

Machine Learning Engineer Nanodegree

Capstone Project Proposal

Bollam Raghavendra Reddy

February -1, 2019

Diabetes Prediction (Yes or No)

Proposal:

Pima Indians Diabetes Prediction

PIMA INDIANS DIABETES PREDICTION

Domain Background

The proposed project is from the medical background or domain of bioinformatics. It is called the pima indians diabetes dataset , a dataset whose participants are females aged 21 or above , of pima indian heritage .**Pima**, North American **Indians** who traditionally lived along the Gila and Salt rivers in Arizona, U.S. The problem aims at predicting , if a given person has diabetes or not, by evaluating several features present in the dataset . Considering in 1990, diabetes detection was a big issue, and it was a very common condition that had to addressed at any cost.IT is estimated that by 2030 , there will be 366 million people approximately, affected by diabetes mellitus .

The problem can be solved by looking at the suspected parameters and collecting data on those parameters, then predicting using supervised classification algorithms, the presence or absence of diabetes.

Original owners: National Institute of Diabetes and Digestive and Kidney Diseases

Donor of database: Vincent Sigillito (vgs@aplcn.apl.jhu.edu) Research Center, RMI Group Leader Applied Physics Laboratory The Johns Hopkins University Johns Hopkins Road Laurel, MD 20707 (301) 953-6231 (c) Date received: 9 May 1990

Problem Statement

The aim of this project is detection of diabetes in females of age 21 and above, pima indians heritage, by monitoring and recording several input features and reaching a conclusion, whether the patient has diabetes or not.

The problem can be evaluated with the help of a supervised classification algorithm, which would label

0 for a negative classification

1 for a positive classification.

The diagnostic, binary-valued variable investigated is whether the patient shows signs of diabetes according to World Health Organization criteria (i.e., if the 2 hour post-load plasma glucose was at least 200 mg/dl at any survey examination or if found during routine medical care). The population lives near Phoenix, Arizona, USA. Diabetes mellitus is a kind of metabolic diseases in which a person has high blood sugar. There are two general reasons for diabetes:

1- body does not produce enough insulin. Only 5-10% of people with diabetes have this form of the disease (type1).

2-cells do not respond to the insulin that is produced (type2).

Datasets and Inputs

The dataset used for this project is the pima-indians-diabetes.csv, which was picked from the website kaggle.com and original owners are National Institute of Diabetes and Digestive and Kidney Diseases

Number of instances-**768**

Number of attributes-**8 plus class**(all numeric)

feature1 -Number of times pregnant

feature2 -Plasma glucose concentration,2 hrs in an oral glucose tolerance test

feature3 - diastolic blood pressure (mm Hg)

feature4 - Triceps skin fold thickness(mm)

feature5 - 2 hr serum insulin (mu U/ml)

feature6 -Body mass index (weight in kg/(height in m)^2)

feature7 -Diabetes pedigree function

feature8 -Age(years)

feature9 – Target Class variable(0 or 1)

There are 500 non-diabetic patients (class = 0) and 268 diabetic ones (class = 1) for an incidence rate of 34.9%. Thus if you simply guess that all are non-diabetic, your accuracy rate is 65.1% (or error rate of 34.9%). All of these features bear importance

in predicting the class variable, intuitively, the feature 2,5 seem to have a strong relevance considering the domain knowledge. feature3 is important as diabetic patients usually have diastolic blood pressure 1-3 mm lower than normal. Older individual seem to have a higher risk of diabetes, it is difficult to contract diabetes at a young age because of the high metabolism, and everything else being normal. Diabetes can be hereditary , hence the diabetes pedigree function is an important feature because, it is a measure of occurrence of diabetes in relatives and blood related individuals to the patient.

The data is open – sourced and can be downloaded for education purpose with no citation. we can download the dataset from following link

<https://www.kaggle.com/dssariya/pima-indians-diabetes-data-set>

Solution Statement

The solution would be to take care of the missing variables, by looking at the mean and median of the dataset features, because good data is better than a good algorithm. The missing values in feature2 can be replaced by the mean of the dataset, it has 5 missing values. The missing values in feature 3, can be replaced by its mean or median, one can experiment with both , and check which yields better results. Feature 6 missing values can be replaced by its mean as well. However, in feature 4 and 5, using mean won't be appropriate because, it has 227 and 374 missing values respectively, so mean would be significantly affected and dragged down, so median is a better metric to be used in these two features.

After applying supervised classification algorithms on the cleansed data, we can look at the confusion matrix , precision and recall to find the correct and incorrect predictions.

We will run tests ,on a variety of algorithms, to find the one which combination yields the best result. An ideal solution would have predicted the correct cases of true positives and true negatives in the test set.

Benchmark Model

The benchmark model used a dataset, which removed the missing values, and performed a voted stacking method of two algorithms , the **svm** and the **backpropagation neural network** .

The benchmark model obtained an f1 score of 0.8804, and the precision and recall values for the negative and positive classification are as follows
precision(0:9206,1:0.7931), recall(0:9062,1:0.8214). The benchmark model had scaled the variables and applied an ensemble of three layer hierarchy multi-classifier. A voted ensemble of svm and BP NN , tuning the kernel functions of svm and

increasing the running time ,no of iterations of neural network , gave it an **accuracy of 0.8804**.

Evaluation Metrics

The **f1_score** is one of the evaluation metrics used in the benchmark model as well as the proposed project. It is created by finding the the harmonic mean of precision and recall.

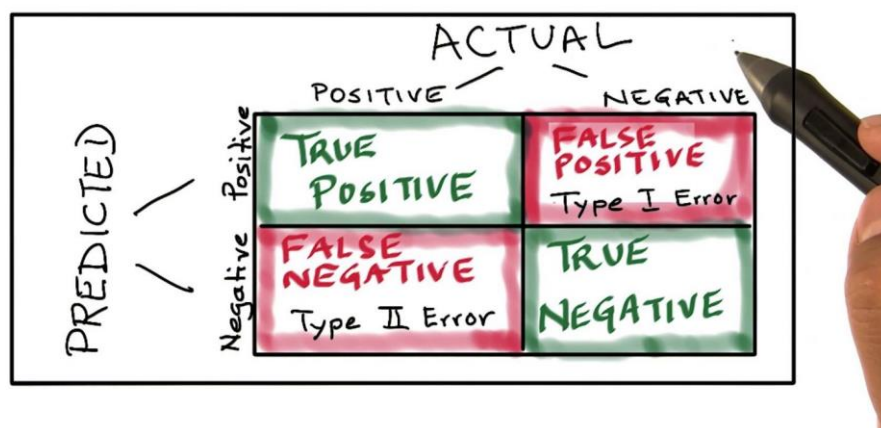
$$F1 = 2 \times (\text{precision} \times \text{recall}) / (\text{precision} + \text{recall})$$

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations

$$\text{Precision} = TP / TP + FP$$

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class – yes

The Confusion Matrix



		ACTUAL	
		POSITIVE	NEGATIVE
PREDICTED	Positive	TRUE POSITIVE	FALSE POSITIVE Type I Error
	Negative	FALSE NEGATIVE Type II Error	TRUE NEGATIVE

Source: [Udacity](https://www.udacity.com/)

$$\text{Recall} = TP / TP + FN$$

Another metric to be used is the **confusion matrix** which gives us the estimates of no of true positives,true negatives,false positives and false negatives.

Project Design

The requirements would be anaconda ipython notebook 2.7, numpy,pandas, scikit learn, matplotlib packages.

This project consist of many steps:

1. Loading the dataset

2. After reading, the csv file containing the dataset, its length would be evaluated, followed by the use of describe function of pandas, which would give the statistical metrics of the features of the dataset, it's mean , median, 1st and 3rd quartiles, maximum and minimum values, standard deviation
3. Then, there would be a check for the missing values in the features of the dataset, since there are a lot of zeroes, they need to be checked, how much data is bad, and is it affordable to remove all of them. If not, replace those features with their corresponding mean or medians
4. Plot the histograms of the features, and evaluate the distribution of individual features using matplotlib package. The features would be then plotted using a scatterplot in pairs to check for correlation between them.
5. The data would then be divided into features and labels to prepare it for applying machine learning algorithms. Once the data is divided into features and labels, the features would be scaled using a MinMaxScaler to get them on the same level, so that features with high magnitudes don't affect the classifier in making decisions.
6. After the above, preprocessing steps, the cleansed data needs to be split into training and testing datasets, to train the classifier and check it's accuracy. For this, we import cross validation from scikit learn, this splits the data into training and testing sets according to the parameters we give, i.e 80-20 split, 70-30 split, etc. From then on, we need to import the machine learning classifiers and evaluation metrics from scikit learn.
7. We will try a bunch of classifiers for this kind of binary classification problem, as the data is not very clean. We will, use Logistic regression, gradient boosting, svm, and extra trees classifier, random forest classifier, naive bayes, AdaBoost and evaluate their accuracies ,f1 scores and confusion matrices, to evaluate which classifier performs better in which scenario. As , this is a task which is measured by f1 scores, outright accuracy is not the only way the performance is measured, but the correct predictions should be evenly spread between the true positives and true negatives, to give a high f1 score.
8. A classifier might detect a lot of true negatives, and still have a poor f1 score and vice versa. After fitting these classifiers individually on the training set using the classifier.fit() function of scikit learn, these classifiers are individually made to predict the outcomes on the testing set, their results are printed using the evaluation metrics like f1_score, accuracy score and confusion matrix.
9. If the individual classifiers fail to come close to the f1 scores of the benchmark model, we can then perform a combination of supervised classification algorithms using stacking.

10. By importing `stacked_generalization`, we first perform a combination of base classifiers, and tune its parameters, fed it into a variable, perform cv folds on the training set, tune it for the best results. There is a blending model, which blends with the base models. The base models first train on the training set, its output is then fed to the blending model to train on it. Then the base models and blending models are stacked together and train on the data and reach a conclusion on the test set