```
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_)
```

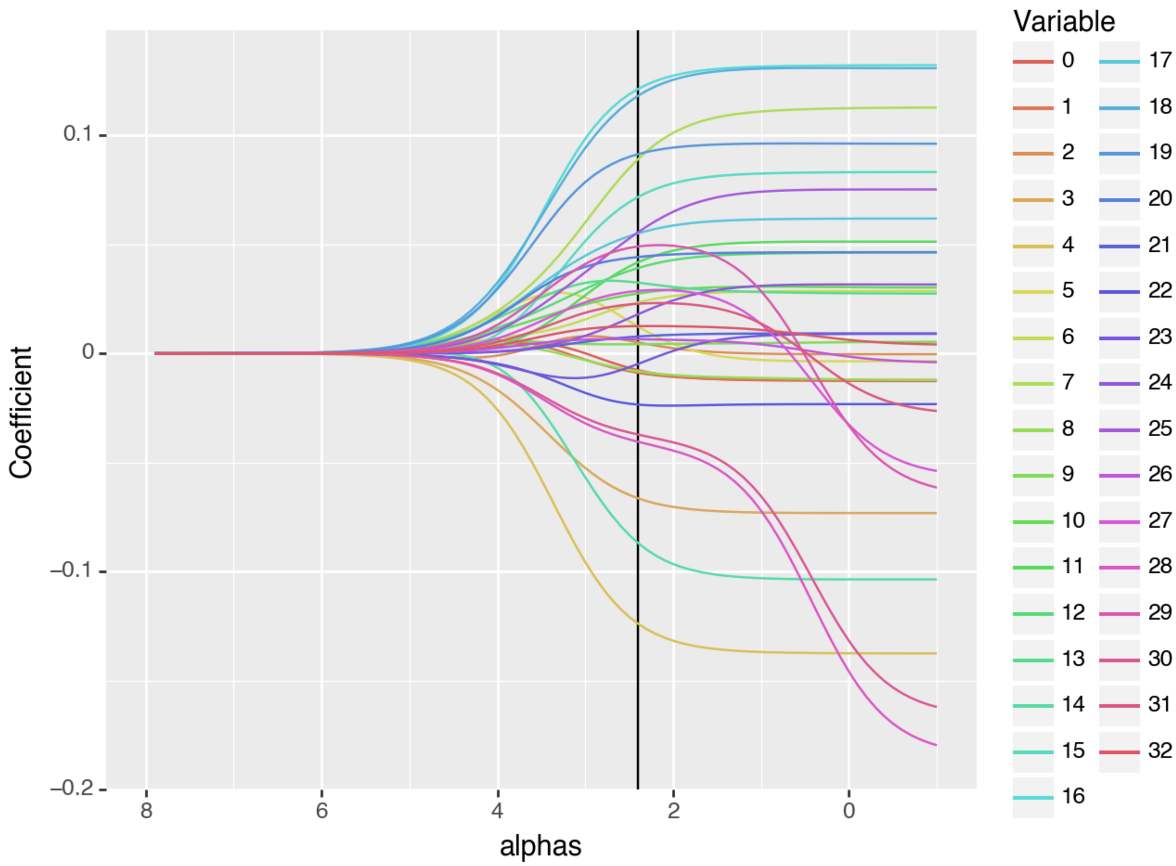
```
print("Coefficents: ", model_ridge.coef_)
print("Based on cross-validation, alpha=", model_ridge.alpha_, "is the best")
```

```
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")
```

