(name of your school)

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

on

“IS THIS SMS A SCAM?”

Natural Language Processing · SDG 16: Peace, Justice and Strong Institutions · SDG 9: Industry, Innovation and Infrastructure

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 **“Is This SMS a Scam?”** 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	8
6	Evaluation	9
7	Deployment	10
8	Ethical Concerns	10
9	The No-Code Version	11
10	Technical Details	11
11	Source Code	12
12	Sample Output	17
13	Testing	19
14	Results and Future Scope	19
15	Conclusion	20
16	References	20

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

INTRODUCTION

Almost everyone in India now gets scam messages: "You have won Rs 25 lakh", "Your KYC is pending, your account will be blocked", "Pay Rs 500 to confirm your job". Some people believe them and lose money or share an OTP.

The hard part is that real messages look similar. A real bank sends "Rs 2,999 debited from your account"; a scammer sends the same words with a phone number to call. Blocking every message about money would block the alerts people need.

This project reads a message and decides whether it looks like a scam, using the Bag of Words from Unit 6.

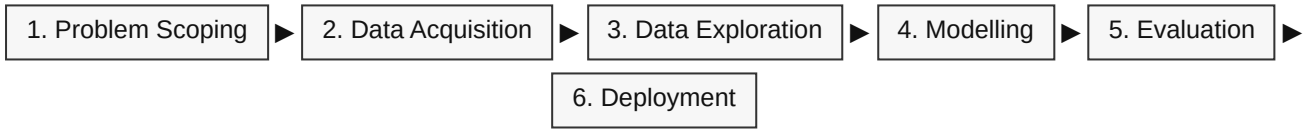
Who would use it: Mobile phone users — especially grandparents and first-time smartphone users — and the makers of SMS and messaging apps.

Link to the Sustainable Development Goals

SDG 16 — Peace, Justice and Strong Institutions; SDG 9 — Industry, Innovation and Infrastructure. SDG 16 includes reducing crime and fraud of every kind, and SDG 9 asks that people can safely use the digital services now built for them. Scam messages — fake prizes, fake KYC alerts, fake job offers — cheat thousands of people in India every day, often those newest to mobile banking. A warning on the message itself, before anyone clicks or pays, is one of the best defences.

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 • Mobile phone users, especially the elderly and first-time users • Banks and phone companies, who are blamed when customers are cheated	**What?** THE PROBLEM • Scam messages trick people into paying or sharing an OTP • Real alerts look like scams, so blocking everything is not an answer
Where? CONTEXT • On every mobile phone • At any time — scams often say "today" or "now" to rush people	**Why?** BENEFIT • A warning at the right moment can stop a fraud • People can trust the genuine messages they receive

Problem statement

> Our **mobile phone users** have a problem of **being cheated by scam text messages** when **scams are written to look like genuine alerts from banks and companies**. An ideal solution would **warn the user when a message looks like a scam, without hiding genuine messages**.

How it is done today

Phone company spam filters. They catch a lot, mostly by blocking known senders. New scams from new numbers get through.

Awareness campaigns. Banks and the government warn people never to share an OTP. Very important, but it relies on the person remembering at the moment of reading.

Blocking every message with a link. Simple, but genuine delivery, school and bank messages also contain links.

Objectives

1. To read a set of labelled SMS messages from a CSV file
2. To normalise the text, turning every link into "link" and every number into "number"
3. To build a Bag of Words table for all the messages
4. To find the words that give a scam away
5. To build a model that compares each message with the average scam and the average genuine message
6. To evaluate it, look at its mistakes, and choose a threshold

DATA ACQUISITION

Each row is one SMS with a label: scam or genuine.

The file has 481 rows. Every web link in it uses the reserved .example domain, so none points to a real website, and every phone number is made up.

The messages were written by a program in the styles common in India — including scams dressed up as bank alerts, and genuine messages that sound urgent — so no real person's messages are used. For your own project, never collect other people's messages. Use a public SMS spam dataset (and credit it), or write examples yourself from the warnings your bank publishes.

> **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
`sms_id`	text	An ID for the message	not used
`text`	text	The message — turned into tokens and a Bag of Words	feature
`label`	text	scam or genuine — what the model must predict	label

The file sms_messages.csv

Scam and genuine text messages. One message was stored twice. The whole file is **481 rows, 46.8 KB**.

