Statistics and Machine Learning

Classification I
Logistic regression and Bayes rule

Week 7 03/01 — 03/05

Contents of Week 7

- Data set in classification task
- Logistic function
- Judging a classifier: true positive, true negative, false positive, and false negative.
- Bayes rule

Regression versus classification

Regression: y is continuous number

TV	Radio	Sales
230.1	37.8	22.1
x_tv	x_radio	y_sales

Classification: y is qualitative, discrete or categorical, i.e. (0,1) (yes, no)

Balance	Income	Default
729.5	44361.6	NO
817.1	12106.1	YES
x_balance	x_income	y_default

Reading this week

P.127 - p.138

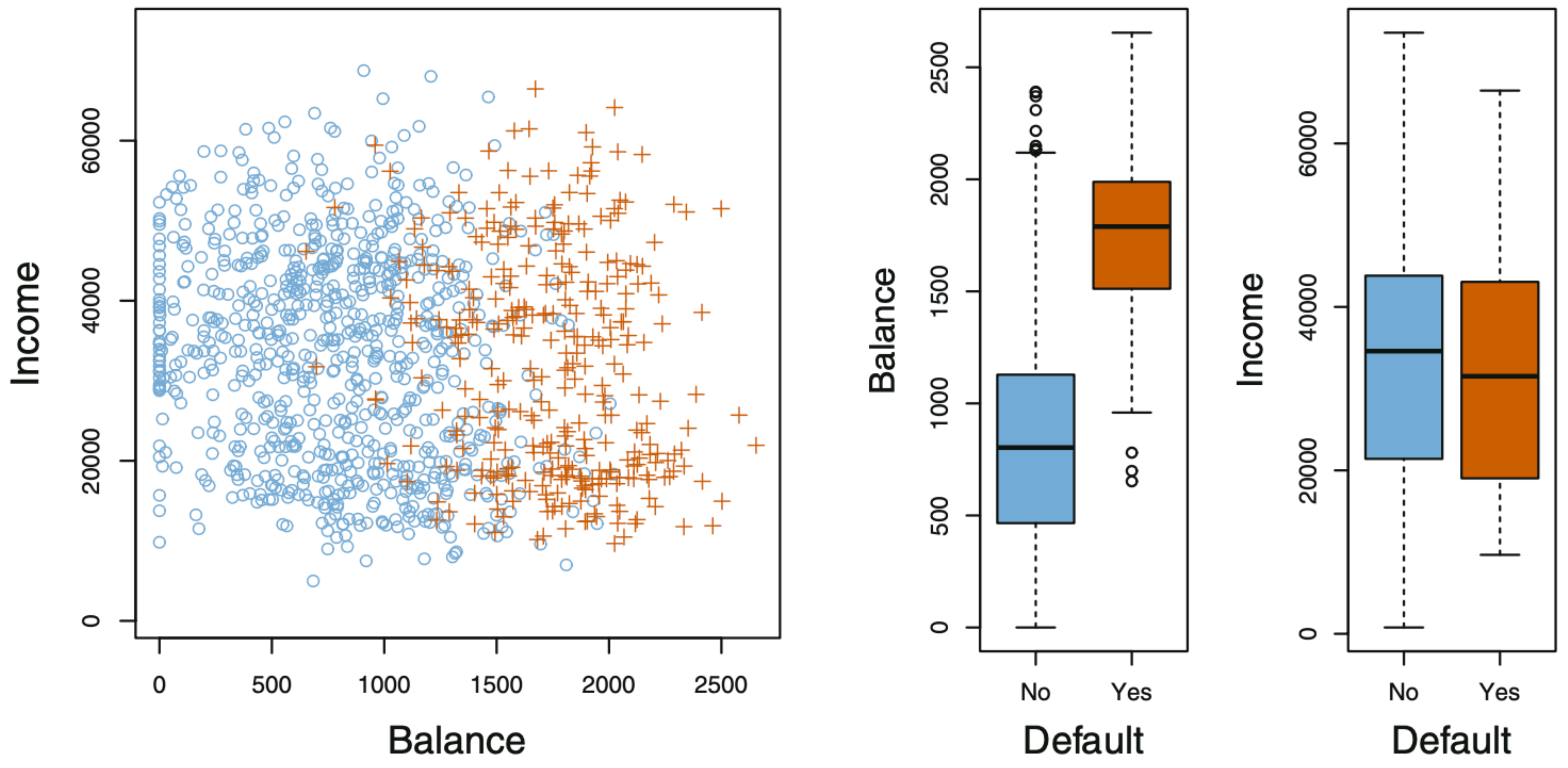


FIGURE 4.1. *The* `Default` *data set.* Left: *The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue.* Center: *Boxplots of* `balance` *as a function of* `default` *status.* Right: *Boxplots of* `income` *as a function of* `default` *status.*

In [2]:
```python
data = pd.read_excel('Default.xlsx')
```

In [3]:
```python
data.head()
```

Out[3]:

	Unnamed: 0	default	student	balance	income
0	1	No	No	729.526495	44361.625074
1	2	No	Yes	817.180407	12106.134700
2	3	No	No	1073.549164	31767.138947
3	4	No	No	529.250605	35704.493935
4	5	No	No	785.655883	38463.495879

In [4]:
```python
type(data)
```

Out[4]:
```
pandas.core.frame.DataFrame
```

In [14]:
```
data['default2'] = data.default.factorize()[0]
```

In [15]:
```
data.head()
```

Out[15]:

	Unnamed: 0	default	student	balance	income	default2
0	1	No	No	729.526495	44361.625074	0
1	2	No	Yes	817.180407	12106.134700	0
2	3	No	No	1073.549164	31767.138947	0
3	4	No	No	529.250605	35704.493935	0
4	5	No	No	785.655883	38463.495879	0

In [17]:
```
X_balance = data.balance.values.reshape(-1,1)
y = data.default2.values.reshape(-1,1)
```

In [14]:
```python
data['default2'] = data.default.factorize()[0]
```

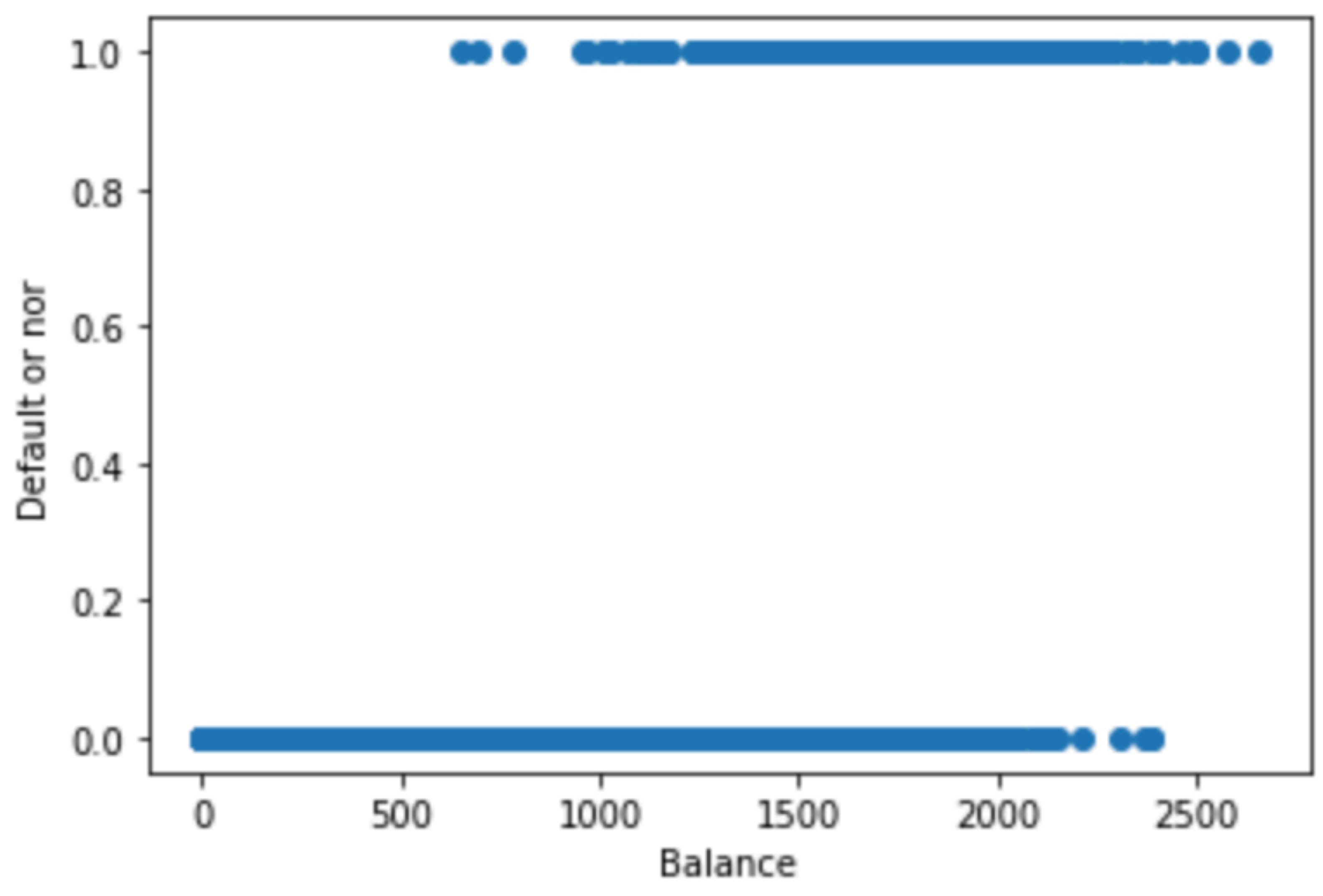
In [15]:
```python
data.head()
```

Out[15]:

	Unnamed: 0	default	student	balance	income	default2
0	1	No	No	729.526495	44361.625074	0
1	2	No	Yes	817.180407	12106.134700	0
2	3	No	No	1073.549164	31767.138947	0
3	4	No	No	529.250605	35704.493935	0
4	5	No	No	785.655883	38463.495879	0

In [17]:
```python
X_balance = data.balance.values.reshape(-1,1)
y = data.default2.values.reshape(-1,1)
```

In [58]:
```python
plt.scatter(X_balance, y)
plt.xlabel('Balance')
plt.ylabel('Default or nor')
plt.show()
```



In [14]:
```python
data['default2'] = data.default.factorize()[0]
```

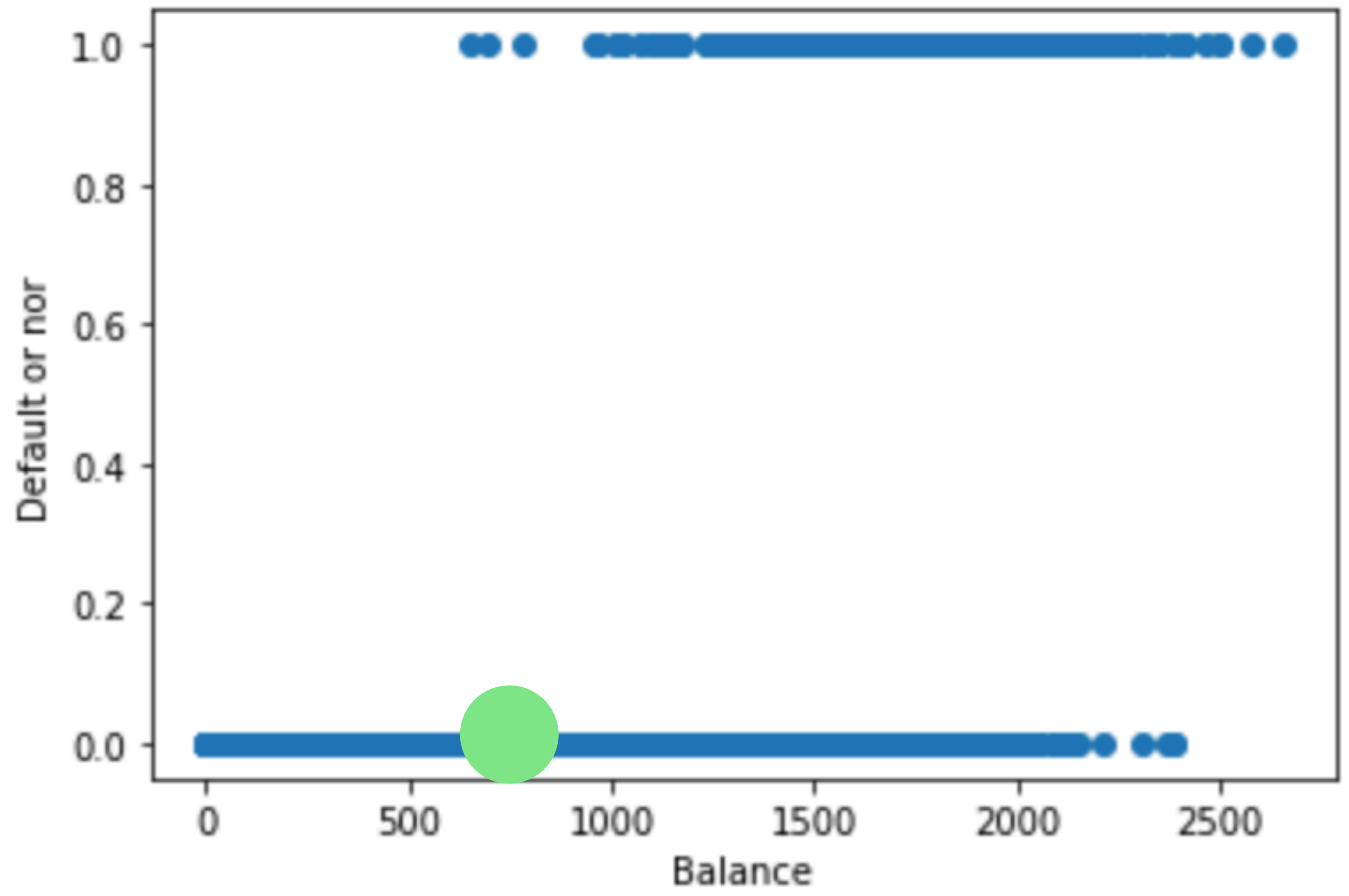
In [15]:
```python
data.head()
```

Out[15]:

	Unnamed: 0	default	student	balance	income	default2
0	1	No	No	729.526495	44361.625074	0
1	2	No	Yes	817.180407	12106.134700	0
2	3	No	No	1073.549164	31767.138947	0
3	4	No	No	529.250605	35704.493935	0
4	5	No	No	785.655883	38463.495879	0

In [17]:
```python
X_balance = data.balance.values.reshape(-1,1)
y = data.default2.values.reshape(-1,1)
```

In [58]:
```python
plt.scatter(X_balance, y)
plt.xlabel('Balance')
plt.ylabel('Default or nor')
plt.show()
```



In [14]:
```
data['default2'] = data.default.factorize()[0]
```

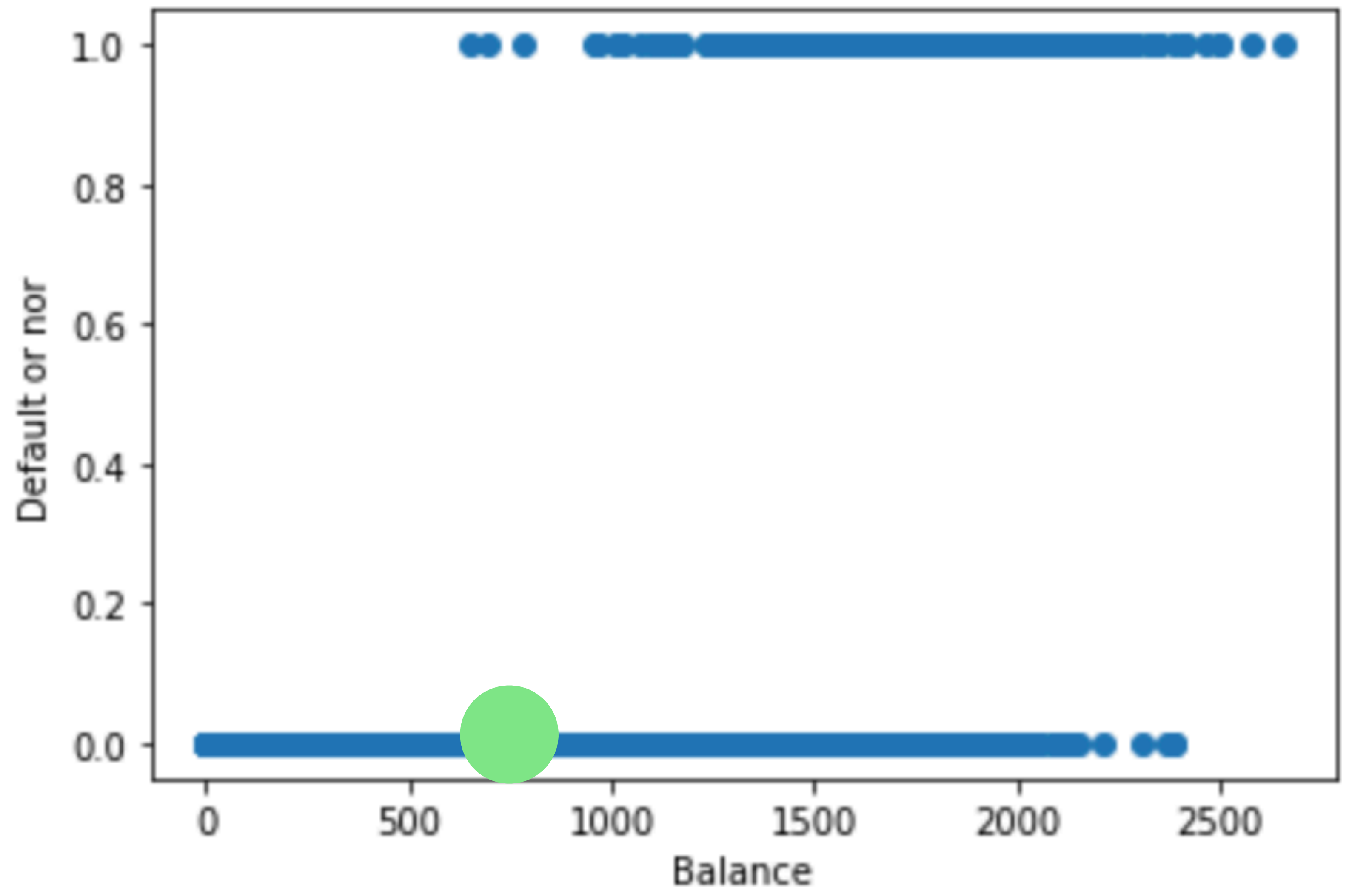
In [15]:
```
data.head()
```

Out[15]:

	Unnamed: 0	default	student	balance	income	default2
0	1	No	No	729.526495	44361.625074	0
1	2	No	Yes	817.180407	12106.134700	0
2	3	No	No	1073.549164	31767.138947	0
3	4	No	No	529.250605	35704.493935	0
4	5	No	No	785.655883	38463.495879	0

In [17]:
```
X_balance = data.balance.values.reshape(-1,1)
y = data.default2.values.reshape(-1,1)
```

In [58]:
```
plt.scatter(X_balance, y)
plt.xlabel('Balance')
plt.ylabel('Default or nor')
plt.show()
```



(Balance = 750, default = 0)

In [14]:
```
data['default2'] = data.default.factorize()[0]
```

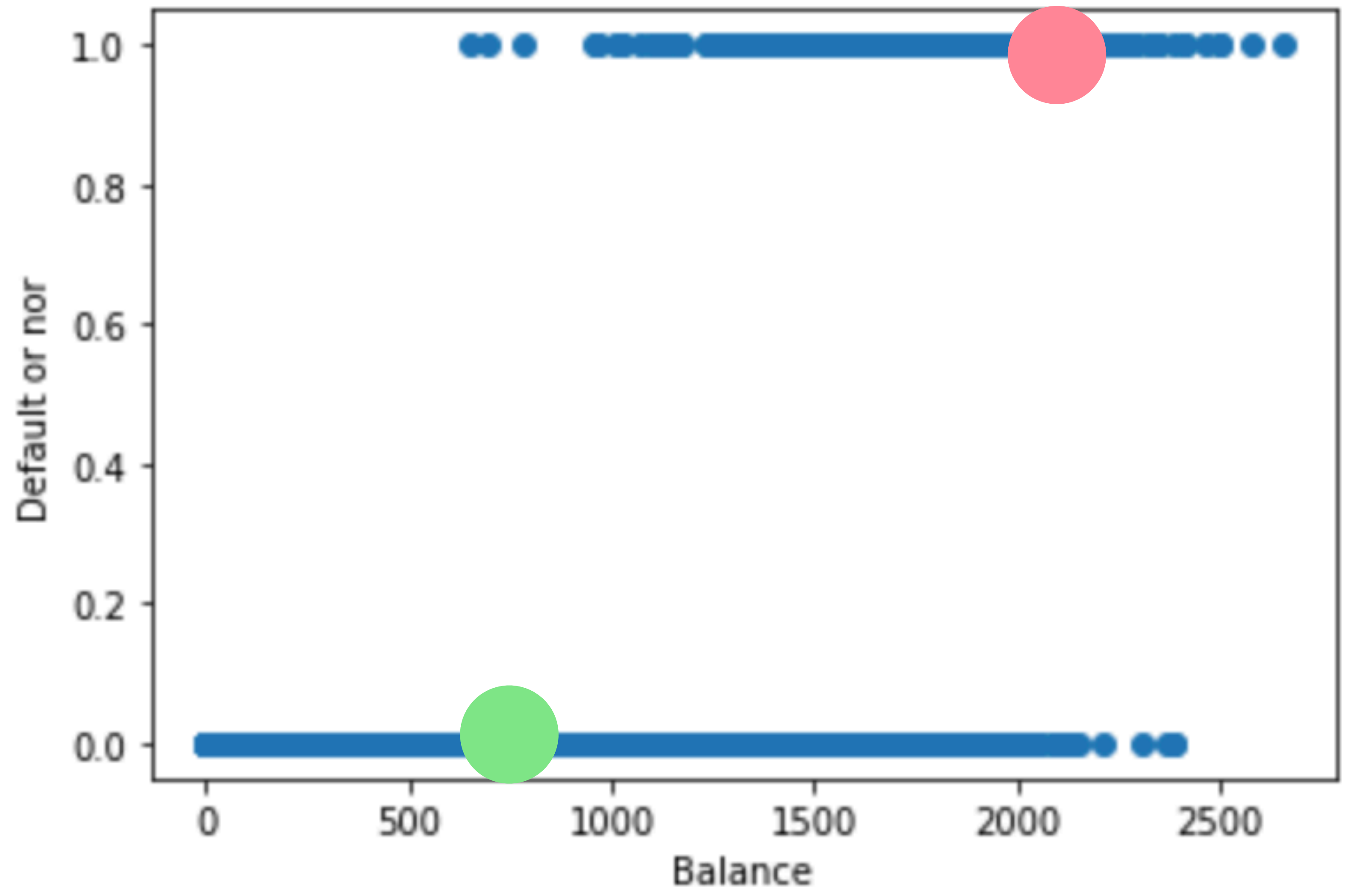
In [15]:
```
data.head()
```

Out[15]:

	Unnamed: 0	default	student	balance	income	default2
0	1	No	No	729.526495	44361.625074	0
1	2	No	Yes	817.180407	12106.134700	0
2	3	No	No	1073.549164	31767.138947	0
3	4	No	No	529.250605	35704.493935	0
4	5	No	No	785.655883	38463.495879	0

In [17]:
```
X_balance = data.balance.values.reshape(-1,1)
y = data.default2.values.reshape(-1,1)
```

In [58]:
```
plt.scatter(X_balance, y)
plt.xlabel('Balance')
plt.ylabel('Default or nor')
plt.show()
```



(Balance = 750, default = 0)

In [14]:
```python
data['default2'] = data.default.factorize()[0]
```

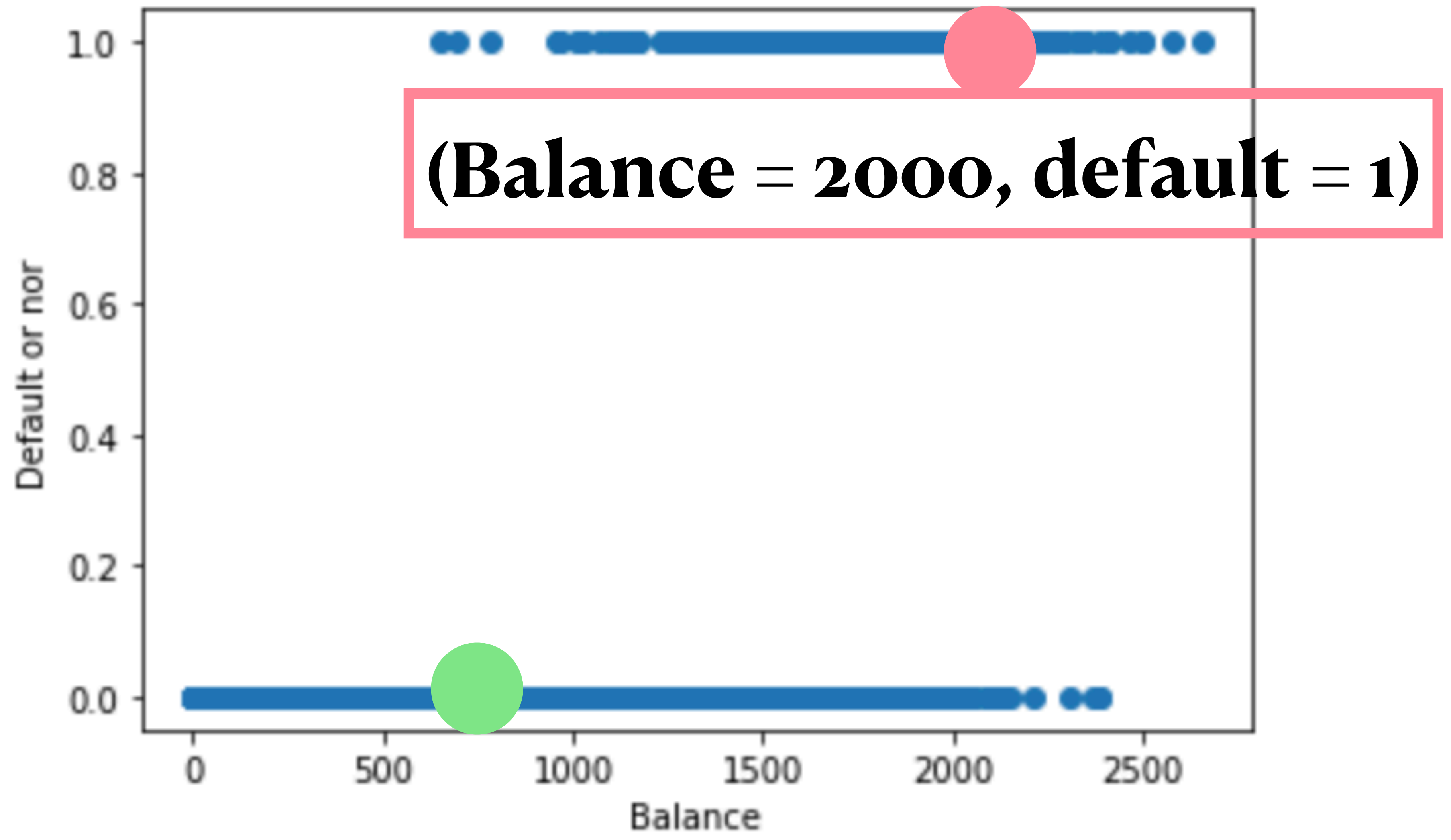
In [15]:
```python
data.head()
```

Out[15]:

	Unnamed: 0	default	student	balance	income	default2
0	1	No	No	729.526495	44361.625074	0
1	2	No	Yes	817.180407	12106.134700	0
2	3	No	No	1073.549164	31767.138947	0
3	4	No	No	529.250605	35704.493935	0
4	5	No	No	785.655883	38463.495879	0

In [17]:
```python
X_balance = data.balance.values.reshape(-1,1)
y = data.default2.values.reshape(-1,1)
```

In [58]:
```python
plt.scatter(X_balance, y)
plt.xlabel('Balance')
plt.ylabel('Default or nor')
plt.show()
```



(Balance = 750, default = 0)

How to deal with categorical output

$-\infty < x < \infty$

$y = 0 \text{ or } 1$

How to deal with categorical output

$$-\infty < x < \infty \qquad\qquad y = 0 \text{ or } 1$$

We can find a function f(x) such that

How to deal with categorical output

$$-\infty < x < \infty \qquad\qquad y = 0 \text{ or } 1$$

We can find a function f(x) such that

$$0 \le f(x) \le 1$$

How to deal with categorical output

$$-\infty < x < \infty \qquad\qquad y = 0 \text{ or } 1$$

We can find a function f(x) such that

$$0 \leq f(x) \leq 1$$

Then at prediction stage:

How to deal with categorical output

$$-\infty < x < \infty \qquad\qquad y = 0 \text{ or } 1$$

We can find a function f(x) such that

$$0 \leq f(x) \leq 1$$

Then at prediction stage:

$$f(x) \geq 0.5 \rightarrow \hat{y} = 1$$

How to deal with categorical output

$$-\infty < x < \infty \qquad\qquad y = 0 \text{ or } 1$$

We can find a function f(x) such that

$$0 \leq f(x) \leq 1$$

Then at prediction stage:

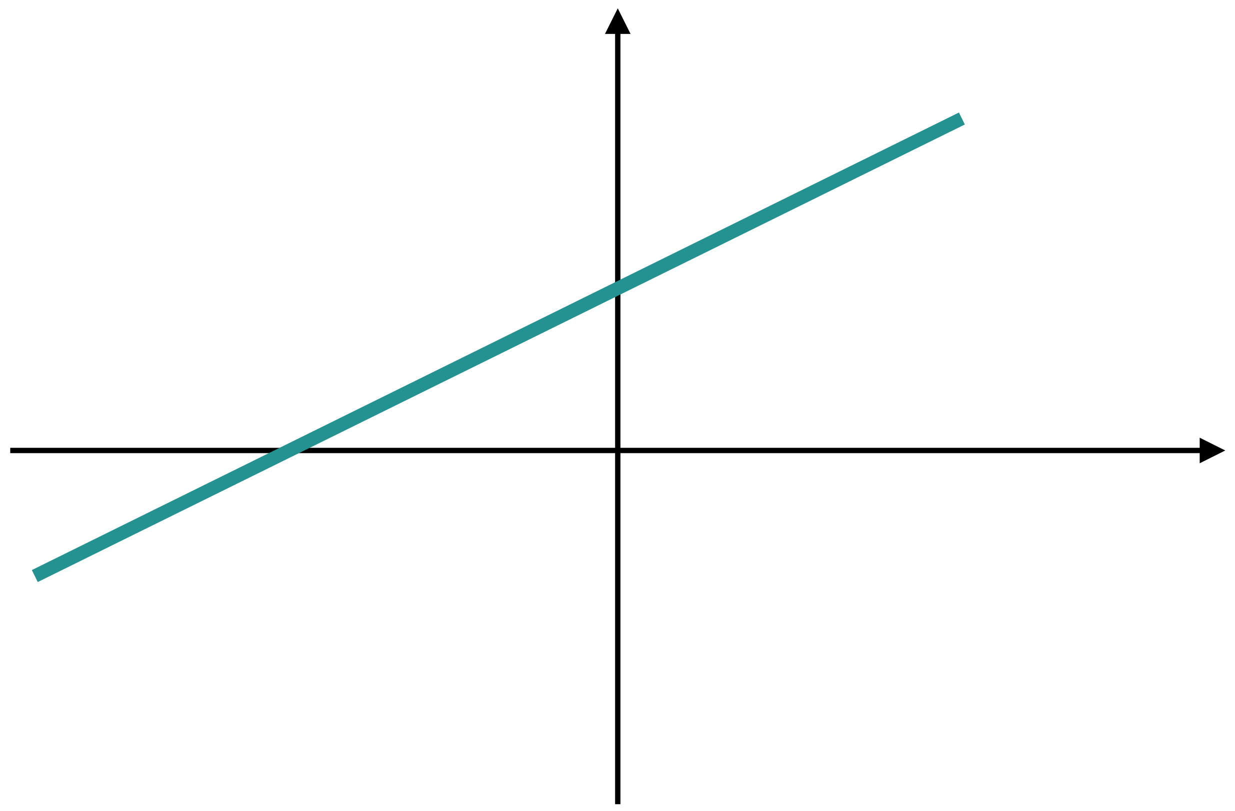
$$f(x) \geq 0.5 \rightarrow \hat{y} = 1$$

$$f(x) \leq 0.5 \rightarrow \hat{y} = 0$$

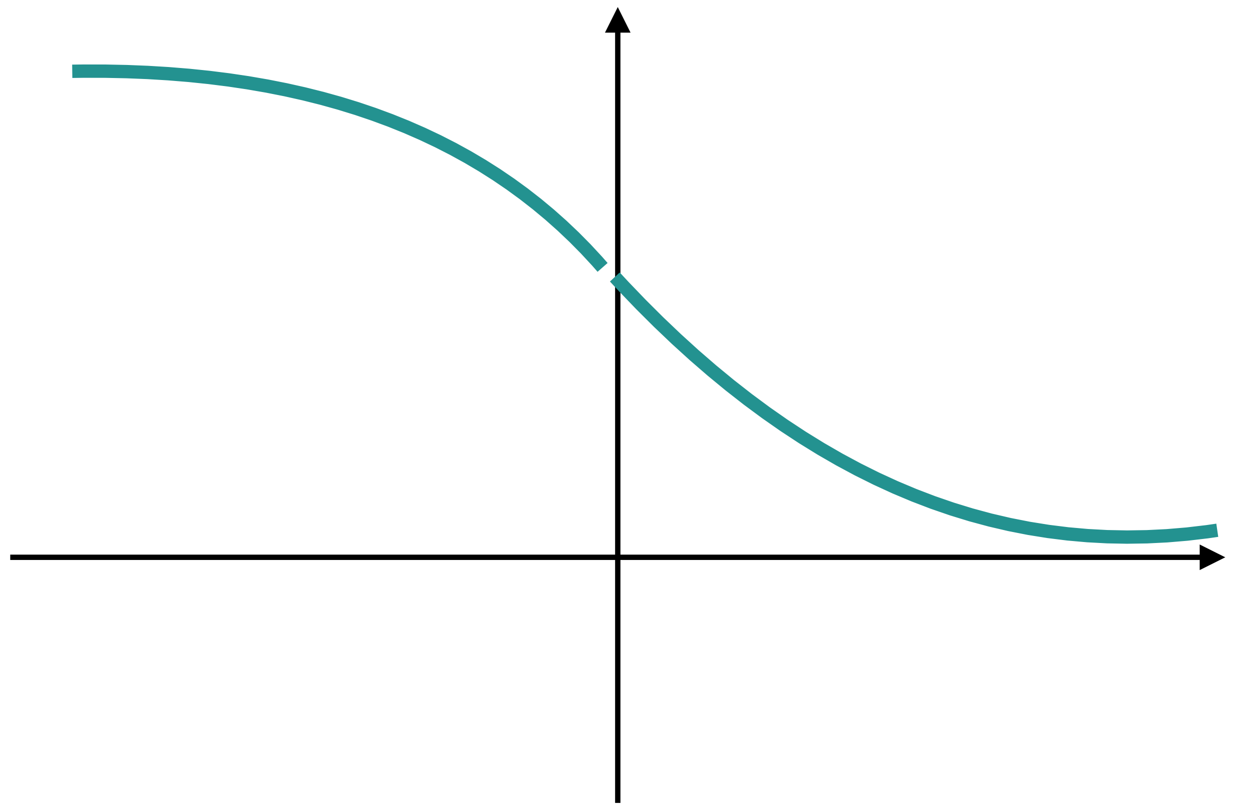
Generalize linear function to logistic function

We want a function whose output is between 0 and 1

$$y = ax + b$$



$$y = \frac{1}{1 + e^{ax+b}}$$



More demos of logistic functions

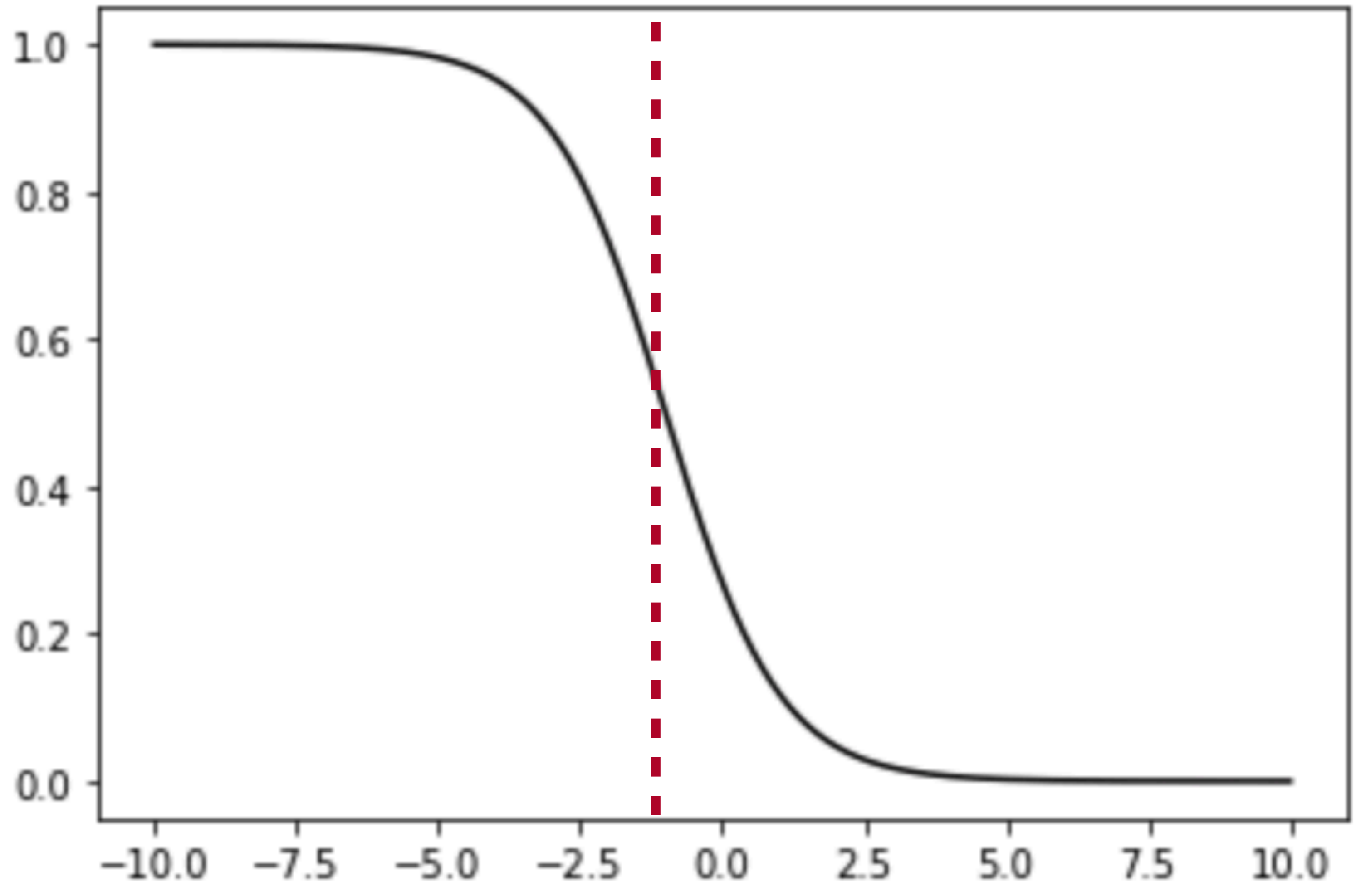
Change a and b values

```
In [6]: def logistic(x,a,b):
            return 1/(1+np.exp(a*x+b))

In [7]: tmpx = np.linspace(-10,10,500)

In [8]: import matplotlib.pyplot as plt

In [9]: plt.plot(tmpx,logistic(tmpx,1,1),'k-')
        plt.show()
```



More demos of logistic functions

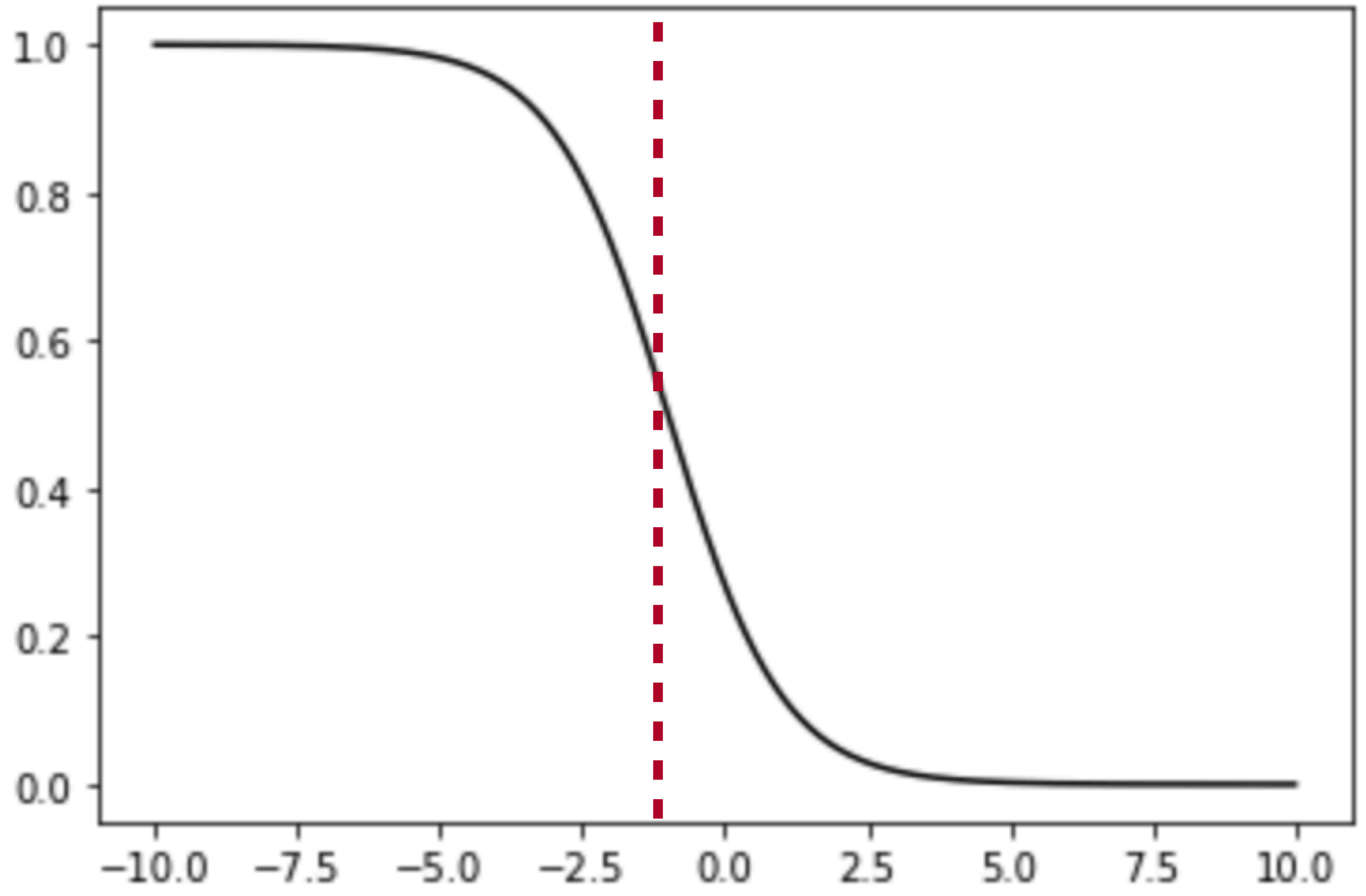
Change a and b values

In [6]:
```
def logistic(x,a,b):
    return 1/(1+np.exp(a*x+b))
```

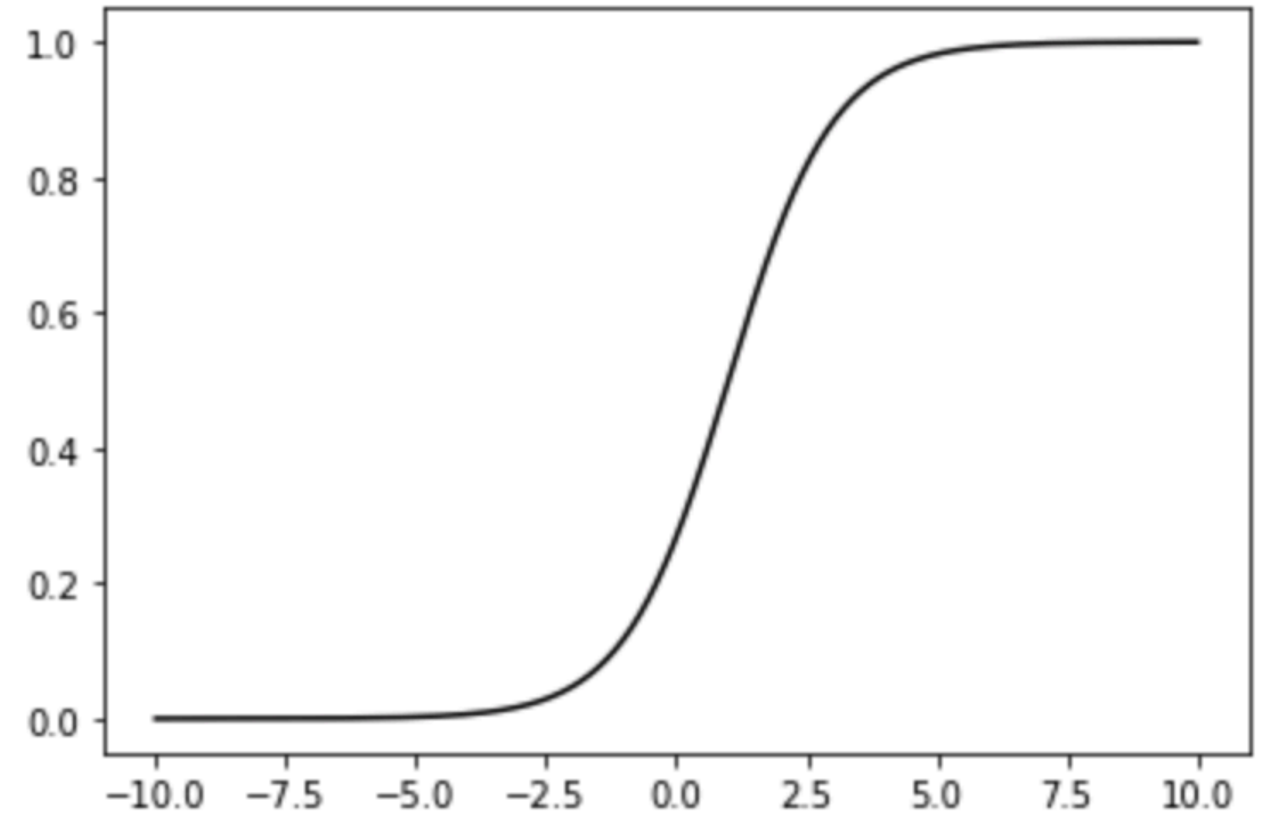
In [7]:
```
tmpx = np.linspace(-10,10,500)
```

In [8]:
```
import matplotlib.pyplot as plt
```

In [9]:
```
plt.plot(tmpx,logistic(tmpx,1,1),'k-')
plt.show()
```



In [10]:
```
plt.plot(tmpx,logistic(tmpx,-1,1),'k-')
plt.show()
```



More demos of logistic functions

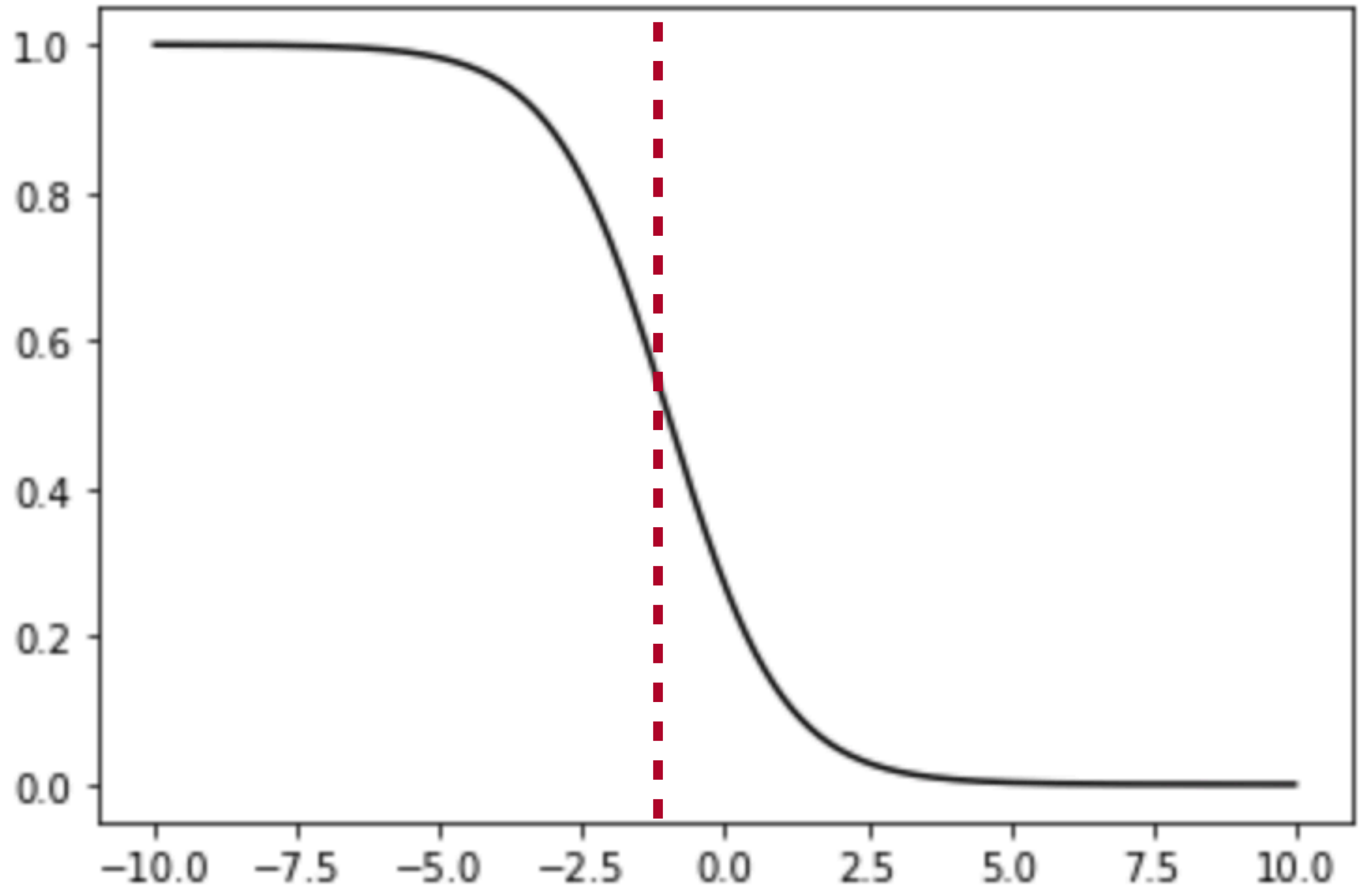
Change a and b values

In [6]:
```
def logistic(x,a,b):
    return 1/(1+np.exp(a*x+b))
```

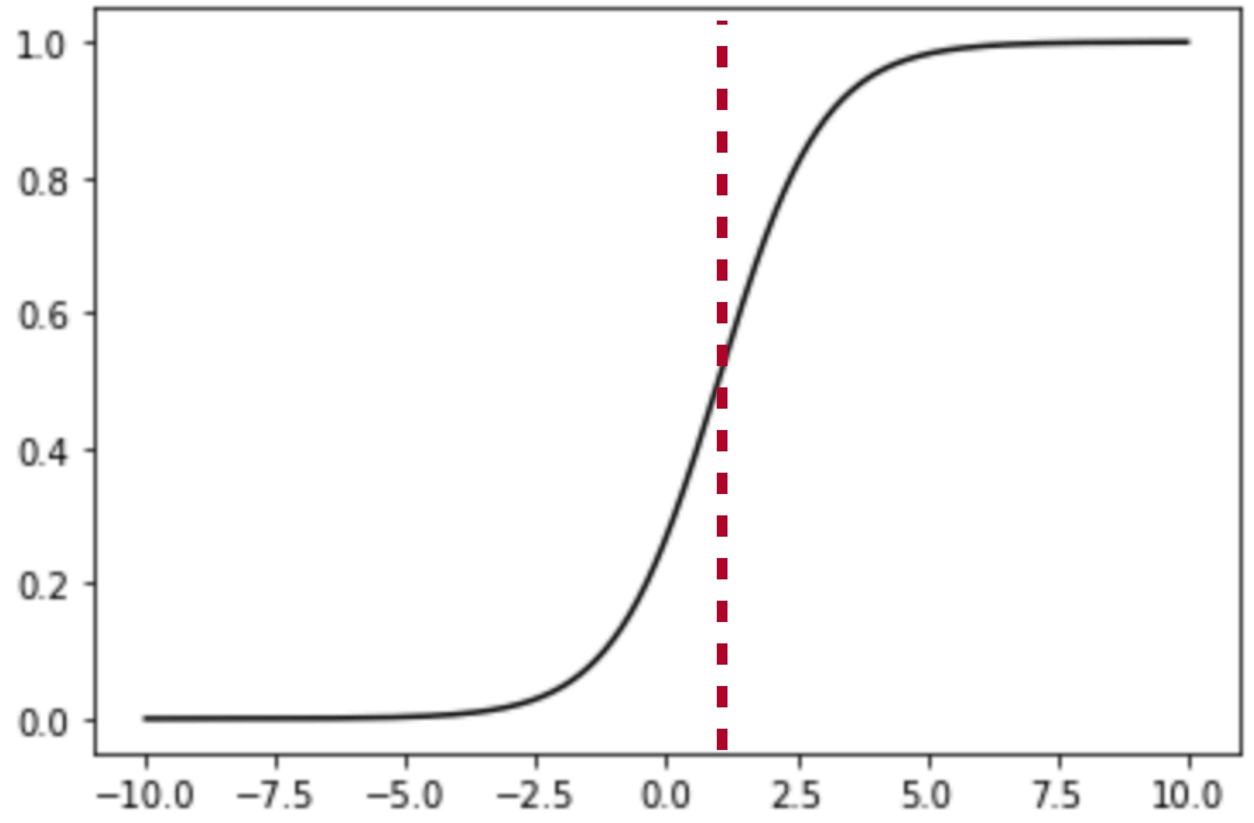
In [7]:
```
tmpx = np.linspace(-10,10,500)
```

In [8]:
```
import matplotlib.pyplot as plt
```

In [9]:
```
plt.plot(tmpx,logistic(tmpx,1,1),'k-')
plt.show()
```



In [10]:
```
plt.plot(tmpx,logistic(tmpx,-1,1),'k-')
plt.show()
```



More demos of logistic functions

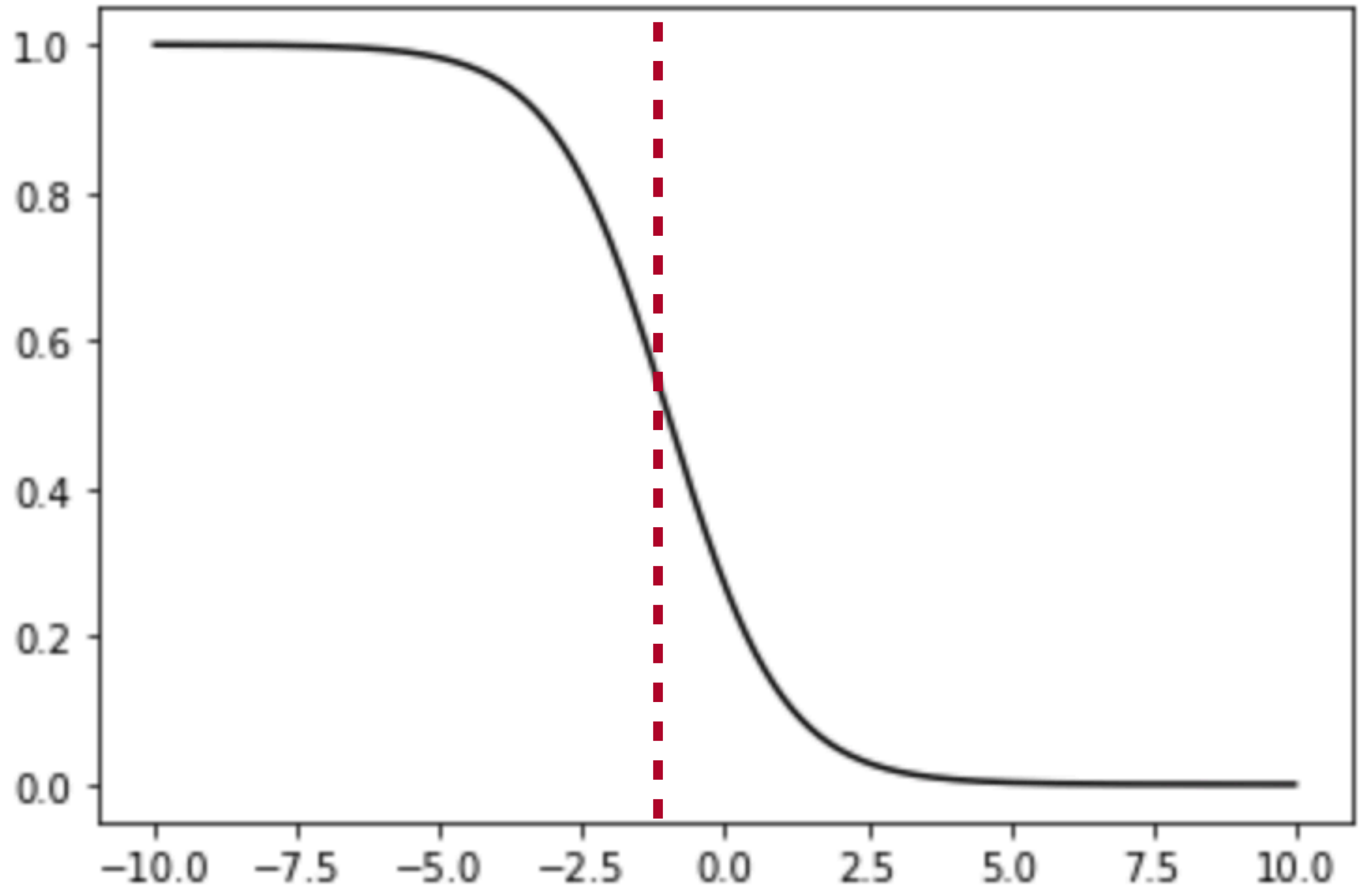
Change a and b values

In [6]:
```
def logistic(x,a,b):
    return 1/(1+np.exp(a*x+b))
```

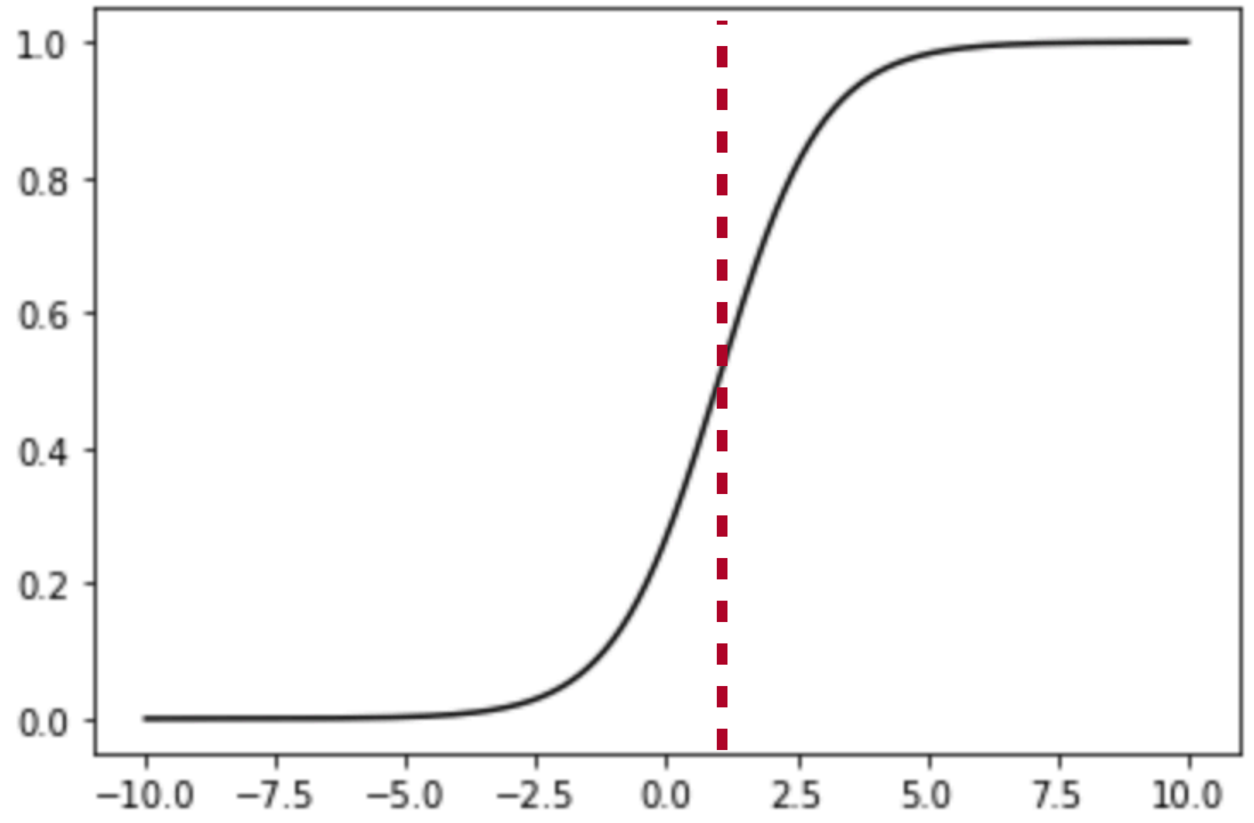
In [7]:
```
tmpx = np.linspace(-10,10,500)
```

In [8]:
```
import matplotlib.pyplot as plt
```

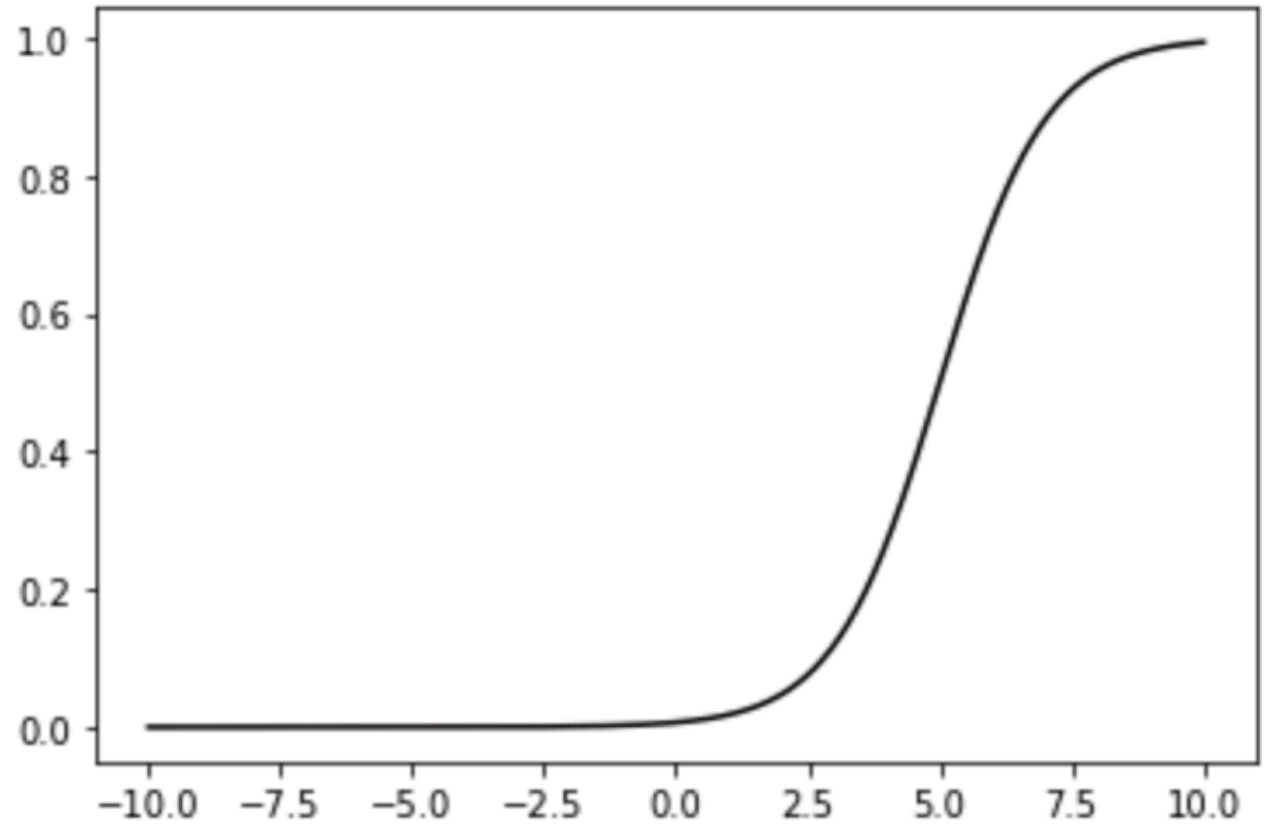
In [9]:
```
plt.plot(tmpx,logistic(tmpx,1,1),'k-')
plt.show()
```



In [10]:
```
plt.plot(tmpx,logistic(tmpx,-1,1),'k-')
plt.show()
```



In [11]:
```
plt.plot(tmpx,logistic(tmpx,-1,5),'k-')
plt.show()
```



More demos of logistic functions

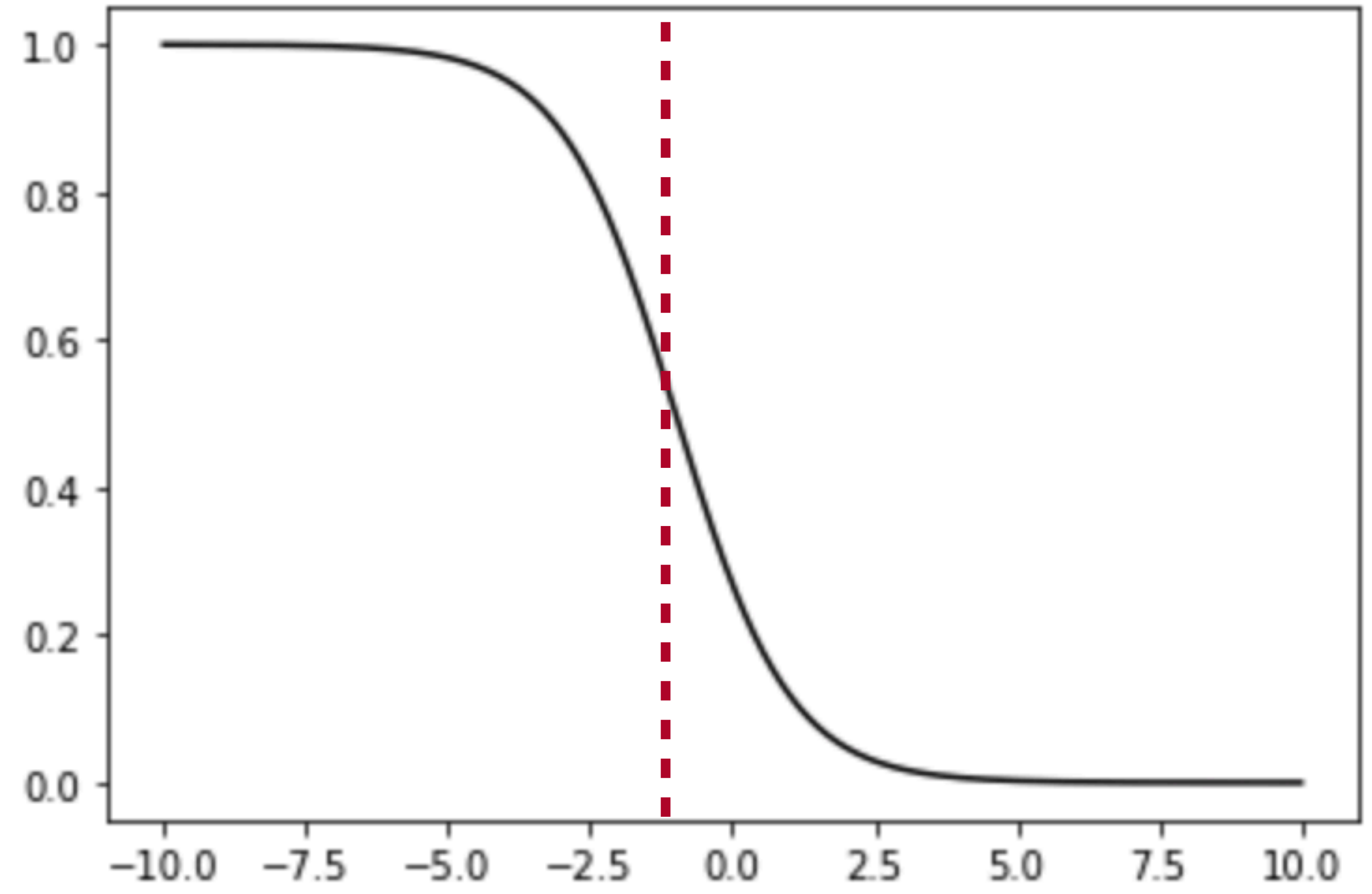
Change a and b values

In [6]:
```python
def logistic(x,a,b):
    return 1/(1+np.exp(a*x+b))
```

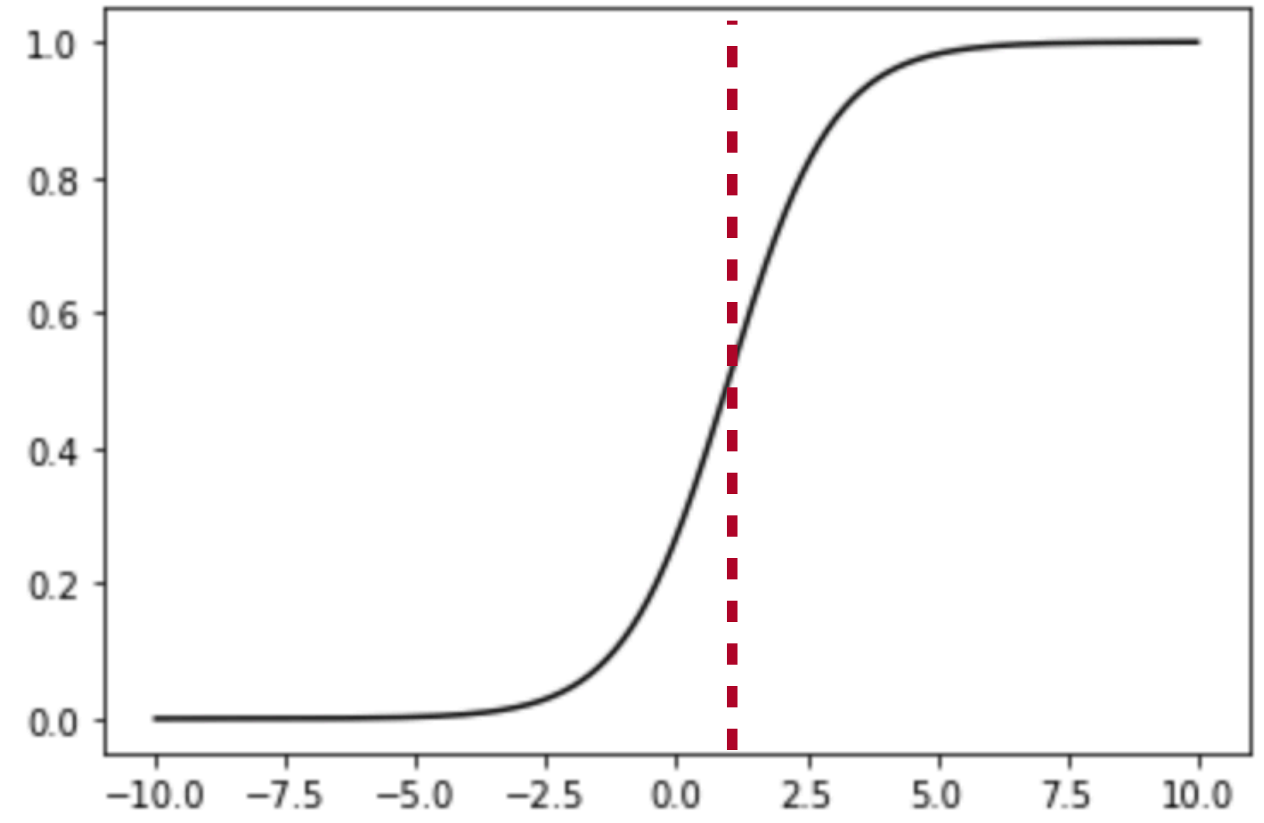
In [7]:
```python
tmpx = np.linspace(-10,10,500)
```

In [8]:
```python
import matplotlib.pyplot as plt
```

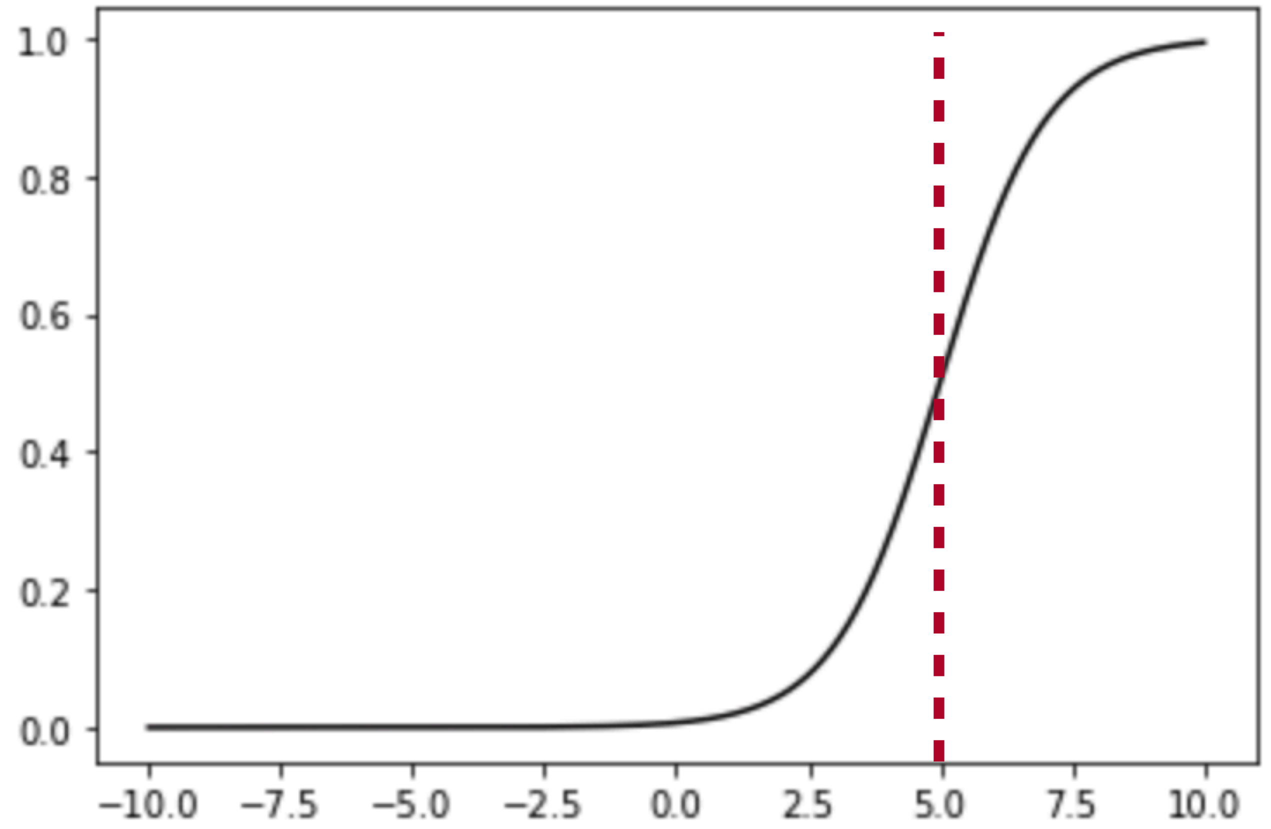
In [9]:
```python
plt.plot(tmpx,logistic(tmpx,1,1),'k-')
plt.show()
```



In [10]:
```python
plt.plot(tmpx,logistic(tmpx,-1,1),'k-')
plt.show()
```



In [11]:
```python
plt.plot(tmpx,logistic(tmpx,-1,5),'k-')
plt.show()
```



sklearn.linear_model.LogisticRegression

class `sklearn.linear_model.`**`LogisticRegression`**(*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) [source]

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the ‘multi_class’ option is set to ‘ovr’, and uses the cross-entropy loss if the ‘multi_class’ option is set to ‘multinomial’. (Currently the ‘multinomial’ option is supported only by the ‘lbfgs’, ‘sag’, ‘saga’ and ‘newton-cg’ solvers.)

Training your first classifier

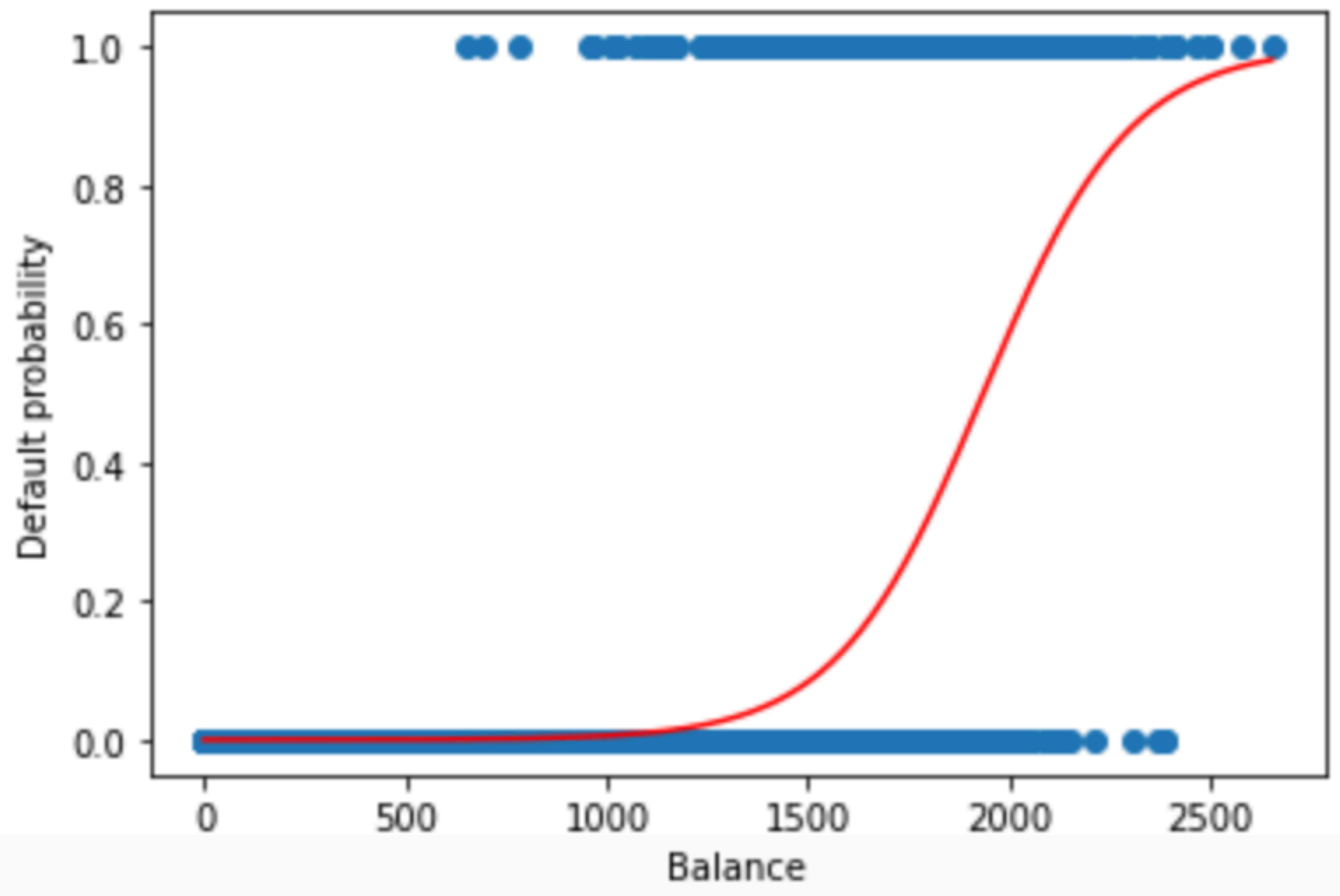
In [22]:
```
import sklearn.linear_model as skl_lm

X_test = np.arange(X_balance.min(), X_balance.max()).reshape(-1,1)
```

In [23]:
```
clf = skl_lm.LogisticRegression(solver='newton-cg')
clf.fit(X_balance, y)
prob = clf.predict_proba(X_test)
```

```
/Users/felix/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:760: DataConversionWarning: A co
lumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
```

In [25]:
```
plt.scatter(X_balance, y)
plt.plot(X_test,prob[:,1],'r-')
plt.xlabel('Balance')
plt.ylabel('Default probability')
plt.show()
```



Training your first classifier

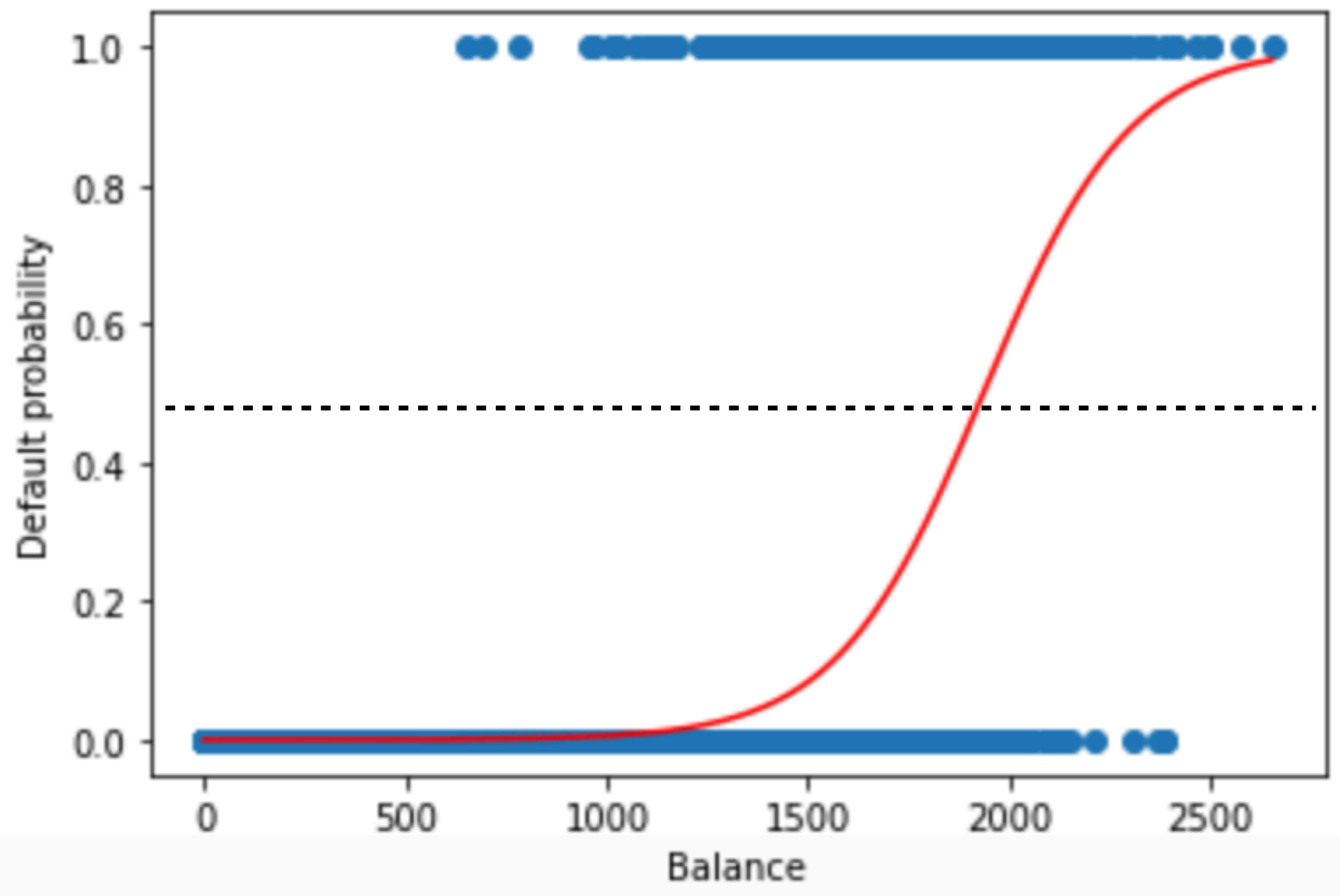
In [22]:
```
import sklearn.linear_model as skl_lm

X_test = np.arange(X_balance.min(), X_balance.max()).reshape(-1,1)
```

In [23]:
```
clf = skl_lm.LogisticRegression(solver='newton-cg')
clf.fit(X_balance, y)
prob = clf.predict_proba(X_test)
```

```
/Users/felix/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:760: DataConversionWarning: A co
lumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
```

In [25]:
```
plt.scatter(X_balance, y)
plt.plot(X_test,prob[:,1],'r-')
plt.xlabel('Balance')
plt.ylabel('Default probability')
plt.show()
```



Training your first classifier

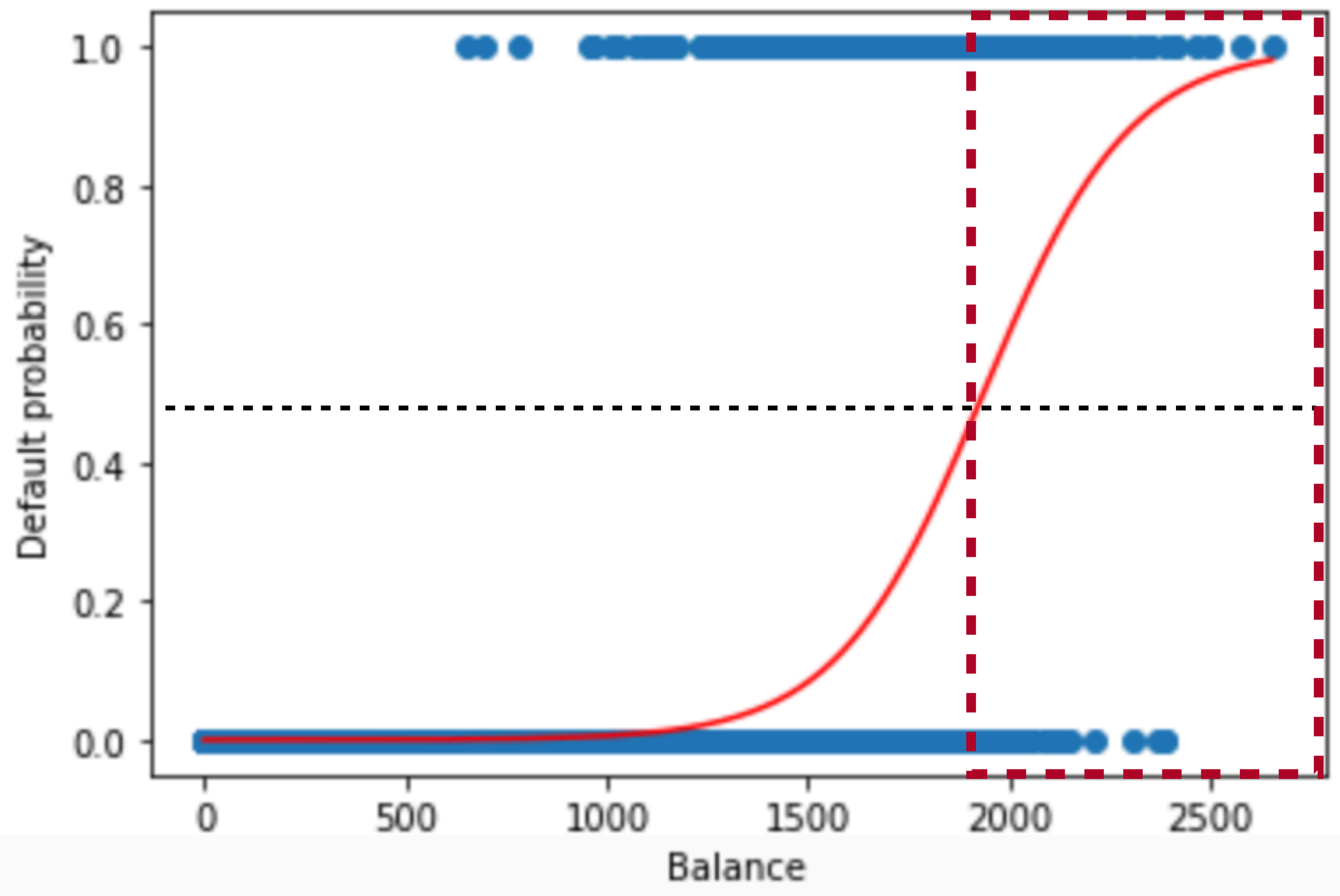
In [22]:
```
import sklearn.linear_model as skl_lm

X_test = np.arange(X_balance.min(), X_balance.max()).reshape(-1,1)
```

In [23]:
```
clf = skl_lm.LogisticRegression(solver='newton-cg')
clf.fit(X_balance, y)
prob = clf.predict_proba(X_test)
```

```
/Users/felix/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:760: DataConversionWarning: A co
lumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
```

In [25]:
```
plt.scatter(X_balance, y)
plt.plot(X_test,prob[:,1],'r-')
plt.xlabel('Balance')
plt.ylabel('Default probability')
plt.show()
```



Training your first classifier

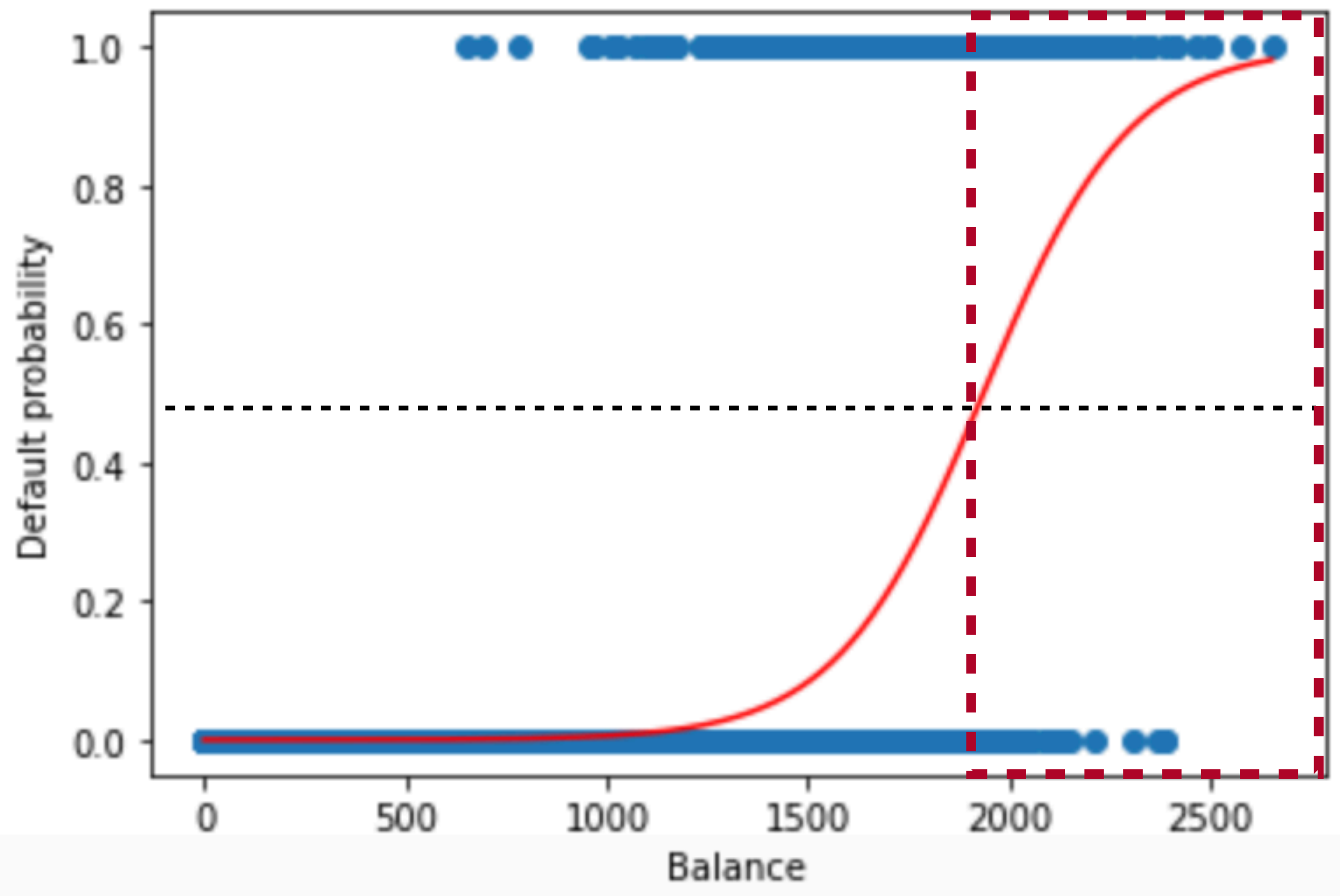
In [22]:
```
import sklearn.linear_model as skl_lm

X_test = np.arange(X_balance.min(), X_balance.max()).reshape(-1,1)
```

In [23]:
```
clf = skl_lm.LogisticRegression(solver='newton-cg')
clf.fit(X_balance, y)
prob = clf.predict_proba(X_test)
```

```
/Users/felix/opt/anaconda3/lib/python3.7/site-packages/sklearn/utils/validation.py:760: DataConversionWarning: A co
lumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
```

In [25]:
```
plt.scatter(X_balance, y)
plt.plot(X_test,prob[:,1],'r-')
plt.xlabel('Balance')
plt.ylabel('Default probability')
plt.show()
```



Around balance > 1900
Predictive to be default

In [26]:
```
print(clf.classes_)
```

```
[0 1]
```

In [27]:
```
print(clf.coef_)
```

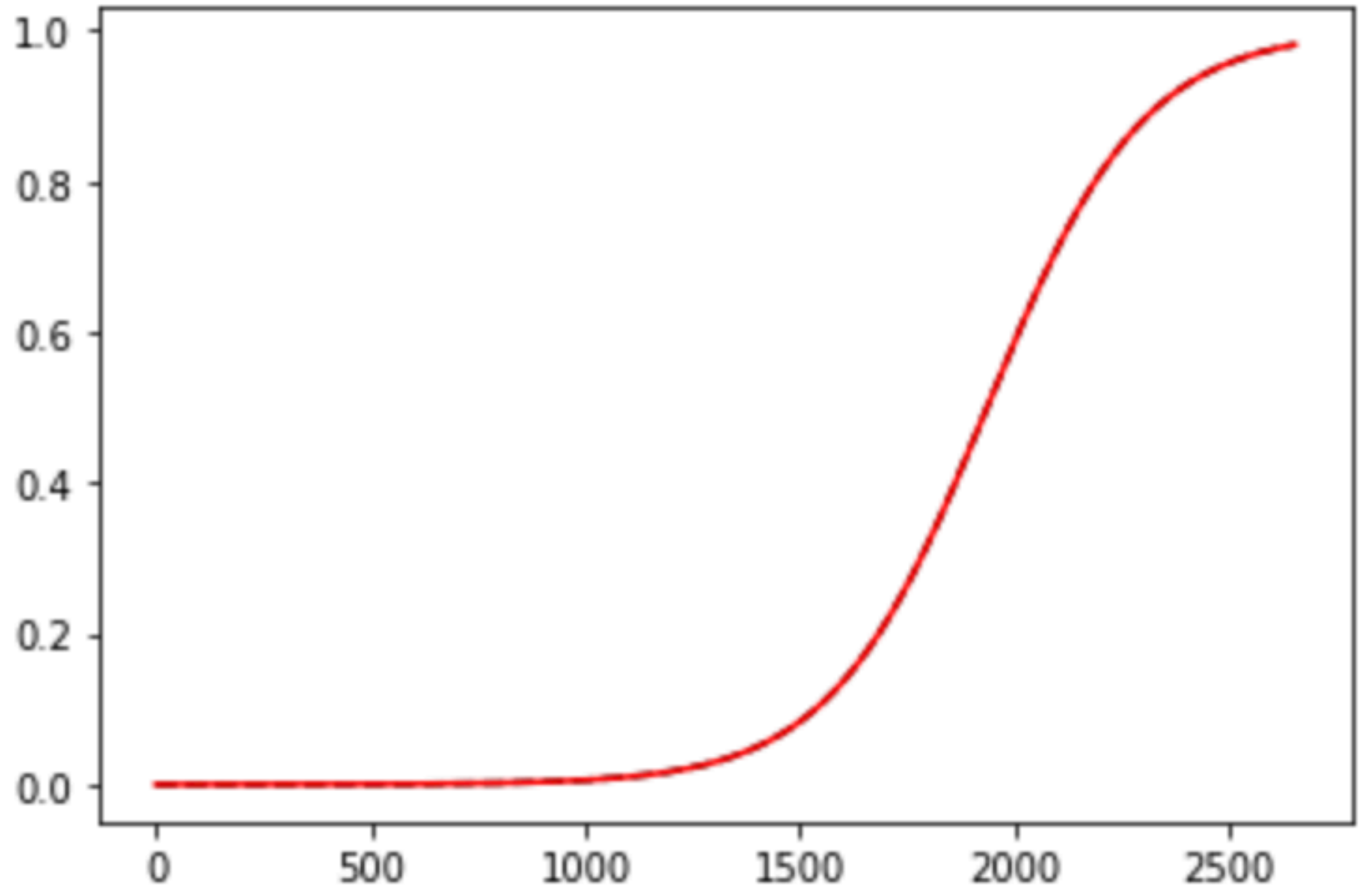
```
[[0.00549891]]
```

In [28]:
```
print(clf.intercept_)
```

```
[-10.65132226]
```

In [29]:
```
plt.plot(X_test,logistic(X_test,(-1.)*clf.coef_,(-1.)*clf.intercept_),'k--')
plt.plot(X_test,prob[:,1],'r-')
```

Out[29]:
```
[<matplotlib.lines.Line2D at 0x7fb8d521b9d0>]
```



In sklearn, the used logistic function is

For single x versus single y

$$y = \frac{1}{1 + e^{-ax-b}}$$

To get the value of a: .coef_
To get the value of b: .intercept_

Important!! Go read and check the textbook in p. 132

To avoid this problem, we must model $p(X)$ using a function that gives outputs between 0 and 1 for all values of X. Many functions meet this description. In logistic regression, we use the *logistic function*,

logistic function

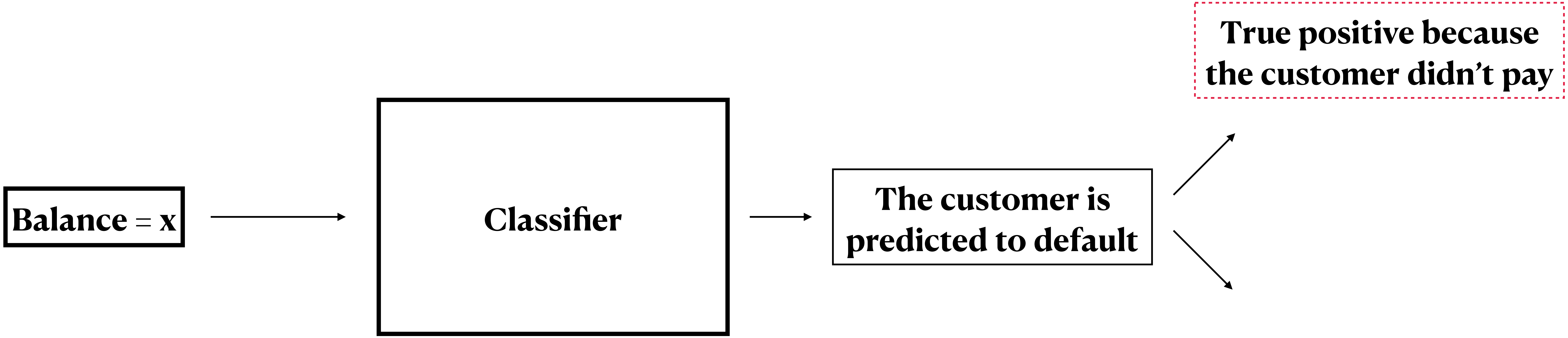
$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}. \quad (4.2)$$

Judge a classifier

True positive, false positive

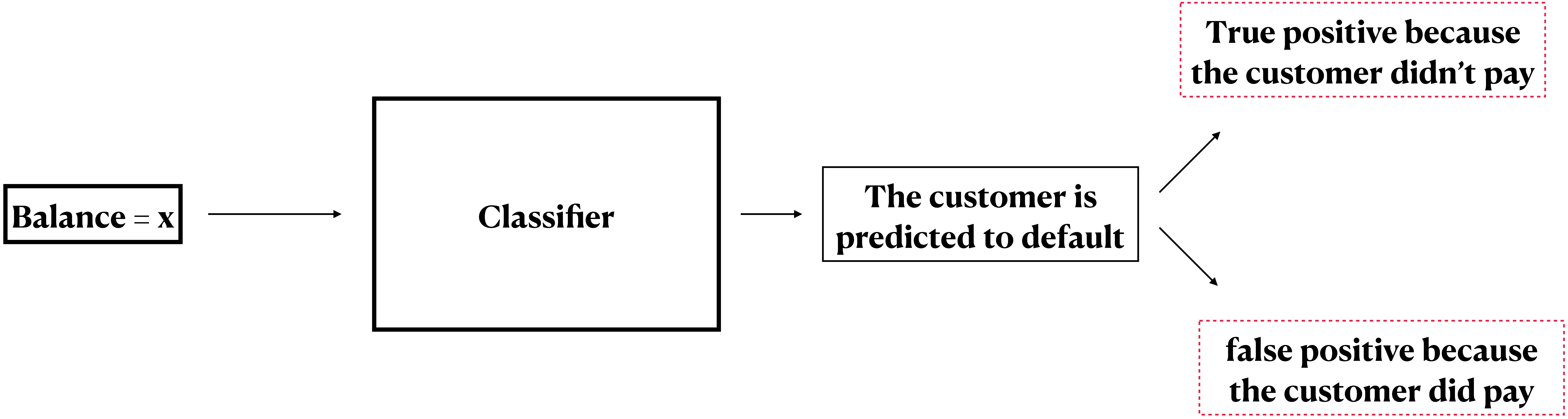
Judge a classifier

True positive, false positive



Judge a classifier

True positive, false positive



Classification report

Precision, recall, and f-1 score

Precision:

$$\frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall:

$$\frac{\text{TP}}{\text{TP} + \text{FN}}$$

F-1 score:

$$2\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Confusion matrix

sklearn.metrics.confusion_matrix

`sklearn.metrics.confusion_matrix`(*y_true*, *y_pred*, ***, *labels=None*, *sample_weight=None*, *normalize=None*) [source]

Compute confusion matrix to evaluate the accuracy of a classification.

By definition a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i and predicted to be in group j.

Thus in binary classification, the count of true negatives is $C_{0,0}$, false negatives is $C_{1,0}$, true positives is $C_{1,1}$ and false positives is $C_{0,1}$.

TN FN

```
In [64]: confusion_matrix(y, y_pred).T

Out[64]: array([[9625,  233],
                [  42,  100]])
```

FP TP

Bayes rule

A practical question

Suppose that a test for using a particular drug is **97% sensitive** and **95% specific**. That is, the test will produce **97% true positive** results for drug users and **95% true negative results** for non-drug users. These are the pieces of data that any screening test will have from their history of tests. Bayes' rule allows us to use this kind of data-driven knowledge to calculate the final probability.

Suppose, we also know that 0.5% of the general population are users of the drug. What is the probability that a randomly selected individual with a positive test is a drug user?

https://towardsdatascience.com/bayes-rule-with-a-simple-and-practical-example-2bce3d0f4ad0

Bayes rule

A practical question

Suppose that a test for using a particular drug is **97% sensitive** and **95% specific**. That is, the test will produce **97% true positive** results for drug users and **95% true negative results** for non-drug users. These are the pieces of data that any screening test will have from their history of tests. Bayes' rule allows us to use this kind of data-driven knowledge to calculate the final probability.

Suppose, we also know that 0.5% of the general population are users of the drug. What is the probability that a randomly selected individual with a positive test is a drug user?

Let us call the variable X:
X = 0 means negative test result
X = 1 means positive test result

https://towardsdatascience.com/bayes-rule-with-a-simple-and-practical-example-2bce3d0f4ad0

Bayes rule

A practical question

Suppose that a test for using a particular drug is **97% sensitive** and **95% specific**. That is, the test will produce **97% true positive** results for drug users and **95% true negative results** for non-drug users. These are the pieces of data that any screening test will have from their history of tests. Bayes' rule allows us to use this kind of data-driven knowledge to calculate the final probability.

Suppose, we also know that 0.5% of the general population are users of the drug. What is the probability that a randomly selected individual with a positive test is a drug user?

Let us call the variable X:
X = 0 means negative test result
X = 1 means positive test result

Let us call the variable Y:
Y = 0 means not a drug user
Y = 1 means a drug user

https://towardsdatascience.com/bayes-rule-with-a-simple-and-practical-example-2bce3d0f4ad0

Two kinds of questions

Why we need to learn probability

Let us call the variable X:
X = 0 means negative test result
X = 1 means positive test result

Type 1: Given y, what is probability for x?
If a drug user goes to test, what is the chance this person will get a positive test result?

Let us call the variable Y:
Y = 0 means not a drug user
Y = 1 means a drug user

Two kinds of questions

Why we need to learn probability

Let us call the variable X:
X = 0 means negative test result
X = 1 means positive test result

Type 1: Given y, what is probability for x?
If a drug user goes to test, what is the chance this person will get a positive test result?

Then we can say:
$p(x=1|y=1) = 0.95$ (sensitivity)
$p(x=0|y=0) = 0.97$ (specificity)

Let us call the variable Y:
Y = 0 means not a drug user
Y = 1 means a drug user

Two kinds of questions

Why we need to learn probability

Let us call the variable X:
X = 0 means negative test result
X = 1 means positive test result

Type 1: Given y, what is probability for x?
If a drug user goes to test, what is the chance this person will get a positive test result?

Then we can say:
$p(x=1|y=1) = 0.95$ (sensitivity)
$p(x=0|y=0) = 0.97$ (specificity)

Additionally, we can say:
$p(x=0|y=1) = 0.05$ (sensitivity)
$p(x=1|y=0) = 0.03$ (specificity)

Let us call the variable Y:
Y = 0 means not a drug user
Y = 1 means a drug user

Two kinds of questions

Why we need to learn probability

Let us call the variable X:
X = 0 means negative test result
X = 1 means positive test result

Type 1: Given y, what is probability for x?
If a drug user goes to test, what is the chance this person will get a positive test result?

Then we can say:
$p(x=1|y=1) = 0.95$ (sensitivity)
$p(x=0|y=0) = 0.97$ (specificity)

Additionally, we can say:
$p(x=0|y=1) = 0.05$ (sensitivity)
$p(x=1|y=0) = 0.03$ (specificity)

Let us call the variable Y:
Y = 0 means not a drug user
Y = 1 means a drug user

Type 2: Given x, what is probability for y?
If one person receive a positive test result, what is the chance this person is truly a drug user?

The Bayes rule

$$p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)}$$

The Bayes rule

$$p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)}$$

The prior probability $p(y)$:
If we choose a random person without doing any test, the chance that this person is drug user is $p(y=1)$.
The chance that this person is not drug user is

$p(y=0) = 1 - p(y=1)$

The Bayes rule

$$p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)}$$

The prior probability $p(y)$:
If we choose a random person without doing any test, the chance that this person is drug user is $p(y = 1)$.
The chance that this person is not drug user is

$$p(y = 0) = 1 - p(y = 1)$$

From the problem text, we learn that

$$p(y = 1) = 0.005$$

The Bayes rule

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

The prior probability $p(y)$:
If we choose a random person without doing any test, the chance that this person is drug user is $p(y=1)$.
The chance that this person is not drug user is
$p(y=0) = 1 - p(y=1)$

The evidence $p(x)$:
The "fact" is that a guy got a positive result, it could mean 1) this guy is drug user, and the test got him; or 2) this guy is not drug user, but the test failed him.

From the problem text, we learn that
$p(y=1) = 0.005$

The Bayes rule

$$p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)}$$

The prior probability $p(y)$:
If we choose a random person without doing any test, the chance that this person is drug user is $p(y=1)$.
The chance that this person is not drug user is

$$p(y=0) = 1 - p(y=1)$$

The evidence $p(x)$:
The "fact" is that a guy got a positive result, it could mean 1) this guy is drug user, and the test got him; or 2) this guy is not drug user, but the test failed him.

So we can follow the general probability rules:

$$p(x=1) = p(x=1 \mid y=1)p(y=1) + p(x=1 \mid y=0)p(y=0)$$

From the problem text, we learn that

$$p(y=1) = 0.005$$

The Bayes rule

$$p(y\,|\,x) = \frac{p(x\,|\,y)p(y)}{p(x)}$$

The prior probability $p(y)$:
If we choose a random person without doing any test, the chance that this person is drug user is $p(y=1)$.
The chance that this person is not drug user is
$p(y=0) = 1 - p(y=1)$

The evidence $p(x)$:
The "fact" is that a guy got a positive result, it could mean 1) this guy is drug user, and the test got him; or 2) this guy is not drug user, but the test failed him.

From the problem text, we learn that
$p(y=1) = 0.005$

So we can follow the general probability rules:
$$p(x=1) = p(x=1\,|\,y=1)p(y=1) + p(x=1\,|\,y=0)p(y=0)$$

sensitivity
0.97

The Bayes rule

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

The prior probability $p(y)$**:**
If we choose a random person without doing any test, the chance that this person is drug user is $p(y=1)$**.**
The chance that this person is not drug user is
$p(y=0) = 1 - p(y=1)$

The evidence $p(x)$ **:**
The "fact" is that a guy got a positive result, it could mean 1) this guy is drug user, and the test got him; or 2) this guy is not drug user, but the test failed him.

From the problem text, we learn that
$p(y=1) = 0.005$

So we can follow the general probability rules:
$$p(x=1) = p(x=1|y=1)p(y=1) + p(x=1|y=0)p(y=0)$$

sensitivity	prior
0.97	0.005

The Bayes rule

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

The prior probability $p(y)$:
If we choose a random person without doing any test, the chance that this person is drug user is $p(y=1)$.
The chance that this person is not drug user is

$$p(y=0) = 1 - p(y=1)$$

The evidence $p(x)$:
The "fact" is that a guy got a positive result, it could mean 1) this guy is drug user, and the test got him; or 2) this guy is not drug user, but the test failed him.

From the problem text, we learn that

$$p(y=1) = 0.005$$

So we can follow the general probability rules:

$$p(x=1) = p(x=1|y=1)p(y=1) + p(x=1|y=0)p(y=0)$$

sensitivity	prior	Opposite of specificity:
0.97	0.005	1-0.95 = 0.05

Given a person who got a positive test result, the probability that he really is a drug user is:

Given a person who got a positive test result, the probability that he really is a drug user is:

$$p(y = 1 \mid x = 1) = \frac{p(x = 1 \mid y = 1)p(y = 1)}{p(x = 1)}$$

Given a person who got a positive test result, the probability that he really is a drug user is:

$$p(y=1 \mid x=1) = \frac{p(x=1 \mid y=1)p(y=1)}{p(x=1)}$$

$$= \frac{0.97 * 0.005}{0.97 * 0.005 + 0.05 * 0.995}$$

Given a person who got a positive test result, the probability that he really is a drug user is:

$$p(y=1 \mid x=1) = \frac{p(x=1 \mid y=1)p(y=1)}{p(x=1)}$$

$$= \frac{0.97 * 0.005}{0.97 * 0.005 + 0.05 * 0.995}$$

$$= \frac{1}{1 + \frac{995 * 5}{97 * 5}} \approx 1/11 \approx 0.09$$

Why we need to know Bayes rule in Machine learning

Discriminative versus generative

Why we need to know Bayes rule in Machine learning

Discriminative versus generative

In Default data set, we are concerned with:
X = balance; Y = default.
And we build a discriminative model:

$$p(y = 1 \mid x) = \frac{1}{1 + e^{-(wx+b)}}$$

Why we need to know Bayes rule in Machine learning

Discriminative versus generative

In Default data set, we are concerned with:
X = balance; Y = default.
And we build a discriminative model:

$$p(y = 1 \mid x) = \frac{1}{1 + e^{-(wx+b)}}$$

In next week, we will talk about the method from the other way around:

$$p(y = 1 \mid x) = \frac{p(x \mid y = 1)p(y = 1) + p(x \mid y = 0)p(y = 0)}{p(x)}$$