## Overview

The adult (at least 21 years age) females in Pima Indian population bear Pheonix, Arizona have shown a high diabetes rate in the past. The National Institute of Diabetes and Digestive and Kidney Diseases has performed a continuous study of the community since 1965. During the study, each community resident was asked to undergo a standardized testing every two year; which included an oral glucose testing. The diagnosis of diabetes was performed based on the guidelines of world heath organization. The diagnosis guidelines were as follows; if the 2 hour post-load plasma glucose was at least 200 mg/dl (11.1 mmol/l) at any survey examination or if the Indian Health Service Hospital serving the community found a glucose concentration of at least 200 mg/dl during the course of routine medical care [1].

Dataset used

The dataset being used in the project is taken from Kaggle (https://www.kaggle.com/uciml/pima-indians-diabetes-database). [2]

There are 768 rows in the dataset. It has eight dependent variables representing various diagnostic measurements and an independent variable showing the diabetic condition of a person. The eight dependent columns are as follows [1]:

‘**Pregnancies’**: showing the number of times the female had been pregnant

‘**Glucose’**: Plasma Glucose Concentration at 2 Hours in an Oral Glucose Tolerance Test (GTT)

**BloodPressure**: Diastolic Blood Pressure (mm Hg)

**SkinThickness**: Triceps Skin Fold Thickness (mm)

**Insulin**: 2-Hour Serum Insulin (µh/ml)

**BMI**: Body Mass Index (Weight in kg / (Height in inches))

**DiabetesPedigreeFunction**: Diabetes Pedigree Function

**Age**: Age (year)

All these columns have values in a range of integers or float values whereas the final column showing the diabetes result is either a 1 or 0 showing diabetes or no diabetes

## The proposed solution, Evaluation metrics, and project design:

Since the objective here is to make a classifier that can classify the diabetes status of a female based on multiple variables, we can use one of many regressors available. The classification can be done using SVM, Decision tree classifiers, AdaBoost, Random forest, naïve Bayes, etc. In this project, I will use a few of these classifiers and compare the accuracy of these classifiers against each other.

**Short description of the above-mentioned classifiers:**

**Decision tree classifier**: Here the classifier learns from simple decision rules inferred from the features in data [3]. It is an easy to understand classifier and does not require data normalization. The number of decisions required to make a classification represents the depth of the tree and higher depth leads to complex decisions and more learning and fitting. But this excess can easily cause overfitting of data and give bad results on testing dataset.

**Random forest classifier**: The random forest classifier uses methodology of the decision tree classifier but at the same time it tries to reduce the overfitting. Random forest randomly selects a few features from the data and classifies the data based on those features. It makes multiple decision trees by selecting random features and makes multiple classifications based on these random trees [4, 5]. Finally, it outputs the value which occurs the most among all the generated trees.

**Gaussian Naïve Bayes classifier**: This classifier applies Bayes theorem to classify data by assuming independence among the features. Despite the naïve assumption of independence among features, this classifier works to a good degree but still the probabilities given by this classifier shouldn't be taken exact. [6]. In Gaussian NB, the likelihood of features is assumed as Gaussian in nature

**Multinomial naïve bayes**: In multinomial NB, the likelihood of data is given by multinomial distribution. [6]

**Adaboost classifier**: Adaboost classifier uses a base classifier to classify the data multiple times, each time focusing more on the mistakes of the last prediction and finally combining all the prediction to get the final prediction.

**SVM (Support vector machine)**: SVMs help in classification and regression. They not only classify the data but try to find the best estimator from the data. These are helpful in high dimensions spaces [7]. SVMs use different Kernel functions to classify data. In this paper we will be using two of these Kernels, namely Linear and RBF.

Linear Kernels fit the data by using a simple line, whereas the RBF kernel weights the every data point with an exponential function weighted by a hyperparameter $\gamma$ to fit the data. Both the kernels utilize a hyperparameter C, that defines the margin of fitting. The bigger margin between two sets is considered a better fit.

In this project, I will use the classifier that gives the best accuracy, precision, and recall (f_beta score) and will optimize its hyper-parameters to obtain the best model using grid search.

In this dataset, it is essential to use sensitivity (rate of true positives) and recall (the ability to detect true positives). Since we want to recognize patient with diabetes correctly and we do not want to miss someone with diabetes. The relative emphasis on precision and recall can be changed by changing the parameter beta in the f_beta score. But here we will just set it to 1 to get an f1 score, which places equal emphasis on recall and precision. In the end, I will to report a score on precision and sensitivity of our model, these values will be described in much more detail in the later sections of the report.

## Data processing:

| | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **0** | 0.352941 | 0.743719 | 0.590164 | 0.353535 | 0.000000 | 0.500745 | 0.234415 | 0.483333 |
| **1** | 0.058824 | 0.427136 | 0.540984 | 0.292929 | 0.000000 | 0.396423 | 0.116567 | 0.166667 |
| **2** | 0.470588 | 0.919598 | 0.524590 | 0.000000 | 0.000000 | 0.347243 | 0.253629 | 0.183333 |
| **3** | 0.058824 | 0.447236 | 0.540984 | 0.232323 | 0.111111 | 0.418778 | 0.038002 | 0.000000 |
| **4** | 0.000000 | 0.688442 | 0.327869 | 0.353535 | 0.198582 | 0.642325 | 0.943638 | 0.200000 |

Table 1: Data head, showing top five rows

| | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** |
|---|---|---|---|---|---|---|---|---|
| **count** | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 |
| **mean** | 0.226180 | 0.607510 | 0.566438 | 0.207439 | 0.094326 | 0.476790 | 0.168179 | 0.204015 |
| **std** | 0.198210 | 0.160666 | 0.158654 | 0.161134 | 0.136222 | 0.117499 | 0.141473 | 0.196004 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 0.058824 | 0.497487 | 0.508197 | 0.000000 | 0.000000 | 0.406855 | 0.070773 | 0.050000 |
| **50%** | 0.176471 | 0.587940 | 0.590164 | 0.232323 | 0.036052 | 0.476900 | 0.125747 | 0.133333 |
| **75%** | 0.352941 | 0.704774 | 0.655738 | 0.323232 | 0.150414 | 0.545455 | 0.234095 | 0.333333 |
| **max** | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |

Table 2: Description of data

Table 1 shows the head of the data with five rows and table 2 describes the statistics of the data used in the project. In addition to these descriptive statistics, the figures below showing the distribution of data help us describe the data.

It seems like most of the features are not too skewed. Though the features ‘Skin thickness’, ‘blood pressure’ and ‘insulin level’ appear a little asymmetrical, they both have high standard deviations and seem a little skewed from the plots below. But in general, the data for every other feature does not seem skewed. It might be a good idea to take into account this skewness.

Hence to remove this skewness, I took the logarithm of the data for analysis purpose and used this log data for classification.

Moreover, the data for ‘Skin thickness’ and ‘bloodpressure’ seems slightly imbalanced but since these features are not very important in determining the diabetes status, as seen in figure 2, I didn’t have to do much to remove their imbalance.

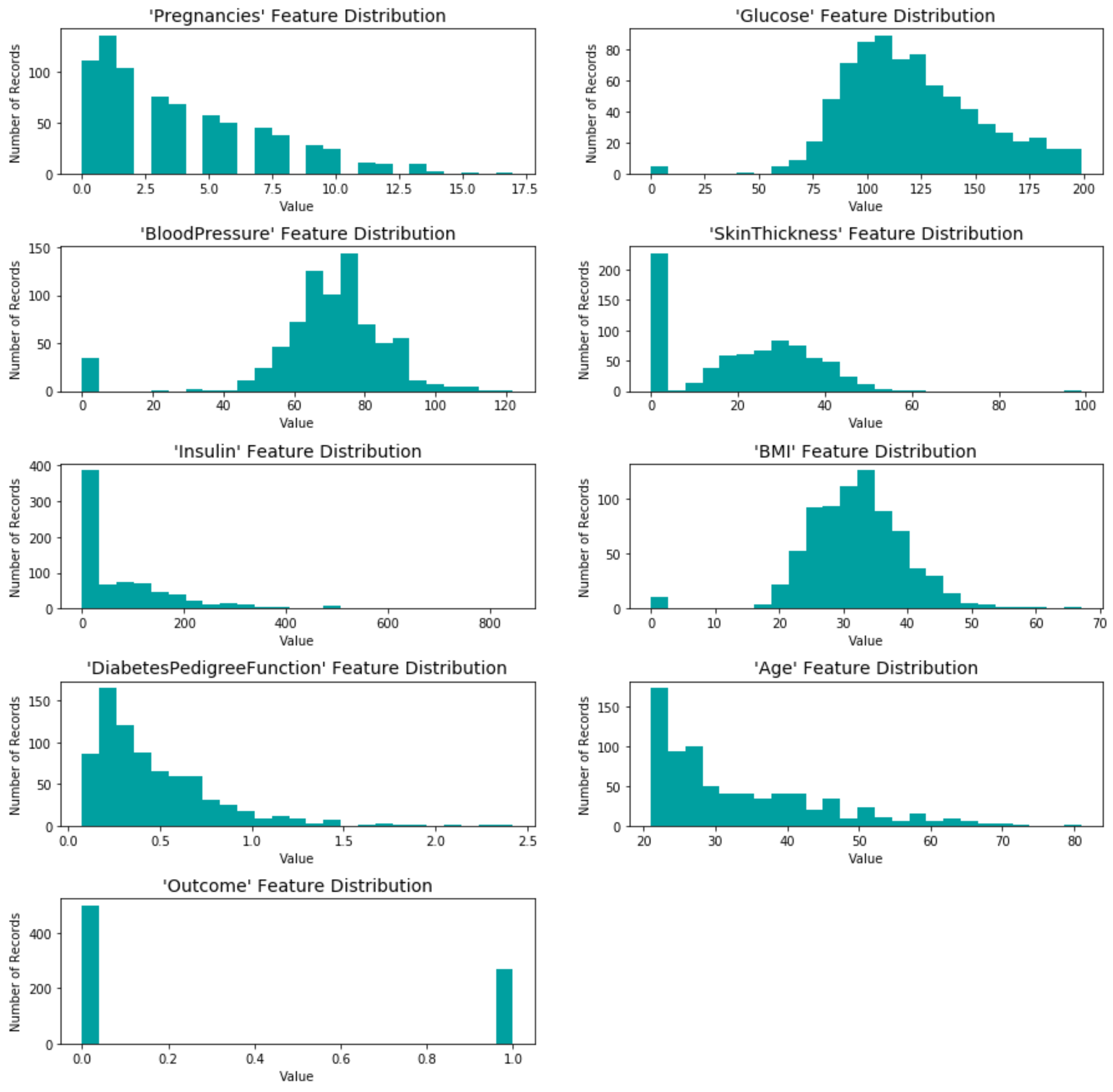

Figure 1. Distribution of data

After the processing of the data by taking the log, we splitted the data into the training and testing parts with a ratio of 75 to 25. The models were trained using the training set and their goodness was determined using the testing set. The goodness parameter is described in the next section.

## Classification of data:

### Preliminary classification using logistic regression (benchmark model)

First, we classified the log data using simple logistic regression; this gave us our benchmark model. The results for this benchmark model are as shown in table 1.

|  |  |
|---|---|
| Training accuracy | 75.4% |
| Testing accuracy | 78.6% |
| F1 score (training) | 57.2% |
| F1 score (testing) | 66.7% |

Table 2: Accuracy and testing score for logistic regressor

## Classification using various regression models:

We used eight different classifiers to classify our data. The regressors are as follows: decision tree classifier, Adaboost classifier, Multinomial Naïve Bayes, SVC (kernel=linear), SVC (kernel = rbf), random forest classifier and gaussian naïve Bayes. I used cross-validation technique with five iterations to obtain the accuracy and f1 score for each of these classifiers. The cross-validation helped to get an average score over multiple random sampling of training data. Table 2 shows the scores for various classifiers.

Though the cross-validation technique was easy to implement here, it could easily be a computationally costly technique if the dataset is large [8]. Hence it might be a better idea to implement other techniques to get initial estimates with a large dataset.

| Classifier | Accuracy | | F1 score | |
|---|---|---|---|---|
| | Training | Testing | Training | Testing |
| DecisionTree | 1.0000 (0.0000) | 0.7329 (0.0474) | 1.0000 (0.0000) | 0.6212 (0.0575) |
| Adaboost | 0.9959 (0.0029) | 0.7426 (0.0258) | 0.9941 (0.0042) | 0.6271 (0.0346) |
| MultinomialNB | 0.6531 (0.0006) | 0.6531 (0.0024) | 0.0000 (0.0000) | 0.0000 (0.0000) |
| SVC (linear) | 0.7683 (0.0078) | 0.7590 (0.0314) | 0.6132 (0.0122) | 0.5973 (0.0532) |
| SVC (rbf) | 0.7537 (0.0076) | 0.7509 (0.0255) | 0.5450 (0.0229) | 0.5372 (0.0488) |
| Random forest | 0.9849 (0.0052) | 0.7556 (0.0246) | 0.9779 (0.0078) | 0.6101 (0.0437) |
| GaussianNB | 0.7516 (0.0109) | 0.7508 (0.0249) | 0.6220 (0.0154) | 0.6164 (0.0348) |

Table 2: Accuracy and F1 score for various classifiers on training and testing datasets.

It seems like pretty much every classifier except 'multinomialNB' classifier gives a decent accuracy and f1 score for testing data set. Hence, I am going to choose one of the above classifiers and fine tune the model using grid search. In my analysis, I chose the SVC classifier with a linear kernel.

The tuning of the SVC with linear Kernel was performed by optimizing the parameter 'C'. This parameter describes the margin between the data on the two sides of the classifying line. The goal here is to maximize this margin from the data on each side. The C values that were used for training are [0.4, 0.5, 0.7, 0.8, 0.9, 1]. The optimized C value was 0.9.

Table 3 shows the results optimized model and compares it against the unoptimized model:

| Unoptimized mode | | Optimized mode | |
|---|---|---|---|
| Accuracy | F1 score | Accuracy | F1 score |
| 0.7597 | 0.6476 | 0.7662 | 0.6538 |

Table 3: Comparison of scores of optimized SVC model with a linear kernel against an unoptimized model

It seems like there is a slight improvement but certainly not very noticeable after the optimization with the grid search.

Moreover, our testing scores are pretty much the same as our training scores for most of the models, hence we can say that these models neither underfit nor overfit the data and hence the classification is robust using the above methods.

In the end, we can also see that the optimized SVM model with linear kernel performs pretty much the same as the simple logistic model, which we considered as our benchmark model. This mostly shows the simple distribution of data.

From here we can go on to calculate the sensitivity and the precision of our model. The sensitivity as mentioned in the earlier section shows the rate of true positives over the actual number of positives present in the data, whereas the precision represents the proportion of true positives among the total positives classified by our model. I used the confusion matrix for calculating these scores. The confusion matrix is a 2X2 matrix where each element ($\mathrm{C_{mn}}$) represents a physical quantity. The various elements of the confusion matrix can be described as: $\mathrm{C_{00}}$: true negatives, $\mathrm{C_{01}}$: false negatives, $\mathrm{C_{10}}$: false positives, $\mathrm{C_{11}}$: true positives.

The precision and the sensitivity can be described in terms of the confusion matrix terms as follows:

$$\mathrm{precision} = \frac{\mathrm{C_{11}}}{\mathrm{C_{11}} + \mathrm{C_{10}}} = 0.6939$$

$$\mathrm{sensitivity} = \frac{\mathrm{C_{11}}}{\mathrm{C_{11}} + \mathrm{C_{10}}} = 0.6182$$

In the end, I also tried to see which of the features are most indicative of the diabetes status. ‘Glucose’, ‘BMI’, and ‘blood pressure’ seem to be the most important features in determining the diabetes status. Figure 2 shows the weightage of the top features indicative of diabetes.

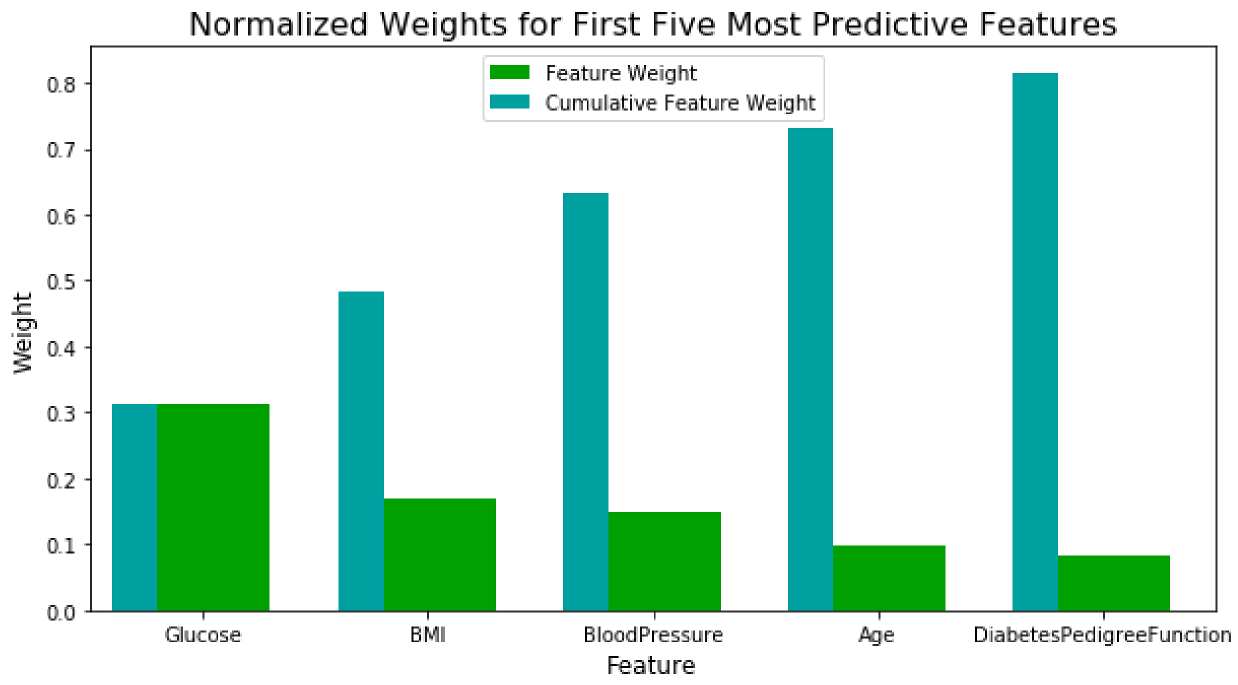

Figure 2: weightage of five most important features in classification of diabetes

## Conclusion:

We used various classifiers for classifying the diabetes status of the patients. The classifiers we used were, SVM, Decision tree classifiers, AdaBoost, Random forest, naïve Bayes. Though, most of our models performed very well on data, we fine-tuned the SVM (linear kernel) to get better results. There was slight improvement when we optimized the SVM with grid search.

We also compared all our models with our benchmark logistic regression, but we did not observe much improvement over our benchmark model. We discussed this by saying that most probably the distribution of data is simple or linear.

The use of machine learning techniques for medical diagnosis seems an interesting area, I think these techniques can be easily extended to other diseases if we can come up with proper data. But many times data may not be available in very usable form and this could always pose a great challenge in using machine learning techniques.

Although, at this point already our scores are much better than the random guessing of diabetes status. We can still use some more sophisticated techniques of 'unsupervised' learning, where we can try to find if there are any clusters in the data which are hard to consider using the supervised learning techniques used in our analysis. Some of the techniques in unsupervised learning include K-clustering, Hierarchal clustering, Gaussian mixture models, etc.

## References:

[1]. Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *In Proceedings of the Symposium on Computer Applications and Medical Care* (pp. 261--265). IEEE Computer Society Press.

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

[3]. http://scikit-learn.org/stable/modules/tree.html

[4]. http://scikit-learn.org/stable/modules/ensemble.html

[5]. Udacity lectures

[6]. http://scikit-learn.org/stable/modules/naive_bayes.html

[7]. http://scikit-learn.org/stable/modules/svm.html

[8]. http://scikit-learn.org/stable/modules/cross_validation.html