Applied Spatial and Temporal Data Analysis

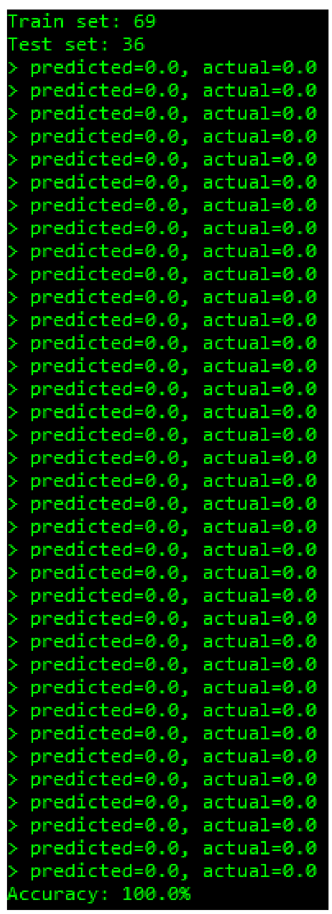
KNN & Decision Tree Analysis
Mohammed Jasam

1. Data Extraction

- Wrote a python script to scrape the articles from CNN.com and saved the data in the csv file.
- Removed all the regular expressions to get meaningful data.
- Created a data matrix in which the first row contains all the meaningful words from every article and the first column represents the article names.
- Ran a frequency analyzer to find out the frequency of those words in every article.

2. Feature Extraction

The above data matrix is a sparse matrix i.e, it has lots of zeroes rather than useful data. So we do a feature extraction to retain the meaningful attributes and delete the useless attributes. This increases the accuracy of the classifiers drastically. I have personally tested the accuracy of kNN classifier before feature extraction and it was almost always 100% but when I reduced the attributes from 8732 to 228 the accuracy stabilized and gave accurate predictions.



```
Train set: 69
Test set: 36
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
> predicted=0.0, actual=0.0
Accuracy: 100.0%
```

Figure 1: kNN Accuracy before Feature Extraction

3. Packages and Libraries

The program has been scripted in Python 3.6 and a varied list of modules were used to generate the results

- SciPy: Used for mathematical computations
- NumPy: Used for mathematical computations
- SciKit-Learn: Machine learning module for python
- Beautiful-Soup: Used to extract data from website
- Matplotlib: Used for visualizing the data
- Seaborn: Used for visualizing the data
- PyPDF2: Used to save the Decision Trees and to merge the files

4. Important Terminology

5-Fold Cross Validation:

In this method we try to get accurate predictions by dividing the dataset into 5 segments and testing the kNN classifier on each segment and by training it by the other 4 segments. This process is iterative and gives accurate results for each segment. We later take the average of each of these accuracy values and finally generate the average accuracy over the whole dataset.

Confusion Matrix:

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice versa). The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another).

F-measure:

In statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results, and r is the number of correct positive results divided by the number of positive results that should have been returned. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst at 0.

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

5. Classification using kNN

Classification (*generalization*) using an *instance-based* classifier can be a simple matter of locating the nearest neighbour in *instance space* and labelling the unknown instance with the same class label as that of the located (known) neighbour. This approach is often referred to as a *nearest neighbour classifier*. The downside of this simple approach is the lack of robustness that characterize the resulting classifiers. The high degree of local sensitivity makes *nearest neighbour classifiers* highly susceptible to noise in the training data.

More robust models can be achieved by locating *k*, where $k > 1$, neighbours and letting the majority vote decide the outcome of the class labelling. A higher value of *k* results in a smoother, less locally sensitive, function. The *nearest neighbour classifier* can be regarded as a special case of the more general *k-nearest neighbours classifier*, hereafter referred to as a *k*NN classifier.

Selecting the Value of K:

Each 'K' value gives varied results and each has a different error_rate, so by selecting least 'K' with minimum error_rate will give more accurate results. Below are the various graphs representing various Error Rate vs. K for the 5 segments in the dataset.

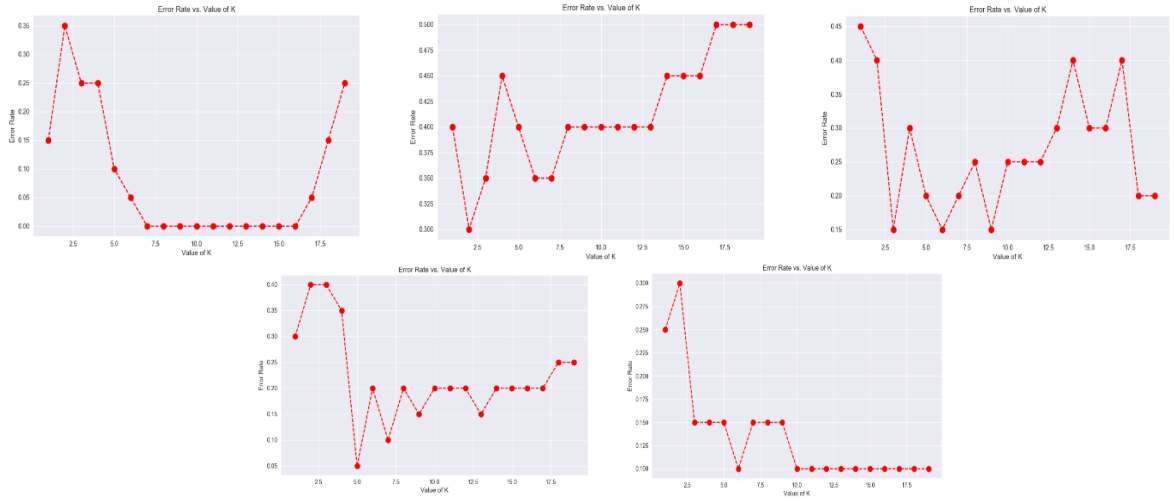


Figure 2: Error Rates vs. K values for 5 segments

By selecting the best values of 'K' with minimum error rate in all five segments of the data and taking the average of it gives me the best value of **K= 5** for the whole dataset.

Execution:

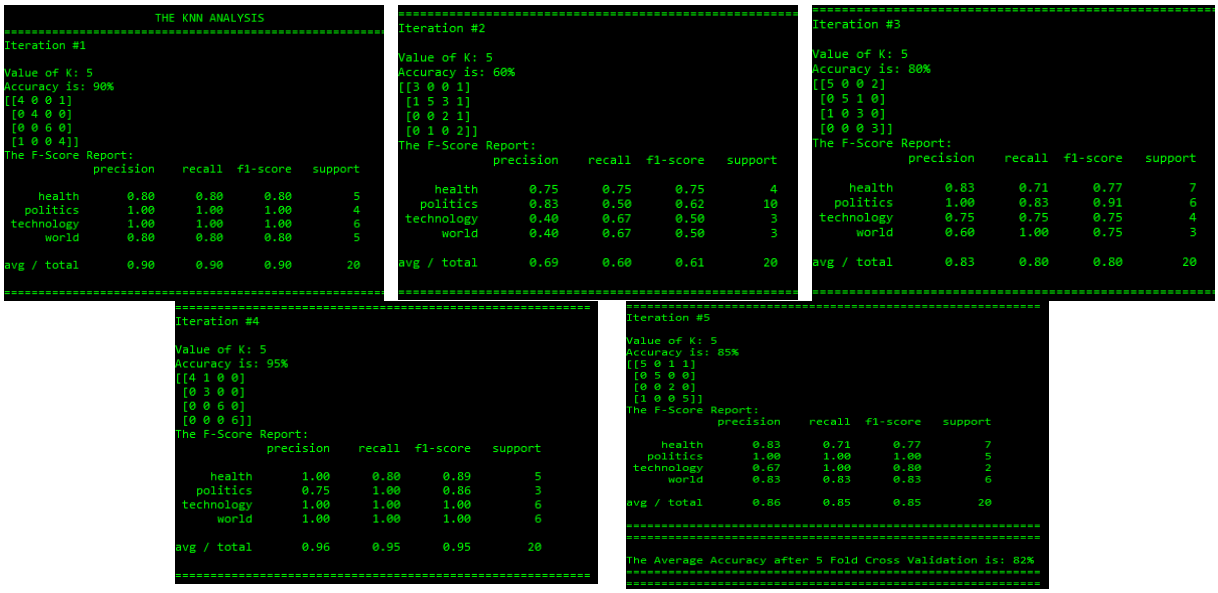


Figure 3: Accuracy. Classification reports including F Score for kNN Classifier

6. Classification using Decision Tree

Decision tree learning is a method commonly used in data mining. The goal is to create a model that predicts the value of a target variable based on several input variables. An example is shown in the diagram at right. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

Each element of the domain of the classification is called a class. A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature. The arcs coming from a node labeled with an input feature are labeled with each of the possible values of the target or output feature or the arc leads to a subordinate decision node on a different input feature. Each leaf of the tree is labeled with a class or a probability distribution over the classes.

A tree can be "learned" by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. See the examples illustrated in the figure for spaces that have and have not been partitioned using recursive partitioning, or recursive binary splitting.

Pre-Processing:

1. The data had to be segregated into predictors and targets.
2. Each column header has been saved into a list to act as predictors.
3. The whole first column is saved as target, which has to be tested.
4. 5-Fold Cross Validation was implemented and F-Score report was generated.

Execution:

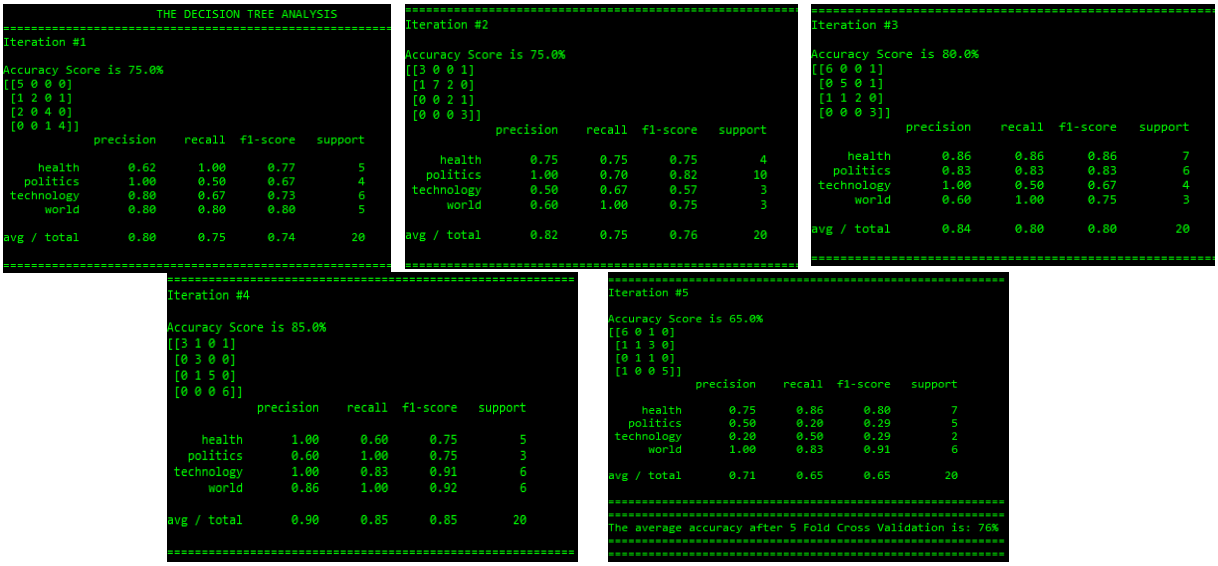


Figure 4: Accuracy, Classification report including F Score for Decision Tree Classifier

7. Visualization

The accuracy values of each classifier have been plotted on all 5 segments of the data. Seaborn and Matplotlib were used to visualize the data.

