COS30018 - Option B - Task 2: Data processing 1

Student Name: Nguyen Duc Le Nguyen

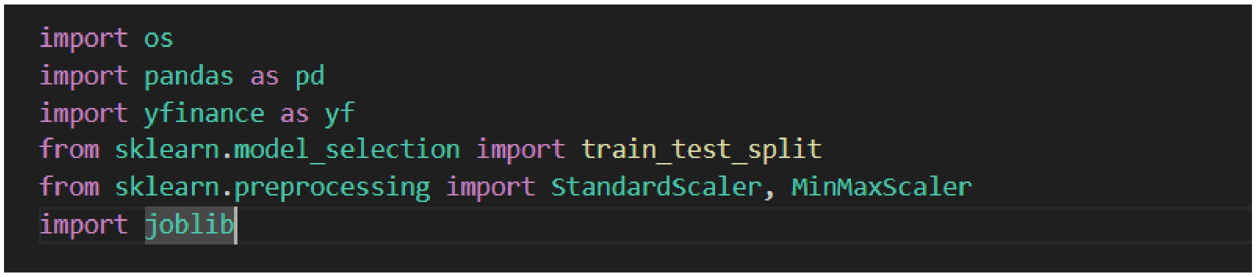
Id: 104224493

Summary of Effort

This report details the development of a function to load and process stock market data with various features, including handling NaN values, splitting data into train/test sets, scaling features, and saving/loading data locally.

Code Breakdown and Explanation

Below is a detailed explanation of the less straightforward lines of code within the function.



```python
import os
import pandas as pd
import yfinance as yf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import joblib
```

os: Provides a way of using operating system-dependent functionality, such as reading or writing to the filesystem.

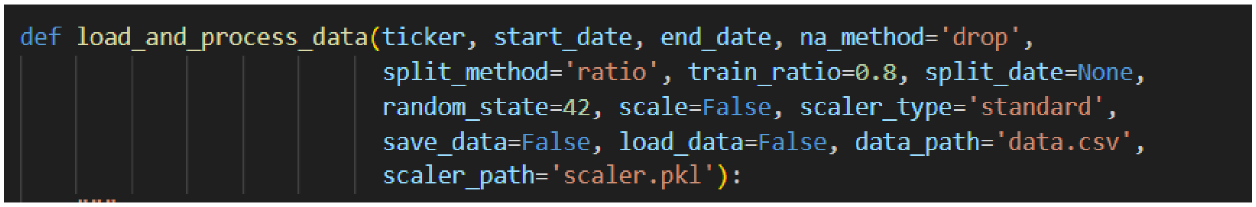
pandas: A powerful data manipulation library.

yfinance: A library to fetch financial data from Yahoo Finance.

sklearn.model_selection.train_test_split: A utility function to split data into train and test sets.

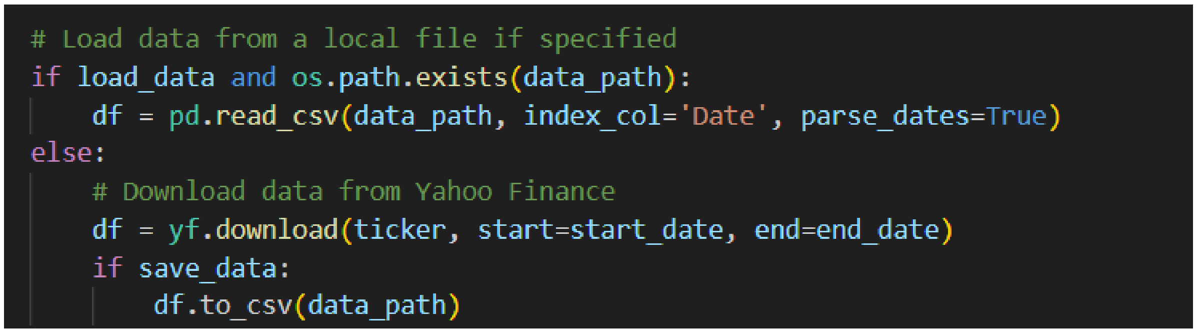
sklearn.preprocessing.StandardScaler and MinMaxScaler: Tools for feature scaling.

joblib: A library for saving and loading Python objects.



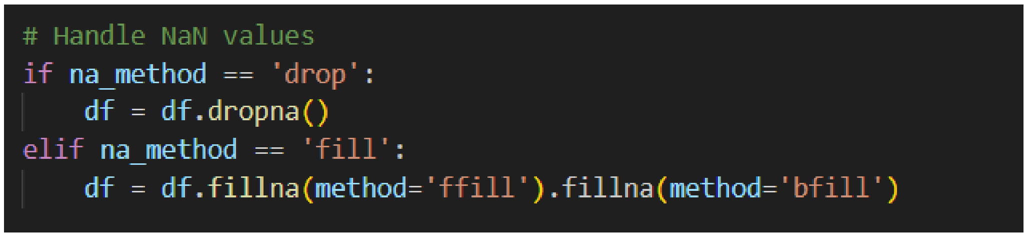
```python
def load_and_process_data(ticker, start_date, end_date, na_method='drop',
                          split_method='ratio', train_ratio=0.8, split_date=None,
                          random_state=42, scale=False, scaler_type='standard',
                          save_data=False, load_data=False, data_path='data.csv',
                          scaler_path='scaler.pkl'):
```

This function initializes with several parameters, allowing flexibility in data loading, processing, and saving.



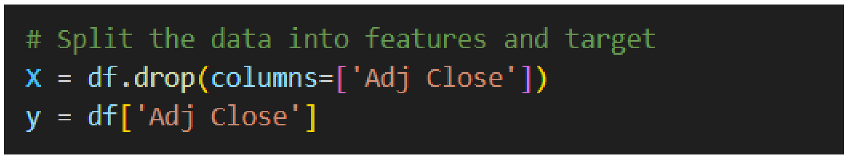
```python
# Load data from a local file if specified
if load_data and os.path.exists(data_path):
    df = pd.read_csv(data_path, index_col='Date', parse_dates=True)
else:
    # Download data from Yahoo Finance
    df = yf.download(ticker, start=start_date, end=end_date)
    if save_data:
        df.to_csv(data_path)
```

Checks if data should be loaded from a local file. If not, it downloads the data from Yahoo Finance and saves it if required.



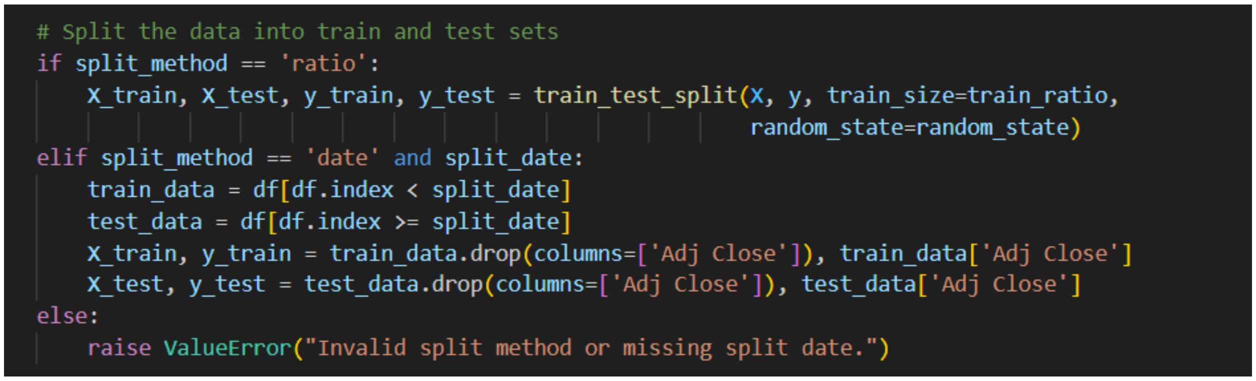
```python
# Handle NaN values
if na_method == 'drop':
    df = df.dropna()
elif na_method == 'fill':
    df = df.fillna(method='ffill').fillna(method='bfill')
```

Handles NaN values by either dropping them or filling them. Forward fill (ffill) and backward fill (bfill) ensure no NaN values remain.



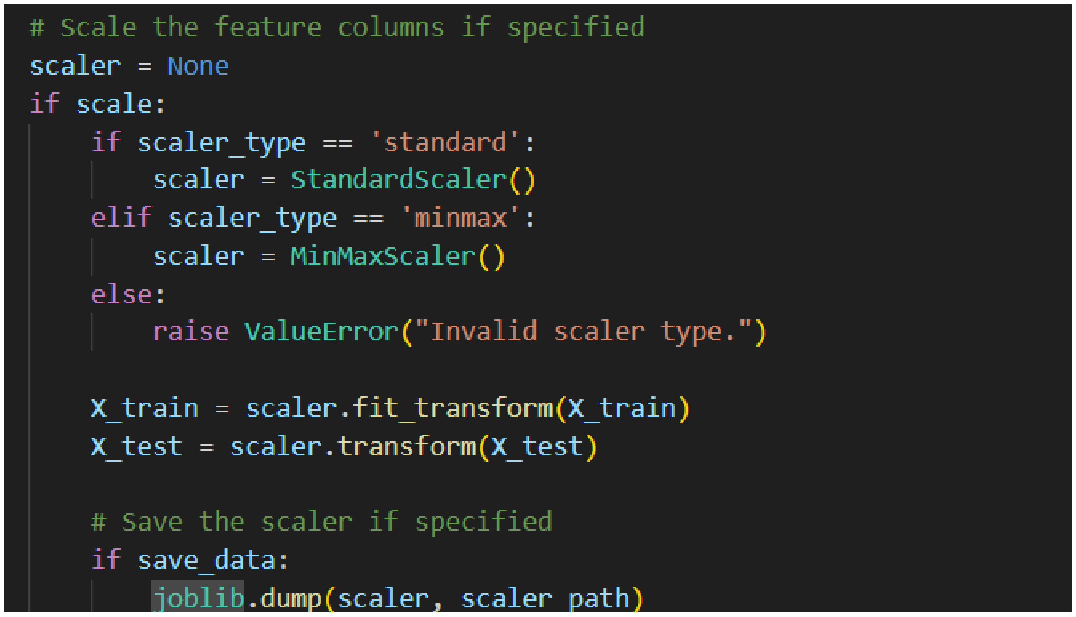
```python
# Split the data into features and target
X = df.drop(columns=['Adj Close'])
y = df['Adj Close']
```

Splits the dataframe into features (X) and target (y). Here, 'Adj Close' is assumed to be the target variable.



```python
# Split the data into train and test sets
if split_method == 'ratio':
    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=train_ratio,
                                                          random_state=random_state)
elif split_method == 'date' and split_date:
    train_data = df[df.index < split_date]
    test_data = df[df.index >= split_date]
    X_train, y_train = train_data.drop(columns=['Adj Close']), train_data['Adj Close']
    X_test, y_test = test_data.drop(columns=['Adj Close']), test_data['Adj Close']
else:
    raise ValueError("Invalid split method or missing split date.")
```

Depending on the split_method, the data is split either randomly according to a specified ratio or by a specific date.

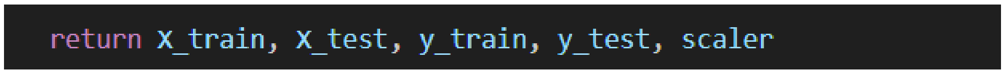


```python
# Scale the feature columns if specified
scaler = None
if scale:
    if scaler_type == 'standard':
        scaler = StandardScaler()
    elif scaler_type == 'minmax':
        scaler = MinMaxScaler()
    else:
        raise ValueError("Invalid scaler type.")

    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # Save the scaler if specified
    if save_data:
        joblib.dump(scaler, scaler_path)
```

If scaling is requested, the function applies either StandardScaler or MinMaxScaler to the feature columns. It also saves the scaler if specified.



```python
    return X_train, X_test, y_train, y_test, scaler
```

Returns the processed data splits and the scaler (if applied).

This report covered the key lines of the load_and_process_data function, explaining each part to ensure clarity. Further inquiries about any specific line of code are welcome.