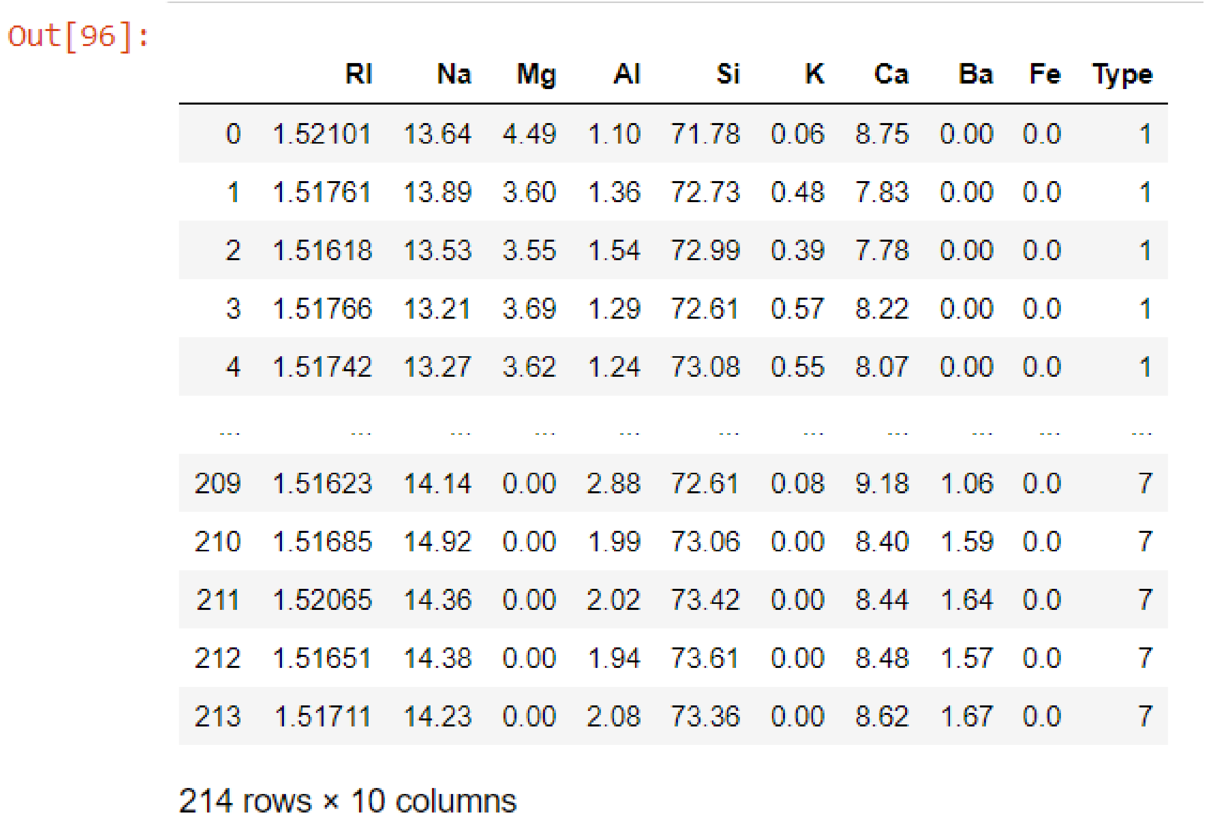
GLASS

import pandas as pd

import numby as np

df=pd.read_csv("glass.csv")

df



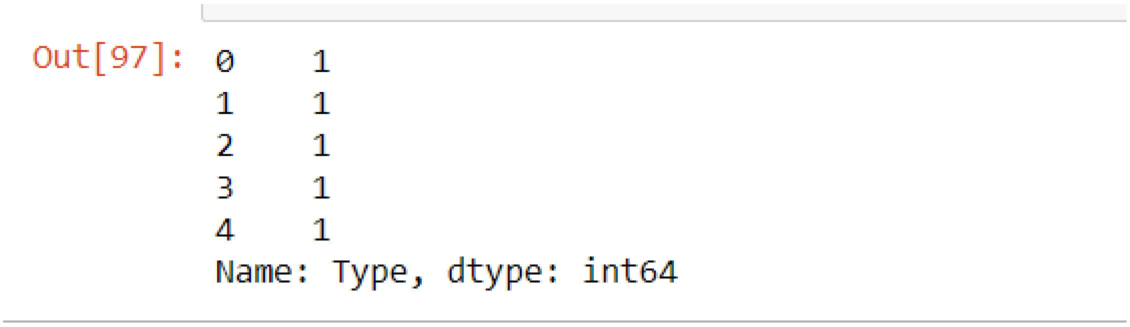
Out[96]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0	1
...	...	...	...	...	...	...	...	...	...	...
209	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
210	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
211	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
212	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
213	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

214 rows × 10 columns

Y= df["Type"]

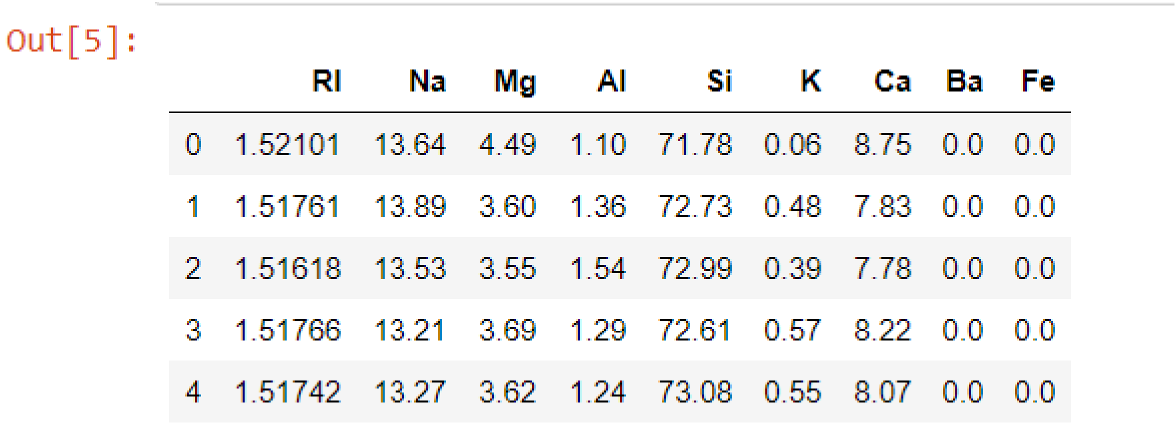
Y.head()



```
Out[97]: 0    1
         1    1
         2    1
         3    1
         4    1
         Name: Type, dtype: int64
```

df = df.drop("Type",axis=1)

df.head()



Out[5]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0

from sklearn import preprocessing

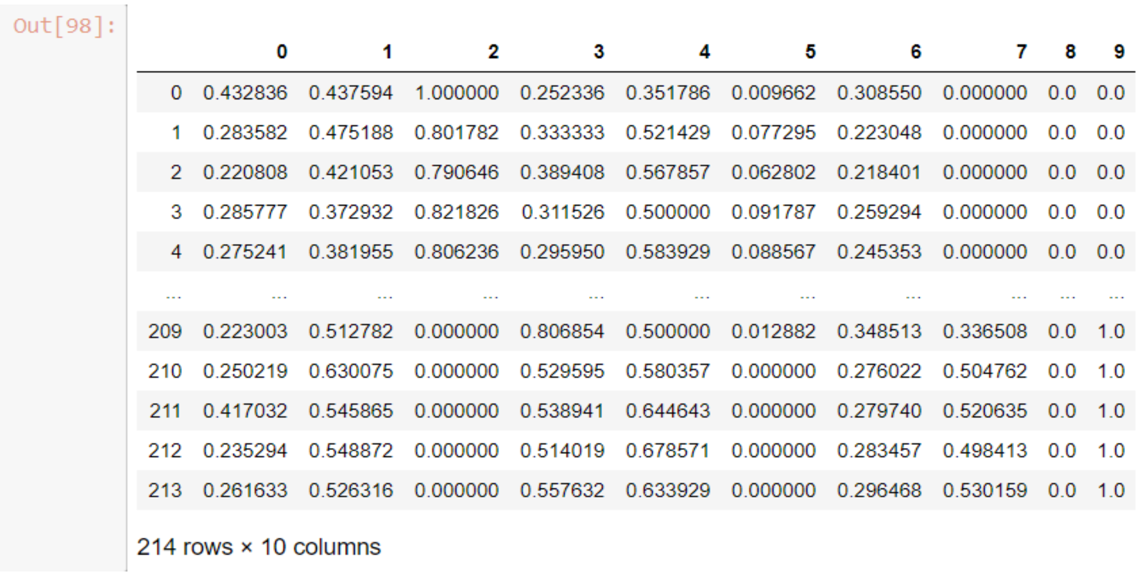
x = df.values

min_max_scaler = preprocessing.MinMaxScaler()

x_scaled = min_max_scaler.fit_transform(x)

df = pd.DataFrame(x_scaled)

df



Out[98]:

	0	1	2	3	4	5	6	7	8	9
0	0.432836	0.437594	1.000000	0.252336	0.351786	0.009662	0.308550	0.000000	0.0	0.0
1	0.283582	0.475188	0.801782	0.333333	0.521429	0.077295	0.223048	0.000000	0.0	0.0
2	0.220808	0.421053	0.790646	0.389408	0.567857	0.062802	0.218401	0.000000	0.0	0.0
3	0.285777	0.372932	0.821826	0.311526	0.500000	0.091787	0.259294	0.000000	0.0	0.0
4	0.275241	0.381955	0.806236	0.295950	0.583929	0.088567	0.245353	0.000000	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
209	0.223003	0.512782	0.000000	0.806854	0.500000	0.012882	0.348513	0.336508	0.0	1.0
210	0.250219	0.630075	0.000000	0.529595	0.580357	0.000000	0.276022	0.504762	0.0	1.0
211	0.417032	0.545865	0.000000	0.538941	0.644643	0.000000	0.279740	0.520635	0.0	1.0
212	0.235294	0.548872	0.000000	0.514019	0.678571	0.000000	0.283457	0.498413	0.0	1.0
213	0.261633	0.526316	0.000000	0.557632	0.633929	0.000000	0.296468	0.530159	0.0	1.0

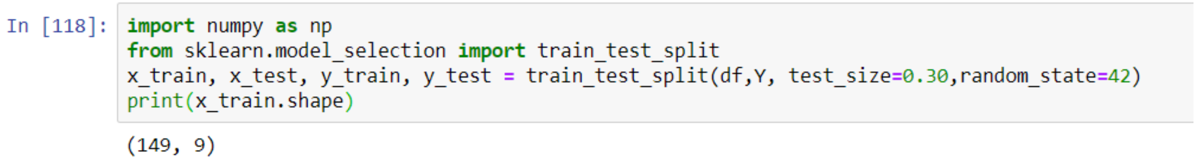
214 rows × 10 columns

import numpy as np

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(df,Y, test_size=0.30,random_state=42)

print(x_train.shape)



```
In [118]: import numpy as np
          from sklearn.model_selection import train_test_split
          x_train, x_test, y_train, y_test = train_test_split(df,Y, test_size=0.30,random_state=42)
          print(x_train.shape)

          (149, 9)
```

from sklearn.tree import DecisionTreeClassifier

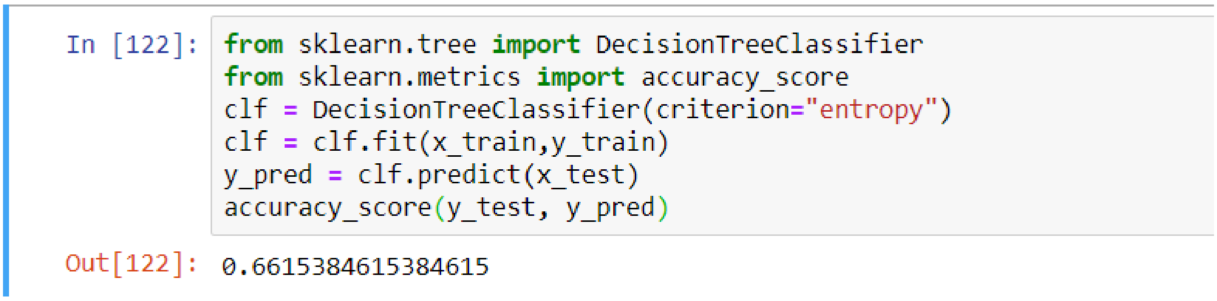
from sklearn.metrics import accuracy_score

clf = DecisionTreeClassifier(criterion="entropy")

clf = clf.fit(x_train,y_train)

y_pred = clf.predict(x_test)

accuracy_score(y_test, y_pred)



```
In [122]: from sklearn.tree import DecisionTreeClassifier
          from sklearn.metrics import accuracy_score
          clf = DecisionTreeClassifier(criterion="entropy")
          clf = clf.fit(x_train,y_train)
          y_pred = clf.predict(x_test)
          accuracy_score(y_test, y_pred)

Out[122]: 0.6615384615384615
```

from sklearn.naive_bayes import GaussianNB

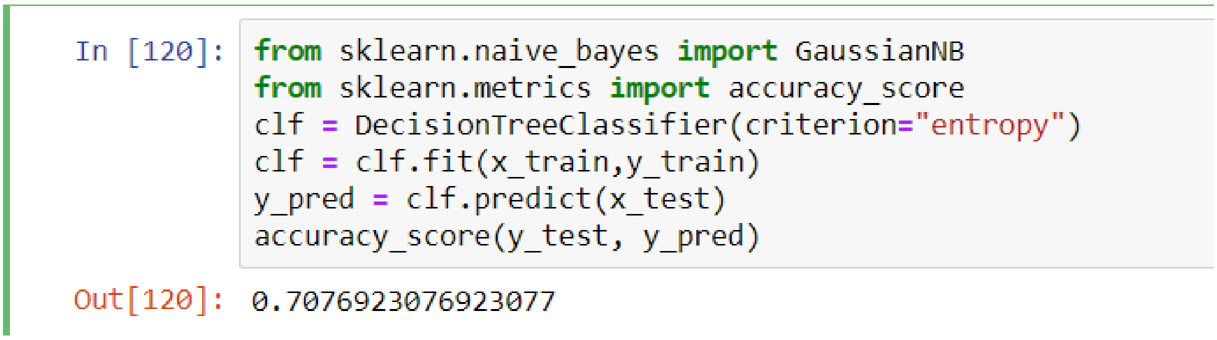
from sklearn.metrics import accuracy_score

clf = DecisionTreeClassifier(criterion="entropy")

clf = clf.fit(x_train,y_train)

y_pred = clf.predict(x_test)

accuracy_score(y_test, y_pred)



```
In [120]: from sklearn.naive_bayes import GaussianNB
          from sklearn.metrics import accuracy_score
          clf = DecisionTreeClassifier(criterion="entropy")
          clf = clf.fit(x_train,y_train)
          y_pred = clf.predict(x_test)
          accuracy_score(y_test, y_pred)

Out[120]: 0.7076923076923077
```

from sklearn.neural_network import MLPClassifier

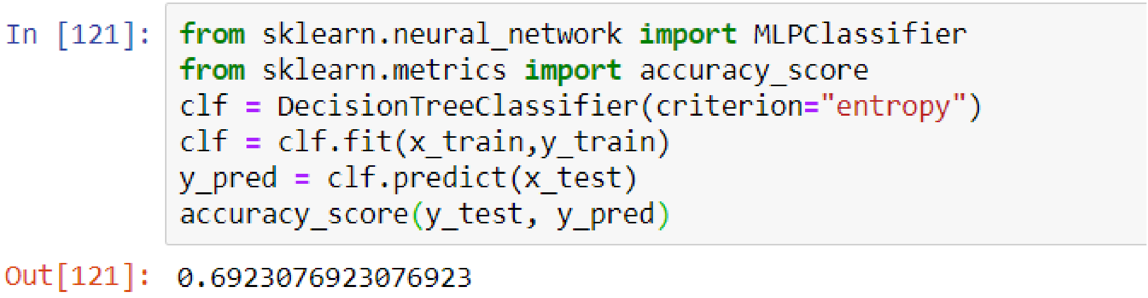
from sklearn.metrics import accuracy_score

```
clf = DecisionTreeClassifier(criterion="entropy")

clf = clf.fit(x_train,y_train)

y_pred = clf.predict(x_test)

accuracy_score(y_test, y_pred)
```



```
In [121]: from sklearn.neural_network import MLPClassifier
          from sklearn.metrics import accuracy_score
          clf = DecisionTreeClassifier(criterion="entropy")
          clf = clf.fit(x_train,y_train)
          y_pred = clf.predict(x_test)
          accuracy_score(y_test, y_pred)

Out[121]: 0.6923076923076923
```

DIABETES

import pandas as pd

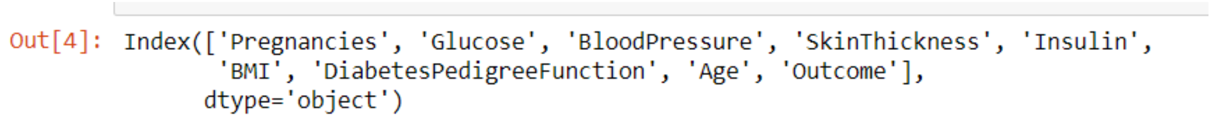
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

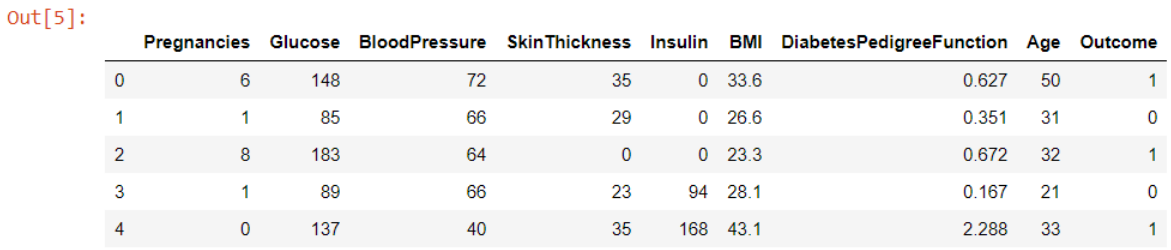
diabetes = pd.read_csv('diabetes.csv')

diabetes.columns



```
Out[4]: Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
               'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
              dtype='object')
```

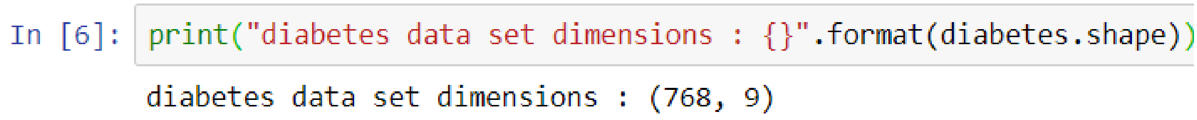
diabetes.head()



Out[5]:

	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

print("diabetes data set dimensions : {}".format(diabetes.shape))

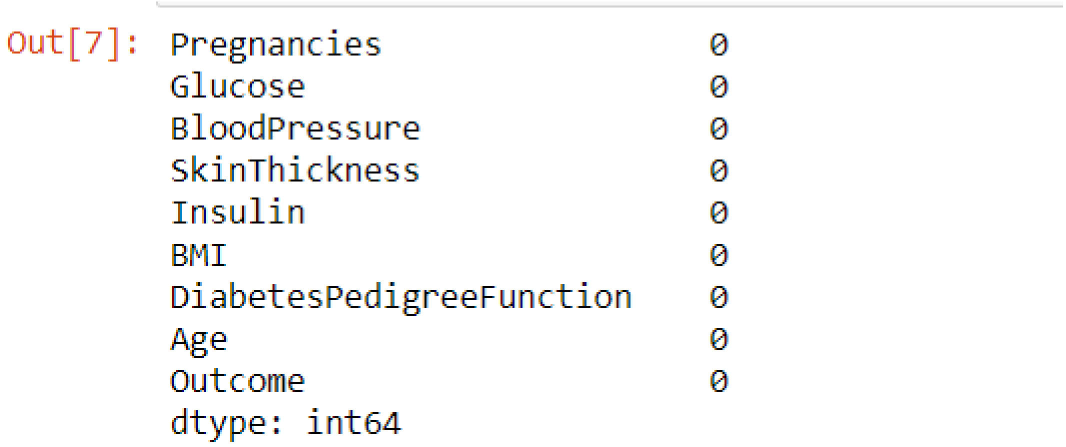


```
In [6]: print("diabetes data set dimensions : {}".format(diabetes.shape))

        diabetes data set dimensions : (768, 9)
```

diabetes.isnull().sum()

diabetes.isna().sum()

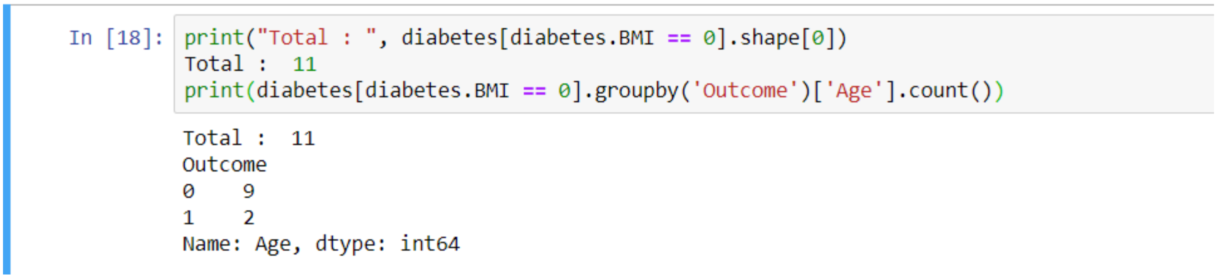


```
Out[7]: Pregnancies                 0
        Glucose                     0
        BloodPressure               0
        SkinThickness               0
        Insulin                     0
        BMI                         0
        DiabetesPedigreeFunction    0
        Age                         0
        Outcome                     0
        dtype: int64
```

print("Total : ", diabetes[diabetes.BMI == 0].shape[0])

Total : 11

print(diabetes[diabetes.BMI == 0].groupby('Outcome')['Age'].count())



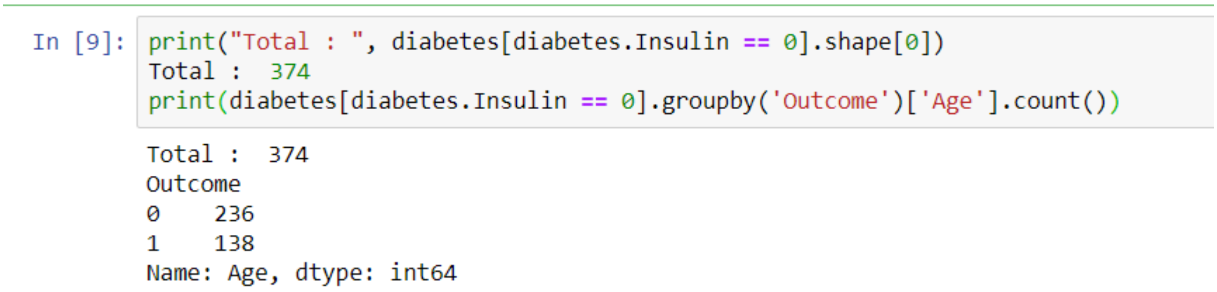
```
In [18]: print("Total : ", diabetes[diabetes.BMI == 0].shape[0])
         Total :  11
         print(diabetes[diabetes.BMI == 0].groupby('Outcome')['Age'].count())

         Total :  11
         Outcome
         0    9
         1    2
         Name: Age, dtype: int64
```

print("Total : ", diabetes[diabetes.Insulin == 0].shape[0])

Total : 374

print(diabetes[diabetes.Insulin == 0].groupby('Outcome')['Age'].count())



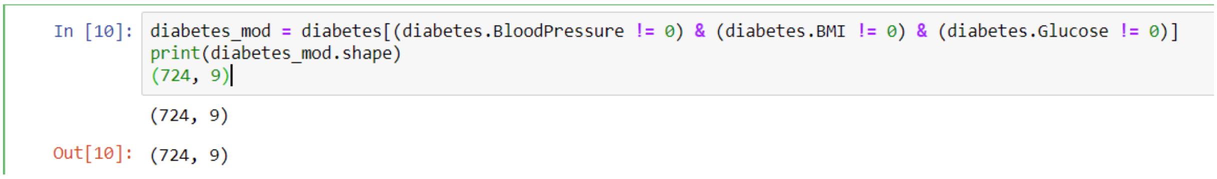
```
In [9]: print("Total : ", diabetes[diabetes.Insulin == 0].shape[0])
        Total :  374
        print(diabetes[diabetes.Insulin == 0].groupby('Outcome')['Age'].count())

        Total :  374
        Outcome
        0    236
        1    138
        Name: Age, dtype: int64
```

diabetes_mod = diabetes[(diabetes.BloodPressure != 0) & (diabetes.BMI != 0) & (diabetes.Glucose != 0)]

print(diabetes_mod.shape)

(724, 9)



```
In [10]: diabetes_mod = diabetes[(diabetes.BloodPressure != 0) & (diabetes.BMI != 0) & (diabetes.Glucose != 0)]
         print(diabetes_mod.shape)
         (724, 9)

         (724, 9)

Out[10]: (724, 9)
```

feature_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']

X = diabetes_mod[feature_names]

y = diabetes_mod.Outcome

from sklearn.neighbors import KNeighborsClassifier

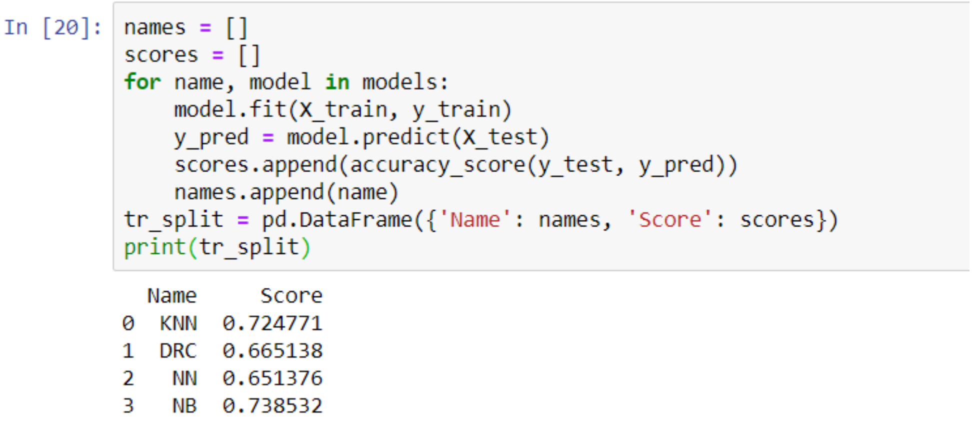
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB

models = []
models.append(('KNN', KNeighborsClassifier()))
models.append(('DRC', DecisionTreeClassifier()))
models.append(('NN', MLPClassifier()))
models.append(('NB', GaussianNB()))

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.30, random_state=0)

names = []
scores = []
for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    scores.append(accuracy_score(y_test, y_pred))
    names.append(name)
tr_split = pd.DataFrame({'Name': names, 'Score': scores})
print(tr_split)
```



```
In [20]: names = []
         scores = []
         for name, model in models:
             model.fit(X_train, y_train)
             y_pred = model.predict(X_test)
             scores.append(accuracy_score(y_test, y_pred))
             names.append(name)
         tr_split = pd.DataFrame({'Name': names, 'Score': scores})
         print(tr_split)
```

```
  Name     Score
0  KNN  0.724771
1  DRC  0.665138
2   NN  0.651376
3   NB  0.738532
```