```python
# Use the Magic Gamma Telescope database:
 # • Apply an ANN with 1 input layer, 2 hidden layers, and 1 output layer.
 # • Report the performance using various activation functions:
 # 1. Sigmoid
 # 2. Tanh
 # 3. ReLU
 # 4. Leaky ReLU

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import accuracy_score, classification_report
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Load the Magic Gamma Telescope dataset
data_url = "https://archive.ics.uci.edu/ml/machine-learning-databases/magic/magic04.data"
columns = ["fLength", "fWidth", "fSize", "fConc", "fConc1", "fAsym", "fM3Long", "fM3Trans",
"fAlpha", "fDist", "class"]
data = pd.read_csv(data_url, header=None, names=columns)

# Encode the target variable ('g' for gamma, 'h' for hadron)
label_encoder = LabelEncoder()
data['class'] = label_encoder.fit_transform(data['class'])

# Split the dataset into features and target
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
```

```
# Standardize the feature data
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, stratify=y)

# Function to build, train, and evaluate the model
def evaluate_model(activation_function):
    # Build the ANN model
    model = Sequential([
        Dense(64, input_dim=X_train.shape[1], activation=activation_function),
        Dense(32, activation=activation_function),
        Dense(1, activation='sigmoid')
    ])

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

    # Train the model
    model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=0)

    # Evaluate the model on test data
    y_pred = (model.predict(X_test) > 0.5).astype(int)
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred, target_names=label_encoder.classes_)

    print(f"Activation Function: {activation_function}")
    print(f"Accuracy: {accuracy:.4f}")
    print(report)
    print("=" * 50)
```

```python
# Evaluate the model with different activation functions
for activation in ['sigmoid', 'tanh', 'relu', 'leaky_relu']:
    if activation == 'leaky_relu':
        # Use Leaky ReLU as a layer instead of string
        model = Sequential([
            Dense(64, input_dim=X_train.shape[1]),
            tf.keras.layers.LeakyReLU(alpha=0.01),
            Dense(32),
            tf.keras.layers.LeakyReLU(alpha=0.01),
            Dense(1, activation='sigmoid')
        ])
        model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
        model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=0)
        y_pred = (model.predict(X_test) > 0.5).astype(int)
        accuracy = accuracy_score(y_test, y_pred)
        report = classification_report(y_test, y_pred, target_names=label_encoder.classes_)

        print(f"Activation Function: Leaky ReLU")
        print(f"Accuracy: {accuracy:.4f}")
        print(report)
        print("=" * 50)
    else:
        evaluate_model(activation)
```