`

Department of Electrical and Computer Engineering

Applied Machine Learning
(ELEC8900-76-R-2021F)

"Feature Selection_10" (FS-10)

Instructor:
Dr. Roozbeh Razavi-Far

Submitted by

Shivangi Dhiman (110056290)

University of Windsor

Date: November 29, 2021

ABSTRACT

High-dimensional data analysis is a challenge for engineers and researchers in the fields of machine learning and data mining. Feature selection is an efficient way of solving this challenge by removing redundant data which helps reduce irrelevant features and provides a better understanding of learning model and data. Feature selection is vital in different areas of applications where datasets with many features is provided. In this approach, we choose a subset of relevant features to improvise the execution time and classification f1 score and accuracy. It helps to remove the not important features to increase the precision and improve the algorithm performance.

General classes of FS algorithms are filter methods, wrapper methods and embedded methods. Filter method applies a statistical measure to assign a scoring to each feature and thus we obtain features ranked by their scores. In wrapper methods, selection of a set of features is prepared, computed, and compared to other combinations. Embedded methods are best contributors to obtain the accuracy of the model.

This project focuses on implementation of three feature selection algorithms such as (Unsupervised Discriminative Feature Selection [UDFS], Local Learning-based Clustering Feature Selection [LLCFS], Correlation-based Feature Selection [CFS]) along with 3 classifiers (Random Forest (RF)**,** Multilayer Perceptron (MLP)**,** k-Nearest Neighbour (k-NN)**)** to obtain the best possible accuracy and f1 score for given dataset which is further cross-validated using k-fold cross validation technique.

Keywords

Classifier, Feature Selection, Unsupervised Discriminative Feature Selection [UDFS], Local Learning-based Clustering Feature Selection [LLCFS], Correlation-based Feature Selection [CFS], Random Forest (RF)**,** Multilayer Perceptron (MLP)**,** k-Nearest Neighbor (k-NN)**,** Machine Learning, F1 Score, accuracy, confusion Matrix, MATLAB, python, libraries

Development Software Requirements
MATLAB R2020a (License required for successful installation)
Anaconda Software with Jupiter Notebook (https://anaconda.com/)
Google Colab (Used for faster computation of data)

INTRODUCTION

Machine learning works on the given datasets and gives us the best prediction. It passes the huge datasets through different algorithms and classifiers to give us desired output. It has two main types of learning: supervised learning and unsupervised learning.

In supervised learning the datasets are labelled, and they are passed through the trained algorithms, whereas in unsupervised learning the datasets forms a cluster of the similar kind of data and it is not labelled. Classification is used to classify the datasets to give desired output and regression is used to make predictions

Feature Selection Algorithms and Classifiers used in the project are as below:

FEATURE SELECTION METHODS:

1. **Unsupervised Discriminative Feature Selection [UDFS]**

   Unsupervised Discriminative Feature Selection (UDFS) algorithm is a one-step approach that aims to select the most discriminative features in batch mode for data representation for unsupervised learning [1]. The algorithm optimizes the features and helps to generate an output with weights and feature rankings. Pseudo-code for the UDFS algorithm is presented in Algorithm 1. The algorithm is proposed based on the L-2, 1 norm regularization approach for minimization of the objective function with orthogonal constraint [1]

2. **Local Learning-based Clustering Feature Selection [LLCFS]**
   Local Learning-based Clustering Feature Selection (LLCFS) algorithm tries to check that the cluster label of each data point is near to the one anticipated by the local regression model with the help of the neighboring points and their cluster label. Firstly, it finds the area where the data points are found and then will try to find the boundary by using the neighboring points [2].

3. **Correlation-based Feature Selection [CFS]**
   The correlation-based Feature Selection (CFS) method identifies a metric to evaluate the efficacy of the feature subset. The idea of CFS states that a good feature subset has features that are highly correlated with the output (predictive of) class, yet uncorrelated to each other. [3].

CLASSIFICATION METHODS:
Classification uses algorithms to divide the assigned dataset precisely into different groups. It tries to find a pattern in the input and labels or defines the group of data accordingly. Some examples of classification algorithms are k-nearest Neighbor, Random Forest, and many more.

a. Random Forest (RF)

Random Forest classifies the data in different decision trees with different samples and then take the votes of all the decision trees. It counts the majority of votes from all the trees and then provide us with the output.

Default parameter as below:
class sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None) [4]

b. Multi-Layer Perceptron (MLP)

In MLP the input layer and output layer are completely connected with multiple hidden layers in between. When one hidden layer output is done it is pushed through the activation function onto the other hidden layer. Basically, the output data of hidden layer is pushed to another layer by doing dot product with the i/p weights given. And through all this we get groups of different data which is very precise at the final output.

Default parameter as below:
class sklearn.neural_network.MLPClassifier(hidden_layer_sizes=(100), activation='relu', *, solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000) [4]

c. k- Nearest Neighbor (k-NN)

K-Nearest Neighbor (k-NN) is an algorithm that assumes that similar kinds of data can be found around each other. It does not classify the data based on similar parameters, it will try to classify the data by the closeness and interrelation to the other provided data. It will find the distance between the points usually using Euclidean distance and then labels the group using the most frequent category or the higher average of the data points.

Default parameter as below:
class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None) [4]

IMPLEMENTATION PROCEDURE:

For this project, we have used MATLAB R2020a for feature selection algorithm. It provides inbuilt feature selection library for different feature selection algorithm and provides the feature ranking.
• To further process the ranking, we have used python language and google colab to execute python code.

• In MATLAB, we initially read our dataset and then we divide the dataset into training (80%) and testing dataset (20%).

• Once the partition is done, we select the algorithm (UDFS, LLCFS, CFS) to get the ranking of features.