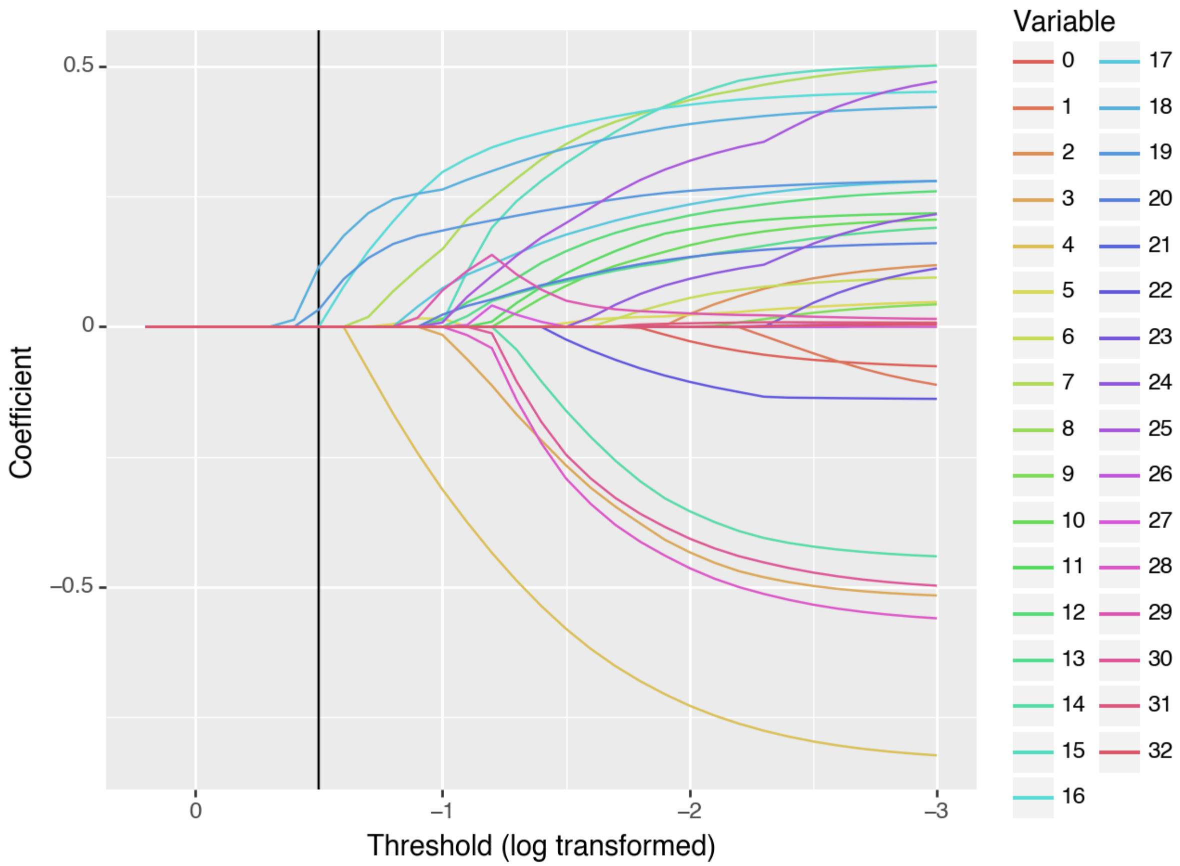
```
Based on cross validation,
The best alpha is
[0.31622777]
The coefficients are
```

```
#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())
```



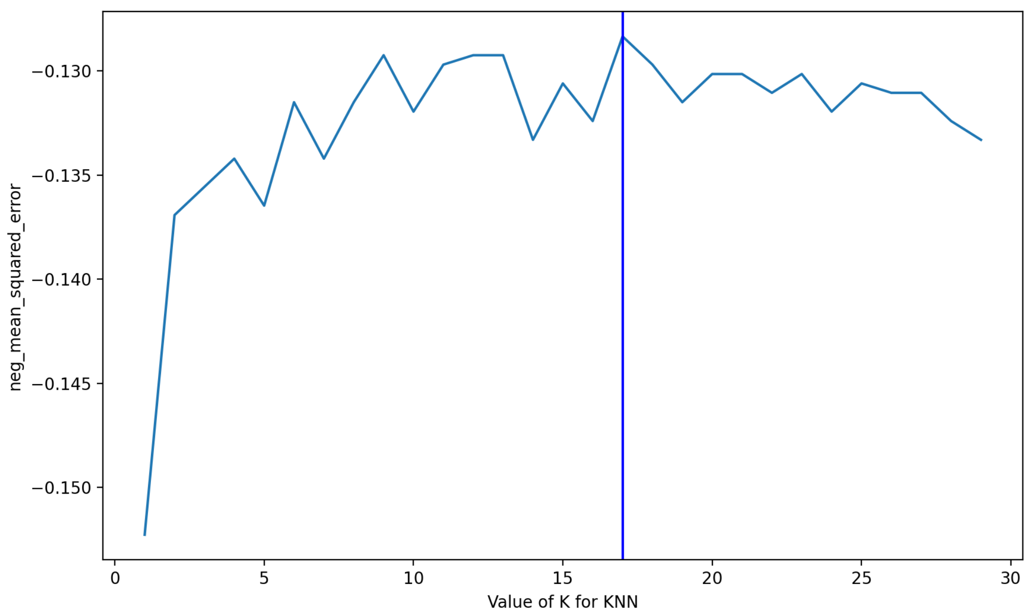

```
<Figure Size: (640 x 480)>
```

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