Input

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import MinMaxScaler

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Dropout, Conv2D, MaxPooling2D,
Flatten

from sklearn.ensemble import RandomForestClassifier

import matplotlib.pyplot as plt

import cv2

import joblib  # For saving Random Forest model


# Load the dataset

df = pd.read_csv('Insurance Fraud.csv')


# Drop unnecessary columns

drop_columns = ["Claim ID", "Claim ID.1", "Street Address", "Claimant Name", "City",
"State", "Country", "Postal Code"]

df_cleaned = df.drop(columns=drop_columns)


# Convert Claim Date to datetime and sort by date

df_cleaned["Claim Date"] = pd.to_datetime(df_cleaned["Claim Date"], format="%d-
%m-%Y")

df_cleaned = df_cleaned.sort_values(by="Claim Date")


# One-hot encode categorical variables
```

```
df_cleaned = pd.get_dummies(df_cleaned, columns=["Claim Status", "Type of Insurance Claim", "Fraud_Types"], drop_first=True)

# Set Claim Date as index and resample data daily
df_time_series = df_cleaned.set_index("Claim Date").resample("D").mean().fillna(0)

# Convert SuspiciousFlag to binary
df_time_series["SuspiciousFlag"] = (df_time_series["SuspiciousFlag"] >= 0.5).astype(int)

# Reset index
df_time_series.reset_index(inplace=True)

# Drop Claim Date for model input
df_time_series.drop(columns=["Claim Date"], inplace=True)

# Define input and target variables
X = df_time_series.drop(columns=["SuspiciousFlag"] + [col for col in df_time_series.columns if "Fraud_Types" in col]).values
y_suspicious = df_time_series["SuspiciousFlag"].values
y_fraud_type = df_time_series[[col for col in df_time_series.columns if "Fraud_Types" in col]].values

# Normalize input features
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Convert data into sequences for LSTM
sequence_length = 10
```

```
X_lstm, y_suspicious_lstm, y_fraud_type_lstm = [], [], []

for i in range(len(X_scaled) - sequence_length):
    X_lstm.append(X_scaled[i : i + sequence_length])
    y_suspicious_lstm.append(y_suspicious[i + sequence_length])
    y_fraud_type_lstm.append(y_fraud_type[i + sequence_length])

X_lstm = np.array(X_lstm)
y_suspicious_lstm = np.array(y_suspicious_lstm)
y_fraud_type_lstm = np.array(y_fraud_type_lstm)

# Train-test split (80-20) and validation split (10% of training data)
X_train, X_test, y_train_susp, y_test_susp, y_train_fraud, y_test_fraud = train_test_split(
    X_lstm, y_suspicious_lstm, y_fraud_type_lstm, test_size=0.2, random_state=42,
stratify=y_suspicious_lstm
)

X_train, X_val, y_train_susp, y_val_susp, y_train_fraud, y_val_fraud = train_test_split(
    X_train, y_train_susp, y_train_fraud, test_size=0.1, random_state=42,
stratify=y_train_susp
)

# LSTM Model
lstm_model = Sequential([
    LSTM(64, return_sequences=True, input_shape=(sequence_length, X_train.shape[2])),
    Dropout(0.2),
    LSTM(32, return_sequences=False),
    Dropout(0.2),
    Dense(16, activation='relu'),
```

```
    Dense(1, activation='sigmoid')
])

lstm_model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

lstm_model.fit(X_train, y_train_susp, validation_data=(X_val, y_val_susp), epochs=50,
batch_size=32, verbose=1)

# Save LSTM Model
lstm_model.save('main_lstm_model.h5')

# Evaluate LSTM Model
train_acc_lstm = lstm_model.evaluate(X_train, y_train_susp, verbose=0)[1] * 100
val_acc_lstm = lstm_model.evaluate(X_val, y_val_susp, verbose=0)[1] * 100
test_acc_lstm = lstm_model.evaluate(X_test, y_test_susp, verbose=0)[1] * 100

print(f"LSTM Model - Train Accuracy: {train_acc_lstm:.2f}%")
print(f"LSTM Model - Validation Accuracy: {val_acc_lstm:.2f}%")
print(f"LSTM Model - Test Accuracy: {test_acc_lstm:.2f}%")

# Train Random Forest Model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
X_train_rf = X_train.reshape(X_train.shape[0], -1)
X_test_rf = X_test.reshape(X_test.shape[0], -1)
rf_model.fit(X_train_rf, y_train_susp)

# Save Random Forest Model
joblib.dump(rf_model, 'random_forest_model.pkl')
```

```
# Evaluate Random Forest Model
train_acc_rf = rf_model.score(X_train_rf, y_train_susp) * 100
test_acc_rf = rf_model.score(X_test_rf, y_test_susp) * 100

print(f"Random Forest Model - Train Accuracy: {train_acc_rf:.2f}%")
print(f"Random Forest Model - Test Accuracy: {test_acc_rf:.2f}%")

# Convert LSTM and RF outputs into grayscale images
def convert_to_image(data, size=(28, 28)):
    data_resized = np.resize(data, size)
    image = (data_resized * 255).astype(np.uint8)
    return image

X_lstm_images = np.array([convert_to_image(x) for x in X_train_rf])
X_rf_images = np.array([convert_to_image(x) for x in X_train_rf])

X_combined_images = np.stack([X_lstm_images, X_rf_images], axis=-1)

# CNN Model
cnn_model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 2)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
```

```
])

cnn_model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

cnn_model.fit(X_combined_images, y_train_susp, validation_split=0.1, epochs=100,
batch_size=32, verbose=1)

# Save CNN Model
cnn_model.save('main_cnn_model.h5')

# Evaluate CNN Model
train_acc_cnn = cnn_model.evaluate(X_combined_images, y_train_susp, verbose=0)[1]
* 100
val_acc_cnn = cnn_model.evaluate(X_combined_images[:len(y_train_susp) // 10],
                        y_train_susp[:len(y_train_susp) // 10], verbose=0)[1] * 100
test_acc_cnn = cnn_model.evaluate(X_combined_images, y_train_susp, verbose=0)[1]
* 100  # Replace with actual test images

print(f"CNN Model - Train Accuracy: {train_acc_cnn:.2f}%")
print(f"CNN Model - Validation Accuracy: {val_acc_cnn:.2f}%")
print(f"CNN Model - Test Accuracy: {test_acc_cnn:.2f}%")
```

CODE RUNNING

```
PS C:\Users\Ansh Gupta\Desktop\Project SBI> python -u "c:\Users\Ansh Gupta\Desktop\Project SBI\SBI_10.py"
2025-02-23 22:13:53.374393: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different
numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environmen
t variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-02-23 22:13:54.114027: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different
numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environmen
t variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-02-23 22:13:57.270067: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use avai
lable CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate
 compiler flags.
C:\Users\Ansh Gupta\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning: Do n
ot pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as
 the first layer in the model instead.
  super().__init__(**kwargs)
Epoch 1/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - accuracy: 0.8013 - loss: 0.5570 - val_accuracy: 0.7877 - val_loss: 0.5230
Epoch 2/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7733 - loss: 0.5336 - val_accuracy: 0.7877 - val_loss: 0.5169
Epoch 3/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7857 - loss: 0.5259 - val_accuracy: 0.7877 - val_loss: 0.5259
Epoch 4/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7872 - loss: 0.5240 - val_accuracy: 0.7877 - val_loss: 0.5172
Epoch 5/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7941 - loss: 0.5072 - val_accuracy: 0.7877 - val_loss: 0.5174
Epoch 6/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7821 - loss: 0.5278 - val_accuracy: 0.7877 - val_loss: 0.5178
Epoch 7/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8064 - loss: 0.4963 - val_accuracy: 0.7877 - val_loss: 0.5173
Epoch 8/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8034 - loss: 0.4995 - val_accuracy: 0.7877 - val_loss: 0.5177
Epoch 9/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7929 - loss: 0.5144 - val_accuracy: 0.7877 - val_loss: 0.5183
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8034 - loss: 0.4995 - val_accuracy: 0.7877 - val_loss: 0.5177
Epoch 9/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7929 - loss: 0.5144 - val_accuracy: 0.7877 - val_loss: 0.5183
Epoch 10/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7740 - loss: 0.5400 - val_accuracy: 0.7877 - val_loss: 0.5222
Epoch 11/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7762 - loss: 0.5418 - val_accuracy: 0.7877 - val_loss: 0.5189
Epoch 12/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7821 - loss: 0.5277 - val_accuracy: 0.7877 - val_loss: 0.5171
Epoch 13/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7824 - loss: 0.5287 - val_accuracy: 0.7877 - val_loss: 0.5173
Epoch 14/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8019 - loss: 0.4997 - val_accuracy: 0.7877 - val_loss: 0.5188
Epoch 15/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7999 - loss: 0.5071 - val_accuracy: 0.7877 - val_loss: 0.5170
Epoch 16/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7998 - loss: 0.5043 - val_accuracy: 0.7877 - val_loss: 0.5172
Epoch 17/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7950 - loss: 0.5096 - val_accuracy: 0.7877 - val_loss: 0.5172
Epoch 18/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7765 - loss: 0.5365 - val_accuracy: 0.7877 - val_loss: 0.5175
Epoch 19/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7868 - loss: 0.5204 - val_accuracy: 0.7877 - val_loss: 0.5195
Epoch 20/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7919 - loss: 0.5163 - val_accuracy: 0.7877 - val_loss: 0.5172
Epoch 21/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8065 - loss: 0.4922 - val_accuracy: 0.7877 - val_loss: 0.5187
Epoch 22/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7957 - loss: 0.5106 - val_accuracy: 0.7877 - val_loss: 0.5175
Epoch 23/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7944 - loss: 0.5065 - val_accuracy: 0.7877 - val_loss: 0.5173
Epoch 24/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7896 - loss: 0.5175 - val_accuracy: 0.7877 - val_loss: 0.5172
```

```
Epoch 20/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7919 - loss: 0.5163 - val_accuracy: 0.7877 - val_loss: 0.5172
Epoch 21/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8065 - loss: 0.4922 - val_accuracy: 0.7877 - val_loss: 0.5187
Epoch 22/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7957 - loss: 0.5106 - val_accuracy: 0.7877 - val_loss: 0.5175
Epoch 23/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7944 - loss: 0.5065 - val_accuracy: 0.7877 - val_loss: 0.5173
Epoch 24/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7896 - loss: 0.5175 - val_accuracy: 0.7877 - val_loss: 0.5172
Epoch 25/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7887 - loss: 0.5187 - val_accuracy: 0.7877 - val_loss: 0.5171
Epoch 26/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7933 - loss: 0.5108 - val_accuracy: 0.7877 - val_loss: 0.5171
Epoch 27/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8104 - loss: 0.4857 - val_accuracy: 0.7877 - val_loss: 0.5208
Epoch 28/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8034 - loss: 0.5014 - val_accuracy: 0.7877 - val_loss: 0.5181
Epoch 29/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7746 - loss: 0.5357 - val_accuracy: 0.7877 - val_loss: 0.5187
Epoch 30/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7807 - loss: 0.5326 - val_accuracy: 0.7877 - val_loss: 0.5184
Epoch 31/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7932 - loss: 0.5101 - val_accuracy: 0.7877 - val_loss: 0.5184
Epoch 32/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7887 - loss: 0.5151 - val_accuracy: 0.7877 - val_loss: 0.5168
Epoch 33/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7890 - loss: 0.5183 - val_accuracy: 0.7877 - val_loss: 0.5175
Epoch 34/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7832 - loss: 0.5258 - val_accuracy: 0.7877 - val_loss: 0.5177
Epoch 35/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7979 - loss: 0.5033 - val_accuracy: 0.7877 - val_loss: 0.5176
Epoch 36/50
```

```
Epoch 35/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7979 - loss: 0.5033 - val_accuracy: 0.7877 - val_loss: 0.5176
Epoch 36/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7931 - loss: 0.5083 - val_accuracy: 0.7877 - val_loss: 0.5177
Epoch 37/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7864 - loss: 0.5164 - val_accuracy: 0.7877 - val_loss: 0.5183
Epoch 38/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7825 - loss: 0.5242 - val_accuracy: 0.7877 - val_loss: 0.5176
Epoch 39/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7809 - loss: 0.5229 - val_accuracy: 0.7877 - val_loss: 0.5172
Epoch 40/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7866 - loss: 0.5178 - val_accuracy: 0.7877 - val_loss: 0.5185
Epoch 41/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7978 - loss: 0.5042 - val_accuracy: 0.7877 - val_loss: 0.5182
Epoch 42/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7920 - loss: 0.5124 - val_accuracy: 0.7877 - val_loss: 0.5179
Epoch 43/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7990 - loss: 0.5031 - val_accuracy: 0.7877 - val_loss: 0.5180
Epoch 44/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.8037 - loss: 0.4950 - val_accuracy: 0.7877 - val_loss: 0.5199
Epoch 45/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7816 - loss: 0.5221 - val_accuracy: 0.7877 - val_loss: 0.5251
Epoch 46/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7932 - loss: 0.5059 - val_accuracy: 0.7877 - val_loss: 0.5182
Epoch 47/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7718 - loss: 0.5317 - val_accuracy: 0.7877 - val_loss: 0.5254
Epoch 48/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7816 - loss: 0.5213 - val_accuracy: 0.7877 - val_loss: 0.5241
Epoch 49/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7948 - loss: 0.5059 - val_accuracy: 0.7877 - val_loss: 0.5258
Epoch 50/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7967 - loss: 0.5024 - val_accuracy: 0.7877 - val_loss: 0.5264
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7816 - loss: 0.5221 - val_accuracy: 0.7877 - val_loss: 0.5251
Epoch 46/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7932 - loss: 0.5059 - val_accuracy: 0.7877 - val_loss: 0.5182
Epoch 47/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7718 - loss: 0.5317 - val_accuracy: 0.7877 - val_loss: 0.5254
Epoch 48/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7816 - loss: 0.5213 - val_accuracy: 0.7877 - val_loss: 0.5241
Epoch 49/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7948 - loss: 0.5059 - val_accuracy: 0.7877 - val_loss: 0.5258
Epoch 50/50
41/41 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7967 - loss: 0.5024 - val_accuracy: 0.7877 - val_loss: 0.5264
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.s
ave_model(model, 'my_model.keras')`.
LSTM Model - Train Accuracy: 79.04%
LSTM Model - Validation Accuracy: 78.77%
LSTM Model - Test Accuracy: 78.85%
Random Forest Model - Train Accuracy: 100.00%
Random Forest Model - Test Accuracy: 78.85%
C:\Users\Ansh Gupta\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: U
serWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(s
hape)` object as the first layer in the model instead.
```

```
C:\Users\Ansh Gupta\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107
serWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Inpu
hape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.6853 - loss: 5.3405 - val_accuracy: 0.7557 - val_loss: 0.7101
Epoch 2/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.7727 - loss: 0.5735 - val_accuracy: 0.7710 - val_loss: 1.2572
Epoch 3/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.6910 - loss: 0.8566 - val_accuracy: 0.5038 - val_loss: 0.8033
Epoch 4/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7425 - loss: 0.5778 - val_accuracy: 0.7328 - val_loss: 0.6321
Epoch 5/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.7808 - loss: 0.4935 - val_accuracy: 0.7710 - val_loss: 0.6378
Epoch 6/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8115 - loss: 0.4424 - val_accuracy: 0.6641 - val_loss: 0.6521
Epoch 7/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8075 - loss: 0.4597 - val_accuracy: 0.7557 - val_loss: 0.6882
Epoch 8/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8421 - loss: 0.3937 - val_accuracy: 0.6794 - val_loss: 0.6457
Epoch 9/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8646 - loss: 0.3538 - val_accuracy: 0.7328 - val_loss: 0.6143
Epoch 10/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8656 - loss: 0.3229 - val_accuracy: 0.7557 - val_loss: 0.6495
Epoch 11/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8957 - loss: 0.2738 - val_accuracy: 0.7405 - val_loss: 0.6802
Epoch 12/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9037 - loss: 0.2500 - val_accuracy: 0.7710 - val_loss: 0.7797
Epoch 13/100
```

```
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9037 - loss: 0.2500 - val_accuracy: 0.7710 - val_loss: 0.7797
Epoch 13/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9079 - loss: 0.2462 - val_accuracy: 0.7710 - val_loss: 0.7021
Epoch 14/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9249 - loss: 0.2140 - val_accuracy: 0.7405 - val_loss: 0.6733
Epoch 15/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9540 - loss: 0.1619 - val_accuracy: 0.6412 - val_loss: 0.7114
Epoch 16/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9556 - loss: 0.1513 - val_accuracy: 0.7176 - val_loss: 0.7367
Epoch 17/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9743 - loss: 0.1194 - val_accuracy: 0.7328 - val_loss: 0.8038
Epoch 18/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9894 - loss: 0.0895 - val_accuracy: 0.6947 - val_loss: 0.7799
Epoch 19/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9943 - loss: 0.0672 - val_accuracy: 0.7023 - val_loss: 0.7994
Epoch 20/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9985 - loss: 0.0569 - val_accuracy: 0.6870 - val_loss: 0.8478
Epoch 21/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9998 - loss: 0.0450 - val_accuracy: 0.6947 - val_loss: 0.8274
Epoch 22/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0417 - val_accuracy: 0.6794 - val_loss: 0.8725
Epoch 23/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0313 - val_accuracy: 0.7099 - val_loss: 0.9134
Epoch 24/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0219 - val_accuracy: 0.7023 - val_loss: 0.9262
Epoch 25/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0192 - val_accuracy: 0.6947 - val_loss: 0.9376
Epoch 26/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0166 - val_accuracy: 0.7099 - val_loss: 0.9918
Epoch 27/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0119 - val_accuracy: 0.7099 - val_loss: 1.0382
Epoch 28/100
```


```
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0192 - val_accuracy: 0.6947 - val_loss: 0.9376
Epoch 26/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0166 - val_accuracy: 0.7099 - val_loss: 0.9918
Epoch 27/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0119 - val_accuracy: 0.7099 - val_loss: 1.0382
Epoch 28/100
```


```
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0102 - val_accuracy: 0.7405 - val_loss: 1.0994
Epoch 29/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0086 - val_accuracy: 0.7099 - val_loss: 1.1180
Epoch 30/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0078 - val_accuracy: 0.7252 - val_loss: 1.0730
Epoch 31/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0069 - val_accuracy: 0.7405 - val_loss: 1.1603
Epoch 32/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0057 - val_accuracy: 0.7328 - val_loss: 1.1730
Epoch 33/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0050 - val_accuracy: 0.7328 - val_loss: 1.1969
Epoch 34/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0041 - val_accuracy: 0.7328 - val_loss: 1.2304
Epoch 35/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0040 - val_accuracy: 0.7328 - val_loss: 1.2587
Epoch 36/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0039 - val_accuracy: 0.7252 - val_loss: 1.2764
Epoch 37/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0032 - val_accuracy: 0.7176 - val_loss: 1.2796
Epoch 38/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0030 - val_accuracy: 0.7252 - val_loss: 1.2992
Epoch 39/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0025 - val_accuracy: 0.7328 - val_loss: 1.2616
Epoch 40/100
```

```
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0025 - val_accuracy: 0.7328 - val_loss: 1.2616
Epoch 40/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 0.0024 - val_accuracy: 0.7176 - val_loss: 1.3687
Epoch 41/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0021 - val_accuracy: 0.7252 - val_loss: 1.3608
Epoch 42/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0020 - val_accuracy: 0.7252 - val_loss: 1.4013
Epoch 43/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0020 - val_accuracy: 0.7252 - val_loss: 1.3841
Epoch 44/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0017 - val_accuracy: 0.7328 - val_loss: 1.3847
Epoch 45/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0016 - val_accuracy: 0.7252 - val_loss: 1.4348
Epoch 46/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0014 - val_accuracy: 0.7328 - val_loss: 1.4223
Epoch 47/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0013 - val_accuracy: 0.7328 - val_loss: 1.4184
Epoch 48/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0013 - val_accuracy: 0.7252 - val_loss: 1.4609
Epoch 49/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0012 - val_accuracy: 0.7328 - val_loss: 1.4513
Epoch 50/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0011 - val_accuracy: 0.7328 - val_loss: 1.4688
Epoch 51/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 0.0010 - val_accuracy: 0.7252 - val_loss: 1.4735
Epoch 52/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 9.5252e-04 - val_accuracy: 0.7328 - val_loss: 1.5023
Epoch 53/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 8.8522e-04 - val_accuracy: 0.7252 - val_loss: 1.4967
Epoch 54/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 8.6073e-04 - val_accuracy: 0.7252 - val_loss: 1.5231
Epoch 55/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 7.3731e-04 - val_accuracy: 0.7252 - val_loss: 1.4856
```

```
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 5.6028e-04 - val_accuracy: 0.7252 - val_loss: 1.6222
Epoch 62/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 5.1656e-04 - val_accuracy: 0.7252 - val_loss: 1.6015
Epoch 63/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 4.9217e-04 - val_accuracy: 0.7252 - val_loss: 1.6242
Epoch 64/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 4.5575e-04 - val_accuracy: 0.7252 - val_loss: 1.6048
Epoch 65/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 4.5039e-04 - val_accuracy: 0.7252 - val_loss: 1.6452
Epoch 66/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 4.1862e-04 - val_accuracy: 0.7252 - val_loss: 1.6502
Epoch 67/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 4.0788e-04 - val_accuracy: 0.7252 - val_loss: 1.6605
Epoch 68/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 3.8557e-04 - val_accuracy: 0.7252 - val_loss: 1.6332
Epoch 69/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 3.4773e-04 - val_accuracy: 0.7252 - val_loss: 1.7082
Epoch 70/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 3.3542e-04 - val_accuracy: 0.7252 - val_loss: 1.6938
Epoch 71/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 3.2520e-04 - val_accuracy: 0.7252 - val_loss: 1.6801
Epoch 72/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 3.1377e-04 - val_accuracy: 0.7252 - val_loss: 1.7272
Epoch 73/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 3.2763e-04 - val_accuracy: 0.7252 - val_loss: 1.7182
Epoch 74/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 2.8979e-04 - val_accuracy: 0.7252 - val_loss: 1.7284
Epoch 75/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 2.8579e-04 - val_accuracy: 0.7252 - val_loss: 1.7430
Epoch 76/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 2.5283e-04 - val_accuracy: 0.7252 - val_loss: 1.7273
Epoch 77/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 2.6149e-04 - val_accuracy: 0.7252 - val_loss: 1.7421
```

```
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 3.3542e-04 - val_accuracy: 0.7252 - val_loss: 1.6938
Epoch 71/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 3.2520e-04 - val_accuracy: 0.7252 - val_loss: 1.6801
Epoch 72/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 3.1377e-04 - val_accuracy: 0.7252 - val_loss: 1.7272
Epoch 73/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 3.2763e-04 - val_accuracy: 0.7252 - val_loss: 1.7182
Epoch 74/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 2.8979e-04 - val_accuracy: 0.7252 - val_loss: 1.7284
Epoch 75/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 1.0000 - loss: 2.8579e-04 - val_accuracy: 0.7252 - val_loss: 1.7430
Epoch 76/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 2.5283e-04 - val_accuracy: 0.7252 - val_loss: 1.7273
Epoch 77/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 2.6149e-04 - val_accuracy: 0.7252 - val_loss: 1.7421
Epoch 78/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 2.2706e-04 - val_accuracy: 0.7405 - val_loss: 1.7919
Epoch 79/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 2.2897e-04 - val_accuracy: 0.7176 - val_loss: 1.7512
Epoch 80/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 2.2339e-04 - val_accuracy: 0.7176 - val_loss: 1.7813
Epoch 81/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 2.2017e-04 - val_accuracy: 0.7176 - val_loss: 1.7745
Epoch 82/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 2.0589e-04 - val_accuracy: 0.7176 - val_loss: 1.7734
Epoch 83/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 2.0157e-04 - val_accuracy: 0.7176 - val_loss: 1.7890
Epoch 84/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.8751e-04 - val_accuracy: 0.7176 - val_loss: 1.7983
Epoch 85/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.8102e-04 - val_accuracy: 0.7176 - val_loss: 1.7896
Epoch 86/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.6298e-04 - val_accuracy: 0.7176 - val_loss: 1.7901
```

```
Epoch 88/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.6095e-04 - val_accuracy: 0.7328 - val_loss: 1.8476
Epoch 89/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.5668e-04 - val_accuracy: 0.7176 - val_loss: 1.8347
Epoch 90/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.4703e-04 - val_accuracy: 0.7252 - val_loss: 1.8290
Epoch 91/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.4550e-04 - val_accuracy: 0.7328 - val_loss: 1.8499
Epoch 92/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.3182e-04 - val_accuracy: 0.7252 - val_loss: 1.8566
Epoch 93/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.3547e-04 - val_accuracy: 0.7176 - val_loss: 1.8409
Epoch 94/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.3449e-04 - val_accuracy: 0.7099 - val_loss: 1.8326
Epoch 95/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.2722e-04 - val_accuracy: 0.7252 - val_loss: 1.8642
Epoch 96/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.1694e-04 - val_accuracy: 0.7328 - val_loss: 1.8904
Epoch 97/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.1101e-04 - val_accuracy: 0.7328 - val_loss: 1.8889
Epoch 98/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.1990e-04 - val_accuracy: 0.7328 - val_loss: 1.8914
Epoch 99/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.0431e-04 - val_accuracy: 0.7252 - val_loss: 1.9220
Epoch 100/100
37/37 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 1.0000 - loss: 1.0228e-04 - val_accuracy: 0.7328 - val_loss: 1.8901
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.s
ave_model(model, 'my_model.keras')`.
CNN Model - Train Accuracy: 97.32%
CNN Model - Validation Accuracy: 100.00%
CNN Model - Test Accuracy: 97.32%
```

OUTPUT

```
LSTM Model - Train Accuracy: 79.04%
LSTM Model - Validation Accuracy: 78.77%
LSTM Model - Test Accuracy: 78.85%
Random Forest Model - Train Accuracy: 100.00%
Random Forest Model - Test Accuracy: 78.85%
```

```
CNN Model - Train Accuracy: 97.32%
CNN Model - Validation Accuracy: 100.00%
CNN Model - Test Accuracy: 97.32%
```