

**Senior-AI-Software-Engineer for Snapp**

### **Task: Pima\_Indians\_Diabetes**

#### [**Abbas Biniaz**](https://colab.research.google.com/drive/abbassbiniaz@gmail.com)

May,2022

# 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

# 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)

|  |
| --- |
| E:\projects\snapp\my_task\download (2).png |
| 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].

|  |
| --- |
| E:\projects\snapp\my_task\download (7).png |
| 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

|  |
| --- |
| E:\projects\snapp\other\index3.png |
| Figure 3: distribution of data of data. |

## Correlation Matrix and the linear data representation

|  |
| --- |
| E:\projects\snapp\my_task\download (3).png |
| 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

|  |
| --- |
| E:\projects\snapp\my_task\download (4).png |
| Figure 5: Displaying the linear relations among the attributes. |

Visual inspection of data to find non-linear

|  |
| --- |
| E:\projects\snapp\my_task\download (9).png |
| Figure 6: Displaying the linear relations among the attributes. |

|  |
| --- |
| E:\projects\snapp\my_task\index.png |
| Figure 7: Displaying the “Outcome” based on the three most correlated features. |

## Correlation of features

There are no strong correlation between the features.

|  |
| --- |
| Table 1 : Correlation matrix for Pima Indians Diabetes dataset |
|  |

## 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

|  |
| --- |
| E:\projects\snapp\my_task\index2.png |
| Figure 8: New Box plot of data; some unreal values were removed. |

## 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

# 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"]

# 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 11: Overall feature selection approaches[52]. |

## Recursive feature elimination with cross-validation

Figure 12 displays Recursive Feature Elimination on imbalanced imputed data. As can be seen by eliminating Features on 3 folds of data accuracy decreased.

|  |
| --- |
| E:\projects\snapp\other\index6.png |
| Figure 12:Recursive Feature Elimination on imbalanced imputed data . |

[('Pregnancies', True),

('Glucose', True),

('BloodPressure', True),

('SkinThickness', True),

('Insulin', True),

('BMI', True),

('DiabetesPedigreeFunction', True),

('Age', True)]

Figure 13 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).

|  |
| --- |
| E:\projects\snapp\other\index7.png |
| Figure 13:Recursive Feature Elimination on |raw data . |

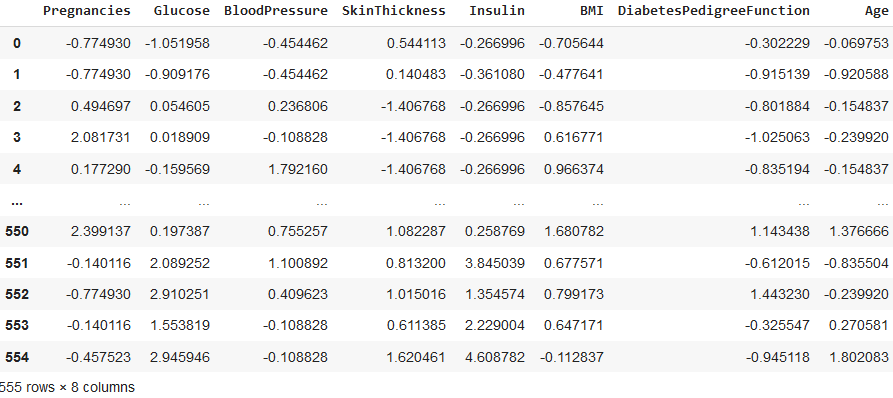
# Classifiers

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](https://scikit-learn.org/stable/modules/naive_bayes.html)
2. [Logistic Regression](https://towardsdatascience.com/understanding-logistic-regression-9b02c2aec102)
3. [K-Nearest Neighbours](https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761)
4. [Support Vector Machine](https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47)
5. [Decision Tree](https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052)
6. [Bagging Decision Tree (Ensemble Learning I)](https://medium.com/ml-research-lab/bagging-ensemble-meta-algorithm-for-reducing-variance-c98fffa5489f)
7. [Boosted Decision Tree (Ensemble Learning II)](https://medium.com/ml-research-lab/boosting-ensemble-meta-algorithm-for-reducing-bias-5b8bfdce281)
8. [Random Forest (Ensemble Learning III)](https://medium.com/all-things-ai/in-depth-parameter-tuning-for-random-forest-d67bb7e920)
9. [Voting Classification (Ensemble Learning IV)](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html)
10. [Neural Network (Deep Learning)](https://medium.com/botsupply/a-beginners-guide-to-deep-learning-5ee814cf7706)

Standard scaled imputed and imbalanced final dataset:



## knn

**10-fold cross validation**

|  |
| --- |
| E:\projects\snapp\other\index5.png |
| Figure 9: Finding optimum number of neighbours. |

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

Raw data accuracy with 10-fiold 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-fiold 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

## Support Vector Machine (SVM)

Raw data accuracy with 10-fiold cross validation:

Scors=array([0.74025974, 0.74025974, 0.74025974, 0.71428571, 0.72727273,

       0.80519481, 0.75324675, 0.80519481, 0.76315789, 0.78947368])

Scors=mean()

0.75

Imputed imbalanced data accuracy with 10-fiold 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

## 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

## 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

## 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 10: 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 10 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]]

**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