**Report**

Knn classification algorithm from scratch using k-fold cross validation has been implemented by using various distance measures like Euclidean distance, Manhattan distance, Chebyshev distance and Minkowski to extend the implementation.

**Steps to run code:**

1.Run the knn.ipynb in Jupyter Notebook.

2.Currently breast-cancer dataset is used. Replace filename with various dataset to see other results.

3.Please input distance measures as string.

Analyzing data inside Weka:

For Weka to understand the data we need to convert the breast-cancer.data file as well as other datasets to the file format with .arff extension. The arff file has 3sections i.e @relation(which gives information about the file name),@attribute(which is the attribute information where we have feature names, feature values, range of the values and datatypes which can be accessed from breast-cancer.names file if opened as a word document)can have two kinds of features either nominal(string) or numeric and @data (which is accessed from the breast-cancer.data file)where the order should be maintained as the features in the attribute section.

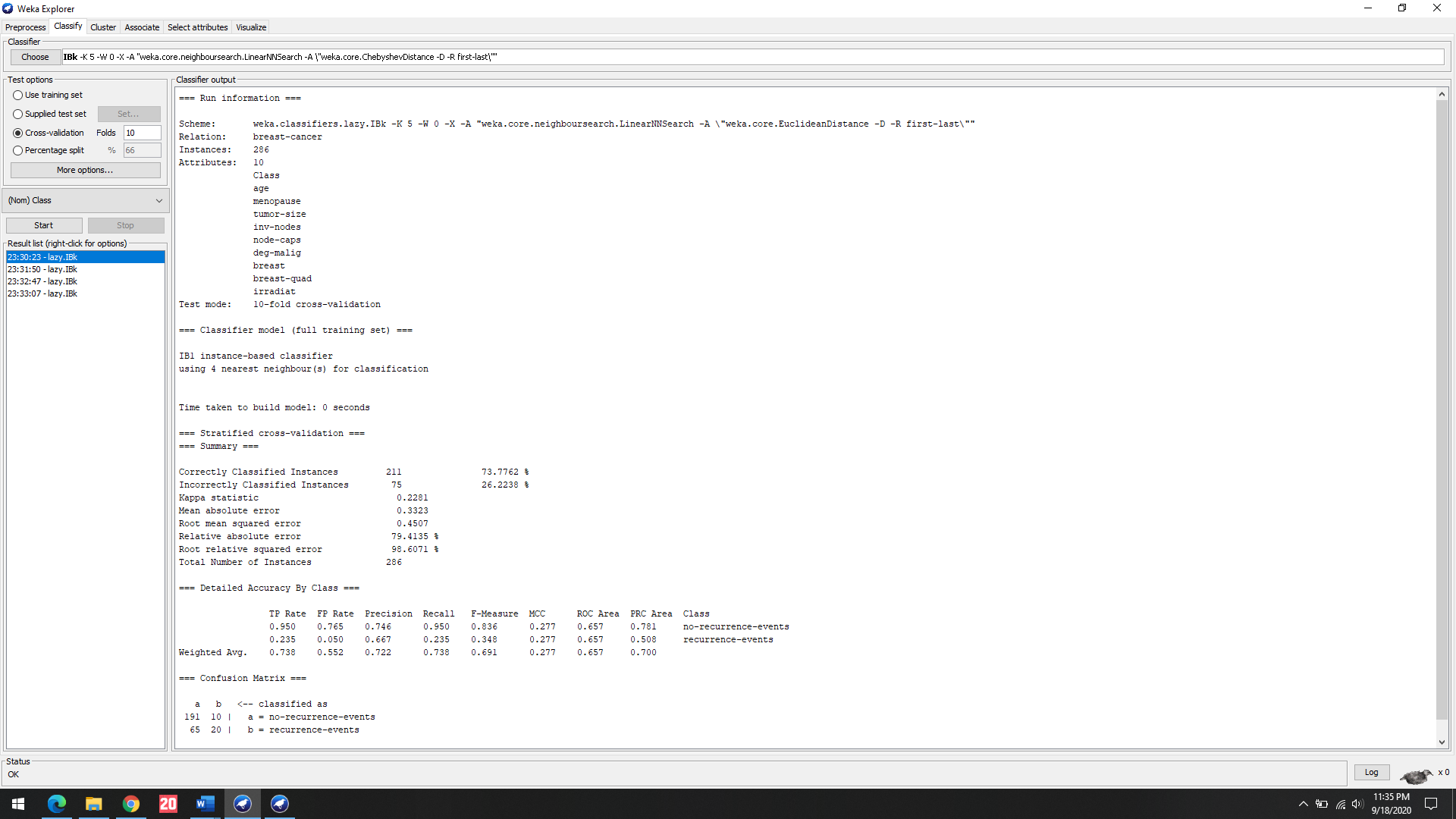
Steps to find accuracy using Weka:

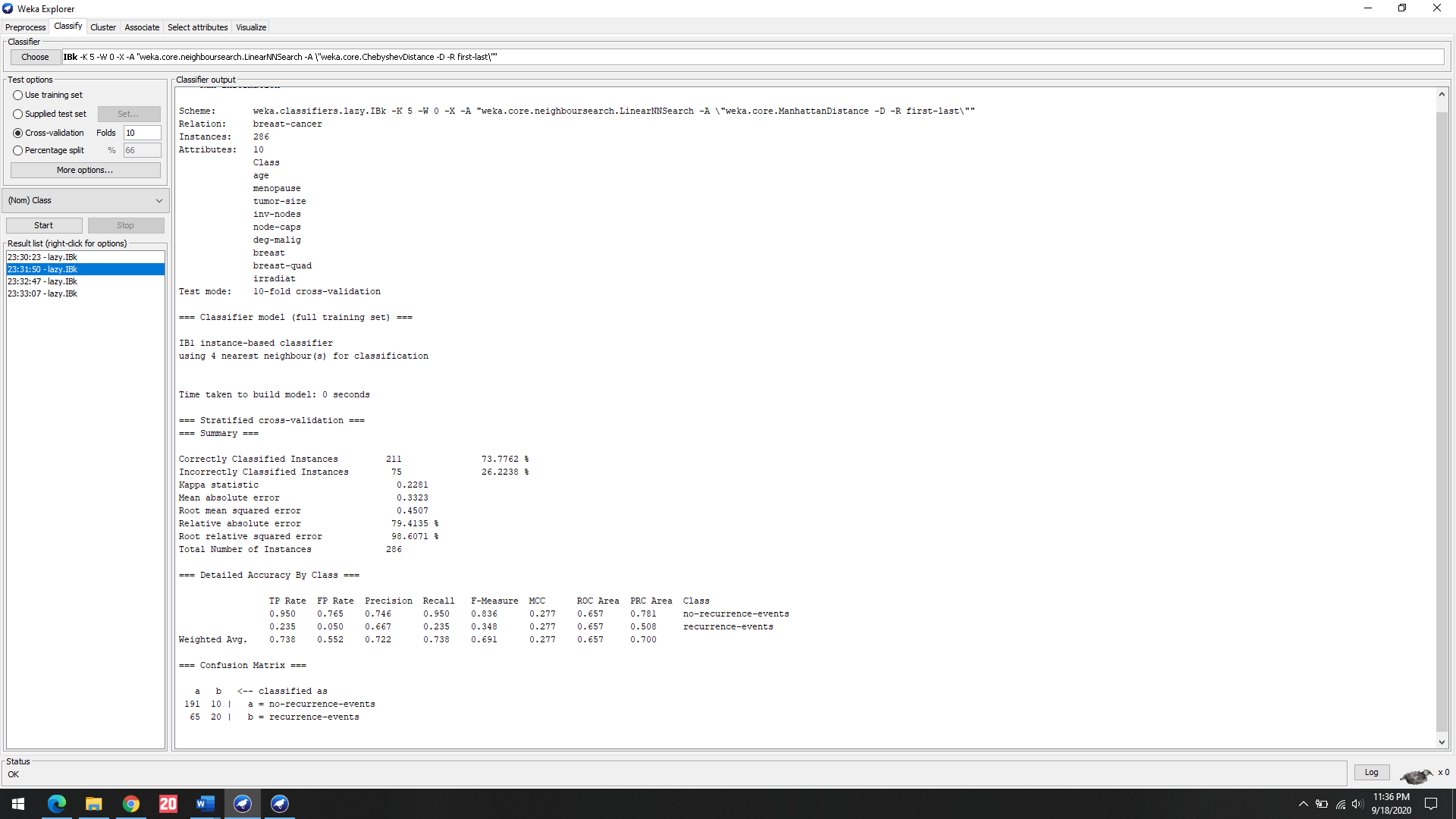
1. Open Weka GUI and choose explorer option.
2. Click open file and select the required .arff file.
3. Click on Classify tab and click choose -> IBK(as it is K nearest neighbor classifier)->click on the textbox adjacent to Choose->a small window pops which helps us choose nearest neighbor search algorithm->KNN enter 10,cross validate as true and select linearNN->Click on the textbox adjacent to Choose again-> Choose Euclidean or any other distance function-> Click Ok-> Start

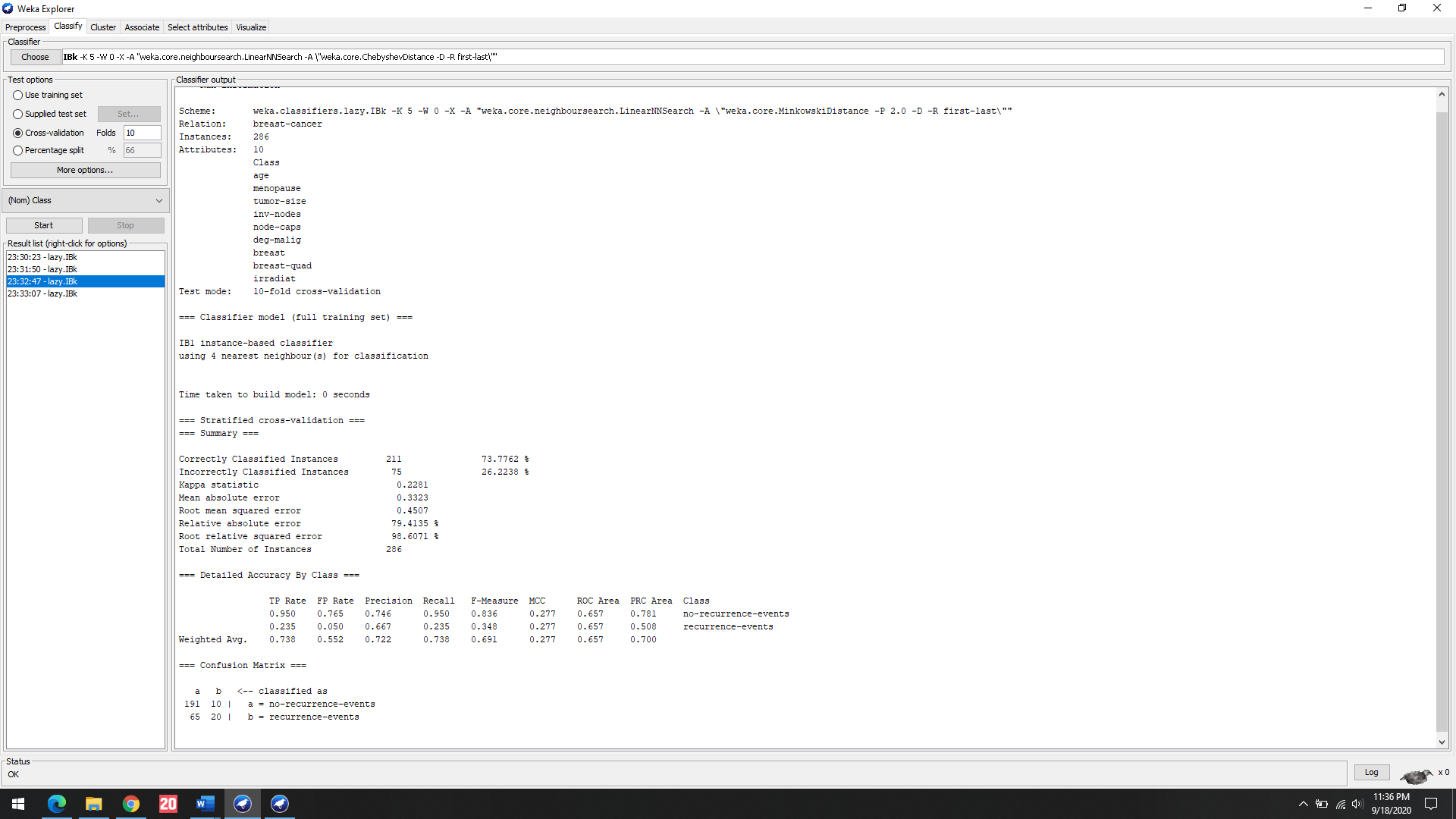
Please find below the comparison of the accuracy of the algorithm using different distance measures with 10-fold cross validation and Weka:

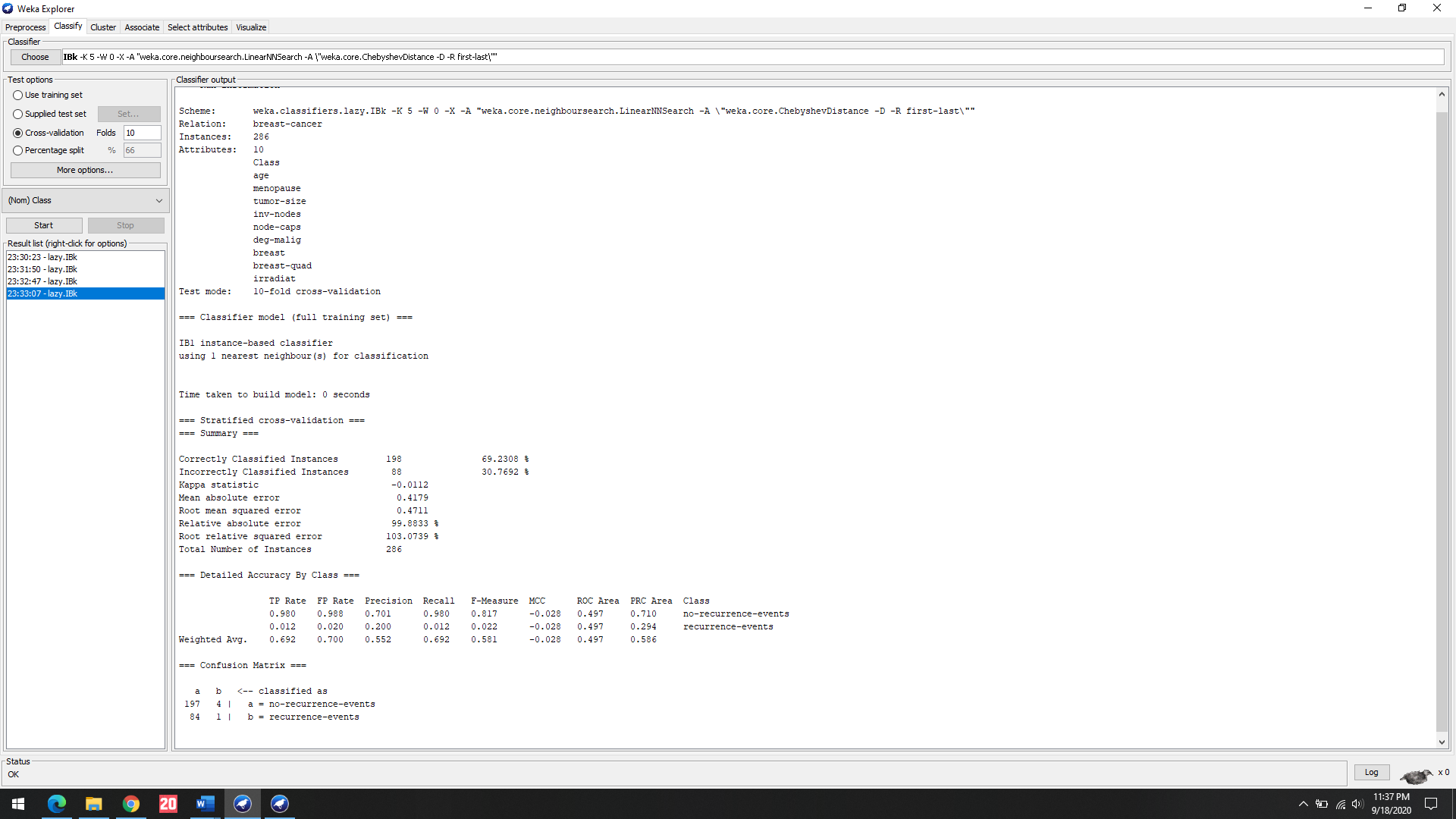
1: Breast cancer dataset

|  |  |  |
| --- | --- | --- |
|  | Weka | k=10 |
| Euclidean Distance | 73. 776% | 73.214% |
| Manhattan Distance | 73. 776% | 73.214% |
| Chebyshev Distance | 69.230% | 73.571% |
| Minkowski Distance | 73. 776% | 76.071% |
| Hamming Distance | - | 74.643% |
| Jaccard Distance | - | 74.643% |



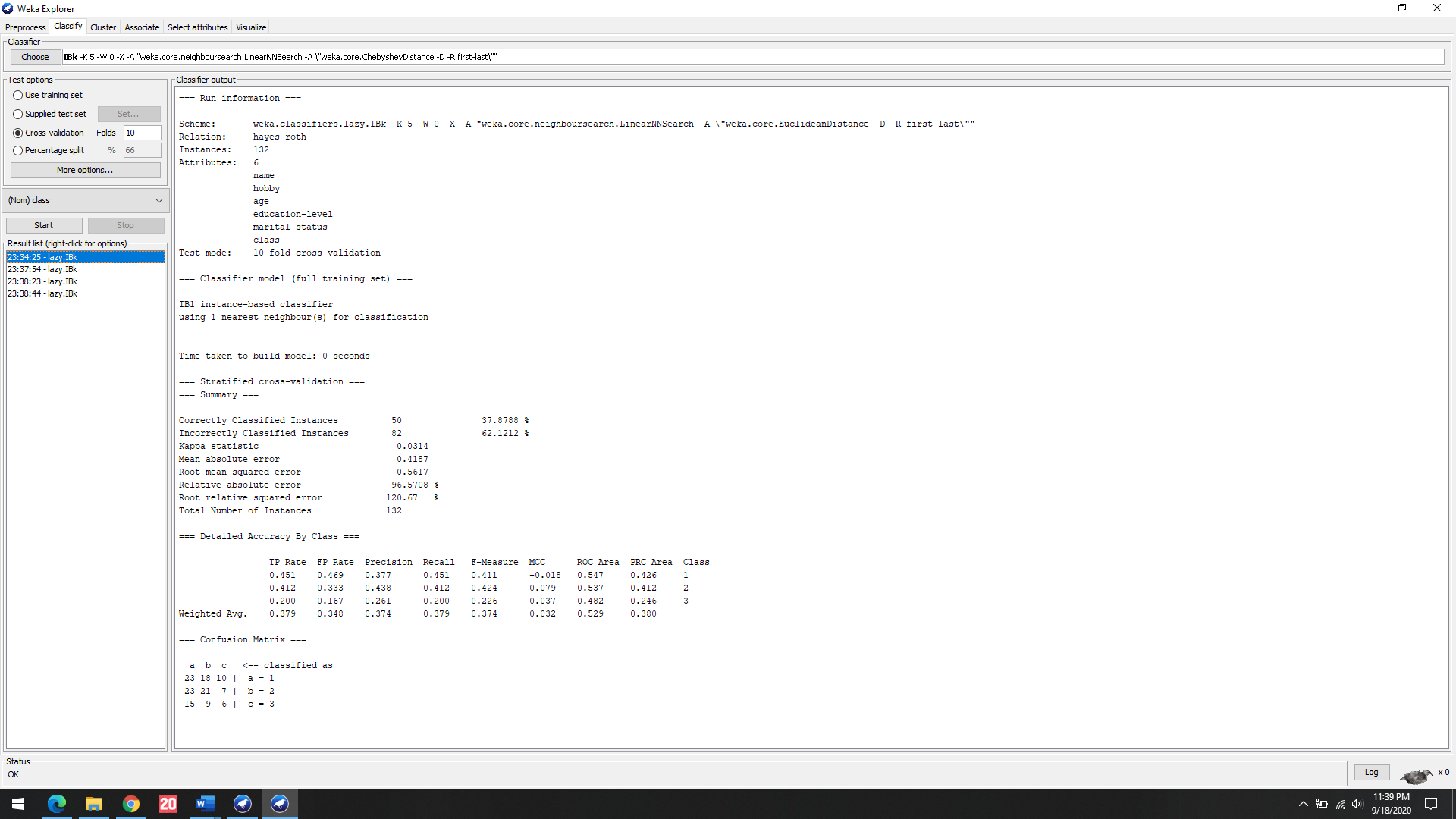


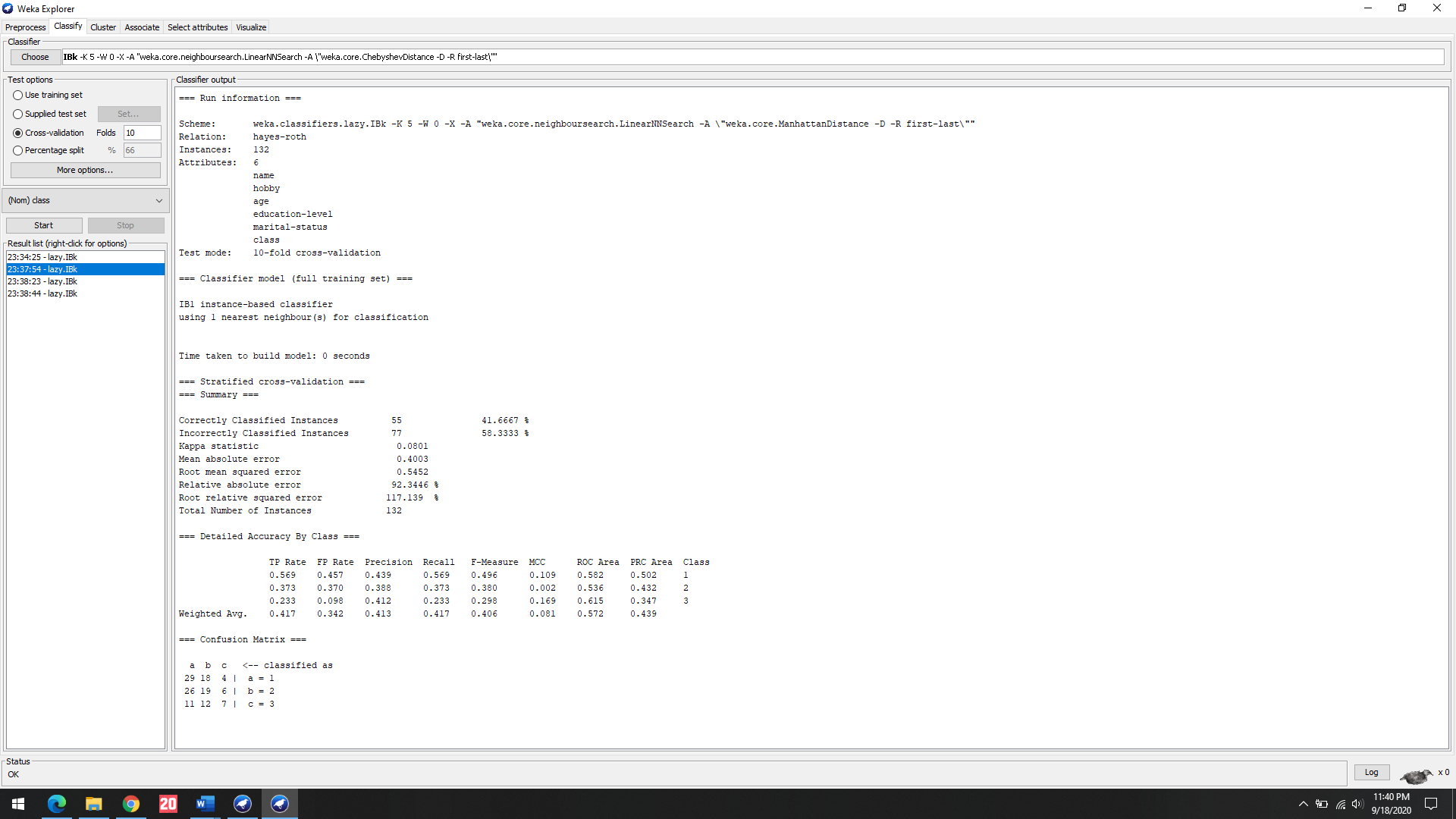


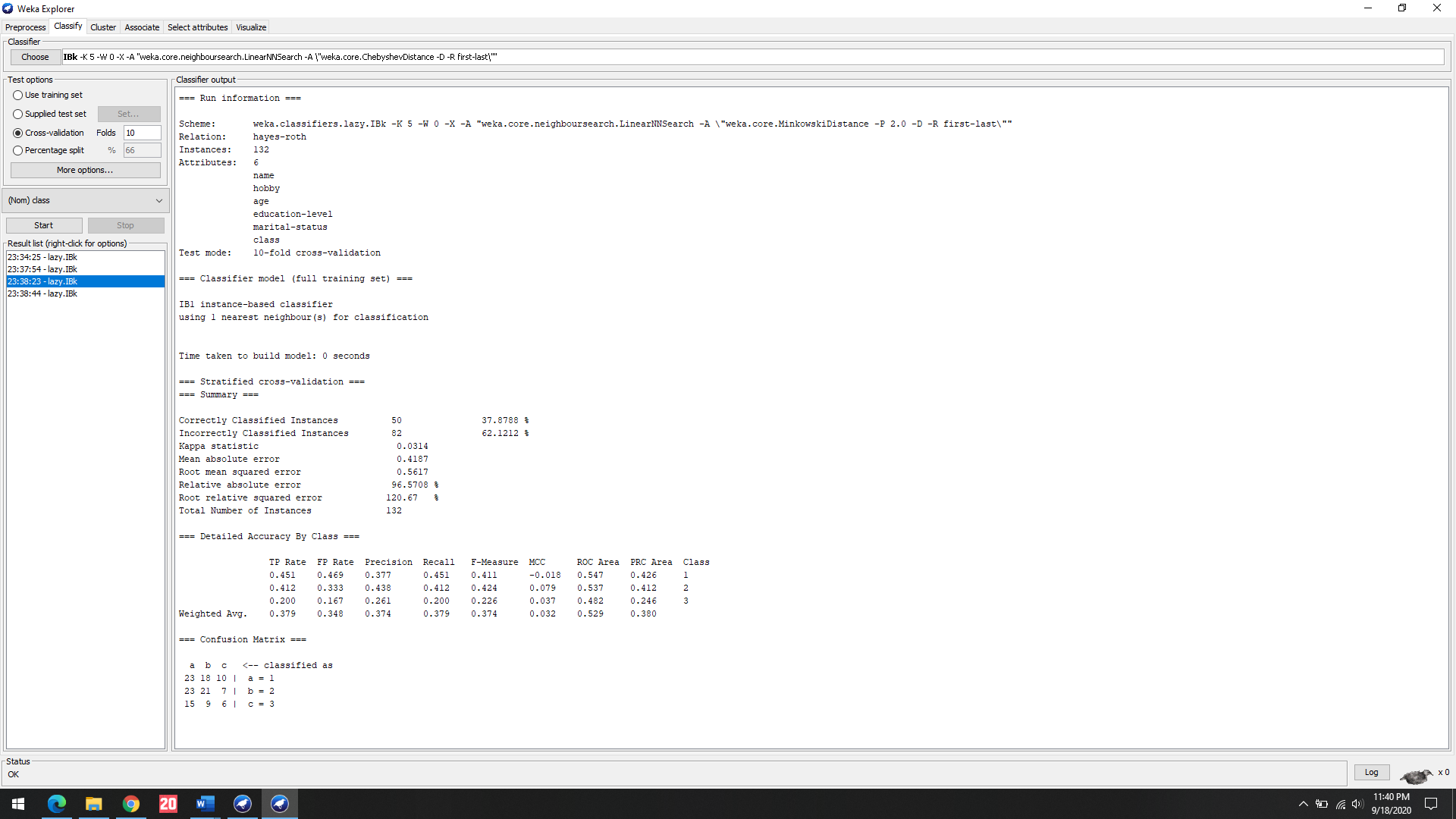


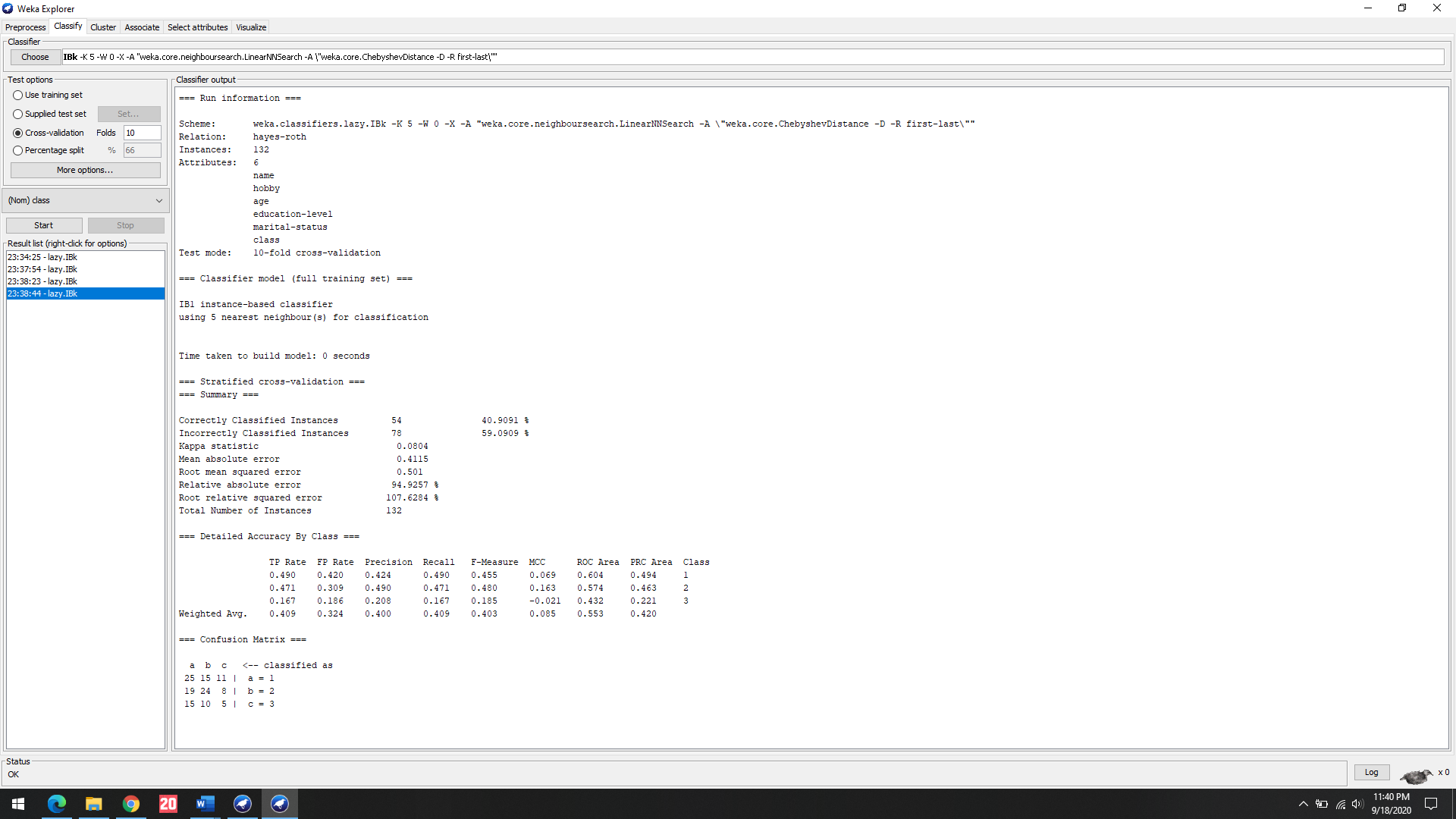
2: Hayes-roth dataset

|  |  |  |
| --- | --- | --- |
|  | Weka | k=10 |
| Euclidean Distance | 37.878% | 36.154% |
| Manhattan Distance | 41.667% | 40.0% |
| Chebyshev Distance | 40.909% | 26.923% |
| Minkowski Distance | 37.878% | 33.846% |
| Hamming Distance | - | 61.538% |
| Jaccard Distance | - | 63.077% |



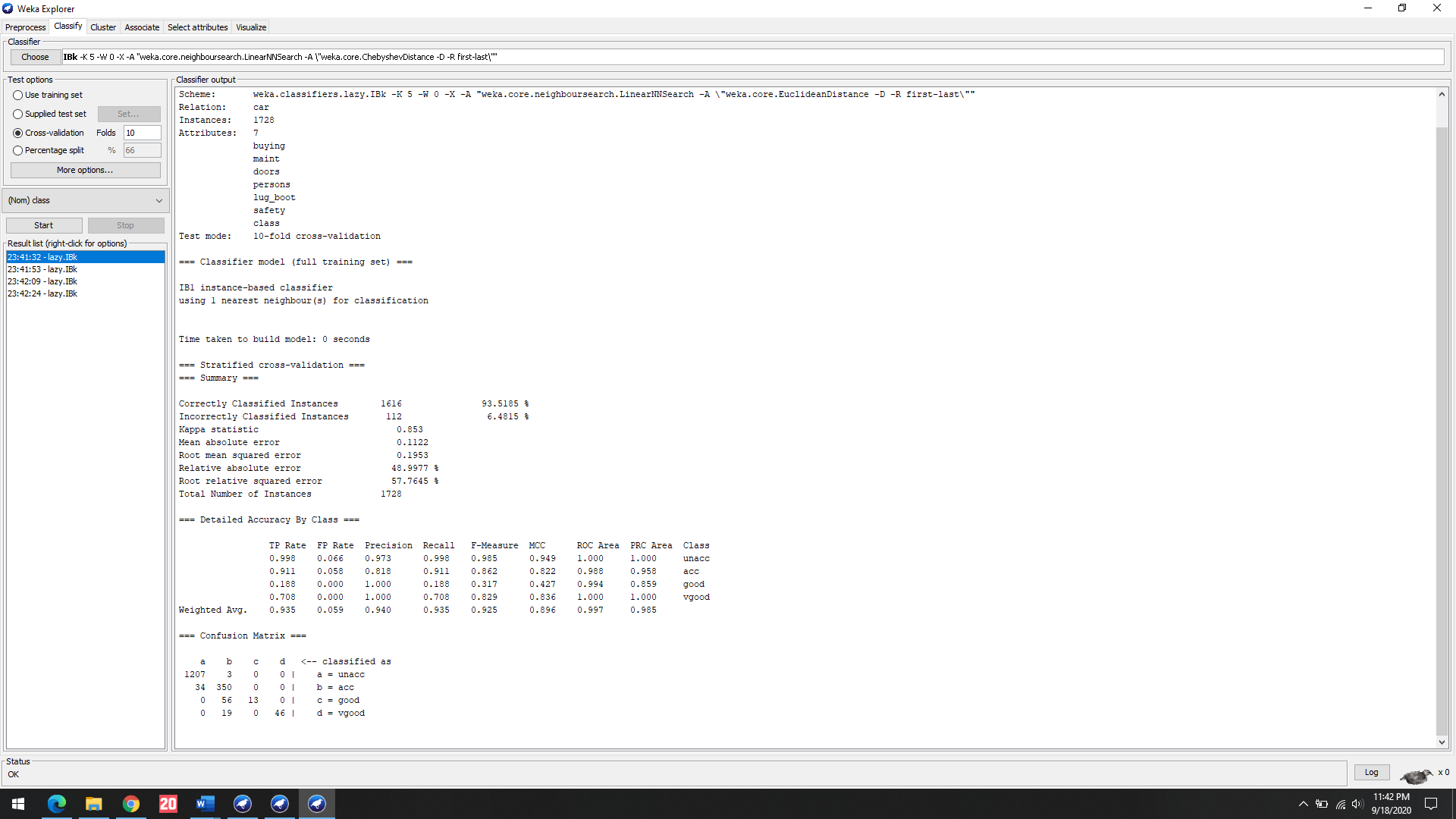


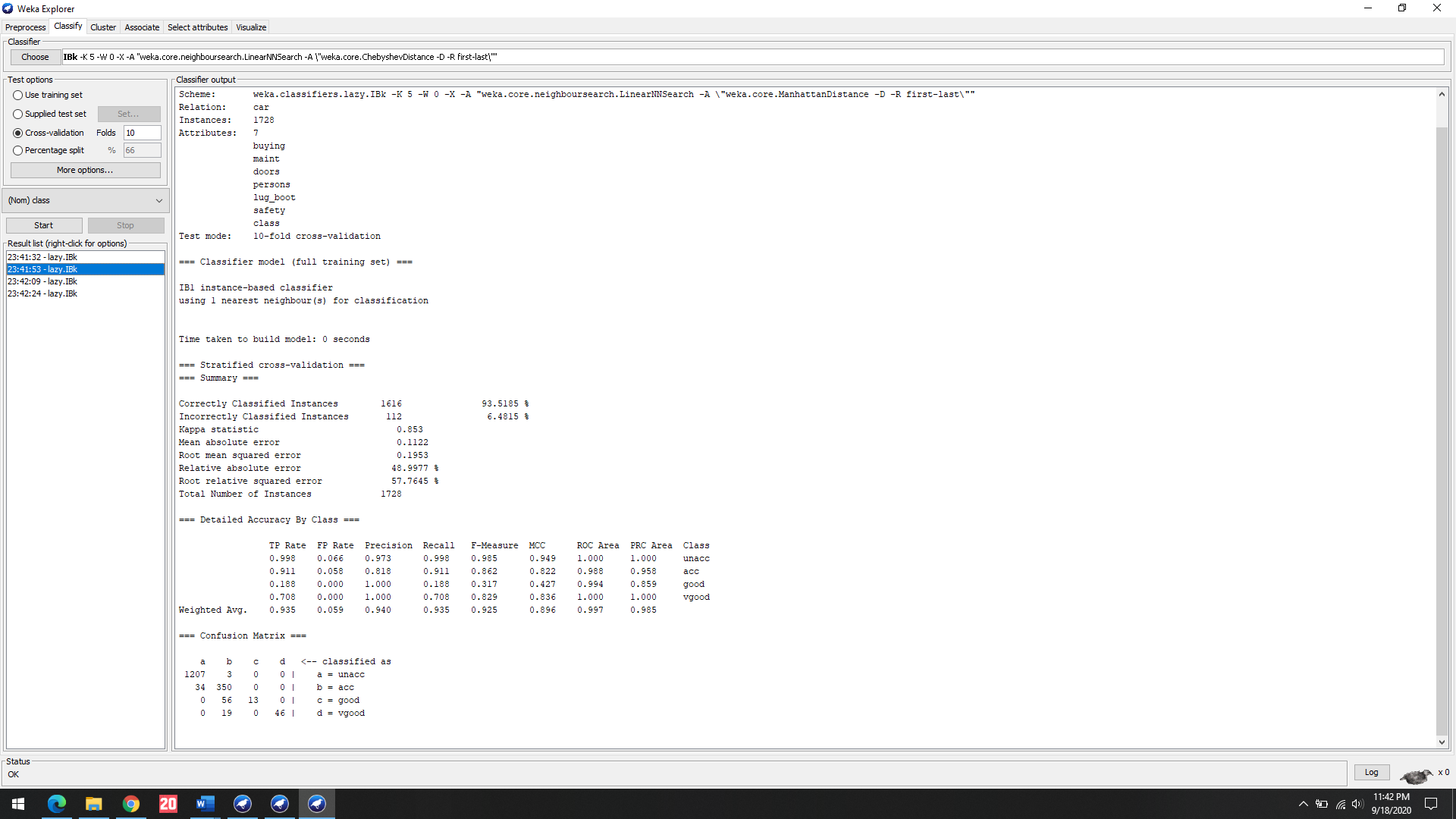


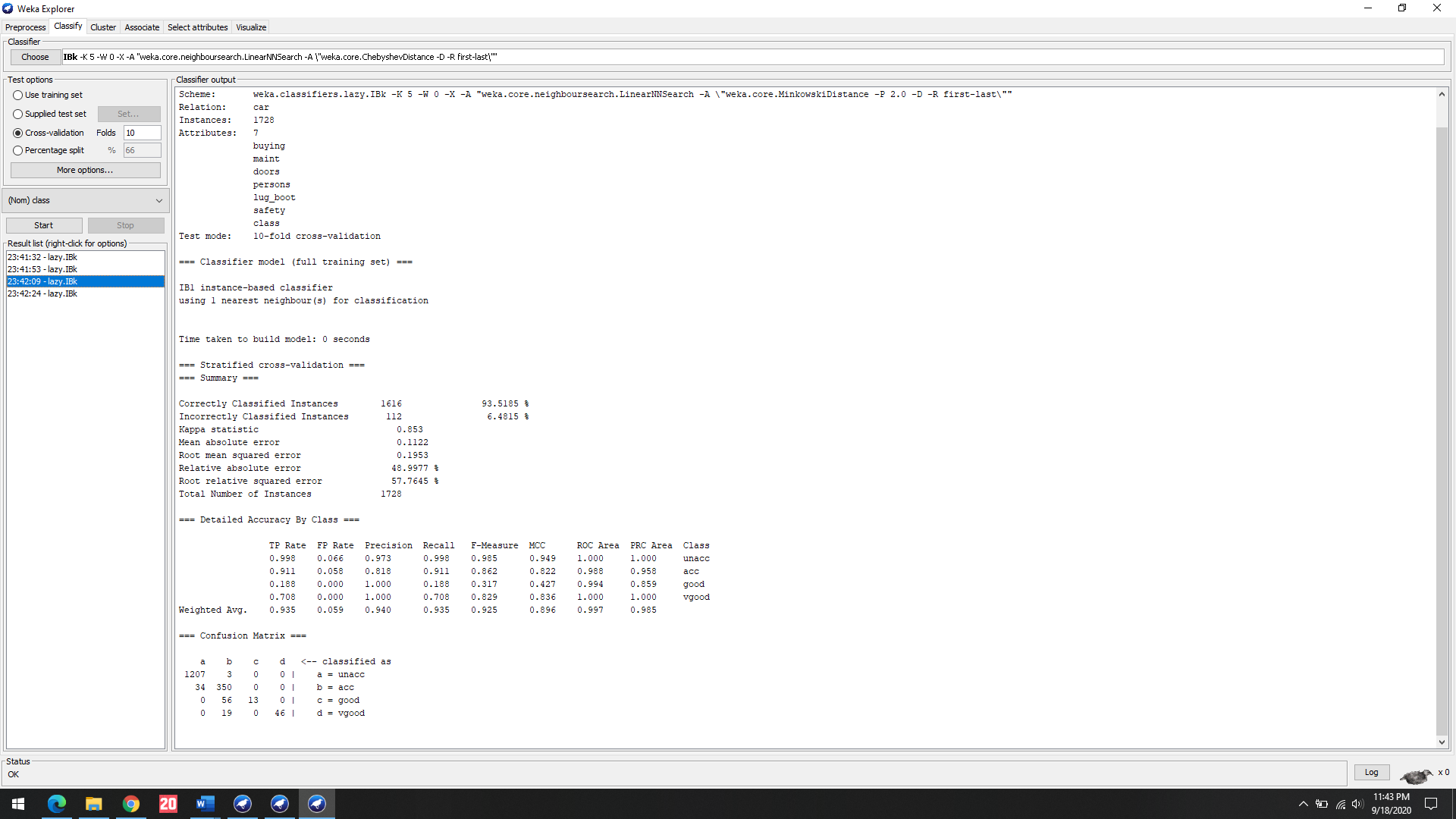


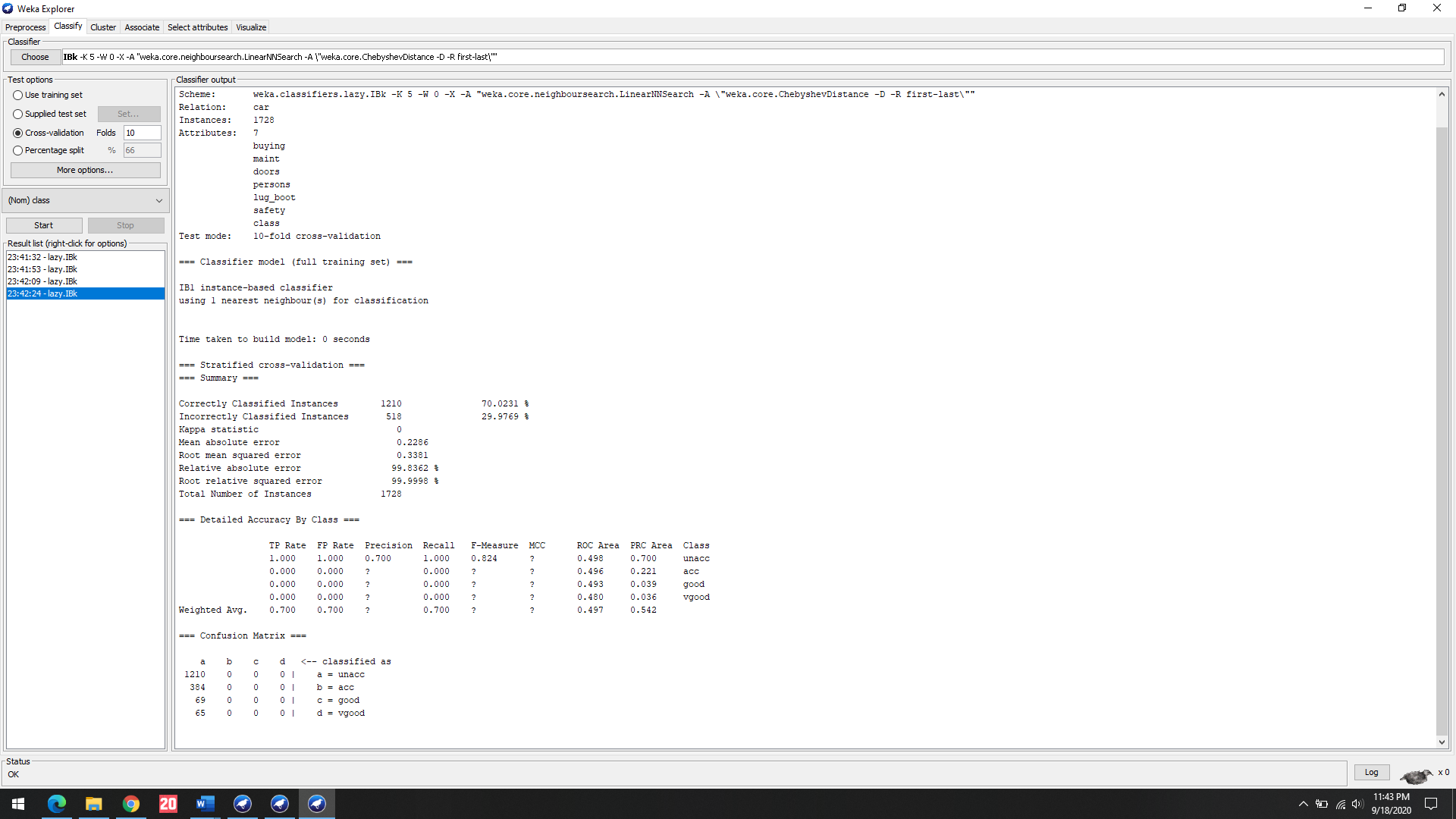
3: Car dataset

|  |  |  |
| --- | --- | --- |
|  | Weka | k=10 |
| Euclidean Distance | 93.518% | 82.5% |
| Manhattan Distance | 93.518% | 90.29% |
| Chebyshev Distance | 70.023% | 67.616% |
| Minkowski Distance | 93.518% | 89.535% |
| Hamming Distance | - | 88.256% |
| Jaccard Distance | - | 64.593% |









References:

1. <https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/>
2. <https://machinelearningmastery.com/k-fold-cross-validation/>
3. <https://machinelearningmastery.com/distance-measures-for-machine-learning/>
4. <https://aiaspirant.com/distance-similarity-measures-in-machine-learning/>

Dataset Links:

<https://archive.ics.uci.edu/ml/datasets/Hayes-Roth>

<https://archive.ics.uci.edu/ml/datasets/car+evaluation>

<https://archive.ics.uci.edu/ml/datasets/breast+cancer>