Capstone Project Report

Project Name: Customer Segmentation Report for Arvato Financial Solutions

Domain Background

Arvato is a financial solutions company with solutions such as credit management and fraud management. Arvato uses analytics to help them understand their customers better and find new customers. Prediction of customer churn has been done in papers such as: Customer churn prediction system: a machine learning approach by Praveen Lalwani, Manas Kumar Mishra, Jasroop Singh and Chadha Pratyush Sethi in 2021. And it describes a study that was conducted to develop a machine learning model to predict customer churn in a telecommunications company. The authors used data from a sample of customers to train and test several machines learning algorithms, including logistic regression, decision tree, random forest, and support vector machine. The study found that the random forest algorithm outperformed the other algorithms in terms of accuracy and F1-score, and was able to identify the most important predictors of customer churn, including monthly charges, tenure, and the presence of tech support and online security services. The authors suggest that the model could be used by the company to identify customers who are at high risk of churn and to implement targeted retention strategies. Overall, the study demonstrates the potential of machine learning algorithms in predicting customer churn and highlights the importance of identifying and addressing the factors that contribute to churn in order to improve customer retention and business performance.

Problem Statement

Arvato aims to improve their customer service, attract new clients, and enhance their product offerings. They plan to achieve this by creating customer segments from their existing clientele. This task involves utilizing classification algorithms to classify customers from the general demographic population. By doing so, Arvato hopes to gain a better understanding of their diverse client base, which will help them expand their business and maximize revenue through targeted marketing and advertising efforts.

Inputs

It was given from avrato

Udacity_AZDIAS_052018.csv: Demographics data for the general population; 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; 42 962 persons (rows) x 367 (columns).

Udacity_MAILOUT_052018_TEST.csv: Demographics data for individuals who were targets of a marketing campaign; 42 8962 persons (rows) x 367 (columns).

```
[7]: # Load in the data
     azdias = pd.read_csv('../../data/Term2/capstone/arvato_data/Udacity_AZDIAS_052018.csv', sep=';')
     customers = pd.read_csv('../../data/Term2/capstone/arvato_data/Udacity_CUSTOMERS_052018.csv', sep=';')
     attributes = pd.read_excel('DIAS Attributes - Values 2017.xlsx')
     print('Data Loaded Successfully.......!')

     Data Loaded Successfully.......!
```

Benchmark Model

I used RF as my benchmark for this project. Although I used 2 models, and created an ensemble model, I found that all of these models performed equally well as the ensemble model

Solution

After conducting an analysis, I developed a model for Arvato that enables them to identify potential customers through their marketing campaigns. Additionally, I created a report that categorizes the general population into different customer segments by clustering them together. This work is valuable for Arvato as it can help them improve their marketing strategies and target customers more effectively.

And created 2 models and created ensemble model all was almost the same out.

Algorithm and techniques

Random forest : is a machine learning algorithm that is used for classification tasks. It works by building multiple decision trees and then combining them to make a more accurate prediction. The algorithm selects random subsets of features and samples from the data to build each decision tree, which helps to reduce overfitting and improve generalization. The final prediction is made based on the predictions of all the decision trees in the forest.

Logistic regression: is a statistical method used for binary classification tasks. It is used to predict the probability of an event occurring based on one or more predictor variables. In logistic regression, the dependent variable is binary, meaning it can only take two possible values. The algorithm estimates the parameters of a logistic function, which is used to model the relationship between the predictor

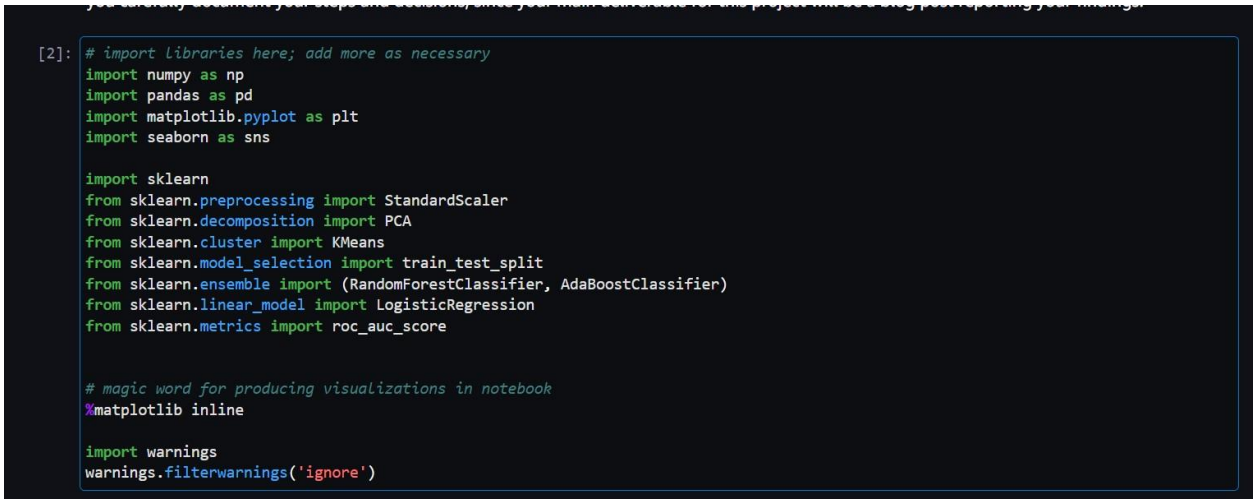
variables and the probability of the event occurring. The output of logistic regression is a probability score between 0 and 1, which can be thresholded to make a binary prediction.

evaluation

to assess the effectiveness of my model, I employed a couple of evaluation metrics. First, I used the elbow method to generate distinct clusters from the general demographic data. The resulting model had 10 clusters, and I also created visualizations of these clusters. To measure the performance of my model, I used the ROC/AUC score, which helped me determine how accurately my model was making predictions. The AUC score was 75%, indicating that my model was fairly good at predicting outcomes. This was especially impressive considering that the target variable was unbalanced, making it more prone to overfitting.

Project Design

Loading libraries



```
[2]: # import libraries here; add more as necessary
     import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns

     import sklearn
     from sklearn.preprocessing import StandardScaler
     from sklearn.decomposition import PCA
     from sklearn.cluster import KMeans
     from sklearn.model_selection import train_test_split
     from sklearn.ensemble import (RandomForestClassifier, AdaBoostClassifier)
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import roc_auc_score


     # magic word for producing visualizations in notebook
     %matplotlib inline

     import warnings
     warnings.filterwarnings('ignore')
```

Cleaning and getting to know the data

The most important part of the project is cleaning.

```
[7]: # Load in the data
     azdias = pd.read_csv('../../data/Term2/capstone/arvato_data/Udacity_AZDIAS_052018.csv', sep=';')
     customers = pd.read_csv('../../data/Term2/capstone/arvato_data/Udacity_CUSTOMERS_052018.csv', sep=';')
     attributes = pd.read_excel('DIAS Attributes - Values 2017.xlsx')
     print('Data Loaded Successfully.......!')

     Data Loaded Successfully.......!
```

I added some general analysis before starting to tell me how many missing values

```
[17]: #no of nuknow
      unknowns2=customers.isnull().sum().sum()
      print(unknowns2)

      13864522
```

And it was clear that there is unknows in form of 0,-1,9,10 in attributes

missing values inform of (-1,0,9 and even 10)

```
[13]: attributes[attributes.Meaning.str.contains('know',na=False)]
```

[13]:

	Unnamed: 0	Attribute	Description	Value	Meaning
0	NaN	AGER_TYP	best-ager typology	-1	unknown
5	NaN	ALTERSKATEGORIE_GROB	age classification through prename analysis	-1, 0	unknown
11	NaN	ALTER_HH	main age within the household	0	unknown / no main age detectable
33	NaN	ANREDE_KZ	gender	-1, 0	unknown
40	NaN	BALLRAUM	distance to next urban centre	-1	unknown
48	NaN	BIP_FLAG	business-flag indicating companies in the buil...	-1	unknown
51	NaN	CAMEO_DEUG_2015	CAMEO classification 2015 - Uppergroup	-1	unknown
105	NaN	CAMEO_DEUINTL_2015	CAMEO classification 2015 - international typo...	-1	unknown
131	NaN	CJT_GESAMTTYP	customer journey typology	0	unknown
138	NaN	D19_BANKEN_ANZ_12	transaction activity BANKS in the last 12 months	0	no transactions known
145	NaN	D19_BANKEN_ANZ_24	transaction activity BANKS in the last 24 months	0	no transactions known
161	NaN	NaN	NaN	10	no transactions known
162	NaN	D19_BANKEN_DIREKT_RZ	transactional activity based on the product gr...	0	no transaction known
170	NaN	D19_BANKEN_GROSS_RZ	transactional activity based on the product gr...	0	no transaction known
178	NaN	D19_BANKEN_LOKAL_RZ	transactional activity based on the product gr...	0	no transaction known
195	NaN	NaN	NaN	10	no transactions known
205	NaN	NaN	NaN	10	no transactions known

I therefore created functions to clean the data, replace the ‘unknowns’ with NaNs for actual cleaning and then also carried out some label encoding, to change some labels into categorical values, and so on. I also dropped some features which had, in particular too many categories in them. I did this for

both datasets and used only 2000 sample of the data instead of almost

100000 and done some standardization

```
[19]: #the data is so big so will use a small amount
      population = azdias.sample(n=2000)
      customers= customers.sample(n=2000)
      print(population.shape,customers.shape)

      (2000, 366) (2000, 369)

[20]: #simple encoding
      customers.replace({'PRODUCT_GROUP':{'COSMETIC':0,'FOOD':1,'COSMETIC_AND_FOOD':2},
                         'CUSTOMER_GROUP':{'MULTI_BUYER':0,'SINGLE_BUYER':1}
                         },inplace=True)
```

before

```
[33]: #all in one function
      def clean_data(data,attributes):

          new_attributes = identify(attributes)
          data = drop(data)
          data = unknown(data,new_attributes)
          data = impute(data,new_attributes)
          data = encode_data(data,new_attributes)

          data.fillna(method='ffill',inplace=True)

          return data

[34]: population.info()

      <class 'pandas.core.frame.DataFrame'>
      Int64Index: 2000 entries, 228465 to 443852
      Columns: 366 entries, LNR to ALTERSKATEGORIE_GROB
      dtypes: float64(267), int64(93), object(6)
      memory usage: 5.6+ MB
```

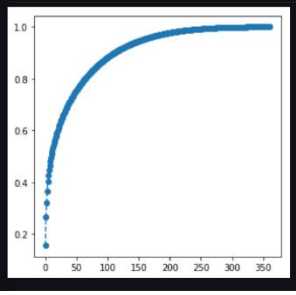
after cleaning

```
[36]: #clean and encode data
      population = clean_data(population, attributes)
      customers = clean_data(customers,attributes)

      •••

[38]: customers.info()

      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 2000 entries, 0 to 1999
      Columns: 359 entries, LNR to ALTERSKATEGORIE_GROB
      dtypes: float64(359)
      memory usage: 5.5 MB
```

Customer Segmentation Report

After gathering the necessary data, I proceeded to divide it into segments. To achieve this, I utilized Principal Component Analysis to identify the essential features that accounted for 70% of the variation. This analysis yielded 37 features that were crucial in the segmentation process. With these features, I transformed the data and developed new datasets for both the general population and the customer population.

```
[42]: #using pca
      pca = PCA()
      pca.fit(population)

      num_components = len(pca.explained_variance_ratio_)
      ind = np.arange(num_components)
      cumulativeValue = pca.explained_variance_ratio_.cumsum()

      #ploting the cumulative value as below
      plt.figure(figsize= (5, 5))
      plt.plot(ind, cumulativeValue, marker = 'o', linestyle="--");
```



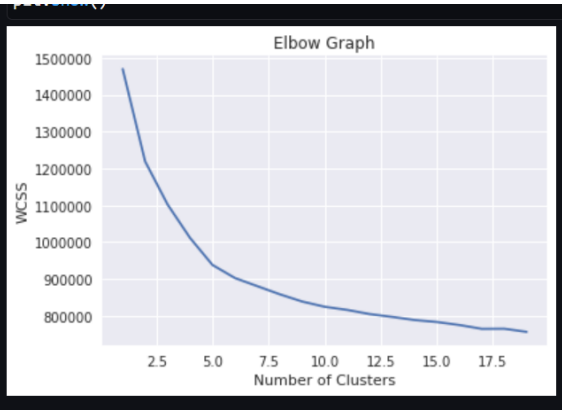

```
[45]: #getting col with explain 70% +
      df= pd.DataFrame(cumulativeValue,columns=["cum_variance"])
      mask=df.cum_variance > 0.70
      k=df[mask].index[0]

      print(k)
```

```
37
```

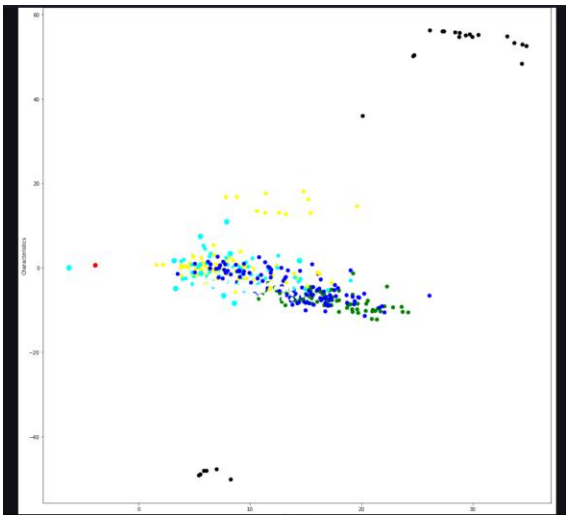
```
[48]: pca = PCA(n_components=37)
      newpop_data = pca.fit_transform(population)
      print(newpop_data.shape)
```

```
(2000, 37)
```

I used elbow graph to know how many cluster and it started stalling after 10

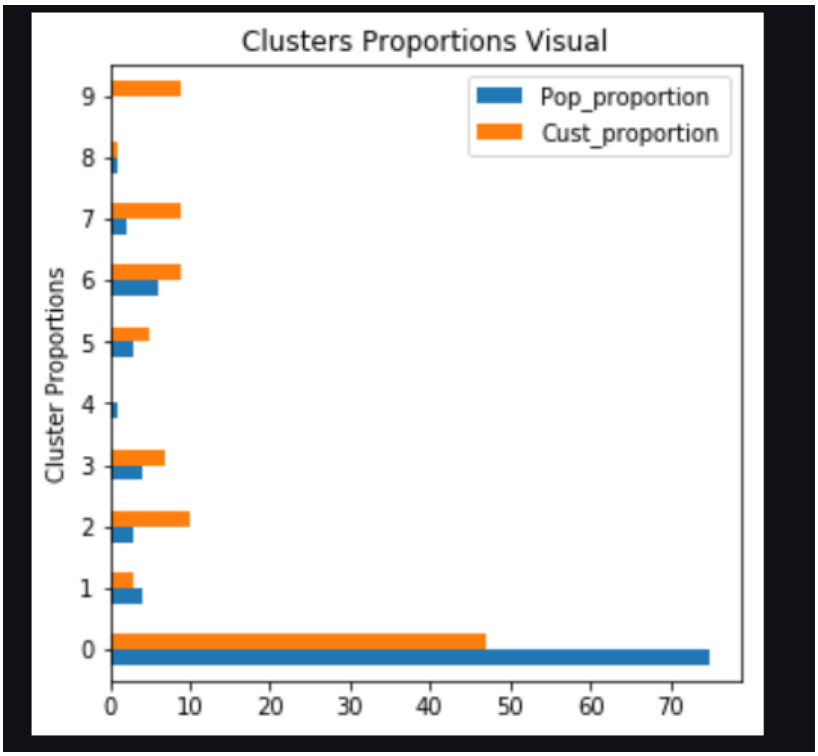


Then ploted the clusters



And there they are

	Cluster	Population	Customers
0	0	1505	933
1	1	84	69
2	2	66	195
3	3	87	148
4	4	18	1
5	5	60	91
6	6	119	177
7	7	44	176
8	8	11	23
9	9	6	187



I concluded that the most customers are in 9,7,6,2,0

And potential customers are in 0,5,3,1

Supervised

```
[62]: mailout_train = pd.read_csv('Udacity_MAILOUT_052018_TRAIN.csv', sep=';')

[67]: mailout_train.shape

[67]: (42962, 367)

[63]: mailout_train.describe()
```

[63]:

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTER_KIND1	ALTER_KIND2	ALTER_KIND3	ALTER_KIND4	ALTERSKATEGORIE_FEIN	AI
count	42962.000000	42962.000000	35993.000000	35993.000000	1988.000000	756.000000	174.000000	41.000000	34807.000000	
mean	42803.120129	0.542922	1.525241	10.285556	12.606137	13.783069	14.655172	14.195122	9.855058	
std	24778.339984	1.412924	1.741500	6.082610	3.924976	3.065817	2.615329	3.034959	4.373539	
min	1.000000	-1.000000	1.000000	0.000000	2.000000	5.000000	6.000000	6.000000	0.000000	
25%	21284.250000	-1.000000	1.000000	8.000000	9.000000	12.000000	13.000000	13.000000	8.000000	
50%	42710.000000	1.000000	1.000000	10.000000	13.000000	14.000000	15.000000	15.000000	10.000000	
75%	64340.500000	2.000000	1.000000	15.000000	16.000000	16.000000	17.000000	17.000000	13.000000	
max	85795.000000	3.000000	9.000000	21.000000	18.000000	18.000000	18.000000	18.000000	25.000000	

Starting by loading the data

```
[64]: mailout_train.info()

      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 42962 entries, 0 to 42961
      Columns: 367 entries, LNR to ALTERSKATEGORIE_GROB
      dtypes: float64(267), int64(94), object(6)
      memory usage: 120.3+ MB

[65]: newmail_data=clean_data(mailout_train,attributes)

      after filling

[66]: newmail_data.info()

      <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 42962 entries, 0 to 42961
      Columns: 357 entries, LNR to ALTERSKATEGORIE_GROB
      dtypes: float64(357)
      memory usage: 117.0 MB
```

and cleaned it like before

```
[74]: from sklearn import preprocessing
      from sklearn import utils

      lab_enc = preprocessing.LabelEncoder()
      Y_encoded = lab_enc.fit_transform(newmail_data.RESPONSE)
      print(utils.multiclass.type_of_target(newmail_data.RESPONSE))
      print(utils.multiclass.type_of_target(newmail_data.RESPONSE.astype('int')))
      print(utils.multiclass.type_of_target(Y_encoded))

      continuous
      binary
      binary
```

splited the data

```
[76]: #splitting the data
      X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = .1, stratify = Y, random_state = 2)
      print(X_train.shape,X_test.shape,Y_train.shape,Y_test.shape)

      (38665, 356) (4297, 356) (38665,) (4297,)
```

I also wanted to use different models, to check performance of my dataset. I usedLogistic Regression, and Random Forest Classifiers. I then used GridSearchCV for the best parameters, chose the best models and then created an ensemble model.

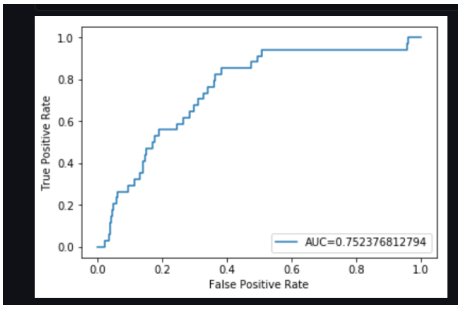
```
[78]: %%time
      #random forset good and general classifier will try it with 5 diffrent values
      rf = RandomForestClassifier()
      params_rf = {'n_estimators': [5,10,25,50,100, 200]}
      rf_gs = GridSearchCV(rf, params_rf, cv=5)
      rf_gs.fit(X_train, Y_train)

      CPU times: user 2min 30s, sys: 0 ns, total: 2min 30s
      Wall time: 2min 31s

[79]: #best model
      rf_best = rf_gs.best_estimator_

[83]: %%time
      #Logistic regression model
      log_reg = LogisticRegression()
      log_reg.fit(X_train, Y_train)

      CPU times: user 21.4 s, sys: 12.2 ms, total: 21.4 s
      Wall time: 21.6 s

[85]: #test the two models
      print('rf: {}'.format(rf_best.score(X_test, Y_test)))
      print('lr: {}'.format(log_reg.score(X_test, Y_test)))

      rf: 0.9920875029090063
      lr: 0.99185478240633

[86]: from sklearn.ensemble import VotingClassifier
      #creating dictionary for models
      estimators=[('rf', rf_best), ('log_reg', log_reg)]
      ensemble = VotingClassifier(estimators, voting='hard')

[87]: %%time
      ensemble.fit(X_train, Y_train)
      print(ensemble.score(X_test, Y_test))

      0.992087502909
      CPU times: user 22.4 s, sys: 0 ns, total: 22.4 s
      Wall time: 22.6 s
```

And the matrix

```
[89]: confusion_matrix(Y_test, Y_pred)

[89]: array([[4263,    0],
             [  34,    0]])
```

All the model done really well

I then plotted an ROC graph for visualization. The score was 75%, which means my model classified most of the data correctly.



Conclusion

this project was really fun and got to use real world data and getting used to it and it made my data cleaning skills better

A few things standout:

1. the model done well.
2. there was a lot of data cleaning and handling .
3. ROC score of 0.75 is good.
4. a lot of data got deletet and more model could have been used like xgboost