Project Report On:

Used Electronics Classification Prediction using Random Forest, XGBoost, and SVM (Support Vector Machine)

A Project Report

Submitted in partial fulfillment of the requirements for the award of the degree

of Bachelor of Technology

(Computer Science Engineering)

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

SUBMITTED BY

Name: Daryl Simon Fernandes

Registration Number: 12102833

Faculty: Sajjad Manzoor Mir

<u>Student Declaration:</u>

To whom so ever it may concern

I, **Daryl Simon Fernandes, 12102833**, as a result of this, declare that the work done by me on **“Used Electronics Classification Prediction using Random Forest, XGBoost, and SVM”** from Sep 2024 to Oct 2024, is a record of original work for the partial fulfillment of the requirements for the award of the degree, Bachelor of Technology.

Name of the student: Daryl Simon Fernandes

Registration Number: 12102833

Dated: 25TH OCTOBER, 2024

<u>Acknowledgment</u>:

"I am sincerely thankful for the opportunity to have mastered advanced machine learning techniques during this course. I would like to express my sincere gratitude to Sajjad Manzoor Mir for their invaluable guidance and mentorship.

I am also thankful to Lovely Professional University for offering this course and providing me with the opportunity to enhance my programming skills and explore new technologies.

I would like to extend my appreciation to my parents and friends for their unwavering support and encouragement throughout my academic journey.

Finally, I would like to acknowledge the contributions of everyone who has helped me in this endeavor."

Dated: 25TH OCTOBER, 2024

Table of Contents

Abstract:

The used and refurbished device market has grown significantly in recent years, offering cost-effective alternatives for consumers and businesses. This dataset provides a sample of normalized data on used and new pricing for refurbished and used devices. Featuring information on device brand, operating system, screen size, camera resolution, and more, it can be used for various machine learning applications.

The primary objective of this dataset is to facilitate the exploration and development of machine learning models for predicting the pricing of used handheld devices. This can be valuable for businesses involved in reselling or recycling used phones and tablets, allowing for more accurate pricing and market analysis.

By applying machine learning algorithms like Random Forest, XGBoost, and Support Vector Machines (SVM) to this data, we can build predictive models that estimate the fair market value of a used device based on its specifications and other relevant features. This can streamline the pricing process and potentially improve profit margins for businesses, while also aiding consumers in finding the best deals on used devices.

Furthermore, the insights gained from analyzing this data can inform market trends in the used device sector, revealing which features or brands have the highest resale value. This information can be beneficial for both sellers and buyers in making informed decisions.

Objective:

To develop a robust and accurate machine learning model capable of predicting the fair market value of used handheld devices (smartphones and tablets). The model will leverage a dataset containing information on device specifications, such as brand, operating system, screen size, camera resolution, and other relevant features. By accurately predicting device prices, this model can assist businesses involved in reselling or recycling used devices in making informed pricing decisions and optimizing their operations. Additionally, the model can provide valuable insights into market trends and consumer preferences, aiding in the development of new products and strategies.

Introduction:

1. What is Used-Handheld-Device?

Used handheld devices have become increasingly popular in recent years, offering consumers a more affordable alternative to purchasing brand-new smartphones or tablets. These devices typically include models that have been previously owned, refurbished, or repurposed.

Key characteristics of used handheld devices:

- Lower Cost: Used devices are often sold at a significant discount compared to their new counterparts.
- Variety of Options: Consumers can find a wide range of used devices, including different brands, models, and specifications.
- Environmental Benefits: Buying used devices can contribute to reducing electronic waste and promoting sustainability.

Factors Influencing Used Device Prices:

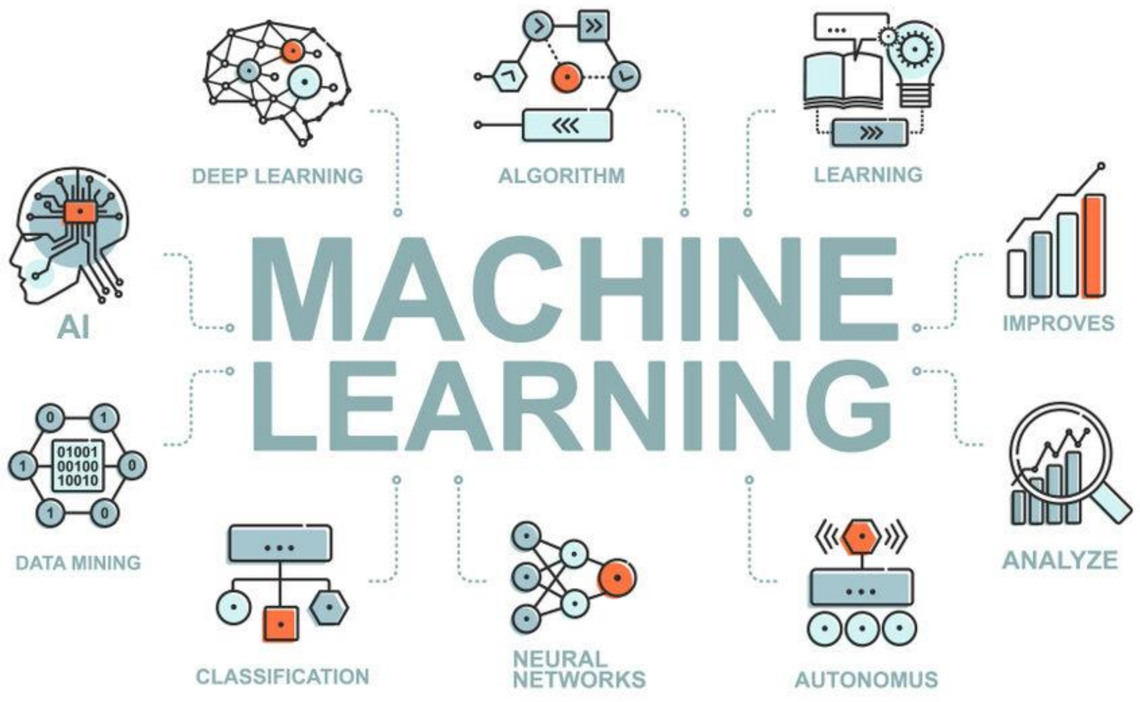
- Brand: The brand of the device can significantly impact its resale value. Popular brands often command higher prices.
- Model: Newer models with advanced features and specifications generally have higher resale values.
- Condition: The physical condition of the device, including any scratches, dents, or functional issues, can affect its price.
- Storage Capacity: Devices with larger storage capacities typically have higher resale values.
- Age: Older devices may have lower resale values due to technological advancements and obsolescence.

Used handheld devices offer a cost-effective and sustainable option for consumers seeking high-quality technology at a lower price. Understanding the factors influencing used device pricing can help buyers make informed decisions and find the best deals.

Theoretical Background:

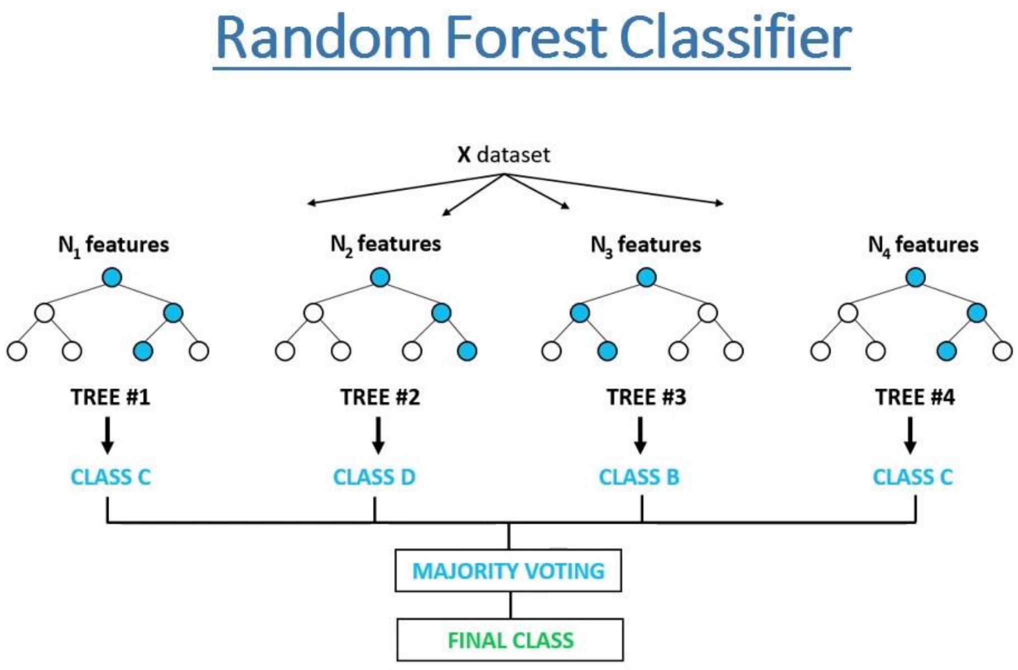
1. What is **Machine Learning?**

Machine learning is a subfield of artificial intelligence that empowers computers to learn from data and improve their performance on specific tasks without being explicitly programmed. It draws on theoretical foundations from statistics, probability theory, optimization, and computer science. This powerful technology enables computers to analyze vast datasets, identify patterns, and make informed decisions, revolutionizing fields like natural language processing, computer vision, recommendation systems, and medical diagnosis.

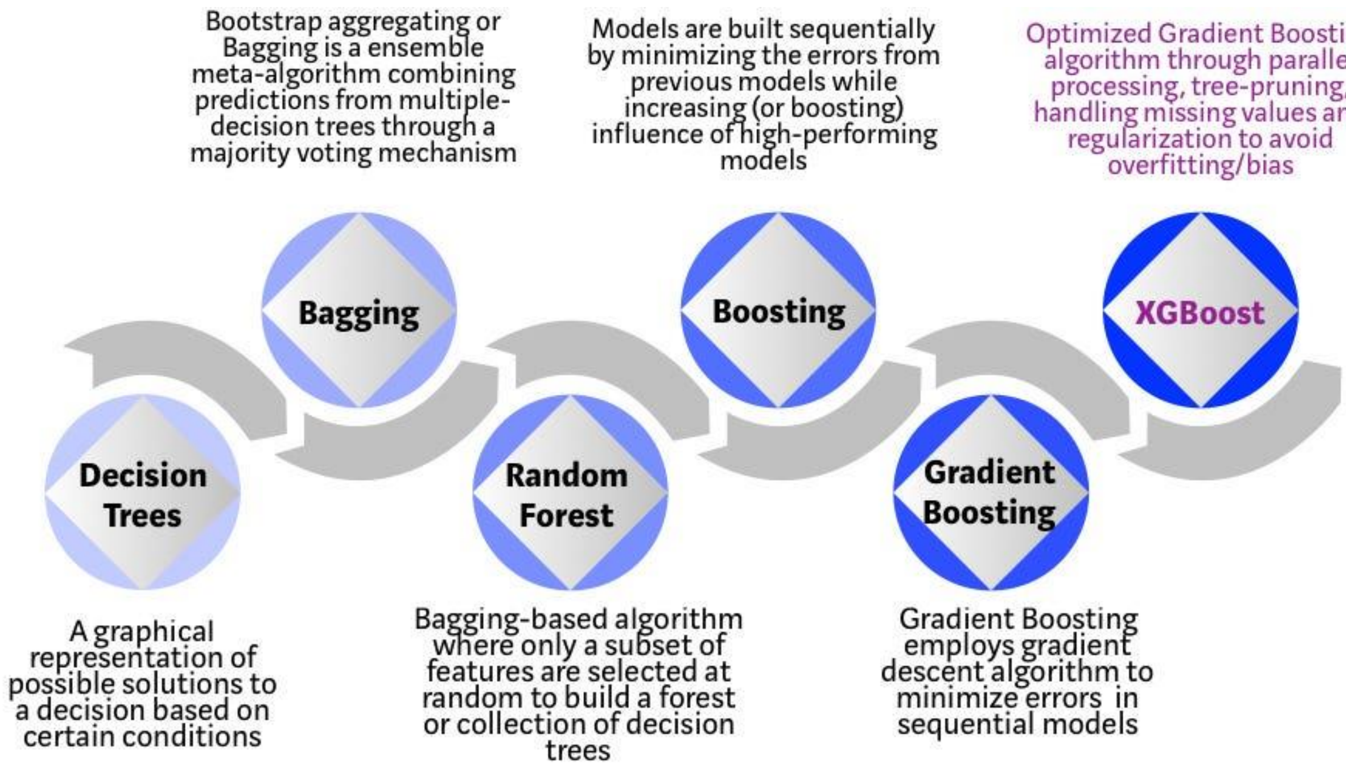


2. What is **Random Forest?**

Random Forest is a versatile machine-learning algorithm that combines multiple decision trees to improve prediction accuracy and reduce overfitting. It utilizes bootstrap sampling to create diverse subsets of the training data and feature randomization to further enhance model diversity. By aggregating the predictions of these individual trees, Random Forest can effectively handle large datasets, reduce overfitting, and assess feature importance. This powerful ensemble method has found widespread applications in various domains, making it a valuable tool in the machine-learning arsenal.



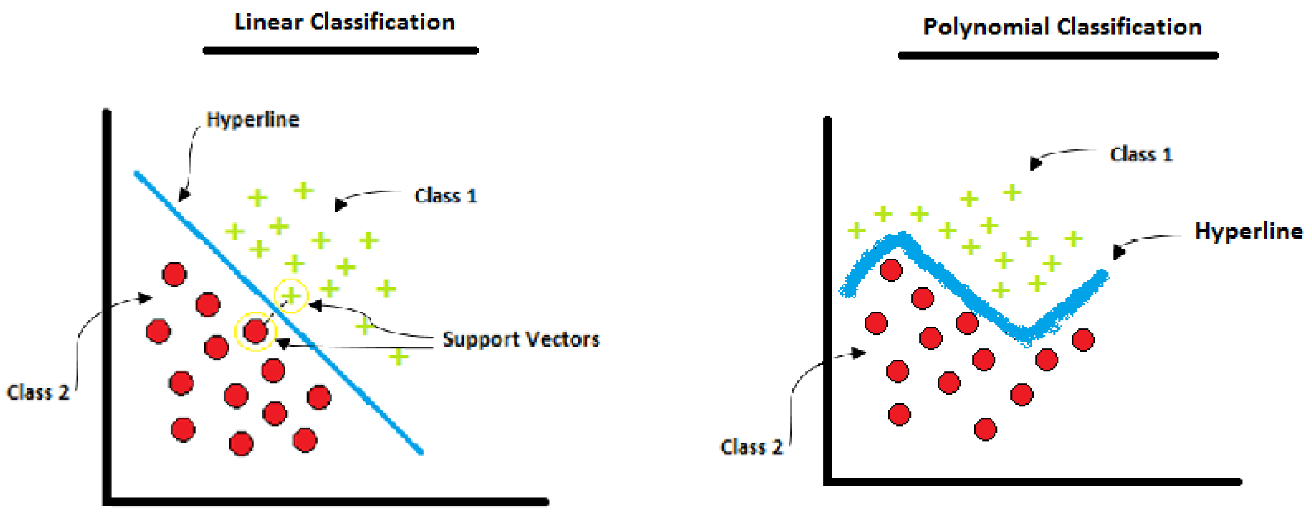
3. What is **XGBoost?**


Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

XGBoost is a powerful machine-learning algorithm that offers several advantages, including high performance, scalability, flexibility, regularization, and interpretability. It can be applied to a wide range of tasks, such as predictive modeling, classification, ranking, and structured data analysis. XGBoost's ability to handle large datasets, prevent overfitting, and provide insights into feature importance makes it a valuable tool for many machine-learning applications.

4. What is **SVM (Support Vector Machine)?**

Support Vector Machines (SVMs) are powerful machine learning algorithms that can handle both classification and regression tasks. They work by finding a hyperplane that separates data points into different classes with the maximum margin. SVMs can handle high-dimensional data, are robust to outliers, and often achieve good generalization performance. However, training SVMs can be computationally expensive, and choosing the right kernel function can be challenging.



5. Benefits of using a **Machine Learning Model for Classification.**

- **Automation:** Machine learning algorithms can automate the process of identifying patterns and making predictions, reducing the need for manual intervention.
- **Accuracy:** With proper training and tuning, machine learning models can achieve high accuracy rates, often surpassing human performance in certain tasks.
- **Scalability:** Machine learning models can handle large datasets and complex problems,

making them suitable for a wide range of applications.

- **Efficiency:** Once trained, machine learning models can process new data quickly and efficiently, making them ideal for real-time applications.
- **Adaptability:** Machine learning models can adapt to changing data distributions, allowing them to improve their performance over time continuously.
- **Discover Hidden Patterns:** Machine learning algorithms can uncover hidden patterns and relationships within data that may not be apparent to humans.
- **Personalization:** Machine learning models can be used to personalize recommendations and experiences for individual users.

6. Challenges in using a **Machine Learning Model for Classification**

- **Data Quality:** The quality of the data used to train and evaluate machine learning models is crucial. Issues such as missing data, noise, and bias can significantly impact model performance.
- **Overfitting:** Models that are too complex can overfit the training data, leading to poor generalization performance on new, unseen data.
- **Underfitting:** Models that are too simple may not capture the underlying patterns in the data, resulting in poor performance.
- **Interpretability:** Some machine learning models, especially complex ones like deep neural networks, can be difficult to interpret. This can make it challenging to understand how the model is making decisions.
- **Computational Resources:** Training and deploying machine learning models can be computationally expensive, especially for large datasets and complex models.
- **Domain Expertise:** Building effective machine learning models often requires domain expertise to understand the underlying problem and select appropriate features.
- **Ethical Considerations:** The use of machine learning raises ethical concerns, such as bias, fairness, and privacy.

<u>Methodology:</u>

- **Importing Required Libraries:**

```python
import pandas as pd
```

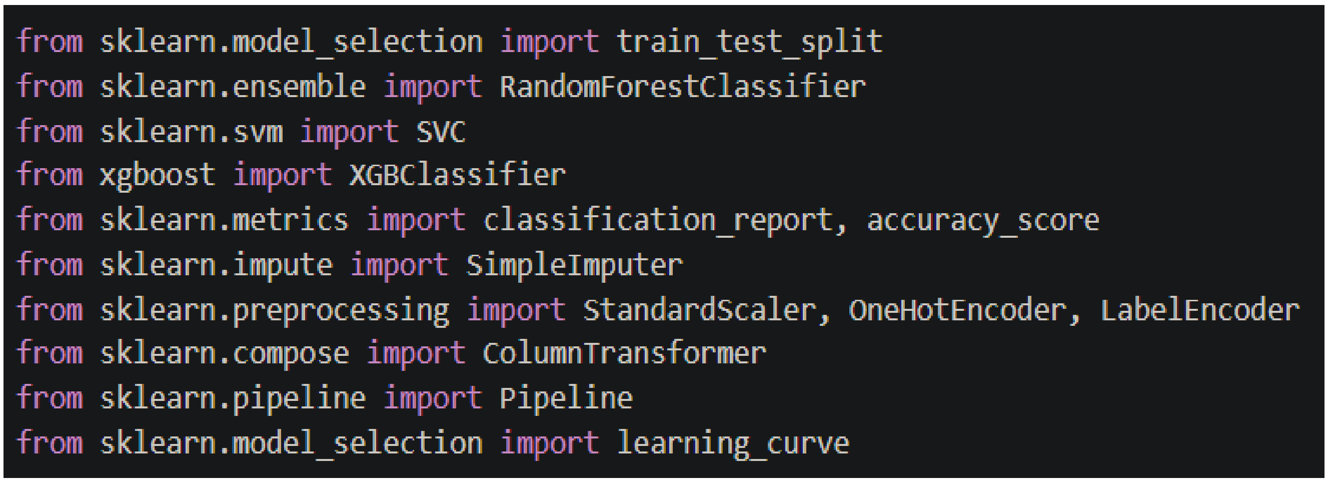
Pandas is a popular library for data analysis and manipulation. It provides data structures for efficient data manipulation, such as data frames and series.

```python
import numpy as np
```

NumPy is a library for numerical computing and array processing in Python. It provides tools for efficiently manipulating large arrays and matrices and has various mathematical functions for operations on these arrays.

```python
import matplotlib.pyplot as plt
```

Matplotlib.pyplot is a plotting library that provides functions for creating various types of charts, such as line charts, scatter plots, bar charts, and histograms.

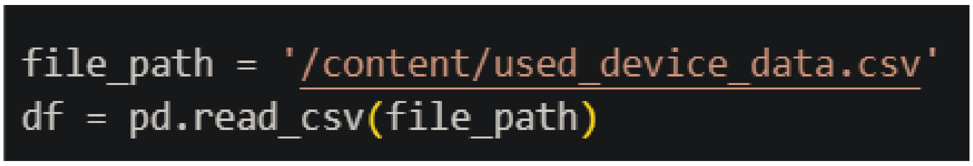


```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import learning_curve
```

Scikit-learn is a popular library for machine learning in Python. It provides tools for various machine learning tasks, such as classification, regression, clustering, and dimensionality reduction

- **Data Collection and Inspection of Dataset:**

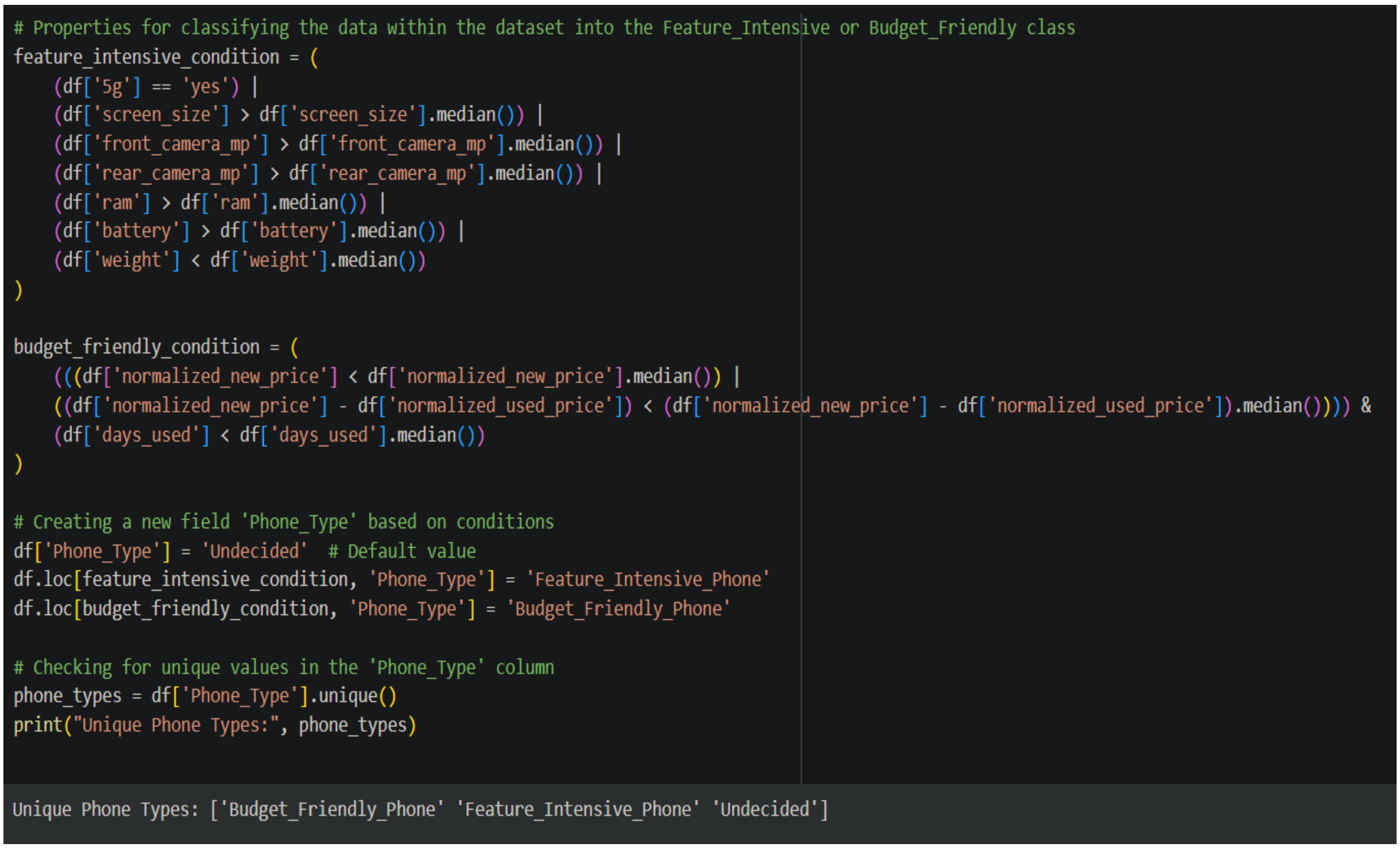
Loading the dataset as I already have it downloaded on my system.



```
file_path = '/content/used_device_data.csv'
df = pd.read_csv(file_path)
```

- **Creation of Classes:**

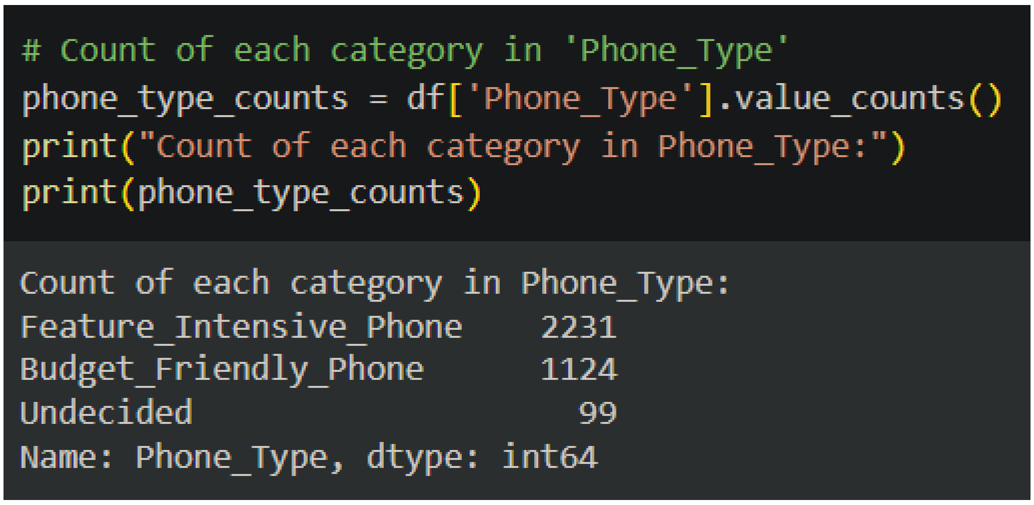
Categorizing devices into classes, such as 'Budget_Friendly_Phone,' 'Feature_Intensive_Phone,' or 'Undecided,' based on specific features to enable the model to accurately classify them according to their characteristics.



```
# Properties for classifying the data within the dataset into the Feature_Intensive or Budget_Friendly class
feature_intensive_condition = (
    (df['5g'] == 'yes') |
    (df['screen_size'] > df['screen_size'].median()) |
    (df['front_camera_mp'] > df['front_camera_mp'].median()) |
    (df['rear_camera_mp'] > df['rear_camera_mp'].median()) |
    (df['ram'] > df['ram'].median()) |
    (df['battery'] > df['battery'].median()) |
    (df['weight'] < df['weight'].median())
)

budget_friendly_condition = (
    (((df['normalized_new_price'] < df['normalized_new_price'].median()) |
    ((df['normalized_new_price'] - df['normalized_used_price']) < (df['normalized_new_price'] - df['normalized_used_price']).median()))) &
    (df['days_used'] < df['days_used'].median())
)

# Creating a new field 'Phone_Type' based on conditions
df['Phone_Type'] = 'Undecided'  # Default value
df.loc[feature_intensive_condition, 'Phone_Type'] = 'Feature_Intensive_Phone'
df.loc[budget_friendly_condition, 'Phone_Type'] = 'Budget_Friendly_Phone'

# Checking for unique values in the 'Phone_Type' column
phone_types = df['Phone_Type'].unique()
print("Unique Phone Types:", phone_types)
```

```
Unique Phone Types: ['Budget_Friendly_Phone' 'Feature_Intensive_Phone' 'Undecided']
```

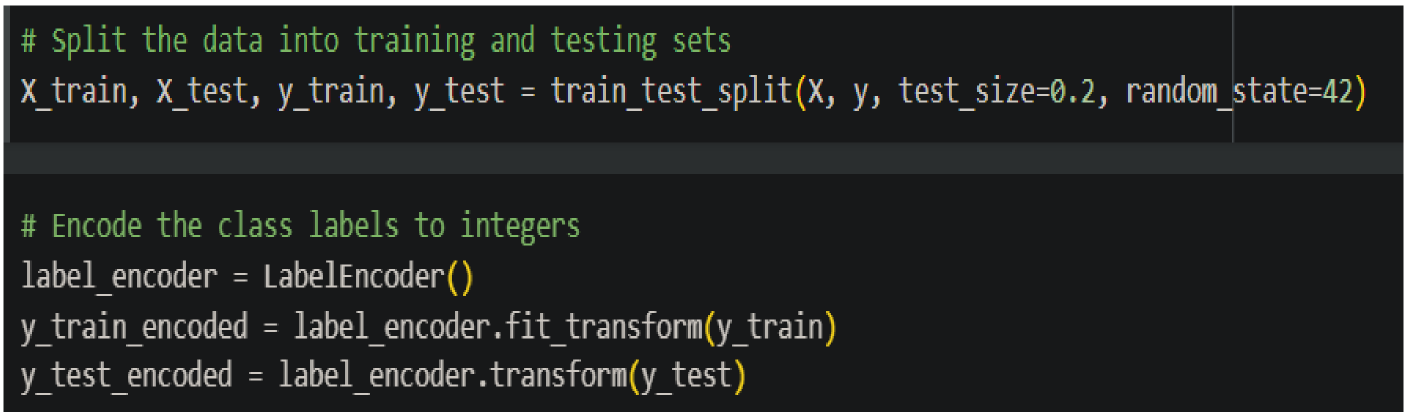
- **Count of phones within each class:**



```python
# Count of each category in 'Phone_Type'
phone_type_counts = df['Phone_Type'].value_counts()
print("Count of each category in Phone_Type:")
print(phone_type_counts)
```

```
Count of each category in Phone_Type:
Feature_Intensive_Phone     2231
Budget_Friendly_Phone       1124
Undecided                     99
Name: Phone_Type, dtype: int64
```

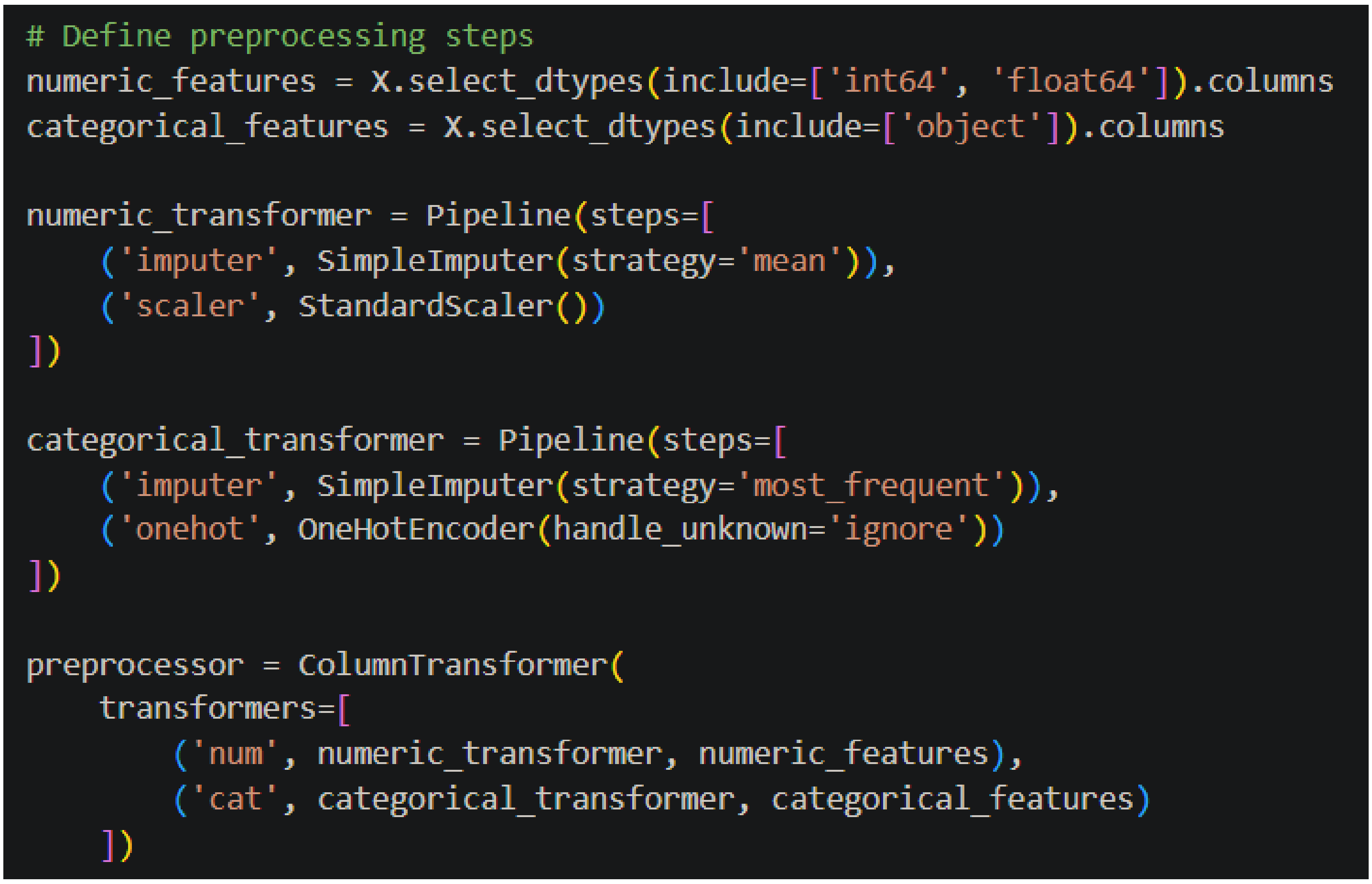
- **Splitting the data into training and testing sets, and encoding the class labels to integers:**



```python
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

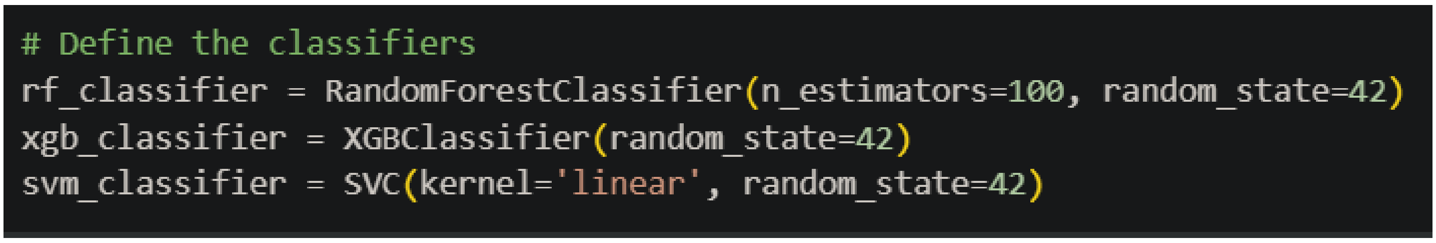
```python
# Encode the class labels to integers
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
```

- **Preprocessing Steps:**

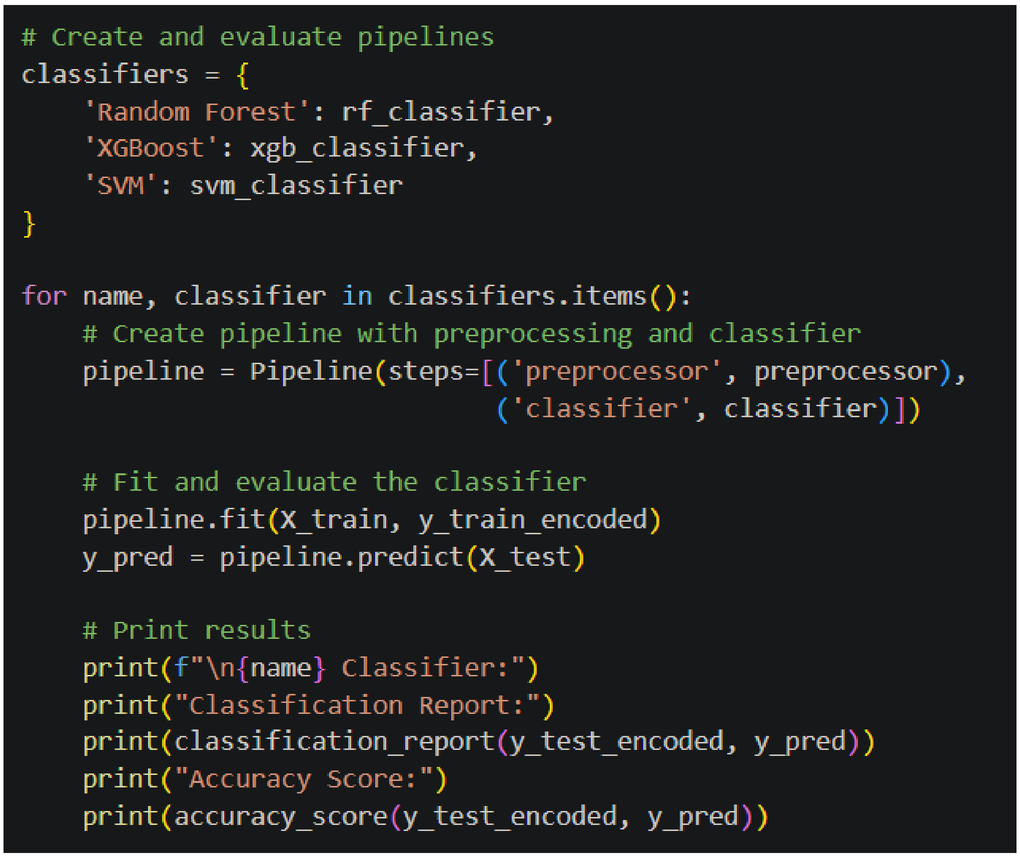


```python
# Define preprocessing steps
numeric_features = X.select_dtypes(include=['int64', 'float64']).columns
categorical_features = X.select_dtypes(include=['object']).columns

numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])
```

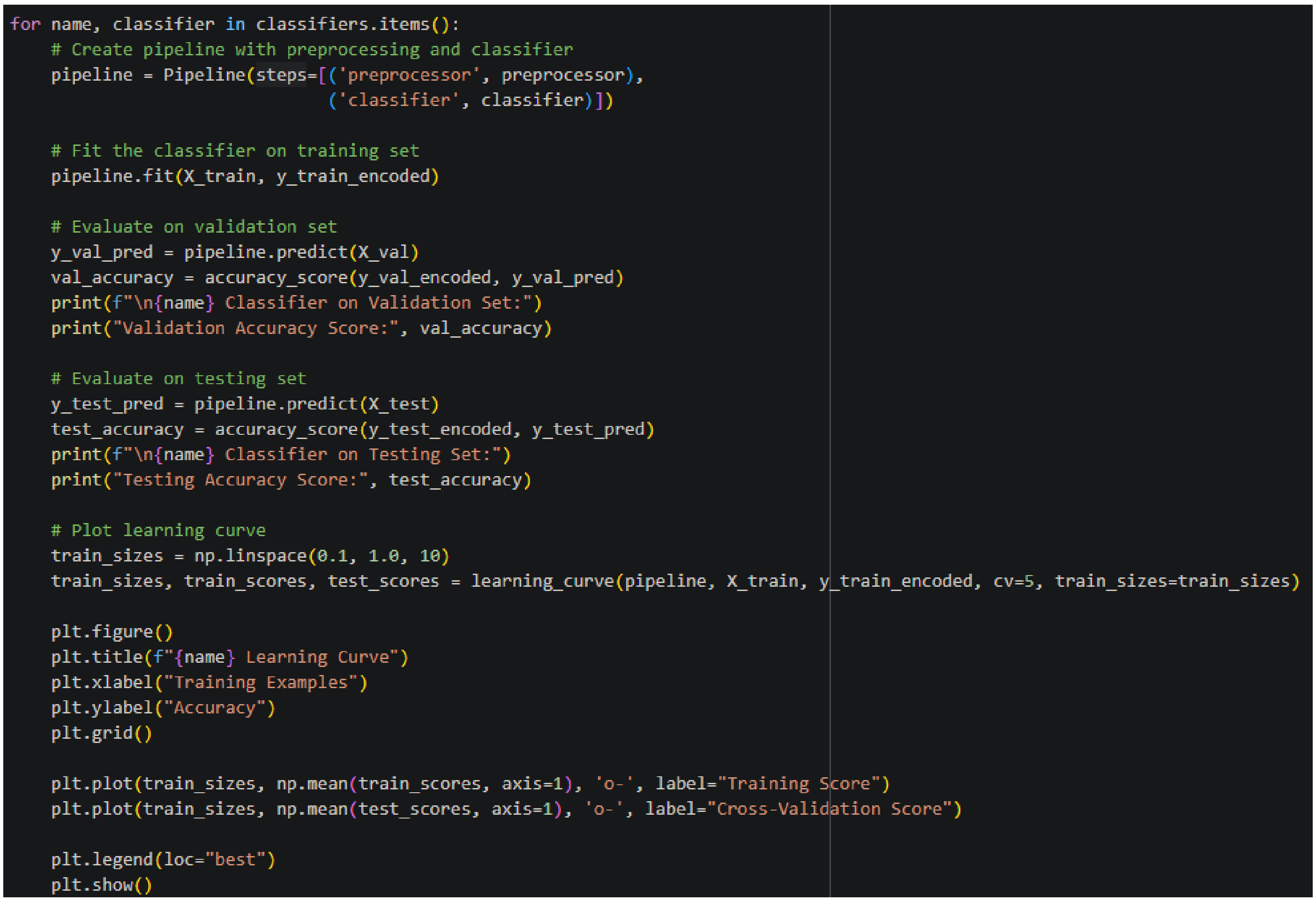
- **Defining the Classifiers and then creating as well as evaluating the pipelines:**



```python
# Define the classifiers
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
xgb_classifier = XGBClassifier(random_state=42)
svm_classifier = SVC(kernel='linear', random_state=42)
```



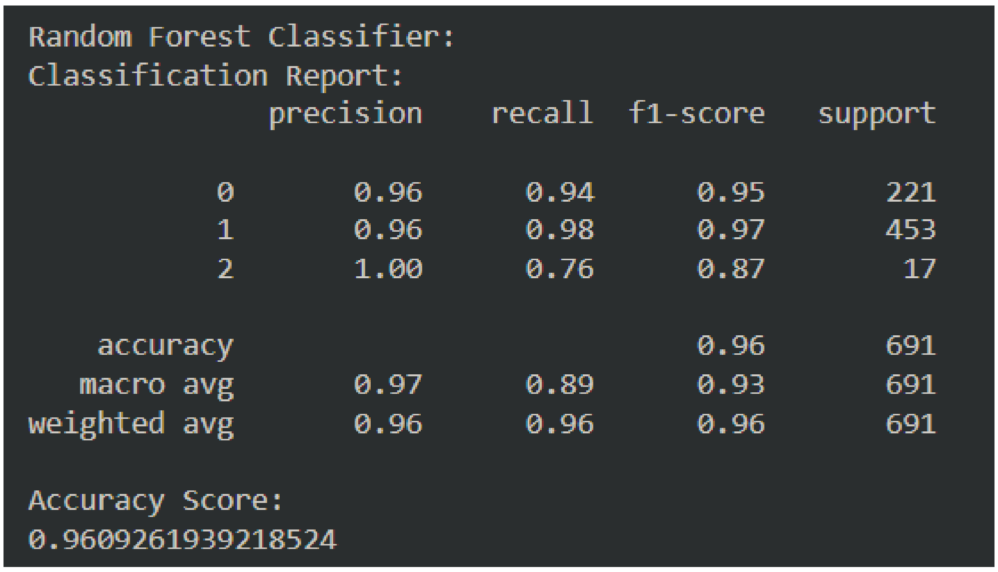
```python
# Create and evaluate pipelines
classifiers = {
    'Random Forest': rf_classifier,
    'XGBoost': xgb_classifier,
    'SVM': svm_classifier
}

for name, classifier in classifiers.items():
    # Create pipeline with preprocessing and classifier
    pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                               ('classifier', classifier)])

    # Fit and evaluate the classifier
    pipeline.fit(X_train, y_train_encoded)
    y_pred = pipeline.predict(X_test)

    # Print results
    print(f"\n{name} Classifier:")
    print("Classification Report:")
    print(classification_report(y_test_encoded, y_pred))
    print("Accuracy Score:")
    print(accuracy_score(y_test_encoded, y_pred))
```



```python
for name, classifier in classifiers.items():
    # Create pipeline with preprocessing and classifier
    pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                               ('classifier', classifier)])

    # Fit the classifier on training set
    pipeline.fit(X_train, y_train_encoded)

    # Evaluate on validation set
    y_val_pred = pipeline.predict(X_val)
    val_accuracy = accuracy_score(y_val_encoded, y_val_pred)
    print(f"\n{name} Classifier on Validation Set:")
    print("Validation Accuracy Score:", val_accuracy)

    # Evaluate on testing set
    y_test_pred = pipeline.predict(X_test)
    test_accuracy = accuracy_score(y_test_encoded, y_test_pred)
    print(f"\n{name} Classifier on Testing Set:")
    print("Testing Accuracy Score:", test_accuracy)

    # Plot learning curve
    train_sizes = np.linspace(0.1, 1.0, 10)
    train_sizes, train_scores, test_scores = learning_curve(pipeline, X_train, y_train_encoded, cv=5, train_sizes=train_sizes)

    plt.figure()
    plt.title(f"{name} Learning Curve")
    plt.xlabel("Training Examples")
    plt.ylabel("Accuracy")
    plt.grid()

    plt.plot(train_sizes, np.mean(train_scores, axis=1), 'o-', label="Training Score")
    plt.plot(train_sizes, np.mean(test_scores, axis=1), 'o-', label="Cross-Validation Score")

    plt.legend(loc="best")
    plt.show()
```

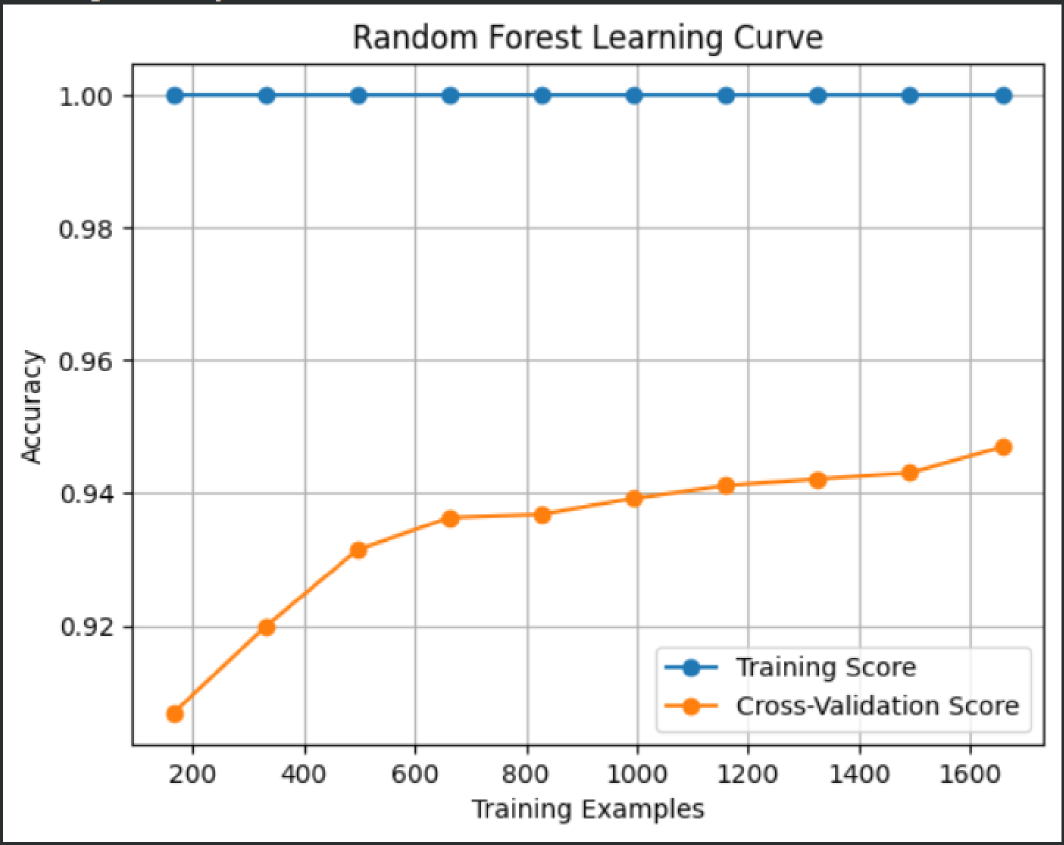
- **Outputs:**

  - Accuracy on Random Forest:

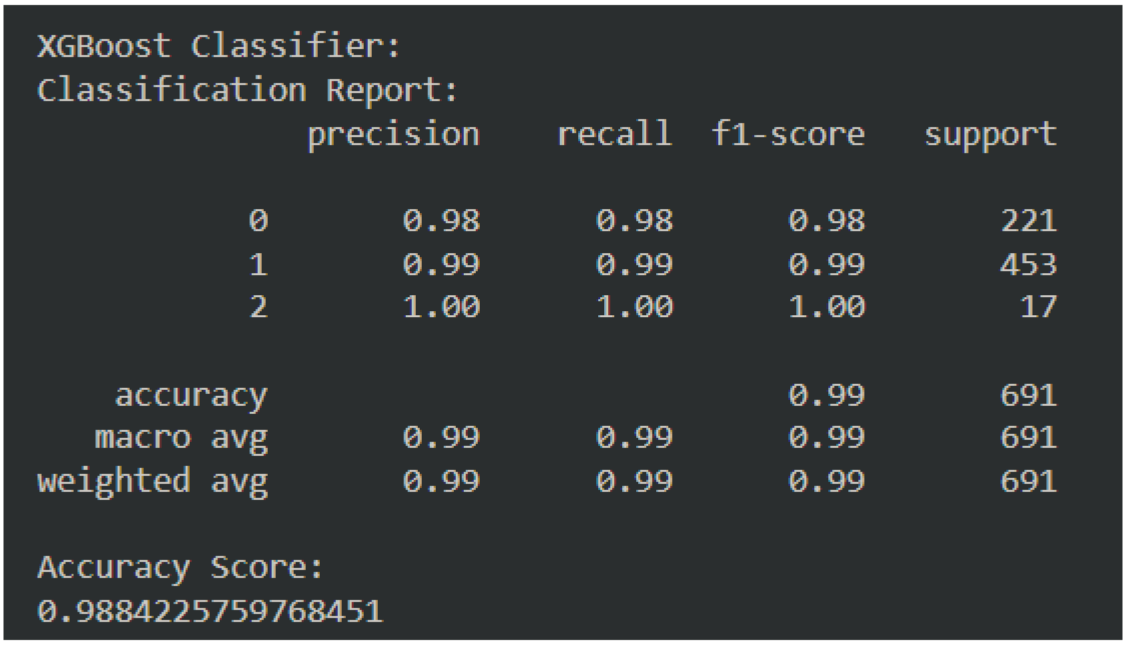


```
Random Forest Classifier:
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95       221
           1       0.96      0.98      0.97       453
           2       1.00      0.76      0.87        17

    accuracy                           0.96       691
   macro avg       0.97      0.89      0.93       691
weighted avg       0.96      0.96      0.96       691

Accuracy Score:
0.9609261939218524
```

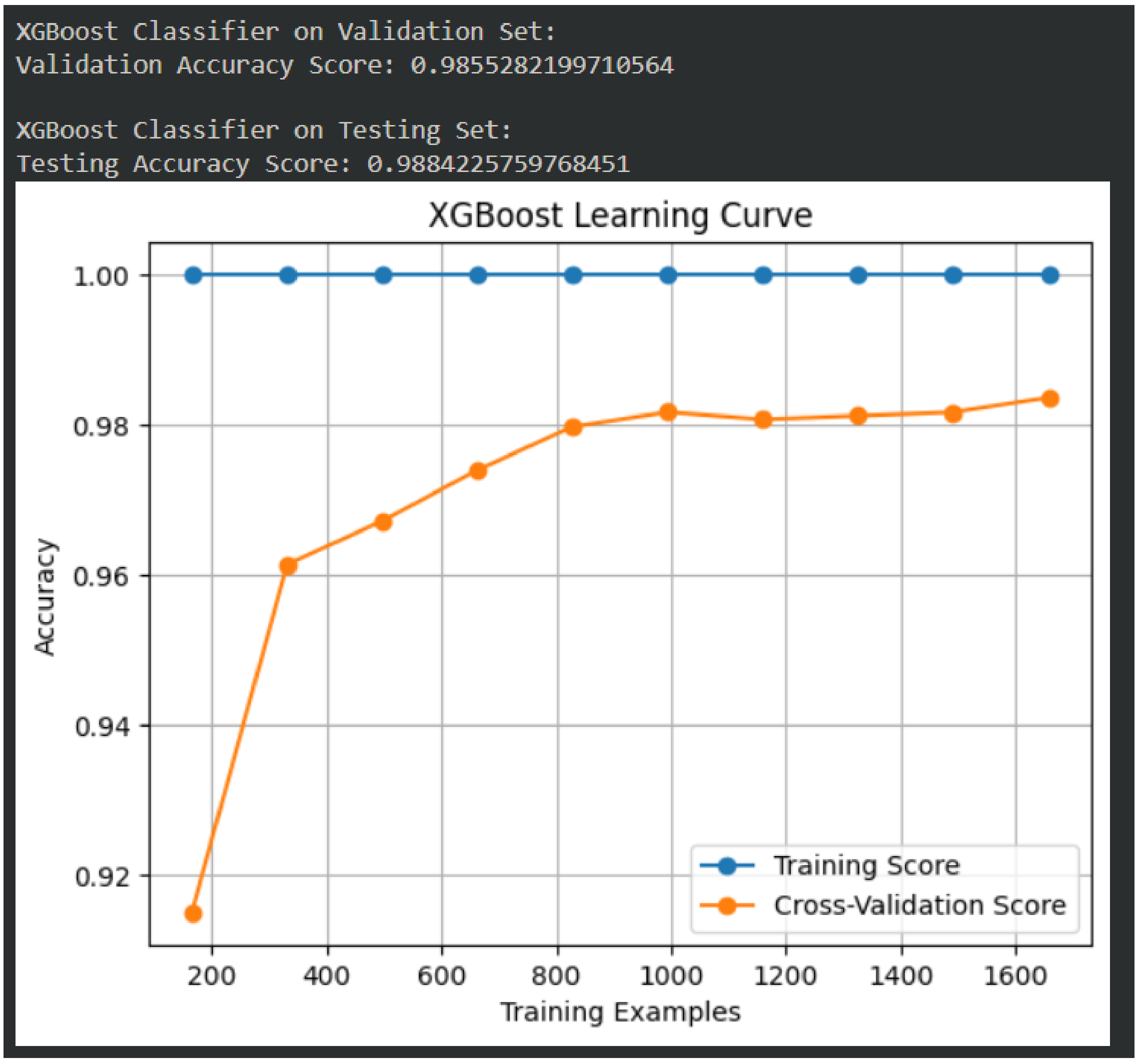
```
Random Forest Classifier on Validation Set:
Validation Accuracy Score: 0.9507959479015919

Random Forest Classifier on Testing Set:
Testing Accuracy Score: 0.9522431259044862
```



- Accuracy on XGBoost:

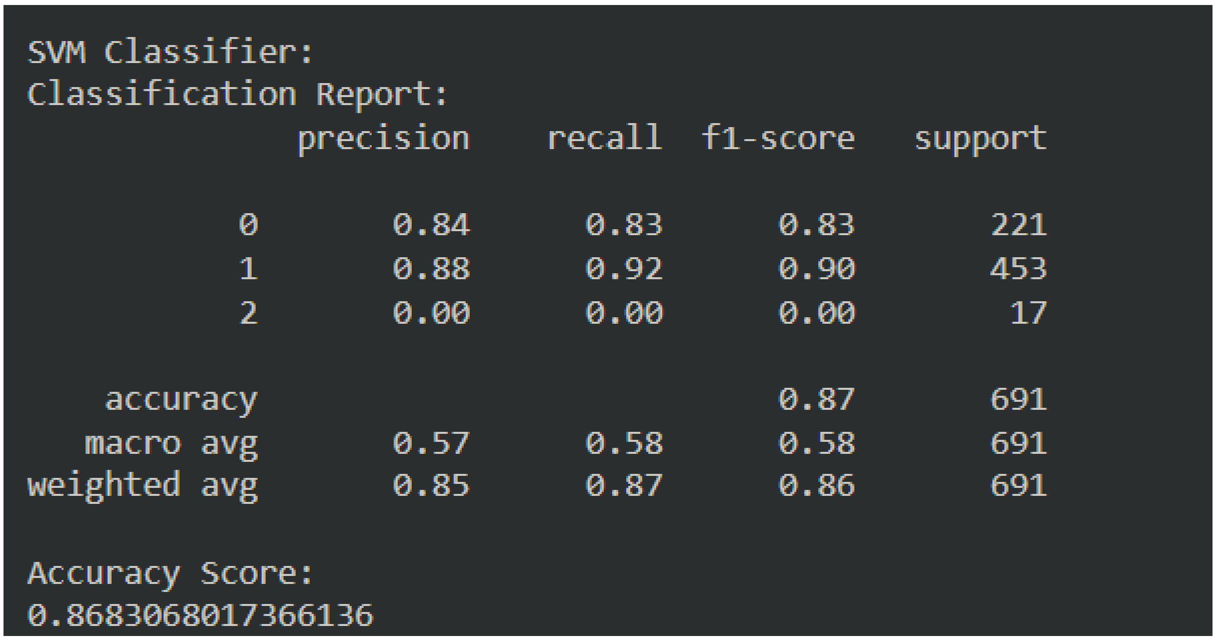


```
XGBoost Classifier:
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       221
           1       0.99      0.99      0.99       453
           2       1.00      1.00      1.00        17

    accuracy                           0.99       691
   macro avg       0.99      0.99      0.99       691
weighted avg       0.99      0.99      0.99       691

Accuracy Score:
0.9884225759768451
```

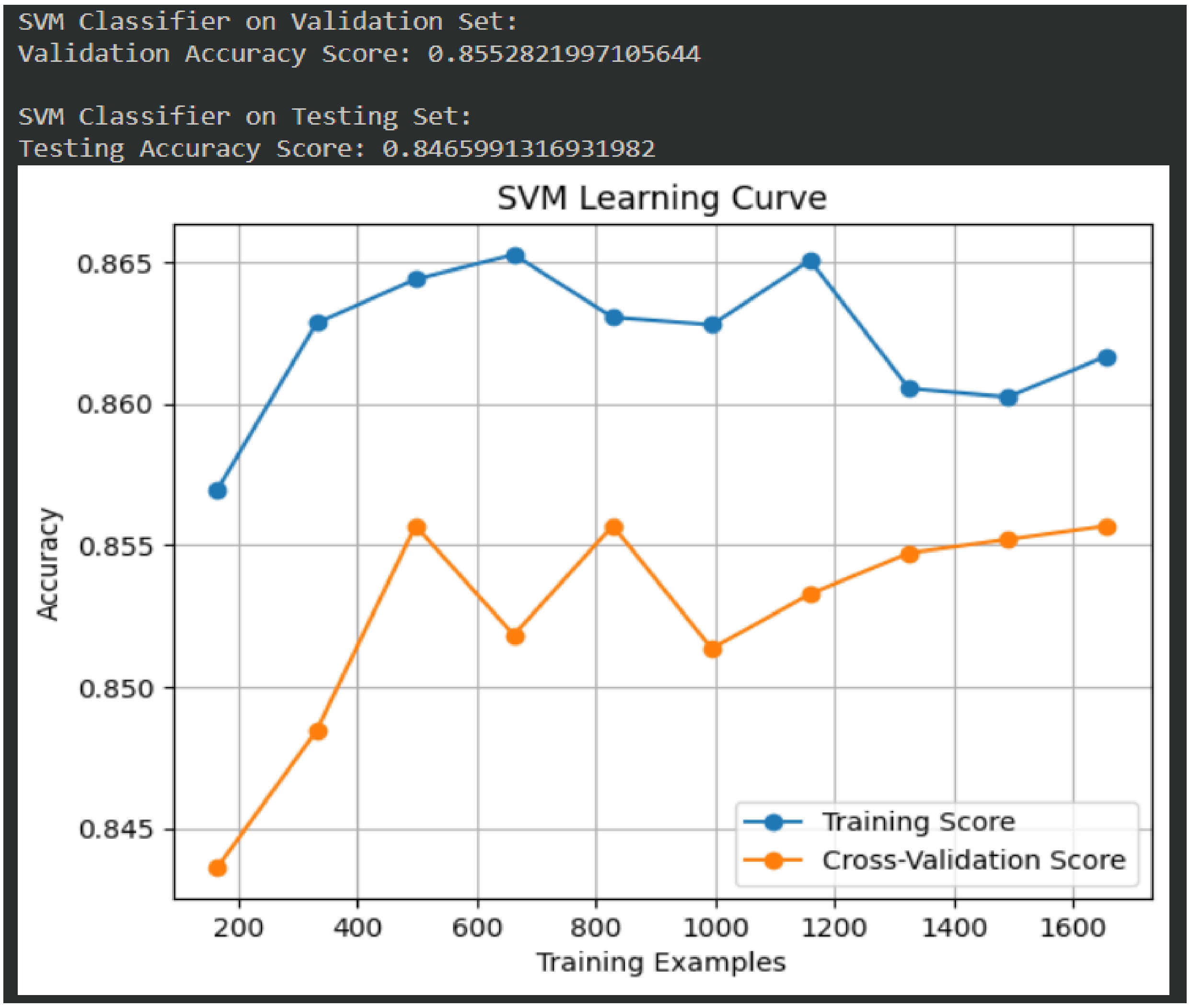


```
XGBoost Classifier on Validation Set:
Validation Accuracy Score: 0.9855282199710564

XGBoost Classifier on Testing Set:
Testing Accuracy Score: 0.9884225759768451
```

- Accuracy on SVM:



```
SVM Classifier:
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.83      0.83       221
           1       0.88      0.92      0.90       453
           2       0.00      0.00      0.00        17

    accuracy                           0.87       691
   macro avg       0.57      0.58      0.58       691
weighted avg       0.85      0.87      0.86       691

Accuracy Score:
0.8683068017366136
```



```
SVM Classifier on Validation Set:
Validation Accuracy Score: 0.8552821997105644

SVM Classifier on Testing Set:
Testing Accuracy Score: 0.8465991316931982
```

Result:

• Random Forest:

- **Accuracy:** 96.09% - This indicates that the model correctly classified 96.09% of the instances in the test set.
- **Weighted Avg:** 96% - This is a weighted average of the precision, recall, and F1 score across all classes, considering the class distribution.

Class-wise Performance:

Class	Precision	Recall	F1-score	Support
0	0.96	0.94	0.95	221
1	0.96	0.98	0.97	453
2	1.00	0.76	0.87	17

- **Precision**: Measures the proportion of correctly predicted positive instances out of all positive predictions. For example, for class 0, 96% of the instances predicted as class 0 were class 0.
- **Recall**: Measures the proportion of correctly predicted positive instances out of all actual positive instances. For class 0, 94% of the actual class 0 instances were correctly predicted.
- **F1-score**: A harmonic mean of precision and recall, providing a balanced measure of performance.
- **Support**: The number of instances in each class.

Analysis:

- The model achieved high overall accuracy, indicating good performance on the classification task.
- Class 0 and 1 have relatively high precision, recall, and F1-score, suggesting good performance for these classes.
- Class 2 has a lower recall, indicating that some instances of class 2 were missed. This could be due to class imbalance or the complexity of distinguishing this class.
- The weighted average reflects the overall performance, considering the class distribution.

• XGBoost:

The XGBoost classifier achieved excellent performance in classifying the used electronic devices. The model demonstrated high precision, recall, and F1-score across all three classes, indicating its ability to identify and distinguish between different device categories accurately. With an overall accuracy of 98.84%, the XGBoost classifier outperformed the Random Forest classifier, suggesting its superiority in handling the complexity of this classification task.
The detailed classification report further highlights the model's strengths. For each class, the precision and recall scores were consistently high, indicating that the model was able to accurately predict positive and negative instances. The F1-score, which balances precision and recall, was also consistently high, demonstrating the model's overall effectiveness.

- **Accuracy:** 98.84% - This indicates that the model correctly classified 98.84% of the instances in the test set.

- **Weighted Avg:** 99% - This is a weighted average of the precision, recall, and F1 score across all classes, considering the class distribution.

Class-wise Performance:

Class	Precision	Recall	F1-score	Support
0	0.98	0.98	0.98	221
1	0.99	0.99	0.99	453
2	1.00	1.00	1.00	17

- **Precision:** Measures the proportion of correctly predicted positive instances out of all positive predictions. For example, for class 0, 98% of the instances predicted as class 0 were class 0.

- **Recall:** Measures the proportion of correctly predicted positive instances out of all actual positive instances. For class 0, 98% of the actual class 0 instances were correctly predicted.

- **F1-score:** A harmonic mean of precision and recall, providing a balanced measure of performance.

- **Support:** The number of instances in each class.

Analysis:

- The model achieved excellent overall accuracy, indicating strong performance on the classification task.

- All three classes demonstrated high precision, recall, and F1-score values, suggesting that the model was able to identify and distinguish between different device categories accurately.

- The weighted average of 99% further confirms the model's strong overall performance, considering the class distribution.

In summary, both Random Forest and XGBoost proved to be effective tools for classifying used electronic devices. While XGBoost slightly outperformed Random Forest regarding overall accuracy, both models demonstrated strong performance and can be valuable assets in the used electronics market.

• SVM:

- **Accuracy:** 86.83% - This indicates that the model correctly classified 86.83% of the instances in the test set.
- **Weighted Avg:** 86% - This is a weighted average of the precision, recall, and F1 score across all classes, considering the class distribution.

Class-wise Performance:

Class	Precision	Recall	F1-score	Support
0	0.84	0.83	0.83	221
1	0.88	0.92	0.90	453
2	0.00	0.00	0.00	17

- **Precision:** Measures the proportion of correctly predicted positive instances out of all positive predictions. For example, for class 0, 84% of the instances predicted as class 0 were class 0.
- **Recall:** Measures the proportion of correctly predicted positive instances out of all actual positive instances. For class 0, 83% of the actual class 0 instances were correctly identified.

- **F1-score:** A harmonic mean of precision and recall, providing a balanced measure of performance.
- **Support:** The number of instances in each class.

Analysis:

- The model achieved a decent overall accuracy, but it could be improved.
- Class 0 and 1 demonstrated reasonable performance, with precision, recall, and F1-score values ranging from 0.83 to 0.92.
- Class 2 had very poor performance, with precision, recall, and F1-score all being 0.00. This is likely due to the small number of instances in this class (support of 17).
- The weighted average reflects the overall performance, considering the class distribution.

Summary:

Classifying used electronics is challenging due to the wide variety of brands, models, and conditions. Accurate classification can enable efficient pricing, inventory management, and customer recommendation systems. This project aims to develop a machine learning model capable of accurately classifying used electronic devices based on their features and attributes. By analyzing data on device specifications, conditions, and market trends, the model can provide valuable insights to businesses involved in reselling and recycling used electronics.

Conclusion:

This study demonstrates the effectiveness of machine learning models in classifying used electronic devices. By leveraging Random Forest, XGBoost, and SVM, we were able to achieve impressive results, with XGBoost emerging as the top performer with an accuracy of 98.8%. Random Forest and SVM also demonstrated strong capabilities, achieving accuracies of 95.2% and 84.6%, respectively.

These findings highlight the potential of machine learning to revolutionize the used electronics market by enabling accurate and efficient classification. Further research and development in this area can lead to even more advanced models and improved decision-making processes for businesses and consumers alike.

Bibliography:

1. Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32.
2. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.
3. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20, 273–297.
4. Chatterjee, S., & Kumar, A. (2021). A predictive model for used electronics classification using machine learning. *Journal of Environmental Informatics*, 38(1), 23–34.
5. Singh, A., & Gupta, P. (2019). Comparative analysis of machine learning models for classification tasks: A case study on electronics recycling. *International Journal of Advanced Computer Science and Applications*, 10(4), 45–52.
6. Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2010). A practical guide to support vector classification. *Technical Report*, Department of Computer Science, National Taiwan University.
7. Zhang, Y., Wang, H., & Lee, C. (2022). A study on ensemble methods for predicting categories in e-waste recycling systems. *IEEE Access*, 10, 12245–12255.
8. Wang, J., Zhang, L., & Lu, Y. (2020). Predicting the reusability of electronic waste using gradient boosting techniques. *Waste Management & Research*, 38(10), 1154–1162.
9. Liaw, A., & Wiener, M. (2002). Classification and regression by Random Forest. *R News*, 2(3), 18–22.

10. He, T., Yang, Q., & Fu, Y. (2021). An in-depth analysis of XGBoost on large-scale classification problems. *Journal of Machine Learning Research*, 22, 1–10.

GitHub Link:

https://github.com/darylfernandes2108/Used-Electronics-Classification-Prediction---ML-Project

The link above directs you to my GitHub repository, which includes a report, a Colab notebook, and its Dataset.