DSA312_Data_Science_with_Python

November 13, 2024

Group 3: Melvin Yong, Tan Hao Yang, Teo Jun Hao, Isaac Leong, Caleb Ang

Part 2B of Project

[1]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

[2]:
```
import numpy as np
import pandas as pd
import random
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,␣
 ↪recall_score, f1_score

# Set seeds for reproducibility for whole kernel
random.seed(888)
tf.random.set_seed(888)
np.random.seed(888)

df = pd.read_csv('lung cancer survey.csv')
df_no_na = df.dropna()

df_age = df_no_na[df_no_na["AGE"] > 21]

# Assuming 'age' is in the second column (index 0) of X
age_index = 1

X = df_age.drop('LUNG_CANCER', axis=1).values
y = df_age['LUNG_CANCER'].values

# Initialize the scaler
scaler = StandardScaler()

# Fit the scaler only on the age column in X_train
X_age = X[:, age_index].reshape(-1, 1)
```

```
scaler.fit(X_age)

# Transform the age column in X_train and X_val
X[:, age_index] = scaler.transform(X_age).flatten()

X



test = pd.read_csv('lung cancer survey_test.csv')
X_test = test.drop('LUNG_CANCER', axis = 1).values
y_test = test['LUNG_CANCER'].values

# Transform X_test age with X scaler
X_test_age = X_test[:, age_index].reshape(-1, 1)
X_test[:, age_index] = scaler.transform(X_test_age).flatten()
```

[2]:
```
StandardScaler()
```

[2]:
```
array([[ 0.        ,  0.02549734,  0.        , …,  1.        ,
         0.        ,  0.        ],
       [ 1.        ,  0.81953211,  1.        , …,  1.        ,
         0.        ,  0.        ],
       [ 1.        , -0.15095483,  0.        , …,  0.        ,
         0.        ,  1.        ],
       …,
       [ 1.        ,  0.20194951,  1.        , …,  1.        ,
         0.        ,  0.        ],
       [ 1.        ,  0.81953211,  1.        , …,  1.        ,
         0.        ,  1.        ],
       [ 1.        ,  0.11372343,  0.        , …,  0.        ,
         1.        ,  0.        ]])
```

[3]:
```
# Set seeds for reproducibility
random.seed(888)
tf.random.set_seed(888)
np.random.seed(888)

# Creating the model
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(15,)),  # Adjust input shape based on feature␣
 ↪count
    tf.keras.layers.Dense(8, activation='linear'),
    tf.keras.layers.Dense(23, activation='relu'),
    tf.keras.layers.Dense(8, activation='sigmoid'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
opt = tf.keras.optimizers.Adam(learning_rate=0.026526058661021926)
# Compile the model
model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])

# Train the model for 150 epochs (each epoch uses the full training set, i.e.␣
 ↪BGD)
model.fit(X, y, epochs=150, batch_size=len(X), verbose=0)
```

[3]:
```
<keras.src.callbacks.history.History at 0x1666e44a0>
```

[4]:
```
# Make predictions on the validation set
y_pred = (model.predict(X_test) > 0.5).astype("int32")  # Threshold at 0.5 for␣
 ↪binary classification

# Calculate F1 score
f1 = f1_score(y_test,y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("Best model Precision on validation set:", precision)
print("Best model Recall on validation set:", recall)
print("Best model F1 score on validation set:", f1)
```

```
32/32             0s 963us/step
Best model Precision on validation set: 0.9019607843137255
Best model Recall on validation set: 0.9975903614457832
Best model F1 score on validation set: 0.9473684210526315
```