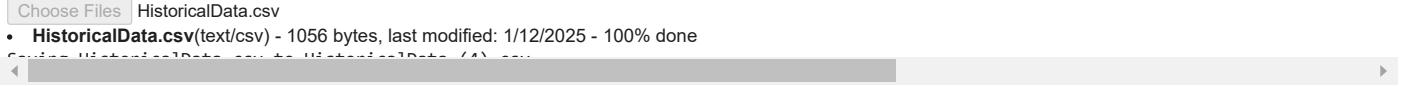
```
# Install necessary libraries
!pip install tensorflow keras tabulate

# Import libraries
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tabulate import tabulate
from google.colab import files
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.17.1)
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (3.5.0)
Requirement already satisfied: tabulate in /usr/local/lib/python3.10/dist-packages (0.9.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.12.23)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.12.1)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.69.0)
Requirement already satisfied: tensorboard<2.18,>=2.17 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.1
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (from keras) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages (from keras) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages (from keras) (0.13.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.45.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (2.3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow) (202
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow) (3.
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorflow) (3.
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich->keras) (0.1.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.18,>=2.
```

```
# Step 1: Upload your dataset (assuming it's a CSV file)
uploaded = files.upload()  # Manually upload the file
file_name = list(uploaded.keys())[0]  # Get the uploaded file name
df = pd.read_csv(file_name)  # Load the dataset
df['Date'] = pd.to_datetime(df['Date'])
```




```
# Step 2: Preprocess the dataset
# Ensure it has a column named 'target' (or update this code to match your data)
for col in ['Close/Last', 'Open', 'High', 'Low']:
    df[col] = df[col].astype(str).str.replace('$', '', regex=False).astype(float)

df.sort_values(by='Date', inplace=True)
features = ['Open', 'High', 'Low', 'Volume']
target = 'Close/Last'
```

```
scaler = MinMaxScaler()
df[features + [target]] = scaler.fit_transform(df[features + [target]])
df = df.dropna()

# Sequence preparation
def create_sequences(data, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        X.append(data[i:i + seq_length, :-1])
        y.append(data[i + seq_length, -1])
    return np.array(X), np.array(y)

seq_length = 10
data = df[features + [target]].values
X, y = create_sequences(data, seq_length)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Step 3: Define the LSTM model builder
def build_lstm(units, dropout_rate, optimizer, activation):
    model = Sequential()
    model.add(LSTM(units, activation=activation, input_shape=(X_train.shape[1], X_train.shape[2])))
    model.add(Dropout(dropout_rate))
    model.add(Dense(1, activation='linear'))
    model.compile(optimizer=optimizer, loss='mean_squared_error')
    return model
```

```
# Step 4: Set up hyperparameter grid
param_grid = {
    'units': [50, 100],
    'dropout_rate': [0.2, 0.3],
    'optimizer': ['adam', 'rmsprop'],
    'activation': ['relu', 'tanh'],
    'batch_size': [16, 32],
    'epochs': [10]
}
```

```
# Step 5: Perform grid search
best_loss = float('inf')
best_params = None
results = []

for units in param_grid['units']:
    for dropout_rate in param_grid['dropout_rate']:
        for optimizer in param_grid['optimizer']:
            for activation in param_grid['activation']:
                for batch_size in param_grid['batch_size']:
                    for epochs in param_grid['epochs']:
                        print(f"Training with units={units}, dropout_rate={dropout_rate}, optimizer={optimizer}, "
                              f"activation={activation}, batch_size={batch_size}, epochs={epochs}")

                        # Build and train the model
                        model = build_lstm(units, dropout_rate, optimizer, activation)
                        model.fit(X_train, y_train, batch_size=batch_size, epochs=epochs, verbose=0)

                        # Evaluate the model
                        loss = model.evaluate(X_test, y_test, verbose=0)
                        print(f"Loss: {loss}")

                        # Store results
                        results.append({
                            'units': units,
                            'dropout_rate': dropout_rate,
                            'optimizer': optimizer,
                            'activation': activation,
```

```
                        'batch_size': batch_size,
                        'epochs': epochs,
                        'loss': loss
                    })

                    # Update best parameters
                    if loss < best_loss:
                        best_loss = loss
                        best_params = {
                            'units': units,
                            'dropout_rate': dropout_rate,
                            'optimizer': optimizer,
                            'activation': activation,
                            'batch_size': batch_size,
                            'epochs': epochs
                        }
```

```
WARNING:tensorflow:5 out of the last 5 calls to <function TensorFlowTrainer.make_test_function.<locals>.one_step_on_iterator at 0x7c1
Loss: 0.1967548131942749
Training with units=50, dropout_rate=0.2, optimizer=rmsprop, activation=relu, batch_size=32, epochs=10
WARNING:tensorflow:6 out of the last 6 calls to <function TensorFlowTrainer.make_test_function.<locals>.one_step_on_iterator at 0x7c1
Loss: 0.1892838180065155
Training with units=50, dropout_rate=0.2, optimizer=rmsprop, activation=tanh, batch_size=16, epochs=10
Loss: 0.19456905126571655
Training with units=50, dropout_rate=0.2, optimizer=rmsprop, activation=tanh, batch_size=32, epochs=10
Loss: 0.2164136916399002
Training with units=50, dropout_rate=0.3, optimizer=adam, activation=relu, batch_size=16, epochs=10
Loss: 0.2169986069202423
Training with units=50, dropout_rate=0.3, optimizer=adam, activation=relu, batch_size=32, epochs=10
Loss: 0.17354604601860046
Training with units=50, dropout_rate=0.3, optimizer=adam, activation=tanh, batch_size=16, epochs=10
Loss: 0.20950748026371002
Training with units=50, dropout_rate=0.3, optimizer=adam, activation=tanh, batch_size=32, epochs=10
Loss: 0.19707760214805603
Training with units=50, dropout_rate=0.3, optimizer=rmsprop, activation=relu, batch_size=16, epochs=10
Loss: 0.1773500144481659
Training with units=50, dropout_rate=0.3, optimizer=rmsprop, activation=relu, batch_size=32, epochs=10
Loss: 0.1687687635421753
Training with units=50, dropout_rate=0.3, optimizer=rmsprop, activation=tanh, batch_size=16, epochs=10
Loss: 0.1871277093887329
Training with units=50, dropout_rate=0.3, optimizer=rmsprop, activation=tanh, batch_size=32, epochs=10
Loss: 0.2220405787229538
Training with units=100, dropout_rate=0.2, optimizer=adam, activation=relu, batch_size=16, epochs=10
Loss: 0.18001335859298706
Training with units=100, dropout_rate=0.2, optimizer=adam, activation=relu, batch_size=32, epochs=10
Loss: 0.20498766005039215
Training with units=100, dropout_rate=0.2, optimizer=adam, activation=tanh, batch_size=16, epochs=10
Loss: 0.37734389305114746
Training with units=100, dropout_rate=0.2, optimizer=adam, activation=tanh, batch_size=32, epochs=10
Loss: 0.27588486671447754
Training with units=100, dropout_rate=0.2, optimizer=rmsprop, activation=relu, batch_size=16, epochs=10
Loss: 0.20038920640945435
Training with units=100, dropout_rate=0.2, optimizer=rmsprop, activation=relu, batch_size=32, epochs=10
Loss: 0.19227394461631775
Training with units=100, dropout_rate=0.2, optimizer=rmsprop, activation=tanh, batch_size=16, epochs=10
Loss: 0.23142211139202118
Training with units=100, dropout_rate=0.2, optimizer=rmsprop, activation=tanh, batch_size=32, epochs=10
Loss: 0.19390887022018433
Training with units=100, dropout_rate=0.3, optimizer=adam, activation=relu, batch_size=16, epochs=10
Loss: 0.19701841473579407
Training with units=100, dropout_rate=0.3, optimizer=adam, activation=relu, batch_size=32, epochs=10
Loss: 0.24380964040756226
Training with units=100, dropout_rate=0.3, optimizer=adam, activation=tanh, batch_size=16, epochs=10
Loss: 0.3544824719429016
Training with units=100, dropout_rate=0.3, optimizer=adam, activation=tanh, batch_size=32, epochs=10
Loss: 0.3392270803451538
Training with units=100, dropout_rate=0.3, optimizer=rmsprop, activation=relu, batch_size=16, epochs=10
Loss: 0.20625640451908112
Training with units=100, dropout_rate=0.3, optimizer=rmsprop, activation=relu, batch_size=32, epochs=10
Loss: 0.23758506774902344
Training with units=100, dropout_rate=0.3, optimizer=rmsprop, activation=tanh, batch_size=16, epochs=10
Loss: 0.20459190011024475
Training with units=100, dropout_rate=0.3, optimizer=rmsprop, activation=tanh, batch_size=32, epochs=10
Loss: 0.23959621787071228
```

```
# Step 6: Save results to CSV
output_file = 'LSTM_GridSearch_Results.csv'
results_df = pd.DataFrame(results)
```

```
results_df.to_csv(output_file, index=False)

print("\nBest Parameters:")
print(tabulate([best_params.values()], headers=best_params.keys(), tablefmt="fancy_grid"))

# Download the CSV file to your PC
files.download(output_file)
```

```
Best Parameters:
```

units	dropout_rate	optimizer	activation	batch_size	epochs
50	0.3	rmsprop	relu	32	10