National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Martin Orr
Student ID: x1915562

School of Computing
National College of Ireland

Supervisor:	Catherine Mulwa
Industry Supervisor:	XXX

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Martin Orr
Student ID:	x1915562
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Catherine Mulwa
Submission Due Date:	16/08/2021
Project Title:	Configuration Manual
Word Count:	439
Page Count:	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	22nd September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Martin Orr
x1915562

1 Python Installs and general info

Please move all the files into the same working directory before you start. I apologise for putting the data files in a folder

Install Jupyter notebook for Python then follow the below

Install Numpy, Pandas, SciPy, SciKit-Learn, QuickDa, MatPlotLib, seaborn, ipywidgets, functools, Keras,tensorflow

There are 3 notebooks that are duplicates of the same notebook. One set of 3 is for one store and another all stores. The only thing different is the CSV names in and out

2 Pulling .grib and .bufr files from the ECMWF

The CSV files featuring compiled .grib and .bufr files have already been made and uploaded with this project.

An API key is needed to pull the data from the API.

3 Data Cleaning and Preprocessing

As long as the Juptyer notebook file is opened in the same location as all the data files then this can just run cell after cell. I've tried to remove any cells that are not working but some still remain

3. Merging Datasets

In [96]:
```
data_first_merge = pd.merge(energy_data,forecast_temp_csv,how='left',left_on=[energy_data.index],right_on=[forecast_ter
```

In [97]:
```
data_first_merge.set_index('key_0',inplace=True)
data_first_merge.index.name = 'timestamp'
```

In [98]:
```
data = pd.merge(data_first_merge,historic_temp_csv,how='left',left_on=[data_first_merge.index],right_on=[historic_temp_
```

In [99]:
```
data.set_index('key_0',inplace=True)
data.index.name = 'timestamp'
```

In [100]:
```
data = data.drop(columns={'utility','step', 'latitude','longitude', 'number', 'time', 'isobaricInhPa','localLatitude',
```

In [101]:
```
data = data[data.index < pd.to_datetime('2020-03-12 00:00:00')] #Consumption profile changed after Covid so this data
```

In [102]:
```
print(data['forecast_temp'].mean())
print(data['historic_temp'].mean())
print(data['forecast_temp'].max())
print(data['historic_temp'].max())
```

```
283.4979210340144
282.88560972740424
293.0515
301.3666666666667
```

4. Data Cleaning

In [104]:
```
data['forecast_temp'] = data['forecast_temp'].interpolate(method='linear', limit=8)#axis={'forecast_temp','historic_te
```

In [105]:
```
data['historic_temp'] = data['historic_temp'].interpolate(method='linear', limit=8)#axis={'forecast_temp','historic_tem
```

In [106]:
```
uped_daily = data.resample('D')
a_daily = grouped_daily.sum()
a_daily['sum'] = data_daily['hvac_consumption']+ data_daily['refrigeration_consumption']+data_daily['remainder_consumpt
a_daily['forecast_temp']=data_daily['forecast_temp']/48 #divide sum by 48 to get average temperature for the day
a_daily['historic_temp']=data_daily['historic_temp']/48 #divide sum by 48 to get average temperature for the day

a_daily[610:700]
```

Out[106]:

timestamp	hvac_consumption	refrigeration_consumption	mains_consumption	mains_value_count	remainder_consumption	forecast_temp	historic_temp	sum
2019-09-03	155.2	1478.3	3381	48	1747.5	287.897167	288.170486	3381.0
2019-09-04	131.1	1547.2	3403	48	1724.7	285.802156	285.342535	3403.0
2019-09-05	184.2	1634.3	3553	48	1734.5	284.948566	285.710417	3553.0
2019-09-06	169.4	1759.7	3679	48	1749.9	285.860401	286.919271	3679.0
2019-09-07	117.3	2183.3	4015	48	1714.4	285.658455	286.320833	4015.0

Figure 1: Data merge and interpolations

4 Regression Analysis

As long as the Juptyer notebook file is opened in the same location as all the data files then this can just run cell after cell. I've tried to remove any cells that are not working but some still remain

```
99]: data = pd.read_csv('cleaned_data.csv', parse_dates=True, index_col='timestamp')
     #data = pd.read_csv('cleaned_data.csv', parse_dates=True, index_col='timestamp')
     data['forecast_temp'] = data['forecast_temp'].shift(-48)

     data = clean(data, method="duplicates")
     data['mains_consumption'] = data['mains_consumption'].shift(-48)
     data = data[data.index < pd.to_datetime('2019-10-12 00:00:00')] #Consumption profile changed after Covid so this data
     data.tail(20)
```

