Neural_Netwrok_PyTorch

May 29, 2025

[1]:
```
# Implementation of Neural Network using DeepLearning Framework (PyTorch)
```

[2]:
```
# Data Processing
```

[17]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
```

[18]:
```
df = pd.read_csv("KaggleV2-May-2016.csv")
df.head()
```

[18]:

	PatientId	AppointmentID	Gender	ScheduledDay	\
0	2.987250e+13	5642903	F	2016-04-29T18:38:08Z	
1	5.589978e+14	5642503	M	2016-04-29T16:08:27Z	
2	4.262962e+12	5642549	F	2016-04-29T16:19:04Z	
3	8.679512e+11	5642828	F	2016-04-29T17:29:31Z	
4	8.841186e+12	5642494	F	2016-04-29T16:07:23Z	

	AppointmentDay	Age	Neighbourhood	Scholarship	Hipertension	\
0	2016-04-29T00:00:00Z	62	JARDIM DA PENHA	0	1	
1	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	0	
2	2016-04-29T00:00:00Z	62	MATA DA PRAIA	0	0	
3	2016-04-29T00:00:00Z	8	PONTAL DE CAMBURI	0	0	
4	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	1	

	Diabetes	Alcoholism	Handcap	SMS_received	No-show
0	0	0	0	0	No
1	0	0	0	0	No
2	0	0	0	0	No
3	0	0	0	0	No
4	1	0	0	0	No

[19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   PatientId       110527 non-null  float64
 1   AppointmentID   110527 non-null  int64  
 2   Gender          110527 non-null  object 
 3   ScheduledDay    110527 non-null  object 
 4   AppointmentDay  110527 non-null  object 
 5   Age             110527 non-null  int64  
 6   Neighbourhood   110527 non-null  object 
 7   Scholarship     110527 non-null  int64  
 8   Hipertension    110527 non-null  int64  
 9   Diabetes        110527 non-null  int64  
 10  Alcoholism      110527 non-null  int64  
 11  Handcap         110527 non-null  int64  
 12  SMS_received    110527 non-null  int64  
 13  No-show         110527 non-null  object 
dtypes: float64(1), int64(8), object(5)
memory usage: 11.8+ MB
```

[20]:
```
df['No-show'] = df['No-show'].map({"No" : 0, "Yes" : 1})
df['Gender'] = df['Gender'].map({"F" : 0, "M" : 1})
```

[21]:
```
duplicates = df.duplicated().sum()
print(duplicates)
```

```
0
```

[22]:
```
df["Neighbourhood"] = df["Neighbourhood"].astype('category').cat.codes + 1
```

[23]:
```
df.head()
```

[23]:

	PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarship	Hipertension
0	2.987250e+13	5642903	0	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	40	0	1
1	5.589978e+14	5642503	1	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	40	0	0
2	4.262962e+12	5642549	0	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	46	0	0
3	8.679512e+11	5642828	0	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	55	0	0
4	8.841186e+12	5642494	0	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	40	0	1

```
   Diabetes  Alcoholism  Handcap  SMS_received  No-show
0         0           0        0             0        0
1         0           0        0             0        0
2         0           0        0             0        0
3         0           0        0             0        0
4         1           0        0             0        0
```

[24]:
```
df["AppointmentDay"] = pd.to_datetime(df["AppointmentDay"])
df["Appointment_Day"] = df["AppointmentDay"].dt.day
df["Appointment_Month"] = df["AppointmentDay"].dt.month
df["ScheduledDay"] = pd.to_datetime(df["ScheduledDay"])
df["Scheduled_Day"] = df["ScheduledDay"].dt.day
df["Scheduled_Month"] = df["ScheduledDay"].dt.month
df.drop(["PatientId","AppointmentID", "ScheduledDay", "AppointmentDay"], axis =␣
↪1, inplace = True)
df.head()
```

[24]:
```
   Gender  Age  Neighbourhood  Scholarship  Hipertension  Diabetes  \
0       0   62             40            0             1         0
1       1   56             40            0             0         0
2       0   62             46            0             0         0
3       0    8             55            0             0         0
4       0   56             40            0             1         1

   Alcoholism  Handcap  SMS_received  No-show  Appointment_Day  \
0           0        0             0        0               29
1           0        0             0        0               29
2           0        0             0        0               29
3           0        0             0        0               29
4           0        0             0        0               29

   Appointment_Month  Scheduled_Day  Scheduled_Month
0                  4             29                4
1                  4             29                4
2                  4             29                4
3                  4             29                4
4                  4             29                4
```

