Sign Language Digits Classification Project Report

Se Yeon Carlos Kim
43425461

ITEC873 - Data Science and Machine Learning
Semester 1, 2019

Abstract

In this paper, the Sign Language Digits Classification is explored with the aim of classifying sign language digit images from 0 to 9. A variety of conventional machine learning models are investigated, compared with a deep learning model called *Convolutional Neural Network* (CNN). Results show that CNN performs significantly better than the conventional machine learning models on this dataset. Feature extraction is used to discover distinctive properties of patterns in the data and thus, reduce its dimensions to boost results by up to 0.5%. We conclude that deep learning models in this area outperforms conventional classifying models for this dataset.

1 Introduction

Conventional machine learning and deep learning algorithms has its pros and cons. Conventional machine learning algorithms take data as input, parse the data, and make sense of its decisions based on what it has learned while being trained. It often performs well when the dataset is small, but limited in processing natural data in its raw form. On the other hand, deep learning algorithms take data as input and make intuitive decisions via artificial neural network. It excels in datasets where relationship between the data can be understood in an implicit manner. Hence, deep learning is capable of discovering intricate structures in high-dimensional data such as images and languages.

In this project, a mixture of conventional machine learning algorithms and deep learning models are employed in a *multinomial* classification problem called *Sign Language Digits Classification*. A similar problem which only employs CNN comes from the kaggle.com website[1], a platform where data scientists and machine learning engineers, enter competitions to solve data science challenges. The goal of the project is to compare performance of conventional machine learning models and deep learning models and analyse the difference.

All programs were written in Python[2], and Scikit-Learn and TensorFlow[3] libraries were used to build conventional machine learning and deep learning models respectively.

2 Related work

A popular MNIST handwritten digits dataset provided insights from multiple sources (LeCun et al., 1998). The kaggle.com platform provided tips regarding CNN (Zeka & Bilimi, 2019). Geron (2017) employs Logistic regression to classify two classes on MNIST dataset. Methods from these sources are partially applied for the project.

[1]https://www.kaggle.com/
[2]https://www.python.org/
[3]https://www.tensorflow.org/

3 Data

Sign Language Digits dataset is prepared by *Turkey Ankara Ayranci Anadolu High School* students. The dataset contains 2,062 sign language digits images from 0 to 9. This indicates the dataset consists of 10 classes, with each training image classified as one of 10 possible sign language digits. Original image size is 100 $\times$ 100 pixels, but processed into 64 $\times$ 64 pixels in this project. The possible sign language digits image classes are shown in Figure 1. Further details of the sign language digits dataset is shown in Appendix A, Table 1.

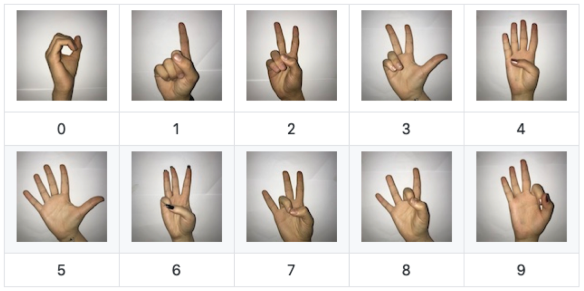

Figure 1: Examples of the data

3.1 Exploration

The dataset contains 2,062 images, randomly dividing the data with 80% training and 20% test data, leading to 1,649 training images and 413 test images. Upon flattening 3 dimensional input array into 2 dimensions, 4,096 features were identified. The original dataset is perfectly balanced indicating that sign language digits are represented equally (Figure 2).

3.2 Feature Extraction

Principal Component Analysis (PCA) was used to discover distinctive properties of patterns in the data. PCA generates linear combinations of features that aim to extract valuable variables from a high dimensional data set (Vidhya, 2016). PCA was employed

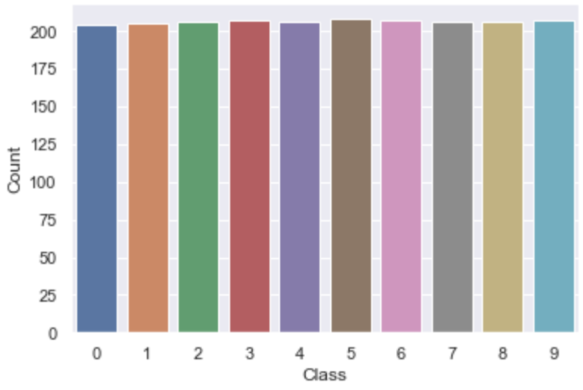

Figure 2: **Bar plot of classes representing equal number of sign language digits images**

on the training set. The graph in Figure 3 shows the adequate number of principal components by measuring against the amount of explained variability. It indicates that 100 and more principal components represent the same data. Therefore, PCA process used 100 principal components.

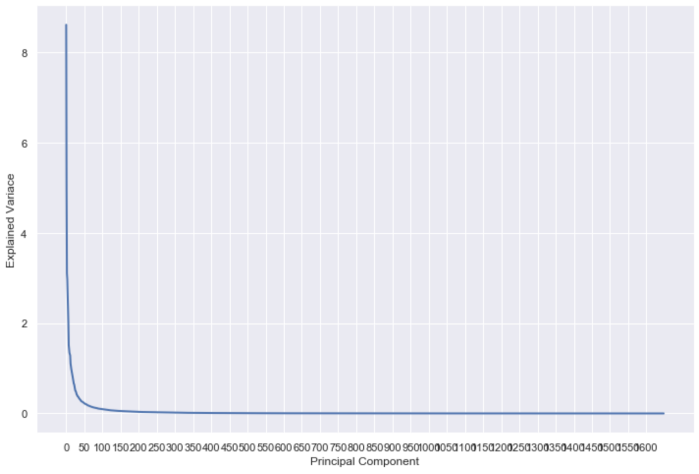

Figure 3: **Plot showing the results of PCA. The horizontal axis shows each principal component and the vertical axis shows the amount of the variance explained by the component.**

4 Methods

Conventional machine learning and deep learning models implemented in the project are explained in this section. It is important to avoid using the test dataset until evaluation process as the test dataset provides an unbiased evaluation of a model fit on the training dataset (Geron, 2017). Hence, test data was untouched during the model tuning process.

4.1 Conventional Machine Learning Models

This section describes classification models used on the training set and the tunings performed for each model. A summary of the models used is shown in Table 2.

Logistic Regression: A common classification algorithm which uses linear combinations of predictors to estimate the probability that an instance belongs to a particular class (Geron, 2017). First logistic regression was implemented with *Ridge regularisation* using Scikit-Learn's [4] *OneVsRestClassifier* package. *One-versus-all* (OVA) approach was employed as we train 10 binary classifiers, one for each class. It learns to return a positive value for the correct class and a negative value for the rest. Second logistic regression was implemented with *Lasso regularisation*, which excludes useless variables from its calculations by shrinking coefficients to 0, while Ridge regularisation shrinks coefficients asymptotically close to 0.

K-Nearest Neighbors (KNN): This model is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve classification problems. KNN is a non-parametric algorithm based on feature similarity, which does not make any assumptions on the underlying data distribution. A drawback of KNN is that the algorithm gets significantly slower as the number of examples and variables increase. When training the model, setting the number of *neighbors* impacts how the model performs. Setting the number of neighbors too high makes the KNN model less sensitive to noise, whereas decreasing the number of neighbors allows capturing finer structure of space. The KNN model for this project selected the number of neighbors by identifying a point when a certain number 'k' produced the least amount of errors. Figure 4 shows that the model performs best when 'k' is set to 5.