Figure 2: Data into regression

```
: grouped_daily = data.resample('D')
  data_daily = grouped_daily.sum()
  data_daily['sum'] = data_daily['hvac_consumption']+ data_daily['refrigeration_consumption']+data_daily['remainder_consu
  data_daily['forecast_temp']=data_daily['forecast_temp']/48 #divide sum by 48 to get average temperature for the day
  data_daily['historic_temp']=data_daily['historic_temp']/48 #divide sum by 48 to get average temperature for the day
  data_daily['day_0']=data_daily['day_0']/48
  data_daily['day_1']=data_daily['day_1']/48
  data_daily['day_2']=data_daily['day_2']/48
  data_daily['day_3']=data_daily['day_3']/48
  data_daily['day_4']=data_daily['day_4']/48
  data_daily['day_5']=data_daily['day_5']/48
  data_daily['day_6']=data_daily['day_6']/48
  data_daily = data_daily.drop(data_daily[data_daily.mains_consumption == 0].index)
  data_daily.tail(25)
```

Figure 3: Regression Resampling

5 LSTM and CNN

As long as the Juptyer notebook file is opened in the same location as all the data files, this can run cell after cell. I've tried to remove any cells that are not working but some still remain.

These models were based on [1]

Avoid running 1D CNNs and univariate LSTM as it's not in the research. Apologies for not deleting it

Each model trains the forecast first, then the historical version

This function is the plot for each model in the CNN and LSTM notebook.

[1] https://towardsdatascience.com/time-series-forecasting-with-2d-convolutions-4f1a0f33dff6

```
In [892]: #Evaluation & Results Function
def CNN_eval(forecast_trained_model,historic_trained_model,y_test,data_test_wide_forecast,data_test_wide_historic):

    predictions_forecast = forecast_trained_model.predict(data_test_wide_forecast)
    test_r2_forecast = r2_score(y_test, predictions_forecast)
    print ("half hourly R2 for day ahead is with forecast data is:", test_r2_forecast)

    predictions_historic = historic_trained_model.predict(data_test_wide_historic)
    test_r2_historic = r2_score(y_test, predictions_historic)
    print ("half hourly R2 for day ahead is with historic data is:", test_r2_historic)

    comparison_df = pd.DataFrame()
    comparison_df['Actual'] = y_test[:,0]
    comparison_df['forecast_Predict'] = predictions_forecast[:,0]
    comparison_df['historic_Predict'] = predictions_historic[:,0]
    comparison_df[0:336].plot(ylabel = 'kWh') # 1 week
    comparison_df[-800:].plot(ylabel = 'kWh')


    comparison_df = comparison_df[::-1] #Reverses the data so that each 48 rows from the first corresponds to a calend

    comparison_df_daily = comparison_df.groupby(comparison_df.index // 48).sum()

    comparison_df_daily = comparison_df_daily[::-1]


    comparison_df_daily = comparison_df_daily[1:] #Drops first row as it is not a full day

    comparison_df_daily.plot(ylabel = 'kWh',xlabel = 'days')


    forecast_daily_rmspe = (np.sqrt(np.mean(np.square((comparison_df_daily['Actual'] - comparison_df_daily['forecast_P
    print('forecast_daily_rmspe is:',forecast_daily_rmspe)

    historic_daily_rmspe = (np.sqrt(np.mean(np.square((comparison_df_daily['Actual'] - comparison_df_daily['historic_P
    print('historic_daily_rmspe is:',historic_daily_rmspe)

    forecast_RMSLE = math.sqrt(mean_squared_log_error(comparison_df['Actual'], comparison_df['forecast_Predict']))
    print('forecast_RMSLE is:',forecast_RMSLE)

    historic_MSLE = math.sqrt(mean_squared_log_error(comparison_df['Actual'], comparison_df['historic_Predict']))
    print('historic_RMSLE is:',historic_MSLE)

    daily_forecast_RMSLE = math.sqrt(mean_squared_log_error(comparison_df_daily['Actual'], comparison_df_daily['forecas
    print("daily_forecast_RMSLE is:",daily_forecast_RMSLE)

    daily_historic_RMSLE = math.sqrt(mean_squared_log_error(comparison_df_daily['Actual'], comparison_df_daily['histor
    print("daily_historic_RMSLE is:",daily_historic_RMSLE)

    daily_test_r2_forecast = r2_score(comparison_df_daily['Actual'], comparison_df_daily['forecast_Predict'])
    print('forecast daily  R squared is:' , daily_test_r2_forecast)

    daily_test_r2_historic = r2_score(comparison_df_daily['Actual'], comparison_df_daily['historic_Predict'])
    print('historic daily R squared is:' , daily_test_r2_historic)

In [616]: data = pd.read_csv('cleaned_data.csv', parse_dates=True, index_col='timestamp')
data['forecast_temp'] = data['forecast_temp'].shift(-48)
data = data[data.index < pd.to_datetime('2020-03-11 00:00:00')] #Consumption profile changed after Covid so this data
```

Figure 4: Data merge and interpolations

Output from above is below.

```
In [887]: CNN_eval(m2_forecast,m2_historic,y_test,data_test_wide_forecast,data_test_wide_historic) #window 144

half hourly R2 for day ahead is with forecast data is: 0.94058702427595
half hourly R2 for day ahead is with historic data is: 0.9371334297559488
forecast_daily_rmspe is: 3.4745775555520444
historic_daily_rmspe is: 3.4994613692301715
forecast_RMSLE is: 0.04679828871561291
historic_RMSLE is: 0.04777052547820176
daily_forecast_RMSLE is: 0.034571462898981484
daily_historic_RMSLE is: 0.03449410282053451
forecast daily  R squared is: 0.3690291687095487
historic daily R squared is: 0.377386762467919
```

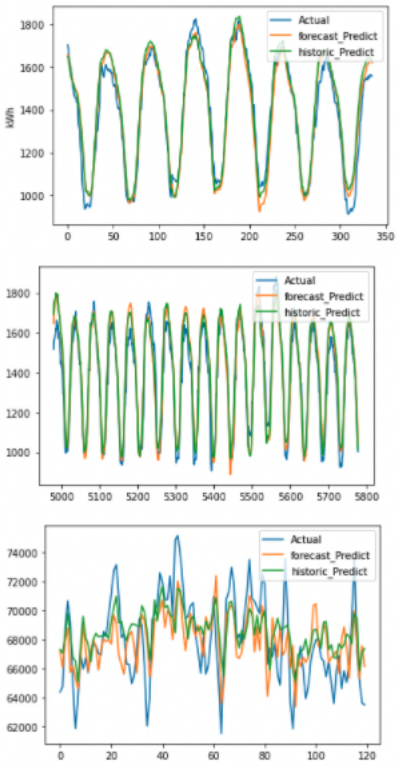

Figure 5: Data merge and interpolations

```
In [617]: # Our handy function for converting our dataset into the correct format

def window_data(df, window, feature_col_number, target_col_number):
    """
    This function accepts the column number for the features (X) and the target (y).
    It chunks the data up with a rolling window of Xt - window to predict Xt.
    It returns two numpy arrays of X and y.
    """
    X = []
    y = []
    for i in range(0,len(df) - window - 1):
        if i + 2*window > (len(df)- window -1):
            break
        features = df.iloc[i : (i + window), feature_col_number].values
        target = df.iloc[(i + window + 48 ),target_col_number]
        #print(features)
        #print (i)
        #print("----")
        #print(target)
        X.append(features)
        y.append(target)
    return np.array(X), np.array(y).astype(np.float64).reshape(-1, 1)

In [618]: data.columns
```

Figure 6: Makes Window of data for LSTM and CNN

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X = scaler.fit_transform(X)

historic_temp = scaler.fit_transform(historic_temp)
forecast_temp = scaler.fit_transform(forecast_temp)
hvac_consumption = scaler.fit_transform(hvac_consumption)
refrigeration_consumption = scaler.fit_transform(refrigeration_consumption)
remainder_consumption = scaler.fit_transform(remainder_consumption)
```

Figure 7: Dataset scaler

The below makes all the datasets for CNNs and LSTM.

Historic train/test

In [839]:

```
split = int(0.8 * len(X))
X_train_historic = X[: split - 1]
X_test_historic = X[split:]

historic_temp_train = historic_temp[: split - 1]

historic_temp_test = historic_temp[split:]

y_train = y[: split - 1]
y_test = y[split:]

hvac_consumption_train = hvac_consumption[: split - 1]
refrigeration_consumption_train = refrigeration_consumption[: split - 1]
remainder_consumption_train = remainder_consumption[: split - 1]
day_0_train = day_0[: split - 1]
day_1_train = day_1[: split - 1]
day_2_train = day_2[: split - 1]
day_3_train = day_3[: split - 1]
day_4_train = day_4[: split - 1]
day_5_train = day_5[: split - 1]
day_6_train = day_6[: split - 1]

hvac_consumption_test = hvac_consumption[split:]
refrigeration_consumption_test = refrigeration_consumption[split:]
remainder_consumption_test = remainder_consumption[split:]
day_0_test = day_0[split:]
day_1_test = day_1[split:]
day_2_test = day_2[split:]
day_3_test = day_3[split:]
day_4_test = day_4[split:]
day_5_test = day_5[split:]
day_6_test = day_6[split:]

#For univariate models
X_train_historic = X_train_historic.reshape((X_train_historic.shape[0], X_train_historic.shape[1], 1))

#For multivariate models
historic_temp_train = historic_temp_train.reshape((historic_temp_train.shape[0], historic_temp_train.shape[1], 1))
hvac_consumption_train = hvac_consumption_train.reshape((hvac_consumption_train.shape[0], hvac_consumption_train.shape[
refrigeration_consumption_train = refrigeration_consumption_train.reshape((refrigeration_consumption_train.shape[0], re
remainder_consumption_train = remainder_consumption_train.reshape((remainder_consumption_train.shape[0], remainder_cons
day_0_train = day_0_train.reshape((day_0_train.shape[0], day_0_train.shape[1], 1))
day_1_train = day_1_train.reshape((day_1_train.shape[0], day_1_train.shape[1], 1))
day_2_train = day_2_train.reshape((day_2_train.shape[0], day_2_train.shape[1], 1))
day_3_train = day_3_train.reshape((day_3_train.shape[0], day_3_train.shape[1], 1))
day_4_train = day_4_train.reshape((day_4_train.shape[0], day_4_train.shape[1], 1))
day_5_train = day_5_train.reshape((day_5_train.shape[0], day_5_train.shape[1], 1))
day_6_train = day_6_train.reshape((day_6_train.shape[0], day_6_train.shape[1], 1))

data_train_historic = np.concatenate((historic_temp_train, hvac_consumption_train, refrigeration_consumption_train, rer
                                      day_0_train,day_1_train,day_2_train,day_3_train,day_4_train,day_5_train,day_6_tra


#For univariate models
X_test_historic = X_test_historic.reshape((X_test_historic.shape[0], X_test_historic.shape[1], 1))

#For multivariate models
historic_temp_test = historic_temp_test.reshape((historic_temp_test.shape[0], historic_temp_test.shape[1], 1))
hvac_consumption_test = hvac_consumption_test.reshape((hvac_consumption_test.shape[0], hvac_consumption_test.shape[1],
refrigeration_consumption_test = refrigeration_consumption_test.reshape((refrigeration_consumption_test.shape[0], refri
remainder_consumption_test = remainder_consumption_test.reshape((remainder_consumption_test.shape[0], remainder_consump
day_0_test = day_0_test.reshape((day_0_test.shape[0], day_0_test.shape[1], 1))
day_1_test = day_1_test.reshape((day_1_test.shape[0], day_1_test.shape[1], 1))
day_2_test = day_2_test.reshape((day_2_test.shape[0], day_2_test.shape[1], 1))
day_3_test = day_3_test.reshape((day_3_test.shape[0], day_3_test.shape[1], 1))
day_4_test = day_4_test.reshape((day_4_test.shape[0], day_4_test.shape[1], 1))
day_5_test = day_5_test.reshape((day_5_test.shape[0], day_5_test.shape[1], 1))
day_6_test = day_6_test.reshape((day_6_test.shape[0], day_6_test.shape[1], 1))

data_test_historic = np.concatenate((historic_temp_test, hvac_consumption_test, refrigeration_consumption_test, remainc
                                     day_0_test,day_1_test,day_2_test,day_3_test,day_4_test,day_5_test,day_6_test), axis
```

Figure 8: Dataset splitter and concatenater

Multi-variate LSTM

forecast

In [872]:
```
multi_ls_model_forecast = basic_LSTM(window_size=window_size, n_features=data_train_forecast.shape[2])
```

```
WARNING:tensorflow:Layer lstm_5 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic
GPU kernel as fallback when running on GPU.
```

In [873]:
```
# This will be true if data_train has any nans
summa = np.sum(data_train_forecast)
check_nan = np.isnan(summa)
check_nan
```

Out[873]:
```
False
```

In [874]:
```
multi_ls_history_forecast = multi_ls_model_forecast.fit(data_train_forecast ,y_train, epochs=5, shuffle=False, batch_si
```

```
Epoch 1/5
2021-08-16 08:39:20.687014: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:112] Plugin opti
mizer for device_type GPU is enabled.
232/232 [==============================] - 539s 2s/step - loss: 6564157.1406
Epoch 2/5
232/232 [==============================] - 436s 2s/step - loss: 2298820.8315
Epoch 3/5
232/232 [==============================] - 409s 2s/step - loss: 1647638.0290
Epoch 4/5
232/232 [==============================] - 413s 2s/step - loss: 208242.6498
Epoch 5/5
232/232 [==============================] - 410s 2s/step - loss: 122402.2762
```

In [875]:
```
multi_ls_model_forecast.summary()
# Note: if we're getting nan here, re-check the dataset to see if we're getting any bad data there
```

```
Model: "sequential_60"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
module_wrapper_312 (ModuleWr (None, 100)               44800
_________________________________________________________________
module_wrapper_313 (ModuleWr (None, 100)               0
_________________________________________________________________
module_wrapper_314 (ModuleWr (None, 1500)              151500
_________________________________________________________________
module_wrapper_315 (ModuleWr (None, 100)               150100
_________________________________________________________________
module_wrapper_316 (ModuleWr (None, 1)                 101
=================================================================
Total params: 346,501
Trainable params: 346,501
Non-trainable params: 0
_________________________________________________________________
```

In [876]:
```
plt.plot(multi_ls_history_forecast.history["loss"])
plt.title("loss function - LSTM Multivariate")
plt.legend(["loss"])
plt.show()
```

Figure 9: Multi-variate LSTM

CNN 2D

Here we create a conv2d with a filter shape of 1 timestep by fsize features, basically turning the convolution window on the Z-axis in our slides.

In [841]:
```
def basic_conv2D(n_filters=10, fsize=4, window_size=window_size, n_features=2):
    new_model = keras.Sequential()
    # Hypothetically, we could also tune the padding and activation here.
    new_model.add(tf.keras.layers.Conv2D(n_filters, (1,fsize), padding="same", activation="relu", input_shape=(window_s
    # Flatten will take our convolution filters and lay them out end to end so our dense layer can predict based on the
    new_model.add(tf.keras.layers.Flatten())
    new_model.add(tf.keras.layers.Dense(1000, activation='relu'))
    new_model.add(tf.keras.layers.Dense(100))
    new_model.add(tf.keras.layers.Dense(1))
    new_model.compile(optimizer="adam", loss="mean_squared_error")
    return new_model
```

Forecast

In [842]:
```
data_train_wide_forecast = data_train_forecast.reshape((data_train_forecast.shape[0], data_train_forecast.shape[1], dat
data_test_wide_forecast = data_test_forecast.reshape((data_test_forecast.shape[0], data_test_forecast.shape[1], data_te
```

In [843]:
```
data_train_wide_forecast.shape
```

Out[843]:
```
(23112, 48, 11, 1)
```

In [844]:
```
2_forecast = basic_conv2D(n_filters=24, fsize=2, window_size=window_size, n_features=data_train_wide_forecast.shape[2])
```

In [845]:
```
m2_forecast_hist = m2_forecast.fit(data_train_wide_forecast, y_train, epochs=10)
```

```
Epoch 1/10
  1/723 [..............................] - ETA: 9:19 - loss: 1907890.5000
2021-08-16 07:36:12.585323: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:112] Plugin opti
mizer for device_type GPU is enabled.
723/723 [==============================] - 17s 23ms/step - loss: 167481.9559
Epoch 2/10
723/723 [==============================] - 15s 20ms/step - loss: 5988.8553
Epoch 3/10
723/723 [==============================] - 15s 20ms/step - loss: 5586.8411
Epoch 4/10
723/723 [==============================] - 15s 21ms/step - loss: 5699.1077
Epoch 5/10
723/723 [==============================] - 15s 20ms/step - loss: 6045.2573
Epoch 6/10
723/723 [==============================] - 15s 20ms/step - loss: 5749.5226
Epoch 7/10
723/723 [==============================] - 14s 20ms/step - loss: 5394.3813
Epoch 8/10
723/723 [==============================] - 15s 20ms/step - loss: 5561.7922
Epoch 9/10
723/723 [==============================] - 14s 20ms/step - loss: 5575.8296
Epoch 10/10
723/723 [==============================] - 15s 20ms/step - loss: 5358.2616
```

In [846]:
```
m2_forecast.summary()
```

Figure 10: CNN

References

The inspiration for CNN and LSTM models is linked in footnote 1. Here is the link also: https://towardsdatascience.com/time-series-forecasting-with-2d-convolutions-4f1a0f33dff6