Diamond's cut Multiclass Classification

Gonçalo Machado
Department of
Electronics, Telecommunications
and Informatics
University of Aveiro
Aveiro, Portugal
goncalofmachado@ua.pt

Vicente Costa
Department of
Electronics, Telecommunications
and Informatics
University of Aveiro
Aveiro, Portugal
vicente.costa@ua.pt

***Abstract*—Classifying the cut quality of a diamond is a problem to many jewelry shops. Having a bad evaluation can not only provoke lost of money but also the owner can be sued for fraud. So, In this work, we propose a machine learning approach to classify the diamond cut by its dimensions, price, clarity, depth and others with a dataset containing almost 54,000 diamonds.**

***Index Terms*—Multi Class Classification, HyperParameter Tuning, Machine Learning, AI**

I. Introduction

This document is a solution of a multiclass classification problem from a dataset of diamonds. With this report we may help jewelry stores to classify the quality cut of a diamond depending of their characteristics. In the following sections, we elaborate the development process to reach the mentioned goal. Section II we will briefly speak of a related work that we were based on. In Section III, it presents where we have retrieved the data to train and how the dataset is composed among which who are the features and how they relate with each other. In section IV, we go through our approach to accomplish our goal. And finally, in Section V, we present our conclusions and accomplishments from the development and analysis of the obtained results.

II. Related Work

In the article Comprehensive Guide to Multiclass Classification With Sklearn [1] the author use the same dataset to solve a multiclass classification problem. We based our report in this article to solve our problem.

III. DataSet

The data used in this paper was found on Kaggle, an online community of data scientists and machine learning practitioners where users can find and publish data sets, as well as build models or enter competitions to solve data science challenges. The data is available at https://www.kaggle.com/datasets/shivam2503/diamonds and is composed of 9 features and the label cut.

A. Label Cut

The only label in this problem is the diamond cut quality which can be in in increasing order of quality fair, Good, Very Good, Premium or Ideal.

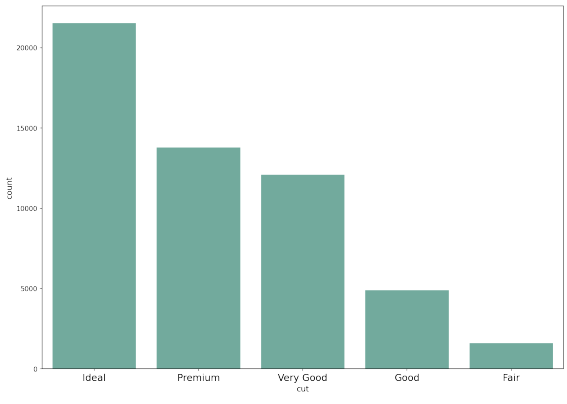

Figure 1. Histogram of the Cut Countdown for each quality of cut

As we can see, we also have and increasing order of samples for each quality.

B. feature carat

The first feature is carat that represents the weight of the diamond.

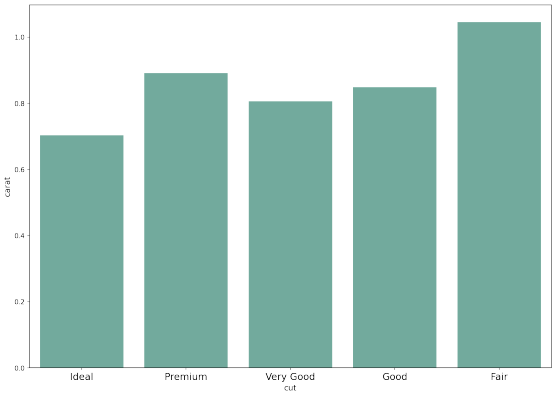

Figure 2. Histogram of the average carat per cut

In the histogram above a diamond with the best cut has a lower weight and the worst cut the highest weight.

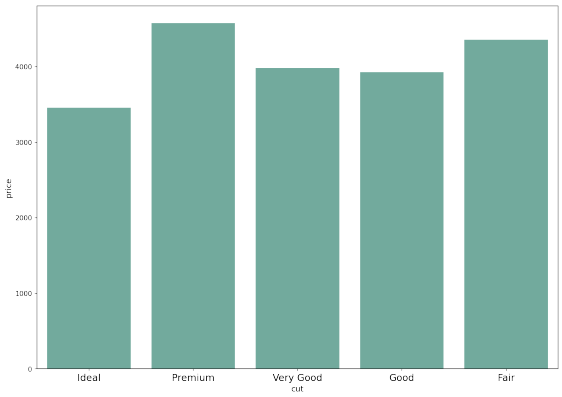

Figure 3. Histogram of the average price per cut

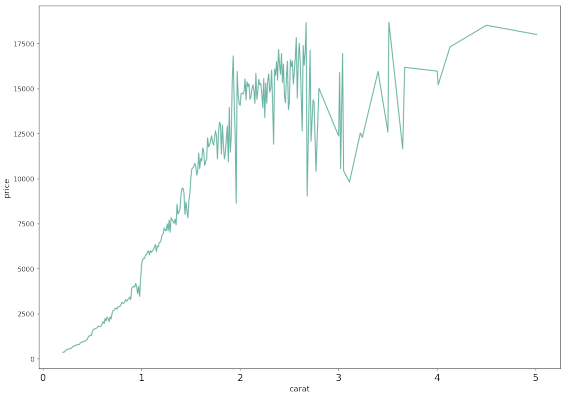

Figure 4. Histogram of the average price per carat

C. feature price

In the figure 3, in this dataset, a diamond with and ideal cut has a lower price than the others, being the premium cut with the highest followed by the fair cut, which is odd, normally we would think the better the cut the high the monetary value. To explain this fact, i suggest to pay attention at the figure 4 and 2. With this two figures we can conclude that price of the diamonds increase with their weight and the diamonds with the quality cut Fair and Premium have the heaviest diamonds and the ideal the lighter, so naturally the diamonds with the ideal cut has a lower price while the premium and fair have a higher price.

D. feature color

The next feature is the color of the diamond where the value could range between ”D” and ”J” being ”D” the best and ”J” the worst.

The average colour for ideal, premium and fair cuts is G followed by the colour E for the good and very good cuts.

The price of the diamond per color lower with the best color due to the weight also lowering however, in the D color has a slightly jump due having the same weight as the E color.

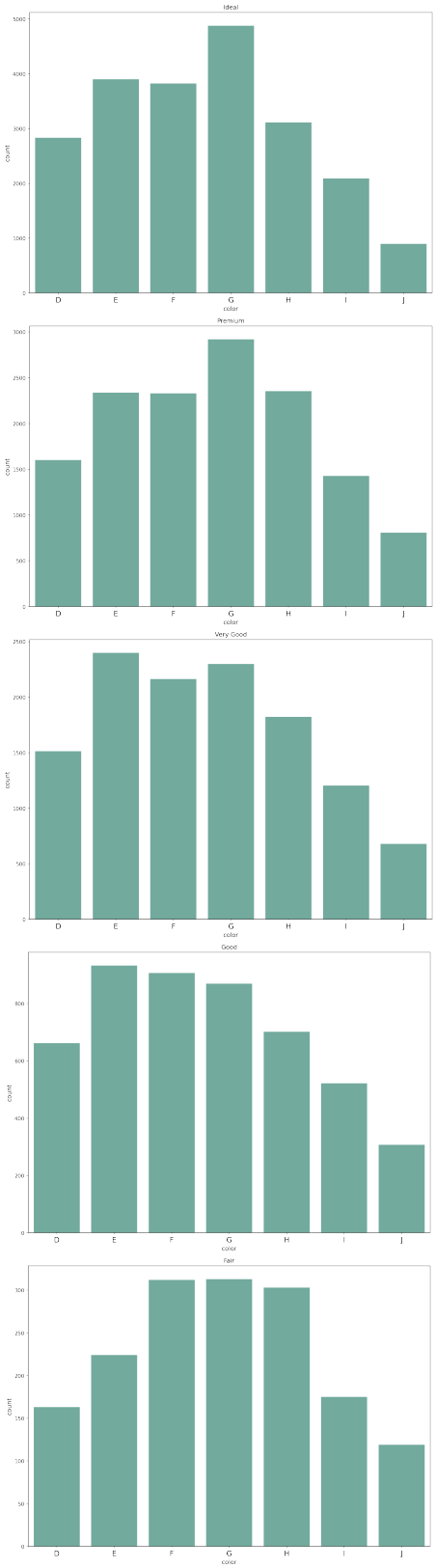

Figure 5. Histogram of the color per cut quality

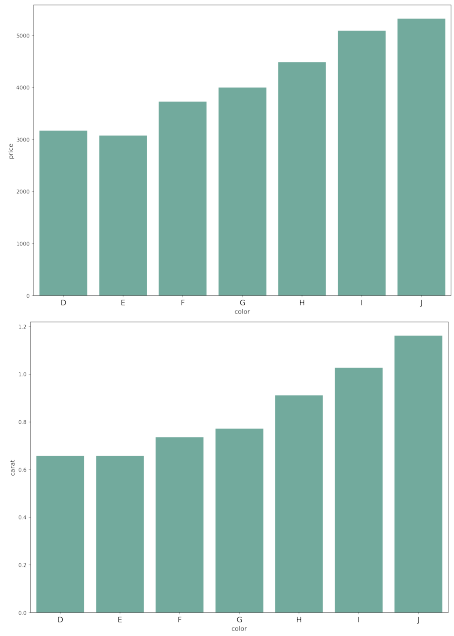

Figure 6. Histograms of the relation of the color with price and carat

E. feature Clarity

The following feature is the clarity which is a measurement of how clear the diamond is in the scale internally Flawless (IF), very very Slightly included (VVS1 e VVS2), very Slightly included (VS1 e VS2), Slightly included (S1 e S2) e Included (I1).

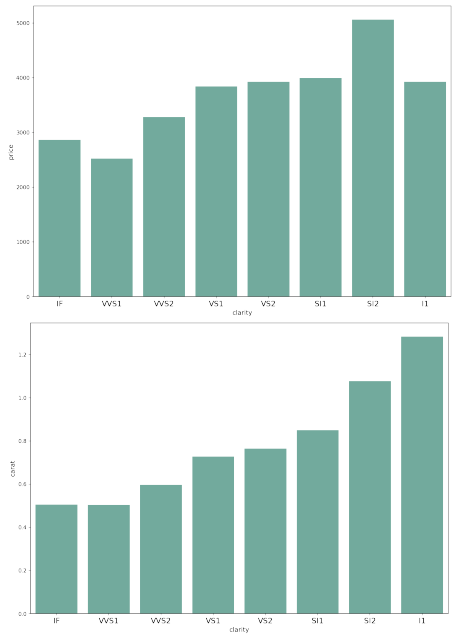

Figure 7. Histograms of the relation of the clarity with price and carat

The diamond clarity has the tendency , in media, to be in VS2, S1 and S2 lowering with cut quality. Also the price has

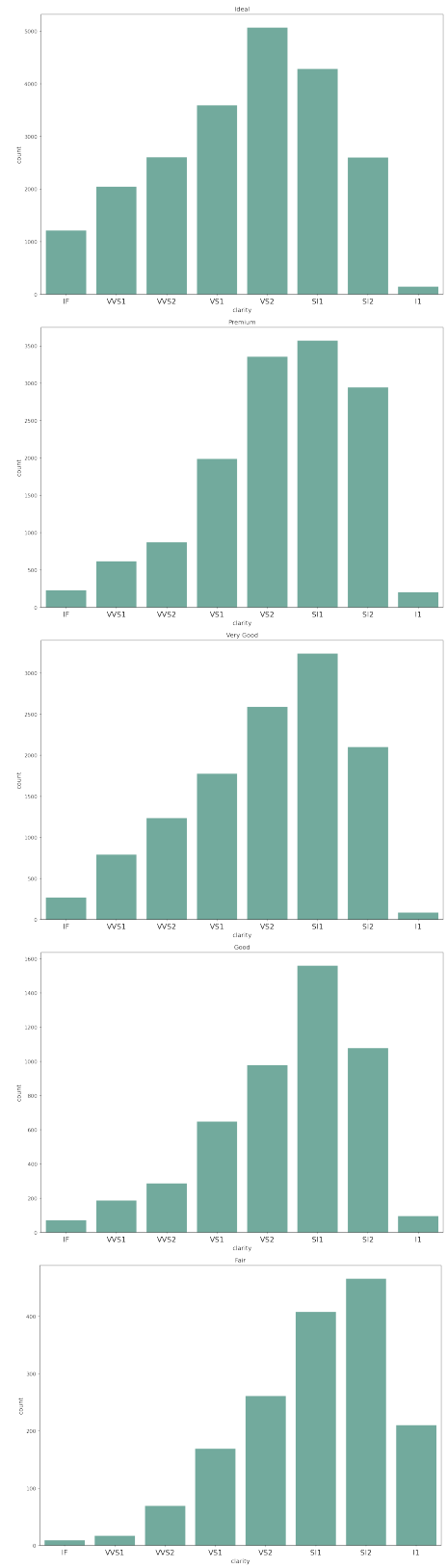

Figure 8. Histogram of the the clarity per cut quality

the tendency to grow if the weight of the diamond stays even as we can see from VVS1 to IF and I1 to S2.

F. feature x, y and z

The posterior feature is the volume of the diamond separated by their dimensions (x, y, z)

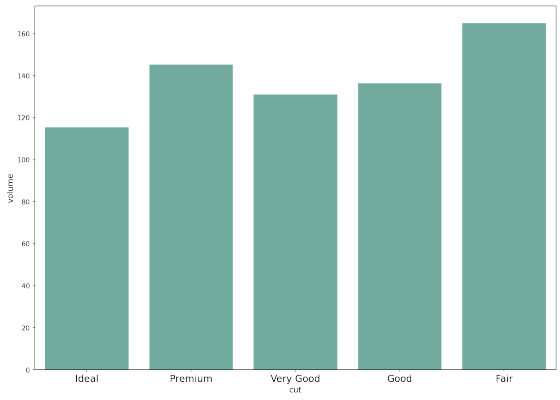

Figure 9. Histogram of the the volume per cut quality

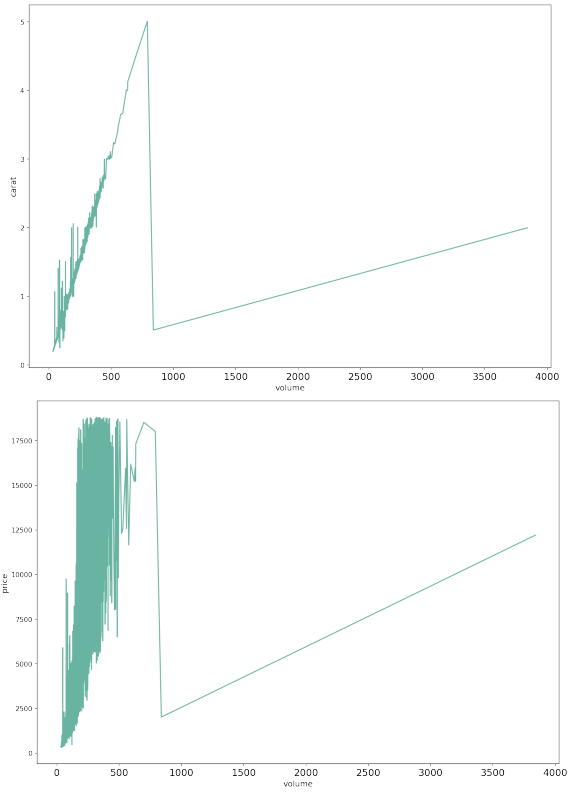

Figure 10. Histograms of the relation of the volume with price and carat

As we can see in figure 9 the bigger diamonds have the lower quality cut excepting the premium cut that have the second bigger diamonds. Looking at the figure 10 we can see the weight of the diamond have a very steep slope until close 1000 mm³ then having a big drop and increasing again this time slower, while the price of the diamond having a similar behavior however with high ups and downs until the same checkpoint. This can be explain because the diamonds with best cut, color and clarity have the smaller ones.

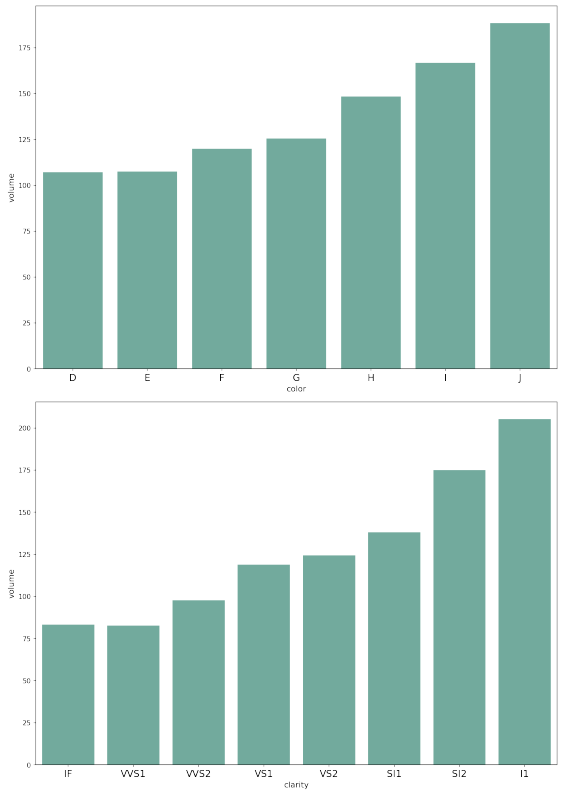

Figure 11. Histograms of the relation of the volume with color and the clarity

G. feature depth

The succeeding feature is the total depth measured by the expression z / mean(x, y) (or 2 * z / (x + y)).

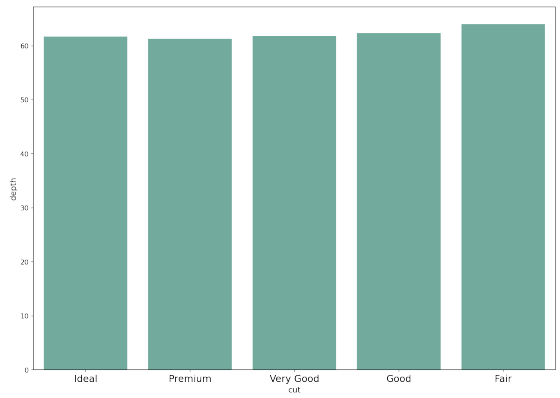

Figure 12. Histograms of the relation of the volume with price and carat

The depth of the diamond is very well balanced between all the cut qualities having the fair quality cut the slightly bigger value than the rest.

H. feature table

The final feature is the width of the top of the diamond relative to the widest point (table).

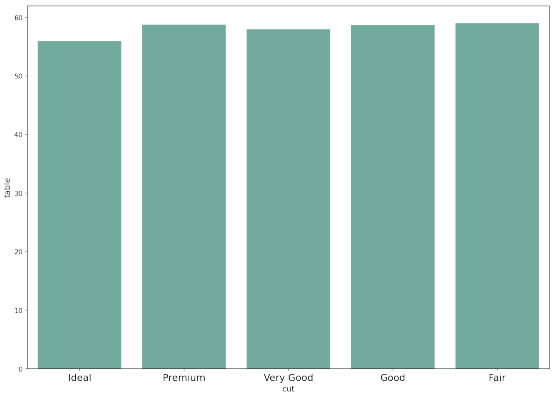

Figure 13. Histograms of the relation of the volume with price and carat

The diamond table also have a very well balance between all the cut qualities like the depth feature having however the premium cut the slight advantage with the other qualities

IV. Methodologies

A. Preprocessing and Standardization

Before starting analysing and building our models, the data set needed to be preprocessed and standardized. After analysing the contents of the data set as well as some statistics related to it, we started the preprocessing by removing the unnamed column that contained the index of each row of the data set, since when converting the data from the .csv file to a pandas DataFrame the index is managed by the Dataframe. To continue the preprocessing, we check the existence of null values, which if there were any it would affect the model building later on, and confirmed that no null values were present. We then spotted that the minimum value for the ”x”, ”y” and ”z” features were 0, which is impossible since these features represent the length, width and height of a diamond, respectively. There were 20 rows of the data set that had 1 or more of these features with value 0, and since we had more than 50000 rows, we decided that these 20 rows could be removed without affecting the models.

After the preprocessing, we started by defining what the target of our problem was, which we decided it would be the ”cut” feature. This feature can have 1 of 5 values ranging in increasing order: Fair, Good, Very Good, Premium, Ideal.

For the standardization, we started with the textual features ”color” and ”clarity”, which would need to be turned into numbers in order to be processed. To do this we decided to use One-hot encoding, which is the process by which categorical data are converted into numerical data for use in machine learning. Categorical features are turned into binary features that are “one-hot” encoded, meaning that if a feature is represented by that column, it receives a 1. Otherwise, it receives a 0.

As for standardization we started by analysing the distribution of each numeric feature to decide what type of normalization to use.

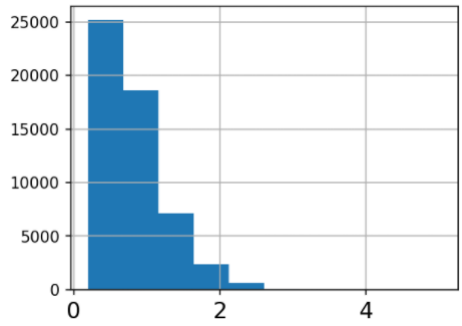

Figure 14. Distribution of the carat feature

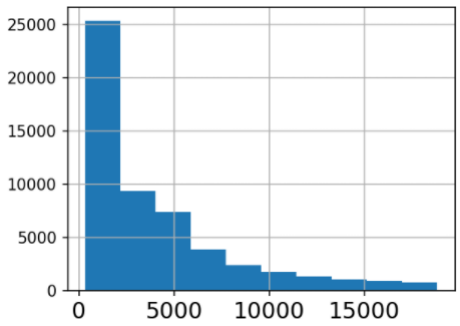

Figure 15. Distribution of the price feature

We found that the ”price” and the ”carat” features were skewed, meaning that one of the tails is longer than the other, as we can see in Figures 14 and 15. To make them as normally distributed as possible, we used a logarithmic transformer. For the other numeric features, since they had normal distributions, we did a simple standardization.

B. Classifiers and Confusion Matrices

After preprocessing and standardizing the data set, we were ready to build the models. For this project, we decided to use 3 classifiers: Random Forest, Logistic Regression and SVM. We started by splitting the data into training and testing sets, with training making 67% of the total data set and testing the remaining 33%. After we fitted each classifier with the training data sets and then used the testing data sets to check the classifiers accuracy, precision, etc..

Finally, in order to analyse the confusion matrix for each classifier, we defined our positive and negative classes. To define them, we thought of the problem at hand, and decided that we wanted the classifiers to differentiate Ideal and Premium cuts better than other types, making these our positive classes and the rest negative classes. After we defined the true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), which were defined as follow:

- **Positive classes:** Ideal and Premium diamonds.
- **Negative classes:** Very Good, Good, and Fair diamonds.

- **True Positives, type 1:** actual Ideal, predicted Ideal.
- **True Positives, type 2:** actual Premium, predicted Premium.
- **True Negatives:** the rest of the diamond types predicted correctly.
- **False Positives:** actual value belongs to any of the 3 negative classes but predicted either Ideal or Premium.
- **False Negatives:** actual value is either Ideal or Premium but predicted by any of the 3 negative classes.

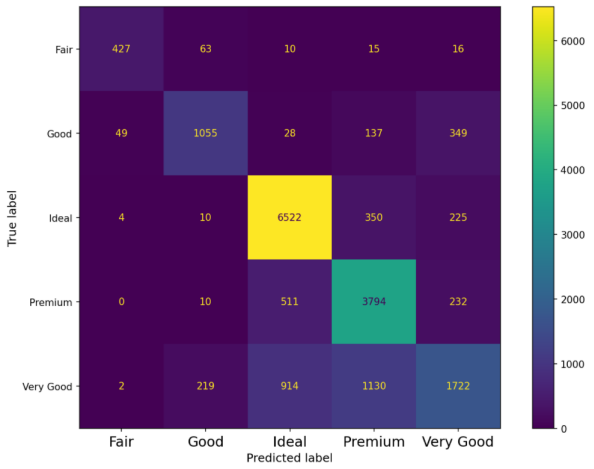

Figure 16. Confusion Matrix of the Random Forest Classifier

1) Random Forest: As we can see in the Figure 16, this classifier has a lot of True Positives and True Negatives. As for the False Positives and False Negatives, we can see that this classifiers has a few for the Fair and Good classes, but has a lot more for the Very Good class, which indicates that this model has difficulty in distinguishing the Very Good class from Ideal and Premium.

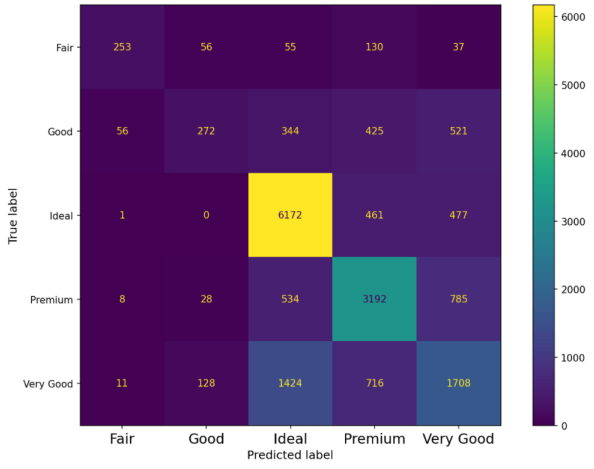

Figure 17. Confusion Matrix of the Logistic Regression Classifier

2) Logistic Regression: As we can see in the Figure 17, this classifier has a lot of True Positives, but contrasting with the Random Forest classifier, has a lot less True Negatives. Hence, there is an increase in the False Negatives and and False Positives, indicating that this classifiers is less accurate than the Random Forest one.

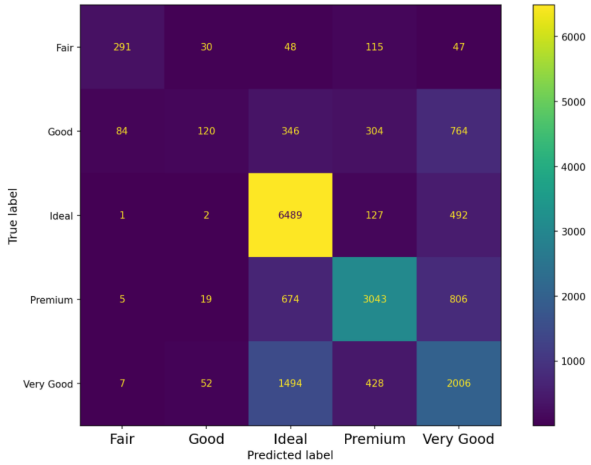

Figure 18. Confusion Matrix of the SVM Classifier

3) SVM: When viewing Figure 18, we can see that there is a high similarity with the confusion matrix of the Logistic Regression classifier, with a slight increase in the True Negative. Thus, we can conclude that this classifier has a similar performance to the Logistic Regression classifier.

C. Precision, Recall and F1 scores

There are some ways of evaluating the performance of the classifiers, and next we explain some of them.

Precision (1) tells us what proportion of predicted positives is truly positive.

$$Precision = TP/(TP + FP) \tag{1}$$

This metric is good if we want to minimize the False Positives.

Recall (2) is calculated similarly, but instead tells us what proportion of true positives were predicted correctly.

$$Precision = TP/(TP + FN) \tag{2}$$

This metric is good if we want to minimize the False Negatives.

But what if we want to reduce both False Negatives and False Positives? That is where the **F1 Score** (3) enters. This metric takes the harmonic mean of precision and recall and produces a value between 0 and 1:

$$F1Score = 2 * (P * R)/(P + R) \tag{3}$$

where:

P = Precision

R = Recall

With all this in mind, we decided to evaluate our classifiers by the F1 Score, but since we want to make our models identify better the Ideal and Premium classes, we will use the weighted F1 Score of these classes.

Analysing the Table I we can see than both in the "All Classes", which is the weighted f1 score of every class, and

Table I
F1 SCORES OF THE CLASSIFIERS

	Random Forest	Log Reg	SVM
All Classes	0.73	0.63	0.64
Ideal and Premium	0.82	0.74	0.77

"Ideal and Premium", which is how we will evaluate our classifiers performance, the **Random Forest** classifier has the higher values and has a significant difference from the other two classifiers, which have very similar values.

