Shoppers intention

‘online_shoppers_intention.csv’ on UCI Machine Learning Repository consists of 12,330 sessions with 84.5% negative class samples, i.e.; it was not ended with shopping. Only 15.5% did shopping.

Data set has 18 attributes.

Repository dataset as it is filtered to make sure 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.

```
Index(['Administrative', 'Administrative_Duration', 'Informational',
       'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration',
       'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month',
       'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',
       'Weekend', 'Revenue'],
      dtype='object')
```

DATA VISUALIZATION :

```
df[['Administrative', 'Administrative_Duration', 'Informational', 'Informational_Duration', 'ProductRelated','BounceRat
```

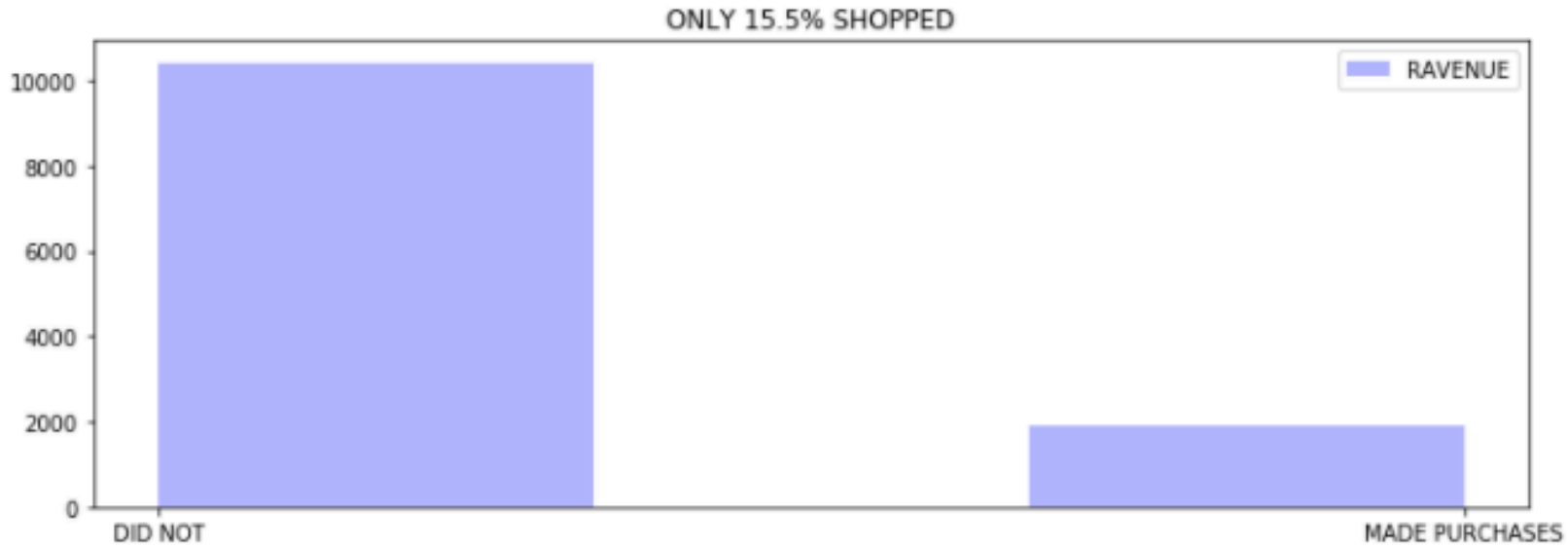
	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	BounceRates	ExitRates	PageValues	SpecialDay	Month
0	0.0	0.0	0.0	0.0	1.0	0.20	0.20	0.0	0.0	Feb
1	0.0	0.0	0.0	0.0	2.0	0.00	0.10	0.0	0.0	Feb
2	0.0	-1.0	0.0	-1.0	1.0	0.20	0.20	0.0	0.0	Feb
3	0.0	0.0	0.0	0.0	2.0	0.05	0.14	0.0	0.0	Feb
4	0.0	0.0	0.0	0.0	10.0	0.02	0.05	0.0	0.0	Feb

```
df [ ['OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType','Weekend', 'Revenue'] ].head()
```

	OperatingSystems	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
0	1	1	1	1	Returning_Visitor	False	False
1	2	2	1	2	Returning_Visitor	False	False
2	4	1	9	3	Returning_Visitor	False	False
3	3	2	2	4	Returning_Visitor	False	False
4	3	3	1	4	Returning_Visitor	True	False

DATA VISUALIZATION :

REVENUE

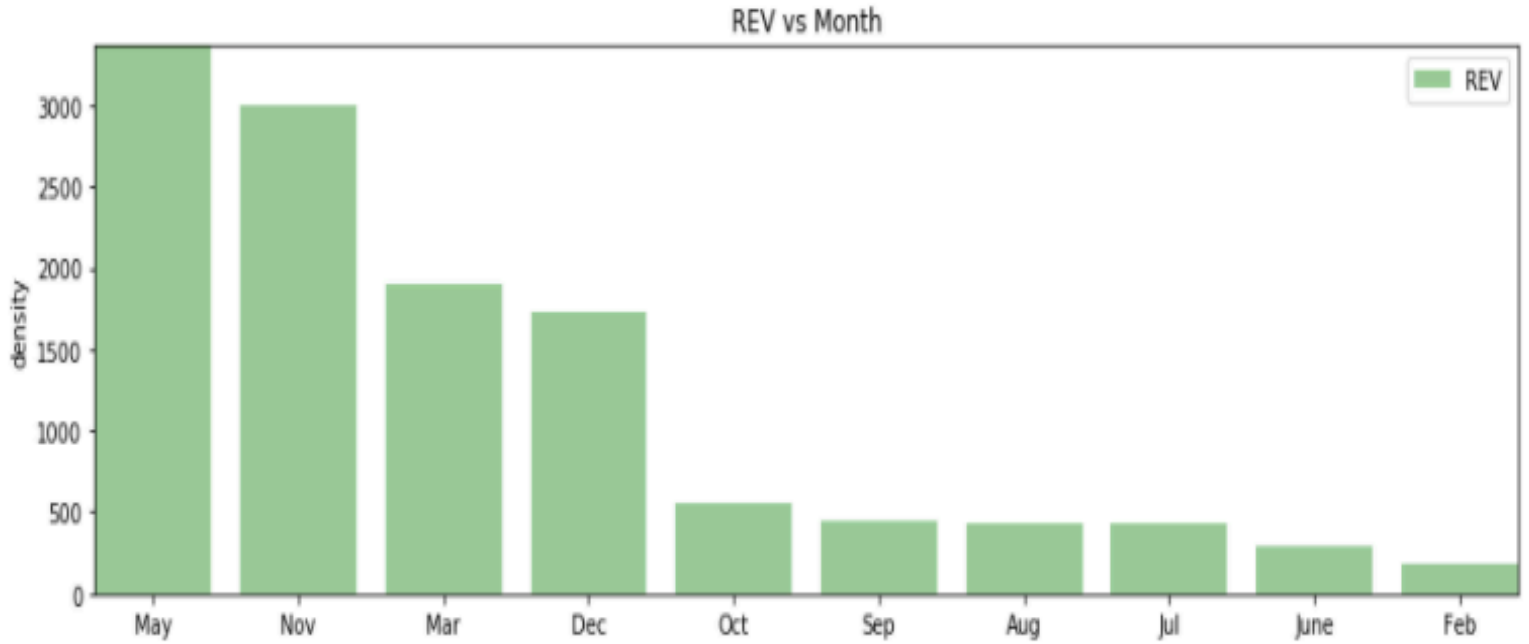



```
df['Revenue'].value_counts()[0]/df['Revenue'].value_counts().sum(), df['Revenue'].value_counts()[1]/df['Revenue'].value
```

```
(0.8452554744525548, 0.15474452554744525)
```

```
plt.figure(figsize=(12,4))
df['Revenue'].apply(lambda x: 1 if (x==True) else False ).hist(bins=3, alpha=0.3, color='blue', label='RAVENUE')
plt.grid()
plt.xticks([0,1],('DID NOT', 'MADE PURCHASES'))
plt.autoscale(enable=True)
plt.legend()
plt.title(' ONLY 15.5% SHOPPED ')
plt.show()
```

