Kingdom of Saudi Arabia
Ministry of Education
King Faisal University
College of Computer Sciences & Information Technology

Crop Recommendation System

A project submitted in partial fulfillment of the requirements for the Artificial Intelligence Course

Team Members

Batool Alsowaiq (ID: 219015035)

Noura almosinad (ID: 219010636)

Roaa Albahrani (ID: 219022805)

Supervised by

Ms. Nura A. Aljaafari

14/5/2022

Project title

Crop recommendation project

Table of Contents

Table of Figures

Introduction

Machine learning (ML) is a subset of artificial intelligence that builds an automated model that can learn from a data sample, make decisions, and recognize patterns from computational learning theory without being programmed to achieve a desired outcome. With the explosion of data, the importance of machine learning is here because it can process large and complex data and obtain results with high speed and accuracy even at extremely large scale. Machine learning has several types of machine learning algorithms, but the three mainly used methods are supervised learning, unsupervised learning, and reinforcement learning. In reinforcement learning, the algorithm learns via trial and error which gives the best rewards. Unsupervised machine learning is used for data with no historical labels therefore the algorithm must discover what is being shown since the system does not have the correct answer. In supervised machine learning, the system has training set of data with result vectors [1], which we will be focusing on it in this project. ML algorithms could be applied in different industries such as in health care, financial services, government, and oil and gas. There are three main tasks that could be done by ML models which they are classification, regression, and clustering. Classification is good for predicting a discrete value, checking whether the input data belongs to a particular class. Regression helps in continuous data, and it estimates the right value of a continuous data. Clustering is an approach which allows us to join different records together and assign them to groups depending on their similarities. To train a machine, we must create a model that is trained on training data, and it will be able to process extra data to generate a prediction. There are many distinct types of models that are used for machine learning systems which they are Linear Regression, Deep Neural Networks, Logistic Regression, Decision Trees, Linear Discriminant Analysis, Naive Bayes, Support Vector Machines, Learning Vector Quantization, K-nearest Neighbors, and Random Forest [2]. In this project, we are trying to create a predictive system that can recommend suitable crops to grow on a particular farm [3].

Literature Review

Many researchers have published articles about machine learning applications in several fields, one of the articles that widely published to journal is crops recommendation, which a system that is able to generate most appropriate crops recommendation in some certain farms. Applying an effective machine learning model would give us the targeted results. In paper [16], researchers discussed and implemented three types of ML models for crop prediction which they are K-nearest Neighbor, Decision Tree, and Random Forest Classifier using two different criterions Gini and Entropy. The two metrics Gini and Entropy are used to choose how to split a tree. Gini index is the probability that any element of the dataset will be incorrectly labeled if it the element is randomly chosen from the set, the formula of calculating it as follows: $GiniIndex$ = 1–$\sum jp2j$, where pj represents the probability of class j.

Entropy is used to measure the disorder in a distribution, calculated using this formula:

$Entropy = -\sum jpj \cdot log2 \cdot pj$, pi stands for the probability of class j [17]. The paper eventually shows that Random Forest model generates the best prediction among the three based on their accuracy.

In paper [18], the crop recommendation system was implemented to serve Indigenous countries. This system is an informed system to decide about the suitable crop to grow depending on certain parameters like Nitrogen, Phosphorous, Potassium, PH Value, Humidity, Temperature, and Rainfall. The researchers applied various machine learning algorithms like Decision Tree, Naïve Bayse (NB), Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF) and XGBoost. Machine learning methods used in the paper are [19]:

- Decision Tree: uses a branching method to explicate each outcome of a decision.
- Naïve Bayse (NB): classifies each value independently from any other value.
- Support Vector Machine (SVM): it basically groups data into categories, by providing a set of training examples, each set joined to the one or the other of the two categories it belongs to.
- Logistic Regression (LR): it estimates the probability of an event depending on the data provided previously.
- Random Forest (RF): it is an ensemble learning algorithm and it combines multiple algorithms to produce better results for classification, regression.
- XGBoost: it is a decision-tree-based ensemble learning method, it uses a gradient boosting framework [20]. The evolution of tree-based algorithms over the years shown in (Figure 0).

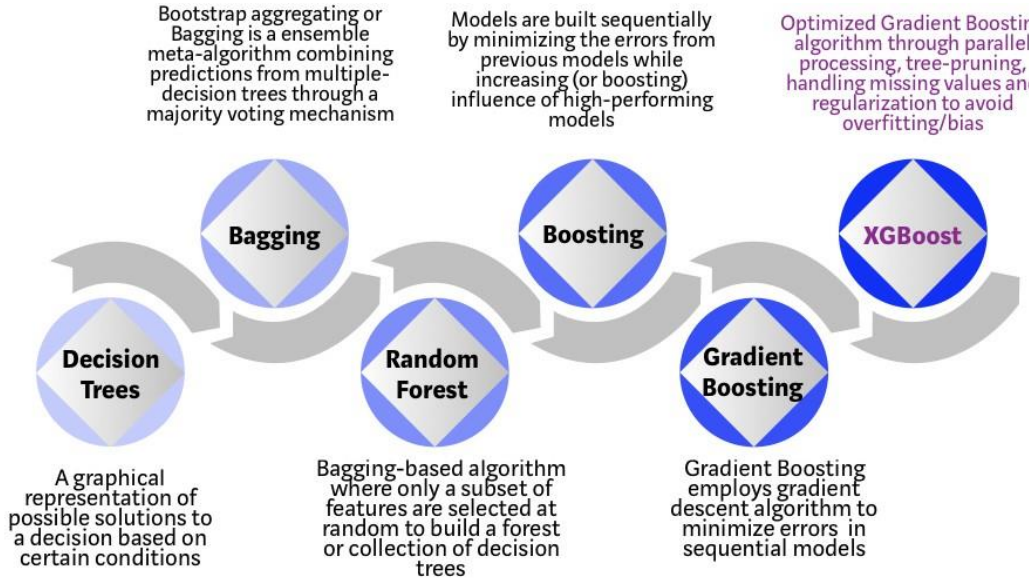

Fig.0. Evolution of XGBoost Algorithm from Decision Trees

After applying all these methods, the researcher found out that XGBoost generated the best accuracy outcome out of the six algorithms.

Methodology

Data preprocessing

When it comes to creating a Machine Learning model, data preprocessing is the first step marking the initiation of the process. Typically, real-world data is incomplete, inconsistent, inaccurate (contains errors or outliers), and often lacks specific attribute values/trends. Data preprocessing, it helps to clean, format, and organize the raw data, thereby making it ready-to-go for Machine Learning models. There are seven significant steps in data preprocessing in Machine Learning as the following [1]:

1. Acquire the dataset
2. Import all the crucial libraries
3. Import the dataset
4. Identifying and handling the missing values
5. Encoding the categorical data
6. Splitting the dataset
7. Feature scaling

Acquire the dataset

We required a dataset as a machine learning model completely works on data. The collected data for a particular problem in a proper format is known as the dataset. The dataset used in our project is Crop recommendation dataset as CSV file from Kaggle. The dataset contains eight columns, 2200 record and 17600 fields. The columns contain N, which is the ratio of Nitrogen content in soil, P which is the ratio of Phosphorous content in soil, K which is the ratio of Potassium content in soil, temperature which is temperature in degree Celsius, humidity which is the relative humidity in %, PH which is the PH value of the soil, rainfall which is rainfall in mm and label which is the classification's name. In each records the datatype for each column is int64 for N, int64 for P, int64 for K, float64 for temperature, float64 for humidity, float64 for PH, float64 for rainfall and object for label.

Import all the crucial libraries

Import all the crucial libraries to perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing such as NumPy, Matplotlib and Pandas. The first library is NumPy, which is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports to add large, multidimensional arrays and matrices. The second library is matplotlib, which is a Python 2D plotting library, and with this library, we need to import a sub-library Pyplot. This library is used to plot any type of charts in Python for the code. The last library is the Pandas library, which is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library [1]. They imported as showed (Figure 1).

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Fig. 1. Code in python for importing important libraries for data preprocessing

Importing the Datasets

we need to import the crop recommendation dataset which we have collected for our machine learning project. After import crop recommendation dataset. We should read dataset using pandas library. We can read it as pd.read_csv("crop_recommendation_dataset.csv").Then, we should use head() to display the first portion of dataset as in (Figure 2).

```
In [2]: #step 1 read dataset
        data = pd.read_csv("Crop_recommendation.csv")
        data.head()
```

Out[2]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

Fig. .2 the head part of dataset

Moreover, we can use many built-in functions to get more information about dataset such as describe (): method returns description of the data in the Data Frame [2]. For each column, it contains description as the following:

1) Count: The number of not-empty values.
2) Mean: The average (mean) value.
3) Min: The minimum value.
4) Max: the maximum value.
5) 25%: The 25% percentile
6) 50%: The 50% percentile
7) 75%: The 75% percentile

Also, we can get the size, shape and columns' name for the dataset as showed in (Figure 3). we can display the datatypes for each column and dependent variable as in (Figure 4).

```
In [5]: print("dataset size",data.size)
        print("dataset shape",data.shape)
        print("dataset columns",data.columns)
        print("-----------------------------------------------------")
        print("dataset describe")
        data.describe()

dataset size 17600
dataset shape (2200, 8)
dataset columns Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')
-----------------------------------------------------
dataset describe
```

Out[5]:

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

Fig. 3. Size, shape, and columns' name for the dataset

```
In [6]: print("the datatype for each column")
        data.dtypes

the datatype for each column
```

Out[6]:

```
N               int64
P               int64
K               int64
temperature   float64
humidity      float64
ph            float64
rainfall      float64
label          object
dtype: object
```

```
In [8]: print("the data in label column",data['label'].unique())

the data in label column ['rice' 'maize' 'chickpea' 'kidneybeans' 'pigeonpeas' 'mothbeans'
 'mungbean' 'blackgram' 'lentil' 'pomegranate' 'banana' 'mango' 'grapes'
 'watermelon' 'muskmelon' 'apple' 'orange' 'papaya' 'coconut' 'cotton'
 'jute' 'coffee']
```

Fig. 4. the datatypes for each column and dependent variable in dataset

Handling Missing data

Handling Missing data to avoid creating a big problem to machine learning model. There are two common ways to handle missing data. First way is deleting a specific row which contains null values. The second way is calculating the mean of the column or row that contain missing values. There are several methods for showing if the dataset has missing values [1]. The first method is, isnull () method as showed in (Figure 5). It is showing a table with Boolean values. The second method is, isnull (). sum () method as showed in (Figure 6). It displays numerical values represent the number of missing values in each column. As we see in below image, the dataset has not any Nan or missing values. Zero indicates that there are no missing values in a record.

```
In [9]: data.isnull()
```

Out[9]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
2195	False	False	False	False	False	False	False	False
2196	False	False	False	False	False	False	False	False
2197	False	False	False	False	False	False	False	False
2198	False	False	False	False	False	False	False	False
2199	False	False	False	False	False	False	False	False

2200 rows × 8 columns

Fig. 5. Code in python for showing a table with Boolean values

```
In [3]: #step 2 check the null value in the data set
        data.isnull().sum()

Out[3]: N              0
        P              0
        K              0
        temperature    0
        humidity       0
        ph             0
        rainfall       0
        label          0
        dtype: int64
```

Fig. 6. Code in python to display numerical values represent the number of missing values in each column.

Also, we can check if the depend on variable is balanced or not using value_count (). It shows count of each label. If the difference is large between them, we can say that the dataset is imbalanced. Therefore, we should use methods such as sampling to make them balance. In our case, the dataset is balanced all count of label are same as shown in (Figure 7).

```
In [4]: #step 3  count entries which have the same label
        data['label'].value_counts()

Out[4]: rice           100
        maize          100
        jute           100
        cotton         100
        coconut        100
        papaya         100
        orange         100
        apple          100
        muskmelon      100
        watermelon     100
        grapes         100
        mango          100
        banana         100
        pomegranate    100
        lentil         100
        blackgram      100
        mungbean       100
        mothbeans      100
        pigeonpeas     100
        kidneybeans    100
        chickpea       100
        coffee         100
        Name: label, dtype: int64
```

Fig. 7. Code in python for showing count of each label

Encoding Categorical data

Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So, it is necessary to encode these categorical variables into numbers [1]. In our case, after converting to numerical data, we get a range of numbers from 0 to 21 as in (Figure 8).

```
In [5]: # step 4 # change label from string to numeric value
        LE = LabelEncoder()
        data['label'] = LE.fit_transform(data['label'])
        print(LE.classes_)
        print(np.sort(data['label'].unique()))
        data.head()

        ['apple' 'banana' 'blackgram' 'chickpea' 'coconut' 'coffee' 'cotton'
         'grapes' 'jute' 'kidneybeans' 'lentil' 'maize' 'mango' 'mothbeans'
         'mungbean' 'muskmelon' 'orange' 'papaya' 'pigeonpeas' 'pomegranate'
         'rice' 'watermelon']
        [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21]
```

Out[5]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	20
1	85	58	41	21.770462	80.319644	7.038096	226.655537	20
2	60	55	44	23.004459	82.320763	7.840207	263.964248	20
3	74	35	40	26.491096	80.158363	6.980401	242.864034	20
4	78	42	42	20.130175	81.604873	7.628473	262.717340	20

Fig. 8. encode the categorical variables into numbers

Splitting the dataset

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also extremely high, but we provide a new dataset to it, then it will decrease the performance. So, we always try to make a machine learning model which performs well with the training set and with the test dataset. Training Set is a subset of dataset to train the machine learning model, and we already know the output. Test set is a subset of dataset to test the machine learning model, and by using the test set, model predicts the output [1]. The independent variables hold by x and the dependent variable holds by y. We pass 0.2 in test size meaning that the test size is taking 20% and remaining 80% for training. Showed in (Figure 9).

```
#divide our dataset into a training set and test set.
x= data[['N','P','K','temperature','humidity','ph','rainfall']]
y = data['label']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 42)
#x_train: features for the training data
#x_test: features for testing data
#y_train: Dependent variables for training data
#y_test: Independent variable for testing data
#random_state is used to set a seed for a random generator
```

Fig. 9. Code to divide our dataset into a training set and test set

Feature scaling

Feature scaling is the last step of data preprocessing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range [4]. There are few ways we can do feature scaling as below:

1. Min Max Scaler
2. Standard Scaler
3. Max Abs Scaler
4. Robust Scaler
5. Quantile Transformer Scaler
6. Power Transformer Scaler
7. Unit Vector Scaler

$$x_{new} = \frac{x - \mu}{\sigma}$$

In our project, we selected Standard Scaler for doing feature scaling. The Standard Scaler assumes data is normally distributed within each feature and scales them such that the distribution centered around zero, with a standard deviation of 1. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. If data is not normally distributed, this is not the best Scaler to use [4]. Showed in (Figure 10) For illustration.

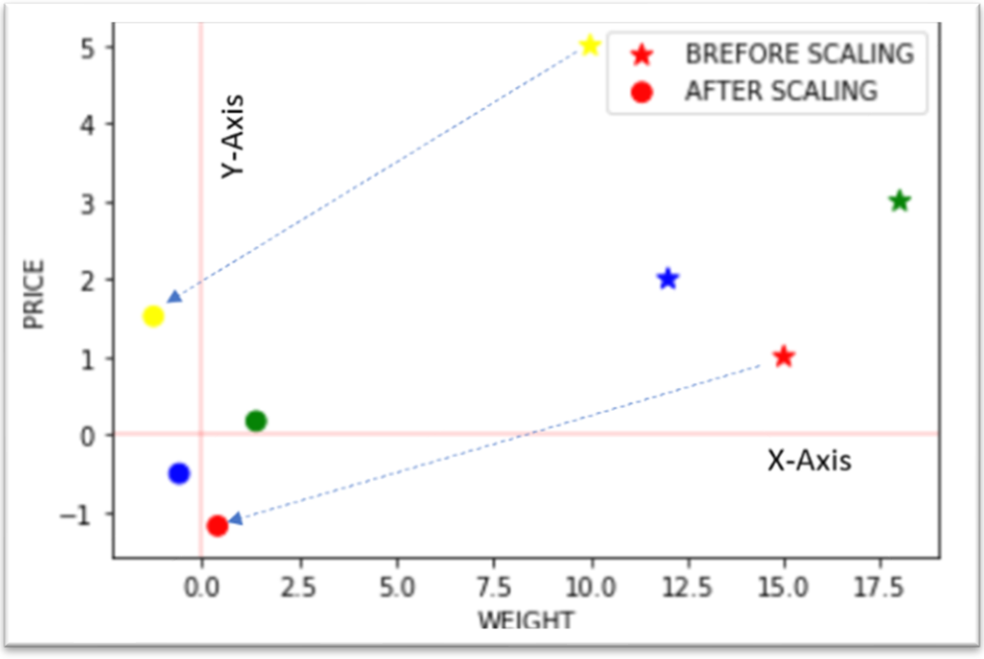

Fig. 10. Illustration graph for feature scaling

How to select ML model

Model selection is the process of choosing a suitable machine learning model from several models for a training dataset. The process could be applied on several types of models or even on same type of models and it is useful for solving complex problems using exceptionally large data with high accuracy and minimum costs. We can train a machine for classification or regression predictive model, but in case we do not know which model will perform best on this problem, we must fit and evaluate of diverse models on the problem. To select the best model, we must determine the type of machine learning algorithm that we will working on it which will be supervised machine learning for our dataset. Supervised learning is beneficial in two areas: classification and regression problems. The incompleteness of the data sample has statistical noise in the data, and limitations of each different model type which will cause the model to have some predictive error. Before choosing the final mode, we should follow these guidelines for selecting machine learning models: collect data, check for anomalies, missing data and clean the data, perform statistical analysis, and initial visualization, build models, check the accuracy, and present the results [11]. Part of following these guidelines is trying various models. Therefore, we examined our problem with four different models which they are [10]:

- Random Forest: It is useful in both regression and classification problems. It performs using multiple decision trees each trained on different data subsets and merges the results together to produce more accurate predictions.
- Support Vector Machines: It is fast and efficient model that analyses limited data. It is applicable for binary classification problems.
- K-nearest neighbors: This is simple model, works on assumption of similarity of exist data, and used for solving regression and classification problems.
- Decision Trees: It is extremely popular, simple, and efficient model used for both regression and classification problems. This model is useful to achieve final decision based on the data from past decisions.

After we applied all these models, we have obtained the best model depending on the high accuracy which was achieved by Random Forest model.

Random forest algorithm

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. It can handle categorical variables in the case of classification. Our project, it provides solution with multiclass problem. Random forest works on the Bagging principle. also known as Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation [5] (Figure 11) will illustrate it.

Random forest algorithm steps:

1. In Random Forest n number of random records are taken from the data set having k number of records.
2. Individual decision trees are constructed for each sample.
3. Each decision tree will generate an output.
4. Final output is considered based on Majority Voting or Averaging for Classification and regression, respectively.

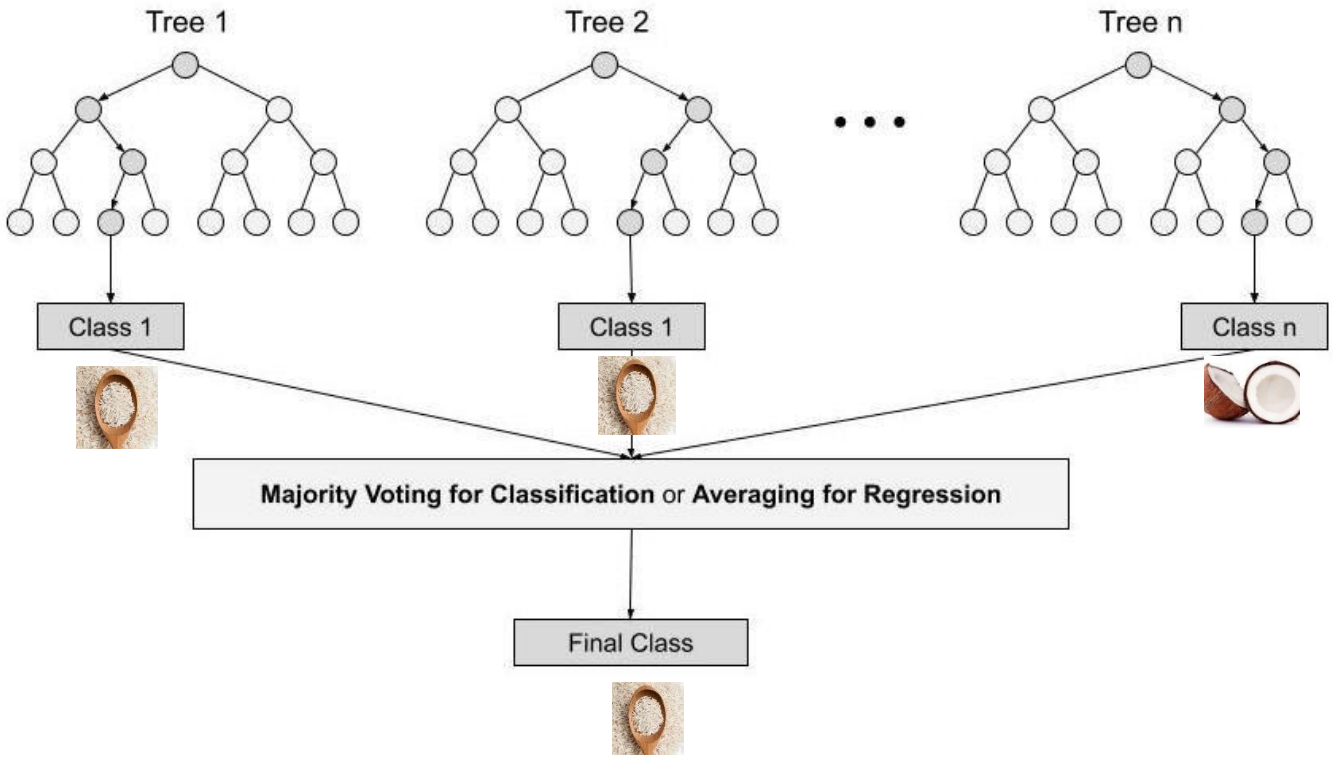

Fig. 11. Showing a process visualization of Random Forest algorithm

We give the Crop recommendations dataset to Random Forest algorithm. It takes random records from the dataset. Then, it generates induvial trees. Each tree will generate out put such as rice, rice, and coconut. The final output depends on majority voting. If the majority tree gives output as a rice when compared to coconut. So, the final output is rice.

Important Features of Random Forest

1. Diversity: Not all attributes/variables/features are considered while making an individual tree, each tree is different.
2. Immune to the curse of dimensionality: Since each tree does not consider all the features, the feature space is reduced.
3. Parallelization: Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.
4. Train-Test split: In a random forest we do not have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
5. Stability: Stability arises because the result is based on majority voting/ averaging.

We implemented Random Forest algorithm in (Figure 12).

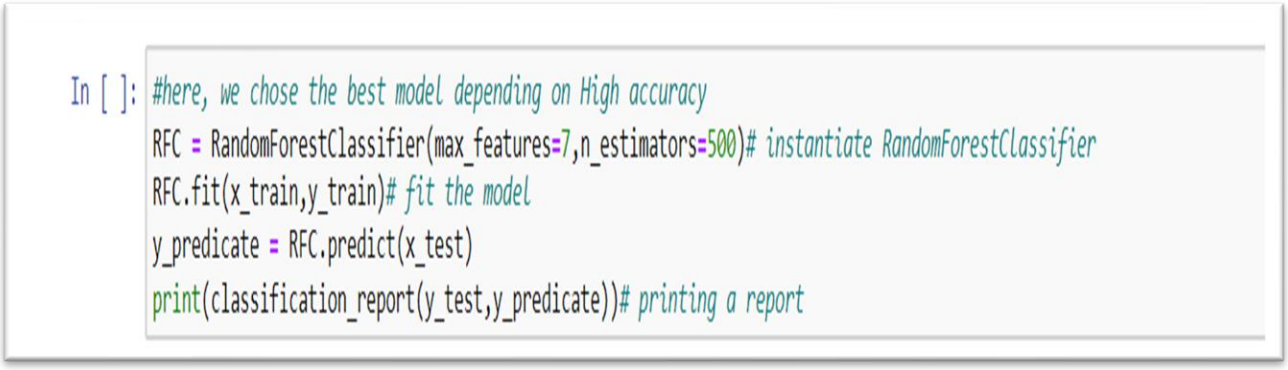

Fig. 12. Code for implemented Random Forest algorithm

In the first line, we create an object form random forest classifier with passing two arguments max features which is the number of features to consider when looking for the best split and n-estimator which is the number of trees in the forest. In the second line, we use fit method which taking x _train and y_train that we produced them in splitting the dataset step. Fit method is building a forest of trees from the training set (x, y).in the third line, we use predict method with passing into it a x_test to predict class for x. In the fourth line, we are just printing a report [3].

Evaluation Metrics

After training the machine learning model using train set of data, it is important to use evaluation metrics in order to know how well the performance of ML model is. There are different evaluation metrics for a separate set of machine learning algorithms. Classification metrics used for evaluating classification models and regression metrics used evaluating regression models. We will discuss only the Classification Evaluation Metrix since our selected dataset 'Crop Recommendation 'belongs to this category [12].

Classification Evaluation Metrics

It is used in classification supervised learning algorithms where the dataset that we give to ML model has set of features in which any combination of them gives different label. There are different approaches for this type of evaluation metrics [13]:

- Confusion matrix
- Accuracy
- Precision
- Recall
- Specificity
- F1 Score, etc.

Confusion Matrix is a tabular summary of number of true and false predicate made by the ML classifier model to measure its performance. It considers as a useful tool for visualizing the measurement of Accuracy, Precision, Recall, and AUC-ROC curve. It simply shows n-dimensional, where n indicate the number of classes in the dataset. It consists of four parts:

- True positive (TP): It indicate the situation where both actual value and the predicated value by ML model are positive.
- True negative (TN): It indicate the situation where both actual value and the predicated value by ML model are negative.
- False positive (FP): It indicate the situation where the actual value is negative, but the predicated value by ML model is positive.
- False negative (FN): It indicate the situation where the actual value is positive, but the predicated value by ML model is negative.

Type of Confusion Matrix

- Binary classification confusion matrix:
  This kind deals with dataset that consists of two classes as output of predication. One of the classes will be represented as one and the other as 0. For example, assume that we have dataset with the classes Apple as one, and Grapes as 0. There is 8 Apple, and 7 Grapes. The test of the ML model with these classes showed five correct predications from class 1 and 5 correct predications from class 0. Remaining three false predications from class 1 and 2 from class 0. As a result, the representation of confusion matrix will be as in (Figure 13) [14].

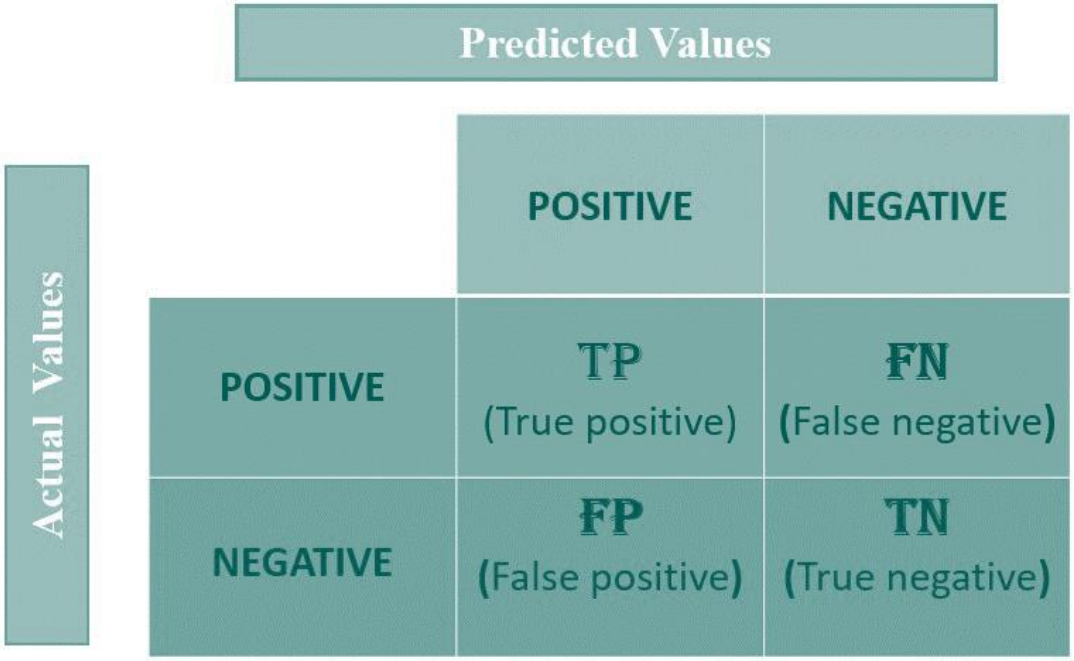

Fig. 13. Showing example of Binary Confusion Matrix

- Multi-classification confusion matrix:
  It is representation for multiclass dataset where the ML model must identify between multiple classes. Hence, the calculation of TP, FN, FP, and TN will differ than the binary confusion matrix. We must calculate them for each class as the following:
  - TP is where actual value and predicate value are same.
  - FP is summation of class's columns except TP value.
  - FN is summation of class's rows except TP value.
  - TN is summation of all value other than TP, FP, FN.

For more illustrations, let have an example of dataset consists of three classes, Versicolor, Virginia, and Setosa. ML model must classify these flowers depending on the length and width of their sepal and their petal as in (Figure 14). The representation of this dataset using confusion matrix is shown in (Figure 15).

Fig.14. Showing dataset classes

Actual Values	Predicted Values: Setosa	Predicted Values: Versicolor	Predicted Values: Virginica
Setosa	16 (cell 1)	0 (cell 2)	0 (cell 3)
Versicolor	0 (cell 4)	17 (cell 5)	1 (cell 6)
Virginica	0 (cell 7)	0 (cell 8)	11 (cell 9)

Fig.15. Showing example of Multi-Classification Confusion Matrix

Now, taking Virginica class the calculation of TP, FN, FP, and TN will be as the following:

- TP = 11 (cell 9)
- FP = 0 (cell 7) + 0 (cell 8) = 0
- FN = 0 (cell 3) + 1 (cell 6) = 1
- TN = 16 (cell 1) + 0 (cell 2) + 0 (cell 4) + 17 (cell 5) = 33

Accuracy is the most common evaluation metrics in classification algorithms. It calculates the ratio of correct predication made by the ML model over the total number of instances evaluated. It is valid technique when a dataset is balanced, in which the proportion of all instances of each class are quite similar. However, it is invalid in the opposite situation where the proportion of total number of instances per class are far than each other. It is formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision is approach that deal with imbalance dataset. It calculates ratio of correct positive predictions to the total number of instances classified in the positive class. I It is formula:

$$Precision = \frac{TP}{TP + FP}$$

Recall or it can call Sensitivity is helpful measurement when a dataset is imbalanced where the minority class is positive. It calculates the ratio of correct positive predictions to the overall number of positive instances in the dataset. It is formula:

$$Recal = \frac{TP}{TP + FN}$$

Specificity is opposite to Sensitivity in which the minority class of imbalanced dataset is negative. It calculates the ratio of correct negative predictions to the overall number of negative instances in the dataset. It is formula:

$$Specificity = \frac{TN}{TN + FP}$$

F1-Score is useful where there is significant importance to avoid false positive and false negative. This metric represents the harmonic mean between recall and precision values. It is formula:

$$F1 - Score = \frac{2(Precision \times Recal)}{Precision + Recal}$$

Our dataset is balanced, so the best approach to evaluate the ML model is Accuracy, but we will test the rest of evaluation Metrex for more efficiency. We use it to compare between the different ML models to select the best of them that fit more with our dataset. First, we initialize three lists, one to store the accuracy result of each model and the second list to store the name of it, the third list is for storing the calculation of ratio of correct positive predictions to the total number of instances classified in the positive class, the fourth list for storing the calculation of the ratio of correct positive predictions to the overall number of positive instances in the dataset, and the last list to store the harmonic mean as in (Figure 16). Then, with the help of dictionary we declare our set of models. We train each model with our x_train and y_train data as in (Figure 17).

```
#initialize lists to use them for plotting
Accuracy = []
model_name = []
prescion=[]
recall=[]
f1=[]
```

Fig.16. Showing initialized lists

```
#try different model to find which one gives best result
models = {
    RandomForestClassifier():'Forest',
    SVC():'Support Vector Machine',
    KNeighborsClassifier():'k-nearest neighbors',
    DecisionTreeClassifier():'Decision Tree',
}

for m in models.keys():
    m.fit(x_train,y_train)
```

Fig.17. Showing models training

As test for each model, we pass the x_test value to calculate model's predication. After that we measure the accuracy, precision, recall, and f1 of the model by passing model's predication and y_test value appending the result to all the lists. Finally, we print the result to visualize the accuracy, precision, recall, and f1 to compare between models as in (figure 18).

```python
#try different model to find which one gives best result
models = {
    #LogisticRegression(max_iter=1000):'Logistic',
    RandomForestClassifier():'Forest',
    SVC():'Support Vector Machine',
    KNeighborsClassifier():'k-nearest neighbors',
    DecisionTreeClassifier():'Decision Tree',
}

for m in models.keys():
    m.fit(x_train,y_train)


for model,name in models.items():
    y_pred = model.predict(x_test)#make prediction
    accuracy = accuracy_score(y_pred,y_test)#here,get the accuracy score for each model
    Precsion = precision_score(y_pred,y_test, average='micro')#here,get the precision score for each model
    Recall = recall_score( y_pred,y_test,average='micro')#here,get the recall score for each model
    F1 = f1_score( y_pred,y_test, average='micro')#here,get the F1 score for each model
    Accuracy.append(accuracy)#append the resulte with an accuracy list
    prescion.append(Precsion)#append the resulte with a prescion list
    recall.append(Recall)#append the resulte with a recall list
    f1.append( F1)#append the resulte with a f1 list
    model_name.append(name)
    #here just we are formatting the output
    formated_result=float("{:0.2f}".format(accuracy*100))
    print("Accuracy for " ,name," is : ",formated_result,"%")
    formated_result=float("{:0.2f}".format(Precsion *100))
    print("Precision for " ,name," is : ",formated_result,"%")
    formated_result=float("{:0.2f}".format(Recall*100))
    print("Recall for " ,name," is : ",formated_result,"%")
    formated_result=float("{:0.2f}".format(F1 *100))
    print("F1 for " ,name," is : ",formated_result,"%")
    print("----------------------------------------")
    #the precion, recall, f1 score will give us same reault as accuracy. We just calculate them for illustration
    #sice, our dataset is balanced so, we can count on accuracy to choose the ML model
```

```
Accuracy for  Forest  is :  99.32 %
Precision for  Forest  is :  99.32 %
Recall for  Forest  is :  99.32 %
F1 for  Forest  is :  99.32 %
----------------------------------------
Accuracy for  Support Vector Machine  is :  96.82 %
Precision for  Support Vector Machine  is :  96.82 %
Recall for  Support Vector Machine  is :  96.82 %
F1 for  Support Vector Machine  is :  96.82 %
----------------------------------------
Accuracy for  k-nearest neighbors  is :  95.68 %
Precision for  k-nearest neighbors  is :  95.68 %
Recall for  k-nearest neighbors  is :  95.68 %
F1 for  k-nearest neighbors  is :  95.68 %
----------------------------------------
Accuracy for  Decision Tree  is :  98.86 %
Precision for  Decision Tree  is :  98.86 %
Recall for  Decision Tree  is :  98.86 %
F1 for  Decision Tree  is :  98.86 %
----------------------------------------
```

Fig.18. Showing accuracy measurement of ML models

We have declared a compression plotting function to visualize the results of all evaluation Metrex lists for all models as in (Figure 19).

```
In [26]: def compartion_plotting(text1,text2,text3,list_name,color):
             """method for printing the plot for accuracy list"""
             plt.figure(figsize=[10,5],dpi = 80)
             plt.title(text1)
             plt.xlabel(text2)
             plt.ylabel(text3)
             sns.barplot(x = list_name,y = model_name,palette=color)
```

Fig.19. Comparison Plotting Function

After the results have been obtained, we visualize the accuracy, precision, recall, and f1 for all the models as below:

```
compartion_plotting('Accuracy Comparison','Accuracy','Algorithm',Accuracy,'rocket')
```

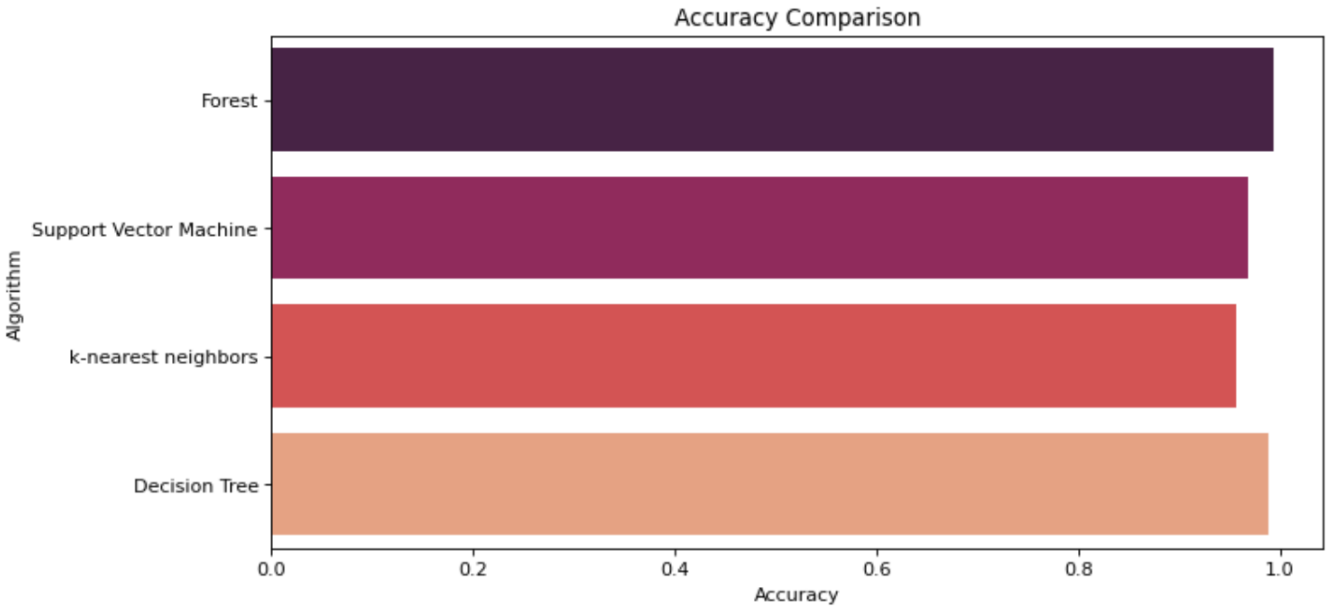

Fig.20. Visualizing the accuracy over all models

```
compartion_plotting('Prescion Comparison','Prescion','Algorithm',prescion,'mako')
```

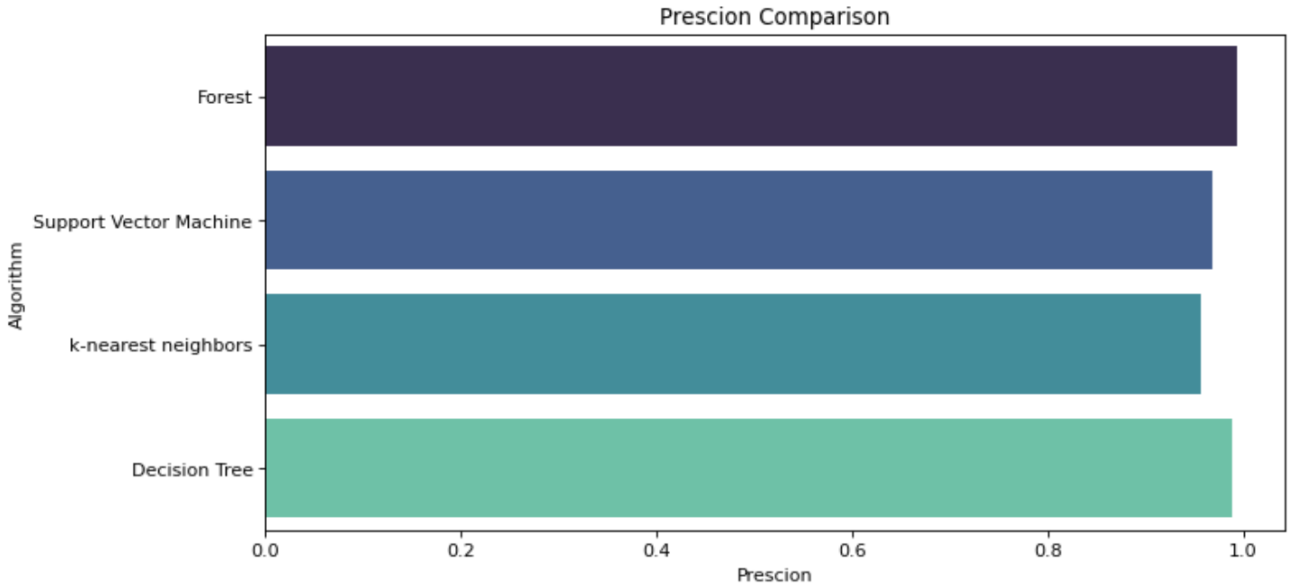

Fig.21. Visualizing the precision over all models

```
compartion_plotting('Recall Comparison','Recall','Algorithm',recall,'rocket_r')
```

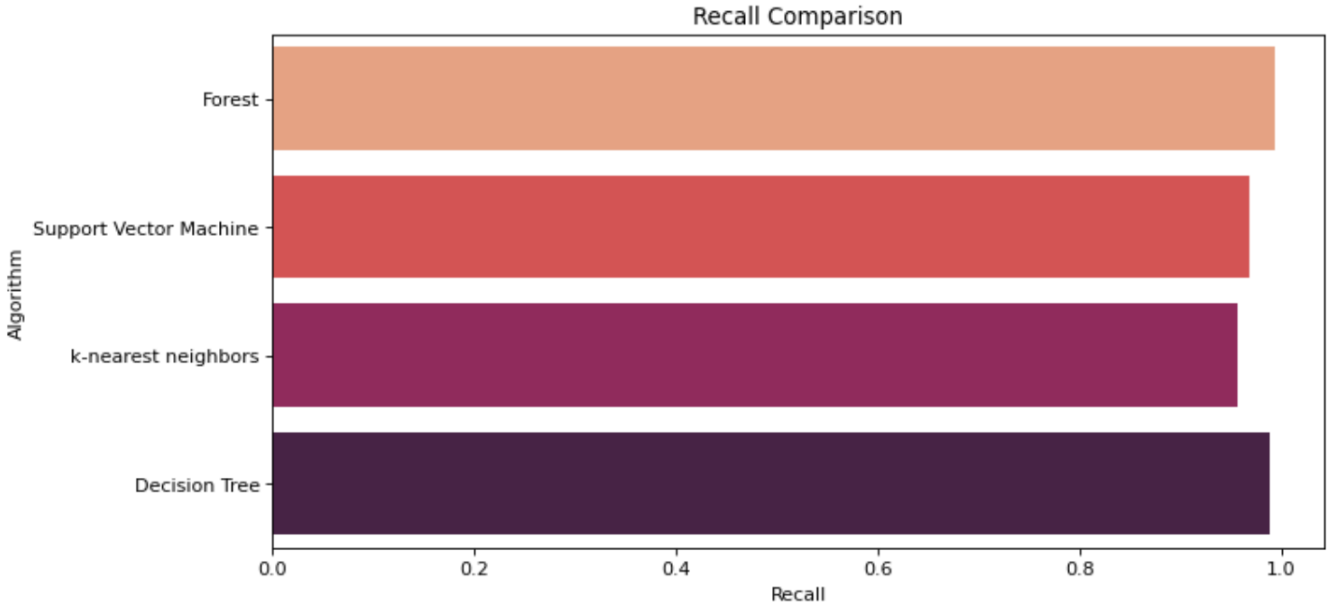

Fig.22. Visualizing the recall over all models

```
compartion_plotting('F1-Score Comparison','F1-Score','Algorithm',f1,'viridis')
```

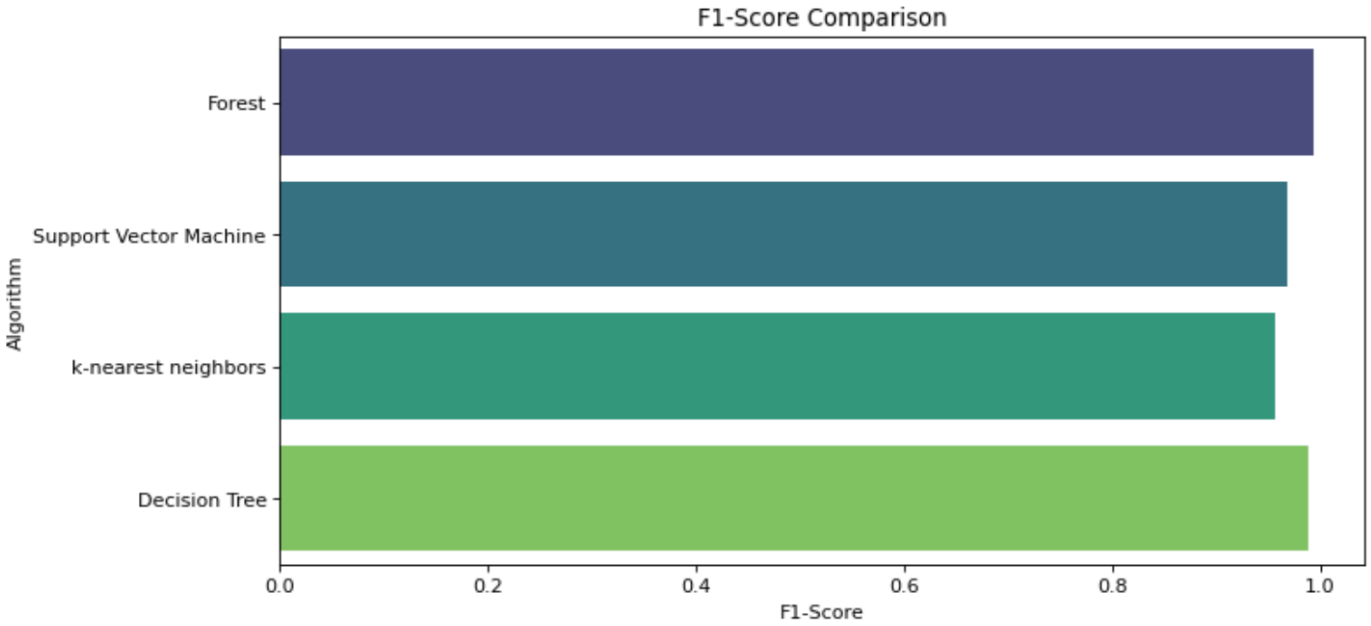

Fig.23. Visualizing the F1 over all models

Also, we use the confusion matrix to visualize the performance measurement of the selected ML model which it is Random Forest. First, we pass y_test and y_predicate of our ML model to confusion matrix function to build the confusion matrix as in (Figure 19). Then with the help of heatmap function, which is a graphical representation of data where values are depicted by color, we display the matrix as in (Figure 24 and Figure 25).

```
#here we are printing confusion matrix for selecting ML model
plt.rcParams['figure.figsize'] = (10, 10)
cnf_matrix = confusion_matrix(y_test,y_predicate)
sns.heatmap(pd.DataFrame(cnf_matrix), annot = True,cmap = 'Blues')
plt.title('Confusion Matrix for Random Forest\n', {'fontsize':20})
plt.ylabel('Actual label\n',fontsize=20)
plt.xlabel('\nPredicted label',fontsize=20)
plt.show()
```

Fig. 24. Showing the implementation of confusion matrix and heatmap

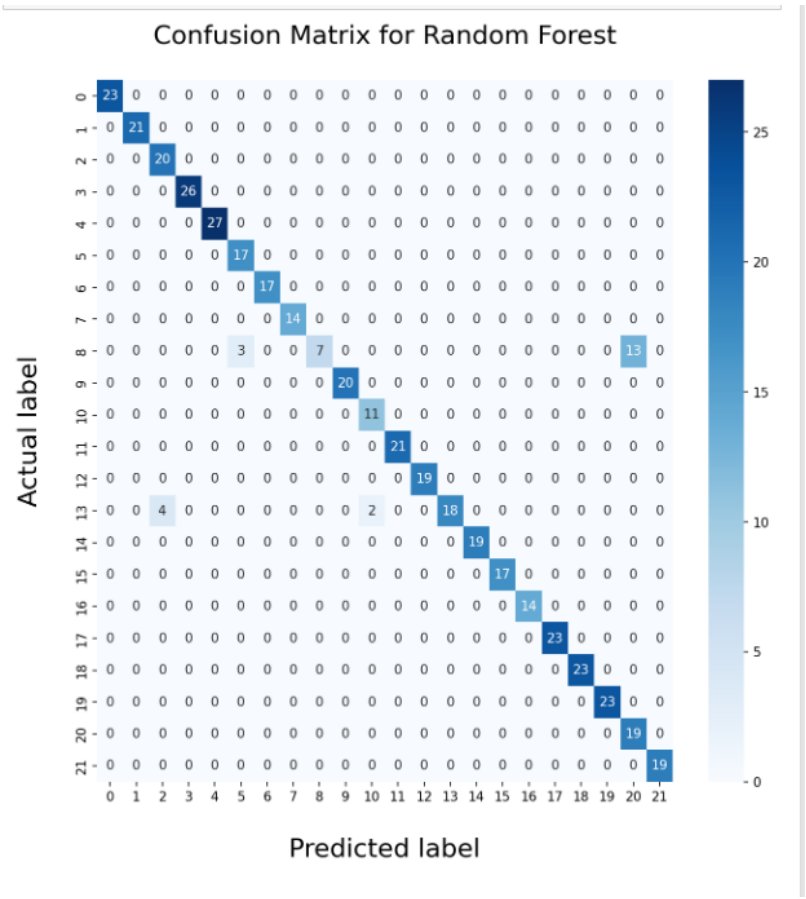

Fig. 25. Showing confusion matrix of Random Forest Model

Checking overfitting and underfitting

In machine learning overfitting and underfitting are consider as major problem that face the ML model and deteriorate its performance. Hence, the aim of each machine learning model is to reach generalization. Generalization can be defined as the ability of ML model to learn well from train set of data. Also, it can predicate correct result efficiently when it takes new data [15].

Overfitting is when the ML model tries to fit to all data from train set. It focuses too much on training data modeling itself closely around them. This leads ML model to cache noise and inaccurate values. As a result, the ML model will contain low bias and high variance. To reduce the chance of having overfitting there are different solutions shown in (Figure.26).

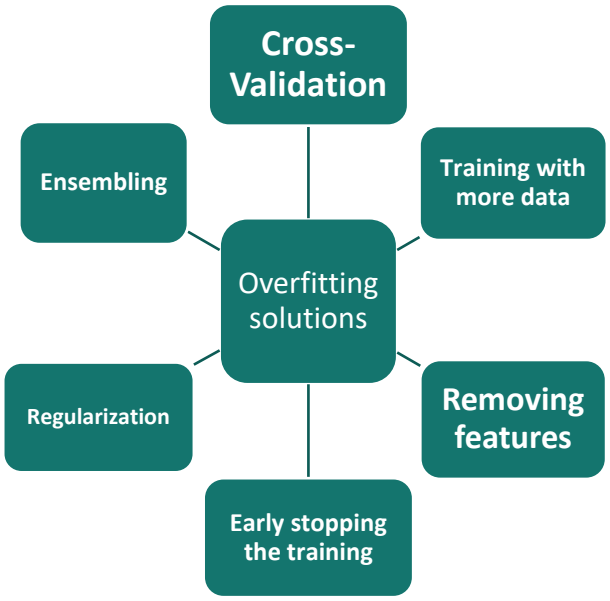

Fig.26. Showing overfitting solutions

Underfitting in contrast to overfitting, the ML does not learn enough from training data. Thus, it may fail from capturing the underling logic of data. This reduces the accuracy and generate unreliable prediction. There is two ways to avoid underfitting either by increasing the training time of the model or by increasing the number of features.

Overfitting and Underfitting Example

To illustrate the difference between the underfitting and overfitting we will apply classification example. Assume that there are two classes in the dataset cats and dogs. A good model will try to capture all data with minor errors like a quadratic function as shown in (Figure.27). This model has low loss and high accuracy [13].

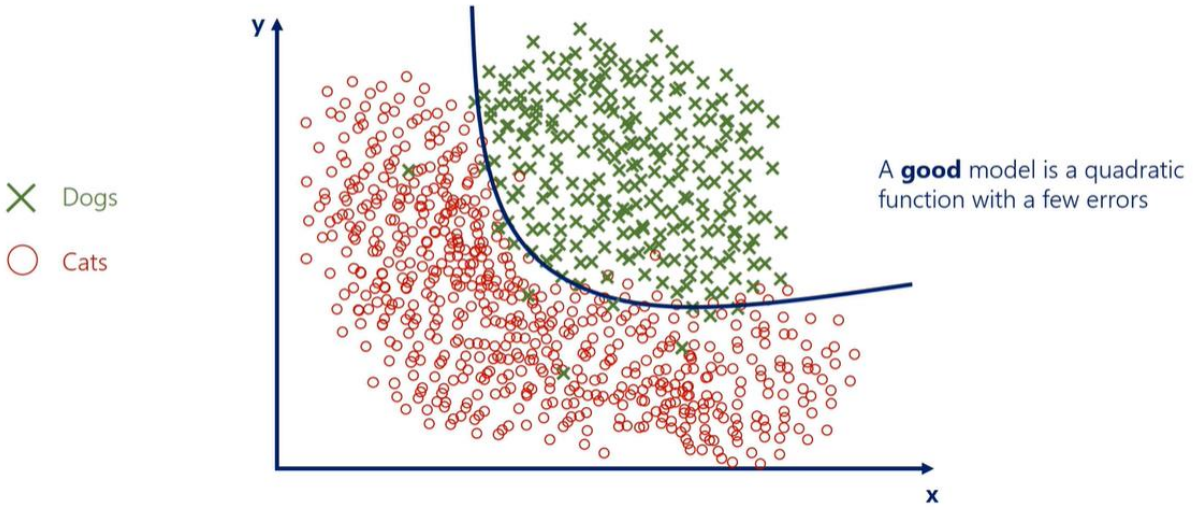

Fig.27. Showing the suitable model for cats and dogs' dataset

Underfitted model will not understand what to do with the data thus, will place line to separate the two classes as shown in f(Figure.28). This model has high loss and low accuracy.

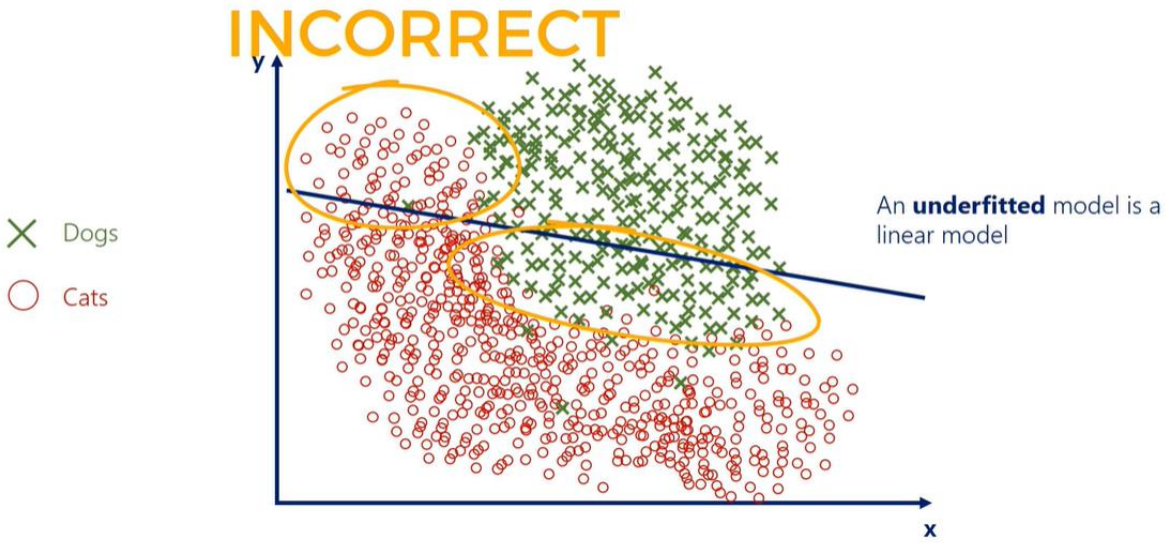

Fig.28. Showing the underfitted model for cats and dogs' dataset

while overfitted model will tries to separate all cats from all dogs thus will capture all noise as shown in (Figure.29). This model has low loss and low accuracy.

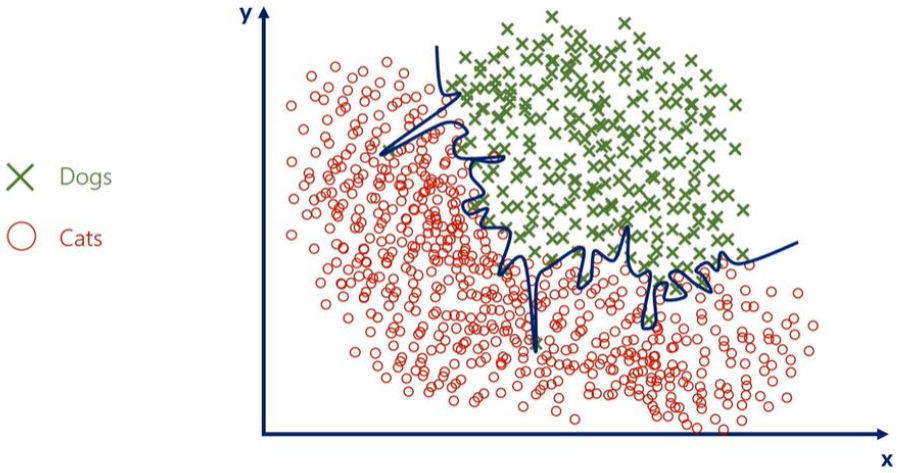

Fig.29. Showing the overfitted model for cats and dogs' dataset

In our dataset, we checked if these problems exist with our model by displaying the evaluating of model with training dataset and testing dataset. If the training set's score is similar and close to the testing set, the model reached the generalization. While the training set' score is good and the tested set got bad score, the model is overfitted. In case of underfitted, the score of both trained and tested set is bad therefore the model did not learn. After we have tested the accuracy in x-train, y-train, x-test, and y-test then we have noticed that the score both trained and tested data is close to each other as in (Figure 30)., therefore the Random Forest model has reached the generalization.

```
# print the scores on training and test set
print('Training set score: {:.2f}'.format((RFC.score(x_train, y_train))*100))
print('Test set score: {:.2f}'.format((RFC.score(x_test, y_test))*100))
```

```
Training set score: 96.65
Test set score: 95.00
```

Fig.31. Code to check the state of Random Forest Model

Hyperparameters

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster as in (Figure 31).

Following hyperparameters increases the predictive power:

1. n_estimators: number of trees the algorithm builds before averaging the predictions.
2. max_features: maximum number of features random forest considers splitting a node.
3. mini_sample_leaf: determines the minimum number of leaves required to split an internal node.

Following hyperparameters increases the speed:

1. n_jobs: it tells the engine how many processors it is allowed to use. If the value is one, it can use only one processor but if the value is -1 there is no limit.
2. random_state: controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyperparameters and the same training data.
3. oob_score: It is a random forest cross-validation method. In this one-third of the sample is not used to train the data instead used to evaluate its performance. These samples are called out of bag samples.

4)hyperparameter tuning for Random Forest

In [34]:
```
rf = RandomForestClassifier(random_state=42, n_jobs=-1)
params = {
    'max_depth': [2,3,5,10,20],
    'min_samples_leaf': [5,10,20,50,100,200],
    'n_estimators': [10,25,30,50,100,200]
}
from sklearn.model_selection import GridSearchCV
# Instantiate the grid search model
grid_search = GridSearchCV(estimator=rf,
                           param_grid=params,
                           cv = 4,
                           n_jobs=-1, verbose=1, scoring="accuracy")
grid_search.fit(x_train, y_train)
```

```
Fitting 4 folds for each of 180 candidates, totalling 720 fits
```

Out[34]:
```
GridSearchCV(cv=4, estimator=RandomForestClassifier(n_jobs=-1, random_state=42),
             n_jobs=-1,
             param_grid={'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy', verbose=1)
```

In [35]:
```
grid_search.best_score_
```

Out[35]:
```
0.9920454545454546
```

In [36]:
```
rf_best = grid_search.best_estimator_
rf_best
#From hyperparameter tuning, we can fetch the best estimator as shown. The best set of parameters identified were
#max_depth=20, min_samples_leaf=5, n_jobs=-1,random_state=42
```

Out[36]:
```
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_jobs=-1,
                       random_state=42)
```

Fig. 30. Code in python for hyperparameter tuning

Results and Discussion

Results

As result of this project, we applied data preprocessing techniques on dataset. Then, we tried different models from a sklearn library. We chose the best model which is Random Forest depending on accuracy because our dataset is balanced. It calculates the ratio of correct predication made by the ML model over the total number of instances evaluated. Moreover, we tested the overfitting and underfitting. We saw that the training set's score is similar and close to the testing set. Therefore, the Random Forest model has reached the generalization. In addition, we used hyperparameter tuning for Random Forest to enhance the model's performance and make it faster.

Discussion

All algorithms gave use high accuracy and equal precision and recall with its accuracy. The evaluation matrix we used in this project is accuracy because the dataset is balanced. The highest accuracy is given by Random Forest algorithm. it is one of the best techniques with high performance. We built ML model for multiclass problem with Random Forest algorithm. It is fast, simple, flexible, and robust model with some limitations. It has gained a significant interest in the recent past, due to its quality performance in several areas.

References

[1] "Machine learning: What it is and why it matters," *SAS*. [Online]. Available: https://www.sas.com/en_sa/insights/analytics/machine-learning.html. [Accessed: 27-Apr-2022].

[2] N. Klingler, "What is an AI model? here's what you need to know," *viso.ai*, 31-Oct-2021. [Online]. Available: https://viso.ai/deep-learning/ml-ai-models/. [Accessed: 27-Apr-2022].

[3] A. Ingle, "Crop recommendation dataset," *Kaggle*, 19-Dec-2020. [Online]. Available: https://www.kaggle.com/datasets/atharvaingle/crop-recommendation-dataset. [Accessed: 26-Apr-2022].

[4] "Sklearn.ensemble.randomforestclassifier," *scikit*. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed: 27-Apr-2022].

[5] *Pandas dataframe describe() method*. [Online]. Available: https://www.w3schools.com/python/pandas/ref_df_describe.asp#:~:text=The%20describe()%20method%20returns,The%20average%20(mean)%20value. [Accessed: 27-Apr-2022].

[6] "Data preprocessing in Machine Learning - Javatpoint," *www.javatpoint.com*. [Online]. Available: https://www.javatpoint.com/data-preprocessing-machine-learning. [Accessed: 27-Apr-2022].

[7] P. author B. R. Vashisht, "Machine learning: When to perform a feature scaling?," *atoti*, 20-Sep-2021. [Online]. Available: https://www.atoti.io/articles/when-to-perform-a-feature-scaling/. [Accessed: 27-Apr-2022].

[8] "Random Forest: Introduction to random forest algorithm," *Analytics Vidhya*, 24-Jun-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#:~:text=Random%20forest%20is%20a%20Supervised,average%20in%20case%20of%20regression. [Accessed: 27-Apr-2022].

[9] A. Two and I. Salian, "Supervised vs. unsupervised learning," *NVIDIA Blog*, 20-Aug-2019. [Online]. Available: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/. [Accessed: 27-Apr-2022].

[10] N. Klingler, "What is an AI model? here's what you need to know," *viso.ai*, 31-Oct-2021. [Online]. Available: https://viso.ai/deep-learning/ml-ai-models/. [Accessed: 27-Apr-2022].

[11] Ajitjaokar and Ajitjaokar, "How to choose a machine learning model – some guidelines," *Data Science Central*, 15-Oct-2018. [Online]. Available: https://www.datasciencecentral.com/how-to-choose-a-machine-learning-model-some-guidelines/. [Accessed: 27-Apr-2022].

[12] "Evaluation Metrics", DeepAI, 2022. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/evaluation-metrics#:~:text=Evaluation%20metrics%20are%20used%20to%20measure%20the%20quality,classification%20accuracy%2C%20logarithmic%20loss%2C%20confusion%20matrix%2C%20and%20others. [Accessed: 27- Apr- 2022].

[13] "A Guide to Evaluation Metrics for Classification Models | Deepchecks", Deepchecks, 2022. [Online]. Available: https://deepchecks.com/a-guide-to-evaluation-metrics-for-classification-models/. [Accessed: 27- Apr- 2022].

[14] "Confusion Matrix for Multi-Class Classification - Analytics Vidhya", *Analytics Vidhya*, 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/. [Accessed: 28- Apr- 2022].

[15] "Overfitting vs. Underfitting: What Is the Difference? | 365 Data Science", 365 Data Science, 2022. [Online]. Available: https://365datascience.com/tutorials/machine-learning-tutorials/overfitting-underfitting/. [Accessed: 27- Apr- 2022].

[16] "ShieldSquare Captcha", *Iopscience.iop.org*, 2022. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/2161/1/012033. [Accessed: 04- May- 2022].

[17] "Decision Trees: Gini vs Entropy ⋆ Quantdare", *Quantdare*, 2022. [Online]. Available: https://quantdare.com/decision-trees-gini-vs-entropy/. [Accessed: 04- May- 2022].

[18] “Crop recommendation system using machine learning - ijsrcseit.com.” [Online]. Available: https://ijsrcseit.com/paper/CSEIT2173129.pdf. [Accessed: 04-May-2022].

[19] "A guide to the types of machine learning algorithms", *Sas.com*, 2022. [Online]. Available: https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html. [Accessed: 04- May- 2022].

[20] "XGBoost Algorithm: Long May She Reign!", *Medium*, 2022. [Online]. Available: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d. [Accessed: 04- May- 2022].