Assignment 7
Probabilistic Models

Session: 2021 – 2025

Submitted by:

Muhammad Umair Shahid

2021-CS-144

Supervised by:

Dr. Khaldoon Syed Khurshid

Department of Computer Science

University of Engineering and Technology Lahore

Pakistan

Contents

1 Abstraction

This report presents a comprehensive analysis and implementation details of a smartphone camera quality evaluation system. The system leverages Bayesian inference methods and a belief network approach to rank smartphones based on user-defined criteria such as minimum user rating and minimum expert rating. The system is implemented in Python and uses a CSV file as a data source.

In this report, we will discuss the system's overall architecture, the data processing pipeline, the inference models, the user interface, and real-world scenarios where such a system could be applied. We will also provide visual aids, sample outputs, code listings (formatted using the `minted` package), and a simple Data Flow Diagram (DFD) to enhance understanding.

2 Introduction

Modern consumers often rely on reviews and ratings to select smartphones with high-quality cameras. With a large number of devices on the market, it becomes challenging to determine which smartphone best meets certain camera performance criteria. This system aims to simplify that decision by:

- Reading a dataset of smartphones and their camera attributes (e.g., megapixels, aperture, optical zoom) along with user and expert ratings.
- Allowing users to input minimum acceptable ratings.
- Applying Bayesian inference and a belief network-based approach to rank smartphones based on these criteria.
- Providing a user-friendly GUI for interactive exploration of the results.

3 Real-World Scenarios

This system can be employed by:

1. **Retailers and Online Marketplaces:** Integrating this tool to help customers quickly find and compare phones that meet their desired camera quality criteria.
2. **Tech Review Websites:** Streamlining the process of listing recommended devices under certain conditions, e.g., "best smartphones with at least 4.5 user rating and professional ratings above 4.0."
3. **Research Analysts:** Using the inference approach to identify trends in the smartphone market, such as which attributes most influence expert ratings.

4 Dataset and Attributes

The dataset is stored in a CSV file. Each row corresponds to a smartphone with the following attributes:

- `Model`: The smartphone model name.
- `Megapixels`: Camera resolution in megapixels.
- `Aperture`: The lens aperture value (a lower value indicates a wider aperture, generally better low-light performance).
- `Optical Zoom`: The optical zoom capability of the camera.
- `User Rating`: The average rating given by general consumers (0 to 5).
- `Expert Rating`: The rating assigned by professional reviewers (0 to 5).

4.1 Example CSV Data

Below is a snippet of the CSV data file, illustrating the format:

```
Model,Megapixels,Aperture,Optical Zoom,User Rating,Expert
  ↪ Rating
Galaxy Ultra Pro,108,1.9,5,4.9,4.9
Pixel Pro Max,48,2.0,3,4.8,4.7
OnePlus Zoom Lite,16,1.6,4,4.0,3.8
```

5 System Components and Libraries Used

- **Python 3**: The main programming language.
- **tkinter and ttk**: Standard Python libraries for building the Graphical User Interface (GUI). They enable creating windows, buttons, labels, and tables easily.
- **csv**: A standard Python library for reading and parsing CSV files, allowing the system to load smartphone data.hting and line numbering for code blocks.

6 Inference Logic

The system uses Bayesian inference to determine the probability that a phone is "relevant" given that it meets certain query criteria (user rating and expert rating). We define:

$$P(R|Q) = \frac{P(Q|R) * P(R)}{P(Q)}$$

Where:

- R: Relevance event.
- Q: Query event (phone meets the user's rating criteria).
- $P(R)$: Prior probability of relevance, assumed here as 0.5 for simplicity.
- $P(Q)$: Probability that a randomly chosen phone meets the criteria.
- $P(Q|R)$: Probability that a phone meets criteria if it is relevant.

7 Belief Network Integration

We extend the model to:

$$P(R|Q, D) = \frac{P(Q|D) * P(R|D) * P(D)}{P(Q)}$$

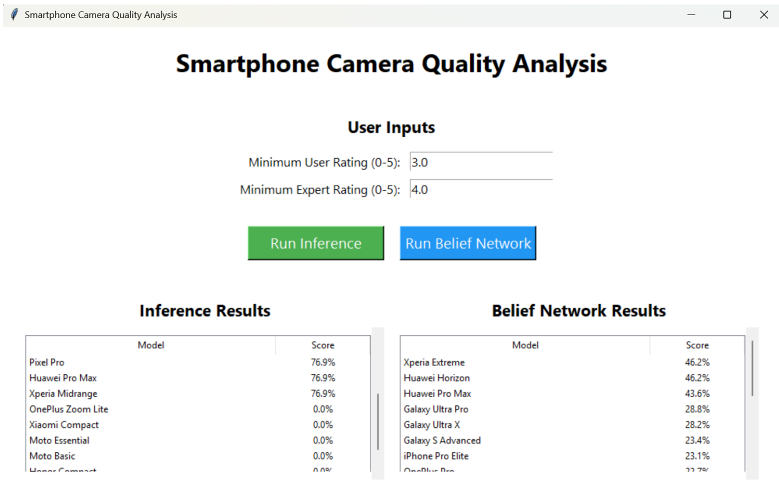
Where D represents the document (phone) itself. We incorporate camera attributes into $P(R|D)$ using a heuristic:

$$P(R|D) = (0.4 \times \frac{\text{Megapixels}}{200}) + (0.3 \times \frac{1}{\text{Aperture}})$$

8 User Interface (UI)

The UI is built using tkinter. The interface:

- Allows users to input minimum user and expert ratings.
- Provides "Run Inference" and "Run Belief Network" buttons.
- Displays results in two separate tables (tree views) for easy comparison.



The UI is designed to be user-friendly, with clear labels, default values, and immediate updates to the inference results whenever the user changes the input criteria.

9 Code Implementation

Below is the main Python code for the system. It consists of:

- Loading data from CSV.
- Computing probabilities.
- Implementing inference and belief network functions.
- Setting up the GUI.

```python
import tkinter as tk
from tkinter import ttk
import csv

# Load data from CSV file
def load_data(file_path):
    smartphones = []
    with open(file_path, mode='r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            smartphones.append({
                "Model": row["Model"],
                "Megapixels": float(row["Megapixels"]),
                "Aperture": float(row["Aperture"]),
                "Optical Zoom": float(row["Optical Zoom"]),
```

```python
                    "User Rating": float(row["User Rating"]),
                    "Expert Rating": float(row["Expert Rating"]),
                })
    return smartphones

def compute_PQ(smartphones, user_rating, expert_rating):
    count = sum(1 for phone in smartphones if phone["User Rating"] >= user_rating
                and phone["Expert Rating"] >= expert_rating)
    if len(smartphones) == 0:
        return 0.0001
    return count / len(smartphones)

def infer_camera_quality(smartphones, user_rating, expert_rating):
    P_R = 0.5
    P_Q = compute_PQ(smartphones, user_rating, expert_rating)
    if P_Q == 0:
        P_Q = 0.0001

    results = []
    for phone in smartphones:
        meets_criteria = (phone["User Rating"] >= user_rating) and \
                         (phone["Expert Rating"] >= expert_rating)
        P_Q_given_R = 1.0 if meets_criteria else 0.0
        P_R_given_Q = (P_Q_given_R * P_R) / P_Q
        results.append((phone["Model"], P_R_given_Q))

    return sorted(results, key=lambda x: x[1], reverse=True)

def belief_network(smartphones, user_rating, expert_rating):
    P_D = 0.5
    P_Q = compute_PQ(smartphones, user_rating, expert_rating)
    if P_Q == 0:
        P_Q = 0.0001

    results = []
    for phone in smartphones:
```

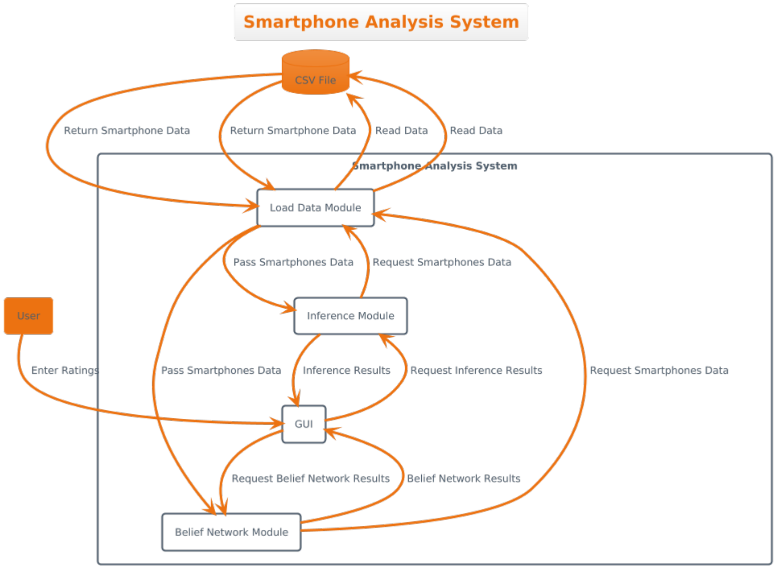
```python
        P_R_given_D = (0.4 * (phone["Megapixels"] / 200)) + \
                      (0.3 * (1 / phone["Aperture"]))
        P_R_given_D = max(0, min(1, P_R_given_D))

        meets_criteria = (phone["User Rating"] >= user_rating) and \
                         (phone["Expert Rating"] >= expert_rating)
        P_Q_given_D = 1.0 if meets_criteria else 0.0

        P_R_given_Q_and_D = (P_Q_given_D * P_R_given_D * P_D) / P_Q
        results.append((phone["Model"], P_R_given_Q_and_D))

    return sorted(results, key=lambda x: x[1], reverse=True)

def run_inference():
    user_rating_input = float(user_rating_var.get())
    expert_rating_input = float(expert_rating_var.get())
    results = infer_camera_quality(smartphones, user_rating_input, expert_rating_input)
    update_results(inference_tree, results)

def run_belief_network():
    user_rating_input = float(user_rating_var.get())
    expert_rating_input = float(expert_rating_var.get())
    results = belief_network(smartphones, user_rating_input, expert_rating_input)
    update_results(belief_tree, results)

def update_results(tree, results):
    for row in tree.get_children():
        tree.delete(row)
    for model, score in results:
        tree.insert("", "end", values=(model, f"{score*100:.1f}%"))

# Load data
file_path = "Smartphones.csv" # Update with actual path
smartphones = load_data(file_path)
```

```python

# GUI Setup
root = tk.Tk()
root.title("Smartphone Camera Quality Analysis")
root.geometry("1000x600")
root.configure(bg="#ffffff")

# Header
header = tk.Label(root, text="Smartphone Camera Quality Analysis",
                   font=("Segoe UI", 24, "bold"), pady=20, bg="#ffffff")
header.pack()

# Input Section
input_frame = tk.Frame(root, bg="#ffffff")
input_frame.pack(pady=10)

tk.Label(input_frame, text="User Inputs", font=("Segoe UI", 16, "bold"),
         bg="#ffffff").grid(row=0, column=0, columnspan=2, pady=10)

# User Rating
tk.Label(input_frame, text="Minimum User Rating (0-5):", font=("Segoe UI", 12),
         bg="#ffffff").grid(row=1, column=0, sticky="e", padx=10)
user_rating_var = tk.StringVar(value="4.0")
tk.Entry(input_frame, textvariable=user_rating_var, font=("Segoe UI", 12)).grid(row=1, column=1, pady=5)

# Expert Rating
tk.Label(input_frame, text="Minimum Expert Rating (0-5):", font=("Segoe UI", 12),
         bg="#ffffff").grid(row=2, column=0, sticky="e", padx=10)
expert_rating_var = tk.StringVar(value="4.0")
tk.Entry(input_frame, textvariable=expert_rating_var, font=("Segoe UI", 12)).grid(row=2, column=1, pady=5)
```

```python

user_rating_var.trace("w", lambda *args: run_inference())
expert_rating_var.trace("w", lambda *args: run_inference())

button_frame = tk.Frame(root, bg="#ffffff")
button_frame.pack(pady=10)

tk.Button(button_frame, text="Run Inference", command=run_inference, bg="#4CAF50",
