`

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Applied Machine Learning

ELEC 8900-76-R-2021F

Date: Nov 29th, 2021

"Feature Selection_09"

(FS-09)

Instructor:

Prof. Roozbeh Razavi Far

Submitted by:

Kenil Doshi **ID: 110031636**

Meghal Kumar Darji **ID: 110041749**

University of Windsor

ON, CANADA

ABSTRACT:

Artificial Intelligence is the most discussed topic in the industry at the moment. Compute scientists all over the world are working on research and development of Machine Learning algorithms for application in various domains. Machine learning is one of the subsets of Artificial Intelligence. Machine learning (ML) attempts to understand learn algorithms and mathematical models used in software systems from patterns observed in data. It tries to develop an algorithmic structure from training data to use it later on future inputs for a specific purpose. Machine learning applications can be seen in many fields such as Computer Vision, Financial Forecasting, Robotics, and many more. Machine learning is also being used in making predictions and data analysis. Machine learning has several learning approaches such as supervised, unsupervised, and reinforcement learning. Machine learning has a primary goal of optimization and to generalize the learning algorithm based on its training to get possible result on test data. The machine learning process starts with learning task or problem. Here, the dataset under consideration is used for training, where the algorithm tries to learn the unknown target function, to make predictions on the test dataset. This project focuses on 3 feature selection methods:

i. Unsupervised Discriminative Feature Selection [UDFS]
ii. Local Learning-based Clustering Feature Selection [LLCFS]
iii. Correlation-based Feature Selection [CFS]

To perform classification the following algorithms are used:

i. Naive Bayes Classifier [NBC]
ii. Support Vector Machines [SVM]
iii. Decision Trees [DT]
iv. Multi-Layer Perceptron [MLP].

The main aim of this project is to try the different combinations of feature selection and classification methods to find the best results on the given data.

Keywords

Classifier, Feature Selection, Parameter Tuning, Decision Tree, Support Vector machines, Hyper Parameter, Multi-layer Perceptron, Naive Bayes classifier, Machine Learning, F1 Score, confusion Matrix, MATLAB, python

Development Software

MATLAB R2019b & MATLAB R2020a

Python, Jupyter Notebook

INTRODUCTION

Machine learning algorithms try to learn patterns from the available data to make predictions on similar but new data. It generally used when dealing with large data sets that has some kind of pattern which cannot be explained by mathematical equations. Machine learning has two main approaches: supervised and unsupervised learning. Supervised learning involves learning from data for which the ground truth labels are known unlike unsupervised learning unsupervised learning where no information is available about the ground truth labels.. Supervised learning has 2 main categories of learning which are classification and regression. Classification simply classifies given data to give discrete responses. On the other hands, regression predicts continuous values.

Feature selection is also known as variable or attribute selection is in demand to the due formation of huge datasets and better machine learning techniques. Feature selection focuses on selecting a method automatic in nature for best possible attribute selection aligned of selecting relevant to data and predictive modeling. The benefits of feature selection are that it reduces over fitting and training time while providing an opportunity to improve model performance.

Feature Selection Methods:

1. **Unsupervised Discriminative Feature Selection [UDFS]**
   UDFS aims to select the most discriminative features for data representation, where manifold structure is considered making it different from the other feature selection algorithms. It is useful for feature selection in data without labels. UDFS algorithm selects the most discriminative feature subset from the whole feature set in batch mode. The UDFS algorithm aims to simultaneously exploiting discriminative information and feature correlations. Use one-step approach to select the most discriminative approach for data representation. UDFS analyzes features jointly and local structure of data distribution.

2. **Local Learning-based Clustering Feature Selection [LLCFS]**
   The key idea here is to break an arbitrarily complex nonlinear problem into a set of locally linear sets through local learning method leading to learning feature relevance globally within the large margin framework. This Feature Selection model has a logarithmic space complexity with respect to a number of features.

3. **Correlation-based Feature Selection [CFS]**
   The Correlation based Feature Selection (CFS) algorithm selects features for classification tasks based on the correlation between features and the class label. A feature is considered useful if it is correlated with or predictive of the output label and at the same time meekly correlated with other features in the dataset. CFS algorithm greedily adds feature to a feature subset and then evaluates the merit of this subset.

Finally the feature subset with the highest merit score is selected when no improvements in the merit score are observed after 5 consecutive iterations. Merit of feature subset is calculated using the following formula

$$\text{Merit}_{S_k} = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}.$$

CFS Optimization Problem:

$$\text{CFS} = \max_{S_k} \left[\frac{r_{cf_1} + r_{cf_2} + \cdots + r_{cf_k}}{\sqrt{k + 2(r_{f_1 f_2} + \cdots + r_{f_i f_j} + \cdots + r_{f_k f_1})}} \right].$$

Where k is the number of features in the subset;
r_{cf} is the average value of all feature-classification correlations
r_{ff} is the average value of all feature-feature correlations

Classification Methods:

a. Naive Bayes Classifier [NBC]

The Naive Bayes Classifier is based on Bayes theorem with the assumption that features selected are independent. There are three main types of NBC's: Multinomial Naive Bayes, Bernoulli Naive Bayes, and Gaussian Naive Bayes. This project uses the Gaussian Naïve Bayes Classifier for all classification tasks.

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})} \quad \text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}}\, p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k).$$

b. Support Vector Machines [SVM]

Support Vector Machines are non-probabilistic binary linear classifiers and are supervised learning models. It works by dividing well-defined wide gap points in space with separate categories.

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0,1-y_i(w\cdot x_i-b)\right)\right]+\lambda\|w\|^2.$$

c. Decision Trees [DT]

Decision Trees are used to display an algorithm with condition control statements and are decision support tools for best possible outcomes with a tree-like model of decisions. It has a structured algorithm with an easy understanding of data classification flow shows the possible consequences. Uses a predictive model on observations of elements to find target elements value

d. Multi-Layer Perceptron [MLP]

Multi-Layer Perceptron (MLP) is a supplement of feed forward neural network. It consists of three types of layers—the input layer, output layer and hidden layer, as shown in Fig. 1. The input layer receives the input signal to be processed. The required task such as prediction and classification is performed by the output layer. An arbitrary number of hidden layers that are placed in between the input and output layer are the true computational engine of the MLP. Similar to a feed forward network in a MLP the data flows in the forward direction from input to output layer.

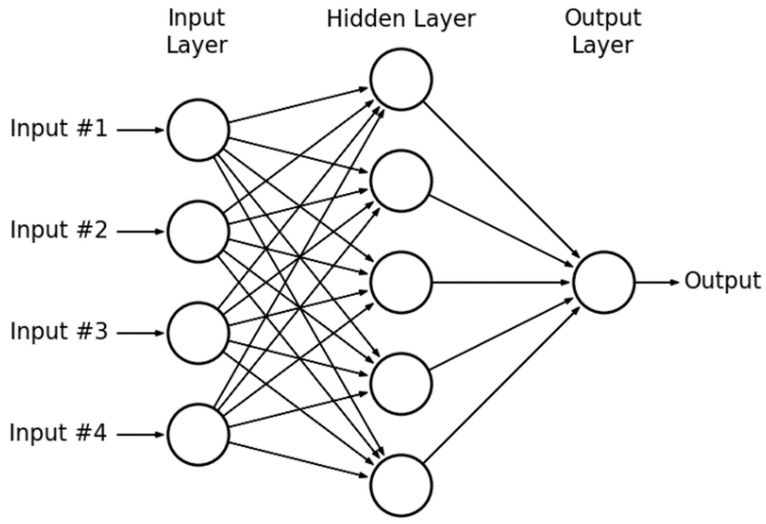


Figure 1: Multi-Layer Perceptron

Implementation Procedure:

- For this project we primarily use Python 3 programming language in the Jupyter Notebook along with MATLAB for one of the feature selection methods. For certain computationally expensive tasks Google Colaboratory (Google Colab) was used for faster execution.
- Before using the given datasets for training and prediction, datasets were examined for missing values. In one the datasets missing value were found which required proper handling to avoid errors and exceptions.
- Next, the feature values and the class labels are read and separated from the provided data files. This training data is then used to obtain baseline classification accuracy and f1 scores with 10-fold stratified cross validation for all the classification algorithms mentioned above.
- After getting the baseline score, the next step in pipe line is to compute the feature rankings using the feature selection methods individually. Several features are then chosen based on the obtained rankings to form a new feature set. This reduced feature set is then used to again perform classification tasks using the same algorithms.
- All training and testing done in this project is done with 10-fold stratified cross-validation to ensure that class balance in the folds is maintained in-case of an imbalanced dataset.
- The accuracy and f1 scores for different classification algorithms are compared to analyze the improvements and identify the combination of feature selection and classification methods that provides improved results for the given datasets.

Key MATLAB and 'python' files used for Program:

The project is completed using the following code files:

1. `baseline.ipynb`: Jupyter notebook code file to obtain baseline classification accuracy and f1 scores on all datasets using the mentioned classification algorithms.

2. `CFS.ipynb`: Jupyter notebook code file to perform Correlation Based Classification.

3.`Classification_after_udfs.ipynb`: Jupyter notebook code file to perform classification using the feature rankings obtained from UDFS method.

4. `UDFS.ipynb`: Jupyter notebook code file to perform UDFS on the given datasets.

5. `dataset_sampling.ipynb`: Jupyter notebook code file to sample data from the given datasets to reduce dataset size. (To successfully perform computationally heavy feature selection methods).

6. `MLP.ipynb`: Jupyter notebook code file to perform classification and obtain baseline measures using MLP on all data.

7. `MLP_UDFS.ipynb`: Jupyter notebook code file to perform classification after feature selection by UDFS method.

8. `FS09_demo.ipynb`: Jupyter notebook code file for class demonstration and to obtain classification results after feature selection for all methods and classifier algorithms.

9. `K_demo.m`: MATLAB code file to obtain feature rankings using LLCFS.

Key Parameters for tuning the Classifiers:

All the classifier methods in this project are implemented using Scikit-learn library in python 3.

1) Support Vector Machines: `SVC(kernel='rbf', gamma=0.5,C=10000)`

SVM's **kernel** shows which type of function we use and according to that our model decide it to linear or nonlinear. **Degree** gives information for the nonlinear degree of the function, and **max iteration** gives info about spoke for higher accurate model.

2) Decision Tree: `DecisionTreeClassifier()`

Here, no tuning parameters are used for Decision Tree Classifier.

3) Naïve Bayes Classifier: `GaussianNB()`

In our case, we do not use any tuning parameter. But the Gaussian Naive bayes performs poorly when compared to other methods in this project.

4) Multilayer Perceptron:

```
MLPClassifier(hidden_layer_sizes=(10,10,10), max_iter=1000)
```

The MLP used here consists of three hidden layers with 10 neurons in each hidden layer. Furthermore, the MLP Iterated for a maximum of 1000 times during training before moving in to the next epoch.

Feature Selection Algorithms:

1) UDFS (Unsupervised Discriminative Feature Selection.)

for i = 1 to n do

1 Bi = (˜XTi˜Xi + λI)−1
2 Mi = SiHk+1BiHk+1STi ;
3 M = X_ni=1MiXT ;
4 Set t = 0 and initialize D0 ∈ Rd×d as an identity matrix;
5 Repeat
6 Pt = M + γDt;
7 Wt = [p1, ..., pc] where p1, ..., pc are the eigenvectors of Pt corresponding to the first c smallest eigenvalues;
8 Update the diagonal matrix Dt+1 asDt+1 =⎡⎢⎣12_w1t _2...12_wdt_2⎤⎥⎦;
9 t = t + 1;
10 until Convergence;
11 Sort each feature fi |di=1 according to__wit__2indescending order and select the top ranked ones.

2) LLCFS (Feature selection for local learning-based clustering algorithm.)

input: X ¼fxign i¼1, size of the neighborhood k, trade-off parameter (Beta)
output: Y; _
1 Initialize _l ¼ 1d , for l ¼ 1; . . . ; d;
2 while not converge do
3 Find k-mutual neighbors for fxign i¼1, using the metric defined);
4. Construct the matrix M in with _i given ,
5. then solve the problem to obtain Y;
6. Compute wc i ; 8i; c and update it for next epoch);
7. End

3) CFS (Correlation Feature Selection)

A correlation-based approach towards hypothesis focused on good feature sets containing features that are high correlated with the class, yet being uncorrelated with each other

CFS is an algorithm using a feature evaluation formula based on ideas from test theory that provides an operational definition of this hypothesis with an appropriate correlation measure and a heuristic search strategy CFS executes faster than other wrapper algorithms Merit of feature subset

On the base of following merits it find the correlation between the set and returns the feature which highly correlate with the dataset.

$$\text{Merit}_{S_k} = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}.$$

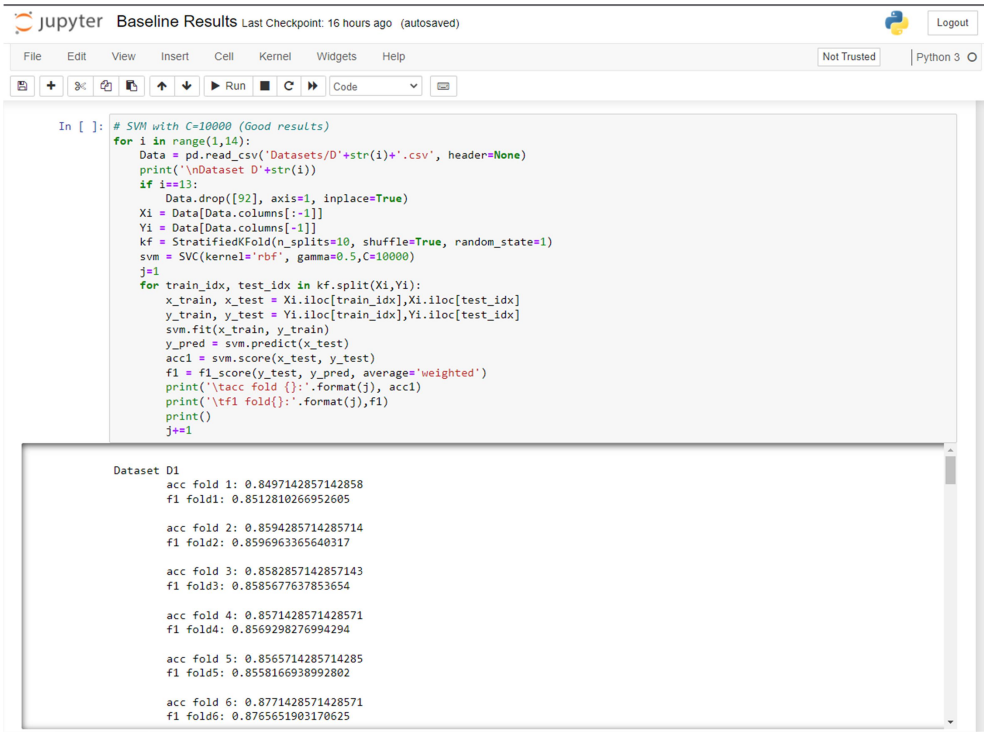
❖ Common Algorithm of Classification:

Steps:

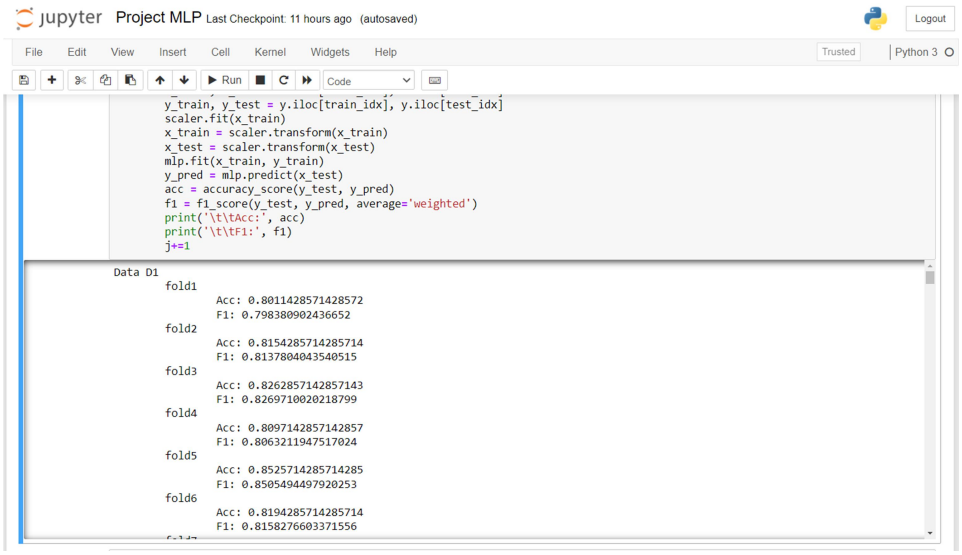
1. Discretize a copy of training set using Fayyas and Irani's method.
2. Search the feature space using either 1 of the 3 searching scheme: forward, backward, or best first
3. Evaluate the merit of a feature using Merit(Sk) equation .
4. Use the features subset with the highest merit to reduce the dimensionality of the training and test set.
5. Perform training and testing for the machine learning task using the reduced data sets.

Sample Results:

Accuracy and F1 Score with SVM classifier:



Accuracy and F1 Score with SVM classifier:



Classification accuracy and f1 scores after feature selection using UDFS:

jupyter Post UDFS Classification Last Checkpoint: Last Saturday at 11:36 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [81]:
```
# Decision Trees
n_features_t = 25
Dt = DecisionTreeClassifier()

d1 = pd.read_csv('Datasets/D1.csv',header=None)
x1 = d1[d1.columns[:-1]]
y1 = d1[d1.columns[-1]]
with open('udfs_results/D1.txt') as file:
    data = file.read()
L = data.replace('\n','')[1:-1].split(' ')
sel = [int(l) for l in L if l != '' ]

X_selected = x1.iloc[:,sel[-n_features_dt:]]
kf = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
j = 1
for train_idx, test_idx in skf.split(X_selected,y1):
    print('fold {}'.format(j))
    x_train, x_test = X_selected.iloc[train_idx],X_selected.iloc[test_idx]
    y_train, y_test = y1.iloc[train_idx],y1.iloc[test_idx]
    Dt.fit(x_train, y_train)
    y_pred = Dt.predict(x_test)
    f1 = f1_score(y_test,y_pred, average='weighted')
    acc = accuracy_score(y_test, y_pred)
    print('\tAcc:',acc)
    print('\tF1:',f1)
    j+=1
```

```
fold 1
        Acc: 0.96
        F1: 0.9602978013216178
fold 2
        Acc: 0.9537142857142857
        F1: 0.9539376327791682
fold 3
        Acc: 0.9531428571428572
        F1: 0.9531066059542204
fold 4
        Acc: 0.9554285714285714
        F1: 0.9550824659551795
fold 5
        Acc: 0.9622857142857143
        F1: 0.9620049634589707
fold 6
        Acc: 0.956
        F1: 0.9556782532096203
fold 7
        Acc: 0.9577142857142857
        F1: 0.9573604089130001
fold 8
        Acc: 0.9497142857142857
        F1: 0.949787009813855
fold 9
        Acc: 0.9628571428571429
        F1: 0.9629490062359083
fold 10
        Acc: 0.9594285714285714
        F1: 0.9594270727261591
```

Baseline Accuracy and F1 Score for NB Classifier:

jupyter Baseline Results Last Checkpoint: 4 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

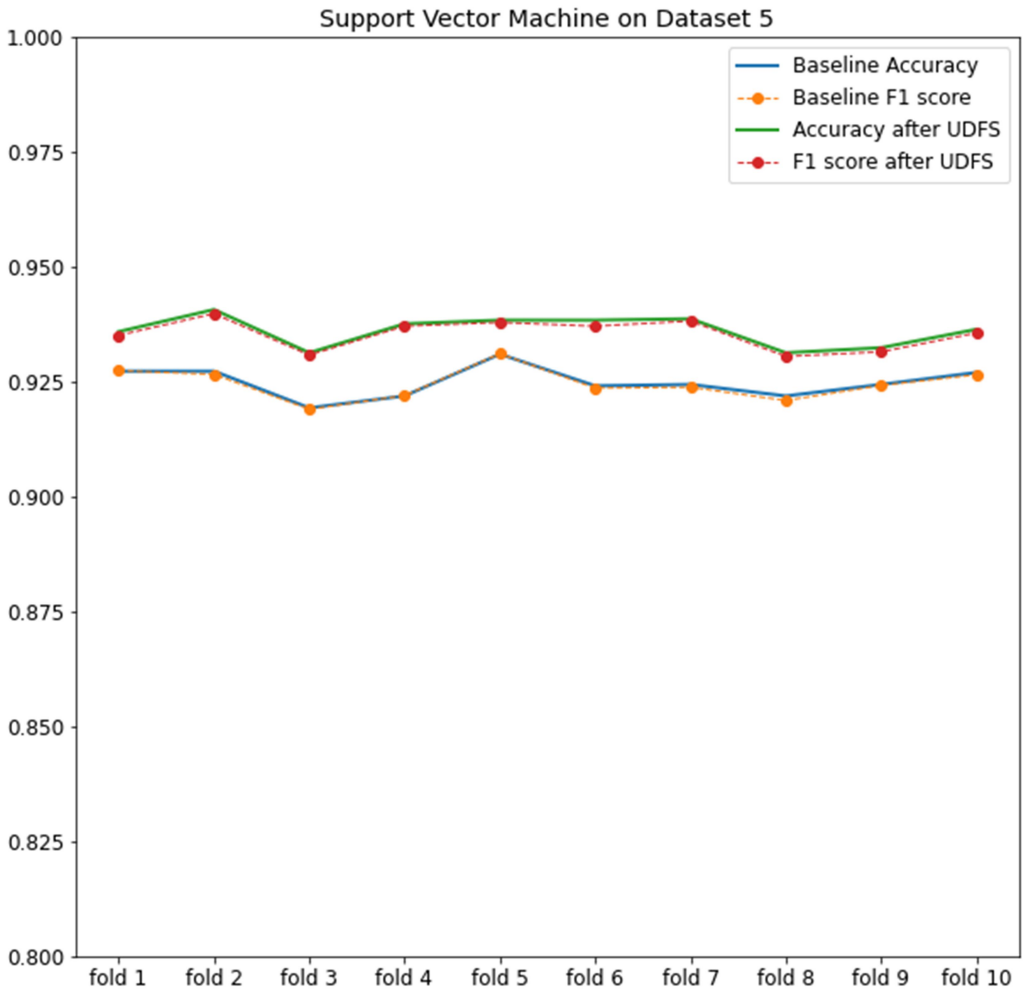
Run Markdown

In [*]:
```
# separate classifier for all datasets
for i in range(1,14):
    filename = 'Datasets/D'+str(i)+'.csv'
    Data = pd.read_csv(filename, header=None)
    if i==13:
        Data.drop([92], axis=1, inplace=True)
    Xi = Data[Data.columns[:-1]]
    Yi = Data[Data.columns[-1]]

    clf = GaussianNB()

    kf = StratifiedKFold(n_splits=10, shuffle=True)
    print("Dataset D{} split into folds. Training now...".format(i))
    j = 1
    for train_idx, test_idx in kf.split(Xi, Yi):
        x_train, x_test = Xi.iloc[train_idx],Xi.iloc[test_idx]
        y_train, y_test = Yi.iloc[train_idx],Yi.iloc[test_idx]

        clf.fit(x_train, y_train)
        y_pred = clf.predict(x_test)
        acc = accuracy_score(y_test, y_pred)
        f1 = f1_score(y_test, y_pred, average='weighted')
        print("\tAccuracy(fold {}):".format(j),acc)
        print("\tf1 (fold{})".format(j), f1)
        j+=1
    print('\n')
```

```
Dataset D1 split into folds. Training now...
        Accuracy(fold 1): 0.5365714285714286
        f1 (fold1) 0.5181082317881291
        Accuracy(fold 2): 0.5194285714285715
        f1 (fold2) 0.5004830612736968
        Accuracy(fold 3): 0.5451428571428572
        f1 (fold3) 0.5320135158905793
        Accuracy(fold 4): 0.5234285714285715
        f1 (fold4) 0.503991770427362
        Accuracy(fold 5): 0.552
        f1 (fold5) 0.5347707618322307
        Accuracy(fold 6): 0.5405714285714286
        f1 (fold6) 0.5249557844058634
        Accuracy(fold 7): 0.5217142857142857
        f1 (fold7) 0.5001341840122281
        Accuracy(fold 8): 0.5217142857142857
        f1 (fold8) 0.505350146795752
        Accuracy(fold 9): 0.5194285714285715
        f1 (fold9) 0.4979122703724286
        Accuracy(fold 10): 0.5068571428571429
        f1 (fold10) 0.4850950181012904


Dataset D2 split into folds. Training now...
```

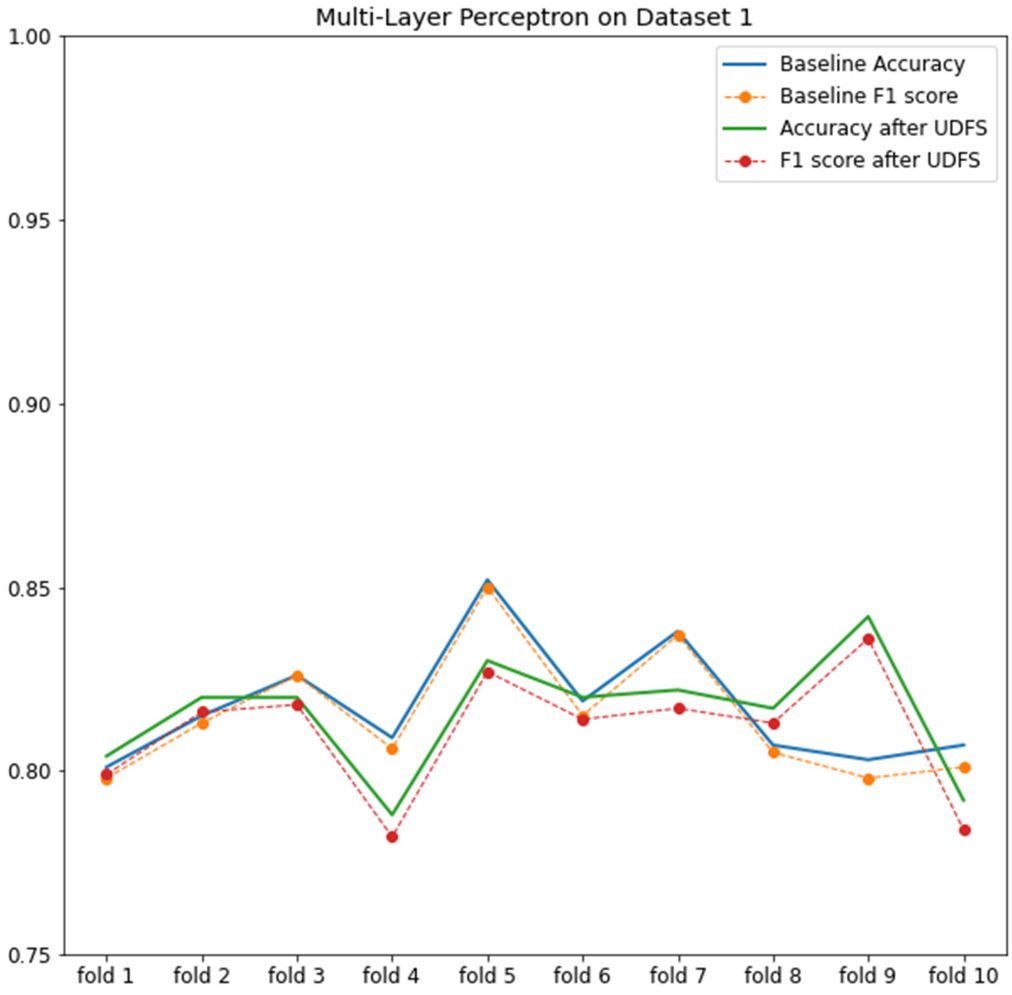
Visualized accuracy and f1 score measures:

