Machine Learning Engineer Nanodegree

Capstone Project

September 2021

This Capstone project is part of the Udacity Machine Learning Engineer Nanodegree. First, the problem to solve is defined and potential solutions and performance metrics investigatd. Next, the problem will be analyzed through visualizations and an initial data exploration to gain a better understanding of what algorithms and features are appropriate for solving it. Further, two algorithms are implemented to tackle the problem. Results will be collected and validated accordingly. Finally, the results will be discussed in regard to possible improvements.

I. Definition

Chapter I will give an overview over the project as well as define the problem statement asked for in this project and the metrics to evaluate.

Project Overview

Starbucks is one of the most well-known companies of the world and the world's largest coffeehouse chain. Its goal is to give his customers always the best service and the best experience. In doing so, Starbucks offers a free app to make orders online, predict the waiting time, to operate a customer rewards program, and to promote special offers. These promotions include drink advertisements, discount offers, and buy one get one free (BOGO) offers. To maximize the effectiveness of these promotional offers, not every customer receives the same promotional offer. Instead, Starbucks tailors promotions and advertisements to the unique characteristics of individual customers and their customer segments.

This capstone project aims to predict customer responses to tailor marketing and promotional offers, also known as propensity modeling. Udacity partnered with Starbucks to provide a real-world data set for this use case. Yet, the given data are not real-life data but rather simulated data mimicking their customer behavior. This ensures data privacy. The data allow for propensity modeling to attempt to predict the likelihood of users of the starbucks mobile web app to react to specific offers. Overall, we want to propose marketing

offers on a personalized selection of users.

Problem Statement

The aim is to predict what user will accept and offer and what user does not. Some customers do not want to receive offers and might be turned off by them, so we want to avoid sending offers to those customers. User are distinguished using personal data like gender, age, and income, etc.

Metrics

The analysis will include looking on the accuracy of the models predictions. The ***accuracy*** defines the proportion of both True Positives and True Negatives among the total number of cases. The best value is one, the worst value is zero.

Additionally, there are two possible errors in prediction which need to be looked in to, False Positives and False Negatives.

A False Positive prediction will most likely result in the user ignoring the marketing effort. This results in wasted effort of the company as well as the user feeling bothered by it. ***Precision*** is used for high False Positives. The best value is one, and the worst value is zero.

A False Negative predictions means there is no offer sent but user would have likely used it, resulting in an wasted business opportunity. ***Recall*** is used for high False Negatives. The best value is one, and the worst value is zero.

Following those thoughts the evaluation will focus on the recall and precision as well to reduce the number of False Negatives. Recall seems to be more important for our business case.

II. Analysis

The structure of the project consists of three separate notebooks. The naming convention states their functionality.

- 1-Data_Exploration.ipynb
- 2-Feature_Engineering.ipynb
- 3-Model-training.ipynb

The data sets for data exploration and for model training will be stored on an AWS S3 bucket and loaded accordingly. All scripts will be executed using AWS Sagemaker.

Data Exploration

The data consists of 3 .json-files containing simulated data that mimic customer behavior on the Starbucks rewards mobile app.

portfolio.json - contains information about the offers

```
portfolio.head()
```

	channels	difficulty	duration	id	offer_type	reward
0	[email, mobile, social]	10	7	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10
1	[web, email, mobile, social]	10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	10
2	[web, email, mobile]	0	4	3f207df678b143eea3cee63160fa8bed	informational	0
3	[web, email, mobile]	5	7	9b98b8c7a33c4b65b9aebfe6a799e6d9	bogo	5
4	[web, email]	20	10	0b1e1539f2cc45b7b9fa7c272da2e1d7	discount	5

Overall, there are three types of offers that can be sent: buy-one-get-one (BOGO), discount, and informational. In a BOGO offer, a user needs to spend a certain amount to get a reward equal to that threshold amount. In a discount, a user gains a reward equal to a fraction of the amount spent. In an informational offer, there is no reward, but neither is there a requisite amount that the user is expected to spend. Offers can be delivered via multiple channels: email, web, mobile, social. Each offer can be identified via an id.

profile.json - contains information about the customers

```
profile.head()
```

	age	became_member_on	gender	id	income
0	118	20170212	None	68be06ca386d4c31939f3a4f0e3dd783	NaN
1	55	20170715	F	0610b486422d4921ae7d2bf64640c50b	112000.0
2	118	20180712	None	38fe809add3b4fcf9315a9694bb96ff5	NaN
3	75	20170509	F	78afa995795e4d85b5d9ceeca43f5fef	100000.0
4	118	20170804	None	a03223e636434f42ac4c3df47e8bac43	NaN

The profile data incorporates demographical data such as age, income, gender, and when the user became a member. Each person can be identified via an id.

transcipt.json - contains information about customer purchases and interaction with the offers

The transcipt data consists of the persons interactions to offers or purchases, denoted as event. There are four types: offer received, offer viewed, offer completed, and transactions. The offer can be identified via an offer*id, and the person can be identified via an person*id.

```
transcript.head()
```

	event	person	time	value
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	{u'offer id': u'9b98b8c7a33c4b65b9aebfe6a799e6...
1	offer received	a03223e636434f42ac4c3df47e8bac43	0	{u'offer id': u'0b1e1539f2cc45b7b9fa7c272da2e1...
2	offer received	e2127556f4f64592b11af22de27a7932	0	{u'offer id': u'2906b810c7d4411798c6938adc9daa...
3	offer received	8ec6ce2a7e7949b1bf142def7d0e0586	0	{u'offer id': u'fafdcd668e3743c1bb461111dcafc2...
4	offer received	68617ca6246f4fbc85e91a2a49552598	0	{u'offer id': u'4d5c57ea9a6940dd891ad53e9dbe8d...

Exploratory Visualization

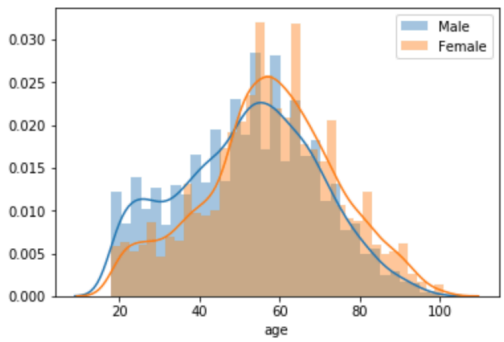
Most revelant for our analysis and as input into the models are the demographical data. It is important to account for a normal (or equal) distribution of feature values. The following two graphs depict the gender-age relation as well as the gender-income relation.

```
profile.gender.value_counts()
```

```
M    8484
F    6129
O     212
Name: gender, dtype: int64
```

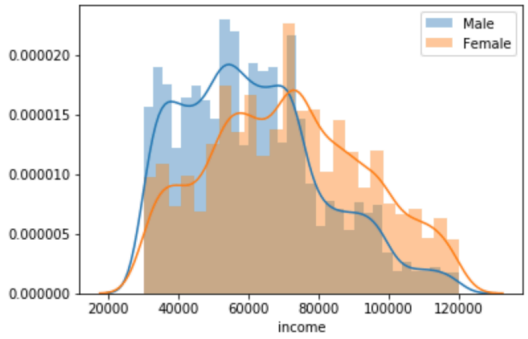
Overall, there are three genders. Male, female, and other. The following analysis will focus on the prior two, as other has fairly few data entries.

```
sns.distplot(profile[profile.gender=='M'].age,label='Male')
sns.distplot(profile[profile.gender=='F'].age,label='Female')
plt.legend()
plt.show()
```



Males and females are normal distributed over age. The minimum age is 20, the maximum age is 118.

```
sns.distplot(profile[profile.gender=='M'].income,label='Male')
sns.distplot(profile[profile.gender=='F'].income,label='Female')
plt.legend()
plt.show()
```



Females tend to be normal distributed over income. Males seem to have a lower income in general compared to females. The income ranges from 30000 to 120000.

Further, as the goal is to predict whether an offer is successful or not, it is important to look at the distribution of the four offer types: offer received, offer viewed, offer completed, and transactions. About 40% of the data entries are transactions. Out of all offer entries, 33575 have been completed. It is important to note that customers can complete an offer without viewing it.

```
transcript.groupby('event').person.count()
```

```
event
offer completed     33579
offer received      76277
offer viewed        57725
transaction        138953
Name: person, dtype: int64
```

Algorithms and Techniques

A Support Vector Machine (SVM) will be used as a **benchmark model**, using the kernel method to utilize non-linearity. The SVM will be implemented using scikit-learn and deployed using sagemaker. Secondly, an Artificial Neural Netwrk (ANN) will be implemented and compared with the benchmark model. The ANN will be implemented using tensorflow and also deployed using sagemaker. The data will be split into a training and testing sets using a 80-20 split.

I am aware that a more simpler more interpretable benchmark model would be to use a logistic regression. However, out of curiousity I have chosen to use an SVM.

III. Methodology

Data Preprocessing

Data preprocessing includes feature engineering and data cleaning of the three given data

tables.

```
# Drop email and reward column as they are not needed
portfolio_ohe = portfolio_ohe.drop(['email','reward'], axis=1)
portfolio_ohe
```

	difficulty	duration	id_offer	mobile	social	web	offer_bogo	offer_discount	offer_informational
0	0.50	0.571429	ae264e3637204a6fb9bb56bc8210ddfd	1	1	0	1	0	0
1	0.50	0.285714	4d5c57ea9a6940dd891ad53e9dbe8da0	1	1	1	1	0	0
2	0.00	0.142857	3f207df678b143eea3cee63160fa8bed	1	0	1	0	0	1
3	0.25	0.571429	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	0	1	1	0	0
4	1.00	1.000000	0b1e1539f2cc45b7b9fa7c272da2e1d7	0	0	1	0	1	0
5	0.35	0.571429	2298d6c36e964ae4a3e7e9706d1fb8c2	1	1	1	0	1	0
6	0.50	1.000000	fafdcd668e3743c1bb461111dcafc2a4	1	1	1	0	1	0
7	0.00	0.000000	5a8bc65990b245e5a138643cd4eb9837	1	1	0	0	0	1
8	0.25	0.285714	f19421c1d4aa40978ebb69ca19b0e20d	1	1	1	1	0	0
9	0.50	0.571429	2906b810c7d4411798c6938adc9daaa5	1	0	1	0	1	0

The portfolio data is one-hot-encoded on the offer types and the channels where these offers are promoted. Additionally, difficulty and duration are normalized.

```
profile_ohe
```

	id_customer	membership_total_days	gender_F	gender_M	gender_O	age_(20, 29]	age_(29, 39]	age_(39, 49]	age_(49, 59]	age_(59, 69]	age_(
1	0610b486422d4921ae7d2bf64640c50b	1518	1	0	0	0	0	0	1	0	
3	78afa995795e4d85b5d9ceeca43f5fef	1585	1	0	0	0	0	0	0	0	
5	e2127556f4f64592b11af22de27a7932	1233	0	1	0	0	0	0	0	1	
8	389bc3fa690240e798340f5a15918d5c	1309	0	1	0	0	0	0	0	1	
12	2eeac8d8feae4a8cad5a6af0499a211d	1399	0	1	0	0	0	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	
16995	6d5f3a774f3d4714ab0c092238f3a1d7	1194	1	0	0	0	0	1	0	0	
16996	2cb4f97358b841b9a9773a7aa05a9d77	1155	0	1	0	0	0	0	0	1	
16997	01d26f638c274aa0b965d24cefe3183f	1688	0	1	0	0	0	1	0	0	
16998	9dc1421481194dcd9400aec7c9ae6366	2013	1	0	0	0	0	0	0	0	
16999	e4052622e5ba45a8b96b59aba68cf068	1511	1	0	0	0	0	0	0	1	

14825 rows × 17 columns

Similar to the portfolio table, the profile data is one-hot-encoded on the genders and age. In order to one-hot-encode age, age groups have been defined in intervals of 10 years. Additionally, the membership of the customers have been calculated to the number of total days (from registration until today).

The transcript data is one-hot-encoded on the event types and value column. Most important from the values are the offer*id and id*offer, as they are used to merge the data in the next step.

```
# Combine data tables based on id_offer and id_customer
combined = pd.merge(transcript_ohe, portfolio_ohe, how='left', on="id_offer")
combined = pd.merge(combined, profile_ohe, how='left', on="id_customer")

# Remove columns that are not needed
combined = combined.drop(['id_customer','id_offer','offer_id','offer_informational','time','reward'], axis=1)

# There are likely to be duplicated we do not need.
print(combined.shape)
combined.drop_duplicates(inplace=True)
combined = combined.dropna()
print(combined.shape)
```

```
(167581, 26)
(59046, 26)
```

```
combined.columns
```

```
Index(['event_offer completed', 'event_offer received', 'event_offer viewed',
       'difficulty', 'duration', 'mobile', 'social', 'web', 'offer_bogo',
       'offer_discount', 'membership_total_days', 'gender_F', 'gender_M',
       'gender_O', 'age_(20, 29]', 'age_(29, 39]', 'age_(39, 49]',
       'age_(49, 59]', 'age_(59, 69]', 'age_(69, 79]', 'age_(79, 89]',
       'age_(89, 99]', 'age_(99, 109]', 'income_(30000, 60000]',
       'income_(60000, 90000]', 'income_(90000, 120000]'],
      dtype='object')
```

All three of the above mentioned and feature-engineered dataframes will be combined to one dataframe including all features. After merging the data tables, the corresponding ids can be dropped as well as other columns that are not needed. Further, duplicated and rows inlucing NaN-values will be dropped as well. This leaves a total of 59046 entries for modeling and 28 features.

```
print(f"received: {combined['event_offer received'].sum()}")
print(f"received: {combined['event_offer completed'].sum()}")
print(f"received: {combined['event_offer viewed'].sum()}")
```

```
received: 30784
received: 10308
received: 17954
```

```
successfull_percent = round(combined['event_offer completed'].sum() / combined['event_offer completed'].count(), 4)*100
print(f"total #: {combined['event_offer completed'].sum()}")
print(f"{successfull_percent}% successful offers.")
```

```
total #: 10308
17.46% successful offers.
```

As the goal is to predict the completed offers, it is important to have a final check on their relation compared to all offers. After feature engineering and data cleaning, this leaves us with a total 10308 completed offers, which is about 17.5% of all offers in the data.

```
train, test = train_test_split(combined, test_size=0.2)
```

```
print(f'Length train: {len(train)}')
print(f'Length test:  {len(test)}')
```

```
Length train: 47236
Length test:  11810
```

The combined dataframe will be split into training and testing set using a 80-20 ratio. This results in 47236 entries for the training, and 11810 entries for the test set.

Implementation

The implementation of both models can be seen in the following images.

The hyperparameters for the SVM & ANN, as well as the structure of the ANN have been defined in an iterative process. Below shown are the final.

SVM

```
## Define a model
model = SVC(C=10000.0, gamma=1e-05)

## Train the model
model.fit(train_x, train_y)
```

ANN

```
input_shape, hidden_shape, output_shape = len(X_train_nn[0]), 42, 1

## Define a model
model = keras.models.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Dense(hidden_shape, activation='sigmoid'),
        layers.Dense(hidden_shape, activation='sigmoid'),
        layers.Dense(hidden_shape, activation='sigmoid'),
        layers.Dense(hidden_shape, activation='sigmoid'),
        layers.Dense(output_shape, activation="sigmoid"),
        layers.Dropout(0.25)
    ]
)

model.summary()

# compile model
model.compile(loss="mean_squared_error", optimizer='adam', metrics=["accuracy"])
# fit model
model.fit(X_train_nn, y_train_nn, epochs=epochs,verbose=1)
```

Refinement

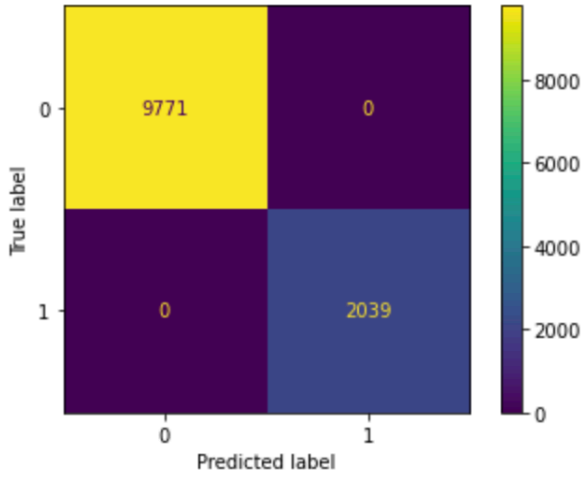
Refinement of the models has been done iteratively during development and while debugging. For example, the structure of the ANN has been adjusted to a different number of layers and nodes, as well as varying activation functions, optimizers, or loss functions.

IV. Results

Model Evaluation and Validation

SVM

```
Recall:    1.0
Precision: 1.0
Accuracy:  100.0%
```



The SVM had a "perfect" prediction of 100%. This means, that Recall and Precision are both at 1.0. The SVM can predict whether a customer will complete its offer.

ANN

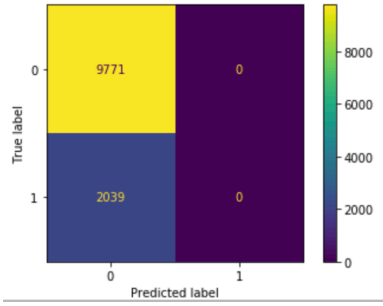
```
# get confusion matrix and display it
cm = confusion_matrix(y_test, tf_predictions_flat)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0, 1])
disp.plot(values_format="n")

# get tn, fp, fn, tp
tn, fp, fn, tp = cm.flatten()

# calculate accuracy, recall, precision
acc = accuracy_score(tf_predictions_flat, y_test)
precision = tp / (tp + fp)
recall = tp / (tp + fn)

print(f'Recall:    {recall}')
print(f'Precision: {precision}')
print(f'Accuracy:  {round(acc,4)*100}%')
```

```
/home/ec2-user/anaconda3/envs/amazonei_tensorflow_p36/lib/python3.6/site-packages/ipykernel/__main__.py:11: RuntimeWarning: invalid value encountered in long_scalars
Recall:    0.0
Precision: nan
Accuracy:  82.73%
```



The ANN seems only to predict zeros, meaning to predict a non completed offer. The accuracy of 83% depicts the ratio of non-completed offers of the fully combined dataset. Thus, the current ANN is not able to predict whether a customer will complete its offer.

Justification

The overall goal was to predict whether an user complete accept or not. The problem can be

solved using the benchmark SVM model. The ANN needs further refinement as it is not able to predict in its current state. Yet, it is a best practice to go with a simpler model in production, meaning I would state the analysis and modeling apprach as successful.

V. Conclusion

Improvement

Clearly, the ANN need improvements such as it is not able to predict the current task.

Further, the feature engineering and data cleaning can be improved. For example, there are multiple entries per person with different event. There might be a correlation over time which could be accounted for. It could also be analyzed what type of channel is relevant for an offer completion.

Reflections

The most difficult part was to get the tensorflow model running on sagemaker, as there were some specifics about the model saving and estimator.predictions.

However, I really enjoyed working on this use case and also on debugging, as I learned a lot about sagemaker and using it to deploy models.