Predicting product cancellations for sales retention

Nathan Kelly[1[0000−1111−2222−3333]] and Hisham Ihshaish[2,3[1111−2222−3333−4444]]

Address of UWE Hisham.Ihshaish@uwe.ac.uk
http://www1.uwe.ac.uk/et/csct/aboutthedepartment.aspx

Abstract. Investigate machine learning techniques to predict the likelyhood of a product cancellation based on existing sales and amount spent in annual maintenance. To enable the targeting of retention resources and offers to those customers who are predicted as at risk.

Keywords: Sales · Retention · Machine learning · Predict · cancellations · Sales · Customers

1 Background

All businesses have problems with retention, keeping those customers they worked hard to sell to on-board when the initial excitement of the new has worn off. All too often the process of retention starts after the customer has already contacted the company to cancel. Then deep cuts in ongoing costs are offered, new features are promised and everything is reactive to the issues rather than proactive. The impact of a predictive model that can score the probability of cancellation would be that those working on retention can target the organisations directly. Proactively contacting them for information on what issues they may have and attempt to solve any problems in order to keep the sale. The added advantage is that features and issues are gleaned from the less vocal customers, the silent majority who leave with no explained reason and have a competitor lined up already.

2 The training data

2.1 Getting the data

The data for this study has been taken from a relational database management system (RDMS). The normalized tables have to first be joined into a single flat source. As we expect this data to be used repeatedly for ongoing model training it makes sense to create a denormalized view in the DBMS system [1]. The rows of interest from the joined tables can then be pivoted into columns for the view. This technique worked for the data in this paper, but for larger datasets the alternative of creating a flat table on a schedule or on data update can be used. The setup of this is out of the scope of this paper, but an example of the SQL used to create the dataset is provided in appendix 1.1.

2.2 Data description

The data in the export is described below, with an example:

Table 1. Data description

Column	Example	Description
`customerId`	Id of the customer in the system	`1`
`amountTotal`	Total value, in currency	`1000`
`salesTotal`	Total sale value, in currency	`1000`
`annualTotal`	Total annual value, in currency	`100`
`monthlyTotalProduct1`	Total monthly value, in currency	`10`
`monthlyTotalProduct2`	Total monthly value, in currency.	`10`
`extraItemsTotal`	Any extra items value, in currency.	`10`
`monthsSinceStarted`	Total number of months as customer	`15`
`daysToPay`	Mean number of days to pay bills	`2`
`creditedAmount`	Total value of credited amount, in currency	`15`
`isCancelled`	Boolean. Dependant variable customer cancelled product(s)	`true`
`hasProduct1`	Boolean, customer has product 1	`true`
`...`	...	`...`
`hasProduct12`	Boolean, customer has product 12	`false`
`administrativeArea`	String, customer location by country admin area	`Bristol`
`subCountryArea`	String, customer location by sub country area	`South West`
`country`	String, customer location by country	`England`
`firstInvoiceDate`	Date ISO, first date of billing	`2018-07-08`

Column `amountTotal` is all the invoices sum'ed together. `salesTotal` is the value of each product sale combined. `annualTotal` is the recurring annual payments (if any).

Columns `hasProduct1`...`hasProduct12` represent the existence of purchased products for the company and is repeated `n` times depending on number of products sold by the company being analysied.

Column `extraItemsTotal` contains any non product specific items sold to the customer that isn't monthly on going fees. Eg a bundle of phone minutes at a reduced rate.

Columns `monthlyTotalProduct1` and `monthlyTotalProduct2` Represent products which have a monthly bill. In this case only product 1 and 2 have monthly bills.

2.3 Data problems

Before pulling the csv file into a Jupyter notebook, it is noticed that the value for each organisation's mean time to pay in months, `monthsSincePaid` is sometimes null. This can be explained by the date the invoice was paid not being recorded. Or that the sale is new and so no paid date has yet been set. In total 14% of the total example dataset has missing values. It makes sense to remove the new customers who have yet to pay, so the sql limits in the where clause. This still leaves null values. They will be left in the dataset and later in the paper we will compare the various imputation methods as outlined in: Missing Data: Our View of the State of the Art [2]. It could well be that the mean value of 1 month to pay is a reasonable assumption.

3 Feature engineering and selection

The next step is to look at the data in the Jupyter Notebook. This is a supervised learning problem, with the discrete `isCancelled` label as our dependant variable: There are 4676 records, 2076 are customers who have cancelled a product at some

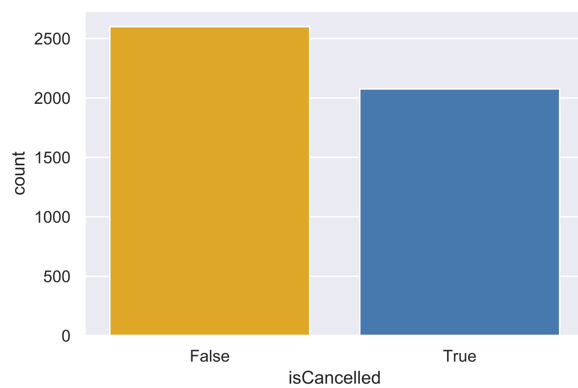

Fig. 1. Dependant variable spread

point. Product 2 is the most common (2874), followed by product 1 (1878) and product 12 (1383). What the products are and do is not considered in this study, but if it was then weighting could be given to certain products at this point. For example if product 4 was a low volume high cost service or item it's status could be flagged with a categorical value. We do though have the total sale value in currency for the customer. This could be in itself enough of a weighting. The dataset is cleaned at the sql stage, a quick check for missing values gives us confidence with the data:[4]:

```
nans = lambda df: df[df.isnull().any(axis=1)]
nans(dataFrame)
```

3.1 Initial pipeline in sklearn

First off, we can take our partially cleaned data from the sql source and get it in a pipeline for further investigation and processing. Using `GridSearchCV` to cross validate our training data and give us a precision score on the test split. Appendix 1.2. We'll start by dropping the catagorical features and columns `customerIdKey`, `firstInvoiceDate` to simplfy the baseline. Randomly selecting a fast classifier `LogisticRegression`. Results are:

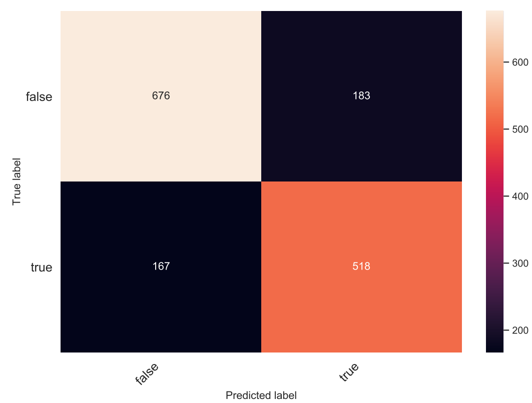

Fig. 2. Initial confusion matrix

	precision	recall	f1-score	support
False	0.80	0.79	0.79	859
True	0.74	0.76	0.75	685
avg / total	0.77	0.77	0.77	1544

The main score we are interested in here is the precision, our aim now is to reduce the number of false positives in the classification so we don't send our retention staff after customers who aren't going to leave.

3.2 Categorical data

Some of the features being ignored currently are categorical. `administrativeArea`, `subCountryArea` and `country`. A quick look at them reveals:

3.3 Inital feature selection

Following the guidance in the conclusion of "An Introduction to Variable and Feature Selection" [5], we'll rank the features first using sklearn's `SelectKBest` method with the classification algorithms: `chi2`, `f`$_{c}lassif$, `mutual`$_{i}nfo_{c}lassif$.`Discuss cross validation and`

1. `Database normalization and de-normalization, https://www.techrepublic.com/blog/software-engineer/c`
   `Last accessed 08 Jul 2018`

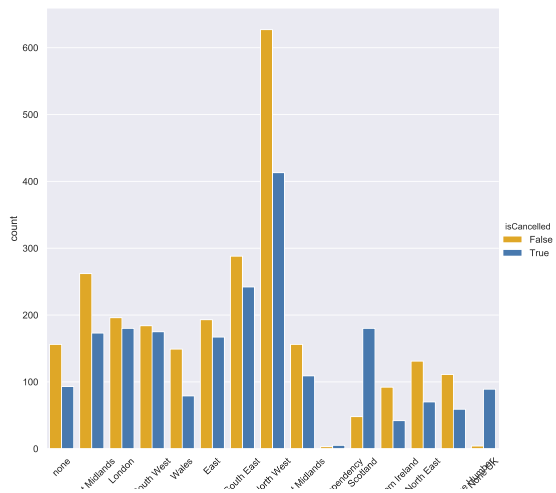

Fig. 3. Sub country range

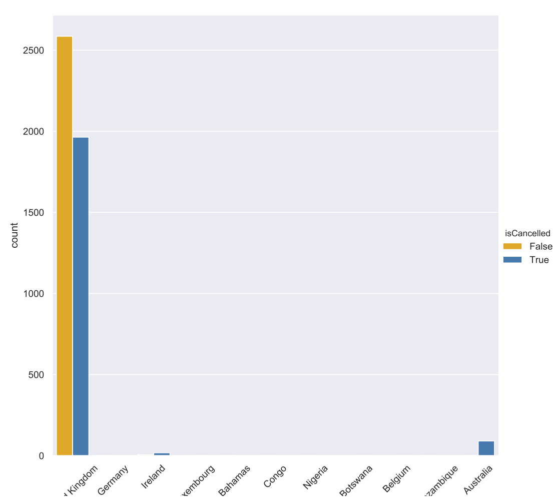

Fig. 4. Country range

2. Joseph L. Schafer and John W. Graham: Missing Data: Our View of the State of the Art, http://www.academia.edu/1045565/Missing_Data_Our_View_of_the_State_of_the_Art. Last accessed 08 Jul 2018
3. A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers https://www.researchgate.net/profile/Kedar_Potdar/publication/320465713_
Last accessed 09 Jul 201
4. Python Pandas How to select rows with one or more nulls from a DataFrame without listing columns explicitly? https://stackoverflow.com/a/44702024. Last accessed 10 Jul 2018
5. An Introduction to Variable and Feature Selection http://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf. Last accessed 10 Jul 2018
6. The importance of proper cross-validation and experimental design https://followthedata.wordpress.com/2013/10/30/the-importance-of-proper-cross-validation-and-exper
Last accessed 23 Jul 2018
7. PCA before Random Forest Regression provide better predictive scores for my dataset than just Random Forest Regression, how to explain it? https://stats.stackexchange.com/a/258945
8. Author, F.: Article title. Journal 2(5), 99--110 (2016)
9. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1--13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
10. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
11. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1--2. Publisher, Location (2010)
12. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017

1 Appendix

1.1 SQL data view

```sql
DECLARE @nowDate datetime
SET @nowDate = getdate()

SELECT
[invoices].[customerId] as customerId
,sum([invoices].[amount]) as amountTotal
,sum(case when invoiceType = 1 then [invoices].[amount] else 0 end) as salesTotal
,sum(case when invoiceType = 2 then [invoices].[amount] else 0 end) as annualTotal
,sum(
    case when invoiceType = 3 AND invoiceProducts = 'Truancy Call'
    then [invoices].[amount] else 0 end
) as monthlyTotalProduct1
,sum(
    case when invoiceType = 3 AND invoiceProducts = 'Call Parents'
    then [invoices].[amount] else 0 end
) as monthlyTotalProduct2
```

```sql
,sum(
    case when invoiceType = 5 then [invoices].[amount] else 0 end
) as extraItemsTotal
,DATEDIFF(
    MONTH, min([invoiceDate]), isnull(cancelledDate, @nowDate)
) as monthsSinceStarted
,avg(daysToPay) as avgDaysToPay -- note if -1 then no payment date recorded yet
,sum([creditedAmount]) as creditedAmount
,isCancelled
,hasProduct1
,hasProduct2
,hasProduct3
,hasProduct4
,hasProduct5
,hasProduct6
,hasProduct7
,hasProduct8
,hasProduct9
,hasProduct10
,hasProduct11
,hasProduct12
,isnull([administrativeArea],'none') as administrativeArea
,isnull([SubCountryArea],'none') as [subCountryArea]
,isnull(invoices.[Country],min(invoices.[country])) as [country]
,min([invoiceDate]) as firstInvoiceDate
FROM
        [invoices] as invoices
inner join
        [OrgHasSaleOverProducts] as sales
ON
        sales.school_Id = invoices.customerId
group by
        [invoices].customerId
        ,isCancelled
        ,hasProduct1
        ,hasProduct2
        ,hasProduct3
        ,hasProduct4
        ,hasProduct5
        ,hasProduct6
        ,hasProduct7
        ,hasProduct8
        ,hasProduct9
        ,hasProduct10
        ,hasProduct11
        ,hasProduct12
        ,[administrativeArea]
        ,[SubCountryArea]
        ,invoices.[Country]
        ,cancelledDate
```

```
order by
        [invoices].customerId
desc
```

1.2 Python initial pipeline view

```
X_train, X_test, y_train, y_test = getTrainTestSplit(dataFrame)
C = uniform(loc=0, scale=4).rvs(10)
refitScore = 'precision_score'
param_grid = [
    {
        'classifier__C': C,
        'classifier__penalty':['l1', 'l2']
    }
]

transformPipeline = Pipeline([
    ('Initial drop cols',
    DFTransform(
        lambda X: X.drop([customerIdKey,'firstInvoiceDate'], axis=1)
        )
    ),
    ('Remove catagorical',
    DFTransform(
        lambda X: X.select_dtypes(exclude=['object'])
        )
    ),
])
clfPipeline = Pipeline([
    ('scaler',StandardScaler()),
    ('classifier', LogisticRegression())
])

transformedDfX_test = transformPipeline.transform(X_train)
bestModel = GridSearchCVOnPipeline(
    transformedDfX_test, y_train, clfPipeline, param_grid, refitScore
    )

transformedDfX_test = transformPipeline.transform(X_test)
BestModelScore(transformedDfX_test, y_test, bestModel, refitScore)
```