Importing Libraries

```
import numpy as np
import tensorflow as tf
import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

np.random.seed(42)
X = np.random.randn(10000, 3)  # 1000 samples, 3 features
y = (np.sin(X[:, 0]) + np.cos(X[:, 1]) + X[:, 2] > 0).astype(int)  # Binary classification
```

Model Building

```
# Step 2: Preprocess Data
# Standardize features
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Split the data into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.4, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# Convert to TensorFlow datasets
train_dataset = tf.data.Dataset.from_tensor_slices((X_train, y_train)).shuffle(1000).batch(32)
val_dataset = tf.data.Dataset.from_tensor_slices((X_val, y_val)).batch(32)
test_dataset = tf.data.Dataset.from_tensor_slices((X_test, y_test)).batch(32)

# Step 3: Define the Model
model = keras.Sequential([
    keras.layers.Dense(10, activation='relu', input_shape=(X_train.shape[1],)),
    keras.layers.Dense(8, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

```

```
1/30
local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argume
er().__init__(activity_regularizer=activity_regularizer, **kwargs)
8 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.5884 - loss: 0.6784 - val_accuracy: 0.8655 - val_loss: 0.4509
2/30
8 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8754 - loss: 0.4046 - val_accuracy: 0.8805 - val_loss: 0.3059
3/30
8 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8899 - loss: 0.2957 - val_accuracy: 0.8990 - val_loss: 0.2421
4/30
8 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9069 - loss: 0.2313 - val_accuracy: 0.9255 - val_loss: 0.1955
5/30
8 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9287 - loss: 0.1899 - val_accuracy: 0.9450 - val_loss: 0.1618
6/30
8 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9482 - loss: 0.1570 - val_accuracy: 0.9530 - val_loss: 0.1425
7/30
8 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9579 - loss: 0.1299 - val_accuracy: 0.9605 - val_loss: 0.1279
8/30
8 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9629 - loss: 0.1268 - val_accuracy: 0.9595 - val_loss: 0.1227
9/30
8 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9648 - loss: 0.1151 - val_accuracy: 0.9640 - val_loss: 0.1136
10/30
8 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9675 - loss: 0.1085 - val_accuracy: 0.9655 - val_loss: 0.1092
11/30
8 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9698 - loss: 0.1058 - val_accuracy: 0.9665 - val_loss: 0.1064
12/30
8 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9720 - loss: 0.0984 - val_accuracy: 0.9655 - val_loss: 0.1040
13/30
8 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9734 - loss: 0.0951 - val_accuracy: 0.9690 - val_loss: 0.1010
14/30
8 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9741 - loss: 0.0947 - val_accuracy: 0.9675 - val_loss: 0.0996
15/30
8 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9732 - loss: 0.0924 - val_accuracy: 0.9685 - val_loss: 0.0984
16/30
8 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9720 - loss: 0.0962 - val_accuracy: 0.9690 - val_loss: 0.0966
17/30
8 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9720 - loss: 0.0877 - val_accuracy: 0.9695 - val_loss: 0.0982
18/30
8 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9747 - loss: 0.0887 - val_accuracy: 0.9700 - val_loss: 0.0956
19/30
8 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9760 - loss: 0.0825 - val_accuracy: 0.9715 - val_loss: 0.0935
20/30
8 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9780 - loss: 0.0821 - val_accuracy: 0.9715 - val_loss: 0.0927
```

```
21/30
38 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9752 - loss: 0.0856 - val_accuracy: 0.9745 - val_loss: 0.0905
22/30
38 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9811 - loss: 0.0770 - val_accuracy: 0.9750 - val_loss: 0.0890
23/30
38 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9753 - loss: 0.0801 - val_accuracy: 0.9730 - val_loss: 0.0897
24/30
38 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9790 - loss: 0.0794 - val_accuracy: 0.9740 - val_loss: 0.0892
25/30
38 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9788 - loss: 0.0766 - val_accuracy: 0.9770 - val_loss: 0.0866
26/30
38 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9779 - loss: 0.0741 - val_accuracy: 0.9745 - val_loss: 0.0864
27/30
38 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9775 - loss: 0.0769 - val_accuracy: 0.9750 - val_loss: 0.0861
```

Compile and Train

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

# Step 4: Train the Model
history = model.fit(train_dataset,
                    validation_data=val_dataset,
                    epochs=30)
```

Plotting

```
# Step 5: Evaluate the Model
test_loss, test_accuracy = model.evaluate(test_dataset)
print(f'Test Loss: {test_loss}')
print(f'Test Accuracy: {test_accuracy}')

import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9806 - loss:
Test Loss: 0.07635298371315002
Test Accuracy: 0.9785000085830688
```