Implementation of kNN classifier on Income and Iris datasets

Abhinav

Introduction

The objective is to implement a kNN classifier by building upon the similarity measure algorithm built previously. The classifier is tested on Income and Iris test datasets and a performance analysis is provided for varying values of the given parameters. The report aims to cover the 3 phases of the project

1. Model description and design choices
2. Performance evaluation
3. Comparison with off-the-shelf implementation

Model description and design choices

In the previous report, it was discovered that there were considerable differences between the Iris and Income dataset, with the initial processing and handling of the data requiring separate steps, as described below. Therefore two programs have been written to handle a database each.

Both programs have two distinct phases, the **training phase** (data preprocessing phase) and the **testing phase** (class prediction phase). The training phase consists of storing the features and the class labels of the training dataset after performing necessary transformations, removals and normalizations. The testing phase begins with processing the testing dataset with the same methods and parameters as in the training phase. Then for each datapoint in the test dataset, k most similar points in the training dataset is obtained using a chosen distance metric. Finally a class is predicted for each datapoint from its k closest points using a chosen technique.

kNN Classifier for Iris Dataset

Data preprocessing

The Iris dataset consists of four features with numeric data and one feature with nominal data. From the histogram plots (*fig 1.1*) it can be observed that all four numeric features display normal distributions with not much of a skew, although some are multimodal. This shows that **no transformations** are required for the data although normalization is necessary to provide equal weightage to all the features during classification, since the value of length can be seen to be twice as much as the width. **Min-max** normalization technique was used to scale down all four features to a range of (0, 1). The nominal feature is the class label whose bar plot (*fig 1.2*) shows equal examples of the 3 classes, will be utilised for classification purposes. It can also be observed from the graphs below that the data does not contain any missing values or outliers that needs to be handled.

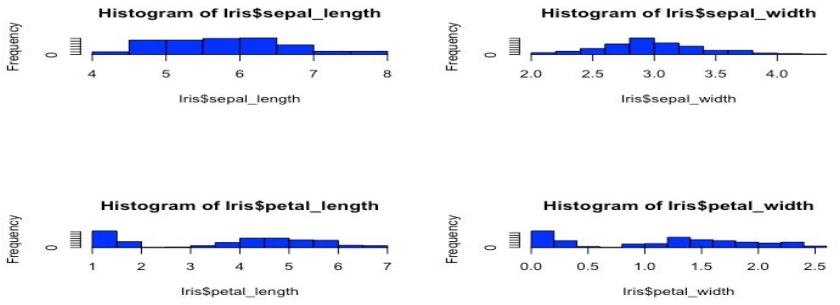


Fig 1.1

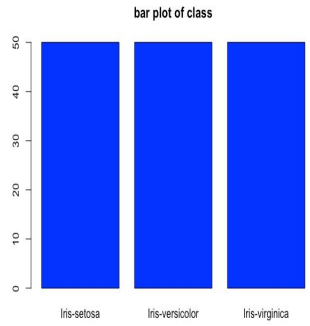


Fig 1.2

Distance measures and class prediction algorithm

To determine the closest k points, **Euclidean distance** and **Cosine similarity** were the two methods chosen to estimate the similarity. The above mentioned methods were specifically selected to compare the results of similarity depending on magnitude and direction of the vector (Euclidean) and similarity depending only on the direction (Cosine). In the case of Euclidean distance, similarity was calculated from distance using the formula s = 1/(1+d), where 'd' is distance and 's' is similarity.

The **voting technique** is used to predict the class because of its ease of implementation, where the most frequent class among the k closest point is assigned to the test datapoint. It was also noted that since the training data had equal number of examples for each class, there was no skew in the training set for classes and the voting technique should perform reasonably well. In case of a tie amongst classes, one of them is assigned randomly since they all have an equal probability of being the predicted class.

kNN Classifier for Income Dataset

Data preprocessing

The Income dataset contains 16 features in total split into 5 ratio features, 2 ordinal features and 9 nominal features. It was observed in the previous report that the column id had unique values used for numbering and held no information about each object. It was also seen that education and education_cat were redundant categories (*fig 1.3*) with education_cat providing a numerical ordering of the education levels. Features **ID and education** were removed to eliminate redundant and uninformative information.

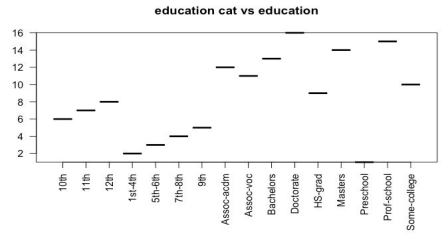


Fig 1.3

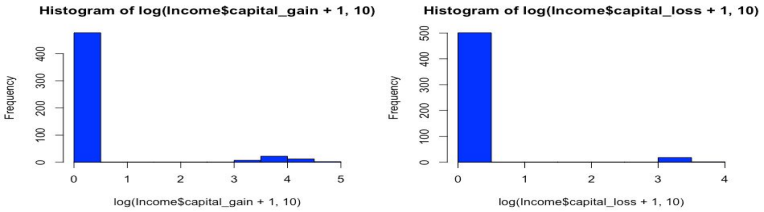


Fig 1.4

During the previous exploratory analysis, it was also observed that the histogram plot of capital gain and capital loss were highly right-skewed and possibly contained outliers, consisting a majority of zeros and few values ranging from 10^3 to 10^5. Therefore a **log transformation** was performed on the data to bring it to a scale that depicted the comparative measure better (*fig 1.4*). It can be seen that this handled the few possible outliers, which unhandled might adversely impacted the similarity calculation. All the ratio attributes were then normalized using the min-max normalization technique to bring the values down to the same range (0-1) for equal weightage.

Distance measures and class prediction algorithm

To calculate the similarity for the data points in the Income dataset, the three different features numerical, ordinal, and nominal had to handled in different ways.

- For the numerical features, Cosine similarity and Euclidean distance were chosen for the reasons mentioned above in the Iris classifier. The formula s=1/(1+d) was used to calculate the similarity value from Euclidean distance.
- The ordinal similarity was estimated using the formula s = 1 - (|p-q|/(n-1)), where p and q are values of the two objects and n is the total number of values.
- The similarity measure used for nominal data was s=1 if values are the same and s=0 otherwise.

The final similarity value was obtained by getting the weighted sum of the three individual measures. To get a fair similarity measure, the weights were distributed proportional to the number of features each category represented. It was seen earlier that two nominal features, namely work class and occupation, had data missing in them. Since nominal similarity measure is a check on whether values are the same, missing data means it doesn't contain any value for comparison and that it could be anything, which would lead to a problem. Occurrence of missing values in a data point was handled by ignoring the attribute for that particular estimation and appropriately altering the weights to take this into account.

The class prediction in the Income dataset also uses the voting technique due to its ease of implementation. In this case though, it was observed that the examples of class "<=50k" (420 examples) greatly outnumbered the examples of class ">50k" (100 examples). The skew number of class examples could influence the majority technique of class prediction, where the class "<=50k" by virtue of its greater number would have a higher chance of being present in the k nearest neighbors. The value of k is chosen to be odd to avoid a tie in the majority voting but in the case of even valued k and a tie amongst classes, one of them is assigned randomly owing to an equal probability of being the predicted class.

Problems, difficulties and assumptions

One of the early mistakes committed in the implementation was processing the test data separately and not on the training parameters, which meant that the test and training data existed in different range of values and therefore resulted in a relatively poor performance of the classifier.

One of the other difficulties faced was accounting for unfamiliar data in the test data set. The existing code was written to handle the training data, but the test data might contain some point which might break the working of the classifier. For instance, during the exploratory analysis it was noted that training data contained missing data in two nominal features but it was discovered that the test data has an extra feature “Native country” containing missing data. Since the algorithm was written to handle missing data in any nominal feature, there were no issues but had it been a numerical or an ordinal feature, it would have caused an issue. The general assumption made in the classifier is that missing data might be present only in the nominal features.

Performance evaluation

In this section we analyze the behavior of the classifier using the various performance metrics that are mentioned below.

1. Produce confusion matrix and error rate for different values of *k* for both datasets.
2. For the Income dataset, compute True Positive, False Positive, True Negative and False Negative rates, Recall, Precision, F-Measure, as well as the ROC curve.

Further detailed analysis of the result is produced using the methods mentioned below.

a. Compute the change in prediction with change in proximity measures.
b. Analyze variation of performance with k, and infer the best value of k.
c. Was there any changes from Report 1

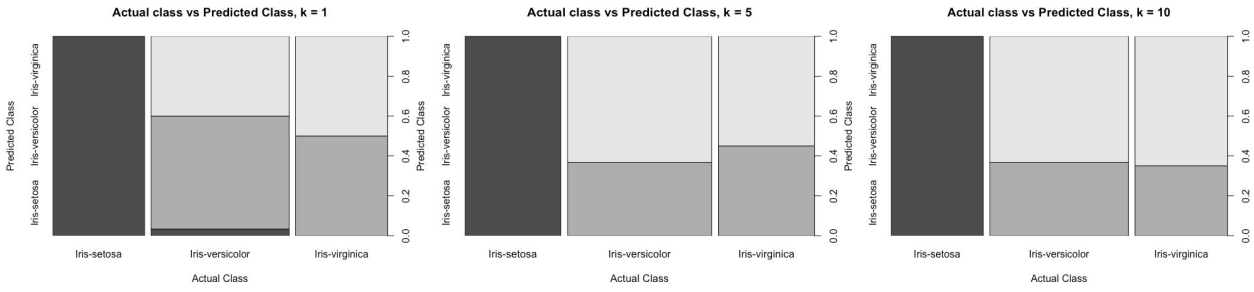
Evaluation of Iris Dataset

Detailed analysis of Iris dataset

From the generated confusion matrix (*fig 2.1 and fig 2.2*), it is observed that the class Setosa is distinct from the other two classes and that there were no classification errors, whereas the classes Virginica and Versicolor have features that are quite close and aligned along the same direction. This can be inferred by the fact that the confusion matrix displays a greater overlap when the similarity measure is chosen to Cosine similarity than when the similarity measure is Euclidean distance

The bar plot of the actual class vs the predicted class represents the confusion matrix visually and provides an easier way to infer patterns. The bar plots also confirm the observation made above, with Setosa being classified only as Setosa but an overlap of classification amongst the other two.

Confusion matrix and Classification Error for Cosine Similarity




```
classification error : 0.328571428571
Iris-setosa Iris-versicolor Iris-virginica
[[ 1.         0.         0.        ]
 [ 0.03333333 0.56666667 0.4       ]
 [ 0.         0.5        0.5       ]]
```

```
classification error : 0.4
Iris-setosa Iris-versicolor Iris-virginica
[[ 1.         0.         0.        ]
 [ 0.         0.36666667 0.63333333]
 [ 0.         0.45       0.55      ]]
```

```
classification error : 0.371428571429
Iris-setosa Iris-versicolor Iris-virginica
[[ 1.         0.         0.        ]
 [ 0.         0.36666667 0.63333333]
 [ 0.         0.35       0.65      ]]
```

Fig 2.1

Confusion matrix and Classification Error for Euclidean Similarity

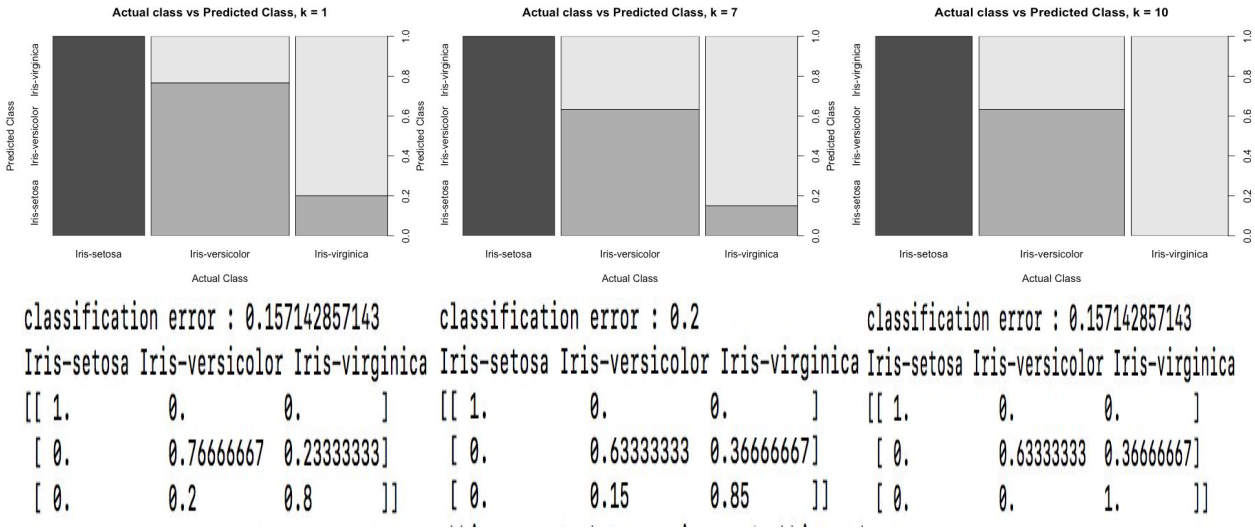


Fig 2.2

Variation with change in proximity measure

From the images below (*fig 2.3*) it can be seen that euclidean distance with an average classification error rate of 17% performs better on the test data set than Cosine similarity, whose error rate is around 37%. Also both proximity measures make zero errors in classifying the Setosa class.

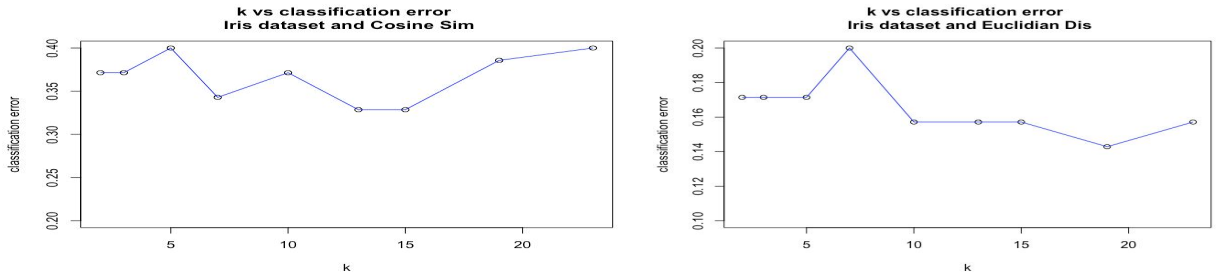


Fig 2.3

From the graphs shown below (*fig 2.4 and 2.5*), it can be noticed that predictions which are correct tend to have a higher confidence value and predictions which are incorrect tend to have a lower confidence value, for both proximity measure. Though Euclidean appears to have a higher confidence in its prediction, especially for the classes Versicolor and Virginica. The class Setosa is always classified with a confidence of 1 in both cases.

K = 5 k = 7 k = 10

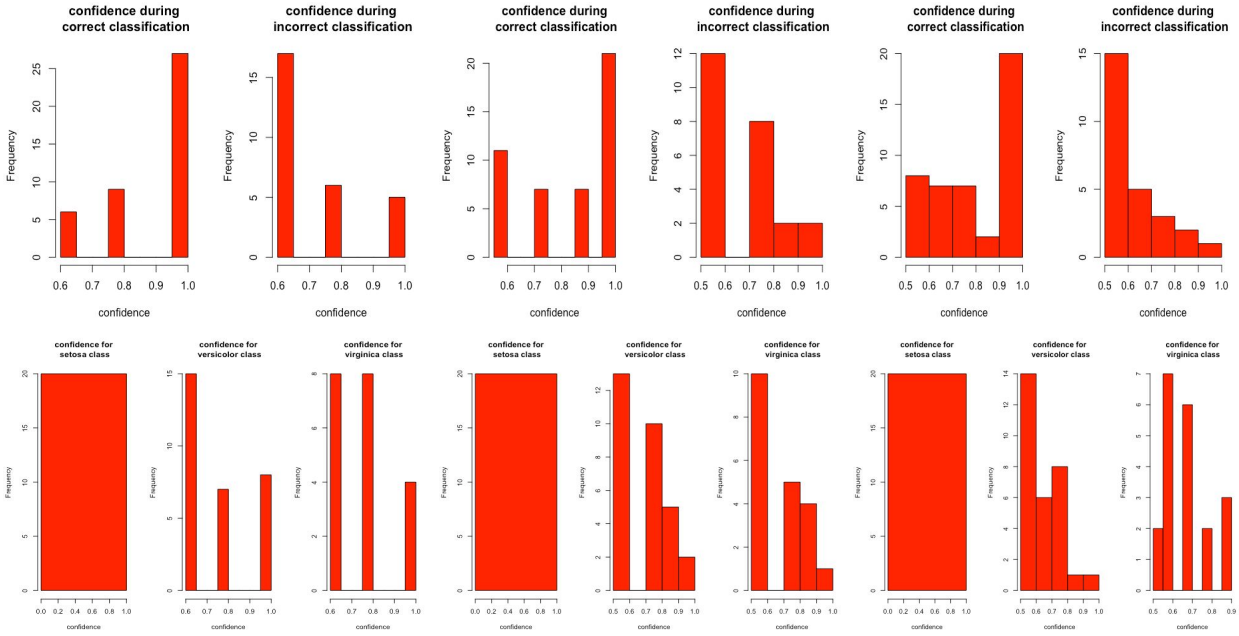


Fig 2.4 - Cosine Similarity

K = 5 k = 7 k = 10

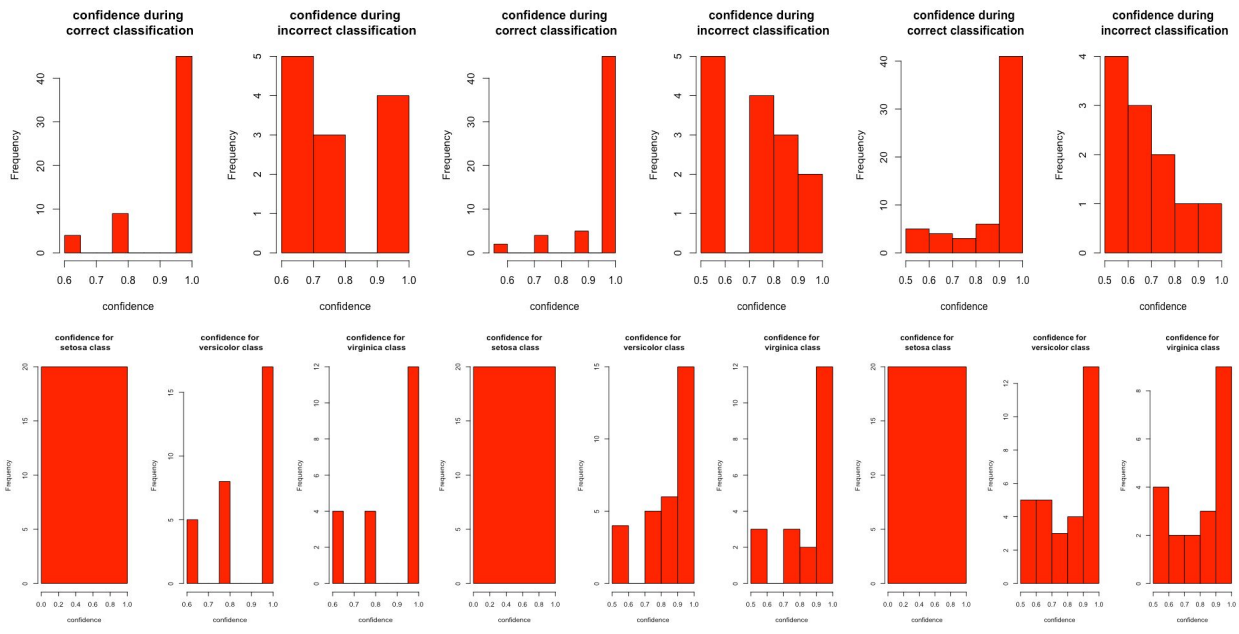


Fig 2.5 - Euclidean Distance

Variation with change in k

For a fixed proximity measure, the plot of classification error for different values of k can be seen in *fig 2.3*. When Cosine similarity is the chosen measure, the error rate fluctuates about 37% as k increases, reaching a minimum of 33% for k =15. It can also be seen in *fig 2.4* that the confidence plot of predictions shifts leftwards for both correct and incorrect predictions. This is due to the decrease in the confidence while predicting the classes Versicolor and Virginica, also shown in the same figure.

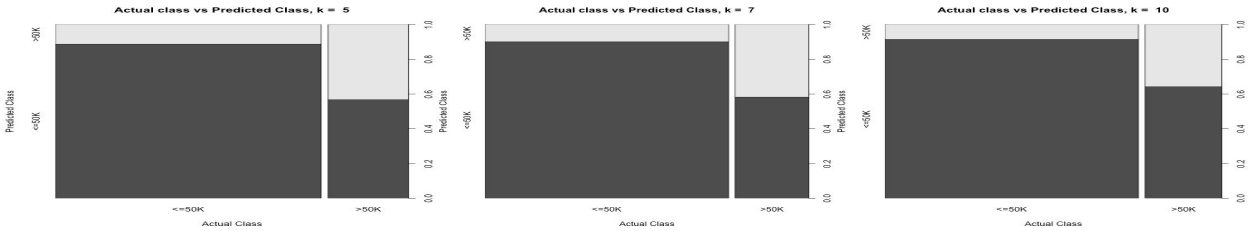
For Euclidean distance, the error rate appears to have decrease slowly with increase in k reaching a minimum of 14%, with an exception at k = 7. The confidence plot of prediction (*fig 2.5*) can be observed shifting leftwards in this case too, for the same reasons as mentioned above.

To determine the best value of k, the value which provided the least classification error for current test set was chosen. Although this value might not be the best value for all test datasets, the values that were chosen are k = 15 for Cosine similarity and k = 19 for Euclidean. The algorithm that was used in the previous report did not require any modifications specific to the test dataset, the similarity estimation process worked as good as it did in the previous report.

Evaluation of Income Dataset

Detailed analysis of Income dataset

From the generated confusion matrix (*fig 2.6 and fig 2.7*) it can be observed that the class “<=50k” has a higher percentage of correct classification (90 %) compared to the 40% of class “>50k” . The patterns inferred could possibly be due to the skewed nature of the training dataset which was mentioned earlier. For analysis purposes, the class “<=50k” was chosen to be the positive class and “>50k” was chosen to be the negative class. From the metrics calculated (*fig 2.6 and fig 2.7*) it can be seen that the classifier predicted a large number of true positives (around 200) and rest of the examples distributed equally among false positives, false negatives and true negatives (around 25).




```
classification error : 0.21875
" <=50K" " >50K"
[[ 0.88687783  0.11312217]
 [ 0.56716418  0.43283582]]
True Positive 196.0
False Negative 25.0
False Positive 38.0
True Negative 29.0
Accuracy 0.78125
Precision 0.837606837607
Recall 0.886877828054
F score 0.861538461538
```

```
classification error : 0.211805555556
" <=50K" " >50K"
[[ 0.90045249  0.09954751]
 [ 0.58208955  0.41791045]]
True Positive 199.0
False Negative 22.0
False Positive 39.0
True Negative 28.0
Accuracy 0.788194444444
Precision 0.836134453782
Recall 0.900452488688
F score 0.867102396514
```

```
classification error : 0.215277777778
" <=50K" " >50K"
[[ 0.91402715  0.08597285]
 [ 0.64179104  0.35820896]]
True Positive 202.0
False Negative 19.0
False Positive 43.0
True Negative 24.0
Accuracy 0.784722222222
Precision 0.824489795918
Recall 0.914027149321
F score 0.8669527897
```

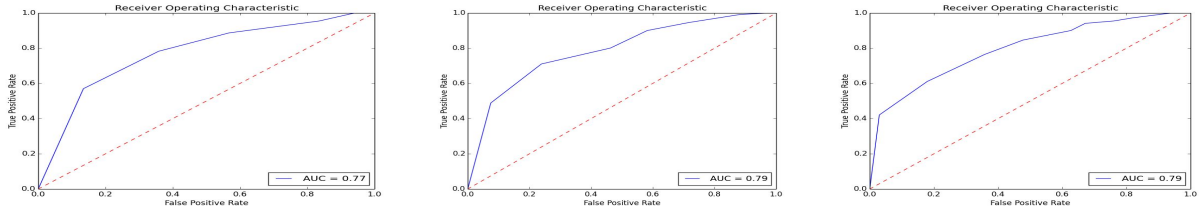
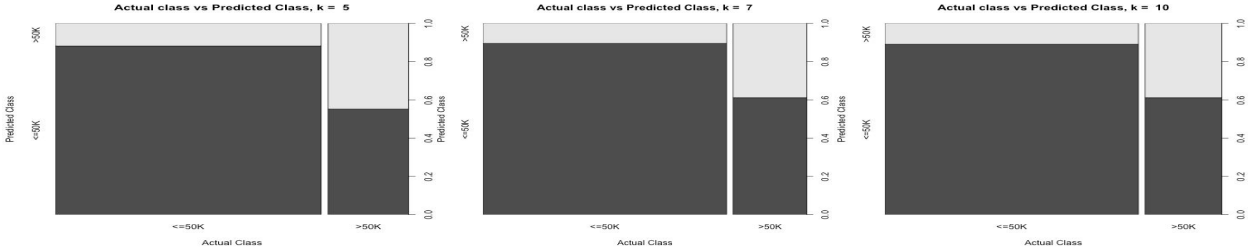


Fig 2.6 - Cosine Similarity




```
classification error : 0.21875
" <=50K" " >50K"
[[ 0.88235294  0.11764706]
 [ 0.55223881  0.44776119]]
True Positive 195.0
False Negative 26.0
False Positive 37.0
True Negative 30.0
Accuracy 0.78125
Precision 0.840517241379
Recall 0.882352941176
F score 0.860927152318
```

```
classification error : 0.222222222222
" <=50K" " >50K"
[[ 0.8959276  0.1040724]
 [ 0.6119403  0.3880597]]
True Positive 198.0
False Negative 23.0
False Positive 41.0
True Negative 26.0
Accuracy 0.777777777778
Precision 0.828451882845
Recall 0.89592760181
F score 0.860869565217
```

```
classification error : 0.225694444444
" <=50K" " >50K"
[[ 0.89140271  0.10859729]
 [ 0.6119403   0.3880597 ]]
True Positive 197.0
False Negative 24.0
False Positive 41.0
True Negative 26.0
Accuracy 0.774305555556
Precision 0.827731092437
Recall 0.891402714932
F score 0.858387799564
```

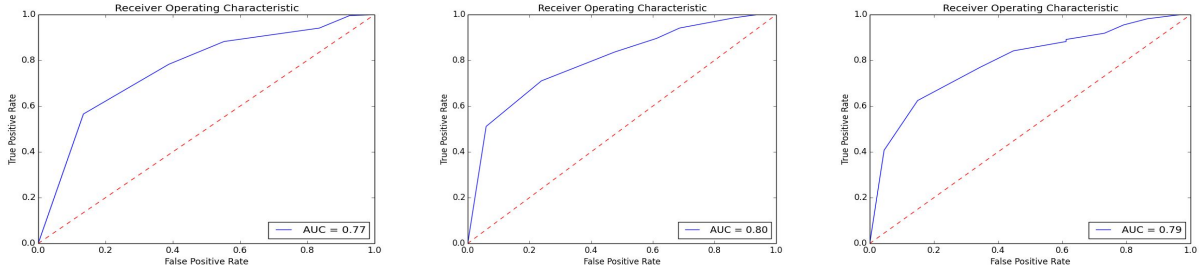


Fig 2.7 - Euclidean Distance

Variation with change in proximity measure

From the data presented in the images above (*fig 2.6 and 2.7*) it can be noticed a change in the proximity measure does not affect the performance of the classifier much. The patterns can be better discerned by plotting the various metrics such as error rate, precision, recall, and F-measure against k for both similarity measures. It can be seen that even though the graphs are not exactly the same, they follow the same pattern and values for all the metrics with Cosine similarity performing a tad better than Euclidean Distance measure. It can be seen in the data shown above, where, for the same values of k, Cosine similarity has a slightly greater value of accuracy, recall and F-score.

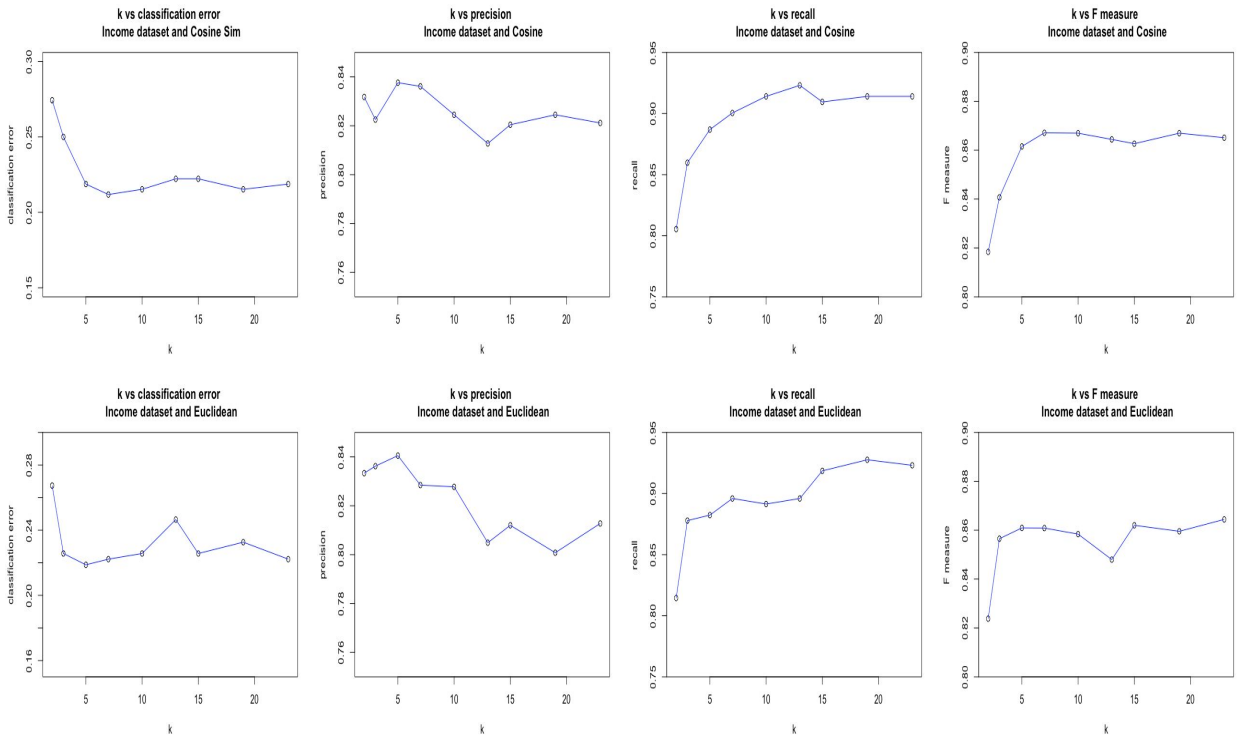


Fig 2.8

Variation with change in k

As the value of k increases, the classification error decreases and then levels out around 22 % for both Cosine and Euclidean measures. The metrics recall and F-index both increase for incremental values of k and then gradually becomes constant, about 91% for recall and 86 % for F-measure.

Similar to the observation made for Iris dataset, it can be seen that the predictions which were correct has a left skewed distribution with a peak around 1 and incorrect predictions have a right skewed distribution with a peak close to 0.6. Also the confidence of the prediction tends to grow smaller as the value of k increases, making the distribution's peak shift leftwards.

K = 5 k = 7 k = 10

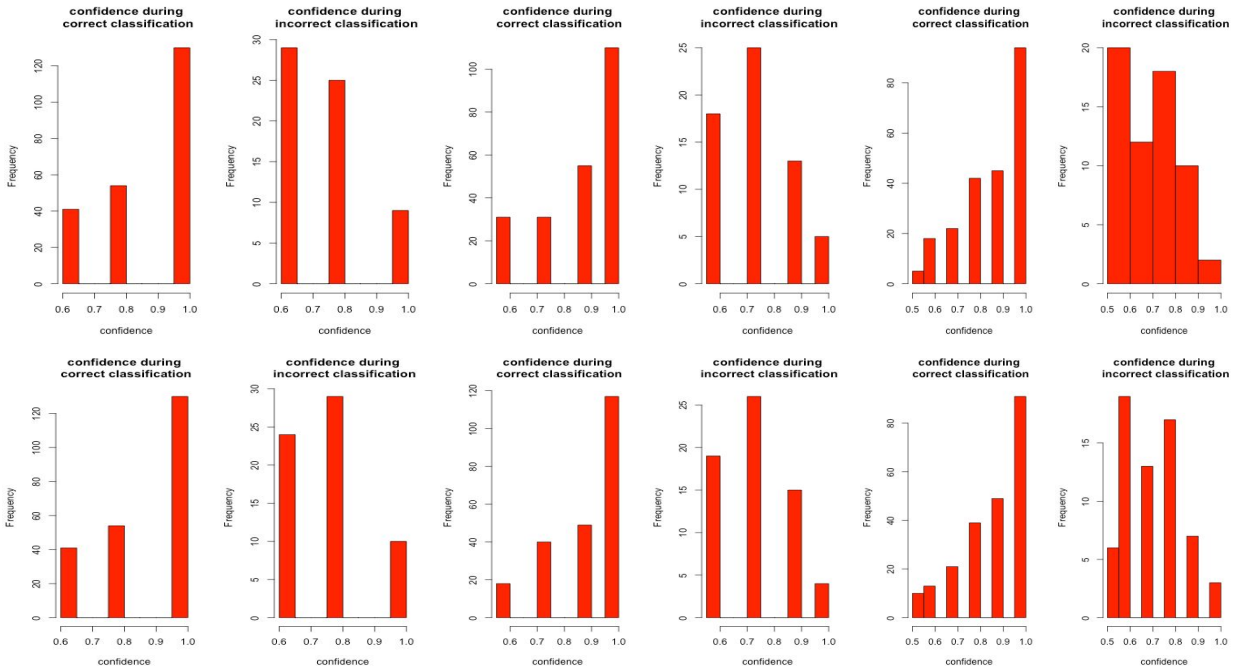
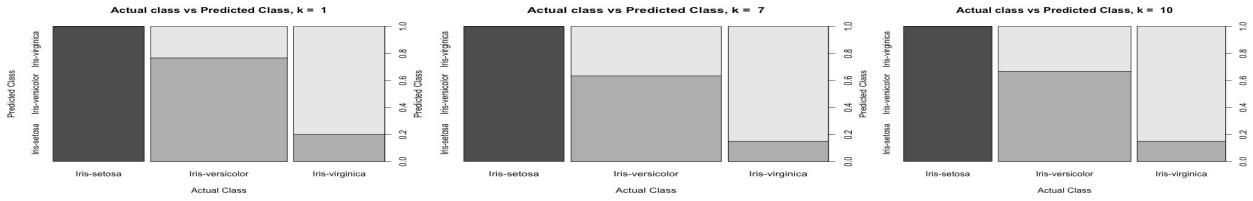


Fig 2.9 : Row 1 - Cosine, Row 2 - Euclidean

To determine the best value of k, the F-measure metric was chosen as it incorporates both precision and recall, providing a good way to compare performance. The value of k that provided the highest value for F-measure for current test set was chosen. The value that was chosen for Cosine similarity was k = 7 and the value chosen for Euclidean Similarity was k = 23. Similar to the Iris dataset, the algorithm that was used here was the same version as the one in the previous report and no modifications were required for it to work well.

Comparison with off-the-shelf classifiers

To compare the performance of the custom built classifier, the KNeighborsClassifier provided by scikit-learn was chosen. The performance on the Iris data set was selected to be compared upon with the off-the-shelf classifier being trained and tested using the same training and test data set. The data was preprocessed using the same method, min-max normalization of the four numeric features. The parameters for the classifier was set to default values, weights = uniform, algorithm = auto, distance metric = Minkowski, p=2. The metric Minkowski with power as 2 is equivalent to the Euclidean measure of calculating distance.




```
classification error : 0.157142857143
Iris-setosa Iris-versicolor Iris-virginica
[[ 1.         0.         0.        ]
 [ 0.         0.76666667 0.23333333]
 [ 0.         0.2        0.8       ]]
```

```
classification error : 0.2
Iris-setosa Iris-versicolor Iris-virginica
[[ 1.         0.         0.        ]
 [ 0.         0.63333333 0.36666667]
 [ 0.         0.15       0.85      ]]
```

```
classification error : 0.185714285714
Iris-setosa Iris-versicolor Iris-virginica
[[ 1.         0.         0.        ]
 [ 0.         0.66666667 0.33333333]
 [ 0.         0.15       0.85      ]]
```

Fig 3.1

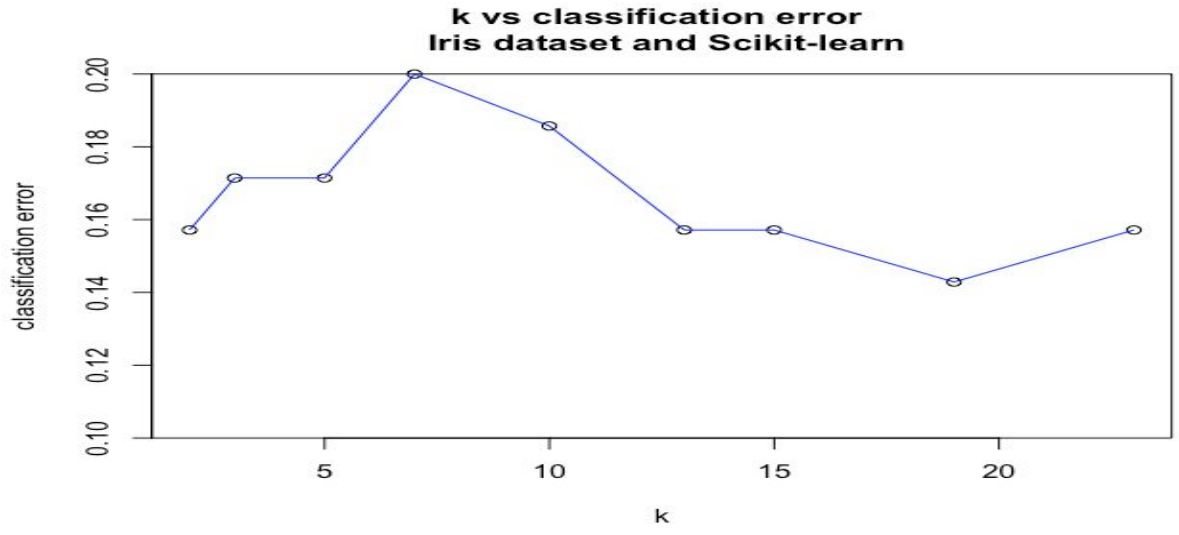


Fig 3.2

The results generated for the scikit-learn classifier can be seen above in *fig 3.1* and *fig 3.2*. Comparing them to the data in *fig 2.1, fig 2.2* and *fig 2.3*, we can observe that although it's performance is better than the custom implementation with Cosine similarity, it is almost similar to the custom classifier with Euclidean measure. In fact, the custom implementation gives lower error rate of 15.7% for k = 10 compared to the 18.5% error rate in the off-the-shelf classifier. The variation of classification error with respect to k in *fig 3.2* is similar to the pattern seen in *fig 2.3* for the custom Euclidean measure. Therefore it can be concluded that the custom built classifier with Euclidean measure performs as good as and at times better than Scikit-learn's KNeighborsClassifier.