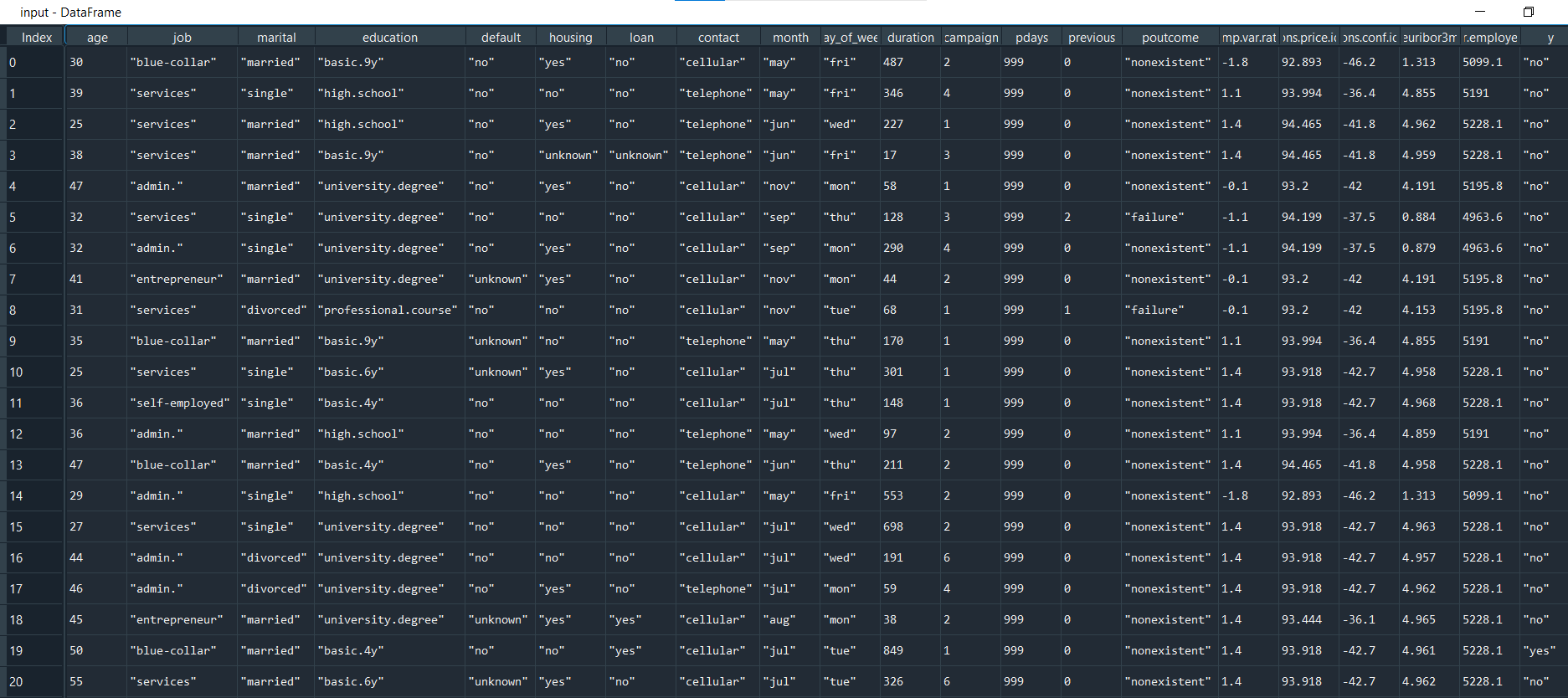
Sylvia Carroll - Project 4

The goal of this project is to predict whether or not people will become customers of a bank based on their personal info that they provided to telemarketers. The dataset contains 4119 samples of 20 inputs each. The training data were scaled, and Bernoulli sampling was used to split the data into train and test groups. Metrics that were calculated included accuracy, precision, recall, root mean square error, and R^2. Two models were used to examine the data: logistic regression and linear regression. Both of these models performed well. Then L2 regularization was used to optimize the linear model.

Here is what the first 20 columns of the input data look like.

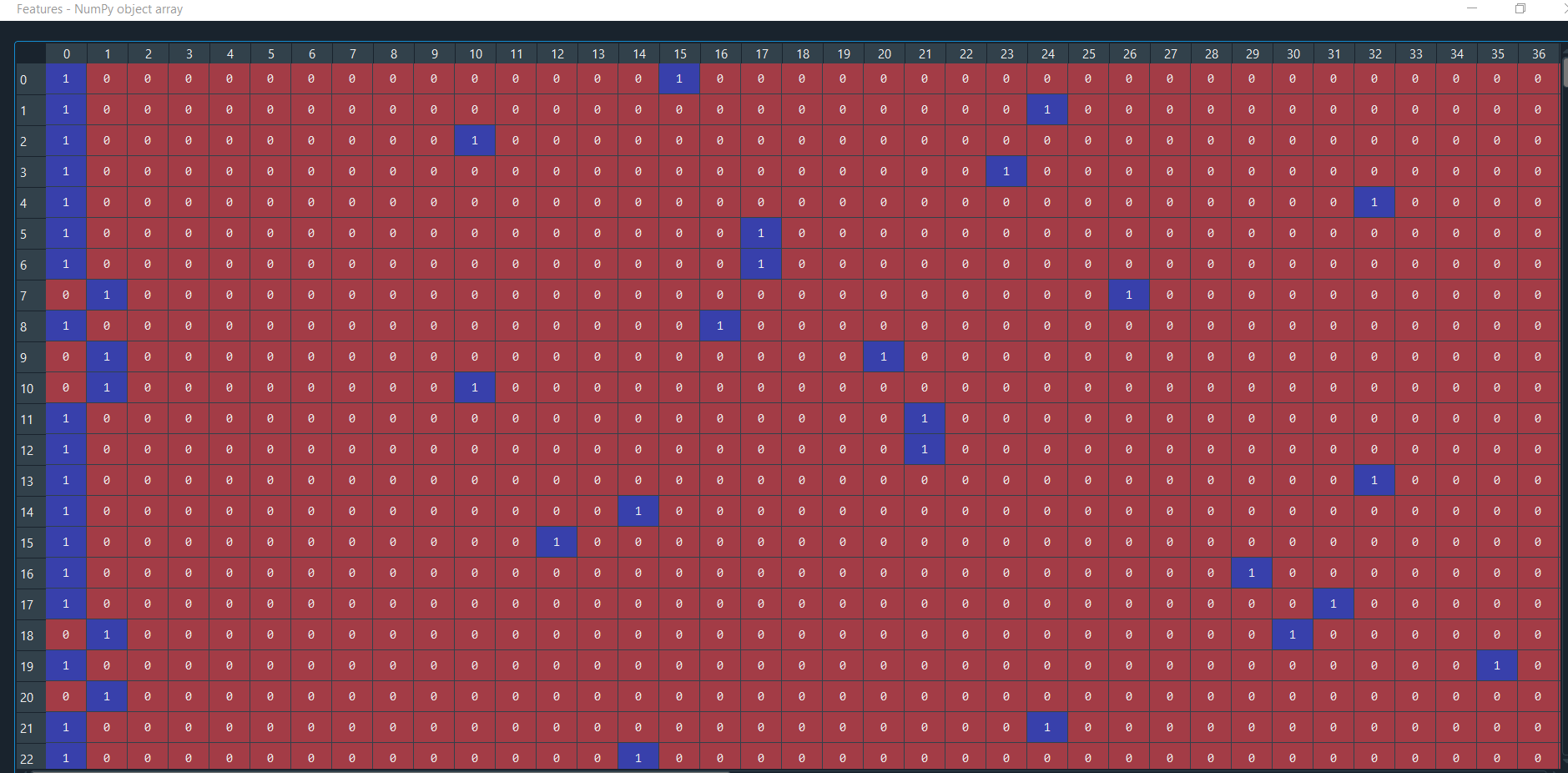


(4119, 21) # input.shape

(4119, 1313) # Features.shape

[[1. 0. 0. ... 0. 1. 0.]

[1. 0. 0. ... 0. 1. 0.]]



With the dummy variables total is 1313

print(logistic\_mod.intercept\_)

print(logistic\_mod.coef\_)

[-4.97417339]

[[ 2.73473109e-02 -2.69625290e-02 -3.87258560e-04 ... -1.10993961e-01

-1.20709756e+00 3.85423776e+00]]

probabilities = logistic\_mod.predict\_proba(X\_test)

print(probabilities[:15,:])

[[9.99742216e-01 2.57783989e-04]

[9.99753154e-01 2.46846165e-04]

[2.39925438e-03 9.97600746e-01]

[9.99692107e-01 3.07892825e-04]

[9.99715385e-01 2.84615437e-04]

[9.99608526e-01 3.91474414e-04]

[9.99663963e-01 3.36036521e-04]

[9.99664897e-01 3.35102621e-04]

[9.99733426e-01 2.66574456e-04]

[9.99654740e-01 3.45260263e-04]

[9.99591630e-01 4.08369747e-04]

[4.10057897e-03 9.95899421e-01]

[9.99650185e-01 3.49815128e-04]

[9.99692115e-01 3.07884535e-04]

[9.99685487e-01 3.14512522e-04]]

The first column is the probability of a score of

0 and the second column is the probability of a score of

1. Notice that for most, but not all cases, the probability of a score of

0 is higher than

1.

def score\_model(probs, threshold):

return np.array([1 if x > threshold else 0 for x in probs[:,1]])

scores = score\_model(probabilities, 0.5)

print(np.array(scores[:15]))

print(y\_test[:15])

[0 0 1 0 0 0 0 0 0 0 0 1 0 0 0]

['"no"' '"no"' '"yes"' '"no"' '"no"' '"no"' '"no"' '"no"' '"no"' '"no"'

'"no"' '"yes"' '"no"' '"no"' '"no"']

def print\_metrics(labels, scores):

metrics = sklm.precision\_recall\_fscore\_support(labels, scores)

conf = sklm.confusion\_matrix(labels, scores)

print(' Confusion matrix')

print(' Score positive Score negative')

print('Actual positive %6d' % conf[0,0] + ' %5d' % conf[0,1])

print('Actual negative %6d' % conf[1,0] + ' %5d' % conf[1,1])

print('')

print('Accuracy %0.2f' % sklm.accuracy\_score(labels, scores))

print(' ')

print(' Positive Negative')

print('Num case %6d' % metrics[3][0] + ' %6d' % metrics[3][1])

print('Precision %6.2f' % metrics[0][0] + ' %6.2f' % metrics[0][1])

print('Recall %6.2f' % metrics[1][0] + ' %6.2f' % metrics[1][1])

print('F1 %6.2f' % metrics[2][0] + ' %6.2f' % metrics[2][1])

print\_metrics(y\_test, scores)

This function calculates various metrics based on the data. The logistic model performed very well, with an accuracy of 100%. Even when I changed the threshold to 0.45, 0.4, 0.3, 0.35 and 0.25, the same metrics were obtained.

Confusion matrix

Score positive Score negative

Actual positive 270 0

Actual negative 0 30

Accuracy 1.00

Positive Negative

Num case 270 30

Precision 1.00 1.00

Recall 1.00 1.00

F1 1.00 1.00

#weighted model

logistic\_mod = linear\_model.LogisticRegression(class\_weight = {0:0.45, 1:0.55})

logistic\_mod.fit(X\_train, y\_train)

probabilities = logistic\_mod.predict\_proba(X\_test)

print(probabilities[:15,:])

scores = score\_model(probabilities, 0.5)

print\_metrics(y\_test, scores)

plot\_auc(y\_test, probabilities)

def test\_threshold(probs, labels, threshold):

scores = score\_model(probs, threshold)

print('')

print('For threshold = ' + str(threshold))

print\_metrics(labels, scores)

thresholds = [0.45, 0.40, 0.35, 0.3, 0.25]

for t in thresholds:

test\_threshold(probabilities, y\_test, t)

For threshold = 0.45

Confusion matrix

Score positive Score negative

Actual positive 270 0

Actual negative 0 30

Accuracy 1.00

Positive Negative

Num case 270 30

Precision 1.00 1.00

Recall 1.00 1.00

F1 1.00 1.00

For threshold = 0.4

Confusion matrix

Score positive Score negative

Actual positive 270 0

Actual negative 0 30

Accuracy 1.00

Positive Negative

Num case 270 30

Precision 1.00 1.00

Recall 1.00 1.00

F1 1.00 1.00

For threshold = 0.35

Confusion matrix

Score positive Score negative

Actual positive 270 0

Actual negative 0 30

Accuracy 1.00

Positive Negative

Num case 270 30

Precision 1.00 1.00

Recall 1.00 1.00

F1 1.00 1.00

For threshold = 0.3

Confusion matrix

Score positive Score negative

Actual positive 270 0

Actual negative 0 30

Accuracy 1.00

Positive Negative

Num case 270 30

Precision 1.00 1.00

Recall 1.00 1.00

F1 1.00 1.00

For threshold = 0.25

Confusion matrix

Score positive Score negative

Actual positive 270 0

Actual negative 0 30

Accuracy 1.00

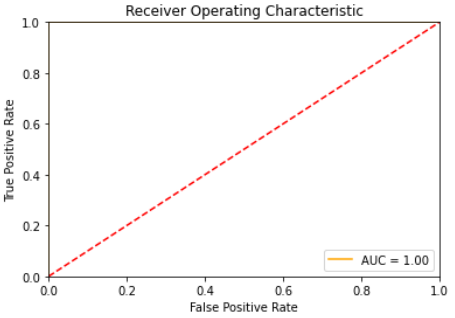
Positive Negative

Num case 270 30

Precision 1.00 1.00

Recall 1.00 1.00

F1 1.00 1.00



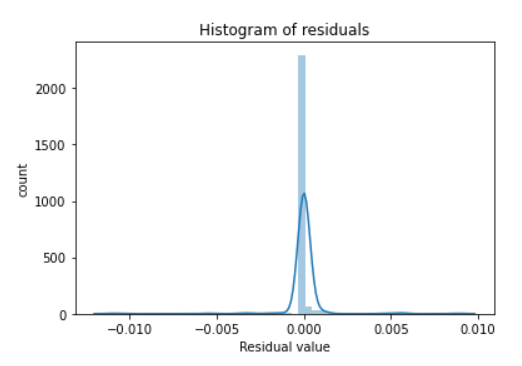
Mean Square Error = 1.1696919141352374e-06

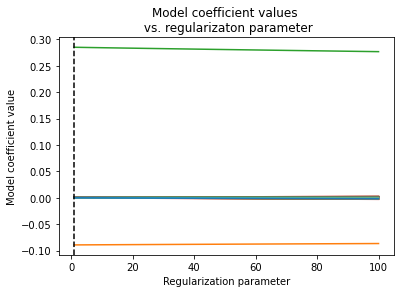
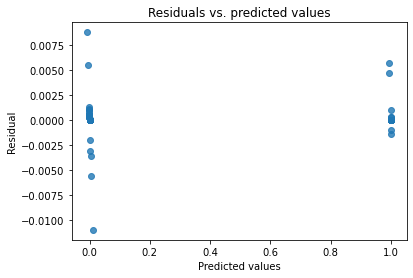
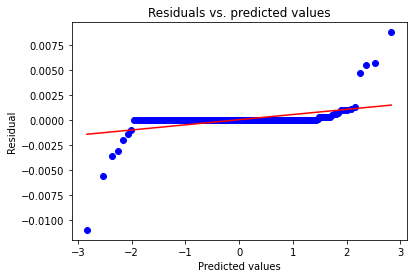
Root Mean Square Error = 0.0010815229605215218

Mean Absolute Error = 0.00020862165652869359

Median Absolute Error = 6.661338147750939e-16

R^2 = 0.9999870034231763





lin\_mod\_l2 = linear\_model.Ridge(alpha = out\_l2[0])

lin\_mod\_l2.fit(X\_train, y\_train)

y\_score\_l2 = lin\_mod\_l2.predict(X\_test)

print\_metrics(y\_test, y\_score\_l2)

hist\_resids(y\_test, y\_score\_l2)

resid\_qq(y\_test, y\_score\_l2)

resid\_plot(y\_test, y\_score\_l2)

Here are the metrics using the L2 parameter.

Mean Square Error = 2.589795877790281e-08

Root Mean Square Error = 0.00016092842750087011

Mean Absolute Error = 9.302527462738944e-05

Median Absolute Error = 3.495279468555823e-05

R^2 = 0.9999997122449025

