TECHNICAL REPORT

LAPTOP PRICE PREDICTION SYSTEM

MACHINE LEARNING POWERED PRICE ESTIMATION

PRESENTED BY KARAR DAOOD SALMAN
NETWORKS GB
MORNING STUDY

HOW IT WORKS ?

Step 1 – Input Specs: The user enters CPU, GPU, RAM, storage, screen, and brand in the GUI
Step 2 – Feature Conversion: The system converts these inputs into numerical scores and calculates RAM_GPU, CPU_GPU, and Total_Perf
Step 3 – Prediction: The Gradient Boosting model predicts the laptop price based on the processed features
Step 4 – Output: The predicted price is displayed instantly in the GUI

LET SEE HOW IS THAT WORKS IN THE CODE IN THE NEXT PAGES

SECTION 1 : Imports and Setup

- IN THIS SECTION WE REQUIRE LIBRARIES FOR (GUI), (DATA PROCESSING), AN (MACHINE LEARNING). (SUPPRESSES WARNINGS)

```
import tkinter as tk
from tkinter import ttk, messagebox
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

SECTION 2 : Constants and Scoring Functions

- IN THIS SECTION, WE DEFINE PERFORMANCE SCORES FOR GPU, CPU, BRAND, SCREEN, AND RAM TYPES, AS WELL AS GUI COLOR SETTINGS. FUNCTIONS ARE CREATED TO CONVERT INPUTS INTO NUMERICAL VALUES FOR PREDICTION

```
GPU_SCORES = {"RTX4090":100,"RTX4080":90,"RTX4070TI":82,"RTX4070":75,"RTX4060":65,"RTX3080":70,"RTX3070":62
              ,"RTX3060TI":58,"RTX3060":52,"RTX3050TI":45,"RTX3050":40,"RTX2070":55,"GTX1660TI":38
              ,"GTX1660":35,"GTX1650":28,"GTX1050TI":22,"INTEGRATED":10}
CPU_SCORES = {"I9":100,"RYZEN 9":95,"I7":80,"RYZEN 7":75,"I5":60,"RYZEN 5":58,"I3":40,"RYZEN 3":38}
BRAND_SCORES = {"ALIENWARE":10,"RAZER":9,"MSI":7,"ASUS":7,"HP":6,"DELL":6,"LENOVO":6,"SAMSUNG":5,"GIGABYTE":5,"ACER":4}
SCREEN_MAP = {"720P":1,"1080P":2,"1440P":3,"4K":4,"8K":5}
RAM_MAP = {"DDR3":1,"DDR4":2,"DDR5":3}
COLORS = {"bg":"#ffffff","card":"#f8f9fa","accent":"#2c3e50","text":"#212529","border":"#dee2e6"}

# Scoring functions
score_gpu = lambda g: next((v for k,v in GPU_SCORES.items() if k in str(g).upper().replace(" ","")), 15)
score_cpu = lambda c: (lambda s: next((v for k,v in CPU_SCORES.items() if k in s), 30) + (next((i-5)*2 for i in range(13,5,-1) if str(i) in s) if any(str(i) in s for i in range(13,5,-1)) else 0))(str(c).upper())
extract_num = lambda v: (lambda n: n * (1024 if "TB" in str(v).upper() else 1))(float(str(v).upper().replace("GB","").replace("TB","").strip())) if v else np.nan
```

SECTION 3 : Load and Prepare Dataset

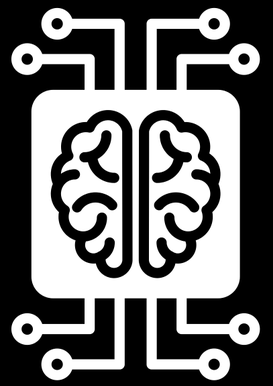
- IN THIS SECTION, WE LOAD THE LAPTOP DATASET, CLEAN MISSING VALUES, AND CONVERT CATEGORICAL FEATURES TO NUMERICAL SCORES. WE ALSO CALCULATE DERIVED FEATURES LIKE RAM_GPU, CPU_GPU, AND TOTAL_PERF

```
df = pd.read_csv("laptop_dataset_5000.csv")
df = df[pd.to_numeric(df["Price"], errors="coerce").between(300, 5000)].dropna(subset=["Price"])
df[["RAM","Storage"]] = df[["RAM","Storage"]].applymap(extract_num)
df = df.dropna(subset=["RAM","Storage"])
df["Screen_Quality"] = df["Screen"].str.upper().map(SCREEN_MAP).dropna()
df = df.dropna(subset=["Screen_Quality"])
df[["GPU_Score","CPU_Score"]] = df[["GPU","CPU"]].apply(lambda col:
                                col.map(score_gpu if col.name=="GPU" else score_cpu))
df["RAM_Type_Score"] = df["RAM_Type"].map(RAM_MAP).dropna()
df = df.dropna(subset=["RAM_Type_Score"])
df["Brand_Score"] = df["Brand"].str.upper().map(BRAND_SCORES).fillna(5)
df["RAM_GPU"] = df["RAM"] * df["GPU_Score"] / 100
df["CPU_GPU"] = df["CPU_Score"] * df["GPU_Score"] / 100
df["Total_Perf"] = (df["CPU_Score"] + df["GPU_Score"]) / 2
```

SECTION 4 : Train Gradient Boosting Model

- IN THIS SECTION, WE SPLIT THE DATA INTO TRAINING AND TESTING SETS, STANDARDIZE THE FEATURES, AND TRAIN A GRADIENT BOOSTING REGRESSOR MODEL TO PREDICT LAPTOP PRICES. MODEL ACCURACY AND MEAN ABSOLUTE ERROR ARE CALCULATED

```
model = GradientBoostingRegressor(
    n_estimators=300, max_depth=6,learning_rate=0.05,subsample=0.85,
    min_samples_split=3,
    min_samples_leaf=1,
    random_state=42
)
model.fit(X_train_s, y_train)
score = model.score(X_test_s, y_test)
mae = np.mean(np.abs(y_test - model.predict(X_test_s)))
```



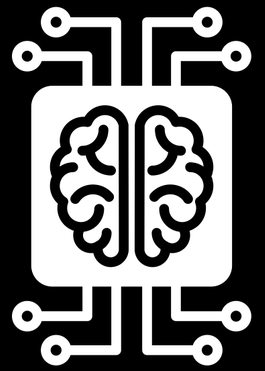
SECTION 5 : GUI Setup

- IN THIS SECTION, WE INITIALIZE THE MAIN TKINTER GUI WINDOW, CONFIGURE ITS SIZE, BACKGROUND COLOR, AND DEFINE STYLES FOR INPUT COMBOBOXES

```
root = tk.Tk()
root.title("Laptop Price Predictor")
root.geometry("650x750")
root.configure(bg=COLORS["bg"])
entries = {}

style = ttk.Style()
style.theme_use('clam')
style.configure("C.TCombobox", fieldbackground="white", background=COLORS["card"], foreground="black", borderwidth=1)
```

SECTION 6 : Input Fields

- IN THIS SECTION, INPUT FIELDS ARE CREATED FOR CPU, RAM, RAM TYPE, GPU, STORAGE, SCREEN, AND BRAND USING COMBOBOXES. USERS CAN SELECT THEIR LAPTOP SPECIFICATIONS FROM THESE FIELDS



```
# Input fields
for lbl, key, vals in [
    ("CPU","CPU",unique_vals["CPU"]),
    ("RAM","RAM",["4GB","8GB","16GB","32GB","64GB"]),
    ("RAM Type","Type",["DDR3","DDR4","DDR5"]),
    ("GPU","GPU",unique_vals["GPU"]),
    ("Storage","Storage",["128GB","256GB","512GB","1TB","2TB","3TB"]),
    ("Screen","Screen",["720p","1080p","1440p","4K","8K"]),
    ("Brand","Brand",unique_vals["Brand"])
]:
    card = tk.Frame(main, bg=COLORS["card"], relief=tk.SOLID, bd=1, highlightbackground=COLORS["border"])
    card.pack(fill=tk.X, pady=6)
    inner = tk.Frame(card, bg=COLORS["card"])
    inner.pack(fill=tk.X, padx=15, pady=10)
    tk.Label(inner, text=lbl+":", font=("Segoe UI",10), bg=COLORS["card"], fg=COLORS["text"], width=12).pack(side=tk.LEFT)
    combo = ttk.Combobox(inner, values=vals, width=35, font=("Segoe UI",9), style="C.TCombobox", state="readonly")
    combo.pack(side=tk.RIGHT)
    combo.set(vals[len(vals)//2])
    entries[key] = combo
```

SECTION 7 : Prediction Function

- IN THIS SECTION, THE PREDICT() FUNCTION COLLECTS USER INPUT, CONVERTS IT TO NUMERICAL FEATURES, CALCULATES ADDITIONAL FEATURES, PREDICTS THE LAPTOP PRICE USING THE TRAINED MODEL, AND STORES RESULTS IN A CACHE

```
def predict():
    try:
        key = tuple(e.get() for e in entries.values())
        if key in cache:
            show_result(cache[key])
            return

        vals = [
            score_cpu(entries["CPU"].get()),
            float(entries["RAM"].get().replace("GB","")),
            RAM_MAP[entries["Type"].get()],
            score_gpu(entries["GPU"].get()),
            extract_num(entries["Storage"].get()),
            SCREEN_MAP[entries["Screen"].get().upper()],
            BRAND_SCORES.get(entries["Brand"].get().upper(),5)
        ]

        vals.extend([
            vals[1]*vals[3]/100,
            vals[0]*vals[3]/100,
            (vals[0]+vals[3])/2
        ])

        pred = model.predict(scaler.transform([vals]))[0]
        cache[key] = pred
        show_result(pred)
    except Exception as e:
        messagebox.showerror("Error", str(e))
```

SECTION 8 : Display Result

- IN THIS SECTION, THE SHOW_RESULT() FUNCTION CREATES A POPUP WINDOW TO DISPLAY THE ESTIMATED LAPTOP PRICE, ALONG WITH ACCURACY, AVERAGE ERROR, AND DATASET SIZE

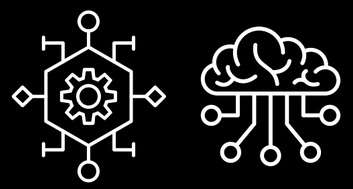
```python
def show_result(pred):
    w = tk.Toplevel(root)
    w.title("Price Prediction")
    w.geometry("400x300")
    w.configure(bg=COLORS["bg"])
    w.resizable(False, False)
    w.transient(root)
    w.grab_set()

    tk.Label(w, text="ESTIMATED PRICE", font=("Segoe UI",12,"bold"), bg=COLORS["bg"], fg=COLORS["text"]).pack(pady=(30,10))

    pf = tk.Frame(w, bg=COLORS["card"], relief=tk.SOLID, bd=2)
    pf.pack(pady=10, padx=40, fill=tk.X)
    tk.Label(pf, text=f"${pred:,.2f}", font=("Segoe UI",32,"bold"), bg=COLORS["card"], fg=COLORS["text"]).pack(pady=20)

    tk.Label(w, text=f"Accuracy: {score*100:.1f}%  |  Avg Error: ±${mae:.0f}", font=("Segoe UI",9), bg=COLORS["bg"], fg="#6c757d").pack(pady=10)
    tk.Label(w, text=f"Dataset: {len(df)} laptops", font=("Segoe UI",8), bg=COLORS["bg"], fg="#6c757d").pack()

    tk.Button(w, text="Close", command=w.destroy, bg=COLORS["accent"], fg="white", font=("Segoe UI",10,"bold"), relief=tk.FLAT, padx=30, pady=8).pack(pady=20)
```

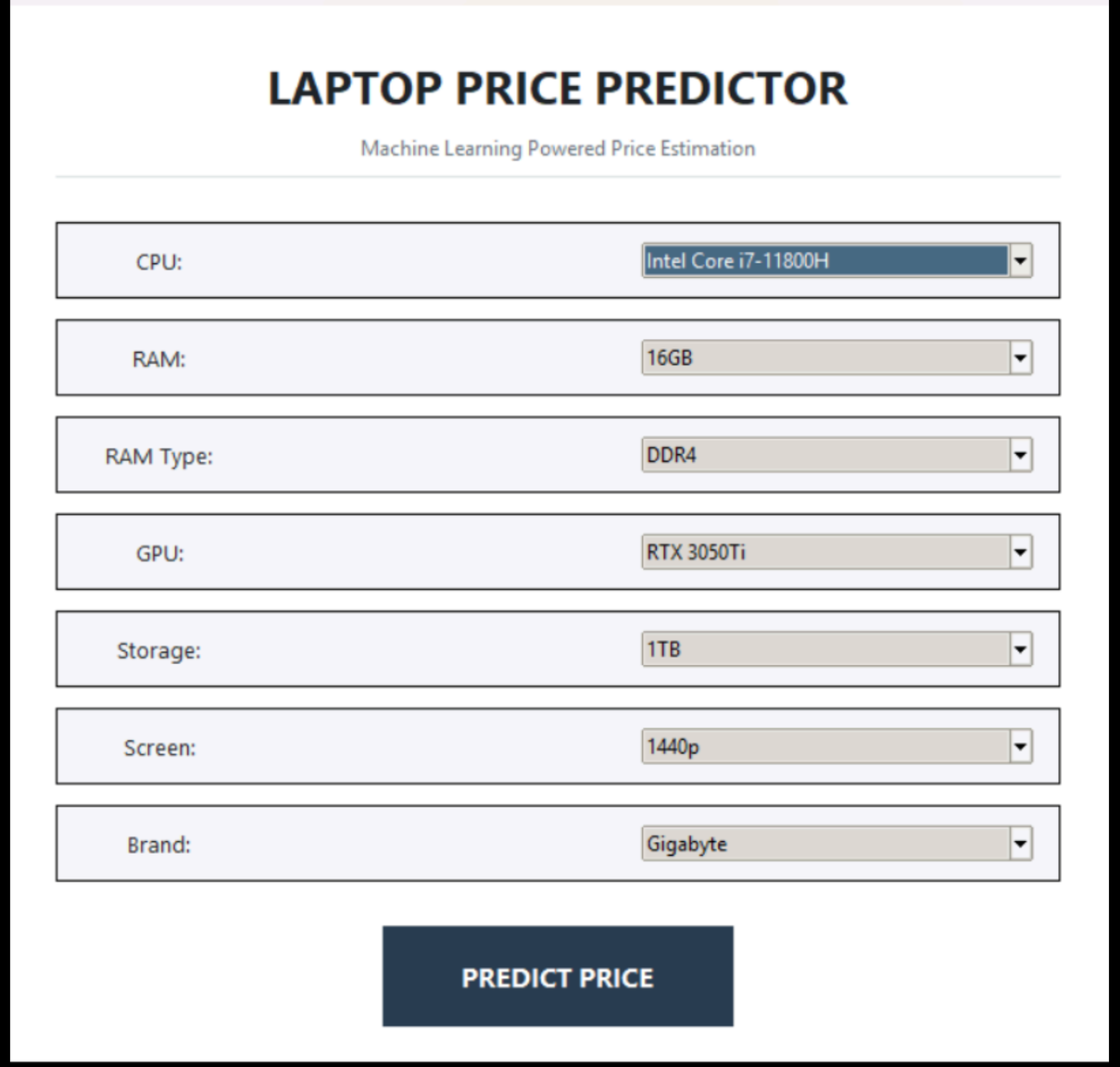
SECTION 9 : Prediction Button & Main Loop

- IN THIS SECTION, A BUTTON IS ADDED TO THE MAIN WINDOW TO TRIGGER THE PREDICTION. THE TKINTER EVENT LOOP IS STARTED TO RUN THE GUI

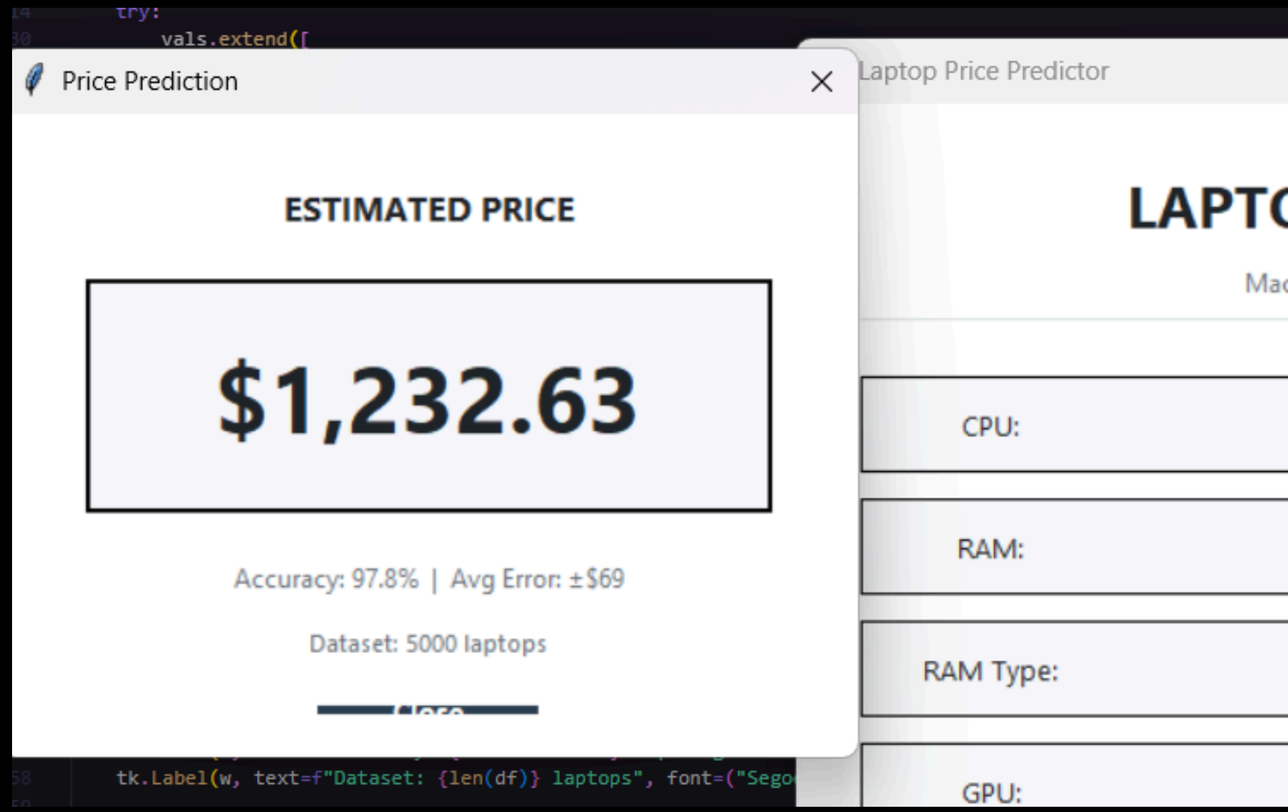
```
btn = tk.Button(main, text="PREDICT PRICE", command=predict, bg=COLORS["accent"], fg="white", font=("Segoe UI",12,"bold"), relief=tk.FLAT, padx=40, pady=12)
btn.pack(pady=20)

