Decisions

1. Euclidian distance is chosen to calculate the Distance.
2. A main program is provided 'abhatkarHW1.m' that runs the two algorithms file name is specific 'letter-recognition.dat'
3. Confusion Matrix is stored in file 'confusionMat.csv'
4. Performance Matrix is stored in file 'performanceMatrix.xlsx'

Algorithms

1. [testY]=testKnn(trainX, trainY, testX, k)

   Input: trainX:: training data co-ordinates matrix (n X 16)
   trainY:: training labels matrix (n X 1)
   testX:: test data co-ordinates matrix (5000 X 16)
   k:: number nearest neighbors to check integer

   Output: testY:: predicted labels matrix (5000 X 16)

   Pseudocode:

   ```
   Calculate Euclidian distance of every point in testX from every point in trainX
   Sort the distances in ascending order
   If(k==1)
       Fetch index of nearest point in trainX for each point in testX
       testY=Fetch the label for each index from trainY
   else
       Fetch indices of k nearest points in trainX for each point in testX
       Fetch the labels for each index from trianY
       For each point in trainX calculate the mode of corresponding k labels
       testY=mode of labels from trianY for each point in testX
   ```

2. [condensedIdx] = condensedata(trainX, trainY)

   Input: trainX:: training data co-ordinates matrix (n X 16)
   trainY:: training labels matrix (n X 16)
   Output: condensedIdx:: condesnsed indices of training set (variable rows X 1)

   Pseudocode:

```
Initialize subset with a row in trainX
While((subset is not full ) && incorrctLabelIndices is not empty)
    Calculate distance of a point in trainX from each point in subset
    Find subset indices of points whose distance is minimum from trainX
    Find corresponding labels from trainY
    Compare labels with all labels in trainY
    Store incorrect labels in incorrectLableIndices matrix
    If(incorrectLabelIndices is not empty)
        Randomly select a label index
        Fetch data from trianX for that label
        Add that data to subset
        Add the index of the label to the condenseIDx
```

My Observations

Using Performance Matrix

1. As the training data size increased the accuracy of prediction increased.
2. After condensing the training data the computation time reduced.
3. Time to condense training data increased with increase in training data.
4. After condensing accuracy decreased, but it was still better than smaller uncondensed training data.
5. Best performance in terms of accuracy was observed with K=1 N=15000 without condensing training data.
6. As the K increased the accuracy decreased.

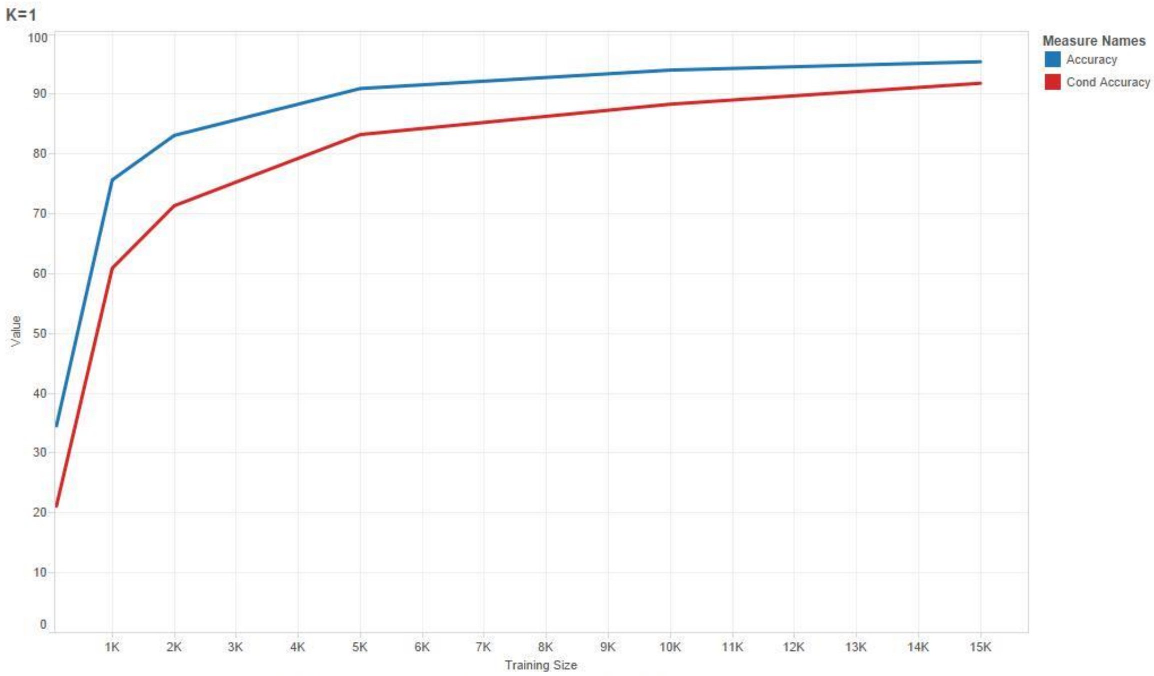
Using Confusion Matrix

Confusion matrix is calculated for K=1 N=15000 without condensing training data.

1. F and P were the most confused letters.

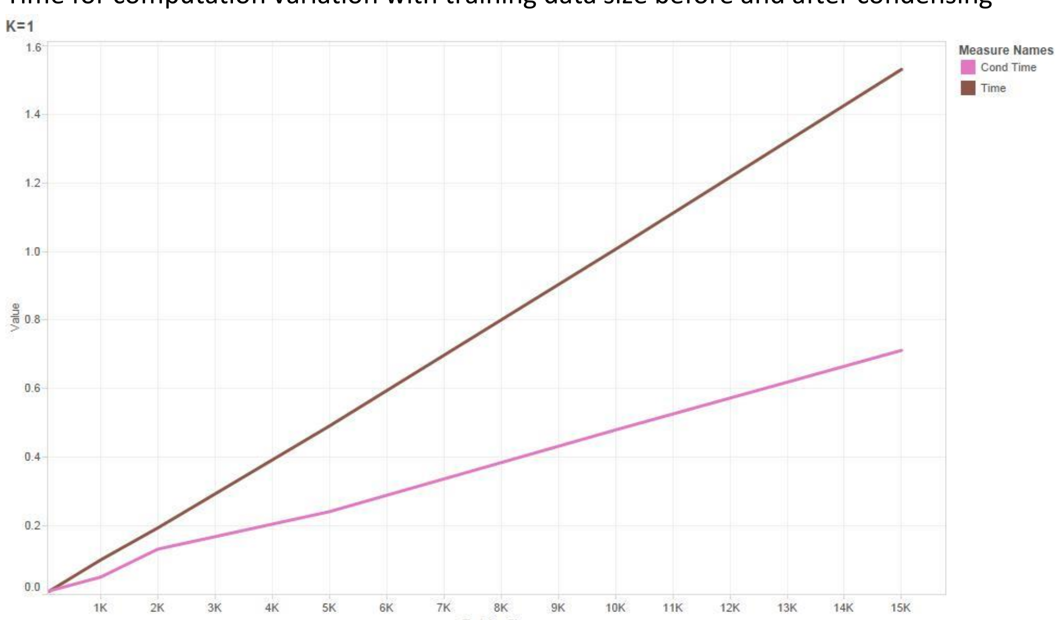
Graphical Representation of Observations

1. Accuracy variation with training data size before and after condensing-



The trends of Accuracy and Cond Accuracy for Training Size. Color shows details about Accuracy and Cond Accuracy.

2. Time for computation variation with training data size before and after condensing-



The trends of Cond Time and Time for Training Size. Color shows details about Cond Time and Time.

Conclusion

For best performance always use K=1 N=15000.

Depending on tolerance level, if one can compute condensing before testing labels efficient performance can be achieved.