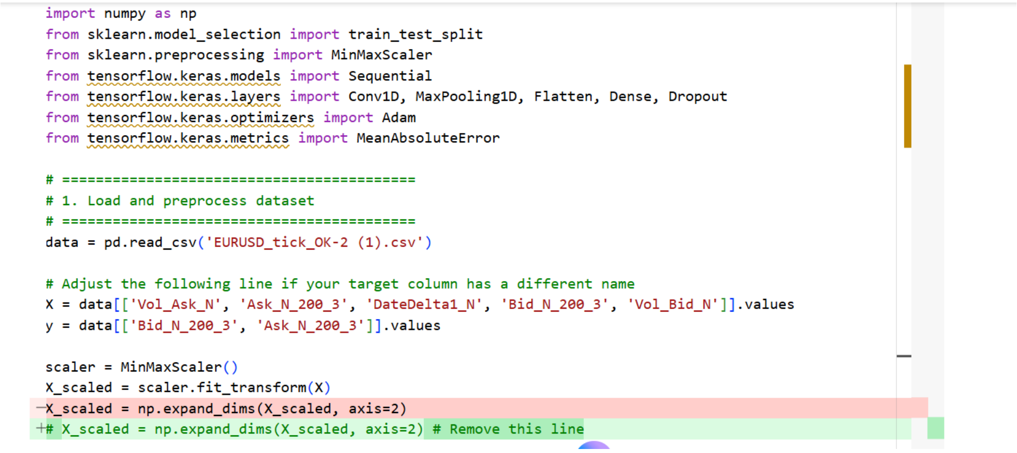
Name :- Rudrapati venkatesh

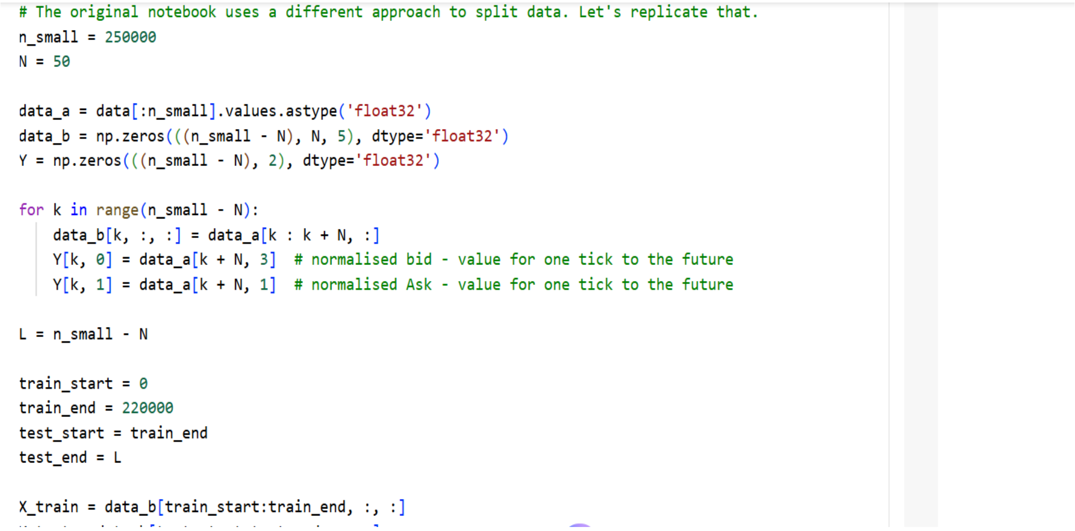
SID:- 2461075

Subjet:- Machine Learning in Finance

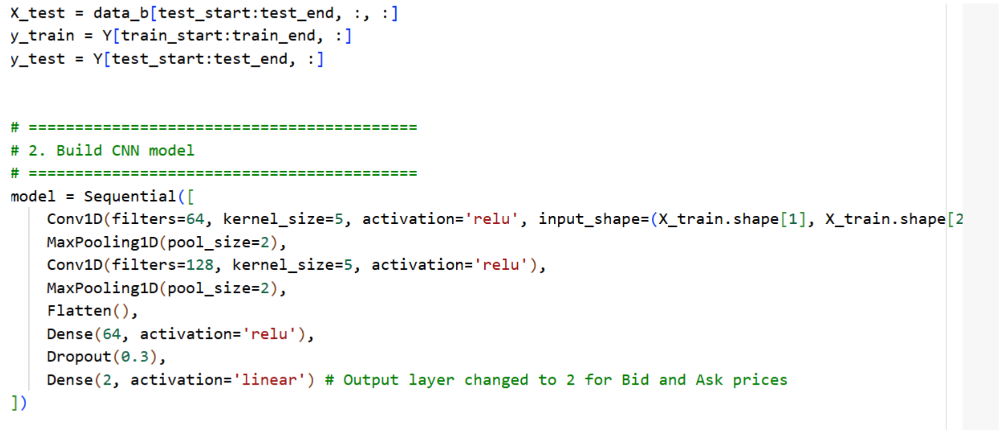
Code:-



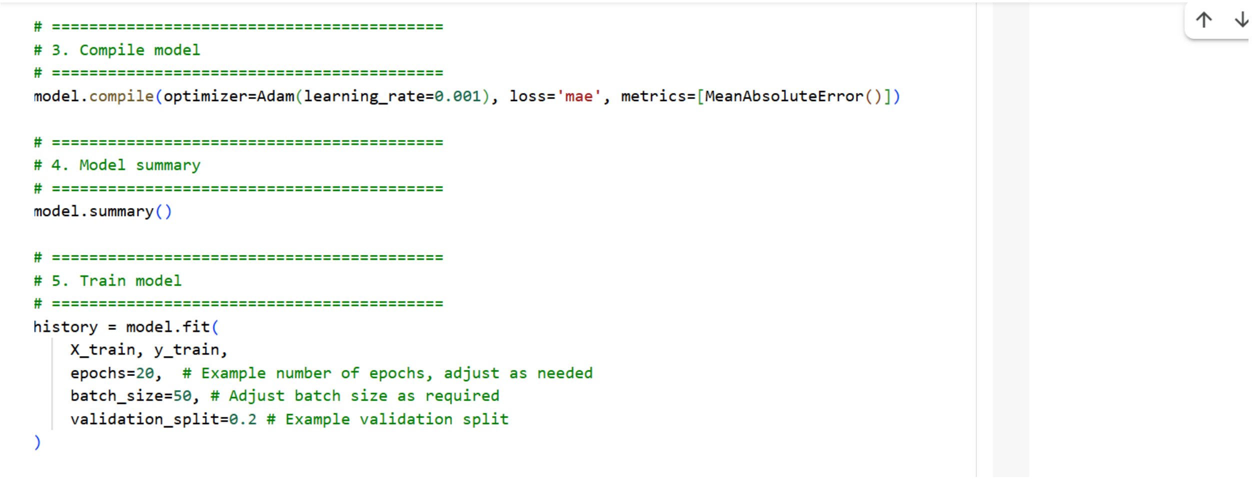
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import MeanAbsoluteError

# ==========================================
# 1. Load and preprocess dataset
# ==========================================
data = pd.read_csv('EURUSD_tick_OK-2 (1).csv')

# Adjust the following line if your target column has a different name
X = data[['Vol_Ask_N', 'Ask_N_200_3', 'DateDelta1_N', 'Bid_N_200_3', 'Vol_Bid_N']].values
y = data[['Bid_N_200_3', 'Ask_N_200_3']].values

scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
-X_scaled = np.expand_dims(X_scaled, axis=2)
+# X_scaled = np.expand_dims(X_scaled, axis=2) # Remove this line
```



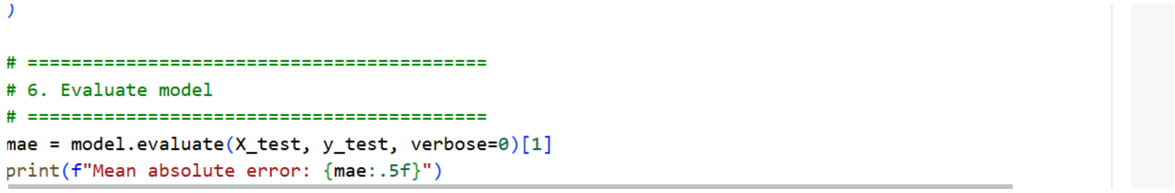
```
# The original notebook uses a different approach to split data. Let's replicate that.
n_small = 250000
N = 50

data_a = data[:n_small].values.astype('float32')
data_b = np.zeros(((n_small - N), N, 5), dtype='float32')
Y = np.zeros(((n_small - N), 2), dtype='float32')

for k in range(n_small - N):
    data_b[k, :, :] = data_a[k : k + N, :]
    Y[k, 0] = data_a[k + N, 3]  # normalised bid - value for one tick to the future
    Y[k, 1] = data_a[k + N, 1]  # normalised Ask - value for one tick to the future

L = n_small - N

train_start = 0
train_end = 220000
test_start = train_end
test_end = L

X_train = data_b[train_start:train_end, :, :]
```



```
X_test = data_b[test_start:test_end, :, :]
y_train = Y[train_start:train_end, :]
y_test = Y[test_start:test_end, :]


# ==========================================
# 2. Build CNN model
# ==========================================
model = Sequential([
    Conv1D(filters=64, kernel_size=5, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2
    MaxPooling1D(pool_size=2),
    Conv1D(filters=128, kernel_size=5, activation='relu'),
    MaxPooling1D(pool_size=2),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(2, activation='linear') # Output layer changed to 2 for Bid and Ask prices
])
```

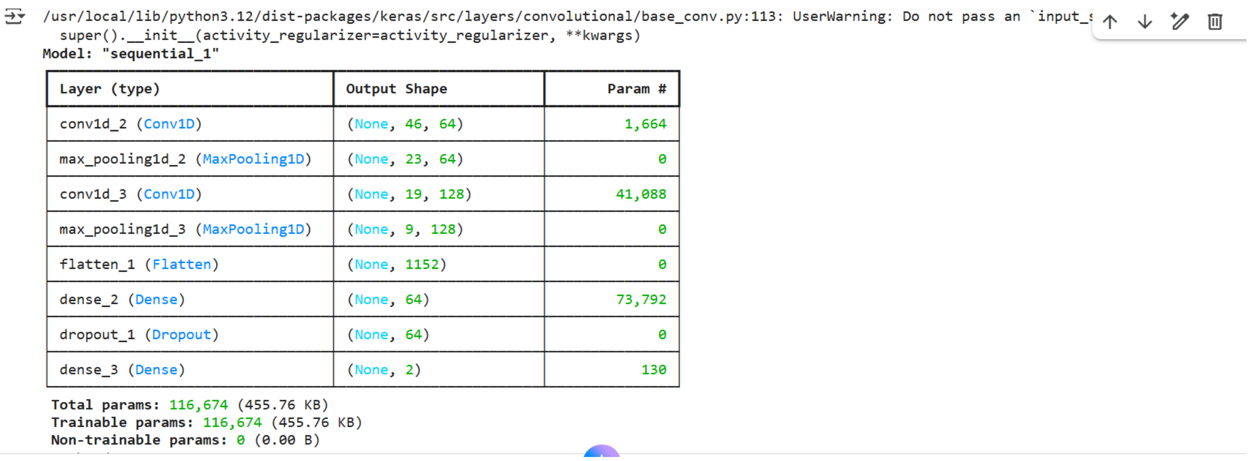


```
# ==========================================
# 3. Compile model
# ==========================================
model.compile(optimizer=Adam(learning_rate=0.001), loss='mae', metrics=[MeanAbsoluteError()])

# ==========================================
# 4. Model summary
# ==========================================
model.summary()

# ==========================================
# 5. Train model
# ==========================================
history = model.fit(
    X_train, y_train,
    epochs=20,  # Example number of epochs, adjust as needed
    batch_size=50, # Adjust batch size as required
    validation_split=0.2 # Example validation split
)
```



```
    )

    # ==========================================
    # 6. Evaluate model
    # ==========================================
    mae = model.evaluate(X_test, y_test, verbose=0)[1]
    print(f"Mean absolute error: {mae:.5f}")
```

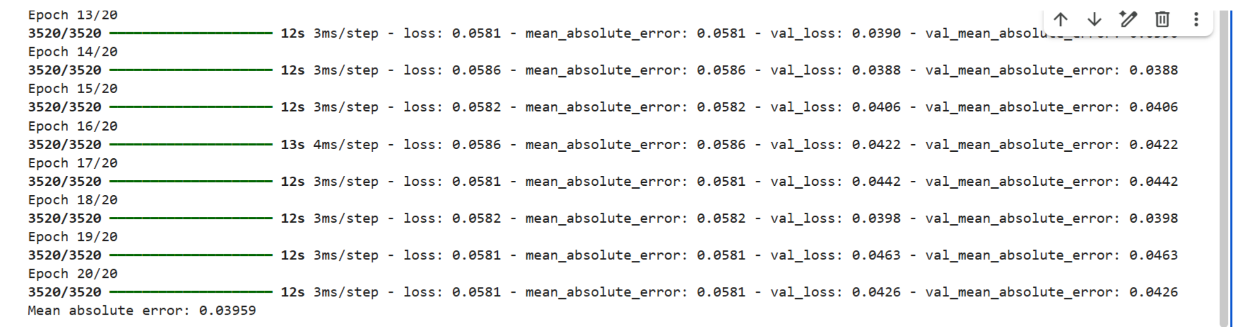
Output:-



```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input_s
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv1d_2 (Conv1D)	(None, 46, 64)	1,664
max_pooling1d_2 (MaxPooling1D)	(None, 23, 64)	0
conv1d_3 (Conv1D)	(None, 19, 128)	41,088
max_pooling1d_3 (MaxPooling1D)	(None, 9, 128)	0
flatten_1 (Flatten)	(None, 1152)	0
dense_2 (Dense)	(None, 64)	73,792
dropout_1 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 2)	130

```
 Total params: 116,674 (455.76 KB)
 Trainable params: 116,674 (455.76 KB)
 Non-trainable params: 0 (0.00 B)
```



```
Epoch 13/20
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step - loss: 0.0581 - mean_absolute_error: 0.0581 - val_loss: 0.0390 - val_mean_absolu
Epoch 14/20
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step - loss: 0.0586 - mean_absolute_error: 0.0586 - val_loss: 0.0388 - val_mean_absolute_error: 0.0388
Epoch 15/20
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step - loss: 0.0582 - mean_absolute_error: 0.0582 - val_loss: 0.0406 - val_mean_absolute_error: 0.0406
Epoch 16/20
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 13s 4ms/step - loss: 0.0586 - mean_absolute_error: 0.0586 - val_loss: 0.0422 - val_mean_absolute_error: 0.0422
Epoch 17/20
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step - loss: 0.0581 - mean_absolute_error: 0.0581 - val_loss: 0.0442 - val_mean_absolute_error: 0.0442
Epoch 18/20
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step - loss: 0.0582 - mean_absolute_error: 0.0582 - val_loss: 0.0398 - val_mean_absolute_error: 0.0398
Epoch 19/20
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step - loss: 0.0581 - mean_absolute_error: 0.0581 - val_loss: 0.0463 - val_mean_absolute_error: 0.0463
Epoch 20/20
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step - loss: 0.0581 - mean_absolute_error: 0.0581 - val_loss: 0.0426 - val_mean_absolute_error: 0.0426
Mean absolute error: 0.03959
```