The first six rows of sms_messages.csv

SMS_ID	TEXT	LABEL
M10001	You have won a free iPhone! Send your address and Rs 2,999 delivery charge to 9547699478	scam
M10002	Rs 75 debited from your account XX5241. If this was not you, call 9355398793 to cancel	scam
M10003	FREE recharge offer for you! Get Rs 2,999 talktime, click http://kyc-update.example/verify before midnight	scam
M10004	4610 is your OTP for login. Do not share it with anyone. It is valid for 10 minutes	genuine
M10005	URGENT: your electricity will be cut tonight as bill is unpaid. Call officer 9967193435 immediately	scam
M10006	Reminder: your library book is due back on Monday	genuine

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
A message stored twice	1 (removed)	Removed with drop_duplicates().
Every link is different	about half of the scams	The exact address does not matter, but having a link does. So every word starting with http becomes the single token link.
Every number is different	most messages	Amounts, phone numbers and OTPs are all different. A run of digits becomes the single token number, so "Rs 25,00,000" becomes rs number.
Capitals and punctuation	"FREE RECHARGE!"	Everything is changed to lower case and punctuation is removed, as in any text normalisation. Stop words are removed too.

Figure 1: Words that give a scam away

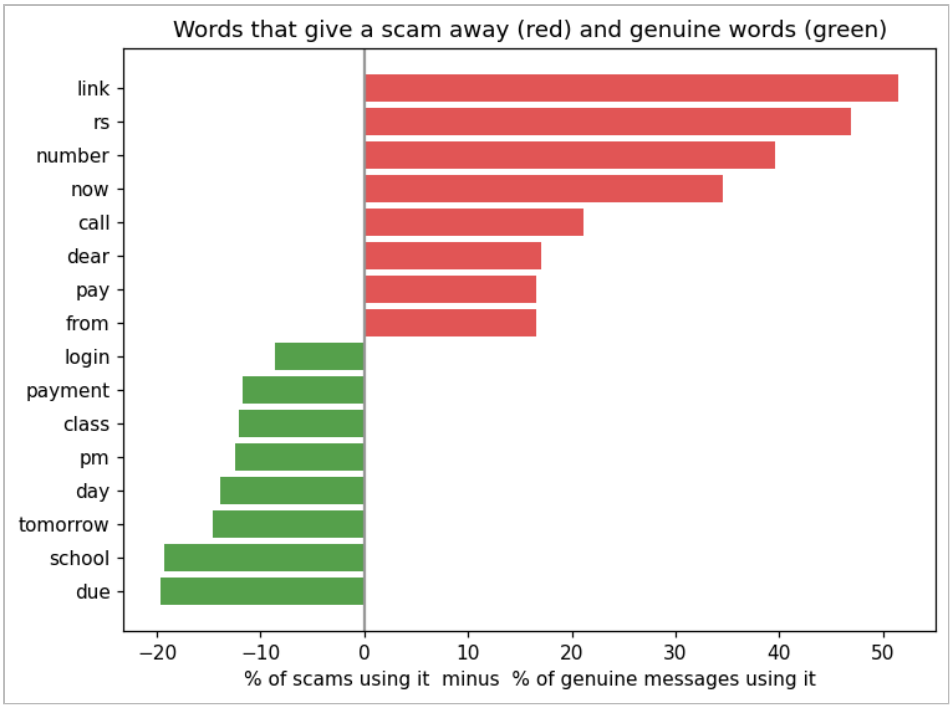


Figure 1 — Words that give a scam away, drawn by the program as a horizontal bar chart and saved as chart1_telltale_words.png.

How to read it. For each word: the share of scams that use it, minus the share of genuine messages that use it. Red words lean towards scam, green words towards genuine.

The strongest scam signs are link, rs, number, now and call. A link appears in about half the scams and in no genuine message at all. The genuine words are everyday ones — due, school, tomorrow, class, login.

Notice that rs and number also appear in many genuine messages. Bank alerts are full of amounts. That overlap is where the model will make its mistakes.

MODELLING

AI model ▶ Learning based ▶ Supervised learning ▶ Classification

The answer is scam or genuine, so this is classification. The training messages are labelled, so it is supervised learning, in the NLP domain.

How the model works

Every message becomes a row of the Bag of Words table: one column for each of the 158 words in the vocabulary, holding how many times that word appears in the message.

The messages are split into 384 for training and 96 for testing. From the training messages the model builds two average rows: the average scam and the average genuine message.

To judge a message, it measures how similar the message is to each average, using cosine similarity — a number from 0 (no words in common) to 1 (exactly the same mix of words). If the message is more similar to the average scam than to the average genuine message by more than the threshold, it is flagged. The threshold starts at 0.

EVALUATION

On 96 testing messages the accuracy was 87.5 %, with recall 100.0 % and precision 78.2 %. Every scam was caught.

