RESEARCH ARTICLE OPEN ACCESS

Comparative analysis of supervised and unsupervisedMachine learning algorithms on Iris Dataset

Pranali Mudgal [1], Shraddha Masih [2]

[1] Student, School of Computer Science and IT, Devi Ahilya Vishwavidyalaya,Indore - India

[2] Associate Professor, School of Computer Science and IT, Devi AhilyaVishwavidyalaya, Indore - India

ABSTRACT

Machine learning is a dominant approach to extracting information from extensive data, recognizing patterns, and predicting results. All this requires a set of algorithms. This paper has classified the Iris flower dataset into its different labels(species) according to their length and width of sepal andpetal. Furthermore, their comparative study according to various algorithms is achieved. Grouping of Iris bloom is achieved by Machine learning Unsupervised Clustering-basedK-Means algorithm and Hierarchical Agglomerative Clustering algorithm and Supervised Classification based Linear Regression, Decision Tree and Random Forest. The purpose is to build a model that automatically recognizes and predicts the group of iris species. Libraries used for this in the paper are NumPy, Pandas, Matplotlib, and Scikit-learn.

***keywords*: -** Classification, Clustering, Machine learning, Supervised, Unsupervised, NumPy, Pandas, Matplotlib, K- Means, Hierarchical Agglomerative Clustering Algorithm, Linear Regression, Decision Tree, Random Forest, Google Collab.

I. INTRODUCTION

The human being can segregate things according to their similarities, and so, they can identify the patterns, but when there is a massive amount of data, humans cannot study that data within seconds; hence to fulfill this need, we must make use of the machine, but the problem arises like, humans it cannot segregate things and find patterns. Hence training the machine is necessary for pattern recognization and data study. For that, we use Machine Learning.

Machine Learning is a developing field that helps find a pattern in a large dataset and predict several future conditions by analyzing and visualizing the dataset with the help of various algorithms.

This research aims to analyze and visualize the Dataset and apply various algorithms, and at last train the machine for the desired predictions that we want from our research.

Here the object is Iris Flower which has numerical and categorical data. The data set of Iris contains three different classes: Setosa, Versicolour, and Virginica. The designed recognition system will distinguish these three different classes of Iris. Initially, we are pre-processing the given dataset and then applies various algorithms and visualize it with the help of graphs. At last, we are training the machine so that the users do not need to tell the computer which class the Iris belongs to; the computer can recognize them all by itself.

The various algorithm is used for clustering and classifying Iris species in this research. Clustering Algorithms applied are K-means and Hierarchical Agglomerative clustering and classification based: Linear Regression, Decision Tree and Random Forest.

There are several different kinds of Machine learning algorithms applied in different fields. Choosing a decent algorithm is essential for each research.

II. DATASET

The dataset is of Iris Flower, which speaks about its category and four parameters. This is comparatively a minimal data set with 150 samples and consists of 50 samples from each of Iris's three species.

1) Iris Setosa

2) Iris Virginica

3) Iris Versicolor.

Four features were measured from each sample. They are:

1) Sepal Length

2) Sepal Width

3) Petal Length

4) Petal Width.

All these four parameters are measured in Centimetres (Cm). Based upon the combination of these four features, the species among the three can be predicted. Iris dataset also has an Id column which is a numeric parameter.

III. PROPOSED SYSTEM

A comparative study of the Machine learning algorithms is performed to design a model that automatically predicts Iris flower species using Machine learning, Python, and Scikit-learn.

1.1 System Architecture: -

1. Load Iris dataset in google colaboratory.
2. Split the dataset into training data and testing data.
3. Algorithms are applied to train the model.
4. the Predicted result is displayed with accuracy.
5. Comparative study is done based on the predicted results.

1.2 Advantage of Proposed System:

The implementation using Python and its libraries make the system fast, accurate, and helpful for further Advancements. Besides, supervised and unsupervised learning makes a comparative study of the Iris dataset clear.

4. Algorithms: -

Segregation of Iris Flower into three different species is achieved using two types of algorithms clustering and classification. These two types include a various algorithm that is analyzed in our paper are:

Clustering Algorithms: -

1. Partition clustering-based: K-Means algorithm
2. Hierarchical clustering based: Agglomerative Clustering algorithm

Classification Algorithms:

1. Linear Regression
2. Decision Tree
3. Random Forest

4.1 Clustering:

Clustering is an essential concept when it comes to unsupervised learning. Its purpose is to find a structure or pattern in a collection of uncategorized data. Clustering algorithms tries to find natural groups if they exist in the data. We can also modifyhow many clusters our algorithm should identify. It permits us to adjust the granularity of these groups.

4.1.1 K-Means

This algorithm is used to partition *n* observations into *k* clusters. Each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a cluster prototype by calculating the mean. Primary, we initialize k points, called means, randomly. Then, we categorize each item to its closest mean, and after that, we update the mean's coordinates, which are the averages of the items categorized in that mean so far. We repeat the process for a given number of iterations, and at the end, we get clusters.

Implementation of K-means algorithm: -

- From the Scikit-learn library, we have used a cluster module that imports the KMeans class.
- After that object km is created, we declared the desired number of clusters to be made by the model.
- Here clusters are made irrespective of species of Iris.
- Then in the y_kmeans_predicted variable, we passed our normalized dataset.
- At last, the array is displayed, showing clusters that are formed.
- It is predicting which data is going in which cluster with the help of the fit predict () method.

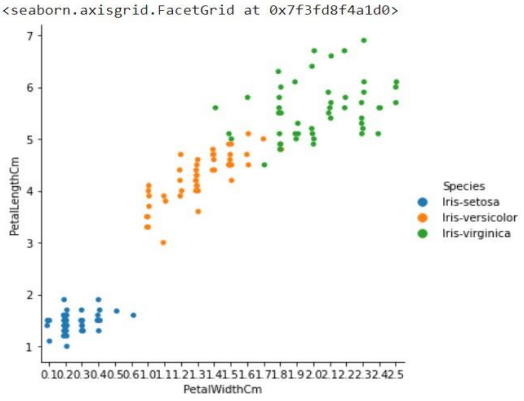
```
[ ] from sklearn.cluster import KMeans
    km = KMeans(n_clusters=3)

[ ] y_kmeans_predicted = km.fit_predict(normalised_data_standard_scaler)

[ ] y_kmeans_predicted

    array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 2,
           1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 2, 1,
           1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2,
           2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 1, 2, 2, 2, 2, 2,
           2, 1, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 1], dtype=int32)

[ ] x_data["Species_pred_kmeans"] = y_kmeans_predicted

    sns.catplot(x="PetalWidthCm",y="PetalLengthCm",hue="Species_pred_kmeans" ,data=x_data)
```

Output: -

The scatter plot is made using the seaborn library having labels as Petal Length and Petal Width Comparing the original dataset clusters and clusters made after prediction.

Original Dataset graph

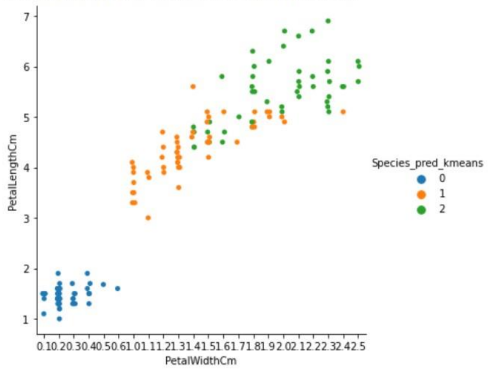
```
sns.catplot(x="PetalWidthCm",y="PetalLengthCm",hue="Species" ,data=final_data)
```



Predicted graph after the model is trained.

```
sns.catplot(x="PetalWidthCm",y="PetalLengthCm",hue="Species_pred_kmeans" ,data=x_data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f3fd8ed0550>
```



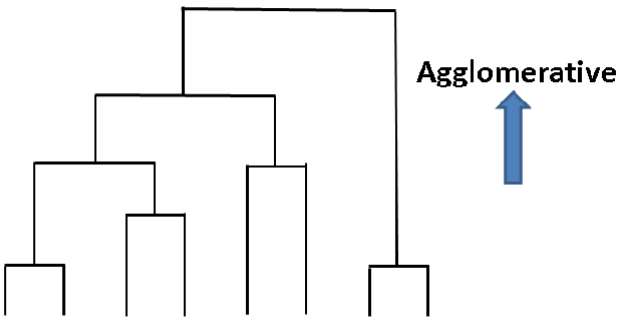
The model we have trained using k-means shows quite an exact result before training the model.

4.1.2. Hierarchical Agglomerative clustering: -

Hierarchical clustering, also recognized as *hierarchical cluster analysis*, is an algorithm that groups similar objects that form *clusters*. The endpoint is a set of clusters, where each cluster is dissimilar from every other cluster, and the objects within each cluster are broadly like each other.

Agglomerative clustering

It is a type of hierarchical clustering which obeys a bottom-up approach. Here small clusters are created, and they join to form a final big cluster. Firstly, the individual points form a cluster by calculating the distance between them; they form clusters; eventually, a final big cluster is formed.



Implementation of Agglomerative clustering: -

- From the scikit-learn library, we have used a cluster module that imports the Agglomerative Clustering class.
- Then the object of this class is made.
- Pass attribute to this class describing number of clusters to be formed.
- Create a variable named y_predicted_agglo.
- Pass normalized data in this variable and apply fit_predict ().
- This method will train a model to predict and form a cluster.
- The array defines the clusters that the trained model forms.

2.1.Agglomerative Clustering

```
[ ] from sklearn.cluster import AgglomerativeClustering

[ ] ac = AgglomerativeClustering(n_clusters=3)

[ ] y_predicted_agglo = ac.fit_predict(normalised_data_standard_scaler)

y_predicted_agglo
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 0,
       2, 0, 2, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2,
       2, 0, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0])
```

```
x_data["Species_pred_agglo"] = y_predicted_agglo

[ ] sns.catplot(x="PetalWidthCm",y="PetalLengthCm",hue="Species_pred_agglo" ,data=x_data)
```

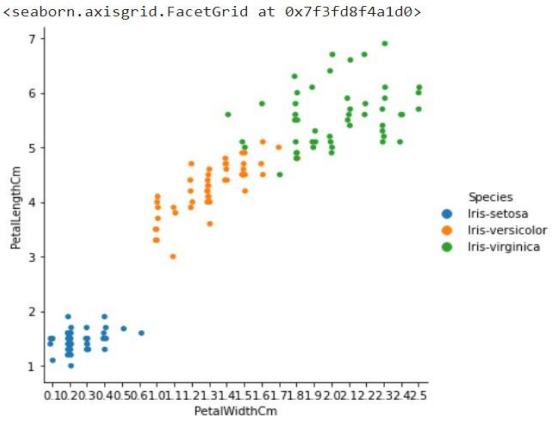
Output: -

The scatter plot is made using the seaborn library having labels as Petal Length and Petal Width, comparing the original dataset cluster and cluster made after prediction. Here clusters are formed irrespective of Species [Iris category].

Original data graph: -

```
sns.catplot(x="PetalWidthCm",y="PetalLengthCm",hue="Species" ,data=final_data)
```

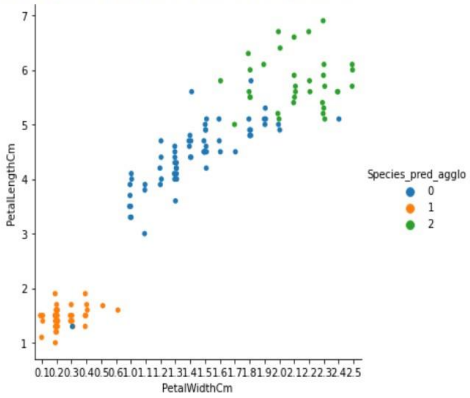
```
<seaborn.axisgrid.FacetGrid at 0x7f3fd8f4a1d0>
```



Predicted graph after the model is trained.

```
sns.catplot(x="PetalWidthCm",y="PetalLengthCm",hue="Species_pred_agglo" ,data=x_data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f3fd89bb210>
```



This clustering prediction is less accurate than k-means.

4.3 Classification: -

As the name suggests, supervised data mining refers to learning algorithms used in classification and prediction. The supervised algorithm learns from the training data, which is labeled. In classification, the model is trained and tested.

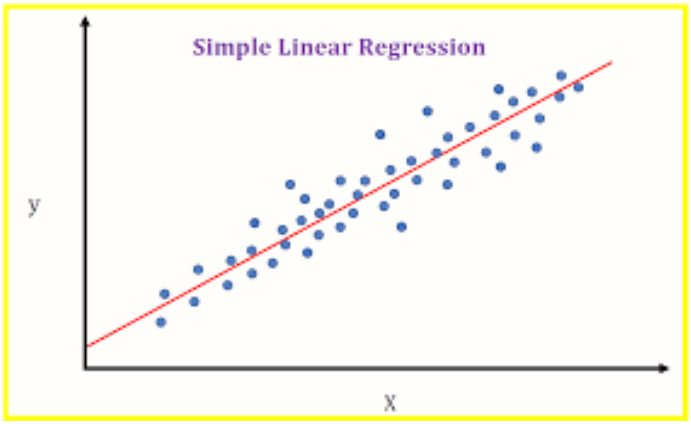
4.3.1. Linear Regression: -

Linear regression is a **linear model**, e.g., a model that implies a linear relationship between the input variables (x) and the single output variable (y). More specifically, y can be calculated from a linear combination of the input variables (x). Here we have trained the dataset, and after that, we have tested by predicting and calculating the accuracy. Formula used:

Line equation=mx+b

Where b is an intercept, m is the slope of the graph, which represents predicted data, and x is actual data without applying linear regression.

After applying linear regression, if these dots lie on this red line(slope), our model accuracy is 100%, which is impossible.



Implementation of linear regression Algorithm: -

- From the scikit-learn use linear-model module to import Linear Regression class
- Then create an object model of that class.
- After that, apply the fit method on the x, y train dataset. The fit method is used to train the model; its object is created, as shown in the figure below.
- Then import NumPy library.
- On the model, the object applies to predict method and pass x (features) test data based on which model will predict the classes.
- This will then typecast to list which is named y_predicted_values.
- However, this will return an array with decimal values, and we want rounded values to visualize the plot easily.
- To round values, we use for loop.
- Finally, the array is displayed, showing three classes of Iris category, which the model predicted.
- Then we append the predicted value to strat_test_set[‘pred_species_linear’] to compare with the actual data before training and testing after the model.
- So here, we train and test the model whether it is predicting correctly or not, and this will be calculated by accuracy.

```
[ ] from sklearn.metrics import accuracy_score
    from sklearn.linear_model import LinearRegression
    model = LinearRegression()
```

```
[ ] model.fit(x_data_train,y_data_train)

    LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
    import numpy as np
    y_predicted_values = list(model.predict(x_data_test))
    y_predicted_round = []
    for i in y_predicted_values:
        a = round(i)
        y_predicted_round.append(a)
    np.array(y_predicted_round)

    array([0, 2, 1, 1, 0, 1, 0, 0, 2, 1, 2, 2, 2, 1, 0, 0, 0, 1, 1, 2, 0, 2,
           1, 2, 2, 1, 1, 0, 2, 0])
```

```
[ ] strat_test_set['pred_species_linear'] = y_predicted_round
```

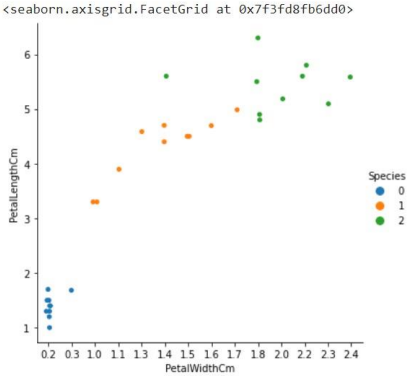
Output: -

The scatter plot is made using the seaborn library having labels as Petal Length and Petal Width Comparing the output of the original dataset and output generated after the prediction or training the model. As seen, two plots are differentiated by hue parameter. Value of original has hue=' Species' and trained has hue= 'pred_species_linear'. Here classes are categorized by numeric value because we have converted the categorical data into numerical using Label Encoder.

Here numerical values of classes represent: -
0=Iris Setosa
1=Iris Versicolor
2=Iris Verginica

Original Data Plot before classification: -

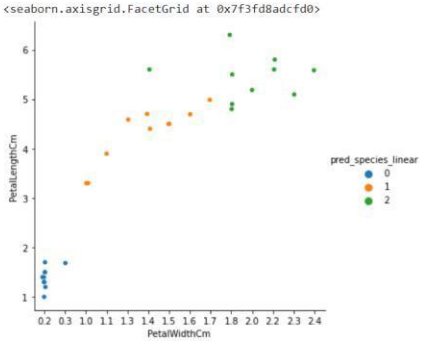
```
    sns.catplot(x="PetalWidthCm",y="PetalLengthCm",hue="Species" ,data=strat_test_set)

    <seaborn.axisgrid.FacetGrid at 0x7f3fd8fb6dd0>
```



Predicted values plot after classification: -

```
[ ] sns.catplot(x="PetalWidthCm",y="PetalLengthCm",hue="pred_species_linear",data=strat_test_set)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f3fd8adcfd0>
```



Accuracy: -

The accuracy rate shows how correctly our model predicted the classes. Our model has a **1.0** accuracy score which is 100% accuracy, but no model will be 100%accurate; our model has very little data that is why it is showing this accuracy score.

Accuracy Calculated: -

- From scikit-learn use metrics module to import accuracy_score
- Then pass two parameters to calculate accuracy: -
    1. strat_test_set[‘Species’]
    2. strat_test_set[‘pred_species_linear’]
- These parameters are passed for comparing the actual target test value with the predicted value.

```
[ ] print("Accuracy Score : ",accuracy_score(strat_test_set["Species"], strat_test_set["pred_species_linear"]))
```

```
Accuracy Score :  1.0
```

4.3.2. Decision Tree:

Here also we are predicting the Iris Category by Training the model with a decision tree algorithm. Decision Tree is used to generate a training model that can be used to predict the class or value of the target variable by learning simple decision rules inferred from preliminary data (training data).In Decision Trees, we start from the tree's root for predicting a class label for a record. We compare the values of the root attribute with the record's attribute. Based on the comparison, we follow the branch corresponding to that value and jump to the next node. It forms the tree where the target column is the root node, and here our target is Species based on which classification is performed.

Implementation of Decision Tree Algorithm: -

- From the scikit-learn library, use the tree module to import DecisionTreeRegressor class.
- Then create an object named model of that class.
- After that, apply the fit method on the x,y train dataset. The fit method is used to train the model; its object is created, as shown in the figure below.
- Then import NumPy library.
- On the model (object), apply to predict method and pass x (features) test data based on which model will predict the classes.
- This will then typecast to list which is named as y_predicted_values.
- However, this will return an array with decimal values, and we want rounded values to visualize the plot easily.
- To round the values, we use for loop.
- Finally, the array is displayed, showing three classes of Iris category, which the model predicted.
- Then we append the predicted value to strat_test_set [‘pred_species_decision’] to compare with the actual data before training and testing the model.
- So here, we train and test the model whether it is predicting correctly or not, and this will be calculated by accuracy rate.

```
[ ] from sklearn.tree import DecisionTreeRegressor
    model = DecisionTreeRegressor()
```

```
[ ] model.fit(x_data_train,y_data_train)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

```
y_predicted_values = (list(model.predict(x_data_test)))
y_predicted_round = []
for i in y_predicted_values:
    a = round(i)
    y_predicted_round.append(a)
np.array(y_predicted_round)
```

```
array([0, 2, 1, 1, 0, 1, 0, 0, 2, 1, 2, 2, 2, 1, 0, 0, 0, 1, 1, 2, 0, 2,
       1, 1, 2, 2, 1, 0, 2, 0])
```

```
[ ] strat_test_set['pred_species_decision'] = y_predicted_round
```

Output: -

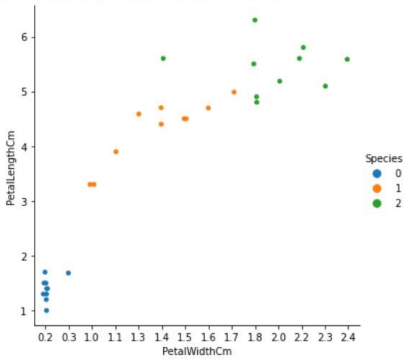
The scatter plot is made using the seaborn library having labels as Petal Length and Petal Width. The two plots compare the actual data output before classification and output generated after the prediction or after training the model. The two plots can be differentiated by hue parameter value of original plot has hue=' Species' and trained has hue= 'pred_species_decision.' Here classes are categorized by numeric value have converted the categorical data into numeric by Label Encoder.

Here numerical values of classes represent: -
0=Iris Setosa
1=Iris Versicolor
2=Iris Verginica

Original Data Plot before classification: -

```
sns.catplot(x="PetalWidthCm",y="PetalLengthCm",hue="Species" ,data=strat_test_set)
```

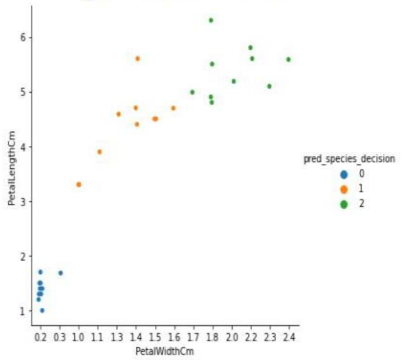
```
<seaborn.axisgrid.FacetGrid at 0x7f3fd8fb6dd0>
```



Predicted values plot after classification: -

```
[ ] sns.catplot(x="PetalWidthCm",y="PetalLengthCm",hue="pred_species_decision",data=strat_test_set)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f3fd9ee46d0>
```



Accuracy: -

The accuracy rate shows how correctly our model predicted the classes. Here our model has a 0.93333 accuracy score which is 93 % accuracy. Hence the accuracy of this model is suitable means the decision tree predicted the correct value, and our model is trained accurately.

Accuracy Calculated: -

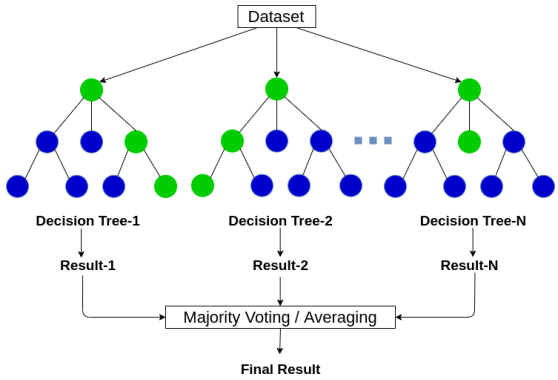
- From scikit-.learn use metrics module to import accuracy_score
- Then pass two parameters to calculate accuracy: -
  1.strat_test_set[‘Species’]
  2.strat_test_set[‘pred_species_decision’]
- These parameters are passed for comparing the actual target test value with the predicted value.

```
[ ] print("Accuracy Score : ",accuracy_score(strat_test_set["Species"], strat_test_set["pred_species_decision"]))
```

```
Accuracy Score :  0.9333333333333333
```

4.3.3. Random Forest:-

Like Linear Regression and Decision Tree, we used Random Forest to predict Iris's category according to their features. Random forest uses many decision trees, say, an ensemble. Where each tree is a little different from the others. When we get new data, we take the majority vote of the ensemble to get a final result.



Implementation of Random Forest Algorithm: -

- From the scikit-learn library, use the ensemble module to import RandomForestRegressor class.
- Then create an object named model of that class.
- After that, apply the fit method on the x, y training dataset. The fit method is used to train the model; its object is created, as shown in the figure below.
- Then import NumPy library.
- On the model (object), apply predict method and pass x (features) test data based on which model will predict the classes.
- This will then typecast to list which is named as y_predicted_values.
- However, it will return an array with decimal values, and we want rounded values to visualize the plot easily.
- To round the values, we used **for** loop.
- Finally, the array is displayed, showing three classes of Iris category, which the model has predicted.
- Then we append the predicted value to start_test_set [‘pred_species_forest’] to compare with the actual data before training and testing the model.
- So here, we train and test the model whether it is predicting correctly or not, and this will be calculated by accuracy.

