Diabetes Diagnostic Measures

By Aman Negassi

Objective

- We want to determine whether or not a patient has diabetes based on the diagnostic measures. The dataset was retrieved from the National Institute of Diabetes and Digestive and Kidney Diseases. The patient sample is composed of females of Pima Indian heritage with the minimum age being 21 years old.

- We will also examine whether there is a relationship among the independent variables as well as a pattern from any of the independent variables to the Outcome.

- The Outcome variable is the dependent variable and whether the person has diabetes or not.

The Diagnostic Measures

The independent variables are Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, and Age as shown in the statistics summary.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
count	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
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000

The Diagnostic Measures cont

To note, Diabetes Pedigree Function means the likelihood of diabetes based on family History whether hereditary or not and the possibilities.

Age	Outcome
768.000000	768.000000
33.240885	0.348958
11.760232	0.476951
21.000000	0.000000
24.000000	0.000000
29.000000	0.000000
41.000000	1.000000
81.000000	1.000000

Data Wrangling

As shown here, the values for the Outcome are 1 and 0.

It’s practically binary.

```
diabetes['Outcome'].unique()
```

```
array([1, 0])
```

That’s why we will be replacing the values with YES and NO. YES is 1 and NO is 0.

```
diabetes['Outcome'] = diabetes['Outcome'].replace(1, 'YES')
diabetes['Outcome'] = diabetes['Outcome'].replace(0, 'NO')
```

Outcome
1
0
1
0
1

```
<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)
memory usage: 54.1 KB
```

```
<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    object
dtypes: float64(2), int64(6), object(1)
memory usage: 54.1+ KB
```

```
diabetes['Outcome'].unique()
```

```
array(['YES', 'NO'], dtype=object)
```

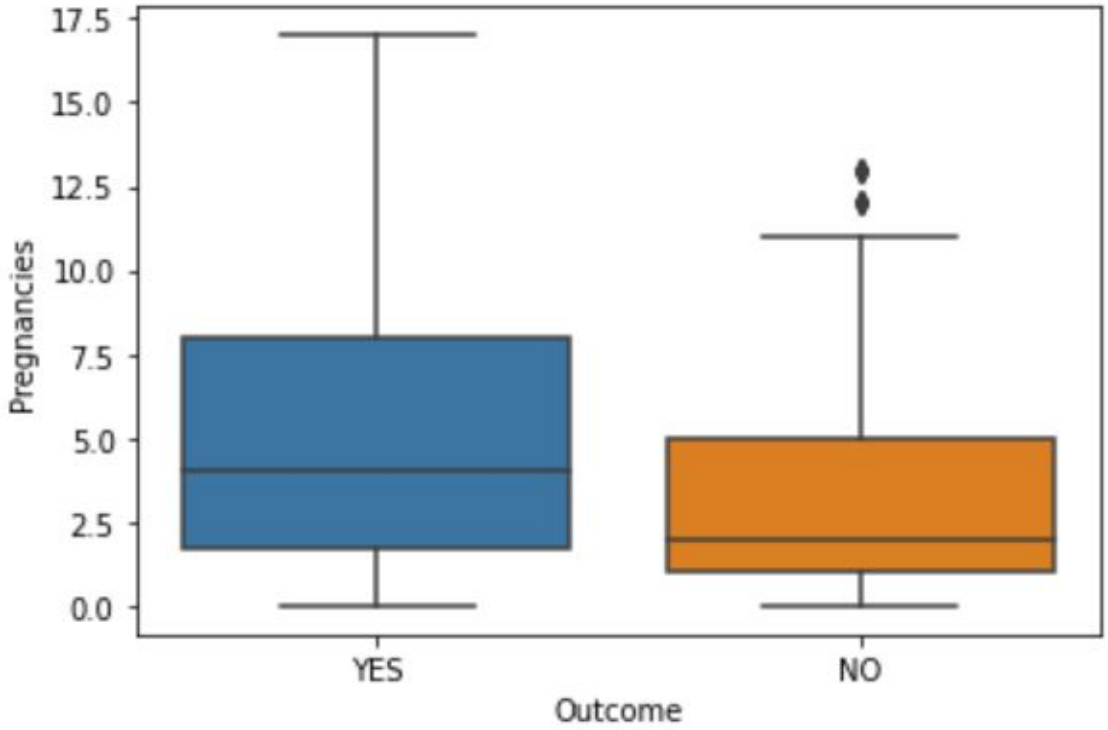
As you can see above, when changing the values on the dependent variable, you end up adding objects to your data types list. We will now improve our readings.

Exploratory Data Analysis

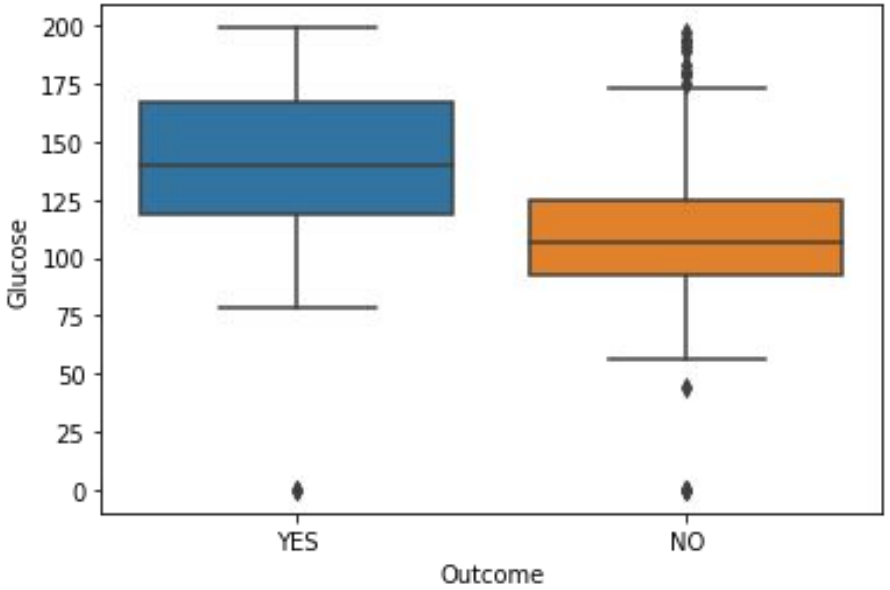
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

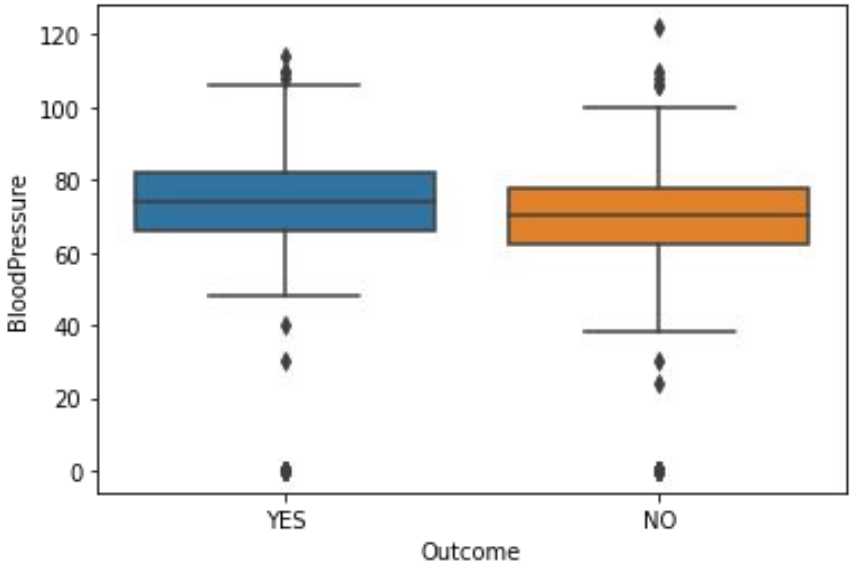
```
diabetes.shape
```

```
(768, 9)
```

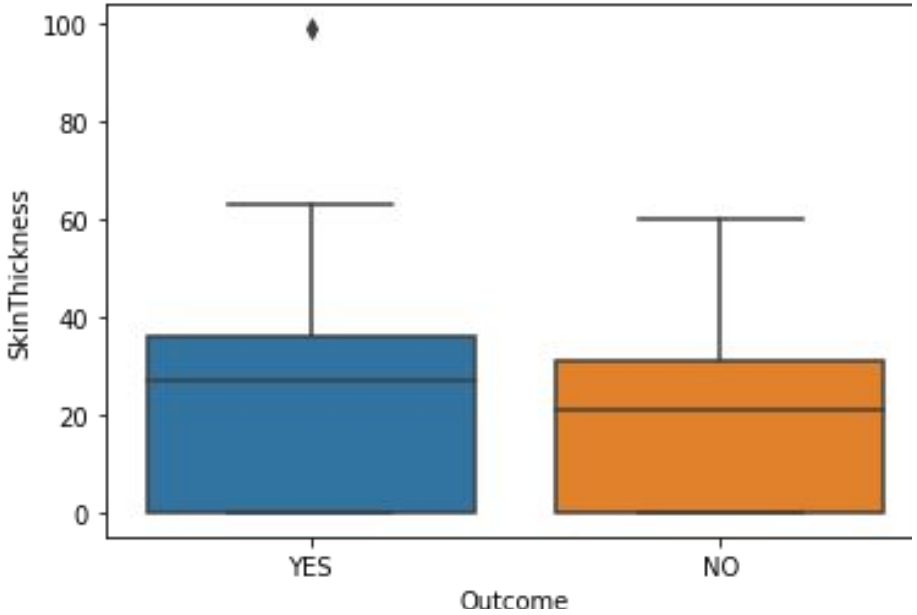


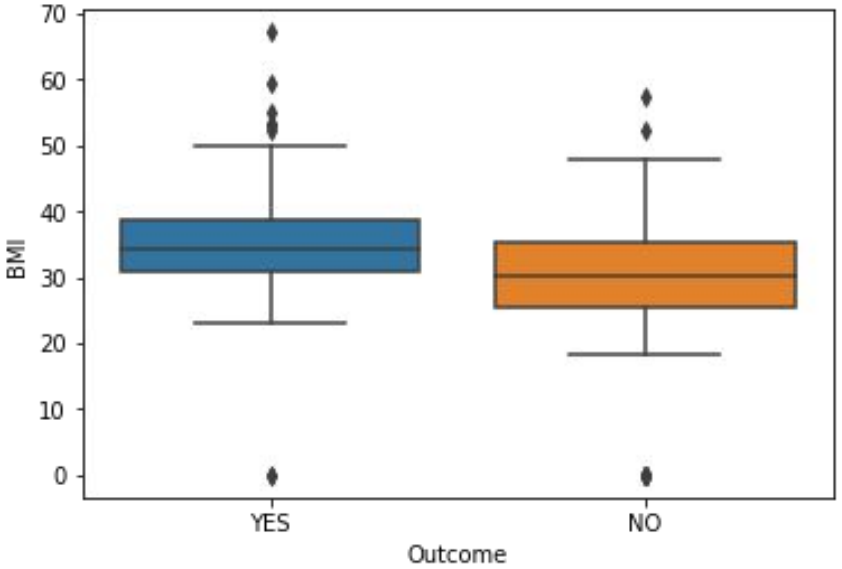
The boxplots are skewed positively as a result of the median being close to the bottom quartile. Although not far apart, patients who contracted diabetes on average more pregnancies than those who did not contract diabetes.





Patients who contracted diabetes on average had higher glucose levels. It seems that blood pressure levels does not seem to make much of a difference in whether a person contracts diabetes.



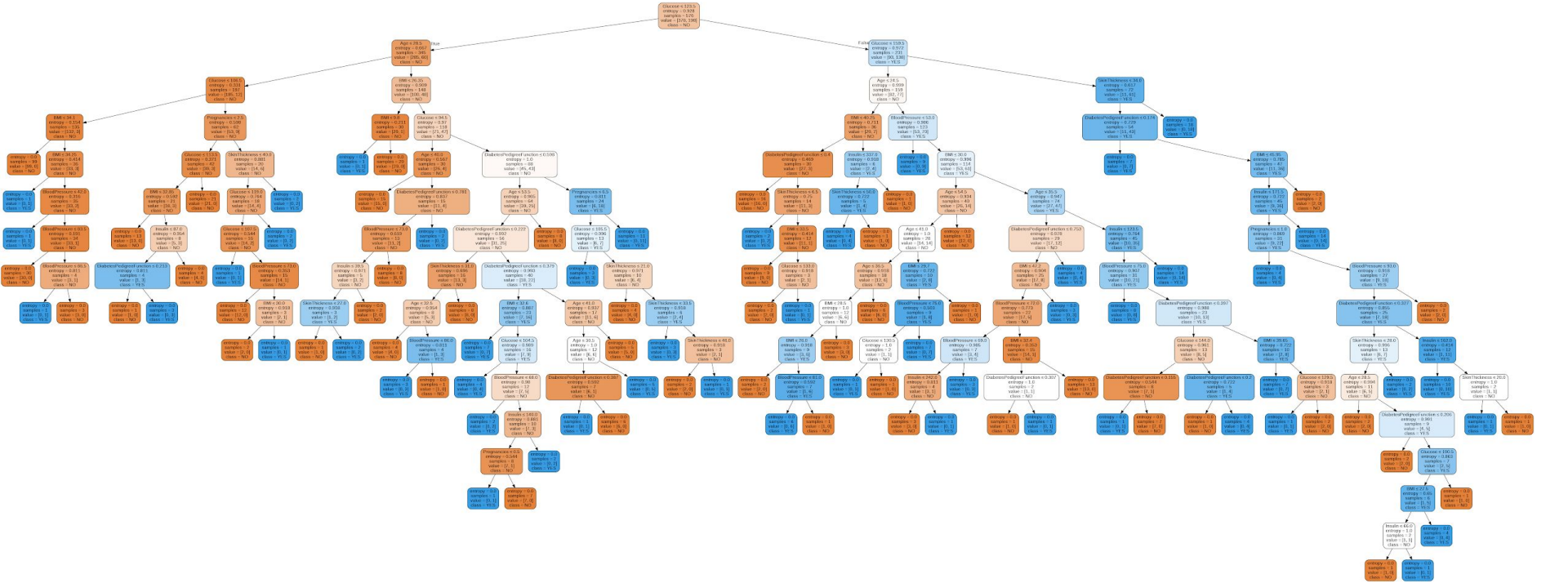


Patients who contracted diabetes had on average a higher BMI and Skin Thickness although barely noticeable.

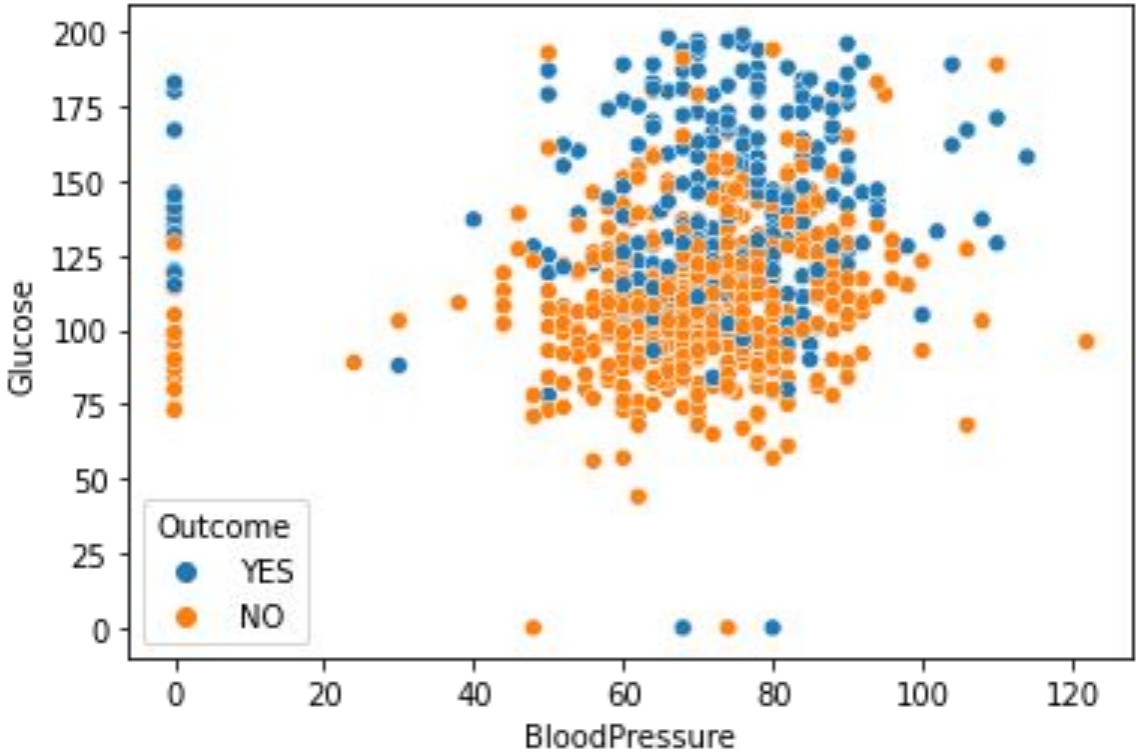
Key Findings

```
Gini impurity  model - max depth 3
Accuracy: 0.6979166666666666
Balanced accuracy: 0.692271662763466
Precision score 0.573170731707317
Recall score 0.7131147540983607
 NO     500
 YES    268
 Name: Outcome, dtype: int64
```

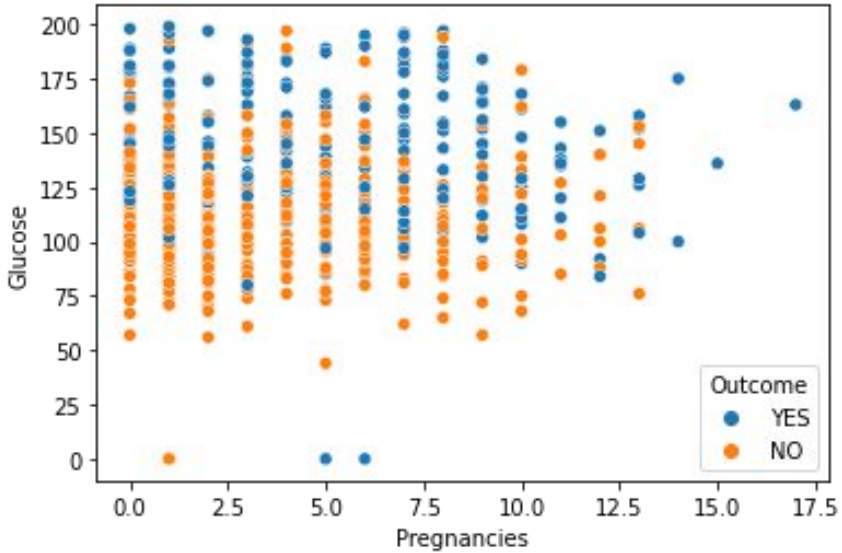
268 out of 768 patients contracted diabetes which is about 35%.

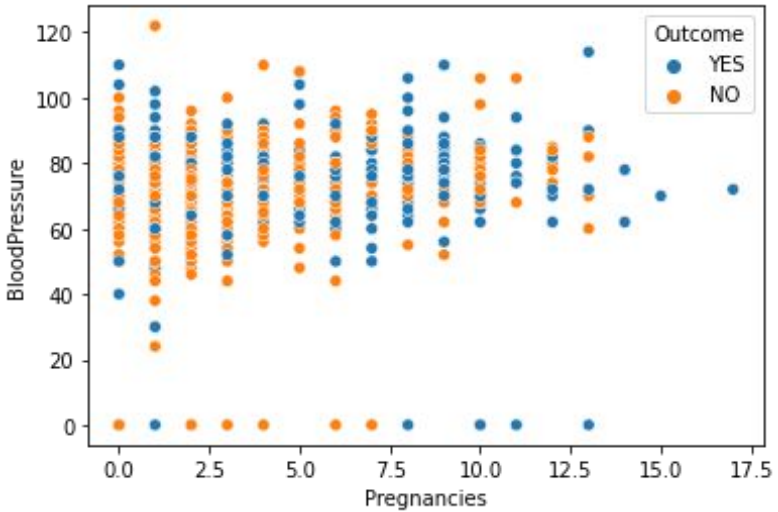


Decision Tree Modeling



Although it is not as strong, there is a direct relationship with the proportionality of the Blood Pressure and Glucose levels.





There was no relationship between Pregnancies and glucose and blood pressure levels which seemed surprising considering the stress a woman goes from being pregnant to having a child.

```
Xlr, Xtestlr, ylr, ytestlr = train_test_split(diabetes[['Insulin','BloodPressure
                                                (diabetes.Outcome == 'YES').values
```

Recommendations and Suggestions

After conducting experiments with the train/test split, it can be said the accuracy score for determining whether a patient has diabetes is higher when using 4 variables. The accuracy score is at its highest when using the variables: Insulin, Blood Pressure, Glucose, and BMI.

```
clf.fit(Xlr, ylr)
print(accuracy_score(clf.predict(Xtestlr), ytestlr))
```

```
0.796875
```