Wisconsin Diagnostic Breast Cancer Dataset Project Report

Youyang Han Jan 1st, 2017

1. Definition

1.1 Project Overview

When I searched datasets in Kaggle, I focused on the subject in health industry for making combination of machine learning and biology and make efforts to improve the better understanding of disease. Finally, I find the newest released dataset subset from classical Wisconsin Diagnostic Breast Cancer Dataset. I think this dataset meet the needs both for myself and Udacity.

This dataset contains mostly numerical features which describe characteristics of different picture qualities and only categorical feature of diagnosis. This is easy to identify target connection that only exists between features and diagnosis. What is more, people are concerning the accuracy of cancer sensing for the great price for miss decision.

This dataset is one image dataset of Wisconsin Diagnostic Breast Cancer (WDBC) which comes from UCI Machine Learning. It is originally created in November 1995. Creators of WDBC are Dr. William H. Wolberg, W. Nick Street and Olvi L. Mangasarian.
They worked in University of Wisconsin, Madison. Donor of this dataset is Nick Street.

This database is also available through the UW CS ftp server:
ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

1.2 Problem Statement

Each feature describes one of characteristics of cell images. All features are statistically computed from image space while lack meaning of connection to real diagnosis. To explore the dataset, one is expected to contribute to find relationship of features and diagnosis by using different machine learning methods. This can be viewed as binary classification problem by definition.

To explore dataset and solve this problem, I would like to separate process into 3 different

categorical of method. First, I do not know if there is linear or non-linear relationship between features and diagnosis. So I will try both method to compare the benchmark accuracy provided by the dataset description. Second, I will try to improve the performance of method in condition. Finally, I will try to apply deep learning method on this problem to see if there is any space for improvement.

From the description of dataset, best predictive accuracy obtained using one separating plane in the 3-D space of Worst Area, Worst Smoothness and Mean Texture. Estimated accuracy 97.5% using repeated 10-fold cross validations. Classifier has correctly diagnosed 176 consecutive new patients as of November 1995. With exploring of dataset, I will apply different method and try to train and optimize the "best" model.

1.3 Metrics

Accuracy is our starting point. It is the number of correct predictions made divided by the total number of predictions made, multiplied by 100 to turn it into a percentage [1].

A clean and unambiguous way to present the prediction results of a classifier is to use a use a **confusion matrix**. For a binary classification problem the table has 2 rows and 2 columns. Across the top is the observed class labels and down the side are the predicted class labels. Each cell contains the number of predictions made by the classifier that fall into that cell.

	Positive	**Negative**
Positive	True Positive	False Positive
Negative	False Negative	True Negative

In this case, a perfect classifier would correctly predict 357 no recurrence and 212 recurrence which would be entered into the top left cell no recurrence/no recurrence (**True Negatives**) and bottom right cell recurrence/recurrence (**True Positives**).

Incorrect predictions are clearly broken down into the two other cells. **False Negatives** which are recurrence that the classifier has marked as no recurrence. We do not have any of those. **False Positives** are no recurrence that the classifier has marked as recurrence.

This is a useful table that presents both the class distribution in the data and the classifiers predicted class distribution with a breakdown of error types.

In a problem where there is a large class imbalance, a model can predict the value of the majority class for all predictions and achieve a high classification accuracy, the problem is that this model is not useful in the problem domain.This is called the **Accuracy Paradox**. For problems like, this additional measures are required to evaluate a classifier.

Precision is the number of True Positives divided by the number of True Positives and False Positives. Put another way, it is the number of positive predictions divided by the total number of

positive class values predicted. It is also called the Positive Predictive Value (PPV).A low precision can also indicate a large number of False Positives.

Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives. Put another way it is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity or the True Positive Rate. Recall can be thought of as a measure of a classifiers completeness. A low recall indicates many False Negatives.

The **F1 Score** is the 2*((precision *recall)/(precision+recall)). It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall.

In summary, I choose accuracy, precision and F1 score to evaluate the performance of the model prediction.

2. Analysis

2.1 Data Exploration

This dataset are linearly separable using all 30 input features. Also, there are 2 predicting fields, diagnosis: B = benign, M = malignant. Total numbers of attributes: 32 (ID, diagnosis, 30 real-valued input features). Total number of instances: 569.

Features of this dataset are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. A few of the images can be found at http://www.cs.wisc.edu/~street/images/

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

Attribute Information:
1) ID number
2) Diagnosis (M = malignant, B = benign)
3)-32):Ten real-valued features are computed for each cell nucleus:
a) radius (mean of distances from center to points on the perimeter)

b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

Specific description of several features:
The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features.
For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

In detail, there is no missing attribute values. All feature values are recoded with four significant digits. Diagnosis class distribution are 357 benign 212 malignant.

For instance of dataset: data[0:5] with all features:

	id	diagnosis
0	842302	M
1	842517	M
2	84300903	M
3	84348301	M
4	84358402	M

Fig.1 Id numbers and diagnosis.

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	_dimension_mean
0	17.99	10.38	122.8	1001	0.1184	0.2776	0.3001	0.1471	0.2419	0.07871
1	20.57	17.77	132.9	1326	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667
2	19.69	21.25	130	1203	0.1096	0.1599	0.1974	0.1279	0.2069	0.05999
3	11.42	20.38	77.58	386.1	0.1425	0.2839	0.2414	0.1052	0.2597	0.09744
4	20.29	14.34	135.1	1297	0.1003	0.1328	0.198	0.1043	0.1809	0.05883
	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave points_se	symmetry_se	al_dimension_se
0	1.095	0.9053	8.589	153.4	0.006399	0.04904	0.05373	0.01587	0.03003	0.006193
1	0.5435	0.7339	3.398	74.08	0.005225	0.01308	0.0186	0.0134	0.01389	0.003532
2	0.7456	0.7869	4.585	94.03	0.00615	0.04006	0.03832	0.02058	0.0225	0.004571
3	0.4956	1.156	3.445	27.23	0.00911	0.07458	0.05661	0.01867	0.05963	0.009208
4	0.7572	0.7813	5.438	94.44	0.01149	0.02461	0.05688	0.01885	0.01756	0.005115
	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	_dimension_worst
0	25.38	17.33	184.6	2019	0.1622	0.6656	0.7119	0.2654	0.4601	0.1189
1	24.99	23.41	158.8	1956	0.1238	0.1866	0.2416	0.186	0.275	0.08902
2	23.57	25.53	152.5	1709	0.1444	0.4245	0.4504	0.243	0.3613	0.08758
3	14.91	26.5	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.173
4	22.54	16.67	152.2	1575	0.1374	0.205	0.4	0.1625	0.2364	0.07678

Fig.2 30 features of dataset.

