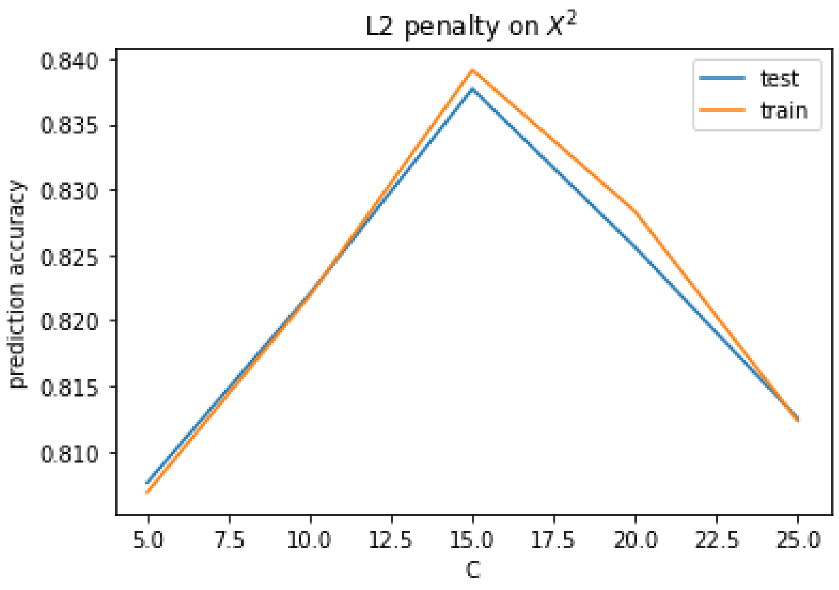
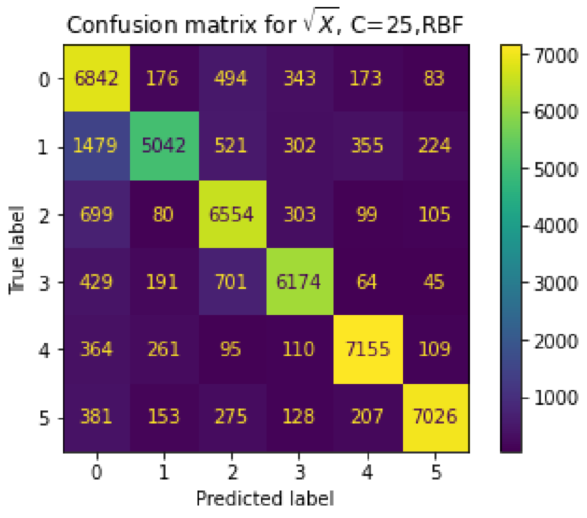


Figure 11: L2 penalty on the square of the data set with RBF kernel

Based on those experiments, we draw the confusion matrix based on the best testing error based on the $C = 25$ L2 penalty on $\sqrt{X}$ with standardized scaling in the ”RBF” kernel.



	0	1	2	3	4	5
0	6842	176	494	343	173	83
1	1479	5042	521	302	355	224
2	699	80	6554	303	99	105
3	429	191	701	6174	64	45
4	364	261	95	110	7155	109
5	381	153	275	128	207	7026

Figure 12: Confusion Matrix

5 Neural Network

The third model we performed experiment on teh dataset was neural networks. We used activation function including Rectified Linear Unit(ReLU), Tanh and Logistic. These activation function were tested on different neural network settings including one hidden layer of 40 neurons, two hidden layers of 40 and 20 neurons and three hidden layers of 80,40 and 20 neurons.

During experiments, we initially considered using C of ranging from 0.0001 to 1000. However, after the result coming out, we found that when C is larger than 1 such as 10 or above, the prediction accuracy was quite low that is not meaningful to execute algorithm using these values.

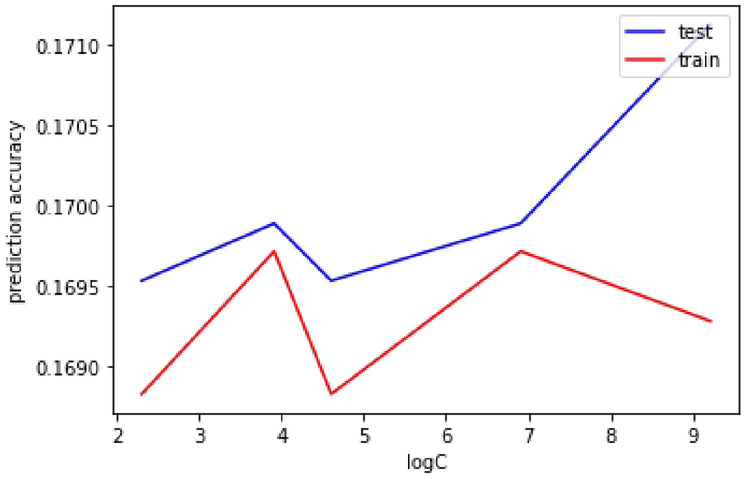


Figure 13: C = [10, 50, 100, 1000, 10000]

Therefore, hyperparameter values of $C = [0.0001, 0.001, 0.01, 0.1, 1]$ were used to perform Ridge Regression regularization(L2) on the neural networks. C values were chosen based on the typical pattern that lower C leads to higher precision recall in neural network. Graphs were plotted that test and training precision accuracy are y-axis while x-axis is the value of logarithmic C to provide an equally distributed x-axis.

Neural network models were implemented using *sklearn.neural_network.MLPClassifier*. All the models with different hidden layers and activation functions presented high precision accuracy when C is small and low precision accuracy when C is relatively large. This is reasonably corresponding to neural network's pattern that lower C tends to have better performance.

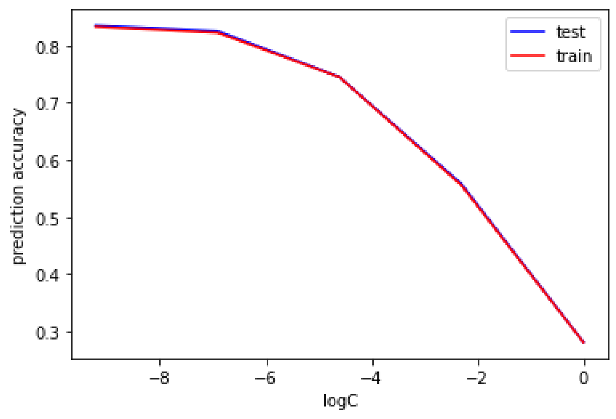


Figure 14: Logistics (40)

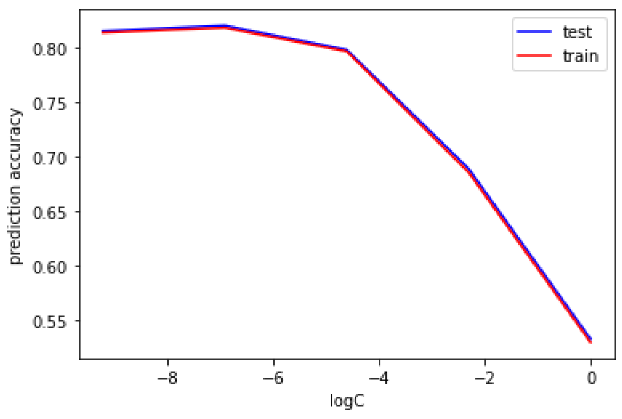


Figure 15: ReLU (40)

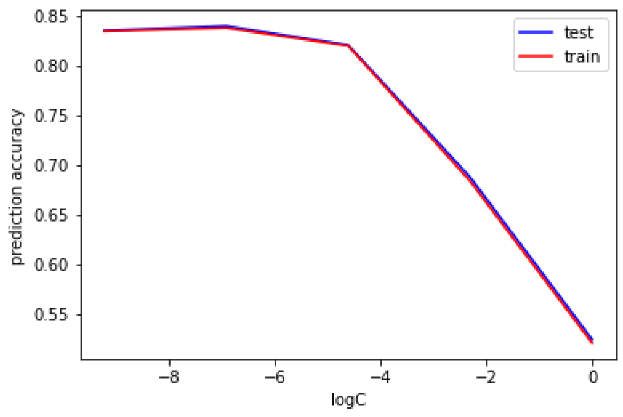


Figure 16: Tanh (40)

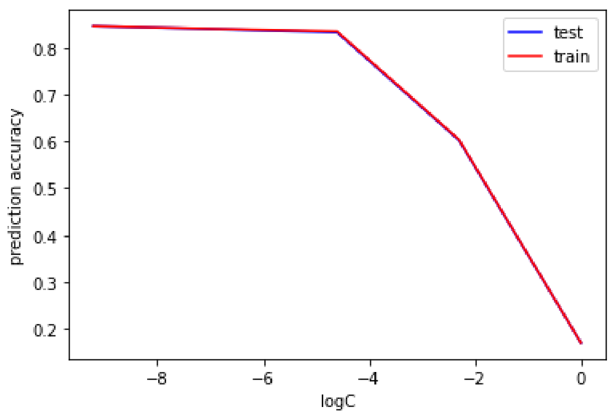


Figure 17: Logistic (40,20)

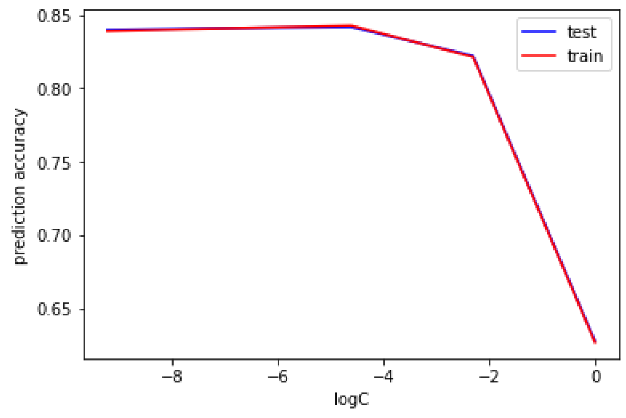


Figure 18: ReLU (40,20)

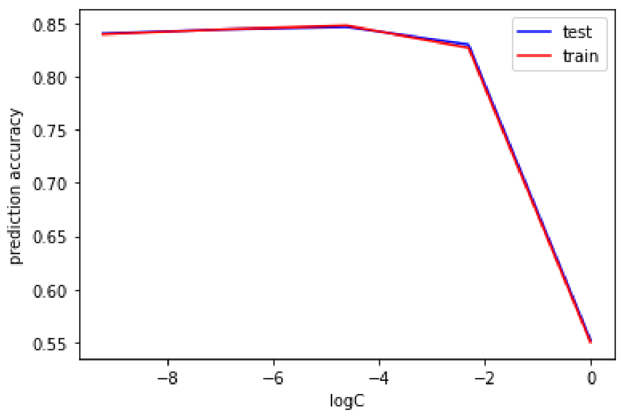


Figure 19: Tanh (40,20)

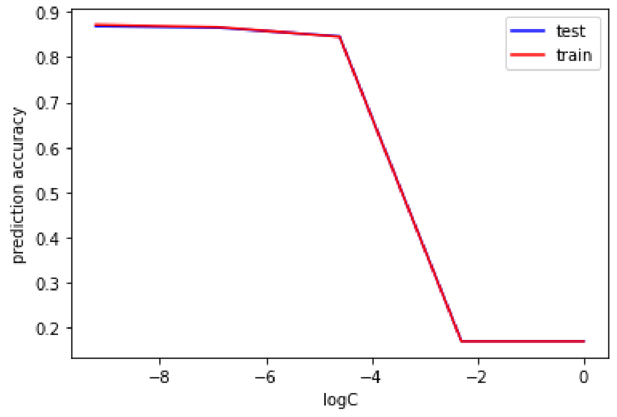


Figure 20: Logistics (80,40,20)

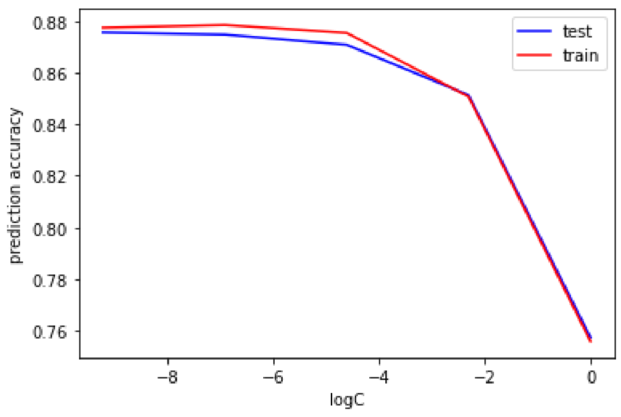


Figure 21: ReLU (80,40,20)

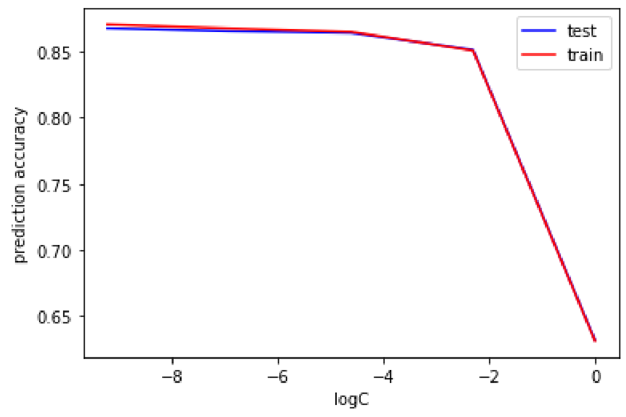


Figure 22: Tanh (80,40,20)

Based on the experiment results, the typical pattern of the neural network models was that more hidden layers led to better average and best performance. Besides, more hidden layers also led to a slower descending accuracy with the increase of C. Test data and train data had low difference in both graph patterns and real data. The result of executing neural network algorithm was that activation function of ReLU with three hidden layers has the highest average performance and best performance in test data prediction accuracy and C-value of 0.0001 provided the best performance of prediction accuracy. After we re-ran the algorithm using ReLU with three hidden layers, C-value of 0.0001 and three types of transformation including $\sqrt{X}, X, X^2$, the confusion matrixes were shown below that we received relatively high accuracy using neural network algorithm while transformation methods didn't seem to affect the result to a degree.

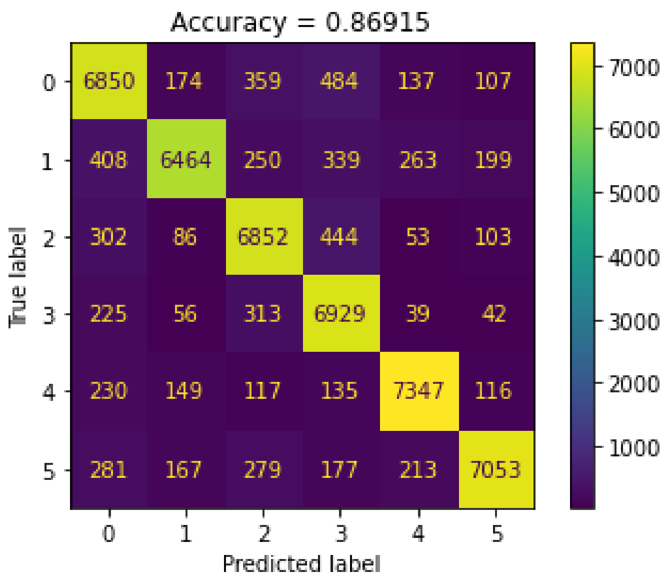


Figure 23: Confusion matrix for X, C=0.0001

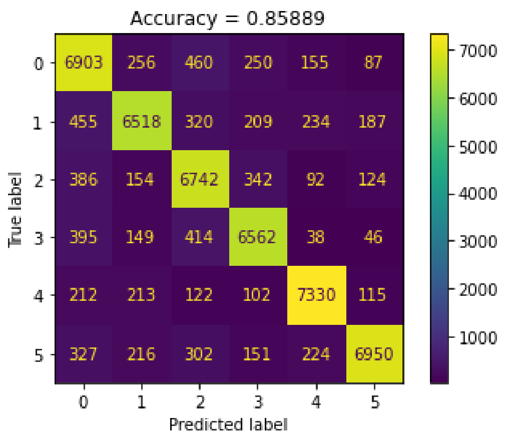


Figure 24: Confusion matrix for $\sqrt{X}$

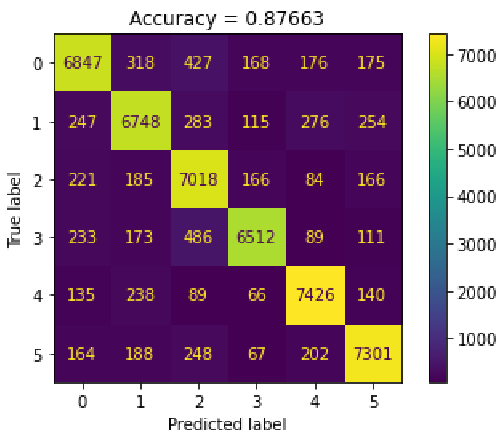


Figure 25: Confusion matrix for X^2

6 Results

Logistic Regression (training prediction accuracy):

	C=0.1	C=0.5	C=1	C=10	C=100	c=1000	c=5000	c=10000
X^2	66.86%	70.36%	73.12%	78.29%	80.39%	80.83%	80.80%	80.79%
X	73.69%	76.56%	78.79%	81.97%	83.30%	83.57%	83.37%	83.69%
$\sqrt{X}$	72.91%	75.77%	77.00%	80.69%	81.59%	81.91%	81.72%	81.76%

Logistic Regression (testing prediction accuracy):

	C=0.1	C=0.5	C=1	C=10	C=100	c=1000	c=5000	c=10000
X^2	66.74%	70.29%	73.06%	78.28%	80.55%	81.03%	80.89%	80.90%
X	73.53%	76.51%	78.58%	81.99%	83.29%	83.47%	83.26%	83.44%
$\sqrt{X}$	72.72%	75.72%	76.88%	80.73%	81.61%	81.99%	81.73%	81.83%

Support Vector Machine (training prediction accuracy):

	C=5	C=10	C=15	C=20	C=25
X^2	80.69%	82.19%	83.91%	82.83%	81.24%
X	85.27%	85.98%	86.76%	86.61%	86.56%
$\sqrt{X}$	85.69%	86.47%	86.83%	86.18%	87.09%

Support Vector Machine (testing prediction accuracy):

	C=5	C=10	C=15	C=20	C=25
X^2	80.76%	82.20%	83.77%	82.56%	81.26%
X	85.27%	85.90%	86.54%	86.43%	86.27%
$\sqrt{X}$	85.59%	86.25%	86.59%	85.94%	86.76%

Neural Network (testing prediction accuracy for Logistic Activation Function):

	C=0.0001	C=0.001	C=0.01	C=0.1	C=1
(40)	83.53%	82.57%	74.52%	55.81%	28.10%
(40,20)	84.67%	84.02%	85.11%	85.25%	85.46%
(80,40,20)	86.82%	86.49%	84.60%	16.95%	16.95%

Neural Network (testing prediction accuracy for ReLU Activation Function):

	C=0.0001	C=0.001	C=0.01	C=0.1	C=1
(40)	81.49%	81.98%	79.76%	68.88%	53.28%
(40,20)	83.90%	84.09%	84.28%	82.14%	62.66%
(80,40,20)	87.57%	87.48%	87.09%	85.14%	75.73%

Neural Network (testing prediction accuracy for Tanh Activation Function):

	C=0.0001	C=0.001	C=0.01	C=0.1	C=1
(40)	81.49%	81.98%	79.76%	68.88%	53.28%
(40,20)	84.03%	84.41%	84.66%	82.99%	55.25%
(80,40,20)	86.76%	86.53%	86.40%	85.12%	63.19%

7 Conclusion

The logistic regression model provided the best prediction accuracy of 83.46% using sklearn.LogistiRegression with multiclass "multinomial", "lbfgs" solver, and L2 penalty with $c = 10000$. The data that produced the best accuracy is training using no feature transformation of the dataset and standard scalar scaling. The data before transformation and scaling was the dataset after adding and deleting features following the procedure we mentioned before. The data with square root or square transformation performed slightly worse. Other solvers like liblinear and saga perform not satisfactorily as "lbfgs".

The support vector machine provided the best prediction accuracy of 86.76% using the "RBF" kernel and L2 regularization $c = 25$. The data that produced the best accuracy is training using the square root transformation of the dataset and standard scalar scaling. The data with no transformation or square transformation performed slightly worse than the square root transformation. Other kernels like linear, sigmoid, precomputed, and poly only results in prediction accuracy from 0.2 to 0.5 which is pretty bad compared to the "RBF" kernel.

The neural network models presented a best prediction accuracy of 87.57% using ReLU with three hidden layers of 80, 40, 20 neurons ,transformation of X and L2 regularization with C-value of 0.0001 in the original test statistics. Different transformations of data including $\sqrt{X}, X, X^2$ had minor differences that X^2 appears to have slight improvement in performance compared to X while X also performed better than $\sqrt{X}$. The confusion matrices proved similar patterns and precison accuracy data among different transformation patterns. The models with logistic activation functions performed the worst that they had relatively high accuracy when C is low but also faster descending performance with higher C-value. Models with ReLU and Tanh had similar graph patterns and slight difference of prediction accuracy. While Tanh outperformed ReLU in both one and two hidden layers, ReLU had better performance with neural network that had three hidden layers.

Logistic Regression	SVM	Neural Network
83.46%	86.76%	87.57%

In conclusion, all three models gained the test and train accuracy over 80% with similar patterns which proves that the train data and test data are similar to each other. In original papers, maximum accuracy can reach up to $happiness$: 96%, $surprise$: 94%, $anger$: 92%, $sadness$: 88%, $disgust$: 90%, $fear$: 89% using Probabilistic Neural Network Algorithm. Therefore, reaching 80% accuracy in most models fulfilled our expectation. In the Logistic Regression section we have discovered that target variable distribution is quite uniform which could be a possible reason that explains the similar pattern and slight difference between randomly selected train and test data. To improve the prediction accuracy in the future, neural network with more hidden layers and more hyperparameter values could be used to approach to the optimized hyperparamter value and more promising prediction accuracy.

8 Citation

Face Emotion Recognition Dataset:

https://www.openml.org/search?type=data&status=active&id=43602&sort=runs

Optimal Geometrical Set for Automated Marker Placement to Virtualized Real-Time Facial Emotions:

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0149003

sklearn neural network documentation:

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.neural_network

sklearn support vector machine documentation:

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

sklearn logistic regression documentation:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_ model.LogisticRegression.html

Virtual Markers based Facial Emotion Recognition using ELM and PNN Classifiers:

https://ieeexplore.ieee.org/document/9068708