DATA 119 Final Project

Kris Peng and Sam Leung

Project A

```
import numpy as np
import pandas as pd
import plotnine as p9
import statsmodels.api as sm
import sklearn.metrics as metrics
```

```
df = pd.read_csv("marketing_campaign.csv", sep="\t")
pd.set_option('display.max_rows', None)
df.head(5)
```

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

```
df.columns
```

```
Index(['ID', 'Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
```

```
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_Revenue', 'Response'],
      dtype='object')
```

```
#These variables are meaningless
#so we decide to delete them

df = df.drop(columns=['ID', 'Z_CostContact','Z_Revenue'])
df.columns
```

```
Index(['Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Response'],
      dtype='object')
```

```
# Name the variables in a more meaning way

df = df.rename(columns={"Response": "AcceptedLastCmp"})
df.columns
```

```
Index(['Year_Birth', 'Education', 'Marital_Status', 'Income', 'Kidhome',
       'Teenhome', 'Dt_Customer', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'AcceptedLastCmp'],
      dtype='object')
```

```
df['Income'].isna().sum() / df.shape[0]
#only 1% missing data, so let's drop it
df = df.dropna()
```

```
#Summary statistics for variables
print(df['Year_Birth'].describe())
```

```
count    2216.000000
mean     1968.820397
std        11.985554
min      1893.000000
25%      1959.000000
50%      1970.000000
75%      1977.000000
max      1996.000000
Name: Year_Birth, dtype: float64
```

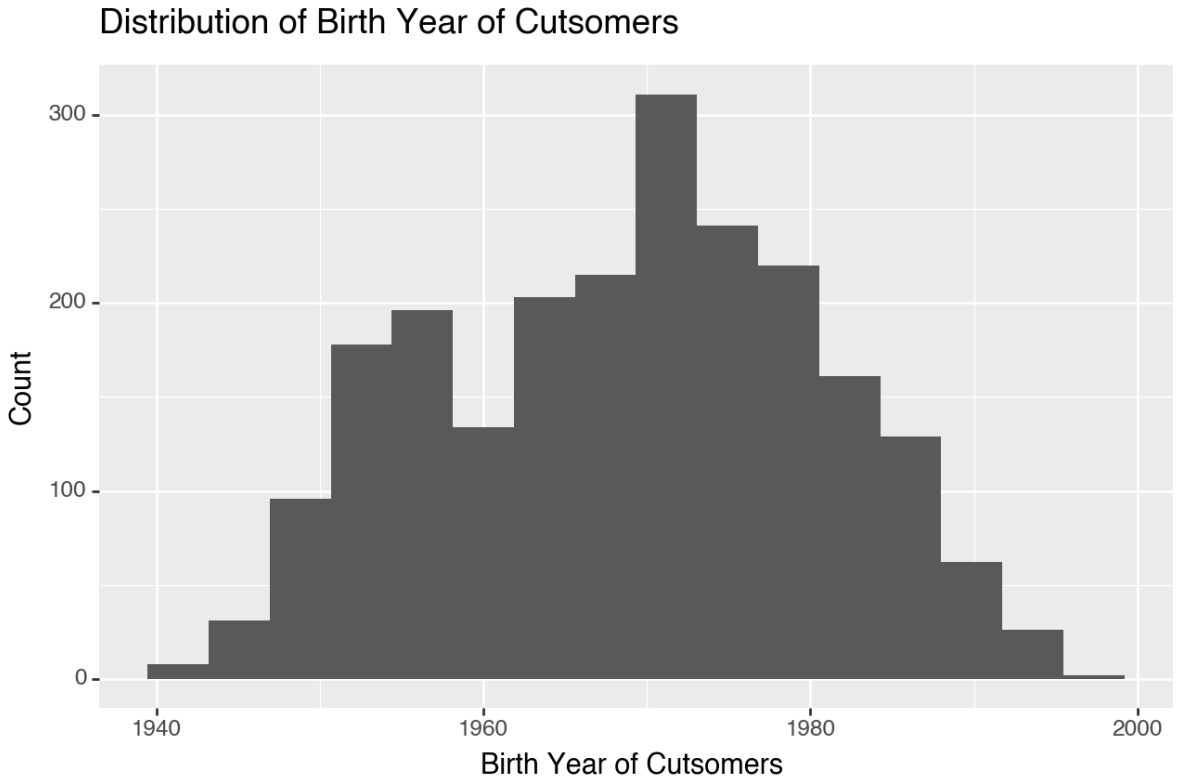
```
print(df[df['Year_Birth'] < 1920]['Dt_Customer'])
print("It looks like some data entry error")
print("because it is quite impossible to be born that early \nand become a customer for th
print("So we can delete the data")

index = df[df['Year_Birth'] < 1920].index
df = df.drop(index)
```

```
192    26-09-2013
239    17-05-2014
339    26-09-2013
Name: Dt_Customer, dtype: object
It looks like some data entry error
because it is quite impossible to be born that early
and become a customer for the first time in 2010s.
So we can delete the data
```

```
#Summary statistics for variables
print(df['Year_Birth'].describe())
```

```
count    2213.000000
mean     1968.917307
std        11.700216
min      1940.000000
25%      1959.000000
50%      1970.000000
75%      1977.000000
max      1996.000000
Name: Year_Birth, dtype: float64
```

```
# Visualize the distirbution of variables
(p9.ggplot(df) +
 p9.aes (x = 'Year_Birth') +
 p9.geom_histogram(bins=16)+
 p9.labs(x = "Birth Year of Cutsomers \n", y = "Count",title= "Distribution of Birth Year
         caption = "This histogram shows the distribution of Birth Year of Cutsomers.\n" +
          "The distribution is centered at approximately 1970, and roughly ranges from 194
          "It is an unimodal, left skewed distribution. There are no obvious unusual value
```



This histogram shows the distribution of Birth Year of Cutsomers.
The distribution is centered at approximately 1970, and roughly ranges from 1940 to 1995.
It is an unimodal, left skewed distribution. There are no obvious unusual values.

```
<Figure Size: (640 x 480)>
```

```
print(df['Income'].describe())
```

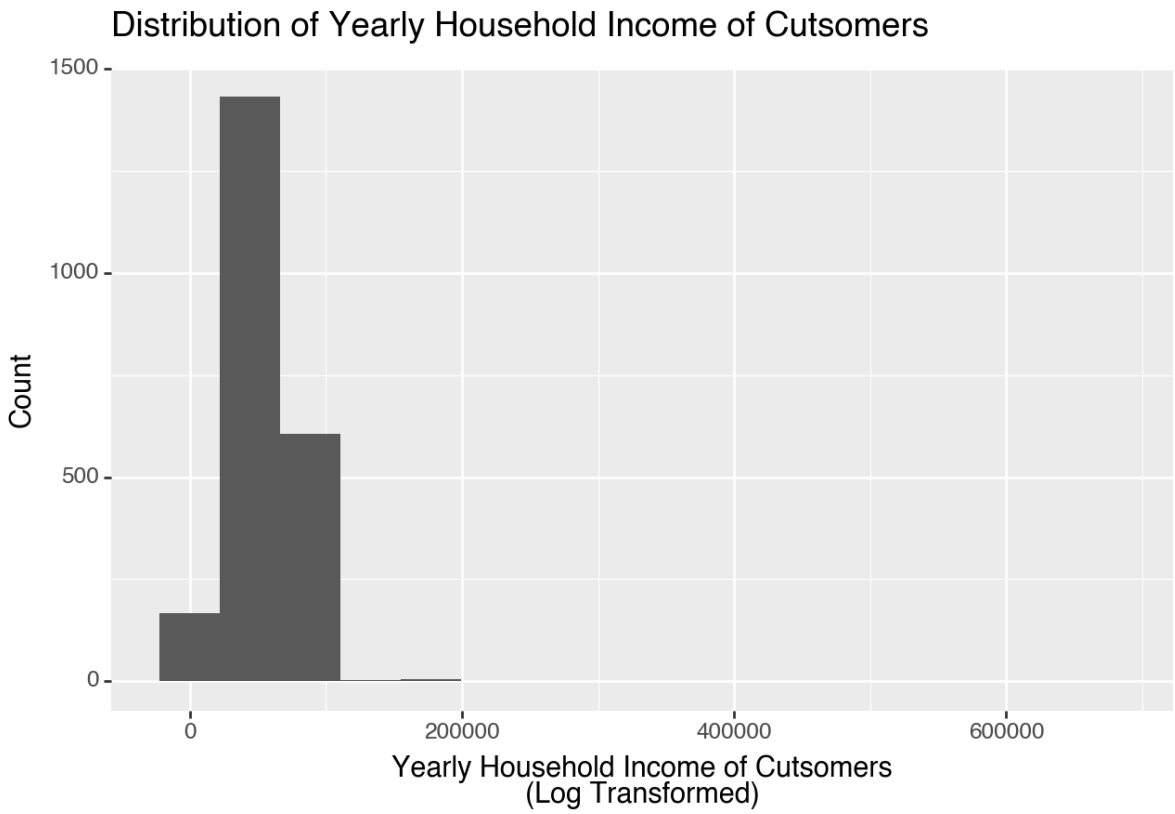
```
count      2213.000000
```

```
mean      52236.581563
std       25178.603047
min        1730.000000
25%       35246.000000
50%       51373.000000
75%       68487.000000
max      666666.000000
Name: Income, dtype: float64
```

```
print(df['Income'].describe())
```

```
count      2213.000000
mean      52236.581563
std       25178.603047
min        1730.000000
25%       35246.000000
50%       51373.000000
75%       68487.000000
max      666666.000000
Name: Income, dtype: float64
```

```
# Visualize the distirbution of variables
(p9.ggplot(df) +
 p9.aes (x = 'Income') +
 p9.geom_histogram(bins=16)+
 p9.labs(x = "Yearly Household Income of Cutsomers\n(Log Transformed)", y = "Count",title=
         caption = "This histogram shows the distribution of yearly household income of cu
          "The distribution is centered at approximately 50,000, and roughly ranges from 1
          "It is an unimodal, left skewed distribution. There are unusually high values.")
```



This histogram shows the distribution of yearly household income of cutsomers.
The distribution is centered at approximately 50,000, and roughly ranges from 1000 to 700,000.
It is an unimodal, left skewed distribution. There are unusually high values.

```
<Figure Size: (640 x 480)>
```

```
print(df['Dt_Customer'].describe())
```

```
count           2213
unique           662
top       31-08-2012
freq              12
Name: Dt_Customer, dtype: object
```

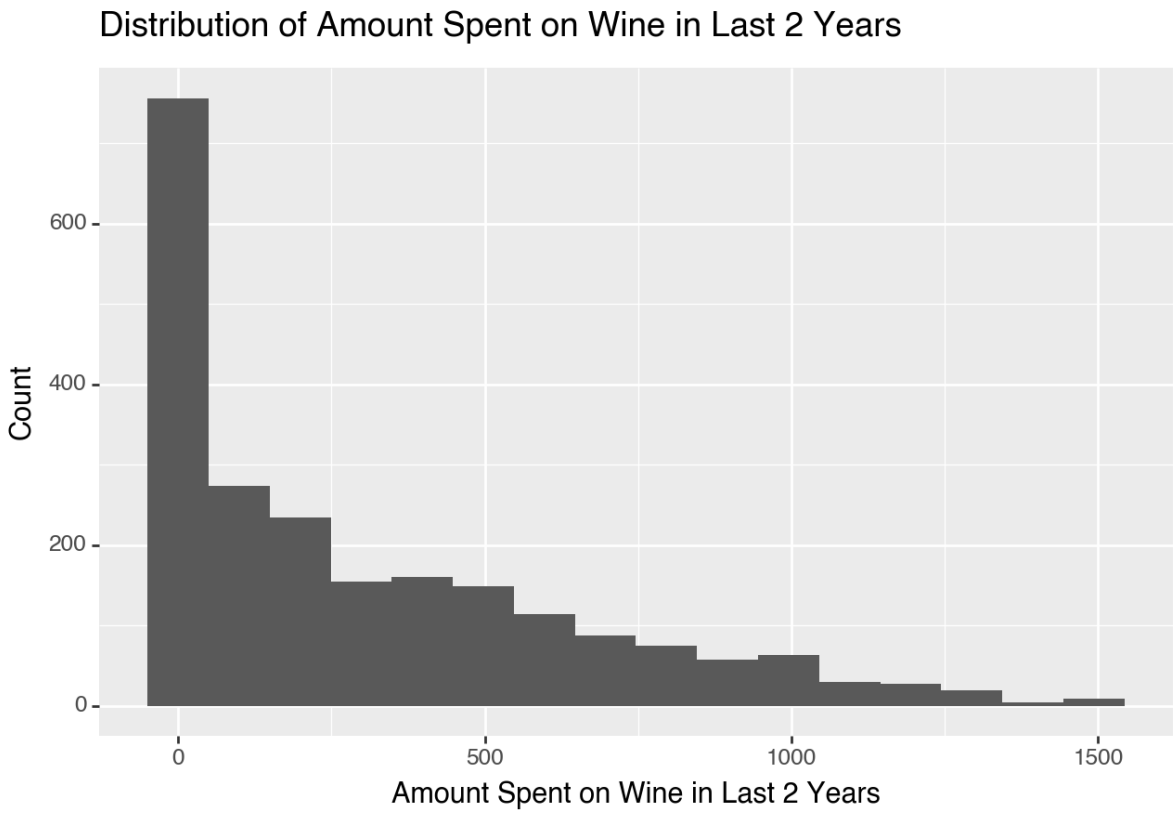
```
print(df['Recency'].describe())
```

```
count    2213.000000
mean       49.007682
std        28.941864
min         0.000000
25%        24.000000
50%        49.000000
75%        74.000000
max        99.000000
Name: Recency, dtype: float64
```

```
print(df['MntWines'].describe())
```

```
count    2213.000000
mean      305.153638
std       337.305490
min         0.000000
25%        24.000000
50%       175.000000
75%       505.000000
max      1493.000000
Name: MntWines, dtype: float64
```

```
# Visualize the distirbution of variables
(p9.ggplot(df) +
 p9.aes (x = 'MntWines') +
 p9.geom_histogram(bins=16)+
 p9.labs(x = "Amount Spent on Wine in Last 2 Years", y = "Count",title= "Distribution of A
         caption = "This histogram shows the distribution of amount spent on wine in last
          "The distribution is centered at approximately 0, and roughly ranges from 0 to 1
          "It is an unimodal, right skewed distribution. There are no unusual values."))
```




```
<Figure Size: (640 x 480)>
```

```
print(df['MntFruits'].describe())
```

```
count    2213.000000
mean       26.323995
std        39.735932
min         0.000000
25%         2.000000
50%         8.000000
75%        33.000000
max       199.000000
Name: MntFruits, dtype: float64
```

```
print(df['MntMeatProducts'].describe())
```

```
count    2213.000000
mean      166.962494
std       224.226178
min         0.000000
25%        16.000000
50%        68.000000
75%       232.000000
max      1725.000000
Name: MntMeatProducts, dtype: float64
```

```
print(df['MntFishProducts'].describe())
```

```
count    2213.000000
mean       37.635337
std        54.763278
min         0.000000
25%         3.000000
50%        12.000000
75%        50.000000
max       259.000000
Name: MntFishProducts, dtype: float64
```

```
print(df['MntSweetProducts'].describe())
```

```
count    2213.000000
mean       27.034794
std        41.085433
min         0.000000
25%         1.000000
50%         8.000000
75%        33.000000
max       262.000000
Name: MntSweetProducts, dtype: float64
```

```
print(df['MntGoldProds'].describe())
```

```
count    2213.000000
mean       43.911432
std        51.699746
min         0.000000
25%         9.000000
50%        24.000000
75%        56.000000
max       321.000000
Name: MntGoldProds, dtype: float64
```

```
print(df['NumDealsPurchases'].describe())
```

```
count    2213.000000
mean        2.325350
std         1.924402
min         0.000000
25%         1.000000
50%         2.000000
75%         3.000000
max        15.000000
Name: NumDealsPurchases, dtype: float64
```

```
print(df['NumWebPurchases'].describe())
```

```
count    2213.000000
mean        4.087664
std         2.741664
min         0.000000
25%         2.000000
50%         4.000000
75%         6.000000
max        27.000000
Name: NumWebPurchases, dtype: float64
```

```
print(df['NumCatalogPurchases'].describe())
```

```
count    2213.000000
mean        2.671487
```

```
std         2.927096
min         0.000000
25%         0.000000
50%         2.000000
75%         4.000000
max        28.000000
Name: NumCatalogPurchases, dtype: float64
```

```
print(df['NumStorePurchases'].describe())
```

```
count    2213.000000
mean        5.805242
std         3.250752
min         0.000000
25%         3.000000
50%         5.000000
75%         8.000000
max        13.000000
Name: NumStorePurchases, dtype: float64
```

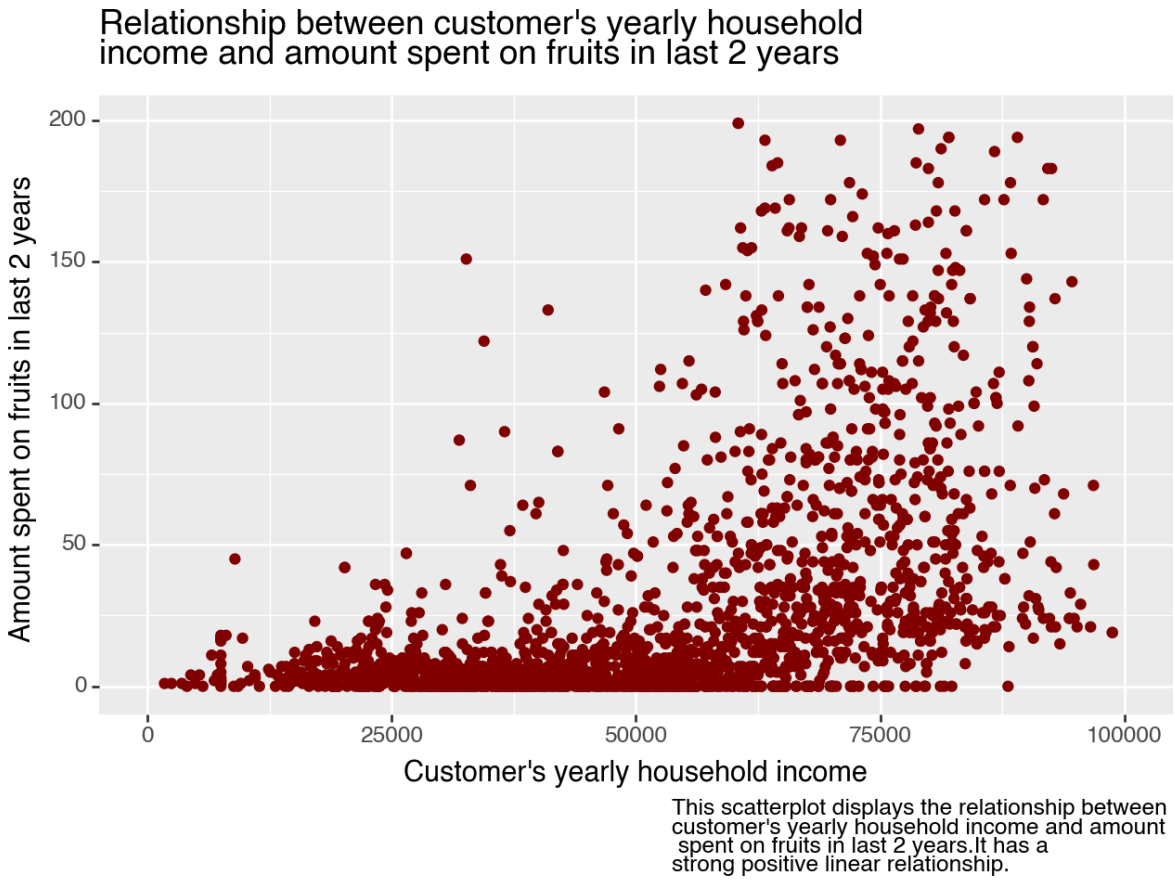
```
print(df['NumWebVisitsMonth'].describe())
```

```
count    2213.000000
mean        5.321735
std         2.425092
min         0.000000
25%         3.000000
50%         6.000000
75%         7.000000
max        20.000000
Name: NumWebVisitsMonth, dtype: float64
```

```
# create scatterplots
(p9.ggplot(df, p9.aes(x = 'Income', y = 'MntFruits'))  +
 p9.geom_point(color = 'maroon') +
 p9.xlim(0,100000) +
 p9.labs(x = "Customer's yearly household income", y = "Amount spent on fruits in last 2 y
        title= "Relationship between customer's yearly household \nincome and amount spent
          caption = "This scatterplot displays the relationship between \ncustomer's yearl
```

```
          + "It has a \nstrong positive linear relationship."))
```

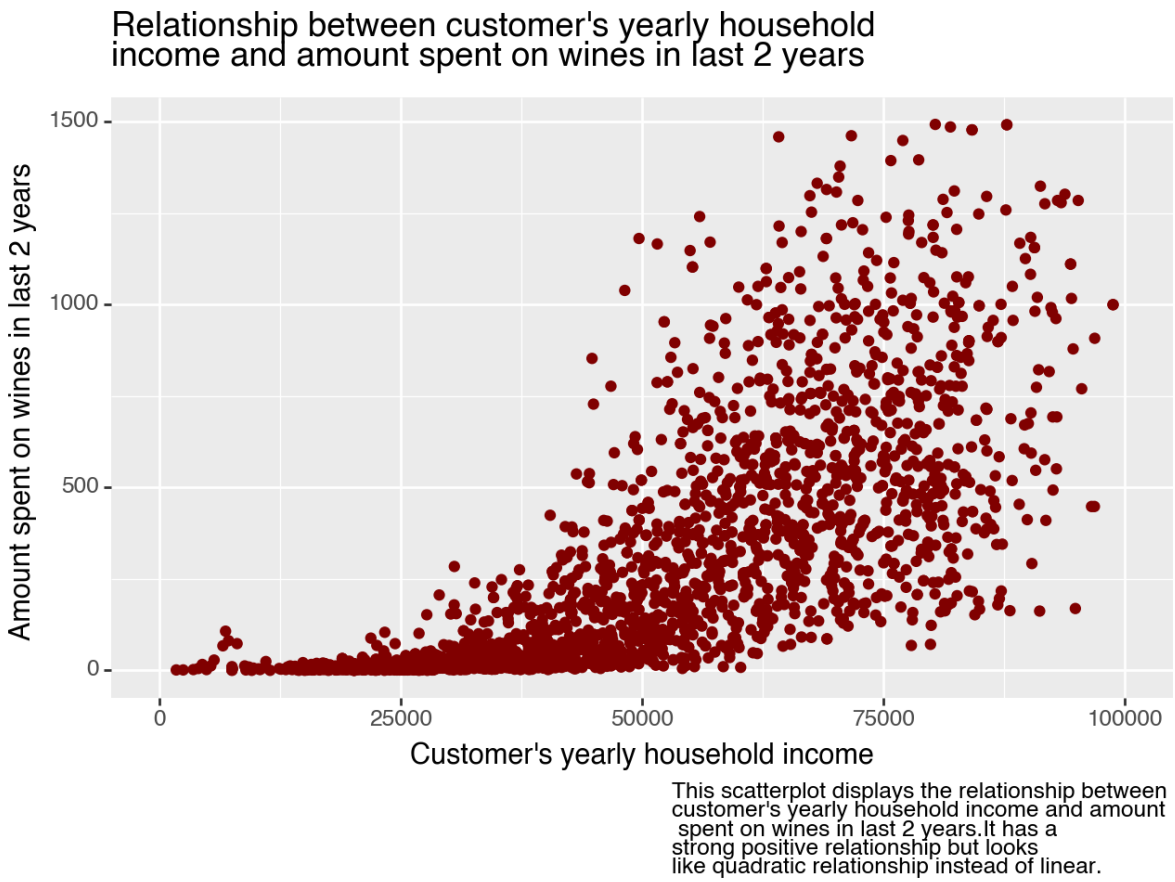
```
/opt/homebrew/lib/python3.9/site-packages/plotnine/layer.py:364: PlotnineWarning: geom_point
```




```
<Figure Size: (640 x 480)>
```

```
# create scatterplots
(p9.ggplot(df, p9.aes(x = 'Income', y = 'MntWines'))  +
 p9.geom_point(color = 'maroon') +
 p9.xlim(0,100000) +
 p9.labs(x = "Customer's yearly household income", y = "Amount spent on wines in last 2 ye
        title= "Relationship between customer's yearly household \nincome and amount spent
          caption = "This scatterplot displays the relationship between \ncustomer's yearl
          + "It has a \nstrong positive relationship but looks \nlike quadratic relationsh
```

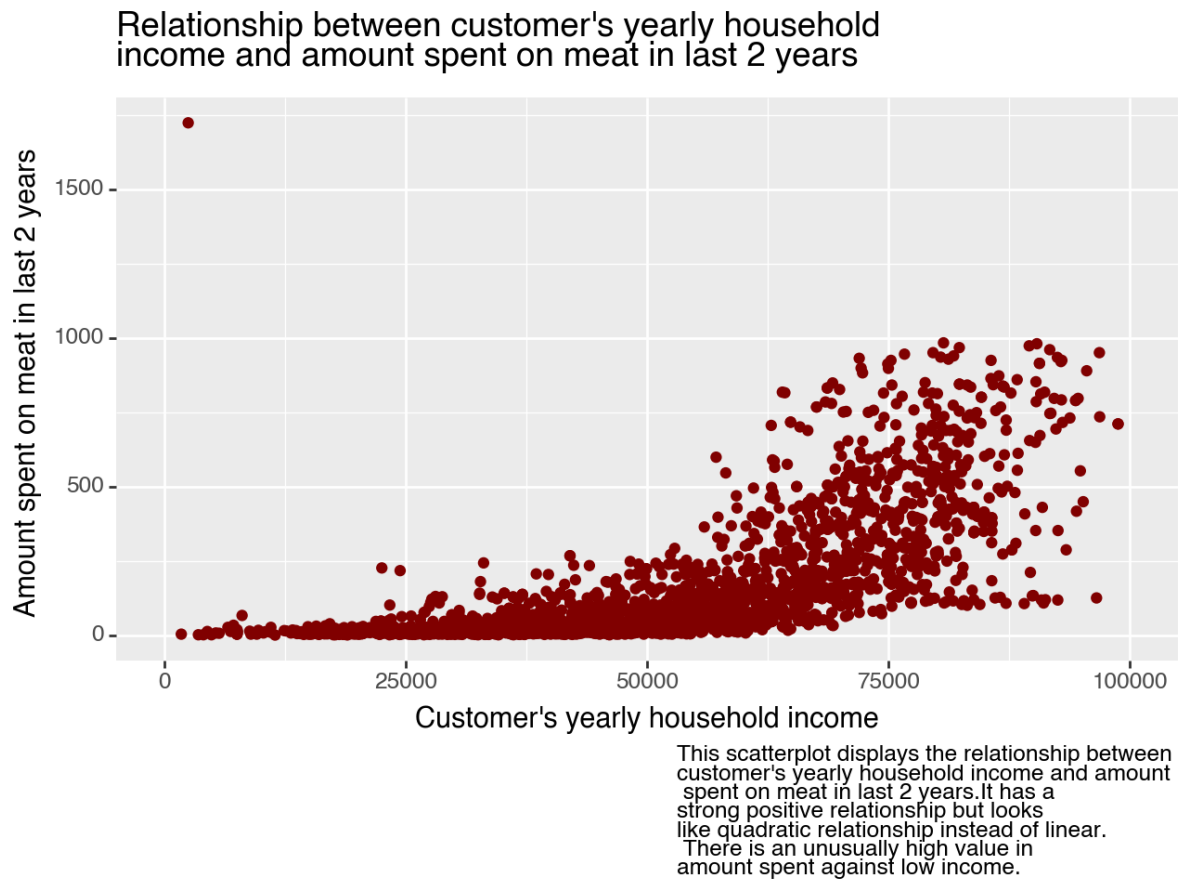
```
/opt/homebrew/lib/python3.9/site-packages/plotnine/layer.py:364: PlotnineWarning: geom_point
```




```
<Figure Size: (640 x 480)>
```

```
# create scatterplots
(p9.ggplot(df, p9.aes(x = 'Income', y = 'MntMeatProducts'))  +
 p9.geom_point(color = 'maroon') +
 p9.xlim(0,100000) +
 p9.labs(x = "Customer's yearly household income", y = "Amount spent on meat in last 2 yea
        title= "Relationship between customer's yearly household \nincome and amount spent
          caption = "This scatterplot displays the relationship between \ncustomer's yearl
          + "It has a \nstrong positive relationship but looks \nlike quadratic relationsh
```

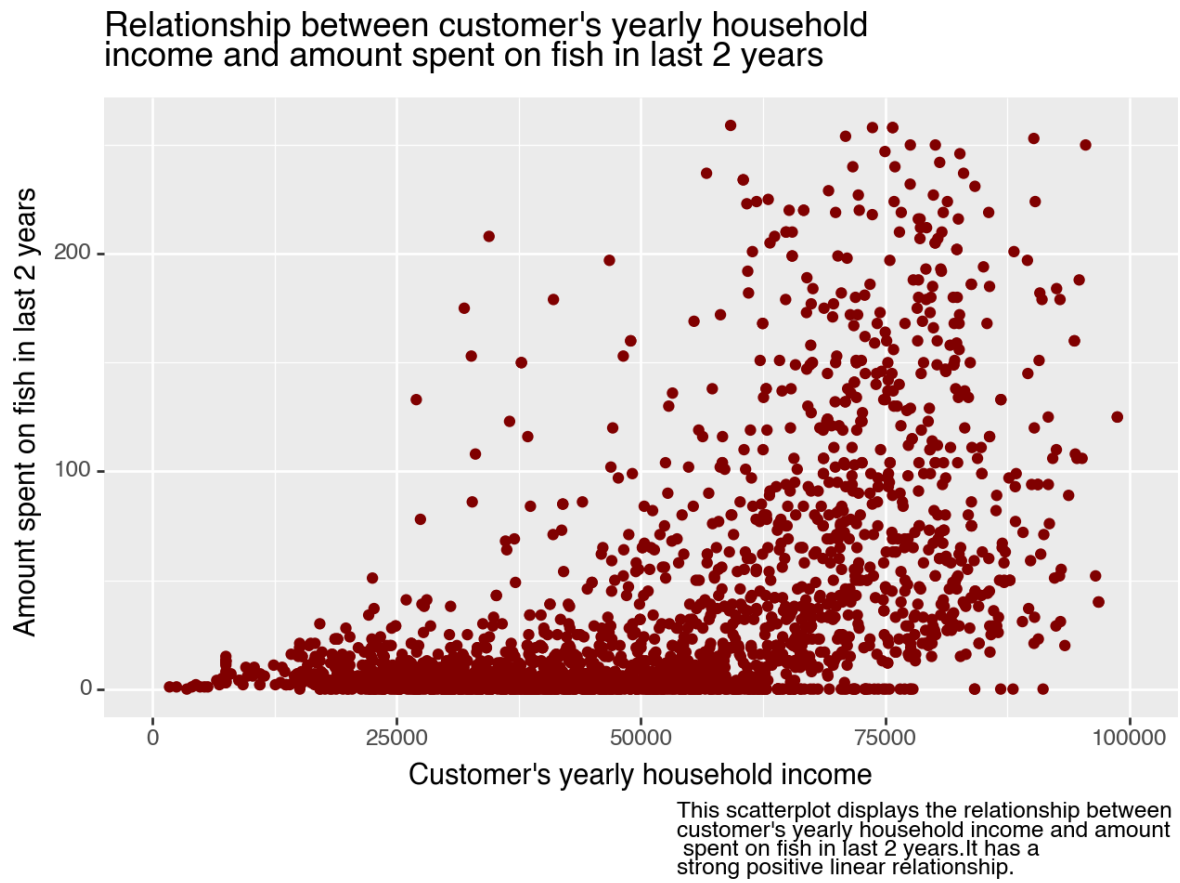
```
/opt/homebrew/lib/python3.9/site-packages/plotnine/layer.py:364: PlotnineWarning: geom_point
```




```
<Figure Size: (640 x 480)>
```

```
# create scatterplots
(p9.ggplot(df, p9.aes(x = 'Income', y = 'MntFishProducts'))  +
 p9.geom_point(color = 'maroon') +
 p9.xlim(0,100000) +
 p9.labs(x = "Customer's yearly household income", y = "Amount spent on fish in last 2 yea
        title= "Relationship between customer's yearly household \nincome and amount spent
          caption = "This scatterplot displays the relationship between \ncustomer's yearl
          + "It has a \nstrong positive linear relationship."))
```

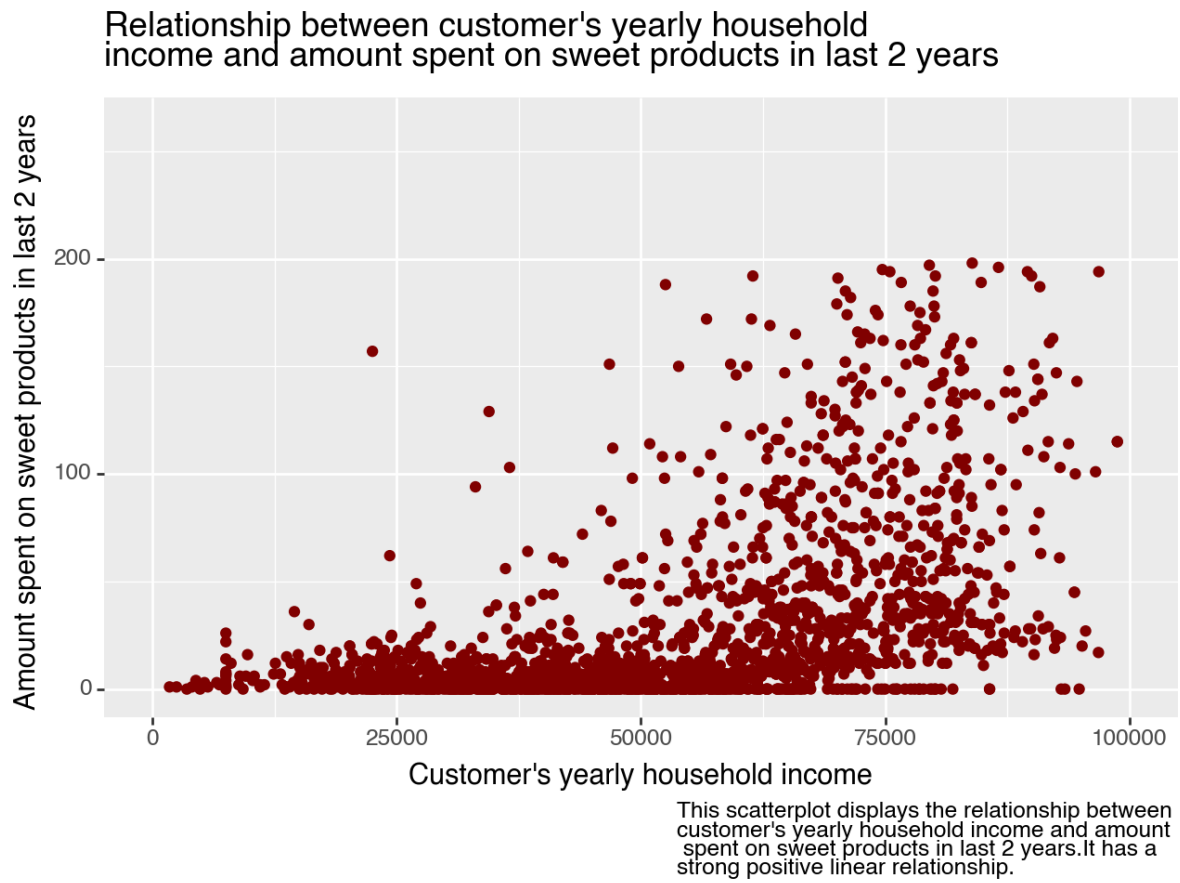
```
/opt/homebrew/lib/python3.9/site-packages/plotnine/layer.py:364: PlotnineWarning: geom_point
```



<Figure Size: (640 x 480)>

```
# create scatterplots
(p9.ggplot(df, p9.aes(x = 'Income', y = 'MntSweetProducts'))  +
 p9.geom_point(color = 'maroon') +
 p9.xlim(0,100000) +
 p9.labs(x = "Customer's yearly household income", y = "Amount spent on sweet products in
        title= "Relationship between customer's yearly household \nincome and amount spent
          caption = "This scatterplot displays the relationship between \ncustomer's yearl
          + "It has a \nstrong positive linear relationship."))
```

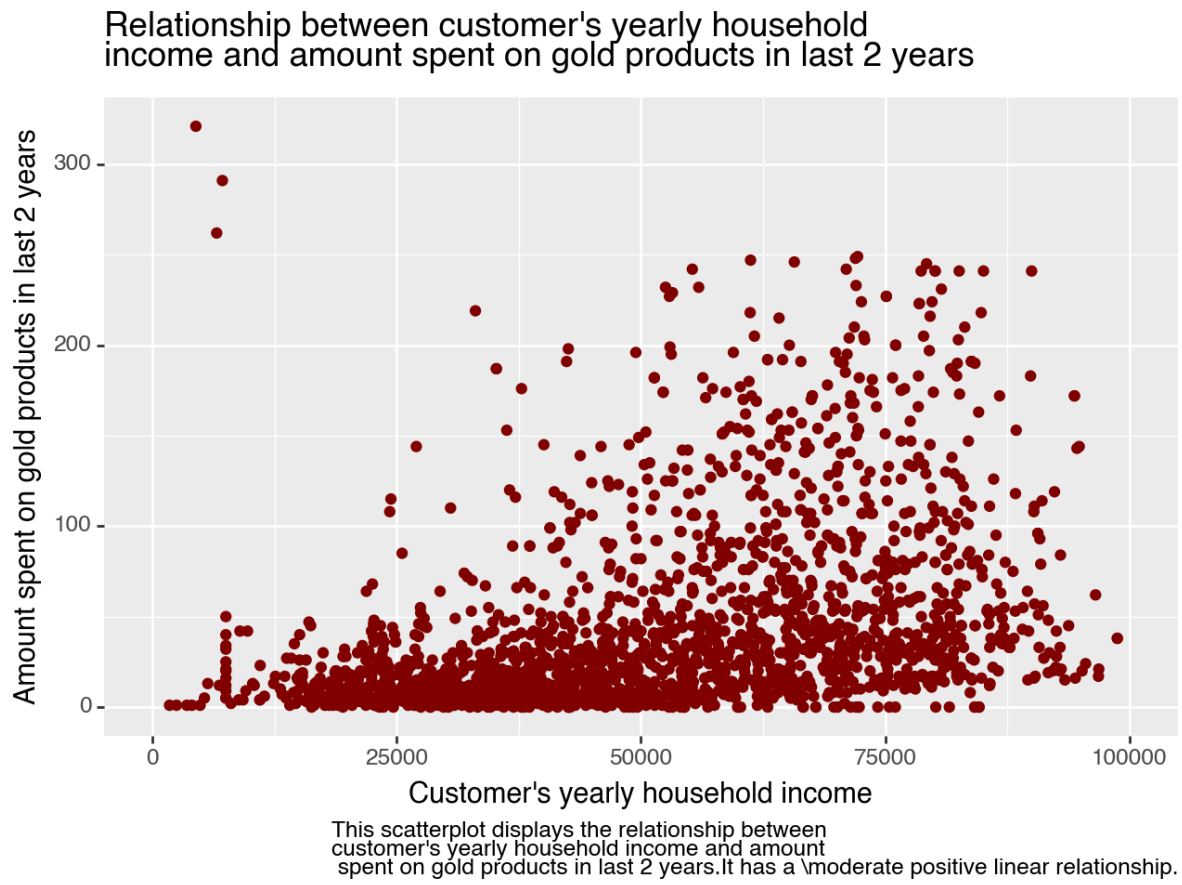
/opt/homebrew/lib/python3.9/site-packages/plotnine/layer.py:364: PlotnineWarning: geom_point



<Figure Size: (640 x 480)>

```
# create scatterplots
(p9.ggplot(df, p9.aes(x = 'Income', y = 'MntGoldProds'))  +
 p9.geom_point(color = 'maroon') +
 p9.xlim(0,100000) +
 p9.labs(x = "Customer's yearly household income", y = "Amount spent on gold products in l
        title= "Relationship between customer's yearly household \nincome and amount spent
          caption = "This scatterplot displays the relationship between \ncustomer's yearl
          + "It has a \moderate positive linear relationship."))
```

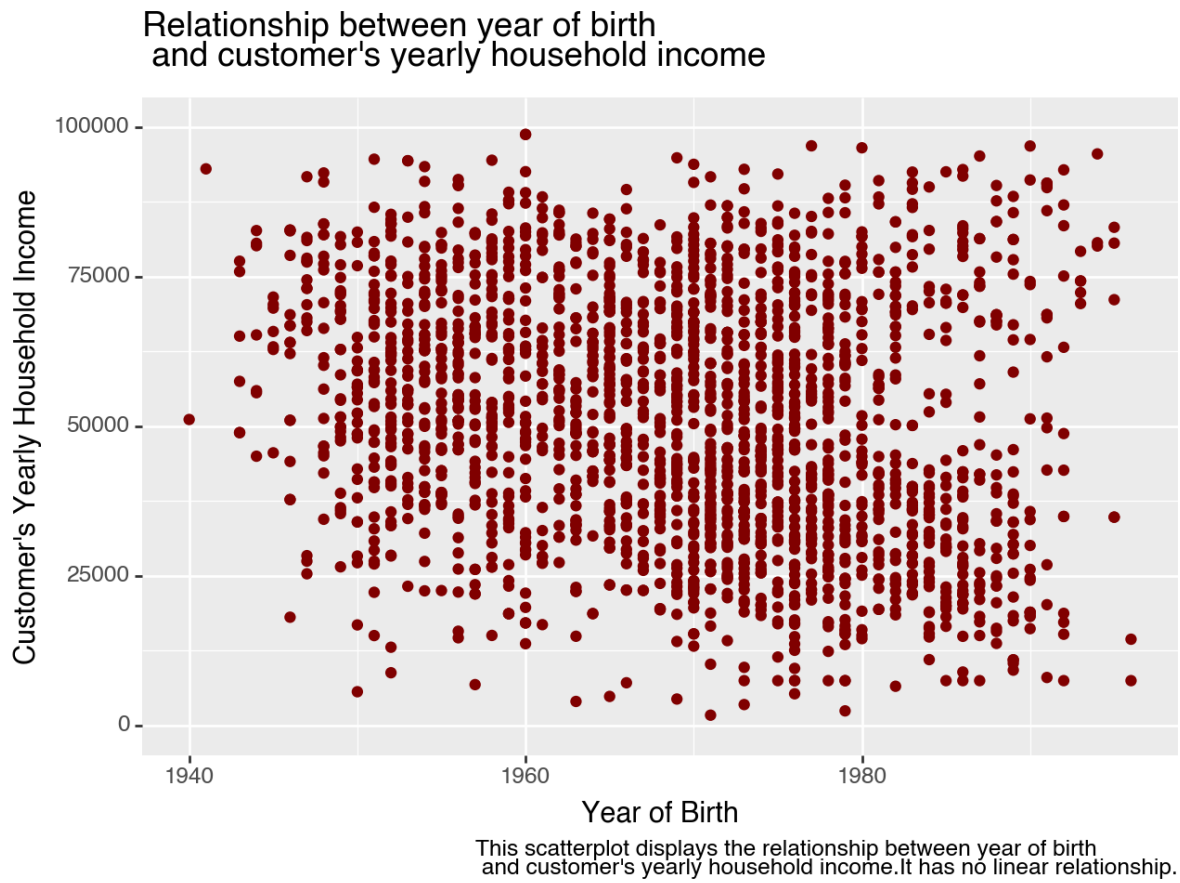
```
/opt/homebrew/lib/python3.9/site-packages/plotnine/layer.py:364: PlotnineWarning: geom_point
```



<Figure Size: (640 x 480)>

```
# create scatterplots
(p9.ggplot(df, p9.aes(x = 'Year_Birth', y = 'Income'))  +
 p9.geom_point(color = 'maroon') +
 p9.ylim(0,100000) +
 p9.labs(x = "Year of Birth", y = "Customer's Yearly Household Income",
        title= "Relationship between year of birth\n and customer's yearly household incom
          caption = "This scatterplot displays the relationship between year of birth\n an
          + "It has no linear relationship."))
```

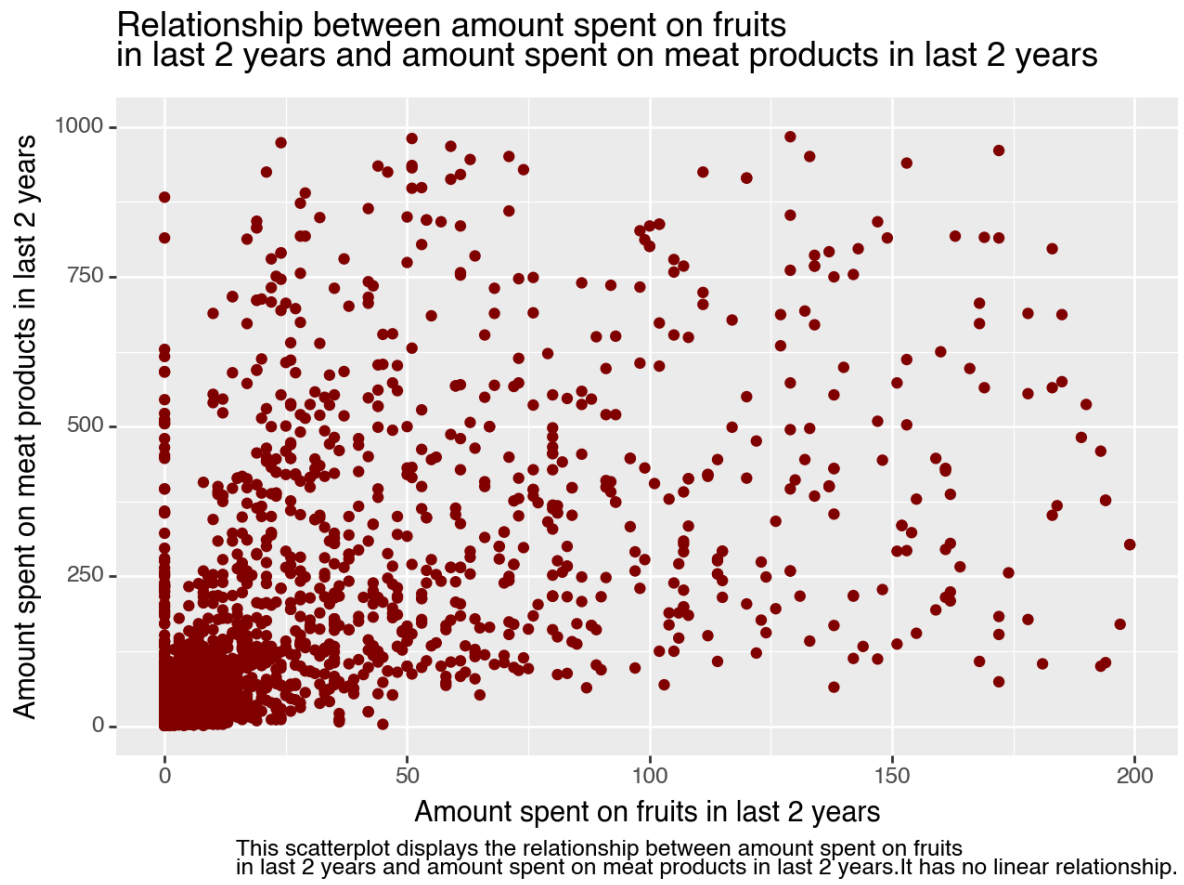
/opt/homebrew/lib/python3.9/site-packages/plotnine/layer.py:364: PlotnineWarning: geom_point



<Figure Size: (640 x 480)>

```
# create scatterplots
(p9.ggplot(df, p9.aes(x = 'MntFruits', y = 'MntMeatProducts'))  +
 p9.geom_point(color = 'maroon') +
 p9.ylim(1,1000) +
 p9.labs(x = "Amount spent on fruits in last 2 years", y = "Amount spent on meat products
        title= "Relationship between amount spent on fruits \nin last 2 years and amount s
          caption = "This scatterplot displays the relationship between amount spent on fr
          + "It has no linear relationship."))
```

/opt/homebrew/lib/python3.9/site-packages/plotnine/layer.py:364: PlotnineWarning: geom_point




```
<Figure Size: (640 x 480)>
```

```
data = pd.get_dummies(df, columns=['Education','Marital_Status'],drop_first = True)
data.head(5)

#remove date because it is string
# Acceted Last Campaign is our response
X = data.drop(['Dt_Customer'], axis = 1)
X.corr()
```

	Year_Birth	Income	Kidhome	Teenhome	Recency	MntWines	MntFru
Year_Birth	1.000000	-0.163295	0.237738	-0.362112	-0.015971	-0.164843	-0.0135
Income	-0.163295	1.000000	-0.428231	0.019285	-0.003111	0.578481	0.4302
Kidhome	0.237738	-0.428231	1.000000	-0.039485	0.010196	-0.497407	-0.3733
Teenhome	-0.362112	0.019285	-0.039485	1.000000	0.014764	0.004312	-0.1757

	Year_Birth	Income	Kidhome	Teenhome	Recency	MntWines	MntFru
Recency	-0.015971	-0.003111	0.010196	0.014764	1.000000	0.016332	-0.0051
MntWines	-0.164843	0.578481	-0.497407	0.004312	0.016332	1.000000	0.38589
MntFruits	-0.013542	0.430248	-0.373305	-0.175736	-0.005129	0.385892	1.00000
MntMeatProducts	-0.033823	0.584361	-0.439192	-0.260778	0.023177	0.568189	0.54676
MntFishProducts	-0.041316	0.438523	-0.388777	-0.204954	0.001007	0.397035	0.59306
MntSweetProducts	-0.021710	0.440532	-0.378014	-0.162794	0.025495	0.389731	0.57149
MntGoldProds	-0.059960	0.325073	-0.355095	-0.018315	0.018394	0.391604	0.39350
NumDealsPurchases	-0.065866	-0.082874	0.216913	0.386298	0.002236	0.008769	-0.1342
NumWebPurchases	-0.162366	0.388183	-0.372410	0.162368	-0.005518	0.553704	0.30234
NumCatalogPurchases	-0.126012	0.589090	-0.504706	-0.112207	0.024423	0.634306	0.48564
NumStorePurchases	-0.139229	0.530120	-0.502062	0.049556	-0.000109	0.640343	0.45990
NumWebVisitsMonth	0.120355	-0.552736	0.447273	0.130839	-0.019075	-0.321666	-0.4177
AcceptedCmp3	0.061001	-0.016063	0.015999	-0.042669	-0.032240	0.061460	0.01468
AcceptedCmp4	-0.070114	0.184615	-0.162201	0.038279	0.017631	0.373389	0.00663
AcceptedCmp5	0.018936	0.335032	-0.204660	-0.189961	0.000347	0.472909	0.20902
AcceptedCmp1	-0.012021	0.277071	-0.174339	-0.145058	-0.021036	0.351647	0.19244
AcceptedCmp2	-0.007857	0.087635	-0.081946	-0.015580	-0.001382	0.206319	-0.0099
Complain	-0.004631	-0.024902	0.037013	0.007784	0.005750	-0.036376	-0.0029
AcceptedLastCmp	0.020803	0.133302	-0.078076	-0.154189	-0.199899	0.246434	0.12305
Education_Basic	0.115537	-0.200604	0.055294	-0.120058	-0.003078	-0.139712	-0.0605
Education_Graduation	0.061987	0.019384	-0.002017	-0.025420	0.030405	-0.060141	0.11519
Education_Master	-0.074821	0.012022	0.012915	0.023358	-0.025955	0.036672	-0.0553
Education_PhD	-0.123004	0.080526	-0.043152	0.093275	-0.007680	0.158160	-0.0852
Marital_Status_Alone	0.012859	-0.012364	0.038300	0.010903	-0.023778	-0.013164	-0.0207
Marital_Status_Divorced	-0.068717	0.009056	-0.019509	0.054853	0.003127	0.021292	0.01027
Marital_Status_Married	0.044240	-0.016157	0.017733	0.007943	-0.019158	-0.012490	-0.0135
Marital_Status_Single	0.125229	-0.026008	0.015005	-0.100826	0.004444	-0.020356	0.01300
Marital_Status_Together	-0.053004	0.022422	0.010024	0.026119	0.020533	0.004322	-0.0153
Marital_Status_Widow	-0.163701	0.031801	-0.072045	0.048207	-0.001336	0.034659	0.02631
Marital_Status_YOLO	0.010497	-0.004546	-0.024759	0.027325	-0.047821	0.001502	-0.0176

Project D

LASSO Classification

```
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
import sklearn
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.model_selection import train_test_split

#standardize data
X = X.drop(['AcceptedLastCmp'], axis = 1)
y = data['AcceptedLastCmp']


scaler = StandardScaler()
scaler.fit(X)
X_pp = pd.DataFrame(scaler.transform(X), columns =['Year_Birth', 'Income', 'Kidhome', 'Tee
       'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Education_Basic', 'Education_Graduation',
       'Education_Master', 'Education_PhD', 'Marital_Status_Alone',
       'Marital_Status_Divorced', 'Marital_Status_Married',
       'Marital_Status_Single', 'Marital_Status_Together',
       'Marital_Status_Widow', 'Marital_Status_YOLO'])
```

```
from sklearn.linear_model import RidgeClassifierCV

#fit ridge classification
coefs_ridge = pd.DataFrame(data = None, columns = ['Year_Birth', 'Income', 'Kidhome', 'Tee
       'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Education_Basic', 'Education_Graduation',
       'Education_Master', 'Education_PhD', 'Marital_Status_Alone',
       'Marital_Status_Divorced', 'Marital_Status_Married',
       'Marital_Status_Single', 'Marital_Status_Together',
       'Marital_Status_Widow', 'Marital_Status_YOLO'])

alphas = np.arange(-2, 8, 0.1)
alphas = np.power(10, alphas)


model_ridge = RidgeClassifierCV(alphas = alphas).fit(X_pp, y)
print("alpha: ", model_ridge.alpha_)
```

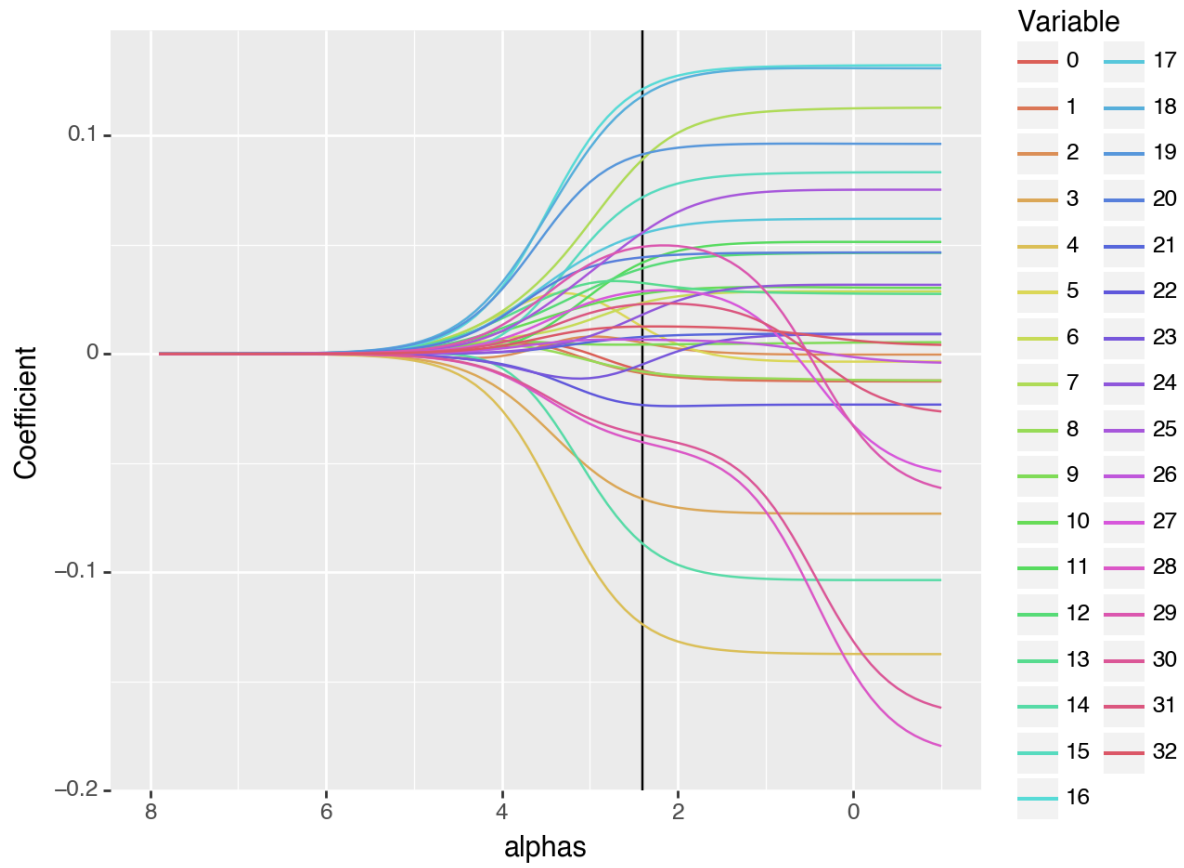
```
print("Coefficents: ", model_ridge.coef_)
print("Based on cross-validation, alpha=", model_ridge.alpha_, "is the best")
```

```
alpha:  251.18864315096027
Coefficents:  [[-0.00766182 -0.00891105  0.00506828 -0.0664298  -0.12389496  0.01231712
   0.02337978  0.08906701 -0.00817197  0.00423442  0.02747556  0.04179369
   0.03916608  0.03244353 -0.08691125  0.07172633  0.12124407  0.055037
   0.11811865  0.09137715  0.04421128  0.00775704 -0.02346739 -0.00482688
   0.01795495  0.05553376  0.00649319  0.02868643 -0.04048278  0.04903153
  -0.03719046  0.02278394  0.01250525]]
Based on cross-validation, alpha= 251.18864315096027 is the best
```

```
from sklearn.linear_model import RidgeClassifier

#fit ridge at best alpha
bestRidge = RidgeClassifier(alpha = model_ridge.alpha_).fit(X_pp, y)
```

```
#graph ridge classification
import math

coefs_ridge = pd.DataFrame()

for i in range(0, len(alphas)):
    temp_model = RidgeClassifierCV(alphas = alphas[i]).fit(X_pp, y)
    coefs_ridge = pd.concat([coefs_ridge, pd.Series(temp_model.coef_[0]).to_frame().T], ig


coefs_ridge['alphas'] = np.log10(alphas)
coefs_ridge_melt = pd.melt(coefs_ridge, id_vars = 'alphas',var_name = 'Variable', value_na

(p9.ggplot(coefs_ridge_melt, p9.aes(x = 'alphas', y = 'Coefficient', color = 'Variable'))
  p9.geom_line() + p9.xlim(8, -1))
```

```
/opt/homebrew/lib/python3.9/site-packages/plotnine/geoms/geom_path.py:98: PlotnineWarning: g
```



<Figure Size: (640 x 480)>

```
from sklearn.linear_model import LogisticRegressionCV
from sklearn.linear_model import LogisticRegression



#fit LASSO

coefs_lasso = pd.DataFrame(data = None, columns = ['Year_Birth', 'Income', 'Kidhome', 'Tee
       'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Education_Basic', 'Education_Graduation',
       'Education_Master', 'Education_PhD', 'Marital_Status_Alone',
```

```
        'Marital_Status_Divorced', 'Marital_Status_Married',
        'Marital_Status_Single', 'Marital_Status_Together',
        'Marital_Status_Widow', 'Marital_Status_YOLO'])

alphas = np.arange(-3, 0.3, 0.1)
alphas = np.power(10, alphas)

Lassocvmodel = LogisticRegressionCV(penalty = 'l1', solver = 'liblinear', Cs=alphas)
Lassocvmodel = Lassocvmodel.fit(X_pp, y)
print("Based on cross validation,")
print("The best alpha is")
print(Lassocvmodel.C_)
print("The coefficients are")
```

