Financial Fraud Detection:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import shap
import joblib

from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, LSTM, Dense, Flatten, Dropout,
BatchNormalization, Input
from tensorflow.keras.optimizers import Adam

# □ Load dataset
df = pd.read_csv("creditcard.csv")

# □ Data Preprocessing
X = df.drop(columns=["Class"])
y = df["Class"]

# □ Apply SMOTE to balance fraud cases
smote = SMOTE(sampling_strategy=0.3, random_state=42)  # Increase fraud cases to 30% of
normal transactions
X_resampled, y_resampled = smote.fit_resample(X, y)

# □ Split Data
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2,
random_state=42)

# □ Train Optimized RandomForest Model
rf_model = RandomForestClassifier(
    n_estimators=200,  # Increased estimators
    max_depth=15,      # Prevents overfitting
    min_samples_split=5,
    min_samples_leaf=2,
    random_state=42,
    n_jobs=-1
)
rf_model.fit(X_train, y_train)

# □ Predictions & Adjusted Threshold
```

```
y_pred_prob = rf_model.predict_proba(X_test)[:, 1]  # Get probabilities
threshold = 0.3  # Lower threshold for better recall
y_pred = (y_pred_prob > threshold).astype(int)

# □ Evaluate Model
print(classification_report(y_test, y_pred))
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# □ SHAP Explanation (Optimized)
explainer = shap.TreeExplainer(rf_model)
shap_values = explainer.shap_values(X_test.iloc[:500])  # Ensure matching shapes

# □ Ensure SHAP Summary Plot Displays Correctly
if isinstance(shap_values, list):  # For classification models
    shap.summary_plot(shap_values[1], X_test.iloc[:500])
else:
    shap.summary_plot(shap_values, X_test.iloc[:500])

plt.show()

# □ Optimized CNN-LSTM Model
X_train_cnn = np.expand_dims(X_train, axis=2)  # Reshape for CNN
X_test_cnn = np.expand_dims(X_test, axis=2)

model = Sequential([
    Input(shape=(X_train.shape[1], 1)),
    Conv1D(filters=64, kernel_size=3, activation='relu'),
    BatchNormalization(),
    LSTM(50, return_sequences=True),
    Dropout(0.3),
    LSTM(50),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer=Adam(learning_rate=0.0005), loss="binary_crossentropy",
metrics=["accuracy"])
history = model.fit(X_train_cnn, y_train, epochs=5, batch_size=128,
validation_data=(X_test_cnn, y_test))

# □ Save Model
joblib.dump(rf_model, "fraud_detection_rf.pkl")
```