Targeted Marketing with Machine Learning

Improve marketing campaign profit using Data Science techniques

Rodolfo Saldanha
May 14 · 8 min read ★

. . .

Content

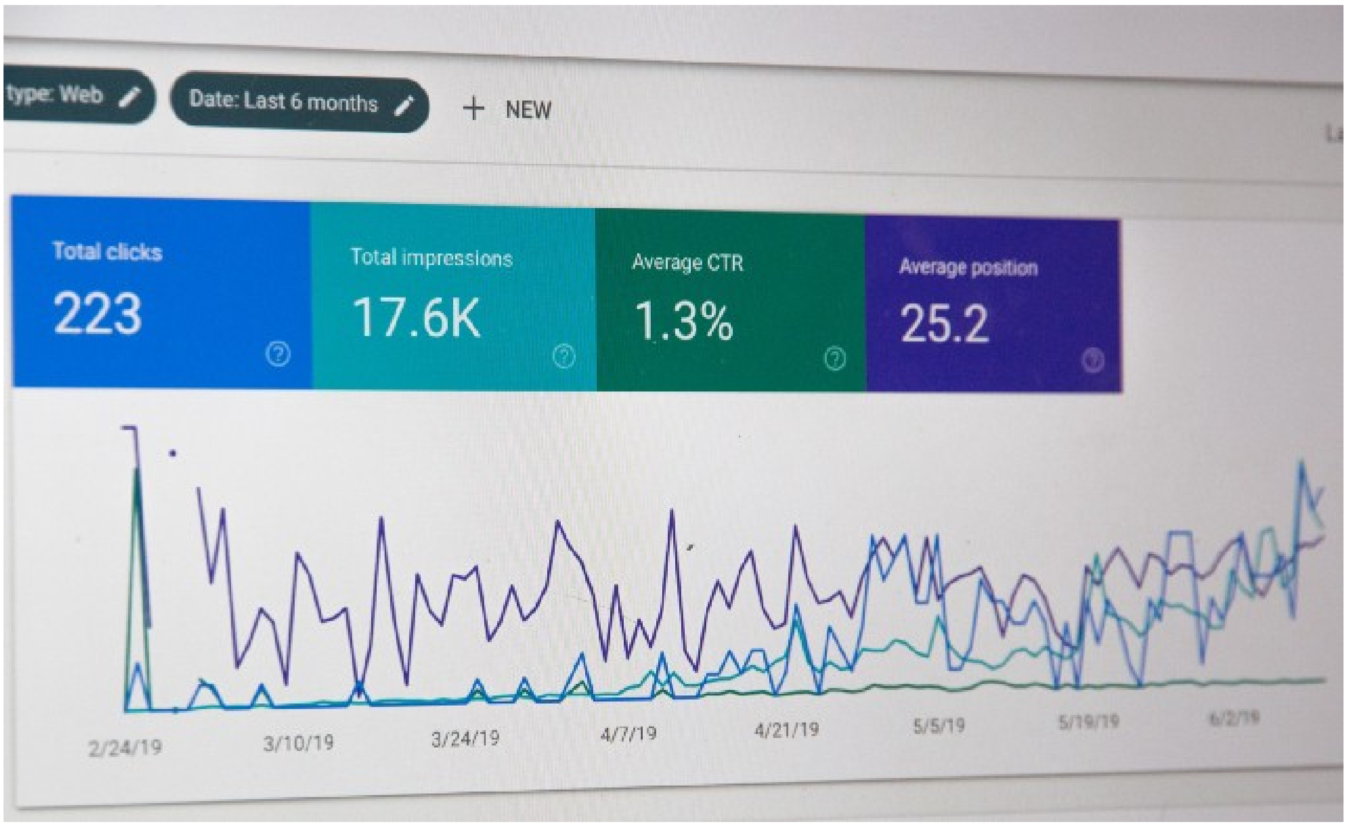
- Context

- **Exploratory Data Analysis (EDA)**
- **Outlier Detection**
- **Feature Engineering**
- **Feature Selection**
- **Data Modeling**
- **Conclusion**

. . .

Context

Marketing is one of the most important things a business can do. Not only does marketing build brand awareness, but it can also increase sales, grow businesses, and engage customers. On the other hand, marketing is also a generous source of expenses, and the investment in this segment should be well thought. Improving the profit margin is a constant concern to most of the companies, but what they are not aware of is that investing in the Data Science sector is a great way of doing that.



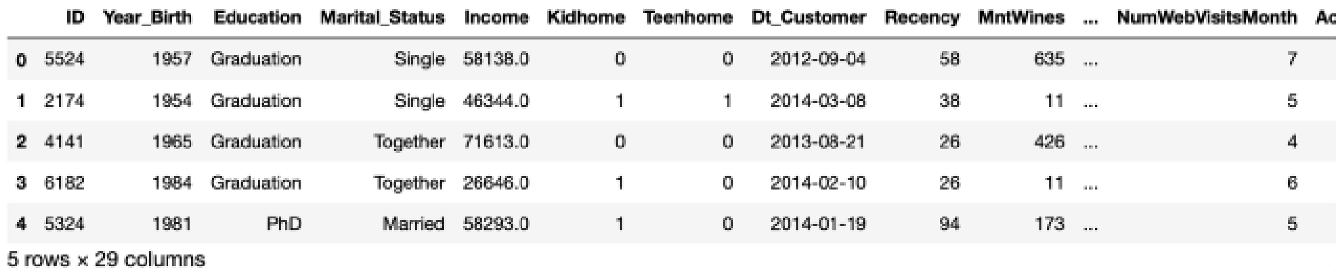
Mass marketing tends to be expensive and not so efficient because it normally does not consider the consumer profile, and the cost of the contact goes to waste, thus the profit decreases. As a solution for this issue, we have targeted marketing, which tends to be more effective. Targeted marketing typically provides specific information and incentives to people, giving them a reason to choose that company over competitors in the same industry. This approach identifies an audience likely to buy services or products and promotes those services or products to that audience. Once these key groups are recognized, companies develop marketing campaigns and specific products for those preferred market segments. When marketing is relevant, people are more likely to spend money on that service or product, and this whole process can be achieved through Machine Learning techniques. In this article, we will analyze the data of a fictitious company to select approachable customers and increase the profit of a marketing campaign. Any questions related to the code, you can find my work here.