root.mainloop()
```

- when we run the program the gui window shows up to the user and let him to choose the labtop parts that he want to check about it price , AND AFTER HE CHOOSE HIS PARTS HE MUST PRESS (PREDICT PRICE)





- AFTER THE USER PRESS THE PREDICT BUTTON NEW WINDOW SHOWS UP WITH THE RESULTS AND TWO OTHER THINGS



THE TWO OTHER THINGS IS

Accuracy (R^2 Score)

Accuracy shows how well the model understands the relationship between laptop specs
.and price

.A higher R^2 means the model makes predictions that closely match real prices

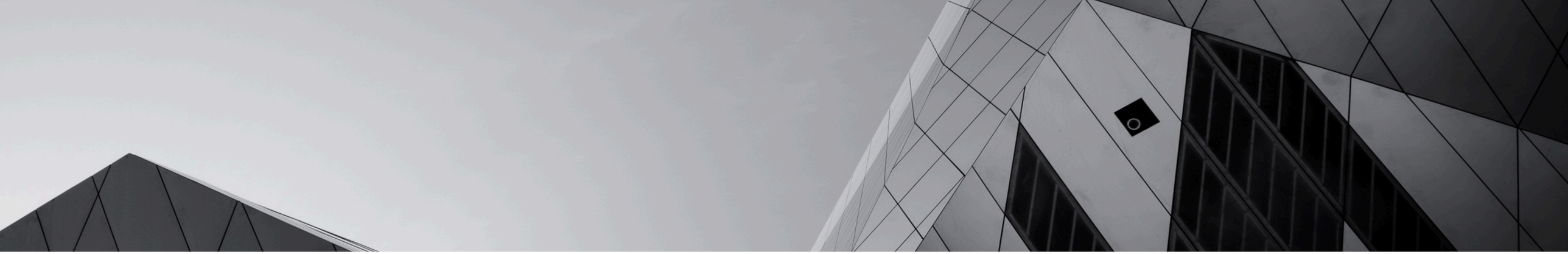
Average Error (MAE)

.MAE shows how much the model is wrong on average in dollars

.A lower MAE means the predictions are more precise and closer to the actual price

CONCLUSION

This project successfully demonstrates how machine learning can predict laptop prices based on hardware specifications. Using Gradient Boosting and engineered features, the system achieved high accuracy with low prediction error. The final model provides fast, reliable price estimates and can be expanded in the future with larger datasets and more advanced techniques



THANK YOU

Thank you for your time and attention throughout this presentation. We look forward to your feedback and continued collaboration.