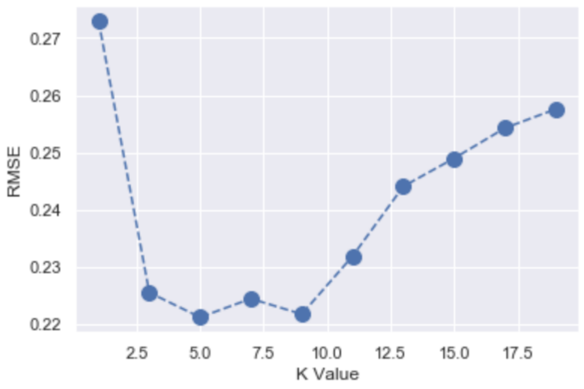

Figure 4: **Plot showing the reduced mean squared error against each k value.**

Decision Tree: A tree-like model of decisions generally used to visually represent decisions, covering both regression and classification. This model is made by taking data from the root node and splitting the data into parts. *Pruning* was necessary as the nature of this model does not generalise the data well. Without pruning, decision tree model results in overfitting. A method of pruning is to set maximum depth of the model. It starts at leaves and removes each node with most popular class in that leaf until it does not deteriorate accuracy. The maximum depth '9' was determined by identifying the least amount of errors as shown in Figure 5.

[4] https://scikit-learn.org/stable/

Model	Tuning
Logistic Regression	Regularization = Ridge regression; OneVsRest Classifier
Logistic Regression	Regularization = Lasso regression; OneVsRest Classifier
K-Nearest Neighbors	N_neighbors = 5
Decision Tree	Max_depth = 9

Table 2: **Models and their hyper-parameter tunings**

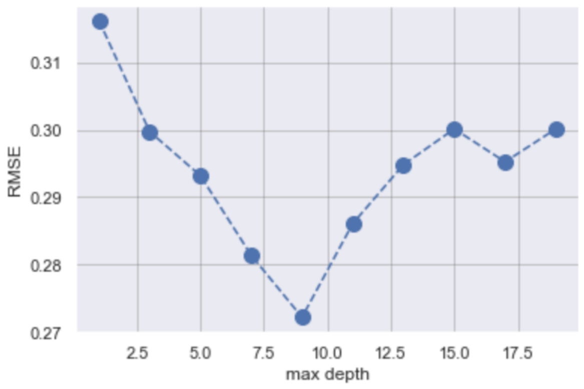

Figure 5: **Plot showing the reduced mean squared error against a value of each maximum depth.**

CNN Model	Architecture
Model	Sequential
Convolutional layer	3 layers, 64 filters, 3 $\times$ 3 kernel size
Activation function	relu
Dropout layer	3 layers of 0.2
Max pooling	pool size = (2,2)
Optimizer	adam
Loss	categorical crossentropy
Dense layer	1 relu, 1 softmax
Other	1 Flatten, 1 Batch normalisation

Table 3: **Convolutional Neural Network model architecture**

4.2 Deep Learning Models

CNN originates from the study of brain's visual cortex detecting complex patterns in visual field. This technique has been used in image recognition since 1980s, and they now power image search services, self-driving cars, automatic video classification, etc (Geron, 2017). A general idea is that the input image builds up to more abstract concepts through a series of *convolutional layers* scanning through an array of pixel values and identifying features.

Standard Convolutional Neural Network without feature extraction was employed initially. A sequential model with 3 convolutional layers, 3 *activation layers*, 3 *max pooling layers*, 3 *dropout layers*, 1 *Batch normalisation* and 1 *Softmax dense layer* was compiled with 10 epochs. Details of the architecture is listed in Table 3.

Convolutional Neural Network with PCA was built using the feature extraction technique PCA, described in section 3.2 Feature Extraction. Theoretically, PCA removes unnecessary features and hence, improves the overall performance. CNN model architecture is the same as the standard CNN model in table 3, except the training data being transformed with 100 principal components.

5 Results

5.1 Conventional Machine Learning Results

Table 4 lists the key accuracy results from the previously described conventional machine learning models on both training and the test data sets. For this section, the KNN model produced the best results, with the number of neighbors set to 5, producing the most accurate predictions. The best result of 58.35% correct classification rate on the test data and 71.86% on the training data. Figure 6 shows the Confusion Matrix of the

Classifier	Accuracy (Training)	Accuracy (Test)	Best Tuning
Logistic Regression	96.30%	52.52%	Ridge regularization
Logistic Regression	76.83%	47.94%	Lasso regularization
K-Nearest Neighbors	71.86%	58.35%	N_neighbors = 5
Decision Tree	78.29%	54.24%	Max_depth = 9

Table 4: **Summary of results from conventional machine learning models**

KNN model, indicating most errors are misclassifying other sign language digits as 0.

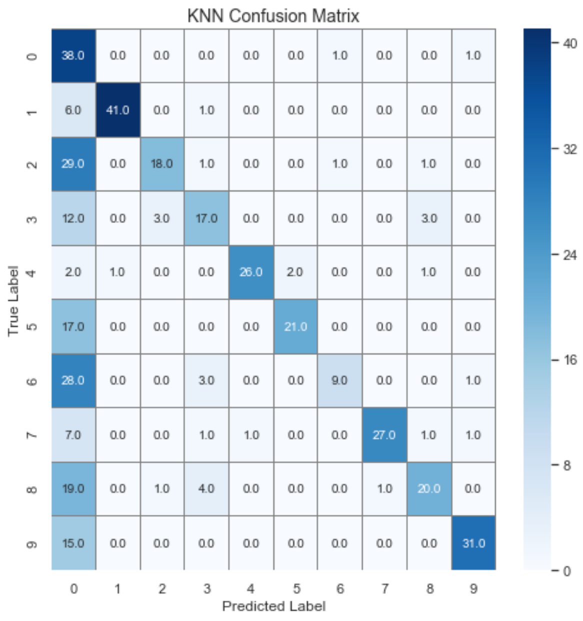

Figure 6: **Confusion matrix of KNN**

Decision Tree model achieved the second highest accuracy result with 54.24% test accuracy and 78.29% training accuracy. Selecting the maximum depth of the tree was critical as the model produced 100% accuracy on the training set without pruning, showing a perfect example of overfitting. Similar to the KNN model, Confusion Matrix of the Decision Tree model in Figure 7 shows misclassification of other sign language digits as 0.

The linear models were the worst performing, suggesting the features do not form a linear relationship. Logistic regression with Ridge regularisation produced 52.52% on the test set and 96.30% on training set. Despite using regularisation technique, the model was clearly overfitting. Logistic regression with Lasso regularisation performed better than the model with Ridge regularisation in terms of avoiding overfitting, producing 47.94% on the test set and 76.83% on the training set. This indicates that the data contains a substantial number of useless features as the Lasso regularisation excludes redundant variables from equations. Confusion Matrix of Logistic regression with Lasso regularisation is displayed in Figure 8 with the same issue. Confusion Matrix with Ridge regularisation is shown in Appendix A, Figure 9. ROC curve and ROC area for each class are not examined in this section, but

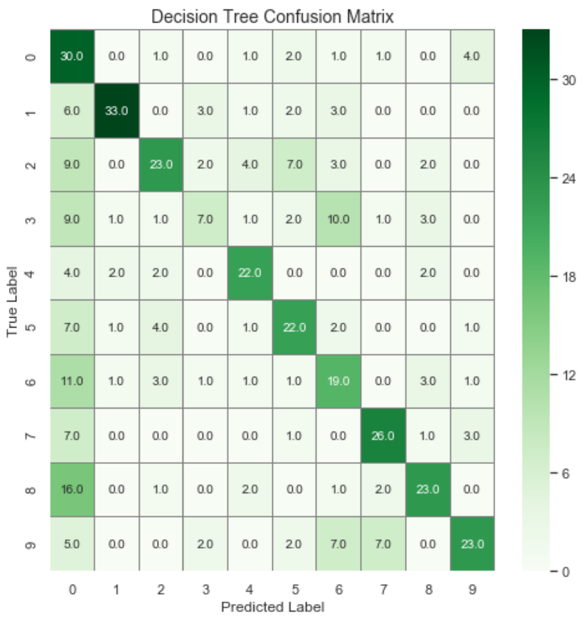

Figure 7: **Confusion matrix of Decision Tree**

included in Appendix A, Figure 15 and Figure 16 for reference.

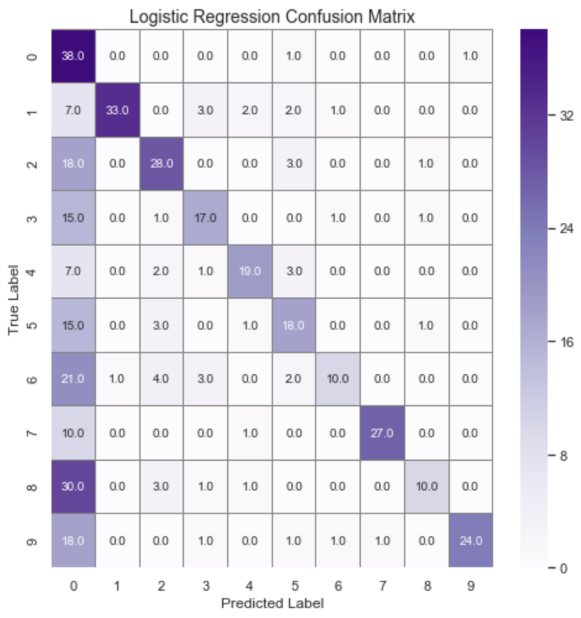

Figure 8: **Confusion matrix of Logistic Regression with Lasso regularisation**

5.2 Deep Learning Results

CNN models displayed outstanding results, producing the most accurate predictions. Standard CNN achieved 93.95% on the test set and 96.91% on the training set. Figure 9 shows that both validation and training accuracy scores continue to increase without overfitting the model and Figure 10 shows the decrease of validation and training losses. This means that we can increase the number of epochs until the the losses begin to increase, and thus, achieve higher accuracy scores.

Confusion Matrix in Figure 11 displays that the CNN model no longer suffers from the misclassification of 0 digit image issue.

The PCA feature engineering performed CNN model was expected to produce even higher accuracy scores. Surprisingly, the validation accuracy score increased by only 0.5% (Table 5).

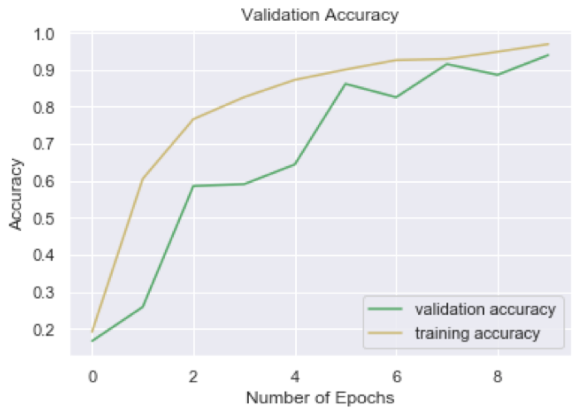

Figure 9: **Standard CNN validation Accuracy plot showing both training and validation accuracy scores continue to increase**

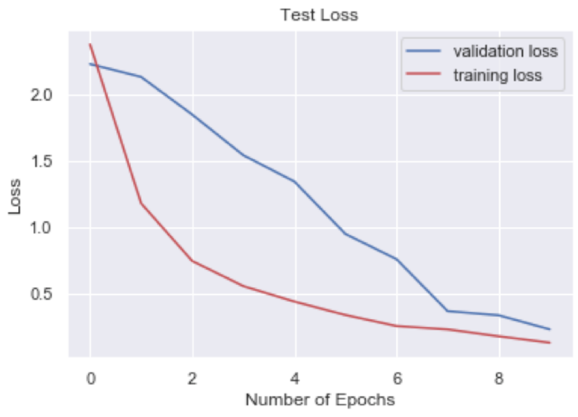

Figure 10: **Standard CNN validation loss plot showing both training and validation losses continue to decrease**

Interesting point to note from PCA CNN model is that the gaps between training and validation losses as well as accuracy scores are more narrow than the standard CNN model (Figure 12 & 13).

5.3 Conventional Machine Learning vs Deep Learning

With only 10 epochs and 3 convolutional layers performed extremely well compared to all 4 conventional machine learning models. Conventional machine learning models pro-

Classifier	Accuracy (Training)	Accuracy (Test)
Standard CNN	96.91%	93.95%
PCA CNN	97.70%	94.42%

Table 5: **Summary of results from CNN models**

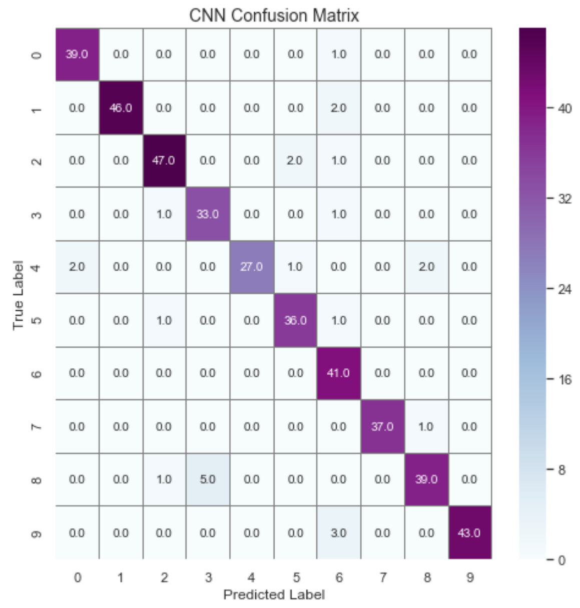

Figure 11: **Standard CNN Confusion matrix**

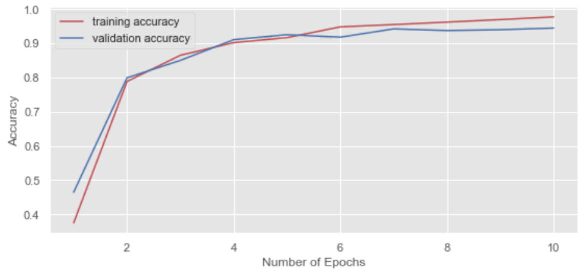

Figure 12: **PCA CNN validation Accuracy plot showing both training and validation accuracy scores continue to increase**

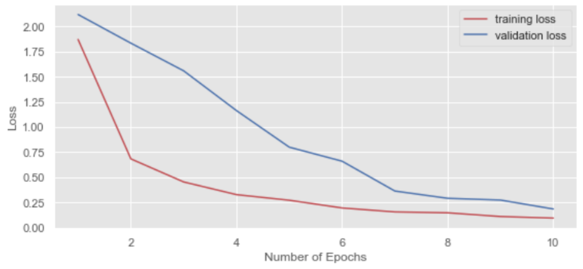

Figure 13: **PCA CNN validation loss plot showing both training and validation losses continue to decrease**

duced test accuracy scores ranging 47 - 58% while overfitting the models. Misclassification of sign language digit '0' was a prominent issue across all conventional machine learning models. When the CNN was employed, this issue vanished as shown in Figure 11. Pros of conventional machine learning was easy-implementation, comprehensive algorithms, and efficient training process.

6 Conclusion

In this paper we evaluated a number of multinomial classification models on the Sign Language Digits image dataset. Deep learning models performed significantly better than the conventional machine learning models, particularly Convolutional Neural Network with Principal Component Analysis feature extraction. Feature engineered CNN model produced final accuracy of 94.42%. Conventional machine learning algorithms such as Logistic Regression with regularisation, K-Nearest Neighbors, Decision Tree failed to produce satisfactory results due to poor accuracy and overfitting. Although CNN is well known for producing outstanding results for image processing, this paper proved that conventional machine learning algorithms were not suitable for the sign language digits dataset. Further work on using larger dataset would improve the accuracy for deep learning models as well as training models with augmented images.

