Project report on Binary classification for item weight prediction on a simulated dataset of wares

Vladislav Veselov, Alexandr Macovei

July 6, 2023

1 Introduction

1.1 Data origins

The dataset: "Port of Los Angeles" is available on Kaggle. It was created by "MIKOŁAJFISH99" and is original. The dataset is simulated and simulates information on international shipments of a port. This information includes the name of the product, its price, weight, length, width, and height, as well as the date it was shipped and the destination port.

1.2 Scientific works with this dataset

This dataset has been used for several paper works. Some of them are: "International transportation costs around the world" , "Anomaly pattern detection in categorical datasets".

The overall point of scientific works mentions that using dataset classification for shipment purposes can prevent anomalies and undesired troubles. It is an easy way to spot the problem on its beginning and evade massive and expensive mistakes. Classification, as well, can help with logistics. Good classification and feature recognition might discover some common patterns , that might optimize the working processes and release resources.

1.3 State the problem

The task is a binary classification problem aimed at predicting the weight of wares based on a simulated dataset. The dataset consists of various attributes and features associated with different wares, and the goal is to develop a machine learning model that can accurately classify the weight of a ware into one of two classes: "Lightweight" or "Heavyweight."

Although in real life it is difficult to imagine a situation where it is necessary to perform a binary classification of the weight of an item, the problem can be generalized: if we get a good classifier model, we can determine the weight of an item from the

external dimensions of the box and its price. Weight above a certain threshold may mean an additional tax or fee.

However, the most interesting part of the project is to find out how various ML techniques will help improve the results of binary classifiers on an artificial dataset like this.

2 Data

2.1 Dataset description and explanatory plots

The dataset we are using is a simulated dataset simulating sea freight transportation. It has 263321 lines of data and contains the following features:

- name
- price ($)
- weight (kg)
- length (m)
- width (m)
- height (m)
- shipment date
- destination port

The dataset consists of a total of 291 item classes. Fig. 1 shows a plot depicting the relationship between the prices of items and their weights. The color represents the density of points on the plot. It is easy to observe that the plot consists of rectangle-shaped clusters, each of which has a uniform density. This happens because each cluster represents a specific class of items, and the attributes for each class were generated by a random number generator within a predefined range, e.g. the fragment in the lower right corner of the plot represents the "Palette of Laptops" item class.

However, the majority of items are concentrated in the lower left corner of the graph - at the origin, where the aforementioned rectangular regions intersect and overlap each other. This area is of particular interest to us. Fig. 2 represents this area.

To understand the distribution of price and weight of items on the graph, we can refer to the quantiles of weight (Table 1) and price (Table 2).

The plot on Fig. 3 depicting the relationship between the height, length, and width of items reveals three groups of item classes, which can be roughly referred to as "small" items, "long" items, and "large" items. The vast majority of items are in the "small" group.

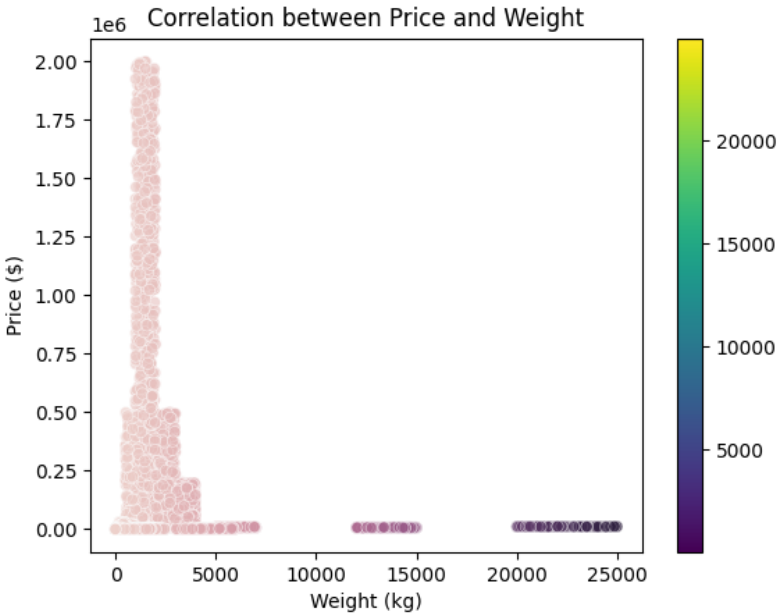

Figure 1: Relation between price and weight with density.

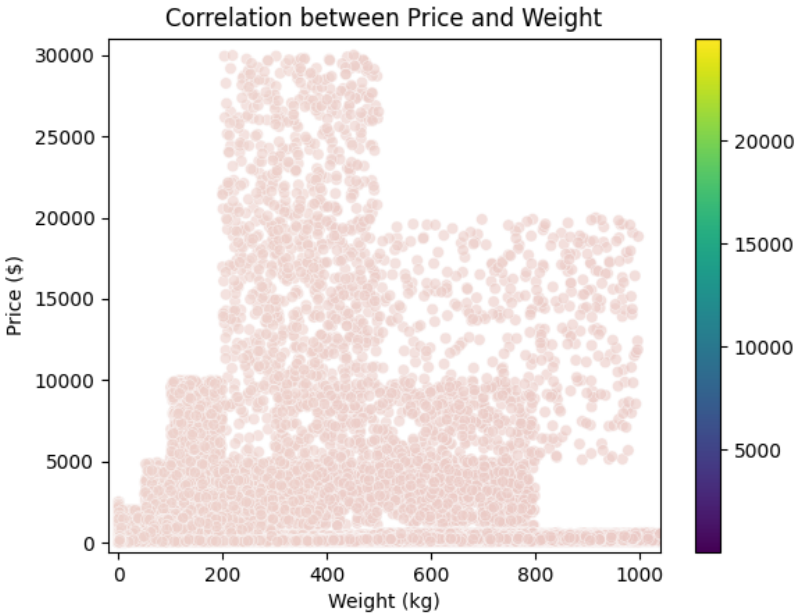

Figure 2: Relation between price and weight with density, zoom in.

Quantile	Weight, kg
0.25	0.71
0.50	2.96
0.75	67.2925
0.95	1372.7930

Table 1: Weight quantiles.

Quantile	Price, $
0.25	27.85
0.50	87.61
0.75	257.76
0.95	1549.33

Table 2: Price quantiles.

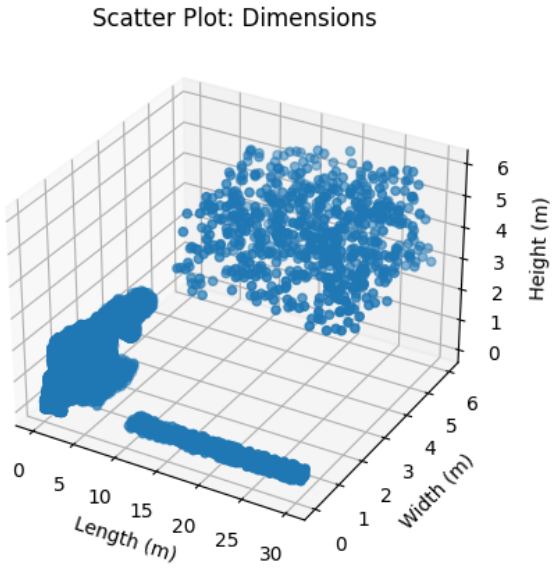

Figure 3: Relation between height, width and length of the items

2.2 Motivation of problem statement

This artificial dataset is nothing like real world data. It is interesting to see how classifiers that apply to the real world and have appropriate assumptions, such as the Gaussian distribution, can handle classification on this dataset. The parameter that we will classify will be weight, because it does not directly depend on size and price, and we do not have a feature with density and volume that could allow us to calculate the exact weight. So, this is a suitable field for ML.

2.3 Exclusions/inclusions

In this project, we are utilizing the following features:

- price ($)
- weight (kg)
- length (m)
- width (m)
- height (m)

Also, 2860 lines of data (1.08 percent of the dataset) were not used in the project and were deleted from the dataset during the pre-processing because of the missing data in these lines.

2.4 Separation into train, validation and test data

To evaluate the performance of our model and implement feature selection, we divided the dataset into training, validation and testing sets. We used a 70-15-15 split, where 70 percent of the data was used for training, 15 percent for the validation and 15 percent for testing.

3 Method

To study the performance of different classification algorithms on a rigidly generated simulated dataset, it is necessary to divide the data of one of the features into two classes. This will make possible binary classification based on other features of the dataset. Taking into account the weight quantiles (Table 1), we can choose the weight median, 2.96 kg, as the split point. All items lighter than this weight will be labeled as "light," while the remaining items will be labeled as "heavy". Such a threshold will make the dataset balanced and will allow using of the 'accuracy' for evaluation of algorithms.

