Senior-AI-Software-Engineer for Snapp

Task: Pima_Indians_Diabetes

Abbas Biniaz

May,2022

1.Introduction

The Pima Indian Diabetes Dataset, originally from the National Institute of Diabetes and Digestive and Kidney Diseases, contains information of 768 women from a population near Phoenix, Arizona, USA. The outcome tested was Diabetes, 258 tested positive and 500 tested negative. Therefore, there is one target (dependent) variable and the following attributes (TYNECKI, 2018)[1]:

- Pregnancies (number of times pregnant),
- Oral glucose tolerance test - OGTT (two hour plasma glucose concentration after 75g anhydrous glucose in mg/dl),
- Blood Pressure (Diastolic Blood Pressure in mmHg),
- Skin Thickness (Triceps skin fold thickness in mm),
- Insulin (2 h serum insulin in mu U/ml),
- BMI (Body Mass Index in kg/m2),
- Age (years),
- Pedigree Diabetes Function ('function that represents how likely they are to get the disease by extrapolating from their ancestor's history')

Based on the Snapp AI tam the following stages may be performed by the volunteer:

● Create a repository on your GitHub

- Make the dataset imbalance with 10% of the True (1) label
- Write your code on python or jupyter file
- Feature Engineering step
- Feature Selection step
- Train an ML model
- Evaluate and Print the result on test data

2.Pima-Indians-diabetes dataset

Here we describe the dataset using the python and Jupyter Notebook IDE.

Some information about the dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
```

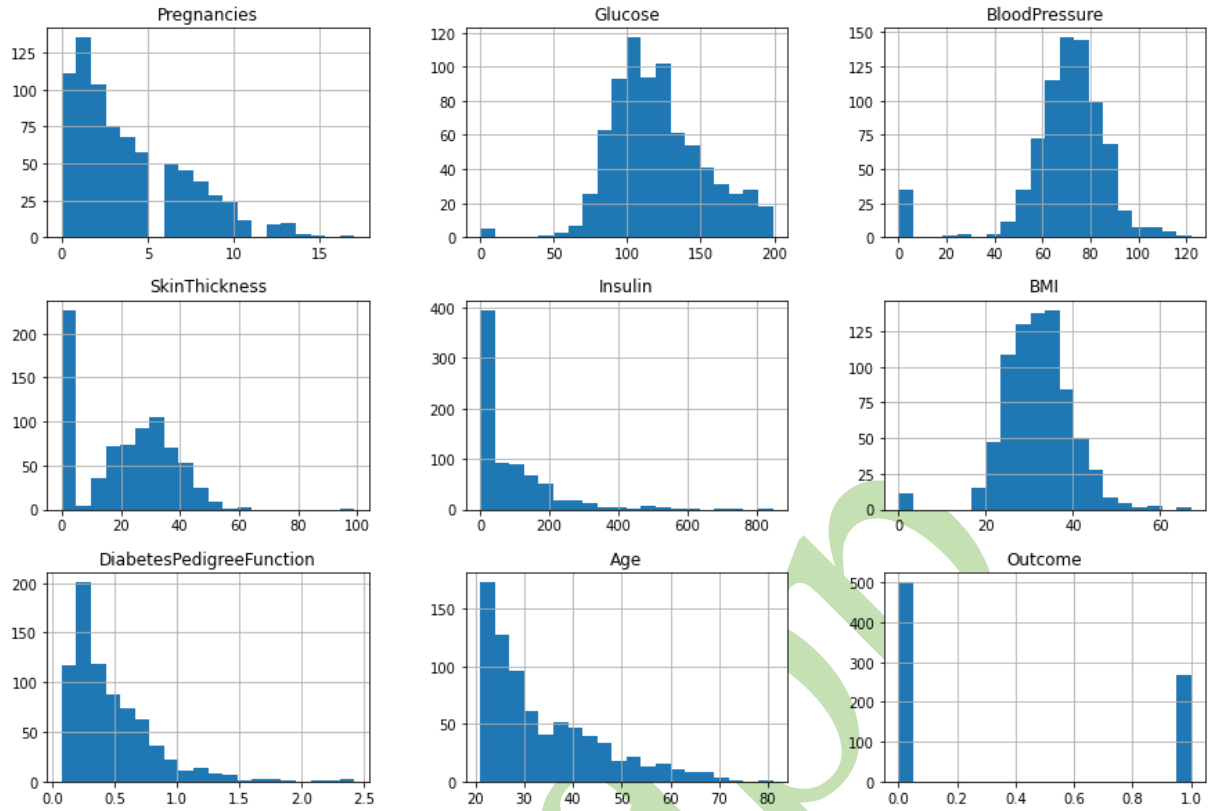

Figure 1: Statistic distribution of data; normal range of features can be selected from this scatterplots.

Gaussian distribution is a favorite types of curves for machine learning projects. As can be seen in the Figure 1 the Pima_Indians_Diabetes there are many features with unreal values in .Based on the studies we should select a nearly normal range of available attributes as following:

- ✓ Normal diastolic blood pressure can't be less than 40mmHg.
- ✓ A blood sugar level less than 140 mg/dL (7.8 mmol/L) is normal. A reading of more than 200 mg/dL (11.1 mmol/L) after two hours indicates diabetes[2].
- ✓ BMI value below 18.5 – you're in the underweight range. between 18.5 and 24.9 – you're in the healthy weight range. between 25 and 29.9[3].
- ✓ The subcutaneous tissue thickness range in males is from 1.65 mm to 14.65 mm, whereas it is from 3.30 mm to 18.20 mm in females [4].

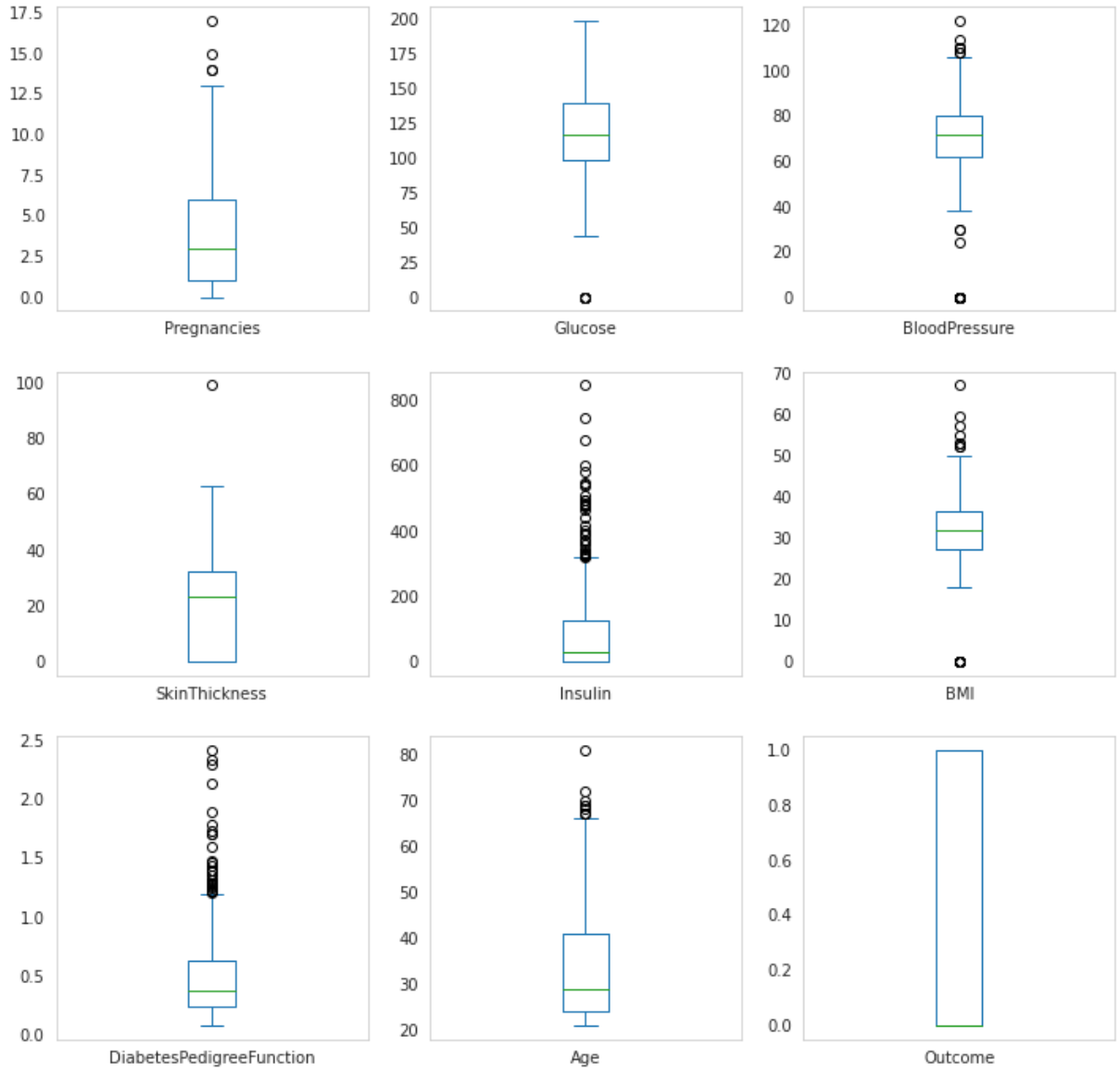

Figure 2: Box plot of data; outliers can be seen above.

Because of imbalance in the data, we will remove abnormal data based on the outliers (Figure 2) from the positive cases. One of the issues about data is heavy tail features.

A description of data percentiles and quarters are as following:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin
count	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479
std	3.369578	31.972618	19.355807	15.952218	115.244002
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000

75%	6.000000	140.250000	80.000000	32.000000	127.250000
max	17.000000	199.000000	122.000000	99.000000	846.000000

	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000
mean	31.992578	0.471876	33.240885	0.348958
std	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.078000	21.000000	0.000000
25%	27.300000	0.243750	24.000000	0.000000
50%	32.000000	0.372500	29.000000	0.000000
75%	36.600000	0.626250	41.000000	1.000000
max	67.100000	2.420000	81.000000	1.000000

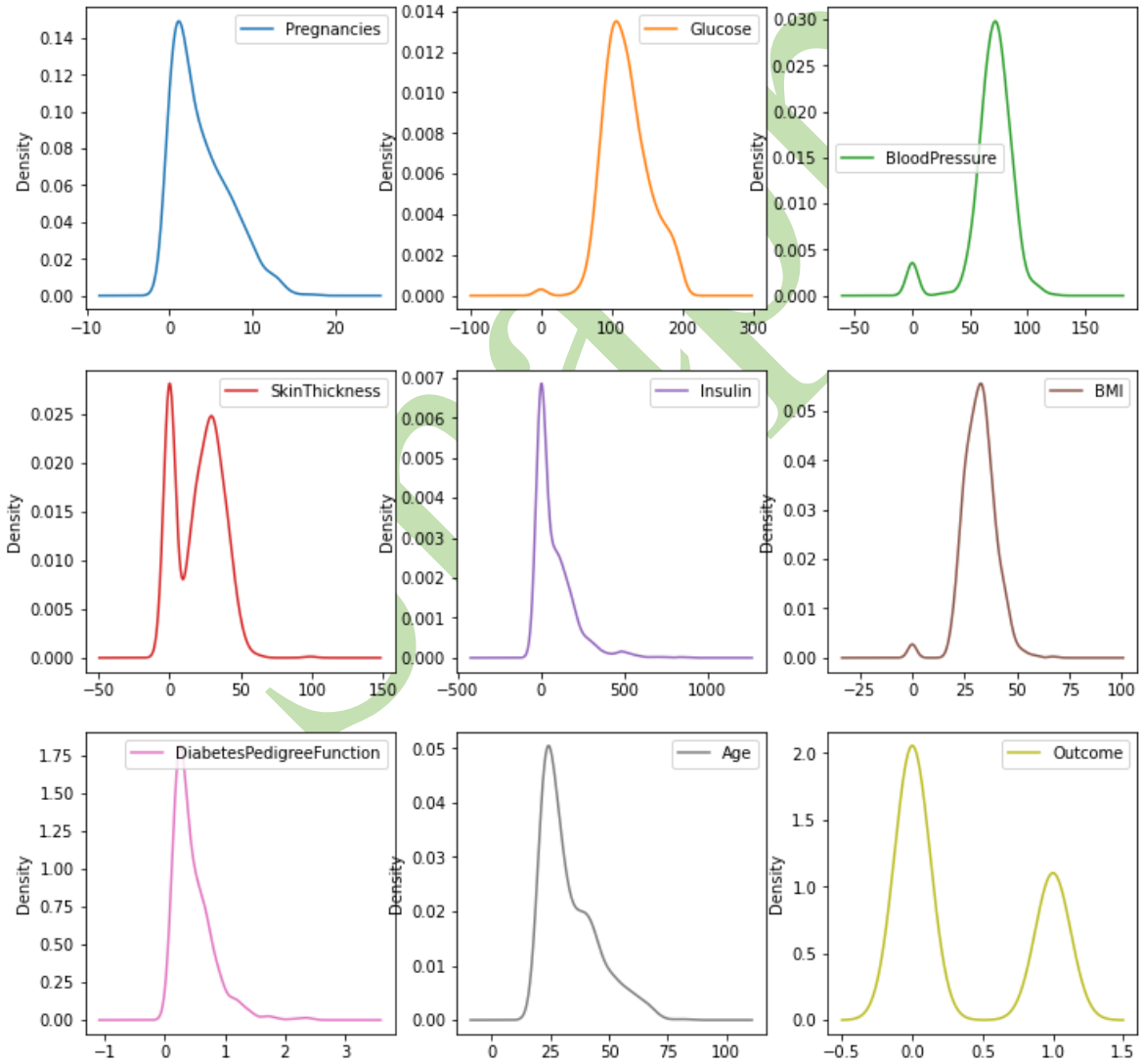

Figure 3: distribution of data of data.

1.1. Correlation Matrix and the linear data representation

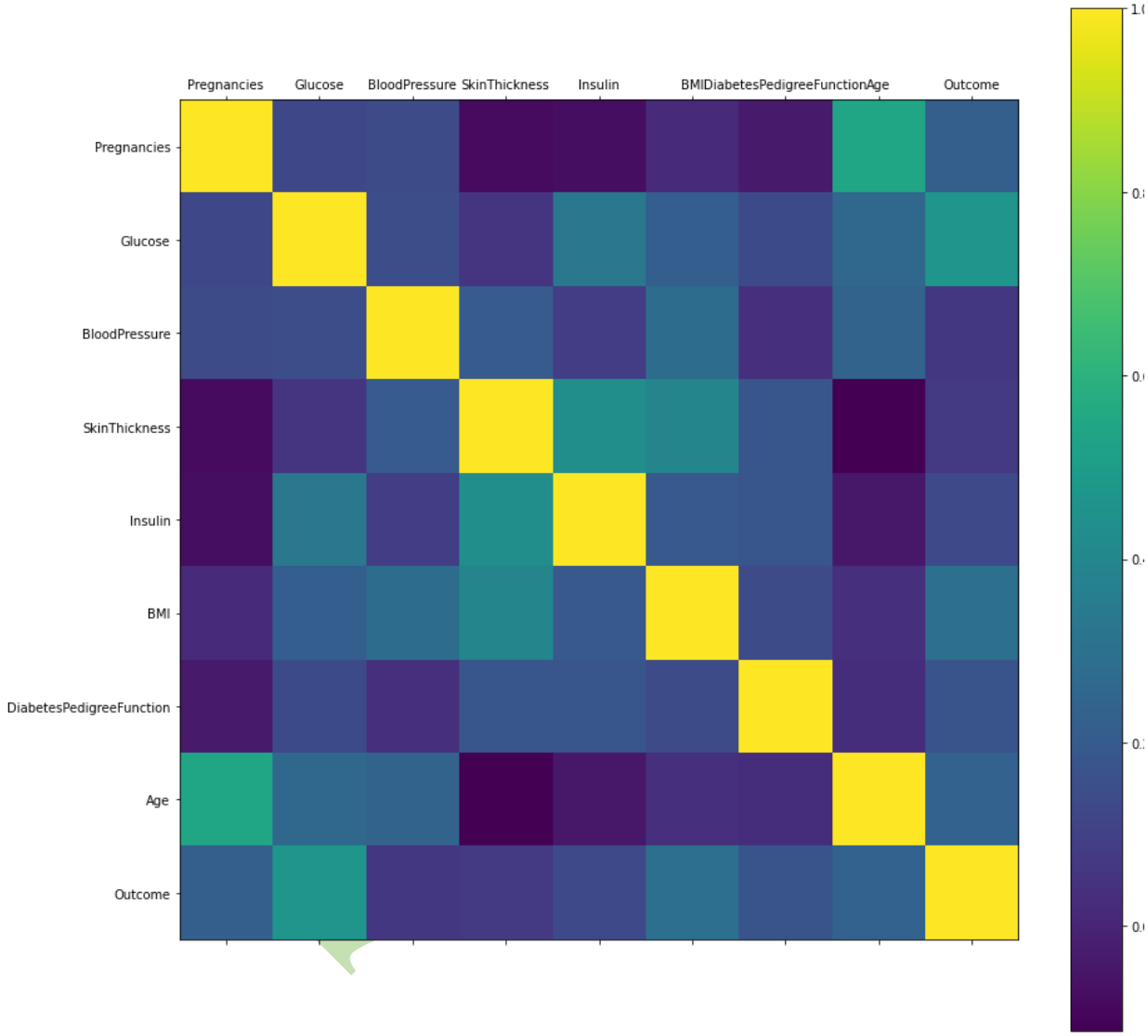

Figure 4: Correlation of attributes

Finding the Features which have the most Linear correlation with outcome sorted in non-ascending order.

```
Outcome                   1.000000
Glucose                   0.466581
BMI                       0.292695
Age                       0.238356
```

```
Pregnancies                 0.221898
DiabetesPedigreeFunction    0.173844
Insulin                     0.130548
SkinThickness               0.074752
BloodPressure               0.065068
Name: Outcome, dtype: float64
```

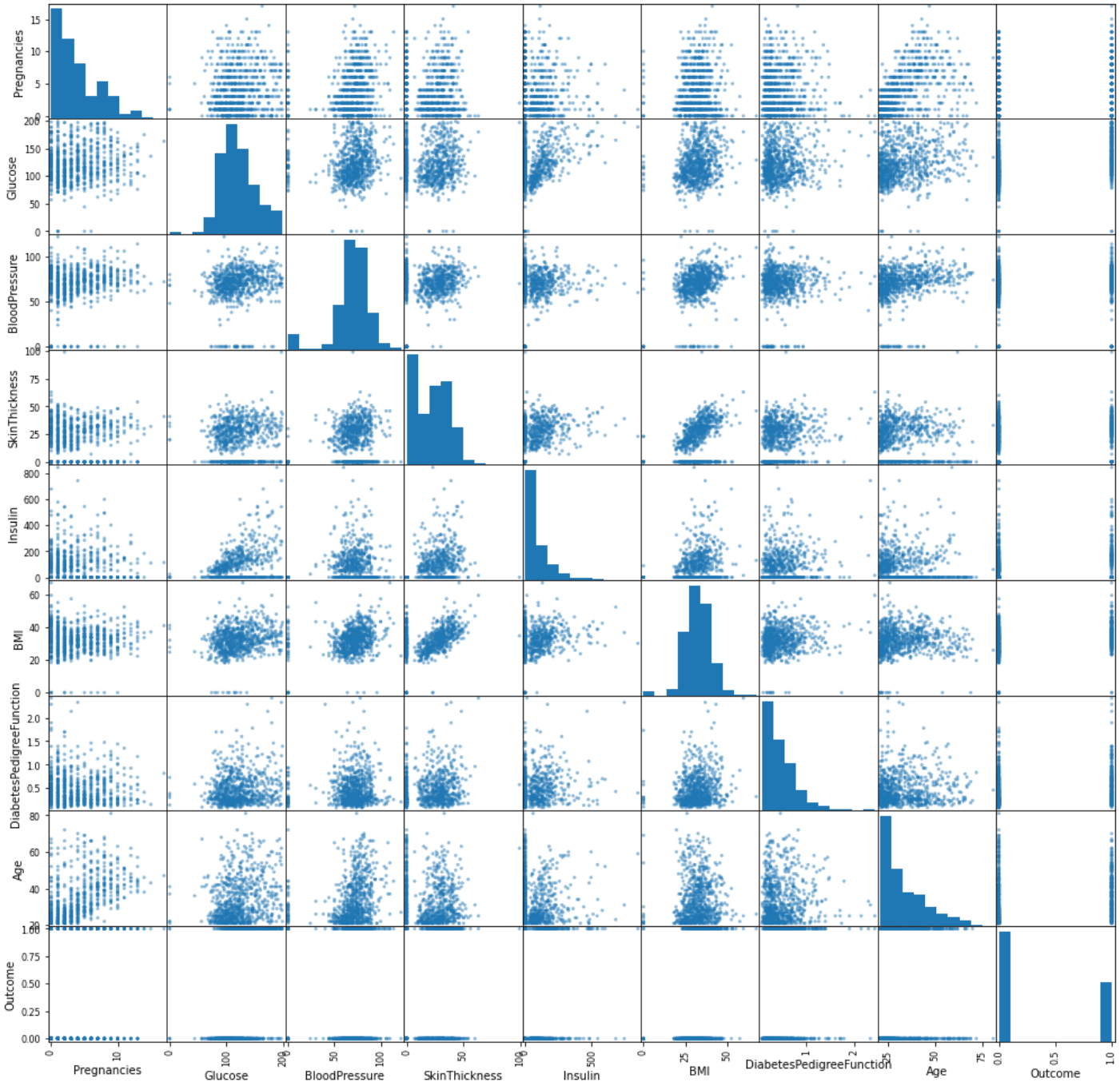

Figure 5: Displaying the linear relations among the attributes.

Visual inspection of data to find non-linear

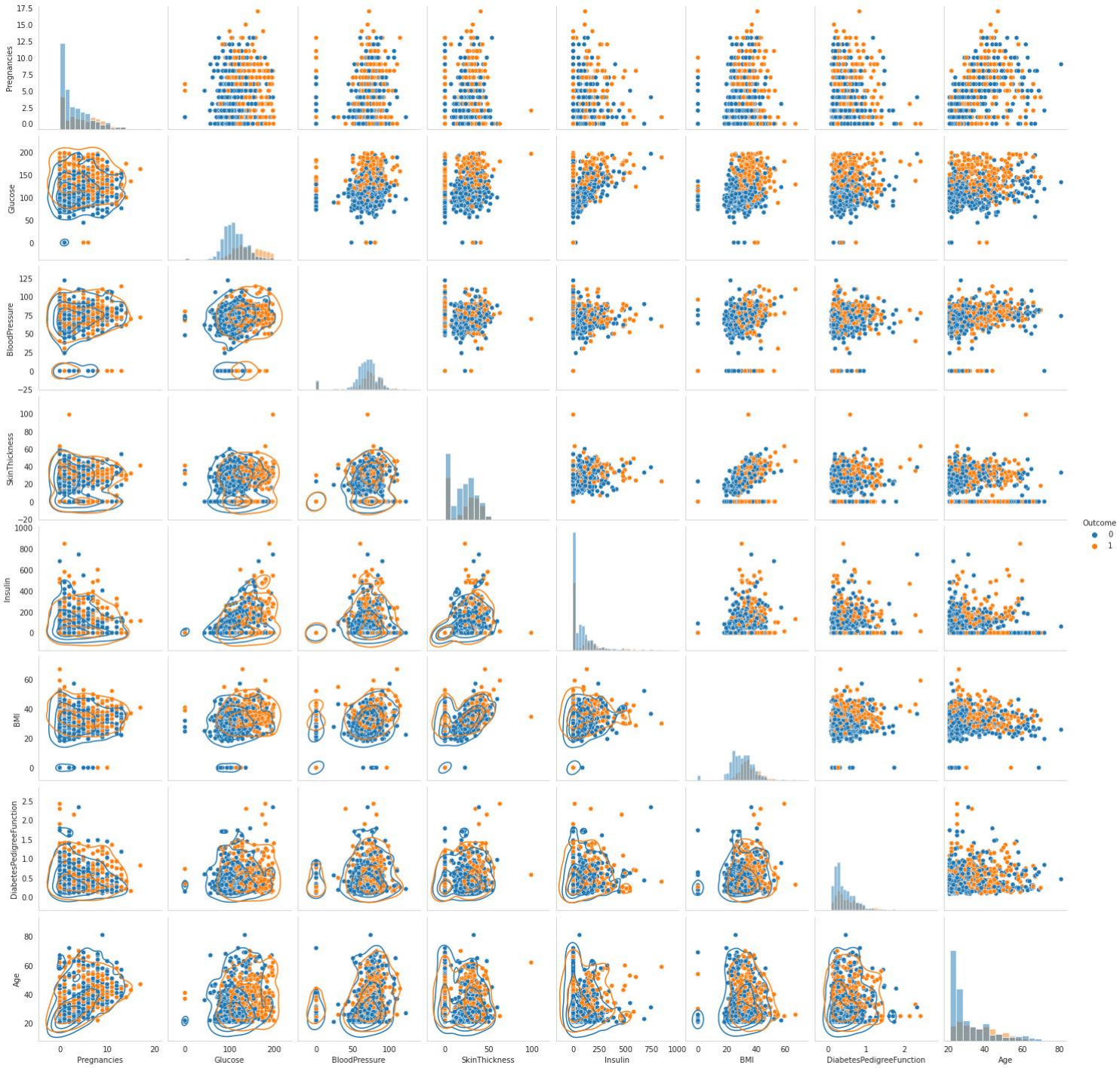

Figure 6: Displaying the linear relations among the attributes.

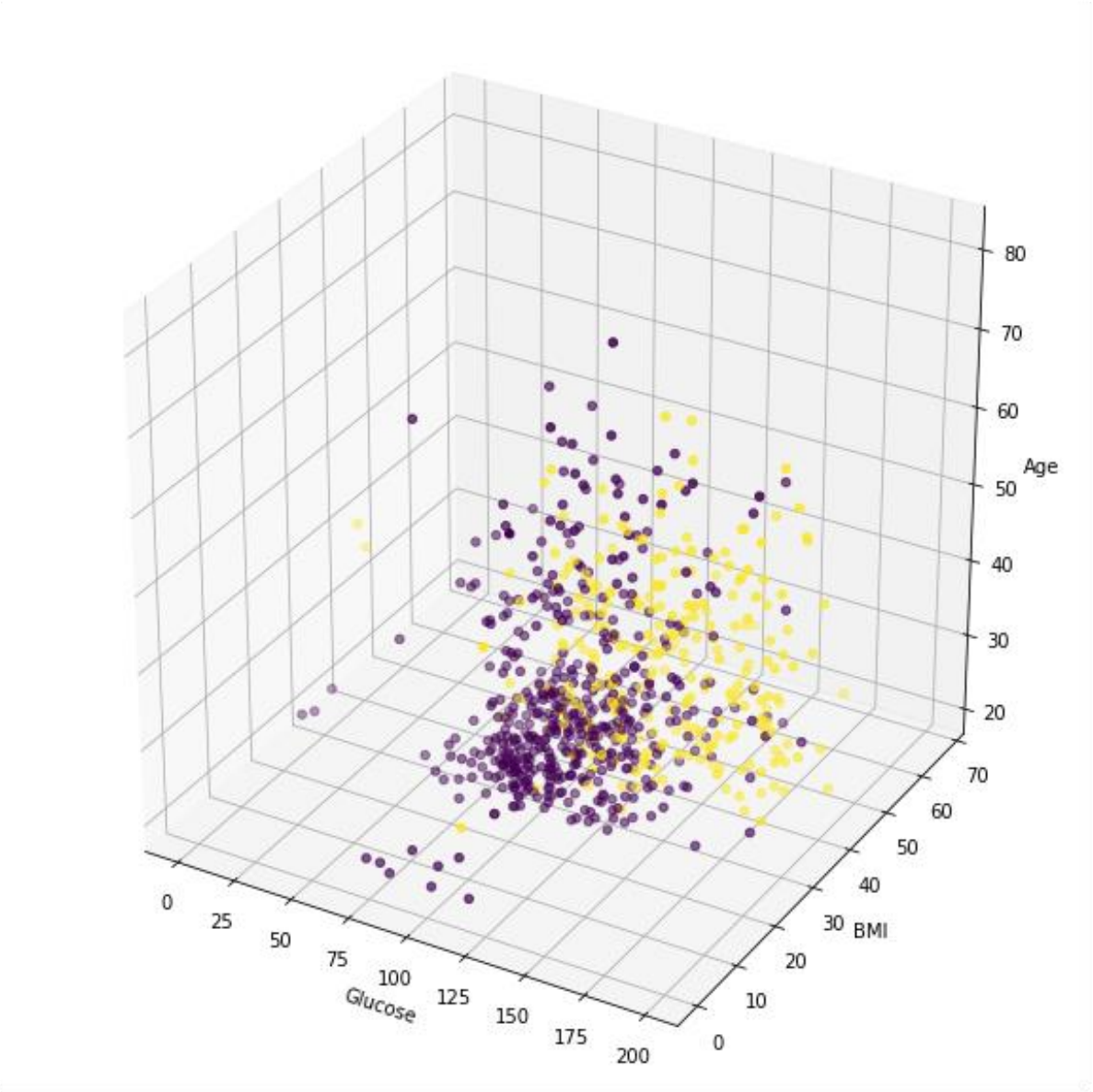

Figure 7: Displaying the "Outcome" based on the three most correlated features.

1.2. Correlation of features

There are no strong correlation between the features.

Table 1 : Correlation matrix for Pima Indians Diabetes dataset

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.195019	0.196291	0.095706	0.075092	-0.017551	0.019485	0.681020	0.261943
Glucose	0.195019	1.000000	0.198174	0.201216	0.579135	0.221125	0.137692	0.341183	0.520044
BloodPressure	0.196291	0.198174	1.000000	0.254755	0.082730	0.365877	0.007069	0.306786	0.211114
SkinThickness	0.095706	0.201216	0.254755	1.000000	0.182870	0.661620	0.161561	0.166117	0.250582
Insulin	0.075092	0.579135	0.082730	0.182870	1.000000	0.234550	0.138277	0.214590	0.302061
BMI	-0.017551	0.221125	0.365877	0.661620	0.234550	1.000000	0.151824	0.070146	0.259661
DiabetesPedigreeFunction	0.019485	0.137692	0.007069	0.161561	0.138277	0.151824	1.000000	0.085811	0.197219
Age	0.681020	0.341183	0.306786	0.166117	0.214590	0.070146	0.085811	1.000000	0.352717
Outcome	0.261943	0.520044	0.211114	0.250582	0.302061	0.259661	0.197219	0.352717	1.000000

1.3. Removing NAN values and replacing unreal values by median

We removed data with unreal from ‘Outcome’= positive and replaced these values with median for ‘Outcome’= negative.

With these new data correlation with ‘Outcome’ column increased as following:

```
Outcome                    1.000000
Glucose                    0.481292
Insulin                    0.375637
SkinThickness              0.356019
BMI                        0.281063
DiabetesPedigreeFunction   0.227562
Age                        0.168093
Pregnancies                0.152744
BloodPressure              0.124123
```

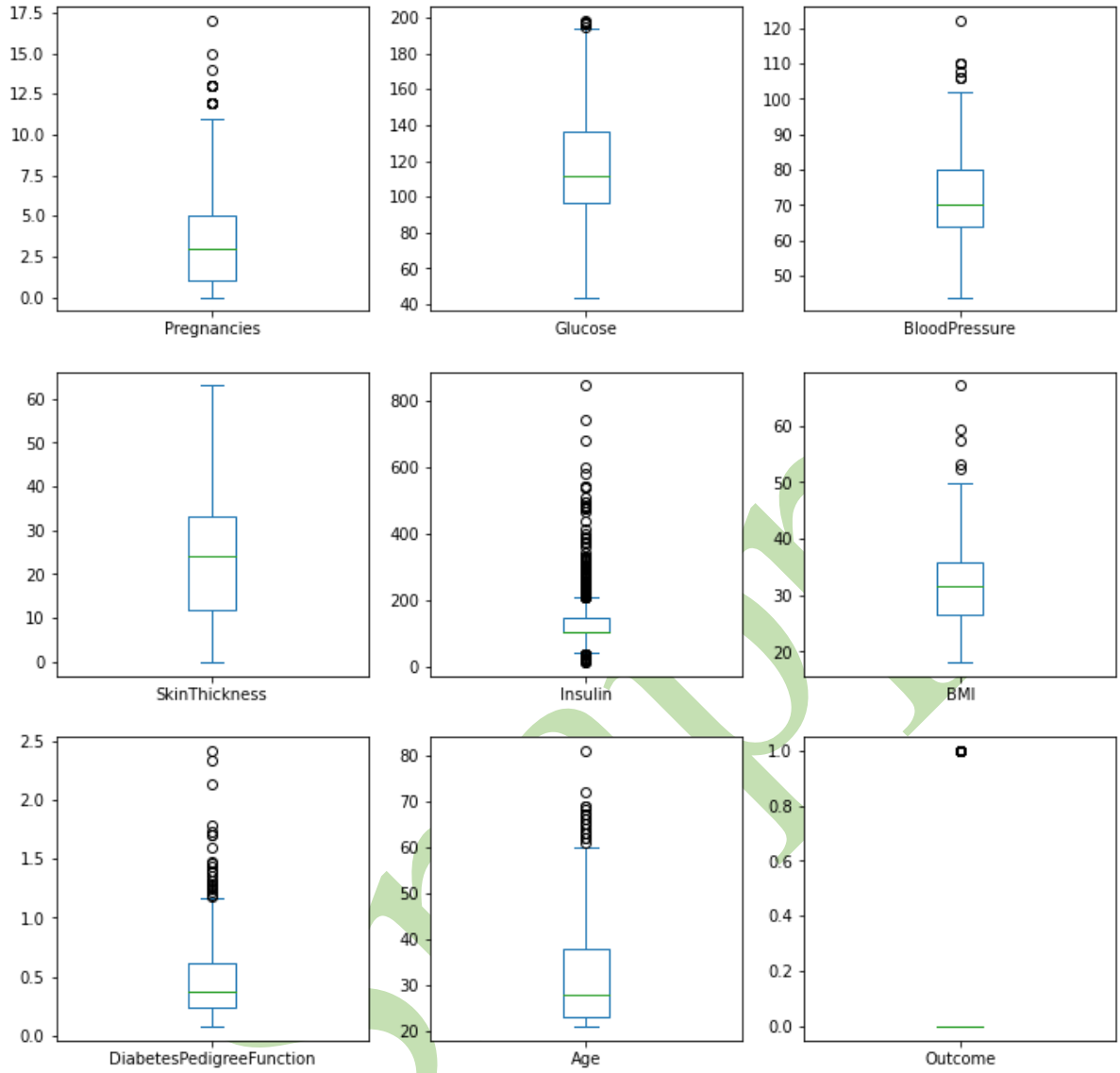

Figure 8: New Box plot of data; some unreal values were removed.

1.4. Making the dataset imbalance with 10% of the True (1) label

Based on the task conditions the 10% imbalance should be considered to build new Pima Indians Diabetes Dataset. We want to have the data as much as possible; then we cleaned the test dataset and t the

Raw dataset

```
Real number of negative and positive cases:
0    500
1    268
Name: Outcome, dtype: int64
the maximum number of label1 for new dataset  with 10%
imbalance: 55
test_size_10percent to randomly select from label1
0.20522388059701493
concatenated 10% of positives and 90% of negatives:
```

Imbalance dataset:

```
pima_imbalanced['Outcome'].value_counts():
0    500
1     55
Name: Outcome, dtype: int64
<class 'pandas.core.frame.DataFrame'>
Int64Index: 555 entries, 1 to 8
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               555 non-null    int64
 1   Glucose                   555 non-null    int64
 2   BloodPressure             555 non-null    int64
 3   SkinThickness             555 non-null    int64
 4   Insulin                   555 non-null    float64
 5   BMI                       555 non-null    float64
 6   DiabetesPedigreeFunction  555 non-null    float64
 7   Age                       555 non-null    int64
 8   Outcome                   555 non-null    int64
dtypes: float64(3), int64(6)
memory usage: 43.4 KB
```

3. Statistical tests

Logic Regression

```
['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
'BMI', 'DiabetesPedigreeFunction', 'Age']
Optimization terminated successfully.
         Current function value: 0.260815
         Iterations 7
                           Logit Regression Results
==============================================================================
=
Dep. Variable:                 Outcome   No. Observations:
555
Model:                           Logit   Df Residuals:
547
Method:                            MLE   Df Model:
7
Date:                Thu, 05 May 2022   Pseudo R-squ.:
0.1928
Time:                        10:22:08   Log-Likelihood:                -
144.75
converged:                       True   LL-Null:                       -
179.32
Covariance Type:            nonrobust   LLR p-value:                 2.209e-
12
==============================================================================
===============
                                 coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------------
---------------
Pregnancies                    0.1281      0.058      2.226      0.026
0.015       0.241
Glucose                        0.0261      0.006      4.196      0.000
0.014       0.038
BloodPressure                 -0.0805      0.014     -5.636      0.000      -
0.109      -0.053
SkinThickness                  0.0646      0.015      4.188      0.000
0.034       0.095
Insulin                        0.0023      0.002      1.514      0.130      -
0.001       0.005
BMI                           -0.0712      0.030     -2.343      0.019      -
0.131      -0.012
DiabetesPedigreeFunction       0.0335      0.498      0.067      0.946      -
0.943       1.010
Age                            0.0006      0.018      0.032      0.975      -
0.034       0.035
==============================================================================
===============
```

1
2
3
4
5
6
7

```
8
9
```

Based on the logic regression only three parameters have p-value less than .05:

```
columns=["Pregnancies", "Glucose","BloodPressure"]
```

4. Feature Selection

Three major buckets of feature selection methods as follow [52]:

- **Filter based methods**: Filtering some features based on the specified metrics (They pick up the intrinsic properties or relevance of the features) for example Mutual Information (MI), Pearson correlation, and chi-square.
- **Wrapper based methods**: Measuring the "usefulness" of features based on the estimator performance. They select a set of features by considering a search problem, for example: (Forward Selection, Backward Elimination, and Recursive Feature Elimination). Any combination of the search strategy and modeling algorithm can be used as a wrapper. These are computationally more expensive compared to filter methods due to the repeated learning steps and cross-validation.
- **Embedded methods**: Using algorithms that have built-in feature selection methods; for instance, L1 (Lasso) regularization and RF.

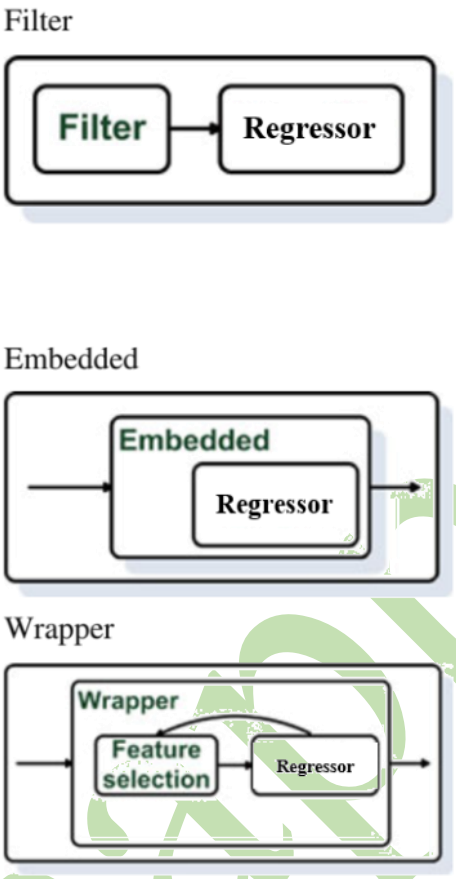

Figure 9: Overall feature selection approaches[52].

4.1. Recursive feature elimination with cross-validation

Figure 10 displays Recursive Feature Elimination on imbalanced imputed data. As can be seen by eliminating Features on 3 folds of data accuracy decreased.

Figure 10:Recursive Feature Elimination on imbalanced imputed data .

```
[('Pregnancies', True),
 ('Glucose', True),
 ('BloodPressure', True),
 ('SkinThickness', True),
 ('Insulin', True),
 ('BMI', True),
 ('DiabetesPedigreeFunction', True),
 ('Age', True)]
```

Figure 11Figure 10 displays Recursive Feature Elimination on raw data. As can be seen by eliminating Features on 3 folds of data accuracy is slightly constant (n_features=5).

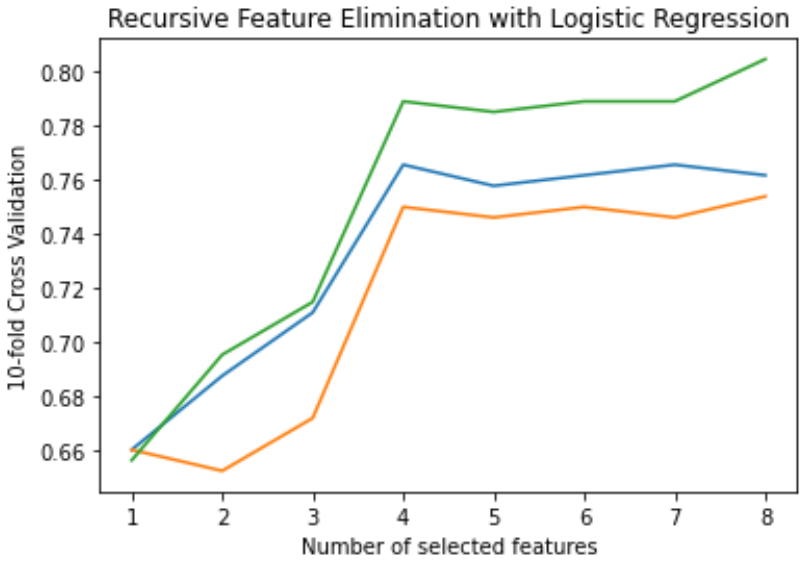

Figure 11:Recursive Feature Elimination on |raw data .

5. Classifiers

Binary classification problems can be solved by a variety of machine learning algorithms ranging from Naive Bayes to deep learning networks. Which solution performs best in terms of runtime and accuracy depends on the data volume (number of samples and features) and data quality (outliers, imbalanced data)

In this project, we will introduce the top 10 most common binary classification algorithms; however will focus on some of them [5]:

1. Naive Bayes
2. Logistic Regression
3. K-Nearest Neighbours
4. Support Vector Machine
5. Decision Tree
6. Bagging Decision Tree (Ensemble Learning I)
7. Boosted Decision Tree (Ensemble Learning II)
8. Random Forest (Ensemble Learning III)
9. Voting Classification (Ensemble Learning IV)
10. Neural Network (Deep Learning)

Standard scaled imputed and imbalanced final dataset:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	-0.774930	-1.051958	-0.454462	0.544113	-0.266996	-0.705644	-0.302229	-0.069753
1	-0.774930	-0.909176	-0.454462	0.140483	-0.361080	-0.477641	-0.915139	-0.920588
2	0.494697	0.054605	0.236806	-1.406768	-0.266996	-0.857645	-0.801884	-0.154837
3	2.081731	0.018909	-0.108828	-1.406768	-0.266996	0.616771	-1.025063	-0.239920
4	0.177290	-0.159569	1.792160	-1.406768	-0.266996	0.966374	-0.835194	-0.154837
...	...	...	...	...	...	...	...	...
550	2.399137	0.197387	0.755257	1.082287	0.258769	1.680782	1.143438	1.376666
551	-0.140116	2.089252	1.100892	0.813200	3.845039	0.677571	-0.612015	-0.835504
552	-0.774930	2.910251	0.409623	1.015016	1.354574	0.799173	1.443230	-0.239920
553	-0.140116	1.553819	-0.108828	0.611385	2.229004	0.647171	-0.325547	0.270581
554	-0.457523	2.945946	-0.108828	1.620461	4.608782	-0.112837	-0.945118	1.802083

555 rows × 8 columns

5.1. knn

10-fold cross validation

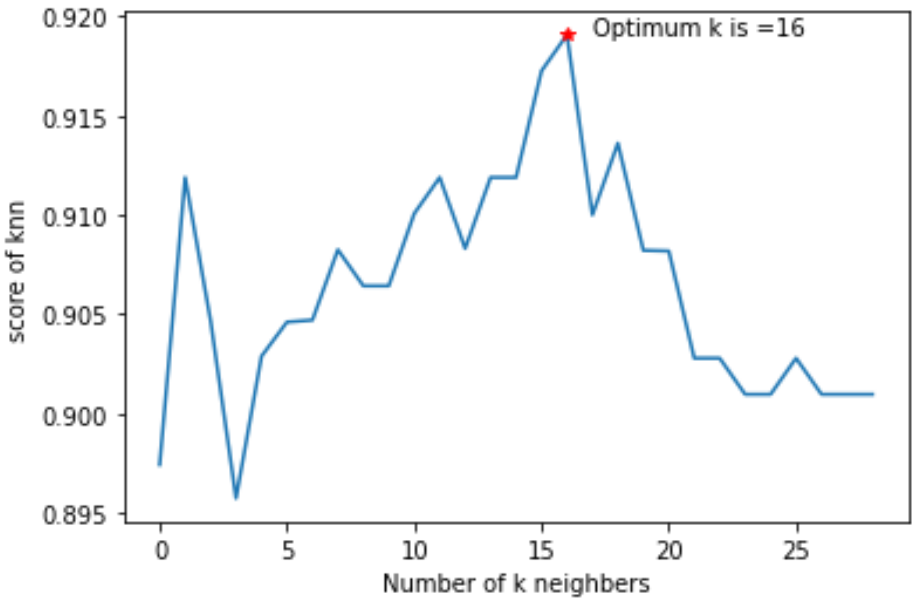

Figure 12: Finding optimum number of neighbours.

Due to 10% imbalance of positive and negative cases in dataset, Type II error is occurred (Figure 13).

Raw data accuracy with 10-fold cross validation:

```
Scors= array([0.75324675, 0.76623377, 0.72727273, 0.67532468,
0.71428571, 0.77922078, 0.76623377, 0.83116883, 0.73684211,
0.77631579])

Scors=mean()

0.752
```

Imputed imbalanced data accuracy with 10-fold cross validation:

```
Scors= array([0.89285714, 0.89285714, 0.92857143, 0.91071429,
0.89285714, 0.94545455, 0.94545455, 0.90909091, 0.94545455,
0.90909091])


Scors.mean():

0.917
```

5.2. Support Vector Machine (SVM)

Raw data accuracy with 10-fold cross validation:

```
Scors=array([0.74025974, 0.74025974, 0.74025974, 0.71428571, 0.7
2727273,
       0.80519481, 0.75324675, 0.80519481, 0.76315789, 0.7894736
8])


Scors=mean()

0.75
```

Imputed imbalanced data accuracy with 10-fold cross validation:

```
Scors= array([0.89285714, 0.89285714, 0.89285714, 0.875     ,
0.89285714, 0.90909091, 0.92727273, 0.90909091, 0.90909091,
0.92727273])


Scors=mean()

0.90
```

5.3. Logistic Regression

Raw data accuracy with 10-fiold cross validation:

```
Scors= array([0.74025974, 0.77922078, 0.80519481, 0.71428571,
0.74025974, 0.76623377, 0.81818182, 0.80519481, 0.76315789,
0.82894737])


Scors=mean()

0.77
```

Imputed imbalanced data accuracy with 10-fiold cross validation:

```
Scors=array([0.89285714, 0.91071429, 0.89285714, 0.875 ,
0.91071429, 0.94545455, 0.92727273, 0.90909091, 0.94545455,
0.92727273])


Scors.mean():

0.9136688311688312
```

5.4. Naïve Bayes

Raw data accuracy with 10-fiold cross validation:

```
Scors= array([0.62337662, 0.58441558, 0.7012987 , 0.48051948,
0.61038961, 0.61038961, 0.5974026 , 0.62337662, 0.55263158,
0.52631579])

Scors=mean()
```

```
0.59
```

Imputed imbalanced data accuracy with 10-fiold cross validation:

```
Scors= array([0.875      , 0.875      , 0.82142857, 0.78571429,
0.80357143, 0.85454545, 0.78181818, 0.81818182, 0.8 ,
0.78181818])

Scors.mean():

0.8190
```

5.5. Random Forest Classifier

Raw data accuracy with 10-fiold cross validation:

```
Scors= array([0.7012987 , 0.79220779, 0.76623377, 0.68831169,
0.75324675,0.81818182, 0.79220779, 0.83116883, 0.68421053,
0.81578947])


Scors=mean()

0.76
```

Imputed imbalanced data accuracy with 10-fiold cross validation:

```
Scors= array([0.92857143, 0.92857143, 0.92857143, 0.89285714,
0.91071429,0.94545455, 0.92727273, 0.92727273, 0.90909091,
0.96363636])

Scors.mean():

0.926
```

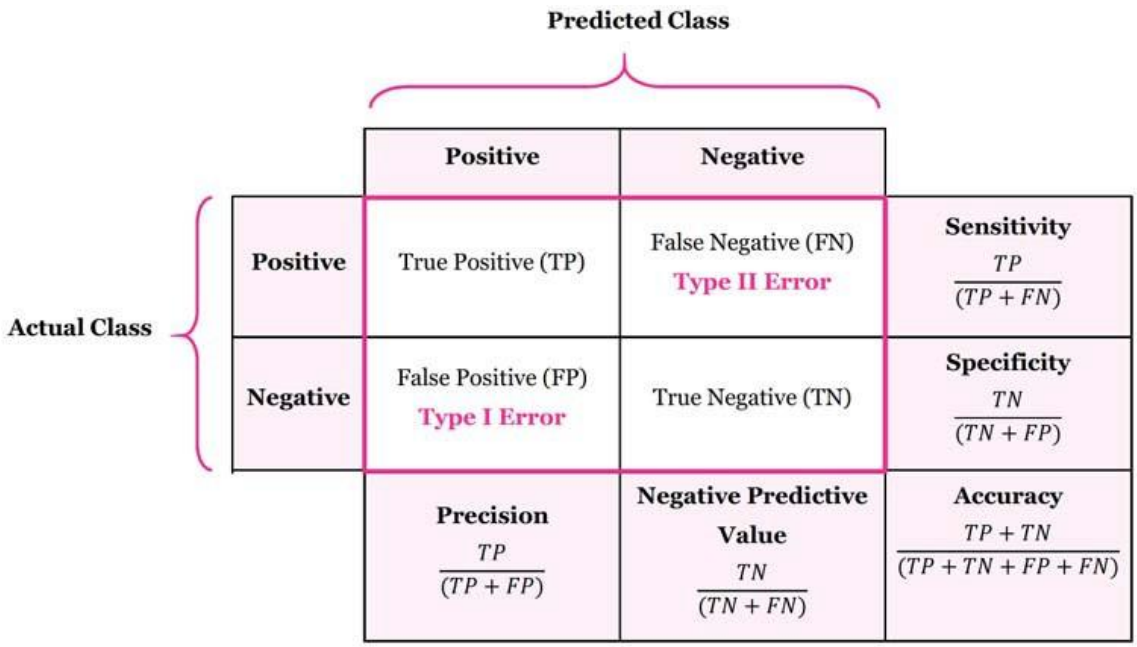

Figure 13: evaluation parameters; Type II error occurrence is due to 10% imbalance of data in dataset .

Based on the results Random forest classifier had a better performance; the second one is logistic regression classifier. However more valuation parameters for this classifiers as explained in Figure 13 is needed for the evaluation.

Final Random Forest Model

We used stratified 4 fold as cross validation method:

precision

```
[0.85, 0.5, 0.8, 0.75]
precision.mean():  0.726
```

recall

```
[0.428, 0.21, 0.28, 0.461]
recall.mean():  0.347
```

accuracy

```
[0.935, 0.899, 0.92, 0.93]
```

```
Accuracy.mean():  0.92
```

Confusion matrix

```
confusion matrix of fold :
  [[124   1]
 [  7   7]]
confusion matrix of fold :
  [[121   4]
 [ 11   3]]
confusion matrix of fold :
  [[124   1]
 [ 10   4]]
confusion matrix of fold :
  [[123   2]
 [  7   6]]
```

6.conclusion

In this task, a classification challenge for Pima Indians Diabetes was analyzed. At first, I created a repository on my GitHub. Then I made the dataset imbalance with 10% of the True (1) label (Positive cases). For simulations, I wrote my code or Jupyter Notebook file using python language. In the Feature Engineering step, some NaN (or zero) values were imputed and I used some statistical tests using logic regression to calculate the p-value. To diminish available uncertainty, I removed outliers from positive cases. The dataset is not too large and a 10 % imbalance also will restrict us more. However, NaN (or zero) values for negative cases were replaced with median values. Using this technique skewness of data also will be diminished. I also used Recursive feature elimination for the Feature Selection step. Some machine learning models using accuracy, precision, and recall were scrutinized in the next step. Based on the results Random Forest classifier had better performance.

References

[1] "method1." https://github.com/npradaschnor/Pima-Indians-Diabetes-Dataset

[2] "." https://www.mayoclinic.org/diseases-conditions/diabetes/diagnosis-treatment/drc-20371451#:~:text=A blood sugar level less,mmol%2FL) indicates prediabetes.

[3] "BMI." https://www.nhs.uk/common-health-questions/lifestyle/what-is-the-body-mass-index-bmi/#:~:text=BMI ranges&text=below 18.5 – you're in,re in the obese range

[4] S. M. Jain, K. Pandey, A. Lahoti, and P. K. Rao, "Evaluation of skin and subcutaneous

tissue thickness at insulin injection sites in Indian, insulin naïve, type-2 diabetic adult population," *Indian J. Endocrinol. Metab.*, vol. 17, no. 5, pp. 864–870, Sep. 2013, doi: 10.4103/2230-8210.117249.

[5] "classifiers." https://medium.com/thinkport/top-10-binary-classification-algorithms-a-beginners-guide-feeacbd7a3e2