In [1]:
```
import numpy as np
import pandas as pd
import logreg_skeleton as l
```

In [24]:
```
#load data
X_train = np.loadtxt('X_train.txt',delimiter=',')
X_val = np.loadtxt('X_val.txt',delimiter=',')
y_train = np.loadtxt('y_train.txt',delimiter=',')
y_val = np.loadtxt('y_val.txt',delimiter=',')
```

In [26]:
```
MIN = np.amin(X_train,axis=0)
MAX = np.amax(X_train,axis=0)
X_normalized = (X_val - MIN)/(MAX - MIN)
bias = np.ones(X_val.shape[0])
X_val = np.c_[X_normalized, bias]
y_val[y_val==0] = -1
```

In [19]:
```
optimal_theta = l.fit_logistic_reg(X_train, y_train, l.f_objective, l2_param=1
).x
```

In [21]:
```
optimal_theta
```

Out[21]:
```
array([ 0.00098726,  0.00086963,  0.00030947,  0.02207397,  0.00024182,
        0.00074032,  0.00011984,  0.00085947,  0.0009058 , -0.01224862,
        0.00195631,  0.00023594,  0.00459852,  0.00012709,  0.00047898,
        0.00117242,  0.00037251, -0.00044843, -0.00044867, -0.00069619,
        0.00178109])
```

In [27]:
```
l.f_objective(optimal_theta,X_val,y_val)
```

Out[27]:
```
0.6925989795847987
```

In [28]:

```
l.fit_logistic_reg(X_train, y_train, l.f_objective, l2_param=1)
```

```
Out[28]:       fun: 0.6924566759413383
         hess_inv: array([[ 9.74191918e-01, -2.87436028e-02, -3.10097113e-02,
                 -2.74374966e-02, -2.58121396e-02, -2.86268776e-02,
                 -2.54308372e-02, -2.75374736e-02, -3.08876812e-02,
                 -2.92462592e-02, -2.56223249e-02, -3.18055373e-02,
                 -2.38610895e-02, -2.92923633e-02, -2.82271810e-02,
                 -2.89203278e-02, -2.56657268e-02, -3.14126209e-02,
                 -2.40546588e-02, -2.53970063e-02, -5.48122290e-02],
                [-2.87436028e-02,  9.67946460e-01, -3.47079621e-02,
                 -2.68344191e-02, -2.88792216e-02, -3.19450828e-02,
                 -2.84739527e-02, -3.07026910e-02, -3.44483459e-02,
                 -3.49497012e-02, -2.83639326e-02, -3.55968387e-02,
                 -2.59055142e-02, -3.28011918e-02, -3.15435785e-02,
                 -3.21982411e-02, -2.86925963e-02, -3.52766817e-02,
                 -2.70328081e-02, -2.85791969e-02, -6.11016639e-02],
                [-3.10097113e-02, -3.47079621e-02,  9.62041531e-01,
                 -1.72713516e-02, -3.15685286e-02, -3.46593241e-02,
                 -3.11916223e-02, -3.32286385e-02, -3.73148549e-02,
                 -4.51461869e-02, -3.00580727e-02, -3.89478452e-02,
                 -2.58716021e-02, -3.59408073e-02, -3.43629693e-02,
                 -3.47001630e-02, -3.12924834e-02, -3.89723522e-02,
                 -2.99237454e-02, -3.17553486e-02, -6.60923825e-02],
                [-2.74374966e-02, -2.68344191e-02, -1.72713516e-02,
                  6.28604523e-01, -1.53922799e-02, -2.47294592e-02,
                 -1.32042744e-02, -2.62422317e-02, -2.84822124e-02,
                  1.87322443e-01, -4.31188388e-02, -1.78592832e-02,
                 -8.68793523e-02, -1.48780872e-02, -2.02927479e-02,
                 -3.18130540e-02, -1.74025975e-02, -6.64611568e-03,
                 -3.32669069e-03, -3.15784211e-05, -5.32295123e-02],
                [-2.58121396e-02, -2.88792216e-02, -3.15685286e-02,
                 -1.53922799e-02,  9.73759626e-01, -2.88321767e-02,
                 -2.59218084e-02, -2.76485106e-02, -3.10457733e-02,
                 -3.69418534e-02, -2.50672800e-02, -3.23722691e-02,
                 -2.16976792e-02, -2.98696848e-02, -2.85736752e-02,
                 -2.88873207e-02, -2.60177236e-02, -3.23593006e-02,
                 -2.48416990e-02, -2.63503939e-02, -5.49983316e-02],
                [-2.86268776e-02, -3.19450828e-02, -3.46593241e-02,
                 -2.47294592e-02, -2.88321767e-02,  9.68151408e-01,
                 -2.84389689e-02, -3.05956389e-02, -3.43337865e-02,
                 -3.60820944e-02, -2.81561438e-02, -3.55453296e-02,
                 -2.54407435e-02, -3.27628945e-02, -3.14721199e-02,
                 -3.20612699e-02, -2.86337045e-02, -3.52893824e-02,
                 -2.70527626e-02, -2.86203371e-02, -6.08828993e-02],
                [-2.54308372e-02, -2.84739527e-02, -3.11916223e-02,
                 -1.32042744e-02, -2.59218084e-02, -2.84389689e-02,
                  9.74381851e-01, -2.72575046e-02, -3.06121581e-02,
                 -3.76471653e-02, -2.46060065e-02, -3.19854656e-02,
                 -2.10252780e-02, -2.95215867e-02, -2.82072674e-02,
                 -2.84544760e-02, -2.56900195e-02, -3.20345708e-02,
                 -2.46021965e-02, -2.61157079e-02, -5.42148111e-02],
                [-2.75374736e-02, -3.07026910e-02, -3.32286385e-02,
                 -2.62422317e-02, -2.76485106e-02, -3.05956389e-02,
                 -2.72575046e-02,  9.70590512e-01, -3.29958958e-02,
                 -3.31433609e-02, -2.71985159e-02, -3.40782391e-02,
                 -2.49127613e-02, -3.13995319e-02, -3.02047318e-02,
                 -3.08488240e-02, -2.74731526e-02, -3.37545673e-02,
                 -2.58637265e-02, -2.73377035e-02, -5.85297655e-02],
                [-3.08876812e-02, -3.44483459e-02, -3.73148549e-02,
                 -2.84822124e-02, -3.10457733e-02, -3.43337865e-02,
                 -3.06121581e-02, -3.29958958e-02,  9.62977694e-01,
                 -3.77835378e-02, -3.04632837e-02, -3.82686081e-02,
                 -2.77724931e-02, -3.52648656e-02, -3.39064889e-02,
                 -3.45989201e-02, -3.08430493e-02, -3.79356145e-02,
                 -2.90722355e-02, -3.07386888e-02, -6.56645374e-02],
                [-2.92462592e-02, -3.49497012e-02, -4.51461869e-02,
```

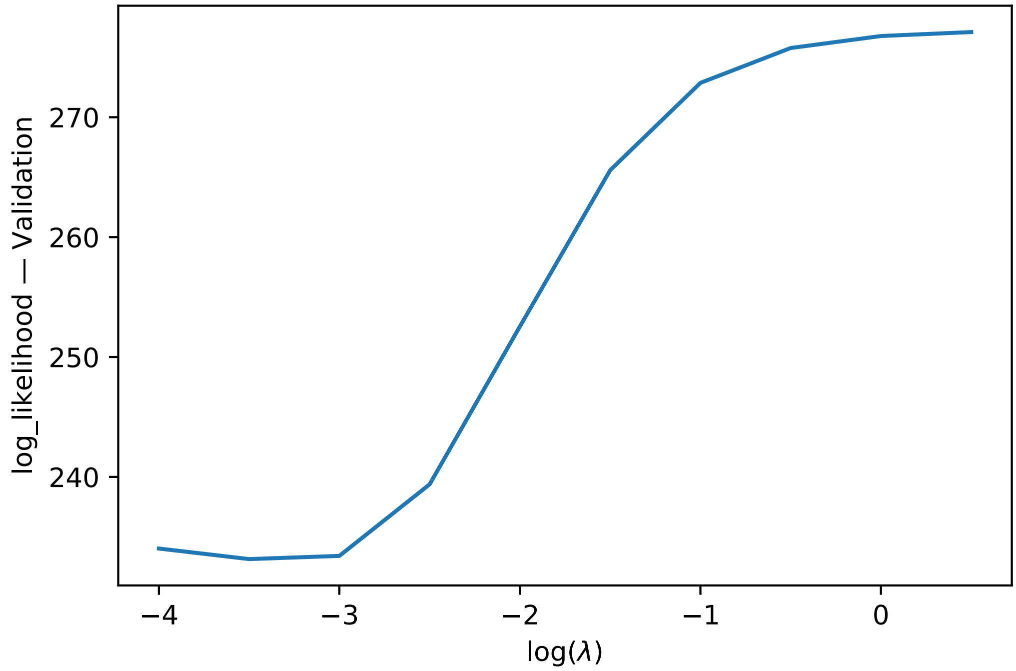
```
 1.87322443e-01, -3.69418534e-02, -3.60820944e-02,
-3.76471653e-02, -3.31433609e-02, -3.77835378e-02,
 8.27467769e-01, -1.89014813e-02, -4.62251478e-02,
 1.22050655e-02, -4.35737176e-02, -3.81894889e-02,
-3.20921761e-02, -3.53931149e-02, -5.26706122e-02,
-4.14578118e-02, -4.59940261e-02, -6.53339396e-02],
[-2.56223249e-02, -2.83639326e-02, -3.00580727e-02,
-4.31188388e-02, -2.50672800e-02, -2.81561438e-02,
-2.46060065e-02, -2.71985159e-02, -3.04632837e-02,
-1.89014813e-02,  9.73825340e-01, -3.08363533e-02,
-2.65435814e-02, -2.83269034e-02, -2.75731819e-02,
-2.87616672e-02, -2.50224134e-02, -2.99453815e-02,
-2.28493086e-02, -2.39629059e-02, -5.41837163e-02],
[-3.18055373e-02, -3.55968387e-02, -3.89478452e-02,
-1.78592832e-02, -3.23722691e-02, -3.55453296e-02,
-3.19854656e-02, -3.40782391e-02, -3.82686081e-02,
-4.62251478e-02, -3.08363533e-02,  9.60059501e-01,
-2.65382991e-02, -3.68577290e-02, -3.52397758e-02,
-3.55912936e-02, -3.20908610e-02, -3.99595401e-02,
-3.06817624e-02, -3.25561852e-02, -6.77849683e-02],
[-2.38610895e-02, -2.59055142e-02, -2.58716021e-02,
-8.68793523e-02, -2.16976792e-02, -2.54407435e-02,
-2.10252780e-02, -2.49127613e-02, -2.77724931e-02,
 1.22050655e-02, -2.65435814e-02, -2.65382991e-02,
 9.66906870e-01, -2.41607388e-02, -2.43502276e-02,
-2.69304406e-02, -2.19524559e-02, -2.42330688e-02,
-1.82407695e-02, -1.86308186e-02, -4.97691307e-02],
[-2.92923633e-02, -3.28011918e-02, -3.59408073e-02,
-1.48780872e-02, -2.98696848e-02, -3.27628945e-02,
-2.95215867e-02, -3.13995319e-02, -3.52648656e-02,
-4.35737176e-02, -2.83269034e-02, -3.68577290e-02,
-2.41607388e-02,  9.65980140e-01, -3.24998821e-02,
-3.27740769e-02, -2.96005202e-02, -3.69248969e-02,
-2.83594823e-02, -3.01076092e-02, -6.24520502e-02],
[-2.82271810e-02, -3.15435785e-02, -3.43629693e-02,
-2.02927479e-02, -2.85736752e-02, -3.14721199e-02,
-2.82072674e-02, -3.02047318e-02, -3.39064889e-02,
-3.81894889e-02, -2.75731819e-02, -3.52397758e-02,
-2.43502276e-02, -3.24998821e-02,  9.68851134e-01,
-3.16008666e-02, -2.83520760e-02, -3.51164716e-02,
-2.69410134e-02, -2.85431651e-02, -6.00931354e-02],
[-2.89203278e-02, -3.21982411e-02, -3.47001630e-02,
-3.18130540e-02, -2.88873207e-02, -3.20612699e-02,
-2.84544760e-02, -3.08488240e-02, -3.45989201e-02,
-3.20921761e-02, -2.87616672e-02, -3.55912936e-02,
-2.69304406e-02, -3.27740769e-02, -3.16008666e-02,
 9.67588791e-01, -2.87299985e-02, -3.51173517e-02,
-2.68861094e-02, -2.83756264e-02, -6.14064266e-02],
[-2.56657268e-02, -2.86925963e-02, -3.12924834e-02,
-1.74025975e-02, -2.60177236e-02, -2.86337045e-02,
-2.56900195e-02, -2.74731526e-02, -3.08430493e-02,
