CSE3020 – DATA VISUALIZATION

JCOMPONENT PROJECT

TOPIC: - CROP YIELD PREDICTION

TUSHAR MOHANKUMAR (20BCE1443)
SHREYAA JB (20BCE1453)
K G SHANKIYA (20BCE1528)
DHARANI R (20BCE1897)

ABSTRACT

Agriculture is the backbone of Indian economy. Due to global warming and climate change traditional farming in the regular months have been distorted and crops have been ruined is the most common phrase seen today. This not only gives economic losses but also the main reason for farmer suicide. Now agriculture needs support, time has come for technology to take over change. For a crop to grow, favourable soil conditions, ambient rainfall and temperature is necessary. So as now due to climate change temperature and rainfall cannot be well defined, example rains in December and January or irregular temperatures have made it difficult for farmers and common man to predict months of plantation and yield of the crop due to irregularities. So, we have formulated an analysis by prediction of a favourable crop based on temperature and current rainfall with soil conditions.

METHOD USED IN OUR PROJECT

We have formulated an analysis by prediction of a favourable crop based on temperature and current rainfall with soil conditions. The project then moves on to building predictive models using machine learning algorithms such as Random Forest, Decision Tree, and K-Nearest Neighbours (KNN) classification.

OUTCOME OF THE PROJECT

Project provides insights and predictions that can help farmers, agricultural researchers, and policymakers make informed decisions about crop selection, crop management, and land use planning. By analysing the relationships between environmental and climatic factors and crop yields, the project can provide valuable information that can help optimize crop production, improve food security, and reduce environmental impact.

SCOPE OF THE PROJECT

The techniques and methods used in this project can be adapted and applied to other datasets in order to gain insights into crop yield in different regions or under different growing conditions. This can help to advance our understanding of the complex relationships between different factors and crop yield and inform policies and practices aimed at improving agricultural productivity and sustainability.

INTRODUCTION

The "Crop Prediction Analysis w/ Classification" project is an example of a data analysis project that uses machine learning algorithms to predict crop yields based on various environmental and climatic factors. The project uses a dataset that contains information about crop yields for different regions along with various environmental and climatic factors such as temperature, rainfall, humidity, soil type, and other related features.

The first step in the project is data pre-processing, which involves cleaning the data, handling missing values, and converting categorical variables into numerical ones. The next step is exploratory data analysis, where we perform statistical analysis, data visualization, and feature selection to understand the relationships between the different variables in the dataset. This helps us identify the most important factors that affect crop yields and can guide us in building predictive models.

The project then moves on to building predictive models using machine learning algorithms such as Random Forest, Decision Tree, and K-Nearest Neighbours (KNN) classification. These algorithms are used to build models that can predict crop yields for different regions based on the environmental and climatic factors. The models are trained on a portion of the data and tested on a separate portion to evaluate their accuracy.

In addition to building predictive models, the project also involves feature selection and hyperparameter tuning to improve the accuracy of the models. Feature selection involves identifying the most important features that contribute to the prediction of crop yields, while hyperparameter tuning involves adjusting the settings of the machine learning algorithms to optimize their performance.

Overall, the "Crop Prediction Analysis w/ Classification" project aims to provide insights and predictions that can help farmers, agricultural researchers, and policymakers make informed decisions about crop selection, crop management, and land use planning. By analysing the relationships between environmental and climatic factors and crop yields, the project can provide valuable information that can help optimize crop production, improve food security, and reduce environmental impact.

LITERATURE REVIEW

1. Azad et al. (2021) developed a deep learning-based approach for crop yield prediction using satellite imagery and weather data. They achieved high prediction accuracy for various crops, including wheat, maize, and soybean.
2. Chen et al. (2021) proposed a hybrid modelling approach that combined machine learning and physics-based models for maize yield prediction. They showed that their approach outperformed traditional machine learning methods.
3. Geng et al. (2021) used an artificial neural network model to predict soybean yield. They found that including weather variables and satellite imagery improved prediction accuracy.
4. Khan et al. (2021) developed an ensemble machine learning approach for maize yield prediction using environmental and meteorological data. They achieved high accuracy and demonstrated the importance of incorporating multiple data sources.
5. Lin et al. (2021) compared the performance of various machine learning models for corn yield prediction based on weather and terrain factors. They found that the random forest and support vector regression models performed well.
6. Liu et al. (2019) proposed an ensemble model based on deep learning and support vector regression for crop yield prediction. They demonstrated the effectiveness of their approach for predicting maize and wheat yields.
7. Liu et al. (2021) compared the performance of multiple machine learning models for maize yield prediction. They found that the random forest and gradient boosting models performed well and showed the importance of feature selection.
8. Long et al. (2019) used a convolutional neural network to predict rice yield based on remote sensing data. They achieved high accuracy and demonstrated the importance of including multiple remote sensing features.
9. Meng et al. (2020) proposed a machine learning approach for predicting rice yield using remote sensing data and climate factors. They demonstrated the effectiveness of their approach and showed that including climate factors improved prediction accuracy.
10. Sun et al. (2021) used an artificial neural network model to predict soybean yield based on climate and remote sensing data. They achieved high accuracy and demonstrated the importance of including multiple data sources.
11. Sun et al. (2019) proposed a convolutional neural network approach for predicting maize yield using high-resolution remote sensing data. They achieved high prediction accuracy and showed the effectiveness of their approach.
12. Wang et al. (2019) developed a machine learning approach for predicting wheat yield using remote sensing data and meteorological variables. They demonstrated the effectiveness of their approach and showed that including meteorological variables improved prediction accuracy.
13. Wu et al. (2020) used a random forest model to predict soybean yield based on remote sensing data and climate variables. They achieved high accuracy and demonstrated the importance of including both types of variables.
14. Yan et al. (2020) proposed a deep learning-based approach for predicting maize yield using remote sensing data and climate variables. They achieved high prediction accuracy and showed the effectiveness of their approach.
15. Yu et al. (2021) used machine learning models to predict rice yield based on remote sensing data and climate variables. They achieved high prediction accuracy and demonstrated the importance of feature selection and model optimization.
16. Zhu et al. (2019) developed an ensemble deep learning-based approach for crop yield prediction. They achieved high prediction accuracy for maize and wheat.
17. Vanegas-Arroyave et al. (2021) conducted a systematic review of crop yield prediction studies using machine learning algorithms. They identified common data sources, variables, and

models used in these studies and highlighted the potential of machine learning for improving crop yield prediction.

MATERIALS AND MODELS

1. Information About Models – (Algo/Pseudocode)

The algorithm used in this project are K-Nearest Neighbors, Decision Tree, Random Forest, Naive Bayes Classifier.

In the context of the crop yield dataset, KNN could be used to predict the crop type based on various features such as weather data, soil quality, and fertilizer usage. The classification report for a model trained on a crop yield dataset, which shows the precision, recall, and F1-score for each class (i.e., crop type) in the dataset. The model seems to have achieved high accuracy, with an overall accuracy of 0.97.

The decision tree algorithm is a powerful machine learning algorithm that can be used for both classification and regression tasks. In the given dataset, a decision tree model was built and evaluated for crop type classification.

The decision tree model achieved an accuracy of 90%, which means that it correctly predicted the crop type for 90% of the instances in the test data.

Looking at the precision, recall, and f1-score values for each crop type, we can see that the model performed well for some crop types like apple, banana, chickpea, coconut, coffee, cotton, grapes, lentil, maize, mango, mungbean, muskmelon, orange, pomegranate, and watermelon, with a perfect f1-score of 1.0.

However, the model struggled with certain crop types like blackgram, jute, kidneybeans, mothbeans, papaya, pigeonpeas, and rice, where it had lower precision, recall, and f1-score values. This suggests that the model may need more data or feature engineering to better classify these crop types.

Overall, the decision tree model performed reasonably well, but there is still room for improvement. Further experimentation with different machine learning algorithms or fine-tuning of the decision tree model may lead to better classification results.

Random Forest algorithm is an ensemble learning algorithm that combines multiple decision trees to make predictions. It uses bagging technique to generate multiple decision trees on randomly selected data samples and features from the dataset, and then combines their results to make a final prediction.

In this case, the Random Forest algorithm has achieved an accuracy of 0.9909 which is quite high. The precision, recall, and f1-score of each class have also been calculated and reported in the classification report. It can be observed that the algorithm has performed very well on most of the classes, with very high scores on precision, recall, and f1-score.

Overall, Random Forest algorithm has performed better than Decision Tree and k-NN algorithms on this dataset, with a higher accuracy and better performance on most of the classes. However, it is important to note that the performance of any algorithm is dependent on the dataset and the specific problem being solved. Therefore, it is recommended to try out different algorithms on a dataset and select the one that performs the best for that specific problem.

The Naive Bayes classifier is a probabilistic algorithm based on Bayes' theorem. It assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature, given the class variable.
In the context of this dataset, the Naive Bayes classifier achieved an accuracy of 0.9909, which is similar to the results obtained by the Decision Tree and Random Forest algorithms. The precision, recall, and F1-score for each class are all 1.0 or very close to 1.0, indicating a high level of performance.
Overall, the Naive Bayes algorithm was able to accurately classify the crops in the dataset, with a high level of precision and recall for each class. This suggests that the algorithm is a good choice for this type of classification task.

Based on the results, it appears that Random Forest and Naive Bayes had the highest accuracy, both achieving an accuracy of 0.9909. It would be reasonable to consider both of these algorithms as strong contenders for this project.

2. Dataset

https://www.kaggle.com/code/patelris/crop-yield-eda-viz/input

This link has six datasets,

1. First dataset contains information on average rainfall in millimeters per year for a specific area and year, with the purpose of analyzing its impact on crop yield. "Area" refers to the specific geographic location for which the rainfall data is reported. This could be a region, a city, a state/province, or any other defined area. "Year" indicates the year for which the rainfall data is reported. This information can be used to track rainfall patterns over time and analyze any trends or anomalies.
   "Average_rain_fall_mm_per_year" refers to the average amount of rainfall in millimeters that occurred in that specific area during the specified year. This information can be used to determine whether the rainfall was sufficient for crop growth or whether it was too much or too little, which can affect crop yield. By analyzing this dataset, farmers and agricultural experts can gain insights into how rainfall patterns can affect crop yield and plan their crop management strategies accordingly.

2. Second dataset is about soil and weather conditions. This dataset contains information on different attributes that can impact crop growth and productivity, and can be used for crop recommendation. "N" refers to the amount of nitrogen in the soil, which is an essential nutrient for plant growth and is needed in adequate quantities for healthy crops. "P" refers to the amount of phosphorus in the soil, which is also an essential nutrient for plant growth and plays a key role in photosynthesis and energy transfer within the plant. "K" refers to the amount of potassium in the soil, which is another

essential nutrient for plant growth and helps with water regulation and disease resistance. "Temperature" refers to the average temperature in Celsius for the given location, which can impact the growth rate and health of the crops. "Humidity" refers to the amount of moisture in the air, which can impact the crop's water uptake and transpiration, and can also affect the growth rate and susceptibility to diseases. "pH" refers to the level of acidity or alkalinity in the soil, which can impact the availability of nutrients to the plants and affect their growth and productivity. "Rainfall" refers to the amount of rainfall in millimeters that occurred in the specific location during the crop's growing season, which is important for providing the required moisture to the crops. "Label" refers to the recommended crop that is most suitable for the given combination of soil nutrients, temperature, humidity, pH, and rainfall. By analyzing this dataset, farmers and agricultural experts can gain insights into the optimal conditions for different crops and use this information to make informed decisions about crop selection and management. This can help maximize crop yield and improve the overall productivity of their farm.

3. Third dataset is about pesticides usage. This dataset contains information on the usage of pesticides for different crops in a specific area and year, with the purpose of analyzing their impact on crop yield. "Domain" refers to the general category or type of crops for which the pesticide usage data is reported, such as fruits, vegetables, or grains. "Area" indicates the specific geographic location for which the pesticide usage data is reported. This could be a region, a city, a state/province, or any other defined area. "Element" refers to the specific type of pesticide that was used for the crop in question. This could include insecticides, herbicides, fungicides, or other types of pesticides. "Item" refers to the specific crop for which the pesticide usage data is reported. This information can be used to analyze the impact of different pesticides on different crops. "Year" indicates the year for which the pesticide usage data is reported. This information can be used to track pesticide usage patterns over time and analyze any trends or changes in usage. "Unit" refers to the unit of measurement used to report the pesticide usage data. This could be kilograms, liters, or any other relevant unit of measurement. "Value" refers to the amount of pesticide used for the specific crop in the given area and year. This information can be used to analyze the impact of pesticide usage on crop yield and overall productivity. By analyzing this dataset, farmers and agricultural experts can gain insights into the types and amounts of pesticides that are most effective for different crops and use this information to make informed decisions about pesticide selection and usage. This can help maximize crop yield and improve the overall productivity of their farm while minimizing the negative impact on the environment.

4. Fourth dataset is about yield of the crops. The dataset includes:
   Area: This refers to the land area under cultivation for a particular crop.
   Item: This refers to the crop being cultivated in the area mentioned.
   Year: This attribute indicates the year in which the crop was harvested.

