# MOBILE PRICE RANGE
# PREDICTION MODEL


Presented By
**Priyanshi Lamba**

# PROBLEM STATEMENT: MOBILE PRICE PREDICTION

In today's highly competitive mobile phone market, pricing a product correctly is crucial for its success. Bob, a new entrepreneur in the mobile industry, aims to compete with established brands like Apple and Samsung but lacks the expertise to estimate the price of the mobile phones his company produces. Simply guessing the price is not an option, as accurate pricing requires data-driven decision-making.

To address this challenge, Bob collects sales data from various mobile companies and seeks to identify relationships between different mobile phone features (such as RAM, internal memory, battery power, and connectivity options) and their selling price. However, instead of predicting the exact price, the goal is to classify mobiles into four price ranges—low, medium, high, and very high.

This project leverages machine learning algorithms, including **Naïve Bayes, Decision Tree**, and **Random Forest**, to develop a Mobile Price Range Prediction Model that helps Bob make informed pricing decisions based on key product specifications.

# VARIABLE DESCRIPTION

| Categorical Variables | Numerical Variables | Target Variable |
|---|---|---|
| **blue**: Has Bluetooth or not | **battery_power**: Total energy a battery can store in one time measured in mAh | **price_range**: 0 (low cost), 1 (medium cost), 2 (high cost), 3 (very high cost) |
| **dual_sim**: Has dual SIM support or not | **clock_speed**: Speed at which microprocessor executes instructions | |
| **four_g**: Has 4G or not | **int_memory**: Internal Memory in Gigabytes | |
| **three_g**: Has 3G or not | **m_dep**: Mobile Depth in cm | |
| **touch_screen**: Has touch screen or not | **mobile_wt**: Weight of mobile phone | |
| **wifi**: Has WiFi or not | **n_cores**: Number of cores of processor | |
| | **pc**: Primary Camera megapixels | |
| | **px_height**: Pixel Resolution Height | |
| | **px_width**: Pixel Resolution Width | |
| | **ram**: Random Access Memory in Megabytes | |
| | **sc_h**: Screen Height of mobile in cm | |
| | **sc_w**: Screen Width of mobile in cm | |
| | **talk_time**: Longest time that a single battery charge will last when you are using it | |

# DATA CLEANING AND PREPROCESSING

To build an accurate Mobile Price Range Prediction Model, it was essential to preprocess and clean the dataset before applying machine learning algorithms. The following steps were taken to ensure data quality and reliability:

1. **Data Type Conversion:** The dataset was initially structured, but some variables needed appropriate type conversion.

- **Categorical Variables:** Certain columns, which represent categorical features (such as **dual_sim, four_g, three_g, touch_screen, wifi,** etc.), were converted into factor data types to improve model efficiency.

```
M[,-c(1,3,6,7,8,9,10,11,12,13,14,15,16)]<-lapply(M[,-c(1,3,6,7,8,9,10,11,12,13,14,15,16)],factor)
```

**2. Missing Value Detection and Handling:**
The dataset was checked for missing values using the **map(M, ~sum(is.na(.)))**function. Fortunately, no missing values were found initially.

**3. Checking for Outliers:**
To identify extreme values that could impact model performance, box plots were used for key numerical features, including **battery_power, clock_speed, int_memory, m_dep, mobile_wt, n_cores, pc, px_height, px_width, sc_h, sc_w, and talk_time.**

**-> Only one outlier was found in the px_height column.**

**4. Outlier Treatment**

- The outlier in px_height was identified and stored in a variable
- The outlier values were replaced with NA
- The missing values were then imputed using the median to avoid distortion in the dataset
- A final box plot check confirmed that outliers were handled appropriately.

**5. Data Partition**
After cleaning, the dataset was partitioned into training (80%) and testing (20%) sets to prepare for model training and validation.

```
#storing outliers in variable named outpx_height
outpx_height<-boxplot(M$px_height)$out
length(outpx_height)

#Replacing outliers values with NA
M[M$px_height %in% outpx_height, "px_height"] = NA

#Imputing NA values with median
M[11]<-impute(M[11], fun = median)

#checking if outliers are gone
boxplot(M$px_height)
```

```
#Data partition
set.seed(100)
Train <- createDataPartition(M1$price_range, p=0.8, list=FALSE)
training <- M1[ Train, ]
testing <- M1[ -Train, ]
```

# NAÏVE BAYES MODEL FOR MOBILE PRICE RANGE PREDICTION

**Introduction to Naïve Bayes:**
Naïve Bayes is a probabilistic machine learning algorithm based on Bayes' theorem, which assumes that all features are independent given the class label. Despite this strong assumption, Naïve Bayes often performs well in classification tasks, especially with large datasets. It is widely used for text classification, spam detection, and medical diagnosis.

In this study, Naïve Bayes is used to classify mobile phones into four price ranges—low, medium, high, and very high—based on key specifications such as RAM, battery power, screen resolution, and connectivity features.

**Model Performance and Results:**
After training the Naïve Bayes model on the dataset, the confusion matrix and classification metrics were obtained to evaluate the model's performance.

**Confusion Matrix Analysis:**
From the confusion matrix:

- Class 0 (Low Price Range): 88 mobiles were correctly classified, with only 7 misclassified as class 1.
- Class 1 (Medium Price Range): 77 mobiles were correctly classified, but 12 were misclassified as class 0, and 8 as class 2.
- Class 2 (High Price Range): 71 mobiles were correctly classified, but 16 were misclassified as class 1, and 15 as class 3.
- Class 3 (Very High Price Range): 85 mobiles were correctly classified, but 11 were misclassified as class 2, and 3 as class 0.

**Overall Model Performance:**

- Accuracy: 80.25%, meaning that the model correctly classifies 80.25% of the test samples.
- Kappa Score: 0.7367, indicating strong agreement between predicted and actual values.
- P-Value: < 2.2e-16, confirming that the model performs significantly better than random classification.

**Class-wise Performance Metrics:**

| Metric | Class 0 (Low) | Class 1 (Medium) | Class 2 (High) | Class 3 (Very High) |
|---|---|---|---|---|
| Sensitivity (Recall) | 88.0% | 77.0% | 71.0% | 85.0% |
| Specificity | 97.67% | 90.00% | 89.67% | 96.33% |
| Positive Predictive Value (Precision) | 92.63% | 71.96% | 69.61% | 88.54% |
| Balanced Accuracy | 92.83% | 83.50% | 80.33% | 90.67% |

**Key Observations:**

- Class 0 (Low Price Range) has the highest recall (88%) and precision (92.63%), making it the best-predicted category.
- Class 2 (High Price Range) has the lowest precision (69.61%), indicating that many phones predicted as high-cost actually belong to other classes.
- Class 3 (Very High Price Range) has the highest specificity (96.33%), meaning it is well-separated from other classes.

**Interpretation and Conclusion:**
The Naïve Bayes model achieved a strong 80.25% accuracy, demonstrating its effectiveness in classifying mobile phones based on technical specifications. The model performs well in identifying low and very high price categories, but struggles slightly with medium and high price categories, likely due to feature overlap.

# RESULT:

```
          Reference
Prediction  0  1  2  3
         0 97  5  0  0
         1  3 93  8  0
         2  0  2 87  7
         3  0  0  5 93

Overall Statistics

               Accuracy : 0.925
                 95% CI : (0.8947, 0.9488)
    No Information Rate : 0.25
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3
Sensitivity            0.9700   0.9300   0.8700   0.9300
Specificity            0.9833   0.9633   0.9700   0.9833
Pos Pred Value         0.9510   0.8942   0.9062   0.9490
Neg Pred Value         0.9899   0.9764   0.9572   0.9768
Prevalence             0.2500   0.2500   0.2500   0.2500
Detection Rate         0.2425   0.2325   0.2175   0.2325
Detection Prevalence   0.2550   0.2600   0.2400   0.2450
Balanced Accuracy      0.9767   0.9467   0.9200   0.9567
```

# DECISION TREE MODEL FOR MOBILE PRICE PREDICTION

**Decision Tree Model for Mobile Price Prediction:**

**Overview:** A Decision Tree is a supervised machine-learning algorithm used for classification and regression tasks. It splits the dataset into branches based on feature values, forming a tree-like structure. The Decision Tree model is particularly useful in understanding hierarchical decision-making processes, making it an ideal choice for predicting mobile price ranges based on various features such as RAM, internal memory, and battery power.

**Model Explanation:** In our case, the Decision Tree model was trained to classify mobile phones into four price ranges (0: Low cost, 1: Medium cost, 2: High cost, 3: Very high cost). The tree splits the dataset based on the most significant features, with RAM appearing as the most influential factor in determining the price range. The tree structure shows how the decision-making process works, with each node representing a feature split, and leaf nodes representing the final classification.

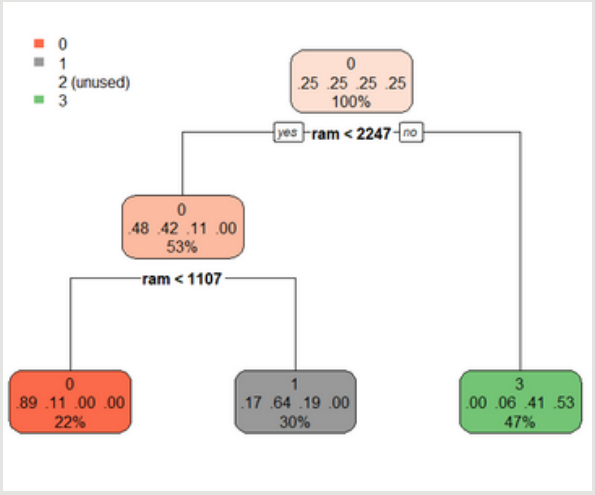
**Decision Tree Diagram Analysis:** The Decision Tree diagram illustrates how mobile phones are classified based on RAM values:

- If RAM is less than 2247 MB, the model further splits into different categories.
- Phones with RAM below 1107 MB are most likely to fall into the lowest price range (0 - Low cost).
- Phones with RAM between 1107 MB and 2247 MB are distributed among categories 0 and 1, with some probability for 2.
- Phones with RAM greater than 2247 MB are more likely to belong to the highest price category (3 - Very high cost).

**Model Performance and Interpretation:** From the confusion matrix and statistical summary:

- The overall accuracy of the model is 66%, meaning it correctly predicts price ranges 66% of the time.
- Sensitivity values for different classes show that the model performs well for class 0 and class 3 but struggles with class 2, as seen from the 0.00 sensitivity value for class 2.
- The specificity values indicate that the model is generally good at identifying when a mobile does not belong to a particular category.
- The Kappa score of 0.5467 suggests a moderate level of agreement between predictions and actual values.

**Conclusion:** The Decision Tree model provides a transparent and interpretable approach to price prediction, with RAM being the most critical factor. However, improvements such as pruning, hyperparameter tuning, or using ensemble methods like Random Forest may enhance its performance, particularly for underrepresented classes like class 2.




```
Confusion Matrix and Statistics

          Reference
Prediction   0   1   2   3
         0  85  10   0   0
         1  15  79  22   0
         2   0   0   0   0
         3   0  11  78 100

Overall Statistics

               Accuracy : 0.66
                 95% CI : (0.6113, 0.7063)
    No Information Rate : 0.25
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5467

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3
Sensitivity            0.8500   0.7900   0.00   1.0000
Specificity            0.9667   0.8767   1.00   0.7033
Pos Pred Value         0.8947   0.6810    NaN   0.5291
Neg Pred Value         0.9508   0.9261   0.75   1.0000
Prevalence             0.2500   0.2500   0.25   0.2500
Detection Rate         0.2125   0.1975   0.00   0.2500
Detection Prevalence   0.2375   0.2900   0.00   0.4725
Balanced Accuracy      0.9083   0.8333   0.50   0.8517
>
```

# RANDOM FOREST MODEL AND INTERPRETATION

**Random Forest Model and Interpretation:**
Introduction to Random Forest Random Forest is an ensemble learning method used for classification and regression tasks. It operates by constructing multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) from all the individual trees. This method reduces overfitting and improves accuracy compared to a single decision tree.

Model Configuration In the given case, the Random Forest classifier was trained with:

- Number of trees: 500
- Number of variables tried at each split: 10
- Out-of-Bag (OOB) error estimate: 10.25%

**Confusion Matrix Analysis:**
The confusion matrix shows the model's classification performance across different classes. The breakdown of misclassifications is as follows:

- Class 0: 378 correctly classified, 22 misclassified.
- Class 1: 347 correctly classified, 27 misclassified.
- Class 2: 343 correctly classified, 32 misclassified.
- Class 3: 368 correctly classified, 32 misclassified.

The class error rates are low, indicating good model performance.

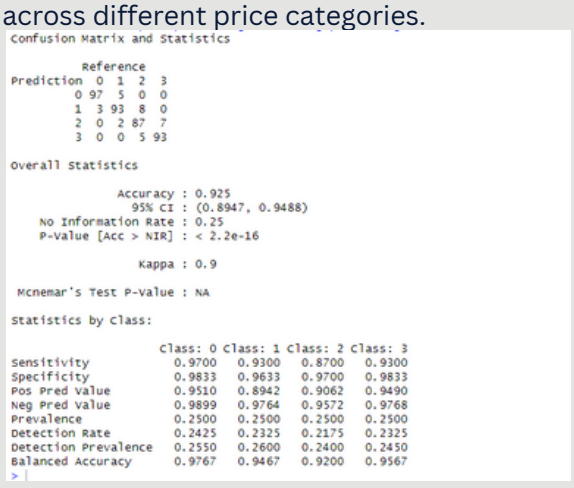
**Performance Metrics:**

- Accuracy: 92.5%, indicating that the model correctly classifies most instances.
- Kappa Score: 0.9, suggesting a high level of agreement between predictions and actual values.
- Sensitivity & Specificity: High values for all classes, with sensitivity ranging from 87% to 97% and specificity above 96% for all classes.
- Balanced Accuracy: High across all classes, with the lowest at 92.0% and the highest at 97.6%, demonstrating that the model is effective across different price ranges.

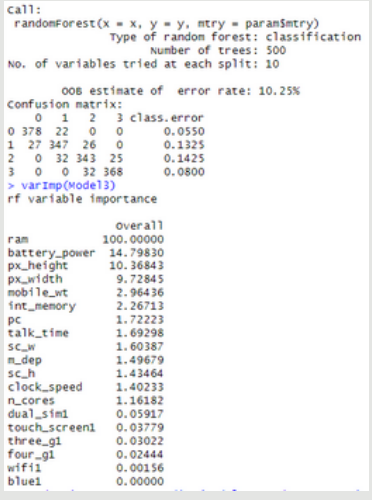
**Feature Importance Analysis** Feature importance ranking highlights the most influential attributes in the prediction:

- **RAM (100.00)** – The most significant factor impacting price prediction.
- **Battery Power (14.79)** – A major contributor.
- **Pixel Height & Width (10.36, 9.72)** – Display resolution is influential.
- **Mobile Weight, Internal Memory, and Processor (PC)** – Secondary but relevant factors.

**Conclusion:** The Random Forest model provides robust predictions with high accuracy and reliability. Given the feature importance rankings, enhancing mobile specifications such as RAM and battery power can significantly impact the predicted price range. The low misclassification rates indicate that the model generalizes well across different price categories.



```
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2  3
         0 97  5  0  0
         1  3 93  8  0
         2  0  2 87  7
         3  0  0  5 93

Overall Statistics

               Accuracy : 0.925
                 95% CI : (0.8947, 0.9488)
    No Information Rate : 0.25
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3
Sensitivity            0.9700   0.9300   0.8700   0.9300
Specificity            0.9833   0.9633   0.9700   0.9833
Pos Pred Value         0.9510   0.8942   0.9062   0.9490
Neg Pred Value         0.9899   0.9764   0.9572   0.9768
Prevalence             0.2500   0.2500   0.2500   0.2500
Detection Rate         0.2425   0.2325   0.2175   0.2325
Detection Prevalence   0.2550   0.2600   0.2400   0.2450
Balanced Accuracy      0.9767   0.9467   0.9200   0.9567
> |
```



```
call:
 randomForest(x = x, y = y, mtry = param$mtry)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 10

        OOB estimate of  error rate: 10.25%
Confusion matrix:
    0   1   2   3 class.error
0 378  22   0   0      0.0550
1  27 347  26   0      0.1325
2   0  32 343  25      0.1425
3   0   0  32 368      0.0800
> varImp(Model3)
rf variable importance

                Overall
ram          100.00000
battery_power  14.79830
px_height      10.36843
px_width        9.72845
mobile_wt       2.96436
int_memory      2.26713
pc              1.72223
talk_time       1.69298
sc_w            1.60387
m_dep           1.49679
sc_h            1.43464
clock_speed     1.40233
n_cores         1.16182
dual_sim1       0.05917
touch_screen1   0.03779
three_g1        0.03022
four_g1         0.02444
wifi1           0.00156
blue1           0.00000
```

# COMPARISON OF NAÏVE BAYES, DECISION TREE, AND RANDOM FOREST MODELS

| Feature | Naïve Bayes | Decision Tree | Random Forest |
|---|---|---|---|
| **Methodology** | Uses probability-based classification. | Splits data into branches based on features. | Constructs multiple decision trees and averages results. |
| **Accuracy** | 66% (Lowest) | 80.25% (Moderate) | 92.5% (Highest) |
| **Overfitting Risk** | Low | High | Low |
| **Interpretability** | High (Simple calculations) | Medium (Tree structure is understandable) | Low (Complex due to multiple trees) |
| **Processing Speed** | Fast (Minimal computation) | Moderate (Grows with data size) | Slower (Requires more computation) |
| **Best Use Case** | Simple problems with independent features. | When clear decision rules are needed. | When high accuracy and generalization are required. |
| **Limitations** | Assumes feature independence, which may not hold in real-world data. | Overfits easily, leading to poor generalization. | Computationally expensive and harder to interpret. |

**Conclusion:**
For applications requiring high accuracy and robustness, Random Forest is the preferred choice. While Decision Trees provide interpretability, they are prone to overfitting. Naïve Bayes works well for simple problems but lacks the predictive power of more advanced models. Given the emphasis on accuracy, Random Forest is the most suitable model for reliable performance.