AGROADVISER USING ML AND DL

CLASSIFICATION METHOD

Under the guidance of S.Naveen Kumar

CONTENT TODAY

- Data Preprocessing
- EDA
- Training the Model
- Comparing the Results
- Conclusion
- Literature Review
- References
- Code

ABOUT THE DATA

The dataset contains information on various crops, environmental factors, and recommended cultivation practices. It aims to assist farmers in optimizing crop yields and quality based on data-driven insights.

OBJECTIVE

The objective is to develop predictive models that can accurately estimate the compressive strength of concrete based on various input variables.

THE PATH

Implementing multiple machine learning and deep learning models to fit the best model for the dataset.

DATA AND DATA QUALITY CHECK

DATA INTRODUCTION

There are 2200 instances and 8 attributes in the given dataset. It consists of 7 quantitative input variables, and 1 quantitative output variable.

MISSING VALUES

There are no missing values in the given data.

DROPPED COLUMNS

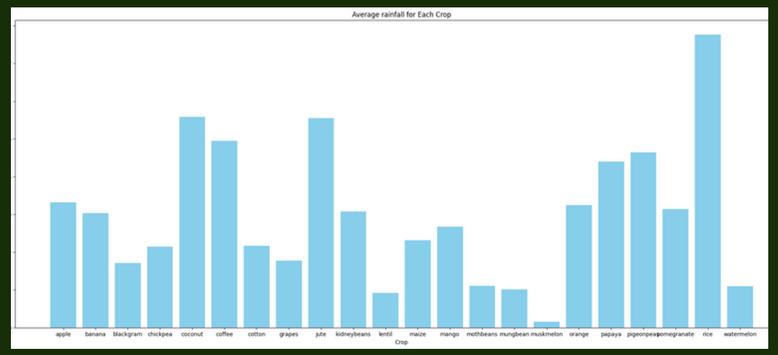
There are no dropped columns.

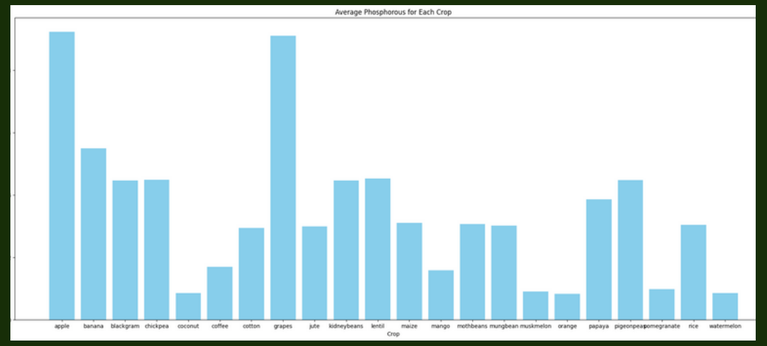
VARIABLES

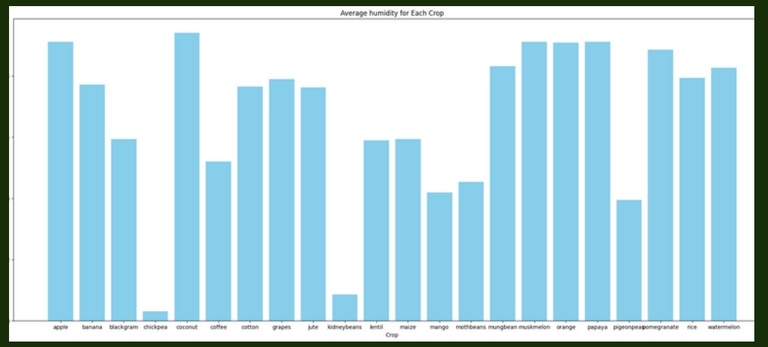
Variable Name	Variable Description
Nitrogen	This is a chemical element essential for plant growth. It plays a crucial role in protein synthesis, leaf development, and overall plant vigor.
Phosphorous	Phosphorus, primarily involved in energy transfer and storage within cells. It plays a crucial role in processes like photosynthesis, DNA synthesis, and root development in plants.
Potassium	Potassium, playing a key role in photosynthesis, enzyme activation, and water regulation. Its deficiency can lead to stunted growth, reduced crop yields, and susceptibility to diseases in plants
Temperature	Temperature significantly influences crop growth, with each crop having an optimal temperature range for optimal development.
Humidity	Humidity refers to the amount of moisture present in the air, influencing plant transpiration and moisture uptake.
pH	pH refers to the measurement of soil acidity or alkalinity, influencing nutrient availability and root health, crucial for optimal plant growth and yield.
Rainfall	Rainfall is a crucial factor, adequate rainfall ensures proper hydration and nutrient absorption, while drought conditions can lead to stunted growth and reduced agricultural productivity.
Crop Name	Crop Name refers to the specific type of plant or crop species recommended for cultivation based on various influencing factors.

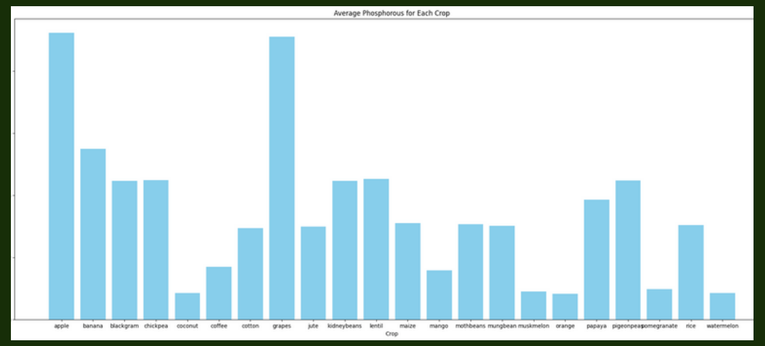
DATA VISUALIZATION

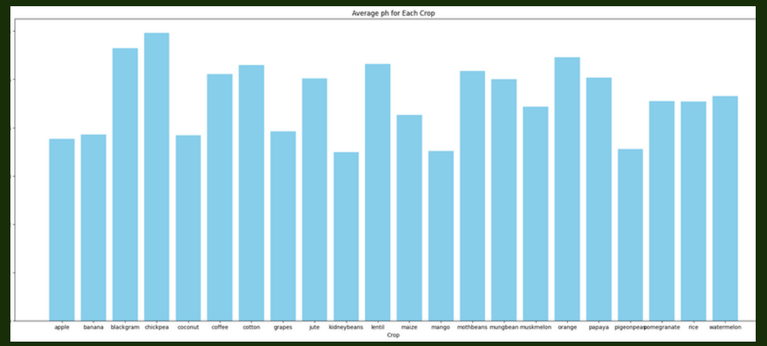
HISTOGRAMS

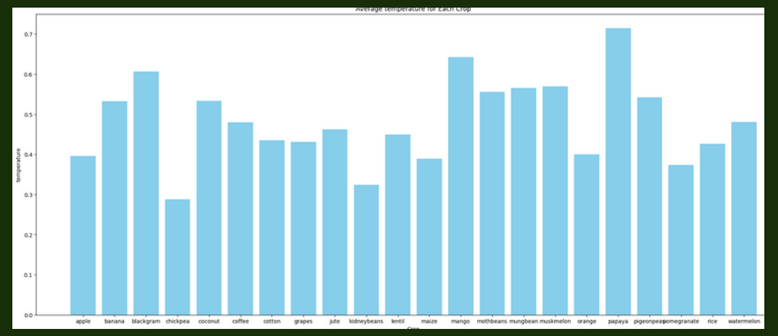






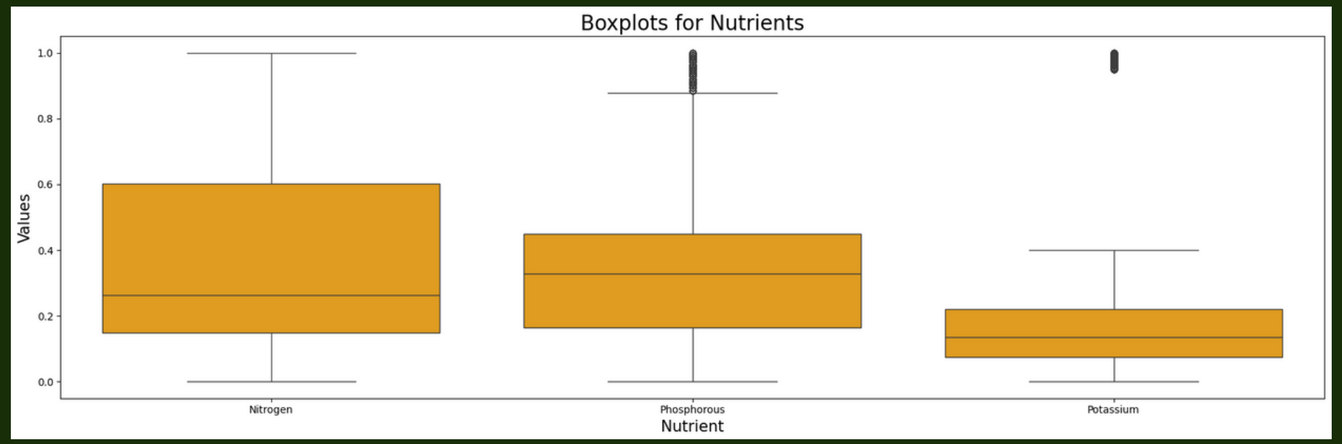


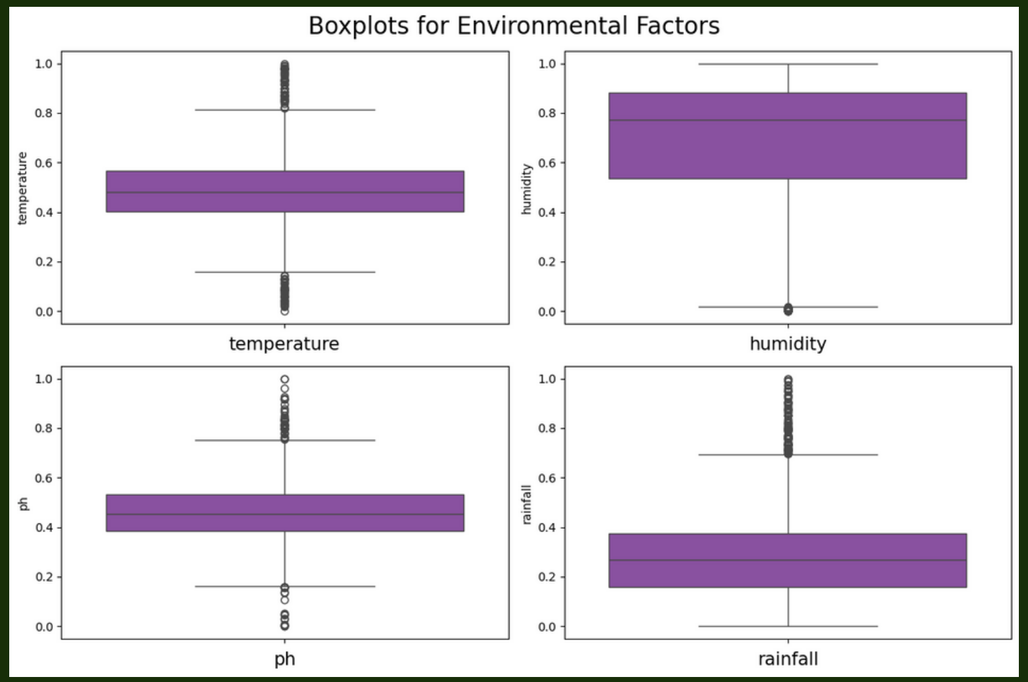




The Histogram effectively displays the average values of Nitrogen (N), Phosphorus (P), Potassium (K), Temperature, Humidity, pH, and rainfall corresponding to each crop. This visualization provides a clear representation of the distribution of these essential agricultural parameters across different crops. (Potassium is disregarded due to its lower distribution in the dataset.)

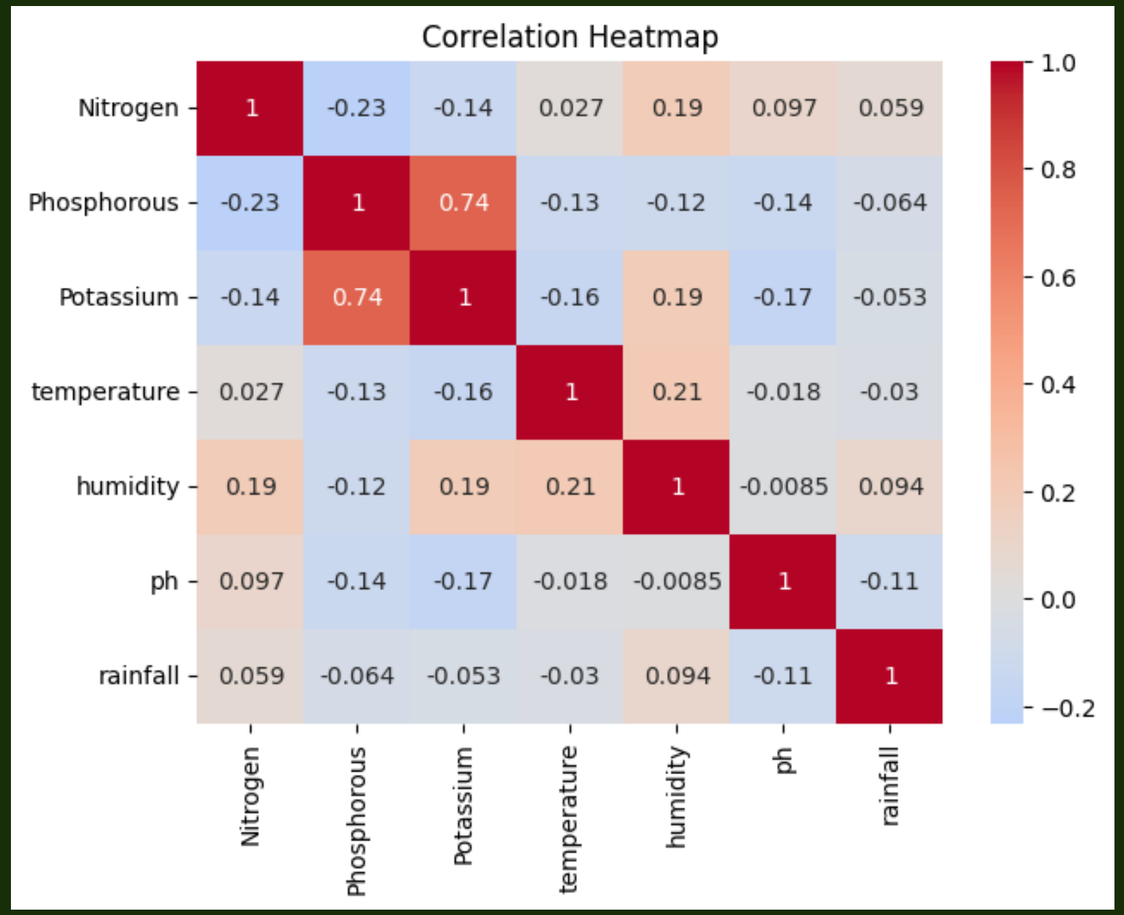
BOX PLOTS





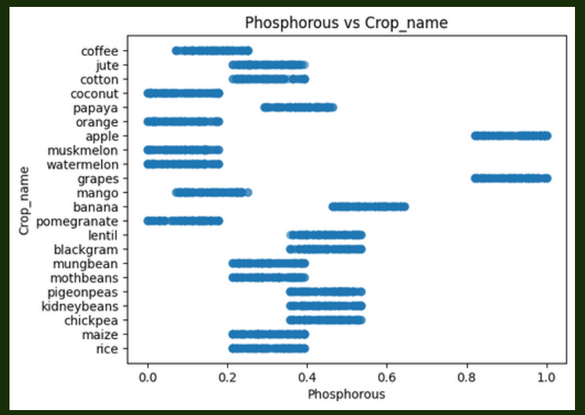
The box plots serve as a tool to visualize the distribution of data and point potential outliers. From the figures, it's evident that the box plot of nutrients exhibits fewer outliers compared to the box plots representing environmental factors.

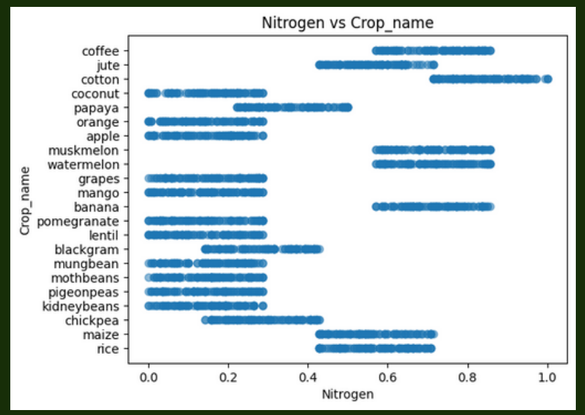
CORRELATION HEATMAP

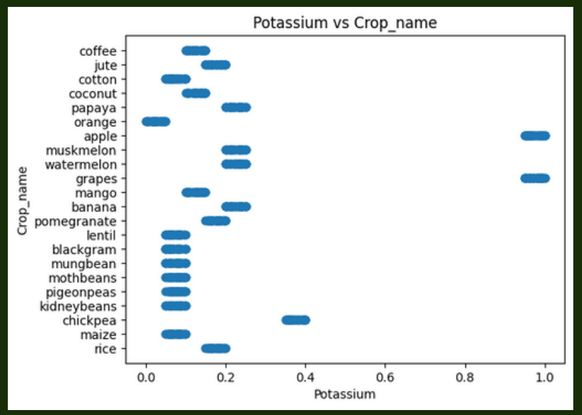


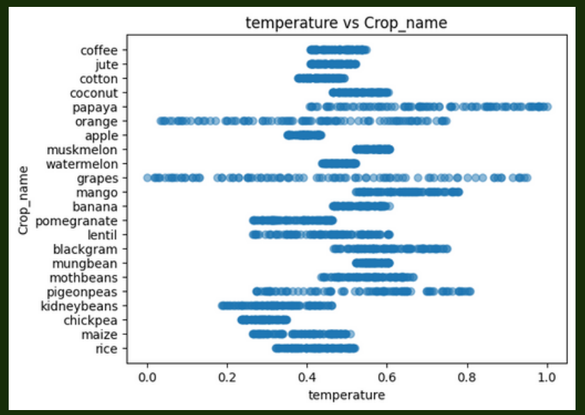
The correlation matrix depicts the relationship between variables, ranging from -1 to 1. A coefficient of 1 signifies perfect positive correlation, while -1 denotes perfect negative correlation, and 0 indicates no correlation. Notably, potassium and phosphorus exhibit a slight positive correlation, while phosphorus and nitrogen show a slight negative correlation.

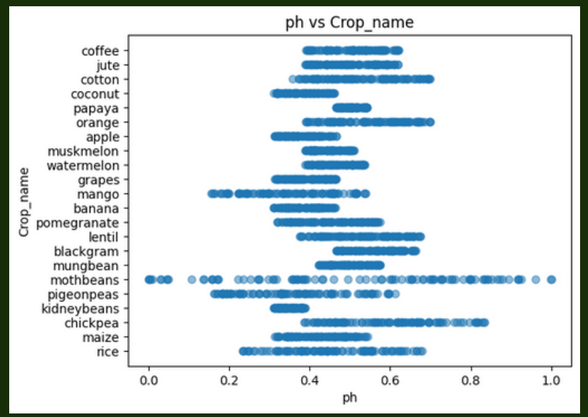
SCATTER PLOTS

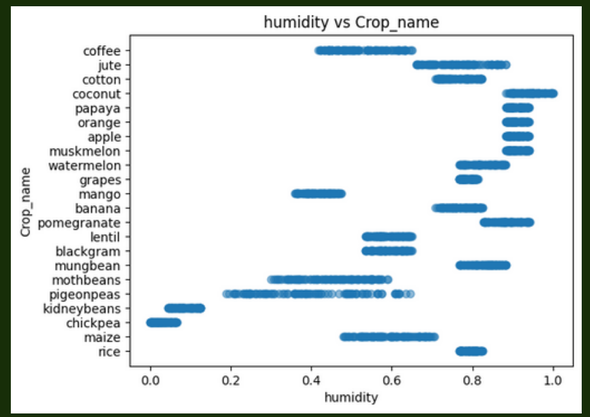


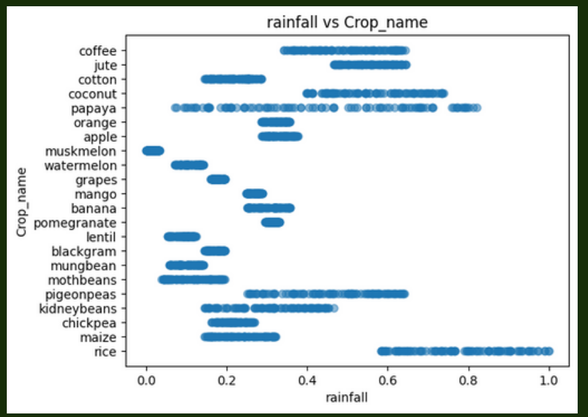










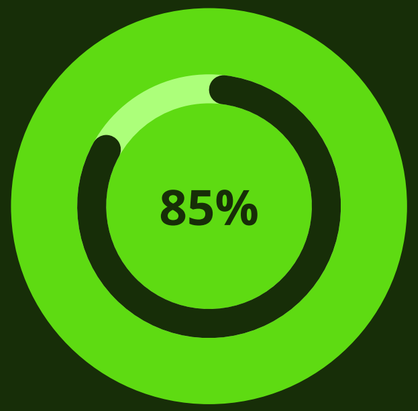


Scatter plots visually depict the relationship between two variables by plotting data points on a graph, facilitating the observation of patterns, trends, and potential correlations. Notably, the plots indicate that potassium has a weaker relationship with crop production compared to nitrogen, which exhibits a strong influence on crop yield

MULTI-COLLINEARITY

	variables	VIF
0	Nitrogen	3.077800
1	Phosphorous	7.344903
2	Potassium	4.811073
3	temperature	10.237423
4	humidity	9.666536
5	ph	9.564064
6	rainfall	3.107186

In our scenario, the VIF values for nitrogen, potassium, and rainfall are less than 5, indicating a low level of multicollinearity. Phosphorus, humidity, pH, and temperature have VIF values more than 5, but their impact is considered minor. VIF values less than 5 are generally considered good, indicating low levels of multicollinearity.



ALGORITHMS

ML ALGORITHMS

KNN	
Train Test Ratio	**Accuracy**
80-20	**97.7%**
70-30	97.4%
60-40	97.2%

Decision Tree	
Train Test Ratio	**Accuracy**
80-20	**90.9%**
70-30	69.8%
60-40	79.5%

Random Forest	
Train Test Ratio	**Accuracy**
80-20	**90.5%**
70-30	87.4%
60-40	87.3%

Naïve Bayes	
Train Test Ratio	**Accuracy**
80-20	**86.5%**
70-30	60%
60-40	53%

XG Boost	
Train Test Ratio	**Accuracy**
80-20	54.5%
70-30	82.7%
60-40	**83%**

ANN	
Train Test Ratio	**Accuracy**
80-20	83.5%
70-30	83%
60-40	**87.5%**

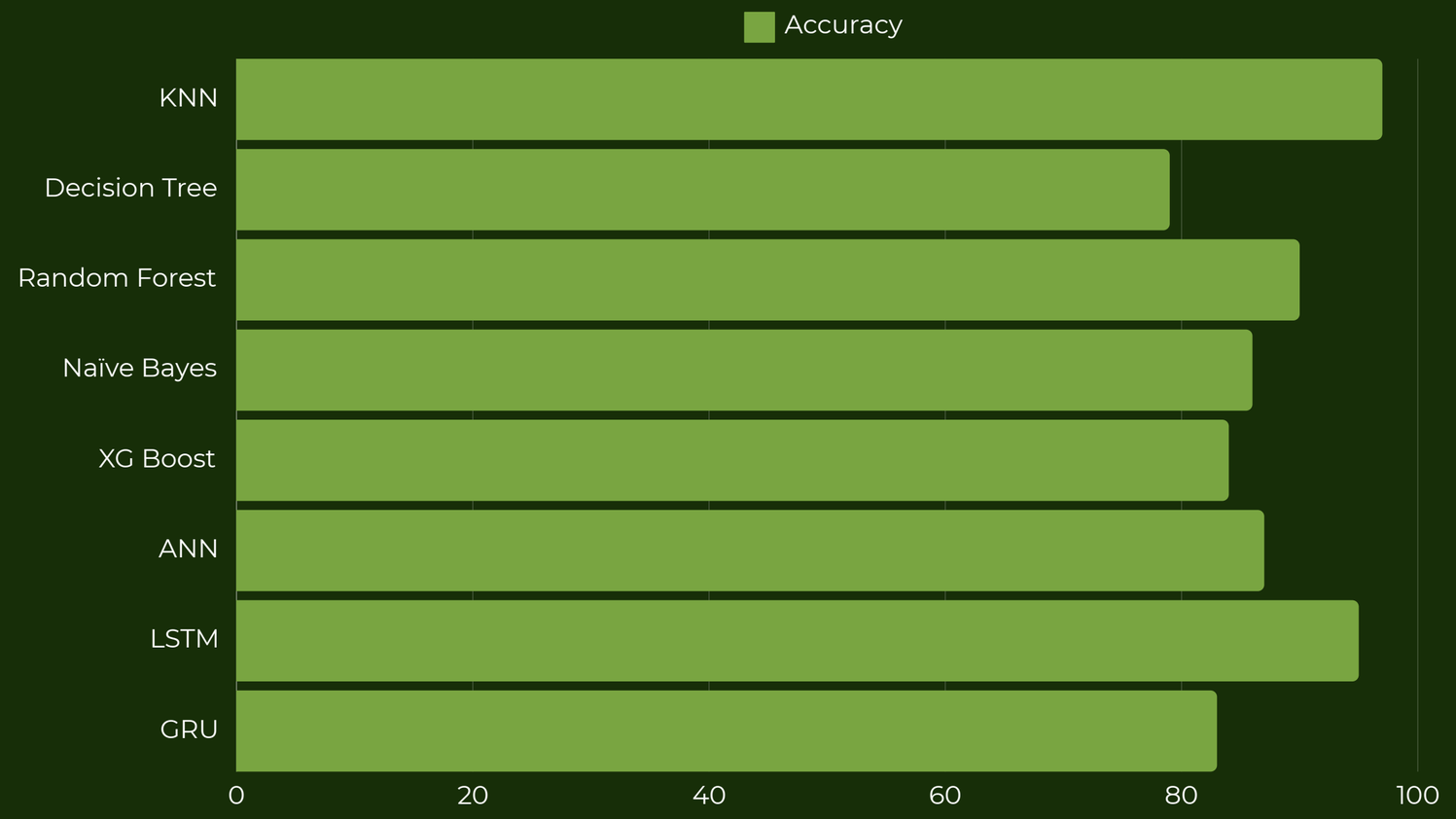
Optimizer Adam
Architecture 32-1
Activation function Relu, Sigmoid

DL ALGORITHMS

LSTM	
Train Test Ratio	**Accuracy**
80-20	**95.2%**
70-30	94.5%
60-40	92.7%

GRU	
Train Test Ratio	**Accuracy**
80-20	75.2%
70-30	**83.4%**
60-40	72%

COMPARING ML AND DL MODELS



CONCLUSION

- For the dataset, we performed six types of ML and two types of DL algorithms
- The visualization of data performed in exploratory data analysis showed clearly which of the features are more and less correlated with the target variable (crop_name).
- From all the analysis and implementations of the algorithm, we found the best results in KNN algorithm for 80-20 train-test ratio with accuracy of 97.7%.
- So, here we can conclude that KNN Algorithm can be considered the best model for the dataset.

LITERATURE REVIEW

Supervised Machine learning Approach for Crop Yield Prediction in Agriculture Sector”. In this proposed system crop yield prediction includes factors such as temperature, humidity, ph, rainfall, crop name. This crop prediction can be done by random forest algorithm and decision tree. By applying random forest algorithm got best accurate value result.[1] “**Crop recommendation system to maximize crop yield in ramtek region using machine learning”**. This proposed system worked on three parameters: soil characteristics, soil types and crop yield data collection based on these parameters suggesting the farmer suitable crop to be cultivated. This proposed systemworked on different machine learning algorithms like random forest, CHAID, K-Nearest Neighbour and Naïve Bayes. By applied this proposed system we can predict particular crop under particular weather condition, state and district values.[2] **“AgroConsultant: Intelligent Crop Recommendation System Using Machine Learning Algorithms**”. This proposed system can be divided into two sub-systems: i) crop suitable predictor ii) Rainfall Predictor. In this proposed system Implemented different machine learning algorithms like Decision Tree, K Nearest Neighbor (K-NN), Random Forest and Neural Network and performed multi-label classification on it. This proposed system achieved 71% accuracy by using rainfall predictor model and achieved 91.00% accuracy by applying neural network algorithm on crop suitable predictor system.[3]

REFRENCES

[1] Kumar, Y. Jeevan Nagendra, V. Spandana, V. S. Vaishnavi, K. Neha, and V. G. R. R. Devi. "Supervised Machine learning Approach for Crop Yield Prediction in Agriculture Sector". In 2020 5th International Conference on Communication and Electronics Systems (ICCES), pp. 736-741. IEEE, 2020.

[2] Reddy, D. Anantha, Bhagyashri Dadore, and Aarti Watekar. "Crop recommendation system to maximize crop yield in ramtek region using machine learning." International Journal of Scientific Research in Science and Technology 6, no. 1 (2019): 485-489.

[3] Doshi, Zeel, Subhash Nadkarni, Rashi Agrawal, and Neepa Shah. "AgroConsultant: Intelligent Crop Recommendation System Using Machine Learning Algorithms." In 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), pp. 1-6. IEEE, 2018.

THANK YOU

AGROADVISER

Presented By
K.Spandana

CODE

TRAINING AND TESTING

```
# Split the data into training and testing sets
y = data['Crop_name']
X = data.drop('Crop_name', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=40)
```

KNN

KNN

```
#KNN
from sklearn.neighbors import KNeighborsClassifier
# Create a KNN model with k=3 (you can adjust the 'n_neighbors' parameter)
knn_model = KNeighborsClassifier(n_neighbors=1)

# Train the KNN model
knn_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = knn_model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.3f}')
```

DECISION TREE

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
# Create a Decision Tree regression model
model = DecisionTreeClassifier(max_depth=6)
model.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=6)
```

```
[26] y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.3f}")
```

RANDOM FOREST

Random Forest

```
from sklearn.ensemble import RandomForestClassifier

# Initializing the Random Forest classifier
random_forest = RandomForestClassifier(max_depth=2,  # Limit tree depth
                                       n_estimators=200,  # Increase number of trees
                                       max_features="sqrt")

# Training the classifier
random_forest.fit(X_train, y_train)

# Making predictions on the test set
predictions = random_forest.predict(X_test)

# Calculating accuracy
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy:.3f}")
```

NAÏVE BAYES

```
#Naive Bayes
from sklearn.naive_bayes import GaussianNB


# Generate synthetic dataset with noise
X, y = make_classification(n_samples=1000, n_features=20, n_informative=10, n_clusters_per_class=2, random_state=42)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

# Create a Gaussian Naive Bayes model
nb_model = GaussianNB()

# Train the Naive Bayes model
nb_model.fit(X_train, y_train)

# Make predictions on the training set
y_train_pred = nb_model.predict(X_train)

# Make predictions on the test set
y_test_pred = nb_model.predict(X_test)

# Evaluate the model

accuracy = accuracy_score(y_test, y_test_pred)

print(f' Accuracy: {accuracy:.3f}')
```

XG BOOST

Gradient Boosting

```
[29] import xgboost as xgb
     # Split the dataset into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
     # Convert data into DMatrix format used by XGBoost
     dtrain = xgb.DMatrix(X_train, label=y_train)
     dtest = xgb.DMatrix(X_test, label=y_test)

     # Set hyperparameters for XGBoost
     params = {
         'objective': 'multi:softmax',
         'num_class': 3,
         'max_depth': 3,
         'learning_rate': 0.1,
         'n_estimators': 100,
         'eval_metric': 'mlogloss'
     }

     # Train the XGBoost model
     xgboost_model = xgb.train(params, dtrain, num_boost_round=10)

     # Make predictions on the test set
     y_pred = xgboost_model.predict(dtest)
```

ANN

```
#ANN
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
from sklearn.preprocessing import StandardScaler

# Generate a sample binary classification dataset
X, y = make_classification(n_samples=1000, n_features=20, n_classes=2, random_state=42)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create an ANN model
model = Sequential()
model.add(Dense(32, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(1, activation='sigmoid'))
```

LSTM

LSTM

```
from tensorflow.keras.layers import LSTM  # Import LSTM layer

# Define the LSTM model
model = Sequential()
model.add(LSTM(units=50, input_shape=(X_train_reshaped.shape[1], X_train_reshaped.shape[2])))
model.add(Dense(units=1, activation='sigmoid'))  # Assuming you want to predict one output with sigmoid activation
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])  # Adjust optimizer and loss function as needed

# Train the model
model.fit(X_train_reshaped, y_train, epochs=150, batch_size=32, validation_split=0.1)  # Adjust epochs and batch_size as needed

# Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test_reshaped, y_test)
print('Test Loss:', test_loss)
print('Test Accuracy:', test_accuracy)
```

GRU

GRU

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dense


# Reshape the input data to be 3-dimensional for GRU input
X_train_reshaped = X_train.reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test_reshaped = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))

# Define the GRU model
model = Sequential()
model.add(GRU(units=50, input_shape=(X_train_reshaped.shape[1], X_train_reshaped.shape[2])))
model.add(Dense(units=1, activation='sigmoid'))  # Assuming you want to predict one output with sigmoid activation
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])  # Adjust optimizer and loss function as needed

# Train the model
model.fit(X_train_reshaped, y_train, epochs=150, batch_size=32, validation_split=0.1)  # Adjust epochs and batch_size as needed

# Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test_reshaped, y_test)
print('Test Loss:', test_loss)
print('Test Accuracy:', test_accuracy)
```