CSC311 HOMEWORK 3

Q1.

(a)

	Dimension
$W^{(1)}$	d×d
$W^{(2)}$	1×d
z_1	d×1
z_2	d×1

(b)

The number of parameters is the number of elements in the weights so is d^2+d

(c)

$$\bar{y} = \frac{\partial \mathcal{L}}{\partial y} = y - t$$

$$\overline{W^{(2)}} = \frac{\partial \mathcal{L}}{\partial W^{(2)}} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial W^{(2)}} = \bar{y} \cdot z_2^T = (y-t)z_2^T$$

$$\bar{z_2} = \frac{\partial \mathcal{L}}{\partial z_2} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial z_2} = \bar{y} \cdot {W^{(2)}}^T = {W^{(2)}}^T(y-t)$$

$$\bar{h} = \frac{\partial \mathcal{L}}{\partial h} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial h} = \bar{z_2} = {W^{(2)}}^T(y-t)$$

$$\bar{z_1} = \frac{\partial \mathcal{L}}{\partial z_1} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial h}\frac{\partial h}{\partial z_1} = \bar{h} \cdot \sigma'(z_1) = {W^{(2)}}^T(y-t) \circ \sigma'(z_1)$$

$$\overline{W^{(1)}} = \frac{\partial \mathcal{L}}{\partial W^{(1)}} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial h}\frac{\partial h}{\partial z_1}\frac{\partial z_1}{\partial W^{(1)}} = \bar{z_1}x^T = {W^{(2)}}^T(y-t) \circ \sigma'(z_1)x^T$$

$$\bar{x} = \frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial z_2}\frac{\partial z_2}{\partial x} = \bar{z_2}\left(\frac{\partial h}{\partial x} + I\right) = \bar{z_2}\left(\frac{\partial h}{\partial z_1}\frac{\partial z_1}{\partial x} + I\right) = \bar{z_2}\left({W^{(1)}}^T\sigma'(z_1) + I\right)$$

$$= {W^{(2)}}^T(y-t) \circ {W^{(1)}}^T\sigma'(z_1) + {W^{(2)}}^T(y-t)$$

Q2.

(a)

Since $y_k = \frac{\exp(z_k)}{\sum_{k'=1}^{K}\exp(z_k')}$

$$\frac{\partial y_k}{\partial z_k{}'} = \frac{\frac{\partial \exp(z_k)}{\partial z_k{}'}\sum_{k'=1}^{K}\exp(z_k') - \exp(z_k)\frac{\partial\left(\sum_{k'=1}^{K}\exp(z_k')\right)}{\partial z_k{}'}}{\left(\sum_{k'=1}^{K}\exp(z_k')\right)^2}$$

We also let $y_k{}' := \frac{\exp(z_k')}{\sum_{k'=1}^{K}\exp(z_k')}$

Case 1: k = k'

Since k = k', $\frac{\partial \exp(z_k)}{\partial z_{k'}} = \frac{\partial \exp(z_k)}{\partial z_k} = \exp(z_k), \frac{\partial\left(\sum_{k'=1}^{K}\exp(z_k')\right)}{\partial z_{k'}} = \frac{\partial \exp(z_{k'})}{\partial z_{k'}} = \exp(z_k{}') = \exp(z_k)$

$$\frac{\partial y_k}{\partial z_k{}'} = \frac{\exp(z_k)\sum_{k'=1}^{K}\exp(z_k') - \exp(z_k)\exp(z_k)}{\left(\sum_{k'=1}^{K}\exp(z_k')\right)^2}$$

$$= \frac{\exp(z_k)}{\sum_{k'=1}^{K}\exp(z_k')} - \left(\frac{\exp(z_k)}{\sum_{k'=1}^{K}\exp(z_k')}\right)^2 = y_k - y_k^2$$

Case 2: k ≠ k'

Since $\frac{\partial \exp(z_k)}{\partial z_{k'}} = 0, \frac{\partial\left(\sum_{k'=1}^{K}\exp(z_k')\right)}{\partial z_{k'}} = \frac{\partial \exp(z_{k'})}{\partial z_{k'}} = \exp(z_k{}')$

$$\frac{\partial y_k}{\partial z_k{}'} = \frac{-\exp(z_k)\exp(z_k{}')}{\left(\sum_{k'=1}^{K}\exp(z_k')\right)^2} = -\frac{\exp(z_k)}{\sum_{k'=1}^{K}\exp(z_k')}\frac{\exp(z_k')}{\sum_{k'=1}^{K}\exp(z_k')} = -y_k y_k{}'$$

(b)

Since $\mathcal{L}_{\mathrm{CE}}(t,y) = -\mathrm{t}^{\mathrm{T}}\log(y) = -\sum_{n=1}^{N} t_n \log(y_n)$

$$\frac{\partial \mathcal{L}_{\mathrm{CE}}(t,y)}{\partial w_k} = \sum_{n=1}^{N}\frac{\partial \mathcal{L}_{\mathrm{CE}}(t,y)}{\partial y_n}\frac{\partial y_n}{\partial z_k}\frac{\partial z_k}{\partial w_k}$$

As $\frac{\partial \mathcal{L}_{\mathrm{CE}}(t,y)}{\partial y_n} = \frac{\partial\left(-\sum_{n=1}^{N} t_n \log(y_n)\right)}{\partial y_n} = \frac{\partial(-t_n\log(y_n))}{\partial y_n} = -\frac{t_n}{y_n}$,

$$\frac{\partial \mathcal{L}_{\mathrm{CE}}(t,y)}{\partial w_k} = -\sum_{n=1}^{N}\frac{t_n}{y_n}\frac{\partial y_n}{\partial z_k}\boldsymbol{x}$$

By (a)

$$\frac{\partial \mathcal{L}_{\mathrm{CE}}(t,y)}{\partial w_k} = -\sum_{n=1}^{N}\frac{t_n}{y_n}\frac{\partial y_n}{\partial z_k}\boldsymbol{x}$$

$$= -\sum_{n=1\cap n\neq k}^{N}\frac{t_n}{y_n}\frac{\partial y_n}{\partial z_k}\boldsymbol{x} - \frac{t_k}{y_k}\frac{\partial y_k}{\partial z_k}\boldsymbol{x} = \sum_{n=1\cap n\neq k}^{N}\frac{t_n}{y_n}y_n y_k\boldsymbol{x} - \frac{t_k}{y_k}(y_k - y_k^2)\boldsymbol{x}$$

$$= \left(y_k\sum_{n=1\cap n\neq k}^{N} t_n - t_k(1-y_k)\right)\boldsymbol{x} = \left(y_k\sum_{n=1\cap n\neq k}^{N} t_n + y_k t_k - t_k\right)\boldsymbol{x} = \left(y_k\sum_{n=1}^{N} t_n - t_k\right)\boldsymbol{x}$$

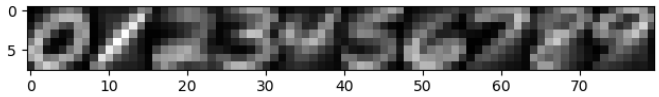
Since t is a one-hot encoding of the output, $\sum_{n=1}^{N} t_n = 1$.

Therefore, $\frac{\partial L_{\mathrm{CE}}(t,y)}{\partial w_k} = (y_k - t_k)\boldsymbol{x}$

Q3.

3.0

The means for each of the digit classes in the training data:



3.1.1

K = 1	Train classification accuracy	1
	Test classification accuracy	0.9688
K = 15	Train classification accuracy	0.9597
	Test classification accuracy	0.9593

3.1.2

I construct a list containing all ties and randomly choose one of them to be the predicted digit.

I choose this method since we don't know which tie is preferred from the data and this method equal possibly chose each tie to be the prediction which prevent any preference on ties and reduce bias.

