

|  |
| --- |
| **RAJALAKSHMI INSTITUTE OF TECHNOLOGY** |
| (An Autonomous Institution, Affiliated to Anna University, Chennai) |

**DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**

**ACADEMIC YEAR 2025 - 2026**

**SEMESTER III**

**ARTIFICIAL INTELLIGENCE LABORATORY**

**MINI PROJECT REPORT**

|  |  |
| --- | --- |
| **REGISTER NUMBER** | 2117240030006 |
| **NAME** | Anjali C |
| **PROJECT TITLE** | AI Quiz Master using Naive Bayes |
| **DATE OF SUBMISSION** | 29/10/2025 |
| **FACULTY IN-CHARGE** | **Mrs. M. Divya** |

**Signature of Faculty In-charge**

**INTRODUCTION**

Artificial Intelligence (AI) focuses on creating systems that can think, reason, and learn like humans. It has applications in various domains such as speech recognition, computer vision, automation, and intelligent tutoring systems. One of the emerging areas of AI is in educational tools that can evaluate a learner’s knowledge through adaptive quizzes. Traditional quiz systems rely on fixed answer matching, but AI-based approaches can analyze free-text user responses and determine their correctness intelligently. This project, titled AI Quiz Master using Naive Bayes, aims to develop a smart quiz application that evaluates user input using a probabilistic model. The system uses the Naive Bayes classifier, a concept based on conditional probability, to predict whether a user’s answer is correct or incorrect. By training on example data, the quiz master can make informed decisions even when the user’s answers are phrased differently. This demonstrates the role of AI in natural language understanding and automated assessment.

**PROBLEM STATEMENT**

To design and implement an intelligent quiz system capable of analyzing user-entered answers and classifying them as correct or incorrect using the Naive Bayes model based on probabilistic reasoning.

**GOAL**

**Expected Result:**

A functional text-based quiz system that intelligently evaluates answers using Naive Bayes classification.

**Possibilities:**

Extend the system to handle large question sets with varied topics.

Integrate voice-based inputs for speech-based answering.

Introduce adaptive learning, where questions adjust to user performance.

Use semantic analysis for better understanding of descriptive answers**.**

**THEORETICAL BACKGROUND**

Naive Bayes Overview

Naive Bayes is a probabilistic model based on Bayes’ Theorem, which provides a mathematical way to calculate the likelihood of an event occurring given prior evidence. The model assumes that the features (words in a sentence) are conditionally independent — hence the term “naive.” Despite this assumption, Naive Bayes performs remarkably well for text classification problems such as spam detection, sentiment analysis, and quiz evaluation.

Bayes’ theorem is expressed as:

P(C∣X)= P(X∣C)×P(C)/ P(X)

where:

P(C∣X) Posterior probability of class C given the input X

P(X∣C) Likelihood of input X given class C

P(C) Prior probability of the class

P(X) Probability of the input data

Application in the Project

In this project, user answers are represented as text features. The system uses TF–IDF (Term Frequency–Inverse Document Frequency) to extract word importance and applies Multinomial Naive Bayes to classify whether the user’s response matches the correct concept.

Literature Survey

* McCallum and Nigam (1998) demonstrated the power of Naive Bayes in document classification tasks.
* Raschka (2015) highlighted its efficiency on small datasets in Python Machine Learning.
* Scikit-learn (2024) offers optimized implementations for Naive Bayes models.
* GeeksforGeeks (2021) provides tutorials on text classification using Naive Bayes.
* Towards Data Science (2022) explored enhancements like smoothing and feature selection.

**ALGORITHM EXPLANATION WITH EXAMPLE**

**Naive Bayes Classification Steps**

* Collect a small training dataset of correct and incorrect sample answers for each question.
* Convert the text into numerical features using TF–IDF Vectorization.
* Train a Multinomial Naive Bayes classifier on this data.
* Accept user input as a free-text answer.
* Predict the probability that the answer is correct.
* Display feedback as “Correct” or “Incorrect” with a confidence score..

**Example**

Question: “What is the capital of France?”

* Positive samples: “Paris”, “The capital is Paris”, “City of Paris”
* Negative samples: “Berlin”, “Madrid”, “Rome”

If the user types “It’s Paris”, the classifier identifies that the word Paris strongly matches the positive class and predicts Correct with high confidence.

**IMPLEMENTATION AND CODE**

from sklearn.naive\_bayes import MultinomialNB

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.pipeline import make\_pipeline

# Build classifier

def build\_classifier(positives, negatives):

texts = positives + negatives

labels = [1]\*len(positives) + [0]\*len(negatives)

model = make\_pipeline(TfidfVectorizer(), MultinomialNB())

model.fit(texts, labels)

return model

# Main Quiz

def main():

print("WELCOME TO AI QUIZ MASTER")

name = input("Enter your name: ").strip().title()

print(f"\nHello, {name}! Let's begin your AI quiz.\n")

quiz = [

{

"q": "What does AI stand for?",

"a": "Artificial Intelligence",

"positives": ["artificial intelligence", "ai means artificial intelligence"],

"negatives": ["machine learning", "artificial information"]

},

{

"q": "Name one popular library used in Python for AI.",

"a": "TensorFlow",

"positives": ["tensorflow", "keras", "pytorch"],

"negatives": ["numpy", "pandas", "matplotlib"]

},

{

"q": "What is Machine Learning in AI?",

"a": "Training machines using data and experience",

"positives": [

"training machines using data",

"training models with experience",

"learn from data"

],

"negatives": [

"ai that plays chess",

"collecting random data",

"storing programs"

]

},

{

"q": "Which field of AI deals with understanding human language?",

"a": "Natural Language Processing",

"positives": ["natural language processing", "nlp", "human language understanding"],

"negatives": ["computer vision", "robotics", "speech synthesis"]

},

{

"q": "What is the full form of NLP in AI?",

"a": "Natural Language Processing",

"positives": ["natural language processing", "nlp full form"],

"negatives": ["neural learning process", "network layer protocol"]

}

]

score = 0

results = []

for i, q in enumerate(quiz, start=1):

clf = build\_classifier(q["positives"], q["negatives"])

print(f"Q{i}: {q['q']}")

ans = input("Your answer: ").strip().lower()

pred = clf.predict([ans])[0]

if pred == 1:

print("Correct!\n")

score += 1

results.append(f"Q{i}: Correct ({ans})")

else:

print(f"Wrong! Correct answer: {q['a']}\n")

results.append(f"Q{i}: Wrong ({ans}) -> Correct: {q['a']}")

# Summary

accuracy = (score / len(quiz)) \* 100

print("-----------------------------------------")

print(f"Quiz Completed! Final Score: {score}/{len(quiz)}")

print(f"Accuracy: {accuracy:.2f}%")

if accuracy == 100:

print("Excellent! Perfect score!")

elif accuracy >= 80:

print("Good job! Keep learning AI concepts.")

else:

print("Nice try! Review your AI basics.")

# Save results

with open("quiz\_results.txt", "w", encoding="utf-8") as f:

f.write(f"AI QUIZ RESULTS - {name}\n")

f.write("\n".join(results))

f.write(f"\n\nFinal Score: {score}/{len(quiz)} ({accuracy:.2f}%)")

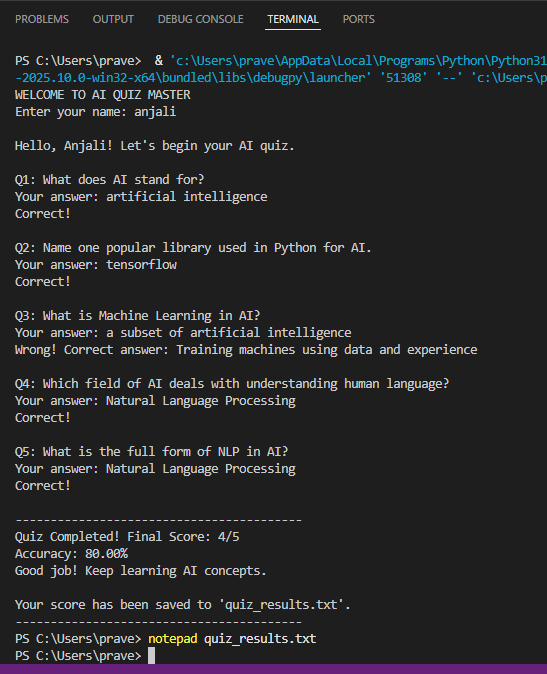
print("\nYour score has been saved to 'quiz\_results.txt'.")

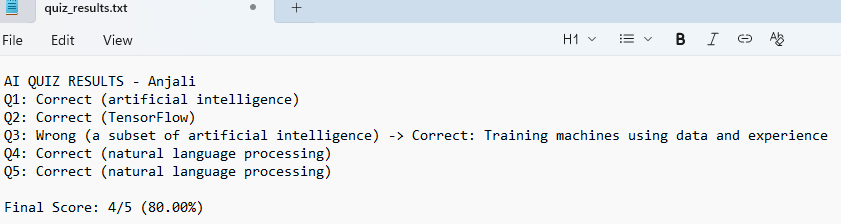
print("-----------------------------------------")

if \_\_name\_\_ == "\_\_main\_\_":

main()

OUTPUT:





**Explanation:**

When executed, the system displays each question and waits for user input. The Naive Bayes model predicts whether the input is correct and provides immediate feedback with a checkmark or cross mark.

**RESULTS AND FUTURE ENHANCEMENT**

**Results:**

Successfully implemented a text-based quiz system using Naive Bayes classification.

Demonstrates how AI can analyze human-like textual responses.

Achieved accurate detection of correct answers for simple questions.

**Future Enhancements:**

Integrate semantic similarity (using Word2Vec or BERT) for deeper understanding.

Add a GUI using Tkinter or Streamlit.

Include a question database and scoring analytics.

Deploy on the web for adaptive student assessments.

|  |  |
| --- | --- |
| **Git Hub Link of the project and report** | **https://github.com/AnjaliChakravarthi/AI-Quiz-Master** |

**REFERENCES**

* McCallum, A., & Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification.
* Raschka, S. (2015). Python Machine Learning. Packt Publishing.
* Scikit-learn Documentation: https://scikit-learn.org/stable/modules/naive\_bayes.html
* GeeksforGeeks (2021). Text Classification using Naive Bayes in Python.
* Towards Data Science (2022). Understanding Naive Bayes for NLP Tasks.