Forest Cover Prediction using Classification Mining Techniques and Deep Learning

Arjun Das (20BDS0129), John Harshith (20BDS0411), Siddharth Pal (20BDS0409)

Guidance Professor: Dr. Arup Ghosh

Fall Semester 2021-22

Abstract

A series of data mining approaches have been performed on the Forest Cover dataset available on UCI Machine Learning Repository. The prime research paper written on the dataset was by Jock A. Blackard and Denis J. Dean who had used Artificial Neural Network and Discriminant Analysis to predict the forest cover type. Due to the archaic technologies of 1999, the prediction percentage was calculated to be 71.1%. We believe that better prediction models can be built using modern machine learning libraries with new and evolved techniques and assessments. Among the classification mining techniques, Gaussian Bayes' model failed to beat the paper while Logistic Regression was at par. Support Vector Classifier, Random Forest ensemble, and K Nearest Neighbors were able to beat the prediction percentage of the research paper. The Deep Learning approach included creating an Artificial Neural Network with Stochastic Gradient Descent Extension (Adam Optimizer) as the learning algorithm which yielded 99.99%.

Introduction

In recent years, there have been great advancements in the field of Machine Learning and Artificial Intelligence, where automation of lengthy, tedious, and manually implemented algorithms has been replaced with powerful and dynamic language libraries that have logic programmed already with easy language syntax, thereby making implementations faster. Data Mining techniques have also seen notable improvements lately that enable people to get a deeper understanding of various

kinds of datasets, and hence infer better-targeted prediction, bringing out several unknown or unobserved interpretations of the data.

Complex datasets that have a considerably huge size and mediocre usability ratings might indicate a possible tendency of mediocrity in the prediction accuracies for the model created based on these datasets. However, it need not be a hard and fast rule and such generalized notions can be claimed incorrectly with the use of the latest advanced computing tools. The project deals with a dataset of a similar description and finds a variety of methods to create suitable prediction models using Classification Mining Techniques and Deep Learning concepts to yield a higher prediction accuracy than the base paper without overfitting the model for higher accuracy.

Literature Survey

The analysis presented by Blackard and Dean[1] compared two alternative techniques for predicting forest cover types from cartographic variables. They evaluated four wilderness areas in the Roosevelt National Forest, located in the Front Range of northern Colorado. Cover type data came from US Forest Service inventory information, while the cartographic variables used to predict cover type consisted of elevation, aspect, and other information derived from standard digital spatial data processed in a geographic information system (GIS). The two alternative techniques were Artificial Neural Network and Discriminant Analysis. The ANN models consistently outperformed the DA models in the prediction of forest cover types. The ANN model was evaluated based both on its absolute accuracy and on its ability relative to a model based on discriminant analysis (DA). In general, the ANN model produced good classification results, with greater accuracy than those produced by the DA model. They achieved an overall best accuracy of 71.1%.

To get an even better prediction accuracy, the application of classification data mining techniques and deep learning on the Roosevelt National Forest Cover Type Dataset will be more authentic as an approach. Moreover, hyperparameter tuning can be done to find out the best performance of the Machine Learning models without underfitting or overfitting them.

Project Objective

In 1999, Jock A. Blackard and Denis J. Dean implemented Artificial Neural Network and Discriminant Analysis to predict the forest cover from various areas of the Roosevelt National Forest in Colorado, USA.

However, we know from his paper that there is only 71.1% prediction accuracy in the model created by them using Artificial Neural Networks on various columns of data. Even though this method was better than previously implemented traditional Discriminant Analysis, our objective in this project is to find solutions to better the accuracy using the exact same dataset by creating various classification models.

This dataset includes information on tree type, shadow coverage, the wilderness of the surrounding, distance to nearby landmarks (roads, etc.), soil type, local topography, etc. The dataset is very large having more than 0.5 million item sets. Its usability is 5.9 according to premiere data science websites.

The above dataset has been taken from the University of California Irvine Machine Learning Repository, and the source can be found here, and the dataset is also present in the main branch of the repository as covtype.csv. Finally, the accuracy and overall prediction performance of the model are gauged for efficient use.

Methodology

1. Understanding the components of the dataset.
2. Creating visuals that will aid us with understanding and enable us to find the proper combination of data columns for high accuracy.
3. Using various classification mining algorithms to create models that yield certain prediction accuracy for a certain combination of data columns.
4. Fitting models appropriately and checking if any model has been overfitted or under fitted.
5. Iterating certain column combinations and algorithms a large number of times to find the best train-test split and record that prediction percentage as well as the split content in a separate binary file for future use.

6. Using Deep Learning techniques on the dataset and yield prediction accuracy.
7. Comparing the performance of all the algorithms and concluding with the most recommended algorithm with this dataset.

Tools Used

To achieve the project objective, we make use of the following tools –

- **Python Language Libraries**
  - Seaborn
  - Matplotlib
  - Scikit-learn
  - Pandas
  - Keras
  - Tensorflow
- **R Language Libraries**
  - Tidyverse
  - Skimr

Implementation

A segment of the visualization has used the R language and its libraries to help us understand the dataset better. The rest of the visualization, classification, and DL analysis has been implemented with the help of Python libraries.

Understanding the Data

There are 13 columns of interest that are included in the dataset:

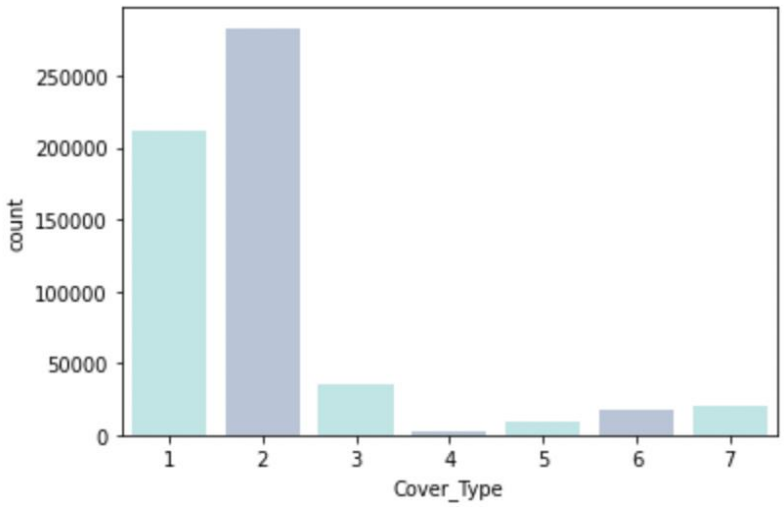
- Cover_Type: One of seven types of tree cover found in the forest. In the data downloaded for the project, the observations are coded 1 through 7. We have renamed them for clarity.

Observations are based on the primary cover type of 30m x 30m areas, as determined by the United States Forest Service. This is our response variable.

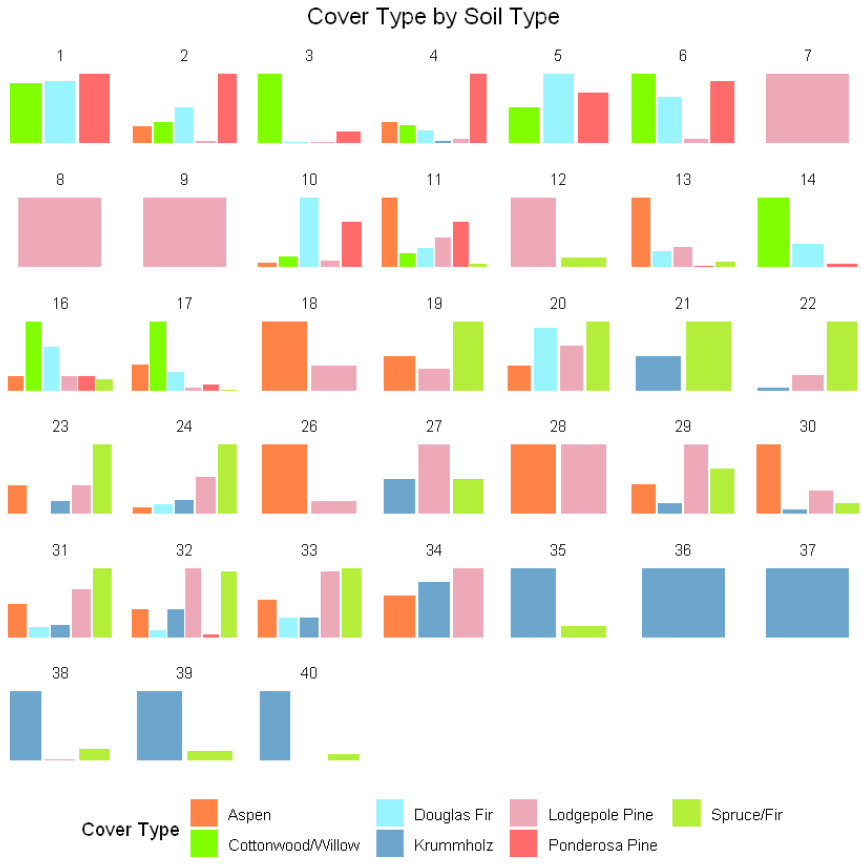
- Wilderness_Area: Of the six wilderness areas in the Roosevelt National Forest, four were used in this dataset. In the original dataset, these were one-hot encoded. We put them in long-form for better visualization and because most machine learning methods will automatically one-hot encode categorical variables for us.
- Soil_Type: 40 soil types were identified in the dataset and more detailed information regarding the types can be found at https://www.kaggle.com/uciml/forest-cover-type-dataset. Similar to Wilderness_Area, Soil_Type was originally one-hot encoded.
- Elevation: The elevation of the observation in meters above sea level.
- Aspect: The aspect of the observation in degrees azimuth.
- Slope: The slope at which the observation is observed in degrees.
- Hillshade_9am: The amount of hillshade for the observation at 09:00 on the summer solstice. This is a value between 0 and 225.
- Hillshade_Noon: The amount of hillshade for the observation at 12:00 on the summer solstice. This is a value between 0 and 225.
- Hillshade_3pm: The amount of hillshade for the observation at 15:00 on the summer solstice. This is a value between 0 and 225.
- Vertical_Distance_To_Hydrology: Vertical distance to the nearest water source in meters. Negative numbers indicate distance below a water source.
- Horizontal_Distance_To_Hydrology: Horizontal distance to the nearest water source in meters.
- Horizontal_Distance_To_Roadways: Horizontal distance to the nearest roadway in meters.
- Horizontal_Distance_To_Fire_Points: Horizontal distance to nearest wildfire ignition point in meters.

Visualization

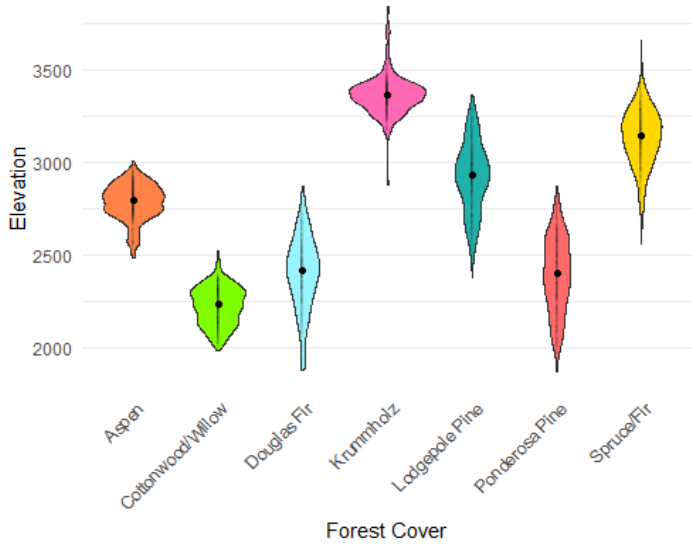
The 7 types of forest cover are very unevenly included in the dataset, as shown in the figure below. This might make learning models difficult, to predict the covers with fewer itemset.



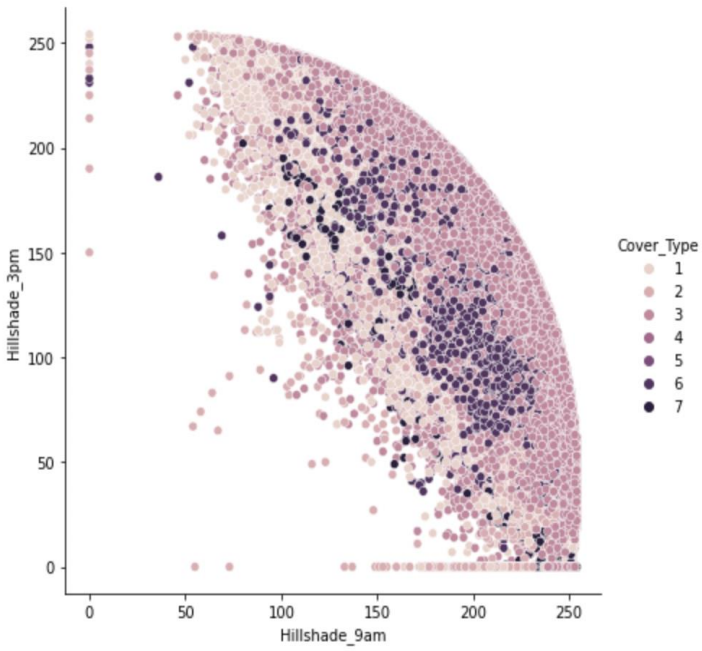
Not all the 40 soil types included in the dataset will be completely helpful for our prediction. From the count plot, we can infer that soil types 11, 16, 31, 32, and 33 will be quite helpful in the prediction-making process because of more representation of forest cover types.



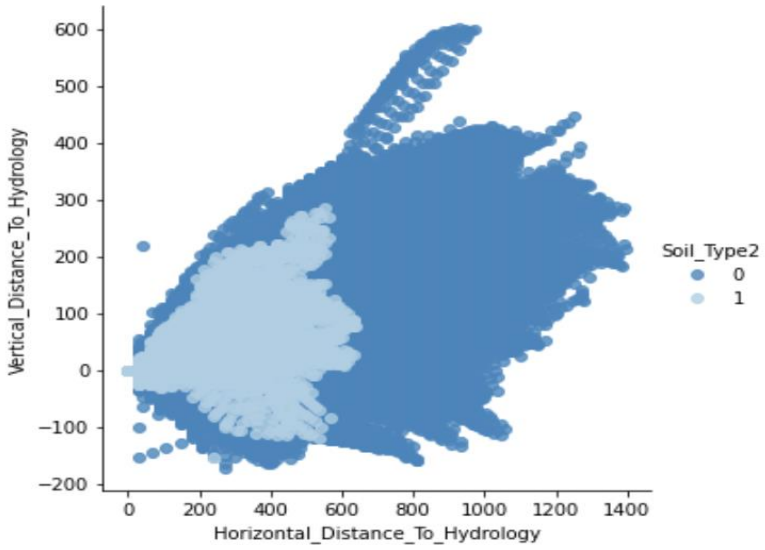
The elevation column might require scaling as we infer from the violin plot given below that the range of elevation is too high when compared with the other column sets, and long ranges of cover points that lie for each cover type.



The plots below show which class a point belongs to when hillshade columns are considered. The class distribution overlaps in the plots. Hillshade patterns give an ellipsoid pattern to each other. Aspect and Hillshades attribute to form a sigmoid pattern if graphed. Horizontal and vertical distance to hydrology will give an almost linear pattern.



From the below scatter plot, we can conclude that the vertical and horizontal distance to hydrology is more spread out when it is not of soil type 2 and is less spread out when it is of the respective type.



Classification Algorithm Determination

There exist various classification mining algorithms that can be used and implemented to form a model that yields very high prediction accuracy. A comparison of various feature selection with various supervised classification learning algorithms has been used and recorded. This tells us about the range of prediction percent one can get while using a particular algorithm (with or without ensemble) for a particular combination of features or data labels.

Gaussian Naive Bayes'

Gaussian Naive Bayes' algorithm has been the least efficient algorithm when it comes to the yield of prediction accuracy. The normal distribution of the continuous variables was not forming well enough to yield a desirably high prediction accuracy, as well as overtake the prediction accuracy as mentioned in the research paper. Therefore, we try other classification algorithms to achieve success.

Logistic Regression

Logistic Regression was the first algorithm to beat the research paper's prediction accuracy for a certain combination of data columns only. This algorithm, when implemented on a huge dataset

makes the prediction very sluggish and it may take hours to run over a single iteration of the algorithm so as to converge the solver.

More information on the algorithm used:

- Penalty type: l2

- Solver: saga

- No class weight, no random state

- Maximum number of iterations taken to converge the solver: 7500

Support Vector Machines

SVM takes the highest time to generate a prediction accuracy for this dataset. Each iteration takes more than 7 hours to run and execute. Such a computationally costly algorithm, although good in prediction, will be highly time-consuming for the users. The prediction accuracy for a particular combination of columns will differ for every variety of train-test split. Therefore, in order to find the highest prediction possible, we iterate the algorithm a few times for a different train-test split and record the highest prediction accuracy's corresponding split in a pickle file for future use.

More information on the algorithm used:

- Kernel used: rbf

- Degree: 3

- Regularization parameter: 1

- No verbosity, no random state

Random Forest

A bagging ensemble technique for the Decision Tree Algorithm, Random Forest generates a constant prediction accuracy at a particular n (number of estimators that gets created) for the column combinations. Therefore, there is no need to store the highest recorded accuracy and its corresponding test-train split in a pickle file. Random forest has also been one of the

computationally cheapest techniques and resulted in one of the highest recorded prediction accuracies for many of the column combinations.

More information on the algorithm used: - Number of estimators - 100 - Node splitting criteria taken - Gini index - No explicit mention of the depth of the tree that is formed - For best split, square root of the features was done instead of taking logarithm - No random state included and no verbosity required

K-Nearest Neighbors

This lazy learner algorithm is highly effective, and better than Random Forest, but it becomes quite computationally expensive when it is run on a dataset of immense size. Due to variations in answers for every train-test split, we had to iterate the algorithm a few times and store the split with the highest accuracy in a pickle. If the time is not an issue, then KNN yields the highest ever recorded prediction percentages.

More information on the algorithm used:

- Number of neighbors set: 3

- Distance set: Minkowski

- Weights set to each point: uniform

Deep Learning

Artificial Neural Networks (ANN) are multi-layer fully connected neural nets and consist of an input layer, multiple hidden layers, and an output layer. ANN is inspired by the design of a human brain and tries to simulate it. An artificial neuron receives a signal then processes it and can signal neurons connected to it. The signal at a connection is a real number, and the output of each neuron is computed by an activation function of the sum of its inputs. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold.

More information on the algorithm used:

- Number of hidden layers: 2

- Learning Algorithm: Stochastic Gradient Descent Extension- Adam Optimizer

- Loss Function: Sparse Categorical Cross Entropy

- Activation function for input layer: Linear (f(x) = x)

- Activation function for the 2 hidden layers: Rectified Linear Activation Unit (RELU)

- Activation function for output layer: Softmax Activation Function

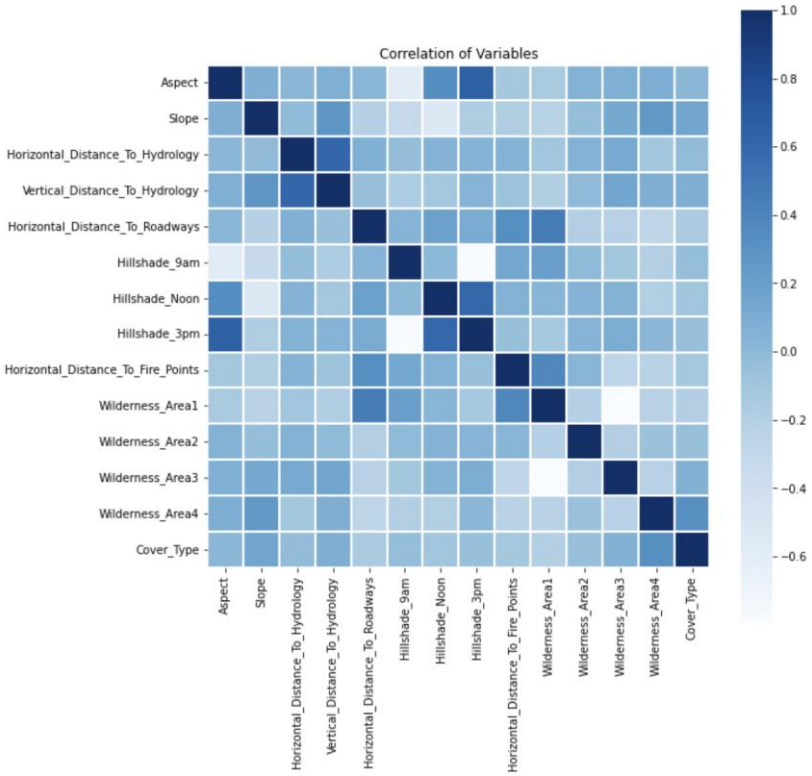
- Number of nodes: 64

- Batch size: 64

- Number of epochs: 8

- Input variables linearly scaled to lie in range [0, 1]

A heatmap to demonstrate the high correlation of columns with the others:



Inference

The classification mining technique algorithms have been run and every prediction accuracy was recorded. The Accuracy Table shows the variation in the performance of every algorithm used for that particular combination of columns.

Columns	KNN	RF	NB	SVM	LR
"Elevation", "Aspect", "Slope", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Hillshade_3pm", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1", “Wilderness_Area2", "Wilderness_Area3", "Wilderness_Area4"	96.900	95.131	55.951	69.238	70.095
"Elevation", "Slope", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Hillshade_3pm", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1", “Wilderness_Area2", "Wilderness_Area3", "Wilderness_Area4"	96.923	95.706	55.848	68.146	70.006
"Elevation", "Slope", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Hillshade_3pm", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1", “Wilderness_Area2", "Wilderness_Area4"	96.917	95.886	56.932	68.301	69.736
"Elevation", "Slope", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Hillshade_3pm", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1", “Wilderness_Area2"	96.955	96.115	59.761	68.372	69.742
"Elevation", "Slope", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1", "Wilderness_Area2"	96.694	96.503	60.136	71.617	66.272
"Elevation", "Slope", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", “Hillshade_9am", "Hillshade_Noon", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1"	96.701	96.319	62.678	71.693	66.319
"Elevation", "Slope", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_Noon", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1"	95.946	95.533	62.678	71.278	63.526
"Elevation", "Slope", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology",	96.215	95.940	62.468	71.104	58.439

"Horizontal_Distance_To_Roadways", "Hillshade_9am", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1"					
"Elevation", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1"	96.563	96.258	63.040	71.676	63.018
"Elevation", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1"	95.115	95.59	63.835	70.646	63.026
"Elevation", "Horizontal_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1"	96.134	96.164	63.669	71.724	62.470
"Horizontal_Distance_To_Hydrology", “Vertical_Distance_To_Hydrology", “Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1"	86.187	90.351	43.435	54.623	50.886
"Elevation", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Wilderness_Area1"	88.627	92.742	64.518	70.064	63.282
"Elevation", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Horizontal_Distance_To_Fire_Points"	96.609	96.019	64.214	72.040	62.337
"Elevation", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Horizontal_Distance_To_Fire_Points", "Soil_Type40"	96.668	95.727	59.964	71.973	62.695
"Elevation", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_3pm", "Hillshade_Noon", "Horizontal_Distance_To_Fire_Points", "Soil_Type40"	96.872	96.035	59.609	71.964	65.269
"Elevation", "Horizontal_Distance_To_Hydrology", "Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_3pm", "Hillshade_Noon", "Horizontal_Distance_To_Fire_Points", "Soil_Type1"	96.823	95.856	64.296	71.746	65.185
"Elevation", "Aspect", "Slope", “Horizontal_Distance_To_Hydrology", “Vertical_Distance_To_Hydrology", "Horizontal_Distance_To_Roadways", "Hillshade_9am", "Hillshade_Noon", "Hillshade_3pm", "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1", "Wilderness_Area2", "Wilderness_Area3", "Wilderness_Area4",	96.948	95.535	46.188	71.459	70.571

"Soil_Type1", "Soil_Type2", "Soil_Type3", "Soil_Type4", "Soil_Type5", "Soil_Type6", "Soil_Type7", "Soil_Type8", "Soil_Type9", "Soil_Type10", "Soil_Type11", "Soil_Type12", "Soil_Type13", "Soil_Type14", "Soil_Type15", "Soil_Type16", "Soil_Type17", "Soil_Type18", "Soil_Type19", "Soil_Type20", "Soil_Type21", "Soil_Type22", "Soil_Type23", "Soil_Type24", "Soil_Type25", "Soil_Type26", "Soil_Type27", "Soil_Type28", "Soil_Type29", "Soil_Type30", "Soil_Type31", "Soil_Type32", "Soil_Type33", "Soil_Type34", "Soil_Type35", "Soil_Type36", "Soil_Type37", "Soil_Type38", "Soil_Type39", "Soil_Type40"					

Deep Learning using ANN on all features	99.99%

Conclusion

The objective of the project was successfully achieved as we found alternative supervised learning approaches that can yield a much higher and more accurate prediction percentage for the given dataset in a reduced time frame, owing to the size of the dataset. The most suitable algorithm would be the K Nearest Neighbors algorithm and the label of interest can be aptly predicted by taking all the columns into account.

This study improves upon the work of Jock A. Blackard and Denis J. Dean in predicting forest covers. After training and testing the same dataset with various classification models, some of the models managed to beat the old accuracy of “71.1%” with a new best accuracy of “97”.

With the optimizer and loss functions used, our deep learning model was able to beat the old accuracy too, with a staggering “99.9%”.

References

- *Blackard, J. A., & Dean, D. J. (1999). Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. Computers and electronics in agriculture, 24(3), 131-151.*
- https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

- https://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
- https://scikitlearn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
- https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Appendix

Classification

```
import pandas as pd
import sklearn
import pickle
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn import preprocessing, svm, metrics

df = pd.read_csv("Forest Cover.csv")
scale = StandardScaler()
le = preprocessing.LabelEncoder()
cov = le.fit_transform(list(df["Cover_Type"]))
y = list(cov)
'''
# x changes because the column value changes for various iterations
x = df[["Elevation", "Aspect", "Slope", "Horizontal_Distance_To_Hydrology",
"Vertical_Distance_To_Hydrology",
        "Horizontal_Distance_To_Roadways", "Hillshade_9am",
"Hillshade_Noon", "Hillshade_3pm",
        "Horizontal_Distance_To_Fire_Points", "Wilderness_Area1",
"Wilderness_Area2",
        "Wilderness_Area3", "Wilderness_Area4"]]
'''
x_train, x_test, y_train, y_test =
sklearn.model_selection.train_test_split(x, y, test_size=0.2)

model = GaussianNB()
model.fit(x_train, y_train)
print("Accuracy using Gaussian Naive Bayes: ", round(model.score(x_test,
```

```
y_test) * 100, 3), "%", sep="")

knnmodel = KNeighborsClassifier(n_neighbors=3)
knnmodel.fit(x_train, y_train)
knnacc = knnmodel.score(x_test, y_test)
print("Accuracy using KNN: ", round(knnacc*100, 3), "%", sep="")

rf = RandomForestClassifier(n_estimators=100)
rf.fit(x_train, y_train)
pred = rf.predict(x_test)
print("Accuracy using Random Forest: ", round(rf.score(x_test,y_test) *
100, 3), "%", sep="")

x_train, x_test, y_train, y_test =
sklearn.model_selection.train_test_split(x, y, test_size=0.2)
lr = LogisticRegression(solver="saga", max_iter=7500)
lrmodel = lr.fit(x_train, y_train)
lracc = lr.score(x_test, y_test)
print("Accuracy using Logistic Regression: ", round(lracc*100, 3), "%",
sep="")

x_train, x_test, y_train, y_test =
sklearn.model_selection.train_test_split(x, y, test_size=0.2)
svmmodel = svm.SVC(kernel="rbf", C=1)
svmmodel.fit(x_train, y_train)
y_pred = svmmodel.predict(x_test)
svmacc = metrics.accuracy_score(y_test, y_pred)
print("Accuracy using SVM: ", round(svmacc*100, 3), "%", sep="")
```

Deep Learning:

```
# imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import tensorflow as tf

dataset = pd.read_csv('../covtype.csv')

X = dataset[dataset.columns[:55]]
scale = StandardScaler()
le = LabelEncoder()
y = le.fit_transform(list(dataset["Cover_Type"]))
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

train_norm = X_train[X_train.columns[0:10]]
test_norm = X_test[X_test.columns[0:10]]

std_scale = StandardScaler().fit(train_norm)

X_train_norm = std_scale.transform(train_norm)
training_norm_col = pd.DataFrame(
    X_train_norm, index=train_norm.index, columns=train_norm.columns)
X_train.update(training_norm_col)

X_test_norm = std_scale.transform(test_norm)
testing_norm_col = pd.DataFrame(
    X_test_norm, index=test_norm.index, columns=test_norm.columns)
X_test.update(testing_norm_col)

cover_model = tf.keras.models.Sequential()

cover_model.add(tf.keras.layers.Dense(
    units=64, activation='relu', input_shape=(X_train.shape[1],)))
cover_model.add(tf.keras.layers.Dense(units=64, activation='relu'))
cover_model.add(tf.keras.layers.Dense(units=8, activation='softmax'))

cover_model.compile(optimizer=tf.optimizers.Adam(
), loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history_cover = cover_model.fit(
    X_train, y_train, epochs=8, batch_size=64, validation_data=(X_test,
y_test))

plt.plot(history_cover.history['accuracy'])
plt.plot(history_cover.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

R Visualisation:

```
library(tidyverse)
library(skimr)

df <- read.csv("C:\\Users\\Arjun\\Downloads\\Datasets\\Forest Cover\\Forest
Cover.csv")
```

```
glimpse(df)
nrow(df) - nrow(distinct(df))
any(is.na(df))
skim(df)
set.seed(1808)

dff <- (df %>%
          group_by(Cover_Type) %>%
          sample_n(size = 1000) %>%
          ungroup() %>%
          gather(Wilderness_Area, Wilderness_Value,
                 Wilderness_Area1:Wilderness_Area4) %>%
          filter(Wilderness_Value >= 1) %>%
          select(-Wilderness_Value) %>%
          mutate(Wilderness_Area = str_extract(Wilderness_Area, '\\d+'),
                 Wilderness_Area = str_replace_all(Wilderness_Area,
                                                   c(`1` = 'Rawah',
                                                     `2` = 'Neota',
                                                     `3` = 'Comanche Peak',
                                                     `4` = 'Cache la
Poudre')),
                 Wilderness_Area = as.factor(Wilderness_Area)) %>%
          gather(Soil_Type, Soil_Value, Soil_Type1:Soil_Type40) %>%
          filter(Soil_Value == 1) %>%
          select(-Soil_Value) %>%
          mutate(Soil_Type = as.factor(str_extract(Soil_Type, '\\d+'))) %>%
          mutate(Cover_Type = str_replace_all(Cover_Type,
                                              c(`1` = 'Spruce/Fir',
                                                `2` = 'Lodgepole Pine',
                                                `3` = 'Ponderosa Pine',
                                                `4` = 'Cottonwood/Willow',
                                                `5` = 'Aspen',
                                                `6` = 'Douglas Fir',
                                                `7` = 'Krummholz')),
                 Cover_Type = as.factor(Cover_Type)) %>%
          select(Cover_Type:Soil_Type, Elevation:Slope,
                 Hillshade_9am:Hillshade_3pm,
Vertical_Distance_To_Hydrology,

Horizontal_Distance_To_Hydrology:Horizontal_Distance_To_Fire_Points))

skim(dff)

palette <- c('sienna1', 'chartreuse', 'lightskyblue1',
             'hotpink', 'mediumturquoise', 'indianred1', 'gold')
ggplot(dff, aes(x = Cover_Type, y = Elevation)) +
  geom_violin(aes(fill = Cover_Type)) +
  geom_point(alpha = 0.01, size = 0.5) +
```

```
  stat_summary(fun = 'median', geom = 'point') +
  labs(x = 'Forest Cover') +
  scale_fill_manual(values = palette) +
  theme_minimal() +
  theme(legend.position = 'none',
        axis.text.x = element_text(angle = 45,
                                   hjust = 1),
        panel.grid.major.x = element_blank())

palette <- c('sienna1', 'chartreuse', 'cadetblue1',
             'skyblue2', 'lightpink', 'indianred1', 'olivedrab1')
ggplot(dff, aes(x = Cover_Type, fill = Cover_Type)) +
  geom_bar() +
  facet_wrap(~reorder(Soil_Type, sort(as.integer(Soil_Type))), scales =
'free') +
  labs(fill = 'Cover Type', title = 'Cover Type by Soil Type') +
  scale_fill_manual(values = palette) +
  theme_minimal() +
  theme(legend.position = 'bottom',
        plot.title = element_text(hjust = 0.5),
        axis.title = element_blank(),
        axis.text = element_blank(),
        panel.grid = element_blank())
```

Python Visualisation:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

dataset = pd.read_csv('covtype.csv', index_col=0)

a = dataset['Cover_Type']
sns.countplot(a,palette=["#AFEEEE","#B0C4DE"])

size = 10
data = dataset.iloc[:, :size]
cols = data.columns
data_corr = data.corr()
threshold = 0.5
corr_list = []
for i in range(0, size):
  for j in range(i+1, size):
    if (data_corr.iloc[i, j] >= threshold and data_corr.iloc[i, j] < 1) or
(data_corr.iloc[i, j] < 0 and data_corr.iloc[i, j] <= -threshold):
       corr_list.append([data_corr.iloc[i, j], i, j])
```

```
s_corr_list = sorted(corr_list, key=lambda x: -abs(x[0]))

for v, i, j in s_corr_list:
  sns.pairplot(dataset, hue="Cover_Type", height=6, x_vars=cols[i],
y_vars=cols[j])
  sns.color_palette("flare", as_cmap=True)
  plt.show()

col_list = dataset.columns
col_list = [col for col in col_list if not col[0:4] == 'Soil']
fig, ax = plt.subplots(figsize=(10, 10))
sns.heatmap(dataset[col_list].corr(), square=True, linewidths=1,
cmap="Blues")
plt.title('Correlation of Variables')
plt.show()

sns.lmplot(x='Horizontal_Distance_To_Hydrology',
y='Vertical_Distance_To_Hydrology', data=dataset, hue='Soil_Type1',
fit_reg=False, palette="Blues_r")

sns.lmplot(x='Horizontal_Distance_To_Hydrology',
y='Vertical_Distance_To_Hydrology', data=dataset,
hue='Wilderness_Area2',fit_reg=False, palette="Blues_r")
```