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	_dimension_mean
count	569	569	569	569	569	569	569	569	569	569
mean	14.127292	19.289649	91.969033	654.889104	0.09636	0.104341	0.088799	0.048919	0.181162	0.062798
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.07972	0.038803	0.027414	0.00706
min	6.981	9.71	43.79	143.5	0.05263	0.01938	0	0	0.106	0.04996
25%	11.7	16.17	75.17	420.3	0.08637	0.06492	0.02956	0.02031	0.1619	0.0577
50%	13.37	18.84	86.24	551.1	0.09587	0.09263	0.06154	0.0335	0.1792	0.06154
75%	15.78	21.8	104.1	782.7	0.1053	0.1304	0.1307	0.074	0.1957	0.06612
max	28.11	39.28	188.5	2501	0.1634	0.3454	0.4268	0.2012	0.304	0.09744
	radius_se	texture_se	perimeter_se	area_se	smoothness_se	compactness_se	concavity_se	concave points_se	symmetry_se	al_dimension_se
count	569	569	569	569	569	569	569	569	569	569
mean	0.405172	1.216853	2.866059	40.337079	0.007041	0.025478	0.031894	0.011796	0.020542	0.003795
std	0.277313	0.551648	2.021855	45.491006	0.003003	0.017908	0.030186	0.00617	0.008266	0.002646
min	0.1115	0.3602	0.757	6.802	0.001713	0.002252	0	0	0.007882	0.000895
25%	0.2324	0.8339	1.606	17.85	0.005169	0.01308	0.01509	0.007638	0.01516	0.002248
50%	0.3242	1.108	2.287	24.53	0.00638	0.02045	0.02589	0.01093	0.01873	0.003187
75%	0.4789	1.474	3.357	45.19	0.008146	0.03245	0.04205	0.01471	0.02348	0.004558
max	2.873	4.885	21.98	542.2	0.03113	0.1354	0.396	0.05279	0.07895	0.02984
	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	_dimension_worst
count	569	569	569	569	569	569	569	569	569	569
mean	16.26919	25.677223	107.261213	880.583128	0.132369	0.254265	0.272188	0.114606	0.290076	0.083946
std	4.833242	6.146258	33.602542	569.356993	0.022832	0.157336	0.208624	0.065732	0.061867	0.018061
min	7.93	12.02	50.41	185.2	0.07117	0.02729	0	0	0.1565	0.05504
25%	13.01	21.08	84.11	515.3	0.1166	0.1472	0.1145	0.06493	0.2504	0.07146
50%	14.97	25.41	97.66	686.5	0.1313	0.2119	0.2267	0.09993	0.2822	0.08004
75%	18.79	29.72	125.4	1084	0.146	0.3391	0.3829	0.1614	0.3179	0.09208
max	36.04	49.54	251.2	4254	0.2226	1.058	1.252	0.291	0.6638	0.2075

Fig.3 Statistics description of 30 features.

As we can see from above, features are all numerical feature whose numbers ranges from 0 to 4254. That is quite a large difference as the characteristics are not essentially related to each other. Since I will see the relationship between features and determine to standardize data into same level for better understanding of possible features. What is more, calculation and accuracy will be benefited from standardization.

2.2 Exploratory Visualization

First I choose to see if there are obvious imbalance in whole dataset. A heat map is a graphical representation of data where the individual values contained in a matrix are represented as colors.

Fractal maps and tree maps both often use a similar system of color-coding to represent the values taken by a variable in a hierarchy.Heat maps originated in 2D displays of the values in a data matrix. Larger values were represented by small dark gray or black squares (pixels) and smaller values by lighter squares.

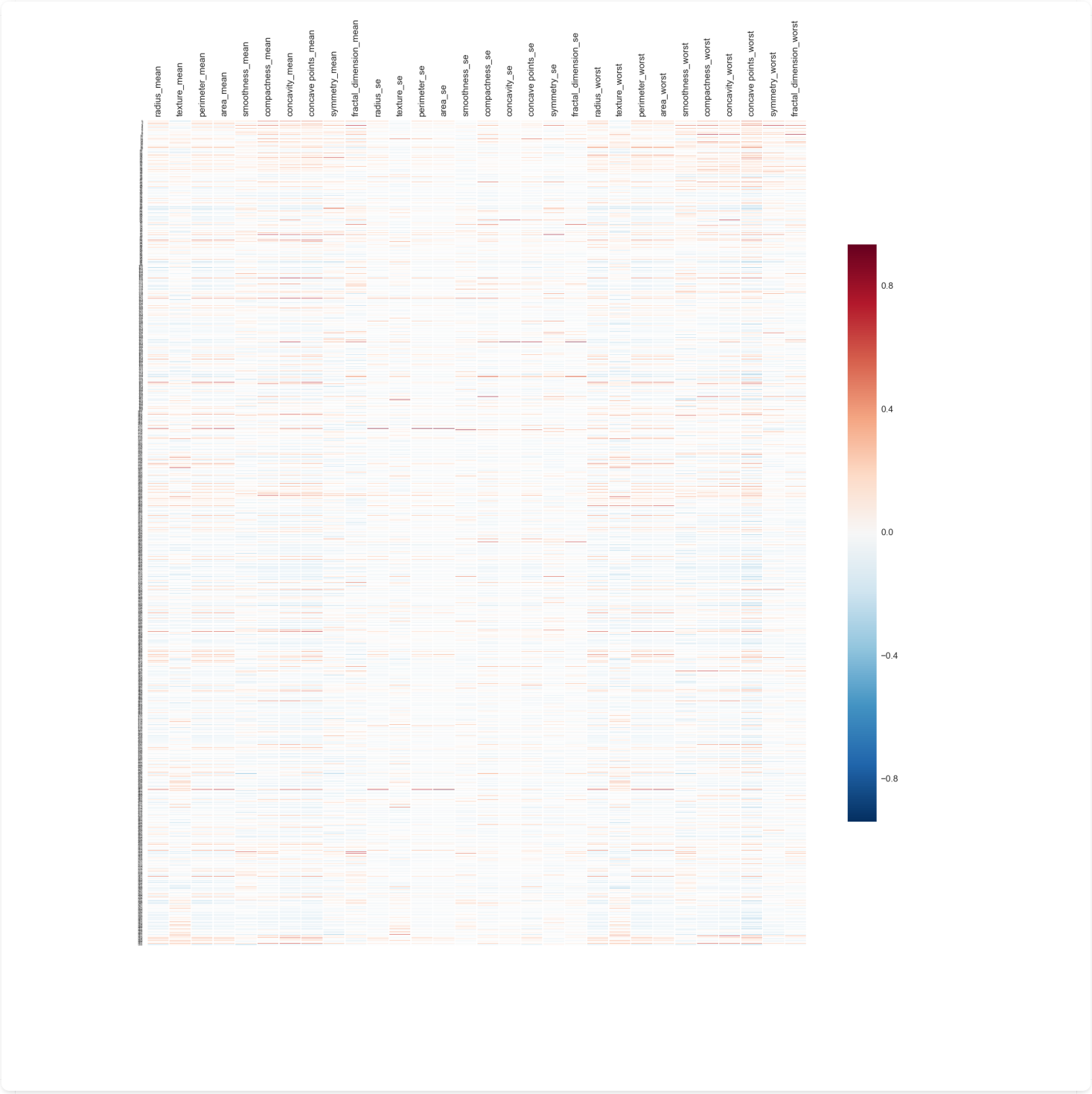


Fig.4 Heatmap of all data shows some imbalance in dataset.

As the map showed, there are no large imbalance for data inside each column to whole dataset, however there are some features do contain large imbalance in dataset which may be unsuitable for further analysis. I used outlier detection and removed those z-score larger than 3 since it may affect actual model we built[2].

Second, I will see if there are irrelative features to other features. In statistics, dependence or association is any statistical relationship, whether causal or not, between two random variables or two sets of data. Correlation is any of a broad class of statistical relationships involving dependence, though in common usage it most often refers to the extent to which two variables have a linear relationship with each other.

The correlation matrix of n random variables X1, …, Xn is the n × n matrix whose i,j entry is corr(xi, xj). Consequently, each is necessarily a positive-semidefinite matrix.
The correlation matrix is symmetric because the correlation between xi and xj is the same as the correlation between xj and xi.

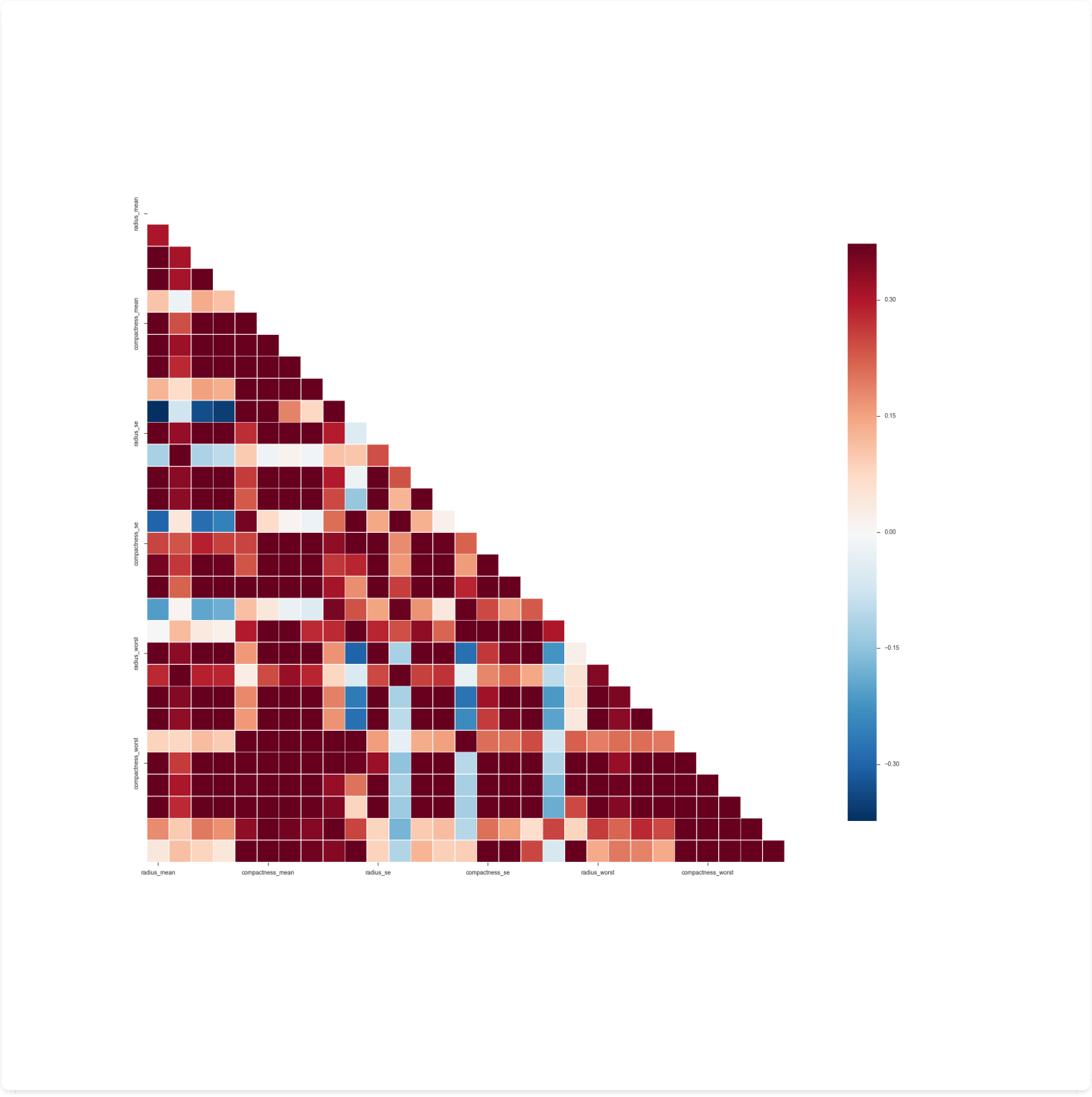


Fig.5 Correlation map of 30 features.

As map shows that most features have weak connection to other features. Only several features are relatively independent to others. Since I will try to reduce useless features to conduct feature selection at data processing.

Using correlation map is not enough for me to determine if features are suitable for classification

since correlation map only show linear relationship between features. I used pair plot since it shows the location of two diagnosis in a single unit map after I used MinMaxScaler function for each feature. Pair Plot for 30 features and each 10 features are all listed in the supplementary.

The pair plot shows that indeed there are indeed several features are linearly correlated but most features can be separated by a 2-D hyperplane. So that I should try both linear and non-linear method and also try to find some different ways to handle 30 features at one model.

2.3 Algorithms and Techniques

Binary classification is the task of classifying the elements of a given set into two groups on the basis of a classification rule. Instancing a decision whether an item has or not some qualitative property, some specified characteristic [3].

Normalization of ratings means adjusting values measured on different scales to a notionally common scale, often prior to averaging. In more complicated cases, normalization may refer to more sophisticated adjustments where the intention is to bring the entire probability distributions of adjusted values into alignment [4].

In the field of machine learning, **linear classifiers** identify by making a classification decision based on the value of a linear combination of the characteristics. An object's characteristics are also known as feature values and are typically presented to the machine in a vector called a feature vector. Such classifiers work well for practical problems such as document classification, and more generally for problems with many variables, reaching accuracy levels comparable to non-linear classifiers while taking less time to train and use [5].

In machine learning, this problem can be defined as **binary classification**. However, there are linear and non-linear methods to solve this problem. In this question, I will use 4 common linear methods to see if we can find solutions at first. Then I choose non-linear method to see the accuracy. Finally, I apply powerful deep learning method of TensorFlow comes from Google to get the best model I can find.

Logistic regression, is a linear model for classification rather than regression. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function [6].

The implementation of logistic regression in scikit-learn can be accessed from class LogisticRegression. This implementation can fit binary, One-vs- Rest, or multinomial logistic regression with optional L2 or L1 regularization.

Stochastic Gradient Descent is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as Support Vector Machines and Logistic Regression. Even though SGD has been around in the machine learning community for a long time,

it has received a considerable amount of attention just recently in the context of large-scale learning[7].

SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Given that the data is sparse, the classifiers easily scale to problems with more than 10^5 training examples and more than 10^5 features.

Support vector machinesare a set of supervised learning methods used for classification, regression and outliers detection. SVM requires that each data instance is represented as a vector of real numbers[8].

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features[9].

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(xi \mid y).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.

Besides the supposition of linear correlation, I also apply non-linear method without supposition.

Decision Trees 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].

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represents classification rules.

In decision analysis a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool, where the expected values (or expected utility) of competing alternatives are calculated.

Feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for three reasons [11] :

1. simplification of models to make them easier to interpret by researchers/users,
2. shorter training times,
3. enhanced generalization by reducing overfitting

The central premise when using a feature selection technique is that the data contains many features that are either redundant or irrelevant, and can thus be removed without incurring much loss of information. Redundant or irrelevant features are two distinct notions, since one relevant feature may be redundant in the presence of another relevant feature with which it is strongly correlated.

Feature extraction is very different from Feature selection: the former consists in transforming arbitrary data, such as text or images, into numerical features usable for machine learning. The latter is a machine learning technique applied on these features.

The class DictVectorizer can be used to convert feature arrays represented as lists of standard Python dict objects to the NumPy/SciPy representation used by scikit-learn estimators.

DictVectorizer implements what is called one-of-K or “one-hot” coding for categorical features. Transforms lists of feature-value mappings to vectors. When feature values are strings, this transformer will do a binary one-hot (aka one-of-K) coding: one boolean-valued feature is constructed for each of the possible string values that the feature can take on. For instance, a feature “f” that can take on the values “ham” and “spam” will become two features in the output, one signifying “f=ham”, the other “f=spam”.

However, note that this transformer will only do a binary one-hot encoding when feature values are of type string. If categorical features are represented as numeric values such as int, the DictVectorizer can be followed by OneHotEncoder to complete binary one-hot encoding.

Chi-square test measures dependence between stochastic variables, so using this function “weeds out” the features that are the most likely to be independent of class and therefore irrelevant for classification. So this test could be a good judgement for keep or discard features.

TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

Softmax regression is a generalization of logistic regression to the case where we want to handle multiple classes. In logistic regression we assumed that the labels were binary: $y(i) \in \{0,1\}$. We used such a classifier to distinguish between two kinds of hand-written digits. Softmax regression allows us to handle $y(i) \in \{1,\ldots,K\} y(i) \in \{1,\ldots,K\}$ where KK is the number of classes[12].

Using softmax regression is based on the model of MINST in TensorFlow. I try to apply powerful open source tool to solve this problem and this linear model method do give surprise in accuracy.

