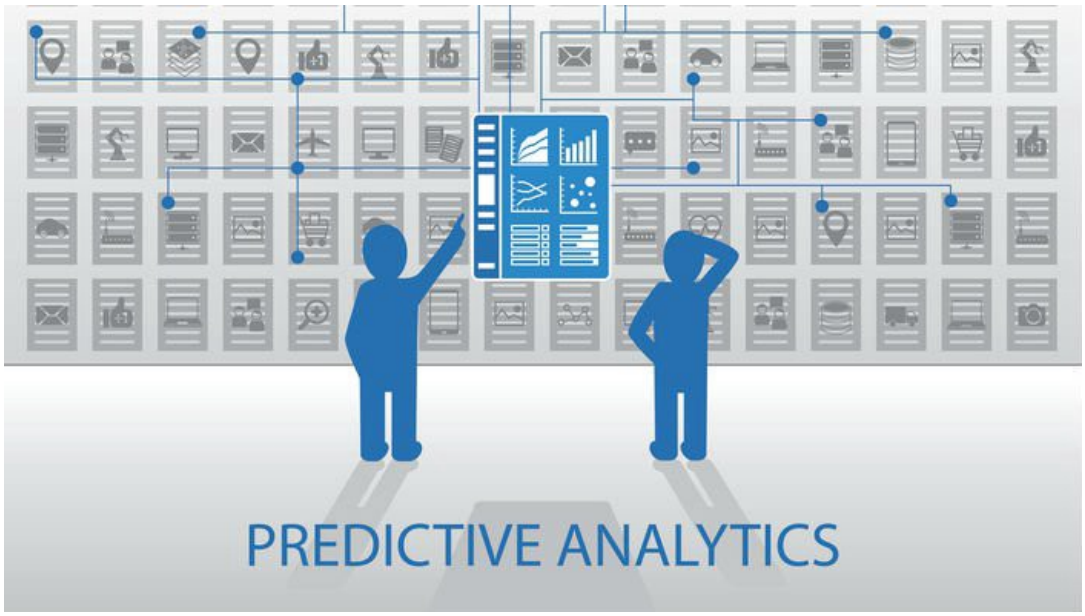
Recommend the Best Channels for Contact

Ryan Aminollahi
Feb 7, 2018 · 10 min read

Predictive Customer Analytics—Part II



Will You Become My Customer?

How do businesses acquire customers? What is the step-by-step process they follow? First, they need to identify their markets and their prospects. The business needs to identify them and qualify them as prospects. Then the business needs to reach out to them with appropriate advertisements and offers. It should use channels of communications that are effective and efficient. They should entice the prospects to visit the on-line website and take a look at the products and services.

When the prospect is interested in the product, the business should engage them, answer their questions, and make offers to help them buy the product.

High propensity prospects

We have a product, for example high-end laptops. Who are the potential customers who have a higher likelihood to buy from us? The middle-aged guy with the family and the decent income? Or The college student with a low-paying job? This inference is based on its demographics. The first big challenge any marketing department has is to identify prospective customers who have a higher likelihood to buy from us.

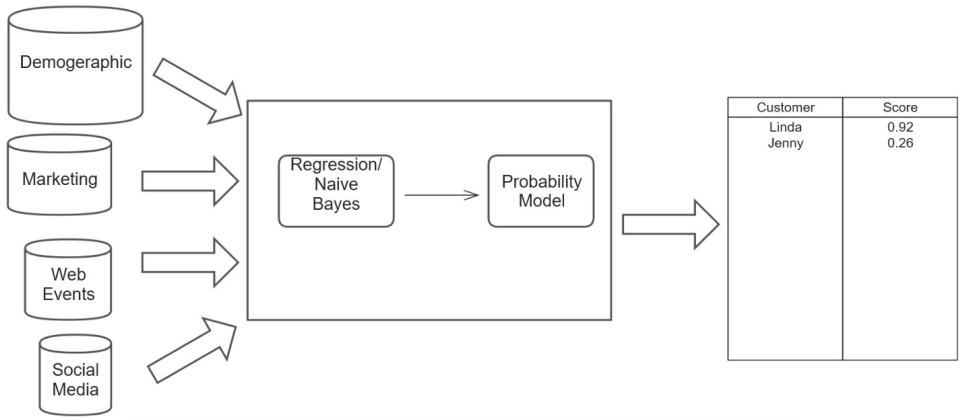
The goal of this use case is to generate a propensity score for each of our prospects

identified by the marketing department. The propensity score can be a binary, zero, or one. Or better yet, it can be a continuous score all the way from zero to one. What data would we use? At this stage, the only set of data that is available to us is prospect demographic attributes, attributes such as age, salary, family, etc.

Regarding events, these prospects may or may not have been involved in any events with our business, so they become optional data. One way to use it would be by using binary flags like did the prospect ever browse our website.

- Did he/she ever respond to our emails?
- Did he/she ever tweet about our business or products?

Of course, all this past data will be tagged with eventual results of the campaigns in the past. What algorithms would we try? The class of algorithms we would use would be regression to come up with a prospect score or we can use *Naive Bayes classification* to come up with the probability of prospect conversion.



Ideally, we want a score between zero and one. This model is then used to score our future prospects. The marketing department can then prune this list to work it off-list or a top X list based on the score. So what is the call to action? We can execute this use case on a periodic basis or whenever our marketing department comes with a list of prospects. We use previous data to build a propensity model. Then based on that model, we generate a score for each of the prospects.

This will then be used by our marketing department to reach out to prospects with offers and deals.

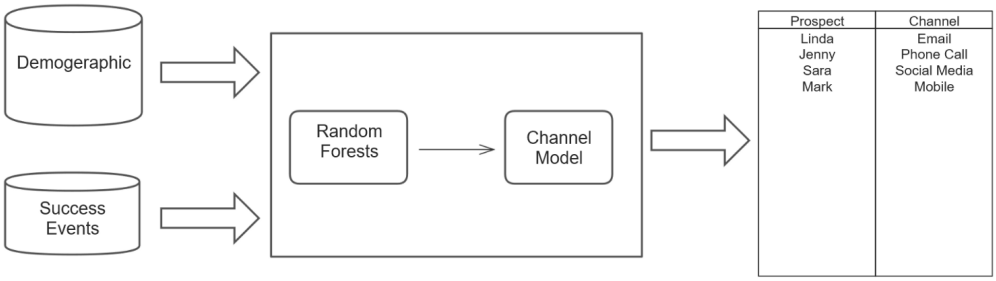
Recommend the best channels for contact

Once we have a list of prospects to reach out we will need to decide how best to communicate with them. There are multiple channels available, like phone calls, emails, mobile, or targeted ads on internet or social media. But different people react differently to different media. Someone likes to pay attention to the marketing emails he/she gets. He always clicks them and checks them out. another one, on the other hand, filters such emails, that go to her/his junk folder. She/He, however, are shown a propensity to respond at the pop-up during her browsing based on her recent searches.

The goal of this use case is to recommend the best channel for contacting each prospective customer. With so many different mediums available, it is important to

target customers in such a way that will receive the most attention and the highest return on investment. What data would we use? Prospect data is always going to be there. We should also use data about past successful events, events in the past that we reached out to a specific prospect in a specific channel and the prospect actually converted.

This data gives us the idea of who will convert to which channel. Using this data, we will be able to build a model that predicts the channel for each of the future prospects. We will then use this model for future prospects to do action prediction. Which algorithms would we use? This is a classic classification problem, so we would experiment with any of the previous algorithms and check for their accuracy. We build a model based on the past data that would recommend a media for contact for each prospective customer.



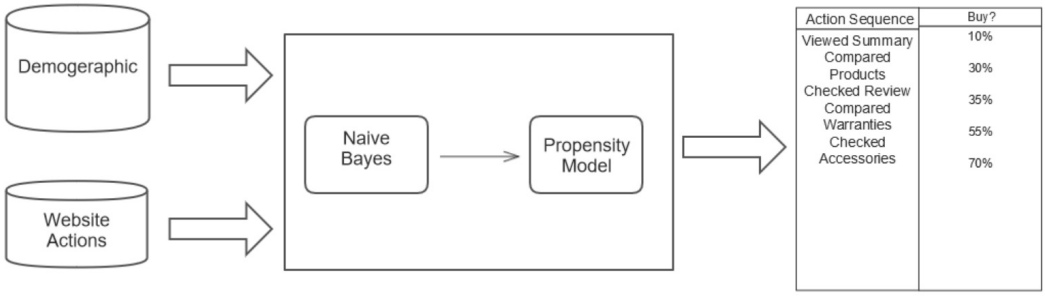
So what is the call to action? We will use past data to build a classification model that would predict the best channel of contact for each of the prospects. This will help ourr marketing team to device targeted campaigns that would help them reach out to each of the prospects in those specific channels.

Offer chat based on visitor propensity

We have on-line sales rep ready to engage our website visitors, and entice customers to buy our products. But we typically have so many visitors, and most of them are window shoppers. We want to offer our sales rep through chat, only to visitors who are serious about buying on that given day. We don't want to waste our sales rep time.

Imagine, customer comparing products, for example laptops. Does that mean he/she's making a decision? He/She starts reading reviews. Is he serious about buying? He/she's checking out warranties. Has he/she made a decision? How do we decide? The goal of this use case is to iteratively predict a customer's propensity to buy, based on real time actions the customer does on our website. As the customer does activities on our website, we want to keep computing and revising the propensity score.

The source data for this use case are prospect attributes along with data about progressive action a prospect engages once he is on our website. This includes the products he checks out, and the actions he does, like does he look at reviews? Does he compare products? Is he interested in warranty? These are measured through web click events generated by the browser, and sent to the website. we need to use past data for each prospect for each visit, and the final result of the visit, like did he buy it or not? This is a classification use case.



Naive Bayes will be the most suited algorithm, since it provides a probability score. We will use past data about customer demographics, and website action applied in Naive Bayes algorithm, and build a propensity model. Then we will use the propensity model to predict the propensity to buy for our current website visitor. Call to action. We build a model offline based on past data to suggest a propensity score based on the customer's action on the website.

When a new prospect visits our website, the web click events are progressively collected. Each time a new event is collected, say, viewing reviews or comparing products, the model is run against the collected data to compute the propensity score once again. Once the propensity score reaches a certain threshold, the online store can make a decision to offer chat. This is something that we can prototype.

I'll show you how to use Python to code and implement this specific use case.

We're going to implement the use case about predicting the propensity of visitors to our website in real-time. When visitors come to our website, they start checking out different links as they explore the product. And what we want to do is, in real-time, based on their actions, predict their propensity and see whether we want to offer chat or not.

Here a data sample of customer browsing

	A	B	C	D	E	F	G	H	I	J	K	L
1	SESSION_ID	IMAGES	REVIEWS	FAQ	SPECS	SHIPPING	BOUGHT_	COMPARE	VIEW_SIMILAR	WARRANT	SPONSORI	BUY
2	1001	0	0	1	0	1	0	0	0	1	0	0
3	1002	0	1	1	0	0	0	0	0	0	1	0
4	1003	1	0	1	1	1	0	0	0	1	0	0
5	1004	1	0	0	0	1	1	1	0	0	0	0
6	1005	1	1	1	0	1	0	1	0	0	0	0
7	1006	1	0	0	1	0	1	1	0	0	0	0
8	1007	1	1	0	1	1	1	0	1	0	1	0
9	1008	0	1	1	1	1	0	1	0	1	1	1
10	1009	1	0	1	0	1	0	1	0	1	1	0
11	1010	1	1	1	1	0	0	1	1	0	1	0
12	1011	1	0	0	1	0	1	0	0	0	1	0
13	1012	0	0	0	1	1	0	0	1	1	1	0
14	1013	0	0	1	0	0	1	1	0	1	0	0
15	1014	1	0	1	0	1	1	1	0	0	1	1
16	1015	1	1	0	1	1	1	1	1	1	0	0
17	1016	1	1	0	0	1	1	1	1	1	0	1
18	1017	0	0	0	0	1	1	0	1	0	0	0
19	1018	0	0	1	0	1	0	0	0	0	1	0
20	1019	1	1	1	0	0	1	0	0	0	0	1
21	1020	0	0	1	0	0	1	0	0	1	1	0

This data contains information about all past sessions by different users. It starts with a session id, then it has a number of boolean variables, which will be our feature variables. These boolean variables carry a one or zero based on the action performed by the visitor.

An images one means the visitor actually viewed various images of the product. A reviews one means the visitor actually viewed reviews for the product. Similarly, we have FAQ, specs, shipping, bought_together, comparison of products, and so on. Finally, there is the target variable, which indicates whether the visitor actually bought the product or did not buy the product. So this is going to be our data set that we're going to use for building the model.

In real world, we would be using a really large data set, if we want to get real accurate predictions.

we are starting off with importing a number of python libraries , and start off by importing the browsing.csv into a dataframe called prospect_data.

```
from pandas import Series, DataFrame
import pandas as pd
import numpy as np
import os
import matplotlib.pylab as plt
from sklearn.model_selection  import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
import sklearn.metrics

prospect_data = pd.read_csv("browsing.csv")
```

```
prospect_data.dtypes
```

Then we just look at the data types to make sure that the data has been loaded correctly.

```
SESSION_ID         int64
IMAGES             int64
REVIEWS            int64
FAQ                int64
SPECS              int64
SHIPPING           int64
BOUGHT_TOGETHER    int64
COMPARE_SIMILAR    int64
VIEW_SIMILAR       int64
WARRANTY           int64
SPONSORED_LINKS    int64
BUY                int64
dtype: object
```

The data contains information about the various links on the website that are clicked by the user during his browsing. This is past data that will be used to build the model.

- Session ID : A unique identifier for the web browsing session

- Buy : Whether the prospect ended up buying the product

- Other columns : a 0 or 1 indicator to show whether the prospect visited that particular page or did the activity mentioned.

```
In [28]: # Look at the top records to understand how the data looks like.
         prospect_data.head()
```

Out[28]:

SESSION_ID	IMAGES	REVIEWS	FAQ	SPECS	SHIPPING	BOUGHT_TOGETHER	COMPARE_SIMILAR	VIEW_SIMILAR	WARRANTY	SPONSORED_LINKS	BUY
1001	0	0	1	0	1	0	0	0	1	0	0
1002	0	1	1	0	0	0	0	0	0	1	0
1003	1	0	1	1	1	0	0	0	1	0	0
1004	1	0	0	0	1	1	1	0	0	0	0
1005	1	1	1	0	1	0	1	0	0	0	0

```
#Do summary statistics analysis of the data
prospect_data.describe()
```

```
In [29]: #Do summary statistics analysis of the data
         prospect_data.describe()
```

Out[29]:

	SESSION_ID	IMAGES	REVIEWS	FAQ	SPECS	SHIPPING	BOUGHT_TOGETHER	COMPARE_SIMILAR	VIEW_SIMILAR	WARRANTY	SPONSO
count	500.000000	500.000000	500.0000	500.000000	500.0000	500.000000	500.000000	500.000000	500.000000	500.000000	
mean	1250.500000	0.510000	0.5200	0.440000	0.4800	0.528000	0.500000	0.580000	0.468000	0.532000	
std	144.481833	0.500401	0.5001	0.496884	0.5001	0.499715	0.500501	0.494053	0.499475	0.499475	
min	1001.000000	0.000000	0.0000	0.000000	0.0000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1125.750000	0.000000	0.0000	0.000000	0.0000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	1250.500000	1.000000	1.0000	0.000000	0.0000	1.000000	0.500000	1.000000	0.000000	1.000000	
75%	1375.250000	1.000000	1.0000	1.000000	1.0000	1.000000	1.000000	1.000000	1.000000	1.000000	
max	1500.000000	1.000000	1.0000	1.000000	1.0000	1.000000	1.000000	1.000000	1.000000	1.000000	

For this, We are going to be using the plain test strip method in this curriculum, and We are splitting in the ratio of 70 to 30. And We are going to check out if the sizes are what we expect them to be, 350 to 150 sounds about right. Next, We move on to model building. We are using the naive_bayes alogrithm available in the sklearn library, the Guassian naive_bayes. We first create the naive_bayes classifier, then build a model using the fit method, applying it on the training prediction analysis and the training targets.

Perform Correlation Analysis

prospect_data.corr()['BUY']

```
SESSION_ID        0.026677
IMAGES            0.046819
REVIEWS           0.404628
FAQ              -0.095136
SPECS             0.009950
SHIPPING         -0.022239
BOUGHT_TOGETHER  -0.103562
COMPARE_SIMILAR   0.190522
VIEW_SIMILAR     -0.096137
WARRANTY          0.179156
SPONSORED_LINKS   0.110328
BUY               1.000000
Name: BUY, dtype: float64
```

Looking at the correlations above we can see that some features like REVIEWS, BRO_TOGETHER, COMPARE_SIMILAR, WARRANTY and SPONSORED_LINKS have medium correlation to the target variable. We will reduce our feature set to that list of variables.

#Drop columns with low correlation

predictors = prospect_data[['REVIEWS','BOUGHT_TOGETHER','COMPARE_SIMILAR','WARRANTY','SPONSORED_LINKS']]

targets = prospect_data.BUY

Training and Testing Split

We now split the model into training and testing data in the ratio of 70:30

pred_train, pred_test, tar_train, tar_test = train_test_split(predictors, targets, test_size=.3)

print("Predictor—Training : ", pred_train.shape, "Predictor—Testing : ", pred_test.shape)

```
Predictor - Training :  (350, 5) Predictor - Testing :  (150, 5)
```

Build Model and Check Accuracy

from sklearn.naive_bayes import GaussianNB

classifier=GaussianNB()

classifier=classifier.fit(pred_train,tar_train)

predictions=classifier.predict(pred_test)

#Analyze accuracy of predictions

sklearn.metrics.confusion_matrix(tar_test,predictions)

```
array([[76, 18],
       [24, 32]])
```

sklearn.metrics.accuracy_score(tar_test, predictions)

```
0.71999999999999997
```

Instead of doing a Yes/No prediction, we can instead do a probability computation to show the probability for the prospect to buy the product

pred_prob=classifier.predict_proba(pred_test)

pred_prob[0,1]

```
0.35088586866049354
```

```
0.35088588660049334
```

The probability above can be read as 35% chance that the prospect will buy the product.

Real time predictions

Now that the model has been built, let me use it for real time predictions. So when the customer starts visiting the pages one by one, we collect that list and then use it to compute the probability. We do that for every new click that comes in.

The prospect just came to our website. There are no significant clicks. Let's compute the probability. The array of values passed has the values for REVIEWS, BOUGHT_TOGETHER, COMPARE_SIMILAR, WARRANTY and SPONSORED_LINKS. So the array is all zeros to begin with

browsing_data = np.array([0,0,0,0,0]).reshape(1, -1)

print("New visitor: propensity :",classifier.predict_proba(browsing_data)[:,1]

```
New visitor: propensity : [ 0.03961746]
```

So the initial probability is 4%. Now, suppose the customer does a comparison of similar products. The array changes to include a 1 for that function. The new probability will be

browsing_data = np.array([0,0,1,0,0]).reshape(1, -1)

print("After checking similar products:
propensity :",classifier.predict_proba(browsing_data)[:,1])

```
After checking similar products: propensity : [ 0.09898671]
```

It goes up. Next, he checks out reviews.

browsing_data = np.array([1,0,1,0,0]).reshape(1, -1)

print("After checking reviews: propensity :",classifier.predict_proba(browsing_data)[:,1])

```
After checking reviews: propensity : [ 0.57538836]
```

It shoots up to 50+%. We can have a threshold for when we want to offer chat. We can keep checking this probability against that threshold to see if we want to popup a chat window.

This example shows how we can use predictive analytics in real time to decide whether a prospect has high propensity to convert and offer him a chat with a sales rep/agent.

So, remember that when these ones keep coming in, it is not important for the propensity to always go up, it might even go down. It all depends upon how the data is. So then we can, at any point, decide when we want to offer a chat window.