MAT394

Introduction to Machine Learning using R

Project Report

Group 6:
Isha Shrivastava
1710110149
is490

Customer purchase prediction: Using Machine learning to predict if a customer will complete their purchase on a website or not?

I) Abstract

In this age of E-Commerce, online shopping is a rather prevalent and common activity people indulge in. With the onset of COVID-19, online shopping rates shot up and up to 68% Indians increased their online shopping activity. However, when users shop online, it is not always that they end up purchasing the item in their cart. I, myself, have done this on countless occasions where I often add products in my shopping cart only to not purchase them later on. Considering this, it might be useful for companies and websites to know whether a particular customer is willing to finish their purchase or not and accordingly provide incentives which leads to a positive outcome (*Narayan*). In this project, we try to predict a customer's behaviour. We employ the use of classification algorithms to construct such a model. Additionally, in this project, we will apply machine learning algorithms such as logistic regression, Naive Bayes, Random Forests and KNN to predict whether a customer will finish their purchase. The dataset used for the project is the Online Shoppers Purchasing Intention dataset taken from UC Irvine's Machine Learning Repository (*UCI Machine Learning Repository*).

II) Introduction

A. Dataset Description:

The dataset consists of feature vectors belonging to 12,330 sessions. The dataset was formed so that each session would belong to a different user in a 1-year period to avoid any tendency to a specific campaign, special day, user profile, or period. Of the 12,330 sessions in the dataset, 84.5% (10,422) were negative class samples that did not end with shopping, and the rest (1908) were positive class samples ending with shopping.

Data Set Characteristics:	Multivariate	**Number of Instances:**	12330	**Area:**	Business
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	18	**Date Donated**	2018-08-31
Associated Tasks:	Classification, Clustering	**Missing Values?**	N/A	**Number of Web Hits:**	129521

Contains data

obs:	12,330
vars:	18
size:	1,072,710

variable name	storage type	display format	value label	variable label
Administrative	byte	%10.0g		**Administrative**
Administrativ~n	double	%10.0g		**Administrative_Duration**
Informational	byte	%10.0g		**Informational**
Informational~n	double	%10.0g		**Informational_Duration**
ProductRelated	int	%10.0g		**ProductRelated**
ProductRelate~n	double	%10.0g		**ProductRelated_Duration**
BounceRates	double	%10.0g		**BounceRates**
ExitRates	double	%10.0g		**ExitRates**
PageValues	double	%10.0g		**PageValues**
SpecialDay	double	%10.0g		**SpecialDay**
Month	str4	%9s		**Month**
OperatingSyst~s	byte	%10.0g		**OperatingSystems**
Browser	byte	%10.0g		**Browser**
Region	byte	%10.0g		**Region**
TrafficType	byte	%10.0g		**TrafficType**
VisitorType	str17	%17s		**VisitorType**
Weekend	byte	%1.0f		**Weekend**
Revenue	byte	%1.0f		**Revenue**

Variable	Obs	Mean	Std. Dev.	Min	Max
Administra~e	12,330	2.315166	3.321784	0	27
Administra~n	12,330	80.81861	176.7791	0	3398.75
Informatio~l	12,330	.5035685	1.270156	0	24
Informatio~n	12,330	34.4724	140.7493	0	2549.375
ProductRel~d	12,330	31.73147	44.4755	0	705
ProductRel~n	12,330	1194.746	1913.669	0	63973.52
BounceRates	12,330	.0221914	.0484883	0	.2
ExitRates	12,330	.0430728	.0485965	0	.2
PageValues	12,330	5.889258	18.56844	0	361.7637
SpecialDay	12,330	.0614274	.1989173	0	1
Month	0				
OperatingS~s	12,330	2.124006	.9113248	1	8
Browser	12,330	2.357097	1.717277	1	13
Region	12,330	3.147364	2.401591	1	9
TrafficType	12,330	4.069586	4.025169	1	20
VisitorType	0				
Weekend	12,330	.2326034	.4225086	0	1
Revenue	12,330	.1547445	.3616756	0	1

B. Mathematics behind the models and parameters

1. Random Forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.
Random forests provide an improvement over bagged trees by way of a random forest small tweak that decorrelates the trees. In bagging, we build a number of decision trees on bootstrapped training samples (*Bagging and Random Forest for Imbalanced Classification*). But when building these decision trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors. A fresh sample of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$ — that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors. (Tibshirani et al.)

$$h(x) = \frac{1}{B}\sum_{j=1}^{B} h_j(x)$$

2. Logistic Regression

Consider, a data set Default, where the response default falls into
one of two categories, Yes or No. Rather than modeling this response Y directly, logistic regression models the probability that Y belongs to a particular category.
For the Default data, logistic regression models the probability of default. For example, the probability of default given balance can be written as

Pr(default = Yes|balance)

The values of Pr(default = Yes|balance), which we abbreviate p(balance), will range between 0 and 1. Then for any given value of balance, a prediction can be made for default. For example, one might predict default = Yes for any individual for whom p(balance) > 0.5. Alternatively, if a company wishes to be conservative in predicting individuals who are at risk for default, then they may choose to use a lower threshold, such as p(balance) > 0.1.

To model the relationship between p(X) = Pr(Y = 1|X) and X we use linear regression model to represent the probabilities:

$$p(X) = \beta_0 + \beta_1 X$$

In logistic regression, we use also the logistic function,

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

which upon further modifications, changes to:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

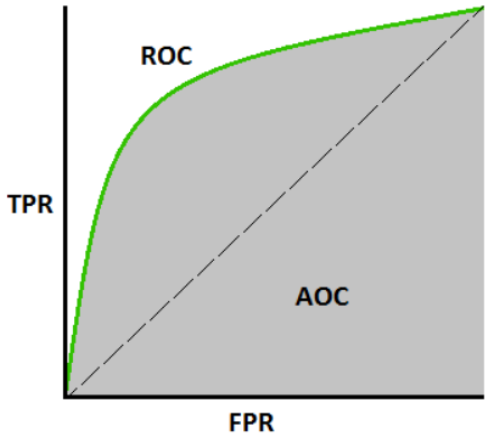
The left-hand side is called the log-odds or logit. The logit of a logistic regression model is linear in *X*. By this, we mean that in a logistic regression model, increasing X by one unit changes the log odds by β1.

In this project, we use the two ML algorithms: Random Forests and Logistic Regression. Firstly, we apply, a complex decision tree-based algorithm, random forests to the data set and evaluate its performance. We then check the accuracy of the results compared to a simple method like logistic regression.

The performance metrics used in this work to compare the algorithms are accuracy and ROC curve.

It is one of the most important evaluation metrics for checking any classification model's performance. It is also written as AUROC (**Area Under the Receiver Operating Characteristics**)

ROC is a probability curve and AUC represent the degree or measure of separability. It tells how much the model is capable of distinguishing between classes (*Understanding AUC - ROC Curve*). Higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s. By analogy, the Higher the AUC, the better the model is at predicting if a customer will buy a product or not. The figure depicting the ROC Curve with True positive rate and false positive rate as y and x axes respectively are shown below.



AUC - ROC Curve [Image 2] (Image courtesy: My Photoshopped Collection)

III) Methodology and Analysis:

- The libraries needed for this project are imported.

```
import pandas as pd #importing pandas for dataframe
import matplotlib.pyplot as plt #plotting library
import seaborn as sns #for pandas dataframes representation
import numpy as np #to perform array operations
from sklearn.model_selection import train_test_split #splitting of training and testing data
from sklearn.metrics import classification_report #to give us accuracy
from sklearn.ensemble import RandomForestClassifier #random forest classifier
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import SMOTE #to oversample the imbalanced data
from sklearn.metrics import plot_roc_curve
%matplotlib inline
```

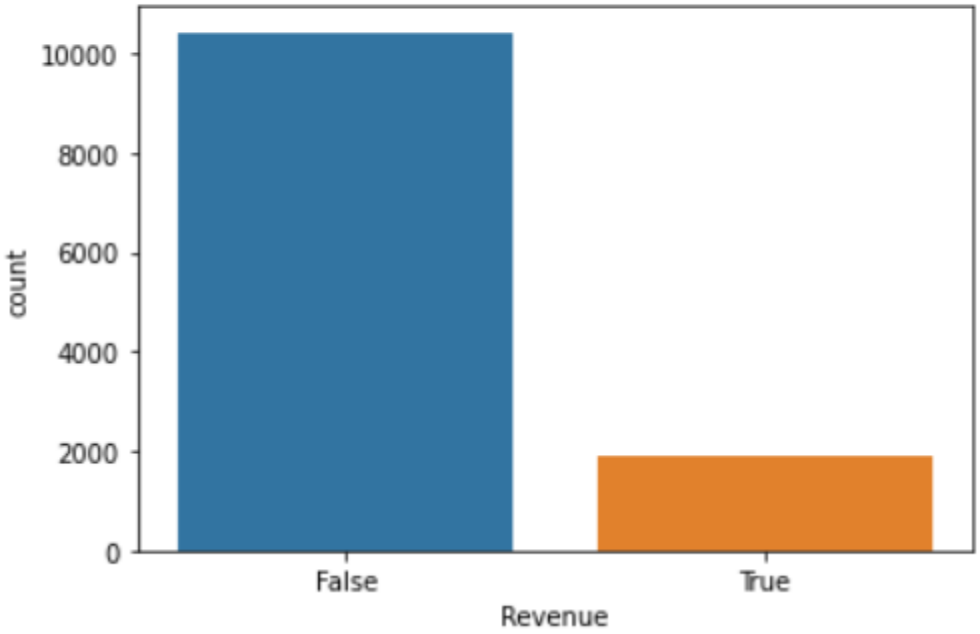
- The data is visualized and shown as a pandas data frame with several attributes.

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageVa
0	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.00(
1	0	0.0	0	0.0	2	64.000000	0.000000	0.100000	0.00(
2	0	0.0	0	0.0	1	0.000000	0.200000	0.200000	0.00(
3	0	0.0	0	0.0	2	2.666667	0.050000	0.140000	0.00(
4	0	0.0	0	0.0	10	627.500000	0.020000	0.050000	0.00(
...	...	...	...	...	...	...	...	...	
12325	3	145.0	0	0.0	53	1783.791667	0.007143	0.029031	12.24
12326	0	0.0	0	0.0	5	465.750000	0.000000	0.021333	0.00(
12327	0	0.0	0	0.0	6	184.250000	0.083333	0.086667	0.00(
12328	4	75.0	0	0.0	15	346.000000	0.000000	0.021053	0.00(
12329	0	0.0	0	0.0	3	21.250000	0.000000	0.066667	0.00(

12330 rows × 18 columns

The label which will aid us in classifying whether the customer will purchase or not is given under the attribute of revenue. The revenue column is visualised below.

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



We can see the graph between false and true data and their counts.

- Data analysis and feature extraction is performed on revenue as to predict if a customer would buy an item or not. The labels mentioned under the revenue attribute are of type string. False or true are mentioned and we cast them into integers to extract features. The graph above also shows the casted form of revenue and it shows 64 bit int.

```
df = pd.get_dummies( df, columns = ['VisitorType','Month']) #converting categorical variable into dummy
df['Weekend'] = df['Weekend'].astype(int) #casting into integer data type
df['Revenue'] = df['Revenue'].astype(int) #casting into integer data type
```

- Oversampling and test train split:

  10422 false and 1908 true values are split into test and train data with 80% of the data being reserved for training and 20% of the data for testing. But now the question arises why do we oversample?

  With an imbalanced dataset, it becomes very difficult to accurately predict the minority class by coming up with a classification. One way to solve this problem is to oversample the examples in the minority class. This can be achieved by simply duplicating examples from the minority class in the training dataset prior to fitting a model. The oversampling

done in this project is through SMOTE (synthetic minority oversampling technique). The library imblearn is imported for this purpose.

Further with algorithms like random forest, where multiple decision trees are combined, using severely imbalanced data would result in a poor performance in predicting minority class. Thus, we use oversampling before training our dataset.

```
X = df.drop('Revenue', axis=1) #input features
Y = df['Revenue'] #target value
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_state=0)
```

Oversampling to counter class imbalance (Using SMOTE)

```
smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(x_train, y_train)
```

- Training of the dataset:
  The training of the dataset for two algorithms considered in this work ie. Random Forests and Logistic Regression are shown.

```
clf_s = RandomForestClassifier(n_estimators=100) #training using random forest classifier
clf_s.fit(X_res, y_res)
#clf_s = RandomForestClassifier(n_estimators=100)
#clf_s.fit(x_train, y_train)
```

Figure 1 Training for Random Forests Algorithm

```
clf_s1 = LogisticRegression() #modelling for logistic regression
clf_s1.fit(X_res, y_res)
```

Figure 2 Training for Logistic Regression Algorithm

IV) Results:

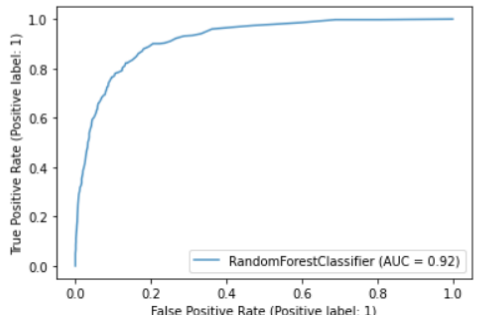
The results and accuracy for both the models are shown below:

```
y_pred = clf_s.predict(x_test) #predicting the testing data
print(classification_report(y_test, y_pred)) #to give us the accuracy between predicted data and test data labels
print("Accuracy :", clf_s.score(X_res, y_res))
```

```
              precision    recall  f1-score   support

           0       0.93      0.94      0.93      2044
           1       0.69      0.66      0.67       422

    accuracy                           0.89      2466
   macro avg       0.81      0.80      0.80      2466
weighted avg       0.89      0.89      0.89      2466

Accuracy : 0.8901054339010543
```

The ROC curve indicating the area under the curve is also shown below for random forests.

```
ax = plt.gca()
rfc_disp = plot_roc_curve(clf_s, X_res, y_res, ax=ax, alpha=0.8) #plotting roc curve for alpha=0.8
plt.show()
```

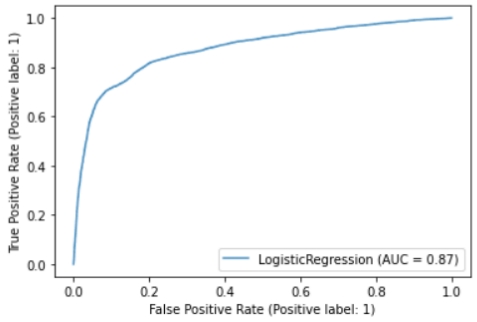


The ROC curve for logistic regression,

```
y_pred = clf_s1.predict(x_test) #predicting the testing data
print(classification_report(y_test, y_pred)) #to give us the accuracy between predicted data and test data labels
print("Accuracy :", clf_s1.score(x_test, y_test))
```

```
              precision    recall  f1-score   support

           0       0.88      0.97      0.92      2044
           1       0.74      0.36      0.48       422

    accuracy                           0.87      2466
   macro avg       0.81      0.66      0.70      2466
weighted avg       0.86      0.87      0.85      2466

Accuracy : 0.8682076236820763
```

```
ax = plt.gca()
rfc_disp = plot_roc_curve(clf_s1, X_res, y_res, ax=ax, alpha=0.8)
plt.show()
```



V) Conclusion:

Clearly, on comparing random forests and logistic regression, we observe that the random forests algorithm has a higher accuracy of 89.01% than 86.8% of logistic regression. The ROC curves also show that random forest algorithm does better than the logistic regression with an Area Under the Curve (AUC) value of 0.92 and 0.87 respectively. Thus, it is seen that for customer purchase prediction, the better algorithm is random forests and we found that the Random Forests algorithm performed with the best accuracy on this dataset.

VI) References:

Bagging and Random Forest for Imbalanced Classification. https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/. Accessed 25 Apr. 2021.

Predict If a Customer Will Purchase on a Website or Not? | by Jay Narayan | Analytics Vidhya | Medium. https://medium.com/analytics-vidhya/predict-if-a-customer-will-purchase-on-a-website-or-not-b785e7f5697f. Accessed 25 Apr. 2021.

Tibshirani, Robert, et al. "An Introduction to Statistical Learning with Applications in R." *European Review for Medical and Pharmacological Sciences*, vol. 16, no. SUPPL. 1, Springer, 2013.

UCI Machine Learning Repository: Online Shoppers Purchasing Intention Dataset Data Set. https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset#. Accessed 25 Apr. 2021.

Understanding AUC - ROC Curve | by Sarang Narkhede | Towards Data Science. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5. Accessed 25 Apr. 2021.

GitHub Repository: https://github.com/is490/Customer-purchase-prediction