

**FINAL REPORT**  
**APPLIED MACHINE LEARNING**



**University  
of Windsor**

**DR19 - Dimensionality Reduction and  
Classification**

**INSTRUCTOR:** Dr. Roozbeh Razavi-Far

**PREPARED BY:**

Neha Goyal- 104941368  
Ahmed Ali Syed – 104932919

# INDEX

1. Introduction
2. Objective of the Project
3. Dimensionality Reduction Methods
  - GPLVM
  - SNE
  - Symmetric SNE
4. Classification Methods
  - Naïve Bayes
  - Decision Tree
  - Support Vector Machine
  - Recurrent Neural Networks
5. Codes for DR Methods
6. Pipeline of the Project
7. Performance Evaluation
  - Special Case for Gear 18
8. Calculation of Variability
9. Computational Complexities
10. Conclusion
11. References

## 1. Introduction:

In machine learning classification problems, there are often too many features on the basis of which the final classification is done. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction. For example, for 3-D classification problem features overlap each other so it can be hard to visualize, whereas a 2-D one can be mapped to a simple 2 dimensional space, and a 1-D problem to a simple line.

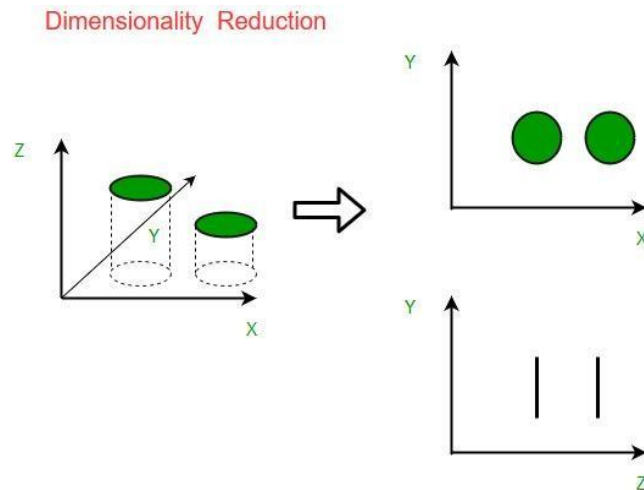


Figure1. 3-D feature space is split into two 1-D feature spaces

There are two components of dimensionality reduction:

1. Feature Selection
2. Feature Extraction

Classification can be defined as the technique of recognizing the category to which new data belongs on the basis of a training set of data containing instances whose category membership is known.

In this report, we will discuss the objectives, the assigned Dimensionality Reduction and Classification methods. Summary of work is also discussed followed by the tentative timeline of the project.

## 2. **Objective of the Project:**

- Reducing the high dimensional datasets to low dimensions by assigned Data Reduction methods.
- Data reduction model is trained with the optimal parameters with the training and validation datasets.
- Using nested Cross - Validation strategy to tune the parameters of classification results.
- For each Data reduction and classification method comparisons are noted.

### 3. Dimensionality Reduction Methods:-

#### a. Gaussian Process Latent Variable Model

Gaussian process latent variable models (GPLVM) are probabilistic dimensionality reduction methods that use Gaussian Processes (GPs) to find a lower dimensional non-linear embedding of high dimensional data. They are an extension of the Probabilistic formulation of PCA. The model is defined probabilistically and the latent variables are then marginalized and parameters are obtained by maximizing the likelihood. Like kernel PCA they use a kernel function to form a nonlinear mapping (in the form of a Gaussian process). However, in the GPLVM the mapping is from the embedded latent space to the data space (like density networks and GTM) whereas in kernel PCA it is in the opposite direction. It was originally proposed for visualization of high dimensional data but has been extended to construct a shared manifold model between two observation spaces

#### b. Stochastic Neighbor Embedding

SNE tries to map high-dimensional into low-dimensional objects by preserving the neighbourhood relationship structure in spite of a trade-off consisting of a misclassification around the far objects.

##### **In high-dimensional space:**

- ☐ This method converts Euclidean distances between datapoints into conditional probabilities that represent similarities.
- ☐ The similarity of datapoint  $x_j$  to datapoint  $x_i$  is the conditional probability,  $p_{ji}$ , that  $x_i$  would pick  $x_j$  as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at  $x_i$ .
- ☐ For nearby datapoints,  $p_{ji}$  is relatively high, whereas for widely separated datapoints,  $p_{ji}$  will be extremely small

##### **In low-dimensional space:**

- ☐ For low-dimensional counterparts  $y_i$  and  $y_j$  of high-dimensional datapoints  $x_i$  and  $x_j$ , it is possible to compute a similar conditional probability denoted by  $q_{ji}$ .
- ☐ Set the variance of the Gaussian employed in computation of conditional probability  $q_{ji}$  to  $1/\sqrt{2}$
- ☐ If the  $y_i$  and  $y_j$  correctly model the datapoints  $x_i$  and  $x_j$ , their probabilities will be equal.
- ☐ This is achieved by minimizing a cost function which is a sum of Kullback-Leibler divergences between the original ( $p_{ji}$ ) and induced ( $q_{ji}$ ) distributions.

### **c. Symmetric Stochastic Neighbor Embedding**

In symmetric SNE, we minimize single Kullback-Leibler divergence between joint probability distribution,  $P$ , in the high-dimensional space and a joint probability distribution,  $Q$ , in the low-dimensional space. The main advantage of the symmetric SNE is the simpler form of its gradient, which is faster to compute. We refer this SNE as symmetric SNE, because it has the property that  $p_{ij} = p_{ji}$  and  $q_{ij} = q_{ji}$  for  $\forall i, j$ .

## **4. Classification Methods:**

### **a. Naïve Bayes**

Naive Bayes classifier is based on Bayes' theorem with the independence assumptions between predictors i.e. it assumes the presence of a feature in a class is unrelated to any other feature. Even if these features depend on each other or upon the existence of the other features, all of these properties independently. Thus, the name Naive Bayes.

### **b. Decision Tree**

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. It follows Iterative Dichotomiser 3 (ID3) algorithm structure for determining the split. It performs classification by finding the hyperplane that maximizes the margin between the two classes with the help of support vectors.

### **c. Support Vector Machine**

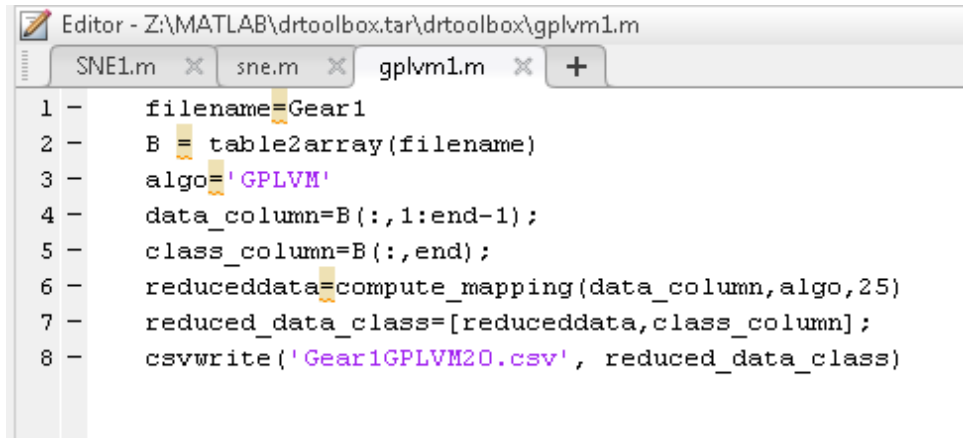
Support Vector is used for both regression and Classification. It is based on the concept of decision planes that define decision boundaries. A decision plane (hyperplane) is one that separates between a set of objects having different class memberships.

### **d. Recurrent Neural Network**

Recurrent means the output at the current time step becomes the input to the next time step. At each element of the sequence, the model considers not just the current input, but what it remembers about the preceding elements. RNN are very powerful as it distributes hidden state that allows them to store a lot of information about the past efficiently. This network is non-linear dynamics that allows them to update their hidden state in complicated ways. There is no need to infer hidden state, pure deterministic and it's a weight sharing model.

## 5. Codes for Dimensionality Reduction Methods:

### a. Gaussian Process Latent Variable Model



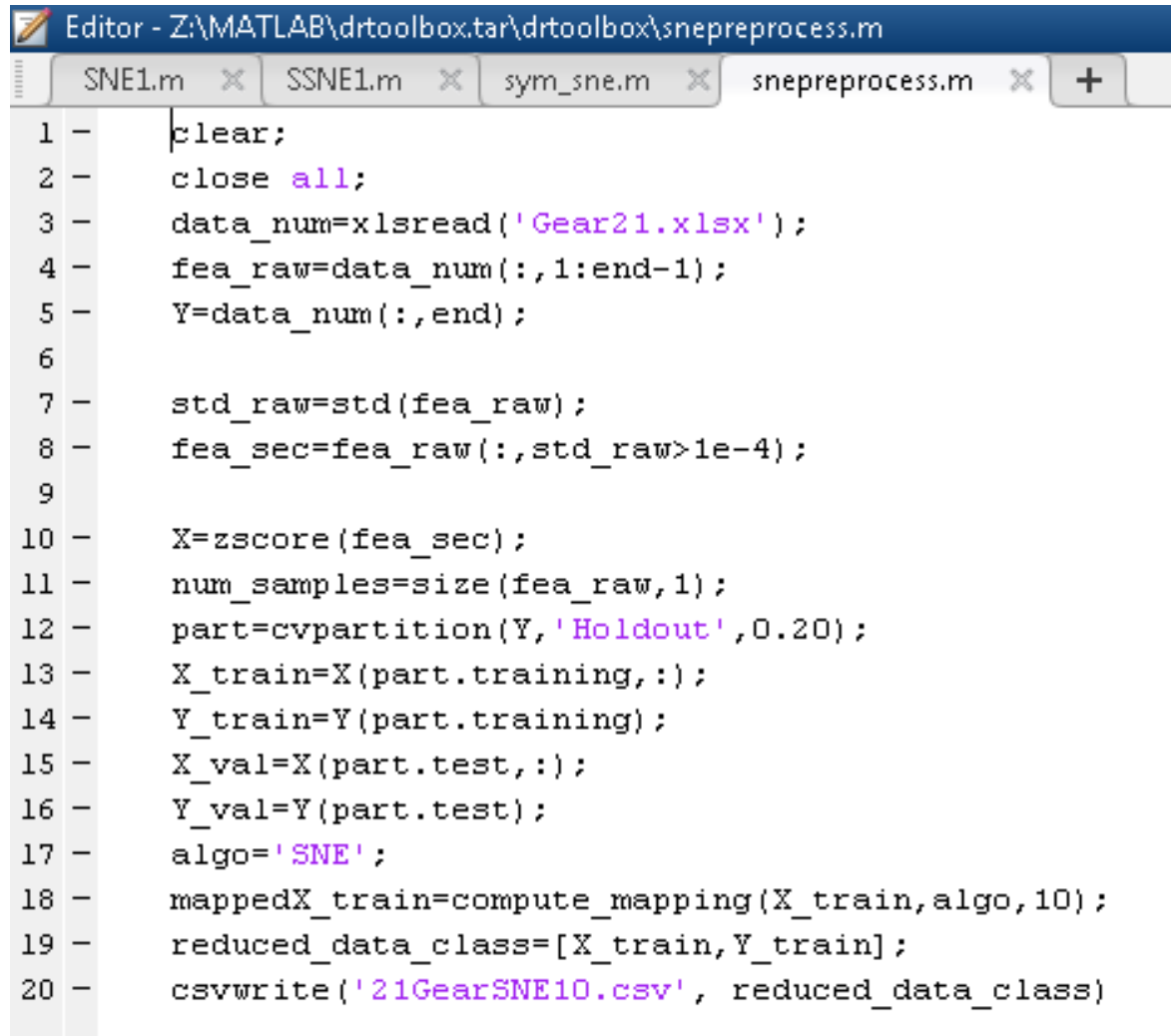
```
Editor - Z:\MATLAB\drtoolbox.tar\drtoolbox\gplvm1.m
SNE1.m x sne.m x gplvm1.m x +
1 - filename=Gear1
2 - B = table2array(filename)
3 - algo='GPLVM'
4 - data_column=B(:,1:end-1);
5 - class_column=B(:,end);
6 - reduceddata=compute_mapping(data_column,algo,25)
7 - reduced_data_class=[reduceddata,class_column];
8 - csvwrite('Gear1GPLVM20.csv', reduced_data_class)
```

#### Explanation:

- Load the database under a different file name and convert the database into array form.
- Separate the label from the data and store these two different data under different file names before computing the data.
- After that compute the data using compute\_mapping function and pass required parameters which includes compute\_mapping (A, type, no\_dim)
  - A -- Data on which reduction needs to be performed
  - type --- type of dimensionality method
  - no\_dim ---- Number of components / dimensions
- After that concatenate the reduced data with the labels and write reduced file with labels in csv to perform classification.

**The parameter 'algo' will call the file "GPLVM" which is already provided in the toolbox and it performs the dimensionality reduction and give the reduced dataset.**

## b. Stochastic Neighbor Embedding

The image shows a MATLAB Editor window with the title bar 'Editor - Z:\MATLAB\drtoolbox.tar\drtoolbox\snepreprocess.m'. There are four tabs open: 'SNE1.m', 'SSNE1.m', 'sym\_sne.m', and 'snepreprocess.m'. The 'snepreprocess.m' tab is active, showing a script with 20 lines of MATLAB code. The code performs data loading, normalization, partitioning, and dimensionality reduction using SNE. The code is as follows:

```
1 - clear;
2 - close all;
3 - data_num=xlsread('Gear21.xlsx');
4 - fea_raw=data_num(:,1:end-1);
5 - Y=data_num(:,end);
6
7 - std_raw=std(fea_raw);
8 - fea_sec=fea_raw(:,std_raw>1e-4);
9
10 - X=zscore(fea_sec);
11 - num_samples=size(fea_raw,1);
12 - part=cvpartition(Y,'Holdout',0.20);
13 - X_train=X(part.training,:);
14 - Y_train=Y(part.training);
15 - X_val=X(part.test,:);
16 - Y_val=Y(part.test);
17 - algo='SNE';
18 - mappedX_train=compute_mapping(X_train,algo,10);
19 - reduced_data_class=[X_train,Y_train];
20 - csvwrite('21GearSNE10.csv', reduced_data_class)
```

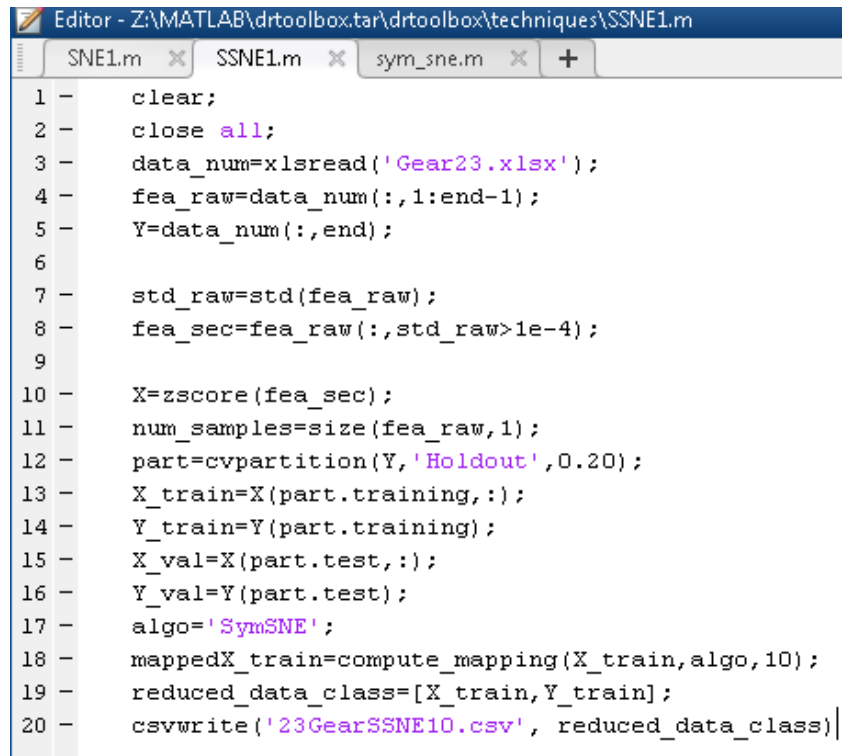
### Explanation:

- Load the database under a different file name and convert the database into array form.
- Separate the label from the data and store these two different data under different file names before computing the data.
- Perform normalization by removing redundant and zero columns from the datasets.
- After that compute the data using compute\_mapping function and pass required parameters which includes compute\_mapping (A, type, perp)
  - A -- Data on which reduction needs to be performed
  - type --- type of dimensionality method
  - perp ---- perplexity
- After that concatenate the reduced data with the labels and write reduced file with labels in csv to perform classification

**The parameter ‘algo’ will call the file “SNE” which is already provided in the toolbox and it performs the dimensionality reduction and give the reduced dataset.**



### c. Symmetric Stochastic Neighbor Embedding



```
1 - clear;
2 - close all;
3 - data_num=xlsread('Gear23.xlsx');
4 - fea_raw=data_num(:,1:end-1);
5 - Y=data_num(:,end);
6
7 - std_raw=std(fea_raw);
8 - fea_sec=fea_raw(:,std_raw>1e-4);
9
10 - X=zscore(fea_sec);
11 - num_samples=size(fea_raw,1);
12 - part=cvpartition(Y, 'Holdout', 0.20);
13 - X_train=X(part.training,:);
14 - Y_train=Y(part.training);
15 - X_val=X(part.test,:);
16 - Y_val=Y(part.test);
17 - algo='SymSNE';
18 - mappedX_train=compute_mapping(X_train,algo,10);
19 - reduced_data_class=[X_train,Y_train];
20 - csvwrite('23GearSSNE10.csv', reduced_data_class)
```

#### Explanation:

- Load the database under a different file name and convert the database into array form.
- Separate the label from the data and store these two different data under different file names before computing the data.
- Perform normalization by removing redundant and zero columns from the datasets.
- After that compute the data using compute\_mapping function and pass required parameters which includes compute\_mapping (A, type, perp)
  - A -- Data on which reduction needs to be performed
  - type --- type of dimensionality method
  - perp ---- perplexity
- After that concatenate the reduced data with the labels and write reduced file with labels in csv to perform classification

**The parameter ‘algo’ will call the file “SymSNE” which is already provided in the toolbox and it performs the dimensionality reduction and give the reduced dataset.**

## **6. Pipeline of the Project:**

In our project Dimensionality Reduction and Classification, we are assigned DR methods and Classification methods that we need to follow with the datasets provided. After reading datasets, data is split into training datasets (60%), testing datasets (20%) and validation datasets (20%). Then with the help of training data, DR model needs to be trained and then tuning DR model's parameters with Training and Validation sets is performed. After getting optimal parameters on the combination of Training/Validation sets, we obtain test dataset's classification performance. For reporting comparison and accuracy, we get classification performance on original dataset. This step is followed by the calculating the performance metrics of the DR methods.

In the second stage of the project, Classification is performed. Initially, nested cross validation strategy is performed for tuning classification methods' parameters. During this method, DR model is trained with optimal parameters obtained and original raw datasets being mapped to compressed features. The resultant of preceding step is used as input to classification method. For comparison, original data is used as input to the classification methods preceded by tuning of the classification methods' parameters by nested cross validation. Then we calculate accuracy and report f1 score in each nested cross validation process.

## To perform Classification without DR:

```
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier

data = pd.read_excel('Datasets/Gear1.xlsx')
# Dataset is now stored in a Pandas Dataframe
data = data.dropna(axis = 1, how = 'all')

#removing columns havin more than 90% zero
data=data.drop(columns=data.columns[((data==0).mean())>0.90]),axis=1)
print(data.info)

#saving tables in y
X_data= data.iloc[:, :-1].values
y_data=data.iloc[:, -1].values

#Normalization
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_data=sc.fit_transform(X_data)

#splitting data into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size=0.2)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=1)
from sklearn.model_selection import cross_val_score
model = DecisionTreeClassifier(random_state=0)

model.fit(X_train,y_train)
y_pred = model.predict(X_test)
import sklearn
from sklearn.model_selection import cross_val_score
Accuracy = cross_val_score(model, X_train, y_train, cv=10 , scoring="accuracy")
f1_score=cross_val_score(model, X_train, y_train, cv=10, scoring='f1_macro')
print('decision tree')

print (Accuracy)
print (f1_score)
```

In this code, we are importing required library and splitting the dataset into training and test dataset and fit the model with the train datasets and perform the accuracy and f-score on the original dataset for comparative results.

## To perform Classification with DR:

```
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
data=pd.read_csv('14GearSSNE10.csv',header=None)
X_data= data.iloc[:, :-1].values
y_data=data.iloc[:, -1].values
#Normalization
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_data=sc.fit_transform(X_data)
print(data)
#splitting data into train and test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size=0.2, random_state=0)
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
model = DecisionTreeClassifier()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
model1 = SVC(kernel= 'poly',gamma = .22,C = 100)
model1.fit(X_train,y_train)
classifier_knn=KNeighborsClassifier()
param_grid_knn = {
    'n_neighbors': [1],
    "metric": ["euclidean", "cityblock"], "algorithm" : ['auto', 'ball_tree', 'kd_tree', 'brute']
}

#tuning parameter of knn
grid_search1 = GridSearchCV(estimator = classifier_knn, param_grid=param_grid_knn, cv = 10, n_jobs = -1, verbose = 2)
grid_search1.fit(X_train,y_train)

optimal_classifier_knn=KNeighborsClassifier(**grid_search1.best_params_)
optimal_classifier_knn.fit(X_train, y_train)

from sklearn.model_selection import cross_val_score
Accuracy = cross_val_score(model, X_train, y_train, cv=10 , scoring="accuracy")
f1_score=cross_val_score(model, X_train, y_train, cv=10, scoring='f1_macro')
Accuracy1 = cross_val_score(model1, X_train, y_train, cv=10 , scoring="accuracy")
f1_score1=cross_val_score(model1, X_train, y_train, cv=10, scoring='f1_macro')
Accuracy2 = cross_val_score(optimal_classifier_knn, X_train, y_train, cv=10 , scoring="accuracy")
f1_score2=cross_val_score(optimal_classifier_knn, X_train, y_train, cv=10, scoring='f1_macro')
print('decision tree')
print ('Accuracy',Accuracy)
print ('F-score' ,f1_score)
print('SVM')
print ('Accuracy',Accuracy1)
print ('F-score',f1_score1)
print('KNN')
print ('Accuracy',Accuracy2)
print ('F-score',f1_score2)
```

In this code, we are importing required library of the classification methods and splitting the dataset into training and test dataset and fit all the classification models model with the train datasets and perform the accuracy and f-score on the reduced datasets.

## Snapshot of the Performance Metrics for Gear1:

Data Set: Gear 1													
Couse Project	DR19			Without Dimentionality Reduction			With Dimentionality Reduction						
DR Methods	GPLVM	SNE	SSNE	Classifier	F-Measure	Accuracy	Classifier	F-Measure			Classifier	Accuracy	
# of Compon ents	20	10	10	KNN			KNN	GPLVM	SNE	SSNE	KNN	GPLVM	SSNE
				fold1	0.858924883	0.867867868	fold1	0.8530182	0.755522	0.8083391	fold1	0.8506024	0.7731343
				fold2	0.820628045	0.819277108	fold2	0.8477455	0.8069138	0.8009299	fold2	0.8759124	0.8090909
				fold3	0.856685003	0.859756098	fold3	0.8884141	0.7784724	0.8382292	fold3	0.8916256	0.8024316
				fold4	0.883910689	0.888544892	fold4	0.8375881	0.7091937	0.8737954	fold4	0.866005	0.7737003
				fold5	0.811421941	0.852201258	fold5	0.8442027	0.7802212	0.8220932	fold5	0.8787879	0.8043478
				fold6	0.829625954	0.85443038	fold6	0.8342532	0.7559584	0.8468987	fold6	0.8451777	0.7886435
				fold7	0.854935425	0.865814696	fold7	0.8661082	0.8023723	0.8655572	fold7	0.8772379	0.8178914
				fold8	0.84026726	0.87987013	fold8	0.8720864	0.7487256	0.8821484	fold8	0.8863049	0.7810458
				fold9	0.83958453	0.862745098	fold9	0.8152699	0.7651184	0.8688997	fold9	0.8264249	0.8139535
				fold10	0.833300064	0.846666667	fold10	0.8870504	0.8300881	0.8532642	fold10	0.8929504	0.8288591
				SVM			SVM	GPLVM	SNE	SSNE	SVM	GPLVM	SSNE
				fold1	0.85	0.86	fold1	0.8424305	0.7764274	0.8248738	fold1	0.8409639	0.7850746
				fold2	0.86	0.87	fold2	0.8138329	0.7812156	0.8525327	fold2	0.8491484	0.7787879
				fold3	0.91	0.91	fold3	0.868079	0.7900848	0.8795177	fold3	0.8694581	0.8085106
				fold4	0.81	0.84	fold4	0.805191	0.7303969	0.8494609	fold4	0.8387097	0.7737003
				fold5	0.84	0.87	fold5	0.8300207	0.7539681	0.8546133	fold5	0.8585859	0.7701863
				fold6	0.86	0.88	fold6	0.7961999	0.7267095	0.8422015	fold6	0.819797	0.7728707
				fold7	0.89	0.91	fold7	0.7790131	0.7801747	0.858764	fold7	0.8209719	0.7955272
				fold8	0.89	0.89	fold8	0.8244225	0.719566	0.8503524	fold8	0.8475452	0.7614379
				fold9	0.8	0.82	fold9	0.7647037	0.7629648	0.8172459	fold9	0.7927461	0.807309
				fold10	0.88	0.89	fold10	0.8264876	0.8121054	0.8540579	fold10	0.843342	0.8020134
				DT			DT	GPLVM	SNE	SSNE	DT	GPLVM	SSNE
				fold1	0.78	0.8	fold1	0.7688194	0.6057274	0.8072928	fold1	0.7951807	0.6268657
				fold2	0.74	0.8	fold2	0.7483616	0.6538734	0.7439725	fold2	0.7858881	0.6545455
				fold3	0.76	0.78	fold3	0.769907	0.61581	0.7522554	fold3	0.7807882	0.6474164
				fold4	0.74	0.76	fold4	0.7079168	0.5370002	0.8363842	fold4	0.7220844	0.6055046
				fold5	0.79	0.82	fold5	0.7591612	0.61056	0.748039	fold5	0.770202	0.6242236
				fold6	0.72	0.76	fold6	0.7004215	0.5862306	0.7186775	fold6	0.7335025	0.6246057
				fold7	0.7	0.74	fold7	0.7649675	0.6342856	0.7880475	fold7	0.7826087	0.6645367
				fold8	0.73	0.74	fold8	0.7753598	0.60711	0.7643169	fold8	0.7932817	0.6732026
				fold9	0.79	0.79	fold9	0.6933266	0.6055412	0.7538204	fold9	0.7202073	0.6777409
				fold10	0.74	0.76	fold10	0.8246498	0.6125062	0.7927145	fold10	0.8355091	0.647651

The following table shows the accuracy and F-Measure for the dataset Gear 1 for three classification methods KNN, Support Vector Machine (SVM) and Decision Tree (DT) without Dimensionality Reduction Method and with Dimensionality Reduction methods. In this figure the dimensionality reduction methods used are Gaussian Process Latent Variable Method (GPLVM), Stochastic Neighbor Embedding (SNE) and Symmetric Stochastic Neighbor Embedding (SSNE) for 10 folds.

## 7. Performance Evaluation:

To evaluate the performance of the algorithm, we rely on accuracy and f1-score which is obtained by confusion or error matrix. Confusion 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 row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa).

### Accuracy:

accuracy (ACC)

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN}$$

### F1-Score:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

---

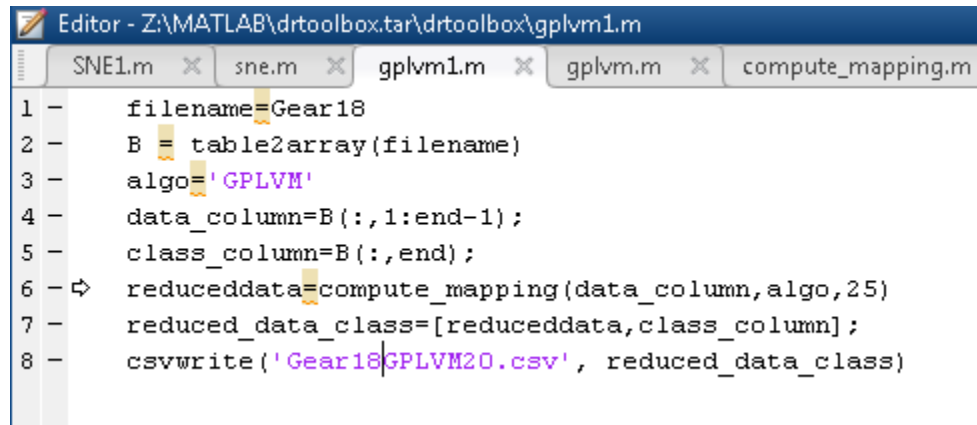
**TP-** True Positive

**TN-** True Negative

**FP-** False Positive

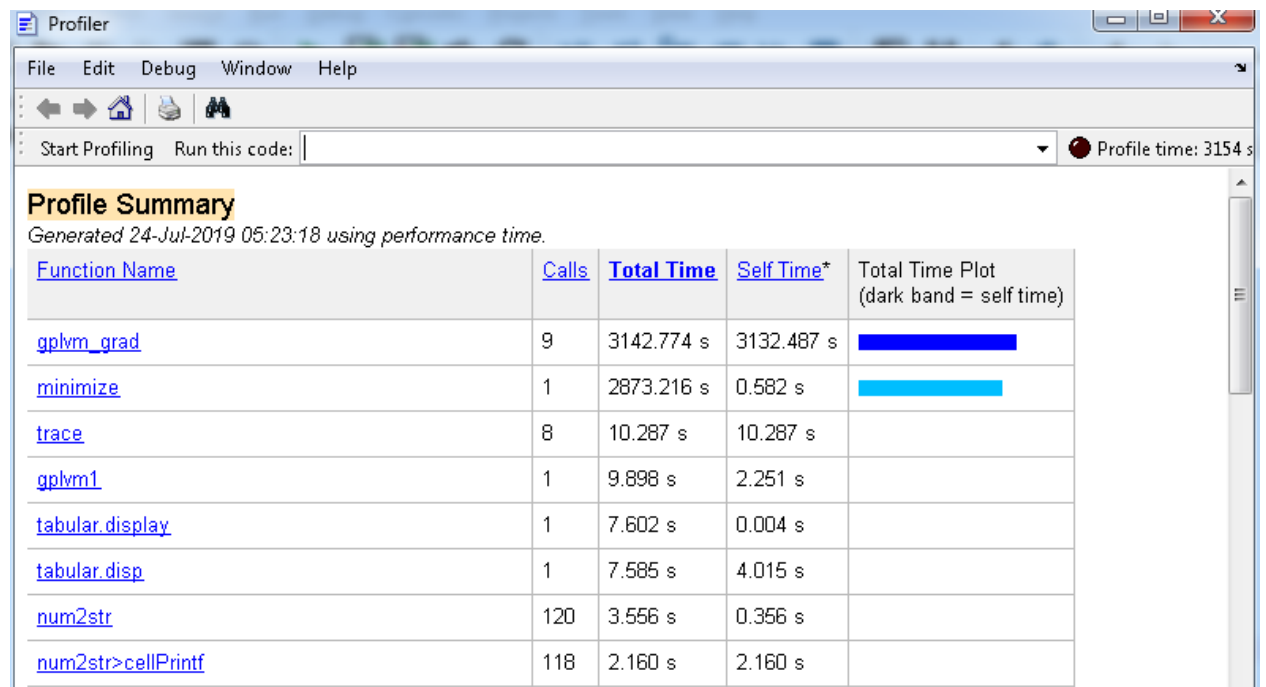
**FN-** False Negative

## Special Case Gear 18:





```
1 - filename='Gear18'
2 - B = table2array(filename)
3 - algo='GPLVM'
4 - data_column=B(:,1:end-1);
5 - class_column=B(:,end);
6 - reduceddata=compute_mapping(data_column,algo,25)
7 - reduced_data_class=[reduceddata,class_column];
8 - csvwrite('Gear18GPLVM20.csv', reduced_data_class)
```

When we are performing Dimensionality reduction method GPLVM on Gear 18, nothing is imputed as it takes long time to reduce the time.



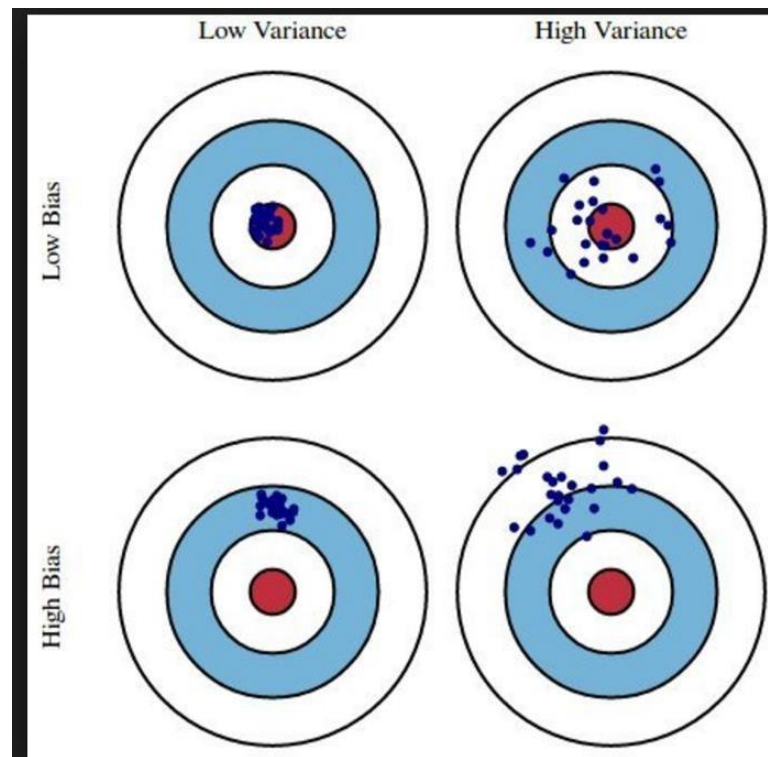
**Profile Summary**  
Generated 24-Jul-2019 05:23:18 using performance time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
<a href="#">gplvm_grad</a>	9	3142.774 s	3132.487 s	
<a href="#">minimize</a>	1	2873.216 s	0.582 s	
<a href="#">trace</a>	8	10.287 s	10.287 s	
<a href="#">gplvm1</a>	1	9.898 s	2.251 s	
<a href="#">tabular.display</a>	1	7.602 s	0.004 s	
<a href="#">tabular disp</a>	1	7.585 s	4.015 s	
<a href="#">num2str</a>	120	3.556 s	0.356 s	
<a href="#">num2str&gt;cellPrintf</a>	118	2.160 s	2.160 s	

So, nothing is being obtained in this case for GPLVM as the dataset Gear 18 is very heavy to impute the results.

## 8. Calculation of Variability:

Variance is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).”



## Formula to calculate Variability:

$$\begin{array}{ccc} & \text{DR} & \\ X_{\text{original}} & \xrightarrow{\quad} & \bar{X} \\ (M \times N) & & (M \times P) \quad P \leq N \end{array}$$

→  $\text{COV} = \text{COV}(\bar{X}) = \bar{X}^T \bar{X}$  (COV is  $P \times P$ )

→  $[\lambda_1, \lambda_2, \dots, \lambda_p] = \text{eigenvalue}(\bar{X}^T \bar{X})$

→ Variability explained by  $i$ th component:

$$\frac{\lambda_i}{\text{sum}(\lambda_1, \lambda_2, \dots, \lambda_p)}$$



## To perform Variability for Reduced Data Dimensionality Methods:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
Components = 30
dff = pd.read_csv('Gear18SNE30.csv')
dff = dff.dropna(axis = 1, how = 'all')
#removing columns havin more than 90% zero
dff=dff.drop(columns=dff.columns[((dff==0).mean())>0.90]),axis=1)
X = dff.iloc[:, :-1].values
y = dff.iloc[:, -1].values
from scipy.stats import variation
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_t = X_train.transpose()
COV = np.matmul(X_t,X_train)
from scipy import linalg as LA
e_vals, e_vecs = LA.eig(COV)
i=1
for item in e_vals:
    Variability=item/sum(e_vals)

    print(" " + str(i) + " - " + str(Variability)[1:8])
    i = i + 1
```

## **Output:**

Variability Explained					
GPLVM		SNE		SSNE	
1 - 0.04038	21-	1 - 0.18630	21-	1 - 0.16820	21-
2 - 0.05884	22-	2 - 0.15601	22-	2 - 0.12646	22-
3 - 0.05754	23-	3 - 0.03804	23-	3 - 0.10617	23-
4 - 0.04407	24-	4 - 0.13055	24-	4 - 0.07478	24-
5 - 0.04482	25-	5 - 0.11239	25-	5 - 0.07234	25-
6 - 0.05492	26-	6 - 0.10051	26-	6 - 0.05690	26-
7 - 0.05381	27-	7 - 0.05244	27-	7 - 0.04556	27-
8 - 0.04706	28-	8 - 0.06479	28-	8 - 0.03219	28-
9 - 0.04764	29-	9 - 0.08045	29-	9 - 0.02472	29-
10 - 0.05211	30-	10 - 0.07847	30-	10 - 0.02163	30-
11 - 0.05159	31-	11-	31-	11-	31-
12 - 0.05134	32-	12-	32-	12-	32-
13 - 0.05053	33-	13-	33-	13-	33-
14 - 0.05044	34-	14-	34-	14-	34-
15 - 0.04991	35-	15-	35-	15-	35-
16 - 0.04852	36-	16-	36-	16-	36-
17 - 0.04871	37-	17-	37-	17-	37-
18 - 0.04944	38-	18-	38-	18-	38-
19 - 0.04925	39-	19-	39-	19-	39-
20 - 0.04898	40-	20-	40-	20-	40-

For 3 of the dimensionality methods for Gear1 variability has been explained:

## 9. Computational Complexities:

- GPLVM's complexity is  $O(N^3)$
- Stochastic Neighbor Embedding and Symmetric Stochastic Neighbor Embedding complexity is  $O(N^2)$
- Naive Bayes complexity =  $O(N.d)$
- Support Vector Machine =  $O(N^3)$
- Decision Tree complexity is  $O(N^2.d)$
- Recurrent Neural Network complexity is  $O(N \log N)$ 
  - where  $N$  is the number of data points
  - $d$  is dimensionality of features

## **10. Conclusions: -**

By analyzing the above results, we made following conclusions: -

- Support Vector Machine has the best results with GPLVM, followed by SNE and Symmetric SNE.
- Decision Tree has the best results with the GPLVM, followed by SNE and SymSNE.
- Even Naïve Bayes has the same criteria, it gives best result with GPLVM then by SNE and SymSNE even though the performance metrics given by them are very low.

## **11. References:**

- <https://www.datacamp.com/community/tutorials/introduction-t-sne>
- [https://www.researchgate.net/publication/310387112\\_A\\_Review\\_on\\_Gaussian\\_Process\\_Latent\\_Variable\\_Models](https://www.researchgate.net/publication/310387112_A_Review_on_Gaussian_Process_Latent_Variable_Models)
- [https://cs.nyu.edu/~roweis/papers/sne\\_final.pdf](https://cs.nyu.edu/~roweis/papers/sne_final.pdf)
- <https://pdfs.semanticscholar.org/1c46/943103bd7b7a2c7be86859995a4144d1938b.pdf>
- <http://www.sciencedirect.com/science/article/pii/S2468232216300828>



