M2 Evidence 3 - Performance

Karla Stefania Cruz Muñiz A01661547

Selected algorithm: Random Forest

Random Forest is an algorithm for machine learning whose core are decision trees. This algorithm combines the using of multiple decision trees for getting the different results of all of them to end with a better prediction. In cases of classification problems, this algorithm takes the majority vote to give a final result about the classification of some prediction.

Selected dataset

For this part of the evidence I choose a dataset for finding a mobile price, it looks like this:

battery_pow	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_scree	wifi	price_range
842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	19	0	0	1	1
1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7	1	1	0	2
563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9	1	1	0	2
615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11	1	0	0	2
1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15	1	1	0	1
1859	0	0.5	1	3	0	22	0.7	164	1	7	1004	1654	1067	17	1	10	1	0	0	1
1821	0	1.7	0	4	1	10	0.8	139	8	10	381	1018	3220	13	8	18	1	0	1	3
1954	0	0.5	1	0	0	24	0.8	187	4	0	512	1149	700	16	3	5	1	1	1	0
1445	1	0.5	0	0	0	53	0.7	174	7	14	386	836	1099	17	1	20	1	0	0	0
509	1	0.6	1	2	1	9	0.1	93	5	15	1137	1224	513	19	10	12	1	0	0	0
769	1	2.9	1	0	0	9	0.1	182	5	1	248	874	3946	5	2	7	0	0	0	3
1520	1	2.2	0	5	1	33	0.5	177	8	18	151	1005	3826	14	9	13	1	1	1	3
1815	0	2.8	0	2	0	33	0.6	159	4	17	607	748	1482	18	0	2	1	0	0	1
803	1	2.1	0	7	0	17	1	198	4	11	344	1440	2680	7	1	4	1	0	1	2
1866	0	0.5	0	13	1	52	0.7	185	1	17	356	563	373	14	9	3	1	0	1	0
775	0	1	0	3	0	46	0.7	159	2	16	862	1864	568	17	15	11	1	1	1	0
838	0	0.5	0	1	1	13	0.1	196	8	4	984	1850	3554	10	9	19	1	0	1	3
595	0	0.9	1	7	1	23	0.1	121	3	17	441	810	3752	10	2	18	1	1	0	3
1131	1	0.5	1	11	0	49	0.6	101	5	18	658	878	1835	19	13	16	1	1	0	1
682	1	0.5	0	4	0	19	1	121	4	11	902	1064	2337	11	1	18	0	1	1	1
772	0	1.1	1	12	0	39	0.8	81	7	14	1314	1854	2819	17	15	3	1	1	0	3
1709	1	2.1	0	1	0	13	1	156	2	2	974	1385	3283	17	1	15	1	0	0	3
1949	0	2.6	1	4	0	47	0.3	199	4	7	407	822	1433	11	5	20	0	0	1	1
1602	1	2.8	1	4	1	38	0.7	114	3	20	466	788	1037	8	7	20	1	0	0	0

As we can see, the dataset contains a lot of relevant information for classifying a mobile phone into a prince range indicating how high the price is. There are 4 possible ranges, 0, 1, 2 and 3.

This dataset was selected because we know Random Forest can be used either for classification or regression, and this dataset is great for classification according to the target data, that is a range of four possible discrete values. Also, we can talk about the type of values in the dataset, numerical, and Random Forest can well manage characteristics numerical and binary.
Lastly defining how useful this dataset is for generalizing, we can conclude that as we have many features that can be correlated and the algorithm takes different sets of these features, this can avoid depending only on one feature.
Also, the width of the dataset works great for Random Forest, considering this kind of algorithm needs a great quantity of examples for training and for validation.

You can find the dataset in the next kaggle link:
https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification/data
I used the training part because it's where the results are shown and I can have supervised learning. You can find this csv as *'train.csv'*.

Performance of the algorithm with this dataset

1. **Separation and evaluation of the model with a test set and a validation set (Train/Test/Validation)**

This dataset was divided in form 80%-20%. When dividing this dataset I selected the 20% for validation because it is a commonly used value in practice ensuring the model is trained with enough data while validating with a representative set.

