**PROJECT REPORT**

**INSTANCE-BASED LEARNING AND FEATURE SELECTION**

**CSCE-633: MACHINE LEARNING**

**Submitted By:**

**Amandeep Singh Bhal**

**UIN: 724003595**

Contents

[OBJECTIVE 3](#_Toc449621892)

[RUNNING THE PROJECT 3](#_Toc449621893)

[IMPLEMENTATION 3](#_Toc449621894)

[DATASETS USED 7](#_Toc449621895)

[EFFECT OF K ON VARIOUS DATASETS 10](#_Toc449621896)

[RESULTS 13](#_Toc449621897)

[T-TEST RESULTS 17](#_Toc449621898)

[CONCLUSION 17](#_Toc449621899)

[SCREENSHOTS OF IMPLEMENTATION 19](#_Toc449621900)

# OBJECTIVE

The objective this assignment is to implement an Instance-Based (or nearest neighbor) algorithm with feature selection. Also if possible try to implement the NTGrowth Algorithm. Compare the performance of the algorithm with the best version of Neural Network.

# RUNNING THE PROJECT

The project has been developed in Python. You can simply import the project folder in your IDE eg: PyCharm and then run the project from there. All the datasets and corresponding data files are included in the project folder itself. This makes it very easy to run different datasets as I have already mentioned all the dataset files and corresponding property files in the main method. You just need to go to the main method in the main.py file and uncomment the set of files on which you want to run the project.

# IMPLEMENTATION

I have implemented the following algorithms:

* KNN Algorithm
* Distance Weighting Algorithm (Shepard’s Algorithm)
* Feature Selection using Stepwise Backward Elimination
* NT Growth Algorithm

Implementation Details:

**Pre-Processing of Data:**

The first step of preprocessing was converting the data to be compatible with the system it will run on. This involved converting it from comma-separated values to list of string values. In the pre-processing the data records have been read from the .txt file and have stored it in a list format.

**Data Normalization:**

To prevent irrelevant attributes from overpowering the KNN algorithm we need to normalize the dataset so that each value is between [0,1]. One drawback of normalizing datasets is that it can discard the information and make it hard to analyze the data. For the classifiers used, preprocessing was done before presenting the inputs to the learning model by normalizing the variances of the input values to a range of [0,1] using following equation:

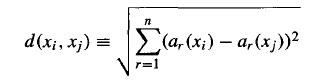
x = x – xmin/xmax – xmin

**Handling Missing Data:**

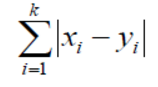
For Discrete attributes I have replaced the missing attribute value with the most frequent value in the dataset for that attribute and for the Continuous attributes I have replaced the missing attribute value with the mean of the values in the dataset.

**KNN Algorithm:**

I have implemented the KNN Algorithm as an Instance based algorithm. In this algorithm we assume that all instances correspond to points in the n-dimensional space. Here n is the number of fields/attributes. The nearest neighbor is defined in terms of the Euclidean Distance for continuous attributes. Thus the distance between 2 instances xi and xj is given by the following formula:



In case of discrete attributes, I have calculated the distance in terms of Hamming Distance which is given by the following formula:



The pseudocode of the algorithm is as follows:

KNN-Algorithm(tData, TraininingData, K)

1. Iterate over tData length taking one tRecord t at a time

Now iterate over all the TrainingData one instance ti at a time

Find the top-k element closest to t.

After finding the top-k element closest to t. Iterate over these elements

Find the majority classifier among these top elements.

Match the predicted class with the class of record t.

Update the accuracy count accordingly.

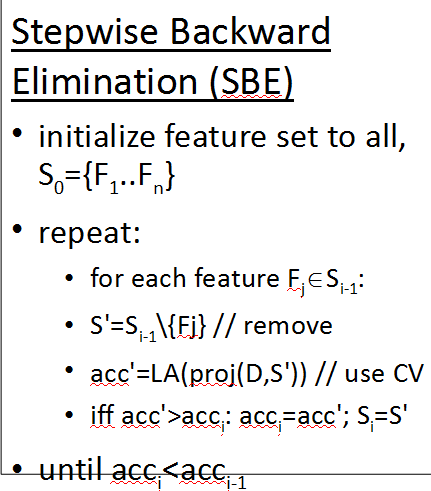
**Distance Weighting Algorithm (Shepard’s Algorithm):**

I have also implemented distance weighting algorithm. It allows all the training examples to have the influence on the classification of the instance x. There is no harm in allowing all the training examples to influence the classification, because the examples which are very far will have very little contribution towards the weight. The weight influences the classification by following formula:

https://lh5.googleusercontent.com/XBCJ_k4mXf9MCGksJnb-CMY2TyzWtf62AoHE97FJ2a7z6Bz9NZd5LrzKAbTeDH4ccxFZALCHKP9TciKa5nz9YB3TSLP1XBZD5_GoTawuxO7xaH7PoKySCQRvAzhFWzI9Yrgz7VxT

**Feature Selection Algorithm: Stepwise Backward Elimination(SBE)**

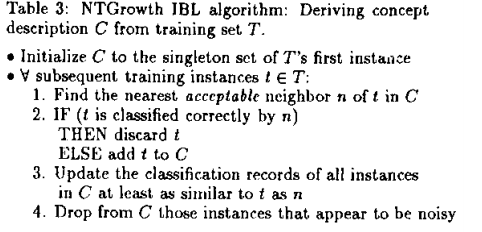
I have implemented Stepwise Backward Elimination for Feature Selection over the DataSets. The pseudo code of the algorithm is as follows:



**NT Growth Algorithm:**

For handling the noisy data, I have implemented the NTGrowth algorithm. Instance based algorithms are highly sensitive to noisy training instances. By implementing NTGrowth noisy data can be detected and removed from the training data. As proved in the Aha and Kibler paper, this method is shown to be performing better in case of noisy data and provides better classification.

Below is the pseudo code of the algorithm:



**10 Cross Validation:**

* We run this whole K-NN Classifier algorithm 10 times using different validation data and test data each time. First step is to randomize the input taken.
* Then we take 10% of training data each time as test data and run the K-NN algorithm to classify using the remaining 90% as training. Each time we take different 10%, that is first iteration is first 10% , second iteration is 2nd 10% and so on.
* The 90% data is further split into 60% and 30% (2/3 and 1/3) as training data and validation data respectively.
* The validation and training data will be used if we do feature selection as explained above.
* We also calculate accuracies and confidence interval like we had done in previous projects.

**Confidence Interval:**

The formula used to calculate this is given below:



# DATASETS USED

1. Tic-Tac-Toe:

This database encodes the complete set of possible board configurations at the end of tic-tac-toe games, where "x" is assumed to have played first. The target concept is "win for x" (i.e., true when "x" has one of 8 possible ways to create a "three-in-a-row"). Interestingly, this raw database gives a stripped-down decision tree algorithm (e.g., ID3) fits. However, the rule-based CN2 algorithm, the simple IB1 instance-based learning algorithm, and the CITRE feature-constructing decision tree algorithm perform well on it.

a. Missing Values: No

b. Attributes: Discrete

1. Credit Screening:

Examples represent positive and negative instances of people who were and were not granted credit. The theory was generated by talking to the individuals at a Japanese company that grants credit.

a. Missing Values: Yes

b. Attributes: Continuous and Discrete

1. Pima Indians Diabetes:

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage. ADAP is an adaptive learning routine that generates and executes digital analogs of perceptron-like devices.

* 1. Missing Values: Yes
  2. Attributes: Continuous

1. Heart:

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1,2,3,4) from absence (value 0).

* 1. Missing Values: Yes

1. Voting:

This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).

* 1. Missing Values: Yes

1. Wine:

These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

* 1. Missing Values: No

# EFFECT OF K ON VARIOUS DATASETS

1. Effect of K on Pima Indians Diabetes DataSet:

As accuracy increases on increasing the K which means that the class clusters are well apart and there may be the case that the data has lot of noise that is scattered throughout the dataset.

1. Effect of K on Heart DataSet:

As accuracy first increases and then remains sort of constant on increasing the K which means that the class clusters are well apart.

1. Effect of K on Tic-Tac-Toe DataSet:

As accuracy increases on increasing the K which means that the class clusters are well apart and there may be the case that the data has lot of noise that is scattered throughout the dataset.

1. Effect of K on Credit Screening DataSet:

As accuracy increases on increasing the K which means that the class clusters are well apart and there may be the case that the data has lot of noise that is scattered throughout the dataset.

1. Effect of K on Voting DataSet:

As accuracy decreases on increasing K which means that the class clusters are close enough.

1. Effect of K on Wine DataSet:

As accuracy decreases on increasing K which means that the class clusters are close enough.

|  |  |  |
| --- | --- | --- |
|  | **Maximum Accuracy** | **Confidence Interval** |
| **Pima Indians Diabetes** | 70.78 (K=7) | +/- 2.30 |
| **Tic-Tac-Toe** | 94.94 (K=9) | +/- 2.12 |
| **Heart** | 81.15 (K=41) | +/- 4.41 |
| **Credit Screening** | 87.23 (K=9) | +/- 3.37 |
| **Wine** | 95.29 (K=6) | +/- 3.24 |
| **Voting** | 92.17 (K=8) | +/- 2.76 |

**General Observation on changing the K over the DataSets:**

There is a general observation over all the datasets that the accuracy of the KNN first increases with the increase in k but then decreases as k approaches the number of features in the datasets. This can be explained by the fact that k should be large enough to minimize the effect of noise in the data and small enough so that it does not include samples from other classes. Therefore as we increase k the accuracy first increases because minimizing the effect of noise but then increasing it further makes the boundaries between classes less distinct.

# RESULTS

1. Tic-Tac-Toe
2. Credit Screening
3. Pima Indians Diabetes
4. Heart
5. Voting
6. Wine

**Observation on Feature Selection:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Mean Accuracy** | **Confidence Interval** | **Best KNN** |
| **Wine** | 94.70 | +/- 2.88 | 95.29[+/- 1.46] |
| **Pima Indians Diabetes** | 69.73 | +/- 4.92 | 70.78[+/- 2.87] |
| **Voting** | 91.3 | +/- 3.06 | 92.17[+/- 2.05] |
| **Heart** | 75 | +/- 6.29 | 81.15[+/- 4.27] |
| **Tic-Tac-Toe** | 93.36 | +/- 3.12 | 94.94[+/- 2.88] |
| **Credit Screening** | 86.61 | +/- 3.37 | 87.23[+/- 2.4] |

\*Best KNN means the KNN using K with max accuracy

On some datasets, like Wine, Voting, Heart and Pima, the accuracy slightly decreases after performing feature selection through SBE. This can be the case when the feature subset used can be lacking sufficient relevant attributes for the KNN to perform well. While on other datasets, like Credit and Tic-Tac-Toe, the accuracy after performing the feature selection is comparable to the KNN. This can be due to fact that there can be some features that do not contribute in classification.

**Observation on Distance Weighting:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Mean Accuracy** | **Confidence Interval** | **Best KNN** |
| **Wine** | 95.88 | +/- 3.50 | 95.29[+/- 1.46] |
| **Pima Indians Diabetes** | 64.86 | +/- 2.76 | 70.78[+/- 2.87] |
| **Voting** | 90.43 | +/- 3.01 | 92.17[+/- 2.05] |
| **Heart** | 79.61 | +/- 5.75 | 81.15[+/- 4.27] |
| **Tic-Tac-Toe** | 65.15 | +/- 2.84 | 94.94[+/- 2.88] |
| **Credit Screening** | 83.38 | +/- 3.44 | 87.23[+/- 2.4] |

\*Best KNN means the KNN using K with max accuracy

For the datasets where the distance weighting has worse accuracy than the KNN, like Tic-Tac-Toe DataSet, Voting DataSet, Pima Indians Datasets and Credit Screening, it can be due to the fact that the data points are already well clustered. For the datasets where distance weighting has more accuracy that the KNN, like Heart DataSet and Wine DataSet, it can be due to the reason that the data points are not well clustered and are distributed all over the feature space.

