# Machine Learning Capstone Project

# 'Pima Female Indians' Diabetes Prediction Algorithm'

Siddhant Tandon

August 14th, 2019

## DEFINITION

### Domain Background:

Today we have the privilege of enjoying many benefits from the comfort of our homes, which has been changing our lifestyle preferences as we grow old. Although these lifestyle changes have improved quality and experience yet has also given birth to diseases which are becoming widespread. Diabetes mellitus is a metabloic diseases that is affecting people across the world due to their lifestyle. USA being one of the developed economies has also started seeing a drastic rise in diabetic population over the last decade. This is quite alarming and thus detection of diabetes in its early stage has become a necessity to avoid severe health problems in both men and women.

The main cause for diabetes as discussed by NIH is when blood glucose level is too high and insulin, a hormone made by the pancreas isn't enough to turn sugar into energy for the cells. Glucose then stays in blood overtime and doesn't reach the cells. This is the reason why many people with diabetes start having health problems due to corruption of their organs by sugar levels.

My family has had diabetic history in recent generations from both my parents' sides. This makes me more susceptible to have diabetes later in life. Thus, my motivation to do this study is to help advance the early detection of diabetes in humans to prevent serious health complications.

This report will cover the data on the Pima Indian population near Phoenix, Arizona. Related research work paper is:

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2245318/pdf/procascamc00018-0276.pdf

In practice, it has become important to use the help of machine learning algorithms to help detect diabetes (in females for this project) so that proper steps can be taken in order to avoid the onset of health problems to those affected by this disease.

Further information on diabetes and introductory literature to machine learning can be found here:

NIDDK (National Institute of Diabetes and Digestive and Kidney Diseases)

Machine Learning

### Datasets and Inputs:

The dataset is obtained from Kaggle.com under the tab 'datasets' for public use. It was posted from the UCI Machine Learning repository. The dataset link is below.

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

The dataset has 768 instances and 9 columns. There are 8 medical predictor variables (independent) and one target (dependable) variable labelled 'outcome'. The independent variables are listed below:

- Pregnancies (Number of times pregnant)
- Glucose (Plasma glucose concentration: a 2 hours in an oral glucose tolerance test)
- Blood Pressure (Diastolic blood pressure (mm Hg))
- Skin Thickness (Triceps skin fold thickness (mm))
- Insulin (2-hour serum insulin (mm U/ml))
- BMI (Body mass index (weight in kg/ height in m^2))
- Diabetes Pedigree function
- Age (years)

The dataset was recorded by National Institute of Diabetes and Digestive and Kidney Diseases.

## Problem Statement:

Build a ML model to accurately predict whether the patients (females of age 21 and older of Pima Indian heritage) in the dataset have diabetes or not.

This is a binary classification problem which can be solved using Supervised learning algorithms in conjunction with ensemble methods. LightGBM and XGBoost are two boosting methods which I expect to show higher accuracy than individual supervised learning algorithms. An introductory literature can be found here.

## Evaluation Metrics:

A good metric based on the above discussion of binary problems is the F1 score. It measures the test's accuracy. The formula for F1 score is:

F1 = 2* (precision*recall)/ (precision + recall)

It can be noted that the F1 score is a weighted average of precision and recall, where a F1 score reaches its best value at 1 and worst at 0.

# ANALYSIS

## Setup and platform:

After downloading the dataset from Kaggle, an instance of jupyter notebook was used to load the dataset and perform analysis using Python language.

## Data Visualization and Exploration:

The diabetes.csv file has 768 instances and 9 columns of which 8 are features as described in section Datasets and Inputs. The last column is the outcome column which has value of 1 or 0, corresponding to positive detection of diabetes and negative detection of diabetes in a female. Fig 1 is a snippet of the python output summarizing the dataset bounds.

| | 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 |

```
Total number of patients: 768
Number of features: 8
Number of patients with diabetes: 268
Number of patients without diabetes: 500
Rate of diabetes in dataset: 34.895833333333336%
```

Figure 1. Snippet of code summarizing 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 |

Figure 2. Snippet of dataset sample

It can be noticed from the preliminary descriptive statistics of the dataset that many features have min value of 0. This should appear as a red flag considering 0 value for most features wouldn't make sense e.g., Glucose, BloodPressure etc. A data preprocessing technique will need to be utilized to correct the discrepancies.

Figure 3. shows the count of the outcome variable to help see the classification between diabetic and non-diabetic females.

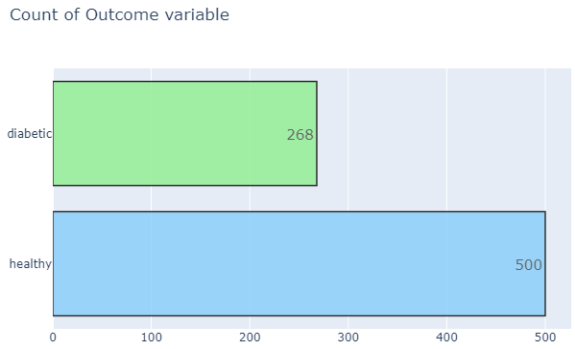



Figure 3. Outcome

As can be observed from the bar chart, the number of healthy persons is 500 and diabetic are 268. But this doesn't yet tell us what the features look like. Fig 4 is a histogram plot of all the attributes of the dataset.

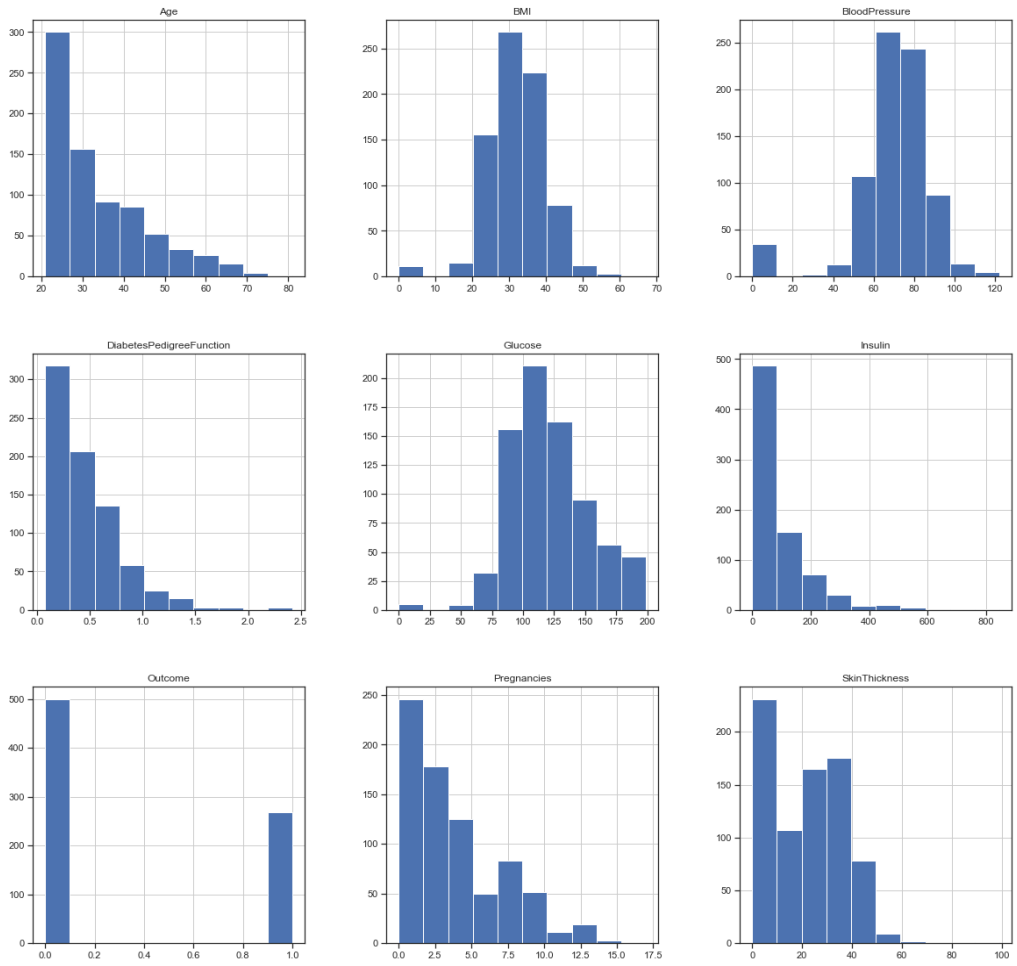



Figure 4. Histogram plots of attributes of the dataset

From the above plots we cross verify the descriptive statistics shown in Fig 1. These discrepancies could either be due to outliers or NaN values in the dataset. To better understand the attributes, we need to check the correlation between them.

In Fig 5, we can see that there is no strong correlation between the attributes. I see a small correlation between Pregnancies, Glucose and Age to being diabetic. This is good in a way as that means we can find out the important features in the dataset and perhaps continue to use them for our algorithm. We will later analyze the data statistically to find out the correlation values.


Figure 5. Correlation plot

## Algorithms and Techniques:

As mentioned previously this is a binary classification problem and its solution can be from our lessons under Supervised learning. The most common solution to approach such prediction problems, is to test and compare the following options:

- Logistic regression
  The logistic function or sigmoid function can take any real valued number and map it into a value between 0 and 1, but never at those limits. This is at the core of logistic regression which uses an equation as representation similar to linear regression. The coefficients of the algorithm are estimated on the training set and the best ones produce a higher prediction.

- Decision Trees
  They are a type of Supervised Learning algorithms where the data is continuously split according to certain parameter. The tree has decision nodes where it splits and leaves which give final outcomes. A good parameter choice well results in an accurate model.

- Naive Bayes
  One of the easiest way of selecting the most probable hypothesis in a classification problem, given the data, is by having prior knowledge about the problem. Bayes' Theorem provides a way that we can calculate the probability of a hypothesis given our knowledge. After calculating the posterior probability for a number of different hypotheses, you can select the one with highest probability. It works well with classification problems since, the attributes are assumed to be conditionally independent of the target value, this voids the need for calculating values for each attribute. Yet, the algorithm performs well on data on which this assumption does not hold true.

- Random Forest vs XGBoost vs LightGBM
  Ensemble methods combine the decisions from multiple models to improve the overall performance. There are different types of Ensemble models, while Random Forest is a bagging algorithm, XGBoost and LightGBM are boosting algorithms. Bagging or bootstrap aggregating is a sampling technique in which we create subsets of observations from the original dataset with replacement. This gives a fair idea of distribution of the whole set. Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model. The succeeding models are dependent on the previous model.
  Random Forest, a bagging technique uses Decision Trees as its base estimators, it randomly selects a set of features which are to decide best split at each node of the decision tree. Thus, it randomly selects data points and features and builds multiple trees.

  XGBoost is an advanced version of gradient boosting, that works on both regression and classification problems. It combines several weak learners to form a strong learner. It has scalability in all scenarios and runs more than 10 times faster than most algorithms.
  LightGBM is very useful for large datasets and it outperforms all the other algorithms for that matter. It uses gradient boosting framework and follows a leaf-wise approach, while many others use level approach. It runs the risk of overfitting in smaller datasets.
  Here is an article to read more on differences between LightGBM and XGBoost.

- Linear Support Vector Clustering
  The objective of a Linear SVC is to fit to the data you provide, returning a best fit hyperplane that categorizes your data. Then after getting the hyperplane, you can then feed some features to classifier to see the predicted class.

- K-Nearest Neighbors
  It is a non-parametric method used for classification and regression. It is a type of instance-based learning where the function is only approximated locally and all computation is deferred until classification. The neighbors are taken from a set of objects for which the class is known. The algorithm is sensitive to the local structure of the data. Classification is computed from a majority vote on the nearest neighbors of each point. The optimal choice of k-neighbors is data dependent.

The idea is to implement ensemble methods in the end to optimize the output from above to produce the best prediction algorithm.

### Benchmark Model:

According to the dataset given, there are 268 females with diabetes and 500 without diabetes. Thus, the rate of diabetes in this sample set is close to 34.9%.

The benchmark model chosen is from the larger class in the dataset, which is the negative result (or samples with no diabetes). That gives us the model benchmark of 65.1%.

Aim is to create a model which can detect whether a person has diabetes or not with greater accuracy than 65.1%.

## METHODOLOGY

### Data Preprocessing:

The discussion above highlights the importance of correcting the data. We will be looking at the null data points and correcting them. Fig 6 shows the bar chart of the missing values for different features of the dataset.

The following is concluded from the chart:

| Features | Missing count |
|---|---|
| Insulin | 374 |
| SkinThickness | 227 |
| BloodPressure | 35 |
| BMI | 11 |
| Glucose | 5 |

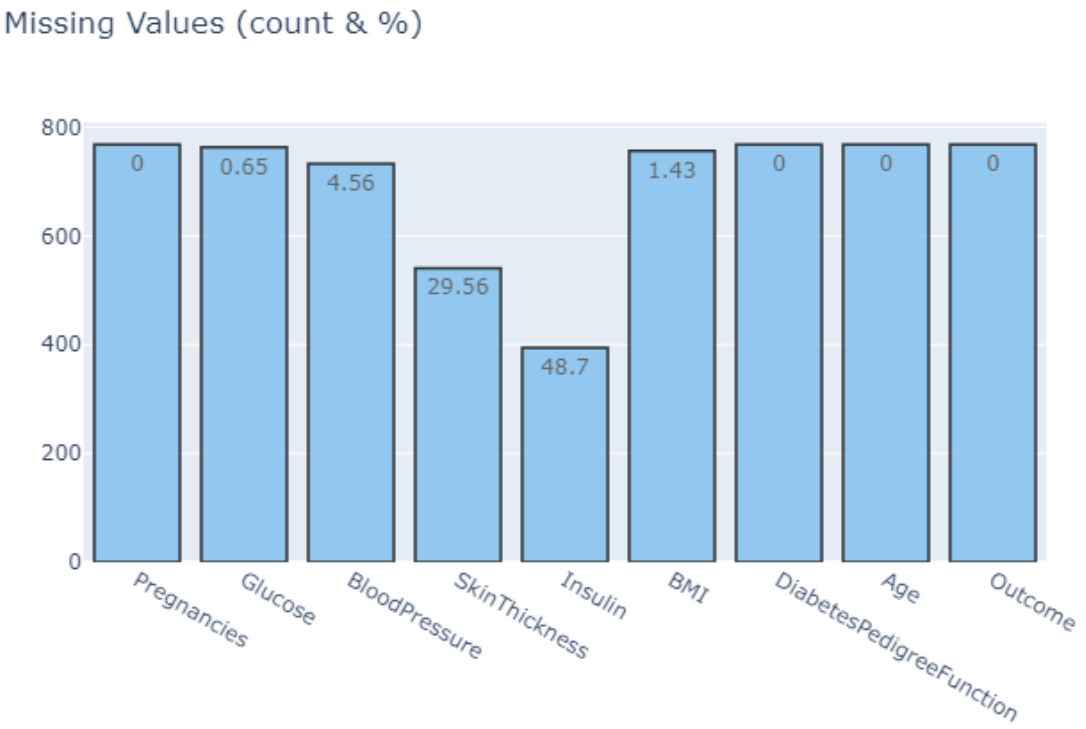



Figure 6. Missing count chart

The dataset's features are important indicators of one's health. Hence, to change the null values to zero will make a heavy biased case and the algorithm designed afterwards will be poor in making predictions. To solve this, we begin with taking median values for all features with the two possible outcomes. Then we take each feature and replace its missing values with the median values of the person's corresponding 'Outcome' value. The code to generate median is below:

```
def median_feature(var):
    place = data[data[var].notnull()]
    place = place[[var, 'Outcome']].groupby(['Outcome'])[[var]].median().reset_index()
    return place
```

The following table lists the median values for all features

Table 1. Median values

| Outcome | Insulin | Glucose | SkinThickness | BP | BMI |
|---|---|---|---|---|---|
| 0 | 102.5 | 107 | 27 | 70 | 30.1 |
| 1 | 169.5 | 140 | 32 | 74.5 | 34.3 |

Post the availability of the above analysis I decided to replace the missing values from the features. An example is given below:

*data.loc[(data['Outcome'] == 0 ) & (data['Insulin'].isnull()), 'Insulin'] = 102.5*

*data.loc[(data['Outcome'] == 1 ) & (data['Insulin'].isnull()), 'Insulin'] = 169.5*

Then finally we can cross verify by generating the chart again to see if we have been successful.

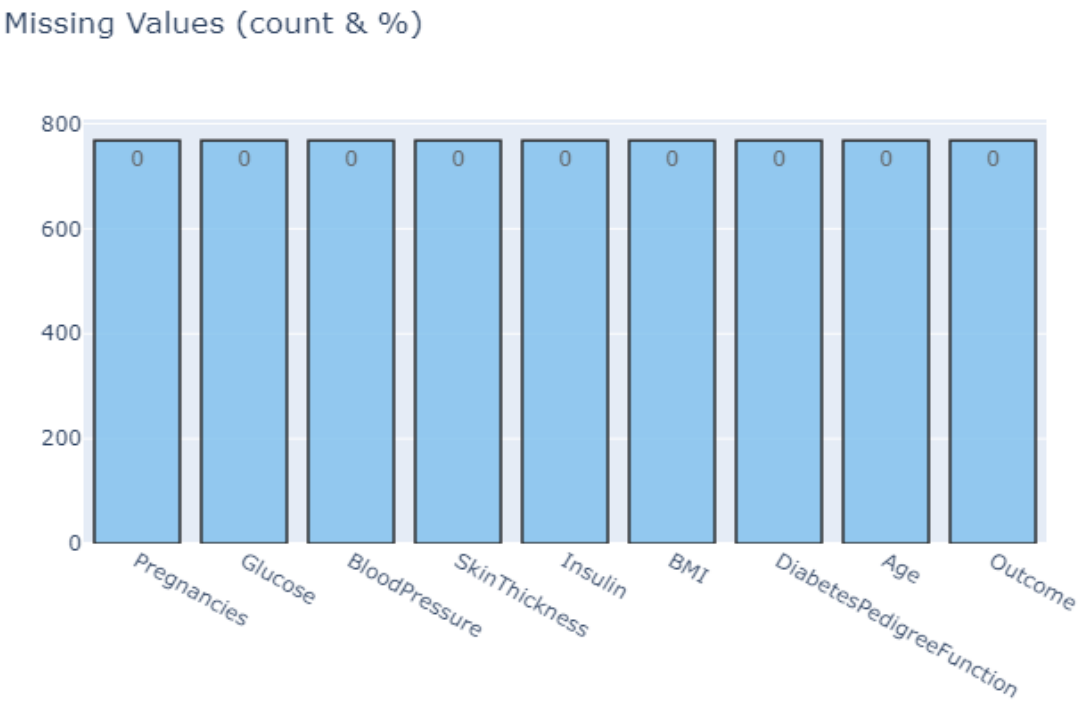



Figure 7. Missing count chart

In Fig 7 we can see that the missing values have been replaced successfully.

The method to use median values stems from the understanding that the person who is not diabetic with a missing value in a certain feature should be assumed to host the same values for that feature as the median of the dataset for non-diabetic people. And similarly, for person who is diabetic should be assumed to host values for a missing feature value as the median of the dataset of people who are diabetic.

## **Implementation:**

To create a robust algorithm, I decided to implement the following order to design my algorithm.

1. Model Comparison – Preliminary score analysis of different classifiers Linear SVC, Decision Tree, Gaussian Naives Bayes, Logistic regression and K-Nearest Neighbors.

2. Feature Selection – to check which features are important and abridge the dataset and compare to previous scores
3. Cross Validation and GridsearchCV scores
4. Ensembling techniques to final tune

## Model Comparison

a. Learning models were imported from sklearn
b. Models were initialized
c. data is split into training and testing sets with test size of 20%
d. Models were trained on the training sets
e. Prediction was done on testing set and scoring was evaluated using F1 score

There was no complication in execution of the above steps.

## Feature Selection

a. Correlation plot was made using seaborn
b. Features with highest values were selected
c. Three features Insulin, Glucose and SkinThickness were separated to a different dataframe
d. Model comparison on the new dataframe was executed

It was strange to see the high feature importance for SkinThickness as compared to Diabetes Pedigree Function or BMI.

## Cross Validation and GridSearchCV

a. Observing small accuracy change due to feature selection, the original dataset was used again instead
b. Kfold parameters (n_splits = 10, random_state = 42)
c. All models under 'Model Comparison' were used
d. New scores were reported
e. GridSearchCV was used to find the best parameters and scoring was done on the same

There was no complication in executing the steps above.

## Ensembling

a. Under basic ensembling techniques, voting method was used amongst all models listed in 'Model comparison'
b. After execution the prediction was scored
c. LightGBM was used with KNN as a boosting technique
d. After testing various parameters and utilizing grid search, the best parameter for n-neighbors and best score (same run) was printed

e. Next, XGBoost was used on the dataset
f. The model was trained and tested
g. Accuracy was scored
h. Lastly, Random Forest Classifier was also trained and tested
i. Accuracy was scored

There was no complication in tuning of the models and execution of the steps above.

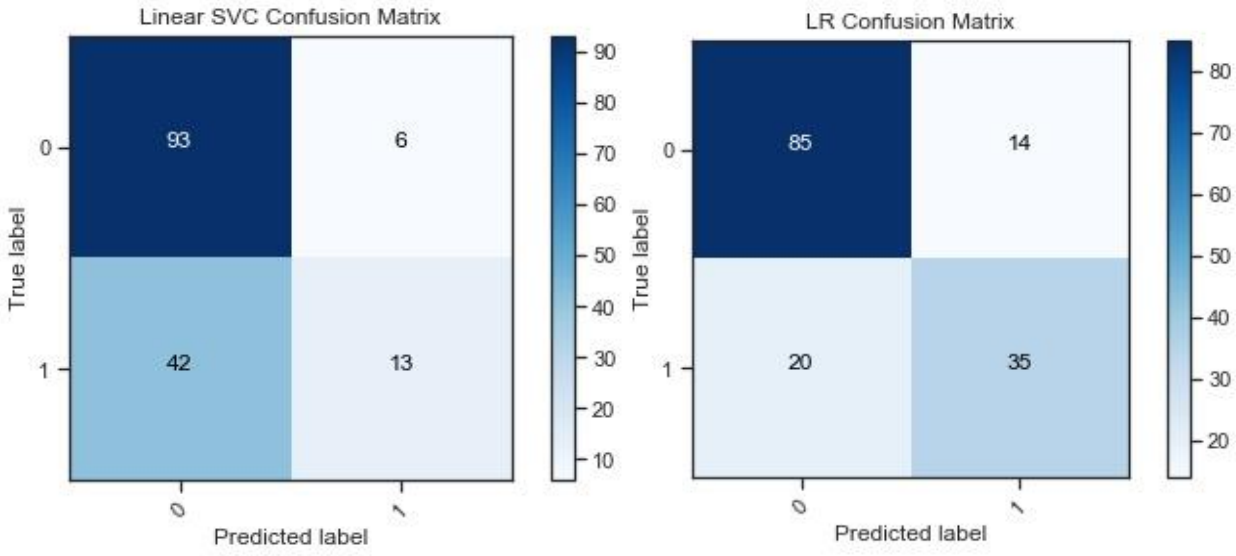
## Refinement:

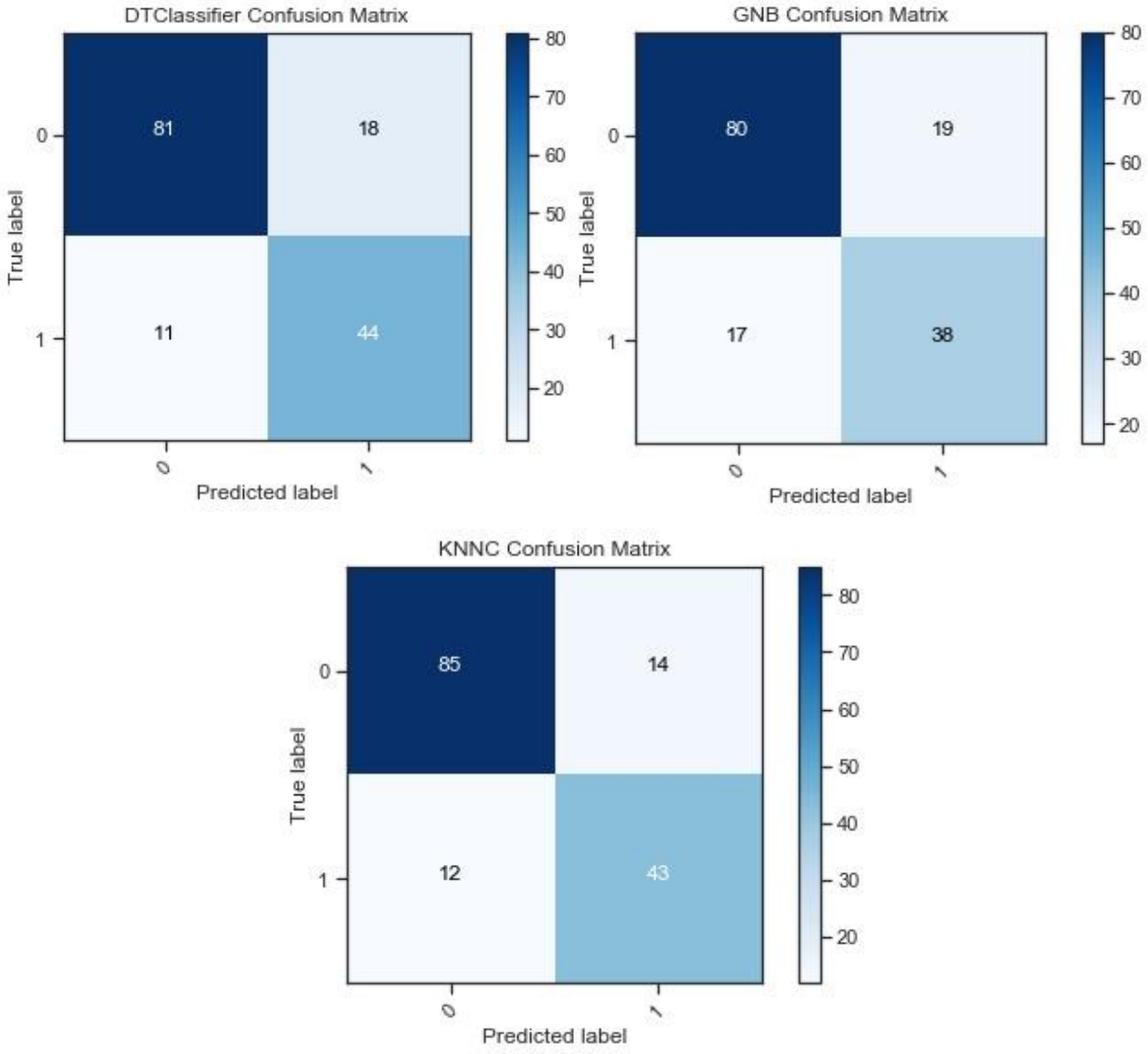
The preliminary model comparison for classifiers trained and tested on the dataset yielded the following F1 scores

Table 2. Preliminary Model Accuracies

| Classifier | F1 score |
|---|---|
| K-Nearest Neighbors | 0.7679 |
| Decision Tree | 0.7521 |
| Linear SVC (didn't converge) | 0.4474 |
| Logistic Regression | 0.6731 |
| Gaussian Naives Bayes | 0.6786 |

They can be interpreted easily using confusion matrix.






After the preliminary analysis of the models, the next step of refinement was to improve the accuracy through feature selection method. The following correlation map made through seaborn will help highlight important features in the dataset. Fig 8 shows the correlation heatmap a simpler representation of Fig 5.

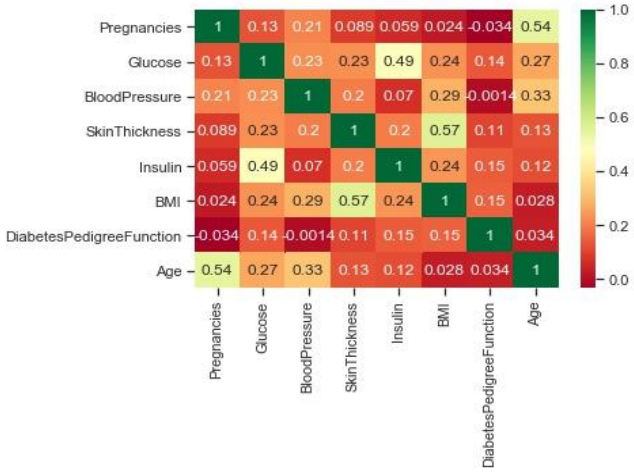



Figure 8. Feature selection seaborn correlation map

After that RandomForestClassifier is used to give a good estimate for the important features using parameters for n_estimators = 0 and random_state = 42.

Table 3. Important Features order

| Features | Values |
|---|---|
| Insulin | 0.361 |
| Glucose | 0.158 |
| SkinThickness | 0.143 |
| BMI | 0.088 |
| Age | 0.086 |
| DiabetesPedigreeFunction | 0.065 |
| BP | 0.050 |
| Pregnancies | 0.048 |

A new dataset was generated by using only the top 3 features – Insulin, Glucose and SkinThickness to measure the accuracy of the models. Table 4 lists the accuracy of the classifiers using the new feature selected dataset.

Color code - RED is decrease in accuracy, GREEN is increase and BLACK is same as before.

Table 4. Feature Selected Scores

| Classifier | F1 score |
|---|---|
| K-Nearest Neighbors | 0.7636 |
| Decision Tree | 0.7857 |
| Linear SVC (didn't converge) | 0 |
| Logistic Regression | 0.6465 |
| Gaussian Naives Bayes | 0.6199 |

From the Table 4 it can be observed that feature selection refinement negatively affected the performance of all classifiers except Decision Tree. It is not clear why but a good guess is that feature addition might be more helpful that feature selection. More on that in Improvements section.

Next step for refinement of the classifiers is to use Cross Validation technique to increase the accuracy. The following parameters and initializations were used for that purpose.

*kfold = KFold(n_splits = 10, random_state= 42)*

All the listed classifiers in Table 4 were used under this method. Table 5 shows the accuracy scores for cross validation. Color code - RED is decrease in accuracy, GREEN is increase and BLACK is same as before.

Table 5. Cross Validation Scores

| Classifier | F1 score |
|---|---|
| K-Nearest Neighbors | 0.8515 |
| Decision Tree | 0.8451 |
| Linear SVC (didn't converge) | 0.6336 |
| Logistic Regression | 0.7786 |
| Gaussian Naives Bayes | 0.7644 |

Cross validation technique had a positive effect on the classifiers and all of them performed better than the preliminary scores.

Next step of refinement was use of GridSearchCV. Here are the parameters used to test the accuracy of GradientBoostingClassifier.

*loss = ['deviance', 'exponential']*

*n_estimators = [30, 60]*

*max_depth = [1, 2, 3, 4, 5]*

*random_state = [0, 24]*

*parameters = {'loss': loss, 'n_estimators' : n_estimators, 'max_depth' : max_depth, 'random_state' : random_state}*

GridSearchCV gave a accuracy score of 0.8639. It seems to be performing as good as K-Nearest Neighbors and Decision Tree Classifier from Table 5.

Last step of refinement is to check if ensembling methods will produced higher accuracy than already attained by our classifiers. Under basic ensembling, we utilized VotingClassifier to select the best classifier of those listed in Table 5.

*ensemble_all = VotingClassifier(estimators = [('Logistic Regression', LR(C=0.1)),('KNN', KNNC(n_neighbors = 7)),('Decision Tree',DTClassifier())], voting = 'soft', weights = [1,1,1]).fit(X_train, y_train)*

*print('Score: ', ensemble_all.score(X_test, y_test))*

Basic ensembling gives an accuracy of 0.8506. It is on the higher side as expected, but not the best yet. Moving on to Boosting techniques, we first use LightGBM with K-Nearest Neighbors.

The following parameters are tested under lightGBM,

*param_test = {'learning_rate' : [0.01, 0.02, 0.03, 0.04, 0.05, 0.08, 0.1, 0.2, 0.3, 0.4],*

*'n_estimators' : [100, 200, 300, 400, 500, 600, 800, 1000, 1500, 2000],*

*'num_leaves': sp_randint(6, 50),*

*'min_child_samples': sp_randint(100, 500),*

*'min_child_weight': [1e-5, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3, 1e4],*

*'subsample': sp_uniform(loc=0.2, scale=0.8),*

*'max_depth': [-1, 1, 2, 3, 4, 5, 6, 7],*

*'colsample_bytree': sp_uniform(loc=0.4, scale=0.6),*

*'reg_alpha': [0, 1e-1, 1, 2, 5, 7, 10, 50, 100],*

*'reg_lambda': [0, 1e-1, 1, 5, 10, 20, 50, 100]}*

With voting classifier as the platform, we give equal weights to both light_classifier and knn_classifier. The accuracy reported by this method is 0.888. This is quite close to 0.9, which means our analysis is going in the right direction for accurately predicting diabetic patients.

To compare LightGBM's performance, we also check the results for XGBoost and RandomForestClassifier (Bagging) methods.

XGBoost gives an F1 score of 0.7928 and XGBoost with cross validation gives an F1 score of 0.8867 (very close to LightGBM). RandomForestClassifier gives an F1 score of .0804 and cross validated F1 score of 0.8699.

# RESULTS

## Model Evaluation and Validation:

The final model chosen for this binary prediction problem is LightGBM & KNN. It uses a set of parameters to tune through using grid search, thereby making it a robust model. This was one of the expected results as mentioned in problem statement, since Boosting techniques under Ensembling methods are quite robust in nature. The model uses a set of parameters to tune through using grid search and produces an accuracy of 0.888 to predict whether a female is diabetic or not.

Based on the use of extensive number of folds, the model is accurate, yet it is susceptible to overfitting.

## Justification:

The end result is greatly improved from the preliminary scores we observed. It should be noted that both XGBoost and RandomForestClassifier under cross validation technique scored close to LightGBM, yet the reason why I choose LightGBM is because it has faster training speed, lower memory usage, better accuracy and is also capable of handling large scale data. This would ensure if my model is used on a larger dataset of Pima Female Indians, it would still perform up to expectations.

As compared to the benchmark model, we can say in favor of the model generated that it does significantly better than 65% with the F1 score of 0.888. Hence it performs well enough to have adequately solved the problem.

# CONCLUSION

## Free-Form Visualization:

Features are the major quality of this project in my opinion. In the beginning, the feature correlations were discussed, and they proved to be quite different when comparing my naïve interpretation of the data to the real statistical analysis. However, the methods used to tune the model did not use feature selection for that matter as they proved to be less impactful. Fig 9 shows the heatmap of correlation from before the missing values were replaced by median values.

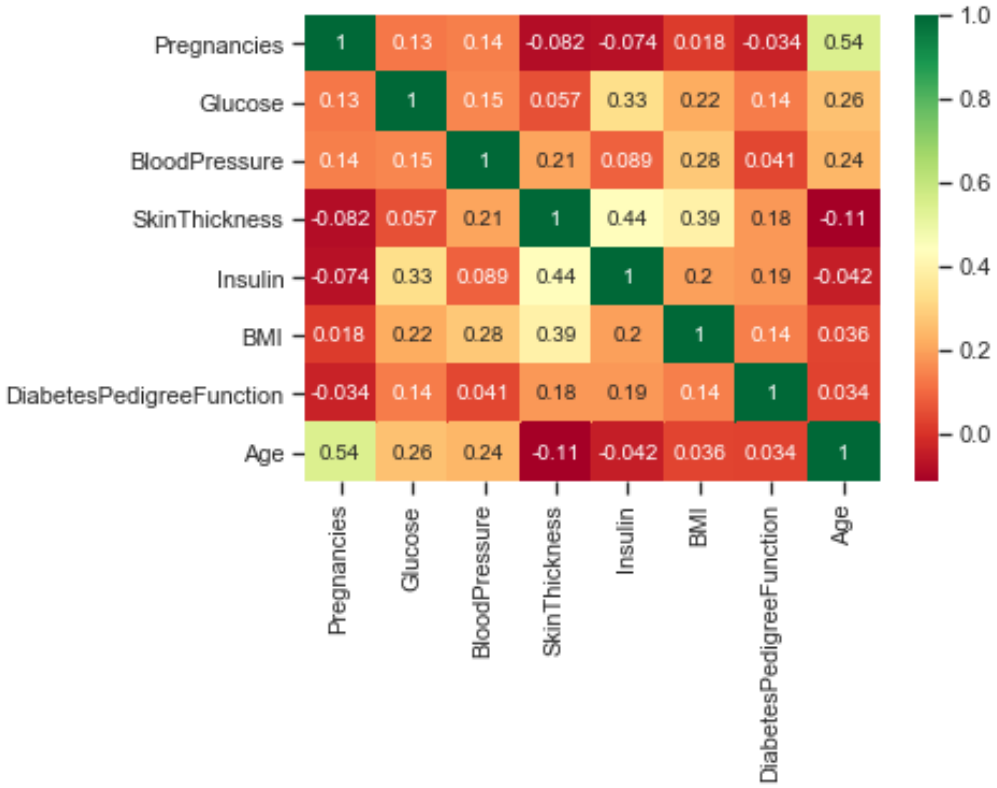



Figure 9. Feature correlation map before data preprocessing.

This map is different from the heatmap shown in Fig 8, as it can be concluded that the replacement of missing values has been key in shifting the rankings of correlations between the features themselves. Yet, I am confident that the justification for using median values to replace the missing values is sound and well placed for real life cases. Thus, although feature selection/addition is an important of creating a robust model, I have managed to achieve adequate results without its use.

## Reflection:

The ability to detect disease or disability early is well rewarded as it gives hope to people to change their lifestyle and improve their health. As diabetes becomes more prominent and impactful in our lives and those of our loved ones, it is now more than ever we need a robust and trustful method of detecting diabetes accurately. I chose this project because I am in threat of being diabetic and so working to generate an efficient model to predict diabetes with high accuracy is a good start for me. I also wanted to analyze my own understanding of the material on supervised learning as it has many applications in the field I study.

## Improvement:

There are many ways to improve upon the model performances. In general, I feel the use of median values to replace missing values is sound, yet it still has more opportunities of providing better statistics by using stratification. This would also have implications in feature selection/addition, where I mostly focused on reducing data size, however, feature addition might be more useful in creating a linear ranking trend between the features which would greatly improve our understanding of the dataset. Since this problem is binary in nature, more traditional approaches were used, instead deep learning or perception algorithms can be used as well which might further improve the performance of the model. Lastly, I didn't play enough with parameters of LightGBM, XGBoost and RandomForestClassifier which can certainly affect the performance of the model.

## References

https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

https://machinelearningmastery.com/logistic-regression-for-machine-learning/https://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html

https://machinelearningmastery.com/naive-bayes-for-machine-learning/

https://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf

https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

https://pythonprogramming.net/linear-svc-example-scikit-learn-svm-python/

https://www.niddk.nih.gov/health-information/diabetes/overview/what-is-diabetes

https://scikit-learn.org/stable/modules/preprocessing_targets.html#preprocessing-targets