LOGISTIC CLASSIFICATION

1/28/25

Prof. Mehrdad Pirnia

Winter 2025

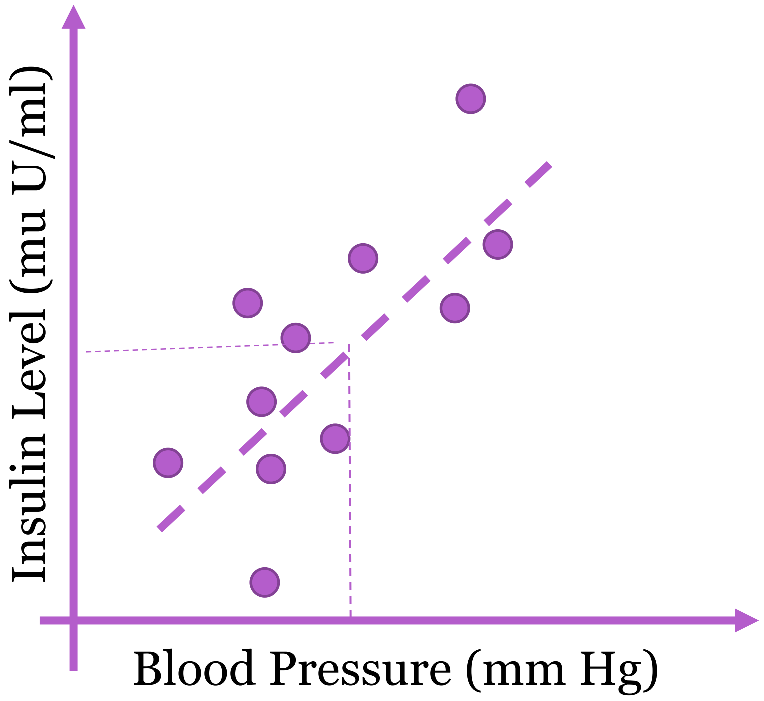
This content is based on concepts from "Introduction to Statistical Learning" by James, Witten, Hastie, and Tibshirani.

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

Lecture Outcomes

- Use regression for classification problems
- Derive Logistic function
- Evaluate the performance of classification problems
- Use Python to model logistic regression

Recall: Linear Regression

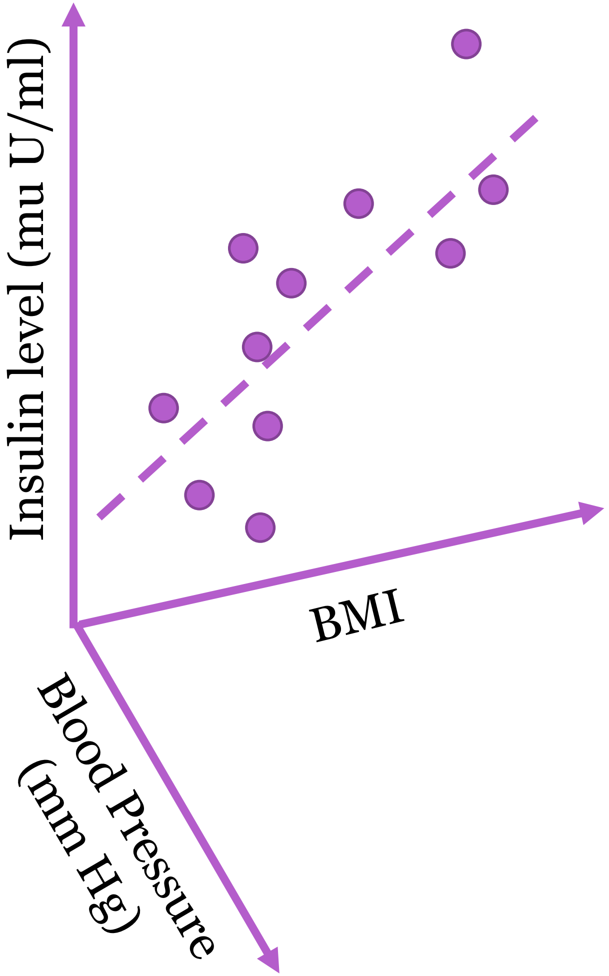


Found the correlation among input variables using R^2

Found the significance of R^2 value using *P-value*

Predict the amount of one variable given the value of the other variable

Recall: Linear Regression



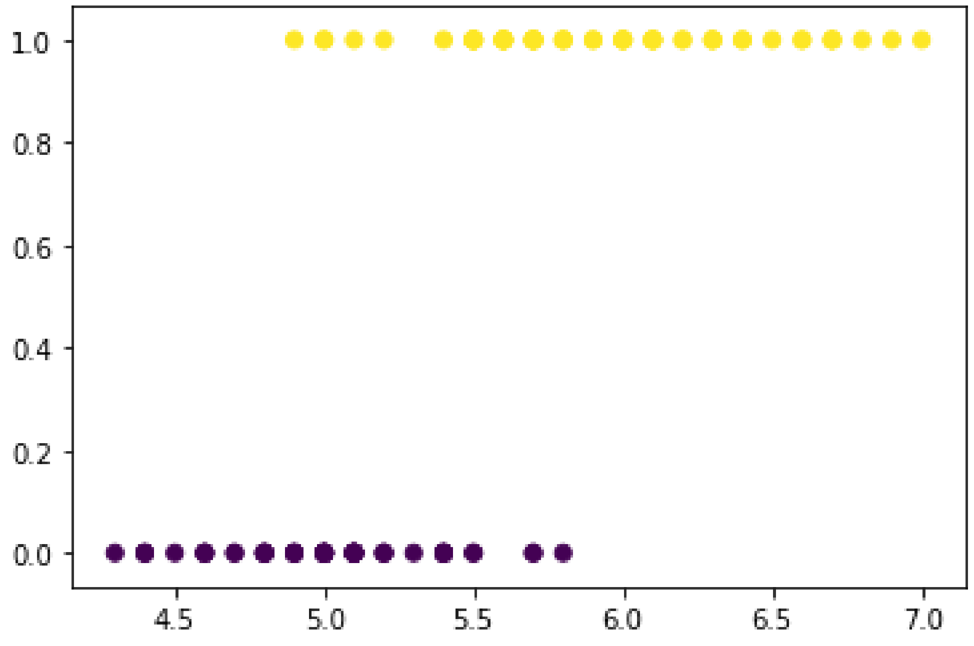
Multiple regression predicts insulin level using more than one feature

For example predicting insulin level using blood pressure and BMI

Use R^2 to determine correlation among input variables

Use *P-value* to find the significance of R^2

Regression for Categorical Data



The data is categorized into different classes: true/false, eye color, obese/non-obese, cancer/no cancer.

Regression is to predict the correct classes based on given values of feature(s).

Linear Regression for Classification

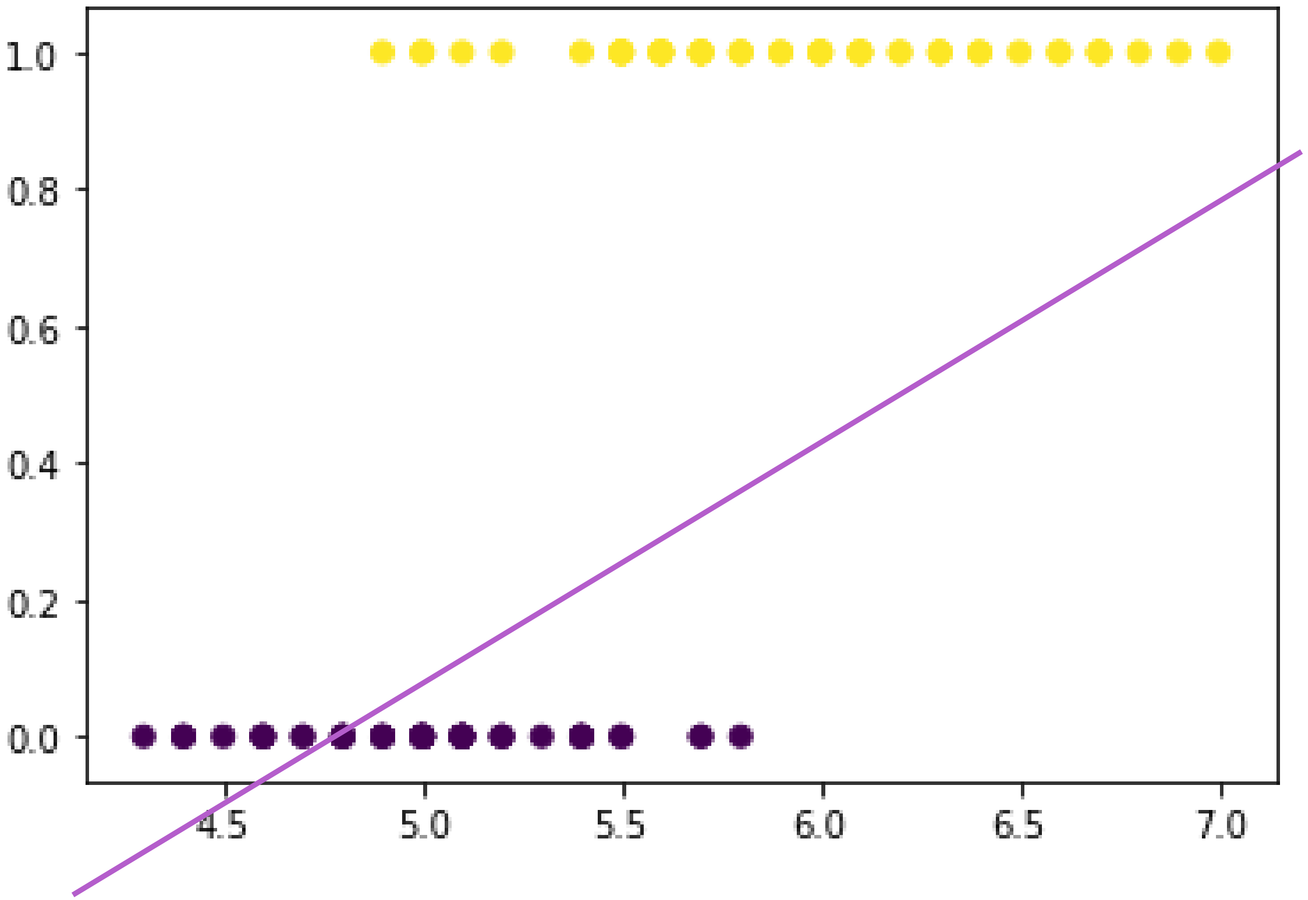
Linear regression cannot be used on qualitative response with more than two classes.

The regression models will not provide meaningful estimates of Pr (Y|X).

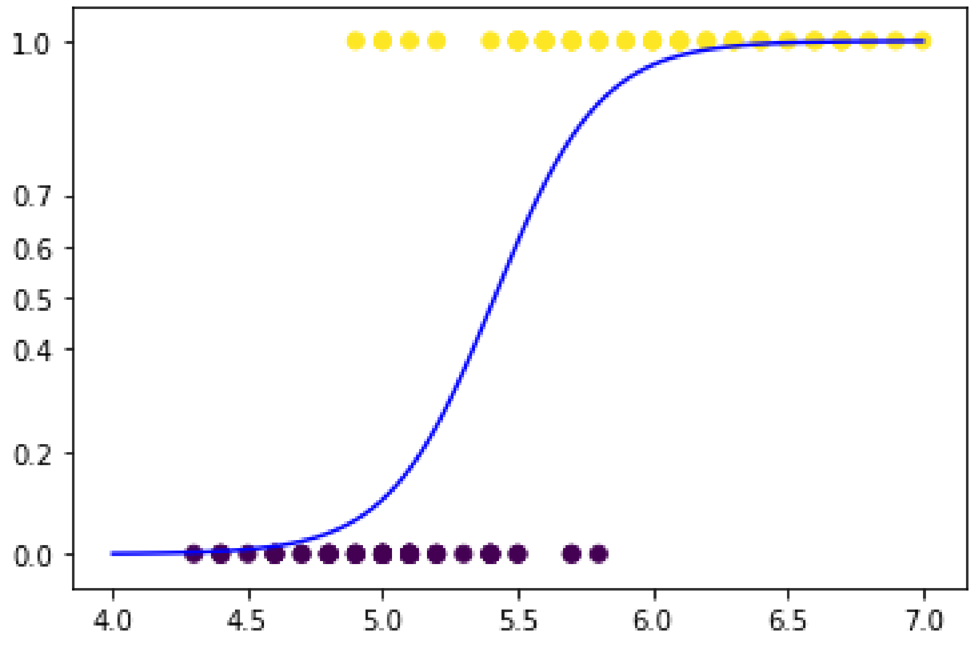
If $p(X)$ is defined as

$$p(X) = \beta_0 + \beta_1 X$$

it can obtain negative and values greater than 1.



Logistic Function



Instead of fitting a line to data, an S-shaped function, called logistic function is used to find the probabilities of different classes of outcomes, given the observed values of features.

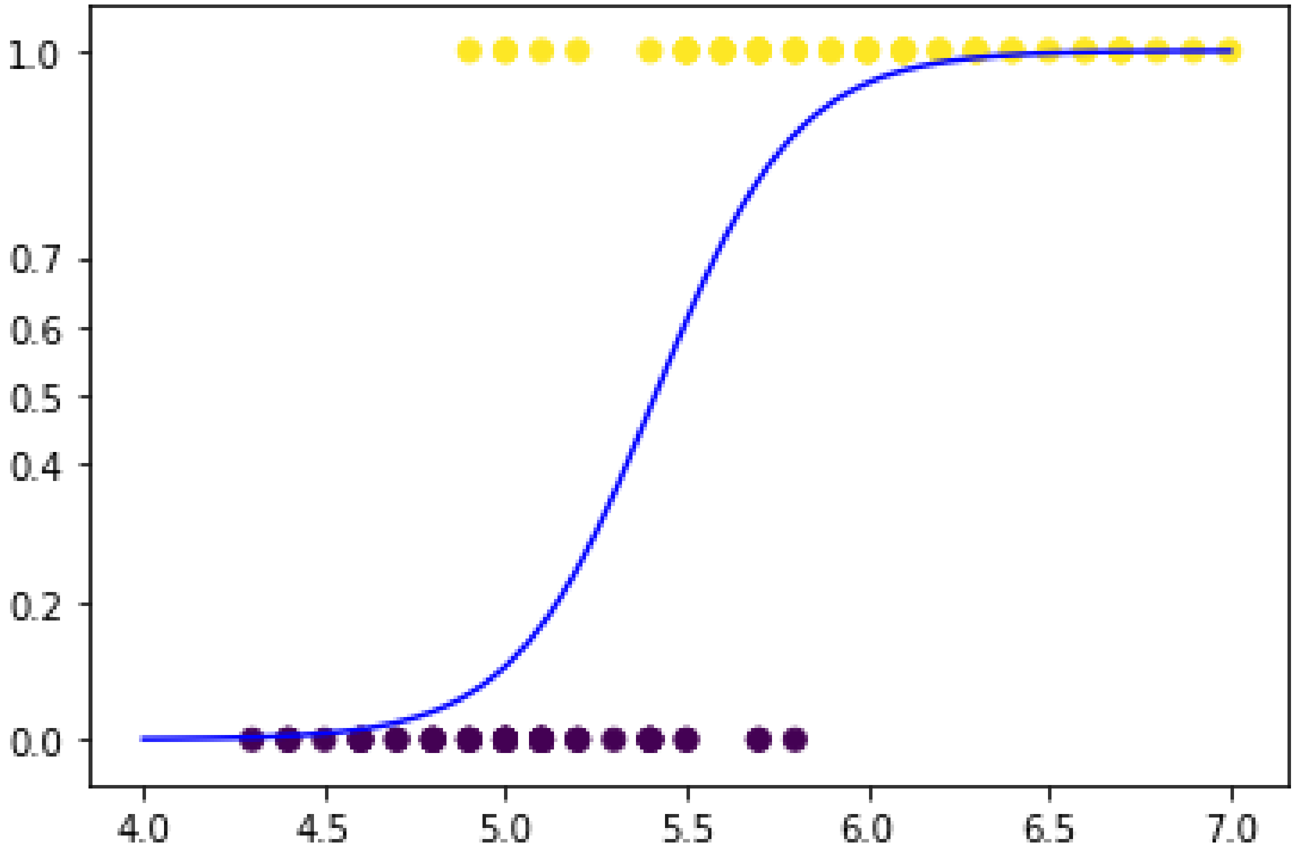
We can also have either simple logistic regression models: diabetes predicted by BMI.

Or more complicated models, obesity predicted by pregnancies, skin thickness, blood pressure and BMI.

Logistic Function

To avoid the problem with the range of $P(X)$, the logistic function is used which guarantees the output values between 0 and 1 for all values of X.

$$P(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$



Estimating the Coefficients

We learned that logistic function is:

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

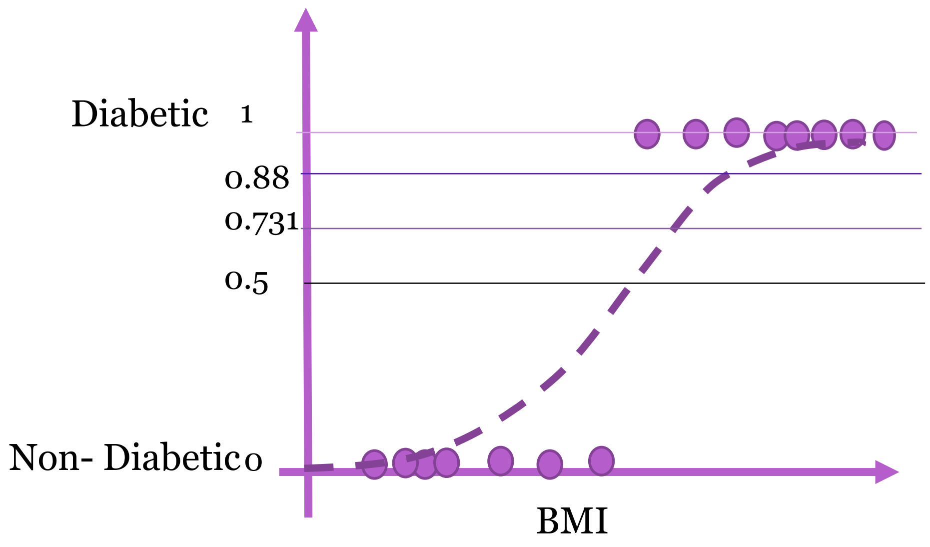
After some manipulation the odds function is defined as:

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

Therefore the log odds or logit function would be:

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

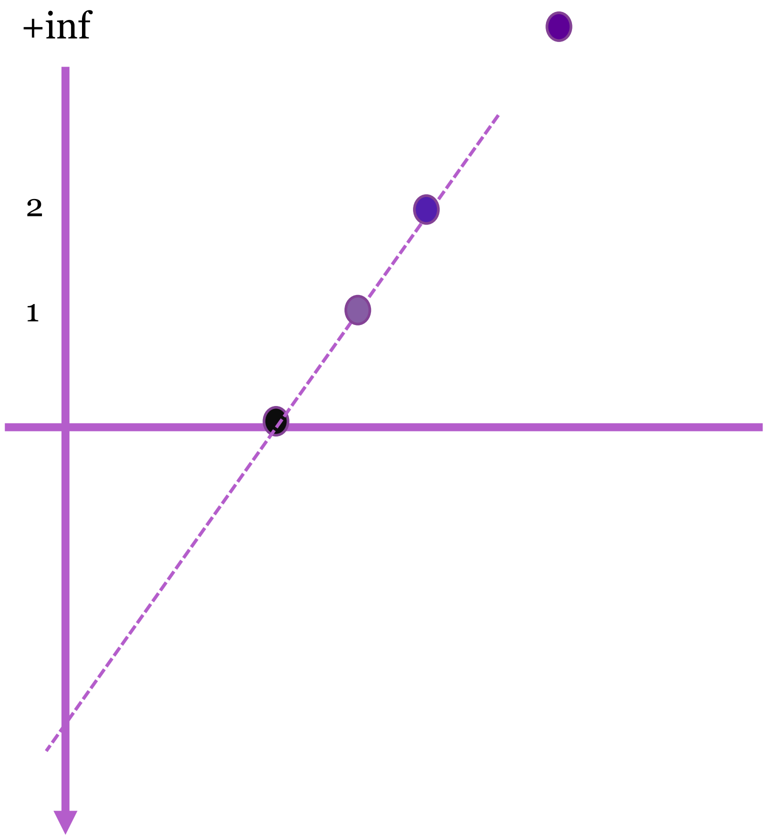
Transformation from Prob to Log (Odds)



$log\left(\frac{p(X)}{1-p(X)}\right)$

p: The probability of a person being diabetic

Odds: $\frac{p(X)}{1-p(X)}$



UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

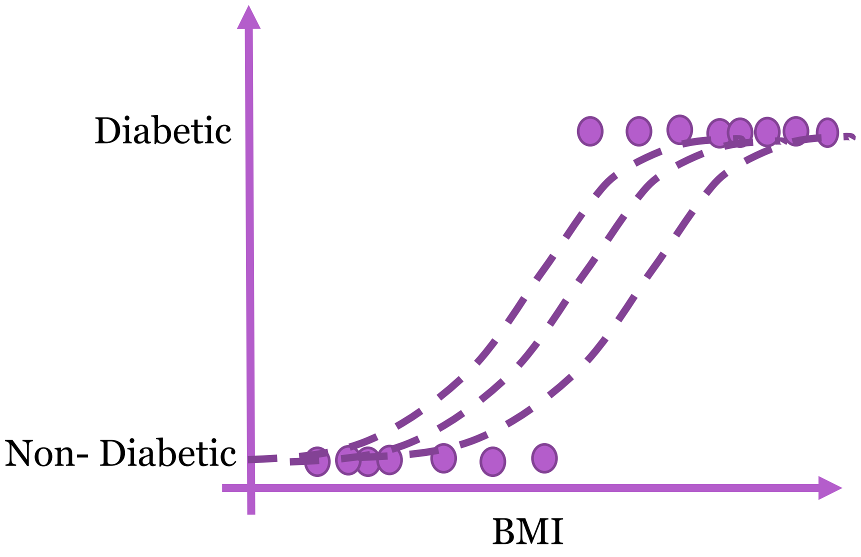
How to Fit Curves

In linear regression, we used RSS to minimize the distance of the points to the regression line.

Since logistic regression doesn't have the same concept as residual, we cannot use least square method to fit a line.

Instead, we use maximum likelihood.

Maximum Likelihood



Start by a curve, and calculate the likelihood of obesity for each data point, given the curve, and multiple all the likelihoods

Then shift the line and calculate the new likelihood

Repeat the process for other lines and choose the curve with the max likelihood as the best fit

Classification

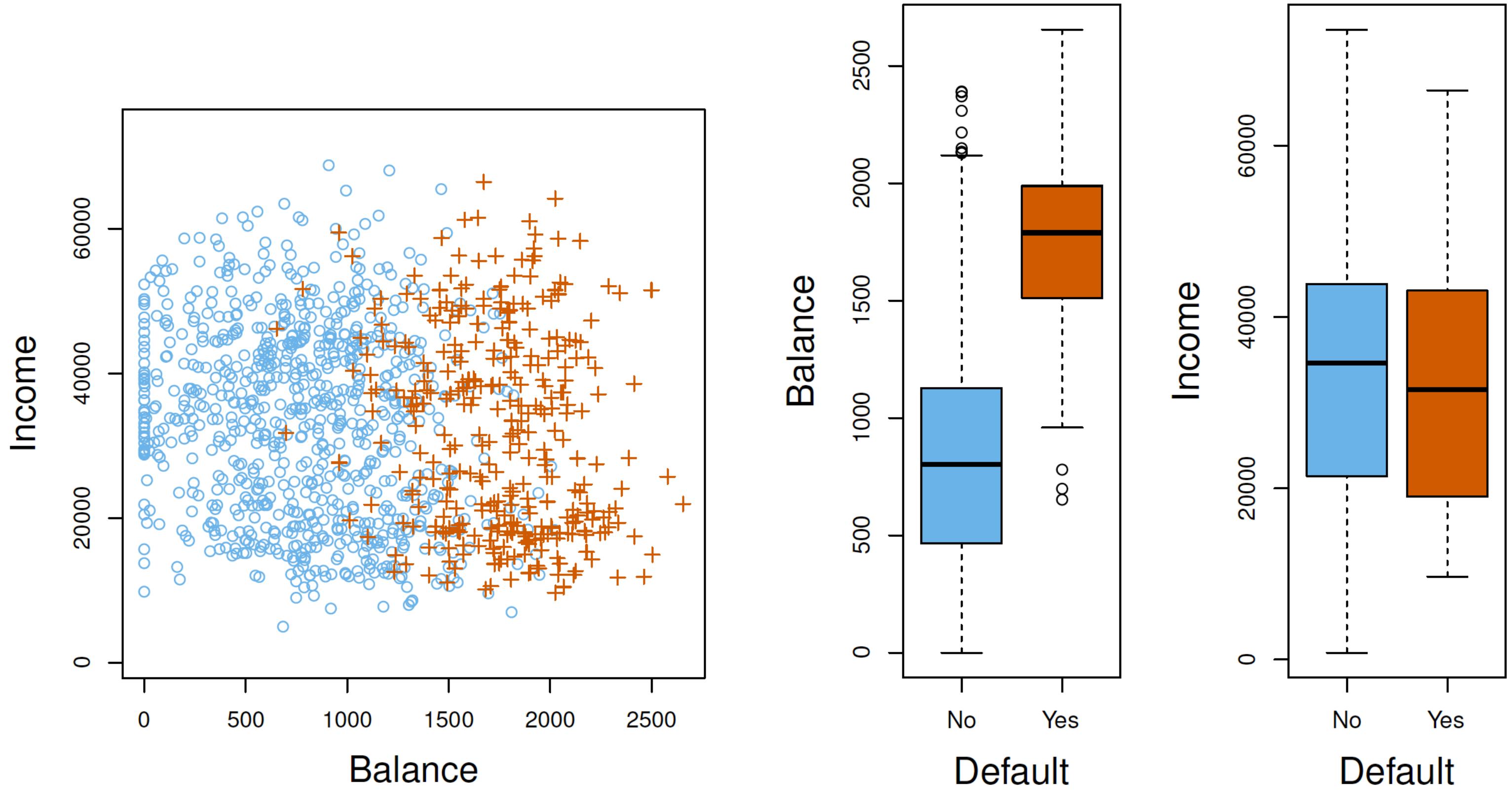
- Qualitative variables take values in an unordered set C, such as:

  eye color $\in$ {**brown**, **blue**, **green**}

  email $\in$ {**spam**, **ham**}.

- Given a feature vector X and a qualitative response Y taking values in the set C, the classification task is to build a function $C(X)$ that takes as input the feature vector X and predicts its value for Y; i.e. $C(X) \in C$.
- Often, we are more interested in estimating the *probabilities* that X belongs to each category in C.
- For example, it is more valuable to have an estimate of the probability that an insurance claim is fraudulent, than a classification like fraudulent or not.

Example: Credit Card Default



Can we use Linear Regression?

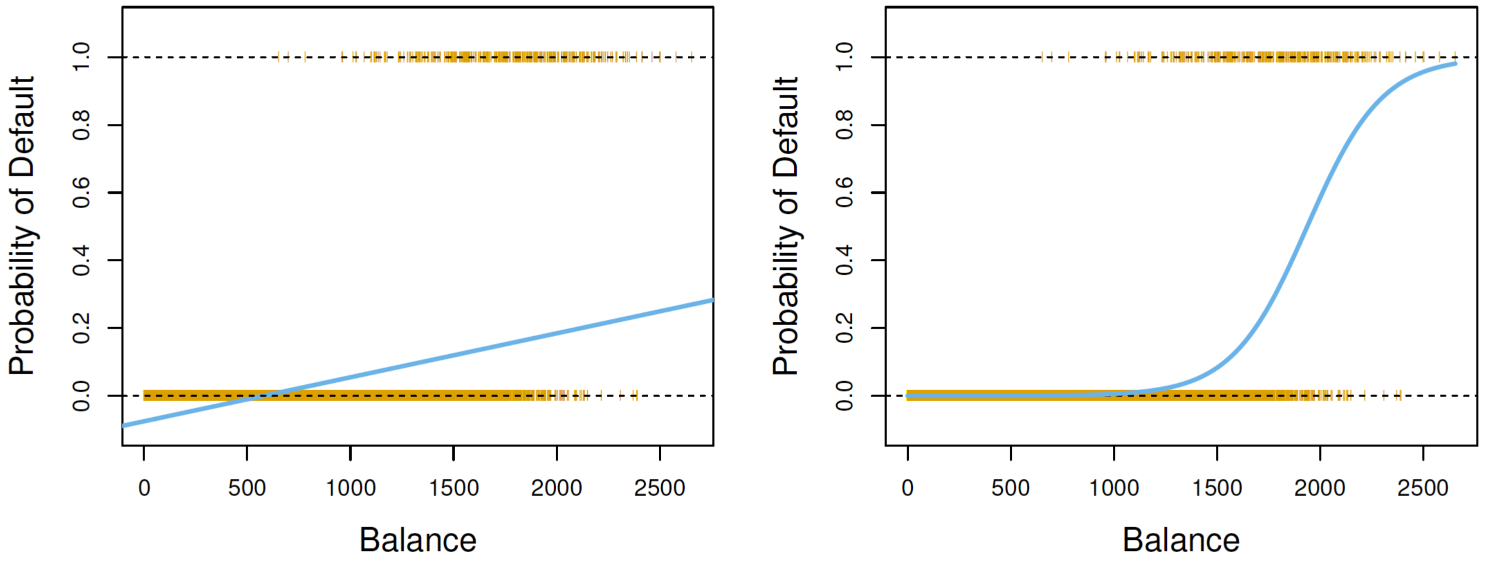
Suppose for the **Default** classification task that we code

$$Y = \begin{cases} 0 & \text{if } **\text{No}** \\ 1 & \text{if } **\text{Yes}**. \end{cases}$$

Can we simply perform a linear regression of Y on X and classify as **Yes** if $\hat{Y} > 0.5$?

- In this case of a binary outcome, linear regression does a good job as a classifier, and is equivalent to *linear discriminant analysis* which we discuss later.
- Since in the population $E(Y \mid X = x) = \Pr(Y = 1 \mid X = x)$, we might think that regression is perfect for this task.
- However, *linear* regression might produce probabilities less than zero or bigger than one. *Logistic regression* is more appropriate.

Linear versus Logistic Regression



The orange marks indicate the response Y, either 0 or 1. Linear regression does not estimate $\Pr(Y = 1 \mid X)$ well. Logistic regression seems well suited to the task.

Linear Regression continued

Now suppose we have a response variable with three possible values.
A patient presents at the emergency room, and we must classify them according to their symptoms.

$$Y = \begin{cases} 1 & \text{if } **\text{stroke}**; \\ 2 & \text{if } **\text{drug overdose}**; \\ 3 & \text{if } **\text{epileptic seizure}**. \end{cases}$$

This coding suggests an ordering, and in fact implies that the difference between **stroke** and **drug overdose** is the same as between **drug overdose** and **epileptic seizure**.

Linear regression is not appropriate here.
Multiclass Logistic Regression is more appropriate.

Logistic Regression

Let's write $p(X) = \Pr(Y = 1 \mid X)$ for short and consider using **balance** to predict **default**.
Logistic regression uses the form

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

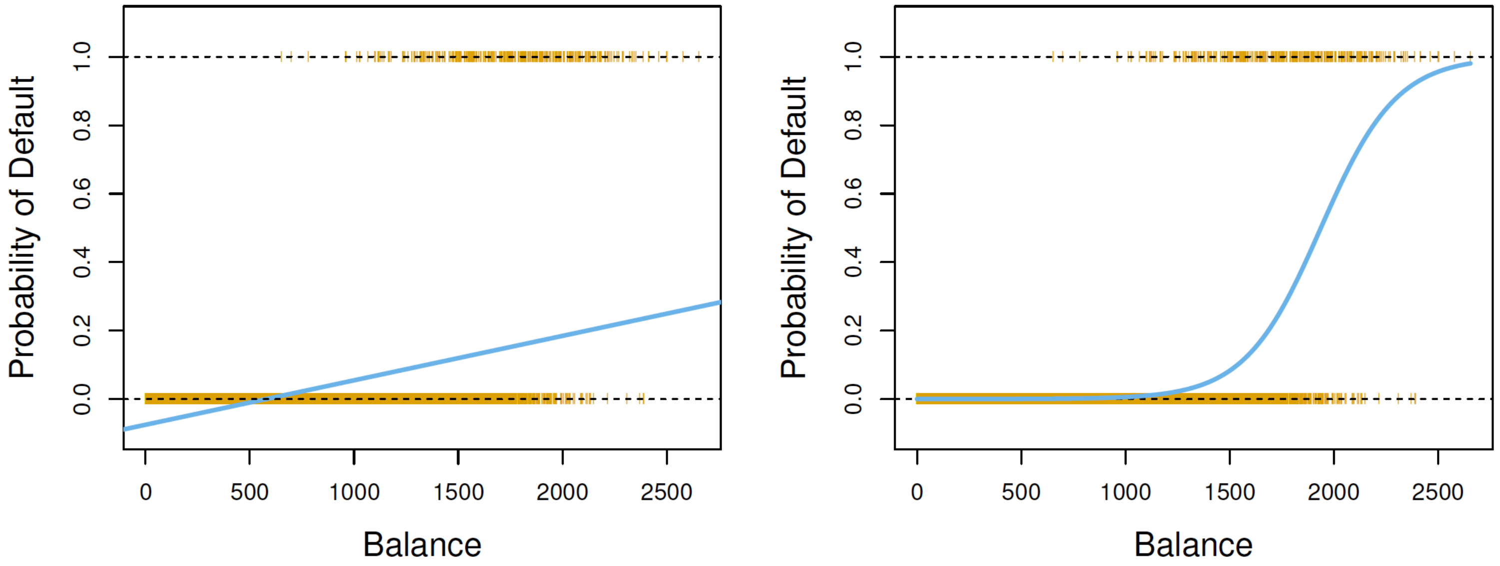
($e \approx 2.71828$ is a mathematical constant [Euler's number].)
It is easy to see that no matter what values β_0, β_1 or X take,
$p(X)$ will have values between 0 and 1.

