

Name : **Syed Mehdi Shah**

Ph.: **03488036361**

Intern ID : **ARCH-2505-0507**

Category : **C**

Organization : **ARCH -
Technologies**

Project Title :

Crop_Prediction

.....

Student Of **BS Computer Science**

Institution : **Hazara University**
Mansehra , KPK.

Contents

1. Project overview.....	page- 3
2. Diagrams & Graphs.....	page- 6
3. Project code.....	page -7
• HTML Code (index.HTML)	page- 8
• CSS Code (index.css)	page- 9
• Python Code (app.py)	page-10
• Python code (ml_model.py)	page- 11
• Google Collab Code (Model.ipynb)	page- 13
4. Theory Task	page- 15
• Questions Answers	
5. Notes & Reference	page- 28
6. YouTube video Demonstration	Page -30
• Link	
7. Conclusion.....	page - 31

1 – Overview

Helping Farmers Grow Smarter

Choosing the right crop is one of the biggest challenges farmers face weather, soil health, market demand, and other factors all play a role. To make this decision easier, we built **Smart Crop Predictor**, a machine learning-powered tool that analyses key farming conditions and recommends the best crops to plant.

See It in Action: Project Demo

Want to see how it works? Check out my [Project Demonstration Video on Youtube](#)

In the demo, we walk you through:

- ✓ How farmers can input their local conditions (soil type, rainfall, temperature, etc.).
- ✓ How the model suggests the best crop for their land.
- ✓ Additional useful details like fertilizer recommendations, seed types, market prices, and expected yield.

By the end, you'll see exactly how this tool can support farmers in making smarter, more profitable decisions.

Why This Matters

Farming isn't just about hard work—it's about making the right choices. A small miscalculation in crop selection can lead to losses. Our **Smart Crop**

Predictor uses machine learning to analyse past data and current conditions, giving farmers a clear recommendation on what to grow for the best results.

Key Features

- ❑ **Smart Crop Suggestions** – Recommends the best crop based on soil, weather, and market trends.
- ❑ **Farming Insights** – Provides fertilizer recommendations, seed types, and expected yield per acre.
- ❑ **Market Price Data** – Helps farmers understand demand and pricing for their crops.
- ☛ **Real-Time Weather Integration** – Adjusts predictions based on live weather updates.
- ❑ **Easy-to-Use Web App** – Simple interface for farmers to get quick recommendations.

How We Built It

- **Data Source:** I used the [Crop Recommendation Dataset](#) from Kaggle, which includes soil nutrients, climate data, and crop details.
 - **Tech Stack:**
 - **Python & Machine Learning** (scikit-learn) – For building the prediction model.
 - **Flask** – Powers the backend of our web app.
 - **HTML/CSS** – For a farmer-friendly frontend.

How to Use It

1. Enter your farm's details (soil type, temperature, rainfall, etc.).
2. Click "**Predict**" – Our model processes the data.
3. Get your **personalized crop recommendation**, along with fertilizer tips, market prices, and yield estimates.

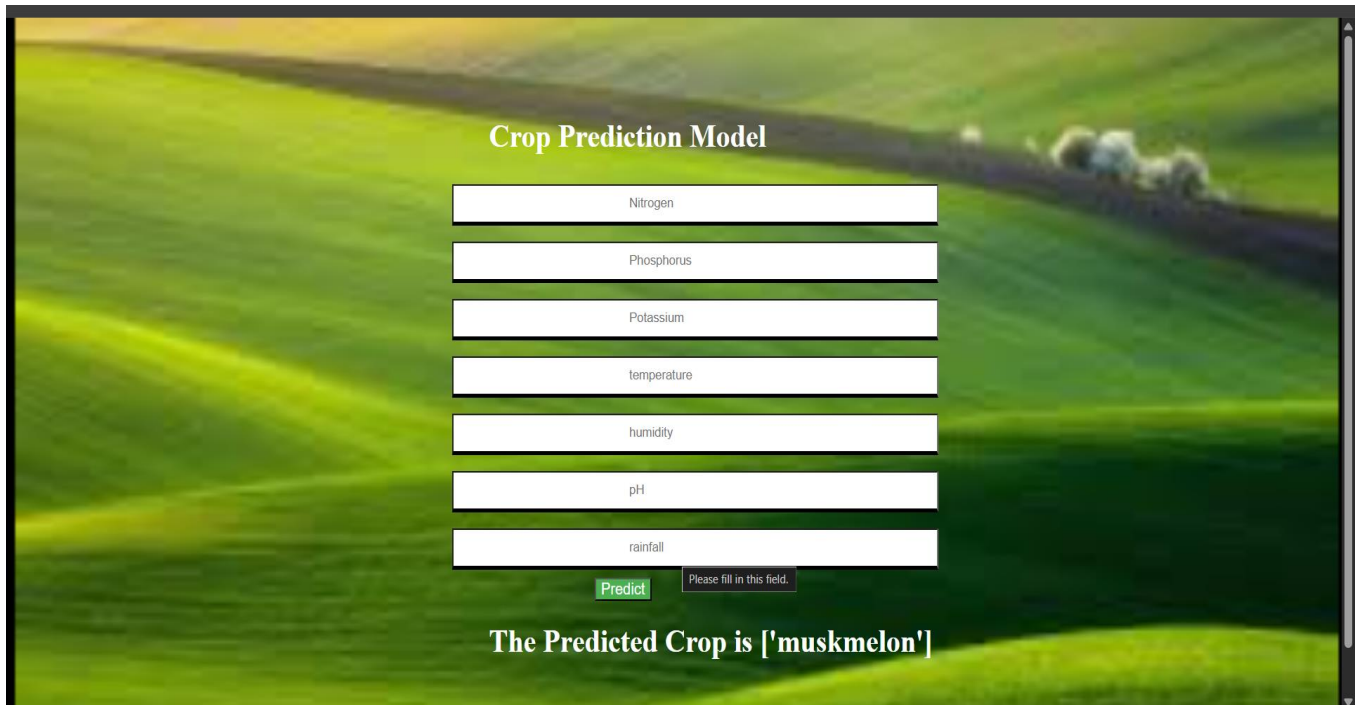
.....

2-Diagrams & Graphs

BackGround Picture



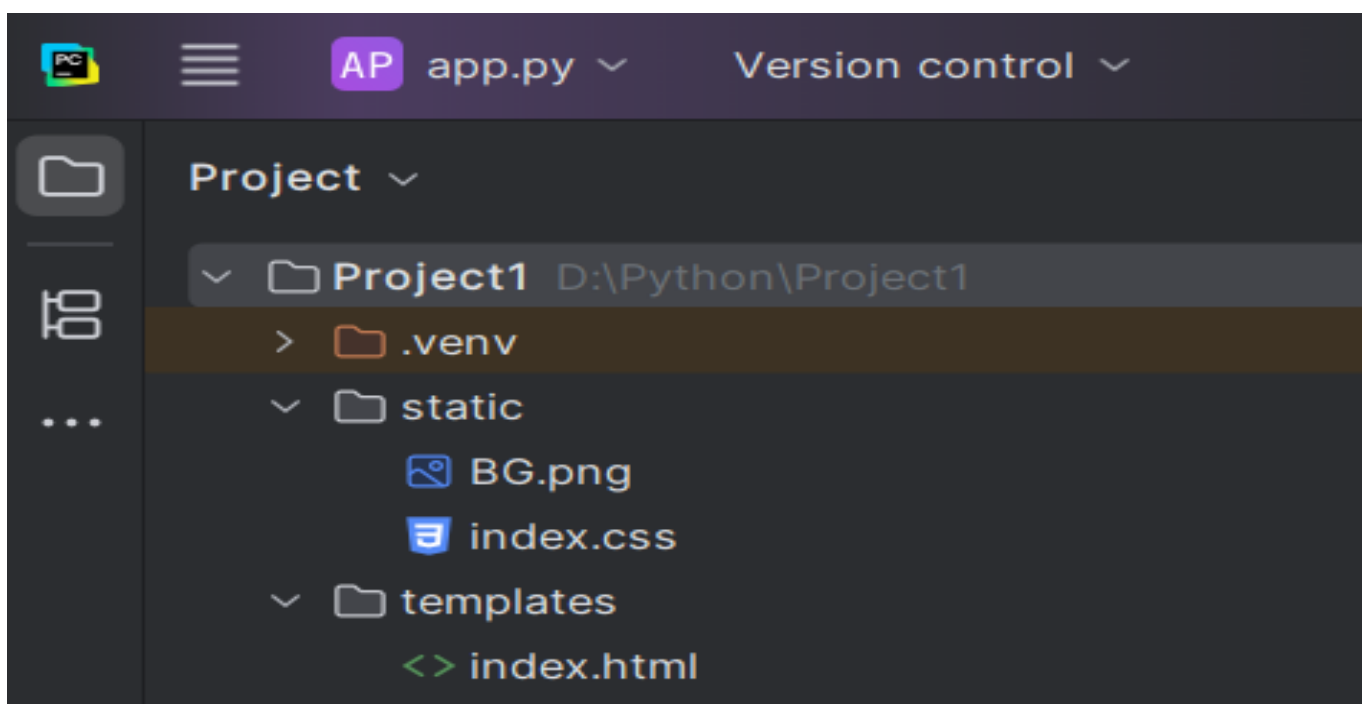
Crop Prediction Model Final Interface



The screenshot shows a web application titled "Crop Prediction Model" set against a background of rolling green hills. The interface includes seven input fields for "Nitrogen", "Phosphorus", "Potassium", "temperature", "humidity", "pH", and "rainfall". A green "Predict" button is located below these fields. A small red error message "Please fill in this field." is visible next to the "rainfall" input. At the bottom, a text display shows "The Predicted Crop is ['muskmelon']".

3-Project Codes

Folder-Project1



Project1 Folder

- > templates -> index.html (code)
- > static -> index.css (code)
& BG.png (image)
- > app.py (web app Code)
- > Crop_recommendation.csv (Excel File)
- > ml_model.py
- > model.ipynb (google colab code)
- > model.pkl

1-Html Code (index.html)

```
<!DOCTYPE html>
<html >

<head>
  <meta charset="UTF-8">
  <title>ML API</title>
  <link rel="stylesheet" href="../static/index.css">
</head>

<body>
  <div class="login">
    <h1>Crop Prediction Model</h1>

    <form action="{{ url_for('predict')}}"method="post">
      <input type="text" name="Nitrogen" placeholder="Nitrogen" required="required"
/>
      <input type="text" name="Phosphorus" placeholder="Phosphorus"
required="required" />
      <input type="text" name="Potassium" placeholder="Potassium" required="required"
/>
      <input type="text" name="temperature" placeholder="temperature"
required="required" />
      <input type="text" name="humidity" placeholder="humidity" required="required"
/>
      <input type="text" name="pH" placeholder="pH" required="required" />
      <input type="text" name="rainfall" placeholder="rainfall" required="required"
/>
      <br>
      <button type="submit" class="btn btn-primary btn-block btn-
large">Predict</button>
    </form>

    <h1 id="predi">
      {{ prediction_text }}
    </h1>

  </div>
```



```
</body>
</html>
```

.....

2-CSS Code (index.css)

```
body {
    background-image: url("BG.png");
    margin-top: 100px;
    margin-bottom: 100px;
    margin-left: 500px;
    margin-right: 150px;
}
body{
    background-repeat:no-repeat;
    background-position: center;
    background-attachment:fixed;
    background-size:cover;
}
h1 {
    margin-left: 50px;
    margin-right: 50px;
    color: white;
}
input{
    width: 150px;
    padding:10px 200px;
    margin :8px;
    border-bottom:4px solid black;
}
button{
    background-color: #4CAF50;
    margin : auto ;
    color:white;
    text-align :center;
    text-decoration: none;
    display:inline-block;
    font-size : 16px;
    margin-left:170px;
}
#predict{
    margin-left:10px;
```

```
margin-bottom:1px;
color:white;

}
```

.....

3-Python Code (app.py)

```
import numpy as np
from flask import Flask, request, render_template
import pickle

# Create flask app
flask_app = Flask(__name__)
model = pickle.load(open("model.pkl", "rb"))

@flask_app.route("/")
def Home():
    return render_template("index.html")

@flask_app.route("/predict", methods = ["POST"])
def predict():
    float_features = [float(x) for x in request.form.values()]
    features = [np.array(float_features)]
    prediction = model.predict(features)
    return render_template("index.html", prediction_text = "The Predicted Crop is {}".format(prediction))

if __name__ == "__main__":
    flask_app.run(debug=True)
```

.....

4.Python Code (ml_model.py)

```
# import pandas as pd
# from sklearn.preprocessing import StandardScaler
```

```

# from sklearn.ensemble import RandomForestClassifier
# from sklearn.model_selection import train_test_split
import pickle
# #
# # Load the csv file
# df = pd.read_csv("Crop_recommendation.csv")
# #
# print(df.head())
#
# X = df[["N", "P", "K", "temperature","humidity","ph","rainfall"]]
# y = df["label"]
#
# # Split the dataset into train and test
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=50)
#
# # Feature scaling
# sc = StandardScaler()
# X_train = sc.fit_transform(X_train)
# X_test= sc.transform(X_test)
#
# # Instantiate the model
# classifier = RandomForestClassifier()
#
# # Fit the model
# classifier.fit(X_train, y_train)
#
# # Make pickle file of our model
# pickle.dump(classifier, open("model.pkl", "wb"))

# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# Load the dataset
data = pd.read_csv('Crop_recommendation.csv') # Replace 'crop_data.csv' with your
dataset file

# Split the data into features and labels
X = data.iloc[:, :-1] # Features
y = data.iloc[:, -1] # Labels

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create the model
model = RandomForestClassifier()

# Train the model

```

```

model.fit(X_train, y_train)

# Make predictions on test data
# predictions = model.predict(X_test)

pickle.dump(model, open("model.pkl", "wb"))
# Evaluate the model
# accuracy = model.score(X_test, y_test)
# print("Accuracy:", accuracy)

# Example usage: Predict crop for a new set of features
# new_features = [[117 ,32,34,26.2724184,52.12739421,6.758792552,127.1752928,]] #
Replace with your own set of features
# predicted_crop = model.predict(new_features)
# print("Predicted crop:", predicted_crop)

```

5-Google Collab Code (Model.ipynb)

```

1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.ensemble import RandomForestClassifier
✓ [18] 13ms

```

```

1 data = pd.read_csv("Crop_recommendation.csv") #
✓ [19] 19ms

```

```

1 data.head()
✓ [20] 40ms

```

5 rows x 8 cols

Edit in Data Wrangler

	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

```

1 data.shape
[4]
(2200, 8)

```

1 data.isnull().sum()

✓ [25] 19ms

8 rows 8 rows x 1 cols

	<unnamed>
N	0
P	0
K	0
temperature	0
humidity	0
ph	0
rainfall	0
label	0

1 # split feature and labels

✓ [26] < 10 ms

1 x = data.iloc[:, :-1] # feature

2 y = data.iloc[:, -1] # labels

✓ [27] 11ms

1 X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 42)

✓ [28] 24ms

1 X_train.head()

✓ [29] 21ms

5 rows 5 rows x 7 cols

Edit in Data Wrangler

	N	P	K	temperature	humidity	ph	rainfall
1656	17	16	14	16.396243	92.181519	6.625539	102.944161
752	37	79	19	27.543848	69.347863	7.143943	69.408782
892	7	73	25	27.521856	63.132153	7.288057	45.208411
1041	101	70	48	25.360592	75.031933	6.012697	116.553145
1179	0	17	30	35.474783	47.972305	6.279134	97.790725

1 y_train.head()

✓ [30] 10ms

5 rows 5 rows x 1 cols

	label
1656	orange
752	blackgram
892	lentil
1041	banana
1179	mango

```
1 model = RandomForestClassifier()
✓ [31] 10ms

1 model.fit(X_train, y_train)
✓ [32] 942ms

RandomForestClassifier
RandomForestClassifier()

1 predictions = model.predict(X_test)
✓ [33] 38ms

1 accuracy = model.score(X_test, y_test)
✓ [34] 36ms

1 print("Accuracy:", accuracy)
✓ [35] < 10 ms
Accuracy: 0.9931818181818182

1 new_features = [[36.58, 25.28, 66024, 59, 8.399136, 36.9263]]
2 predicted_crop = model.predict(new_features)
3 print("predicted crop:", predicted_crop)
✓ [36] 49ms
predicted crop: ['mothbeans']

C:\Users\Bismillah Traders\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
```

4. Theory Tasks

Q1) How would you define Machine Learning?

Ans: Machine Learning is like teaching a robot or computer to learn things from examples instead of giving it direct instructions. It's like how you would learn to ride a bike by practicing instead of reading a step-by-step manual.

The field of study that gives computers the ability to learn without being explicitly programmed is called Machine Learning. (Arthur Samuel, 1959)

Example :

For example, your spam filter is a Machine Learning program that can learn to flag

spam given examples of spam emails (e.g., flagged by users) and examples of regular

(Non spam, also called “ham”) emails.

Q2) Can you name four types of problems where it shines?

Ans: Here are **four types of problems** where it excels:

1) Classification

- What it does : Assigns a category or label to each input.

Example:

> is this email spam or not spam?

> what type of animal is this photo: dog, cat, or bird?

Real-life use cases:

- > Medical diagnosis (e.g., detecting diseases from scans).
- > fraud detection (e.g., flagging suspicious transactions).

2) Regression

- What it does: Predict a continuous value or number.

Example:

> What will the temperature be tomorrow?

> How much will this house sell for?

Real-life use cases:

- > stock price prediction.
- > Estimating demand for production or services.

3) Clustering Problems

Example: Grouping customers based on their shopping habits.

Real-world: Market segmentation, organizing photos by similar content or recommending product.

4) Reinforcement Learning Problem

Example: Teaching a robot to play chess by rewarding it for winning moves.

Real-world: Self-driving cars, game AI (like AlphaGo) or optimizing delivery routes.

Q3) What is labeled training set?

Ans: A labeled training set is a collection of examples used to teach machine learning model.

1) **Input Data:** The information model from.




> **Example:** A photo of a cat .

2) **Lable (Output) :** The correct answer or category association with the input.

> **Example:** The label for the photo would be “Cat”.

Example:

If you’re training a model to recognize animals:

Image	Label
 Dog photo	Dog
 Cat photo	Cat
 Bird Photo	Bird

> Here the image are the inputs, and the animal names are the labels.
Together, they form a labelled training set!

Q4) What are the two most common supervised tasks?

Ans: The two most common supervised learning tasks are ?

1) **Classification**

- What it does : Assigns a category or label to each input.

Example:

- > is this email spam or not spam?
- > what type of animal is this photo: dog, cat, or bird?

Real-life use cases:

- > Medical diagnosis (e.g., detecting diseases from scans).
- > fraud detection (e.g., flagging suspicious transactions).

2) Regression

- What it does: Predict a continuous value or number.

Example:

- > What will the temperature be tomorrow?
- > How much will this house sell for?

Real-life use cases:

- > stock price prediction.
- > Estimating demand for production or services.

Q5) Can you name four common Unsupervised tasks?

Ans: Here are four common tasks in unsupervised learning

- 1) Clustering
- 2) Dimensionality Reduction
- 3) Anomaly Detection
- 4) Association Rule Learning

Q6) What type of Machine Learning algorithm would you use to allow a robot to walk in various unknown terrains?

Ans:

To allow a robot to walk in various unknown terrains, we would use a
Reinforcement Learning (RL) Algorithm

7. What type of algorithm would you use to segment your customers into multiple Groups?

Ans:

To segment customers into multiple groups, we would use a **Clustering Algorithm** (which is Type of Unsupervised Learning)

8. Would you frame the problem of spam detection as a supervised learning problem or an unsupervised learning problem?

Ans:

The problems of spam detection is best framed as a **supervised learning Algorithm**

9. What is an online learning system?

Ans:

An online learning system is a type of machine learning approach where the module learns incrementally, one data point or a small batch of data at a time , instead of processing the entire dataset at once.

Key Characteristics:

- 1) Increment Updates
- 2) Adaptability
- 3) Real time Learning

10. What is out-of-core learning?

Ans:

Out-of –core learning refers to a type of machine learning that allows a model to train on data too large to fit into memory all at once. Instead of loading the entire data set into Memory, the algorithm processes the data in smaller batches or chunks. This approach is especially useful for big data scenarios where

traditional machine learning technique would be impractical due to memory limitation.

Key Characteristics:

- 1) Incremental Learning
- 2) batch processing
- 3) Memory Efficiency

11. What type of learning algorithm relies on a similarity measure to make predictions?

Ans:

The type of learning algorithm that relies on a similarity measure to make prediction called an **instance-based learning algorithm**. The most well

Example of this type of algorithm is K-Nearest Neighbors (K-NN)

Key characteristics:

- 1) Similarity Measure
- 2) No Explicit Model
- 3) Decision rule

12. What is the difference between a model parameter and a learning algorithm's Hyperparameter?

Ans:

The difference between a **model parameter** and a learning algorithm's

Hyperparameter lies in their roles and how they are determined

1) Model Parameter

2) Hyperparameter

<ul style="list-style-type: none"> • <u>Definition</u> >These are the internal variables of a machine learning model that are learned or adjusted during training 	<ul style="list-style-type: none"> • <u>Definition</u> >These are settings or configurations that you must specify before training the model; they govern how the training process operates and influence the model's performance.
<ul style="list-style-type: none"> • <u>Key Characteristics</u> 	<ul style="list-style-type: none"> • <u>Key Characteristics</u>

> Determined automatically by the training process. > Represent what the model has learned from the data	> set manually or optimized through techniques like grid search or random search > Not learned from the training data directly.
• <u>Example</u> > In linear regression: The weight and bias. > in neural networks: The weights of connection between neurons	• <u>Example</u> ≥ Number of hidden layers or units in a neural network. > The value of k in k-nearest neighbors(KNN)

13. What do model-based learning algorithms search for? What is the most common strategy they use to succeed? How do they make predictions?

Ans:

Strategy they use to succeed: how do they make predictions?

- Model-based learning algorithms aim to find an **optimal function (or model)** that maps input features (XXX) to outputs (yyy). This function should generalize well to new, unseen data.

Most Common Strategy:

- The most common strategy is **minimizing a cost/loss function** during training.

Cost/Loss Function:

1. A mathematical measure of how far the model's predictions are from the actual target values.
2. Examples:
 - a. Mean Squared Error (MSE) for regression.
 - b. Cross-Entropy Loss for classification.

Optimization Algorithms:

1. The model iteratively adjusts its parameters (e.g., weights and biases) to minimize this loss.
2. Common optimization methods:
 - a. Gradient Descent: Updates parameters by computing the gradient of the loss function and taking steps in the opposite direction.
 - b. Adam Optimizer: A more advanced method for faster convergence.

How Do They Make Predictions?

- Once trained, the model uses the learned function to make predictions on new data.
1. Input Features: The model takes unseen input data.
 2. Apply the Learned Function: The input is passed through the trained model (e.g., applying weights in a neural network or solving the regression equation).
 3. Output Predictions: The model produces predictions for the target variable (yyy).

14. Can you name four of the main challenges in Machine Learning?

Ans

Here are 4 main challenges in Machine Learning

- 1) **Insufficient or Poor-Quality Data**
- 2) **Overfitting and Underfitting**
- 3) **Feature Engineering and Selection**
- 4) **Scalability and Computational Resources**

15. If your model performs great on the training data but generalizes poorly to new instances, what is happening? Can you name three possible solutions?

Ans:

If your model performs well on the training data but generalizes poorly to new instances, it is likely experiencing overfitting. Overfitting occurs when the model learns not only the underlying patterns in the data but also the noise or irrelevant details, making it too specific to the training data.

Three Possible Solutions to Overfitting:

- 1. Regularization:**
- 2. Use Simpler Models:**
- 3. Increase Training Data or Use Data Augmentation:**

16. What is a test set and why would you want to use it?

Ans:

A test set is a subset of your dataset that is used to evaluate the final performance of your machine learning model after it has been trained on the training set and validated on the validation set (if used). The test set contains data that the model has never seen during training or hyperparameter tuning.

