Module 6 Homework

ISE-529, Summer 2022

1) The file "bank_evals.csv" contains data on 20 banks with the following attributes:

- FinanciallyWeak: (1 - financially weak condition, 0 - other)
- CaptitalAssetRatio: Ratio of total capital to asses
- ExpAssetRatio: Ratio of total expenses to total assets
- LoansLeasesToAssetRatio: Ratio of total loans and leasses to total assets

a) Create a logistic regression that models the status of the bank (whether or not it is weak) as a function of TotLns&Lses/Assets and TotExp/Assets. Use all 20 observations as your training data. Show the resulting intercept and coefficients of your model.

In [1]:
```
import math;
import numpy;
import pandas;
import statsmodels.api as sm;
import matplotlib.pyplot as plt;
import sklearn.metrics as metrics;
from sklearn.linear_model import LogisticRegression;

df1 = pandas.read_csv("bank_evals.csv");

X = df1[["ExpAssetRatio", "LoansLeasesToAssetRatio"]];
Y = df1["FinanciallyWeak"];

df1_logrg_model = LogisticRegression().fit(X, Y);
print("intercept: ", df1_logrg_model.intercept_);
print("coefficients: ", df1_logrg_model.coef_);
```

```
intercept:  [-0.47333652]
coefficients:  [[0.16075579 0.72642506]]
```

b) Using the LaTex capabilities in Jupyter Notebook, write the equation of the probability of being financially weak as a function of these two predictors.

$$P = \frac{e^{-0.47333652+0.16075579*ExpAssetRatio+0.72642506*LoansLeasesToAssetRatio}}{1+e^{-0.47333652+0.16075579*ExpAssetRatio+0.72642506*LoansLeasesToAssetRatio}}$$

c) If we had a new bank whose total loans and leases to asset ratio was 0.6 and total expenses to assets ratio was 0.11, what is the probability that it would be classified as “weak” by regulatory experts? What are the odds?

In [2]:
```
ExpAssetRatio = 0.11;
LoansLeasesToAssetRatio = 0.6;
e_term = numpy.exp(df1_logrg_model.intercept_[0] + df1_logrg_model.coef_[0][0] * ExpAssetRatio + df1_logrg_model.coef_[0][1] * L
new_bank_weak_p = e_term / (1 + e_term);
ExpAssetRatio_odds = numpy.exp(df1_logrg_model.coef_[0][0]);
LoansLeasesToAssetRatio_odds = numpy.exp(df1_logrg_model.coef_[0][1]);

print("The probability is: ", new_bank_weak_p);
print("The ExpAssetRatio_odds is: ", ExpAssetRatio_odds);
print("The LoansLeasesToAssetRatio_odds is: ", LoansLeasesToAssetRatio_odds);
```

```
The probability is:  0.4950505751897964
The ExpAssetRatio_odds is:  1.1743981391928293
The LoansLeasesToAssetRatio_odds is:  2.0676755730015777
```

d) Interpret the estimated coefficient for the ratio of total loans and leasses to total assets in terms of the odds being financially weak.

An increase of 1 unit of the ratio of total loans and leasses to total assets multiplies the odds of financially weak by 2.0677.

e) When a weak bank is misclassified as strong, the implications of that mistake are much greater than when a strong bank is misclassified as weak. To minimize the effects of this, should the cutoff value be increased or decreased from a default value of 0.5?

The cutoff value should be decreased from a default value of 0.5, because it regulates the data supposed to be weak but now to be strong.

2) Suppose we collect data for a group of students in a statistics class with variables $X_1 =$ hours studied, $X_2 =$ undergrad GPA, and $Y =$ receive an A. We fit a logistic regression and produce estimated coefficients $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, and $\hat{\beta}_2 = 0.9$ (20 points)

a) Estimate the probability that a student who studies for 20 hours and has an undergrad GPA of 3.5 gets an A in the class: 13.5873%

In [3]:
```
hours = 20;
GPA = 3.5;
e_term = numpy.exp(-6 + 0.05 * hours + 0.9 * GPA);
p = e_term / (1 + e_term);
print("The probability is: ", p);
```

```
The probability is:  0.13587289700909425
```

b) How many hours would the student in part (a) need to study to have a 60 % chance of getting an A in the class? 146.6338 hours

```
In [4]: p = 0.6;
        GPA = 3.5;
        hours = (numpy.log(-p / (p - 1)) - 0.9 * GPA + 6) / 0.05;
        print("The hours is: ", hours);
```

```
The hours is:  65.10930216216327
```

3) The file model_assessment_data.csv contains the output of a classification model for 100 observations. The first column – P(Y) contains the probability that the model assigned to that observation that it contained the event. The second column indicates whether it actually contained the event.

a) Create a dataframe with the following lift curve data as columns

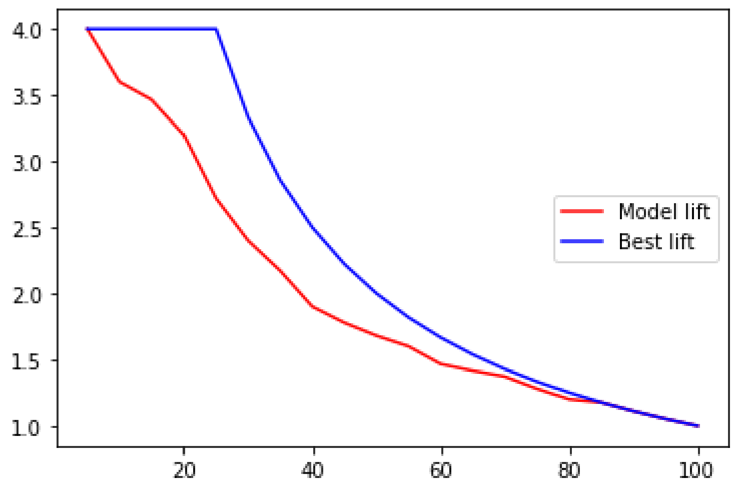
- Percentile: values from 5 to 100 by 5 (5, 10, 15, ..., 100)
- Model lift (value at the corresponding percentile)
- Best lift (value at the corresponding percentile)

```
In [5]: df3 = pandas.read_csv("model_assessment_data.csv");
        df3 = df3.sort_values(by = ["P(Y)"], ascending = False).reset_index(drop = True);


        step = 5;
        event_total = 0;
        event_count_list = [0] * int(len(df3) / step);
        percentile_list = [];
        model_lift_list = [];
        best_lift_list = [];


        for idx, row in df3.iterrows():
            if(row["Event?"] == 1):
                event_total += 1;
                event_count_list[int(idx / step)] += 1;

        event_prob = event_total / len(df3);


        cur_event_count = 0;
        cur_best_event_count = 0;
        for idx in range(0, 20):
            percentile = (idx + 1) * step;
            percentile_list.append(percentile);

            percentile_size = (percentile / 100) * len(df3);
            random_event_count = percentile_size * event_prob;
            cur_event_count += event_count_list[idx];
            model_lift = cur_event_count / random_event_count;
            model_lift_list.append(model_lift);

            if cur_best_event_count + step <= event_total:
                cur_best_event_count += step;
            else:
                cur_best_event_count = event_total;
            best_model_lift = cur_best_event_count / random_event_count;
            best_lift_list.append(best_model_lift);


        df3_lift = pandas.DataFrame(
            {
                "Percentile": percentile_list,
                "Model lift": model_lift_list,
                "Best lift": best_lift_list
            }
        )
        print(df3_lift);
```

```
    Percentile  Model lift  Best lift
0            5    4.000000   4.000000
1           10    3.600000   4.000000
2           15    3.466667   4.000000
3           20    3.200000   4.000000
4           25    2.720000   4.000000
5           30    2.400000   3.333333
6           35    2.171429   2.857143
7           40    1.900000   2.500000
8           45    1.777778   2.222222
9           50    1.680000   2.000000
10          55    1.600000   1.818182
11          60    1.466667   1.666667
12          65    1.415385   1.538462
13          70    1.371429   1.428571
14          75    1.280000   1.333333
15          80    1.200000   1.250000
16          85    1.176471   1.176471
17          90    1.111111   1.111111
18          95    1.052632   1.052632
19         100    1.000000   1.000000
```

b) Create a lift curve graphic using this data

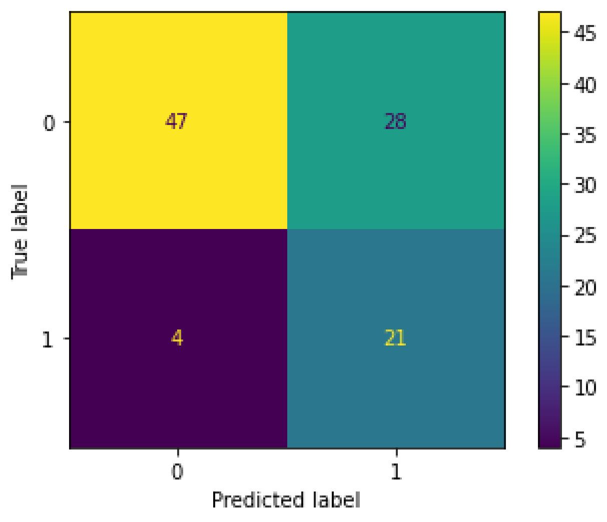
In [6]:
```
plt.plot(df3_lift["Percentile"], df3_lift["Model lift"], color = "red", label = "Model lift");
plt.plot(df3_lift["Percentile"], df3_lift["Best lift"], color = "blue", label = "Best lift");
plt.legend(loc = "right");
```



c) Calculate and display the confusion matrix, sensitivity, and specificity with a classification threshold of 0.5

In [7]:
```
threshold = 0.5;
Y_pred_list = [];
for idx, row in df3.iterrows():
    if row["P(Y)"] <= threshold:
        Y_pred_list.append(0);
    else:
        Y_pred_list.append(1);

Y = df3["Event?"];
cnf_matrix = metrics.confusion_matrix(Y, Y_pred_list);
metrics.ConfusionMatrixDisplay(cnf_matrix).plot();

TN = cnf_matrix[0,0];
FP = cnf_matrix[0,1];
FN = cnf_matrix[1,0];
TP = cnf_matrix[1,1];
print('True Negatives:', TN);
print('False Positives:', FP);
print('False Negatives:', FN);
print('True Positives:', TP);

sensitivity = TP / (TP + FN);
specificity = TN / (TN + FP);
print('Sensitivity:', sensitivity);
print('Specificity:', specificity);
```

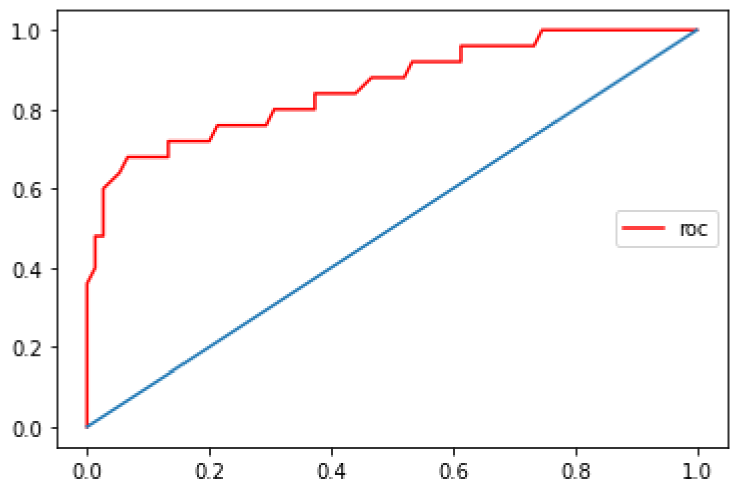
```
True Negatives: 47
False Positives: 28
False Negatives: 4
True Positives: 21
Sensitivity: 0.84
Specificity: 0.6266666666666667
```



d) Create an ROC chart (using 1% intervals for the threshold)

In [8]:
```
threshold_list = [];
FPR_list = [];
TPR_list = [];

for a in numpy.arange(0, 1 + 0.01, 0.01):
    threshold_list.append(a);

for cur_threshold in threshold_list:
    Y_pred_list = [];
    for idx, row in df3.iterrows():
        if row["P(Y)"] <= cur_threshold:
            Y_pred_list.append(0);
        else:
            Y_pred_list.append(1);

    cnf_matrix = metrics.confusion_matrix(Y, Y_pred_list);
    TN = cnf_matrix[0,0];
    FP = cnf_matrix[0,1];
    FN = cnf_matrix[1,0];
    TP = cnf_matrix[1,1];
    sensitivity = TP / (TP + FN);
    specificity = TN / (TN + FP);

    FPR = 1 - specificity;
    TPR = sensitivity;

    FPR_list.append(FPR);
    TPR_list.append(TPR);

df3_roc = pandas.DataFrame(
    {
        "threshold": threshold_list,
        "FPR": FPR_list,
        "TPR": TPR_list
    }
);
print(df3_roc);

plt.plot(df3_roc["FPR"], df3_roc["TPR"], color = "red", label = "roc");
ident = [0.0, 1.0];
plt.plot(ident, ident);
plt.legend(loc = "right");
```

```
     threshold       FPR   TPR
0         0.00  1.000000  1.00
1         0.01  1.000000  1.00
2         0.02  0.973333  1.00
3         0.03  0.973333  1.00
4         0.04  0.960000  1.00
..         ...       ...   ...
96        0.96  0.000000  0.12
97        0.97  0.000000  0.12
98        0.98  0.000000  0.00
99        0.99  0.000000  0.00
100       1.00  0.000000  0.00

[101 rows x 3 columns]
```



e) What value of threshold yields the largest ROC separation?

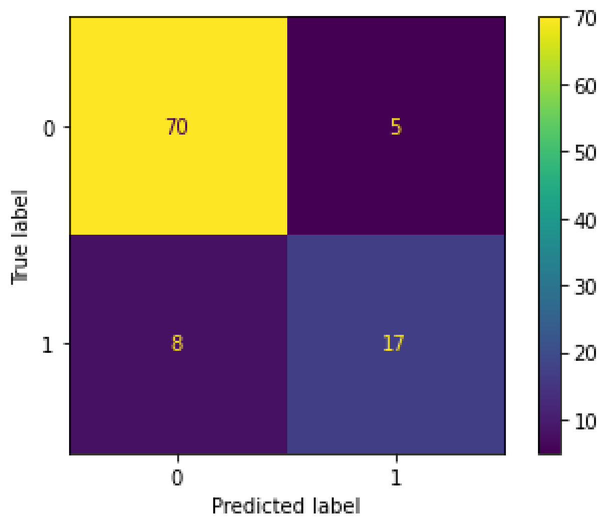
In [9]:
```
max_separation = -1;
max_separation_threshold = 0;

for idx, row in df3_roc.iterrows():
    cur_separation = row["TPR"] - row["FPR"];
    if(cur_separation > max_separation):
        max_separation = cur_separation;
        max_separation_threshold = row["threshold"];

print("threshold: ", max_separation_threshold);
```

```
threshold:  0.84
```

f) Recalculate the confusion matrix, sensitivity, and specificity using the optimal threshold you found in part (e):

In [10]:
```
threshold = max_separation_threshold;
Y_pred_list = [];
for idx, row in df3.iterrows():
    if row["P(Y)"] <= threshold:
        Y_pred_list.append(0);
    else:
        Y_pred_list.append(1);

Y = df3["Event?"];
cnf_matrix = metrics.confusion_matrix(Y, Y_pred_list);
metrics.ConfusionMatrixDisplay(cnf_matrix).plot();

TN = cnf_matrix[0,0];
FP = cnf_matrix[0,1];
FN = cnf_matrix[1,0];
TP = cnf_matrix[1,1];
print('True Negatives:', TN);
print('False Positives:', FP);
print('False Negatives:', FN);
print('True Positives:', TP);

sensitivity = TP / (TP + FN);
specificity = TN / (TN + FP);
print('Sensitivity:', sensitivity);
print('Specificity:', specificity);
```

```
True Negatives: 70
False Positives: 5
False Negatives: 8
True Positives: 17
Sensitivity: 0.68
Specificity: 0.9333333333333333
```


True label
Predicted label