A

Data science

Project

on

“Smart Lead Scoring Engine”

Submitted by:

Santosh Arvind Dharam

ACKNOWLEDGMENT

I feel great pleasure to present the Project entitled “Smart Lead Scoring Engine”. But it would be unfair on our part if I do not acknowledge efforts without that this Project would not have been a success. I would like to express my sincere thanks and appreciation to ‘Analytics vidya’ for their JOB A THON. Most importantly I would like to express our sincere gratitude towards my Friend & Family for always being there when I needed them most.

Mr. Santosh Arvind Dharam

PROBLEM STATEMENT

A D2C startup develops products using cutting edge technologies like Web 3.0. Over the past few months, the company has started multiple marketing campaigns offline and digital both. As a result, the users have started showing interest in the product on the website. These users with intent to buy product(s) are generally known as leads (Potential Customers).

Leads are captured in 2 ways - Directly and Indirectly.

Direct leads are captured via forms embedded in the website while indirect leads are captured based on certain activity of a user on the platform such as time spent on the website, number of user sessions, etc.

Now, the marketing & sales team wants to identify the leads who are more likely to buy the product so that the sales team can manage their bandwidth efficiently by targeting these potential leads and increase the sales in a shorter span of time.

Analytical Problem Framing

EDA steps:

1) import necessary libraries:

first we will import all the necessary libraries which will be usefull for analysis of data

```
In [1]: #import all libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
from sklearn.linear_model import LogisticRegression,Lasso,LinearRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor,GradientBoostingRegressor
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.decomposition import PCA
from scipy.stats import zscore
from sklearn.model_selection import cross_val_score
```

in thid case we have to import all the necessary library that are usefull for data analysis in jupyter notebook

2)extract the dataset in jupyter notebook:

```
In [2]: data=pd.read_csv("C://Users//SAI BABA//Desktop//Job a thon//train_wn75k28.csv")
data.head()
```

Out[2]:

	id	created_at	campaign_var_1	campaign_var_2	products_purchased	signup_date	user_activity_var_1	user_activity_var_2	user_activity_var_3	user_acti
0	1	2021-01-01	1	2	2.0	2020-09-24	0	0	0	
1	2	2021-01-01	2	1	2.0	2020-09-19	1	0	1	
2	3	2021-01-01	9	3	3.0	2021-08-11	1	0	0	
3	4	2021-01-01	6	7	2.0	2017-10-04	0	0	0	
4	5	2021-01-01	4	6	NaN	2020-06-08	0	0	0	

```
In [3]: data.shape
```

Out[3]: (39161, 19)

data contains 39161 rows and 19 columns

lets we will check the null valuse present in our dataset if any..

Data is extracted for further analysis in jupyter notebook

data contains 39161 rows and 19 columns
lets we will check the null valuse present in our dataset if any..

2) Encoding the dataset:
In this case as our data contains some object column having object type of data it is necessary to convert it into numerical form by LabelEncoder

data encoding

```
In [6]: from sklearn.preprocessing import LabelEncoder

In [7]: le=LabelEncoder()
        label=le.fit_transform(data["created_at"])
        label
        data=data.drop("created_at",axis='columns')
        data["created_at"]=label

In [8]: le=LabelEncoder()
        label=le.fit_transform(data["signup_date"])
        label
        data=data.drop("signup_date",axis='columns')
        data["signup_date"]=label
        data
```

Out[8]:

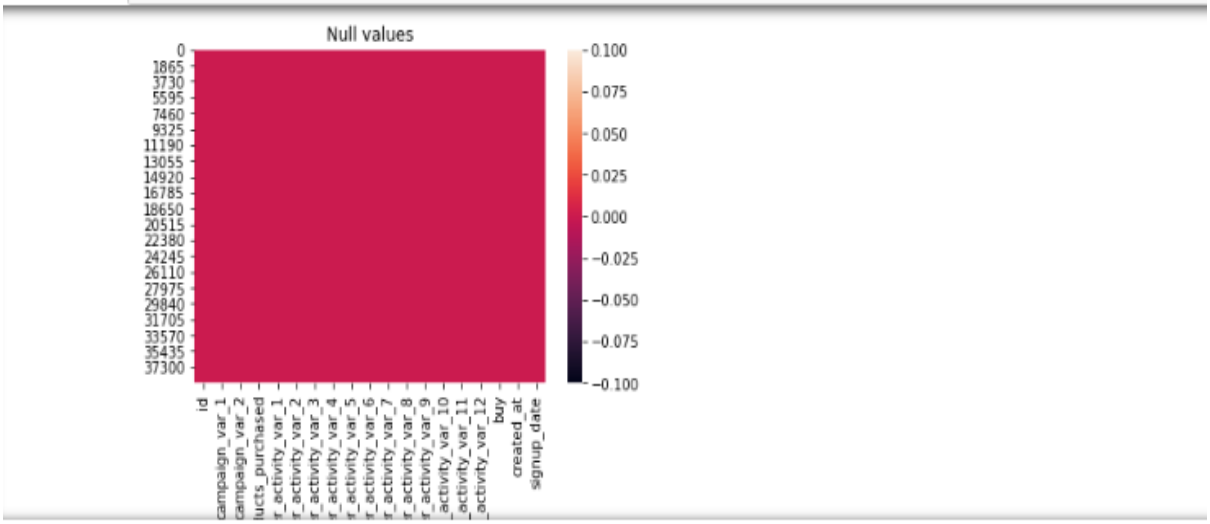
	id	campaign_var_1	campaign_var_2	products_purchased	user_activity_var_1	user_activity_var_2	user_activity_var_3	user_activity_var_4	user_a
0	1	1	2	2.0	0	0	0	0	
1	2	2	1	2.0	1	0	1	0	
2	3	9	3	3.0	1	0	0	0	
3	4	6	7	2.0	0	0	0	0	
4	5	4	6	NaN	0	0	0	0	
...	...	...	...	...	...	...	...	...	
39156	39157	11	11	2.0	1	0	0	0	
39157	39158	3	9	3.0	0	0	0	0	
39158	39159	8	7	2.0	1	0	0	0	
39159	39160	7	12	2.0	0	0	0	0	
39160	39161	2	5	NaN	1	0	0	0	

Data contains 19 columns and 39161 rows

4)checking null values:
In this case we have to find out the null values present in our data set if yes it is required to remove it. it is removed and now dataset does not contains any null

values which is shown by heatmap

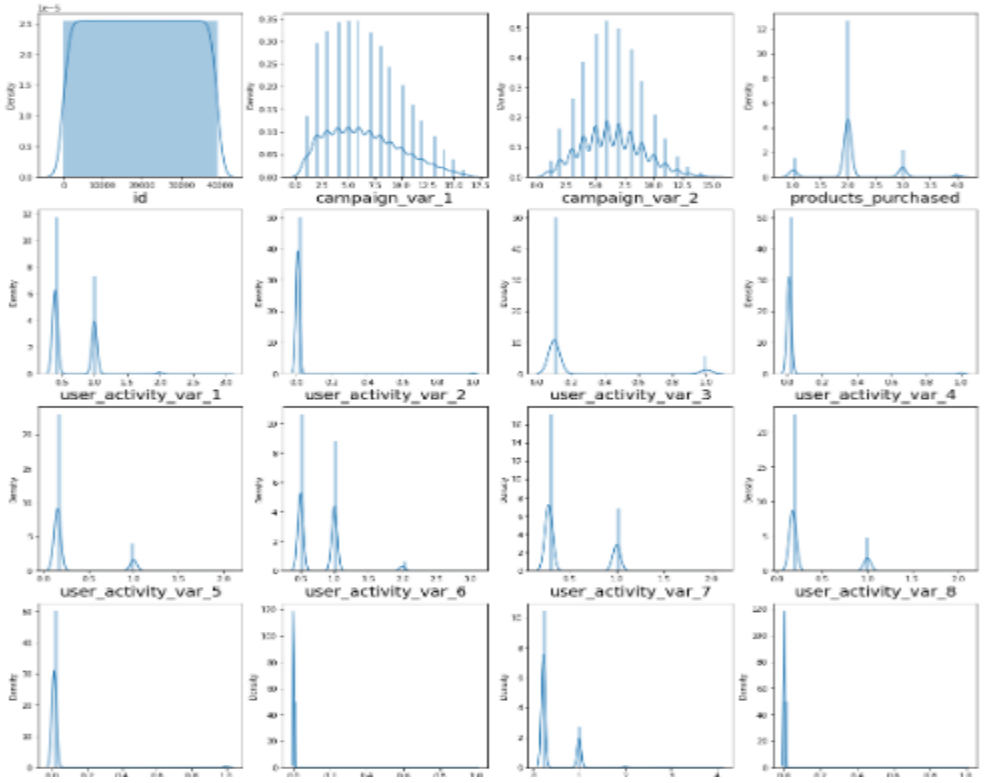
```
In [17]: sns.heatmap(data.isnull())
         plt.title("Null values")
         plt.show()
```



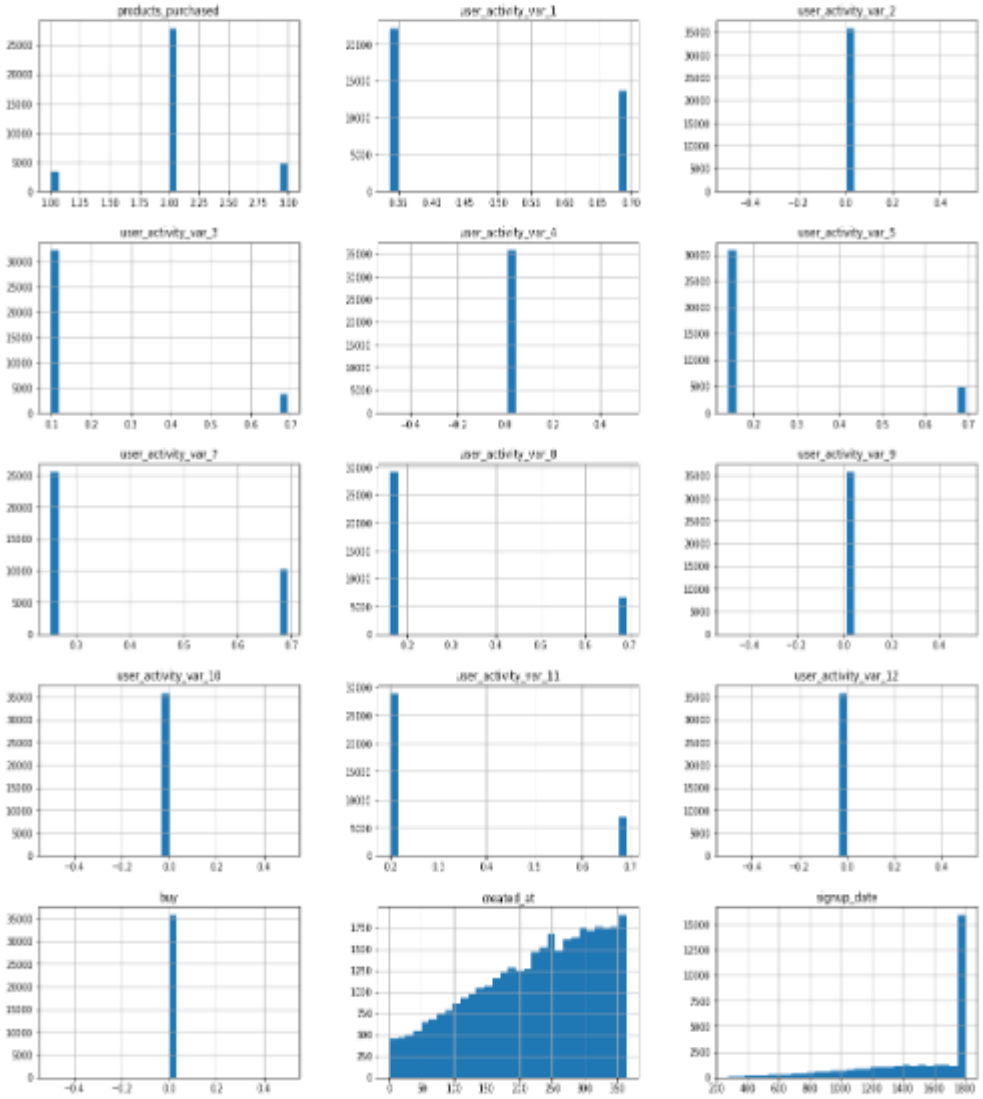
heat map shows that data does not contains any null values in it

5) visualization:

```
In [19]: #now lets see how data is distributed in each column
         plt.figure(figsize=(20,25),facecolor='white')
         plotnumber=1
         for column in data:
             if plotnumber<=19:
                 ax=plt.subplot(5,4,plotnumber)
                 sns.distplot(data[column])
                 plt.xlabel(column,fontsize=20)
             plotnumber+=1
         plt.show()
```



```
 <AxesSubplot:title={'center':'user_activity_var_11'}>,
 <AxesSubplot:title={'center':'user_activity_var_12'}>],
[<AxesSubplot:title={'center':'buy'}>,
 <AxesSubplot:title={'center':'created_at'}>,
 <AxesSubplot:title={'center':'signup_date'}>]], dtype=object)
```



Graph shows that data is uniformely distributed as we have removed outliers and null values in it

7) Data Correlation:

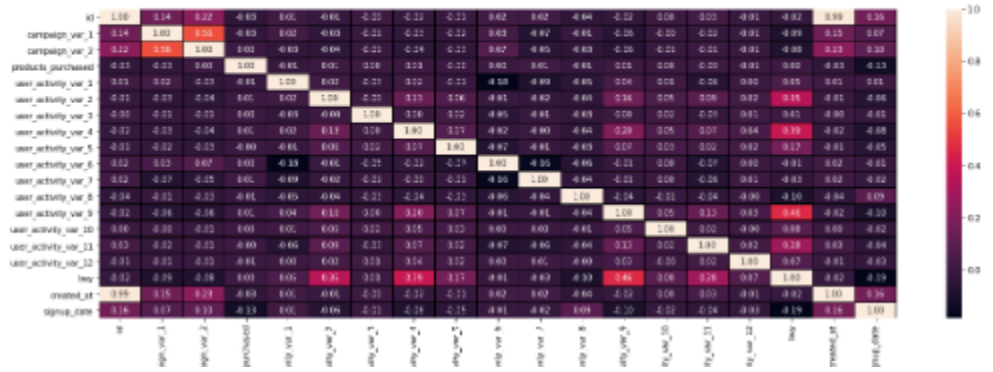
```
In [21]: data.corr()
```

Out[21]:

	id	campaign var 1	campaign var 2	products purchased	user activity var 1	user activity var 2	user activity var 3	user activity
id	1.000000	0.143723	0.222748	-0.030068	0.011285	-0.011075	-0.000572	-0.
campaign var 1	0.143723	1.000000	0.561489	-0.027047	0.023557	-0.034548	-0.012870	-0.
campaign var 2	0.222748	0.561489	1.000000	0.001733	-0.033137	-0.039833	-0.008048	-0.
products purchased	-0.030068	-0.027047	0.001733	1.000000	-0.010374	0.007820	0.002928	0.
user activity var 1	0.011285	0.023557	-0.033137	-0.010374	1.000000	0.016089	-0.025808	0.
user activity var 2	-0.011075	-0.034548	-0.039833	0.007820	0.016089	1.000000	-0.001075	0.
user activity var 3	-0.000572	-0.012870	-0.008048	0.002928	-0.025808	-0.001075	1.000000	0.
user activity var 4	-0.015203	-0.033510	-0.038928	0.007489	0.016419	0.132108	0.003023	1.
user activity var 5	-0.006841	-0.023151	-0.030998	-0.003080	-0.014431	0.061054	0.023084	0.
user activity var 6	0.015883	0.027985	0.066333	0.002700	-0.160008	-0.005904	-0.047802	-0.
user activity var 7	0.015888	-0.072677	-0.053510	0.009725	-0.065322	-0.020150	-0.006458	-0.
user activity var 8	-0.035248	-0.014312	-0.033888	-0.012487	-0.048974	-0.035302	-0.025671	-0.
user activity var 9	-0.019808	-0.058284	-0.057942	0.007955	0.035843	0.155794	0.003898	0.
user activity var 10	0.001002	-0.004851	-0.009879	0.004665	0.012004	0.046324	0.019152	0.
user activity var 11	0.025936	-0.023447	-0.005309	-0.000482	-0.060461	0.089337	-0.026822	0.
user activity var 12	-0.009223	-0.008389	-0.008228	-0.005170	0.002137	0.024440	0.013281	0.
buy	-0.020782	-0.087202	-0.080064	0.003556	0.048011	0.354827	0.005174	0.
created at	0.987469	0.146240	0.228339	-0.031341	0.009378	-0.009899	-0.000204	-0.
signup date	0.160149	0.068320	0.102779	-0.128335	0.009823	-0.083058	-0.013070	-0.

```
In [22]: #Heat map
plt.figure(figsize=(22,7))
sns.heatmap(data.corr(),annot=True,linewidths=0.1,linecolor='black',fmt='0.2f')
```

Out[22]: <AxesSubplot:>



Data is correlated with other column data and also with its own it also gives the positive negative correlation of data with respective one another

10) Removing outliers:

In [25]:
```
#making the skew less than or equal to 0.55 for better prediction and plotting Normal distribution
skew=('user_activity_var_1','user_activity_var_2','user_activity_var_3','user_activity_var_4','user_activity_var_5','user_activ
for col in skew :
    if data.skew().loc[col]>0.55:
        data[col] = np.log1p(data[col])
```

In [26]:
```
from scipy.stats import zscore
import numpy as np
z=np.abs(zscore(data))
threshold=3
np.where(z>3)
```

Out[26]:
```
(array([    3,    13,    18, ..., 39139, 39151, 39156], dtype=int64),
 array([14, 14, 14, ..., 12,  0, 14], dtype=int64))
```

In [27]:
```
data_new_z=data[(z<3).all(axis=1)]
data_new_z
```

Out[27]:

	products_purchased	user_activity_var_1	user_activity_var_2	user_activity_var_3	user_activity_var_4	user_activity_var_5	user_activity_var_7	user_acti
0	2.0	0.336538	0.006693	0.097881	0.0114	0.141068	0.693147	
1	2.0	0.693147	0.006693	0.693147	0.0114	0.141068	0.693147	
2	3.0	0.693147	0.006693	0.097881	0.0114	0.141068	0.252012	
4	2.0	0.336538	0.006693	0.097881	0.0114	0.141068	0.693147	
5	3.0	0.336538	0.006693	0.097881	0.0114	0.141068	0.252012	
...	...	...	...	...	...	...	...	
39155	2.0	0.336538	0.006693	0.097881	0.0114	0.141068	0.252012	
39157	3.0	0.336538	0.006693	0.097881	0.0114	0.141068	0.252012	
39158	2.0	0.693147	0.006693	0.097881	0.0114	0.693147	0.693147	
39159	2.0	0.336538	0.006693	0.097881	0.0114	0.141068	0.252012	
39160	2.0	0.693147	0.006693	0.097881	0.0114	0.141068	0.693147	

35795 rows × 15 columns

In [28]:
```
data_new_z.shape
```

Out[28]:
```
(35795, 15)
```

So outliers are removed from data after removing outliers we have 35795 rows and 15 column remains

12)Model building

In [41]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

In [42]:
```
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.24,random_state=40)
```

In [43]:
```
log_reg=LogisticRegression()
log_reg.fit(x_train, y_train)
preddt=log_reg.predict(x_test)
print(accuracy_score(y_test,preddt))
print(confusion_matrix(y_test,preddt))
print(classification_report(y_test,preddt))
```

```
0.974146185764443
[[8922   16]
 [ 227  234]]
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      8938
           1       0.94      0.51      0.66       461

    accuracy                           0.97      9399
   macro avg       0.96      0.75      0.82      9399
weighted avg       0.97      0.97      0.97      9399
```

In [44]:
```
dt=DecisionTreeClassifier()
dt.fit(x_train, y_train)
preddt=dt.predict(x_test)
print(accuracy_score(y_test,preddt))
print(confusion_matrix(y_test,preddt))
print(classification_report(y_test,preddt))
```

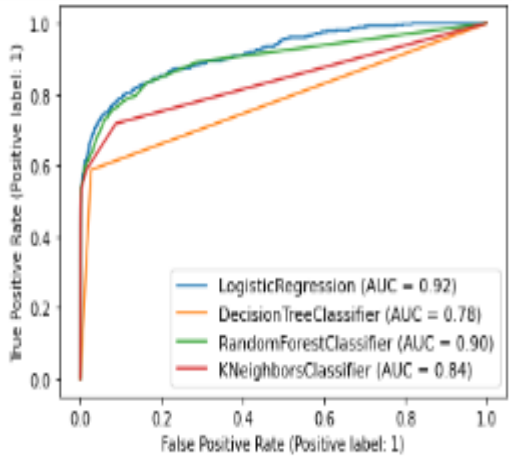
```
0.9551016065538888
[[8706  232]
 [ 190  271]]
              precision    recall  f1-score   support

           0       0.98      0.97      0.98      8938
           1       0.54      0.59      0.56       461

    accuracy                           0.96      9399
   macro avg       0.76      0.78      0.77      9399
weighted avg       0.96      0.96      0.96      9399
```

We have tested model though various algorithm and found a best suitable algorithm among them also we have drawn AUC-ROC CUREVE

13)Cross Validation:



from this it is found that LogisticRegression has good accuracy among all also we will cross validate the result further

```
[49]: log_reg=LogisticRegression()

[50]: log_reg.fit(x_train,y_train)

t[50]: LogisticRegression()
```

cross validation

```
[51]: from sklearn.model_selection import cross_val_score

[52]: for i in range(2,10):
          cv=cross_val_score(log_reg,x,y,cv=i)
          print(log_reg,cv.mean())
```

```
LogisticRegression() 0.9630246318701621
LogisticRegression() 0.9706851177538308
LogisticRegression() 0.971757632893146
LogisticRegression() 0.9713745909079824
LogisticRegression() 0.971400123891888
LogisticRegression() 0.9714001491905047
LogisticRegression() 0.9716299176080034
LogisticRegression() 0.9704553576726407
```

thus we have sucessfully validated the result we can also predict the output below

From cross validation also it is found that Logistic Regression() has good accuracy

15)Model Saving:

Saving Model

```
In [57]: #saving model to the local file system
filename='finalized_model_Smart_Lead_Scoring_Engine.pickle'
pickle.dump(log_reg,open(filename,'wb'))
#prediction using the saved model
Loaded_model=pickle.load(open(filename,'rb'))
a=Loaded_model.predict(scalar.transform([[2.000000,0.423036,0.006649,0.089276,0.011272,0.423036,0.526589,0.149097,0.011311,0.000
a
Out[57]: array([0], dtype=int64)

In [58]: #Adjusted R2 score
log_reg.score(x_train,y_train)
Out[58]: 0.9726496875209999

In [59]: #check how well model fits the test data
log_reg.score(x_test,y_test)
Out[59]: 0.974146185764443

In [60]: #lets plot and visualize
y_pred=log_reg.predict(x_test)
y_pred
Out[60]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

So we have saved model and also found prediction

16)conclusion:

```
In [80]: #saving model to the local file system
filename='finalized_model_Smart_Lead_Scoring_Engine.pickle'
pickle.dump(log_reg,open(filename,'wb'))
#prediction using the saved model
Loaded_model=pickle.load(open(filename,'rb'))
a=Loaded_model.predict(testdata_new_z)
a
Out[80]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

So in this way we have saved model and also it tested on test dataset and got prediction for the future