• The next step is to use these feature ranking and apply different classifiers to calculate the accuracy and f-measure.

• We use different classifiers (KNN,RF, and MLP) to calculate the f-measure and accuracy.

• Then, we perform k-fold cross validation for 10 folds for better prediction.

• Till the point we got the f-measure and accuracy of each FS algorithm with different classifiers.
• Now we apply different classifiers directly on dataset without feature selection.

• We perform k-fold cross validation and calculate the f-measure and accuracy.

• The comparison of accuracy and f-measure with feature ranking and without feature ranking provides us the information about the best method for feature selection.

KEY MATLAB AND 'PYTHON' FILES USED FOR PROJECT

We have used the following codes to implement the three-feature selection algorithm and classifiers. These files are generated using MATLAB feature selection library and Anaconda (Jupyter Notebook) python-based environment which we have ran on Google Colab.
To execute .m files we have used Matlab and to execute python code we have used Google Colab.

1. **Baseline code : baseline.py**

This python file contains the code that separates the dataset into testing and training data. Post splitting the data, we have used data pre-processing to simplify the training and testing process by appropriately transforming the whole dataset. Once the separation is done, the classifiers (RF, KNN, and MLP) are applied to calculate the accuracy and f-measure of respective classifiers.

2. FS10 Algo: FS10_FSAlgo.m
This matlab file contain feature selection algorithms (UDFS, CFS, LLCDFS) and provides the ranking as an output.

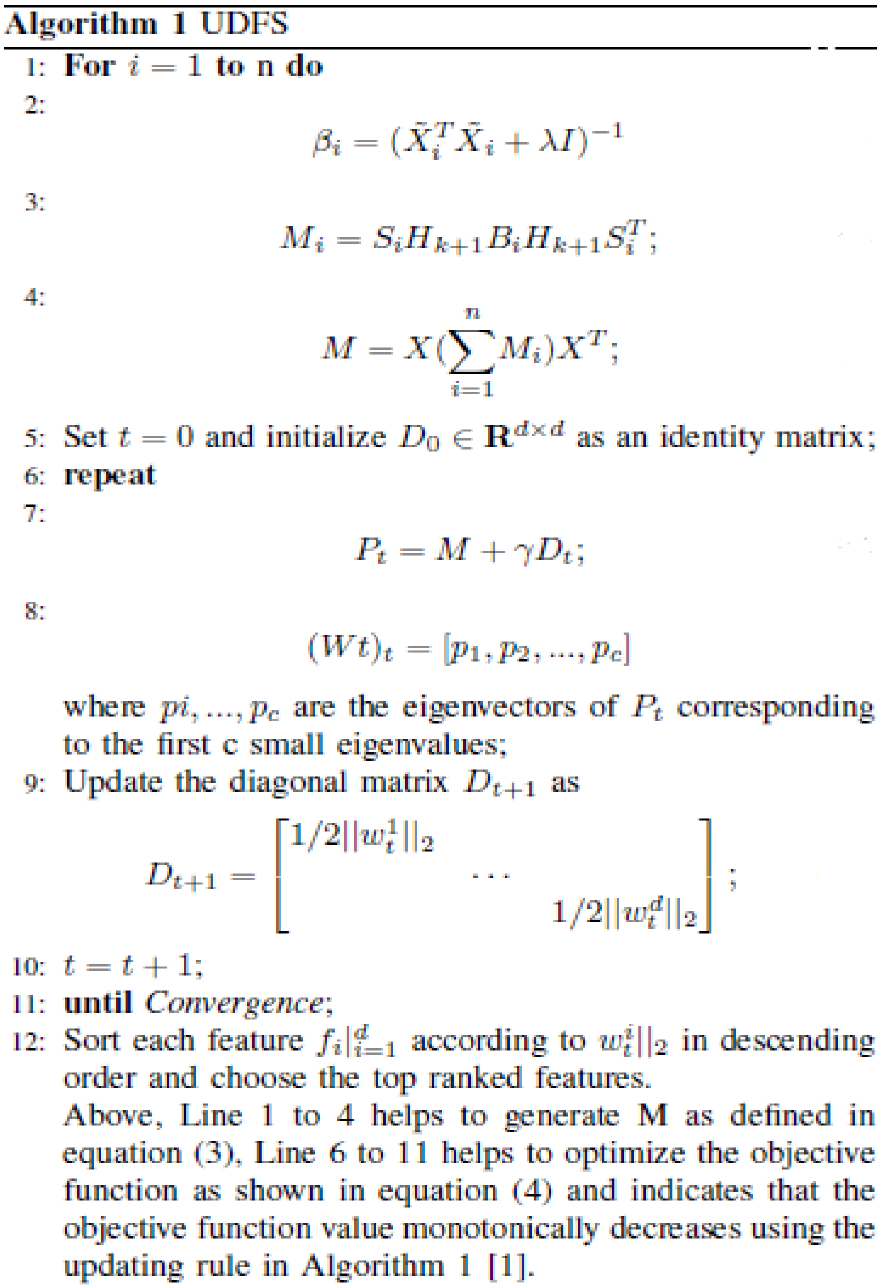
3**. UDFS Algo : UDFS.m**
This file contains the code of UDFS algorithm in matlab.
4.**CFS Algo : cfs.m**
This file contains the code of CFS algorithm in matlab.

5. LLCFS Algo : llcfs.m
This file contains the code of LLCFS algorithm in matlab.

6. FS10 Classifiers : FS10_Classifiers.py
This file uses the ranking which we got from FS10_FSAlgo.m file. Then we apply different classifiers (RF, KNN, and MLP) to calculate the accuracy and f-measure of each algorithm.

FEATURE SELECTION ALGORITHM STEPS

1) UDFS (Unsupervised Discriminative Feature Selection)

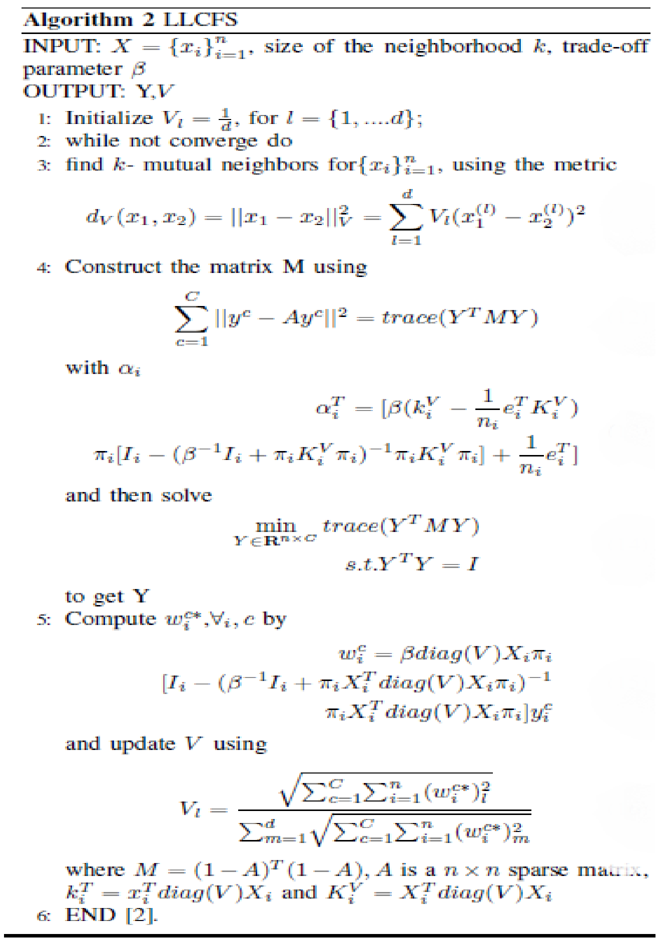


```
Algorithm 1 UDFS
1:  For $i = 1$ to n do
2:      $\beta_i = (\tilde{X}_i^T \tilde{X}_i + \lambda I)^{-1}$
3:      $M_i = S_i H_{k+1} B_i H_{k+1} S_i^T;$
4:      $M = X(\sum_{i=1}^{n} M_i) X^T;$
5:  Set $t = 0$ and initialize $D_0 \in \mathbf{R}^{d \times d}$ as an identity matrix;
6:  repeat
7:      $P_t = M + \gamma D_t;$
8:      $(Wt)_t = [p_1, p_2, ..., p_c]$
        where $pi, ..., p_c$ are the eigenvectors of $P_t$ corresponding
        to the first c small eigenvalues;
9:      Update the diagonal matrix $D_{t+1}$ as
        $D_{t+1} = \begin{bmatrix} 1/2||w_t^1||_2 & & \\ & \cdots & \\ & & 1/2||w_t^d||_2 \end{bmatrix};$
10:     $t = t + 1;$
11: until Convergence;
12: Sort each feature $f_i|_{i=1}^{d}$ according to $w_t^i||_2$ in descending
    order and choose the top ranked features.
```

Above, Line 1 to 4 helps to generate M as defined in equation (3), Line 6 to 11 helps to optimize the objective function as shown in equation (4) and indicates that the objective function value monotonically decreases using the updating rule in Algorithm 1 [1].

Figure 1: UDFS Algorithm (Pseudo-code) [1]

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



Algorithm 2 LLCFS

```
INPUT: $X = \{x_i\}_{i=1}^{n}$, size of the neighborhood $k$, trade-off parameter $\beta$
OUTPUT: Y,$V$
1: Initialize $V_l = \frac{1}{d}$, for $l = \{1, ....d\}$;
2: while not converge do
3: find $k$- mutual neighbors for$\{x_i\}_{i=1}^{n}$, using the metric
       $$d_V(x_1, x_2) = ||x_1 - x_2||_V^2 = \sum_{l=1}^{d} V_l(x_1^{(l)} - x_2^{(l)})^2$$
4: Construct the matrix M using
       $$\sum_{c=1}^{C} ||y^c - Ay^c||^2 = trace(Y^T MY)$$
   with $\alpha_i$
       $$\alpha_i^T = [\beta(k_i^V - \frac{1}{n_i}e_i^T K_i^V) \pi_i[I_i - (\beta^{-1}I_i + \pi_i K_i^V \pi_i)^{-1}\pi_i K_i^V \pi_i] + \frac{1}{n_i}e_i^T]$$
   and then solve
       $$\min_{Y \in \mathbf{R}^{n \times C}} trace(Y^T MY) \quad s.t. Y^T Y = I$$
   to get Y
5: Compute $w_i^{c*}$,$\forall_i, c$ by
       $$w_i^c = \beta diag(V) X_i \pi_i [I_i - (\beta^{-1}I_i + \pi_i X_i^T diag(V) X_i \pi_i)^{-1} \pi_i X_i^T diag(V) X_i \pi_i] y_i^c$$
   and update $V$ using
       $$V_l = \frac{\sqrt{\sum_{c=1}^{C}\sum_{i=1}^{n}(w_i^{c*})_l^2}}{\sum_{m=1}^{d}\sqrt{\sum_{c=1}^{C}\sum_{i=1}^{n}(w_i^{c*})_m^2}}$$
   where $M = (1-A)^T(1-A)$, $A$ is a $n \times n$ sparse matrix, $k_i^T = x_i^T diag(V) X_i$ and $K_i^V = X_i^T diag(V) X_i$
6: END [2].
```

Figure 2: LLCFS Algorithm (Pseudo-code) [2]

3) CFS (Correlation Feature Selection)

1. Training data is discretized first in the data pre-processing stage.
2. It is further passed to CFS where CFS computes feature- feature and feature class correlations using any one measure as stated. There are three measures i.e. relief which uses symmetrical relief to measure correlations, minimum description length (MDL) which uses normalized symmetrical MDL to measure correlations, and symmetric uncertainty which uses uncertainty to measure correlation.
3. Then, it searched for future subset space.
4. The subset with the highest efficacy/merit during the search will be used to decrease the dimensionality of both testing and training data.
5. Both decreased data sets are then passed on to a machine learning algorithm for testing and training the data [3].

