**REPORT ON ANN PROJECT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Artificial Neural Networks lab** | | **A picture containing text, clipart  Description automatically generated** | |
| **Semester** | **V** | **Subject** | **Artificial neural networks** |
| **Section** | **B22** | **Code** | **19EAI331P** |
| **Name** | **N. SRUTHI HASINI** | | |
| **Registration No** | **122010322017** | **Branch** | **CSE -CS / AIML** |
| **Year** | **2022** | **Date** | **22-10-22** |

***DECISION TREE ALGORITHM***

***COVID-19 DATASET***

N. SRUTHI HASINI

122010322017

B-22, AIML

**IMPLEMENTATION USING DECISION TREE ALGORITHM ON DATASET COVID-19**

AIM:

Implementation of decision tree algorithm on covid-19 dataset.

Pre-processing:

* In general, every dataset requires pre-processing to get accurate result or maximum accuracy.
* For our dataset covid-19, we applied three methods of pre-processing techniques:

1. Eliminating Null values/checking null values
2. Encoding categorial values
3. Normalization
4. Eliminating Null values/checking null values

* To check whether there are null values in our dataset, we used a function isnull()

1. Encoding categorial values

* No need to do this because there are no categorial values in our dataset, all values are binary itself.

1. Normalization

* For normalization of a specific column called location, we applied it using a function called normalize()
* As a result, location column is being normalized.

Code for pre-processing:

Code for importing the dataset:

**data\_set = pd.read\_excel('covid\_19.xlsx')**

**X = data\_set.iloc[:, :-1].values**

**y = data\_set.iloc[:, -1].values**

**print(X)**

Code for checking null values:

**data\_set.isnull()**

The output here is, false in the whole dataset in every column and row.

Code for Normalization:

**data\_set = pd.read\_excel('covid\_19.xlsx')**

**x\_array = np.array(data\_set['location'])**

**normalized\_arr = preprocessing.normalize([x\_array])**

**print(normalized\_arr)**

The output here is, normalized values displayed in location column

Code for the whole pre-processing methods:

**data\_set = pd.read\_excel('covid\_19.xlsx')**

**X = data\_set.iloc[:, :-1].values**

**y = data\_set.iloc[:, -1].values**

**print(X)**

**data\_set.isnull()**

**data\_set = pd.read\_excel('covid\_19.xlsx')**

**x\_array = np.array(data\_set['location'])**

**normalized\_arr = preprocessing.normalize([x\_array])**

**print(normalized\_arr)**

Decision tree on dataset – covid-19

* Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems.
* It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
* In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node.
* Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
* The decisions or the test are performed on the basis of features of the given dataset.
* It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
* It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

Code:

**import numpy as np**

**import pandas as pd**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.metrics import accuracy\_score**

**from sklearn.metrics import classification\_report**

**# Function for importing Dataset**

**def importdata():**

**balance\_data = pd.read\_excel('covid\_19.xlsx')**

**#The dataset length i.e, number of rows**

**#print ("Dataset Length: ", len(balance\_data))**

**#The dataset shape i.e, number of rows and columns**

**#print ("Dataset Shape: ", balance\_data.shape)**

**# dataset observations**

**#print ("Dataset: ",balance\_data.head())**

**return balance\_data**

**# Function to split the dataset**

**def splitdataset(balance\_data):**

**# Separating the target variable**

**X = balance\_data.values[:,:-1]**

**Y = balance\_data.values[:, -1]**

**# Splitting the dataset into train and test**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(**

**X, Y, test\_size = 0.2, random\_state = 100)**

**return X, Y, X\_train, X\_test, y\_train, y\_test**

**# Function to perform training with giniIndex.**

**def train\_using\_gini(X\_train, X\_test, y\_train):**

**# Creating the classifier object**

**clf\_gini = DecisionTreeClassifier(criterion = "gini",**

**random\_state = 100,max\_depth=3, min\_samples\_leaf=2)**

**# Performing training**

**clf\_gini.fit(X\_train, y\_train)**

**return clf\_gini**

**# Function to perform training with entropy.**

**def train\_using\_entropy(X\_train, X\_test, y\_train):**

**# Decision tree with entropy**

**clf\_entropy = DecisionTreeClassifier(**

**criterion = "entropy", random\_state = 100,**

**max\_depth = 3, min\_samples\_leaf = 5)**

**# Performing training**

**clf\_entropy.fit(X\_train, y\_train)**

**return clf\_entropy**

**# Function to make predictions**

**def prediction(X\_test, clf\_object):**

**# Predicton on test with giniIndex**

**y\_pred = clf\_object.predict(X\_test)**

**return y\_pred**

**# Function to calculate accuracy**

**def cal\_accuracy(y\_test, y\_pred):**

**#printing the accuracy**

**accuracy\_accquired =(accuracy\_score(y\_test,y\_pred))**

**print ("Accuracy : ",accuracy\_accquired\*100)**

**#printing the report**

**print("Report : ",classification\_report(y\_test, y\_pred))**

**# Driver code**

**def main():**

**# Building Phase**

**data = importdata()**

**X, Y, X\_train, X\_test, y\_train, y\_test = splitdataset(data)**

**clf\_gini = train\_using\_gini(X\_train, X\_test, y\_train)**

**clf\_entropy = tarin\_using\_entropy(X\_train, X\_test, y\_train)**

**# Operational Phase**

**# Prediction of results using gini**

**y\_pred\_gini = prediction(X\_test, clf\_gini)**

**cal\_accuracy(y\_test, y\_pred\_gini)**

**# Prediction of results using entropy**

**y\_pred\_entropy = prediction(X\_test, clf\_entropy)**

**# Calling main function**

**if \_\_name\_\_=="\_\_main\_\_":**

**main()**

**OUTPUT:**

**Accuracy : 95.95375722543352**

**Report : precision recall f1-score support**

**0.0 0.98 0.97 0.98 152**

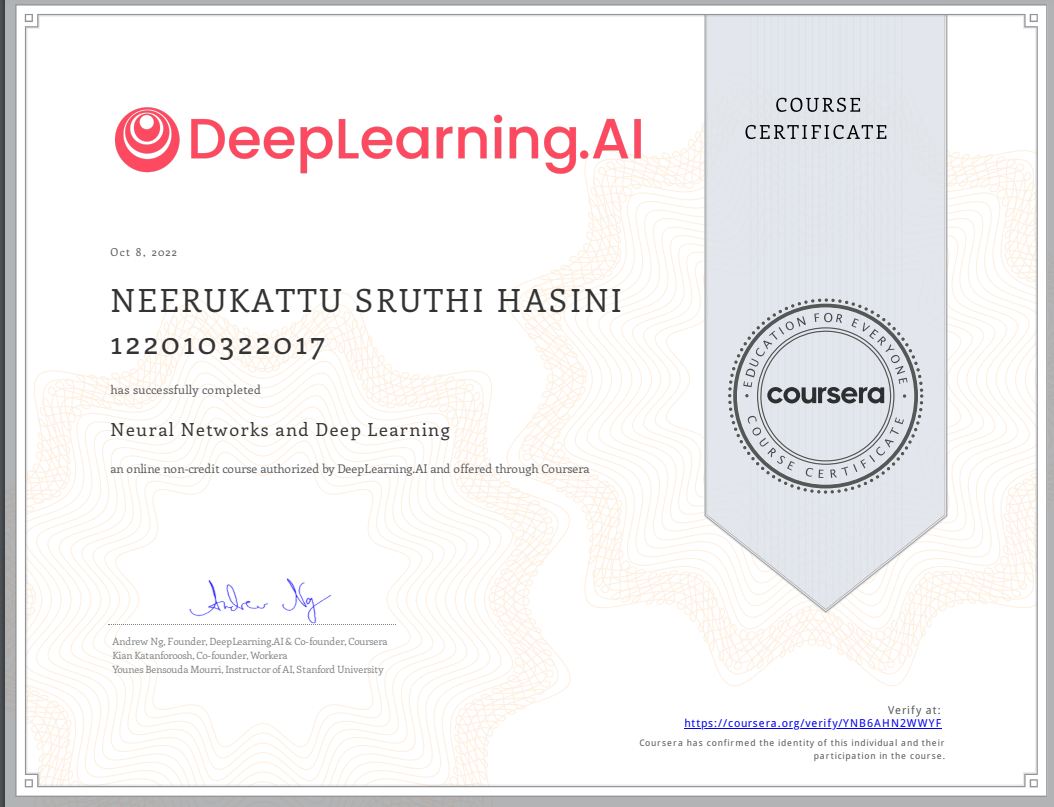
**1.0 0.82 0.86 0.84 21**

**accuracy 0.96 173**

**macro avg 0.90 0.92 0.91 173**

**weighted avg 0.96 0.96 0.96 173**

Coursera certificate:



Link:

<https://www.coursera.org/account/accomplishments/verify/YNB6AHN2WWYF>

References

<https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm>