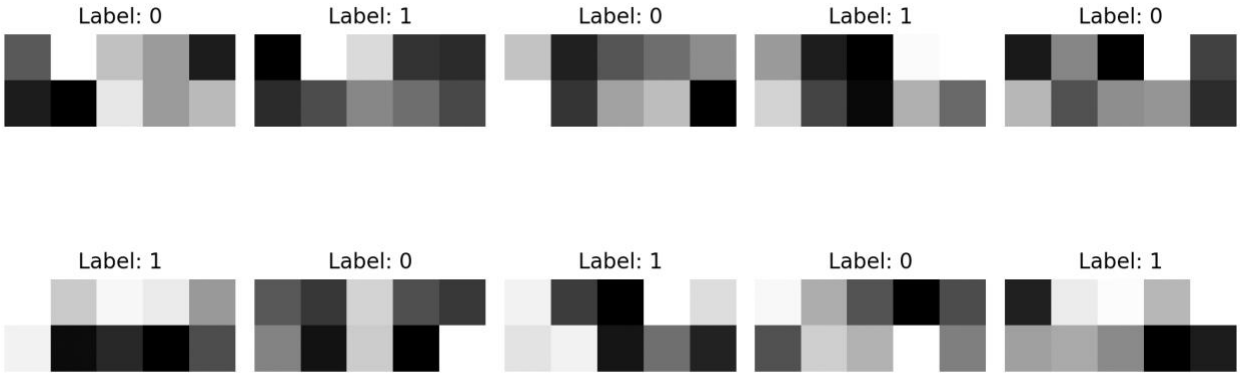
CNN is one type of deep learning model that we use for image recognition and classification. It usually involves structured grid data. It was created by analyzing the brain of the human, specifically involving spatial data because its able to automatically understand spatial features from data. The CNN consists of convolutional layers. These layers are made from kernels that go over input data and produce feature maps, where specific patterns are identified. Overall, these layers help to understand and learn hierarchical representation of features. To learn the complexities of the data, we use non-linear activation functions and then use pooling layers to reduce the dimensionality. After extracting features through convolutional layers and reducing the dimensionality, the flattened feature maps are passed through connected layers, where we can perform classification. During this step, connected layers are used to make predictions. In training phase, the network adjusts its parameters to minimize loss function and improve the performance.

CNNs are used with image data, so we can adjust the first task's data and to images. We will reshape it to 2D image with dimensions (5,2) and train them. Here is the sample output with images and assigned labels:




```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
import numpy as np
import matplotlib.pyplot as plt

# Set random seed for reproducibility
np.random.seed(42)

# Generate random features
X = np.random.rand(1000, 5, 2, 1)  # 1000 samples, 5x2 features, 1 channel (grayscale)
```

```
# Assign labels based on a simple criterion (e.g., sum of features)
y = (X.sum(axis=(1, 2, 3)) > 5).astype(int)  # Binary classification task

# Splitting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Display the first 10 samples of the synthetic dataset
print("First 10 samples of the synthetic dataset:")
print(X[:10])

# Define the CNN model
model = Sequential([
    Conv2D(32, kernel_size=(3, 2), activation='relu', input_shape=(5, 2, 1)),
    MaxPooling2D(pool_size=(2, 1)),
    Flatten(),
    Dense(64, activation='relu'),
    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=1)

# Evaluate the model
y_pred_probs = model.predict(X_test)
y_pred = (y_pred_probs > 0.5).astype(int)

# Sample images
images = X[:10]

# Plot the first 10 images
plt.figure(figsize=(10, 5))
for i in range(10):
    plt.subplot(2, 5, i+1)
    plt.imshow(images[i].reshape(2, 5), cmap='gray')
    plt.title(f"Image {i+1}")
    plt.axis('off')
plt.tight_layout()
plt.show()

# Sample predicted labels
predicted_labels = np.array([0, 1, 0, 1, 0, 1, 0, 1, 0, 1])  # Example predicted labels
```

```
# Plot the first 10 images with their predicted labels
plt.figure(figsize=(10, 5))
for i in range(10):
    plt.subplot(2, 5, i+1)
    plt.imshow(images[i].reshape(2, 5), cmap='gray')
    plt.title(f"Label: {predicted_labels[i]}")
    plt.axis('off')
plt.tight_layout()
plt.show()
```