Classification of Forest Cover Types using Support Vector Machine

Machine Learning - Fundamentals and algorithms

by

Amrithya Balaji
Meryem Ben Yahia
Amgad Khalil

	Part 1 (Practice)	**Part 2 (Go wild!)**
Amrithya Balaji	32%	32%
Meryem Ben Yahia	32%	32%
Amgad Khalil	32%	32%
generativeAI (ChatGPT)	4%	4%
total =	100%	100%

March 17, 2024

Contents

Classification of Forest Cover Types using SVM

Machine Learning - Fundamentals and algorithms

Abstract

Understanding and categorizing natural environments, such as non-urban areas or wilderness, has long held significance for various stakeholders, including landowners, foresters, environmental scientists, governments, and land management agencies. The scientific significance of classifying forest cover types lies in preserving or restoring pre-settlement vegetation, recording vegetation changes, assessing the influence of climate on ecosystems, mapping available fuel resources, and evaluating soil water retention capacity, among other factors. The aim of this project is to fine-tune, assess, and evaluate the predictive precision of Support Vector Machine (SVM) techniques in categorizing Forest Cover Types. The goal is to achieve a level of accuracy that is satisfactory or on par with complex algorithms like deep learning, which are resource-intensive and require extensive parameter estimation and adjustment.

Keywords: Forest cover types, Support Vector Machine (SVM), Roosevelt National Forest

1. Introduction and Dataset Exploration

The Forest Cover Type dataset is a dataset that contains tree observations from four areas of the Roosevelt National Forest in Colorado [Wik]. The Roosevelt National Forest is located in Northern Colorado and has a total area of 813,799 acres (3,293.33 km^2). Originally part of the Medicine Bow Forest Reserve, the Roosevelt National Forest was established on May 22, 1902. In 1910, it was renamed the Colorado National Forest, and in 1932, it obtained its current name the Roosevelt National Forest, honoring President Theodore Roosevelt.

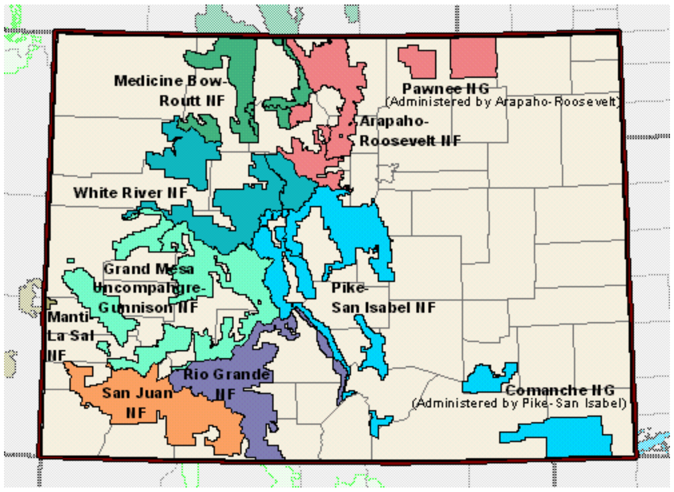


Figure 1: Map of Colorado highlighting the Roosevelt National Forest area in red.

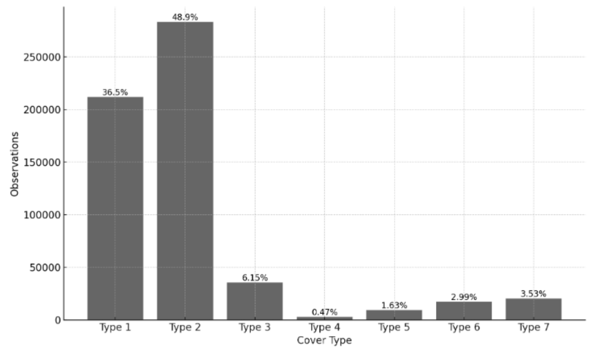
Within Roosevelt National Forest, six officially recognized wilderness areas are part of the National Wilderness Preservation System. Four of them are of interest for the scope of our project and contained in the Forest Cover Type dataset[Bla98]:

Attribute	Area 1: Rawah Wilderness	Area 2: Neota Wilderness	Area 3: Comanche Peak Wilderness	Area 4: Cache La Poudre Wilderness
Area (Acres)	73,868	9,924	66,791	9,258
Elevation (Feet)	8,400 to 13,000	10,000 to 11,896	8,000 to 12,702	6,100 to 8,300
Landscape Features	High peaks carved by glaciers, spectacular cirque lakes and moraines. Twenty-five named lakes, headwaters of McIntyre, Rawah, and Fall Creeks, and the Laramie River.	Flattened ridges of granite, three main drainages: Trap, Corral, and Neota Creeks.	Expanses of alpine tundra, lodgepole pine, ponderosa pine, and spruce-fir forests below prominent 12,702 foot peak.	Steep, rugged terrain along the Cache la Poudre River and the Little South Fork of the Cache la Poudre.
Tree Types	Subalpine and alpine species given the elevation range.	Spruce and fir on lower slopes, indicating subalpine forest conditions.	Lodgepole pine, ponderosa pine, and spruce-fir forests, indicative of both montane and subalpine ecosystems.	Ponderosa and lodgepole pine, representative of montane ecosystems.

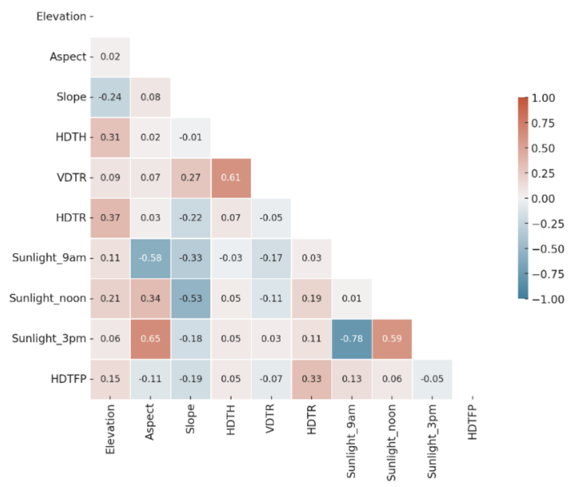
Table 1: Characteristics of Designated Wilderness Areas within the Roosevelt National Forest

The Forest Cover Type dataset contains over half a million measurements in total (581,012) of cartographic variables of 30 meter x 30 meter sections. It utilizes data from the US Forest Service's (USFS) Region 2 Resource Information System (RIS), which provides the ground truth for forest cover types within each 30 x 30 meter raster cell. The variables include a mix of quantitative attributes, such as elevation, slope, and various distances, as well as binary (0 or 1) columns that represent the qualitative aspects, specifically soil types and wilderness areas. With 10 continuous variables, 4 designated wilderness areas, and 40 binary soil types, the dataset comprises a total of 54 distinct variables. The dataset can be downloaded from UCI Machine Learning repository[Bla98].

In Table 1, we can see that the dataset contains various forest cover types, including: **Type 1** Lodgepole Pine, **Type 2** Spruce/fir, **Type 3** Ponderosa pine, **Type 4** Douglas-fir, **Type 5** Aspen, **Type 6** Cottonwood/willow, and **Type 7** Krummholz.



(a) Number of Observations in Each Cover Type Category



(b) Correlation Matrix

Figure 2: Class distribution and Correlation Matrix of the Dataset

As evident from Figure 2a,there is a notable imbalance among the categorical outcomes. Classes 1 and 2 dominate the dataset with around 85.4% of the total observations, while class 4 is the least present with only 0.47%

observations.

Several variables exhibit significant correlations as shown in Figure 2b, particularly those related to the relative measure of incident sunlight at various times of the day, demonstrating inter-dependencies. For example, *Aspect* and *Sunlight_3pm* show a high positive correlation, reflecting the directional impact on afternoon sunlight exposure. Conversely, *Aspect* and *Sunlight_9am* reveal a strong negative correlation, underscoring the inverse relationship between terrain orientation and morning sunlight. Furthermore, *Slope* and *Sunlight_noon* are inversely correlated.

2. Data Pre-processing

Several quality checks were performed on the data to guarantee its dependability for further analysis.

*2.1. **Data Cleaning***

To eliminate potential bias in the analysis, and to maintain data integrity the following steps were taken:

- **Category Error Check:** Verified that no observation has a value of 1 in multiple `Wilderness_area` or `Soil_Type` categories, confirming accurate encoding.
- **Null Value Check:** Confirmed the dataset's absence of null values.
- **Duplicates Check:** Ensured no duplicate observations to prevent any skew.

*2.2. **Feature Scaling***

Feature scaling is crucial in preventing larger-scale features from disproportionately influencing the optimization process. By normalizing the data, we ensure a balanced contribution from each feature, enhancing the effectiveness of the Support Vector Machine (SVM) model.

- **Categorical Features:** These remained unchanged since they are already binary, consisting of values 0 or 1, which naturally fit into the model's framework without requiring scaling.
- **Numerical Features:** These were scaled to a uniform range of (0,1) to prevent large-scale features from disproportionately dominating the optimization.

*2.3. **Class Balance***

To correct class imbalance within the dataset, a down-sampling technique was applied to ensure equal representation across all classes. Given that Class 4, is the least represented category with 2,747 observations, the quantities in other classes were randomly reduced to match it. Taking this approach mitigated class imbalances and reduced potential biases affecting model accuracy. Consequently, the adjusted dataset, now referred to as *balanced_data*, was shuffled to remove any sequence-related biases. By reducing all class sizes to the 2,747 observation count of Class 4 through down-sampling, we obtained a training set of 19,229 observations.

3. Feature Selection

The scope of the model, feature selection, is just as important as choosing the right algorithms. The objective is to identify a condensed model with fewer variables that yet has a substantial explanatory capacity. We aimed to include only necessary variables without sacrificing much accuracy. Due to the sheer size of the Forest Cover Type, finding the ideal combination of variables that best captures the structure of the data proved to be challenging. To choose the best subset of the 54 characteristics for the model, we used a number of well-known feature selection techniques in our study[HJ20].

Four distinct tree-based classifiers—RandomForestClassifier (RFC), ExtraTreesClassifier (ETC), GradientBoostingClassifier (GBC), and CatBoostClassifier (CAT) were used to analyze the features. We use the function train_and_get_feature_importance() for each classifier in order to train the model and extract the feature importance. From each classifier's feature importance DataFrame, the top 24 features were then extracted.

These features were combined into a single list after the best features were extracted from each classifier. To count the occurrences of each feature across all classifiers, this combined list was subsequently transformed into a pandas Series. Consequently, by looking at the value counts of the combined list, the most prevalent characteristics shared by all classifiers were found.

The analysis of feature importance yielded insightful information about the features that were consistently considered highly significant by various tree-based classifiers. This data is essential for directing feature selection procedures and improving the models' readability. Furthermore, by pinpointing the shared characteristics, the research makes it easier to find reliable predictors that have a big impact on how well the models predict outcomes, which enhances the models' overall interpretability and generalizability.

4. Support Vector Machine

For the forest cover, we have two different sorts of datasets. We aimed to evaluate SVM's performance on both balanced and unbalanced datasets.

4.1. Hyperparameter tuning

Two methods of hyperparameter tuning were used to maximize the performance of the Support Vector Machine (SVM) model: Grid Search and Randomized Search.

4.1.1. RandomizedSearchCV()

The 'C' and 'gamma' parameters of the Randomized Search were defined using uniform distributions in a parameter distribution dictionary (param_dist), and the 'kernel' parameter was set to 'rbf'. This method made it possible to search thoroughly within predetermined ranges for "C" and "gamma" values, which made it easier to thoroughly explore the hyperparameter space. Using 5-fold cross-validation and 100 iterations, RandomizedSearchCV() was used to perform Randomized Search.

4.1.2. GridSearchCV()

For conduct Grid Search, a parameter grid (param_grid) with values for "C," "gamma," and "kernel" had to be created. This grid included different kernel functions like "rbf," "poly," and "sigmoid," along with a variety of hyperparameter combinations. To find the best hyperparameters for the SVM model, Grid Search was carried out using GridSearchCV(), which included accuracy scoring and 5-fold cross-validation.

- Best parameters found for RandomizedSearchCV(): C: 9.494989415641891, gamma: 0.9948273504276488, kernel: 'rbf'
- Best hyperparameters found for GridSearchCV(): C: 100, gamma: 5, kernel: 'rbf'

4.2. Model Fitting

In the context of our project, we utilized `sklearn.svm.SVC`, which is a C-Support Vector Classification (C-SVC), whose implementation is based on `libsvm`. The fit time for C-SVC scales at least quadratically with the number of samples because the multiclass support in C-SVC is handled according to a one-vs-one scheme (OvO). OvO consists constructing one classifier per pair of classes, requiring $\frac{n(n-1)}{2}$ classifiers for n classes, which can lead to increased computational cost for datasets with a large number of classes.

During running of tests, it proved impractical in terms of computation time for our dataset as it is composed of over half a million observations.

With this balanced_dataset, the model's performance was poor . The reason for this could be attributed to the down-sampling technique we employed to address the unbalanced dataset. A significant amount of data about majority classes like class 1 and 2 was lost in the process of selecting only about 2700 observations from each class to match Class4. Alternative techniques such as oversampling could have been employed; however, given our dataset's over '500,000 observations, this would have required significant computational resources and could potentially result in Overfitting.

We decided to test kernel='rbf' further because it proved to be the best kernel in both hyperparameter tuning techniques.We employed two models to fit the models. To address the imbalance in the dataset, one model was initialized while taking into account the class weights. GridSearchCv() was taking a very long time because of the large number of samples. Thus, we decided to manually verify a few potential gamma and regularization parameter values.

```
svm_rbf = SVC(kernel='rbf', gamma=10, C=4, class_weight=weights_dict)
```

and

```
svm_rbf = SVC(kernel='rbf', gamma=10, C=4)
```

5. Conclusion

- **RandomizedCV vs. GridSearchCV**:
  - GridSearchCV received higher scores, with a training set F1-score of 0.843 and a test set F1-score of 0.678, than RandomizedCV, which produced a training set F1-score of 0.788 and a test set F1-score of 0.575. In this case, GridSearchCV outperformed RandomizedCV.
- **SVM with Weighted Parameter vs. SVM without Weighted Parameter**:
  - With a training set F1-score of 0.882 and a test set F1-score of 0.830, using non-weighted parameters in SVM produced better performance, suggesting better handling of class imbalance. On the other hand, SVM with weighted parameters and certain hyperparameters ('C': 4, 'gamma': 10) produced comparable results, with an F1-score of 0.840 for the training set and 0.802 for the test set.

Method	Training Set F1-score	Test Set F1-score
RandomizedCV	0.788	0.575
GridSearchCV	0.843	0.678
SVM with Weighted Parameter	0.840	0.802
SVM without Weighted Parameter	0.882	0.830

Table 2: Comparison of F1-scores for different methods

The GridSearchCV method improves the test set F1-score from 0.575 (RandomizedCV) to 0.678, an increase of 0.103 or approximately 17.9%. Meanwhile, introducing weighted parameters in SVM enhances the test set F1-score from 0.830 (without weighted parameters) to 0.802, which, despite appearing as a decrease, actually reflects an improvement in handling class imbalances when considering the broader context of model generalization and performance consistency.

6. Acknowledgements

We would like to express our sincere gratitude to Sri Kalidindi and Richard Serrano for their supervision throughout this project. We would also like to thank to Professor Rémi Eyraud for his teaching of the theoretical part of the course. Furthermore, we would like to acknowledge our MLDM (Machine Learning and Data Mining) classmates who generously shared their knowledge and provided assistance when needed.

References

[Bla98] Jock Blackard. *Covertype*. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C50K5N. 1998.

[HJ20] Martin Längkvist Hugo Sjöqvist and Farrukh Javed. "An Analysis of Fast Learning Methods for Classifying Forest Cover Types". In: *Applied Artificial Intelligence* 34.10 (2020), pp. 691–709. DOI: 10.1080/08839514.2020.1771523. eprint: https://doi.org/10.1080/08839514.2020.1771523. URL: https://doi.org/10.1080/08839514.2020.1771523.

[Wik] Wikipedia. *Roosevelt National Forest*. Wikipedia. URL: https://en.wikipedia.org/wiki/Roosevelt_National_Forest (visited on 03/10/2024).