          fg="white", font=("Segoe UI", 14), width=15).grid(row=0, column=0, padx=10, pady=10)
tk.Button(button_frame, text="Run Belief Network", command=run_belief_network, bg="#2196F3",
          fg="white", font=("Segoe UI", 14), width=15).grid(row=0, column=1, padx=10, pady=10)

# Results Section
result_frame = tk.Frame(root, bg="#ffffff")
result_frame.pack(pady=20, fill="both", expand=True)

result_inner_frame = tk.Frame(result_frame, bg="#ffffff")
result_inner_frame.pack(side="top", fill="x", padx=20)

# Inference Results
inference_frame = tk.Frame(result_inner_frame, bg="#ffffff")
inference_frame.pack(side="left", fill="both", expand=True, padx=10)

inference_label = tk.Label(inference_frame, text="Inference Results",
                           font=("Segoe UI", 16, "bold"), bg="#ffffff")
inference_label.pack(anchor="n", pady=5)

inference_tree = ttk.Treeview(inference_frame, columns=("Model", "Score"), show="headings", height=10)
inference_tree.heading("Model", text="Model")
inference_tree.heading("Score", text="Score")
inference_tree.column("Model", width=300, anchor="w")
inference_tree.column("Score", width=100, anchor="center")
```

```python
scrollbar = ttk.Scrollbar(inference_frame, orient="vertical", command=inference_tree.yview)
inference_tree.configure(yscrollcommand=scrollbar.set)
inference_tree.pack(side="left", fill="both", expand=True, pady=10)
scrollbar.pack(side="left", fill="y")

# Belief Network Results
belief_frame = tk.Frame(result_inner_frame, bg="#ffffff")
belief_frame.pack(side="left", fill="both", expand=True, padx=10)

belief_label = tk.Label(belief_frame, text="Belief Network Results",
                        font=("Segoe UI", 16, "bold"), bg="#ffffff")
belief_label.pack(anchor="n", pady=5)

belief_tree = ttk.Treeview(belief_frame, columns=("Model", "Score"), show="headings", height=10)
belief_tree.heading("Model", text="Model")
belief_tree.heading("Score", text="Score")
belief_tree.column("Model", width=300, anchor="w")
belief_tree.column("Score", width=100, anchor="center")
scrollbar2 = ttk.Scrollbar(belief_frame, orient="vertical", command=belief_tree.yview)
belief_tree.configure(yscrollcommand=scrollbar2.set)
belief_tree.pack(side="left", fill="both", expand=True, pady=10)
scrollbar2.pack(side="left", fill="y")

root.mainloop()
```

10 DFD



11 Discussion

The presented system is a simplified model but offers an insightful approach to filtering and ranking smartphones. By adjusting criteria, users can quickly see how the Bayesian inference changes the relevance scores. Advanced techniques could incorporate probabilistic distributions or weight attributes differently.

12 Conclusion

We have explained the architecture, dataset, Bayesian inference model, belief network extension, and GUI of a smartphone camera quality analysis tool. The combination of code, user-friendly interface, and theoretical underpinnings provides a robust platform for informed decision-making.