STØTTET AF
Promilleafgiftsfonden for landbrug

Maize field-level predictions using satellite images

This notebook implements the experiment pipeline for predicting the field-level

1. harvested dry matter yield in tons per hectare and
2. dry mater percentage of the yield

on maize fields using satellite images.

The model scores and artifacts of the individual experiment are stored on the MLflow server maintained by the SEGES data science team, but all experiments are described and concluded in the bottom of this notebook.

For more project details see the JIRA feature MPSI-15 (https://jira.seges.dk/browse/MPSI-15) and the "Maize Prognosis from Satellite Images" Confluence space (https://confluence.seges.dk/display/MPSI/Maize+Prognosis+from+Satellite+Images).

Important notebook execution details:

- The notebook uses multiple utility scripts and to ensure the correct versions of these scripts the notebook should be executed from the git tag: "MPSI-15_report (https://bitbucket.seges.dk/projects/DDS/repos/maize_prognosis_from_satellite_images/browse?at=refs%2Ftags%2FMPSI-15_report)".
- The notebook have been executed with the "py37_v8" Conda environment. You can find the environment YAML file on this Confluence page (https://confluence.seges.dk/x/E5QQBw).
- To reproduce the notebook from an empty data folder on Ahsoka, change the cell under the section Collect satellite, DTM, and DMI data from "Raw" to "Code". Without SatHub cache the cell takes approx. 3,5 hour to complete and with cache it takes approx. 20 min.

Table of Contents

In [1]:
```
from IPython.display import HTML
HTML('''<script>code_show_err=true;function code_toggle_err(){if(code_show_err){$('div.output_stderr').hide();}else{$('div.output_stderr').show();}code_show_err=!code_show_err}$( document ).ready(code_toggle_err);</script>To toggle on/off output_stderr, click <a href="javascript:code_toggle_err()">here</a>.''')
```

Out[1]: To toggle on/off output_stderr, click here.

In [2]:

```
%load_ext watermark
print('Watermark of notebook execution:')
%watermark
%matplotlib inline

from pathlib import Path
import shutil

import dask
import dask.distributed
from gaffa.dask_task.remote import mkdir
from gaffa.local_task.environment import check_conda_env, get_conda_env_name
import git
import geopandas as gpd
import geoviews as gv
from IPython.display import display
import pandas as pd
from sklearn.ensemble import GradientBoostingRegressor

from MPSI_15_DMDB_data_utils import (
    add_DMDB_polygons, check_gdf_equals, collect_maize_field_level_yield_df,
    viz_gdf_polygons)
from MPSI_15_raster_data_utils import run_EO_dataset_pipeline
from MPSI_15_feature_utils import run_featurization_pipeline
from MPSI_15_ML_utils import (
    run_split_data_pipeline, run_training_pipeline, run_prediction_pipeline,
    present_model_scores)
from MPSI_15_utils import (
    collect_MLflow_runs, save_MLflow_experiment, query_yes_no,
    run_number_of_images, show_score_table, viz_model_scores)

# NOTE: hvplot has to be imported after pandas have been imported.
import hvplot.pandas
pd.set_option('display.max_columns', 100)

# Create dask client to Ahsoka
client = dask.distributed.Client('localhost:8786')
client.upload_file('./MPSI_15_DMDB_data_utils.py')
client.upload_file('./MPSI_15_raster_data_utils.py')
client.upload_file('./MPSI_15_feature_utils.py')
client.upload_file('./MPSI_15_ML_utils.py')
client.upload_file('./MPSI_15_utils.py')
display(client)

# Check Conda environment file and running locally and on Dask scheduler
conda_env_name = get_conda_env_name(Path('./conda_environment.yml'))
print(check_conda_env(conda_env_name))
print(dask.delayed(check_conda_env)(conda_env_name).compute())

# Global variables
DATA_PATH = Path('/scratch/MPSI-15_maize_field_level_yield_prediction/')
mkdir(DATA_PATH).compute()
print(f'Created MPSI folder "{DATA_PATH.absolute()}" on Ahsoka.')
WIDGET_WIDTH = 900  # Restricted to 900 px as it must fit in the HTML export.
HV_HEIGHT = 600
HV_XROTATION = 60

DATA_CONFIG = {
    'yield_yearly_path': Path('./tsf20191111_udbyttedata_2017_2018_2019.xlsx'),
    'yield_2019_path': Path('./tsf20191120_udbyttedata_2019.xlsx'),
    'valid_harvest_date_interval': {
        'start_month':7, 'start_day': 20,
        'end_month':11, 'end_day': 15},
    'sample_indices_names': ['FieldCropId', 'harvestYear']
}
EO_CONFIG = {
    'cache_path': Path('/scratch/MPSI-15_sentinelhub_cache/'),
    'resolution': 10,
    'season_start': '02-01',  # Note: Date in MM-DD format
    'season_end': '12-01',  # Note: Date in MM-DD format
    'satellite': 'S2',
    'product': 'L1C',
    'output': [
        # Actual channels analyses
        'B01', 'B02', 'B03', 'B04', 'B05', 'B06', 'B07', 'B08', 'B8A', 'B09',
        'B10', 'B11', 'B12', 'NDRE',
        # Note: build_data_dependencies() in SatHub v1.4 does not support
        # multiple cloud masks, thus we manually specify the bands for the
        # utilized cloud masks.
        # Note: Bands used for MSScvv
        'B03', 'B04',
        # Note: Bands used for S2cloudless
        'B01', 'B02', 'B04', 'B05', 'B08', 'B8A', 'B09', 'B10', 'B11', 'B12'],
    'cloud_masks': ['S2cloudless', 'MSScvm'],
    'S2cloudless_threshold': 0.4,
    'DK_DTM_tile_path': Path(
        '/projects/BDICG/3 Hoejdedata DK/DDS_unpacked/tiles/'),
    'data_params': ['airtemp', 'glorad', 'maxtemp', 'mintemp', 'prec', 'evapo',
                    'soiltemp'],
    'data_sources': ['obs'],
    'data_path': DATA_PATH/'EO_datasets'

}
```

```
FEATURE_CONFIG = {
    'time_column_str_format': '%m-%d - %B %-d',
}
FEATURE_HYPERPARAM = {
    'max_cloud_proba': 0.7,
    'time_span': ('04-01', '09-15'),  # Note: Date in MM-DD format
    'feature_time_delta': '7D',
    'S2_time_series_features': [
        'S2_L1C_B01', 'S2_L1C_B02', 'S2_L1C_B03', 'S2_L1C_B04',
        'S2_L1C_B05', 'S2_L1C_B06', 'S2_L1C_B07', 'S2_L1C_B08', 'S2_L1C_B8A',
        'S2_L1C_B09', 'S2_L1C_B10', 'S2_L1C_B11', 'S2_L1C_B12', 'S2_L1C_NDRE'],
    'DTM_features': [
        'field_relative_mean', 'slope_pct', 'slope_angle', 'slope_aspect'],
    'position_and_precrop_features': ['geox', 'geoy', 'regionId'],
    'DMI_agg_features': {
        'DMI_air_temperature': ['mean', 'std', 'min', 'max'],
        'DMI_evaporation': ['mean', 'std', 'min', 'max', 'sum'],
        'DMI_global_radiation': ['mean', 'std', 'min', 'max', 'sum'],
        'DMI_maximum_temperature': ['mean', 'std', 'min', 'max'],
        'DMI_minimum_temperature': ['mean', 'std', 'min', 'max'],
        'DMI_corrected_precipitation': ['mean', 'std', 'min', 'max', 'sum'],
        'DMI_soil_temperature': ['mean', 'std', 'min', 'max']},
    'DMI_prognosis_days_delta': 0,
    'target_name': 'drymatterPercent',
    'select_harvest_year': None  # None if select all years available
}
ML_CONFIG = {
    'random_state': 42,
    'validation_size': 0.2,
    'validation_year': None,
    'model_path': DATA_PATH/'regressor_maize_yield.joblib',
    'train_predictions_path': DATA_PATH/'training_predictions.parquet.brotli',
    'val_predictions_path': DATA_PATH/'validation_predictions.parquet.brotli'
    }
ML_HYPERPARAM = {
    'sklearn_regressor': GradientBoostingRegressor
    }
# The keyword arguments passed to the model.
# NOTE: use the elements from the CONFIG and HYPERPARAM dictionaries above.
MODEL_KEY_ARGS = {
    'random_state': ML_CONFIG['random_state']
}
# Pipeline configurations and hyperparameters save as MLflow artifacts
CONFIGURATIONS = {
    'DATA_CONFIG': DATA_CONFIG,
    'EO_CONFIG': EO_CONFIG,
    'FEATURE_CONFIG': FEATURE_CONFIG,
    'ML_CONFIG': ML_CONFIG
}
HYPERPARAMETERS = {
    'FEATURE_HYPERPARAM': FEATURE_HYPERPARAM,
    'ML_HYPERPARAM': ML_HYPERPARAM,
    'MODEL_KEY_ARGS': MODEL_KEY_ARGS
}
```

```
Watermark of notebook execution:
2019-12-09T11:37:01+01:00

CPython 3.7.4
IPython 7.8.0

compiler   : GCC 7.3.0
system     : Linux
release    : 5.0.0-37-generic
machine    : x86_64
processor  : x86_64
CPU cores  : 4
interpreter: 64bit
```

Client
Scheduler: tcp://localhost:8786
Dashboard: http://localhost:8787/status (http://localhost:8787/status)

Cluster
Workers: 20
Cores: 20
Memory: 8.01 TB

```
Conda environment check passed on "ubuntuVM". Environment in use is: "py37_v8" from: "Jupyter notebook"
Conda environment check passed on "ahsoka". Environment in use is: "py37_v8" from: "Python script"
Created MPSI folder "/scratch/MPSI-15_maize_field_level_yield_prediction" on Ahsoka.
```

Collect maize field-level yield data

See notebook "MPSI-14_understand_data.ipynb" for more in depth understanding and exploration of the maize field-level yield data.