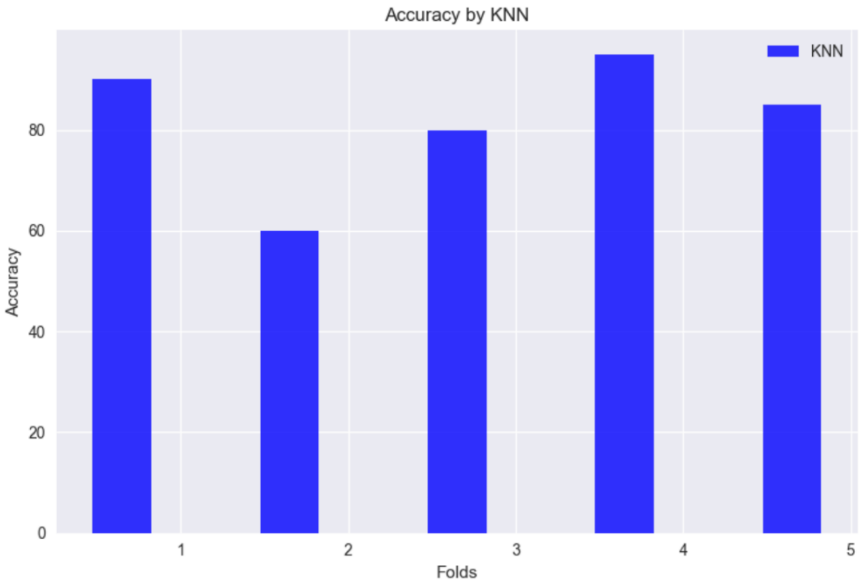


Figure 5: Accuracy Values of kNN Classifier

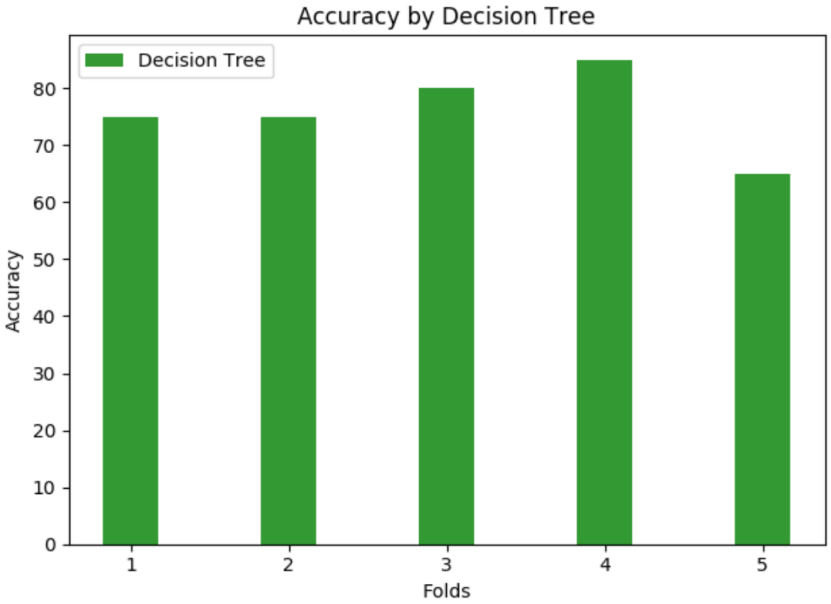


Figure 6: Accuracy Values of Decision Tree Classifier

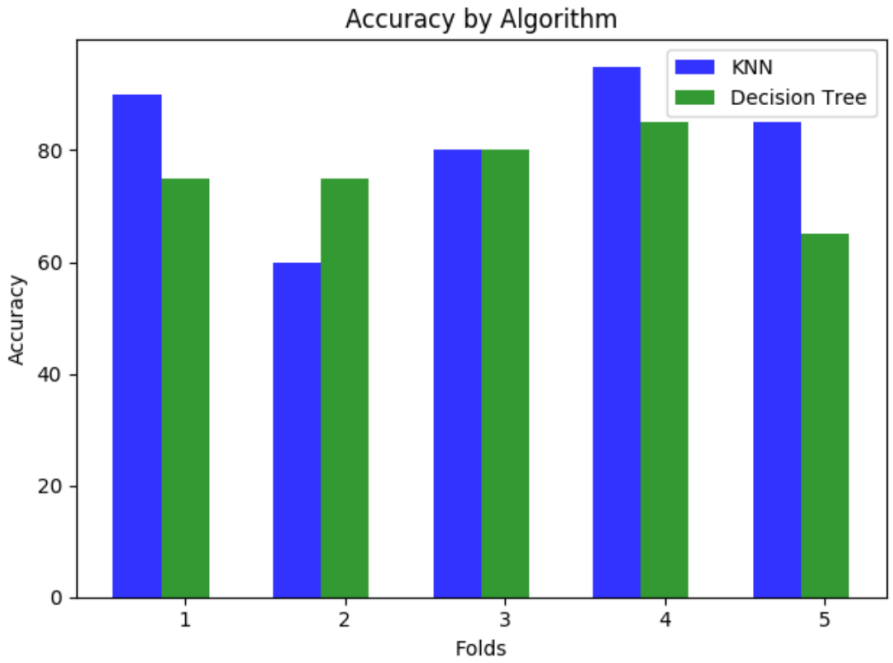


Figure 7: Comparison between kNN and Decision Tree Accuracy Values

8. Interpretation & Analysis

The results from both the kNN and Decision Tree Classifiers show the accuracy values generated by doing a 5-Fold Cross Validation. As mentioned above cross validation ensures better accuracy and gives better predictions.

kNN Classifier:

I have calculated the best value of k with minimum error rate, this came out to be **5**, after doing the cross validation I could generate accurate results on the overall data rather than just running the classifier on the whole dataset one single time. I have also performed an analysis on varying k values for each segment and have secured better accuracy than average k value. But the intention was fixing the k value before the execution the data segment in iterative manner.

Decision Tree Classifier:

Selecting the proper features gave me better accuracy rather than testing on the whole master data set, the results were inaccurate as mentioned in Section 2. After feature extraction I got better and accurate results.

Comparision:

By comparing the results generated from the classifiers and the graphs, I can say that the kNN classifier did better prediction than the Decision Tree Classifier on my data set.