Determining Online Shopper's Intention to Purchase Using Website Analysis Data and Session Information

Shubhdeep Singh[1], Shaunak Phaldessai[1]

School of Science, Computer Science and Information Technology,
RMIT University, Australia

{s3764546, s3767517}@student.rmit.edu.au

TABLE OF CONTENTS

ABSTRACT/INTRODUCTION 2
ABSTRACT 2
INTRODUCTION 2
METHODOLOGY 2
DATA PREPARATION 2
DATA EXPLORATION 3
SAMPLING TECHNIQUE 5
MODEL SELECTION 6
K-NEAREST NEIGHBOR CLASSIFICATION 6
Hill climbing feature selection *6*
K –Nearest Neighbor classifier *6*
Parameter Tuning *6*
DECISION TREE CLASSIFICATION 7
RESULTS 8
DISCUSSIONS 10
CONCLUSION/REFERENCES 10
CONCLUSION 10
REFERENCES 10

Abstract. We address the problem of identifying user behaviourto buy or abandon an eCommerce website based on the available ClickStream and Web-Analytics data. Classification techniques are used to predict the intention of user visiting a website to buy a product leading to revenue generationor to abandon the website. Prediction accuracy was compared by selecting specific features from the oversampled data based on two different classification techniques. The proposed method was able to achieve a classification error rate of 0.092.

Keywords: SMOTE, K-Neighbours Classification, Decision Tree Classifier, Hill Climbing

1 Introduction

Online visibility and access to users across varied geographical locations influences online businesses the most. The ability to know the traffic behaviour and user response has allowed online websites to optimize their content and in turn generate more revenue. It is, therefore, important to know whether a user visiting an e-commerce website will buy or abandon the website.The Online Shopper's Intention dataset from UCI's machine learning repository was used. The data includes both numerical and categorical values. The class label 'Revenue' is used to classify the data. The 'Administrative', 'Informational' and 'ProductRelated' attributes represent the number of respective pages visited in a session. The 'Administrative_Duration', 'Information_Duration' and the 'ProductRelated_Duration' attributes represent the total amount of time spent by the visitor on these respective pages. The 'Bounce Rate' value is the average bounce rate value of the pages visited by the visitor. The 'Exit Rate' value is the average exit value of the pages visited by the visitor. The 'PageValues' attribute is a web-analytics attribute that represents the number of pages visited by a visitor before arriving at the page. The 'BounceRate', 'ExitRate' and 'PageValues' attribute is available in web-analytics is available in the dataset. Based on the data we try to determine the visitor likelihood of buying or abandoning the website.

How to handle the imbalanced data?

The Online Shopper's intention dataset included the imbalanced data with one out of two class labels highly under-represented. For better prediction with classification techniques, oversampling of data was done using SMOTE: "Synthetic Minority Over-sampling Technique"

2 Methodology

2.1 Data Preparation

The data was taken from UCI machine learning repository. Using the simple "df.isna().sum()" was used to identify any null values present in the data. Fortunately enough, the data was clean without any null values. Descriptive statistics provided insights for

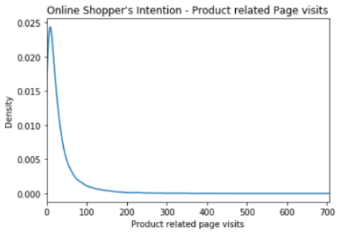
outliers. Since all the potential outliers lie inside the range of values specified by the data description page, we chose not to remove them. Oversampling was used to supplement the under-represented class. We also used factorization techniques to change categorical values into numerical for feeding to the classifiers.

2.2 Data Exploration

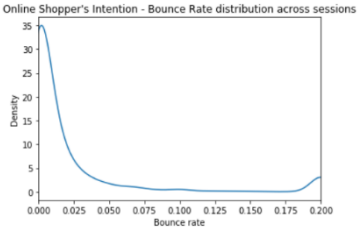
Administrative pages of a website are the account management pages. The graph depicts the popularity of 'Administrative'pages amongst various sessions. It can be noticed that it is highly unlikely for a session to visit a large number of account related pages. A density plot has been used to represent the decreasing trend in the number of Administrative page visits.

'Informational'attribute represents the number of pages visited by the visitor about the website, communication and address and other information on the website. The density plot represents the trend in the form of a ripple depicting how number of page visits fluctuate for small number of page visits and is right skewed representing a decrease in the density as number of informational page visits increase.

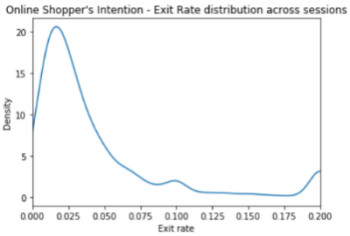
The 'ProductRelated' column represents the number of Product related page visits during a session. The density plot was appropriate to use in this case to present the proportion of online shopper's visiting product related pages. It is evident from the graph that the amount of small number of product related page visits is more as compared to larger number of page visits.



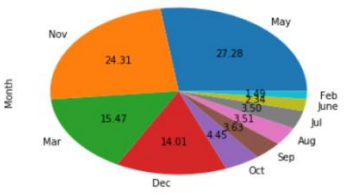
A bounce is defined as a single page session on a website. As per google analytics, the 'Bounce Rate' is the percentage of sessions on the website where the users viewed only one page. The density plot is used to represent the distribution of bounce rate across sessions. As depicted by the graph there is a positive trend towards the two ends



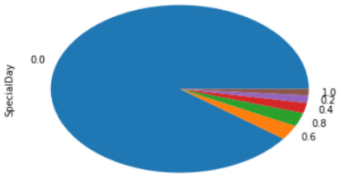
The 'Exit rate' of a webpage is the percentage of times the page was last in the session for all the pageviews to the page. As per the density plot, it is significant that lower exit rates are quite popular and the plot is right skewed with an uptick at the end representing that some good amount of sessions have a larger exit rate as compared to others.



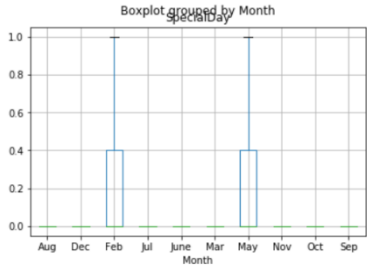
The 'Month' attribute represent the popularity of website based on the month of the year. A pie chart has been used to present the data well.



The ‘Special day’ attribute represents the closeness of site visit time to a special day. Since the sessions are more likely to finalize into a transaction when there is a special day incoming, it is vital to plot the amount of sessions around special days according to how close they are to a special day. A value of ‘0.0’ represents that the session was either before the one week prior to a special day or after it.

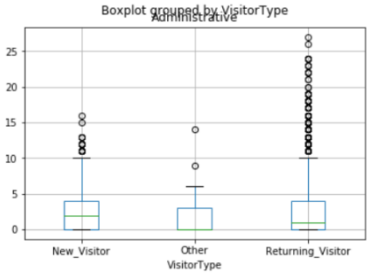


Hypothesis1.The number of sessions immediately before a special day should belong to particular months of the year. The ‘SpecialDay’ feature indicates the closeness of the session visit to a special day. Since there are certain months having a higher number of special day as compared to others with either less or no special day, it was interesting to explore the relationship. A boxplot provides better summary statistics and hence was ideal to represent the distribution of values. The hypothesis was correct since sessions before a special day only belong to the month of February and May.



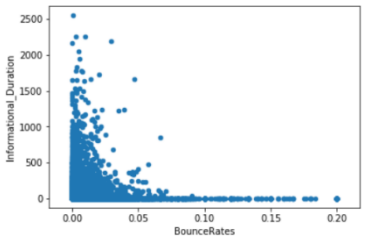
Hypothesis2. Returning visitors should have a higher number of account related page visits.

The ‘Administrative’ feature is the number of page visits on the account related pages during a particular session. A boxplot is used to represent the descriptive statistics better and compare the distribution of Administrative page visits based on the ‘VisitorType’. The hypothesis is correct, given that Returning Visitor spends a higher amount of time on account related pages.



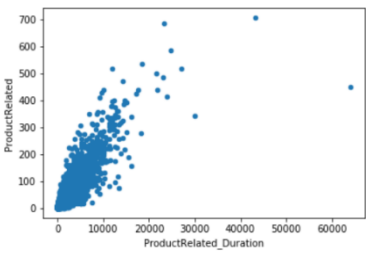
Hypothesis3. The bounce rate is inversely proportional to the Informational page view durations.

The ‘Informational_Duration’ feature represents the total amount of time (in seconds) spend by a visitor on Informational pages. Since a better informational page should increase the page visit time, it should be a factor influencing the bounce rates inversely. A scatter plot is used to depict the relation and it correctly identifies the bounce rate decreasing with a decrease in the duration of visits on an informational page.



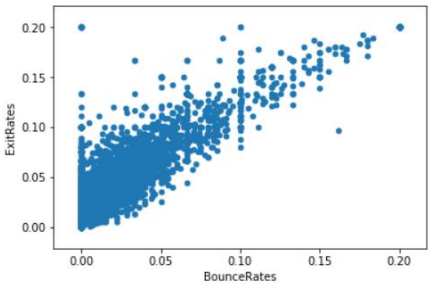
Hypothesis4. If the duration of a product related page increases, the number of page visits on product related page should also increase.

Since every time someone visits a product related page for a certain time period, the underlying number of product related page visits should also increase. The scatter plot was used to depict the relationship between these attributes. The



hypothesis is correct as represented by the plot.

Hypothesis5. The bounce rate values should tend to increase with an increase in the exit rate values.



The 'BounceRate' and 'ExitRate' attribute according to the website analytics definition is 'the percentage of visitors who enter the site from that page and then leave the website without triggering any other requests to the analytics server during that session' and 'the percentage of pageviews that were last in the session for all the pageviews of the page' respectively. Since both the features represent the probability of a user leaving the website, it seems relevant to study the relationship between both of these features. A scatter plot is used to represent the relationship well.

2.3 Sampling Technique

Through Exploratory data analysis, we identified an imbalance in the class distribution. The data for sessions leading to a transaction is highly misrepresented. Out of the 12330 unique sessions over a year, only 1908 led to generating revenue/conversion. Since, Machine learning algorithms, in general make better predictions after being trained with balanced data and in an effort to not lose the valuable information, we chose to oversample the data.

```
df['Revenue'].value_counts()
```

```
False    10422
True      1908
Name: Revenue, dtype: int64
```

SMOTE: "Synthetic Minority Over-sampling Technique"

The Random Oversampling, although robust, runs the risk of providing highly biased results. The sheer amount of duplicated under-represented values can lead to overfitting of data. As such, SMOTE was used to oversample the data. SMOTE was preferred over 'Random oversampling' to supplement the data. SMOTE uses K nearest neighbours of a random sample followed by taking a vector between one of the K-neighbours and the current data point. Multiplying the vector with a random number between 0 and 1 and adding to the current data point generates a new data point which is added to the new data set. Instead of operating on data, SMOTE operates on features to create larger datasets with less specific outcomes as compared to Random Sampling. SMOTE is part of the imbalanced-learn package and the following code was used to import SMOTE into IPython notebook.

```
#Installing imbalanced-learn package to perform oversampling using SMOTE
import sys
!{sys.executable} -m pip install imbalanced-learn

#Importing SMOTE from imblearn
from imblearn.over_sampling import SMOTE
```

Oversampling with SMOTE

A new dataset was generated with equal number of records for both the classes. The misrepresented class label wassupplemented with additional observations as shown below.

```
#OverSampling script using SMOTE
print("Before OverSampling, counts of label 'True': {}".format(sum(df_target==True)))
print("Before OverSampling, counts of label 'False': {} \n".format(sum(df_target==False)))

sm = SMOTE(random_state=2)
df_data_over, df_target_over = sm.fit_sample(df_data, df_target.ravel())

print('After OverSampling, the shape of train_X: {}'.format(df_data_over.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(df_target_over.shape))

print("After OverSampling, counts of label 'True': {}".format(sum(df_target_over==True)))
print("After OverSampling, counts of label 'False': {}".format(sum(df_target_over==False)))
```

2.4 Model Selection

Classification is a type of supervised learning algorithm used to identify the category to which a new observation belongs. In the Online shopper's Intention dataset, class labels were provided for 'Revenue' column to recognize whether a session will lead to a conversion or not. As such, we used "K-Nearest Neighbours classification" and "Decision tree" classification to model the data.

2.5 K-Nearest Neighbours Classification

Hill-Climbing feature selection

We apply the hill-climbing feature selection technique to determine whether some features are better at predicting the intention of a shopper to buy or abandon the website. The Online Shopper's Intention data includes both session related features and the web analysis data based on page views. The dataset includes features such as 'Bounce Rates', 'Exit Rates', and 'PageValues' which are used intensively in online website analytics. The 'SpecialDay' feature representing the closeness of session date to a special day is also highly likely to determine whether a session will lead to a person buying from the website.

The results from hill-climbing constitute some of the features repeating and hence signifying that some features are better at classifying the results into classes. The 'ProductRelated', 'PageValues' and 'SpecialDay' features were chosen for further analysis with K-nearest neighbour classifier.

K-nearest neighbour classifier

We apply K-Nearest neighbour classifier to the dataset having the selected features. The selected features included the numerical features 'ProductRelated', 'PageValues', 'SpecialDay'. The dataset was oversampled to generate equal number of records belonging to both classes. The results were better as compared to those predicted by taking into account all the numerical features.

Parameter Tuning

While we keep the k-neighbours=5, we tried and changed the weights parameter of the k-neighbours classifier, weights = 'distance'. The results for all three splits improved.

The 60-40 split gave the best result of all and we went further by tuning the classifier more using the p-value.

P-value – Hill-climbing was used to select specific most influential features and therefore, we applied further tuning on the 60-40 split classifier. Assuming that already selected features will have equal contribution in predicting the class, we tried p=1.

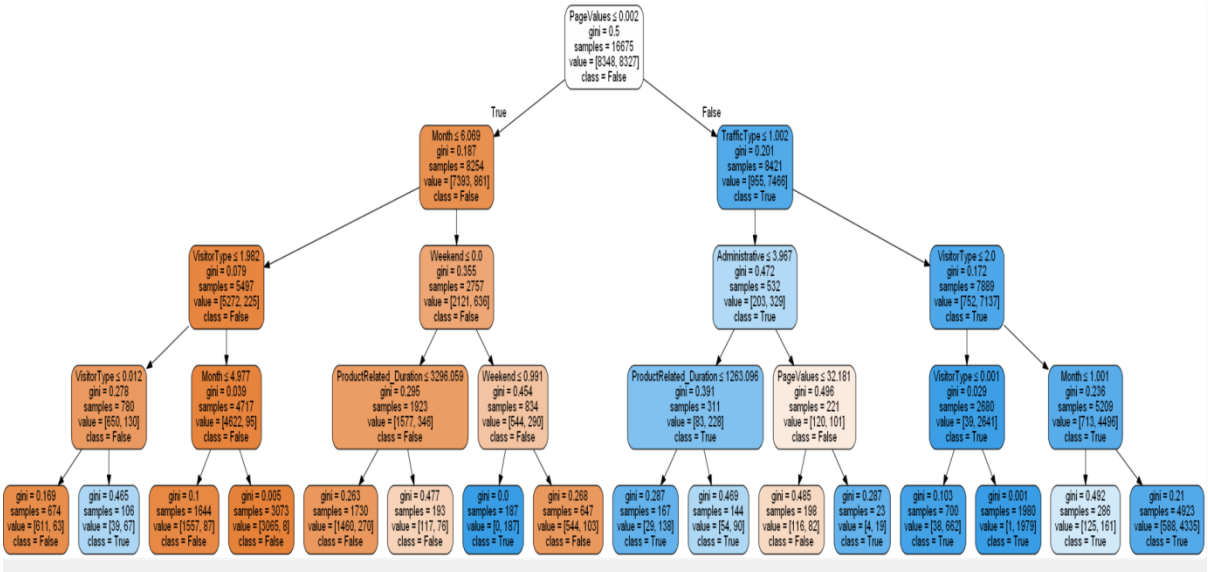
We also tried the p=2 but the results were better with p=1 and since the features have already been selected using hill-climbing technique, it is presumed that each feature contributes equally to predicting the visitor intention to buy or abandon the website.

```
#k-neighbor after smote 60-40 with tune with selected features
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_hill_data_over, df_hill_target_over, test_size=0.40, random_state=4)
print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(5, weights='distance', p=1)
fit = clf.fit(X_train, y_train)
y_pre=fit.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pre)
from sklearn.metrics import classification_report
print classification_report(y_test, y_pre)
print cm
```

2.6 Decision-Tree Classifier

Decision tree classification technique was used to include both numerical and categorical features in predicting the user intention to buy/convert. The categorical columns ‘Month’ and ‘VisitorType’ is factorized to numerical values to feed to the decision tree classifier. The whole dataset was oversampledusing SMOTE and train-test split was performed over the oversampled data. The decision tree classifier was trained with a max-depth of 3.

The decision tree classifier was modelled for 50-50, 60-40 and 80-20 split. The results were significantly better as compared to k-neighbours classifier. The 60-40 split gave the best f1-score and the precision for the oversampled data including all the features. We tried to tune the decision tree better by using max-depth = 4. As a result we achieved a classification error rate of 0.092 i.e an almost 91% accuracy in predicting the visitor’s intention to convert or abandon the website.



3 Results

The k-neighbour classifier was modelled using both the original data as well as the oversampled data from SMOTE. The table below shows the data fed to the classifier with the corresponding split and the observed classification error rates using the k-neighbours classifier. The best classification error rate we were able to achieve with the k-neighbours classification technique after tuning the parameters was 0.133.

Online Shopper's Intention Data		
K-Neighbour Classifier		
Classification Data	Split	Classification Error Rate
Original Data With all the Features	50-50	0.14
	60-40	0.13
	80-20	0.14
Oversampled Data with all the Features	50-50	0.17
	60-40	0.16
	80-20	0.14
Oversampled Data with Selected Features	50-50	0.16
	60-40	0.139
	80-20	0.153
Oversampled Data with Selected Features and weights=distance	50-50	0.157
	60-40	0.134
	80-20	0.148

Online Shopper's Intention Data		
K-Neighbour Classifier		
Classification Data	p-value	Classification Error Rate
Oversampled Data with Selected Features and weights=distance and 60-40 split	p=1	0.133
	p=2	0.134

Following is the confusion matrix for the K-neighbours classifier with oversampled data after tuning the parameters. The Classification error rate for the completely tuned K-neighbours classifier is 0.13.

```
[[3835  418]
 [ 695 3390]]
```

Confusion Matrix for K-neighbours classifier 60-40 split

The Decision tree results were better as compared to the K-neighbours classifier. Following is the confusion matrix for decision tree classifier with a 60-40 split on the oversampled data and a max-depth=4. The classification error rate for Decision tree comes out to be 0.092.

```
array([[1843,  231],
       [ 155, 1940]], dtype=int64)
```

Confusion matrix for Decision tree

Classification Report:

Following are the classification report output from both the k-neighbours classifier and the decision tree classifier.
As shown, both the precision and the f1-score in case of a decision tree is better.

k-neighbour classifier:

```
#k-neighbor after smote 60-40 with tune with selected features
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_hill_data_over, df_hill_target_over, test_size=0.40, random_state=4)
print("Number transactions X_train dataset: ", X_train.shape)
print("Number transactions y_train dataset: ", y_train.shape)
print("Number transactions X_test dataset: ", X_test.shape)
print("Number transactions y_test dataset: ", y_test.shape)
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(5, weights='distance', p=1)
fit = clf.fit(X_train, y_train)
y_pre=fit.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pre)
from sklearn.metrics import classification_report
print classification_report(y_test, y_pre)
print cm
```

```
('Number transactions X_train dataset: ', (12506L, 3L))
('Number transactions y_train dataset: ', (12506L,))
('Number transactions X_test dataset: ', (8338L, 3L))
('Number transactions y_test dataset: ', (8338L,))
             precision    recall  f1-score   support

       False       0.85      0.90      0.87      4253
        True       0.89      0.83      0.86      4085

   micro avg       0.87      0.87      0.87      8338
   macro avg       0.87      0.87      0.87      8338
weighted avg       0.87      0.87      0.87      8338

[[3835  418]
 [ 695 3390]]
```

Decision tree classifier:

```
#Decision tree with oversampled data of all the features and 80-20 split with max-depth=4
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df_data_tree_over, df_target_tree_over, test_size=0.40, random_state=0)
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(max_depth = 4)
fit = clf.fit(X_train, y_train)
y_pre = fit.predict(X_test)
y_pre
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pre)
print cm
from sklearn.metrics import classification_report
print classification_report(y_test,y_pre)
```

```
[[1843  231]
 [ 155 1940]]
             precision    recall  f1-score   support

       False       0.92      0.89      0.91      2074
        True       0.89      0.93      0.91      2095

   micro avg       0.91      0.91      0.91      4169
   macro avg       0.91      0.91      0.91      4169
weighted avg       0.91      0.91      0.91      4169
```

4 Discussion

The data modelling for Online Shopper's intention data was done using two different classification techniques namely, 'K-Neighbour's classifier' and 'Decision –tree classifier'. The k-neighbours classifier was trained using the numerical values available in the dataset. Using all the numerical features, we oversampled the data to adjust the class distribution of data. Feature selection was done using Hill-climbing technique. Three most prominent features were selected out of all the numerical features and were used for K-neighbours classification using 50-50, 60-40 and 80-20 split. With the selected features we were able to obtain a better classification error rate of 0.139 with a 60-40 split of oversampled data. We further investigate and try to tune the parameter for better prediction. With the weights parameter and p=1, we achieved a better classification error rate of 0.133.

On the other hand, with the decision tree model having a max-depth parameter set to 3, we got the best classification error rate of 0.1041. To tune the decision tree model we tried the 'max-depth'=4 and got a classification error rate of 0.092 with a 60-40 split.

The f1-score and precision also improved for the under-represented data in case of a decision tree. Since in case of a decision tree all the attributes were used in predicting the visitor's intent, and we got a better precision and f1-score, it is evident that a decision tree is a better classifical model technique for online shopper's intention dataset.

5 Conclusion/ References

5.1Conclusion

The decision tree classifier outperform the k-neighbours classifier in predicting the visitor's intention to buy. The tuned K-neighbours classifier with the categorical features alone is not advisable to be used with the 'online-Shopper's Intent' dataset. A decision-tree with its ability to work well with both numerical and categorical data provides better predictions with 90.8% accuracy.

5.2 References

1. http://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset
2. https://link.springer.com/article/10.1007%2Fs00521-018-3523-0
3. https://support.google.com/analytics/answer/2695658?hl=en
4. https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis
5. https://www3.nd.edu/~dial/publications/chawla2005data.pdf