Why Use a Test Set?

- 1. Assess Generalization:**
 - a. The test set evaluates how well the model generalizes to new, unseen data.
 - b. This simulates the real-world scenario where the model is applied to new inputs.
- 2. Prevent Overfitting to Validation Data:**
 - a. If the model is evaluated solely on the validation set, it might overfit to it during hyperparameter tuning.
 - b. A separate test set ensures an unbiased performance estimate.
- 3. Final Check of Model Quality:**
 - a. Before deploying the model, the test set provides a clear picture of how it will perform in production.

Analogy:

- Think of a test set as a final exam:
 - The training set is your study material.
 - The validation set is your practice test.
 - The test set is the actual exam that measures how well you truly understood the subject.

- Proper use of a test set ensures your model is ready for real-world applications!

17. What is the purpose of a validation set?

Ans:

A validation set is a subset of your dataset used to evaluate the model's performance during training, specifically for tuning its hyperparameters and making decisions about the model's architecture or complexity. Its primary purpose is to help select the best-performing model without overfitting to the training data.

Key Purposes of a Validation Set:

1. Hyperparameter Tuning:

- a. Helps adjust settings like learning rate, number of layers, regularization strength, or the number of estimators in ensemble methods.
- b. Example: Choosing the best value for k in K-Nearest Neighbors (KNN).

2. Model Selection:

- a. Helps compare multiple models to pick the one that generalizes best to unseen data.
- b. Example: Comparing a Random Forest and a Support Vector Machine (SVM) using validation metrics.

3. Prevent Overfitting:

- a. Acts as a checkpoint to monitor the model's generalization ability during training.
- b. Implements early stopping if validation performance deteriorates.

4. Evaluating Intermediate Performance:

- a. Ensures the model learns meaningful patterns rather than memorizing training data.
- b. Provides feedback on how well the model performs on data it hasn't seen during training.

18. What can go wrong if you tune hyperparameters using the test set?

Ans:

Using the test set to tune hyperparameters defeats its primary purpose as a final, unbiased evaluation of your model's performance.

Here's what can go wrong:

1. Overfitting to the Test Set

- a. **What Happens:** Repeatedly tweaking hyperparameters based on test set performance tailors the model to that specific data.
- b. **Consequence:** The model performs well on the test set but poorly on new data, failing to generalize.

2. Unrealistic Performance Estimates

- a. **What Happens:** The test set, meant to simulate real-world data, becomes part of tuning, inflating performance metrics.
- b. **Consequence:** Deployed models may underperform in production due to misleading optimism.

3. Loss of True Validation

- a. **What Happens:** Without an independent test set, you cannot verify the model's true generalization ability.
- b. **Consequence:** No way to distinguish between genuine robustness and lucky performance on specific test examples.

Analogy:

- Tuning hyperparameters on the test set is like "peeking at final exam answers during practice"—it invalidates the evaluation.
- **Solution:** Always keep the test set untouched until the final evaluation to ensure reliable model deployment.

19. What is repeated cross-validation and why would you prefer it to using a single validation set?

Ans:

Repeated Cross-Validation is an enhanced validation technique that improves the reliability of model evaluation by performing multiple rounds of k-fold cross-validation with different random splits.

Key Steps:

1. Split the Data:

Randomly divide the dataset into k folds (e.g., $k=5$ or 10).

2. Perform k-Fold Cross-Validation:

Train the model k times, each time using a different fold as the validation set and the remaining folds for training.

3. Repeat:

Randomly resplit the data into k folds and repeat the process multiple times (e.g., 5 or 10 repetitions).

4. Aggregate Results:

Average performance metrics (e.g., accuracy, MSE) across all repetitions to obtain a robust estimate.

Why prefer Repeat Cross-Validation Over a Single Validation Set?

1) Reduces Variance in Estimates:

- A single validation set, or one round of cross-validation might produce results influenced by a lucky or unlucky split of the data.
- Repeated cross-validation smooths out these effects by averaging results over multiple splits.

2) Improves Generalization Insights:

- Repeated splits ensure that all data points are tested multiple times in different training/validation splits.
- This provides a more comprehensive evaluation of how well the model generalizes to unseen data.

3) Prevents Overfitting to a Specific Validation Set:

- Using a single validation set risks the model being tuned to perform well on that specific data.

- Repeated cross-validation uses multiple validation sets, reducing this risk.

4) More Reliable Model Comparison:

- When comparing different models, repeated cross-validation provides more consistent and reliable performance estimates.

Analogy:

- Using repeated cross-validation is like taking multiple mock exams with different sets of questions before the final test. You get a more reliable sense of how well you understand the material compared to practicing with just one mock exam.
- **When to Use It:**
 - Small datasets where every data point matters.
 - Comparing multiple models to ensure fair evaluation.
 - Situations where performance estimates must be robust and trustworthy.
- Repeated cross-validation ensures that your model is evaluated fairly and that performance metrics are as accurate as possible!

5 - Note and References

- **Python tutorial**

I learn python basic from YouTube tutorial.

[Python Tutorial - Apna College](#)

Key references from the tutorial:

- 1. Variables and Data Types
- 2. Basic Operations

- 3. Input and Output
 - 4. Conditional Statements
 - 5. Loops
 - 6. Functions
 - 7. String Methods
 - 8. File Handling
-
- **Python Installation**
 - > Python
 - > PyScripter
 - > PyCharm
-
- **Google colab**
 - [Google colab tutorial-Doga Ozgon](#)
 - > What is Google Colab
 - > How it Work
 - > How to write Code
 - > How to run
-
- **Web Development**
 - > I already learn the Basic Frontend Web Development from the
 - F ollowing YouTube tutorials

> HTML Tutorial from

[HTML Tutorial For Beginners - Apna College](#)

> CSS Tutorial from

[CSS Tutorial for Begginers - Apna College](#)

- I used the [Crop Recommendation Dataset](#) from Kaggle

- **Flask Web Application – Machine Learning Project**

This crop prediction web application was developed based on tutorial content from the following YouTube playlist:

[Machines Learning Project - Crop Prediction](#)

Key references from the playlist:

- 1- **Part 1 | Machine Learning Project | Flask Web Application**
- 2- **Part 2 | Demo Video | Machine Learning Model**
- 3- **Part 3 | Model Building | Crop Prediction | Machine Learning Model**
- 4- **Part 4 | Web Page Layout | Machine Learning Mode**
- 5- **Part 5 | CSS in Web Page | Machine Learning Model**
- 6- **Part 6 | Deploy Machine Learning Model on Flask | ML Model**

.....

6-YouTube Video Demonstration

[My Project Demo on YouTube video - Crop Prediction](#)

The video demonstrates a Flask-based crop prediction web application developed in PyCharm using Python for the backend, HTML/CSS for the

frontend interface, and a machine learning model trained in Google Colab. The application allows users to input soil nutrient values (Nitrogen, Phosphorus, Potassium) and pH levels through a web form, then displays the recommended crop prediction. The HTML provides the structure while CSS styles the responsive interface, with Flask handling the backend logic and model integration. The pre-trained machine learning model (saved as crop_model.pkl) processes the user inputs to generate predictions. The demonstration shows the complete workflow from code implementation in PyCharm to live testing in Chrome browser. The video highlights key components including the Flask routes, template rendering, form handling, and prediction display. This project effectively combines web development with machine learning to create a practical agricultural tool. The presentation aims to clearly showcase both the technical implementation and functional aspects of the application. The working prototype demonstrates how such solutions can be developed using accessible technologies.

.....

7-Conclusion

(I am **SYED MEHDI SHAH** undergraduate student of **BS Computer Science** at **HAZARA UNIVERSITY**, currently enrolled in 4th semester. According to category selection I am in Category: B, but I choose **Category-C** because it is easy and cannot impact on my Academic study. Also, my exam is coming, and I do not have enough time for Category-B, it is too lengthy and time consuming. Due to the following reason, I chose Category-C instead of Category-B.

I hope you accept my work / task and give me more opportunity to learn and work on more projects)

Thanks

Regard

Syed Mehdi Shah

.....

THE END