diabetes-prediction-using-ann

March 25, 2024

[10]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[11]:
```
from sklearn.datasets?
```

Object `sklearn.datasets` not found.

[12]:
```
df = pd.read_csv('diabetes.csv')
df.head()
```

[12]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	2	138	62	35	0	33.6	0.127	47	1
1	0	84	82	31	125	38.2	0.233	23	0
2	0	145	0	0	0	44.2	0.630	31	1
3	0	135	68	42	250	42.3	0.365	24	1
4	1	139	62	41	480	40.7	0.536	21	0

[13]:
```
df.isnull().sum()
```

[13]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               2000 non-null   int64  
 1   Glucose                   2000 non-null   int64  
 2   BloodPressure             2000 non-null   int64  
 3   SkinThickness             2000 non-null   int64  
 4   Insulin                   2000 non-null   int64  
 5   BMI                       2000 non-null   float64
 6   DiabetesPedigreeFunction  2000 non-null   float64
 7   Age                       2000 non-null   int64  
 8   Outcome                   2000 non-null   int64  
dtypes: float64(2), int64(7)
memory usage: 140.8 KB
```

[15]:
```
df.duplicated()
```

[15]:
```
0        False
1        False
2        False
3        False
4        False
        ...
1995      True
1996      True
1997      True
1998      True
1999      True
Length: 2000, dtype: bool
```

[16]:
```
df.duplicated().sum()
```

[16]:
```
1256
```

[17]:
```
features = [x for x in df.columns if x not in df.columns[-1]]
features
```

[17]:
```
['Pregnancies',
 'Glucose',
 'BloodPressure',
 'SkinThickness',
 'Insulin',
 'BMI',
```

