Machine Learning Engineer Nanodegree

Capstone Project

Classifying Diabetes Disease

R Leela Samyuktha

February 7th, 2019

I.Definition

Project Overview

The selected project is from the area of bioinformatics or clinical background. The purpose of the challenge is to apply supervised classification algorithms at the pima-Indians-diabetes dataset. The dataset includes records about female of age 21 and above of pima Indian historical past who may additionally or may not have diabetes. The set of rules enables in classifying the dataset, as to which person has diabetes and which person doesn't. Diabetes mellitus (DM) or simply diabetes, is a group of metabolic diseases in which a person has high blood sugar. This high blood sugar produces the symptoms of frequent urination, increased thirst, and increased hunger. Untreated, diabetes can cause many complications. Acute complications include diabetic ketoacidosis and nonketotic hyperosmolar coma. Serious long-term complications include heart disease, kidney failure, and damage to the eyes. Considering in 1990, diabetes detection was a big issue. IT is estimated that by 2030 , there will be 366 million people approximately, affected by diabetes mellitus and by 2040, the world's diabetic patients will reach 642 million, which means that one of the ten adults in the future is suffering from diabetes. There is no doubt that this alarming figure needs great attention. With the rapid development of machine learning, machine learning has been applied to many aspects of medical health. In this study, we used decision tree, random forest, logistic regression etc. to predict diabetes mellitus. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.

The dataset that has been extracted from the kaggle datasets
https://www.kaggle.com/uciml/pima-indians-diabetes-database

The related academic work is found at
http://www.ijera.com/papers/Vol8_issue1/Part2/C0801020913.pdf

Problem Statement:

The main aim of my project is to classify the diabetic and non diabetic patients. For doing this I selected the dataset from Kaggle (https://www.kaggle.com/uciml/pima-indians-diabetes-database). So my goal is to the diabetic and non diabetic patients. Here I am using classification models to find the f-score of each model and select the best model with high f-score to predict the approvals. Here the input parameters are training data that we took and the output will be whether the patient has diabetes or not.

The tasks involved in it are:

1. Download the data

2. Cleaning the data and removing the null data points, duplicated data rows.

3. Visualizing the data.

4. Split the data and train and test which classifier performs good on the

dataset based on the evaluation metric we have chosen.

5. Choose the best classifier that gives the best accuracy scores.

Evaluation Metric:

I use f score as an evaluation metric for the classification of Diabetes disease. In the data set the class labels 0 and 1 are imbalanced. So I can use f score as the evaluation metric. Here I am calculating the f score of the selected models. I will select a model whose f score is greater than all the other models and we treat it as the best.

Formula for f score is:

f1_score=2*(precision*recall)/(precision+recall), where

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).
Precision can be thought of as a measure of a classifiers exactness. A low precision can also indicate a large number of False Positives.

Precision= true positive/(true positive+false positive) and

Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives. 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.

Recall= true positive/(false negative + true positive)

True Positives: are the values which are correctly predicted as positives

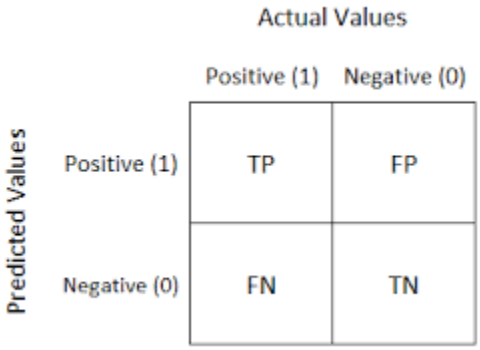
True Negatives: are the values which are correctly classified as negatives.

False Positives: are the values which are wrongly classified as positives. These are also type-1 errors.

False Negatives: are the values which are wrongly classified as negatives. These are also called as type-2 errors.

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.

The format of confusion matrix is as follows:



II.Analysis

Data Exploration:

The dataset that I am working is downloaded from
https://www.kaggle.com/uciml/pima-indians-diabetes-database

The dataset is Pima Indian Diabetes Database.csv. The number of instances are 768. It is a multivariate data set, contain 9 variables that are Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age and Outcome. All values are real integers. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The features are:

Pregnancies: Number of times pregnant
Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
BloodPressure: Diastolic blood pressure (mm Hg)
SkinThickness: Triceps skin fold thickness (mm)
Insulin: 2-Hour serum insulin (mu U/ml)
BMI: Body mass index (weight in kg/(height in m)^2)
DiabetesPedigreeFunction: Diabetes pedigree function
Age: Age (years)
The target variable is:
Outcome: Class variable (0 or 1) 268 of 768 are 1, the others are 0

First we can import the vital libraries and import our dataset to the Jupyter notebook. We can have a look at the referred to columns inside the dataset.

After importing dataset the first five instances of the dataset can be examined by using
dataset.head()

```
dataset.head()
```

	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

After that the information of the data is gained by using dataset.describe(). It is shown as follows:

```
In [11]: dataset.describe()
```

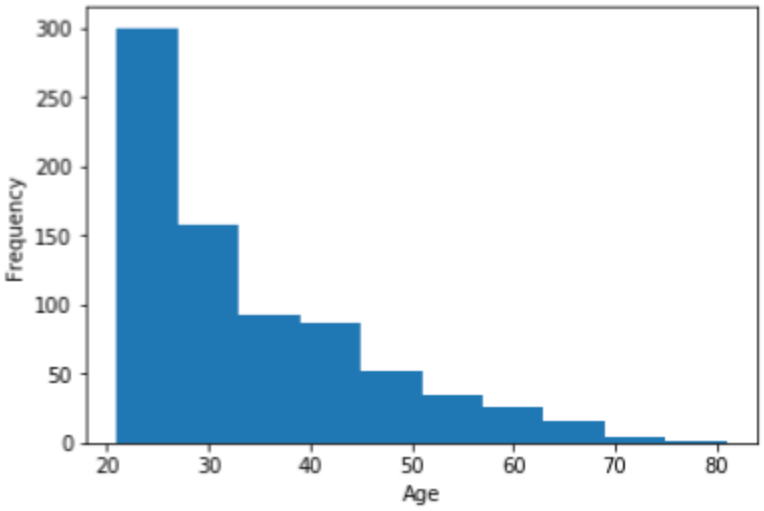
Out[11]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

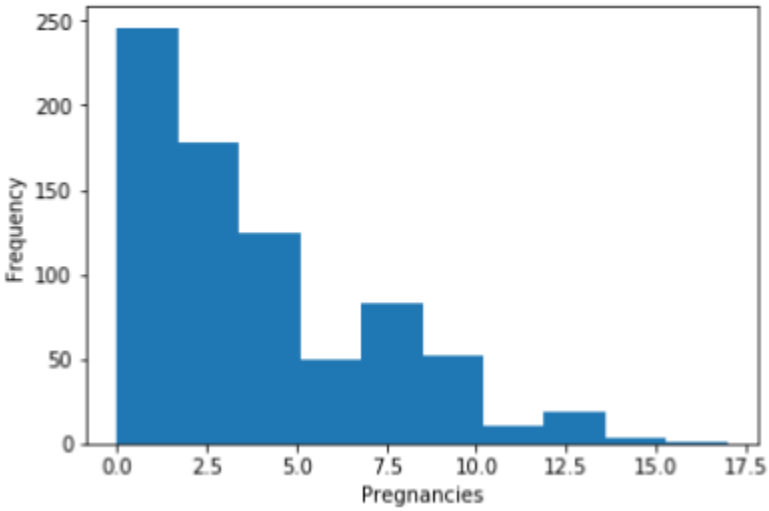
Then by using dataset.info(), it is known that all features don't have null values. This data set contains 500 non diabetic patient records and 268 diabetic patient records.

Explarotary Visualization:

The plots that visualized are:

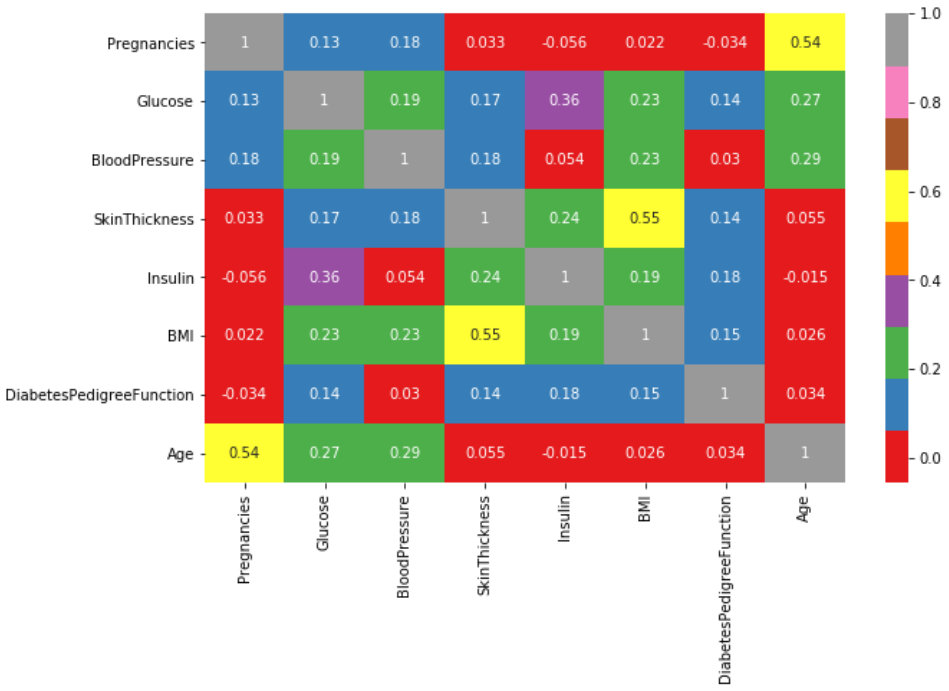


This histogram is used to understand the frequency of age of the females. So from this plot we can say that most of the females are aged around 20-25.



This pregnancy count histogram shows that most of the females have pregrenancy count ranged 0-2. Alike all features histograms are plotted to understand their frequencies.

The heatmap is a 2-D representation of data in which values are represented by colours. A simple heatmap provides the immediate visual summary of information. More elaborate heatmaps allow the user to understand complex data.



From the above heatmap we will represent the correlation of the numerical attributes in the data. The above heatmap clearly visualize the correlation between the numerical attributes. It shows how much the features are correlated to each other. So that Age and pregnancies, BMI and SkinThickness are correlated well followed by Glucose and Insulin.

Algorithms and Techniques:

Support vector machine (svm):

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

Advantages: SVM's are very good when we have no idea on the data. Works well with even unstructured and semi structured data like text, Images and trees. The kernel trick is real strength of SVM. With an appropriate kernel function, we can

solve any complex problem. Unlike in neural networks, SVM is not solved for local optima. It scales relatively well to high dimensional data. SVM models have generalization in practice; the risk of overfitting is less in SVM.

Disadvantages: Choosing a good kernel function is not easy. Long training time for large datasets. Difficult to understand and interpret the final model, variable weights and individual impact. Since the final model is not so easy to see, we can not do small calibrations to the model hence it's tough to incorporate our business logic.

Parameters: class sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated ', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, clas s_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', random_state=None)

Logistic Regression:

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

Advantages: Because of its efficient and straightforward nature, doesn't require high computation power, easy to implement, easily interpretable, used widely by data analyst and scientist. Also, it doesn't require scaling of features. Logistic regression provides a probability score for observations.

Disadvantages: Logistic regression is not able to handle a large number of categorical features/variables. It is vulnerable to over fitting. Also, can't solve the nonlinear problem with the logistic regression that is why it requires a transformation of non-linear features. Logistic regression will not perform well with independent variables that are not correlated to the target variable and are very similar or correlated to each other.

Parameters: Class sklearn.linear_model.LogisticRegression (penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1,

class_weight=None, random_state=None, solver='warn', max_iter=100, multi_class='warn', verbose=0, warm_start=False, n_jobs=None) The application is classification oriented . So, techniques that are used are taken from Classification techniques.

Naive bayes:

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other. The fundamental Naive Bayes assumption is that each feature makes an:

• independent

• equal contribution to the outcome.

Advantages:

• Very simple, easy to implement and fast.

• If the NB conditional independence assumption holds, then it will converge quicker than discriminative models like logistic regression.

• Even if the NB assumption doesn't hold, it works great in practice.

• Need less training data.

• Highly scalable. It scales linearly with the number of predictors and data points.

• Can be used for both binary and multiclass classification problems.

• Can make probabilistic predictions.

• Handles continuous and discrete data.

• Not sensitive to irrelevant features.

Disadvantages:

1 .The first disadvantage is that the Naive Bayes classifier makes a very strong assumption on the shape of your data distribution, i.e. any two features are independent given the output class. Due to this, the result can be potentially very bad - hence, a “naive” classifier. This is not as terrible as people generally think, because the NB classifier can be optimal even if the assumption is violated, and its results can be good even in the case of sub-optimality.

2. Another problem happens due to data scarcity. For any possible value of a feature, you need to estimate a likelihood value by a frequents approach. This can result in probabilities going towards 0 or 1, which in turn leads to numerical instabilities and worse results. In this case, you need to smooth in some way your probabilities, or to impose some prior on your data, however you may argue that the resulting classifier is not naive anymore.

3. A third problem arises for continuous features. It is common to use a binning procedure to make them discrete, but if you are not careful you can throw away a lot of information.

Parameters: alpha : float, optional (default=1.0), fit_prior : boolean, optional (default=True), class_prior : array-like, size (n_classes,), optional (default=None), X : {array-like, sparse matrix}, shape = [n_samples, n_features], y : array-like, shape = [n_samples], sample_weight : array-like, shape = [n_samples], (default=None), deep : boolean, optional, classes : arraylike, shape = [n_classes] (default=None)

KNearest Neighbours:

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

1. Ease to interpret output

2. Calculation time

3. Predictive Power

Advantages: The cost of the learning process is zero No assumptions about the characteristics of the concepts to learn have to be done Complex concepts can be learned by local approximation using simple procedures.

Disadvantages: The model can not be interpreted (there is no description of the learned concepts) It is computationally expensive to find the k nearest neighbours when the dataset is very large Performance depends on the number of dimensions that we have (curse of dimensionality) ⟹ Attribute Selection

Parameters: class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, weights='uniform' , algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=Non e, n_jobs=None, **kwargs)

Random Forest:

Tree models are known to be high variance, low bias models. In consequence, they are prone to overfit the training data. This is catchy if we recapitulate what a tree model does if we do not prune it or introduce early stopping criteria like a minimum number of instances per leaf node. Well, it tries to split the data along the features until the instances are pure regarding the value of the target feature, there are no data left, or there are no features left to spit the dataset on. If one of the above holds true, we grow a leaf node. The consequence is that the tree model is grown to the maximal depth and therewith tries to reshape the training data as precise as possible which can easily lead to over fitting. Another drawback of classical tree models like the (ID3 or CART) is that they are relatively unstable. This instability can lead to the situation that a small change in the composition of the dataset leads to a completely different tree model.

Advantages: 1.Reduction in over fitting: by averaging several trees, there is a significantly lower risk of over fitting.

2.Less variance: By using multiple trees, you reduce the chance of stumbling across a classifier that doesn't perform well because of the relationship between the train and test data.

Disadvantages : It takes more time to train samples.

Parameters: Class sklearn.ensemble.RandomForestClassifier (n_estimators='warn', criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None

Decision Trees:

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predict the value. Some advantages of decision trees are:

Simple to understand and to interpret. Trees can be visualized. Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values. The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree. Able to handle both numerical and categorical data. Other techniques are usually specialized in analyzing datasets that have only one type of variable. See algorithms for more information. Able to handle multi-output problems. Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by Boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret. Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model. Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

Decision-tree learners can create over-complex trees that do not generalize the data well. This is called over fitting. Mechanisms such as pruning (not currently supported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.

Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble. The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement. There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems. Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

Parameters: Class sklearn.tree.DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False

Benchmark model:

For this problem we will choose SVM classification model as the benchmark model.

By applying this Support Vector Machine we achieved an f-score of 0.81.

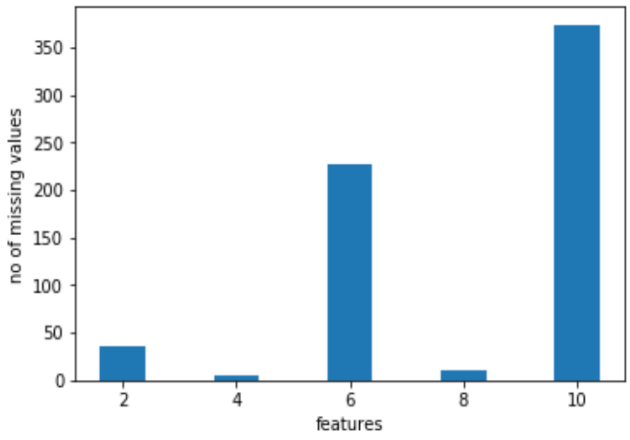
Now we will try and achieve better accuracy than this model by using the above mentioned classification models.

III.Methodology

Data Preprocessing:

There are some mistakes inside the data. Although the database is categorised as having no missing values, a person liberally added zeros in which there have been missing values. Five patients had a glucose of 0, 11 extra had a frame mass index

of 0, 35 others had a diastolic blood strain of 0, 227 others had skinfold thickness readings of 0, and 374 others had serum insulin stages of 0. Studies that did now not realize the previous zeros have been in truth missing variables basically used a rule of substituting zero for the missing variable. While entering the data, the missing values were put as 0 in five of these features, namely, Glucose, BloodPressure, SkinThickness, Insulin and BMI.



The above plot showing the number of missing values of those five features. No. of missing values are as follows:

BloodPressure : 35

Glucose : 5

SkinThickness : 227

BMI : 11

Insulin : 374

These missing values of the features are replace by their corresponding median and mean. The feature BloodPressure missing values are replace by its median value. The Glucose missing values are replaced by its mean. The SkinThickness missing values are replace by median value as the no of missing values is too high, the appropriate metric for this feature was the median. The BMI missing values are replaced by median value. The Insulin missing values are replaced by its

median value as the no of missing values is too high. 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. Then the data is split into training and testing data, the split was decided to 80-20.

Implementation:

From sklearn, the algorithms are imported. The evaluation metrics like confusion matrix, f1_score are imported as well to measure the performance of every algorithm. All algorithms are used to fit the model. The first algorithm is logistic regression, second is KNN, third is SVC, fourth is DecisionTree, fifth is RandomForest and last is NaiveBayes algorithm. These algorithms are used to train the model. Then evaluated the model by making predictions on the testing data as well as f-scores are recorded.

Logistic Regression: f-score is 0.66

Knn: f-score is 0.56

SVC: f-score is 0.65

Decision tree: f-score is 0.62

Random forest: f-score is 0.64

NaiveBayes: f-score is 0.64

Among the above reports logistic regression seems to be performing well.

Refinement:

I found out logistic regression as the best classifier out of the chosen classifiers. Now we will perform tuning of logistic regression in order to achieve the better fscore. We will assign C=10.0 and penalty=11.

Here we will use GridSearchCV() The fscore of tuned random forest is 0.71, which is better than the untuned logistic regression.

IV.Results

Model evaluation and Validation:

The final model we have chosen is tuned random forest which gave us more accuracy that is 0.71. In order to achieve this accuracy we assigned C=[10.0] and cpenalty='l1' but without using the parameters we achieved the fscore less than unturned logistic regression. Here we can say that the solution is reasonable because we are getting much less accuracy while using other models. The final model that is tuned logistic regression has been tested with various inputs to evaluate whether the model generalises well. This model is also robust enough for the given problem. We tested the logistic regression for various random states and we can clearly see that there is no big change in the fscore. We have tested for the random states 1, 200, 300 and we have achieved the fscore of random state =100 So from that we can say that Small changes in the training data will not affect the results greatly. So the results found from this model can be trusted.

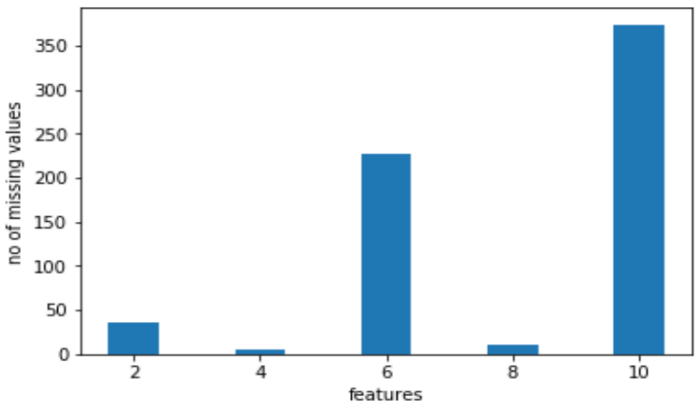
Justification:

My final model's solution is better than the benchmark model.

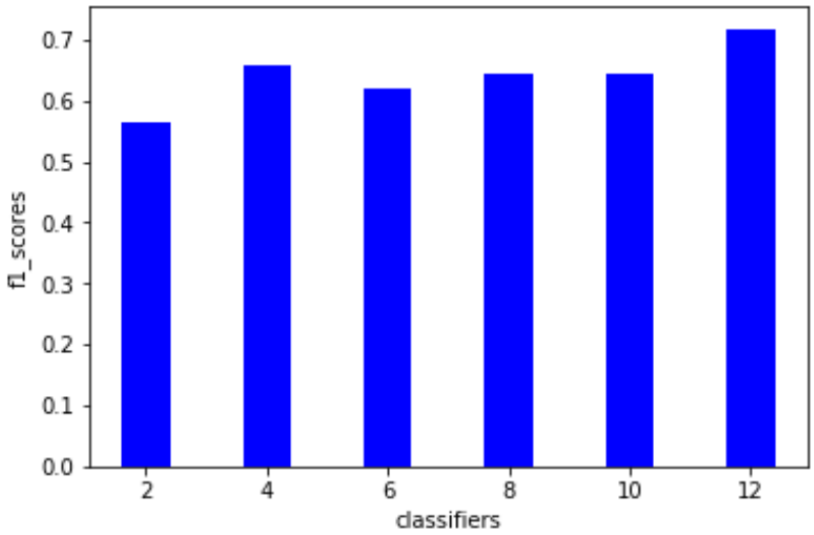
	Tuned logistic regression	Benchmark model
F-score	0.66	0.71

From the above we can conclude that the results for the final model are stronger than the benchmark model. Hence we can say that tuned logistic regression provides the significant to solve the problem of classifying diabetes disease.

V.Conclusion



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 blood pressure, second being the Glucose,third being skin thickness, fourth being BMI, and fifth being Insulin. 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 knn, the second bar being svm,the third bar being decision trees ,fourth bar being Random forest, and the sixth bar being the tuned logistic regression. One can see the performance enhancement in the final bar using random state of 100 as the input training and testing sets.

Reflection:

The problem answer is summarized as follows :

First importing the necessary libraries, reading the dataset,searching at the statistics of the dataset, figuring out the abnormality with missing values for each feature, the hassle arises is what to do with these missing values .Discarding them would shorten the dataset significantly.Therefore, changing them with their mean and median values. Separating the features variables and class labels, then scaling the feature variables to be fitted to the classifier, setting apart the features and labels into training and testing states, putting a random state. Creating the classifier, fitting the data , noting down the f1 ratings and confusion matrix by different classifiers. Checking the f1 scores over different random states to affirm that the model has generalized well.

It was a tough challenge to deal with dirty data and preprocessing, additionally tuning the various classifiers and deciding on the right classifiers after checking their scores and performance become a tedious mission.

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 0.71 in the 10 tested random states.

Improvement:

I used some classification algorithms on which I have good idea but 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 glucose and 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. May be, if I knew how to implement a neural net stacking , I would have tried differently.