As the division works with a fixed random seed, guaranteeing every time we run the code, the data shuffling and splitting process is done the same, obtaining the same sets, the next images show the .head() of each of this sets:

```
First rows of the training set (X_train):
     battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
968           1923     0          0.5         1   7       0          46
240            633     1          2.2         0   0       1          49
819           1236     0          0.9         1   2       1          57
692            781     0          1.1         0   2       0          38
420           1456     1          0.5         1   7       0           7

     m_dep  mobile_wt  n_cores  pc  px_height  px_width   ram  sc_h  sc_w  \
968    0.5        191        1  10        767      1759  1489    10     9
240    0.1        139        8   1        529      1009  3560    11     1
819    0.1        188        1  14        517       809  1406    14    12
692    0.4        198        5   7        304      1674  3508    13     8
420    0.4        105        5  12        823      1104  1587     6     5

     talk_time  three_g  touch_screen  wifi
968          3        1             1     1
240         16        1             1     1
819         20        1             0     1
692          5        0             0     1
420         20        1             0     1
```

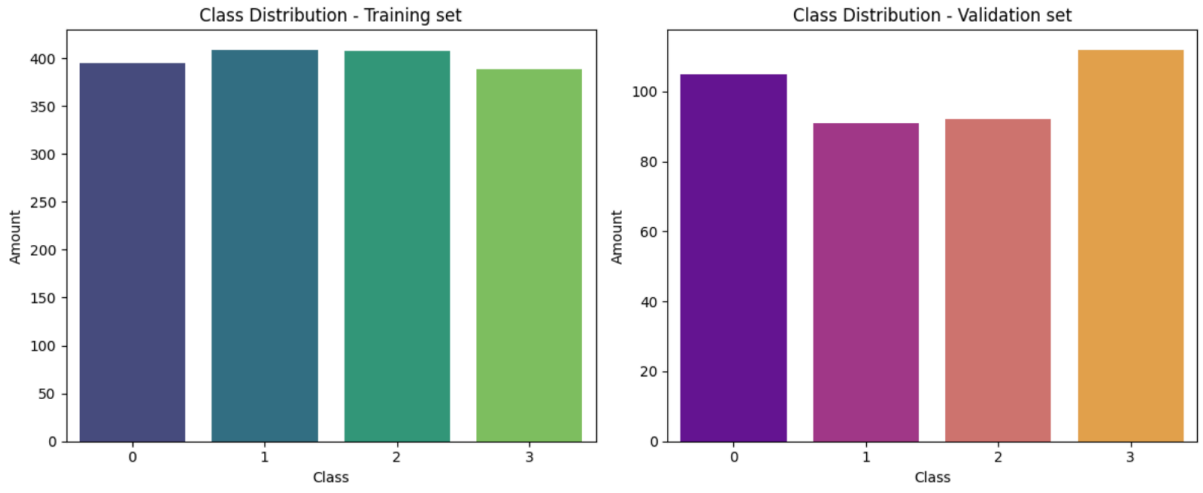
```
First rows of the validation set (X_val):
     battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
1860           1646     0          2.5         0   3       1          25
353            1182     0          0.5         0   7       1           8
1333           1972     0          2.9         0   9       0          14
905             989     1          2.0         0   4       0          17
1289            615     1          0.5         1   7       0          58

     m_dep  mobile_wt  n_cores  pc  px_height  px_width   ram  sc_h  sc_w  \
1860   0.6        200        2   5        211      1608   686     8     6
353    0.5        138        8  16        275       986  2563    19    17
1333   0.4        196        7  18        293       952  1316     8     1
905    0.2        166        3  19        256      1394  3892    18     7
1289   0.5        130        5   8       1021      1958  1906    14     5

     talk_time  three_g  touch_screen  wifi
1860         11        1             1     0
353          19        1             0     0
1333          8        1             1     0
905          19        1             1     0
1289          5        1             0     0
```

Then, I used the function value_counts() for counting the occurrences of each class in y_train, and also I use normalize=True to return the proportion of each class in the relation to the total number of examples, resulting like this:

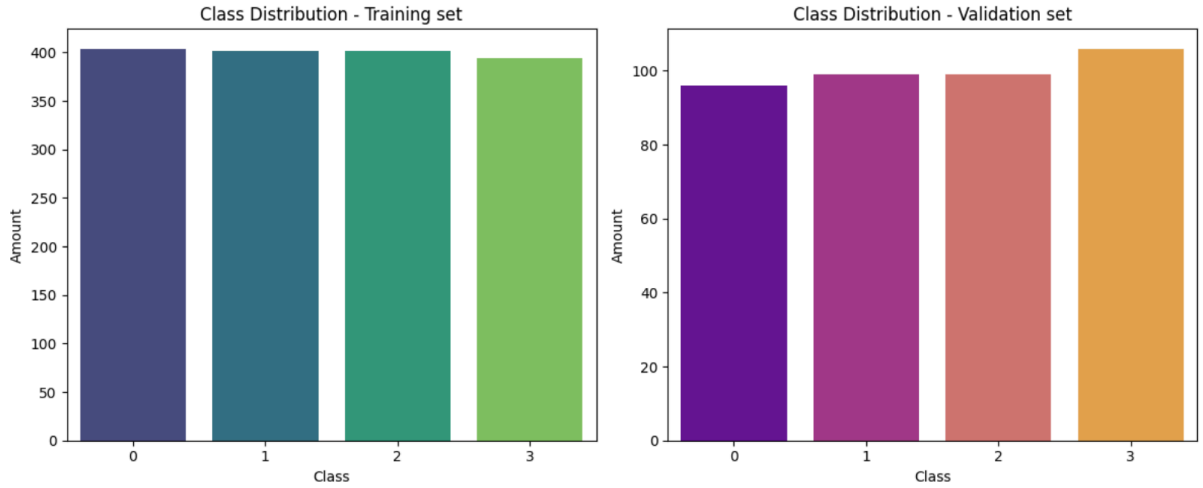
```
Distribution of classes in the training set:
price_range
1    0.255625
2    0.255000
0    0.246875
3    0.242500
Name: proportion, dtype: float64

Distribution of classes in the validation set:
price_range
3    0.2800
0    0.2625
2    0.2300
1    0.2275
Name: proportion, dtype: float64
```

So, explaining this output, we can see that for the training set the data of each class is evenly distributed among the different price ranges in the training set but for the validation set there are some classes with significative differences, like between the class 3 and 1, so the proportions are fairly balanced, this is due the random splitting but we can be sure every class is represented in the division.

Lastly for this point, I made a plot so I can visualize the distribution between each class of each set.



Even if the data of the validation set is not perfectly balanced, the random seed with number 42 was the one to give better accuracy. I tried with 41 and the distribution seemed balanced but the accuracy went down. Example in the same image:



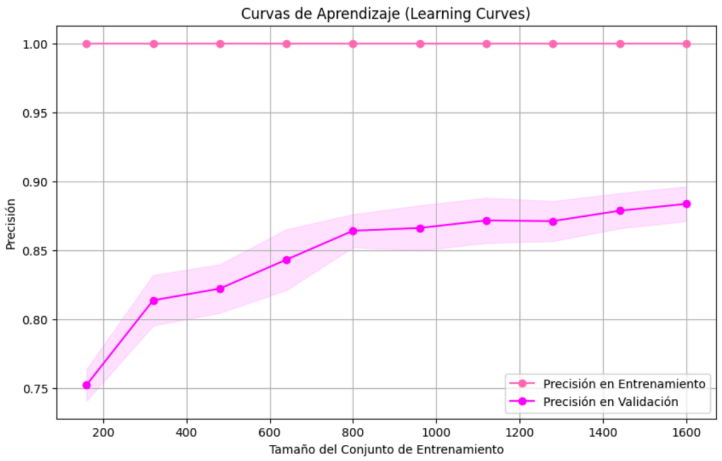

```
Accuracy in validation data: 0.8475
Accuracy in training data: 0.965
```

2. **Diagnosis and explanation of the degree of bias: low medium high**

```
Accuracy in validation data: 0.8925
Accuracy in training data: 1.0
```

```
Average accuracy with cross-validation: 0.89150673887516
```

The previous images are a clear sample of Overfitting. In this part we can check the Learning curves but based on just the values obtained with the accuracy we can distinct an Overfitting.



Now with the learning curves plot the overfitting is more than obvious because of the constant 1 precision of the training curve.

In conclusion, we have a low bias in the validation data, nos as low as an Overfitting some times should show but maybe with more data the graph would evidentiate a little bit more the overfitting.

3. **Diagnosis and explanation of the degree of variance: low medium high**

About the variance, we can look at the differences between the accuracy of the model obtained without cross validation and the one that actually has cross validation. The next images show both accuracies.

```
Accuracy in validation data: 0.8925
Accuracy in training data: 1.0
```

```
Average accuracy with cross-validation: 0.89150673887516
```

As the difference between both accuracies is not that high, we can say the variance is low too. This should mean the model is generalizing not seen data.

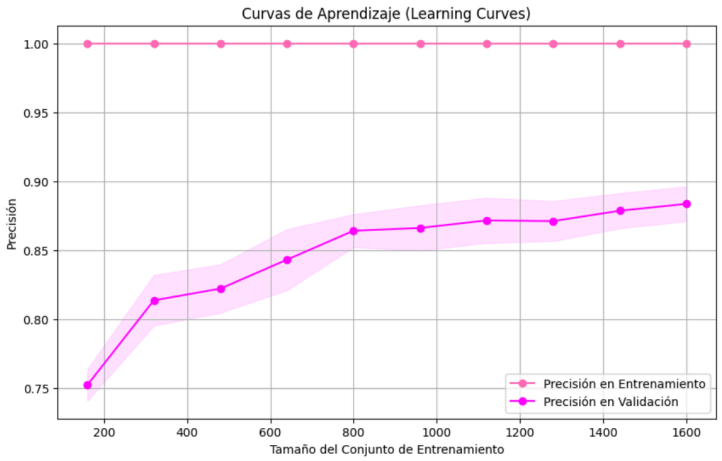
4. **Diagnosis and explanation of the model fit level: underfit fit overfit**

At this point, overfitting is not so evident, because when the model performs well in both training and validation, it is an indication of good fit, but there's a thing that changes everything and is the accuracy of 1 for training. At this point and analyzing all data obtained, I conclude the model is overfitting the training data but still generalizing well to unseen data.

But how is this possible?
As mentioned before, Random Forest is an ensemble of decision trees, and if the trees are allowed to grow too deep, they can learn to perfectly classify the training data; but, despite overfitting on the training data, the random selection of features for each tree and the averaging of multiple trees' predictions reduce variance and help avoid overfitting to the point of poor test performance, letting the model to generalize well.
While each tree might overfit, the combined predictions of many trees can still generalize well on unseen data.

5. **Regularization or parameter adjustment techniques to improve the performance of the model**

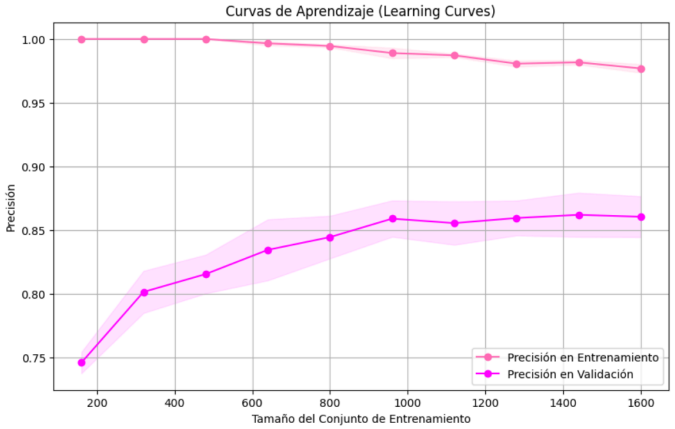
For this part I previously made an adjustment for hyperparameters, but without making any comparison between training and testing, so today I realized my code was overfitting thanks to the graph we previously saw.



In this graph we can see that the training curve is always one, showing the overfitting. The validation one is growing but we never see the training one is moving.

Based on this graph I made changes about the maximum depth the trees can reach. The value applied when generating this graph for *max_depth* was 17, and the logic between that parameter was the fact the accuracy in the validation test was higher with that depth, but I wasn't considering the accuracy with training. Lowering the values for *max_depth* allows me to get the next learning curves.

For this image now we can see how the training curve is getting down which means the model is finding it more difficult to “memorize” the answers when more data is provided, normalizing its functionality and reducing the overfitting.



It was a great balance between the accuracy of the training and the accuracy of validation, as shown here in this image.

```
Accuracy in validation data: 0.8825
Accuracy in training data: 0.97875
```

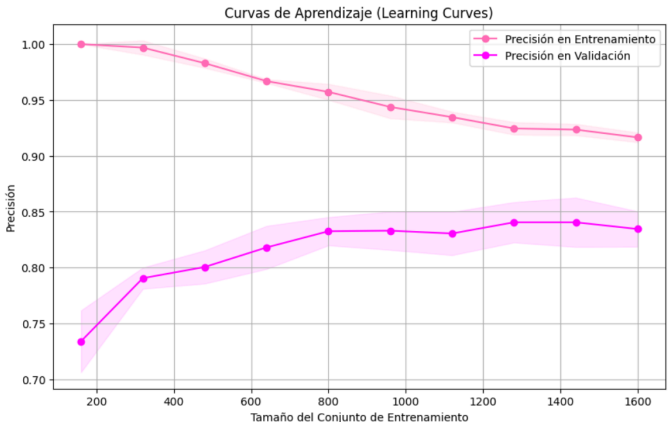
Previously, when having an overfit, the values I obtained were the ones of this second image.

```
Accuracy in validation data: 0.8925
Accuracy in training data: 1.0
```

Showing the difference between the accuracy with a big depth and a lower depth is not as significant for the validation data and causes an overfitting.
Also, I tested with a max depth of 5, causing the accuracy in validation data lows “a lot”, so that’s why the final selected depth was 7.

```
Accuracy in validation data: 0.86
Accuracy in training data: 0.91125
```

This was the “middle” test and the results were not as favorable as the test with 7 and the test with 6 was more or less the same so that’s why I took the decision of using a 7.



After this first analysis of learning curves and adjustment of the max depth value we can proceed with adjustment for other hyperparameters and regularization. First of all I did an adjustment of estimators, the param that controls the number of trees, and in this case the first value tested was 50, after 100, 200, 300 and 400 giving the next results:

n_estimators = 50

```
Accuracy in validation data: 0.865
Accuracy in training data: 0.973125
```

```
Average accuracy with cross-validation: 0.843
```

n_estimators = 100

```
Accuracy in validation data: 0.885
Accuracy in training data: 0.976875
```

```
Average accuracy with cross-validation: 0.8545
```

n_estimators = 200

```
Accuracy in validation data: 0.88
Accuracy in training data: 0.978125
```

```
Average accuracy with cross-validation: 0.864
```

n_estimators = 300

```
Accuracy in validation data: 0.8825
Accuracy in training data: 0.97875
```

```
Average accuracy with cross-validation: 0.867
```

n_estimators = 400

```
Accuracy in validation data: 0.8725
Accuracy in training data: 0.97625
```

```
Average accuracy with cross-validation: 0.867
```

Based on the analysis and runs of the code with these different values of estimators, I conclude the best form is gonna be 300 because it is the best accuracy run with only 300 trees and the one of the 400 trees is returning the same result but with more runs, more resources and is not worth it.

Also, another thing to do is considering using the *min_samples_leaf* hyperparameter, because this parameter helps avoiding overfitting and helps the model in its ability to generalize. When this parameter is not set to a very small number, avoids overfitting by not letting the model to adjust perfectly to the training data.
So the values we are testing for this hyperparameter are: 3, 4, 5 and 7.

min_samples_leaf = 2

```
Accuracy in validation data: 0.885
Accuracy in training data: 0.973125
```

```
Average accuracy with cross-validation: 0.8634999999999999
```

min_samples_leaf = 3

```
Accuracy in validation data: 0.885
Accuracy in training data: 0.969375
```

```
Average accuracy with cross-validation: 0.8645
```

min_samples_leaf = 4

```
Accuracy in validation data: 0.8775
Accuracy in training data: 0.961875
```

```
Average accuracy with cross-validation: 0.8665
```

min_samples_leaf = 5

```
Accuracy in validation data: 0.8825
Accuracy in training data: 0.95875
```

```
Average accuracy with cross-validation: 0.8634999999999999
```

min_samples_leaf = 7

```
Accuracy in validation data: 0.8825
Accuracy in training data: 0.95375
```

```
Average accuracy with cross-validation: 0.8625
```

So the final min_sample_leaf number selected was 7.

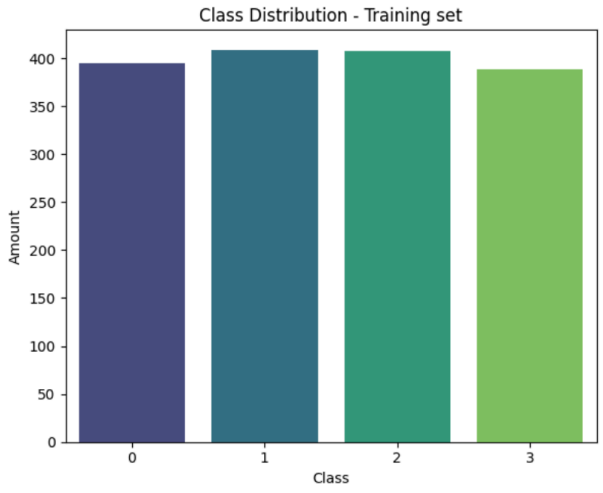
Also, and just to finish with this part, I had to adjust the number of folds at the cross validation because the performance was getting lower with the new modifications and the old value of 7 folds.

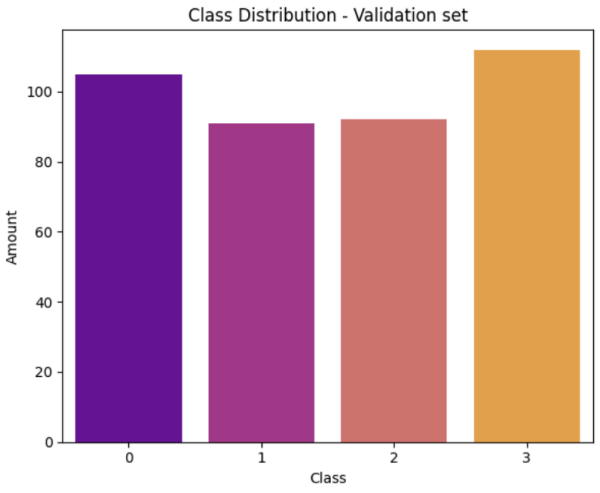
This is the final output for this final model.

```
First rows of the training set (X_train):
     battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
968           1923     0          0.5         1   7       0          46
240            633     1          2.2         0   0       1          49
819           1236     0          0.9         1   2       1          57
692            781     0          1.1         0   2       0          38
420           1456     1          0.5         1   7       0           7

     m_dep  mobile_wt  n_cores  pc  px_height  px_width   ram  sc_h  sc_w  \
968    0.5        191        1  10        767      1759  1489    10     9
240    0.1        139        8   1        529      1009  3560    11     1
819    0.1        188        1  14        517       809  1406    14    12
692    0.4        198        5   7        304      1674  3508    13     8
420    0.4        105        5  12        823      1104  1587     6     5

     talk_time  three_g  touch_screen  wifi
968          3        1             1     1
240         16        1             1     1
819         20        1             0     1
692          5        0             0     1
420         20        1             0     1
```

```
First rows of the validation set (X_val):
      battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
1860           1646     0          2.5         0   3       1          25
353            1182     0          0.5         0   7       1           8
1333           1972     0          2.9         0   9       0          14
905             989     1          2.0         0   4       0          17
1289            615     1          0.5         1   7       0          58

      m_dep  mobile_wt  n_cores  pc  px_height  px_width   ram  sc_h  sc_w  \
1860    0.6        200        2   5        211      1608   686     8     6
353     0.5        138        8  16        275       986  2563    19    17
1333    0.4        196        7  18        293       952  1316     8     1
905     0.2        166        3  19        256      1394  3892    18     7
1289    0.5        130        5   8       1021      1958  1906    14     5

      talk_time  three_g  touch_screen  wifi
1860         11        1             1     0
353          19        1             0     0
1333          8        1             1     0
905          19        1             1     0
1289          5        1             0     0
```

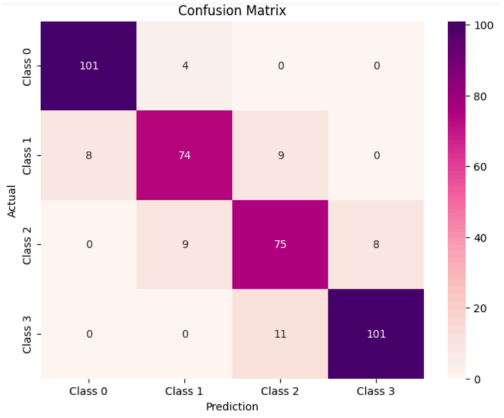
```
Distribution of classes in the training set:
price_range
1    0.255625
2    0.255000
0    0.246875
3    0.242500
Name: proportion, dtype: float64

Distribution of classes in the validation set:
price_range
3    0.2800
0    0.2625
2    0.2300
1    0.2275
Name: proportion, dtype: float64
```

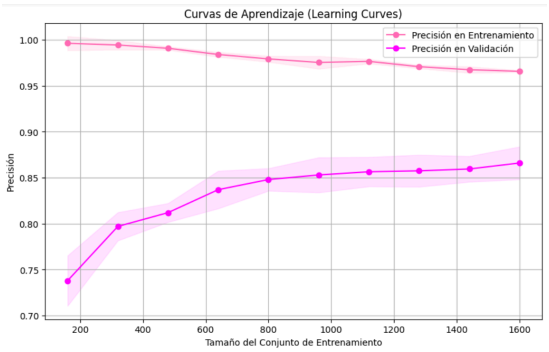





```
Accuracy in validation data: 0.8775
Accuracy in training data: 0.961875
```

```
Confusion Matrix:
[[101   4   0   0]
 [  8  74   9   0]
 [  0   9  75   8]
 [  0   0  11 101]]
```






```
Diagnosing the degree of bias and variance:
The model is well-fitted (good fit).
```

```
Average accuracy with cross-validation: 0.8665
```

```
Feature importance:
- ram: 0.6322150358034059
- battery_power: 0.06955435061582804
- px_width: 0.04758139098466046
- px_height: 0.046860578298662844
- mobile_wt: 0.027167756661371372
- int_memory: 0.02631745903842021
- talk_time: 0.018381420631368026
- pc: 0.01820026614583776
- sc_w: 0.01704406913027349
- sc_h: 0.016488502345250497
- clock_speed: 0.016287623656005563
- fc: 0.01581838160966329
- m_dep: 0.014411948759367275
- n_cores: 0.013538477333053725
- blue: 0.003928974852119108
- dual_sim: 0.0037563903590419278
- four_g: 0.003298682063001821
- touch_screen: 0.0032983468675262355
- wifi: 0.003139115341987562
- three_g: 0.0027112295031549724
```