Evaluation of Machine Learning Models for Estimating Sales in Physical Retail

Geovanne O. Alves[1], Jorge C. B. Fonsêca[1], Alexandre M. A. Maciel[1]

Universidade de Pernambuco, Brazil
goa@ecomp.poli.br
jorge.fonseca@upe.br
alexandre.maciel@upe.br

Abstract. The amount of sales in a store is a strong indicator that contributes to managers' decision making. In physical retail, unlike e-commerce, it is more difficult to collect sales and customer behavior metrics because it depends on great sensing and integration between systems. In a shopping mall scenario, we use real WiFi data, People Flow and Sales create a dataset. In this article we propose an evaluation of machine learning models with the objective of estimating the next hour sales in Low, Medium and High, thus providing a tool to assist in decision making. We use the PyCaret library to perform the training of the 13 compared algorithms. The F1-score metric was used to evaluate the models. The Gradient Booster Classifier was the model that got the best result with a score of 84.75%. Among the estimated classes, the High class showed the greatest error in the confusion matrix, reaching 60%, possibly a reflection of the low amount of records in the high class.

CCS Concepts: • **Computing methodologies** → **Machine learning algorithms**.

Keywords: data mining, estimate sales, evaluation models, machine learning

1. INTRODUCTION

In retail, the amount of sales is one of the main indicators that can inform the health of the company, therefore, generating sales forecasts is a good tool that can help the manager in directing the business [Zunic et al. 2020]. When this store is located on the internet as an e-commerce, various metrics can be collected in relation to customer behavior and sales, for example: time spent viewing a product, number of clicks, items in the cart, number of sales, average sales time and others [Shaytura et al. 2017]. Once these data are collected in a structured way, they can be input to an artificial intelligence system and thus be able to generate estimates, clusters and classifications to help the administrator's decision making.

However when we are dealing with physical retail, especially with a shopping mall that has several stores inserted in the context, making the extracted data to be converted into metrics in the physical location is a difficult task, moreover, when the company is still starting to acculturate data driven in your environment. With these unstructured data, preprocessing is essential in order to refine and evaluate the data with a considerable effect on the overall results analytics [Taleb et al. 2015].

The purpose of this article is to perform a comparison of machine learning models in predicting sales in a physical location that helps support the organization's decision. For this it was necessary to create a database with information on sales and public behavior and also identify and remove outliers arising from data extraction problems.

this work was partially funded by XXX (grant #0000)

2. BACKGROUND

2.1 Machine Learning

Machine Learning (ML) is a subfield of artificial intelligence, in which mathematical models are built from a set of data, known as training data, seeking to make predictions or decisions without being programmed to perform this specific function [Zhang 2020]. It can thus be said that ML models learn to estimate or classify from training data. There are two major classifications for ML, the supervised learning model, which together contains the expected output as a label, and the unsupervised model, in which the learning seeks to find patterns and knowledge in unlabeled data.

The supervised algorithms used in this article are: Logistic Regression (LR)[1], Gradient Boosting Classifier (GBC)[1], Light Gradient Boosting (LGBM)[2], Linear Discriminant Analysis (LDA)[1], Random Forest Classifier (RF)[1], Extra Trees Classifier (ET)[1], Ridge Classifier (ridge)[1], SVM - Linear Kernel (SVM)[1], Decision Tree Classifier (DT)[1], KNeighbors Classifier (KNN)[3], Ada Boost Classifier (ADA)[1], Naive Bayes (NB)[3], Quadratic Discriminant Analysis (QDA)[4]; And the unsupervised algorithm used is KMeans[5].

2.2 Evaluation Metrics

The metrics used to assess the performance of the models was the F1-score. This metric is calculated through the confusion matrix that portrays the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). To calculate the F1-Score, we first need to calculate recall, which is TP divided by the sum of TP and FN, and precision, which is TP divided by the sum of TP and FP. Recall indicates the correct classifications among all that it expects to be correct, and precision indicates the correct classifications among all that are classified as positive. With these two metrics calculated, we can then calculate the F1-Score (Eq. 1).

$$f1\text{-}score = 2 * \frac{precision * recall}{precision + recall} \tag{1}$$

The F1-Score is built from the harmonic mean between precision and recall. This is a good metric to use when seeking to achieve a balance between recall and precision, especially when there is imbalance between classes [Nabipour et al. 2020].

2.3 Related Works

The article by [Triayudi et al. 2020], they used a company's historical sales transaction data as input to a time series method to forecast each month's sales. Three years were used as training and 1 year as a test, of which the month with the highest accuracy was 99.68% and the one with the lowest accuracy was still above 50%. The work of [Gonçalves 2019], Two sets of data, Facebook Ads and Google Ads, were used, with indicators such as the number of clicks and customer impressions to predict the real-time cost and revenue of e-commerce. The authors performed a comparison between multiple regression, which showed the best result, and Arima.

In this article, we seek to use data on customer behavior, WiFi database and People flow, together with sales data, seeking to forecast sales for the next hour in the physical shopping mall.

[1][Géron 2019]
[2][Taha and Malebary 2020]
[3][Zhang 2020]
[4][Srivastava et al. 2007]
[5][Sinaga and Yang 2020]

3. EXPERIMENTAL METHODOLOGY

In this session, we will demonstrate the steps used to compare algorithms, from designing the dataset to obtaining the metrics. The image 1 represents the steps performed which will be described below.

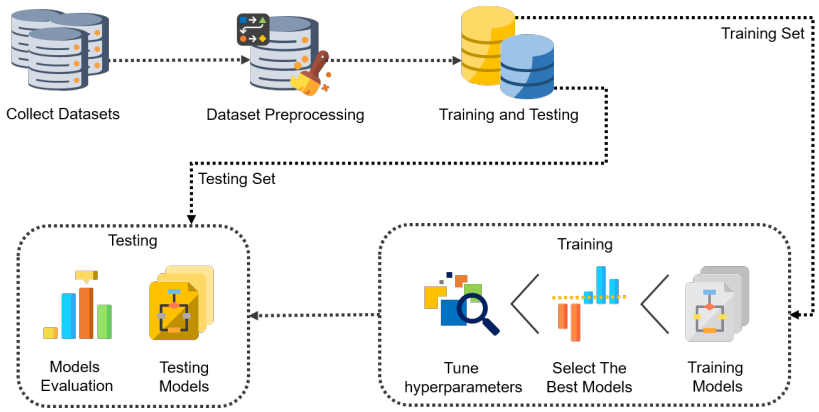


Fig. 1: Experimental methodology used to compare algorithms.

3.1 Collect Datasets

In order to make the forecast more accurate, we use data that indicates the behavior and characteristics of users in addition to sales data. The data used for this process corresponds to 8 months in the period from October 2020 to May 2021 and includes WiFi, People flow and Sales data.

The WiFi dataset is maintained by the Social Hotspot Platform[6], which collects user data from questionnaires when the customer logs in and information during use, initially we have 164356 lines and 27 columns. The People Flow database is maintained by the Fxdata Intelligence system[7], which counts the flow of people in all entrances and exits of the shopping mall through sensors, in this base we have 5832 lines and 2 columns. The Sales database is maintained by Esphera Napp Solutions[8]. This is an audit system integrated with store cashiers to collect sales data, in this set we have 582929 lines and 4 columns.

3.2 Dataset Preprocessing

To build a dataset that could be used as input in ML systems, iterative preprocessing steps were necessary. The image 2 represents the steps for preprocessing and these steps are described below.

In the **Flow dataset preprocessing**, two columns were selected, Key and Value. The Value column, which represents the total number of people entering the mall, has been renamed to Number of Entries for ease of understanding. This set doesn't are N/A (Not Applicable) values. To generate a future dataset key, the values were converted and some new columns were created, this same processing was applied to other datasets described below. The columns Date, Day and Hour were added and populate with a conversion of column Key, where Date is day, month and year, Day is weekday name, and Hour is an hour number. Thus, after this conversion, we have 5832 rows and 4 columns.

In the **WiFi dataset preprocessing**, five columns were selected, Login date, Gender, Age, Online time, Mac Address, which have information about Date, qualitative personal identification and unique

[6]https://dtnetwork.com.br/servicos-e-solucoes-em-ti/hotspot-social-wifi-para-clientes-completo/
[7]https://www.fxdata.com.br
[8]http://conteudo.nappsolutions.com/3tom-esphera-versao-ingles

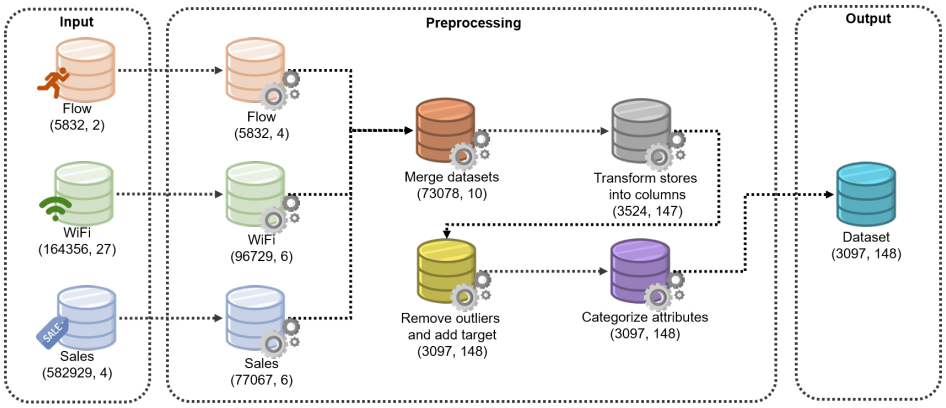


Fig. 2: Step-by-step Preprocessing of datasets.

identification. Also were removed the N/A values. After this selection and remotion we have 155951 rows and 5 columns. In column Age were removed inconsistencies like some lines without value and registers with value more than 100 years old. Also are removed the suffix 'Years' and converting the value string to a number. After this remotion and transformation, this Flow dataset has 155846 rows and 5 columns.

Still in the WiFi preprocessing, in the column Gender were removed inconsistencies where the value is equals 'Gender not informed'. After this remotion, we have 155190 rows and 5 columns. To generate a future key, the columns Date, Day and Hour were added and populate with a conversion of column Login date. Also, the column Online time is converted to DateTime format aiming to facilitate future aggregations. After this conversion, we have 155190 rows and 7 columns.

To resolve a problem of the data extraction platform, which has a lot of re-connections when the user tries to use WiFi generating many similar registers just modified the column Online time. Therefore were unified repeated rows by hour summing Online time by MAC Address. So, we have removed the column Mac Address. After this aggregation, we have 96729 rows and 6 columns.

To facilitate to apply of the algorithm in this dataset were transformed the column Online time from DateTime to seconds, and the column Gender binarized from Female, Male to 0,1. Lastly, this WiFi dataset has 96729 rows and 6 columns.

In the **Sales dataset preprocessing**, three columns were selected, Name, Total and Date, which have information about Date, sales values and store identification. In dataset doesn't are N/A values. After this selection, we have 582929 rows and 3 columns. As with the previous datasets, the future key, which is made up of the Date, Day, and Time columns, has been added and populated with a conversion of the Date column. After this conversion, we have 582929 rows and 5 columns.

Aiming creates a unique register per hour for each store, was converted the column Total into the Number of tickets and the Average tickets. Lastly, we are removing the column Total. In the end, this Sales dataset has 77067 rows and 6 columns.

In step **Merge datasets**, the aiming is to append and concatenate the three datasets previously processed to create just one dataset. We used a Flow dataset as a baseline to start concatenation because this dataset has all days and all hours populated. Always using Date, Day and Hour as Key. At first, we intend to concatenate Flow dataset and WiFi dataset, but notice that the WiFi dataset has more rows than the Flow dataset, soon we have to create some values which represent the set in that Key. They're below:

—In WiFi[Gender], we get the mode for each subset, thus getting the Gender that more appears;

—In WIFI[Age], we get the Median for each subset, getting the cental value among the sample avoiding the extreme points;

—In WIFI[Online time], we also get the Median for each subset, thus getting the cental value among the sample avoiding the extreme points.

After this aggregation, we have the Dataset Flow and Wifi with 5832 rows and 7 columns.

Going on with the preprocessing, is the time to concatenate a new baseline dataset, conceived at the junction of the Flow and WiFi dataset, with Sales dataset, we have to merge these sets without getting rid of information. Therefore for each line of the Sales dataset is added into the baseline dataset, in case of has a repeated key in the Sales dataset is duplicate the line referring to the key in the baseline dataset. I.e, the lines of the baseline dataset are replicating enough to adjust with the Sales dataset. After this aggregation, we have the one dataset containing Flow, Wifi and Sales with 78973 rows and 10 columns.

How is known, the Flow dataset contains all keys. When were unified the other datasets some inconsistencies appear. For example: Missing some WIFI values in some keys, the same way missing some Sales values. To resolve this question we remove any line with a NaN (Not a Number), in other words, Undefined Value. After this remotion, we have the new dataset which containing Flow, Wifi and Sales information with 73078 rows and 10 columns.

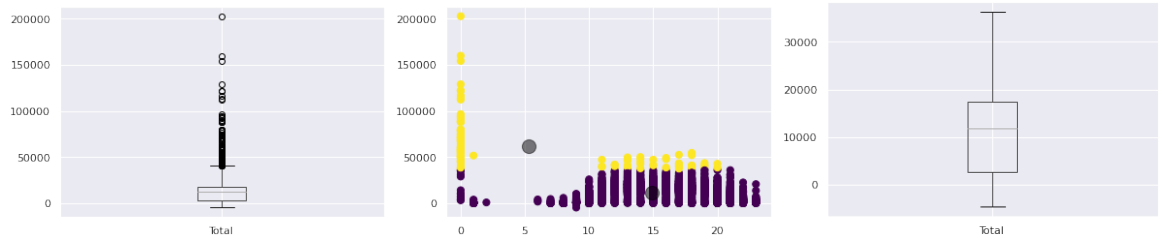
In step **Transform stores**, the aiming is to transform the Sales[Store] information into columns. For each unique set of key, which is repeated for each store, it will be kept only in one line, for this it is necessary to transform the store information into columns.

The store name and its information were concatenated with a dot (.), for the two columns referring to the store we have something like this, Store name.Number of tickets and Store name.Average ticket. So for each store, we created two columns. Stores that have no value will be set to zero. After this transformation, we have the new dataset with 3524 rows and 147 columns.

In step **Remove outliers and add target**, the aiming is to remove the outliers analyzing through clustering and create a new column called Total (T+1) to sum the amount collected for the next hour. To start, we need to calculate the Total for each line as a new column using the sum of multiplication the Number of tickets and Average tickets, and your dispersion was analyzed by a boxplot chart (image 3.a), which is evident the outliers are very discrepant.

We seek an approach to identify and separate outliers, which would possibly bias the ML model, from the dataset. Were used the Kmeans Clustering with 2 classes, seeking split between normal and outliers data, with columns Hour as an identifier and Total like a target (image 3.b).

With the graphical representation of Kmeans Clustering we were able to analyze the creation of groups and the distribution of the Total as a function of Hour. In the yellow color group that is represented in the chart, it contains 130 items, and in the purple color group, it contains 3394 items. It can be observed that items with high value in the Total are commonly registered at midnight, when questioning this behavior with data providers it was exposed that these values come from stores that are not fully compatible with the audit system and do not represent reality. So we consider the yellow group as an outlier and remove it. As the result of this action, we notice that the boxplot chart no longer considers any outlier (image 3.c). After this outlier removal, we have the dataset with 3394 rows and 148 columns.



(a) Total column boxplot with outliers (b) Kmeans clustering (c) Total column boxplot without outliers

Fig. 3: Outliers detection

Aiming at greater reliability were filtered the dataset between 9 and 22 hours, that is the business hours of the mall, and updated the target to get the total sales for the next hour, Total (T+1), based on the Total column created earlier. As we will not have data in the last line, it is necessary to delete it. Here too is dropped the column Total because not will be used. After this Filter and Update, we have the dataset with 3097 rows and 148 columns.

In step **Categorize attributes**, the aiming is categorizing the numerical attributes into 3 classes. We create categories for each column except Date, Day, Time and Gender. The objective is to provide an abstraction of numerical data in classes, contributing to the classifier's learning in estimating Total Sales. For this, the maximum and minimum values of the attributes were obtained, then we created a coefficient using the division by 3 of the difference between maximum and minimum, that is, the coefficient represents one-third of the attribute distribution (Eq. 2).

$$coefficient = \frac{(maximum - minimum)}{3} \tag{2}$$

With this coefficient, 3 classes were created that represent low (0), medium (1) and high (2) in each column, including the target. The replacement of dataset values by categories is given by the function $f(x)$ (Eq. 3), which $f(x)$ between 0 and 1 are replaced by 0 (low), $f(x)$ values above 1 up to 2 are replaced by 1 (medium), and values of $f(x)$ above 2 are replaced by 2 (high).

$$f(x) = minimum + (coefficient * x) \tag{3}$$

Lastly, was transform the values of column Day to number, where Sunday, Monday, Tuesday, Wednesday, Thursday, Friday and Saturday were replaced respectively from 0 up to 6. Also was remove column Date because it is not relevant to the ML model. After this categorization, we have a new dataset that has 3097 rows and 148 columns.

Analyzing the target, we have an unbalanced distribution between classes with 36.7% (1137) lines in the Low class, 53.6% (1661) in the Medium class and only 9.7% (299) in the High class.

3.3 Training and Testing

For the training of ML models, 70% of the final dataset was used and the other 30% of the dataset for testing. For training we use the open source library PyCaret[9], it makes it possible to perform the comparison of classification algorithms with little code and quickly.

[9]https://pycaret.org/guide/

At first, we need to configure the PyCaret environment, to build this setup we provide the training base, the target we want to classify 'Total (T+1)', the features that are all the other columns in the dataset. Also the minmax normalization preprocessing task and the division of the dataset into 70% for training and 30% for validation in a stratified way, i.e balanced distribution between classes.

Using the function compare_models[10], the algorithms, presented in the section 2.1, were trained with the hyperparameters default declared by Pycaret, the cross-validation technique with 10 folds and 10 random iterations in the search space. The models were ranked by the F1-score metric and, searching for the best, the 5 models that had the highest score in this metric were selected.

These selected models were submitted to the tune_models[11] function to optimize the hyperparameters with the randomized grid search technique, this technique depends on the number of iteration to find a better fit, so to tune we use 30 random iterations of the search space and 10-folds for cross-validation. The metric to be optimized was the F1-score.

In the test dataset, minmax normalization was also applied. And for each model selected and tuned, the test dataset was executed and its due matrices of confusion were generated, as will be exposed in the section 4.

4. ANALYSIS AND DISCUSSION OF RESULTS

During training, 13 classification algorithms were compared. They were analyzed from the perspective of the F1-score metric as we sought a balance between precision and recall. In the Auto column of the table I, we can see the values of the F1-score metric when training all models with the library's default hyperparameters. It is noted that among all the algorithms only QDA and NB had a score below 70%. Among the other 11 algorithms, we had a proximity between scores ranging from 75.07% to 84.75%, aiming to improve the best models, we chose to select the top 5 models, they are GBC, LGBM, LR, ET and RF.

Table I: Algorithm results using the F1-score metric.

	AUTO	TUNE
GBC	0.8475	0.8615
LGBM	0.8352	0.8492
LR	0.8299	0.8305
ET	0.8268	0.8279
RF	0.8213	0.8339
LDA	0.8166	-
SVM	0.8089	-
RIDGE	0.7991	-
ADA	0.7854	-
DT	0.7790	-
KNN	0.7507	-
QDA	0.4956	-
NB	0.2590	-

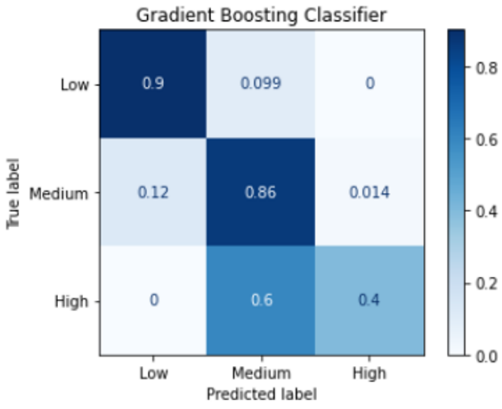


Fig. 4: GBC model confusion matrix.

The 5 selected models were tuned looking for hyperparameters that best fit the data. All tuned models resulted in an improvement in the F1-score metric, however, the order remained with the GBC being the model with the best F1-score, followed by LGBM, LR, ET and RF.

When we analyze the GBC model confusion matrix generated from the test dataset (image 4). We noticed that in the High class it was the one that got the biggest error, 60% error rate, a factor that

[10]https://pycaret.org/compare-models/
[11]https://pycaret.org/tune-model/

could be influencing would be the small number of records belonging to the High class, resulting from the imbalance between the classes. Already in the Low class presented a great result with 90% of correctness, in the Medium class with 86% of correctness rate.

5. CONCLUSION AND FUTURE WORKS

As discussed at the beginning of this article, sales estimates are a good tool for managers to have in their hands, but when placed in the physical context of a shopping mall, difficulties arise in extracting and building a representative dataset.

Seeking to meet the objectives, we created a new dataset with real information from a shopping mall and used this set to compare ML models in order to classify the total future sales in the next hour into Low, Medium or High. Among the algorithms compared, the one that had the best performance in the F1-score metric was the Gradient Booster Classifier (GBC).

As improvements and future work, we intend to perform hypothesis testing to choose the best algorithm for future sales classification, verify the performance of ensembles with the best models, use time series algorithms for a more sophisticated prediction, divide the dataset by same-industry stores to predict a specific group, increase the granularity of hour-to-day prediction, and use feature selection techniques to decrease the dimensionality of the dataset.

ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. I appreciate all the support of the POLI/UPE, the Data Science and Analytics Group (GPCDA), Group of Smart Research Labs (SmartLabs) and the LVF ventures.

REFERENCES

Géron, A. *Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow*. Alta Books, 2019.

Gonçalves, A. P. G. *Previsão e-commerce: indicadores de desempenho por canal*. Ph.D. thesis, 2019.

Nabipour, M., Nayyeri, P., Jabani, H., Shahab, S., and Mosavi, A. Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis. *IEEE Access* vol. 8, pp. 150199–150212, 2020.

Shaytura, S. V., Kozhayev, Y. P., Ordov, K. V., Antonenkova, A. V., and Zhenova, N. A. Performance evaluation of the electronic commerce systems. *Revista Espacios*, 2017.

Sinaga, K. P. and Yang, M.-S. Unsupervised k-means clustering algorithm. *IEEE Access* vol. 8, pp. 80716–80727, 2020.

Srivastava, S., Gupta, M. R., and Frigyik, B. A. Bayesian quadratic discriminant analysis. *Journal of Machine Learning Research* 8 (6), 2007.

Taha, A. A. and Malebary, S. J. An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine. *IEEE Access* vol. 8, pp. 25579–25587, 2020.

Taleb, I., Dssouli, R., and Serhani, M. A. Big data pre-processing: A quality framework. In *2015 IEEE international congress on big data*. IEEE, pp. 191–198, 2015.

Triayudi, A., Sumiati, S., Nurhadiyan, T., and Rosalina, V. Data mining implementation to predict sales using time series method. *Proceeding of the Electrical Engineering Computer Science and Informatics* 7 (2): 1–6, 2020.

Zhang, X.-D. Machine learning. In *A Matrix Algebra Approach to Artificial Intelligence*. Springer, pp. 223–440, 2020.

Zunic, E., Korjenic, K., Hodzic, K., and Donko, D. Application of facebook's prophet algorithm for successful sales forecasting based on real-world data. *arXiv preprint arXiv:2005.07575*, 2020.