main

October 19, 2024

[62]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler
```

[18]:
```
# Download latest version
path = kagglehub.dataset_download("nanditapore/healthcare-diabetes")

print("Path to dataset files:", path)
```

```
Path to dataset files:
/home/jovyan/.cache/kagglehub/datasets/nanditapore/healthcare-
diabetes/versions/1
```

[26]:
```
csv_file = f"{path}/Healthcare-Diabetes.csv"  # Adjust filename based on actual␣
↪dataset contents
df = pd.read_csv(csv_file)
```

#In the first part I will analyze all datas, and clean them

[28]:
```
df.columns
```

[28]:
```
Index(['Id', 'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
       'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

#I will drop the id to not influence the target y

[34]:
```
df.drop('Id', axis=1)
df.columns
```

[34]:
```
Index(['Id', 'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
       'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

[59]:
```
df.columns = ['Id', 'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',␣
↪'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
```

[35]:
```
print(df.isnull().sum())
```

```
Id                          0
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

1 a good point is that we dont have null values :)

[37]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2768 entries, 0 to 2767
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Id                        2768 non-null   int64
 1   Pregnancies               2768 non-null   int64
 2   Glucose                   2768 non-null   int64
 3   BloodPressure             2768 non-null   int64
 4   SkinThickness             2768 non-null   int64
 5   Insulin                   2768 non-null   int64
 6   BMI                       2768 non-null   float64
 7   DiabetesPedigreeFunction  2768 non-null   float64
 8   Age                       2768 non-null   int64
 9   Outcome                   2768 non-null   int64
dtypes: float64(2), int64(8)
memory usage: 216.4 KB
```

[39]:
```
df.describe().all
```

[39]:
```
<bound method DataFrame.all of                 Id  Pregnancies      Glucose
BloodPressure  SkinThickness  \
count  2768.000000  2768.000000  2768.000000    2768.000000    2768.000000
mean   1384.500000     3.742775   121.102601      69.134393      20.824422
std     799.197097     3.323801    32.036508      19.231438      16.059596
min       1.000000     0.000000     0.000000       0.000000       0.000000
25%     692.750000     1.000000    99.000000      62.000000       0.000000
50%    1384.500000     3.000000   117.000000      72.000000      23.000000
75%    2076.250000     6.000000   141.000000      80.000000      32.000000
```

```
max    2768.000000    17.000000   199.000000     122.000000     110.000000

           Insulin          BMI  DiabetesPedigreeFunction          Age  \
count  2768.000000  2768.000000               2768.000000  2768.000000
mean     80.127890    32.137392                  0.471193    33.132225
std     112.301933     8.076127                  0.325669    11.777230
min       0.000000     0.000000                  0.078000    21.000000
25%       0.000000    27.300000                  0.244000    24.000000
50%      37.000000    32.200000                  0.375000    29.000000
75%     130.000000    36.625000                  0.624000    40.000000
max     846.000000    80.600000                  2.420000    81.000000

           Outcome
count  2768.000000
mean      0.343931
std       0.475104
min       0.000000
25%       0.000000
50%       0.000000
75%       1.000000
max       1.000000  >
```

[50]:
```
#Futures is X that sotare all columns (axis 1) but Outcome
X = df.drop('Outcome', axis=1)
#y is the target
y = df['Outcome']
```

#split df ito training and testing

[54]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[56]:
```
# Initialize the scaler
scaler = StandardScaler()
```

[57]:
```
X_train_scaled = scaler.fit_transform(X_train) #we must fit the scaler data
```

[58]:
```
X_test_scaled = scaler.transform(X_test)
```

#begin to create the model

[64]:
```
# Initialize the Random Forest Classifier
model = RandomForestClassifier(random_state=42)
```

[65]:
```
model.fit(X_train_scaled, y_train) #we fit the data
```

[65]:
```
RandomForestClassifier(random_state=42)
```

[67]:
```
y_pred = model.predict(X_test_scaled)
```

#calcualate acurancy

[68]:
```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

```
Accuracy: 0.98
```

[70]:
```
report = classification_report(y_test, y_pred)  # Generate classification report
print(report)
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       367
           1       0.98      0.96      0.97       187

    accuracy                           0.98       554
   macro avg       0.98      0.98      0.98       554
weighted avg       0.98      0.98      0.98       554
```

[ ]:

[ ]: