# Intellihack NextGen

## Task 01

Group Name: Cryptonovate

## Contents

## Question 01

You've been provided with a dataset containing information about crops, including their nutrient levels (N, P, K), environmental factors (temperature, humidity, pH, rainfall), and corresponding labels. Your task is to create a predictive model that recommends the best three crops based on the provided conditions. Using machine learning techniques, build a model that takes into account the input features and predicts the most suitable crops for cultivation.

Find the data set here

Tasks:

1. Data Exploration and Preprocessing:

• Explore the dataset to understand its structure, features, and distributions.

• Perform any necessary preprocessing steps such as handling missing values, encoding categorical variables and scaling numerical features.

2. Model Training:

• Choose an appropriate machine learning algorithm (e.g., Decision Trees, Random Forests,

Support Vector Machines) for building the predictive model.

• Split the dataset into training and testing sets.

• Train the model using the training data.

3. Model Evaluation:

• Evaluate the trained model's accuracy in predicting the crop labels using the testing dataset.

• Provide insights into how well the model performs in suggesting appropriate crops based on

the given environmental conditions.

4. Joblib Model Creation and Prediction:

• Create a joblib model (.joblib) from the trained model.

• Use the created joblib model to make predictions on new environmental conditions.

• Calculate the accuracy of the predictions.

**Submission Requirements:**

1. Submit your Python code implementing the data preprocessing, model training, evaluation, and prediction.

2. Include a brief report discussing the approach taken, challenges faced, insights gained from model evaluation, and suggestions for improving the model's performance.

3. Provide instructions for running your code and reproducing the results.

Note: You may use any relevant libraries and techniques for this task, ensuring clear documentation and organization of your code.

## Code

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import joblib


data = pd.read_csv("Crop_Dataset.csv")


numerical_features = ['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall']
data = data[numerical_features + ["Label_Encoded"]]


X = data.drop(columns=["Label_Encoded"])
y = data["Label_Encoded"]


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,
stratify=y, random_state=42)


scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


model = SVC(kernel='linear', C=1.0, probability=True)
model.fit(X_train_scaled, y_train)


y_pred = model.predict(X_test_scaled)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)


joblib.dump(model, "crop_recommendation_model.joblib")

def predict_most_suitable_crops():
    N = float(input("Enter the value for 'N' (Nitrogen): "))
    P = float(input("Enter the value for 'P' (Phosphorus): "))
    K = float(input("Enter the value for 'K' (Potassium): "))
    temperature = float(input("Enter the value for 'temperature': "))
    humidity = float(input("Enter the value for 'humidity': "))
    ph = float(input("Enter the value for 'ph': "))
    rainfall = float(input("Enter the value for 'rainfall': "))

    input_data = pd.DataFrame([[N, P, K, temperature, humidity, ph, rainfall]],
columns=numerical_features)
    input_data_scaled = scaler.transform(input_data)
    probabilities = model.predict_proba(input_data_scaled)[0]  # Get
probabilities for each class
    probabilities_dict = {get_name(i): prob for i, prob in
enumerate(probabilities)}
    sorted_probabilities = sorted(probabilities_dict.items(), key=lambda x: x[1],
reverse=True)  # Sort probabilities

    # Select the top three crops
    top_three_crops = [crop for crop, prob in sorted_probabilities[:3]]
    return top_three_crops


def get_name(key):
    my_dict = {
        0: "wheat",
        1: "barley",
        2: "lettuce",
        3: "spinach",
        4: "cauliflower",
        5: "brussels_sprouts",
        6: "cabbage",
        7: "beans",
        8: "peas",
        9: "turnips",
        10: "carrots",
        11: "beets",
        12: "cherries",
        13: "plums",
        14: "raspberries",
        15: "pears",
        16: "blackcurrants",
        17: "strawberries",
        18: "apples",
        19: "potatoes",
        20: "rapeseed",
        21: "tomatoes"
    }
    return my_dict[key]


predicted_crops = predict_most_suitable_crops()
print("Top three predicted crops:", predicted_crops)
```

## Code Explanation

1. **Data Exploration and Preprocessing:**
    - Data Loading: The code starts by loading the dataset from the provided CSV file into a pandas DataFrame.
    - Feature Selection: It selects only the numerical features ('N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall') along with the encoded labels.
    - Splitting Data: The dataset is divided into input features (X) and target labels (y).
    - Train-Test Split: It splits the dataset into training and testing sets using 90% of the data for training and 10% for testing. Stratified splitting ensures that the proportion of classes is maintained in both sets.
    - Feature Scaling: The numerical features are scaled using StandardScaler to standardize them around zero mean and unit variance.
2. **Model Training:**
    - Model Selection: It selects Support Vector Machine (SVM) with a linear kernel as the predictive model.
    - Model Training: The SVM model is trained on the scaled training data (X_train_scaled, y_train).

3. **Model Evaluation:**

- Model Evaluation: It evaluates the trained SVM model's accuracy on the testing dataset (X_test_scaled, y_test).
- Accuracy Calculation: The accuracy score is calculated using the predicted labels (y_pred) and the actual labels (y_test).

4. **Joblib Model Creation and Prediction:**

- Model Serialization: The trained SVM model is serialized into a .joblib file named "crop_recommendation_model.joblib" using joblib.dump.

- Prediction Function: It defines a function predict_most_suitable_crops() to predict the top three most suitable crops based on user-input environmental conditions.

- Prediction Process:

    i. It takes user inputs for environmental conditions (N, P, K, temperature, humidity, ph, rainfall).
    ii. Creates a DataFrame with the input data.
    iii. Scales the input data using the same scaler used for training data.
    iv. Uses the trained SVM model to predict the probabilities of each crop.
    v. Selects the top three crops with the highest probabilities.

- Crop Mapping: It defines a helper function get_name() to map numerical labels to crop names for better readability.

- Prediction Output: It prints the top three predicted crops based on the provided environmental conditions.

Additionally, the code defines a prediction function that allows users to input environmental conditions and receive recommendations for the most suitable crops. This function utilizes the trained SVM model to predict the probabilities of each crop based on the provided inputs. The top three crops with the highest probabilities are then recommended to the user.

By combining data preprocessing, model training, evaluation, serialization, and prediction functionalities, the code provides a comprehensive solution for recommending crops based on environmental conditions. It demonstrates the end-to-end process of building and deploying a machine learning model for practical agricultural applications.

## Output

The output begins by indicating the dataset used for training and validation, which is the Crop_Dataset.csv. This dataset likely contains a comprehensive set of information about various crops, including their nutrient levels (N, P, K) and environmental factors such as temperature, humidity, pH, and rainfall.

Following this, the output displays the accuracy of the trained model, which is a crucial metric indicating the model's performance in predicting crop labels based on the provided environmental conditions. In the provided example, the accuracy score is 0.9909, signifying that the model achieves high accuracy in its predictions.

After showcasing the model's accuracy, the output prompts the user to input specific environmental conditions such as nutrient levels (N, P, K), temperature, humidity, pH, and rainfall. These inputs are vital as they represent the environmental conditions under consideration for crop cultivation.

Once the user inputs these environmental conditions, the model then processes this data by analyzing the dataset. It utilizes the trained machine learning algorithm to make predictions based on the user-provided inputs. The model considers the relationships between environmental factors and crop types learned during the training phase.

Finally, based on this analysis, the output provides the top three predicted crops that are most suitable for cultivation under the given environmental conditions. These predicted crops are determined by the model's probabilities, ranking them based on their likelihood of thriving in the provided conditions.

In the provided example, the user input environmental conditions such as Nitrogen (N) level of 80, Phosphorus (P) level of 40, Potassium (K) level of 35, temperature of 25°C, humidity of 83%, pH level of 7, and rainfall of 250 mm. Based on these inputs, the model predicts that the most suitable crops for cultivation are wheat, rapeseed, and apples. These recommendations are derived from the model's analysis of the dataset and its learned patterns regarding crop-environment relationships.

Overall, the output exemplifies a systematic approach to crop recommendation based on machine learning techniques, incorporating model accuracy assessment, user input interaction, and data analysis to provide actionable insights for crop cultivation decisions.

## Approach Taken

In this project, two machine learning models were trained and evaluated for crop recommendation: Support Vector Machine (SVM) and Decision Tree. Both models were trained on the same dataset containing information about crops and environmental conditions.

Support Vector Machine (SVM) is a powerful classification algorithm known for its effectiveness in handling complex datasets and high-dimensional feature spaces. SVM aims to find the optimal hyperplane that best separates different classes, maximizing the margin between them. In this context, SVM excelled in accurately predicting crop labels based on environmental factors, achieving an accuracy score of 0.986.

On the other hand, Decision Tree is a simple yet interpretable model that partitions the feature space into regions based on feature values, making decisions based on the majority class within each region. Despite its simplicity, Decision Trees can capture complex relationships in the data. In this case, the Decision Tree model achieved a slightly higher accuracy score of 0.991, indicating its effectiveness in this specific task.

While both models performed well, the Decision Tree model slightly outperformed SVM in terms of accuracy. However, the choice between these models may also depend on other factors such as interpretability, computational efficiency, and the specific requirements of the application.

## Challenges Faced

During the development of the crop recommendation model, several challenges were encountered, ranging from data preprocessing to version control issues. These challenges required careful consideration and problem-solving to ensure the successful implementation of the model.

One significant challenge arose during the data preprocessing stage. When preprocessing the dataset, it was necessary to determine which conditions or features could be ignored without compromising the model's predictive performance. This decision-making process involved analyzing the relevance of each feature to crop suitability and considering factors such as multicollinearity and feature importance. Determining the optimal subset of features required thorough exploration and experimentation to strike a balance between model complexity and performance.

Another challenge was identifying and handling missing data within the dataset. Missing data can significantly impact model training and prediction accuracy, necessitating careful strategies for imputation or removal of incomplete records. Devising robust methods for handling missing data involved assessing the extent of missingness, exploring patterns in missing values, and selecting appropriate imputation techniques to ensure data integrity and model robustness.

Additionally, the development process encountered challenges related to version control and collaboration using GitHub. Managing code changes, resolving conflicts, and coordinating contributions from multiple team members presented logistical hurdles that required effective communication and coordination. Ensuring consistency and reliability in version-controlled code repositories demanded adherence to best practices for branching, merging, and documenting changes, mitigating the risk of errors and inconsistencies in the codebase.

Overcoming these challenges required a combination of technical expertise, critical thinking, and effective collaboration. By addressing data preprocessing complexities, handling missing data effectively, and streamlining version control processes, the development team was able to overcome obstacles and deliver a reliable crop recommendation model. These challenges underscored the importance of meticulous attention to detail, robust problem-solving skills, and effective teamwork in the development of machine learning solutions.

## Suggestions for Improving The Model's Performance

1. Feature Engineering: Explore additional features or derive new features that may better capture relationships between environmental conditions and crop suitability. Feature engineering techniques such as polynomial features, interaction terms, or domain-specific knowledge incorporation can enhance the model's predictive power.

2. Hyperparameter Tuning: Experiment with different hyperparameter settings for the chosen machine learning algorithms. Techniques like grid search or random search can help identify optimal hyperparameter configurations that maximize model performance.

3. Ensemble Methods: Consider ensemble methods such as Random Forests or Gradient Boosting Machines, which combine multiple models to improve predictive accuracy. Ensemble methods often outperform individual models by leveraging the strengths of diverse models and mitigating their weaknesses.

4. Model Evaluation Metrics: Explore alternative evaluation metrics beyond accuracy, such as precision, recall, F1-score, or area under the ROC curve (AUC). Choosing appropriate evaluation metrics that align with the specific requirements and objectives of the crop recommendation task can provide deeper insights into model performance.