CNN is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. Individual cortical neurons respond to stimuli in a restricted region of space known as the receptive field. The receptive fields of different neurons partially overlap such that they tile the visual field. The response of an individual neuron to stimuli within its receptive field can be approximated mathematically by a convolution operation[13].

2.4 Benchmark

As mentioned before, there is one of the best predictive accuracy obtained using one separating plane in the 3-D space of Worst Area, Worst Smoothness and Mean Texture. Estimated accuracy 97.5% using repeated 10-fold cross validations. Classifier has correctly diagnosed 176 consecutive new patients as of November 1995.

3. Methodology

3.1 Data Preprocessing

Import data from file of breast cancer file with pandas package which provides method 'pd.read_csv' to read csv file.

Rename each column with labels and features. Since I already know most features have weak connections, feature selection is reserved for further refinement of analyzing. Giving 30 features their names can also help me select meaningful features based on prior knowledge if I have.

Raw data shape is (569, 33) which means 569 rows and 33 imported features all have been read into form of dataframe. But there are **2 labels** and**actually only 30 feature** in description and I manually delete one wrongly read column from file.

Watch and reshape the form of data to show the exact what dataset contains. This is necessary process for continuing analyze data.

Normalize features by removing the mean and scaling to unit variance. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the

training set. Mean and standard deviation are then stored to be used on later data using the transform method.

Outliers are detected and removed and there are 495 data left. Since there are 30 features and only 70 data removed if it contains any outliers in all features. Actually it is indeed small and can be accepted since on average there are 2 data removed because outliers in each features.

3.2 Implementation

General models

Load and applyeach algorithm. After I know the potential correlative connection between all features, I apply linear method first to see if it is best fit this dataset. What is more, I already find there is no data imbalance in dataset so I can standardize all features for less necessary calculation and more easy understandable meaning from distribution of percentile.

LogisticRegression, **SGDClassifier**, **SVC** and **GaussianNB** are all applied based on their default setting. Based on the different theory of each classifier, results give direction for further improvement.

```
Accuracy of LR:  0.959568733154
             precision    recall  f1-score   support

     Benigh       0.99      1.00      0.99        85
  Malignant       1.00      0.97      0.99        39

avg / total       0.99      0.99      0.99       124

Accuracy of SGDC:  0.946091644205
             precision    recall  f1-score   support

     Benigh       0.94      1.00      0.97        85
  Malignant       1.00      0.87      0.93        39

avg / total       0.96      0.96      0.96       124

Accuracy of SVC:  0.940700808625
             precision    recall  f1-score   support

     Benigh       0.92      1.00      0.96        85
  Malignant       1.00      0.82      0.90        39

avg / total       0.95      0.94      0.94       124

Accuracy of GB:  0.940700808625
             precision    recall  f1-score   support

     Benigh       0.95      0.98      0.97        85
  Malignant       0.95      0.90      0.92        39

avg / total       0.95      0.95      0.95       124

Accuracy of DT:  1.0
             precision    recall  f1-score   support

     Benigh       0.96      0.96      0.96        85
  Malignant       0.92      0.92      0.92        39

avg / total       0.95      0.95      0.95       124
```

Fig6. Apply of five basic machine learning models with default parameters.

As the figure showed, we get the accuracy, precision, recall, and F1 score of each algorithm. None of algorithms are better than benchmark of previous study and Decision-Tree is obviously over-fitting. I will try to explore a basic linear TensorFlow method to see if there are more interpretable results.

TensorFlow

Firstly, I try to **import, reshape and rename**data into the form that diagnosis can be labeled as numbers to clearly define this as continuous variable. Also, I add one column to label as benign and malignant to give different diagnosis a unique label for further calculation. So I can get this problem into a linear method with matrix for classification.

Secondly, since all features and labels are numbers and the dataset is balance which I know by heatmap, I apply MinMaxScale function to them so that number scale are same. Because laterly I will apply my algorithm not by nature of features but by the reduced mean of numbers and this can be useful for max pooling.

Thirdly, I construct a matrix calculation method to compute the y_ . Then I choose the least square method to describe the difference between y_ and real y. Besides, I set 0.01 as gradient to separate both labels which are benign and malignant respectively. All those process compose a whole flow of linear model building.

Besides, tf.argmax(y, 1) return value of axi 1 as well as tf.argmax(y_, 1). Then it check if values are match with function of tf.equal(). Finally this will return a list of booleans. This is only one part of function of computing accuracy.

The reduced_mean function is also one part to calculate accuracy. Indeed it is a way of testing difference of numbers.

I use CNN to set each data as a 1x30 matrix to apply neuron network.

Finally, I initialize all variables and use training set and test set to optimize model. Each time, the 200 randomly chosen data compose an unit and make matrix calculation with unknown w and b. The results will be computed and optimized with real y. After 20000 times calculation, the model shows trained results with 3000 times testing for every 200 times and get average accuracy.

```
step 200, linear model of TensorFlow testing accuracy 1.000000
step 400, linear model of TensorFlow testing accuracy 0.965000
step 600, linear model of TensorFlow testing accuracy 0.955000
step 800, linear model of TensorFlow testing accuracy 0.965000
step 1000, linear model of TensorFlow testing accuracy 0.985000
step 1200, linear model of TensorFlow testing accuracy 0.980000
step 1400, linear model of TensorFlow testing accuracy 0.975000
step 1600, linear model of TensorFlow testing accuracy 0.980000
step 1800, linear model of TensorFlow testing accuracy 0.960000
step 2000, linear model of TensorFlow testing accuracy 0.975000
step 2200, linear model of TensorFlow testing accuracy 0.960000
step 2400, linear model of TensorFlow testing accuracy 0.965000
step 2600, linear model of TensorFlow testing accuracy 0.965000
step 2800, linear model of TensorFlow testing accuracy 0.970000
step 3000, linear model of TensorFlow testing accuracy 0.995000
Average testing accuracy is 0.972352.
```

Fig.7 TensorFlow accuracy of linear method.

Although I trained 20000 times with randomly chosen data, the testing accuracy is indeed below benchmark for only 0.972 but it is still possible for improvement to apply powerful tools of neuro network on dataset.

3.3 Refinement

As the accuracy showed, decision-tree and TensorFlow should give more attention for improvement. I choose those two method with parameter tuning and feature selection and this is helpful in different ways.

3.3.1 Decision Tree refinement

While this is only have accuracy of 1 which is obviously overfit in the method I tried. So I try to improve this method with usual idea of reduce unnecessary features.

There are many potential benefits of **variable and feature selection**: facilitating data visualization and data understanding, reducing the measurement and storage requirements, reducing training and utilization times, defying the curse of dimensionality to improve prediction performance. Some methods put more emphasis on one aspect than another, and this is another point of distinction between this special issue and previous work

Selecting the most relevant variables is usually suboptimal for building a predictor, particularly if the variables are redundant. Conversely, a subset of useful variables may exclude many redundant, but relevant, variables.

Change all numbers to list record and build sparse matrix and then use DictVectorizer to fit and transform x_train while only transform x_test. After this process, I get all features showed in order of alphabet.

Calculate every 5 percentile of features accuracy and use cross validation. **Compute chi-squared**

stats between each feature. Since chi-square test measures dependence between stochastic variables, so using this function "weeds out" the features that are the most likely to be independent of class and therefore irrelevant for classification.

```
[ 0.8409241   0.85173491  0.91361372  0.90572381  0.91635345  0.91635345
  0.90831741  0.90831741  0.90817031  0.90294706  0.90546662  0.9246447
  0.91931137  0.92460866  0.921942    0.92190596  0.92201407  0.92731137
  0.9166447   0.91401506]
Optimal number of features 86 percentile
```

Fig.8 Optimal percentile of features

Finally, I get that the former 86 percentile of features give best performance in dataset. I follow what the results reported and change the percentile of features to 86. Run the decisiontree training and testing process before, I get several results which is similar to what before feature selection.

