PROJECT REPORT

E-COMMERCE PURCHASE PREDICTION

Submitted towards the partial fulfilment of the criteria for award of Post Graduate Program in Data Analytics by Imarticus

Submitted By:

Pooja K R(IL016761)

Course and Batch: PGA-20 Mar 2021

Table of Contents

Abstract

The massive adoption of the Web an e-commerce platform has led to a fundamental change in the way that businesses of all sizes interact with their customers. Customers today make a purchase from anywhere, anytime and from any device just in one-click. By using the visitor details we explore about the customer behaviour and customer segmentation and Using this it is easy for the recommendations for the future analysis. This approach drives more repeat customers to the website. The main application of machine learning helps web shoppers make better shopping decisions by identifying the efficient customers to the website.

Acknowledgements

We are using this opportunity to express our gratitude to everyone who supported us throughout the course of this group project. We are thankful for their aspiring guidance, invaluably constructive and friendly advice during the project work. We are sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

Further, we were fortunate to have great teachers who readily shared their immense knowledge in data analytics and guided us in a manner that the outcome resulted in enhancing our data skills.

We wish to thank, all the faculties, as this project utilized knowledge gained from every course that formed the PGA program.

We certify that the work done by me for conceptualizing and completing this project is original and authentic.

Date: Mar 20,2021 Pooja K R

Place: Chennai

Certificate of Completion

I hereby certify that the project titled “**E-COMMERCE PURCHASE PREDICTION”** was undertaken and completed under supervision by Pooja from the batch of PGA-15(Mar 2021)

Date: Mar 20,2021

Place: Chennai

CHAPTER 1: INTRODUCTION

1.1Title & Objective of the study

‘E-commerce purchase prediction’ is the project that I am working upon which falls under (E-commerce Sector). The data has been collected from a real-world ecommerce website. It is raw data, i.e., without any content transformations, however, all values are hashed due to confidential issues. The primary purpose of working in this project is to Predict the customer’s purchase by using the past data so, given a set of new predictor variables, we need to predict the target variable as “1”- They will purchase in the site or “0” -They will not make a purchase.

1.2 Need of the Study

In this project, the main purpose is to predict whether the customer will make a purchase or not by using the item and customer identities.so it will be useful to recommend the pr006Fducts to those valid customers. We can give more importance to the customers who purchase regularly and provide offers, discounts to those customers to revisit the site again. This will be useful for recommending relevant products on the purchase to those customers, helps web shoppers make better shopping decisions.

1.3 Business or Enterprise under study

E-commerce purchase prediction contains visitor’s information with the timestamp of the event occurred and many more information in it. Process of analysing ecommerce data include very important part of data cleaning. Researchers noticed that in some cases browsing data include up to 40% of abnormal traffic.

1.4 Business Model of Enterprise

We will also see the valuable insights about the customers behaviour and do customer segmentation and predict the probability of the customer who will make a purchase in future by performing different types of algorithm on the data

1.5 Data Sources

E-commerce purchase prediction-Data contains timestamp,itemid, visitorid,event,transaction details

Data Set Description:

After merging four csv excel files, Final data Contains 39539 rows and 5 columns that were collected over a period of 4.5 months. The response variable is ‘purchased’ with ‘0’ for Visitors and ‘1’ for Consumers

Tools & Techniques

Tools: JupyterNotebook.

Techniques: Logistic Regression, Decision Tree Classifier, Random forest Classifier, XGBoost Classifier, Support vector classifier, Naive bayes classifier, K-Nearest Neighbour classifier.

CHAPTER 2: DATA PREPARATION AND UNDERSTANDING

One of the first step we engaged in was to outline the sequence of steps that we will be following for our project. Each of these steps are elaborated below

We have three excel files namely,

*Category tree
*Events
*Item properties part-1
*Item properties part-2

*Category tree→ The excel which explains about the category we have Category id, Parent id. Where Category id represents the unique identifier of the category and parent id represents the identifier of the parent category If it's empty the parent doesn't exist.

*Events→ The excel which has behaviour data of customers. We have timestamp, Visitor id, Event, Itemid, Transaction Id. The timestamp which will tell us when the event is occurred, Visitor id is the unique identifier of the visitor. Event is the type of event which has add to{cart, view, transaction},Item id is the unique identifier of the item.

*Item properties part-1→The excel which item properties. We have timestamp which will tell us when the event is occurred, item id which is the unique Id of the item, property which is the property of the item id all of them which is converted into small parts excluding categoryid and available, value which is the property value of that item

*Item properties part2→The excel which is the continuation of item properties 1 and has the same features and resembles the same details as Item properties

2.1 Phase I –Data Extraction and Cleaning:

We will now get into the project now.

- After importing necessary packages like numpy,pandas,matplotlib,seaborn. Reading the four csv by using pandas.

```
In [21]: 1 Category.head(2)
```

Out[21]:

	categoryid	parentid
0	1016	213.0
1	809	169.0

```
In [18]: 1 Item_properties_1.head(2)
```

Out[18]:

	timestamp	itemid	property	value
0	1435460400000	460429	categoryid	1338
1	1441508400000	206783	888	1116713 960601 n277.200

```
In [19]: 1 Item_properties_2.head(2)
```

Out[19]:

	timestamp	itemid	property	value
0	1433041200000	183478	561	769062
1	1439694000000	132256	976	n26.400 1135780

```
In [20]: 1 #Let's take a peek at the Events dataframe
         2 Events.head(2)
```

Out[20]:

	timestamp	visitorid	event	itemid	transactionid
0	1.433220e+12	257597	view	355908	NaN
1	1.433220e+12	992329	view	248676	NaN

- We will then convert the timestamp to the date time format by using pandas

```
In [22]: 1 Events['timestamp']=pd.to_datetime(Events['timestamp'])
```

```
In [23]: 1 Events
```

Out[23]:

	timestamp	visitorid	event	itemid	transactionid
0	1970-01-01 00:23:53.220	257597	view	355908	NaN
1	1970-01-01 00:23:53.220	992329	view	248676	NaN
2	1970-01-01 00:23:53.220	111016	view	318965	NaN
3	1970-01-01 00:23:53.220	483717	view	253185	NaN
4	1970-01-01 00:23:53.220	951259	view	367447	NaN

- To find the transaction status alone in the events data we will use the below code to extract

```
In [25]:
#Which event has a value in its transaction id
Events[Events.transactionid.notnull()].event.unique()

Out[25]: array(['transaction'], dtype=object)
```

```
#Which event has a value in its transaction id
Events[Events.transactionid.notnull()].event.count()
```

```
8654
```

So the rest of the events NAN transaction ids are either view or add to cart

- We will start separating the customers into two categories, those who purchased something and those who didn't

```
customer_purchased=Events[Events.transactionid.notnull()].visitorid.unique()
```

```
customer_purchased.size
```

```
11719
```

Since we have no information that there were any repeat users who bought something from the site, we will have to assume that 11,719 visitors are unique and made at least a single purchase

- Now we will find the no of visitors in the site

```
In [176]:
#Let's get all unique visitor ids as well
All_customers=Events.visitorid.unique()
All_customers.size

Out[176]: 1407580

In [177]:
Events.shape

Out[177]: (2756101, 5)
```

- Now we will find the no of customers who didn't buy anything with the customer purchased with all the customers in the site

```
In [80]:
customer_browsed=[x for x in All_customers if x not in customer_purchased]

In [81]:
len(customer_browsed)
#Since it is in list..

Out[81]: 1395861
```

So there are 1395861 unique site visitors who didn't buy anything, again assuming that there were no repeat users with different visitor IDs

- Seeing a sample of customer purchased below

```
In [49]: customer_purchased[:10]

Out[49]: array([ 599528,  121688,  552148,  102019,  189384,  350566,  404403,
                 505565,  945184, 1406787], dtype=int64)
```

- We will take a unique id and check what his buying journey is

```
In [118]:
Events[Events.visitorid==483717].sort_values('timestamp')
#So this visitorid_483717 has only viewed it
```

Out[118]:

	timestamp	visitorid	event	itemid	transactionid
8048	1970-01-01 00:23:53.221804507	483717	view	253185	NaN
16809	1970-01-01 00:23:53.221871581	483717	view	353936	NaN
3	1970-01-01 00:23:53.221955914	483717	view	253185	NaN

2.2 Phase II Recommendation

- What insights can we offer the visitor to guide them in their buying journey-Perhaps we can offer the list of what previous visitors bought together with the item they are currently viewing
  - ❖ So first we will take only the purchased items from the customers id and transaction

In [280]:
```
purchased_items = []
# Create another list that contains all their purchases
for i in customer_purchased:
    purchased_items.append(list(Events.loc[(Events.visitorid==i)& (Events.transactionid.notnull())].itemid.values))
```

In [281]:
```
print(purchased_items)
```

```
[[356475], [15335, 380775, 237753, 317178, 12836, 400969, 105792, 25353, 200793, 80582, 302422], [81345], [150318,
49521], [310791, 299044], [54058, 284871, 251130, 268335, 183049, 261940, 369093, 370745, 192990, 277119, 241716, 2
83766, 16417, 217068, 36372, 68923, 428015, 69533, 13520, 385638, 442871, 136526, 247862, 93828, 230911, 382595, 34
853, 216260, 154812, 445241, 57702, 347850, 151855, 226327, 288525, 51354, 345994, 170438, 254301, 266439, 193718,
388558, 26745, 184086, 79956, 252040, 82232, 309821, 394518, 462070, 331980, 353111, 200527, 235933, 68532, 358882,
60012, 29741, 270487, 163689, 6913, 156457, 341578, 163352, 234493, 135174, 452481, 241755, 56323, 210137, 184397,
285202, 198690, 195958, 239210, 71640, 189108, 369112, 346186, 211207, 134330, 257070, 302239, 459480, 57577, 19523
4, 215904, 374092, 170262, 170262, 229577, 264801, 94344, 204209, 442725, 9087, 259964, 361554, 442131, 23251, 3327
21, 85914, 302391, 200242, 234255, 372845, 404129, 214757, 222208, 372845, 206809, 249207], [150100, 50934, 36013,
26210, 118199, 234199, 416187, 167985, 146735, 4887, 218626, 338148, 197980, 240558, 272813, 197968, 332224, 92668,
268335, 120259, 378845, 449946, 95426, 85579, 17478, 16813, 150215, 187200, 111083, 359040, 428067, 179136, 422425,
242521, 21886, 448494, 175555, 421606, 86894, 243376, 138434, 422481, 235748, 63859, 55706], [243566], [245400], [3
36832], [202052, 336832, 259078, 111306, 44882], [7943], [185598, 57245], [338308], [318333, 318333], [68242], [185
896], [277119, 251130, 261940], [63769, 135703, 150185, 100367, 351982, 351982], [103311], [428257], [256146, 41058
7, 140527, 124065, 120098, 246997], [167827, 387504, 390167], [201598], [367664], [10034], [159822], [380493], [899
77], [383396, 321712, 9376, 339917, 119736, 403566, 345211, 164035, 116475, 30024, 268335, 416012, 15466, 325310, 1
17677, 119736, 331517, 23912, 307035, 4001, 444931, 72944, 241555, 362606, 134191, 355994, 317978, 119736, 136553,
404936, 408218, 461686, 24728, 369158, 74518, 422964, 439679, 391662, 332691, 5848, 94521, 454640, 411886, 190806,
325999, 26634, 390250, 257040, 76414, 97158, 148666, 119736, 137102, 275339, 456056, 163440, 113440, 99667, 318941,
```

- ❖ Now we have the purchased item list now we will define a function to recommend items that were bought together

```
# Write a function that would show items that were bought together (same of different dates)
def recommender_bought_bought(item_id, purchased_items):

    # Perhaps implement a binary search for that item id in the list of arrays
    # Then put the arrays containing that item id in a new list
    # Then merge all items in that list and get rid of duplicates
    recommender_list = []
    for x in purchased_items:
        if item_id in x:
            recommender_list += x

    #Then merge recommender list and remove the item id
    recommender_list = list(set(recommender_list) - set([item_id]))

    return recommender_list
```

❖ Here purchased_items are those items which we extracted in the before step now we will give one item id and will see the recommendation items.

```
n [136]: 1 recommender_bought_bought(80582,purchased_items)
ut[136]: [105792, 200793, 12836, 380775, 15335, 400969, 25353, 302422, 237753, 317178]
```

❖ So this is one basic way of recommending products to visitor a list of other items a customer previously bought along with what item the current visitor is viewing.

2.3 Valuable insights

- We can also get some valuable insights from the data like how many have only viewed, added to cart and transitioned.

```
In [152]: 1 Events['event'].value_counts()
Out[152]: view           2664312
          addtocart        69332
          transaction      22457
          Name: event, dtype: int64
```

- Now we will see the viewers rate in the E-commerce site

```
All_visitors=Events.visitorid.sort_values().unique()
```

```
All_visitors.size
```

```
1407580
```

```
#Buying visitors
Buying_visitors=Events[Events.event=='transaction'].visitorid.sort_values().unique()
```

```
Buying_visitors
Buying_visitors.size
```

```
11719
```

```
viewing_visitors_list = list(set(All_visitors) - set(Buying_visitors))
```

```
len(viewing_visitors_list)
```

```
1395861
```

- So out of 1407,580unique customers 13,95,861are just viewers. Will get deep into this. We will go check the buying visitors info.

- we will create a function for the buying customers from knowing the products viewed, total no of views for that product, total no of purchases and finally put either zero or one if they made a purchase.

In [58]:

```
def create_dataframe(visitor_list):

    array_for_df = []
    for index in visitor_list:

        #Create that visitor's dataframe once
        v_df = Events[Events.visitorid == index]

        temp = []
        #Add the visitor id
        temp.append(index)

        #Add the total number of unique products viewed
        temp.append(v_df[v_df.event == 'view'].itemid.unique().size)

        #Add the total number of views regardless of product type
        temp.append(v_df[v_df.event == 'view'].event.count())

        #Add the total number of purchases
        number_of_items_bought = v_df[v_df.event == 'transaction'].event.count()
        temp.append(number_of_items_bought)

        #Then put either a zero or one if they made a purchase
        if(number_of_items_bought == 0):
            temp.append(0)
        else:
            temp.append(1)

        array_for_df.append(temp)

    return pd.DataFrame(array_for_df, columns=['visitorid', 'num_items_viewed', 'view_count', 'bought_count'
                                               , 'purchased'])

```

- First step in the above code represents extracting the data from buying visitors, creating that visitor's data frame once, adding the visitor id to another list, adding the total no of unique products viewed in the same list, adding the total number of views regardless of the product type, then adding the total number of purchases and then assigning zero or one if they made a purchase

```
In [106]: 1 Buying_visitors

Out[106]: array([    172,     186,     539, ..., 1406087, 1406787, 1407110],
          dtype=int64)

In [167]: 1 buying_visitors_df=create_dataframe(Buying_visitors)

In [168]: 1 buying_visitors_df
```

Out[168]:

	visitorid	num_items_viewed	view_count	bought_count	purchased
0	172	22	33	2	1
1	186	1	2	1	1
2	264	2	3	2	1
3	419	3	4	1	1
4	539	1	4	1	1

- So we are now done with the data frame of buying visitors information, so we have 11719 buying unique visitors

```
In [210]: 1 buying_visitors_df=create_dataframe(Buying_visitors)

In [283]: 1 buying_visitors_df.shape

Out[283]: (11719, 5)
```

```
In [214]: 1 len(viewing_visitors_list)

Out[214]: 1395861
```

- So we now have the Final data Since the no .of viewers are more than purchased we will take 70-30 split
  - ❖ 11719-(30%)
  - ❖ X-(70%)
- By solving this we get 27344 as 70%,Now we will shuffle the viewers list for randomness and apply it to the create_dataframe function to get the viewers data

```
In [239]:
#Let's shuffle the viewing visitors list for randomness
import random
random.shuffle(viewing_visitors_list)
```

```
In [240]:
viewing_visitors_list
```

```
Out[240]: [1352647,
 1068371,
 694219,
 1305944,
 839707,
 1095176,
 636087,
 906141,
 868502,
 62929,
 141040,
 978701,
 1349768,
 75496,
 611948,
 1364767,
 35273,
 1223920,
 248203,
```

```
In [241]:
viewing_visitors_df = create_dataframe(viewing_visitors_list[0:27344])
```

```
In [242]:
viewing_visitors_df.shape
```

```
Out[242]: (27344, 5)
```

- Now we will concat the viewers data and purchase data to get the final result

```
In [243]:
main_df = pd.concat([buying_visitors_df, viewing_visitors_df], ignore_index=True)
```

2.4 EDA

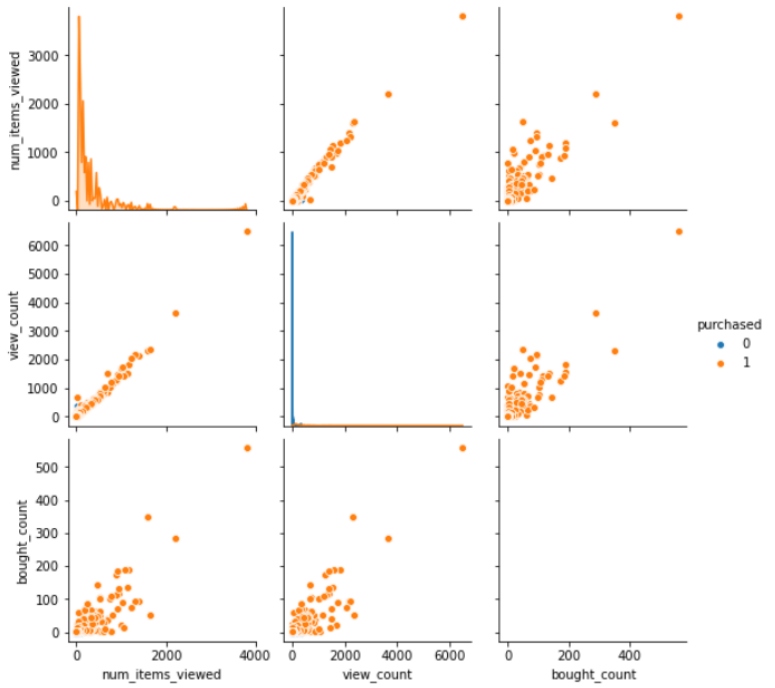
We will now perform Exploratory data analysis for the Final data

- Pair plot

```
In [285]:
#Let's shuffle main_df first
main_df = main_df.sample(frac=1)
```

```
In [286]:
sns.pairplot(main_df, x_vars = ['num_items_viewed', 'view_count', 'bought_count'],
             y_vars = ['num_items_viewed', 'view_count', 'bought_count'],  hue = 'purchased')
```

```
Out[286]: <seaborn.axisgrid.PairGrid at 0x1f481697b80>
```



The above plot shows that higher the view counts the higher the chances of the visitor buying products

3.1 Data Analysis

- Below are the basic steps that we use for Data analysis

```
Below are the steps used for analysis
1.Importing packages
2.Loading data
3.Data preparation:
    *Statistical Summary
    *Splitting target variable
    *Feature engineering
    *Splitting dataframe into numerical and categorical columns.
    *Reducing skewness
    *Mean Normalization
    *Dummy coding
4.Missing data analysis:
    *Missing values in the categorical columns
    *Missing values in the numerical columns
5.Correlation:
    *Correlation plot
6.Model building
```

- Importing packages from necessary libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score,KFold
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

- Statistical Summary:

```
main_df.describe()
```

	visitorid	num_items_viewed	view_count	bought_count	purchased
count	3.906300e+04	39063.000000	39063.000000	39063.000000	39063.000000
mean	7.083368e+05	3.776745	5.856949	0.574892	0.300003
std	4.057171e+05	37.582108	59.229652	4.926412	0.458265
min	7.300000e+01	0.000000	0.000000	0.000000	0.000000
25%	3.576835e+05	1.000000	1.000000	0.000000	0.000000
50%	7.136250e+05	1.000000	1.000000	0.000000	0.000000
75%	1.058981e+06	2.000000	3.000000	1.000000	1.000000
max	1.407521e+06	3809.000000	6479.000000	559.000000	1.000000

- Splitting target variable and checking whether it is balanced/not:

```
In [377]:
x=main_df.iloc[:,:4]
x.head(5)
```

Out[377]:

	visitorid	num_items_viewed	view_count	bought_count
6219	760430	61	122	4
12568	431650	1	1	0
10181	1224201	12	15	7
33992	550488	1	1	0
21309	1263965	1	1	0

```
In [250]:
y=main_df['purchased']
y_df=pd.DataFrame(y)
```

```
In [251]:
y_df['purchased'].value_counts()/y_df.shape[0]
#70-30 More likely balanced data
```

```
Out[251]: 0    0.699997
          1    0.300003
          Name: purchased, dtype: float64
```

- We will balance the dependent variable perfectly by using smote(synthetic minority over sampling)

```
In [164]:
from imblearn.over_sampling import SMOTE as sm
oversample=sm()
x,y=oversample.fit_resample(inde,dep)
```

```
In [168]:
y_df=pd.DataFrame(y)
y_df['purchased'].value_counts()
```

```
Out[168]: 1    27344
          0    27344
          Name: purchased, dtype: int64
```

- Feature Engineering: New features from the features

Since we have no columns to be converted into string or any columns to be clubbed, we will pass this step.

- Splitting data frame into numerical and categorical

This step is actually performed to reduce skewness and normalize the data for numerical columns and for categorical columns we generally do Dummy trial imputation, Since all the columns we extracted are counts and views we will also pass this step

- Reducing skewness

This step is for numerical features to reduce the skewness, it is the measure of asymmetry of the probability distribution.

- Mean normalization

We use this step for normalizing the data we actually use mean max normalization after skewness, If it is used before skewness we use min max normalization. Generally If we don't know the distribution we go for normalization, If we know the distribution is a normal distribution we go for standardization.

- Dummy coding

A Dummy variable is a binary variable that indicates whether a separate categorical variable takes on a specific value. This step is used for the categorical features, to convert them into numerical by pandas 'pd.get_dummies'

- Missing data Analysis

This step is used to fill the missing values by various methods, we will use knn imputer to fill missing the categorical and numerical columns. We also use mean, median mode to impute the missing values

- ❖ Mean→For the normal data
- ❖ Median→ For the skewed data
- ❖ Mode→For categorical data

```
1 main_df.isnull().sum()
```

```
visitorid           0
num_items_viewed    0
view_count          0
bought_count        0
purchased           0
dtype: int64
```

- Correlation

Correlation is the linear association ship and relationship between variables. We will know the correlation between the variables by using seaborn heatmap.

```
1 corr=x.corr()
2 sns.heatmap(corr,annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x257e0d4b310>
```

- Feature Selection

Process of trimming down a large data frame so that we can identify the factors which largely affect our target variable. Most machine learning models tend to show a dip in accuracy when the no of variables is significantly higher than the optimal.

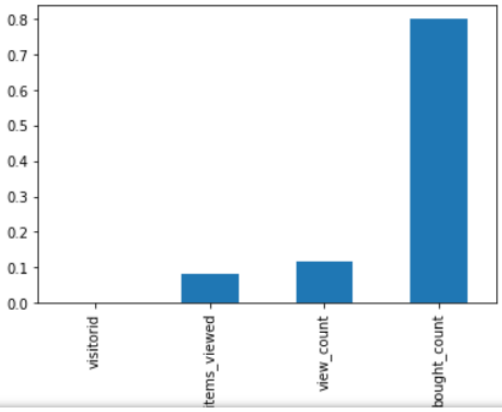
Feature importance

In [198]:
```
from sklearn.ensemble import ExtraTreesClassifier
model=ExtraTreesClassifier()
model.fit(x,y)
print(model.feature_importances_)
```

```
[0.00111346 0.08258094 0.11499132 0.80131428]
```

In [200]:
```
feature_importance=pd.Series(model.feature_importances_,index=x.columns)
feature_importance.plot(kind='bar')
```

Out[200]: <matplotlib.axes._subplots.AxesSubplot at 0x257b910d070>



3.3 Model Building

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data.

In [252]:
```
X_train, X_test, y_train, y_test = train_test_split(main_df, y_df, random_state = 42, train_size = 0.7)
```

- Now we will compare all the models and will find the optimum model which will give us the good accuracy.

In [253]:
```
model=[]
model.append(('LR',LogisticRegression()))
model.append(('LDA',LinearDiscriminantAnalysis()))
model.append(('DT',DecisionTreeClassifier()))
model.append(('RF',RandomForestClassifier()))
model.append(('NB',GaussianNB()))
model.append(('SVC',SVC()))
model.append(('KNN',KNeighborsClassifier()))
model.append(('XGB',XGBClassifier()))
```

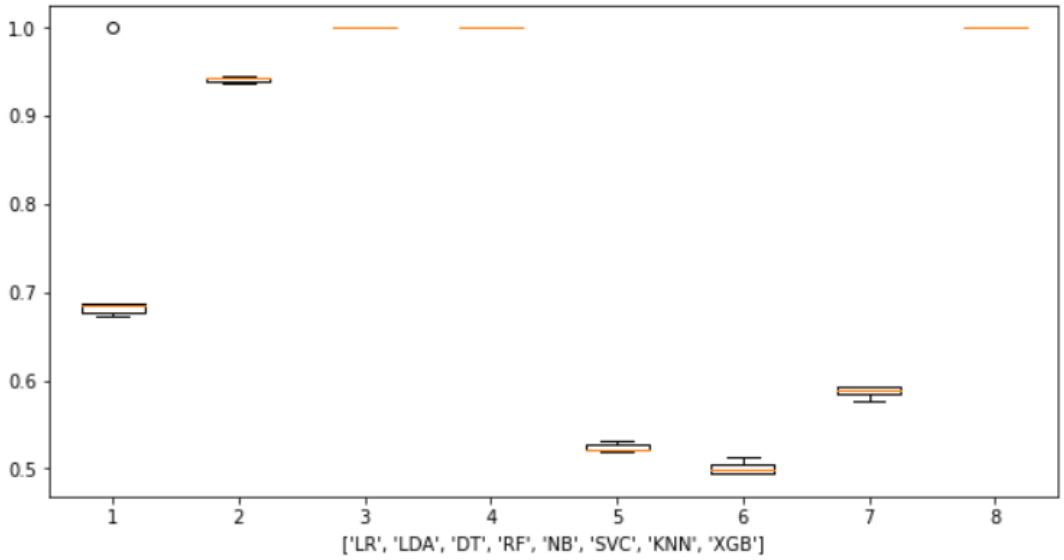
:
```
seed=7
results=[]
names=[]
scoring='accuracy'
for name,models in model:
    kfold=KFold(n_splits=5,random_state=seed)
    cv_results=cross_val_score(models,X_train,y_train,cv=kfold,scoring=scoring)
    results.append(cv_results)
    names.append(name)
    final_output=name,cv_results.mean(),cv_results.std()
    print(final_output)
```

```
('LR', 0.7443133423374627, 0.12794059498982108)
('LDA', 0.9421383633637014, 0.002729312253062299)
('DT', 1.0, 0.0)
('RF', 1.0, 0.0)
('NB', 0.5234712529045413, 0.004612353122644117)
('SVC', 0.5012148990092319, 0.007156084576113011)
('KNN', 0.5875761437789422, 0.005917396021854975)
('XGB', 1.0, 0.0)
```

- We evaluate the score by using cross validation, cross validation is a technique for evaluating ML models on the subsets of the input data and evaluating them on the complementary subsets of data. Benefits of using Cross validation technique is detect overfitting. I have used k-fold cross validation in which we split the input data into k subsets of data(also known as folds).We train an ML model on all but one(k-1) of the subsets, then evaluate the model on the subset that was not used for the training. This process is repeated k times with a different subset reserved for the evaluation (and excluded from training)

```
fig=plt.figure(figsize=(10,5))
plt.boxplot(results)
plt.xlabel(names)
```

```
Text(0.5, 0, "['LR', 'LDA', 'DT', 'RF', 'NB', 'SVC', 'KNN', 'XGB']")
```



- The above boxplot tells us the performance of various models, outliers, we will know the measures of central tendency-Mean, Median, Mode, skewness and their variations.
- Since Random forest also have the least Standard deviation which means the amount of variation is less. Also we have 100% accuracy in Random forest, Decision tree and XGBoost. We will check for the test accuracy as well.

CHAPTER4: FINALMODEL

Why Random Forest:

- A combination of learning models which increases the classification accuracy
- It uses Boruta algorithm for feature selection
- Random forest uses Max voting for the predictions, It has uncorrelated decision trees

- The algorithm decreases the variance which in turn decreases the chances of overfitting

```
In [174]:
1 model=RandomForestClassifier().fit(X_train,y_train)
2 predtest=model.predict(X_test)
3 accuracy_score(y_test,predtest)

Out[174]: 1.0
```

```
In [175]:
1 model=RandomForestClassifier().fit(X_train,y_train)
2 predtrain=model.predict(X_train)
3 accuracy_score(y_train,predtrain)

Out[175]: 1.0
```

- So I will choose Random forest for the model prediction, Also for a classifier we will also look up into ROC_AUC curve, F1 score, precision,Recall,Cohen kappa score to find the optimized model for the classification problem.

```
1 from sklearn.metrics import classification_report
2 from sklearn.metrics import cohen_kappa_score
3 from sklearn.metrics import f1_score
4 from sklearn.metrics import roc_curve,roc_auc_score
```

Classification Report

It Build a text report showing the main classification metrics like precision,recall,f1 score, support

```
: 1 print(classification_report(y_test,predtest))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	8172
1	1.00	1.00	1.00	8235
accuracy			1.00	16407
macro avg	1.00	1.00	1.00	16407
weighted avg	1.00	1.00	1.00	16407

```
: 1 print(classification_report(y_train,predtrain))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	19172
1	1.00	1.00	1.00	19109
accuracy			1.00	38281
macro avg	1.00	1.00	1.00	38281
weighted avg	1.00	1.00	1.00	38281

Cohen Kappa score:

The kappa statistic, which is a number between -1 and 1. The maximum value means complete agreement; zero or lower means chance agreement.

```
1 cohen_kappa_score(y_train,predtrain)
```

```
1.0
```

```
1 cohen_kappa_score(y_test,predtest)
```

```
1.0
```

ROC AUC CURVE

- Trade-off between sensitivity(True positive Rate) and Specificity(False positive rate).
- The closer the curve follows the left-hand border and then the top border of the ROC space, more accurate the test .
- The closer the curve comes to 45 degree diagonal of the ROC curve, less accurate the test
- The area under the curve is a measure of the text accuracy.

- Predicted purchase of the E-commerce site is given below:

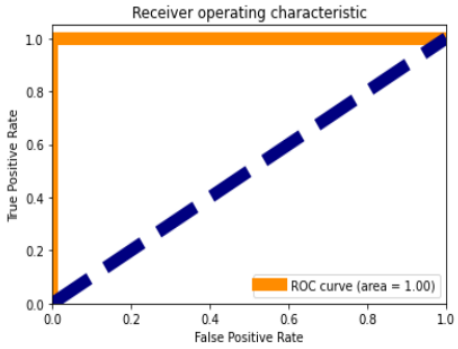
```python
# Generate the prediction values for each of the test observations using predict_proba() function rather than just p
preds = model.predict_proba(X_test)[:,1]

# Store the false positive rate(fpr), true positive rate (tpr) in vectors for use in the graph
fpr, tpr, _ = metrics.roc_curve(y_test, preds)

# Store the Area Under the Curve (AUC) so we can annotate our graph with theis metric
roc_auc = metrics.auc(fpr, tpr)

# Plot the ROC Curve
plt.figure()
lw = 10
plt.plot(fpr, tpr, color='darkorange', lw = lw, label = 'ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color = 'navy', lw = lw, linestyle = '--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc = "lower right")
```

```
<matplotlib.legend.Legend at 0x257bc1e6f70>
```



CONCLUSION

Out of all the algorithms used, Random forest is the better model which gave us 100% accuracy despite of hyperparametric tuning. Also it has the least Standard deviation and the evaluation metrics go best with this model.