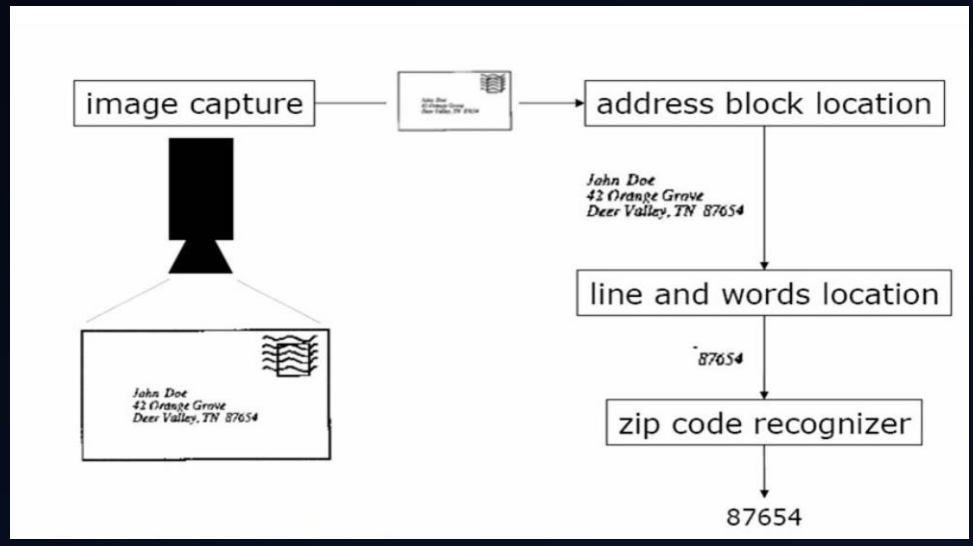
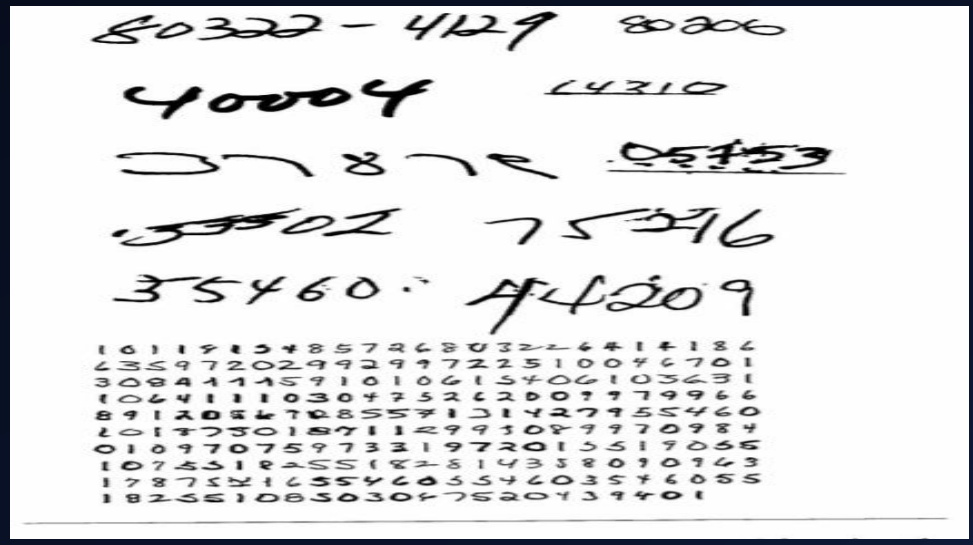
DIGIT RECOGNIZER

BY NEHA GUPTA

Introduction

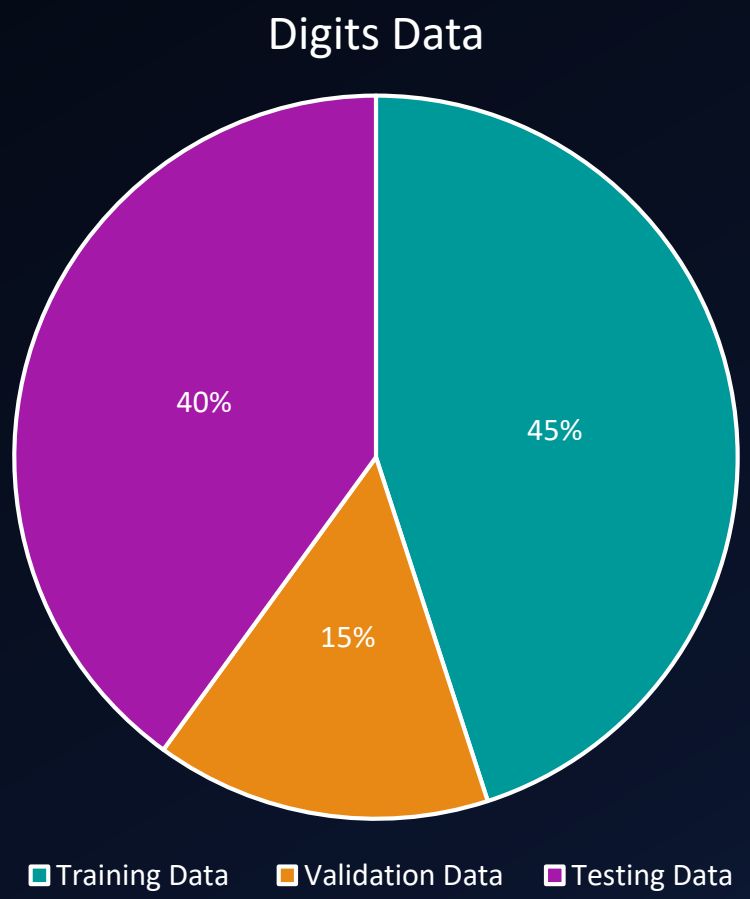
- Machines read handwritten digits
- In 1989, Yann LeCun built a system which was eventually used to read handwritten checks and zip codes by companies.
- The system he built was also successfully applied to the recognition of handwritten zip code digits provided by the U.S. Postal Service





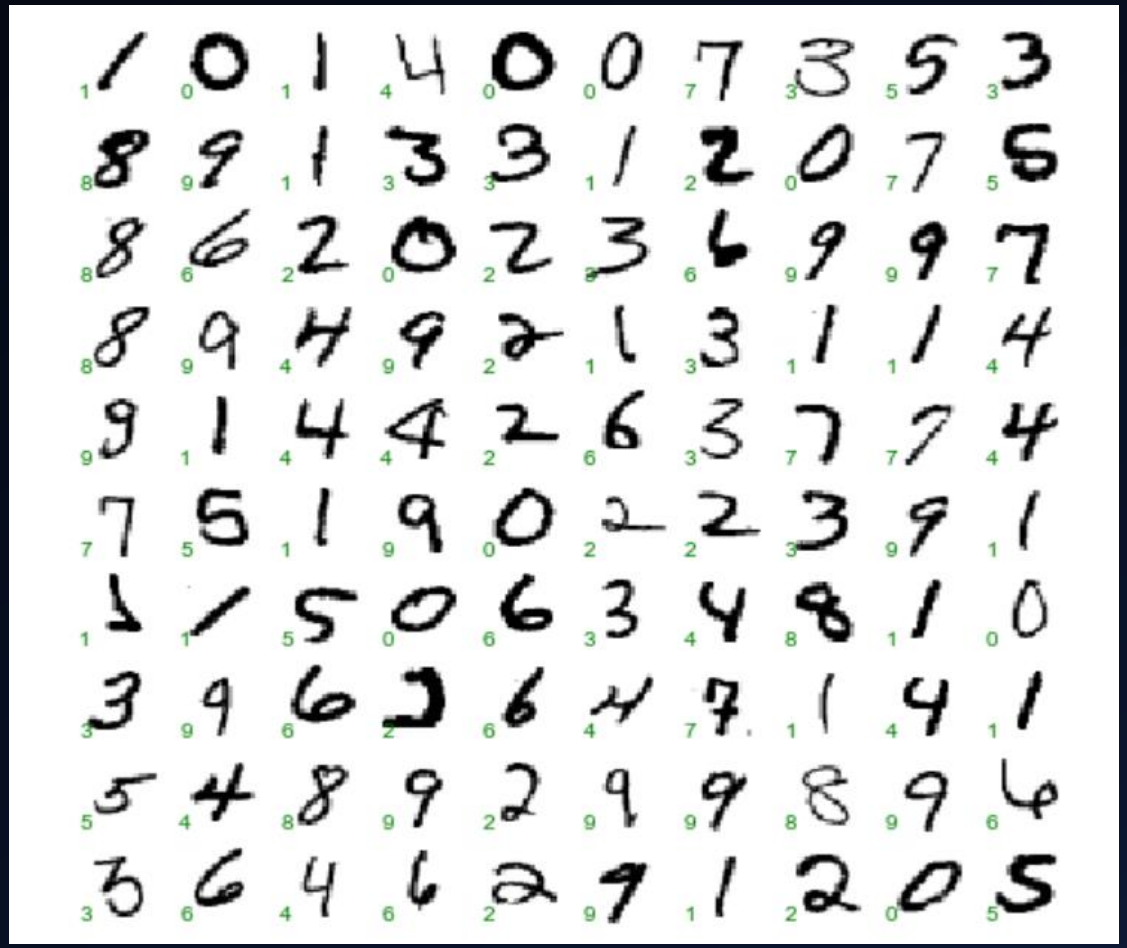
Dataset

- Source: https://www.kaggle.com/c/digit-recognizer
- Training Data:
  - Rows: 31500
  - Columns : 785
- Validation Data:
  - Rows: 10500
  - Columns : 785
- Testing Data:
  - Rows: 28000
  - Columns : 784



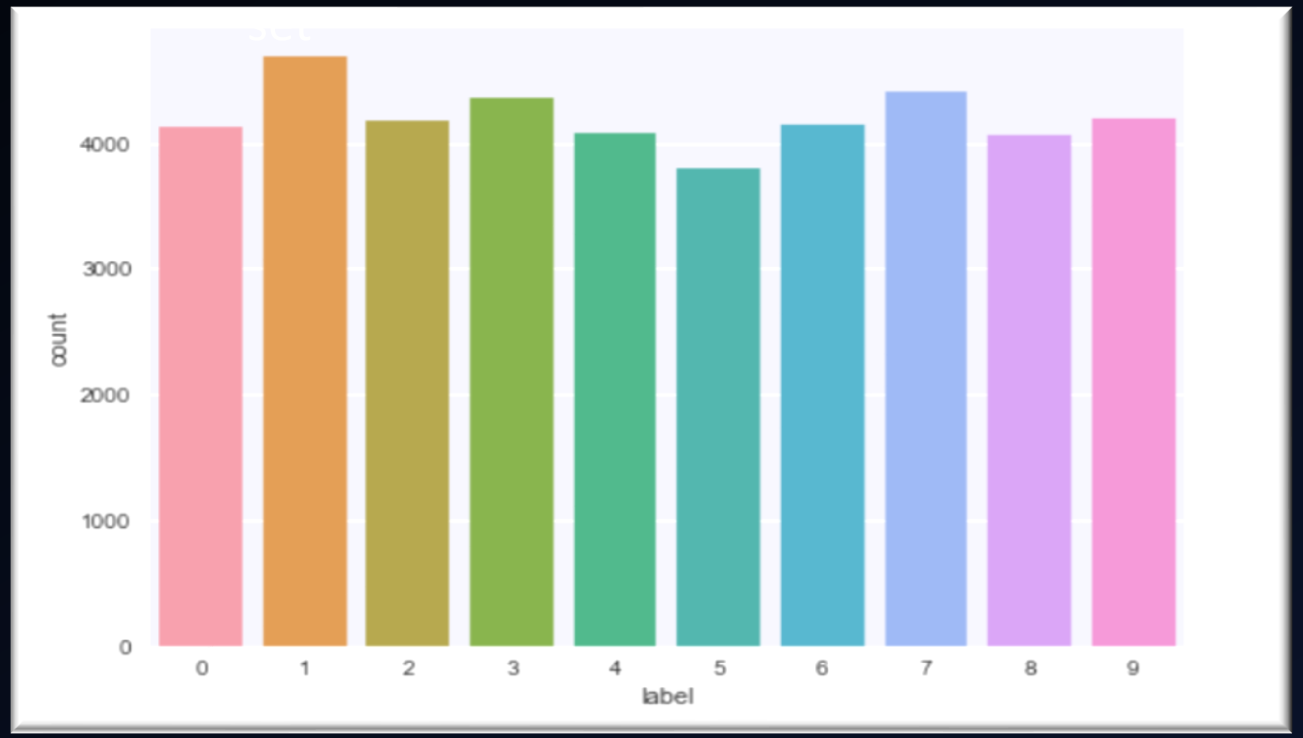
Label & Features

- Label(y) : The digit. The first column in dataset
- Features(X) : The flattened pixels. All the columns except the first.



Data Distribution

Digits data distribution in training data

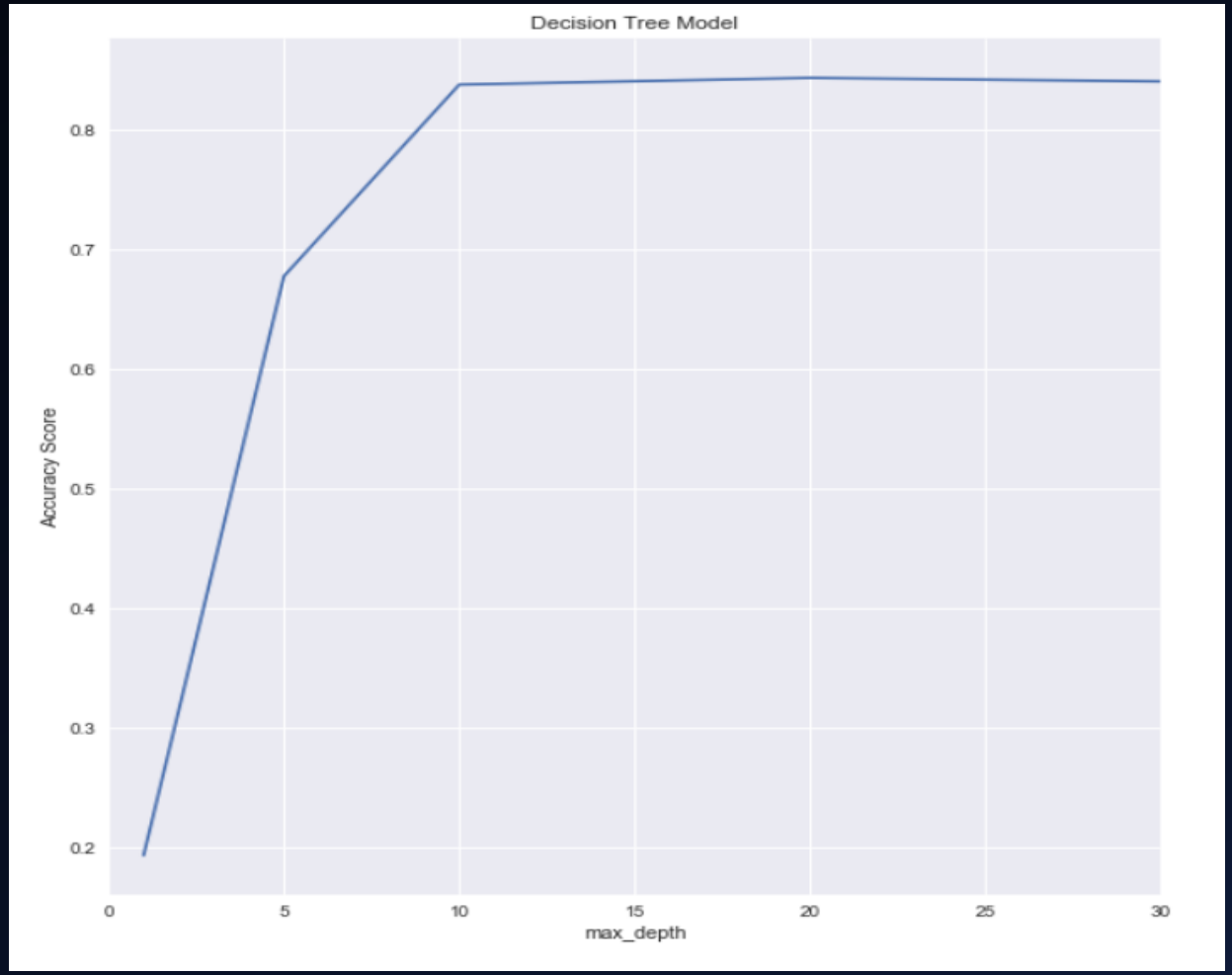


Machine Learning Model Evaluation

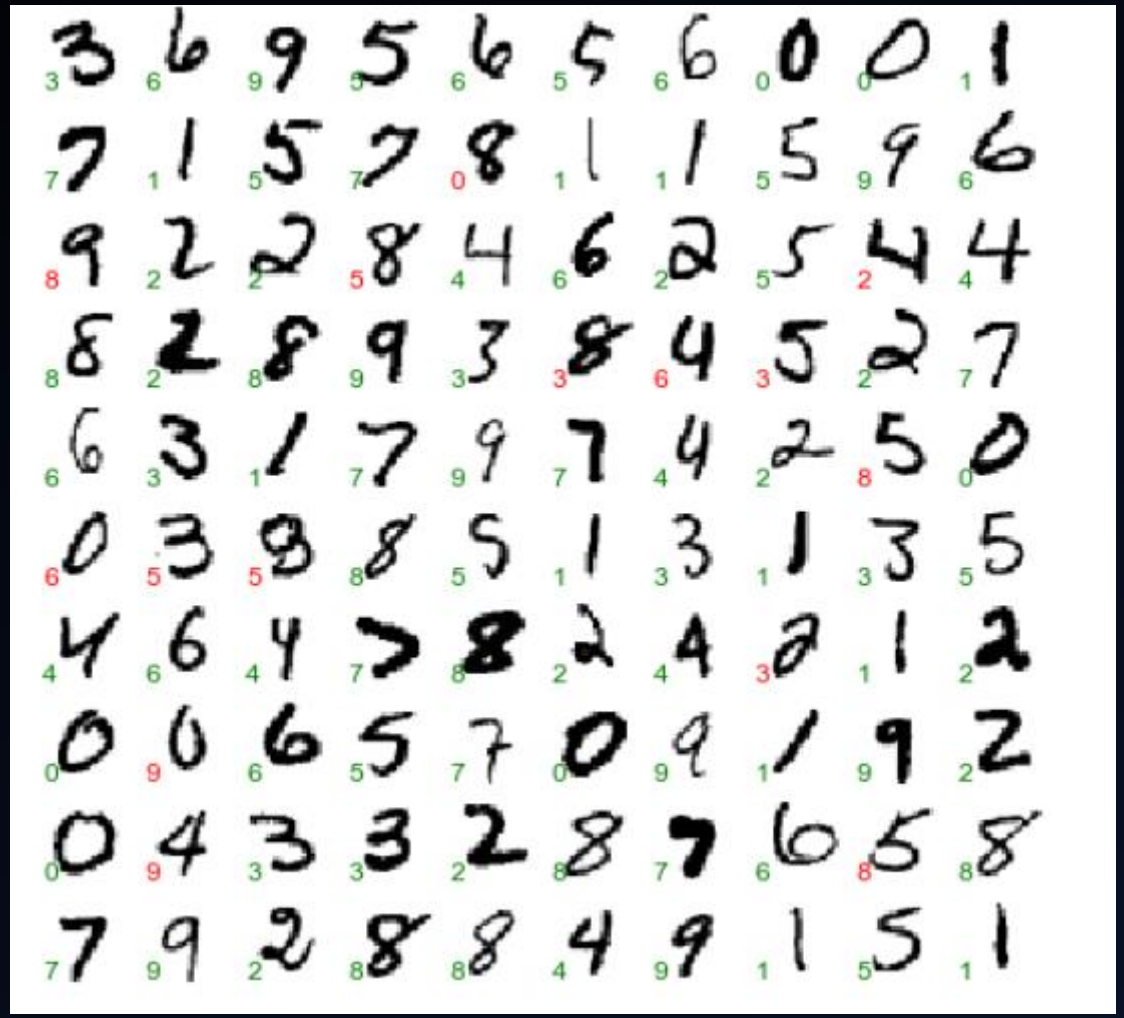
- Decision Tree
- Logistic Regression
- Random Forest
- K-Nearest Neighbors (KNN)
- Support Vector Machines (SVM)

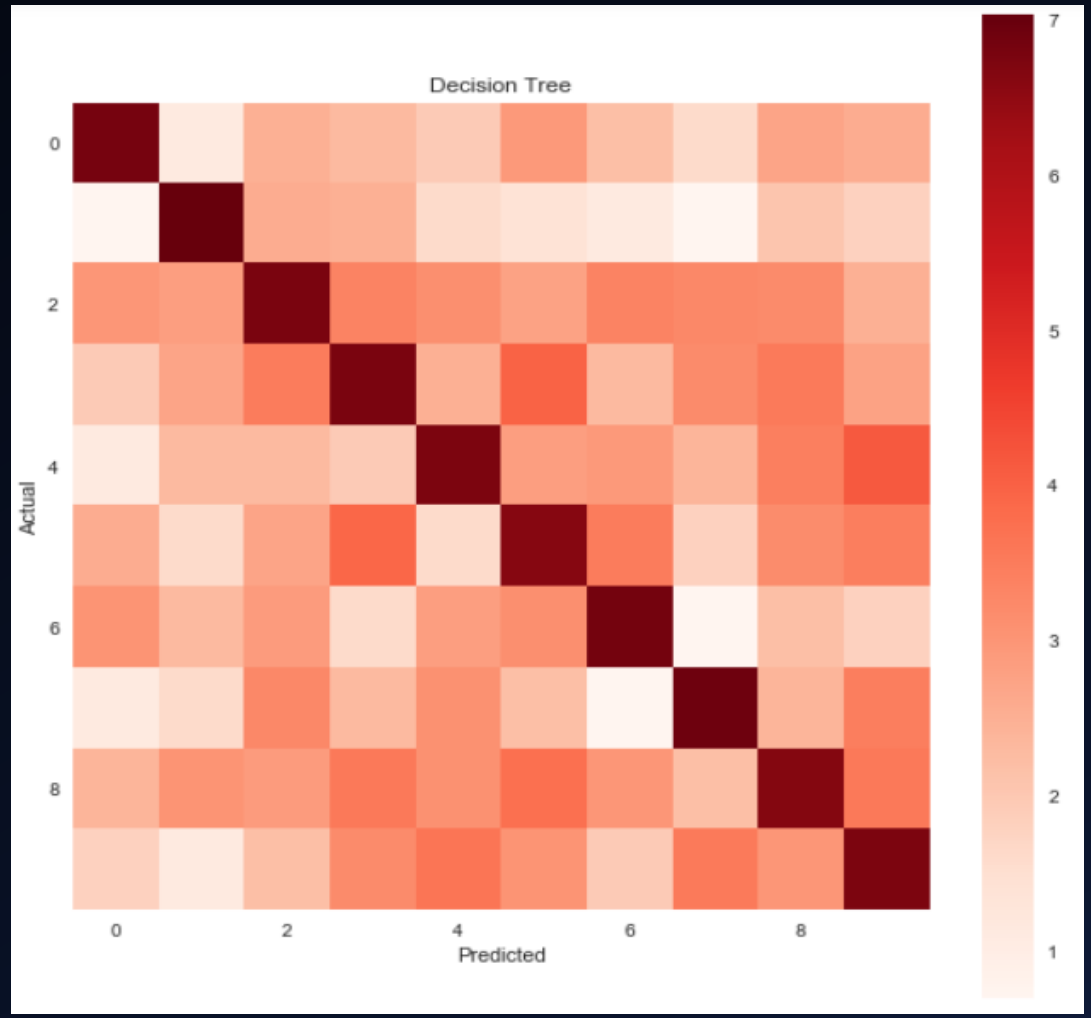
Decision Trees – Cross Validation

- Tuned the max_depth with 5-fold cross-validation
- Checked for max_depth in (1,5,10,20,30,40,50,60,70,80,90,100)
- Achieved optimal accuracy of 85.61% at max_depth = 10

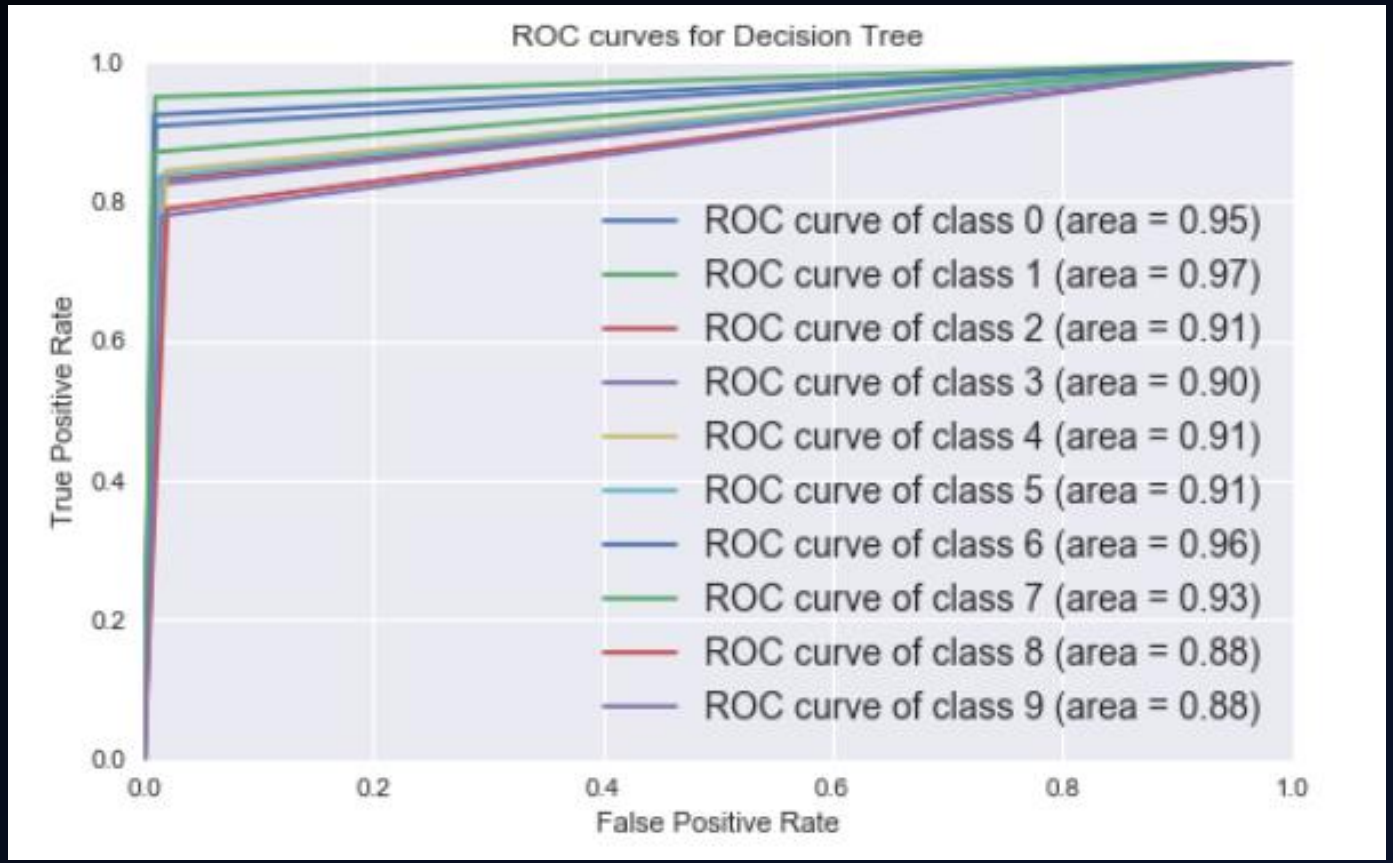


Decision Trees – Results





Decision Trees – ROC Curve

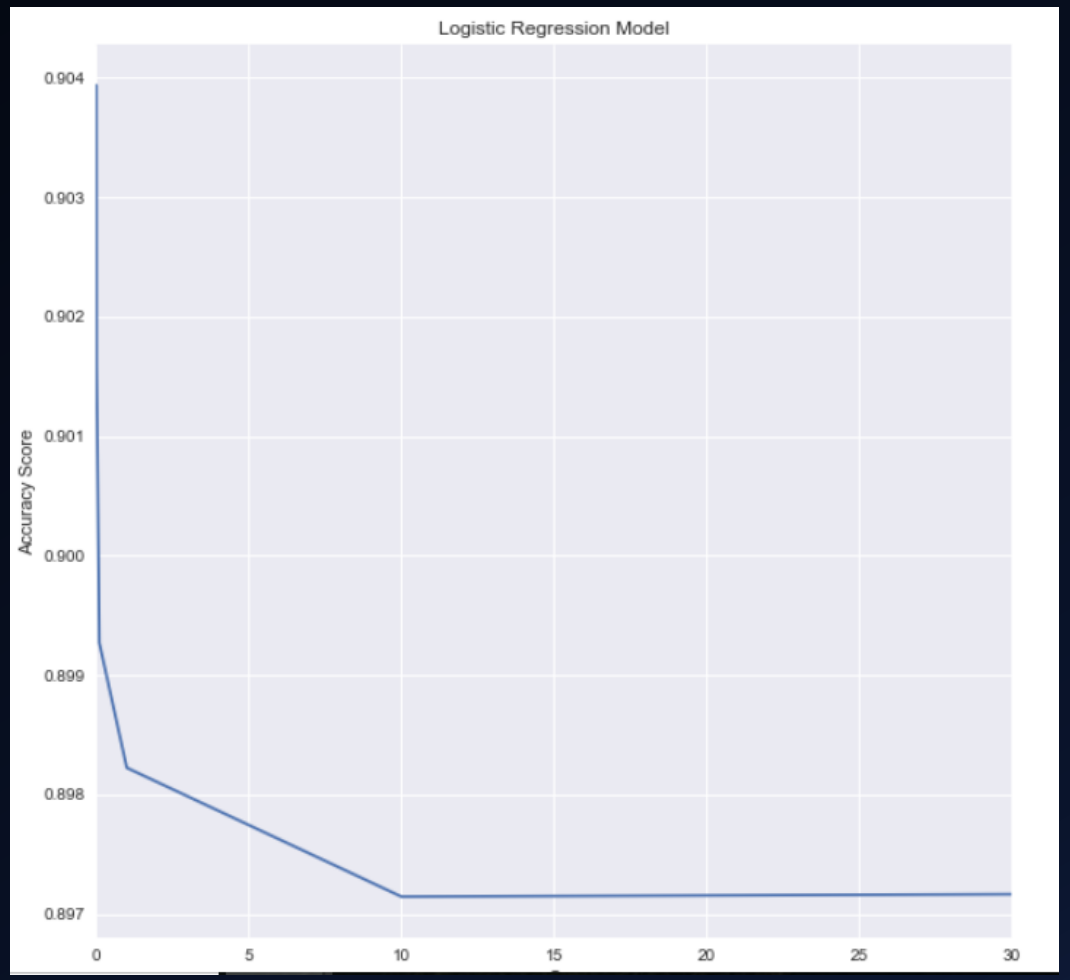


Decision Trees - Summary

Method	Accuracy
Default Parameters	84.54%
Max Depth = 10	85.61%
OneVsRestClassifier	78.74%

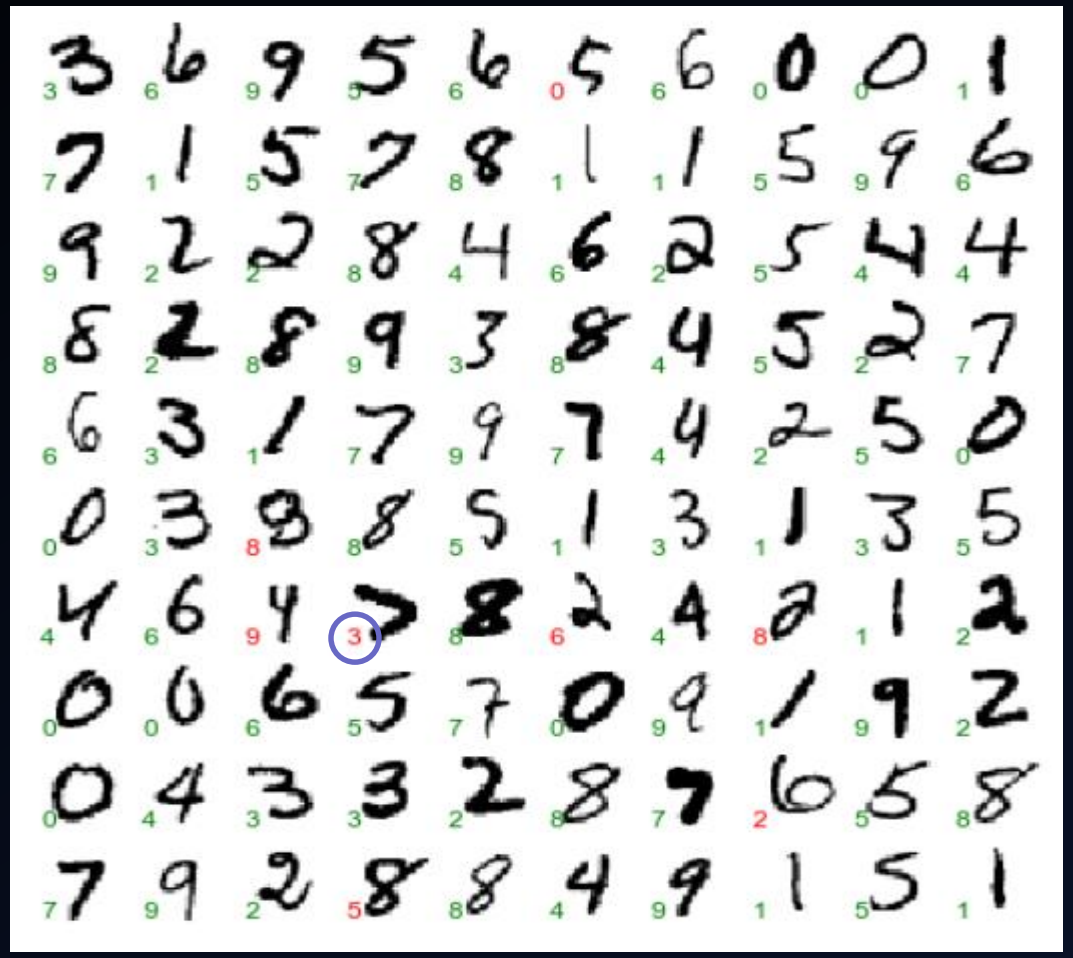
Logistic Regression – Cross Validation

- Tuned the ‘C’ with 5-fold cross-validation
- Checked for ‘C ’ in (0.001,0.01,0.1,1,10,100)
- Achieved optimal accuracy of 90.86% at C = 0.001

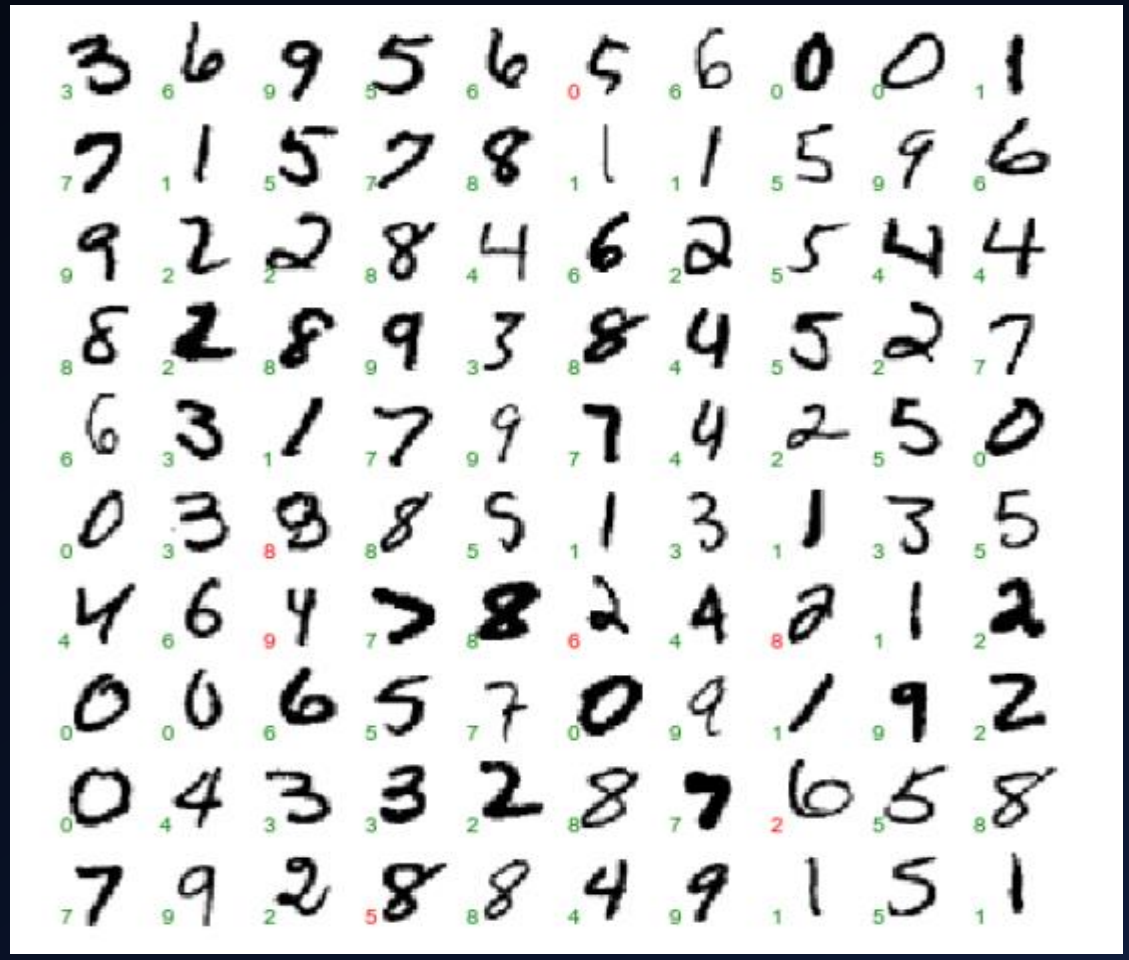


Logistic Regression – Results

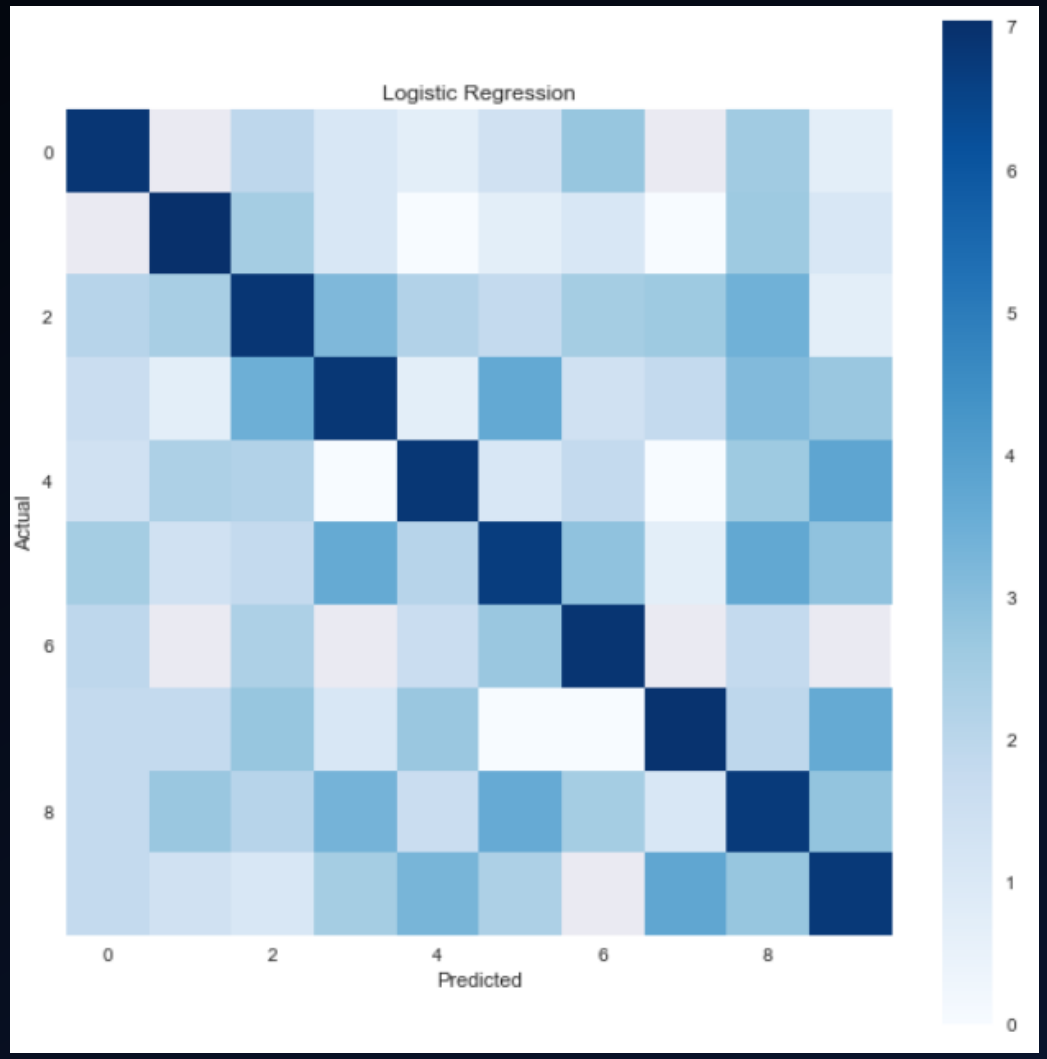
C=1



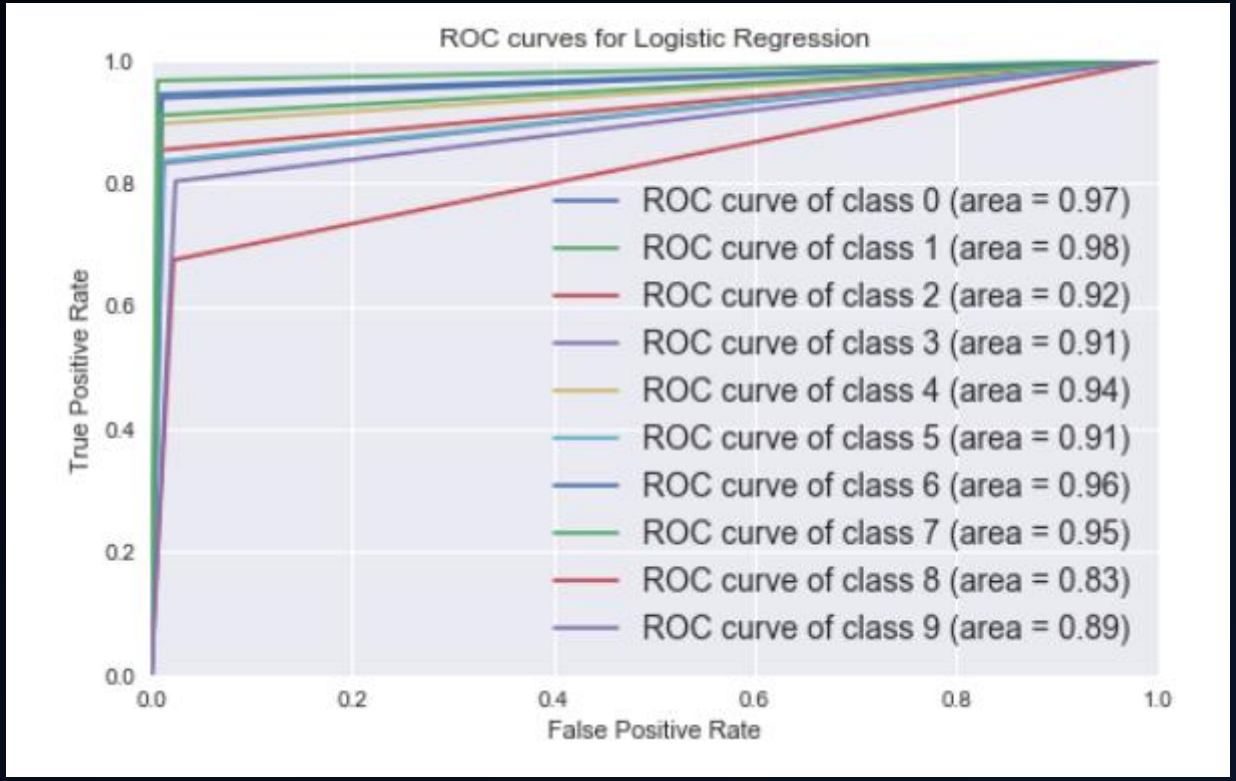
C=0.001



Logistic Regression – Results



Logistic Regression – ROC Curve



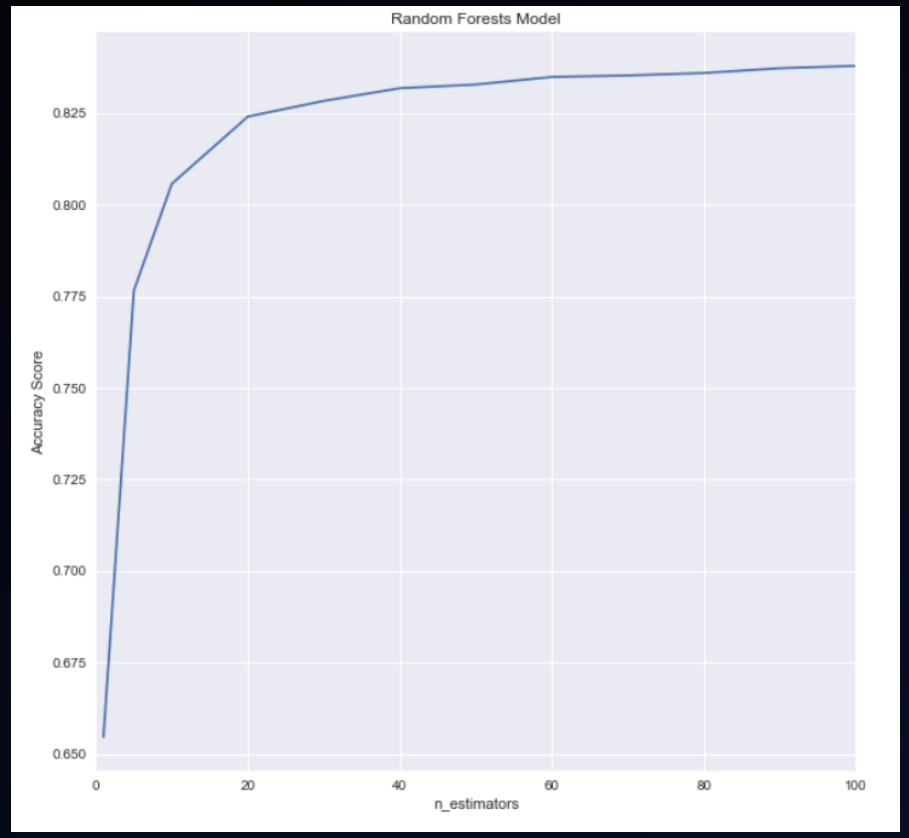
Logistic Regression - Summary

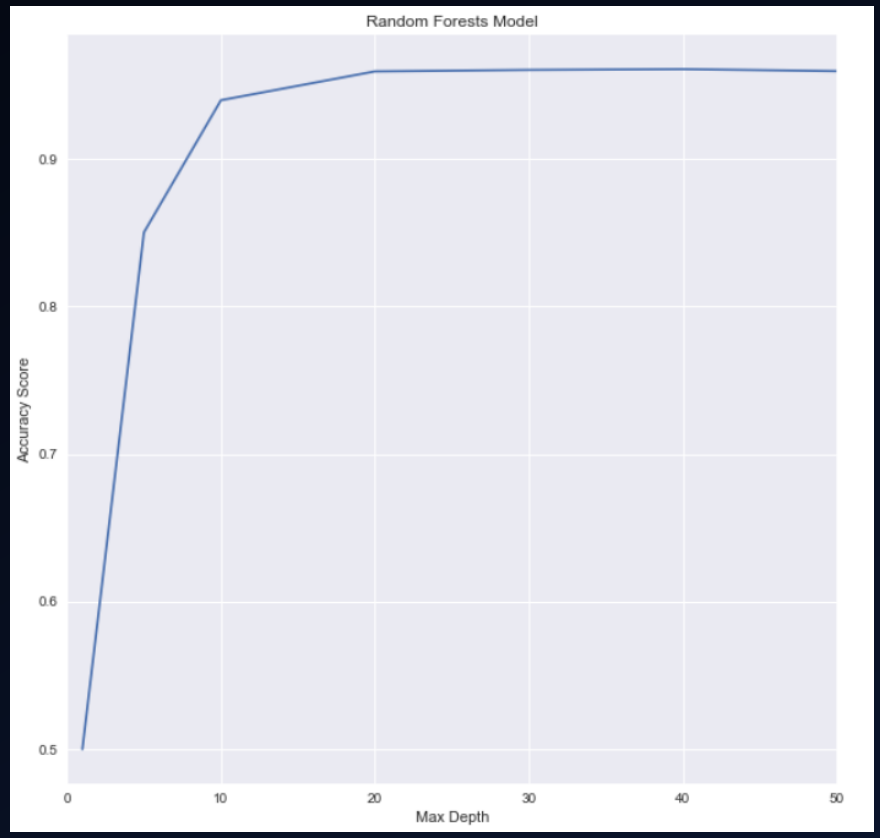
Method	Accuracy
Default Parameters	90.14%
C = 0.001	90.86%
OneVsRestClassifier	80.48%

Random Forests – Cross Validation

- Tuned the ‘n’ with 5-fold cross-validation
- Checked for ‘n_estimators ’ in(1,5,10,20,30,40,50,60,70,80,90,100)
- Checked for ‘max_depth’ in (1,5,10,20,30,40,50,60,70,80,90,100)
- Optimal Combination:
  n_estimators: 90
  max_depth: 30
  Accuracy: 96.32%

Random Forests – Cross Validation

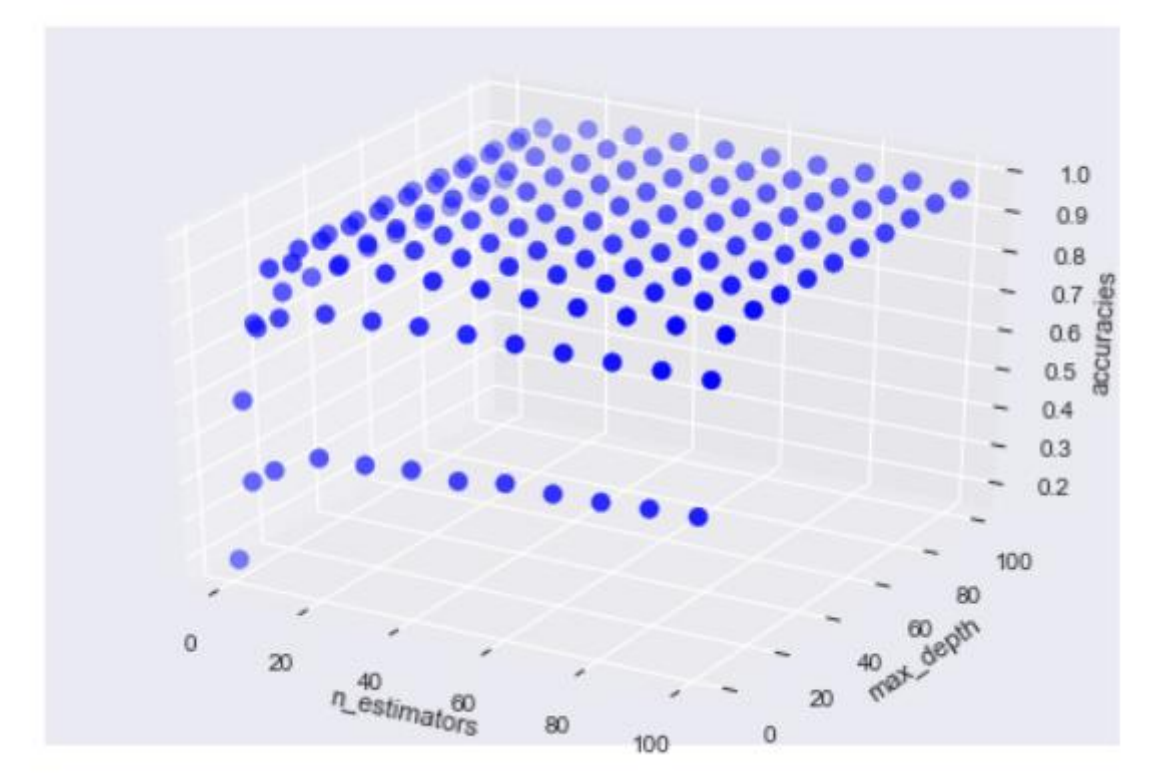




The optimal value for n_esitmators is 40 and Max Depth is 20

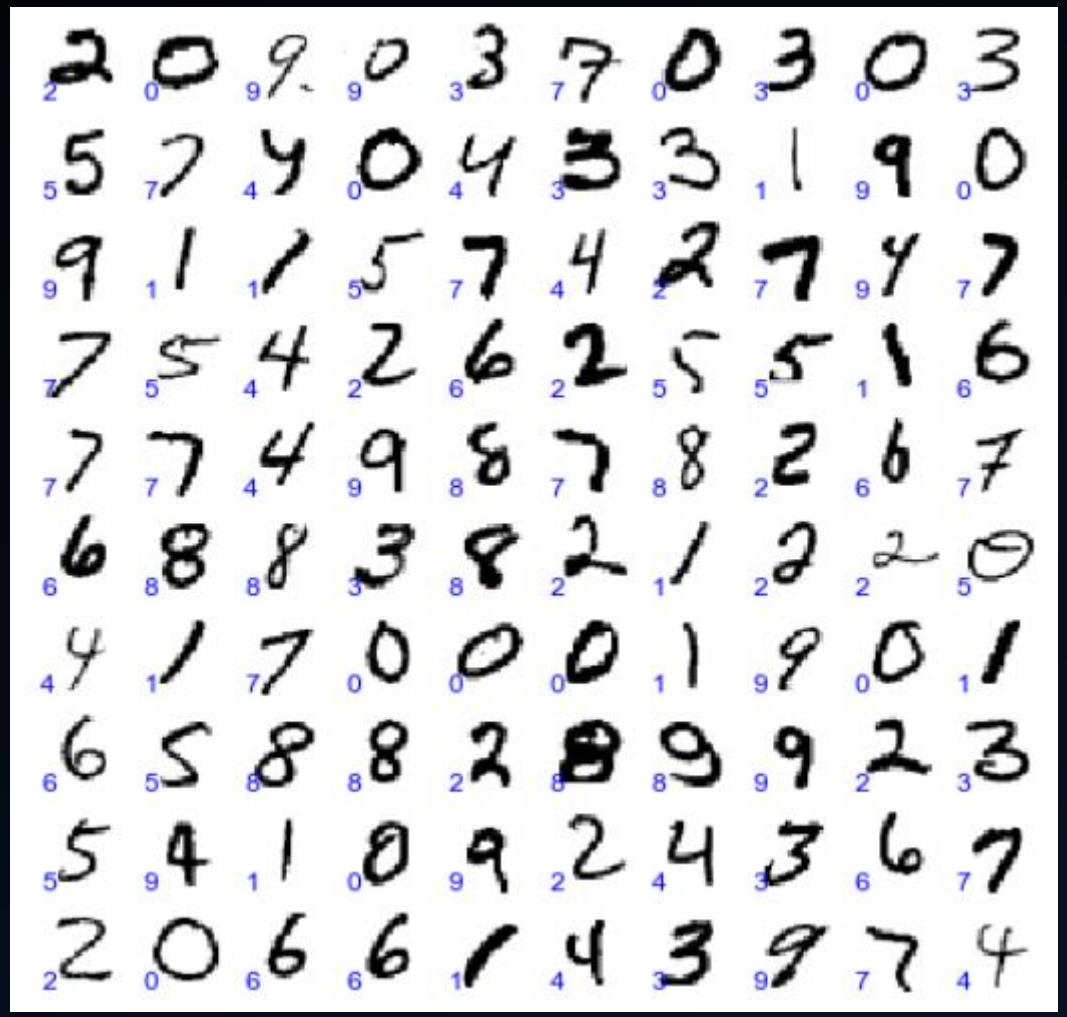
Accuracy : 96.0%

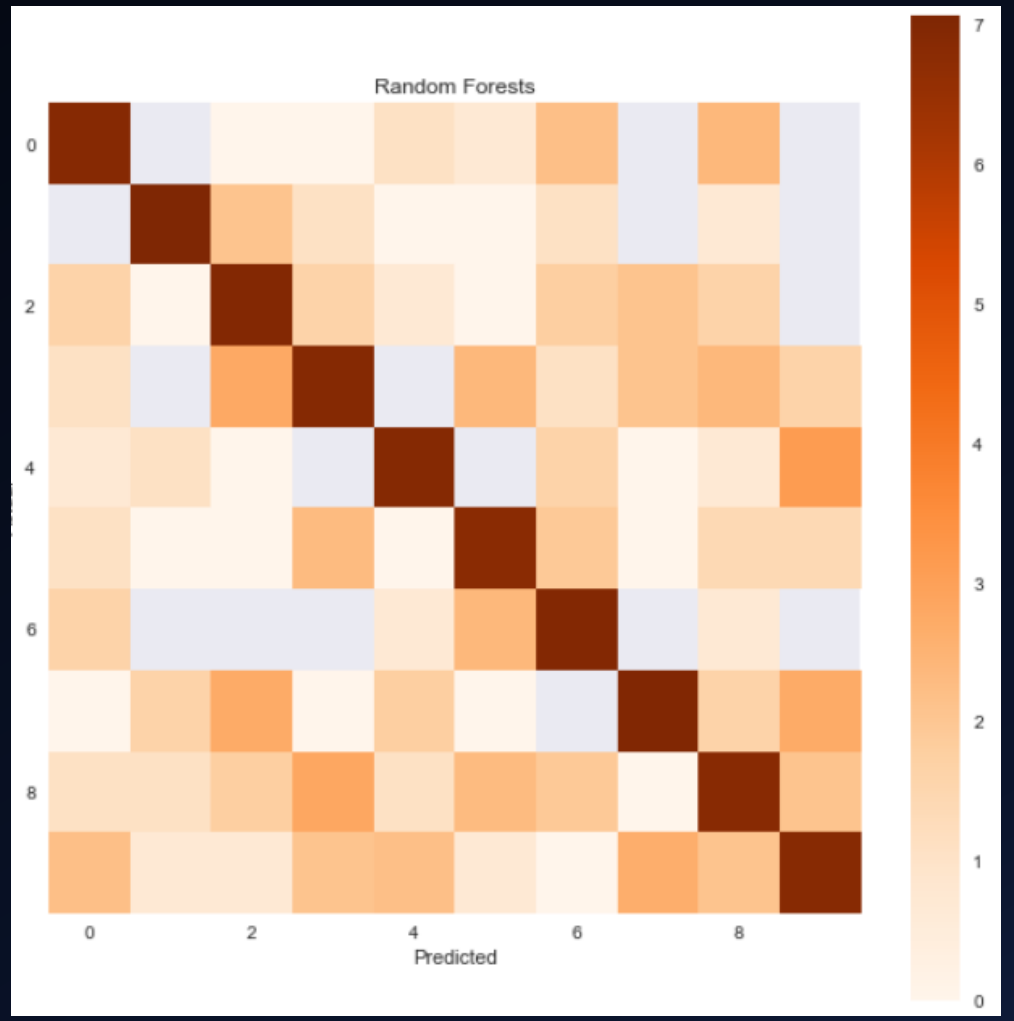
Random Forests – Cross Validation



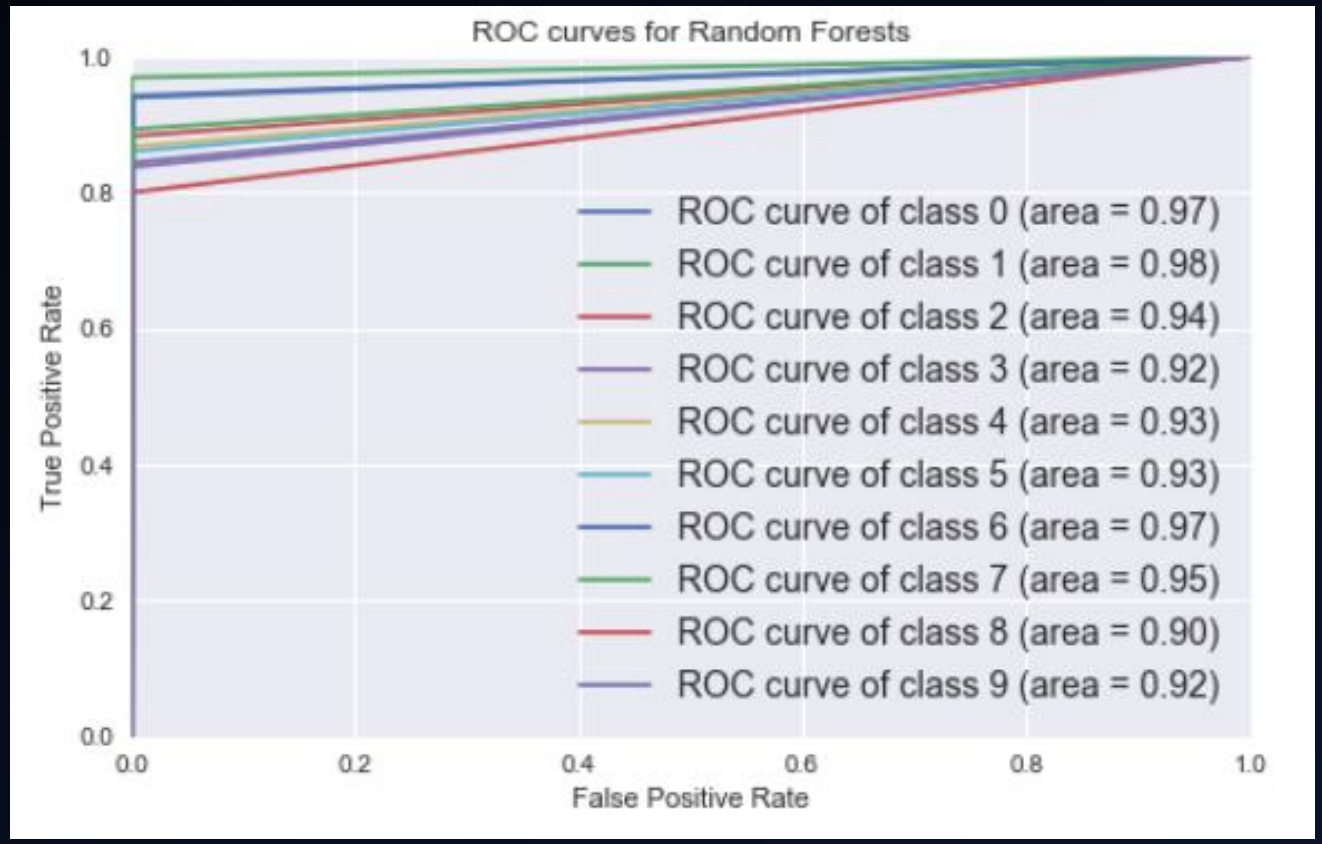
The optimal value for n_esitmators is 90 and Max Depth is 30
Accuracy : 96.32%

Random Forests – Results





Random Forests – ROC Curve

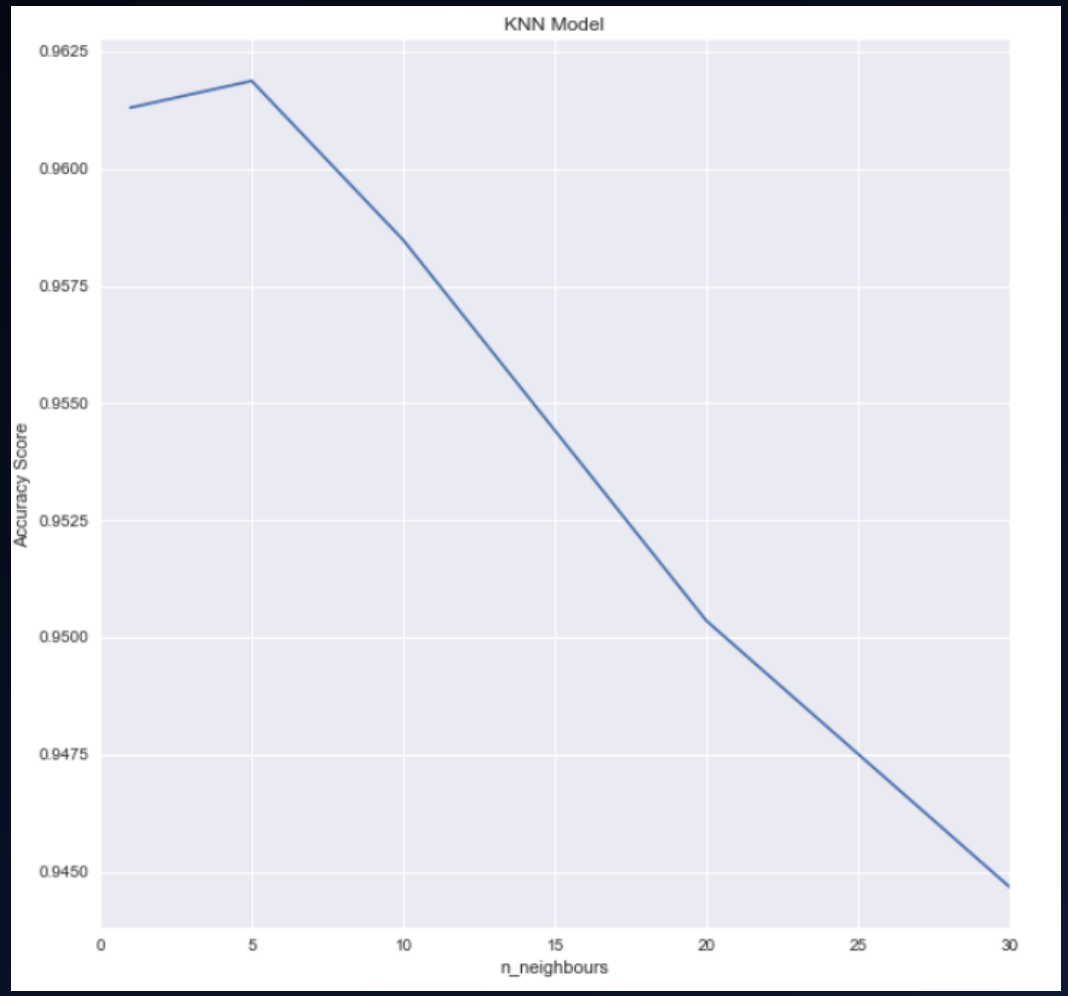


Random Forests - Summary

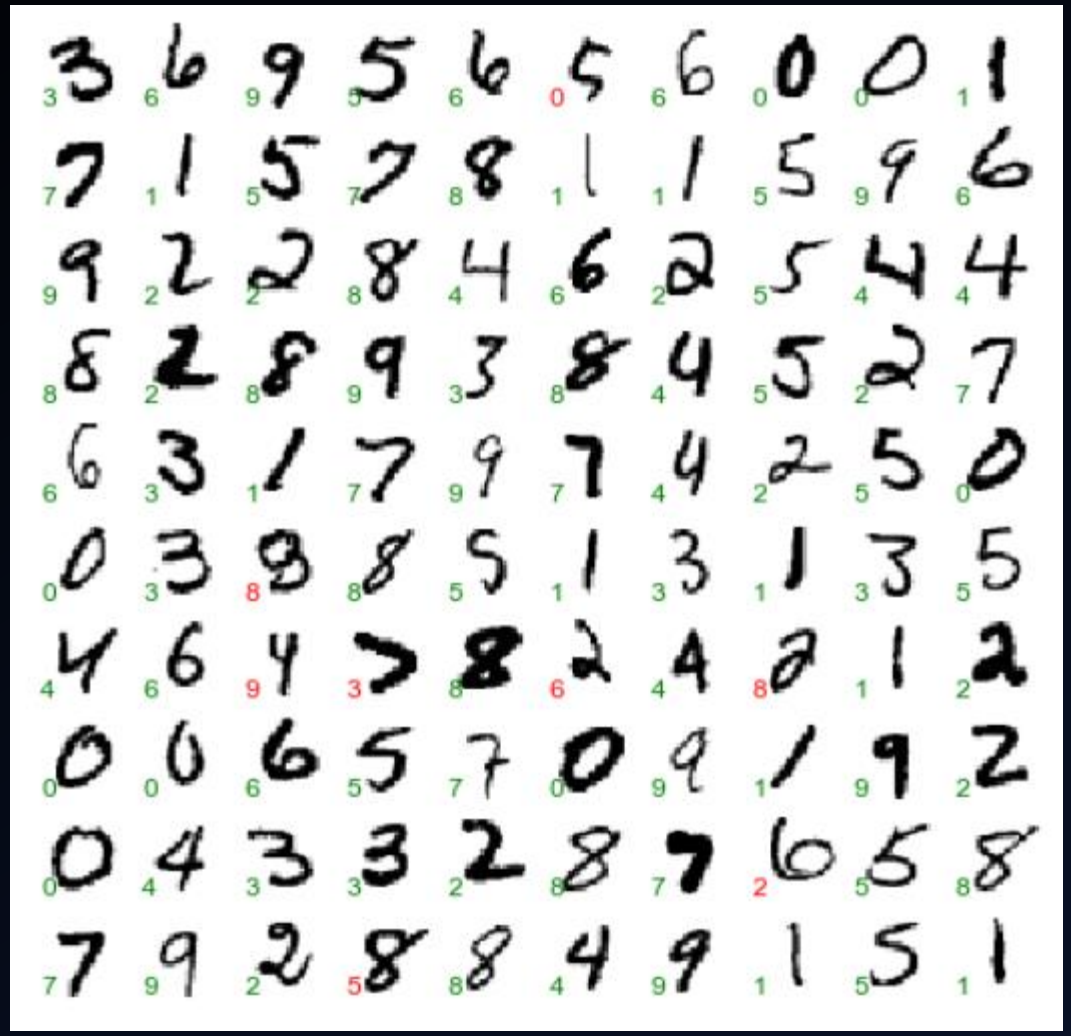
Method	Accuracy
Default Parameters	93.61%
Max_depth = 20 n_esitmators = 40	96.00%
Max_depth = 30 n_esitmators = 90	96.32%
OneVsRestClassifier	88.40%

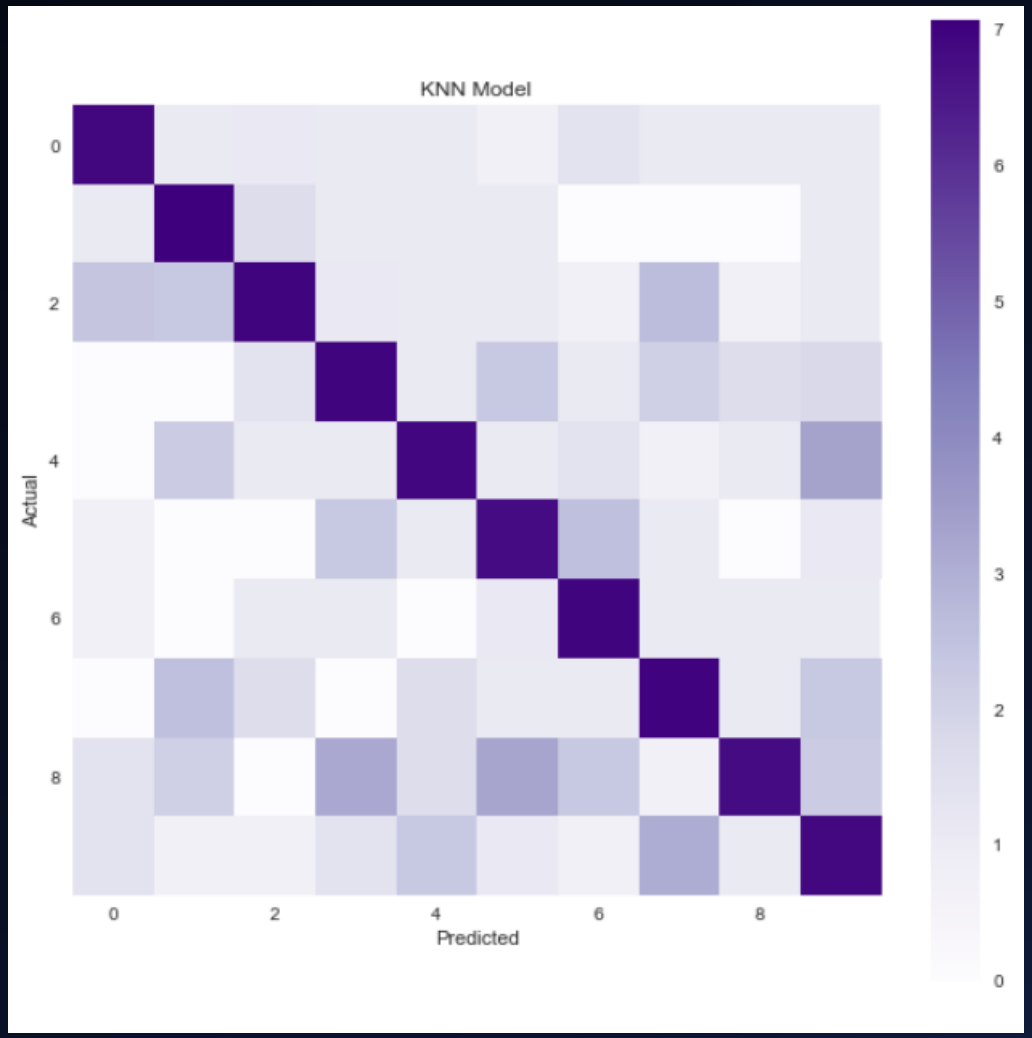
KNN – Cross Validation

- Tuned the ‘n_ neighbours’ with 5-fold cross-validation
- Checked for ‘n_ neighbours’ in (1, 5, 10, 20, 30)
- Achieved optimal accuracy of 96.67% at n_neighbours of 5

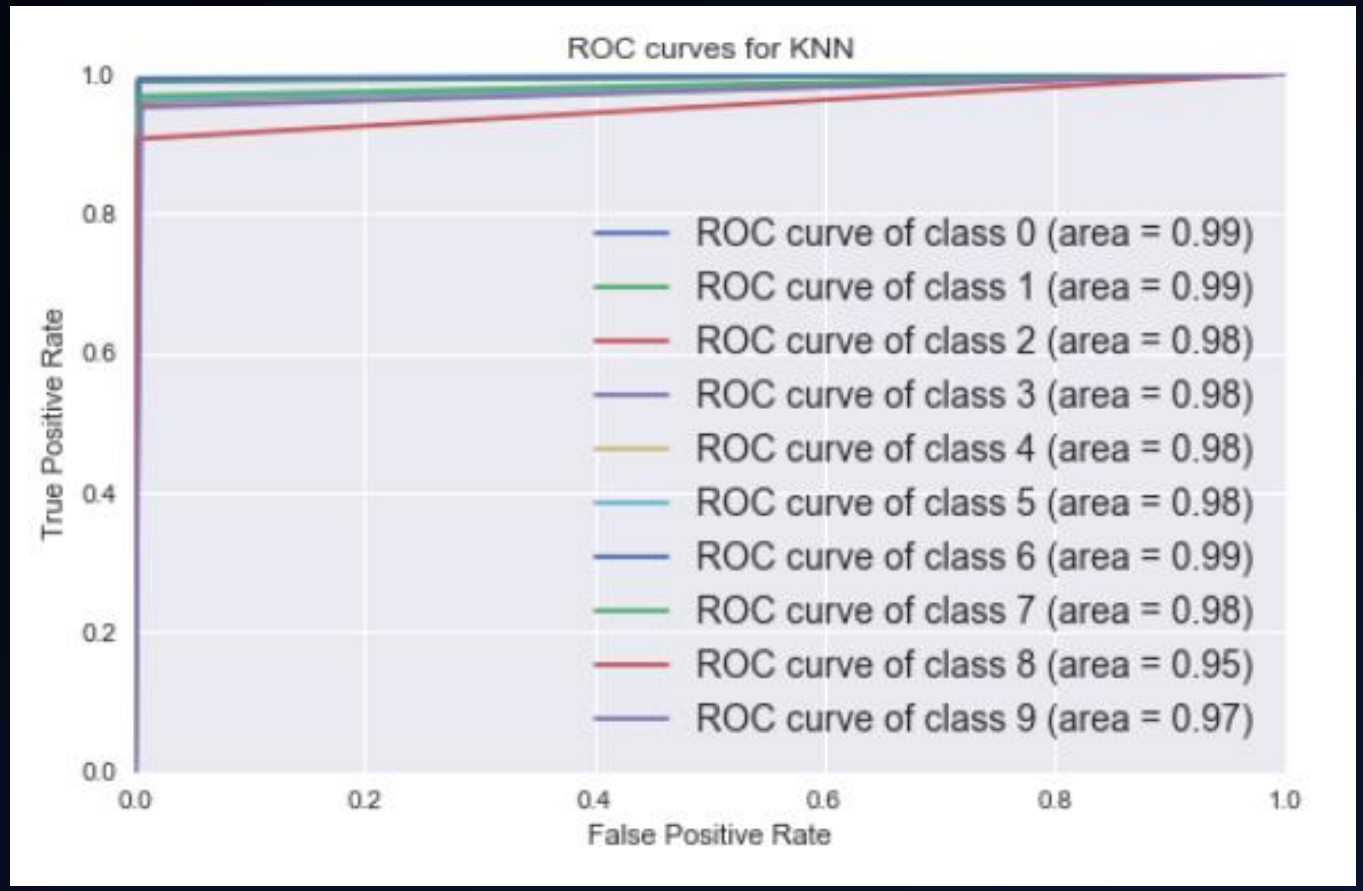


KNN – Results





KNN – ROC Curve



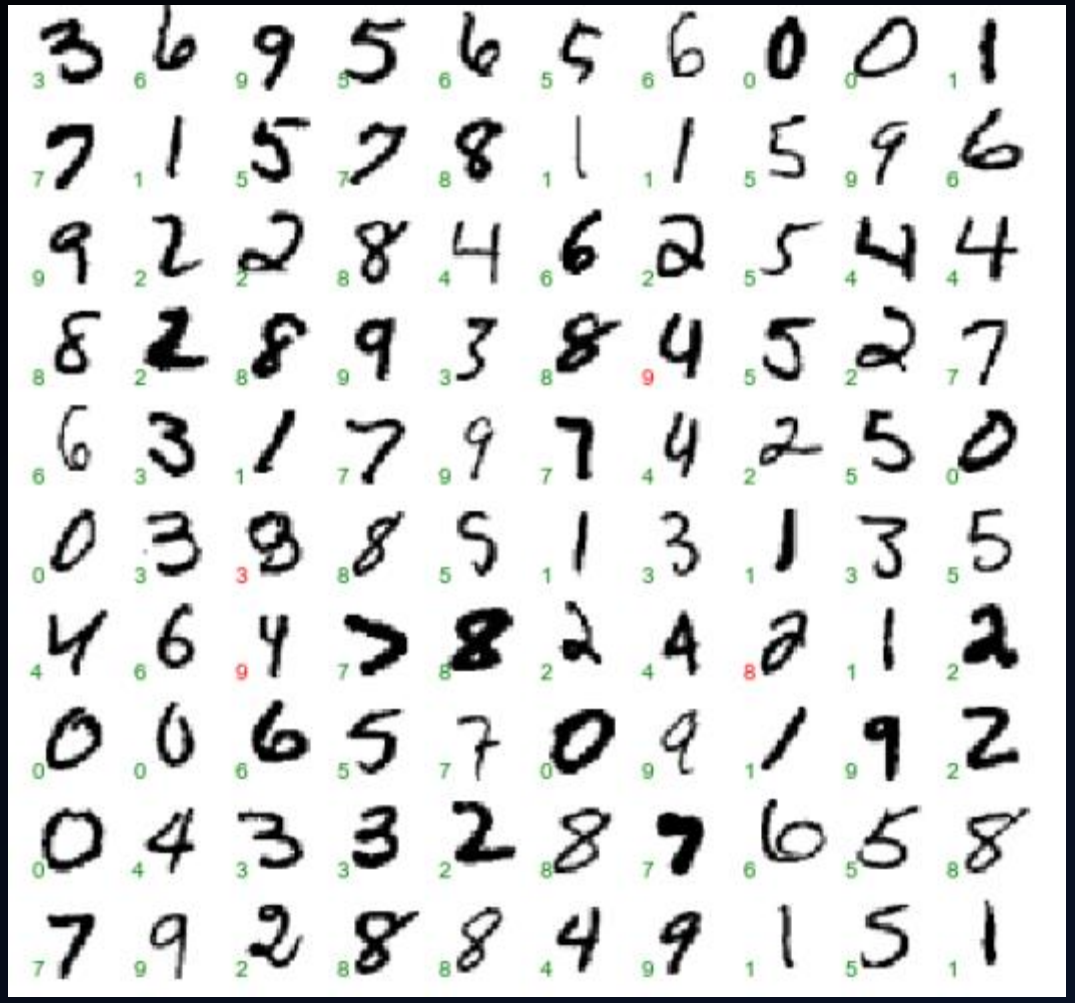
KNN- Summary

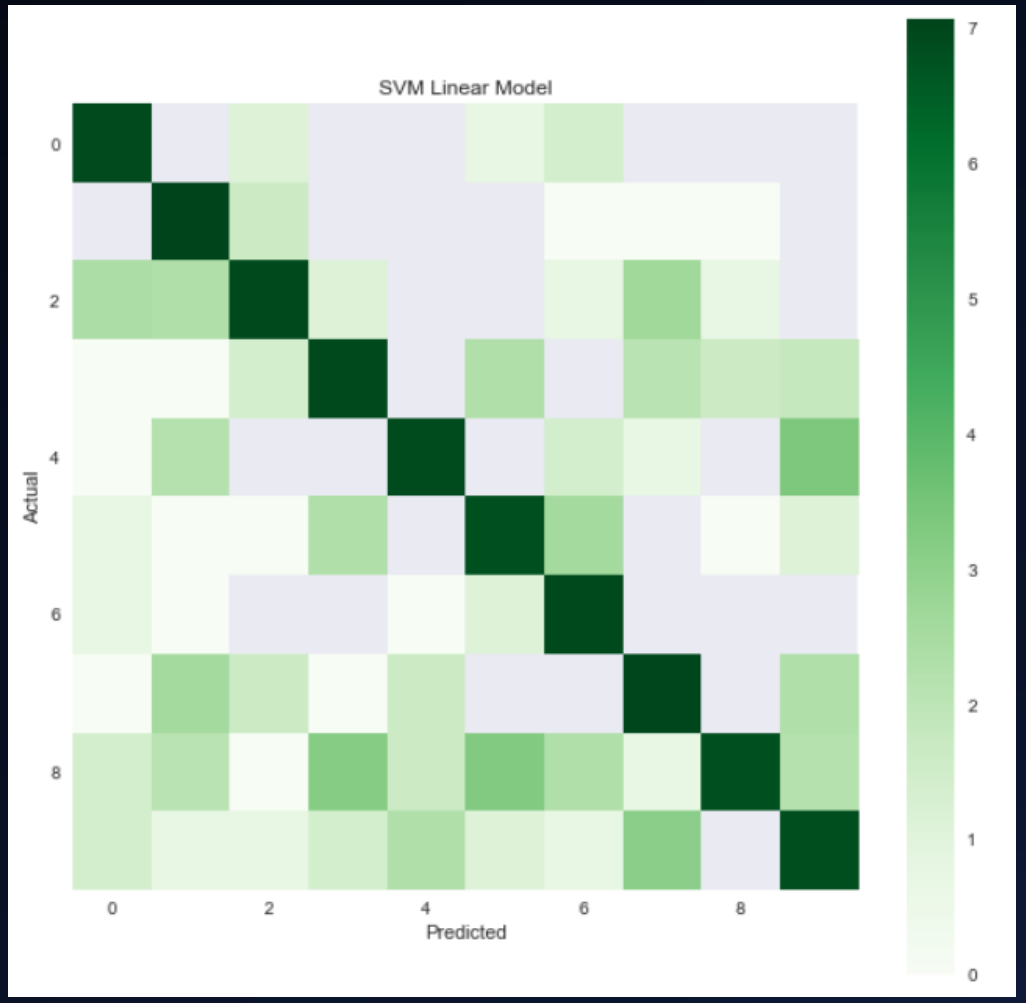
Method	Accuracy
Default Parameters	96.67%
n = 5 (Optimal same as default)	96.67%
OneVsRestClassifier	96.39%

SVM – Cross Validation

- Tuned the ‘C’ and gamma with 5-fold cross-validation
- Checked for ‘C ’ in (1, 5, 10)
- Checked for ‘gamma’ in (0.01,0.1,0.01)
- Optimal Values were Default values in case of linear kernel

SVM – Results

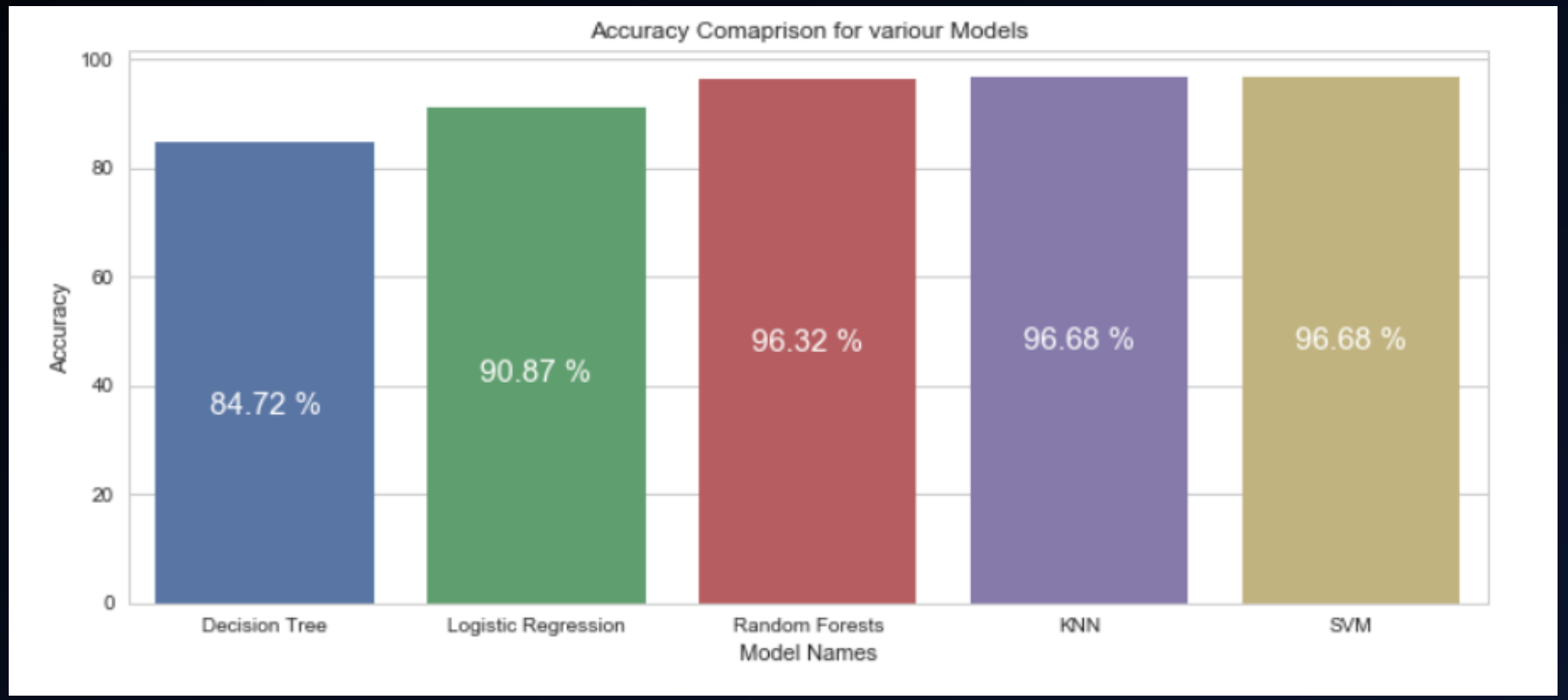




SVM - Summary

Method	Accuracy
Default Parameters with Linear Kernel	96.67%

Comparison Summary

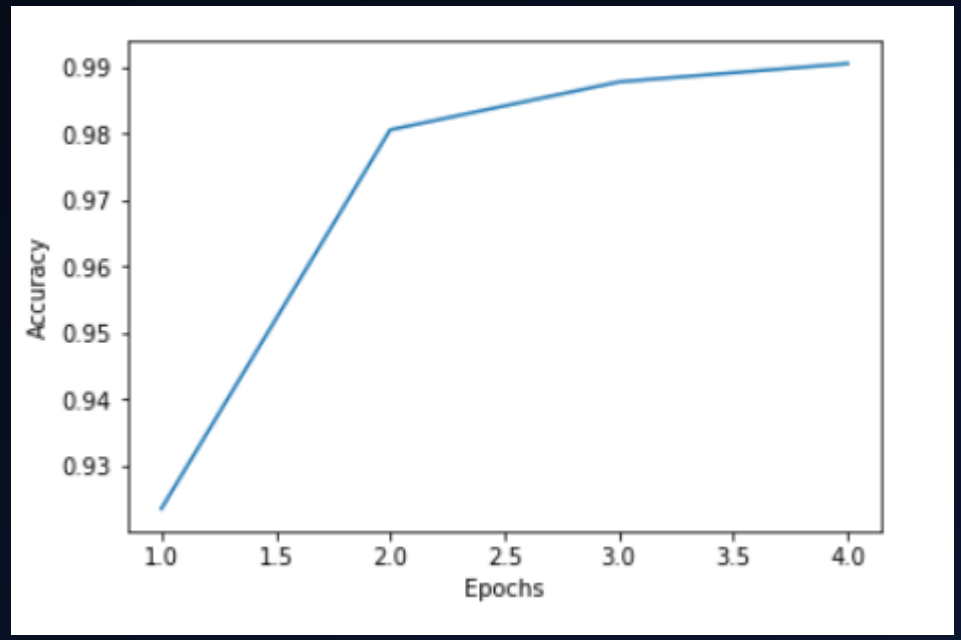


Neural Nets

- Standard Neural Nets (SNN)
- Convolutional Neural Nets (CNN)

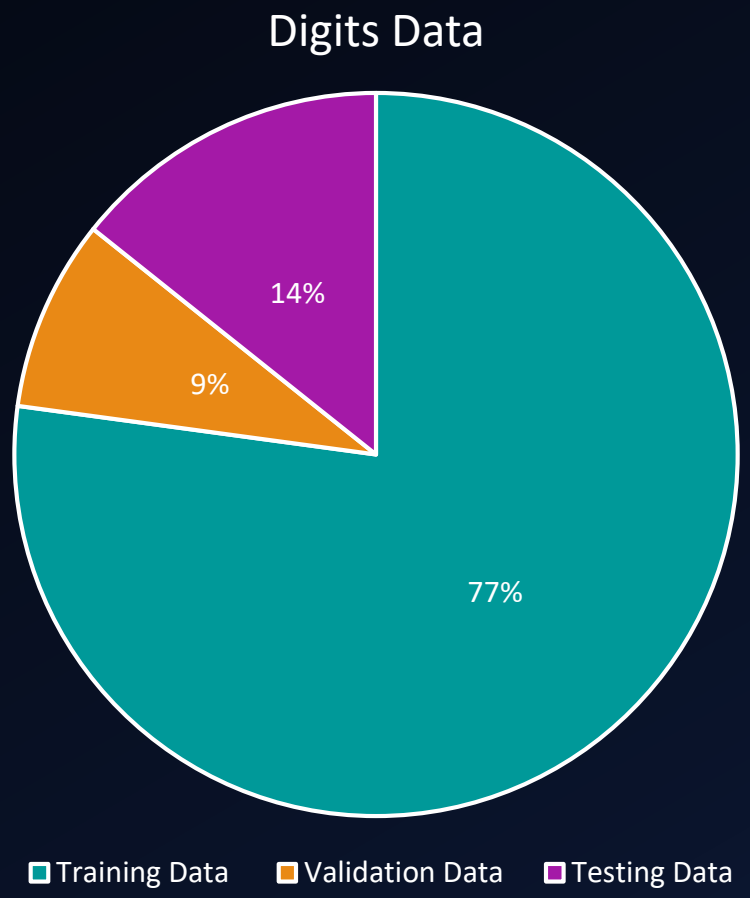
Standard Neural Net Results

- 4 Epochs, Batch Size 128, Validation Set – 20%
- Model loss: 5%
- Model accuracy: 98%
- Epoch Accuracy:
  - Epoch 1: 0.9235863095238095
  - Epoch 2: 0.9806175595238096
  - Epoch 3: 0.9878348214285714
  - Epoch 4: 0.9905877976190476



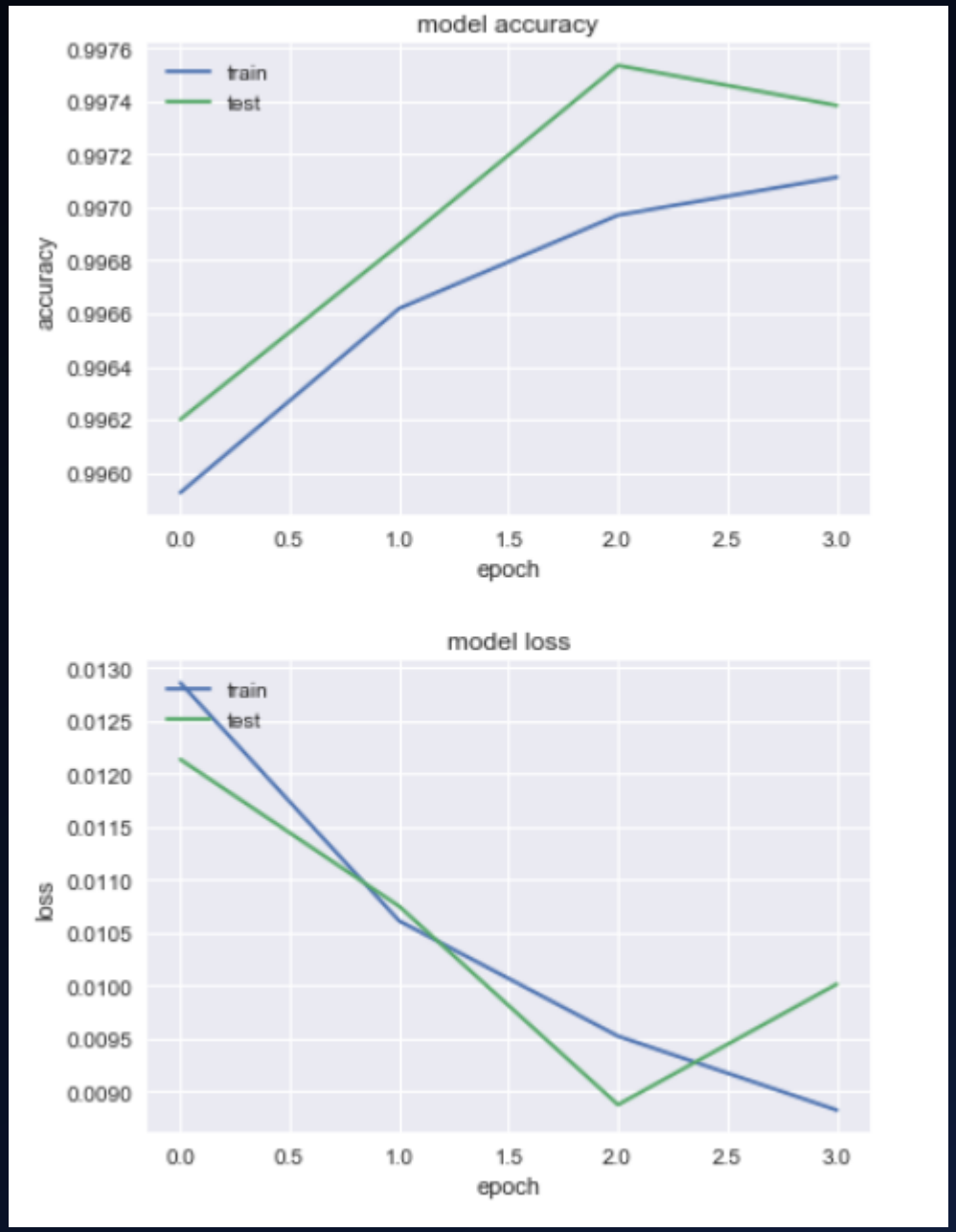
Convolutional Neural Nets (CNN)

- Data Source:
  - http://yann.lecun.com/exdb/mnist/
- Converted to png images
  - https://github.com/myleott/mnist_png
- Training Data: 54004
- Validation Data: 5996
  - 10% randomly selected images for validation
- Testing Data: 10000

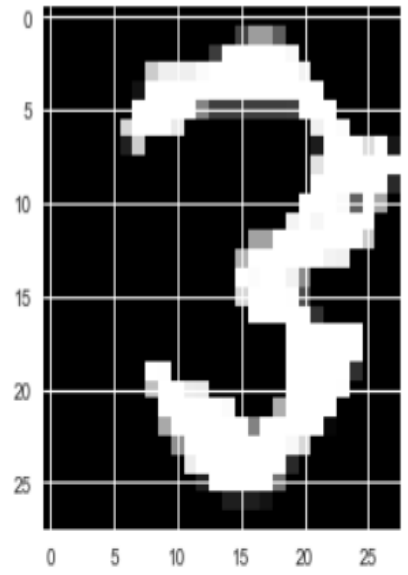


CNN Results

- 4 Epochs
- Model loss: 0.9%
- Model accuracy: 99.7%
- Epoch Accuracy:
  - Epoch 1: 0.9961997726198805
  - Epoch 2: 0.9968593064521052
  - Epoch 3: 0.9975354490786731
  - Epoch 4: 0.9973843235976326

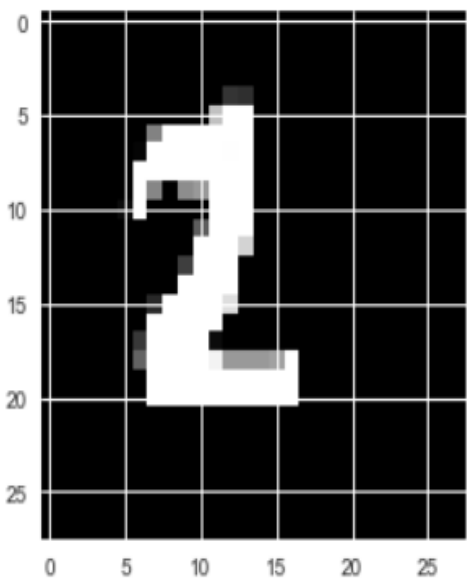


Demo



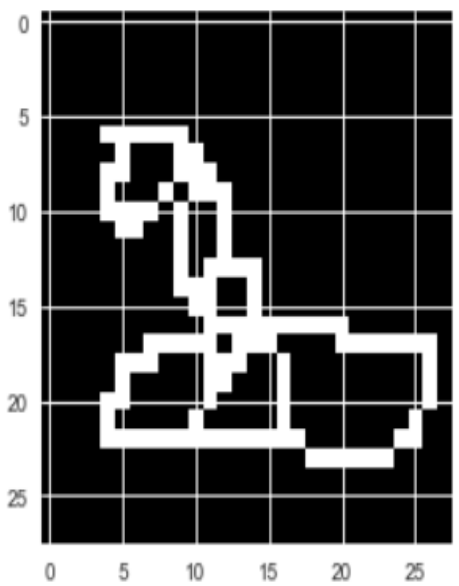
```
arr = np.array(testImg).reshape((shape_ord))
arr = np.expand_dims(arr, axis=0)
prediction = model.predict(arr)[0]
finalclass = ''
finalconf = -1
for n in [0,1,2,3,4,5,6,7,8,9]:
    if (prediction[n] > finalconf):
        finalclass = str(n)
        finalconf = prediction[n]
print ('This digit is a ' + finalclass + ' w
```

```
This digit is a 3 with 100.0% confidence.
```



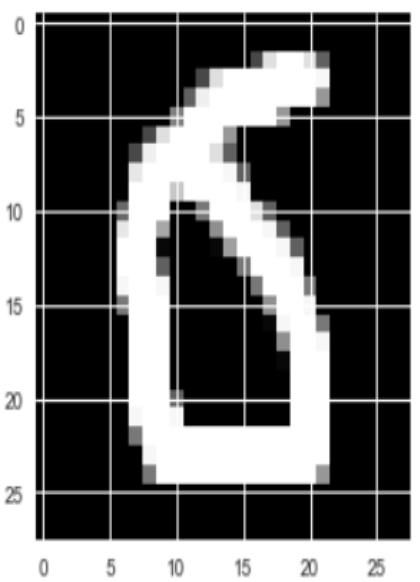
```
arr = np.array(testImg).reshape((shape_orc
arr = np.expand_dims(arr, axis=0)
prediction = model.predict(arr)[0]
finalclass = ''
finalconf = -1
for n in [0,1,2,3,4,5,6,7,8,9]:
    if (prediction[n] > finalconf):
        finalclass = str(n)
        finalconf = prediction[n]
print ('This digit is a ' + finalclass +
```

```
This digit is a 2 with 99.99725818634033%
```



```
arr = np.array(testImg).reshape((shape_o
arr = np.expand_dims(arr, axis=0)
prediction = model.predict(arr)[0]
finalclass = ''
finalconf = -1
for n in [0,1,2,3,4,5,6,7,8,9]:
    if (prediction[n] > finalconf):
        finalclass = str(n)
        finalconf = prediction[n]
print ('This digit is a ' + finalclass +
```

```
This digit is a 2 with 100.0% confidence
```



```
arr = np.array(testImg).reshape((shape_o
arr = np.expand_dims(arr, axis=0)
prediction = model.predict(arr)[0]
finalclass = ''
finalconf = -1
for n in [0,1,2,3,4,5,6,7,8,9]:
    if (prediction[n] > finalconf):
        finalclass = str(n)
        finalconf = prediction[n]
print ('This digit is a ' + finalclass +
```

```
This digit is a 0 with 100.0% confidence
```

THANK YOU