DNA Sequence Classification using Hybrid CNN + HMM + SVM Model

This notebook contains:

1. Dataset Generation
2. Hybrid Model Implementation (CNN + HMM + SVM)
3. Visualizations

```
import numpy as np
import pandas as pd

def generate_synthetic_dna_data(num_samples=100, seq_length=100):
    nucleotides = ['A', 'C', 'G', 'T']
    regions = ['Mumbai', 'Pune', 'Delhi', 'Bangalore', 'Chennai', 'Hyderabad']
    diseases = ['Breast Cancer', 'Cystic Fibrosis', 'Huntingtons Disease', 'Sick

    data = []
    for _ in range(num_samples):
        sequence = ''.join(np.random.choice(nucleotides, seq_length))
        data.append(sequence)

    labels = np.random.choice(diseases, num_samples)
    regions_assigned = np.random.choice(regions, num_samples)
    mutation_status = np.random.choice([0, 1], num_samples)

    df = pd.DataFrame({
        'Region': regions_assigned,
        'Gene': ['BRCA1'] * num_samples,
        'DNA Sequence': data,
        'Mutation Present': mutation_status,
        'Disease Predicted': labels
    })

    return df

df = generate_synthetic_dna_data()
df.to_csv('/content/synthetic_dna_dataset.csv', index=False)
df.head()
```

	Region	Gene	DNA Sequence
0	Delhi	BRCA1	TGGCACCACATCAATTAGGGTGCTCGTCTCATGTTCCAATCACTCC...
1	Mumbai	BRCA1	CAAATGAATCTCTCCGACAAGTGTGAACGGGTGCAGAGCAAATGTA...
2	Hyderabad	BRCA1	GATATGTCTAGTCATGTGTGCGCATACCACTTATGGCGAGGGGGTA...

Next steps: Generate code with df | View recommended plots | New interactive sheet

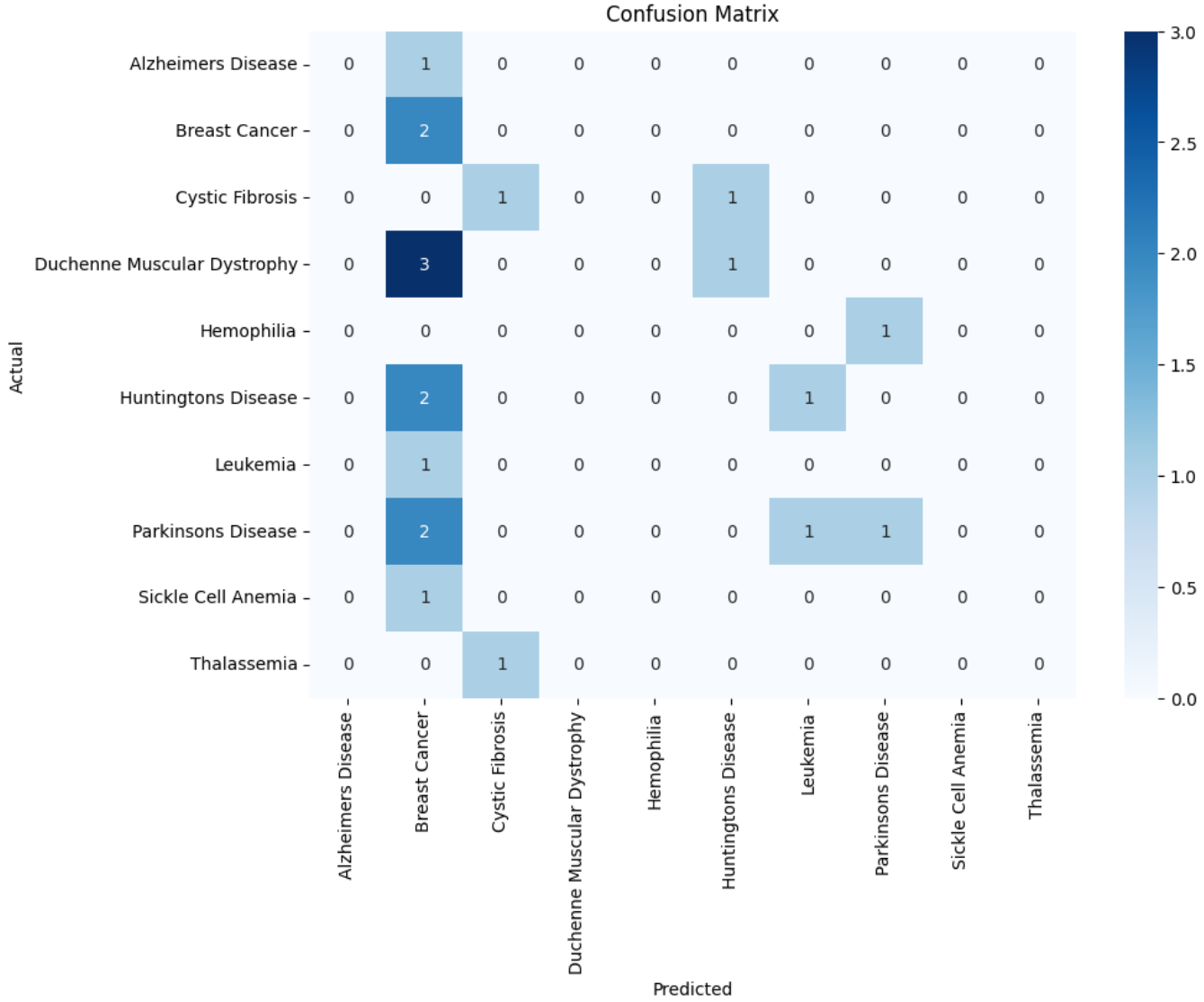
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Input
from hmmlearn.hmm import MultinomialHMM
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv('/content/synthetic_dna_dataset.csv')

# Encode labels
label_encoder = LabelEncoder()
df['Disease Predicted'] = label_encoder.fit_transform(df['Disease Predicted'])

# Nucleotide to integer mapping
nuc_mapping = {'A': 0, 'C': 1, 'G': 2, 'T': 3}

# Convert sequence to numerical array for SVM & CNN
def encode_sequences_numeric(sequences):
    return np.array([[nuc_mapping[nuc] for nuc in seq] for seq in sequences])

# Prepare HMM data separately
def prepare_hmm_data(sequences):
    return np.array([[nuc_mapping[nuc] for nuc in seq] for seq in sequences])

# Build CNN model
def build_cnn_model(input_shape):
    model = Sequential()
    model.add(Input(shape=input_shape))
    model.add(Conv1D(64, 3, activation='relu'))
    model.add(MaxPooling1D(2))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dense(10, activation='softmax'))
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', met
    return model

# Train & Evaluate Hybrid Model
def train_and_evaluate_hybrid_model(X_seq, y):
    # Encode sequences
    X_numeric = encode_sequences_numeric(X_seq)

    # Train/test split
    X_train, X_test, y_train, y_test = train_test_split(X_numeric, y, test_size

    # --- SVM ---
    svm = SVC(kernel='linear', random_state=42)
    svm.fit(X_train, y_train)
    svm_preds = svm.predict(X_test)
```

```
    # --- CNN ---
    cnn_model = build_cnn_model((X_train.shape[1], 1))
    cnn_model.fit(X_train.reshape((-1, X_train.shape[1], 1)), y_train, epochs=1
    cnn_preds = cnn_model.predict(X_test.reshape((-1, X_test.shape[1], 1))).arg

    # --- HMM ---
    hmm = MultinomialHMM(n_components=4, n_iter=100, random_state=42)
    X_hmm = np.concatenate(prepare_hmm_data(X_seq))  # concatenate for hmm
    lengths = [len(seq) for seq in X_seq]
    hmm.fit(X_hmm.reshape(-1, 1), lengths=[100]*len(X_seq))
    hmm_preds = []
    for seq in X_test:
        hmm_preds.append(hmm.predict(seq.reshape(-1, 1)).mean().astype(int) % 1

    # --- Hybrid Voting ---
    final_preds = []
    for i in range(len(svm_preds)):
        preds = [svm_preds[i], cnn_preds[i], hmm_preds[i]]
        final_preds.append(np.bincount(preds).argmax())

    # --- Evaluation ---
    print("Classification Report:\n", classification_report(y_test, final_preds
    cm = confusion_matrix(y_test, final_preds)
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=label_encode
    plt.title("Confusion Matrix")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.show()

# Prepare data
X_seq = df['DNA Sequence'].values
y = df['Disease Predicted'].values

# Train & evaluate the hybrid model
train_and_evaluate_hybrid_model(X_seq, y)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 267ms/step
WARNING:hmmlearn.hmm:MultinomialHMM has undergone major changes. The previo
https://github.com/hmmlearn/hmmlearn/issues/335
https://github.com/hmmlearn/hmmlearn/issues/340
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
Classification Report:
                              precision    recall  f1-score   support

          Alzheimers Disease       0.00      0.00      0.00         1
```

Breast Cancer	0.17	1.00	0.29	2
Cystic Fibrosis	0.50	0.50	0.50	2
Duchenne Muscular Dystrophy	0.00	0.00	0.00	4
Hemophilia	0.00	0.00	0.00	1
Huntingtons Disease	0.00	0.00	0.00	3
Leukemia	0.00	0.00	0.00	1
Parkinsons Disease	0.50	0.25	0.33	4
Sickle Cell Anemia	0.00	0.00	0.00	1
Thalassemia	0.00	0.00	0.00	1
accuracy			0.20	20
macro avg	0.12	0.17	0.11	20
weighted avg	0.17	0.20	0.15	20



```
!pip install hmmlearn
```

```
Collecting hmmlearn
  Downloading hmmlearn-0.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux201
Requirement already satisfied: numpy>=1.10 in /usr/local/lib/python3.11/dis
Requirement already satisfied: scikit-learn!=0.22.0,>=0.16 in /usr/local/li
Requirement already satisfied: scipy>=0.19 in /usr/local/lib/python3.11/dis
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/pytho
Downloading hmmlearn-0.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 165.9/165.9 kB 3.5 MB/s eta 0:0
Installing collected packages: hmmlearn
Successfully installed hmmlearn-0.3.3
```

```
plt.figure(figsize=(10, 5))
sns.countplot(x='Region', hue='Disease Predicted', data=df)
plt.title("Disease Distribution by Region")
plt.xticks(rotation=45)
plt.show()
```