The plot (Fig **??**) depicts how heavy and light objects are located in height-weight-width coordinates.

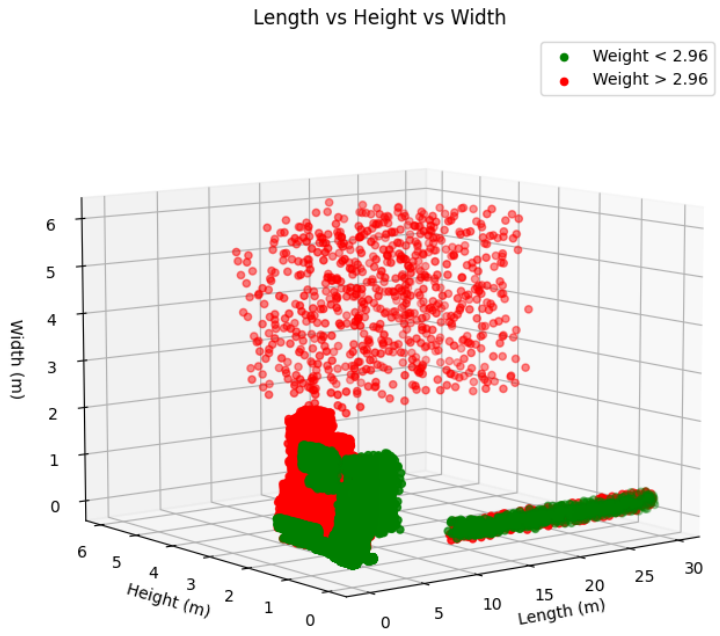

Figure 4: Red and green dots overlap each other in the most dense clusters

1. After creating the classes for classification, we used some classificators to see how they work without any additional techniques and evaluated the results.
2. Then we implemented dataset normalization and repeated step 1.
3. Then we lifted up feature space and repeated step 1.
4. Then we did feature selection and repeated step 1. Feature selection was done by forward feature selection with backward elimination.
5. Then we implemented outliars selection and repeated step 1.
6. Then we implemented resampling and repeated step 1.
7. Then we evaluated our best models with cross-validation.

The following classification algorithms were studied on the dataset:

1. GNB (Gaussian Naive Bayes)
2. KNN (k-nearest neighbors algorithm)
3. LDA (Linear Discriminant Analysis)
4. QDA (Quadratic Discriminant Analysis)

On each step, the accuracy of performance was evaluated. More detailed analysis of the accuracy was conducted using confusion matrices and ROC curves.

Based on the knowledge of the strengths, weaknesses, and assumptions of each algorithm, an analysis was performed to identify factors preventing each algorithm from achieving maximum accuracy.

4 Results and discussion

4.1 On a raw dataset

GNB Gave the worst accuracy: 0.5688. The confusion matrix is the following: [6264 32884][877 38264]. The matrix shows that most of the decisions were negative, that is why we have so many false negative decisions. ROC on Fig. 5 doesn't demonstrate this flow.

LDA Gave accuracy: 0.8021 with the following confusion matrix: [26991 12157][3336 35805]

QDA Gave accuracy: 0.6894. Confusion matrix: [16668 22480][1838 37303]

KNN Gave the best accuracy: 0.8777. Confusion matrix: [34441 4707][4865 34276]

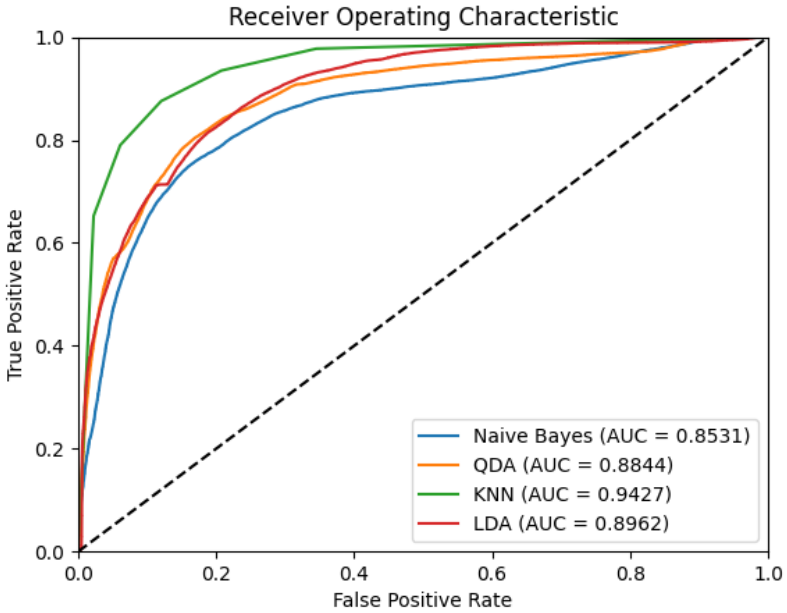

Figure 5: ROC can be misleading: Naive Bayes on a raw dataset works worse than it seems in the ROC diagram

4.2 On a normalized dataset

Z-score normalization was applied to the training data. This formula was used to perform a z-score normalization on every value in dataset:

New value = (x – (Mean of data)) / (Standard deviation of data)

GNB Accuracy improved tremendously: 0.7088. The confusion matrix is the following: [18180 20968][1830 37311]

to understand why there were so many false negatives before normalization and why this problem disappeared after normalization, we tested how the classifier works without certain features. It turned out that the feature called 'price ($) has the most negative impact on the dataset.

Let's break down how how the GNB predicts:

$P(y_k \mid x) = P(y_k \mid x_1) \times P(y_k \mid x_2) \times P(y_k \mid x_3) \times ... \times P(y_k \mid x_p)$.

That is, every feature is treated equally in terms of the result. The fact is that before normalization class Price had a much greater variance then the others, and the distribution was such that items with any price, except for a small interval from 0 to 1000 dollars, are overwhelmingly heavy. Because of this, the classifier learned ti assign almost any x from the set of Price an extremely high probability that it is heavy. Thus the feature Price overshadowed the rest of the features.

After normalization, variance decreased and the feature Price was not dominating

the result as powerful as before. We did not expect this, since when the dataset is normalized, the structure of the feature does not change, but only the scale. The reason for this improvement could not be established. It is clear that variance has decreased, but it is not clear how its reduction improved feature price predictions.

LDA Same performance as before. Which is expected, because scaling doesn't influence LDA decision boundary.

QDA Same performance as before. Which is expected, because scaling doesn't influence LDA decision boundary.

KNN Accuracy improved a bit: 0.8801. Confusion matrix: [34180 4968][4363 34778]

4.3 On a dataset with feature space lifted

During the feature space lifting, part of the feature engineering (FE) process, new features were added to the dataset:

- volume (m3) (length*height*width)
- surface area (m2) (height*width+height*length+width*length)
- length-to-width ratio (length/width)
- price-to-volume ratio (price/volume)

GNB Accuracy decreased a bit: 0.6936. The confusion matrix is the following: [16435 22713][1274 37867]

LDA Accuracy increased a bit: 0.8072. Confusion matrix: [28098 11050][4047 35094]

QDA Accuracy decreased: 0.6358. Confusion matrix: [11941 27207][1304 37837] QDA assumes quadratic decision boundaries, and new features could be unrelevant for that, while LDA with linear decision boundaries improved it's accuracy.

KNN Accuracy increased: 0.8954. Confusion matrix: [35014 4134][4058 35083]

4.4 On a dataset after feature selection

Forward selection and backward elimination algorithm was implemented for feature selection. This is the part of the project, where we used validation data (we could not use test data to avoid overfitting). The results:

GNB The following features were selected: 'width (m)', 'height (m)', 'price-to-volume ratio', 'length (m)', 'length-to-width ratio'. The accuracy increased to 0.8147.

LDA No increase in accuracy - best results with the most features.

QDA The following features were selected: 'width (m)', 'height (m)', 'length-to-width ratio' The accuracy increased to 0.7942

KNN No increase in accuracy - best results with the most features.

4.5 On a dataset with feature space lifted II

When we did the initial feature space lift, we did it carefully, as we were afraid of the Dimensional Curse. But during the evaluation process, we were surprised that two classifiers, LDA and KNN, improved their performance with an increase in the number of features - that is, the more features, the better they worked. Because of this, we've created even more features, most of which don't make sense in the real world, to see how classifiers will be affected. We were expecting to see some decrease.

Our new features were the following:

- height-to-width ratio
- height-to-length ratio
- price-to-length ratio
- price-to-width ratio
- price-to-height ratio
- horizontal area (length * width)
- vertical area 1 (length * height)
- vertical area 2 (width *height)
- price squared
- length squared
- width squared
- height squared

GNB Accuracy: 0.6531. Confusion matrix: [13042 26106][1053 38088]

LDA Accuracy: 0.8198. Confusion matrix: [30598 8550][5560 33581]

QDA Accuracy: 0.7097. Confusion matrix: [17467 21681][1044 38097]

KNN Accuracy: 0.8981. Confusion matrix: [35091 4057][3918 35223]

That is, despite accuracy decreased (as expected) for QDA and GNB, accuracy still increased for both LDA and KNN! A possible explanation is that our dataset with hundred thousands of samples is large enough for such a number of features, and even though most of them are not applicable to the real-world, they can still increase LDA and KNN.

4.6 On a dataset after outliers handling

Thus, our best accuracy values for the classificators are:

- Accuracy for GNB: 0.8147
- Accuracy for LDA: 0.8198
- Accuracy for QDA: 0.7942
- Accuracy for KNN: 0.8981

To increase the accuracy, we used outliars handling. In our specific case, we encountered outliers represented by light objects with unusually high prices. By excluding these outliers, we obtained a dataset where the relationship between price and volume became more pronounced and unidirectional. The removed outliars are represented on Fig. 6

Thus, after outliars handling we have:

- Accuracy for GNB: 0.8129
- Accuracy for LDA: 0.8208
- Accuracy for QDA: 0.7971
- Accuracy for KNN: 0.8981

LDA and QDA benefited, GNB's accuracy decreased a bit and KNN's accuracy didn't change. Therefore, outliers were not influencing the KNN with k=5.

4.7 On a dataset after resampling

In the provided code, the oversampling technique, random oversampling, specifically targets the minority class, which is defined as the "heavy" and "cheap" items in the dataset. By applying random oversampling to the minority class, synthetic examples are generated to balance the class distribution. This ensures that the model has an adequate representation of the minority class during training, which is crucial when the minority class is underrepresented. The goal is to improve the classifier's ability to accurately classify the "heavy" and "cheap" items, which are typically fewer in number compared to the majority class. By addressing this class imbalance through oversampling, the model can achieve better performance and make more reliable predictions for the minority class.

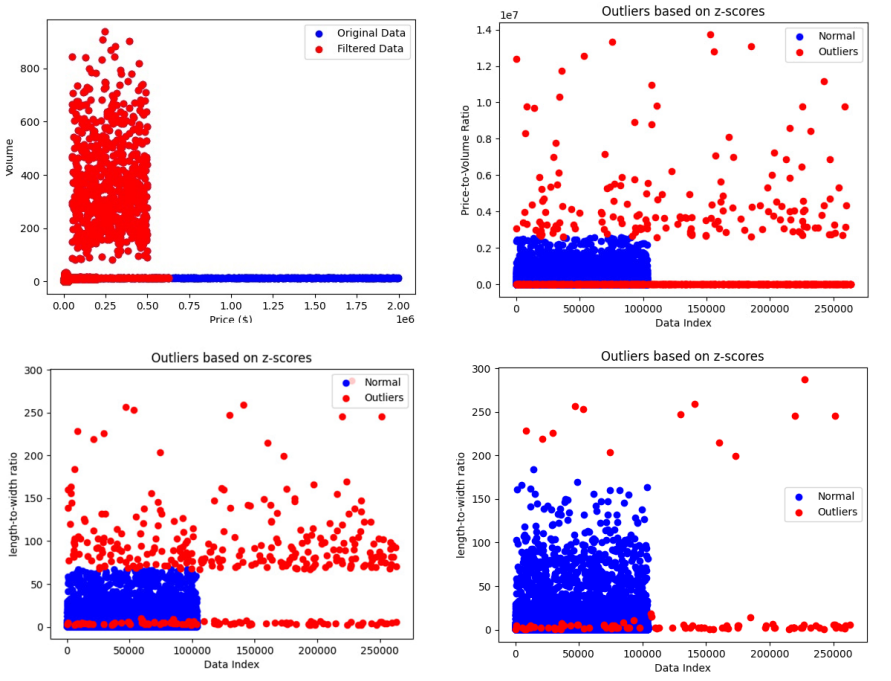

Figure 6: Outliars on different coordinates.

- Accuracy for GNB: 0.8321
- Accuracy for LDA: 0.8220
- Accuracy for QDA: 0.8003
- Accuracy for KNN: 0.9001

Indeed, every classifier's accuracy increased.

4.8 Cross-validation

...

5 Conclusion

5.1 Lessons learned

...