Zachary Hyman

U13384191

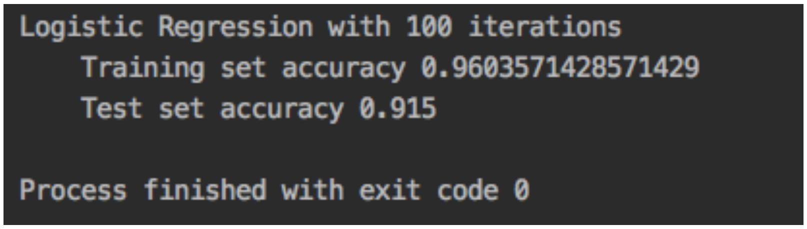
CS506

HW02: Classification and Dimensionality Reduction

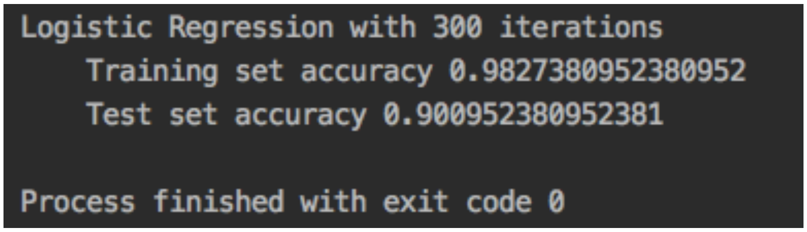
1. a.) See section 1.a. in hw02.py

b.) See section 1.b for logistic regression implementation.

Train and test accuracy depicted below with test size of 20% of sample data.

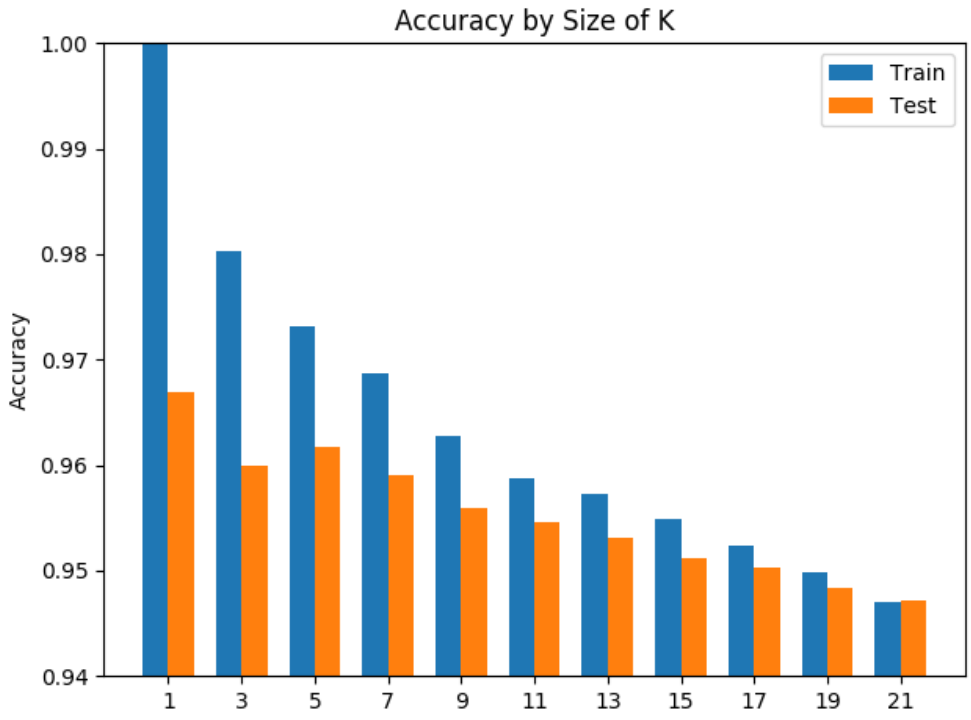


```
Logistic Regression with 100 iterations
    Training set accuracy 0.9603571428571429
    Test set accuracy 0.915

Process finished with exit code 0
```



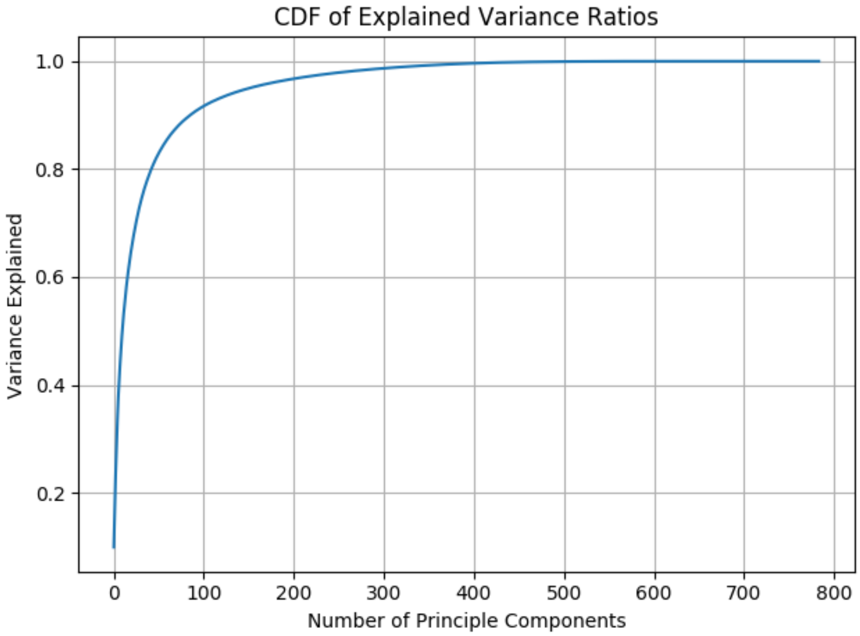
```
Logistic Regression with 300 iterations
    Training set accuracy 0.9827380952380952
    Test set accuracy 0.900952380952381

Process finished with exit code 0
```

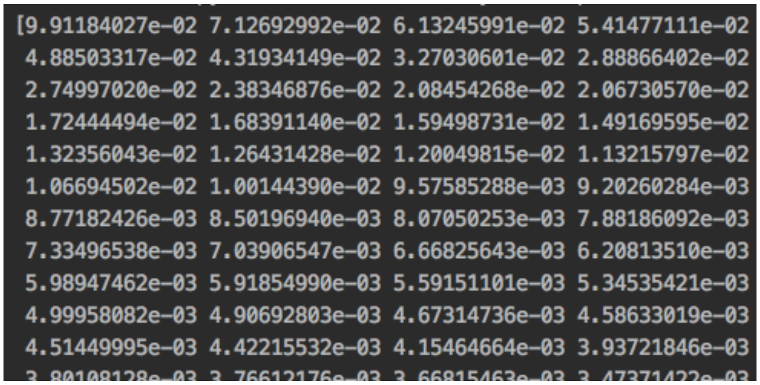
c.)



d.) From the above observations of fitting and predicting the test data, I found that the KNN clustering provided a better accuracy than the findings of the logistic regression. The logistic regression function produced a test accuracy of 0.915 with 100 iterations of adjusting the algorithm weights with a training accuracy of 0.96. When increasing the number of iterations to 300 I saw an increase to 0.98 accuracy with the training set, yet a decrease to only approximately 0.9 accuracy in the test set. With more iterations I believe that the model was conforming to noise or outliers found in the training set so it may better describe that, demonstrated in the increase in accuracy, yet produced worst test set predictions as a result. When working with the KNN classifying algorithm I found that the size of K which produced the greatest test accuracy to be a K size of one. Depicted in the graph above in 1.c show the training and test accuracies for all sizes of K stepping by 2 from 1 through 25. As shown in the graph, the larger K became, the large the classifying clusters become, and thus were collecting and classifying points incorrectly and generalizing as a result. The graph demonstrates a downward trend for any K greater than 25 as well. In this case we were not over-classifying with a K size of one as in the data it produces the greatest prediction accuracy. Alternatively, the second best K size determined would be a K size of 5 shown by the peak in the graph.

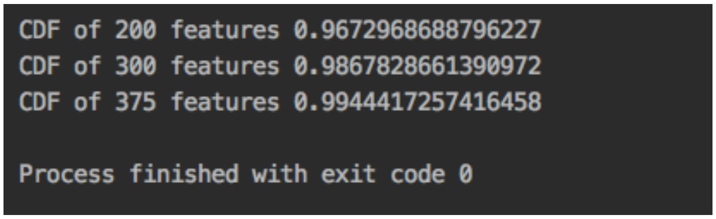
2. a. & b.)