REVENUE VS MONTH :



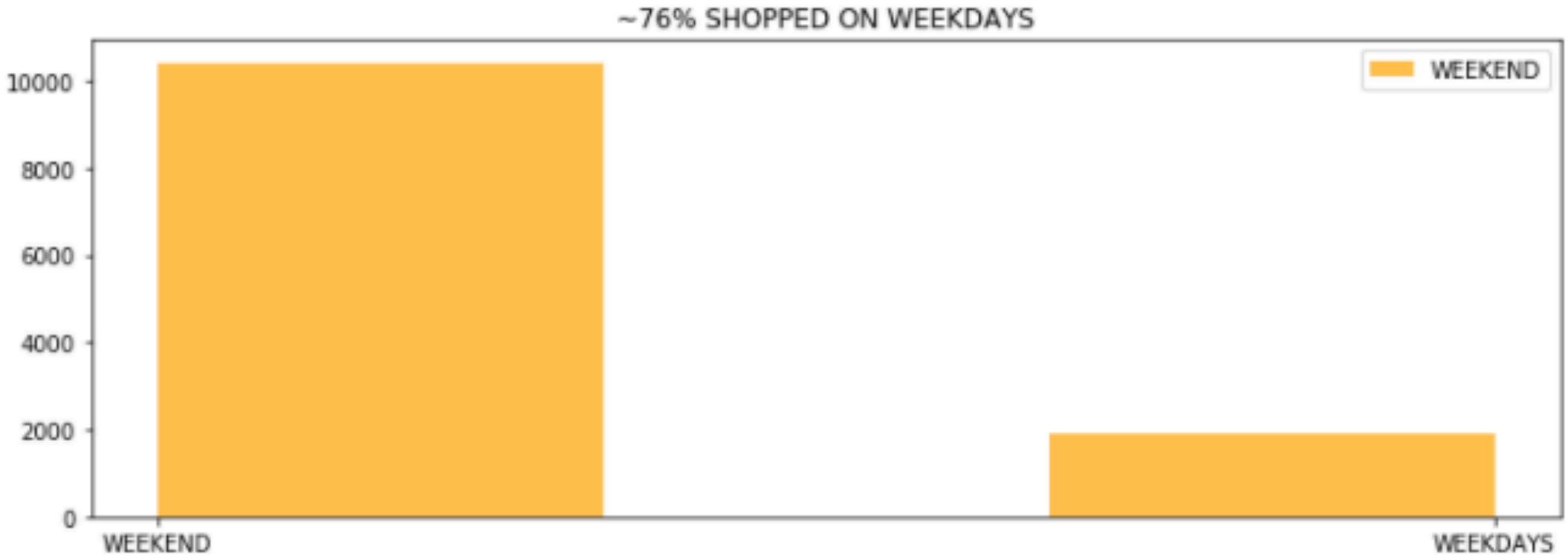

```
df1 = df[ df['Revenue']==True ]
df2 = df[ df['Revenue']==False ]
```

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

```
May      3364
Nov      2998
Mar      1907
Dec      1727
Oct       549
Sep       448
Aug       433
Jul       432
June      288
Feb       184
```

As usual MAY, NOV, MAR and DEC are the months for max sales

REVENUE



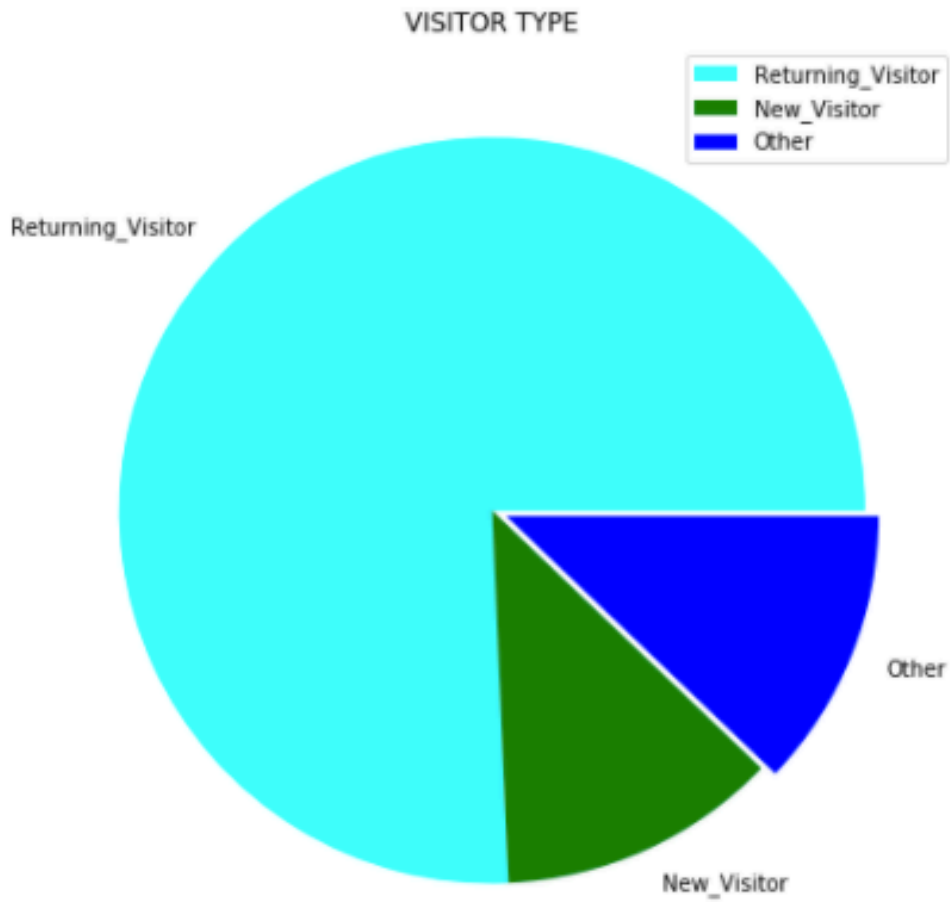
WEEKEND	0.23
WEEK-DAYS	0.76

VISITOR TYPE :

VISITOR TYPE

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

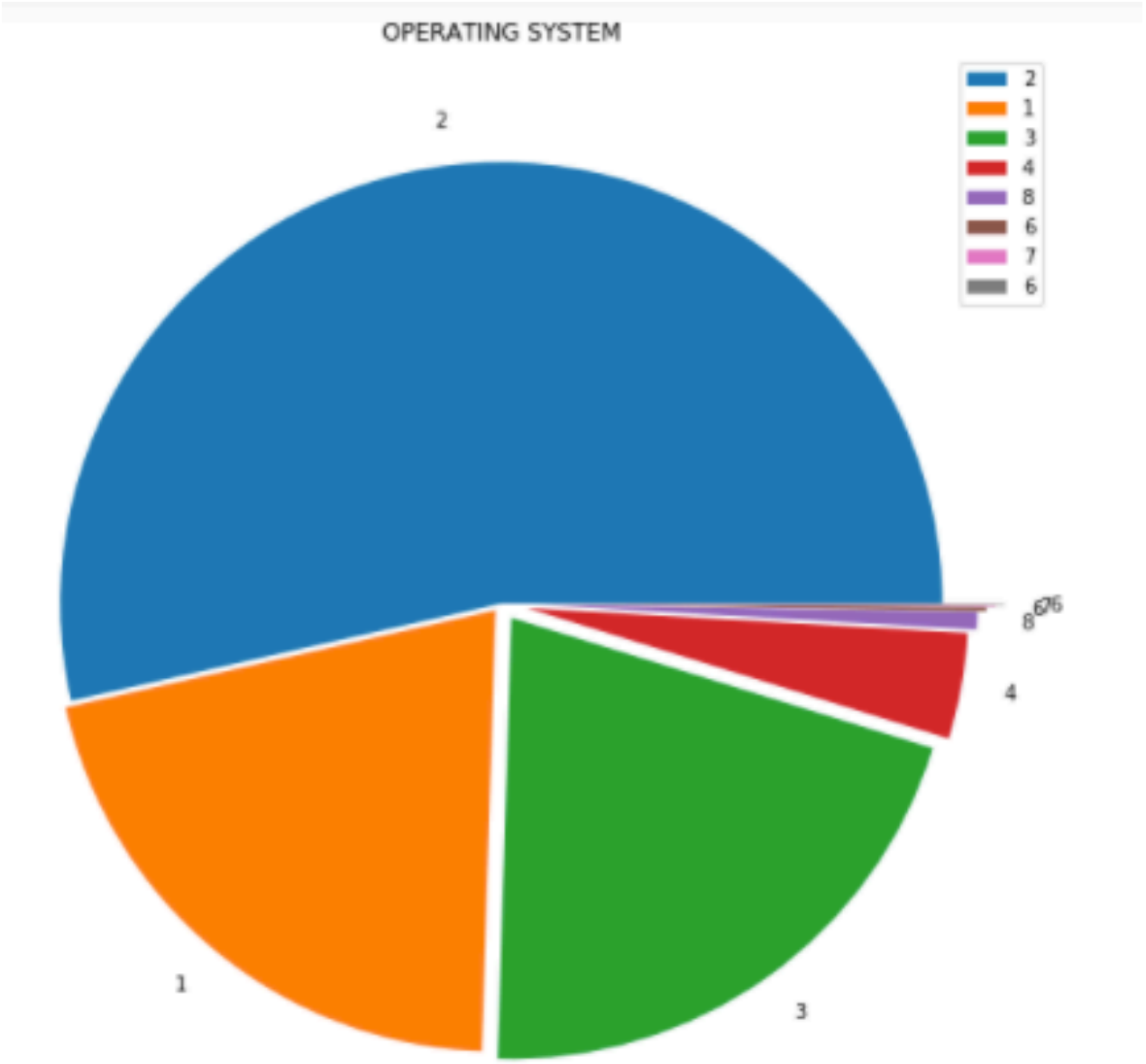
```
Returning_Visitor    10551
New_Visitor           1694
Other                   85
Name: VisitorType, dtype: int64
```



OPERATING SYSTEMS :

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

```
2    6601
1    2585
3    2555
4     478
8      79
6      19
7       7
5       6
Name: OperatingSystems, dtype: int64
```



Statistics :

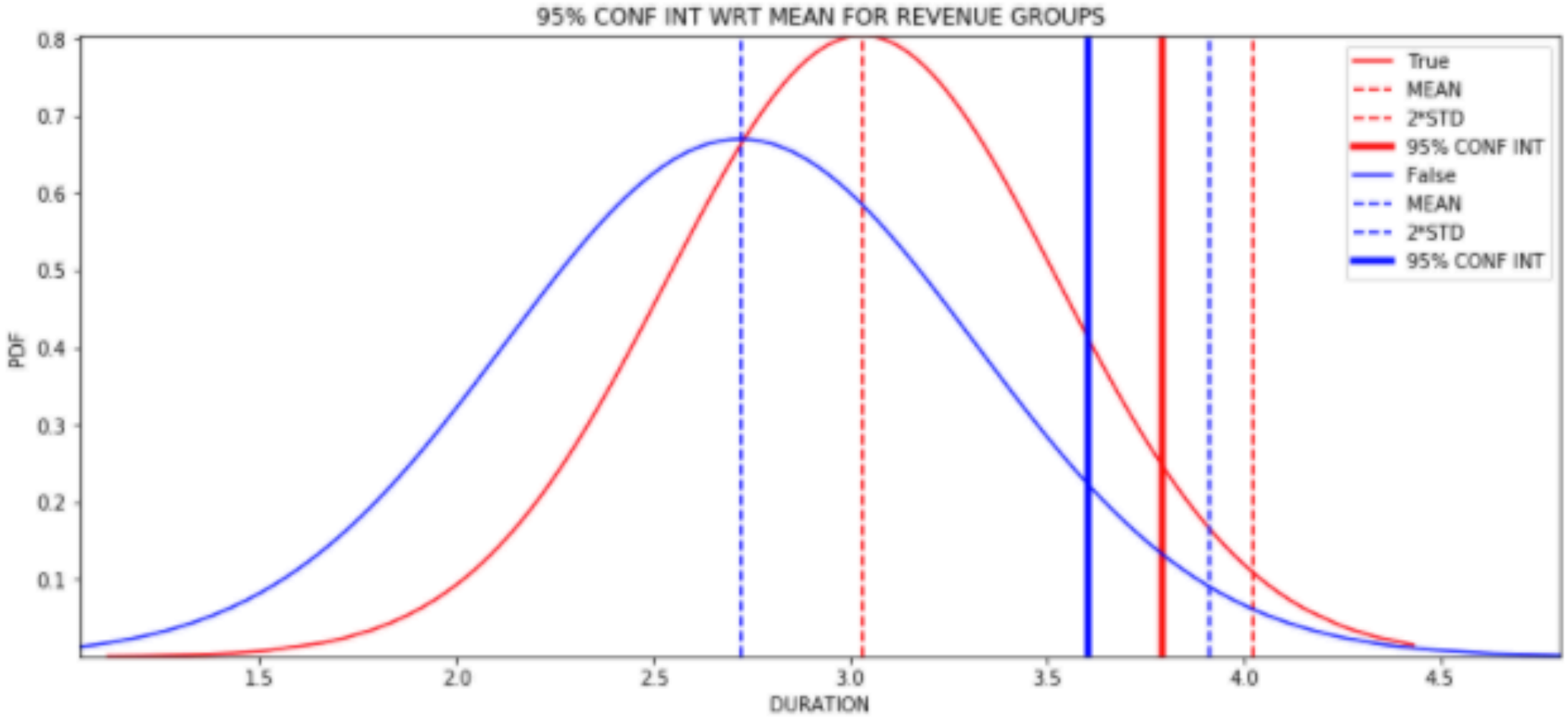
```
len(df.columns), df.columns[0:6], df.columns[8], df.columns[17],
```

```
(18, Index(['Administrative', 'Administrative_Duration', 'Informational',
        'Informational_Duration', 'ProductRelated', 'ProductRelated_Duration'],
       dtype='object'), 'PageValues', 'Revenue')
```

```
df.describe()
```

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates
count	12316.000000	12316.000000	12316.000000	12316.000000	12316.000000	12316.000000	12316.000000	12316.000000
mean	2.317798	80.906176	0.503979	34.506387	31.763884	1196.037057	0.022152	0.043003
std	3.322754	176.860432	1.270701	140.825479	44.490339	1914.372511	0.048427	0.048527
min	0.000000	-1.000000	0.000000	-1.000000	0.000000	-1.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	7.000000	185.000000	0.000000	0.014286
50%	1.000000	8.000000	0.000000	0.000000	18.000000	599.766190	0.003119	0.025124
75%	4.000000	93.500000	0.000000	0.000000	38.000000	1466.479902	0.016684	0.050000
max	27.000000	3398.750000	24.000000	2549.375000	705.000000	63973.522230	0.200000	0.200000

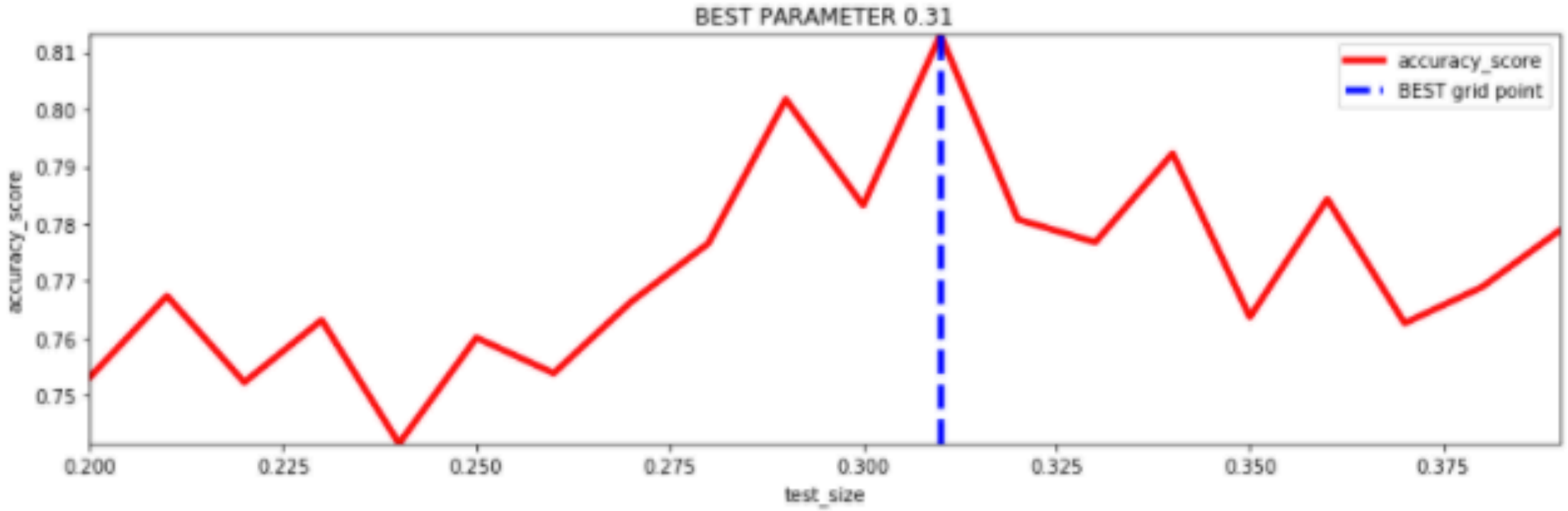
Inferential Statistics :



Based on shoppers' time spend on product, 95% confidence interval with respect to mean is close to 2 STD.

ML PREDICTION :

Logistic Regression




```
BEST PARAMETER 0.31
accuracy_score 0.81
```

Ensemble Gradient Boosting Classifier

```
x_tr, x_t, y_tr, y_t = train_test_split(df[attr].values,(df['REV_B']).values, test_size=0.30)
clf = ensemble.GradientBoostingClassifier()
clf.fit(x_tr, y_tr)
y_p = clf.predict(x_t)
y_tr_p = clf.predict(x_tr)
print('Accuracy score over the test set %0.2f' %accuracy_score(y_t, y_p))
print('Accuracy score over the training set %0.2f' %accuracy_score(y_tr, y_tr_p))
```

Accuracy score over the test set 0.79
Accuracy score over the training set 0.90

GaussianNB

Accuracy score over the test set 0.76
Accuracy score over the training set 0.77

Decision Tree Classifier

ROC Accuracy score over the test set 0.68
ROC score over the test set 0.59

Summary :

ML in Depth

```
MLclf = [LogisticRegression(), GaussianNB(), DecisionTreeClassifier(), ExtraTreesClassifier(), RandomForestClassifier(
roc_list = []
accu_list = []
for clf in MLclf:
    y_p = clf.fit(x_tr, y_tr).predict(x_t)
    roc_list.append(roc_auc_score(y_t, y_p))
    accu_list.append(accuracy_score(y_t, y_p))
    print(roc_auc_score(y_t, y_p))
```

```
0.5882930443641752
0.5900018523663981
0.6053996480503844
0.6001713438918218
0.6034731869963879
```

The best classifier from default parameter : Gradient Boosting Classifier()

with ROC score : 0.796895213454075