Machine Learning Engineer Nanodegree
Capstone Project

Tumor Diagnosis Using Machine Learning Algorithms

1. Definition

1.1 Project Overview

Machine learning is a prevalent type of artificial intelligence technique that enables the computers to learn from data without being explicitly programmed. In the past decades, the promotion of computation ability of computers and the availability of big data make the massive application of this technology possible. Machine learning technology is promising in many fields due to its excellent performance on data analysis and pattern cognition. One field that is applying this technology is healthcare, especially disease diagnoses.

Generally, the medical diagnosis is based on a deep probing of the physical examination data; essentially, it is a data analysis and pattern cognition problem. However, the data is often huge, analyzing it manually by doctors is a tough and time consuming job, which leads to the long wait time of patients before receiving proper treatment. Thus, automating diagnosis by machine learning algorithms is a promising solution for this problem. Many scholars have discussed the feasibility of heart disease diagnosis (Ghumbre & Ghatol, 2012) and cancer prognosis (Kourou, Exarchos, Exarchos, Karamouzis, & Fotiadis, 2015) using machine learning. According to BBC, IBM AI system Watson is to diagnose rare diseases in Germany (BBC, 2016).

For most of the machine learning applications in medical field, large number of physical examination datasets are fed into certain machine learning algorithms that can automatically train the datasets and generate appropriate models which are capable to diagnose certain kind of diseases on the basis of the training datasets.

1.2 Problem Statement

In this project, I will try to find an optimal machine learning algorithm and train an appropriate model to diagnose the breast tumor as ‘malignant’ or ‘benign’ based on the given features (geometry of tumors) and the labels (diagnostic results).

This problem is quite typical in both disease diagnosis and machine learning. Given the features and the labels, various supervised machine learning algorithms can be applied. However, the unsupervised machine learning algorithms can also be performed if ignoring the labels.

Each algorithm owns its advantages and disadvantages. The selection of the algorithms will be based on the evaluation metrics.

1.3 Evaluation Metrics

The evaluation metrics can be used to quantify the performance are accuracy and F-score. The accuracy can be simply calculated by dividing the number of samples whose predicted labels equal the actual labels by the total number of samples.

$$Accuracy = \frac{number\ of\ samples\ with\ right\ diagnosis}{total\ number\ of\ samples}$$

Accuracy is the essential criterion in diagnostic problems, higher the accuracy, better the performance of this model. In binary classification, F-score is another measure of the accuracy, it considers both the precision and recall to compute the score. F-score can be expressed by the following equation:

$$F - score = 2 \times \frac{precision \times recall}{precision + recall}$$

The precision and recall can be calculated by confusion matrix easily.

At last, time efficiency of each training process will also be considered for model evaluation.

2. Analysis

2.1 Data Exploration

The inputs are typically physical examination data; it is created by Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian, General Surgery Department and Computer Science Department, University of Wisconsin. The data folder is downloaded from UCI machine learning repository (Wolberg, Street, & Mangasarian, n.d.) as the form of csv file.

The dataset includes 569 samples (rows) and 32 columns. The first column is patients' ID number (numerical); the second column is the diagnostic results (categorical, labeled as 'M'=malignant, 'B'=benign); the last thirty columns are the features computed from a digitized image of a fine needle aspirate of a breast mass (numerical).

The features include:

- Radius (mean of distances from center to points on the perimeter): Its mean, standard error, and worst case value located in the 3rd, 13th, and 23rd column, respectively.
- Texture (standard deviation of gray-scale values): Its mean, standard error, and worst case value located in the 4th, 14th, and 24th column, respectively.
- Perimeter: Its mean, standard error, and worst case value located in the 5th, 15th, and 25th column, respectively.
- Area: Its mean, standard error, and worst case value located in the 6th, 16th, and 26th column, respectively.
- Smoothness (local variation in radius lengths): Its mean, standard error, and worst case value located in the 7th, 17th, and 27th column, respectively.
- Compactness (perimeter^2/area-1.0): Its mean, standard error, and worst case value located in the 8th, 18th, and 28th column, respectively.

- Concavity (severity of concave portions of the contour): Its mean, standard error, and worst case value located in the 9th, 19th, and 29th column, respectively.
- Concave points (number of concave portions of the contour): Its mean, standard error, and worst case value located in the 10th, 20th, and 30th column, respectively.
- Symmetry: Its mean, standard error, and worst case value located in the 11th, 21st, and 31st column, respectively.
- Fractal dimension ("coastline approximation"-1): Its mean, standard error, and worst case value located in the 12th, 22nd, and 32nd column, respectively.

The dataset is complete and no one is abnormal. Figure 1 shows how the dataset looks like (just lists the first five samples).

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	\
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

	smoothness_mean	compactness_mean	concavity_mean	concave_points_mean	\
0	0.11840	0.27760	0.3001	0.14710	
1	0.08474	0.07864	0.0869	0.07017	
2	0.10960	0.15990	0.1974	0.12790	
3	0.14250	0.28390	0.2414	0.10520	
4	0.10030	0.13280	0.1980	0.10430	

	...	radius_worst	texture_worst	perimeter_worst	\
0	...	25.38	17.33	184.60	
1	...	24.99	23.41	158.80	
2	...	23.57	25.53	152.50	
3	...	14.91	26.50	98.87	
4	...	22.54	16.67	152.20	

	area_worst	smoothness_worst	compactness_worst	concavity_worst	\
0	2019.0	0.1622	0.6656	0.7119	
1	1956.0	0.1238	0.1866	0.2416	
2	1709.0	0.1444	0.4245	0.4504	
3	567.7	0.2098	0.8663	0.6869	
4	1575.0	0.1374	0.2050	0.4000	

	concave_points_worst	symmetry_worst	fractal_dimension_worst
0	0.2654	0.4601	0.11890
1	0.1860	0.2750	0.08902
2	0.2430	0.3613	0.08758
3	0.2575	0.6638	0.17300
4	0.1625	0.2364	0.07678

[5 rows x 32 columns]

Figure 1 First Five Samples of Dataset

2.2 Exploratory Visualization

The given dataset can be divided into three parts

Table 1 Dataset Division

Patients' ID (1^{st} column)	Diagnostic Results (2^{nd} column)	Features (3^{rd}-32^{nd} columns)

The first column is used for patients' identification; however, it is irrelevant to the diagnostic model, thus it will be dropped. The second column is used as label in the supervised learning; however, the values of the second column are categorical, thus they have be converted to numerical values. In this case, the label 'malignant' will be replaced by the numerical value 0, while 'benign' will be replaced by the numerical value 1.

The rest of the columns are used as features. The values in these columns are all numerical. Table 2 contains some statistical descriptions of the first five features. It shows the number of the samples, the average value, standard deviation, minimum value, maximum value, as well as the percentile values of each feature. It gives us an overview of this dataset.