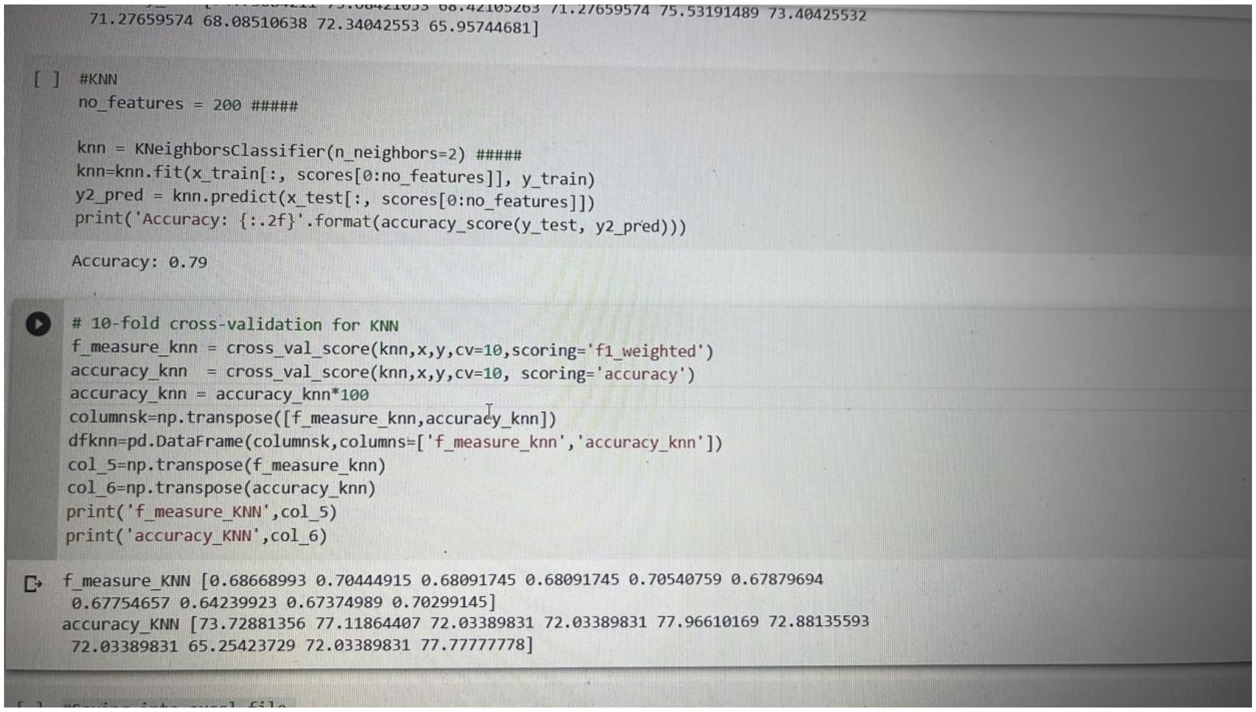
6: Thus, the final evaluation computes providing estimated accuracy of tested and trained data.

MAIN ALGORITHM OF CLASSIFICATION METHOD ACROSS ALL FS:

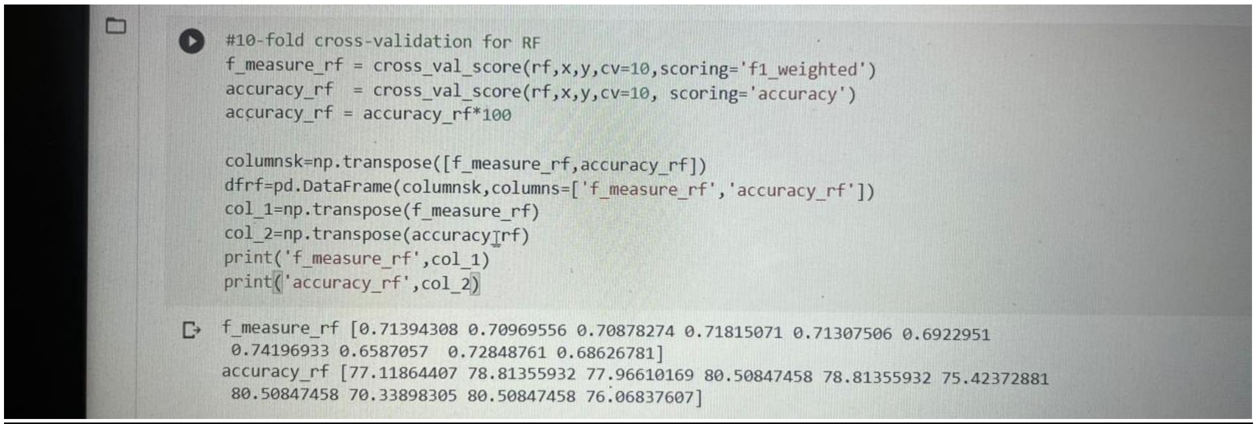
1) Separate the given dataset in X and Y variables
2) Splitting the data set in train: test: validation = 80:20:20
3) Run the python code in google colab required according to classifier (SVM,DT,RNN,NBS)
4) Choose the training parameter according to the code requirement.
5) Fit the data in the model you named.
6) Predict the y-test prediction score for output based on $X_{test;}$
7) Determine the accuracy and the f1 score between the (train, test) and (train,validation) set for K-fold cross validationmethod.

