ML Data Cleaning and Feature Selection

In this assignment, you will use a dataset for predictive learning and check the quality of the data and determine which features are important.

Answer the following questions:

- What are the data types? (Only numeric and categorical)
- Are there missing values?
- What are the likely distributions of the numeric variables?
- Which independent variables are useful to predict a target (dependent variable)? (Use at least three methods)
- Which independent variables have missing data? How much?
- Do the training and test sets have the same data?
- In the predictor variables independent of all the other predictor variables?
- Which predictor variables are the most important?
- Do the ranges of the predictor variables make sense?
- What are the distributions of the predictor variables?
- Remove outliers and keep outliers (does if have an effect of the final predictive model)?
- Remove 1%, 5%, and 10% of your data randomly and impute the values back using at least 3 imputation methods. How well did the methods recover the missing values? That is remove some data, check the % error on residuals for numeric data and check for bias and variance of the error.
- For categorical data, calculate the accuracy and a confusion matrix.

Abstract

Market Sales data contains customer specific information about a customers details like age, income and shopping habits like amount spent, website visits etc. We aim to find if a customer will repond positively to a marketting campaign or not. Various techniques are used to explore data like correlation heatmaps and boxplots, Q-Q plots etc. Finally a logistic regression model is built to predict customer response. It appears that only some columns were of significance while predicting dependent variable. Different impute methods and their effectiveness is explored

About Dataset - Marketing Campaign

Context A response model can provide a significant boost to the efficiency of a marketing campaign by increasing responses or reducing expenses. The objective is to predict who will respond to an offer for a product or service

Content

1. AcceptedCmp1 - 1 if customer accepted the offer in the 1st campaign, 0 otherwise
2. AcceptedCmp2 - 1 if customer accepted the offer in the 2nd campaign, 0 otherwise
3. AcceptedCmp3 - 1 if customer accepted the offer in the 3rd campaign, 0 otherwise
4. AcceptedCmp4 - 1 if customer accepted the offer in the 4th campaign, 0 otherwise
5. AcceptedCmp5 - 1 if customer accepted the offer in the 5th campaign, 0 otherwise
6. Response (target) - 1 if customer accepted the offer in the last campaign, 0 otherwise
7. Complain - 1 if customer complained in the last 2 years
8. DtCustomer - date of customer’s enrolment with the company
9. Education - customer’s level of education
10. Marital - customer’s marital status
11. Kidhome - number of small children in customer’s household
12. Teenhome - number of teenagers in customer’s household
13. Income - customer’s yearly household income
14. MntFishProducts - amount spent on fish products in the last 2 years
15. MntMeatProducts - amount spent on meat products in the last 2 years
16. MntFruits - amount spent on fruits products in the last 2 years
17. MntSweetProducts - amount spent on sweet products in the last 2 years
18. MntWines - amount spent on wine products in the last 2 years
19. MntGoldProds - amount spent on gold products in the last 2 years
20. NumDealsPurchases - number of purchases made with discount

21. NumCatalogPurchases - number of purchases made using catalogue
22. NumStorePurchases - number of purchases made directly in stores
23. NumWebPurchases - number of purchases made through company's web site
24. NumWebVisitsMonth - number of visits to company's web site in the last month
25. Recency - number of days since the last purchase

[ ] ↳ 2 cells hidden

Data Types

[ ] ↳ 7 cells hidden

Null Values

[ ] ↳ 5 cells hidden

Numeric Data Distribution

[ ] ↳ 8 cells hidden

Data Transformation

[ ] ↳ 19 cells hidden

Data Normalization

```
plt.figure(figsize=(20,7))
x = sns.boxplot(data=data)
x.set_xticklabels(x.get_xticklabels(),rotation=45)
```

```
[Text(0, 0, 'Age'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'Income'),
 Text(0, 0, 'Dt_Customer'),
 Text(0, 0, 'Recency'),
 Text(0, 0, 'NumDealsPurchases'),
```

Box plot shows Income having outliers and there is a vast difference between range of values between Income and other columns. We need to normalize data so all variables have equal weightage

```
 Text(0, 0, 'AmountSpent'),
```

```
from sklearn import preprocessing

# Create x to store scaled values as floats
x = data[["Age",    "Education",    "Income",   "Dt_Customer",  "Recency",  "NumDealsPurchases",    "NumWebVisitsMonth",    "

# Preparing for normalizing
min_max_scaler = preprocessing.MinMaxScaler()

# Transform the data to fit minmax processor
x_scaled = min_max_scaler.fit_transform(x)

# Run the normalizer on the dataframe
data[["Age",    "Education",    "Income",   "Dt_Customer",  "Recency",  "NumDealsPurchases",    "NumWebVisitsMonth",    "Chil

data.head()
```

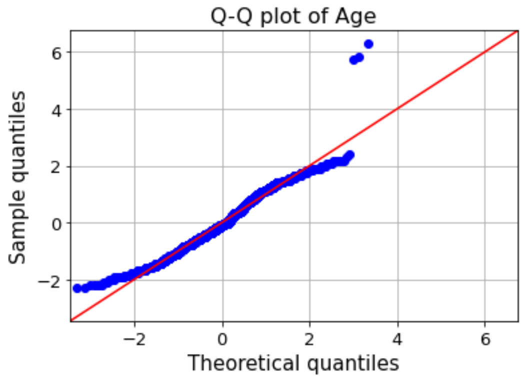
	Age	Education	Income	Dt_Customer	Recency	NumDealsPurchases	NumWebVisitsMonth	Complain	Response	Children	Amoun
0	0.378641	0.333333	0.084832	0.948498	0.585859	0.200000	0.35	0	1	0.000000	0.
1	0.407767	0.333333	0.067095	0.161660	0.383838	0.133333	0.25	0	0	0.666667	0.
2	0.300971	0.333333	0.105097	0.446352	0.262626	0.066667	0.20	0	0	0.000000	0.
3	0.116505	0.333333	0.037471	0.198856	0.262626	0.133333	0.30	0	0	0.333333	0.
4	0.145631	1.000000	0.085065	0.230329	0.949495	0.333333	0.25	0	0	0.333333	0.

```
plt.figure(figsize=(20,7))
x = sns.boxplot(data=data)
x.set_xticklabels(x.get_xticklabels(),rotation=45)
```

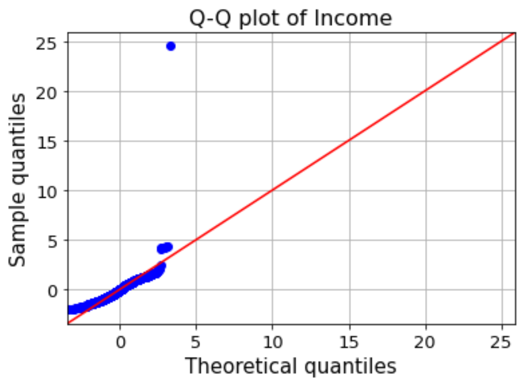
```
[Text(0, 0, 'Age'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'Income'),
 Text(0, 0, 'Dt_Customer'),
 Text(0, 0, 'Recency'),
 Text(0, 0, 'NumDealsPurchases'),
 Text(0, 0, 'NumWebVisitsMonth'),
 Text(0, 0, 'Complain'),
 Text(0, 0, 'Response'),
 Text(0, 0, 'Children')
```

```
#checking the distribution of independent variables
data = data.dropna()
from statsmodels.graphics.gofplots import qqplot
data_norm=data[['Age','Income','Dt_Customer','Recency','NumWebVisitsMonth','AmountSpent']]
for c in data_norm.columns[:]:
  plt.figure(figsize=(8,5))
  fig=qqplot(data_norm[c],line='45',fit='True')
  plt.xticks(fontsize=13)
  plt.yticks(fontsize=13)
  plt.xlabel("Theoretical quantiles",fontsize=15)
  plt.ylabel("Sample quantiles",fontsize=15)
  plt.title("Q-Q plot of {}".format(c),fontsize=16)
  plt.grid(True)
  plt.show()
```

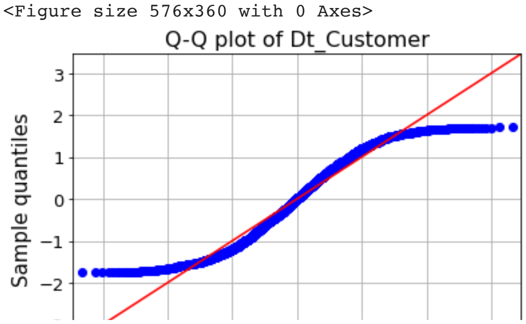
```
<Figure size 576x360 with 0 Axes>
```




```
<Figure size 576x360 with 0 Axes>
```




```
<Figure size 576x360 with 0 Axes>
```



- Q-Q plots show most of the data is normally distributed
- There are few Outliers in Income and age

```
<Figure size 576x360 with 0 Axes>
```

```
#pair plot to check the colinearity
# sns.pairplot(data)
```

```
#Using OLS for finding the p value to check the significant features
import statsmodels.api as sm

model = sm.OLS(data['Response'], data[["Age",   "Education",    "Income",   "Dt_Customer",
                                       "Recency"    ,"NumDealsPurchases",   "NumWebVisitsMonth",
                                       "Complain",  "Children", "AmountSpent"   ,"NumPurchased",
                                       "Prev_campaigns",    "Marital_Status_Couple",    "Marital_Status_Single"]]).fit()

# Print out the statistics
model.summary()
```

OLS Regression Results

Dep. Variable:	Response	**R-squared:**	0.301
Model:	OLS	**Adj. R-squared:**	0.297
Method:	Least Squares	**F-statistic:**	73.64
Date:	Sun, 05 Feb 2023	**Prob (F-statistic):**	3.34e-162
Time:	04:47:01	**Log-Likelihood:**	-465.46
No. Observations:	2240	**AIC:**	958.9
Df Residuals:	2226	**BIC:**	1039.
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Age	-0.0404	0.057	-0.712	0.477	-0.152	0.071
Education	0.1118	0.023	4.859	0.000	0.067	0.157
Income	-0.0596	0.246	-0.242	0.809	-0.543	0.423
Dt_Customer	0.2200	0.025	8.872	0.000	0.171	0.269
Recency	-0.2388	0.022	-11.030	0.000	-0.281	-0.196
NumDealsPurchases	0.1661	0.063	2.648	0.008	0.043	0.289
NumWebVisitsMonth	0.1149	0.075	1.534	0.125	-0.032	0.262
Complain	0.0383	0.066	0.582	0.560	-0.091	0.167
Children	-0.1185	0.034	-3.469	0.001	-0.186	-0.052
AmountSpent	0.2087	0.057	3.642	0.000	0.096	0.321
NumPurchased	-0.1920	0.054	-3.571	0.000	-0.297	-0.087

- Education, Dt_Customer, Recency, Childrean, Amount spent, NumPurchases, Prev_campaign have a p value of 0.00 denoting a very high significance
- Income have a p value of 0.8 which shows it does not have a lot of significance.
- Marital_Status_Couple has p value 0.666 whereas Marital_Status_Single has 0.001 which is interesting as they both originate from same data column. It shows the significance of creating dummy variables for categorical data

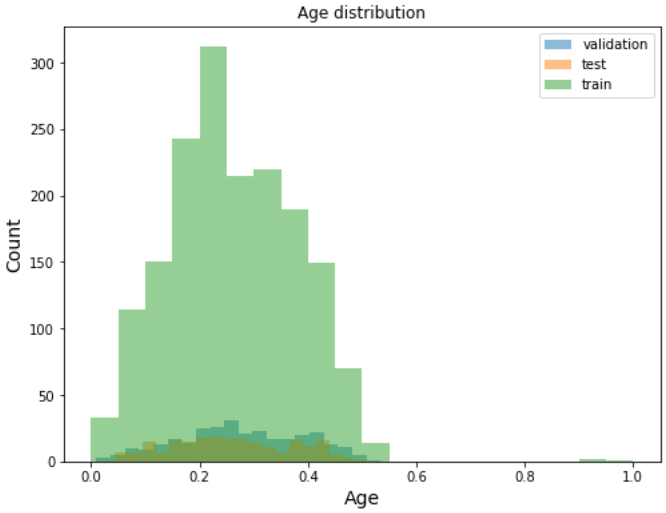
Logistic Regression

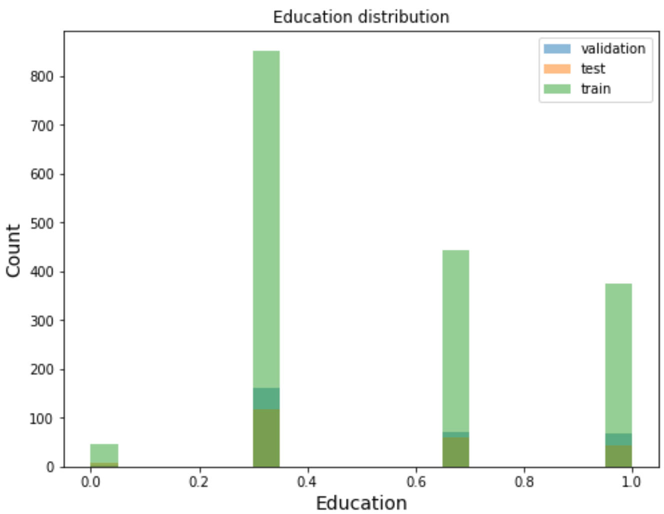
```
from sklearn.model_selection import  train_test_split

X = data[ ["Age",   "Education",    "Income",   "Dt_Customer",
                                       "Recency"    ,"NumDealsPurchases",   "NumWebVisitsMonth",
                                       "Complain",  "Children", "AmountSpent"   ,"NumPurchased",
                                       "Prev_campaigns",    "Marital_Status_Couple",    "Marital_Status_Single"]]

y = data['Response']

#Spliting data into Training 76.5%, Validation set 13.5% and Test set 10%

X_t, X_test, y_t, y_test = train_test_split(X, y, test_size=0.1, random_state=1)

X_train, X_val, y_train, y_val = train_test_split(X_t, y_t, test_size=0.15, random_state=1)
```

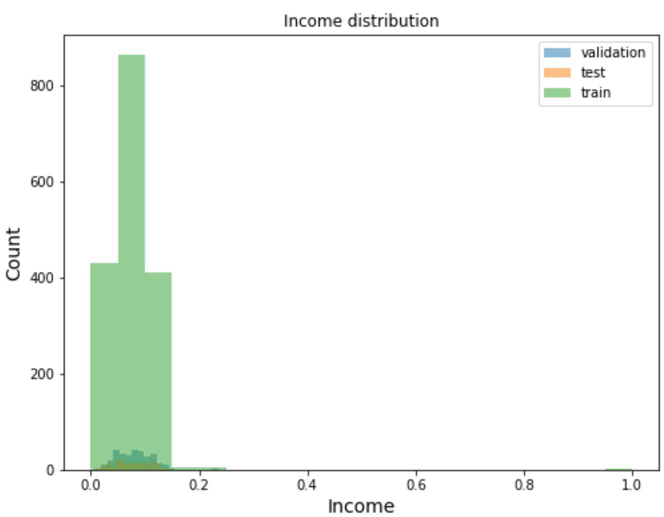
Splitting data for train test and validation 10% testing 15% of remaining train data for validation

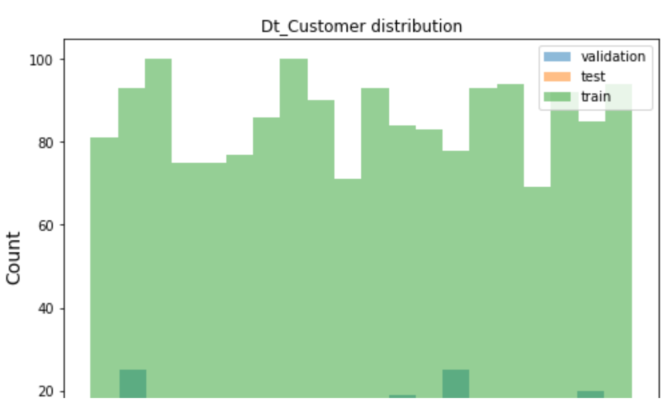
```
# Looking the data for test, training and validation set
X_test_plot = X_test[["Age",    "Education",    "Income",   "Dt_Customer",
                                       "Recency"    ,"NumDealsPurchases",   "NumWebVisitsMonth",
                                       "Complain",  "Children", "AmountSpent"   ,"NumPurchased",
                                       "Prev_campaigns",    "Marital_Status_Couple",    "Marital_Status_Single"]]

X_val_plot = X_val[["Age",  "Education",    "Income",   "Dt_Customer",
                                       "Recency"    ,"NumDealsPurchases",   "NumWebVisitsMonth",
                                       "Complain",  "Children", "AmountSpent"   ,"NumPurchased",
                                       "Prev_campaigns",    "Marital_Status_Couple",    "Marital_Status_Single"]]

X_train_plot = X_train[["Age",  "Education",    "Income",   "Dt_Customer",
                                       "Recency"    ,"NumDealsPurchases",   "NumWebVisitsMonth",
                                       "Complain",  "Children", "AmountSpent"   ,"NumPurchased",
                                       "Prev_campaigns",    "Marital_Status_Couple",    "Marital_Status_Single"]]

# Plotting the data to see the histogram
for c in X_test_plot.columns[:]:
  plt.figure(figsize=(8,6))
  plt.hist(X_val_plot[c], bins=20, alpha=0.5, label="validation")
```

```
plt.hist(X_test_plot[c], bins=20, alpha=0.5, label="test")
plt.hist(X_train_plot[c], bins=20, alpha=0.5, label="train")
plt.xlabel(c, size=14)
plt.ylabel("Count", size=14)
plt.legend(loc='upper right')
plt.title("{} distribution".format(c))
plt.show()
```


Age distribution
validation
test
train
Count
Age


Education distribution
validation
test
train
Count
Education


Income distribution
validation
test
train
Count
Income


Dt_Customer distribution
validation
test
train
Count

Distribution shows our data splits are evenly distributed

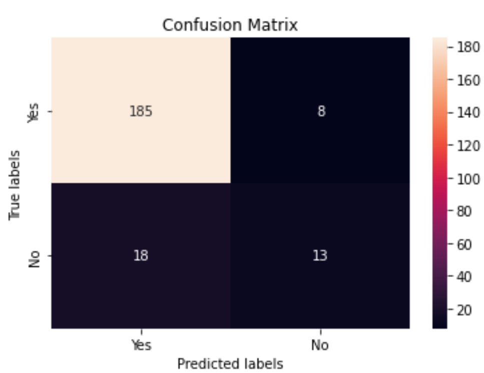
```
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from scipy import stats
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,confusion_matrix
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn import svm
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
from sklearn.inspection import permutation_importance
logreg=LogisticRegression()
loggreg_final=logreg.fit(X_train,y_train)
print(loggreg_final.score(X_train,y_train))
```

```
0.8832457676590777
```

Making Predictions

Our model has 88.32% training accuracy

```
y_pred=logreg.predict(X_test)

cm=confusion_matrix(y_test, y_pred)#confusion matrix for the logistic model prediction

ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);  #annot=True to annotate cells, ftm='g' to disable scientific notation

# labels, title and ticks
ax.set_xlabel('Predicted labels');
ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['Yes', 'No']);
ax.yaxis.set_ticklabels(['Yes', 'No']);
```



Above confusion matrix shows a good percentage of testing data is accurately predicted

```
print(classification_report(y_test, y_pred))
#classification report for logistic model prediction
```

```
              precision    recall  f1-score   support

           0       0.91      0.96      0.93       193
           1       0.62      0.42      0.50        31

    accuracy                           0.88       224
   macro avg       0.77      0.69      0.72       224
weighted avg       0.87      0.88      0.87       224
```

We have a higher precision for "No" i.e 0 of 0.91 while precision for "Yes" is 0.62 indicating we have a more accurate prediction chance for a negative customer response

```
#Understanding the important features
import eli5
from eli5.sklearn import PermutationImportance
perm = PermutationImportance(logreg, random_state=1).fit(X_test, y_test)
eli5.show_weights(perm, feature_names = X_test.columns.tolist())
```

Weight	Feature
0.0384 ± 0.0134	Prev_campaigns
0.0098 ± 0.0067	NumWebVisitsMonth
0.0036 ± 0.0143	NumPurchased
0.0036 ± 0.0067	NumDealsPurchases
0 ± 0.0000	Complain
0 ± 0.0000	Income
-0.0027 ± 0.0044	Age
-0.0027 ± 0.0091	Children
-0.0036 ± 0.0261	Dt_Customer
-0.0045 ± 0.0253	Education
-0.0054 ± 0.0242	Recency
-0.0089 ± 0.0056	AmountSpent
-0.0107 ± 0.0134	Marital_Status_Single
-0.0152 ± 0.0134	Marital_Status_Couple

Features in increasing order of significance as evident from permutaion importance

1. Prev_campaigns
2. NumWebVisitsMonth
3. NumPurchased
4. NumDealsPurchases
5. Education
6. Marital_Status_Couple
7. AmountSpent
8. Recency

Removing Outliers

```
data.Income.quantile(0.999)
```

```
0.23812708290722912
```

```
data.drop(data[data['Income'] >= 0.2].index, inplace = True)
```

```
data.isnull().sum()
```

```
Age                      0
Education                0
Income                   0
Dt_Customer              0
Recency                  0
NumDealsPurchases        0
NumWebVisitsMonth        0
Complain                 0
Response                 0
Children                 0
AmountSpent              0
NumPurchased             0
Prev_campaigns           0
Marital_Status_Couple    0
Marital_Status_Single    0
dtype: int64
```

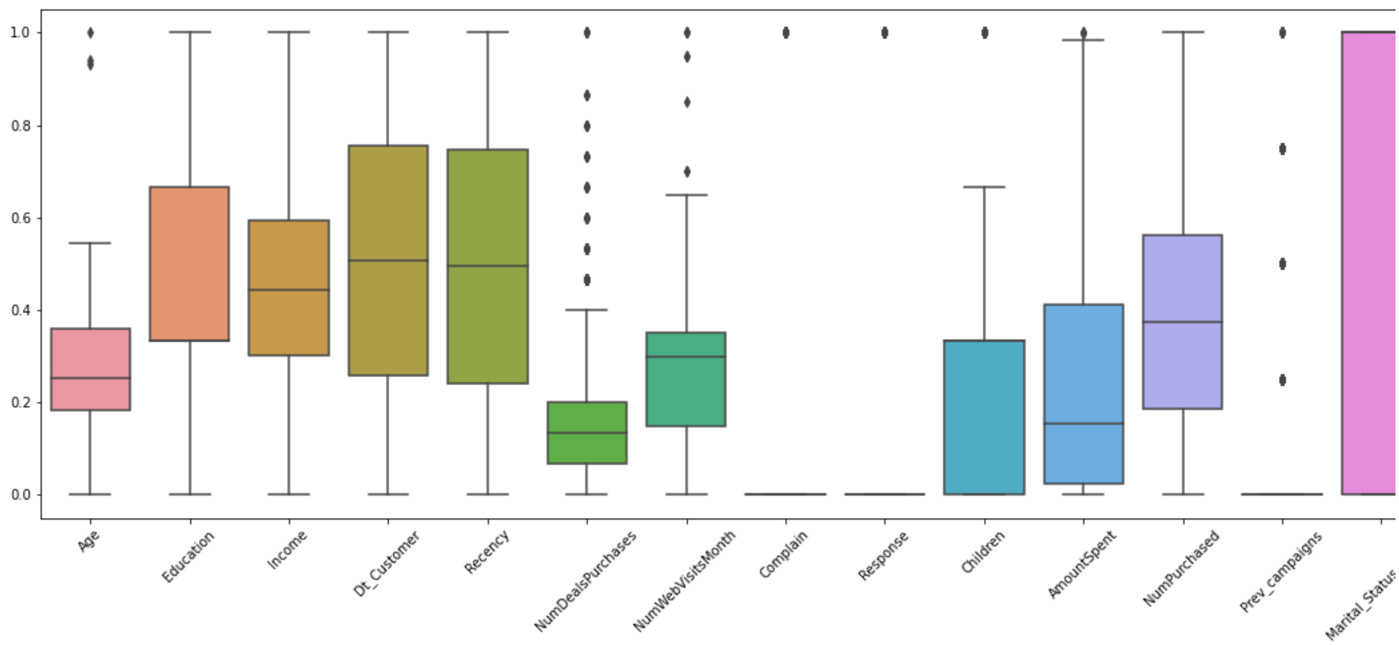
```
# Create x to store scaled values as floats
x = data[["Age",    "Education",    "Income",    "Dt_Customer",   "Recency",   "NumDealsPurchases",    "NumWebVisitsMonth",    "

# Preparing for normalizing
min_max_scaler = preprocessing.MinMaxScaler()

```

```
# Transform the data to fit minmax processor
x_scaled = min_max_scaler.fit_transform(x)

# Run the normalizer on the dataframe
data[["Age",    "Education",    "Income",   "Dt_Customer",  "Recency",  "NumDealsPurchases",    "NumWebVisitsMonth",    "Chil

data.head()
```

	Age	Education	Income	Dt_Customer	Recency	NumDealsPurchases	NumWebVisitsMonth	Complain	Response	Children	Amoun
0	0.378641	0.333333	0.503625	0.948498	0.585859	0.200000	0.35	0	1	0.000000	0.
1	0.407767	0.333333	0.398325	0.161660	0.383838	0.133333	0.25	0	0	0.666667	0.
2	0.300971	0.333333	0.623933	0.446352	0.262626	0.066667	0.20	0	0	0.000000	0.
3	0.116505	0.333333	0.222456	0.198856	0.262626	0.133333	0.30	0	0	0.333333	0.
4	0.145631	1.000000	0.505009	0.230329	0.949495	0.333333	0.25	0	0	0.333333	0.

```
plt.figure(figsize=(20,7))
x = sns.boxplot(data=data)
x.set_xticklabels(x.get_xticklabels(),rotation=45)
```

```
[Text(0, 0, 'Age'),
 Text(0, 0, 'Education'),
 Text(0, 0, 'Income'),
 Text(0, 0, 'Dt_Customer'),
 Text(0, 0, 'Recency'),
 Text(0, 0, 'NumDealsPurchases'),
 Text(0, 0, 'NumWebVisitsMonth'),
 Text(0, 0, 'Complain'),
 Text(0, 0, 'Response'),
 Text(0, 0, 'Children'),
 Text(0, 0, 'AmountSpent'),
 Text(0, 0, 'NumPurchased'),
 Text(0, 0, 'Prev_campaigns'),
 Text(0, 0, 'Marital_Status_Couple'),
 Text(0, 0, 'Marital_Status_Single')]
```



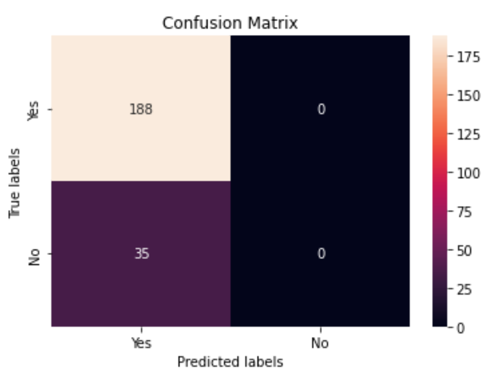
We Normalize data again after removing outliers from Income column

```
from sklearn.linear_model import LogisticRegression

data = data.dropna()
X = data[["Age",  "Education",  "Income", "Dt_Customer",
                                    "Recency"  ,"NumDealsPurchases", "NumWebVisitsMonth",
                                    "Complain",  "Children", "AmountSpent" ,"NumPurchased",
                                    "Prev_campaigns",  "Marital_Status_Couple",  "Marital_Status_Single"]]
```

```

y = data['Response']

#Spliting data into Training 76.5%, Validation set 13.5% and Test set 10%

X_t, X_test, y_t, y_test = train_test_split(X, y, test_size=0.1, random_state=1)

X_train, X_val, y_train, y_val = train_test_split(X_t, y_t, test_size=0.15, random_state=1)
logreg=LogisticRegression()
loggreg_final=logreg.fit(X_train,y_train)
print(loggreg_final.score(X_train,y_train))
```

```
0.8518518518518519
```

```
y_pred=logreg.predict(X_test)

cm=confusion_matrix(y_test, y_pred)#confusion matrix for the logistic model prediction

ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);  #annot=True to annotate cells, ftm='g' to disable scientific notation

# labels, title and ticks
ax.set_xlabel('Predicted labels');
ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['Yes', 'No']);
ax.yaxis.set_ticklabels(['Yes', 'No']);
```




```
print(classification_report(y_test, y_pred))
#classification report for logistic model prediction
```

```
              precision    recall  f1-score   support

           0       0.84      1.00      0.91       188
           1       0.00      0.00      0.00        35

    accuracy                           0.84       223
   macro avg       0.42      0.50      0.46       223
weighted avg       0.71      0.84      0.77       223

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318: UndefinedMetricWarning: Precision and F-score
  _warn_prf(average, modifier, msg_start, len(result))
```

```
from sklearn.impute import KNNImputer
from sklearn.preprocessing import MinMaxScaler
def create_missing(dataframe, percent, col):
    dataframe.loc[dataframe.sample(frac = percent).index, col] = np.nan
```

```
data_original = data.copy()
create_missing(data, 0.01, 'Income')
```

```
#checking if the any data is missing
def checkMissing(dataset):
  percent_missing = dataset.isnull().sum() * 100 / len(data)
  null_values_total = dataset.isnull().sum()
```

```
    missing_value_df = pd.DataFrame({
                                'Missing_Total' : null_values_total,
                                'percent_missing': percent_missing,
                                })
    return missing_value_df

checkMissing(data)
```

	Missing_Total	percent_missing
Age	0	0.000000
Education	0	0.000000
Income	22	0.988764
Dt_Customer	0	0.000000
Recency	0	0.000000
NumDealsPurchases	0	0.000000
NumWebVisitsMonth	0	0.000000
Complain	0	0.000000
Response	0	0.000000
Children	0	0.000000
AmountSpent	0	0.000000
NumPurchased	0	0.000000
Prev_campaigns	0	0.000000
Marital_Status_Couple	0	0.000000
Marital_Status_Single	0	0.000000

Average

```
number_1_idx = list(np.where(data['Income'].isna())[0])
```

```
data['Income'].fillna(value=data['Income'].mean(), inplace=True)
checkMissing(data)
```

	Missing_Total	percent_missing
Age	0	0.0
Education	0	0.0
Income	0	0.0
Dt_Customer	0	0.0
Recency	0	0.0
NumDealsPurchases	0	0.0
NumWebVisitsMonth	0	0.0
Complain	0	0.0
Response	0	0.0
Children	0	0.0
AmountSpent	0	0.0
NumPurchased	0	0.0
Prev_campaigns	0	0.0
Marital_Status_Couple	0	0.0
Marital_Status_Single	0	0.0

```
data_mn = data.iloc[number_1_idx]
data_og = data_original.iloc[number_1_idx]
```

```
# The mean squared error
print('Mean squared error: %.2f'% mean_squared_error(data_og['Income'], data_mn['Income']))
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f'% r2_score(data_og['Income'], data_mn['Income']))
r2 = r2_score(data_og['Income'], data_mn['Income'])
print('R^2 score on test set =',r2)

```

```
Mean squared error: 0.03
Coefficient of determination: -0.00
R^2 score on test set = -0.0009562959256552706
```

Categorical mean

```
data = data_original.copy()
create_missing(data, 0.05, 'Income')
checkMissing(data)

```

	Missing_Total	percent_missing
Age	0	0.000000
Education	0	0.000000
Income	111	4.988764
Dt_Customer	0	0.000000
Recency	0	0.000000
NumDealsPurchases	0	0.000000
NumWebVisitsMonth	0	0.000000
Complain	0	0.000000
Response	0	0.000000
Children	0	0.000000
AmountSpent	0	0.000000
NumPurchased	0	0.000000
Prev_campaigns	0	0.000000
Marital_Status_Couple	0	0.000000
Marital_Status_Single	0	0.000000

```
number_5_idx = list(np.where(data['Income'].isna())[0])
data['Income'].fillna(data.groupby('Education')['Income'].transform('mean'), inplace = True)
checkMissing(data)
```

	Missing_Total	percent_missing
Age	0	0.0
Education	0	0.0

```
data_mn = data.iloc[number_5_idx]
data_og = data_original.iloc[number_5_idx]
# The mean squared error
print('Mean squared error: %.2f'% mean_squared_error(data_original['Income'], data['Income']))
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f'% r2_score(data_original['Income'], data['Income']))
r2 = r2_score(data_original['Income'], data['Income'])
print('R^2 score on test set =',r2)
```

```
Mean squared error: 0.00
Coefficient of determination: 0.95
R^2 score on test set = 0.9531689754402813
```

Children	0	0.0

KNN inpute

```
data = data_original.copy()
create_missing(data, 0.1, 'Income')
checkMissing(data)

```

	Missing_Total	percent_missing
Age	0	0.000000
Education	0	0.000000
Income	222	9.977528
Dt_Customer	0	0.000000
Recency	0	0.000000
NumDealsPurchases	0	0.000000
NumWebVisitsMonth	0	0.000000
Complain	0	0.000000
Response	0	0.000000
Children	0	0.000000
AmountSpent	0	0.000000
NumPurchased	0	0.000000
Prev_campaigns	0	0.000000
Marital_Status_Couple	0	0.000000
Marital_Status_Single	0	0.000000

```
number_10_idx = list(np.where(data['Income'].isna())[0])
```

```
imputer = KNNImputer(n_neighbors=5)
data = pd.DataFrame(imputer.fit_transform(data), columns = data.columns)
checkMissing(data)
```

	Missing_Total	percent_missing
Age	0	0.0
Education	0	0.0
Income	0	0.0
Dt_Customer	0	0.0
Recency	0	0.0
NumDealsPurchases	0	0.0

```
data_mn = data.iloc[number_10_idx]
data_og = data_original.iloc[number_10_idx]
# The mean squared error
print('Mean squared error: %.2f'% mean_squared_error(data_original['Income'], data['Income']))
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f'% r2_score(data_original['Income'], data['Income']))
r2 = r2_score(data_original['Income'], data['Income'])
print('R^2 score on test set =',r2)
```

```
Mean squared error: 0.00
Coefficient of determination: 0.98
R^2 score on test set = 0.9794301329289421
```

Answer the following questions:

- Do the training and test sets have the same data?

```
No they donot have the same values
```

- In the predictor variables independent of all the other predictor variables?

```
There are some variables like amount spent and number of orders that are correlated
```

- Which predictor variables are the most important?

1. Prev_campaigns
2. NumWebVisitsMonth
3. NumPurchased
4. NumDealsPurchases
5. Education
6. Marital_Status_Couple
7. AmountSpent
8. Recency

- Remove outliers and keep outliers (does if have an effect of the final predictive model)?

```
From the confusion matrix it is evident we are getting better model and predictions
```

- Remove 1%, 5%, and 10% of your data randomly and impute the values back using at least 3 imputation methods. How well did the methods recover the missing values? That is remove some data, check the % error on residuals for numeric data and check for bias and variance of the error.

```
For our dataset while removing values for Income column all 3 methods have similar results although one can argu
```