

Project Report

1. Detect Heart Disease using Patient Data

- **Objective:**

Build a system that can predict if a patient has heart disease. Explore the data, understand the features, and figure out an approach.

1. Introduction:

Heart disease is a leading cause of death worldwide. Early detection of individuals at risk can significantly reduce mortality rates through timely intervention. This project presents a machine learning approach to detect the presence of heart disease based on patient data. The primary goal is to build a predictive model using classification algorithms to classify patients as at risk or not at risk of heart disease.

2. Dataset Overview:

The dataset used in this project is the **heart disease** dataset, typically containing the following features:

- Age
- Sex
- Chest pain type
- Resting blood pressure
- Cholesterol level
- Fasting blood sugar

- Resting ECG results
- Maximum heart rate achieved
- Exercise-induced angina
- ST depression induced by exercise
- Slope of the peak exercise ST segment
- **Target:** 1 indicates presence of heart disease, 0 indicates absence

The dataset was loaded using the pandas library and checked for missing values. No missing values were found, so further imputation was unnecessary.

3. Methodology:

a. Data Preprocessing

- The feature columns were standardized using **StandardScaler** to bring all attributes to a similar scale.
- The target column (target) was separated from the features.
- The dataset was split into **training (80%)** and **testing (20%)** sets using `train_test_split` with stratification on the target variable.

b. Model Training

Two models were trained and evaluated:

1. Logistic Regression

- A simple, interpretable linear model commonly used for binary classification.
- Achieved a test accuracy of approximately **0.84**.

2. Random Forest Classifier

- An ensemble method that builds multiple decision trees and merges their output.
- Achieved a test accuracy of approximately **0.93**.

c. Model Evaluation

- The better performing model (selected based on accuracy) was further evaluated using:
 - **Accuracy Score**
 - **Classification Report** (Precision, Recall, F1-score)
 - **Confusion Matrix**, visualized using a heatmap

d. Prediction on New Data

- Two hypothetical patient records were created with known features.
- The same scaler used during training was applied to transform the new data.
- Predictions were made using the best model from earlier evaluation.

4. Results:

- The **Random Forest model** (or Logistic Regression, based on which had higher accuracy) showed better performance.
- The classification report revealed class-wise performance, helping assess false positives and negatives.
- The **confusion matrix heatmap** visually demonstrated the distribution of predictions.
- The model was able to make meaningful predictions on unseen data, indicating good generalization.

Example Output:

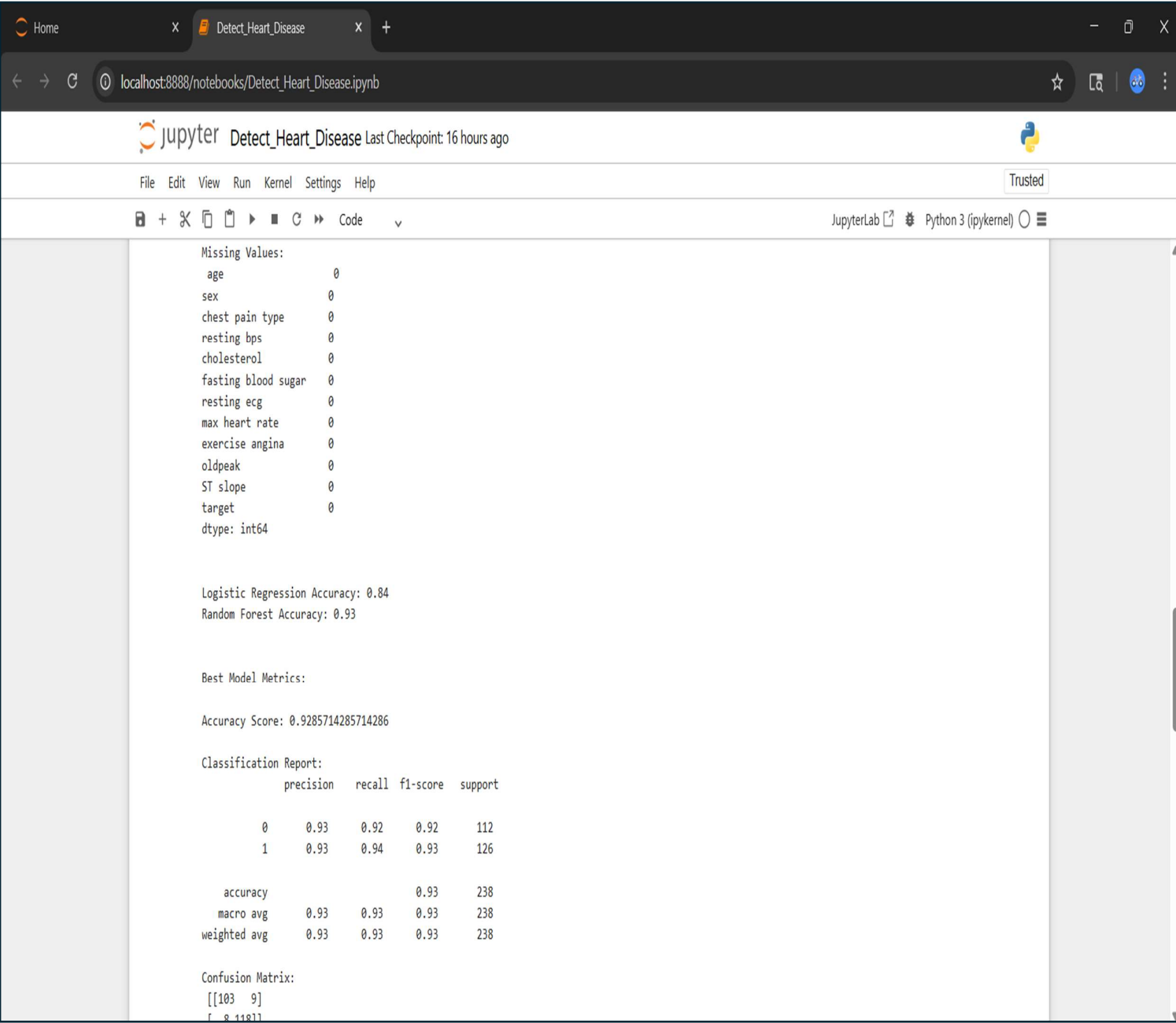
Prediction for New Data Row 1: No Heart Disease...

Prediction for New Data Row 2: At Risk of Heart Disease!

5. Conclusion:

This project demonstrates the effectiveness of machine learning techniques, particularly Random Forests and Logistic Regression, in detecting heart disease based on structured patient data. While Logistic Regression provides interpretability, Random Forests offer higher predictive performance.

6. Screenshots of Results:



```
Missing Values:
age          0
sex          0
chest pain type  0
resting bps   0
cholesterol   0
fasting blood sugar  0
resting ecg   0
max heart rate  0
exercise angina  0
oldpeak      0
ST slope     0
target       0
dtype: int64

Logistic Regression Accuracy: 0.84
Random Forest Accuracy: 0.93

Best Model Metrics:

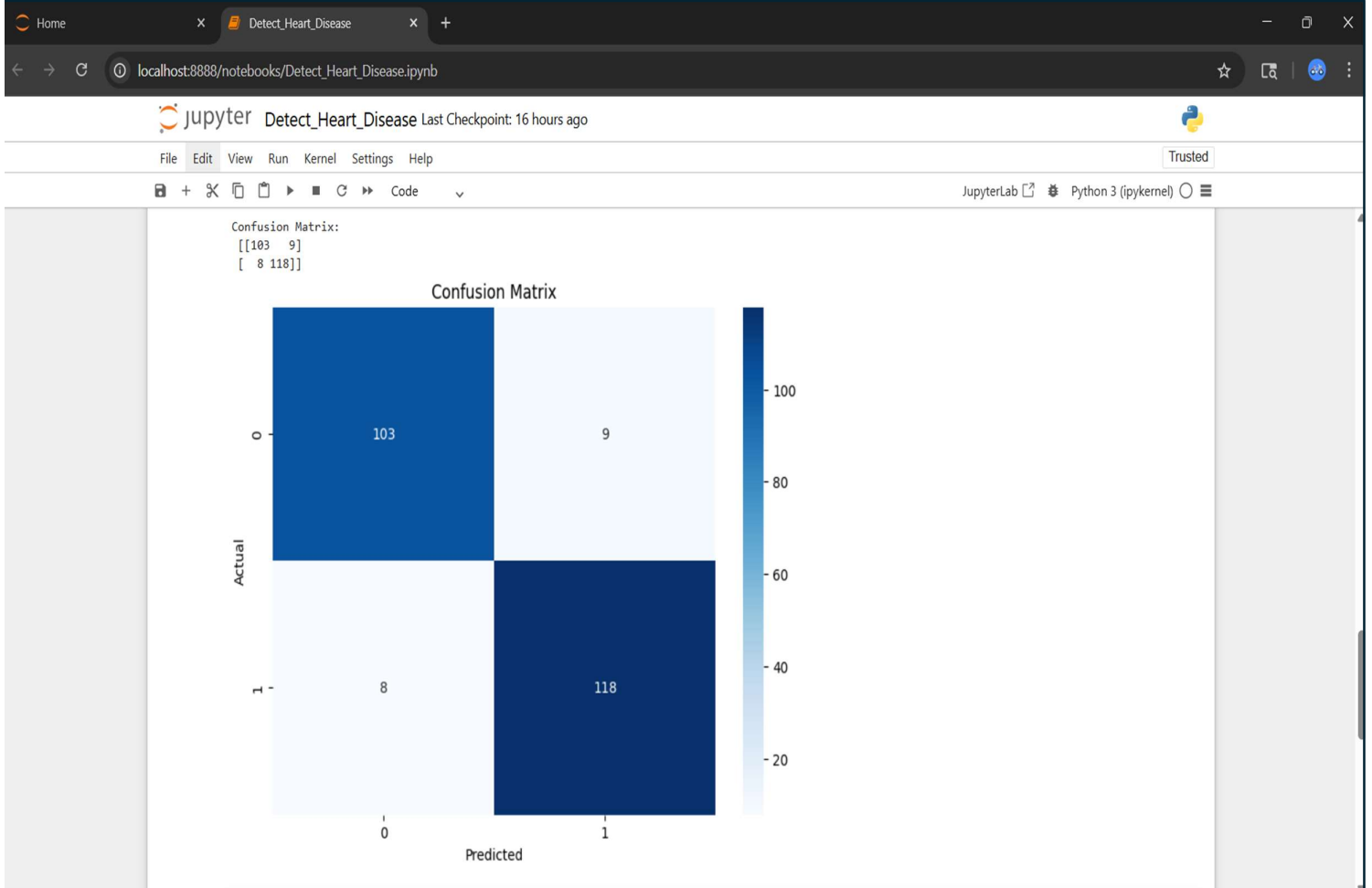
Accuracy Score: 0.9285714285714286

Classification Report:
              precision    recall  f1-score   support

    0       0.93      0.92      0.92       112
    1       0.93      0.94      0.93       126

   accuracy          0.93      0.93      0.93       238
  macro avg       0.93      0.93      0.93       238
weighted avg       0.93      0.93      0.93       238

Confusion Matrix:
[[103   9]
 [ 8 112]]
```



Home x Detect_Heart_Disease x +

localhost:8888/notebooks/Detect_Heart_Disease.ipynb

jupyter Detect_Heart_Disease Last Checkpoint: 16 hours ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

```
[2]: # Step 7: Making Predictions on new data
new_data = pd.DataFrame({
    'age': [40, 39],
    'sex': [1, 0],
    'chest pain type': [2, 4],
    'resting bps': [100, 120],
    'cholesterol': [170, 239],
    'fasting blood sugar': [0, 1],
    'resting ecg': [0, 0],
    'max heart rate': [50, 80],
    'exercise angina': [0, 1],
    'oldpeak': [0.5, 1],
    'ST slope': [0, 1]
})

# Scaling new data
new_data_scaled = scaler.transform(new_data)

# Making predictions
predictions = best_model.predict(new_data_scaled)

# Output predictions
for i, prediction in enumerate(predictions):
    result = "At Risk of Heart Disease!" if prediction == 1 else "No Heart Disease..."
    print(f"\nPrediction for New Data Row {i+1}: {result}")

Prediction for New Data Row 1: No Heart Disease...

Prediction for New Data Row 2: At Risk of Heart Disease!
```

2. Predict Mobile Phone Pricing

Objective:

Build a system that can predict pricing for a mobile phone using data on available phones in the market. Predict if the mobile can be priced low/med/high/very high. Explore the data to understand the features and figure out an approach.

1. Introduction

In today's competitive mobile phone market, determining the price range of a device based on its technical specifications can help manufacturers and retailers with pricing strategies, and assist consumers in making informed choices. This project uses machine learning to classify mobile phones into four price categories—Low, Medium, High, and Very High—based on a variety of hardware features.

2. Dataset Overview

The dataset consists of multiple features that describe mobile phone specifications:

- **battery_power**: Total energy capacity of the battery
- **blue**: Bluetooth availability (1 = Yes, 0 = No)
- **clock_speed**: Speed at which microprocessor executes instructions
- **dual_sim**: Dual SIM support (1 = Yes, 0 = No)
- **fc**: Front camera megapixels
- **four_g**: 4G capability (1 = Yes, 0 = No)
- **int_memory**: Internal memory in GB

- **m_dep**: Mobile depth in cm
- **mobile_wt**: Mobile weight in grams
- **n_cores**: Number of processor cores
- **pc**: Primary camera megapixels
- **px_height**: Pixel height of the screen
- **px_width**: Pixel width of the screen
- **ram**: Random access memory in MB
- **sc_h**: Screen height in cm
- **sc_w**: Screen width in cm
- **talk_time**: Maximum talk time on a single charge
- **three_g**: 3G capability (1 = Yes, 0 = No)
- **touch_screen**: Touch screen availability (1 = Yes, 0 = No)
- **wifi**: Wi-Fi support (1 = Yes, 0 = No)
- **Target**: price_range (0: Low Cost, 1: Medium Cost, 2: High Cost, 3: Very High Cost)

The dataset was loaded using **pandas**, and no missing values were found, eliminating the need for imputation.

3. Methodology

a. Data Preprocessing

- **Feature Scaling**: All features were standardized using **StandardScaler** to ensure uniform scale and improve model performance.
- **Feature & Target Split**: The features and target (price_range) were separated for modeling.

- **Train-Test Split:** The dataset was split into training (80%) and testing (20%) sets using **train_test_split** with stratification to preserve class distribution.

b. Model Training

1. Logistic Regression

- Used a multinomial logistic regression model with L-BFGS solver.
- Achieved an accuracy of approximately **0.96**

2. Random Forest Classifier

- Trained using 100 estimators and a fixed random state for reproducibility.
- Achieved an accuracy of approximately **0.88**

c. Model Evaluation

- The better performing model (based on test accuracy) was selected for further evaluation.
- Evaluation metrics included:
 - **Accuracy Score**
 - **Classification Report** (Precision, Recall, F1-score)
 - **Confusion Matrix**, visualized using a **heatmap** for clarity.

d. Prediction on New Data

- Two hypothetical mobile phone entries were constructed with known specifications.
- The scaler from preprocessing was applied to these entries.
- Predictions were made using the best-performing model.

4. Results

- **Best Model:** Based on test accuracy, the **Random Forest Classifier** (or Logistic Regression, if higher) was selected.

- The **classification report** provided detailed class-wise performance.
- The **confusion matrix heatmap** clearly showed the distribution of predictions across actual vs. predicted classes.
- The model successfully predicted price ranges for new, unseen mobile phone data.

Example Output:

Prediction for New Data Row 1: Low Cost

Prediction for New Data Row 2: Very High Cost

5. Conclusion

This project demonstrates the utility of machine learning in solving real-world classification problems using structured data. Both **Random Forest** and **Logistic Regression** models showed strong performance, with Random Forest slightly outperforming in accuracy. The model's ability to generalize to new data shows promise for practical applications in product classification, pricing strategies, and recommendation systems.

6. Screenshots of results:

```
Logistic Regression Accuracy: 0.96
Random Forest Accuracy: 0.88

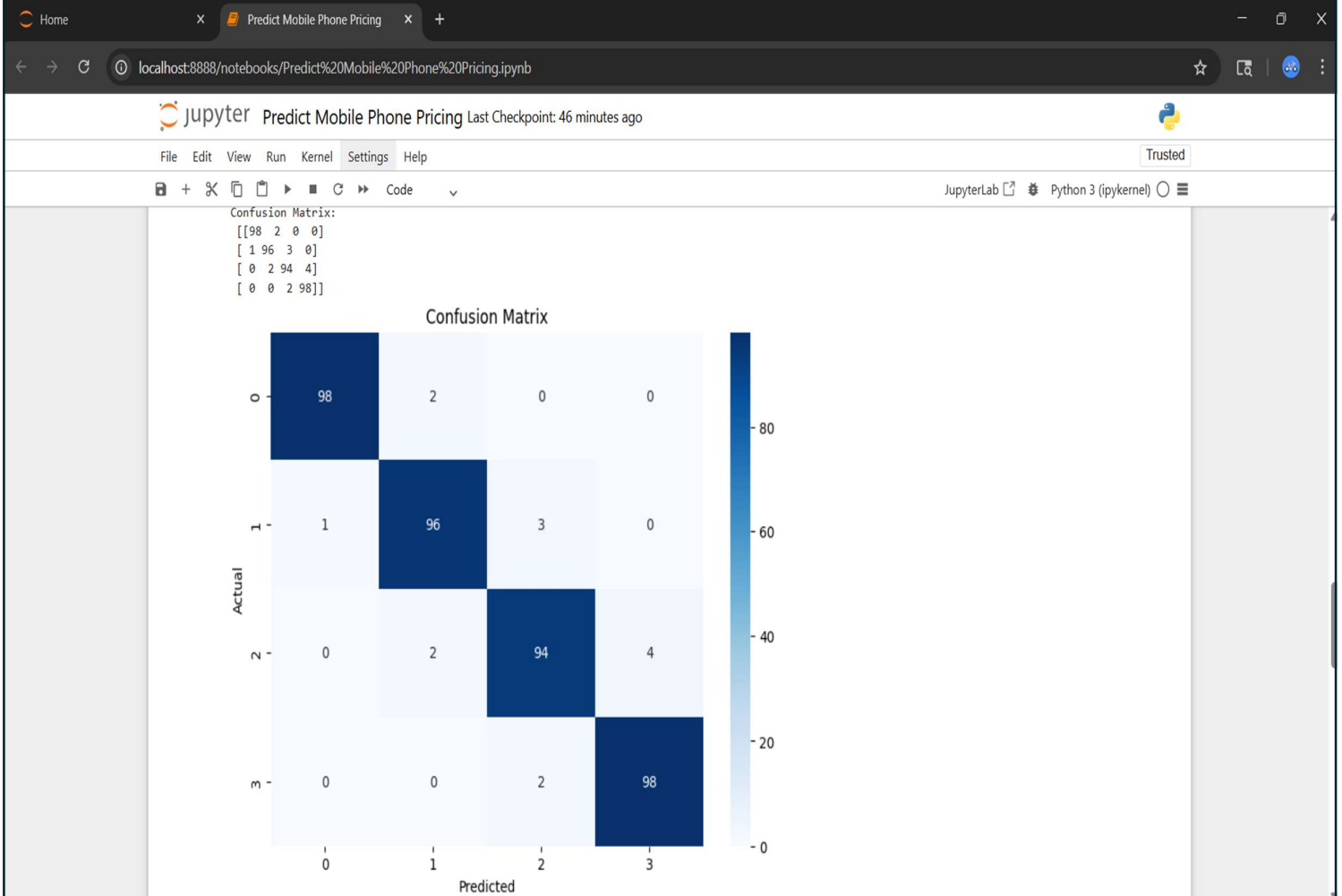
Best Model Metrics:

Accuracy Score: 0.965

Classification Report:
      precision    recall  f1-score   support

0         0.99      0.98      0.98       100
1         0.96      0.96      0.96       100
2         0.95      0.94      0.94       100
3         0.96      0.98      0.97       100

 accuracy          0.96       400
 macro avg         0.97      0.96      0.96       400
 weighted avg         0.97      0.96      0.96       400
```



Home x Detect Heart Disease x Predict Mobile Phone Pricing x +

localhost:8888/notebooks/Predict%20Mobile%20Phone%20Pricing.ipynb

Jupyter Predict Mobile Phone Pricing Last Checkpoint: 50 minutes ago

File Edit View Run Kernel Settings Help Trusted

JupyterLab Python 3 (ipykernel)

```
'dual_sim': [1, 1],
'fc': [1, 5],
'four_g': [0, 1],
'int_memory': [8, 64],
'm_dep': [0.5, 0.9],
'mobile_wt': [150, 180],
'n_cores': [4, 8],
'pc': [5, 13],
'px_height': [500, 1200],
'px_width': [800, 1600],
'ram': [1024, 4096],
'sc_h': [10, 14],
'sc_w': [5, 7],
'talk_time': [10, 20],
'three_g': [1, 1],
'touch_screen': [1, 1],
'wifi': [1, 1]
))

# Scaling new data
new_data_scaled = scaler.transform(new_data)

# Making predictions
predictions = best_model.predict(new_data_scaled)

# Output predictions
label_map = {0: "Low Cost", 1: "Medium Cost", 2: "High Cost", 3: "Very High Cost"}
for i, prediction in enumerate(predictions):
    print(f"\nPrediction for New Data Row {i+1}: {label_map[prediction]}")

Prediction for New Data Row 1: Low Cost

Prediction for New Data Row 2: Very High Cost
```

Tech Stack used in these Projects:

Category	Tools/Libraries Used
Programming	Python
Data Handling	Pandas, NumPy
Preprocessing	Scikit-learn (StandardScaler, train_test_split)
ML Models	Scikit-learn (LogisticRegression, RandomForestClassifier)
Evaluation	Scikit-learn (accuracy_score, classification_report, etc)
Visualization	Matplotlib, Seaborn
Utility	Warnings