RESULTS:

Accuracy and F1 Score of D13 with classifiers:

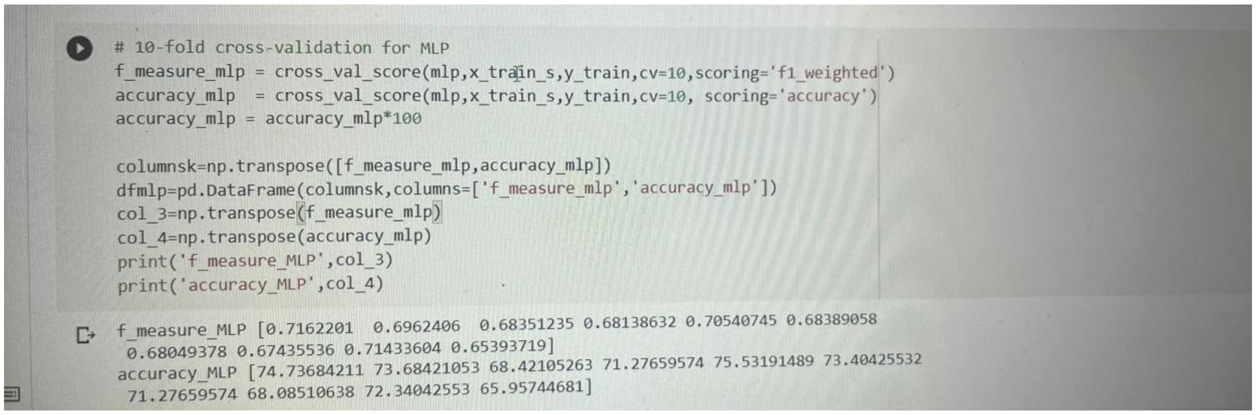


```
 71.27659574 68.08510638 72.34042553 65.95744681]

[ ] #KNN
    no_features = 200 #####

    knn = KNeighborsClassifier(n_neighbors=2) #####
    knn=knn.fit(x_train[:, scores[0:no_features]], y_train)
    y2_pred = knn.predict(x_test[:, scores[0:no_features]])
    print('Accuracy: {:.2f}'.format(accuracy_score(y_test, y2_pred)))

    Accuracy: 0.79

    # 10-fold cross-validation for KNN
    f_measure_knn = cross_val_score(knn,x,y,cv=10,scoring='f1_weighted')
    accuracy_knn  = cross_val_score(knn,x,y,cv=10, scoring='accuracy')
    accuracy_knn = accuracy_knn*100
    columnsk=np.transpose([f_measure_knn,accuracy_knn])
    dfknn=pd.DataFrame(columnsk,columns=['f_measure_knn','accuracy_knn'])
    col_5=np.transpose(f_measure_knn)
    col_6=np.transpose(accuracy_knn)
    print('f_measure_KNN',col_5)
    print('accuracy_KNN',col_6)

    f_measure_KNN [0.68668993 0.70444915 0.68091745 0.68091745 0.70540759 0.67879694
     0.67754657 0.64239923 0.67374989 0.70299145]
    accuracy_KNN [73.72881356 77.11864407 72.03389831 72.03389831 77.96610169 72.88135593
     72.03389831 65.25423729 72.03389831 77.77777778]
```



```
#10-fold cross-validation for RF
f_measure_rf = cross_val_score(rf,x,y,cv=10,scoring='f1_weighted')
accuracy_rf  = cross_val_score(rf,x,y,cv=10, scoring='accuracy')
accuracy_rf = accuracy_rf*100

columnsk=np.transpose([f_measure_rf,accuracy_rf])
dfrf=pd.DataFrame(columnsk,columns=['f_measure_rf','accuracy_rf'])
col_1=np.transpose(f_measure_rf)
col_2=np.transpose(accuracy_rf)
print('f_measure_rf',col_1)
print('accuracy_rf',col_2)
```

```
f_measure_rf [0.71394308 0.70969556 0.70878274 0.71815071 0.71307506 0.6922951
 0.74196933 0.6587057  0.72848761 0.68626781]
accuracy_rf [77.11864407 78.81355932 77.96610169 80.50847458 78.81355932 75.42372881
 80.50847458 70.33898305 80.50847458 76.06837607]
```



```
# 10-fold cross-validation for MLP
f_measure_mlp = cross_val_score(mlp,x_train_s,y_train,cv=10,scoring='f1_weighted')
accuracy_mlp  = cross_val_score(mlp,x_train_s,y_train,cv=10, scoring='accuracy')
accuracy_mlp = accuracy_mlp*100

columnsk=np.transpose([f_measure_mlp,accuracy_mlp])
dfmlp=pd.DataFrame(columnsk,columns=['f_measure_mlp','accuracy_mlp'])
col_3=np.transpose(f_measure_mlp)
col_4=np.transpose(accuracy_mlp)
print('f_measure_MLP',col_3)
print('accuracy_MLP',col_4)
```

```
f_measure_MLP [0.7162201  0.6962406  0.68351235 0.68138632 0.70540745 0.68389058
 0.68049378 0.67435536 0.71433604 0.65393719]
accuracy_MLP [74.73684211 73.68421053 68.42105263 71.27659574 75.53191489 73.40425532
 71.27659574 68.08510638 72.34042553 65.95744681]
```

DETAILED ANALYSIS WITH SPECIFIC EXAMPLE

Here, we are showing the results of **an example Dataset 13** to visualize the idea behind the project i.e. to train the Machine learning model with different feature selection algorithms and classifier. The goal is to select the best matching model with the use of Accuracy and F1 score evaluation. Firstly, we analyze the Evaluation for the Raw data (Baseline) i.e. without feature selection algorithms:

Baseline	RF		MLP		Knn	
Fold	**F-Measure**	**Accuracy**	**F-Measure**	**Accuracy**	**F-Measure**	**Accuracy**
fold1	0.713943	0.7881356	0.71203	0.7473684	0.595063	0.5762712
fold2	0.708783	0.7881356	0.695266	0.7263158	0.658706	0.6864407
fold3	0.713075	0.7881356	0.691483	0.7263158	0.641491	0.6271186
fold4	0.718151	0.8050847	0.695105	0.7234043	0.559591	0.5169492
fold5	0.717327	0.779661	0.682877	0.7021277	0.631948	0.6186441
fold6	0.687839	0.7457627	0.726727	0.7340426	0.612564	0.5932203
fold7	0.741969	0.7966102	0.689954	0.7234043	0.601602	0.5762712
fold8	0.689674	0.7457627	0.687234	0.7234043	0.511012	0.4491525
fold9	0.731966	0.8050847	0.738355	0.7765957	0.652658	0.6355932
fold10	0.701781	0.7435897	0.673979	0.7234043	0.641324	0.6410256

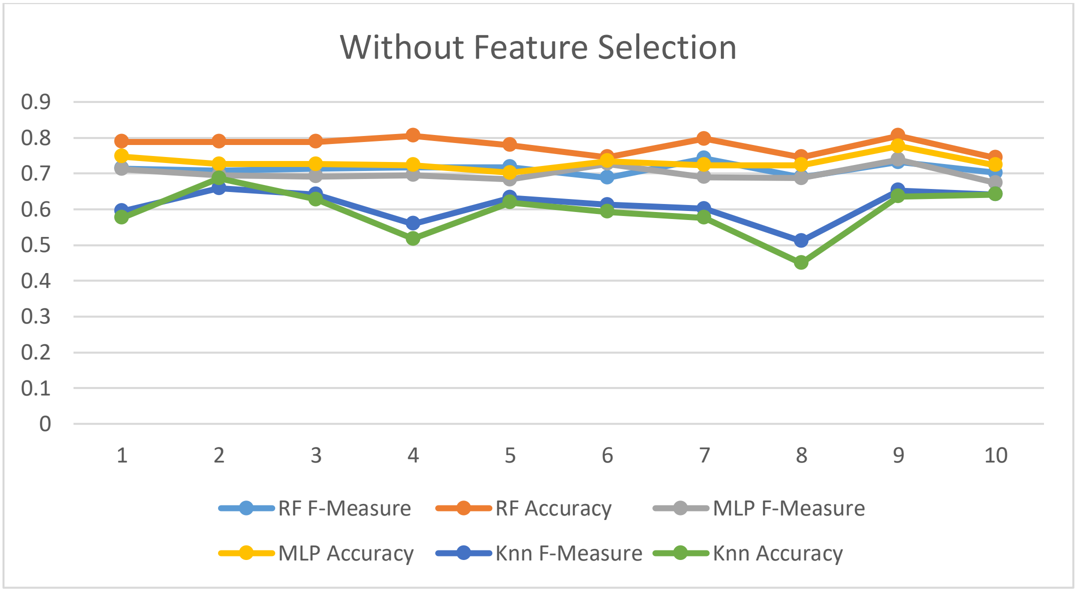


Figure 3: Without Feature Selection analysis with all Classifiers

Now, we select to the individual classifier with Evaluation Parameters as below:

Random Forest (RF)

RF	**F-Measure**			**Accuracy**		
Fold	**UDFS**	**LLCFS**	**CFS**	**UDFS**	**LLCFS**	**CFS**
fold1	0.705408	0.709696	0.713943	0.7711864	0.7881356	0.7711864
fold2	0.713075	0.709696	0.709696	0.779661	0.779661	0.7881356
fold3	0.713075	0.713075	0.708783	0.7711864	0.7881356	0.779661
fold4	0.718151	0.718151	0.718151	0.8050847	0.8050847	0.8050847
fold5	0.716487	0.716487	0.713075	0.779661	0.7966102	0.7881356
fold6	0.687839	0.692295	0.692295	0.7457627	0.7288136	0.7542373
fold7	0.72841	0.72841	0.741969	0.8135593	0.8135593	0.8050847
fold8	0.638899	0.66822	0.658706	0.7118644	0.6610169	0.7033898
fold9	0.731966	0.728488	0.728488	0.8050847	0.8050847	0.8050847
fold10	0.701341	0.704542	0.686268	0.7435897	0.7521368	0.7606838

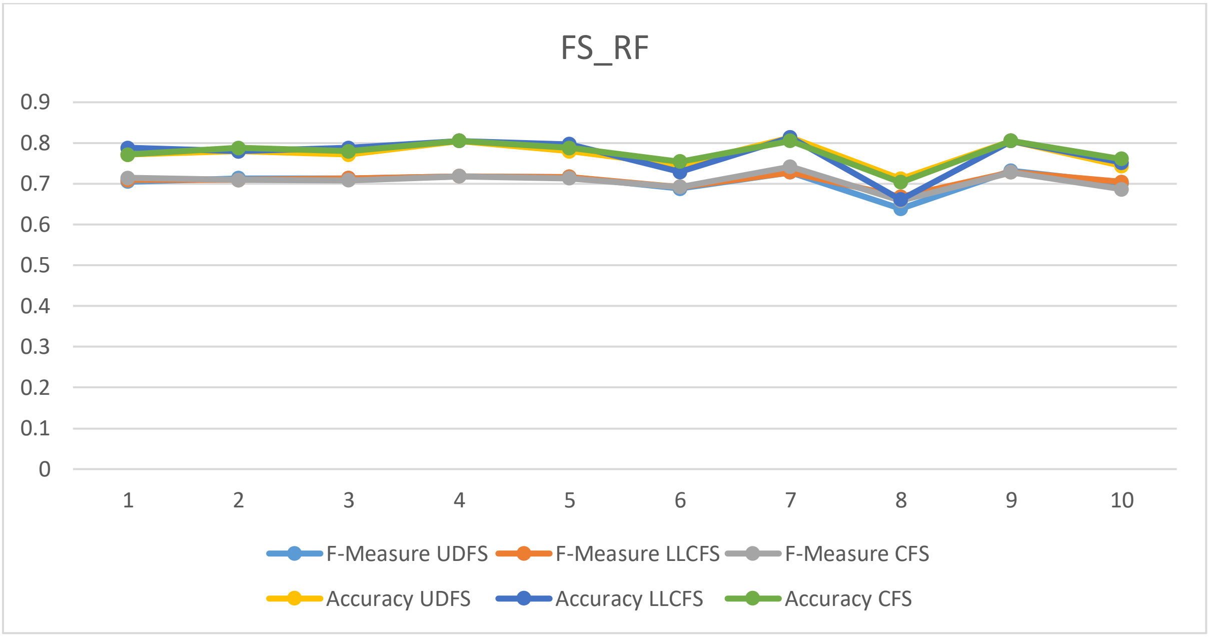


Figure 4: FS_RF Computation

Analysis:

The Average Value of winning Accuracy value is for the UDFS, LLCFS and CFS with RF classifier is approximately 77% with Acceptable F1- Score.

Multilayer Perceptron (MLP)

MLP	F-Measure			Accuracy		
Fold	UDFS	LLCFS	CFS	UDFS	LLCFS	CFS
fold1	0.704892	0.67842	0.71622	0.7684211	0.6947368	0.7473684
fold2	0.70862	0.735162	0.696241	0.7789474	0.7894737	0.7368421
fold3	0.647773	0.692519	0.683512	0.6736842	0.7157895	0.6842105
fold4	0.716479	0.703191	0.681386	0.7659574	0.7340426	0.712766
fold5	0.698138	0.701726	0.705407	0.7446809	0.7340426	0.7553191
fold6	0.712512	0.725508	0.683891	0.7234043	0.7659574	0.7340426
fold7	0.683891	0.714539	0.680494	0.7340426	0.7553191	0.712766
fold8	0.699536	0.747099	0.674355	0.7234043	0.7765957	0.6808511
fold9	0.685352	0.690585	0.714336	0.712766	0.7021277	0.7234043
fold10	0.671907	0.714267	0.653937	0.712766	0.7659574	0.6595745

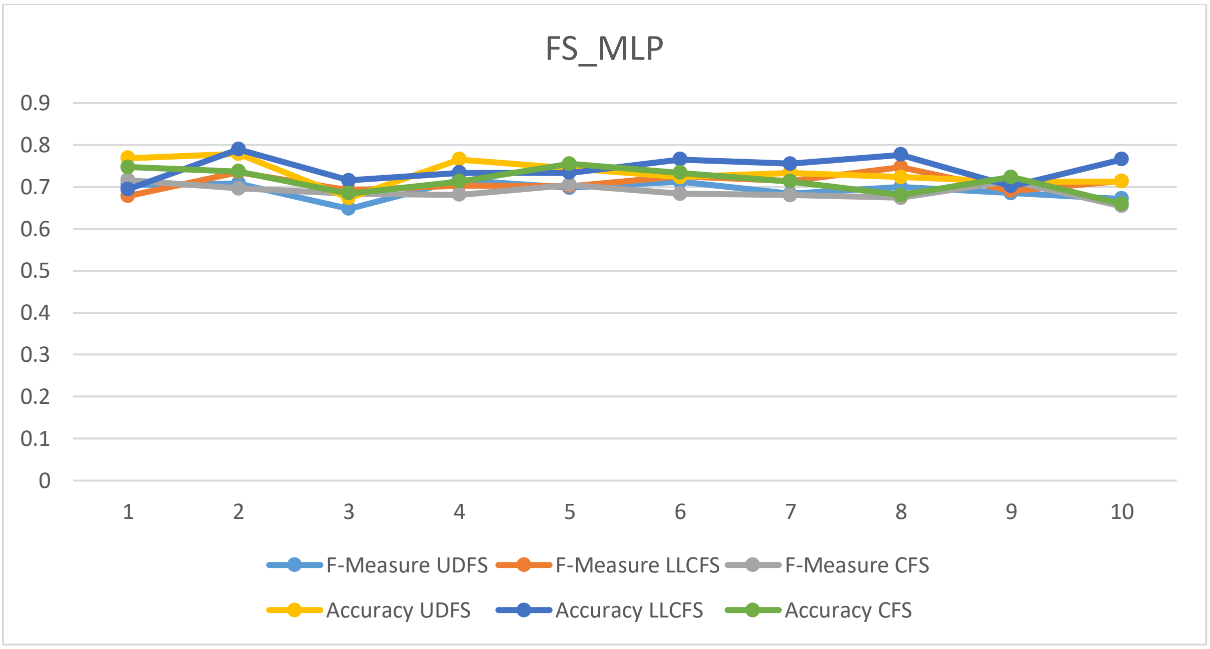


Figure 5: FS_MLP Computation

Analysis:

The Average Value of winning Accuracy value is for the UDFS, LLCFS and CFS with MLP classifier is approximately 73% with Acceptable F1- Score.