```
[ ] from sklearn.ensemble import RandomForestRegressor
    model = RandomForestRegressor()
```

```
[ ] model.fit(x_data_train,y_data_train)
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

```
y_predicted_values = (list(model.predict(x_data_test)))
y_predicted_round = []
for i in y_predicted_values:
    a = round(i)
    y_predicted_round.append(a)
np.array(y_predicted_round)
```

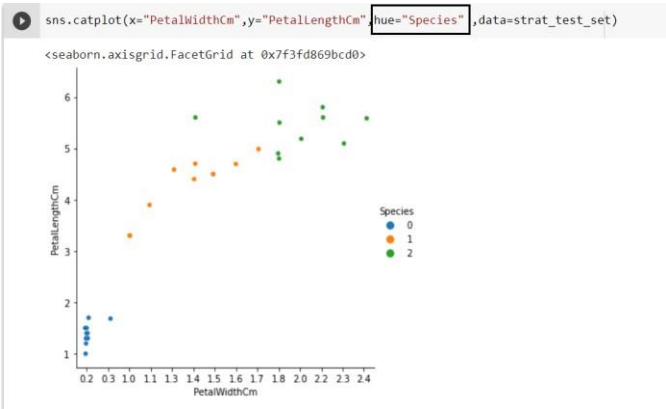
```
array([0, 2, 1, 1, 0, 1, 0, 0, 2, 1, 2, 2, 2, 1, 0, 0, 0, 1, 1, 2, 0, 2,
       1, 2, 2, 2, 1, 0, 2, 0])
```

```
[ ] strat_test_set['pred_species_forest'] = y_predicted_round
```

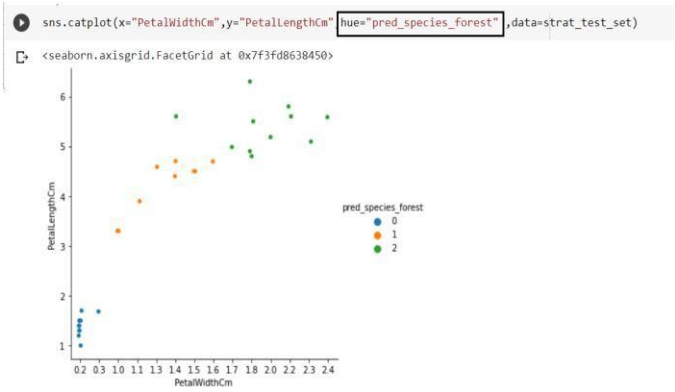
Output: -

The scatter plot is made using the seaborn library having labels as Petal Length and Petal Width comparing the output of the original dataset before classification and output generated after the prediction or training of the model. As seen, the two plots are differentiated by hue parameter. Value of original plot has hue=' Species' and trained has hue='pred_species_forest.' Here, classes are categorized by numeric value because we have converted the categorical data into numerical data using Label Encoder.
Here numerical values of classes represent: -
0=Iris Setosa
1=Iris Versicolor
2=Iris Verginica

Original Data Plot before classification: -



Predicted values plot after classification: -



Accuracy:

The accuracy rate shows how correctly our model predicted the classes. Here our model has a 0.966 accuracy score which is 96 % accuracy. Hence the accuracy of this model is excellent, which means it predicted the correct value and is trained accurately.

Accuracy Calculated: -

- From scikit-learn use metrics module to import accuracy_score
- Then pass two parameters to calculate accuracy: -
  1.strat_test_set['Species']
  2.strat_test_set['pred_species_decision']
- These parameters are passed for comparing the actual target test value with the predicted value.

IV. MATHEMATICAL MODELS

5.1 Euclidean distance: -

In K-means clustering, we have practiced the Euclidian distance to measure the mean of the Iris dataset based upon which the clusters of Iris species are determined. Euclidean is the most used measure; it is generally used when information is

$$d(x,\ y)\ =\ \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

non-stop.

x, y =two points in Euclidean n-space

Xi, Yi = Euclidean vectors, starting from the origin of the space (initial point)

n=n-space

5.2 Manhattan distance: -

In Agglomerative clustering, we have practiced this distance formula to compute the distance between two dataset points to create a cluster. Also, in K-means, this distance measure formula can be used instead of Euclidean as it is more rational to use Manhattan when two vectors are in integer feature space. The Manhattan distance between two points $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ in n-dimensional space is the sum of the distances in each dimension.

$$d(x,\ y)=\sum_{i=1}^{n}|x_i - y_i|.$$

5.3 Linear regression: -

In classification linear regression algorithm is used where this formula is referred to predict the result of an algorithm. A **linear regression** line has an equation:

Y = a + bX,

here X is the descriptive variable, and Y is the dependent variable. The slope of the line is b, and a is the intercept (the value of y when x = 0).

V. RESULT

Estimating the result is the final part of this research. For each scientific research, the final result should be tested and evaluated if that is acceptable. For every Machine learning, algorithm exceptions will always exist. To obtain the best result, analysis is necessary.
In both clustering and classification, we are predicting the clusters and labels(classes), and the result of the following algorithms are: -

6.1. Clustering Result: -

We have performed two algorithms K-means and agglomerative; from these two, the k-means predicted more accurate clusters than agglomerative because, on a larger dataset, the partitioning clustering is more accurate than hierarchical clustering.

6.2. Classification Result: -

Here we performed three algorithms Linear Regression, Decision Tree, and Random Forest, and calculated their accuracy; given below is the accuracy table to predict the result.

Sno.	Algorithms	Accuracy Score
1.	Linear Regression	0.1
2.	Decision Tree	0.93
3.	Random Forest	0.96

The best algorithm we performed is Random Forest that predicted the accurate result. Here Linear regression shows a 100% accuracy rate which is not possible; no model is 100%

correct, so we considered the best algorithm is Random Forest that predicted the correct classes of the dataset.

VI. CONCLUSION

In this paper, we have concluded which of our algorithms are best to analyze and visualize the Iris dataset and predicted the category correctly. We have used both Supervised and Unsupervised algorithms both. Some samples provide misclassified results like Linear Regression. Hence to analyze large datasets, we need machine learning algorithms.

REFERENCES

[1] C. Geetha, Raghu Ram, Nazeer Vali, "IRIS - Flower Classification", Bharath Institute of Higher Education & Research, Selaiyur, Chennai. 06 January 2017, Eurasian Journal of Analytical Chemistry ISSN , UGC Approved.

[2] Patric Granhom, " A Study Of Pattern Recognition Of Iris Flower Based On Machine Learning ", TURKU UNIVERSITY OF APPLIED SCIENCES, 22 Aug 2013.

[3] Gayatri Naik,"Classification of Iris Flower Species Using Machine Learning", Shivajirao S. Jondhle College of Engineering & Technology, Asangaon, Maharashtra, India, IJREAM 2019.

[4] Shashidhar T Halakatti1, Shambulinga T Halakatti," Identification Of Iris Flower Species Using Machine Learning", Rural Engineering College,Hulkoti – 582205 Dist : Gadag State : Karnataka Country : India, BVVS Polytechnic, Bagalkot – 587101 Dist : Bagalkot State : Karnataka Country: India, IPASJ International Journal of Computer Science (IIJCS), Volume 5, Issue 8, August 2017.

[5] https://www.geeksforgeeks.org/supervised-unsupervised- learning.

[6] https://medium.com/swlh/machine-learning-101-classification-regression-gradient-descent-and-clustering-b3449f270dbe.

[7] https://www.researchgate.net/publication/3075172 4 8_Machine_Learning.

[8] Sathya Duraisamy, Ganesh Kumar Pugalendhi, Prasanalakshmi Balaji."Reducing energy consumption of wireless sensor networks using rules and extreme learning machine algorithm."

The Journal of Engineering, Vol. 2019, Issue. 9, pp. 5443-5448. DOI:10.1049/joe.2018.5288