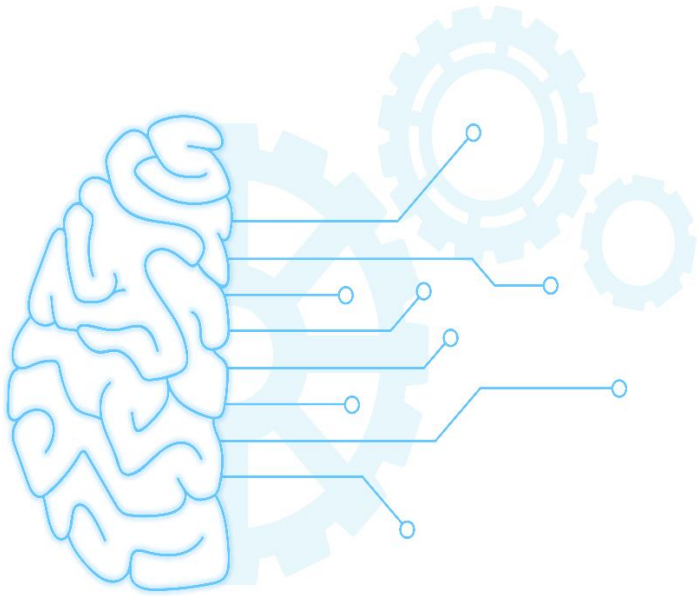
MACHINE LEARNING PROJECT REPORT



Prepared By – Dishit Vasani – N01440553

Data Set Information

- ### Context
  This dataset contains tree observations from four areas of the Roosevelt National Forest in Colorado. All observations are cartographic variables (no remote sensing) from 30-meter x 30-meter sections of the forest. There are over half a million measurements in total.

- ### Content
  This dataset includes information on tree type, shadow coverage, distance to nearby landmarks (roads etcetera), soil type, and local topography.

- ### Acknowledgment
  This dataset is part of the UCI Machine Learning Repository, and the original source can be found at https://archive.ics.uci.edu/ml/datasets/Covertype . The original database owners are Jock A. Blackard, Dr. Denis J. Dean, and Dr. Charles W. Anderson of the Remote Sensing and GIS Program at Colorado State University.

- ### Inspiration
  - Can you build a model that predicts what types of trees grow in an area based on the surrounding characteristics?

  - What kinds of trees are most common in the Roosevelt National Forest?

  - Which tree types can grow in more diverse environments? Are there certain tree types that are sensitive to an environmental factor, such as elevation or soil type?

Unsupervised Learning

Unsupervised learning is a type of machine learning where the users don't have to watch over the model. Instead, it enables the model to operate independently and find previously overlooked patterns and information. It mostly addresses unlabeled data.

Unsupervised Learning Algorithms

Compared to supervised learning, unsupervised learning algorithms enable users to carry out more complicated processing tasks. Unsupervised learning, however, can be more unpredictable than other types of natural learning. Algorithms for unsupervised learning include neural networks, anomaly detection, and grouping.

Clustering

In terms of unsupervised learning, clustering is a key idea. The primary focus is on identifying a structure or pattern in a set of uncategorized data. Unsupervised Education If there are any natural clusters or groupings in your data, clustering algorithms will process them and locate them. You can alter the number of clusters your algorithms should find as well. You can change the level of granularity for these categories.

K-Means Clustering

It is an iterative clustering algorithm, denoted by the letter K, that aids in locating the highest value for each iteration. The desired number of clusters is initially chosen. You must divide the data points into k groups to use this clustering technique. In the same way, a bigger k results in smaller groups with more granularity. Less granularity and larger groups are the results of a lower k.

The algorithm produces a collection of "labels" as its result. A data point is assigned to one of the k groups. Each group in k-means clustering is identified by the creation of a centroid for that group. The centroids act as the cluster's "heart," capturing and incorporating the nearby points into the whole.

Principal Components Analysis

In case you want a higher-dimensional space. You need to select a basis for that space and only the 200 most important scores of that basis. This base is known as a principal component. The subset you select constitute is a new space which is small in size compared to original space. It maintains as much of the complexity of data as possible.

Notebook

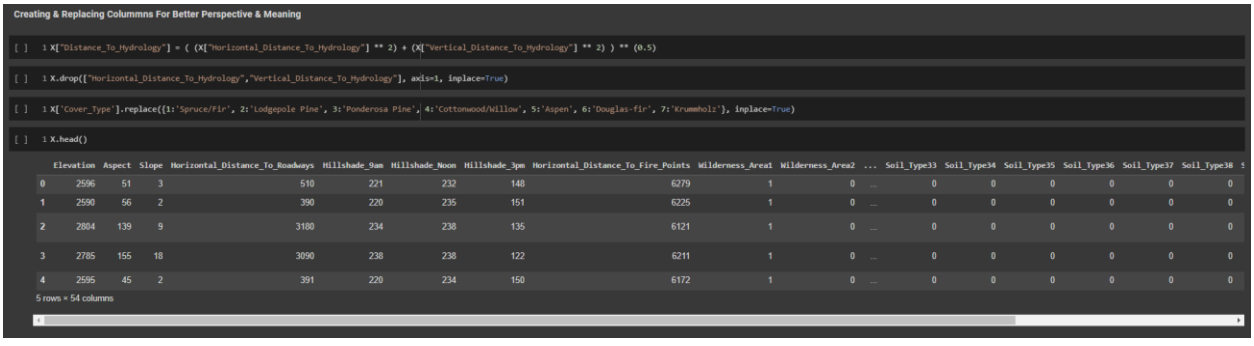
In this notebook we will be exploring method that can be used to visualize clusters that were formed on high-dimensional data (data with more than three dimensions).

First, we will clean our data so that it's in a proper format for clustering, then, we will divide the data into three different clusters using K-Means Clustering. After that, we will go ahead and visualize our three clusters using method: Principal Component Analysis (PCA).

- Imports

- Creating New Data frame to work on & checking Null Values

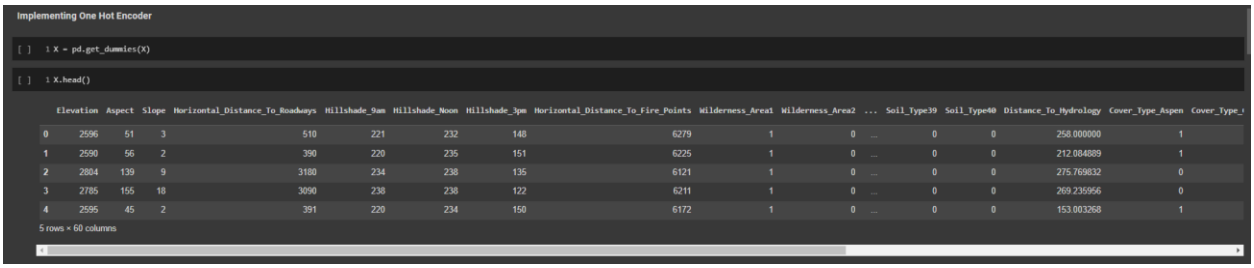
- If we look at the columns: X["Horizontal_Distance_To_Hydrology"] and X[Vertical_Distance_To_Hydrology"], we see that we can create from them, a new column X[Distance_To_Hydrology], which measures the shortest distance to Hydrology. We can calculate the values of this column through using the equation from the Pythagorean Theorem.
-Now that we have X["Distance_To_Hydrology"], and because there's nothing extra special about Vertical or Horizontal Distances to Hydrology, we can drop the original two columns:



-if we look at the values contained within X['Cover_Type'], you'll notice that it contains numerically encoded categorical data. If we head over to the column descriptions on the Forest Cover Type Dataset page, it says that:

1 = "Spruce/Fir", 2 = "Lodgepole Pine", 3 = "Ponderosa Pine", 4 = "Cottonwood/Willow", 5 = "Aspen", 6 = "Douglas-fir", and 7 = "Krummholz".

-We'll relabel our data so that the values in X['Cover_Type'] are more descriptive of what's really contained within it. We'll also do it so that we can easily apply a one-hot-encoding to it, afterwards - so that X['Cover_Type'] will be properly encoded along with the rest of the categorical data in X.


Implementing One Hot Encoder

```
[ ]  1 X = pd.get_dummies(X)
```

```
[ ]  1 X.head()
```

	Elevation	Aspect	Slope	Horizontal_Distance_To_Roadways	Hillshade_9am	Hillshade_Noon	Hillshade_3pm	Horizontal_Distance_To_Fire_Points	Wilderness_Area1	Wilderness_Area2	...	Soil_Type39	Soil_Type40	Distance_To_Hydrology	Cover_Type_Aspen	Cover_Type_
0	2596	51	3	510	221	232	148	6279	1	0	...	0	0	258.000000	1	
1	2590	56	2	390	220	235	151	6225	1	0	...	0	0	212.084889	1	
2	2804	139	9	3180	234	238	135	6121	1	0	...	0	0	275.769832	0	
3	2785	155	18	3090	238	238	122	6211	1	0	...	0	0	269.235956	0	
4	2595	45	2	391	220	234	150	6172	1	0	...	0	0	153.003268	1	

5 rows × 60 columns

- We need to do this because, while each of our categorical variables hold values of either 0 or 1, some of our numerical variables hold values like 2596 and 2785. If we were to leave our data like this, then K-Means Clustering would not give us such a nice result, since K-Means Clustering measures the Euclidean distance between data-points. This means that, if I were to leave our numerical variables un-scaled, then most of the distance measured between points would be attributed to the larger numerical variables, rather than any of the categorical variables.

-To fix this problem we will scale all our numerical variables using sklearn's StandardScaler tool. This tool allows us to scale each numerical variable such that each numerical variable's mean becomes 0, and its variance becomes 1. This is a good way to make sure that all the numerical variables are on roughly the same scale that the categorical (binary) variables are on.

-But, to make sure we scale only our numerical variables -- and not our categorical variables --, we'll split our current Data Frame, X, into two other Data Frames: numer and cater; feature-scale. numer, then recombine the two Data Frames together again into a Data Frame that is suitable for clustering.

Feature Scale Numerical Data

```
numer = X[["Elevation","Aspect","Slope","Horizontal_Distance_To_Roadways","Hillshade_9am","Hillshade_Noon","Hillshade_3pm","Horizontal_Distance_To_Fire_Points","Distance_To_Hydrology"]]
cater = X[["Wilderness_Area1","Wilderness_Area2","Wilderness_Area3","Wilderness_Area4","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_Type1
numer.head()
```

	Elevation	Aspect	Slope	Horizontal_Distance_To_Roadways	Hillshade_9am	Hillshade_Noon	Hillshade_3pm	Horizontal_Distance_To_Fire_Points	Distance_To_Hydrology
0	2596	51	3	510	221	232	148	6279	258.000000
1	2590	56	2	390	220	235	151	6225	212.084889
2	2804	139	9	3180	234	238	135	6121	275.769832
3	2785	155	18	3090	238	238	122	6211	269.235956
4	2595	45	2	391	220	234	150	6172	153.003268

```
cater.head()
```

	Wilderness_Area1	Wilderness_Area2	Wilderness_Area3	Wilderness_Area4	Soil_Type1	Soil_Type2	Soil_Type3	Soil_Type4	Soil_Type5	Soil_Type6	...	Soil_Type38	Soil_Type39	Soil_Type40	Cover_Type_Aspen	Cover_Type_Cottonwood/Willow	Cover_Type_Douglas-fir	Cove
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	
1	1	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
4	1	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0	0	

5 rows × 51 columns

-Feature Scale

```
scaler = StandardScaler()
numer = pd.DataFrame(scaler.fit_transform(numer))
numer.columns = ["Elevation_Scaled","Aspect_Scaled","Slope_Scaled","Horizontal_Distance_To_Roadways_Scaled","Hillshade_9am_Scaled","Hillshade_Noon_Scaled","Hillshade_3pm_Scaled","Horizontal_Distance_To_Fire_Points_Scaled","Distance_To_Hydrology_Scaled"]
X = pd.concat([numer, cater], axis=1, join='inner')
X.head()
```

	Elevation_Scaled	Aspect_Scaled	Slope_Scaled	Horizontal_Distance_To_Roadways_Scaled	Hillshade_9am_Scaled	Hillshade_Noon_Scaled	Hillshade_3pm_Scaled	Horizontal_Distance_To_Fire_Points_Scaled	Distance_To_Hydrology_Scaled	Wilderness_Area1	...	Soil_Type
0	-1.297805	-0.935157	-1.482820	-1.180146	0.330743	0.439143	0.142960	3.246283	-0.083233	1	...	
1	-1.319235	-0.890480	-1.616363	-1.257106	0.293388	0.590899	0.221342	3.205504	-0.294777	1	...	
2	-0.554907	-0.148836	-0.681563	0.532212	0.816364	0.742654	-0.196691	3.126965	-0.001362	1	...	
3	-0.622768	-0.005869	0.520322	0.474492	0.965786	0.742654	-0.536343	3.194931	-0.031466	1	...	
4	-1.301377	-0.988770	-1.616363	-1.256464	0.293388	0.540313	0.195215	3.165479	-0.566983	1	...	

5 rows × 60 columns

-Time to build Clusters, I will be visualizing only three different clusters on our data. I chose three because I found it to be a good number of clusters to help us visualize our data in a non-complicated way.

Time To Build Our Clusters

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
KMeans(n_clusters=3)
clusters = kmeans.predict(X)
X["Cluster"] = clusters
```

- PCA is an algorithm that is used for dimensionality reduction - meaning, informally, that it can take in a Data Frame with many columns and return a Data Frame with a reduced number of columns that still retains much of the information from the columns of the original Data Frame. The columns of the Data Frame produced from the PCA procedure are called Principal Components. I would use these principal components to help visualize the clusters in 1-D, 2-D, and 3-D space, since I cannot easily visualize the data, we have in higher dimensions. For example, we can use two principal components to visualize the clusters in 2-D space, or three principal components to visualize the clusters in 3-D space.

Principal Component Analysis

```
plotX = pd.DataFrame(np.array(X.sample(5000)))
plotX.columns = X.columns
```

```
pca_1d = PCA(n_components=1)
pca_2d = PCA(n_components=2)
pca_3d = PCA(n_components=3)
```

```
PCs_1d = pd.DataFrame(pca_1d.fit_transform(plotX.drop(["Cluster"], axis=1)))
PCs_2d = pd.DataFrame(pca_2d.fit_transform(plotX.drop(["Cluster"], axis=1)))
PCs_3d = pd.DataFrame(pca_3d.fit_transform(plotX.drop(["Cluster"], axis=1)))
```

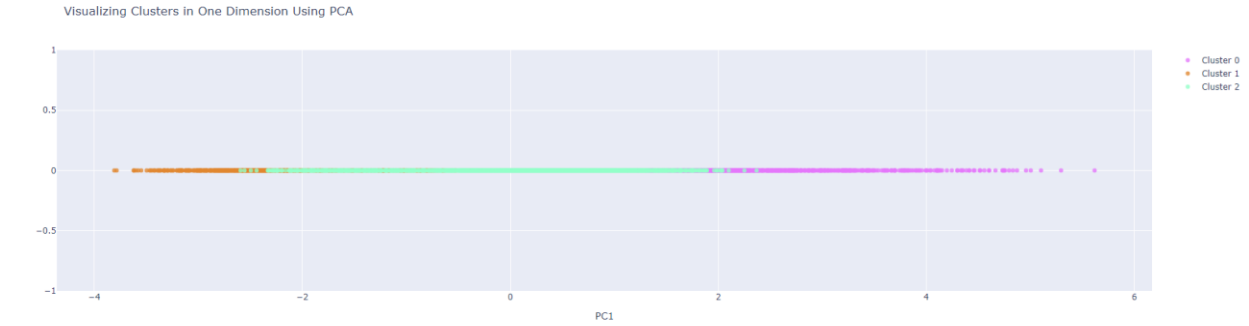
```
PCs_1d.columns = ["PC1_1d"]
PCs_2d.columns = ["PC1_2d", "PC2_2d"]
PCs_3d.columns = ["PC1_3d", "PC2_3d", "PC3_3d"]
```

```
plotX = pd.concat([plotX,PCs_1d,PCs_2d,PCs_3d], axis=1, join='inner')
```

```
plotX["dummy"] = 0
```

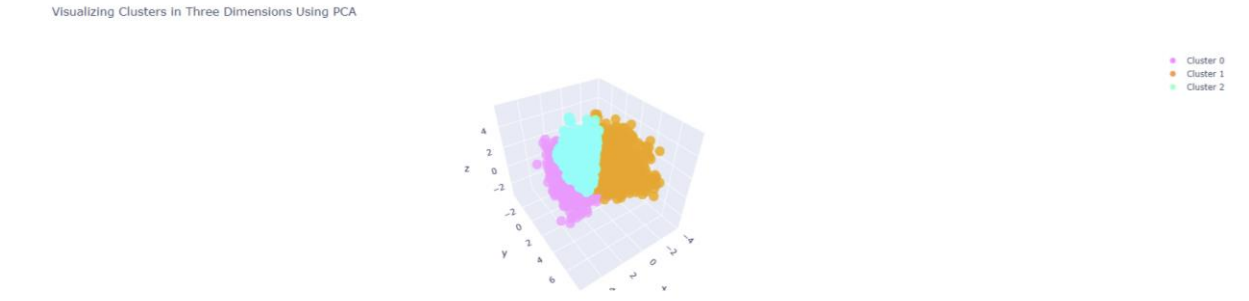
```
cluster0 = plotX[plotX["Cluster"] == 0]
cluster1 = plotX[plotX["Cluster"] == 1]
cluster2 = plotX[plotX["Cluster"] == 2]
```

- 1-D Visualization

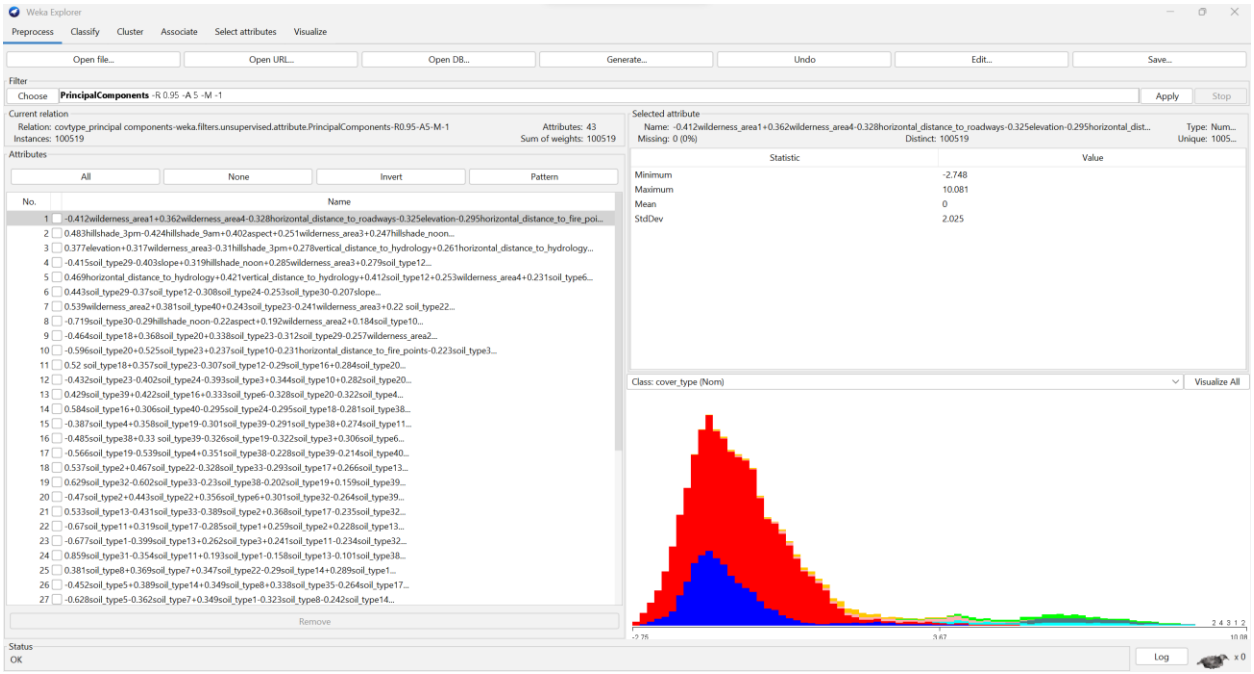


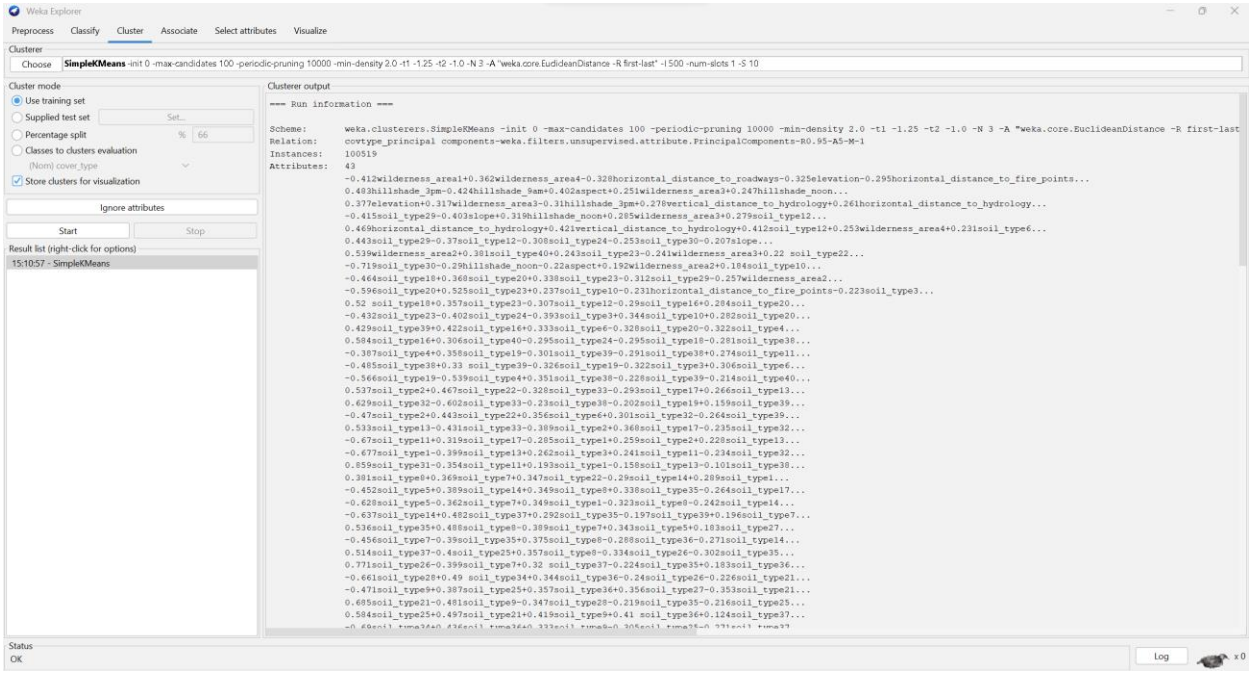
-2-D Visualization

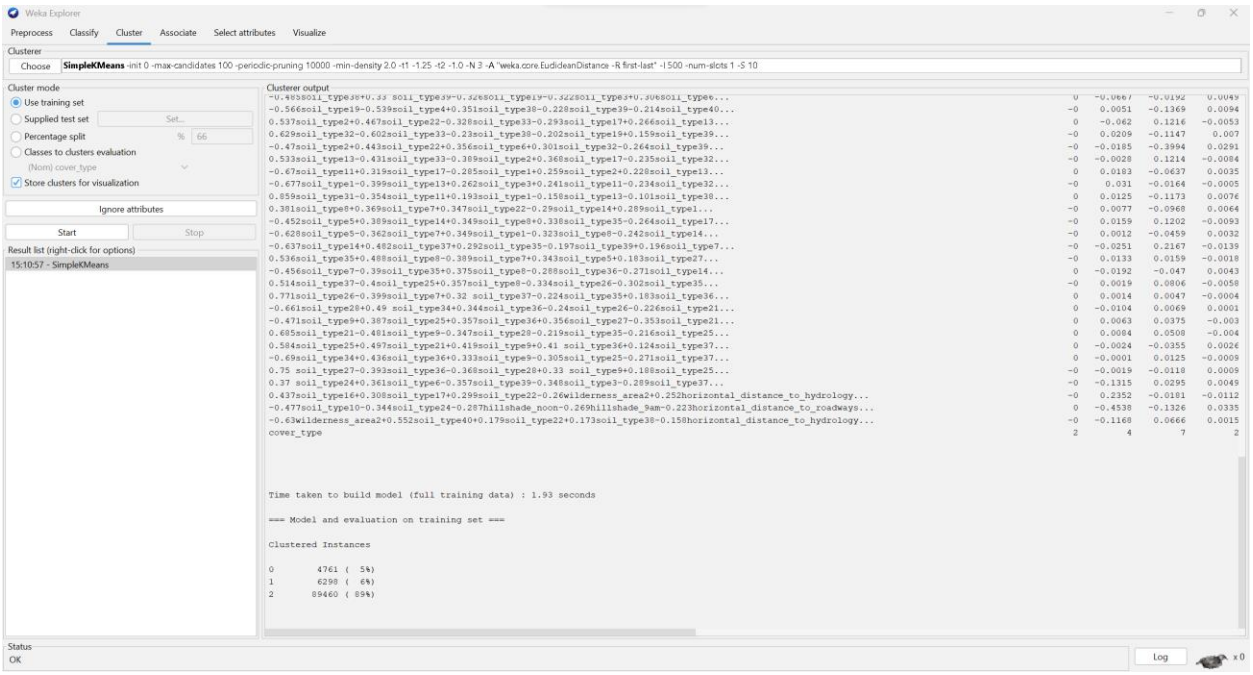
-3-D Visualization

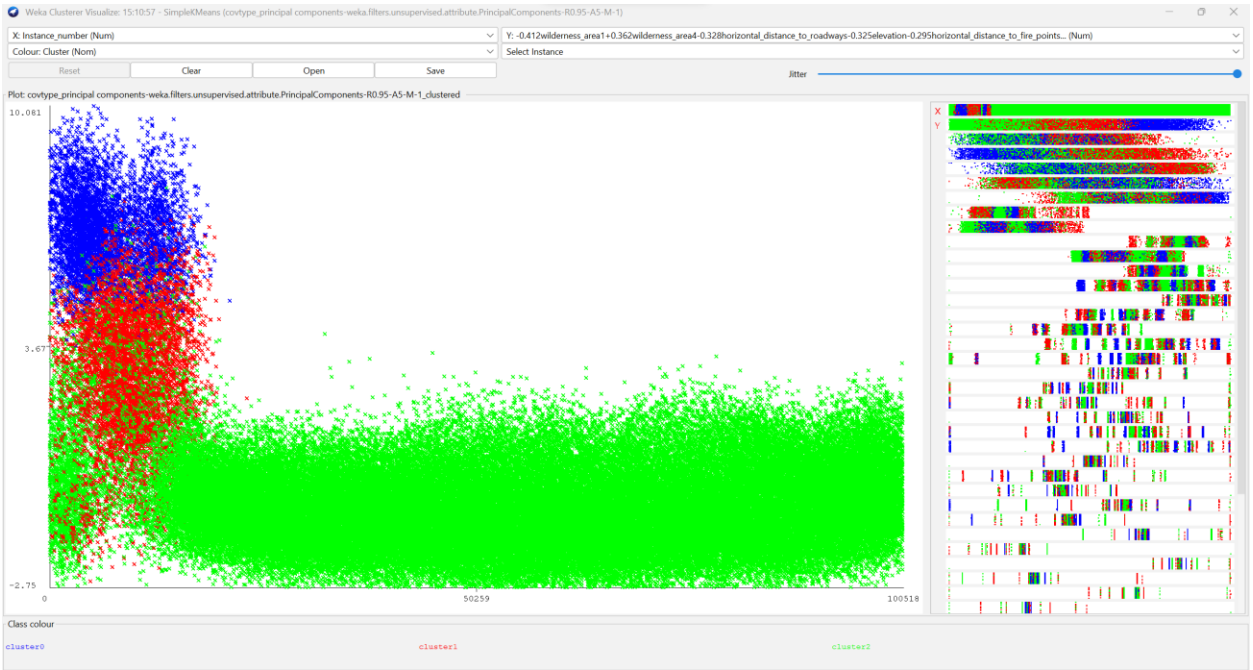


-Weka


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
PrincipalComponents -R 0.95 -A 5 -M -1
Apply
Stop
Current relation
Relation: covtype_principal components-weka.filters.unsupervised.attribute.PrincipalComponents-R0.95-A5-M-1
Instances: 100519
Attributes: 43
Sum of weights: 100519
Selected attribute
Name: -0.412wilderness_area1+0.362wilderness_area4-0.328horizontal_distance_to_roadways-0.325elevation-0.295horizontal_dist...
Missing: 0 (0%)
Distinct: 100519
Type: Num...
Unique: 1005...
Attributes
All
None
Invert
Pattern
No.
Name
Statistic
Value
Minimum
-2.748
Maximum
10.081
Mean
0
StdDev
2.025
Class: cover_type (Nom)
Visualize All
Remove
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) cover_type
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
15:10:57 - SimpleKMeans
Clusterer output
=== Run information ===
Scheme: weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last
Relation: covtype_principal components-weka.filters.unsupervised.attribute.PrincipalComponents-R0.95-A5-M-1
Instances: 100519
Attributes: 43
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) cover_type
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
15:10:57 - SimpleKMeans
Clusterer output
-0.566soil_type19-0.539soil_type4+0.351soil_type38-0.228soil_type39-0.214soil_type40...
0.537soil_type2+0.467soil_type22-0.328soil_type33-0.293soil_type17+0.266soil_type13...
0.629soil_type32-0.602soil_type33-0.23soil_type38-0.202soil_type19+0.159soil_type39...
-0.47soil_type2+0.443soil_type22+0.356soil_type4+0.301soil_type32-0.264soil_type39...
0.533soil_type13-0.431soil_type33-0.389soil_type2+0.368soil_type17-0.235soil_type32...
-0.67soil_type11+0.319soil_type17-0.285soil_type1+0.259soil_type2+0.228soil_type13...
-0.677soil_type1-0.399soil_type13+0.262soil_type3+0.241soil_type11-0.234soil_type32...
0.859soil_type31-0.354soil_type11+0.193soil_type1-0.158soil_type13-0.101soil_type38...
0.381soil_type8+0.369soil_type7+0.347soil_type22-0.29soil_type14+0.289soil_type1...
-0.452soil_type5+0.389soil_type14+0.349soil_type8+0.338soil_type35-0.264soil_type17...
-0.628soil_type5-0.362soil_type7+0.349soil_type1-0.323soil_type8-0.242soil_type14...
-0.637soil_type14+0.482soil_type37+0.292soil_type35-0.197soil_type39+0.196soil_type7...
0.536soil_type35+0.488soil_type8-0.389soil_type7+0.343soil_type5+0.183soil_type27...
-0.456soil_type7-0.39soil_type35+0.375soil_type8-0.288soil_type36-0.271soil_type14...
0.514soil_type37-0.4soil_type25+0.357soil_type8-0.334soil_type26-0.302soil_type35...
0.771soil_type26-0.399soil_type7+0.32 soil_type37-0.224soil_type35+0.183soil_type36...
-0.661soil_type28+0.49 soil_type34+0.344soil_type36-0.24soil_type26-0.226soil_type21...
-0.471soil_type9+0.387soil_type25+0.357soil_type36+0.356soil_type27-0.353soil_type21...
0.685soil_type21-0.481soil_type9-0.347soil_type28-0.219soil_type35-0.216soil_type25...
0.584soil_type25+0.497soil_type21+0.419soil_type9+0.41 soil_type36+0.124soil_type37...
-0.69soil_type34+0.436soil_type36+0.333soil_type9-0.305soil_type25-0.271soil_type37...
0.75 soil_type27-0.393soil_type36-0.368soil_type28+0.33 soil_type9+0.188soil_type25...
0.37 soil_type24+0.361soil_type6-0.357soil_type39-0.348soil_type3-0.289soil_type37...
0.437soil_type16+0.308soil_type17+0.299soil_type22-0.26wilderness_area2+0.252horizontal_distance_to_hydrology...
-0.477soil_type10-0.344soil_type24-0.287hillshade_noon-0.269hillshade_9am-0.223horizontal_distance_to_roadways...
-0.63wilderness_area2+0.552soil_type40+0.179soil_type22+0.173soil_type38-0.158horizontal_distance_to_hydrology...
cover_type
Time taken to build model (full training data) : 1.93 seconds
=== Model and evaluation on training set ===
Clustered Instances
0 4761 (5%)
1 6298 (6%)
2 89460 (89%)
Status
OK
Log
x 0


Weka Clusterer Visualize: 15:10:57 - SimpleKMeans (covtype_principal components-weka.filters.unsupervised.attribute.PrincipalComponents-R0.95-A5-M-1)
X: Instance_number (Num)
Y: -0.412wilderness_area1+0.362wilderness_area4-0.328horizontal_distance_to_roadways-0.325elevation-0.295horizontal_distance_to_fire_points... (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: covtype_principal components-weka.filters.unsupervised.attribute.PrincipalComponents-R0.95-A5-M-1_clustered
10.081
3.677
-2.75
0
50259
100518
X
Y
Class colour
cluster0
cluster1
cluster2

Conclusion

-As we can see from the plots above: if you have data that is highly cluster able, then PCA is a pretty good way to view the clusters formed on the original data. Also, visualizing the clusters is more effective when the clusters are visualized using more principal components, rather than less. For example, the 2-D plot did a better job of providing a clear visual representation of the clusters than the 1-D plot; and the 3-D plot did a better job than the 2-D plot!
-Comparing the result set with WEKA we would observe a great variation as the Data we selected for WEKA contained only around 5000 Instances due to Machine limitations.

Supervised Machine Learning

Supervised Machine Learning is an algorithm that learns from labeled training data to help you predict outcomes for unforeseen data. In Supervised learning, you train the machine using data that is well “labeled.” It means some data is already tagged with correct answers. It can be compared to learning in the presence of a supervisor or a teacher.

Random Forest

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

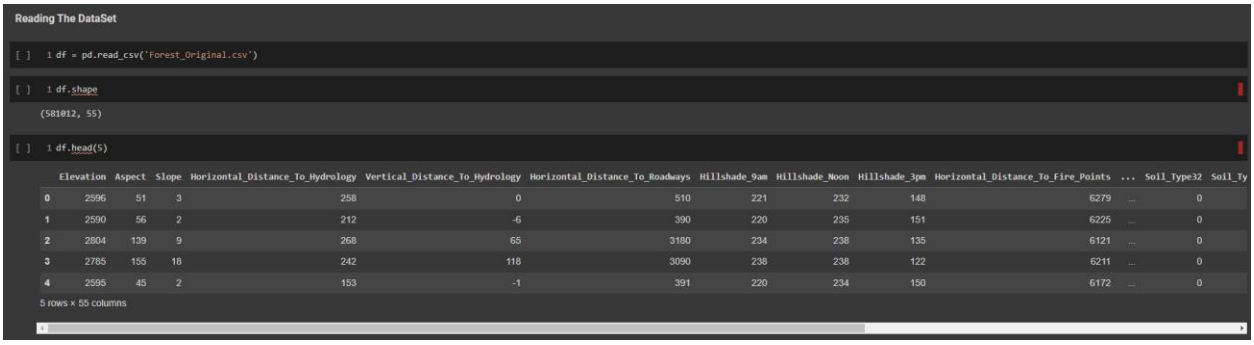
One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

Notebook

In this Notebook we are going to Implement Random Forest Algorithm and compare the same result with the output produced by WEKA.

- Imports

-Reading the Dataset



-Model

Model

```
[ ]  1 X = df.drop(['Cover_Type'] , axis = 1)
```

```
[ ]  1 y = df['Cover_Type']
```

```
[ ]  1 X.shape
```

```
(581012, 54)
```

```
[ ]  1 y.shape
```

```
(581012,)
```

-Train Test Split

Train Test Split

```
[ ]  1 X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.30, random_state=42)
```

-Random Forest Classifier

Random Forest Classifier

```
[ ]  1 random = RandomForestClassifier()
```

```
[ ]  1 random.fit(X_train , y_train)
```

```
RandomForestClassifier()
```

```
[ ]  1 random.score(X_test,y_test)
```

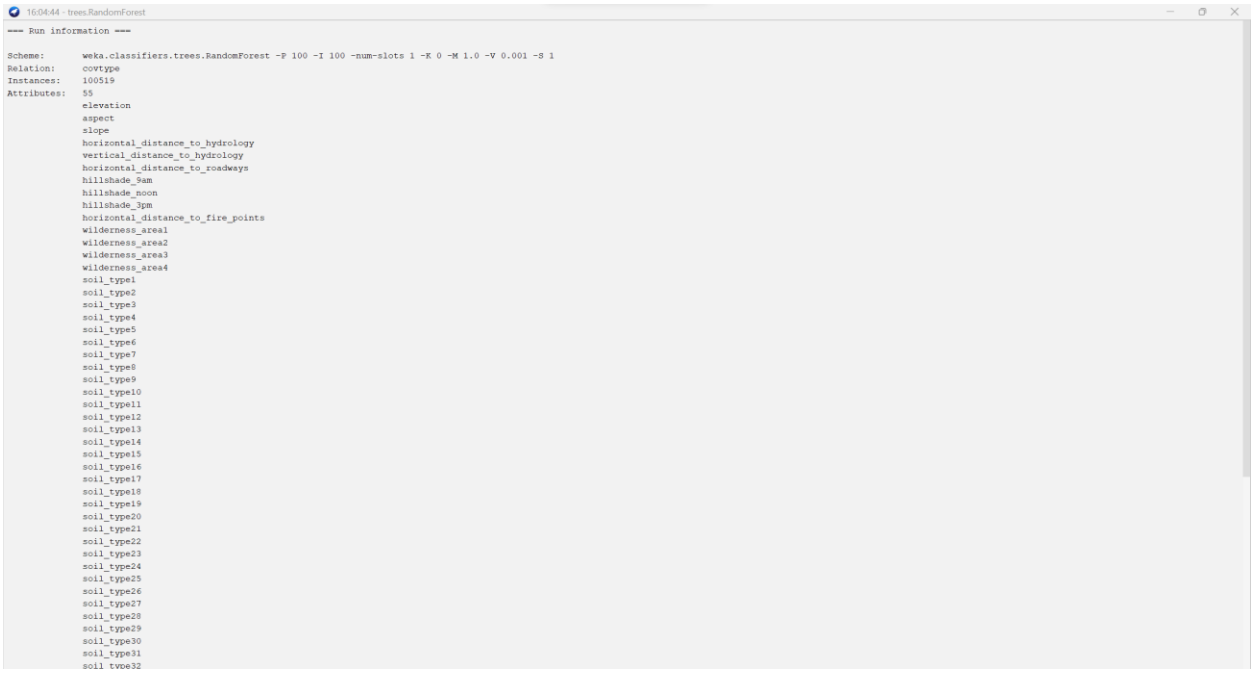
```
0.952181246577382
```

