```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from sklearn.metrics import confusion_matrix
from sklearn.metrics import recall_score, precision_score, f1_score

# Load the CSV file into a Pandas DataFrame
df = pd.read_csv('machine_failure_cleaned.csv')

# Extract the features and target variable
features = df.iloc[:, :-1].values
target = df.iloc[:, -1].values

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

# Normalize the features
X_train = (X_train - np.mean(X_train, axis=0)) / np.std(X_train, axis=0)
X_test = (X_test - np.mean(X_test, axis=0)) / np.std(X_test, axis=0)

#print(df)
#print(df.iloc[0,:])
```



```
# Create an ANN with 3 layers
model = Sequential()

# First layer with 6 neurons
model.add(Dense(units=6, activation='relu', input_shape=(X_train.shape[1],)))

# Second layer with 4 neurons and a 0.2 dropout
model.add(Dense(units=4, activation='relu'))
model.add(Dropout(0.2))

# Last layer with 1 neuron and binary classification output
model.add(Dense(units=1, activation='sigmoid'))

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

# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=32)

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print('Loss:', loss)
print('Accuracy:', accuracy)
```

```
246/246 [==============================] - 1s 3ms/step - loss: 0.0282 - accuracy: 0.9926
Epoch 89/100
246/246 [==============================] - 1s 5ms/step - loss: 0.0268 - accuracy: 0.9927
Epoch 90/100
246/246 [==============================] - 2s 9ms/step - loss: 0.0251 - accuracy: 0.9940
Epoch 91/100
246/246 [==============================] - 2s 9ms/step - loss: 0.0301 - accuracy: 0.9911
Epoch 92/100
246/246 [==============================] - 1s 4ms/step - loss: 0.0264 - accuracy: 0.9931
Epoch 93/100
246/246 [==============================] - 1s 4ms/step - loss: 0.0260 - accuracy: 0.9934
Epoch 94/100
246/246 [==============================] - 1s 3ms/step - loss: 0.0246 - accuracy: 0.9941
Epoch 95/100
246/246 [==============================] - 1s 5ms/step - loss: 0.0243 - accuracy: 0.9940
Epoch 96/100
246/246 [==============================] - 2s 8ms/step - loss: 0.0251 - accuracy: 0.9934
Epoch 97/100
246/246 [==============================] - 2s 10ms/step - loss: 0.0248 - accuracy: 0.9936
Epoch 98/100
246/246 [==============================] - 1s 5ms/step - loss: 0.0248 - accuracy: 0.9936
Epoch 99/100
246/246 [==============================] - 1s 5ms/step - loss: 0.0269 - accuracy: 0.9930
Epoch 100/100
246/246 [==============================] - 1s 4ms/step - loss: 0.0290 - accuracy: 0.9917
62/62 [==============================] - 0s 3ms/step - loss: 0.0048 - accuracy: 0.9995
Loss: 0.004784504882991314
Accuracy: 0.9994905591011047
```

```
# Predict the labels for the test set
y_pred = model.predict(X_test)

# Convert the predictions to binary values (0 or 1)
y_pred_binary = [int(round(p[0])) for p in y_pred]

# Get the confusion matrix
cm = confusion_matrix(y_test, y_pred_binary)
```

```
62/62 [==============================] - 0s 5ms/step
```

```
# Calculate recall, precision and F score
recall = recall_score(y_test, y_pred_binary)
precision = precision_score(y_test, y_pred_binary)
f1 = f1_score(y_test, y_pred_binary)

# Print the confusion matrix with recall, precision and F score
print("Confusion Matrix:")
print(cm)
print("\nRecall:", recall)
print("Precision:", precision)
print("F1 Score:", f1)

print("\nThe confusion matrix shows that the model achieves very high precision recall, which is perfect.")
print("This shows that the model is extremely effective at correctly identifying both positive and negative cases with very low error.")
```

```
Confusion Matrix:
[[1911    0]
 [   1   51]]

Recall: 0.9807692307692307
Precision: 1.0
F1 Score: 0.9902912621359222

The confusion matrix shows that the model achieves very high precision recall, which is perfect.
This shows that the model is extremely effective at correctly identifying both positive and negative cases with very low error.
```