Machine Learning Engineer Nanodegree

Capstone Project Proposal

Bertlesmen-Arvato Customer Segementation Project

J. P. Bedran

1. PROJECT OVERVIEW

Machine learning techniques are widely used these days to target potential customers by companies across all fields. Technique such as clustering has proven to be extremely helpful in this area. Arvato Financial Solutions has paired up with Udacity for their nanodegree program to create a customer segmentation report for them and predict which individuals are most likely to become a customer of their company.

The data that I will use has been provided by Udacity's partners at Bertelsmann Arvato Analytics, and represents a real-life data science task.

There are four data files associated with this project:

1. **Udacity_AZDIAS_052018.csv**: Demographics data for the general population of Germany; 891 211 persons (rows) x 366 features (columns).
2. **Udacity_CUSTOMERS_052018.csv**: Demographics data for customers of a mail-order company; 191 652 persons (rows) x 369 features (columns).
3. **Udacity_MAILOUT_052018_TRAIN.csv**: Demographics data for individuals who were targets of a marketing campaign; 42 982 persons (rows) x 367 (columns).
4. **Udacity_MAILOUT_052018_TEST.csv**: Demographics data for individuals who were targets of a marketing campaign; 42 833 persons (rows) x 366 (columns).

.

2. PROBLEM STATEMENT

As stated by Timo Reis, our main goal with this project is to acquire more customers efficiently for our client utilizing our technical expertise and the datasets provided by Arvato and German govt.

I will start by analyzing the first file, AZDIAS, which contains demographics for the general population. This process involves the cleaning and pre-preparation of the data. I will do the same for the customers data, although with not the same level of analysis. In AZDIAS file I will perform customer segmentation with PCA and KMeans, clustering the data so I can identify segments of the population that are compatible with the mail-order company.

I will then utilize supervised learning techniques to predict which individuals are most likely to become a customer of the company. As a part of this I will try out various classifiers like Logistic Regression, Random Forest and Gradient Boosting, the Regressor and Classifier of GB. To aid in the analysis of finding the best model I will use Grid Search.

3. Evaluation Metrics

The evaluation metric for this problem that I have chosen is AUC for the ROC (receiver operating characteristic curve) curve.

- True Positive Rate: This is also known as recall and is defined as follows:

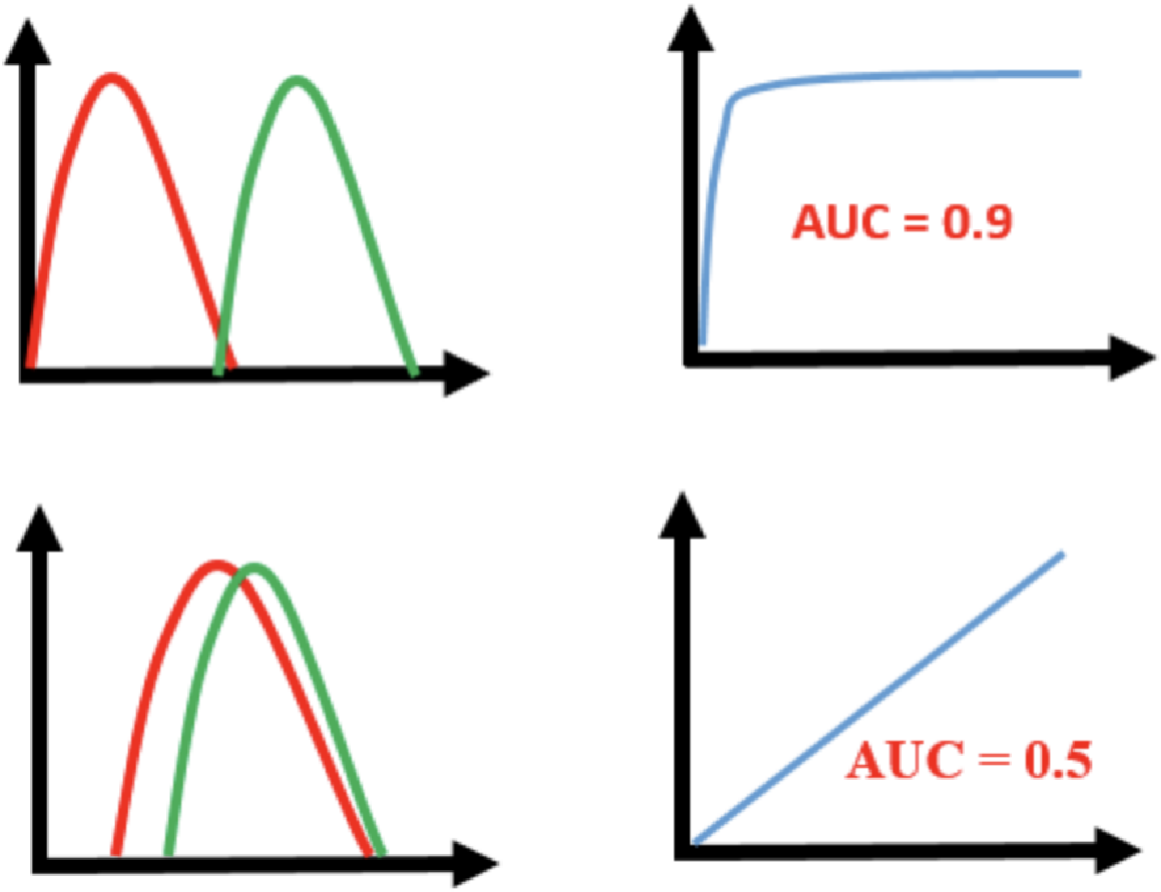
$$TPR = \frac{TP}{TP + FN}$$

- False Positive Rate: This is defined as follows:

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

The ROC curve plots TPR vs FPR at different points of classification thresholds. Lowering the classification threshold marks more item as positive which increases both False Positives and True Positives.

AUC is the area under the ROC curve. This gives us a good idea about the performance of the model. Let us take a few examples to see how the AUC varies according to the performance of the model.



The red distribution in the example above represents the individual who did not respond to the campaign and the green distribution represents the individuals who responded to the campaign. The first model does a good job of distinguishing the positive and negative values. Therefore the AUC score is high as the area under the ROC curve is high. The second model does not distinguish the values and as a result it has overlapping values and therefore AUC score is low.

3. DATA ANALYSIS AND PRE-PROCESSSING

I start by analyzing all the data I will be using through this project. The first part of da ta to analyze is the Udacity_AZDIAS file, which contains Demographics data for the general population. At first glance we can see that there are 33492923 missing values in AZDIAS.

```
In [4]: azdias.isnull().sum().sum()
Out[4]: 33492923
```

It will be necessary to clean the data and then use impute and Scaler to complete missi ng data after cleaned

```
In [6]: azdias.head()
```

Out[6]:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTI
0	910215	-1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1	910220	-1	9.0	0.0	NaN	NaN	NaN	NaN	21.0	11.(
2	910225	-1	9.0	17.0	NaN	NaN	NaN	NaN	17.0	10.(
3	910226	2	1.0	13.0	NaN	NaN	NaN	NaN	13.0	1.(
4	910241	-1	1.0	20.0	NaN	NaN	NaN	NaN	14.0	3.(

We can see that there are a lot of coded values that could represent missing or NaN. I n another step I will need to create a DataFrame that stores these values so I can transf orm them in AZDIAS to NaN.

Analyzing the 3 extra columns in CUSTOMERS, there are no demographic data, so I will go ahead and drop them.

```
In [5]: customers[['CUSTOMER_GROUP', 'ONLINE_PURCHASE', 'PRODUCT_GROUP']]
```

Out[5]:

	CUSTOMER_GROUP	ONLINE_PURCHASE	PRODUCT_GROUP
0	MULTI_BUYER	0	COSMETIC_AND_FOOD
1	SINGLE_BUYER	0	FOOD
2	MULTI_BUYER	0	COSMETIC_AND_FOOD
3	MULTI_BUYER	0	COSMETIC

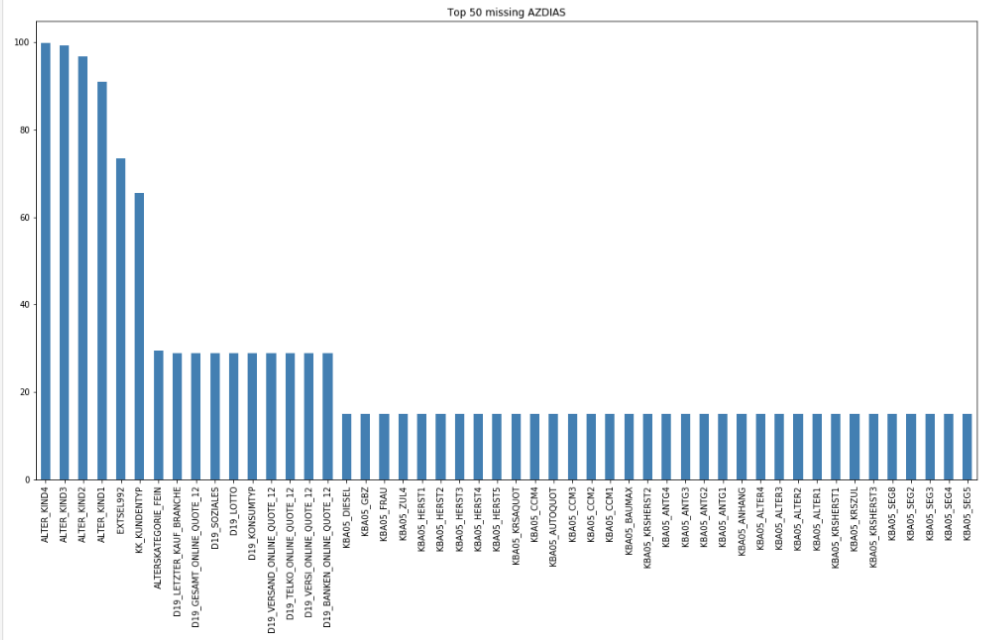
As previously discussed I will now create a DF with missing and unknown values fro m the excel file DIAS Attributes – Values 2017.xlsx

I will now evaluate by Column the missing data and sort it in descending order. This will help in visualizing which data to drop, making it clean but losing as little as possible as to maintain the highest possible proximity to the true data.

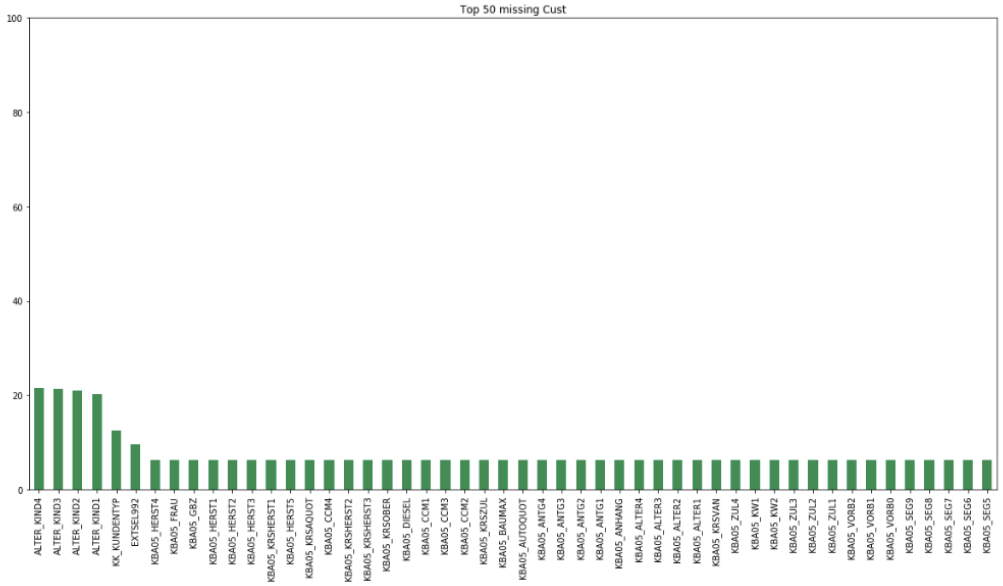
```
In [ ]: #function that returns % of missing data in Descending order
def missing_data(df):

    miss_data = df.isnull().sum()
    perc = np.round(miss_data*100/len(azdias),2)
    data_dict={
        'Total':miss_data.values,
        'Percentage':perc
    }

    missing_frame = pd.DataFrame(data=data_dict , index=miss_data.index)
    return missing_frame.sort_values(by='Total', ascending=False)
miss_az = missing_data(azdias)
miss_cust = missing_data(customers)
```

Upon first inspection we see that there are 16 columns in AZDIAS that have more than 20% of their values missing.

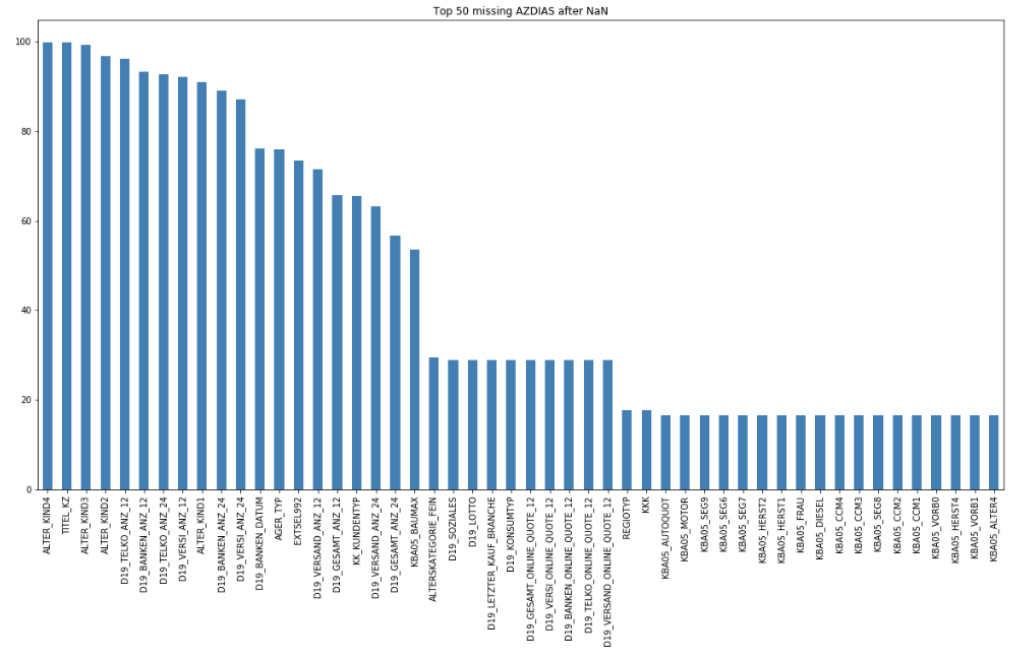
Out[48]: Text(0.5,1,'Top 50 missing Cust')

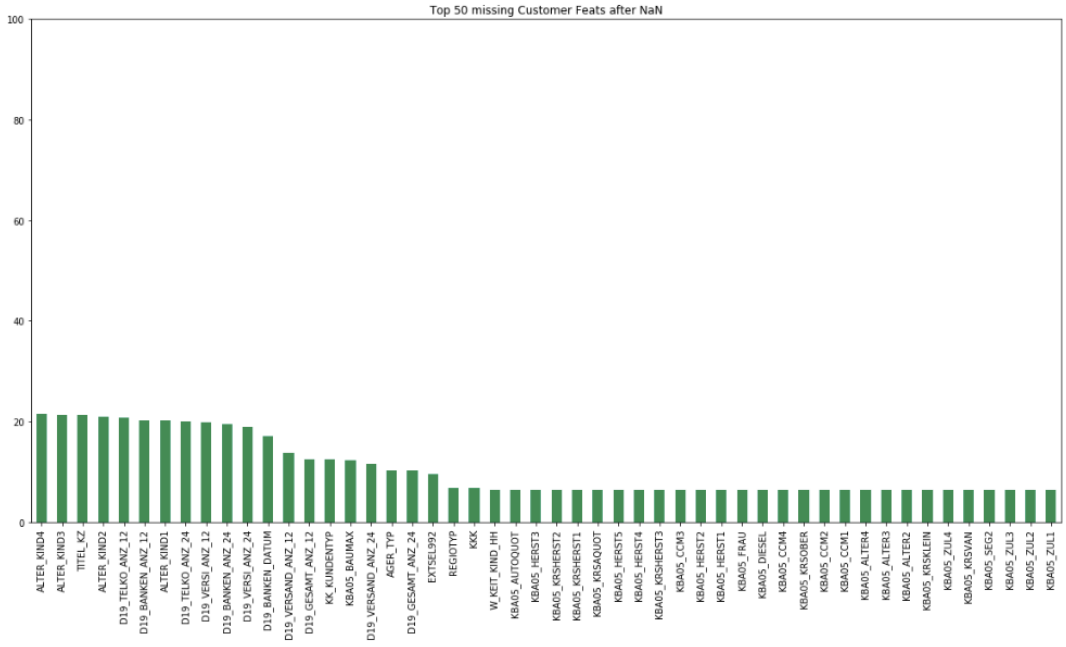


and in the CUSTOMERS we have only 4



After applying the missing_to_nan function, which takes values that are categorized as missing from the DataFrame previously created and transforms them to NaN, we can see a statistically significant increase in missing data in both AZDIAS and CUSTOMERS.





After data has been processed and missing values in both Data's are transformed to NaN, I can now evaluate any correlation between the Datasets to establish which ones should be dropped. Leaving strongly correlated Data in the set can cause Omitted Variable Bias in case of regressions in the last step of the project.

```
#identify which features to drop without causing OVB (Ommited Variable Bias)
strong_corr_az = az_corr.loc[over20_overall]

print(strong_corr_az)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:3: FutureWarning:
Passing list-likes to .loc or [] with any missing label will raise
KeyError in the future, you can use .reindex() as an alternative.

See the documentation here:
https://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate-loc-reindex-listlike
  This is separate from the ipykernel package so we can avoid doing imports until
```

	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	\
ALTER_KIND4	0.007744	0.026464	0.008581	0.477439	
ALTER_KIND3	-0.006071	0.011188	-0.053774	0.610869	
ALTER_KIND2	0.003560	-0.006282	-0.071733	0.781195	
ALTER_KIND1	0.026300	0.031251	-0.068980	1.000000	
EXTSEL992	0.172257	-0.169208	0.004846	-0.008019	
KK_KUNDENTYP	0.078280	0.020464	-0.127906	-0.007003	
ALTERSKATEGORIE_FEIN	-0.390886	0.306277	0.518291	-0.130236	
D19_LETZTER_KAUF_BRANCHE	NaN	NaN	NaN	NaN	
D19_GESAMT_ONLINE_QUOTE_12	-0.080315	-0.194122	0.228642	-0.017212	
D19_SOZIALES	0.168882	-0.153368	-0.007574	0.076138	
D19_LOTTO	0.132370	-0.145898	0.020901	0.095158	

In [ ]:

In [58]:
```
top_cust_corr = strong_corr_az.abs().unstack().sort_values(kind="quicksort", ascending=False).drop_duplicates()
top_cust_corr1 = pd.DataFrame(top_cust_corr, columns = ['Corr'])
```

In [59]:
```
over_80=top_cust_corr1[top_cust_corr1['Corr']>0.6]
print(over_80.index)
```

```
MultiIndex(levels=[['AGER_TYP', 'AKT_DAT_KL', 'ALTER_HH', 'ALTER_KIND1', 'ALTER_KIND2', 'ALTER_KIND3', 'ALTER_KIND4',
'ALTERSKATEGORIE_FEIN', 'ANZ_HAUSHALTE_AKTIV', 'ANZ_HH_TITEL', 'ANZ_KINDER', 'ANZ_PERSONEN', 'ANZ_STATISTISCHE_HAUSHA
LTE', 'ANZ_TITEL', 'ARBEIT', 'BALLRAUM', 'CJT_GESAMTTYP', 'CJT_KATALOGNUTZER', 'CJT_TYP_1', 'CJT_TYP_2', 'CJT_TYP_3',
'CJT_TYP_4', 'CJT_TYP_5', 'CJT_TYP_6', 'D19_BANKEN_ANZ_12', 'D19_BANKEN_ANZ_24', 'D19_BANKEN_DATUM', 'D19_BANKEN_DIRE
KT', 'D19_BANKEN_GROSS', 'D19_BANKEN_LOKAL', 'D19_BANKEN_OFFLINE_DATUM', 'D19_BANKEN_ONLINE_DATUM', 'D19_BANKEN_ONLIN
E_QUOTE_12', 'D19_BANKEN_REST', 'D19_BEKLEIDUNG_GEH', 'D19_BEKLEIDUNG_REST', 'D19_BILDUNG', 'D19_BIO_OEKO', 'D19_BUCH
_CD', 'D19_DIGIT_SERV', 'D19_DROGERIEARTIKEL', 'D19_ENERGIE', 'D19_FREIZEIT', 'D19_GARTEN', 'D19_GESAMT_ANZ_12', 'D19
_GESAMT_ANZ_24', 'D19_GESAMT_DATUM', 'D19_GESAMT_OFFLINE_DATUM', 'D19_GESAMT_ONLINE_DATUM', 'D19_GESAMT_ONLINE_QUOTE_
12', 'D19_HANDWERK', 'D19_HAUS_DEKO', 'D19_KINDERARTIKEL', 'D19_KONSUMTYP', 'D19_KONSUMTYP_MAX', 'D19_KOSMETIK', 'D19
_LEBENSMITTEL', 'D19_LOTTO', 'D19_NAHRUNGSERGAENZUNG', 'D19_RATGEBER', 'D19_REISEN', 'D19_SAMMELARTIKEL', 'D19_SCHUH
```

Correlation analysis indicates that any strong correlation of the features are mainly between the Strongly missing values Data Sets themselves, meaning I will choose to drop the features with over 20% missing values on both Data sets for optimal results. This totals to 20 features being dropped.

In [28]:
```
print(len(over20_overall))
```

```
20
```

I will create a random sample of the AZDIAS of 20% of the data as to be able to process PCA effectively and reduce processing time, while maintaining as much as possible the Originality of the data. As random samples are statistically comparable to original data, there should not be any divergence of results as compared to processing 100% of the data.

In [32]:
```
#create sample to do PCA
with active_session():

    def sampler(df):
        sample_df = df.sample(frac=0.2, random_state=42)
        return sample_df
    sample_az = sampler(clean_az)
```

To deal with the mixed typed columns, I will be applying a function that creates two new columns of information out of the mixed type column.

make_decade and make_movement will be applied to the PRAEGENDE_JUGENDJAHRE column

In [33]:
```
# Creating decade_dict and decade_list to be used in make_decade function.
decade_dict = {1: [1, 2], 2: [3, 4], 3: [5, 6, 7], 4: [8, 9], 5: [10, 11, 12, 13], 6:[14, 15]}
#decade_dict.items()
decade_dict.values()
decade_list = []
for dd in decade_dict.values():
    for i in dd:
        decade_list.append(i)

def make_decade(x):

    if pd.isnull(x):
        return np.nan
    else:
        for key, array in decade_dict.items():
            if x in array:
                return key
            elif x not in decade_list:
                print('There is some error while mapping decade. Please check.')


def make_movement(x):
    if pd.isnull(x):
        return np.nan
    elif x in (2,4,6,7,9,11,13,15):
        return 0
    elif x in (1,3,5,8,10,12,14):
        return 1
    else:
        print('There is some error while mapping movement. Please check.')
```

and make_wealth and make_life_stage will be applied to the CAMEO_INTL_2015 column.

After transforming features, I will be dropping object features which are not relevant to the data set.

```
sample_az[['CAMEO_DEU_2015', 'CAMEO_DEUG_2015', 'EINGEFUEGT_AM']]
```

	CAMEO_DEU_2015	CAMEO_DEUG_2015	EINGEFUEGT_AM
848815	9B	9	1992-02-12 00:00:00
299816	4C	4	1992-02-10 00:00:00
570748	4C	4	1994-08-30 00:00:00

In [39]:
```
sample_az.drop(['LP_LEBENSPHASE_FEIN','LP_LEBENSPHASE_GROB','WOHNLAGE','PLZ8_BAUMAX','PRAEGENDE_JUGENDJAHRE','CAMEO_INT
```

In [40]:
```
sample_az.drop(['CAMEO_DEUG_2015', 'CAMEO_DEU_2015', 'EINGEFUEGT_AM'], axis=1, inplace=True)
```

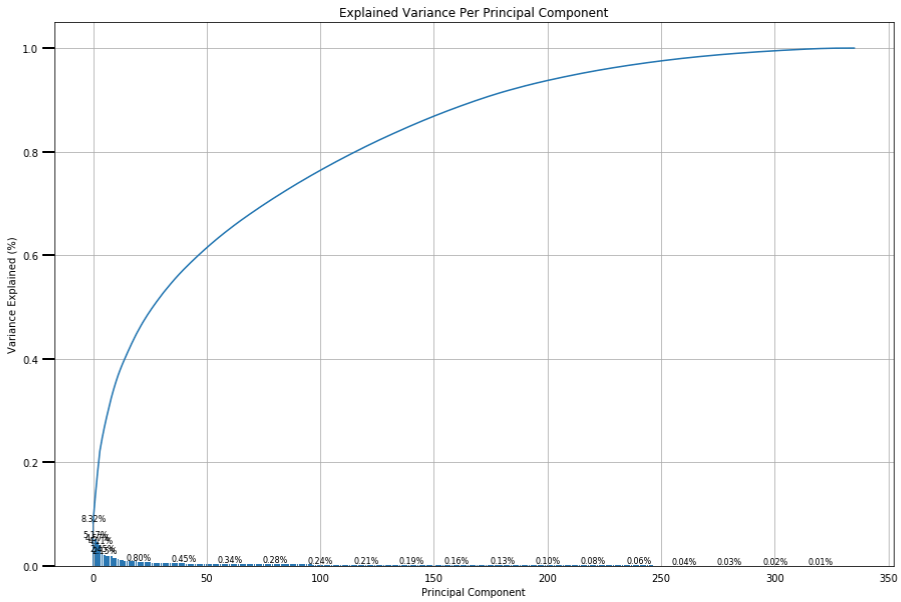
Now that the data has been pre-processed and cleaned, I will Impute and Scale the data as to prepare it for PCA application on it.

In [41]:
```
with active_session():

    imputer = Imputer(missing_values='NaN',strategy='most_frequent',axis=0)
    sample_az_imputed = imputer.fit_transform(sample_az)
    sample_az_imputed = pd.DataFrame(sample_az_imputed)
```

In [42]:
```
with active_session():
    scaler = StandardScaler()
    sample_az_scaled = scaler.fit_transform(sample_az_imputed)
    sample_az_scaled = pd.DataFrame(sample_az_scaled, columns=list(sample_az.columns.values))
```

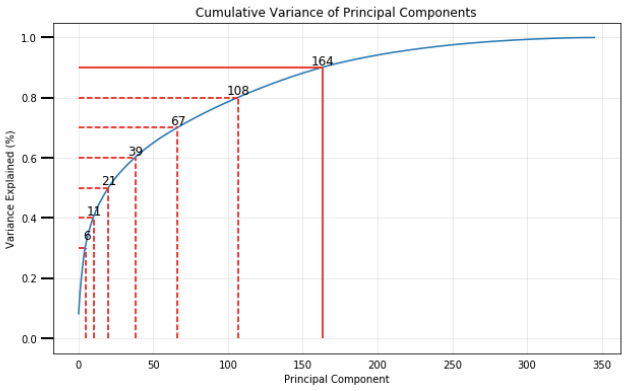
4. UNSUPERVISED LEARNING

As a primary step, I will apply PCA to the sampled AZDIAS to have data on variance so I can be able to choose the number of dimension the data should be reduced to. To better visualize the explained variance I will use two functions given by Udacity. The scree plot and the Cumulative Variance explained by components. These two functions will plot the information returned by the first application of PCA to the data, aiding me in identifying the optimum number of dimensions.



From this graph we can see that the first 200 components explain about 90% of the variance of the dataset. To get a more accurate result, I will apply the Cumulative Variance.

```
In [51]: plot_cum_var(pca)
```

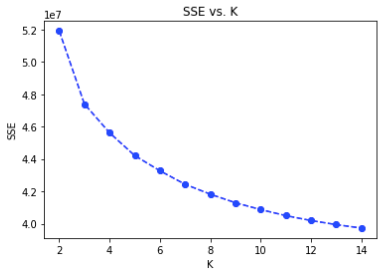


Here we can see the the exact number of components that explain for 90% of the variance is 164.
I will apply PCA to the dataset preserving 164 components.

I will now analyze the PCA data

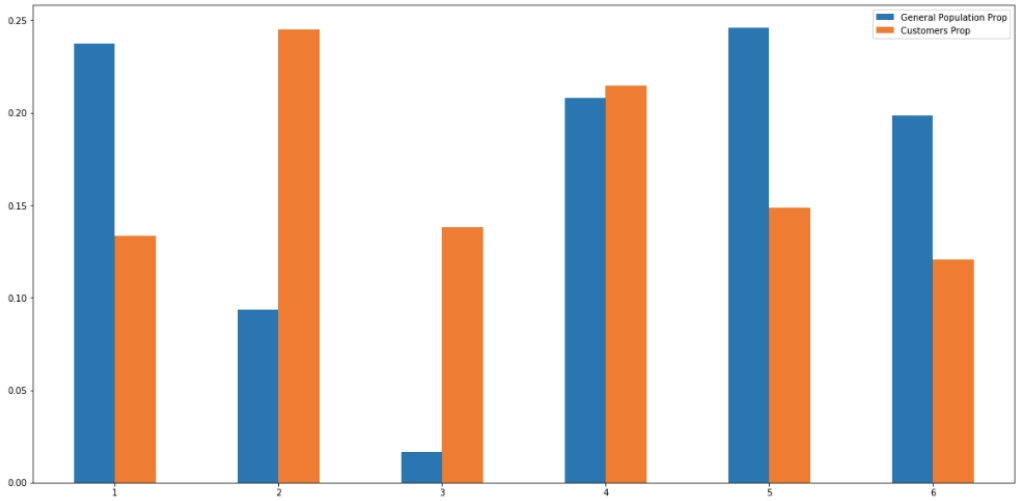
After analyzing PCA I will apply Clustering to the data using KMeans.

```
In [51]: plt.plot(centers, scores, linestyle='--', marker='o', color='b');
         plt.xlabel('K');
         plt.ylabel('SSE');
         plt.title('SSE vs. K');
```



Using the Elbow method, there is not a 100% accurate visual of the point where the distortion starts decreasing, I will choose 6 clusters, as it is the best approximated guess. I have tried to install and use Yellowbrick module that returns the exact number of cluster to be used, but unfortunately, I was unable to install it in the jupyter notebook provided.

I will now apply all the steps used in AZDIAS to CUSTOMERS, preparing the data. Comparing results form both Datasets we have the following table,



We can see that there are 2 clusters that are Over-represented in CUSTOMERS in comparison to AZDIAS. The characteristics of the first over-represented cluster, cluster number 2, is as follows: High income, not financially averse, with new residences as their property and a high number of cars in the neighborhood. Senior Age members in this group, retired. This cluster is a good match for the Mail-Order company.
Cluster number 3, or the second over-represented cluster in CUSTOMERS: Younger then previous cluster, own more properties especially vehicles. Good income, not financially averse. Age could be between 45 and 60 years. Appears to be a good match for the mail company awell.

5. SUPERVISED LEARNING

I will now pass the data through 4 different Models: Logistic Regression, Random Forest Regressor, Gradient Boosting Regressor and Gradient Boosting Classifier. Using GridSearch to aid me in selecting the best models, I will afterwards tune the best model for optimum results.

In [102]:
```
# Find best classification algorithm
clf_names = []
clf_scores = []
clf_best_ests = []
clf_time_taken = []
clf_dict = {}

for clf in [lor, rdf, gbc]:
    best_score, best_est, time_taken = fit_clf(clf, {})
    clf_names.append(clf.__class__.__name__)
    clf_scores.append(best_score)
    clf_best_ests.append(best_est)
    clf_time_taken.append(time_taken)
```

```
Training LogisticRegression :
LogisticRegression
Time taken : 80.21 secs
Best score : 0.6183
****************************************
Training RandomForestRegressor :
RandomForestRegressor
Time taken : 76.84 secs
Best score : 0.565
****************************************
Training GradientBoostingClassifier :
GradientBoostingClassifier
Time taken : 206.03 secs
Best score : 0.7409
****************************************
```

Gradient Boosting Regressor and Gradient Boosting Classifier came out with almost the exact result, so I chose to drop GBR and keep GBC. The score for GBC was 0.7482

I will now tune the Hyperparameters so the model can produce the best result possible. I will also reprocess the data as to make sure all theoretical steps of data prep were done correctly.

After tuning and reprocessing, I had a few different result being the best of them a score of 0.79. The last changes were not satisfactory and brought the model down to 0.758 again.
The best hyperparameters were:

Out[140]:
```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
                           learning_rate=0.01, loss='exponential', max_depth=8,
                           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)
```

I will now utilize this model to predict the responses of MAILOUT_TEST, save it to a CSV and upload it to Kaggle as it will provide a Benchmark for the project. Before doing this, I will analyze MAILOUT_TEST and have a general overview of the data contained in it. I will also separate the data and prepare if for upload as a CSV.

6. SUMMARY

Utilizing the GBC model to predict the responses of MAILOUT_TEST and then saving it to a CSV file so I could upload to the Kaggle competition, I could achieve a score of 0.80370, earning 19th place. Which is excellent in terms of model accuracy.

I tried different approaches to the data prep and what I find that returned better results was to not drop rows. Even though the rows have a lot of missing data, imputing and scaling return better results then dropping rows with more than 20% missing values. Also, number of cluster in the Kmeans part of project, had a bigger influence in the results than I had expected.

The last changes gave worse results, but I kept my main model, the one reported and described here, as the optimum achievement.

7. REFERENCES

https://www.sas.com/en_gb/insights/articles/analytics/machine-learning-algorithms.html

https://www.r-bloggers.com/how-to-prepare-and-apply-machine-learning-to-your-dataset/

https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

https://machinelearningmastery.com/quick-and-dirty-data-analysis-for-your-machine-learning-problem/

Artificial Intelligence: A Modern Approach, Global Edition Paperback – 2011. - by Stuart Russell;Peter Norvig (Author)