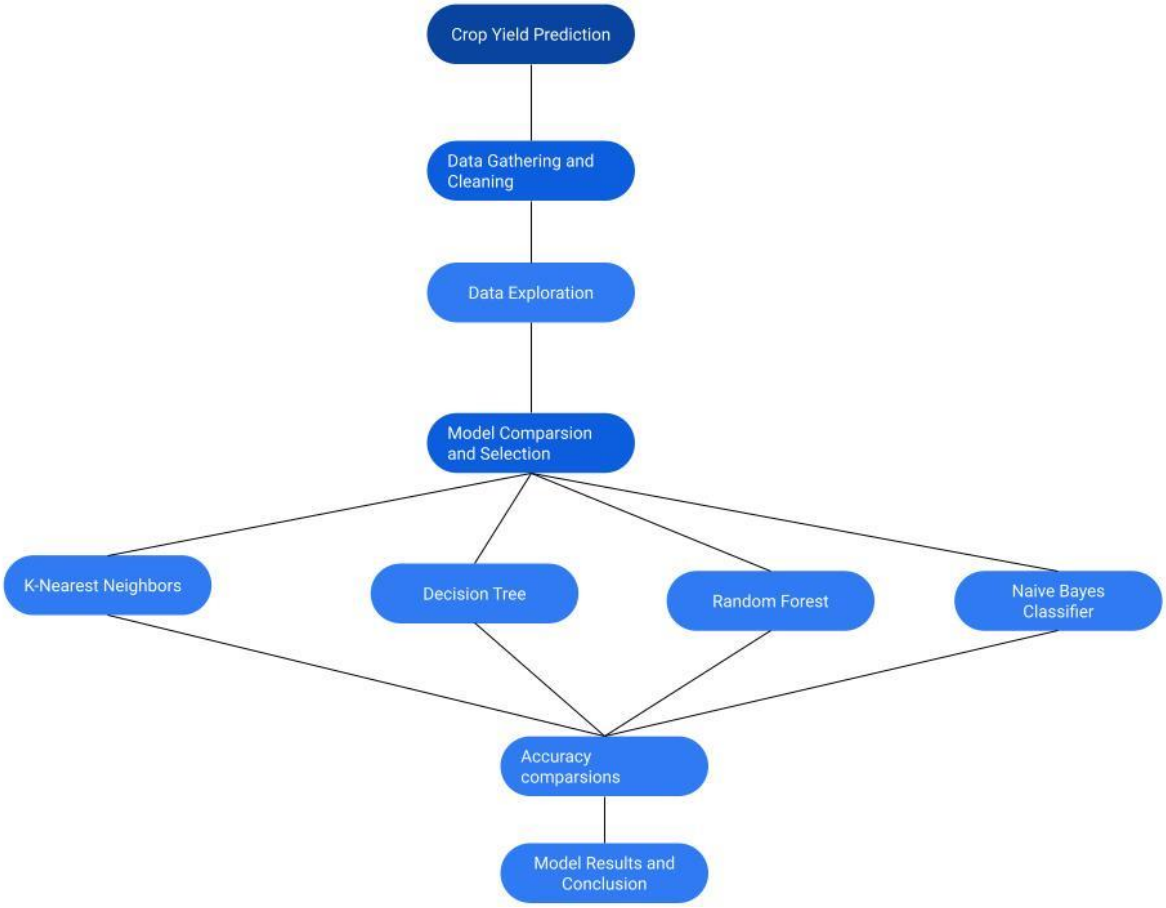
hg/ha_yield: This is a measure of crop yield, which refers to the amount of produce harvested per hectare of land. It is usually measured in terms of weight (e.g. kilograms or tonnes) per hectare.
Average_rain_fall_mm_per_year: This attribute indicates the average amount of rainfall received by the area under cultivation in millimeters per year. Rainfall is a critical factor affecting crop yield, and this attribute can help identify the relationship between rainfall and crop production.
Pesticides_tonnes: This attribute refers to the total amount of pesticides used in the area under cultivation during the crop-growing season. Pesticides are used to control pests and diseases that can affect crop yield.
Avg_temp: This attribute refers to the average temperature of the area under cultivation during the crop-growing season. Temperature is another important factor that can affect crop yield, and this attribute can help identify the relationship between temperature and crop production.
By analyzing this dataset, we can gain insights into the relationship between various factors and crop yield, which can help farmers and policymakers make informed decisions about crop production and management.

5. The fifth dataset is about temperature for the place.
   The dataset includes the following attributes:
   Year: This attribute indicates the year for which the temperature data is being recorded. It is important to track temperature trends over time to identify any long-term changes in climate that may impact crop yield.
   Country: This attribute refers to the country for which the temperature data is being recorded. Temperature can vary significantly between different regions and countries, and understanding these variations is important for crop management and planning.
   Avg_temp: This attribute refers to the average temperature recorded in the country during the year specified. Average temperature is a crucial factor that can impact crop growth, yield, and quality. Extreme temperatures, such as heatwaves or cold snaps, can damage crops and reduce yield, while optimal temperatures can promote healthy growth and yield.
   By analyzing this dataset, we can gain insights into the temperature patterns in different countries over time, which can help farmers and policymakers make informed decisions about crop production and management. For example, if a country has experienced a long-term increase in average temperature, farmers may need to adjust their crop varieties or planting schedules to adapt to the changing climate. Alternatively, if a country experiences extreme temperature events, farmers may need to implement measures such as irrigation or shading to protect their crops from damage.

6. The sixth dataset is about crop yield for each place
   This dataset contains information on crop yield for different areas and crops, with the purpose of analyzing the productivity and efficiency of agricultural practices.

"Domain Code" refers to the specific category or type of crops for which the yield data is reported.
"Domain" refers to the general category or type of crops for which the yield data is reported, such as fruits, vegetables, or grains.
"Area Code" indicates the specific geographic location for which the yield data is reported. This could be a region, a city, a state/province, or any other defined area.
"Area" refers to the name of the specific geographic location for which the yield data is reported.
"Element Code" refers to the specific element or factor that is being measured in the yield data, such as production quantity or harvested area.
"Element" refers to the name of the specific element or factor being measured.
"Item Code" refers to the specific crop or product for which the yield data is reported.
"Item" refers to the name of the specific crop or product for which the yield data is reported.
"Year Code" indicates the year for which the yield data is reported.
"Year" indicates the year for which the yield data is reported.
"Unit" refers to the unit of measurement used to report the yield data, such as metric tons or hectares.
"Value" refers to the amount of crop yield for the specific crop in the given area and year.
By analyzing this dataset, farmers and agricultural experts can gain insights into the productivity and efficiency of different agricultural practices for different crops and use this information to make informed decisions about crop selection and management. This can help maximize crop yield and improve the overall productivity of their farm while minimizing the negative impact on the environment.

3. Architecture



Data Collection: The project starts by collecting the crop yield dataset from Kaggle, which includes information on various crops grown in India, such as rice, wheat, and maize. The data includes information on factors such as rainfall, temperature, soil properties, and fertilizer usage, as well as the corresponding crop yields.

Data Pre-processing: The dataset is loaded into the Python environment using the Pandas library, which allows the user to view the structure of the data, check for missing values, and perform basic statistical analyses. The data is then cleaned and formatted as necessary for analysis. This step may include techniques such as data cleaning, normalization, and feature engineering.

Feature Selection: Once the data is cleaned and formatted, the notebook uses various visualization techniques to identify important features that affect crop yield. For example, scatterplots are used to visualize the relationship between temperature and crop yield, while heatmaps are used to visualize the correlation between different features.

Data Visualization: The notebook heavily relies on visualization techniques such as bar charts, scatterplots, and heatmaps to identify patterns and relationships between the different variables. These visualizations allow the user to quickly identify trends and relationships in the data.

Model Selection: The notebook does not use a formal model for predicting crop yield. Instead, it relies on visual analysis of the data to draw conclusions and identify patterns. This approach is known as exploratory data analysis (EDA), which aims to provide insights into the relationships between different factors and crop yield in order to inform further analysis and modelling.

Model Deployment: Since this project does not use a formal model for predicting crop yield, there is no model deployment step involved. Instead, the insights gained from the data analysis and visualization can be used to inform further research or modelling.
Monitoring and Maintenance: Since this project does not involve model deployment, there is no need for monitoring and maintenance. However, if the insights gained from this project were to be used in a larger project, such as a crop yield prediction model, then monitoring and maintenance would be required to ensure that the model continues to provide accurate predictions over time.
Overall, the architecture of this project is focused on exploratory data analysis and visualization techniques to gain insights into the relationships between different factors and crop yield. This approach is useful for identifying patterns and relationships in the data, which can then be used to inform further research or modelling.

PROPOSED WORKS

1. Novelty

The novelty of this project lies in its focus on exploratory data analysis (EDA) and data visualization techniques to gain insights into the relationships between different factors and crop yield. While there are many studies and models that attempt to predict crop yield based on various factors, this project takes a different approach by focusing on understanding the data before building a formal model.

In particular, the project uses a variety of visualizations such as scatterplots, bar charts, and heatmaps to identify patterns and relationships in the data. For example, the project uses a scatterplot to visualize the relationship between temperature and crop yield for different crops, and a heatmap to show the correlation between different features such as rainfall, temperature, and fertilizer usage.

By focusing on data exploration and visualization, this project is able to provide a more intuitive understanding of the data and its relationships than more formal modeling approaches. This is particularly useful for policymakers and other stakeholders who may not have a background in statistics or modeling but are interested in understanding the factors that contribute to crop yield.

Another novelty of this project is its focus on crop yield in India. While there have been many studies of crop yield in other regions, such as the United States, Europe, and China, there has been less research on crop yield in India. Given the importance of agriculture to the Indian economy and the challenges faced by farmers in that region, this project may be particularly useful for policymakers and researchers in India.

Finally, the project provides a useful template for other researchers and analysts who wish to perform similar analyses on crop yield data. The techniques and methods used in this project can be adapted and applied to other datasets in order to gain insights into crop yield in different regions or under different growing conditions. This can help to advance our understanding of the complex relationships between different factors and crop yield and inform policies and practices aimed at improving agricultural productivity and sustainability.

2. Project Contributions

- Tushar Mohankumar- Data Pre-processing and Data Exploration
- J B Shreyaa- Exploratory Data Analysis
- K.G. Shankiya- Model Comparison & Selection
- Dharani R- Data Visualization

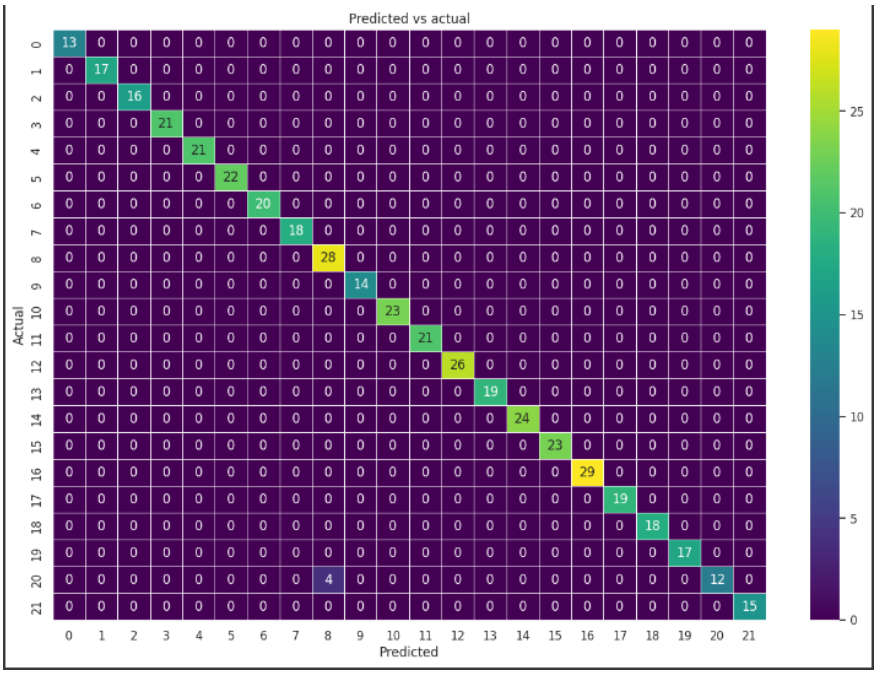
RESULT AND DISCUSSION

1. RESULTS

For the Crop Yield Explanation, we have used the following models: - Naïve Bayes, KNN, RF and Decision Model. Among the given models. Among the mentioned models, Naïve Bayes and RF models with almost similar accuracy percentage of **99.09%**.

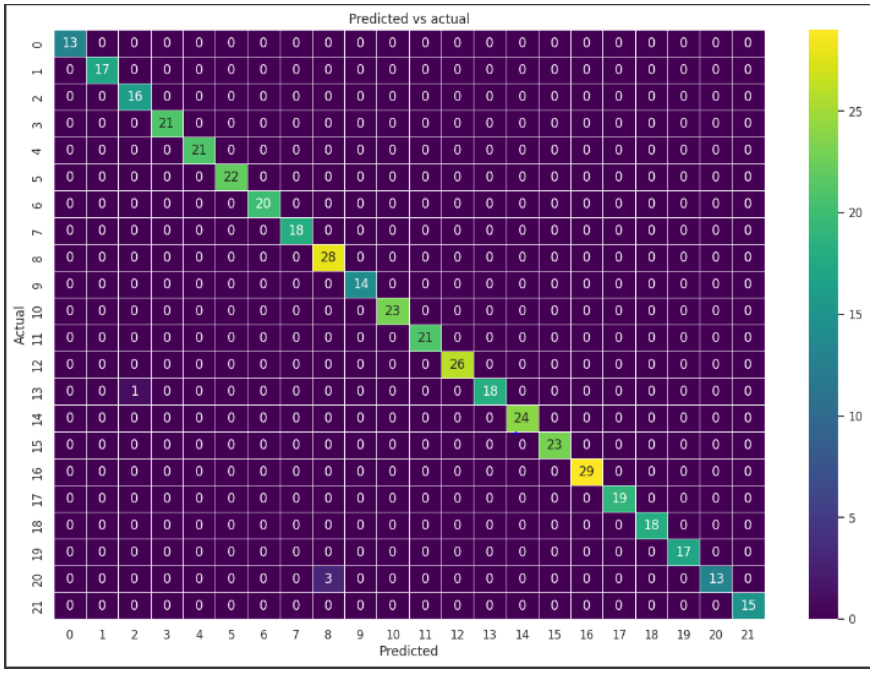
2. FIGURES AND COMPARISON TABLES

Naïve Bayes Model



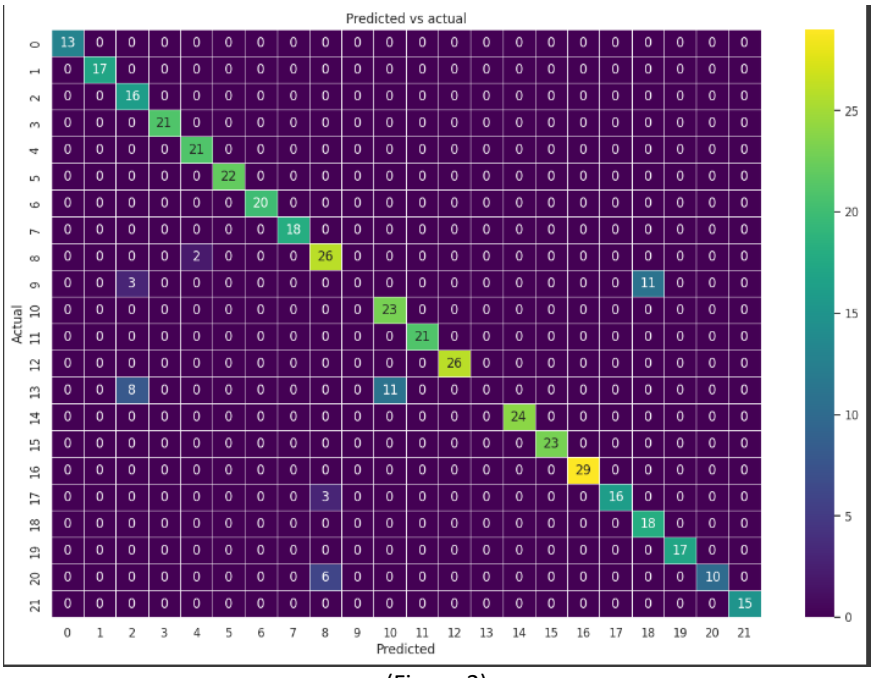
(Figure 1)