Table 2 Feature Description (First Five Columns of Feature)

	1	2	3	4	5
Count	569	569	569	569	569
Mean	14.13	19.29	91.97	654.89	0.096
Std	3.52	4.30	24.30	351.91	0.014
Min	6.98	9.71	43.79	143.50	0.053
25% pctl	11.70	16.17	75.17	420.30	0.086
50% pctl	13.37	18.84	86.24	551.10	0.096
75% pctl	15.78	21.80	104.10	782.70	0.105
Max	28.11	39.28	188.50	2501.00	0.163

For training the data and testing the model in supervised learning, the whole dataset has to be splitted into the training set and the test set. The test set takes 25% of the samples randomly and the rest belongs to the training set.

Table 3 Samples Split

	Labels	Features
Sample Number in Training Set	$569 - 569 \times 0.25$	$569 - 569 \times 0.25$
Sample Number in Test Set	569×0.25	569×0.25

To understand the dataset better, I am going to visualize the scatter plot of the samples. However, thirty features (dimensions) cannot be visualized, I should perform PCA to the dataset firstly and then extract two most important components which account for more than 99% of the total variance.

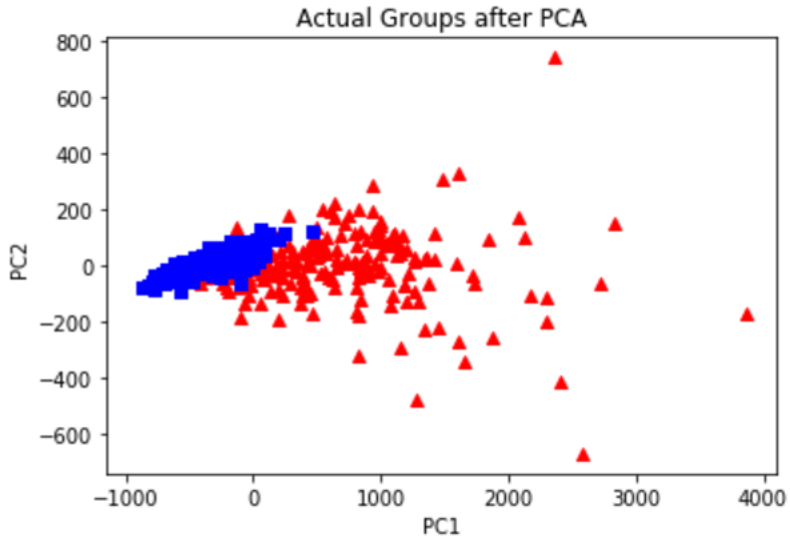

Figure 2 Scatter of Samples (PC=2) with Labels (Blue='B', Red='M')

From Figure 2, we can see that two clusters ('B' and 'M') are mingled with each other and cannot be linearly separated. It is also easy to discover that the samples labeled as 'B' are much more concentratedly distributed than the samples labeled as 'M'. In order to solve this complex non-linear separation problem, the following algorithms are to be performed.

2.3 Techniques and Algorithms

Since this is a typical classification problem, there are several kinds of machine learning algorithm that can be applied.

2.3.1 Logistic Regression

The first method I consider is logistic regression. In statistics, logistic regression is a regression model where the dependent variable is categorical, thus this method can deal with the classification problems well. Actually, logistic regression had been applied in the medical fields many years ago. For instance, the Trauma and Injury Severity Score, which is widely used for mortality prediction, is using logistic regression (Boyd, Tolson, & Copes, 1987).

To deal with the problem that labels are discrete values, instead of continuous numbers, changing the activation function to sigmoid. With this function, the outcome is always bounded between 0 and 1 (Ng).

$$g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

where g is called the logistic function or sigmoid function, θ can be considered as weight and x is the values of features.

Its derivative can be written as:

$$g'(\theta^T x) = g(\theta^T x)[1 - g(\theta^T x)]$$

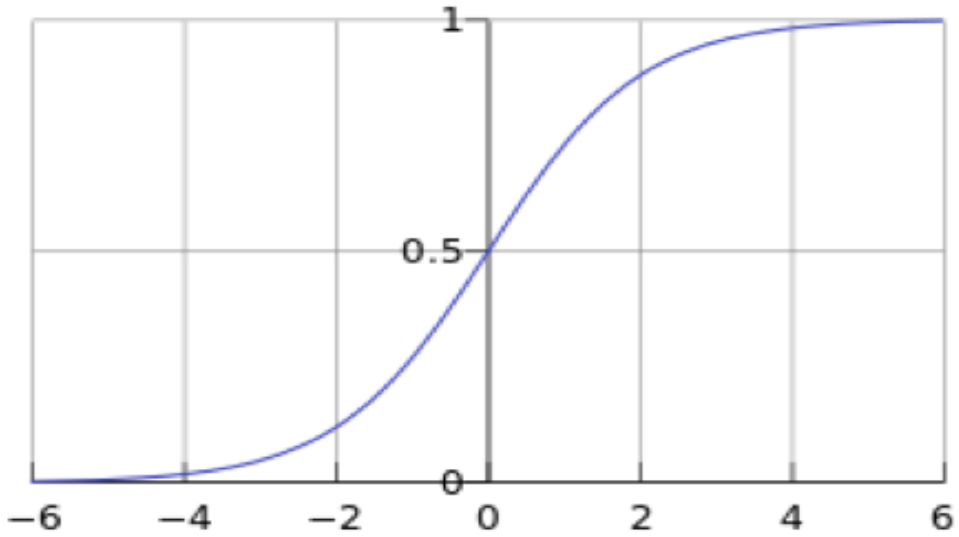

Figure 3 Sigmoid Function

To update the weight θ, stochastic gradient ascent rule can be applied:

$$\theta_j := \theta_j + \alpha[\nabla_\theta l(\theta)]$$

where

$$\nabla_\theta l(\theta) = \frac{\partial l(\theta)}{\partial \theta_j} = \sum_i [y_i - g(\theta^T x)] x_{i,j}$$

θ: is the weight after updated, α is the learning rate, y is the value of label, i, j stand for the index of samples and features, respectively.

However, overfitting is an annoying issue. The solution is penalizing certain parameters in loss function to fix that. This technique is called regularization (Mukherjee).

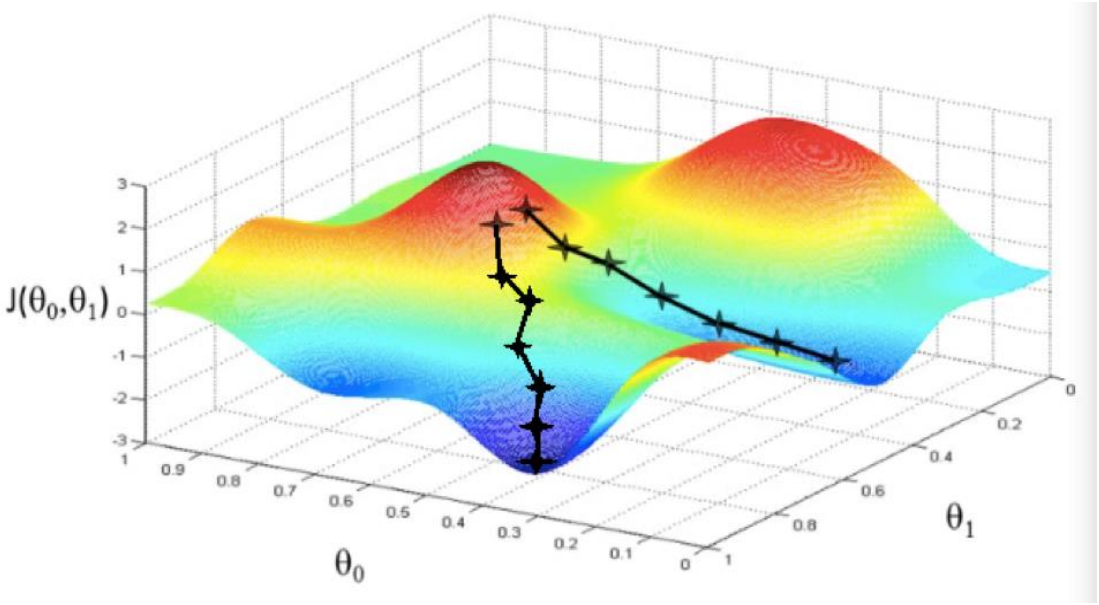

Figure 4 Gradient Ascent Rule

2.3.2 Support Vector Machine

Formally, support vector machine builds a hyper-plane or set of hyper-planes in a high dimensional space, which can not only be used for classification, but also regression. An excellent separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (called margin), since the larger the margin, generally the lower the error of the classifier (Wikipedia, n.d.). To solve the SVM problem, constructing the Lagrangian:

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(w^T x_i + b) - 1\right]$$

where $\alpha \geq 0$ is lagrange multiplier, w, b are the weight and bias respectively. Solving the optimization problem, the optimal values of w *and* b are:

$$w^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i$$

$$b^* = \frac{max_{i:y_i=-1} w^* x_i + min_{i:y_i=1} w^* x_i}{2}$$

The hyper-plane dividing the points appears to be:

$$w^* x + b^* = 0$$

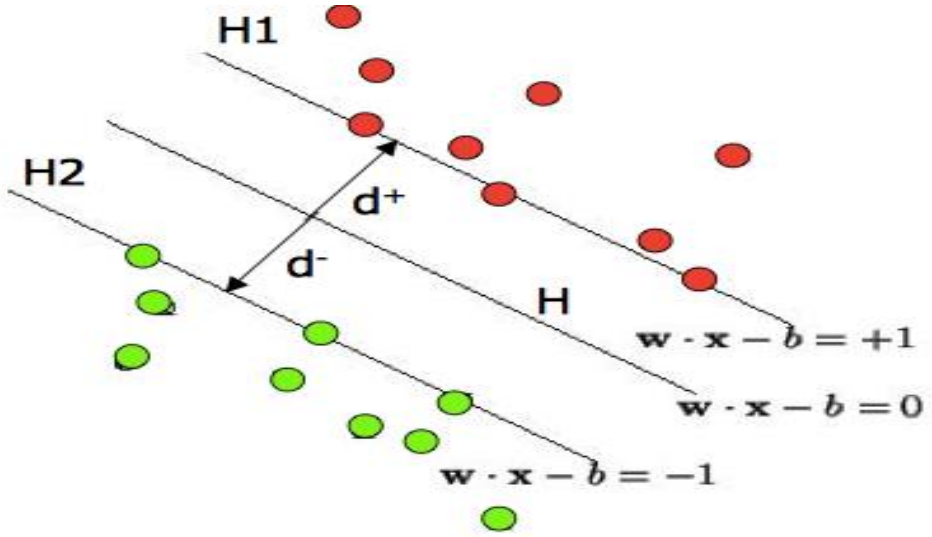

Figure 5 Linearly Separable Support Vector Machine

However, in machine learning problems, many datasets are not linearly separable. Thus an approach called 'kernel trick' has to be applied. By using the kernel functions, the data points are mapped into a high-dimensional, implicit feature space without computing the coordinates of the data in that space; instead, the inner products between images of all pairs of data in the feature space are to be calculated. There are various kinds of kernels, such as polynomial kernel, sigmoid kernel, etc. one of the most important kernels is Gaussian kernel:

$$K(x_1, x_2) = \emptyset(x_1)^T \emptyset(x_2) = \exp(-\frac{||x_1 - x_2||^2}{2\sigma^2})$$

To make the algorithm work for non-linearly separable datasets and be less sensitive to outliers, additional parameters, such as penalty parameter C, can be added, that is called regularization.

2.3.3 Decision Tree and Random Forest

Decision tree is a non-parametric supervised learning method used for both classification and regression. Its goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

This method is relative simple and easy to understand, it is able to handle the non-linearly separable datasets. Its structure looks like a tree, the input is entered at the top node and traverses down the tree, the data gets bucketed into smaller and smaller subsets.

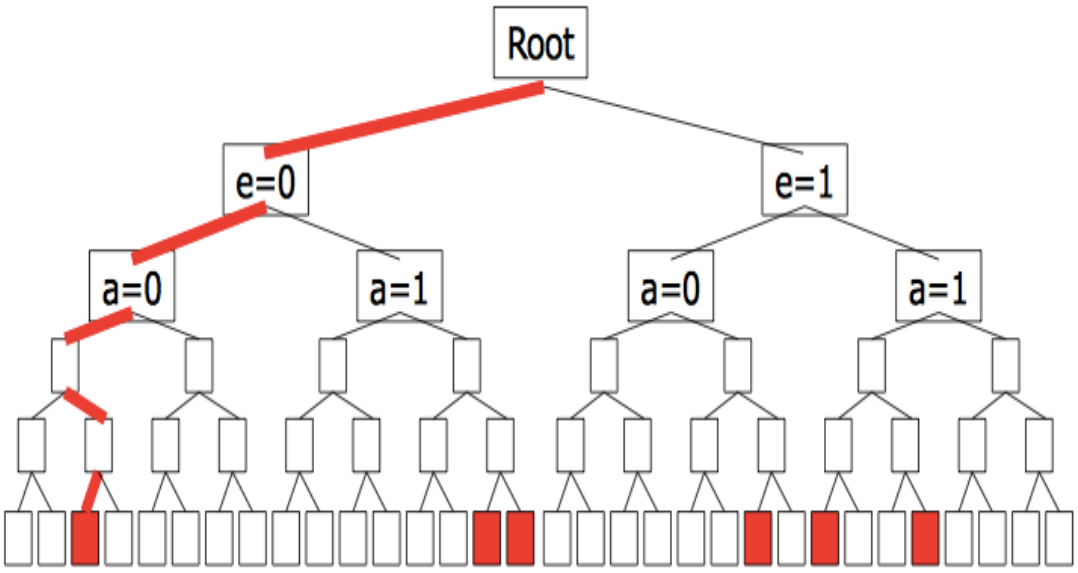

Figure 6 Decision Tree

Algorithm for constructing decision tree works top-down. Selecting a variable at each step that best splits the set of items is essential for this method. The prevalent metrics for measuring "best split" include CART and ID3 (information gain). ID3 is

achieved by calculating entropy, while CART is computing gini coefficient, instead of entropy.

However, decision tree can create over-complicated tree that do not generalize the data well (overfitting), and decision tree is also unstable since small variations in dataset may lead to a completely different tree.

To solve the problem, ensemble methods can be used. In this case, random forest tree could be a good choice since it has an excellent performance on overfitting avoidance.

Random forest is a typical ensemble method whose principle is that a group of "weak learners" can come together to form a "strong learner". In this case, the "weak learner" is decision trees. Random forest grows many decision trees randomly, and then the input data is put down each of the trees in the forest. Every tree will classify the input and vote for the final classification. At last, the algorithm will choose the classification having the most votes over all the trees in the forest.

Random forest algorithm attempts to mitigate the problems of high variance and high bias by averaging to find a natural balance between the two extremes (Walker, 2013).

2.3.4 Principal Component Analysis (PCA)

PCA is a powerful tool in data analysis. It finds a linear projection of high dimensional data into a lower dimensional subspace such that the variance retained is maximized, and the least square reconstruction error is minimized.

The main steps of operation describe as:

- Step 1: given the dataset, subtract the mean and calculate the covariance matrix.
- Step 2: calculate the eigenvectors and eigenvalues of the covariance matrix.
- Step 3: project the data.

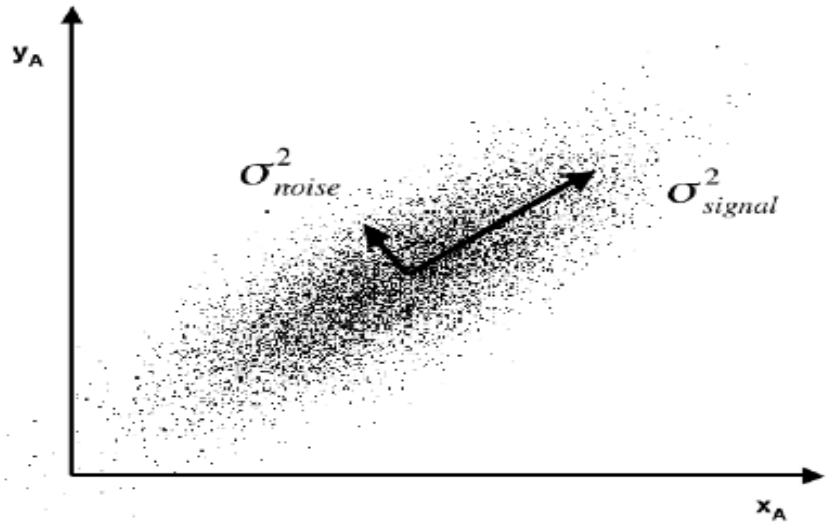

Figure 7 PCA (Shlens, 2003)

2.3.5 Gaussian Mixture Model (GMM)

In unsupervised learning algorithms, clustering is the most frequently used. It is able to learn from the data without any labels. K-means clustering is the simplest clustering technique that aims at minimizing the total mean squared error between the training sample points and their representative cluster centroids. However, if the sample points are mingled with each other, K-means algorithm may encounter problem to classify the dataset correctly.

If the uncertainty of the dataset exists, Gaussian Mixture Model often used. Unlike K-means, each cluster in GMM is mathematically represented by a parametric distribution (Gaussian); the entire dataset is modeled by a mixture of these distributions. GMM is usually known as the soft version of K-means clustering.

The EM algorithm for GMM is as following (Rosenberg, 2015):

- Initialize parameters μ, π, Σ
- "E step" --- Evaluate the responsibilities using current parameters:

$$\gamma_i^j = \frac{\pi_j \aleph(x_i|\mu_j, \Sigma_j)}{\sum_{j=1}^{k} \pi_c \aleph(x_i|\mu_j, \Sigma_j)}$$

- "M step" --- Re-estimate the parameters using responsibilities:

$$\mu_j^{MLE} = \frac{1}{n_j}\sum_{i=1}^{n} \Upsilon_i^j x_i$$

$$\Sigma_j^{MLE} = \frac{1}{n_j}\sum_{i=1}^{n} \Upsilon_i^j \left(x_i - \mu_j^{MLE}\right)^T \left(x_i - \mu_j^{MLE}\right)$$

$$\pi_j^{MLE} = \frac{n_j}{n}$$

where

$$\aleph\left(x_i|\mu_j, \Sigma_j\right) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_j|}} \exp[\frac{-\left(x_i - \mu_j\right)^T \Sigma_j^{-1} \left(x_i - \mu_j\right)}{2}]$$

π is cluster probability; μ is cluster centroid; Σ is cluster covariance matrix; x is the values of sample; Υ is exactly the soft assignment for x; n is the number of samples; k is the number of clusters; d is the dimension of x; i, j are the index of samples and clusters respectively.

Then repeat until log-likelihood converges.

2.4 Benchmark

Several types of machine learning algorithm (including supervised learning and unsupervised learning) will be applied to the tumor dataset. Each algorithm will be evaluated by the evaluation metrics.

The final results given by the evaluation metrics (accuracy and F-score) are certainly ranging from 0 to 1.

Since the requirement for accuracy in medical field is much more rigorous than most of other fields, the judgment standard is recommended as follows (G, et al., 2016):

- $0.975 \sim 1 \rightarrow excellent\ (A)$
- $0.95 \sim 0.975 \rightarrow good\ (B)$
- $0.9 \sim 0.95 \rightarrow acceptable\ (C)$

- $below\ 0.9\ \rightarrow fail\ (F)$

The final score is based on the average value of accuracy and F-score of the test dataset. The algorithms yielding grade 'A' are preferred.

3. Methodology

3.1 Data Preprocessing

The data preprocess functions are defined in the class Data_Process(). The function get_data() imports the dataset from Cancer_Data.csv file, and then divides the whole dataset into features and labels, it also turns the categorical labels to numerical.

The function data_split() splits the samples into training set and test set with the ratio of 3 to 1.

```
# split the data into training and test sets, 25% of the data is the test dataset
def data_split(self, X, Y):
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25, random_state=0)
    return X_train, X_test, y_train, y_test
```

The function data_normalozation() normalizes the data by columns using the following standard:

$$\frac{x_i - \frac{1}{n}\sum_{i=1}^{n} x_i}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \frac{1}{n}\sum_{i=1}^{n} x_i)^2}}$$

Function data_PCA() and feature_extract() perform PCA on the dataset and extract the most important components after PCA by setting the parameter 'n'.

3.2 Algorithm Implementation

3.2.1 Supervised Learning

Logistic regression, support vector machine, decision tree, as well as random forest algorithms have been performed to train the preprocessed training dataset.

Initially, without deep probing the parameters of each algorithm in sklearn package, I naïvely use the default settings directly. Since the codes for realizing each algorithm are repeated, I use for loop to implement them all.

```
# perform four kinds of supervised machine learning algorithm on the data set
for smla in [LogisticRegression(random_state=0), SVC(random_state=0),
             DecisionTreeClassifier(random_state=0), RandomForestClassifier(random_state=0)]:
    smla_name = smla.__class__.__name__
    time_results={}    # for calculate the time efficiency
    # data training
    start1 = time()
    smla.fit(X_train, y_train)
    end1 = time()
    time_results['training time'] = end1 - start1
```

Then the models are trained by the above code. After training, the models achieved will be used to predict the labels according to the input features. Finally, the evaluation metrics of these models will be calculated by the following code.

```
# make prediction
start2 = time()
y_train_pred = smla.predict(X_train)
y_test_pred = smla.predict(X_test)
end2 = time()
time_results['prediction time'] = end2 - start2

# calculating the metrics
train_accuracy = 1.0 * np.mean(y_train_pred == y_train)
test_accuracy = 1.0 * np.mean(y_test_pred == y_test)
train_fscore = f1_score(y_train, y_train_pred)
test_fscore = f1_score(y_test, y_test_pred)
```

3.2.2 Unsupervised Learning

Initially, the simplest unsupervised learning technique, K-means clustering is applied to the dataset. Unlike supervised learning, only the features will be used in the training section, and the dataset is unnecessary to be splitted into two subsets. Following code is the implementation of K-means clustering.

```
# data training
start1=time()
kmeans=KMeans(n_clusters=2, random_state=0).fit(X)     # use 2 clusters
end1=time()
time_results['training time'] = end1 - start1

# make prediction
start2=time()
pred=kmeans.labels_
end2=time()
time_results['prediction time'] = end2 - start2
```

The performance of this algorithm can be evaluated by comparing the labels' true value and predicted value.

```
print accuracy_score(Y, predict)
print f1_score(Y, predict)
```

3.2.3 Problems

The naive implementations of the intended algorithms are relatively easy, fast; however, the results are not quite satisfactory. None of the algorithm achieves the 'A' level performance, and severe problem of overfitting is emerged in some algorithms (SVM and decision tree). The results of the first implementation can be seen in Table 4.

Table 4 Performance of Initial Implementation of Each Learning Algorithm

Algorithms	Accuracy		F-Score		Training Time (s)	Prediction Time (s)
	Train	Test	Train	Test		
Logistic Regression	0.9601	0.9580	0.9683	0.9659	0.0040	0.0000
Support Vector Machine	1.0000	0.6294	1.0000	0.7725	0.0310	0.0310
Decision Tree	1.0000	0.8811	1.0000	0.8994	0.0160	0.0000
Random Forest	1.0000	0.9510	1.0000	0.9600	0.1000	0.0320
K-means Clustering	0.9051		0.9262		0.0520	0.0000

3.3 Algorithm Refinement

3.3.1 Supervised Learning Refinements

To achieve higher accuracy, several important parameters in each supervised learning algorithm have to be tuned. GridSearchCV in sklearn.model_selection class, which allows us to create a special model that can find its optimal parameter values by cross validation, will be used to adjust the parameters.

In GridSearchCV, we can feed any parameters willing to try to the argument 'param_grid', then system will make combination of these parameters and score them by cross validation. Finally, the model with the highest score (for the validation dataset) will be selected as the final model to make predictions.

Parameter tunings for logistic regression, SVM (normalized), decision tree and random forest are demonstrated by the following codes sequentially. The comments describe the parameters to be tuned.

```
LR = LogisticRegression(random_state=0)
# Exhaustive search over specified parameter values for an estimator, including:
#  1. The penalty term (L1--power 1 or L2--power 2)
#  2. 'C' means inverse of regularization strength. Like in support vector machines,
#      smaller values specify stronger regularization.
#  3. 'fit_intercept' specifies if a constant (a.k.a. bias or intercept) should be added to the decision function.
parameters={'penalty':['l1', 'l2'], 'C':[0.1, 1, 10], 'fit_intercept':[True, False]}
scorer=make_scorer(accuracy_score)    # make accuracy as the evaluation metrics in parameeter tuning

# data training (including parameter tuning)
start1 = time()
LR_obj=GridSearchCV(estimator=LR, param_grid=parameters, scoring=scorer)
LR_fit=LR_obj.fit(X_train, y_train)
LR_best=LR_fit.best_estimator_
end1 = time()
time_results['training time'] = end1 - start1
```

```
svc = SVC(random_state=0)
# Exhaustive search over specified parameter values for an estimator, including:
#  1.'C': Penalty parameter of the error term. smaller values specify stronger regularization.
#  2.'kernel': Specifies the kernel type to be used in the algorithm, including polynomial, gaussian and sigmoid
#  3.'gamma': defines how far the influence of a single training example reaches,
#     a small gamma value define a Gaussian function with a large variance
parameters = {'C': [1, 10, 20, 50], 'kernel': ['poly', 'rbf', 'sigmoid'], 'gamma': [0.1, 0.01, 0.001, 0.0001]}
scorer = make_scorer(accuracy_score)   # make accuracy as the evaluation metrics in parameeter tuning

# data training (including parameter tuning)
start1 = time()
svc_obj=GridSearchCV(estimator=svc, param_grid=parameters, scoring=scorer)
svc_fit=svc_obj.fit(X_train, y_train)
svc_best = svc_fit.best_estimator_
end1 = time()
time_results['training time'] = end1 - start1
```

```
DTC = DecisionTreeClassifier(random_state=0)
# Exhaustive search over specified parameter values for an estimator, including:
#  1. The criterion used (calculating gini or entropy)
#  2. The selection of max number of features will be used
#  3. The selection of max depth the tree will reach
parameters = {'criterion': ['gini', 'entropy'], 'max_features': ['auto', 'log2', None], 'max_depth': [3, 5, None]}
scorer = make_scorer(accuracy_score)    # make accuracy as the evaluation metrics in parameeter tuning

# data training (including parameter tuning)
start1 = time()
DTC_obj=GridSearchCV(estimator=DTC, param_grid=parameters, scoring=scorer)
DTC_fit=DTC_obj.fit(X_train, y_train)
DTC_best = DTC_fit.best_estimator_
end1 = time()
time_results['training time'] = end1 - start1
```

```
RFC = RandomForestClassifier(random_state=0)
# Exhaustive search over specified parameter values for an estimator, including:
#  1.'n_estimators': The number of trees in the forest.
#  2.The criterion used for calculation('gini' or 'entropy')
#  3.The max depth the trees will reach
parameters = {'n_estimators': [50, 100, 200], 'criterion': ['gini', 'entropy'], 'max_depth': [3, 4, 5, None]}
scorer = make_scorer(accuracy_score)   # make accuracy as the evaluation metrics in parameeter tuning

# data training (including parameter tuning)
start1 = time()
RFC_obj=GridSearchCV(estimator=RFC, param_grid=parameters, scoring=scorer)
RFC_fit=RFC_obj.fit(X_train, y_train)
RFC_best = RFC_fit.best_estimator_
end1 = time()
time_results['training time'] = end1 - start1
```

