CS7641: Supervised Learning Report

[1st] Nha Van Thanh Do
College of Computing
Georgia Institute of Technology
Atlanta, Georgia
ndo46@gatech.edu

***Abstract*—This report builds a reproducible supervised learning evaluation harness for a single multiclass classification problem using the Forest Cover Type dataset with four different algorithms: Decision Trees, k-Nearest Neighbors, Support Vector Machines, and Neural Networks (MLP, SGD only).**

I. Introduction

Supervised Learning is a machine learning technique that uses labeled dataset to train a model to identify the underlying patterns and relationships [1] and generate predictions. There are many supervised learning algorithms but the selection depends heavily on the dataset. Also, understanding the dataset is mandatory to come up with the right metrics for model evaluation. For this particular project, since the data is extremely imbalanced, from around nearly 50% (type 2) down to under 1% (type 4), accuracy itself can be dominated by the majority classes and can be misunderstood about model performance. Therefore, some other metrics such as Macro-F1 must be included. The goal of this project is to compare and analyze model performance of selected supervised learning algorithms.

I have implemented four different algorithms: Decision Trees, kNN, SVM and Neural Networks evaluated by accuracy, macro-F1, balanced accuracy, confusion matrix and per-class precision/recall/F1 metrics. Every experiment uses a single subset of 20,000 instance stratified sample from the full dataset that contains more than half a million instances. I also propose a hypothesis about algorithmic performance and will compare with the actual result after final experiment.

II. Data, EDA, and Sampling Design

A. Forest Cover Type Dataset

This dataset contains tree observations from four areas of the Roosevelt National Forest in Colorado [2] with the total of 581,012 instances and 54 features. Ten of the features are continuous: elevation, aspect, slope, four distance to utility columns and three hillshade indices. Four of them are binary wilderness area indicators and the remaining forty are binary soil type indicators along with one cover type which is our target column for prediction. Cover_type is an integer ranging from 1 to 7 corresponding to 7 primary tree species in these areas: Spruce/Fir, Lodgepole Pine, Ponderosa Pine, Cottonwood/Willow, Aspen, Douglas-fir, and Krummholz [2].

B. Exploratory Data Analysis (EDA)

- Class imbalance: The 7 classes are distributed very unevenly. As seen in Figure 2, classes 1 and 2 hold about 85% of the entire dataset while class 4 has only 0.5% which makes the prediction for class 4 becomes extremely difficult because of data shortage. The remaining classes is also low. This imbalance makes accuracy metric itself less reliable because the model might just predict the majority class every time. Macro-F1 score will be used to evaluate model performance. For this project, both accuracy and macro-f1 will be used.
- Specific attribute: Among the attributes, elevation is the strongest single discriminator. Type 7 Krummholz could be separated by this attribute because of its high elevation compared to others.

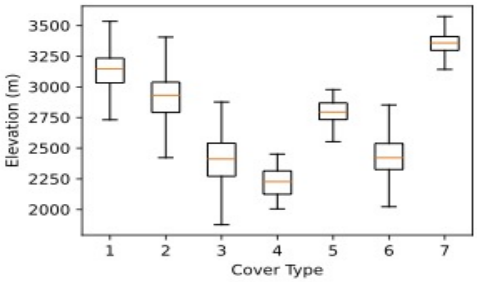


Fig. 1: Elevation Distribution of Cover Type

- Feature scale: Table 1 shows that 10 continuous features have very wide ranges. Elevation is ranging from 1876 to 3856 meters while hillshade indices are 0-255 and distance attributes are scaled from 0 to more than 7000. These high variance will have significant impact to distance and margin based models like kNN, SVM and MLP because they are dominated by large magnitude. Preprocessing to standardize the continuous features is required.

C. Sampling Design

Since the full dataset is too large, a smaller dataset of approximately 20,000 instances is sampled with StratifiedShuffleSplit and seed of 85 so that we can always reproduce the result. This sampling design is chosen to ensure that the ratio of classes in the training and testing sets matches the ratio in the original dataset. It's critical for this imbalanced

TABLE I: Feature Scale

Feature	***min***	***max***	***mean***	***std***
Elevation	1876	3846	2959.65	280.76
Aspect	0	360	154.94	111.64
Slope	0	55	14.08	7.49
H. Dist. Hydrology	0	1361	269.30	211.35
V. Dist. Hydrology	−139	595	46.22	58.26
H. Dist. Roadways	0	7043	2338.21	1559.98
Hillshade 9am	58	254	212.28	26.77
Hillshade Noon	97	254	223.26	19.65
Hillshade 3pm	0	252	142.30	38.16
H. Dist. Fire Points	30	7081	1964.67	1309.82

dataset where random or other splits might result in a missing of minority class and prevents the model from becoming biased toward the majority class due to a skewed training set. However, even stratified, class 4 still ends up with only around 100 instances.

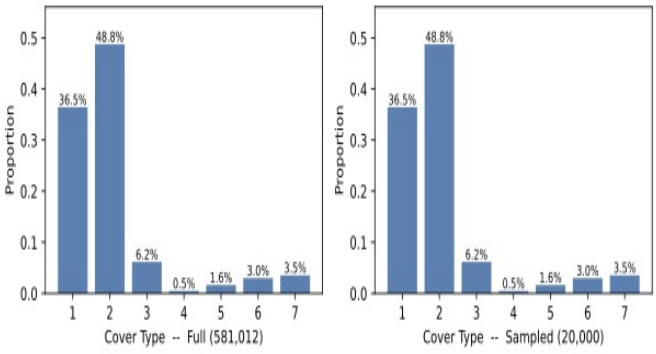


Fig. 2: Class Distribution for Covertype Dataset

III. Methodology and Hypothesis

A. Methodology

I use stratified 80/20 train/test split with seed of 85 which gives 16,000 training and 4,000 test records. Inside the training set, I run GridSearchCV with stratified k-fold splits and scoring="f1_macro" so the inner objective matches the imbalance the outer evaluation will be judged on. I also use 5-fold CV for Decision Trees and kNN and drop to 3-fold CV for SVM and MLPs.

Among the models, DT will see raw data without scaling because it has no impact to this algorithm. The remaining models will see standardized continuous features explained above. At final-fit time, after CV picks the best hyperparameters, the pipeline refits on the full 16,000 training set. For every evaluation point, I report accuracy, macro-f1, balanced accuracy, confusion matrix and per-class precision, recall, and F1. Confusion matrix will help to see which cover types get confused and how does it impact to the model performance.

To prove the score is not impacted by leakage, I run most frequent dummy, stratified random dummy and decision tree trained on shuffled labels using built-in scikit-learn DummyClassifier function. The accuracy metric is high (still less than 0.5) but that is a result of an imbalanced dataset while macro-F1 and balanced accuracy are very low. It confirms there is no target leakage hiding in the feature matrix.

TABLE II: Leakage Evaluation

Model	***Accuracy***	***Macro-F1***	***Balanced Accuracy***
Dummy (most-frequent)	0.488	0.094	0.143
Dummy (stratified)	0.393	0.147	0.147
DT (shuffled labels)	0.476	0.104	0.141

B. Hypothesis

Based on the observation from EDA, I think Neural Networks would be the best algorithm. This dataset contains numerous environmental variables and the relationships are not linear which requires complex decision boundaries. Although the dataset is reduced to 20,000 instances, this still provides enough training data point for Neural Networks to learn non-linear patterns. In similar logic, the second best algorithm would be RBF-SVM because it could capture complex geometric structures in the data to form non-linear boundaries. My expectation is that Neural Networks and RBF-SVM might be comparable in terms of performance depending on hyperparameter choices. The third one might be Decision Trees. In the second point of EDA, I observed that type 7 could be differentiate by elevation. Since decision trees usually perform well with threshold, it could do relatively well on this task. KNN is expected to be moderate to weak performance because with 54 dimensions, it makes nearest neighbors become less distinct. In addition, except elevation, other attributes are close and overlap to each other, that will cause confusion for KNN to calculate the distance. KNN is highly impacted by dimensionality. The worst algorithm would be linear-SVM because this model can do well only for linear and sparse data but the boundaries are not likely to be linear.

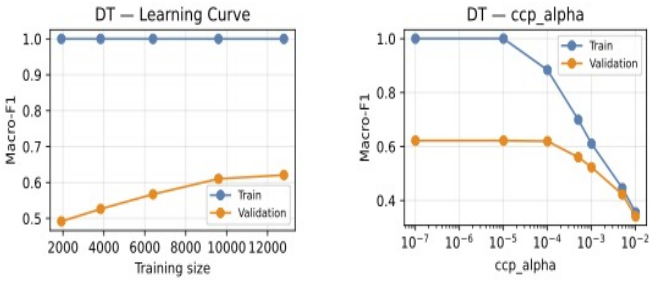
IV. Experiments

A. Decision Trees

I used Gini impurity as the splitting factor and balanced class weight to improve macro-f1 and balanced accuracy. Using this criteria ensures minority classes receive larger weights. Without balancing, the trees will favor class 1 and 2 over others. Post-prunning is controlled via ccp_alpha together with a max_depth cap. Grid search is applied with a wide range of values from 0 for no pruning at all to larger alpha when pruning becomes aggressive. After running the experiment, the results came up with `ccp_alpha=0` and `max_depth=None` giving a tree of depth 44 with 3,014 leaves over 16,000 training rows. CV Macro-F1 $= 0.628$. On the test set Macro-F1 $= 0.674$, balanced accuracy $= 0.672$, accuracy $= 0.756$. Result from grid search shows that for this dataset, the best tree is the one with no pruning at all.

Looking at the learning curve in Fig 3a, the training curve is always at 1.0 because unpruned tree will just memorize

the training fold, causing overfitting. Validation curve climbs steadily from 0.5 at 2,000 training records to about 0.63 at 12,800 rows. This is because 5-fold CV is being used so 1 fold is reserved for validation. The final tree is then refit on all 16,000 training data. The gap between train and validation is huge. It's the usual variance signature of an unprunred tree. However, as we can see that the curve is still rising, meaning more data would be helpful.



(a) Learning Curve (b) Model Complexity Curve

Fig. 3: Learning Curve and Model Complexity Curve for DT

The model complexity curve in Fig 3b shows validation macro-f1 is flat for small `ccp_alpha` and then starts falling off once it becomes larger than 10^{-4}. When `ccp_alpha` increases, the pruning starts removing the branches the minority classes need and causes the score dropping significantly. That means reducing tree complexity does not improve validation Macro-F1. Although the learning curve reveals evidence of overfitting, complexity curve demonstrates that additional regularization does not improve generalization performance. The rarest classes give up so little signal that aggressive pruning does more harm than good.

TABLE III: Per-Class Classification Performance for DT

Class	Precision	Recall	F1
Spruce/Fir	0.757	0.746	0.752
Lodgepole Pine	0.787	0.794	0.790
Ponderosa Pine	0.709	0.732	0.720
Cottonwood/Willow	0.778	0.737	0.757
Aspen	0.470	0.477	0.473
Douglas-fir	0.496	0.500	0.498
Krummholz	0.743	0.716	0.729

The per-class evaluation shown in Table III demonstrates that Decision Tree is effective at identifying the majority of forest cover types, particularly Spruce/Fir (class 1) and Lodgepole Pine (class 2). Krummholz (class 7) is a minority class but this has a special elevation attribute that can differ it with others and that's what Decision Tree is good at. Aspen and Douglas-fir sitting in the middle of the elevation range and has overlapped attributes that may confuse the model. As a result, prediction for these 2 minority classes is challenging. The most interesting thing is Cottonwood (class 4), which has the least data points but it ends up having a very high precision 0.778 and recall 0.737 and F1 0.757 which are comparable with the 2 majority classes. However, because the support for this class is very small, this class might be extremely sensitive and the result is less reliable.

The confusion matrix in Fig 4 explains why class 4 surprisingly performs well. Because it is confused with only class 3 and essentially no others. That also means class 4 has unique region of feature space that could separate it from the remaining classes. Class 5 and 6 are confusing with multiple classes that lowers their overall performance.

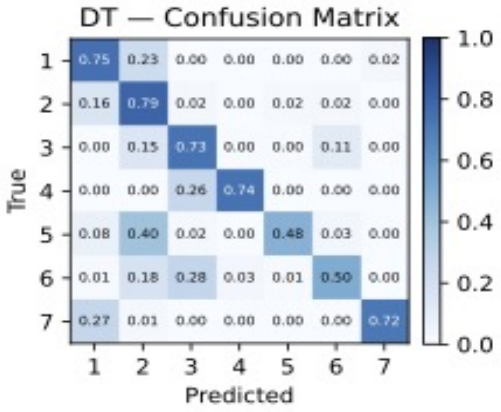


Fig. 4: Confusion Matrix for DT

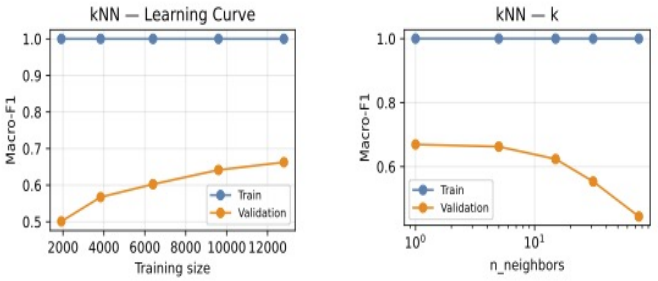
B. k-Nearest Neighbors

For KNN, similar approach is applied and grid search will examine through all possible combinations with $k \in \{1, 5, 15, 31, 75\}$ across `metric` $\in$ {Euclidean, Manhattan} and `weights` $\in$ {uniform, distance}. The major difference is scaling which is mandatory for correct implementation. Otherwise feature like elevation that has very high number will dominate everything else. A `StandardScalar` is applied ensuring each feature dimension contributes equally to the distance calculation. Grid search chose the best combination with $k = 5$, `metric=manhattan`, `weights=distance`. CV Macro-F1 $= 0.669$. On the test set Macro-F1 $= 0.680$, balanced accuracy $= 0.658$, accuracy $= 0.800$. L_1 distances hold up better than L_2 when a lot of irrelevant or sparse coordinates inflate the L_2 norm, and that is what the other binary soil-type indicators do. A prior work by Aggarwal, Hinneburg and Keim supports this result. Their experiments showed that the Manhattan distance metric is consistently more preferable than The Euclidean distance metric for high dimensional data mining application [3].

As seen in Fig 5a of learning curve, the training curve pinned at 1. The KNN memorizes the training fold by construction and because of distance weight applied, the point itself is one of the 5 neighbors and the self-neighbor gets essentially infinite influence keeping the training score always 1. However, the validation curve is still rising from 0.5 to about 0.65 meaning more data would still help.

Fig 5b is showing model complexity curve for KNN and the Macro-F1 reaches its maximum at $k = 5$ then starts dropping

as k increases. This behavior reflects the bias-variance in KNN and also implies the dimensionality sensitive of this algorithm. Bigger neighborhoods drag the minority classes toward the majority classes and causes the class boundaries to become less distinct which results in lower performance. Due to the dataset imbalance, class 5 (Aspen) has only 262 data points in the training split scattered across 54-D space, once the neighborhood pulls in dozens of points it gets swamped by majority classes (1 and 2).



(a) Learning Curve (b) Model Complexity Curve

Fig. 5: Learning Curve and Model Complexity Curve for KNN

Now, looking at table IV to deep dive into per-class metrics, there is no surprise when class 1 and 2 continue to outperform and class 7 has also high score due to its distinct feature. Cottonwood/Willow continues to do very well in KNN despite data shortages. Compared to Decision Trees, KNN seems to do slightly better for majority classes but worse for minority classes. This is because minority classes were absorbed into larger neighboring classes.

TABLE IV: Per-Class Classification Performance for KNN

Class	Precision	Recall	F1
Spruce/Fir	0.816	0.770	0.793
Lodgepole Pine	0.814	0.864	0.838
Ponderosa Pine	0.762	0.780	0.771
Cottonwood/Willow	0.688	0.579	0.629
Aspen	0.5	0.385	0.435
Douglas-fir	0.548	0.525	0.536
Krummholz	0.818	0.702	0.756

The confusion matrix keeps the strong majority diagonal but Cottonwood/Willow drops to 58% compared to Decision Trees but again the reason why this class still has pretty good accuracy because it gets confused only with Ponderosa Pine. The bad performance of class 5 and 6 is consistent with the characteristics of KNN where classification decisions are heavily influenced by local class density and class imbalance.

C. Support Vector Machines

The 2 kernels being used for SVM are linear and RBF, regularization parameter C and kernel coefficient gamma γ. Both use standardized features and `class_weight="balanced"` to give minority classes more importance. C and γ must be chosen properly to avoid overfitting or underfitting. Grid search returns the

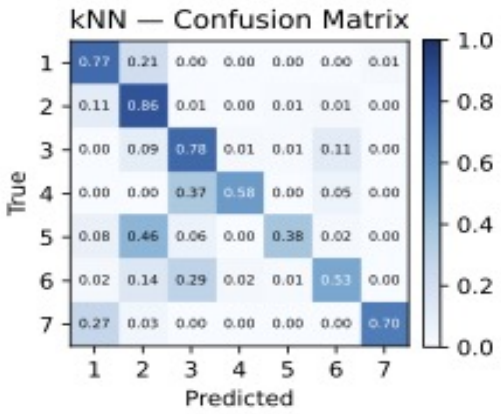
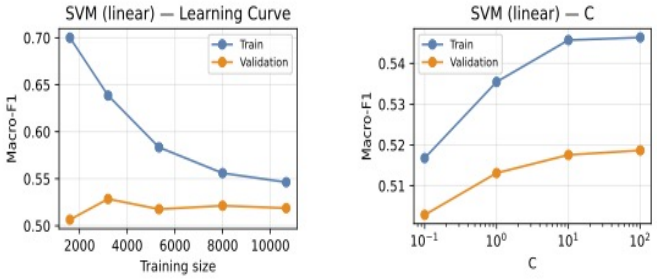


Fig. 6: Confusion Matrix for KNN

best combination for linear with $C = 100$, CV Macro-F1 $= 0.516$, test Macro-F1 $= 0.520$, balanced accuracy $= 0.697$, accuracy $= 0.609$ and for RBF with $C = 100$, $\gamma = 0.1$, CV Macro-F1 $= 0.690$ test Macro-F1 $= 0.707$, balanced accuracy $= 0.746$, accuracy $= 0.801$. This is a strong evidence that with the same parameter C, RBF outperforms linear kernel. It also implies that nonlinear boundaries must be the solution for cover type dataset.

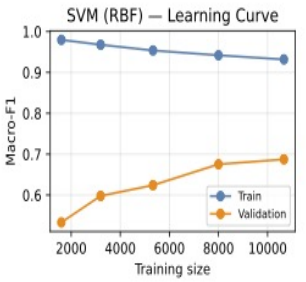


(a) Learning Curve (b) MCC - C parameter

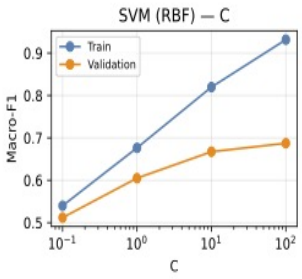
Fig. 7: Learning Curve and MCC for Linear SVM

The linear learning curve is shown in Fig 7a. As training size increases, train score decreases while validation is almost flat and both are close to around 0.52 and tend to remain stable as data grows. This means the model is underfitting and the limitation is not the amount of data but the model itself so adding more data points will not help, simply because linear boundaries do not work for this dataset. The complexity curve in Fig 7b indicates that both training and validation Macro-F1 increased as C increased. This suggests that the model was not overfitting but instead remained limited by the representational capacity of a linear decision boundary. This is expected situation as we know that forest cover type dataset contains complex nonlinear relationships that the linear kernel will not be sufficient.

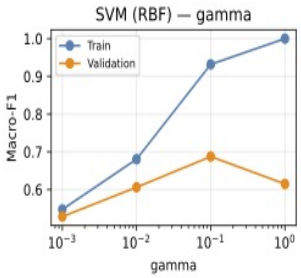
On the other hand, the RBF learning curve in Fig 8a is showing different story. Validation seems to grow healthy as



(a) Learning Curve



(b) MCC - C parameter



(c) MCC - γ parameter

Fig. 8: Learning Curve and MCC for RBF SVM

training size increases. That means the model is not overfit yet so more data would help. It also implies that if the model creates nonlinear boundaries, the Macro-F1 will be improved. The MCC curve for C parameter in Fig 8b continues to rise across the training size with the best value $C = 100$ at the edge of the grid. That means the chosen C might not good enough. The best C will be something bigger so I need to extend the grid even further. In addition, in Fig 8c, train curve keeps rising with larger gamma meaning model starts capturing more complex nonlinear relationships in training set while the validation curve dropping after $\gamma = 0.1$. It's because the decision boundary becomes localized and begins overfitting. Therefore, $\gamma = 0.1$ provides the best balance among the chosen hyperparameters.

TABLE V: Per-Class Performance - Linear SVM

Class	Precision	Recall	F1
Spruce/Fir	0.677	0.670	0.674
Lodgepole Pine	0.778	0.529	0.630
Ponderosa Pine	0.674	0.646	0.659
Cottonwood/Willow	0.318	0.737	0.444
Aspen	0.112	0.754	0.194
Douglas-fir	0.359	0.667	0.466
Krummholz	0.427	0.872	0.573

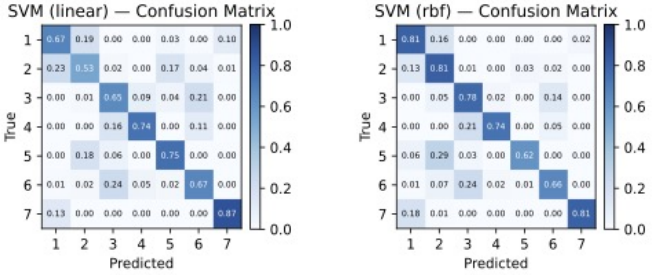
Looking at per-class metrics for linear SVM, F1 score is low compared to other algorithms in general. For minority classes: Cottonwood/Willow, Aspen, Douglas-fir and Krummholz, the recall is high because of the use of balanced class weights. This model becomes aggressive at predicting minority classes that increases recall but the precision is decreases as consequences because of false positives.

TABLE VI: Per-Class Performance - RBF SVM

Class	Precision	Recall	F1
Spruce/Fir	0.801	0.806	0.804
Lodgepole Pine	0.850	0.814	0.832
Ponderosa Pine	0.793	0.780	0.787
Cottonwood/Willow	0.609	0.737	0.667
Aspen	0.408	0.615	0.491
Douglas-fir	0.545	0.658	0.596
Krummholz	0.735	0.805	0.770

Now compare to RBF, the per-class results demonstrate a substantial advantage of the RBF kernel over the linear kernel. While the recall is still high even for minority classes, this model still maintains competitive precisions. The RBF kernel improved performance by effectively learning nonlinear class boundaries.

Fig 9a shows the confusion matrix for linear SVM which decreases the scores in the diagonal across almost every class. Even class 1 and 2 have been impacted because of balanced class weights. The high recall is simply false positives and not reliable at all. The RBF matrix in Fig 9b tightens the diagonal on 5 of 7 classes. Similarly to other algorithms, class 5 and 6 are still challenging, but the overall performance has improved clearly.



(a) CM - Linear SVM (b) CM - RBF SVM

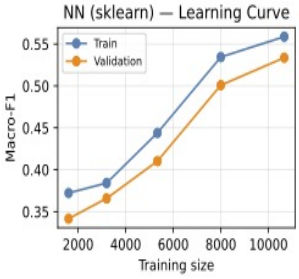
Fig. 9: Confusion Matrix for SVM

D. Multilayer Perceptron (Scikit-learn, SGD only)

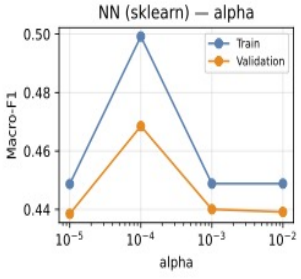
For Scikit-learn implementation, I use SGD solver, ReLU activation, constant learning rate with max iteration = 300 and early stoping, no momentum and no Nesterov. The grid covers the 3 parameters, 4 `alpha` (L_2) values, and 3 learning rates. The detailed parameters will be summarized in next section. The best configuration was $(64, 64)$, $\alpha = 10^{-4}$, lr$= 0.1$: CV Macro-F1 $= 0.572$; test Macro-F1 $= 0.492$, balanced accuracy $= 0.480$, accuracy $= 0.743$. Among the parameters, GridSearch CV selected $(64, 64)$ for both MLPs. Moderate depth outperformed both single wide layer (128) and the deeper narrow network $(48, 48, 48)$ under the SGD-only constraint. The model trains for 39 epochs before early stopping kicks in.

The learning curve in Fig 10a indicates that both training and validation Macro-F1 continue to increase and stay close together as more training data are introduced. That means

the model is not overfitting and the network will continue to benefit from additional data. The complexity curve shows the peak α is at $10^{-}4$ then it decreases and becomes flat, so regularization is not what is limiting this model.



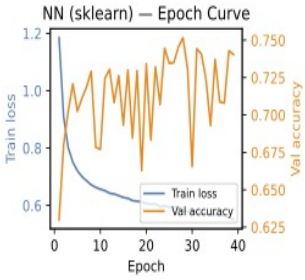
(a) MLP SKLearn LC



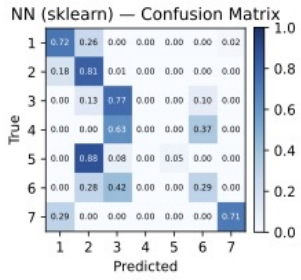
(b) MLP SKLearn MCC - α

Fig. 10: Learning Curve and MCC for MLP Scikit-Learn

In Fig 11a for Epoch curve, training loss decreases from about 1.2 to 0.60 meaning the network is learning. However, validation accuracy fluctuates after 10 epochs because I used pure SGD so any updates are noisy. After 39 epochs, Macro-F1 lands well below the RBF SVM. Early stopping matters here because the validation accuracy stops improving after 40 epochs which triggers stopping.



(a) SKLearn Epoch



(b) SKLearn Confusion Matrix

Fig. 11: Epoch Curve and Confusion Matrix for MLP SKLearn

Per-metric classes in Table VII reveal a strong imbalance in the MLP performance. Majority classes achieved high score while minority classes were completely failed. Cottonwood/Willow was good with other models but was never predicted here resulting in 0 precision, recall and F1. Aspen achieved perfect precision but extremely low recall and F1. It indicates that the model only predicted Aspen in a high confident cases while failing to identify most true instances. This observation aligns with the confusion matrix in Fig 11b. Balanced accuracy is very low (0.480). The small groups collapse onto larger neighbors. A perfect diagonal is no longer valid in this implementation, so the SGD optimizer only learns the majority classes

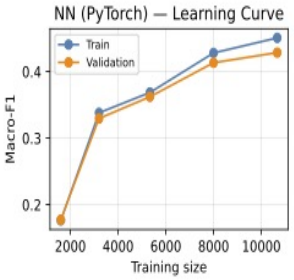
E. Multilayer Perceptron (PyTorch, SGD only

The second MLP model is implemented with PyTorch and same parameters. Manual early stopping on validation

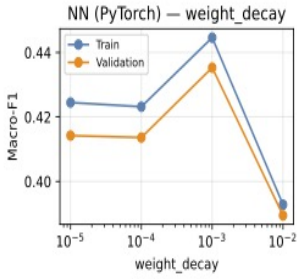
TABLE VII: Per-Class Performance - MLP SKLearn

Class	Precision	Recall	F1
Spruce/Fir	0.732	0.724	0.728
Lodgepole Pine	0.762	0.814	0.787
Ponderosa Pine	0.709	0.772	0.740
Cottonwood/Willow	0.0	0.0	0.0
Aspen	1.0	0.046	0.088
Douglas-fir	0.486	0.292	0.365
Krummholz	0.769	0.709	0.738

accuracy with $patience = 10$. The best configuration was $(64, 64)$, $weight\_decay = 10^{-4}$, lr$= 0.1$: CV Macro-F1 $= 0.459$; test Macro-F1 $= 0.432$, balanced accuracy $= 0.428$, accuracy $= 0.736$ in 32 epochs.



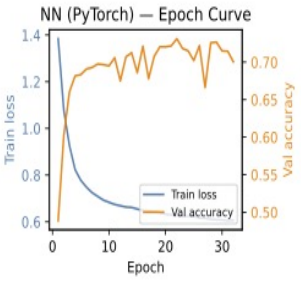
(a) MLP PyTorch LC



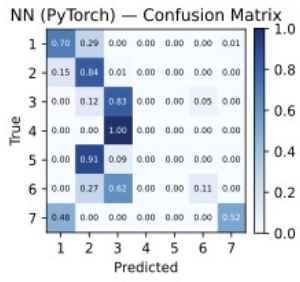
(b) MLP PyTorch MCC

Fig. 12: Learning Curve and MCC for MLP PyTorch

Comparing Fig 12 of PyTorch and Fig 10 of Scikit-Learn, their behaviors are almost the same under SGD constraint. Learning curve is increased for both training and validation but F1 accuracy is low. PyTorch uses $weigh\_decay$ but it's similar to α. Although, the peak is at 10^{-3}, the final model still uses $weigh\_decay = 10^{-4}$ because the joint GridSearchCV also combines with learning rate and architecture over its own folds. Epoch curve in Fig 13a shows training loss falling smoothly while validation accuracy climbs to 0.70 and less fluctuate compared to Scikit-learn.



(a) PyTorch Epoch



(b) PyTorch Confusion Matrix

Fig. 13: Epoch Curve and Confusion Matrix for MLP PyTorch

Per-class metrics are even worse with lower performance in almost every classes. PyTorch additionally hits another 0 on Aspen. In confusion matrix, 3 minority classes are almost

TABLE VIII: Per-Class Performance - MLP PyTorch

Class	Precision	Recall	F1
Spruce/Fir	0.742	0.703	0.722
Lodgepole Pine	0.752	0.835	0.792
Ponderosa Pine	0.617	0.825	0.706
Cottonwood/Willow	0.0	0.0	0.0
Aspen	0.0	0.0	0.0
Douglas-fir	0.394	0.108	0.170
Krummholz	0.813	0.525	0.638

absorbed by larger neighbors, which drives the lowest balanced accuracy 0.428 and confirms the gap is the shared SGD-only constraint, not the implementation.

V. Cross-Model Discussion

A. Final Test Metrics

TABLE IX: Final Test Metrics and Ranking

Model	Acc	Macro-F1	Balanced Acc	Fit(s)	Pred(s)
DT	0.756	0.674	0.672	0.13	0.001
KNN	0.800	0.680	0.658	0.00	0.206
SVM-LIN	0.609	0.520	0.697	26.65	1.002
SVM-RBF	**0.801**	**0.707**	**0.746**	4.17	2.075
MLP-SKL	0.743	0.492	0.480	1.01	0.003
MLP-PYT	0.736	0.432	0.428	0.34	0.003

All fit/predict times are wall-clock on a Macbook Pro M4 (10 cores), 16 GB RAM, macOS 26.1, Python 3.12.7. PyTorch ran on CPU. By *accuracy*: RBF-SVM (0.801) $\gtrsim$ kNN (0.800) $>$ DT (0.756) $\approx$ MLP-skl (0.743) $>$ MLP-pyt (0.736) $>$ Linear-SVM (0.609). By *Macro-F1*: RBF-SVM (0.707) $>$ kNN (0.680) $\approx$ DT (0.674) $>$ Linear-SVM (0.520) $>$ MLP-skl (0.492) $>$ MLP-pyt (0.432). By *balanced accuracy*: RBF-SVM (0.746) $>$ Linear-SVM (0.697) $>$ DT (0.672) $>$ kNN (0.658) $>$ MLP-skl (0.480) $>$ MLP-pyt (0.428).

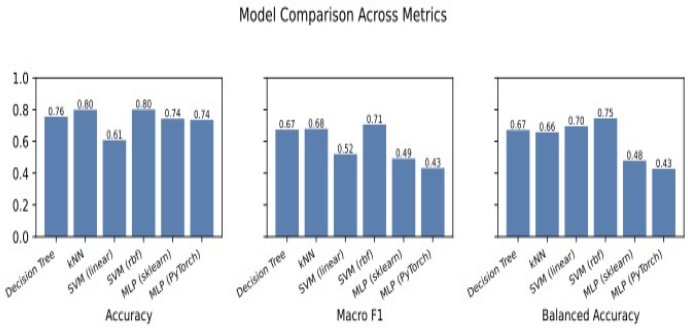


Fig. 14: Model Comparison

The RBF SVM achieved the best overall performance. KNN is about the same in accuracy but slightly lower in F1 score. Since Macrof-F1 is more reliable for Covertype dataset, I will use it to compare with my hypothesis. My hypothesis was MLP $\approx$ RBF SVM $>$ DT $>$ KNN $>$ Linear SVM but the actual result was RBF SVM $>$ KNN $\approx$ DT $>$ Linear SVM $>$ MLP. It's still partially correct. RBF SVM is the top model as expected. This dataset has some thresholds that could benefit DT and in fact DT is where I predict. I expected MLP should be the top model since Neural Networks are powerful model but it ended up the worst model, even lower than linear SVM. The reason is because SGD algorithm was likely getting stuck. Without momentum, adaptive optimizer, this lowers the overall performance. Lastly, I did not expect KNN is an effective algorithm because of the curse of dimensionality but the 54D space turned out to be tractable enough that Manhattan distance KNN essentially ties the DT on Macro-F1.

B. Trade-offs and Class-level Behavior

The biggest single error is between class 1 and 2. Every model confuses them because their elevation, slope, and aspect distributions overlap. Those 2 classes are also account for most of the correct predictions. Correct predictions on class 1 and 2 can increase the accuracy but it's not the whole story. Neural Networks have very high accuracy but failed on minority classes so the overall performances are poor. The best model is RBF SVM but it also consumes a lot of resources to run. DT is extremely efficient and the overall performance is not bad. This could be a candidate to run the entire dataset. Neural Networks provided fast inference but both models completely failed on small groups so they're not reliable.

C. Curve-based Interpretation, Limitations, and Improvements

The details for each algorithm and curve-based reasoning have been discussed in Experiments Section. Yet, for summary purpose, the learning curves split the models into 2 parts. DT and KNN show training curves at 1 implying overfitting with a slowly rising validation curve. However, it's due to the hyperparameter tuning so unpruned tree and KNN with distance weight are selected. The RBF SVM shows train and validation curves that rise healthy without overfitting or underfitting. Both MLPs show tightly coupled curves with the bias coming from the SGD constraint.

Apart from imbalance dataset issue, both SVM kernels could be improved by expanding grid search. The maximum tested $C = 100$ suggesting that SVM may benefit by larger values. Keskar et al in their paper at ICLR 2017 [4] suggested that using Adam would probably close the gap between the MLPs and the RBF SVM. Therefore, relaxing the SGD constraint is a promising direction for further improving performance.

VI. Extra Credit

Using the same architecture $(64, 64)$, learning rate (0.05), `weight_decay` (10^{-4}), batch size (256), seed (85), and patience (20), I evaluate 4 activation functions: ReLU, GELU, SiLU/Swish and tanh. The final metrics are as follows:

The activation study shows that activation choice has a significant impact to neural network performance. ReLU has the highest accuracy but lowest macro-f1 meaning it failed to predict minority classes. On the other hand, GELU, SiLU and tanh have higher macro-f1 and stay close to each other. This result suggests that smoother, zero-centered activation

TABLE X: Hyperparameter Search Grids

Model	Grid	CV
DT	`ccp_alpha` $\in \{0, 10^{-5}, 10^{-4}, 5{\times}10^{-4}, 10^{-3}, 5{\times}10^{-3}, 10^{-2}\}$; `max_depth` $\in \{5, 10, 15, 20, 30, \text{None}\}$	5-fold
kNN	$k \in \{1, 5, 15, 31, 75\}$; `metric` $\in$ {Euclidean, Manhattan}; `weights` $\in$ {uniform, distance}	5-fold
SVM-LIN	$C \in \{0.1, 1, 10, 100\}$	3-fold
SVM-RBF	$C \in \{0.1, 1, 10, 100\}$; $\gamma \in \{\texttt{scale}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$	3-fold
MLP-SKL	arch $\in \{(128), (64, 64), (48, 48, 48)\}$; $\alpha \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$; lr $\in \{10^{-3}, 10^{-2}, 10^{-1}\}$; batch $= 256$	3-fold
MLP-PYT	same arch; lr $\in \{10^{-3}, 10^{-2}, 10^{-1}\}$; `weight_decay` $\in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$	3-fold

TABLE XI: Activation Study

Activation	Acc	Macro-F1	Balanced Acc	Epochs	Fit(s)
ReLU	0.751	0.487	0.470	136	1.53
GELU	0.738	0.528	0.508	150	2.32
SiLU	0.734	0.541	0.516	158	1.98
tanh	0.751	**0.551**	**0.531**	184	2.06

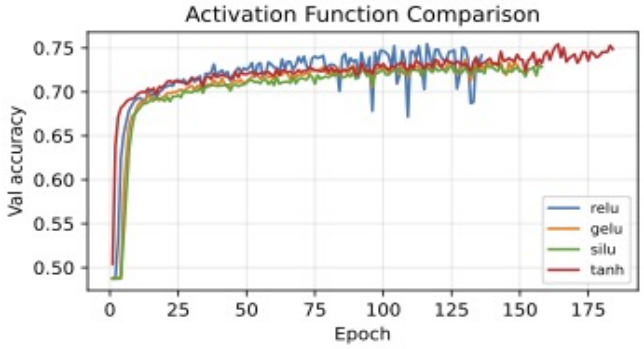


Fig. 15: Activation Comparison

functions provide better predictions for plain SGD. In addition, ReLU converges fastest with only 136 epochs because it's piecewise-linear form with simple gradients while tanh requires 184 epochs. Tanh reaches a better solution but requires more optimization steps. GELU and SiLU stay in the middle, offering improved minority-class performance without too much trade-offs.

VII. Conclusion

The confusion matrices revealed both error types throughout the experiments. Class 5 was frequently predicted as its neighborhoods resulting in a high number of false negatives while it creates false positives for the classes receiving those predictions. These errors explain why the models usually have very high accuracy but low Macro-F1 and balanced accuracy in general.

Although I was thinking Neural Networks should be the optimal solution, the actual experiment did not show that. Yet RBF SVM is still my correct choice and it is the model I would defend as the final choice. It has highest Macro-F1 and balanced accuracy. It also wins on the 2 hardest classes (5 and 6). However, the most important caveat is the prediction latency which would be a major constraint for any testing on a full dataset. With a very imbalanced dataset like Covertype, I think all goods model will have the same latency issue. Even if the neural networks are allowed to use other optimizations and no other constraint or Random Forest algorithm, to fully tackle this dataset is not a simple task and definitely requires some trade-offs.

This experiment teaches me that no single metric or model is sufficient to characterize model performance. Learning curves, complexity curves, confusion matrices, runtime, class metrics demonstrate different aspects of bias, variance, regularization and optimization behavior. In fact, most real world datasets are imbalance so future researchers or practitioners should be cautious about selecting models and should consider multiple metrics during evaluation process. On top of it, understand the data and come up with a good strategy on data pre-processing is a must. If we can clean up, transform or resample the data properly, it would be helpful on model development.

AI USE STATEMENT

I used ChatGPT, Claude and Windsurf IDE to brainstorm the project plan, generate/ debug/ refactor code and OverLeaf AI suggestions for Latex syntax. I also used ChatGPT to learn more about the concepts and Python libraries. The final design, workflow and analysis are my own. I reviewed, verified, and understand all assisted content.

References

[1] I. Belcic, C. Stryker, "What is supervised learning?" IBM, 2026.
[2] J. Blackard "Covertype" UCI Machine Learning Repository.
[3] C. C. Aggarwal, A. Hinneburg, and D. A. Keim "On the surprising behavior of distance metrics in high dimensional space" in Proc. 8th Int. Conf. Database Theory (ICDT), London, UK, Jan. 2001
[4] N. S. Keskar, D. Mudigere, J. Nocedal, M.Smelyanskiy, and P. Tang "On Large-Batch Training for DEEP Learning: Generalization Gap and Sharp Minima" in Proc. 5th Int. Conf. Learning Representations (ICLR), Toulon, France, Apr. 2017.