(name of your school)

A I P R O J E C T R E P O R T

on

“A HELPLINE BOT FOR FARMERS”

Natural Language Processing · SDG 2: Zero Hunger · SDG 1: No Poverty

Class X

Artificial Intelligence — Subject Code 417
Part C: Practical and Project Work

Session _____

Submitted by:

Name _____

Roll No. _____

Under the guidance of:

_____

Teacher (Artificial Intelligence)

CERTIFICATE

This is to certify that _____ of Class X, Roll No. _____, has satisfactorily completed the Artificial Intelligence project entitled **"A Helpline Bot for Farmers"** during the academic session _____, under my guidance and supervision.

The work has been carried out as the Project Work of Part C of Artificial Intelligence (Subject Code 417), in accordance with the curriculum prescribed by the Central Board of Secondary Education.

_____ _____

Teacher-in-charge External Examiner

_____

Principal

Date: _____

ACKNOWLEDGEMENT

I would like to thank _____, my Artificial Intelligence teacher, for the guidance and encouragement given throughout this project.

I am grateful to the Principal of _____ for the computer lab and the facilities that made this work possible.

My thanks go to _____, who helped me understand the problem this project tries to solve and how it is handled today.

I thank my classmates, who tried the model and told me where it went wrong, and my family for their support and patience while the project was built.

_____

(Candidate)

INDEX

SL. NO.	TOPIC	PAGE NO.
1	Introduction	5
2	Problem Scoping	5
3	Data Acquisition	6
4	Data Exploration	7
5	Modelling	9
6	Evaluation	10
7	Deployment	10
8	Ethical Concerns	11
9	The No-Code Version	12
10	Technical Details	12
11	Source Code	13
12	Sample Output	18
13	Testing	19
14	Results and Future Scope	20
15	Conclusion	21
16	References	21

> Page numbers refer to the pages of this report as printed. The cover page, certificate, acknowledgement and this index are not numbered.

INTRODUCTION

Agriculture helplines get thousands of calls, and a large share are the same few questions: how do I get my soil tested, how do I insure my crop, where can I see mandi prices, how do I check my PM-KISAN payment.

Farmers ask these questions in their own words, often mixing English and Hindi typed in English letters: "mitti ki jaanch kahan hogi", "fasal bima kaise karein". A simple menu of options does not work well for them.

This project builds a chatbot that reads a question, finds the closest of 12 common questions using TFIDF from Unit 6, and replies with its answer. Just as importantly, it says "I am not sure" when the question is not one it knows.

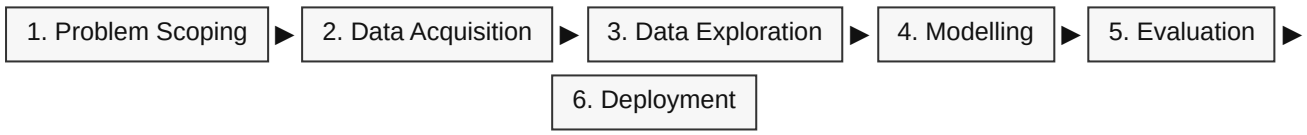
Who would use it: Farmers with a basic mobile phone, and the staff of agriculture helplines who answer the same questions many times a day.

Link to the Sustainable Development Goals

SDG 2 — Zero Hunger; SDG 1 — No Poverty. Small farmers lose money when they miss a scheme, sell below the market price or act too late on a pest. Much of the help they need already exists — the Soil Health Card, crop insurance, the Kisan Credit Card, mandi prices online — but finding it is hard. A helpline bot that answers common questions at any hour, in the farmer's own words, helps farmers earn more and grow more: SDG 1 and SDG 2.

The AI Project Cycle

This is a **Natural Language Processing** project. It follows the six stages of the AI Project Cycle, and the report that follows takes them in order, one section each.



PROBLEM SCOPING

The problem was scoped with the **4Ws Problem Canvas**: who has the problem, what it is, where it happens, and why solving it matters.

Who? STAKEHOLDERS • Small farmers with a mobile phone • Helpline staff who answer the same questions all day	**What?** THE PROBLEM • Common questions take up most of the helpline's time • Farmers miss schemes and services because they cannot find the information
Where? CONTEXT • Villages, often far from an agriculture office • At any hour, including when the helpline is closed	**Why?** BENEFIT • Farmers get correct answers quickly, in their own words • Helpline staff have more time for hard questions

Problem statement

> Our **small farmers** have a problem of **getting answers to common farming questions** when **helplines are busy and information is hard to find**. An ideal solution would **answer common questions instantly, in the farmer's own words, and send anything else to a person**.

How it is done today

Kisan Call Centre. A free helpline (1800-180-1551) where experts answer in local languages. Excellent, but staff time is limited and calls can wait.

Government websites and apps. The information is there, but finding the right page means knowing what to search for, often in English.

Script bots with menus. "Press 1 for insurance, 2 for loans." They only follow fixed paths and cannot understand a question typed in the farmer's own words.

Objectives

1. To read 12 common questions and answers, and many ways farmers ask them
2. To normalise the questions, keeping Hindi words typed in English letters
3. To build one document per answer and work out TF, DF, IDF and TFIDF for every word
4. To match a new question to an answer using TFIDF and cosine similarity
5. To compare TFIDF with simple word overlap, on farming and off-topic questions
6. To deploy it as a short chat, with a safe reply when it is not sure

DATA ACQUISITION

There are two files. faq.csv holds the 12 common questions and their answers. questions.csv holds 180 ways a farmer might ask them, each labelled with the answer it needs.

The answers point to real public services — the Soil Health Card, PM Fasal Bima Yojana, the Kisan Credit Card, e-NAM, the Meghdoot app, PM-KISAN — and deliberately give no pesticide names or doses, which only an expert should advise.

The farmers' questions were written for this project, mixing English and Hindi typed in English letters as people really write. For your own project, collect questions people actually ask — at a Krishi Vigyan Kendra, or from family — with their permission, and check every answer with an expert before using it.

> **About this dataset.** The data used here is a LambdaLab sample, made by a program to behave like the real thing so that the project could be run and tested. It is not real data. If you submit this project, replace it with data you collected or a public dataset, and say here where it came from and that you had permission to use it.

Features and label

The features are what the model is given; the label is the answer it must learn to give.

COLUMN	TYPE	WHAT IT HOLDS	ROLE
`question_id`	text	An ID for the question	not used
`question`	text	The farmer's question — turned into tokens, then TFIDF numbers	feature

faq_id	text	Which of the 12 answers it needs — the label	label
answer	text	The reply the bot gives (in faq.csv)	not used

The file faq.csv

The 12 common questions and the answer the bot gives for each. The whole file is **12 rows, 2.2 KB**.

The file questions.csv

Ways farmers ask the 12 questions, each labelled with the answer it needs. The whole file is **180 rows, 8.5 KB**.

The first four rows of faq.csv

FAQ_ID	QUESTION	ANSWER
soil_test	How can I get my soil tested?	Take a soil sample to your nearest soil testing laboratory or Krishi Vigyan Kendra. Under the Soil Health Card scheme you get a card showing your soil's nutrients and the fertiliser it needs.
weather	Where can I get the weather forecast for farming?	The India Meteorological Department gives district-level weather forecasts and farm advice. The Meghdoot app shows them on your phone.
insurance	How do I insure my crop?	Under the Pradhan Mantri Fasal Bima Yojana you can insure your crop through your bank, a Common Service Centre or the PMFBY website, before the last date for the season.
pests	Insects are attacking my crop. What should I do?	First find out which pest it is. Send a photo to your Krishi Vigyan Kendra or call the Kisan Call Centre before spraying anything, and use only what they recommend.

The first eight rows of questions.csv

QUESTION_ID	QUESTION	FAQ_ID
Q16	will it rain this week please	weather
Q34	Namaste, pmfby registration process please	insurance
Q20	rain forecast for my district?	weather
Q55	locusts in my field??	pests
Q146	khaad kitni dalein?	fertiliser
Q160	use of paddy straw please	stubble
Q99	Sir, crop selling price today??	mandi_price
Q165	Sir, do not burn stubble what to do?	stubble

DATA EXPLORATION

Before any model was built, the data was checked for problems and then drawn, to find the patterns a model could use.

Problems found in the data

WHAT WAS WRONG	HOW MUCH	WHAT WAS DONE
Capitals, punctuation and greetings	"Sir, khad kitni dalni chahiye?"	Everything is changed to lower case, punctuation is removed, and greetings like sir, namaste and please are stop words.

Hindi written in English letters	`mitti, beej, keede, bima, mandi bhav`	Kept — these words carry the meaning. Hindi grammar words that carry none (kaise, kahan, hai, mein) are added to the stop words, just as "how" and "where" are.
Words the bot has never seen	`some testing questions`	A word that is not in the vocabulary has no IDF, so it is simply ignored. This is the bot's biggest weakness, and the evaluation shows it.

Figure 1: IDF: common words count for little, rare words for a lot

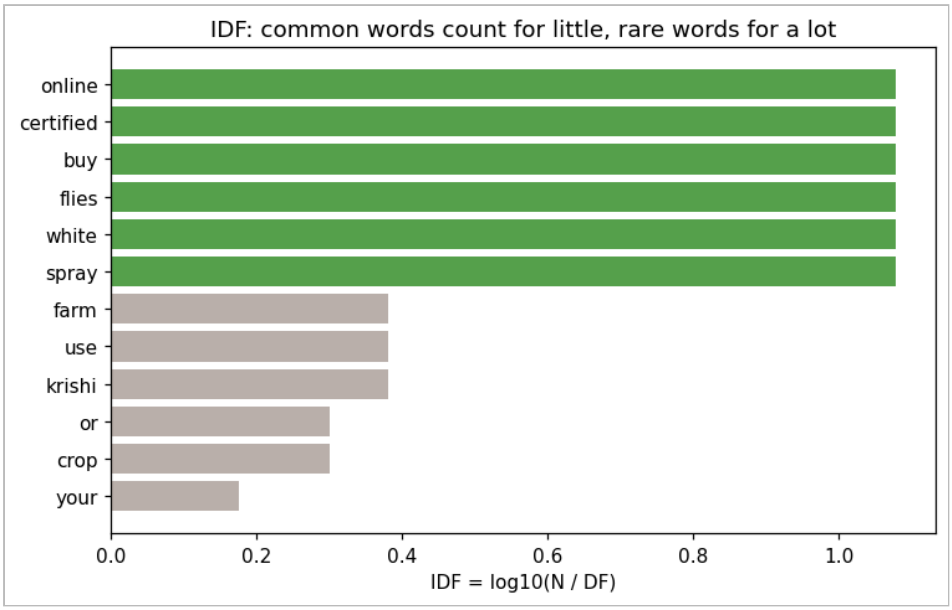


Figure 1 — IDF: common words count for little, rare words for a lot, drawn by the program as a horizontal bar chart and saved as chart1_idf.png.

How to read it. The IDF of the six most common and the six rarest words across the 12 documents. IDF = log10(N / DF), where N = 12 documents and DF is how many documents contain the word.

A word like crop appears in half of the 12 documents, so it says little about which answer a question needs. A word like mandi appears in just one document, so it gets the highest IDF possible, log10(12) ≈ 1.079. A word in every document would get log10(1) = 0. This is exactly the TFIDF idea from Unit 6: words that are everywhere are worth nothing; words that are rare are worth a lot.

Figure 2: The TFIDF table

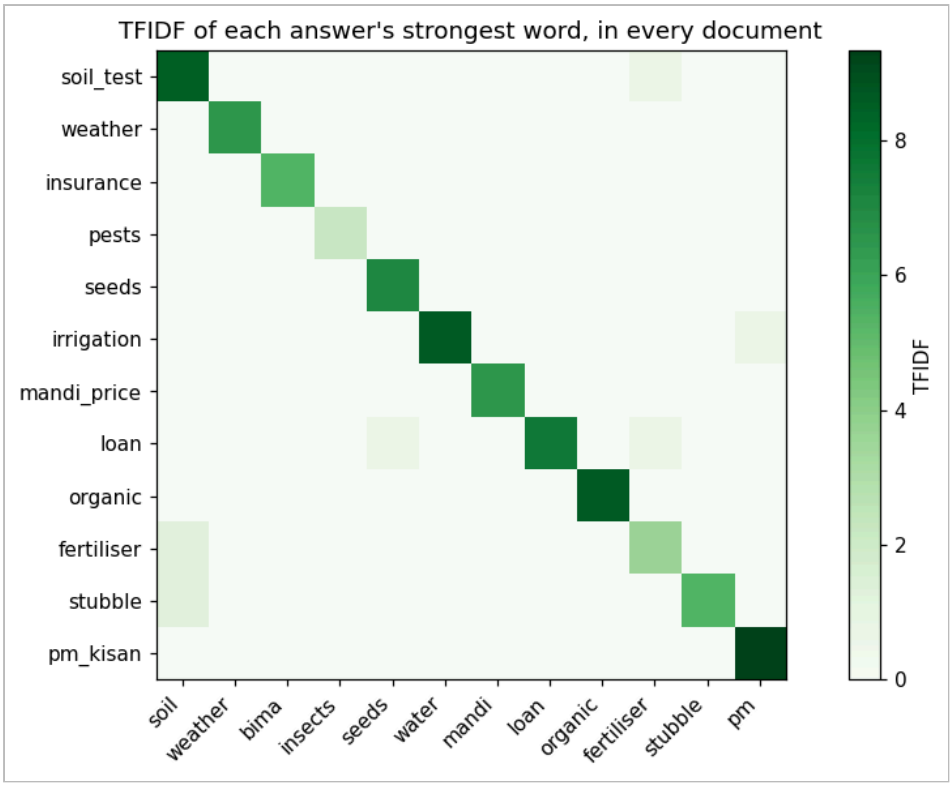


Figure 2 — The TFIDF table, drawn by the program as a confusion matrix and saved as chart2_tfidf_table.png.

How to read it. Rows are the 12 answers' documents; columns are the strongest word of each document. The darker the box, the higher that word's TFIDF in that document.

Each strong word is dark in one row and pale everywhere else. That is what makes matching work: a question containing one of these words is pulled firmly towards one answer. These words were found by the TFIDF arithmetic, not chosen by a person.

MODELLING

AI model ▶ Learning based ▶ Supervised learning ▶ Classification

Each question needs one of 12 answers, so this is classification with 12 classes. The training questions are labelled, so it is supervised learning. Because it chats with people in their own language, it is an NLP project — and a smart bot rather than a script bot.

How the model works

The labelled questions are split into 125 for training and 55 for testing. For each of the 12 answers, a document is made from its own question and answer plus all the training questions that ask for it. That gives N = 12 documents and a vocabulary of 309 words.

For every word, the program works out TF (how often it appears in each document), DF (how many documents contain it) and IDF = log10(N / DF), and multiplies TF × IDF to get the TFIDF table — exactly the method of Unit 6.

A new question is turned into TFIDF numbers in the same way, and compared with each document using cosine similarity (0 = nothing in common, 1 = the same words in the same mix). The bot replies with the answer of the most similar document — but only if the similarity is at least 0.1. Below that, it says it is not sure and gives the Kisan Call Centre number.

The simpler method it is compared with, word overlap, just counts how many words a question shares with each document.

EVALUATION

On 55 farming questions the bot had never seen, TFIDF found the right answer for 45 of 55 farming questions — an accuracy of 81.8 %. Word overlap did the same.

The misses are questions using words the bot has never met — a new spelling of a Hindi word, or a word like "locusts" that no training question used. A bot like this is only as good as the questions it has seen, which is why real helpline bots keep learning from new questions.

The second test matters more for a helpline. A bot that confidently gives a farming answer to "Who won the cricket match?" looks foolish, but one that gives a wrong farming answer to a real farming question can do harm. TFIDF's similarity score lets the bot tell a strong match from a weak one, so it refused 7 of 8 off-topic questions, against 4 of 8 for word overlap. Being able to say "I'm not sure" is part of what makes a smart bot trustworthy.

Figure 3: Answering well, and knowing when not to answer

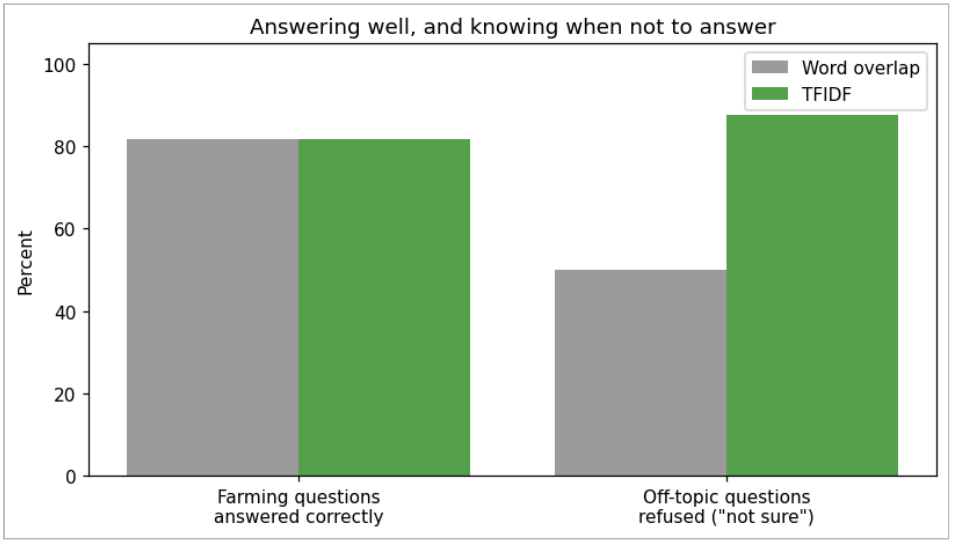


Figure 3 — Answering well, and knowing when not to answer, drawn by the program as a bar chart and saved as chart3_methods.png.

How to read it. Left pair: the share of unseen farming questions each method matched to the right answer. Right pair: the share of off-topic questions each method refused to answer.

On farming questions the two methods do equally well, 81.8 % each. The difference is on the right. Word overlap can only refuse when a question shares no word at all, so it refused 4 of 8, and gave a farming answer to the rest. TFIDF gives every match a similarity score, so the bot can see that one shared everyday word is a weak match — it refused 7 of 8.

DEPLOYMENT

The program ends with a short chat: four questions and the bot's replies. It answers a soil question asked in Hindi, a pest question and a mandi-price question, and politely refuses a cricket question, giving the Kisan Call Centre number instead.

The comment in the code shows how to chat with it live: replace the list of questions with input(). In real use the bot would sit behind an SMS number or WhatsApp, answer the common questions at any hour, and pass everything else to a person.

Figure 4: How the bot chose its answer to one question

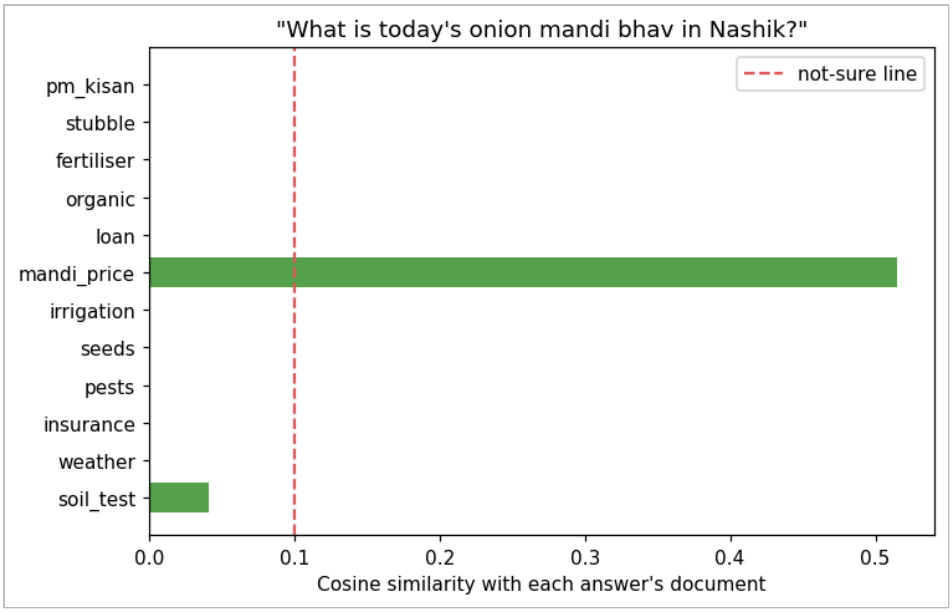


Figure 4 — How the bot chose its answer to one question, drawn by the program as a horizontal bar chart and saved as chart4_one_question.png.

How to read it. For the question shown, its cosine similarity with each of the 12 answers' documents. The red dashed line is the not-sure line.

One answer stands far above the others, well past the not-sure line, so the bot answers with confidence. For an off-topic question, every bar stays short, below the line, and the bot says it is not sure instead of guessing.

ETHICAL CONCERNS

A model that works is not finished until it has been checked for fairness (bias), for whether it can explain itself (transparency), and for who answers for its mistakes (accountability).

Advice that could cause harm

A wrong answer about pesticides or doses could harm crops, people and the soil. So the answers point farmers to experts and official services, and the bot never names a chemical or a dose. Every answer must be checked by an agriculture expert before the bot is used.

Knowing its limits

The not-sure reply is an ethical choice, not just a technical one. A bot that always gives an answer, even a wrong one, can mislead people who trust it. Saying "I am not sure, please call the helpline" keeps a person in charge of the hard cases.

Fair to every farmer

The bot understands English and Hindi in English letters. Farmers who write in Tamil, Marathi or Bengali, or in Devanagari script, would not be understood. A real bot must be built for the languages its users speak.

Accountability

The organisation running the bot is responsible for its answers, and must keep them up to date — schemes and dates change.

THE NO-CODE VERSION

Orange can turn text into TFIDF numbers with its Bag of Words widget, and compare documents with the Distances and Neighbors widgets — the same matching this program does. The tool used is **Orange Data Mining (with the Text add-on)**.

STEP	WIDGET	WHAT IT DOES
1	**Corpus**	Load questions.csv with question as the text feature and faq_id as the target.
2	**Preprocess Text**	Lowercase, remove punctuation, and filter stop words — add the Hindi ones too.
3	**Bag of Words**	Set Term Frequency to Count and Document Frequency to IDF. This gives TFIDF.
4	**Word Cloud**	Compare the words of one faq_id with another.
5	**Test and Score + kNN (cosine distance)**	Classify the questions and read the accuracy.
6	**Neighbors**	Give it one new question and see which training questions are closest — the same idea as the bot's match.

TECHNICAL DETAILS

Domain of AI	Natural Language Processing
Programming language	Python 3
Libraries	pandas, NumPy and matplotlib (Unit 7 of the curriculum)
Where to run it	Jupyter Notebook (farmer_helpline.ipynb), or any Python editor (farmer_helpline.py)
No-code tool	Orange Data Mining (with the Text add-on)
Model type	Learning based → Supervised learning → Classification
Charts drawn	4

Setting it up

1. Install Python 3 (Anaconda includes Jupyter Notebook and all three libraries).
2. If they are missing, install the libraries: **pip install pandas numpy matplotlib**
3. Keep the program and its data together in one folder.
4. Run **python farmer_helpline.py**, or open **farmer_helpline.ipynb** and run each cell in order.

The library calls it uses

CALL	STAGE	WHAT IT DOES

pd.read_csv()	Stage 2	Reads the FAQ and the questions
ch.isalpha() / split() / stop_words	Stage 3	Text normalisation
doc = doc + tokens	Stage 4	Joins all the questions for one answer into one document
np.zeros((N, words))	Stage 4	An empty TF table
(tf > 0).sum(axis=0)	Stage 4	DF: how many documents contain each word
np.log10(N / df_count)	Stage 4	IDF for every word
tf * idf	Stage 4	The TFIDF table
(a * b).sum() / (len_a * len_b)	Stages 5, 6	Cosine similarity
np.argmax(sims)	Stages 5, 6	The answer with the highest similarity
input()	Stage 6 (comment)	How to chat with the bot yourself

SOURCE CODE

The complete program. Every figure and number in this report came from running it.

farmer_helpline.py

```
# ---------------------------------------------------------------------------
# farmer_helpline.py
#
# AI project on Natural Language Processing: a helpline chatbot that reads
# a farmer's question, typed in their own words, and replies with the right
# answer from a list of 12 common questions.
#
# CBSE Class X, Artificial Intelligence (417), Part C project work.
# SDG 2 (Zero Hunger) and SDG 1 (No Poverty).
#
# It uses TFIDF, exactly as Unit 6 defines it:
#     TF(W)    = how many times word W appears in a document
#     IDF(W)   = log10( N / DF(W) )   N = number of documents,
#                                     DF = documents containing W
#     TFIDF(W) = TF(W) x IDF(W)
# Each of the 12 answers gets a "document" made of the ways farmers ask for
# it. A new question is turned into TFIDF numbers too, and the bot replies
# with the answer whose document is most similar (cosine similarity).
# If nothing is similar enough, it says so, instead of guessing.
#
# Libraries: pandas (the questions), numpy (the arithmetic), matplotlib.
# ---------------------------------------------------------------------------

# %% Stage 1 - Problem scoping
print("=" * 60)
print("A HELPLINE BOT FOR FARMERS")
print("=" * 60)
print("Who   : farmers with a question and a mobile phone")
print("What  : the same questions asked again and again; helplines are busy")
print("Where : villages, at any hour, in the farmer's own words")
```

```
print("Why   : quick, correct answers help farmers act in time")
print()

# %% Stage 2 - Data acquisition
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

pd.set_option("display.width", 140)
pd.set_option("display.max_colwidth", 70)

faq = pd.read_csv("faq.csv")
questions = pd.read_csv("questions.csv")

print("--- Stage 2: Data acquisition ---")
print("Answers in the FAQ            :", len(faq))
print("Farmers' questions (labelled) :", len(questions))
print()
print(faq[["faq_id", "question"]].to_string(index=False))
print()

# %% Stage 3 - Data exploration
print("--- Stage 3: Data exploration ---")

# Words that appear in almost every question and say nothing about the
# topic - English and Hindi alike.
stop_words = ["how", "to", "my", "i", "the", "a", "an", "is", "what", "where", "can", "do",
              "in", "for", "of", "on", "at", "and", "it", "sir", "please", "tell", "namaste",
              "help", "kaise", "kahan", "kya", "hai", "ka", "ki", "ke", "mein", "se", "karein",
              "which", "when", "there", "any", "me", "get", "should", "will", "this", "are",
              "be", "with", "after", "now", "what", "kab", "hain"]

tokens_list = []
for text in questions["question"]:
    clean = ""
    for ch in text.lower():
        if ch.isalpha():
            clean = clean + ch
        else:
            clean = clean + " "
    tokens = []
    for w in clean.split():
        if w not in stop_words:
            tokens.append(w)
    tokens_list.append(tokens)
questions["tokens"] = tokens_list

print("Text normalisation of one question:")
print("  Original :", questions.iloc[0]["question"])
print("  Tokens   :", questions.iloc[0]["tokens"])
print()

# %% Stage 4 - Modelling
print("--- Stage 4: Modelling ---")
questions = questions.sample(frac=1, random_state=4)
cut = int(len(questions) * 0.7)
train = questions[:cut]
test = questions[cut:]
print("Training questions :", len(train))
print("Testing questions  :", len(test))
```

```
# One document per answer: the FAQ's own question and answer, plus all the
# training questions that ask for it.
faq_ids = list(faq["faq_id"])
documents = []
for k in range(len(faq)):
    doc = []
    clean = ""
    for ch in (faq.iloc[k]["question"] + " " + faq.iloc[k]["answer"]).lower():
        if ch.isalpha():
            clean = clean + ch
        else:
            clean = clean + " "
    for w in clean.split():
        if w not in stop_words:
            doc.append(w)
    for tokens in train[train["faq_id"] == faq_ids[k]]["tokens"]:
        doc = doc + tokens
    documents.append(doc)

vocabulary = []
for doc in documents:
    for w in doc:
        if w not in vocabulary:
            vocabulary.append(w)
N = len(documents)
print("Documents (N)      :", N)
print("Words in vocabulary:", len(vocabulary))
print()

# TF: a table with one row per document and one column per word.
tf = np.zeros((N, len(vocabulary)))
for d in range(N):
    for w in documents[d]:
        tf[d, vocabulary.index(w)] = tf[d, vocabulary.index(w)] + 1

# DF and IDF for every word.
df_count = (tf > 0).sum(axis=0)
idf = np.log10(N / df_count)
tfidf = tf * idf

idf_table = pd.Series(idf, index=vocabulary).round(3)
print("IDF of some words (low = in many documents, high = in one):")
for w in ["crop", "farming", "water", "mandi", "bima", "keede"]:
    if w in vocabulary:
        print("  " + w.ljust(10), "DF =", str(df_count[vocabulary.index(w)]).rjust(2), "  IDF =",
idf_table[w])
print()

# Chart 1: IDF of the most and least common words.
show = pd.concat([idf_table.sort_values().head(6), idf_table.sort_values().tail(6)])
plt.figure(figsize=(7, 4.5))
plt.barh(show.index, show.values, color=["#bab0ac"] * 6 + ["#59a14f"] * 6)
plt.title("IDF: common words count for little, rare words for a lot")
plt.xlabel("IDF = log10(N / DF)")
plt.tight_layout()
plt.savefig("chart1_idf.png")
plt.close()

# Chart 2: the TFIDF table for the strongest word of each answer.
top_words = []
for d in range(N):
```

```
    w = vocabulary[int(np.argmax(tfidf[d]))]
    if w not in top_words:
        top_words.append(w)
grid = np.zeros((N, len(top_words)))
for k in range(len(top_words)):
    grid[:, k] = tfidf[:, vocabulary.index(top_words[k])]
plt.figure(figsize=(9, 5.5))
plt.imshow(grid, cmap="Greens")
plt.xticks(range(len(top_words)), top_words, rotation=45, ha="right")
plt.yticks(range(N), faq_ids)
plt.colorbar(label="TFIDF")
plt.title("TFIDF of each answer's strongest word, in every document")
plt.tight_layout()
plt.savefig("chart2_tfidf_table.png")
plt.close()

# %% Stage 5 - Evaluation
print("--- Stage 5: Evaluation ---")
threshold = 0.1          # below this similarity, the TFIDF bot admits it is not sure

# Test 1 - farming questions the bot has never seen: is the best match the
# right answer?
overlap_right = 0
tfidf_right = 0
for i in range(len(test)):
    q = np.zeros(len(vocabulary))
    for w in test.iloc[i]["tokens"]:
        if w in vocabulary:
            q[vocabulary.index(w)] = q[vocabulary.index(w)] + 1
    # Model A - word overlap: the answer whose document shares the most words.
    shared = ((tf > 0) * (q > 0)).sum(axis=1)
    if faq_ids[int(np.argmax(shared))] == test.iloc[i]["faq_id"]:
        overlap_right = overlap_right + 1
    # Model B - TFIDF and cosine similarity.
    qv = q * idf
    sims = np.zeros(N)
    if (qv ** 2).sum() > 0:
        sims = (tfidf * qv).sum(axis=1) / (np.sqrt((tfidf ** 2).sum(axis=1)) * np.sqrt((qv **
2).sum()))
    if faq_ids[int(np.argmax(sims))] == test.iloc[i]["faq_id"]:
        tfidf_right = tfidf_right + 1

overlap_acc = overlap_right / len(test) * 100
tfidf_acc = tfidf_right / len(test) * 100
print("Word overlap : right answer for", overlap_right, "of", len(test), "farming questions")
print("TFIDF        : right answer for", tfidf_right, "of", len(test), "farming questions")
print("Accuracy of word overlap :", round(overlap_acc, 1), "%")
print("Accuracy of TFIDF        :", round(tfidf_acc, 1), "%")
print()

# Test 2 - questions that are not about farming at all. A good bot should
# say "I am not sure" instead of giving a farming answer. Word overlap can
# only refuse when not a single word is shared; TFIDF refuses when the best
# similarity is below the threshold.
off_topic = ["Who won the cricket match yesterday?", "What is the capital of France?",
             "My phone screen is broken", "Tell me a joke", "How to make masala tea",
             "Train timings to Delhi", "Best mobile under 10000", "When is Diwali this year?"]
overlap_refused = 0
tfidf_refused = 0
print("Off-topic question                     words shared   best TFIDF similarity")
for text in off_topic:
```

```
    clean = ""
    for ch in text.lower():
        if ch.isalpha():
            clean = clean + ch
        else:
            clean = clean + " "
    q = np.zeros(len(vocabulary))
    for w in clean.split():
        if w not in stop_words and w in vocabulary:
            q[vocabulary.index(w)] = q[vocabulary.index(w)] + 1
    shared = ((tf > 0) * (q > 0)).sum(axis=1).max()
    qv = q * idf
    best = 0
    if (qv ** 2).sum() > 0:
        best = ((tfidf * qv).sum(axis=1) / (np.sqrt((tfidf ** 2).sum(axis=1)) * np.sqrt((qv **
2).sum()))).max()
    if shared == 0:
        overlap_refused = overlap_refused + 1
    if best < threshold:
        tfidf_refused = tfidf_refused + 1
    print(text.ljust(40), str(shared).rjust(6), str(round(best, 3)).rjust(16))
print()
print("Off-topic questions refused by word overlap :", overlap_refused, "of", len(off_topic))
print("Off-topic questions refused by TFIDF        :", tfidf_refused, "of", len(off_topic))
print()

# Chart 3: the two methods on both tests.
x = np.arange(2)
plt.figure(figsize=(7.5, 4.3))
plt.bar(x - 0.2, [overlap_acc, overlap_refused / len(off_topic) * 100], width=0.4,
        label="Word overlap", color="#9c9c9c")
plt.bar(x + 0.2, [tfidf_acc, tfidf_refused / len(off_topic) * 100], width=0.4,
        label="TFIDF", color="#59a14f")
plt.xticks(x, ["Farming questions\nanswered correctly", "Off-topic questions\nrefused (\"not
sure\")"])
plt.ylim(0, 105)
plt.ylabel("Percent")
plt.title("Answering well, and knowing when not to answer")
plt.legend()
plt.tight_layout()
plt.savefig("chart3_methods.png")
plt.close()

# %% Stage 6 - Deployment
print("--- Stage 6: Deployment ---")
# To chat with the bot yourself, replace this list with:
#     chat = [input("Your question: ")]
chat = [
    "Mere khet ki mitti ki jaanch karwani hai",
    "Insects are eating my cotton crop, what to do?",
    "What is today's onion mandi bhav in Nashik?",
    "Who is the captain of the Indian cricket team?",
]
for text in chat:
    clean = ""
    for ch in text.lower():
        if ch.isalpha():
            clean = clean + ch
        else:
            clean = clean + " "
    q = np.zeros(len(vocabulary))
```

```
    for w in clean.split():
        if w not in stop_words and w in vocabulary:
            q[vocabulary.index(w)] = q[vocabulary.index(w)] + 1
    qv = q * idf
    sims = np.zeros(N)
    if (qv ** 2).sum() > 0:
        sims = (tfidf * qv).sum(axis=1) / (np.sqrt((tfidf ** 2).sum(axis=1)) * np.sqrt((qv **
2).sum()))
    if text == chat[2]:
        third = sims.copy()             # kept for chart 4
    print("Farmer :", text)
    if sims.max() >= threshold:
        best = int(np.argmax(sims))
        print("Bot    :", faq.iloc[best]["answer"])
        print("         (matched '" + faq_ids[best] + "', similarity " + str(round(sims.max(), 2))
+ ")")
    else:
        print("Bot    : Sorry, I am not sure. Please call the Kisan Call Centre on 1800-180-1551.")
    print()

# Chart 4: how similar the third question was to each answer.
plt.figure(figsize=(7, 4.5))
plt.barh(faq_ids, third, color="#59a14f")
plt.axvline(threshold, color="#e15759", linestyle="--", label="not-sure line")
plt.title("\"" + chat[2] + "\"")
plt.xlabel("Cosine similarity with each answer's document")
plt.legend()
plt.tight_layout()
plt.savefig("chart4_one_question.png")
plt.close()
```