**Observation on NT Growth:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Number of Data Instances before NT Growth** | **Number of Data Instances after NT Growth** | **Ratio Reduced** | **Mean Accuracy (1NN)** | **Confidence Interval** |
| **Wine** | 161 | 20 | 0.87 | 84.70 | +/- 4.46 |
| **Pima Indians Diabetes** | 692 | 160 | 0.76 | 65.65 | +/- 3.42 |
| **Voting** | 209 | 18 | 0.91 | 92.60 | +/- 4.08 |
| **Heart** | 241 | 53 | 0.78 | 73.46 | +/- 3.34 |
| **Tic-Tac-Toe** | 958 | 184 | 0.80 | 93.36 | +/- 2.35 |
| **Credit Screening** | 653 | 171 | 0.73 | 85.84 | +/- 3.11 |

After performing the NT Growth over all the datasets we observe the maximum ratio reduced is of Wine and Voting Dataset. This can be due to the fact that it may have noisy data or data that is more scattered over the feature space.

# T-TEST RESULTS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Decision Tree** | **Neural Network** | **Instance Based Algorithm (KNN)** | **T-Test significance with Decision Tree** | **T-Test significance with Neural Network** |
| **Wine** | 91.2[+/- 2.24] | 94.2[+/- 1.87] | **95.29[+/- 1.46]** | Y(better) | N |
| **Pima Indians Diabetes** | 70.2[+/- 2.4] | **73[+/- 2.7]** | 70.78[+/- 2.87] | N | N |
| **Voting** | 93.2[+/- 3.89] | **94.5[+/- 3.04]** | 92.17[+/- 2.05] | N | N |
| **Heart** | 61.4[+/- 2.23] | 80.1[+/- 2.23] | **81.15[+/- 4.27]** | Y(better) | N |
| **Tic-Tac-Toe** | 81.27[+/- 2.6] | **98.96[+/- 2.35]** | 94.94[+/- 2.88] | Y(better) | N |
| **Credit Screening** | 85.43[+/- 2.11] | **98.57[+/- 2.78]** | 87.23[+/- 2.4] | N | Y(poorer) |

Using T-Test I have compared the statistical performance of Decision Tree with IBL and Neural Network with IBL. The best among three is shown in bold.

# CONCLUSION

In this project I have implemented the Instance Based Learning Algorithm (KNN), Shepard’s Algorithm, Feature Selection Algorithm (SBE) and NTGrowth Algorithm.

The first set of experiment was done on KNN on choosing the best value of K for a particular dataset. There I found a general observation over all the datasets that the accuracy of the KNN first increases with the increase in k but then decreases as k approaches the number of features in the datasets. This can be explained by the fact that k should be large enough to minimize the effect of noise in the data and small enough so that it does not include samples from other classes. Therefore, as we increase k the accuracy first increases because minimizing the effect of noise but then increasing it further makes the boundaries between classes less distinct. After determining the best value of K rest of the experiments are performed using that k.

Second experiment is done on Feature Selection Algorithms to compare its performance with normal KNN. On some datasets, like Wine, Voting, Heart and Pima, the accuracy slightly decreases after performing feature selection through SBE. This can be the case when the feature subset used can be lacking sufficient relevant attributes for the KNN to perform well. While on other datasets, like Credit and Tic-Tac-Toe, the accuracy after performing the feature selection is comparable to the KNN. This can be due to fact that there can be some features that do not contribute in classification hence removing those features did not affect the accuracy. Feature Selection can increase the generalization capability of the model by selecting the relevant inputs. It was also observed that the speed of the KNN algorithm increases after applying SBE over the dataset. This can be due to the fact that, removing irrelevant and noisy features reduces the dimensionality and thus the search space. The result is that less time is spent on learning.

