Machine Learning Engineer Nanodegree

Diabetes Detection

Supervised Classification

Capstone Project

Nikhil Yadav

February 22nd,2017

1.Definition

Project overview

The chosen project is from the domain of bioinformatics or medical background.The aim of the project is to use supervised classification algorithms on the pima-indians-diabetes dataset.The dataset consists of data about females of age 21 and above of pima indian heritage who may or may not have diabetes.The algorithm helps in classifying the dataset , as to which individual has diabetes and which individual doesn't.IT's a binary classification problem.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.Diabetes mellitus is a chronic, lifelong condition that affects your body's ability to use the energy found in food.All types of diabetes mellitus have something in common. Normally, your body breaks down the sugars and carbohydrates you eat into a special sugar called glucose. Glucose fuels the cells in your body. But the cells need insulin, a hormone, in your bloodstream in order to take in the glucose and use it for energy. With diabetes mellitus, either your body doesn't make enough insulin it can't use the insulin it does produce, or a combination of both.Since the cells can't take in the glucose, it builds up in your blood. High levels of blood glucose can damage the tiny blood vessels in your kidney,heart,eyes, or nervous system. That's why diabetes-- especially if left untreated -- can eventually cause heart disease,stroke,kidney disease, blindness, and nerve damage to nerves in the feet.

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 (b) Donor of database: Vincent Sigillito (vgs@aplcen.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.The dataset though has been extracted from the kaggle

website's collection of datasets
https://www.kaggle.com/dssariya/pima-indians-diabetes-data-set

Problem Statement

The problem to be solved is the correct classification of females , on the basis of the presence or absence of diabetes mellitus by evaluating the given input features of the dataset.The dataset has 8 attributes, some of them contain missing values, which needs to be cleansed and preprocessed by replacing them with respective mean or median values, the data then needs to analyzed, the class variable to be separated from the features or attributes.The data then needs to be split into training and testing sets,and the features should then be scaled, to be made ready to be fit into a classification algorithm.The predictive capability of the algorithm will be checked on the testing set, if the accuracy matches the standards of being close to the benchmark model, the best algorithm will be chosen, else,the combination of algorithms will be stacked together to gain better results.As the input data is not very clean . The attempt will be to reach as close to the scores of the benchmark model of 0.8804 ,although the dataset used in the benchmark model was much more clean with respect to missing values, and the dataset used in kaggle has been modified to make it dirtier.So, the direct comparison would be slightly unfair as good data is always better than a good algorithm.

Evaluation Metrics

The f1_score is one of the evaluation metrics used in the benchmark model as well as the proposed project. f1_score=2*(precision*recall)/(precision+recall), where

precision= true positive/(true positive+false positive) or
true negative/(true negative + false negative)

recall= true positive/(false negative + true positive) or
true negative/(false positive + true negative)

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.

Example of a confusion matrix :-

Predicted	0.0	1.0
True		
0	363	61
1	80	210

It is one of the case studies where f1_score is preferred because getting true positives accurate is as important as getting true negatives accurate, because the classifier might be good in classifying true negatives right but bad in detecting true positives , hence it must be rewarded accordingly.

2.Analysis

Data Exploration

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)

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%). There are a few errors in the data. Although the database is labeled as having no missing values, someone liberally added zeros where there were missing values. Five patients had a glucose of 0, 11 more had a body mass index of 0, 35 others had a diastolic blood pressure of 0, 227 others had skinfold thickness readings of 0, and 374 others had serum insulin levels of 0. Studies that did not realize the previous zeros were in fact missing variables essentially used a rule of substituting zero for the missing variables. Ages range from 21 to 81.

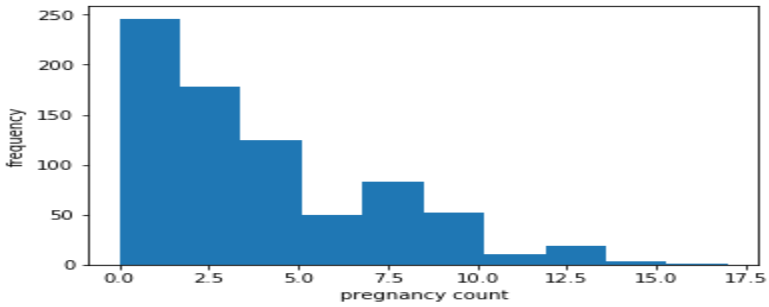
All of these features bear importance in predicting the class variable, intuitively, the feature 2,5 seem to have a strong relevance considering the domain knowledge.feature3 is important as diabetic patients usually have diastolic blood pressure 1-3 mm lower than normal. Older individual seem to have 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.

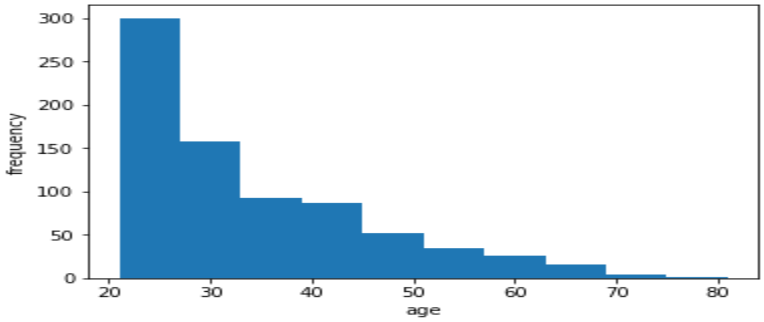
	preg_cnt	glu_conc	dia_bp	tris_skin	ser_insuli	bmi	dpf	age	diabetes
count	768	768	768	768	768	768	768	768	768
mean	3.845052	120.8945	69.10546	20.53645	79.7479	31.9925	0.471876	33.24088	0.348958
std	3.369578	31.97261	19.355	15.95221	15.24400	7.88416	0.331329	11.76023	0.476951
min	0	0	0	0	0	0	0.078	21	0
25%	1	99	62	0	0	27.3	0.24375	24	0
50%	3	117	72	23	30.5	32	0.3725	29	0
75%	6	140.25	80	32	127.25	36	0.62625	41	1
max	17	199	122	99	846	67.1	2.42	81	1

preg_cnt	glu_conc	dia_bp	tris_skin	tris_insuli n	bmi	dpf	age	diabetes
6	148	72	35	0	33.6	0.627	50	1

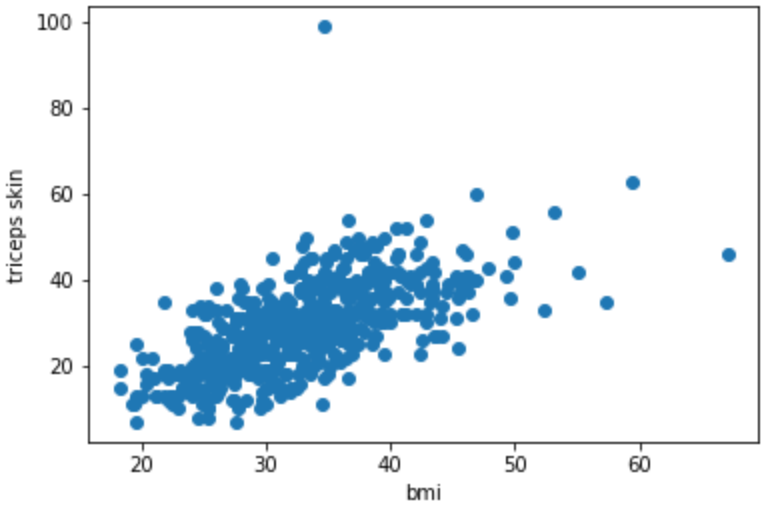
Sample of the dataset

Exploratory visualization

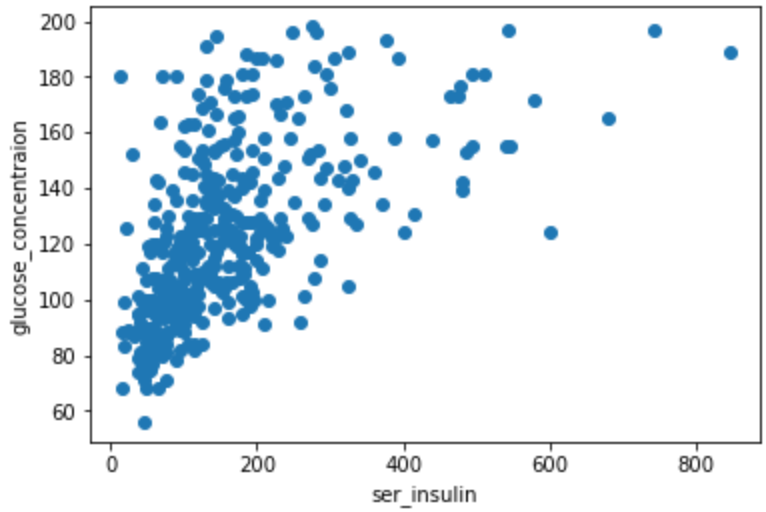




The following are the histograms of pregnancy count and age respectively, showing the distribution of the data. The pregnant count histogram shows that most of the females were pregnant in the range of 0-2 and most of the females in the study were aged around 20-25.



The following is a scatter plot of body mass index and triceps skin, showing some correlation, indicating that obese people have thicker triceps skin.



This is the single most important scatter plot , confirming the intuition about the dataset that individuals with higher glucose concentration also have a high serum insulin concentration. These are probably two of the best indicators of presence of diabetes.

Algorithms and Techniques

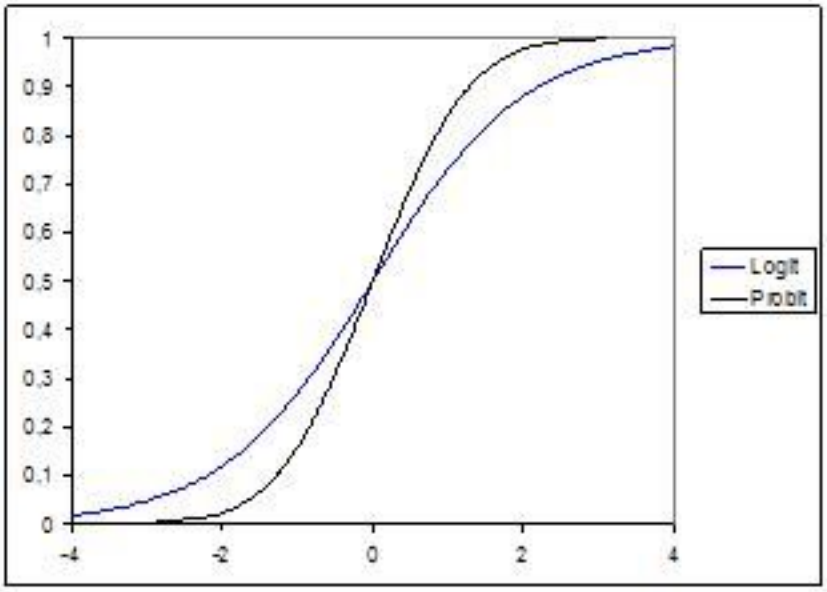
The algorithms used In this project are Implemented after the data is cross validated after splitting it into training and testing sets, and after setting a random state. The algorithms used are import from the scikit learn library of python. A plethora of supervised classification algorithms were trained and tested on the dataset, and the ones which showed the best accuracies were shortlisted and stacked to form a combination classifier.The algorithms tested were Decision tree classifier, random forest classifier, gradient boosting classifier,adaboost classifier ,Support vector classifier,Extra trees classifier, and logistic regression.The shortlisted algorithms after setting a random state to each of them and training them separately were extra trees classifier, decision tree classifier, gradient boosting classifier and logistic regression. The individual classifies were fine tuned and trained to extract the best results out of them, which wasn't enough, as it wasn't nearing the benchmark f1 score of 0.8804.So, we stacked them together using a stacked_generalization library which uses base and blending models to combine their effects and output a better result.

Decision Tree Classifier is a simple and widely used classification technique. It applies a straightforward idea to solve the classification problem. Decision Tree Classifier poses a series of carefully crafted questions about the attributes of the test record. Each time time it receive an answer, a follow-up question is asked until a conclusion about the class label of the record is reached.Once the decision tree has been constructed, classifying a test record is straightforward. Starting from the root node, we apply the test condition to the record and follow the appropriate branch based on the outcome of the test. It then lead us either to another internal node, for which a new test condition is applied, or to a leaf node. When we reach the leaf node, the class label associated with the leaf node is then assigned to the record, it traces the path in the decision tree to predict the class label of the test record, and the path terminates at a leaf node labeled NO.

Logistic regression,also known as maximum entropy, is really good when dealing with binary labels.Logistic regression assumes that the dependent variable is a stochastic event. For example, if we analyze a pesticides kill rate, the outcome event is either killed or alive. Since even the most resistant bug can only be either of these two states, logistic

regression thinks in likelihoods of the bug getting killed. If the likelihood of killing the bug is > 0.5 it is assumed dead, if it is < 0.5 it is assumed alive.

The outcome variable – which must be coded as 0 and 1 – is placed in the first box labeled Dependent, while all predictors are entered into the Covariates box (categorical variables should be appropriately dummy coded). Sometimes instead of a logit model for logistic regression a probit model is used. The following graph shows the difference for a logit and a probit model for different values (-4,4). Both models are commonly used in logistic regression, and in most cases, a model is fitted with both functions and the function with the better fit is chosen. However, probit assumes normal distribution of the probability of the event, when logit assumes the log distribution. Thus the difference between logit and probit is typically seen in small samples.



At the center of the logistic regression analysis is the task estimating the log odds of an event. Mathematically, logistic regression estimates a multiple linear regression function defined as:

logit(p) $= \log\left(\frac{p(y=1)}{1-(p=1)}\right) = \beta_0 + \beta_1 x_{i2} + \beta_2 \cdot x_{i2} + \_ + \beta_p \cdot x_{in}$

Gradient boosting is one of the boosting methods which consists of weak learners ,which learn from their past mistakes.it consists of three parts, a loss function, a weak learner to make predictions and an additive model to add weak learners to minimize the loss function.Basically, after every decision and step, the model learns from its mistake , getting rid of that type of mistake and falling down the gradient ,or making a descent to reduce the error function. After continuous gradient descents and adding to the additive sections what it learned, it reaches a local minima. That is the optimized state of the algorithm, any more tweaks, and it would ascend the gradient function of error or loss.The learning rate is one of the parameters that can be tweaked and The output of an individual tree is added to sequence of trees in an effort to improve the final output of the model.

Extra trees classifier is an extreme case of random forest classifiers which is an ensemble of decision trees which implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.In extremely randomized trees randomness goes one step further in the way splits are computed. As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias

Stacking of these algorithms is done using stacked _generaliation library, which consists of a base model and a blender model, the above classifiers and put in a variable called base model, over which the data trains,individually on each classifier and is fitted ,and the outputs of these layer(probabilities) is passed onto another layer or blender, then the decision is calculated by the final layer or the blender in this case.The goal of this method is to use the best quality from each classifier to increase the predictive performance of the model.

Benchmark

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 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.

[1]IJCEM International Journal of Computational Engineering & Management, Vol. 15 Issue 4, July 2012

3.Methodology

Data PreProcessing

The data consisted of zeroes at inappropriate places,it is impossible for some of these features to have a value of zero. While entering the data, the missing values were put as 0 in five of these features, namely, glu_conc, dia_bp,tris_skin,ser_insulin and bmi. The pandas data.describe() function was used to confirm the suspicion and carry out statistical analysis of the features. Some of these features had their first quartiles and min values as zero.

The zeroes in the above mentioned 5 features were replaced by nan, using pandas.replace() function.Individual counting of the no of missing values on each of these features was performed.

Missing values in glucose concentration:5
Missing Values in DiastoliC Blood Pressure:35
Missing values in Triceps skin thickness:227
Missing values in 2-hour serum insulin:374
Missing values in body mass index:11

The glucose concentration features missing values were replaced by its mean. For Diastolic blood pressure, the missing values were replaced by its median,the mean and median were same for this feature.Triceps skin thickness missing values were replaced by its median as the no of missing values is too high, the appropriate metric for this feature was the median, similar replacement was done in the case of serum insulin. The body mass index was replaced by its mean.After this, the data was separated into two separate variables, X and y, i.e features and labels. The features were scaled together using MInmaxscaler function from the preprocessing module of scikit learn.This data was Stored in X. X and y were later fed into the cross validation module to be split into training and testing states, the split was decided to be 70-30.

Implementation

The algorithms were imported and their random states and hyper parameters set, time was imported to evaluate the training and testing time of every algorithm. The evaluation metrics like confusion matrix, f1_score were imported as well to measure the performance of every algorithm.Extra trees classifier was the first algorithm to be fit and it's training and testing time ,f1 scores were recorded.The second algorithm that was fitted was Gradient boosting algorithm,It's scores were recorded as well. Logistic Regression was the third classifier to be fit, Followed by Decision tree classifier.Some of the other classifiers used were svc, gaussian naive bayes and random forest, their scores were not as good as the other mentioned classifiers. The chosen classifiers were decision trees, extra trees , gradient boosting and logistic regression. The stacked classifiers were Then fitted to the stacked generalization function and the no of folds for training was decided to be 9 after tuning with different fold numbers.There were complications in downloading and implementing the stacked generalization library, which was later resolved on seeing the documentation and sample code.

Refinement

The results of untuned extra trees classifier were having an fi_score of 0.51 and accuracy score of 0.7359, it was tuned by setting the n_estimators=100, min_samples_split=6,max_depth=8 and the results improved significantly to accuracy score of 0.7835 and f1 score=0.59.

The gradient boosting classifier was having an untuned performance of accuracy score=0.7705 and f1 score=0.624, the parameters were tuned by changing learning_rate=0.5, max_depth=6,min_samples_split=9 and the performance improved to accuracy score=0.796 and f1 score=0.6845.

The logistic regression and decision tree classifier were left untuned, as their performance remained similar.
The discarded algorithms consisted of :-

GaussianNB -f1 score=0.57,accuracy score=0.7748,

RandomForestClassifier - f1 score=0.57,accuracy score=0.7575

Support vector classifier -f1 score=0.55,accuracy score=0.76623

AdaBoost Classifier-f1 score=0.57, accuracy score=0.7748

The adaboost classifier was discarded as for stacking as its results were similar to logistic regression, two classifiers showing similar behavior is not preferred as their confusion matrices were similar as well.

4.Results

Model Evaluation and Validation

The final model is a stacking of base classifiers and blending classifiers,The base classifiers consists of
1.clf=ExtraTreesClassifier(n_estimators=100,min_samples_split=6,max_depth=8,random_state=6)

2.clf1=GradientBoostingClassifier(learning_rate=0.5,max_depth=6,min_samples_split=9,random_state=0)

3.clf2=LogisticRegression(random_state=6)

4.clf3=DecisionTreeClassifier(random_state=0)

And the blending classifier is the clf or the extra tree classifier

blending_model=ExtraTreesClassifier(n_estimators=100,min_samples_split=6,max_depth=8,random_state=6)

The no of cross validation folds is set to be 9.There is an attempt to generalize the results by changing the random states during cross validation ,which distribute the training and testing set with a different permutation in each state. The important metric for the model,that is the f1_score is evaluated for different random states and the best possible result is observed at random state=0(f1_score=0.81) and the worst result is noticed at random state=8(f1_score=0.73), which is better than the results obtained by any of the single model as none of them managed to cross the 0.7 f1 score barrier.The average f1_score of the final stacked model over 10 random states is 0.77.

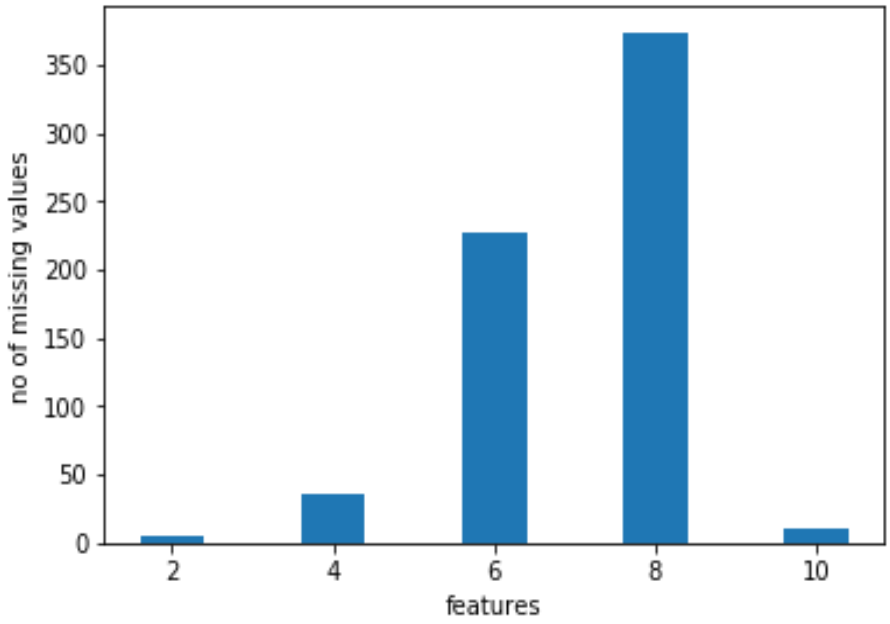
Justification

The average f1_score of 0.77,which is significantly less than the benchmark model score of 0.8804.But, we must take into consideration that the input data issued by kaggle had far more missing values than the benchmark model used, the important features had upto 374 missing values in this case, which makes it a decent enough model. The benchmark model had fewer missing values ,which it removed completely,hence a very small amount of data to train with and genralize.As good data is better than a good algorithm, and an accuracy score of 0.69 is considered acceptable, we consider this an acceptable model, but not a great model by any means.Th dataset used by the benchmark model had 140 missing values of ser_insulin compared to the kaggle issued dataset which had 374 missing values.Similarly, benchmark model used dataset containing 192 triceps skin missing values, compared to 227 issued by kaggle.Therefore, they could afford to delete those rows and still manage to train their classifier on a

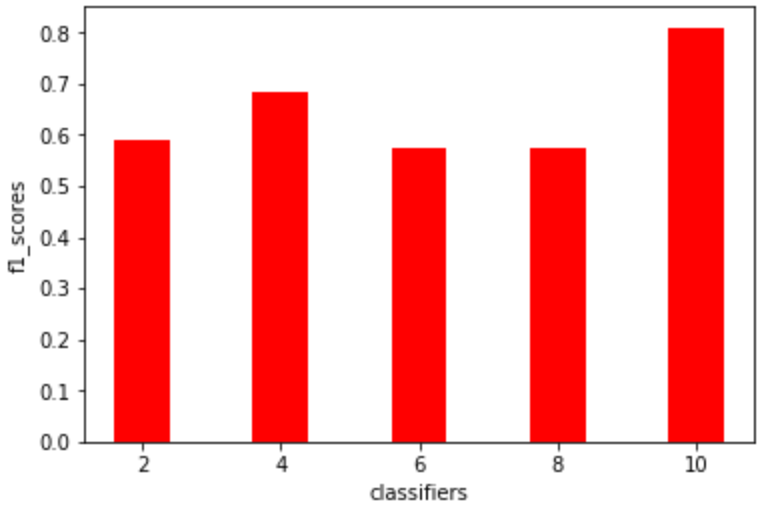
cleaner data whereas,This would result in loss of more than half of the dataset in our case, Hence it had to be preprocessed and the missing values replaced by mean and medians.Hence, the final model is weaker than the benchmark model.It is a decent result, when compared with the scores in kaggle,and considering that it is above 69% accuracy, it is still an acceptable model.So,the problem is considered to be solved, to an extent of the quality of the data.

5.Conclusion

Free-form Visualization



The following is the visualization of the no of missing features in the dataset in the features with missing values, the first bar is the glucose concentration, second being the diastolic blood pressure,third being triceps skin thickness, fourth being serum insulin, and fifth being body mass index. These missing values deteriorated the quality of the dataset,making it a challenging task to fit a classifier that performs the classification tasks thoroughly.



Above given bar chart is the comparison of f1 scores of all the chosen classifiers in the project, the first bar being clf=extratreesclassifier, the second bar being clf1=gradientboostingclassifier,the third bar being clf2=logistic regression,fourth bar being clf3=Decisiontreeclassifier, and the sixth bar being the stacked mode of all the above mentioned classifiers . one can see the performance enhancement in the final bar using random state of 0 as the input training and testing sets.

Reflection

The problem solution is summarized as follows , importing the necessary libraries, reading the dataset,looking at the statistics of the dataset, identifying the abnormality with missing values and dataset, replacing the zeroes with NaN, counting the NaN values, for each feature, the problem arises is what to do with these missing values .Discarding them would shorten the dataset considerably.Therefore, replacing them with their mean and median values accordingly. Separating the feature variables and class labels, then scaling the feature variables to be fitted to the classifier, separating the features and labels into training and testing states, setting a random state. Creating the classifier, fitting the data , noting down the f1 scores and confusion matrix.Trying different classifier.Importing the library stacked generalization and fitting the classifiers, trying different base and blender combinations, trying different no of folds.Checking the f1 scores over different random states to verify that the model has generalized well.

Dealing with the dirty data and preprocessing was a difficult task, also tuning the various classifiers and choosing the right classifiers after checking their scores and performance was a tedious task.Not having pre knowledge about stacking libraries, it was an interesting scenario, where the github link to a stacking library , not present in PyPI was discovered. Then, the stacked generalization library was downloaded and deployed on the dataset.

Considering, the quality of the dataset, the model has done a decent job to predict true positives and true negatives correctly, as it's f1 scores never drop below 0.73 in the 10 tested random states.

Improvement

There could be a better combination of algorithms or tuning, which i haven't come across, it's always a probability, a better cleansed data, would have yielded better results. Also, if the intuition was applied on the glu_conc and ser_insulin to cleanse the data, by shifting missing values to 3rd quartiles in case diabetes was present, and kept it median in absence of diabetes, maybe the classifiers would have responded well. Maybe , if i knew how to implement a neural net stacking , i would have tried differently. I have set the current solution as my benchmark as i was having trouble finding anything better than this.