```
 'DiabetesPedigreeFunction',
 'Age']
```

[18]:
```
target = df.columns[-1]
target
```

[18] : 'Outcome'

[19]:
```
df_features = df.drop("Outcome", axis = 1)
df_features.head()
```

[19] :

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	2	138	62	35	0	33.6	
1	0	84	82	31	125	38.2	
2	0	145	0	0	0	44.2	
3	0	135	68	42	250	42.3	
4	1	139	62	41	480	40.7	

	DiabetesPedigreeFunction	Age
0	0.127	47
1	0.233	23
2	0.630	31
3	0.365	24
4	0.536	21

[20]:
```
x = np.array(df_features)    # feature array
x[:5,:]
```

[20]:
```
array([[2.00e+00, 1.38e+02, 6.20e+01, 3.50e+01, 0.00e+00, 3.36e+01,
        1.27e-01, 4.70e+01],
       [0.00e+00, 8.40e+01, 8.20e+01, 3.10e+01, 1.25e+02, 3.82e+01,
        2.33e-01, 2.30e+01],
       [0.00e+00, 1.45e+02, 0.00e+00, 0.00e+00, 0.00e+00, 4.42e+01,
        6.30e-01, 3.10e+01],
       [0.00e+00, 1.35e+02, 6.80e+01, 4.20e+01, 2.50e+02, 4.23e+01,
        3.65e-01, 2.40e+01],
       [1.00e+00, 1.39e+02, 6.20e+01, 4.10e+01, 4.80e+02, 4.07e+01,
        5.36e-01, 2.10e+01]])
```

[21]:
```
y = np.array(df["Outcome"])
y = np.expand_dims(y,axis=1) # reshaping to 2d array
y[:5]
```

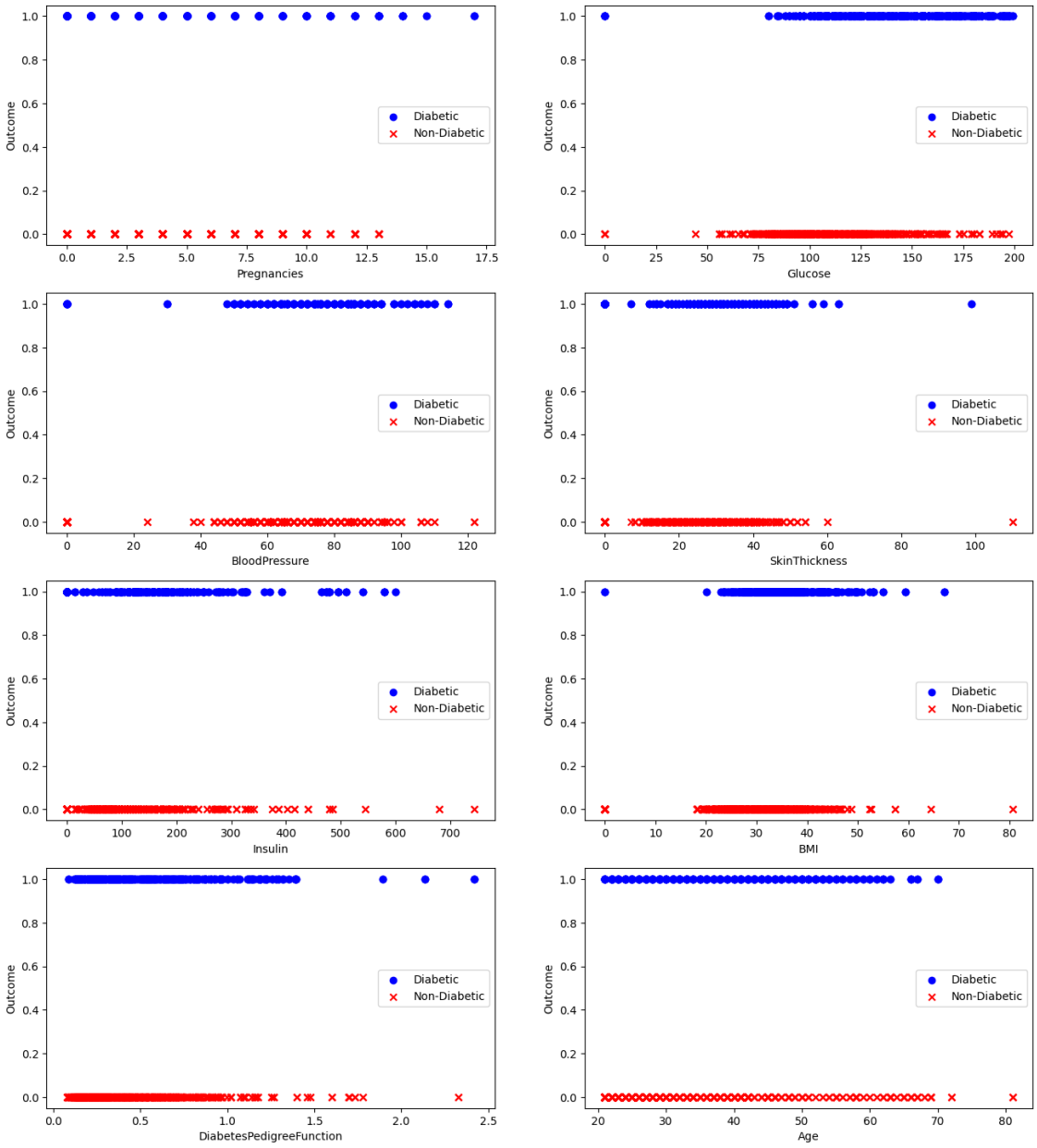
[21]:
```
array([[1],
       [0],
       [1],
       [1],
```

[0]], dtype=int64)

[22]:
```
fig, ax = plt.subplots(4,2, figsize=(16,18))
ax = ax.flatten()
for i in range(len(features)):

    ax[i].scatter([x_val for x_val, y_val in zip(x[:,i], y) if y_val == 1],
            [y_val for y_val in y if y_val == 1],
            marker="o", color="blue", label="Diabetic")

    ax[i].scatter([x_val for x_val, y_val in zip(x[:,i], y) if y_val == 0],
            [y_val for y_val in y if y_val == 0],
            marker="x", color="red",  label="Non-Diabetic")

    ax[i].set_xlabel(features[i])
    ax[i].set_ylabel(target)
    ax[i].legend()

fig.suptitle("Feature vs target plot", fontsize=25)
```

[22]: Text(0.5, 0.98, 'Feature vs target plot')

Feature vs target plot



[25]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import joblib
```

