In [20]:
```
import pandas as pd
import numpy as np
```

In [21]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn import tree
from sklearn import model_selection
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import Lasso
from sklearn.linear_model import LogisticRegression
import seaborn as sns
```

Importing dataset

In [22]:
```
df = pd.read_csv("D:\Celina Python\marketing_campaign.csv",header=0,sep=';')
df.head()
```

Out[22]:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWine
0	5524	1957	Graduation	Single	58138.0	0	0	2012-09-04	58	63
1	2174	1954	Graduation	Single	46344.0	1	1	2014-03-08	38	1
2	4141	1965	Graduation	Together	71613.0	0	0	2013-08-21	26	42
3	6182	1984	Graduation	Together	26646.0	1	0	2014-02-10	26	1
4	5324	1981	PhD	Married	58293.0	1	0	2014-01-19	94	17

5 rows × 29 columns

Investigating missing values & duplicates in the data.

In [23]:
```
df = df.dropna()
#Select duplicate rows except first occurrence based on all columns
duplicateRowsDF = df[df.duplicated()]
print("Duplicate Rows except first occurrence based on all columns are :")
print(duplicateRowsDF)
```

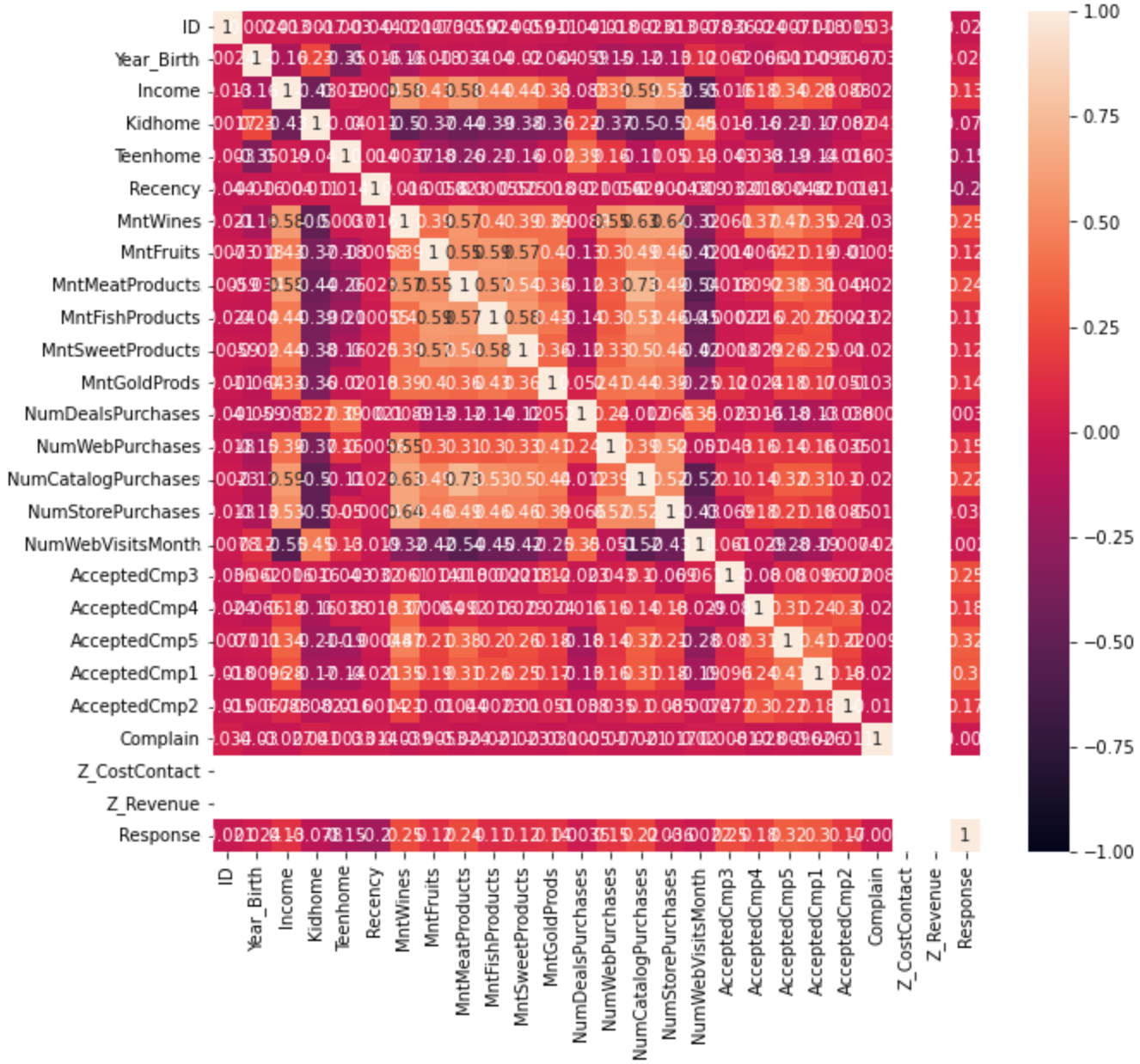
```
Duplicate Rows except first occurrence based on all columns are :
Empty DataFrame
Columns: [ID, Year_Birth, Education, Marital_Status, Income, Kidhome, Teenhome, Dt_Custom
er, Recency, MntWines, MntFruits, MntMeatProducts, MntFishProducts, MntSweetProducts, Mnt
GoldProds, NumDealsPurchases, NumWebPurchases, NumCatalogPurchases, NumStorePurchases, Nu
mWebVisitsMonth, AcceptedCmp3, AcceptedCmp4, AcceptedCmp5, AcceptedCmp1, AcceptedCmp2, Co
mplain, Z_CostContact, Z_Revenue, Response]
Index: []

[0 rows x 29 columns]
```

Heatmap of correlation

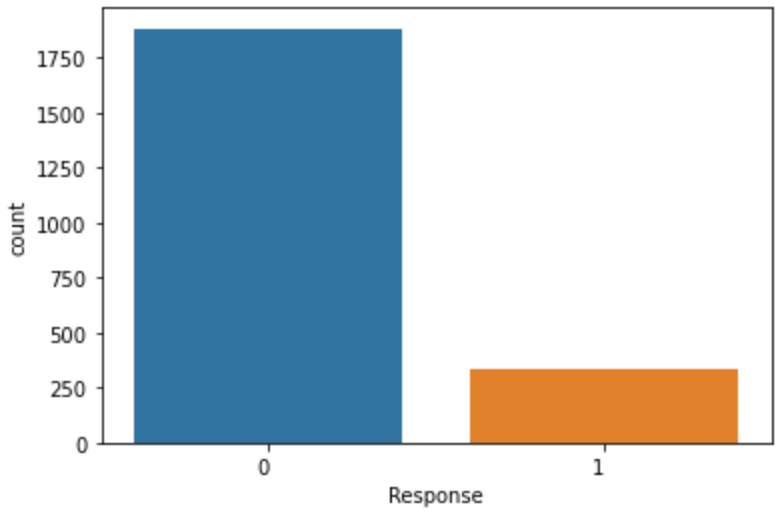
In [24]:

```
from matplotlib import pyplot
fig = pyplot.figure(figsize=(10, 9))
sns.heatmap(df.corr(),vmin=-1, vmax=1, annot=True);
```



In [25]:
```
ax= sns.countplot(x="Response", data=df)
ax.set_xticklabels(ax.get_xticklabels(), ha="right")
```

Out[25]:
```
[Text(0, 0, '0'), Text(1, 0, '1')]
```



Defining dependent and independent variables

In [26]:
```
# Label/Response set
y = df['Response']

# Drop the labels and store the features
X = df[["Income","Kidhome"]]
```

Spliting the dataset in to training and testing sets

In [27]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
from sklearn.model_selection import train_test_split
print (X_train.shape,X_test.shape,y_train.shape, y_test.shape)
```

```
(1772, 2) (444, 2) (1772,) (444,)
```

Logistic regression model

In [28]:
```
clf_LR = LogisticRegression()
clf_LR.fit(X_train,y_train)
y_test_pred = clf_LR.predict(X_test)
from sklearn.metrics import accuracy_score, confusion_matrix
confusion_matrix(y_test, y_test_pred)
accuracy_score(y_test, y_test_pred)
```

Out[28]:
```
0.8355855855855856
```

In [29]:
```
accuracy_score(y_test, y_test_pred)
```

Out[29]:
```
0.8355855855855856
```

KNN algorithm

In [30]:
```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_s= scaler.transform(X_train)
scaler = preprocessing.StandardScaler().fit(X_test)
X_test_s= scaler.transform(X_test)
X_test_s
from sklearn.neighbors import KNeighborsClassifier
clf_knn_1 = KNeighborsClassifier(n_neighbors=2)
clf_knn_1.fit(X_train_s, y_train)
confusion_matrix(y_test, clf_knn_1.predict(X_test_s))
```

Out[30]:
```
array([[346,  25],
       [ 60,  13]], dtype=int64)
```

In [31]:
```
accuracy_score(y_test, clf_knn_1.predict(X_test_s))
```

Out[31]:
```
0.8085585585585585
```

KNN with optimized K

In [32]:
```
from sklearn.model_selection import GridSearchCV
params = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20,30]}
grid_search_cv = GridSearchCV(KNeighborsClassifier(), params)
grid_search_cv.fit(X_train_s, y_train)
grid_search_cv.best_params_
optimised_KNN = grid_search_cv.best_estimator_
y_test_pred = optimised_KNN.predict(X_test_s)
confusion_matrix(y_test, y_test_pred)
```

Out[32]:
```
array([[350,  21],
       [ 58,  15]], dtype=int64)
```

In [33]:
```
accuracy_score(y_test, y_test_pred)
```

Out[33]:
```
0.8220720720720721
```

In [ ]:

In [ ]:

In [ ]: