7)Analytics, Machine Learning

Class Imbalance

Data is large and unbalanced. Oversampling has many challanges for this data. So we use UNDERSAMPLING technique. We choose a sample according to Cover Type 4 that has the least row counts(2747).

In [24]:
```
data.Cover_Type.value_counts()
```

Out[24]:
```
2    283301
1    211840
3     35754
7     20510
6     17367
5      9493
4      2747
Name: Cover_Type, dtype: int64
```

In [25]:
```
row_num=data.Cover_Type.value_counts().min()
df2=pd.DataFrame()

for i in data.Cover_Type.unique():
    df2=pd.concat([df2,data[data.Cover_Type==i].sample(row_num)])
```

In [26]:
```
df2.Cover_Type.value_counts()
```

Out[26]:
```
5    2747
2    2747
1    2747
7    2747
3    2747
6    2747
4    2747
Name: Cover_Type, dtype: int64
```

Model Building

```
Model Building and Prediction

Classification algorithms used:

Logistic Regression
Decision Tree (Use DecisionTreeClassifier model from sklearn.tree module)
Random Forest (Use RandomForestClassifier model from sklearn.ensemble module)
Visualizing the Result
Use yellowbrick, seaborn or matplotlib modules to visualize the model results.

Show three plots for the results:

Class Prediction Error Bar Plot
Confusion Matrix
Classification Report.
```

In [27]:
```
X=data.loc[:,'Elevation':'Soil_Type40']
y=data['Cover_Type']
```

In [28]:
```
rem=['Hillshade_3pm','Soil_Type7','Soil_Type8','Soil_Type14','Soil_Type15',
     'Soil_Type21','Soil_Type25','Soil_Type28','Soil_Type36','Soil_Type37']
```

In [29]:
```
X.drop(rem, axis=1, inplace=True)
```

In [30]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=100)
```

In [31]:
```
AC = [] # Accuracy comparisons of the algorithms
```

Logistic Regression

In [32]:
```
logreg = LogisticRegression(solver='liblinear', multi_class='ovr')
logreg.fit(x_train, y_train)
logreg_pred = logreg.predict(x_test)
logreg_accuracy = accuracy_score(logreg_pred , y_test)
AC.append(logreg_accuracy)
```

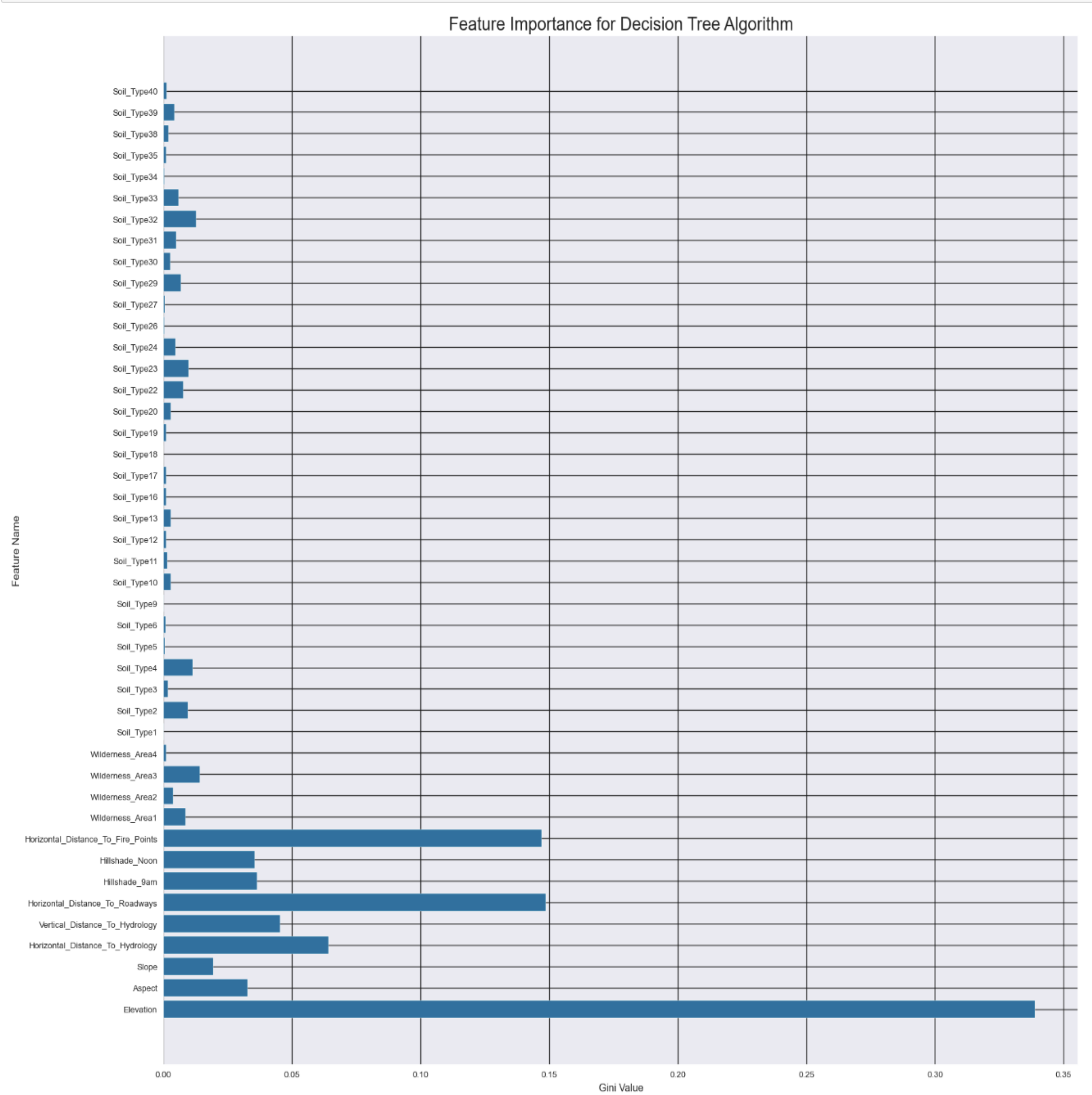
Decision Tree Classifier

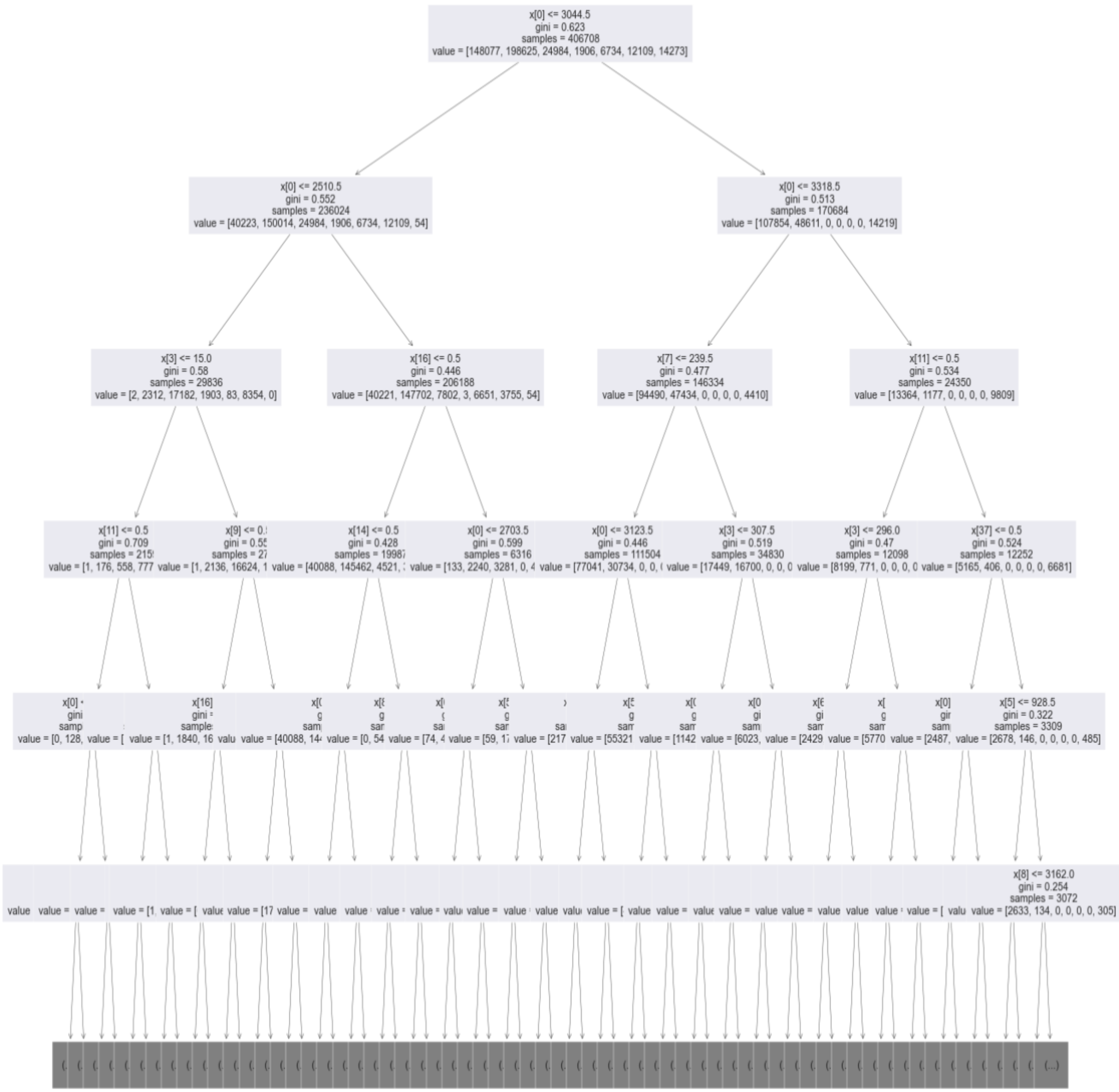
In [33]:
```
dectree = DecisionTreeClassifier()
dectree.fit(x_train, y_train)
dectree_pred = dectree.predict(x_test)
dectree_accuracy = accuracy_score(dectree_pred , y_test)
AC.append(dectree_accuracy)
```

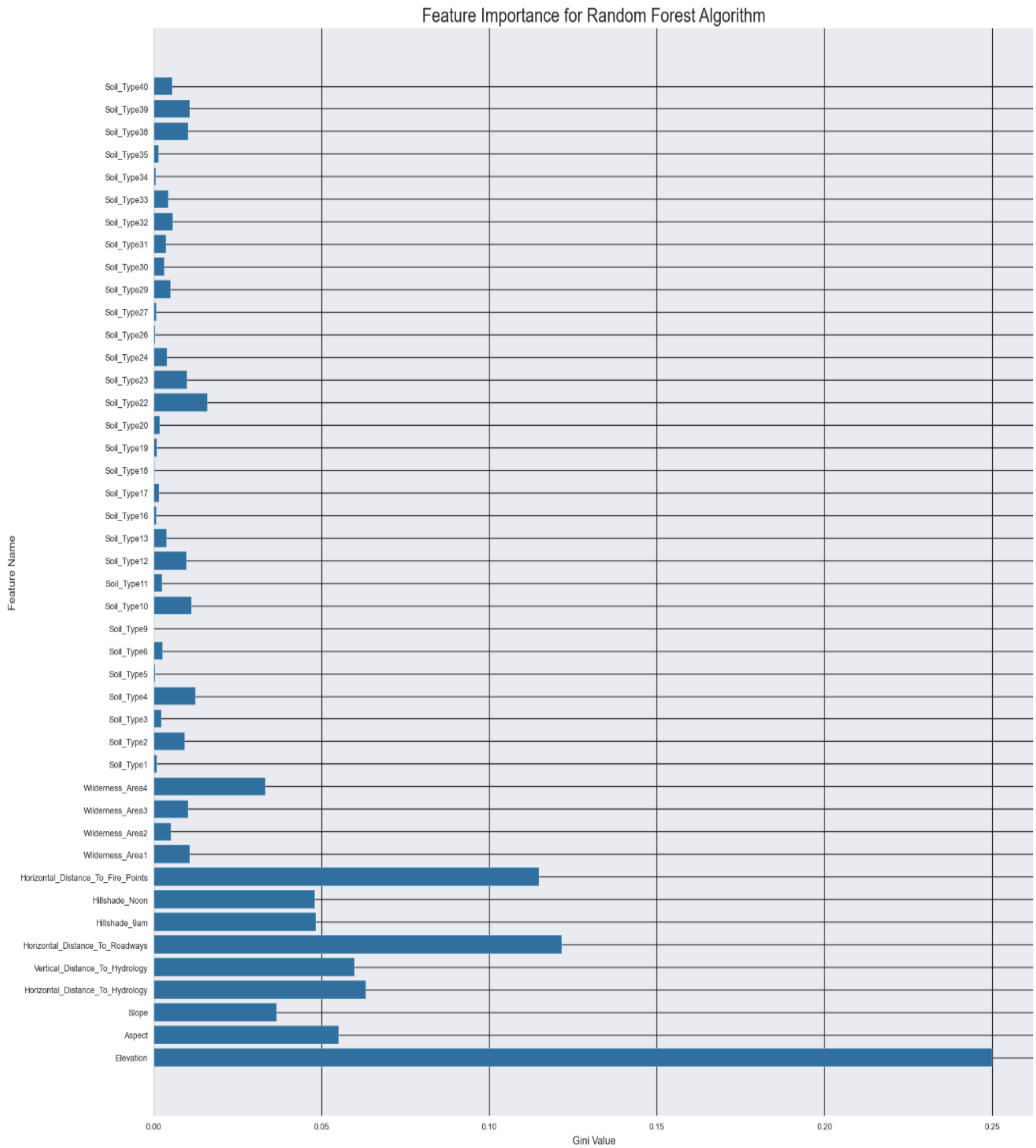
Random Forest Classifier

]:
```
rfc = RandomForestClassifier()
rfc.fit(x_train, y_train)
y_pred = rfc.predict(x_test)
randfor_accuracy = accuracy_score(y_pred , y_test)
AC.append(randfor_accuracy)
```

8)Evaluation and Optimization




x[0] <= 3044.5
gini = 0.623
samples = 406708
value = [148077, 198625, 24984, 1906, 6734, 12109, 14273]
x[0] <= 2510.5
gini = 0.552
samples = 236024
value = [40223, 150014, 24984, 1906, 6734, 12109, 54]
x[0] <= 3318.5
gini = 0.513
samples = 170684
value = [107854, 48611, 0, 0, 0, 0, 14219]
x[3] <= 15.0
gini = 0.58
samples = 29836
value = [2, 2312, 17182, 1903, 83, 8354, 0]
x[16] <= 0.5
gini = 0.446
samples = 206188
value = [40221, 147702, 7802, 3, 6651, 3755, 54]
x[7] <= 239.5
gini = 0.477
samples = 146334
value = [94490, 47434, 0, 0, 0, 0, 4410]
x[11] <= 0.5
gini = 0.534
samples = 24350
value = [13364, 1177, 0, 0, 0, 0, 9809]
x[37] <= 0.5
gini = 0.524
samples = 12252
value = [5165, 406, 0, 0, 0, 0, 6681]
x[5] <= 928.5
gini = 0.322
samples = 3309
value = [2678, 146, 0, 0, 0, 0, 485]
x[8] <= 3162.0
gini = 0.254
samples = 3072
value = [2633, 134, 0, 0, 0, 0, 305]
(...)


Feature Importance for Random Forest Algorithm
Feature Name
Gini Value
Soil_Type40
Soil_Type39
Soil_Type38
Soil_Type35
Soil_Type34
Soil_Type33
Soil_Type32
Soil_Type31
Soil_Type30
Soil_Type29
Soil_Type27
Soil_Type26
Soil_Type24
Soil_Type23
Soil_Type22
Soil_Type20
Soil_Type19
Soil_Type18
Soil_Type17
Soil_Type16
Soil_Type13
Soil_Type12
Soil_Type11
Soil_Type10
Soil_Type9
Soil_Type6
Soil_Type5
Soil_Type4
Soil_Type3
Soil_Type2
Soil_Type1
Wilderness_Area4
Wilderness_Area3
Wilderness_Area2
Wilderness_Area1
Horizontal_Distance_To_Fire_Points
Hillshade_Noon
Hillshade_9am
Horizontal_Distance_To_Roadways
Vertical_Distance_To_Hydrology
Horizontal_Distance_To_Hydrology
Slope
Aspect
Elevation
0.00
0.05
0.10
0.15
0.20
0.25

10)Results

]:

	Accuracy
Logistic Regression	0.698182
Decision Tree Classifier	0.934138
Random Forest Classifier	0.956312

Random Forest Classifer gives the highest accuracy for the dataset. Lets look at the confusion matrix and class prediction error for the model.

]:
```
print('Confusion Matrix:',*confusion_matrix(y_test,y_pred), sep="\n")
print(classification_report(y_test, y_pred))
```

```
Confusion Matrix:
[60303  3274     1     0    21    12   152]
[ 1753 82535   151     0   105   104    28]
[    0   149 10371    52     8   190     0]
[   0    0   95  728    0   18    0]
[   39  537    44     0 2132     7     0]
[    8  142  367    24     6 4711     0]
[ 295   33    0    0    0    0 5909]
              precision    recall  f1-score   support

           1       0.97      0.95      0.96     63763
           2       0.95      0.97      0.96     84676
           3       0.94      0.96      0.95     10770
           4       0.91      0.87      0.89       841
           5       0.94      0.77      0.85      2759
           6       0.93      0.90      0.91      5258
           7       0.97      0.95      0.96      6237

    accuracy                           0.96    174304
   macro avg       0.94      0.91      0.93    174304
weighted avg       0.96      0.96      0.96    174304
```

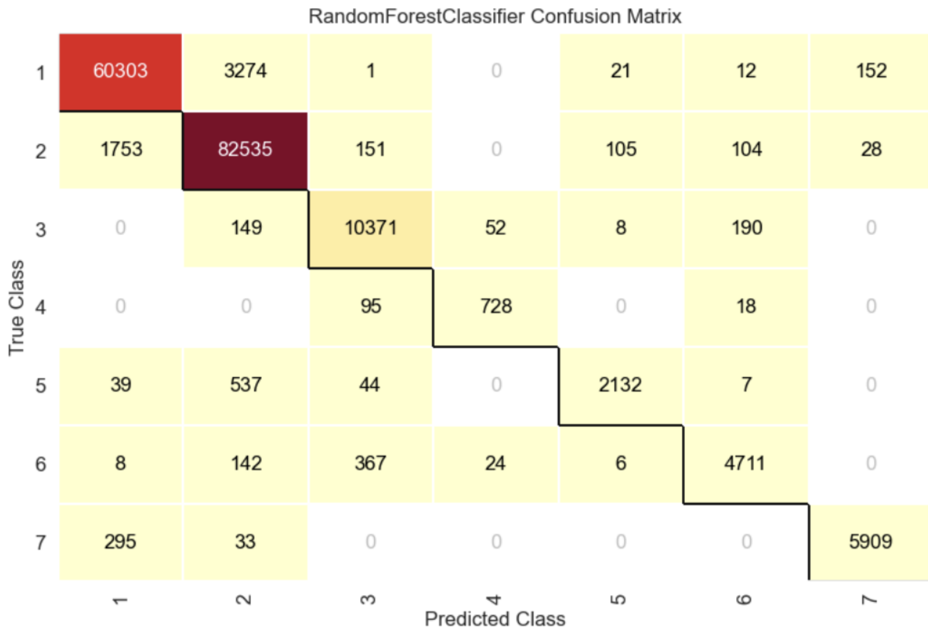
]:
```
rfc_accuracy = accuracy_score(y_test, y_pred)
rfc_f1_score = f1_score(y_test, y_pred, average='weighted')
rfc_recall = recall_score(y_test, y_pred, average='weighted')
print('rfc_accuracy:',rfc_accuracy,
      '\nrfc_f1_score:',rfc_f1_score,
      '\nrfc_recall:',rfc_recall)
```

```
rfc_accuracy: 0.9563119607123187
rfc_f1_score: 0.9560895993941124
rfc_recall: 0.9563119607123187
```

Cross Validation Scores

]:
```
rfc_accuracy = cross_val_score(rfc, x_test, y_test,cv = 10).mean()
rfc_f1_score = cross_val_score(rfc, x_test, y_test,cv = 10,scoring='f1_weighted').mean()
rfc_recall = cross_val_score(rfc, x_test, y_test,cv = 10,scoring='recall_weighted').mean()
print('rfc_accuracy:',rfc_accuracy,
      '\nrfc_f1_score:',rfc_f1_score,
      '\nrfc_recall:',rfc_recall)
```

```
rfc_accuracy: 0.9308105375269816
rfc_f1_score: 0.9304263217514084
rfc_recall: 0.931309668055011
```

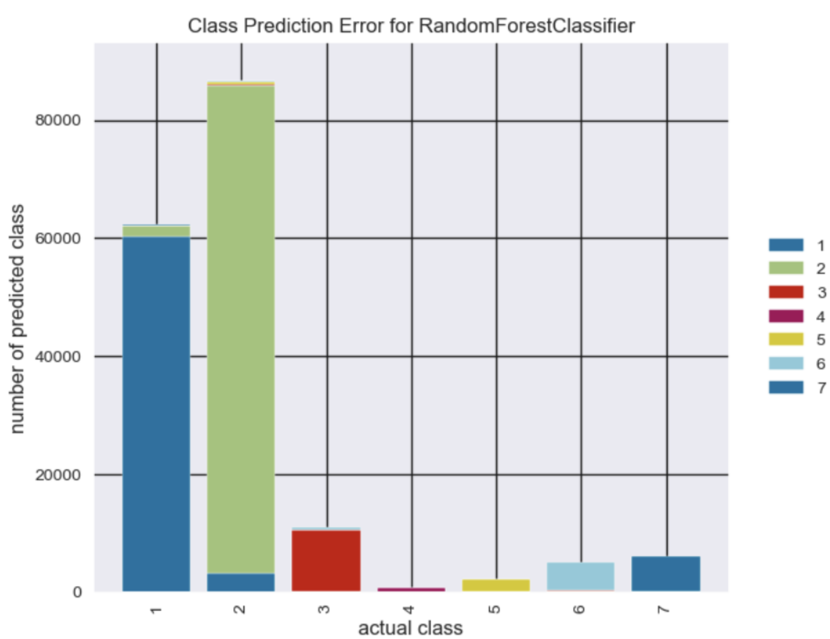


]:
```
visualizer = ClassPredictionError(rfc)

# Fit the training data to the visualizer
visualizer.fit(x_train, y_train)

# Evaluate the model on the test data
visualizer.score(x_test, y_test)

# Draw visualization
visualizer.show();
```



10) Future Work, Comments-

1. What was unique about the data? Did you have to deal with imbalance? What data cleaning did you do? Outlier treatment? Imputation?

Ans)

- This dataset contains tree observations from four areas of the Roosevelt National Forest in Colorado. All observations are cartographic(no remote sensing). The dataset consists of attributes such as elevation, slope, Horizontal distance to hydrology which are very important feature when predicting the cover type of the forest.
- During the project we had deal with the imbalance in the dataset. Cover_Type 1 and 2 i.e Spruce/Fir and Lodgepole Pine seems to dominate the area. To deal with it we can use techniques like undersampling. We can choose a sample according to Cover Type 4 that has the least row counts(2747).
- For the data cleaning part some of the Variables are heavily skewed hence need to be corrected and our dataset did not contain any missing values.
- For the outlier treatment we removed the features with low Std deviation as demonstrated and also remove one of the co-related variable.

2. Did you create any new additional features / variables?

Ans)

we did not create any new feature during the project.

3. What was the process you used for evaluation? What was the best result?

Ans)

The evaluation process typically involves dividing the data into training and testing sets, training the model on the training set, and evaluating the model's performance on the testing set using metrics such as accuracy, precision, recall, F1 score, and AUC-ROC. We also checked how different features are important in predicting the target variable. For this project we created three classification models using Logistic Regression, Decision Tree and Random forest classifier supervised learning techniques. Out of the three model Random forest classifier gave the highest accuracy of 95.6%.

4. What were the problems you faced? How did you solve them?

Ans)

We worked collaboratively by assigning diverse duties to different team members and working together to address any blockages that arose. One such problem we faced is when we wanted to visualize the class prediction error for our ML model. We tried all things from our knowledge but we couldn’t do it. Then one of our teammates browsed the internet and found a python library called ‘yellowbrick’ which had a function ‘ClassPredictionError’ which takes our fitted model as input and gives output a bar chart.

5. What future work would you like to do?

Ans)

The dataset has a lot potential for more future work such as we can predict what types of trees grow in an area based on the surrounding characteristics?. You can also explore different algorithms such as gradient boosting, neural networks, and deep learning models, and evaluate their performance on the dataset.

6. Instructions for individuals that may want to use your work.

Ans) Make sure you import all the necessary libraries especially from yellowbrick.classifier import ClassPredictionError.

If you don't have it installed run

pip install yellowbrick

To install the library.

Also Perform all data cleansing, transformation, and preparation to ensure that the majority of outliers are removed from the dataset. This would result in precise visuals and machine learning model accuracy.