```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, SimpleRNN
import tensorflow as tf

# Generate synthetic dataset (replace this with your actual data loading/preprocessing)
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=10000, n_features=20, n_classes=2, weights=[0.9, 0.1], random_state=42)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the RNN model
model = Sequential([
    SimpleRNN(units=64, input_shape=(X_train.shape[1], 1)),  # Assuming X_train is 2D
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Reshape input data (RNN expects 3D input)
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))

# Evaluate the model
y_pred_proba = model.predict(X_test)
y_pred = (y_pred_proba > 0.5).astype(int)  # Convert probabilities to binary predictions
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
Epoch 1/10
250/250 [==============================] - 7s 15ms/step - loss: 0.2177 - accuracy: 0.9162 - val_loss: 0.1951 - val_accuracy: 0.9110
Epoch 2/10
250/250 [==============================] - 3s 11ms/step - loss: 0.1773 - accuracy: 0.9308 - val_loss: 0.1522 - val_accuracy: 0.9375
Epoch 3/10
250/250 [==============================] - 3s 12ms/step - loss: 0.1675 - accuracy: 0.9327 - val_loss: 0.1461 - val_accuracy: 0.9375
Epoch 4/10
250/250 [==============================] - 3s 12ms/step - loss: 0.1641 - accuracy: 0.9380 - val_loss: 0.1421 - val_accuracy: 0.9405
Epoch 5/10
250/250 [==============================] - 2s 10ms/step - loss: 0.1608 - accuracy: 0.9364 - val_loss: 0.1422 - val_accuracy: 0.9355
Epoch 6/10
250/250 [==============================] - 2s 6ms/step - loss: 0.1581 - accuracy: 0.9381 - val_loss: 0.1385 - val_accuracy: 0.9365
Epoch 7/10
250/250 [==============================] - 2s 6ms/step - loss: 0.1560 - accuracy: 0.9395 - val_loss: 0.1367 - val_accuracy: 0.9340
Epoch 8/10
250/250 [==============================] - 2s 9ms/step - loss: 0.1537 - accuracy: 0.9376 - val_loss: 0.1352 - val_accuracy: 0.9340
Epoch 9/10
250/250 [==============================] - 2s 6ms/step - loss: 0.1496 - accuracy: 0.9410 - val_loss: 0.1419 - val_accuracy: 0.9350
Epoch 10/10
250/250 [==============================] - 1s 6ms/step - loss: 0.1503 - accuracy: 0.9416 - val_loss: 0.1400 - val_accuracy: 0.9335
63/63 [==============================] - 0s 3ms/step
Accuracy: 0.9335
Precision: 0.8321167883211679
Recall: 0.5089285714285714
F1 Score: 0.631578947368421
```