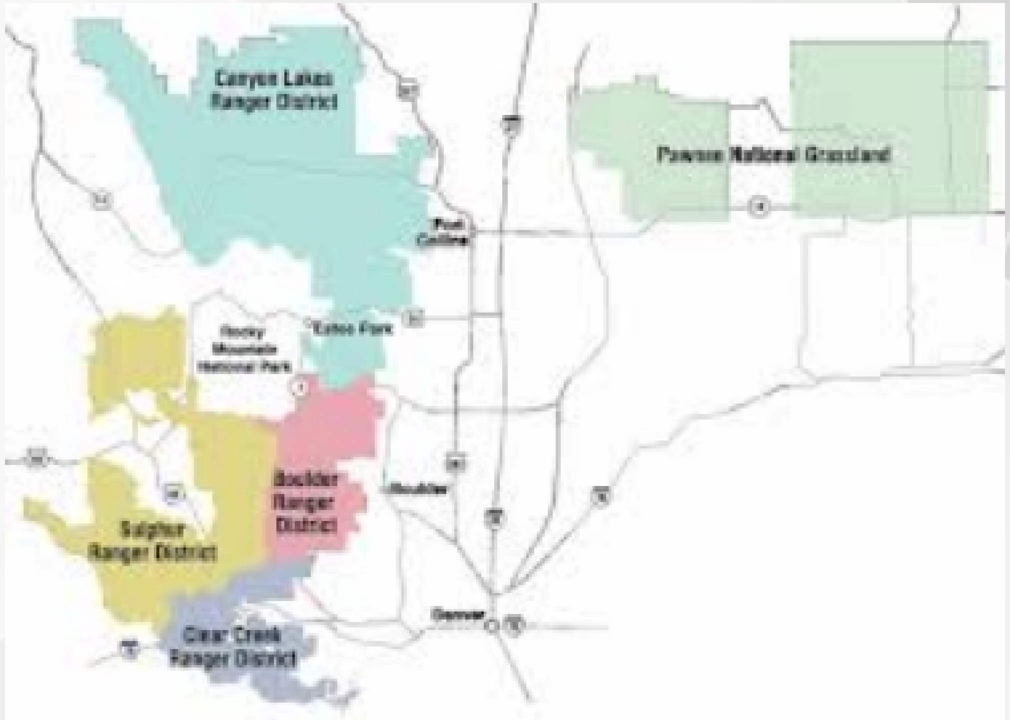
Prediction of Forest Cover Type in Roosevelt National Forest Colorado, US



Lori Newhouse
March 2021

Outline

- project description
- data overview and exploration
- initial model investigation (`sklearn` and trees)
- model development process
- modeling results
- model tuning and finalization
- model deployment

Project Description

- classification problem in environmental science
  - ✓ develop resource management strategies in ecological area
  - ✓ need forest type variation and distribution
  - ✓ use cartographic variables to predict for new area based on known area
- dataset is for Roosevelt National Forest, Colorado, USA
- http://archive.ics.uci.edu/ml/datasets/Covertype
- Blackard, Jock A., Dean, Denis J., Anderson, Charles W. (1998). Covertype Data Set : University of California--Irvine Machine Learning Repository.

Features

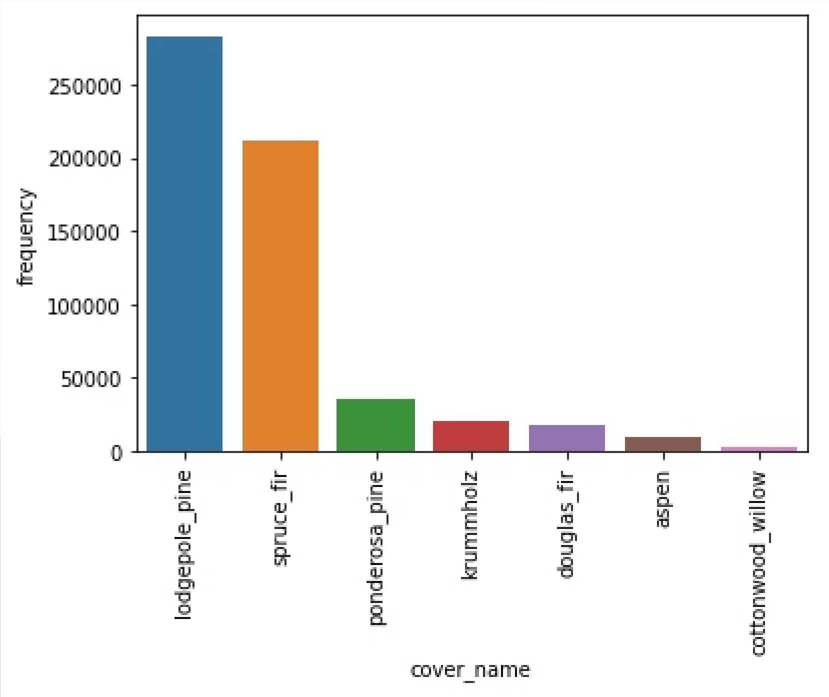
12 features

- 10 numerical
- 1st categorical feature (non-ordinal): 4 categories
- 2nd categorical feature (non-ordinal): 40 categories

original data

- csv file, gzip format
- 581,012 observations (each is a 30 meter by 30 meter area)
- clean, no missing values
- both categorical features only given as 1-hot encoded columns
- additional information provided on meaning of each encoded value
- pre-processing reversed 1-hot encoding to make columns with category value

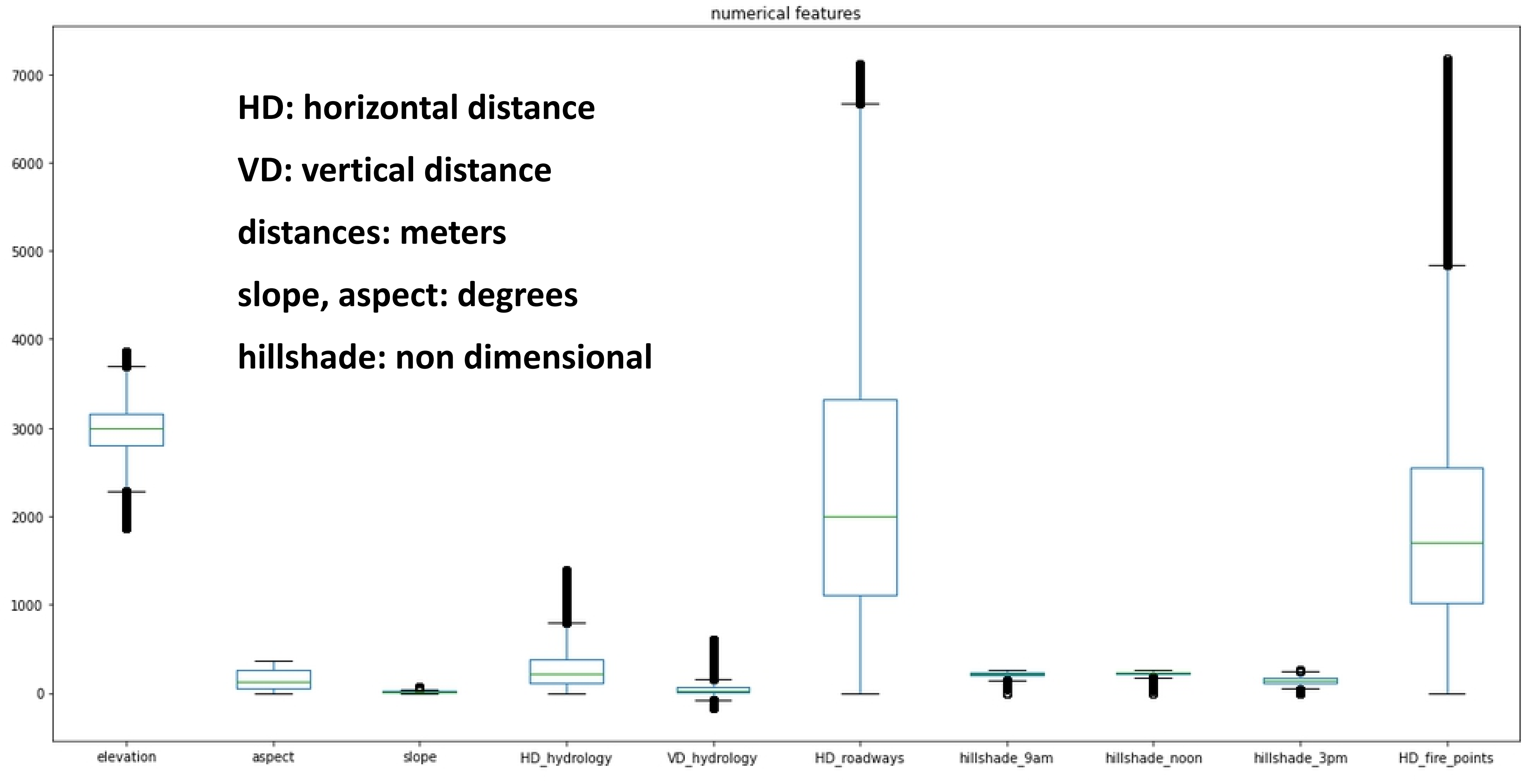
Cover Type (Target Values)



cover_name	count	fraction
lodgepole_pine	283301	0.487599
spruce_fir	211840	0.364605
ponderosa_pine	35754	0.061537
krummholz	20510	0.035300
douglas_fir	17367	0.029891
aspen	9493	0.016339
cottonwood_willow	2747	0.004728