The graph below shows the accuracy and f1 score measures on dataset D5 before and after feature selection. It can be seen that performance improves after feature selection.



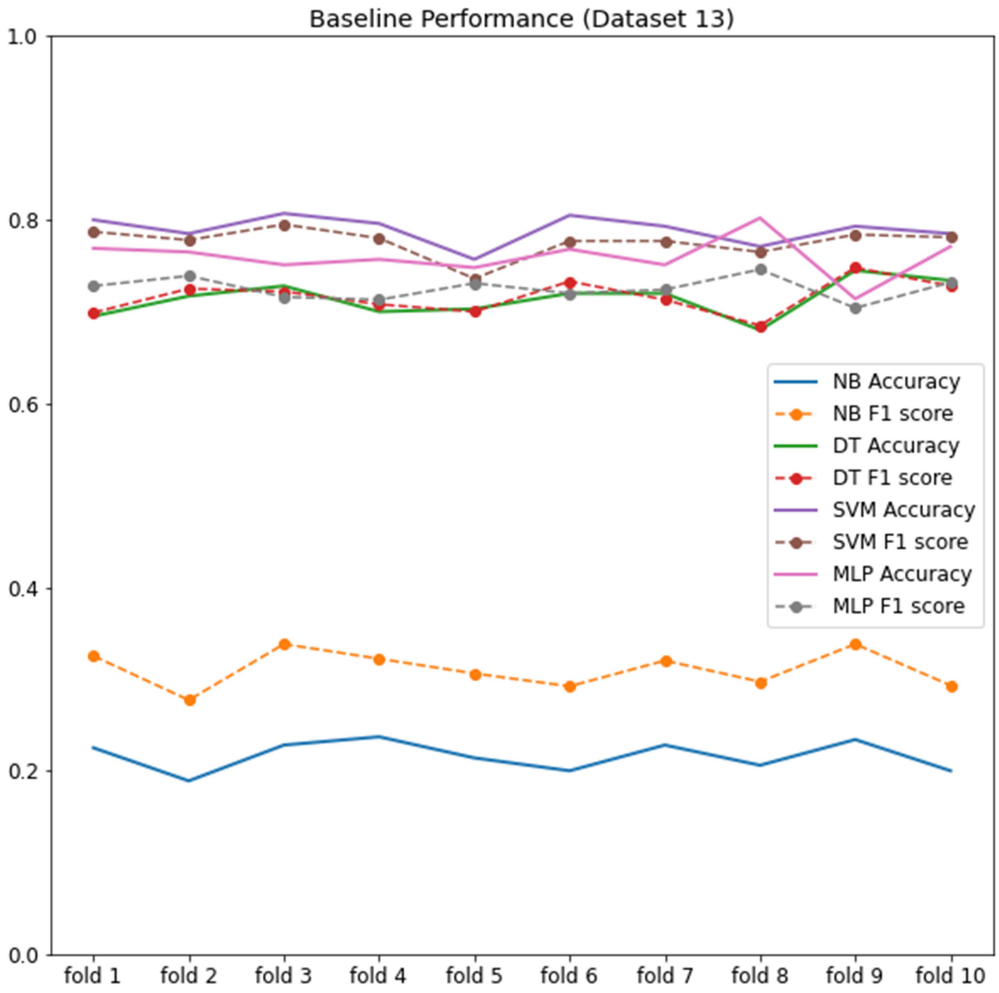
Feature_Selection-09

The graph below shows the accuracy and f1 score measures for dataset D1 and MLP classifier using UDFS.

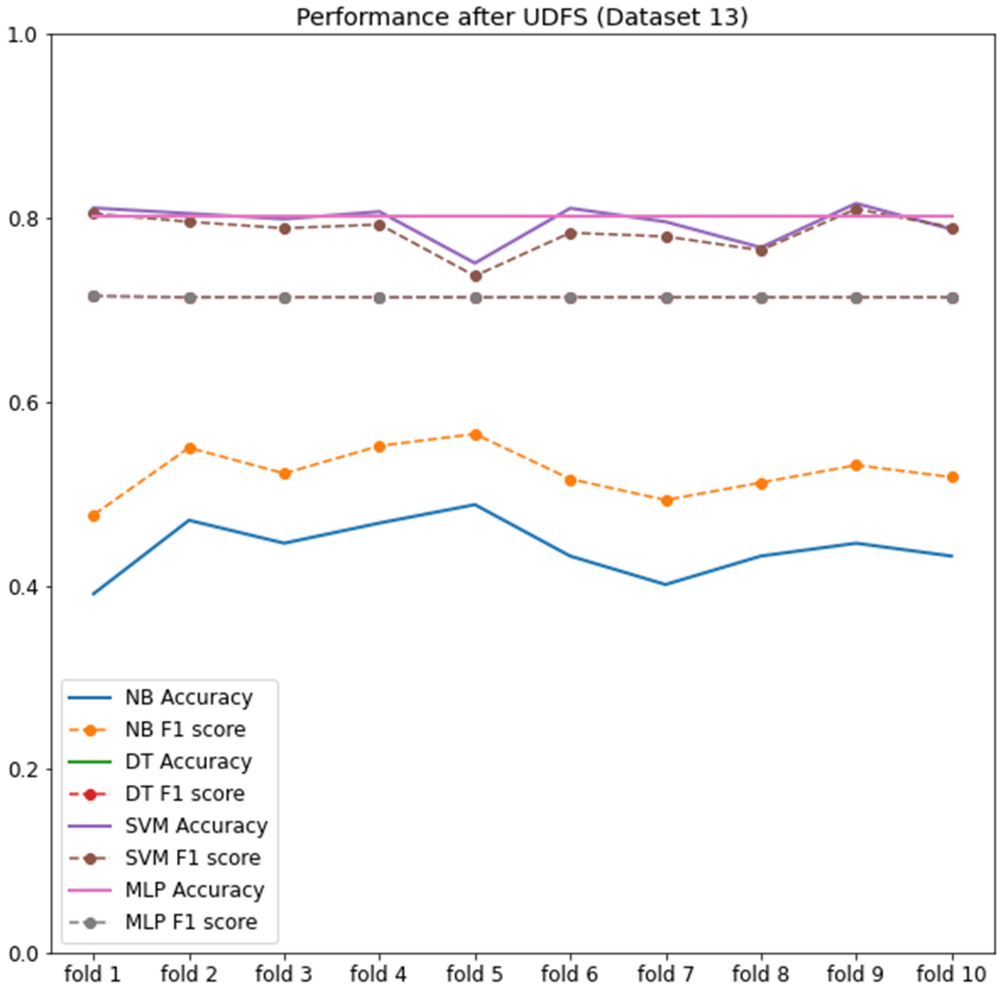


The graphs below compare the accuracy and f1 score measures for all classification methods on Dataset D13 before and after performing feature selection using UDFS.

Feature_Selection-09



Feature_Selection-09



References:

[1] [Yi Yang1, Heng Tao Shen1, Zhigang Ma2, Zi Huang1, Xiaofang Zhou1 (2019). *Norm Regularized Discriminative Feature Selection for Unsupervised Learning*. [online] Ijcai.org. Available at: https://www.ijcai.org/Proceedings/11/Papers/267.pdf [Accessed 14 Jul. 2019].

[2] YM, Z. (2019). *Feature Selection and Kernel Learning for Local Learning-Based Clustering. - PubMed - NCBI*. [online] Ncbi.nlm.nih.gov. Available at: https://www.ncbi.nlm.nih.gov/pubmed/21135434 [Accessed 16 Jul. 2019].

[3] Hall, M. (2019). *Correlation-based Feature Selection for Machine Learning*. [online] Cs.waikato.ac.nz. Available at: https://www.cs.waikato.ac.nz/~mhall/thesis.pdf [Accessed 14 Jul. 2019].