K- Nearest Neighbor Classifier

Knn	F-Measure			Accuracy		
Fold	UDFS	LLCFS	CFS	UDFS	LLCFS	CFS
fold1	0.592755	0.68669	0.68669	0.5677966	0.7372881	0.7372881
fold2	0.672907	0.704449	0.704449	0.7118644	0.7711864	0.7711864
fold3	0.627868	0.680917	0.680917	0.6186441	0.720339	0.720339
fold4	0.571151	0.680917	0.680917	0.5338983	0.720339	0.720339
fold5	0.640605	0.705408	0.705408	0.6271186	0.779661	0.779661
fold6	0.635135	0.678797	0.678797	0.6186441	0.7288136	0.7288136
fold7	0.584698	0.677547	0.677547	0.5508475	0.720339	0.720339
fold8	0.51409	0.642399	0.642399	0.4491525	0.6525424	0.6525424
fold9	0.658943	0.67375	0.67375	0.6440678	0.720339	0.720339
fold10	0.625321	0.702991	0.702991	0.6153846	0.7777778	0.7777778

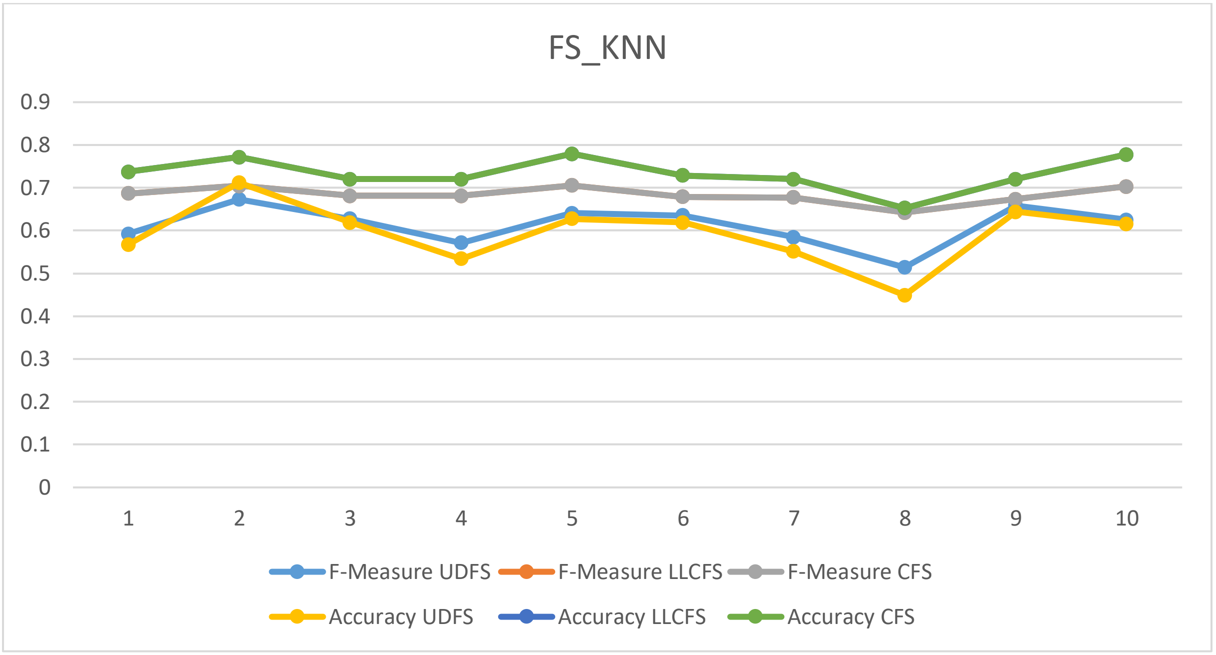


Figure 6: FS_KNN Computation

Analysis:

The Average Value of winning Accuracy value is for the UDFS, LLCFS and CFS with KNN classifier, which is approximately 77.9% with Acceptable F1- Score.

Conclusion of Dataset:

- The final comparison of the data set's value for **Dataset 13** shows that the k-nn classifier with **LLCFS and CFS give us the highest Accuracy**. And that is more than what was obtained without Feature selection. There is certain percent of the increase in Accuracy with Acceptable F1-Score.
- The Advantage is the reduction of data due to Feature selection returns the important features only for the measurement.
- This merit us in term of time and computational complexity and precision.

F1 score:

F1_Score is term used for the accuracy in the matrix dimensional dataset. It's derived from the precision (P) and Recall (R). To obtain the score, we need to use the correct positive result and divided it with the sum of all sample that should have to be identify the positive. The F1 Score is the harmonic average of the precision and recall. Formula is as below:
F1 = (2*P*R) / (P+R)

Accuracy:

In Reality, accuracy score give the error rate of the set from the original dataset and in the data visualization such data give the distance difference in scatter plotting and show how much they are far from the true value in 2D dimension.

Computational Complexity:

- Computational complexity is the number of resources required to execute a particular algorithm with given set of inputs.
- Resources involved in the complexity are time and space. Time resources indicate how much time an algorithm is taking
- to execute and space resource indicates how much memory an algorithm is utilizing to execute.
- In machine learning, computational complexity increases or decreases with the dimensions of dataset.

Algorithm	**Complexity**
UDFS	O((n^2/2)*T)
CFS	O(T^3 + nT^2)

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].

[4] Journal of Machine Learning Research, 2011. Scikit-learn: Machine Learning in Python. 12, pp.2825--2830. Available Online: https://scikit-learn.org/stable/.