SUBMITTED BY- MAZIZ YAMIN RAHMAN(102008721)

COS30018 - Option B - Task 2: Data processing 1

In this task, it was asked to write a function and process a dataset with multiple features with the following requirements:

a. This function will allow you to specify the start date and the end date for the whole dataset as inputs.

```
#Allowing to specify dates
START = dt.datetime.strptime(input("Enter Start Date(YYYY-MM-DD): "), "%Y-
%m-%d")      # Start date to read
END = dt.datetime.strptime(input("Enter End Date(YYYY-MM-DD): "), "%Y-%m-
%d")          # End date to read

data = web.DataReader(company, source, START, END)
```

b. This function will allow you to deal with the NaN issue in the data.

```
#Handeling NAN values
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(required_data)
required_data = imputer.transform(required_data)
```

c. This function will also allow you to use different methods to split the data into train/test data; e.g. you can split it according to some specified ratio of train/test and you can specify to split it by date or randomly.

```
#Splitting on basis of dates or randomly
if split_by_dates:
    x_train, x_test = X[:int(len(X)*train_size)],
X[int(len(X)*train_size):]
    y_train, y_test = Y[:int(len(Y)*train_size)],
Y[int(len(Y)*train_size):]
else:
    x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 1
- train_size, random_state = 42)

return x_train, x_test, y_train, y_test, scaler
```

d. This function will have the option to allow you to store the downloaded data on your local machine for future uses and to load the data locally to save time.

```
#Saving the downloaded data with user's consent
consent = '0'
while consent not in 'yYnN':
    consent = input("Do you want to save the downloaded data? (y/n) : ")
    if consent == 'y':
        data.to_csv(f'{company}.csv')

required_data = data[PRICE_VALUE].values.reshape(-1,1)
```

e. This function will also allow you to have an option to scale your feature columns and store the scalers in a data structure to allow future access to these scalers

```
#Feature Scaling
scaler = StandardScaler()
scaled_data = scaler.fit_transform(required_data)

X, Y = [], []
for i in range(days, len(scaled_data)):
    X.append(scaled_data[i-days:i])
    Y.append(scaled_data[i])

X,Y = np.array(X), np.array(Y)
```