In [3]:
```
df_detail_maize_yields = collect_maize_field_level_yield_df(DATA_CONFIG)
print(f'The dataset contains data from {df_detail_maize_yields.shape[0]} fields')
```

```
Missing fields (i.e. FieldCropId) from former 2019 sheet in the new 2019 sheet:
[23197647 23197647]

Original maize yield DataFrame contains "3114" rows.
```

	FieldCropId	FarmId	harvestYear	successionNo	sproutingDate	stdCropNormNumber	stdCropName	preCropId	preCropName	theDate	area	registered
1290	18864494	29606	2018	1	NaT	6204	Majshelsæd	5909.0	Vårbyg	2018-09-12	7.83	1
1291	18908796	63955	2018	1	NaT	6204	Majs, helsæd	6204.0	Majs, helsæd	2018-08-28	0.62	1
1292	18908796	63955	2018	1	NaT	6204	Majs, helsæd	6204.0	Majs, helsæd	2018-08-28	0.64	1
1293	18908796	63955	2018	1	NaT	6204	Majs, helsæd	6204.0	Majs, helsæd	2018-08-28	0.65	1
1294	18908796	63955	2018	1	NaT	6204	Majs, helsæd	6204.0	Majs, helsæd	2018-08-28	7.24	1

```
The dataset contains data from 3114 fields
```

Clean maize yield DataFrame

Remove rows with all column values duplicated

In [4]:
```
df_detail_maize_yields = df_detail_maize_yields.drop_duplicates(keep=False)

assert (len(df_detail_maize_yields[
    df_detail_maize_yields.duplicated(keep=False)]) == 0), (
        'Dataframe still contains duplicated rows!')
print(f'DataFrame now contains "{len(df_detail_maize_yields)}" rows.')
```

```
DataFrame now contains "3112" rows.
```

Remove all rows with duplicated FieldCropId

We assume the rows with duplicated FieldCropId have been harvested in two individual task, thus their area such sum up to near the total area of the field polygon.

However, as a simple start on our work, we have decided to remove all rows with duplicated FieldCropId.

In [5]:
```
df_detail_maize_yields = df_detail_maize_yields.drop_duplicates(
    subset='FieldCropId', keep=False)

assert len(df_detail_maize_yields[df_detail_maize_yields.duplicated(
    subset='FieldCropId', keep=False)]) == 0, (
        'Dataframe still contains duplicated FieldCropIds!')
print(f'DataFrame now contains "{len(df_detail_maize_yields)}" rows.')
```

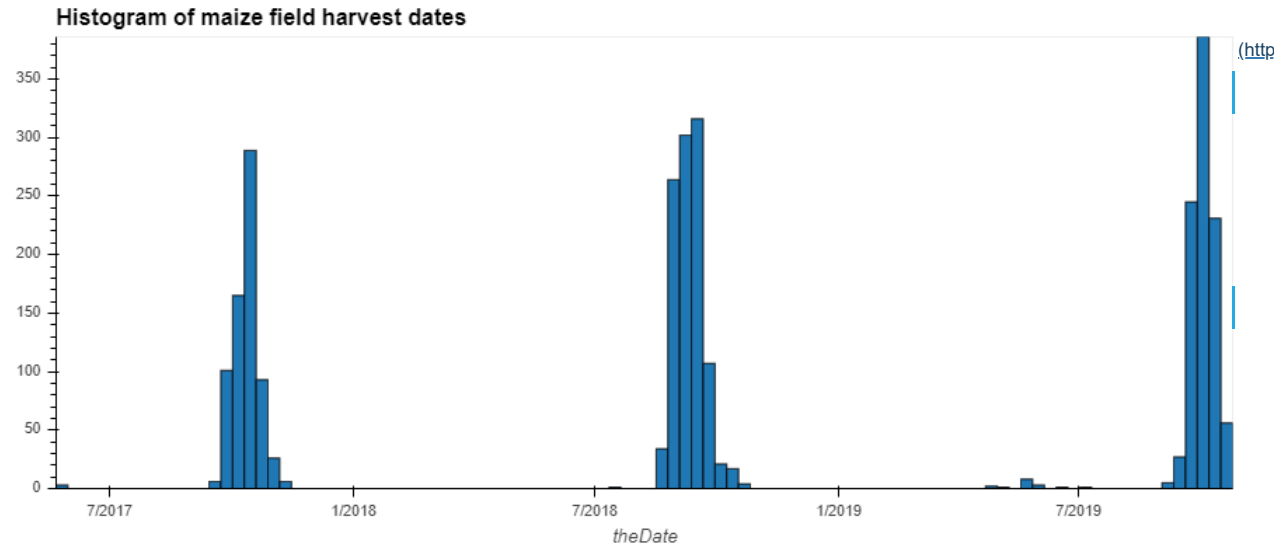
```
DataFrame now contains "2721" rows.
```

Remove fields harvest date outside interval

As seen in the histogram below, some yields have a earlier harvest date (in the "theDate" column) than the majority of the maize fields. We assume the fields with an pearly harvest data as faulty registrations, thus removing them from our analysis.

In [6]:
```
df_detail_maize_yields.theDate.hvplot.hist(
    bins=100, height=400, width=WIDGET_WIDTH,
    title='Histogram of maize field harvest dates')
```

Out[6]:



In [7]:
```
df_detail_maize_yields = df_detail_maize_yields[
    (((df_detail_maize_yields.theDate.dt.month == DATA_CONFIG['valid_harvest_date_interval']['start_month']) &
      (df_detail_maize_yields.theDate.dt.day >= DATA_CONFIG['valid_harvest_date_interval']['start_day'])) |
    (df_detail_maize_yields.theDate.dt.month > DATA_CONFIG['valid_harvest_date_interval']['start_month'])) &
    (((df_detail_maize_yields.theDate.dt.month == DATA_CONFIG['valid_harvest_date_interval']['end_month']) &
      (df_detail_maize_yields.theDate.dt.day <= DATA_CONFIG['valid_harvest_date_interval']['end_day'])) |
    (df_detail_maize_yields.theDate.dt.month < DATA_CONFIG['valid_harvest_date_interval']['end_month']))]

print(f'DataFrame now contains "{len(df_detail_maize_yields)}" rows.')
```

```
DataFrame now contains "2701" rows.
```

Remove fields with yield outside range 5-26 t/ha

We see that the standard deviation is much higher than expected and that the minimum and maximum yield in t/ha is also outside the expected range 5-26 t/ha. Thus we need to remove these outliers.

In [8]:
```
df_detail_maize_yields.drymatterYield_ton.describe()
```

Out[8]:
```
count    2701.000000
mean       14.024317
std        92.787771
min         0.500200
25%         9.677419
50%        11.711227
75%        13.477050
max      3420.000000
Name: drymatterYield_ton, dtype: float64
```

In [9]:
```
df_detail_maize_yields = df_detail_maize_yields[
    df_detail_maize_yields.drymatterYield_ton.between(5, 26)]
print(f'DataFrame now contains "{len(df_detail_maize_yields)}" rows.')
```

```
DataFrame now contains "2618" rows.
```

Add DMDB field polygons to maize yield data

For the later collection of earth observation data we need the area of interest, i.e. the field location. We collect this as the field polygon stored in DMDB (Dansk mark database).

Note that, because not all FieldCropIds in the maize data has a corresponding field polygon in the data fetched from DMDB, we remove the maize yield rows without an polygon.

In [10]:
```
gdf_maize_yields = add_DMDB_polygons(
    df_detail_maize_yields, DATA_CONFIG, client)
```

```
Enter your credentials to the host: "dev-sql-plant.vfltest.dk"

Connection is established using valid credentials.

Starting DMDB data collection...
[########################################] | 100% Completed |  2min 33.1s


df_field_polygons now contains "2563" rows.

Merging maize yield with DMDB field polygons...
[########################################] | 100% Completed |  0.2s


df_detail_maize_yields now contains "2618" rows.

df_detail_maize_polygon_yields now contains "2563" rows.

Constructing GeoDataFrame with field polygons...
[########################################] | 100% Completed |  0.7s


gdf_maize_yields now contains "2563" rows. And we index the fields on : "['FieldCropId', 'harvestYear']".
```

		FarmId	successionNo	sproutingDate	stdCropNormNumber	stdCropName	preCropId	preCropName	theDate	area	registered	yieldNa
FieldCropId	**harvestYear**											
18864494	**2018**	29606	1	NaT	6204	Majshelsæd	5909.0	Vårbyg	2018-09-12	7.83	1	Grø (udbytt
18908801	**2018**	63955	1	NaT	6204	Majs, helsæd	6204.0	Majs, helsæd	2018-08-28	4.29	1	Grø (udbytt
18908802	**2018**	63955	1	NaT	6204	Majs, helsæd	6204.0	Majs, helsæd	2018-08-28	1.63	1	Grø (udbytt
19117041	**2018**	38974	1	NaT	6204	Silomajs	6082.0	Kl.græs, s. 11-30	2018-09-06	3.23	1	Grø (udbytt
19127207	**2018**	51913	1	NaT	6204	Silomajs	5923.0	Vinterhvede	2018-09-04	9.08	1	Grø (udbytt

Remove edge case fields

Remove too narrow fields which will end up with a Dataset polygon mask filled with False

We remove the field with the `FieldCropId == 24359059` as its polygon mask will be filled with False values when rasterized by SatHub. As seen below the polygon has a small diagonal area. This results in that each pixel of the rasterized polygon will have an area overlapping with an area not inside the polygon, thus removed by SatHub as we see such polygons pixels to contains noise.

However, after later iteration of the pipeline, the field "24359059" have already been removed by the section Remove fields with yield outside range 5-26 t/ha, thus the following cells have been changed to "Raw".

FieldCropId_narrow = 24359059 gdf_too_narrow_field = gdf_maize_yields.loc[FieldCropId_narrow, :] print(f'field area of FieldCropId == {FieldCropId_narrow}: ' + f'" {gdf_too_narrow_field.iloc[0].area.compute()}" km2.') viz_gdf_polygons(gdf_too_narrow_field, 'FieldCropId24359059')# Remove the single field from the GeoDataFrame gdf_maize_yields = gdf_maize_yields.drop(index=[FieldCropId_narrow], level='FieldCropId') print('\ngdf_maize_yields now contains ' + f'"{gdf_maize_yields.shape[0].compute()}" rows.')

Remove fields with too cloudy images to create non-null features

We remove fields from the analysis as the collected Sentinel-2 images of them contains to many cloudy images to create non-null features. The problem is that we need cloud free images to interpolate the images values over the growth season. The fields we remove are different for different prediction date hence we remove them accordingly.

In [11]:
```
fieldCropIds_too_cloudy = []

# We use 2019 as a year, but it has no influence on the if statements as they
#    only differ on the dates.
dummy_year = 2019
time_span_end = pd.Timestamp(
    f'{dummy_year}-{FEATURE_HYPERPARAM["time_span"][1]}')
# We only check on the timespan end date as our experiment curently only
#    have changes this date
if time_span_end >= pd.Timestamp(f'{dummy_year}-07-15'):
    fieldCropIds_too_cloudy.extend([
        19274676, 23197646, 19330525, 19330523, 19937848, 19330524, 22701507,
        21345424, 21499353, 21588033])
if time_span_end >= pd.Timestamp(f'{dummy_year}-08-01'):
    fieldCropIds_too_cloudy.extend([20111507])
if time_span_end >= pd.Timestamp(f'{dummy_year}-08-15'):
    fieldCropIds_too_cloudy.extend([19273294, 19273421, 22576367, 23667607])
if time_span_end >= pd.Timestamp(f'{dummy_year}-09-01'):
    fieldCropIds_too_cloudy.extend([
        21313758, 21520350, 21608796, 21861556, 21861623, 22016451, 22017077,
        22495431, 22651406, 23173865, 23173891, 23196074, 23227198, 23384518,
        23474729, 23536045, 23537045, 23537180, 23537189, 23593262, 23667626,
        23849865])

gdf_maize_yields = gdf_maize_yields.drop(
    index=fieldCropIds_too_cloudy, level='FieldCropId')
print(f'The following "{len(fieldCropIds_too_cloudy)}" fields with the '
      'FieldCropId have been removed from the analysis: ' +
      f'"{fieldCropIds_too_cloudy}".')
print('\ngdf_maize_yields now contains ' +
      f'"{gdf_maize_yields.shape[0].compute()}" rows.')
```

```
The following "37" fields with the FieldCropId have been removed from the analysis: "[19274676, 23197646, 19330525, 19330523, 19
937848, 19330524, 22701507, 21345424, 21499353, 21588033, 20111507, 19273294, 19273421, 22576367, 23667607, 21313758, 21520350,
21608796, 21861556, 21861623, 22016451, 22017077, 22495431, 22651406, 23173865, 23173891, 23196074, 23227198, 23384518, 2347472
9, 23536045, 23537045, 23537180, 23537189, 23593262, 23667626, 23849865]".

gdf_maize_yields now contains "2526" rows.
```

Remove following fields due to nan-values from new precrop feature

The field level features `preCropId` and `preCropDirectorateCropNormNumber` with precrop informations result in some of fields containing nan-values. We remove these fields if we have the two mentioned features.

In [12]:
```
fieldCropIds_with_nans = []
if 'preCropDirectorateCropNormNumber' in FEATURE_HYPERPARAM[
        'position_and_precrop_features'] or 'preCropId' in FEATURE_HYPERPARAM[
        'position_and_precrop_features']:
    fieldCropIds_with_nans = gdf_maize_yields[FEATURE_HYPERPARAM[
        'position_and_precrop_features']][
            gdf_maize_yields['preCropDirectorateCropNormNumber'].isna() |
            gdf_maize_yields['preCropId'].isna()].index.get_level_values(
                'FieldCropId').to_list().compute()

    gdf_maize_yields = gdf_maize_yields.drop(
        index=fieldCropIds_with_nans, level='FieldCropId')

print(f'The following "{len(fieldCropIds_with_nans)}" fields with the '
      'FieldCropId have been removed from the analysis: ' +
      f'"{fieldCropIds_with_nans}".')
print('\ngdf_maize_yields now contains ' +
      f'"{gdf_maize_yields.shape[0].compute()}" rows.')
```

```
The following "0" fields with the FieldCropId have been removed from the analysis: "[]".

gdf_maize_yields now contains "2526" rows.
```

Present maize yield data

Present the number of rows and first 5 rows in the GeoDataFrame.

In [13]:
```
print('gdf_maize_yields now contains  ' +
      f'"{gdf_maize_yields.shape[0].compute()}" rows.')
gdf_maize_yields.head().compute()
```

```
gdf_maize_yields now contains  "2526" rows.
```

Out[13]:

		FarmId	successionNo	sproutingDate	stdCropNormNumber	stdCropName	preCropId	preCropName	theDate	area	registered	yieldNa
FieldCropId	**harvestYear**											
18864494	**2018**	29606	1	NaT	6204	Majshelsæd	5909.0	Vårbyg	2018-09-12	7.83	1	Grø (udbytt
18908801	**2018**	63955	1	NaT	6204	Majs, helsæd	6204.0	Majs, helsæd	2018-08-28	4.29	1	Grø (udbytt
18908802	**2018**	63955	1	NaT	6204	Majs, helsæd	6204.0	Majs, helsæd	2018-08-28	1.63	1	Grø (udbytt
19117041	**2018**	38974	1	NaT	6204	Silomajs	6082.0	Kl.græs, s. 11-30	2018-09-06	3.23	1	Grø (udbytt
19127207	**2018**	51913	1	NaT	6204	Silomajs	5923.0	Vinterhvede	2018-09-04	9.08	1	Grø (udbytt

Present non-null row count, mean, std, min, 25% percentile, 50% percentile, 75% percentile, and max per numeric column in the GeoDataFrame.

In [14]:
```
gdf_maize_yields.describe().compute()
```

Out[14]:

	FarmId	successionNo	stdCropNormNumber	preCropId	area	registered	quantity	normQuantity	totalQuantity	unitId	qualitypara
count	2526.000000	2526.000000	2526.000000	2.513000e+03	2526.000000	2526.000000	2526.000000	0.0	2526.000000	2526.0	
mean	47834.109660	1.014648	6276.264450	1.094050e+04	6.846952	0.993666	34.553487	NaN	237.921741	3.0	
std	26079.709224	0.120162	737.317685	8.792603e+04	5.492695	0.079350	8.560839	NaN	200.300580	0.0	
min	6542.000000	1.000000	6204.000000	5.909000e+03	0.020000	0.000000	13.111785	NaN	0.556364	3.0	
25%	27499.000000	1.000000	6204.000000	6.086000e+03	3.010000	1.000000	29.051742	NaN	96.595750	3.0	
50%	41022.000000	1.000000	6204.000000	6.204000e+03	5.350000	1.000000	34.478365	NaN	182.723168	3.0	
75%	68503.000000	1.000000	6204.000000	6.204000e+03	9.330000	1.000000	39.899170	NaN	314.418275	3.0	
max	95446.000000	2.000000	13803.000000	1.678547e+06	55.000000	1.000000	76.493151	NaN	1825.686000	3.0	

Present non-null row count, unique count, top (most common value), frequency (most common value's frequency), first value, and last value per text column in the GeoDataFrame.

In [15]:
```
gdf_maize_yields.describe(include='object').compute()
```

Out[15]:

	stdCropName	preCropName	yieldName	unitName	QualityName	dirCrop	dirPreCrop	Postdistrikt	KOMNAVN	regionNavn
count	2526	2513	2526	2526	2526	2526	2513	2425	2512	2526
unique	11	109	1	1	1	1	37	87	25	3
top	Silomajs	Silomajs	Grønmasse (udbyttemåling)	ton	Tørstof %	Silomajs	Silomajs	Rødding	Vejen	Syddanmark
freq	1101	776	2526	2526	2526	2526	1579	151	419	1315

Present non-null row count, unique count, top (most common value), frequency (most common value's frequency), first timestamp, and last timestamp per time column in the GeoDataFrame.

In [16]:
```
gdf_maize_yields.describe(include='datetime64[ns]').compute()
```

Out[16]:

	sproutingDate	theDate
count	112	2526
unique	29	123
top	2019-05-17 00:00:00	2018-09-06 00:00:00
freq	21	74
first	2017-05-08 00:00:00	2017-09-15 00:00:00
last	2019-05-17 00:00:00	2019-10-25 00:00:00

Present yearly non-null row count, mean, std, min, 25% percentile, 50% percentile, 75% percentile, and max for `drymatterYield_ton` :

In [17]:
```
seleted_columns = ['drymatterYield_ton']
gdf_maize_yields[seleted_columns].groupby(
    level='harvestYear').describe().T.compute()
```

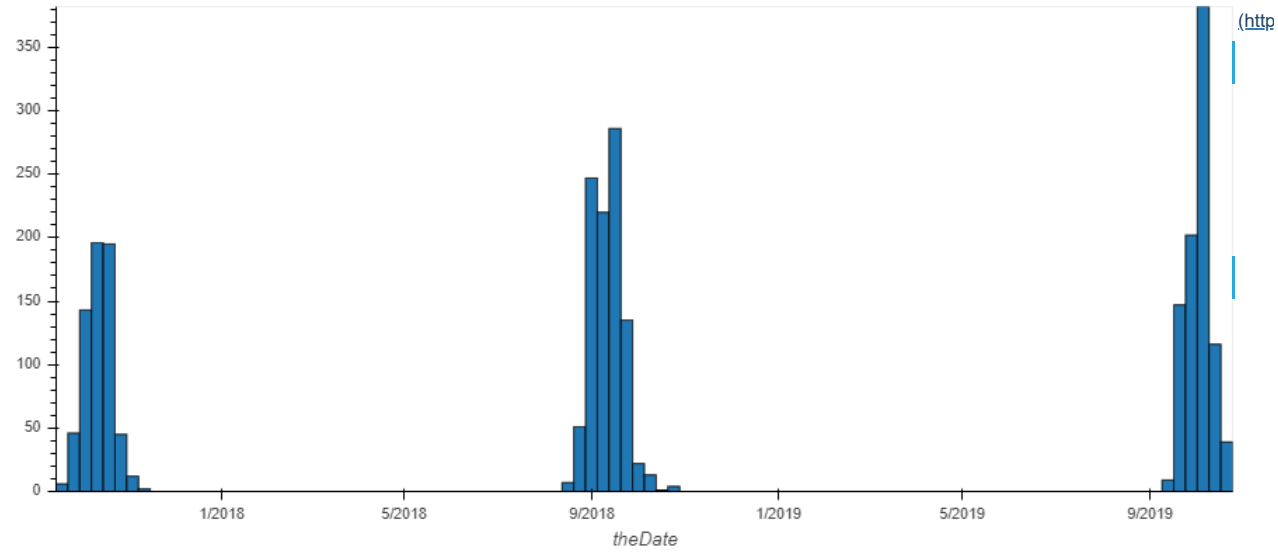
Out[17]:

	harvestYear	2017	2018	2019
drymatterYield_ton	**count**	645.000000	986.000000	895.000000
	mean	11.668460	11.166845	12.581244
	std	2.500917	3.071559	2.751556
	min	5.227670	5.162719	5.011600
	25%	10.033602	9.011576	11.138400
	50%	11.719155	11.140094	12.599154
	75%	13.155224	12.944145	14.306906
	max	22.745600	25.015365	25.070468

Histogram of harvest dates.

In [18]:
```
gdf_maize_yields.theDate.compute().hvplot.hist(
    bins=100, height=400, width=WIDGET_WIDTH)
```

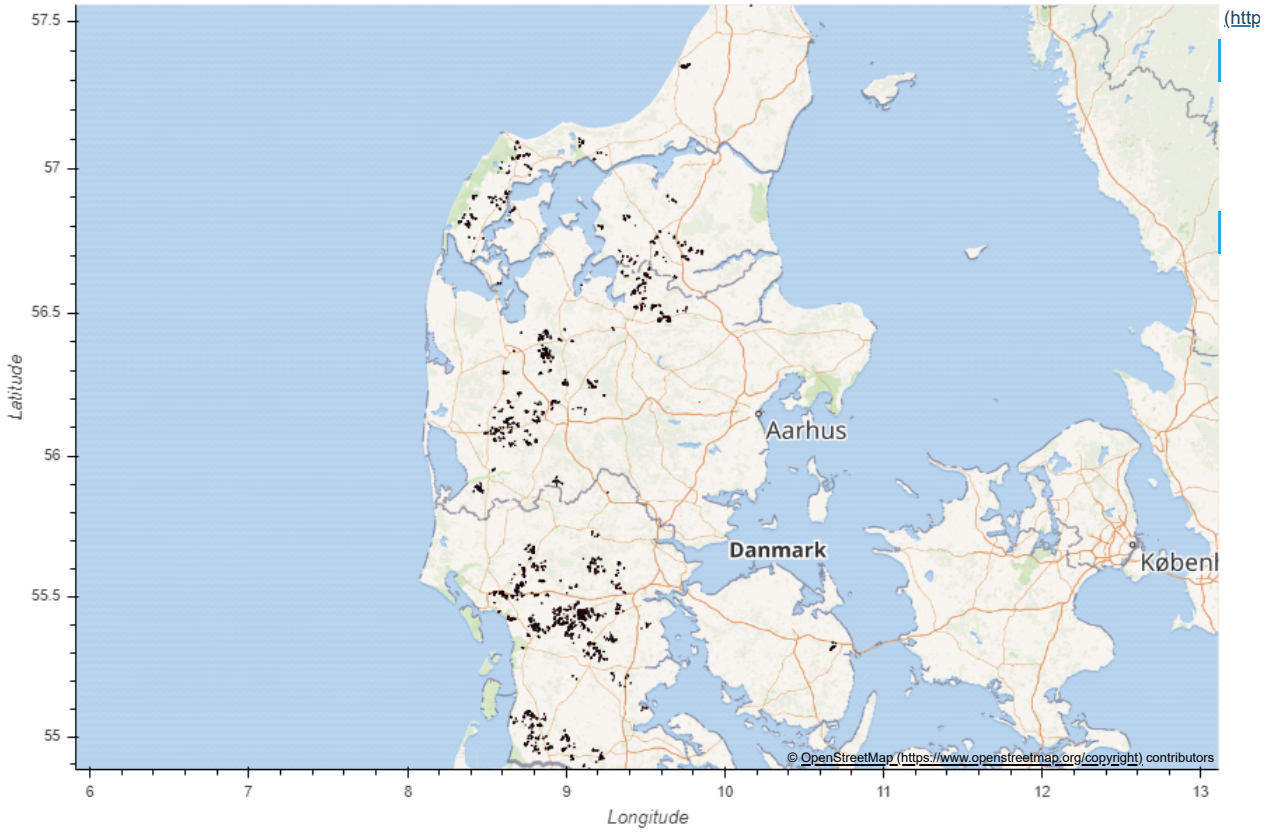
Out[18]:



Visualize maize field locations

In [19]:
```
gv_field = viz_gdf_polygons(
    gdf_maize_yields, title='Field polygons', width=WIDGET_WIDTH)
gv_field
```

Out[19]:



Save maize yield GeoDataFrame to file

In [20]:
```
gdf_maize_yields_saved = gdf_maize_yields
del gdf_maize_yields
# Reset index to save the index as columns, because `to_file()` do not save the
#   index to file.
gdf_maize_yields_reset_index = gdf_maize_yields_saved.reset_index()
gdf_maize_yields_reset_index.to_file(
    DATA_PATH/'gdf_maize_yields.gpkg', layer='maize_yields',
    driver="GPKG").compute()
del gdf_maize_yields_reset_index
```

Reload the maize yield GeoDataFrame from file

In [21]:
```
# Load and set indeces on GeoDataFrame
gdf_maize_yields = dask.delayed(gpd.read_file)(
    DATA_PATH/'gdf_maize_yields.gpkg')
gdf_maize_yields = gdf_maize_yields.set_index(
    DATA_CONFIG['sample_indices_names'])

# Change dtypes to same as saved GeoDataFrame
gdf_maize_yields = gdf_maize_yields.assign(
    sproutingDate=gdf_maize_yields.sproutingDate.astype('datetime64'),
    theDate=gdf_maize_yields.theDate.astype('datetime64'),
    normQuantity=gdf_maize_yields.normQuantity.astype('float64'))
```

In [22]:
```
check_gdf_equals(gdf_maize_yields_saved, gdf_maize_yields)
del gdf_maize_yields_saved
```

NOTE: we remove or mark this cell as raw, for testing purposes gdf_maize_yields = gdf_maize_yields.sample(10, random_state=42) gdf_maize_yields = client.persist(gdf_maize_yields) gdf_maize_yields.compute()

Collect satellite, DTM, and DMI data

We collect satellite images, DTM (Danish terrain model) data, and Danish Meteorological Institute (DMI) data.

In [23]:
```
run_EO_dataset_pipeline(
    gdf_maize_yields, EO_CONFIG, client)
```

```
Starting Sathub workflow...
[########################################] | 100% Completed | 20min 19.8s
```

Machine learning workflow

Create features and target

In [24]:
```
df_features, df_target = run_featurization_pipeline(
    gdf_maize_yields, EO_CONFIG['data_path'], DATA_CONFIG, FEATURE_CONFIG,
    FEATURE_HYPERPARAM, client)
```

```
Starting featurization...
[########################################] | 100% Completed |  7min  4.8s

df_features now contains "2526" samples and "1087" features.
```

		S2_L1C_B01_04-01 - April 1	S2_L1C_B01_04-08 - April 8	S2_L1C_B01_04-15 - April 15	S2_L1C_B01_04-22 - April 22	S2_L1C_B01_04-29 - April 29	S2_L1C_B01_05-06 - May 6	S2_L1C_B01_05-13 - May 13	S2_L1C_B01-20 - May 20
FieldCropId	**harvestYear**								
16456433	**2017**	0.146990	0.144004	0.141018	0.138031	0.135045	0.132059	0.132992	0.13
16456460	**2017**	0.145500	0.143364	0.141239	0.139114	0.136989	0.134864	0.136347	0.13
18099423	**2019**	0.133002	0.153474	0.140524	0.124062	0.129247	0.127983	0.126718	0.12
18099424	**2019**	0.134235	0.155353	0.141377	0.124055	0.129546	0.128706	0.127865	0.12
18099425	**2019**	0.134846	0.152259	0.141530	0.124549	0.126453	0.121435	0.122000	0.12

5 rows × 1087 columns

```
df_target now contains "2526" samples and "1" target.
```

		drymatterPercent
FieldCropId	**harvestYear**	
16456433	**2017**	34.676
16456460	**2017**	36.891
18099423	**2019**	30.900
18099424	**2019**	29.690
18099425	**2019**	29.223

Split features and target into training and validation sets

In [25]:
```
(df_train_features, df_val_features,
 df_train_target, df_val_target) = run_split_data_pipeline(
    df_features, df_target, ML_CONFIG, client)
```

```
Splitting data randomly into "0.8/0.2" training and validation set, respectively...
```

First 5 rows in training feature DataFrame:

		S2_L1C_B01_04-01 - April 1	S2_L1C_B01_04-08 - April 8	S2_L1C_B01_04-15 - April 15	S2_L1C_B01_04-22 - April 22	S2_L1C_B01_04-29 - April 29	S2_L1C_B01_05-06 - May 6	S2_L1C_B01_05-13 - May 13	S2_L1C_B01-20 - May 20
FieldCropId	**harvestYear**								
19099351	**2017**	0.126654	0.127923	0.129193	0.130462	0.131731	0.132483	0.132545	0.13
23566976	**2019**	0.123926	0.152000	0.138670	0.127840	0.148555	0.149164	0.148662	0.15
23593290	**2019**	0.135063	0.123074	0.132935	0.131010	0.137516	0.140021	0.142525	0.15
23594526	**2019**	0.143858	0.131961	0.140074	0.134050	0.144998	0.158149	0.143220	0.14
19080099	**2017**	0.143357	0.144811	0.145693	0.146575	0.147457	0.127240	0.130425	0.13

5 rows × 1087 columns

First 5 rows in training target DataFrame:

		drymatterPercent
FieldCropId	**harvestYear**	
19099351	**2017**	36.664
23566976	**2019**	34.718
23593290	**2019**	28.000
23594526	**2019**	39.554
19080099	**2017**	42.077

First 5 rows in validation feature DataFrame:

		S2_L1C_B01_04-01 - April 1	S2_L1C_B01_04-08 - April 8	S2_L1C_B01_04-15 - April 15	S2_L1C_B01_04-22 - April 22	S2_L1C_B01_04-29 - April 29	S2_L1C_B01_05-06 - May 6	S2_L1C_B01_05-13 - May 13	S2_L1C_B01-20 - May 20
FieldCropId	**harvestYear**								
23526633	**2018**	0.144604	0.147815	0.145275	0.149693	0.143042	0.134893	0.148858	0.14
21247344	**2018**	0.148126	0.148161	0.146736	0.145866	0.143454	0.138483	0.142393	0.14
19126864	**2017**	0.132990	0.134579	0.136167	0.137756	0.139345	0.139817	0.138800	0.13
21470608	**2018**	0.453744	0.148822	0.140510	0.136469	0.124326	0.119678	0.137719	0.13
22001372	**2018**	0.139936	0.136129	0.134022	0.129035	0.129650	0.130265	0.143127	0.13

5 rows × 1087 columns

First 5 rows in validation target DataFrame:

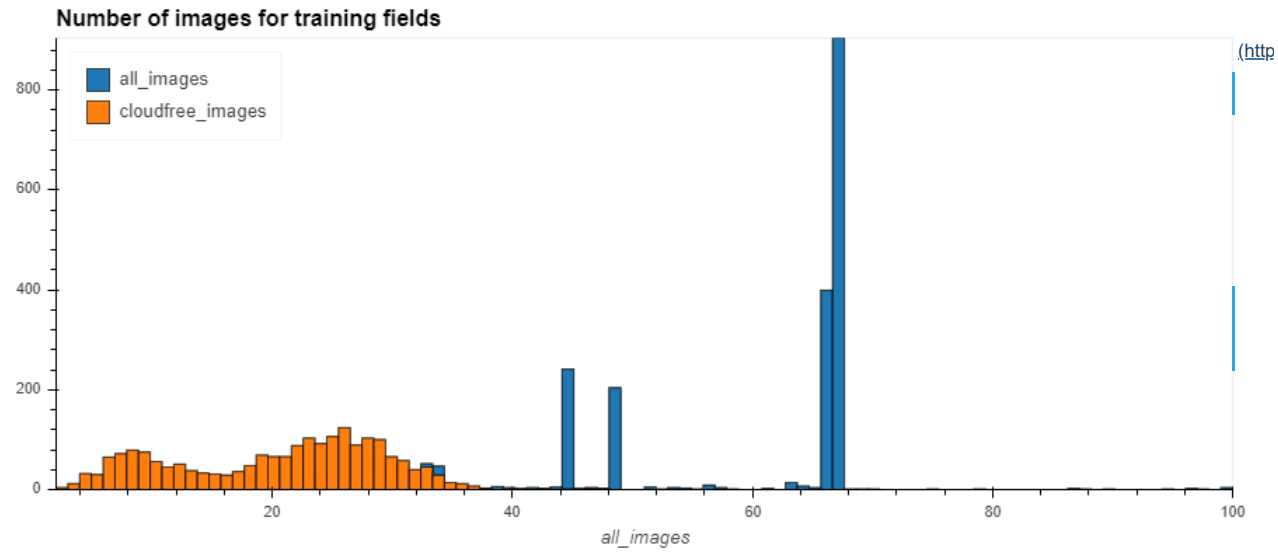
		drymatterPercent
FieldCropId	**harvestYear**	
23526633	**2018**	34.437
21247344	**2018**	37.634
19126864	**2017**	37.367
21470608	**2018**	35.531
22001372	**2018**	38.700

Present number of cloudfree images per field

In [26]:
```
df_number_images_train = run_number_of_images(
    df_train_features.index.compute(), EO_CONFIG['data_path'],
    FEATURE_HYPERPARAM['time_span']).compute()
df_number_images_val = run_number_of_images(
    df_val_features.index.compute(), EO_CONFIG['data_path'],
    FEATURE_HYPERPARAM['time_span']).compute()
mean_number_images = [
    df_number_images_train.mean().add_suffix('_mean_train').to_dict(),
    df_number_images_val.mean().add_suffix('_mean_val').to_dict()]
```

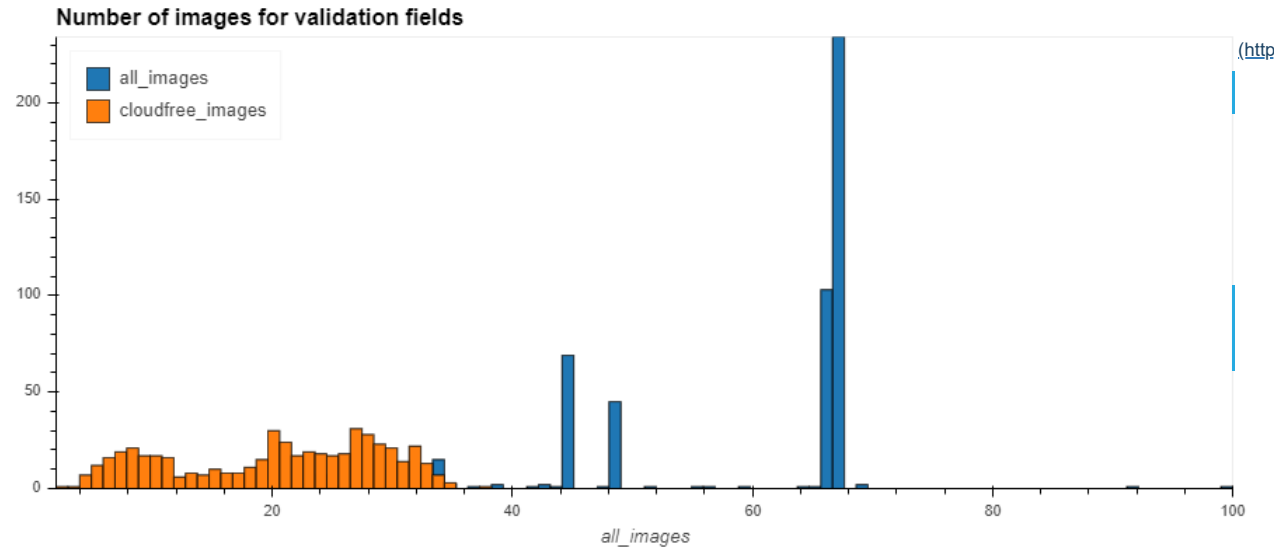
In [27]:
```
df_number_images_train.hvplot.hist(
    bins=100, height=400, width=WIDGET_WIDTH,
    title='Number of images for training fields').opts(
    legend_position='top_left')
```

Out[27]:



In [28]:
```
df_number_images_val.hvplot.hist(
    bins=100, height=400, width=WIDGET_WIDTH,
    title='Number of images for validation fields').opts(
    legend_position='top_left')
```

Out[28]:



In [29]:
```
print('Present non-null row count, mean, std, min, 25% percentile, ' +
      '50% percentile, 75% percentile, and max number of images over ' +
      'all training fields.')
display(df_number_images_train.describe())

print('Present non-null row count, mean, std, min, 25% percentile, ' +
      '50% percentile, 75% percentile, and max number of images over ' +
      'all validation fields.')
display(df_number_images_val.describe())
```

```
Present non-null row count, mean, std, min, 25% percentile, 50% percentile, 75% percentile, and max number of images over all tr
aining fields.
```

	all_images	cloudfree_images
count	2020.000000	2020.000000
mean	58.939604	20.370792
std	12.321607	8.781799
min	24.000000	2.000000
25%	49.000000	12.000000
50%	66.000000	22.000000
75%	67.000000	27.000000
max	100.000000	48.000000

```
Present non-null row count, mean, std, min, 25% percentile, 50% percentile, 75% percentile, and max number of images over all va
lidation fields.
```

	all_images	cloudfree_images
count	506.000000	506.000000
mean	59.215415	20.288538
std	11.970055	8.827486
min	24.000000	2.000000
25%	49.000000	11.250000
50%	66.000000	21.000000
75%	67.000000	28.000000
max	100.000000	38.000000

Train machine learning model

In [30]:
```
df_train_predictions = run_training_pipeline(
    df_train_features, df_train_target, ML_HYPERPARAM['sklearn_regressor'],
    MODEL_KEY_ARGS, ML_CONFIG, client)
```

```
Training and predicting on training set...


First 5 training predictions.
```

		drymatterPercent	predictions
FieldCropId	**harvestYear**		
19099351	**2017**	36.664	37.693395
23566976	**2019**	34.718	35.583079
23593290	**2019**	28.000	30.321763
23594526	**2019**	39.554	35.652937
19080099	**2017**	42.077	38.955448

Validate trained model

In [31]:
```
df_val_predictions = run_prediction_pipeline(
    df_val_features, df_val_target, ML_CONFIG, client)
```

```
Predicting on validation set using trained model...
```

First 5 validation predictions.

FieldCropId	harvestYear	drymatterPercent	predictions
23526633	**2018**	34.437	34.971181
21247344	**2018**	37.634	35.019879
19126864	**2017**	37.367	38.222145
21470608	**2018**	35.531	37.789435
22001372	**2018**	38.700	34.855329

Present model scores

In [32]:
```
df_scores = present_model_scores(
    df_train_predictions, df_val_predictions, FEATURE_HYPERPARAM, client)
```

```
Computing model scores on the training and validation set...
```

	MAE	R2	Samples
Training	1.854630	0.600412	2020
Validation	2.289356	0.267158	506

Save experiment in MLflow

In [33]:
```
artifacts = {} # TODO: add polygon plot file as artifact
save_MLflow_experiment(
    df_scores, mean_number_images, CONFIGURATIONS, HYPERPARAMETERS, DATA_PATH,
    artifacts, client)
```

```
The experiment have been saved in MLflow.
```

Cleanup files on Ahsoka

In [34]:
```
answer = query_yes_no(
    'Do you want to delete all MPSI files on Ahsoka?', default='no', timeout=5)

if answer:
    print('Deleting all MPSI files from Ahsoka...')
    dask.delayed(shutil.rmtree)(DATA_PATH).compute()
    print('All MPSI files from Ahsoka have been deleted')
else:
    print('Not deleting any MPSI files from Ahsoka.')
```

```
Do you want to delete all MPSI files on Ahsoka? [y/N]
You have "5" seconds to answer.
Timed out.
Default answer: "no" was selected.
Not deleting any MPSI files from Ahsoka.
```

Present MLflow experiments

In [35]:
```
df_experiments = collect_MLflow_runs(DATA_PATH)
# Show only experiments:
#    - where the "expr" tag have been set, and
#    - after no. 8, i.e. where dataset have been filtered on yield internal
df_experiments = df_experiments[df_experiments['tags.expr'].notnull()]
filter_expr_no = 8
filter_expr_drymatter_percentage = 30
```

```
Experiment: "MPSI-15_maize_field_level_yield_prediction" has id: "7".
```

Maize dry matter yield

In [36]:

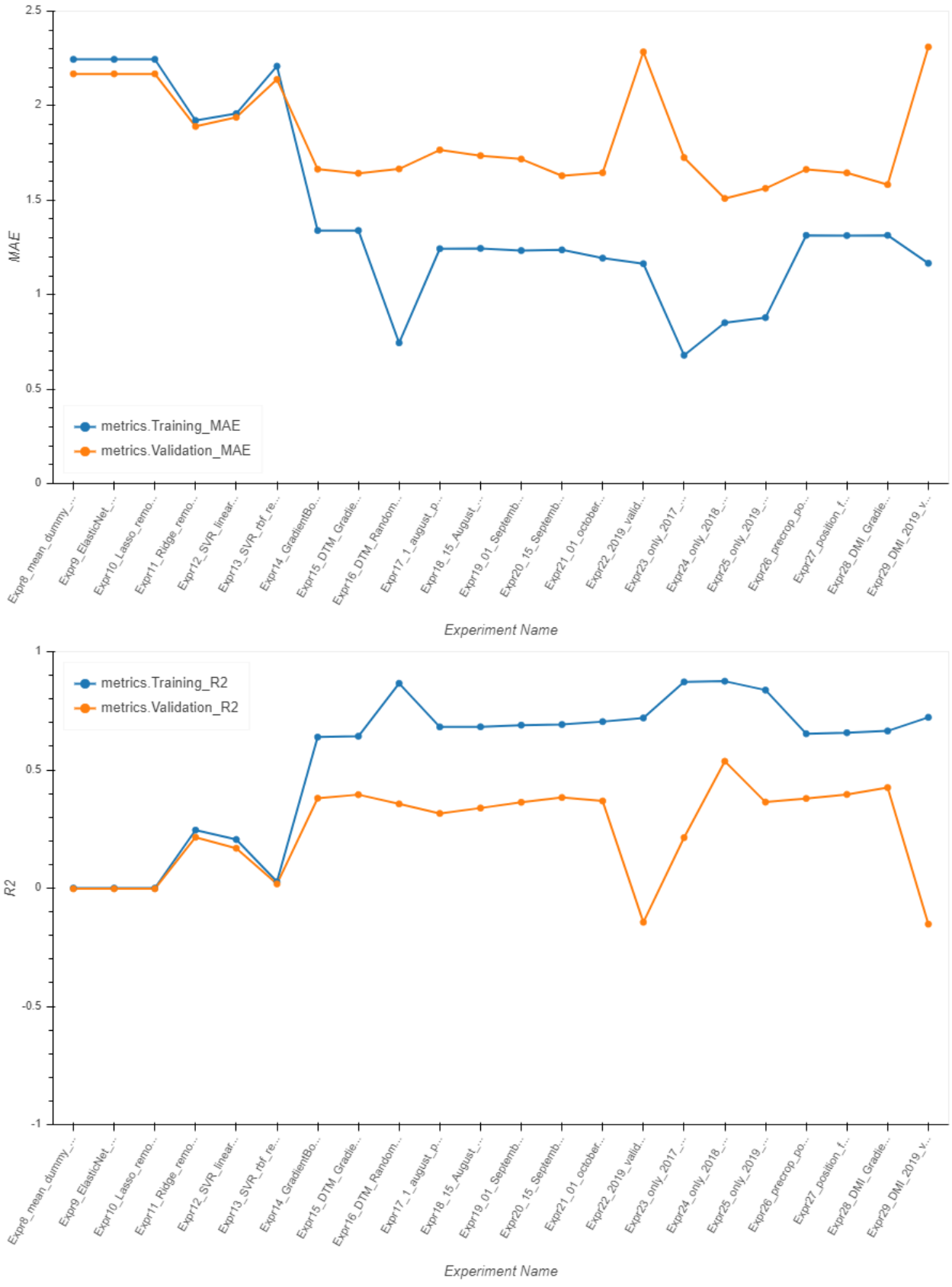
```
df_experiments_yield = df_experiments[
    (df_experiments['tags.expr'].astype(int) >= filter_expr_no) &
    (df_experiments[
        'tags.expr'].astype(int) < filter_expr_drymatter_percentage)]
display(show_score_table(df_experiments_yield))
viz_model_scores(
    df_experiments_yield, WIDGET_WIDTH, HV_HEIGHT, HV_XROTATION)
```

	Training			Validation		
	Samples	MAE	R2	Samples	MAE	R2
Name						
Expr8_mean_dummy_baseline_removed_yield_outliers	2042	2.245	0.000	511	2.168	-0.003
Expr9_ElasticNet_removed_yield_outliers	2042	2.245	0.000	511	2.168	-0.003
Expr10_Lasso_removed_yield_outliers	2042	2.245	0.000	511	2.168	-0.003
Expr11_Ridge_removed_yield_outliers	2042	1.922	0.245	511	1.890	0.215
Expr12_SVR_linear_removed_yield_outliers	2042	1.958	0.206	511	1.938	0.169
Expr13_SVR_rbf_removed_yield_outliers	2042	2.209	0.027	511	2.139	0.018
Expr14_GradientBoosting_default_removed_yield_outliers	2042	1.340	0.639	511	1.664	0.380
Expr15_DTM_GradientBoosting	2042	1.340	0.642	511	1.642	0.395
Expr16_DTM_RandomForest	2042	0.745	0.866	511	1.665	0.356
Expr17_1_august_prediction	2041	1.243	0.682	511	1.766	0.316
Expr18_15_August_prediction	2041	1.245	0.682	511	1.736	0.339
Expr19_01_September_prediction	2041	1.233	0.689	511	1.718	0.363
Expr20_15_September_prediction	2038	1.237	0.692	510	1.629	0.383
Expr21_01_october_prediction	2020	1.193	0.704	506	1.645	0.369
Expr22_2019_validation_and_2017_2018_train	1644	1.164	0.720	909	2.285	-0.144
Expr23_only_2017_data	518	0.679	0.872	130	1.725	0.214
Expr24_only_2018_data	796	0.851	0.875	200	1.509	0.537
Expr25_only_2019_data	727	0.879	0.838	182	1.562	0.364
Expr26_precrop_position_features	2032	1.314	0.653	508	1.663	0.379
Expr27_position_features	2042	1.313	0.657	511	1.645	0.396
Expr28_DMI_Gradient_Boosting	2042	1.314	0.665	511	1.582	0.425
Expr29_DMI_2019_validation_and_2017_2018_train	1644	1.166	0.722	909	2.311	-0.152

Out[36]:

Model score over experiments

(https://bol



Maize dry matter percentage

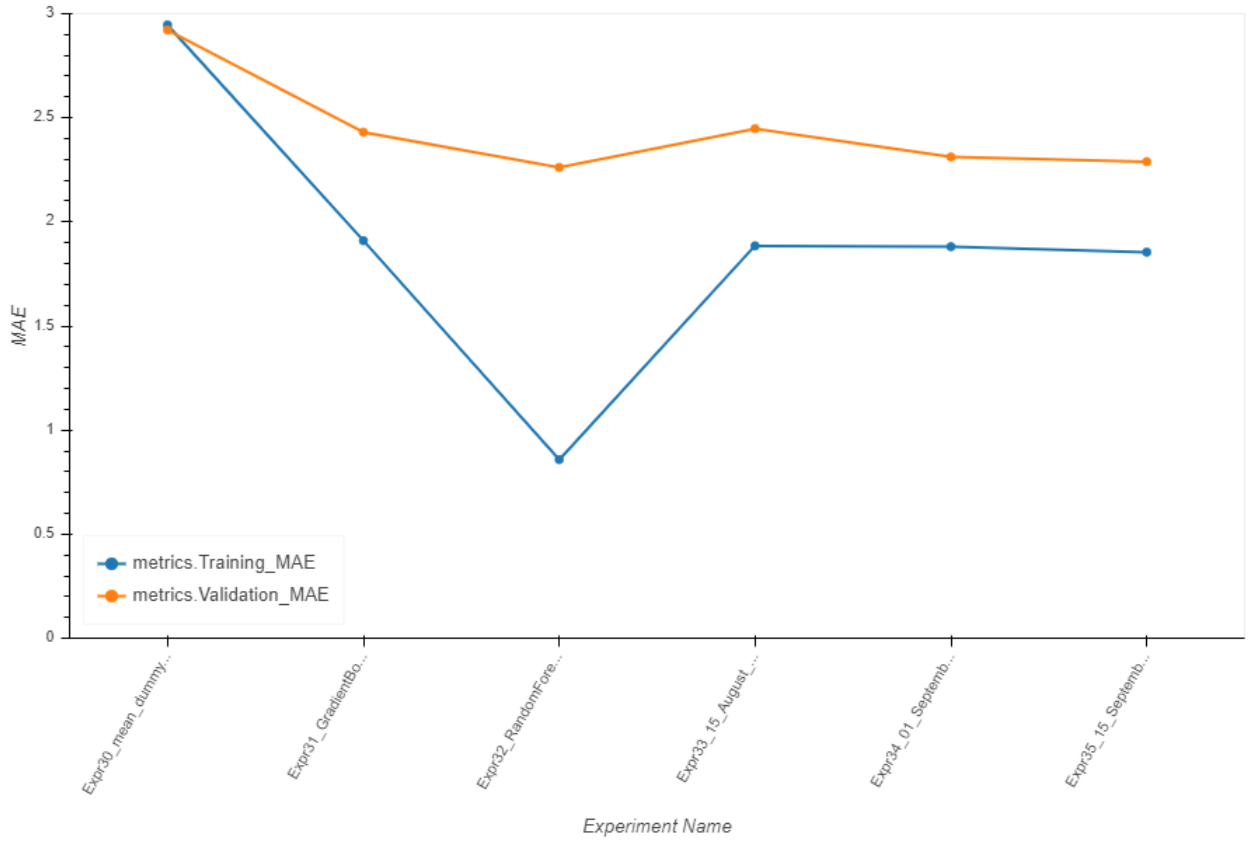
In [37]:

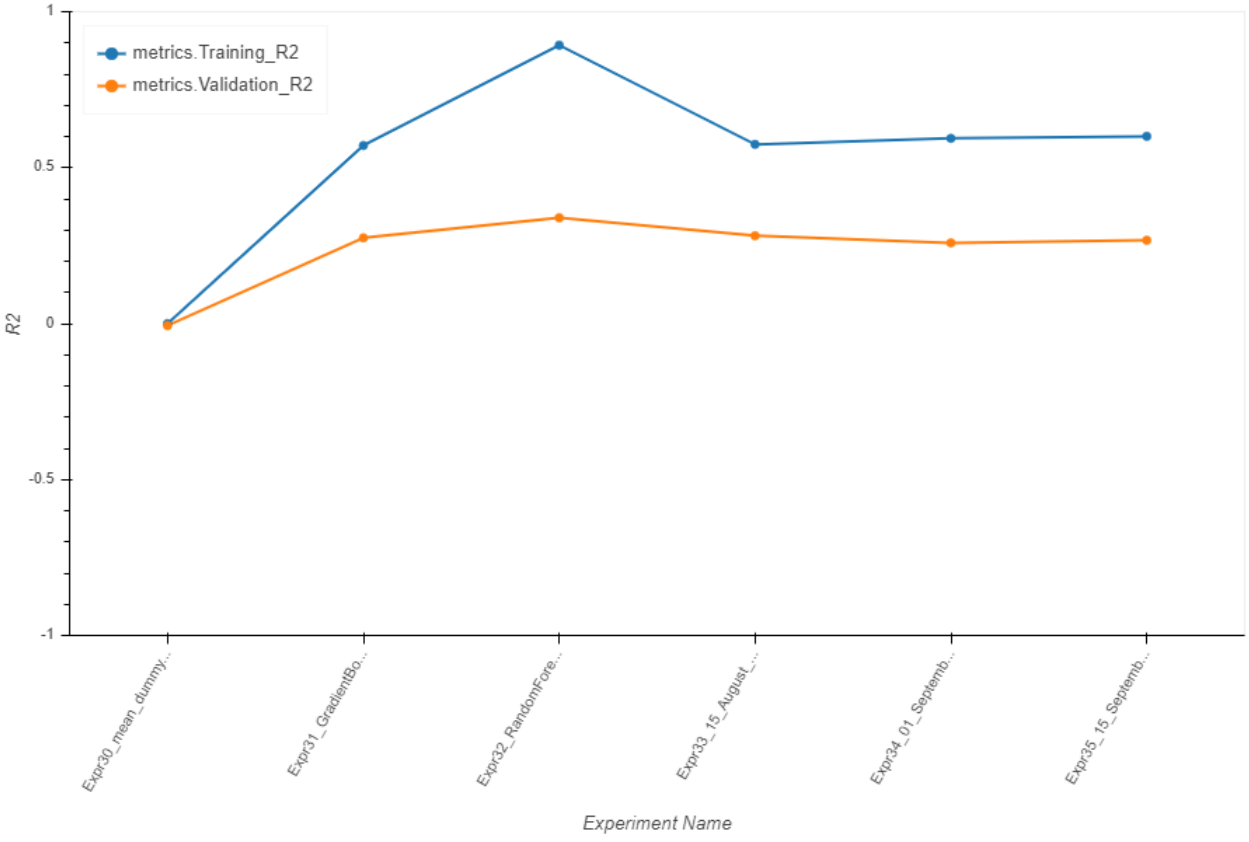
```
df_experiments_drymatter = df_experiments[
    df_experiments[
        'tags.expr'].astype(int) >= filter_expr_drymatter_percentage]
display(show_score_table(df_experiments_drymatter))
viz_model_scores(
    df_experiments_drymatter, WIDGET_WIDTH, HV_HEIGHT,
    HV_XROTATION, y_lim_MAE=(0, 3))
```

	Training			Validation		
	Samples	MAE	R2	Samples	MAE	R2
Name						
Expr30_mean_dummy_baseline_drymatter_prediction	2042	2.946	0.000	511	2.922	-0.006
Expr31_GradientBoosting_drymatter_prediction	2042	1.910	0.571	511	2.431	0.275
Expr32_RandomForest_drymatter_prediction	2042	0.858	0.892	511	2.262	0.339
Expr33_15_August_drymatter_prediction	2038	1.885	0.575	510	2.448	0.282
Expr34_01_September_drymatter_prediction	2020	1.881	0.594	506	2.312	0.259
Expr35_15_September_drymatter_prediction	2020	1.855	0.600	506	2.289	0.267

Out[37]:

Model score over experiments





Experiment descriptions

This section describes the maize yield prediction experiments, that are visualized above, in details. We describe all experiment settings in details such that the following descriptions only explain the setting difference between the experiments. Our experiment pipeline is divided into the three steps:

1. data collection and cleaning: The available maize data consists of data of 3114 individual fields from the years 2017, 2018, and 2019. However, after adding available field polygons from DMDB and data cleaning (see details in top of this notebook) we end up with a dataset of 2553 fields. We also collect Sentinel-2 (S2) L1C satellite images of the polygon areas in a 10x10 meter resolution consisting of all 13 bands, computed the normalized difference red edge index (NDRE), and cloud mask using the S2cloudless algorithm. These resulting cloud free images are used as features for the model.
2. featurization: To simplify the spatial dimension of the S2 data we take the mean of each satellite image (i.e. all 13 bands and NDRE) for only the pixels fully within the field polygon, thus we obtain one feature sample per field and not per pixel. To simplify the temporal dimension of the S2 data we interpolate linearly to a interval of 7 days, thus creating a feature each 7 day in the time period from 1 April to 15 July resulting a feature set of 224 S2 feature given to the model. The target for the model are the maize yield values from the column "drymatterYield_ton".
3. machine learning split, model training, and prediction: We split the maize data into training and validation set based on a 80/20 % split, respectively, resulting in 2042 training samples and 511 validation samples. We use the dummy regression method described above as the trained model in experiment 8 and use it for predictions and scoring on the training and validation set.

Maize yield experiments

The results start from experiment 8 because we decided to exclude the experiments where we had not removed the outliers that were disturbing the predictions.

- **Expr8_mean_dummy_baseline_removed_yield_outliers**: First we established a baseline for predicting maize yield using a dummy regression method, which simply always predicts the constant value being the mean yield of the training targets. Such baseline results in a training mean absolute error (MAE) of 2.245 t/ha and validation MAE of 2.168 t/ha and a coefficient of determination (R^2) about 0 for both training and validation.
- **Expr9_ElasticNet_removed_yield_outliers**: We only change the model in the experiment pipeline, to an ElasticNet model which is a linear regression with combined L1 and L2 priors as regularizer. The model scores do not change at all, from the ones reported for experiment 8, thus the ElasticNet model must have stabilized on the same constant as the mean.
- **Expr10_Lasso_removed_yield_outliers**: We again only change the model in the experiment pipeline, to an Lasso model which is a linear regression with only L1 prior as regularizer and again the model scores do not change at all.
- **Expr11_Ridge_removed_yield_outliers**: We change the model to an Ridge model which is a linear regression with only L2 as regularizer. However, experiment 11 results in a decreased validation MAE of 1.89 t/ha from the MAE of 2.168 t/ha see in experiment 8, 9, and 10. The validation R^2 also increased from 0 to 0,245.
- **Expr12_SVR_linear_removed_yield_outliers**: We change the model to an support vector regression (SVR) model using a linear kernel. This model worsened the scores as it results in an increased validation MAE to 1.938 t/ha and a decreased in R^2 to 0,169.
- **Expr13_SVR_rbf_removed_yield_outliers**: We keep using a support vector regression (SVR) model however this time with a radial basis function (RBF) kernel which enables the SVR to be the first non-linear regression model we have experimented on the maize yield data. However, it worsened the scores further to a validation MAE of 2.139 t/ha and R^2 scores both about 0.
- **Expr14_GradientBoosting_default_removed_yield_outliers**: We changed the model to a Gradient Boosting regression model, which is a more complex non-linear model that builds multiple regression tree used in an ensemble fashion to provide a final prediction. It improved the scores drastically as the training MAE decrease to 1.34, validation MAE decrease to 1.664 t/ha, and the validation R^2 increased to 0.38.
- **Expr15_DTM_GradientBoosting**: Using the Gradient Boosting model, we added the Danish terrain model (DTM) features (i.e. relative hight, percentage of slope, angle of slope, aspect of slope) and the resulting MAE on the validation set was 1.642 t/ha, which is not a significant decrease in the error. The R^2 was 0.395 on the validation set.
- **Expr16_DTM_RandomForest**: A Random Forest regression model was tried and it resulted in same validation scores with MAE 1.665 t/ha, however it decreased on the training score drastically with MAE of 0.745 t/ha, thus overfitting the data. This was expected as a Random forest algorithm does not handle minimization of the data bias which Gradient Boosting algorithm does. Thus, we decide to choose the Gradient Boosting model in future experiments.

These experiments indicated that Gradient Boosting is the best algorithm as it produces the best validation MAE of around 1.6 t/ha and with less overfitting as the training MAE is 1.34 t/ha. The experiments also show that the S2 bands contain the most descriptive information wrt. maize yield, as adding the DTM features did not significantly improved the MAE.

- **Expr17_1_august_prediction**: Changing the features to include S2 data from 1 April to 1 August (i.e. prediction date of 1 August) increased the validation MAE a bit to 1.766 and R^2 decreased to 0.316 in comparison with the best results in experiment 15.
- **Expr18_15_August_prediction**: Changing the prediction date to 15 August increased the validation MAE to 1.736 t/ha and R^2 decreased to 0.339 wrt. experiment 15.
- **Expr19_01_September_prediction**: Changing the prediction date to 1 September increased the validation MAE to 1.718 t/ha and R^2 decreased to 0.363 wrt. experiment 15.
- **Expr20_15_September_prediction**: Changing the prediction date to 15 September increased the validation MAE to 1.629 t/ha and R^2 decreased to 0.383 wrt. experiment 15.
- **Expr21_01_october_prediction**: Changing the prediction date to 1 October increased the validation MAE to 1.645 t/ha and R^2 decreased to 0.369 wrt. experiment 15.

These experiments indicated that adding S2 data in the period from 1 august to 1 September, increases the MAE, thus we assume the S2 data from this period contains differences between fields or noise of some sort. One explanation could be the difference of flowering between maize varieties. Hence, we continued our experiments using S2 data of the period from 1 April to 15 July.

- **Expr22_2019_validation_and_train_2017_2018**: In this experiment we trained a Gradient Boosting model on data from 2017 and 2018 and validated it only on data from 2019. This resulted in a validation MAE of 2.285 t/ha and a R^2 of -0.144.

We expected the MAE to worsen as the model is not trained on any samples from the same year as it is validate on. However, putting a maize yield model into production would have such challenges, as it would deliver predictions on the 15 July but the ground truth data is not obtained until harvest later in the year. One should expect an MAE of around 2.285 t/ha if the model is set in production to predict the maize yields of 2020, however it should be improved if the 2019 data is included in its training set.

- **Expr23_only_2017_data**: We ran the experiment pipeline only on data from 2017, which resulted in MAE on the validation set of 1.725 t/ha and an R^2 of 0.214. We also reported the mean number of available S2 images and available cloud-free images over the fields in the period April 1st to July 15th used as features for the Gradient Boosting model. In 2017 there were 4.87/22.7 (available cloud-free images out of available all S2 images) cloudless images per field on average per field on the training set and 4.63/22.69 cloudless images on average per field on the validation set.
- **Expr24_only_2018_data**: Running the pipeline only on data from 2018 resulted in a MAE on the validation set of 1.509 t/ha and an R^2 of 0.537. In 2018 there was 19.91/40.61 of cloudless images per field on the training set and 19.8/39.75 cloudless images on the validation set.
- **Expr25_only_2019_data**: Running the pipeline only on data from 2019, resulting in an MAE on the validation set of 1.562 t/ha and an R^2 of 0.364. In 2019 there was a mean of 15.23/40.5 cloudless images per field of available S2 images and available cloud-free images on the training set and 15.33/40.35 cloudless images on average on the validation set.

These experiments showed that the year of 2017 was the year with less available S2 images. This was expected as images from Sentinel-2B was first available in the summer of 2017, thus only images from Sentinel-2A was available to our model. However, comparing the number of cloud-free images with the weather conditions the individual years, we see a clear trend as 2017 had a lot of rain and cloudy weather as well as the lowest ratio between available and cloud-free image, whereas 2018 had a lot of sunny weather and the largest available to cloud-free image ratio. We also see a possible correlation between cloud-free images given the ML model and a decreased MAE of such model as the 2018 model has a much lower MAE than the 2017 model.

- **Expr26_precrop_position_features**: We added features to the model of both positional data of each field (i.e. centroid coordinate and identifier for its Danish region) and pre-crop data (i.e. DMDB pre-crop identifier and directorate pre-crop identifier). Adding the pre-crop data effected the number of training and validation samples to 2032 and 508, respectively as some fields do not had a registered pre-crop. However, this did not improve the results as the validation MAE was 1.663 and R^2 of 0.379, which was a bit worse in comparison with the best results in experiment 15.
- **Expr27_position_features**: To see of the removed samples in experiment 26 caused the change in MAE, we also tried only adding the positional features. This resulted in a validation MAE was 1.645 t/ha and R^2 of 0.395, which was the same in comparison with the best results in experiment 15.

We conclude that adding the positional data and/or the pre-crop data as features did not change the validation MAE or R^2 of our model.

- **Expr28_DMI_Gradient_Boosting**: We added features to the model of weather data from Danish Meteorological Institute (DMI). This consisted of the 8 weather measurements: mean air temperature, sum of evaporation, sum of global radiation, maximum air temperature, minimum air temperature, sum of precipitation, mean soil temperature. These are measured daily in the interval from 1 April to 15 July but are aggregated to a 7 days interval, similar to the S2 data. Adding the DMI data as features resulted in the best Gradient Boosting model of all experiments so fare, as the validation MAE decreased to 1.582 t/ha and R^2 increased to 0.425.

We conclude that adding DMI data did improved the yield prediction model by a reduction in validation MAE from 1.642 t/ha in experiment 15 to 1.582 t/ha in experiment 28 and increase in validation R^2 from 0.395 to 0.425, respectively.

- **Expr29_DMI_2019_validation_and_2017_2018_train**: As experiment 28 was the last one where we tried to improve the prediction, we performed experiment 22 again (i.e. trained the model on data from 2017 and 2018 and validated it only on data from 2019), but this time including the position, and DMI features. This resulted in a worsened validation MAE as it increased from 2.285 t/ha in experiment 22 to 2.311 t/ha in this experiment and the same for validation R^2 as it deceased from -0.144 to -0.152.

Adding DMI features in experiment 28 improved the model, but it worsened the model in experiment 29. We assume the big difference in weather over the years 2017 to 2019 (as explained earlier wrt. experiment 23 to 25) is the reason for this difference in model performance. It could be that the difference in weather data does not enable the model to find any common patterns in the data between the years.

Maize dry matter percentage experiments

In the following experiments, we changed the target such that we predicted the dry matter percentage of the yield. The same features are used as in experiment 28.

- **Expr30_Mean_dummy_baseline_drymatter_prediction**: Using the dummy baseline regression model to predict the mean dry matter percentage we had a MAE of 2.922 % and a R^2 of -0.006 on the validation set.
- **Expr31_GradientBoosting_drymatter_prediction**: The Gradient Boosting model improved the results a bit with a MAE of 2.431 % and a R^2 of 0.275 on the validation set.
- **expr32_RandomForest_drymatter_prediction**: The Random Forest model overfitted again with a MAE of 0.858 % on the training set and a MAE of 2.262 % on the validation set. Furthermore, the R^2 on the training set was 0.571 and 0.339 on the validation set.

Because of the overfitting by the Random Forest model, we choose to use the Gradient Bossting model for the following experiments, where we changed the prediction date.

- **Expr33_15_August_drymatter_prediction**: With a prediction date the 15 August, the MAE was 2.448 % and the R^2 was 0.282, thus the results had not improved significantly compared to experiment 31.
- **expr34_01_September_drymatter_prediction**: With a prediction date the 1 September, the MAE was 2.312 % and the R^2 was 0.259, thus the results had still not improved significantly compared to experiment 31.
- **Expr35_15_September_drymatter_prediction**: With a prediction date the 15 September, the MAE was 2.289 % and the R^2 was 0.267, this the results had still not improved significantly compared to experiment 31, even though we see a slight decrease in MAE.

Predicting later on the growth season reduced the MAE with only 0.1 %.

Conclusion

We conclude following for the 3 different topics of this project.

The dataset

Several cleaning steps on the dataset were done where we started from having data from 3114 fields and the cleaning procedures reduced it to 2553 fields. However, depending on the experiment, e.g. changing the prediction date or restriction of year, the number of sample differed. The final dataset, common for most experiments, consisted on a small amount of fields which resulted in only 2042 training samples and 511 validation samples. The fields were mainly located in Jylland.

The maize yield

A baseline regression model that always predicts the mean of the maize yield had a mean absolute error (MAE) of 2.168 t/ha and coefficient of determination (R^2) -0.003. The best performing model was the Gradient Boosting regression model produced in experiment 28 where the MAE was 1.582 t/ha and the R^2 was 0.425. The results of experiment 28 was achieved using the following features, i.e. input data to the model:

- Sentinel-2 images, without clouds, linear interpolated, resampled to 7 days intervals from 1 April to 15 July, and averaged over all pixels within the field polygon:\ `S2_L1C_B01`, `S2_L1C_B02`, `S2_L1C_B03`, `S2_L1C_B04`, `S2_L1C_B05`, `S2_L1C_B06`, `S2_L1C_B07`, `S2_L1C_B08`, `S2_L1C_B8A`, `S2_L1C_B09`, `S2_L1C_B10`, `S2_L1C_B11`, `S2_L1C_B12`, and `S2_L1C_NDRE`.
- Danish terrain height model (DTM) averaged over all pixels within the field polygon:\ `field_relative_mean`, `slope_pct`, `slope_angle`, and `slope_aspect`.
- Field centroid coordinate position and the Danish region the field lies within:\ `geox`, `geoy`, and `regionId`.
- Danish Meteorological Institute (DMI) daily measurements resampled to 7 days intervals from 1 April to 15 July using the following aggregation methods per measurement:\ `DMI_air_temperature`: `mean`, `std`, `min`, and `max`,\ `DMI_evaporation`: `mean`, `std`, `min`, `max`, and `sum`\ `DMI_global_radiation`: `mean`, `std`, `min`, `max`, and `sum`,\ `DMI_maximum_temperature`: `mean`, `std`, `min`, and `max`,\ `DMI_minimum_temperature`: `mean`, `std`, `min`, and `max`,\ `DMI_corrected_precipitation`: `mean`, `std`, `min`, `max`, and `sum`,\ `DMI_soil_temperature`: `mean`, `std`, `min`, and `max`.

Adding the DTM, position, and DMI features decreased the MAE with only 0.082 t/ha (from 1.664 t/ha to 1.582 t/ha) and increased the R^2 with 0.045 (from 0.38 to 0.425) which is not a significant improvement. That means that the Sentinel-2 data already represent the information found in these added features. However, the Gradient Boosting model performed better than the baseline model with a decreased MAE of 0,586 t/ha (from 2.168 t/ha to 1.582 t/ha).

The maize dry matter percentage

The maize experiments were performed using the same features as explained above for experiment 28. The baseline model that always predicts the mean dry matter percentage had a MAE of 2.946 % and a R^2 of -0.006. The best performing machine learning algorithm was again the Gradient Boosting model. Predicting dry matter percentage on 15 July had a MAE of 2.431 %, whereas predicting later on the growth season (i.e. the 15 September) resulted in the best performing model with a MAE of 2.289 and a R^2 of 0.267. The best performing model is slightly better than the baseline model with a decreased MAE of 0.657 % (from 2.946 % to 2.289 %).

General observations

Generally for the maize project, we conclude that the limited amount of data was not representable for all the maize fields in Denmark and that the machine learning models need more data and of a higher quality, before significant improvements can be achieved. The result of the Random Forest experiments (experiment 16 and 32) suggest that there is potential for achieving better results if more data was available, as the model was able to find a strong coherence between the features and targets on the training set. Furthermore, experiment 24, 25, and 26 show that the number of cloud-free images used for features effected the model performance, as more images resulted in a decreased MAE. 2018 were the year with most more cloud-free images and the model trained and validated on data from this year had a validation MAE of 1.509 t/ha which was the lowest achieved in our experiments.

Future work

We describe the following future work topics in a non prioritized list.

- Expand the amount of the maize yield and dry matter percentage data.
- Add maize varieties as features, perhaps based on unsupervised clustering, see MPSI-132 (https://jira.seges.dk/browse/MPSI-132).
- Add soil type as features.
- The prediction error could be decreased if we remove fields that do not have images in the date range June 20 to July 20. In production minimum one image has to be present in that date range in order to make prediction.
- Use Sentinel-1 radar data to when there are cloudy Sentinel-2 images.
- Use generated cloud-free Sentinel-2 data based on Sentinel-1 data, which we have worked on in the Satellite Images in Cloudy Weather (https://confluence.seges.dk/display/SIICW) project.
- Feature selection: almost all of the Sentinel-2 bands are mutually correlated thus we should only select a few as features instead of all of them.
- Use other models which captures the temporal and spatial relation, e.g.RNNs and/or CNNs.
- Impute NaN feature values, see MPSI-124 (https://jira.seges.dk/browse/MPSI-124).