7 Appendix A

Image size	100 × 100
Colour space	RGB
Number of classes	10 (Digits:
Number of participant students	218
Number of samples per student	10

Table 1: Details of dataset

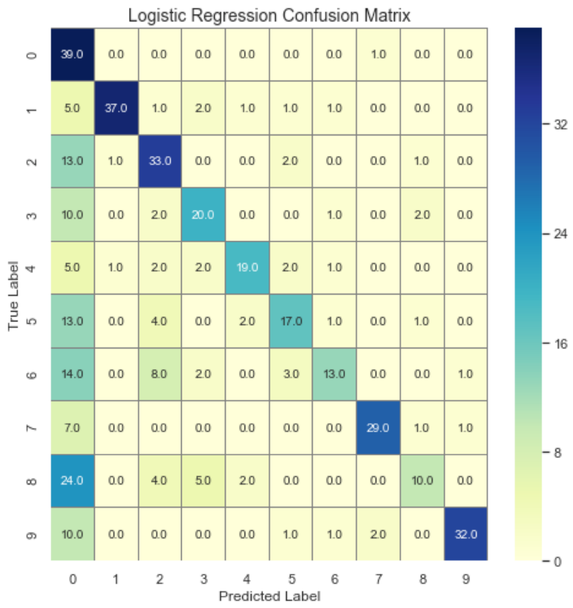

Figure 14: **Confusion matrix of Logistic Regression with Ridge regularisation**

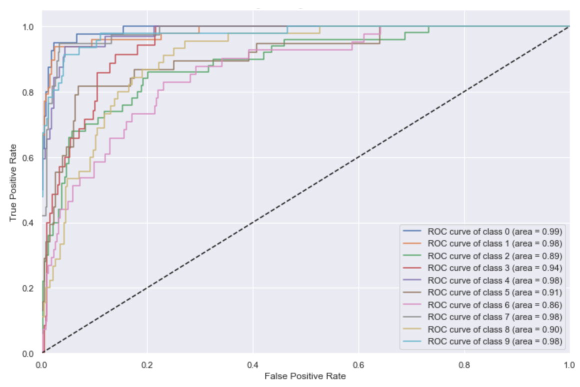

Figure 15: **ROC curve and ROC area for each class for Logistic Regression with Ridge regularisation**

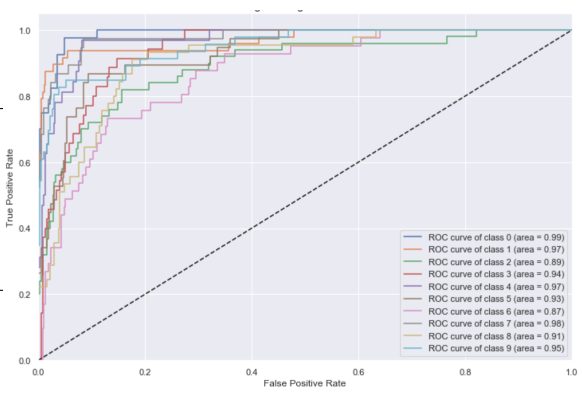

Figure 16: **ROC curve and ROC area for each class for Logistic Regression with Lasso regularisation**

References

Analytics Vidhya Content Team 2016, *'Practical guide to Principal Component Analysis (PCA) in R & Python'*, viewed 30 May 2019, available at <https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/>.

Ciresan, D., Meier, U., and Schmidhuber, J. 2012, 'Multi-column deep neural networks for image classification', *Institute of Electrical and Electronics Engineers (IEEE)*, pp. 3642–3649, February 2012.

Geron, A. 2017, *'Hands-on machine learning with Scikit-Learn & TensorFlow'*, Sebastopol, O'Reilly.

Lecun,Y., Bottou, L., Bengio, Y., and Haffner, P. 1998, 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE*, pp. 2278-2324, November 1998.

Zeka,Y. and Bilimi, V. 2019, Deep learning tutorial for beginners, viewed 20 May 2019, available at <https://www.kaggle.com/kanncaa1/deep-learning-tutorial-for-beginners>.