[50]:
```
df.head(3)
```

[50]:
```
   Gender  Age  Neighbourhood  Scholarship  Hipertension  Diabetes  \
0       0   62             40            0             1         0
1       1   56             40            0             0         0
2       0   62             46            0             0         0

   Alcoholism  Handcap  SMS_received  No-show  Appointment_Day  \
0           0        0             0        0               29
```

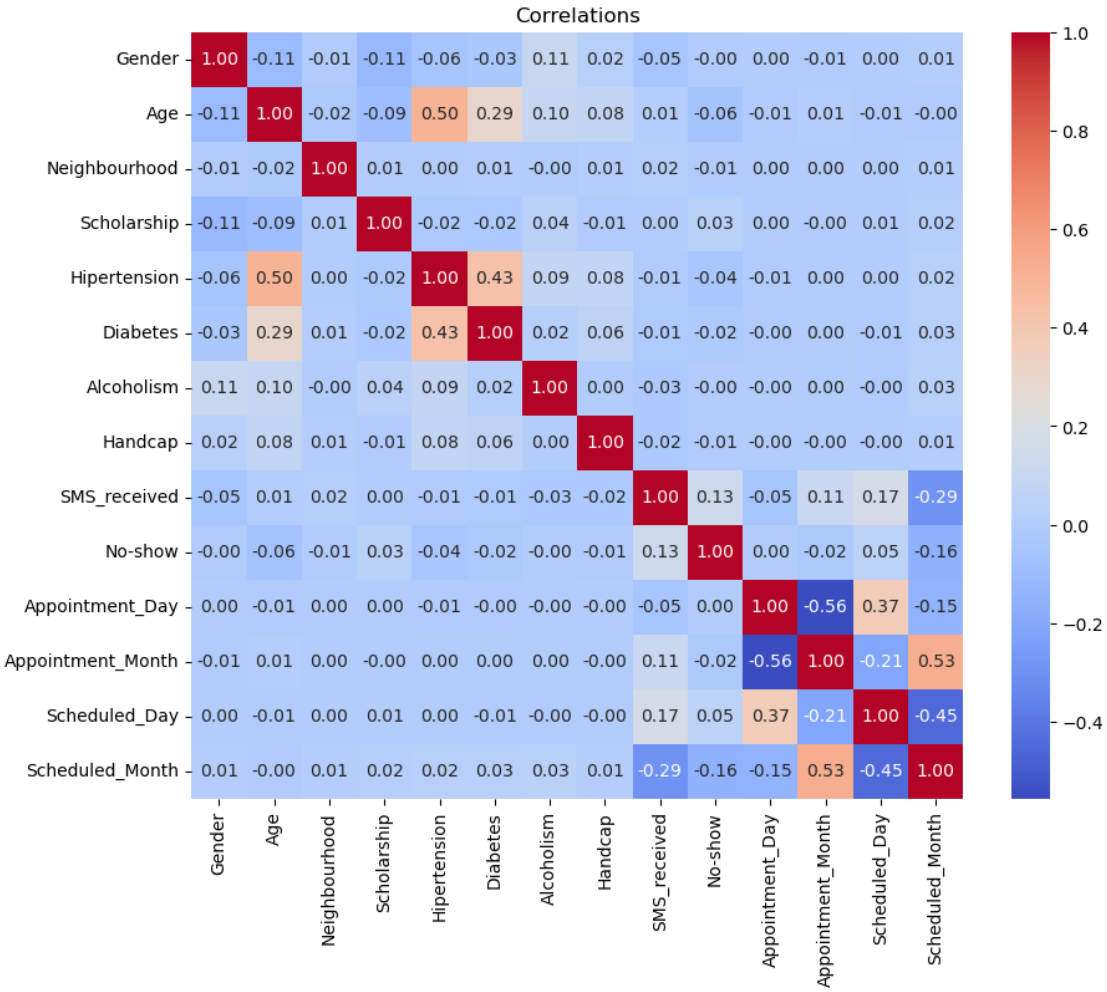
```
1              0          0                0          0                29
2              0          0                0          0                29

   Appointment_Month  Scheduled_Day  Scheduled_Month
0                  4             29                4
1                  4             29                4
2                  4             29                4
```

[25]:
```
#Factor Selection : Age, Sms, Schedule day, Schedule month
```

[26]:
```
corr_matrix = df.corr()
plt.figure(figsize = (10,8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlations")
plt.show
```

[26]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



[75]:
```
#Data Splitting
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110527 entries, 0 to 110526
Data columns (total 14 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Gender             110527 non-null  int64
 1   Age                110527 non-null  int64
 2   Neighbourhood      110527 non-null  int8
 3   Scholarship        110527 non-null  int64
 4   Hipertension       110527 non-null  int64
 5   Diabetes           110527 non-null  int64
 6   Alcoholism         110527 non-null  int64
 7   Handcap            110527 non-null  int64
 8   SMS_received       110527 non-null  int64
 9   No-show            110527 non-null  int64
 10  Appointment_Day    110527 non-null  int32
 11  Appointment_Month  110527 non-null  int32
 12  Scheduled_Day      110527 non-null  int32
 13  Scheduled_Month    110527 non-null  int32
dtypes: int32(4), int64(9), int8(1)
memory usage: 9.4 MB
```

[51]:
```
from sklearn.model_selection import train_test_split
x = df[["SMS_received","Scheduled_Day","Scheduled_Month","Age","Hipertension"]]
y = df["No-show"]
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.
 ↪2,random_state=27)
```

[52]:
```
# Neural Network Implementation
```

[53]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

[54]:
```
class BinaryClassifier(nn.Module):
    def __init__(self, input_dim):
        super(BinaryClassifier, self).__init__()
        self.fc1 = nn.Linear(input_dim, 16)
        self.fc2 = nn.Linear(16,16)
        self.fc3 = nn.Linear(16,8)
        self.output = nn.Linear(8,1)
```

```
    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = F.relu(self.fc3(x))
        x = F.sigmoid(self.output(x))
        return x
```

[55]:
```
input_dim = x_train.shape[1]
model = BinaryClassifier(input_dim)

pos_weight = torch.tensor([4.0])  # Tune based on class imbalance
criterion = nn.BCEWithLogitsLoss(pos_weight=pos_weight)
```

[56]:
```
x_train_tensor = torch.tensor(x_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32).view(-1, 1)

x_test_tensor = torch.tensor(x_test, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32).view(-1, 1)
```

[69]:
```
import time
import torch.optim as optim
optimizer = optim.Adam(model.parameters(), lr=0.001)
start_time = time.time()

epochs = 1000

for epoch in range(epochs):
    model.train()

    # Forward pass
    y_pred = model(x_train_tensor)
    loss = criterion(y_pred, y_train_tensor)

    # Backward pass
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()


    if (epoch+1) % 100 == 0:
        print(f"Epoch [{epoch+1}/{epochs}], Loss: {loss.item():.4f}")
end_time = time.time()
```

```
Epoch [100/1000], Loss: 1.0735
Epoch [200/1000], Loss: 1.0734
Epoch [300/1000], Loss: 1.0734
Epoch [400/1000], Loss: 1.0733
```

```
Epoch [500/1000], Loss: 1.0732
Epoch [600/1000], Loss: 1.0732
Epoch [700/1000], Loss: 1.0731
Epoch [800/1000], Loss: 1.0731
Epoch [900/1000], Loss: 1.0730
Epoch [1000/1000], Loss: 1.0729
```

[72]:
```
from sklearn.metrics import classification_report

model.eval()
with torch.no_grad():
    y_pred_test = model(x_test_tensor)
    y_pred_labels = (y_pred_test > 0.3).int()

print(classification_report(y_test_tensor, y_pred_labels))
```

```
              precision    recall  f1-score   support

         0.0       0.84      0.78      0.81     17628
         1.0       0.34      0.43      0.38      4478

    accuracy                           0.71     22106
   macro avg       0.59      0.61      0.60     22106
weighted avg       0.74      0.71      0.72     22106
```

[73]:
```
from sklearn.metrics import f1_score

# Binary classification: Use 'binary'
f1 = f1_score(y_test_tensor, y_pred_labels, average='binary')
print("F1 Score (binary):", f1)
```

```
F1 Score (binary): 0.3787745529170331
```

[ ]:
```

```