

# Predict personalised offer portfolios for Starbucks customers using Machine Learning

What are the benefits of a figured out promotional strategy? It makes you **save marketing expenses** and, by the way, **increases customer satisfaction**. Here is my solution how to schedule a smart and simple **promotional strategy for Starbucks**: Firstly we analyse simulated data of the [Starbucks Rewards mobile app](#) and then we prepare machine learning models to **predict personalised offer portfolios** to each customer.

For those who haven't heard about Starbucks yet, here is a short introduction: [Starbucks Corporation](#), existing since 1971, is a multinational chain of coffee houses and roastery reserves with headquarter in Seattle, United States. It's the world's largest coffee house chain with over 30,000 locations worldwide in more than 70 countries. Typical Starbucks products are whole-bean coffee, microground instant coffee, caffe latte, espresso, miscellaneous tea products, juices and snacks such as chips or crackers.

Starbucks has a very **popular app** which includes engaging loyalty programs and functions for mobile payment and ordering. The promotional strategy will be drafted on simulated data which mimics customer behavior on the [Starbucks Rewards mobile app](#) for one product. The data is stored in three data sets with following informations:

- **portfolio** — data set containing characteristics of each single offer as offer type, difficulty, reward, duration and communication channels
- **profile** — data set containing demographics of each customer as age, registration date, gender and income
- **transcript** — data set containing records of customer activities as transaction values for purchases, receive, view or complete and offer and time since start of test

## portfolio (10 rows, 6 columns)

	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd

## profile (17,000 rows, 5 columns)

	gender	age	id	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN

## transcript (306,534 rows, 4 columns)

	person	event	value	time
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0

Figure 1 — first row of dataframes portfolio, profile and transcript

This **mobile app works as follows**: once every few days, Starbucks sends out a promotional offer to its customers. An offer can be an advertisement for a drink or an actual offer such as a discount or BOGO (i.e. Buy One Get One free). There are also informational offers which are only providing information about a product without any reward. Every offer has a validity period. Some customers might not receive any offer for weeks and not all customers receive the same offer. The transactional data shows user purchases made on the app including timestamp of each purchase and amount of money spent on a purchase. This transactional data records when actually a customer receives, views or completes an offer.

	person	event	value	time	
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	{offer_id: '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	0	BOGO difficulty 5 USD reward 5 USD duration 7 days
15561	78afa995795e4d85b5d9ceeca43f5fef	offer viewed	{offer_id: '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	6	
47582	78afa995795e4d85b5d9ceeca43f5fef	transaction	{amount: 19.89}	132	
47583	78afa995795e4d85b5d9ceeca43f5fef	offer completed	{offer_id: '9b98b8c7a33c4b65b9aebfe6a799e6d9'}	132	
49502	78afa995795e4d85b5d9ceeca43f5fef	transaction	{amount: 17.78}	144	INFORMATIONAL difficulty 0 USD reward 0 USD duration 3 day
53176	78afa995795e4d85b5d9ceeca43f5fef	offer received	{offer_id: '5a8bc65990b245e5a138643cd4eb9837'}	168	
85291	78afa995795e4d85b5d9ceeca43f5fef	offer viewed	{offer_id: '5a8bc65990b245e5a138643cd4eb9837'}	216	
87134	78afa995795e4d85b5d9ceeca43f5fef	transaction	{amount: 19.67}	222	BOGO difficulty 10 USD reward 10 USD duration 7 days
92104	78afa995795e4d85b5d9ceeca43f5fef	transaction	{amount: 29.72}	240	
141566	78afa995795e4d85b5d9ceeca43f5fef	transaction	{amount: 23.93}	378	
150598	78afa995795e4d85b5d9ceeca43f5fef	offer received	{offer_id: 'ae264e3637204a6fb9bb56bc8210ddfd'}	408	
163375	78afa995795e4d85b5d9ceeca43f5fef	offer viewed	{offer_id: 'ae264e3637204a6fb9bb56bc8210ddfd'}	408	BOGO difficulty 5 USD reward 5 USD duration 5 days
201572	78afa995795e4d85b5d9ceeca43f5fef	offer received	{offer_id: 'f19421c1d4aa40978ebb69ca19b0e20d'}	504	
218393	78afa995795e4d85b5d9ceeca43f5fef	transaction	{amount: 21.72}	510	
218394	78afa995795e4d85b5d9ceeca43f5fef	offer completed	{offer_id: 'ae264e3637204a6fb9bb56bc8210ddfd'}	510	
218395	78afa995795e4d85b5d9ceeca43f5fef	offer completed	{offer_id: 'f19421c1d4aa40978ebb69ca19b0e20d'}	510	BOGO difficulty 5 USD reward 5 USD duration 5 days
230412	78afa995795e4d85b5d9ceeca43f5fef	transaction	{amount: 26.56}	534	
262138	78afa995795e4d85b5d9ceeca43f5fef	offer viewed	{offer_id: 'f19421c1d4aa40978ebb69ca19b0e20d'}	582	

Figure 2 — transcript with events and transactions of one customer for whole test period

The **steps to complete an offer** are explained according to the table above for the customer “78afa995795e4d85b5d9ceeca43f5fef” with the offer “9b98b8c7a33c4b65b9aebfe6a799e6d9” (BOGO: difficulty 5 USD, reward 5 USD, duration 7 days):

- **time 0 hours** — offer was received by customer
- **time 6 hours** — offer was viewed by customer
- **time 132 hours** — customer made a purchase with a transaction of 19.89 USD
- **time 132 hours** — as the transaction was above 5 USD and prior 7 days (168 hours) from offer receive the offer was completed and the customer got a reward of 5 USD

The promotional offer campaign is always **successful** when the **customer first views** and **then completes** the offer.

# 1. Data Preparation

As **analysis workflow** we firstly explore the three data sets and then prepare each data set for further analysis. All three data sets are stored as json files and loaded as pandas dataframes into a jupyter notebook for python 3.

For **portfolio** dataframe we map offer IDs from alphanumerical unreadable code to integers and remove old offer id column. We change duration from days to hours and resort rows according offer\_type, difficulty and duration.

	reward	channels	difficulty	duration	offer_type	offer_id
8	5	[web, email, mobile, social]	5	120	bogo	9
3	5	[web, email, mobile]	5	168	bogo	4
1	10	[web, email, mobile, social]	10	120	bogo	2
0	10	[email, mobile, social]	10	168	bogo	1
5	3	[web, email, mobile, social]	7	168	discount	6
9	2	[web, email, mobile]	10	168	discount	10
6	2	[web, email, mobile, social]	10	240	discount	7
4	5	[web, email]	20	240	discount	5
7	0	[email, mobile, social]	0	72	informational	8
2	0	[web, email, mobile]	0	96	informational	3

Figure 3 — portfolio dataframe after data preparation

For **profile** dataframe we explored at exactly 2,175 of 17,000 customers missing data for gender and income. Aside we have for those customers 118 as value for age. This looks pretty like data entry error where for gender and income no values were inserted and for age 118 was inserted as placeholder. These 2,175 registrations are appr. 12.8 % of total customers. We decide to remove these customers rather than to compute a value by imputation for the missing data. Furthermore we map user IDs from alphanumerical unreadable code to integers and remove old user id column. We create dummy variable with binaries of gender column and create new columns with income range and age group.

	gender	age	became_member_on	income	customer_id	F	M	O	income_range	age_group
1	F	55	2017-07-15	112000	1	1	0	0	105k-120k	45-55

(14825, 10)

Figure 4 — first row and shape of profile dataframe after data preparation

For **trancript** dataframe we drop rows of customers with missing informations as gender, age and income. We map customer and offer IDs from alphanumerical unreadable code to integers and remove old id columns. Then we create dummy variable with binaries of value and event column. At last we explored 374 duplicated events which we remove.

	event	time	customer_id	reward	amount	offer_id	offer completed	offer received	offer viewed	transaction
0	offer received	0	2	NaN	NaN	4.0	0	1	0	0

(272388, 10)

Figure 5 — first row and shape of transcript dataframe after data preparation

Finally we create a **master dataframe** where we merge all relevant data of transcript (transaction), profile (demographic) and portfolio (offer) together including label for promotion success (1 for success and 0 for no success).

	customer_id	offer_id	promotion success	reward (USD)	difficulty (USD)	duration (hrs)	offer_type	gender	age	became_member_on	income year (USD)	gender F	gender M	gender O	income_range	age_group
0	1	3.0	0	0	0	96	informational	F	55	2017-07-15	112000	1	0	0	105k-120k	45-55
1	1	4.0	0	5	5	168	bogo	F	55	2017-07-15	112000	1	0	0	105k-120k	45-55
2	2	1.0	1	10	10	168	bogo	F	75	2017-05-09	100000	1	0	0	90k-105k	>65

Figure 6— first 3 rows of master dataframe

Now let's discuss the outcome of the exploratory data analysis:

## 2. Most popular offers

Here we compare counts of completed offers (success) with counts of not completed offers (no success) for each offer id. Furthermore we compute a success rate for each offer.

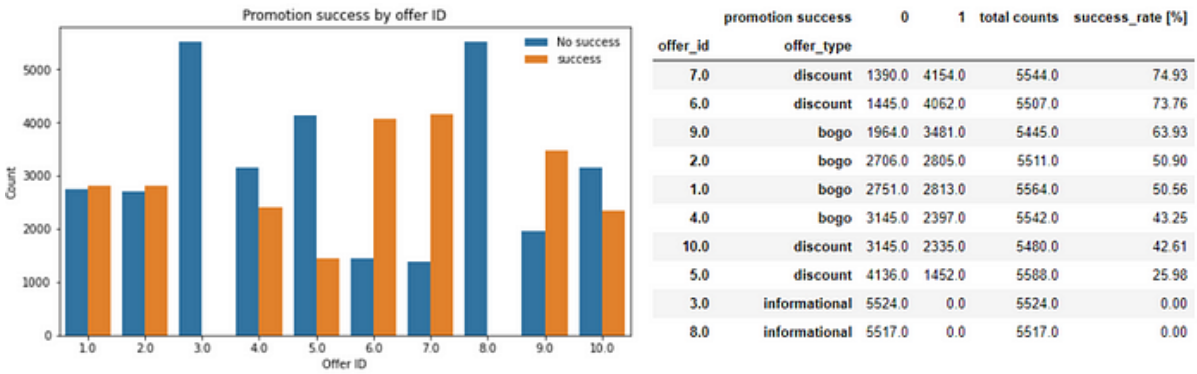


Figure 7— Promotion success rate by offer ID

We can observe that the discount offer 6 and 7 are the **most popular offers** with a success rate of 75 and 74 % followed by the four BOGO offers with success rates between 64 and 43 %. The **least popular offers** are the discount offers 10 and 5 with a success rate of 43 and 26 %. Offer 3 and 8 are informational offers where a completion of the offer with a reward is not foreseen and therefore the success rate is 0.

### 3. Difference in demographics for promotion success

Here we explore if there is a significant difference between the distribution means of successful and unsuccessful promotions for the demographics age, income and registration date.

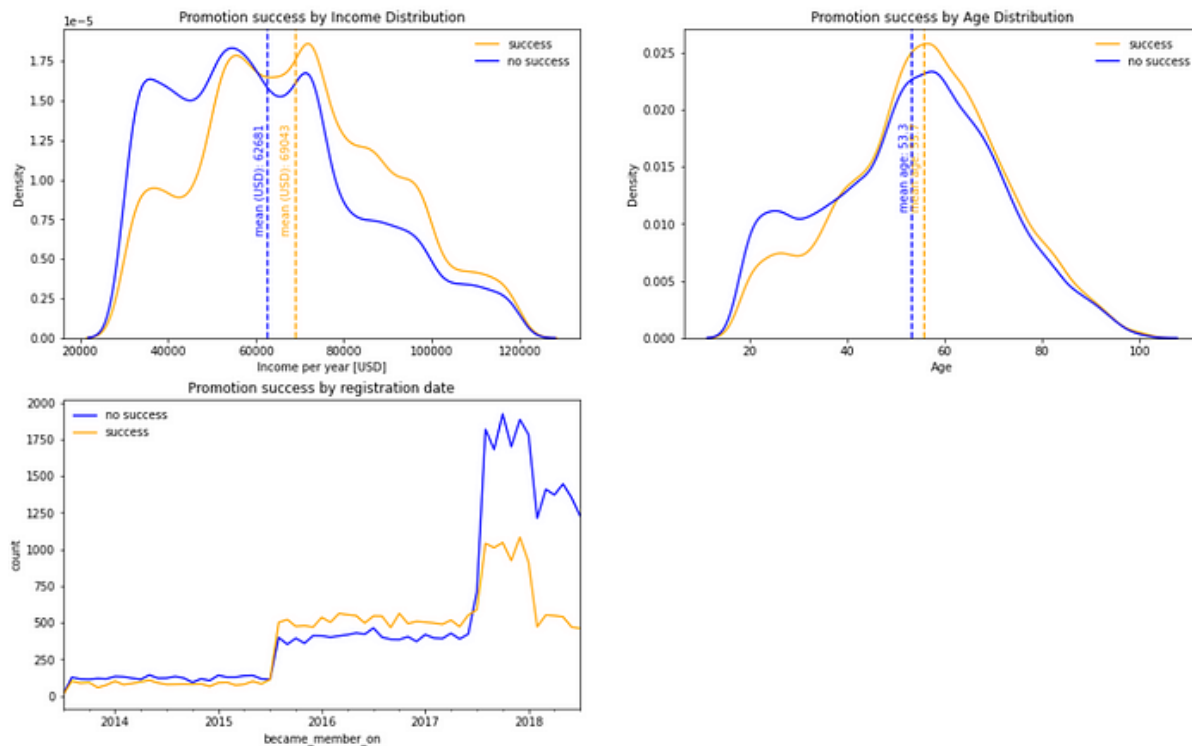


Figure 8— distribution of yearly income, age and registration date separated for success and no success

For **income** we can observe that customers of the success group **tend to have higher yearly income** than customers of the no success group. The mean yearly income for success group is 69,043 USD while for no success group it's 62,681 USD. The mean difference is 6,362 USD and **is statistically significant** (for p-value lower 0.05) between success and no success samples i.e. the two samples are not from the same population as the mean difference is caused by something other than chance. Furthermore the line chart shows that no success distribution of incomes is a positively skewed distribution while the success distribution of incomes tends to be a skew normally distribution

For **age** we can observe that customers of the success group **tend to be older** than customers of the no success group. The mean age for success group is 56 while for no success group it is 53 years. The mean difference is 2.4 years and **is statistically significant** (for p-value lower 0.05) between success and no success samples. Furthermore the line chart shows that there is a hump of young customers in the no success distribution of age while the success distribution of age tends to be a skew normally distribution

For **registration date** we can observe that customers who are registered for short time **tend to ignore** the offers **more often**. The mean difference **is statistically significant** (for p-value lower 0.05) between success and no success samples. Furthermore the line chart shows that for success group slightly more customers inregistered 2016 and 2017 and for no success group far more customers inregistered 2018. Thus it makes a difference if customer is a regular one or not.

We have for all three demographic variables a **significant mean difference** between success and no success class. These variables will serve as **appropriate target variables** for class prediction with machine learning algorithm.

## 4. Predict promotion success with Machine Learning

Now we build a model to predict which promotional offers will be successful when sending to new customer based on their age, income, registration date and gender. There are following **steps for modeling and evaluation** to be followed:

- **Preapre data frame for modeling** — Split master into 8 pieces where for each offer id (only BOGO and discount offer) one dataframe is created
- **Select features** — Select features with strong multicollinearity in realtion to target variable promotion success
- **Scale features** — Normalize features in range [0, 1] to bring every feature in same footing
- **Train test split of data** — Spilt data frames into train and test sets for validation purposes with equal test size of 0.2
- **Pick and tune algorithm** — Tune in total 6 algorithms on train data set for each single offer using GridSearchCV for hyperparameter tuning
- **Validate and evaluate** — Evaluate predictive performance of each algorithm using the test data set, compare evaluation metrics for each singe offer and then choose algorithm with best predictive performance and finally evaluate effect of hyperparameter tuning

Our predictive model is a classification model with binary output (1 for success and 0 for no success). We perform modeling on a diversified portfolio of algorithms with class output like Support Vector Machines and K Nearest Neighbors or with probability output like Logistic Regression and Random Forest. Below are listed the selected algorithms with few characteristics:

- [Gaussian Naive Bayes](#) (**GNB**) as baseline algorithm which is very fast and has a relatively high accuracy
- [Logistic Regression](#) (**LogR**) as second baseline algorithm which is very fast and has a relatively high accuracy
- [Support Vector Machines](#) (**SVM**) as slow classification algorithm which scales well and is less prone to overfitting and effect of outliers
- [Decision Tree](#) (**DT**) as fast classification algorithm which has high accuracy but is prone to overfitting
- [Random Forest](#) (**RF**) as slow classification algorithm which has high accuracy and is less prone to overfitting
- [K Nearest Neighbors](#) (**KNN**) as slow classification algorithm which has high accuracy and needs high computational ressources

We compute the metrics **Accuracy, F1, Recall, Precision** for every algorithm and offer and finally select algorithm with best predictive performance depending on the output of the confusion matrix and its metrics. Following steps are to be perused for every offer:

**Step 1** — Firstly we check if the predicted classes are roughly balanced. If Precision and Recall scores are roughly equal than the output is roughly balanced.



**Step 2** — In case one or more algorithms have roughly balanced classes we pick the algorithm with the highest Accuracy score. For algorithms with imbalanced output we ignore the Accuracy.

**Step 3** — In case there is more than one algorithm in range of the highest Accuracy and F1 scores we look closer at Precision and Recall scores. As for our business it is better to send out offers to customer who wouldn't respond to it than to not send out offers to customers that would respond, we prefer the classifier to predict more False Positives than False Negatives. Therefore we need the Recall score to be higher than the Precision score.

**Step 4** — In case all algorithms to be compared have imbalanced classes we pay more attention to F1 score and pick algorithm with highest F1 score where Recall is higher than Precision score.

#### offer 1

classifier	GNB	LogR	SVM	DT	RF	KNN	high score
accuracy	61.73	61.55	65.50	65.95	66.04	65.05	RF
f1-score	59.27	60.81	66.20	65.51	65.82	65.48	SVM
recall	57.94	62.06	70.28	67.29	68.04	68.97	SVM
precision	60.67	59.61	62.56	63.83	63.75	62.33	DT

#### offer 4

classifier	GNB	LogR	SVM	DT	RF	KNN	high score
accuracy	57.08	56.99	60.96	62.85	63.39	61.68	RF
f1-score	34.97	39.24	47.64	50.95	52.90	47.07	RF
recall	27.18	32.70	41.83	45.44	48.41	40.13	RF
precision	49.04	49.04	55.34	57.99	58.31	56.93	RF

#### offer 6

classifier	GNB	LogR	SVM	DT	RF	KNN	high score
accuracy	72.87	74.32	74.86	74.23	74.68	75.23	KNN
f1-score	82.17	84.76	84.94	84.10	85.51	84.91	RF
recall	83.72	95.63	94.90	91.25	100.00	93.32	RF
precision	80.68	76.11	76.87	77.99	74.68	77.89	GNB

#### offer 9

classifier	GNB	LogR	SVM	DT	RF	KNN	high score
accuracy	70.16	70.89	71.63	69.42	68.60	72.54	KNN
f1-score	77.26	79.97	79.19	77.24	79.20	80.26	KNN
recall	77.75	89.15	82.82	79.58	91.69	85.63	RF
precision	76.77	72.51	75.87	75.03	69.70	75.53	GNB

#### offer 2

classifier	GNB	LogR	SVM	DT	RF	KNN	high score
accuracy	65.10	67.54	69.45	70.72	71.26	69.99	RF
f1-score	62.95	67.51	69.39	70.29	71.10	70.89	RF
recall	57.88	65.84	67.61	67.61	69.03	71.33	KNN
precision	68.99	69.27	71.27	73.18	73.31	70.45	RF

#### offer 5

classifier	GNB	LogR	SVM	DT	RF	KNN	high score
accuracy	74.51	75.04	75.67	76.21	74.78	75.58	DT
f1-score	4.04	4.12	13.92	24.43	4.08	9.90	DT
recall	2.12	2.12	7.77	15.19	2.12	5.30	DT
precision	42.86	75.00	66.67	62.32	54.55	75.00	LogR

#### offer 7

classifier	GNB	LogR	SVM	DT	RF	KNN	high score
accuracy	74.75	75.20	75.38	74.93	72.50	75.29	SVM
f1-score	83.07	84.87	84.57	83.76	84.06	84.66	LogR
recall	85.45	95.90	93.03	89.18	100.00	94.03	RF
precision	80.82	76.11	77.51	78.96	72.50	76.99	GNB

#### offer 10

classifier	GNB	LogR	SVM	DT	RF	KNN	high score
accuracy	59.40	58.67	63.41	63.96	64.23	62.50	RF
f1-score	37.59	41.55	48.66	51.29	53.99	52.04	RF
recall	27.92	33.54	39.58	43.33	47.92	46.46	RF
precision	57.51	54.58	63.12	62.84	61.83	59.15	SVM

Figure 9 — overview evaluation metrics by offer ID and selected algorithms with best predictive performance

The result of the modeling gives us a classification algorithm with best predictive performance according to our needs for each individual offer data set. The selected algorithms are:

**offer 1** — [Random Forest \(RF\)](#) as it has a roughly balanced output with the highest Accuracy score where the Recall is higher than the Precision score

**offer 2** — [Random Forest \(RF\)](#) as it has a roughly balanced output with the highest Accuracy score and the second best Recall score

**offer 4** — [Random Forest \(RF\)](#) as it has an imbalanced output, as all the other algorithms too, with the highest F1 and Recall score

**offer 5** — [Decision Tree \(DT\)](#) as it has an imbalanced output, as all the other algorithms too, with the highest F1 and Recall score



**offer 6—** [Support Vector Machines \(SVM\)](#) as it has an imbalanced output as all other algorithms with Accuracy nearby the highest Accuracy score (KNN) and the second best F1-score after RF. We don't select RF as the model seems to be over-fitted because its Recall is 100 % with 0 True Negative ( $Tn$ ) and 0 False Negative ( $Fn$ )

**offer 7—** [Gaussian Naive Bayes \(GNB\)](#) as it has as single algorithm a roughly balanced output with an Accuracy nearby the highest Accuracy score (KNN). Furthermore the F1 score is also nearby the highest F1 score (LogR) and the Recall is higher than the Precision score

**offer 9—** [K Nearest Neighbors \(KNN\)](#) as it has a roughly balanced output with the highest Accuracy and F1 score and has the Recall higher than the Precision score

**offer 10—** [Random Forest \(RF\)](#) as it has an imbalanced output, as all the other algorithms too, with the highest F1 and the second best Recall score

Next we denote **best values of hyperparameters** for the selected algorithms that cross-validation experiments found:

- [Random Forest \(RF\)](#) for **offer 1, 2, 4, 10**: `{'bootstrap': True, 'max_depth': 6, 'max_features': 2, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 20}`
- [Decision Tree \(DT\)](#) for **offer 5**: `{'criterion': 'entropy', 'max_depth': None, 'max_features': 0.5, 'min_samples_leaf': 50, 'min_samples_split': 100}`
- [Support Vector Machines \(SVM\)](#) for **offer 6**: `{'C': 10, 'gamma': 1, 'kernel': 'rbf'}`
- [Gaussian Naive Bayes \(GNB\)](#) for **offer 7**: `{}`
- [K Nearest Neighbors \(KNN\)](#) for **offer 9**: `{'metric': 'manhattan', 'n_neighbors': 150, 'p': 0.5, 'weights': 'distance'}`

In the following figure we can see the range of parameters to be available for hyperparameter tuning. The selected best parameters which were optimized by cross-validated grid-search (GridSearchCV) are marked in red.

### Random Forest (RF) – offer 1, 2, 4, 10

```
parameters = {"bootstrap": [True],
              "max_depth": [2, 6, 10],
              "max_features": [0.5, 1, 2],
              "min_samples_leaf": [1, 5],
              "min_samples_split": [2, 5],
              "n_estimators": [10, 20]}
RF = RandomForestClassifier()
clf = GridSearchCV(RF, parameters, scoring = "roc_auc", cv = 4, n_jobs = 4, verbose = 2)
```

### Decision Tree (DT) – offer 5

```
parameters = {"criterion": ("gini", "entropy"),
              "max_features": [0.5, 0.75, None],
              "max_depth": [0, 10, None],
              "min_samples_split": [100, 20, 2],
              "min_samples_leaf": [50, 10, 1]}
DT = DecisionTreeClassifier(random_state = 0)
clf = GridSearchCV(DT, parameters, scoring = "roc_auc", cv = 4, n_jobs = 4, verbose = 2)
```

### Support Vector Machines (SVM) – offer 6

```
parameters = {"kernel": ["rbf"],
              "C": [0.1, 10],
              "gamma": [1, 10]}
SVM = SVC()
clf = GridSearchCV(SVM, parameters)
```

### Gaussian Naive Bayes (GNB) – offer 7

```
parameters = {}
GNB = StratifiedKFold(n_splits = 4)
clf = GridSearchCV(GaussianNB(), cv=GNB, param_grid=parameters)
```

### K Nearest Neighbors (KNN) – offer 9

```
parameters = {"n_neighbors": [50, 100, 150],
              "p": [0.5, 1, 2],
              "weights": ["uniform", "distance"],
              "metric": ["euclidean", "manhattan"]}
KNN = KNeighborsClassifier()
clf = GridSearchCV(KNN, parameters, scoring = 'roc_auc', cv=4, n_jobs=4, verbose=2)
```


 Best parameters of the estimator which were optimized by cross-validated grid-search (GridSearchCV)

Figure 10 — overview selected best parameter of estimator (GridSearchCV)

In the end we measure the effect of hyperparameter tuning performed with GridSearchCV. Here we compare the computed metrics Accuracy, F1 score, Recall and Precision once for best parameters, using GridSearchCV, and once for default parameters. The outcome is a percentage of how much the model performed better using GridSearchCV.

**offer 1 — [Random Forest \(RF\)](#)** — hyperparameter tuning was fully successful and increased metrics Accuracy, F1-Score, Recall and Precision by 3 to 5 %

**offer 2 — [Random Forest \(RF\)](#)** — hyperparameter tuning was fully successful and increased metrics Accuracy, F1-Score, Recall and Precision by 1 to 3 %

**offer 4 — [Random Forest \(RF\)](#)** — hyperparameter tuning was successful and increased metrics Accuracy, F1-Score and Precision by 0 to 2 %. Recall score decreased by 1 % which can be neglected

**offer 5 — [Decision Tree \(DT\)](#)** — hyperparameter tuning was partially successful and increased metrics Accuracy and Precision by 13 and 32 %. F1 and Recall score decreased by 7 and 17 %. It has to be considered to probably tune the algorithm once again with slightly different parameters

**offer 6 — [Support Vector Machines \(SVM\)](#)** — hyperparameter tuning was modestly successful and increased metrics Accuracy, F1-Score and Precision by 0 to 2 %. Recall score decreased by 2 % which can be neglected

**offer 7 — [Gaussian Naive Bayes \(GNB\)](#)** — no hyperparameter tuning possible for this classifier

**offer 9 — [K Nearest Neighbors \(KNN\)](#) —** hyperparameter tuning was fully successful and increased metrics Accuracy, F1-Score, Recall and Precision by 2 to 8 %

**offer 10— [Random Forest \(RF\)](#) —** hyperparameter tuning was successful and increased metrics Accuracy, F1-Score and Precision by 1 to 3 %. Recall score decreased by 1 % which can be neglected

#### offer 1 - RF

classifier	best params	default params	improvement [%]
accuracy	66.04	62.62	3.4
f1-score	65.82	61.97	3.8
recall	68.04	63.36	4.7
precision	63.75	60.64	3.1

#### offer 4 - RF

classifier	best params	default params	improvement [%]
accuracy	63.39	62.22	1.2
f1-score	52.90	52.87	0.0
recall	48.41	49.89	-1.5
precision	58.31	56.22	2.1

#### offer 6 - SVM

classifier	best params	default params	improvement [%]
accuracy	74.86	74.14	0.7
f1-score	84.94	84.91	0.0
recall	94.90	97.45	-2.5
precision	76.87	75.23	1.6

#### offer 9 - KNN

classifier	best params	default params	improvement [%]
accuracy	72.54	66.94	5.6
f1-score	80.26	75.48	4.8
recall	85.63	78.03	7.6
precision	75.53	73.09	2.4

#### offer 2 - RF

classifier	best params	default params	improvement [%]
accuracy	71.26	68.99	2.3
f1-score	71.10	68.57	2.5
recall	69.03	66.02	3.0
precision	73.31	71.32	2.0

#### offer 5 - DT

classifier	best params	default params	improvement [%]
accuracy	76.21	63.60	12.6
f1-score	24.43	31.13	-6.7
recall	15.19	32.51	-17.3
precision	62.32	29.87	32.4

#### offer 7 - GNB

classifier	best params	default params	improvement [%]
accuracy	74.75	74.75	0.0
f1-score	83.07	83.07	0.0
recall	85.45	85.45	0.0
precision	80.82	80.82	0.0

#### offer 10 - RF

classifier	best params	default params	improvement [%]
accuracy	64.23	62.50	1.7
f1-score	53.99	53.24	0.8
recall	47.92	48.75	-0.8
precision	61.83	58.65	3.2

Figure 11 — effect of hyperparameter tuning as comparison of best parameters (GridSearchCV) and of default parameters

Thus the results of the hyperparameter tuning are satisfying for algorithms **RF**, **SVM**, **GNB** and **KNN**. For Decision Tree (**DT**) we take into consideration to tune the algorithm once again with slightly different parameters.

## 5. Deployment of Machine Learning models for business

In real business life companies like Starbucks use Data Science to investigate customer preferences on specific promotional offers before sending out an offer portfolio to these customers. Main focus of our promotional strategy is to push the customer to transact more purchases independently of reward size. We filter out which customers would likely complete a specific offer. Thus as benefits the company saves unnecessary expenses by not sending offers to customers who very likely won't respond to them. Besides it is pleasant for the customer to not get stalked by any additional advertising he is not interested in.

Now we take **10 fictitious customers** with information of their age, income, registration date and gender and **predict** on basis of this data which offer would be completed by whom employing the Machine Learning models we selected before.

	age	became_member_on	income year (USD)	gender	offer_01	offer_02	offer_04	offer_05	offer_06	offer_07	offer_09	offer_10	sum	success
customer_01	20	2013-07-29	60000	M	0	0	0	0	0	0	0	0	0	0
customer_02	25	2016-07-29	40000	F	0	0	0	0	0	0	0	0	0	0
customer_03	30	2018-07-26	100000	F	0	0	0	0	0	0	0	0	0	0
customer_04	35	2013-07-29	100000	F	0	1	0	1	1	1	1	0	5	5
customer_05	45	2016-07-29	70000	O	1	1	1	1	1	1	1	1	8	8
customer_06	50	2018-07-26	40000	M	0	0	0	0	0	0	0	0	0	0
customer_07	55	2013-07-29	30000	F	0	0	0	0	0	1	0	0	1	1
customer_08	60	2015-07-29	80000	F	1	1	1	1	1	1	1	1	8	8
customer_09	70	2017-07-29	40000	F	0	0	0	0	0	0	0	0	0	0
customer_10	80	2017-07-29	100000	M	0	1	0	0	0	0	1	0	2	2

1

predicted promotional offer will be completed (totally 24 of 80)

0

0

predicted promotional offer will not be completed (totally 56 of 80)

Figure 11— dataframe with prediction offer success for 10 sample customers

In the end we predicted that altogether 24 offers would be completed by those 10 customers. Without customer analysis we would send out 80 offers (10 customers \* 8 offers) instead of these 24 predicted. Thus we only send out 30 % of the offers due to our target-oriented procedure and would likely achieve almost the number of completions as if we would have sent all offers to all customers.

## Conclusion and Reflection

We saw that the **popularity** of the offers is depending on **offer type** and **offer id**. Furthermore we found out that there is a **statistically significant mean difference** between offer **success group** and **no success group** for the customer demographics **income**, **age** and **registration date**.

Finally we built a **well working classification model to predict promotional offer success** for new customers based on their demographical data age, income, registration date and gender. For each offer we have one classification model with satisfying results at hyperparameter tuning. This procedure can be used in real business life as promotional strategy with the benefits that company saves expenses and customers feel less stalked by undesirable advertising.

It was surprising that the customer demographics income, age and registration date show a clear statistical significant mean difference between offer success group and no success group. Furthermore prediction of offer success was quite straightforward on basis of a few selected target variables. Most real life classification problems won't work that simple.

In another subsequent analysis we could **improve predictive performance** by:

- finding or creating new variables with optional customer demographics to be used as additional feature for the modeling part or
- by looking for other classification algorithm with high predictive performance and successful hyperparameter tuning.

Furthermore we could **investigate more subjects** we might be interested in as:

- predicting expected transaction amount one customer would spend when completing an offers and determining which offers are most successful in generating revenue,
- analyzing how informational offers influence the purchasing behaviour of a customer or
- finding out which meta data like amount of discount or duration of offer has most influence on offer success.