The mistakes are printed in the output, and every one is a genuine bank alert — money credited or debited, with an amount and a call to the bank. These share rs, number, account and call with the scams. That is not a fault in the program so much as a fact about scams: they are written to look exactly like these alerts.

Here, both mistakes matter. A missed scam can empty a bank account. But a genuine alert marked as a scam may never be read — and that alert might be the warning that a real fraud is happening. So this project uses the F1 score, which balances precision and recall, to choose the threshold. Raising it from 0 to 0.05 gives the best balance.

Figure 2: Confusion matrix

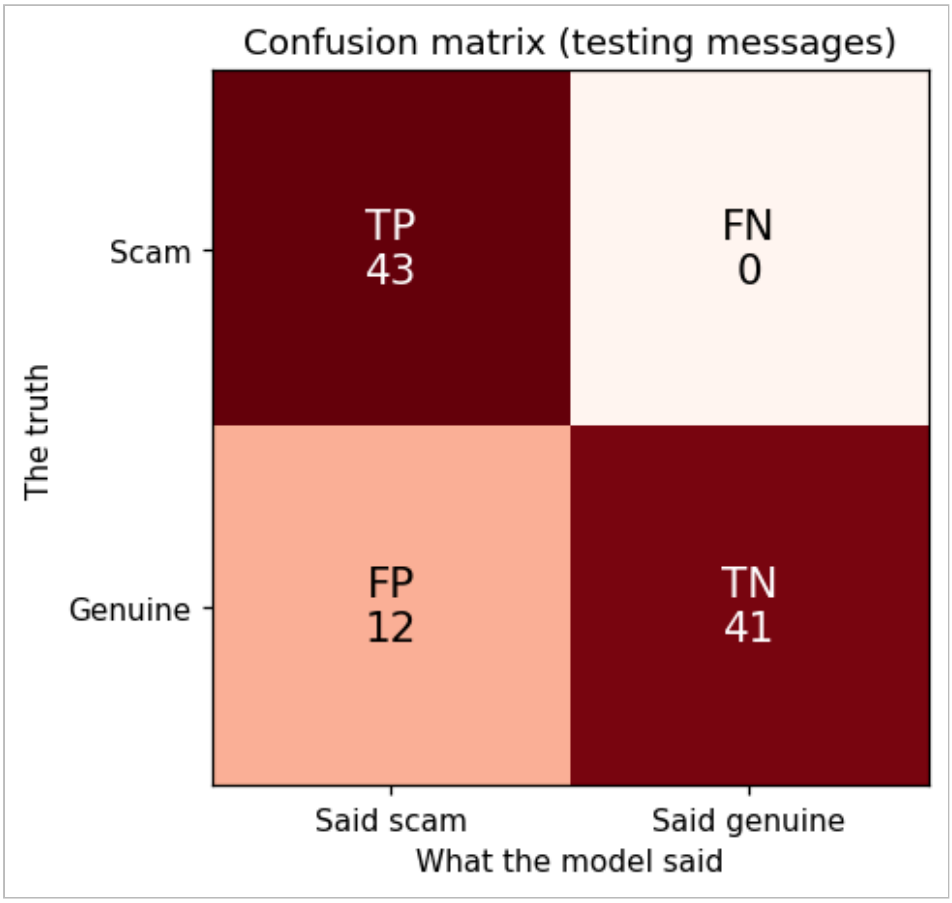


Figure 2 — Confusion matrix, drawn by the program as a confusion matrix and saved as chart2_matrix.png.

How to read it. Scam is the positive class. TP: scam and flagged. FN: scam but missed. FP: genuine but flagged. TN: genuine and passed.

On the testing messages the counts are 43 0 12 41 (TP, FN, FP, TN). No scam was missed, but some genuine messages were flagged.

Figure 3: Moving the threshold

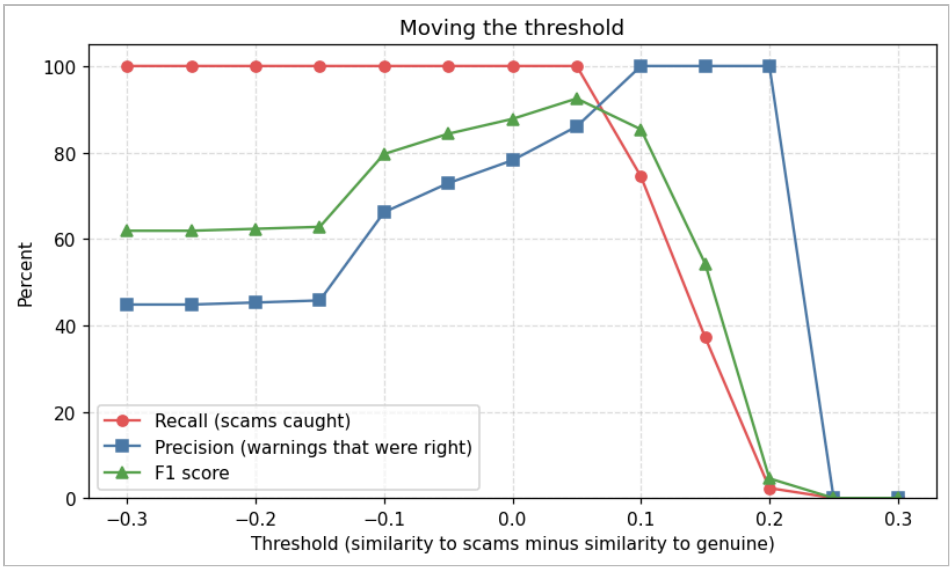


Figure 3 — Moving the threshold, drawn by the program as a line chart and saved as chart3_threshold.png.

How to read it. Across: the threshold. A message is a scam if its similarity to the average scam is higher than its similarity to the average genuine message by more than this amount. Red is recall, blue precision, green the F1 score.

At low thresholds every scam is caught but many genuine messages are flagged. As the threshold rises, fewer genuine messages are flagged, until scams start to slip through. The F1 score, which balances the two, is highest at 0.05, where it reaches 92.5 %.

DEPLOYMENT

The program checks four new messages: a genuine OTP, a fake KYC alert, a message from a child to a parent, and a fake prize. It warns about the two scams and passes the other two.

In a messaging app, a flagged message would not be deleted — it would carry a clear warning: "This looks like a scam. Never share an OTP or pay to claim a prize." The user still sees every message and decides.

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).

Never hide a message

A false positive could hide a genuine bank alert. Flagged messages should be marked with a warning, never deleted, so the user always has the final say.

Privacy

Messages are private. A real filter should check messages on the phone itself, without sending them anywhere, and must never be trained on people's messages without their permission.

Bias

The model knows English scams in the styles it was shown. Scams in Hindi or other Indian languages, or a new trick, would get through. It must be updated as scams change.

No harmful content

This dataset uses made-up numbers and reserved .example links, so it cannot direct anyone to a real scam.

THE NO-CODE VERSION

The same pipeline in Orange: text preprocessing, a Bag of Words, and a model that learns from it. The tool used is **Orange Data Mining (with the Text add-on)**.

STEP	WIDGET	WHAT IT DOES
1	**Corpus**	Load sms_messages.csv with text as the text feature and label as the target.
2	**Preprocess Text**	Lowercase, tokenise, remove stop words. Try the option to keep or remove numbers and see what changes.
3	**Bag of Words**	Turn each message into word counts.
4	**Word Cloud**	Select only the scam messages to see their most common words.
5	**Test and Score + Naive Bayes or kNN**	Train a model and read accuracy, precision and recall with target class scam.
6	**Confusion Matrix → Corpus Viewer**	Read the genuine messages that were flagged. Are they bank alerts here, too?

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 (scam_sms.ipynb), or any Python editor (scam_sms.py)
No-code tool	Orange Data Mining (with the Text add-on)
Model type	Learning based → Supervised learning → Classification
Charts drawn	3

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 scam_sms.py**, or open **scam_sms.ipynb** and run each cell in order.

The library calls it uses

CALL	STAGE	WHAT IT DOES
`pd.read_csv() / drop_duplicates()`	Stages 2, 3	Reads the messages and removes the repeat

`raw.startswith("http")`	Stage 3	Spots a web link and turns it into the token link
`ch.isdigit()`	Stage 3	Spots digits; a run of them becomes the token number
`np.zeros((messages, words))`	Stage 3	An empty Bag of Words table
`bow[m, vocabulary.index(w)] += 1`	Stage 3	Counts each word into the table
`bow[...].mean(axis=0)`	Stage 4	The average scam and the average genuine message
`(v * a).sum() / (len_v * len_a)`	Stage 4	Cosine similarity between two rows
`gaps > threshold`	Stage 4	Flags a message as a scam
`np.argmax(f_list)`	Stage 5	The threshold with the best F1 score

SOURCE CODE

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

`scam_sms.py`

```
# ---------------------------------------------------------------------------
# scam_sms.py
#
# AI project on Natural Language Processing: spotting scam text messages -
# fake prizes, fake KYC alerts, fake job offers - before someone is cheated.
#
# CBSE Class X, Artificial Intelligence (417), Part C project work.
# SDG 16 (Peace, Justice and Strong Institutions) and SDG 9 (Industry,
# Innovation and Infrastructure).
#
# Each message is turned into a BAG OF WORDS: how many times each word of
# the vocabulary appears in it. The model then builds two "average messages"
# from the training data - an average scam and an average genuine message -
# and asks of each new message: which average does it share more words with?
#
# How much two messages share is measured with cosine similarity:
#     similarity = (sum of count1 x count2 over every word)
#                  / (length of message 1 x length of message 2)
# It is 1 for messages with exactly the same mix of words and 0 for messages
# with no word in common.
#
# Libraries: pandas (the messages), numpy (the arithmetic), matplotlib.
# ---------------------------------------------------------------------------

# %% Stage 1 - Problem scoping
print("=" * 60)
print("IS THIS SMS A SCAM?")
print("=" * 60)
print("Who   : mobile phone users, especially the elderly and first-time users")
print("What  : scam messages trick people into paying or sharing an OTP")
print("Where : on every phone, every day")
print("Why   : a warning at the right moment can stop a fraud")
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", 100)

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

print("--- Stage 2: Data acquisition ---")
print("Messages read :", len(data))
print()
print(data.head(6).to_string(index=False))
print()

# %% Stage 3 - Data exploration
print("--- Stage 3: Data exploration ---")
before = len(data)
data = data.drop_duplicates()
print("Messages stored twice :", before - len(data), "(removed)")
print("Messages left         :", len(data))
print(data["label"].value_counts().to_string())
print()

stop_words = ["the", "a", "an", "is", "are", "to", "for", "of", "in", "on", "at", "your",
              "you", "it", "and", "with", "be", "will", "has", "have", "been", "this", "we",
              "our", "my", "me", "i", "or", "us", "if"]

# Text normalisation. Two extra steps for SMS: a web link becomes the
# single word "link", and any run of digits becomes the word "number" -
# the exact link or number does not matter, but having one does.
tokens_list = []
for text in data["text"]:
    words = []
    for raw in text.lower().split():
        if raw.startswith("http"):
            words.append("link")
            continue
        clean = ""
        previous_digit = False
        for ch in raw:
            if ch.isdigit():
                if not previous_digit:          # a run of digits is one "number"
                    clean = clean + " number "
                previous_digit = True
                continue
            previous_digit = False
            if ch.isalpha():
                clean = clean + ch
            elif ch != ",":                     # 25,00,000 is still one number
                clean = clean + " "
        for w in clean.split():
            if w not in stop_words:
                words.append(w)
    tokens_list.append(words)
data["tokens"] = tokens_list

print("Text normalisation of one message:")
print("  Original :", data.iloc[0]["text"])
print("  Tokens   :", data.iloc[0]["tokens"])
print()

vocabulary = []
```

```
for tokens in data["tokens"]:
    for w in tokens:
        if w not in vocabulary:
            vocabulary.append(w)
print("Words in the vocabulary :", len(vocabulary))

# The Bag of Words: one row per message, one column per vocabulary word.
bow = np.zeros((len(data), len(vocabulary)))
for m in range(len(data)):
    for w in data.iloc[m]["tokens"]:
        bow[m, vocabulary.index(w)] = bow[m, vocabulary.index(w)] + 1
print("Bag of Words table size  :", bow.shape[0], "messages x", bow.shape[1], "words")
print()

# Which words are most typical of each kind? (share of messages using them)
is_scam = (data["label"] == "scam").values
scam_share = (bow[is_scam] > 0).mean(axis=0) * 100
genuine_share = (bow[~is_scam] > 0).mean(axis=0) * 100
words = pd.DataFrame({"scam %": scam_share, "genuine %": genuine_share}, index=vocabulary)
words["difference"] = words["scam %"] - words["genuine %"]
top = words.sort_values("difference")
print("Words most typical of scams:")
print(top.tail(6)[::-1].round(1))
print()

# Chart 1: the words most typical of scams and of genuine messages.
pick = pd.concat([top.head(8), top.tail(8)])
plt.figure(figsize=(7, 5.2))
colours = []
for v in pick["difference"]:
    if v > 0:
        colours.append("#e15759")
    else:
        colours.append("#59a14f")
plt.barh(pick.index, pick["difference"], color=colours)
plt.axvline(0, color="#999999")
plt.title("Words that give a scam away (red) and genuine words (green)")
plt.xlabel("% of scams using it  minus  % of genuine messages using it")
plt.tight_layout()
plt.savefig("chart1_telltale_words.png")
plt.close()

# %% Stage 4 - Modelling
print("--- Stage 4: Modelling ---")
order = np.random.default_rng(3).permutation(len(data))
cut = int(len(data) * 0.8)
train_rows = order[:cut]
test_rows = order[cut:]
print("Training messages :", len(train_rows))
print("Testing messages  :", len(test_rows))
print()

train_scam = train_rows[is_scam[train_rows]]
train_genuine = train_rows[~is_scam[train_rows]]
average_scam = bow[train_scam].mean(axis=0)
average_genuine = bow[train_genuine].mean(axis=0)

# For every testing message: similarity to the average scam minus
# similarity to the average genuine message. Above the threshold = scam.
gaps = []
for m in test_rows:
```

```
    v = bow[m]
    length = np.sqrt((v ** 2).sum())
    if length == 0:
        gaps.append(0)
        continue
    sim_scam = (v * average_scam).sum() / (length * np.sqrt((average_scam ** 2).sum()))
    sim_genuine = (v * average_genuine).sum() / (length * np.sqrt((average_genuine ** 2).sum()))
    gaps.append(sim_scam - sim_genuine)
gaps = np.array(gaps)
actual = is_scam[test_rows]

threshold = 0.0
predicted = gaps > threshold

# %% Stage 5 - Evaluation
print("--- Stage 5: Evaluation ---")
tp = (predicted & actual).sum()
fn = (~predicted & actual).sum()
fp = (predicted & ~actual).sum()
tn = (~predicted & ~actual).sum()
accuracy = (tp + tn) / len(actual) * 100
precision = tp / (tp + fp) * 100
recall = tp / (tp + fn) * 100
f1 = 2 * precision * recall / (precision + recall)
print("TP, FN, FP, TN :", tp, fn, fp, tn)
print("Accuracy       :", round(accuracy, 1), "%")
print("Precision      :", round(precision, 1), "%")
print("Recall         :", round(recall, 1), "%")
print("F1 score       :", round(f1, 1), "%")
print()

print("Messages the model got wrong:")
for i in range(len(test_rows)):
    if predicted[i] != actual[i]:
        if actual[i]:
            label = "scam"
        else:
            label = "genuine"
        print("  [really " + label + "]", data.iloc[test_rows[i]]["text"])
print()

# Chart 2: confusion matrix.
grid = [[tp, fn], [fp, tn]]
plt.figure(figsize=(5, 4.3))
plt.imshow(grid, cmap="Reds")
names = [["TP", "FN"], ["FP", "TN"]]
for r in range(2):
    for c in range(2):
        if grid[r][c] > max(max(grid)) / 2:
            ink = "white"
        else:
            ink = "black"
        plt.text(c, r, names[r][c] + "\n" + str(grid[r][c]), ha="center", va="center", fontsize=14,
color=ink)
plt.xticks([0, 1], ["Said scam", "Said genuine"])
plt.yticks([0, 1], ["Scam", "Genuine"])
plt.xlabel("What the model said")
plt.ylabel("The truth")
plt.title("Confusion matrix (testing messages)")
plt.tight_layout()
plt.savefig("chart2_matrix.png")
```

```
plt.close()

# Both mistakes hurt here: a missed scam can empty a bank account, but a
# genuine OTP or bank alert marked as scam may never be read. So every
# threshold is tried, to see the trade-off.
thresholds = np.round(np.arange(-0.3, 0.31, 0.05), 2)
p_list = []
r_list = []
f_list = []
for t in thresholds:
    pred = gaps > t
    tp_t = (pred & actual).sum()
    fp_t = (pred & ~actual).sum()
    fn_t = (~pred & actual).sum()
    p = 0
    if tp_t + fp_t > 0:
        p = tp_t / (tp_t + fp_t) * 100
    r = tp_t / (tp_t + fn_t) * 100
    f = 0
    if p + r > 0:
        f = 2 * p * r / (p + r)
    p_list.append(p)
    r_list.append(r)
    f_list.append(f)
best = int(np.argmax(f_list))
print("Threshold with the best F1 score :", round(thresholds[best], 2))
print("Best F1 score                    :", round(f_list[best], 1), "%")
print()

# Chart 3: precision, recall and F1 for every threshold.
plt.figure(figsize=(7.5, 4.5))
plt.plot(thresholds, r_list, marker="o", color="#e15759", label="Recall (scams caught)")
plt.plot(thresholds, p_list, marker="s", color="#4e79a7", label="Precision (warnings that were
right)")
plt.plot(thresholds, f_list, marker="^", color="#59a14f", label="F1 score")
plt.title("Moving the threshold")
plt.xlabel("Threshold (similarity to scams minus similarity to genuine)")
plt.ylabel("Percent")
plt.ylim(0, 105)
plt.legend()
plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.savefig("chart3_threshold.png")
plt.close()

# %% Stage 6 - Deployment
print("--- Stage 6: Deployment ---")
new_messages = [
    "Your OTP for payment is 5512. Do not share it with anyone.",
    "Dear customer, your KYC has expired. Update now at http://kyc-update.example or account will
be blocked",
    "Mummy, I reached school. Will call at lunch",
    "Congratulations you won Rs 50,000 in our lucky draw, click http://claim-prize.example",
]
for text in new_messages:
    words = []
    for raw in text.lower().split():
        if raw.startswith("http"):
            words.append("link")
            continue
        clean = ""
```

```
        previous_digit = False
        for ch in raw:
            if ch.isdigit():
                if not previous_digit:          # a run of digits is one "number"
                    clean = clean + " number "
                previous_digit = True
                continue
            previous_digit = False
            if ch.isalpha():
                clean = clean + ch
            elif ch != ",":                     # 25,00,000 is still one number
                clean = clean + " "
        for w in clean.split():
            if w not in stop_words:
                words.append(w)
    v = np.zeros(len(vocabulary))
    for w in words:
        if w in vocabulary:                   # a word never seen in training is ignored
            v[vocabulary.index(w)] = v[vocabulary.index(w)] + 1
    length = np.sqrt((v ** 2).sum())
    gap = 0
    if length > 0:
        gap = ((v * average_scam).sum() / (length * np.sqrt((average_scam ** 2).sum()))
               - (v * average_genuine).sum() / (length * np.sqrt((average_genuine ** 2).sum())))
    if gap > threshold:
        verdict = "WARNING: looks like a scam"
    else:
        verdict = "looks genuine"
    print(text[:60].ljust(60), str(round(gap, 2)).rjust(6), "->", verdict)
```

SAMPLE OUTPUT

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

Output — python scam_sms.py

```
============================================================
IS THIS SMS A SCAM?
============================================================
Who   : mobile phone users, especially the elderly and first-time users
What  : scam messages trick people into paying or sharing an OTP
Where : on every phone, every day
Why   : a warning at the right moment can stop a fraud

--- Stage 2: Data acquisition ---
Messages read : 481

sms_id
text   label
M10001                  You have won a free iPhone! Send your address and Rs 2,999 delivery charge to
9547699478    scam
M10002                    Rs 75 debited from your account XX5241. If this was not you, call 9355398793
to cancel    scam
M10003 FREE recharge offer for you! Get Rs 2,999 talktime, click http://kyc-update.example/verify before
midnight    scam
M10004                       4610 is your OTP for login. Do not share it with anyone. It is valid for
10 minutes genuine
M10005        URGENT: your electricity will be cut tonight as bill is unpaid. Call officer 9967193435
immediately    scam
M10006                                                        Reminder: your library book is due back
on Monday genuine
```

```
--- Stage 3: Data exploration ---
Messages stored twice : 1 (removed)
Messages left         : 480
label
genuine    280
scam       200

Text normalisation of one message:
  Original : You have won a free iPhone! Send your address and Rs 2,999 delivery charge to 9547699478
  Tokens   : ['won', 'free', 'iphone', 'send', 'address', 'rs', 'number', 'number', 'delivery',
'charge', 'number']

Words in the vocabulary : 158
Bag of Words table size  : 480 messages x 158 words

Words most typical of scams:
        scam %  genuine %  difference
link      51.5        0.0        51.5
rs        65.5       18.6        46.9
number    92.5       52.9        39.6
now       34.5        0.0        34.5
call      27.5        6.4        21.1
dear      17.0        0.0        17.0

--- Stage 4: Modelling ---
Training messages : 384
Testing messages  : 96

--- Stage 5: Evaluation ---
TP, FN, FP, TN : 43 0 12 41
Accuracy       : 87.5 %
Precision      : 78.2 %
Recall         : 100.0 %
F1 score       : 87.8 %

Messages the model got wrong:
  [really genuine] Rs 1,240 credited to your account XX9038. Available balance Rs 75
  [really genuine] Rs 2,999 debited from your account XX7872 on 12-03 for UPI payment. Not you? Call
your bank
  [really genuine] Rs 10,000 debited from your account XX6964 on 12-03 for UPI payment. Not you? Call
your bank
  [really genuine] RS 10,000 DEBITED FROM YOUR ACCOUNT XX3911 ON 12-03 FOR UPI PAYMENT. NOT YOU? CALL
YOUR BANK
  [really genuine] Rs 1,240 debited from your account XX2862 on 12-03 for UPI payment. Not you? Call
your bank
  [really genuine] Rs 500 credited to your account XX6132. Available balance Rs 500
  [really genuine] Rs 75 credited to your account XX1196. Available balance Rs 25,00,000
  [really genuine] Rs 10,000 credited to your account XX5822. Available balance Rs 10,000
  [really genuine] Rs 10,000 debited from your account XX3984 on 12-03 for UPI payment. Not you? Call
your bank
  [really genuine] Rs 500 credited to your account XX1201. Available balance Rs 500
  [really genuine] Rs 10,000 credited to your account XX6294. Available balance Rs 500
  [really genuine] Rs 25,00,000 credited to your account XX7401. Available balance Rs 1,240

Threshold with the best F1 score : 0.05
Best F1 score                    : 92.5 %

--- Stage 6: Deployment ---
Your OTP for payment is 5512. Do not share it with anyone.    -0.11 -> looks genuine
Dear customer, your KYC has expired. Update now at http://ky   0.21 -> WARNING: looks like a scam
Mummy, I reached school. Will call at lunch                   -0.06 -> looks genuine
Congratulations you won Rs 50,000 in our lucky draw, click h   0.14 -> WARNING: looks like a scam
```

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 messages are read	`Messages read : 481`	`Messages read : 481`	Pass
A message stored twice is removed	`Messages stored twice : 1 (removed)`	`Messages stored twice : 1 (removed)`	Pass
A phone number is one token, not ten	`Tokens : ['won', 'free', 'iphone', 'send', 'address', 'rs', 'number', 'number', 'delivery', 'charge', 'number']`	`Tokens : ['won', 'free', 'iphone', 'send', 'address', 'rs', 'number', 'number', 'delivery', 'charge', 'number']`	Pass
Every scam in the testing set is caught	`Recall : 100.0 %`	`Recall : 100.0 %`	Pass
A slightly higher threshold gives a better balance	`Threshold with the best F1 score : 0.05`	`Threshold with the best F1 score : 0.05`	Pass
Scams and chat with no words in common: no mistakes	`Accuracy : 100.0 %`	`Accuracy : 100.0 %`	Pass
A new KYC scam with a link is flagged	`Dear customer, your KYC has expired. Update now at http://ky 0.58 -> WARNING: looks like a scam`	`Dear customer, your KYC has expired. Update now at http://ky 0.58 -> WARNING: looks like a scam`	Pass
The data file is missing	`FileNotFoundError: [Errno 2] No such file or directory: 'sms_messages.csv'`	`FileNotFoundError: [Errno 2] No such file or directory: 'sms_messages.csv'`	Pass

RESULTS AND FUTURE SCOPE

What the project found

1. A link is the strongest single sign of a scam: about half the scams had one, and no genuine message did.
2. rs, number, now and call are also typical of scams — but rs and number are common in genuine bank alerts too.
3. With the threshold at 0, every scam was caught, with an accuracy of 87.5 %.
4. Every mistake was a genuine bank alert flagged as a scam, because scams copy their wording.
5. Raising the threshold to 0.05 gave the best balance, an F1 score of 92.5 %.

Limitations

- Made-up messages, more regular than real ones
- English only
- It does not know the sender; a real bank's verified sender ID is a strong clue it ignores
- New kinds of scam, with words it has never seen, may get through

Future scope

- Add the sender as a feature: is it a verified bank or company ID, or a ten-digit number?
- Add messages in Hindi or another language you know
- Turn it into a two-stage check: scam words first, then a stricter look at messages about money
- Build a poster or short guide from chart 1: the words that give a scam away

What had to be learnt along the way

- Bag of Words as a table, with one row per message
- Cosine similarity — how alike two messages are
- Choosing a threshold when both kinds of mistake matter, using the F1 score
- How common SMS scams work — your bank's website has a page on this

CONCLUSION

The project built a scam detector from the Bag of Words alone. It caught every scam in the testing set, and with a well-chosen threshold its F1 score reached 92.5 %.

Its mistakes were the most instructive part. Every genuine message it flagged was a bank alert, because scammers deliberately copy bank wording. Words alone cannot fully separate the two — which is why real filters also check who sent the message.

Most important of all is how such a tool is used. It should warn, never delete, and leave the decision with the person reading.

REFERENCES

1. Indian Cybercrime Coordination Centre, Ministry of Home Affairs — https://cybercrime.gov.in/ (report frauds on 1930)
2. Reserve Bank of India, "BE(A)WARE" booklet on common frauds — https://rbi.org.in/
3. Almeida, T. A. and Gómez Hidalgo, J. M. (2011). SMS Spam Collection, UCI Machine Learning Repository — a public dataset for practice.
4. Orange Data Mining, Text add-on — https://orangedatamining.com/widget-catalog/text-mining/
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