SAMPLE OUTPUT

An actual run of the program, exactly as it appeared on screen.

Output — python farmer_helpline.py

```
============================================================
A HELPLINE BOT FOR FARMERS
============================================================
Who   : farmers with a question and a mobile phone
What  : the same questions asked again and again; helplines are busy
Where : villages, at any hour, in the farmer's own words
Why   : quick, correct answers help farmers act in time

--- Stage 2: Data acquisition ---
Answers in the FAQ           : 12
Farmers' questions (labelled) : 180

     faq_id                                        question
  soil_test                  How can I get my soil tested?
    weather Where can I get the weather forecast for farming?
  insurance                         How do I insure my crop?
      pests  Insects are attacking my crop. What should I do?
      seeds             Where can I buy good quality seeds?
 irrigation             How can I save water in irrigation?
mandi_price                 Where can I check mandi prices?
       loan                How do I get a loan for farming?
    organic                How can I start organic farming?
 fertiliser              How much fertiliser should I use?
    stubble    What can I do instead of burning crop stubble?
```

```
   pm_kisan              How do I check my PM-KISAN payment?

--- Stage 3: Data exploration ---
Text normalisation of one question:
  Original : will it rain this week please
  Tokens   : ['rain', 'week']

--- Stage 4: Modelling ---
Training questions : 125
Testing questions  : 55
Documents (N)      : 12
Words in vocabulary: 309

IDF of some words (low = in many documents, high = in one):
  crop       DF =  6   IDF = 0.301
  farming    DF =  4   IDF = 0.477
  water      DF =  1   IDF = 1.079
  mandi      DF =  1   IDF = 1.079
  bima       DF =  1   IDF = 1.079

--- Stage 5: Evaluation ---
Word overlap : right answer for 45 of 55 farming questions
TFIDF        : right answer for 45 of 55 farming questions
Accuracy of word overlap : 81.8 %
Accuracy of TFIDF        : 81.8 %

Off-topic question                     words shared   best TFIDF similarity
Who won the cricket match yesterday?          1            0.093
What is the capital of France?                0                0
My phone screen is broken                     1            0.104
Tell me a joke                                0                0
How to make masala tea                        1            0.086
Train timings to Delhi                        0                0
Best mobile under 10000                       1            0.056
When is Diwali this year?                     0                0

Off-topic questions refused by word overlap : 4 of 8
Off-topic questions refused by TFIDF        : 7 of 8

--- Stage 6: Deployment ---
Farmer : Mere khet ki mitti ki jaanch karwani hai
Bot    : Take a soil sample to your nearest soil testing laboratory or Krishi Vigyan Kendra. Under the
Soil Health Card scheme you get a card showing your soil's nutrients and the fertiliser it needs.
         (matched 'soil_test', similarity 0.27)

Farmer : Insects are eating my cotton crop, what to do?
Bot    : First find out which pest it is. Send a photo to your Krishi Vigyan Kendra or call the Kisan
Call Centre before spraying anything, and use only what they recommend.
         (matched 'pests', similarity 0.33)

Farmer : What is today's onion mandi bhav in Nashik?
Bot    : The e-NAM portal and Agmarknet show the prices of crops in mandis across India, updated every
day.
         (matched 'mandi_price', similarity 0.51)

Farmer : Who is the captain of the Indian cricket team?
Bot    : Sorry, I am not sure. Please call the Kisan Call Centre on 1800-180-1551.
```

TESTING

The program was tested by running it. Every case gives the whole program a small input of its own and records what actually came back.

TEST CASE	EXPECTED	ACTUAL RESULT	RESULT

All 12 answers are read	`Answers in the FAQ : 12`	`Answers in the FAQ : 12`	Pass
One document for each answer: N = 12	`Documents (N) : 12`	`Documents (N) : 12`	Pass
A word in only one document gets the highest IDF, log10(12/1)	`mandi DF = 1 IDF = 1.079`	`mandi DF = 1 IDF = 1.079`	Pass
A word in many documents gets a low IDF	`crop DF = 6 IDF = 0.301`	`crop DF = 6 IDF = 0.301`	Pass
TFIDF refuses off-topic questions that word overlap would answer	`Off-topic questions refused by TFIDF : 7 of 8`	`Off-topic questions refused by TFIDF : 7 of 8`	Pass
An off-topic question gets the safe reply	`Bot : Sorry, I am not sure. Please call the Kisan Call Centre on 1800-180-1551.`	`Bot : Sorry, I am not sure. Please call the Kisan Call Centre on 1800-180-1551.`	Pass
Three answers with nothing in common: every testing question matched	`Accuracy of TFIDF : 100.0 %`	`Accuracy of TFIDF : 100.0 %`	Pass
With N = 3, a word in all three documents would get IDF log10(3/3) = 0; one in a single document gets log10(3) = 0.477	`mandi DF = 1 IDF = 0.477`	`mandi DF = 1 IDF = 0.477`	Pass
The FAQ file is missing	`FileNotFoundError: [Errno 2] No such file or directory: 'faq.csv'`	`FileNotFoundError: [Errno 2] No such file or directory: 'faq.csv'`	Pass

RESULTS AND FUTURE SCOPE

What the project found

1. Common words like crop get a low IDF; words in only one document, like mandi, get the highest, log10(12) ≈ 1.079.
2. TFIDF matched 81.8 % of unseen farming questions to the right answer.
3. Its misses were questions with words it had never seen — the bot is only as good as its examples.
4. TFIDF refused 7 of 8 off-topic questions; word overlap refused only 4 of 8.
5. A similarity score lets a bot say "I am not sure", which makes it safer to trust.

Limitations

- Only 12 questions; a real helpline has hundreds
- Words it has never seen are ignored, so new spellings and new topics are missed
- English and Hindi in English letters only
- It matches whole questions; it cannot hold a conversation or ask a follow-up question

Future scope

- Save every question the bot is not sure about, so a person can answer it and add it to the data

- Add more questions and more ways of asking each one
- Add another language you know
- Ask a follow-up question when two answers score almost the same

What had to be learnt along the way

- TF, DF, IDF and TFIDF, worked out on real documents
- Cosine similarity between a question and a document
- The difference between a script bot and a smart bot
- Using input() to make a program interactive

CONCLUSION

The project built a working helpline bot from TFIDF and cosine similarity alone. It matched 81.8 % of unseen farming questions to the right answer, in English and in Hindi typed in English letters.

Its most valuable skill turned out to be refusing. Word overlap and TFIDF found the right answer equally often, but only TFIDF's similarity score let the bot see when a match was weak — so it said "I am not sure" to 7 of 8 off-topic questions instead of giving a confident wrong answer.

A bot like this does not replace the Kisan Call Centre. It answers the common questions at any hour, and sends everything else to the people who know.

REFERENCES

1. Kisan Call Centre, Department of Agriculture and Farmers Welfare — 1800-180-1551
2. Soil Health Card — https://soilhealth.dac.gov.in/ ; PM Fasal Bima Yojana — https://pmfby.gov.in/ ; PM-KISAN — https://pmkisan.gov.in/
3. e-NAM — https://enam.gov.in/ ; Agmarknet — https://agmarknet.gov.in/
4. Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28(1) — where IDF comes from.
5. CBSE, Artificial Intelligence (Subject Code 417), Class X curriculum, 2026-27 — Part C, Practical and Project Work.
6. CBSE, Artificial Intelligence Facilitator Handbook, Class X.
7. pandas documentation — https://pandas.pydata.org/docs/
8. NumPy documentation — https://numpy.org/doc/
9. Matplotlib documentation — https://matplotlib.org/stable/
10. LambdaLab — https://www.lambdalab.in