ccr_model-out-cmp

June 18, 2025

[415]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import mean_squared_error, r2_score,␣
↪root_mean_squared_error, mean_absolute_percentage_error,␣
↪median_absolute_error
```

[416]:
```
pop = "ccr" # ccr/jla/tsz/nrv
```

[417]:
```
match pop:
    case "ccr":
        ground = pd.read_csv("data/ccr_hourly_data.csv")
        stderr = pd.read_csv("data/ccr_std_error.csv")
        pred = pd.read_csv("./garisom/02_program_code/out/
↪timesteps_output_POFR_CCR-COL_DBG.csv").iloc[:-1]
    case "jla":
        ground = pd.read_csv("data/jla_hourly_data.csv")
        stderr = pd.read_csv("data/jla_std_error.csv")
        pred = pd.read_csv("./garisom/02_program_code/out/
↪timesteps_output_POFR_JLA-JAK_DBG.csv").iloc[:-1]
    case "tsz":
        ground = pd.read_csv("data/tsz_hourly_data.csv")
        stderr = pd.read_csv("data/tsz_std_error.csv")
        pred = pd.read_csv("./garisom/02_program_code/out/
↪timesteps_output_POFR_TSZ-SAN_DBG.csv").iloc[:-1]
    case "nrv":
        ground = pd.read_csv("data/nrv_hourly_data.csv")
        stderr = pd.read_csv("data/nrv_std_error.csv")
        pred = pd.read_csv("./garisom/02_program_code/out/
↪timesteps_output_POFR_NRV-NEW_DBG.csv").iloc[:-1]
```

[418]:
```
ground.columns
```

[418]:
```
Index(['year', 'julian-day', 'standard-time', 'solar', 'rain', 'wind', 'T-air',
       'T-soil', 'D-MD', 'GW', 'E-MD', 'P-PD', 'P-MD', 'K-plant'],
      dtype='object')
```

[419]:
```
pred.columns
```

[419]:
```
Index(['year', 'julian-day', 'standard-time', 'solar', 'rain', 'wind', 'T-air',
       'T-soil', 'D-MD', 'P0', 'P1', 'P2', 'P3', 'P4', 'P5', 'P-PD', 'P-MD',
       'E-MD', 'GW', 'leaf-air-vpd', 'leaftemp', 'Anet-la', 'ci', 'PPFD',
       'S-P-MD', 'S-E-MD', 'S-GW', 'S-leaf-air-vpd', 'S-leaftempt',
       'S-Anet-la', 'S-ci', 'S-PPFD', 'S-E-tree', 'Anet-tree', 'Pcrit',
       'Ecrit', 'P-leaf', 'P-stem', 'P-root', 'K-stem', 'K-leaf', 'K-plant',
       'K-xylem', 'K-root-1', 'K-root-2', 'K-root-3', 'K-root-4', 'K-root-5',
       'K-root-all', 'E-root-1', 'E-root-2', 'E-root-3', 'E-root-4',
       'E-root-5', 'water-content', 'water-content-delta', 'end-rain',
       'end-ground-water', 'end-E', 'end-drainage', 'end-soil-evap', 'end-ET',
       'end-Anet-la', 'end-total-water-input', 'end-PLC-plant',
       'end-PLC-xylem', 'end-runoff'],
      dtype='object')
```

[420]:
```
def cmp_pred_to_ground_metrics(columns, ground, pred, start, end):

    fit = {}

    for col in columns:

        col_ground = ground[ground['julian-day'].between(start, end)][col].
↪dropna()
        col_pred = pred[col]

        col_pred = col_pred.loc[col_ground.index]

        mse = mean_squared_error(col_ground, col_pred)
        rmse = root_mean_squared_error(col_ground, col_pred)
        mape = mean_absolute_percentage_error(col_ground, col_pred)
        made = median_absolute_error(col_ground, col_pred)
        r2 = r2_score(col_ground, col_pred)

        fit[col] = {
            'mse' : mse,
            'rmse' : rmse,
            'mape' : mape,
            'made' : made,
            'r2' : r2
        }

        # Plot ground vs pred with fitted line and 1:1 correspondence
        plt.figure(figsize=(8, 6))
        plt.scatter(col_ground, col_pred, label='Data Points', alpha=0.7)
        plt.plot([col_ground.min(), col_ground.max()], [col_ground.min(),␣
↪col_ground.max()], 'r--', label='1:1 Line')

        # Fit a line to the data
```

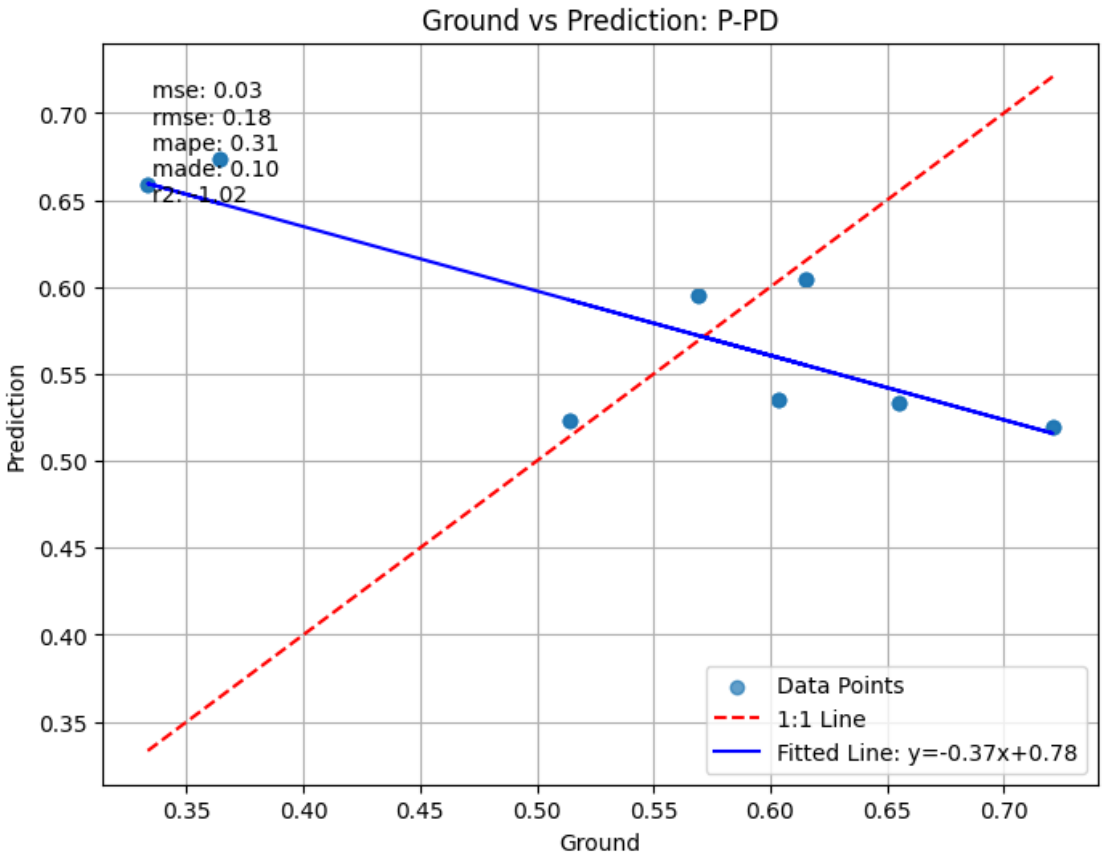
```
        fit_coeff = np.polyfit(col_ground, col_pred, 1)
        fit_line = np.poly1d(fit_coeff)
        plt.plot(col_ground, fit_line(col_ground), 'b-', label=f'Fitted Line:␣
↪y={fit_coeff[0]:.2f}x+{fit_coeff[1]:.2f}')

        # Add metrics as text
        plt.text(0.05, 0.95, ''.join([f'{key}: {value:.2f}\n' for key, value in␣
↪fit[col].items()]),
             transform=plt.gca().transAxes, fontsize=10,␣
↪verticalalignment='top')

        plt.xlabel('Ground')
        plt.ylabel('Prediction')
        plt.title(f'Ground vs Prediction: {col}')
        plt.legend()
        plt.grid(True)
        plt.show()

    return fit
```

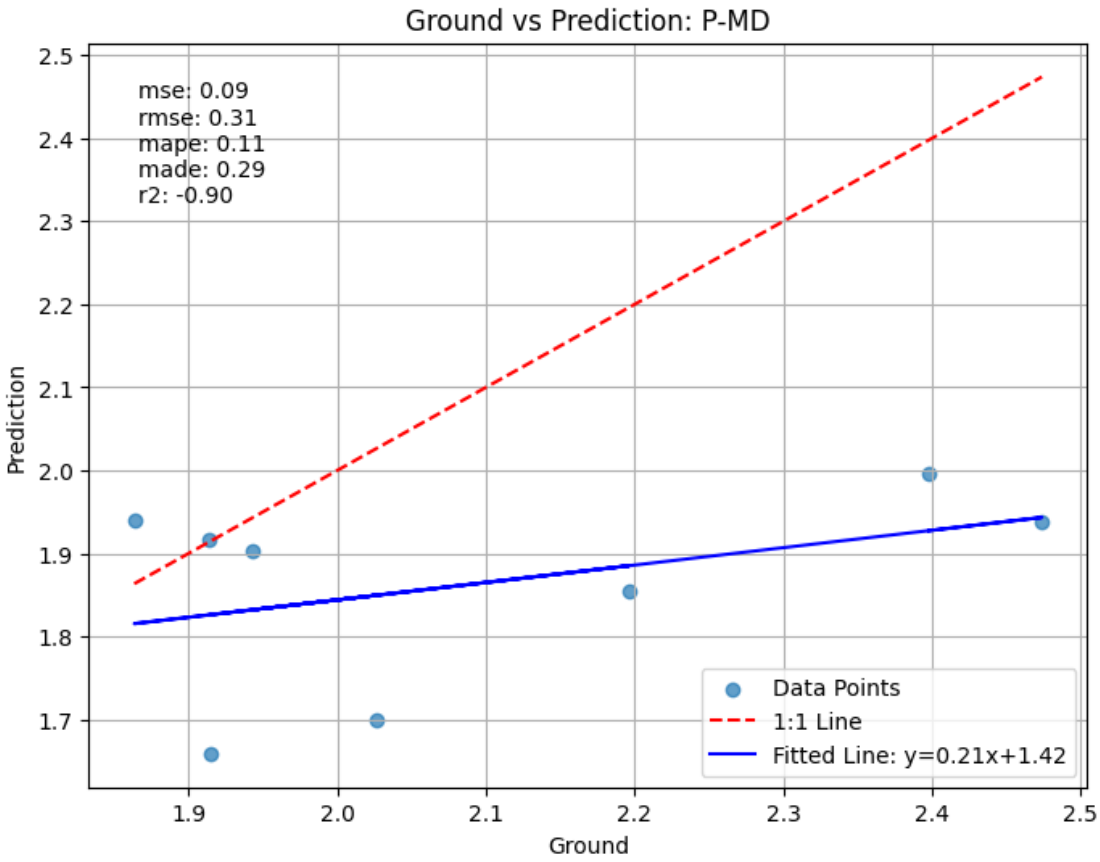
[421]:
```
columns = ['P-PD', 'P-MD', 'GW', 'K-plant', 'E-MD']
start_day = 201
stress_begin = 223
predrought_cutoff = 237
drought = 240           # drought measurement period, collected 3 days after␣
↪initial drought began
post_drought = 241
```

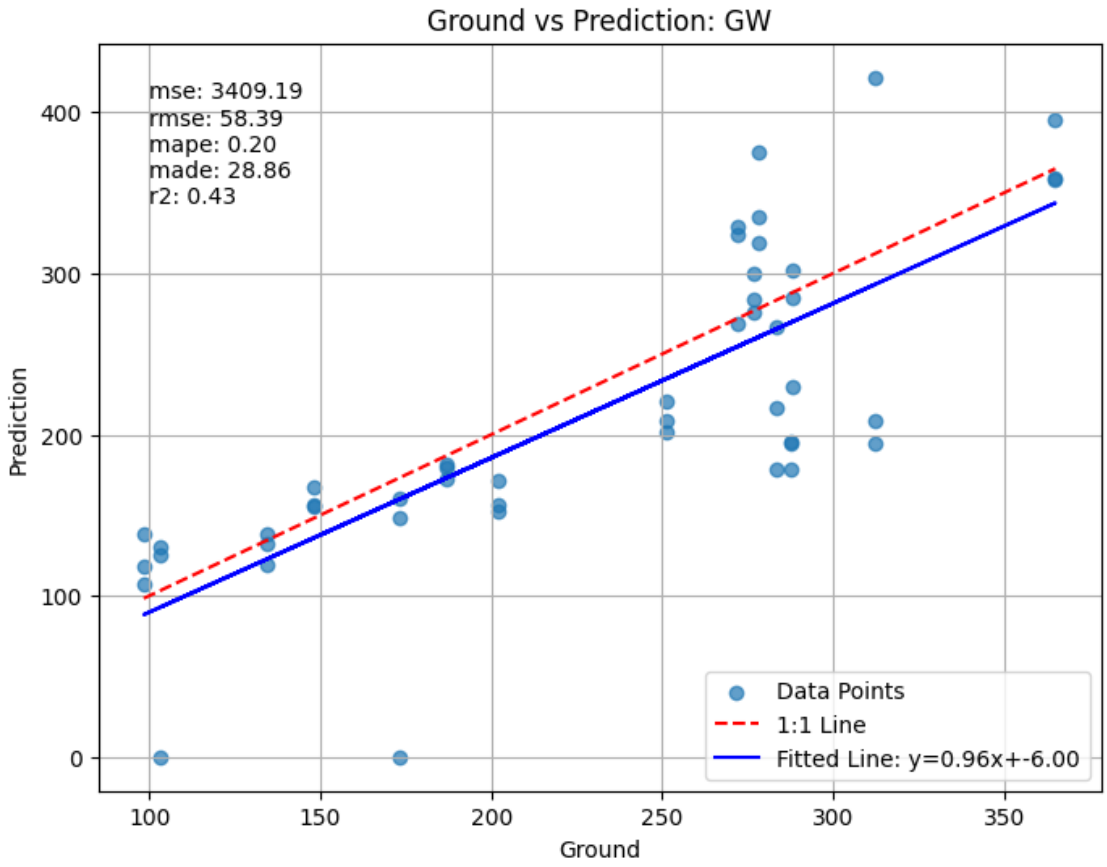
[422]:
```
start_timestep = pred[pred['julian-day'] == start_day].index[0]
predrought_timestep = pred[pred['julian-day'] == predrought_cutoff+1].index[0]
drought_timestep = pred[pred['julian-day'] == drought].index[0]
post_timestep = pred[pred['julian-day'] == post_drought].index[0]
stress_timestep = pred[pred['julian-day'] == stress_begin].index[0]

print(f"Start Timestep: {start_timestep}, Predrought Timestep:␣
↪{predrought_timestep}, Drought Timestep: {drought_timestep}")
```

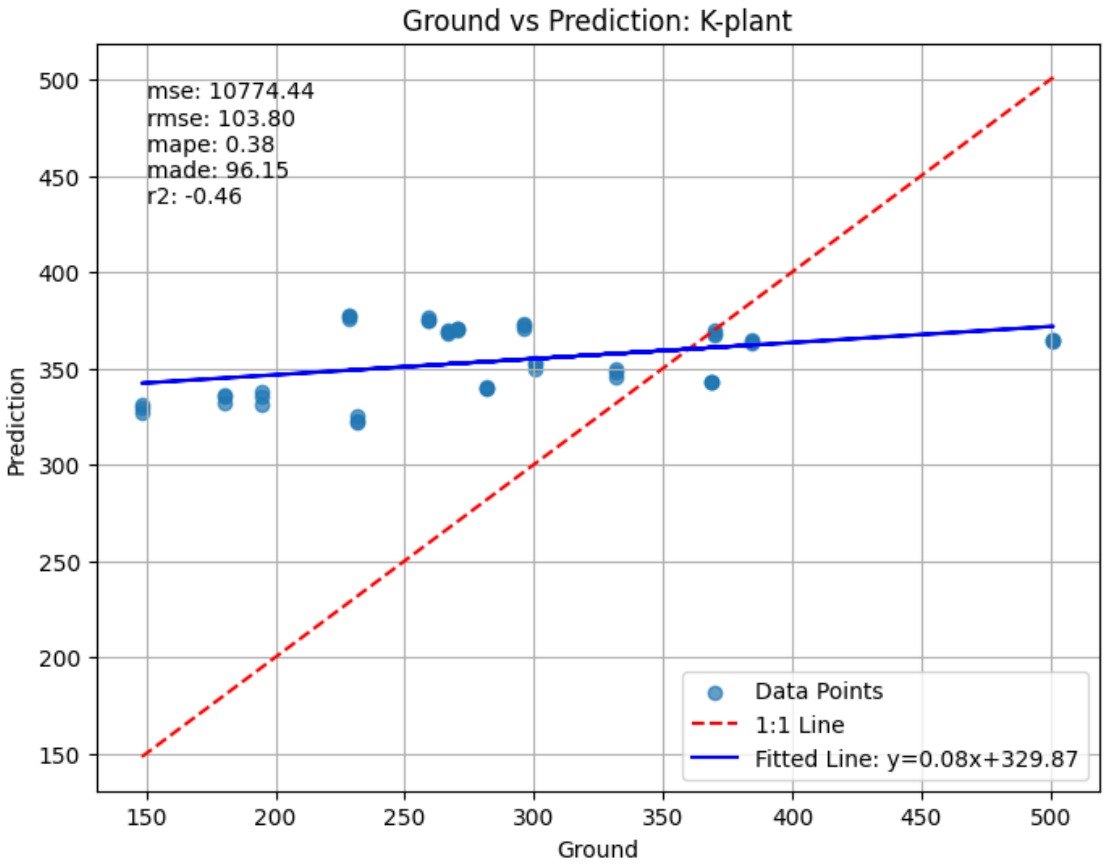
```
Start Timestep: 0, Predrought Timestep: 888, Drought Timestep: 936
```

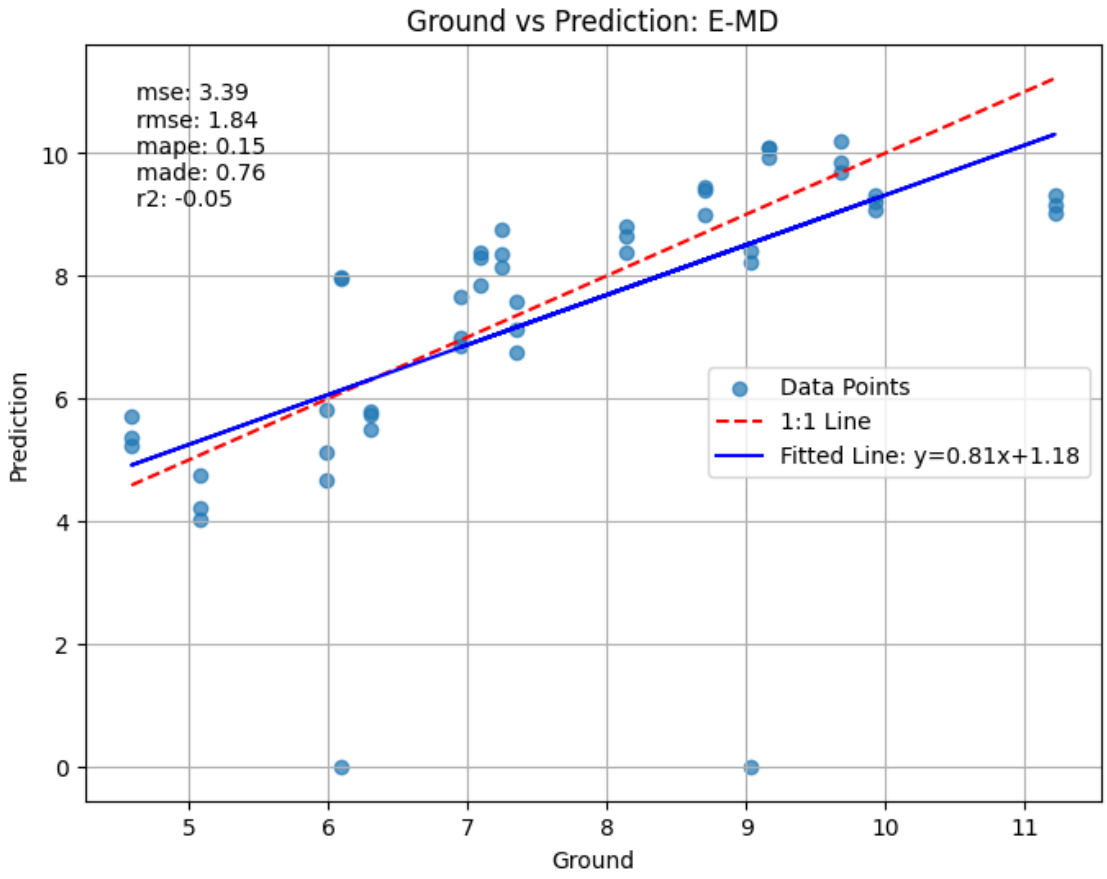
[423]:
```
res = cmp_pred_to_ground_metrics(columns, ground, pred, start_day,␣
↪predrought_cutoff)
```


Ground vs Prediction: P-PD
mse: 0.03
rmse: 0.18
mape: 0.31
made: 0.10
r2: -1.02
Prediction
Ground
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
Data Points
1:1 Line
Fitted Line: y=-0.37x+0.78


Ground vs Prediction: P-MD
mse: 0.09
rmse: 0.31
mape: 0.11
made: 0.29
r2: -0.90
Prediction
Ground
Data Points
1:1 Line
Fitted Line: y=0.21x+1.42
2.5
2.4
2.3
2.2
2.1
2.0
1.9
1.8
1.7
1.9
2.0
2.1
2.2
2.3
2.4
2.5


Ground vs Prediction: GW
mse: 3409.19
rmse: 58.39
mape: 0.20
made: 28.86
r2: 0.43
Prediction
Ground
Data Points
1:1 Line
Fitted Line: y=0.96x+-6.00
0
100
200
300
400
100
150
200
250
300
350


Ground vs Prediction: K-plant
mse: 10774.44
rmse: 103.80
mape: 0.38
made: 96.15
r2: -0.46
Data Points
1:1 Line
Fitted Line: y=0.08x+329.87
Prediction
Ground
500
450
400
350
300
250
200
150
150
200
250
300
350
400
450
500



[424]:
```
res
```

[424]:
```
{'P-PD': {'mse': 0.032845581755281274,
  'rmse': 0.18123350064290342,
  'mape': 0.31065370128815767,
  'made': np.float64(0.09514853460299999),
  'r2': -1.0167656978527955},
 'P-MD': {'mse': 0.09331269448037696,
  'rmse': 0.3054712662107141,
  'mape': 0.11239033769481799,
  'made': np.float64(0.2917019722140557),
  'r2': -0.9010915429468334},
 'GW': {'mse': 3409.1941088650547,
  'rmse': 58.38830455549343,
  'mape': 0.1960786166963492,
  'made': np.float64(28.85572914468034),
  'r2': 0.4335055254727078},
 'K-plant': {'mse': 10774.442076123798,
  'rmse': 103.80001000059585,
```

```
 'mape': 0.3829090629850022,
 'made': np.float64(96.153046830464),
 'r2': -0.4574131580891072},
'E-MD': {'mse': 3.3868673529297886,
 'rmse': 1.840344357159765,
 'mape': 0.1492213744281181,
 'made': np.float64(0.7581452394615003),
 'r2': -0.04779937016323821}}
```

[425]:
```
unit = {
    "P-PD" : "-MPa",
    "P-MD" : "-MPa",
    "GW" : "mmol s-1 m-2 (LA)",
    "E-MD" : "mmol s-1 m-2 (LA)",
    "K-plant" : "kg hr-1 m-2 (BA)"
}
```

[426]:
```
def plot_ground_pred(stop, start=0):

    ground_pre_and_drought = ground[start:stop]
    pred_pre_and_drought = pred[start:stop]
    pred_pre_and_drought.loc[pred_pre_and_drought['GW'] > 1000, 'GW'] = 0

    for col in columns:

        # Make plot
        plt.figure(figsize=(10, 6))
        plt.scatter(range(start, stop), ground_pre_and_drought[col],␣
↪label=f"Ground", color='navy')
        plt.plot(range(start, stop), pred_pre_and_drought[col], color="green",␣
↪alpha=1, label=f'Prediction')
        plt.ylim(bottom=0)
        plt.title(f"Pred versus Ground: {col}")
        plt.xlabel("Timestep")
        plt.ylabel(f"{col} {unit[col]}")
        plt.axvline(x=stress_begin, color='navy', linestyle='--',␣
↪label='Pre-stress Cutoff')
        plt.axvline(x=predrought_timestep, color='navy', linestyle='--',␣
↪label='Predrought Cutoff')
        plt.axvline(x=post_timestep, color='navy', linestyle='--',␣
↪label='Drought Cutoff')
        plt.legend()

    plt.figure(figsize=(10, 6))
    plt.plot(range(start, stop), pred_pre_and_drought['end-PLC-plant'],␣
↪color="green", alpha=1, label=f'Prediction')
    plt.title(f"Predicted PLC")
```

```
    plt.xlabel("Timestep")
    plt.ylabel("PLC %")
    plt.axvline(x=stress_begin, color='navy', linestyle='--', label='Pre-stress␣
↪Cutoff')
    plt.axvline(x=predrought_timestep, color='navy', linestyle='--',␣
↪label='Predrought Cutoff')
    plt.axvline(x=post_timestep, color='navy', linestyle='--', label='Drought␣
↪Cutoff')
    plt.legend()

    # Compute cumulative PLC
    cumulative_plc = pred_pre_and_drought['end-PLC-plant'].cumsum()
    plt.figure(figsize=(10, 6))
    plt.plot(range(start, stop), cumulative_plc, color="blue", alpha=1,␣
↪label=f'Cumulative PLC')
    plt.title(f"Cumulative PLC")
    plt.xlabel("Timestep")
    plt.ylabel("Cumulative PLC %")
    plt.axvline(x=stress_begin, color='navy', linestyle='--', label='Pre-stress␣
↪Cutoff')
    plt.axvline(x=predrought_timestep, color='navy', linestyle='--',␣
↪label='Predrought Cutoff')
    plt.axvline(x=post_timestep, color='navy', linestyle='--', label='Drought␣
↪Cutoff')
    plt.legend()

    # Summary stats
    print("Ground")
    display(ground_pre_and_drought[columns].describe())
    print("Pred")
    display(pred_pre_and_drought[columns].describe())
```

[427]:
```
plot_ground_pred(predrought_timestep)
```

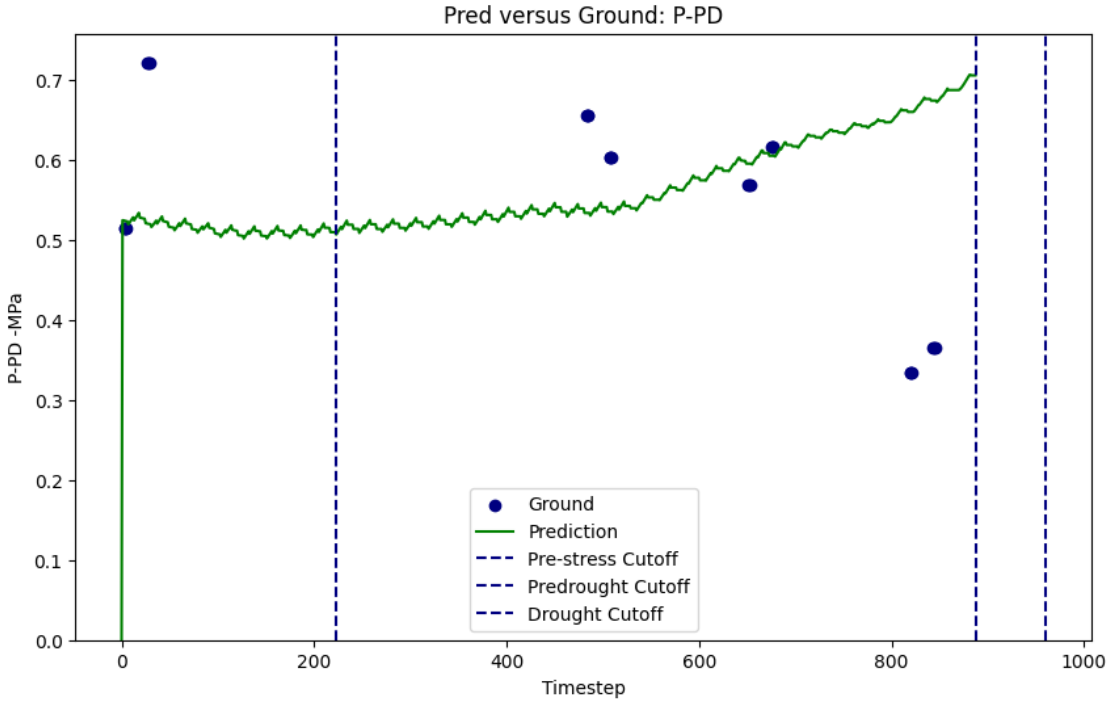
```
Ground
```

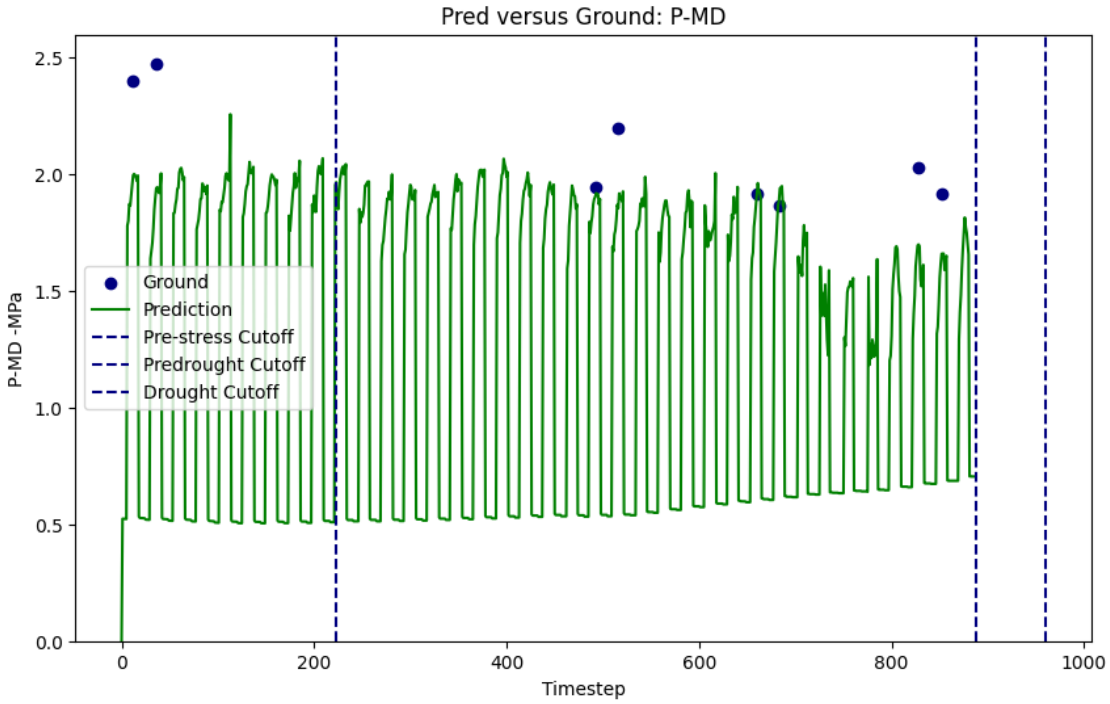
	P-PD	P-MD	GW	K-plant	E-MD
count	24.000000	8.000000	48.000000	48.000000	48.000000
mean	0.546880	2.091111	228.821925	288.337919	7.661155
std	0.130362	0.236845	78.397049	86.891590	1.816903
min	0.333333	1.863333	98.333333	148.174410	4.585000
25%	0.476611	1.914167	167.000000	230.932715	6.253750
50%	0.586132	1.984762	261.714286	276.205930	7.301500
75%	0.625000	2.246688	284.375000	340.808041	9.069833
max	0.721000	2.474000	365.000000	501.123371	11.221250

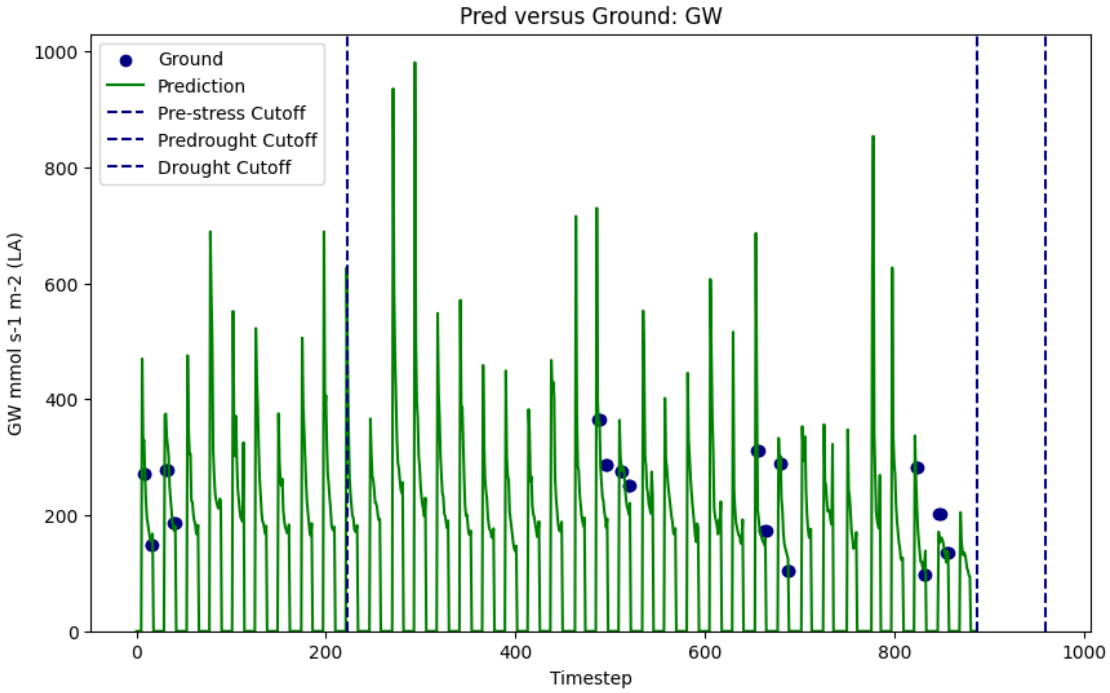
```
Pred
```

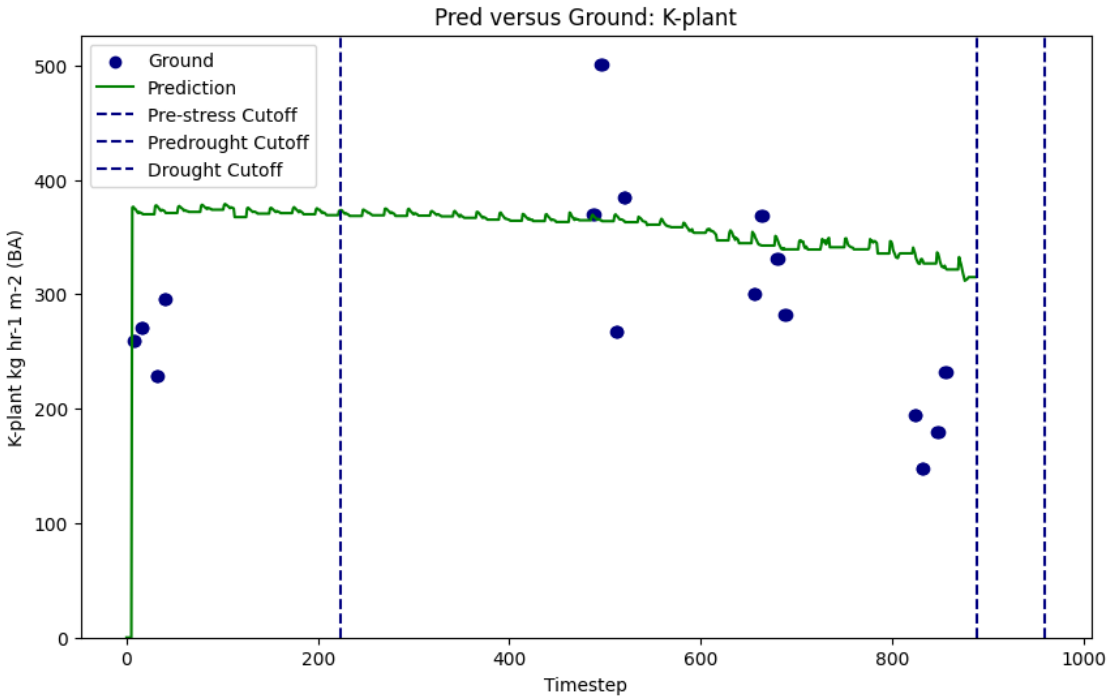
	P-PD	P-MD	GW	K-plant	E-MD

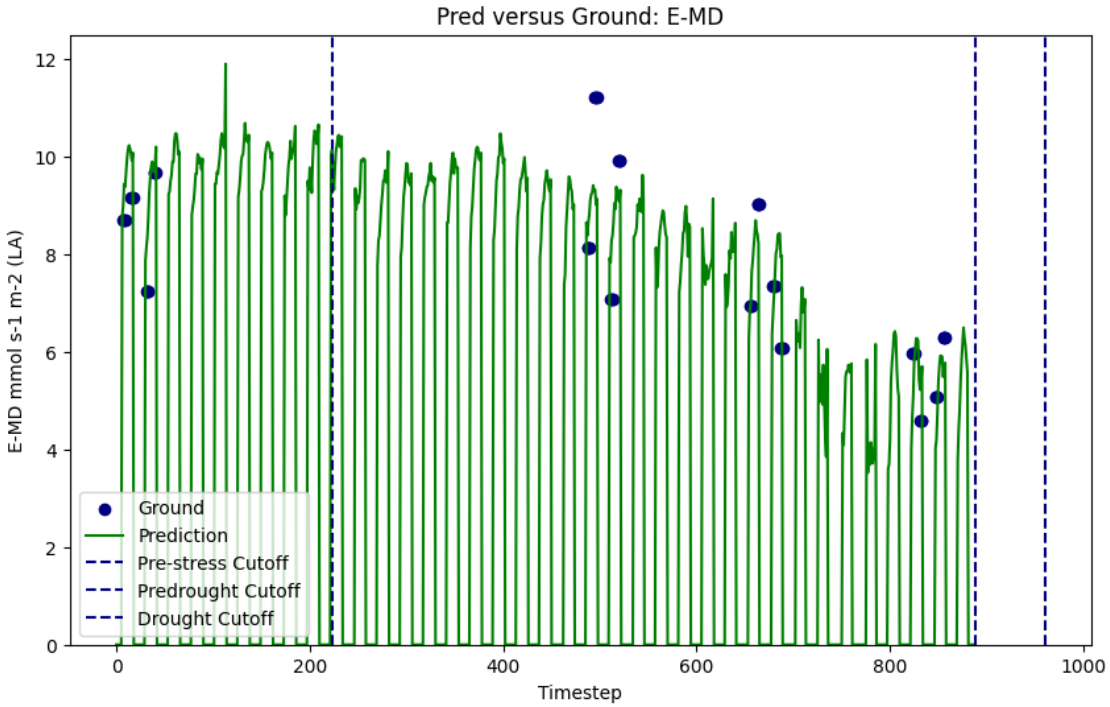
count	888.000000	888.000000	888.000000	888.000000	888.000000
mean	0.560664	1.162892	121.363999	356.180462	4.069054
std	0.059288	0.637809	153.457919	33.179891	4.396842
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.518500	0.534652	0.000000	345.457037	0.000000
50%	0.534634	0.686627	0.000000	365.026863	0.000000
75%	0.604579	1.851713	209.002486	369.945565	8.992475
max	0.705598	2.256529	980.774549	378.888343	11.908954

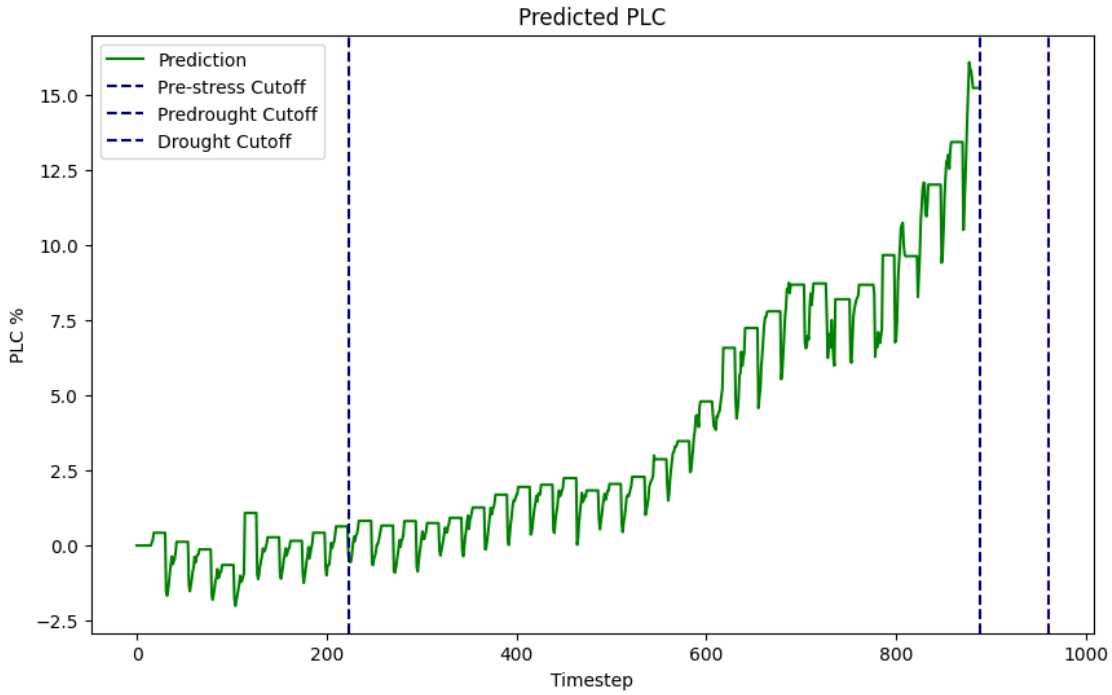


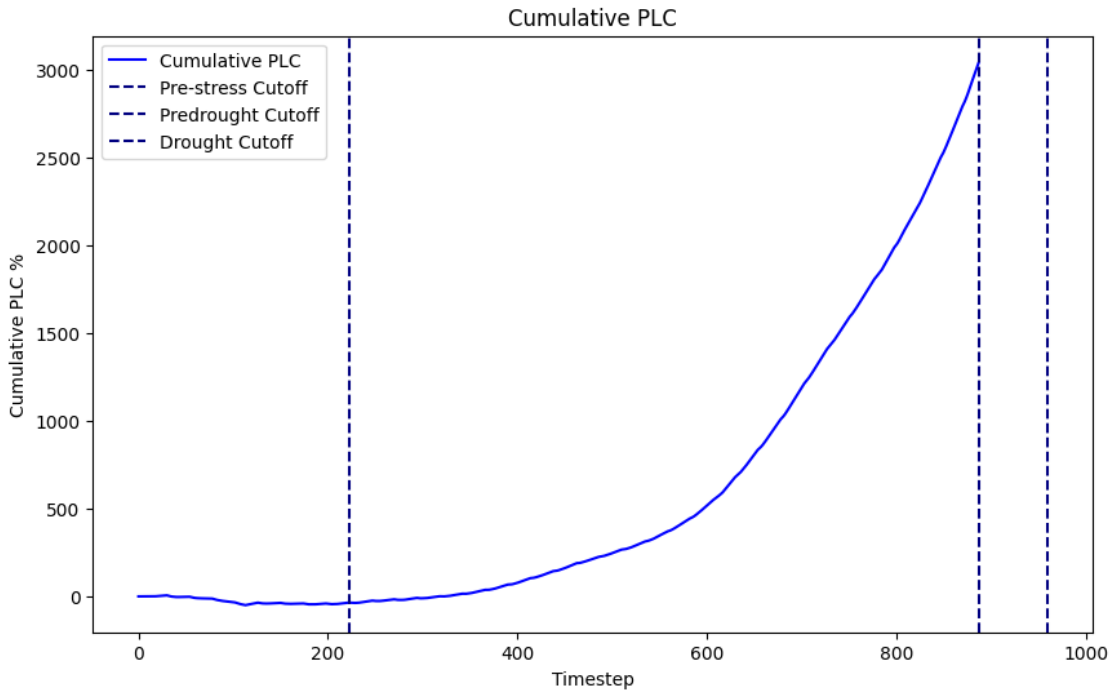

Pred versus Ground: P-MD
P-MD -MPa
Timestep
Ground
Prediction
Pre-stress Cutoff
Predrought Cutoff
Drought Cutoff
0.0
0.5
1.0
1.5
2.0
2.5
0
200
400
600
800
1000


Pred versus Ground: GW
GW mmol s-1 m-2 (LA)
Timestep
Ground
Prediction
Pre-stress Cutoff
Predrought Cutoff
Drought Cutoff
0
200
400
600
800
1000
0
200
400
600
800
1000

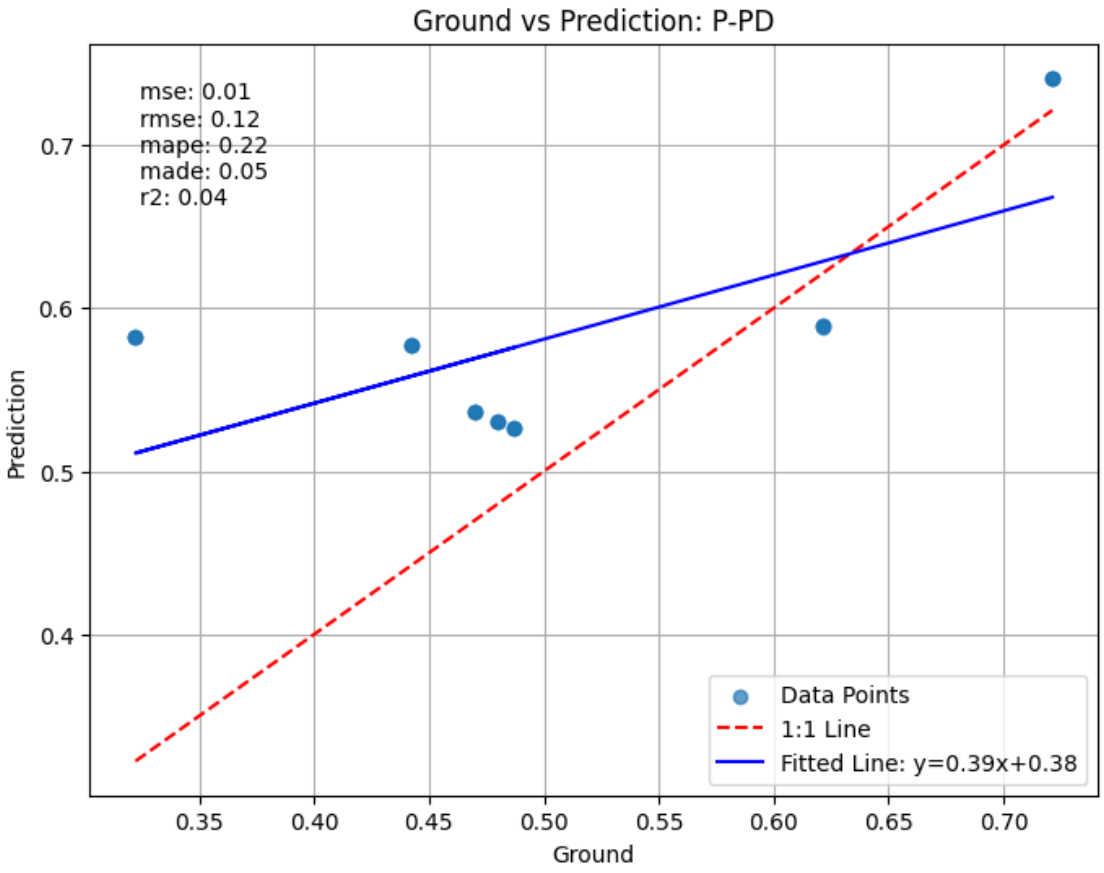

Pred versus Ground: K-plant
Ground
Prediction
Pre-stress Cutoff
Predrought Cutoff
Drought Cutoff
K-plant kg hr-1 m-2 (BA)
Timestep
0
100
200
300
400
500
0
200
400
600
800
1000

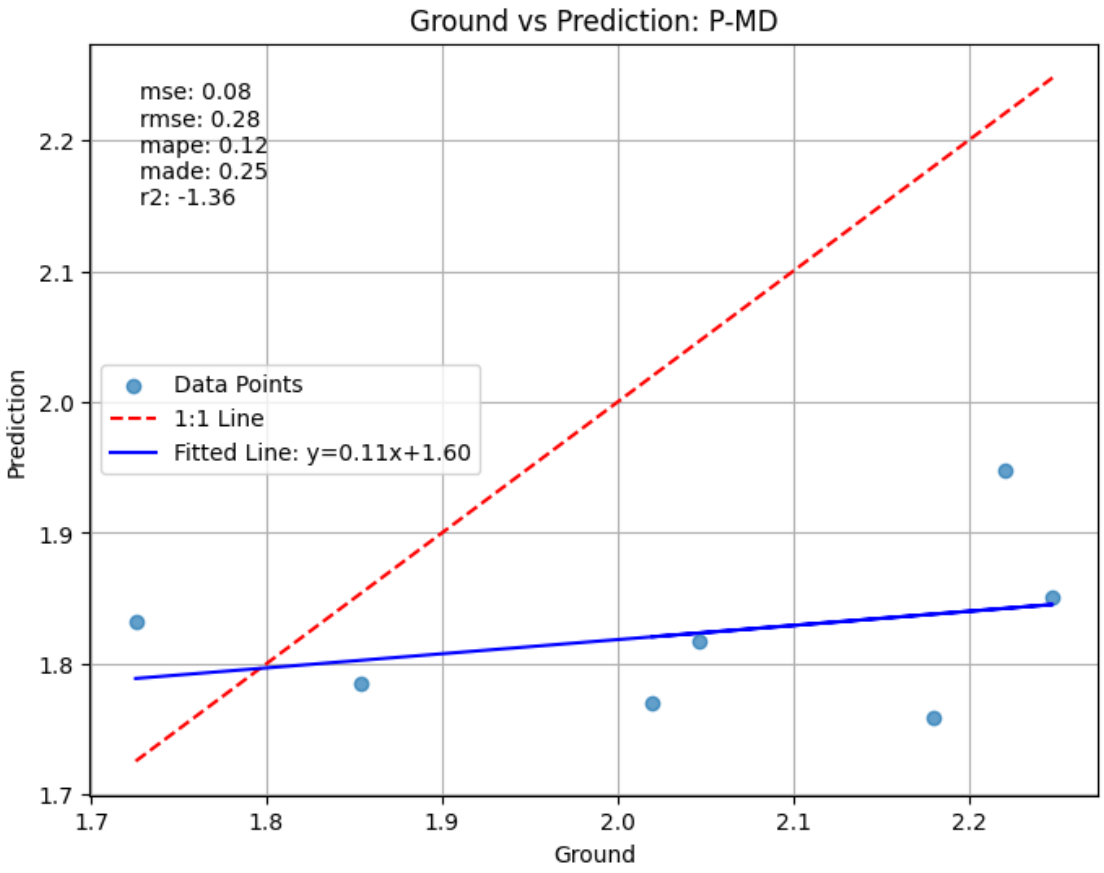

Pred versus Ground: E-MD
Ground
Prediction
Pre-stress Cutoff
Predrought Cutoff
Drought Cutoff
E-MD mmol s-1 m-2 (LA)
Timestep
0
2
4
6
8
10
12
0
200
400
600
800
1000

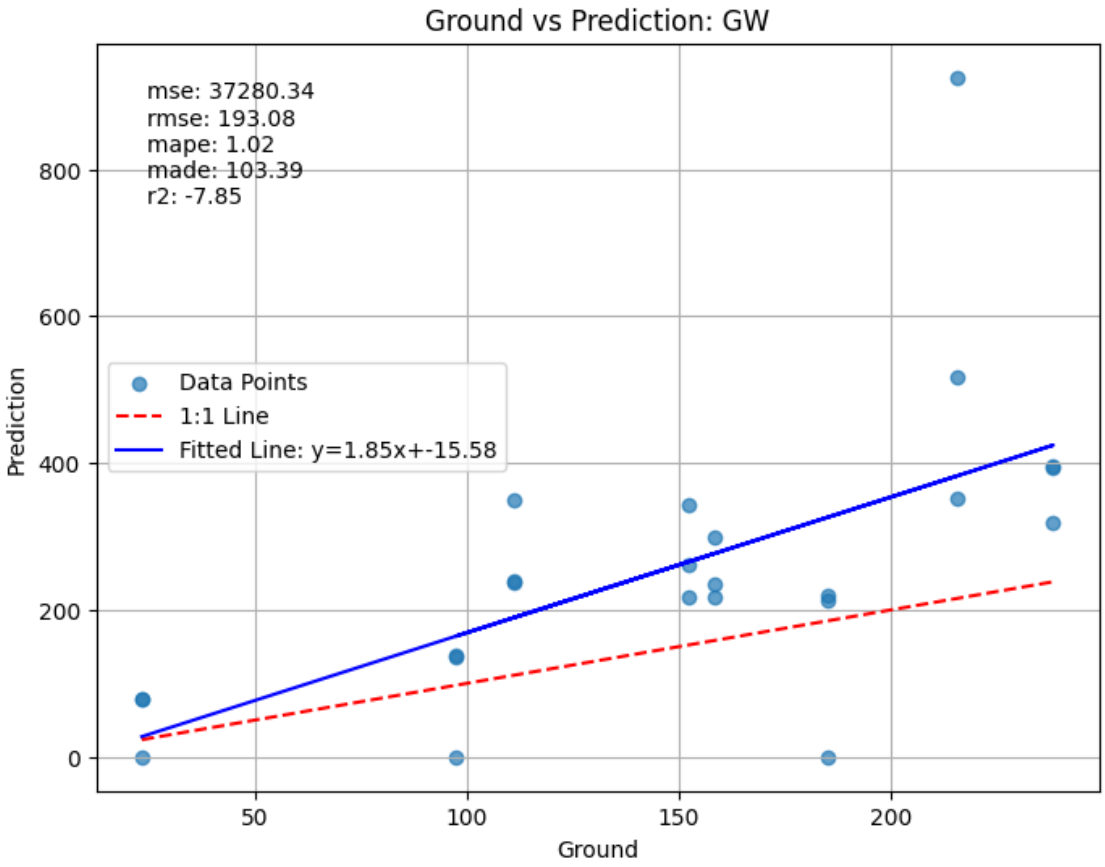


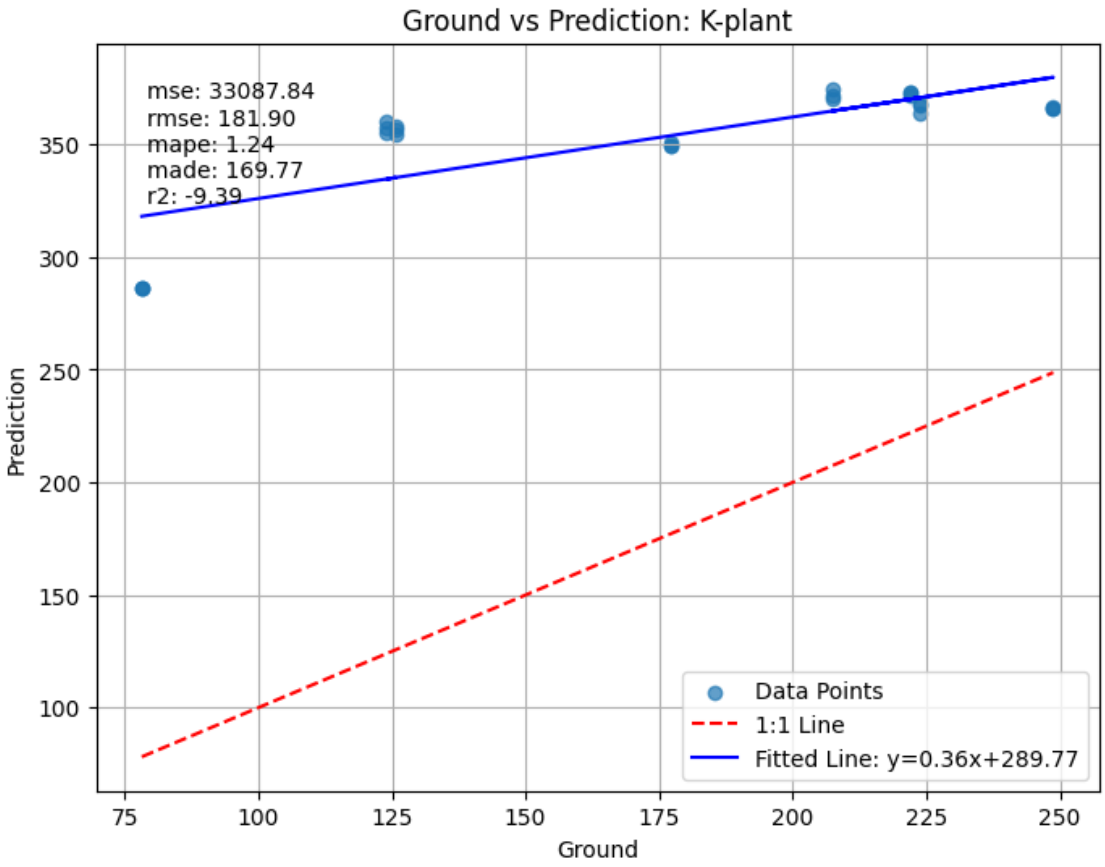


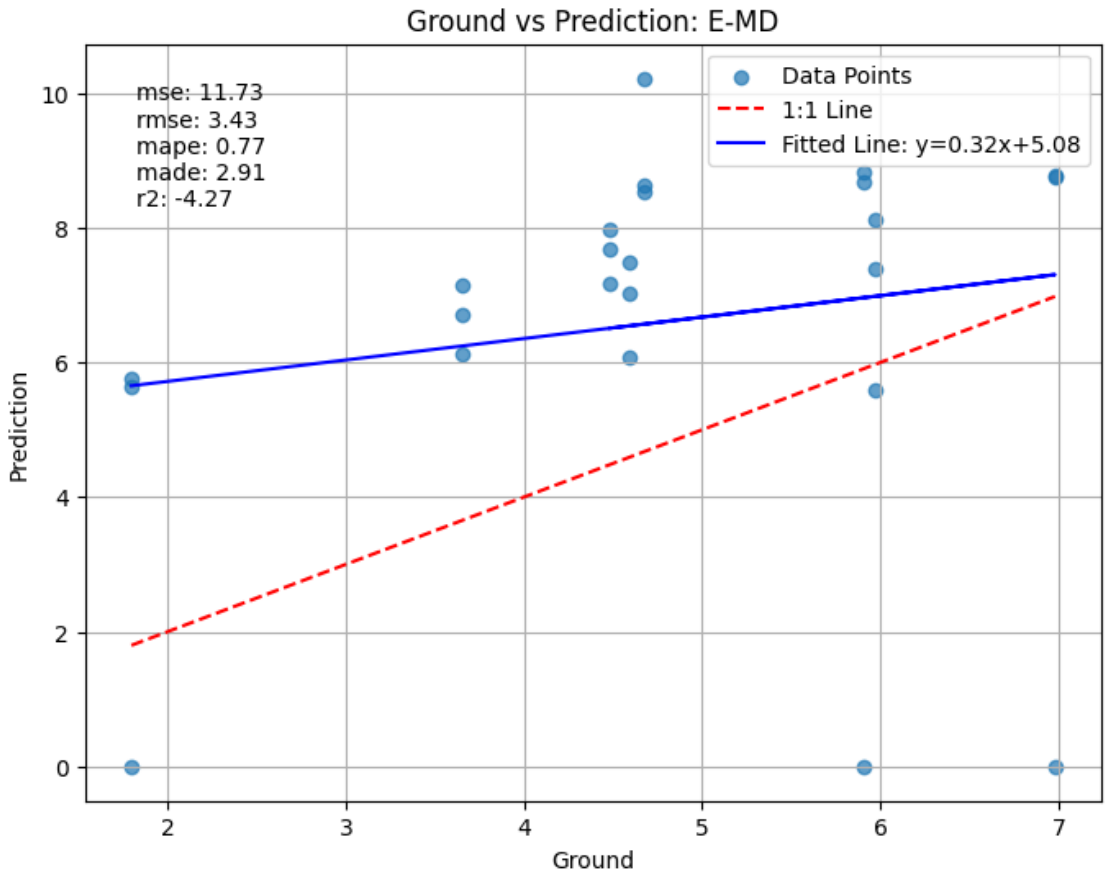
[428]:
```
res = cmp_pred_to_ground_metrics(columns, ground, pred, drought, len(pred))
```


Ground vs Prediction: P-PD
mse: 0.01
rmse: 0.12
mape: 0.22
made: 0.05
r2: 0.04
Prediction
Ground
0.7
0.6
0.5
0.4
0.35
0.40
0.45
0.50
0.55
0.60
0.65
0.70
Data Points
1:1 Line
Fitted Line: y=0.39x+0.38


Ground vs Prediction: P-MD
mse: 0.08
rmse: 0.28
mape: 0.12
made: 0.25
r2: -1.36
Data Points
1:1 Line
Fitted Line: y=0.11x+1.60
Prediction
Ground
1.7
1.8
1.9
2.0
2.1
2.2


Ground vs Prediction: GW
mse: 37280.34
rmse: 193.08
mape: 1.02
made: 103.39
r2: -7.85
Data Points
1:1 Line
Fitted Line: y=1.85x+-15.58
Prediction
Ground
0
200
400
600
800
50
100
150
200


Ground vs Prediction: K-plant
mse: 33087.84
rmse: 181.90
mape: 1.24
made: 169.77
r2: -9.39
Prediction
Ground
Data Points
1:1 Line
Fitted Line: y=0.36x+289.77
100
150
200
250
300
350
75
100
125
150
175
200
225
250



[429]:
```
res
```

[429]:
```
{'P-PD': {'mse': 0.013755136211917918,
  'rmse': 0.11728229283194423,
  'mape': 0.21778017572988526,
  'made': np.float64(0.05068852771400001),
  'r2': 0.03678178104802188},
 'P-MD': {'mse': 0.07723176197492178,
  'rmse': 0.27790603083582366,
  'mape': 0.11821911593401087,
  'made': np.float64(0.25013166101100004),
  'r2': -1.3577376212313657},
 'GW': {'mse': 37280.33577944383,
  'rmse': 193.08116370957532,
  'mape': 1.0214443528218944,
  'made': np.float64(103.38594292078955),
  'r2': -7.853561133708231},
 'K-plant': {'mse': 33087.84420649213,
  'rmse': 181.90064377701398,
```

```
  'mape': 1.2415495920126318,
  'made': np.float64(169.768816594944),
  'r2': -9.391832460390612},
 'E-MD': {'mse': 11.731249163031526,
  'rmse': 3.425091117478705,
  'mape': 0.7654901032450727,
  'made': np.float64(2.9089843726596114),
  'r2': -4.27049742276706}}
```

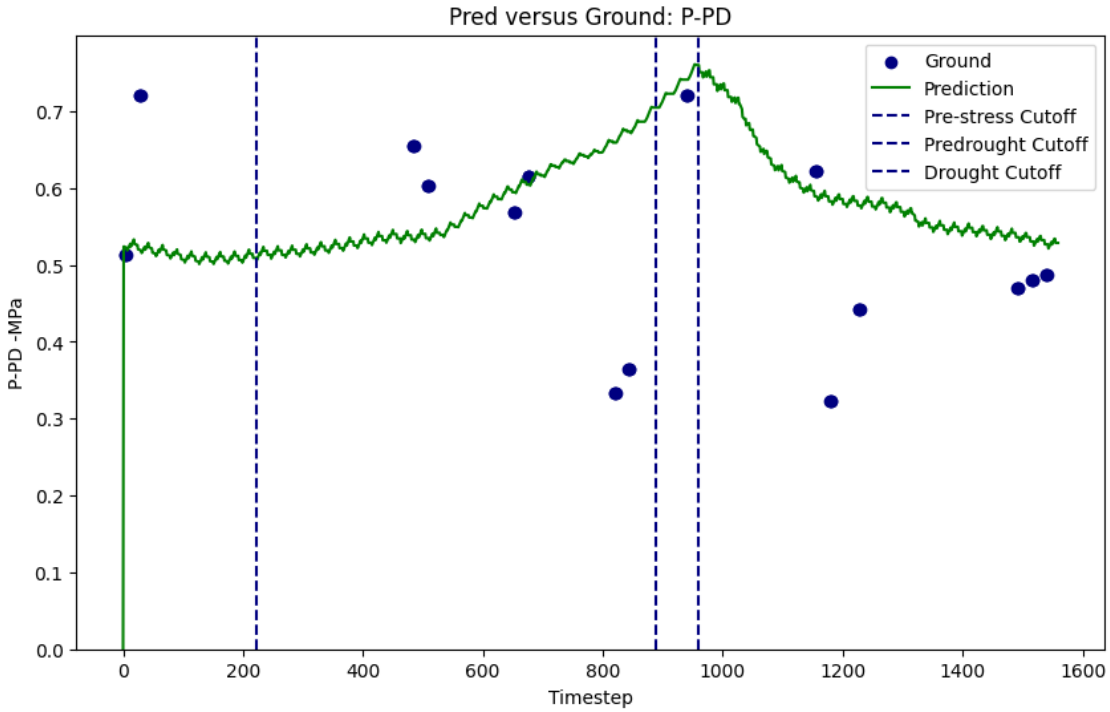
[430]:
```
plot_ground_pred(len(pred))
```

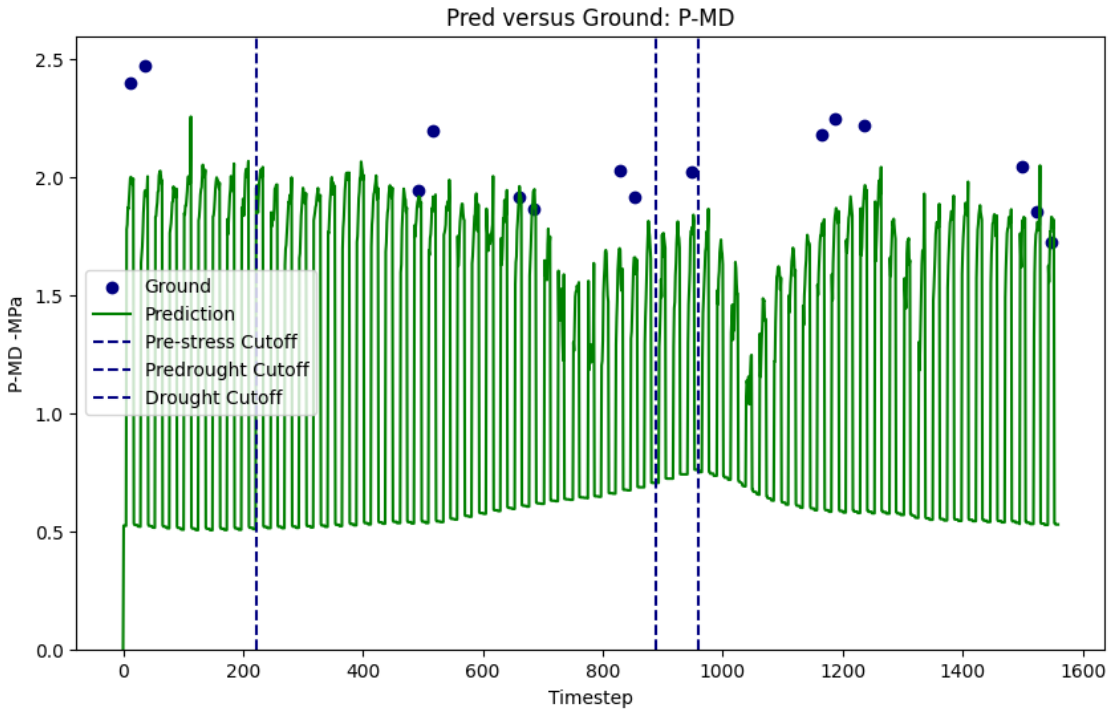
Ground

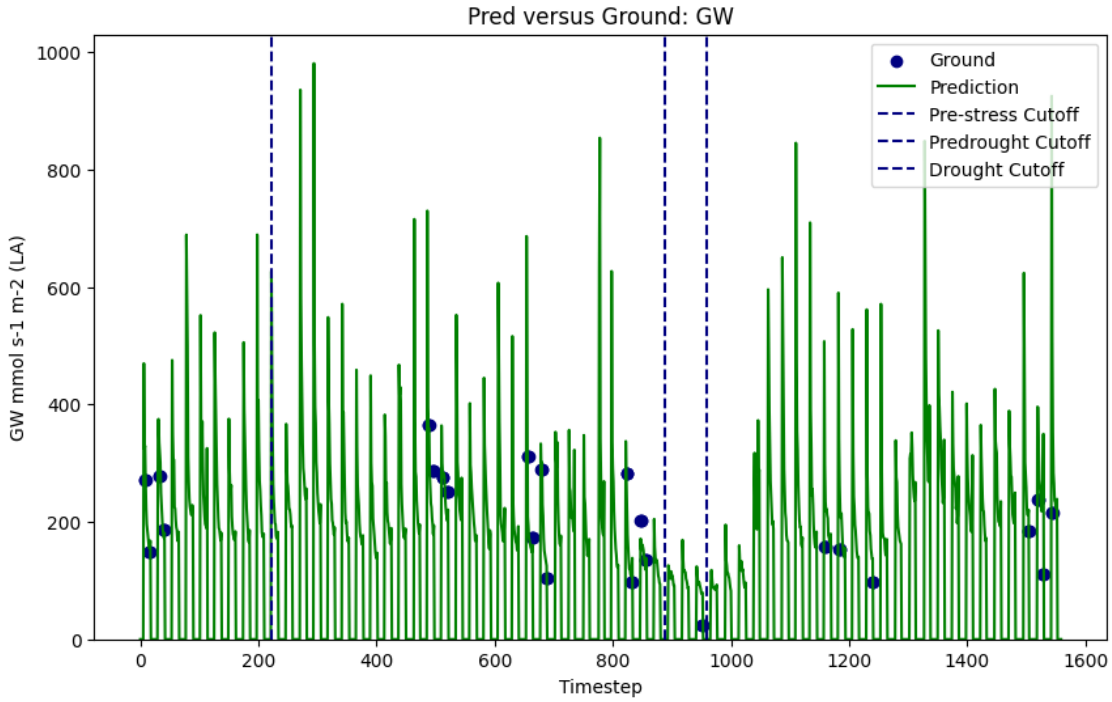
	P-PD	P-MD	GW	K-plant	E-MD
count	45.000000	15.000000	72.000000	72.000000	72.000000
mean	0.527929	2.068237	201.765542	250.851590	6.693108
std	0.126960	0.212290	83.526163	94.468560	2.199609
min	0.322222	1.725556	23.333333	78.198562	1.797333
25%	0.442000	1.913889	144.611111	190.840086	4.980278
50%	0.514000	2.026667	194.611111	240.266048	6.630000
75%	0.621667	2.208458	277.000000	297.252491	8.280750
max	0.721333	2.474000	365.000000	501.123371	11.221250

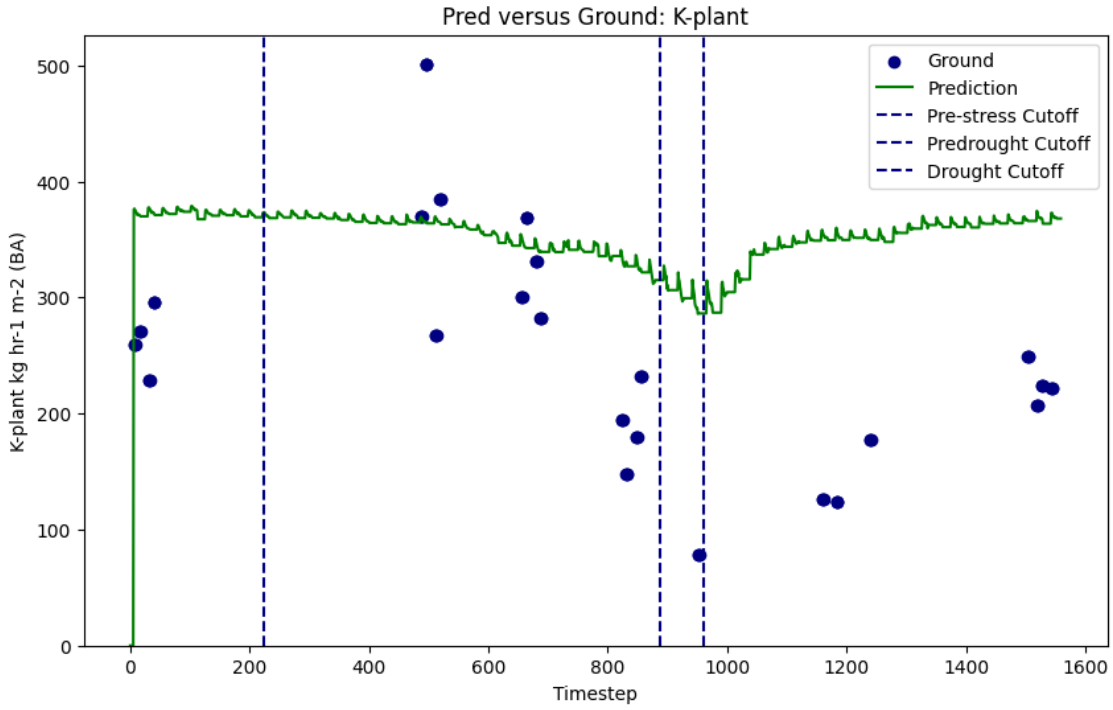
Pred

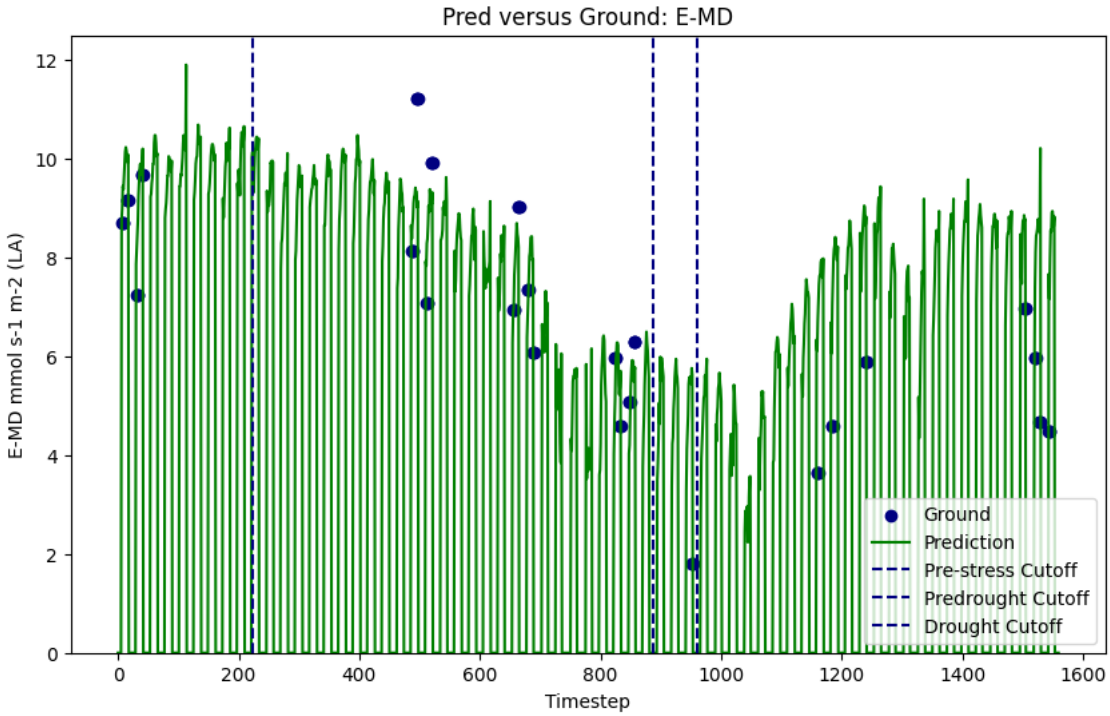
	P-PD	P-MD	GW	K-plant	E-MD
count	1560.000000	1560.000000	1560.000000	1560.000000	1560.000000
mean	0.581350	1.131199	115.215173	351.113367	3.663078
std	0.069968	0.600583	151.158052	30.163790	4.106808
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.527800	0.551694	0.000000	342.515674	0.000000
50%	0.554411	0.727496	0.000000	360.583868	0.000000
75%	0.623209	1.777177	206.812284	367.989418	8.169720
max	0.761098	2.256529	980.774549	378.888343	11.908954

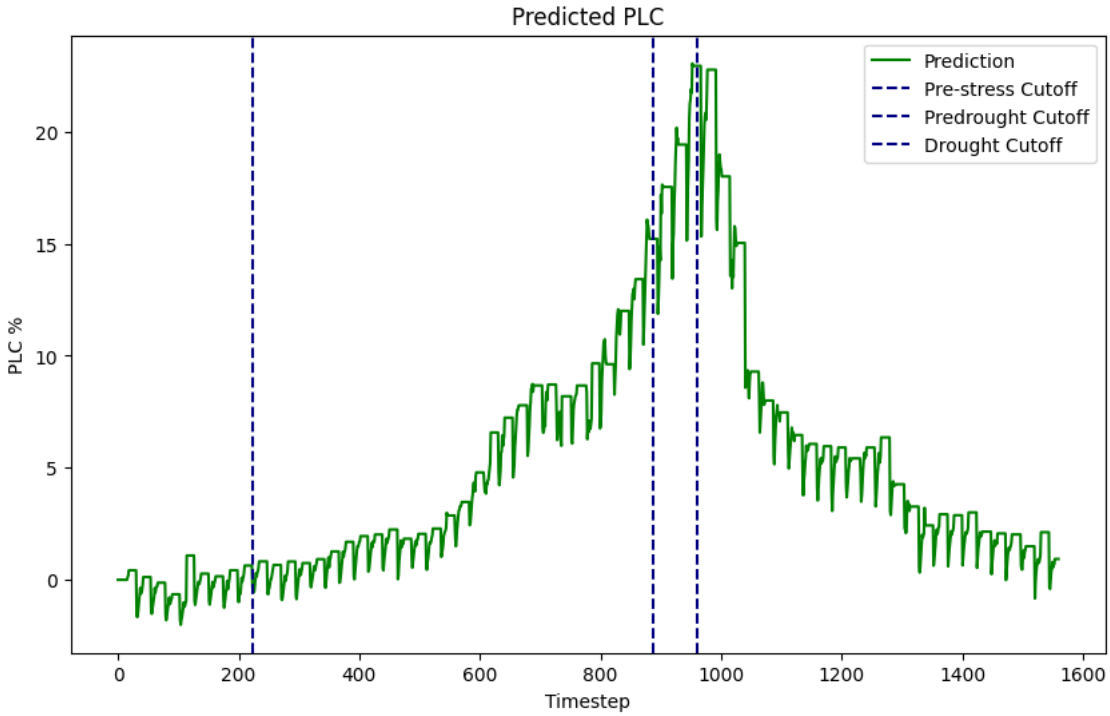

Pred versus Ground: P-PD
Ground
Prediction
Pre-stress Cutoff
Predrought Cutoff
Drought Cutoff
P-PD -MPa
Timestep
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
200
400
600
800
1000
1200
1400
1600

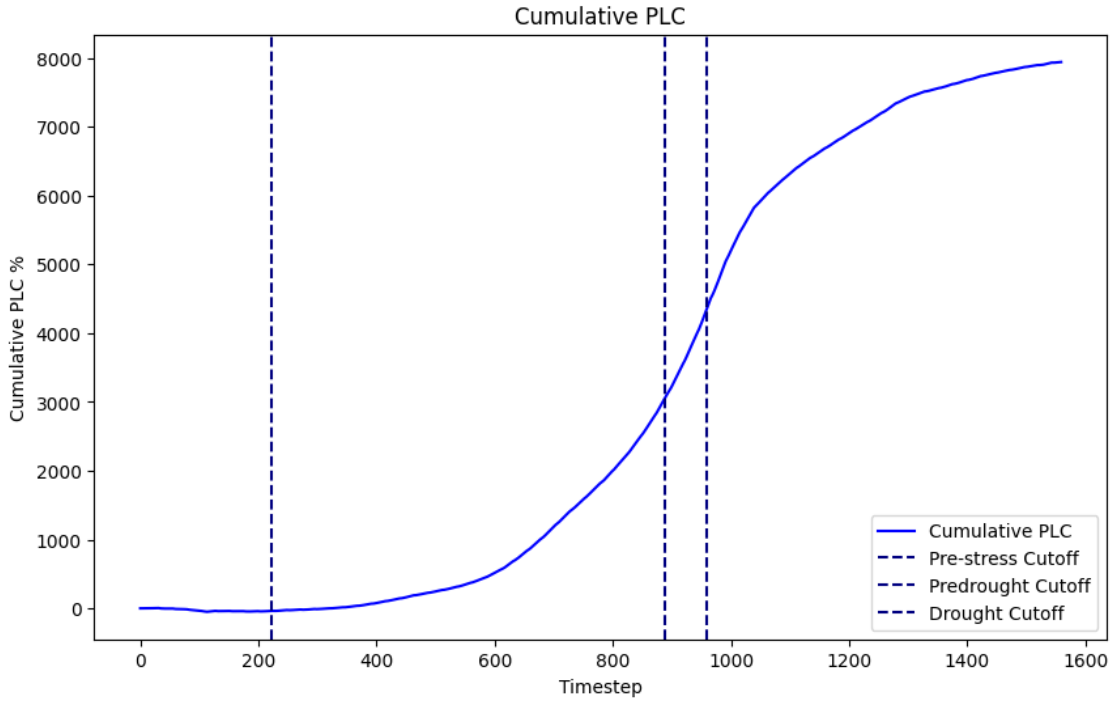

Pred versus Ground: P-MD
Ground
Prediction
Pre-stress Cutoff
Predrought Cutoff
Drought Cutoff
P-MD -MPa
Timestep
0.0
0.5
1.0
1.5
2.0
2.5
0
200
400
600
800
1000
1200
1400
1600


Pred versus Ground: GW
Ground
Prediction
Pre-stress Cutoff
Predrought Cutoff
Drought Cutoff
GW mmol s-1 m-2 (LA)
Timestep
0
200
400
600
800
1000
1200
1400
1600


Pred versus Ground: K-plant
Ground
Prediction
Pre-stress Cutoff
Predrought Cutoff
Drought Cutoff
K-plant kg hr-1 m-2 (BA)
Timestep
0
100
200
300
400
500
600
800
1000
1200
1400
1600


Pred versus Ground: E-MD
E-MD mmol s-1 m-2 (LA)
Timestep
Ground
Prediction
Pre-stress Cutoff
Predrought Cutoff
Drought Cutoff

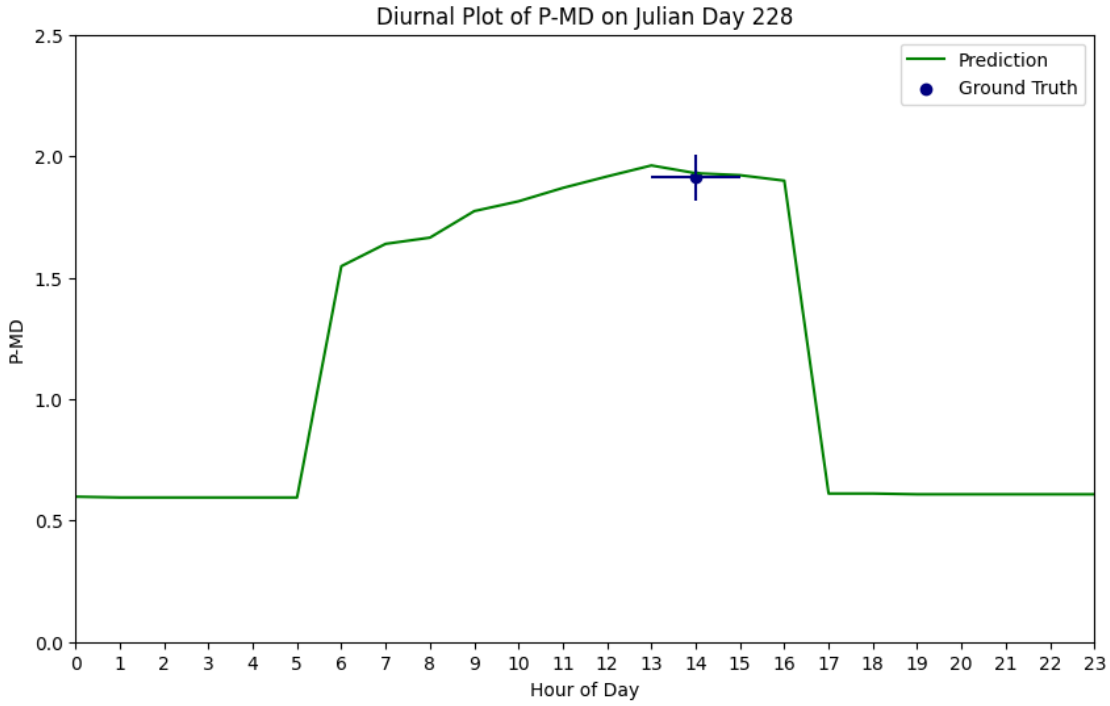

Predicted PLC
PLC %
Timestep
Prediction
Pre-stress Cutoff
Predrought Cutoff
Drought Cutoff



[431]:
```
var = 'P-MD'
day = 228
hour = 12
xerr = 1
```

[432]:
```
plt.figure(figsize=(10, 6))
plt.plot(range(0, 24), pred.loc[pred['julian-day'] == day][var],␣
 ↪label='Prediction', color='green')
plt.scatter(14, ground.loc[(ground['julian-day'] == day) &␣
 ↪(ground['standard-time'] == hour)][var], label='Ground Truth', color='navy')

m_stderr = stderr.loc[(stderr['julian-day'] == day) & (stderr['standard-time']␣
 ↪== hour)][var].item()
m_ground = ground.loc[(ground['julian-day'] == day) & (ground['standard-time']␣
 ↪== hour)][var].item()
plt.errorbar(14, m_ground, yerr=1.96*m_stderr, xerr=xerr, color='navy')

plt.xticks(range(0, 24))
plt.xlim(0, 23)  # Ensure no gap on the left side of the origin
plt.ylim(0, 2.5)
plt.xlabel('Hour of Day')
plt.ylabel(var)
plt.title(f'Diurnal Plot of {var} on Julian Day {day}')
plt.legend()
```

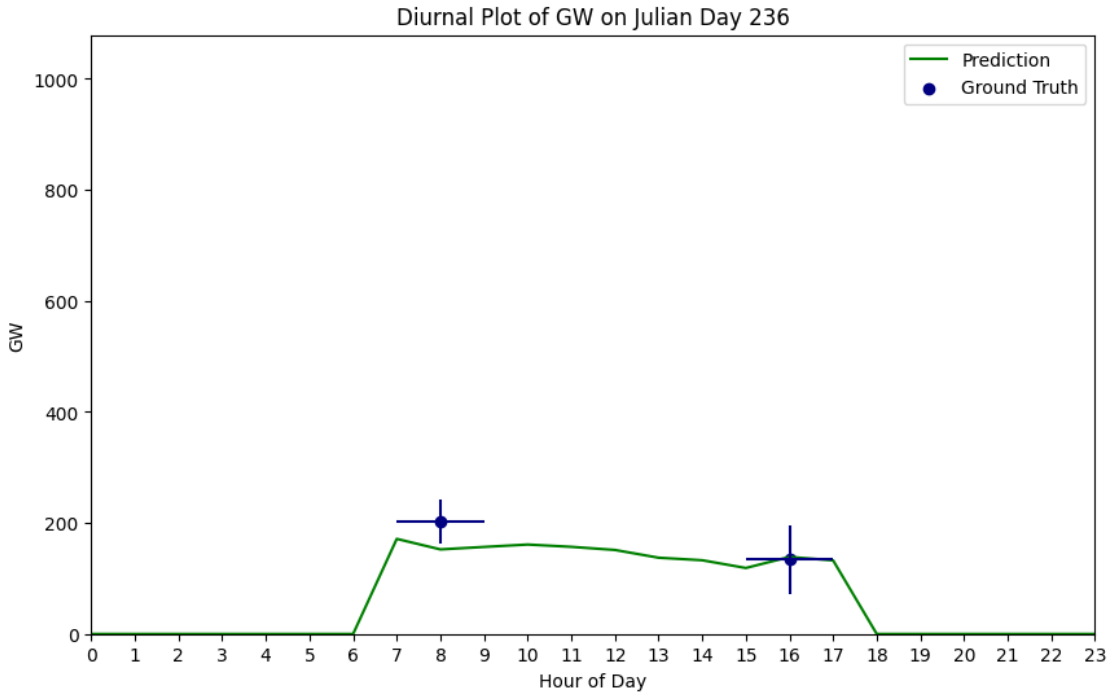
```
plt.grid(False)
plt.show()
```



[433]:
```
var = 'GW'
# var = 'P-MD'
# var = 'P-PD'
day = 236
hours = (8,16)
# hours = (12,)
# hours = (4,)
xerr = 1 # can set depending on time collection ranges, usually we are taking a␣
 ↪three hour collection range
```

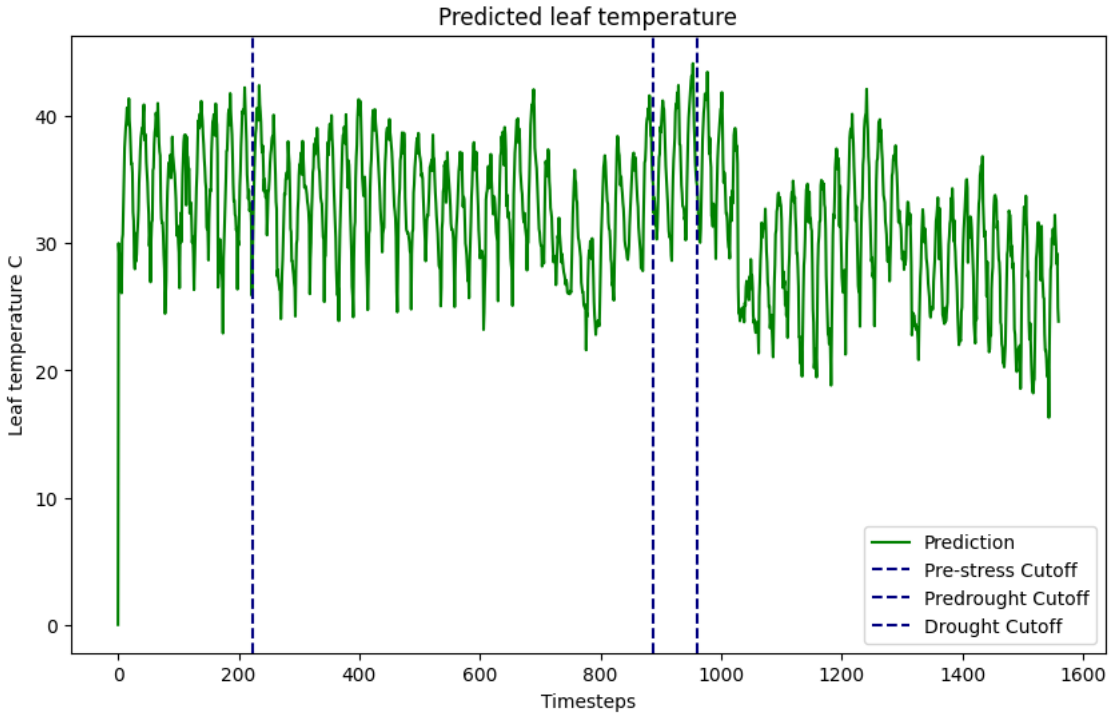
[434]:
```
plt.figure(figsize=(10, 6))
plt.plot(range(0, 24), pred.loc[pred['julian-day'] == day][var],␣
 ↪label='Prediction', color='green')
plt.scatter(hours, ground.loc[(ground['julian-day'] == day) &␣
 ↪(ground['standard-time'].isin(hours))][var], label='Ground Truth',␣
 ↪color='navy')

for hour in hours:
    m_stderr = stderr.loc[(stderr['julian-day'] == day) &␣
 ↪(stderr['standard-time'] == hour)][var].item()
```

```
    m_ground = ground.loc[(ground['julian-day'] == day) &␣
 ↪(ground['standard-time'] == hour)][var].item()
    plt.errorbar(hour, m_ground, yerr=1.96*m_stderr, xerr=xerr, color='navy')

plt.xticks(range(0, 24))
plt.xlim(0, 23)  # Ensure no gap on the left side of the origin
plt.ylim(0, np.max(pred[var]) + 0.1 * np.max(pred[var]))
plt.xlabel('Hour of Day')
plt.ylabel(var)
plt.title(f'Diurnal Plot of {var} on Julian Day {day}')
plt.legend()
plt.grid(False)
plt.show()
```

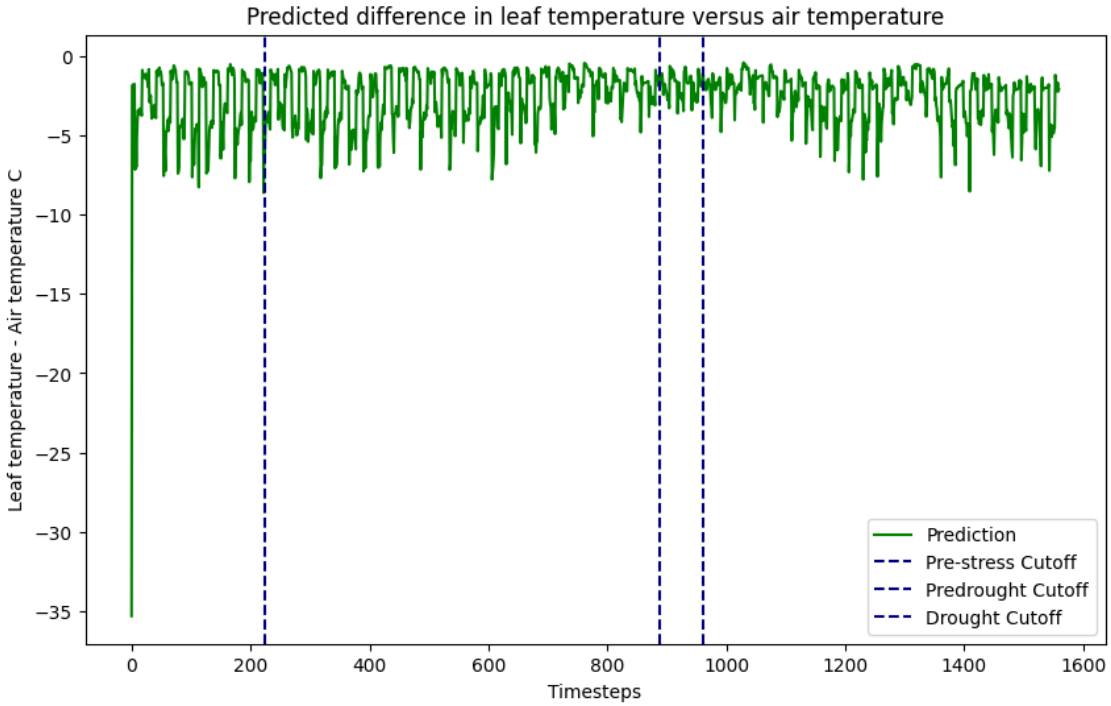


[435]:
```
plt.figure(figsize=(10,6))
plt.plot(range(0, len(pred)), pred['leaftemp'], color="green", alpha=1,␣
 ↪label=f'Prediction')
plt.axvline(x=stress_begin, color='navy', linestyle='--', label='Pre-stress␣
 ↪Cutoff')
plt.axvline(x=predrought_timestep, color='navy', linestyle='--',␣
 ↪label='Predrought Cutoff')
plt.axvline(x=post_timestep, color='navy', linestyle='--', label='Drought␣
 ↪Cutoff')
plt.xlabel("Timesteps")
```

```
plt.ylabel("Leaf temperature C")
plt.title("Predicted leaf temperature")
plt.legend()
plt.show()
```



[436]:
```
plt.figure(figsize=(10,6))
plt.plot(range(0, len(pred)), pred['leaftemp'] - pred['T-air'], color="green",␣
 ↪alpha=1, label=f'Prediction')
plt.axvline(x=stress_begin, color='navy', linestyle='--', label='Pre-stress␣
 ↪Cutoff')
plt.axvline(x=predrought_timestep, color='navy', linestyle='--',␣
 ↪label='Predrought Cutoff')
plt.axvline(x=post_timestep, color='navy', linestyle='--', label='Drought␣
 ↪Cutoff')
plt.xlabel("Timesteps")
plt.ylabel("Leaf temperature - Air temperature C")
plt.title("Predicted difference in leaf temperature versus air temperature")
plt.legend()
plt.show()
```


Predicted difference in leaf temperature versus air temperature
Leaf temperature - Air temperature C
Timesteps
0
−5
−10
−15
−20
−25
−30
−35
0
200
400
600
800
1000
1200
1400
1600
Prediction
Pre-stress Cutoff
Predrought Cutoff
Drought Cutoff