. . .

Exploratory Data Analysis (EDA)

The dataset contains 29 attributes from customers related to their personal information, amount spent in each segment, participation in previous marketing campaigns, and whether the client accepted or not this campaign (the target that we want to predict).

```
data.head()
```

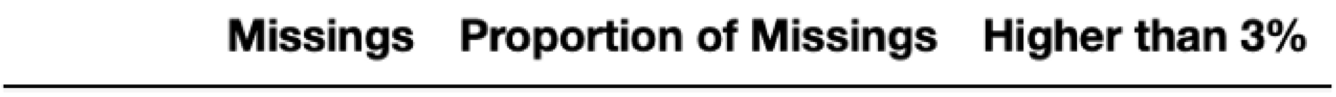


	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...	NumWebVisitsMonth	Ac
0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-04	58	635	...	7	
1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-08	38	11	...	5	
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21	26	426	...	4	
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-10	26	11	...	6	
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19	94	173	...	5	

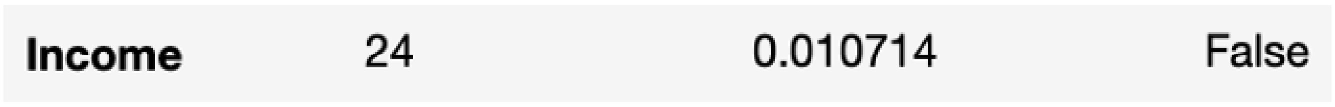
5 rows × 29 columns

Figure 1 — First rows of the dataset

The number of missing values in the dataset is very low, which is great.



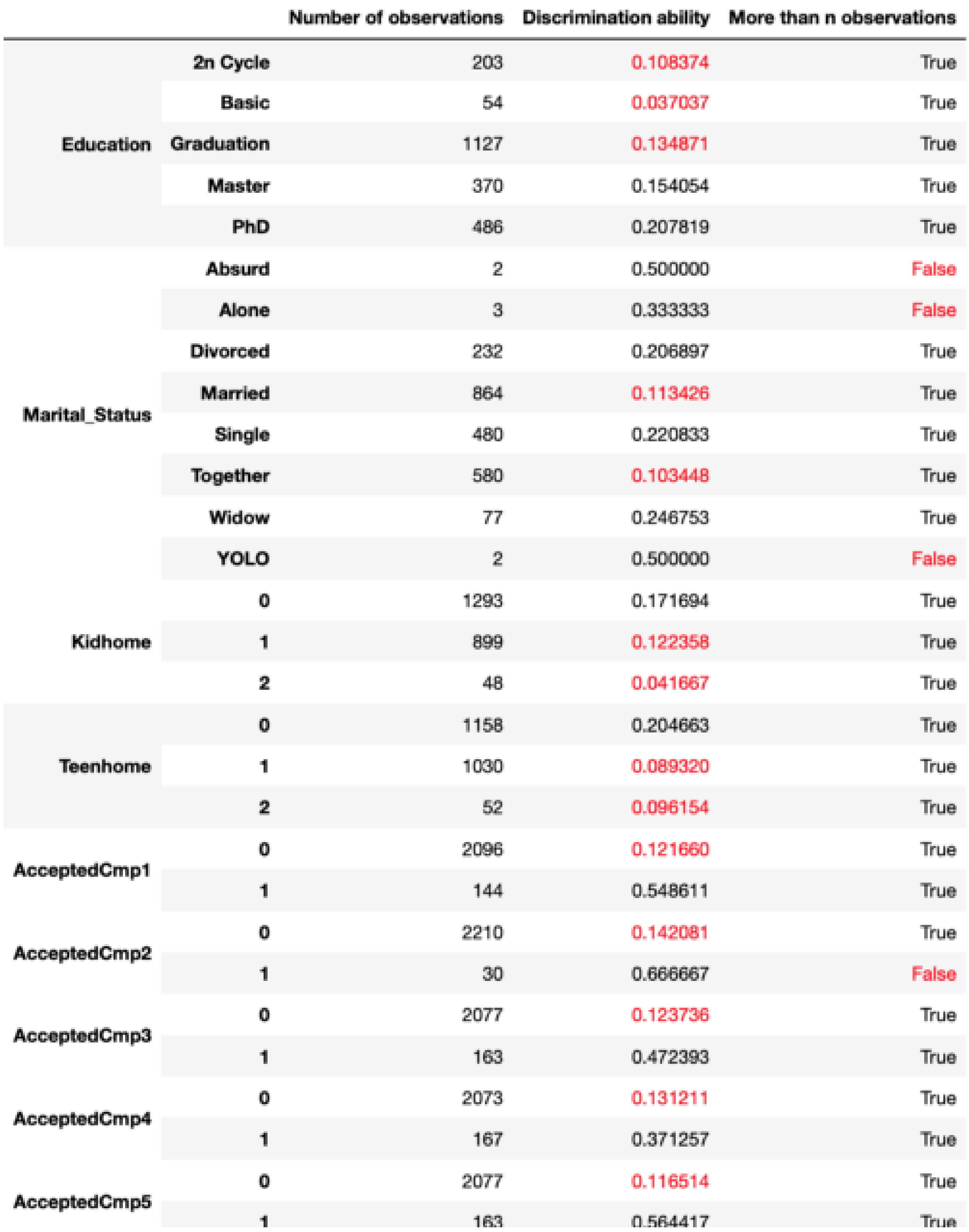
	Missings	Proportion of Missings	Higher than 3%



Income	24	0.010714	False

Figure 2 — Number of missing values

It is possible to see that we have some categorical values, so let's explore these attributes and check if they have meaningful information.



		Number of observations	Discrimination ability	More than n observations
Education	**2n Cycle**	203	0.108374	True
	Basic	54	0.037037	True
	Graduation	1127	0.134871	True
	Master	370	0.154054	True
	PhD	486	0.207819	True
Marital_Status	**Absurd**	2	0.500000	False
	Alone	3	0.333333	False
	Divorced	232	0.206897	True
	Married	864	0.113426	True
	Single	480	0.220833	True
	Together	580	0.103448	True
	Widow	77	0.246753	True
	YOLO	2	0.500000	False
Kidhome	**0**	1293	0.171694	True
	1	899	0.122358	True
	2	48	0.041667	True
Teenhome	**0**	1158	0.204663	True
	1	1030	0.089320	True
	2	52	0.096154	True
AcceptedCmp1	**0**	2096	0.121660	True
	1	144	0.548611	True
AcceptedCmp2	**0**	2210	0.142081	True
	1	30	0.666667	False
AcceptedCmp3	**0**	2077	0.123736	True
	1	163	0.472393	True
AcceptedCmp4	**0**	2073	0.131211	True
	1	167	0.371257	True
AcceptedCmp5	**0**	2077	0.116514	True
	1	163	0.564417	True

Complain	0	2219	0.149166	True
	1	21	0.142857	False

Figure 3 — Table of categories' discrimination ability

The discrimination ability stands for how well the category correlates to the target, and it is also important to check if the class has a significant number of data points. We arbitrarily set the discrimination ability and the *n* amount of observations as 15% and 40, respectively.

Some of the categories with low discrimination ability or not a significant number of observations will be clustered together to facilitate the model learn because they often bias the learning process.

Generally speaking, no category stands out having a high percentage of discrimination ability and a significant number of observations.

We create a new category *NumberOff*, which represents the number of offsprings to see if it has a better discrimination ability, but this is not the case.

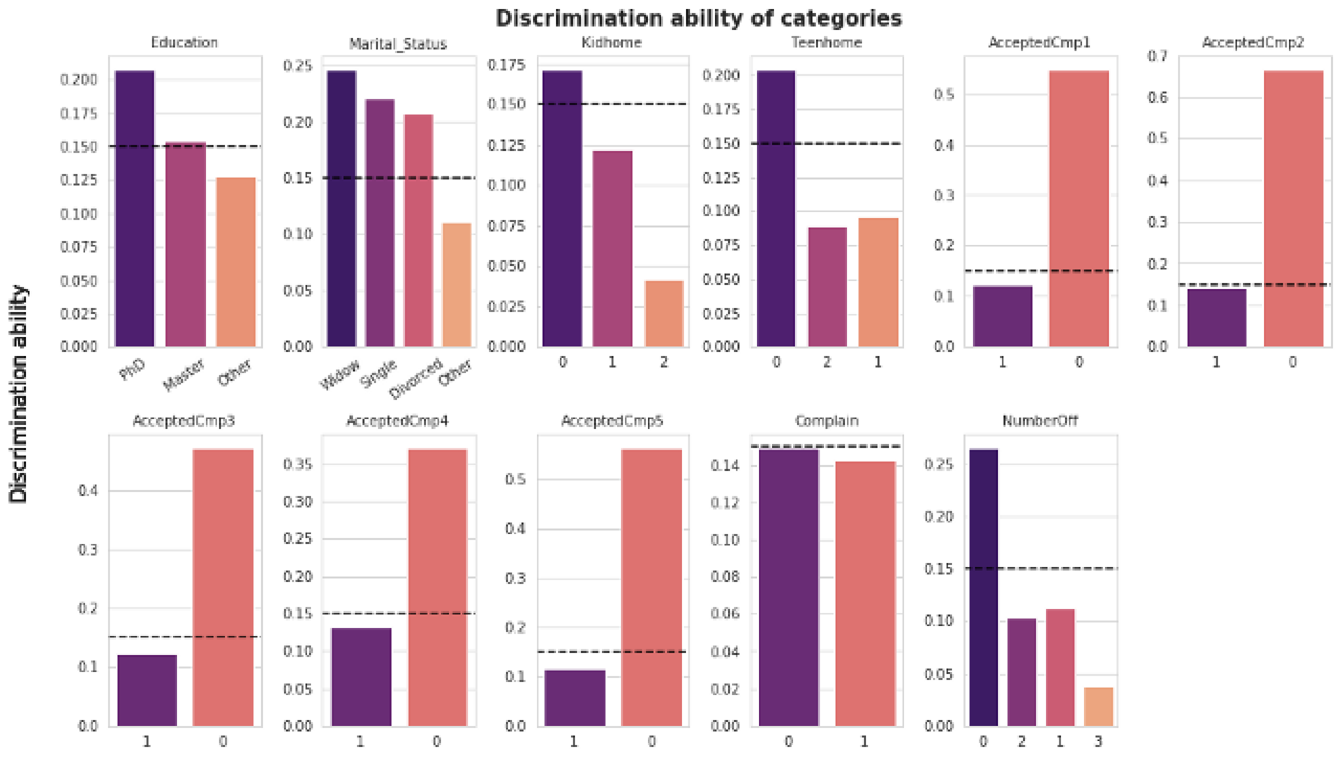
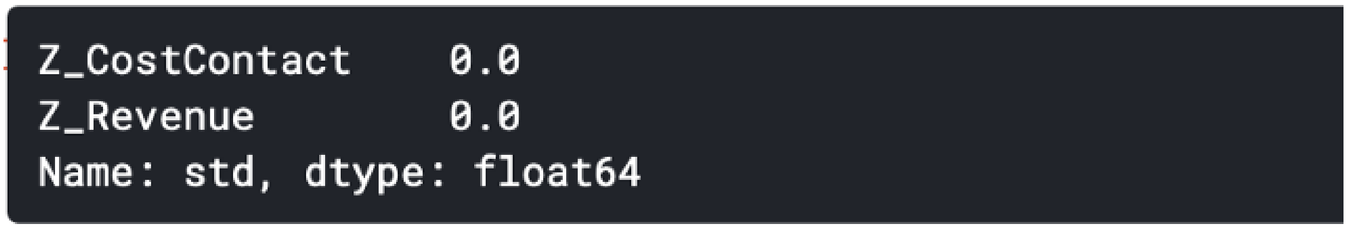


Figure 4 — Discrimination ability of categories

Now is the time to have a better understanding of the discreet attributes. We are going to check for constant variables by measuring the standard deviation, and also verify inconsistent values.

```
std = data[feat_n].describe().iloc[2,:]
const_lab = [std[std<0.05].index[0], std[std<0.05].index[1]]
std[std<0.05]
```

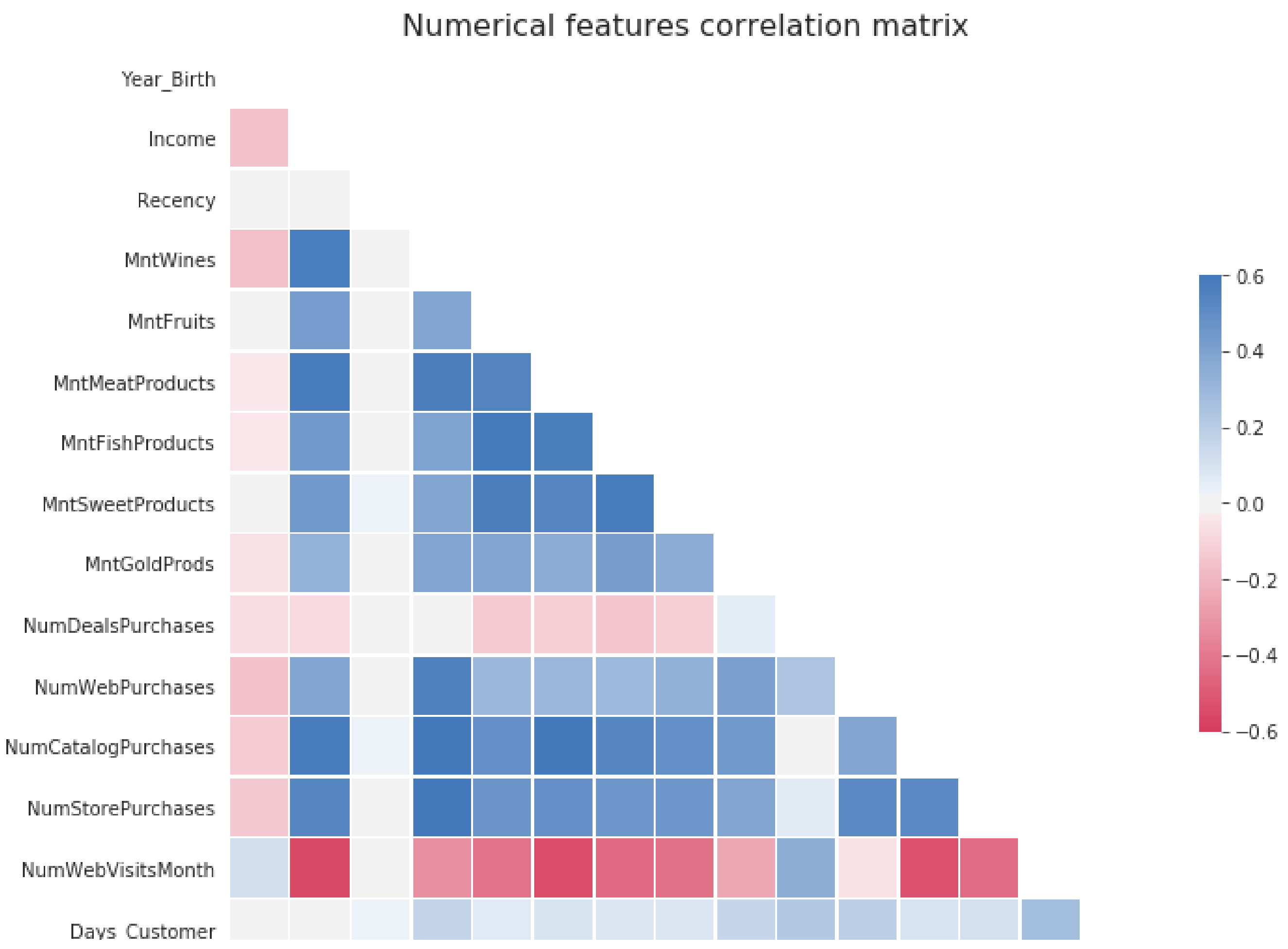


```
Z_CostContact    0.0
Z_Revenue        0.0
Name: std, dtype: float64
```

These two variables will be dropped since they are constants. Observations with age above 90 years will be dropped as well, since it is not likely that they are real values.

```
data[(2020 - data["Year_Birth"])>90]
```

Let's observe the correlation matrix of the discreet variables.



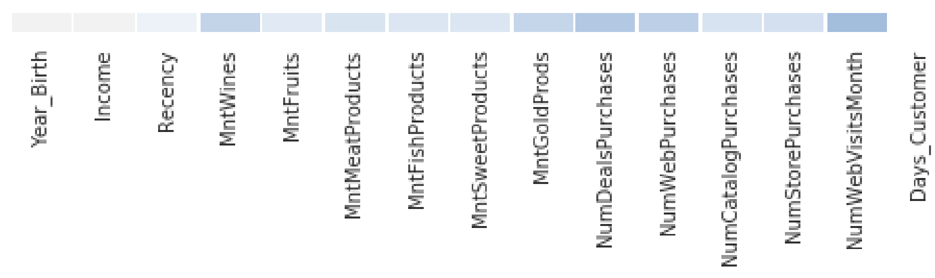


Figure 5— Correlation matrix

And the values distribution according to the campaign response.

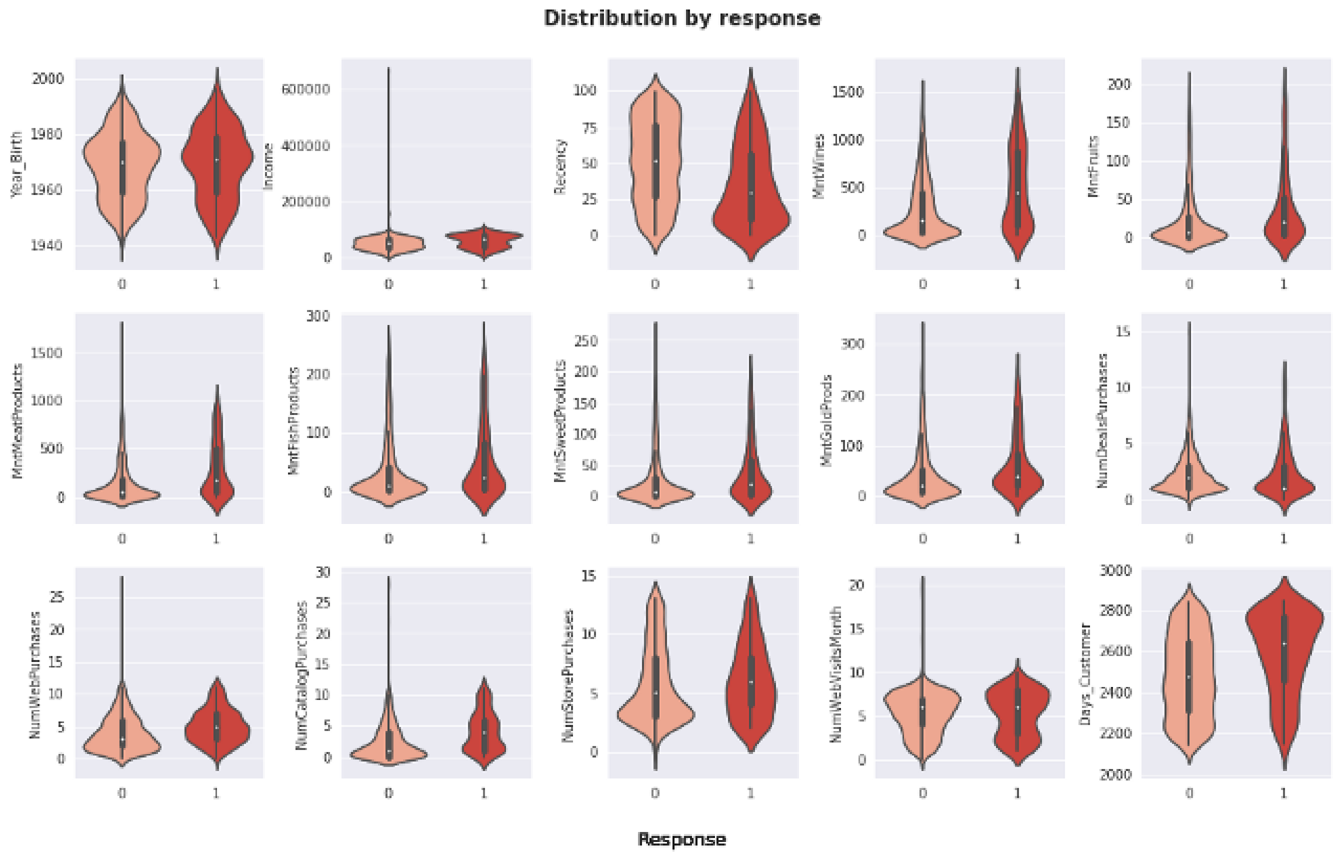


Figure 6— Values distribution by target

. . .

Outlier Detection

Through the use of the IsolationForest algorithm, from *scikit-learn*, we can calculate the anomaly score of each point of the numerical features and, then, search for outliers.

Red areas indicate where the probability of observations existing there is low. The blue line, representing the anomaly score, tends to be similar to the distribution of the variable.

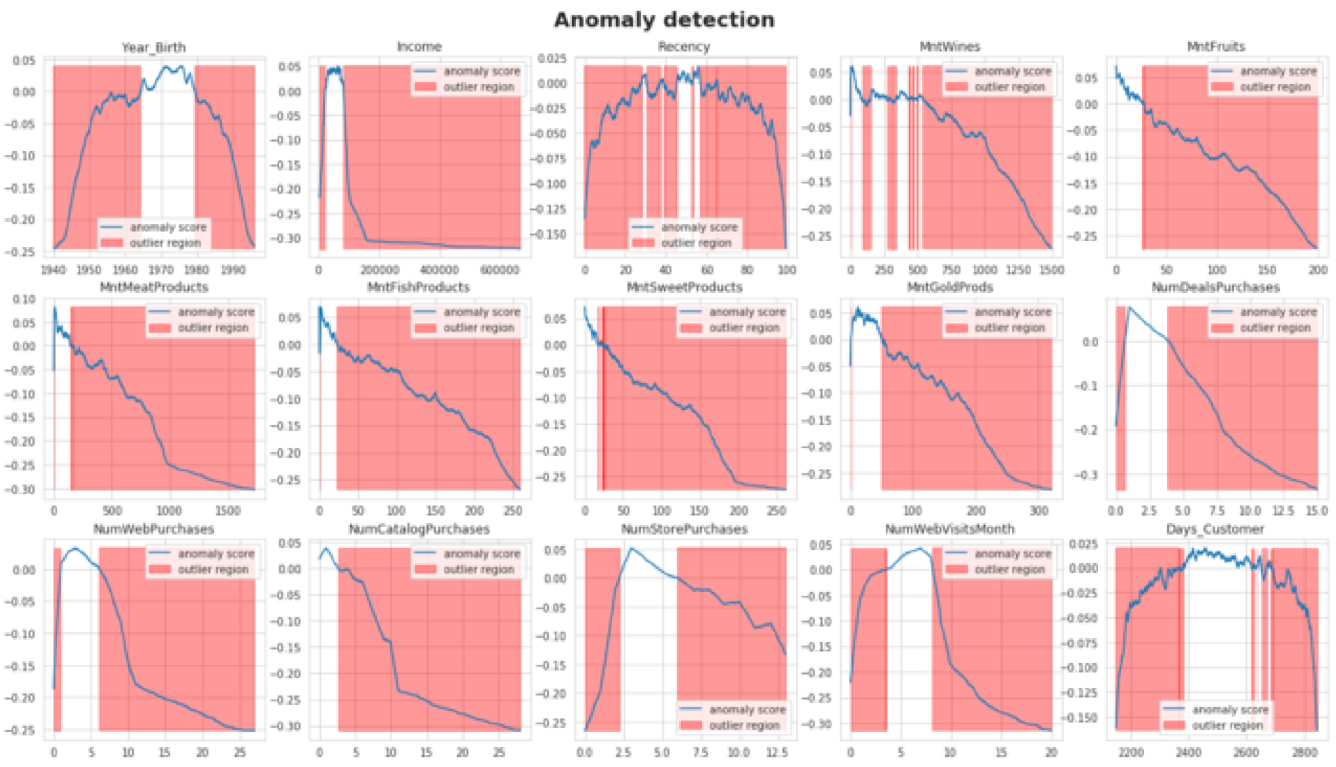


Figure 7 — Anomaly detection using IsolationForest

After splitting the data into train and test set, we can search for outliers.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data,
data["Response"], test_size=0.3,random_state=seeds[0])
```

There are several forms to detect outliers, but we are going to use here the Mahalanobis distance to find multivariate outliers. The Euclidian distance is known to fail to find outliers when dealing with multi-dimensional data. So we use the Mahalanobis Distance, because, since it uses the Eigenvalues of the variables instead of the original axis, it can make something similar to a feature scaling.

Simply putting, it calculates the distance of each point to the center of mass measured in standard deviations through the inverted covariance matrix. With these functions we can find the rows which represent multivariate outliers and remove them.

```
# The function to calculate the Mahalanobis Distance. Returns a list
of distances.
def MahalanobisDist(data):
    covariance_matrix = np.cov(data, rowvar=False)
    if is_pos_def(covariance_matrix):
        inv_covariance_matrix = np.linalg.inv(covariance_matrix)
        if is_pos_def(inv_covariance_matrix):
            vars_mean = []
            for i in range(data.shape[0]):
                vars_mean.append(list(data.mean(axis=0)))
            diff = data - vars_mean
            md = []
            for i in range(len(diff)):

md.append(np.sqrt(diff[i].dot(inv_covariance_matrix).dot(diff[i])))
            return md
        else:
            print("Error: Inverse of Covariance Matrix is not
positive definite!")
    else:
        print("Error: Covariance Matrix is not positive definite!")


# Function to detect multivariate outliers from the Mahalanobis
Distances. Returns an array of indexes of the outliers.
def MD_detectOutliers(data, extreme=False):
    MD = MahalanobisDist(data)

    std = np.std(MD)
    k = 3. * std if extreme else 2. * std
    m = np.mean(MD)
    up_t = m + k
    low_t = m - k
    outliers = []
    for i in range(len(MD)):
        if (MD[i] >= up_t) or (MD[i] <= low_t):
            outliers.append(i)  # index of the outlier
    return np.array(outliers)
```

. . .

Feature Engineering

Now, to compare features according to their value, we can extract new features from the ones we already have to search for better features. We will create all the new features and will compare and rank them afterward.

First of all, we focus on the creation of business-oriented features. Then, we apply the Box-Cox transformations on all features, trying to find the ones that best fit in terms of discrimination ability. Finally, we merge the rest of the categories with low discrimination ability, and apply PCA to try to reduce the dimensionality of the attributes.

Business-oriented features:

- RFM(Recency, Frequency, and Monetary value);
- Number of campaigns accepted;
- Proportion of money spent in each product (wines, meat, gold products, etc.);
- Monetary value (total spent);
- Buy potential (proportion of total spent and income);
- Frequency of product shopping.

Now we apply the Box-Cox Transformation.

> *A Box-Cox transformation is a way to transform non-normal dependent variables into a normal shape. Normality is an important assumption for many statistical techniques; if your data isn't normal, applying a Box-Cox means that you are able to run a broader number of tests.*

```
Best Power Transformation for each feature:
         -> NumCatalogPurchases sqrt
         -> NumStorePurchases sqrt
         -> NumWebVisitsMonth sqrt
         -> Days_Customer **2
         -> PrpGoldProds **1/4
         -> NmbAccCmps x
         -> PrpAccCmps x
         -> PrpWines sqrt
         -> PrpFruits sqrt
         -> PrpMeat sqrt
         -> PrpFish **1/4
         -> Mnt **2
         -> BuyPot exp
         -> Freq **2
         -> RFM **2
```

Now we merge the less significant categories and we get dummy variables for the left ones. Then we apply PCA, which can either lead to a good summarization of the data or an excessive loss of information. Note that it is not correct to use categorical variables for principal component analysis.

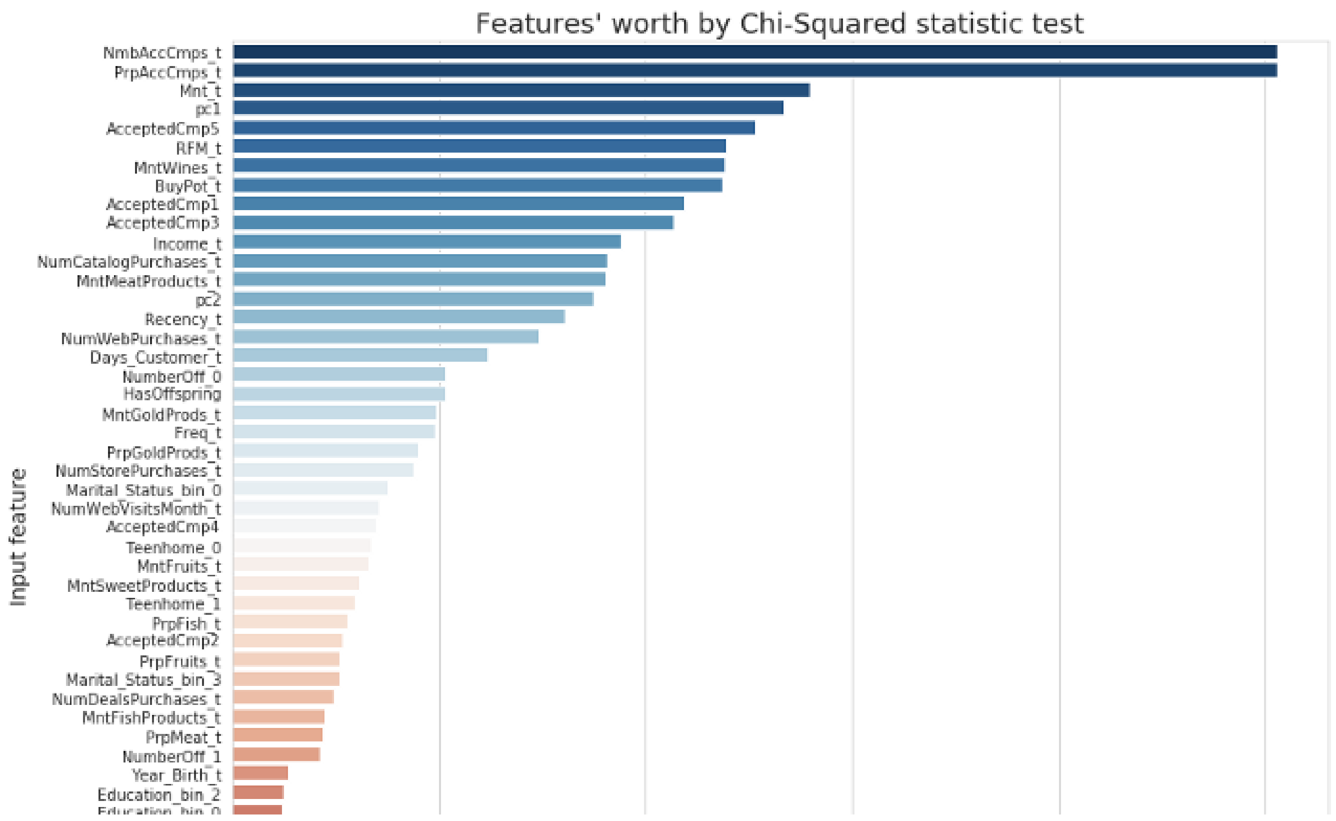
> *The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent.*

```
from sklearn.decomposition import PCA

columns = X_train.columns
columns =
columns.drop(['Kidhome','Teenhome','NumberOff','AcceptedCmp3',
'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1','AcceptedCmp2',
'Complain', 'Response', 'Marital_Status_bin', 'Education_bin',
'HasOffspring'])

pca = PCA(n_components=2)
principalComponents = pca.fit_transform(X_train[columns])
principalComponents_test = pca.transform(X_test[columns])

X_train["pc1"] = principalComponents[:,0]
X_train["pc2"] = principalComponents[:,1]
X_test["pc1"] = principalComponents_test[:,0]
X_test["pc2"] = principalComponents_test[:,1]
```

. . .

Feature Selection

Feature selection is the process of selecting a subset of relevant features for use in model construction. There is a variety of tests that can be used to measure the worth of the attributes, but the one we use is the Chi-Square test. Here is the graph of how each attribute performed.



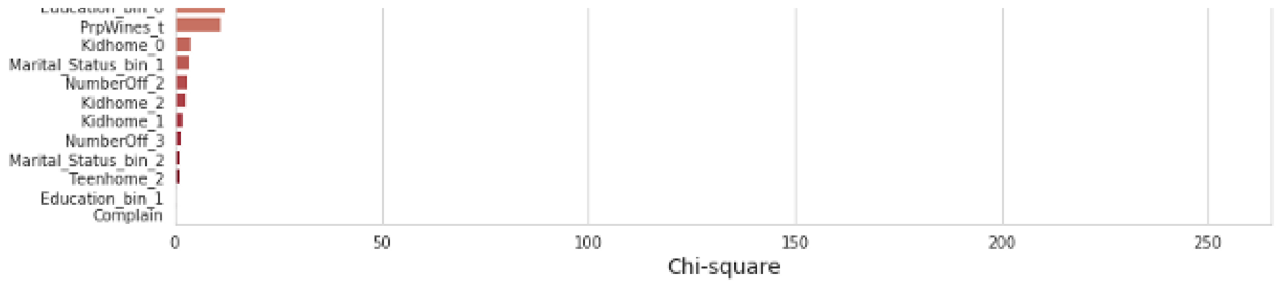


Figure 8— Features' worth

We are selecting the 15 features with highest score to model the data.

. . .

Data Modeling

The training set is imbalanced, so the first step is fix this issue.

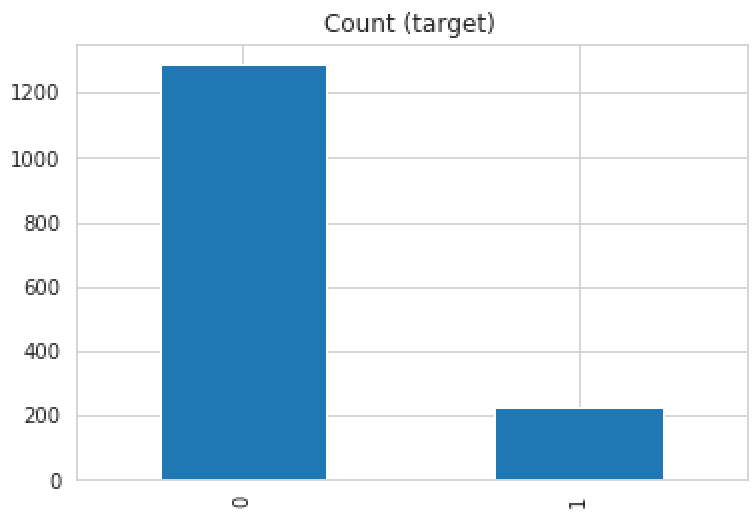


Figure 9 — Dataset balance

```
from imblearn.over_sampling import SMOTE

smote = SMOTE()
X_train, y_train = smote.fit_resample(X_train, y_train)

y_train.value_counts().plot(kind='bar', title='Count (target)');
```

After having applied the oversampling technique we can proceed to the modeling step.

Now applying a Logistic Regression.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

LR = LogisticRegression()
LR.fit(X_train, y_train)

y_pred = LR.predict(X_test)
print('ROC score: {}'.format(roc_auc_score(y_test, y_pred)))
```

ROC score: 0.98

. . .

Conclusion

In this article is described how Machine Learning can serve as a tool for targeted marketing. It also showed all the steps of how to do that. Obviously, this is a toy example, and it is not likely that the performance of the model will be this high in a real case scenario. Anyway, even if the model reaches a score of about 85%, which is pretty reasonable, the profit margin will suffer a significant boost by eliminating unnecessary contacts.

Feel free to check out my code or message me on my LinkedIn.

A note from the Plain English team

Machine Learning Artificial Intelligence Marketing Data Science Python

App Store', and if clicked it will lead you to the iOS App store

Google Play', and if clicked it will lead you to the Google Play store