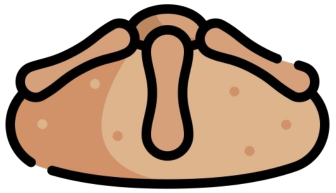
Should you keep eating Pan de Muerto?



Montserrat Garrido | A01208955

Decision Trees and Clustering

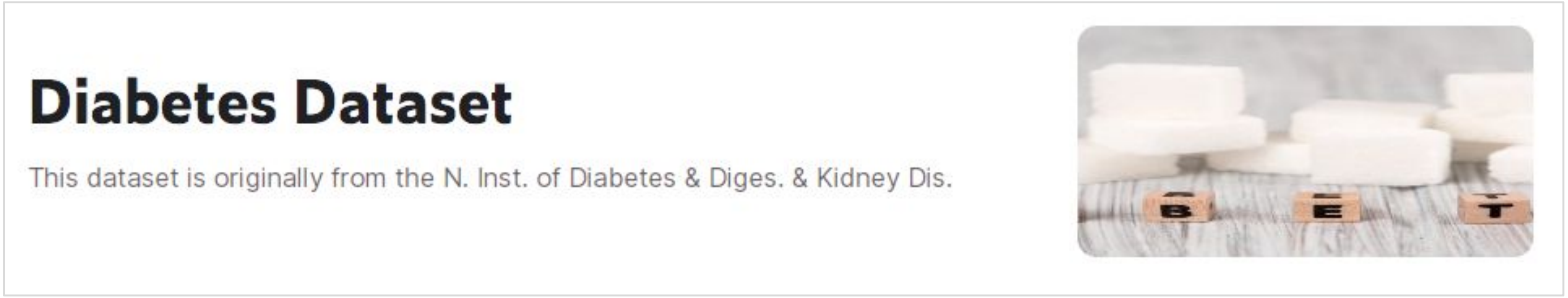
- Mexico occupies the 10th place in obesity worldwide.
- In 2020 there where more than 150k deaths related to diabetes.

I created a program that evaluates some parameters in order to evaluate your possibility of having diabetes in the near future.
The results are based on whether you should keep eating Pan de Muerto, or maybe go see a nutritionist and exercise a bit.

The parameters you will need for the study are:

- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- Blood Pressure: Diastolic blood pressure (mm Hg)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- Age: Age (years)

I used a dataset originally from the National Institute of Diabetes and Digestive and Kidney Diseases, obtained from Kaggle.



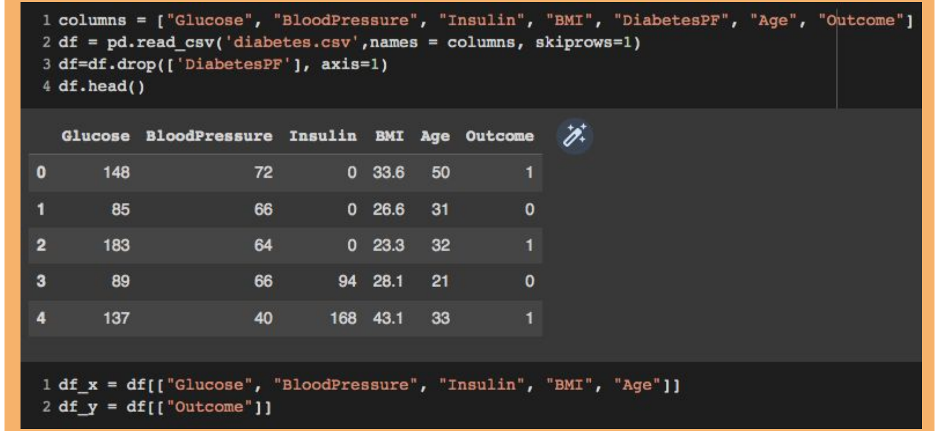
Diabetes Dataset

This dataset is originally from the N. Inst. of Diabetes & Diges. & Kidney Dis.

The objective is to predict, based on diagnostic measurements, whether a patient has diabetes. all patients here are females at least 21 years old of Pima Indian heritage.

1. Decision Tree

On the first half of the study, a Decision Tree was generated with the results available being 0-1 whether a patient has diabetes.



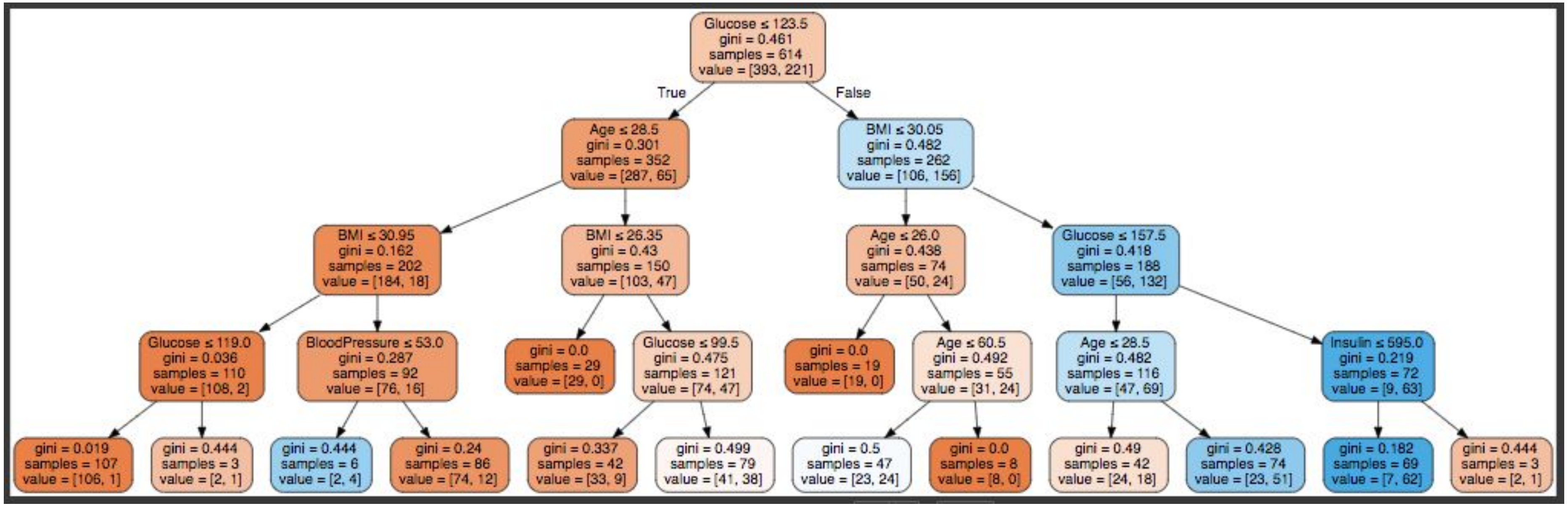
```
columns = ["Glucose", "BloodPressure", "Insulin", "BMI", "DiabetesPF", "Age", "Outcome"]
df = pd.read_csv('diabetes.csv',names = columns, skiprows=1)
df=df.drop(['DiabetesPF'], axis=1)
df.head()
```

```
   Glucose  BloodPressure  Insulin   BMI  Age  Outcome
0      148             72        0  33.6   50        1
1       85             66        0  26.6   31        0
2      183             64        0  23.3   32        1
3       89             66       94  28.1   21        0
4      137             40      168  43.1   33        1
```

```
df_x = df[["Glucose", "BloodPressure", "Insulin", "BMI", "Age"]]
df_y = df[["Outcome"]]
```

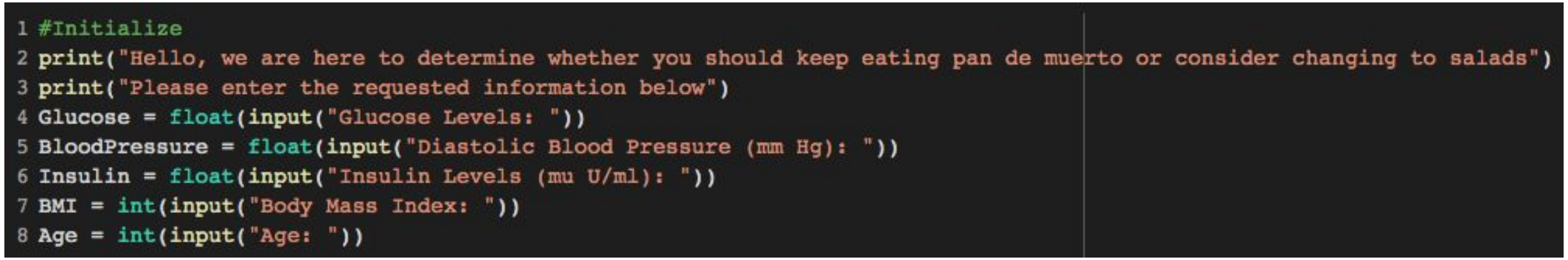
I chose 5 features as my x's and 'Outcome' as my y.

I used DecisionTreeClassifier from sklearn.tree to generate the Tree



This was the resulting Tree using max depth=4

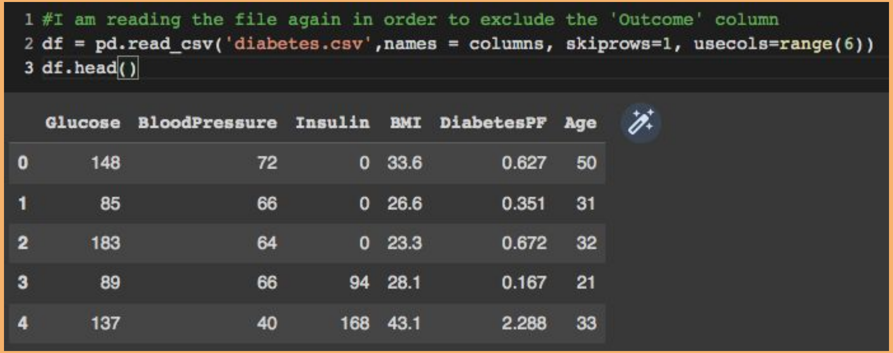
After generating the Tree, there are some initialization questions needed to make a prediction.



```
#Initialize
print("Hello, we are here to determine whether you should keep eating pan de muerto or consider changing to salads")
print("Please enter the requested information below")
Glucose = float(input("Glucose Levels: "))
BloodPressure = float(input("Diastolic Blood Pressure (mm Hg): "))
Insulin = float(input("Insulin Levels (mu U/ml): "))
BMI = int(input("Body Mass Index: "))
Age = int(input("Age: "))
```

Once that information is provided, a prediction will be made resulting in:

- “Your numbers seem fine, you may add Nutella to your Pan de Muerto”
- Or, “You should probably stop eating Pan de Muerto and see a doctor”



```
#I am reading the file again in order to exclude the 'Outcome' column
df = pd.read_csv('diabetes.csv',names = columns, skiprows=1, usecols=range(6))
df.head()
```

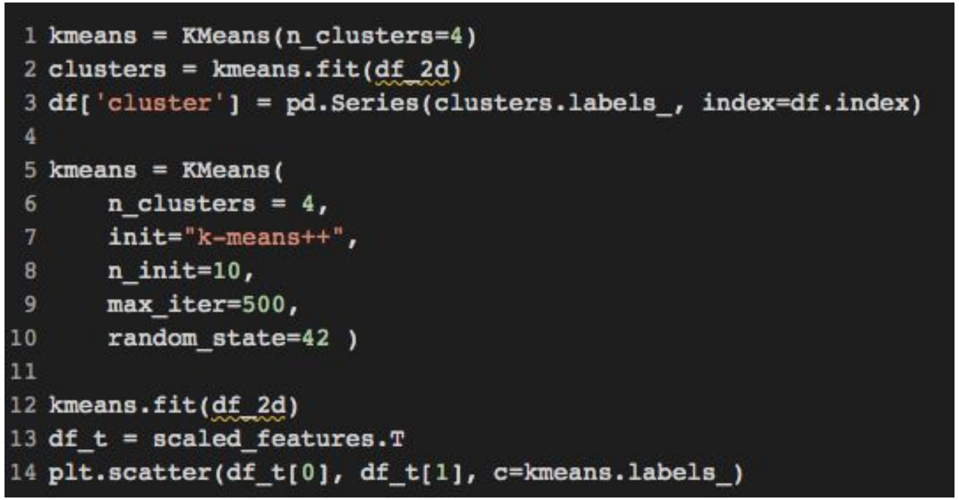
```
   Glucose  BloodPressure  Insulin   BMI  DiabetesPF  Age
0      148             72        0  33.6       0.627   50
1       85             66        0  26.6       0.351   31
2      183             64        0  23.3       0.672   32
3       89             66       94  28.1       0.167   21
4      137             40      168  43.1       2.288   33
```

2. Clustering

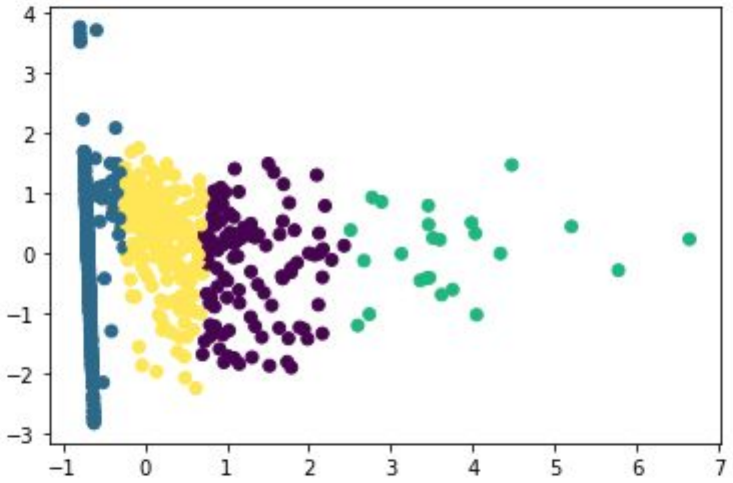
I first ran a Principal Component Analysis and then ran the KMeans clustering method in order to obtain groups suitable for making predictions.

In this case, the column ‘Outcome’ was excluded, instead I used the ‘DiabetesPF’ as my y, because it scores likelihood of diabetes based on family history

After some trial and error I ended up choosing 4 clustering groups, each indicating a different level of 'DiabetesPF', closer to 7 being the higher possibility of having Diabetes.

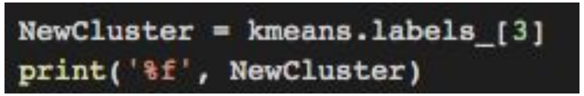


```
kmeans = KMeans(n_clusters=4)
clusters = kmeans.fit(df_2d)
df['cluster'] = pd.Series(clusters.labels_, index=df.index)

kmeans = KMeans(
    n_clusters = 4,
    init="k-means++",
    n_init=10,
    max_iter=500,
    random_state=42 )

kmeans.fit(df_2d)
df_t = scaled_features.T
plt.scatter(df_t[0], df_t[1], c=kmeans.labels_)
```



Finally, using the same inputs as the Decision Tree, a prediction is made based on the cluster you land on, the possible results being:

1. This group represents that your numbers came out almost perfectly, yo may keep eating Pan de Muerto and even add an Abuelita hot chocolate
2. This group represents that your numbers came out so-so, you could keep eating Pan de Muerto but maybe exercise a bit
3. This group represents that your numbers came out bad, but not terrible, you should consider limiting your Pan de Muerto intake.
4. This group represents that your numbers came out terribly, you should consider trying lettuce and visiting a doctor



```
NewCluster = kmeans.labels_[3]
print('%f', NewCluster)
```

Now lets run an example