D. Hyperparameter Tuning

So far, all the classifiers that were used had custom parameters defined by us, but there are an infinite number of combinations of parameters and we do not know if the parameters that we chose are the best ones. So in order to find out the best parameters for our classifiers in a given range, since it's impossible to check every combination, we performed **Hyperparameter Tuning**. This process consists of a few steps:

- Define a model
- Define the range of possible values for all hyperparameters we want to test
- Define a method for sampling hyperparameter values
- Define an evaluative criteria to judge the model
- Define a cross-validation method

The first step is already done, as we already chose the **Random Forest, Logistic Regression and SVM** classifiers. The second step will be detailed further in the report. For the third step, we decided to use **Random Search**, which is similar to **Grid Search**, but instead of build and evaluting a model for every combination of values given, it randomly samples values from the values given. For the fourth step, as it was already mentioned, we will use the **weighted F1 Score of the Ideal and Premium classes**. For the final step, we chose to do a 3-fold cross validation.

The following 3 sub-subsections will depict what hyperparameters were chosen as well as the best combination for each classifier. The last sub-subsection will present the results of the classifiers with the best combination of hyperparameters and compare them with each other and the results from the first classifiers.

1) Random Forest: For the Random Forest classifier, we choose 4 hyperparameters:

- **n estimators:** The numbers of trees in a forest.
- **max depth:** The maximum depth of the tree.
- **min samples split:** The minimum number of samples required to split an internal node.
- **min samples leaf:** The minimum number of samples required to be at a leaf node.

After the hyperparameter tuning, the best combination of parameters was:

- **n estimators:** 1200
- **max depth:** 10
- **min samples split:** 5
- **min samples leaf:** 4

2) Logistic Regression: For the Logistic Regression classifier, we choose 2 hyperparameters:

- **C:** Inverse of regularization strength
- **penalty:** Norm of the penalty

After the hyperparameter tuning, the best combination of parameters was:

- **C:** 10000
- **penalty:** l2

3) SVM: For the Logistic Regression classifier, we choose 1 hyperparameters:

- **C:** Inverse of regularization strength

After the hyperparameter tuning, the best combination of parameters was:

- **C:** 0.18420699693267145

4) Results: After the hyperparameter tuning and the discovery of the best combination of hyperparameters for each classifier, we fitted 3 new classifiers, each with its best combination, and withdrew the tuned f1 scores.

Table II
F1 SCORES OF THE CLASSIFIERS BEFORE AND AFTER TUNING

	Random Forest	Log Reg	SVM
Base	0.824	0.744	0.767
Tuned	0.811	0.746	0.806

As we can see in Table II, there was a slight decrease in the Random Forest Classifier from the first classifier to the classifier with the best parameters. The Logistic Regression classifiers saw a very slight increase in value, whereas the SVM classifier got a significant increase in the F1 Score. Both before and after the hyperparameter tuning the **Random Forest** classifier has the highest value, with the **SVM** classifier coming close to it with the parameters obtained from hyperparameter tuning.

V. CONCLUSIONS

After all the different classifiers and hyperparameters tested and by analysing the results, mainly the ones present in Table II, we can say that the **Random Forest** classifier was the best classifier present, reaching the maximum value of 0.824. Although there is an decrease in value when using hyperparameter tuning, this is likely due to the method we used to sample the hyperparameters values, which has a random aspect to it, therefore not testing the combination of hyperparameters used in the base classifier. Nonetheless, while we believe that this classifier is the most suited for this problem, there can be improvements, specifically in the hyperparameter tuning of each classifier, that was limited to few parameters and few values, due to time and resources.

This project, composed of this report, code and a presentation, was made by both elements of the group, each with the same contribution of 50

REFERENCES

[1] https://towardsdatascience.com/comprehensive-guide-to-multiclass-classification-with-sklearn-127cc500f362
[2] https://www.kaggle.com/code/abdalrahmanshahrour/diamonds-data-pt1-preprocessing
[3] https://www.kaggle.com/code/karnikakapoor/diamond-price-prediction
[4] https://www.kaggle.com/code/fuzzywizard/diamonds-in-depth-analysis