# Applied Spatial and Temporal Data Analysis

Vijay K. Shah

## I. Data Retrieval

*Q 1. Find 100 CNN news articles online (try to find them in different categories, e.g., sports and finance). You need to find the new article by yourself. Pls ignore picture or other non-text data in the new article.*

In this section, I first describe how 100 CNN News links were retrieved followed by the discussion on data retrieval from each link. **Note:** I actually crawled 200 CNN News articles for this assignment.

### A. *100 CNN News Link Retrieval:*

**Step 1:** I downloaded the urls for all the articles in the webpage - http://www.cnn.com/US/archive/
**Step 2:** I stored them in a text file **links.txt** in home folder **helloProject**.
Step 1 and 2 are handled by script file **cnn_spider.py** stored in folder **helloProject/helloProject/Spiders/**

### B. *Data (Articles) Retrieval from all links:*

I created a script **pages_spider.py** to navigate through all urls (stored in links.txt) and download the content body using Scrappy request. Here, I used Scrapy selector.xpath('//p/text()') , selector.xpath('//h/text()') and selector.xpath('//span/text()') so that only useful texts are downloaded instead of entire html file. After that, I only consider alphabetical words with atleast 3 alphabets using a regrex [a-zA-Z]$\{3,\}$

**Note:** The script **pages_spider.py** can be found in folder "helloProject/helloProject/Spiders" Each downloaded article is stored as article- **article_%Article-headerName%.txt** in folder **Articles/**

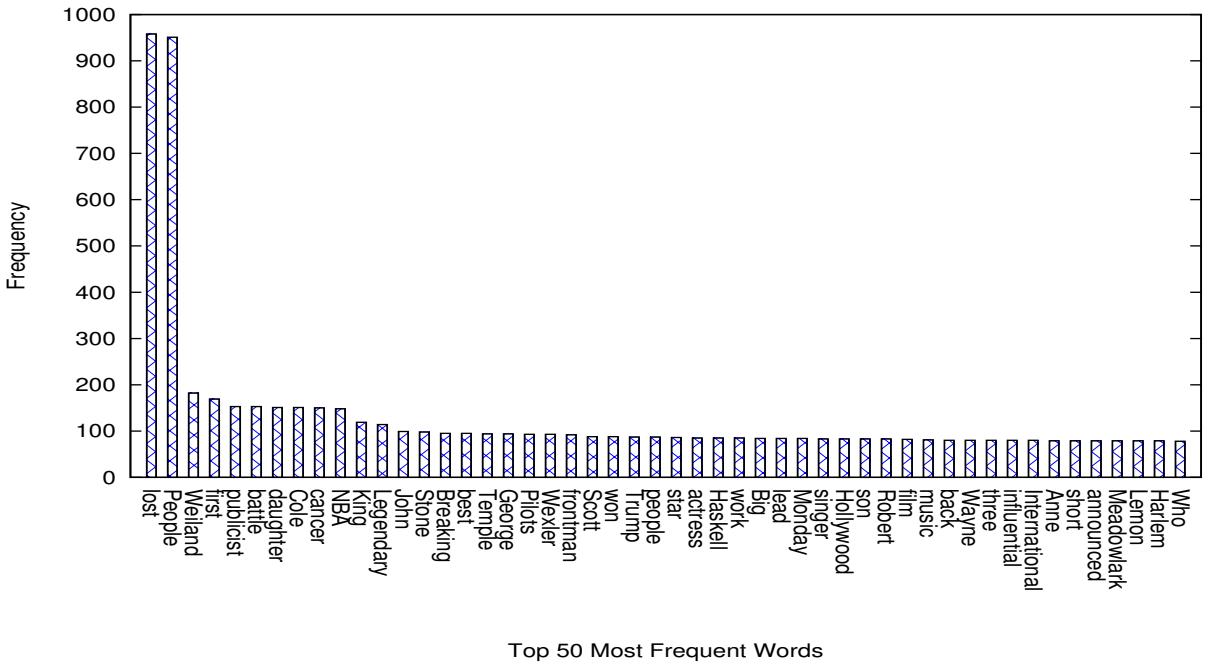


Fig. 1: Frequency vs Top K features (most frequent)

## II. Feature Selection

*Q 2. Convert them to data matrix (each row is an article and each column is a unique term).*

In this section, I present the details of forming a data matrix where each row is an article and each column is a unique word.

### A. *Creation of data matrix*

**Step 1:** I filtered out insignificant words from the data retrieved from each article as discussed in Homework 1.
**Step 2:** I sorted out the list of words in decreasing order of its frequency. (only until I get first top K frequent unique words).
**Step 3:** I created a data matrix of size (Number of Articles) X (Top K Most Frequent Words). Each cell contains the frequency of each unique word (Column) for following article (Row).

### B. *Addition of Category Column to the data matrix*

I have added a new column called *Category*, which represents the category of the article (row). The category list is as follows:

- **entertainment**
- **us**
- **health**
- **politics**
- **opinions**

## III. Classifier

*Q 3. Run kNN and Decision Tree with 5-folder cross validation. (label each cnn new article with the category)*

I used **sklearn** to run both kNN and Decision Tree with 5-folder cross validation. The schematic pseudocode for Decision Tree Classifier is given in Algorithm 1.

**Algorithm 1** DecisionTreeClassifier

```
1:  procedure DECISIONTREECLASSIFIER()
2:      df = pd.read_csv("dataMatrix.csv") Comment: Read the data matrix csv file
3:
4:      Comment: Convert non-numerical label "Category" to numerical label
5:      label_encoder = preprocessing.LabelEncoder()
6:      encoded_category = label_encoder.fit_transform(df["Category"])
7:
8:      Comment: call Decision Tree classifier
9:      tree_model = tree.DecisionTreeClassifier()
10:
11:     Comment: cross validation with 5 folds
12:     cv = KFold(n=len(df), n_folds=5) Number of folds is 5
13:
14:     for train_index, valid_index in cv do
15:         train = df.loc[train_index] Comment: Extract train data with cv indices
16:         valid = df.loc[valid_index] Comment: Extract valid data with cv indices
17:
18:         Comment: training dataset (4 folds out of 5 folds)
19:         ytrain = label_encoder.fit_transform(train["Category"])
20:         Xtrain = train[list(train.columns[1:50])]
21:
22:          Comment: test dataset (1 fold out of 5 folds)
23:         ytest = label_encoder.fit_transform(valid["Category"])
24:         Xtest = valid[list(valid.columns[1:50])]
25:
26:         Comment: Predict the test dataset with the decision tree classifier
27:         y_pred = tree_model.predict(Xtest)
28:
29:         Comment: Accuracy and F-Score
30:         ac_score = accuracy_score(ytest, y_pred)
31:         fscore = f1_score(ytest, y_pred, average = "weighted")
```

**Note:** For kNN classifier, only one change has to be made, which is as follows: In Step 9, instead of DecisionTreeClassifier, KNeighborsClassifier has to be called.

**knnClassifier = neighbors.KNeighborsClassifier(n_neighbors=5)**

## IV. Analysis

*Q 4. Evaluate classification results with accuracy, F-measure*

In this section, I first define the metrics (i) Accuracy and (ii) F-measure following the evaluation of classification results.

### A. Accuracy

Accuracy is defined as the number of correct predictions made divided by the total number of predictions made, multiplied by 100. Let $N_c$ be the number of correct predictions made and $N_t$ be the total number of predictions made, then, the accuracy is calculated as

$$Accuracy = \frac{N_c}{N_t} \times 100 \tag{1}$$

### B. F-measure

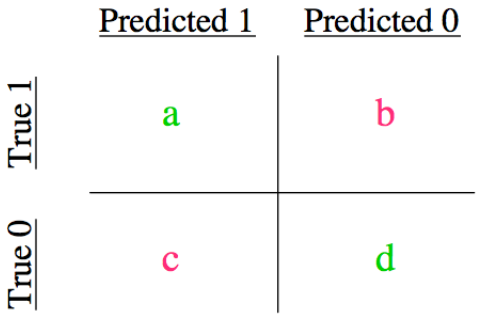


Fig. 2: Precision/Recall

To understand F-measure, it is important to define two parameters: (i) Precision and (ii) Recall.

*1)* ***Precision****:* Precision is defined as the number of True Positives divided by the number of True Positives and False Positives. In other words, it is the number of positive predictions divided by number of positive class values predicted.

Taking Fig. 2 as a reference, the precision can be calculated as

$$Precision = \frac{a}{a+c} \tag{2}$$

*2)* ***Recall****:* Recall is defined as the number of True Positives divided by number of True Positives and number of False Negatives. In other words, it is the number of positive predictions divided by the number of positive class values in the test data.

Taking Fig. 2 as a reference, the recall can be calculated as

$$Recall = \frac{a}{a+b} \tag{3}$$

F-Score is the harmonic mean of precision and recall, calculated as follows:

$$F - Score = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

### C. Evaluation of Decision Tree Classifier

In this section, we evaluate the performance of decision tree classifier in terms of Accuracy and F-Score.

Fig. 3(a) depicts the accuracy score achieved in each fold of cross validation and the average accuracy score when predictions are made over unshuffled training set (of data matrix). Whereas Fig. 3(b) shows the accuracy score of each cross validation fold and average score when predictions are made over shuffled training datasets. It is evident that shuffled training data set give higher accuracy in each fold compared to that of unshuffled training set. This is because in shuffled data set, the category labels (which is being predicted) are more uniformly distributed whereas in unshuffled data set, they are clustered together. Hence, when predictions are done using cross validation technique in unshuffled data set, the training data sets may only be trained for a fixed set of labels (due to clustered data set), hence leading to poor accuracy score when predicted on different chunk of data set (which may have different clustered labels).

Similar is the case with F-score values calculated over unshuffled data set and shuffled data set as shown in Fig. 4(a) and 4(b) respectively.

As shown in Fig. 5, the average accuracy and F-score achieved for different number of top K features (most frequent words) computed for shuffled data set is more or less same. It shows that both metrics (i) Accuracy and (ii) F-score works well for given data set.

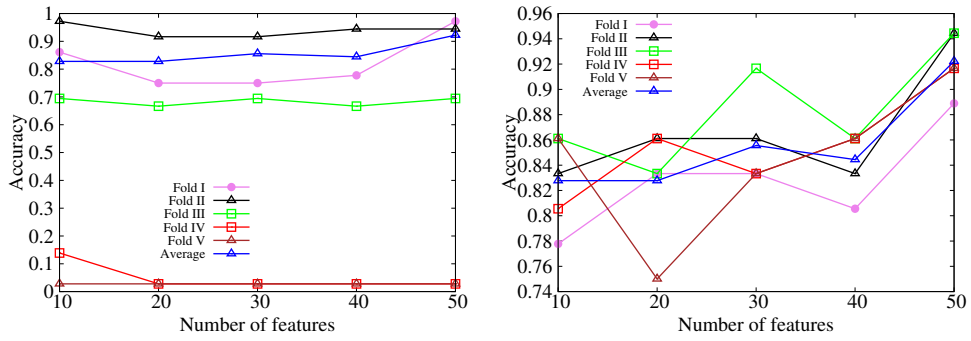


Fig. 3: Decision Tree Classifier: Accuracy vs (a. Unshuffled Data Matrix b. Shuffled Data Matrix)

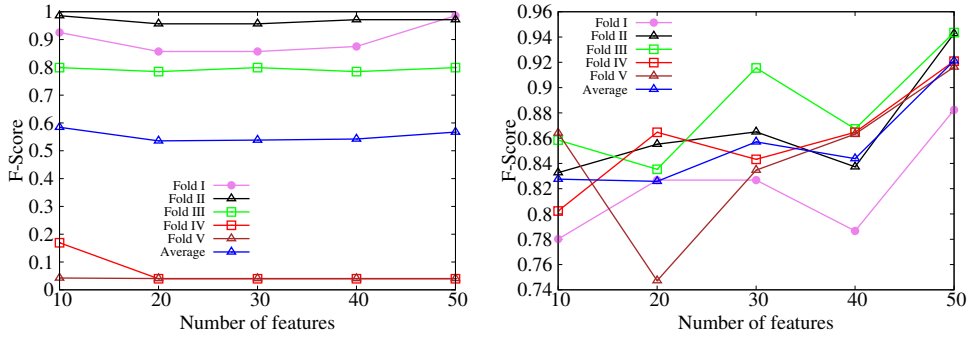


Fig. 4: Decision Tree Classifier: F-Score vs (a. Unshuffled Data Matrix b. Shuffled Data Matrix)

### *D. Evaluation of kNN*

In this section, we evaluate K Neighbors classifier in terms of accuracy and F-score.

As shown in Fig. 6(a) and 6(b), the accuracy score achieved in shuffled training data set is higher than that of unshuffled training data set. As aforementioned, the reason is clustered labels in unshuffled data set (data matrix). Hence, the training is performed on a fixed set of clustered labels, whereas the testing is done against another chunk of dataset (which may have different set of clustered labels).

Similar is the case with F-score values calculated over unshuffled data set and shuffled data set as shown in Fig. 7(a) and 7(b) respectively.

Fig 8(a) shows that the average accuracy achieved by kNN classifier is best when number of neighbors is low (such as 5). However, when the number of neighbors increase, the accuracy dips. It is because with increase in number of neighbors, the far off neighbors with uncorrelated labels are also taken into consideration.

As shown in Fig. 8(b), the average accuracy score achieved is almost same as that computed using F-Score metric over shuffled data set for the kNN classifier. Hence, it shows that both metrics work well for given data set as well.

### *E. Decision Tree vs kNN*

In this section, we present the evaluation results for the performance of Decision Tree and kNN classifiers for shuffled data set in terms of both metrics (i) Accuracy and (ii) F-Score

As shown in Fig. 9(a) and 9(b), the decision tree classifier works better than kNN classifier. It is obvious because decision tree is robust, takes the weights of each feature in consideration to form a well-defined tree based on features list. Whereas, kNN is based on majority votes of k nearest neighbors. It is not able to capture the unequal weight of each feature.

## V. Conclusion

From above analysis, it is evident that Decision Tree classifier works better than kNN classifier for a given data set. Also, it is clear that shuffled data set is preferred compared to unshuffled data set for training as it removes any clustered labels (polarized data sets). Moreover, both the metrics (i) Accuracy and (ii) F-Score works well for the given data set irrespective of number of features considered.

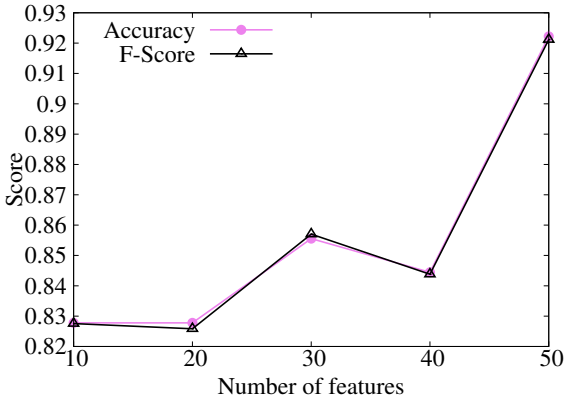


Fig. 5: Decision Tree Classifier: Score vs Top K Features (Shuffled data set)

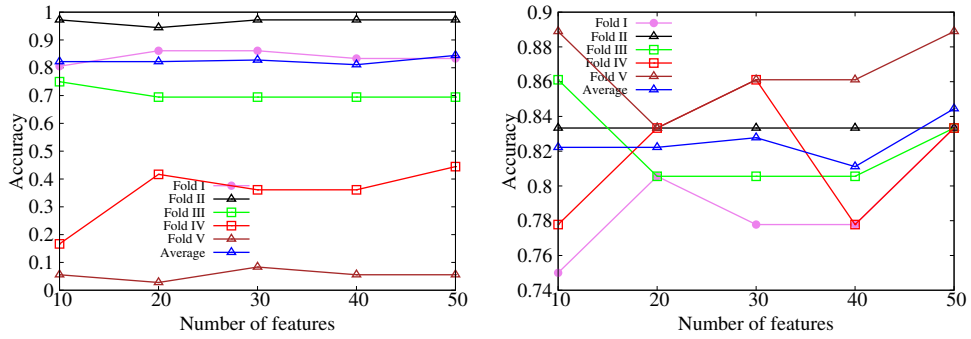


Fig. 6: kNN Classifier: Accuracy vs (a. Unshuffled Data Matrix b. Shuffled Data Matrix)

Lastly, the achieved score (using any metric) improves with higher number of features in both the classifiers which is quite intuitive.

