In [ ]:

```
pip install tensorflow
```

In [ ]:

```
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Generate synthetic data for binary classification
X = np.random.rand(1000, 10)  # Example feature matrix
y = np.random.randint(2, size=1000)  # Example binary labels

# Split the data 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)

# Build the classification model
model_classification = tf.keras.Sequential([
    tf.keras.layers.Dense(32, activation='relu', input_shape=(10,)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(16, activation='tanh'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

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

# Train the model
model_classification.fit(X_train, y_train, epochs=10, batch_size=32, verbose=2)

# Evaluate the model
y_pred = (model_classification.predict(X_test) > 0.5).astype(int)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

In [ ]:

```
# Generate synthetic data for regression
X = np.random.rand(1000, 10)  # Example feature matrix
y = np.random.rand(1000)  # Example regression targets

# Split the data 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)

# Build the regression model
model_regression = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(10,)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(32, activation='linear'),
    tf.keras.layers.Dense(1, activation='linear')
])

# Compile the model with optimizer and loss function for regression
model_regression.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model_regression.fit(X_train, y_train, epochs=10, batch_size=32, verbose=2)

# Evaluate the model
y_pred = model_regression.predict(X_test)
# You can use metrics like Mean Absolute Error (MAE) or Mean Squared Error (MSE) to evaluate regression performance
mae = np.mean(np.abs(y_test - y_pred))
mse = np.mean((y_test - y_pred) ** 2)
print(f"MAE: {mae}, MSE: {mse}")
```

In [ ]: