

## *Targeted Campaigning, Arvato Financial Solutions*

***Rohan Digambar Gawade***

Udacity Machine Learning Nanodegree Capstone Project Report



Customer Segmentation and Target Marketing

### **Introduction**

Bertelsmann, Arvato Financial Solutions client, the mail-order company, wanted to have an efficient way for targeting the customers for their mailing campaign. The project caught my eye as it wanted to provide a solution to the real world data science problem and also covered most of the machine learning concepts, needing to apply them on a real world data.

The two problems I tried solving here are:

1. **Customer Segmentation**, where using demographic information of the general population of Germany and the customer demographic information I tried representing them in cluster groups
2. **Supervised Learning**, where I created model to predict if the person will respond to mailing campaign or not.

The main goal here is to provide an opportunity for the client, the mail-order company to focus and allocate its precious resources efficiently on the potential customers rather than whole set of population.

## Data Cleaning

The data for both customer segmentation problem and supervised learning model problem was already provided along with some metadata files.

There were 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).

Along with these four files, the metadata information about the attributes were also given in two files. One of the file had following information, which I used a lot to clean and filter unwanted columns and interpret the values.

```
In [14]: df_attributes_val.head(10)
```

Out[14]:

	Attribute	Description	Value	Meaning
0	AGER_TYP	best-ager typology	-1	unknown
1	AGER_TYP	best-ager typology	0	no classification possible
2	AGER_TYP	best-ager typology	1	passive elderly
3	AGER_TYP	best-ager typology	2	cultural elderly
4	AGER_TYP	best-ager typology	3	experience-driven elderly
5	ALTERSKATEGORIE_GROB	age classification through prename analysis	-1, 0	unknown
6	ALTERSKATEGORIE_GROB	age classification through prename analysis	1	< 30 years
7	ALTERSKATEGORIE_GROB	age classification through prename analysis	2	30 - 45 years
8	ALTERSKATEGORIE_GROB	age classification through prename analysis	3	46 - 60 years
9	ALTERSKATEGORIE_GROB	age classification through prename analysis	4	> 60 years

Initially it was very difficult to interpret the columns but since most of columns had information in the metadata file and the meaning of their corresponding values, it became a bit easy to interpret. After doing some cleaning in the attributes names, I was able to bring down the unknown column information list from 97 to 60.

It seems D19 features have different name in the metadata files. Lets get it consistent. We will remove RZ from metadata attribute list

```
In [34]: df_attr_val_level['Attribute'] = df_attr_val_level['Attribute'] \
        .apply(lambda x: re.sub('_RZ', '', x) if 'D19' in x else x)

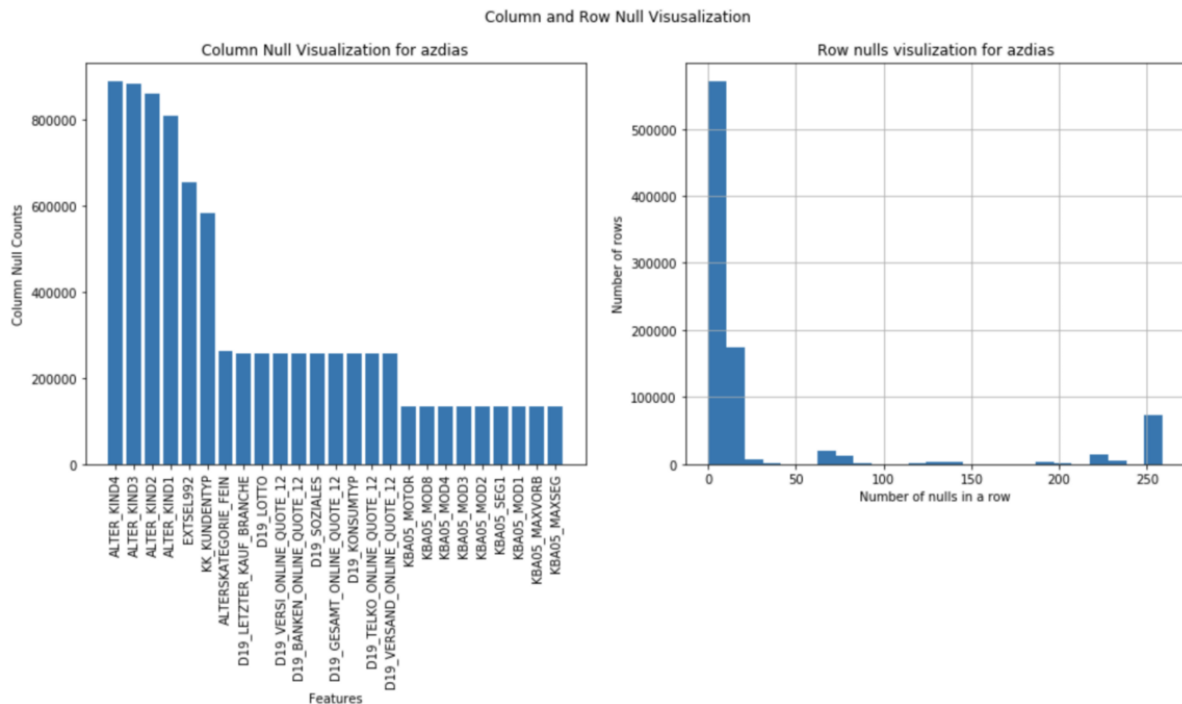
In [35]: len(set(customers.columns) - set(df_attr_val_level['Attribute']))

Out[35]: 60
```

## Visualizing nulls, cleaning and Imputing

```
j: visualize_nulls(azdias_copy, 'azdias')

visualize column and row nulls for azdias
```



**Imputation :** Once the columns were cleaned and had consistent values wrt to NaN, they were imputed using the most frequent value in a column for the whole dataset.

There was a catch here, as most of the columns had numeric data, but each numeric value represented a meaning to the attribute in the metadata. Imputing these values with mean wouldn't have made sense.

```
In [66]: azdias_copy.columns[azdias_copy.isnull().any()]
Out[66]: Index([], dtype='object')
```

```
In [67]: azdias_copy.describe()
```

```
Out[67]:
```

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PER
count	7.487460e+05	748746.000000	748746.000000	748746.000000	748746.000000	748746.000000	748746.000000	748746.000000	748746
mean	6.367552e+05	-0.293437	4.445350	10.819564	13.94974	8.509348	0.040948	0.144061	1
std	2.580217e+05	1.233744	3.650765	7.637613	4.51500	15.663501	0.318953	0.485245	1
min	1.916530e+05	-1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0
25%	4.131675e+05	-1.000000	1.000000	0.000000	12.00000	2.000000	0.000000	0.000000	1
50%	6.349305e+05	-1.000000	3.000000	13.000000	15.00000	4.000000	0.000000	0.000000	1
75%	8.617158e+05	1.000000	9.000000	17.000000	16.00000	10.000000	0.000000	0.000000	2
max	1.082872e+06	3.000000	9.000000	21.000000	25.00000	536.000000	20.000000	11.000000	45

Some categorical columns 'CAMEO\_DEU\_2015', 'OST\_WEST\_KZ' were one hot encoded using pandas get\_dummies function.

I also removed few columns, by keeping one of the two highly correlated features. (<https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf> ).

Now we have clean data which can be used for creating customer segmentation report.

## CUSTOMER SEGMENTATION

The unsupervised learning technique is used here to gain the understanding of relationship between the general population in Germany and the customer of mail-order company.

For using 'kmeans' it's important that the features are scaled. I used MinMax Scaler. Since we have values that represents a category it seems to be better to us MinMaxScaler as it will transform each value of column to range between 0 and 1. (<https://benalexkeen.com/feature-scaling-with-scikit-learn/>)

## PCA

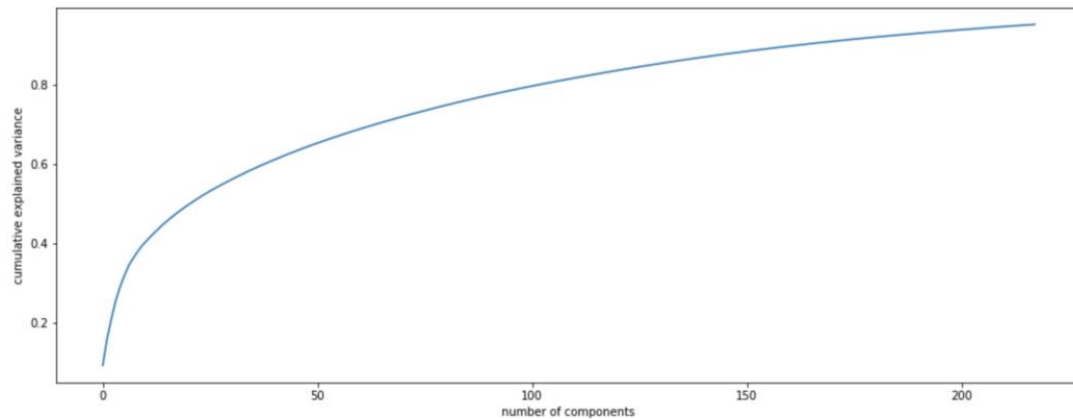
The next step is to reducing the features for which I used Principal Components Analysis. To explain the 95% of the data, PCA created 218 principal components based on general population which was then used to transform the customer dataset

```
In [89]: pca = PCA(.95)
azdias_pca = pca.fit(azdias_scaled)
```

```
In [90]: azdias_pca.n_components_
```

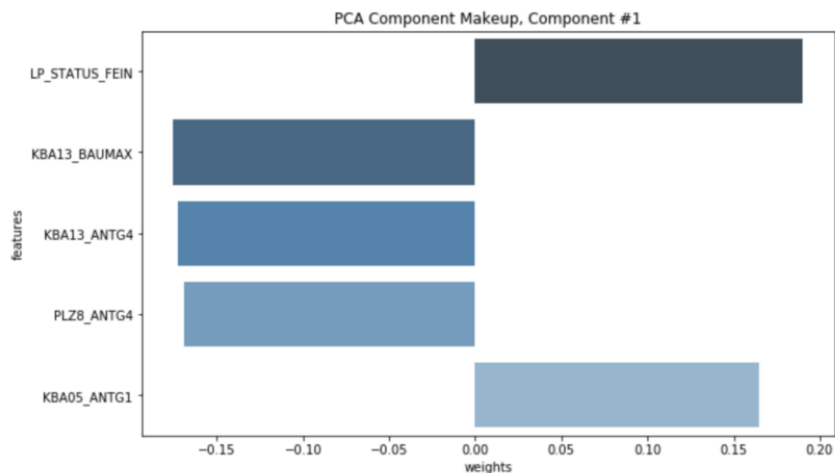
```
Out[90]: 218
```

```
In [91]: plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
```



The features are being transformed into weighted linear combination that explains the largest possible variability in the dataset.

```
In [93]: for num in range(1,6):
display_component(pca.components_, azdias_scaled.columns.values, component_num=num, n_weights=5)
```

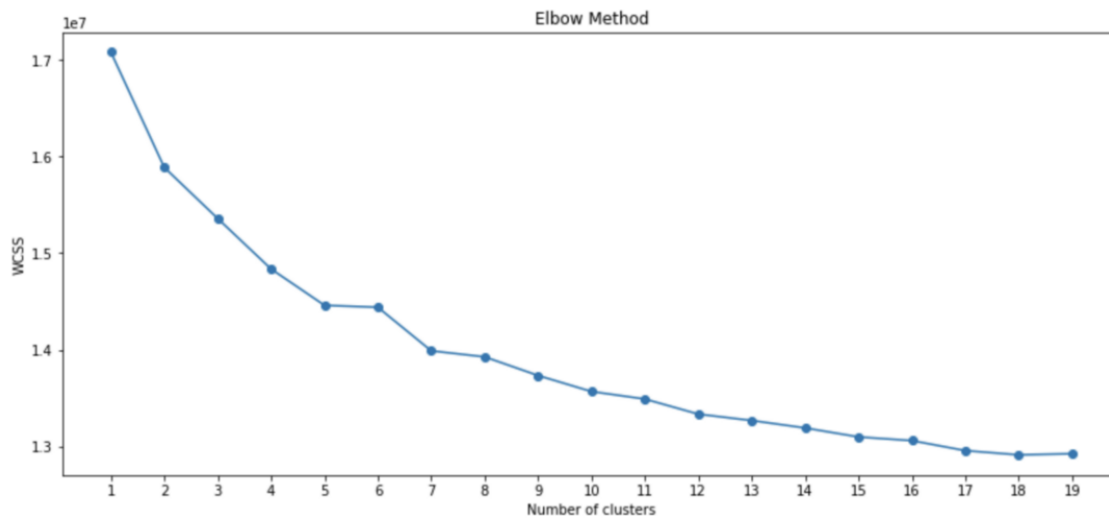


Please go through the notebook from the github link below for more components

## KMEANS

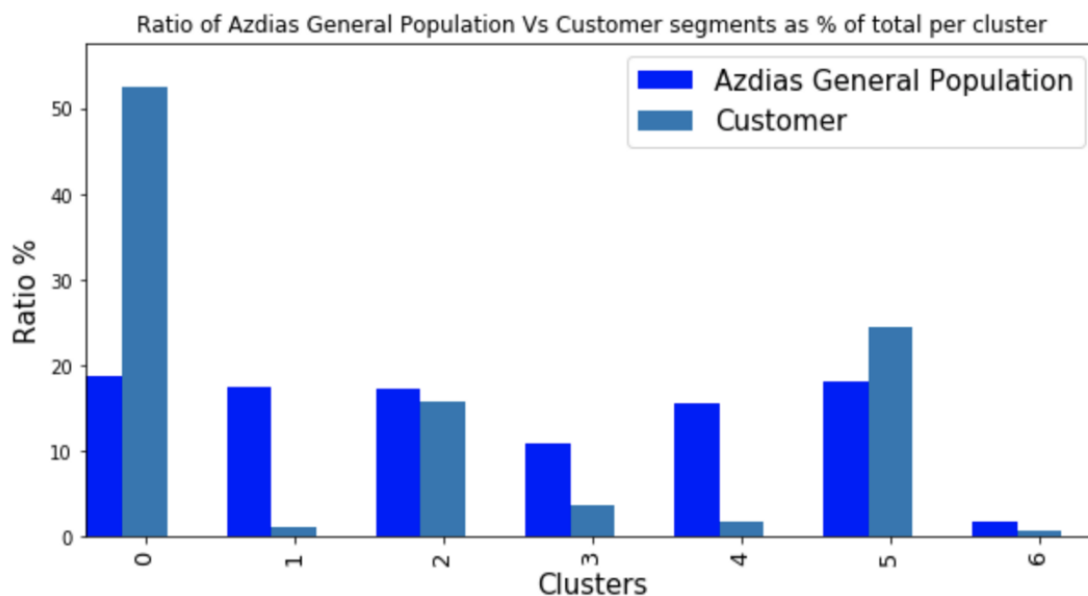
KMeans clustering is an unsupervised machine learning model, which helps in classifying/grouping data without having known labels. I used KMeans to cluster the general population data. Since we don't know what the number of clusters (k) is, I used elbow method to determine it. The within-cluster sum of squares value is plotted across different 'k' and we check

for the elbow like curve to be formed. The point from which the wcss value starts decreasing gradually, we consider it as 'k'



We want the within cluster sum of squares to be as small as possible as it measures the compactness of the clustering. We have a knee bend at 5 and 7. We will choose 7 as wcss starts to decrease slowly from this bend.

The two dataset are segmented as below:



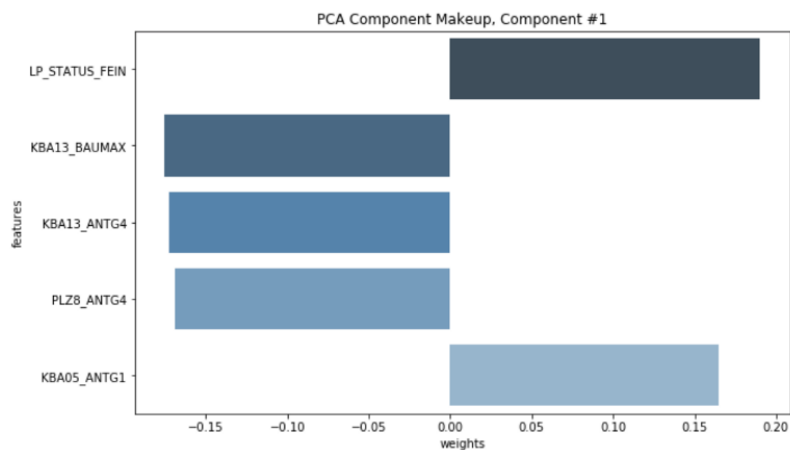
Customers in Clusters 0 and 5 over-represented the population while clusters 1,2, 3,4, 6 are under-representing the population.

This means the people in cluster 0 and 5 have certain set of features in the principal components which drives that people belonging to this clusters are more likely to be our customers.

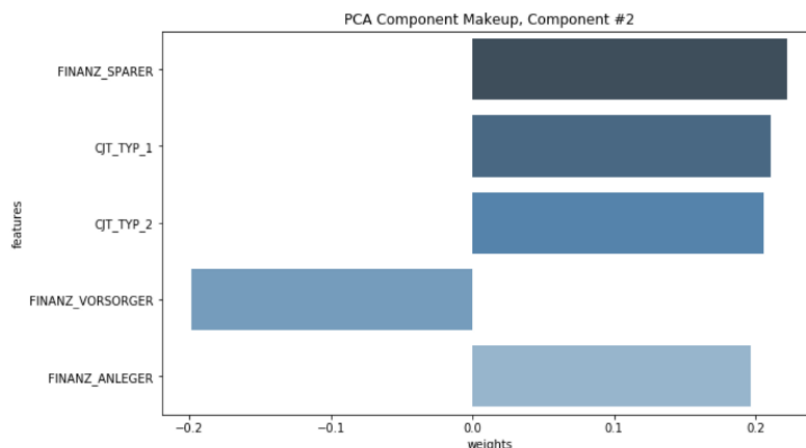
The top 5 positive and negative weight features that drive the cluster 0 are plotted as below:

```
In [113]: featuresImpactOnClusters(0)
```

First plot - Features from component with positive weight value for cluster 0



Second plot - Features from component with negative weight value for cluster 0



As we can see the customers in cluster 0 are driven by features like social status, number of family houses, financially prepared or investor. Few features didn't have information in metadata, hence was not able to describe.

```

In [118]: featureComponentValueMeaning(features_in_components_0)

LP_STATUS_FEIN - social status fine
Value Meaning ['typical low-income earners' 'orientationseeking low-income earners'
'aspiring low-income earners ' 'villagers'
'minimalistic high-income earners' 'independant workers'
'title holder-households' 'new houseowners' 'houseowners' 'top earners ']

PLZ8_ANTG4 - number of >10 family houses in the PLZ8
Value Meaning ['unknown' 'none' 'low share' 'high share']

KBA05_ANTG1 - number of 1-2 family houses in the cell
Value Meaning ['unknown' 'no 1-2 family homes' 'lower share of 1-2 family homes'
'average share of 1-2 family homes' 'high share of 1-2 family homes'
'very high share of 1-2 family homes']

FINANZ_SPARER - financial typology: money saver
Value Meaning ['unknown' 'very high' 'high' 'average' 'low' 'very low']

FINANZ_VORSORGER - financial typology: be prepared
Value Meaning ['unknown' 'very high' 'high' 'average' 'low' 'very low']

FINANZ_ANLEGER - financial typology: investor
Value Meaning ['unknown' 'very high' 'high' 'average' 'low' 'very low']

```

Though I was able to get the features driving a cluster, I actually wanted to know which values are actually driving this. Probably an improvement i can work on later!!!



## SUPERVISED MACHINE LEARNING MODEL

### *Data Preprocessing*

The Arvato Financial Solutions provided the train and test dataset for the mailout campaign for May 2018. The two files were

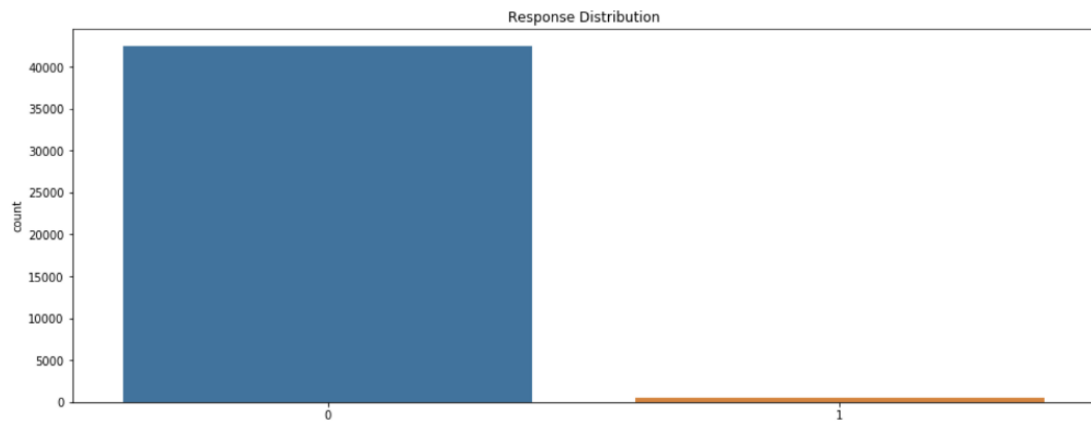
1. Udacity\_MAILOUT\_052018\_TRAIN.csv
2. Udacity\_MAILOUT\_052018\_TEST.csv

Each of the files have columns similar to general population and customer dataset. Here I used the same set of steps based on the analysis on the segmentation data for data cleansing.

The data is highly imbalanced in terms of RESPONSE variable.

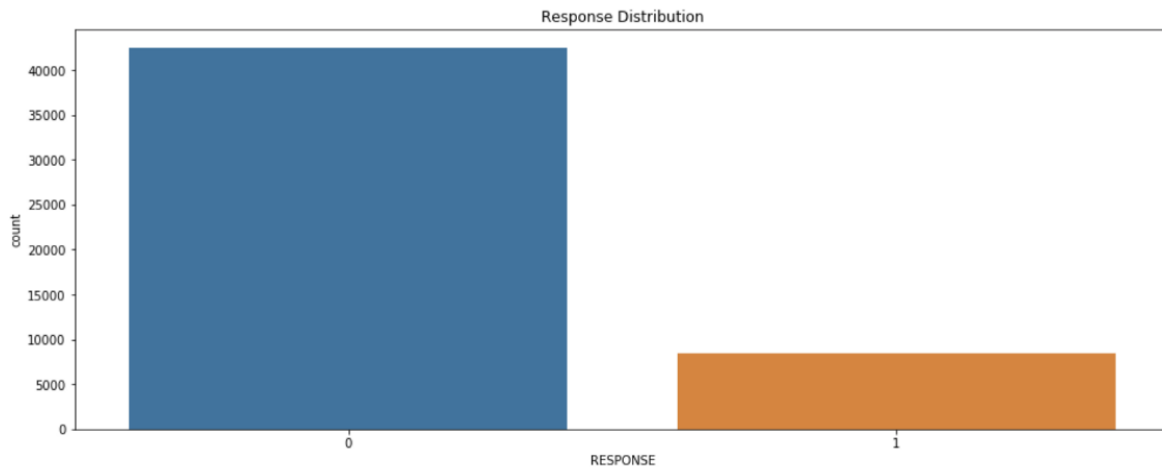
```
In [127]: mailout_train_copy = dataPreprocess(mailout_train_copy)
(42962, 402)
```

```
In [128]: sns.countplot(mailout_train_copy['RESPONSE']).set_title("Response Distribution");
```



As we can see, out of 42K + total data points, we hardly have 1% of responses. Here I used oversampling method .i.e Up Sampled the data points for responses = 1. I first oversampled the data with the ratio 1:1 which resulted in a highly overfitting model. So to have some more data points and also have a realistic scenario, the ratio for RESPONSE = 1 to RESPONSE = 0 was kept to 1:5.

```
: sns.countplot(y).set_title("Response Distribution");
```



### *Implementation*

Along with oversampling, the train data was divided into train and validation set ( I named it X\_test and y\_test) using train\_test\_split into a 80:20 split with stratified sampling.

On the X\_train, and y\_train dataset, I further did StratifiedKfold Cross validation while training the models.

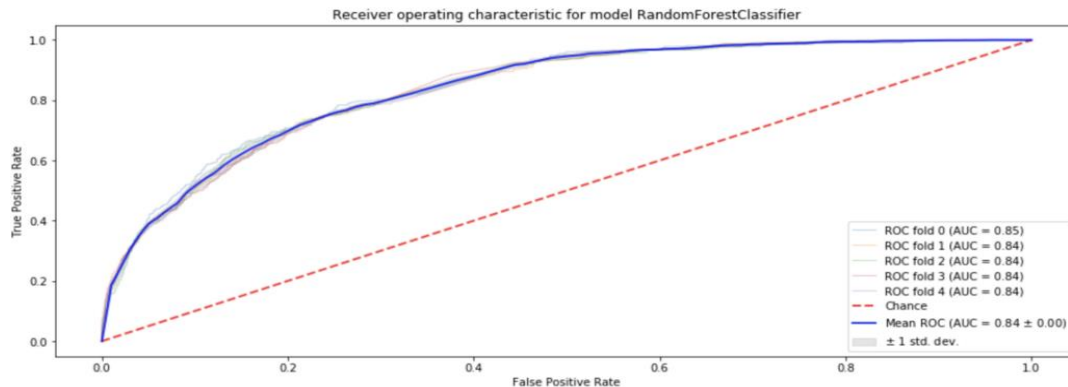
I am using Bagging and Boosting models, hence I wont be doing any scaling here.

Also since classes are imbalance we will be using ROC AUC to determine the perfomance of the model and not accuracy measure.

The model I used for benchmarking is RandomForestClassifier, where I was able to achieve mean ROC AUC of 0.84 on the StratifiedKfold train set.

```
In [141]: %time
rfc = RandomForestClassifier(max_depth=4,max_features=30,random_state=1)
rfc = train_and_plot_roc(rfc,'RandomForestClassifier')

RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=4, max_features=30,
                       max_leaf_nodes=None, max_samples=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,
                       n_jobs=None, oob_score=False, random_state=1, verbose=0,
                       warm_start=False)
```



CPU times: user 22.5 s, sys: 245 ms, total: 22.8 s  
 Wall time: 23 s

```
In [164]: roc_auc_score(y_test,rfc.predict(X_test))
```

Out[164]: 0.5

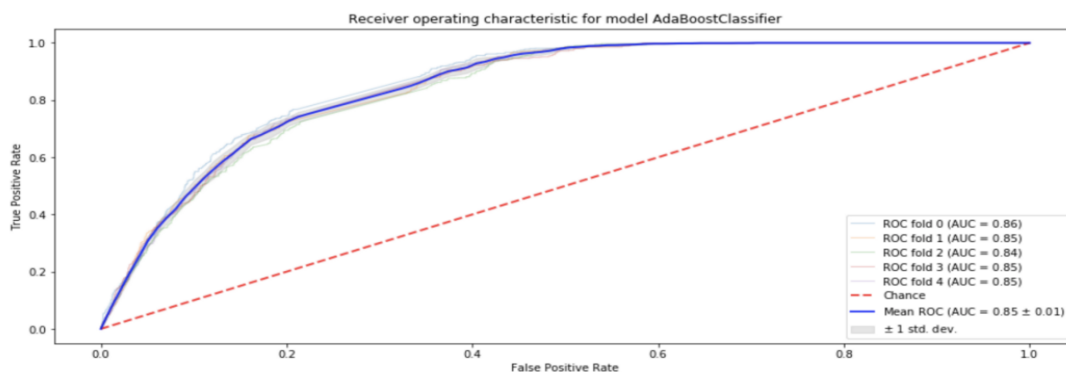
## Performance of RandomForestClassifier

But as you can see the ROC AUC on validation set is just 0.5

I then used AdaBoost and GradientBoosting Classifier

```
In [142]: %time
abc = AdaBoostClassifier(random_state=1,n_estimators=100)
abc = train_and_plot_roc(abc,'AdaBoostClassifier')

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
                  n_estimators=100, random_state=1)
```



CPU times: user 2min 18s, sys: 3.56 s, total: 2min 22s  
 Wall time: 2min 29s

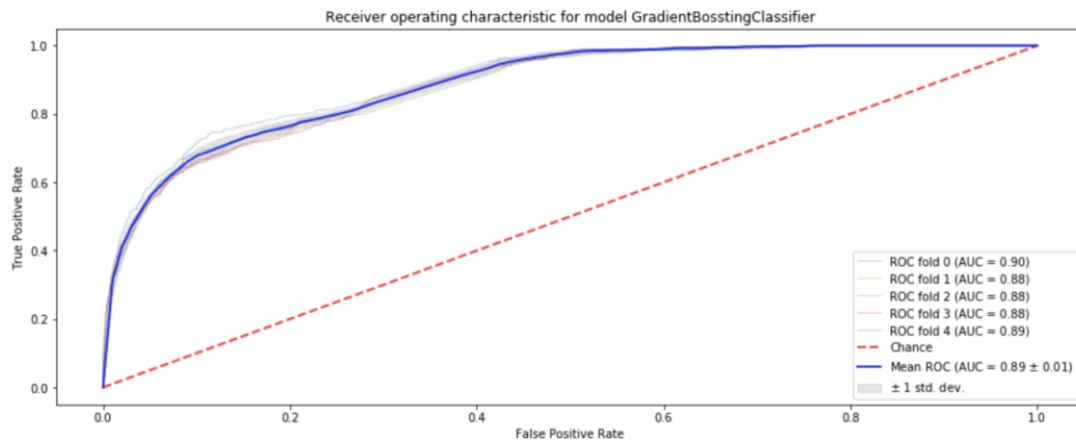
```
In [143]: roc_auc_score(y_test,abc.predict(X_test))
```

Out[143]: 0.6101405463338535

## Performance of AdaBoostClassifier

```
In [144]: %%time
gbc = GradientBoostingClassifier(max_features = 30, random_state = 1)
gbc = train_and_plot_roc(gbc, 'GradientBosstingClassifier')
```

```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=30, 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,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=1, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```



```
CPU times: user 28.4 s, sys: 447 ms, total: 28.8 s
Wall time: 29.3 s
```

```
In [145]: roc_auc_score(y_test, gbc.predict(X_test))
```

```
Out[145]: 0.5469063607906967
```

## Performance of GradientBoostingClassifier

Since the AdaBoost had better results on validation set, I decided to refine it.

The model refine for AdaBoost is done using GridSearchCV, which finds the best model based on the parameter grid passed to it.

I provided learning\_rate : 0.1,1.0 and n\_estimators [ 50,100,150] as param\_grid to grid search.

```

In [146]: param_grid = {'learning_rate': [0.1, 1.0],
                        'n_estimators': [50, 100, 150]}

grid = GridSearchCV(estimator=AdaBoostClassifier(), param_grid=param_grid, scoring='roc_auc', cv=5)

In [147]: grid.fit(X_train, y_train)

Out[147]: GridSearchCV(cv=5, error_score=nan,
                      estimator=AdaBoostClassifier(algorithm='SAMME.R',
                                                    base_estimator=None,
                                                    learning_rate=1.0, n_estimators=50,
                                                    random_state=None),
                      iid='deprecated', n_jobs=None,
                      param_grid={'learning_rate': [0.1, 1.0],
                                   'n_estimators': [50, 100, 150]},
                      pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                      scoring='roc_auc', verbose=0)

In [148]: grid.best_score_

Out[148]: 0.8662491072688321

In [149]: grid.best_estimator_

Out[149]: AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
                              n_estimators=150, random_state=None)

In [150]: roc_auc_score(y_test, grid.best_estimator_.predict(X_test))

Out[150]: 0.6233987521141383


```

Best Model

## KAGGLE PREDICTION

Performed similar data processing steps on test dataset and used the best model (AdaBoostClassifier) with learning\_rate = 1.0 and n\_estimators = 150.

The prediction csv file was submitted on kaggle and got the ROC AUC value of 0.719 :)


InClass Prediction Competition

## Udacity+Arvato: Identify Customer Segments

Apply machine learning techniques to predict customers using data provided by Arvato Financial Solutions.

191 teams · a year to go

Overview
Data
Notebooks
Discussion
Leaderboard
Rules
Team
My Submissions
Submit Predictions

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
kaggle_submission.csv	4 hours ago	0 seconds	0 seconds	0.71963

Complete

[Jump to your position on the leaderboard](#)

## CONCLUSION

There is scope for improvement to the model, but this model can be used to predict whether potential customers will respond to the campaign or not. i.e. If the model thinks they can respond to our campaign then there is no harm in investing our resources to that potential customer.

### *Reflection*

In this project, I attempted to solve the real world business problem, by creating a machine learning process right from analysing the data, cleaning, imputing, to applying feature reduction techniques like PCA and performing unsupervised learning. Then creating supervised Machine Learning model for predicting whether a potential customer will respond to the campaign and be our customer.

The problem was pretty interesting, but I felt the metadata information was incomplete, and since the field names were in German it was a bit difficult in beginning to interpret.

For using the model, I think there is still room for improvements. I will state those in improvements and future work.

s

1. The first improvement I would like to do is using one hot encoding on even the features whose values represent certain class of people. There are a lot of such ordinal features in the dataset, which are currently treated as numeric values.
2. Instead of omitting D19\_UNBEKANNT and D19\_BUCH\_CD we could have one hot encoded them. Probably could have some impact as they constitute a good number of data points
3. Using clustering information as features somehow could provide improvements. I think if the metadata would have been complete it would have helped to form clusters in a better way. The information level in metadata could play a significant role during clustering.

s

## References

<https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f>

[https://scikit-learn.org/stable/auto\\_examples/model\\_selection/plot\\_roc\\_crossval.html](https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc_crossval.html)

<https://towardsdatascience.com/choosing-the-right-encoding-method-label-vs-onehot-encoder-a4434493149b>