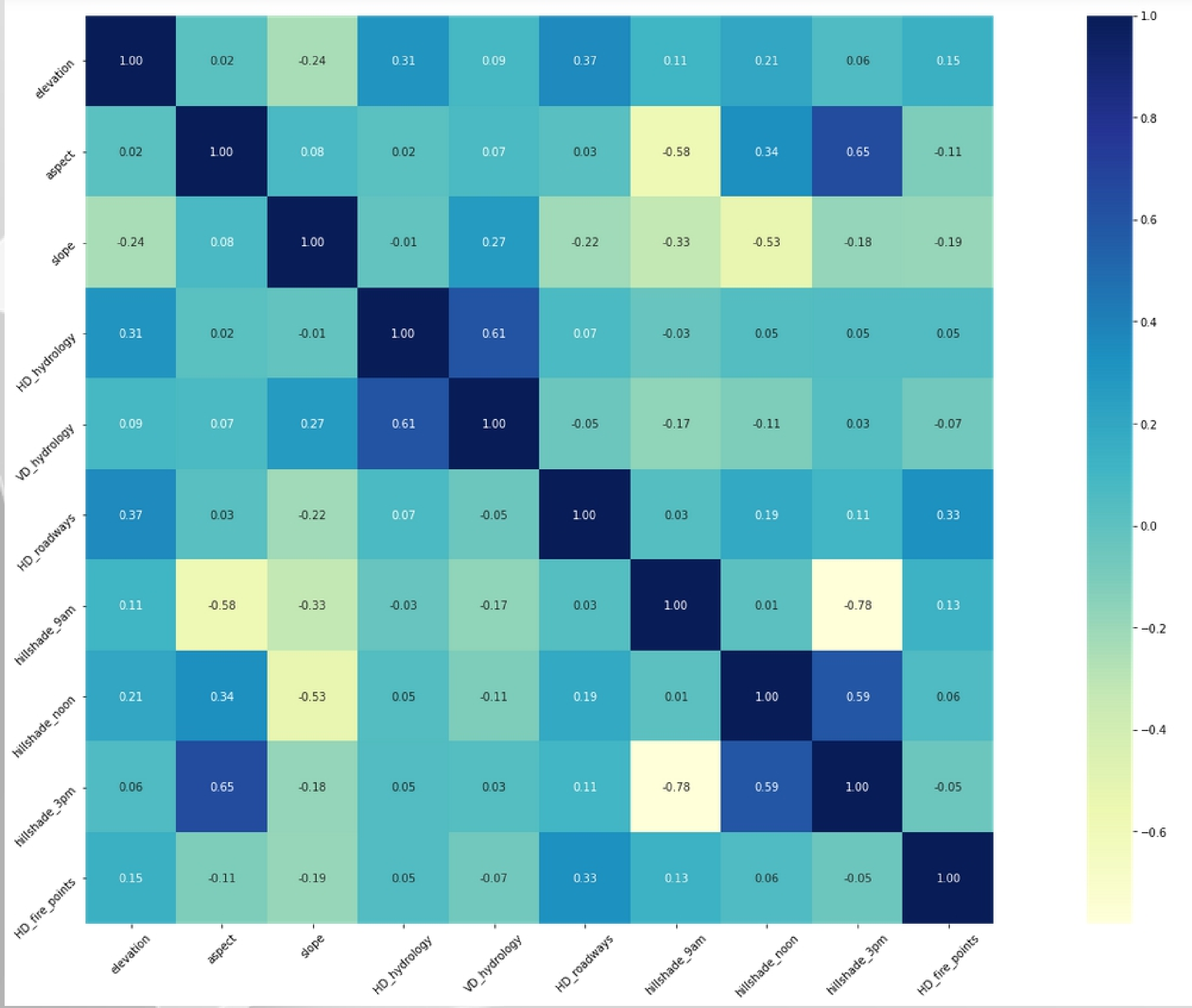
- 7 values for cover type
- significant class imbalance
- 85% of observations in 2 largest classes
- 5 smallest classes present at 0.5 to 6% each

Data Exploration: Numerical Features (Box)



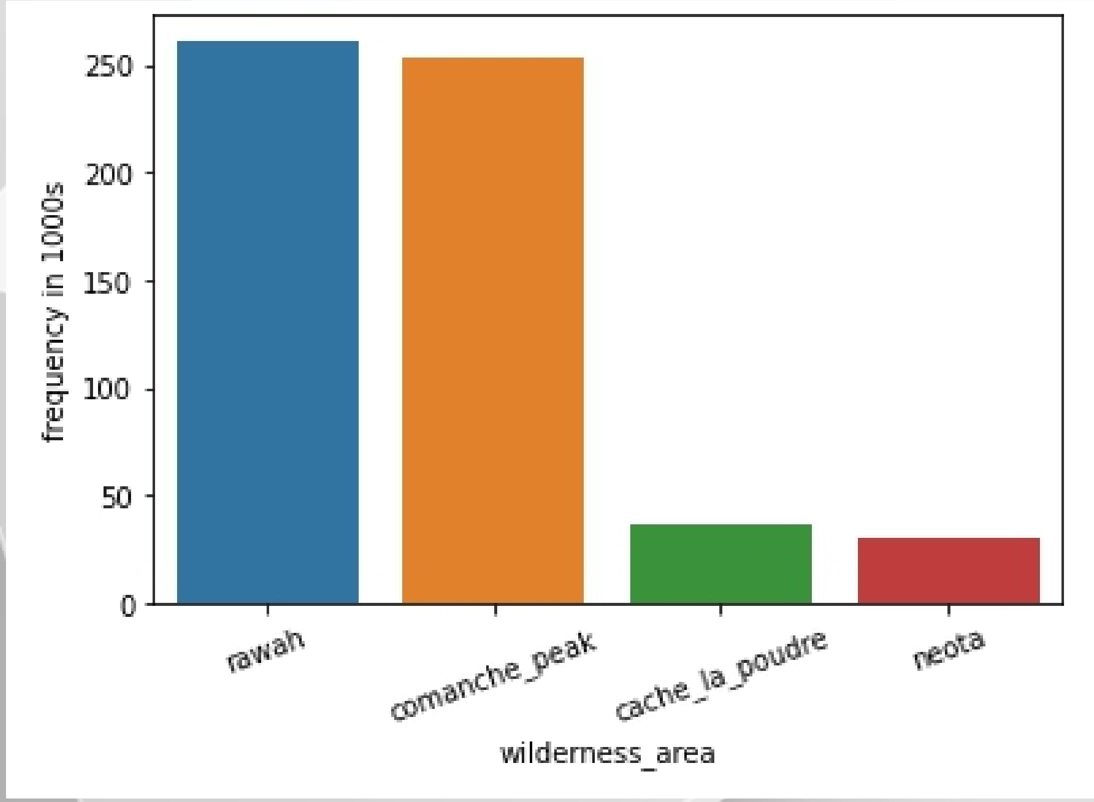
- different magnitudes
- not normally distributed

Data Exploration: Numerical Features (Correlation)



- no strong correlations

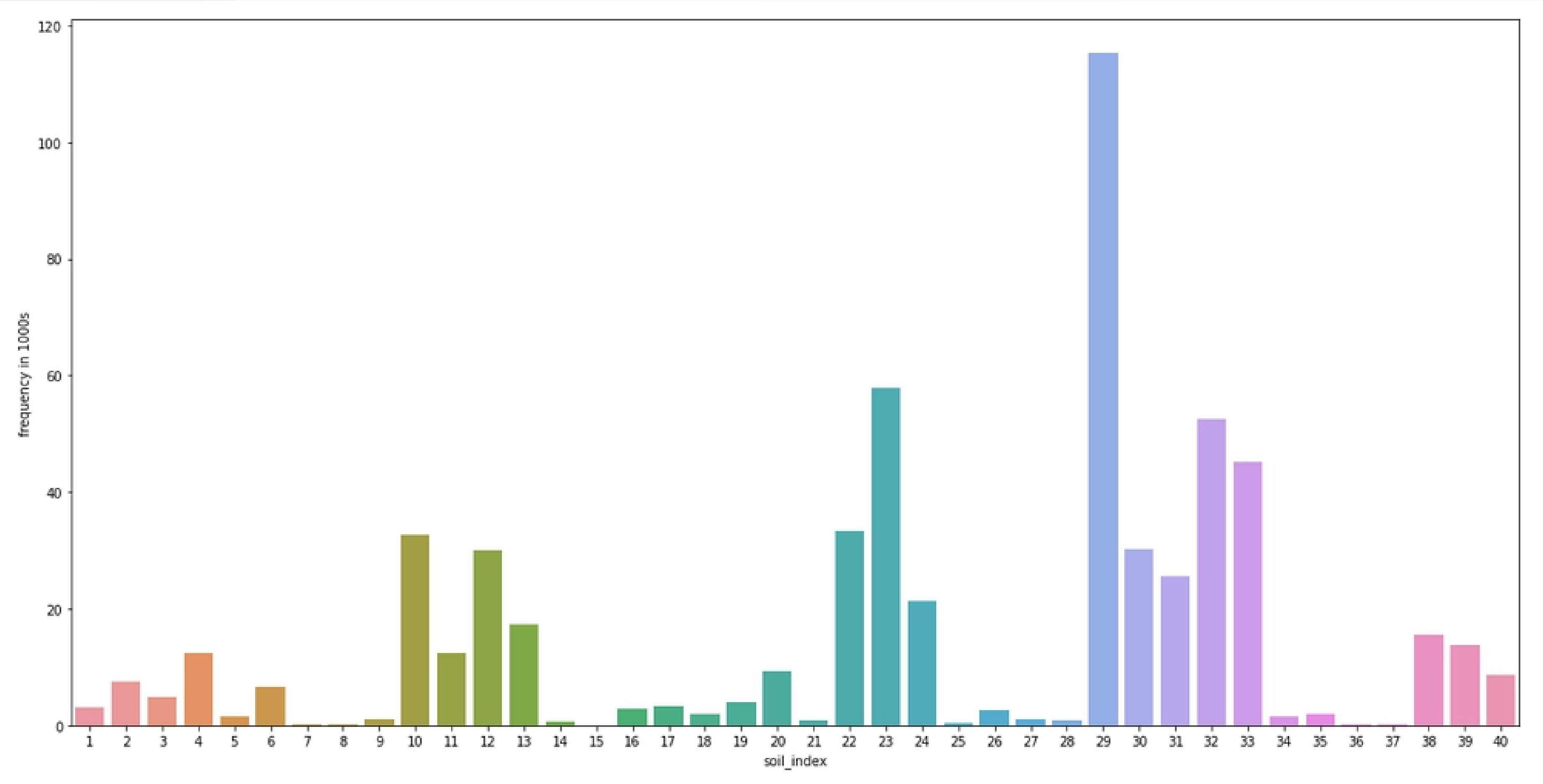
Data Exploration: Categorical Feature Wilderness Area



4 wilderness areas within

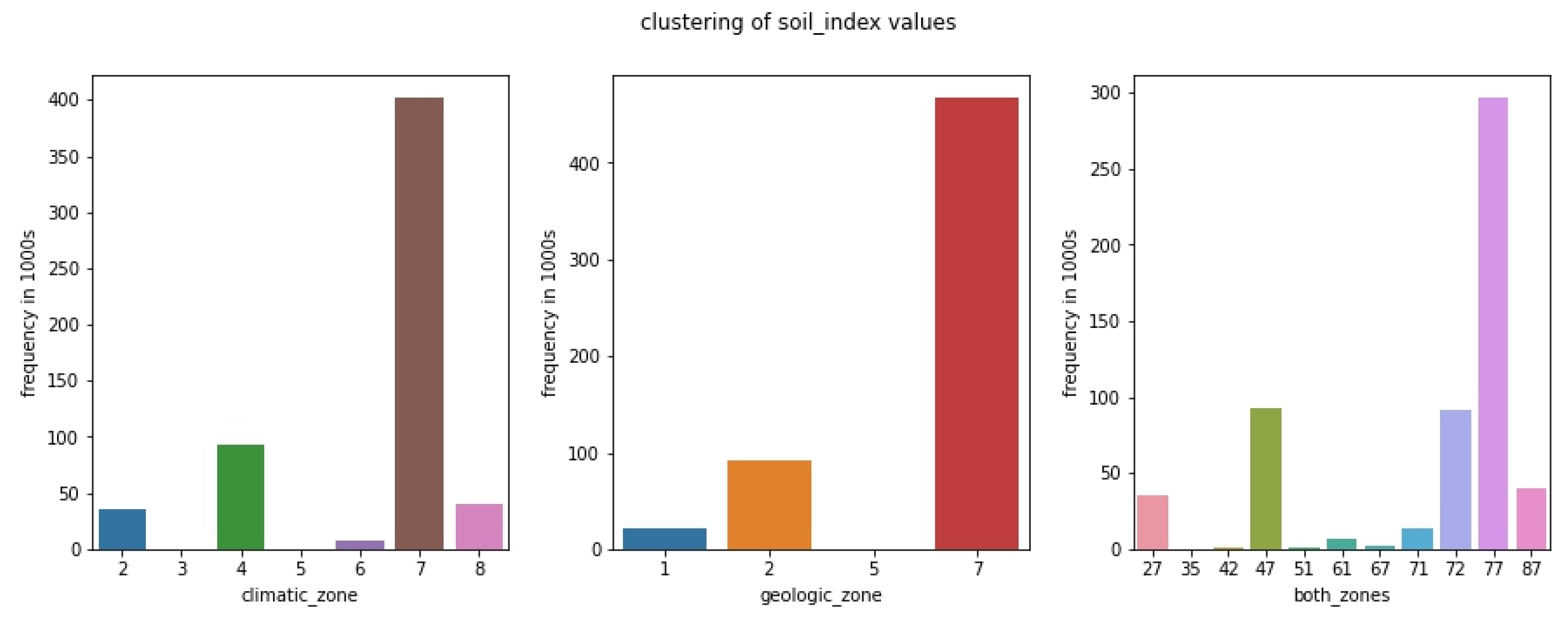
Roosevelt National Forest

Data Exploration: Categorical Feature Soil Type



- each category represents a US Forestry Service Ecological Landtype Unit (ELU)
- ELU code contains a digit for climatic zone and a digit for geologic zone

Data Exploration: Soil Type Clusters



•each zone value on horizontal axis IS IN dataset

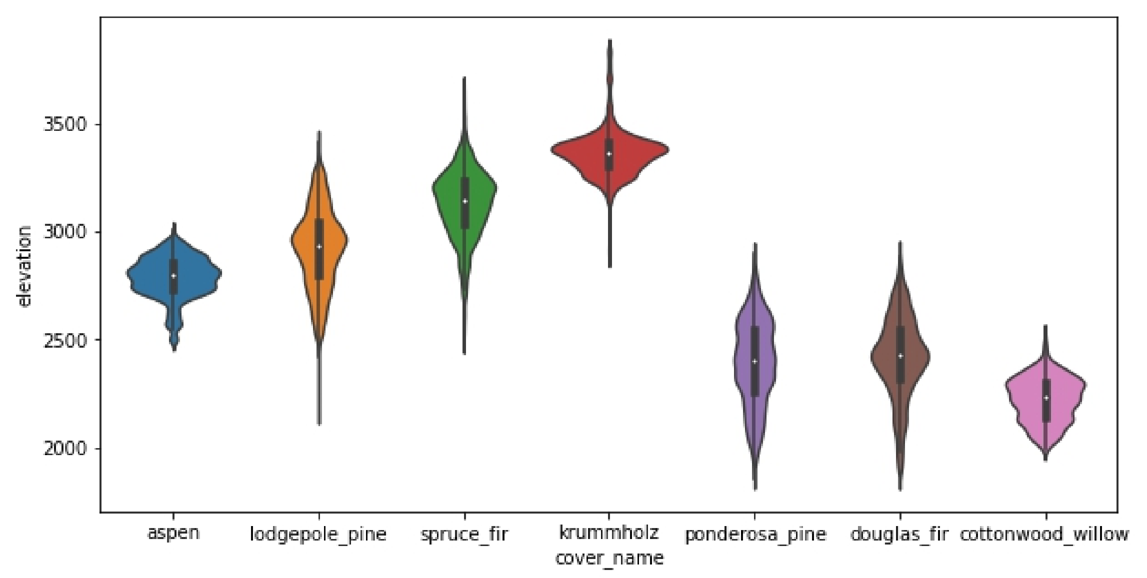
•bar may be too small to see

•soil type, climatic zone, and geologic zone descriptive information is included with dataset

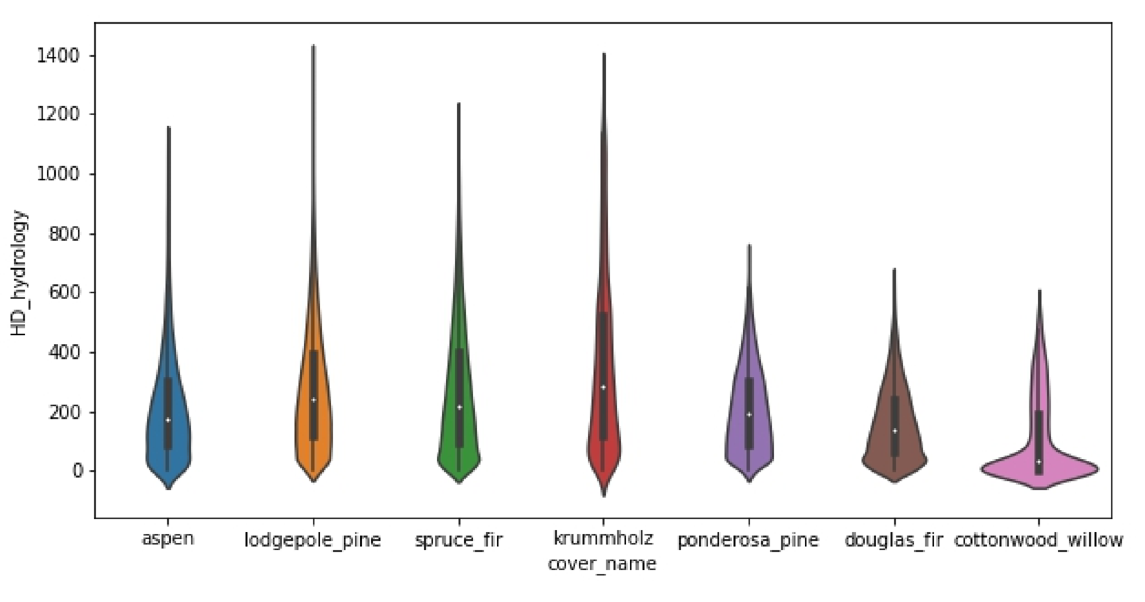
•pre-process soil type for 3 clustering techniques

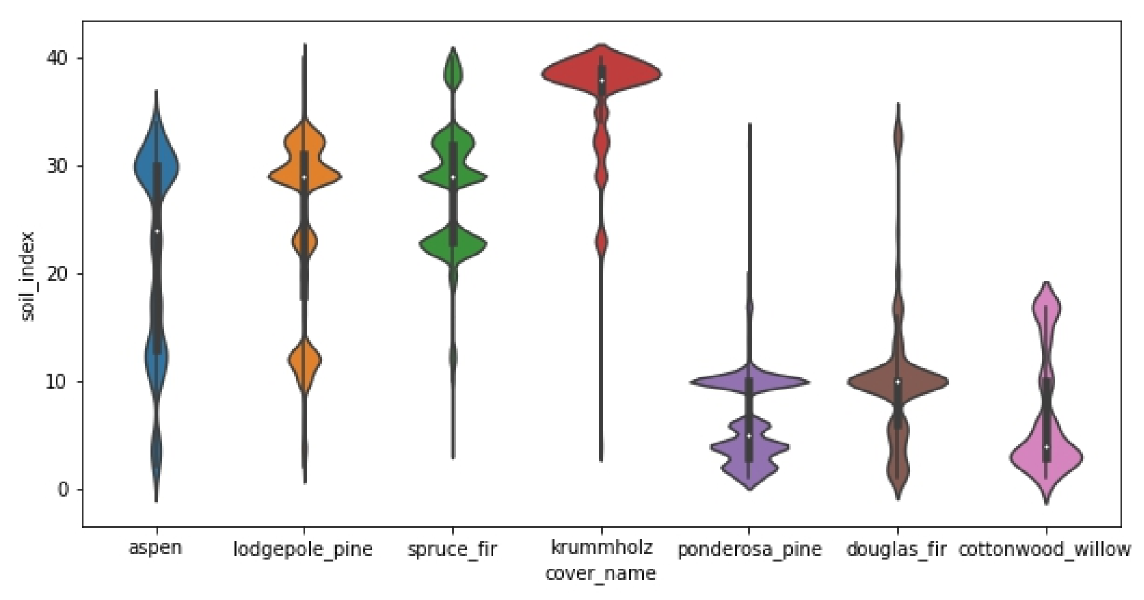
- climatic zone (7 values)
- geologic zone (4 values)
- unique combinations of climatic and geologic (11 values)

Data Exploration: Feature Distribution by Target



- distribution of elevation and soil index varies by target class
- distribution of HD_hydrology does not





Initial Model Investigation

- `fit` **model, default hyper-parameter values**
- **data pre-processing**
  - ✓ **min-max scaling of numerical values**
  - ✓ **one-hot encoding of soil type (40 values)**
  - ✓ **one-hot encoding of wilderness area**
- `sklearn` **models**
  - ✓ logistic regression
  - ✓ SVM
  - ✓ gaussian naive bayes
  - ✓ multinomial naive bayes
  - ✓ complement naive bayes
- **tree models**
  - ✓ decision tree
  - ✓ random forest
  - ✓ gradient boosting (GB)
  - ✓ XGBoost
  - ✓ LightGBM
  - ✓ CatBoost

Initial Model Investigation (con’t)

- overall model metrics
  - ✓ accuracy
  - ✓ average and weighted average of class precision, recall, f1-score
  - ✓ training data: 3 fold cross-validation result from fit method
  - ✓ test data: prediction using fitted model
- metrics by class
  - ✓ accuracy, precision, recall, f1-score
  - ✓ not provided by fit method cross validation calculation
  - ✓ training and test data: prediction using fitted model
- create dataframes with metric results
  - ✓ wrote pickle files
  - ✓ wrote visualization code (matplotlib, seaborn)

Initial Model Investigation: `sklearn` Results

- SVM ran for 2.5 hr but did not complete
  - ✓ others finished in < 10 min; did not try to resolve
- Gaussian NB: very large confusion among several classes
- model comparison
  - ✓ similar performance on train and test data
  - ✓ gaussian NB has worse performance
  - ✓ multinomial and complement NB: almost identical performance
  - ✓ logistic regression has slightly better performance than NB
  - ✓ weighted averages of precision, recall, f1 larger than macro averages due to decreased emphasis on poor performing small classes

Initial Model Investigation: `sklearn` Results (con't)

- **performance by class**
  - ✓ similar performance on train and test data
  - ✓ all models have difficulty with 3 smallest classes
  - ✓ complement NB has no predictions for 2 small classes
  - ✓ gaussian NB has high recall for 3 smallest classes but poor precision for them

Initial Model Investigation: Tree Results

- model comparison
  - ✓ GB, XGBoost, LightGBM, CatBoost almost identical performance
  - ✓ they are better than decision tree and random forest
- performance by class
  - ✓ random forest does not predict any instances of 4 smallest classes
  - ✓ decision tree performance on 4 smallest classes poorer than on 3 largest classes
- GB, XGBoost, LightGBM, CatBoost on 4 smallest classes
  - ✓ generally better precision than recall
  - ✓ XGBoost slightly better precision and recall than others; CatBoost generally performs worse
- LightGBM, CatBoost (use categorical features directly)
  - ✓ very similar performance
  - ✓ LightGBM slightly better for precision, CatBoost slightly better for recall

Model Development Process

- use `pycaret`
  - ✓ feature pre-processing
  - ✓ fit and compare several models (default hyper-parameters)
  - ✓ hyper-parameter tuning of a few selected models
  - ✓ finalize model and save to pickle file
- work with subset of data on local machine
  - ✓ develop and debug Jupyter notebooks
  - ✓ techniques for encoding high cardinality feature (soil type)
  - ✓ oversampling of target minority classes
  - ✓ how to access train and test datasets
  - ✓ make predictions
  - ✓ prepare result dataframes used for plotting
  - ✓ prepare plots (from pickled result dataframes)

Model Development Process (con't)

- work with full dataset on Paperspace (cloud provider)
  - ✓ initial investigations with free machine (8 vCPU, 30 GB memory, 6 hr time limit)
  - ✓ tuning with paid machine (12 vCPU, 30 GB memory, no time limit)
- model metrics from 3-fold cross-validation
  - ✓ accuracy
  - ✓ average and weighted average of precision, recall, f1-score
- metrics by class (predict training data with fitted model)
  - ✓ accuracy, precision, recall, f1-score

Data Pre-Processing

- soil index: non-ordinal, high cardinality (40 values)
  - ✓ manual clustering using domain knowledge
  - ✓ 3 clustering techniques (climatic zone, geologic zone, both zones)
- min-max scaling of numerical features
- over sampling of minority classes using `imblearn`

Model Development: Soil Type

- non-ordinal, high categorical feature (40 values)
- investigated these techniques for handling
  - ✓ one-hot encoding
  - ✓ frequency encoding
  - ✓ climatic zone clustering (7 values)
  - ✓ geologic zone clustering (4 values)
  - ✓ climatic + geologic zone clustering (11 combinations present)
- investigate these models (default hyper-parameters)
  - ✓ logistic regression
  - ✓ decision tree
  - ✓ XGBoost
  - ✓ LightGBM (uses categorical features directly, not one-hot encoded)
  - ✓ CatBoost (uses categorical features directly, not one-hot encoded)

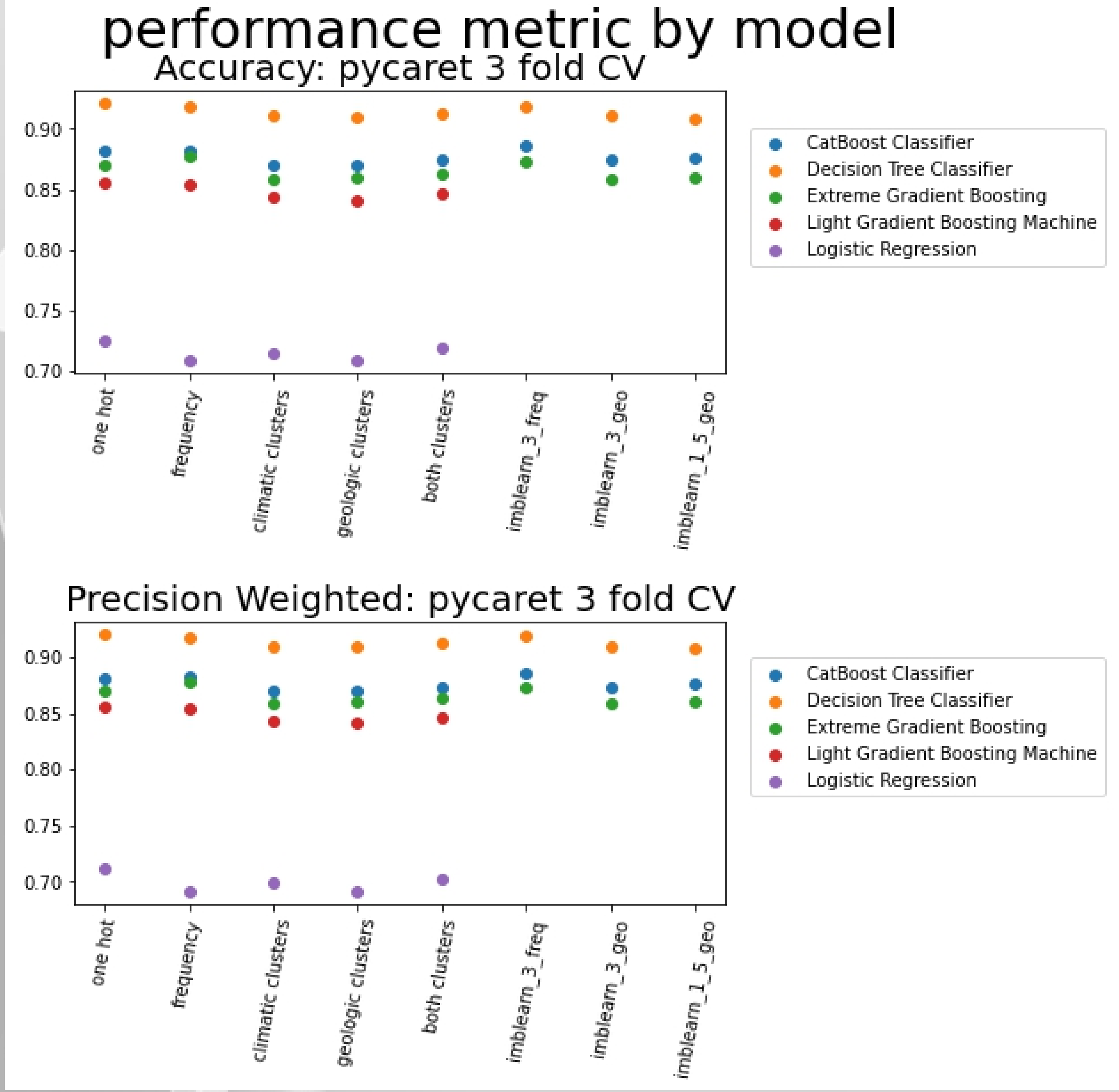
Model Development: Soil Type Results

- for each model
  - ✓ accuracy, weighted metrics, and average metrics slightly better for one-hot and frequency encoding
- decision tree
  - ✓ significantly better accuracy than other models
  - ✓ other metrics (except average precision) also better
- logistic regression significantly worse than other models
- 3 remaining models have similar performance
  - ✓ ordering best to worse: CatBoost, XGBoost, LightGBM
- for models that do not directly use categorical features
  - ✓ one-hot takes 20-40% more CPU time to train than frequency

Model Development: Soil Type Results (con’t)

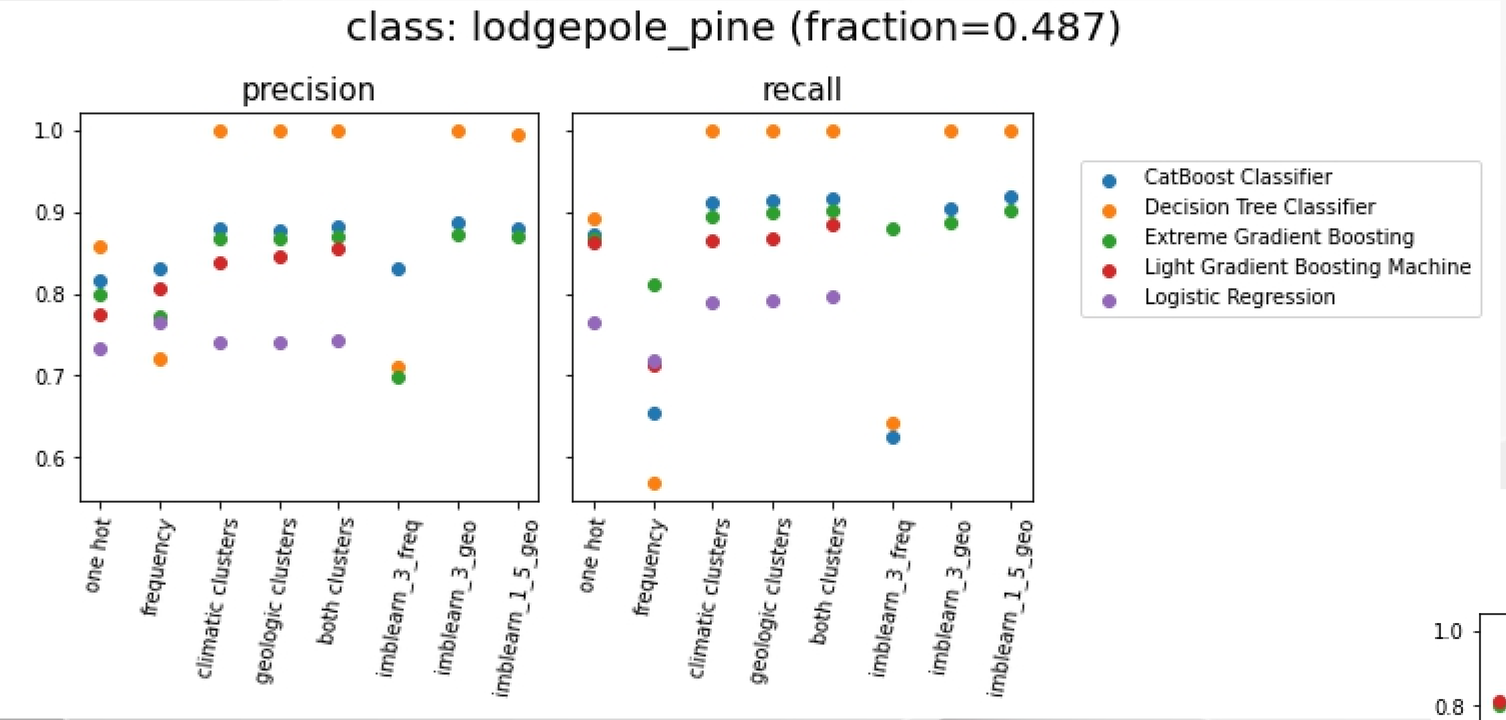
- **metrics by class (predict entire training set)**
  - ✓ generally, 3 clustering techniques perform better for each model
  - ✓ exception is logistic regression, which has significantly poorer performance than other models
  - ✓ little difference in performance among clustering techniques

Model Metrics: Screening Plots

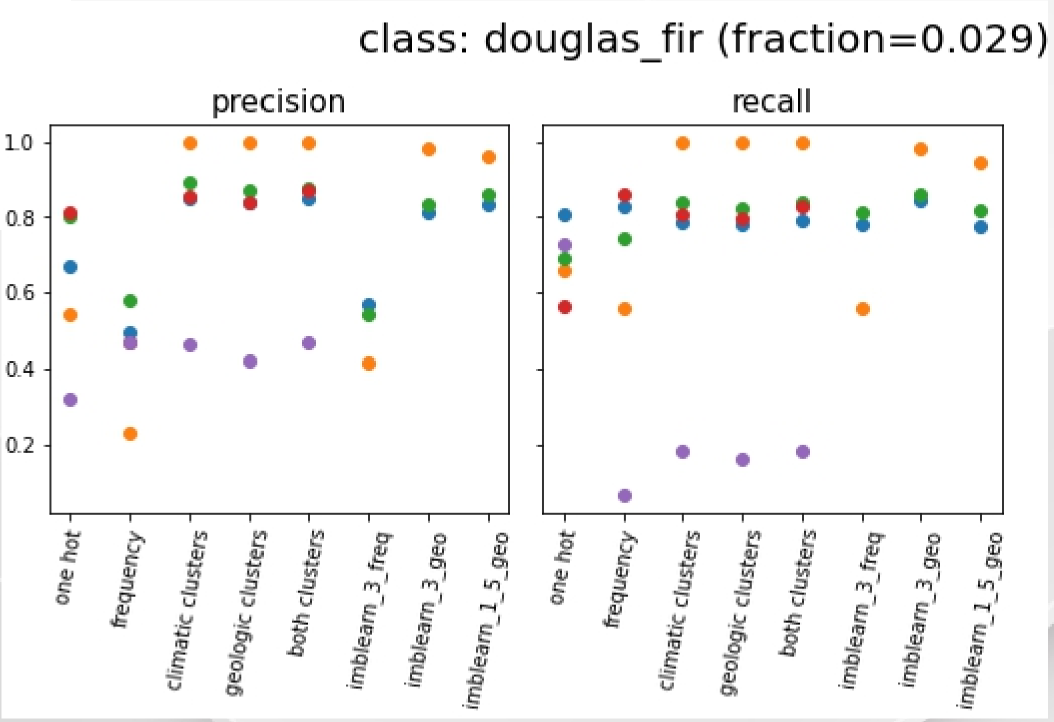


- metrics investigated
  - ✓ accuracy
  - ✓ average and weighted average of precision, recall, f1
  - ✓ 2 metrics shown; conclusions apply to all metrics
- one-hot and frequency encoding have slightly better performance for all models
- over-sampling of 5 smallest classes has only slight impact on performance

Class Metrics: Screening Plots



- lodgepole_pine: majority class
- douglas_fir: 3rd smallest class
- conclusions same for all classes



- 3 cluster techniques
  - ✓ similar performance for all models
  - ✓ significantly better than one-hot and frequency
- over-sample factor = 3
  - ✓ geologic cluster significantly better than frequency

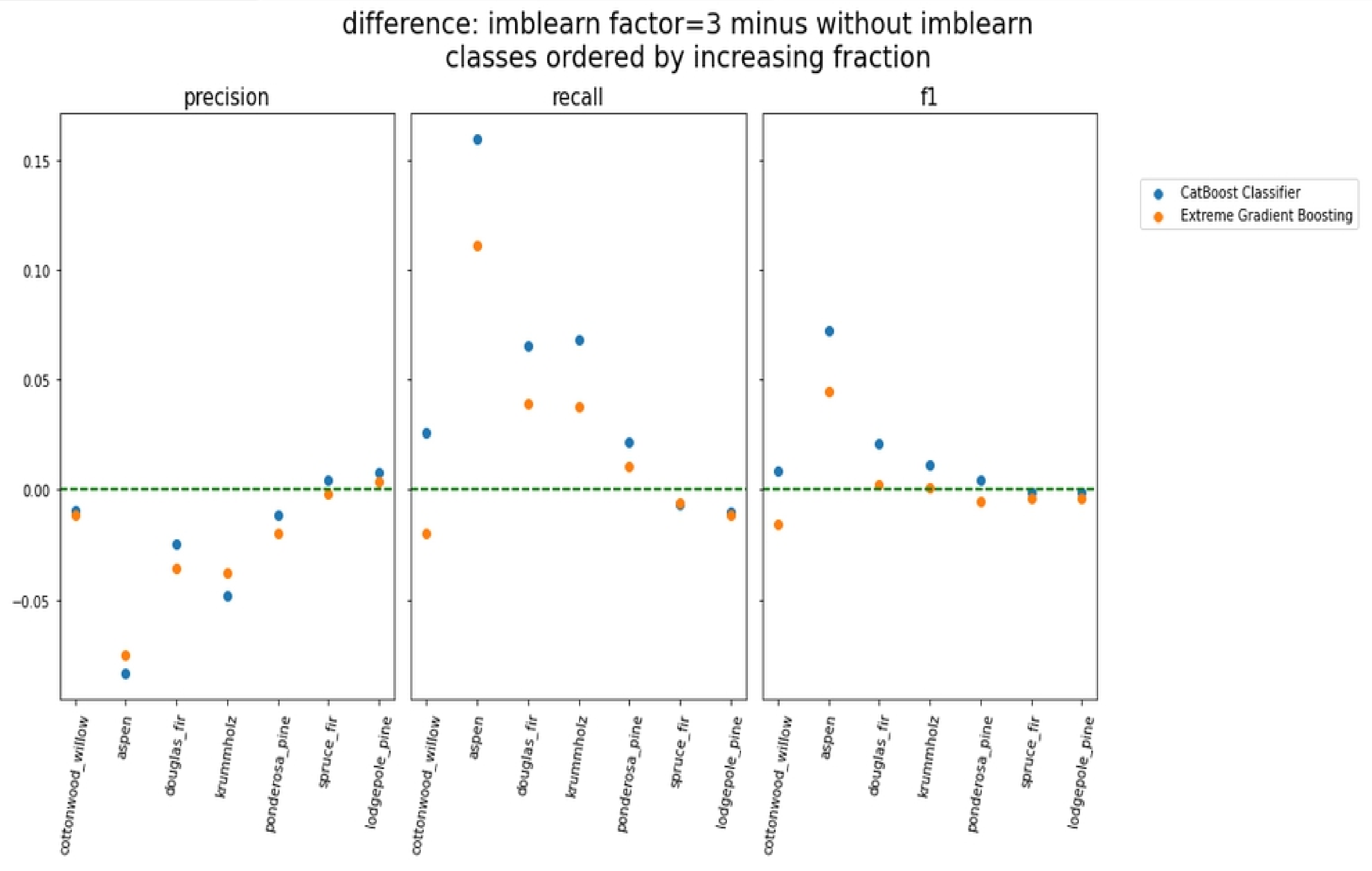
Model Development: Class Imbalance

- used `pycaret` functionality for including `imbalanced_learn`
  - ✓ over-sample and clean
- 5 smallest classes over-sampled to increase total count of each by factor of 3 in train data
- based on 3-fold CV results: use decision tree, XGBoost, CatBoost
- round 1: based on 3-fold CV results
  - ✓ use frequency encoding of soil type
- round 2: based on performance metrics by class
  - ✓ use geologic zone encoding (fewest clusters, 4 verses 7 or 11)
- round 3: over-sample by factor of 1.5
  - ✓ geologic zone encoding

Model Development: Class Imbalance Results

- over-sample with factor = 3: model metrics
  - ✓ little difference between with and without over-sampling
- over-sample with factor = 3: metrics for 5 smallest classes
  - ✓ with over-sampling, precision slightly lower and recall slightly higher
- over-sample with factor = 1.5: metrics for 5 smallest classes
  - ✓ only slight differences with and without over-sampling
  - ✓ differences smaller than factor = 3
  - ✓ no clear pattern to differences

Class Metrics: Over-Sampling



- over-sample 5 smallest classes
  - ✓ precision reduced
  - ✓ recall increased

Model Tuning: Parameters

- geologic clustering for soil type
- over-sample factor = 3
- 3-fold cross-validation, 15 iterations
- random grid search

Decision Tree

- parameter grid

```
tune_grid = {'max_depth': [5, 10, 15, 20, 25],
             'min_samples_leaf': [2, 4, 6, 8, 10],
             'min_samples_split': [6, 8, 10],
             'criterion': ['gini', 'entropy'],
             'max_features': [1.0, 'sqrt', 'log2']
            }
```

- best model

```
{'actual_estimator__min_samples_split': 6,
 'actual_estimator__min_samples_leaf': 4,
 'actual_estimator__max_features': 1.0,
 'actual_estimator__max_depth': 20,
 'actual_estimator__criterion': 'entropy'}
```

CatBoost

- parameter grid 1

```
tune_grid_cb = {'n_estimators': [50, 100, 150, 200, 300],
                'random_strength': [0.0, 0.2, 0.4, 0.6, 0.8],
                'l2_leaf_reg': [0.1, 1, 10, 100],
                'depth': [2, 4, 6, 8, 10]
               }
```

- best model 1
  - ✓ significantly worse than best decision tree

```
{'actual_estimator__random_strength': 0.0,
 'actual_estimator__n_estimators': 300,
 'actual_estimator__l2_leaf_reg': 0.1,
 'actual_estimator__depth': 8}
```

- parameter grid 2, same as above except

```
tune_grid_cb['n_estimators'] = [200, 300, 400, 500]
```

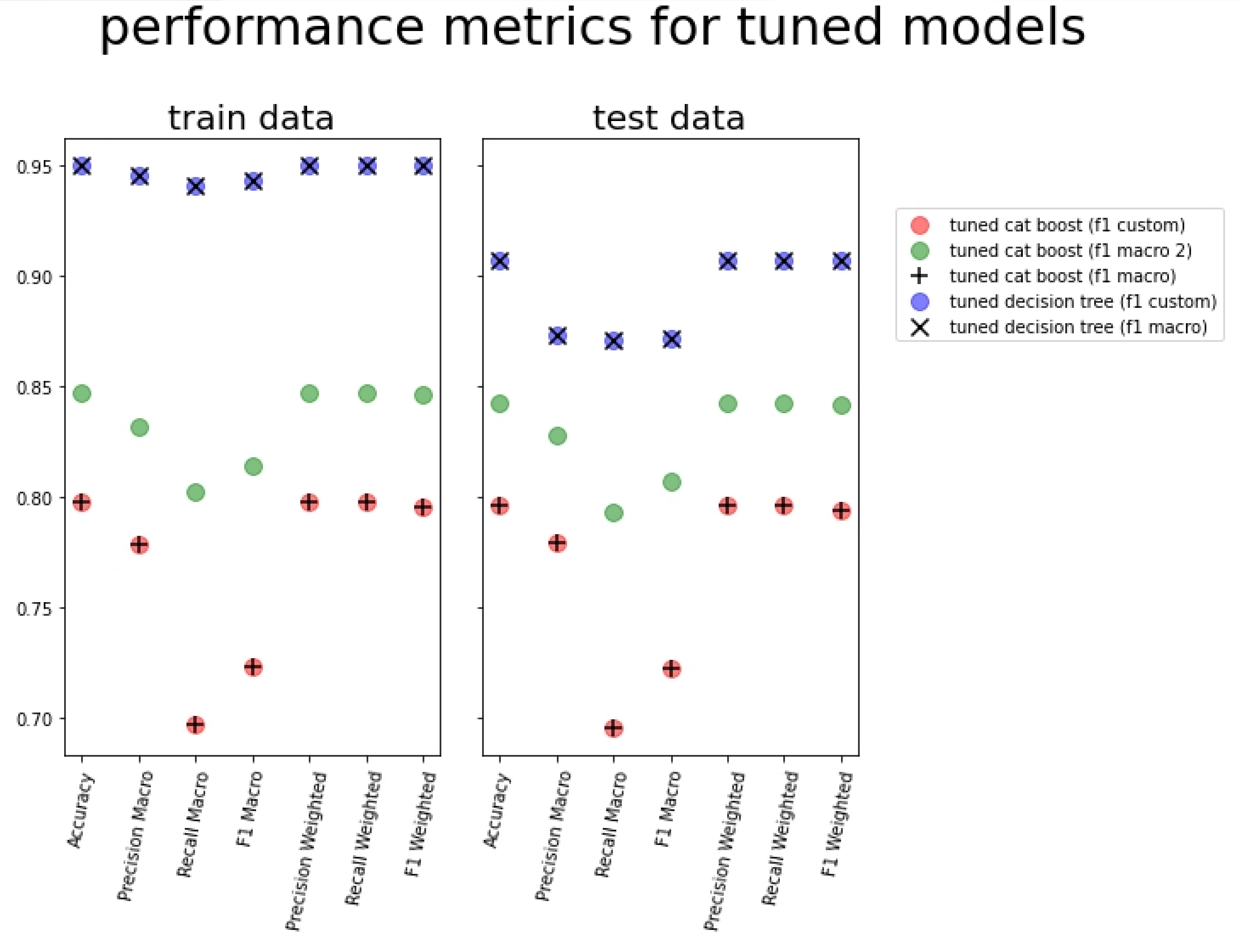
- best model 2

```
{'actual_estimator__random_strength': 0.2,
 'actual_estimator__n_estimators': 400,
 'actual_estimator__l2_leaf_reg': 1,
 'actual_estimator__depth': 10}
```

Model Tuning: Metrics

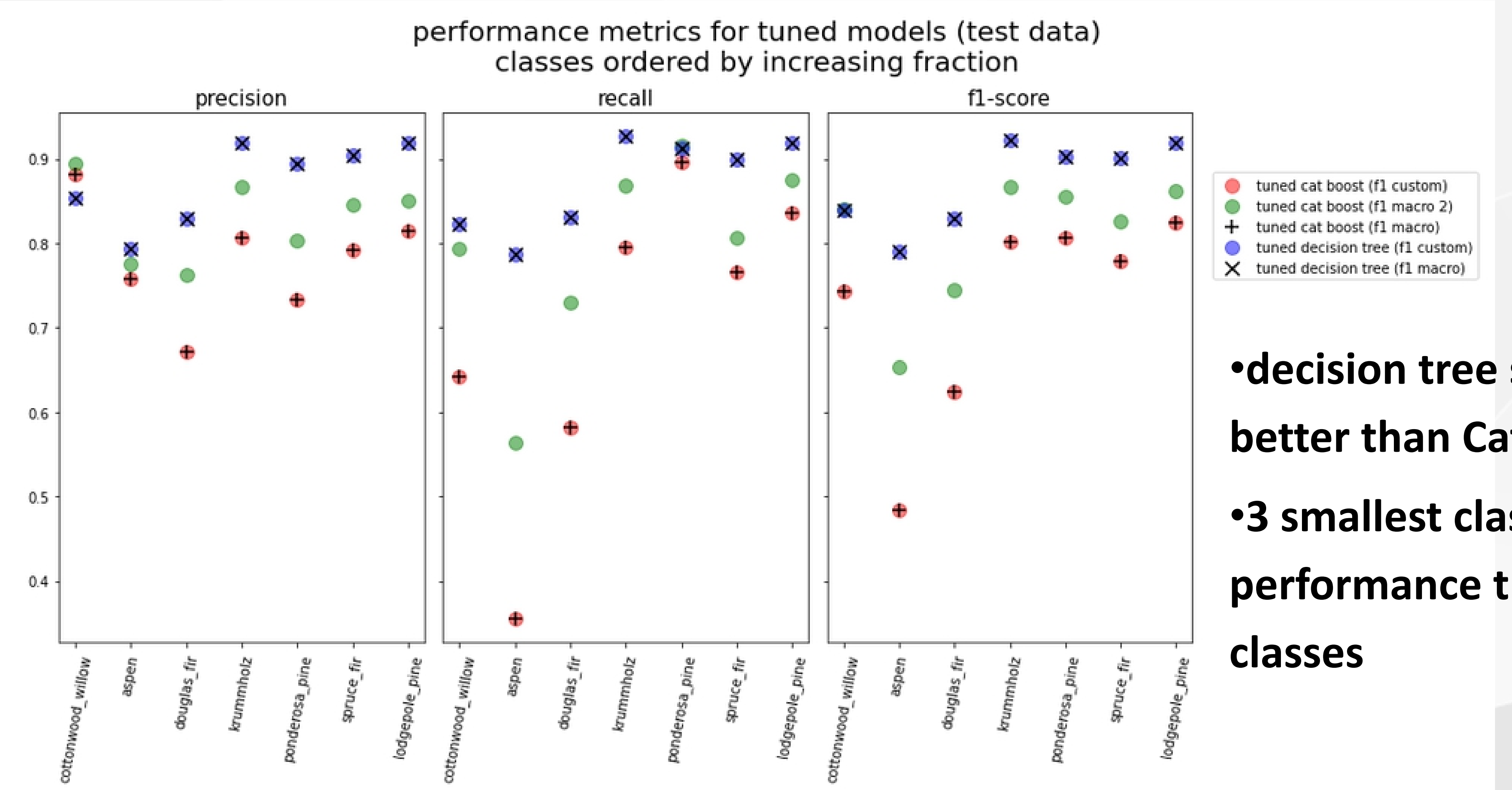
- f1 macro
  - ✓ average f1-score for all 7 classes, each class weight = 1
- f1 custom
  - ✓ average f1-score, 5 smallest classes weight = 3, 2 largest weight = 1

Model Tuning: Overall Results



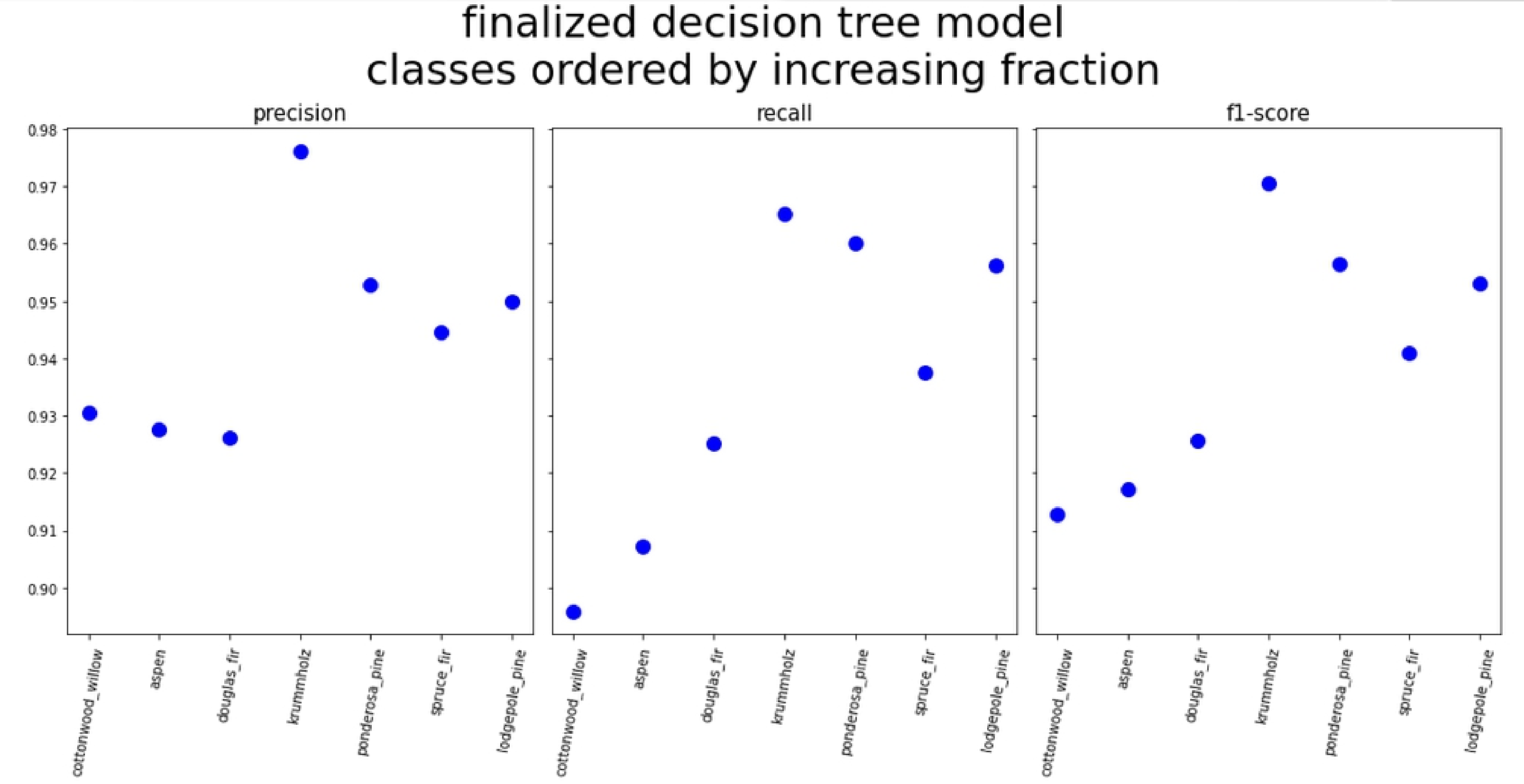
- optimization metric has no impact on results
- decision tree significantly better than CatBoost
- weighted metrics larger than macro metrics due to decreased emphasis on poorer performing small classes

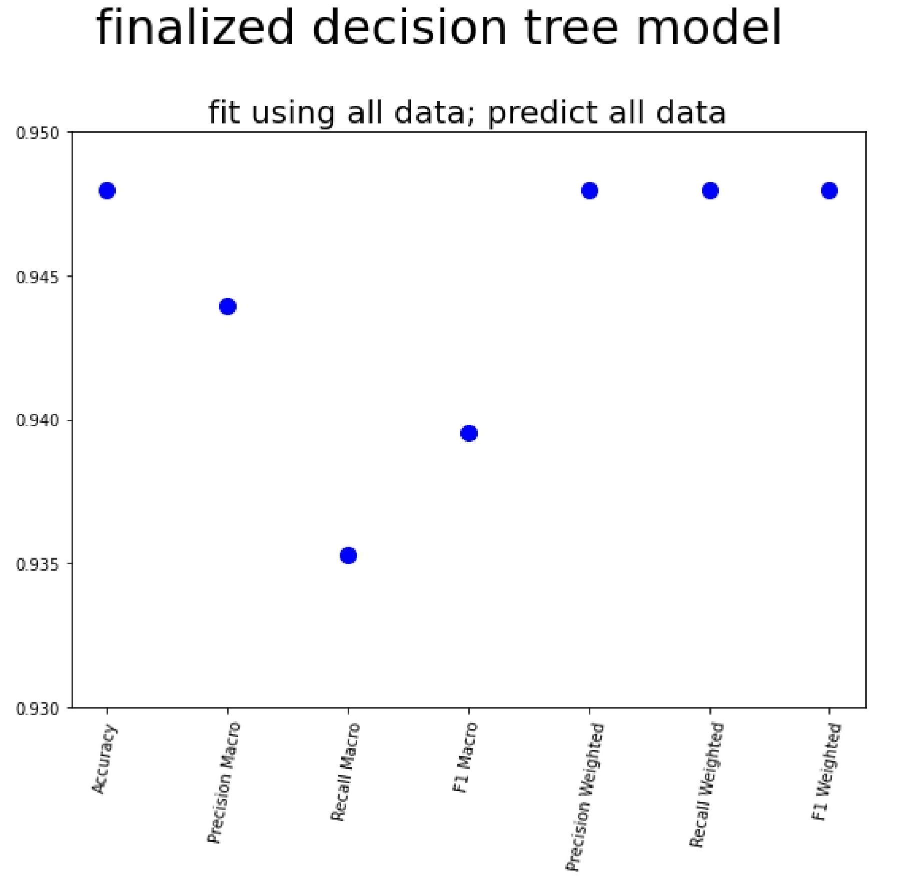
Model Tuning: Results by Class



- decision tree significantly better than CatBoost
- 3 smallest classes worse performance than other classes

Finalize Model





- decision tree selected for finalization
- fit using all data

Model Deployment

- python script for making prediction
  - ✓ command line argument--csv file containing input data
  - ✓ output--new csv file with 2 additional columns for cover index and name
  - ✓ log file--data check information, steps executed
  - ✓ reads pickle file of fitted model
  - ✓ if input data error found, message written to log file and execution stops

Model Deployment (con’t)

- **reproduce model**
  - ✓ **GitHub repository with all code**
  - ✓ **read original data, pre-process data, fit model, write pickle file of fitted model**
  - ✓ **ReadMe with step-by-step instructions**
- **testing**
  - ✓ **several csv files with good bad and various types of bad data**
  - ✓ **python functions for data checking**
  - ✓ **python script executes all functions on all csv files**
  - ✓ **writes log file with information about checks that passed and failed**