ML ND CAPSTONE PROJECT

Project Report

Amar Singh | ML ND | 13-07-2020

arvato

BERTELSMANN

Table of Contents

Domain background:

In this world making the business grow faster and better is the toughest job than starting the new business. Finding or you can say targeting the specific customers that can use you product is very important for the company as it reduces the marketing campaign cost and increase its profit.
Arvato Financial Solution does the same thing it provides solution to different company so that they can grow their business quickly. They provide financial solution also.
Arvato company has teamed up with the udacity team to provide the project for ML nanodegree in which we have to find customer segmentation which will respond to the mail beig sent by the company. Bertelsmann was founded in 1835 and provide guidance for business development.

Project Overview

Nowdays every company wants to target customers which will buy or use their service and invest money accordingly in campaign so we have done something similarly in this project for Arvato company .This capstone project is about the machine learning technique used to identify customer segment to which the mail must be sent in order to get

maximum probablity of customer getting turned towards company mail.

Following steps are done in this project:

- Data Exploration and Cleaning.
- Data Vizualization.
- Data Preprocessing.
- Supervised Learning -model selection
- Model Evaluation and validation.
- Model testing/Kagle Competition.

Dataset Used:

Source of data set: Provided by Arvato company for completing the project Identifying the customer segmentation

Note: This data set should be used for this project and cannot be available to any other individual

Below are the Data sets given in this project:

AZDIAS Data set: Demographics data for the general population of Germany.(891,211 customers x 366 features)

CUSTOMER Data set: Demographics data for customers of a mail-order company.(191,652 customers x 366 features)

Mailout_train data set: Demographics data for individuals who were targets of a marketing campaign(Training data set that were send to the population of Germany)(42,982 customers x 367 features)

Mailout_test data set: Testing data set that were send to the population of Germany(top-level list of attributes and descriptions, organized by informational category)(42,982 customers x 366 features))

Attributes 2017.xlsx : In Depth information of the attributes.

Values 2017.xlsx: In depth explanation of the feature attribute. (detailed mapping of data values for each feature in alphabetical order)

Azdias and customer data set should be used for unsupervised learning of this project while mailout_train

data for making model for supervised section and then mailout_test data should be used to test your model finally.

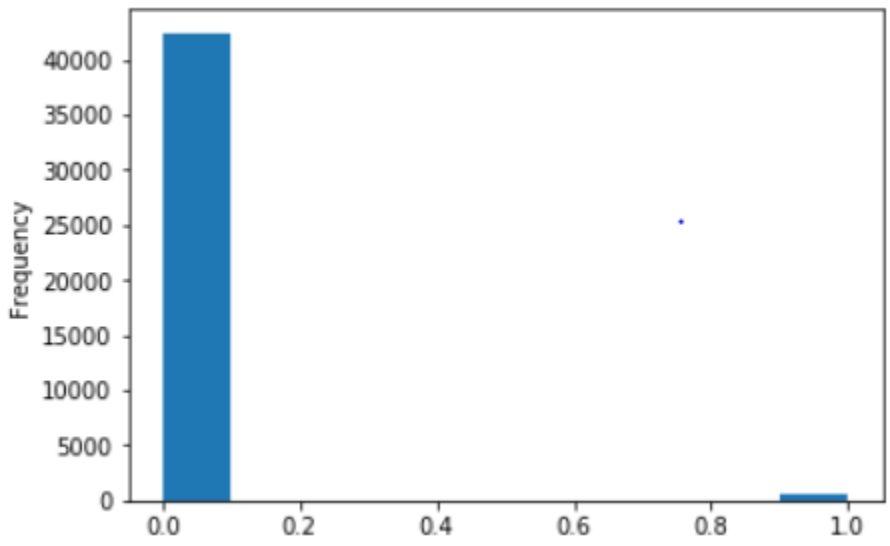
Descriptive statistics of the data set used is:

	ANZ_HAUSHALTE_AKTIV	ANZ_PERSONEN	ANZ_TITEL	ARBEIT	BALLRAUM	EWDICHTE	FINANZ_ANLEGER	FINANZ_HAUSBAUER
count	798032.000000	798032.000000	798032.000000	798032.000000	798032.000000	798032.000000	798032.000000	798032.000000
mean	8.295524	1.728854	0.004161	3.172117	4.154381	3.940747	2.840963	3.114160
std	15.624266	1.156546	0.068889	1.001564	2.183489	1.719264	1.472787	1.408089
min	1.000000	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	1.000000	1.000000	0.000000	3.000000	2.000000	2.000000	1.000000	2.000000
50%	4.000000	1.000000	0.000000	3.000000	5.000000	4.000000	3.000000	3.000000
75%	9.000000	2.000000	0.000000	4.000000	6.000000	6.000000	4.000000	4.000000
max	595.000000	45.000000	6.000000	9.000000	7.000000	6.000000	5.000000	5.000000

8 rows × 160 columns

These are some of the attributes shown in the azdias dataset same attributes are present in the other data sets.

The target response distribution is givent below in the form of Histogram plot:



This is the distribution of people which have responded/not responded to the mail earlier sent by the company. *By training our model with the given features in* **mail_out train** *data set with the* **response** *column in it will help us to build a good model*

Problem Statement

Statement:

Making A Machine learning model to predict the customer segment of the population of Germany to which the mail must be sent in order to target the maximum response from the customer using selective mails thereby decreasing the cost in campaign and increasing profit in the business. Also that services are provided to the customer segment who really needs them.

Evaluation Metrics

Kmeans was used to in unsupervised section of the project to determine the minimum cluster required so that it shows a variability of at least 60 percent.

ROC_AUC was used by me in the supervised section of the arvato project.

Justification: These were used because the datset provided was imbalanced and therefore we used ROC to get us the best accuracy scoring. ROC AUC iterate very beautifully between the sensitivity and also the specificity for each and every threshold value i.e choosing best combination of both sensitivity and specificity ,which makes it a better metrics than other for this data sets.

Solution Statement

In order to solve this project following step should be done:

-We'll use unsupervised learning techniques to perform customer segmentation in order to find the section of the population which must be targeted by company.

- Then, by applying supervised technique on the company mailout train dataset in order to get the model ready for making prediction about the likeliness of the customer to respond to mail campaign.

- Then we'll test the model made in supervised section to predict the probability of the customer to respond to the mail.

Benchmark Model

Different Classifier will be used like Adaboost Classifier, Gradientboost, logisticregression and Randomforest classifier for modelling of the given data.

GradientBoostClassifier were chosen to be the benchmark model for this type of data set as it has a great historical relevance in Ml industry.

Steps Involved Throughout the project are listed below:

Preprocessing

Data given were explored of its null values using graphs and a cleaning function was defined to clean all the data set given in the project. Below is the Function.

Cleaning Code:

```
def clean_data(azdiasd):
    """
    Perform feature trimming, re-encoding, and engineering for
demographics
    data

    INPUT: Demographics DataFrame
    OUTPUT: Trimmed and cleaned demographics DataFrame
    """
```

```
    # Load in the general demographics data.
    #azdiasd = pd.read_csv('customers.csv')
    #azdiasd = azdiasd.drop('Unnamed: 0',axis=1)
# Load in the feature summary file.
    feat_infod = pd.read_csv('AZDIAS_Feature_Summary.csv',sep=";")


    for i,j in enumerate(azdiasd.iteritems()):
            missing_unknown = feat_infod['missing_or_unknown'][i]
            column_name_miss = j[0]
            missing_unknown = missing_unknown[1:-1].split(',')
            if missing_unknown != ['']:
                replaced = []
                for k in missing_unknown:
                    if k in ['X','XX']:
                        replaced.append(k)
                    else:
                        replaced.append(int(k))
                azdiasd[column_name_miss] =
azdiasd[column_name_miss].replace(replaced,np.nan)



    row_miss_transpose= azdiasd.isnull().transpose()
    row_miss=row_miss_transpose.sum()
    azdias_lowd = azdiasd[row_miss<26]
    azdias_highd = azdiasd[row_miss>=26];


# Assess categorical variables: which are binary, which are multi-
level, and
# which one needs to be re-encoded?

categor=feat_infod[feat_infod['type']=='categorical'].attribute.valu
es
    category=[]
    for cat in categor:
        if cat in azdiasd.columns:
            category.append(cat)
    binary_one,binary_two,multi = [],[],[]
    for x in category:
        if azdiasd[x].nunique()==1:
            binary_one.append(x)
        elif azdiasd[x].nunique()==2:
            binary_two.append(x)
```

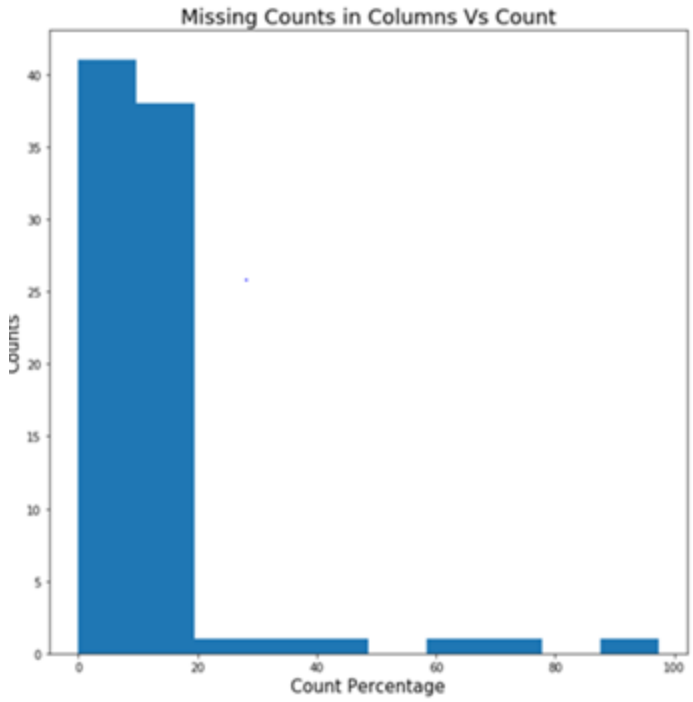
```
        elif azdiasd[x].nunique() >2:
            multi.append(x)

#changing the label as per mentioned above in the markdown cell
    azdias_lowd['OST_WEST_KZ'] =
azdias_lowd['OST_WEST_KZ'].map({'O':0,'W':1})

# Removing the attribute 'CAMEO_DEU_2015' as it contain lots of
variable
    azdias_lowd.drop('CAMEO_DEU_2015',axis=1,inplace=True)# using
drop method

    multi.remove('CAMEO_DEU_2015')#removing the cameo_deu_2015column
name from the list

# Re-encode categorical variable(s) to be kept in the analysis.
    for each in multi:
        dummy_variable =
pd.get_dummies(azdias_lowd[each][azdias_lowd[each].notnull()],prefix
=each)#creating dummies for each colum im multivariable
        #azdias_lowd = azdias_lowd.join(dummy_variable)
        pd.concat([azdias_lowd,dummy_variable],axis=1)
        azdias_lowd=azdias_lowd.drop(each,axis=1)#dropping the
column from our testing data i.e azdias_low


    mix =
feat_infod[feat_infod['type']=='mixed']['attribute'].values
    mixed = []
    for x in mix:
        if x in azdias_lowd.columns:
            mixed.append(x)


# Investigate "PRAEGENDE_JUGENDJAHRE" and engineer two new
variables.


    def avat(data_x):
        main = [1, 3, 5, 8, 10, 12, 14]
        avant = [2, 4, 6, 7, 9, 11, 13, 15]
        avat=[]
        for x in data_x['PRAEGENDE_JUGENDJAHRE']:

            if x in main:
```

```
                avat.append(0)
            elif x in avant:
                avat.append(1)
            else:
                avat.append(np.nan)

        frame = pd.DataFrame(avat)
        pd.concat([data_x,frame],axis=1)

        dicts =
{1:1,2:1,3:2,4:2,5:3,6:3,7:3,8:4,9:4,10:5,11:5,12:5,13:5,14:6,15:6}
        data_x['dekade'] =
data_x['PRAEGENDE_JUGENDJAHRE'].map(dicts)

        return data_x

    azdias_lowd = avat(azdias_lowd)

    feat_infod.missing_or_unknown =
feat_infod.missing_or_unknown.replace('[-1,XX]', "[-1,'XX']")
    feat_infod.missing_or_unknown =
feat_infod.missing_or_unknown.replace('[XX]', "['XX']")
    feat_infod.missing_or_unknown =
feat_infod.missing_or_unknown.replace('[-1,X]', "[-1,'X']")
    from ast import literal_eval
# Convert string representation to a list
    feat_infod.missing_or_unknown =
feat_infod.missing_or_unknown.apply(literal_eval)
    # Investigate "CAMEO_INTL_2015" and engineer two new variables.
    def cameo(data_y):
        CAMEO_INTL_2015_NEW = []

data_y['CAMEO_INTL_2015']=data_y['CAMEO_INTL_2015'].astype(float)
        for x in data_y['CAMEO_INTL_2015']:
            if x<=20:
                CAMEO_INTL_2015_NEW.append(1)
            elif x<=30:
                CAMEO_INTL_2015_NEW.append(2)
            elif x<=40:
                CAMEO_INTL_2015_NEW.append(3)
            elif x<=50:
                CAMEO_INTL_2015_NEW.append(4)
            elif x<=60:
                CAMEO_INTL_2015_NEW.append(5)
            else:
```

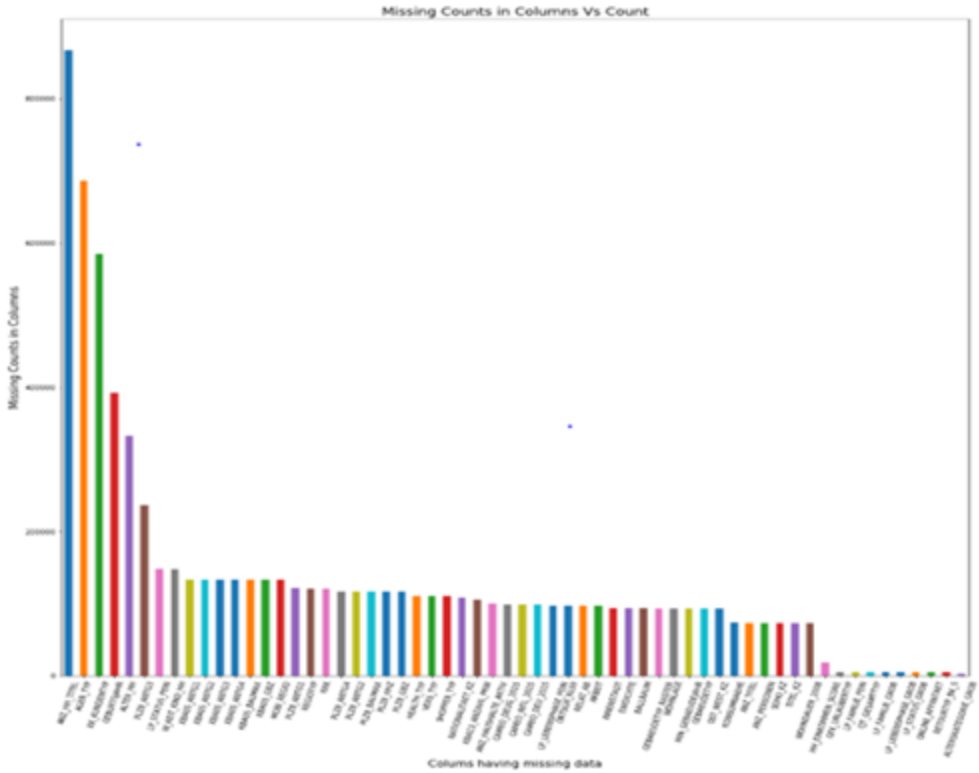
```
                CAMEO_INTL_2015_NEW.append(np.nan)
        frame2 = pd.DataFrame(CAMEO_INTL_2015_NEW)
        pd.concat([data_y,frame2],axis=1)

        return data_y

    #azdias_lowd=cameo(azdias_lowd)


#dropping all other columns which is not to be used in analysis in
futher proceedings


    azdias_lowd.drop(mixed,axis=1,inplace=True)
    mul = ['ALTER_HH', 'GEBURTSJAHR','ANZ_HH_TITEL', 'PLZ8_ANTG3']
    azdias_lowd.drop(mul,axis=1,inplace=True)
    columns = [col for col in azdias_low if col not in azdias_lowd]
    for column in columns:
        azdias_lowd[column] = 0

    return azdias_lowd
```

Missing data in Columns and Across Rows:

Analysis were done on the missing values in rows and columns and below is the visualization for the percentage of missing values in column. The outliers were columns having more than 40 percentage.



The no. of the missing value in each columns can be shown below:




```
percentage of rows having missing data is 39.26972097829831
```

Feature Encoding and Transformation

For categorical data, we would ordinarily need to encode the levels as dummy variables. Depending on the number of categories, perform one of the following:

- For binary (two-level) categorical that take numeric values, we can keep them without needing to do anything.

- There is one binary variable that takes on non-numeric values. For this one, we need to re-encode the values as numbers or create a dummy variable.
- For multi-level categorical (three or more values), we can choose to encode the values using multiple dummy variables (e.g. via OneHotEncoder), or (to keep things straightforward) just drop them from the analysis.
- CAMEO_DEU_2015 attribute having 44 variable seems to be useless according to me as it will contribute very less in prediction so i decided to remove it.
- Now then i have checked every attribute having binary value which take two value and found that attribute 'OST_WEST_KZ' needs mapping from 'O' to 0 and 'W' to 1.
- Then one hotcoder was used to deal with the multi variable attributes using get_dummies available from the pandas library.
- PRAEGENDE_JUGENDJAHRE column was investigated and two new columns were made for avantagram and mainstram after looking into the Data_dictionaty.md file.
- also one more columns i.e 'DEKADE' was added based on the years.

- For CAMEO_INTL_2015 the columns were mapped with 5 values based on the wealth of the person.
- For other mixed-type features (‘LP_LEBENSPHASE_FEIN’,’LP_LEBENSPHASE_GROB’EBENSPHASE_FEIN’, ‘WOHNLAGE’, ‘PLZ8_BAUMAX’]) were dropped on the basis that the information in these features were already included in other features in the dataset, like “ALTERSKATEGORIE_GROB”,”LP_FAMILIE_FEIN” and “LP_STATUS_FEIN” so of. And therefore according to me it wasn’t necessary to re-engineer and keep them in the dataset.

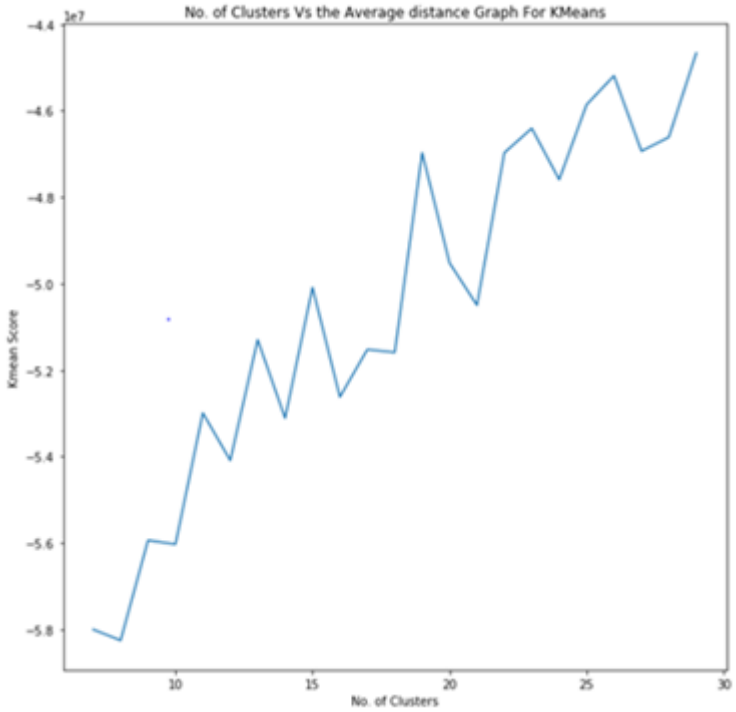
Reducing Dimensionality

PCA were used to reduce the dimensionality of the data set from very large features it was successfully reduced to 41 features. Using 41 components, we can explain 60.37% of the variability in the original data.

```
# Apply PCA to the data.

def do_pca(n_components, data):
    '''
    Transforms data using PCA to create n_components, and provides back the results of the
    transformation.

    INPUT: n_components - int - the number of principal components to create
           data - the data you would like to transform

    OUTPUT: pca - the pca object created after fitting the data
            X_pca - the transformed X matrix with new number of components
    '''

    pca = PCA(n_components)
    data_pca = pca.fit_transform(data)
    return pca, data_pca
```

K Mean Cluster

K means were used to further found the no. of cluster to segment the population after seeing the elbow graph.



Algorithm and Technique

After The unsupervised Learning done on the demographic data for the whole population and company customer mail. The supervised learning is to be performed and for that we have decided to iterate between Three main classifier which is know for dealing with imbalanced data very effectively. Those are listed below:

· Gradient Boost Classifier

- **Strength** :
  - It builds new trees which complement the already built trees. The new trees which will be built will help to correct errors in the previously built trees. This can produce highly accurate results with less trees.
  - Can handle different types of predictor variables and accommodate missing data.
- **Weakness** :
  - Unable to compute conditional class probabilities
  - Suffers from long sequential computation times.
  - More parameters to tune.

· Random Forest Classifier.

- **Strength** :
  - Scale quickly, have ability to deal with unbalanced and missing data
  - Generates an internal unbiased estimate of generalization error as forest building progresses.
  - Provides an expeimental way to detect variable interactions.

- **Weakness** :
  - Less effective on noisier-larger datasets with overlapping classes.
  - large number of trees may lead to slow real-time prediction in some cases.

Based on the Roc Score the Gradient Boost Classifier tops among the other two classifier and was therefore used further to predict the mailout test data set. The parameter was tuned to improve the roc score of the classifier using grid searchCV.

```
def classifier(clf, param_grid, X=train_feat_sc, y=train_response):
    """
    Fits a classifier to its training data using GridSearchCV and calculates ROC AUC score

    INPUT:
    - clf (classifier): classifier to fit
    - param_grid (dict): classifier parameters used with GridSearchCV
    - X_train (DataFrame): training input
    - y_train (DataFrame): training output

    OUTPUT:
    - classifier: input classifier fitted to the training data
    """

    # cv uses StratifiedKFold
    # scoring roc_auc available as parameter

    grid_search = GridSearchCV(estimator=clf, param_grid=param_grid, scoring='roc_auc',cv=5, verbose=0
    print("Training {} and below are the outputs :".format(clf.__class__.__name__))
    grid_search.fit(X, y)


|


    print("Best score : {}".format(round(grid_search.best_score_,2)))
    print("best eastimator: {}".format(grid_search.best_estimator_))
    print("-"*35)

    return grid_search.best_score_, grid_search.best_estimator_
```

Initial best Parameters

```
best eastimator:
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              presort='auto', random_state=42, subsample=1.0,
```

```
verbose=0,
              warm_start=False)
```

Note: The Roc Score for the model was found to be 56 since the number of features used in this project was 85 originally in the Azdas attribute_summary.csv file.

Algorithm & technique Improvement:

Model You'll have access to a third dataset with attributes from targets of a mail order campaign. We'll use the previous analysis to build a machine learning model that predicts whether or not each individual will respond to the campaign.

Model Tuning Parameters

```
param_grid = {'loss': ['exponential'],
              'learning_rate': [0.01,0.1],
              'n_estimators': [100],
              'max_depth': [3,5],
              'min_samples_split': [2]
             }

grad = GradientBoostingClassifier(random_state=42)
grad_best_score, grad_best_estimator= classifier(grad, param_grid)
grad_best_estimator
```

Final Parameters

```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.01, loss='exponential', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
```

```
              min_weight_fraction_leaf=0.0, n_estimators=100,
              presort='auto', random_state=42, subsample=1.0,
verbose=0,
              warm_start=False)
```

Important feature was fount to be :

MIN_GEBAEUDEJAHR: year the building was first mentioned in our database with weightage of 0.207

	features_weight
MIN_GEBAEUDEJAHR	0.206923
KBA13_ANZAHL_PKW	0.193731
ANZ_PERSONEN	0.135676
SEMIO_DOM	0.091119
dekade	0.067009

Kaggle submisstion :

In order to participate in this competition we have to submit a csv file in which there is user and the corresponding probability associated with it.

	User	RESPONSE
32491	82622	0.022628
6205	2425	0.022589
8851	30209	0.022464
19338	44603	0.022464
13201	72762	0.022464

Compare Customer Data to Demographics Data

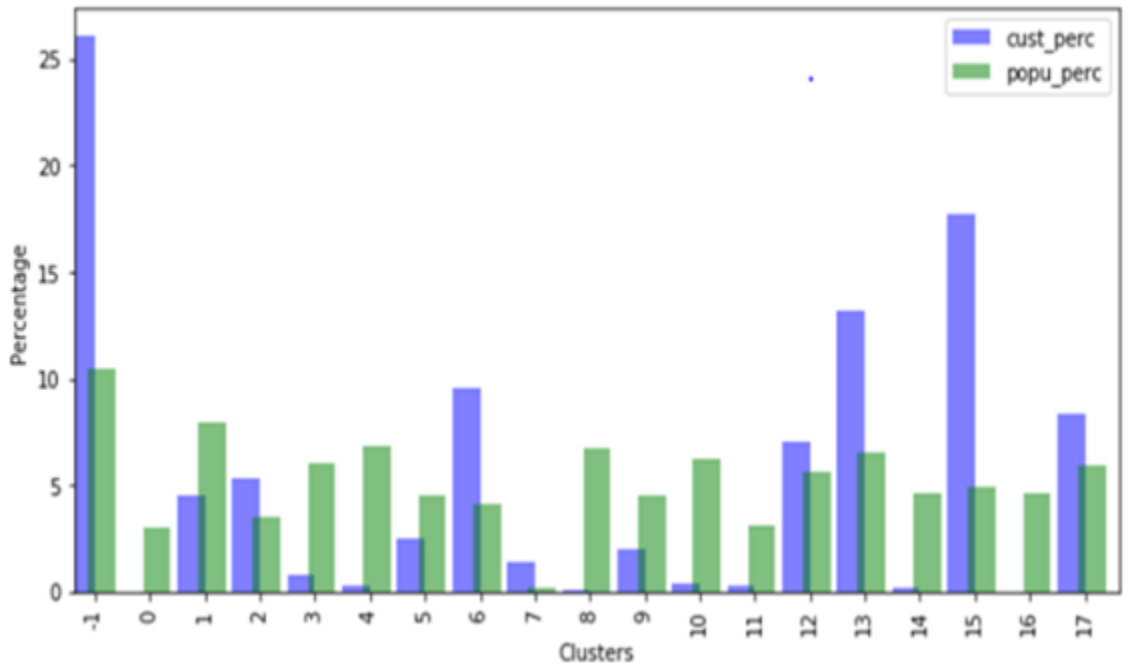
The population segment contains more missing data than the customer segment so results might get affected.

Segments of the population that are relatively popular with the mail-order company are of type:

- 46–60 years old
- having low financial interest
- money saver
- sanitary affine
- person having combative attitude

Segment relatively unpopular with the company:

- Relatively young age
- investors types
- inconspicuous
- high financial interest



Result and Improvement

Result:

The project was done successfully has given by the udacity team at the company the azdias and customer dataset was

preprocessed and relation between the customer and population was found then using the mailout_train dataset the supervised learning was performed successfully and **gradient boost classifier was chosen** with ROC Best Score.

- Gradient Boost Classifier: Score: 0.56
- AdaBoost Classifier: Score: 0.51
- Random Forest Classifier: 0.54

Note: The data columns feature used were only 85 that is why the roc score was less this is due to the availability of only 85 features attribute csv file.

Parameters iterated were:

```
param_grid = {'loss': ['deviance', 'exponential'],
              'learning_rate': [0.01, 0.1, 0.5],
              'n_estimators': [50, 80, 100],
              'max_depth': [2, 3, 4, 6, 8],
              'min_samples_split': [2,3,4]

             }
```

Following were the **best values of the hyperparameter** that cross-validation found:

```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.01, loss='exponential', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
```

```
              min_weight_fraction_leaf=0.0, n_estimators=100,
              presort='auto', random_state=42, subsample=1.0,
verbose=0,
              warm_start=False)
```

Imrovement:

- the model can be improved by tuning the parameter of the gradient Boost Classifier using GridSearchCV.
- Also by increasing the PCA components we can improve the model.

Base Line Model	Gradient Boost Classifier (Untuned)	Gradient Boost Classifier(tuned)
Metric used		
Roc Score	0.54	0.56