Machine learning and pattern recognition approaches on Prostate cancer data to discover meaningful biomarkers

RUTURAJ R. RAVAL
raval115@uwindsor.ca
University of Windsor

1. Abstract

A machine learning approach to identify meaningful biomarkers by recognizing the possible patterns in the genes data given from the cBioPortal [1] platform for the prostate cancer data. By having 495 different samples (patients) with 60K+ genes combinations, classification and feature selection, solving multi-class problems, dimensionality reduction, etc. techniques will be applied using clinical dataset along the genes dataset. The visualization will be applied to follow up using seaborn and matplotlib libraries to bifurcate distinguished features using a different kind of maps and graphs. Right estimator algorithms are applied to test the accuracy to avoid the overfitting. Different classifiers are applied with the standard procedures and the accuracy is measured using cross-validation. The classifiers used in this scenario are *Stochastic Gradient Descent* (SGD), *Support Vector Machine* (SVM), *Nearest Neighbours* (NN), Naïve Bayes, Forest and tree methods. Hence, previously selected features using feature selection methods will be compared with the complete set of datasets. Each estimator has a different set of hyper-parameters and using a different set of hyper-parameters combinations using grid search approach over best performing classifier model.

2. Introduction

The process of finding the meaningful biomarkers from huge dataset of patients suffering from prostate cancer, involves many different steps in the pipeline during the process, starting from the classification, feature selection, regression, dimensionality reduction, etc. [2] approaches of the machine learning and the pattern recognition, to help discovering the important patterns in an existing set of data. Firstly, I have used 11 different classifiers like *k-Nearest Neighbours* (k-NN), *Naïve Bayes* (NB), *Support Vector Machine* (SVM): Linear kernel, SVM: Polynomial kernel, SVM: RBF kernel, *Stochastic Gradient Descent* (SGD), *Support Vector Classification* (SVC), *Nu SVC* (NuSVC), *Random Forest* (RF), *Extra Trees* (ET), *Decision Trees* (DT), etc. to learn the behaviour of the genes dataset with different classifiers, by implementing *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), *False Negative* (FN), etc. to find the performance measures like *Positive Predicted Value* (PPV), *Negative Predicted Value* (NPV), Specificity, Sensitivity, Accuracy, etc. as shown in the Table 1 below.

Table 1 Performance measures based on TP, TN, FP, FN for 11 different classifiers

TP, TN, FP, FN based performance measures					
Classifiers	***Performance measures***				
	PPV	***NPV***	***Specificity***	***Sensitivity***	***Accuracy***
k-NN	0.6057	0.8266	0.4644	0.8955	0.6718
Naïve Bayes	1.0	1.0	1.0	1.0	1.0
SVM- Linear	0.7593	0.7701	0.7738	0.7457	0.7612
SVM- Poly	0.8730	0.6150	0.9457	0.3556	0.6619
SVM- RBF	0.8216	0.7526	0.8632	0.6974	0.7815
SGD	0.7469	0.7523	0.7764	0.7110	0.7490
SVC	0.7890	0.7448	0.8351	0.6878	0.7631
NuSVC	0.7898	0.7519	0.8274	0.6988	0.7629
Random Forest	0.7033	0.7538	0.7114	0.7377	0.7206
Extra trees	0.7070	0.7447	0.7077	0.7428	0.7166
Decision trees	1.0	1.0	1.0	1.0	1.0

3. Classification on dataset

The **k-NN (k-Nearest Neighbors)** classifiers [3] are kind of a non-parametric method used for classification. The implementation of k-NN is an unsupervised nearest neighbours learning which is implemented from NearestNeighbors library. k-NN is an instance-based learning. In k-NN, the k is the number of the nearest neighbours, the number of neighbours is the core factor in a decision-making process. If the class number is 2 then k will be odd. Here, when k is 1 then the algorithm is known as the nearest neighbour (1-NN). The **Naïve Bayes** classifiers [3] are kind of probabilistic classifiers which represents the assumptions between the features which are independent based on Bayes' theorem in the field of machine learning. Simple Bayes is another name for Naïve Bayes which performs in the direction of the classifier's decision rules. Naïve Bayes methods are set of supervised learning. **SVM (Support Vector machine)** [5] is a most widely used technique for the classification. SVM is suitable for small training datasets and it works well with a large number of features and has a good amount of generalization power. It is a supervised learning model with the labelled data. The error of the SVM depends on the number of support vectors only and the expectation is distributed over all training sets. Selection of the kernel depends on the problem domain. The tuning parameters for kernels are usually done through normalization including the grid search and cross-validation. Normalization is done through scaling data to [0,1] or [-1,1]. The grid search depends on the values of c, γ (gamma), d (degree). These parameters are data dependent which helps in producing approximate the same solution for certain types of data. **SVM-L** [6] is considered to be a part of the Linear classification when some functions are not linear on x but are linear on y, we need a linear function in a higher-dimensional space known as y-space, which works as a feature space. This kernel works better than the non-linear classifiers. Value of c usually ranges from 1 to 10^6. Which uses logarithmic increments. **SVM-P** [7] is considered to be a part of the Non-Linear classification, that is a general form of a Bayesian classifier with normal distributions generalized to be a polynomial discriminant function. Value of d, degree for polynomial varies from 2 to 8. **SVM-R** [7] is considered to be a part of the Non-Linear classification, it is also called the Gaussian kernel. Value of c, usually ranges from 1 to 106. Which uses logarithmic increments. The RBF γ, gamma value ranges between 0.01 to 2^8.

Table 2 Accuracy and Cross-Validation score discovery using 11 different classifiers

GENES data	***Results based on 10 sample data***				***Results based on the whole dataset***			
Classifier	***CV score %***	***CV (+/-) %***	***Execution time***	***Accuracy %***	***CV score %***	***CV (+/-) %***	***Execution time***	***Accuracy %***
SGD	53.84	10.25	0.073 s	59.59	58.87	17.72	3.1187 s	77.78
SVC	60.75	17.61	0.066 s	69.70	65.37	13.65	33.962 s	78.79
NuSVC	60.71	13.34	0.081 s	69.70	66.20	14.51	37.04 s	78.79
Linear SVC	52.64	17.96	0.24 s	65.66	62.92	14.22	13.89 s	72.73
Poly SVC (degree= 2)	58.08	5.75	0.0595 s	65.66	60.91	5.72	37.396 s	69.70
RBF SVC	52.02	0.04	0.533 s	49.49	52.02	0.04	160.91 s	49.49
k-NN (k = 1)	60.73	12.50	0.024 s	63.64	59.51	0.80	14.499 s	67.68
Naïve Bayes	53.04	11.69	0.029 s	68.69	59.65	16.57	4.1606 s	68.69
Random Forest	57.44	20.17	0.171 s	69.70	59.70	9.66	4.3777 s	71.72
Extra Trees	57.69	16.04	0.141 s	63.64	59.68	20.95	3.3553 s	76.77
Decision Tree	53.64	6.60	0.042 s	65.66	59.35	8.78	43.479 s	70.71

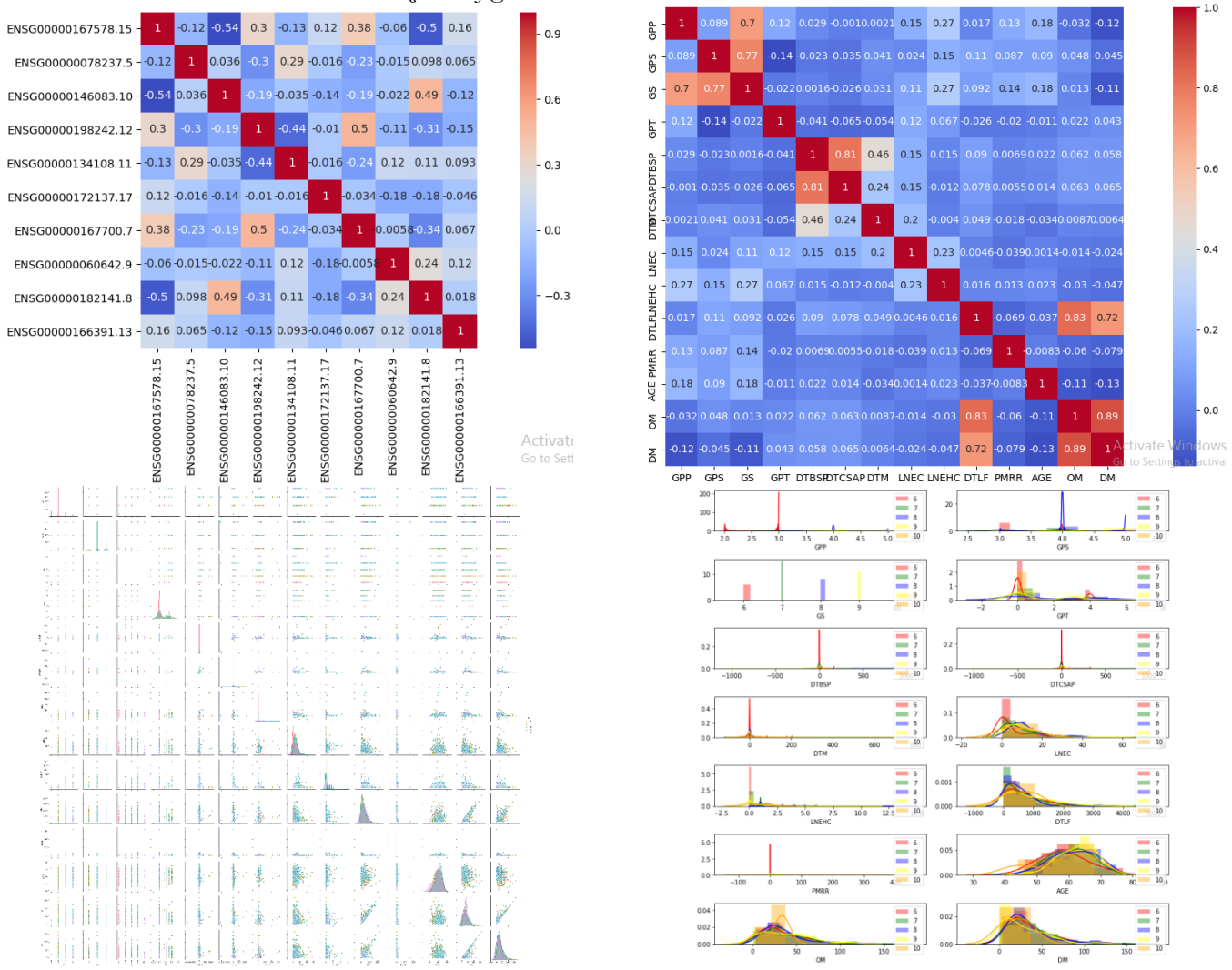
Nu-Support Vector Classification (NuSVC) [6], similar to SVC but uses a parameter to control the number of support vectors. **Stochastic Gradient Descent (SGD)** [10] is a simple, yet very efficient approach to discriminative learning of linear classifiers under convex loss function s such as Linear SVMs and logistic regression. SGDs are very efficient and offers opportunities for code tuning. SGD is sensitive to feature scaling and it requires a number of hyperparameters such as regularization parameters and number of iterations which is a disadvantage. The **Random Forest classifier** [11], is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default). The **Extra Trees classifier** [11] class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. **Decision Trees (DTs)** [13] are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

