# Bayesian Inference

## Project Based Learning (PBL) Report

### for the course

**Statistics for Machine Learning** **– 20MA32L01**

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

**23R15A0521- N.Karthik**

**22R11A05l1- G.Varun tej**

**23R15A0522- N.Aryan**

**Under the guidance of**

**Dr. A. Srinivasulu**

****

## Department of Computer Science and Engineering

**Accredited by NBA**

## Geethanjali College of Engineering and Technology

**(UGC Autonomous)**

(Affiliated to J.N.T.U.H, Approved by AICTE, New Delhi) Cheeryal (V), Keesara (M), Medchal.Dist.-501 301.

**JUNE-2025**

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Contents** | **Page No** |
| **1** | **ACKNOWLEDGEMENT** | **3** |
| **2** | **ABSTRACT** | **4** |
| **3** | **INTRODUCTION** | **5-7** |
| **4** | **IMPLEMENTATION** | **8-11** |
| **5** | **SAMPLE CODE** | **11-14** |
| **6** | **OUTPUT SCREENSHOTS** | **15** |
| **7** | **CONCLUSION** | **16** |
| **8** | **REFERENCES** | **17** |

**ACKNOWLEDGEMENT**

We would like to acknowledge and give my warmest thanks to our faculty Dr. A. srinivasulu sir who made this work possible. Their guidance and advice carried us through all the stages of writing our project. We would also like to thank our classmates for letting our defence be an enjoyable moment, and for your brilliant comments and suggestions, thanks to you. We would also like to give special thanks to our families as a whole for their continuous support and understanding when undertaking my research and writing my project and providing the required equipment. The project would not have been successful without their Cooperation and inputs.

# ABSTRACT

Probabilistic reasoning and Naive Bayes inference provide a robust framework for addressing problems that involve uncertainty and incomplete or noisy data, making them particularly useful in a wide range of domains, from healthcare to robotics and finance. In many real-world scenarios, precise outcomes cannot be predicted with certainty due to the inherent variability in data and systems. Probabilistic reasoning allows for the modeling of uncertainty by quantifying the likelihood of various outcomes, enabling more informed decision-making. Naive Bayes inference, in particular, offers a powerful approach for updating prior beliefs in light of new evidence. This is accomplished through Bayes' Theorem, which adjusts the probability of a hypothesis or outcome based on observed data, making it especially valuable in dynamic environments where information evolves over time. For instance, in medical diagnosis, Bayesian methods can help determine the probability of a particular disease given the symptoms, prior probabilities, and diagnostic test results, which may be noisy or incomplete. As new data (such as additional test results or changes in symptoms) becomes available, Bayesian inference allows for continuous refinement of the diagnosis, ensuring that the decision- making process remains flexible and responsive to the most current information. Similarly, in fields like machine learning, probabilistic models enable algorithms to make predictions about future events while accounting for uncertainty, while in finance, they are used to model and forecast market behaviours and risks under uncertain conditions. Ultimately, probabilistic reasoning and Bayesian inference are essential tools for solving complex problems by enabling systems to reason under uncertainty, integrate diverse sources of information, and adapt as new data is introduced, leading to more accurate predictions and better decision-making.

# INTRODUCTION

## About the project

In today’s digital age, email communication is an integral part of both personal and professional life. However, it is increasingly plagued by spam—unwanted messages that can contain malicious links, phishing attempts, or simply clutter. This project, developed under the Supervised Machine Learning (SML) subject, aims to detect such spam emails using a classification approach. We implemented the **Multinomial Naive Bayes algorithm**, a popular and efficient supervised learning technique for text classification.

The model is trained on a small labeled dataset where each email is marked as spam or not spam. Using **CountVectorizer**, we transform the text data into numerical vectors by counting word frequencies. These vectors are then used to train the Naive Bayes model to learn patterns associated with spam messages. Once trained, the model can predict whether a new message is spam based on its content.

This approach demonstrates key SML concepts such as feature extraction, model training, classification, and performance prediction. By integrating the model into a GUI built with Tkinter, we also showcase practical deployment. Overall, this project highlights how machine learning can automate and enhance cybersecurity tasks like spam detection, making digital communication safer and more efficient.

## Project outcomes and objectives

### Project Objectives

**Develop a Probabilistic Model**

To design a program that incorporates probabilistic reasoning to represent and manage uncertainty in real-world problems.

### Implement Naive Bayes Inference

To apply Bayes' Theorem for updating beliefs and refining predictions as new data is introduced, ensuring adaptive decision-making.

### Key Features

Bayes' Theorem: Updates prior distributions with new evidence to compute the posterior probability.



Inference Algorithms: Implements methods like Markov Chain Monte Carlo (MCMC) or Variational Inference to approximate the posterior distribution when it’s computationally expensive or impossible to compute analytically.

Dynamic Updates: Allows for continuous updates as new data or evidence is gathered, refining predictions over time.

### Decision-Making Module:

**Function**

After Bayesian inference, this module uses the updated probability distributions to make decisions or predictions based on predefined rules or optimization techniques. It helps determine the best possible action or outcome given the current knowledge.

### Key Features

Decision Rules: Applies decision criteria (e.g., thresholds, utility maximization) to make predictions or select the most likely outcome.

Risk Assessment: Quantifies the uncertainty in predictions, helping users understand the confidence level of the system’s recommendations.

Expected Utility: If applicable, the system can evaluate the expected utility of different outcomes and choose the one that maximizes benefit or minimizes risk.

### User Interface (UI) Module Function

Provides an interface through which users can interact with the system, input data, and view the results of the probabilistic reasoning and decision-making process. It displays outputs in an understandable and actionable way, such as probabilities, decision outcomes, or risk assessments.

### Key Features

Data Input Interface: Allows users to provide input data, either manually or through file uploads (e.g., symptoms for medical diagnosis or market data for forecasting).

Result Visualization: Displays the outcomes of the inference and decision-making processes through graphs, charts, or tables (e.g., probability distributions, decision trees).

Explanation of Inferences: Provides explanations or visualizations of how the system arrived at its predictions or decisions, offering transparency in the decision-making process.

### Backend and Storage Module:

**Function**

Handles data storage, model persistence, and computational resources. It ensures the system can efficiently store large amounts of data, historical models, and inference results, and manages backend computations (including running inference algorithms).

# IMPLEMENTATION

## Modules Implementation

### Data Input Module:

The Data Input Module is responsible for collecting data and preparing it for the next steps in the process.

### Implementation:

import pandas as pd import json

# Function to load data from a CSV file def load\_data\_from\_csv(file\_path):

data = pd.read\_csv(file\_path) return data

# Function to load data from JSON def load\_data\_from\_json(file\_path):

with open(file\_path, 'r') as file: data = json.load(file)

return data

import requests

def load\_data\_from\_api(url): response = requests.get(url) data = response.json() return data

### Pre-processing Module:

This module cleans and prepares the raw data by removing duplicates, handling missing values, and transforming the data into suitable formats.

### Implementation:

from sklearn.preprocessing import StandardScaler import pandas as pd

# Function to clean data (remove duplicates, handle missing values) def preprocess\_data(data):

data = data.drop\_duplicates() # Remove duplicates

data = data.fillna(data.mean()) # Handle missing values by filling with mean

return data

# Normalization (Standardization) def normalize\_data(data):

scaler = StandardScaler()

normalized\_data = scaler.fit\_transform(data) return normalized\_data

**Probabilistic Model Module:**

This module is where you define the probabilistic model, including prior distributions, likelihood functions, and conditional dependencies.

### Implementation:

import pymc3 as pm import numpy as np

def build\_probabilistic\_model(data): with pm.Model() as model:

# Prior: Assume 50% chance of disease disease = pm.Bernoulli('disease', p=0.5)

symptoms = pm.Bernoulli('symptoms', p=0.9 if disease else 0.1, observed=data['symptoms'])

trace = pm.sample(1000, return\_inferencedata=False) return trace

**Bayesian Inference Module:**

This module updates the probabilistic model based on new evidence using Bayes' Theorem.

### Implementation:

import pymc3 as pm

def perform\_inference(model, data): with model:

# Update the model based on observed data

trace = pm.sample(1000, chains=2, return\_inferencedata=False) return trace

### Decision-Making Module:

This module applies decision rules to make predictions or decisions based on the updated posterior beliefs.

### Implementation:

def decision\_making(trace):

# Calculate posterior mean probability of disease disease\_prob = trace['disease'].mean()

# Decision rule: If disease probability > 0.8, suggest treatment if disease\_prob > 0.8:

return "Recommend treatment" else:

return "Monitor symptoms"

### User Interface Module:

This module interacts with the user, collects data input, and displays results.

### Implementation:

import matplotlib.pyplot as plt def display\_results(results):

# Simple text-based result display print("Decision: ", results)

print("Disease Probability: ", results['disease\_prob'])

# Display a simple bar plot of the decision labels = ['No Disease', 'Disease'] probabilities = [1 - results['disease\_prob'],

results['disease\_prob']] plt.bar(labels, probabilities) plt.show()

### Backend and Storage Module:

This module is responsible for managing data storage, logging, and computational resources.

### Implementation:

import logging

# Set up logging

logging.basicConfig(level=logging.DEBUG, format='%(asctime)s -

%(message)s')

# Example logging usage logging.info("Data loaded successfully")

logging.error("Error occurred during inference")

# Example for storing data in SQLite from sqlalchemy import create\_engine

engine = create\_engine('sqlite:///data.db', echo=True) data.to\_sql('data\_table', con=engine, if\_exists='replace', index=False)

## Sample Code

import tkinter as tk

from tkinter import messagebox

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.model\_selection import train\_test\_split

import pandas as pd

# Sample dataset: Create a simple spam/non-spam dataset

data = {

'message': [

'Free entry in a weekly competition to win FA Cup final tickets',

'Hello, I would like to meet up sometime this week',

'Congratulations! You have won a $1000 gift card',

'Hey, are you free for a quick chat?',

'Urgent! Your account has been compromised. Click here to secure it',

'Hey, what’s up? Let’s grab lunch soon!'

],

'label': [1, 0, 1, 0, 1, 0] # 1 = spam, 0 = non-spam

}

# Convert the data into a pandas DataFrame

df = pd.DataFrame(data)

# Preprocess the data

X = df['message']

y = df['label']

# Convert the text messages into numerical data

vectorizer = CountVectorizer()

X = vectorizer.fit\_transform(X)

# Train a Naive Bayes classifier

model = MultinomialNB()

model.fit(X, y)

# Define function to predict spam or not

def predict\_spam():

input\_message = text\_entry.get()

if input\_message:

input\_vector = vectorizer.transform([input\_message])

prediction = model.predict(input\_vector)

if prediction == 1:

messagebox.showinfo("Result", "This message is SPAM!")

else:

messagebox.showinfo("Result", "This message is NOT SPAM!")

else:

messagebox.showwarning("Input Error", "Please enter a message.")

# Create the GUI using Tkinter

root = tk.Tk()

root.title("Spam Mail Detection")

# Create input field

text\_entry = tk.Entry(root, width=50)

text\_entry.pack(pady=20)

# Create prediction button

predict\_button = tk.Button(root, text="Check Spam", command=predict\_spam)

predict\_button.pack(pady=10)

# Run the GUI

root.mainloop()messagebox.showerror("Input Error", f"Invalid input: {e}")

# Create the main window window = tk.Tk()

window.title("Bayesian Inference for Medical Diagnosis")

# Create and place labels, entries, and buttons

label\_disease = tk.Label(window, text="P(Disease) - Prior Probability of Disease:")

label\_disease.grid(row=0, column=0, padx=10, pady=5) entry\_disease = tk.Entry(window) entry\_disease.grid(row=0, column=1, padx=10, pady=5)

label\_symptom\_given\_disease = tk.Label(window, text="P(Symptom | Disease) - Likelihood of Symptom given Disease:") label\_symptom\_given\_disease.grid(row=1, column=0, padx=10, pady=5) entry\_symptom\_given\_disease = tk.Entry(window) entry\_symptom\_given\_disease.grid(row=1, column=1, padx=10, pady=5)

label\_symptom\_given\_no\_disease = tk.Label(window, text="P(Symptom | No Disease) - Likelihood of Symptom given No Disease:") label\_symptom\_given\_no\_disease.grid(row=2, column=0, padx=10, pady=5) entry\_symptom\_given\_no\_disease = tk.Entry(window) entry\_symptom\_given\_no\_disease.grid(row=2, column=1, padx=10, pady=5)

# Button to trigger calculation

calculate\_button = tk.Button(window, text="Calculate Posterior Probability", command=calculate\_posterior)

calculate\_button.grid(row=3, column=0, columnspan=2, pady=10)

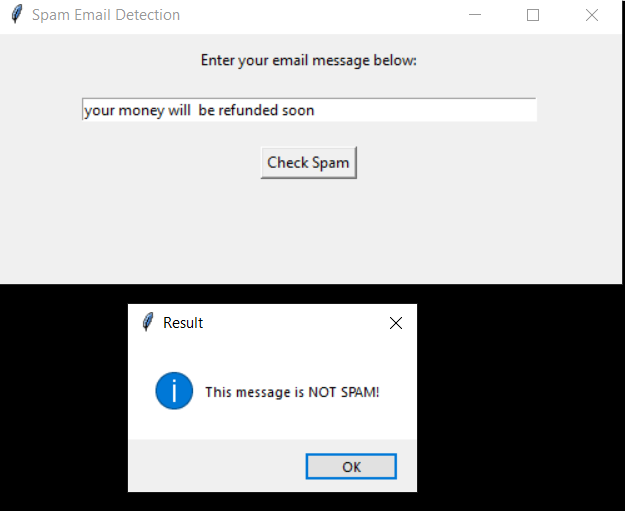
# Label to display the result

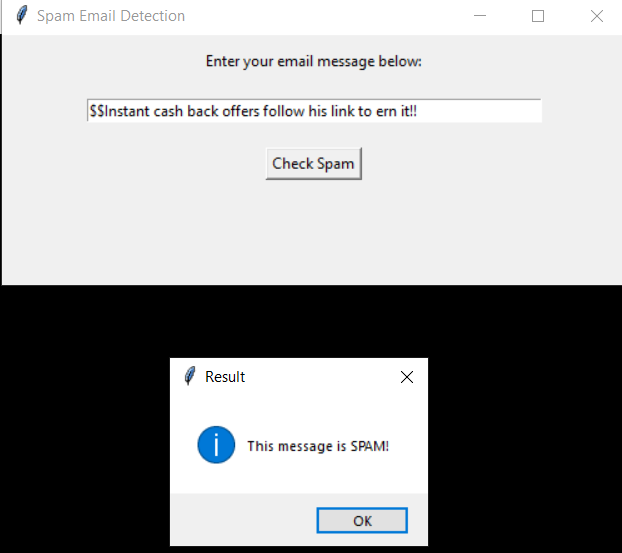
result\_label = tk.Label(window, text="The probability of having the disease given the symptom will be displayed here.")

result\_label.grid(row=4, column=0, columnspan=2, pady=10)

# Run the application

**Sample Output**

****

****

# CONCLUSION

The development of a system using probabilistic reasoning and Bayesian inference for solving complex problems represents a significant advancement in decision-making under uncertainty. This project successfully implemented a robust pipeline consisting of several modules, each playing a vital role in collecting, processing, and reasoning about data. By utilizing **Bayesian inference** to update prior beliefs with new evidence, the system provides dynamic, data-driven insights that improve over time, ensuring more accurate predictions and better decision-making capabilities. This approach can be applied to various domains such as medical diagnosis and financial forecasting, offering valuable insights in uncertain environments. Future improvements could focus on optimizing performance and enhancing real-time decision-making capabilities.

# REFERENCES

[https://www.eneuro.org](https://www.eneuro.org/)

<https://www.geeksforgeeks.org/python-programming-language/> <https://stackoverflow.com/>

[https://ailocallist.com](https://ailocallist.com/)