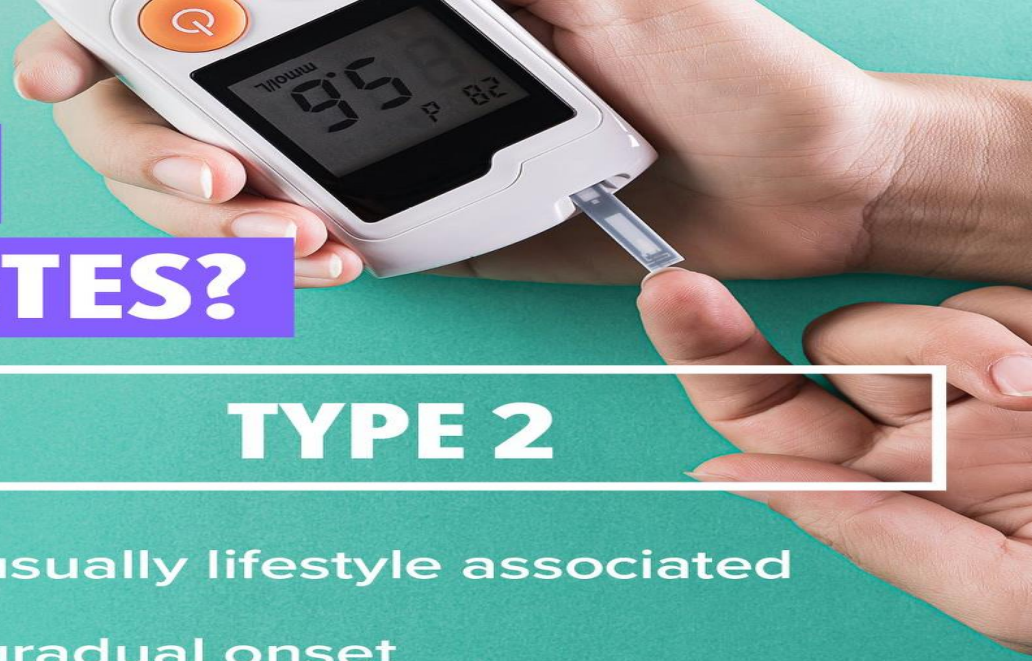
DIABETES TYPES

THE GOAL IS GET 100 % FOR Diabetes Expectation Class

What is the difference between TYPE 1 AND TYPE 2 DIABETES?



TYPE 1

- not usually lifestyle associated
- abrupt onset
- normally found in people aged under 20 years
- obvious symptoms
- autoimmune condition
- no cells that create insulin
- no insulin production
- insulin therapy required
- normally no family history

TYPE 2

- usually lifestyle associated
- gradual onset
- normally found in people aged over 35 years
- frequently no symptoms
- irregular function of cells that create insulin
- insulin normal or raised
- oral medication can be used
- family history common

Exploratory Data Analysis EDA

We used machine learning as an essential element in data science to make all data numeric

```
data.sample(10)
```

	Age	BS Fast	BS pp	Plasma R	Plasma F	HbA1c	Type	Class
106	27	4.8	7.7	11.0	6.1	36	Normal	0
302	35	41.0	6.8	10.9	4.2	33	Normal	0
234	23	28.0	7.7	11.0	6.1	36	Normal	0
463	37	30.0	5.6	10.2	5.4	32	Normal	0
114	39	6.8	8.8	11.2	7.2	62	Type1	1
887	52	5.8	4.2	11.4	8.4	53	Type2	1
639	28	12.0	7.7	11.0	6.1	36	Normal	0
623	21	27.0	5.6	10.2	5.4	32	Normal	0
397	22	5.8	6.3	12.6	8.1	49	Type1	1
726	25	29.0	7.7	11.0	6.1	36	Normal	0

check the null vallues.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1009 entries, 0 to 1008
Data columns (total 8 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Age        1009 non-null   int64
 1   BS Fast    1009 non-null   float64
 2   BS pp      1009 non-null   float64
 3   Plasma R   1009 non-null   float64
 4   Plasma F   1009 non-null   float64
 5   HbA1c      1009 non-null   int64
 6   Type       1009 non-null   object
 7   Class      1009 non-null   int64
dtypes: float64(4), int64(3), object(1)
memory usage: 63.2+ KB
```

check the duplicates .

```
data.shape
```

```
(1009, 8)
```

```
data = data.drop_duplicates()
```

```
data.shape
```

```
(623, 8)
```

```
data.Type.unique()
```

```
array(['Type1', 'Normal', 'Type2'], dtype=object)
```

```
data['Type'] = data['Type'].map({'Normal':0, 'Type1': 1, 'Type2': 2 })
```

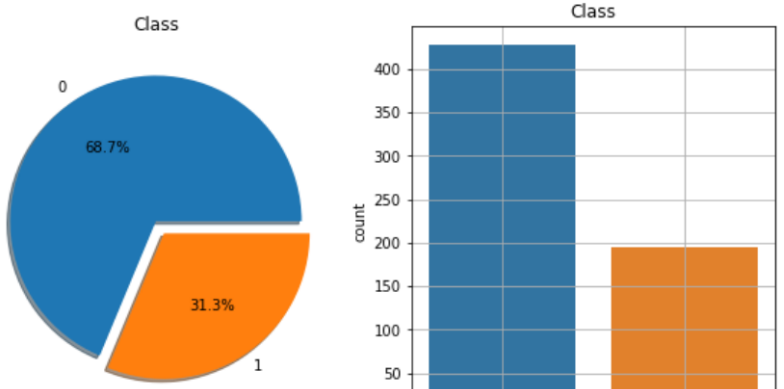
```
data.head(10)
```

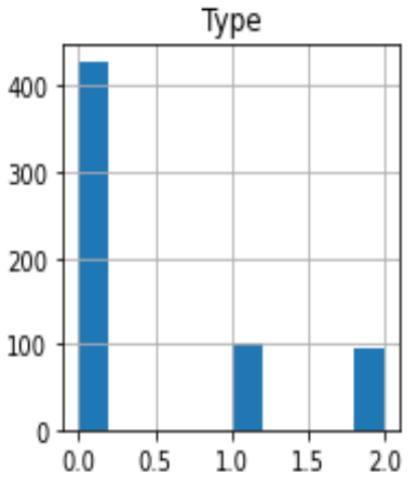
	Age	BS Fast	BS pp	Plasma R	Plasma F	HbA1c	Type	Class
0	50	6.8	8.8	11.2	7.2	62	1	1
1	31	5.2	6.8	10.9	4.2	33	0	0
2	32	6.8	8.8	11.2	7.2	62	1	1
3	21	5.7	5.8	10.7	4.8	49	0	0
4	33	6.8	8.8	11.2	7.2	62	1	1
5	30	5.2	7.4	8.7	5.6	41	0	0
6	26	5.8	4.2	11.4	8.4	53	2	1
7	29	5.2	7.4	8.7	5.6	41	0	0
8	53	6.9	8.4	11.2	7.2	62	2	1
9	54	6.3	4.2	12.2	7.8	57	2	1

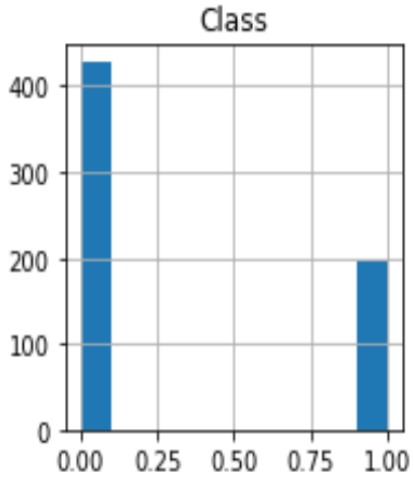
Helpful Visualizations

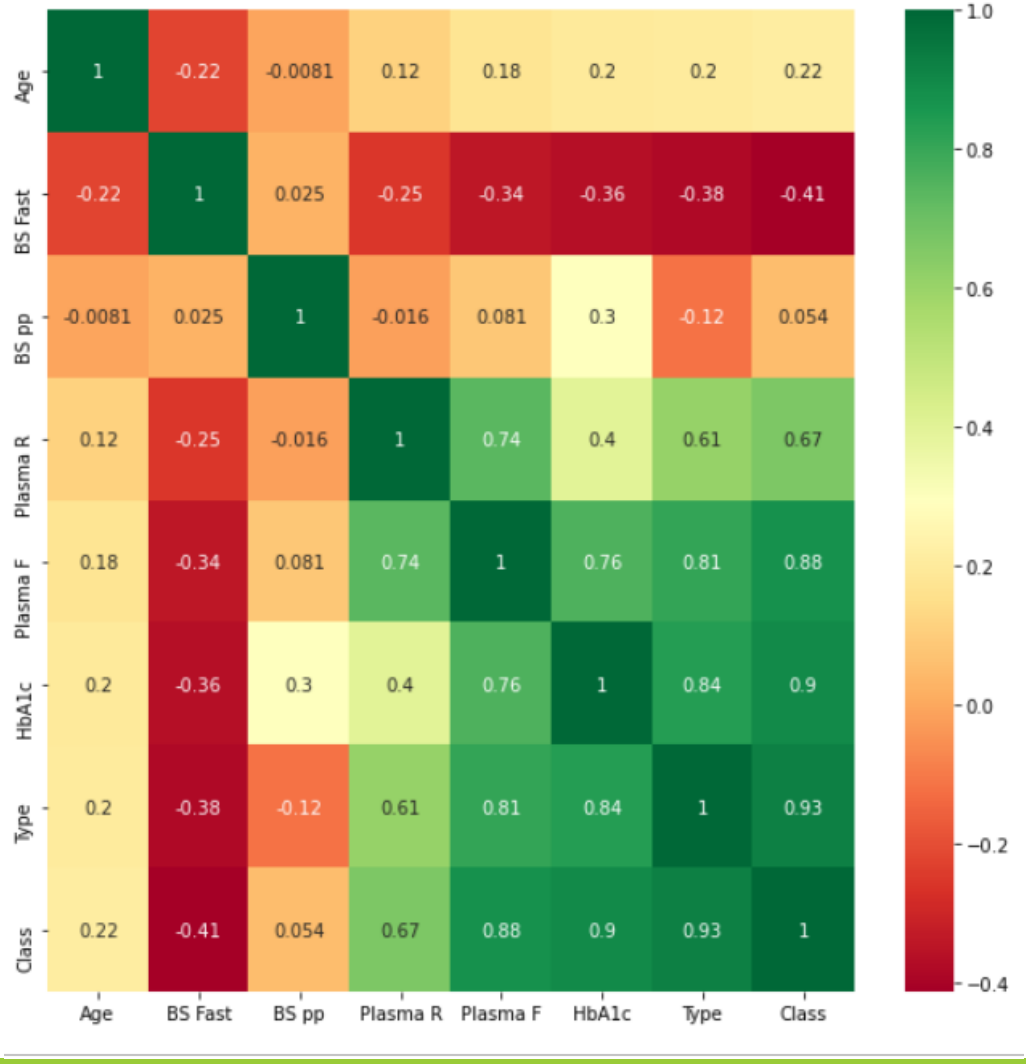
```
Negative (0):  428
Positive (1):  195
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```









start to split the data and get Accuracy score

```
x = data.iloc[:, [0,1,2,3,4,5,7]].values
```

```
y= data.iloc[:, 6].values
```

```
x
```

```
array([[50. ,  6.8,  8.8, ...,  7.2, 62. ,  1. ],
       [31. ,  5.2,  6.8, ...,  4.2, 33. ,  0. ],
       [32. ,  6.8,  8.8, ...,  7.2, 62. ,  1. ],
       ...,
       [45. , 40. ,  7.4, ...,  5.6, 41. ,  0. ],
       [27. , 34. ,  7.7, ...,  6.1, 36. ,  0. ],
       [21. , 23. ,  7.7, ...,  6.1, 36. ,  0. ]])
```

```
y
```

```
array([1, 0, 1, 0, 1, 0, 2, 0, 2, 2, 0, 2, 0, 2, 2, 2, 2, 2, 0, 0, 0, 2,
       2, 2, 2, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       1, 0, 2, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
```

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test =  train_test_split(x, y, test_size = 0.25, random_state = 0)
```

```
y_test
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 2, 2, 0, 0,
       2, 2, 1, 2, 0, 0, 0, 0, 1, 0, 1, 0, 2, 1, 2, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 2, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 1, 2, 0, 2, 2, 0, 0,
       1, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 2, 0, 1, 0, 0,
       1, 0, 0, 2, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 2, 1,
       1, 2], dtype=int64)
```

Apply standard scaler

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
x_train = sc_x.fit_transform(x_train)
x_test = sc_x.transform(x_test)
```

```
from sklearn.tree import DecisionTreeClassifier
Classifier = DecisionTreeClassifier(criterion = 'entropy', random_state=0)
Classifier.fit(x_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
y_pred = Classifier.predict(x_test)
```

```
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 2, 2, 0, 0,
       2, 2, 1, 2, 0, 0, 0, 0, 1, 0, 1, 0, 2, 1, 2, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 2, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 1, 2, 0, 2, 2, 0, 0,
       1, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 2, 0, 1, 0, 0,
       1, 0, 0, 2, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 2, 1,
       1, 2], dtype=int64)
```

```
y_test
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 2, 2, 0, 0,
       2, 2, 1, 2, 0, 0, 0, 0, 1, 0, 1, 0, 2, 1, 2, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 2, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 1, 2, 0, 2, 2, 0, 0,
       1, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 2, 0, 0, 0, 0, 2, 0, 1, 0, 0,
       1, 0, 0, 2, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 2, 1,
       1, 2], dtype=int64)
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

```
cm
```

```
array([[110,   0,   0],
       [  0,  20,   0],
       [  0,   0,  26]], dtype=int64)
```

```
from sklearn.metrics import accuracy_score
print('Accuracy score is :' ,accuracy_score(y_test, y_pred))
```

```
Accuracy score is : 1.0
```

we tested the model's results using two methods
1- confusion matrix 2- accuracy_score and got a 100% perfect

MANAGING DIABETES successfully requires:

self-discipline and a positive mindset of the patient,

as well as understanding support from loved ones

Prepare by: Atif Eid Alnefaie

Supervisor: Dr-Mariam – Dr-Shikh

Data scientist: Abdulrahman – Rasha