Due to the adjustments of the validation dataset (splitted from training dataset), the problem of overfitting can be avoided, and instead of default parameters, substantial number of combinations of parameters input also help improve the model performance.

After the refinement process, the performance of each algorithm has been obviously promoted, and the problem of overfitting has been largely alleviated. The details of the model performance after fine tuning can be seen in Table 5 and further discussion will be made in the next chapter.

However, this refinement process is not easy. Much time is consumed on finding optimal parameters. Since the parameter tuning process is manually, it is usually quite slow. Plus, program's running time is also soaring.

In this dataset, there are 30 features. PCA algorithm can be used to check if there is any possibility of feature reduction.

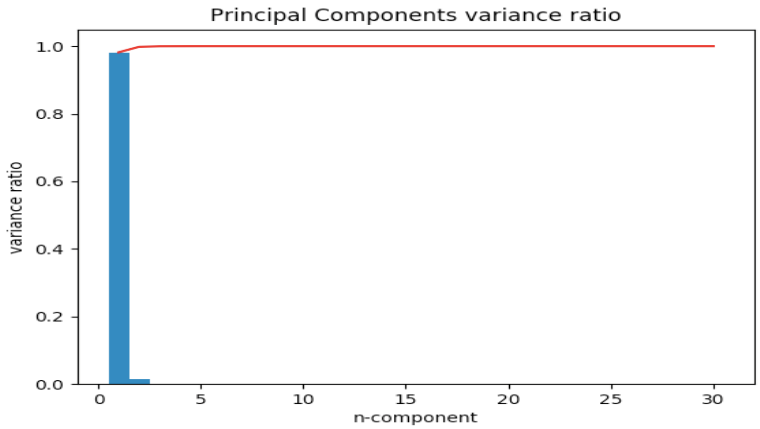

Figure 8 Doing PCA on Dataset

Figure 8 shows that after principal components analysis, the first and second principal components account for most of the variance of all the components. Thus, for better visualization and faster computation, the two components will be treated as the new features. Then the number of features are reduced from 30 to 2. The algorithms used previously will be performed again using the two new features. The details of the model performance after feature reduction will be shown in Table 6 and further discussion will be made in the next chapter.

3.3.2 Unsupervised Learning Refinements

Figure 2 indicates that the sample points are mingled with each other. In this case, the probabilistic model, GMM, which is also known as the "soft version" of K-means clustering, can be tried to improve the performance of the clustering. The results of this improvement will be shown in Table 5 and Table 6, further discussion will be made in the next chapter.

4. Results

4.1 Model Evaluation and Validation

Table 5 shows the results of each algorithm after parameter tuning. Comparison can be made with Table 4, which shows the results of the initial implementation.

Table 5 Performance of Each Learning Algorithm with Refinements

Algorithms	Accuracy		F-Score		Training Time (s)	Prediction Time (s)
	Train	Test	Train	Test		
Logistic Regression	0.9742	0.9720	0.9794	0.9773	4.7830	0.0000
Support Vector Machine	0.9883	0.9720	0.9907	0.9780	2.2030	0.0000
Decision Tree	0.9930	0.8951	0.9944	0.9133	0.5510	0.0160
Random Forest	0.9953	0.9790	0.9962	0.9832	74.6760	0.2340
GMM Clustering	0.9508		0.9611		0.1160	0.0000

From the results table above, the performance of each algorithm after parameter tuning has been promoted more or less, while the time consumption also increases. The performance of SVM sees the most significant improvement of all due to the successfully avoidance of overfitting. Random forest algorithm yields the highest accuracy and F-score, however, its time consumption is considerably more than the others. Decision tree has superior performance on training set, however, its performance on test set lags behind, that is the indication of overfitting.

Although random forest algorithm owns the best performance on the predictive accuracy and F-score, SVM is the optimal choice if the time efficiency is also considered. The predictive accuracy of SVM is just slightly lower than that of random forest, but the speed of SVM is much faster.

GMM, as an unsupervised learning algorithm, has remarkably satisfactory performance on this classification problem. More than 95% of the samples are classified correctly by this approach.

Table 6 shows the results of each algorithm after feature reduction.

Table 6 Performance of Each Learning Algorithm after PCA

Algorithms	Accuracy		F-Score		Training Time (s)	Prediction Time (s)
	Train	Test	Train	Test		
Logistic Regression	0.9272	0.9441	0.9433	0.9560	0.2640	0.0000
Support Vector Machine	0.9343	0.9441	0.9493	0.9560	2.0550	0.0000
Decision Tree	0.9413	0.9371	0.9538	0.9497	0.3840	0.0000
Random Forest	0.9554	0.9580	0.9653	0.9670	70.4610	0.1250
GMM Clustering	0.9332		0.9485		0.0220	0.0000

After feature reduction, the accuracy of predictions of all the algorithms, except decision tree, is getting worse. However, the time consumptions are all decreasing. The predictive accuracy of decision tree is promoted since less number of features makes the problem of overfitting alleviated.

After feature reduction, the time efficiency of logistic regression sees a strikingly improvement --- faster much more than 10 times!

Based on the evaluation metrics, this refinement (feature reduction) is not quite satisfactory, all the algorithms fail to achieve grade 'A' level.

4.2 Justification

According to the benchmark, the final grade of each algorithm is listed below. To incorporate time efficiency, running time that exceeds 4 second is marked as slow, 1-4 seconds is medium, and less than 1 second is fast.

Table 7 Grade of Each Learning Algorithm with Refinements

Algorithms	30 Features with Tuning		2 Features with Tuning	
	Grade	Time	Grade	Time
Logistic Regression	B	Slow	B	Fast
Support Vector Machine	A	Medium	B	Medium
Decision Tree	C	Fast	C	Fast
Random Forest	A	Very Slow	B	Very Slow
GMM Clustering	B	Very Fast	C	Very Fast

According to the table above, the optimal solution should be the SVM of 30 features with parameter tuning. This solution has excellent performance on both predicative accuracy (grade 'A') and time efficiency (medium). I think this model is competent to solve this problem. This optimal model is fully described as below:

```
SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.1, kernel='rbf',
  max_iter=-1, probability=False, random_state=0, shrinking=True,
  tol=0.001, verbose=False)
```

However, in many real-world problems, labels are not certainly available; if there are no adequate labels available, GMM clustering is also a reasonable alternative approach.

5. Conclusion

5.1 Classification Visualization

The four images below (Fig 9-12) generally show the characteristics and performance of the four supervised learning classifiers sequentially.

Since each feature represents one dimension, thirty features are tough to be visualized. The visualizations will only present two principal components after PCA.

The kernel functions applied by SVM help it separate the non-linearly separable data points very well. It is good at dealing with dataset of high dimensions.

However, data normalization and parameter tuning are very important to SVM. In the naïve implementation, SVM has very bad performance on test dataset prediction and presents severe problem of overfitting. However, after normalization and parameter tuning, SVM achieves the best performance.

Figure 13 shows the result of GMM clustering of the dataset with two principal components. In this figure, blue points stand for the predicted 'benign' samples, while red points stand for the predicted 'malignant' samples. For sample points that mingled with each other, GMM clustering is often a better approach than K-means clustering. Comparing with Figure 2, we can see that GMM clustering generates quite accurate classification without using labels.

Figure 14 shows the vector values of each cluster center. For most of the features, comparing with the benign tumors, the values of the malignant tumors are much bigger and more widely dispersed distributed.

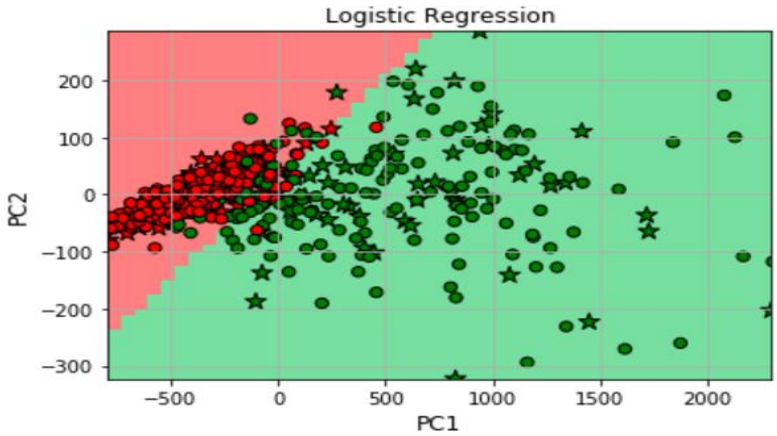

Figure 9 Classification by Logistic Regression (After PCA)

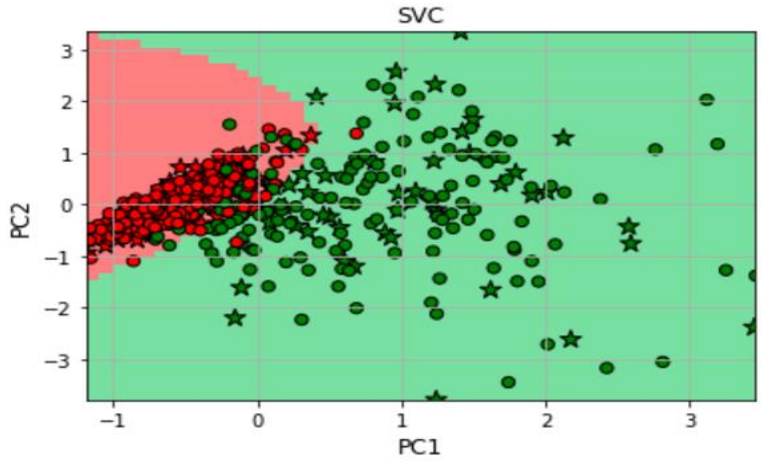

Figure 10 Classification by SVM (After PCA)

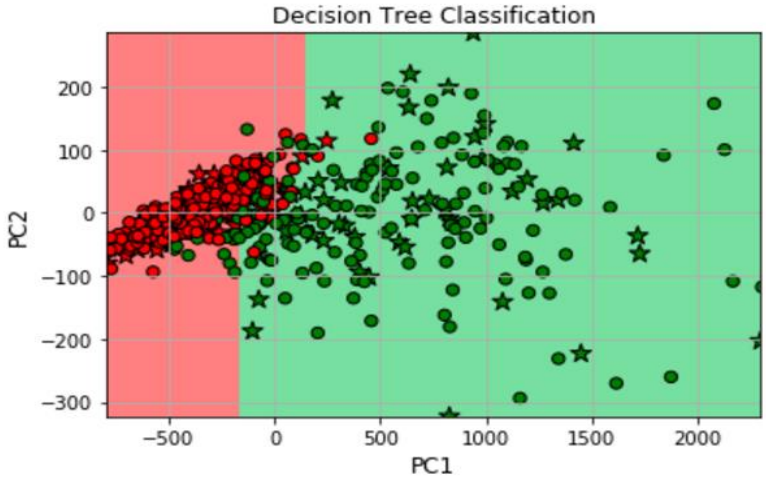

Figure 11 Classification by Decision Tree (After PCA)

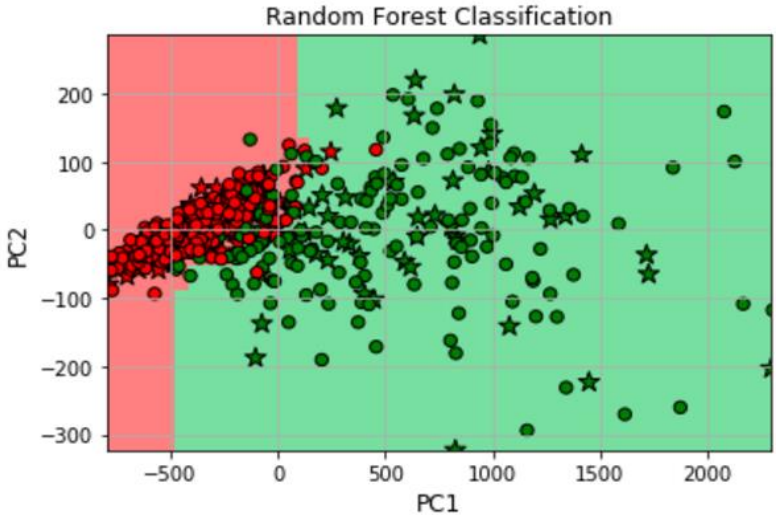

Figure 12 Classification by Random Forest (After PCA)

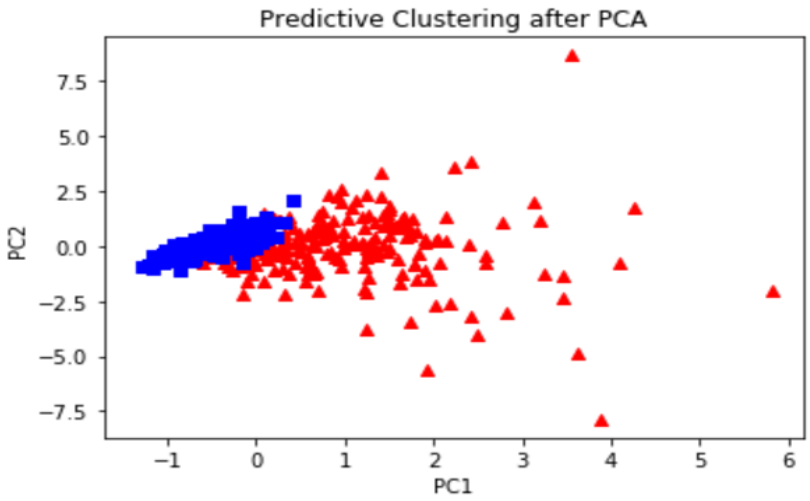

Figure 13 GMM Clustering (After PCA)

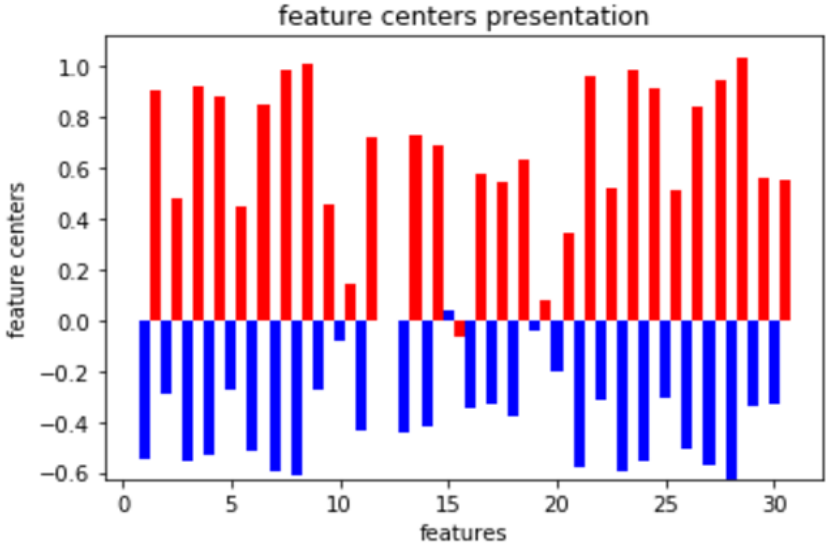

Figure 14 Cluster Centers with Thirty Features
(PS: blue bin represents the center of cluster 1 'benign' wrt. each feature, red bin represents the center of cluster 0 'malignant' wrt. each feature)

5.2 Summary and Reflection

In this project, I am going to find a machine learning algorithm that can diagnose breast cancer tumor as benign or malignant with high accuracy in reasonable time. Firstly, the dataset was analyzed, and I naïvely implemented four kinds of supervised learning (logistic regression, SVM, decision tree, random forest) and one kind of unsupervised learning (K-means clustering) algorithms with default parameters

without any refinements. However, the results can hardly achieve my expectation. After applying GridSearchCV to the supervised learning algorithms for parameters refining, and switching K-means clustering to GMM clustering, the results improve much. Both SVM and random forest achieve grade 'A' level on accuracy, but the latter consumes too much time. If involving the time efficiency, SVM becomes the optimal solution which is competent for this problem. In fact, even unsupervised learning algorithm (GMM) can generate quite accurate classification on the dataset, it achieves grade 'B'. It can be an appropriate alternative if labels are missing.

By PCA, I found that the first and second principal components account for more than 99% of the total variance. However, just using these two components as features will obviously decrease the predictive accuracy. So, the feature reduced model will not be used as the optimal model.

At last the final model chosen is SVM with 30 features with parameters refinements.

In this project, one of the most difficult part is the parameter tuning of each algorithm. There are so many parameter in each algorithm. How to select the parameters to be tuned, and how to decide the values of these parameter are all tough problems to solve. Studying the dataset carefully and several rounds of tryouts should be done to achieve the optimal parameters.

5.3 Improvement

In this project, I did some feature reduction operations, but with the decreasing of the features, the predictive accuracy also obviously dropped. In the future, more work can be done on how to reduce/extract features without obviously influencing the results.

In the real-world machine learning applications, there always substantial number of features involved, many of them may be redundant. Training all the features is time consuming and sometimes unnecessary. Thus, finding an appropriate method to

reduce the features without influencing the accuracy is meaningful. Future work can be done in this area.

References

BBC. (2016). *BBC-Technology*. Retrieved from BBC: http://www.bbc.com/news/technology-37653588

Boyd, C. R., Tolson, M. A., & Copes, W. S. (1987). *Evaluating trauma care: The TRISS method. Trauma Score and the Injury Severity Score.* The Journal of trauma.

G, R., M, C., S, A., D, M., A, F., & G, L. (2016). *Accuracy of clinical diagnosis of Parkinson disease: A systematic review and meta-analysis.*

Ghumbre, S. U., & Ghatol, A. A. (2012). *Heart Disease Diagnosis Using Machine Learning Algorithm.* Visakhapatnam.

Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., & Fotiadis, D. I. (2015). *Machine learning applications in cancer prognosis and prediction.* Computational and Structural Biotechnology Journal.

Mukherjee, A. (n.d.). *A study of Classification Problems using Logistic Regression and an insight to the admissions problem.*

Ng, A. (n.d.). *CS229 Lecture notes.* Stanford.

Rosenberg, D. (2015). *K-Means and Gaussian Mixture Models.* New York University.

Shlens, J. (2003). *A TUTORIAL ON PRINCIPAL COMPONENT ANALYSIS Derivation, Discussion and Singular Value Decomposition.* UCSD.

Walker, M. (2013). *Data Science Central*. Retrieved from http://www.datasciencecentral.com/profiles/blogs/random-forests-algorithm

Wikipedia. (n.d.). *Support Vector Machine*. Retrieved from https://en.wikipedia.org/wiki/Support_vector_machine

Wolberg, W. H., Street, N. W., & Mangasarian, O. L. (n.d.). *UCI-Machine Learning Repository*. Retrieved from https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)