CS235 Fall'22 Project Implementation Correctness Report: Heart Disease Classification Using Five Different Classification Techniques

MANOJSAI KALAGANTI #1, NetID: mkala011
ARCHANA KALLAKURI #2, NetID: akall011
SATYA SRI NANDAN PARITALA #3, NetID: pnand006
SREE CHARAN REDDY GANGIREDDY #4, NetID: sgang011
VEDANT CHAUBEY #5, NetID: vchau024

Additional Key Words and Phrases: data, mining

ACM Reference Format:
Manojsai Kalaganti #1, Archana Kallakuri #2, Satya Sri Nandan Paritala #3, Sree Charan Reddy Gangireddy #4, and Vedant Chaubey #5. 2022. CS235 Fall'22 Project Implementation Correctness Report: Heart Disease Classification Using Five Different Classification Techniques. 1, 1 (December 2022), 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 RANDOM FOREST IMPLEMENTED BY MANOJSAI KALAGANTI

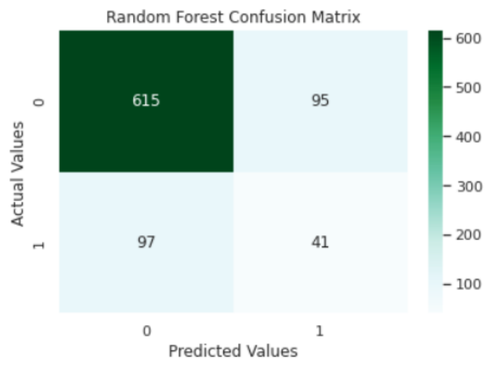
- **Algorithm**: We used a classification algorithm which is made up of numerous decision trees. It is widely known as Random Forest.
  Source: $https://en.wikipedia.org/wiki/Random_forest$
- **Baseline**: For the off-the-shelf implementation and result comparison, we used scikit-learn.
  Source: $https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html$
- **Potential discrepancies**: The maxFeature hyper-parameter was used in the baseline implementation, but not in the main implementation. So, in this case, we would anticipate that every tree will be identical. Therefore, the projections need to be the same regardless of how many trees are there in the forest. Furthermore, we also changed the maxDepth from 80 to 100 in an effort to improve the accuracy.
- **Measures of comparison**: we used 2 different metrics to compare our work with the baseline implementation: accuracy and F1 score. These metrics were useful for determining whether algorithms had an impact on classification and how well they worked.
  Accuracy : It is calculated as the number of correct predictions divided by the total number of predictions, multiplied by 100.
  F1-Score : Classification accuracy is not always a reliable measure of your model's performance. Especially when your class distribution is unbalanced. F1-score, which is the harmonic mean of precision and recall and is

Authors' addresses: Manojsai Kalaganti #1NetID: mkala011; Archana Kallakuri #2NetID: akall011; Satya Sri Nandan Paritala #3NetID: pnand006; Sree Charan Reddy Gangireddy #4NetID: sgang011; Vedant Chaubey #5NetID: vchau024.

commonly used as a metric that combines precision and recall.

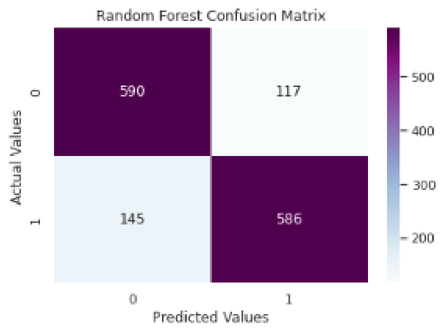
- **Experimental results**:
  Scikit-Learn results:

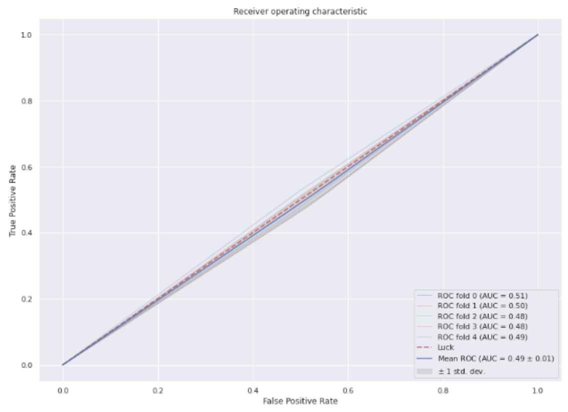
```
Classification Report

              precision    recall  f1-score   support

           0       0.86      0.87      0.86       710
           1       0.30      0.30      0.30       138

    accuracy                           0.77       848
   macro avg       0.58      0.58      0.58       848
weighted avg       0.77      0.77      0.77       848
```



Actual Results:

```
Classification Report for the main implementation

              precision    recall  f1-score   support

           0       0.80      0.83      0.82       707
           1       0.83      0.80      0.82       731

    accuracy                           0.82      1438
   macro avg       0.82      0.82      0.82      1438
weighted avg       0.82      0.82      0.82      1438
```



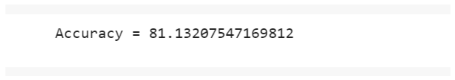


	AUC Train	AUC Test
0	0.864297	0.514085
1	0.848008	0.503443
2	0.861874	0.483312
3	0.864648	0.481706
4	0.856604	0.488890

- **Justification of discrepancies**: We tested a range of n trees and max depth numbers. The settings listed above will allow us to make the most of the RandomForestClassifier in regard to precision. While we employed the built-in library, which had an accuracy of 81.4%, we were able to implement the Random Forest Procedure from scratch and get the accuracy as high as 82%. Because I slightly altered the pre-processing step, I was able to achieve a marginally higher accuracy. Instead of doing it the other way around, I balanced the unbalanced class first before splitting the data this time.

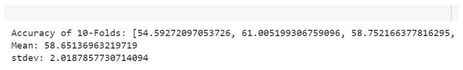
2 GAUSSIAN NAIVE BAYES BY ARCHANA KALLAKURI

- **Algorithm**:We used the Gaussian Naive Bayes Classifier based on Bayes' theorem.Naive Bayes is a of "probabilistic classifier",this supervised machine learning technique is predicated on high feature independence.
  Source: $https://en.wikipedia.org/wiki/Naive_Bayes_classifier$
- **Baseline**: We have used the scikit-learn Gaussian naive bayes implementation as a base-line for our results and comparision.
  Source:$https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html$
- **Potential discrepancies**: The calculation of the probabilities is the same for our model as for the baseline model. But during the design we faced the issue of Zero Frequency error for which I have implemented the Laplacian Smoothing.
- **Measures of comparison**: I have calculated the Accuracy, Precision, Recall and F1 Score using a Confusion matrix. I have taken F1 score as well as our class has imbalanced data and only accuracy is not the best measure in this case. To validate these results I have used 10-Fold Stratified Cross validation. Since our data is imbalanced I have opted for the Stratified Cross Validation. And subsequently calculated the mean and standard deviations of the obtained accuracy of each folds. The Test Accuracy and the 10-Fold Stratified CV accuracy are as below.



```
Accuracy = 81.13207547169812
```

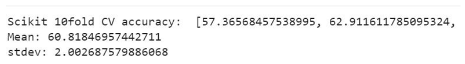
Fig. 1. *Accuracy of our model*



```
Accuracy of 10-Folds: [54.59272097053726, 61.005199306759096, 58.752166377816295,
Mean: 58.65136963219719
stdev: 2.0187857730714094
```

Fig. 2. *Accuracy,Mean, Standard Deviation using 10 Fold Cross Validation*

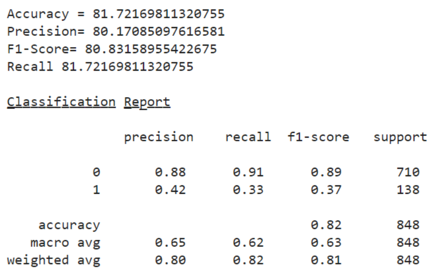
Since there is a gap between the obtained accuracy and the Cross validation accuracy, I have does similar 10 Fold Cross Validation on the Off shelf library(scikit-learn) implementation and obtained similar results. As you can see the evaluation metrics for both models are approximately similar results.



```
Scikit 10fold CV accuracy:  [57.36568457538995, 62.911611785095324,
Mean: 60.81846957442711
stdev: 2.002687579886068
```

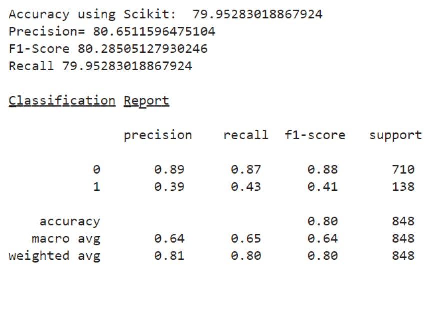
Fig. 3. *Accuracy,Mean, Standard Deviation using 10 Fold Cross Validation in Scikit learn implementation*

- **Experimental results**: I have calculated and compared the Accuracy, Precision, Recall and F1 Score using a Confusion matrix of my model with the results of the baseline implementation. The final evaluation of both models is almost the same and can be seen below.



```
Accuracy = 81.72169811320755
Precision= 80.17085097616581
F1-Score= 80.83158955422675
Recall 81.72169811320755

Classification Report

              precision    recall  f1-score   support

           0       0.88      0.91      0.89       710
           1       0.42      0.33      0.37       138

    accuracy                           0.82       848
   macro avg       0.65      0.62      0.63       848
weighted avg       0.80      0.82      0.81       848
```

Fig. 4. *Evaluation Metrics using our model*



```
Accuracy using Scikit:  79.95283018867924
Precision= 80.6511596475104
F1-Score 80.28505127930246
Recall 79.95283018867924

Classification Report

              precision    recall  f1-score   support

           0       0.89      0.87      0.88       710
           1       0.39      0.43      0.41       138

    accuracy                           0.80       848
   macro avg       0.64      0.65      0.64       848
weighted avg       0.81      0.80      0.80       848
```

Fig. 5. *Evaluation Metrics using Scikit Learn*

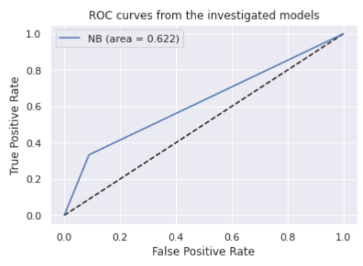


Fig. 6. *ROC curve*

- **Justification of discrepancies**: The difference we see between test and Cross Validation accuracies is caused by the data quality (which we have tried to make better using different Preprocessing techniques like Scaling, SMOTE etc.,)
  Our from scratch model has an accuracy of 81.13% and the built-in model had an accuracy of 80.77% which is very close and hence we can conclude that there aren't any discrepencies in the construction of our model.

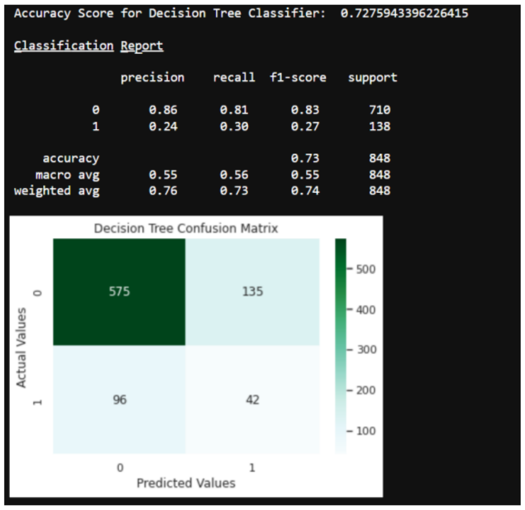
3 DECISION TREE CLASSIFIER- BY SATYA SRI NANDAN PARITALA

- **Algorithm**: Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree. Modified the tree's values,

specifying the maxdepth, minsamples for leaves, and minsample for split parameters, and predicted the output using the model.
Source:*https://www.geeksforgeeks.org/decision-tree-introduction-example/*

- **Baseline**: Used the scikit-learn Decision Tree Classifier implementation as a base-line for our results and comparision. Source: *https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html*
- **Potential discrepancies**: First, we trained the decision tree classifier using the sklearn Python module. Our data was suitable for direct usage with Sklearn after preprocessing. The precision of Decision Trees is controlled by a number of hyperparameters. We used the values "max depth," "min samples leaf," and "min samples split.
- **Measures of comparison**: Accuracy and F1 score were the two measures we used to compare our work to the baseline implementation.
  Accuracy : The accuracy of this classifier is given as the percentage of total correct predictions divided by the total number of instances.
  F1-Score : The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean. It is primarily used to compare the performance of two classifiers.

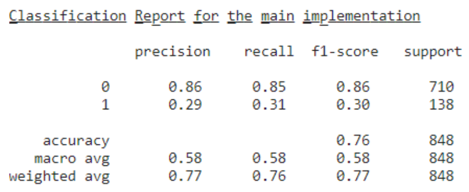
- **Experimental results**:
  Scikit-Learn results:



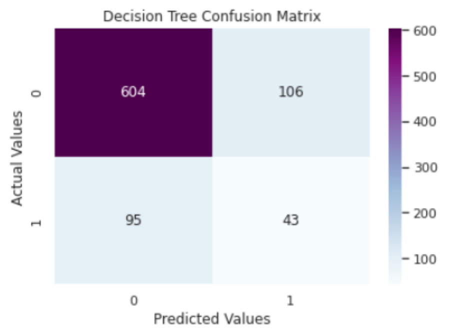
```
Accuracy Score for Decision Tree Classifier:  0.7275943396226415

Classification Report

              precision    recall  f1-score   support

           0       0.86      0.81      0.83       710
           1       0.24      0.30      0.27       138

    accuracy                           0.73       848
   macro avg       0.55      0.56      0.55       848
weighted avg       0.76      0.73      0.74       848
```

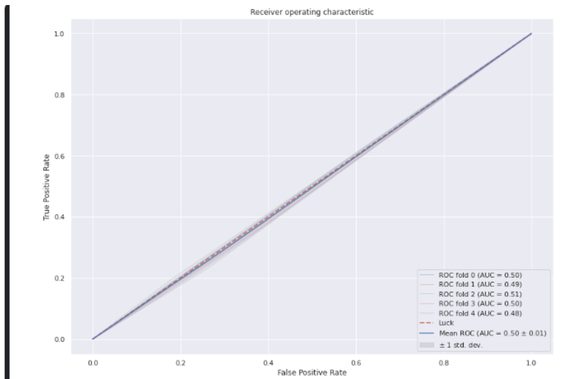
Fig. 5. Classification Report for Decision Tree Classifier

Actual Results



```
Classification Report for the main implementation

              precision    recall  f1-score   support

           0       0.86      0.85      0.86       710
           1       0.29      0.31      0.30       138

    accuracy                           0.76       848
   macro avg       0.58      0.58      0.58       848
weighted avg       0.77      0.76      0.77       848
```





[31...

	AUC Train	AUC Test
0	0.585410	0.504873
1	0.566030	0.490576
2	0.545715	0.507644
3	0.596802	0.496127
4	0.602122	0.484936

- **Justification of discrepancies**: We tested a range of n trees and max depth numbers. While we worked on Decision Tree using library, we could get a total modal accuracy of 73%. After changing some parameters in the algorithm we ultimately were able to reach a total modal accuracy of 75%.The method gave us the ability to investigate a variety of hyper-parameters.

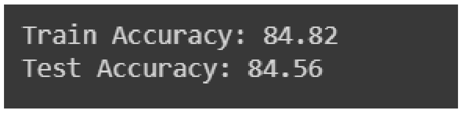
4 SCALAR VECTOR MACHINES BY SREE CHARAN REDDY GANGIREDDY

- **Algorithm**:We have implemented the SVM algorithm. In order to understand the concepts of SVM I have studied the algorithm from the text book "Data mining concepts and techniques" by Jiawei Han (chapter 9) and also refered the video
  Source: $https://www.youtube.com/watch?v = bM4\_AstaBZo$

- **Baseline**:We have used Algorithms from Scikit learn for initial implementation, then compared the the results with the our implementation.

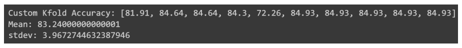
- **Potential discrepancies**:Compared to the baseline implementation we have changed the preprocessing because the data is being bloated by balancing as a result it became difficult for the SVM algorithm to draw a hyperplane. Also we have used cost function and Regularization in the algorithm for optimising weights.

- **Measures of comparison**: We have calculated the Accuracy, Precision, Recall and F1 Score using a Confusion matrix.
  Custom Kfold Accuracy: [81.91, 84.64, 84.64, 84.3, 72.26, 84.93, 84.93, 84.93,84.93, 84.93]
  Mean: 83.24000000000001
  stdev: 3.9672744632387946



```
Train Accuracy: 84.82
Test Accuracy: 84.56
```

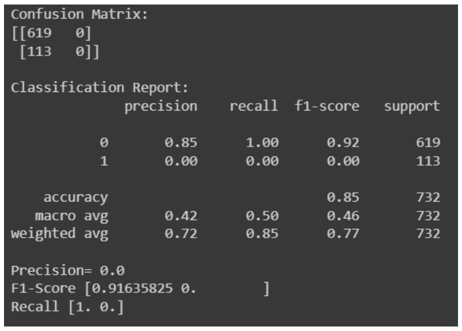
Fig. 7. *Accuracy of our model*



```
Custom Kfold Accuracy: [81.91, 84.64, 84.64, 84.3, 72.26, 84.93, 84.93, 84.93, 84.93, 84.93]
Mean: 83.24000000000001
stdev: 3.9672744632387946
```

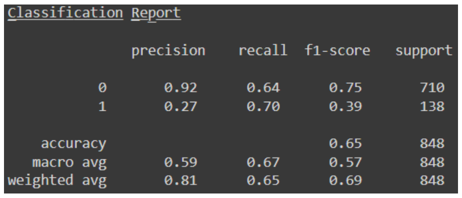
Fig. 8. *Accuracy,Mean, Standard Deviation using 10 Fold Cross Validation*

- **Experiment results**: I have calculated and compared the Accuracy, Precision, Recall and F1 Score using a Confusion matrix of my model with the results of the baseline implementation. The final evaluation of both models is almost the same and can be seen below.



```
Confusion Matrix:
[[619   0]
 [113   0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.85      1.00      0.92       619
           1       0.00      0.00      0.00       113

    accuracy                           0.85       732
   macro avg       0.42      0.50      0.46       732
weighted avg       0.72      0.85      0.77       732

Precision= 0.0
F1-Score [0.91635825 0.        ]
Recall [1. 0.]
```

Fig. 9. *Evaluation Metrics using our model*



```
Classification Report

              precision    recall  f1-score   support

           0       0.92      0.64      0.75       710
           1       0.27      0.70      0.39       138

    accuracy                           0.65       848
   macro avg       0.59      0.67      0.57       848
weighted avg       0.81      0.65      0.69       848
```

Fig. 10. *Evaluation Metrics using Scikit Learn*

- **Justification of discrepancies**: Except with minor preprocessing changes, there are no major difference from the baseline implementation, as even the accuracy that our algorithm achieved is similar to the baseline implementation.

5 K NEAREST NEIGHBORS CLASSIFIER BY VEDANT CHAUBEY

- **Algorithm**:We have implemented the KNN Classifier algorithm. In order to understand the concepts of KNN I have studied the algorithm from the text book "Data mining concepts and techniques" by Jiawei Han (chapter 9) and also refered to the articles on "Towards Data Science" And alos for implementing Area under the CUrve

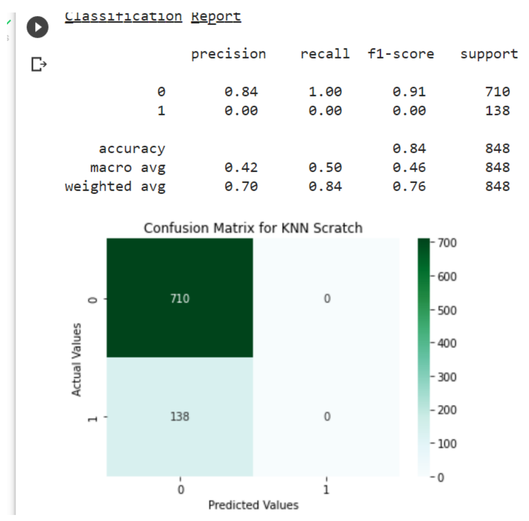
ROC for K fold cross validation I referred to sci-kit documentation

Source: $https : //towardsdatascience.com/knn - algorithm - what - when - why - how - 41405c16c36f$
Source: $https : //scikit - learn.org/stable/auto_examples/model_selection/plot_roc_crossval.$

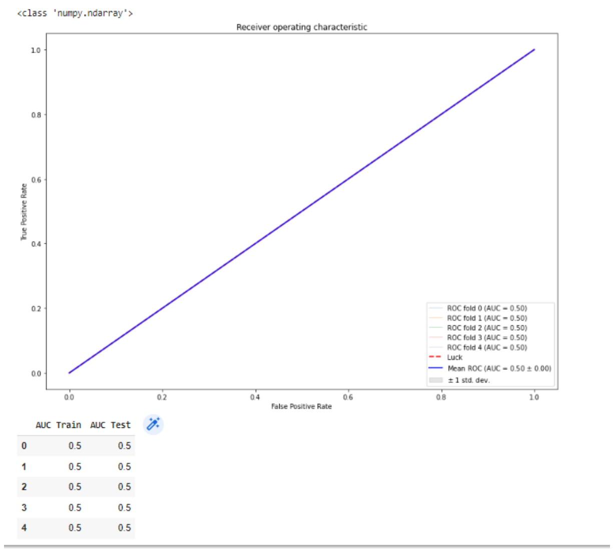
- **Baseline**:We have used Algorithms from Scikit learn for initial implementation, then compared the the results with the our implementation.

- **Potential discrepancies**:Compared to the baseline implementation we have changed the preprocessing because the data is being bloated by balancing as a result it became difficult for the KNN algorithm to fit into the array. Also we have checked for different values of K during the initial research in the algorithm for and came up with the value of K=3.

- **Measures of comparison**: We have calculated the Accuracy, Precision, Recall and F1 Score using a Confusion matrix. To validate these results we have used 10-Fold Stratified Cross validation. and area under the curve of the obtained accuracy of each folds. The Test Accuracy and the ROC graph are as below.



```
Classification Report

              precision    recall  f1-score   support

           0       0.84      1.00      0.91       710
           1       0.00      0.00      0.00       138

    accuracy                           0.84       848
   macro avg       0.42      0.50      0.46       848
weighted avg       0.70      0.84      0.76       848
```



```
<class 'numpy.ndarray'>
```

	AUC Train	AUC Test
0	0.5	0.5
1	0.5	0.5
2	0.5	0.5
3	0.5	0.5
4	0.5	0.5

- **Experiment results**: I have calculated and compared the Accuracy, Precision, Recall and F1 Score using a Confusion matrix of my model with the results of the baseline implementation. I have also compared the accuracy of KNN scratch implimentation to KNN SciKit implimentation for 5 distance metrics. The final evaluation of both models is almost the same and can be seen below.

- **Justification of discrepancies**: Except with minor preprocessing changes, there are no major difference from the baseline implementation, as even the accuracy that our algorithm achieved is similar to the baseline implementation.

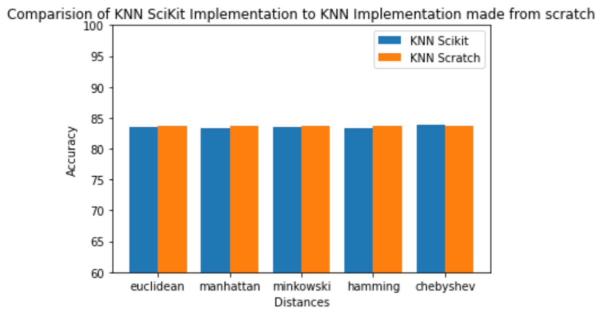


Fig. 11. *Comparasion of KNN scratch implementation to KNN scikit for 5 distances*

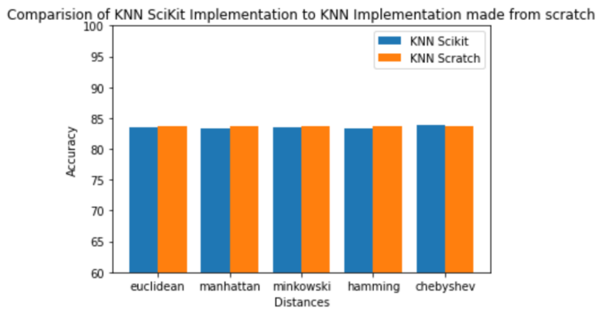


Fig. 12. *Comparasion of KNN scratch implementation to KNN scikit for 5 distances*