

# Proyecto de ML

Bootcamp Lanbide-BBK 2024

Este es el trabajo final del modulo Machine Learning en The Bridge, del programa Bootcamp 2024 de BBK y Lanbide.

AC by Americo Carrillo





# Objetivos del proyecto

■ Es desarrollar un modelo predictivo para un caso de negocio basado en datos.

El proyecto se divide en fases, incluyendo la obtención y preparación de datos, análisis exploratorio, selección de modelos y evaluación de resultados.

Se deben presentar conclusiones y mejoras, documentando y justificando cada paso para asegurar la reproducibilidad y claridad del proyecto.

# Recopilación y preparación de datos

## Fuentes de datos

Se tomo del portal <https://www.kaggle.com>, el reto llamado Predicción de precios de dispositivos móviles (MobilePricePrediction), creado por **Mohammad Bagher Soroush**.

1. Compuesto por dos data set, llamados test.csv y train.csv
2. Posee 21 campos y 2000 registros.
3. El objetivo es predecir el rango de precios de un teléfono móvil mediante la construcción de un modelo que tenga en cuenta varias características proporcionadas en el conjunto de datos. Y campo price\_range, es la variable Target y usaremos como valor 0: (Costo bajo), 1: (Costo medio), 2: (Costo alto) y 3: (Costo muy alto)

The screenshot shows two code cells in a Jupyter Notebook environment. The top cell contains the command `df_train.head()` and the bottom cell contains `df_test.head()`. Both cells have a green checkmark and a time indicator of "0.0s". The output for each cell is a Pandas DataFrame with 5 rows and 21 columns. The columns are labeled: id, battery\_power, blue, clock\_speed, dual\_sim, fc, four\_g, int\_memory, m\_dep, mobile\_wt, n\_cores, pc, px\_height, px\_width, ram, sc\_h. The data values are numerical, representing various smartphone specifications like battery capacity, screen size, and memory.

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	11
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	10
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8

5 rows × 21 columns

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	...	pc	px_height	px_width	ram	sc_h	
0	1	1043	1	1.8	1	14	0	5	0.1	193	...	16	226	1412	3476	12
1	2	841	1	0.5	1	4	1	61	0.8	191	...	12	746	857	3895	6
2	3	1807	1	2.8	0	1	0	27	0.9	186	...	4	1270	1366	2396	17
3	4	1546	0	0.5	1	18	1	25	0.5	96	...	20	295	1752	3893	10
4	5	1434	0	1.4	0	11	1	49	0.5	108	...	18	749	810	1773	15

5 rows × 21 columns

## Estructura de los datos

Los datos fueron revisados y podemos observar la estructura, compuesta por 2 campos de tipo float64 y 19 campos de tipo int64.

Podemos observar que el fichero test.csv posee un campo adicional llamado id.

**test.csv**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          1000 non-null    int64  
 1   battery_power 1000 non-null  int64  
 2   blue         1000 non-null    int64  
 3   clock_speed  1000 non-null    float64 
 4   dual_sim     1000 non-null    int64  
 5   fc           1000 non-null    int64  
 6   four_g       1000 non-null    int64  
 7   int_memory   1000 non-null    int64  
 8   m_dep        1000 non-null    float64 
 9   mobile_wt    1000 non-null    int64  
 10  n_cores      1000 non-null    int64  
 11  pc           1000 non-null    int64  
 12  px_height   1000 non-null    int64  
 13  px_width    1000 non-null    int64  
 14  ram          1000 non-null    int64  
 15  sc_h         1000 non-null    int64  
 16  sc_w         1000 non-null    int64  
 17  talk_time    1000 non-null    int64  
 18  three_g      1000 non-null    int64  
 19  touch_screen 1000 non-null    int64  
 20  wifi         1000 non-null    int64  
dtypes: float64(2), int64(19)
memory usage: 164.2 KB
```

**train.csv**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   battery_power 2000 non-null  int64  
 1   blue         2000 non-null  int64  
 2   clock_speed  2000 non-null  float64 
 3   dual_sim     2000 non-null  int64  
 4   fc           2000 non-null  int64  
 5   four_g       2000 non-null  int64  
 6   int_memory   2000 non-null  int64  
 7   m_dep        2000 non-null  float64 
 8   mobile_wt    2000 non-null  int64  
 9   n_cores      2000 non-null  int64  
 10  pc           2000 non-null  int64  
 11  px_height   2000 non-null  int64  
 12  px_width    2000 non-null  int64  
 13  ram          2000 non-null  int64  
 14  sc_h         2000 non-null  int64  
 15  sc_w         2000 non-null  int64  
 16  talk_time    2000 non-null  int64  
 17  three_g      2000 non-null  int64  
 18  touch_screen 2000 non-null  int64  
 19  wifi         2000 non-null  int64  
 20  price_range  2000 non-null  int64  
dtypes: float64(2), int64(19)
memory usage: 328.3 KB
```

# Limpieza de datos

Los datos fueron revisados y no se encontraron valores nulos.

test.csv

```
df_train.isna().sum()  
✓ 0.0s  
  
battery_power      0  
blue               0  
clock_speed        0  
dual_sim           0  
fc                 0  
four_g              0  
int_memory          0  
m_dep              0  
mobile_wt           0  
n_cores             0  
pc                 0  
px_height           0  
px_width            0  
ram                0  
sc_h                0  
sc_w                0  
talk_time            0  
three_g              0  
touch_screen         0  
wifi                0  
price_range          0  
dtype: int64
```

train.csv

```
df_test.isna().sum()  
✓ 0.0s  
  
id                  0  
battery_power        0  
blue                0  
clock_speed          0  
dual_sim             0  
fc                  0  
four_g               0  
int_memory            0  
m_dep                0  
mobile_wt             0  
n_cores               0  
pc                  0  
px_height             0  
px_width              0  
ram                  0  
sc_h                 0  
sc_w                 0  
talk_time              0  
three_g                0  
touch_screen           0  
wifi                  0  
dtype: int64
```

## Transformación de datos

Detectamos que el campo ID del data set test.csv, nos daría mucho ruido en los modelos que usaríamos durante el proyecto.

Se creo el data frame llamado df\_test\_proce y lo guardamos en la ruta de carpeta data/processed para ser usado.

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	s
0	1043	1	1.8	1	14	0	5	0.1	193	3	16	226	1412	3476	
1	841	1	0.5	1	4	1	61	0.8	191	5	12	746	857	3895	
2	1807	1	2.8	0	1	0	27	0.9	186	3	4	1270	1366	2396	
3	1546	0	0.5	1	18	1	25	0.5	96	8	20	295	1752	3893	
4	1434	0	1.4	0	11	1	49	0.5	108	6	18	749	810	1773	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	1700	1	1.9	0	0	1	54	0.5	170	7	17	644	913	2121	
996	609	0	1.8	1	0	0	13	0.9	186	4	2	1152	1632	1933	
997	1185	0	1.4	0	1	1	8	0.5	80	1	12	477	825	1223	
998	1533	1	0.5	1	0	0	50	0.4	171	2	12	38	832	2509	
999	1270	1	0.5	0	4	1	35	0.1	140	6	19	457	608	2828	

1000 rows × 20 columns

# Modelo y entrenamiento.

1

## Selección del modelo

Elegir el algoritmo de Machine Learning más adecuado para el problema.

2

## Definición de hiperparámetros

Configurar los parámetros del modelo para optimizar su rendimiento.

3

## Entrenamiento del modelo

Utilizar los datos preparados para entrenar el modelo y ajustar sus parámetros.



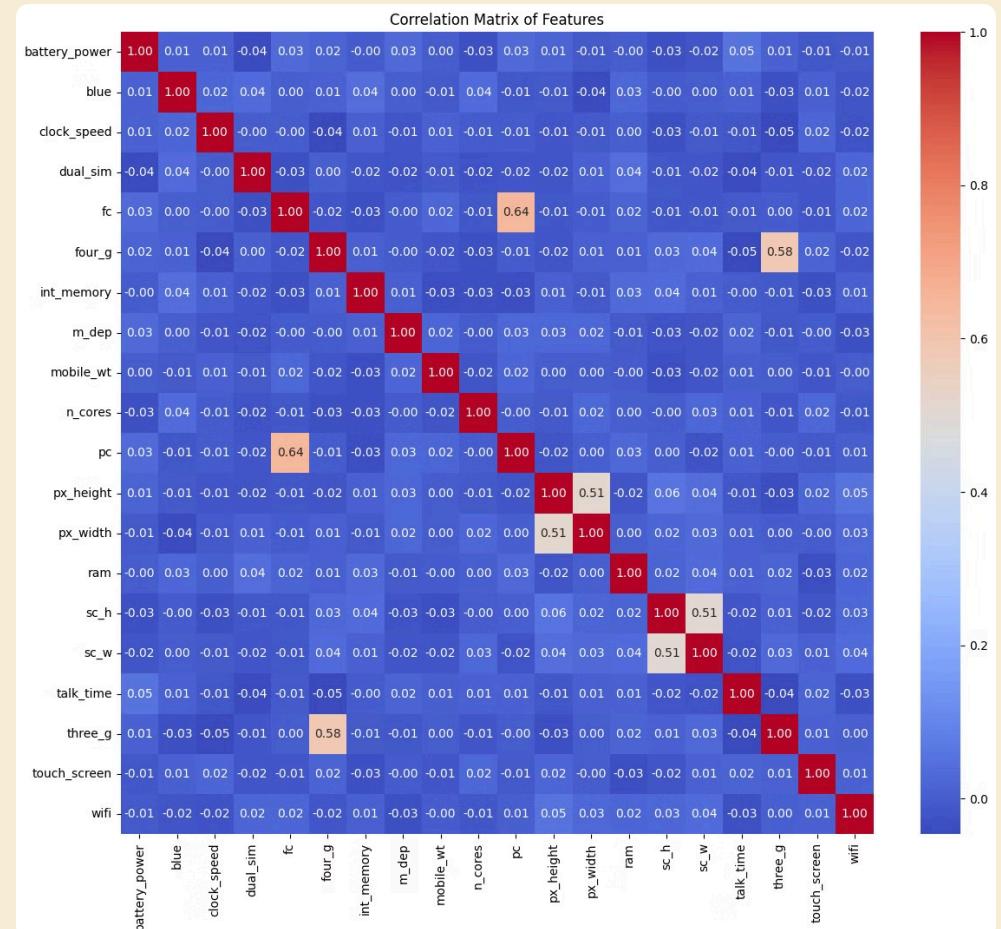
## Selección del modelo

El problema lo ubicamos como clasificación y bajo esta mirada usamos los siguiente modelos:

- a. Modelo supervisados:
  - i. Random Forest (Regularización + Cross-Validation)
  - ii. Support Vector Classifier (SVC) (Regularización + Cross-Validation)
  - iii. Gradient Boosting (Regularización + Cross-Validation)
  - iv. Logistic Regression (Regularización L2)
- b. Modelo no Supervisado
  - i. K-Means Clustering (Modelo No Supervisado)



Gráficos de dispersión



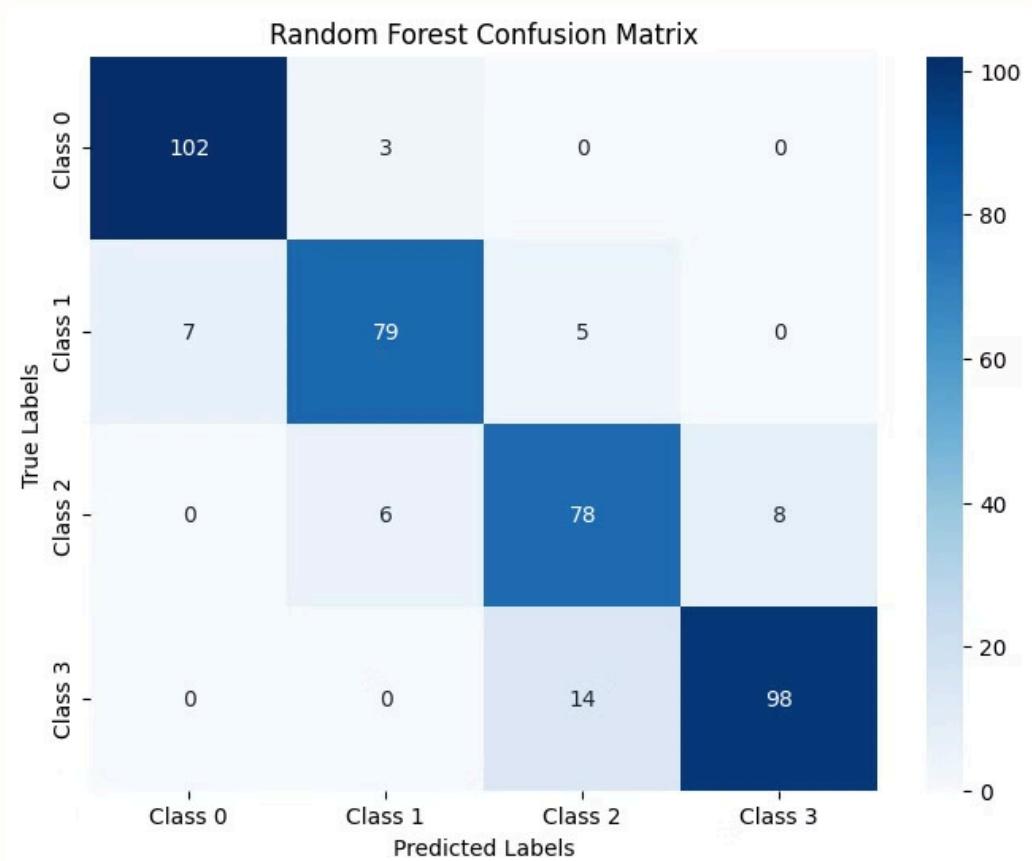
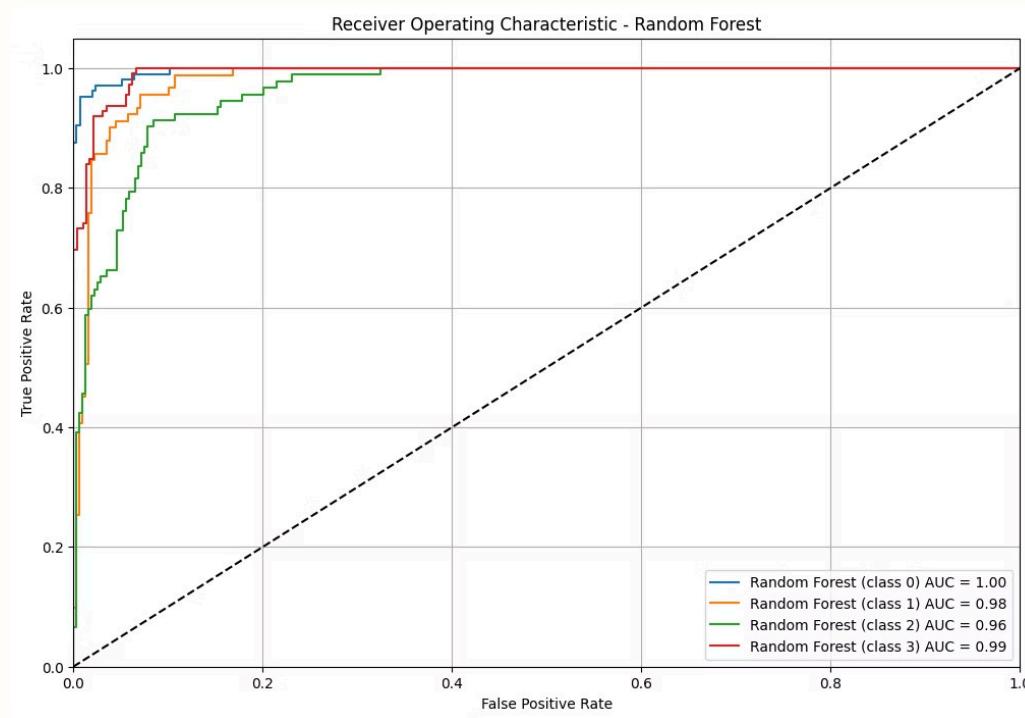
Gráfica de correlación de características

## Random Forest (Regularización + Cross-Validation)

```
Random Forest Classifier:  
a - Cross-Validation Scores: [0.853125 0.878125 0.86875 0.8375 0.85 ]  
b - Mean CV Score: 0.8574999999999999  
c - Accuracy on Validation: 0.8925  
d- Classification Report:  
precision recall f1-score support  
  
0 0.94 0.97 0.95 105  
1 0.90 0.87 0.88 91  
2 0.80 0.85 0.83 92  
3 0.92 0.88 0.90 112  
  
accuracy 0.89 0.89 0.89 400  
macro avg 0.89 0.89 0.89 400  
weighted avg 0.89 0.89 0.89 400
```

**Random Forest**

Accuracy	0.892500
Precision	0.890540
Recall	0.890597
F1-score	0.890108

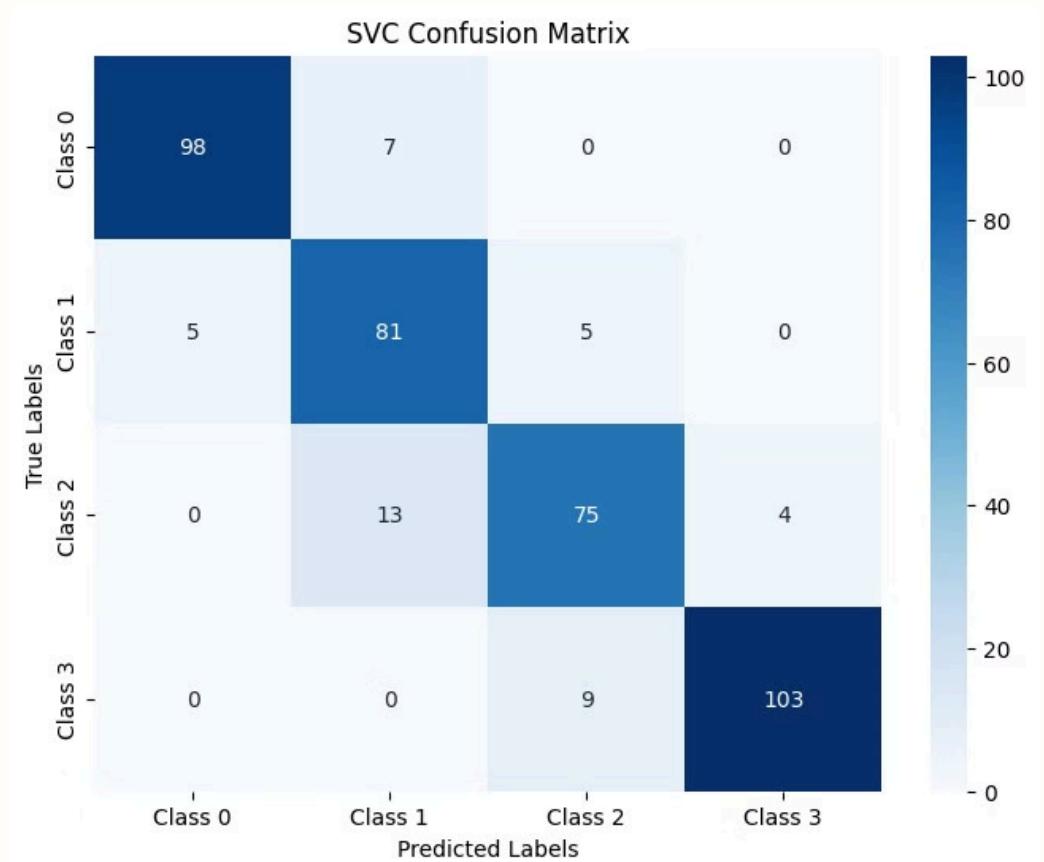
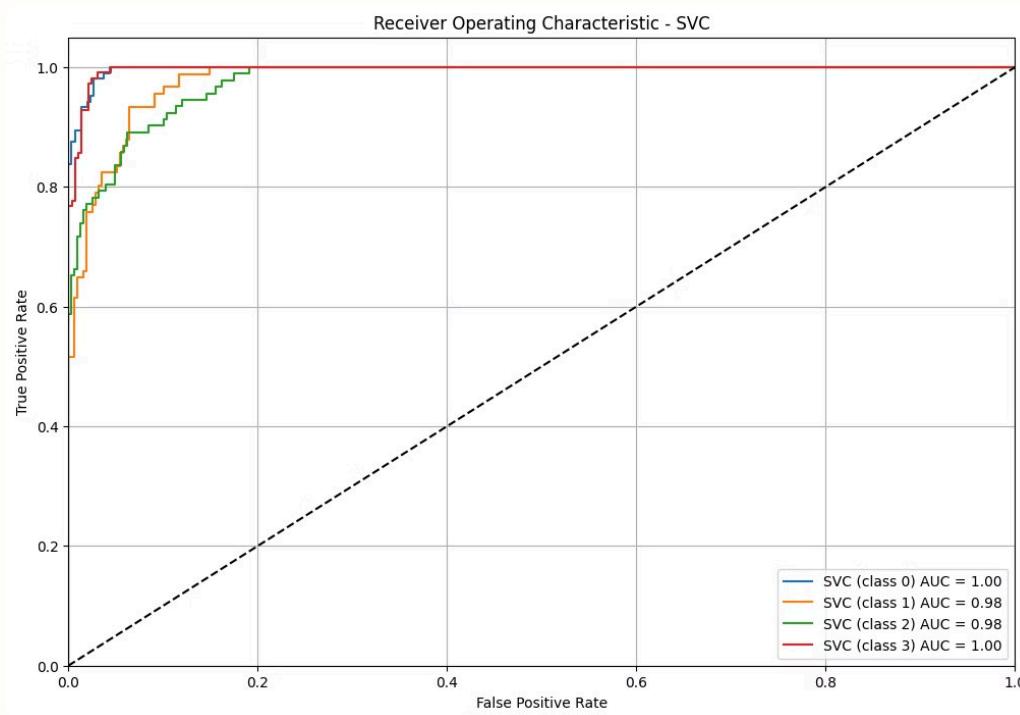


## Support Vector Classifier (SVC) (Regularización + Cross-Validation)

```
Support Vector Classifier (SVC):
a - Cross-validation Scores: [0.853125 0.89375 0.878125 0.846875 0.85      ]
b - Mean CV Score: 0.8643749999999999
c - Accuracy on Validation: 0.8925
d -Classification Report:
      precision    recall   f1-score  support
0       0.95     0.93     0.94     105
1       0.80     0.89     0.84      91
2       0.84     0.82     0.83      92
3       0.96     0.92     0.94     112
accuracy          0.89     0.89     0.89     400
macro avg       0.89     0.89     0.89     400
weighted avg    0.90     0.89     0.89     400
```

**SVC**

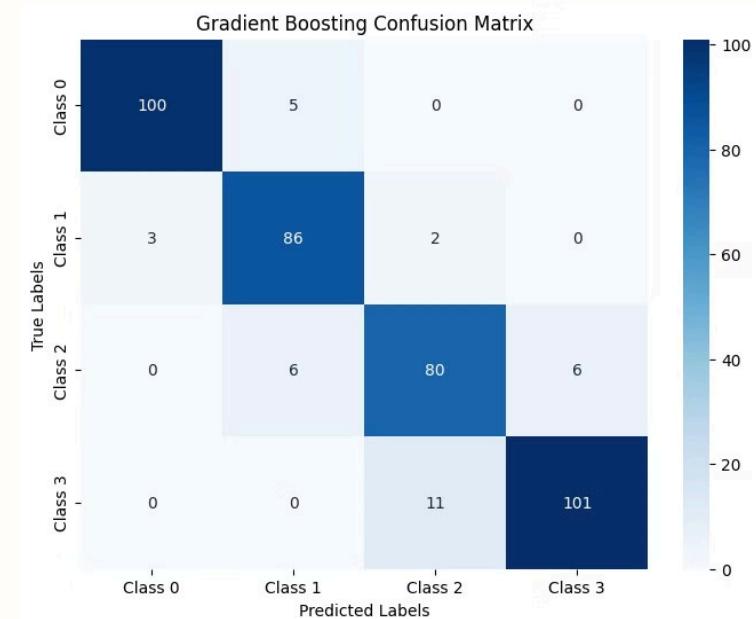
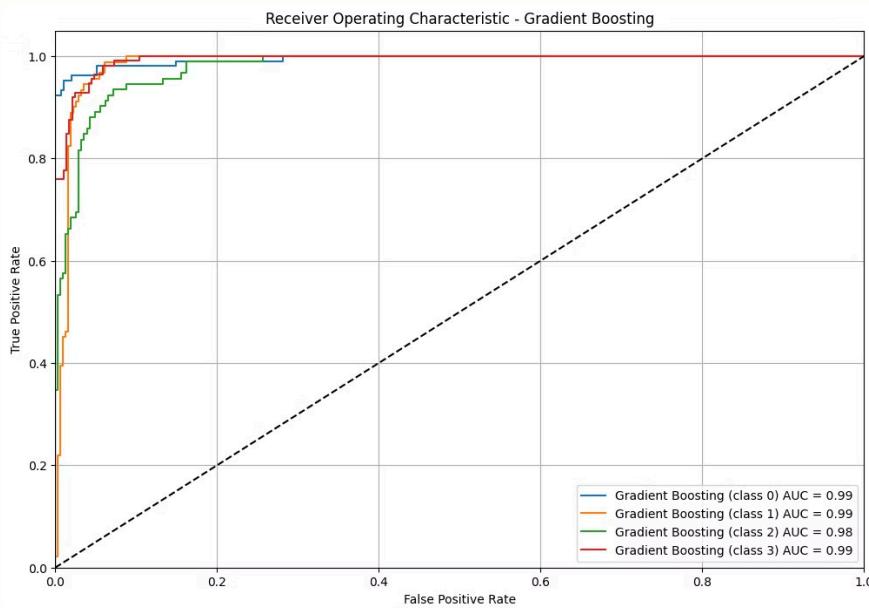
Accuracy	0.892500
Precision	0.889687
Recall	0.889576
F1-score	0.888857



## Gradient Boosting (Regularización + Cross-Validation)

```
Gradient Boosting Classifier:  
a - Cross-Validation Scores: [ 0.89375  0.9      0.896875  0.884375  0.903125]  
b - Mean CV Score: 0.8956250000000001  
c - Accuracy on Validation: 0.9175  
d - Classification Report:  
          precision    recall   f1-score  support  
  
          0       0.97     0.95     0.96     105  
          1       0.89     0.95     0.91      91  
          2       0.86     0.87     0.86      92  
          3       0.94     0.90     0.92     112  
  
accuracy                           0.92      400  
macro avg       0.92     0.92     0.92     400  
weighted avg    0.92     0.92     0.92     400
```

Gradient Boosting	
Accuracy	0.917500
Precision	0.915403
Recall	0.917197
F1-score	0.915918



## K-Means Clustering (Modelo No Supervisado)

```
KMeans Clustering (Validation Data):
```

col_0	0	1	2	3
price_range				
0	19	25	23	38
1	21	22	16	32
2	16	21	28	27
3	33	27	30	22

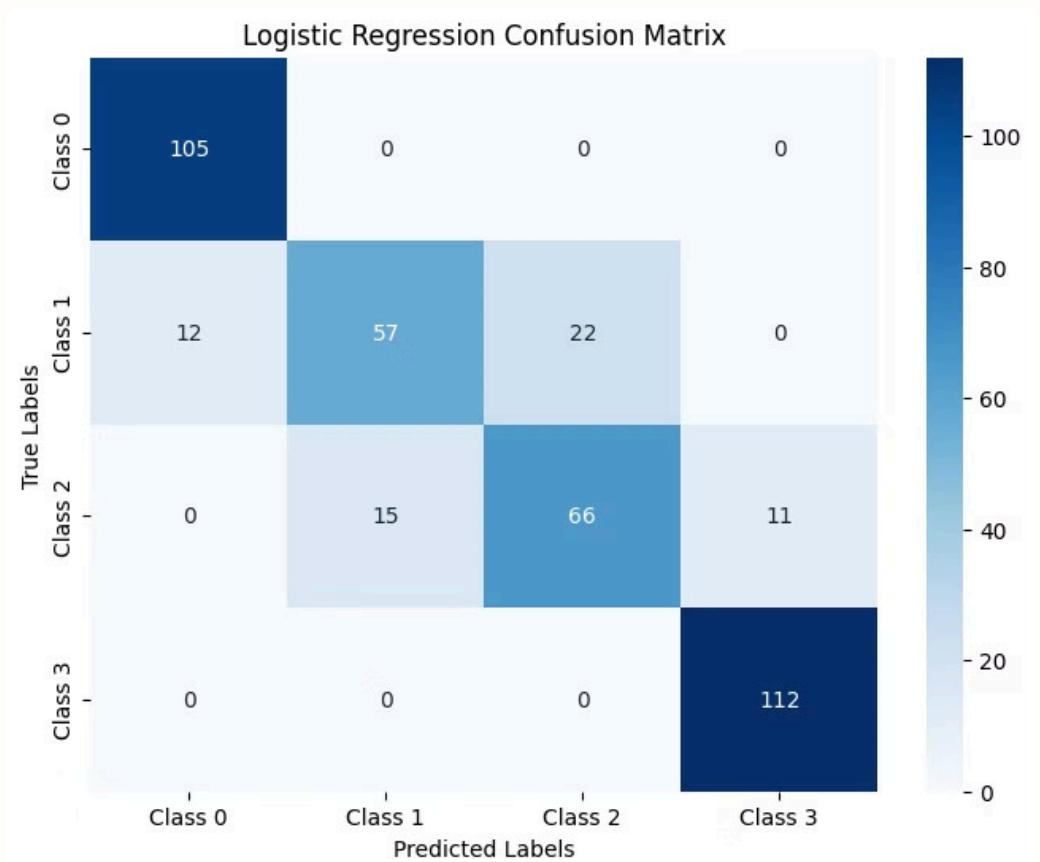
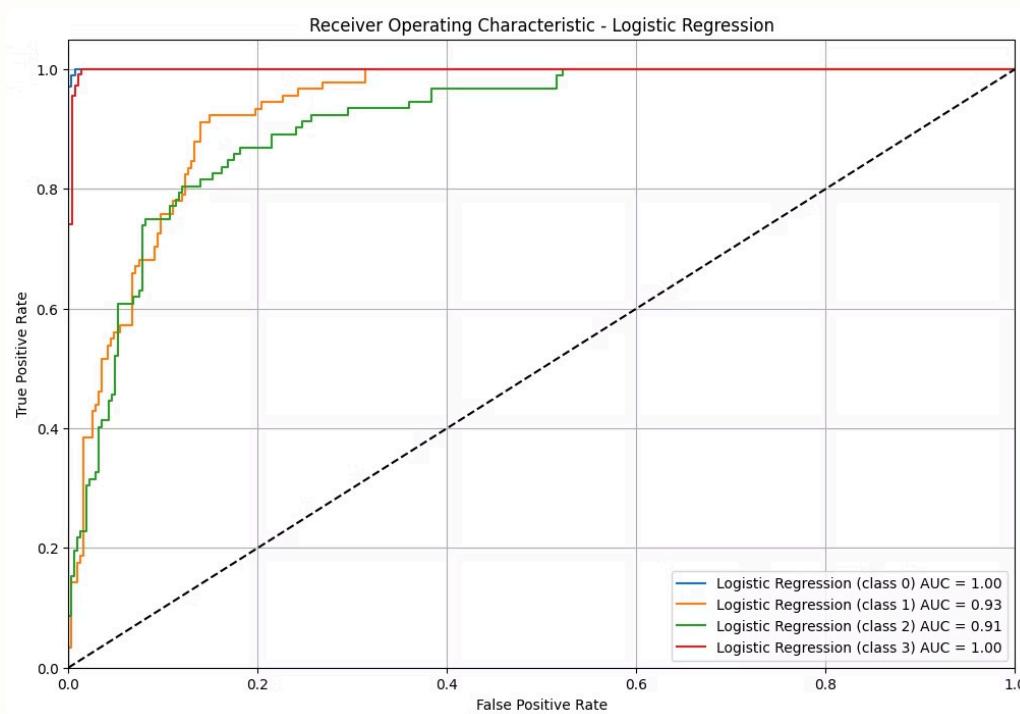
```
KMeans Clustering (Test Data):
```

col_0	0	1	2	3
row_0				
0	200	0	0	0
1	0	239	0	0
2	0	0	235	0
3	0	0	0	326

## Logistic Regression (Regularización L2)n)

```
Logistic Regression:  
a - Cross-Validation Scores: [0.8375  0.83125  0.825   0.80625  0.803125]  
b - Mean CV Score: 0.820625  
c - Accuracy on Validation: 0.85  
d - Classification Report:  
precision    recall  f1-score   support  
  
          0       0.90      1.00     0.95     105  
          1       0.79      0.63     0.70      91  
          2       0.75      0.72     0.73      92  
          3       0.91      1.00     0.95     112  
  
   accuracy                           0.85  
  macro avg       0.84      0.84     0.83     400  
weighted avg       0.84      0.85     0.84     400
```

Logistic Regression	
Accuracy	0.850000
Precision	0.837418
Recall	0.835941
F1-score	0.832964



# Evaluación y validación de resultados



# Evaluación y validación de resultados

## Cuadro de comparación de resultados

	<b>Random Forest</b>	<b>SVC</b>	<b>Gradient Boosting</b>	<b>Logistic Regression</b>
Accuracy	0.892500	0.892500	0.917500	0.850000
Precision	0.890540	0.889687	0.915403	0.837418
Recall	0.890597	0.889576	0.917197	0.835941
F1-score	0.890108	0.888857	0.915918	0.832964

### 1. Cross-Validation Scores

- **Scores:** [0.853125, 0.878125, 0.86875, 0.8375, 0.85]
- **Mean CV Score:** 0.8575

Los puntajes de cross-validation son bastante consistentes, con una media de 0.8575. Esto indica que el modelo es estable y tiene un buen desempeño en diferentes subconjuntos del conjunto de entrenamiento. Un valor por encima de 0.85 es generalmente considerado bueno.

### 2. Accuracy on Validation

- **Accuracy:** 0.8925

La precisión en el conjunto de validación es 0.8925, lo que sugiere que el modelo predice correctamente en casi el 89.25% de los casos. Este es un buen resultado, especialmente si se compara con otros modelos o con las expectativas del problema.

### 3. Classification Report

- **Precision:** Mide la exactitud de las predicciones positivas. Los valores son altos, especialmente para las clases 0 y 3.
- **Recall:** Mide la capacidad del modelo para capturar verdaderos positivos. Nuevamente, los valores son altos, con un recall algo menor en la clase 1.
- **F1-Score:** Es una media armónica de la precisión y el recall. La mayoría de los valores están alrededor de 0.89 o más, lo que indica un buen equilibrio entre precisión y recall.

El modelo parece funcionar mejor en las clases 0 y 3, mientras que las clases 1 y 2 tienen un desempeño ligeramente inferior. Sin embargo, la variación no es drástica.

### 4. ROC Curve

- Los valores de la curva ROC sugieren que el modelo tiene una buena capacidad para distinguir entre las clases, con una Tasa de Falsos Positivos (FPR) baja y una Tasa de Verdaderos Positivos (TPR) alta.

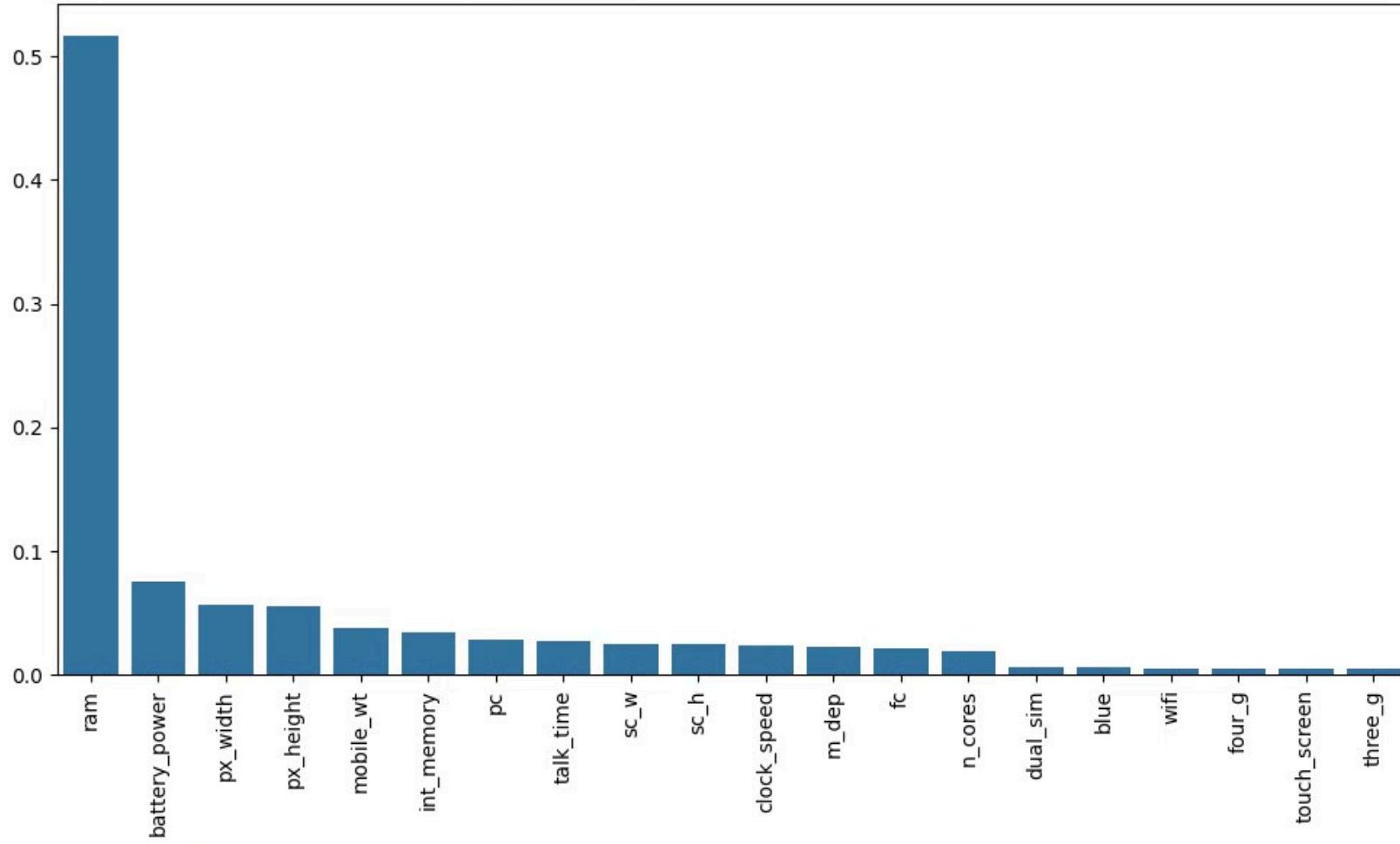
- **Viabilidad:** Sí, el modelo parece ser viable. Los puntajes de validación cruzada son consistentes y altos, la precisión y el recall son buenos, y el modelo tiene un buen rendimiento en la curva ROC. Aunque hay algunas áreas donde el modelo podría mejorar, como en la precisión de las clases 1 y 2, en general, se trata de un modelo robusto y efectivo.

### 5. Confusion Matrix

- La matriz de confusión muestra que el modelo tiene una buena capacidad de predicción para la mayoría de las clases, especialmente para las clases 0 y 3. Hay algunos errores de clasificación en las clases 1 y 2, pero no parecen ser demasiado significativos.

# Interpretación de Variables

Feature Importances - Random Forest



# Obstáculos y próximos pasos



## Obstáculos

- falta de conocimiento de la implementación de los modelos
- Realizar los análisis a cada resultado
- Falta de conocimiento de Python y los paquetes para ML
- No supe crear el archivo final con el resultado solicitado que era tener el id y price\_range, es la variable Target y usaremos como valor 0: (Costo bajo), 1: (Costo medio), 2: (Costo alto) y 3: (Costo muy alto)



## Próximos pasos

- Cerrar esa brechas de conocimiento.



## Reto

Mejorar el rendimiento del modelo.