A bit of rearrangement gives

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

This monotone transformation is called the *log odds* or *logit* transformation of $p(X)$.
(by log we mean *natural log*: ln.)

Linear versus Logistic Regression



Logistic regression ensures that our estimate for $p(X)$ lies between 0 and 1.

Maximum Likelihood

We use maximum likelihood to estimate the parameters.

$$\ell(\beta_0, \beta) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} \left(1 - p(x_i)\right).$$

This *likelihood* gives the probability of the observed zeros and ones in the data. We pick β_0 and β_1 to maximize the likelihood of the observed data.

Most statistical packages can fit linear logistic regression models by maximum likelihood. In **R** we use the glm function.

	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-10.6513	0.3612	-29.5	< 0.0001
`balance`	0.0055	0.0002	24.9	< 0.0001

Making Predictions

What is our estimated probability of **default** for someone with a balance of \$1000?

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1000}}{1+e^{-10.6513+0.0055\times 1000}} = 0.006$$

With a balance of \$2000?

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 2000}}{1+e^{-10.6513+0.0055\times 2000}} = 0.586$$

Let's do it again, using **student** as the predictor.

	Coefficient	Std. Error	Z-statistic	P-value
Intercept	-3.5041	0.0707	-49.55	< 0.0001
student[Yes]	0.4049	0.1150	3.52	0.0004

$$\widehat{Pr}(\text{default=Yes} \mid \text{student=Yes}) = \frac{e^{-3.5041+0.4049\times 1}}{1+e^{-3.5041+0.4049\times 1}} = 0.0431$$

$$\widehat{Pr}(\text{default=Yes} \mid \text{student=No}) = \frac{e^{-3.5041+0.4049\times 0}}{1+e^{-3.5041+0.4049\times 0}} = 0.0292$$

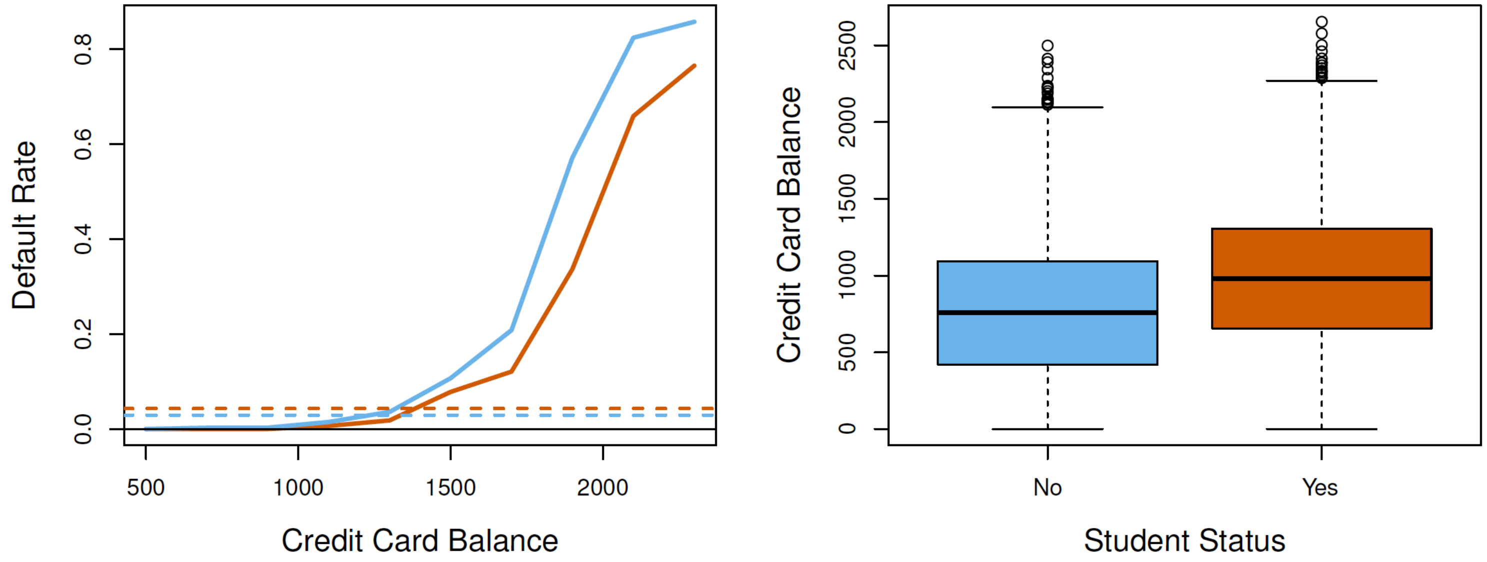
$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-10.8690	0.4923	-22.08	< 0.0001
`balance`	0.0057	0.0002	24.74	< 0.0001
`income`	0.0030	0.0082	0.37	0.7115
`student[Yes]`	-0.6468	0.2362	-2.74	0.0062

Why is the coefficient for `student` negative, while it was positive before?

Confounding



- Students tend to have higher balances than non-students, so their marginal default rate is higher than for non-students.
- But for each level of balance, students default less than non-students. Multiple logistic regression can tease this out.

Logistic Regression with Python: Steps

1. Explore data: import data, find features, evaluate their shape, statistics, etc.
2. Select features
3. Split data into test and training
4. Develop model and fit into data
5. Use model to predict
6. Evaluate the model using prediction matrices
7. Visualize accuracy with confusion matrix

1. Explore data

```
import pandas as pd

diabData = pd.read_csv("diabetes.csv")
```

```
diabData.head(3)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1

Data has been downloaded from https://www.kaggle.com/uciml/pima-indians-diabetes-database

2. Select features

```
outcome = diabData['Outcome']
data = diabData[diabData.columns[:8]]
```

3. Split data

```python
from sklearn.model_selection import train_test_split

train,test = train_test_split(diabData,test_size=0.25,random_state=0,stratify=diabData['Outcome'])

train_X = train[train.columns[:8]]

train_Y = train['Outcome']

test_X = test[test.columns[:8]]

test_Y = test['Outcome']
```

4. Develop and fit model

```
# import the class from sklearn
from sklearn.linear_model import LogisticRegression

# instantiate the model with default parameter
modelDiab = LogisticRegression

# instantiate the model with specific solver
modelDiab = LogisticRegression(solver='liblinear')

# fit the instantiated model with diabetes data
modelDiab.fit(train_X,train_Y)
```

Coefficients and intercept

```
print("The coefficients of the model for all features are:",modelDiab.coef_)
```

```
The coefficients of the model for all features are: [[ 1.07903087e-01  2.38686136e-02 -1.53769693e-02 -2.03047196e-03
   4.20953865e-04  5.10325832e-02  6.36314198e-01  1.14270286e-02]]
```

```
print("The intercept of the model is:",modelDiab.intercept_)
```

```
The intercept of the model is: [-5.25941126]
```

5. Predict using model

```
predictDiab = modelDiab.predict(test_X)
```

6. Evaluate the model

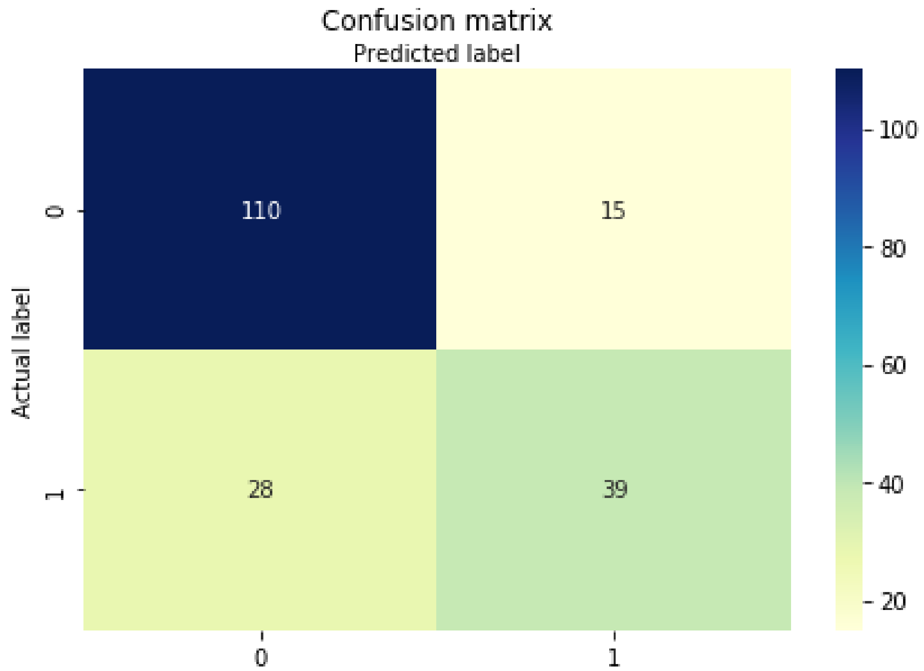
```
from sklearn import metrics

print("Accuracy:",metrics.accuracy_score(test_Y, predictDiab))

print("Precision:",metrics.precision_score(test_Y, predictDiab))

print("Recall:",metrics.recall_score(test_Y, predictDiab))
```

```
Accuracy: 0.7760416666666666
Precision: 0.7222222222222222
Recall: 0.582089552238806
```

7. Confusion matrix

```
conf_matrix = metrics.confusion_matrix(test_Y,predictDiab)
conf_matrix
```

```
array([[110,  15],
       [ 28,  39]], dtype=int64)
```



7. Visualizing confusion matrix

```
# import modules

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)

# create confusion matrix with heatmap
sns.heatmap(pd.DataFrame(conf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

MEASURING ACCURACY

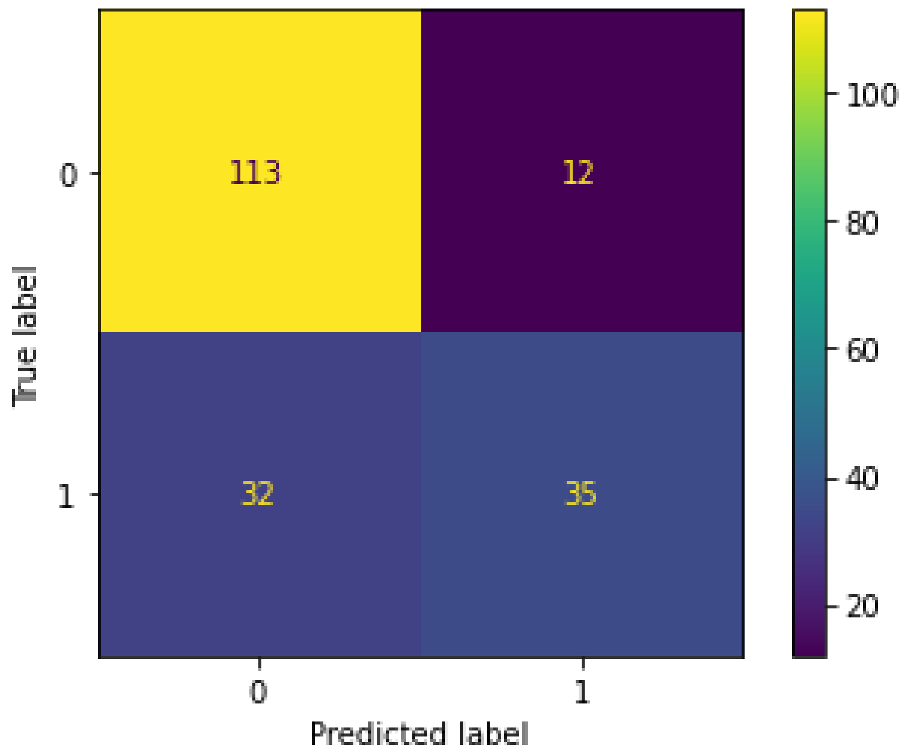
Measuring accuracy

In this section we will talk about tools to measure the accuracy of the classification algorithms:

- Confusion matrix
- Sensitivity & specificity
- Performance indices
- Receiver Operator Characteristic (ROC)
- Area under the Curve (AUC)

Confusion matrix

Confusion matrix represents predictions vs actual points in the data sets, e.g., predicted vs actual people with diabetes



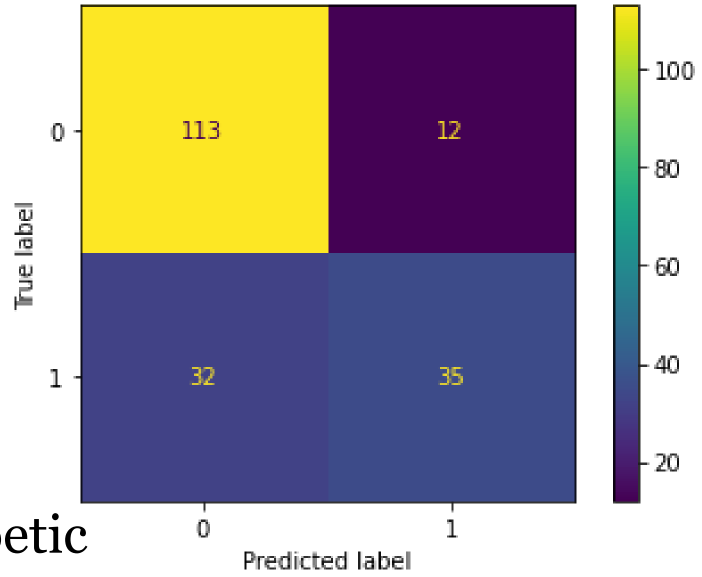
Sensitivity and Specificity

- Sensitivity: It describes what percentage of diabetic patients are correctly identified

$$\text{Sensitivity} = \text{True positive rate} = \frac{TP}{TP+FN} = \frac{35}{35+32} = 0.52$$

- Specificity: It shows what percentage of patients who are non-diabetic and are correctly identified

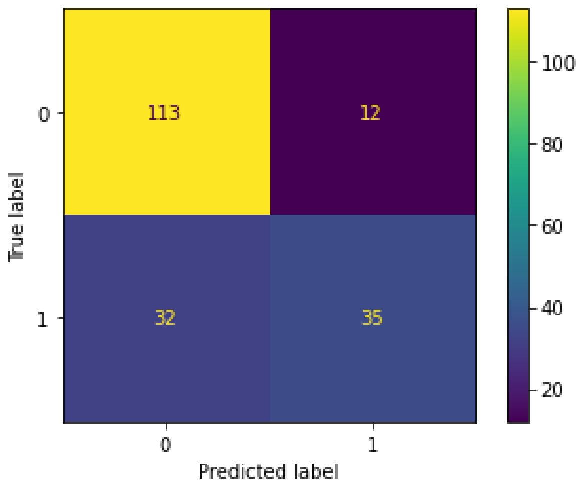
$$\text{Specificity} = \frac{TN}{TN+FP} = \frac{113}{113+12} = 0.90$$



Precision Score

- Precision indicates how many predictions made by model are actually positive out of all positive predictions.
- It is a useful index to evaluate the success of prediction, when classes are imbalanced
- It can be defined as: $\frac{TP}{FP+TP}$
- The precision score of logistic regression on diabetic data is

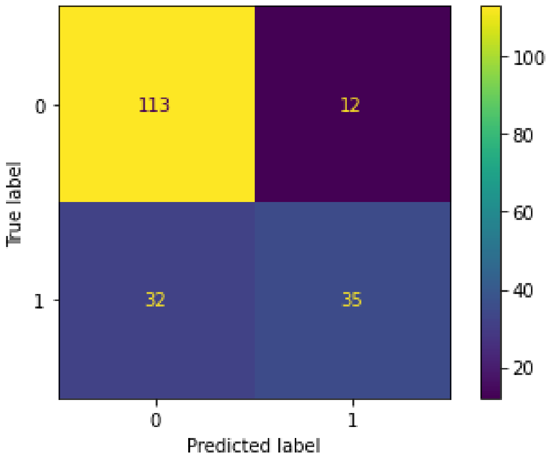
$$\frac{35}{12+35} = 0.74$$



- This score could be used a lot in medical applications as the doctors ideally want the models without any false positive, as it could create lots of stress for patients.

Recall Score

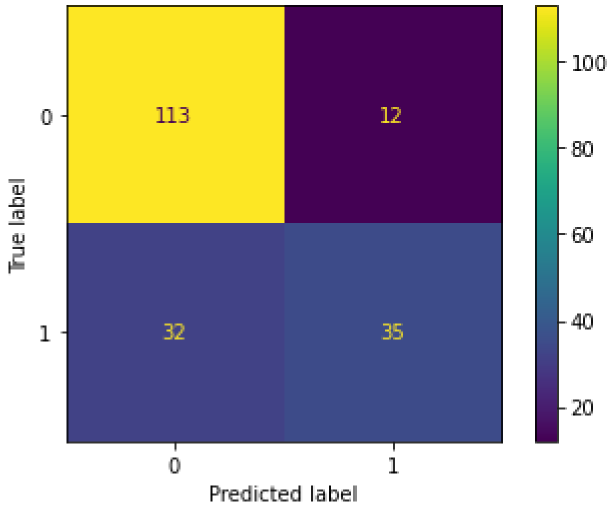
- Recall score indicates the model's ability to predict the positives out of true positives.
- It is different than precision score where the actual positives out of all positive predictions were measured.
- It is a useful index to evaluate the success of prediction, whe imbalanced
- It can be defined as: $\frac{TP}{FN+TP}$
- The recall score of logistic regression on diabetic data is



$$\frac{35}{32+35} = 0.52$$

Accuracy Score

- Accuracy score is an overall index to indicate the model's ability to predict the true positives and negatives to all observations.
- It tells us how we can rely on our ML model for correct classification.
- It is a useful index to evaluate the success of prediction, when classes are imbalanced
- It can be defined as: $\frac{TP+TN}{TP+FN+TN+FP}$
- The accuracy score of logistic regression on diabetic data i

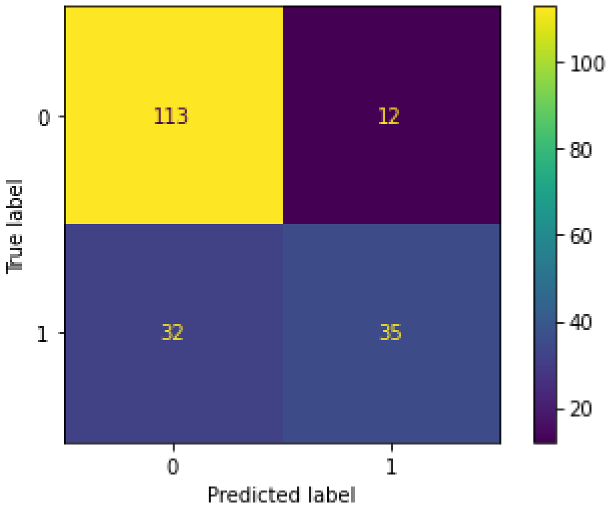


$$\frac{35+113}{35+32+113+12} = 0.81$$

F1-Score

- F1-Score provides a measurement of performance of the model by using both recall and precision scores.
- It can be considered as an alternative to the Accuracy Score, by not requiring to know the total number of observations.
- It is a useful index to evaluate the success of prediction, when classes are imbalanced
- It can be defined as: $\frac{2*precision*recall}{precision+recall}$
- The F1-Score of the logistic regression model on diabetic da

$$\frac{2*0.74*0.52}{0.74+0.52} = 0.61$$



OF
RING

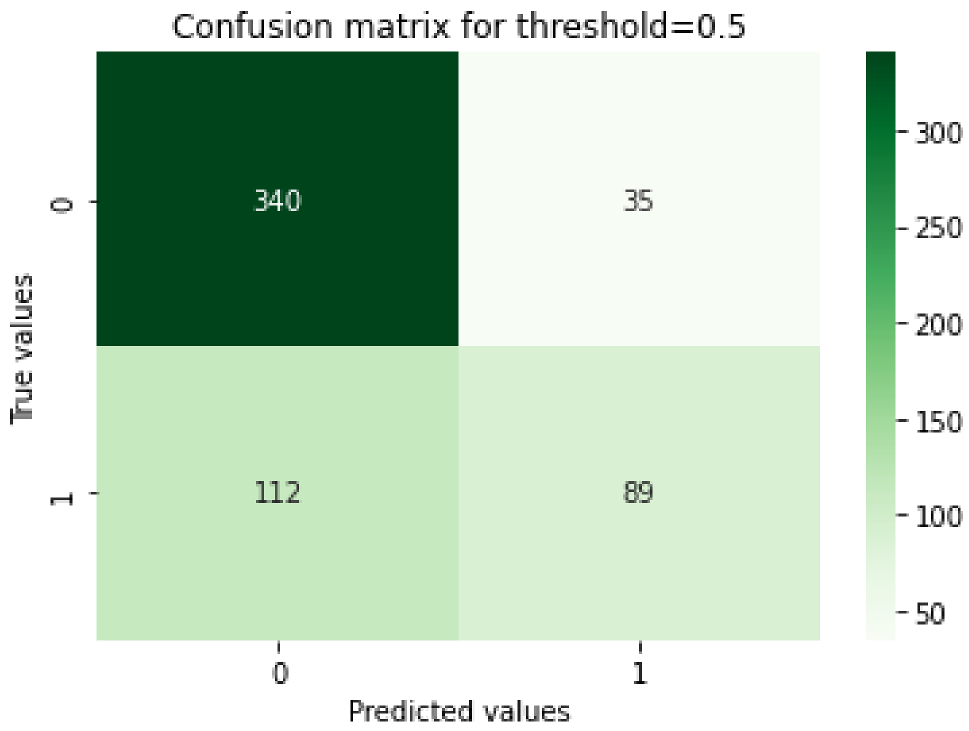
ROC / AUC

- We used the logistic regression curves to find the probability for classifying observation into different groups, e.g., diabatic or non-diabatic.

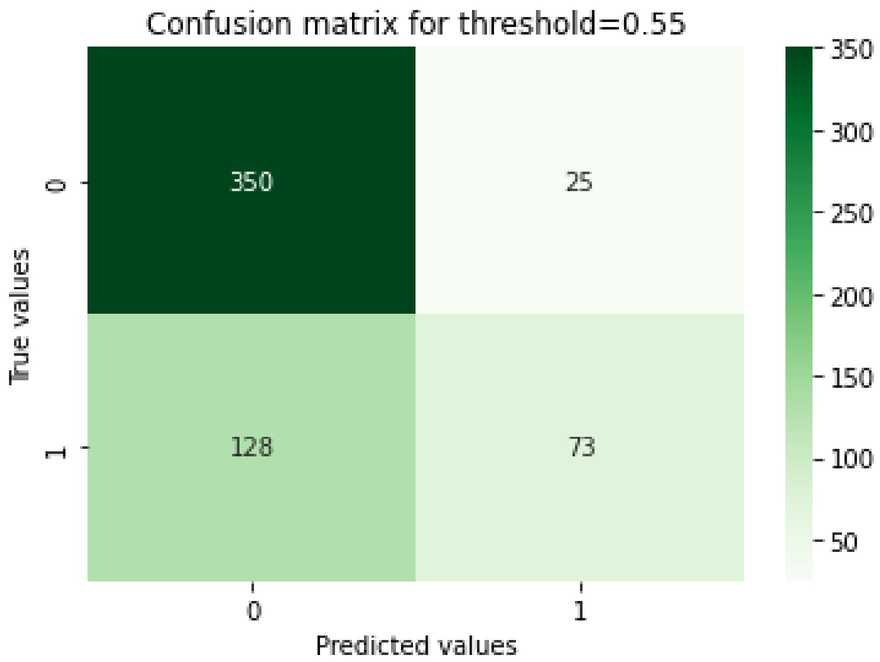
- In order to translate the probabilities to classification we used a threshold.

- Different threshold create different performance measurements.

ROC/AUC



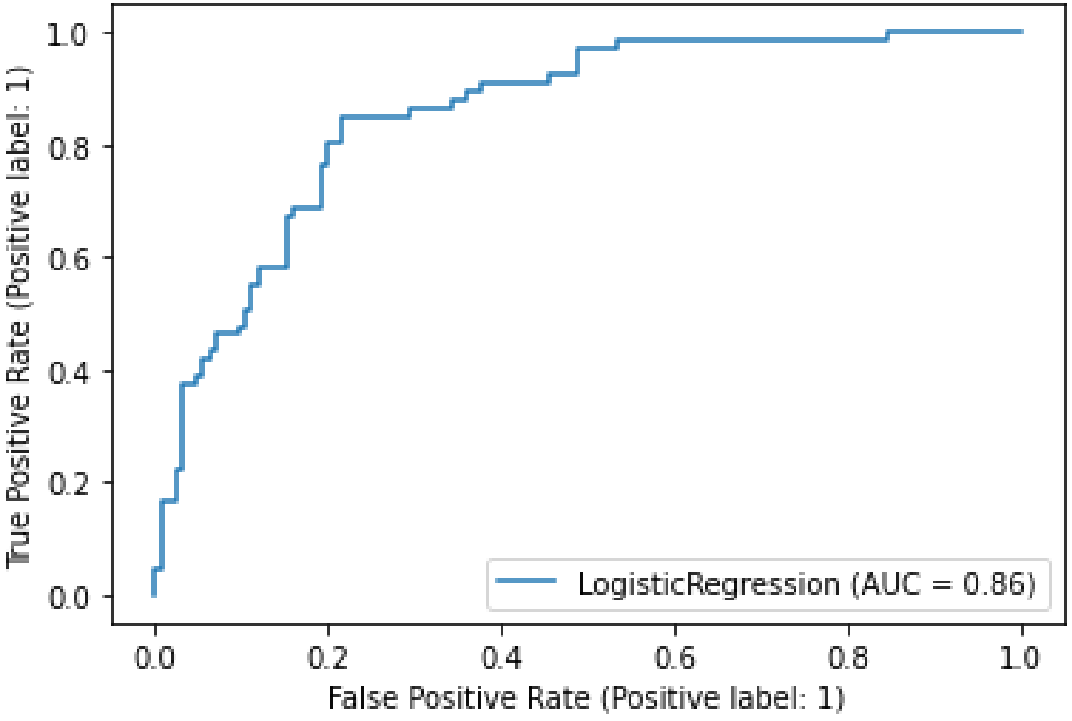
Confusion matrix for threshold=0.5
340
35
112
89
True values
Predicted values
0
1
300
250
200
150
100
50

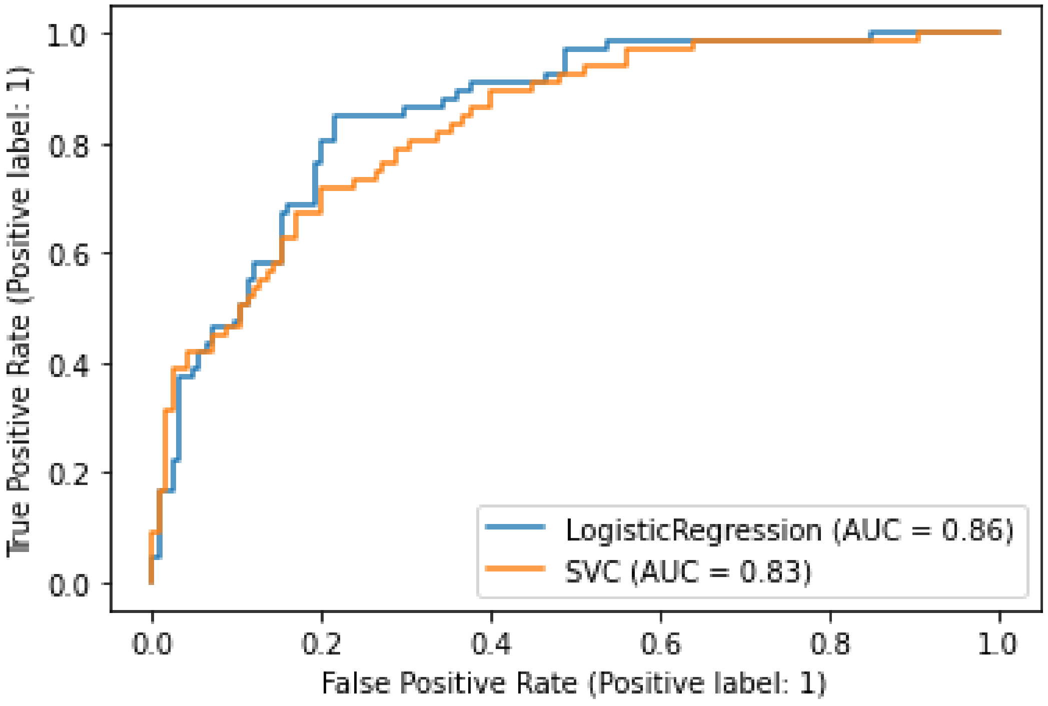


Confusion matrix for threshold=0.55
350
25
128
73
True values
Predicted values
0
1
350
300
250
200
150
100
50

ROC/AUC

In order to summarize all of the information with different settings in the classification algorithm, we use ROC curves.



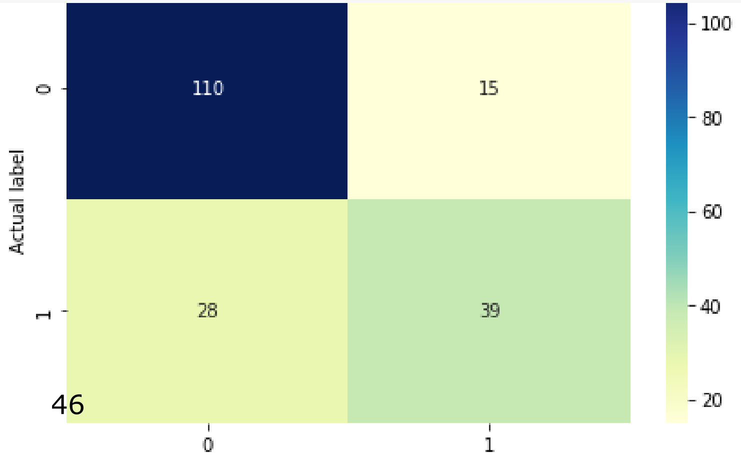


Python: Confusion Matrix

```
conf_matrix = metrics.confusion_matrix(test_Y,predictDiab)
conf_matrix
```

```
array([[110,  15],
       [ 28,  39]], dtype=int64)
```

```
metrics.plot_confusion_matrix(best_model,test_X1,test_Y1)
plt.show()
```



Python Performance Indices

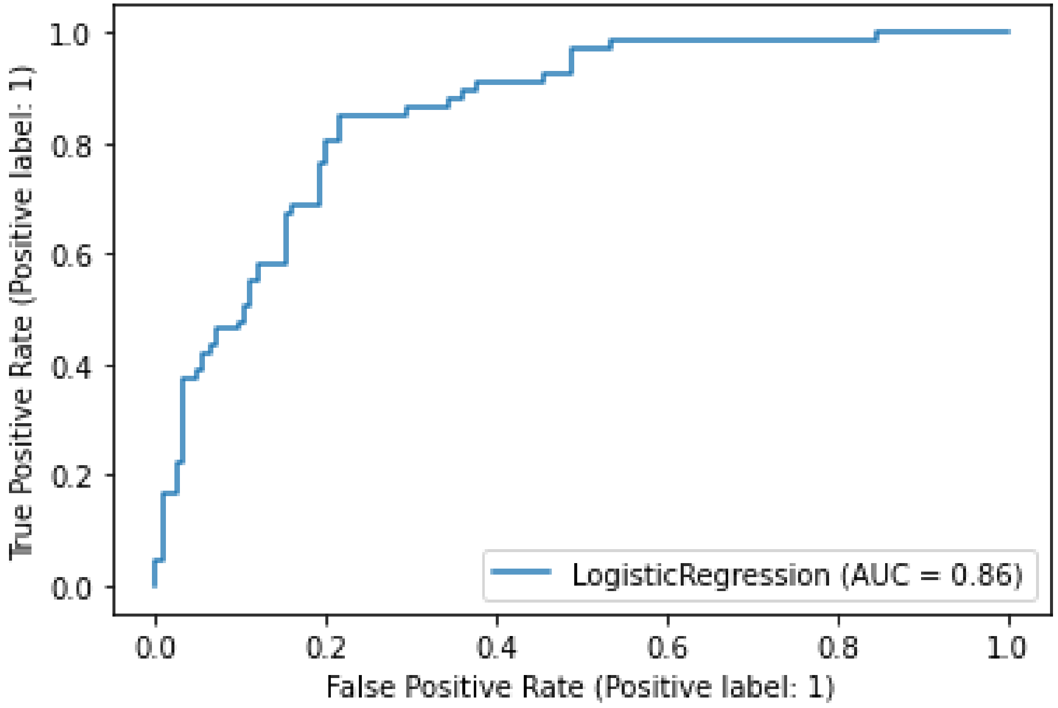
```
print('Model accuracy score is:',accuracy_score(test_Y,prediction))
print('Model precision score is:',precision_score(test_Y,prediction))
print('Model recall score is:',recall_score(test_Y,prediction))
print('Model F1 score is:',f1_score(test_Y,prediction))
```

```
Model accuracy score is: 0.7708333333333334
Model precision score is: 0.7169811320754716
Model recall score is: 0.5671641791044776
Model F1 score is: 0.6333333333333333
```

Python: ROC/AUC

```
[ ]  plot_roc_curve(model, test_X, test_Y)
```

```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7f4f9867a890>
```



UNIVERSITY OF
WATERLOO

FACULTY OF ENGINEERING