```
[ ]  1 y_pred = random.predict(X_test)
```

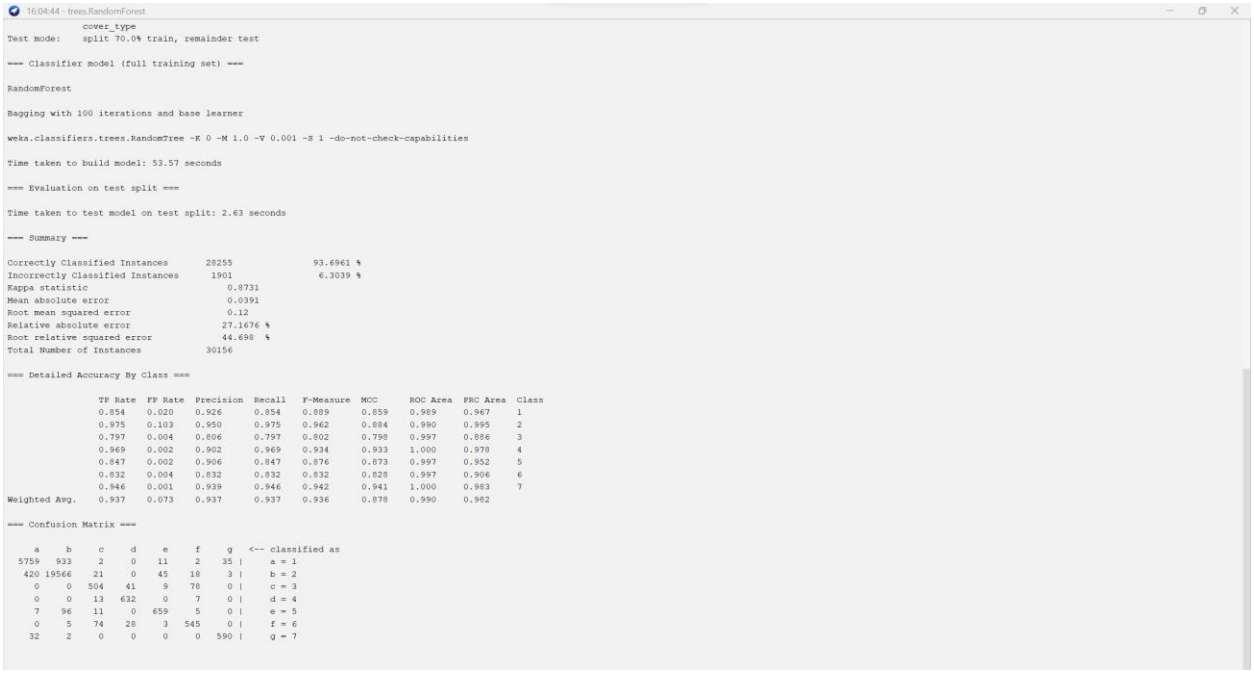
```
[ ]  1 print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           1       0.96      0.94      0.95     63556
           2       0.95      0.97      0.96     85078
           3       0.94      0.96      0.95     10638
           4       0.91      0.85      0.88       795
           5       0.94      0.75      0.83      2941
           6       0.93      0.89      0.91      5227
           7       0.97      0.95      0.96      6069

    accuracy                           0.95    174304
   macro avg       0.94      0.90      0.92    174304
weighted avg       0.95      0.95      0.95    174304
```

-Weka



```
16:04:44 - trees.RandomForest
=== Run information ===

Scheme:       weka.classifiers.trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1
Relation:     covtype
Instances:    100519
Attributes:   55
              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
              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
```



```
16:04:44 - trees.RandomForest
              cover_type
Test mode:    split 70.0% train, remainder test

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 53.57 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 2.63 seconds

=== Summary ===

Correctly Classified Instances       28255               93.6961 %
Incorrectly Classified Instances      1901                6.3039 %
Kappa statistic                          0.8731
Mean absolute error                      0.0391
Root mean squared error                  0.12
Relative absolute error                 27.1676 %
Root relative squared error             44.698  %
Total Number of Instances            30156

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.854    0.020    0.926      0.854    0.889      0.859    0.989     0.967     1
               0.975    0.103    0.950      0.975    0.962      0.884    0.990     0.995     2
               0.797    0.004    0.806      0.797    0.802      0.798    0.997     0.886     3
               0.969    0.002    0.902      0.969    0.934      0.933    1.000     0.978     4
               0.847    0.002    0.906      0.847    0.876      0.873    0.997     0.952     5
               0.832    0.004    0.832      0.832    0.832      0.828    0.997     0.906     6
               0.946    0.001    0.939      0.946    0.942      0.941    1.000     0.983     7
Weighted Avg.  0.937    0.073    0.937      0.937    0.936      0.878    0.990     0.982

=== Confusion Matrix ===

     a     b     c     d     e     f     g   <-- classified as
  5759   933     2     0    11     2    35 |     a = 1
   420 19566    21     0    45    18     3 |     b = 2
     0     0   504    41     9    78     0 |     c = 3
     0     0    13   632     0     7     0 |     d = 4
     7    96    11     0   659     5     0 |     e = 5
     0     5    74    28     3   545     0 |     f = 6
    32     2     0     0     0     0   590 |     g = 7
```

Conclusion

- *As we can see both the outputs Running Random Forest on Python gave us accuracy of 95.21% here, we took the whole Dataset for better evaluation. On the other hand running the Random Forest on WEKA fetched us with accuracy of 93.69%, in here we ran the algorithm on fewer instances due to machine incompatibility. Then to there was not much difference in their accuracy score.*

References

- https://www.guru99.com/r-random-forest-tutorial.html
- https://www.kaggle.com
- https://www.geeksforgeeks.org/k-means-clustering-using-weka/
- https://ikuz.eu/csv2arff/
- https://www.guru99.com/unsupervised-machine-learning.html
- https://archive.ics.uci.edu/ml/datasets.php