COMP6208 Group Project: Forest Cover Type Prediction

Nikolaos Chrysovalantis Tsagkopoulos, Subhasish Ghosh, Jingxuan Su and Lingeshwaran Kanniappan

***Abstract*— In this study we analyze the tabular Forest Cover Type Prediction dataset available as a competition from Kaggle. We first visualize the dataset and its feature correlations, apply different preprocessing techniques and evaluate on specific models. We then apply feature engineering, and feature reduction techniques to improve the model prediction accuracies. Finally, we present metrics and analysis of the evaluated models and the challenges we faced during optimisation. In conclusion, we state our Kaggle score and present the models which performed best.**

I. INTRODUCTION

The Forest Cover Type represents cartographic data of 7 forest types from the Roosevelt National Forest of northern Colorado, US. This dataset was made available from the US forest service inventory system for a 30x30 meter cell. The forest cover types presented here, are a result of natural ecological evolution of the flora and fauna of the forest rather than the result of forest management processes.

The dataset consists of 55 features, both numerical and binary categorical data. The numerical type features consist of countable attributes like elevation, aspect, slope etc., while the categorical ones consist of qualitative independent variables, that is wilderness area and soil type which describe the fauna and the flora of each forest patch. The data is in its raw form i.e. not scaled, there were no missing data values apart from the features soil_type7 and soil_type15 which were empty, so they have been removed.

Further, the dataset was made available as test and train csv files, having the test file unlabelled. The testing set consisted of 565,892 samples while the training set only 15,120, meaning that the training set was 2% of the testing set.

II. DATA EXPLANATORY STUDIES

A. Data Correlation

Figure 1, shows the heat map for the numerical data set. It can be clearly observed that certain feature pairs are corelated (positively or negatively), while others are independent. For example, Aspect and Hillshade 3pm are highly correlated while Hillshade and 'Distance to Hydrology features' are independent. This information can be utillized for feature selection, as correlation can be problematic in the learning phase of a model.

To build efficient machine learning models, its important to understand hidden patterns in the dataset. For example, in this section we show how data visualization can augment the understanding derived from the heat map.

From the correlation matrix (Figure 1) and data visualization (Figure 2) we can further verify the following conclusions:

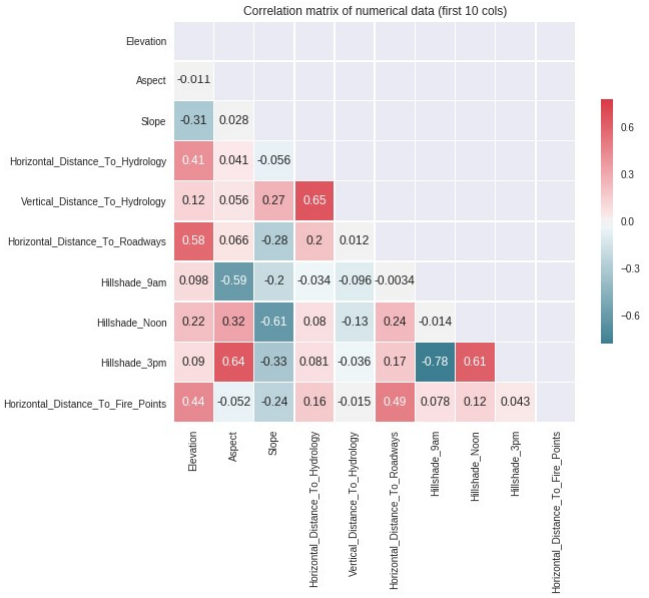


Fig. 1. Heat Map of Feature Set

- Hillshade 3pm and Hillshade 9am are highly correlated and this is rational since hill shade its a function between a place and suns position. It can be verified also from the ellipsoidal graph in Figure 2. Vertical Distance To Hydrology[1] and Horizontal Distance To Hydrology[2] are highly correlated, and this depends from various factors rather than being a general truth. It could be the case that in the particular forest, a lake or a river implies also an aquifer. The data visualization plot does not reveal any strict underlying function but makes it clear that the closer one gets to a water body, underground water is expected to be found.
- Hillshade 9am and Aspect reveal a sinusoidal pattern, and this makes perfect sense since shadows depend from the combination of the suns position and azimuth.
- Hillshade 3pm and Hillshade Noon are correlated the same way as Hillshade 3pm and Hillshade 9am do.
- Hillshade Noon and Slope indicates that the sun is brighter at noon for patches with soft slope rather than those with steep slope.
- Hillshade 9am and Aspect appear to have the same pattern as Aspect has with Hillshade 3pm with a phase difference.

[1]Horizontal distance to water features. (e.g. lake, river)
[2]Vertical distance to water features.

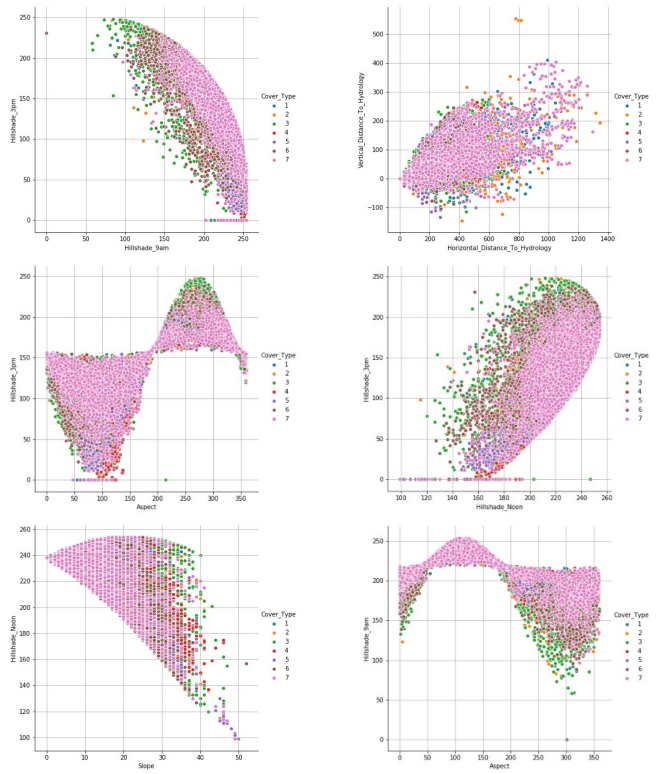


Fig. 2. Highly correlated pairs

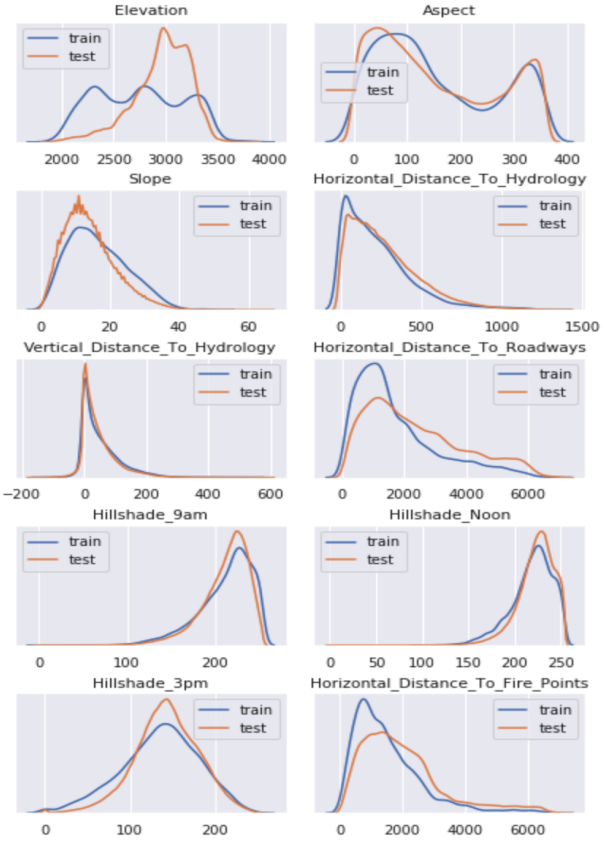


Fig. 3. KDE plots comparing train and test data

B. *Probability Density Functions*

It can be seen from Figure 3, that the probability density functions for numerical features are very similar between train and test data, apart from 'Elevation' which values show that are concentrated around 3000 in testing data.

Figure 4, shows that 'Elevation' is the most discriminating feature, thus variation between test and train set might affect accuracy.

III. Pre-processing

Dataset preprocessing is one of the critical steps in machine learning. If dataset collection is not well controlled, this can result into out-of-range and impossible data combinations. This can lead to the garbage-in-garbage-out idiom, where the learnings performed by the machine becomes irrelevant. Further, many standard objective functions (such as regularizers and RBF kernels) assume the data to be mean/variance centered. If there are imbalances between the feature sets, this can result in models which are biased towards specific features while ignoring others altogether.

The following were the key pre-processing techniques applied to the dataset.

A. *Dataset Inspection Cleansing*

Dataset inspection and cleansing are the basic steps taken in detecting and correcting corrupt data.

In this step a high-level inspection of the dataset is performed. This can detect basic issues with the dataset and

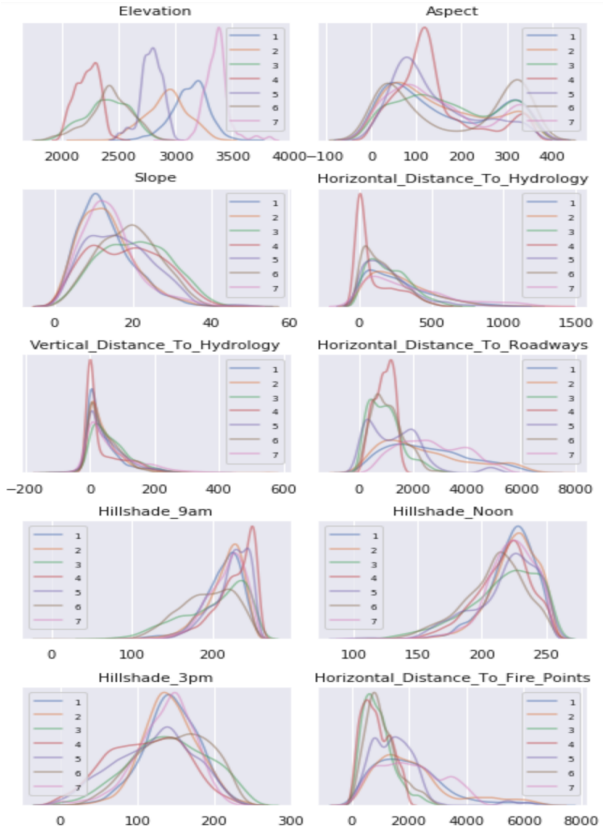


Fig. 4. KDE plots comparing features for every class

helps plan further preprocessing steps. For this dataset, the following observations were made:

- The dataset was in raw form.
- It contained both categorical and numerical data.
- The training dataset was only about 2.6% of the test set.
- There were no intermittent missing values.
- The training dataset categories were equally balanced across all classes.
- There were two feature columns which were empty ('Soil type 7' and 'Soil type 15').

B. Standardisation, Normalisation, MinMaxation and Log-Transformation

As a first step of pre-processing, the dataset was split into numerical and categorical types and three different pre-processing techniques were applied.

In standardisation, the dataset was mapped to zero mean and unit variance. The centering and scaling were done independently for each feature.

In normalisation, the samples were mapped to unit norm, i.e. each row of the dataset was rescaled independently to unit norm.

In case if MinMax scaling, each feature is scaled to a specified range e.g., between 0 and 1. The transformation can be represented as follows:

$$Xstd = \frac{(X - X.min(axis = 0))}{(X.max(axis = 0) - X.min(axis = 0))} \quad (1)$$

$$Xscaled = Xstd * (max - min) + min \quad (2)$$

Log transformation is used to evolve the features to incorporate the skewness in the dataset. This unbiases the large variances in the features and brings uniformity. The general formula used to evaluate the log transform is shown in the following equations.

$$log_{10}(\sigma + value), skew \geq 0, \sigma = max(value) \quad (3)$$

$$log_{10}(value - \sigma), skew \leq 0, \sigma = min(value) \quad (4)$$

Further, we observed that our training dataset was about 2.6% of the training data. This would affect the statistical properties based upon which we pre-process the data. For example, the mean and variance of the train dataset would not reflect that of the complete dataset. Hence, we extracted mean and variance from the test dataset and standardized the training dataset.

IV. Feature Selection Engineering

Feature selection is the process of pruning the features of a dataset to either improve the models accuracy or increase training speed for very high dimensional data. On the other hand, feature engineering, is the manual process of adding new features to a dataset from domain understanding of the dataset[3].

A. Automatic Feature Reduction

Automatic variance-based feature selection can operate based on either the columns/features or the rows/samples of a dataset.

Statistical techniques such as PCA, LDA or QDA operate based on variance across the columns/features. However, if PCA is applied to classification, then it may cause the classes to overlap depending upon the dataset. This resulted in poor prediction accuracy. LDA and QDA yielded satisfactorily results.

Another approach to automatic feature selection is to observe the variance of data across rows/samples. This helps prune features based upon a threshold variance criterion. For example, consider a feature having only 1 in every row, this would cause a classifier to confuse the classes [3]. Thus, removing such features can improve the accuracy of the models.

Figure 5, shows the feature importance selection for the dataset. The scores were derived from chi function, as it yielded the best results comparing with other score functions. As we shall show in the following sections, even with a grid search over the suitable number of features starting from those with the highest score, some algorithms were improved while others were indifferent.

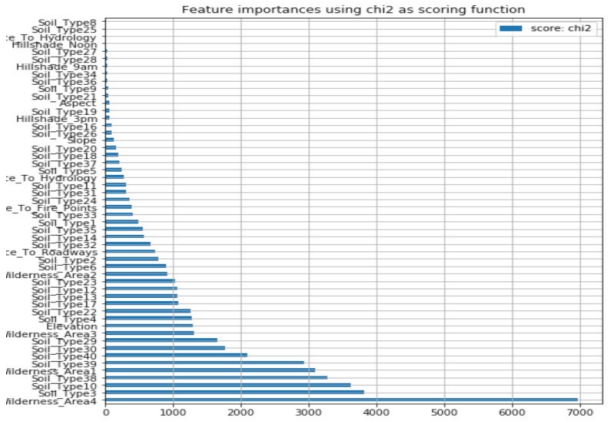


Fig. 5. Feature importances using chi2 function

In Figure 5, it can be seen that a specific fauna type has more than double score comparing the others. Also, as expected flora types are very important in the determination of a type of a forest patch.

B. From One-hot to vector embeddings

We implemented also an autoencoder to transform the categorical one-hot encoded data to vector dimension embeddings to reduce the sparsity in a try to get better performance.

[3]It turned out that this method did not bring significant result changes.

C. Manual Feature Engineering

In feature engineering, domain knowledge of the data is utilised to create new features that can make the machine learning algorithm perform better. This process is both expensive and time consuming and requires iteratively experimenting with various features and evaluating performance. We created the followed features:

$$dist\_to\_hydrology = \sqrt{H^2 + V^2} \tag{5}$$

where H and V are the horizontal and vertical distance to hydrology. We took this feature to see if there is a direct relation between these.

$$distance = \frac{Hydr + Road + Fire}{3} \tag{6}$$

where distance is the average (horizontal and vertical respectiveley) distance from hydrology, roadways and firepoints. Same for the average hill shade:

$$avg.hillshade = \frac{3pm + Noon + 9am}{3} \tag{7}$$

Creating new features for highly negative correlated pairs:

$$A\_H = \frac{Aspect \times Hillshade\_9am}{255} \tag{8}$$

$$S\_H = \frac{Slope \times Hillshade\_Noon}{255} \tag{9}$$

V. Data Augmentation

Data size is the biggest problem in machine learning, the more the better. People usually try to enlarge their dataset by adding modifications of the current ones. This is not easy with data in tabular form, so it was easier to enlarge our dataset by adding more original data which were found on the UCI repository[4] [2]. This provided us the opportunity to train and evaluate all our models against a slightly bigger dataset and see if the number of samples affect the performance of some models in this particular problem.

VI. Handling Categorical Data

In the forest cover type dataset, two variants of categorical data were present, wilderness_area and soil_type. Both variants were one-hot-encoded. Further, these columns were separated out during standardization, normalization operations to avoid corrupting the encodings.

VII. Model Analysis

Based upon the data analysis conducted hitherto, few models were identified which were expected to perform better. For example, Decision trees, Ensembles and Voting classifiers are expected to perform better for categorical and mixture data. Further, in [1] it was observed that neural networks are expected to perform well. Hence, as a part of our study, we evaluated the following models:

- **Decision Trees:** using gini criterion and no maximum depth.
- **Random Forest:** using gini criterion with 200 estimators and a maximum depth of 26.
- **Extra Trees:** using gini criterion with a maximum depth of 45.
- **AdaBoost:** with 100 estimators, a learning rate of 0.6 and 'SAMME' algorithm
- **KNN:** using only 1 neighbour.
- **MLP:** with 2 hidden layers of 100 neurons and ADAM optimiser
- **Voting:** using as individual classifiers the best models: Random Forests, KNN and Decision Trees with soft voting.
- **SVM:** with $C = 1$ and $\gamma = 100$ with RBF kernel.
- **LDA**
- **QDA**
- **XGBoost**

We evaluate our models under the 4 preprocessing techniques (Original, Standardized, MinMaxed, Normalized and Log transformation[5]) under:

- full features
- with reduced features as chi2 score function indicates
- with artificially created features
- with full features after changing the encoding of the categorical data (from 44d One-hot to 5d vector embeddings)

A. Model Analysis with all features

Figure 6, shows the baseline accuracy metrics with all features, models and preprocessing for 6-fold validation. Figure 7 shows the confusion matrices for the same. The following observations were made:

- While there is confusion between some classes, the biggest problem seems to be in class 1 and 2.
- For MLP, the confusion between class 1 and 2 was the lowest.
- The confusion between class 1 and 2 was highest for SVM, while in other classes performed better than the other models.
- For Minmaxed and Normalized dataset, the accuracies are approximately at 0.75%.
- For SVMs the accuracies for standardized and original dataset is below 0.2, however shows better performance for Minmaxed and Normalized data.
- Voting classifier provides the best accuracies irrespective of any preprocessing technique.
- XGBoost is most affected by preprocessing technique used. For Normalized dataset the accuracies drop from 0.71 to 0.56 range.
- QDA classifier, surprisingly, performed much worse than LDA classifier.
- SVM showed that preprocessing matters. The accuracies for standardized and original dataset was below 0.2,

[4]UCIs dataset was later found that is a labelled combination of our training and testing dataset provided from Kaggle, so we did not use it in the original model evaluation for the competition.

[5]Log's transformation results are in separate final plot evaluated under the best models

while for MinMaxed and normalized data performed well. We performed a grid search for RBF parameters.

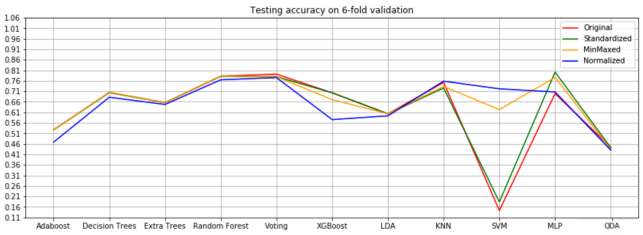
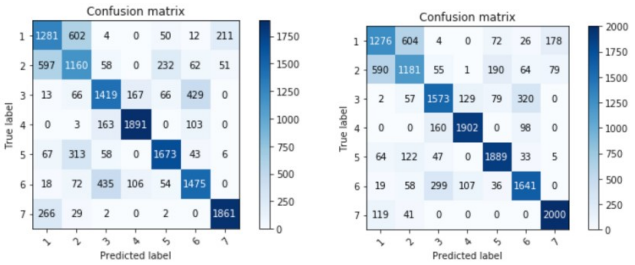
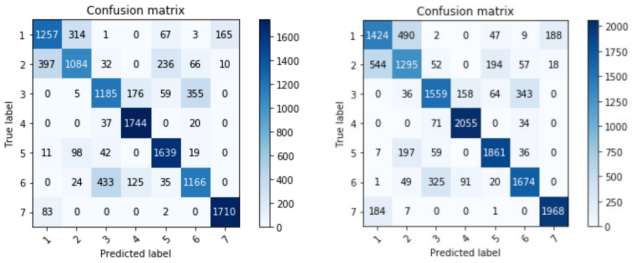


Fig. 6. Model Accuracies



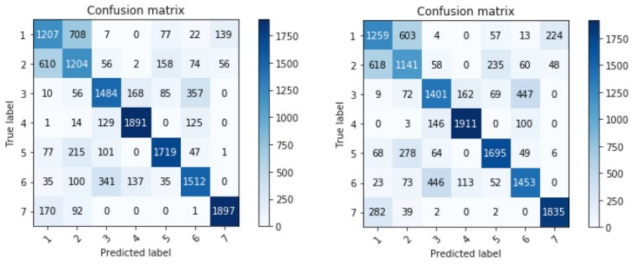
(a) Decision Trees

(b) KNN



(c) MLP

(d) Random Forest



(e) SVM

(f) Voting

Fig. 7. Confusion matrices

B. Model Analysis after log transformation

In Figure 8, we see that (ignoring that the models are with different order in x axis) results are very similar. Apart from SVM's results which are really bad.

C. Model Analysis with feature reduction

Figure 9, shows the accuracies using features based on their score from Figure 5. Using this technique, we can identify the features which contribute towards the target, above a certain threshold. Following, are the motivations for using this method:

Fig. 8. Accuracy after log transformation

- Reduces overfitting: We observed that our models were overfitting the small training dataset. We were having high training scores, however upon Kaggle submission, our scores were lowered.
- Improves accuracy: This method also may help improve accuracies by removing features with low variances, which might confuse the classifier.

We also performed a grid search and identified the optimal number of features for each model. Overall, we observed nominal improvement in the accuracies. However, seems that Adaboost can achieve the same results with only 14 features instead of 52 (this is not an improvement it just did not work well with full features), SVM with 33 features had an improvement of 30% in the accuracy of the standardized data, and in general seems that the majority of the models could achieve same (or slightly) better performance using around 43 of the features.

D. Model Analysis after changing categorical encoding

As part of this attempt, we encoded the 44 dimensional categorical data to a 5d latent vector. The motivations for using an Autoencoder was to reduce the sparsity of the binarized categorical data using a neural network and see if the performance improves. Figure 9, shows the accuracies for each model using this approach. Results were not worse than results from the original dataset, but not significant improvement was noticed.

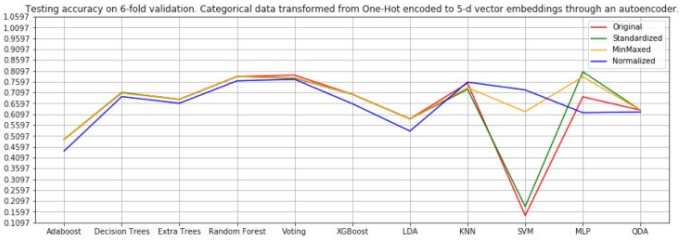


Fig. 9. Results for dense features

E. Model Analysis with enlarged dataset

In our try to enlarge our dataset with more data we came across a very large dataset for forest cover type prediction. Instead of using it immediately and celebrate our 100% Kaggle submission score (some will say cheating), we performed 10-fold cross validation and compare with our small dataset.

Figure 10, shows the observed results. With this approach we were able to achieve over 0.97 accuracy proving that the size of the dataset matters.

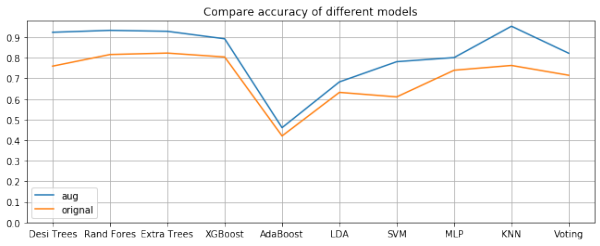


Fig. 10. Results with enlarged dataset

VIII. Best model's discussion

Table 1, shows our final metrics. We observed that without the big dataset, feature engineering with standardized dataset performed best. However, with the augmented dataset, we observed that Neural Networks performed best. This observation was also described in [1]. A combination of our best models (Random Forest, KNN and Decision Trees) yielded the best results individually, and their combination through voting classifier improved their performance by small rate.

TABLE I
Final Metrics

Models	Voting+ Autoencoder	Random Forest+Feature Engineering	Voting+Kbest
Original	**0.7897**	0.7966	**0.7870**
Standardized	0.7755	**0.7968**	0.7749
MinMax	0.7779	0.7961	0.7780
Normal	0.7705	0.7493	0.7646

IX. CONCLUSIONS

In this competition we tried several methods to successively predict forest cover types by some geomorphological features. With a training dataset with a size of 2% of the testing dataset, we achieved an evaluation of 0.80%. We tried various models on Original, Standardized, MinMaxed and Normalized data. We also tried to automatically discard some features while maintaining same or better accuracy. In addition, we transformed our 44d categorical data to 5d vector embeddings to see if sparsity affects the model. In the end we artificially created some features to improve upon these models without particular success. In conclusion, after trying a larger dataset upon the same methods we observed that the accuracy was of 0.97% meaning that the size of the dataset plays a vital role for this problem.

References

[1] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Computers and Electronics in Agriculture*, vol. vol.24, pp. 131–151, 1999.

[2] UCI. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/covertype

[3] sci-kit feature selection. [Online]. Available: https://scikit-learn.org/stable/modules/feature_selection.html