

MGMTMFE 431:

Data Analytics and Machine Learning

Topic 3: Logistic regressions
FinTech and credit data

Spring 2025

Professor Lars A. Lochstoer

Advanced Multiple Regression Topics

- a. The Logistic Regression Model
- b. Interpretation of the Coefficients
- c. A Simple Example
- d. The Likelihood Function
- e. A Simple Example (continued)
- f. A More Complicated Example
- g. Lift Tables
- h. ROC Curves
- i. FinTech: Lending Club

c. The Logistic regression model

Suppose we have a binary dependent variable.

Examples:

1. Purchase of a product ($Y=1$ if purchase, $Y=0$ if not)
2. Click on display ad ($Y=1$ if click, $Y=0$ if not)
3. Predict direction of market ($Y=1$ up, $Y=0$ down)
3. Default on loan ($Y=1$ default, $Y=0$ no default)

All of these can be formulated as a conditional prediction problem: Given X variables, what is your prediction of Y?

Since Y is binary, the predictions are probabilities that $Y = 1$ (see next slide)

c. The Logistic regression model

What is a regression model, in general?

A model for the conditional distribution of $Y | X$.

What is the regression line? It is $E[Y|X]$.

If Y is binary (0,1 are the only possible values),

$$\begin{aligned} E[Y|X] &= \Pr(Y = 1|X) \times 1 + (1 - \Pr(Y = 1|X)) \times 0 \\ &= \Pr(Y = 1|X) \end{aligned}$$

c. The Logistic regression model

How can we link the X variables to the probability that $Y = 1$?

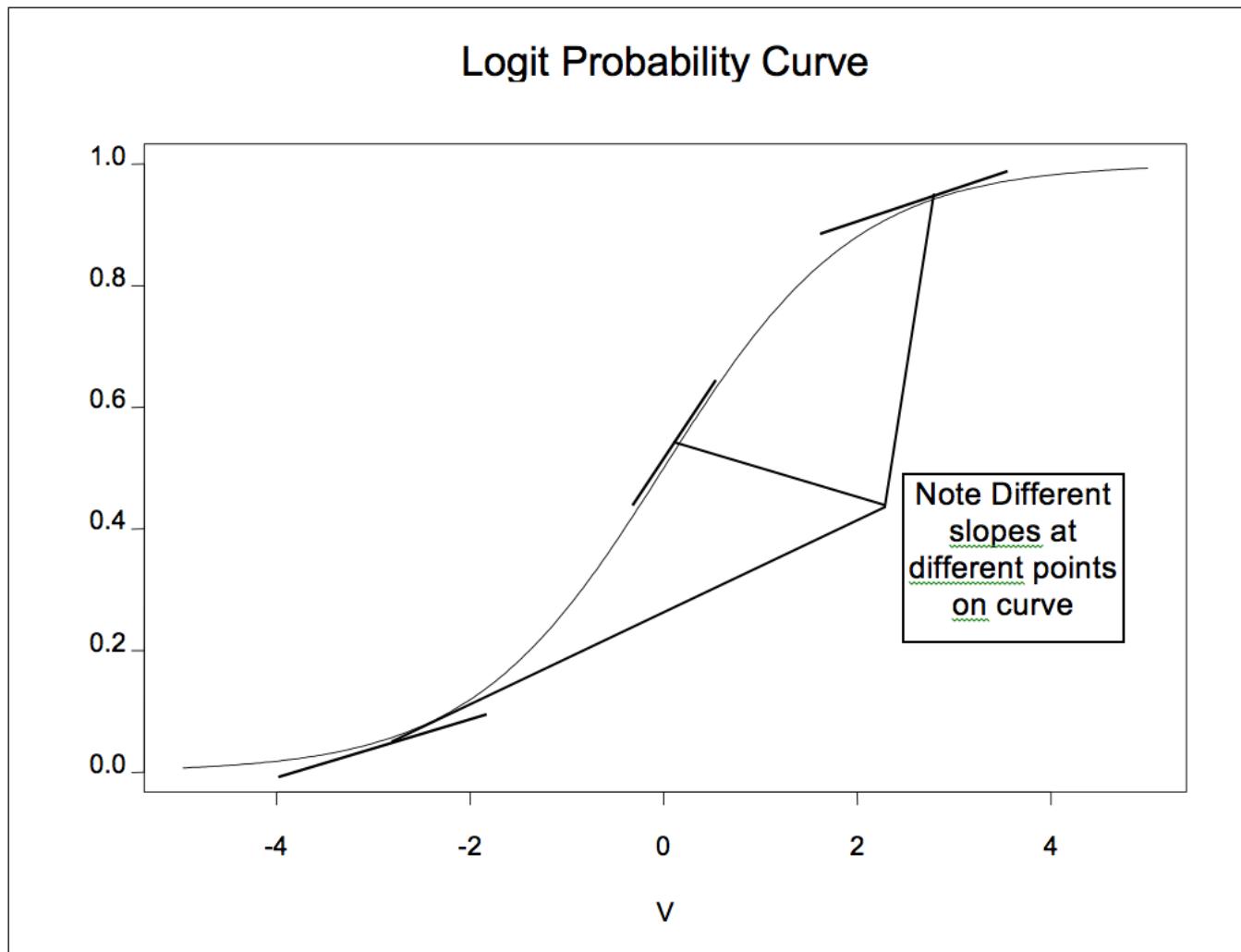
$$\Pr(Y = 1) = \frac{\exp(\beta_0 + \beta_1 X_1 + \dots + \beta_k X_k)}{1 + \exp(\beta_0 + \beta_1 X_1 + \dots + \beta_k X_k)}$$

We can think of $V = \beta_0 + \beta_1 X_1 + \dots + \beta_k X_k$ as a “score.” That is, what is the utility of buying.

As V gets large, the probability that $Y=1$ should get very close to 1. As V gets small, the probability that $Y=1$ should get close to zero.

$$\Pr(Y = 1) = \frac{\exp(V)}{1 + \exp(V)}$$

c. The Logistic regression model



d. Interpretation of Logistic slope coefficients

In a standard linear regression model, the slope coefficients should be interpreted as the average change in Y of a one unit change in the particular X variable.

We cannot interpret the slopes in a logit model as the change in the probability that $Y=1$ since the model is non-linear.

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \dots + \beta_k x_k$$

Log-odds is linear not the probability.

d. Interpretation of Logistic slope coefficients

The change in the probability of $Y=1$ with respect to a specific X variable is given by:

$$\frac{\partial \Pr(Y = 1|X)}{\partial X_j} = \beta_j \Pr(Y = 1|X) (1 - \Pr(Y = 1|X))$$

As a practical manner, we will simply use the fitted model to predict probabilities for different values of X and use this to determine change in probability for different values of X .

e. An Example

Consider the problem of predicting whether a borrower will default on a loan given their FICO score (300-850, higher is better) on application for the loan. We simulate some binary data (see code snippets for details).

Here X is the FICO score and Y = 1 if default, Y=0 if not.

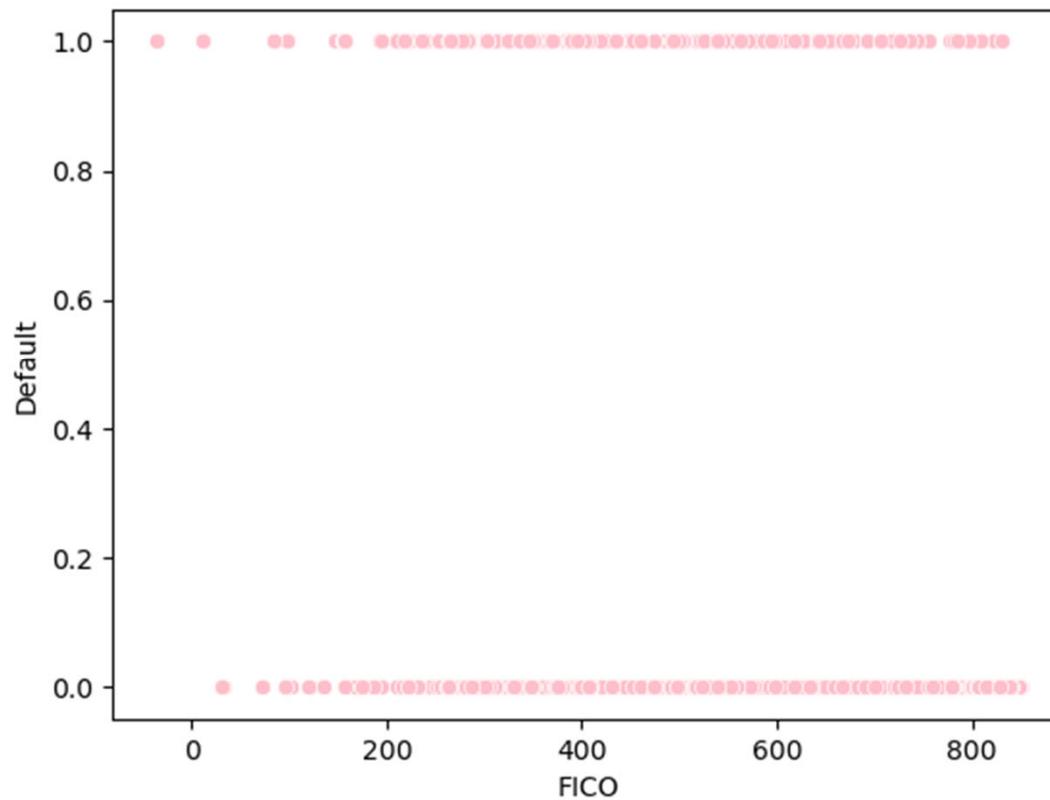
Let's look at the data.

```
>> data[0:10]
```

	Default	FICO
0	0	815.0
1	0	746.0
2	0	832.0
3	0	536.0
4	0	629.0
5	0	654.0
6	0	462.0
7	0	779.0
8	1	493.0
9	0	772.0

e. An Example

If we attempt a scatterplot of y vs. x, we will only have two values of y. We use the alpha setting to see the density of X values.

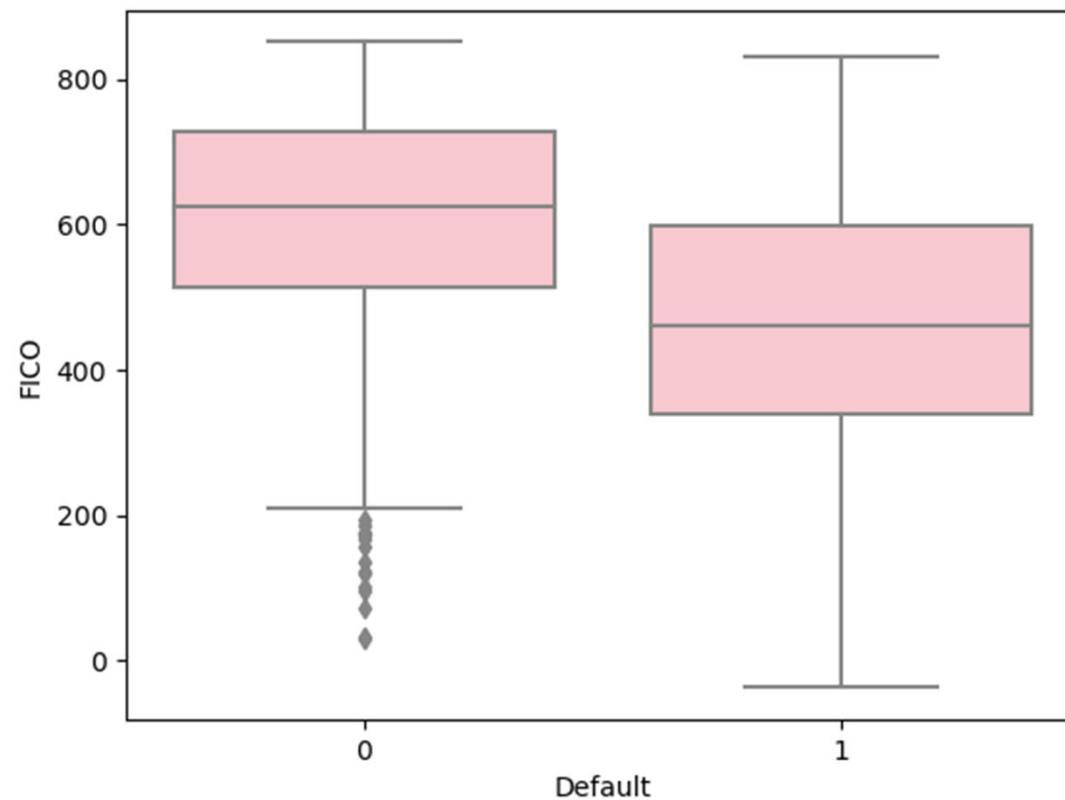


e. An Example

Hard to see what is going on. Let's do boxplots of FICO for the various values of Default.

Can I use
FICO to
classify the
observations?

Note that distri-
butions of FICO
scores overlap.



e. An Example

Let's fit the model and show coefficients.

```
data['c'] = 1
out      = Logit(data['Default'],data[['c','FICO']] ).fit()
Optimization terminated successfully.
    Current function value: 0.305533
    Iterations 7

out.summary2()
Out[33]:
Results: Logit
=====
Model:           Logit          Pseudo R-squared: 0.106
Dependent Variable: Default      AIC:            1042.2001
Date:             2021-04-10 20:05  BIC:            1053.0757
No. Observations: 1699          Log-Likelihood:   -519.10
Df Model:         1              LL-Null:        -580.55
Df Residuals:     1697          LLR p-value:    1.4654e-28
Converged:        1.0000        Scale:          1.0000
No. Iterations:   7.0000

Coef.      Std.Err.      z      P>|z|      [0.025      0.975]
-----
c          0.7103      0.2554    2.7812    0.0054      0.2097    1.2108
FICO      -0.0052     0.0005   -10.6507   0.0000     -0.0062   -0.0043
-----
```

f. The Likelihood Function

How does Python fit this model to the data? There are no standard residuals. We can't do least squares.

The model is fit using the idea of maximum likelihood -- maximize the probability of observations.

Let's consider a coin toss of a not necessarily fair coin. Suppose we see 3 Heads in 10 coin tosses. Most of us would estimate the probability of a head for this coin to be 3/10.

Let's call θ the probability of a head. What is the likelihood of the data? It depends on theta!

f. The Likelihood Function

If we set $\theta = .5$, then what is the likelihood of the data?

$$L(\theta) = \theta^3(1 - \theta)^{10-3}$$

$$\text{if } \theta = 0.5$$

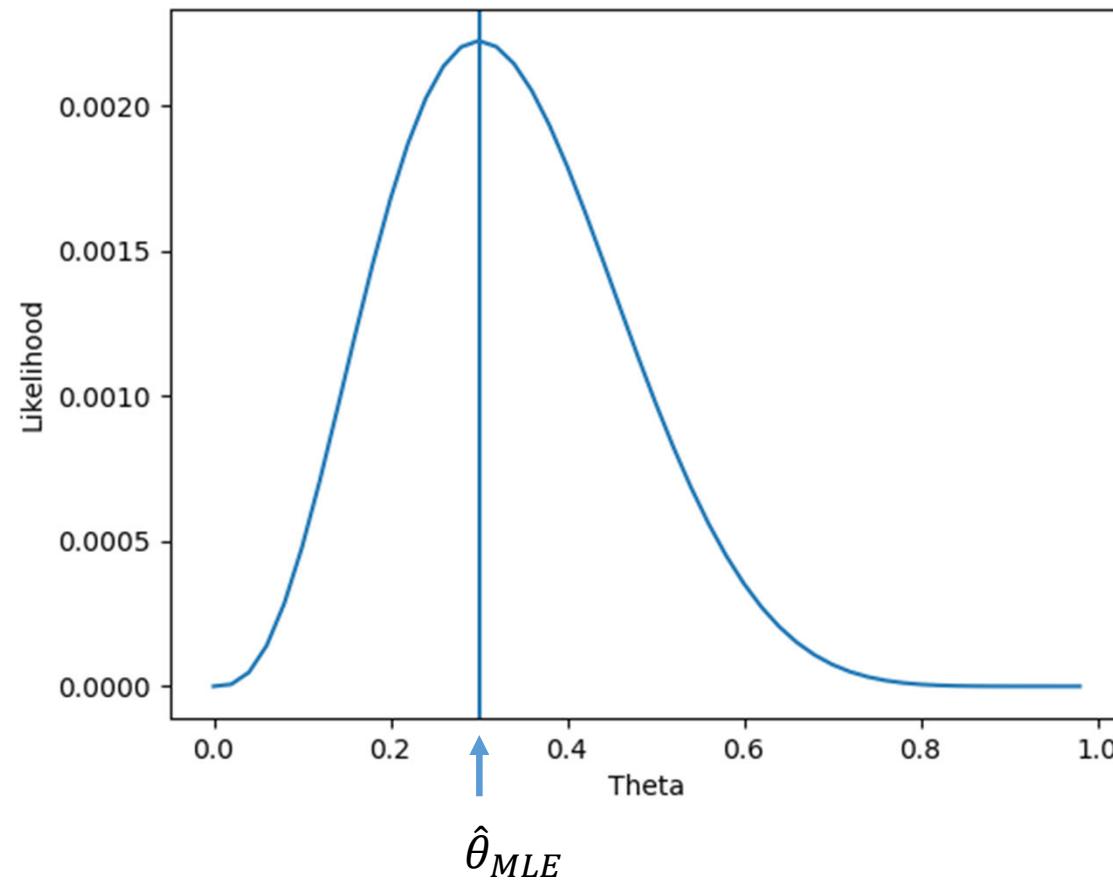
$$L(0.5) = 0.5^3(1 - 0.5)^7$$

Let's find the value of theta, which maximizes the "likelihood" of the observed data (3 Heads from 10).

Best way to do this is via a graph.

f. The Likelihood Function

Likelihood for coin toss, nhead=3, N=10



f. The Likelihood Function

So the idea is to pick logit regression parameters to maximize the probability the observed defaults given FICO.

For any value of beta0 and beta1, we can compute the probability of default from the Logit Model. The likelihood becomes the “score” for that pair of “guesses” of beta0 and beta1.

The likelihood is simply all of the probabilities for the observed defaults multiplied together. We ask the computer to maximize the likelihood for us by searching over possible values of beta0 and beta1.

f. The Likelihood Function

If we guess $\beta_0 = 0$ and $\beta_1 = -0.005$,

$$Pr(Y_1) = Pr(\text{No Default} | \text{FICO} = 815)$$

$$= 1 - \frac{\exp(0 - 0.005 \times 815)}{1 + \exp(0 - 0.005 \times 815)} = 1 - 0.017$$

...

$$Pr(Y_7) = Pr(\text{No Default} | \text{FICO} = 779)$$

$$= 1 - \frac{\exp(0 - 0.005 \times 779)}{1 + \exp(0 - 0.005 \times 779)} = 1 - 0.020$$

$$Pr(Y_8) = Pr(\text{Default} | \text{FICO} = 493)$$

$$= \frac{\exp(0 - 0.005 \times 493)}{1 + \exp(0 - 0.005 \times 493)} = 0.078$$

	Default	FICO
0	0	815.0
1	0	746.0
2	0	832.0
3	0	536.0
4	0	629.0
5	0	654.0
6	0	462.0
7	0	779.0
8	1	493.0
9	0	772.0

f. The Likelihood Function

If we guess $\beta_0 = 0$ and $\beta_1 = -0.005$,

$$\begin{aligned} L(\beta_0 = 0, \beta_1 = -0.005) &= Pr(Y_1|FICO_1) \times \cdots \times Pr(Y_7|FICO_7) \times \\ &\quad Pr(Y_8|FICO_8) \times \cdots \times Pr(Y_N|FICO_N) \\ &= (1 - 0.017) \times \cdots \times (1 - 0.02) \times 0.078 \times \dots \end{aligned}$$

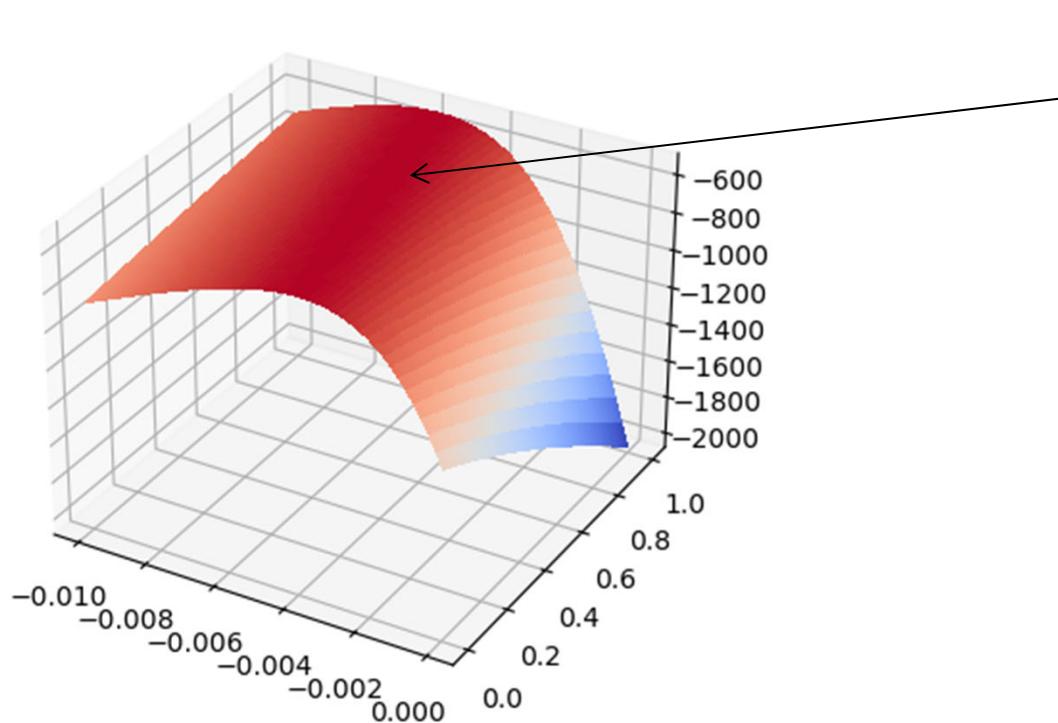
We ask the computer to find the values of the coefficients that maximize the likelihood of the observed data. Likelihood is the “scorecard” like SSE was for linear regression.

Using X values, we are trying to make the fitted probabilities of default as large as possible for all observations where Y=1 and as small as possible for all observations with Y=0.

This is called the method of **Maximum Likelihood**.

f. The Likelihood Function

Let's look at the surface that is being maximized:



Maximum
Likelihood
Estimate
“top of the
Mountain”

f. The Likelihood Function

Another way to see this is to write down the likelihood for the general logit model.

$$L(\beta | y, X) = \prod_{i=1}^N \Pr(Y_i = 1)^{y_i} (1 - \Pr(Y_i = 1))^{1-y_i}$$

$$\Pr(Y_i = 1) = f(\beta) = \frac{\exp(x_i^\top \beta)}{1 + \exp(x_i^\top \beta)}$$

$$X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_N^\top \end{bmatrix}$$

Let's code it up and let Python find the maximum!

f. The Likelihood Function

`minimize()` is an optimizer that finds the “minimum” of any function you give it using numerical derivatives by default.

- Note that the estimated parameters are the same as in the built-in function

```
y = data.Default
x = data[['c','FICO']]

def loglike(beta,y,x):
    ind = np.dot(x,beta)#np.matmul(x, beta)
    pr = np.exp(ind)
    pr = pr/(1+pr)
    return -sum( y*np.log(pr) + (1-y)*np.log(1-pr) )

beta_guess = np.array([0.5,0.01])
solution = minimize(loglike,beta_guess,args=(y,x),method =
                     'Nelder-Mead',options={'disp': True})
print('ML estimates: '+ str(solution.x))
Optimization terminated successfully.
    Current function value: 519.100037
    Iterations: 45
    Function evaluations: 85

ML estimates: [ 0.71022528 -0.00522286]
```

f. Deviance

Note the “ R^2 ” is not a natural measure of fit for a logistic regression.

- Errors are not normal (where variance makes sense) or even symmetric (so skewness is to be expected)
- Of this reason, we use “*Deviance*” as a measure of fit

Consider again the likelihood function:

$$L(\beta|y, X) = \prod_{i=1}^N Pr(Y_i = 1)^{y_i} (1 - Pr(Y_i = 1))^{1-y_i}$$

Note that if there are enough parameters to fit each observation perfectly, we have that $L = 1$.

- Call this the *saturated model*, M_s

Let the *null* model, M_n , be the one with all coefficients, except the intercept coefficient, equals zero.

Let the proposed logit model be the candidate model, M_c , where the K betas are all estimated.

f. Deviance (cont'd)

Define the *null deviance* as:

$$d_{\text{null}} = 2(\ln L(M_s) - \ln L(M_n))$$

Define the *residual deviance* as:

$$d_{\text{residual}} = 2(\ln L(M_s) - \ln L(M_c))$$

Notice that these are 2 times the difference in *log likelihood ratios* between the saturated and the null and candidate models

- Thus, from the standard **likelihood ratio test**, this difference is Chi2-distributed with degrees of freedom equal to the number of observations in the sample minus the number of parameters in the non-saturated models

g. A Simple Example continued

Let's use the model and the *predict* feature to compute the expected change in default probability as we move FICO from 800 to 500.

```
from scipy.stats import chi2

logit_model = sm.GLM(data['Default'],data[['FICO','c']], family=
                      sm.families.Binomial())
logit_results = logit_model.fit()
Optimization terminated successfully.
    Current function value: 0.305533
    Iterations 7

logit_results.predict([500,1])-logit_results.predict([800,1])

Out[41]: array([0.0997374])
```

g. A Simple Example continued

If you have only one regressor in the logistic regression, we just use the “t” (z) statistics to test for the significance of the regression. For more than one regressor, we need something like the overall F test. Here there is a chi-squared test that uses “deviance” computations.

Here the null is that all model slopes are zero. P-value can be used to test null. Deviance is sort of like SSE in a regular regression.

```
test_stat      = logit_results.null_deviance-logit_results.deviance
k              = 1
p_value_chisq = 1-chi2.cdf(test_stat,k)
test_stat
Out[45]: 122.90153992410433
k
Out[46]: 1
p_value_chisq
Out[47]: 0.0
```

g. A Simple Example continued

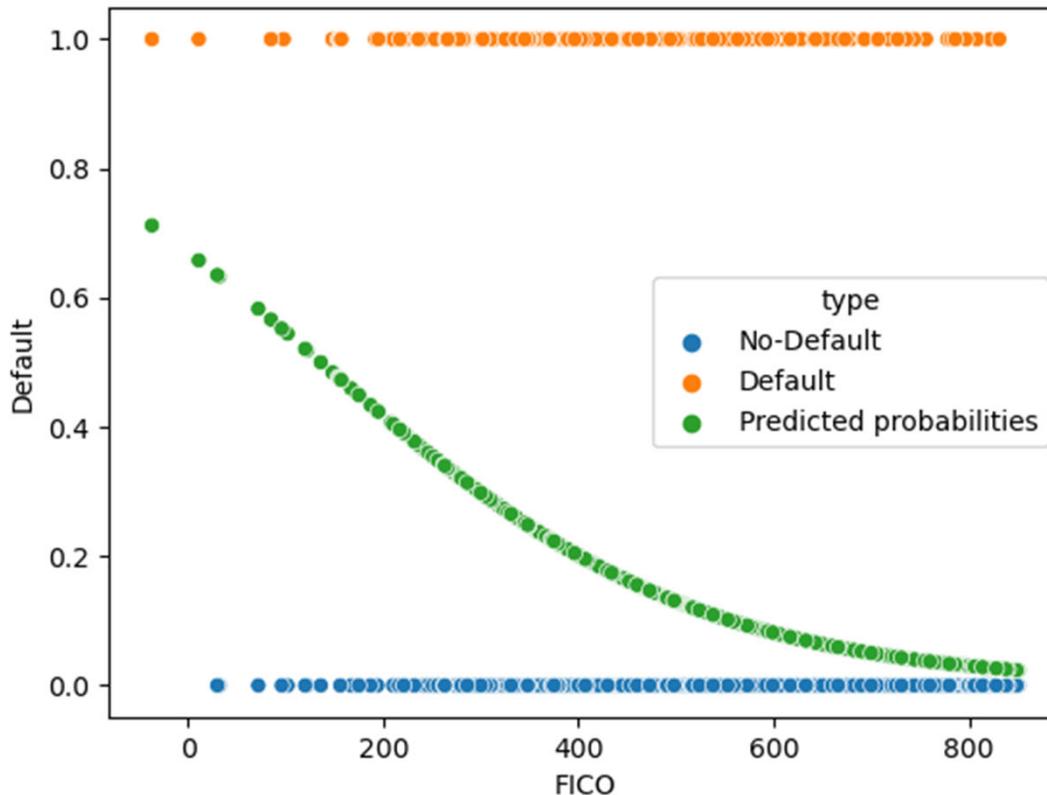
What is the null model here?

It is simply a logistic regression with an intercept term and no slope. This is a model for which the intercept will be chosen to make the fitted probabilities that $Y = 1$ equal the frequency for which $Y = 1$ in the data.

That is, the null model is simply to ignore X and compute the marginal probability that $Y = 1$ as opposed to the model which conditions on X !

g. A Simple Example continued

For logistic regression, to visualize the model, the best we can do is compare outcomes to fitted probabilities in a plot. (see script)



h. A More Complicated Example

Consider a very common problem in Business Data Analytics:

Predict default on a consumer loan given information available at the time the loan is offered.

Explanatory Variables:

1. Credit History of Borrower
2. Terms of the Loan
3. Demographics (behavior usually trumps demos!)

h. A More Complicated Example

The **loans** dataset has information on 1,000 loans. Let's build a logistic regression.

First, let's import data and define y and X variables in our full model:

```
import statsmodels.api as sm
from patsy import dmatrices

#import feather
loans = pd.read_feather('loans.feather')
y, Xf = dmatrices('default ~ C(StatChkA) + Duration + C(CrdHist) + C(Purpose) \
+ CrdAmt + C(Sav_Bnd) + Empl + C(InstallRate) \
+ C(Pstatus) + C(OthrDebt) + Resid + C(Proprty) + Age \
+ C(OthrInstall) + C(Housing) + Ncredits + C(job) \
+ Nsupport + C(Telephone) + Foreign',
loans, return_type = 'dataframe')
```

h. A More Complicated Example

Most of the variables are **categorical** or **qualitative** variables.

Examples:

`Credit.history`

`all credits at this bank paid back duly`

`critical account/other credits existing (not at this bank)`

`delay in paying off in the past`

`existing credits paid back duly till now`

`no credits taken/all credits paid back duly`

h. A More Complicated Example

Let's fit the full model. There will be a lot of coefficients, because we created dummy variables for all of the categorical variables using the C() function two slides ago.

```
logit_model_full = sm.GLM(y,Xf, family=sm.families.Binomial()).fit()
```

The “summary” of the model fit takes up a lot of space because of the long text descriptions of the factor levels.

h. A More Complicated Example

```
print(logit_model_full.summary())
Generalized Linear Model Regression Results
```

	coef	std err	z	P> z	[0.025	0.975]
Intercept	-2.6325	1.381	-1.906	0.057	-5.339	0.074
C(StatChkA)[T.... >= 200 DM]	-0.9648	0.370	-2.611	0.009	-1.689	-0.241
C(StatChkA)[T.0 <= ... < 200 DM]	-0.3744	0.218	-1.718	0.086	-0.802	0.053
C(StatChkA)[T.no checking account]	-1.7107	0.232	-7.365	0.000	-2.166	-1.255
C(CrdHist)[T.critical account]	-1.5782	0.438	-3.600	0.000	-2.437	-0.719
C(CrdHist)[T.delay in paying off in the past]	-0.9972	0.470	-2.120	0.034	-1.919	-0.075
C(CrdHist)[T.existing credits paid back duly till now]	-0.7310	0.385	-1.897	0.058	-1.486	0.024
C(Purpose)[T.car (new)]	0.7399	0.334	2.212	0.027	0.084	1.395
C(Purpose)[T.car (used)]	-0.9328	0.442	-2.113	0.035	-1.798	-0.067
C(Sav_Bnd)[T.... < 100 DM]	1.3423	0.526	2.554	0.011	0.312	2.372
C(InstallRate)[T.3]	0.6956	0.339	2.054	0.040	0.032	1.359
C(InstallRate)[T.4]	0.9730	0.302	3.226	0.001	0.382	1.564
C(Pstatus)[T.male : single]	-0.5395	0.211	-2.559	0.010	-0.953	-0.126
C(OthrDebt)[T.guarantor]	-1.4181	0.569	-2.493	0.013	-2.533	-0.303
C(OthrInstall)[T.none]	-0.6459	0.239	-2.701	0.007	-1.115	-0.177
Foreign[T.yes]	1.3986	0.628	2.228	0.026	0.168	2.629
Duration	0.0279	0.009	3.002	0.003	0.010	0.046
CrdAmt	0.0001	4.45e-05	2.890	0.004	4.13e-05	0.000

Note: how Python created dummy variables for each of the possible values of the Sav_Bnd variable except one (> 1000 in savings bonds)

- You should interpret the coefficients as whether the probability of default will increase for the category versus the reference category.
- e.g. if you have less than 100 in savings account, then you are more likely to default than someone with > 1000!

I am only giving you coefficients that are significant here.

h. A More Complicated Example

```
# Let's refit with only the factors that have significant coefficients
#
y, x = dmatrices('default ~ C(StatChkA) + Duration+ C(CrdHist) + C(Purpose) \
                  + CrdAmt + C(Sav_Bnd)+C(InstallRate) + C(Pstatus) \
                  + C(OthrDebt) + C(OthrInstall) + Foreign',
                  loans, return_type = 'dataframe')

logit_model = sm.GLM(y,x, family=sm.families.Binomial()).fit()
```

Can we do a partial-f test to see if those factors I threw out should be kept out?

We don't have a F-test but we do have a Chi-squared test based on the change in deviance or fit versus the number of variables removed. In other words, as we drop variables are we dropping degrees of freedom faster than reduction in fit?

h. A More Complicated Example

Inclusion-Exclusion Test:

Let's compare the deviance from the full (all variables) with the restricted (insignificant variables are removed) just as we compared the R-squared of the full with the R-squared of the restricted for the F-test.

We also need to count how many variables were dropped. I can fetch this information from the summary() output.

h. A More Complicated Example

```
delta_df= logit_model_full.df_model - logit_model.df_model
```

```
delta_df
```

```
Out[55]: 17
```

```
delta_dev=logit_model.deviance-logit_model_full.deviance
```

```
delta_dev
```

```
Out[57]: 25.059807961647152
```

```
p_value_chisq = 1-chi2.cdf(delta_dev,delta_df)
```

```
p_value_chisq
```

```
Out[59]: 0.09337890681139305
```

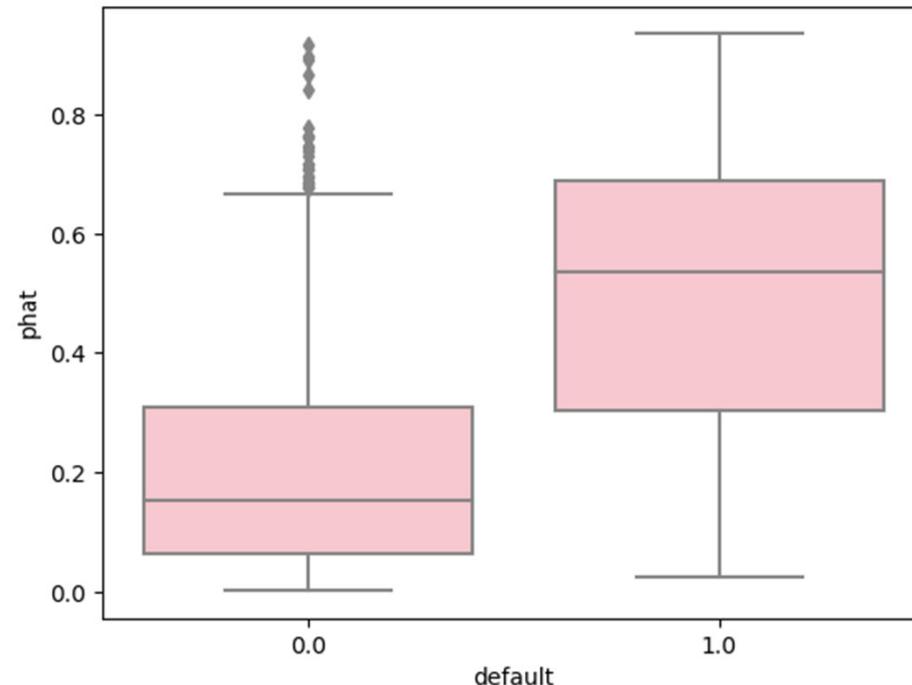
P-value for test. Null:
variables have zero
coefficients (should be
thrown out). Restricted
model not rejected at 5%
level

Change in fit. Fit is worse
for simpler model by
25.06 “deviance” points.

17 variables thrown out
(variables not factors)

h. A More Complicated Example

Let's see how well the model does by plotting the distribution of fitted probabilities by default.



```
loans['phat'] = logit_model.predict(X)
plt.figure()
ax = sns.boxplot(x='default', y='phat', data=loans,color='pink')
```

i. Lift Tables

Lift Table:

There is no “R-squared” for this model.

How should we evaluate the ability of the model to predict default?

A common practice is to create a “lift” table.

Sort the data by fitted probabilities and then compute the mean of the Y variable (mean response – in this case, mean default rate) for each decile of fitted probabilities.

If the model works well, then we should see much higher default rates for the higher fitted probabilities.

i. Lift Tables

This is converted in to a “lift” factor by dividing by the average response rate or overall average of Y.

A poor fitting model must not sort the observations well – the mean response rate for high fitted probabilities would only be marginally better than for small fitted probabilities.

A good fit is evidenced by good discrimination of the data. The mean response rate for high fitted probabilities would be much greater than for low fitted probabilities.

This is called the “**Lift Table**”

i. Lift Tables

```

Find Deciles
    loans['deciles'] = pd.qcut(loans['phat'], 10, labels=np.arange(1, 11, 1))
    df = loans[['deciles', 'phat', 'default']]
    lift = df.groupby('deciles')[['deciles', 'default']].mean()
    lift['Lift Factor'] = lift['default']/np.mean(loans.default)

lift
Out[62]:
      default  Lift Factor
deciles
1           0.03     0.100000
2           0.04     0.133333
3           0.11     0.366667
4           0.13     0.433333
5           0.19     0.633333
6           0.27     0.900000
7           0.39     1.300000
8           0.40     1.333333
9           0.67     2.233333
10          0.77     2.566667

```

Compute Mean Response for each Decile

Here we are looking for even increase in Lift from 1st thru 10th decile and large values for highest deciles.

j. ROC Curves

Receiver Operating Characteristics (ROC) graphs are useful for organizing classifiers and visualizing their performance

An alternative to Lift Tables (which was introduced instead of R²)

Popular metric so should know about this as well!

Benefits:

- Insensitive to changes in outcome distribution (overall positive outcomes (say, 1) versus negative outcomes (say, 0))
- This could be important if, say, instances of fraud changes from month to month
- Two-dimensional graph can be reduced to a single number of model fit (Area Under Curve) that has intuitive interpretation

j. Error Types

When considering whether to extend a loan or not, we can think of two types of borrowers

1. Good borrowers (repay loan)
2. Bad borrowers (default on loan)

All lending institutions, including Market Place Lenders (MPLs) such as Lending Club, have sophisticated tools to try and distinguish between them

Is the goal to minimize defaults?

- No. Easy to achieve: do not extend loans
- Giving credit to bad borrowers is costly but denying credit to good borrowers is costly in terms of opportunity cost

Ideal model: Lend to 100% good borrowers, 0% bad borrowers

j. Error Types - An example

ID	FICO	Status
5	670	Default
3	690	Default
1	710	Paid
6	730	Default
2	770	Paid
4	790	Paid

FICO: Fair Isaac Co. A credit score.

- Assume this is the only information we have for this example

j. Error Types – FICO example

FICO is related to defaults, but not perfectly

- Other factors and randomness at play

You need to decide on the FICO cutoff below which you will deny credit

- Cutoff 1 = 700
- Cutoff 2 = 740

j. Using Cutoff of 700

ID	FICO	Status	Prediction
5	670	Default	Default
3	690	Default	Default
1	710	Paid	Pay
6	730	Default	Pay
2	770	Paid	Pay
4	790	Paid	Pay

j. Model Performance @700

	True Default	True Paid
Predicted Default	2	0
Predicted Paid	1	3

In this case, for a model that predicts default (where ‘default’ is the ‘positive’ outcome):

- True positive (TP) = 2
- False positive (FP) = 0

j. Using Cutoff @740

ID	FICO	Status	Prediction
5	670	Default	Default
3	690	Default	Default
1	710	Paid	Default
6	730	Default	Default
2	770	Paid	Pay
4	790	Paid	Pay

j. Model Performance @740

	True Default	True Paid
Predicted Default	3	1
Predicted Paid	0	2

In this case, for a model that predicts default (where this is the ‘positive’ outcome):

- True positive (TP) = 3
- False positive (FP) = 1

j. The Confusion Matrix

Classification problems can be represented by the aptly named *Confusion Matrix*

T. Fawcett / Pattern Recognition Letters 27 (2006) 861–874

		<u>True class</u>			
		P	n		
		True Positives	False Positives		
<u>Hypothesized class</u>	Y			$\text{fp rate} = \frac{FP}{N}$	$\text{tp rate} = \frac{TP}{P}$
	N	False Negatives	True Negatives	$\text{precision} = \frac{TP}{TP+FP}$	$\text{recall} = \frac{TP}{P}$
Column totals:		P	N	$\text{accuracy} = \frac{TP+TN}{P+N}$	
				$\text{F-measure} = \frac{2}{1/\text{precision}+1/\text{recall}}$	

Fig. 1. Confusion matrix and common performance metrics calculated from it.

j. The ROC Curve

Tracing out the true and false positives for different cutoffs of the score (in this case, FICO) gives us the data for the scatter plot that is the **ROC Curve**

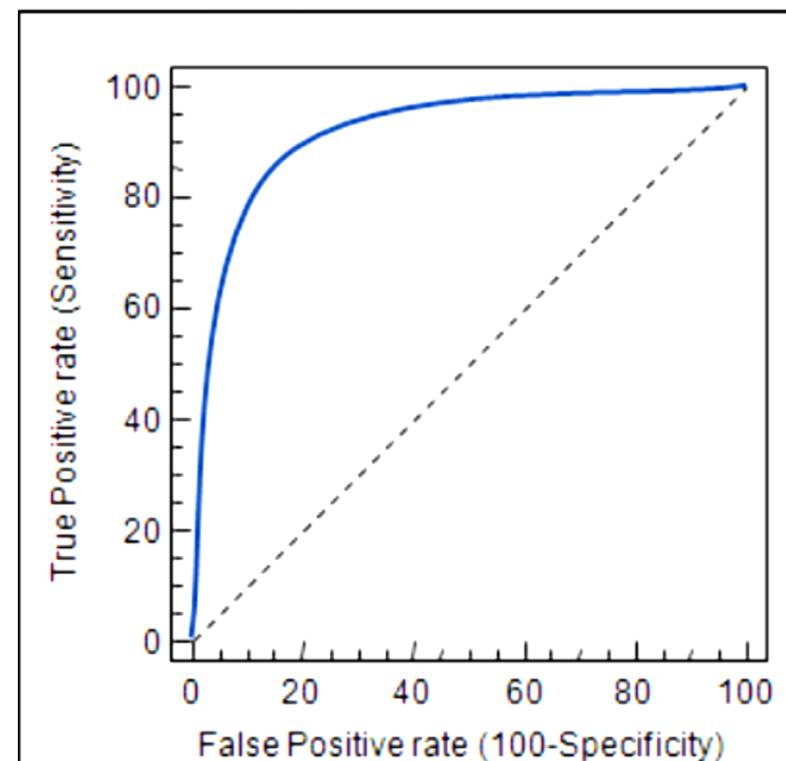
- Ideal model: vertical line at zero from zero to one (on y-axis), straight line at one thereafter from zero to one (on x-axis)

We use it to measure model performance

- The 45 degree line is the baseline, random guess case

Area Under Curve (AUC) summarizes model fit in one number

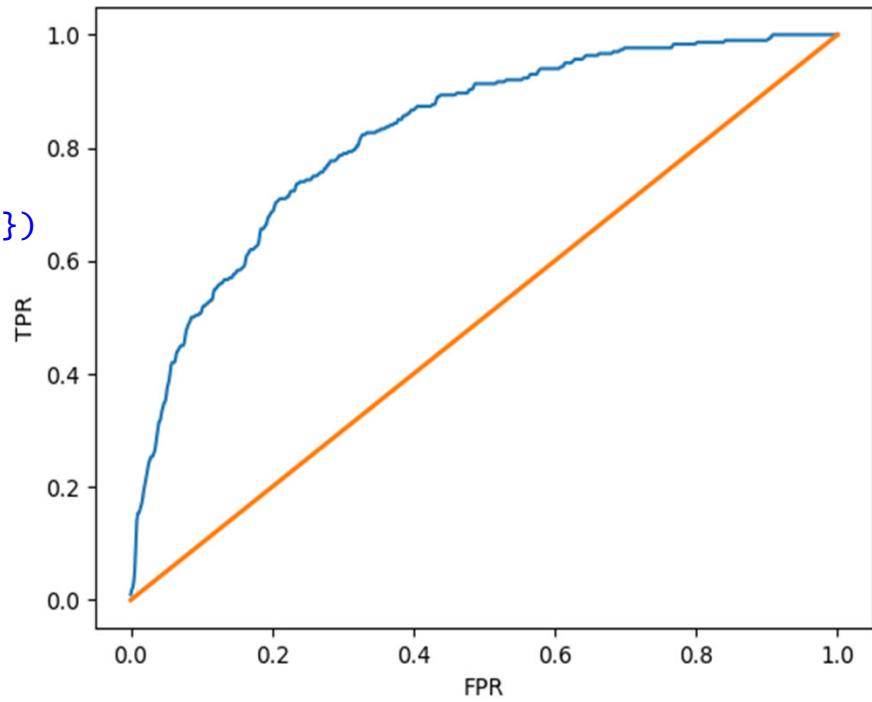
- Equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance.
- Random guess has probability 0.5, which is area under 45 degree line



j. ROC Curve from logistic regression model

Let's construct the ROC curve from the restricted logistic regression model,
where the “positive” in this case is the default prediction

```
def simple_roc(labels,scores):
    labels = np.flip([x for _,x
in sorted(zip(scores,labels),reverse = False)])
    return pd.DataFrame({'TPR':
        np.cumsum(labels)/sum(labels),
        'FPR': np.cumsum(~labels)/sum(~labels)})  
  
glm_simple_roc = simple_roc(loans.default==1,
                             loans.phat)  
  
plt.figure()
ax = sns.lineplot(data=glm_simple_roc,
                  x="FPR", y="TPR")
plt.plot([0, 1], [0, 1], linewidth=2)
```



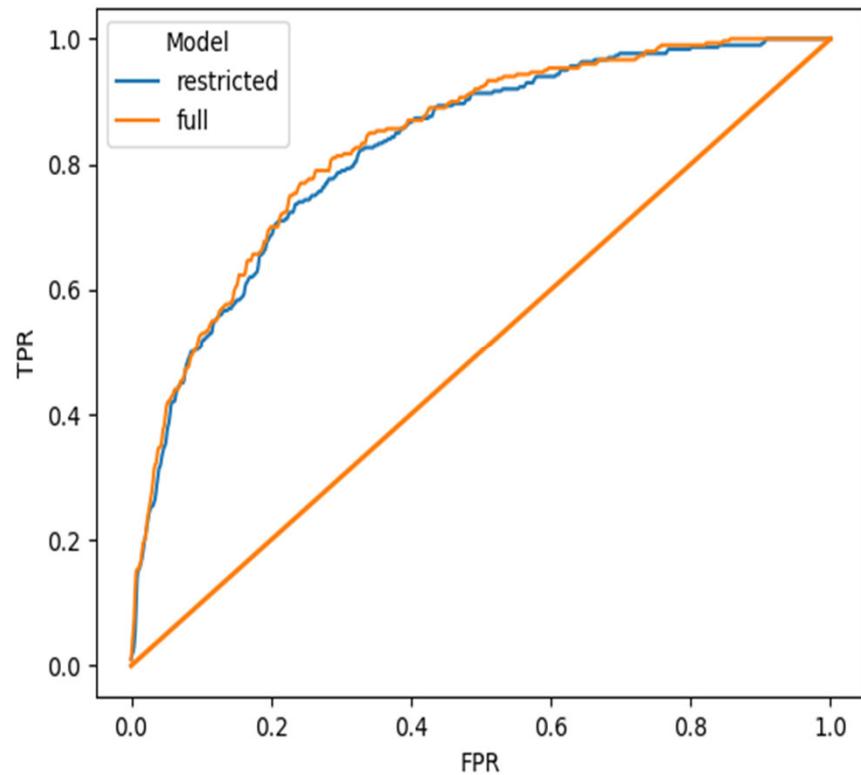
j. ROC Curve from logistic regression model

Let's add the ROC for the full (unrestricted) logistic regression model

- Not much difference, as expected

```
# plot ROC curve for the full model, that uses
# all features
glm_simple_roc_full      =
simple_roc(loans.default==1,phat_full)
glm_simple_roc['Model']    = 'restricted'
glm_simple_roc_full['Model'] = 'full'

New_ROC =
glm_simple_roc.append(glm_simple_roc_full)
plt.figure()
ax = sns.lineplot(data=New_ROC, x="FPR",
                   y="TPR",hue='Model')
plt.plot([0, 1], [0, 1], linewidth=2)
```



j. Bank profit maximization

Example:

Say you are a bank that wants to maximize profits from loans.

- Every time you give a loan and it is paid back you make money
- Every time you give a loan and borrower defaults you lose money

Problem: find FICO score (or probability of not defaulting from a more general model) **cutoff** for giving loan to an applicant that maximizes profits:

$$\max_{\{cutoff\}} \text{NrLoans}(cutoff) \times \text{ExpectedProfitPerLoan}(cutoff)$$

j. Bank profit maximization

With our default prediction model, the natural cutoff is to choose the highest accepted probability of default

- Note that from the bank's perspective, the positive outcome is no default, whereas the positive (high) outcome in our logistic regression was a default
 - I know. This is confusing. But, it's good to note that you have to be careful about these things when you are faced with this type of problem.
- This will often be how models are run/default prediction results are reported.
- Thus, a True Negative (**TN**) from our model is good (loan given, no default), whereas a False Negative (**FN**; loan given, default) is bad!
- Note: these are not *rates* but actual number of cases in the sample
 - The total number of loans you give does matter for your overall profit!
- The maximization problem can then be written:

$$\max_{\{cutoff\}} TN(cutoff) \times Profit_{NoDefault} - FN(cutoff) \times Loss_{Default}$$

j. Profit maximization and ROC curves

How is the ROC curve related to the profit maximization?

- *It is not that directly related*
1. We are not using TP or TN rates in the bank's problem, as the number of loans given matters for profits
 2. We need profit per good outcome and loss per bad outcome in addition to the information provided by the ROC curve

ROC curves are a measure of how informative a given model is relative to

- A random guess for a given cutoff
- Another candidate model (model horse race)

In addition, you can see **where** in the TPR versus FPR space the model performs well or not so well

j. Profit maximization and ROC curves

To see how the ROC curve relates to the profit maximization in our case, rewrite the profit maximization as follows

- *Note: N are total negative (no defaults), P are total positives (default) using the convention from our estimated logistic regression*
- *FP is number of false positives (not rate); TP is number of true positives*
- *FPR is false positive rate; TPR is true positive rate (as in ROC curve)*

$$\begin{aligned}TN &= N - FP = N(1 - FPR) \\FN &= P - TP = P(1 - TPR)\end{aligned}$$

So:

$$\max_{\{cutoff\}} \begin{aligned} &N\{1 - FPR(cutoff)\} \times \text{Profit}_{\text{NoDefault}} \\ &-P\{1 - TPR(cutoff)\} \times \text{Loss}_{\text{Default}} \end{aligned}$$

k. Lending Club



In the next Problem Set, you will work with loan data from a large marketplace peer-to-peer lender: Lending Club

How does an online credit marketplace work?

Lending Club uses technology to operate a credit marketplace at a ***lower cost than traditional bank loan programs***, passing the savings on to borrowers in the form of lower rates and to investors in the form of solid returns. Borrowers who used a personal loan via Lending Club to consolidate debt or pay off high interest credit cards report in a survey that the interest rate on their loan was an average of 30% lower than they were paying on their outstanding debt or credit cards.

k. Lending Club

Lending Club is the world's largest marketplace connecting borrowers and investors, where consumers and small business owners lower the cost of their credit and enjoy a better experience than traditional bank lending, and investors earn attractive risk-adjusted returns.

Here's how it works:

- Customers interested in a loan complete a simple application at LendingClub.com
- Lending Club leverage online data and technology to quickly assess risk, determine a credit rating and assign appropriate interest rates. Qualified applicants receive offers in just minutes and can evaluate loan options with no impact to their credit score
- Investors ranging from individuals to institutions select loans in which to invest and can earn monthly returns
- The entire process is online, using technology to lower the cost of credit and pass the savings back in the form of lower rates for borrowers and solid returns for investors.