Thirst experiment is done with NTGrowth Algorithm to compare its performance with normal KNN. After performing the NT Growth over all the datasets, we observe the maximum ratio reduced is of Wine and Voting Dataset. This can be due to the fact that it may have noisy data or data that is more scattered over the feature space. Whereas in other datasets the reduction ratio is comparatively low which can be because these datasets had less noise or the data was dense over the feature space.

Comparing KNN with Neural Network on Heart and Wine DataSet it is clear that the simple, yet powerful k-NN outperforms the complex NN. That leads to the conclusion that a simple distance measure as a similarity metric works better for these datasets than a complicated blackbox search for finding suboptimal weights to exploit any patterns in the data. It also states that these datasets are more difficult to learn with NN. As we can see (refer the class distribution chart in upper section) that the class distribution of Heart DataSet is very unbalanced. This poor class distribution can lead to poor results during the learning phase due to lack of training examples for certain classes and an overpowering of other classes. The k-NN classifier avoids this issue because it has no real learning phase, that is, it does not build a learning model (which also means that new or modified examples do not affect its behavior). This lack of a model serves as a benefit of eager learning in this case.

After performing all the experiments and looking at the results it can be concluded that KNN is a powerful learning algorithm with a very fast training phase and could learn complex target functions. However, its drawback is the possibility to be fooled by irrelevant attributed and noisy data. KNN classification is easy to implement and easy to understand technique.

# SCREENSHOTS OF IMPLEMENTATION

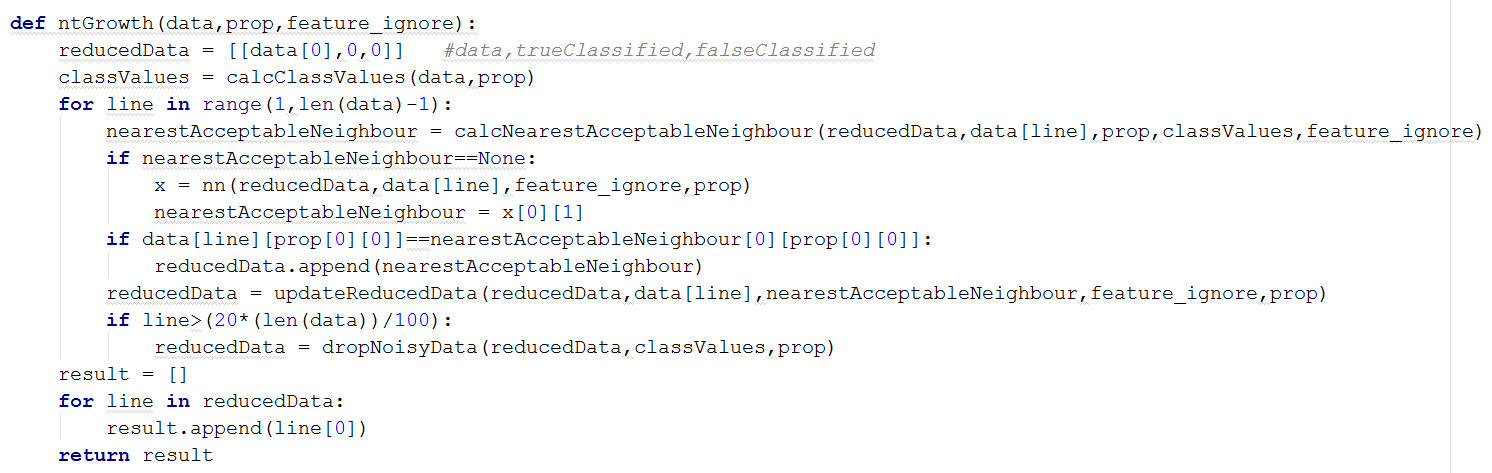
1. KNN Algorithm



1. Stepwise Backward Elimination (SBE)



1. NTGrowth Algorithm



1. Distance Weighting