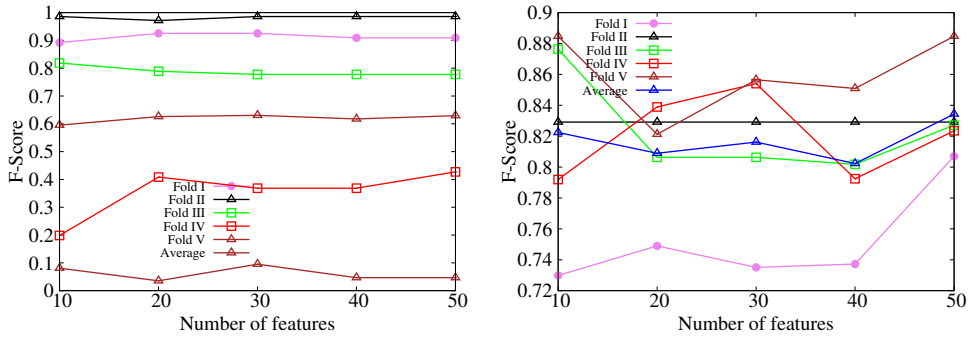


Fig. 7: kNN Classifier: F-Score vs (a. Unshuffled Data Matrix b. Shuffled Data Matrix)

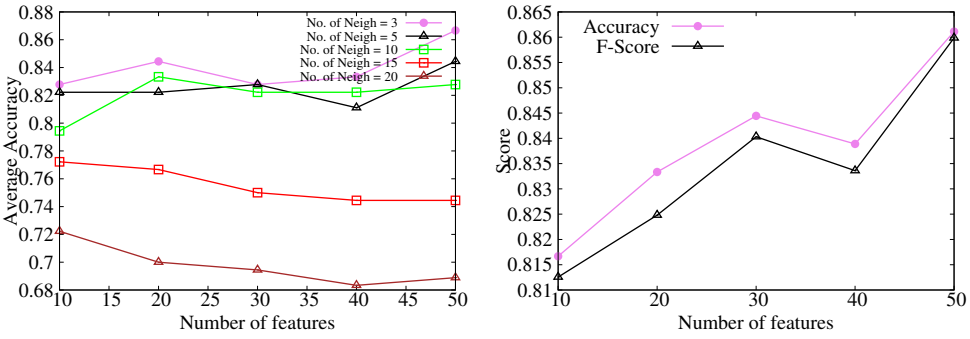


Fig. 8: kNN Classifier: a. Avg. Accuracy vs Number of Neighbors b. Score vs Top K Features (Shuffled data set)

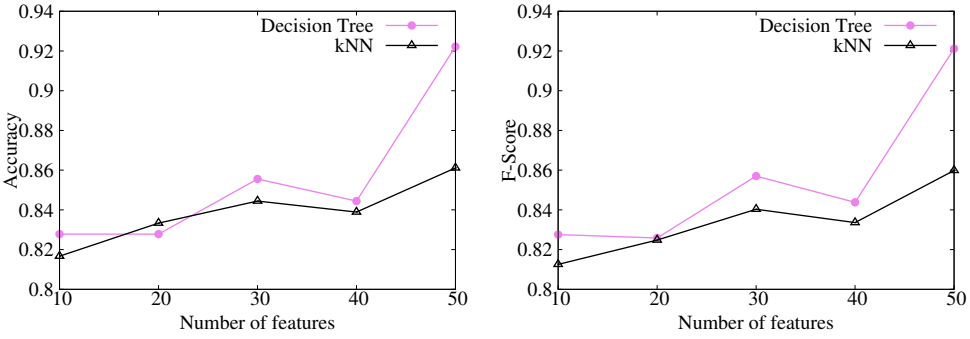


Fig. 9: Decision Tree vs kNN classifier: (a. Accuracy b. F-Score) vs Top K features