3.1.3

The optimal K is 1

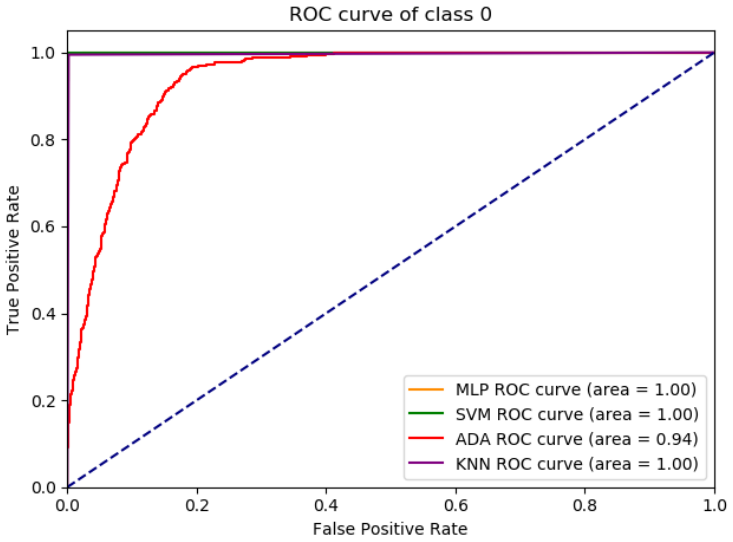
Its train classification is 1.0

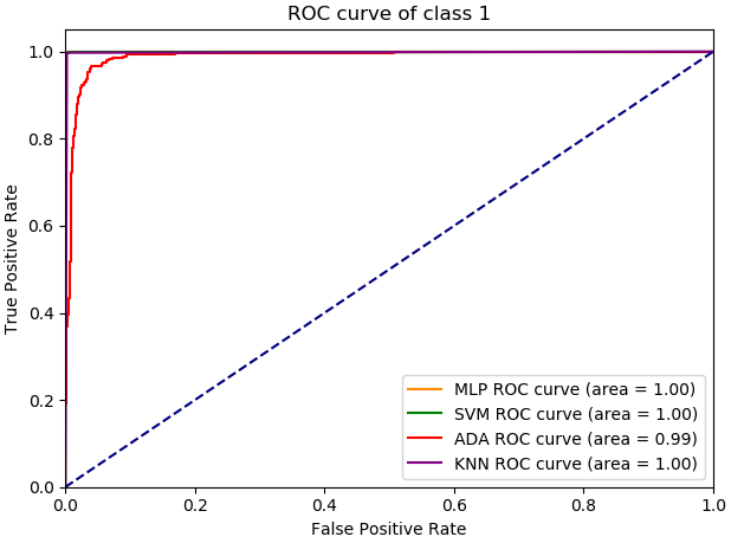
Its average accuracy across folds is 0.9644285714285715

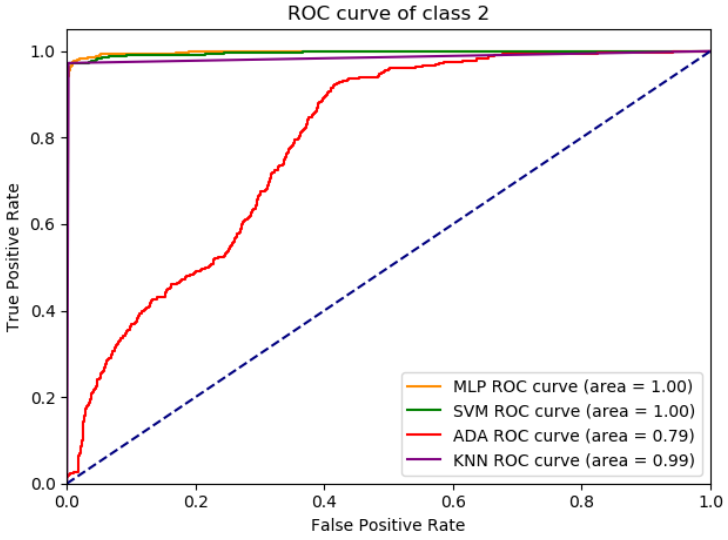
Its test accuracy is 0.96875

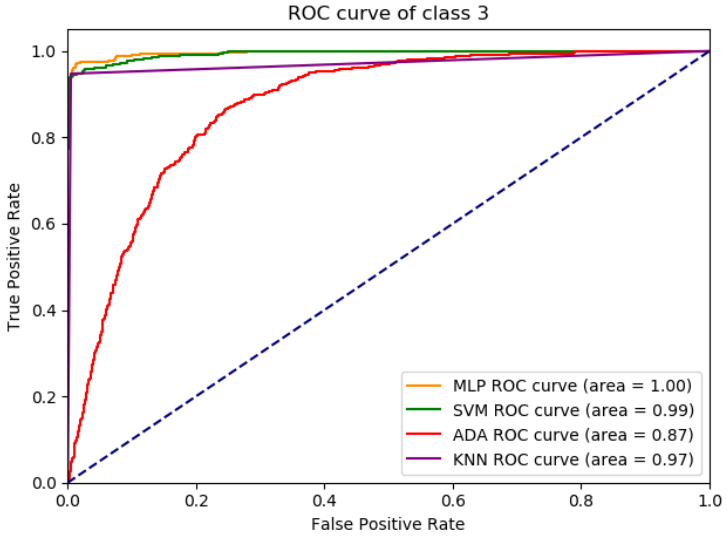
3.3

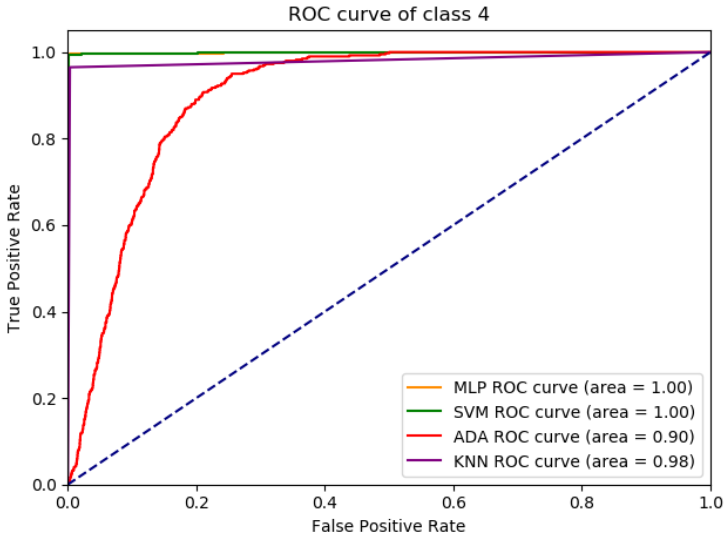
1.ROC curve

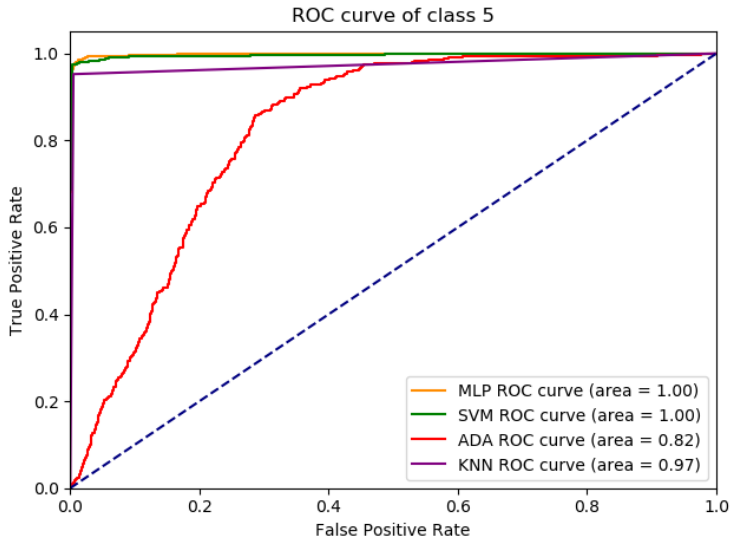


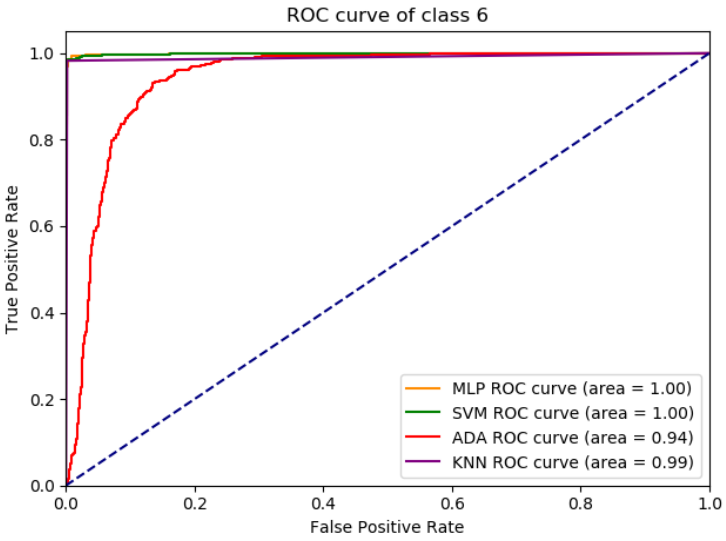


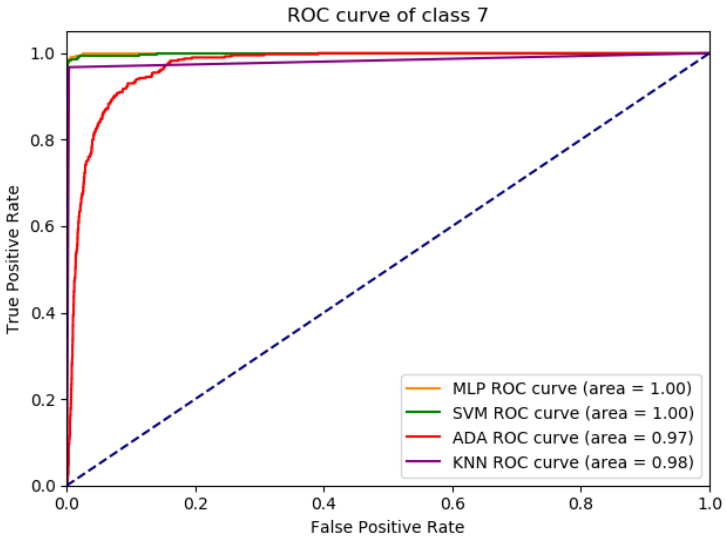


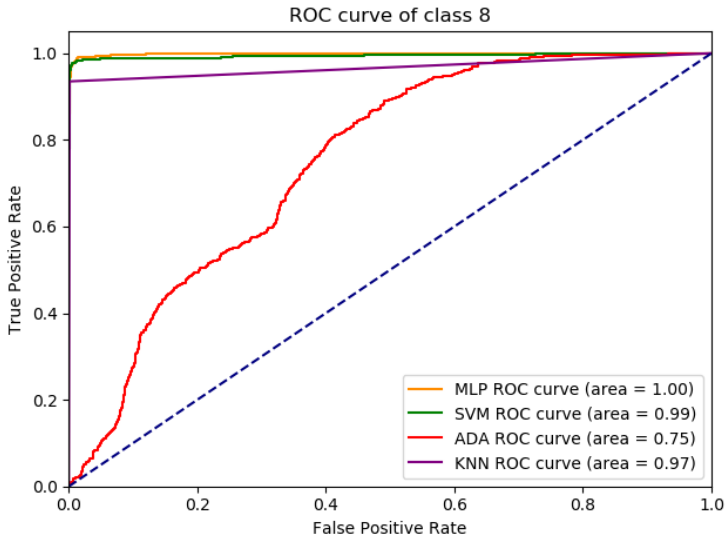


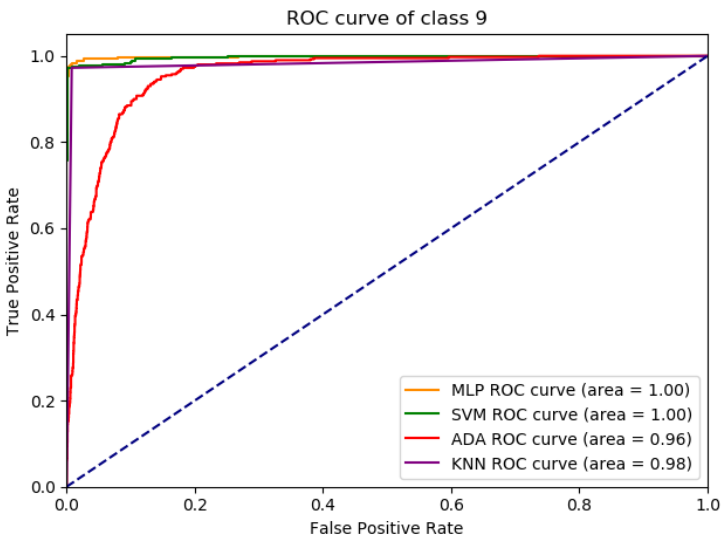












2. Confusion matrix (for each digit from 0 to 9)

MLP:

```
[[398   0   0   0   0   0   1   0   1   0]
 [  0 400   0   0   0   0   0   0   0   0]
 [  1   1 381   3   1   3   6   1   3   0]
 [  1   0   8 374   0   8   0   1   5   3]
 [  0   0   0   0 398   0   2   0   0   0]
 [  2   1   1   7   0 387   0   1   1   0]
 [  1   2   3   0   1   2 391   0   0   0]
 [  0   0   2   0   1   0   0 389   0   8]
 [  1   0   0   5   0   3   1   1 388   1]
 [  0   1   0   3   3   1   1   3   3 385]]
```

SVM:

```
[[400   0   0   0   0   0   0   0   0   0]
 [  0 399   0   0   0   0   1   0   0   0]
```

```
 [  0   0 390   2   0   2   4   1   1   0]
 [  0   1   6 377   0   7   0   2   6   1]
 [  0   0   0   0 397   0   2   0   0   1]
 [  1   0   0   4   0 390   2   1   2   0]
 [  1   0   1   0   3   0 395   0   0   0]
 [  0   0   1   0   5   0   0 391   0   3]
 [  1   0   0   4   0   5   0   0 388   2]
 [  0   1   0   2   3   0   0   4   1 389]]
```

ADA:

```
 [[345   1   8  10   1   7  12   0  16   0]
 [  1 344  11   7  21   4   0   0  12   0]
 [ 50   2 253  15   9  22  22   1  25   1]
 [  3   0  30 320   0  28   0   0  16   3]
 [  2   6   2   0 318   0  17   4   7  44]
 [  6   4   9  43   8 311   5   1  12   1]
 [166   1  17   1   7  25 179   0   4   0]
 [  0   4   5  18   4   1   0 291   6  71]
 [ 36   3  35  12   4  16   1   1 283   9]
 [  0   1   2   5  24   0   0  31  11 326]]
```

KNN:

```
 [[398   0   0   0   0   0   1   1   0   0]
 [  0 399   1   0   0   0   0   0   0   0]
 [  4   0 389   3   1   0   0   1   1   1]
 [  0   1   4 379   0  11   1   2   1   1]
 [  0   0   0   0 386   0   2   2   0  10]
 [  1   0   0  12   0 381   3   1   2   0]
 [  0   4   2   0   0   0 393   0   1   0]
 [  0   1   1   0   3   0   0 387   0   8]
 [  2   2   1   2   1   7   0   2 374   9]
 [  0   0   0   1   7   0   0   3   0 389]]
```

3. Accuracy

MLP: 0.97275

SVM: 0.979

ADA: 0.7425

KNN: 0.96875

4. Precision (for each digit from 0 to 9)

MLP:

```
[0.98514851 0.98765432 0.96455696 0.95408163 0.98514851 0.95792079
 0.97263682 0.98232323 0.96758105 0.9697733 ]
```

SVM:

```
[0.99255583 0.99501247 0.9798995  0.96915167 0.97303922 0.96534653
```

0.97772277 0.97994987 0.97487437 0.98232323]
ADA:
[0.56650246 0.93989071 0.68010753 0.7424594 0.8030303 0.75120773
0.75847458 0.88449848 0.72193878 0.71648352]
KNN:
[0.98271605 0.98034398 0.97738693 0.95465995 0.96984925 0.95488722
0.9825 0.96992481 0.98680739 0.93062201]

5. Recall (for each digit from 0 to 9)
MLP: [0.995 1. 0.9525 0.935 0.995 0.9675 0.9775 0.9725 0.97 0.9625]
SVM: [1. 0.9975 0.975 0.9425 0.9925 0.975 0.9875 0.9775 0.97 0.9725]
ADA: [0.8625 0.86 0.6325 0.8 0.795 0.7775 0.4475 0.7275 0.7075 0.815]
KNN: [0.995 0.9975 0.9725 0.9475 0.965 0.9525 0.9825 0.9675 0.935 0.9725]

ROC shows that Multi-Layer Perceptron Neural Network and SVM classifier are the best classifiers compared to the random guessing which is the diagonal line while AdaBoost classifier is closet to random guessing. This is the most obvious for digits 2, 5 and 8.
This is as expected as AdaBoost classifier starts from a weak classifier which is slightly better than random guessing and improves it by learning from the training data. From some digits, the learning may not improve the performance then the result is close to random guessing compared to other classifiers.
The columns of confusion matrix represent true labels/digits and the rows represent predicted labels/digits by the classifier. The diagonal elements in the confusion matrix are the number of correctly predicted data points for each digit. MLP and SVM have the largest diagonal elements while ADA has the smallest indicating that MLP and SVM have the largest number of correctly predicted data points while ADA has the least.
Accuracy shows that the percentage of correctly predicted data points is the largest for Multi-Layer Perceptron Neural Network and SVM classifier while the smallest for AdaBoost classifier.
Precision for each digit calculates the proportion of positive predictions that are actually correct. Recall for each digit calculates the proportion of actual positives that were identified correctly. Both the precision and recall for ADA of each digit are significantly smaller than other classifiers. This indicates that the class is not so correctly recognized by ADA and an example labelled as a specific digit by ADA is not so indeed that digit compared to other classifiers.
An interesting thing to observed is that the recall of digit 1 for MLP and digit 0 for SVM are 1, indicating that all actual 1s is correctly classified by MLP and all actual 0s is correctly classified by SVM.

All the metrics and plots show that Multi-Layer Perceptron Neural Network and SVM classifier are the best classifiers for Handwritten Digit Classification while AdaBoost Classifier performed worst. Since our dataset has a large size, it is expected that MLP and SVM perform well. AdaBoost's performance depends on the choice of a base learner and whether the weak learning assumption is satisfied in the provided dataset. So maybe for this dataset, it happens to perform badly. Also, AdaBoost is less possible to be overfitted compared to other classifiers, this may also be the reason that it doesn’t perform so well.