Capstone Project

Machine Learning Engineer Nanodegree

Derek Cheng

April 10, 2018

Definition

Project Overview

Diabetes is a disease that occurs when your blood glucose, also called blood sugar, is too high. Over time, having too much glucose in your blood can cause health problems, such as heart disease, nerve damage, eye problems, and kidney disease. Diabetes is caused mainly due to the consumption of highly processed food, bad consumption habits, and so on. According to WHO, almost half of all deaths attributable to high blood glucose occur before the age of 70 years, and diabetes will be the seventh leading cause of death in 2030.

Since diabetes is also an inherited disease in my family, I would like to know whether machine learning techniques can be utilized to predict the occurrence of diabetes. In this project, I will use the famous "Pima Indians Diabetes Database" dataset to assess the possibility of this goal. The "Pima Indians Diabetes Database" dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases, and provided by the UCI Machine Learning Repository (can also be downloaded at Kaggle[1]). The datasets consist of several medical measurable variables that can be used as machine learning model predictors.

By building a prediction model for diabetes, we may have further understanding on which factors are mostly related to this kind of disease, and therefore find a way to prevent or delay the onset of diabetes beforehand.

[1] https://www.kaggle.com/uciml/pima-indians-diabetes-database

Problem Statement

The goal is to use an off-the-shelf diabetes dataset to establish a binary classification model that can correctly predict the whether a person has diabetes or not. The model metric score should at least exceed the benchmark model and be able

to answer the questions such as “which factors are mostly related to the onset of diabetes” and “what the probability of getting diabetes is if the predictor values are given?” The tasks involved in the workflow of this project can be summarized as the following:

1. Download the “Pima Indians Diabetes Database” dataset and perform data exploration to gain basic understanding about the potential features.
2. Perform data cleaning to handle invalid data values.
3. Do some basic feature engineering and feature scaling if the scales of the features are inconsistent.
4. Train classifiers based on several machine learning algorithms and perform model selection to choose the best model that performs better with the diabetes data set with default parameters. The machine learning algorithms considered at this phase are Logistic Regression, K-Nearest Neighbors, Support Vector Classifier, Gaussian Naive Bayes, Random Forest and Gradient Boosting.
5. After selecting the best classification model, do feature selection to select which features most importantly affect the performance of the model and get rid of the features that do not improve the model.
6. Model hyper-parameter tuning: perform an exhaustive search like the grid search technique to further increase the metric score of the model.

Metrics

Because our dataset shows a little imbalanced between two classes. It would be unsuitable to use the prediction accuracy for quantifying the performance of the benchmark model and the solution model. On the contrary, Matthews correlation coefficient (MCC)[2] would be a more suitable measure of the quality of binary (two-class) classifications. It considers true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes.

Two binary variables are considered positively associated if most of the data falls along the diagonal cells. In contrast, two binary variables are considered negatively associated if most of the data falls off the diagonal. If we have a 2×2 table for two random variables x and y as follows:

	$y = 1$	$y = 0$	Total
$x = 1$	n_{11}	n_{10}	n_{1*}
$x = 0$	n_{01}	n_{00}	n_{0*}
Total	n_{*1}	n_{*0}	n

where n_{11}, n_{10}, n_{01}, n_{00}, are non-negative counts of numbers of observations that sum to n, the total number of observations. The MCC that describes the association of x and y is:

$$\phi = \frac{n_{11}n_{00} - n_{10}n_{01}}{\sqrt{n_{1*}n_{0*}n_{*0}n_{*1}}}$$

The MCC is in essence a correlation coefficient value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction[3].

[2] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html

[3] https://en.wikipedia.org/wiki/Phi_coefficient

Analysis

Data Exploration

The "Pima Indians Diabetes Database" datasets consist of several medical predictor variables including:

- **Pregnancies**: Number of times pregnant – It is reported that "gestational diabetes mellitus" which is a form of diabetes that occurs only during pregnancy, would let some women continue to have high blood glucose levels after delivery[1].
- **Glucose**: Plasma glucose concentration – diabetes is a group of metabolic disorders in which there are high blood sugar levels over a prolonged period, and therefore glucose would be the most direct indicator of diabetes.

- **BloodPressure**: Diastolic blood pressure (mm Hg) - Having diabetes increases the risk of developing high blood pressure and other cardiovascular problems, because diabetes predisposing the arteries to atherosclerosis[2].
- **SkinThickness**: Triceps skin fold thickness (mm) - Skin thickness is primarily determined by collagen content and is increased in insulin-dependent diabetes mellitus (IDDM)[3].
- **Insulin**: 2-Hour serum insulin (mu U/ml) - Diabetes is due to either the pancreas not producing enough insulin or the cells of the body not responding properly to the insulin produced[4].
- **BMI**: Body mass index (weight in kg/(height in m)^2) - Increased BMI was associated with increased prevalence of diabetes mellitus[5]
- **DiabetesPedigreeFunction**: Diabetes pedigree function - a synthesis of the diabetes mellitus history in relatives and the genetic relationship of those relatives to the subject. It utilizes information from a person's family history to predict how diabetes will affect that individual[6].
- **Age**: Age (years) – Though the age at which someone develops the condition depends on many differing factors, it has been reported that age greatly increases the chances of developing type II diabetes.
- **Outcome**: Class variable (0 or 1)

The table below are the first five samples in this dataset:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

We can see that Patient 3 has a measured skin thickness of zero, which is not reasonable in real world. In addition, Patient 0,1, and 2 all have zero insulin values, which are also probably due to incorrect measurements.

The descriptive statistics of this dataset can also be summarized below:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	724.000000	724.000000	724.000000	724.000000	724.000000	724.000000	724.000000	724.00
mean	3.866022	121.882597	72.400552	21.443370	84.494475	32.467127	0.474765	33.350
std	3.362803	30.750030	12.379870	15.732756	117.016513	6.888941	0.332315	11.765
min	0.000000	44.000000	24.000000	0.000000	0.000000	18.200000	0.078000	21.000
25%	1.000000	99.750000	64.000000	0.000000	0.000000	27.500000	0.245000	24.000
50%	3.000000	117.000000	72.000000	24.000000	48.000000	32.400000	0.379000	29.000
75%	6.000000	142.000000	80.000000	33.000000	130.500000	36.600000	0.627500	41.000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000

From the table above, we can see that the data distributions of **Pregnancies**, **Glucose**, **BloodPressure**, and **BMI** are reasonable in that their standard deviation values are relatively smaller than their mean values, and their 50% values are close to their mean values as well. On the other hand, the data distributions of **SkinThickness**, **Insulin**, and **DiabetesPedigreeFunction** are much more abnormal because their standard deviation values are too large compared with their mean values, and the data may be skewed based on their quantiles.

Several constraints were placed on the selection of these instances from a larger database. All patients here are females at least 21 years old of Pima Indian heritage. There are 768 persons contained in this dataset, 500 are labeled as 0 (non-diabetic) and 268 as 1 (diabetic), so the two classes are a little imbalanced, which can be seen in **Figure 1** (in Exploratory Visualization section). Since the number of dataset is limited, the whole data of the two classes will be utilized even though they are imbalanced, and the whole dataset will be split into training and testing set with a test size of 0.2. During the model training, 10-fold cross validation will be used to efficiently use the limited data in training set and provide a more accurate estimate of out-of-sample accuracy. After the training and hyper-parameter tuning, the previously separated testing set will be used to evaluate the model performance by calculating the evaluation metrics.

[1] https://www.diabetesaustralia.com.au/gestational-diabetes

[2] https://www.webmd.boots.com/diabetes/guide/diabetes-bp

[3] Collier, A., Patrick, A. W., Bell, D., Matthews, D. M., Maclntyre, C. C., Ewing, D. J., & Clarke, B. F. (1989). Relationship of skin thickness to duration of diabetes, glycemic control, and diabetic complications in male IDDM patients. Diabetes Care, 12(5), 309-312.

[4] Gardner, D. G., Shoback, D., & Greenspan, F. S. (2007). Greenspan's basic & clinical endocrinology. McGraw-Hill Medical,.

[5] Bays, H. E., Chapman, R. H., & Grandy, S. (2007). The relationship of body mass index to diabetes mellitus, hypertension and dyslipidaemia: comparison of data from two national surveys. International journal of clinical practice, 61(5), 737-747.

[6] Shanker, M., Hu, M. Y., & Hung, M. S. (2000). Estimating probabilities of diabetes mellitus using neural networks. SAR and QSAR in Environmental Research, 11(2), 133-147.

Exploratory Visualization

The figure shown below illustrate the number of non-diabetic and diabetic patients in the dataset. It can be seen that the number of non-diabetic persons is almost twice as diabetic persons, which indicates the imbalance of the two groups. Therefore, it would be unsuitable to use the prediction accuracy for quantifying the performance of our models.

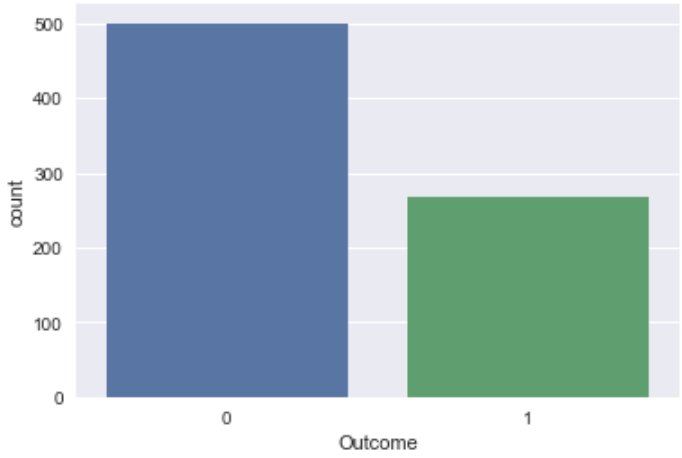


Fig. 1 The number of non-diabetic and diabetic persons in “Pima Indians Diabetes Database” dataset

Figure 2 shows how the data distribution of the features in the two groups. The distributions of **Age** and **Glucose** show great difference between the two groups, and therefore these features would be helpful for discriminating whether a person is diabetic or not. In addition, there are some zero values in **BMI**, **BloodPressure**, and **Glucose**, which can be viewed as outliers since normal person would not have zero values of these measures. We will perform data cleaning to get rid of thses outliers before building our models.

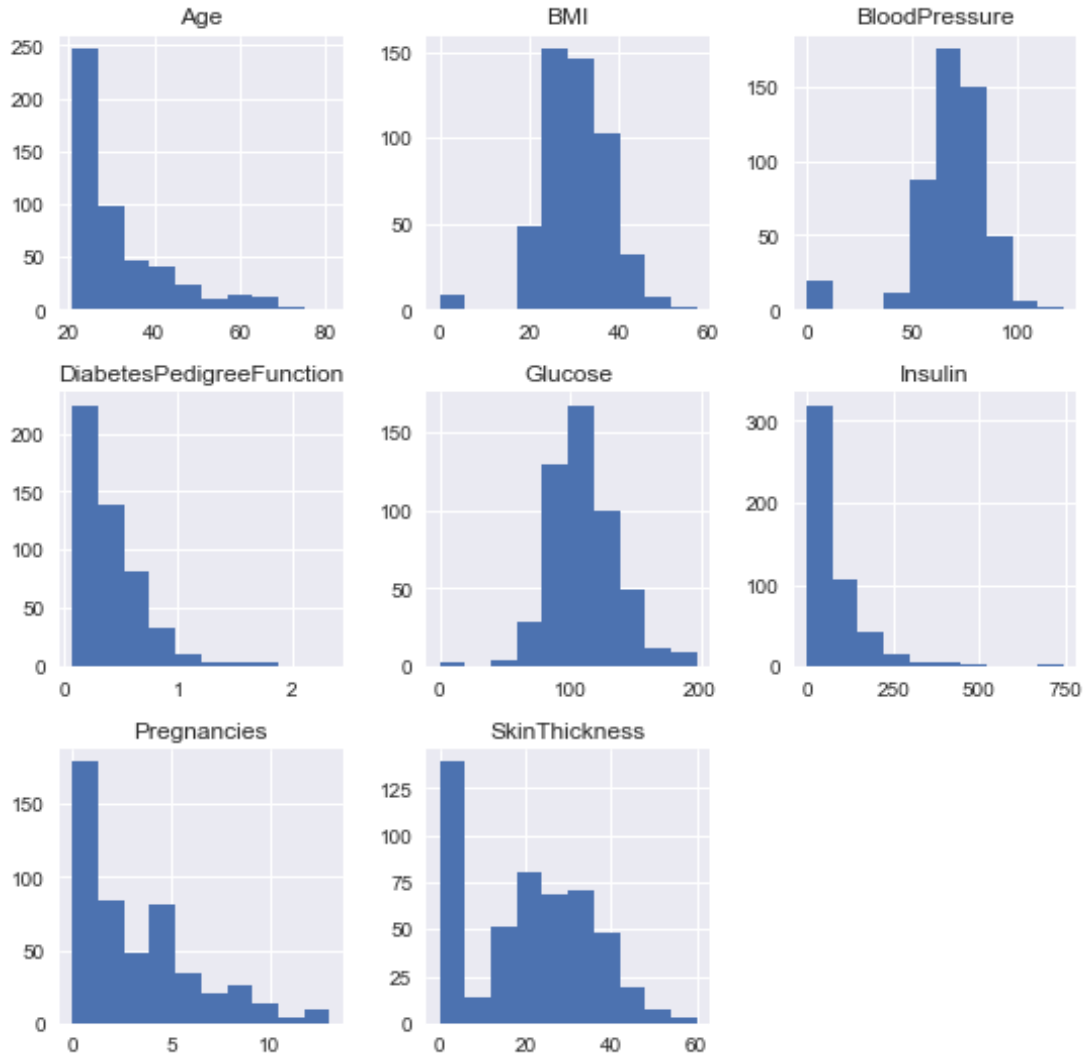


Fig. 2-1 Data distribution of the features in non-diabetic persons

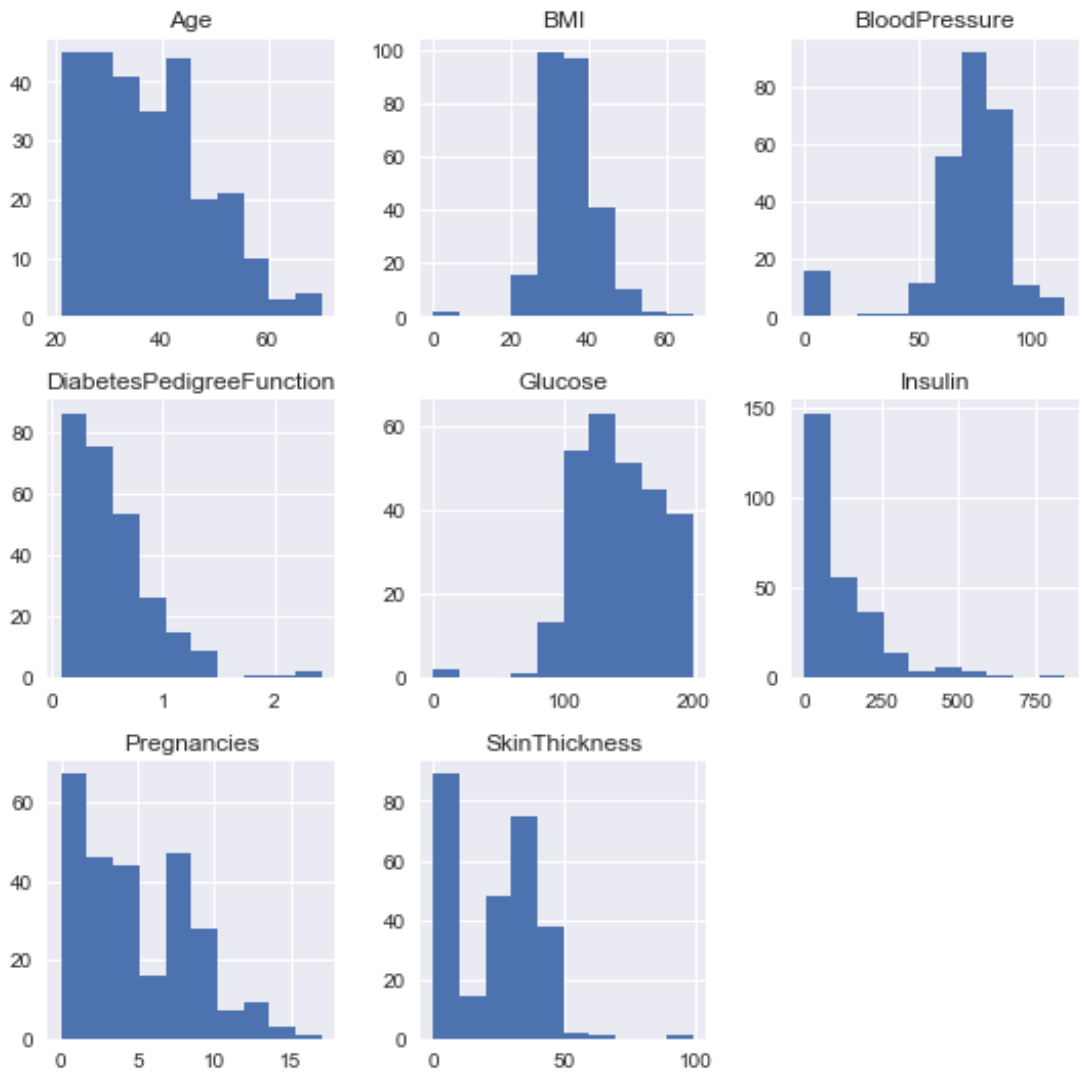


Fig. 2-2 Data distribution of the features in diabetic persons

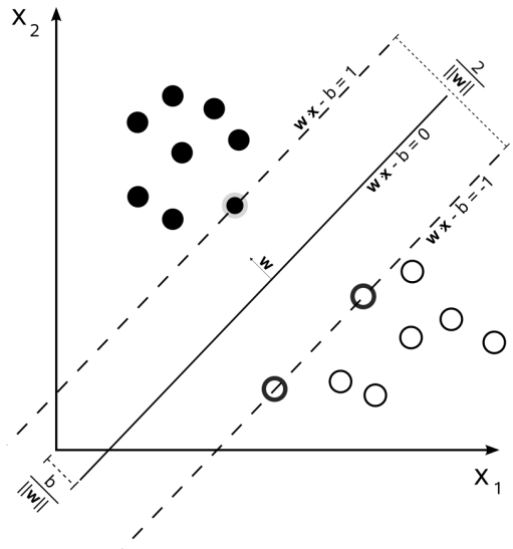
Algorithms and Techniques

In this project, several classifiers, including **Logistic Regression**, **K-Nearest Neighbors**, **Support Vector Classifier**, **Gaussian Naive Bayes**, **Random Forest** and **Gradient Boosting** are tested for their performance on predicting diabetes. These classifiers are the most popular ones that are used to evaluate baseline predicting performance at first. The basic properties of these model can be summarized as follows:

- **Logistic Regression:** The algorithm is usually "the first model" for binary classification problems. It outputs probabilistic interpretation of the two groups, and chooses the class that has the biggest probability. The algorithm can be regularized to avoid overfitting.

- Parameters: Regularization penalty type, regularization strength, class weight, intercept, solver type.

- **K-Nearest Neighbors:** The *k*-NN algorithm is among the simplest of all machine learning algorithms. An object is classified by a majority vote of its neighbors. Since our dataset is not very large and the number of dimension is relatively small, it is worthwhile to give it a try.
  - Parameters: number of neighbors, weight function for prediction, computing algorithms, distance metric.
- **Support Vector Classifier:** The principle of Support Vector Classifier (SVC) model is that the sample points are mapped in space so that the examples of the separate categories samples are divided by a clear gap (defined by two hyperplanes) 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. A simple illustration can be shown below:



Where $\vec{x}$ is the sample point vector, and $\vec{w}$ is the normal vector to the maximum-margin hyperplane which satisfies:

$$\vec{x} \cdot \vec{w} - b = 0$$

SVC can provide high accuracy, nice theoretical guarantees regarding overfitting. In addition, by introducing the kernel, SVC gains flexibility in the choice of the form of kernel functions and would be adapt to our case. Even though SVC doesn't scale well to large dataset, our dataset has only 768 instances and this should not be the problem.

- Parameters: Penalty parameter, kernel type, kernel coefficient, class weight.

- **Gaussian Naive Bayes:** Naive Bayes method is based on applying Bayes' theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector x_1 through x_n, Bayes' theorem states the following relationship:

$$\boldsymbol{P(y|x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n|y)}{P(x_1, \dots, x_n)}}$$

The naive independence assumption means that:

$$\boldsymbol{P(x_i|y, x_1, \dots, x_n) = P(x_i|y)}$$

The Bayes' theorem relationship is simplified to:

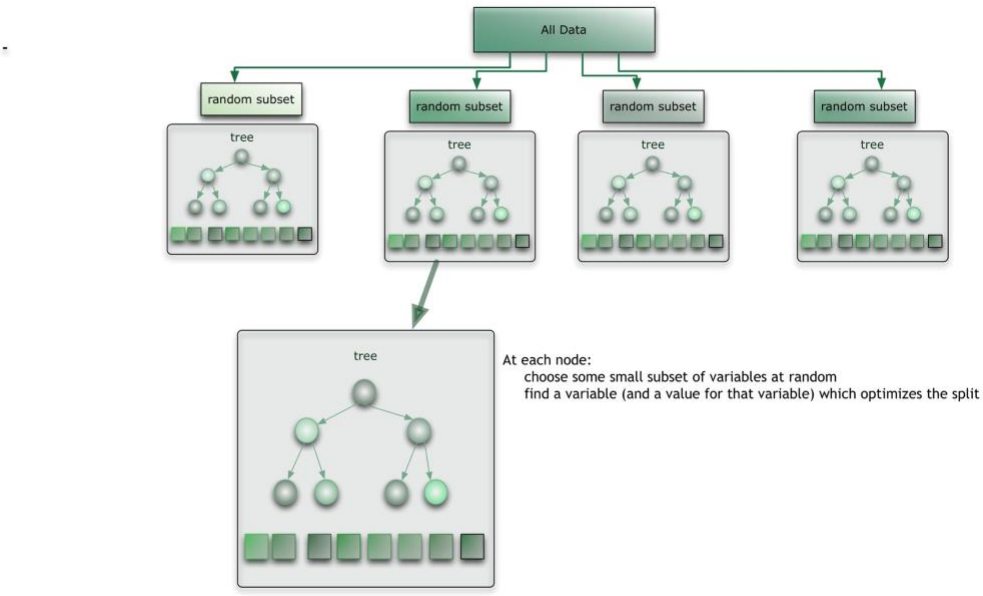
$$\boldsymbol{P(y|x_1, \dots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \dots, x_n)} \propto P(y)\prod_{i=1}^{n} P(x_i|y)}$$

And therefore, the corresponding Naïve Bayes classifier:

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, \boldsymbol{P(y)}\prod_{i=1}^{n} \boldsymbol{P(x_i|y)}$$

Naive Bayes classifiers can be extremely fast compared to more sophisticated methods. Since our training set is small, Naive Bayes (high bias/low variance classifiers) have an advantage over low bias/high variance classifiers, which tend to overfit.

- **Random Forest:** The random forest is an ensemble method that combine decision trees, which correspond to weak learners to form a strong learner. The notion of random forest can be illustrated as follows:



The algorithm of random forest is summarized below:

1 For some number of trees T, sample N cases at random with replacement to create a subset of the data. The subset should be about 66% of the total set.

2 At each node:

2.1 For some number m, m predictor variables are selected at random from all the predictor variables.

2.2 The predictor variable that provides the best split, according to some objective function, is used to do a binary split on that node.

2.3 At the next node, choose another m variables at random from all predictor variables and do the same.

Random forests are often the winner for lots of problems in classification. Random forest can be applied to a wide range of data types, even if the problems are nonlinear or involve complex high-order interaction effects. Since the features in our dataset are physical measures, which are correlated with complex interactions, random forest would be a good candidate to tackle this problem.

- Parameters: Number of trees in the forest, splitting criteria, number of features for splitting, maximum depth of the tree, the minimum number of samples required to be at a leaf node.

- **Gradient Boosting:** Unlike random forest, which is a Bagging ensemble technique that builds many independent predictors and combine them using some model averaging techniques, gradient boosting is an ensemble technique that the predictors are not made independently, but sequentially by adding weak learners using a gradient descent like procedure. The training steps of gradient boosting can be summarized as follows:

1 Fit a simple linear regressor or decision tree on data

2 Calculate error residuals.

3 Fit a new model on error residuals as target variable with same input variables.

4 Add the predicted residuals to the previous predictions.

5 Fit another model on residuals that is still left. and repeat steps 2 to 5 until it starts overfitting or the sum of residuals become constant.

Like random forest, gradient boosting can learn complex non-linear relationships. It often has higher performance ceilings with careful parameter tuning. Besides, it can also cope with small and imbalanced dataset.

- Parameters: Loss function, learning rate, number of estimators, maximum depth of the individual estimators.

Benchmark

Logistic regression is the model of choice in many medical data classification tasks, especially for binary classification problem. The power of the study in case of the logistic regression is between 0.80 to 0.857. The logistic regression model calculates the class membership probability for one of the two categories in the data set:

$$P(1|x,\alpha) = \frac{1}{1+e^{-(\alpha \cdot x)}}, \qquad P(0|x,\alpha) = 1 - P(1|x,\alpha)$$

Here $P(1|x,\alpha)$ is the dependence of the posterior distribution on the parameters α. The hyperplane of all points α satisfying the equation $\alpha \cdot x = 0$.

To build a simple logistic model for benchmark, I followed the procedure that provided at Kaggle[8]. First, it builds a correlation matrix to see which features are highly correlated with the target variable **Outcome**. A simplest choice is to the use the first two features (in this dataset, these are **Glucose** and **BMI**) who have the highest correlation. A simple logistic regression model then uses these two features achieve an average MCC score of 0.4368 with 10-fold cross validation.

[7] Agresti, A. (2013). Categorical data analysis. John Wiley & Sons.

[8] https://www.kaggle.com/mshirlaw/pima-indians-diabetes-simple-logistic-regression

Methodology

Data Preprocessing

The data preprocessing consists of the following step:

1. Check whether there are missing and null data points in the dataset.
2. Check whether there are unexpected outliers in the dataset.
3. After data cleaning, perform feature scaling so that the scale of each feature is consistent. In this way our learning models can handle data correctly.
4. Split the whole dataset into training and testing set with a test size of 0.2. During the training, training set is divided into 10-fold for cross validation.

When cleaning the data, we found that there are no missing or not available data in the "Pima Indians Diabetes Database". However, we found that there are many zero values existed in each feature, which are viewed as outliers. There are some possible ways to handle these values:

1. Ignore/remove these cases: In most cases it would mean losing valuable information. In this case **SkinThickness** and **Insulin** have a lot of invalid points, so we cannot take this strategy for them. But it might work for **BMI**, **BloodPressure**, and **Glucose**.
2. Put average/mean values: This might work for some data sets, but in our case putting a mean value to the **BloodPressure** column would send a wrong signal to the model.
3. Avoid using features: It is possible to not use the features with a lot of invalid values for the model. However, we would keep as much as information first to build our first model.

In this work, since our dataset is small, we decide to do as minor as modification to our dataset by just removing the data points whose **BMI**, **BloodPressure**, and **Glucose** are zero. After removing the outliers, our dataset has 724 data points left.

Implementation

The implementation process contains two main stages:

1. Training classifiers that we previously mentioned in Algorithm and Techniques section.
2. Perform feature selection by finding which features contribute most for predicting the onset of diabetes.

In the first stage, the 6 classifiers are trained by preprocessed data. Like we have done for benchmark model, the data were first split into training set and testing set, and then the training set was randomly divided into 10-fold for cross validation. A performance metric function *performance_metric(…)* was also implemented to help automatically calculate the average cross-validation MCC score. The 6 classifiers were trained by their default parameters suggested by *Scikit Learn* library with all features included. To facilitate the interpretation of the model performance, the we used the bar-plot API provided by *seaborn* library to show the MCC scores of each classifier, which can be illustrated as **Figure 3**:

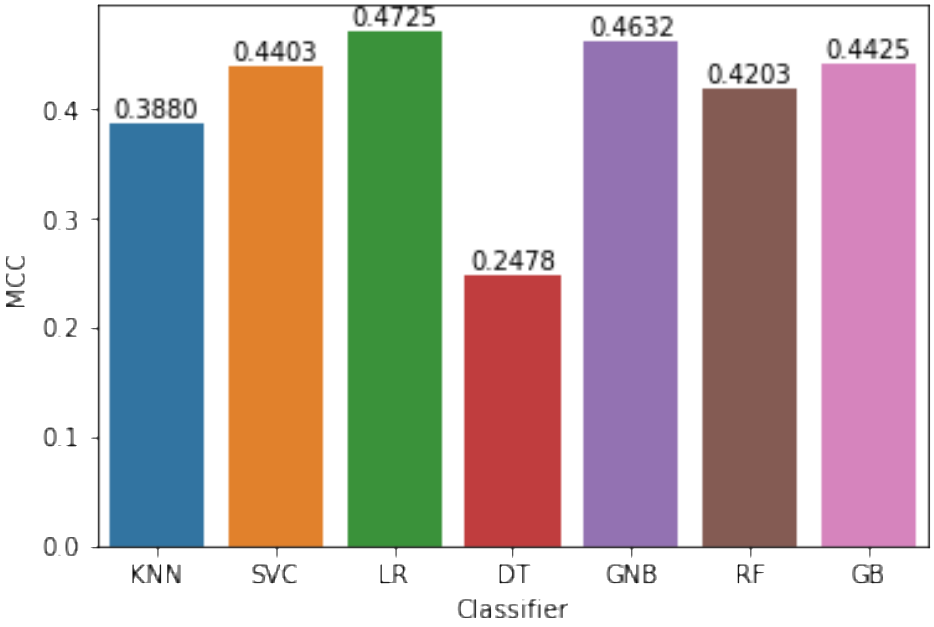


Fig. 3 Average MCC score of different trained classifers.

In the second stage, the Recursive Feature Elimination (RFE) mothed provided by *Scikit Learn* library was adopted to help find out the most import features for predicting the diabetes. This method was applied to the winner model in the first stage and a 10-fold cross validation was also adopted for evaluating model performance with different feature numbers. We plotted the MCC score versus the number of features using the basic *Matplotlib* API to determine how many features were needed for the following parameter fine-tuning stage, which can be shown in **Figure 4**:

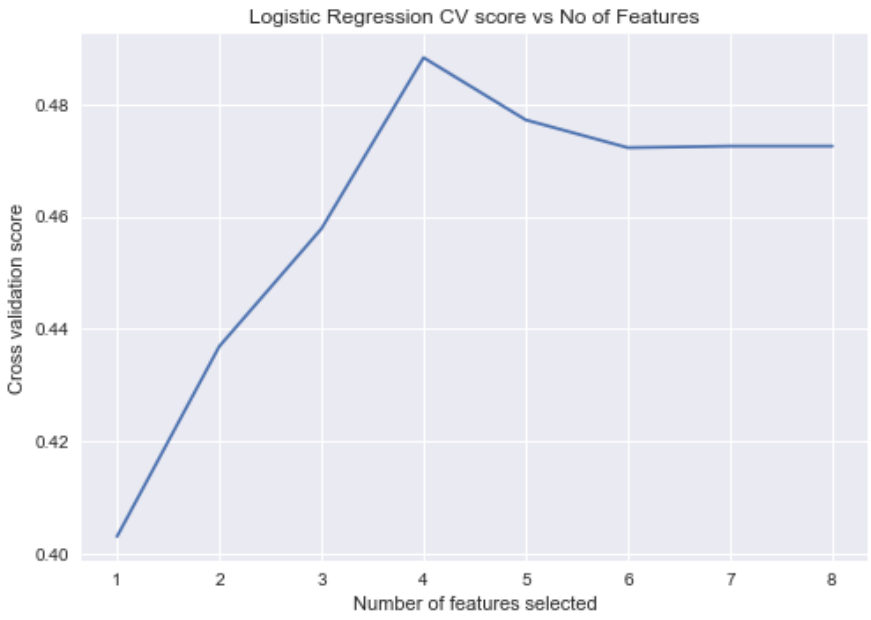


Fig. 4 Cross validation MCC score v.s. Number of features selected.

The implementation of the first stage is simple in that we could easily divide the training set in to 10 consecutive folds with *Scikit Learn' KFold* API and sequentially calculate the average cross validation MCC scores for different algorithms using *cross_val_score* function. The only difficulty of this implementation may be how to find a example for using these functions in *Scikit Learn's* document. Likewise, the implementation of the second stage is also straightforward by calling *Scikit Learn's* powerful *RFECV* API for the chosen model. However, as we will discuss in the Improvement section, the pitfall of the implementation workflow here is that we will find that the "winner" model in the first stage may not be generalized well to unseen data. Therefore, it would be a better flow if all the initial training and the subsequent feature selection procedure can be *pipelined* for every algorithm. I did not find a simple way for implementing pipelining, but there indeed exists a *Pipeline* API in *Scikit Learn* library, which would help achieve this goal easily.

Refinement

In the Benchmark section, the simple logistic regression model with only two features have achieved an average MCC score of 0.4368. For the initial logistic regression model, which is the winner among the all algorithms we tested in the Implement section, a better average MCC score of 0.4725 can be achieved, probably because all the features were included into the model and therefore the prediction power was enhanced. After performing feature selection, we found that by just using **Pregnancies**, **Glucose**, **BMI**, and **DiabetesPedigreeFunction**, we could further push the MCC score to 0.4883 with default model parameter settings. There should be some room to enhance model performance, in that we can fine-tune the model parameters to seek an even higher MCC score.

The fine-tuning procedure was accomplished by applying the grid search technique, which was implemented as *GridSearchCV* in *Scikit Learn* library. The philosophy of hyper-parameter combinations that used in the grid search can be described as follows:

- **C - Inverse of regularization strength:** smaller values specify stronger regularization. The main parameter that need to be determined. We apply three ranges of values:
  - C = 1 to C =10 with step of 0.5
  - C = 1 to C =100 with step of 10

- C = 0.1 to C =1 with step of 0.1

to find out the best possible solution.

- **Class weight:** It may work in case of class imbalance if it is set to 'balanced'.
  - Class weight values: None and 'balanced'
- **Penalty:** Regularization type for preventing overfitting.
  - Regularization values: 'l1' and 'l2'
- **Solver:** Even though our dataset is small and the 'liblinear' solver is suggested by *Scikit Learn*, other solvers like 'newton-cg', 'lbfgs', 'sag' were also included in the grid search in hope of finding better model performance.
  - Solver values: 'liblinear', 'newton-cg', 'lbfgs', and 'sag'
- **Constant intercept and the intercept scaling:** The intercept was turned on by default to work as a synthetic feature and a scaling factor that is indeed the intercept was also included for model fitting
  - Intercept scaling = 1 to 10 with step of 0.5

Even though the average cross validation MCC score of the final model is the same as that of the model with feature selection, the test MCC score of the final model (0.551) is higher than that of the model with feature selection (0.5138).

Results

Model Evaluation and Validation

During the development, 10-fold cross validation approach was used to evaluate the model performance. The final model and the hyper-parameters were chosen because they performed the best among all the grid-searched parameter combinations. The final model has the following parameters:

```
{'C': 1.0, 'intercept_scaling': 5.5, 'solver': 'liblinear', 'penalty': '
l1', 'multi_class': 'ovr', 'class_weight': 'balanced'}
```

The final parameters of the model seem to be appropriate. The C-value is normal (with default value 1.0), and the solver is 'liblinear', which is suitable for our small dataset. In addition, the class weight is also chosen as 'balanced' as expected because of the imbalance of our data.

To verify the robustness of the final model, a testing set that was unseen during the training was applied to evaluate the model ability of generalization. The results show that the test MCC score of the final model is higher than both the initial model and the model with feature selection, indicating that our final model is robust and can generalize well on unseen data.

Justification

The test MCC score was also used to compare the model performance of benchmark model and the final model. Unlike the initial model, who has higher average cross validation MCC score (0.4725) but lower testing MCC score (0.468) than the benchmark model (0.4368 and 0.4762, respectively), our final model has both higher average cross validation MCC score and testing MCC score (0.4883 and 0.551) than the benchmark model, showing that the final model is not overfitted and has supreme prediction capability on diabetes.

Conclusions

Free-Form Visualization

Figure 5 shows the average cross validation MCC score and test MCC score of the implemented models in this project. As we previously discussed, the average cross validation MCC score compared with the benchmark model could be improved by adding more features (initial model), performing feature selection, and fine-tuning the model parameters. However, the test MCC score showed that adding more features induce the risk of overfitting, and therefore a feature selection procedure before fine-tuning hyper-parameters is preferred even though there are only 8 features in the dataset.

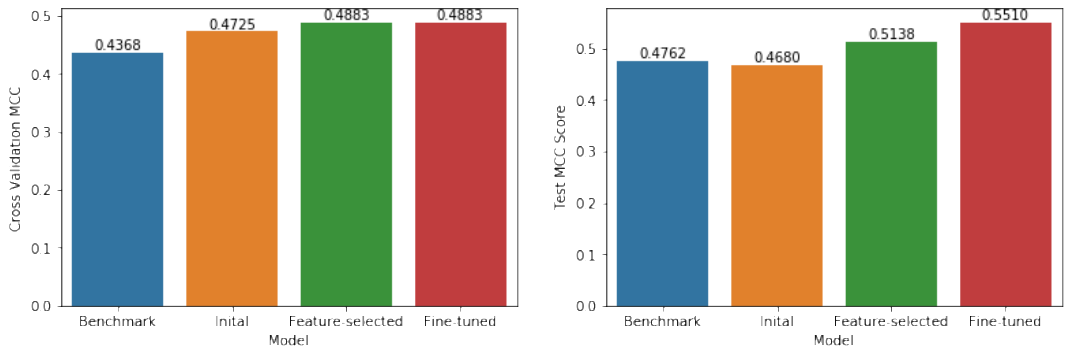


Fig. 5 Model performance comparison of **Benchmark**, **Initial**, **Feature-selected**, and **Parameter Fine-tuned** models.

Reflection

The process used for this project can be summarized using the following steps:

1. Came out an initial problem that interested me most, and then found out a public dataset at hand that can be easily assessed.
2. Downloaded and preprocessed the dataset.
3. To establish a benchmark model, searching for previous model built by other people with the most basic assumption and settings.
4. Train candidate classifiers with different algorithms, choosing one of them that perform best to build the initial model.
5. Perform feature engineering and hyper-parameter fine-tuning to further improve model performance.
6. Use the testing set that is initially separated before model training to check the model robustness on newly unseen data.

Throughout the whole process, I found the step 3 most interesting and a little tricky. The are many previous models using the "Pima Indians Diabetes Database" dataset can be found on the internet. The performance metrics of these models, however, simply rely on the model prediction accuracy, which is not that suitable for predicting diabetes since the two classes in this dataset are imbalanced. In addition, most of the methods used in building these models are not simple and clear enough for benchmarking. It took me a while to find a simple and reasonable logistic regression model at Kaggle that can act as the benchmark model.

The most interesting and exciting aspects of the project is the model selection phase - I got the chance of extensively surveying the properties and usages of different classification algorithms in *Scikit Learn* library. It surely prepares me the ability of quickly finding available methods to tackle different classification problems in the future.

Improvement

In this project, we determined the initial model by running different algorithms with their default parameter settings. It surprised me a little that the best candidate is still the logistic regression classifier. The subsequent procedures were conducted

based on this algorithm. However, we later found that even though the average cross validation MCC score of the initial model is the highest, it has a worse test MCC score compared with the benchmark model. It seems that the initial model was a little overfitted and maybe other algorithms, such as **Gaussian Naive Bayes** or **Gradient Boosting**, could generalize better and achieve superior test MCC scores. These algorithms also worth further feature engineering and hyper-parameter tuning to see whether they have better general model performance.

Moreover, other technique, such as stacking classifier that combine multiple model together, would be helpful to further improve the model performance. Finally, in this project, we use only MCC score for evaluating the model performance. Other performance metrics like F-score or precision-recall, which are suitable for imbalanced data, may also be included to help us better understand the model properties and make appropriate decisions on model fitting approaches.