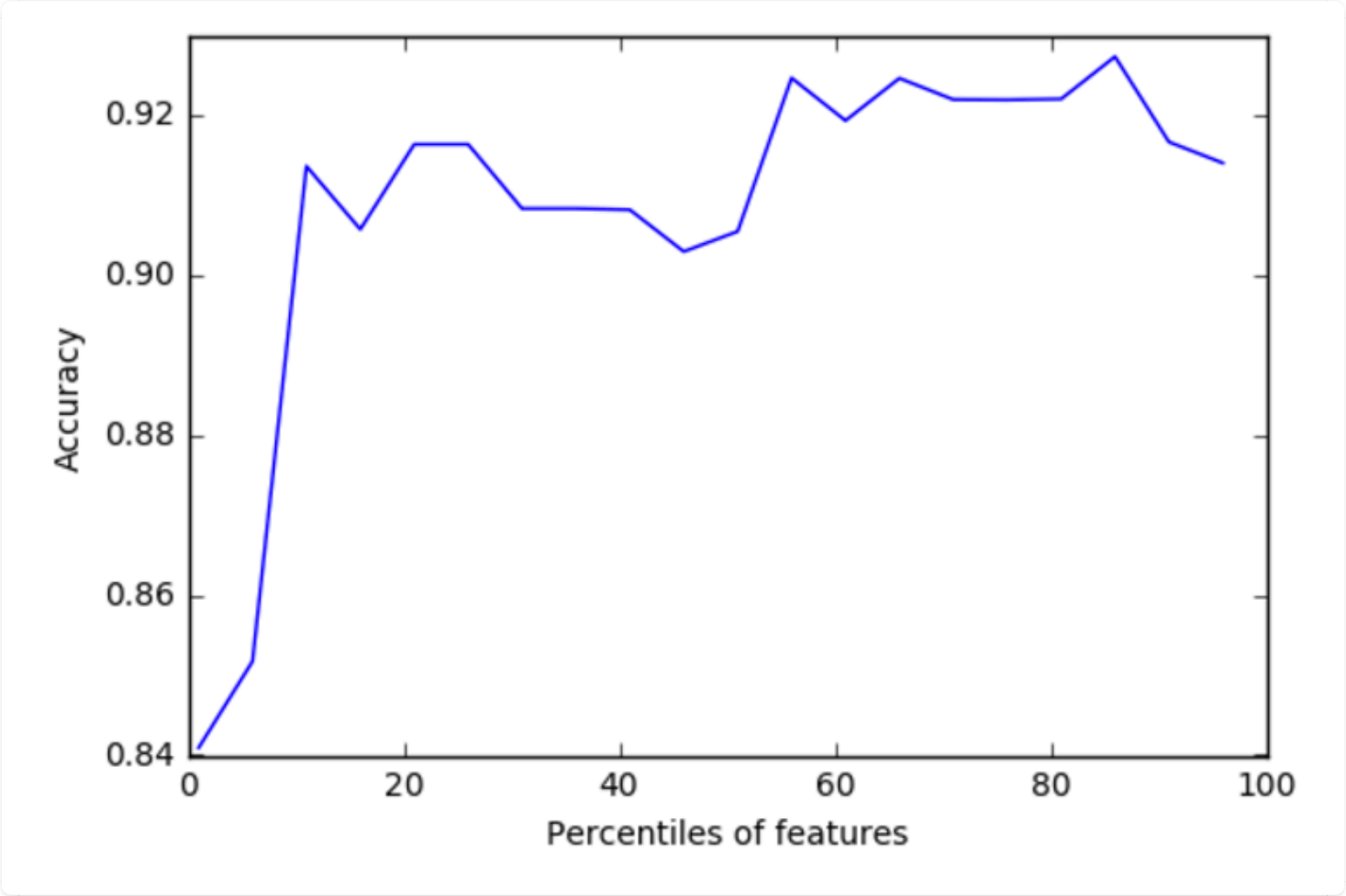


Fig.9 Accuracy by percentiles of features

The score of selected features can be **0.984**. This is a better results but it is **better than benchmark** of 0.975 Although accuracy can not meet needs, the method indeed provide ways of reduce running time and space.

Using feature selection also affects max_depth and min_sample_split in someway. I do think this can be useful for improve decision tree model in pratice.

Then I will apply deep learning method which is CNN, to further improve the accuracy in TensorFlow for refinement.

3.3.2 CNN of TensorFlow refinement

Since I already noticed that the accuracy of basic linear method is not superior to benchmark, I try to apply convolution neuron network on this dataset by taking each line of data as a slice of image which contains all information to get accuracy.

Firstly, I initialize weight and bias of model as linear model do. Taking zero would lead same weight into consideration, I give weight a random matrix by function tf.truncated_normal() and give it a shape. Then I create basis with normal constant numbers as it will be trained with weight and also get different numbers to avoid identical numbers.

Secondly, I give convolution and pooling same opreation as tutorial[14] introduced for set same convolution boundarries and stride size. This could make output same size of input size and a 2x2 plain old max pooling block.

Then I construct first layer which consist of convolution followed by max pooling. The convolution will compute 30 newly created features for each 5x5 patch. Weight tensor will have a shape of [5, 5, 1, 30]. The first two dimensions are patch size, the next is the number of input channels, and the last is the number of output channels. I will also have a bias vector with a component for each output channel.

With applying of CNN, I reshape data to 4d tensor, with the second and third dimensions corresponding to image width and height, and the final dimension corresponding to the number of diagnosis numbers which are two including benign and malignant.

Following that I construct my second layer which contains 60 features for each 5x5 patch. Based on the pattern of 'SAME' pooling, the image size has been reduced to 2x2. I add a fully-connected layer with 600 neurons to allow processing on the entire image. I reshape the tensor from pooling layer into a batch of vectors, multiply by a weight matrix, ass a bias, and apply a RELU.

To reduce overfitting, I apply dropout before the readout layer. I created a placeholder for the probability that a neuron's out put is kept during dropout. This allows us to turn dropout on during training, and it turn it off during testing. TensorFlow's tf.nn.dropout op automatically handles scaling neuron outputs in addition to masking them, so dropout just works without any additional scaling.

Besides, I also add l2 loss function in each layer to reduce overfitting. When model was training, the change in each weight is multiply by 0.001 for robust of initial model and sensitity training times.

Finally, I added a layer, just like for the one layer softmax regression above. When the classifier numbers are two, the softmax function is equal to matmul calculation.

To test how well the model do at this time, I use 200 data as a batch to train and test model. I run model for training 20000 times and testing 3000 times. Final accuracy are averaged for results tested.

Difference are:

> *Steepest gradient descent optimizer with more sophisticated ADAM optimizer.*
> *I include the additional parameter keep_prob in feed_dict to control the dropout rate.*
> *I add l2 loss function to each layer for robustness.*

```
step 200, testing accuracy 1.000000
step 400, testing accuracy 0.985000
step 600, testing accuracy 0.990000
step 800, testing accuracy 0.975000
step 1000, testing accuracy 0.985000
step 1200, testing accuracy 0.935000
step 1400, testing accuracy 0.985000
step 1600, testing accuracy 0.965000
step 1800, testing accuracy 0.970000
step 2000, testing accuracy 0.975000
step 2200, testing accuracy 0.985000
step 2400, testing accuracy 0.980000
step 2600, testing accuracy 0.980000
step 2800, testing accuracy 0.975000
step 3000, testing accuracy 0.975000
Average CNN testing accuracy is 0.976682.
```

Fig.10 Testing accuracy of TensorFlow CNN

The final average CNN testing accuracy is **0.977**which is **better than benchmark**.

4. Results

4.1 Model Evaluation and Validation

When I try to improve decision tree model, I separate data into training, testing and validation sets. I convert all data into list and reorder the feature by alphabet. Then I use feature selection to choose optimal percentile and apply this percentile on decision tree. Successfully, the model shows 0.984 accuracy.

Accuracy of DT: 0.983870967742

	precision	recall	f1-score	support
Benigh	1.00	0.98	0.99	85
Malignant	0.95	1.00	0.97	39
avg / total	0.98	0.98	0.98	124

Fig.11 Confusion matrix of feature selected decision tree.

At the beginning of applying TensorFlow, I separate dataset into training, testing and validation sets. I choose data from 1 to 450 to train and test while set last 45 data to validation set since this data are balance and collected randomly. I used training set with every 200 random arrays to train model by stochastic training. After training and testing process, I used validation set to get the accuracy of prediction and it turn out to be 0.977 which is good.

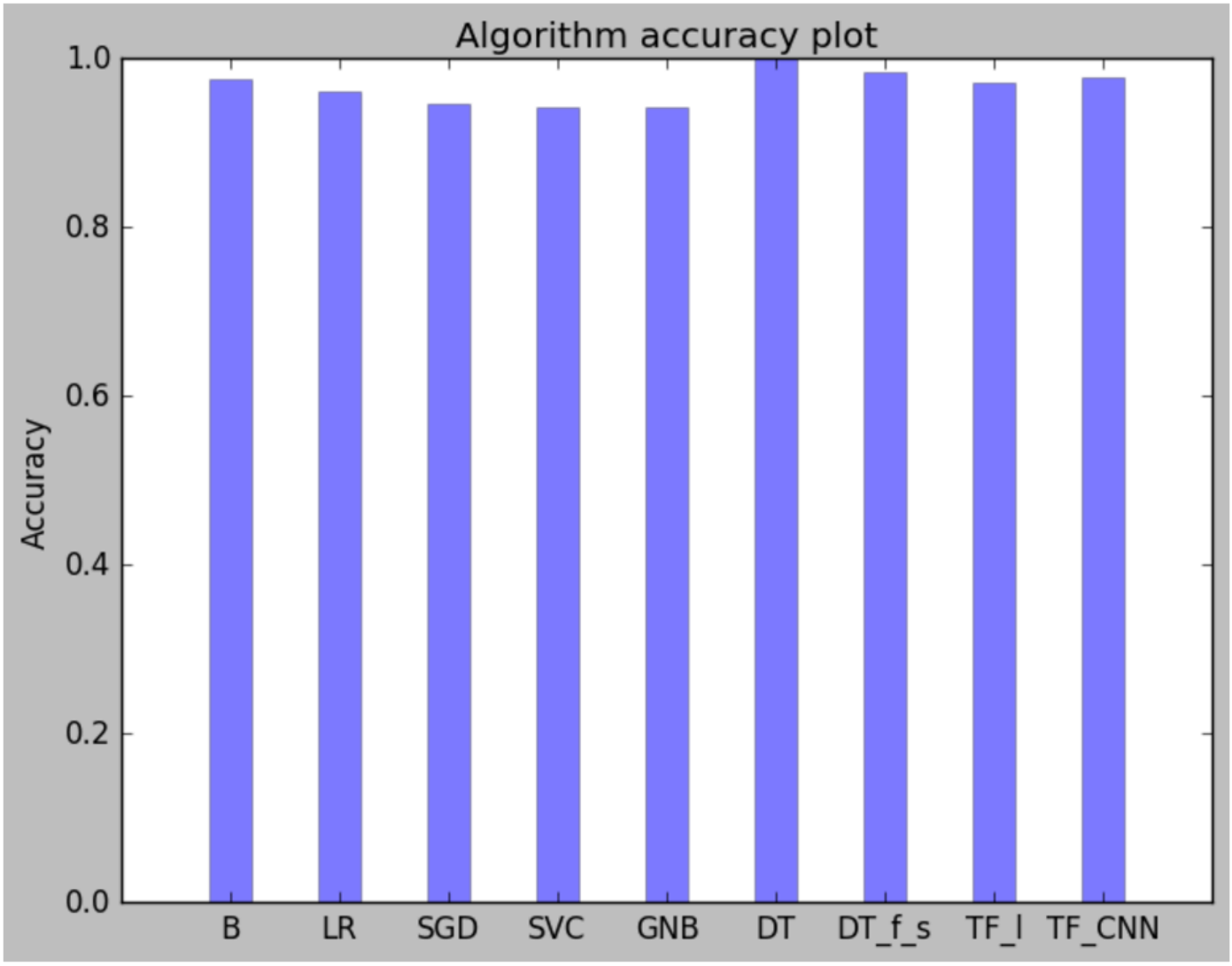


Fig.12 Accuracy of each model

4.2 Justification

In summary, I conclude all accuracy a table:

Benchmark	LogisticRegression	SGD	SVC	GaussianNB
0.975	0.960	0.946	0.941	0.941

DecisionTree	DecisionTree_feature_selection	TensorFlow_linear	TensorFlow_CNN
1	0.984	0.972	0.977

As table showed that, there are only 2 method is effectively superior to the benchmark the dataset provided, 0.975. The winning method are DecisionTree_feature_select and TensorFlow. Highest accuracy comes from DecisionTree_feature_select which is 0.984.

5. Conclusion

5.1 Reflection

Interesting aspects:

> *Although features are only different aspects of characteristics of cancer cell images, accuracy suppose there are some linear relationship between all features and diagnosis.*
>
> *Besides the suppose of linear and non-linear relationship, TensorFlow use tensors to mimic the neural network and solve this problem easily.*

Difficult aspects:

> *Data computed by TensorFlow should include correct shape of matrix calculation. Vectorize of numbers do make me confused at the beginning.*
>
> *CNN is hard to understand for starter of my level.*
>
> *I should spend more time on how to choose method and improve performance of algorithms.*

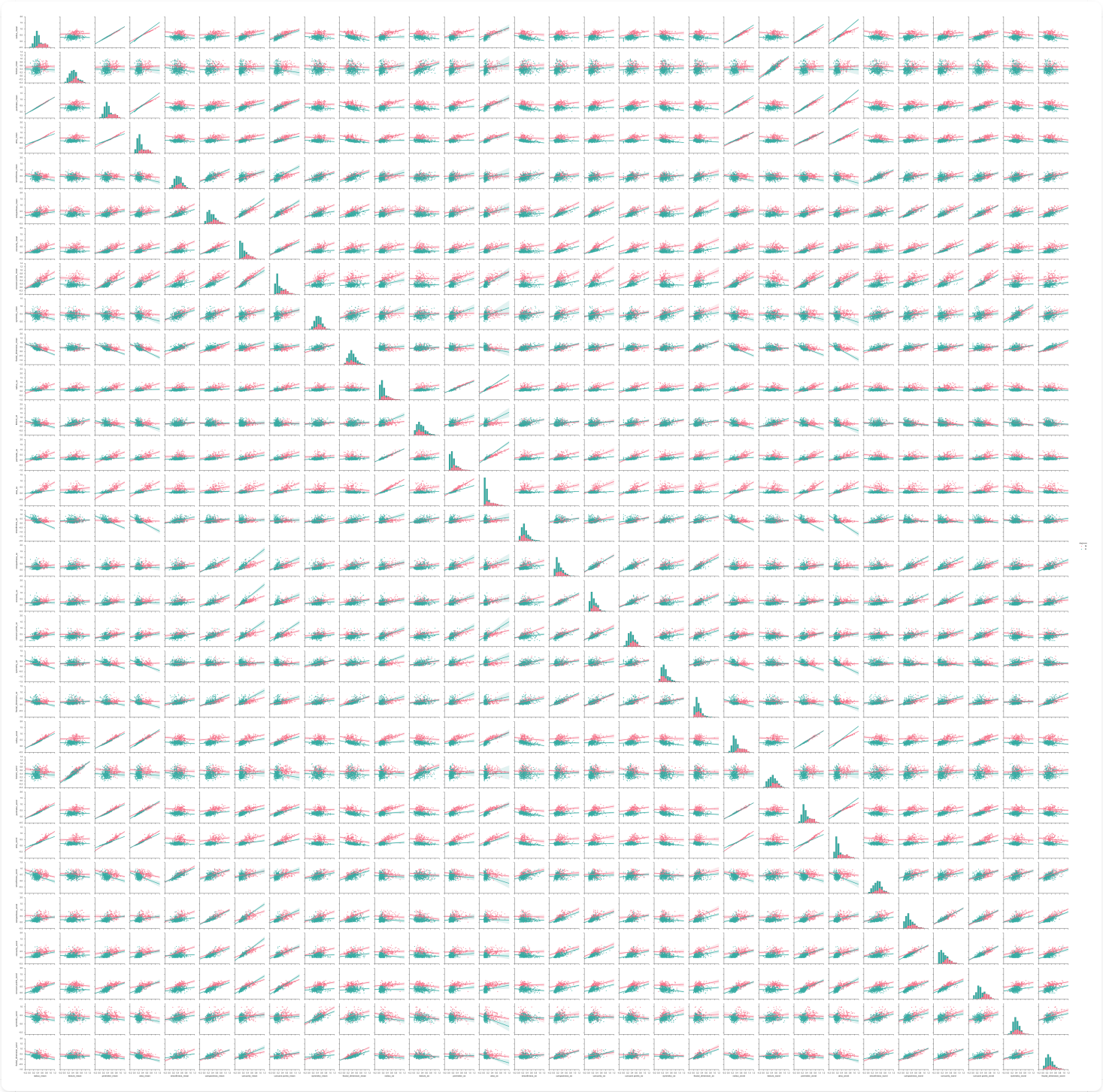
5.2 Improvement

In this problem, there may be more algorithm to apply on this dataset while I just use several of them. What is more, there are different parameters to define the process algorithm could process and I should try this.

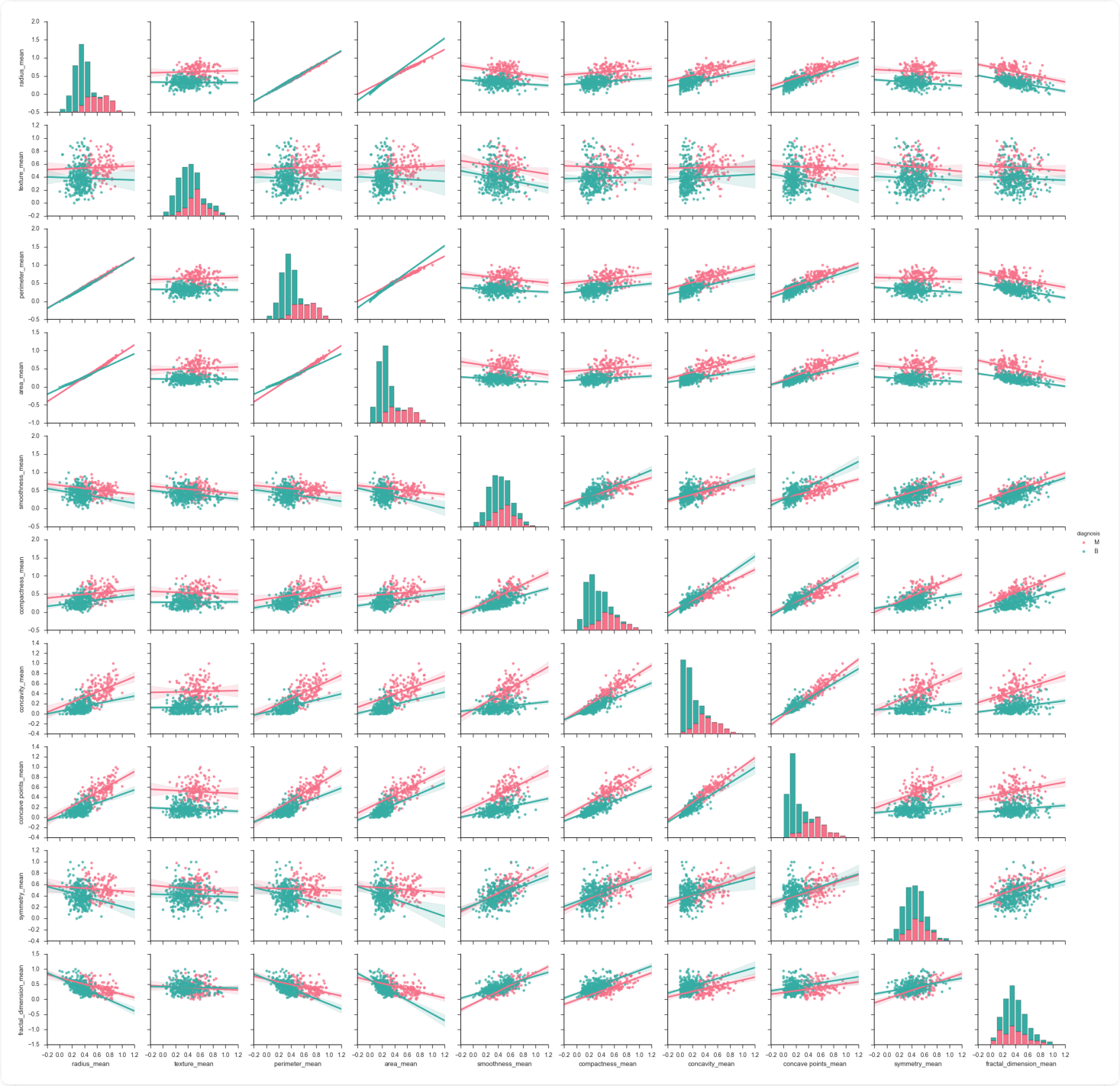
To define this problem is binary classification, one can use limited method. However, if anyone try to find the special connection of features between features or to select features and combine them, it is still another problem to solve.

In summary, I successfully apply TensorFlow on this dataset and produce a perfect match with 0.977 accuracy while that is 0.984 for decision tree after I made feature selection. I suppose this scale of data is enough to follow a patient and give accurate advice though the larger dataset is the better model will be. Also, there are a bright future to train model with larger dataset and get what I can know to support health industry future development to make a better world.