-3.53931149e-02, -2.50224134e-02, -3.20908610e-02,
-2.19524559e-02, -2.96005202e-02, -2.83520760e-02,
-2.87299985e-02,  9.74190417e-01, -3.20118510e-02,
-2.45644515e-02, -2.60357221e-02, -5.46555067e-02],
[-3.14126209e-02, -3.52766817e-02, -3.89723522e-02,
-6.64611568e-03, -3.23593006e-02, -3.52893824e-02,
-3.20345708e-02, -3.37545673e-02, -3.79356145e-02,
-5.26706122e-02, -2.99453815e-02, -3.99595401e-02,
-2.42330688e-02, -3.69248969e-02, -3.51164716e-02,
-3.51173517e-02, -3.20118510e-02,  9.59675001e-01,
-3.10172720e-02, -3.30200709e-02, -6.71094686e-02],
[-2.40546588e-02, -2.70328081e-02, -2.99237454e-02,
-3.32669069e-03, -2.48416990e-02, -2.70527626e-02,
-2.46021965e-02, -2.58637265e-02, -2.90722355e-02,
```

```
        -4.14578118e-02, -2.28493086e-02, -3.06817624e-02,
        -1.82407695e-02, -2.83594823e-02, -2.69410134e-02,
        -2.68861094e-02, -2.45644515e-02, -3.10172720e-02,
         9.76133484e-01, -2.54245831e-02, -5.14160499e-02],
       [-2.53970063e-02, -2.85791969e-02, -3.17553486e-02,
        -3.15784211e-05, -2.63503939e-02, -2.86203371e-02,
        -2.61157079e-02, -2.73377035e-02, -3.07386888e-02,
        -4.59940261e-02, -2.39629059e-02, -3.25561852e-02,
        -1.86308186e-02, -3.01076092e-02, -2.85431651e-02,
        -2.83756264e-02, -2.60357221e-02, -3.30200709e-02,
        -2.54245831e-02,  9.72882584e-01, -5.43364294e-02],
       [-5.48122290e-02, -6.11016639e-02, -6.60923825e-02,
        -5.32295123e-02, -5.49983316e-02, -6.08828993e-02,
        -5.42148111e-02, -5.85297655e-02, -6.56645374e-02,
        -6.53339396e-02, -5.41837163e-02, -6.77849683e-02,
        -4.97691307e-02, -6.24520502e-02, -6.00931354e-02,
        -6.14064266e-02, -5.46555067e-02, -6.71094686e-02,
        -5.14160499e-02, -5.43364294e-02,  8.83513386e-01]])
      jac: array([ 7.45058060e-08,  1.49011612e-08,  1.49011612e-08, -1.71363
354e-07,
        1.34110451e-07,  0.00000000e+00, -6.70552254e-08, -7.45058060e-09,
        7.45058060e-09,  2.30967999e-07, -8.94069672e-08, -7.45058060e-08,
        3.72529030e-08,  2.23517418e-08,  2.23517418e-08,  4.47034836e-08,
        0.00000000e+00,  0.00000000e+00,  5.96046448e-08,  6.70552254e-08,
        1.49011612e-08])
  message: 'Optimization terminated successfully.'
     nfev: 161
      nit: 3
     njev: 7
   status: 0
  success: True
        x: array([ 0.00098726,  0.00086963,  0.00030947,  0.02207397,  0.0002
4182,
        0.00074032,  0.00011984,  0.00085947,  0.0009058 , -0.01224862,
        0.00195631,  0.00023594,  0.00459852,  0.00012709,  0.00047898,
        0.00117242,  0.00037251, -0.00044843, -0.00044867, -0.00069619,
        0.00178109])
```

