In [75]:
```python
# import modules and libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV, RepeatedStratifiedKFold
from sklearn.metrics import classification_report, accuracy_score
import xgboost as xgb
from sklearn.ensemble import AdaBoostClassifier
```

In [56]:
```python
# import train dataset
train_data = pd.read_csv('Train.csv')
train_data.head()
```

Out[56]:

	ID	Year_of_Birth	Education_Level	Marital_Status	Disposable_Income	No_of_Kids_in_home	No_of_Teen_in_home	Date_Customer
0	ID_4A9AR6FQ	1955	Graduation	Divorced	77504.4	1	1	22-06-2014
1	ID_X28T3VEK	1958	Graduation	Together	56784.0	0	1	01-08-2013
2	ID_AWXARH57	1962	Graduation	Single	103714.8	0	0	21-11-2013
3	ID_FQVZHE81	1979	2n Cycle	Single	46311.6	1	0	13-09-2013
4	ID_QVLWGPRN	1959	Graduation	Married	87486.0	0	0	21-01-2014

5 rows × 27 columns

In [57]:
```python
# import test dataset
test_data = pd.read_csv('Test.csv')
test_data.head()
```

Out[57]:

	ID	Year_of_Birth	Education_Level	Marital_Status	Disposable_Income	No_of_Kids_in_home	No_of_Teen_in_home	Date_Customer
0	ID_ZPMABNVX	1954	Graduation	Single	48556.8	0	1	11-01-2013
1	ID_WFE91NAA	1961	Graduation	Widow	57499.2	0	1	22-11-2012
2	ID_JV11RBRK	1973	Basic	Married	17025.6	0	0	28-02-2013
3	ID_6B7SVKY9	1970	Graduation	Together	91983.6	0	0	16-08-2013
4	ID_GOVUZ545	1959	Graduation	Together	78235.2	0	2	25-07-2013

5 rows × 26 columns

TRAIN DATA

In [58]:
```python
#get the shape of the train data
train_data.shape
```

Out[58]: (1568, 27)

In [59]:
```python
#get the statistical description of the float columns of the train data
train_data.describe()
```

Out[59]:

	Year_of_Birth	Disposable_Income	No_of_Kids_in_home	No_of_Teen_in_home	Recency	Discounted_Purchases	WebPurchases
count	1568.000000	1552.000000	1568.000000	1568.000000	1568.000000	1568.000000	1568.000000
mean	1970.073342	62381.186598	0.460459	0.497449	55.408801	2.292730	4.001276
std	11.920781	32089.169563	0.540361	0.544151	28.788037	1.937544	2.773748
min	1900.000000	2076.000000	0.000000	0.000000	7.000000	0.000000	0.000000
25%	1960.000000	41612.400000	0.000000	0.000000	31.000000	1.000000	2.000000
50%	1971.000000	60964.200000	0.000000	0.000000	56.000000	2.000000	3.000000
75%	1979.000000	81493.200000	1.000000	1.000000	80.000000	3.000000	6.000000
max	1997.000000	799999.200000	2.000000	2.000000	106.000000	15.000000	27.000000

8 rows × 23 columns

In [60]:
```python
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1568 entries, 0 to 1567
Data columns (total 27 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ID                      1568 non-null   object
 1   Year_of_Birth           1568 non-null   int64
 2   Education_Level         1568 non-null   object
 3   Marital_Status          1568 non-null   object
 4   Disposable_Income       1552 non-null   float64
 5   No_of_Kids_in_home      1568 non-null   int64
 6   No_of_Teen_in_home      1568 non-null   int64
 7   Date_Customer           1568 non-null   object
 8   Recency                 1568 non-null   int64
 9   Discounted_Purchases    1568 non-null   int64
 10  WebPurchases            1568 non-null   int64
 11  CatalogPurchases        1568 non-null   int64
 12  StorePurchases          1568 non-null   int64
 13  Amount_on_Wines         1568 non-null   int64
 14  Amount_on_Fruits        1568 non-null   int64
 15  Amount_on_MeatProducts  1568 non-null   int64
 16  Amount_on_FishProducts  1568 non-null   int64
 17  Amount_on_SweetProducts 1568 non-null   int64
 18  Amount_on_GoldProds     1568 non-null   int64
 19  WebVisitsMonth          1568 non-null   int64
 20  Cmp3Accepted            1568 non-null   int64
 21  Cmp4Accepted            1568 non-null   int64
 22  Cmp5Accepted            1568 non-null   int64
 23  Cmp1Accepted            1568 non-null   int64
 24  Cmp2Accepted            1568 non-null   int64
 25  Any_Complain            1568 non-null   int64
 26  Response                1568 non-null   int64
dtypes: float64(1), int64(22), object(4)
memory usage: 330.9+ KB
```

In [61]:
```python
train_data.isnull().sum()
```

Out[61]:
```
ID                          0
Year_of_Birth               0
Education_Level             0
Marital_Status              0
Disposable_Income          16
No_of_Kids_in_home          0
No_of_Teen_in_home          0
Date_Customer               0
Recency                     0
Discounted_Purchases        0
WebPurchases                0
CatalogPurchases            0
StorePurchases              0
Amount_on_Wines             0
Amount_on_Fruits            0
Amount_on_MeatProducts      0
Amount_on_FishProducts      0
Amount_on_SweetProducts     0
Amount_on_GoldProds         0
WebVisitsMonth              0
Cmp3Accepted                0
Cmp4Accepted                0
Cmp5Accepted                0
Cmp1Accepted                0
Cmp2Accepted                0
Any_Complain                0
Response                    0
dtype: int64
```

In [62]:
```python
# fill the missing values with the mean
train_data['Disposable_Income'] = train_data['Disposable_Income'].fillna(train_data['Disposable_Income'].mean()
```

In [63]:
```python
# change the data type of the Disposable Income column from Object to float
train_data['Disposable_Income'] = pd.to_numeric(train_data['Disposable_Income'], errors='coerce')
```

TEST DATA

In [64]:
```python
#get the shape of the test data
test_data.shape
```

Out[64]: (672, 26)

In [65]:
```python
#get the statistical description of the float columns of the test data
test_data.describe()
```

Out[65]:

	Year_of_Birth	Disposable_Income	No_of_Kids_in_home	No_of_Teen_in_home	Recency	Discounted_Purchases	WebPurchases
count	672.000000	664.000000	672.000000	672.000000	672.000000	672.000000	672.000000
mean	1969.181548	63434.170482	0.406250	0.526786	57.744048	2.400298	4.279762
std	12.116416	25276.585476	0.532259	0.545293	29.321893	1.919125	2.782585
min	1894.000000	5313.600000	0.000000	0.000000	7.000000	0.000000	0.000000
25%	1960.000000	44219.700000	0.000000	0.000000	33.000000	1.000000	2.000000
50%	1970.000000	64007.400000	0.000000	1.000000	58.000000	2.000000	4.000000
75%	1978.000000	83466.300000	1.000000	1.000000	84.000000	3.000000	6.000000
max	1997.000000	194876.400000	2.000000	2.000000	106.000000	15.000000	25.000000

8 rows × 22 columns

In [66]:
```python
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 672 entries, 0 to 671
Data columns (total 26 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ID                      672 non-null    object
 1   Year_of_Birth           672 non-null    int64
 2   Education_Level         672 non-null    object
 3   Marital_Status          672 non-null    object
 4   Disposable_Income       664 non-null    float64
 5   No_of_Kids_in_home      672 non-null    int64
 6   No_of_Teen_in_home      672 non-null    int64
 7   Date_Customer           672 non-null    object
 8   Recency                 672 non-null    int64
 9   Discounted_Purchases    672 non-null    int64
 10  WebPurchases            672 non-null    int64
 11  CatalogPurchases        672 non-null    int64
 12  StorePurchases          672 non-null    int64
 13  Amount_on_Wines         672 non-null    int64
 14  Amount_on_Fruits        672 non-null    int64
 15  Amount_on_MeatProducts  672 non-null    int64
 16  Amount_on_FishProducts  672 non-null    int64
 17  Amount_on_SweetProducts 672 non-null    int64
 18  Amount_on_GoldProds     672 non-null    int64
 19  WebVisitsMonth          672 non-null    int64
 20  Cmp3Accepted            672 non-null    int64
 21  Cmp4Accepted            672 non-null    int64
 22  Cmp5Accepted            672 non-null    int64
 23  Cmp1Accepted            672 non-null    int64
 24  Cmp2Accepted            672 non-null    int64
 25  Any_Complain            672 non-null    int64
dtypes: float64(1), int64(21), object(4)
memory usage: 136.6+ KB
```

In [67]:
```python
test_data.isnull().sum()
```

Out[67]:
```
ID                         0
Year_of_Birth              0
Education_Level            0
Marital_Status             0
Disposable_Income          8
No_of_Kids_in_home         0
No_of_Teen_in_home         0
Date_Customer              0
Recency                    0
Discounted_Purchases       0
WebPurchases               0
CatalogPurchases           0
StorePurchases             0
Amount_on_Wines            0
Amount_on_Fruits           0
Amount_on_MeatProducts     0
Amount_on_FishProducts     0
Amount_on_SweetProducts    0
Amount_on_GoldProds        0
WebVisitsMonth             0
Cmp3Accepted               0
Cmp4Accepted               0
Cmp5Accepted               0
Cmp1Accepted               0
Cmp2Accepted               0
Any_Complain               0
dtype: int64
```

In [68]:
```python
# fill the missing values with the mean
test_data['Disposable_Income'] = test_data['Disposable_Income'].fillna(test_data['Disposable_Income'].mean())
```

In [69]:
```python
# change the data type of the Disposable Income column from Object to float
test_data['Disposable_Income'] = pd.to_numeric(test_data['Disposable_Income'], errors='coerce')
```

In [70]:
```python
X_train = train_data.drop(['ID', 'Date_Customer','Response'], axis=1)
y_train = train_data['Response']
X_test = test_data.drop(['ID', 'Date_Customer'], axis=1)
```

In [71]:
```python
X_train.shape, X_test.shape, y_train.shape
```

Out[71]: ((1568, 24), (672, 24), (1568,))

In [72]:
```python
X_train = X_train.replace({'Marital_Status' :
            {'YOLO': 'Single', 'Alone' : 'Single'}
             })
```

In [73]:
```python
categ = ["Education_Level", 'Marital_Status']
```

Get Dummies

In [76]:
```python
le = LabelEncoder()
X_train[categ] = X_train[categ].apply(le.fit_transform)
X_test[categ] = X_test[categ].apply(le.fit_transform)
```

In [77]:
```python
# feature scale the X_train and X_test values

sscaler = StandardScaler().fit(X_train)

#transform the training data
X_train = sscaler.transform(X_train)

# transform the testing data
X_test = sscaler.transform(X_test)

print(X_train)
print('\n')
print(X_test)
```

```
[[-1.26486265 -0.34284993 -1.63410463 ... -0.262389   -0.11366572
  -0.09143374]
 [-1.013121   -0.34284993  1.20341038 ... -0.262389   -0.11366572
  -0.09143374]
 [-0.67746547 -0.34284993  0.25757205 ...  3.81113533 -0.11366572
  -0.09143374]
 ...
 [-1.5166043   0.54719757 -0.68826629 ... -0.262389   -0.11366572
  -0.09143374]
 [-0.5096377   1.43724508  0.25757205 ... -0.262389   -0.11366572
  -0.09143374]
 [-0.0061544  -0.34284993  1.20341038 ... -0.262389   -0.11366572
  -0.09143374]]


[[-1.34877654 -0.34284993  0.25757205 ... -0.262389   -0.11366572
  -0.09143374]
 [-0.76137935 -0.34284993  2.14924872 ... -0.262389   -0.11366572
  -0.09143374]
 [ 0.24558725 -1.23289744 -0.68826629 ... -0.262389   -0.11366572
  -0.09143374]
 ...
 [-1.5166043   1.43724508 -0.68826629 ... -0.262389   -0.11366572
  -0.09143374]
 [-0.84529324 -0.34284993  1.20341038 ... -0.262389   -0.11366572
  10.93688185]
 [-0.76137935  0.54719757 -0.68826629 ... -0.262389   -0.11366572
  -0.09143374]]
```

In [78]:
```python
logregression = LogisticRegression()
svc = SVC()
knn = KNeighborsClassifier()
random_forest = RandomForestClassifier()
decision_tree = DecisionTreeClassifier()
xgboost =xgb.XGBClassifier()
adaboost = AdaBoostClassifier(random_state=1)
```

In [79]:
```python
logregression.fit(X_train, y_train)
svc.fit(X_train, y_train)
knn.fit(X_train, y_train)
random_forest.fit(X_train, y_train)
decision_tree.fit(X_train,y_train)
xgboost.fit(X_train,y_train)
adaboost.fit(X_train,y_train)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label encoder in XGBC
lassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) P
ass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as int
egers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[21:07:08] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with
the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like
to restore the old behavior.
```

Out[79]: AdaBoostClassifier(random_state=1)

In [80]:
```python
print("Accuracy of train(Logistic Regression): {}".format(round(logregression.score(X_train,y_train) * 100, 4))
print()
print("Accuracy of train(SVM): {}".format(round(svc.score(X_train,y_train) * 100, 4)))
print()
print("Accuracy of train(KNN): {}".format(round(knn.score(X_train,y_train) * 100, 4)))
print()
print("Accuracy of train(Random Forrest): {}".format(round(random_forest.score(X_train,y_train) * 100, 4)))
print()
print("Accuracy of train(Decision Tree): {}".format(round(decision_tree.score(X_train,y_train) * 100, 4)))
print()
print("Accuracy of train(XG Boost): {}".format(round(xgboost.score(X_train,y_train) * 100, 4)))
print()
print("Accuracy of train(Ada Boost): {}".format(round(adaboost.score(X_train,y_train) * 100, 4)))
```

```
Accuracy of train(Logistic Regression): 88.7117

Accuracy of train(SVM): 91.2628

Accuracy of train(KNN): 90.2423

Accuracy of train(Random Forrest): 99.426

Accuracy of train(Decision Tree): 99.426

Accuracy of train(XG Boost): 99.426

Accuracy of train(Ada Boost): 90.5612
```

In [81]:
```python
xgboost =xgb.XGBClassifier()
```

In [83]:
```python
# define grid search
param_grid = {
    'clf__n_estimators': [50, 100, 150, 200],
    'clf__learning_rate': [0.01, 0.1, 0.2, 0.3],
    'clf__max_depth': range(3, 10),
    'clf__colsample_bytree': [i/10.0 for i in range(1, 3)],
    'clf__gamma': [i/10.0 for i in range(3)],
    'fs__k': [10]
           }
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
CV_rfc = GridSearchCV(estimator=xgboost, param_grid=param_grid, n_jobs=-1, cv= cv, scoring='accuracy',error_sc
CV_rfc.fit(X_train, y_train)

# summarize results
print("Best: %f using %s" % (CV_rfc.best_score_, CV_rfc.best_params_))
```

```
C:\Users\Dell\anaconda3\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label encoder in XGBC
lassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) P
ass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as int
egers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
[22:40:43] WARNING: ..\src\learner.cc:541:
Parameters: { clf__colsample_bytree, clf__gamma, clf__learning_rate, clf__max_depth, clf__n_estimators, fs__k }
might not be used.

  This may not be accurate due to some parameters are only used in language bindings but
  passed down to XGBoost core.  Or some parameters are not used but slip through this
  verification. Please open an issue if you find above cases.


[22:40:43] WARNING: ..\src\learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with
the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like
to restore the old behavior.
Best: 0.879254 using {'clf__colsample_bytree': 0.1, 'clf__gamma': 0.0, 'clf__learning_rate': 0.01, 'clf__max_de
pth': 3, 'clf__n_estimators': 50, 'fs__k': 10}
```

In [84]:
```python
#Predict with the best parameters
y_pred = CV_rfc.predict(X_test)
```

In [85]:
```python
submission = pd.DataFrame(
    {
        'ID': test_data["ID"],
        'Response': y_pred,
    }
)
submission.to_csv(f"submission.csv", index=False)
```

In [ ]: