

Machine Learning Engineer Nanodegree

Capstone Project

Jair Miranda

April 29 at, 2021

I-Definition

Project Overview

Customer segmentation looking to general population data, and Arvato Company clients data,

The idea identifies with general population data those who have the greatest chance of becoming customers.

In the project I used a lot of different algorithms and also created pipelines to clean up the data,

and made some feature engineering.

Problem Statement

Customer segmentation is a common problem on machine learning projects, mainly if you're looking to improve the marketing campaign results.

Arvato Financial Solutions has a lot of demographic and customer data, and we'll try to find patterns on this data,

Understanding the difference between customers and non-customers.

We have train, test dataset too, but all 4 datasets have a serious problem with nullity data, and wrong types.

The most interesting thing about this project is that it is a real problem, and we can violate it in any company,

besides that having a good result can generate high profitability for the company

It was hard to deal with all data, heavy for the memory.

We'll start analyse the data and use catboost to check feature importance

The goal is to create a model or model pipeline to understand what's the change of new lead to be a client for that we ll:

1- Analyse the Data.

Understand the Data, creating nullity dictionary, and reading the description, of each column, trying to get an insight, 'fast analyze'.

2- Clean Up the Data.

First I created dictionary with all possible missing data by column, after that i impute nan for all invalid our null values.

I also remove columns without description.

3- Create a Data Wrangler pipeline.

I create a class and function to make life much more easy when deal with other dataframes with the same colums

4- Make Unsupervised Analysis.

I used PCA to reduce the high dimensional space, and after that Knn to clusterize the data, and try to find similarities.

5- Create Unsupervised Pipeline.

1. Sklearn Function transform
2. Sklearn SimpleImputer - to transform nans.
3. Sklearn Pipeline - to create a pipeline for each type of data.
4. Sklearn StandardScale - to deal with numeric transformation and make the computation part much more ez.
5. Sklearn ColumnTransform to use all pipeline together, and make complete transformation pipeline

6- Make Supervised Learning:

1. Catboost
2. Xgboost
3. AdaBoost
4. Logistic Regression

7- Create a Supervised Pipeline:

I create supervised pipeline with mlflow to make life more ez when run different algorithms, as described above, also used sklearn `RepeatedStratifiedKFold` to avoid overfitting.

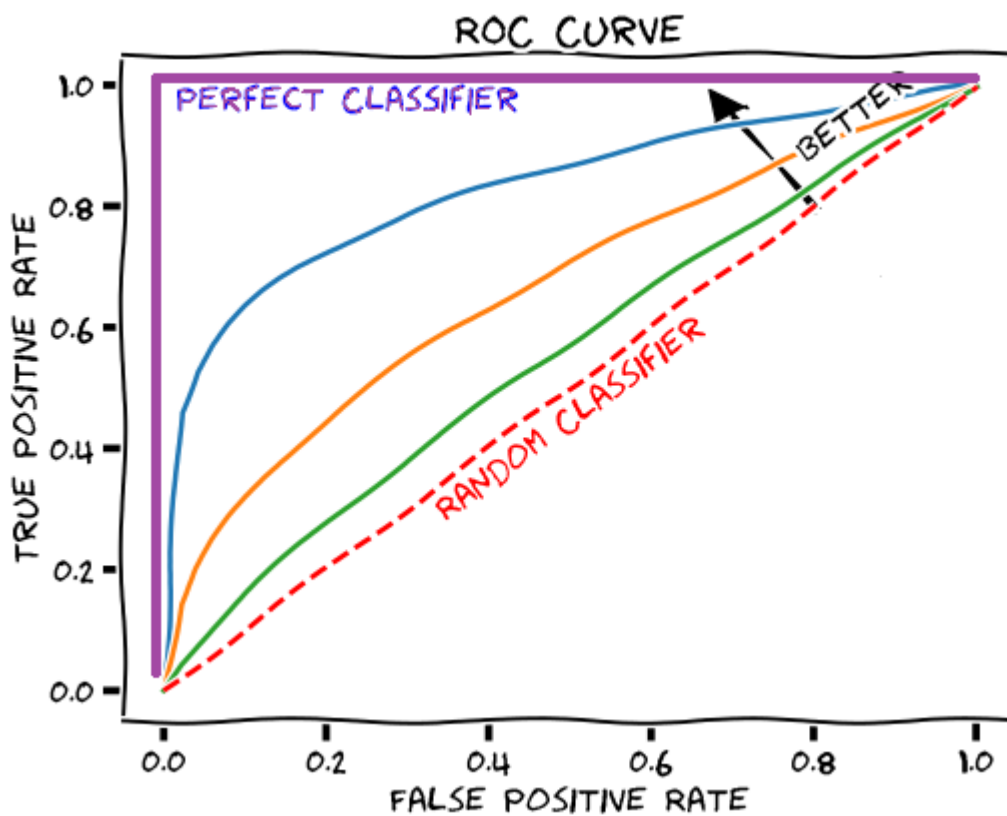
8- Evaluate.

GridSearchCV was the choice to find the best hyperparameters, to get a little more % on result.

Metrics

Measuring Performance: AUC (AUROC).

I choose AUCROC because on the kaggle that was the metric, for submission, also rocauc help us to find the balance between TP/FP and TN/FN, bringing the solution closer to presenting a real value.



How we calculate Roc/AUC Curve:

TPR (True Positive Rate) / Recall / Sensitivity

$$\text{TPR / Recall / Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Image 3

Specificity

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Image 4

FPR

$$\text{FPR} = 1 - \text{Specificity}$$

$$= \frac{\text{FP}}{\text{TN} + \text{FP}}$$

Image 5

image from <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>

Data and Inputs

There are four data files associated with this project:

- `Udacity_AZDIAS_052018.csv`: Demographics data for the general population of Germany; 891 211 persons (rows) x 366 features (columns).
- `Udacity_CUSTOMERS_052018.csv`: Demographics data for customers of a mail-order company; 191 652 persons (rows) x 369 features (columns).

- `Udacity_MAILOUT_052018_TRAIN.csv`: Demographics data for individuals who were targets of a marketing campaign; 42 982 persons (rows) x 367 (columns).

- `Udacity_MAILOUT_052018_TEST.csv`: Demographics data for individuals who were targets of a marketing campaign; 42 833 persons (rows) x 366 (columns).

II-Analyses

Data Exploration

I used Azdias dataset to make analyses

Most of the columns have less than 20% missing values.

I drop those columns that have greater than 20% missing values

And below graph shows the top 50 names of columns along with its missing percent.

I also removed any columns that are not in attr or info dataframe.

After that we moved to 243 columns, better than 366.

After that I did a lot of transformation in different columns, and created a pipeline for feature selection.

I found categorical data and numeric data, but to choose what to do on pre processing

i did some analysis as used pandas skew, to check high skewed data

Important Notes:

- *If the skewness is between -0.5 and 0.5, the data are fairly symmetrical*
- *If the skewness is between -1 and — 0.5 or between 0.5 and 1, the data are moderately skewed*
- *If the skewness is less than -1 or greater than 1, the data are highly skewed*

from -> <https://medium.com/@atanudan/kurtosis-skew-function-in-pandas-aa63d72e20de>

We Found a lot of unbalance data when i applied:

```
12 skewed = pd.DataFrame(azdias_df[numeric_columns].skew(skipna=True), columns=['value'])
```

```
8 skewed.head()
```

	value
AGER_TYP	0.405509
ALTERSKATEGORIE_GROB	0.201513
ALTER_HH	-0.409425
ANZ_HH_TITEL	7.273441
ANZ_PERSONEN	2.112416

So I did a Log transform on these columns to normalize the data, using sklearn Function Transformer I could use `np.log` inside the pipeline, fixing data.

```
# Log Transform
```

```
transformer = FunctionTransformer(np.log1p)
```

Log pipeline:

```
log_pipeline = Pipeline(steps=[
```

```
    ('natural_log', transformer),
```

```
    ('imputer', SimpleImputer(missing_values=np.nan, strategy='median')),
```

```
    ('scaler', StandardScaler())
```

```
])
```

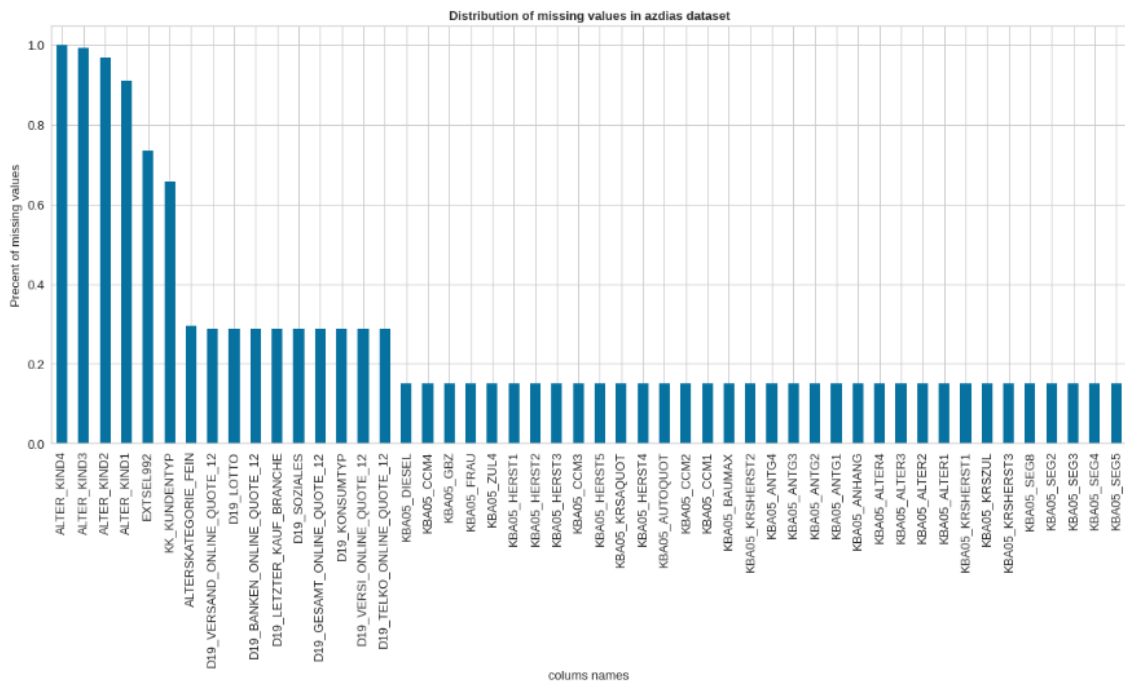
On feature Selection i also check the high correlated columns and remove, because i dont think it can bring a lot of different value for the unsupervised part. For that I created a function, with a threshold of 95% correlation.

```
def correlated_columns_to_drop(df, min_corr_level=0.95):
    """Drop columns based on high correlated columns.

    Args:
        df (pd.DataFrame): Dataframe with columns to check correlation.
        min_corr_level (float, optional): Minimum correlation to choose to drop. Defaults to 0.95.

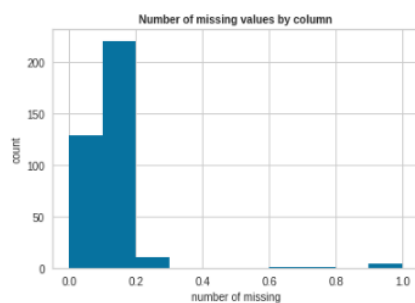
    Returns:
        List[List[str], pd.DataFrame]: List with columns dropped, and dataframe without this columns.
    """
```

Exploratory Visualization



```
In [14]: plt.hist(missing_by_col)
plt.title("Number of missing values by column",fontsize=10,fontweight="bold")
plt.xlabel("number of missing",fontsize=10)
plt.ylabel("count",fontsize=10)
```

Out[14]: Text(0, 0.5, 'count')



Algorithms and Techniques

First I applied catboost to discover the feature importance and after that I saw 80% of values = 1 in D19_SOZIALES are customers.

Unfortunately I don't have a description for these columns. I can use later to reach better results.

After That I did some transforms using sklearn pipeline and Column Transform:

Time to create or pipeline of transformation

```
In [16]: # Using Pd Get Dummies to transform category columns to Dummies
azdias_df = pd.get_dummies(azdias_df, columns=CATEGORICAL_COLUMNS)
```

```
In [21]: # Saving Columns to compare with others dataframes
import json

with open('../data/cleaned_data/azdias_columns.json', 'w') as jsonfile:
    json.dump({'columns':azdias_df.columns.to_list()}, jsonfile)
```

```
In [17]: #Numeric transformation
numeric_pipeline = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

# For Binary
binary_pipeline = SimpleImputer(missing_values=np.nan, strategy='most_frequent')

# Log Transform
transformer = FunctionTransformer(np.log1p)

log_pipeline = Pipeline(steps=[
    ('natural_log', transformer),
    ('imputer', SimpleImputer(missing_values=np.nan, strategy='median')),
    ('scaler', StandardScaler())
])
```

```
In [18]: transformers = [('numeric', numeric_pipeline, list(numeric_columns_final)),
                        ('binary', binary_pipeline, binary_columns),
                        ('log', log_pipeline, skewed_columns)
                        ]
```

```
In [19]: column_transformer = ColumnTransformer(transformers=transformers, remainder='passthrough')
```

```
In [18]: column_transformer.fit(azdias_df)
```

```
Out[18]: ColumnTransformer(remainder='passthrough',
                           transformers=[('numeric',
                                           Pipeline(steps=[('imputer',
                                                             SimpleImputer(strategy='median')),
                                                             ('scaler', StandardScaler())]),
                                           ['KBA13_HERST_ASIEN', 'LP_STATUS_FEIN',
                                            'WOHNLAG', 'PLZ8_ANTG1', 'KBA13_KW_110',
                                            'FINANZ_ANLEGER', 'ORTSGR_KLS9',
                                            'KBA13_FAB_SONSTIGE', 'KBA13_KW_60',
                                            'KBA13_B1_2009', 'INNENSTADT', 'KBA13_KW_120']
```

I also use PCA to reduce the dimensionality, because 366 columns it's a lot of information, and heavily processed on the computer. Also some information can present the same value, or even disturb each other.

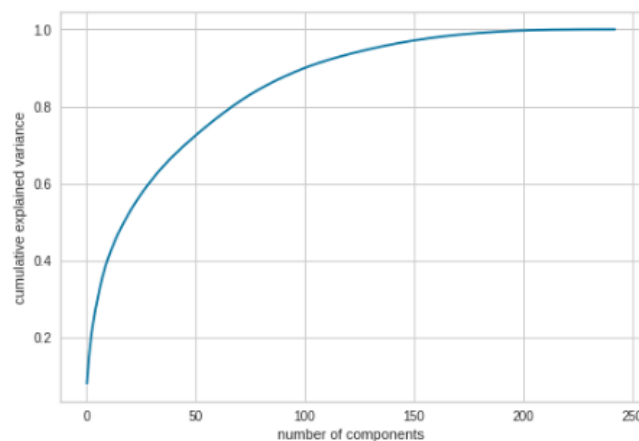
How does PCA do this; pca is an unsupervised learning method that finds the linear combination of features, the principal components, basing on direction of variance, and large space.

The first components explain most the variance, and after that you'll have other components with features, the greater the component the less it explains the variance, for this pca compute the Eigenvalues and Eigenvectors.

$$A\vec{v} = \lambda\vec{v}$$

We have 97% of explicability with 160 features

```
!ut[29]: <Figure size 720x1080 with 0 Axes>
```



```
<Figure size 720x1080 with 0 Axes>
```

```
!n [30]: # The first 5 components explain 27.43% of variance
pca.explained_variance_ratio_[:5].sum()
```

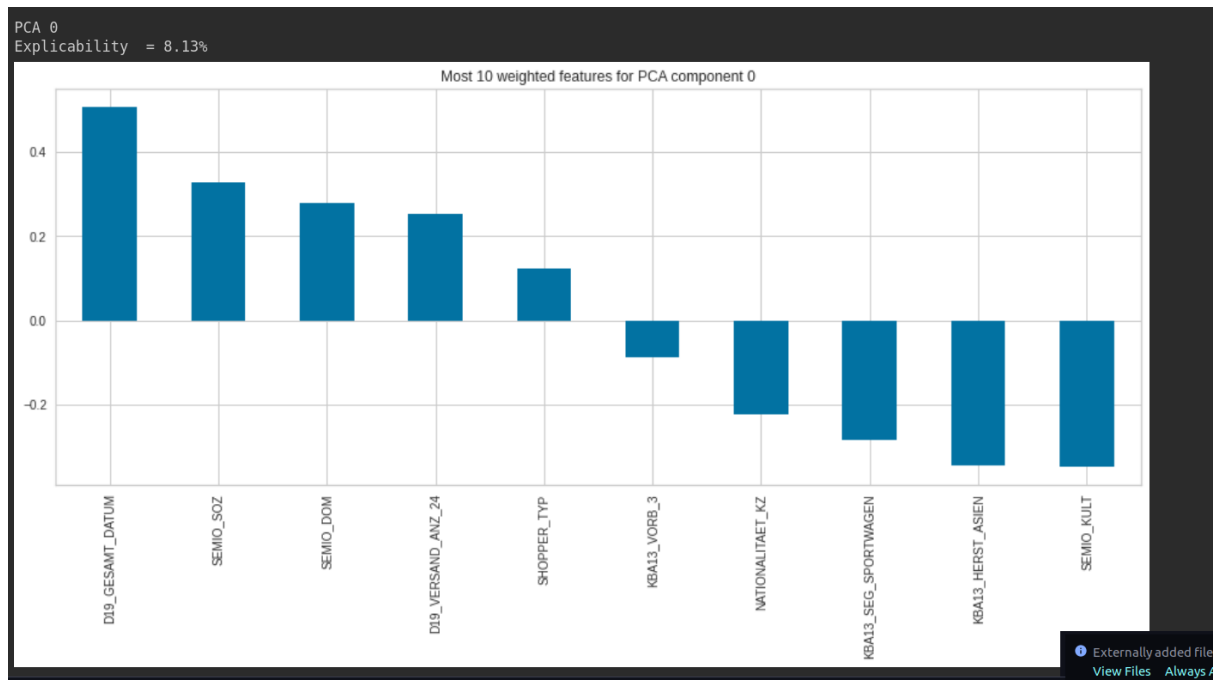
```
!ut[30]: 0.27431025412905397
```

```
!n [31]: for i in np.arange(10, 190, 10):
        print('{} components explain {}% of variance.'.format(i, pca.explained_variance_ratio_[:i].sum().
```

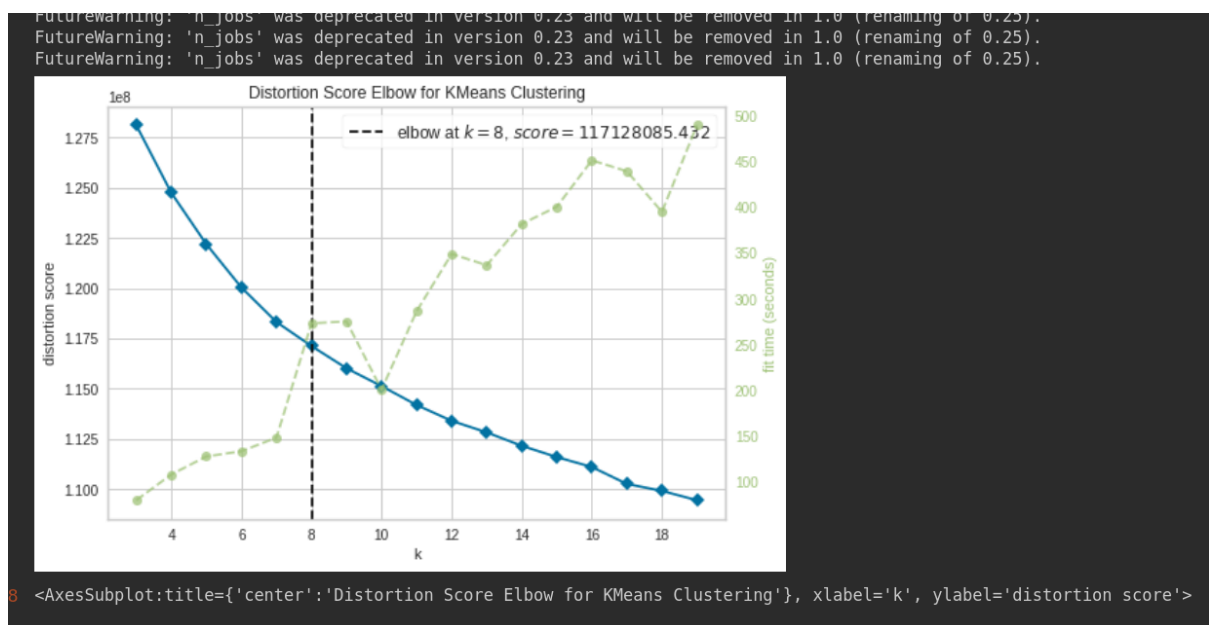
10 components explain 39.4% of variance.
20 components explain 52.0% of variance.
30 components explain 60.4% of variance.
40 components explain 66.8% of variance.
50 components explain 72.1% of variance.
60 components explain 76.7% of variance.
70 components explain 80.9% of variance.
80 components explain 84.39999999999999% of variance.
90 components explain 87.4% of variance.
100 components explain 89.8% of variance.
110 components explain 91.8% of variance.
120 components explain 93.4% of variance.
130 components explain 94.8% of variance.
140 components explain 96.0% of variance.
150 components explain 97.0% of variance.
160 components explain 97.89999999999999% of variance.
170 components explain 98.5% of variance.
180 components explain 99.0% of variance.

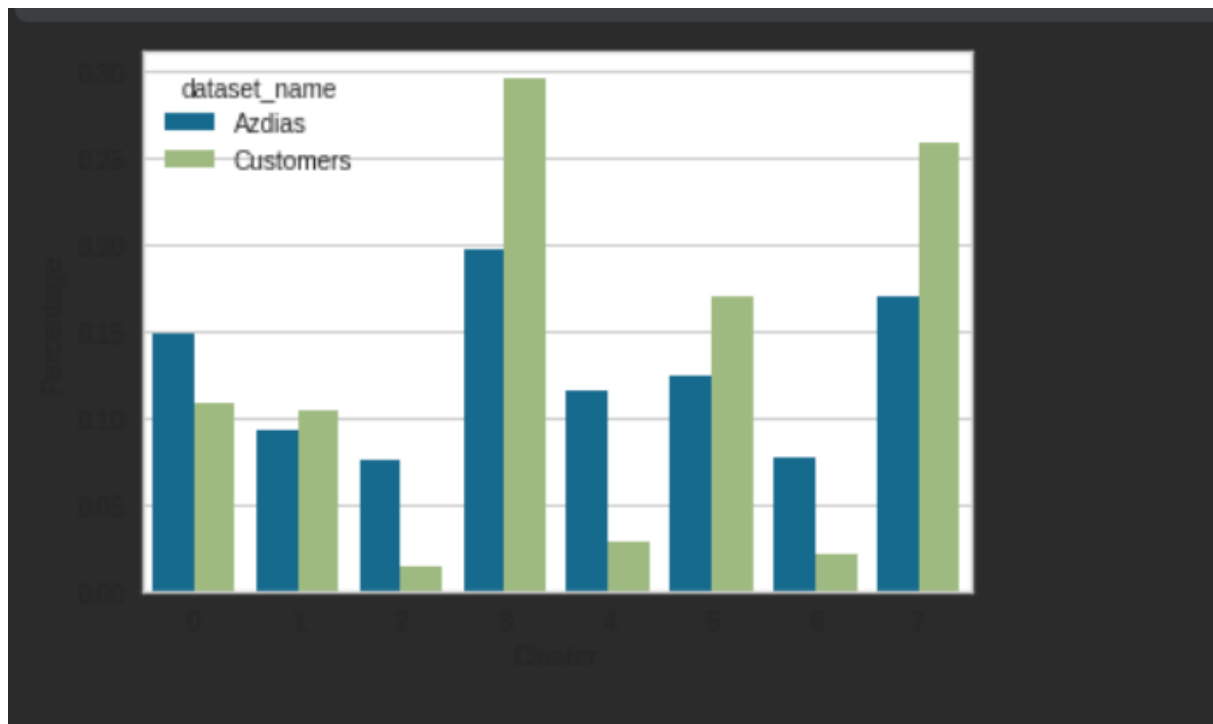
And we can understand better the data here

https://github.com/Jair-Ai/arvatoKaggle/blob/master/notebooks/analyse_one.ipynb



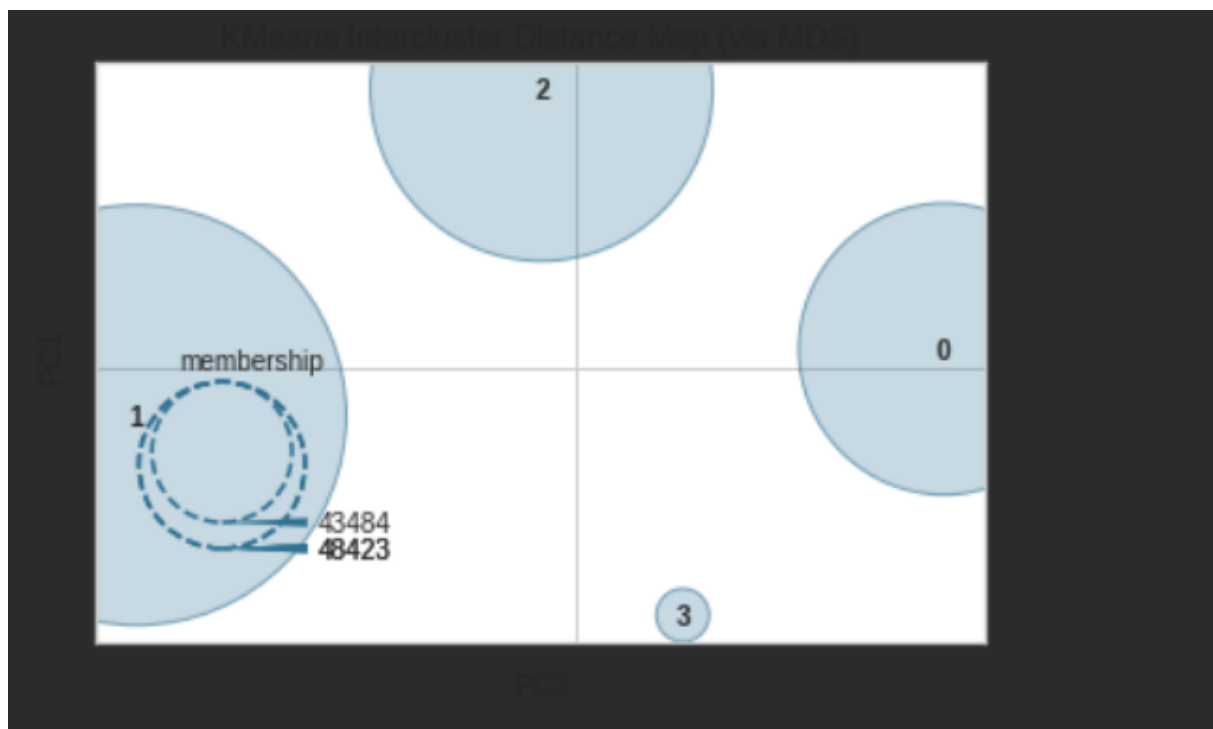
And Knn for feature selection, Knn is an algorithm, based on feature similarity, try to find the centroids and cluster the data around this centroids, for that we need to find the better number of centroid, who represent the best explanation about the data, so i trained knn from 1 to 21 centroids, and plot all to try to find the Elbow, between the k and distorsion score:





As we can see some clusters have more representative data, and that's an important part, because with less data, we have a chance to have a cluster misclassified;

But with 4 clusters we have a very good distance between the centroids, to have this vision I used the excellent plot library yellowbrick.

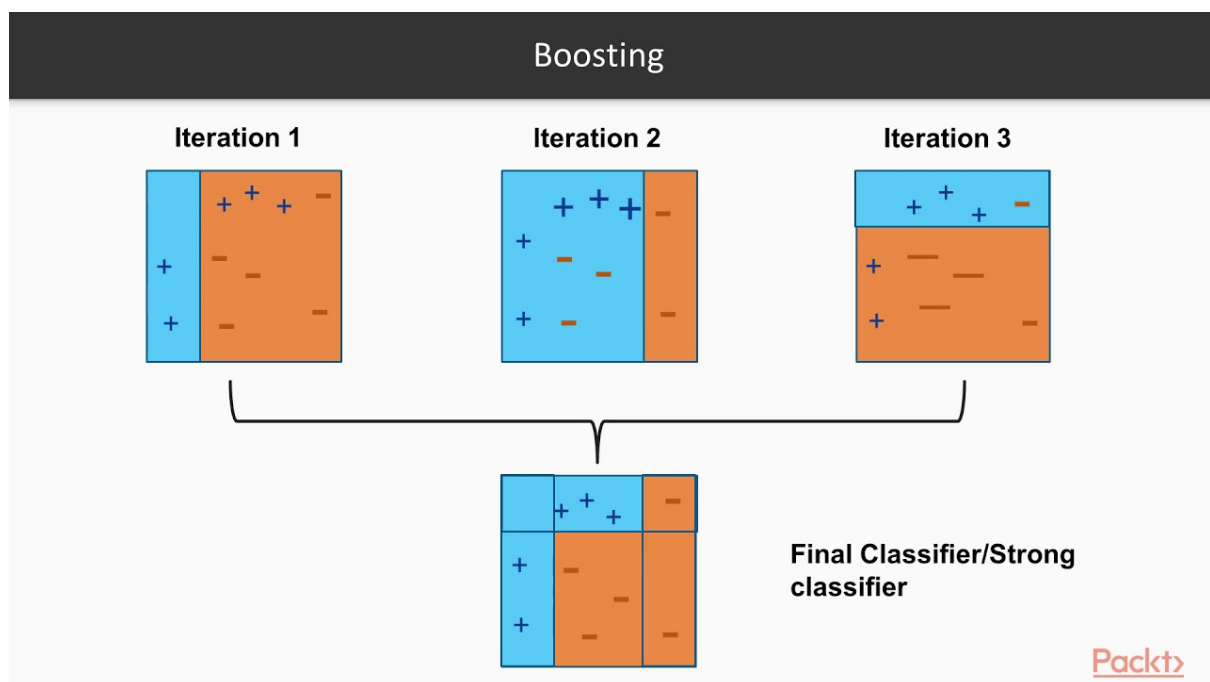


For Supervised learning I used catboost and adaboost, and Logistic Regression with GridSearchCV.

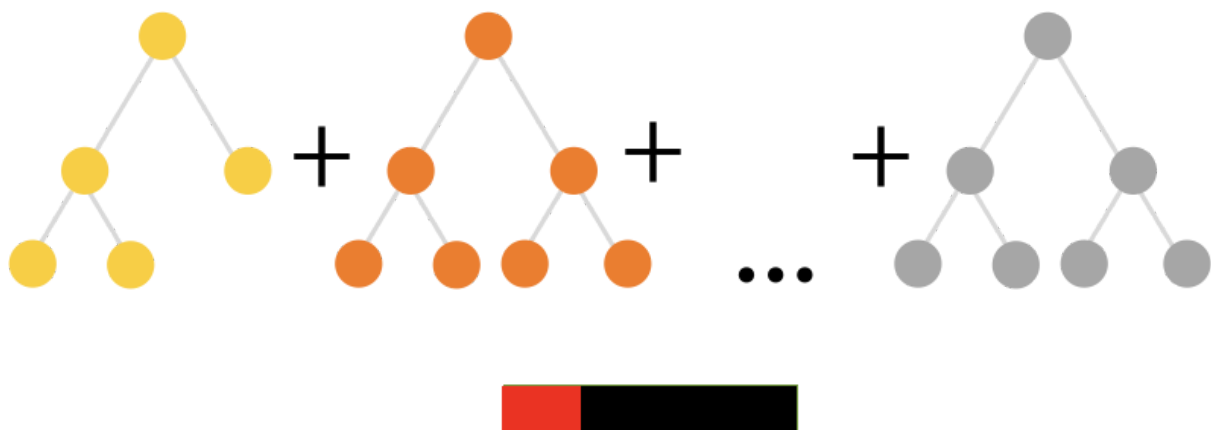
Logistic Regression is a base model, decisions are between 0 and 1, according with wikipedia LR is a statistical model using logistic function to model the binary problem, it's like logistic regression for classification models;

$$\ell = \log_b \frac{p}{1-p} = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Adaboost is a boosting method using trying to combine weak learners in a strong classification, and as i base weak learners i used DecisionTree.



Catboost is a library who can deal with categorical data, it's based on decision tree and using gradient boost, ensemble of week learners, also we can use GPU to accelerate training process



BenchMark

Kaggle leaderboard was my benchmark, i dont have so many time to try, but i did 18 attempts and the best result was using D19_SOZIALES

[catboost_all_features_weights.csv](#)

2 months ago by [Jair Ai](#)

0.79228



CatBoost [30 trees, 3 depth] 0.2 sample, all features, wheights for socialez

III-Methodology

Data Preprocessing

I created a dataframe with all null values from info and attr dataframe, we 0,-1,X,XX and sometimes 9 or 10 in some columns.

I created and applied 5 different pipelines to pre- processing data:

1- Data Wrangler Pipeline -> data_wrangler.py.

1. Load the nan dataframe with all invalid values, and set all as nan values
2. Check and create list with all columns with description on Attr and Info datasets
3. Remove all columns without description(i did this because the number of columns are so big 366).

2- Feature Engineer Pipeline -> feature_engineer.py.

1. On CAMEO_DEUINTEL_2015 we have 2 information Wealth and Life Age, i separated in 2 different columns and dropped the original columns.
2. On PLZ8_BAUMAX we have 2 informations too Family and business, same process as above.
3. On PRAEGENDE_JUGENDJAHRE has information about generation and movement, same process.
4. Take the Binary columns and set values as 0 and 1 ('OST_WEST_KZ', 'VERS_TYP', 'ANREDE_KZ')
5. Remove the correlated columns.
6. Removing columns with not relevant information.
7. Make get_dummies with pandas on categorical columns.

3- Preparing for unsupervised -> models/unsupervised_transform.joblib.

1. PCA using 97% of explanation features (160 components)
2. Knn using elbow technique to get n clusters - Choose 8 and 4.

4- Unsupervised Learning pipe -> models/unsupervised_transform.joblib.

5- Supervised Learning -> train.py.

Implementation

After create pipelines i start to train using this [Jupyter Notebook](https://github.com/Jair-Ai/arvatoKaggle/blob/master/notebooks/supervised_learning.ipynb)

I think I documented everything very well.

Refinement

I tried to use Hyperopt for bayesian hyperparameters tuning, but i didn't had time to finish it with mflow, so i used GridSearchCV

with RepeatedStratifiedKFold, to find the best algo.

Also I did some tests with the most important feature D19_SOZIALES, but I didn't have time to make reports with that.

Was hard to work with this amount of data, my computer crashed a lot of times.

GridSearchCV guarantees that we reach the best result as we can without overfitting, avoiding to put a bad model in production.

But the results without or with are not good AUC/ROC give me 0.5%, and 98% of accuracy, because we have much more than as not client.

I think the number of clusters should increase as a result, also I did some play with feature importance and weak learners went from 0.5 to 0.75 on ROC/AUC.

During the code process the big problem for me was the amount of data, my computer always crashes because of memory issues, so it took a lot of time to make analyses, it's better working with samples like 0.2 and 0.4 of data population.

IV-Results

Model Evaluation and Validation

I used Mflow to track the improvements, it worked very well, and could document the features n of columns and dataset used.

mlflow

Experiments

Models

Experiments

Search Experiments

Default

CatBoostClassifier

Cartboost_2

LogisticRegression

AdaBoost

Track machine learning training runs in an experiment. [Learn more](#)

Experiment ID: 1

Artifact Location: .\artifacts\1

Notes

Search Run: Filter Search Clear

Showing 19 matching runs [Compare](#) [Delete](#) [Download CSV](#)

	Start Time	Run Name	User	Source	Version	Models	cat_features	class_weights	cv	TestAUC	TestAcc	TrainAUC	dataset	sample	algo
<input type="checkbox"/>	2021-09-19 08:12:55	Fit_Lat	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 4.6802180313, -	-	0.925	0.882	0.927	population_costs...	0.2	catboost	
<input type="checkbox"/>	2021-09-19 08:16:59	Fit_Lat	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 4.6802180313, -	-	0.925	0.882	0.927	population_costs...	0.2	catboost	
<input type="checkbox"/>	2021-09-19 08:40:20	Fit_Lat	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 4.6802180313, -	-	-	-	-	population_costs...	0.2	catboost	
<input type="checkbox"/>	2021-09-19 08:58:08	Fit_Lat	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 4.6802180313, -	-	-	-	-	population_costs...	0.2	catboost	
<input type="checkbox"/>	2021-09-19 08:34:03	Fit_Lat	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 4.6802180313, -	-	-	-	-	population_costs...	0.2	catboost	
<input type="checkbox"/>	2021-09-19 08:32:39	Fit_Lat	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 4.6802180313, -	-	-	-	-	population_costs...	0.2	catboost	
<input type="checkbox"/>	2021-09-08 09:52:57	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	[CAMEL_SGL_SRL, (5, 70.9451070488, -	-	0.939	0.813	0.749	train	1	catboost	
<input type="checkbox"/>	2021-09-08 06:58:02	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 4.6801980102, Inverted	-	0.893	0.797	0.896	population_costs...	0.4	catboost	
<input type="checkbox"/>	2021-09-08 06:46:19	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, -	-	0.893	0.797	0.896	population_costs...	0.4	catboost	
<input type="checkbox"/>	2021-09-08 17:47:09	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, -	-	0.888	0.864	0.89	population_costs...	0.2	catboost	
<input type="checkbox"/>	2021-09-08 17:15:45	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, -	-	0.888	0.864	0.89	population_costs...	0.2	catboost	
<input type="checkbox"/>	2021-09-08 17:05:24	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, -	-	0.888	0.864	0.89	population_costs...	0.4	catboost	
<input type="checkbox"/>	2021-09-08 09:48:29	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 4.6802283686, -	-	0.888	0.795	0.891	population_costs...	0.4	catboost	
<input type="checkbox"/>	2021-09-08 09:43:51	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 4.6802283686, -	-	0.888	0.792	0.898	population_costs...	1	catboost	
<input type="checkbox"/>	2021-09-08 09:42:45	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	-	-	-	-	-	population_costs...	1	catboost	
<input type="checkbox"/>	2021-09-08 09:27:06	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 4.6801980102, -	-	0.895	0.799	0.898	population_costs...	0.4	catboost	
<input type="checkbox"/>	2021-09-08 08:51:17	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 4.6802283686, -	-	0.889	0.76	0.888	population_costs...	1	catboost	
<input type="checkbox"/>	2021-09-08 08:46:46	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 3.9602991276, -	-	0.893	0.786	0.867	population_costs...	0.4	catboost	
<input type="checkbox"/>	2021-09-04 20:51:27	-	jai	<input type="checkbox"/> Ignored_Jausher	-	-	Indiv[CAMEL_D, (5, 3.1081970868, -	-	0.867	0.781	0.868	population_costs...	0.2	catboost	

Load more

Experiments

Models

CatBoostClassifier > Run 17fd467acb334cb6be4e505546e17466 ▾

Date: 2021-03-05 09:21:06

Source: ipykernel_launcher.py

Duration: 2.6s

Status: FINISHED

▼ Notes

None

▼ Parameters

Name	Value
cat_features	Index(['CAMEO_DEU_2015', 'OST_WEST_KZ'], dtype='object')
class_weights	(1, 4.650150010249147)
eval_metric	AUC
max_depth	5
min_child_samples	30
num_trees	30
od_type	lter
od_wait	40
one_hot_max_size	5
random_state	42
task_type	GPU

▼ Metrics

Name	Value
TestAUC	0.895
TestAcc	0.799
TrainAUC	0.898
TrainAcc	0.801
ValidAUC	0.896
ValidAcc	0.8

I got this cat boost with max depth 5 and 30 trees, so as we can see the algorithm combines 30 different trees to find the best result, and we don't have overfitting here, because i find very good results on the test set.

I trained AdaBoost and LogisticRegression with RepeatedStratifiedFold and gridsearch but without a good result, maybe because the number of clusters(8) was not good!

```
def train_grid_search(self, model, grid: Dict[str, Union[str, float, List[Union[str, int, float]]]]):
    cv = RepeatedStratifiedKFold(n_splits=20, n_repeats=3, random_state=settings.RANDOM_STATE)
    grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv, scoring='roc_auc', verbose=2)



    grid_result = grid_search.fit(self.X_train, self.y_train)

    print(grid_result.best_score_)
    print(grid_result.best_estimator_)
    compute_metrics(grid_result.best_estimator_, self.X_test, self.y_test)
```


Justification

My final solution is still a catboost with all features. the pipeline transformation show me a lot of data interpretation,

but I couldn't find anything to beate Catboost. 79.238 % on Kaggle dataset and this follow result on train/test/validation

165	Jair Ai		0.79238	18	6m
Your Best Entry 					
Your submission scored 0.50092, which is not an improvement of your best score. Keep trying!					

In the real world this result is very good because if you look on the curva ROC-AUC we have a good % of chance of each person to be a client of the company, spending less money, because we draw here some profiles of possible customers.

V-Conclusion

It was a little disappointing to make so many transformations, understand the data and have such a bad result even with Gridserchcv,

50%, nothing better than random walk, while catboost proved to be very efficient, doing a wonderful job.

```
Fitting 60 folds for each of 32 candidates, totalling 1920 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done 17 tasks      | elapsed: 0.1s
[Parallel(n_jobs=-1)]: Done 852 tasks    | elapsed: 2.4min
[Parallel(n_jobs=-1)]: Done 1640 tasks   | elapsed: 5.7min
[Parallel(n_jobs=-1)]: Done 1920 out of 1920 | elapsed: 9.6min finished
0.542608204278298
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1),
                    learning_rate=0.1, n_estimators=30, random_state=42)
```

Reflection

I think in the real life I can do a better work with the features if I have more time using cluster with feature importance,

and using a Bayesian method to improve and find better hyper parameters.

Was a great project, kind of hard because I had this dirty data, and need a good computer power to process everything well.

But 79.9% is not so bad, isn't?