UNIFIED MENTOR
YOUR SKILL, SUCCESS & JOURNEY

PREDICT MOBILE PHONE PRICING

Using Machine Learning Techniques

George Mathew
16th December 2024

EXECUTIVE SUMMARY

Overview:

• The project aims to predict mobile phone pricing categories using machine learning techniques.

• Classification models are developed to categorize phones into "Low," "Medium," "High," and "Very High" price ranges.

• Insights gained from analysis can aid mobile manufacturers and marketers in decision-making.

Key Highlights:

• Dataset of 2000 records with 21 features.

• Comparative analysis of five machine learning algorithms.

• Decision Tree and Random Forest models provided the highest accuracy.

• Significant predictors include "RAM," "Battery Power," and "Pixel Resolution."

OUTLINE

1. Introduction
2. Objective
3. Dataset Overview
4. Data Preprocessing
5. Methodology
6. Exploratory Data Analysis
7. Feature Importance
8. Model Building
9. Model Evaluation
10. Insights and Recommendations
11. Conclusion
12. Appendix

INTRODUCTION

Context:

• The rapid growth in smartphone technology and competition makes pricing prediction essential.

• Understanding factors influencing price helps optimize manufacturing and marketing strategies.

Relevance:

• A robust pricing strategy enhances profitability and market share.

Why Machine Learning?

• Predictive models identify key pricing determinants efficiently.

OBJECTIVE

Primary Objective:

• To predict the price range of mobile phones using classification models.

Specific Goals:

• Identify key features affecting mobile phone pricing.

• Compare the performance of various machine learning algorithms.

Sample Size Justification:

• A sample of **2000 records** was chosen to balance computational efficiency and statistical reliability.

• **Dataset size:** Moderate enough to ensure robust model training.

• **Data distribution:** Representative of diverse mobile phone features.
Ensures sufficient variability across price categories.

DATASET OVERVIEW. (1/3)

Data Source:

• Simulated dataset containing 2000 rows and 21 columns.

Features:

1. Battery Power
2. RAM
3. Pixel Resolution (Height and Width)
4. Internal Memory
5. Number of CPU Cores
6. Screen Dimensions
7. 3G and 4G Support
8. WiFi and Bluetooth Availability
9. Price Range (Target Variable)

Target Variable:

• Price Range: Categorized as
0 (Low),
1 (Medium),
2 (High),
3 (Very High).

Summary Statistics:

• Continuous variables: Mean, median, standard deviation.

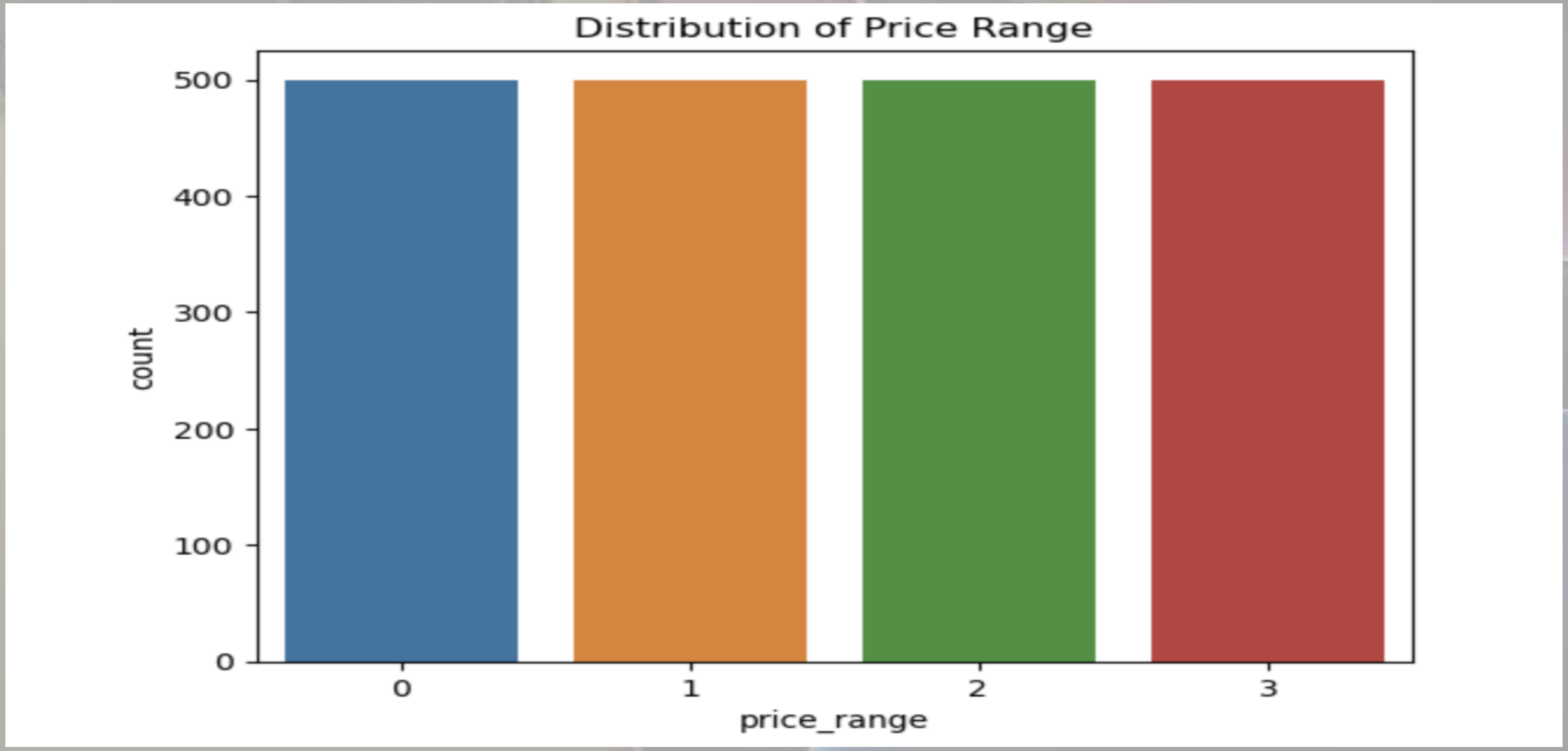
• Categorical variables: Frequency counts.

Graphs / Images:

Visual representations of price ranges and categorical variables can be critical for better understanding. (Slides of the histogram of price ranges and bar chart of categorical variables attached)

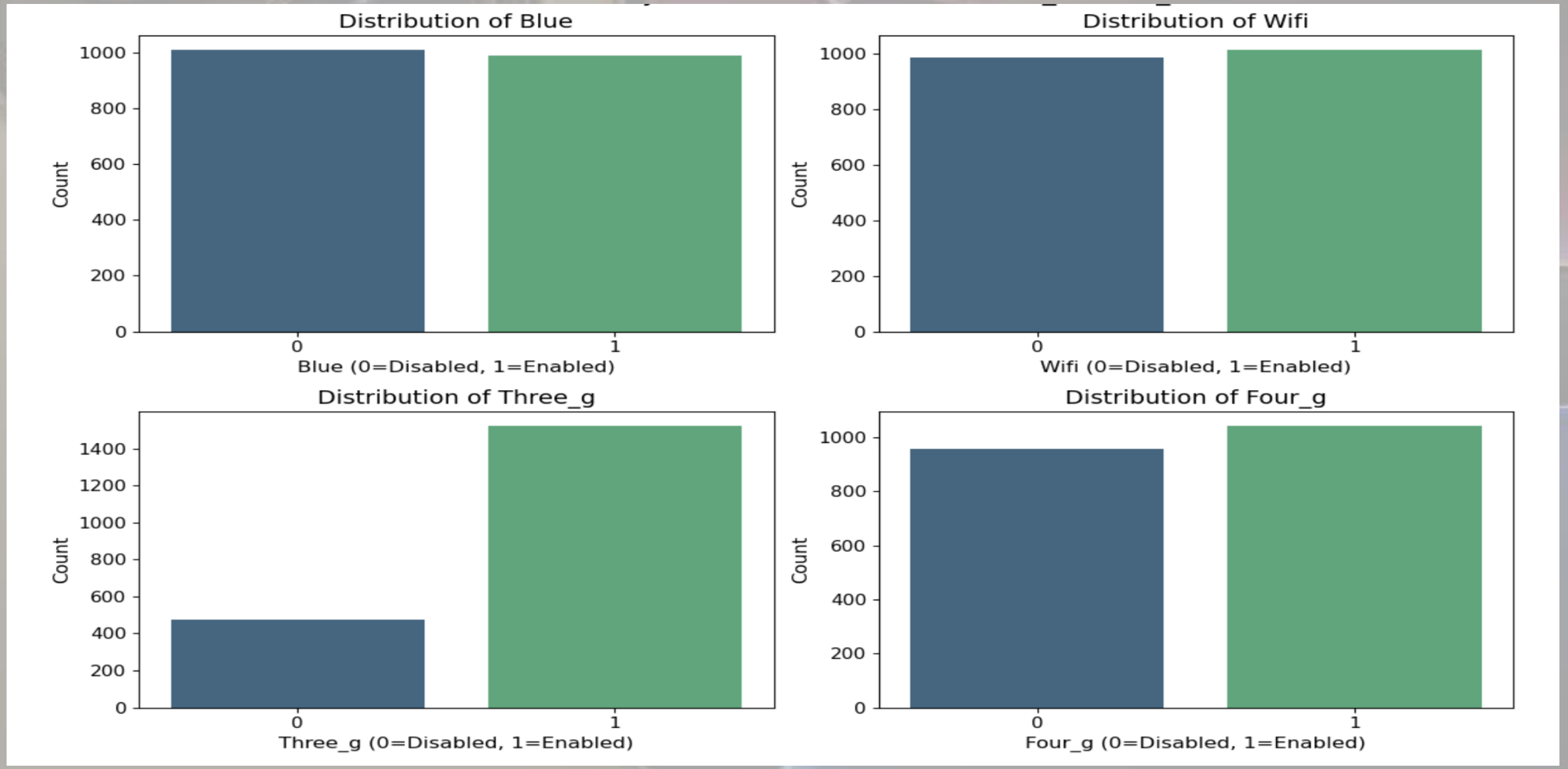
DATASET OVERVIEW (2/3)

Histogram of price range :



DATASET OVERVIEW (3/3)

Bar Chart Showing Distribution of Categorical Variables 'Bluetooth' 'Wi-Fi' '3G' and '4G':



DATA PREPROCESSING

Steps Taken:

Handling Missing Values:

• Checked for missing values using data.isnull().sum().

• Replaced 'px_height' and 'sc_w' values of 0 with nearest matches.

Handling Outliers:

• Removed outliers in 'px_height' using IQR method.

Data Scaling:

• Applied Standard Scaling to normalize numerical features using StandardScaler.

Data Exploration:

• Visualized feature correlations using a heatmap.

• Analyzed the distribution of the target variable (price_range) and key features using count plots (Bar Chart) and pair plots (Scatter Plot).

Modeling and Evaluation:

• Implemented and evaluated Logistic Regression, KNN, SVM (Linear and RBF Kernels), Decision Tree and Random Forest models.

• Compared model performance based on accuracy and F1 Scores (Bar Chart).

• Plotted a Training Accuracy Vs Validation Accuracy curve for the Logistic Regression model to analyze its performance.

METHODOLOGY

Framework:

•Comparative analysis of five machine learning models:

1. Logistic Regression
2. K-Nearest Neighbors (KNN)
3. Support Vector Machines (Linear and RBF Kernels)
4. Decision Tree.
5. Random Forest

Cross-Validation:

• Data split into training and testing sets using an 80-20 ratio.

Evaluation Metrics:

1. Accuracy 2. Precision 3. Recall 4. F1 Score 5. Confusion Matrix for performance insights.

Data Exploration:

• Correlation heatmap and pair plots for feature analysis and visualization.

Hyperparameter Tuning:

• Hyperparameter tuning was not explicitly performed for all models.

• Default parameters were used except for Random Forest (n_estimators=100) and KNN (n_neighbors=5).

EXPLORATORY DATA ANALYSIS

Insights Gained:

- RAM and Battery Power strongly correlate with the price range.
- Pixel Resolution (px_height × px_width) is another significant predictor.

Visualizations:

- Scatter plots: Pair plot of Key Features with Price Range. (Appendix – 4)
- Feature Correlation Heatmap . (Appendix -3)

Key Observations:

- Higher RAM leads to a higher price range.
- Phones with higher battery power tend to cost more.
- 4G-enabled phones are generally more expensive.

FEATURE IMPORTANCE (1/2)

(A) Key Predictors & their Statistical Insights

Rank	Feature	Mean	Median	Std. Dev.	Importance
1.	ram	2124.04	2146.50	1085.05	56.26 %
2.	battery_power	1238.44	1225.00	439.62	13.88 %
3.	px_width	1250.79	1247.00	431.80	8.26 %

(B) Interpretation of Feature Statistics

Importance (%)

- **RAM** contributes the most (56.26 %) to the model's prediction accuracy, indicating that it has the strongest influence on the pricing range of mobile phones.
- **Battery Power** follows with 13.88 %, suggesting it has a moderate impact.
- **Pixel Width** has the least contribution among the top three predictors, with 8.26 %.

FEATURE IMPORTANCE (2/2)

(B) Interpretation of Feature Statistics contd/-

Mean and Median

• The **mean** and **median** provide central tendencies of the features:

• **RAM** has a mean of 2124.04 and a median of 2146.50, indicating that most mobile phones in the dataset have RAM close to these values.

• For **Battery Power**, the mean of 1238.44 and median of 1225.00 are close, reflecting a balanced distribution.

• Similarly, **Pixel Width** shows a mean of 1250.79 and a median of 1247.00, indicating minimal skewness.

Standard Deviation (Std. Dev.)

• **RAM** has the highest standard deviation of 1085.05, implying a broader range of values and greater variability in RAM across mobile phones.

• **Battery Power** of 439.62 and **Pixel Width** of 431.80 have similar, lower standard deviations, showing less variability in these features.

Key Takeaway

• The model identifies **RAM** as the most significant predictor due to its high variability and strong correlation with the pricing range.

• **Battery Power** and **Pixel Width**, while less significant, still contribute meaningful insights for predicting price.

MODEL BUILDING (1/3)

Implementation Steps:

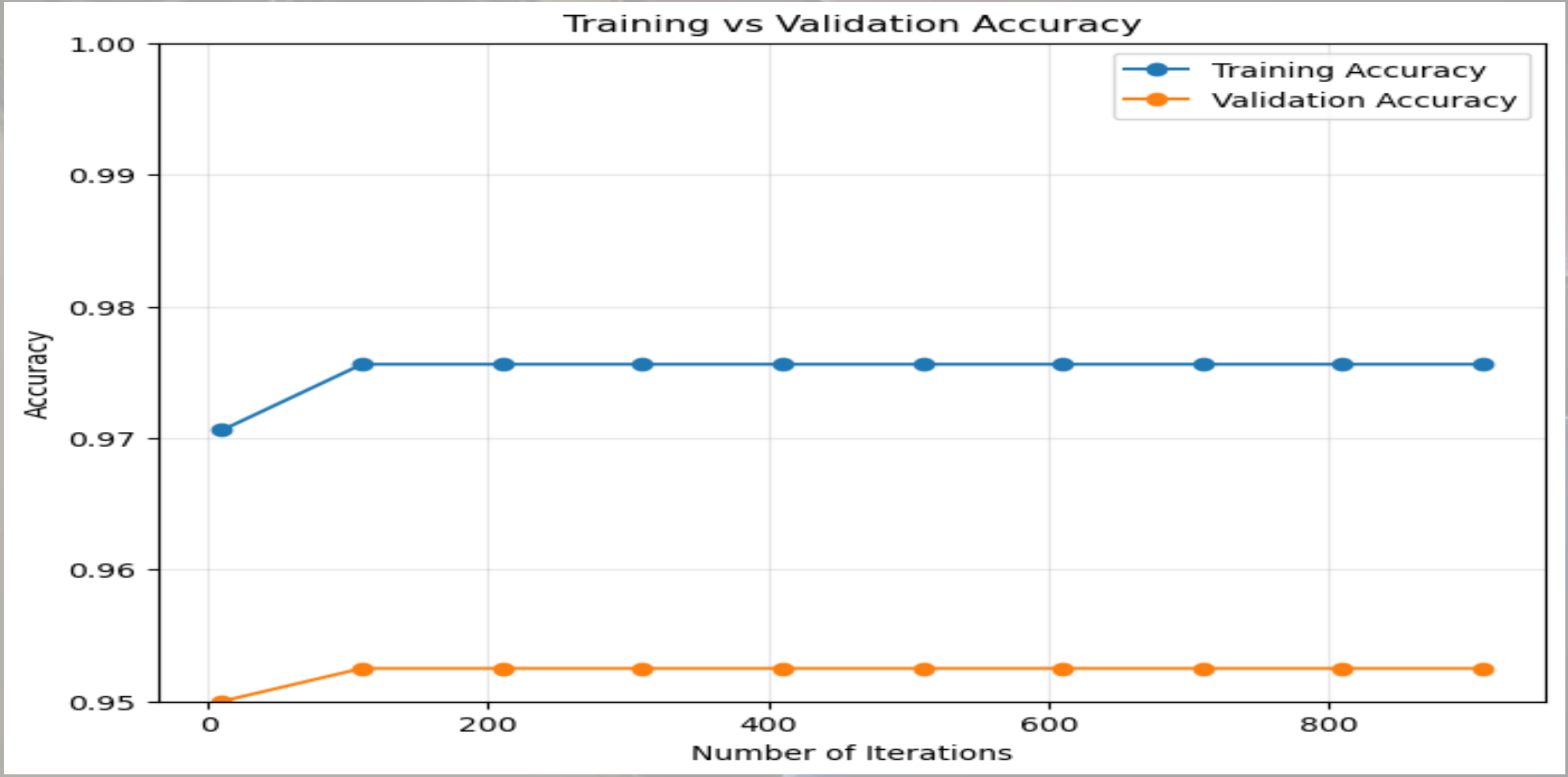
1. Data split into training (80%) and test (20%) sets.
2. Applied standard scaling for numerical features.
3. Trained models using default hyperparameters.

Graphs/Images:

Visual representations of Training vs. Validation Accuracy plot and Feature importance plot can be critical for better understanding. (Slides of the Training vs. Validation accuracy plots and Feature importance plot are attached for clarity)

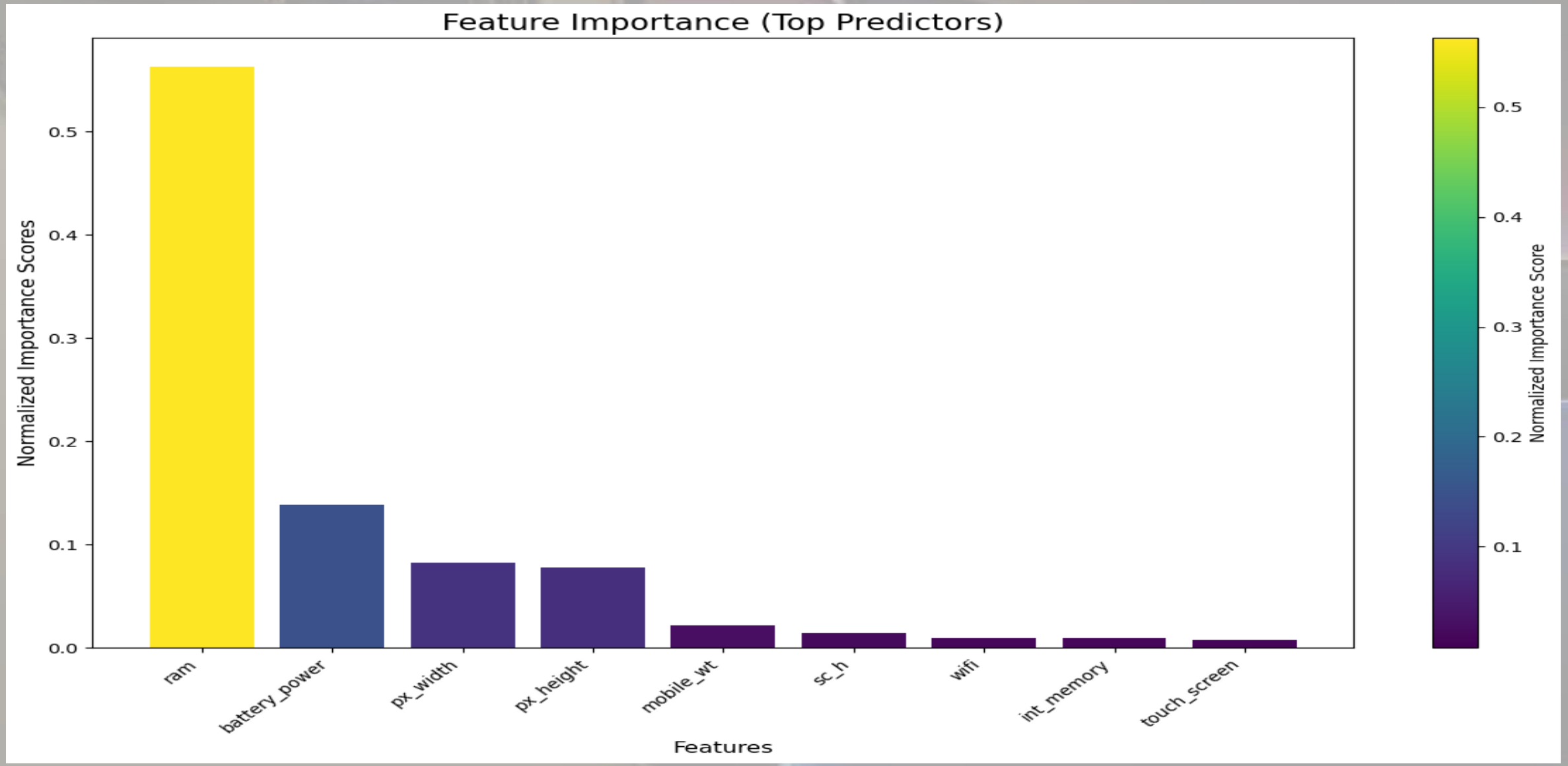
MODEL BUILDING (2/3)

Training vs. Validation accuracy plots



MODEL BUILDING (3/3)

Bar Chart of Feature Importance Scores – Random Forest



MODEL EVALUATION (1/2)

Results:

- **Decision Tree:** Accuracy = ~79% ; F1 Score = ~79%
- **Random Forest:** Accuracy = ~89% ; F1 Score = ~89%
- **Logistic Regression:** Accuracy = ~95% ; F1 Score = ~95%
- **KNN:** Accuracy = ~92% ; F1 Score = ~92%
- **SVM (Linear):** Accuracy = ~97% ; F1 Score = ~97%
- **SVM (RBF):** Accuracy = ~95% ; F1 Score = ~95%

Confusion Matrices:

- **SVM (Linear)** provides the highest accuracy and minimal misclassifications.
- **Logistic Regression** remains a strong contender with consistent performance across classes and interpretability.

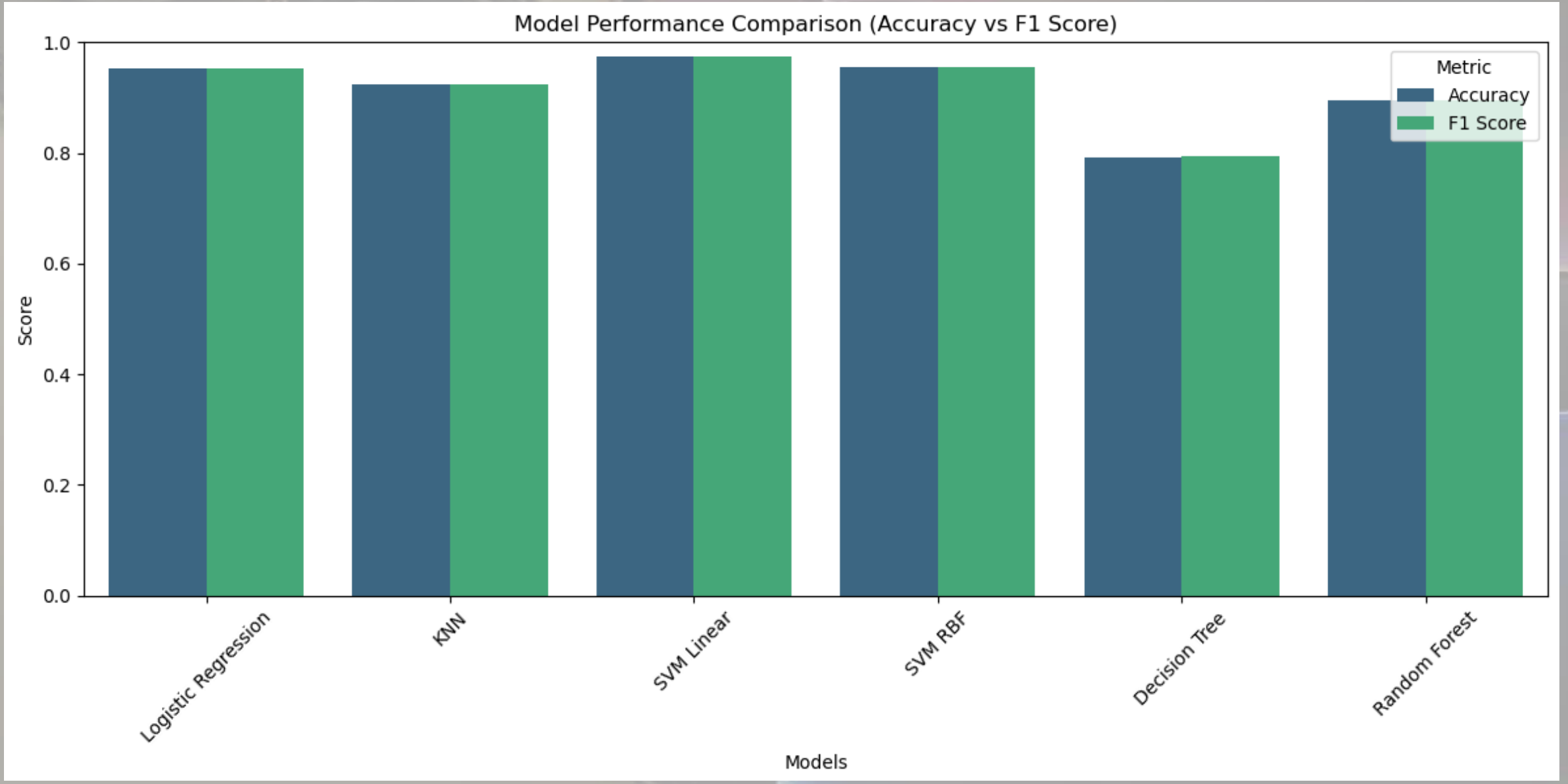
Decision Tree: Underperforms significantly, showing overfitting to certain classes.
Random Forest: Improves over Decision Tree but shows minor overfitting tendencies.
KNN struggles with balancing precision and recall, making it less suitable for datasets with intricate decision boundaries

Graphs / Images:

Visual representations of accuracy and F1 scores can be critical for better understanding. (Slide of the comparative bar charts of accuracy and F1 scores attached)

MODEL EVALUATION (2/2)

Bar Chart Showing Model Accuracy Vs F1 Score



INSIGHTS & RECOMMENDATIONS

Insights:

- RAM is the most critical factor for price prediction.
- SVM Linear is the best performing model with Logistic Regression being a close contender.
- Phones with higher RAM and battery power are categorized in higher price ranges.

Recommendations:

- For budget phones, emphasize optimizing RAM and battery power.
- For premium phones emphasize RAM, Battery Power and pixel resolution.
- Leverage predictive models for competitive pricing strategies.

CONCLUSION (1/4)

Key Takeaways:

- Predictive models can reliably categorize mobile phone prices.
- Despite SVM Linear being the best performing model Logistic Regression can be preferred for the reasons given in next slide.
- Findings can guide product design, pricing and marketing decisions.

CONCLUSION (2/4)

Why Logistic Regression Over SVM ?

1. Interpretability and Explainability

- **Logistic Regression:**
  - Provides direct **probabilities** for class membership, making predictions easily interpretable.
  - Coefficients indicate the impact of each feature on the prediction.
  - Suitable for stakeholders who require clear decision-making insights (e.g., business analysts, domain experts).
- **SVM:**
  - Black-box model with complex decision boundaries.
  - Does not provide probabilities by default, making it harder to interpret predictions.

2. Computational Efficiency

- **Logistic Regression:**
  - Faster to train, even on large datasets.
  - Scales well with high-dimensional data and multiple classes.
- **SVM:**
  - Slower, especially for large datasets and non-linear kernels.
  - Quadratic complexity in training time with larger datasets.

CONCLUSION (3/4)

Why Logistic Regression Over SVM ?

3. Generalization

- **Logistic Regression:**
  - Performs consistently across diverse datasets, especially in linearly separable problems.
  - Less prone to overfitting compared to SVM with non-linear kernels like RBF.
- **SVM:**
  - May require extensive **hyperparameter tuning** (e.g., kernel choice, regularization) to achieve similar generalization.
  - Risks overfitting with complex kernels.

4. Multiclass Classification

- **Logistic Regression:**
  - Natively supports **multiclass classification** (via multinomial logistic regression).
  - Simple and efficient implementation for multi-class problems.
- **SVM:**
  - Requires additional steps (e.g., one-vs-one or one-vs-rest schemes) for multi-class classification.
  - Increases computational complexity.

CONCLUSION (4/4)

Why Logistic Regression Over SVM ?

5. Practical Use Cases

• Logistic Regression is widely used in industries requiring explainability, such as:

- **Healthcare:** Diagnosing diseases with clear contributing factors.
- **Marketing:** Customer segmentation and churn prediction.
- **Finance:** Credit scoring and fraud detection.

• SVM is better suited for highly **complex, non-linear problems** but less practical for interpretability-driven tasks.

6. Performance vs. Usability

• While SVM (Linear) marginally outperforms Logistic Regression in terms of raw accuracy (based on the confusion matrix), the difference is small and does not justify the added complexity and lack of interpretability.

• Logistic Regression:

- Provides **balanced accuracy** with simplicity and ease of deployment.

• SVM:

- Requires higher computational resources and tuning for minor performance gains.

Final Conclusion

Choose **Logistic Regression** for:

- Scalability, simplicity, and interpretability.
- Real-world applications where decision-making transparency is crucial.

APPENDIX

- **Code snippets for:**

1. The train-test split, Data preprocessing (scaling) and model evaluation.
2. Logistic Regression training and hyperparameter tuning.

- **Plots/Charts:**

3. Features Correlation heatmap
4. Pair plots for Key Features.
5. Heat Map of ‘0’ value counts of variables

- **Supporting Data:**

6. Confusion matrices for each model, as these were calculated in the code.

APPENDIX - 1

Code snippets for the train-test split, scaling and model evaluation.

```
# Split data first
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Fit scaler on training data and transform both training and test data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Logistic Regression
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
y_pred_log_reg = log_reg.predict(X_test)

# Evaluation
print("Logistic Regression Results:")
print(classification_report(y_test, y_pred_log_reg))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_log_reg))
```

```
# KNN
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

# Evaluation
print("KNN Results:")
print(classification_report(y_test, y_pred_knn))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_knn))
```

```
# SVM Linear Kernel
svm_linear = SVC(kernel='linear')
svm_linear.fit(X_train, y_train)
y_pred_svm_linear = svm_linear.predict(X_test)

# Evaluation
print("SVM Linear Kernel Results:")
print(classification_report(y_test, y_pred_svm_linear))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svm_linear))

# SVM RBF Kernel
svm_rbf = SVC(kernel='rbf')
svm_rbf.fit(X_train, y_train)
y_pred_svm_rbf = svm_rbf.predict(X_test)

# Evaluation
print("SVM RBF Kernel Results:")
print(classification_report(y_test, y_pred_svm_rbf))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svm_rbf))
```

```
# Decision Tree
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
y_pred_tree = decision_tree.predict(X_test)

# Evaluation
print("Decision Tree Results:")
print(classification_report(y_test, y_pred_tree))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_tree))
```

```
# Random Forest
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest.fit(X_train, y_train)
y_pred_forest = random_forest.predict(X_test)

# Evaluation
print("Random Forest Results:")
print(classification_report(y_test, y_pred_forest))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_forest))
```

APPENDIX - 2

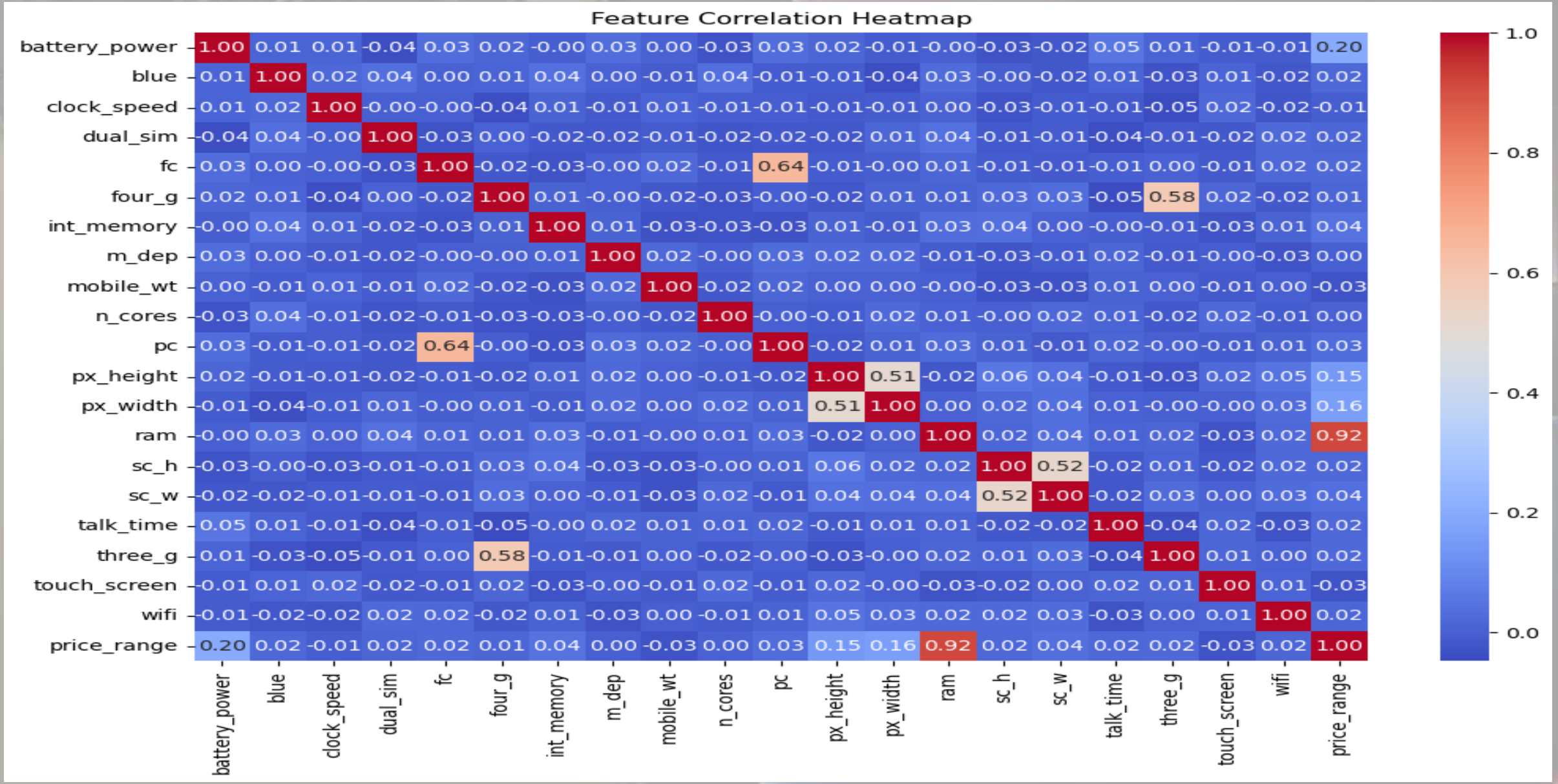
Code snippet for Logistic Regression training and hyperparameter tuning

```
# Logistic Regression
log_reg = LogisticRegression(solver='saga', max_iter=1000, random_state=42)
log_reg.fit(X_train_scaled, y_train)
y_pred_log_reg = log_reg.predict(X_test_scaled)

# Evaluation
print("Logistic Regression Results:")
print(classification_report(y_test, y_pred_log_reg))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_log_reg))
```

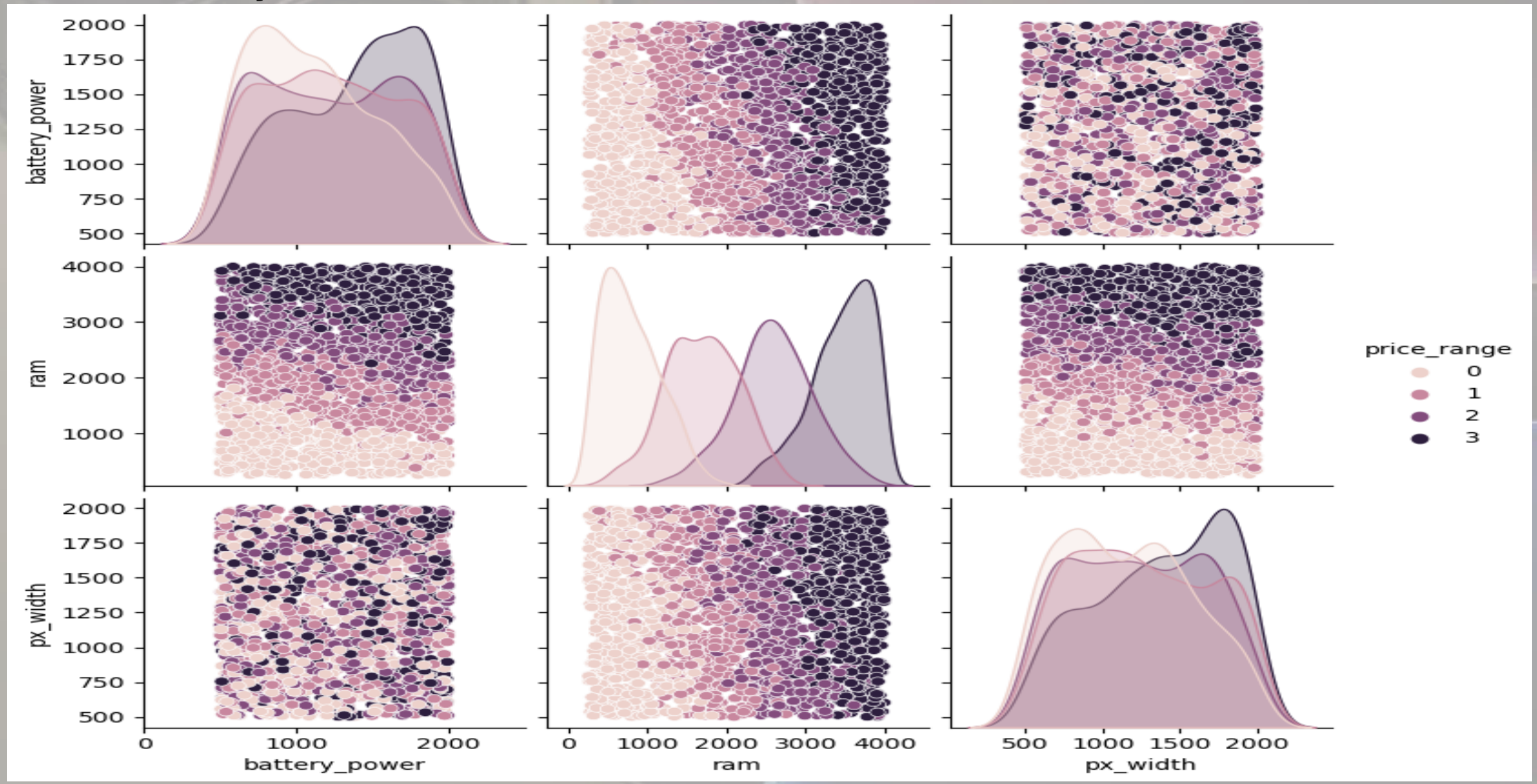
APPENDIX - 3

Feature Correlation Heat Map



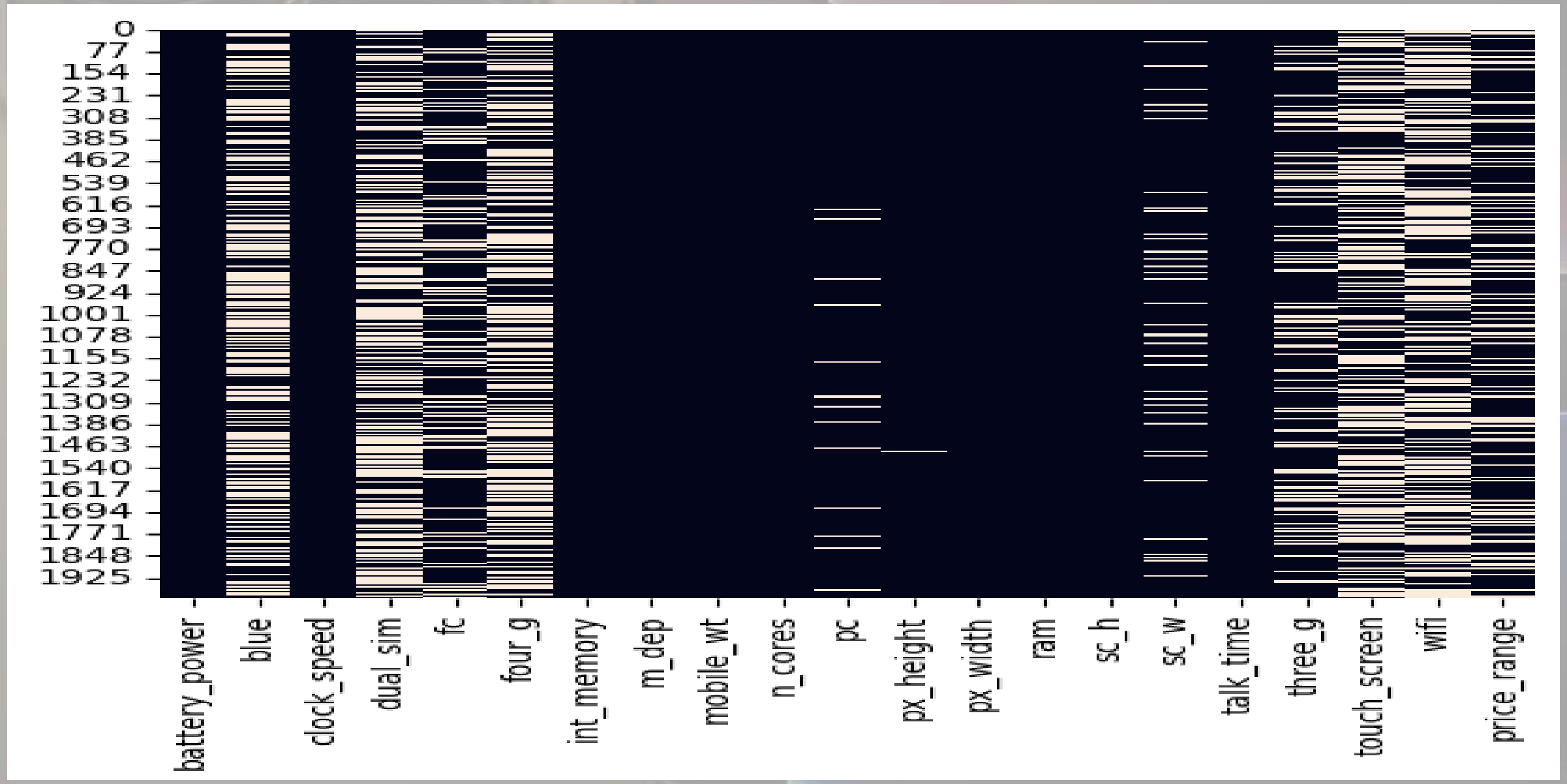
APPENDIX - 4

Pair Plot for Key Features



APPENDIX - 5

‘0’ values Count Heat Map



APPENDIX - 6

Model Scores & Confusion Matrix

Logistic Regression Results:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	100
1	0.94	0.95	0.95	100
2	0.93	0.90	0.91	100
3	0.95	0.97	0.96	100
accuracy			0.95	400
macro avg	0.95	0.95	0.95	400
weighted avg	0.95	0.95	0.95	400

Confusion Matrix:

```
[[99  1  0  0]
 [ 1 95  4  0]
 [ 0  5 90  5]
 [ 0  0  3 97]]
```

KNN Results:

	precision	recall	f1-score	support
0	0.97	0.98	0.98	100
1	0.91	0.92	0.92	100
2	0.87	0.88	0.88	100
3	0.95	0.92	0.93	100
accuracy			0.93	400
macro avg	0.93	0.92	0.93	400
weighted avg	0.93	0.93	0.93	400

Confusion Matrix:

```
[[98  2  0  0]
 [ 3 92  5  0]
 [ 0  7 88  5]
 [ 0  0  8 92]]
```

Decision Tree Results:

	precision	recall	f1-score	support
0	0.90	0.88	0.89	100
1	0.75	0.80	0.77	100
2	0.69	0.70	0.70	100
3	0.84	0.79	0.81	100
accuracy			0.79	400
macro avg	0.79	0.79	0.79	400
weighted avg	0.79	0.79	0.79	400

Confusion Matrix:

```
[[88 12  0  0]
 [10 80 10  0]
 [ 0 15 70 15]
 [ 0  0 21 79]]
```

SVM Linear Kernel Results:

	precision	recall	f1-score	support
0	0.99	1.00	1.00	100
1	0.99	0.96	0.97	100
2	0.95	0.96	0.96	100
3	0.97	0.98	0.98	100
accuracy			0.97	400
macro avg	0.98	0.97	0.97	400
weighted avg	0.98	0.97	0.97	400

Confusion Matrix:

```
[[100   0   0   0]
 [  1  96   3   0]
 [  0   1  96   3]
 [  0   0   2  98]]
```

Random Forest Results:

	precision	recall	f1-score	support
0	0.96	0.95	0.95	98
1	0.91	0.89	0.90	103
2	0.77	0.87	0.82	94
3	0.92	0.84	0.88	105
accuracy			0.89	400
macro avg	0.89	0.89	0.89	400
weighted avg	0.89	0.89	0.89	400

Confusion Matrix:

```
[[93  5  0  0]
 [ 4 92  7  0]
 [ 0  4 82  8]
 [ 0  0 17 88]]
```

SVM RBF Kernel Results:

	precision	recall	f1-score	support
0	0.98	1.00	0.99	100
1	0.93	0.97	0.95	100
2	0.95	0.89	0.92	100
3	0.96	0.96	0.96	100
accuracy			0.95	400
macro avg	0.95	0.95	0.95	400
weighted avg	0.95	0.95	0.95	400

Confusion Matrix:

```
[[100   0   0   0]
 [  2  97   1   0]
 [  0   7  89   4]
 [  0   0   4  96]]
```