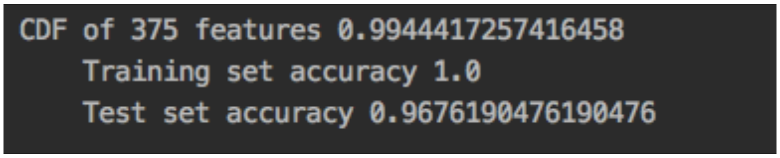
```
[9.91184027e-02 7.12692992e-02 6.13245991e-02 5.41477111e-02
 4.88503317e-02 4.31934149e-02 3.27030601e-02 2.88866402e-02
 2.74997020e-02 2.38346876e-02 2.08454268e-02 2.06730570e-02
 1.72444494e-02 1.68391140e-02 1.59498731e-02 1.49169595e-02
 1.32356043e-02 1.26431428e-02 1.20049815e-02 1.13215797e-02
 1.06694502e-02 1.00144390e-02 9.57585288e-03 9.20260284e-03
 8.77182426e-03 8.50196940e-03 8.07050253e-03 7.88186092e-03
 7.33496538e-03 7.03906547e-03 6.66825643e-03 6.20813510e-03
 5.98947462e-03 5.91854990e-03 5.59151101e-03 5.34535421e-03
 4.99958082e-03 4.90692803e-03 4.67314736e-03 4.58633019e-03
 4.51449995e-03 4.42215532e-03 4.15464664e-03 3.93721846e-03
 3.80108128e-03 3.76612176e-03 3.66815463e-03 3.47371422e-03
```

Snippet of explained variance table



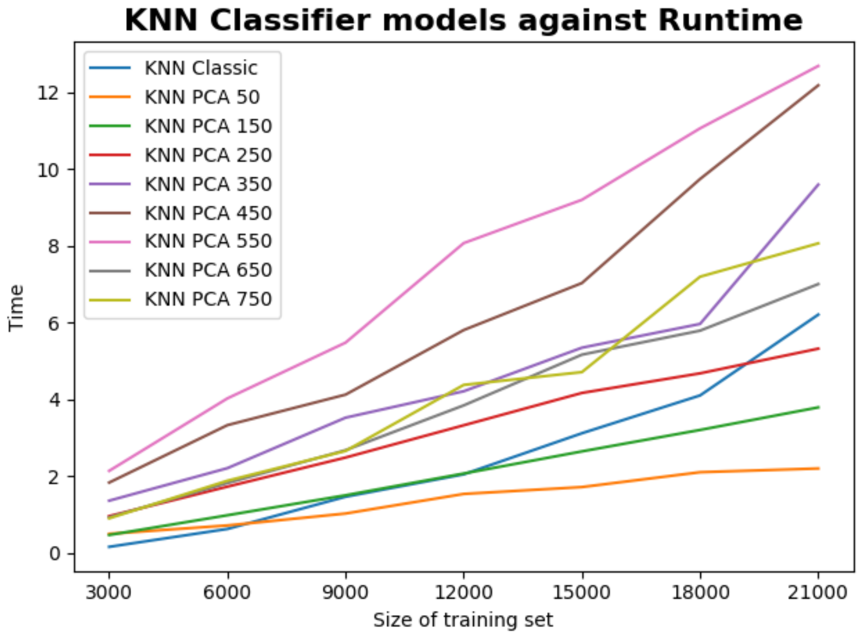
```
CDF of 200 features 0.9672968688796227
CDF of 300 features 0.9867828661390972
CDF of 375 features 0.9944417257416458

Process finished with exit code 0
```

c.) For this data set I determined that an optimal number of principle components would be a size of 375 components. As shown above 375 components were able to capture 0.99 of the variance displayed by the data and yet is still roughly half the size of components as the original data set ~750. In order to maintain a high level of predictive accuracy I chose a principle component size that captures nearly all variance, yet was still a number of components far less than the total amount. Below are the results of using KNN on 375 components with a K size of one.



```
CDF of 375 features 0.9944417257416458
    Training set accuracy 1.0
    Test set accuracy 0.9676190476190476
```

d.) In the plot below I found that the amount of principle components affects the run time of an algorithm much more than the size of a data set. This is demonstrated by the vast difference between principle component analysis of 50, as opposed to many of the larger numbers taking much longer. The PCA of 50 also did not change significantly with any increase in the data size and remained rather consistent demonstrating the impact of reduce dimensionality of the data. What I also found interesting in the data was that PCA analysis of 550 values led to some of the longest times, though this may be due to optimizations in the PCA algorithm for dealing with nearly or all fields of a data set running much faster.


KNN Classifier models against Runtime
KNN Classic
KNN PCA 50
KNN PCA 150
KNN PCA 250
KNN PCA 350
KNN PCA 450
KNN PCA 550
KNN PCA 650
KNN PCA 750
Time
Size of training set