

You have 2 free stories left this month. Sign up and get an extra one for free.



Picture Credits: www.diabetes.co.uk

PIMA Indian Diabetes Prediction

Predicting the onset of diabetes



Ishan Choudhary

Mar 12 · 12 min read ★

Diabetes is a chronic condition in which the body develops a resistance to insulin, a hormone which converts food into glucose. Diabetes affect many people worldwide and is normally divided into Type 1 and Type 2 diabetes. Both have different characteristics. This article intends to analyze and create a model on the PIMA Indian Diabetes dataset to predict if a particular observation is at a risk of developing diabetes,

given the independent factors. This article contains the methods followed to create a suitable model, including EDA along with the model.

Dataset

The dataset can be found on the Kaggle website. This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases and can be used to predict whether a patient has diabetes based on certain diagnostic factors. Starting off, I use Python 3.3 to implement the model. It is important to perform some basic analysis to get an overall idea of the dataset.

```
#Importing basic packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pandas_profiling import ProfileReport

#Importing the Dataset
diabetes = pd.read_csv("diabetes.csv")
dataset = diabetes

#EDA using Pandas Profiling
file = ProfileReport(dataset)
file.to_file(output_file='output.html')
```

Pandas profiling is an efficient way to get an overall as well as in-depth information about the dataset and the variables in it. However, caution must be exercised if the dataset is very large as *Pandas Profiling* is time-consuming. Since the dataset has only 768 observations and 9 columns, we use this function. The output gets saved as an *HTML* report in the working directory.

Overview

Dataset info		Variables types	
Number of variables	9	NUM	8
Number of observations	768	BOOL	1
Missing cells	0 (0.0%)		
Duplicate rows	0 (0.0%)		
Total size in memory	54.1 KiB		
Average record size in memory	72.1 B		
Warnings			
BloodPressure has 35 (4.6%) zeros			

BMI has 11 (1.4%) zeros
Insulin has 374 (48.7%) zeros
Pregnancies has 111 (14.5%) zeros
SkinThickness has 227 (29.6%) zeros

Zeros
Zeros
Zeros
Zeros

Overview of the Dataset

We can see the basic information about the dataset such as the size, missing values, etc. On the top right, we see 8 of numerical columns and 1 Boolean column (which is our dependent variable). In the lower panel, (%) of zeros are given in every column, which will be an useful information for us later. We do not have any categorical variable as an independent variable.

Exploratory Data Analysis (EDA)

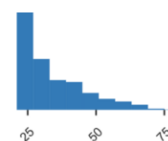
Having observed the basic characteristics of the dataset, we now move on to observe characteristics of the variables involved in the study. Again, *Pandas Profiling* comes to our rescue. The same HTML report gives information on the variables.

Age

Real number ($\mathbb{R}_{\geq 0}$)

Distinct count	52
Unique (%)	6.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	33.24088542
Minimum	21
Maximum	81
Zeros	0
Zeros (%)	0.0%
Memory size	6.1 KiB



[Toggle details](#)

Statistics

[Histogram\(s\)](#)

[Common values](#)

[Extreme values](#)

Quantile statistics

Minimum	21
5-th percentile	21
Q1	24
median	29
Q3	41
95-th percentile	58
Maximum	81
Range	60
Interquartile range (IQR)	17

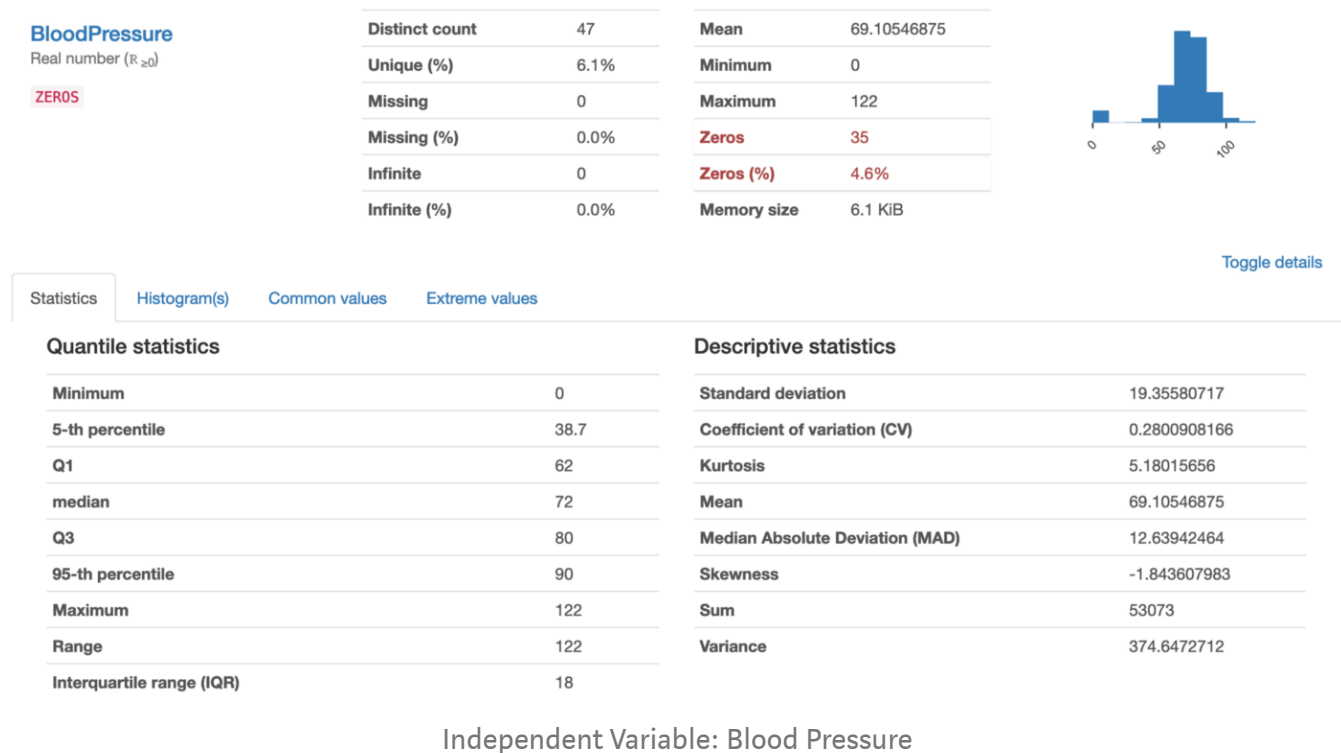
Descriptive statistics

Standard deviation	11.76023154
Coefficient of variation (CV)	0.3537881556
Kurtosis	0.6431588885
Mean	33.24088542
Median Absolute Deviation (MAD)	9.586405436
Skewness	1.129596701
Sum	25529
Variance	138.3030459

Independent Variable: Age

Let's have a look at *Age*. By having a quick look at the histogram on the top right, we can observe much of the characteristics. The variable does not follow a normal distribution as it is skewed to the right (or positively skewed). The average age is 33 whereas the median age is 29. This further confirms our analysis (as in case of normal distribution, the mean should be approximately equal to the median). Most

importantly, none of the values seem abnormal, that is, the minimum age of 21 and the maximum age of 81 are possible for the study. Let's have a look at other variables.



Doing the same for Blood Pressure, we can see that the variable can approximate to a normal distribution. However, we can not confirm that visually. We hence perform the Shapiro-Wilk test of normality. The null hypothesis (H_0) is that the data is normal.

```
from scipy.stats import shapiro

stat, p = shapiro(dataset['BloodPressure'])
print('Statistics=%.3f, p=%.3f' % (stat, p))

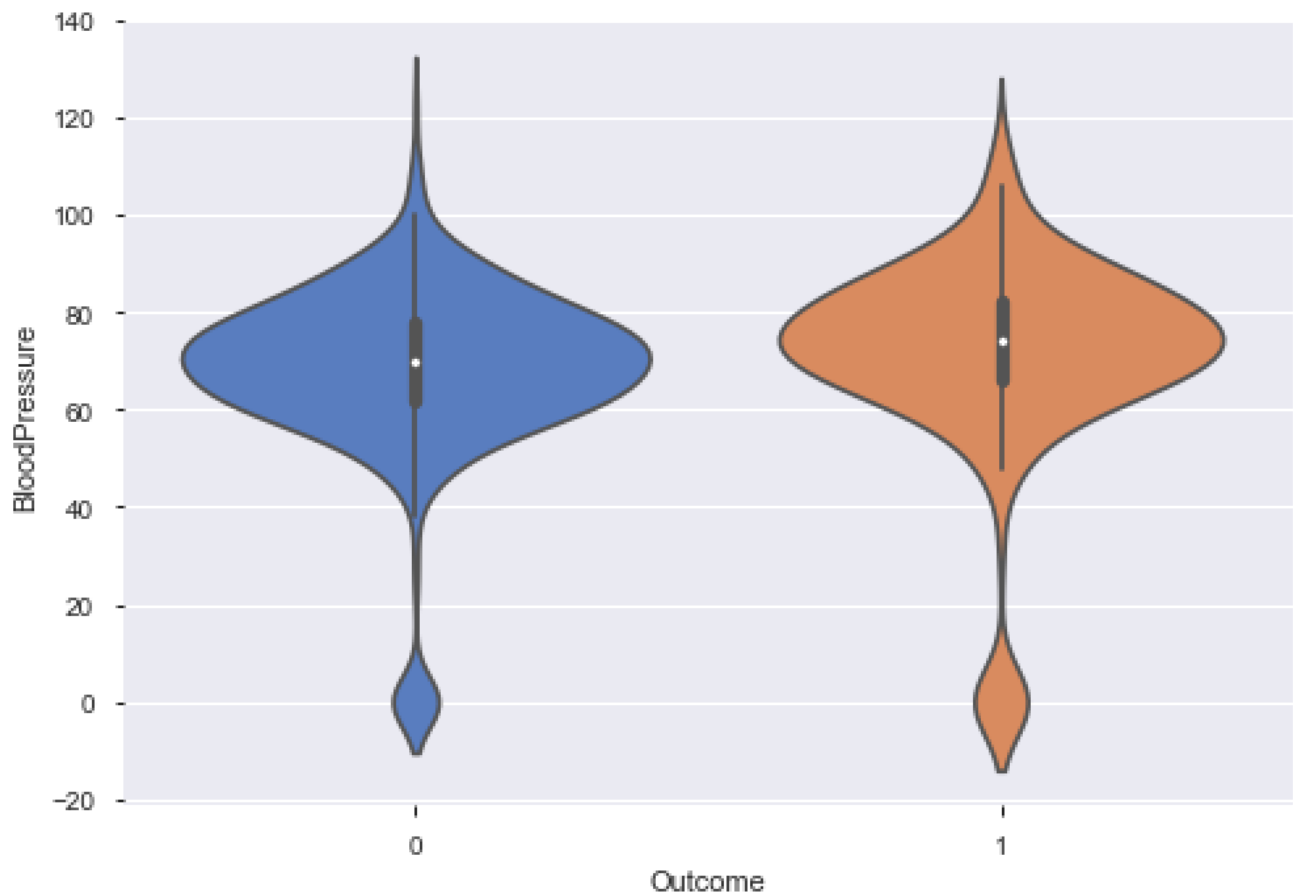
Statistics=0.819, p=0.000
```

The p -value is less than 0.001, implying that at 5% Level of Significance (LOS) we reject our null hypothesis (H_0). Therefore, the variable does not follow a normal distribution.

Secondly, if we observe the minimum value for Blood Pressure, it is given as 0 (which is not possible). Therefore, there is incorrect information given. We have two choices now. One is to drop such observations (which results in 4.6% loss of data) or we can replace such values with median (imputing). I prefer imputing as we have a small dataset (768 observation only). Hence, every information is important.

We can simply replace the zero values by median or we can classify the variable based on outcome and then replace the two separately. The latter seems more efficient. We draw a violin graph to have a look at the behavior.

```
#Classifying the Blood Pressure based on class
ax = sns.violinplot(x="Outcome", y="BloodPressure", data=dataset,
palette="muted", split=True)
```



We get the above graph. If we observe minutely, we can see that the box plot for 1 (*Diabetic*) inside the *violin* is a little more away from the horizontal axis than the box plot for 0 (*Non Diabetic*). It can be implied that diabetics seem to have a higher blood pressure than the non-diabetics. The bottom tail of the *violins* indicates the zero values we need to replace. We will replace the zeros for 1 with median of 1 and same for 0.

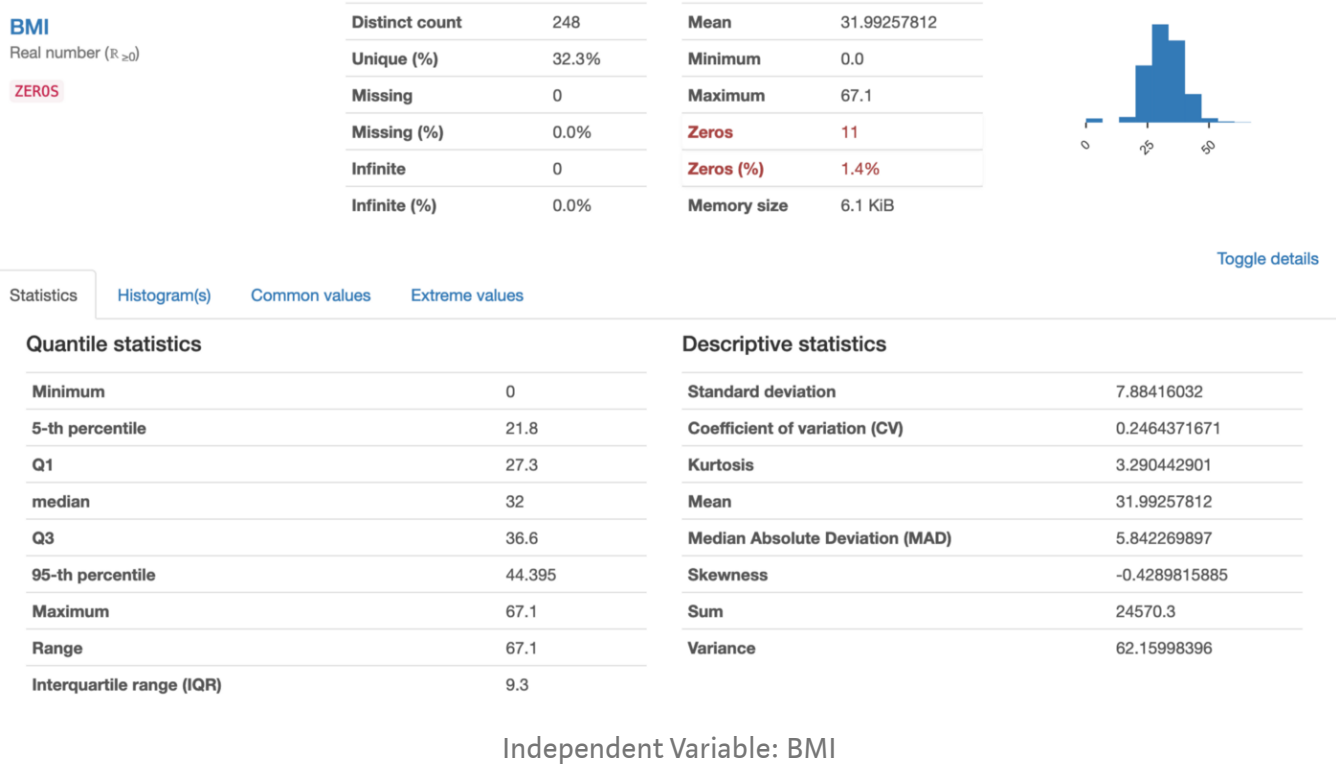
```
#Replacing the zero-values for Blood Pressure
df1 = dataset.loc[dataset['Outcome'] == 1]
df2 = dataset.loc[dataset['Outcome'] == 0]

df1 = df1.replace({'BloodPressure':0},
np.median(df1['BloodPressure']))
```

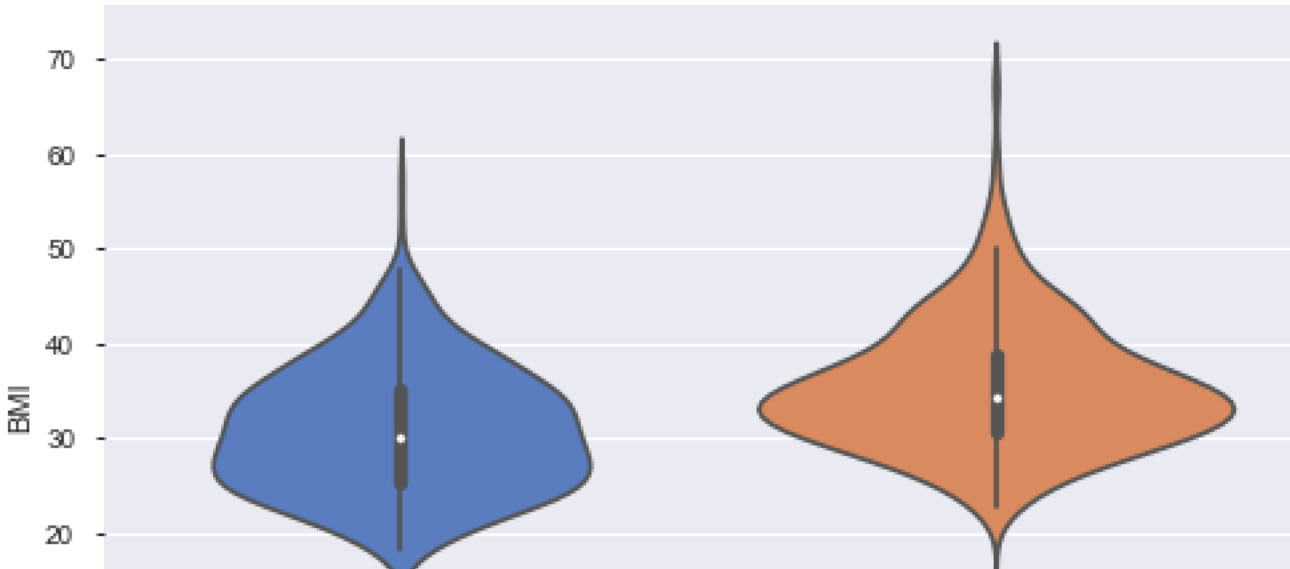
```
df2 = df2.replace({'BloodPressure':0},
np.median(df2['BloodPressure']))

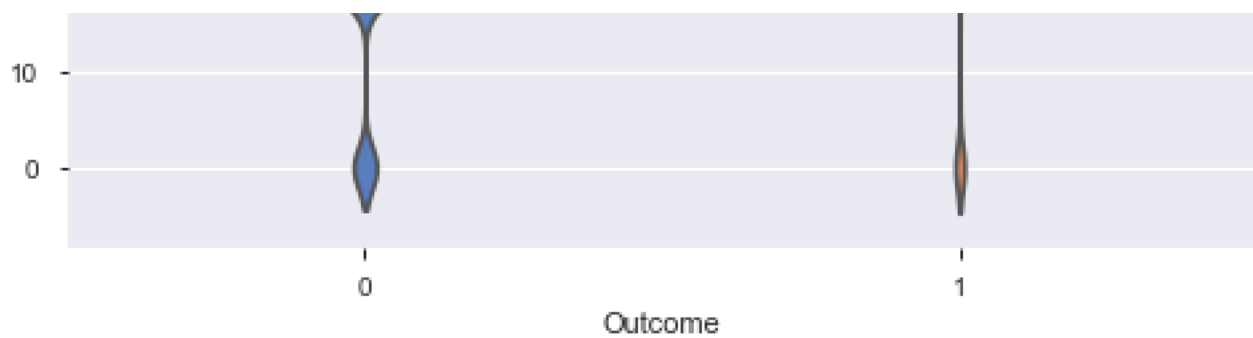
dataframe = [df1, df2]
dataset = pd.concat(dataframe)
```

There won't be any zero-values in *BloodPressure* column post this. Let's move to the next variable.



The variable seems to be closely following the normal distribution as the mean and median are approximately equal. However, it faces the same problem as before, that is, the existence of zero-values. Let's draw a violin plot for this variable.

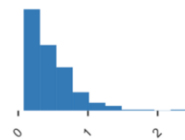




We can observe that the 1 follows normal distribution, while 0 doesn't. Also, BMI for diabetics is more than BMI for non-diabetics (can be observed using box plot). We do the same treatment for zero-values in BMI that we followed for zero-values for Blood Pressure. In this case, we can also replace by mean, however, I will stick with median. We run the same code after changing the column name. Moving on to the next variable.

DiabetesPedigreeFunction
Real number ($\mathbb{R}_{\geq 0}$)

Distinct count	517	Mean	0.4718763021
Unique (%)	67.3%	Minimum	0.078
Missing	0	Maximum	2.42
Missing (%)	0.0%	Zeros	0
Infinite	0	Zeros (%)	0.0%
Infinite (%)	0.0%	Memory size	6.1 KiB



[Toggle details](#)

Statistics

[Histogram\(s\)](#)

[Common values](#)

[Extreme values](#)

Quantile statistics

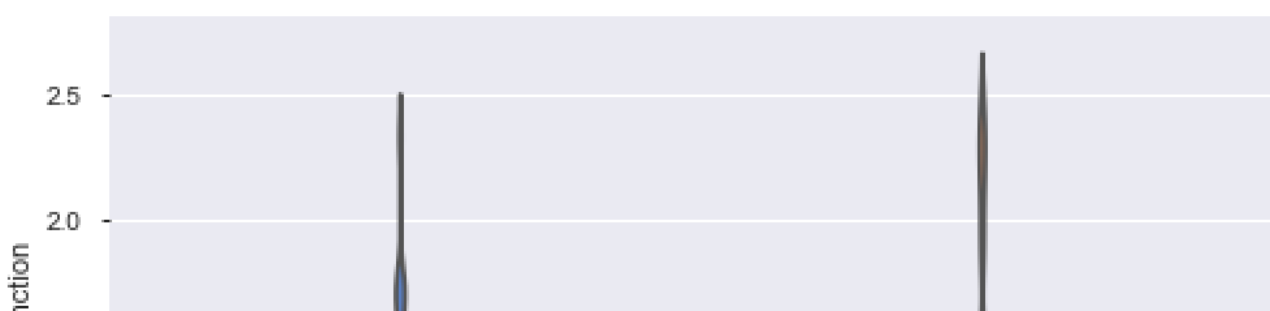
Minimum	0.078
5-th percentile	0.14035
Q1	0.24375
median	0.3725
Q3	0.62625
95-th percentile	1.13285
Maximum	2.42
Range	2.342
Interquartile range (IQR)	0.3825

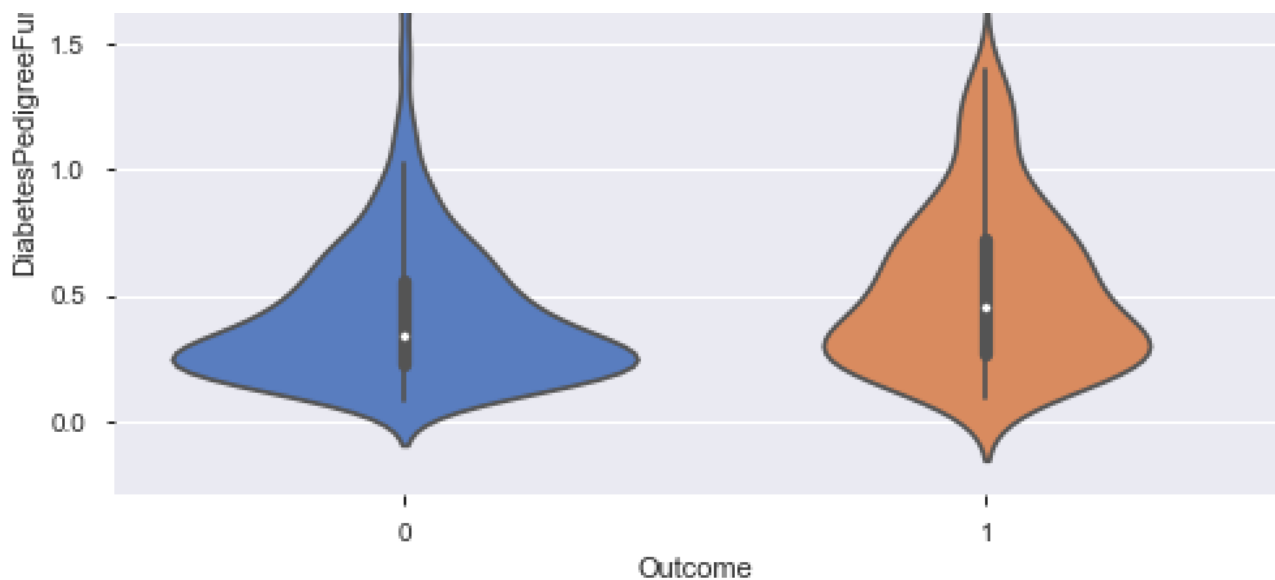
Descriptive statistics

Standard deviation	0.331328595
Coefficient of variation (CV)	0.7021513764
Kurtosis	5.594953528
Mean	0.4718763021
Median Absolute Deviation (MAD)	0.24730857
Skewness	1.919911066
Sum	362.401
Variance	0.1097786379

Independent Variable: DiabetesPedigreeFunction

Diabetes Pedigree Function is a positively skewed variable with no zero values. We use the same violin plot to observe the characteristics.



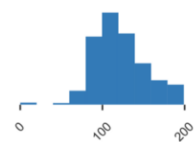


Same hypothesis can be formed. Diabetics seem to have a higher pedigree function than the non-diabetics. Moving on to the 5th independent variable.

Glucose

Real number ($\mathbb{R}_{\geq 0}$)

Distinct count	136	Mean	120.8945312
Unique (%)	17.7%	Minimum	0
Missing	0	Maximum	199
Missing (%)	0.0%	Zeros	5
Infinite	0	Zeros (%)	0.7%
Infinite (%)	0.0%	Memory size	6.1 KiB



[Toggle details](#)

Statistics

[Histogram\(s\)](#)

[Common values](#)

[Extreme values](#)

Quantile statistics

Minimum	0
5-th percentile	79
Q1	99
median	117
Q3	140.25
95-th percentile	181
Maximum	199
Range	199
Interquartile range (IQR)	41.25

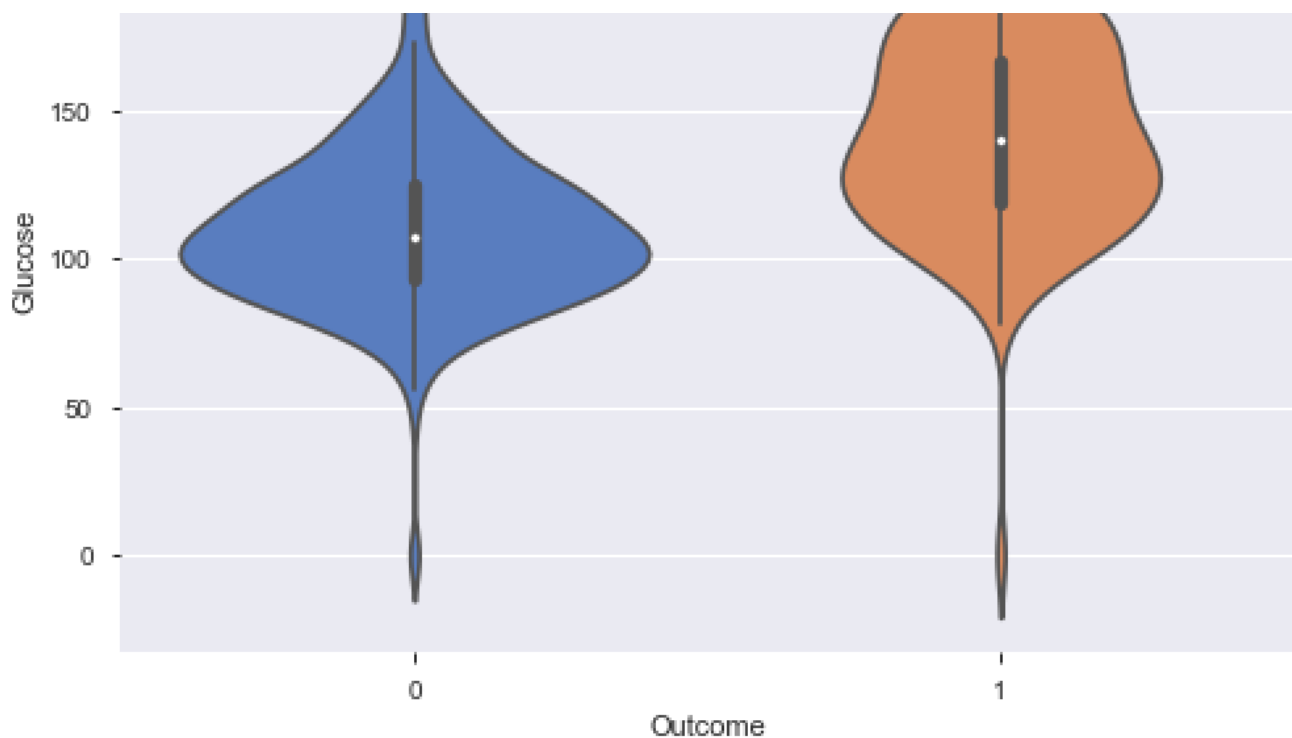
Descriptive statistics

Standard deviation	31.9726182
Coefficient of variation (CV)	0.2644670347
Kurtosis	0.6407798204
Mean	120.8945312
Median Absolute Deviation (MAD)	25.18179321
Skewness	0.1737535018
Sum	92847
Variance	1022.248314

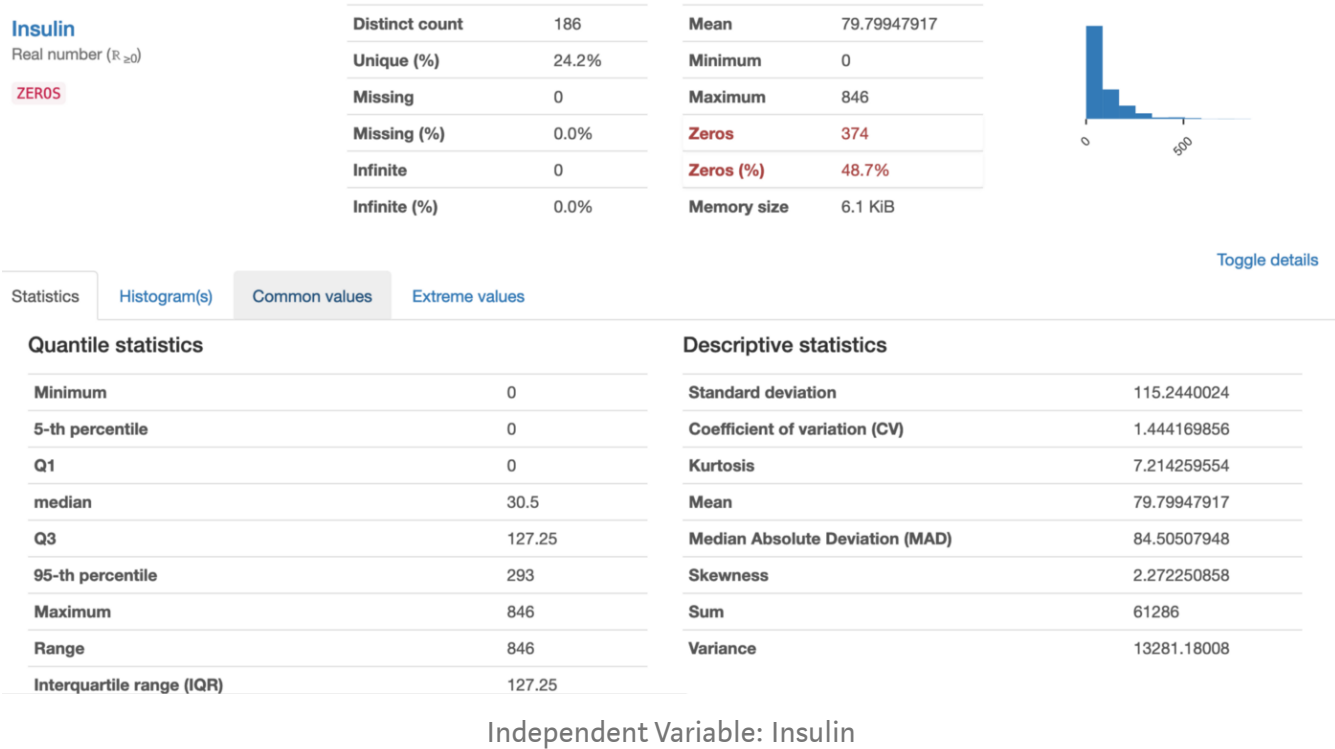
Independent Variable: Glucose

Analyzing *Glucose*, we observe the variable not following the normal distribution. We encounter zero-values in this instance as well. There are 5 such values for which treatment is required. We perform the same treatment as before, replacing by median (class-wise).



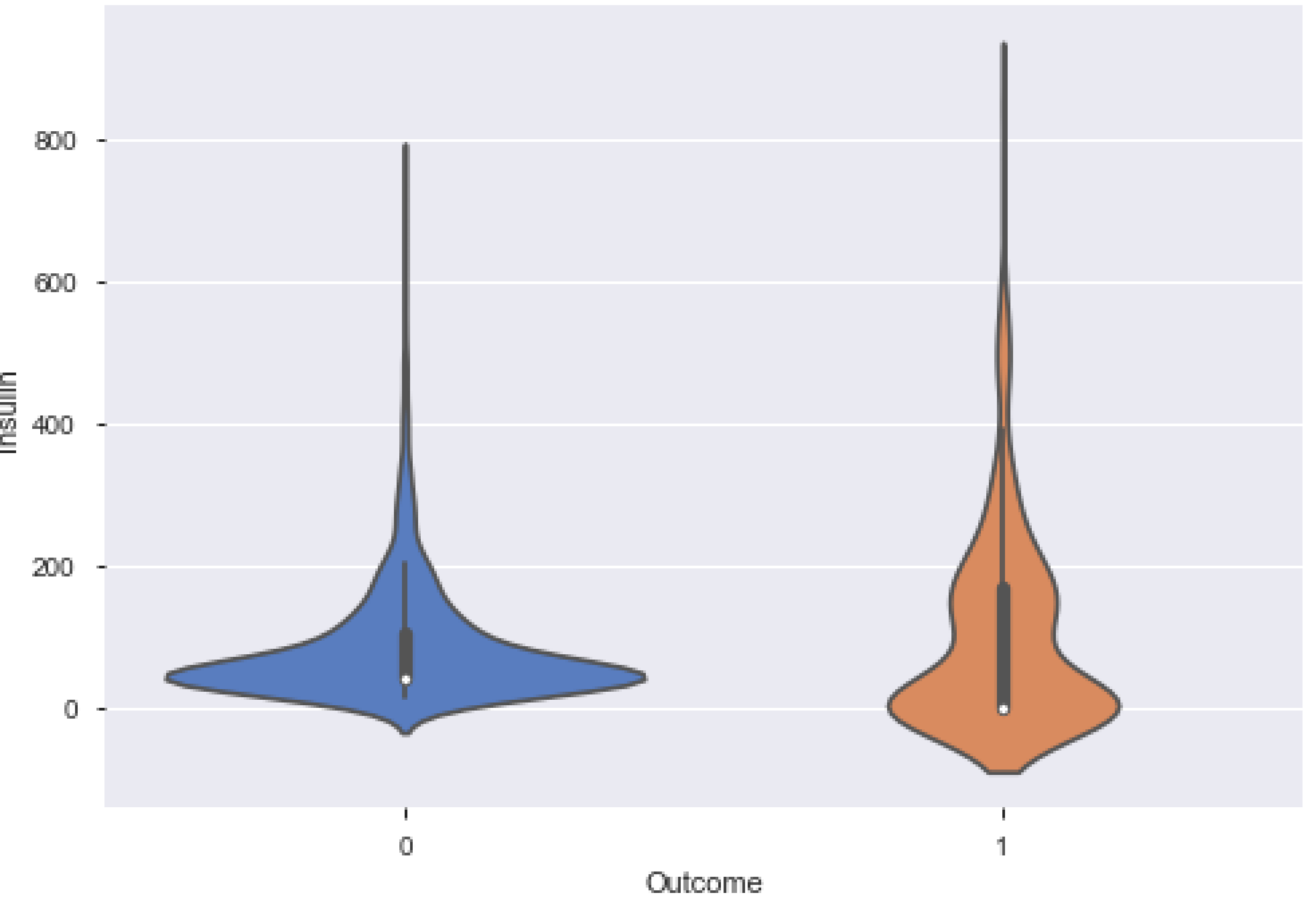


Observing the violin plot, we see a massive vertical distance between the box-plot for Diabetics and Non-Diabetics. This indicates that Glucose can be a very important variable in model-building. After treating the zero values, we move to the next variable.



As mentioned in the data dictionary available with the dataset, *Insulin* is the 2-Hour serum insulin (mu U/ml). The variable is positively skewed. However, the occurrence

of zero-values is high in this case, making up 48.7% of the data. This data has to be imputed. We first treat these.

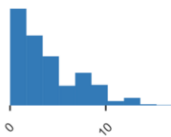


After treating the zero-values, we derive at this graph. We can still see 0 as the median for Insulin for *Diabetics*. However, for *Non-Diabetics*, Insulin is a little higher. It can be roughly hypothesized that Insulin for Diabetics is lower than Non-Diabetics. Moving forward to Pregnancies.

Pregnancies
Real number (R_{≥0})
ZEROS

Distinct count	17
Unique (%)	2.2%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	3.845052083
Minimum	0
Maximum	17
Zeros	111
Zeros (%)	14.5%
Memory size	6.1 KiB



Statistics

Histogram(s)Common valuesExtreme values

Quantile statistics

Minimum	0
5-th percentile	0
Q1	1
median	3
Q3	6
95-th percentile	10
Maximum	17
Range	17

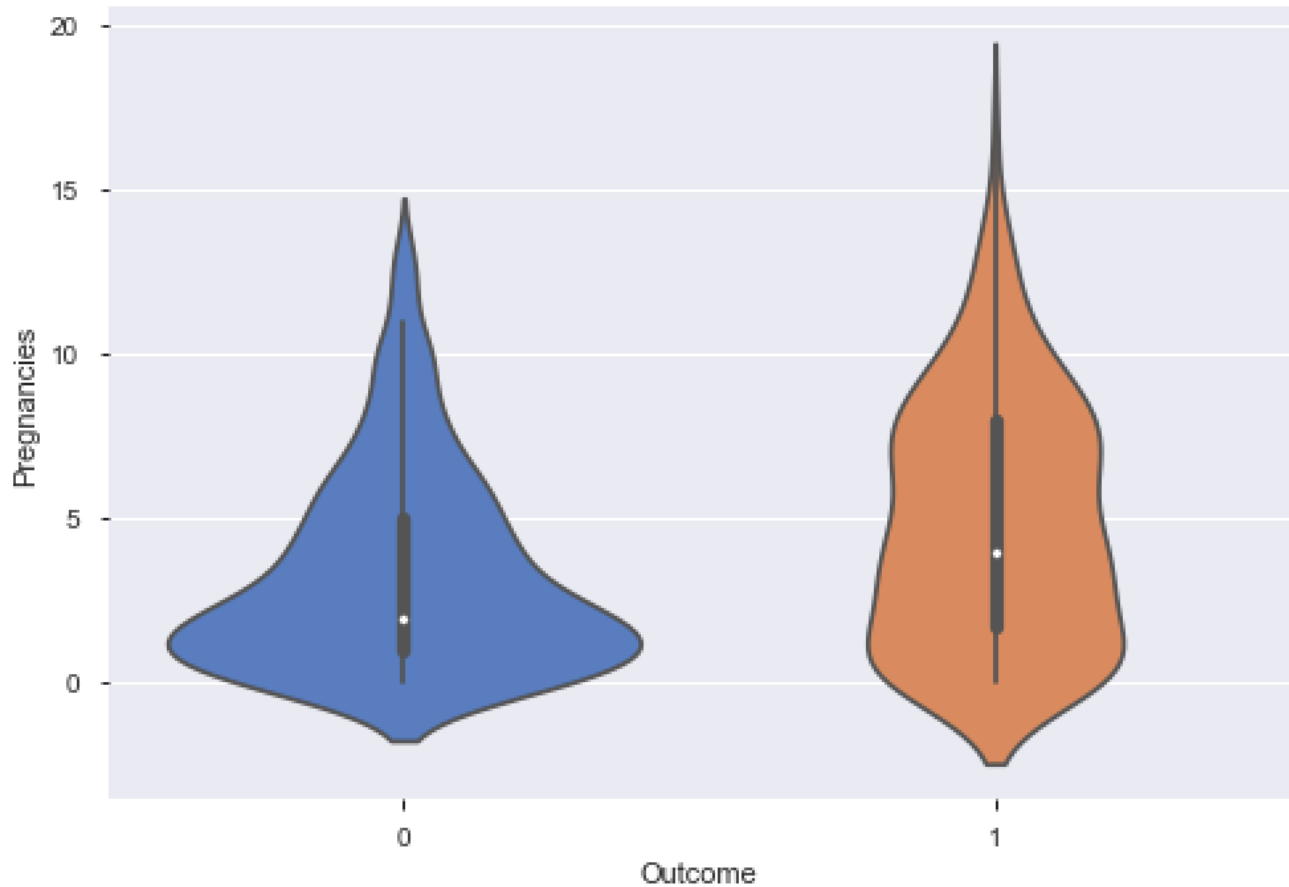
Descriptive statistics

Standard deviation	3.369578063
Coefficient of variation (CV)	0.8763413316
Kurtosis	0.1592197775
Mean	3.845052083
Median Absolute Deviation (MAD)	2.771620009
Skewness	0.9016739792
Sum	2953
Variance	11.35405632

[Toggle details](#)

Independent Variable: Pregnancies

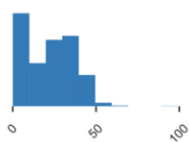
The variable is positively skewed with 14.5% zero values. We need not treat zero values as its not an abnormal occurrence. However, given the fact that 85.5% of the values is non-zero, we can infer that this study is done for females only. Looking at the behavior using violin plots we observe diabetic women had more pregnancies than non-diabetic.



SkinThickness
Real number (R ≥0)
ZEROS

Distinct count	51
Unique (%)	6.6%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%

Mean	20.53645833
Minimum	0
Maximum	99
Zeros	227
Zeros (%)	29.6%
Memory size	6.1 KIB



[Toggle details](#)

Statistics [Histogram\(s\)](#) [Common values](#) [Extreme values](#)

Quantile statistics

Minimum	0
5-th percentile	0
Q1	0
median	23
Q3	32
95-th percentile	44

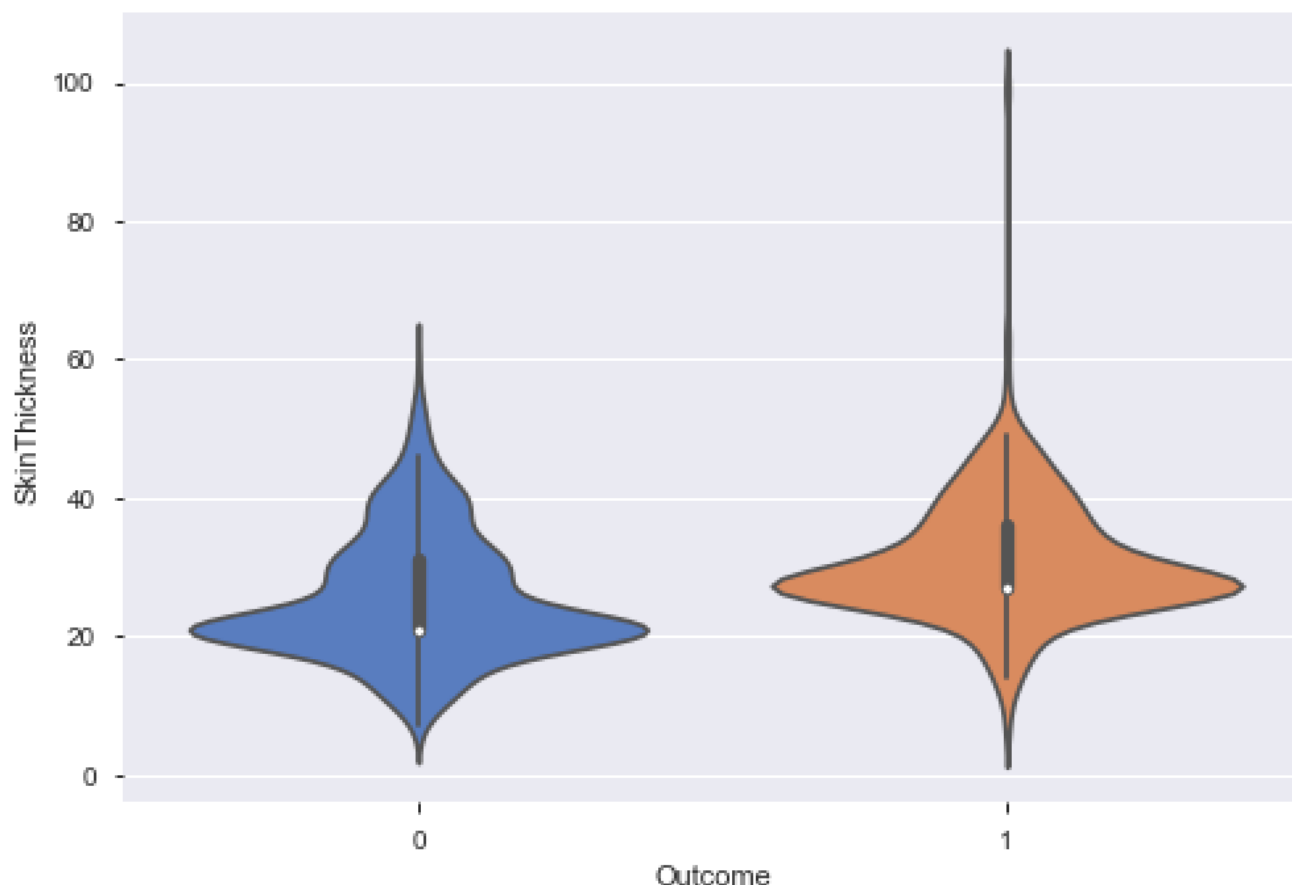
Descriptive statistics

Standard deviation	15.95221757
Coefficient of variation (CV)	0.776775494
Kurtosis	-0.5200718662
Mean	20.53645833
Median Absolute Deviation (MAD)	13.65962728
Skewness	0.1093724965

Maximum	99	Sum	15772
Range	99	Variance	254.4732453
Interquartile range (IQR)	32		

Independent Variable: SkinThickness

Moving on to our last independent variable, we observe the same patterns as most of the previous variable. The data is positively skewed with 29.6% of zero values. After treating, we use the violin plot to observe a crude relationship.



Skin Thickness for Diabetics is more than that of Non-Diabetics.

Our Dependent Variable '*Outcome*' takes Boolean values 0 and 1. 0 indicates non-diabetic and 1 indicates diabetic. To examine the occurrence, we use a simple bar plot

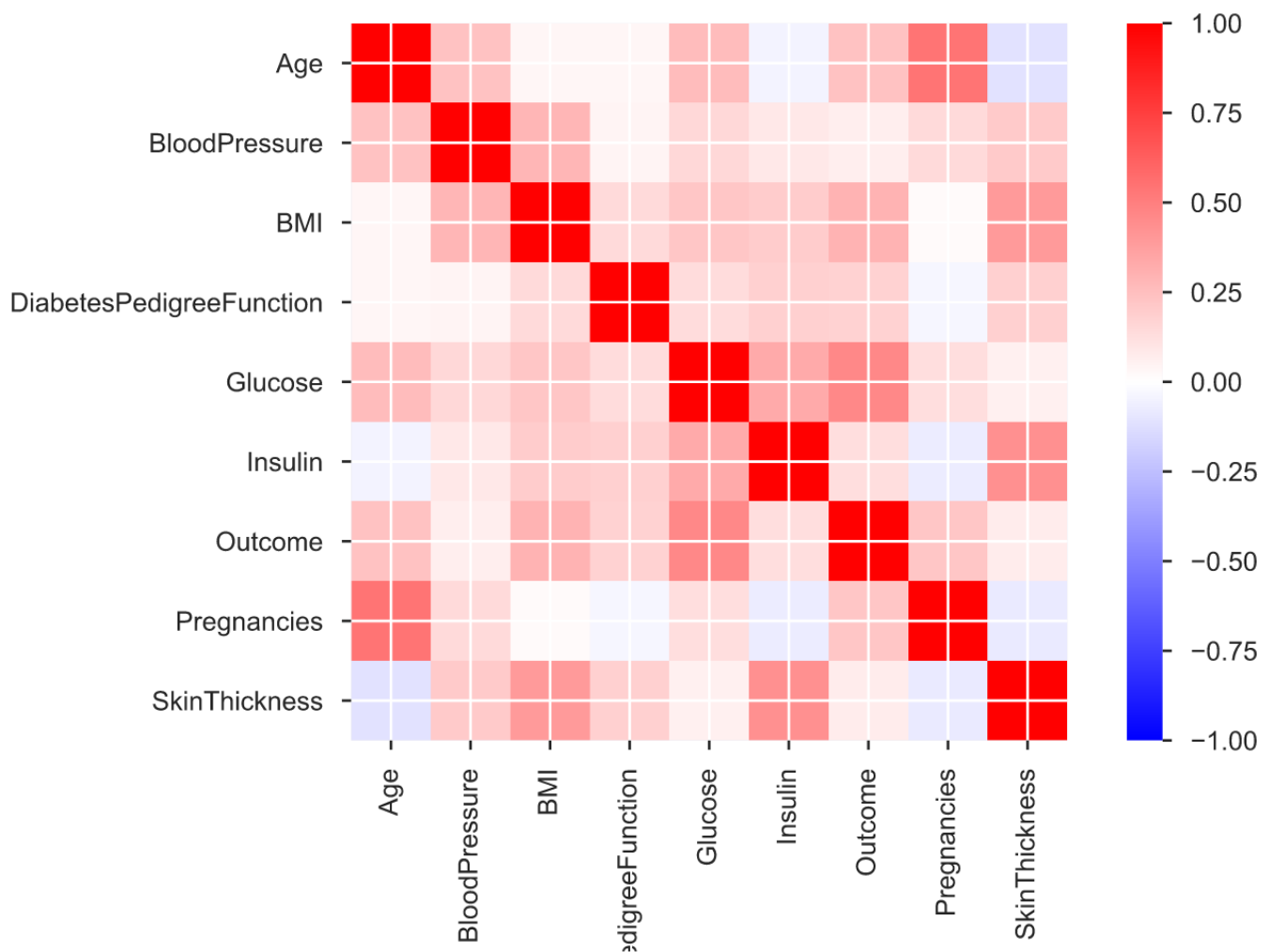
Value	Count	Frequency (%)
0	500	65.1%
1	268	34.9%

Frequency of 0 and 1 in Outcome

The imbalance in the data can be clearly seen with 0 (*Non-Diabetic*) being the modal class. We will treat this imbalance later in the process.

Checking for Multi collinearity

The correlation matrix below uses Pearson's correlation coefficient to illustrate the relationship between variables. From the figure, a significant correlation can be observed between *Pregnancies* and *Age*. To further confirm, we calculate the correlation coefficient.



```
from scipy.stats import pearsonr

corr, _ = pearsonr(dataset['Age'], dataset['Pregnancies'])
print('Pearsons correlation: %.3f' % corr)
Pearsons correlation: 0.544
```

The *correlation coefficient* (r) is 0.544. By a rule of thumb, in case of an r above 0.70, multi-collinearity is expected. Hence, no significant case of multi-collinearity is observed.

Treating Outliers and Non-Normality

Outliers are extreme values existing in the dataset. It is necessary to treat outliers if a distance-based algorithm (logistic regression, SVM, etc) is applied on the dataset. Outliers do not affect a tree-based algorithm. Since we will use both distance and tree-based algorithms, we will scale our data to treat outliers. We use *Standard Scaler* for the process. Standard Scaler transforms the feature by subtracting the mean and dividing with the standard deviation. This way the feature also gets close to standard normal distribution with mean 0.

```
#Splitting the data into dependent and independent variables
Y = dataset.Outcome
x = dataset.drop('Outcome', axis = 1)
columns = x.columns

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(x)

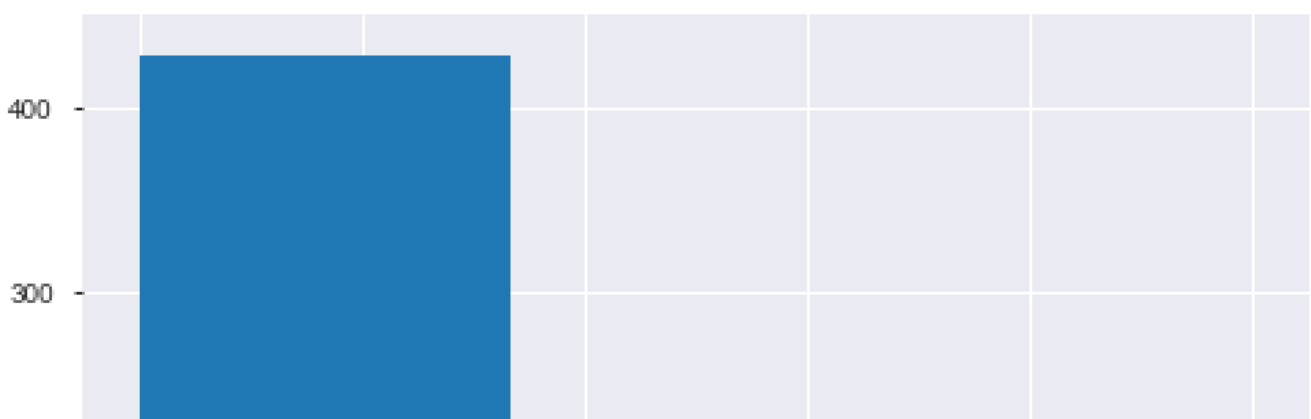
data_x = pd.DataFrame(X, columns = columns)
```

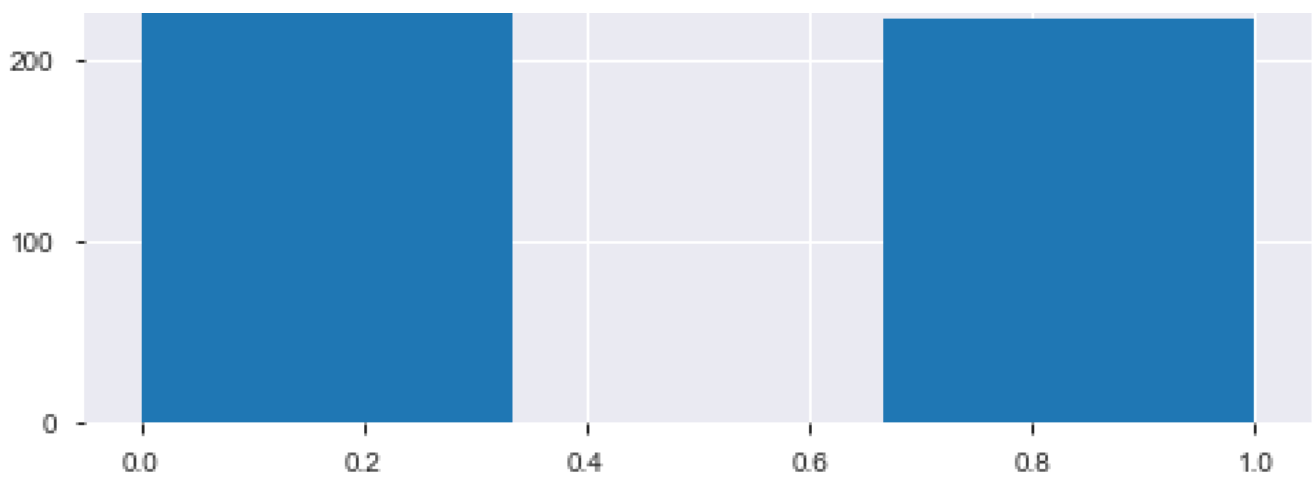
We have scaled our X values.

Splitting the dataset into Training and Test data

We now split our processed dataset into Training and Test data. The Test data size is take to be 15% of the entire data (which means 115 observations) and the model will be trained on 653 observations.

```
#Splitting the data into training and test
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data_x, Y,
test_size = 0.15, random_state = 45)
```





Plot showing frequency of 0 and 1 in Y train

A huge imbalance can be observed in the *y_train* values. To overcome this problem, we use the SMOTE technique.

Synthetic Minority Oversampling Technique (SMOTE) is used to remove the imbalance in the training data by creating samples using the current data. It does not create duplicates. Remember it is always done on the Training Data and not on the original data as the Test Data should only contain real-life values and not the synthetic sample.

```
from imblearn.over_sampling import SMOTE

smt = SMOTE()

x_train, y_train = smt.fit_sample(x_train, y_train)

np.bincount(y_train)
Out[74]: array([430, 430])
```

We now have a balanced Training data.

Our data is now prepared to fit a model

Model Fitting: Logistic Regression

The first model we fit on the training data is the Logistic Regression.

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(x_train, y_train)
```

```

y_pred = logreg.predict(x_test)
print('Accuracy of logistic regression classifier on test set:
{:.2f}'.format(logreg.score(x_test, y_test)))

Out[76]: Accuracy of logistic regression classifier on test set:
0.73

```

We get a **73% accuracy** score on the test data.

```

print(f1_score(y_test, y_pred, average="macro"))
print(precision_score(y_test, y_pred, average="macro"))
print(recall_score(y_test, y_pred, average="macro"))
0.723703419131771
0.7220530003045994
0.7263975155279503

```

Our Precision for the model stands at **0.722**. This indicates that 72% of the time our model classified the patients in a high risk category when they actually had a high risk of getting diabetes.

The Recall/Sensitivity is **0.726**, implying of 72% of the time people having actually having high risk were classified correctly by our model.

Model Fitting: Support Vector Machine (Kernel: rbf)

The first model we fit on the training data is the Support Vector Machine (SVM). SVM uses many kernels to classify the data. We use *rbf/Gaussian* kernel to fit the first model.

```

from sklearn.svm import SVC

classifier_rbf = SVC(kernel = 'rbf')
classifier_rbf.fit(x_train, y_train)

y_pred = classifier_rbf.predict(x_test)

print('Accuracy of SVC (RBF) classifier on test set:
{:.2f}'.format(classifier_rbf.score(x_test, y_test)))
Out[76]: Accuracy of SVC (RBF) classifier on test set: 0.75

print(f1_score(y_test, y_pred, average="macro"))
print(precision_score(y_test, y_pred, average="macro"))
print(recall_score(y_test, y_pred, average="macro"))
0.7431080565101182
0.7410256410256411
0.7481366459627329

```


We have an improved accuracy using SVM with rbf kernel. The model accuracy comes to **75%**, with improved Precision and Recall values compared to Logistic Regression.

Model Fitting: Random Forest

We use Random Forest Classifier, with 300 trees (derived at after tuning the model) to fit a model on the data.

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=300, bootstrap = True,
max_features = 'sqrt')

model.fit(x_train, y_train)

y_pred = model.predict(x_test)
print('Accuracy of Random Forest on test set:
{:.2f}'.format(model.score(x_test, y_test)))

Out[95]: Accuracy of Random Forest on test set: 0.88

print(f1_score(y_test, y_pred, average="macro"))
print(precision_score(y_test, y_pred, average="macro"))
print(recall_score(y_test, y_pred, average="macro"))
0.8729264475743349
0.8762626262626263
0.8701863354037267
```

We get the highest accuracy for Random Forest, with the score reaching **88%**. This implies, our model predicted classified correctly 88% of the times.

The *Precision* score stood at 0.876, implying our model correctly classified observations with high risk in the high risk category **87.6%** of the times. The *Recall* stood at **0.870**.

We also have an F1 score of 0.872. The F1 score is the harmonic mean of precision and recall. It assigns equal weight to both the metrics. However, for our analysis it is relatively more important for the model to have low false negative cases (as it will be dangerous to classify high risk patients in low risk category). Therefore, we individually look at Precision and Recall.



The figure above shows the relative importance of features and their contribution to the model. Since it was a small dataset with less columns, I didn't use Feature Selection technique such as PCA.

Conclusion

We thus select the Random Forest Classifier as the right model due to high accuracy, precision and recall score. One reason why Random Forest Classifier showed an improved performance was because of the presence of outliers. As mentioned before, since Random Forest is not a distance based algorithm, it is not much affected by outliers, whereas distance based algorithm such as Logistic Regression and Support Vector showed a lower performance.

Based on the feature importance:

1. Glucose is the most important factor in determining the onset of diabetes followed by BMI and Age.

2. Other factors such as Diabetes Pedigree Function, Pregnancies, Blood Pressure, Skin Thickness and Insulin also contributes to the prediction.

As we can see, the results derived from Feature Importance makes sense as one of the first things that actually is monitored in high-risk patients is the Glucose level. An increased BMI might also indicate a risk of developing Type II Diabetes. Normally, especially in case of Type II Diabetes, there is a high risk of developing as the age of a person increases (given other factors).

We now come to the end of the project. I did not go in-depth of the techniques I used. However, there are some really good articles that helped me in doing the same.

As always, I welcome constructive criticism, feedback, and discussion. I can be reached on Gmail: icy.algorithms@gmail.com

References

Dataset and Data Information: <https://www.kaggle.com/uciml/pima-indians-diabetes-database>

Pandas Profiling: <https://towardsdatascience.com/speed-up-your-exploratory-data-analysis-with-pandas-profiling-88b33dc53625>

Violin Plots: <https://seaborn.pydata.org/generated/seaborn.violinplot.html>

Shapiro-Wilk Test for Normality: <https://machinelearningmastery.com/a-gentle-introduction-to-normality-tests-in-python/>

Scaling/Outlier Treatment: <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>

Outlier Treatment: <https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba>

SMOTE: <https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/>

Precision and Recall: <https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c>

[Random Forest](#)

[Predictive Analytics](#)

[Machine Learning](#)

[Diabetes](#)

[Exploratory Data Analysis](#)

[About](#) [Help](#) [Legal](#)

Get the Medium app

