

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
In [2]: df1 = pd.read_csv('train_wn75k28.csv')
df1.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_v
0	1	2021-01-01	1	2	2.0	2020-09-24	0	0	
1	2	2021-01-01	2	1	2.0	2020-09-19	1	0	
2	3	2021-01-01	9	3	3.0	2021-08-11	1	0	
3	4	2021-01-01	6	7	2.0	2017-10-04	0	0	
4	5	2021-01-01	4	6	NaN	2020-06-08	0	0	

```
In [3]: df2 = pd.read_csv('test_Wf7sxXF.csv')
df2.head()
```

```
Out[3]:
```

	id	created_at	campaign_var_1	campaign_var_2	products_purchased	signup_date	user_activity_var_1	user_activity_var_2	user_activi
0	39162	2022-01-01	2	2	NaN	2021-08-17	1	1	
1	39163	2022-01-01	4	7	3.0	2020-05-21	1	0	
2	39164	2022-01-01	8	7	NaN	NaN	0	0	
3	39165	2022-01-01	9	8	2.0	2020-06-22	0	0	
4	39166	2022-01-01	4	5	2.0	2021-03-10	1	0	

```
In [4]: df3 = pd.read_csv('sample_submission_2zvVjBu.csv')
```

```
In [5]: df3.head()
```

```
Out[5]:
```

	id	buy
0	39162	1
1	39163	1
2	39164	1
3	39165	1
4	39166	1

```
In [6]: df = pd.merge(df1,df2,how='left')
```

```
In [7]: df.head()
```

```
Out[7]:
```

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

```
In [8]: df.tail()
```

Out[8]:	id	created_at	campaign_var_1	campaign_var_2	products_purchased	signup_date	user_activity_var_1	user_activity_var_2	user_
	39156	39157	2021-12-31	11	11	2.0	2017-10-19	1	0
	39157	39158	2021-12-31	3	9	3.0	NaN	0	0
	39158	39159	2021-12-31	8	7	2.0	NaN	1	0
	39159	39160	2021-12-31	7	12	2.0	NaN	0	0
	39160	39161	2021-12-31	2	5	NaN	2019-08-11	1	0

```
In [9]: df=pd.merge(df,df3,how='left')
```

```
In [10]: df.head()
```

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

```
In [11]: df.shape
```

```
Out[11]: (39161, 19)
```

```
In [12]: df.columns
```

```
Out[12]: Index(['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_activity_var_4',
            'user_activity_var_5', 'user_activity_var_6', 'user_activity_var_7',
            'user_activity_var_8', 'user_activity_var_9', 'user_activity_var_10',
            'user_activity_var_11', 'user_activity_var_12', 'buy'],
            dtype='object')
```

```
In [13]: df.isnull().sum()
```

```
Out[13]: id                0
created_at              0
campaign_var_1          0
campaign_var_2          0
products_purchased    20911
signup_date           15113
user_activity_var_1     0
user_activity_var_2     0
user_activity_var_3     0
user_activity_var_4     0
user_activity_var_5     0
user_activity_var_6     0
user_activity_var_7     0
user_activity_var_8     0
user_activity_var_9     0
user_activity_var_10    0
user_activity_var_11    0
user_activity_var_12    0
buy                    0
dtype: int64
```

```
In [14]: df=df.dropna()
```

```
In [15]: df.isnull().sum()
```

```
Out[15]: id 0
created_at 0
campaign_var_1 0
campaign_var_2 0
products_purchased 0
signup_date 0
user_activity_var_1 0
user_activity_var_2 0
user_activity_var_3 0
user_activity_var_4 0
user_activity_var_5 0
user_activity_var_6 0
user_activity_var_7 0
user_activity_var_8 0
user_activity_var_9 0
user_activity_var_10 0
user_activity_var_11 0
user_activity_var_12 0
buy 0
dtype: int64
```

```
In [16]: df.head()
```

```
Out[16]:
```

	id	created_at	campaign_var_1	campaign_var_2	products_purchased	signup_date	user_activity_var_1	user_activity_var_2	user_activity_v
0	1	2021-01-01	1	2	2.0	2020-09-24	0	0	
1	2	2021-01-01	2	1	2.0	2020-09-19	1	0	
2	3	2021-01-01	9	3	3.0	2021-08-11	1	0	
3	4	2021-01-01	6	7	2.0	2017-10-04	0	0	
5	6	2021-01-01	3	4	3.0	2019-07-02	0	0	

```
In [17]: df.drop(['id'],axis=1,inplace =True)
```

```
In [18]: df.drop(['created_at'],axis=1,inplace=True)
```

```
In [19]: from sklearn.preprocessing import LabelEncoder
label_encoder=LabelEncoder()
```

```
In [20]: cat_cols= df.select_dtypes(include=['object']).columns
```

```
In [21]: df.head()
```

```
Out[21]:
```

	campaign_var_1	campaign_var_2	products_purchased	signup_date	user_activity_var_1	user_activity_var_2	user_activity_var_3	user_activ
0	1	2	2.0	2020-09-24	0	0	0	
1	2	1	2.0	2020-09-19	1	0	1	
2	9	3	3.0	2021-08-11	1	0	0	
3	6	7	2.0	2017-10-04	0	0	0	
5	3	4	3.0	2019-07-02	0	0	0	

```
In [22]: cat_cols
```

```
Out[22]: Index(['signup_date'], dtype='object')
```

```
In [23]: en =LabelEncoder()
for i in cat_cols:
    df[i]=en.fit_transform(df[i])
```

```
In [24]: df.head()
```

```
Out[24]:
```

	campaign_var_1	campaign_var_2	products_purchased	signup_date	user_activity_var_1	user_activity_var_2	user_activity_var_3	user_activ
0	1	2	2.0	1176	0	0	0	
1	2	1	2.0	1171	1	0	1	
2	9	3	3.0	1497	1	0	0	
3	6	7	2.0	161	0	0	0	
5	3	4	3.0	726	0	0	0	

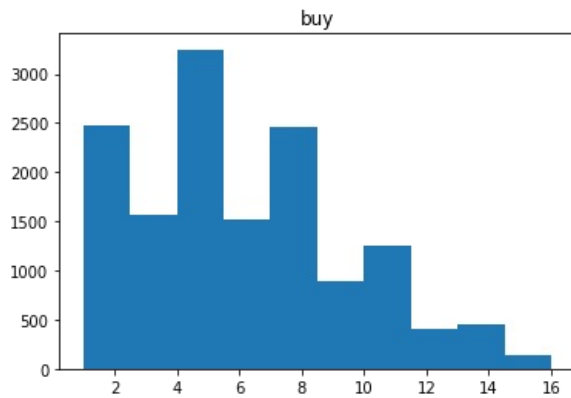
```
In [25]: df['products_purchased'].value_counts()
```

```
Out[25]: 2.0    6971
          3.0    3953
          1.0    2902
          4.0     572
          Name: products_purchased, dtype: int64
```

```
In [26]: df['campaign_var_1'].value_counts()
```

```
Out[26]: 2      1648
          5      1626
          4      1623
          3      1564
          6      1524
          7      1305
          8      1153
          9       884
          1       828
          10       720
          11       538
          12       398
          13       297
          14       155
          15        99
          16        36
          Name: campaign_var_1, dtype: int64
```

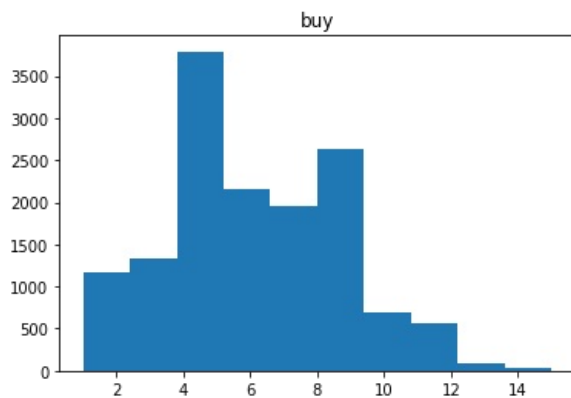
```
In [27]: plt.hist(x='campaign_var_1',data=df)
          plt.title("buy")
          plt.show()
```



```
In [28]: df['campaign_var_2'].value_counts()
```

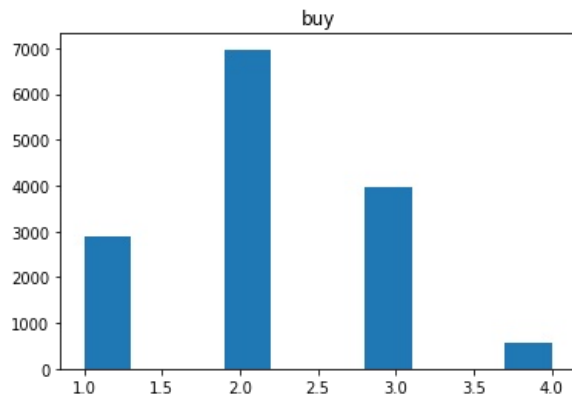
```
Out[28]: 6      2164
          5      2092
          7      1949
          4      1699
          8      1527
          3      1338
          9      1103
          2       874
          10      685
          11      362
          1       292
          12      193
          13       89
          14       29
          15        2
          Name: campaign_var_2, dtype: int64
```

```
In [29]: plt.hist(x = "campaign_var_2",data=df)
          plt.title("buy")
          plt.show()
```



```
In [30]: plt.hist(x = "products_purchased",data=df)
```

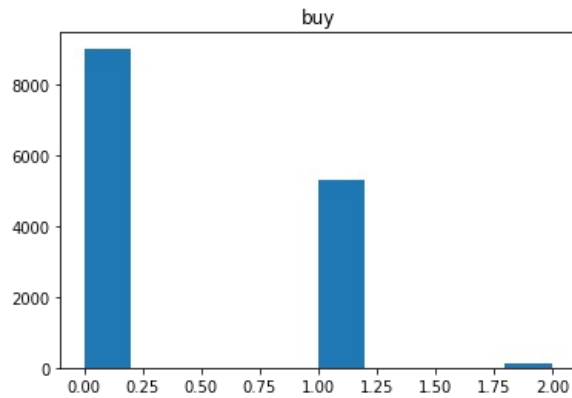
```
plt.title("buy")
plt.show()
```



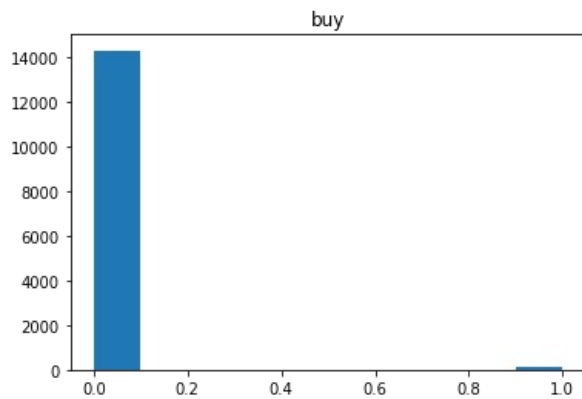
```
In [31]: df['user_activity_var_1'].value_counts()
```

```
Out[31]: 0    8997
         1    5282
         2     119
         Name: user_activity_var_1, dtype: int64
```

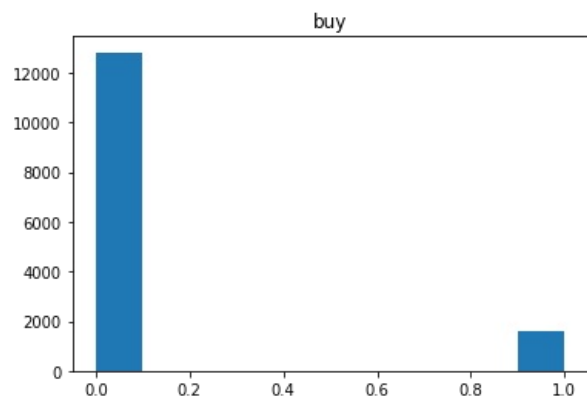
```
In [32]: plt.hist(x = "user_activity_var_1",data=df)
plt.title("buy")
plt.show()
```



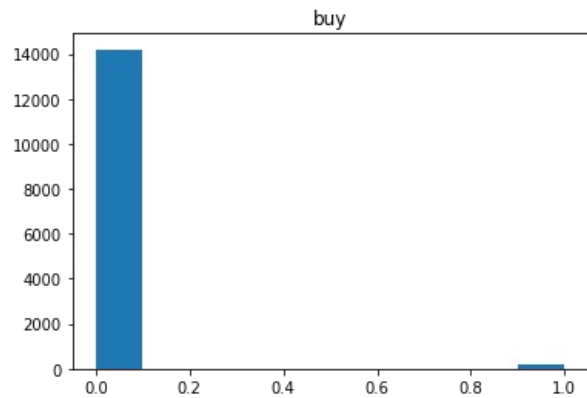
```
In [33]: plt.hist(x = "user_activity_var_2",data=df)
plt.title("buy")
plt.show()
```



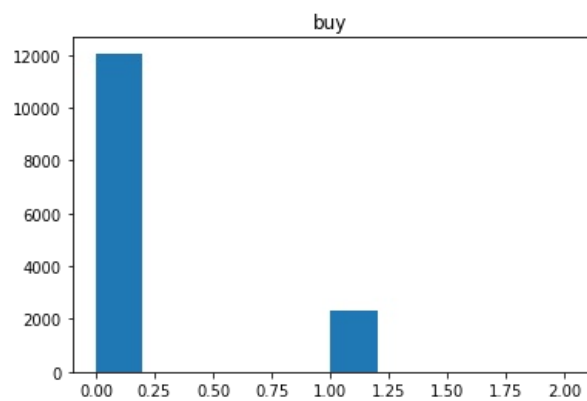
```
In [34]: plt.hist(x = "user_activity_var_3",data=df)
plt.title("buy")
plt.show()
```



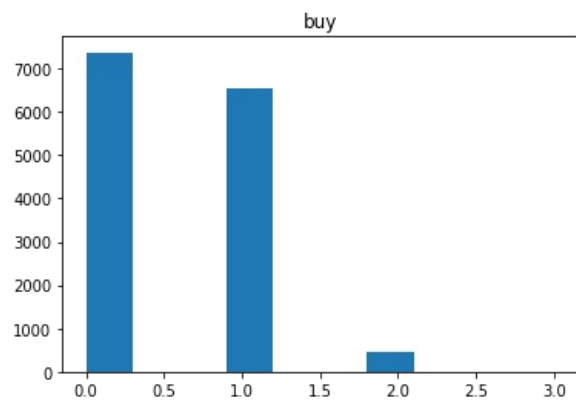
```
In [35]: plt.hist(x = "user_activity_var_4",data=df)
plt.title("buy")
plt.show()
```



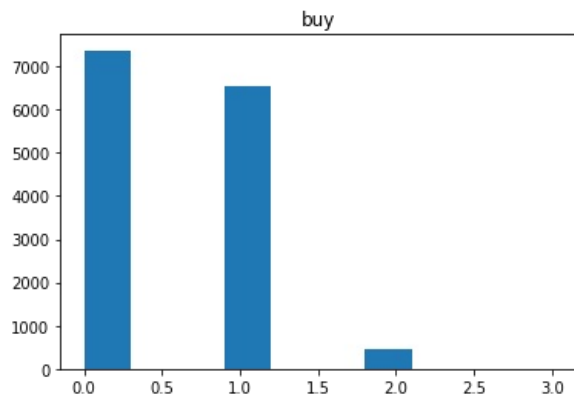
```
In [36]: plt.hist(x = "user_activity_var_5",data=df)
plt.title("buy")
plt.show()
```



```
In [37]: plt.hist(x = "user_activity_var_6",data=df)
plt.title("buy")
plt.show()
```



```
In [38]: plt.hist(x = "user_activity_var_6",data=df)
plt.title("buy")
plt.show()
```



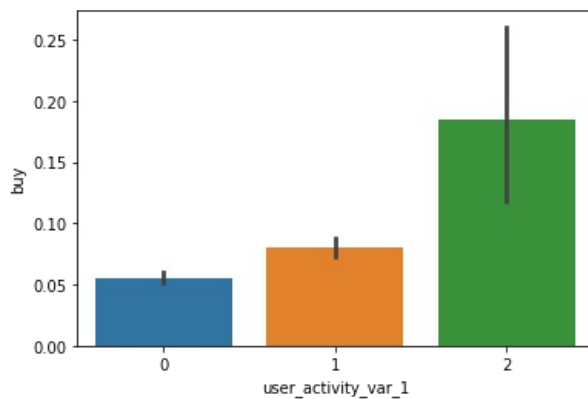
```
In [39]: import seaborn as sns
```

```
In [40]: sns.barplot(df['user_activity_var_1'],df['buy'])
```

C:\Users\sajid\.conda\envs\sajid\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

```
Out[40]: <AxesSubplot:xlabel='user_activity_var_1', ylabel='buy'>
```

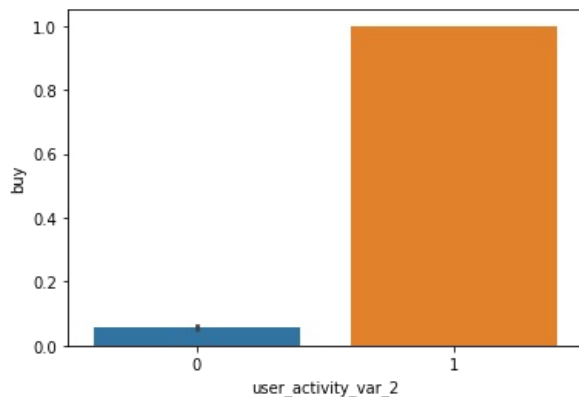


```
In [41]: sns.barplot(df['user_activity_var_2'],df['buy'])
```

C:\Users\sajid\.conda\envs\sajid\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

```
Out[41]: <AxesSubplot:xlabel='user_activity_var_2', ylabel='buy'>
```

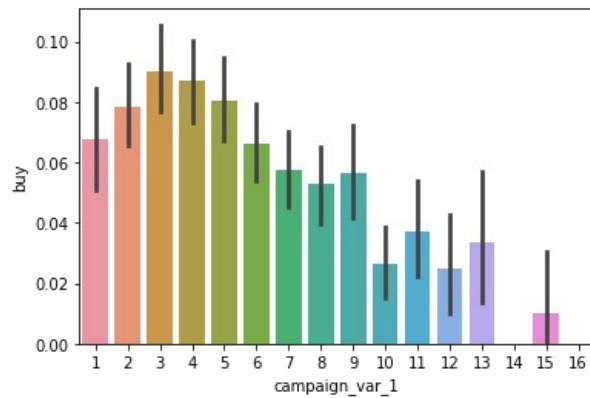


```
In [42]: sns.barplot(df['campaign_var_1'],df['buy'])
```

C:\Users\sajid\.conda\envs\sajid\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

warnings.warn(

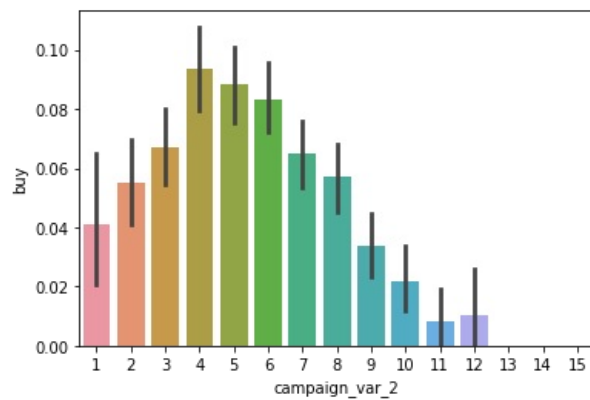
```
Out[42]: <AxesSubplot:xlabel='campaign_var_1', ylabel='buy'>
```



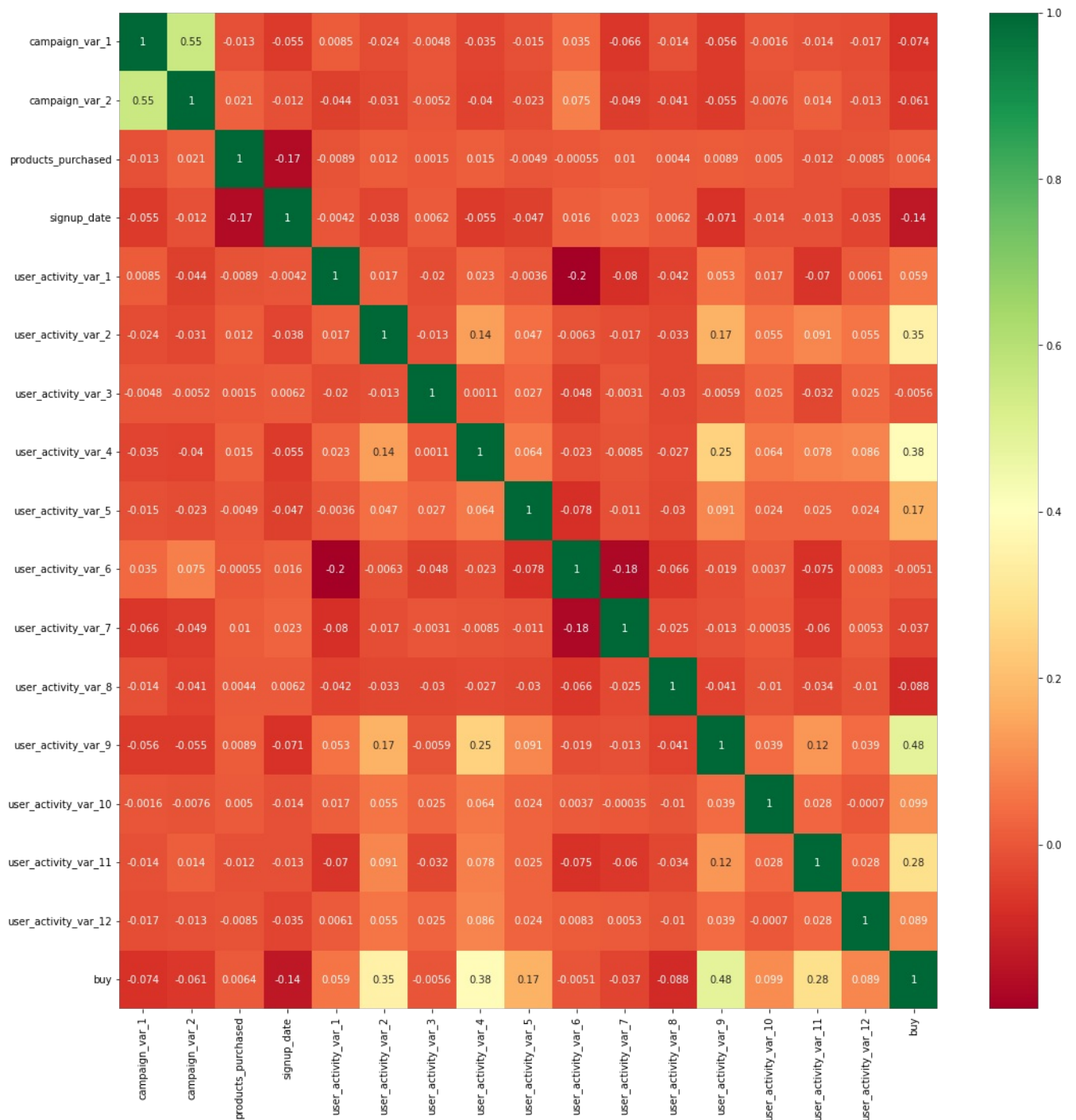
In [43]: `sns.barplot(df['campaign_var_2'],df['buy'])`

C:\Users\sajid\.conda\envs\sajid\lib\site-packages\seaborn\decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
warnings.warn(

Out[43]: `<AxesSubplot:xlabel='campaign_var_2', ylabel='buy'>`



In [44]: `plt.figure(figsize=(18,18))
sns.heatmap(df.corr(), annot = True, cmap = "RdYlGn")
plt.show()`



```
In [45]: X = df.drop('buy',axis=1)
y= df['buy']
```

```
In [46]: X
```

Out[46]:

	campaign_var_1	campaign_var_2	products_purchased	signup_date	user_activity_var_1	user_activity_var_2	user_activity_var_3	user_
0	1	2	2.0	1176	0	0	0	
1	2	1	2.0	1171	1	0	1	
2	9	3	3.0	1497	1	0	0	
3	6	7	2.0	161	0	0	0	
5	3	4	3.0	726	0	0	0	
...	
39141	5	7	1.0	1014	0	0	0	
39145	14	10	3.0	706	0	0	0	
39149	1	5	3.0	1414	0	0	0	
39154	7	11	3.0	1216	0	0	0	
39156	11	11	2.0	171	1	0	0	

14398 rows × 16 columns

```
In [47]: v
```

```
Out[47]: 0      0
1      0
2      0
3      0
5      0
..
39141   0
39145   0
39149   0
39154   0
39156   0
Name: buy, Length: 14398, dtype: int64
```

```
In [48]: from sklearn.model_selection import train_test_split
```

```
In [49]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=42)
```

```
In [50]: X_train
```

```
Out[50]:
```

	campaign_var_1	campaign_var_2	products_purchased	signup_date	user_activity_var_1	user_activity_var_2	user_activity_var_3	user_
33957	10	9	2.0	1532	1	0	0	
22073	5	7	3.0	602	1	0	0	
9811	4	2	3.0	1395	0	0	0	
32178	9	10	2.0	587	1	0	0	
36757	6	8	2.0	1613	1	0	0	
...	
12264	3	2	2.0	1260	0	0	0	
36111	7	2	1.0	1521	1	0	0	
12763	8	9	1.0	1388	0	0	0	
1788	1	3	2.0	1137	0	0	0	
17683	7	6	3.0	1473	0	0	0	

11518 rows × 16 columns



```
In [51]: X_train.shape
```

```
Out[51]: (11518, 16)
```

```
In [52]: y_train
```

```
Out[52]: 33957   0
22073   0
9811    0
32178   0
36757   0
..
12264   0
36111   0
12763   0
1788    0
17683   0
Name: buy, Length: 11518, dtype: int64
```

```
In [53]: y_train.shape
```

```
Out[53]: (11518,)
```

```
In [54]: X_test
```

	campaign_var_1	campaign_var_2	products_purchased	signup_date	user_activity_var_1	user_activity_var_2	user_activity_var_3	user_
6792	5	8	3.0	881	0	0	0	
17438	8	2	1.0	1429	0	0	1	
7967	9	8	2.0	865	0	0	0	
16116	2	4	1.0	545	0	0	0	
32696	10	8	3.0	573	1	0	0	
...	
19587	4	8	1.0	1303	0	0	0	
24516	1	1	1.0	309	1	1	0	
31446	2	3	3.0	1561	1	0	0	
27018	9	7	3.0	1305	0	0	0	
12756	2	8	2.0	1436	0	0	0	

2880 rows × 16 columns

```
In [55]: y_test
```

```
Out[55]: 6792    0
17438    0
7967     1
16116    0
32696    0
..
19587    0
24516    1
31446    0
27018    0
12756    0
Name: buy, Length: 2880, dtype: int64
```

```
In [56]: from sklearn.linear_model import LogisticRegression
```

```
In [57]: model = LogisticRegression()
```

```
In [58]: model.fit(X_train,y_train)
```

```
C:\Users\sajid\.conda\envs\sajid\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
  https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
  https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
LogisticRegression()
```

```
Out[58]: LogisticRegression()
```

```
In [59]: y_predict = model.predict(X_test)
y_predict
```

```
Out[59]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
In [60]: from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
In [61]: accuracy_score(y_test,y_predict)
```

```
Out[61]: 0.9631944444444445
```

```
In [62]: print(classification_report(y_test,y_predict))
```

```

              precision    recall  f1-score   support

     0       0.97      0.99      0.98        2710
     1       0.85      0.46      0.60         170

 accuracy
macro avg       0.91      0.73      0.79        2880
weighted avg       0.96      0.96      0.96        2880
```

```
In [63]: confusion_matrix(y_test,y_predict)
```

```
Out[63]: array([[2696, 14],
               [ 92, 78]], dtype=int64)
```

```
In [64]: print("Training Accuracy:", model.score(X_train,y_train))
print('Testing Accuracy:',model.score(X_test,y_test))
```

```
Training Accuracy: 0.9583260982809515
Testing Accuracy: 0.9631944444444445
```

```
In [65]: from sklearn.tree import DecisionTreeClassifier
```

```
In [66]: model = DecisionTreeClassifier()
model.fit(X_train,y_train)
y_predict = model.predict(X_test)
accuracy_score(y_test,y_predict)
```

```
Out[66]: 0.9451388888888889
```

```
In [67]: DecisionTreeClassifier()
```

```
Out[67]: DecisionTreeClassifier()
```

```
In [68]: y_predict = model.predict(X_test)
y_predict
```

```
Out[68]: array([0, 0, 1, ..., 0, 0, 0], dtype=int64)
```

```
In [69]: y_test
```

```
Out[69]: 6792      0
17438      0
7967       1
16116      0
32696      0
..
19587      0
24516      1
31446      0
27018      0
12756      0
Name: buy, Length: 2880, dtype: int64
```

```
In [70]: accuracy_score(y_test,y_predict)
```

```
Out[70]: 0.9451388888888889
```

```
In [71]: print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
0	0.98	0.96	0.97	2710
1	0.53	0.65	0.58	170
accuracy			0.95	2880
macro avg	0.75	0.81	0.78	2880
weighted avg	0.95	0.95	0.95	2880

```
In [72]: confusion_matrix(y_test,y_test)
```

```
Out[72]: array([[2710,    0],
               [    0,   170]], dtype=int64)
```

```
In [73]: print("Training Accuracy:", model.score(X_train,y_train))
print('Testing Accuracy:',model.score(X_test,y_test))
```

```
Training Accuracy: 1.0
Testing Accuracy: 0.9451388888888889
```

```
In [74]: import matplotlib.pyplot as plt
```

```
In [75]: from xgboost import XGBClassifier
```

```
In [76]: model = XGBClassifier()
model.fit(X_train,y_train)
y_predict = model.predict(X_test)
accuracy_score(y_test,y_predict)
```

```
Out[76]: 0.9739583333333334
```

```
In [77]: XGBClassifier()
```

```

Out[77]: XGBClassifier(base_score=None, booster=None, callbacks=None,
                    colsample_bylevel=None, colsample_bynode=None,
                    colsample_bytree=None, early_stopping_rounds=None,
                    enable_categorical=False, eval_metric=None, gamma=None,
                    gpu_id=None, grow_policy=None, importance_type=None,
                    interaction_constraints=None, learning_rate=None, max_bin=None,
                    max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
                    max_leaves=None, min_child_weight=None, missing=nan,
                    monotone_constraints=None, n_estimators=100, n_jobs=None,
                    num_parallel_tree=None, predictor=None, random_state=None,
                    reg_alpha=None, reg_lambda=None, ...)

In [78]: y_predict = model.predict(X_test)
y_predict

Out[78]: array([0, 0, 1, ..., 0, 0, 0])

In [79]: accuracy_score(y_test,y_predict)

Out[79]: 0.9739583333333334

In [80]: print(classification_report(y_test,y_predict))

              precision    recall  f1-score   support

         0       0.98        1.00        0.99        2710
         1       0.89        0.64        0.74         170

   accuracy          0.97          0.97          0.97          2880
  macro avg          0.94          0.82          0.86          2880
 weighted avg          0.97          0.97          0.97          2880

In [81]: confusion_matrix(y_test,y_test)

Out[81]: array([[2710,    0],
                [    0,   170]], dtype=int64)

In [82]: print("Training Accuracy:", model.score(X_train,y_train))
print('Testing Accuracy:',model.score(X_test,y_test))

Training Accuracy: 0.9781212015974996
Testing Accuracy: 0.9739583333333334

In [83]: from sklearn.ensemble import RandomForestClassifier

In [84]: model = RandomForestClassifier()
model.fit(X_train,y_train)
y_predict = model.predict(X_test)
accuracy_score(y_test,y_predict)

Out[84]: 0.9725694444444445

In [85]: model=RandomForestClassifier(n_estimators=80,max_depth=5,criterion='gini')
model.fit(X_train,y_train)
y_predict = model.predict(X_test)
accuracy_score(y_test,y_predict)

Out[85]: 0.9736111111111111

In [86]: print(classification_report(y_test,y_predict))

              precision    recall  f1-score   support

         0       0.97        1.00        0.99        2710
         1       0.94        0.59        0.72         170

   accuracy          0.97          0.97          0.97          2880
  macro avg          0.96          0.79          0.86          2880
 weighted avg          0.97          0.97          0.97          2880

In [87]: confusion_matrix(y_test,y_test)

Out[87]: array([[2710,    0],
                [    0,   170]], dtype=int64)

In [88]: print("Training Accuracy:", model.score(X_train,y_train))
print('Testing Accuracy:',model.score(X_test,y_test))

Training Accuracy: 0.9668345198819239
Testing Accuracy: 0.9736111111111111

In [91]: input_data = (1,2,2.0,1176,0,0,0,0,0,1,1,0,0,0,0)

input_data_as_numpy_array = np.asarray(input_data)

input_data_resaped = input_data_as_numpy_array.reshape(1,-1)

```

```
prediction = model.predict(input_data_resaped)
print(prediction)
```

```
[0]
```

```
C:\Users\sajid\conda\envs\sajid\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
warnings.warn(
```

```
In [92]: if (prediction[0])==0:
          print('buy')
        else:
          print('not buy')
```

```
buy
```

```
In [93]: import pickle
```

```
In [94]: filename = 'trained_model.sav'
pickle.dump(model,open(filename,'wb'))
```

```
In [95]: loaded_model = pickle.load(open('trained_model.sav','rb'))
```

```
In [98]: input_data = (1,2,2.0,1176,0,0,0,0,0,0,1,1,0,0,0,0)

input_data_as_numpy_array = np.asarray(input_data)

input_data_resaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_resaped)
print(prediction)

if (prediction[0])==0:
    print('buy')
else:
    print('not buy')
```

```
[0]
```

```
buy
```

```
C:\Users\sajid\conda\envs\sajid\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
warnings.warn(
```

```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js