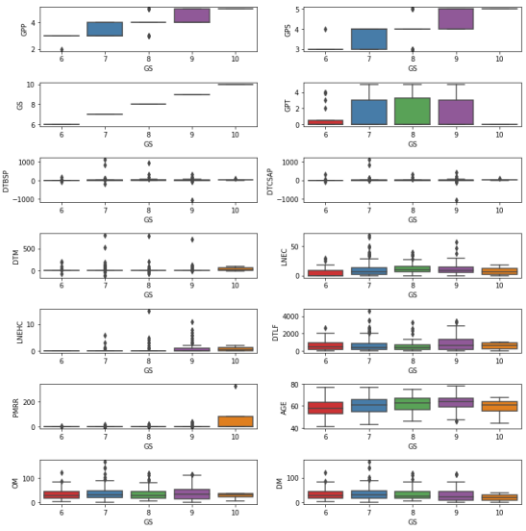
As shown in Table 2 accuracy and cross-validation score can be found with +/- cross-validation difference. As you can see in Table 1 and Table 2 respectively, Naïve Bayes and Decision Trees classifiers give the higher accuracy; SVC and NuSVC and Random Forest classifiers for 10 sample data and SVC and NuSVC gives higher accuracy than other classifiers.

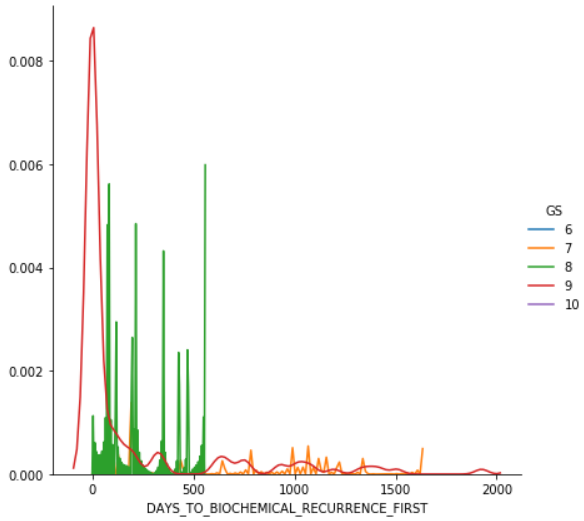
4. Visualization

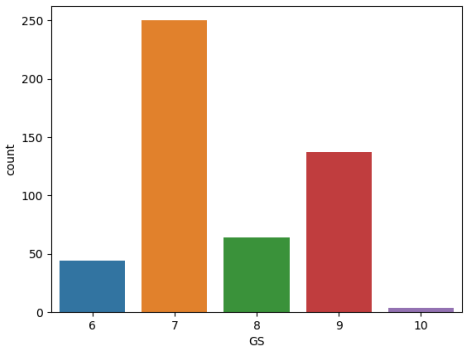
Visualization can be developed based on the dataset we have to understand and to glorify the best features and highlight important information.

Table 3 Visualization of genes_data and clinical_data to understand data well

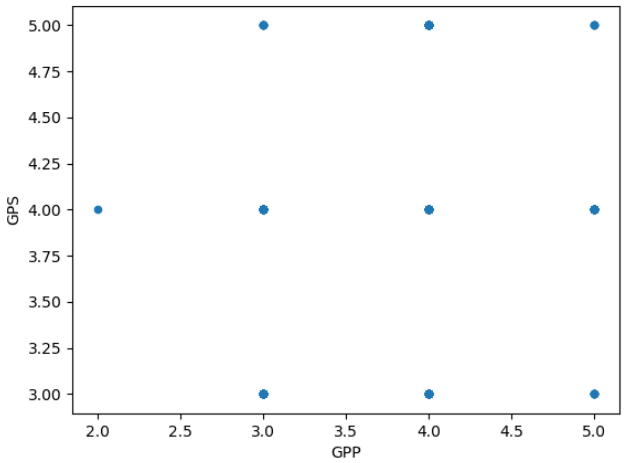


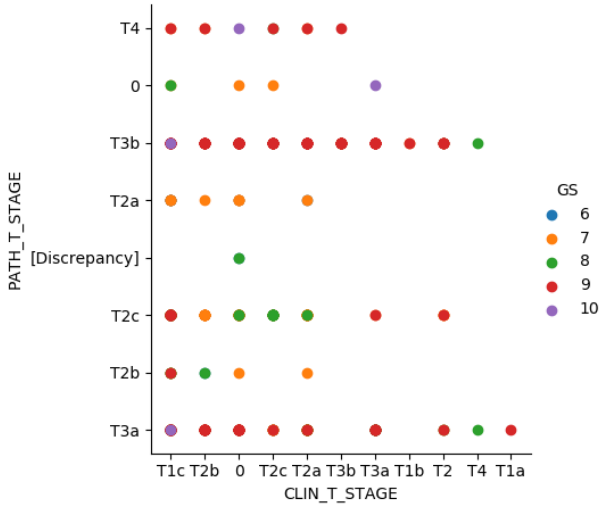

GPP
GPS
GS
GPT
DTBSP
DTCSAP
DTM
LNEC
LNEHC
DTLF
PMRR
AGE
OM
DM

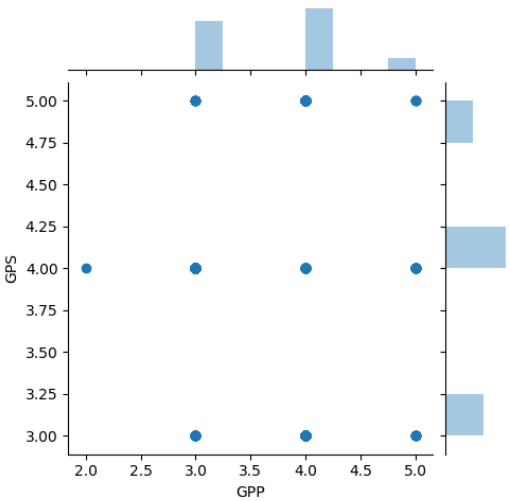

0.008
0.006
0.004
0.002
0.000
0
500
1000
1500
2000
DAYS_TO_BIOCHEMICAL_RECURRENCE_FIRST
GS
6
7
8
9
10

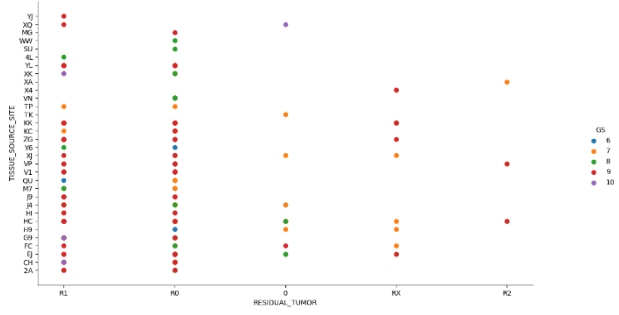

count
GS
6
7
8
9
10

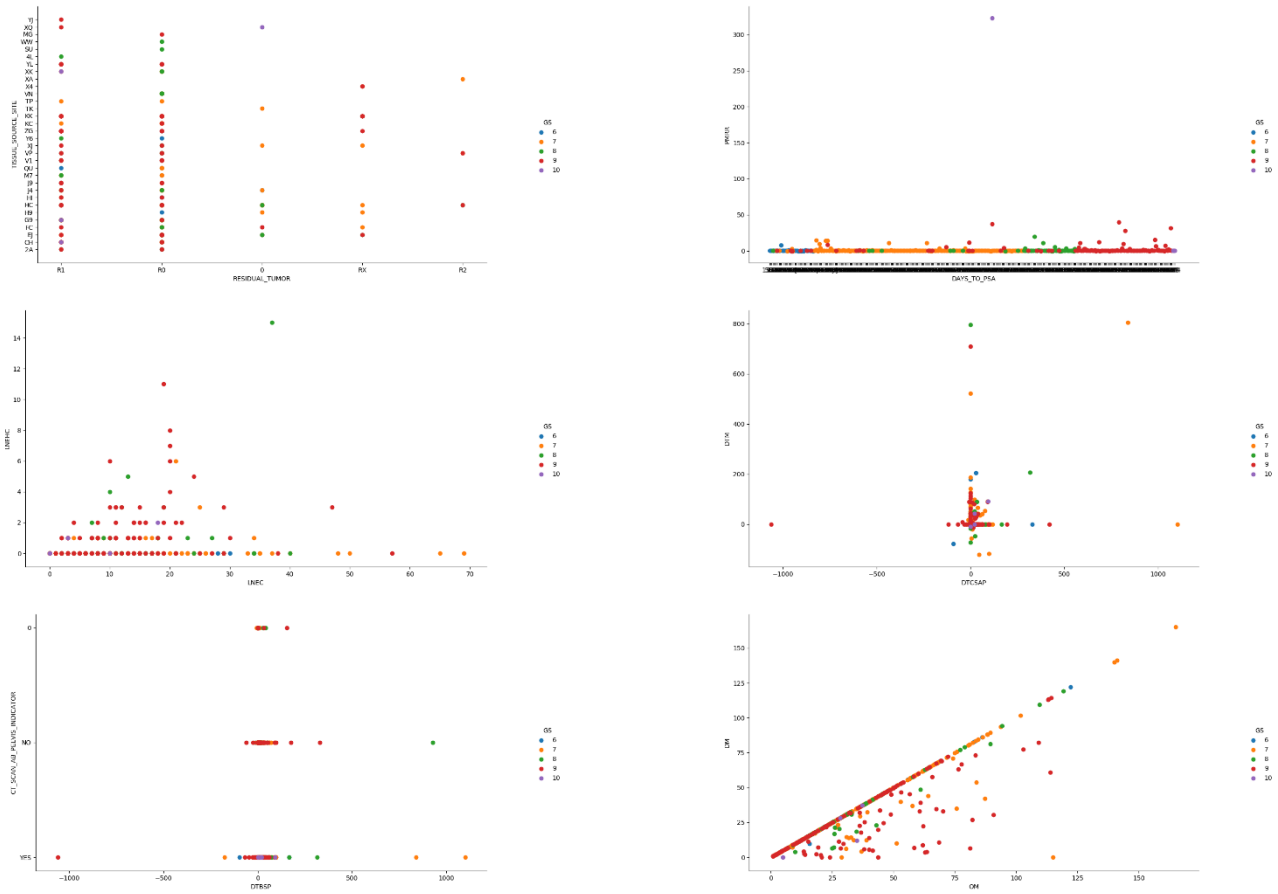

GPP
GPS
GS
GPT
DTBSP
DTCSAP
DTM
LNEC
LNEHC
DTLF
PMRR
AGE
OM
DM


GPS
GPP
5.00
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00
2.0
2.5
3.0
3.5
4.0
4.5
5.0


PATH_T_STAGE
T4
0
T3b
T2a
[Discrepancy]
T2c
T2b
T3a
CLIN_T_STAGE
T1c T2b 0 T2c T2a T3b T3a T1b T2 T4 T1a
GS
6
7
8
9
10


GPS
GPP
5.00
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00
2.0
2.5
3.0
3.5
4.0
4.5
5.0


TISSUE_SOURCE_SITE
RESIDUAL_TUMOR
R1
R0
0
RX
R2
GS
6
7
8
9
10



The prostate cancer patients dataset has two sets, genes_dataset and clinical_dataset based on patients and visualization can be shown as Table 3 above.

These visualization images represents the structure of two datasets as mentioned, to enable myself for the scope of these datasets to learn and understand how data flows how can I perform the feature selection task and achieve the goal of discovering the meaningful biomarkers from given set of data. The specification over the abbreviations used in all the visualization graphs are as given below, to express visualization of long column names easily.

gleason_pattern_primary (gpp), *gleason_pattern_secondary* (gps), *gleason_score* (gs), *gleason_pattern_tertiary* (gpt), *days_to_bone_scan_performed* (dtbsp), *days_to_ct_scan_ab_pelvis* (dtcsap), *days_to_mri* (dtm), *lymph_node_examined_count* (lnec), *days_to_last_followup* (dtlf), *days_to_psa* (dtp), *psa_most_recent_results* (pmrr), *os_months* (om), *dfs_months* (dm).

5. The literature on Cancer data research

There has been a two-phase search strategy proposed to identify the biomarkers in gene expression dataset for the prostate cancer diagnosis using statistical filtering method employed to remove the noisiest data. The first phase includes multi-objective optimization based on binary particle swarm optimization algorithm tuned by the chaotic method and in the second-phase search strategy involves cache-based modification of the *Sequential Forward Floating Selection* (SFFS) algorithm, to find the discriminant genes. [14] My approach which I have used, SFS doesn't use the floating step as the floating algorithms have an additional exclusion or inclusion step to remove features once they were included (or excluded) so that a larger number of feature subset combinations can be sampled. The explanation made in this paper [15] represents the different steps of learning procedure which includes data pre-processing, feature selection and classification, etc. having meaningful approaches to gain the desired results by applying state-of-the-art approaches over the genes data which shows a comparison between different algorithms and how they are outperformed by the results. The other work based on Fuzzy Support Vector Machine [16] is an efficient rule-based classification technique. Very good amount of work [17] is carried out in the field of cancer research including supervised prostate cancer classification

[18], which can be a milestone in the cancer research area. The combinatorial optimization approach [18] for integrated feature selection is useful while working with the multiple datasets and it is useful in transcriptomics.

6. Feature selection

It is possible to automatically select those features in your data that are most useful or most relevant for the problem you are working on. Feature Selection is a very critical component in a Data Scientist's workflow. When presented data with very high dimensionality, models usually choke because **Training time** increases exponentially with a number of features; Models have an increasing risk of **overfitting** with an increasing number of features. Feature Selection methods help with these problems by reducing the dimensions without much loss of the total information. It also helps to make sense of the features and its importance. [14] This is a process called **feature selection**. Feature selection is also called variable selection or attribute selection. It is the automatic selection of attributes in your data (such as columns in tabular data) that are most relevant to the predictive modelling problem you are working on. [2] Feature selection is itself useful, but it mostly acts as a filter, muting out features that aren't useful in addition to your existing features. The objective of variable selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data. [3] There are three general classes of feature selection algorithms: *filter methods*, *wrapper methods* and *embedded methods*. [2] **Filter feature selection methods** apply a statistical measure to assign a scoring to each feature. The features are ranked by the score and either selected to be kept or removed from the dataset. The methods are often univariate and consider the feature independently, or with regard to the dependent variable. Some examples of some filter methods include the Chi-squared test, information gain and correlation coefficient scores. **Wrapper methods** consider the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. A predictive model us used to evaluate a combination of features and assign a score based on model accuracy. The search process may be methodical such as a best-first search, it may stochastic such as a random hill-climbing algorithm, or it may use heuristics, like forward and backward passes to add and remove features. An example if a wrapper method is the recursive feature elimination algorithm. **Embedded methods** learn which features best contribute to the accuracy of the model while the model is being created. The most common type of embedded feature selection methods is regularization methods. Regularization methods are also called penalization methods that introduce additional constraints into the optimization of a predictive algorithm (such as a regression algorithm) that bias the model toward lower complexity (fewer coefficients). Examples of regularization algorithms are the LASSO, Elastic Net and Ridge Regression. While considering the aspects of feature selection, **wrapper methods** consider the feature selection process for the type of model, which is being built, to evaluate feature subsets to detect the model performance between the features to select the best performing subset, subsequently. [4] To perform the feature selection, I have used the *Sequential Feature Selector* (SFS). The SFS algorithm [17] can be explained algorithmically, as described below. Let's say, *Input Y*, *Output* X_k, Initialization $X_0 = \varphi$, $k = 0$. $Y = \{y_1, y_2, \ldots, y_d\}$, the SFS algorithm takes the whole d-dimensional feature set as input. SFS returns a subset of features, the selected number of features k, where $k < d$, has to be a priori representing $X_k = \{x_j \mid j = 1, 2, \ldots, k; x_j \in Y\}$, where $k = (0, 1, 2, \ldots, d)$. After the initialization with an empty set φ, so that $k = 0$, where is k is the size of the subset.

Inclusion step:

i. $x^+ = arg\ max\ J(x_k + x)$, *where* $x \in Y - X_k$

ii. $X_{k+1} = X_k + x^+$

iii. $k = k + 1$

iv. *Go to step 1*

- Now add an additional feature, x^+, the feature subset X_k
- x^+ is the feature that maximizes the criterion function, that is the feature associated with the best classifier performance if added to X_k
- Repeat until the termination criterion is satisfied.

Termination $k = p$

- Add the features from the feature subset X_k until the feature subset of size k contains the number of desired features p that specifies the priori.

As shown below in Table 4 the accuracy is calculated and the best features are selected, based on training and testing data, calculating total duration and selecting features in total.

Table 4 Feature selection and discovery of accuracy of training and testing data and accuracy

Features selection	***Training data***			***Testing data***			***Average feature score of 5 features***	***5 best features in this specified range of dataset***	***Total taken time***
Dataset range	***Dataset shape***	***Accuracy on selected features***	***Accuracy on all features***	***Dataset shape***	***Accuracy on selected features***	***Accuracy on all features***			
1-1000	(395, 999) (395,)	0.800	0.919	(99, 999) (99,)	0.737	0.768	0.71	[206, 483, 861]	77 m 10 s
1000-2000	(395, 1000) (395,)	0.813	0.924	(99, 1000) (99,)	0.768	0.778	0.73	[3, 99, 899]	83 m 50 s
2000-3000		0.82	0.924		0.778	0.798	0.73	[46, 64, 707]	103 m 20 s
3000-4000		0.795	0.911		0.768	0.808	0.72	[39, 401, 999]	166 m 20 s
4000-5000		0.815	0.916		0.717	0.788	0.75	[190, 408, 940]	167 m
5000-6000		0.835	0.901		0.737	0.798	0.71	[83, 669, 993]	93 m 50 s
6000-7000		0.843	0.919		0.768	0.788	0.74	[218, 600, 800]	170 m
7000-8000		0.803	0.924		0.697	0.798	0.72	[44, 194, 664]	111 m 50 s
8000-9000		0.82	0.919		0.687	0.788	0.71	[5, 200, 846]	173 m 30 s
9000-10000		0.823	0.919		0.727	0.798	0.71	[31, 78, 661]	105 m 20 s
10000-11000		0.813	0.919		0.687	0.788	0.71	[217, 617, 997]	166 m 10 s
11000-12000		0.815	0.942		0.717	0.808	0.71	[431, 512, 997]	110 m 50 s
12000-13000		0.828	0.916		0.778	0.768	0.73	[102, 105, 890]	168 m 50 s
13000-14000		0.825	0.914		0.697	0.778	0.72	[3, 43, 692]	112 m 50 s
14000-15000		0.825	0.934		0.707	0.778	0.73	[60, 365, 908]	113 m 10 s
15000-16000		0.828	0.919		0.717	0.778	0.73	[261, 289, 565]	168 m 40

									s
16000-17000		0.856	0.939		0.778	0.768	0.74	[10, 92, 982]	92 m 40 s
17000-18000		0.815	0.911		0.737	0.778	0.72	[100, 328, 805]	63 m 10 s
18000-19000		0.805	0.919		0.778	0.788	0.73	[10, 33, 994]	145 m 10 s
19000-20000		0.81	0.924		0.707	0.798	0.72	[12, 471, 730]	171 m 30 s
20000-21000		0.846	0.914		0.758	0.778	0.74	[133, 158, 991]	172 m 20 s
21000-21529	(395, 529) (395,)	0.841	0.932	(99, 529) (99,)	0.697	0.788	0.73	[89, 105, 386]	88 m 30 s
Overall 21529	***(395, 21529) (395,)***	***0.82***	***0.92***	***(99, 21529) (99,)***	***0.73***	***0.79***	***0.72***	***66 features***	***2826 m (47 h 10 m)***

For the SFS implementation, first, split the dataset into training and testing sets. Then after defining the classifier, pass the classifier as a feature selector, then define the subset of features which is *k_features = 5*, in this case. The scoring measures are set to accuracy, which is a metric, which can be used to score the resulting models built on the selected features. Feature selection helps in increasing the interpretability of the model; reduces the complexity of the model; reduce the training time of the model. [19]

As shown in Table 4 overall training accuracy over selected features is 82% and 92% for the whole dataset, whereas the accuracy over testing selected features is 73% and on whole dataset, it is 79% concluding the average feature score at 72% over 110 features. Firstly, I removed all the zero-ed rows from the dataset and removed the rows with more number of zeros then I get 21529 samples of genes and out of which I have selected 66 features to classify again to calculate features accuracy using previously used 11 classifiers as per previous chapters.

7. Classification after feature selection on the dataset

I applied the same classifiers after the feature selection and it is visible that, accuracy has improved in most of the cases after comparing the results of Table 2 and Table 5.

Table 5 Classification result on selected features

Classifier	***CV score %***	***CV (+/-) %***	***Execution time***	***Accuracy %***
SGD	55.48	11	0.092 s	68.69
SVC	61.53	17.31	0.19 s	80.81
NuSVC	60.32	14.31	0.23 s	80.81
Linear SVC	58.51	21.2	0.68 s	68.69
Poly SVC	59.69	5.5	0.18 s	67.68
RBF SVC	52.02	0.04	1.4 s	49.49
k-NN	59.87	11.14	0.082 s	65.66
Naïve Bayes	59.46	19.08	0.018 s	72.73
Random Forest	61.93	10.39	0.22 s	72.73
Extra Trees	59.65	16.82	0.15 s	75.76
Decision Tree	58.64	12.7	0.22 s	65.66

As visible from the results, SVC and NuSVC offer the highest accuracy among other classification techniques.

