Machine Learning - ML
Masters in Data Science

Final project

Kick them off

Predicting whether or not it is advisable to buy a car using kick transaction data

Diana Rocío Galindo
Sebastian Pagglia

Kick them off

June 10th, 2022

Problem

Given a set of variables related to cars we aim to predict if the car purchased at the auction is a good or bad buy. In order to achieve this goal we will follow the suggested pipeline including: splitting data, performing preprocessing data techniques, univariate analysis, correlation with among the variables, feature selection, modeling and comparing models to establish the best solution for this problem.

We have chosen this problem because it will allow us to understand a real case scenario in which we could apply ML techniques. In this case, to understand kick car transactions and establish whether there is a good buy or not, to obtain metrics that allow avoiding costly expenses or high losses for vehicle resellers. We intend to follow a complete workflow to solve this kind of real-life problem from an available data set which accomplishes all the requirements established for the project.

1 Dataset

The original dataset contains 72.983 observations among 33 distinct variables explained as follows:

1. *IsBadBuy:* Identifies if the kicked vehicle was an avoidable purchase and it is the response variable.
2. *PurchDate:* The Date the vehicle was Purchased at Auction.
3. *Auction:* Auction provider at which the vehicle was purchased.
4. *VehYear:* The manufacturer's year of the vehicle.
5. *VehicleAge:* The Years elapsed since the manufacturer's year.
6. *Make:* Vehicle Manufacturer.
7. *Model:* Vehicle Model.
8. *Trim:* Vehicle Trim Level.
9. *SubModel:* Vehicle Submodel.

10. *Color:* Vehicle Color.
11. *Transmission:* Vehicles transmission type (Automatic, Manual).
12. *WheelTypeID:* The type id of the vehicle wheel.
13. *WheelType:* The vehicle wheel type description (Alloy, Covers).
14. *VehOdo:* The vehicles odometer reading.
15. *Nationality:* The Manufacturer's country.
16. *Size:* The size category of the vehicle (Compact, SUV, etc.).
17. *TopThreeAmericanName:* Identifies if the manufacturer is one of the top three American manufacturers.
18. *MMRAcquisitionAuctionAveragePrice:* Acquisition price for this vehicle in average condition at time of purchase.
19. *MMRAcquisitionAuctionCleanPrice:* Acquisition price for this vehicle in the above Average condition at time of purchase.
20. *MMRAcquisitionRetailAveragePrice:* Acquisition price for this vehicle in the retail market in average condition at time of purchase.
21. *MMRAcquisitonRetailCleanPrice:* Acquisition price for this vehicle in the retail market in above average condition at time of purchase.
22. *MMRCurrentAuctionAveragePrice:* Acquisition price for this vehicle in average condition as of current day.
23. *MMRCurrentAuctionCleanPrice:* Acquisition price for this vehicle in the above condition as of current day.
24. *MMRCurrentRetailAveragePrice:* Acquisition price for this vehicle in the retail market in average condition as of current day.
25. *MMRCurrentRetailCleanPrice:* Acquisition price for this vehicle in the retail market in above average condition as of current day.
26. *PRIMEUNIT:* Identifies if the vehicle would have a higher demand than a standard purchase.
27. *AUCGUART:* The level guarantee provided by auction for the vehicle (Green light - Guaranteed/arbitratable, Yellow Light - caution/issue, red light - sold as is)
28. *BYRNO:* Unique number assigned to the buyer that purchased the vehicle.
29. *VNZIP1:* Zipcode where the car was purchased.
30. *VNST:* State where the the car was purchased.
31. *VehBCost:* Acquisition cost paid for the vehicle at time of purchase.
32. *IsOnlineSale:* Identifies if the vehicle was originally purchased online
33. *WarrantyCost:* Warranty price (term=36month and millage=36K).

A complete output for raw data description is available at the Profile$_{K}ickInitial$ file attached to this report.

For our purpose, we considered 19 categorical and 14 numerical variables: *IsBadBuy* (target-boolean); *Auction* (3 levels); *Make* (33 levels); *Model* (1063 levels); *Trim* (134 levels); *SubModel* (863 levels); *Color (16 levels)*; *Transmission* (2 levels); *WheelTypeID (4 levels)*; *WheelType* (3 levels); *Nationality* (4 levels); *Size* (12 levels); *TopThreeAmericanName* (4 levels); *PRIMEUNIT* (2 levels); *AUCGUART (2 levels)*; *BYRNO* (74 levels); *VNZIP1*(153 levels);*VNST*(37 levels); *IsOnlineSale* (2 levels).

We realized that our dataset is highly unbalanced in some variables, but most importantly in our target variable, having 64.007 (87,7%) observations with value 0 and 8.976 with value 1 (12,3%).

2 Methodology

To carry out this project, we have followed the illustrated methodology (1): we started with the data cleaning and preprocessing to refine the input data, including an exploratory data analysis of the variables, missing values imputation, recategorization of variables to model the BadBuy prediction, and data splitting for model validation. Afterwards, the modeling step included trials with different algorithms and hyperparameters to get the best approximation to classify data. We approached this phase recursively until we reached the best model per algorithm based on the results obtained with training dataset. Once selected the best combination of parameters for an specific algorithm the final model was selected and we get the predicted values and the corresponding metrics. Finally we made a final analysis with the results.

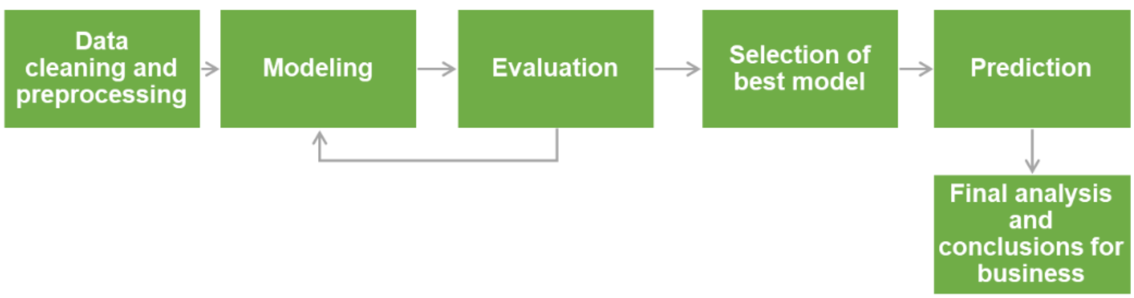


Figure 1: Methodology used for this project

3 Data Cleaning and preprocessing

The first data cleaning task we identified, was that some observations had a quote mark, so we replaced them for NAs. Moreover, we identified that some variables were not well categorized, therefore, we set them correctly (as numerical or categorical). The variable *PurchDate* was not correctly formated, then we looked for the correct timestamp and set it as date format. In addition, we decided to create another variable named *pMonth*, were we could describe the month of the purchase. At the same time, this variable was grouped by season of the year in a new one called *season*.

Once a first cleaning of the data was completed we took some decisions regarding which features do we want to keep and wich ones don't, and why.

Based on the Profile Report obtained, we dropped the following variables:

1. *PurchDate* and *pMonth* since variable *season* was created
2. VehYear because we have the same information in the variable *VehicleAge*
3. *Make* and *TopThreeAmericanName* which have a high correlation with *Nationality*.
4. *WheelTypeID* because it is describing an identifier of the variable *WheelType* which is conserved into the analysis
5. All *MMR* variables are related to prices and hence, high correlated with the variable *VehBCost* as depicted in the figure (2) of correlation below
6. The variable *Trim* corresponds to a version of a particular model with a particular configuration and has 134 categories which can produce noise into the model. Also, *model* has more than 1.000 categories, adding noise the model, so we decided to eliminate both and keep *SubModel* in the dataset
7. *PRIMEUNIT* and *AUCGUART* due to its high percentage of missing values. In this case, we had two options, to eliminate those observations or impute them. However, if we eliminate the observations our dataset would lose too many data and if we impute them the data would be synthetic data (artificially manufactured). Since those features won't provide us much information in these conditions, we preferred to eliminate them.
8. *VNZIP1* is highly correlated with *VNST* *BYRNO* is the identifier of the buyer and does not add value to the prediction analysis.

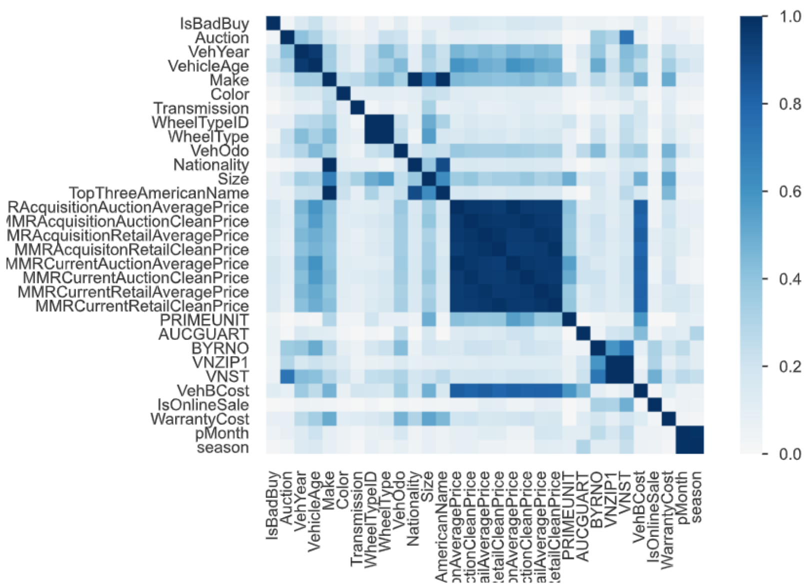


Figure 2: Correlation using Phik's method

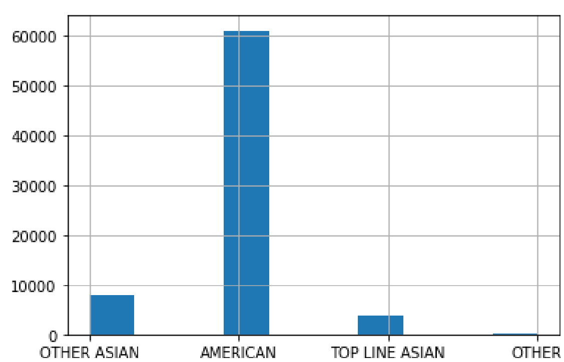
Continuing with the analysis, other preprocessing tasks were made such as: for the variable *Color*, we reassign the category "NOT AVAIL" to NA values; for the variable *Transmission* we unified the uppercase labeling.

We also processed some of the variables before to be included in the modeling step. This was the case for the variable *Submodel* which had 863 categories. The purpose was to extract the most general information of the inside the text. We obtained an initial list of submodels by removing numerical characters, punctuation marks and words of shorter length using the str function of python and regular expressions. After this cleaning, we reduced from 863 different categories, to 19. Finally, we decided to regroup those with less than 1000 observations and the individuals in the remaining categories into the category *OTHER*. Thus, the variable Submodel was incorporated into the model with 10 categories.

As next step, we transformed the variable *VehicleAge* into a categorical one with three groups: 0 to 3 years, 3 to 6 years and 6 to 9 years.

For the variable *Nationality* we unified the categories OTHER ASIAN and TOP LINE ASIAN into ASIAN category, taking into account the distribution of the data in this category and the geographic coincidence.

We reagrouped the variable *size* from 12 to 7 categories. In this case considered the sizes SMALL, MEDIUM and LARGE in a general way and the CROSSOVER category since it is a type of SUV was merged with this category. In the category OTHER were grouped the sizes SPECIALTY and SPORTS given the distribution of the data.



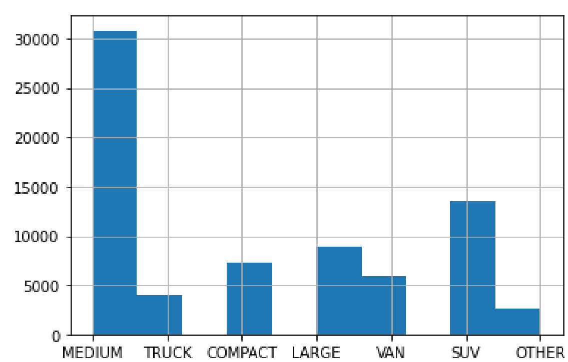


Figure 3: Histograms of data distribution for variables *Nationality* (left) and *size* (right) after preprocessing

We found five duplicated observations, which were removed.

3.1 Outliers

We explored the distribution of each numeric variable to check the presence of outlier data (see figure 4). As a rule of thumb we used the quartiles information to detect outlier data per variable. We set as a threshold, three times below and above the first and third quartile, respectively.

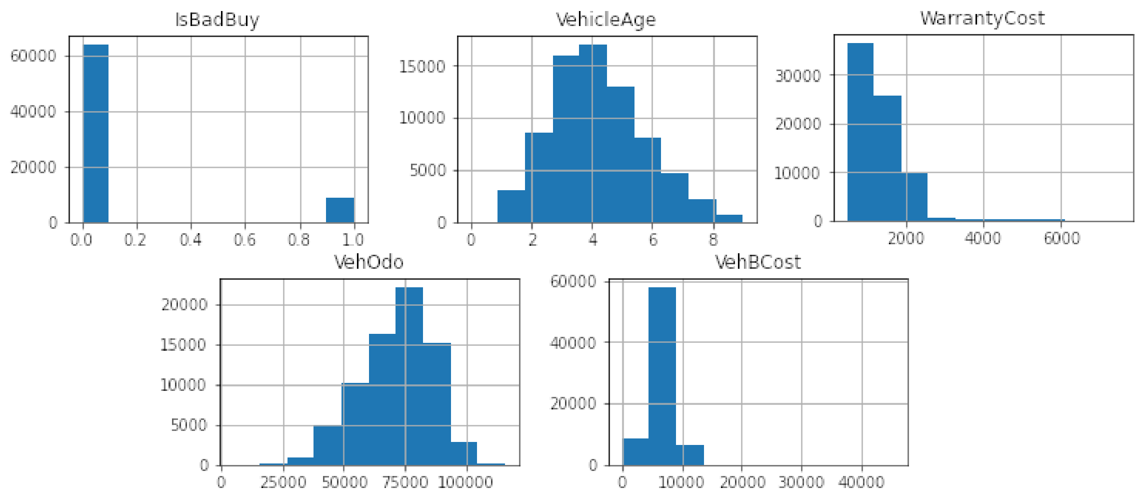


Figure 4: Histograms of numerical variables

The variable *VehBCost* has 13 high values considered as outliers according to the distribution of the data, however, we decided to keep them since all of them correspond to cars with high quality characteristics (short age, SPORT *submodel*) and it is expected a high cost compared to the others. All of these individuals, with one exception, belong to *BadBuy* category of the target variable. During this exploration we found a car with a price of one dollar that clearly is a wrong value. Then, we set it as NA observation.

The variable *WarrantyCost* presented 466 high values outliers being the maximum value 7498. We did not found evidence to consider them a high warranty cost value and it depends on the car being warranted. Therefore we did not delete them.

3.2 Missing values and data imputation

The variables *VehBCost, Size, Nationality, WheelType, Transmission, Color* present missing values. We imputed data for the numerical variables using Multivariate imputation by chained equations (MICE) imputation method implemented in python. The 3.297 missing observations of categorical variables were removed since corresponds to less than the 5% of the data.

We created new categorical variables from numeric ones to test the performance of each approach in the model selection stage.

3.3 Training, validation and test splitting

Once the cleaning and preprocessing is finished, we have our dataset prepared to start evaluating and improving models, in seek of the best possible results, selecting different hyperparameters and different machine learning methods.

In order to do so, first we must split our dataset. We decided to use 3-way split (train, validation and test) rather than Cross-validation because C-V is usually used in machine learning for improving model prediction when we dont have enough data to apply other more efficient methods like the one we selected. Since we have enough data, we can use the 3-way splitting. It is necessary to split it into 3 taking into account that we cannot use the train dataset to evaluate our results, because it might give artificially good performance, and we need a sample to test the final results where the data is not treated nor trained.