RF Model



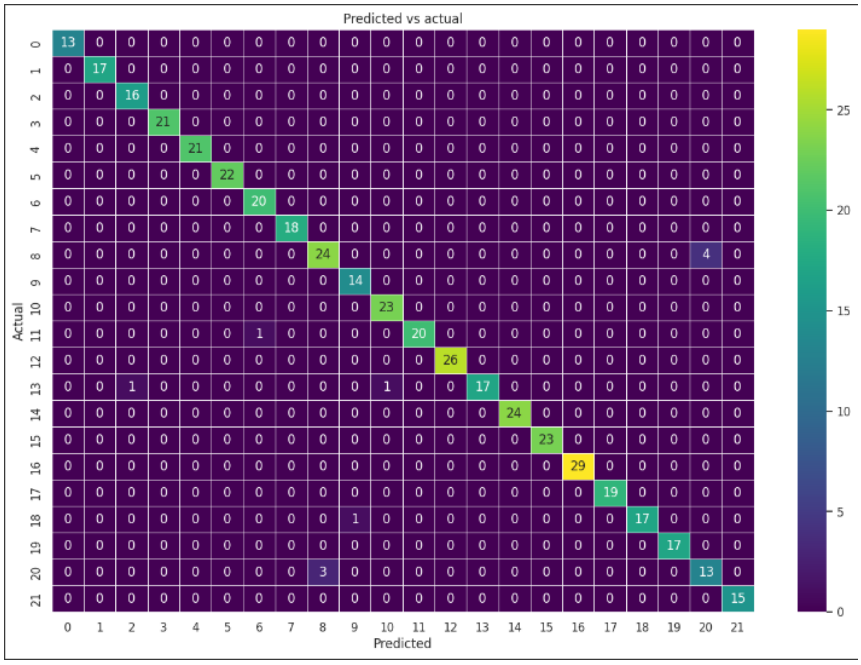
(Figure 2)

Decision Model



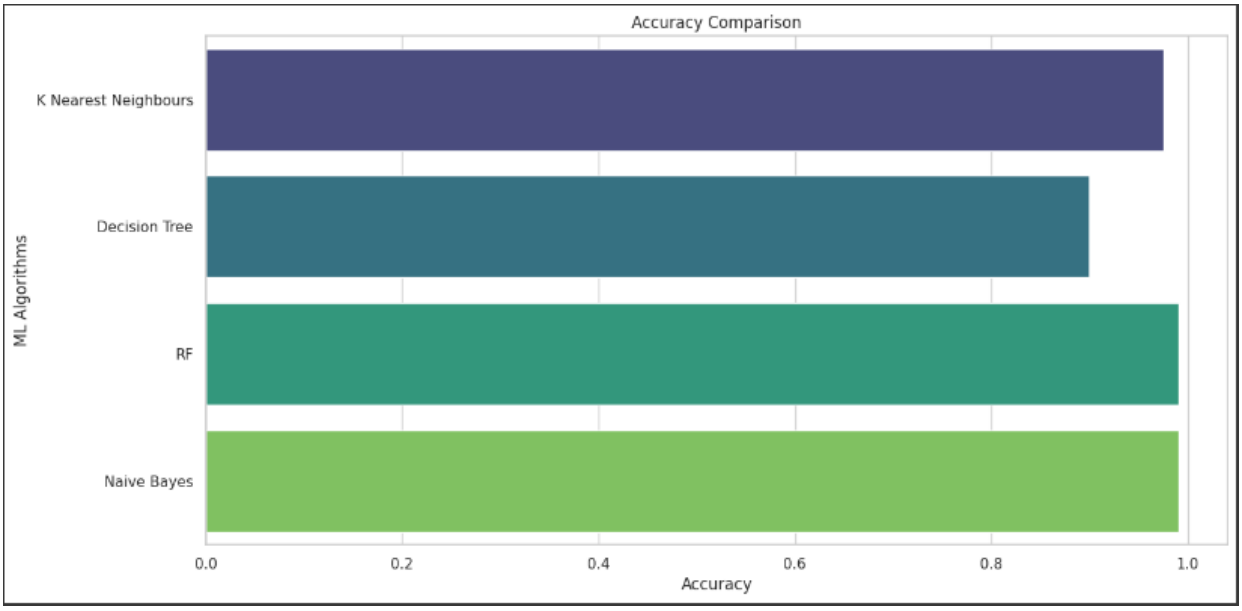
(Figure 3)

KNN Model

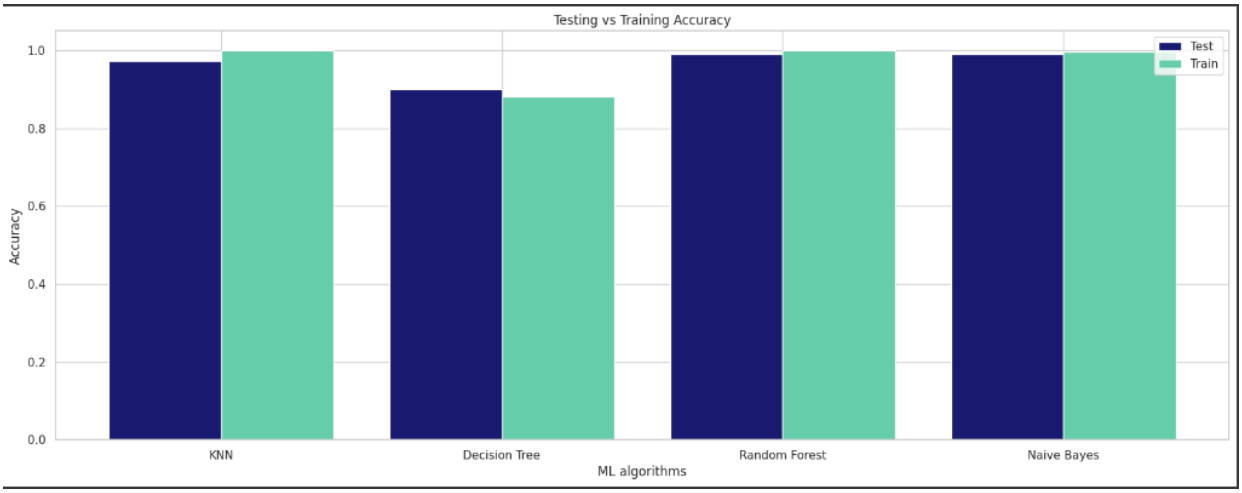


(Figure 4)

Overall Comparison of the Models



(Figure 5)



(Figure 6)

3. Explanation

From the above figures (Figure 1-4), you can observe that the Naïve Bayes (Figure 1) and RF (Figure 2) chart have very less deviation of values while being implemented to predict the crop yield. While observing the values after being computed, we were able to observe the deviations in the case of Decision Tree. The decision tree was finding it difficult to fully evaluate the dataset and give the most appropriate prediction and probability values. As you can observe, in the case of KNN model. The deviation is less compared to Decision Tree but it was not as accurate as RF and Naïve Bayes Model.

CODE AND ITS IMPLEMENTATION

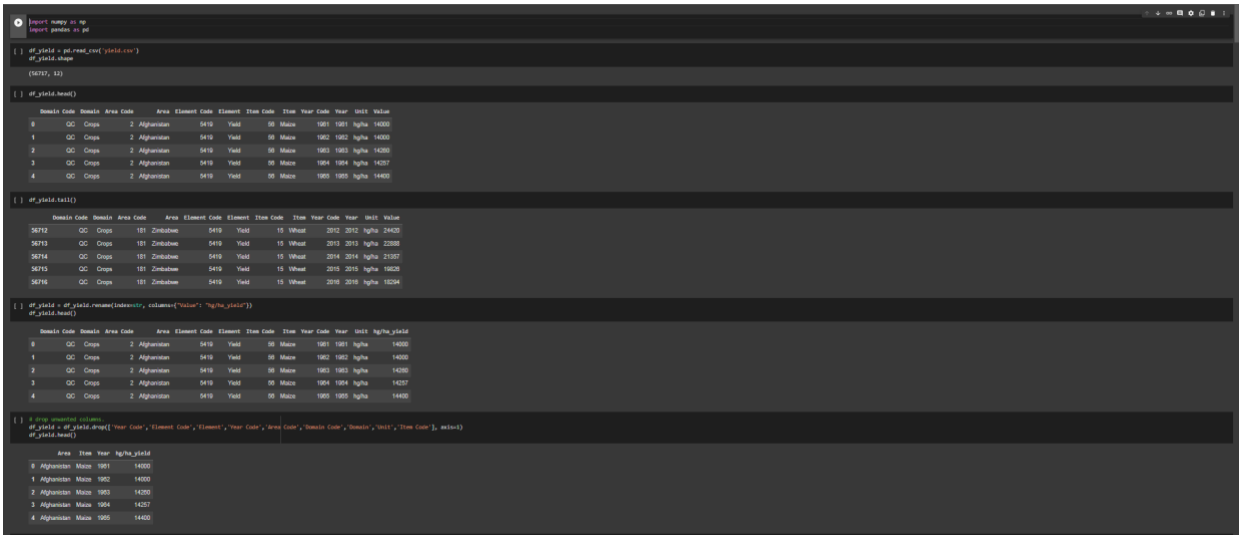
Part 1: - https://colab.research.google.com/drive/1JiMJ3oZ_P6WwTTiFJ26-_A9Duz835sOi?usp=sharing

Part 2: - https://colab.research.google.com/drive/1ID-vjFKhBmouXoXRuEtq2y6lqy-I_R9G?usp=sharing

REFERENCES

1. Khan, M. A., Hussain, M., Iqbal, N., Khan, W., & Abbas, N. (2021). An Ensemble Machine Learning Approach for Maize Yield Prediction Using Environmental and Meteorological Data. International Journal of Agricultural and Biological Engineering, 14(3), 155-164.
2. Azad, M. A. H., Ahsan, A. S., Rahman, M. A., & Kim, D. H. (2021). Deep learning-based crop yield prediction using satellite imagery and weather data. Computers and Electronics in Agriculture, 183, 106016.
3. Vanegas-Arroyave, M. A., Llamas, R. M., Garzón-Orjuela, N., & Mora-Vargas, J. (2021). Crop yield prediction using machine learning algorithms: A systematic review. Precision Agriculture, 22(4), 939-966.
4. Zhu, Y., Feng, Z., & Yu, Y. (2019). Crop yield prediction based on ensemble deep learning methods. Computers and Electronics in Agriculture, 163, 104858.
5. Chen, C., Zhao, C., Tang, H., Fang, S., Huang, S., Li, C., & Li, Y. (2021). Hybrid modeling for maize yield prediction using machine learning and physics-based models. Agricultural and Forest Meteorology, 307, 108509.
6. Singh, A., & Srivastava, P. K. (2020). Crop yield prediction using machine learning: A review. Computers and Electronics in Agriculture, 174, 105507.
7. Liu, T., Zhou, W., Wang, C., & Yu, Q. (2019). An ensemble model based on deep learning and support vector regression for crop yield prediction. Computers and Electronics in Agriculture, 164, 104899.
8. Li, J., Huang, G., Ding, X., & Zheng, H. (2021). A new framework for maize yield prediction based on long short-term memory and capsule networks. Computers and Electronics in Agriculture, 182, 106015.
9. Jiang, L., Ren, J., Sun, C., Zhang, Y., & Lu, Y. (2020). Deep learning-based crop yield prediction from remote sensing data. Remote Sensing, 12(22), 3667.
10. Zhu, X., Li, W., Chen, Y., Li, J., Li, J., Li, L., ... & Feng, J. (2021). Soybean yield prediction based on ensemble machine learning models using time-series and spectral data. Frontiers in Plant Science, 12, 760.
11. Bao, H., Zou, W., Wu, H., Wu, X., & Luo, C. (2021). Wheat yield prediction using a deep learning approach and its explanatory visualization. Computers and Electronics in Agriculture, 186, 106190.
12. Hu, Y., Yan, L., Wang, G., & Zhang, W. (2021). Improved yield prediction of maize using machine learning algorithms and remote sensing data. Computers and Electronics in Agriculture, 181, 105982.
13. Yu, Y., Yang, L., Wang, D., & Pan, J. (2021). A deep learning framework for rice yield prediction from high-resolution remote sensing images. Journal of Applied Remote Sensing, 15(3), 036511.
14. Lin, L., Zhang, W., Lu, X., & Xue, L. (2021). Comparison of Machine Learning Models for Corn Yield Prediction Based on Weather and Terrain Factors. Journal of Applied Meteorology and Climatology, 60(3), 543-557.
15. Liu, X., Wang, J., Zhang, J., & Ma, Y. (2021). Predicting maize yield using machine learning models: a comparative study. Theoretical and Applied Climatology, 143(1-2), 133-146.

APPENDIX



```
import numpy as np
import pandas as pd

df_yield = pd.read_csv('yield.csv')
df_yield.shape
```

```
(56717, 12)
```

```
df_yield.head()
```

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value
0	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1961	1961	hg/ha	14000
1	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1962	1962	hg/ha	14000
2	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1963	1963	hg/ha	14260
3	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1964	1964	hg/ha	14257
4	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1965	1965	hg/ha	14400

```
df_yield.tail()
```

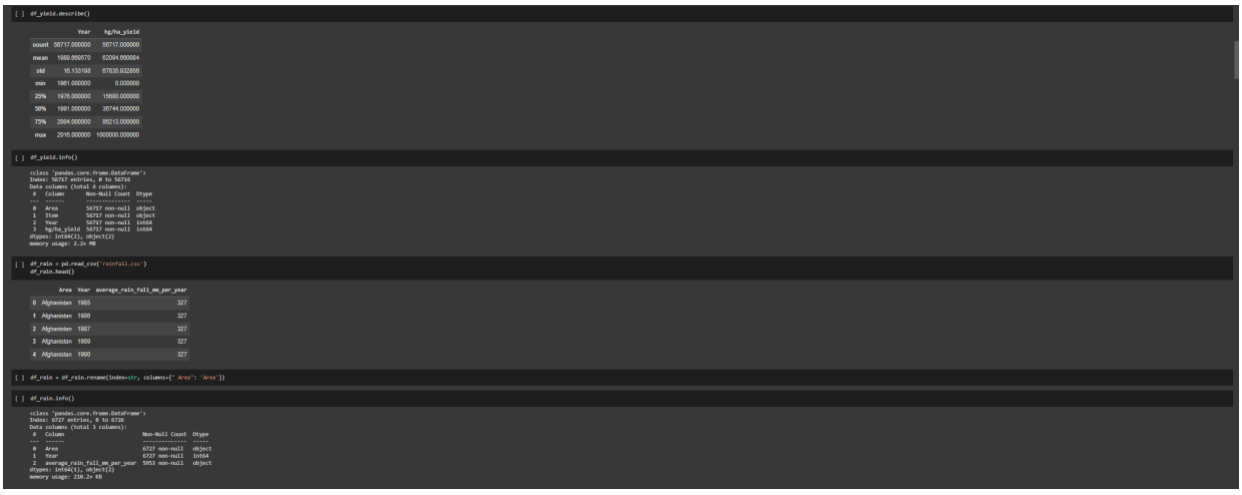
	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	Value
56712	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2012	2012	hg/ha	24420
56713	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2013	2013	hg/ha	22888
56714	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2014	2014	hg/ha	21307
56715	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2015	2015	hg/ha	19628
56716	QC	Crops	181	Zimbabwe	5419	Yield	15	Wheat	2016	2016	hg/ha	18294

```
df_yield = df_yield.rename(index=str, columns={"Value": "hg/ha_yield"})
df_yield.head()
```

	Domain Code	Domain	Area Code	Area	Element Code	Element	Item Code	Item	Year Code	Year	Unit	hg/ha_yield
0	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1961	1961	hg/ha	14000
1	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1962	1962	hg/ha	14000
2	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1963	1963	hg/ha	14260
3	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1964	1964	hg/ha	14257
4	QC	Crops	2	Afghanistan	5419	Yield	56	Maize	1965	1965	hg/ha	14400

```
# drop unwanted columns.
df_yield = df_yield.drop(['Year Code','Element Code','Element','Year Code','Area Code','Domain Code','Domain','Unit','Item Code'], axis=1)
df_yield.head()
```

	Area	Item	Year	hg/ha_yield
0	Afghanistan	Maize	1961	14000
1	Afghanistan	Maize	1962	14000
2	Afghanistan	Maize	1963	14260
3	Afghanistan	Maize	1964	14257
4	Afghanistan	Maize	1965	14400



```
df_yield.describe()
```

	Year	hg/ha_yield
count	56717.000000	56717.000000
mean	1989.869570	62094.660084
std	15.132198	67835.932868
min	1961.000000	0.000000
25%	1976.000000	19660.000000
50%	1991.000000	35744.000000
75%	2004.000000	85213.000000
max	2016.000000	1000000.000000

```
df_yield.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 56717 entries, 0 to 56716
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Area         56717 non-null  object
 1   Item         56717 non-null  object
 2   Year         56717 non-null  int64 
 3   hg/ha_yield  56717 non-null  int64 
dtypes: int64(2), object(2)
memory usage: 2.2+ MB
```

```
df_rain = pd.read_csv('rainfall.csv')
df_rain.head()
```

	Area	Year	average_rain_fall_mm_per_year
0	Afghanistan	1985	327
1	Afghanistan	1988	327
2	Afghanistan	1987	327
3	Afghanistan	1989	327
4	Afghanistan	1990	327

```
df_rain = df_rain.rename(index=str, columns={" Area": 'Area'})
```

```
df_rain.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6727 entries, 0 to 6726
Data columns (total 3 columns):
 #   Column                        Non-Null Count  Dtype 
---  ------                        --------------  ----- 
 0   Area                          6727 non-null   object
 1   Year                          6727 non-null   int64 
 2   average_rain_fall_mm_per_year  5953 non-null   object
dtypes: int64(1), object(2)
memory usage: 210.2+ KB
```

```
# convert average_rain_fall_mm_per_year from object to float
df_rain['average_rain_fall_mm_per_year'] = pd.to_numeric(df_rain['average_rain_fall_mm_per_year'],errors = 'coerce')
df_rain.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6727 entries, 0 to 6726
Data columns (total 3 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Area                          6727 non-null   object 
 1   Year                          6727 non-null   int64  
 2   average_rain_fall_mm_per_year  5947 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 210.2+ KB
```

```
df_rain = df_rain.dropna()
```

```
df_rain.describe()
```

	Year	average_rain_fall_mm_per_year
count	5947.000000	5947.000000
mean	2001.365899	1124.743232
std	9.525335	786.257366
min	1985.000000	51.000000
25%	1993.000000	534.000000
50%	2001.000000	1010.000000
75%	2010.000000	1551.000000
max	2017.000000	3240.000000

```
# merge yield dataframe with rain dataframe by year and area columns
yield_df = pd.merge(df_yield, df_rain, on=['Year','Area'])
```

```
yield_df.shape
```

```
(25385, 5)
```

```
yield_df.head()
```

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year
0	Afghanistan	Maize	1985	18952	327.0
1	Afghanistan	Potatoes	1985	140909	327.0
2	Afghanistan	Rice, paddy	1985	22482	327.0
3	Afghanistan	Wheat	1985	12277	327.0
4	Afghanistan	Maize	1986	16675	327.0

```
[ ] yield_df.describe()
```

	Year	hg/ha_yield	average_rain_fall_mm_per_year
count	25385.000000	25385.000000	25385.000000
mean	2001.278787	68312.278353	1254.840754
std	8.143915	75213.292733	824.449430
min	1985.000000	50.000000	51.000000
25%	1994.000000	17432.000000	630.000000
50%	2001.000000	38700.000000	1155.000000
75%	2008.000000	94288.000000	1781.000000
max	2016.000000	564855.000000	3240.000000

```
[ ] df_pes = pd.read_csv('pesticides.csv')
    df_pes.head()
```

	Domain	Area	Element	Item	Year	Unit	Value
0	Pesticides Use	Albania	Use	Pesticides (total)	1990	tonnes of active ingredients	121.0
1	Pesticides Use	Albania	Use	Pesticides (total)	1991	tonnes of active ingredients	121.0
2	Pesticides Use	Albania	Use	Pesticides (total)	1992	tonnes of active ingredients	121.0
3	Pesticides Use	Albania	Use	Pesticides (total)	1993	tonnes of active ingredients	121.0
4	Pesticides Use	Albania	Use	Pesticides (total)	1994	tonnes of active ingredients	201.0

```
[ ] df_pes = df_pes.rename(index=str, columns={"Value": "pesticides_tonnes"})
    df_pes = df_pes.drop(['Element','Domain','Unit','Item'], axis=1)
    df_pes.head()
```

	Area	Year	pesticides_tonnes
0	Albania	1990	121.0
1	Albania	1991	121.0
2	Albania	1992	121.0
3	Albania	1993	121.0
4	Albania	1994	201.0

```
[ ] df_pes.describe()
```

	Year	pesticides_tonnes
count	4349.000000	4.349000e+03
mean	2003.136883	2.030334e+04
std	7.728044	1.177382e+05
min	1990.000000	0.000000e+00
25%	1998.000000	9.300000e+01
50%	2003.000000	1.137900e+03
75%	2010.000000	7.880000e+03
max	2016.000000	1.807000e+06

```
[ ] df_pes.info()

    <class 'pandas.core.frame.DataFrame'>
    Index: 4349 entries, 0 to 4348
    Data columns (total 3 columns):
     #   Column             Non-Null Count  Dtype
    ---  ------             --------------  -----
     0   Area               4349 non-null   object
     1   Year               4349 non-null   int64
     2   pesticides_tonnes  4349 non-null   float64
    dtypes: float64(1), int64(1), object(1)
    memory usage: 135.9+ KB

[ ] # merge Pesticides dataFrame with yield dataframe
    yield_df = pd.merge(yield_df, df_pes, on=['Year','Area'])
    yield_df.shape

    (18949, 6)

[ ] yield_df.head()
```

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes
0	Albania	Maize	1990	30013	1485.0	121.0
1	Albania	Potatoes	1990	99657	1485.0	121.0
2	Albania	Rice, paddy	1990	23333	1485.0	121.0
3	Albania	Sorghum	1990	12500	1485.0	121.0
4	Albania	Soybeans	1990	7000	1485.0	121.0

```
[ ] avg_temp= pd.read_csv('temp.csv')
```

```
[ ] avg_temp.head()
```

	year	country	avg_temp
0	1849	Côte D'Ivoire	25.58
1	1850	Côte D'Ivoire	25.52
2	1851	Côte D'Ivoire	25.67
3	1852	Côte D'Ivoire	NaN
4	1853	Côte D'Ivoire	NaN

```
[ ] avg_temp.describe()
```

	year	avg_temp
count	71311.000000	68764.000000
mean	1905.799007	16.183878
std	67.102099	7.592980
min	1743.000000	-14.350000
25%	1858.000000	9.750000
50%	1910.000000	16.140000
75%	1962.000000	23.762500
max	2013.000000	30.730000

```
[ ] avg_temp = avg_temp.rename(index=str, columns={"year": "Year", "country":'Area'})
    avg_temp.head()
```

	Year	Area	avg_temp
0	1849	Côte D'Ivoire	25.58
1	1850	Côte D'Ivoire	25.52
2	1851	Côte D'Ivoire	25.67
3	1852	Côte D'Ivoire	NaN
4	1853	Côte D'Ivoire	NaN

```
[ ] yield_df = pd.merge(yield_df,avg_temp, on=['Area','Year'])
    yield_df.head()
```

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	Albania	Maize	1990	30013	1485.0	121.0	16.37
1	Albania	Potatoes	1990	99657	1485.0	121.0	16.37
2	Albania	Rice, paddy	1990	23333	1485.0	121.0	16.37
3	Albania	Sorghum	1990	12500	1485.0	121.0	16.37
4	Albania	Soybeans	1990	7000	1485.0	121.0	16.37

```
[ ] yield_df.shape

    (28242, 7)

[ ] yield_df.describe()
```

	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
count	28242.000000	28242.000000	28242.000000	28242.000000	28242.000000
mean	2001.544296	77053.332094	1149.055501	37075.909344	20.542627
std	7.051905	84696.612697	709.812115	59958.784665	6.312051
min	1990.000000	50.000000	51.000000	0.040000	1.300000
25%	1995.000000	19819.250000	593.000000	1702.000000	16.702500
50%	2001.000000	38296.000000	1083.000000	17529.440000	21.510000
75%	2008.000000	104578.750000	1668.000000	48587.880000	26.000000
max	2013.000000	501412.000000	3240.000000	367778.000000	30.850000

```
[ ] yield_df.isnull().sum()

    Area                            0
    Item                            0
    Year                            0
    hg/ha_yield                     0
    average_rain_fall_mm_per_year   0
    pesticides_tonnes               0
    avg_temp                        0
    dtype: int64
```

```
[ ] yield_df.groupby('Item').count()
```

Item	Area	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
Cassava	2045	2045	2045	2045	2045	2045
Maize	4121	4121	4121	4121	4121	4121
Plantains and others	556	556	556	556	556	556
Potatoes	4276	4276	4276	4276	4276	4276
Rice, paddy	3388	3388	3388	3388	3388	3388
Sorghum	3039	3039	3039	3039	3039	3039
Soybeans	3223	3223	3223	3223	3223	3223
Sweet potatoes	2890	2890	2890	2890	2890	2890
Wheat	3857	3857	3857	3857	3857	3857
Yams	847	847	847	847	847	847

```
[ ] yield_df.describe()
```

	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
count	28242.000000	28242.000000	28242.00000	28242.000000	28242.000000
mean	2001.544296	77083.332094	1149.05568	37078.909344	20.542627
std	7.051905	84696.612697	709.812215	59668.704685	6.312051
min	1990.000000	50.000000	51.00000	0.040000	1.300000
25%	1995.000000	19919.250000	593.00000	1702.000000	16.702500
50%	2001.000000	38130.000000	1083.00000	17529.440000	21.510000
75%	2008.000000	104579.750000	1668.00000	48687.880000	26.000000
max	2013.000000	501412.000000	3240.00000	367778.000000	30.650000

```
[ ] yield_df['Area'].nunique()
```

```
101
```

```
[ ] yield_df.groupby(['Area'],sort=True)['hg/ha_yield'].sum().nlargest(10)
```

```
Area
India             327428324
Brazil            167550906
Mexico            138788528
Japan             124470912
Australia         109113062
Pakistan           73897436
Indonesia          69193586
United Kingdom     55410990
Turkey             52263950
Spain              46773540
Name: hg/ha_yield, dtype: int64
```

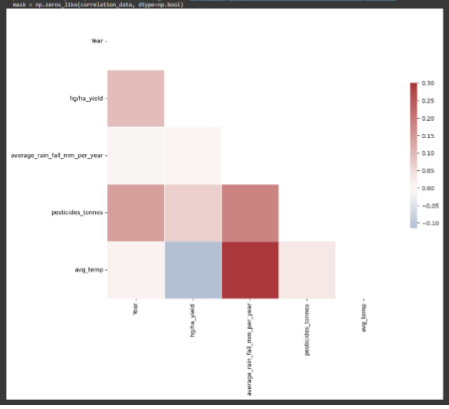
```
[ ] yield_df.groupby(['Item','Area'],sort=True)['hg/ha_yield'].sum().nlargest(10)
```

```
Item          Area
Cassava       India             142810626
Potatoes      India              92122514
              Brazil             49682168
              United Kingdom     46705145
              Australia          45679346
Sweet potatoes India             44439538
Potatoes      Japan              42918726
              Mexico             42053880
Sweet potatoes Mexico            35880932
              Australia          35558294
Name: hg/ha_yield, dtype: int64
```

```
[ ] import sklearn
    import seaborn as sns
    import matplotlib.pyplot as plt
```

```
[ ] correlation_data=yield_df.select_dtypes(include=[np.number]).corr()

    mask = np.zeros_like(correlation_data, dtype=np.bool)
    mask[np.tril_indices_from(mask)] = True

    f, ax = plt.subplots(figsize=(11, 9))

    # Generate a custom diverging colormap
    cmap = sns.palette="vlag"

    # Draw the heatmap with the mask and correct aspect ratio
    sns.heatmap(correlation_data, mask=mask, cmap=cmap, vmax=.3, center=0,
                square=True, linewidths=.5, cbar_kws={"shrink": .5});
```

```
<ipython-input-41-42a3a7d39da1>:1: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To silence this warning, use `bool` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.bool_` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  mask = np.zeros_like(correlation_data, dtype=np.bool)
```




```
[ ] yield_df.head()
```

	Area	Item	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	Albania	Maize	1990	36613	1485.0	121.0	16.37
1	Albania	Potatoes	1990	66667	1485.0	121.0	16.37
2	Albania	Rice, paddy	1990	23333	1485.0	121.0	16.37
3	Albania	Sorghum	1990	12500	1485.0	121.0	16.37
4	Albania	Soybeans	1990	7000	1485.0	121.0	16.37

```
[ ] from sklearn.preprocessing import OneHotEncoder
```

```
[ ] yield_df_onehot = pd.get_dummies(yield_df, columns=['Area','Item'], prefix = ['Country','Item'])
    features=yield_df_onehot.loc[:, yield_df_onehot.columns != 'hg/ha_yield']
    labels=yield_df['hg/ha_yield']
    features.head()
```

	Year	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Country_Argentina	Country_Armenia	Country_Australia	...	Item_Cassava	Item_Maize	Item_Plantains and others	Item_Potatoes	Item_Rice, paddy	Item_Sorghum	Item_Soybeans	Item_Sweet potatoes	Item_Wheat	Item_Yams
0	1990	1485.0	121.0	16.37	1	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
1	1990	1485.0	121.0	16.37	1	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
2	1990	1485.0	121.0	16.37	1	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
3	1990	1485.0	121.0	16.37	1	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
4	1990	1485.0	121.0	16.37	1	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0

5 rows × 115 columns

```
[ ] features = features.drop(['Year'], axis=1)
```

```
[ ] features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28242 entries, 0 to 28241
Columns: 114 entries, average_rain_fall_mm_per_year to Item_Yams
dtypes: float64(3), uint8(111)
memory usage: 3.9 MB
```

```
[ ] features.head()
```

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Country_Argentina	Country_Armenia	Country_Australia	Country_Austria	...	Item_Cassava	Item_Maize	Item_Plantains and others	Item_Potatoes	Item_Rice, paddy	Item_Sorghum	Item_Soybeans	Item_Sweet potatoes	Item_Wheat	Item_Yams
0	1485.0	121.0	16.37	1	0	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
1	1485.0	121.0	16.37	1	0	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
2	1485.0	121.0	16.37	1	0	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
3	1485.0	121.0	16.37	1	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
4	1485.0	121.0	16.37	1	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0

5 rows × 114 columns

```
[ ] from sklearn.preprocessing import MinMaxScaler
    scaler=MinMaxScaler()
    features=scaler.fit_transform(features)
```

```
features
```

```
array([[4.49670743e-01, 3.28094097e-04, 5.13458262e-01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [4.49670743e-01, 3.28094097e-04, 5.13458262e-01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [4.49670743e-01, 3.28094097e-04, 5.13458262e-01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [1.98028222e-01, 6.91361288e-03, 6.28968818e-01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.98028222e-01, 6.91361288e-03, 6.28968818e-01, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.98028222e-01, 6.91361288e-03, 6.28968818e-01, ...,
        0.00000000e+00, 1.00000000e+00, 0.00000000e+00]])
```

```
from sklearn.model_selection import train_test_split
train_data, test_data, train_labels, test_labels = train_test_split(features, label, test_size=0.3, random_state=42)
```

```
#write final df to csv file
#yield_df.to_csv('yield_df.csv')
```

```
from sklearn.model_selection import train_test_split
train_data, test_data, train_labels, test_labels = train_test_split(features, label, test_size=0.3, random_state=42)
```

```
from sklearn.metrics import r2_score
def compare_models(model):
    model_name = model.__class__.__name__
    fit=model.fit(train_data,train_labels)
    y_pred=fit.predict(test_data)
    r2=r2_score(test_labels,y_pred)
    return([model_name,r2])
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn import svm
from sklearn.tree import DecisionTreeRegressor

models = [
    GradientBoostingRegressor(n_estimators=200, max_depth=3, random_state=0),
    RandomForestRegressor(n_estimators=200, max_depth=3, random_state=0),
    svm.SVR(),
   DecisionTreeRegressor()
   ]
```

```
model_train=list(map(compare_models,models))
```

```
print("model_train, sep = "\n")
```

```
['GradientBoostingRegressor', 0.8965756891926441]
['RandomForestRegressor', 0.6843532317855172]
['SVR', -0.10543218673573705]
['DecisionTreeRegressor', 0.9688015631029338]
```

```
yield_df_onehot = yield_df_onehot.drop(['Year'], axis=1)
```

```
yield_df_onehot.head()
```

	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Country_Argentina	Country_Armenia	Country_Australia	...	Item_Cassava	Item_Maize	Item_Plantains and others	Item_Potatoes	Item_Rice, paddy	Item_Sorghum	Item_Soybeans	Item_Sweet potatoes	Item_Wheat	Item_Yams
0	36613	1485.0	121.0	16.37	1	0	0	0	0	0	...	0	1	0	0	0	0	0	0	0	0
1	66667	1485.0	121.0	16.37	1	0	0	0	0	0	...	0	0	0	1	0	0	0	0	0	0
2	23333	1485.0	121.0	16.37	1	0	0	0	0	0	...	0	0	0	0	1	0	0	0	0	0
3	12500	1485.0	121.0	16.37	1	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
4	7000	1485.0	121.0	16.37	1	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0

5 rows × 115 columns

```
#setting test data to columns from dataframe and excluding 'hg/ha_yield' values where ML model should be predicting

test_df=pd.DataFrame(test_data,columns=yield_df_onehot.loc[:, yield_df_onehot.columns != 'hg/ha_yield'].columns)

# using stack function to return a reshaped DataFrame by pivoting the columns of the current dataFrame

cntry=test_df[[col for col in test_df.columns if 'Country' in col]].stack()[test_df[[col for col in test_df.columns if 'Country' in col]].stack()>0]
cntrylist=list(pd.DataFrame(cntry).index.get_level_values(1))
countries=[i.split("_")[1] for i in cntrylist]
itm=test_df[[col for col in test_df.columns if 'Item' in col]].stack()[test_df[[col for col in test_df.columns if 'Item' in col]].stack()>0]
itmlist=list(pd.DataFrame(itm).index.get_level_values(1))
items=[i.split("_")[1] for i in itmlist]
```

```
test_df.head()
```

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country_Albania	Country_Algeria	Country_Angola	Country_Argentina	Country_Armenia	Country_Australia	Country_Austria	...	Item_Cassava	Item_Maize	Item_Plantains and others	Item_Potatoes	Item_Rice, paddy	Item_Sorghum	Item_Soybeans	Item_Sweet potatoes	Item_Wheat	Item_Yams
0	0.183443	0.110718	0.542078	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
1	0.458451	0.000413	0.827257	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	1.0	0.0
2	0.183443	0.108159	0.516228	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
3	1.000000	0.224154	0.860971	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.458451	0.000355	0.825213	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	1.0	0.0	0.0

5 rows × 114 columns

```
test_df.drop([col for col in test_df.columns if 'Item' in col],axis=1,inplace=True)
test_df.drop([col for col in test_df.columns if 'Country' in col],axis=1,inplace=True)
test_df.head()
```

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
0	0.183443	0.110718	0.542078
1	0.458451	0.000413	0.827257
2	0.183443	0.108159	0.516228
3	1.000000	0.224154	0.860971
4	0.458451	0.000355	0.825213

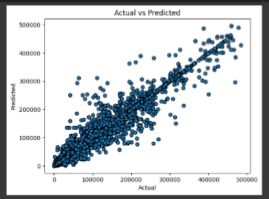
```
test_df['Country']=countries
test_df['Item']=items
test_df.head()
```

	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp	Country	Item
0	0.183443	0.110718	0.542078	Spain	Rice, paddy
1	0.458451	0.000413	0.827257	Madagascar	Wheat
2	0.183443	0.108159	0.516228	Spain	Sorghum
3	1.000000	0.224154	0.860971	Colombia	Potatoes
4	0.458451	0.000355	0.825213	Madagascar	Sweet potatoes

```
clf=DecisionTreeRegressor()
model=clf.fit(train_data,train_labels)

test_df['yield_predicted']= model.predict(test_data)
test_df['yield_actual']=pd.DataFrame(test_labels)["hg/ha_yield"].tolist()
test_group=test_df.groupby("Item")
test_group.apply(lambda x: r2_score(x.yield_actual,x.yield_predicted))
```

```
Item
Cassava                0.928120
Maize                  0.892341
Plantains and others   0.773807
Potatoes               0.909178
Rice, paddy            0.891916
Sorghum                0.787865
Soybeans               0.816894
Sweet potatoes         0.867001
Wheat                  0.924464
Yams                   0.938674
dtype: float64
```

```
# So let's run the model actual values against the predicted ones

fig, ax = plt.subplots()

ax.scatter(test_df["yield_actual"], test_df["yield_predicted"],edgecolors=(0, 0, 0))

ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
ax.set_title("Actual vs Predicted")
plt.show()
```



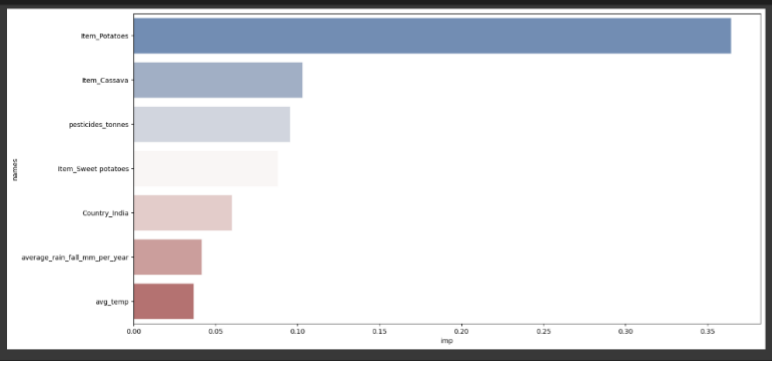

```
def adjusted_r_squared(y,yhat,x):
    score=1- (((1-(r2_score(y,yhat)))*(len(y)-1))/(len(y)-x.shape[1]-1))
    return score

test_group.apply(lambda x: adjusted_r_squared(x.yield_actual,x.yield_predicted,x))
```

```
Item
Cassava                0.927586
Maize                  0.891645
Plantains and others   0.762227
Potatoes               0.908618
Rice, paddy            0.891057
Sorghum                0.785964
Soybeans               0.815408
Sweet potatoes         0.845576
Wheat                  0.923918
Yams                   0.928242
dtype: float64
```

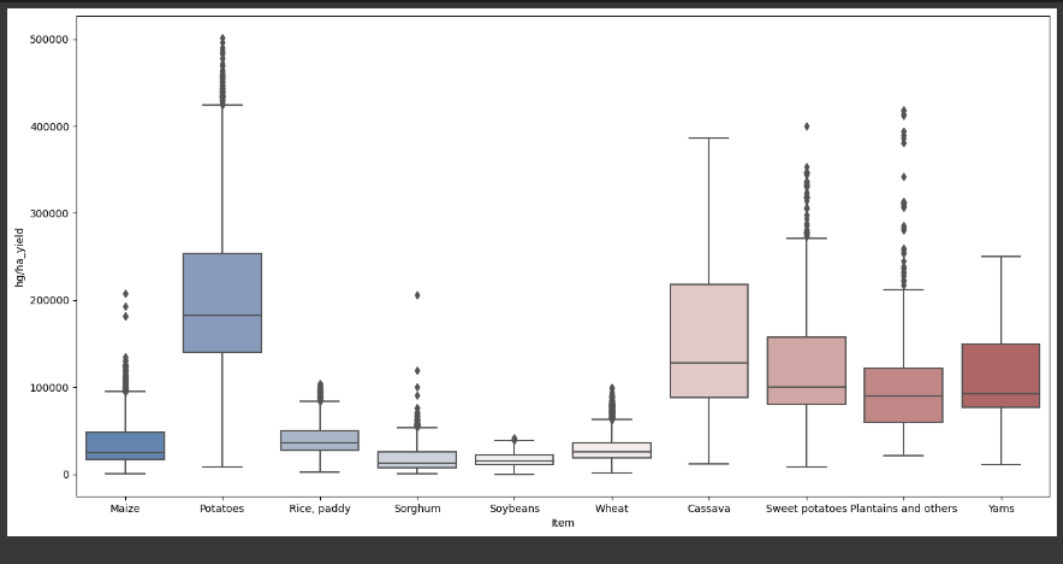
```
varimp= {'imp':model.feature_importances_, 'names':yield_df_onehot.columns[yield_df_onehot.columns!="hg/ha_yield"]}
```

```
[ ] a4_dims = (8.27,16.7)

    fig, ax = plt.subplots(figsize=a4_dims)
    df=pd.DataFrame.from_dict(varimp)
    df.sort_values(ascending=False,by=["imp"],inplace=True)
    df=df.dropna()
    sns.barplot(x="imp",y="names",palette="vlag",data=df,orient="h",ax=ax);
```


```
[ ] #7 most important factors that affect crops
    a4_dims = (16.7, 8.27)

    fig, ax = plt.subplots(figsize=a4_dims)
    df=pd.DataFrame.from_dict(varimp)
    df.sort_values(ascending=False,by=["imp"],inplace=True)
    df=df.dropna()
    df=df.nlargest(7, 'imp')
    sns.barplot(x='imp',y='names',palette="vlag",data=df,orient="h",ax=ax);
```




```
[ ] #Boxplot that shows yield for each item
    a4_dims = (16.7, 8.27)

    fig, ax = plt.subplots(figsize=a4_dims)
    sns.boxplot(x="Item",y="hg/ha_yield",palette="vlag",data=yield_df,ax=ax);
```



```
[ ] from __future__ import print_function
    import pandas as pd # data analysis
    import numpy as np # linear algebra

    #import libraries for data visualization
    import matplotlib.pyplot as plt
    %matplotlib inline
    import seaborn as sns
    import plotly.graph_objects as go
    import plotly.express as px
    from plotly.subplots import make_subplots


    from sklearn.metrics import classification_report
    from sklearn import metrics
    from sklearn import tree
    from sklearn.model_selection import cross_val_score
    import warnings
    warnings.filterwarnings('ignore')
```

```
[ ] crop = pd.read_csv('Crop_recommendation.csv')
    crop.head(8)
```

	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
5	69	37	42	23.058049	83.370118	7.073454	251.055000	rice
6	69	55	38	22.708838	82.639414	5.700806	271.324860	rice
7	94	53	40	20.277744	82.894086	5.718627	241.974195	rice

```
[ ] crop.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   N            2200 non-null   int64  
 1   P            2200 non-null   int64  
 2   K            2200 non-null   int64  
 3   temperature  2200 non-null   float64
 4   humidity     2200 non-null   float64
 5   ph           2200 non-null   float64
 6   rainfall     2200 non-null   float64
 7   label        2200 non-null   object 
dtypes: float64(4), int64(3), object(1)
memory usage: 137.6+ KB
```

```
[ ] crop.describe()
```

	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

```
[ ] crop.columns
```

```
Index(['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label'], dtype='object')
```

```
[ ] crop.shape
```

```
(2200, 8)
```

```
[ ] crop['label'].unique()
```

```
array(['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas',
       'mothbeans', 'mungbean', 'blackgram', 'lentil', 'pomegranate',
       'banana', 'mango', 'grapes', 'watermelon', 'muskmelon', 'apple',
       'orange', 'papaya', 'coconut', 'cotton', 'jute', 'coffee'],
      dtype=object)
```

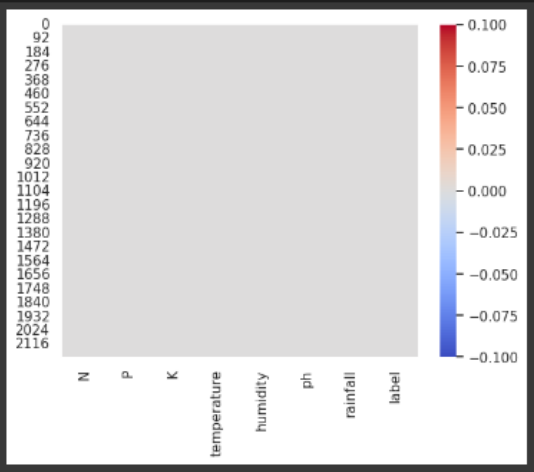
```
[ ] crop['label'].nunique()
```

```
22
```

```
[ ] crop['label'].value_counts()
```

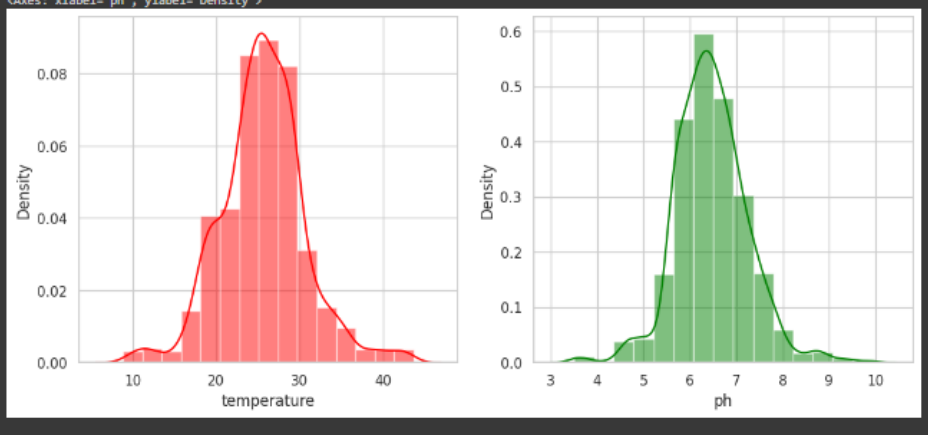
```
rice           100
maize          100
jute           100
cotton         100
coconut        100
papaya         100
orange         100
apple          100
muskmelon      100
watermelon     100
grapes         100
mango          100
banana         100
pomegranate    100
lentil         100
blackgram  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

```
[ ] sns.heatmap(crop.isnull(),cmap="coolwarm")
    plt.show()
```



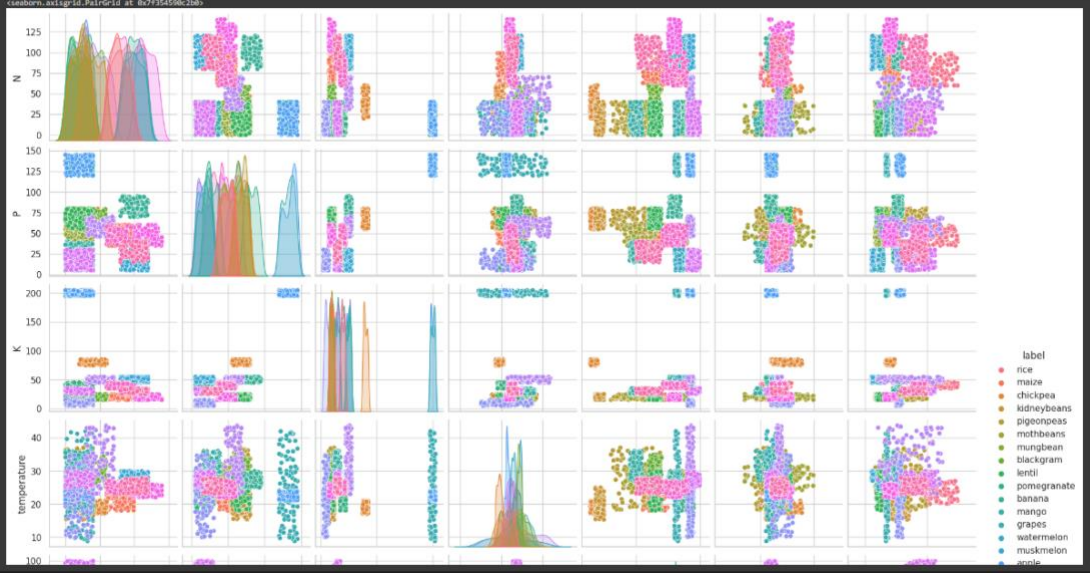

```
[ ] plt.figure(figsize=(12,5))
    plt.subplot(1, 2, 1)
    # sns.distplot(df_setosa['sepal_length'],kde=True,color='green',bins=20,hist_kws={'alpha':0.3})
    sns.distplot(crop['temperature'],color="red",bins=15,hist_kws={'alpha':0.5})
    plt.subplot(1, 2, 2)
    sns.distplot(crop['ph'],color="green",bins=15,hist_kws={'alpha':0.5})
```

```
<Axes: xlabel='ph', ylabel='Density'>
```

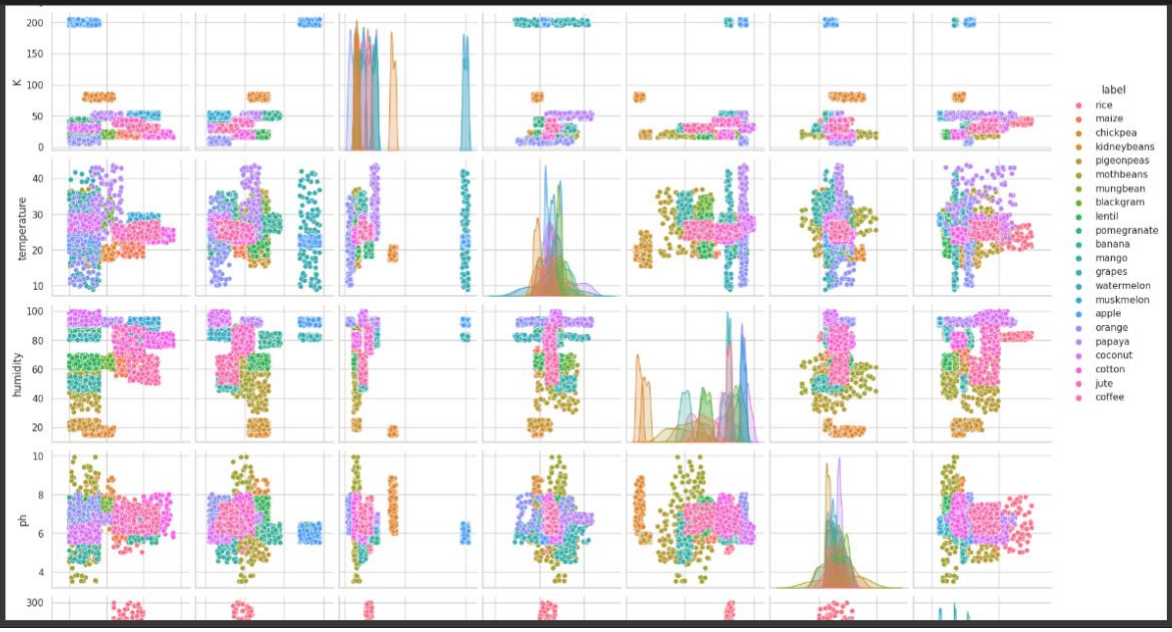



```
[ ] sns.pairplot(crop,hue = 'label')
```

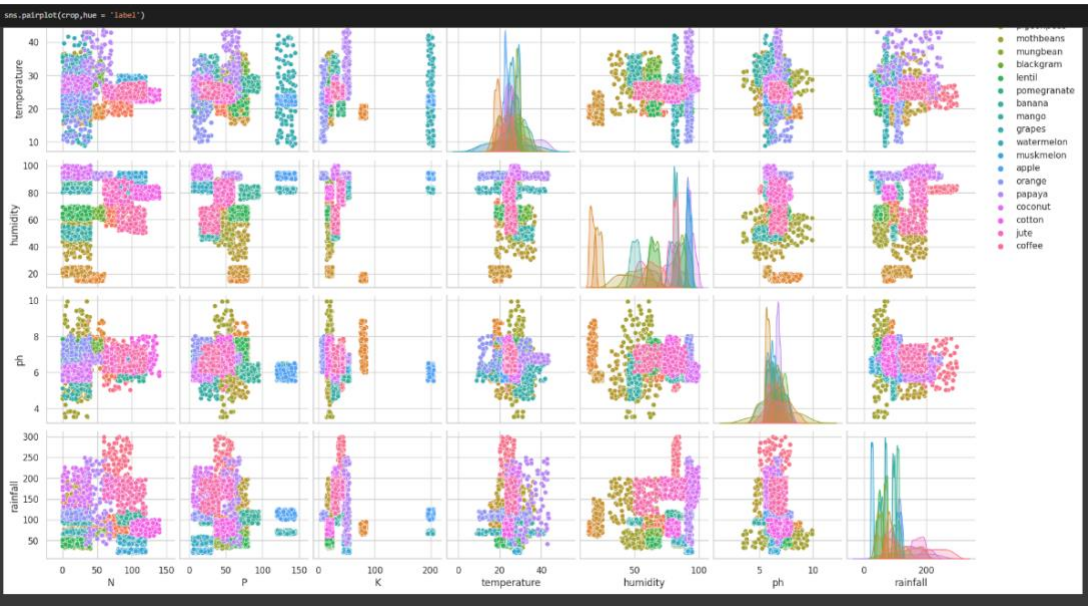
```
<seaborn.axisgrid.PairGrid at 0x7f35459bc2b0>
```



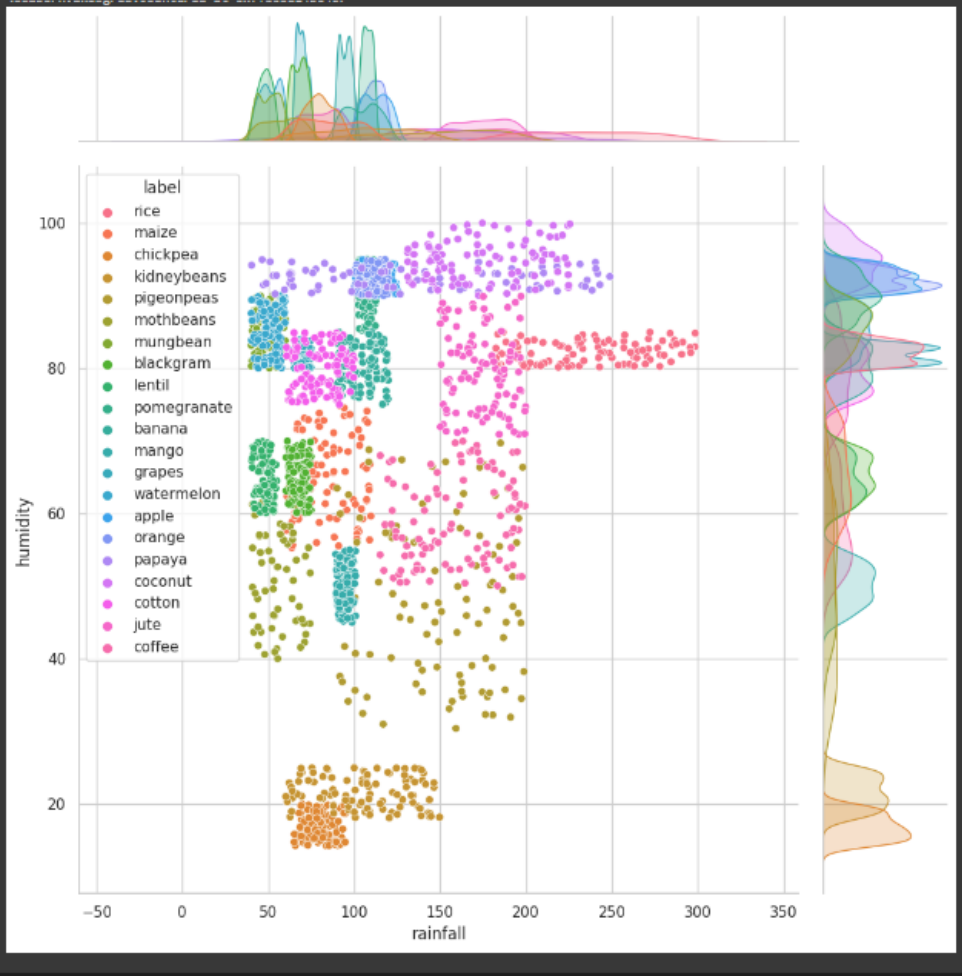
```
sns.pairplot(crop,hue = 'label')
```



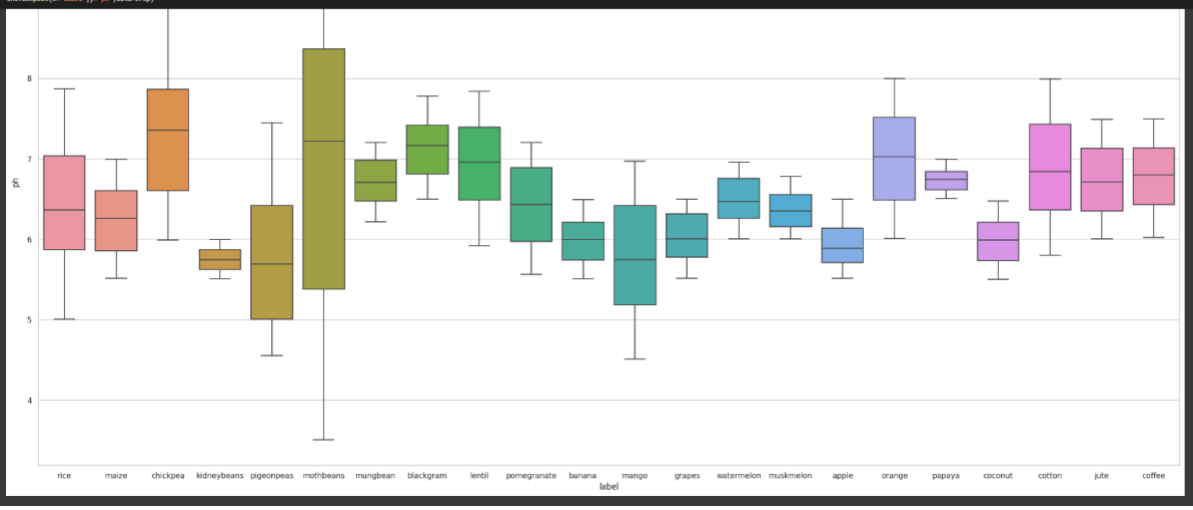

```
[ ] sns.pairplot(crop,hue = 'label')
```



```
[ ] sns.jointplot(x="rainfall",y="humidity",data=crop[(crop['temperature']<40) &
                                                      (crop['rainfall']>40)],height=10,hue="label")
```




```
[ ] sns.set_theme(style="whitegrid")
    fig, ax = plt.subplots(figsize=(30,15))
    sns.boxplot(x='label',y='ph',data=crop)
```

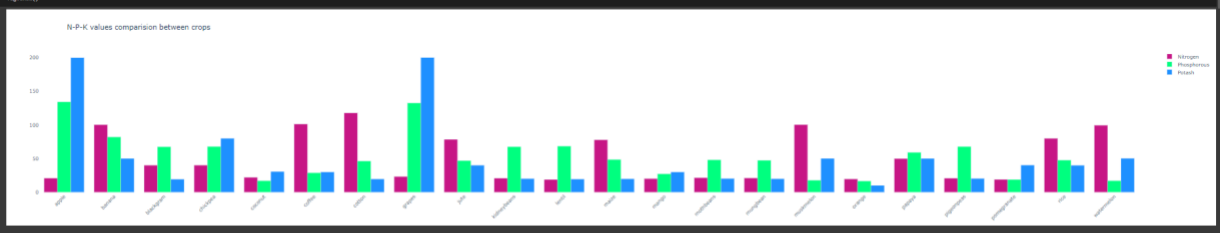


```
[ ] fig, ax = plt.subplots(1, 1, figsize=(15, 9))
    sns.heatmap(crop.corr(), annot=True,cmap='viridis')
    ax.set(xlabel='features')
    ax.set(ylabel='features')

    plt.title('Correlation between different features', fontsize = 15, c='black')
    plt.show()
```


```
[ ] crop_summary = pd.pivot_table(crop,index=['label'],aggfunc='mean')
    crop_summary.head()
```

label	K	N	P	humidity	ph	rainfall	temperature
apple	199.89	20.80	134.22	92.333383	5.929663	112.654779	22.630942
banana	50.05	100.23	82.01	80.358123	5.983893	104.626980	27.376798
blackgram	19.24	40.02	67.47	65.118426	7.133952	67.884151	29.973340
chickpea	79.92	40.09	67.79	16.860439	7.336957	80.058977	18.872847
coconut	30.59	21.98	16.93	94.844272	5.976562	175.686646	27.409892

```
fig = go.Figure()
fig.add_trace(go.Bar(
    x=crop_summary.index,
    y=crop_summary['N'],
    name='Nitrogen',
    marker_color='mediumvioletred'
))
fig.add_trace(go.Bar(
    x=crop_summary.index,
    y=crop_summary['P'],
    name='Phosphorous',
    marker_color='springgreen'
))
fig.add_trace(go.Bar(
    x=crop_summary.index,
    y=crop_summary['K'],
    name='Potash',
    marker_color='dodgerblue'
))

fig.update_layout(title='N-P-K values comparision between crops',
                  plot_bgcolor='white',
                  barmode='group',
                  xaxis_tickangle=-45)

fig.show()
```




```
features = crop[['N', 'P','K','temperature', 'humidity', 'ph', 'rainfall']]
target = crop['label']
```

```
acc = []
model = []
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(features,target,test_size = 0.2,random_state =2)
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(features,target,test_size = 0.2,random_state =2)
```

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()

knn.fit(x_train,y_train)

predicted_values = knn.predict(x_test)

x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)
model.append('K Nearest Neighbours')
print("KNN Accuracy is: ", x)

print(classification_report(y_test,predicted_values))
```

```
KNN Accuracy is:  0.975
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.94      1.00      0.97        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       0.95      1.00      0.98        20
      grapes       1.00      1.00      1.00        18
        jute       0.89      0.86      0.87        28
 kidneybeans       0.93      1.00      0.97        14
       lentil       0.96      1.00      0.98        23
        maize       1.00      0.95      0.98        21
        mango       1.00      1.00      1.00        26
    mothbeans       1.00      0.89      0.94        19
     mungbean       1.00      1.00      1.00        24
    muskmelon       1.00      1.00      1.00        23
       orange       1.00      1.00      1.00        29
       papaya       1.00      1.00      1.00        19
   pigeonpeas       1.00      0.94      0.97        18
  pomegranate       1.00      1.00      1.00        17
         rice       0.76      0.81      0.79        16
   watermelon       1.00      1.00      1.00        15

     accuracy                           0.97       440
    macro avg       0.97      0.98      0.97       440
 weighted avg       0.98      0.97      0.98       440
```

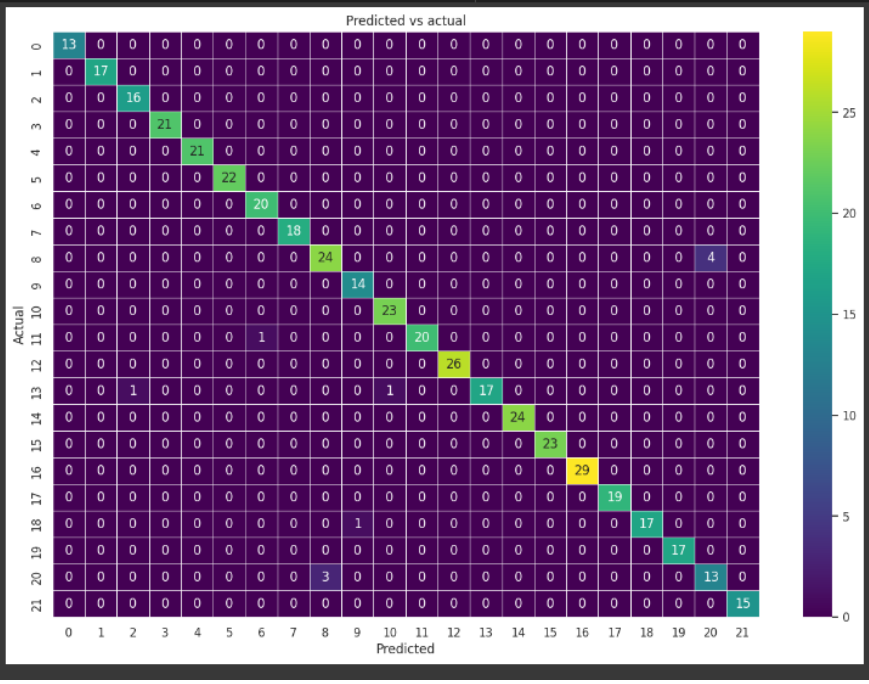
```
score = cross_val_score(knn,features,target,cv=5)
print('Cross validation score: ',score)
```

```
Cross validation score:  [0.97727273 0.98181818 0.97954545 0.97954545 0.97954545]
```

```
#Print Train Accuracy
knn_train_accuracy = knn.score(x_train,y_train)
print('knn_train_accuracy = ',knn.score(x_train,y_train))
#Print Test Accuracy
knn_test_accuracy = knn.score(x_test,y_test)
print('knn_test_accuracy = ',knn.score(x_test,y_test))
```

```
knn_train_accuracy =  0.9886363636363636
knn_test_accuracy =  0.975
```

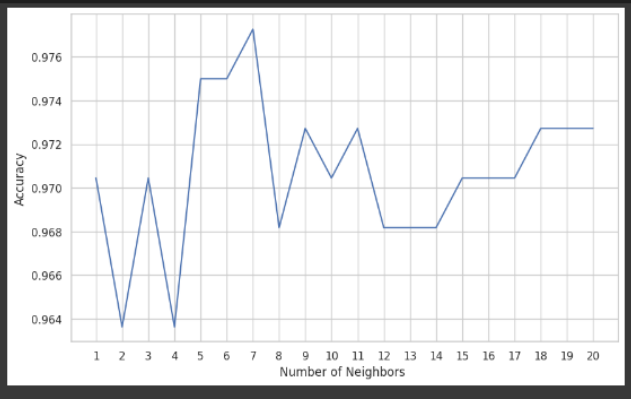
```
y_pred = knn.predict(x_test)
y_true = y_test

from sklearn.metrics import confusion_matrix

cm_knn = confusion_matrix(y_true,y_pred)

f, ax = plt.subplots(figsize=(15,10))
sns.heatmap(cm_knn, annot=True, linewidth=0.5, fmt=".0f",cmap='viridis', ax = ax)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Predicted vs actual')
plt.show()
```



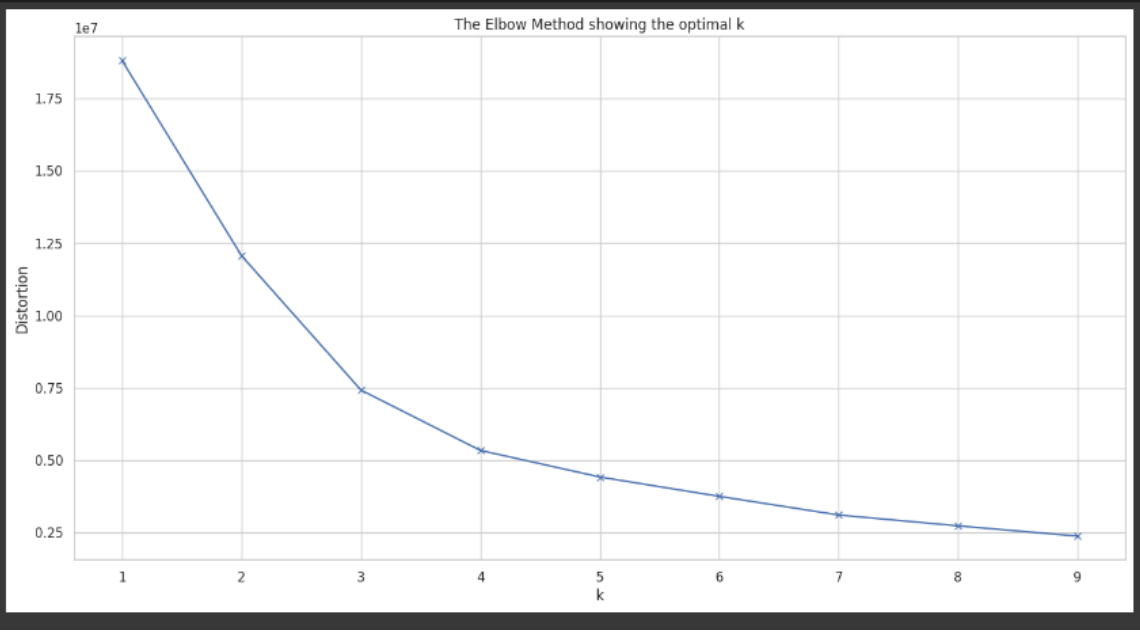

```
mean_acc = np.zeros(20)
for i in range(1,21):
    #Train Model and Predict
    knn = KNeighborsClassifier(n_neighbors = i).fit(x_train,y_train)
    yhat= knn.predict(x_test)
    mean_acc[i-1] = metrics.accuracy_score(y_test, yhat)

mean_acc
```

```
array([0.97045455, 0.96363636, 0.97045455, 0.96363636, 0.975     ,
       0.975     , 0.97727273, 0.96818182, 0.97272727, 0.97045455,
       0.97272727, 0.96818182, 0.96818182, 0.96818182, 0.97045455,
       0.97045455, 0.97045455, 0.97272727, 0.97272727, 0.97272727])
```

```
loc = np.arange(1,21,step=1.0)
plt.figure(figsize = (10, 6))
plt.plot(range(1,21), mean_acc)
plt.xticks(loc)
plt.xlabel('Number of Neighbors ')
plt.ylabel('Accuracy')
plt.show()
```



```
[ ] import matplotlib.pyplot as plt
    %matplotlib inline
    from sklearn.cluster import KMeans
    distortions = []
    K = range(1,10)
    for k in K:
        kmeanModel = KMeans(n_clusters=k)
        kmeanModel.fit(features)
        distortions.append(kmeanModel.inertia_)

    plt.figure(figsize=(16,8))
    plt.plot(K, distortions, 'bx-')
    plt.xlabel('k')
    plt.ylabel('Distortion')
    plt.title('The Elbow Method showing the optimal k')
    plt.show()
```




```
[ ] from sklearn.model_selection import GridSearchCV

[ ] grid_params = { 'n_neighbors' : [12,13,14,15,16,17,18],
                    'weights' : ['uniform','distance'],
                    'metric' : ['minkowski','euclidean','manhattan']}

[ ] gs = GridSearchCV(KNeighborsClassifier(), grid_params, verbose = 1, cv=3, n_jobs = -1)

[ ] g_res = gs.fit(x_train, y_train)

    Fitting 3 folds for each of 42 candidates, totalling 126 fits

[ ] g_res.best_score_

    0.9789756440743992

[ ] g_res.best_params_

    {'metric': 'manhattan', 'n_neighbors': 12, 'weights': 'distance'}

[ ] # Using the best hyperparameters
    knn_1 = KNeighborsClassifier(n_neighbors = 12, weights = 'distance',algorithm = 'brute',metric = 'manhattan')
    knn_1.fit(x_train, y_train)

    KNeighborsClassifier
    KNeighborsClassifier(algorithm='brute', metric='manhattan', n_neighbors=12,
                         weights='distance')

[ ] # Training & Testing accuracy after applying hyper parameter
    knn_train_accuracy = knn_1.score(x_train,y_train)
    print("knn_train_accuracy = ",knn_1.score(x_train,y_train))
    #Print Test Accuracy
    knn_test_accuracy = knn_1.score(x_test,y_test)
    print("knn_test_accuracy = ",knn_1.score(x_test,y_test))

    knn_train_accuracy =  1.0
    knn_test_accuracy =  0.9727272727272728
```

```
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier(criterion="entropy",random_state=2,max_depth=5)

DT.fit(x_train,y_train)

predicted_values = DT.predict(x_test)
x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)
model.append('Decision Tree')
print("Decision Tree's Accuracy is: ", x*100)

print(classification_report(y_test,predicted_values))
```

```
Decision Tree's Accuracy is:  90.0
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.59      1.00      0.74        16
    chickpea       1.00      1.00      1.00        21
     coconut       0.91      1.00      0.95        21
      coffee       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.74      0.93      0.83        28
 kidneybeans       0.00      0.00      0.00        14
      lentil       0.68      1.00      0.81        23
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        26
   mothbeans       0.00      0.00      0.00        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      0.84      0.91        19
  pigeonpeas       0.62      1.00      0.77        18
 pomegranate       1.00      1.00      1.00        17
        rice       1.00      0.62      0.77        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.90       440
   macro avg       0.84      0.88      0.85       440
weighted avg       0.86      0.90      0.87       440
```

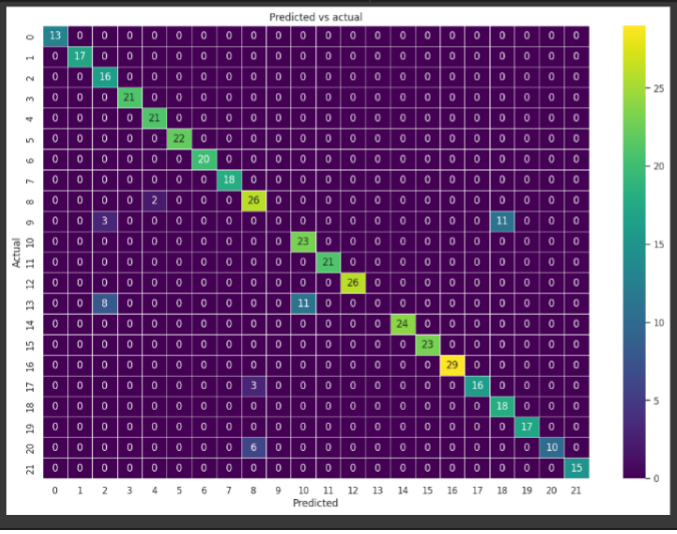
```
score = cross_val_score(DT, features, target,cv=5)
print('Cross validation score: ',score)
```

```
Cross validation score:  [0.93636364 0.90909091 0.91818182 0.87045455 0.93636364]
```

```
#Print Train Accuracy
dt_train_accuracy = DT.score(x_train,y_train)
print("Training accuracy = ",DT.score(x_train,y_train))
#Print Test Accuracy
dt_test_accuracy = DT.score(x_test,y_test)
print("Testing accuracy = ",DT.score(x_test,y_test))
```

```
Training accuracy =  0.8818181818181818
Testing accuracy =  0.9
```

```
y_pred = DT.predict(x_test)
y_true = y_test

from sklearn.metrics import confusion_matrix

cm_dt = confusion_matrix(y_true,y_pred)

f, ax = plt.subplots(figsize=(15,10))
sns.heatmap(cm_dt, annot=True, linewidth=0.5, fmt=".0f",  cmap='viridis', ax = ax)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title('Predicted vs actual')
plt.show()
```



```
[ ] from sklearn.ensemble import RandomForestClassifier

    RF = RandomForestClassifier(n_estimators=20, random_state=0)
    RF.fit(x_train,y_train)

    predicted_values = RF.predict(x_test)

    x = metrics.accuracy_score(y_test, predicted_values)
    acc.append(x)
    model.append('RF')
    print('Random Forest Accuracy is: ', x)

    print(classification_report(y_test,predicted_values))
```

```
Random Forest Accuracy is:  0.990909090909091
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       0.94      1.00      0.97        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.90      1.00      0.95        28
 kidneybeans       1.00      1.00      1.00        14
       lentil       1.00      1.00      1.00        23
       maize       1.00      1.00      1.00        21
       mango       1.00      1.00      1.00        26
   mothbeans       1.00      0.95      0.97        19
    mungbean       1.00      1.00      1.00        24
   muskmelon       1.00      1.00      1.00        23
      orange       1.00      1.00      1.00        29
      papaya       1.00      1.00      1.00        19
  pigeonpeas       1.00      1.00      1.00        18
 pomegranate       1.00      1.00      1.00        17
        rice       1.00      0.81      0.90        16
  watermelon       1.00      1.00      1.00        15

    accuracy                           0.99       440
   macro avg       0.99      0.99      0.99       440
weighted avg       0.99      0.99      0.99       440
```

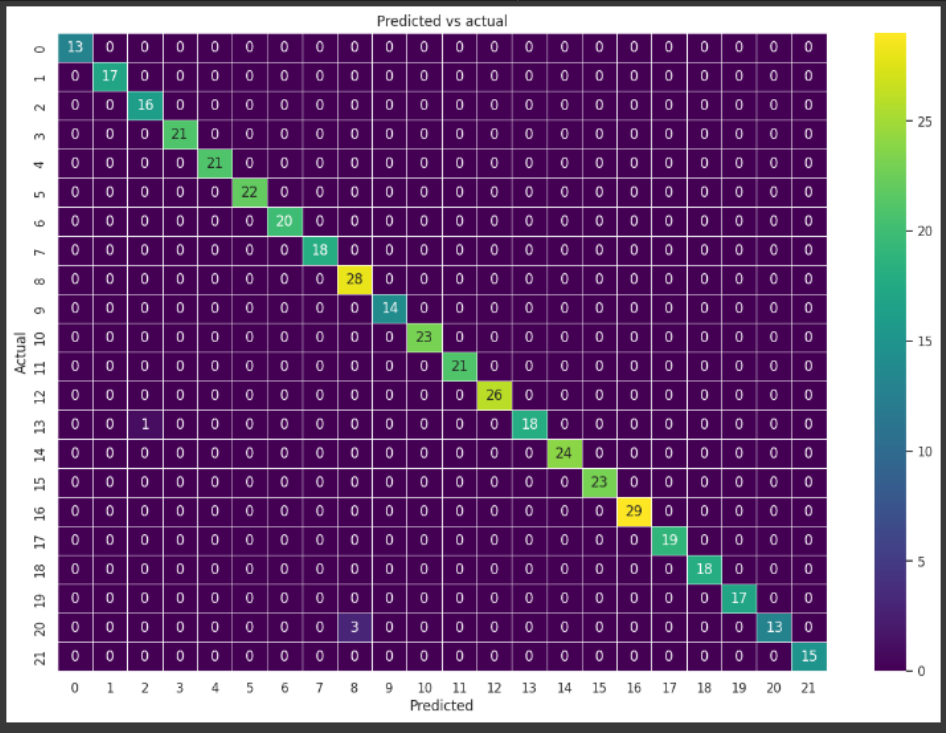
```
[ ] score = cross_val_score(RF,features,target,cv=5)
    print('Cross validation score: ',score)
```

```
Cross validation score:  [0.99772727 0.99545455 0.99772727 0.99318182 0.98863636]
```

```
[ ] #Print Train Accuracy
    rf_train_accuracy = RF.score(x_train,y_train)
    print("Training accuracy = ",RF.score(x_train,y_train))
    #Print Test Accuracy
    rf_test_accuracy = RF.score(x_test,y_test)
    print("Testing accuracy = ",RF.score(x_test,y_test))
```

```
Training accuracy =  1.0
Testing accuracy =  0.990909090909091
```

```
y_pred = RF.predict(x_test)
y_true = y_test

from sklearn.metrics import confusion_matrix

cm_rf = confusion_matrix(y_true,y_pred)

f, ax = plt.subplots(figsize=(15,10))
sns.heatmap(cm_rf, annot=True, linewidth=0.5, fmt=".0f",  cmap='viridis', ax = ax)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title('Predicted vs actual')
plt.show()
```



```
from sklearn.naive_bayes import GaussianNB
NaiveBayes = GaussianNB()

NaiveBayes.fit(x_train,y_train)

predicted_values = NaiveBayes.predict(x_test)
x = metrics.accuracy_score(y_test, predicted_values)
acc.append(x)
model.append('Naive Bayes')
print("Naive Bayes Accuracy is: ", x)

print(classification_report(y_test,predicted_values))
```

```
Naive Bayes Accuracy is:  0.990909090909091
              precision    recall  f1-score   support

       apple       1.00      1.00      1.00        13
      banana       1.00      1.00      1.00        17
   blackgram       1.00      1.00      1.00        16
    chickpea       1.00      1.00      1.00        21
     coconut       1.00      1.00      1.00        21
      coffee       1.00      1.00      1.00        22
      cotton       1.00      1.00      1.00        20
      grapes       1.00      1.00      1.00        18
        jute       0.88      1.00      0.93        28
 kidneybeans       1.00      1.00      1.00        14
       lentil       1.00      1.00      1.00        23
        maize       1.00      1.00      1.00        21
        mango       1.00      1.00      1.00        26
    mothbeans       1.00      1.00      1.00        19
     mungbean       1.00      1.00      1.00        24
    muskmelon       1.00      1.00      1.00        23
       orange       1.00      1.00      1.00        29
       papaya       1.00      1.00      1.00        19
   pigeonpeas       1.00      1.00      1.00        18
  pomegranate       1.00      1.00      1.00        17
         rice       1.00      0.75      0.86        16
   watermelon       1.00      1.00      1.00        15

     accuracy                           0.99       440
    macro avg       0.99      0.99      0.99       440
 weighted avg       0.99      0.99      0.99       440
```

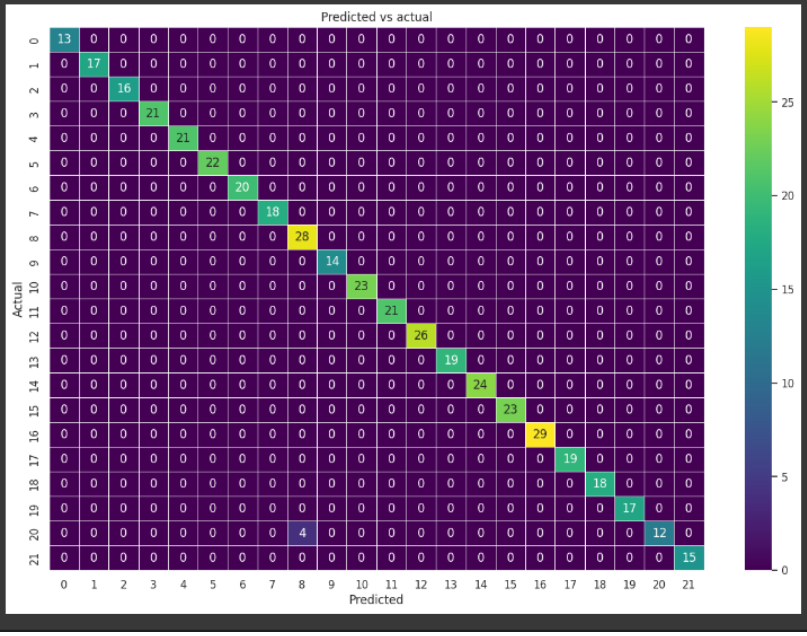
```
[ ] score = cross_val_score(NaiveBayes,features,target,cv=5)
    print('Cross validation score: ',score)
```

```
Cross validation score:  [0.99772727 0.99545455 0.99545455 0.99545455 0.99090909]
```

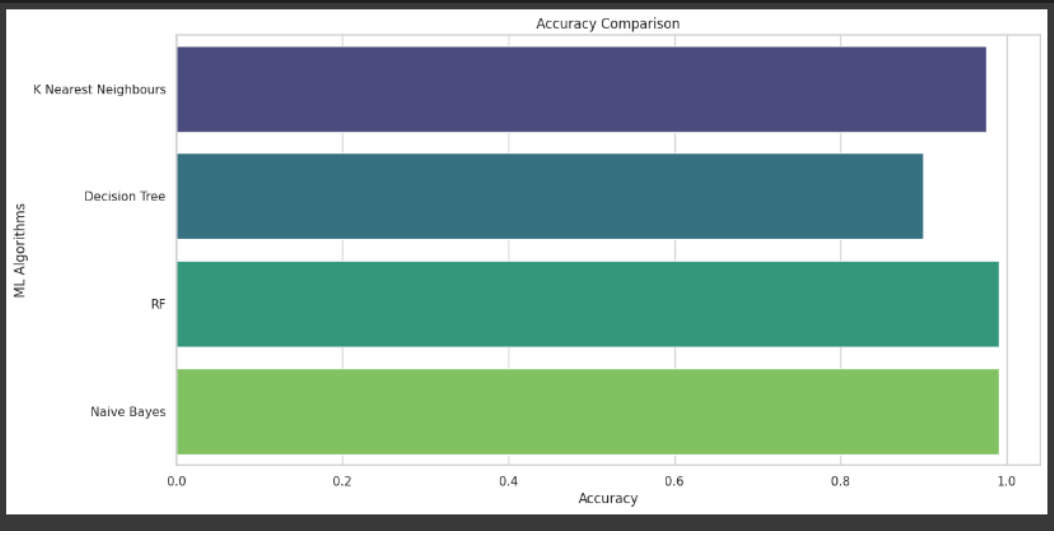
```
[ ] #Print Train Accuracy
    nb_train_accuracy = NaiveBayes.score(x_train,y_train)
    print("Training accuracy = ",NaiveBayes.score(x_train,y_train))
    #Print Test Accuracy
    nb_test_accuracy = NaiveBayes.score(x_test,y_test)
    print("Testing accuracy = ",NaiveBayes.score(x_test,y_test))
```

```
Training accuracy =  0.9960227272727272
Testing accuracy =  0.990909090909091
```

```
[ ] y_pred = NaiveBayes.predict(x_test)
    y_true = y_test

    from sklearn.metrics import confusion_matrix

    cm_nb = confusion_matrix(y_true,y_pred)

    f, ax = plt.subplots(figsize=(15,10))
    sns.heatmap(cm_nb, annot=True, linewidth=0.5, fmt=".0f",  cmap='viridis', ax = ax)
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title('Predicted vs actual')
    plt.show()
```



```
[ ] plt.figure(figsize=[14,7],dpi = 100, facecolor='white')
    plt.title('Accuracy Comparison')
    plt.xlabel('Accuracy')
    plt.ylabel('ML Algorithms')
    sns.barplot(x = acc,y = model,palette='viridis')
    plt.savefig('plot.png', dpi=300, bbox_inches='tight')
```




```
label = ['KNN', 'Decision Tree','Random Forest','Naive Bayes']
Test = [knn_test_accuracy, dt_test_accuracy,rf_test_accuracy,
        nb_test_accuracy]
Train = [knn_train_accuracy,  dt_train_accuracy, rf_train_accuracy,
         nb_train_accuracy]

f, ax = plt.subplots(figsize=(20,7)) # set the size that you'd like (width, height)
X_axis = np.arange(len(label))
plt.bar(X_axis - 0.2,Test, 0.4, label = 'Test', color=('midnightblue'))
plt.bar(X_axis + 0.2,Train, 0.4, label = 'Train', color=('mediumaquamarine'))
plt.xticks(X_axis, label)
plt.xlabel("ML algorithms")
plt.ylabel("Accuracy")
plt.title("Testing vs Training Accuracy")
plt.legend()
#plt.savefig('train vs test.png')
plt.show()
```