8. Improving the model

The *Principal Component Analysis* (PCA) kernel is applied initially before applying the GridSearch. This is where the input features are huge in number,

and it needs decomposition into a few but preserving the variance across the features. The package comes with KernelPCA routine to condense the features into a smaller set. The method can take various kernels to perform PCA. The data has to scaled for PCA. The results I get is as written below, which is then passed to the GridSearch.

```
[[-0.3654283   0.01406002 -0.37134299  0.22005371]
 [ 0.03449783  0.19461805  0.00658418  0.01832307]
 [-0.46101512 -0.29994101 -0.20307323 -
0.21289277]
 ...
 [-0.35584145 -0.43794787  0.0458943   0.07503793]
 [-0.10312669  0.18948282 -0.09555745 -
0.04915244]
 [ 0.27288907  0.07697138 -0.13328894
0.19329518]]
```

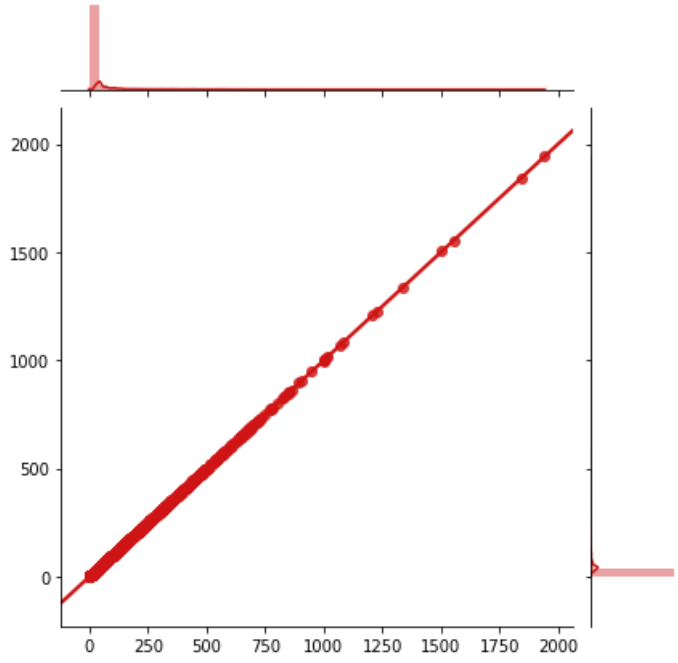
Not all parameters of a classifier are learned from the estimators. Those parameters are called hyper-parameters and are passed as arguments to the constructor of the classifier. Each estimator has a different set of hyper-parameters, which can be found in the corresponding documentation. We can search for the best performance of the classifier sampling different hyper-parameter combinations. This will be done with an exhaustive grid search, provided by the GridSearchCV function. The model parameter tuning is a daunting task, and multiple iterations have to be logged in with their performance metrics until one reaches the best set of parameters. Parameter tuning is mostly simplified in Scikit-learn by the GridSearchCV routine. Given a list of model parameter combinations, the method runs all possible combinations and returns the best model parameter along with the best estimator. The method also performs cross-validation, so the best estimator does not overfit the training data. The grid search will be done only on the best models, which are Random Forest, Extra Trees and SVC, NuSVC. After running the piece of codes below, it will be presented the accuracy, the cross-validation score and the best set of parameters.

Table 6 Results after implementing GridSearch approach

Classifiers	***Scores***				***Best parameters***					
	Accuracy %	***CV score %***	***CV (+/-) %***	***Execution time***	***Criterion***	***n_estimators***	***c***	***gamma***	***d***	***nu***
Random Forest	74.75	59.32	19.57	316.86 s	entropy	83	N/A			
Extra Trees	69.7	59.16	16.24	260.23 s	gini	69				
SVC (kernel= RBF)	72.73	63.51	17.02	4524.2 s	N/A		100	0.0001	N/A	
NuSVC	55.56	58.88	6.65	4.77 s	N/A				2	0.9

9. Evaluation & Results & Discussion

The evaluation can be discussed as shown in the table below, that why one such particular works or doesn't work in terms of fetching the higher accuracy and the reason behind it. The below-given image represents the features plotted, after the feature selection step, which means, meaningful and important features have been deployed to show the performance of the selected features.



Classifiers	*Accuracy after feature selection*	*Accuracy after GridSearch*	*Improved?*	*Helpful?*	*Reason*
SGD	68.69	Didn't implement	N/A	No	This implementation works with data represented as dense or sparse arrays of floating point values for the features. The model it fits can be controlled with the loss parameter; by default, it fits a linear support vector machine (SVM).
SVC	80.81	72.73	NO		The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to a dataset with more than a couple of 10000 samples. The multiclass support is handled according to a one-vs-one scheme.
NuSVC	80.81	55.56			Similar to SVC but uses a parameter to control the number of support vectors.
Linear SVC	68.69	Didn't implement	N/A		Similar to SVC with parameter kernel='linear', This class supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme.
Poly SVC	67.68				If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial. SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.
RBF SVC	49.49			Somewhat YES	Unlike other classifiers, SVM-R doesn't get affected by the degree or class size and that is the reason, it outperforms when values of C and gamma are tuned well to separate the classes using higher dimensional space.
k-NN	65.66				The drawback of increasing the value of k is of course that as k approaches n, where n is the size of the instance base, the performance of the classifier will approach that of the most straightforward statistical baseline, the assumption that all unknown instances belong to the class most frequently represented in the training data.
Naïve Bayes	72.73			YES	In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions

					means that each distribution can be independently estimated as a one-dimensional distribution. This, in turn, helps to alleviate problems stemming from the curse of dimensionality.
Random Forest	72.73	74.75	YES		A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size. if the improvement of the criterion is identical for several splits enumerated during the search of the best split. To obtain a deterministic behaviour during fitting, random_state has to be fixed. The split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases
Extra Trees	75.76	69.7	NO		This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.
Decision Tree	65.66	Didn't implement	N/A	Somewhat YES	Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

10. Conclusion

By performing all the 11 classifiers on the prostate cancer genes and clinical dataset for 494 samples (patients) over 21K+ genes (features), conclusion can be made that, after classifying based on different machine learning algorithm we get some accuracy and then after performing the feature selection approach, we get 66 important features to improve the classification and yet it didn't improve, therefore I took help of GridSearchCV approach to approve the accuracy but eventually it only helped RandomForest classifier to improve its accuracy.

11. Future work

As mentioned in [2] in future we can apply that exact architecture to follow the route to discover the possibility of finding the meaningful biomarkers to detect whether the patient is affected by the prostate cancer or not and in future if a new patient(s) data is there, then similar approach can be used to detect and conclude whether that patient will have cause or not. In future, more concepts like dimensionality reduction, PCA, other classification techniques, regression can be applied to improve the accuracy for given classifiers and for the given set of datasets which can be the huge milestone in the cancer research area.

References

[1] "Prostate Adenocarcinoma (TCGA, Provisional)," cBioPortal: For canncer genomics, [Online]. Available: http://www.cbioportal.org/study?id=prad_tcga#summary. [Accessed 09 December 2018].

[2] "Choosing the right estimator," scikit learn, [Online]. Available: https://scikit-learn.org/stable/tutorial/machine_learning_ma

p/index.html. [Accessed 15 December 2018].

[3] "Nearest Neighbours," scikit learn, [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html. [Accessed 15 December 2018].

[4] "Naive Bayes," scikit learn, [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html. [Accessed 15 December 2018].

[5] "Support Vector Machines (SVMs)," scikit learn, [Online]. Available: https://scikit-learn.org/stable/modules/svm.html#classification. [Accessed 15 December 2018].

[6] "LinearSVC," scikit learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC. [Accessed 15 December 2018].

[7] "Polynomial kernel," The wikipedia contributors, [Online]. Available: https://en.wikipedia.org/wiki/Polynomial_kernel. [Accessed 15 December 2018].

[8] "RBF SVM parameter," scikit learn, [Online]. Available: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html. [Accessed 15 December 2018].

[9] "NuSVC," scikit learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html#sklearn.svm.NuSVC. [Accessed 15 December 2018].

[10] "Stochastic Gradient Descent," scikit learn, [Online]. Available: https://scikit-learn.org/stable/modules/sgd.html#classification. [Accessed 15 December 2018].

[11] "Random Forest classifier," scikit learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed 15 December 2018].

[12] "Extra Trees classifier," scikit learn, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html. [Accessed 15 December 2018].

[13] "Decision Trees," scikit learn, [Online]. Available: https://scikit-learn.org/stable/modules/tree.html. [Accessed 15 December 2018].

[14] S. Shahbeig, A. Rahideh, M. S. Helfroush and K. Kazemi, "Gene expression feature selection for prostate cancer diagnosis using a two-phase heuristic–deterministic search strategy," *IET digital library,* vol. 12, no. 4, pp. 162-169, 2018.

[15] K. Sechidis, "Comparison of different preprocessing techniques and feature selection algorithms in cancer datasets," 2018.

[16] M. Hajiloo, H. R. Rabiee and M. Anooshahpour, "Fuzzy support vector machine: An efficient rule-based classification technique for microarrays," *BMC Bioinformatics,* vol. 14, no. 13, 2013.

[17] "Selected articles from the 9th Annual Biotechnology and Bioinformatics Symposium (BIOT 2012)," BMC Bioinformatics, [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/supplements/volume-14-supplement-13. [Accessed 16 December 2018].

[18] S. H. Bouazza, A. Zeroual and K. Auhmani, "Gene expression data analyses for supervised prostate cancer classification based on feature subset selection combined with different classifiers," *ICMCS,* 2016.

[19] N. Puthiyedth, C. Riveros, R. Berretta and P. Moscato , "A New Combinatorial Optimization Approach for Integrated Feature Selection Using Different Datasets: A Prostate Cancer Transcriptomic Study," *PLOS | ONE,* vol. 10, no. 6, 2015.

[20] S. Asaithambi, "Why, How and When to apply Feature Selection," Towards Data Science, 31

January 2018. [Online]. Available: https://towardsdatascience.com/why-how-and-when-to-apply-feature-selection-e9c69adfabf2. [Accessed 15 December 2018].

[21] J. Brownlee, "An Introduction to Feature Selection," Machine learning process, 06 October 2014. [Online]. Available: https://machinelearningmastery.com/an-introduction-to-feature-selection/. [Accessed 15 December 2018].

[22] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research,* pp. 1157-1182, 2003.

[23] M. Mayo, "Step Forward Feature Selection: A Practical Example in Python," KDnuggets, [Online]. Available: https://www.kdnuggets.com/2018/06/step-forward-feature-selection-python.html. [Accessed 15 December 2018].

[24] "Sequential Feature Selection (SFS)," mlxtend, [Online]. Available: https://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/. [Accessed 14 December 2018].

[25] S. Kaushik, "Introduction to Feature Selection methods with an example (or how to select the right variables?)," Analytics Vidhya, 1 December 2016. [Online]. Available: https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/. [Accessed 15 December 2018].