Arvato Machine Learning AWS Project Report

1. Project Overview

Effective marketing campaigns require a deep understanding of the target audience, especially in the competitive mail-order retail industry. In Germany, mail-order companies face the challenge of distinguishing their core customer base from the general population to optimize campaign targeting and resource allocation. Leveraging machine learning, businesses can analyze large-scale demographic datasets to gain actionable insights and build predictive models to improve marketing outcomes.
This project is based on real-life data provided by **Bertelsmann Arvato Analytics**, a global leader in data-driven marketing solutions. It involves analyzing customer and general population demographics to uncover key traits of the company's customer base and applying these insights to predict responses to a mail-order campaign.

For this project, we will incorporate AWS cloud services that focus on Machine Learning tools for the project to utilizes the power of the cloud and resources that will be available to get the most out of our data.

2. Problem Statement

The mail-order company struggles to identify which individuals within the general population are most likely to convert into customers. This inefficiency leads to wasted marketing resources and suboptimal campaign performance.

The specific challenges are:

1. Understanding how existing customers differ from the general population.
2. Predicting which campaign recipients are most likely to become customers based on demographic data.

This problem is measurable and replicable, with a solution directly impacting marketing efficiency and ROI. In other words, we are looking for a better view of the type of customers to determine how to grow the business and leverage the ability to gain revenue through marketing and adverting to target the customers based on their location.

3. Solution Statement

The solution involves two major steps:

1. **Customer Segmentation Report**:
    - Perform unsupervised learning on demographic data from existing customers and the general population to identify patterns and segments. This was done by creating clusters grouping customers using KMeans.
    - Identify demographic groups most aligned with the company's core customer base and those less likely to engage.
    -

2. **Supervised Learning Model**:
    - Develop a predictive model using labeled campaign data (TRAIN subset) to determine which individuals are likely to respond to future campaigns.
    - Use the resulting model to generate predictions for the TEST dataset.

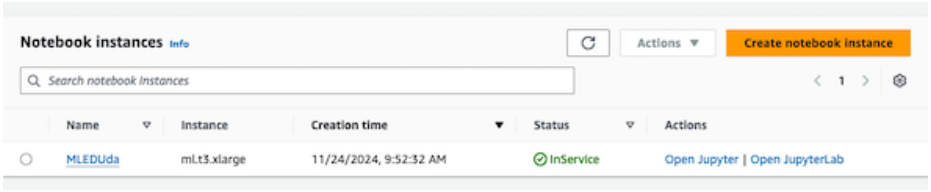
This structured approach combines exploratory insights and predictive power to solve the problem effectively.

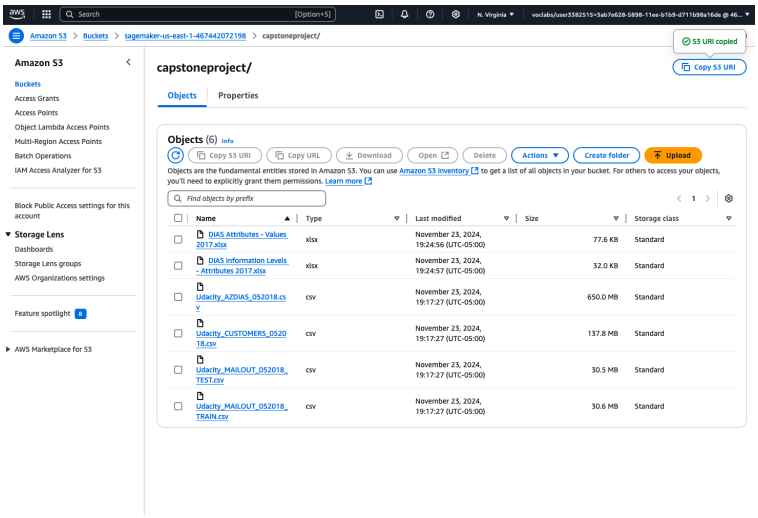
4. Datasets and Inputs

The project utilizes four datasets:

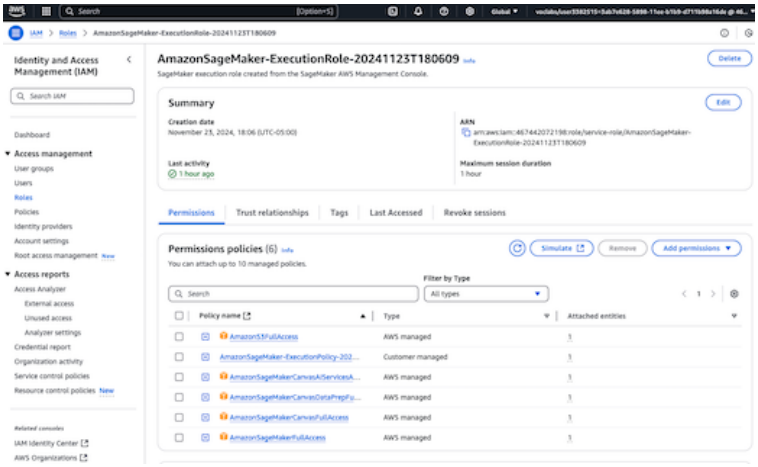
1. **General Population Data** (`AZDIAS`): 891,211 rows x 366 columns; demographic data for the general German population.
2. **Customer Data** (`CUSTOMERS`): 191,652 rows x 369 columns; demographic data for the company's existing customers, including three additional columns: `CUSTOMER_GROUP`, `ONLINE_PURCHASE`, and `PRODUCT_GROUP`.
3. **Campaign Training Data** (`MAILOUT_TRAIN`): 42,982 rows x 367 columns; demographic data for individuals targeted in a campaign, with a `RESPONSE` column indicating whether they became customers.
4. **Campaign Test Data** (`MAILOUT_TEST`): 42,833 rows x 366 columns; demographic data for campaign targets, with the `RESPONSE` column withheld for evaluation.

Each dataset provides detailed demographic attributes, including individual, household, building, and neighborhood-level information. The datasets require extensive preprocessing, including handling missing values, standardizing formats, and encoding categorical data. The data was stored on AWS S3 cloud storage to be pulled into SageMaker's Notebook Instance to be worked on.





The IAM role permissions for S3 Access was given to SageMaker's so that the Notebook Instance gets access, this was due to issues in loading the dataset.



5. Benchmark Model

For the predictive phase, a logistic regression model has served as the benchmark. Logistic regression is widely used in binary classification due to its simplicity and interpretability. It will provide a baseline for evaluating the performance of more advanced models, such as gradient-boosted trees or neural networks.

6. Evaluation Metrics

- **Customer Segmentation**:
  - **Silhouette Score**: Measures the quality of clustering by evaluating cohesion and separation. We ranged our clusters up to 20 clusters and the Silhouette score incrementally gets smaller to a value closer to 0.1.

```
# List to store inertia values for each k
DiffClusNum = []

# Loop to try different cluster numbers
for i in range(1, 20):  # Clustering from 1 to 19 clusters
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(newpop_data)  # Fit the data to the model

    # Skip silhouette score calculation if there is only 1 cluster
    if i > 1:
        score = silhouette_score(newpop_data, kmeans.labels_)
        print(f"Silhouette Score for k={i}: {score}")
    else:
        print(f"Silhouette Score for k={i}: Not applicable (only 1 cluster)

    # Append the inertia (within-cluster sum of squared distances) to the l
    DiffClusNum.append(kmeans.inertia_)
```

- **Davies-Bouldin Index**: Was originally going to be used, but the issues in getting it to import to be used.
- **Cluster Visualization**: PCA was used to visualize high-dimensional data in a 2D space. The result we get for the new Population Data after PCA: (5000, 81). The code is added below:

```
#obtaining the best features
pca = PCA() # init pca
pca.fit(population) # fit the dataset into pca model

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

- **Predictive Model**:
  - **Accuracy**: Overall correctness of predictions.
  - **Precision**: Correctly predicted positive responses.
  - **Recall**: Ability to identify all true positive responses.
  - **F1-Score**: Harmonic mean of precision and recall.

```
f1_macro = metrics.f1_score(y_test, y_pred, average='macro')  # Macro average (unweighted)
f1_weighted = metrics.f1_score(y_test, y_pred, average='weighted')  # Weighted average
print(f"F1-score (macro): {f1_macro}")
print(f"F1-score (weighted): {f1_weighted}")
```

```
F1-score (macro): 0.49689731881512705
F1-score (weighted): 0.9815369890860597
```

  - **ROC-AUC**: Measures the model's ability to distinguish between classes. A plot was also made to show a clear image of our predictions our best model has made.

These metrics ensure both segmentation quality and predictive model performance are appropriately evaluated.

7. Algorithm and Techniques

KMeans Clustering for Segmentation
We use this to grouping our customers for the training phase for our algorithm to learn what data points will be grouped into the many clusters that will have from our dataset.

Three Classification models

We test out our dataset against three different classification models to determine which model's accuracy will give us the best performing score and at a reasonable amount of time and computing resources.

- **Random Forest Classifier:** is classification algorithm that is used supervised machine learning algorithm used for classification tasks. It is based on an ensemble of decision trees, but with a bunch of them working using random ness when build each individual tree to create an uncorrelated forest that will "vote" for the most accurate out that a individual tree can't do alone.
- **K-Nearest Neighbor** or **KNN**: a simple and widely used supervised machine learning algorithm for classification and regression tasks. It is based on the principle of similarity or proximity between data points. This makes it simple and robust is common to use with classification problems like this project.
- **Logistic Regression:** a supervised machine learning algorithm used for binary classification (or multi-class classification, with extensions). Despite its name, it is a classification algorithm, not a regression algorithm. It predicts the probability of a data point belonging to one of two classes using a logistic (sigmoid) function.

All these algorithms are robust and simple. They have been for years to be classical Machine learning algorithms that will work well for the tabular dataset the is provided. These algorithms are also common in real world finance companies.

8. Evaluation Metrics

To get the ideal number of clusters that will determine the will represent the general number of cluster we're looking for. Using the elbow method we get the ideal number of clusters to be about 6 and have visualized them in a cluster plot. For the performance, ROC_AUC score was used and determine a AUC - Area under curve score of 67%, this shows that our model has done fairly well. There is a chance the model could be undercutting, but prone to overfitting due to target variable being a little unbalanced.

9. Project Design:

The first thing to do is load the data and libraries the well be used during the project. It is noted that some libraries were not used, but placed at the start of working of the project and kept just in case it will be needed later. Load the data which is larger than normal, so the use of AWS S3 will be great place to hold the data and load it into our notebook when needed. Since we are using S3 will be using SageMaker Notebook Instance and give the IAM role permissions to work with the

dataset in S3.

We then preprocess our data to clean it from any missing values and normalization of the data for a balanced and functional model output. As the image below can show how much missing data can be found in one of our datasets.

```
unknowns=azdias.isnull().sum().sum()
print('Total Unknown Values for the Population Dataset is: ',unknowns)
```

```
Total Unknown Values for the Population Dataset is:  33492923
```

This was quite a messy data set, where the first row had to be drop because it took up the attribute values row with an empty row. Some general analysis show that out attribute file also has missing and unknown values. So the values in the customer, population and attributes were replaced with encoded data points for categorical data and standardized the numerical data points. We output our dataset with the updated values:

```
customers.head()
```

	LNR	AGER_TYP	AKT_DAT_KL	ALTER_HH	ALTERSKATEGORIE_FEIN	ANZ_HAUSHALTE_AKTIV	ANZ_HH_TITEL	ANZ_KINDER	ANZ_PERSONEN	ANZ_STATISTISCHE_HAUSHALTE
0	-1.028363	0.470668	-0.320888	0.187661	-0.070114	-0.213285	-0.095985	-0.225771	0.656693	-0.203953
1	0.137201	1.903013	-0.320888	-1.172795	0.776672	-0.213285	-0.095985	-0.225771	-0.173303	-0.203953
2	-1.560757	0.470668	-0.320888	0.187661	-0.070114	-0.213285	-0.095985	-0.225771	-0.173303	-0.203953
3	1.192718	-0.961676	-0.320888	-0.492567	-2.892733	-0.213285	-0.095985	-0.225771	-1.003299	-0.203953
4	0.530655	-0.961676	0.270283	0.731844	1.058934	-0.213285	-0.095985	-0.225771	-0.173303	-0.203953

5 rows × 359 columns

After cleaning the data and exploring the dataset, we do some analytical work to check the changes will be useful for our model to use.

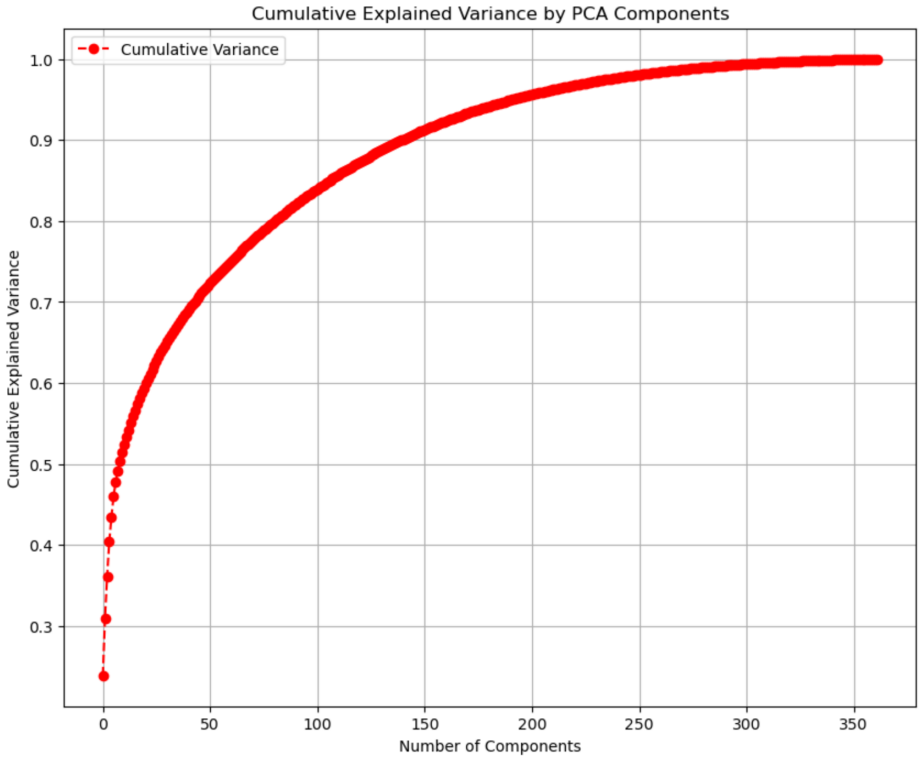
Customer Segmentation Report:

We now do look for insights by segmentation and perform the PCA - Principal Component Analysis and determine which features were most important. We aim for over 80% variance score and 81 features in our entire dataset.We can see from the code and output we are set to have 81 features that are important:

```
#obtaining columns that have explain 80% variance in the population dataset
df= pd.DataFrame(cumulativeValue,columns=["cum_variance"])
mask=df.cum_variance > 0.80
k=df[mask].index[0]

print('Columns with 80% variance: ',k)
```

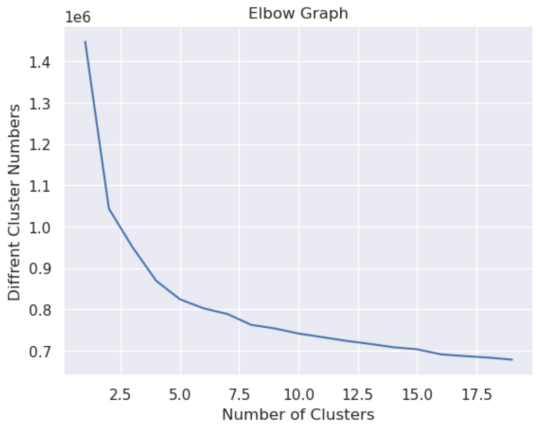
```
Columns with 80% variance:  81
```



We have a graph to show above to show where in the curve the number of components that well us for our new dataset to train our model on. This dataset show be able to create clusters so we calcite the silhouette score to evaluate how well our data points would cluster together. We have a score of around 0.1 which shows that we will likely have overlap of our clusters with the more interactions we do.

```
Silhouette Score for k=1: Not applicable (only
Silhouette Score for k=2: 0.41146114882282364
Silhouette Score for k=3: 0.4254171516425973
Silhouette Score for k=4: 0.18438425790645374
Silhouette Score for k=5: 0.13752429370319186
Silhouette Score for k=6: 0.12700962722211687
Silhouette Score for k=7: 0.11719708046066657
Silhouette Score for k=8: 0.12191662326126888
Silhouette Score for k=9: 0.11976775843971622
Silhouette Score for k=10: 0.11880369308427657
Silhouette Score for k=11: 0.1149062993940427
Silhouette Score for k=12: 0.11719334974866083
Silhouette Score for k=13: 0.11578680549523754
Silhouette Score for k=14: 0.11737579465792118
Silhouette Score for k=15: 0.1145100533868505
Silhouette Score for k=16: 0.10166940546757812
Silhouette Score for k=17: 0.10242272594856665
Silhouette Score for k=18: 0.10189408009872118
Silhouette Score for k=19: 0.10005139877651331
```

After using the elbow graph, a good determined number of clusters would be about 6, where the graph starts to curve flat real quick. This shows we will have benefits the more we continue increasing the numbers of clusters that we could use.



We then visualize our clusters and the visible groups that exist in our clusters. We would have mostly customers in 2 and 5, but most potential customer will come from cluster 5 and 1.

```
cluster_info
```

	Cluster	Population	Customers	Pop_proportion	Cust_proportion
0	0	1123	227	22.0	5.0
1	1	1159	721	23.0	14.0
2	2	930	1223	19.0	24.0
3	3	83	658	2.0	13.0
4	4	586	881	12.0	18.0
5	5	1119	1290	22.0	26.0

Predictive Model Results:

We start with the provided Mail-out dataset to create models and clean this dataset by using a pervious function. The image below shows that action being taken. It said a lot of time since this test dataset will encode the target variable as in pervious datasets.

```
Cleaning attributes data ...
==============================
Deleting Nas columns ...
Old Data shape:  (42962, 367)
==============================
New Data shape:  (42962, 361)
==============================
==============================
Replacing Unknowns by Nas ...
==============================
Imputing data ...
Number of Nas before filling : 2007026
Number of Nas after filling Numeric features : 2007026
Number of Nas after  filling All features: 0
==============================
Encoding data ...
==============================
Final data size:  (42962, 357)
```

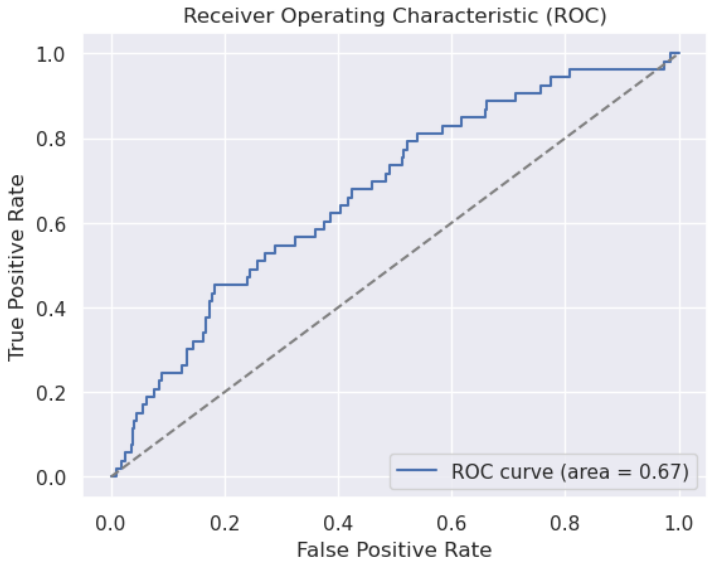
Then the dataset is split into a X-train, y-train, X-test, y-test split and used to train against our three different classification models. We first ran it on our Random Forest Classifier model and GridSearch possible value to estimate the best possible hyper parameter the model could use, which for this was `{'n_estimators': 10}.` For our K-Neighbors Classifier , we used GridSearch as well and found our best estimator was `{'n_neighbors': 4}.` For the Logistic Regression model, all we had to do was train our model. We then print out our best scores from our models and run a Voting Classifier to print out the best model and score.

```
knn: 0.9876658133581568
rf: 0.9867349313474517
log_reg: 0.9876658133581568
```

```
from sklearn.ensemble import VotingClassifier
#VotingClassifer is used to determine best model
estimators=[('knn', knn_best), ('rfc', rfc_best), ('log_reg', log_reg)]
ensemble = VotingClassifier(estimators, voting='hard')
ensemble.fit(X_train, y_train)
print(ensemble.score(X_test, y_test))
```

```
0.9876658133581568
```

The ROC-AUC curve was plotted to determine the area curve of possible the true positive rate against a false positive rate which is 0.67 or 67% of our models accuracy. Meaning the model is relatively classifying correct data, but also may need improving.



Last thing to do is to determine the F1-score, however comparing macro vs weight scores. Both compute F1-score, but Marco does not consider proportion for each label in dataset where weighted does. So we get `(F1-score (macro): 0.49689731881512705, F1-score (weighted): 0.9815369890860597 )` so we have an imbalance in the dataset and our model is performing poorly on the minority classes but excelling in the majority classes in our dataset after using our model.

Conclusion

This project will provide actionable insights through customer segmentation and predictive modeling, enabling the financial institution to optimize its marketing strategies. By leveraging unsupervised and supervised learning, it shows that we can still have room for improvement from our models. The likely action needed will be further hyper parameter tuning or just using another model such as XGBoost which can be better for this dataset.