3.3.4

In [43]:
```
l2_param_list = np.arange(-4,1,0.5)
loss_list = []
for l2_param in l2_param_list:
    l2 = 10 **l2_param
    opt_theta = l.fit_logistic_reg(X_train, y_train, l.f_objective, l2_param=l
2).x
    loss = l.f_objective(opt_theta,X_val,y_val,l2_param=0) * len(y_val)
    loss_list.append(loss)
```

In [717]:
```
import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'svg'
plt.plot(l2_param_list,loss_list)
plt.xlabel('log($\lambda$)')
plt.ylabel("log_likelihood — Validation")
plt.savefig('1.png')
plt.show()
```



- l2_param = 0.001 minimizes the log_likelihood

3.3.5

In [49]:
```
opt_theta = l.fit_logistic_reg(X_train, y_train, l.f_objective, l2_param=0.001
).x
```

In [697]:
```
y_pred = 1/(1+ np.exp(-np.dot(X_val,opt_theta)))
```

In [303]:
```
y_pred = np.zeros(len(y_val))
for i in range(len(y_pred)):
    y_pred[i] = 1/(1+ np.exp(-np.dot(opt_theta,X_val[i])))
```

In [702]:
```
#original table
import pandas as pd
df = pd.DataFrame(columns=['y_val','y_pred'])

for idx in range(len(y_val)):
    df.loc[idx] = [y_val[idx],y_pred[idx]]
```

In [703]:
```
bins1 = np.arange(0,1.1,0.1)
labels1 = np.arange(0.05,1,0.1)
bins2 = np.arange(-0.05,1.1,0.1)
labels2 =np.arange(0.1,1.1,0.1)
labels2 = np.append(0.025,labels2)
labels2
```

Out[703]:
```
array([0.025, 0.1  , 0.2  , 0.3  , 0.4  , 0.5  , 0.6  , 0.7  , 0.8  ,
       0.9  , 1.   ])
```

In [704]:
```
# pd.value_counts(pd.cut(df['y_bin'],bins1))
df['y_bin'] = pd.cut(df['y_pred'], bins=bins1, labels=labels)
```

In [705]:
```
g = df.groupby(["y_bin", "y_val"]).count()
g = g.fillna(0)
```

In [706]:
```
g1 = g.groupby(level=[0]).apply(lambda g: g/g.sum())
g1 = g1.iloc[g1.index.get_level_values('y_val')==1]
g1 = g1.reset_index()

g1['y_bin'] = g1['y_bin'].astype('float')
g1.rename(index=str, columns={"y_pred": "percentage"})
```

Out[706]:

	y_bin	y_val	percentage
0	0.05	1.0	0.000000
1	0.15	1.0	0.454545
2	0.25	1.0	0.254902
3	0.35	1.0	0.241379
4	0.45	1.0	0.350649
5	0.55	1.0	0.666667
6	0.65	1.0	0.840909
7	0.75	1.0	0.818182
8	0.85	1.0	0.812500
9	0.95	1.0	0.800000

In [707]:
```
df['y_bin'] = pd.cut(df['y_pred'], bins=bins2, labels=labels2)
```

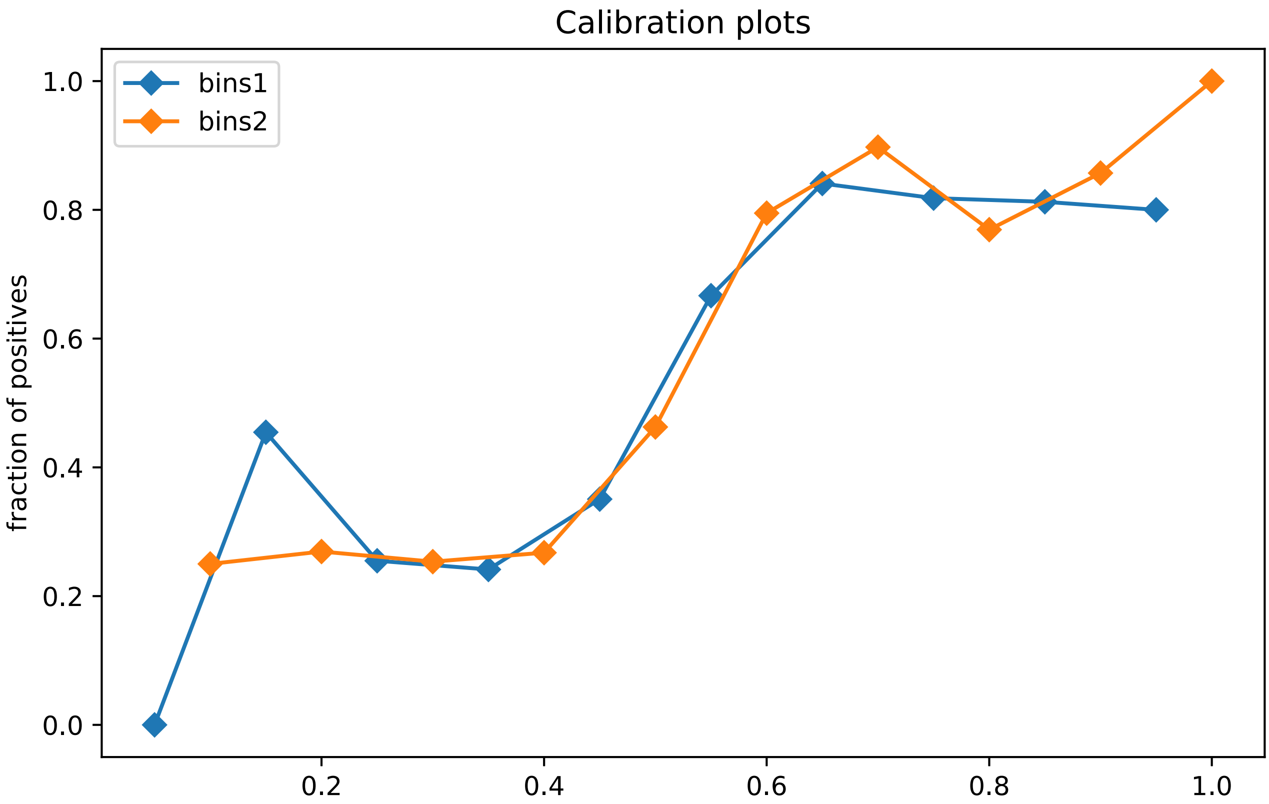
In [708]:
```
g = df.groupby(["y_bin", "y_val"]).count()
g = g.fillna(0)
```

In [709]:
```
g2 = g.groupby(level=[0]).apply(lambda g: g/g.sum())
g2 = g2.iloc[g2.index.get_level_values('y_val')==1]
g2 = g2.reset_index()

g2['y_bin'] = g2['y_bin'].astype('float')
g2.rename(index=str, columns={"y_pred": "percentage"})
```

Out[709]:

	y_bin	y_val	percentage
0	0.025	1.0	NaN
1	0.100	1.0	0.250000
2	0.200	1.0	0.269231
3	0.300	1.0	0.253521
4	0.400	1.0	0.267442
5	0.500	1.0	0.462687
6	0.600	1.0	0.794872
7	0.700	1.0	0.897436
8	0.800	1.0	0.769231
9	0.900	1.0	0.857143
10	1.000	1.0	1.000000

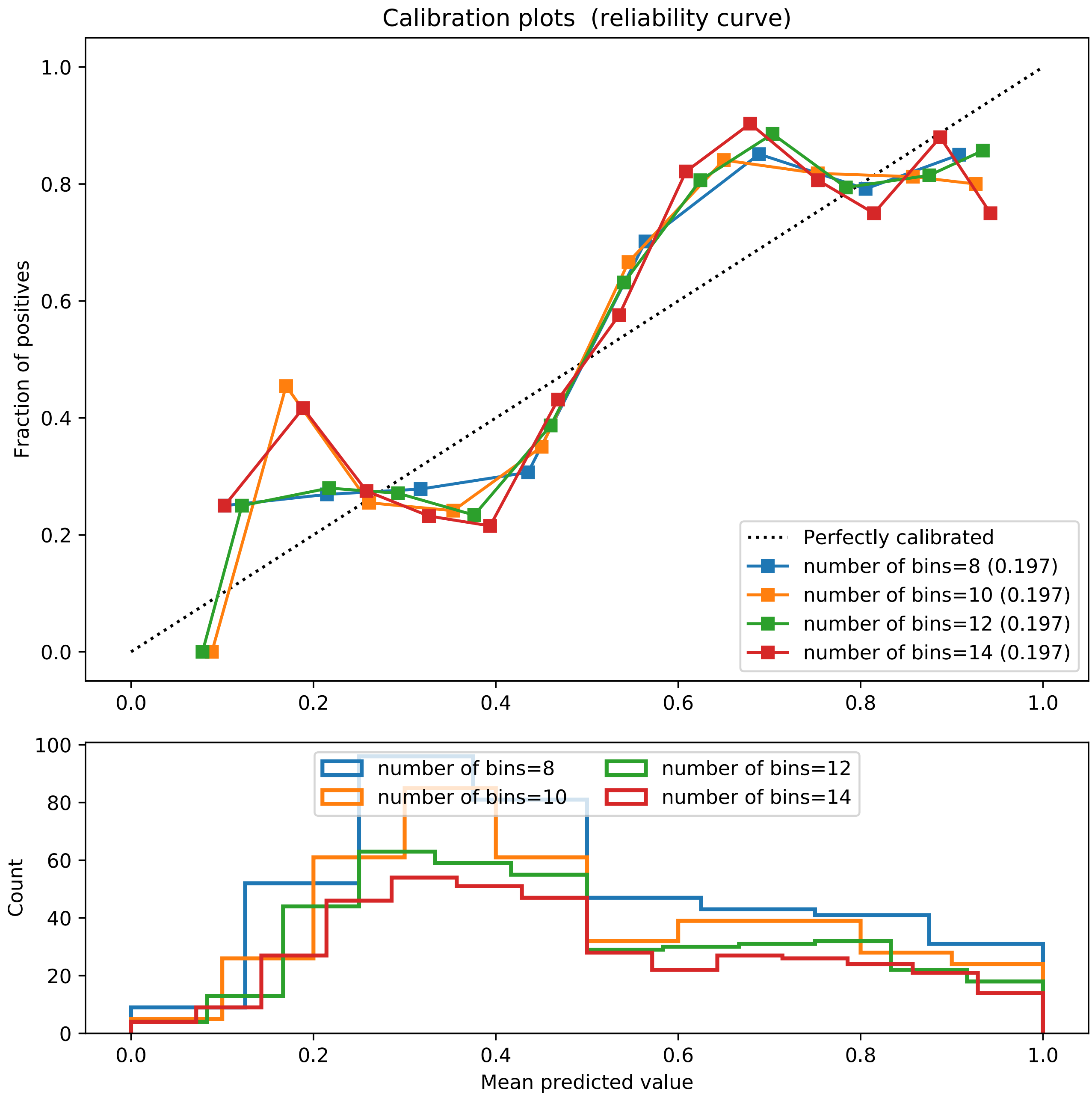
In [716]:
```
plt.figure(figsize=(8, 5))

plt.plot(g1['y_bin'],g1['y_pred'],marker='D', label ='bins1')
plt.plot(g2['y_bin'],g2['y_pred'],marker='D', label ='bins2')
x = np.linspace(0, 1, 1000)

plt.legend()
plt.ylabel('fraction of positives')
plt.title('Calibration plots')
plt.savefig('2.png')
plt.show()
```



In [660]:
```
prob_pos = (y_pred - y_pred.min()) / (y_pred.max() - y_pred.min())
```

In [718]:

```
from sklearn.calibration import calibration_curve
plt.figure(figsize=(8, 8))
ax1 = plt.subplot2grid((3, 1), (0, 0), rowspan=2)
ax2 = plt.subplot2grid((3, 1), (2, 0))

ax1.plot([0, 1], [0, 1], "k:", label="Perfectly calibrated")
bins = [8,10,12,14]
for b in bins:
    fraction_of_positives, mean_predicted_value = calibration_curve(y_val, y_p
red, n_bins=b)
    ax1.plot(mean_predicted_value, fraction_of_positives, "s-", label="number
 of bins=%s (%1.3f)" % (b, clf_score))

    ax2.hist(prob_pos, range=(0, 1), bins=b,histtype="step", lw=2,label="numbe
r of bins=%s" % (b, ))

ax1.set_ylabel("Fraction of positives")
ax1.set_ylim([-0.05, 1.05])
ax1.legend(loc="lower right")
ax1.set_title('Calibration plots  (reliability curve)')

ax2.set_xlabel("Mean predicted value")
ax2.set_ylabel("Count")
ax2.legend(loc="upper center", ncol=2)

plt.tight_layout()
plt.savefig('3.png')
plt.show()
```



Calibration plots (reliability curve)
Fraction of positives
Perfectly calibrated
number of bins=8 (0.197)
number of bins=10 (0.197)
number of bins=12 (0.197)
number of bins=14 (0.197)
Count
number of bins=8
number of bins=10
number of bins=12
number of bins=14
Mean predicted value


```
In [ ]:
```