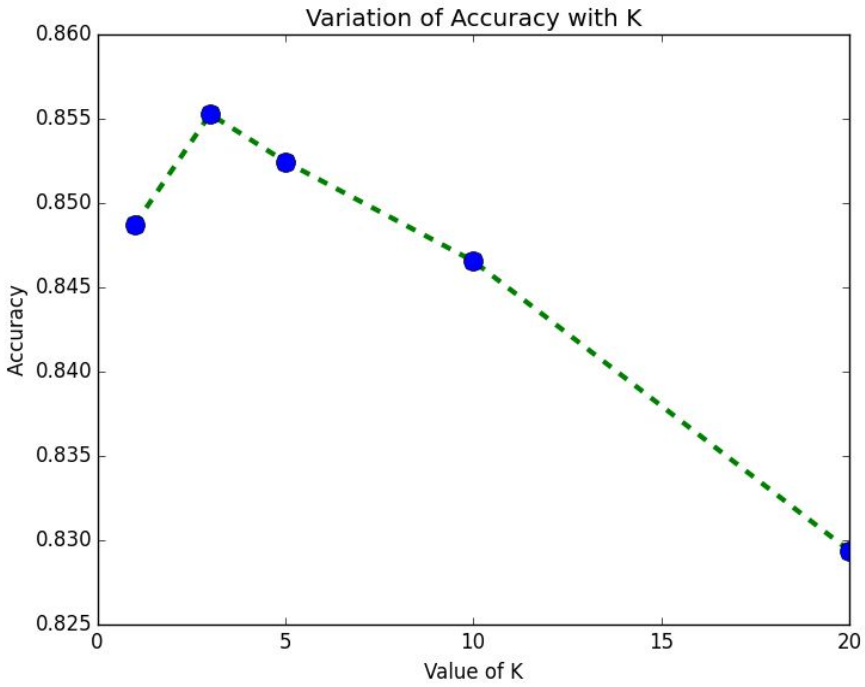
INF 2B
LEARNING
Task 1 – K-NN classification
REPORTS

S1674417

Task 1.3(a)
Time taken to compute Knn = 46.5806751251 seconds

Value of K	N	Nerrs	Accuracy
1	7800	1180	0.848717948718
3	7800	S1129	0.855256410256
5	7800	1151	0.852435897436
10	7800	1197	0.846538461538
20	7800	1331	0.829358974359



Task 1.4

Task 1 requires the following calculations to be made:

The (squared) Euclidean distance d_{ij} is given as

$$d\_\{ij\} = \| X\_i - Y\_j \|\hat{}2 = (X\_i - Y\_j) * (X\_i - Y\_j)\hat{}T$$

In k-NN, for each test data instance X_i, we need to calculate
d_{i1}, ..., d_{iM}, to find the k closest data points in Y, so that, for all test instances, we calculate distance matrix DI of N-by-M:

```
        / d_{11} ... d_{1M} \
DI = |    .        .     |
     |    .        .     |
     \ d_{N1} ... d_{NM} /
```

where X_i = (x_{i1}, ..., x_{iD}) denotes the i-th test data, and
Y_j = (y_{j1}, ..., y_{jD}) denotes the j-th training data.

At first, note that d_{ij} can be decomposed in the following manner.

$$d\_\{ij\} = (X\_i - Y\_j) * (X\_i - Y\_j)\hat{}T$$
$$= X\_i * X\_i\hat{}T - 2 X\_i * Y\_j\hat{}T + Y\_j * Y\_j\hat{}T$$

```
Let,      / X_1 * X_1^T \            / Y_1 * Y_1^T \
   XX = |     .       | ,     YY = |     .       | ,
        |     .       |            |     .       |
        \ X_N * X_N^T /            \ Y_M * Y_M^T /
```

so that XX and YY are N-by-1 and M-by-1 matrices, respectively.
Then DI can be obtained with standard matrix operations:

```
DI = (XX,...,XX) - 2 * X * Y^T + (YY,...,YY)^T
     <-- M -->              <-- N -->
```

Following suit as illustrated above, XX can be easily calculated by taking an element wise square of each element in the XX matrix that represents the Xtst data.
YY is calculated in a similar way and represents the Xtrn data.

The most expensive step while find the K nearest neighbours is finding the distance between test and training data point.

DI is calculated using the above formula which, using Python's handy broadcasting feature simplifies extending matrices in order to multiply matrices of different dimensions.

Another optimization is done using argsort which returns an array of indices of the same shape as that of the index data along the given axis in sorted order.