Safari File Edit View History Bookmarks Window Help Mon 24 Nov 10:50 PM

colab.research.google.com

Spam Filter AI - Colab

Spam Filter AI

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all

RAM Disk

[1] 2m

```python
import math
import re

class ManualNaiveBayes:
    """
    A specific implementation of a Spam Filter using Naive Bayes logic from scratch.
    This avoids using pre-built AI libraries like sklearn to demonstrate the raw math.
    """
    def __init__(self):
        # Dictionaries to store the frequency of every word found in Spam vs Ham
        self.spam_word_counts = {}
        self.ham_word_counts = {}

        # Counters for the total number of words observed in each category
        self.spam_total_words = 0
        self.ham_total_words = 0

        # Counters for the number of messages
        self.spam_msg_count = 0
        self.ham_msg_count = 0

        # A set to keep track of every unique word we have ever seen
        self.vocabulary = set()

        # Prior probabilities (e.g., probability of any random message being spam)
        self.prob_spam_prior = 0.0
        self.prob_ham_prior = 0.0

        # Smoothing factor (alpha) prevents division by zero if a word is new
        self.smoothing_alpha = 1

    def clean_text(self, text):
        """
```

Variables Terminal

10:45 PM Python 3

Safari File Edit View History Bookmarks Window Help Mon 24 Nov 10:50 PM

colab.research.google.com

Spam Filter AI - Colab

Spam Filter AI

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all

RAM Disk

[1] 2m

```
def clean_text(self, text):
    """
    Converts text to lowercase and strips punctuation using Regex.
    Returns a list of clean words.
    """
    text = text.lower()
    # Find all sequences of alphanumeric characters (words)
    words = re.findall(r'\b\w+\b', text)
    return words

def train(self, training_data):
    print("Training model from scratch (calculating probabilities)...")
    total_msgs = len(training_data)

    for text, label in training_data:
        words = self.clean_text(text)

        if label == 'spam':
            self.spam_msg_count += 1
            # Add word counts to the spam dictionary
            for word in words:
                self.spam_word_counts[word] = self.spam_word_counts.get(word, 0) + 1
                self.spam_total_words += 1
                self.vocabulary.add(word)
        else:
            self.ham_msg_count += 1
            # Add word counts to the ham dictionary
            for word in words:
                self.ham_word_counts[word] = self.ham_word_counts.get(word, 0) + 1
                self.ham_total_words += 1
                self.vocabulary.add(word)

    # Calculate Prior Probabilities: P(Spam) and P(Ham)
    self.prob_spam_prior = self.spam_msg_count / total_msgs
    self.prob_ham_prior = self.ham_msg_count / total_msgs
```

Variables Terminal

10:45 PM Python 3

Spam Filter AI

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all

RAM Disk

[1] ✓ 2m

```
        print(f"Learned {len(self.vocabulary)} unique words.")
        print(f"Processed {self.spam_msg_count} spam and {self.ham_msg_count} non-spam messages.")

    def calculate_word_probability(self, word, is_spam_class):
        """
        Calculates P(word | class) using Laplace Smoothing.
        Formula: (Count of word in class + alpha) / (Total words in class + alpha * vocab_size)
        """
        vocab_size = len(self.vocabulary)

        if is_spam_class:
            numerator = self.spam_word_counts.get(word, 0) + self.smoothing_alpha
            denominator = self.spam_total_words + (self.smoothing_alpha * vocab_size)
        else:
            numerator = self.ham_word_counts.get(word, 0) + self.smoothing_alpha
            denominator = self.ham_total_words + (self.smoothing_alpha * vocab_size)

        return numerator / denominator

    def predict(self, message):
        words = self.clean_text(message)

        # We use Logarithms to prevent 'underflow'.
        # Multiplying many tiny probabilities (e.g., 0.0001 * 0.005) results in zero.
        # Adding logarithms is mathematically equivalent to multiplying normal probabilities.

        # Start with the Log Prior Probability
        spam_score = math.log(self.prob_spam_prior)
        ham_score = math.log(self.prob_ham_prior)

        for word in words:
            # We only calculate probabilities for words we learned during training
            if word in self.vocabulary:
                # Add log(P(word | spam)) to spam_score
                spam_score += math.log(self.calculate_word_probability(word, True))
```

Safari File Edit View History Bookmarks Window Help Mon 24 Nov 10:50 PM

colab.research.google.com

Spam Filter AI - Colab

Spam Filter AI

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all

RAM Disk

[1] 2m

```
                # Add log(P(word | ham)) to ham_score
                ham_score += math.log(self.calculate_word_probability(word, False))

        # Compare the final scores
        if spam_score > ham_score:
            return "SPAM", spam_score
        else:
            return "NOT SPAM", ham_score

# --- Data and Execution ---

def get_manual_dataset():
    # A custom list of tuples: (Message, Label)
    # Using 'spam' and 'ham' labels
    return [
        # Suspicious / Spam
        ("Win a cash prize now", "spam"),
        ("Urgent reply needed for your free ticket", "spam"),
        ("Click here to claim your reward", "spam"),
        ("Your account is locked verify immediately", "spam"),
        ("Cheap watches and meds buy now", "spam"),
        ("You have won a lottery", "spam"),
        ("Limited time offer exclusively for you", "spam"),
        ("Crypto investment huge returns guaranteed", "spam"),
        ("Lose weight fast with this pill", "spam"),
        ("Make money from home quickly", "spam"),
        ("Congratulations you are a winner", "spam"),
        ("Double your income in one week", "spam"),
        ("100% free just sign up", "spam"),

        # Normal / Ham
        ("Meeting is at 5pm today", "ham"),
        ("Hey how are you doing", "ham"),
        ("Don't forget to buy milk", "ham"),
        ("Can we reschedule our call?", "ham"),
```

Variables Terminal

10:45 PM Python 3

Safari File Edit View History Bookmarks Window Help Mon 24 Nov 10:50 PM

colab.research.google.com

Spam Filter AI - Colab

Spam Filter AI

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all

RAM Disk

[1] 2m

```
        # Normal / Ham
        ("Meeting is at 5pm today", "ham"),
        ("Hey how are you doing", "ham"),
        ("Don't forget to buy milk", "ham"),
        ("Can we reschedule our call?", "ham"),
        ("I will be late for work", "ham"),
        ("The project deadline is Monday", "ham"),
        ("Let's grab lunch tomorrow", "ham"),
        ("Did you see the game last night", "ham"),
        ("Please review the attached document", "ham"),
        ("Walking the dog right now", "ham"),
        ("Happy birthday have a great one", "ham"),
        ("Where are the keys?", "ham"),
        ("Thanks for your help yesterday", "ham")
    ]

if __name__ == "__main__":
    print("=== Custom Spam Filter (No External Libraries) ===")

    # 1. Initialize
    classifier = ManualNaiveBayes()

    # 2. Load Data
    data = get_manual_dataset()

    # 3. Train
    classifier.train(data)

    print("\nSystem ready. Type a message to check.")
    print("Type 'exit' to quit.\n")

    while True:
        try:
            user_input = input("Enter message: ")
```

Variables Terminal

10:45 PM Python 3

Safari File Edit View History Bookmarks Window Help Mon 24 Nov 10:51 PM

colab.research.google.com

Spam Filter AI - Colab

Spam Filter AI

File Edit View Insert Runtime Tools Help

Share

Commands + Code + Text Run all

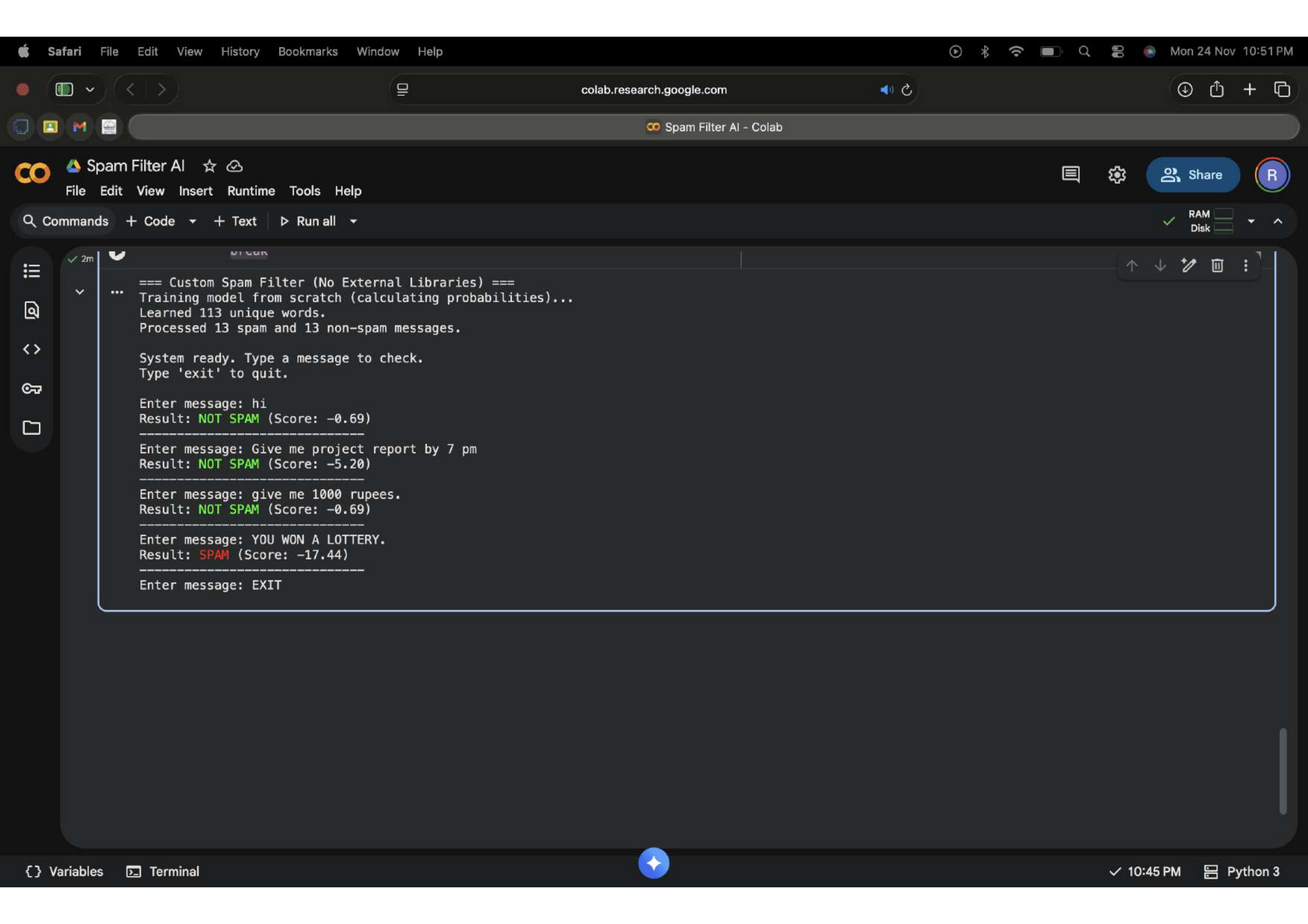
RAM Disk

[1] 2m

```
# 1. Initialize
classifier = ManualNaiveBayes()

# 2. Load Data
data = get_manual_dataset()

# 3. Train
classifier.train(data)

print("\nSystem ready. Type a message to check.")
print("Type 'exit' to quit.\n")

while True:
    try:
        user_input = input("Enter message: ")

        if user_input.lower() in ['exit', 'quit', 'q']:
            break

        if not user_input.strip():
            continue

        prediction, score = classifier.predict(user_input)

        # Simple color output
        if prediction == "SPAM":
            print(f"Result: \033[91m{prediction}\033[0m (Score: {score:.2f})")
        else:
            print(f"Result: \033[92m{prediction}\033[0m (Score: {score:.2f})")
        print("-" * 30)

    except KeyboardInterrupt:
        print("\nExiting...")
        break
```

Variables Terminal

10:45 PM Python 3


Safari File Edit View History Bookmarks Window Help
Mon 24 Nov 10:51 PM
colab.research.google.com
Spam Filter AI - Colab
Spam Filter AI
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
2m
=== Custom Spam Filter (No External Libraries) ===
Training model from scratch (calculating probabilities)...
Learned 113 unique words.
Processed 13 spam and 13 non-spam messages.
System ready. Type a message to check.
Type 'exit' to quit.
Enter message: hi
Result: NOT SPAM (Score: -0.69)
Enter message: Give me project report by 7 pm
Result: NOT SPAM (Score: -5.20)
Enter message: give me 1000 rupees.
Result: NOT SPAM (Score: -0.69)
Enter message: YOU WON A LOTTERY.
Result: SPAM (Score: -17.44)
Enter message: EXIT
Variables
Terminal
10:45 PM
Python 3