4 Modeling

In the modeling stage we considered four algorithms to predict the positive classification of a bad buy: decision tree, random forest, logistic regression and support vector machine classifiers. We considered the first two options for the clear interpretability of the parameters and computational processing time, the logistic regression for the nature of response variable and the SVM classifier to evaluate more complex models and to compare the result among them.

We performed experiments tuning the corresponding hyperparameters per algorithm pursuing a good performance and estimation. We took as a first glance reference the F1 score, but the evaluation of the best model per algorithm are explained in the corresponding section. In this case, a false positive corresponds to a prediction of a bad buy when the car is not broken. In pursue of the best model, we tried two different datasets, one considering the numerical variables as numerical; another one reclassifying them as categorical variables. In any case, the categorical variables were transformed using one hot encoding.

4.1 Decision tree classifier:

The first algorithm tested was a decision tree classifier. Scikit learn decision trees are not able to handle categorical attributes so we transform them to numerical using onehot-encoding. We know that this model in particular is fast to train but it can overfited easily, so we will check this point afterwards. To reduce the posibility of overfitting and to obtain the best combination of the parameters for this classifier, we use the tool "GridSearchCV" to test different parameters combination. The combination included test of gini and entropy criterion, max depth using values "None", 4, 6, 10, 20, considering simple approach of 4 variables and 20 being the half of the features considered; minimum samples split and minimum groups or min samples leaf values of 1, 2, 4, 9, 90 considering different numbers of individuals and testing maximum features with parameters sqrt (the squared root of the number total of features, log2 and none. This procedure was run out for both datasets considering numerical variables and categorical values and all the variables categorized.

The figure 5 shows the results for the dataset considering numeric variables and the figure 6 shows the results for the dataset considering just categorical values.

	criterion	max_depth	max_features	min_samples_leaf	min_samples_split	mean_test_f1_mac	mean_test_f1_class_0	mean_test_f1_class_1	mean_test_acc
53	gini	None	None	1	9	0.527422	0.917264	0.137580	0.849018
437	entropy	None	None	4	4	0.526100	0.919240	0.132960	0.852245
438	entropy	None	None	4	9	0.524922	0.919382	0.130462	0.852447
436	entropy	None	None	4	2	0.524903	0.919314	0.130492	0.852335
433	entropy	None	None	2	9	0.523046	0.916002	0.130090	0.846800

Figure 5: Decision tree best parameter search for numeric and categorical data

	criterion	max_depth	max_features	min_samples_leaf	min_samples_split	mean_test_f1_mac	mean_test_f1_class_0	mean_test_f1_class_1	mean_test_acc
401	entropy	None	log2	1	2	0.523729	0.911038	0.136420	0.838711
51	gini	None	None	1	2	0.523085	0.903607	0.142564	0.826701
376	entropy	None	sqrt	1	2	0.522925	0.908952	0.136897	0.835283
1	gini	None	sqrt	1	2	0.522818	0.909882	0.135754	0.836784
26	gini	None	log2	1	2	0.520712	0.910184	0.131239	0.837210

Figure 6: Decision tree best parameter search for categorical data

From these experiments the best results were obtained from the dataset with numeric and categorical data decision tree with a mean test accuracy of 84,90%, a mean_test_f1_class_0 of 91.73%, a mean_test_f1_class_1 of 13.76% and a mean_test_f1_mac of 52.74%. This decision tree considers *min_samples_unit*: 1 BadBuy as the minimum number of sample in an internal node, a minimum number of 9 BadBuy sales of samples used in a node. The decision tree classifier shows a poor capability to predict NotBadBuy but this is expected from the unbalanced data to NotBadBuy rows compared with the BadBuy ones.

4.2 Random Forest classifier:

Based on the results of the initial Decision tree, we evaluated the random forest algorithm, taking into account its way of analysis and calculating its variance average, what makes it less probable to overfit.

The first important thing to mention when performing Random Forest classifier method is to apply Out of Bag (OOB) error, because it is very useful when it comes to model selection and avoiding cross validation (because its highly cost). In both dataset cases we performed an initial regression applying OOB as reference.

As well the GridSearchCV function to test different parameters combination. For this method, we also incorporate testing the number of trees to be included in hyperparameter tuning. In summary, for this algorithm we tried the following parameters ntrees: 100,200 and 500, max depth: None, 4, 6, 10 and 20, min samples split of 1, 2, 4 and 9; min samples leaf of 1, 2, 4, 9 and None, balanced and balanced_subsample categories.

The figure 7 shows the results for the GridSearchCV dataset considering numeric variables and the figure 8 shows the results for the dataset considering just categorical values.

	max_depth	min_samples_leaf	min_samples_split	mean_test_f1_mac	mean_test_f1_class_0	mean_test_f1_class_1	mean_test_acc
242	None	9	9	0.568314	0.894725	0.241904	0.815138
128	None	9	2	0.568248	0.893954	0.242542	0.813973
236	None	9	2	0.567690	0.894274	0.241107	0.814421
134	None	9	9	0.567536	0.894446	0.240626	0.814668
240	None	9	9	0.567307	0.892455	0.242159	0.811665

Figure 7: Random forest best parameter search for numeric and categorical data

	max_depth	min_samples_leaf	min_samples_split	mean_test_f1_mac	mean_test_f1_class_0	mean_test_f1_class_1	mean_test_acc
705	20	4	9	0.557143	0.903465	0.210822	0.828000
271	None	4	4	0.556272	0.913424	0.199120	0.843753
463	20	4	4	0.555625	0.904332	0.206918	0.829277
275	None	4	9	0.555517	0.912995	0.198039	0.843036
267	None	4	2	0.555483	0.912567	0.198398	0.842341

Figure 8: Random forest best parameter search for categorical data

For the categorical dataset, the best model used 100 trees whereas for the numerical it used 500. The results for the dataset considering numerical and categorical data presented a slightly better performance regarding the classification of IsBadBuy, in terms of F1class1 and F1MacroAvg. This random forest classifier has a mean test accuracy of 81.51%, a mean_test_f1_class_0 of 89.47%, a mean_test_f1_class_1 of 24.19% and a mean_test_f1_mac of 56.83%.

The random forest classifier showed a better capability to predict NotBadBuy increasing around around a 7% compared with the decision tree algorithm.

4.3 Logistic regression:

We considered the logistic regression algorithm and evaluated for BadBuy prediction. In this case, we perfomed a first exploratory logistic regression including all the initial as illustrated in the figure 9, the results were extremely poor for both cases, the dataset considering numeric and categorical variables (left side) and the one considering only categorical values (right side).

```
Optimization terminated successfully.
         Current function value: 0.301238
         Iterations 7
                         Results: Logit
==================================================================
Model:              Logit            Pseudo R-squared: 0.054
Dependent Variable: IsBadBuy         AIC:              26979.2895
Date:               2022-06-10 15:18 BIC:              27379.7709
No. Observations:   44628            Log-Likelihood:   -13444.
Df Model:           45               LL-Null:          -14218.
Df Residuals:       44582            LLR p-value:      2.1900e-295
Converged:          1.0000           Scale:            1.0000
No. Iterations:     7.0000
------------------------------------------------------------------
                    Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
------------------------------------------------------------------
VehicleAge           0.4735  0.0186  25.4777 0.0000  0.4371  0.5099
VehOdo               0.1623  0.0191   8.4860 0.0000  0.1248  0.1998
VehBCost            -0.0803  0.0168  -4.7760 0.0000 -0.1133 -0.0474
WarrantyCost         0.0361  0.0214   1.6885 0.0913 -0.0058  0.0779
Auction_MANHEIM      0.3099  0.0445   6.9709 0.0000  0.2228  0.3970
Auction_OTHER       -0.0287  0.0558  -0.5147 0.6068 -0.1381  0.0807
SubModel_COUPE      -2.2446  0.1511 -14.8529 0.0000 -2.5408 -1.9484
SubModel_CUV        -2.0161  0.1978 -10.1905 0.0000 -2.4038 -1.6283
SubModel_MINIVAN    -2.5488  0.2816  -9.0508 0.0000 -3.1007 -1.9968
SubModel_OTHER      -2.1374  0.1757 -12.1654 0.0000 -2.4817 -1.7930
SubModel_PASSENGER  -2.4904  0.2849  -8.7410 0.0000 -3.0488 -1.9320
SubModel_SEDAN      -2.2196  0.1336 -16.6099 0.0000 -2.4815 -1.9577
SubModel_SPORT      -2.0761  0.1874 -11.0802 0.0000 -2.4433 -1.7089
SubModel_SUV        -2.1253  0.1705 -12.4621 0.0000 -2.4596 -1.7911
SubModel_WAGON      -2.0923  0.1539 -13.5913 0.0000 -2.3941 -1.7906
Color_BLACK          0.1560  0.1230   1.2686 0.2046 -0.0850  0.3970
Color_BLUE           0.0016  0.1206   0.0137 0.9891 -0.2347  0.2380
Color_BROWN          0.2942  0.2182   1.3479 0.1777 -0.1336  0.7219
Color_GOLD           0.1202  0.1254   0.9586 0.3378 -0.1256  0.3660
Color_GREEN         -0.0197  0.1347  -0.1464 0.8836 -0.2838  0.2444
Color_GREY           0.0623  0.1225   0.5087 0.6109 -0.1778  0.3024
Color_MAROON         0.1477  0.1441   1.0253 0.3052 -0.1347  0.4301
Color_ORANGE        -0.1443  0.2941  -0.4905 0.6238 -0.7208  0.4322
Color_OTHER          0.1310  0.3708   0.3532 0.7239 -0.5957  0.8577
Color_PURPLE         0.3302  0.2265   1.4580 0.1448 -0.1137  0.7742
Color_RED            0.1430  0.1242   1.1514 0.2496 -0.1005  0.3865
Color_SILVER         0.1268  0.1169   1.0854 0.2777 -0.1022  0.3559
Color_WHITE          0.1018  0.1182   0.8609 0.3893 -0.1299  0.3334
Color_YELLOW         0.0323  0.2774   0.1163 0.9074 -0.5115  0.5760
Transmission_MANUAL  0.0218  0.0823   0.2646 0.7913 -0.1395  0.1830
WheelType_Covers    -0.0707  0.0376  -1.8780 0.0604 -0.1445  0.0031
WheelType_Special    0.1459  0.1433   1.0182 0.3086 -0.1349  0.4266
Nationality_ASIAN    0.0616  0.0501   1.2306 0.2185 -0.0365  0.1598
Nationality_OTHER   -0.1962  0.3323  -0.5905 0.5549 -0.8476  0.4551
Size_LARGE          -0.5300  0.0815  -6.4994 0.0000 -0.6898 -0.3701
Size_MEDIUM         -0.1608  0.0564  -2.8497 0.0044 -0.2714 -0.0502
Size_OTHER          -0.2483  0.1065  -2.3324 0.0197 -0.4570 -0.0397
Size_SUV            -0.3767  0.1206  -3.1223 0.0018 -0.6131 -0.1402
Size_TRUCK          -2.6905  0.1403 -19.1754 0.0000 -2.9655 -2.4155
Size_VAN            -0.0136  0.2420  -0.0561 0.9552 -0.4879  0.4607
VNST_EAST           -0.2624  0.0370  -7.0834 0.0000 -0.3349 -0.1898
VNST_WEST           -0.1283  0.0459  -2.7945 0.0052 -0.2183 -0.0383
IsOnlineSale_yes    -0.1819  0.1161  -1.5668 0.1172 -0.4095  0.0456
season_spring       -0.0435  0.0458  -0.9501 0.3421 -0.1334  0.0463
season_summer       -0.0614  0.0460  -1.3332 0.1825 -0.1516  0.0289
season_winter       -0.0104  0.0458  -0.2272 0.8203 -0.1002  0.0793
==================================================================
```

```
Optimization terminated successfully.
         Current function value: 0.303174
         Iterations 7
                         Results: Logit
===========================================================================
Model:              Logit            Pseudo R-squared:    0.048
Dependent Variable: IsBadBuy         AIC:                 27166.0897
Date:               2022-06-10 15:18 BIC:                 27627.5139
No. Observations:   44628            Log-Likelihood:      -13530.
Df Model:           52               LL-Null:             -14218.
Df Residuals:       44575            LLR p-value:         7.6129e-254
Converged:          1.0000           Scale:               1.0000
No. Iterations:     7.0000
---------------------------------------------------------------------------
                             Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
---------------------------------------------------------------------------
Auction_MANHEIM               0.3174  0.0444   7.1506 0.0000  0.2304  0.4044
Auction_OTHER                -0.0482  0.0559  -0.8617 0.3889 -0.1577  0.0614
SubModel_COUPE               -2.8069  0.1628 -17.2392 0.0000 -3.1260 -2.4877
SubModel_CUV                 -2.6665  0.2040 -13.0730 0.0000 -3.0663 -2.2667
SubModel_MINIVAN             -3.1773  0.2843 -11.1753 0.0000 -3.7346 -2.6201
SubModel_OTHER               -2.7049  0.1844 -14.6670 0.0000 -3.0663 -2.3434
SubModel_PASSENGER           -3.0650  0.2870 -10.6805 0.0000 -3.6275 -2.5026
SubModel_SEDAN               -2.8190  0.1464 -19.2577 0.0000 -3.1059 -2.5321
SubModel_SPORT               -2.6948  0.1944 -13.8631 0.0000 -3.0758 -2.3138
SubModel_SUV                 -2.7334  0.1784 -15.3212 0.0000 -3.0830 -2.3837
SubModel_WAGON               -2.7234  0.1620 -16.8124 0.0000 -3.0409 -2.4059
Color_BLACK                   0.1239  0.1225   1.0112 0.3119 -0.1162  0.3640
Color_BLUE                   -0.0293  0.1202  -0.2438 0.8074 -0.2649  0.2063
Color_BROWN                   0.2774  0.2177   1.2744 0.2025 -0.1492  0.7041
Color_GOLD                    0.1181  0.1250   0.9449 0.3447 -0.1269  0.3631
Color_GREEN                  -0.0265  0.1344  -0.1972 0.8437 -0.2898  0.2369
Color_GREY                    0.0299  0.1221   0.2452 0.8063 -0.2093  0.2692
Color_MAROON                  0.1401  0.1436   0.9757 0.3292 -0.1414  0.4217
Color_ORANGE                 -0.2052  0.2934  -0.6993 0.4844 -0.7803  0.3699
Color_OTHER                   0.1052  0.3707   0.2839 0.7765 -0.6213  0.8317
Color_PURPLE                  0.3224  0.2256   1.4290 0.1530 -0.1198  0.7646
Color_RED                     0.1161  0.1238   0.9378 0.3484 -0.1266  0.3587
Color_SILVER                  0.1044  0.1165   0.8964 0.3700 -0.1239  0.3327
Color_WHITE                   0.0865  0.1178   0.7338 0.4631 -0.1445  0.3174
Color_YELLOW                  0.0665  0.2767   0.2405 0.8099 -0.4757  0.6088
Transmission_MANUAL           0.0485  0.0821   0.5911 0.5545 -0.1124  0.2095
WheelType_Covers             -0.1158  0.0375  -3.0896 0.0020 -0.1893 -0.0423
WheelType_Special             0.1410  0.1428   0.9878 0.3233 -0.1388  0.4209
Nationality_ASIAN             0.0542  0.0539   1.0044 0.3152 -0.0515  0.1598
Nationality_OTHER            -0.2665  0.3314  -0.8040 0.4214 -0.9161  0.3831
Size_LARGE                   -0.5461  0.0867  -6.3013 0.0000 -0.7159 -0.3762
Size_MEDIUM                  -0.1676  0.0569  -2.9483 0.0032 -0.2791 -0.0562
Size_OTHER                   -0.2393  0.1077  -2.2213 0.0263 -0.4504 -0.0281
Size_SUV                     -0.3527  0.1234  -2.8590 0.0042 -0.5945 -0.1109
Size_TRUCK                   -3.2636  0.1550 -21.0606 0.0000 -3.5673 -2.9599
Size_VAN                      0.0153  0.2431   0.0628 0.9499 -0.4612  0.4918
VNST_EAST                    -0.2332  0.0369  -6.3228 0.0000 -0.3054 -0.1609
VNST_WEST                    -0.1204  0.0458  -2.6271 0.0086 -0.2102 -0.0306
IsOnlineSale_yes             -0.1745  0.1160  -1.5051 0.1323 -0.4018  0.0527
season_spring                -0.0309  0.0457  -0.6760 0.4991 -0.1204  0.0587
season_summer                -0.0462  0.0459  -1.0061 0.3144 -0.1362  0.0438
season_winter                 0.0063  0.0456   0.1371 0.8909 -0.0832  0.0957
age_cat_3to6_years            0.6861  0.0439  15.6174 0.0000  0.6000  0.7722
age_cat_6to9_years            1.3274  0.0585  22.6864 0.0000  1.2127  1.4421
VehOdo_cat_61815-73322        0.1441  0.0537   2.6833 0.0073  0.0389  0.2494
VehOdo_cat_73322-82383        0.4003  0.0533   7.5157 0.0000  0.2959  0.5047
VehOdo_cat_>82383             0.4432  0.0550   8.0600 0.0000  0.3355  0.5510
VehBCost_cat_5440-6710       -0.1074  0.0442  -2.4291 0.0151 -0.1941 -0.0207
VehBCost_cat_6710-7900       -0.1685  0.0461  -3.6555 0.0003 -0.2589 -0.0782
VehBCost_cat_>7900           -0.2515  0.0471  -5.3390 0.0000 -0.3438 -0.1591
WarrantyCost_cat_1155-1623    0.0052  0.0608   0.0855 0.9319 -0.1140  0.1244
WarrantyCost_cat_837-1155     0.0145  0.0547   0.2649 0.7911 -0.0928  0.1218
WarrantyCost_cat_>1623        0.0844  0.0742   1.1377 0.2552 -0.0610  0.2298
===========================================================================
```

Figure 9:

Logistic regression

The figure 10 exposes the ROC and AUC for both models, for dataset with numeric variables (left) and only categorical values (right). Taking as a reference the pseudo-R metric and the confusion matrix this algorithm could not predict any case of positive BadBuy.

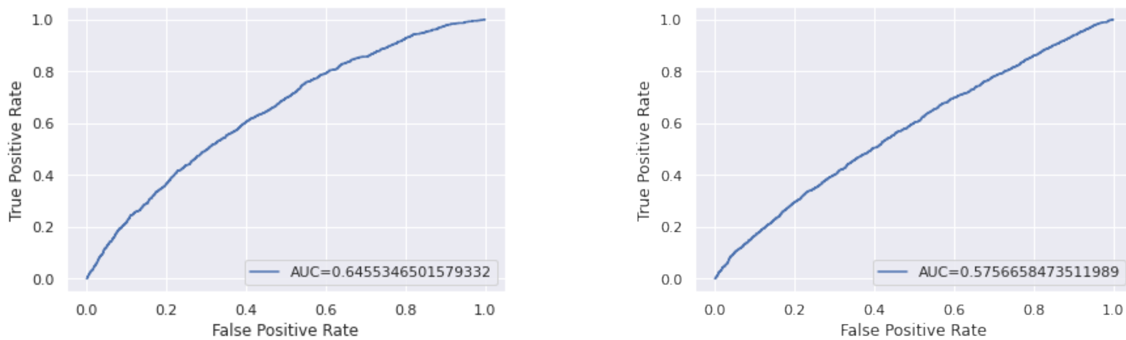


Figure 10:

ROC and AUC for logistic regression models

From the LogisticRegressionCV testing for best parameter, in both cases was obtained with a C= 0.001 (see figure 11)

```
print("Tuned Hyperparameters :", clf.best_params_)
print("Accuracy :",clf.best_score_)

Tuned Hyperparameters : {'C': 0.001, 'penalty': 'l2'}
Accuracy : 0.9029308966307262
```

```
print("Tuned Hyperparameters :", clf.best_params_)
print("Accuracy :",clf.best_score_)

Tuned Hyperparameters : {'C': 0.001, 'penalty': 'l2'}
Accuracy : 0.9029308966307262
```

Figure 11:

ROC and AUC for logistic regression models

4.4 Support Vector Machine (SVM):

The last algorithm tried was the SVM. This algorithm considers two parameters: C and gamma and kernel function definition. In this case we firstly performed analysis using a linear and a polynomic kernel of grade 2. There were no significant differences among the error results improvements obtained for the training and validation dataset (see figure 12). For this reason we consider the linear kernel, maintaining the simplicity of the calculations.

```
print("Test error: {}".format(1-linear_kick.score(X_test, y_test)))
print("Training error: {}".format(1-linear_kick.score(X_train, y_train)))

Test error: 0.09392700939270093
Training error: 0.09706910459801021
```

```
print("Training error: {}".format(1-poly_svm.score(X_train, y_train)))
print("Test error: {}".format(1-poly_svm.score(X_test, y_test)))

Training error: 0.09706910459801021
Test error: 0.09392700939270093
```

Figure 12: Linear and polynomic kernel function error for train and validation test

As well, we used the GridSearchCV for gamma and C parameters dataset considering numeric variables and the 13 shows the results for the dataset considering just categorical values.

	param_C	param_gamma	mean_test_score
0	0.125	0.125	0.902931
1	0.125	1	0.902931
2	0.125	10	0.902931
3	0.125	20	0.902931
4	1	0.125	0.902931

```
print(clf.best_params_)
{'C': 0.125, 'gamma': 0.125}
```

	param_C	param_gamma	mean_test_score
0	0.125	0.125	0.902931
1	0.125	1	0.902931
2	0.125	10	0.902931
3	0.125	20	0.902931
4	1	0.125	0.902931

```
print(clf.best_params_)
{'C': 0.125, 'gamma': 0.125}
```

Figure 13: SVM best parameter search for dataset considering numeric and categorical variables and just categorical data (right)

5 Evaluation and selection of best model

Once all the experiments were completed, we ran out the best option per algorithm with the validation test. The table below 14 present the summary of algorithms and parameters selected. In order to test and evaluate the best results, we selected the F1 macro average percentage as a guideline.

As we can observe from the table above, and the work presented on the previous chapters, the algorithm with the best performance is Random Forest, including numerical and categorical variables. The selected parameters are:

1. number of estimators = 500
2. criterion = gini
3. maximum depth = None
4. minimum samples split = 9
5. minimum samples leaf = 9
6. maximum features = None class weight = balanced_subsample

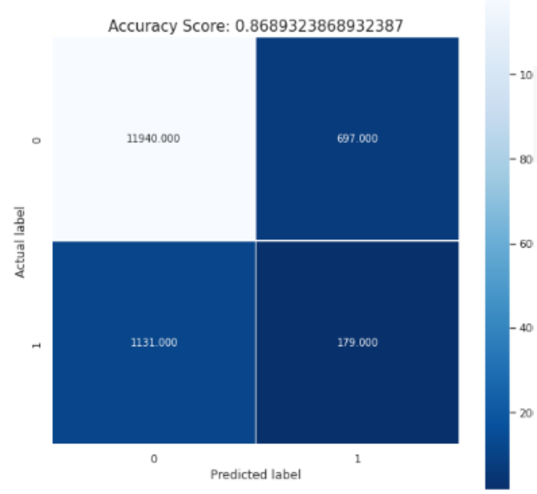
At this point, we already obtained the best model with the best parameters and algorithm. The next and final step is to test it with the test sample dataset that so far it was not modified or assessed at all.

The figure 15 present the final results and scores of the prediction using the test partition data and the corresponding metrics. As we can infer from the plot, the unbalanced characteristics

	Dataset including numeric vars	Dataset categorical vars
DT	n_estimators = 100 criterion = 'gini' max_depth=None min_samples_split = 9 min_samples_leaf = 2 max_features = None	n_estimators = 100 criterion = 'gini' max_depth=None min_samples_split = 2 min_samples_leaf = 1 max_features = 'log2'
Random Forest	n_estimators = 500 criterion = 'gini' max_depth=None min_samples_split = 9 min_samples_leaf = 9 max_features = None class_weight = 'balanced_subsample'	n_estimators = 100 criterion = 'gini' max_depth=20 min_samples_split = 9 min_samples_leaf = 4 max_features = None
LR	penalty = 'l2' C = 0.001 solver = 'lbfgs' max_iter = 1000	penalty = 'l2' C = 0.001 solver = 'lbfgs' max_iter = 1000
SVM	C=1 gamma=32 kernel='linear'	C= 0.000125 gamma= 0.000125 kernel='linear' class_weight='balanced'

Figure 14: Random forest best parameter search for categorical data

of the dataset, makes our final model better at predicting as NotBadBuy than IsBadBuy, as the F1 suggests with their percentage.




```
y_final_pred = model_RF_num.predict(X_test)

print(classification_report(y_test,y_final_pred))

              precision    recall  f1-score   support

           0       0.91      0.94      0.93     12637
           1       0.20      0.14      0.16      1310

    accuracy                           0.87     13947
   macro avg       0.56      0.54      0.55     13947
weighted avg       0.85      0.87      0.86     13947

score = model_RF_num.score(X_test, y_test)
print(score)

0.8689323868932387
```

Figure 15: Best model confusion matrix and scores

6 Final analysis and conclusions

Once we obtained our final model, we can reach now to the final conclusions of the project. We went through a lot of work during the preprocessing stage, as it usually happens in real-life and business companies.

The main obstacle to the exploration of different algorithms was the computing time in the selection of the best parameters for each case. The lightest as expected was the decision tree and the heaviest was the support vector machine that took, in some cases, more than 6 hours of processing in the cloud using the google colab tool.

In general, the best results were obtained from decision trees and random forests as discussed in the previous sections.

In the particular case of logistic regression, the results did not present a good fit of the models. However, in this case, in order to improve the performance of this algorithm, it is clear that a greater number of experiments are required considering the exclusion of some variables or the transformation of numerical variables that allow to improve the results.

The objective of the project was focused on carrying out experiments with the parameters of the different algorithms proposed to obtain a model with good prediction capacity. In this case, the algorithm that presented the best performance was the Random Forest.However, none of the algorithms evaluated satisfactorily predicted the IsABadBuy class.

This can be due to the highly unbalanced nature of the data. This implies that to improve predictions, in a future work, incorporate different methods that allow to solve this situation. This could mean the use of undersampling or oversampling techniques.

Another option could be to incorporate into the processing of the proposed the reduction of their dimensionality. This conclusion can be obtained from the poor results obtained in logistic regression.