



# Canadian Grain Outcomes Predictions

Dharmit Anghan  
Raman Bhandari  
Jay Dodhiawala  
Daniel Mai  
Joseffus Santos  
Dane Wanke

# Acknowledgements



## Canadian Grain Commission

- Dr. Sean Walkowiak
- Ms. Tiffany Chin

## University of Manitoba

- Dr. Rob Guderian
- Dr. Olivier Tremblay-Savard

# Grain Quality



Canadian Grain  
Commission  
Commission canadienne  
des grains

Français Help

Search grainscanada.gc.ca



Grain quality

Licensing

Research and data

Producer protection

Direction for the industry

About us



Sign up now to get your kit!

A personalized crop quality report at no cost to you!



Pause

## Initiatives



As of September 23, 2022, this service has been enhanced through changes to the Canada Grain Regulations.



Our goals for grain science and research in response to the latest trends and developments in the grain sector.



The latest information on producer compensation claims, licensee receiverships and conditional licences.



The latest harvest quality and export data for the 2022 to 2023 crop year.

## Popular topics:

- [How to request a Final Quality Determination](#)
- [Delivery eligibility declaration questions and answers](#)
- [Forms](#)
- [Official Grain Grading Guide](#)
- [Grain Statistics Weekly](#)
- [Protection for grain producers](#)
- [Harvest Sample Program](#)
- [Licensed grain companies](#)
- [Guide to Taking a Representative Sample](#)
- [How to request an inspection](#)
- [Exports of Canadian grain and wheat flour](#)
- [Product segregation codes](#)
- [Apply for a producer railway car](#)
- [Moisture content](#)

## Producer support



Harvest Sample  
Program



Insect identification



Licensed grain  
companies



Variety designation  
lists



Producer railway cars

## Preliminary degrading factor data for Canada Western Amber Durum wheat in 2022

Table information:

- Results are unlikely to represent samples harvested within 2 weeks of the date of this report
- The number of samples may not reflect actual production distributions across grades
- Samples were supplied by producers
- Downgrading can be due to 1 or more grading factors present in the same sample
- Information will be updated as more samples are received

Degrading factors present (%) in Canada Western Amber Durum wheat from all provinces as of December 21, 2022

	No. 1 CWAD	No. 2 CWAD	No. 3 CWAD	No. 4 CWAD	No. 5 CWAD
Number of samples	576	184	118	19	42
% of total	61.3	19.6	12.6	2.0	4.5
Broken (BKN)	0.0	0.5	1.7	5.3	0.0
Degermed (DGM)	0.0	0.0	0.0	0.0	2.4
Ergot (ERG)	0.0	1.1	22.9	0.0	52.4
Exoreta (EXCR)	0.0	0.0	0.0	0.0	2.4
Frost (FR)	0.0	0.0	0.8	5.3	0.0
Fusarium damage (FUS DMG)	0.0	0.0	24.6	0.0	11.9
Green (GR)	0.0	0.0	0.8	0.0	0.0
Grass green kernels (GRASS GR)	0.0	1.1	0.0	0.0	0.0
Heated kernels (HTD)	0.0	0.0	0.0	5.3	4.8
Hard vitreous kernels (HVK)	0.0	23.9	11.9	15.8	2.4
Insect Damage (I DMG)	0.0	0.5	0.0	0.0	0.0
Midge damage (MDGE DMG)	0.0	1.6	0.0	0.0	0.0
Mildew (MIL)	0.0	2.2	0.8	0.0	0.0
Matter other than cereal grains (MOTCG)	0.0	0.0	2.5	0.0	7.1
Rotted kernels (ROT KRNL)	0.0	0.0	0.0	5.3	0.0
Severe midge damaged kernels (SEVMIDGE)	0.0	9.8	15.3	0.0	2.4
Severely sprouted kernels (SEVSPTD)	0.0	1.1	0.8	0.0	0.0
Sawfly damage (SFLY DMG)	0.0	1.1	0.0	0.0	0.0
Shrunken kernels (SHR)	0.0	0.0	0.0	0.0	16.7
Smudge (SM)	0.0	2.2	0.0	0.0	0.0
Total foreign material (TFM)	0.0	2.7	0.0	10.5	0.0
Test weight (TWT)	0.0	56.0	21.2	21.1	7.1
Wheat of other classes (WOOC)	0.0	7.6	7.6	36.8	0.0

# Problem statement

## Wheat

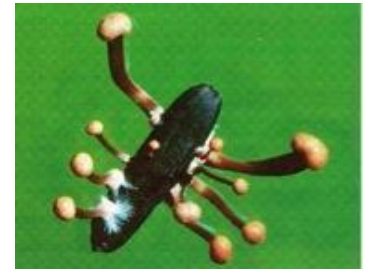
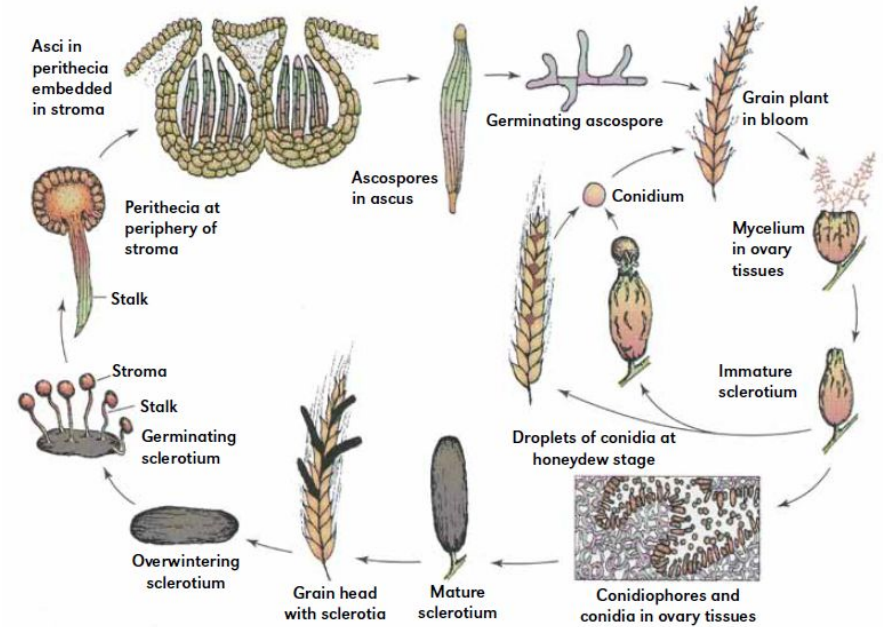
- Local economic importance:
  - **An estimated total economic impact of \$68.8 billion.**
  - **Contributing more than 370,000 Canadian jobs.**
  - **And \$27 billion in wages.**
- In 2021, Canada exported **\$8.3 billion** worth of **wheat**, making it the **fourth largest exporter** of wheat in the world.
- The main destinations for Canadian wheat include
  - China (\$831 million),
  - Japan (\$666 million),
  - Indonesia (\$634 million),
  - Peru (\$558 million),
  - Colombia (\$512 million),
  - United States (\$498 million).



# Problem statement

## Ergot

- What is it?
- Why it's important?
- Why we did modeling on it?



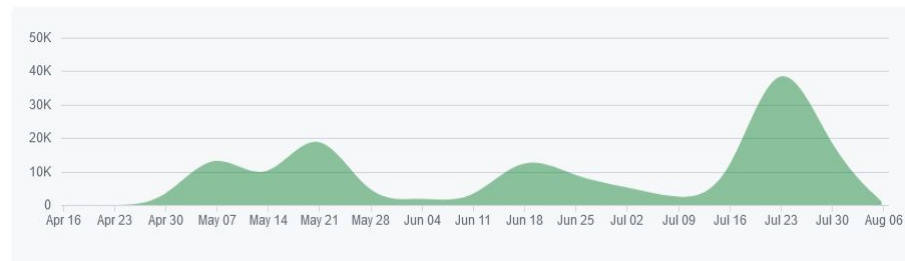
# Our Deliverables

1. 28 years of weather station, satellite and soil data
2. Docker swarm for ETL and ML
3. ETL scripts for updating
4. ETL Pipeline notebook
5. ML Model builder notebook
6. ML Model examples
7. Documentation

Apr 16, 2023 – Aug 7, 2023

Contributions: Additions ▼

Contributions to main, excluding merge commits and bot accounts



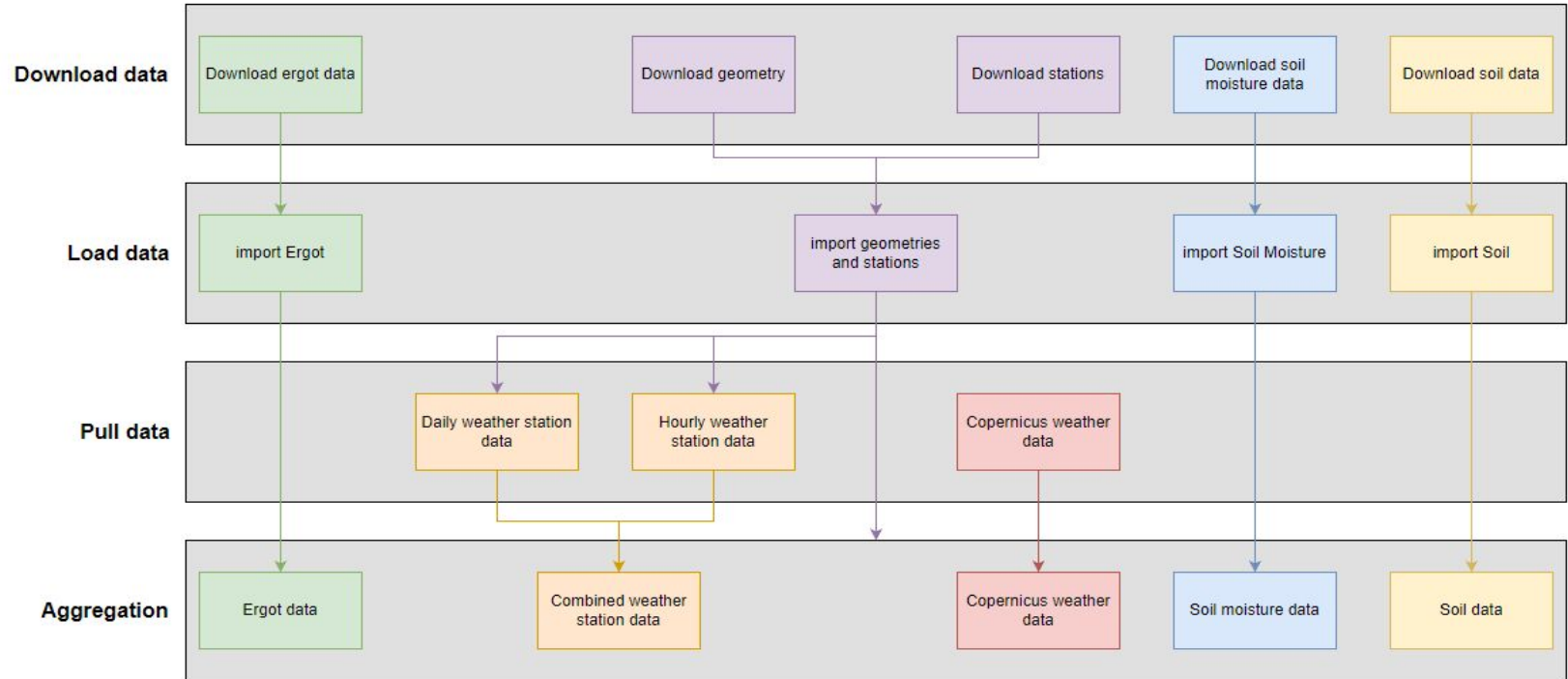
# About Dataset



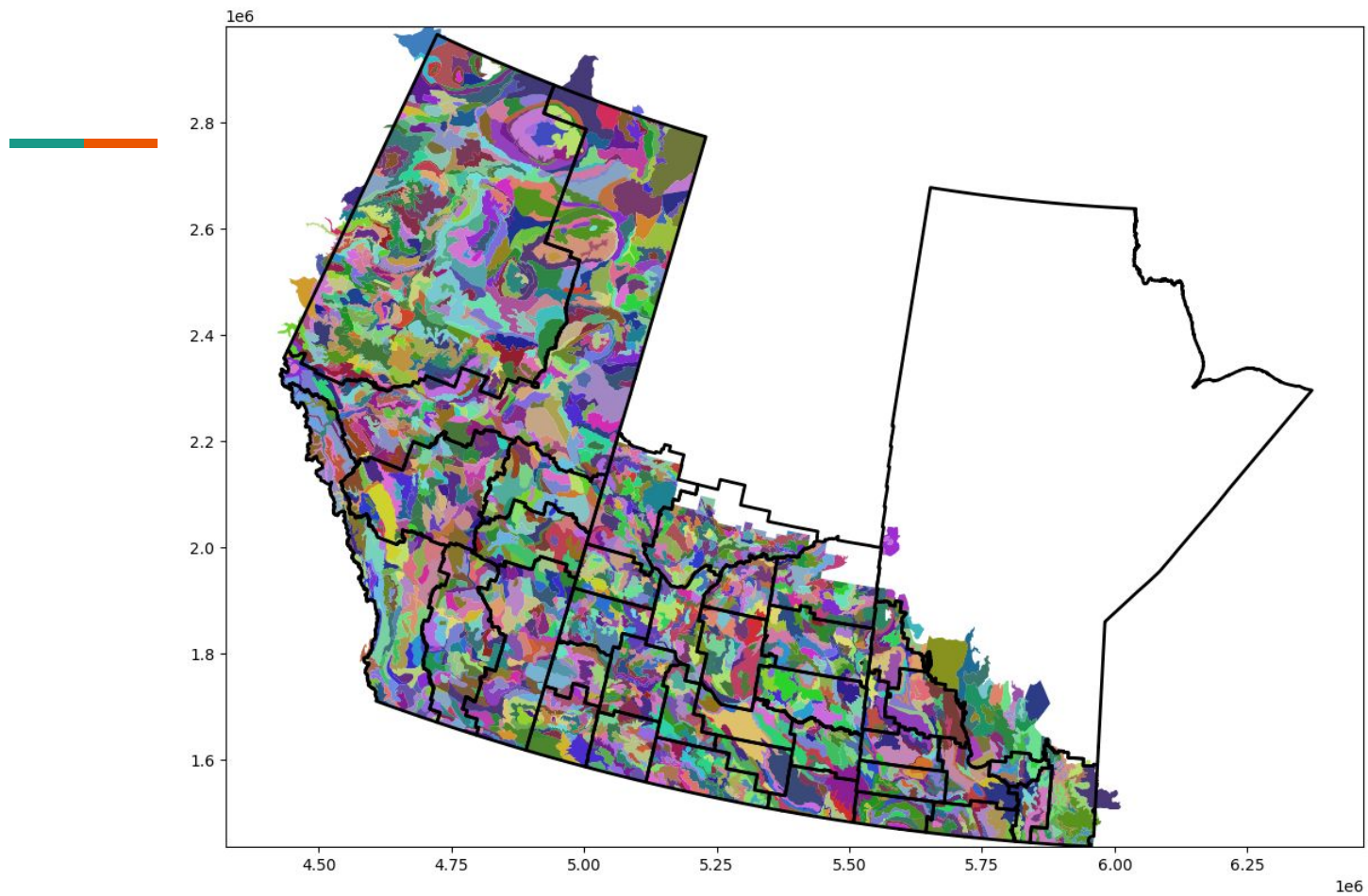
## Data Sources:

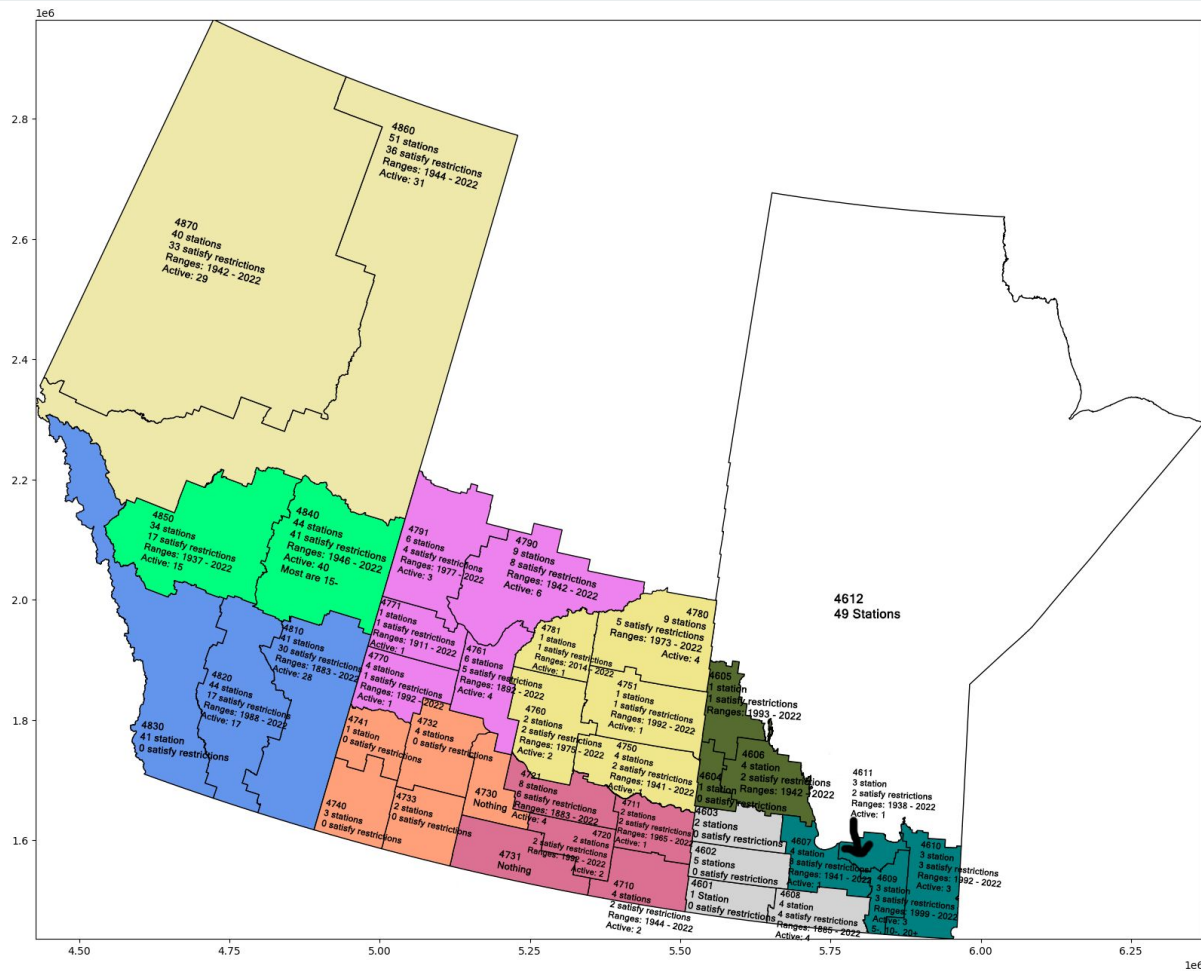
Agriculture and Agri-Food Canada	Canadian Soil data
EU Copernicus climate data store	Satellite weather data
Canadian Grain Commission	Ergot data for individual wheat samples
European space agency	Satellite soil moisture data
Environment Canada	Canadian ground weather station data
Statistics Canada	Canadian agriculture regions geographic boundaries

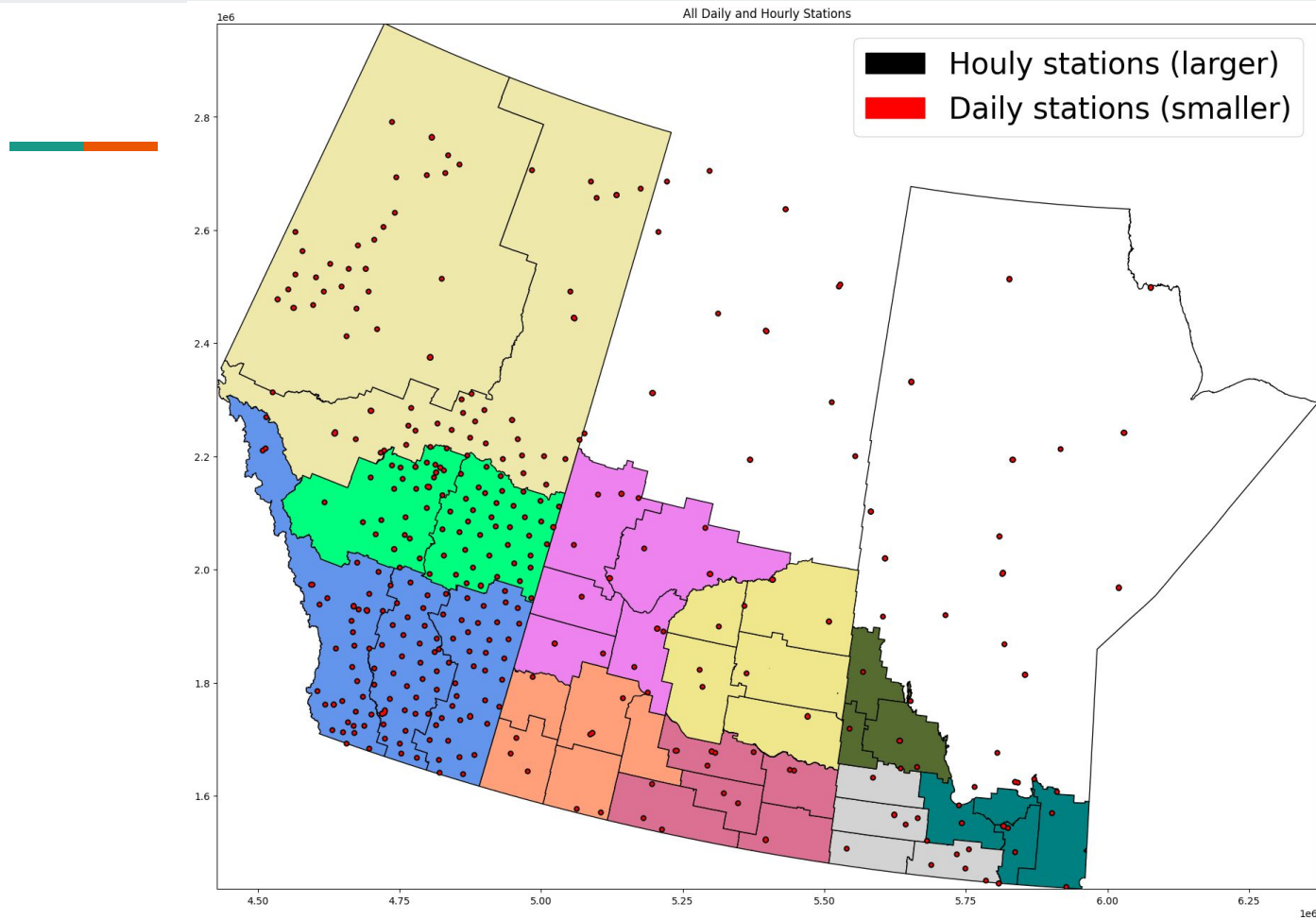
# About Dataset



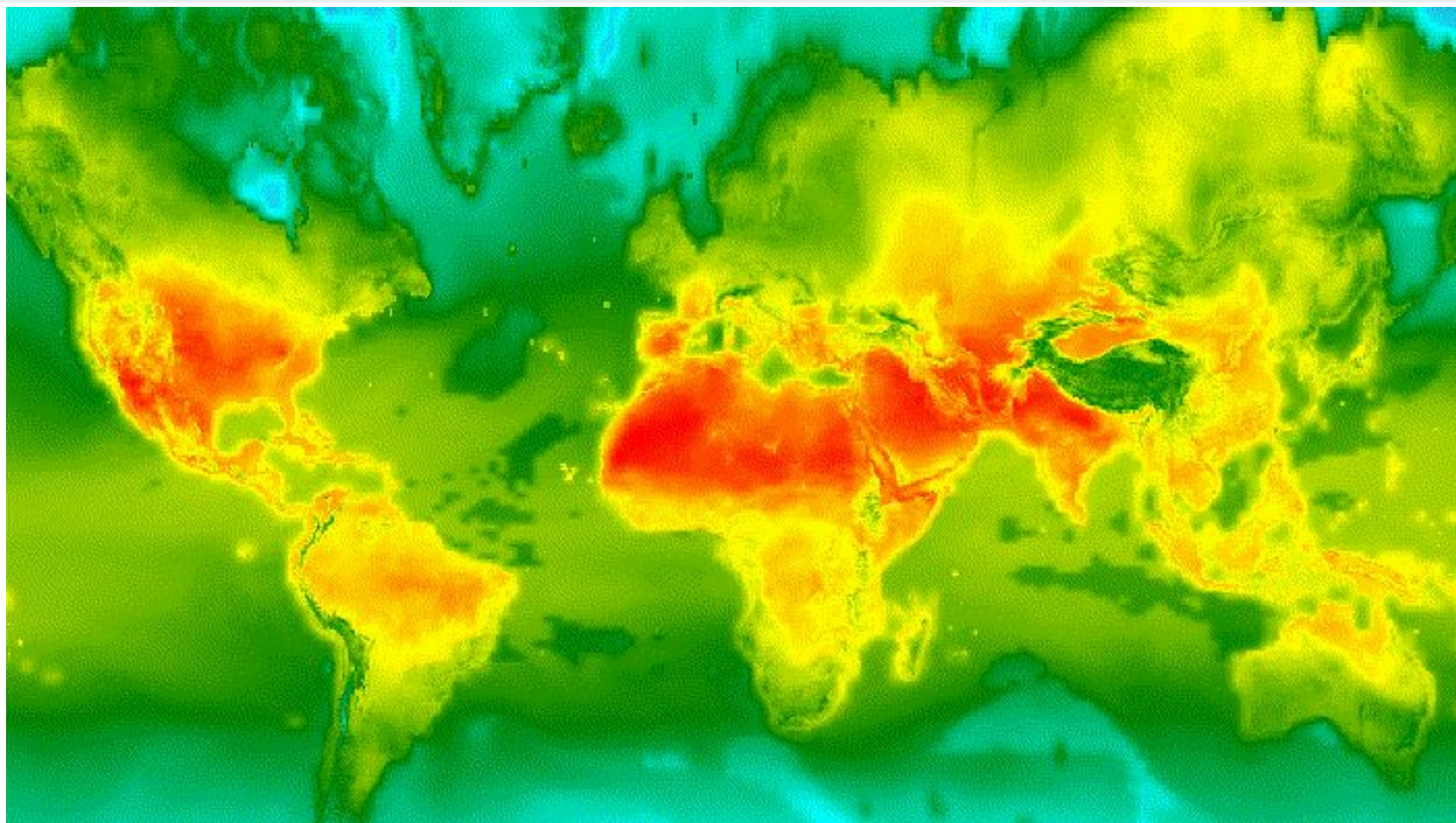


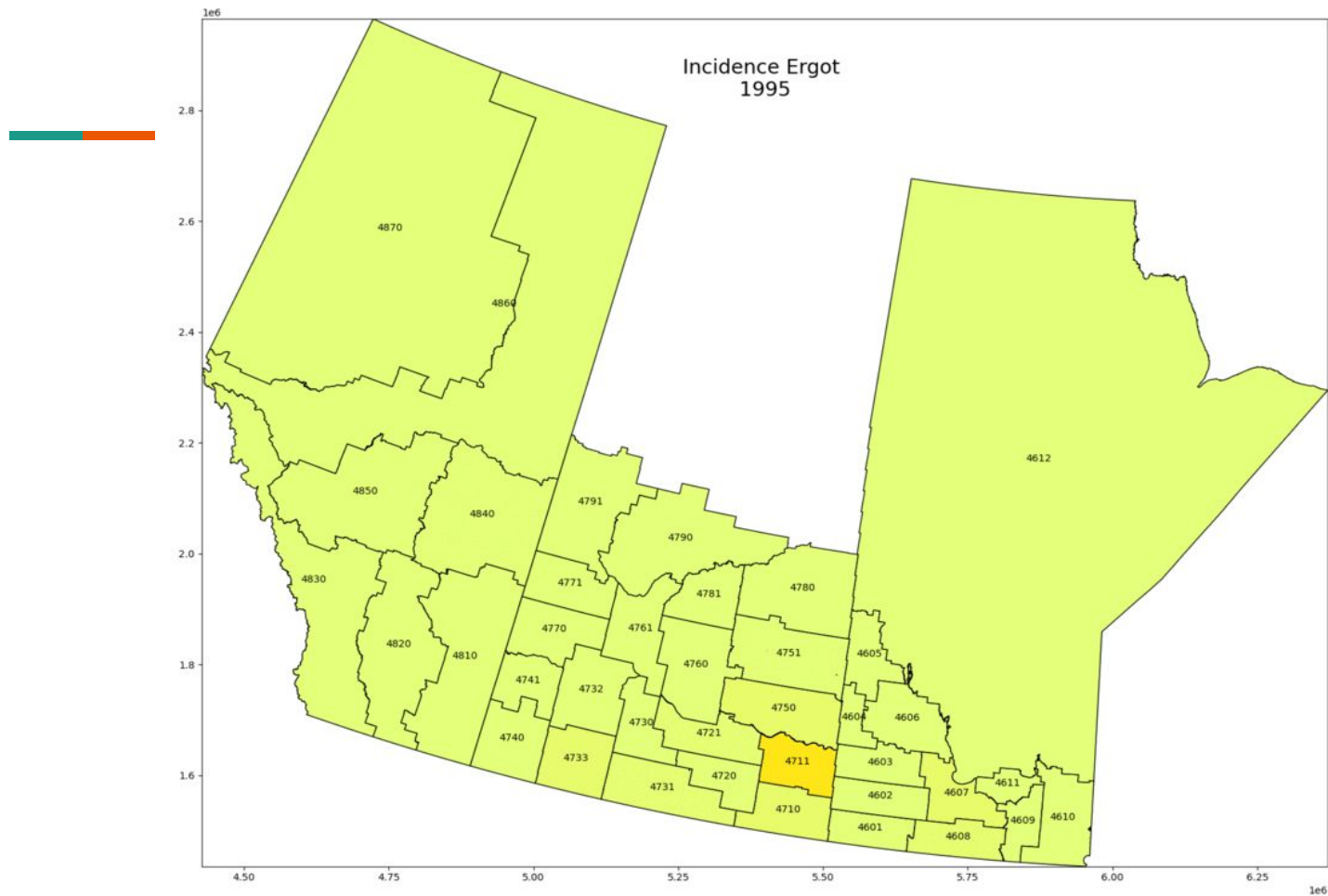


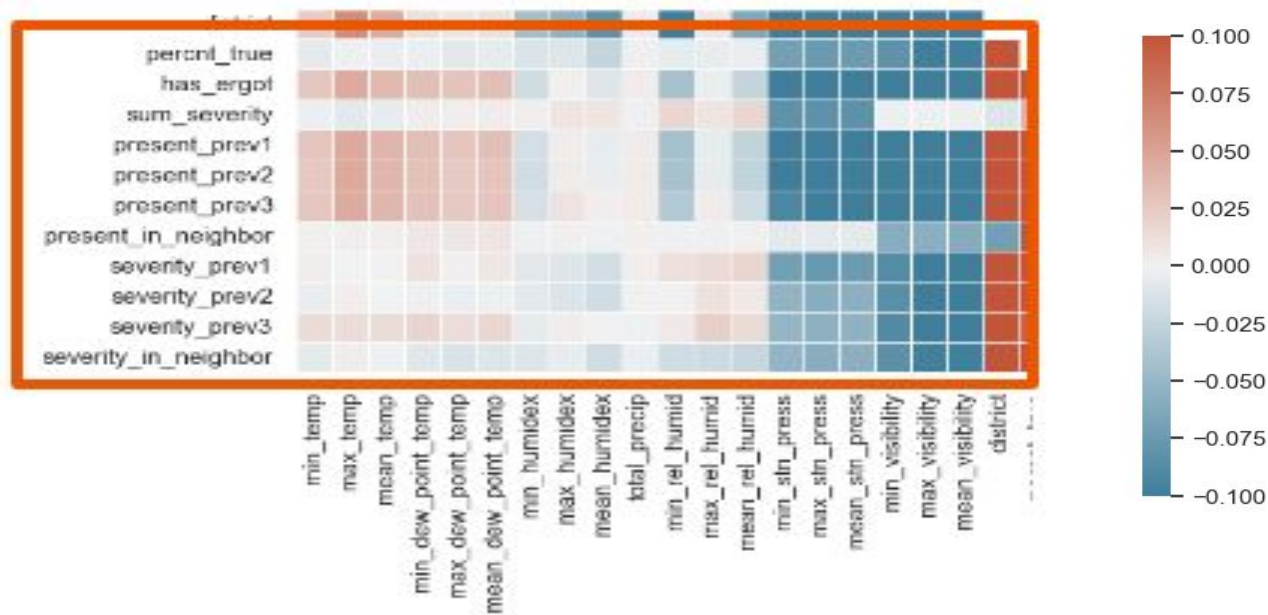








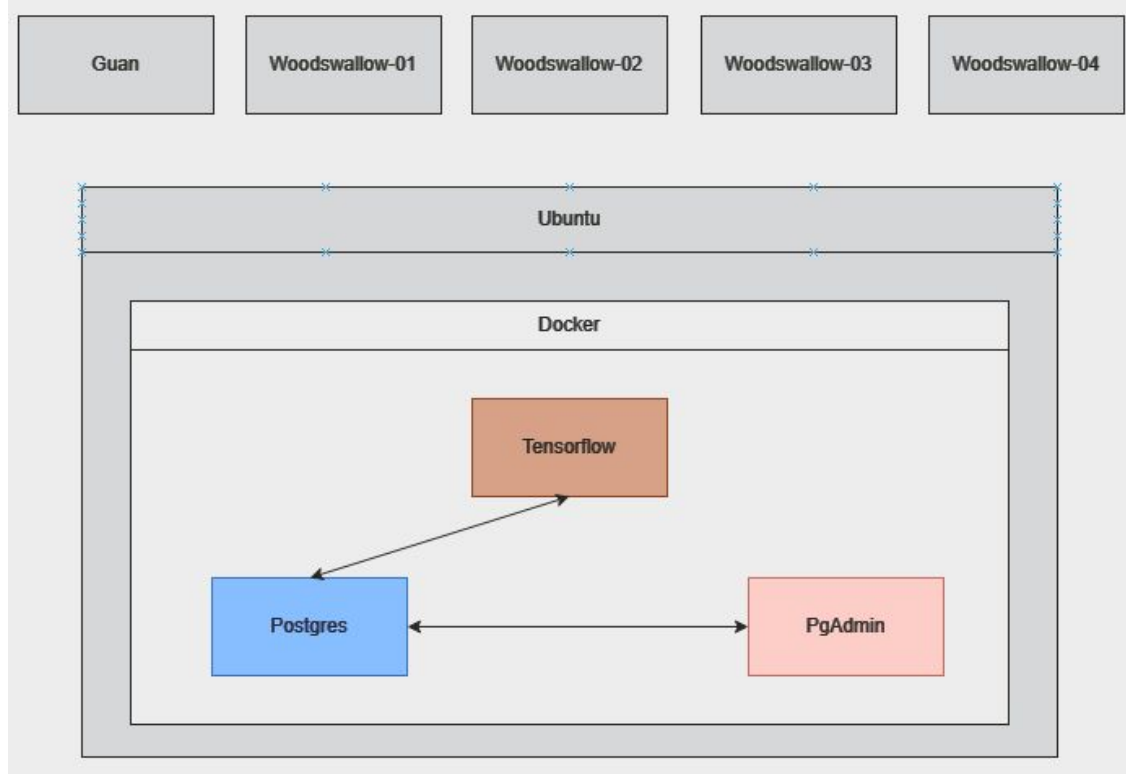




## Our tech stack



# Our tech stack





## Our tech stack

[illegible]

# Our tech stack

The screenshot displays a JupyterLab environment with two windows open. The left window, titled 'localhost:18888/lab/tree/src/Models', shows a file explorer for the '/ src / Models /' directory. It contains a table of files and folders:

Name	Last Modified
Boost	yesterday
Forest	8 days ago
KNN	yesterday
LSTM	yesterday
SVM	yesterday
classifiers.i...	2 days ago
decisionTre...	2 days ago
dimReducti...	2 days ago
evaluator.py	2 days ago
IMBClassifi...	8 days ago
ml_models...	8 days ago
models_ens...	8 days ago
regression.i...	2 days ago

The right window, titled 'localhost:18888/lab/tree/src/Models/Forest', shows a file explorer for the '/ ... / Models / Forest /' directory. It contains a table of files:

Name	Last Modified
ForestSatellite.ipynb	8 days ago
ForestSatellitePrevErg.ipynb	8 days ago
ForestStation.ipynb	8 days ago
ForestStationPrevErg.ipynb	8 days ago

The main editor area in the right window displays the 'BoostSatellite.ipynb' file. The title is 'Gradient Boosting Model on Satellite Data'. The content includes a list of dependencies and code for training a Gradient Boosting Classifier.

```
[26]: # dependencies
import pandas as pd
import sqlalchemy as sq
import sys, os
import pickle
from imblearn.combine import SMOTENMI, SMOTomek
from xgboost import XGBClassifier
from sklearn.ensemble import ( # type: ignore
    GradientBoostingClassifier,
)
from imblearn.ensemble import ( # type: ignore
    RUSBoostClassifier,
)

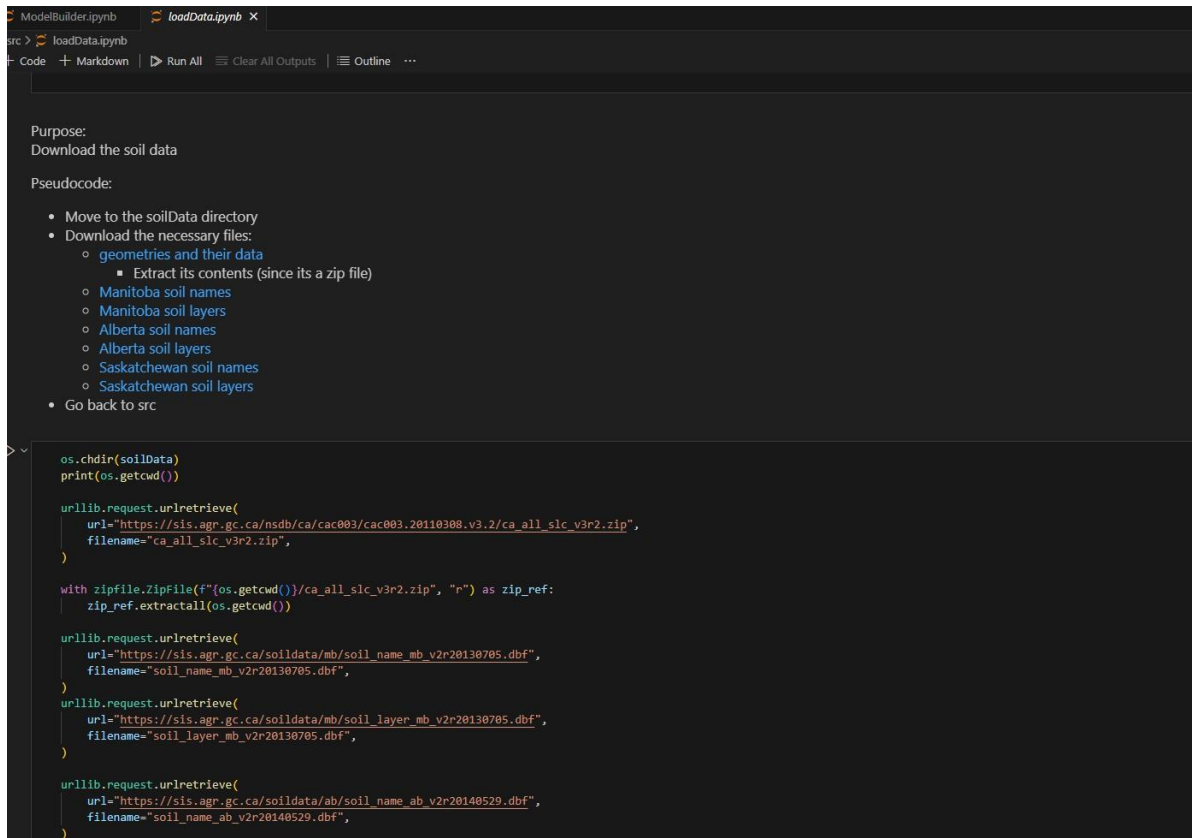
from sklearn.metrics import ( # type: ignore
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    roc_auc_score,
    classification_report,
)

sys.path.append("../..")
os.chdir("../..")
from ModelBuilderMethods import getConn, extractYears

[27]: # unlimited line output
ed.get_output("display_max_col_width" None)
```

The bottom status bar shows 'Simple' mode, '0' files, '0' cells, and a 'Launcher' icon.

# Notebook Sample



ModelBuilder.ipynb    **loadData.ipynb** X

src > loadData.ipynb

Code   +   Markdown   ▶ Run All   Clear All Outputs   Outline   ...

Purpose:  
Download the soil data

Pseudocode:

- Move to the soilData directory
- Download the necessary files:
  - geometries and their data
    - Extract its contents (since its a zip file)
  - Manitoba soil names
  - Manitoba soil layers
  - Alberta soil names
  - Alberta soil layers
  - Saskatchewan soil names
  - Saskatchewan soil layers
- Go back to src

```
os.chdir(soilData)
print(os.getcwd())

urllib.request.urlretrieve(
    url="https://sis.agr.gc.ca/nsdb/ca/cac003/cac003.20110308.v3.2/ca_all_slc_v3r2.zip",
    filename="ca_all_slc_v3r2.zip",
)

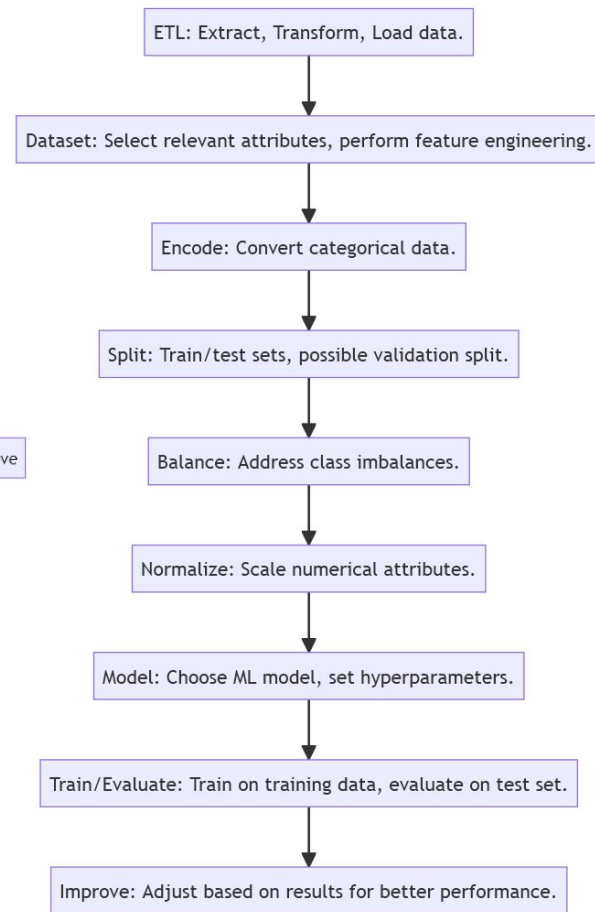
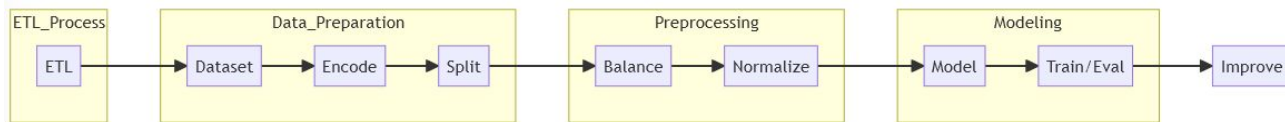
with zipfile.ZipFile(f"{os.getcwd()}/ca_all_slc_v3r2.zip", "r") as zip_ref:
    zip_ref.extractall(os.getcwd())

urllib.request.urlretrieve(
    url="https://sis.agr.gc.ca/soildata/mb/soil_name_mb_v2r20130705.dbf",
    filename="soil_name_mb_v2r20130705.dbf",
)

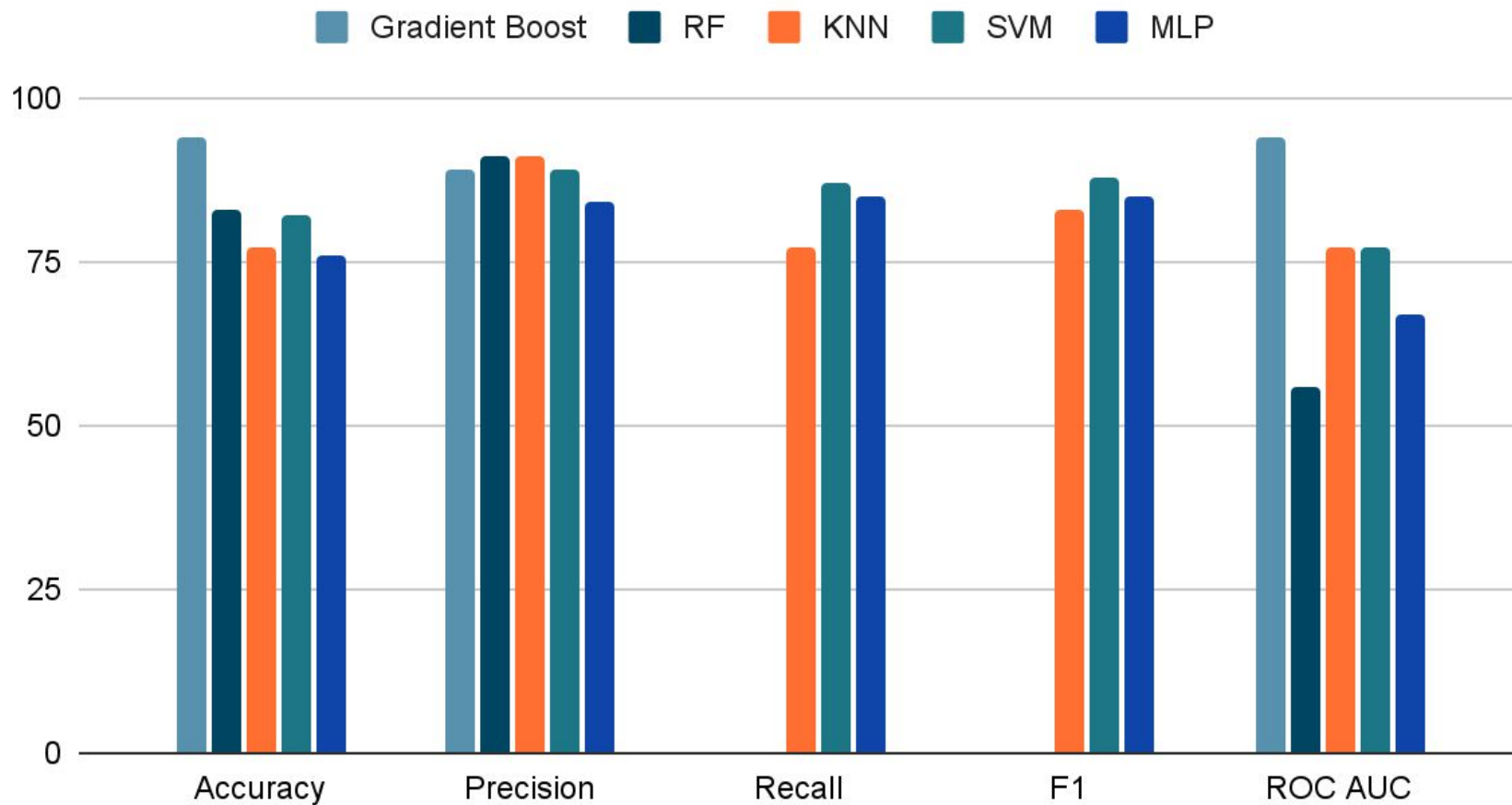
urllib.request.urlretrieve(
    url="https://sis.agr.gc.ca/soildata/mb/soil_layer_mb_v2r20130705.dbf",
    filename="soil_layer_mb_v2r20130705.dbf",
)

urllib.request.urlretrieve(
    url="https://sis.agr.gc.ca/soildata/ab/soil_name_ab_v2r20140529.dbf",
    filename="soil_name_ab_v2r20140529.dbf",
)
```

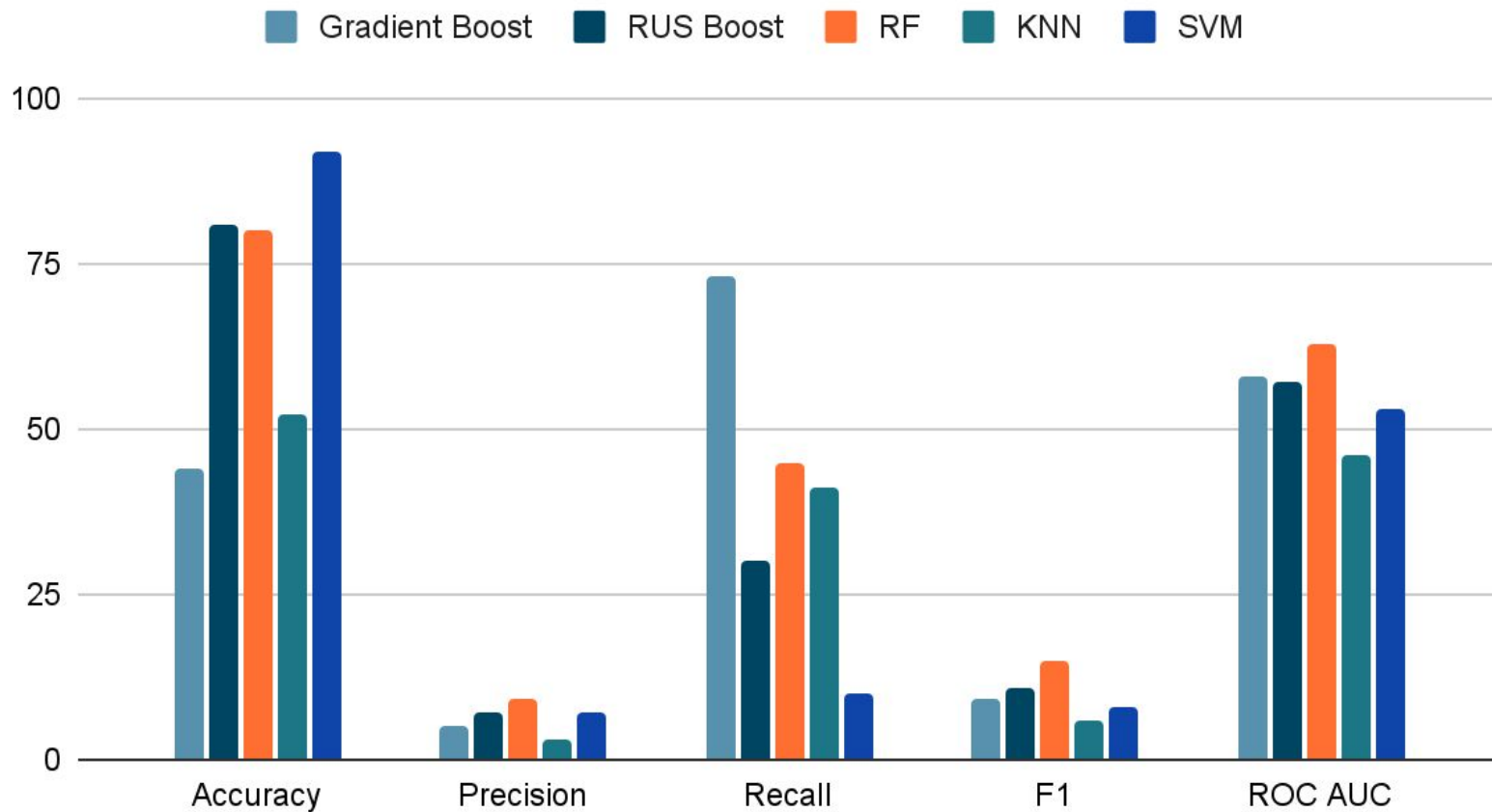
# Model Pipeline



# Early Models



## Later Models

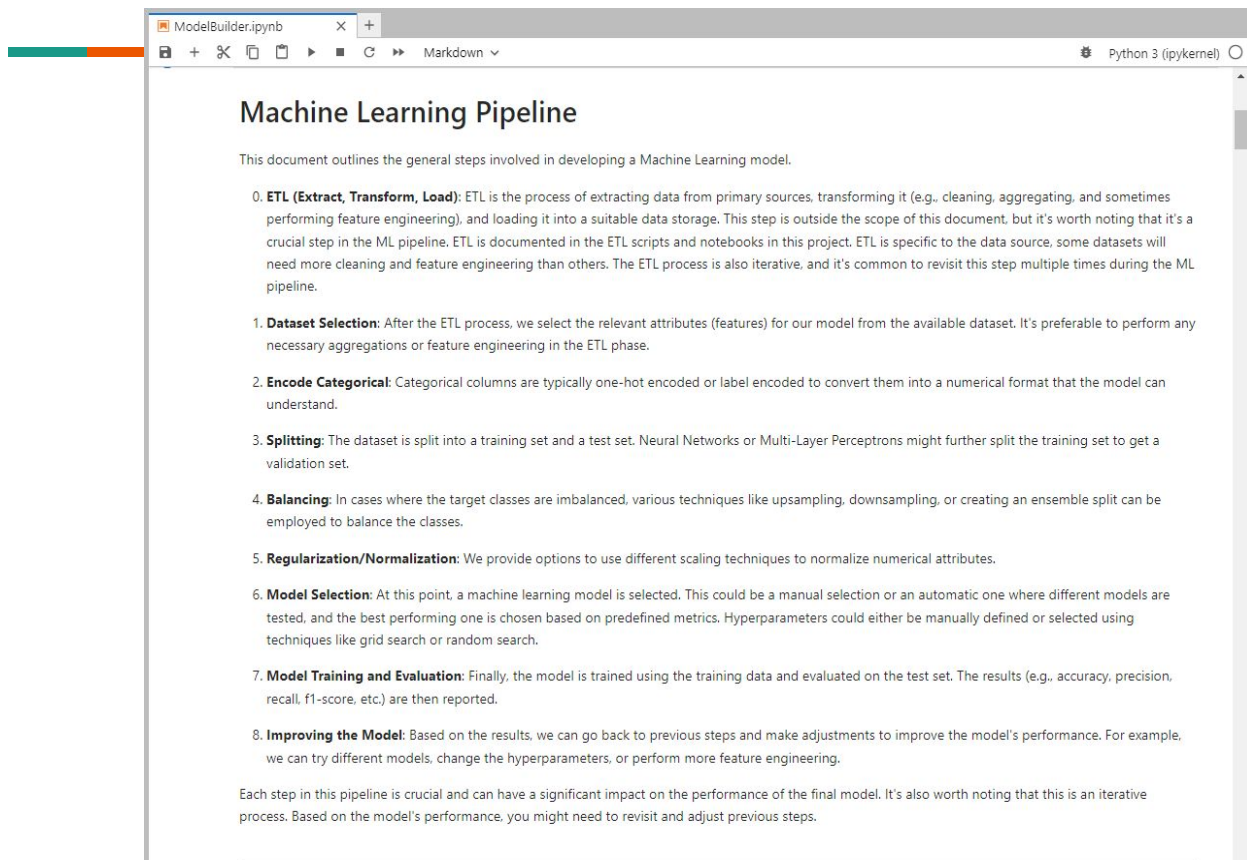




## Seemingly important data

- Humidity
- Temperature
- Soil characteristics:
  - Silt
  - Carbon
  - pH
  - Water retention
  - Wood

# ModelBuilder Notebook



ModelBuilder.ipynb Python 3 (ipykernel)

## Machine Learning Pipeline

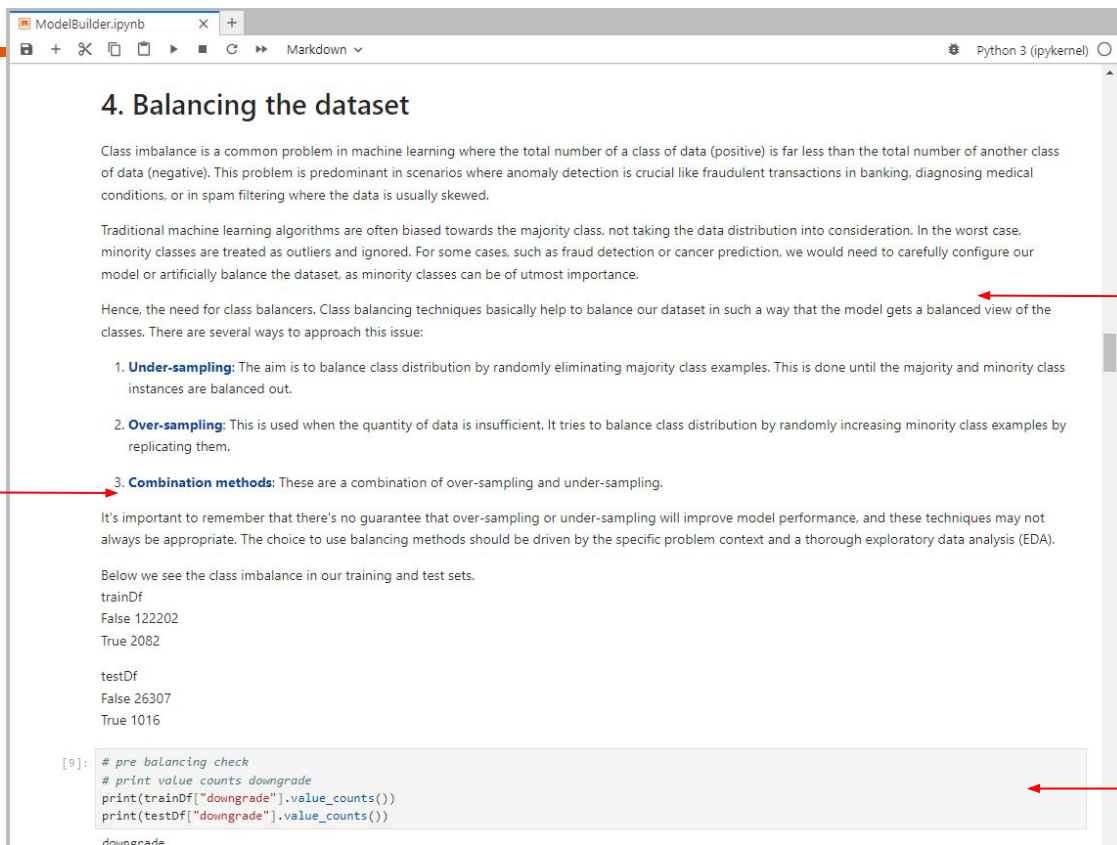
This document outlines the general steps involved in developing a Machine Learning model.

- 0. ETL (Extract, Transform, Load):** ETL is the process of extracting data from primary sources, transforming it (e.g., cleaning, aggregating, and sometimes performing feature engineering), and loading it into a suitable data storage. This step is outside the scope of this document, but it's worth noting that it's a crucial step in the ML pipeline. ETL is documented in the ETL scripts and notebooks in this project. ETL is specific to the data source, some datasets will need more cleaning and feature engineering than others. The ETL process is also iterative, and it's common to revisit this step multiple times during the ML pipeline.
- 1. Dataset Selection:** After the ETL process, we select the relevant attributes (features) for our model from the available dataset. It's preferable to perform any necessary aggregations or feature engineering in the ETL phase.
- 2. Encode Categorical:** Categorical columns are typically one-hot encoded or label encoded to convert them into a numerical format that the model can understand.
- 3. Splitting:** The dataset is split into a training set and a test set. Neural Networks or Multi-Layer Perceptrons might further split the training set to get a validation set.
- 4. Balancing:** In cases where the target classes are imbalanced, various techniques like upsampling, downsampling, or creating an ensemble split can be employed to balance the classes.
- 5. Regularization/Normalization:** We provide options to use different scaling techniques to normalize numerical attributes.
- 6. Model Selection:** At this point, a machine learning model is selected. This could be a manual selection or an automatic one where different models are tested, and the best performing one is chosen based on predefined metrics. Hyperparameters could either be manually defined or selected using techniques like grid search or random search.
- 7. Model Training and Evaluation:** Finally, the model is trained using the training data and evaluated on the test set. The results (e.g., accuracy, precision, recall, f1-score, etc.) are then reported.
- 8. Improving the Model:** Based on the results, we can go back to previous steps and make adjustments to improve the model's performance. For example, we can try different models, change the hyperparameters, or perform more feature engineering.

Each step in this pipeline is crucial and can have a significant impact on the performance of the final model. It's also worth noting that this is an iterative process. Based on the model's performance, you might need to revisit and adjust previous steps.



# ModelBuilder Notebook



The screenshot shows a Jupyter Notebook interface with a tab labeled 'ModelBuilder.ipynb'. The notebook is in 'Markdown' view. The first cell is a markdown cell with the following content:

## 4. Balancing the dataset

Class imbalance is a common problem in machine learning where the total number of a class of data (positive) is far less than the total number of another class of data (negative). This problem is predominant in scenarios where anomaly detection is crucial like fraudulent transactions in banking, diagnosing medical conditions, or in spam filtering where the data is usually skewed.

Traditional machine learning algorithms are often biased towards the majority class, not taking the data distribution into consideration. In the worst case, minority classes are treated as outliers and ignored. For some cases, such as fraud detection or cancer prediction, we would need to carefully configure our model or artificially balance the dataset, as minority classes can be of utmost importance.

Hence, the need for class balancers. Class balancing techniques basically help to balance our dataset in such a way that the model gets a balanced view of the classes. There are several ways to approach this issue:

1. **Under-sampling:** The aim is to balance class distribution by randomly eliminating majority class examples. This is done until the majority and minority class instances are balanced out.
2. **Over-sampling:** This is used when the quantity of data is insufficient. It tries to balance class distribution by randomly increasing minority class examples by replicating them.
3. **Combination methods:** These are a combination of over-sampling and under-sampling.

It's important to remember that there's no guarantee that over-sampling or under-sampling will improve model performance, and these techniques may not always be appropriate. The choice to use balancing methods should be driven by the specific problem context and a thorough exploratory data analysis (EDA).

Below we see the class imbalance in our training and test sets.

```
trainDf
False 122202
True 2082

testDf
False 26307
True 1016
```

[9]:

```
# pre balancing check
# print value counts downgrade
print(trainDf["downgrade"].value_counts())
print(testDf["downgrade"].value_counts())

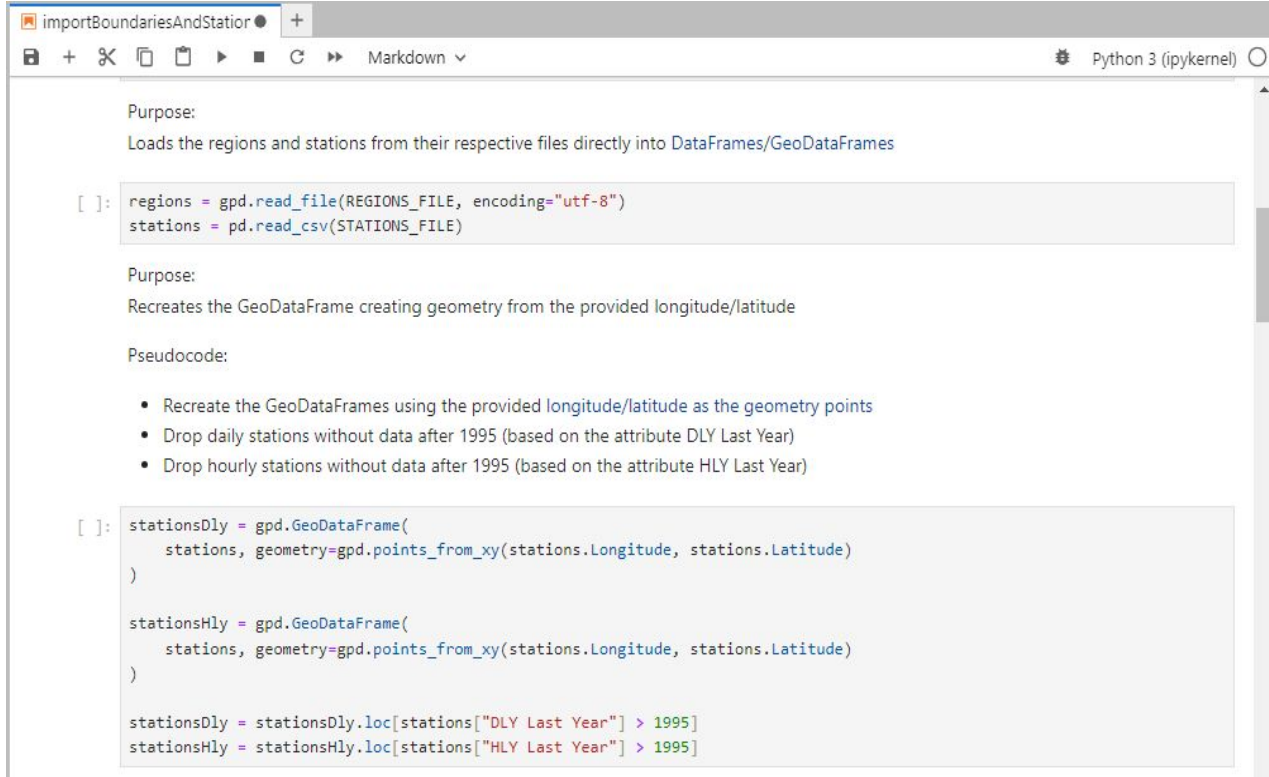
downgrade
```

Link to docs

Markdown - explanation

Code

# Notebook Sample



The screenshot shows a Jupyter Notebook window with a single tab titled 'importBoundariesAndStation'. The interface includes a toolbar with icons for saving, adding, deleting, and running code, as well as a dropdown menu set to 'Markdown'. The notebook content is as follows:

Purpose:  
Loads the regions and stations from their respective files directly into [DataFrames/GeoDataFrames](#)

```
[ ]: regions = gpd.read_file(REGIONS_FILE, encoding="utf-8")
stations = pd.read_csv(STATIONS_FILE)
```

Purpose:  
Recreates the GeoDataFrame creating geometry from the provided longitude/latitude

Pseudocode:

- Recreate the GeoDataFrames using the provided [longitude/latitude](#) as the geometry points
- Drop daily stations without data after 1995 (based on the attribute DLY Last Year)
- Drop hourly stations without data after 1995 (based on the attribute HLY Last Year)

```
[ ]: stationsDly = gpd.GeoDataFrame(
    stations, geometry=gpd.points_from_xy(stations.Longitude, stations.Latitude)
)

stationsHly = gpd.GeoDataFrame(
    stations, geometry=gpd.points_from_xy(stations.Longitude, stations.Latitude)
)

stationsDly = stationsDly.loc[stations["DLY Last Year"] > 1995]
stationsHly = stationsHly.loc[stations["HLY Last Year"] > 1995]
```

# Repo examples

The screenshot displays the JupyterLab interface. The top toolbar includes navigation icons and the address bar shows the current path: `localhost:18888/lab/tree/src/Models/Forest`. The left sidebar contains a file explorer with a search bar and a list of files and folders. The main area shows a code editor with a Jupyter Notebook file named `BoostSatellite.ipynb`.

**File Explorer (Left Sidebar):**

Name	Last Modified
Boost	yesterday
Forest	8 days ago
KNN	yesterday
LSTM	yesterday
SVM	yesterday
classifiers.i...	2 days ago
decisionTre...	2 days ago
dimReducti...	2 days ago
evaluator.py	2 days ago
IMBClassifi...	8 days ago
ml_models....	8 days ago
models_ens...	8 days ago
regression.i...	2 days ago

**Code Editor (Main Area):**

The code editor displays the content of `BoostSatellite.ipynb`. The title is "Gradient Boosting Model on Satellite Data". The code includes imports for `pandas`, `sqlalchemy`, `sys`, `os`, `pickle`, `imblearn.combine`, `SGDTEENN`, `SGDTEomek`, `xgboost`, `XGBClassifier`, `sklearn.ensemble`, `GradientBoostingClassifier`, `imblearn.ensemble`, `RUSBoostClassifier`, `sklearn.metrics`, `accuracy_score`, `precision_score`, `recall_score`, `f1_score`, `roc_auc_score`, `classification_report`, and `ModelBuilderMethods`.

```
[26]: # dependencies
import pandas as pd
import sqlalchemy as sq
import sys, os
import pickle
from imblearn.combine import SGDTEENN, SGDTEomek
from xgboost import XGBClassifier
from sklearn.ensemble import ( # type: ignore
    GradientBoostingClassifier,
)
from imblearn.ensemble import ( # type: ignore
    RUSBoostClassifier,
)


from sklearn.metrics import ( # type: ignore
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    roc_auc_score,
    classification_report,
)

sys.path.append("../..")
os.chdir("../..")
from ModelBuilderMethods import getConn, extractYears
```

[27]: # unlimited line output

The bottom status bar shows "Simple" mode, a progress bar, and a "Launcher" button.

# Future work

- 
- Importance of domain knowledge
  - Model improvements
  - Feature selection
  - Hyperparameter tuning
  - Exploring other models
    - RERF
    - MARS

# Questions