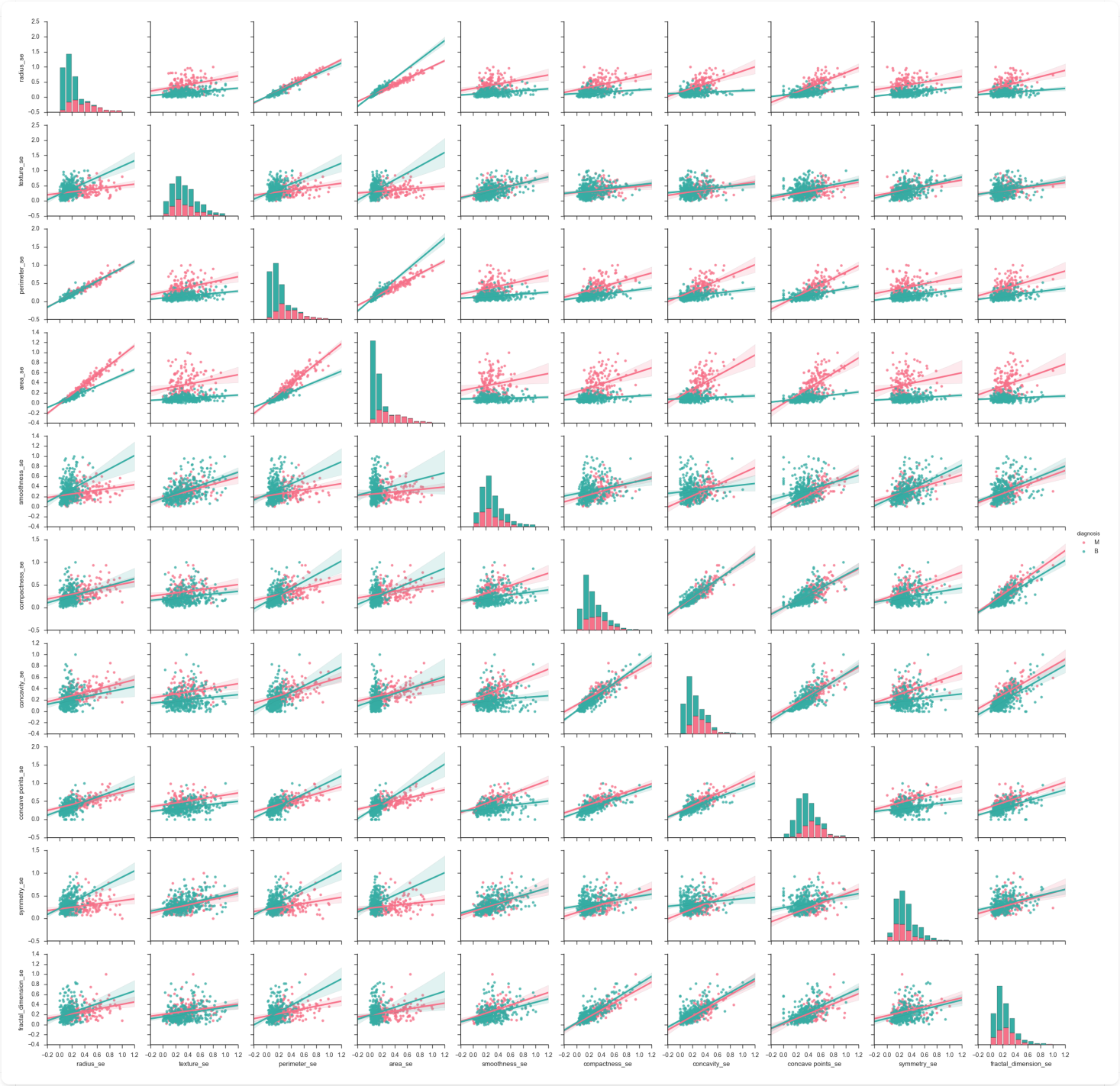
6.Supplementary



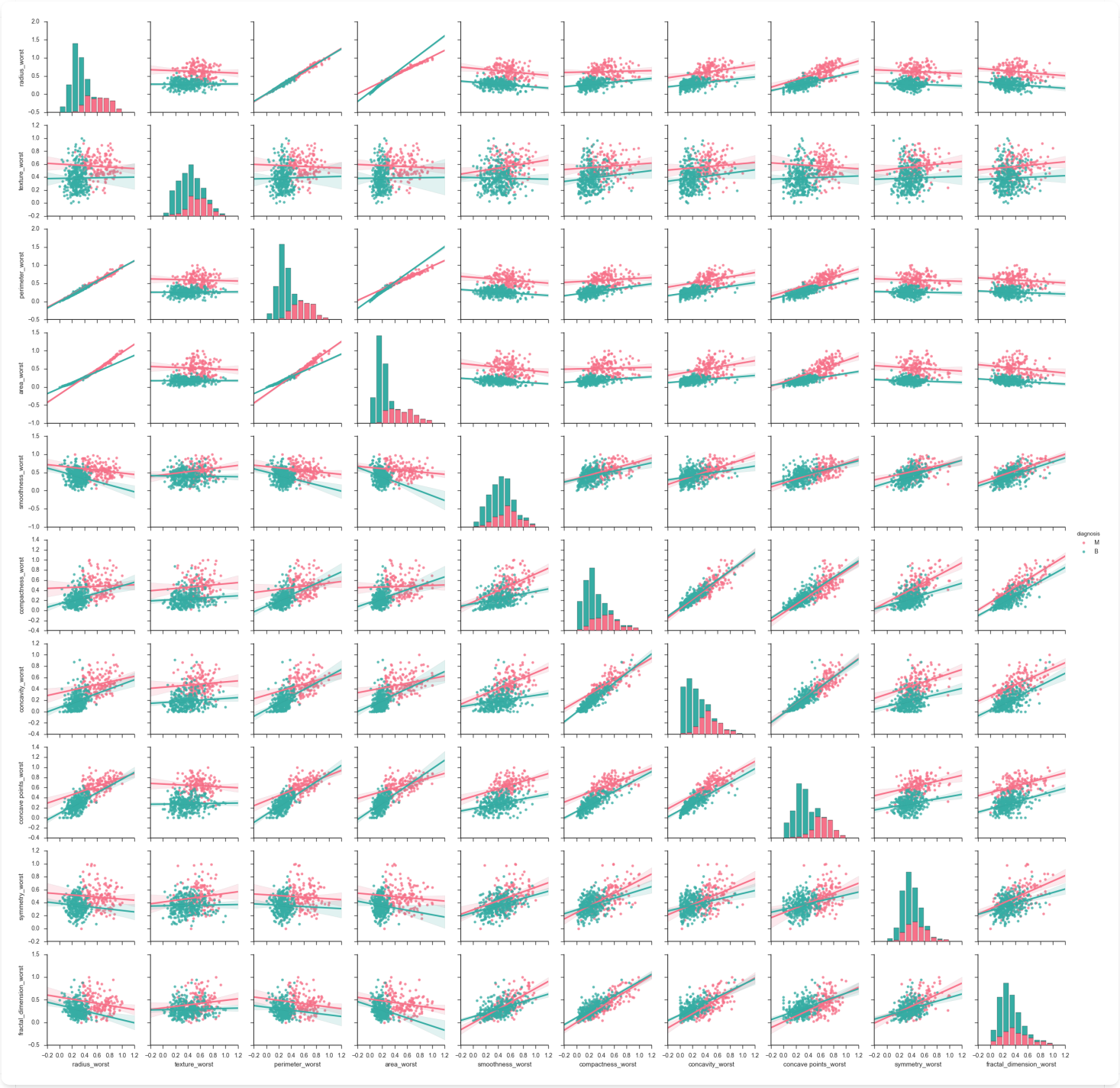
Sfig.1 Outliers_removed_30 features



Sfig.2 Outliers_removed_first_10 features



Sfig.3 Outliers_removed_second_10 features



Sfig.4 Outliers_third_10 features

---

1. Classification Accuracy is Not Enough: More Performance Measures You Can Use ↩
2. Outlier https://en.wikipedia.org/wiki/Outlier; http://stackoverflow.com/questions/23199796/detect-and-exclude-outliers-in-pandas-dataframe ↩
3. Binary classification, https://en.wikipedia.org/wiki/Binary_classification ↩
4. Normalization https://en.wikipedia.org/wiki/Normalization_(statistics)#cite_note-Dodge-1 ↩
5. Y. Yang, X. Liu, "A re-examination of text categorization", Proc. ACM SIGIR Conference, pp. 42–49, (1999). ↩

6. sklearn.linear_model.LogisticRegressi, http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression ↩
7. Stochastic Gradient Descent, http://scikit-learn.org/stable/modules/sgd.html ↩
8. Support vector machine, https://en.wikipedia.org/wiki/Support_vector_machine ↩
9. Naive Bayes, http://scikit-learn.org/stable/modules/naive_bayes.html ↩
10. Decision Trees, http://scikit-learn.org/stable/modules/tree.html ↩
11. Feature selection, https://en.wikipedia.org/wiki/Feature_selection ↩
12. Softmax Regression, http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/ ↩
13. "Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation". DeepLearning 0.1. LISA Lab. Retrieved 31 August 2013. ↩
14. TensorFlow official handbook https://www.tensorflow.org/versions/r0.12/resources/uses.html ↩