[26]:
```
x_train, x_temp, y_train, y_temp = train_test_split(x,y, test_size=0.4,␣
↪random_state = 1)
x_cv, x_test, y_cv, y_test = train_test_split(x_temp, y_temp, test_size=0.4,␣
↪random_state = 1)
```

[27]:
```
scaler = StandardScaler()
x_train_norm = scaler.fit_transform(x_train)
x_cv_norm = scaler.transform(x_cv)
x_test_norm = scaler.transform(x_test)
joblib.dump(scaler, 'scaler.joblib')
```

[27]: ['scaler.joblib']

[28]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.regularizers import l2
```

[29]:
```
def build_models():
    model_1 = Sequential([
        Dense(units=20, activation='relu', kernel_regularizer = l2(0.01),␣
↪name='L1'), # using Ridge regularization
        Dense(units=10, activation='relu', kernel_regularizer = l2(0.01),␣
↪name='L2'),
        Dense(units=1, activation='linear', kernel_regularizer = l2(0.01),␣
↪name='L3')
    ], name = 'model_1')

    model_2 = Sequential([
        Dense(units=25, activation='relu', kernel_regularizer = l2(0.01),␣
↪name='L1'),
        Dense(units=15, activation='relu', kernel_regularizer = l2(0.01),␣
↪name='L2'),
        Dense(units=1, activation='linear', kernel_regularizer = l2(0.01),␣
↪name='L3')
    ], name = 'model_2')


    model_3 = Sequential([
        Dense(units=20, activation='relu', kernel_regularizer = l2(0.01),␣
↪name='L1'),
        Dense(units=12, activation='relu', kernel_regularizer = l2(0.01),␣
↪name='L2'),
        Dense(units=12, activation='relu', kernel_regularizer = l2(0.01),␣
↪name='L3'),
```

```
        Dense(units=1, activation='linear', kernel_regularizer = l2(0.01),␣
↪name='L4')
    ], name = 'model_3')

    model_4 = Sequential([
        Dense(units=32, activation='relu', kernel_regularizer = l2(0.01),␣
↪name='L1'),
        Dense(units=16, activation='relu', kernel_regularizer = l2(0.01),␣
↪name='L2'),
        Dense(units=8, activation='relu', kernel_regularizer = l2(0.01),␣
↪name='L3'),
        Dense(units=4, activation='relu', kernel_regularizer = l2(0.01),␣
↪name='L4'),
        Dense(units=12, activation='relu', kernel_regularizer = l2(0.01),␣
↪name='L5'),
        Dense(units=1, activation='linear', kernel_regularizer = l2(0.01),␣
↪name='L6')
    ], name = 'model_4')

    model_list=[model_1, model_2, model_3, model_4]

    return model_list
```

[30]:
```
models = build_models()
```

[31]:
```
# Training error of each model
train_error_list= []

# Validation error of each model
val_error_list= []


 # training each model
 for model in models:
    model.compile(
        loss = tf.keras.losses.BinaryCrossentropy(from_logits=True),
        # sigmoid activation funciton is internally applied while calculating␣
↪BinaryCrossentopy loss
        # so that the roundoff error are minimized
        optimizer = tf.keras.optimizers.Adam(learning_rate = 0.01 )
    )

    print(f'training... {model.name}')
    model.fit(
        x_train, y_train,
        epochs=200,
```

```
        verbose=0 # not displaying training progress
    )

    # Computing training error
    yhat = model.predict(x_train)  # yhat is a linear function z = w.x + b
    yhat_sigmoid = tf.nn.sigmoid(yhat) # g(z) = 1/(1+e^(-z))

    threshold = 0.5

    yhat = np.where(yhat_sigmoid >= threshold, 1, 0)
    train_error = np.mean(yhat != y_train)
    train_error_list.append(train_error)

    # Computing validation error
    yhat = model.predict(x_cv)
    # getting probability values since the output layer is 'linear'
    yhat_sigmoid = tf.nn.sigmoid(yhat)

    yhat = np.where(yhat_sigmoid >= threshold, 1, 0)
    val_error = np.mean(yhat != y_cv)
    val_error_list.append(val_error)
```

```
training... model_1
38/38              0s 3ms/step
15/15              0s 2ms/step
training... model_2
38/38              0s 2ms/step
15/15              0s 2ms/step
training... model_3
38/38              0s 4ms/step
15/15              0s 3ms/step
training... model_4
38/38              0s 3ms/step
15/15              0s 3ms/step
```

[32]:
```
print('Displaying Training and Validation error for each models\n')
for i,j in enumerate(models):
    print(f'Training error for {j.name} = {train_error_list[i]}')
    print(f'Validation error for {j.name} = {val_error_list[i]}\n')
```

```
Displaying Training and Validation error for each models

Training error for model_1 = 0.23083333333333333
Validation error for model_1 = 0.225

Training error for model_2 = 0.2325
Validation error for model_2 = 0.25833333333333336
```

Training error for model_3 = 0.22416666666666665
Validation error for model_3 = 0.23333333333333334

Training error for model_4 = 0.2525
Validation error for model_4 = 0.2375

[33]:
```
model_index = np.argmin(val_error_list)models[model_index].name
```

[33]: 'model_1'

[34]:
```
best_model = models[model_index]
best_model.summary()
```

Model: "model_1"

Layer (type)	Output Shape	Param #
L1 (Dense)	(None, 20)	180
L2 (Dense)	(None, 10)	210
L3 (Dense)	(None, 1)	11

Total params: 1,205 (4.71 KB)

Trainable params: 401 (1.57 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 804 (3.14 KB)

[36]:
```
from sklearn.metrics import accuracy_score
```

[37]:
```
yhat = best_model.predict(x_test)
yhat_sigmoid = tf.nn.sigmoid(yhat)
yhat = np.where(yhat_sigmoid >= threshold, 1, 0)
test_error = yhat != y_test
test_error = np.mean(test_error)
print(f'Test error = {test_error}')
```

```
10/10                0s 2ms/step
Test error = 0.221875
```

[38]:
```
tf.keras.models.save_model(best_model,'model.h5')
```

[39]:
```
nn_model = tf.keras.models.load_model('model.h5')
saved_scaler = joblib.load('scaler.joblib')
```

[40]:
```
print('Enter the following data:\n')
x_input = []
for i in range(len(features)):
    print(features[i])
    x_i = float(input())
    x_input.append(x_i)

x_input = np.array(x_input).reshape(1, x_train.shape[1])
x_input_norm = saved_scaler.transform(x_input)
pred = nn_model.predict(x_input_norm)
pred = tf.nn.sigmoid(pred)
```

```
Enter the following data:

Pregnancies
3
Glucose
84
BloodPressure
72
SkinThickness
32
Insulin
0
BMI
37.2
DiabetesPedigreeFunction
0.267
Age
28
```

1/1 **0s** 79ms/step

[41]:
```
if pred >= threshold:
    print('Sorry! You have a higher risk of developing diabetes based on your␣
 ↪medical data.')
else:
    print('Congratulations! You have a low risk of developing diabetes based on␣
 ↪your medical data.')
```

Congratulations! You have a low risk of developing diabetes based on your medical data.

[ ]: