LTSF Cttn

Multivariable Time Series Cotton Prices Forecasting.

This notebook executes LTSF (Long Time Series Forecasting) with the following models: Transformer, Fedformer, Informer, Autoformer, Dlinear and Linear.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import os
import torch
import torch.nn.functional as nn
```

```
# pip install pmdarima  # Install just if Arima is going to be used
```

```
pip install einops
```

```
Collecting einops
  Downloading einops-0.7.0-py3-none-any.whl (44 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.6/44.6 kB 2.1 MB/s eta 0:00:00
Installing collected packages: einops
Successfully installed einops-0.7.0
```

```
! pip install -U kaleido
```

```
Collecting kaleido
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.9/79.9 MB 20.5 MB/s eta 0:00:00
Installing collected packages: kaleido
Successfully installed kaleido-0.2.1
```

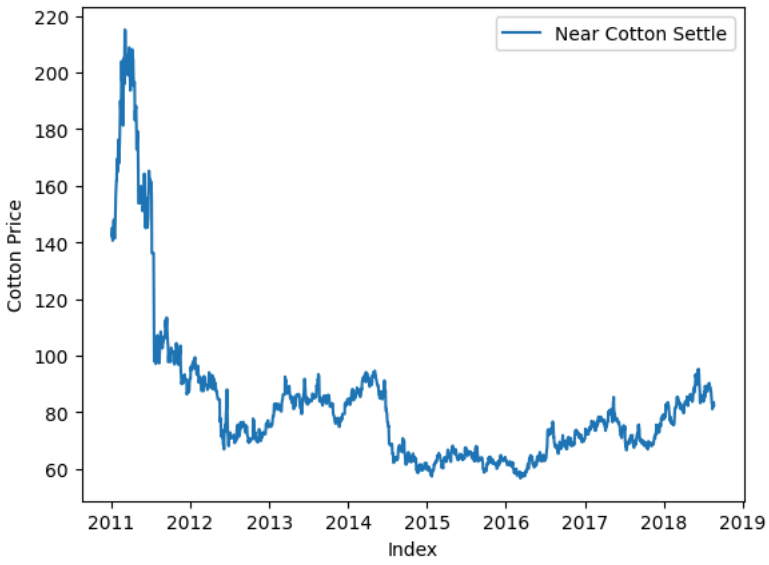
Data Retrieving

```
# Retrieving Daily Cured Data from drive

path = "drive/My Drive/Colab Notebooks/Tesis/Datasets/dfCT.csv"
df = pd.read_csv(path,index_col= [0]).sort_index(ascending=True)
df.index = pd.to_datetime(df.index)
df = df[df.columns].astype(np.float32)
date_range = df.index.min(), df.index.max()
df.index = df.index.rename('date')

## Moving 'Cotton' column to the end like required by the InformerCode
desired_order = df.columns.tolist()  # Make a copy of the column list
desired_order.remove('Cotton')
desired_order.append('Cotton')
desired_order.remove('Value China')
desired_order.remove('Value USA')
df = df[desired_order]
#df.reindex(columns=desired_order, inplace=True)  # Set inplace=True
df = df.iloc[:2000,:]
print('df.shape: ',df.shape, end='   ///    ')
print("DateTime Index Range:", date_range)
df.head(2)
```

```
df.shape:  (2000, 34)   ///    DateTime Index Range: (Timestamp('2011-01-03 00:00:00
```

date	Cotton 2nd Near	Cotton 3rd Near	Cotton 4th Near	Sugar	Corn	Soybeans	Wheat	Rice	Dolla Inde
2011-01-03	0.958509	0.906048	0.777778	32.119999	620.5	1370.25	805.50	14.185	79.3840(

```
# Plotting data
plt.plot(df['Cotton'], label='Near Cotton Settle')  # Plot the Cotton data
plt.xlabel('Index')  # Set x-axis label
plt.ylabel('Cotton Price')  # Set y-axis label
plt.legend()  # Add legend

plt.show()
```



GitHubs Clonning

```
# clonning LTSF github
!git clone https://github.com/cure-lab/LTSF-Linear.git
#!git clone https://github.com/MAZiqing/FEDformer.git
```

```
Cloning into 'LTSF-Linear'...
remote: Enumerating objects: 393, done.
remote: Counting objects: 100% (184/184), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 393 (delta 130), reused 110 (delta 107), pack-reused 209
Receiving objects: 100% (393/393), 5.82 MiB | 28.00 MiB/s, done.
Resolving deltas: 100% (192/192), done.
```

```
import sys
if not 'FEDformer' in sys.path:
    sys.path += ['FEDformer']

# !pip install -r ./Informer2020/requirements.txt
```

```
sys.path.append("/content/LTSF-Linear/FEDformer")
```

```
%%script false --no-raise-error
## ETDataset not required here, but used in foundational work
%cd /content/LTSF-Linear
!git clone https://github.com/zhouhaoyi/ETDataset.git
try:
  os.rename('/content/LTSF-Linear/ETDataset', '/content/LTSF-Linear/dataset')
except OSError as error:
  print(f"Error renaming directory: {error}")
```

```
# saving data retrieved
os.makedirs("/content/LTSF-Linear/dataset", exist_ok=True)
df.to_csv("/content/LTSF-Linear/dataset/dfCT.csv", index=True)
```

```
# Retrieving scripts from Drive
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/LTSF_Scripts/run_LTSF_CS_New.sh /content/LTSF-Line
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/LTSF_Scripts/run_LTSF_CMS_New.sh /content/LTSF-Lin
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/LTSF_Scripts/run_LTSF_CM_New.sh /content/LTSF-Line
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/LTSF_Scripts/run_LookBackLin_CS.sh /content/LTSF-L
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/LTSF_Scripts/run_LookBackDLin_CS.sh /content/LTSF-
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/LTSF_Scripts/run_LookBackStat_CS.sh /content/LTSF-
```

```
# Removing a comment marker from the code
with open("/content/LTSF-Linear/exp/exp_main.py", 'r') as file:
    lines = file.readlines()
lines[300] = lines[300].replace("# ", "")  # Remove comment symbol and space
with open("/content/LTSF-Linear/exp/exp_main.py", 'w') as file:
    file.writelines(lines)
```

```
%%script false --no-raise-error
##### Execute Only if prerunned results are required to be load #####
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/checkpoints /content/LTSF-Linear/FEDformer/
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/logs /content/LTSF-Linear/FEDformer/
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/results /content/LTSF-Linear/FEDformer/
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/result.txt /content/LTSF-Linear/FEDformer/

!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/Base_Line/checkpoints /content/LTSF-Linear/
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/Base_Line/logs /content/LTSF-Linear/
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/Base_Line/results /content/LTSF-Linear/
!cp -r /content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/Base_Line/result.txt /content/LTSF-Linear/
```

Functions

```
# Retrieving the name of the last test
def retrieve_test_name(last_n_test=1):
  with open("./result.txt", "r") as file:
    lines = [line for line in file if line.startswith("Cttn")]
  # Check if there are at least two lines in the file
  if len(lines) >= 1:
    # Access the second-to-last line (assuming the file is not empty)
    setting = lines[-last_n_test].strip()
    # print(f"setting: {setting}")
  else:
    setting = "The file is empty or has only one line."
    # print("The file is empty or has only one line.")
  return setting
```

```
# Retrieving a names list of tests
def retrieve_tests_list(features='S'):
  settings_list = []
  with open("./result.txt", "r") as file:
    settings_list = [line.rstrip() for line in file if line.startswith("Cttn") and len(line.split('_')) > 5 and lin
  settings_list = sorted(list(set(settings_list)), reverse=True)
  return settings_list
```

```
# Plotting test
def plot_results(setting, dfresults):

  preds = np.load('./results/'+setting+'/pred.npy')[:,:,-1:]
  trues = np.load('./results/'+setting+'/true.npy')[:,:,-1:]
  mse = torch.nn.functional.mse_loss(torch.tensor(preds), torch.tensor(trues))
  mse = nn.mse_loss(torch.tensor(preds), torch.tensor(trues)).mean().item()
  mae = nn.l1_loss(torch.tensor(preds), torch.tensor(trues)).mean().item()

  model = setting.split('_')[1] ; features = setting.split('_')[5]
  seq_len = setting.split('_')[6][2:] ;  label_len = setting.split('_')[7][2:] ;  pred_len = setting.split('_')[8][
  model_type = {'M':'multivariate predict multivariate', 'S':'univariate predict univariate', 'MS':'multivariate pr

  step = int(float(preds.shape[1])*6/6) if preds.shape and preds.shape[1] else 60  # change *6/6 to modify predicti
  num_samples = (preds.shape[0] // step) + 1  # Number of samples (including the first row)
  all_preds = []
  all_trues = []
  for i in range(num_samples):
      start_index = i * step
      sampled_preds = preds[start_index, :, 0]  # Select specific row and squeeze last dimension
      sampled_trues = trues[start_index, :, 0]
      all_preds.append(sampled_preds)
      all_trues.append(sampled_trues)

  x_labels = range(preds.shape[1])
  fig, ax = plt.subplots(figsize=(12, 6))  # Adjust the figure size as needed
  for i in range(num_samples):
      offset = range(i * step -1,i * step -1 + len( all_preds[i]))  # Calculate offset for each sample
      offset_list = range(i * step -1, i * step -1 + len( all_preds[i]))  # Create a shifted list using list compre
      if i == 0:
          ax.plot(offset_list, all_preds[i], label=f'Trues (Timestep (all))', color='black')
      ax.plot(offset_list, all_trues[i], color='black')  # Plot all trues in black
      ax.plot(offset_list, all_preds[i], label=f'Preds (Timestep {i * step})')

  ax.set_xlabel(f'Feature Index \n {setting} ')
  ax.set_ylabel('Value')

  ax.set_title(f'{model} ( {model_type[features[2:]]} )  \n mse = ({mse:.6f}), mae = ({mae:.6f}), seq_len = {seq_le
  ax.legend()
  ax.grid(True)

  experiment = f'{model}_{features}_{str(seq_len)}_{str(label_len)}_{str(pred_len)}'

  # plt.savefig(f'/content/drive/MyDrive/ThesisGitHub/LTSF/LTSF_CheckPoints/Graphs/{experiment}.png')
  print('\n\n')
  plt.show()
  spaces5 = " " * 5
  spaces2 = " " * 2
  print(f'{spaces5}{setting}')
  print(f'{spaces2} mse = ({mse:.6f}), mae = ({mae:.6f}), preds.shape {preds.shape}, trues.shape {trues.shape}, lab

  data = {
      'model': [model], 'features': [features],
      'seq_len': [str(seq_len)], 'label_len': [str(label_len)],
      'pred_len': [str(pred_len)],  'mse': [mse], 'mae': [mae]
  }
  new_row = pd.DataFrame(data, index=[experiment])

  if dfresults.empty:
      dfresults = new_row.copy()
  else:
      dfresults = dfresults.combine_first(new_row)

  return dfresults
```

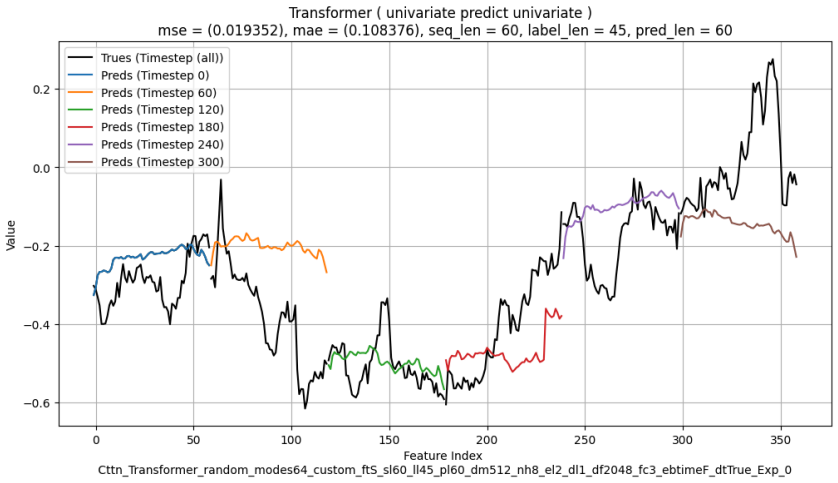
Models

```
dfresults = pd.DataFrame([]).rename_axis('index', axis=0)
```

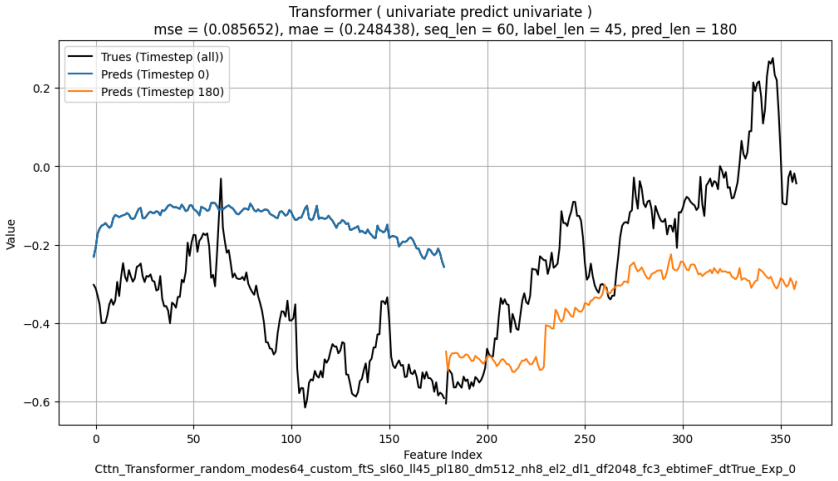
Univariate

```
%cd /content/LTSF-Linear/FEDformer
!chmod +x ./scripts/run_LTSF_CS_New.sh
!./scripts/run_LTSF_CS_New.sh > /dev/null

for test in retrieve_tests_list('S'):
  dfresults = plot_results(test, dfresults)
```

```
/content/LTSF-Linear/FEDformer
```




```
Cttn_Transformer_random_modes64_custom_ftS_sl60_ll45_pl60_dm512_nh8_el2_dl1_d
mse = (0.019352), mae = (0.108376), preds.shape (320, 60, 1), trues.shape (320,
```




```
Cttn_Transformer_random_modes64_custom_ftS_sl60_ll45_pl180_dm512_nh8_el2_dl1_
mse = (0.085652), mae = (0.248438), preds.shape (192, 180, 1), trues.shape (192
```