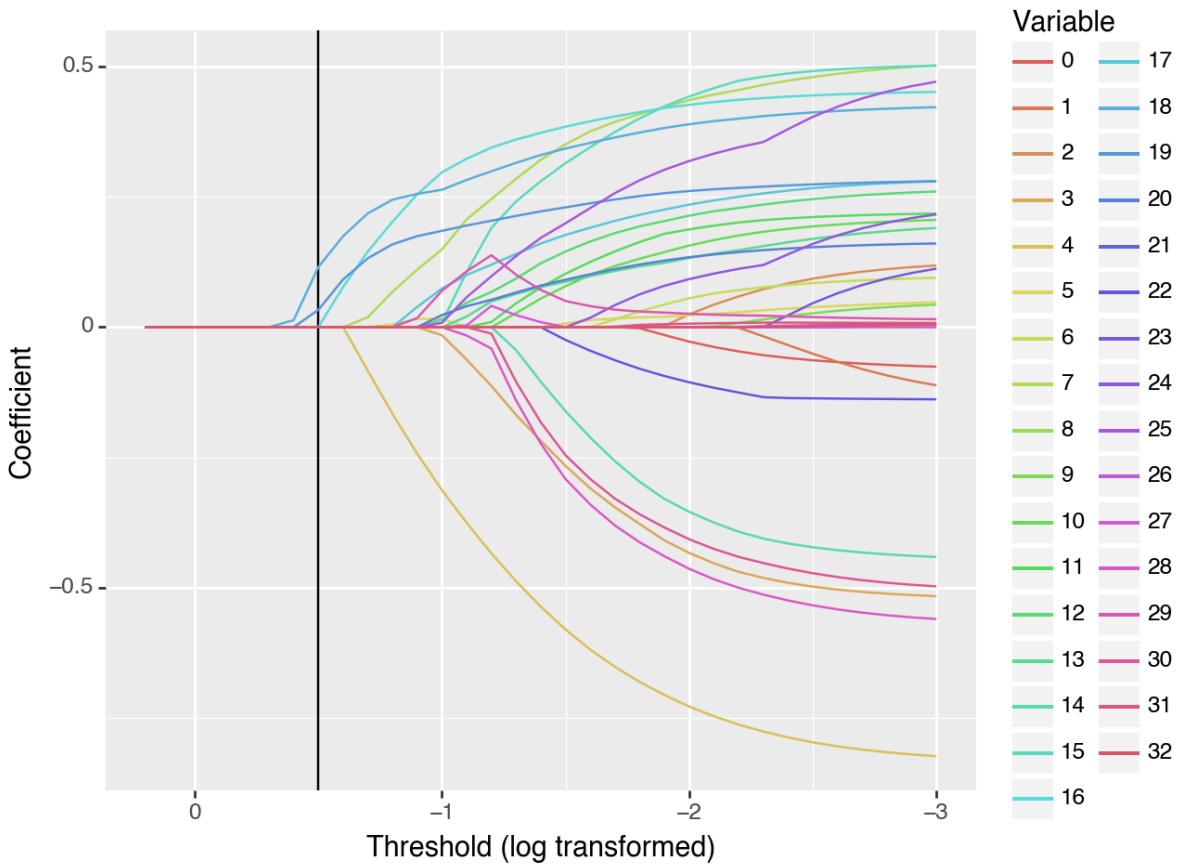
```
Based on cross validation,
The best alpha is
[0.31622777]
The coefficients are
```

```
#fit best lasso
bestLasso = LogisticRegression(penalty = 'l1', solver = 'liblinear', C=Lassocvmodel.C_[0])
bestLasso = bestLasso.fit(X_pp, y)
bestLasso.coef_
```

```
array([[-0.0537513 , -0.01757567,  0.07293286, -0.48108146, -0.77587533,
         0.03248612,  0.07678184,  0.46495109,  0.        ,  0.01461655,
         0.18309188,  0.20533359,  0.23521451,  0.15547739, -0.40569267,
         0.48058019,  0.43951778,  0.25667642,  0.40485997,  0.2696717 ,
         0.14775686,  0.        , -0.13433533,  0.        ,  0.11921204,
         0.35529035,  0.00197923,  0.        , -0.51313461,  0.02207345,
        -0.45261997,  0.0086923 ,  0.00093212]])
```

```
#create LASSO coefficient plots
from sklearn.linear_model import Lasso

coefs_lasso = pd.DataFrame()


for threshold in alphas:
  model = LogisticRegression(penalty = 'l1', solver = 'liblinear', C=threshold)
```

```
  tempmodel = model.fit(X_pp, y)
  coefs_lasso = pd.concat([coefs_lasso, pd.Series(tempmodel.coef_[0]).to_frame().T], ignor



low_lasso_plt = coefs_lasso

#create LASSO coefficient plotslow_lasso_plt = coefs_lasso
#low_lasso_plt.rename(columns={0:'Year_Birth', 1:'Income', 2:'Kidhome', 3:'Teenhome', 4:'R
#       6:'MntFruits', 7:'MntMeatProducts', 8:'MntFishProducts', 9:'MntSweetProducts',
#       10:'MntGoldProds', 11:'NumDealsPurchases', 12:'NumWebPurchases',
#       13:'NumCatalogPurchases', 14:'NumStorePurchases', 15:'NumWebVisitsMonth',
#       16:'AcceptedCmp3', 17:'AcceptedCmp4', 18:'AcceptedCmp5', 19:'AcceptedCmp1',
#       20:'AcceptedCmp2', 21:'Complain', 22:'Education_Basic', 23:'Education_Graduation',
#       24:'Education_Master', 25:'Education_PhD', 26:'Marital_Status_Alone',
#       27:'Marital_Status_Divorced', 28:'Marital_Status_Married',
#       29:'Marital_Status_Single', 30:'Marital_Status_Together',
#       31:'Marital_Status_Widow', 32:'Marital_Status_YOLO'}, inplace = True)


#low_lasso_plt['Threshold'] = alphas[::-1]
low_lasso_plt['Threshold'] = np.log10(alphas[::-1])

low_lasso_plt_melt = pd.melt(low_lasso_plt, id_vars = 'Threshold', var_name = 'Variable',

(p9.ggplot(low_lasso_plt_melt, p9.aes(x = 'Threshold', y = 'Coefficient', color = 'Variabl
  p9.labs(x = "Threshold (log transformed)") + p9.geom_line() + p9.scale_x_reverse())
```



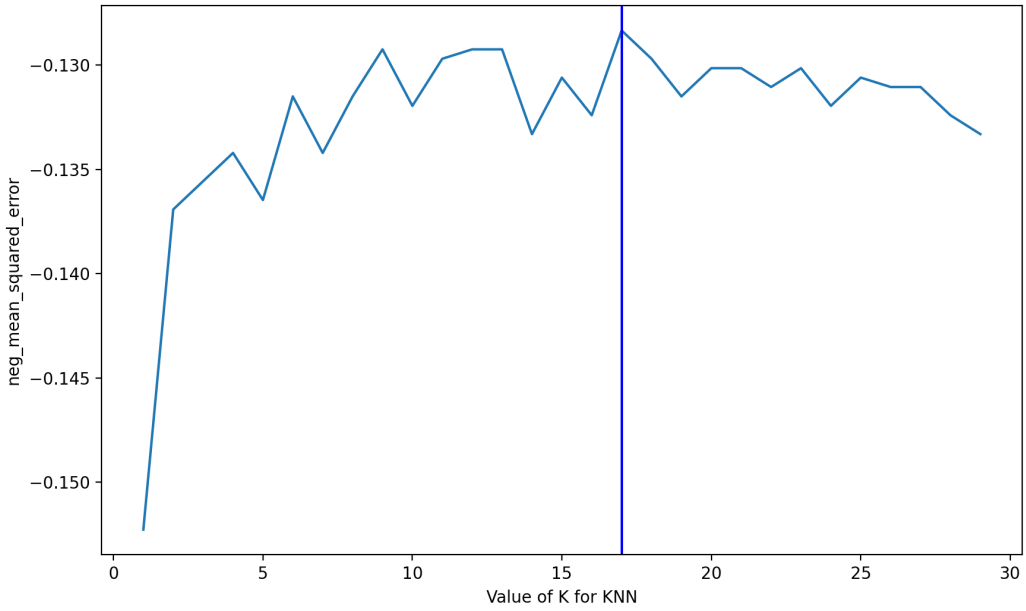
<Figure Size: (640 x 480)>

In LASSO model, whether consumers accepted the offer in the last campaign is depended on most factors in the dataset. There are four variables LASSO believes that are not important: MntFishProducts, Complain, Education_Graduation, and Marital_Status_Divorced.

KNN Classification

```
#perform KNN Classification

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import LeaveOneOut
```

```
cv = LeaveOneOut()

k_range = range(1, 30)
k_scores = []

for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_pp, y, cv=cv, scoring='neg_mean_squared_error')
    k_scores.append(scores.mean())

plt.figure(figsize=(10,6))

plt.plot(k_range, k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('neg_mean_squared_error')
plt.axvline(x = 17, color = 'b')
plt.show()
print("Using LeaveOneOut, k=17 is the best")
```




```
Using LeaveOneOut, k=17 is the best
```

```
#fit knn at best alpha
knnModel = KNeighborsClassifier(n_neighbors=17)
```

```
from sklearn.linear_model import LinearRegression

#modelLinear = LinearRegression().fit(X[['CRBI', 'Walks', 'Division_W', 'Hits', 'CRuns', '
#linearScores = cross_val_score(modelLinear, df1[['CRBI', 'Walks', 'Division_W', 'Hits', '
#meanLinear = np.mean(linearScores)
#print("For linear regression, the neg mean absolute error is")
#print(meanLinear)

ridgeScores = cross_val_score(bestRidge, X_pp, y, scoring='neg_mean_absolute_error',cv=cv)
meanRidge = np.mean(ridgeScores)
print("For ridge regression, the neg absolute error is")
print(meanRidge)


lassoScores = cross_val_score(bestLasso, X_pp, y, scoring='neg_mean_absolute_error',cv=cv)
meanLASSO = np.mean(lassoScores)
print("For LASSO, the neg mean absolute error is")
print(meanLASSO)


knnScore = cross_val_score(knnModel, X_pp, y, scoring='neg_mean_absolute_error',cv=cv)
meanKNN = np.mean(knnScore)
print("For kNN, the neg mean absolute error is")
print(meanKNN)
```

```
For ridge regression, the neg absolute error is
-0.12065070040668775
For LASSO, the neg mean absolute error is
-0.11296882060551287
For kNN, the neg mean absolute error is
-0.12833258020786262
```

The mean_absolute_error function computes mean absolute error, a risk metric corresponding to the expected value of the absolute error loss or l1-norm loss. Among ridge, LASSO, and KNN models, the the neg mean absolute error is actually pretty similar. However, LASSO model is a little bit better because its mean neg mean absolute error is closest to 0 compared to others. Thus, I would recommend the LASSO model