Computer Science

# CSC380: Principles of Data Science

## Basic machine learning 2

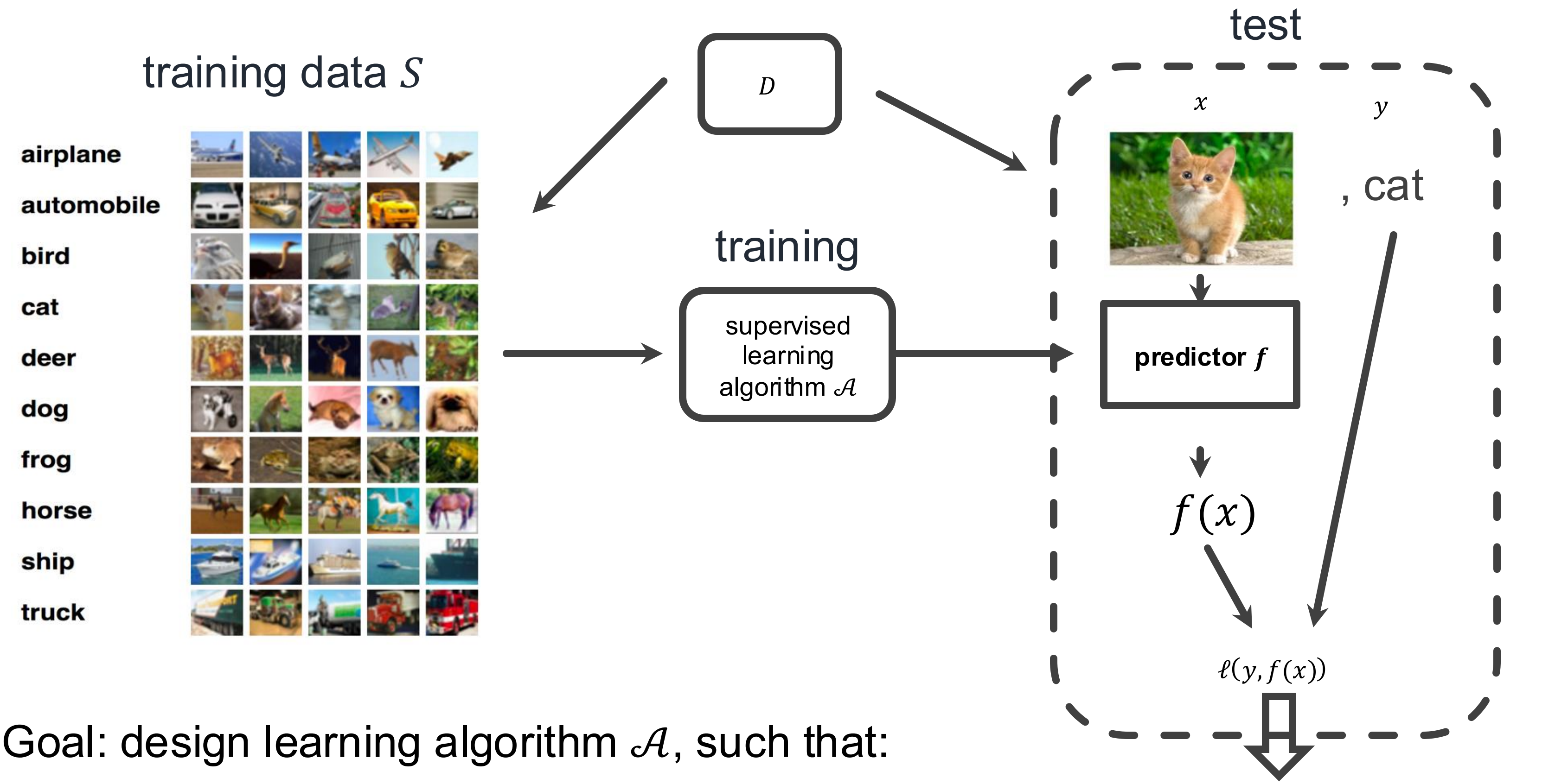
Xinchen Yu

# Outline

- Classification basics
- Nearest neighbor Classification
- Logistic regression
- Classification: other considerations
  - Binary classification beyond accuracy
  - Multiclass classification

# Classification recap

# Supervised learning setup in one figure



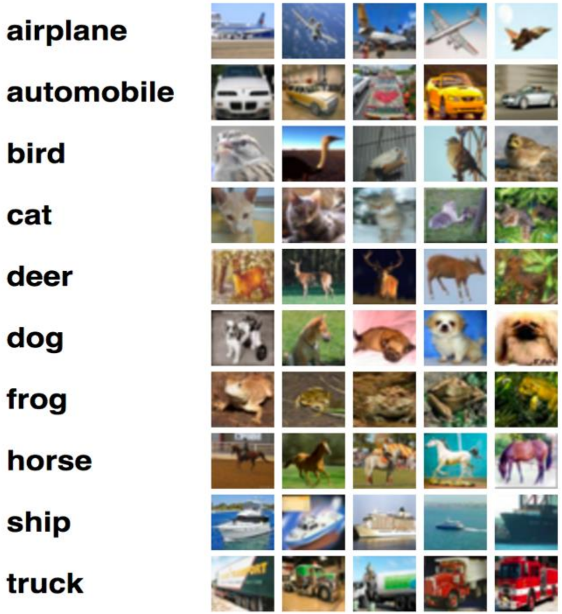
Test error: average of $\ell(y, f(x))$ in test set

- Goal: design learning algorithm $\mathcal{A}$, such that:
- after training, its output predictor $f$ has low test error

# Classification

- The labels are categorical
- Loss function $\ell$: measures the quality of prediction $\hat{y}$ respect to true label $y$
  - $\ell(y, \hat{y}) = I(y \neq \hat{y})$
  - $I$: indicator of predicate; 1 if true; 0 if false

- A classifier $f$'s error on a dataset S is the fraction of examples in S that it predicts incorrectly.
  - $f$'s training / test error is its error on training / test set
  - Accuracy = 1 – error



# In-class activity: finding test error

A company develops a simple **spam classifier** $f$ that predicts whether an email is **spam (1)** or **not spam (0)** based on the number of capital letters in the subject line.

$f$ outputs **Spam** if the number of capital letters ≥ 5, and **Not Spam** otherwise.

Suppose the test dataset is as follows. Find $f$'s test error.

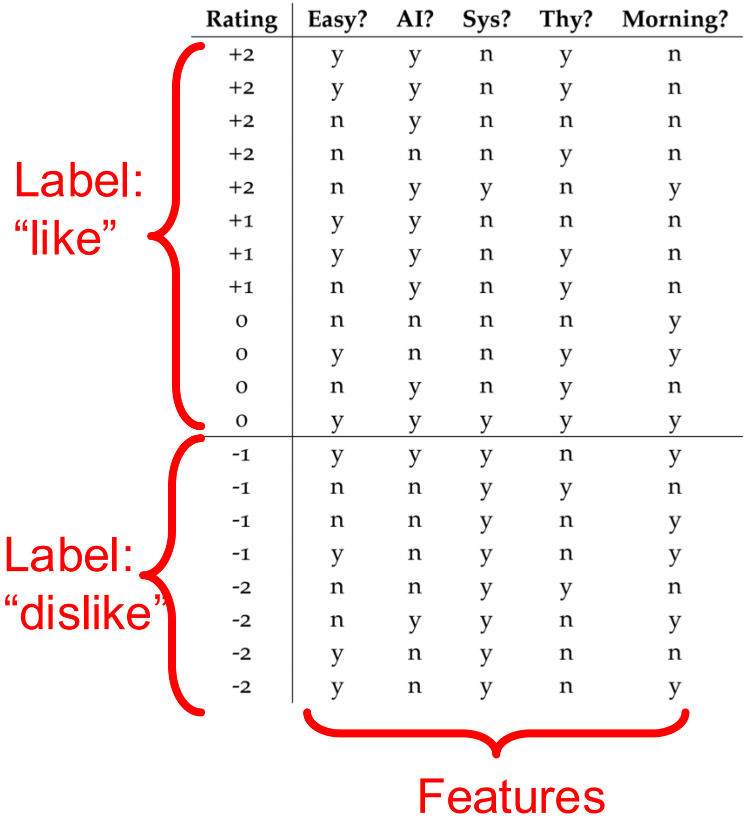
| Subject | True label |
|---|---|
| "WIN A FREE VACATION NOW!!!" | 1 |
| Meeting rescheduled to 3 PM | 0 |
| "HUGE DISCOUNT ON ALL ITEMS!!!" | 1 |
| URGENT: Please submit your report | 0 |
| Can you review this document? | 0 |

| Predicted label |
|---|
| 1 |
| 0 |
| 1 |
| 1 |
| 0 |

$f$'s test error = 1/5 = 20%

# Nearest Neighbor Classification

# Example: Course Recommendation



| Rating | Easy? | AI? | Sys? | Thy? | Morning? |
|---|---|---|---|---|---|
| +2 | y | y | n | y | n |
| +2 | y | y | n | y | n |
| +2 | n | y | n | n | n |
| +2 | n | n | n | y | n |
| +2 | n | y | y | n | y |
| +1 | y | y | n | n | n |
| +1 | y | y | n | y | n |
| +1 | n | y | n | y | n |
| 0 | n | n | n | n | y |
| 0 | y | n | n | y | y |
| 0 | n | y | n | y | n |
| 0 | y | y | y | y | y |
| -1 | y | y | y | n | y |
| -1 | n | n | y | y | n |
| -1 | n | n | y | n | y |
| -1 | y | n | y | n | y |
| -2 | n | n | y | y | n |
| -2 | n | y | y | n | y |
| -2 | y | n | y | n | n |
| -2 | y | n | y | n | y |

Label: “like”

Label: “dislike”

Features

Suppose we’d like to build a recommendation system for classes

We’ve collected information about many past classes

We can frame this as a classification problem:

Predict like/dislike from class features

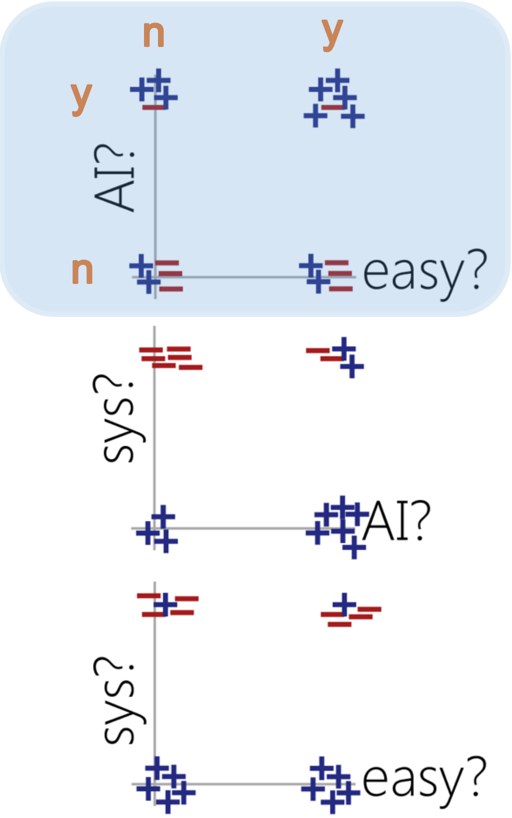
# Example: Course Recommendation

| Rating | Easy? | AI? | Sys? | Thy? | Morning? |
|---|---|---|---|---|---|
| +2 | y | y | n | y | n |
| +2 | y | y | n | y | n |
| +2 | n | y | n | n | n |
| +2 | n | n | n | y | n |
| +2 | n | y | y | n | y |
| +1 | y | y | n | n | n |
| +1 | y | y | n | y | n |
| +1 | n | y | n | y | n |
| 0 | n | n | n | n | y |
| 0 | y | n | n | y | y |
| 0 | n | y | n | y | n |
| 0 | y | y | y | y | y |
| -1 | y | y | y | n | y |
| -1 | n | n | y | y | n |
| -1 | n | n | y | n | y |
| -1 | y | n | y | n | y |
| -2 | n | n | y | y | n |
| -2 | n | y | y | n | y |
| -2 | y | n | y | n | n |
| -2 | y | n | y | n | y |

Label: +

Label: -

Features



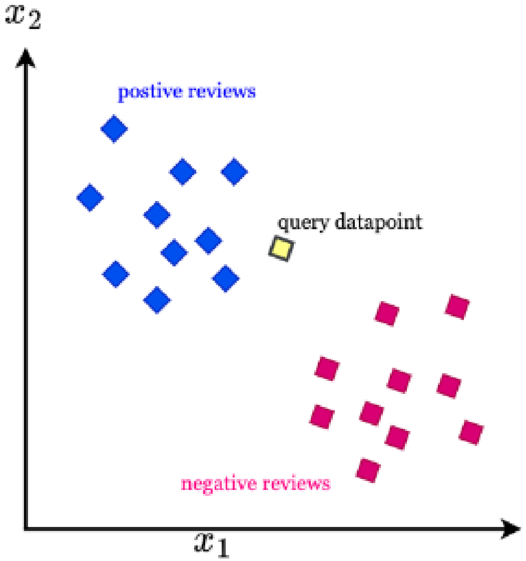
Each course's feature is Represented as points in 5-dimensional space

That's too many dimensions to plot…so we look at 2D projections…

Observation: examples with same labels tend to be closer!

# Nearest neighbor classification

- Given a new course, would like to predict its label (+/-)
- Idea: Find its most similar course in the training set, and use that course's label to predict

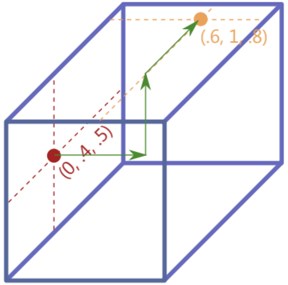


# Measuring nearest neighbors

- Oftentimes convenient to work with feature $x \in \mathrm{R}^d$
- Distances in $\mathrm{R}^d$:

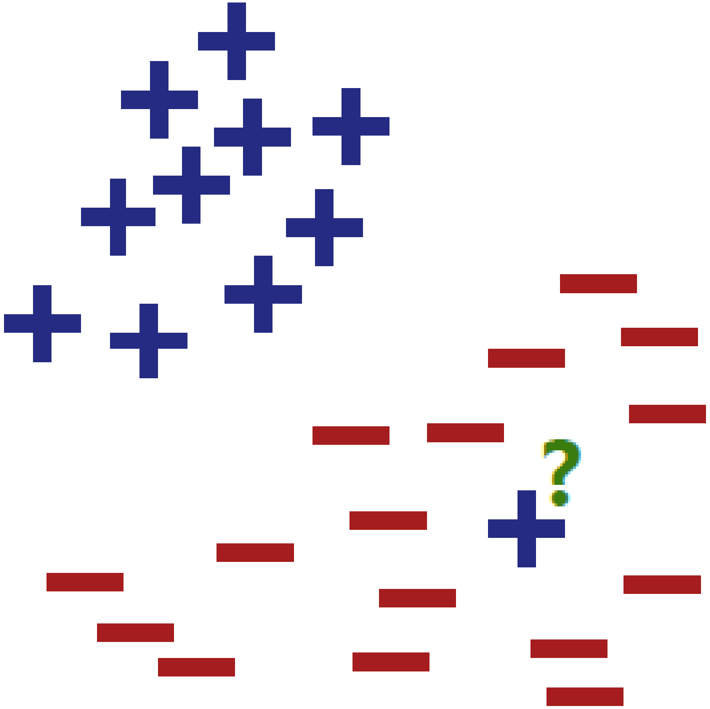
  notation $x(f)$: $x = (x(1), \dots, x(d))$

  - (popular) Euclidean distance $d_2(x, x') = \sqrt{\sum_{f=1}^{d} \big(x(f) - x'(f)\big)^2}$
  - Manhattan distance $d_1(x, x') = \sum_{f=1}^{d} |x(f) - x'(f)|$



- How to extract features as **<u>real values</u>**?
  - Boolean features: {Y, N} -> {0,1}
  - Categorical features: {Red, Blue, Green, Black}
    - Convert to {1, 2, 3, 4}?
    - Better one-hot encoding: (1,0,0,0), .., (0,0,0,1) (IsRed?/isGreen?/isBlue?/IsBlack?)

# Robustify Nearest Neighbor Classification



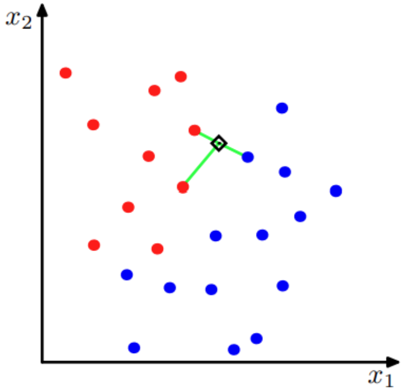
Q: Can we predict using 1 nearest neighbor's?

Query point ? Will be classified as + but should be -

**Problem:** predicting using 1 nearest neighbor's label can be sensitive to noisy data

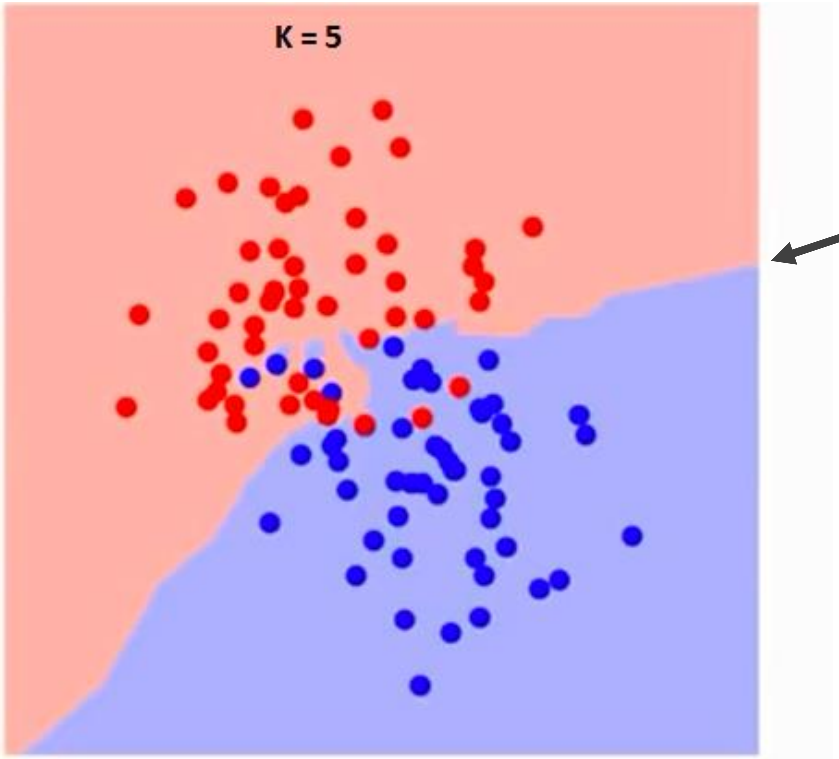
How to mitigate this?

# $k$-nearest neighbors ($k$-NN): main concept



- Training set: $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
- **Key insight**: given test example $x$, its label should resemble the labels of *nearby points*
- Function
  - input: $x$
  - find the $k$ nearest points to $x$ from $S$; call their indices $N(x)$
  - output:
    - (classification) the majority vote of $\{y_i : i \in N(x)\}$
    - (regression) the average of $\{y_i : i \in N(x)\}$

# k-NN classification example



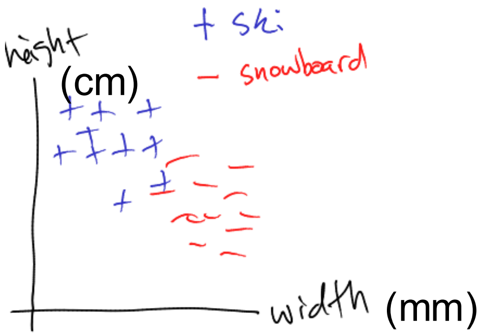
decision boundary

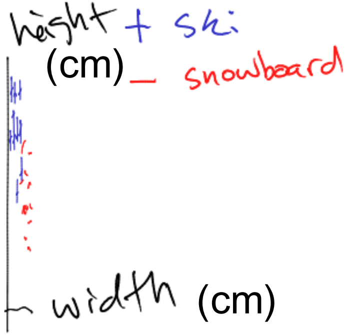
# Issue 1: scaling

- Features having different scales can be problematic.

- Ex: ski vs. snowboard classification

$$d = \sqrt{(height_1 - height_2)^2 + (weight_1 - weight_2)^2}$$





- One solution: feature standardization

# Make sure features are scaled fairly

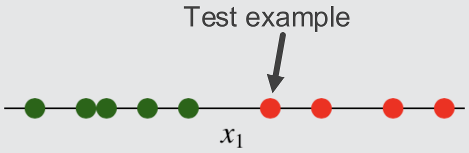
- Features having different scale can be problematic
- [Definition] **Standardization**
  - For each feature f, compute $\mu_f = \frac{1}{m}\sum_{i=1}^{m} x_f^{(i)}$ , $\sigma_f = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(x_f^{(i)} - \mu_f\right)^2}$
  - Then, transform the data by $\forall f \in \{1,\ldots,d\}, \forall i \in \{1,\ldots,m\},\ x_f^{(i)} \leftarrow \frac{x_f^{(i)} - \mu_f}{\sigma_f}$

after transformation, each feature has mean 0 and variance 1

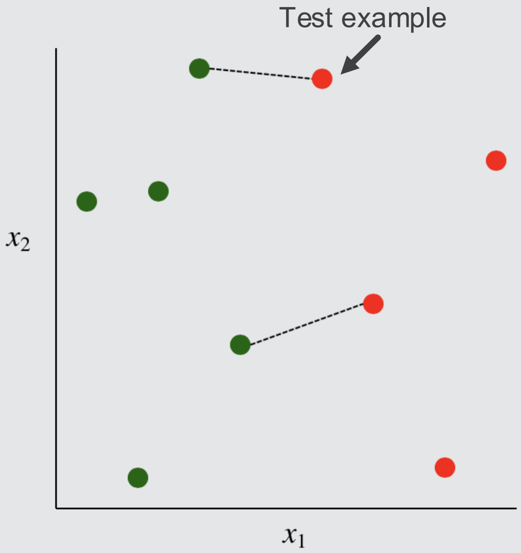
- Be sure to keep the "standardize" function and apply it to the test points.
  - Save $\{(\mu_f, \sigma_f)\}_{f=1}^{d}$
  - For test point $x^*$, apply $x_f^* \leftarrow \frac{x_f^* - \mu_f}{\sigma_f}, \forall f$

# Issue 2: irrelevant features

here’s a case in which there is one relevant feature $x_1$ and a 1-NN rule classifies each instance correctly



consider the effect of an irrelevant feature $x_2$ on distances and nearest neighbors

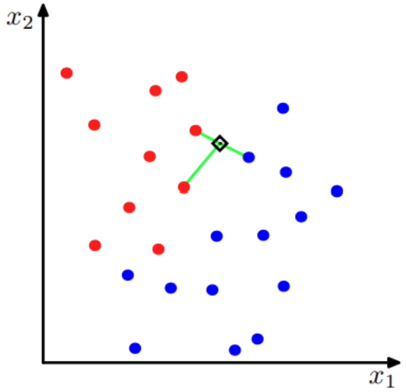


- Mitigation: feature selection

# Issue 3: choosing k

- Q: How would a $k$-NN classifier predict when $k$=training set size?
  - Predict majority label everywhere
  - Underfitting

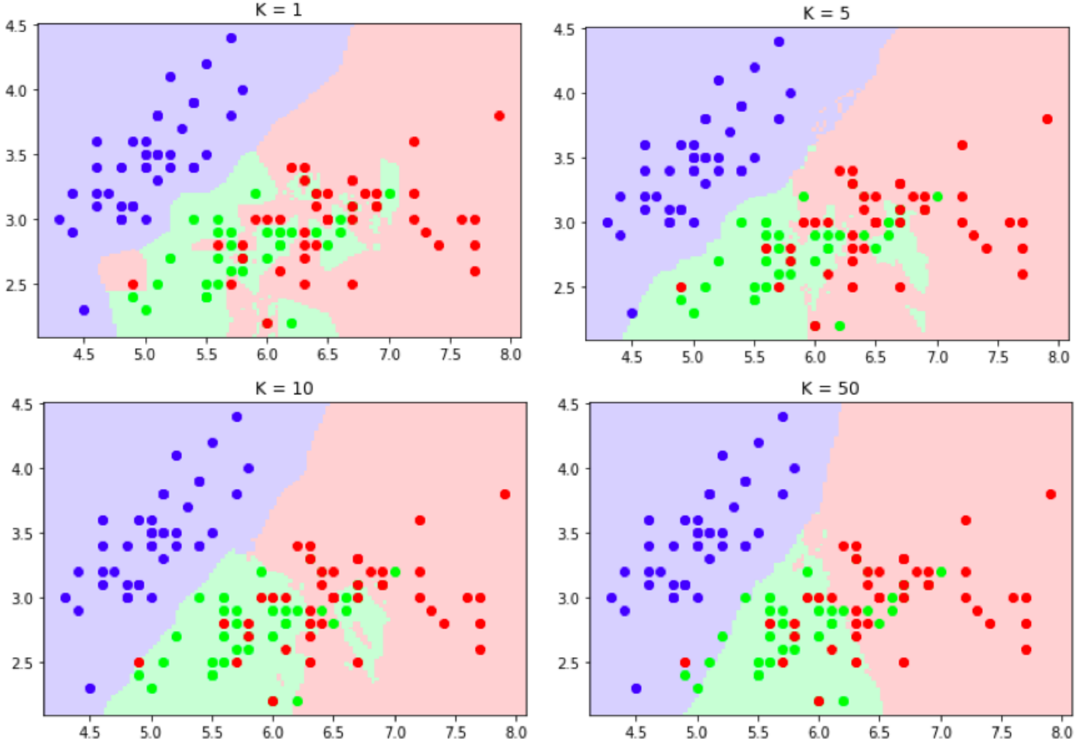
- Q: What is the training error of a 1-NN classifier?
  - 0
  - Overfitting



# Issue 3: choosing k

$k$ can be viewed as a model complexity measure

Smaller $k$ results in a more complex model

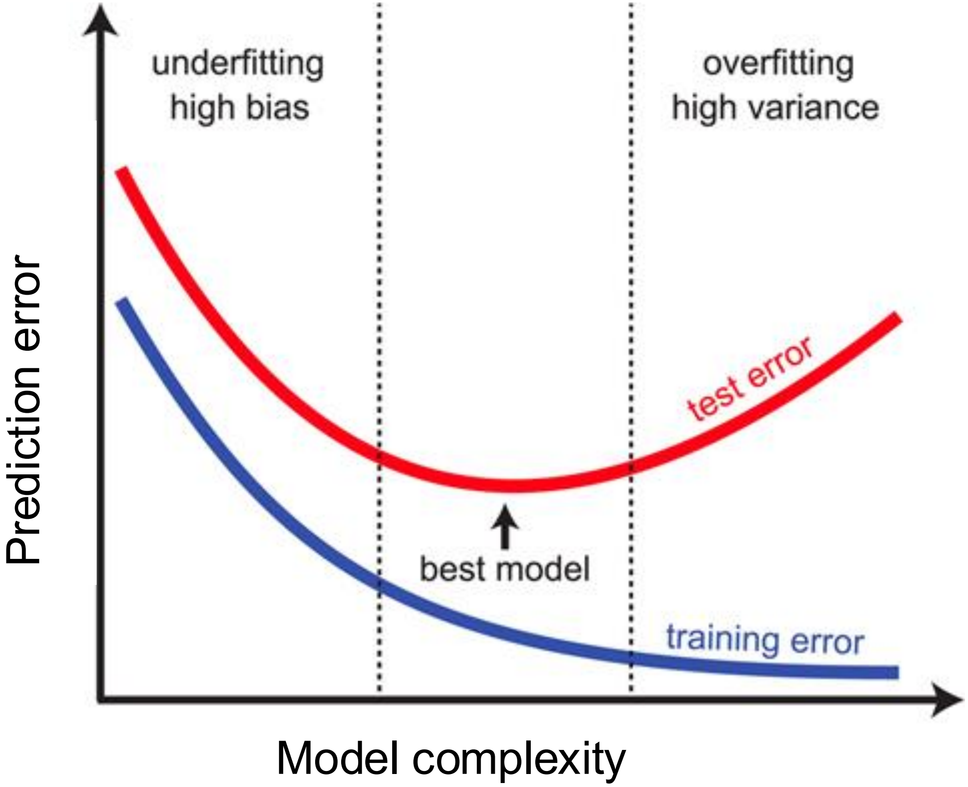


# Issue 3: choosing k

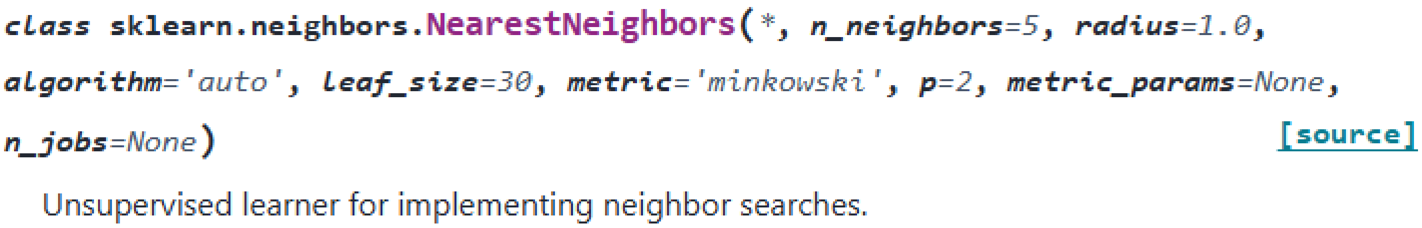
We'd like to choose appropriate $k$ to balance model bias and complexity

We can choose $k$ in the same way we chose $\lambda$ in ridge regression

- Cross validation



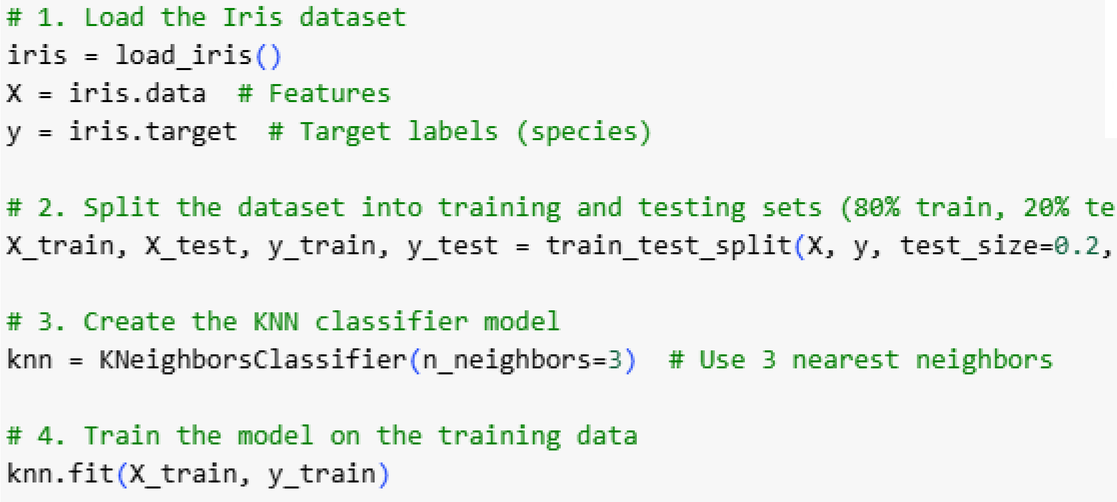
# Scikit-learn nearest neighbors



```
class sklearn.neighbors.NearestNeighbors(*, n_neighbors=5, radius=1.0,
algorithm='auto', leaf_size=30, metric='minkowski', p=2, metric_params=None,
n_jobs=None)
```

[source]

Unsupervised learner for implementing neighbor searches.



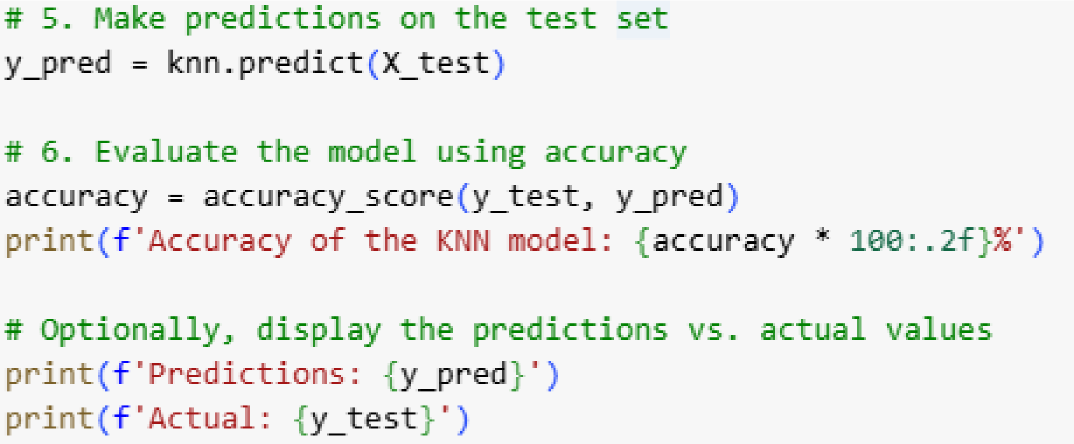
```
# 1. Load the Iris dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target  # Target labels (species)

# 2. Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 3. Create the KNN classifier model
knn = KNeighborsClassifier(n_neighbors=3)  # Use 3 nearest neighbors

# 4. Train the model on the training data
knn.fit(X_train, y_train)
```

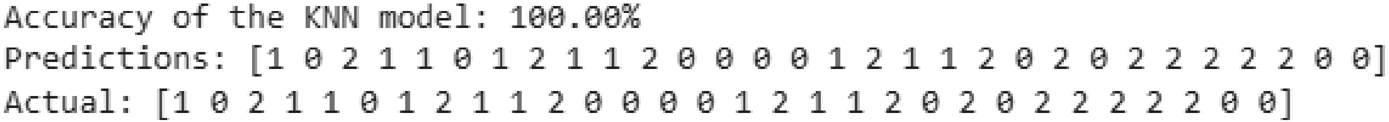
# Scikit-learn nearest neighbors



```python
# 5. Make predictions on the test set
y_pred = knn.predict(X_test)

# 6. Evaluate the model using accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy of the KNN model: {accuracy * 100:.2f}%')

# Optionally, display the predictions vs. actual values
print(f'Predictions: {y_pred}')
print(f'Actual: {y_test}')
```



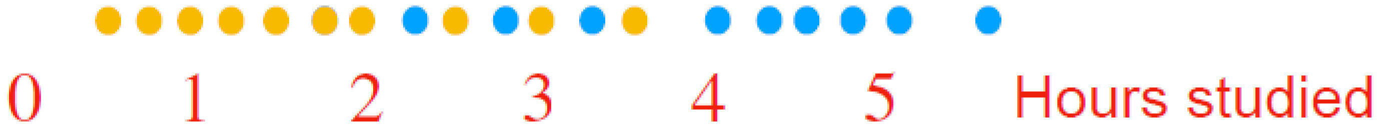
```
Accuracy of the KNN model: 100.00%
Predictions: [1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
Actual: [1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
```

# Logistic regression

# Classification with logistic regression

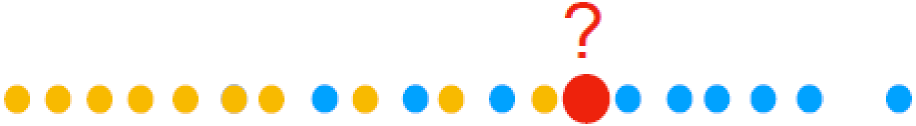
Training data: number of hours studied for the course.

Labels: Pass (1) or Fail (0)



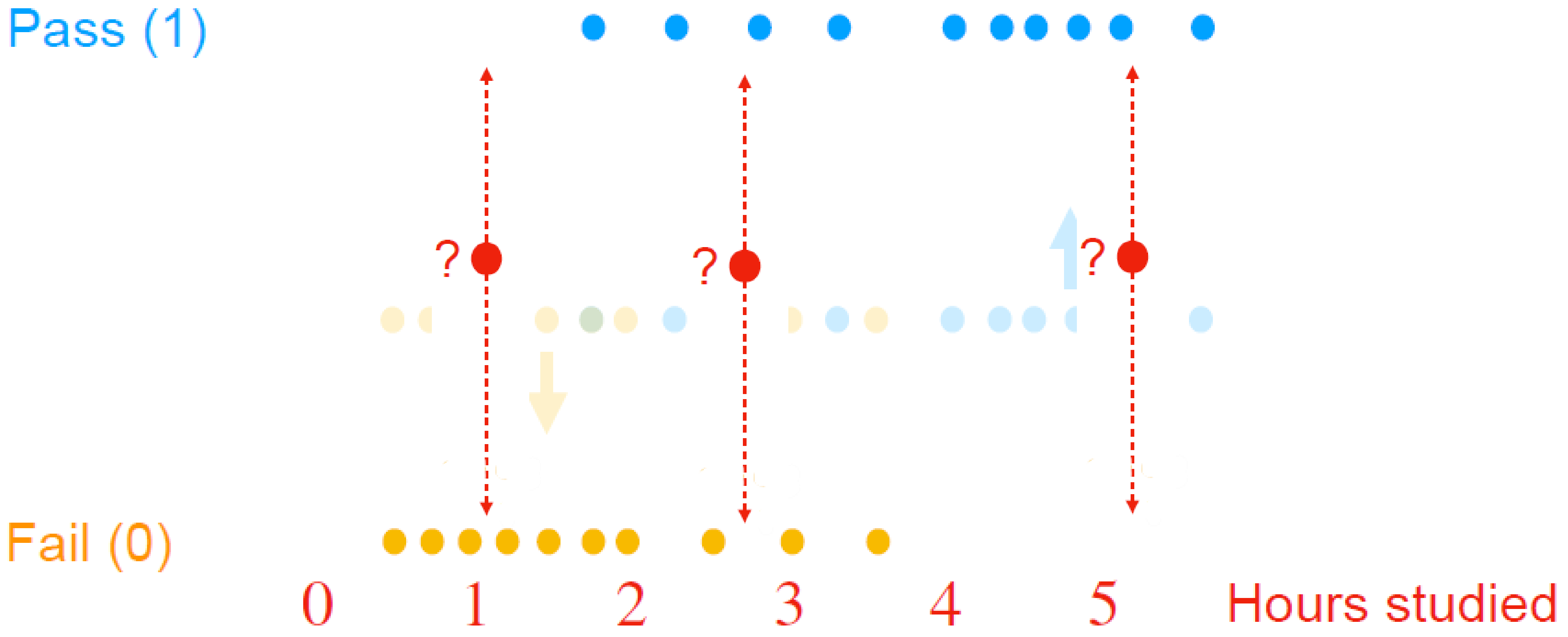
# Classification with logistic regression

- Can we train a model so that given a new data point, we can predict whether that student passes or fails?



- Nearest neighbor: a geometric approach for this problem
- We will now approach this using an alternative probabilistic view

# Classification with logistic regression



Likely fail

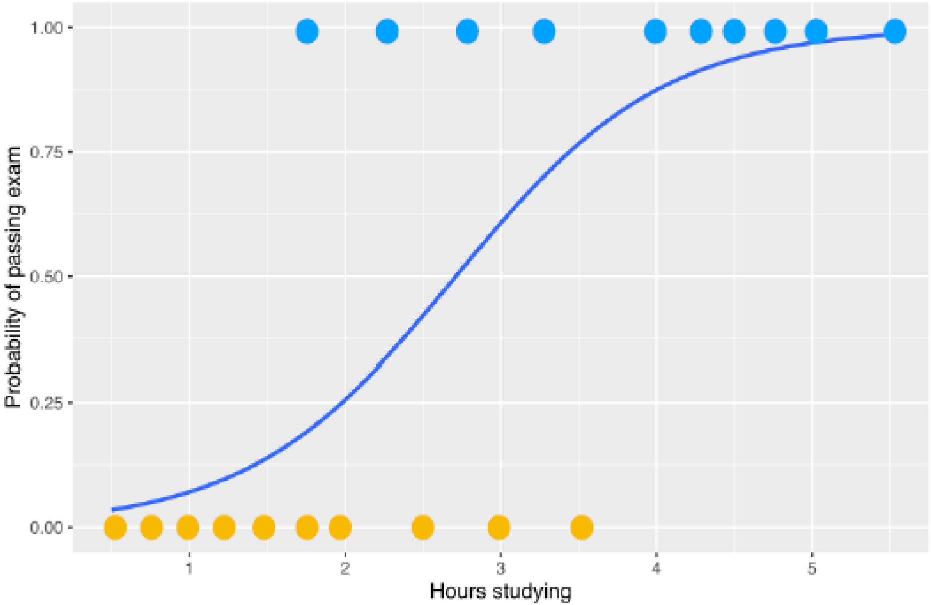
Perhaps 50-50%

Likely pass

# Classification with logistic regression

$Y = 1$

$Y = 0$

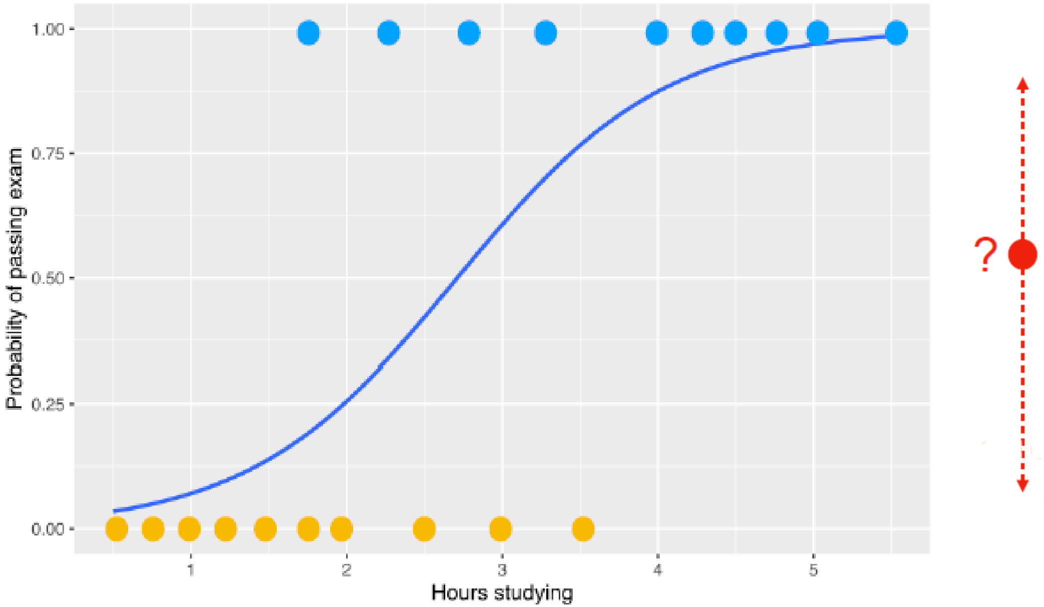


X: feature

Y: label

$$P(Y = 1 \mid X = x)$$

# Classification with logistic regression



Blue curve plots:

$$P(Y = 1 \mid X = x)$$

We can predict the class of test point using blue curve:

If prob < 0.5

predict fail

Else

predict pass

What is a reasonable form of $P(Y = 1 \mid X = x)$?

# Classification with logistic regression

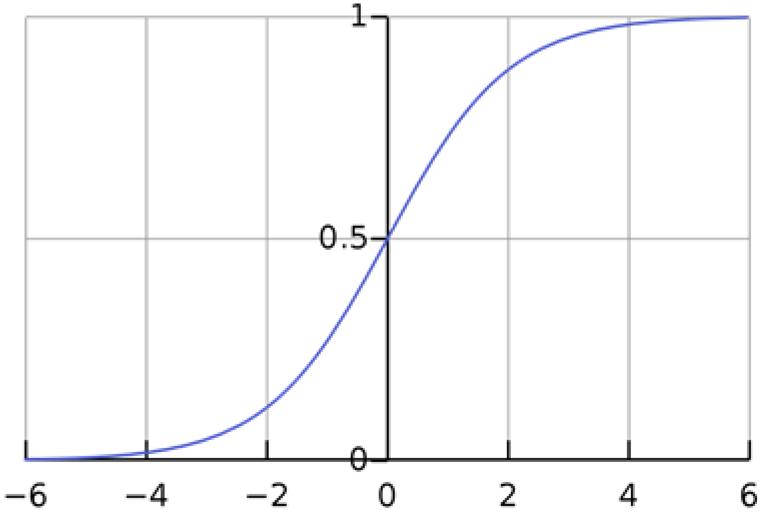
We will assume that:

$$P(Y = 1 \mid X = x) = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

i.e., $\sigma(w \cdot x + b)$, for some $w$, $b$

For d-dim x, this is dot product

$\sigma(z) := \frac{1}{1+e^{-z}}$ is the *logistic function*



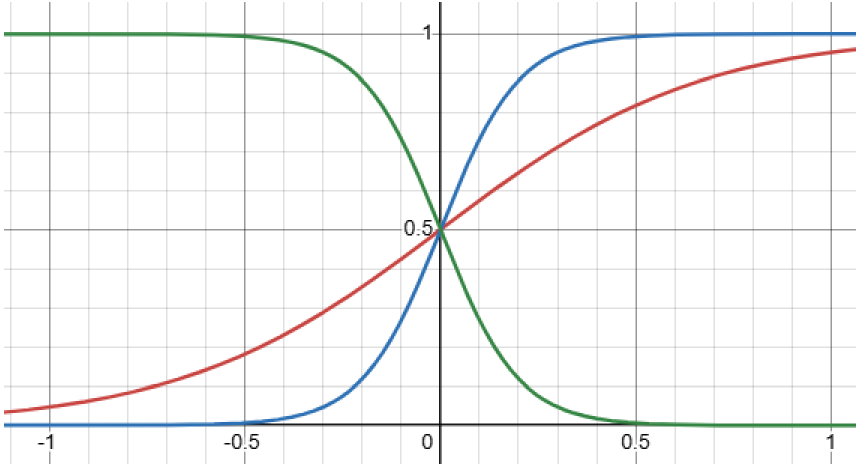
# Logistic regression model

$w$ controls the shape of the probability curve

$$p = \frac{1}{1 + e^{-10\,x}}$$

$$p = \frac{1}{1 + e^{-3\,x}}$$

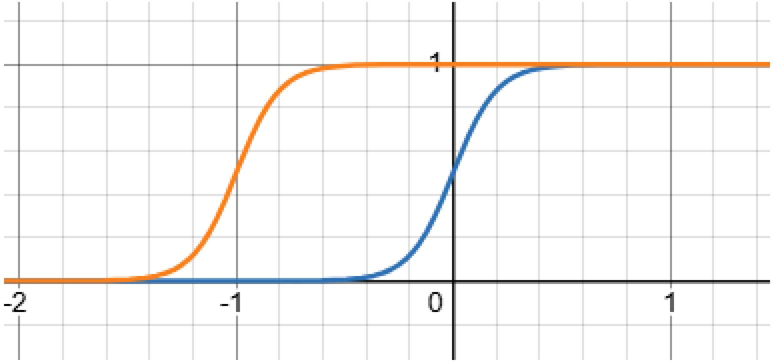
$$p = \frac{1}{1 + e^{10\,x}}$$



$b$ controls the location of the probability curve

$$p = \frac{1}{1 + e^{-10\,x}}$$

$$p = \frac{1}{1 + e^{-10\,(x+1)}}$$



# Classification with logistic regression

**Example** Suppose we fit logistic regression model with b = 0.15 and w = 0.575. What is the model's predicted probability that a student who have studied for x = 2 hours passes?

$P(Y = 1 \mid X = x) = \frac{1}{1+e^{-z}}$, where $z = w \cdot x + b = 1$

Thus, the predicted pass prob = $\frac{1}{1+e^{-1}} = 0.73$

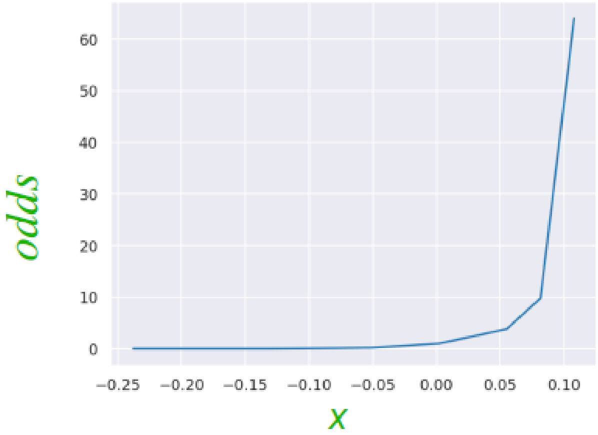
# Classification with logistic regression

Where does the logistic function come from?

- Linear regression $w \cdot x + b$ is good at predicting unbounded outputs $y$
- Idea: transform $p$ to a good unbounded function

$$\text{odd} = \frac{P(Y=1|x)}{P(Y=0|x)} = \frac{p}{1-p}$$

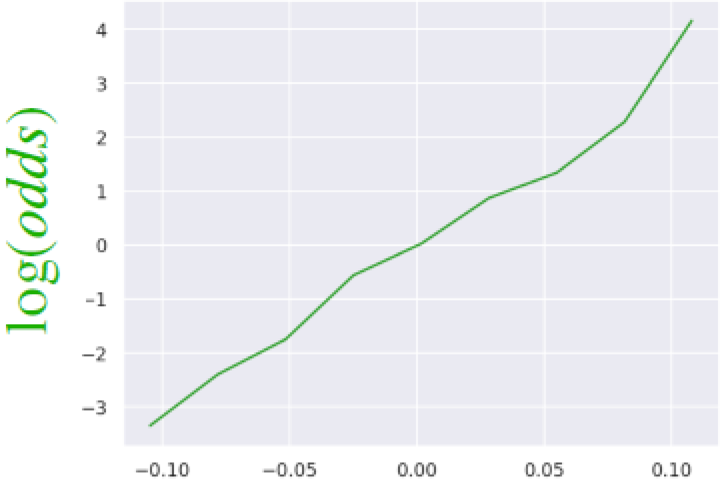
- Still not ideal: odd bounded from below



# Classification with logistic regression

## Where does the logistic function come from?

- Linear regression $w \cdot x + b$ is good at predicting unbounded outputs
- $\log \text{odd} = \ln \frac{p}{1-p}$
- This now can take +/- values

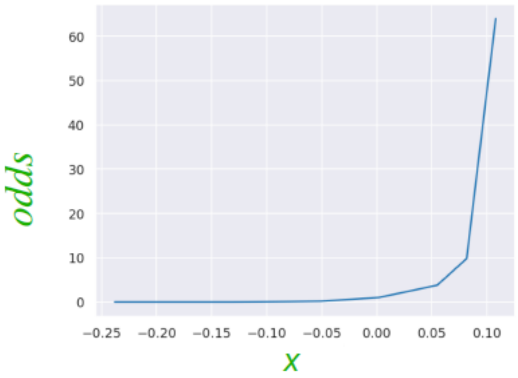


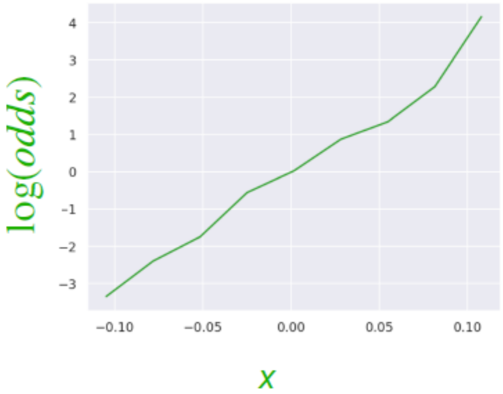
$$\ln \frac{p}{1-p} = w \cdot x + b \quad \Rightarrow \quad \frac{p}{1-p} = e^{w\,x+b} \quad \Rightarrow \quad p = \frac{1}{1+e^{-(w \cdot x+b)}}$$

# Where logistic function come from?

- $w \cdot x + b$ is unbounded
- We want to transform $p$ into a form that produce unbounded outputs
  - $p \rightarrow \frac{p}{1-p} \rightarrow \ln\frac{p}{1-p}$

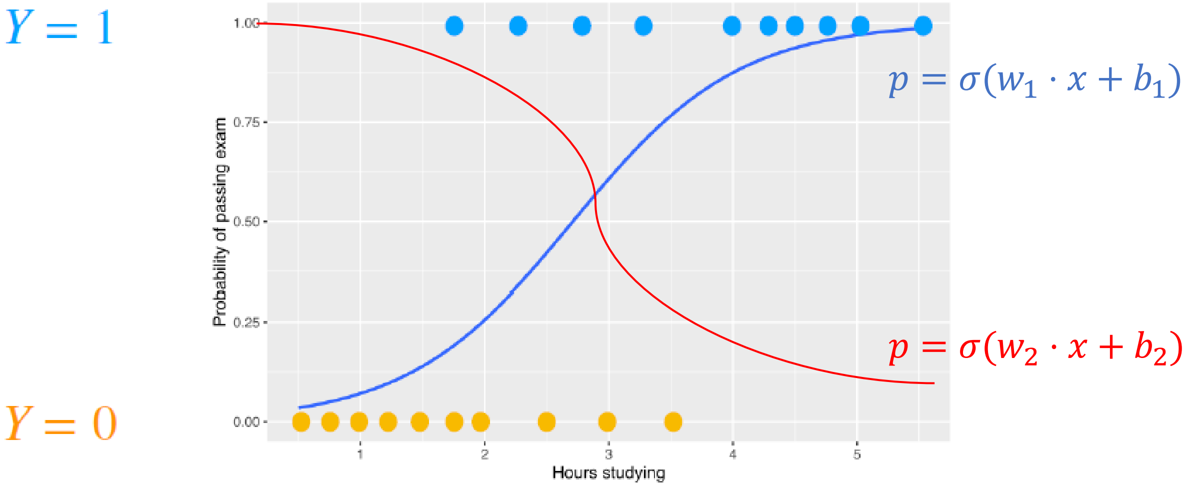
[0, 1] [0, +∞] [-∞, +∞]





# Fitting a logistic regression model

- Recall: loss for linear regression was MSE $\frac{1}{n}\sum_i (y_i - w \cdot x_i)^2$
- How about logistic regression?
  - $y_i$'s are in 0, 1



Which logistic regression model fits data better, red or blue?

# Fitting a logistic regression model

We'd like to choose $w$ and $b$ such that:

for $x$ whose label is more likely to be 1

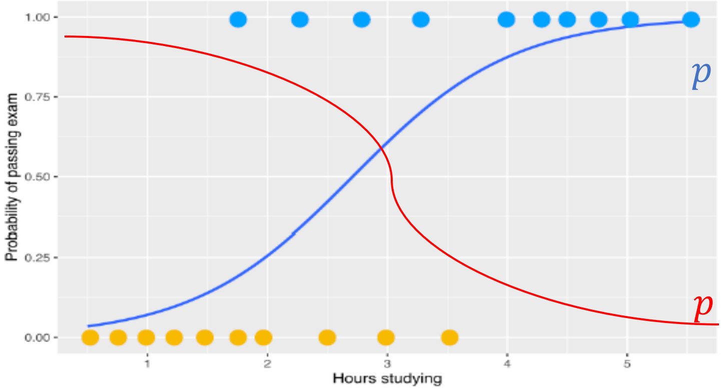
=> $p$ is large

=> $w \cdot x + b$ is large

=> penalize more if $p$ is small

$Y = 1$

$Y = 0$



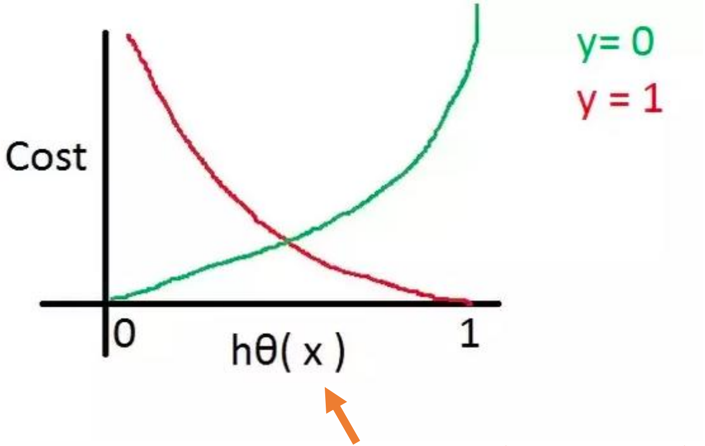
$p = \sigma(w_1 \cdot x + b_1)$

$p = \sigma(w_2 \cdot x + b_2)$

# Cross entropy loss

$$\ell(y,p) = y \ln\frac{1}{p} + (1-y)\ln\frac{1}{1-p}$$

$$= \begin{cases} \ln\frac{1}{p}, \ y = 1 \\ \ln\frac{1}{1-p}, y = 0 \end{cases}$$



$p_i = \sigma(w \cdot x_i + b)$

**Minimizing CE loss incentivizes the model's predictive probability to align with labels**

# Fitting a logistic regression model

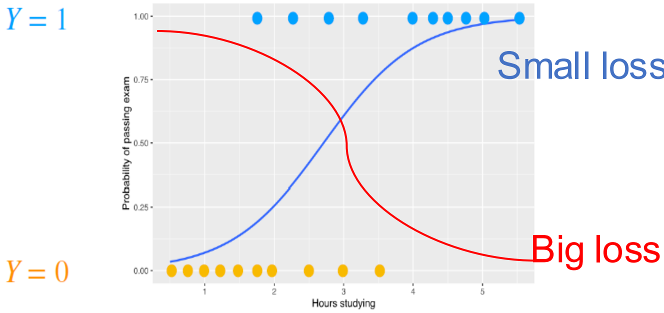
- We find $w$ and $b$ to minimize:

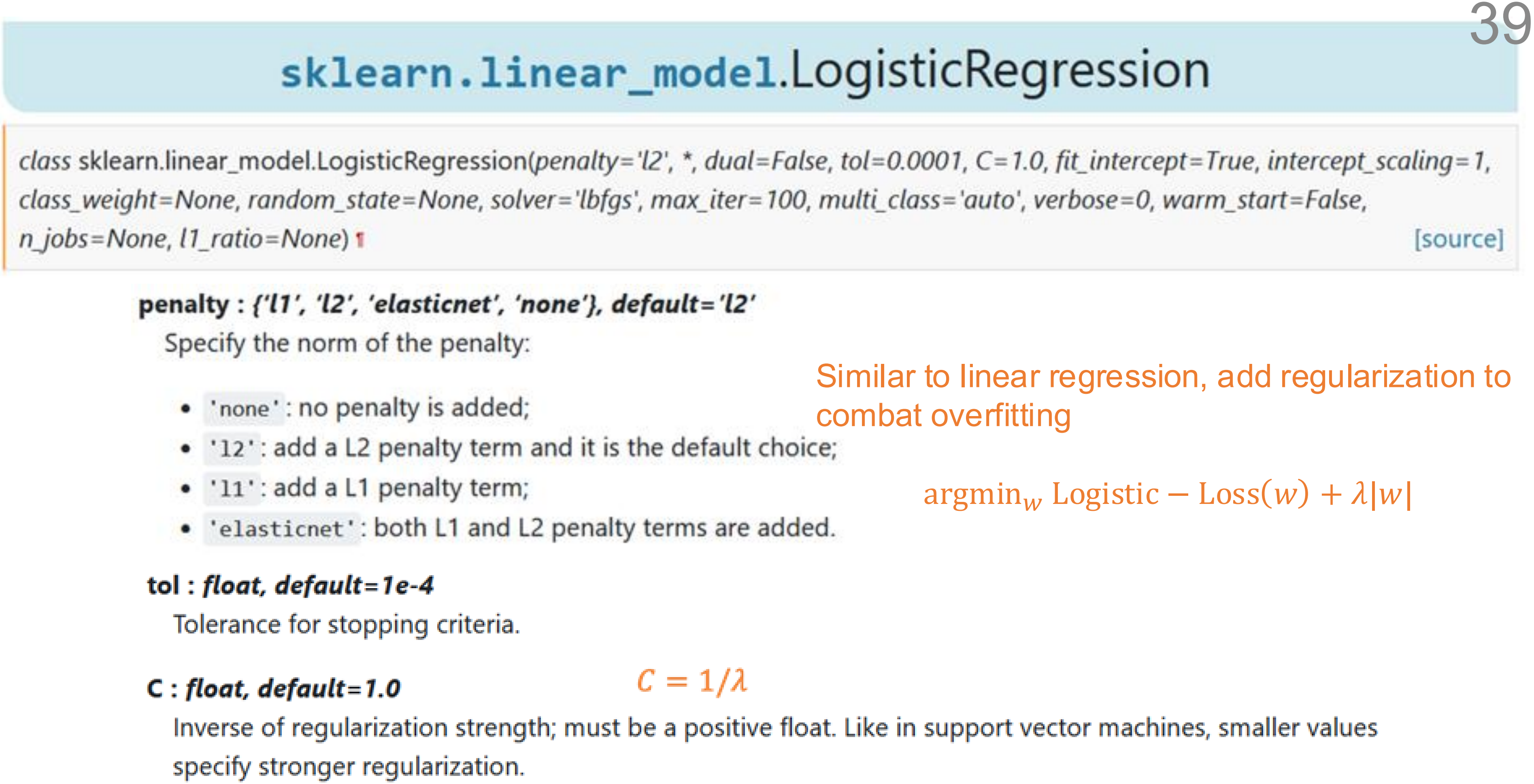
$$\sum_i \left( y_i \ln \frac{1}{p_i} + (1 - y_i) \ln \frac{1}{1-p_i} \right),$$

Logistic loss, aka Cross-entropy (CE) loss

where $p_i = P(\,Y = 1 \mid x_i\,) = \frac{1}{1+e^{-(w \cdot x_i + b)}}$

- What is the $i$-th example's loss when:
  - $y_i = 1$ and $p_i \approx 1$? $\approx 0$
  - $y_i = 1$ and $p_i \approx 0$? Large
  - $y_i = 0$ and $p_i \approx 1$? Large
  - $y_i = 0$ and $p_i \approx 0$? $\approx 0$





# 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]

**penalty : *{'l1', 'l2', 'elasticnet', 'none'}, default='l2'***

Specify the norm of the penalty:

- `'none'`: no penalty is added;
- `'l2'`: add a L2 penalty term and it is the default choice;
- `'l1'`: add a L1 penalty term;
- `'elasticnet'`: both L1 and L2 penalty terms are added.

**tol : *float, default=1e-4***

Tolerance for stopping criteria.

**C : *float, default=1.0***

Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

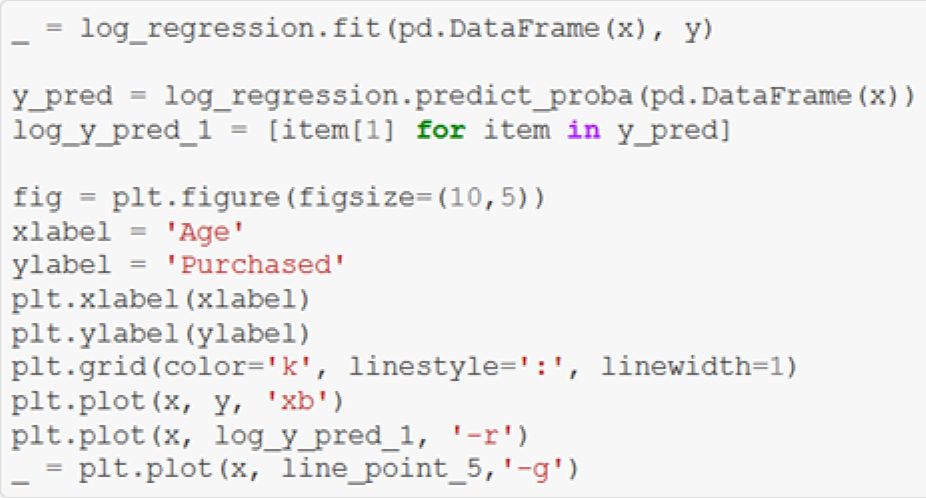
Similar to linear regression, add regularization to combat overfitting

$$\text{argmin}_w \text{ Logistic} - \text{Loss}(w) + \lambda|w|$$

$$C = 1/\lambda$$

# Scikit-Learn Logistic Regression

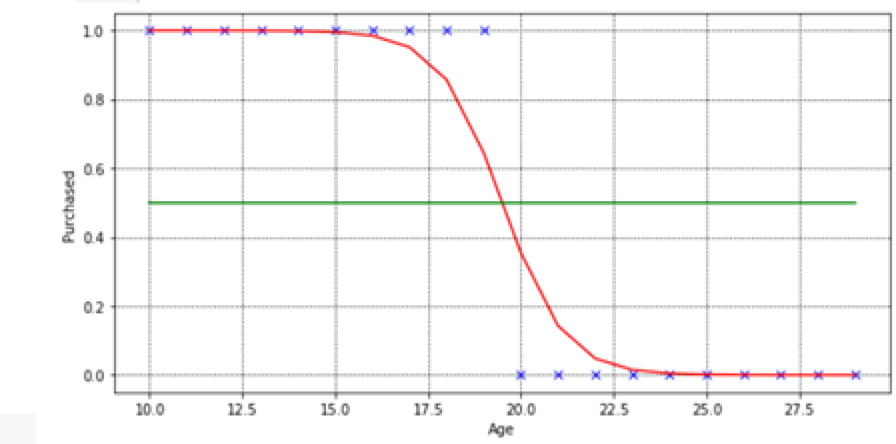
log_regression = sklearn.linear_model.LogisticRegression()



```
_ = log_regression.fit(pd.DataFrame(x), y)

y_pred = log_regression.predict_proba(pd.DataFrame(x))
log_y_pred_1 = [item[1] for item in y_pred]

fig = plt.figure(figsize=(10,5))
xlabel = 'Age'
ylabel = 'Purchased'
plt.xlabel(xlabel)
plt.ylabel(ylabel)
plt.grid(color='k', linestyle=':', linewidth=1)
plt.plot(x, y, 'xb')
plt.plot(x, log_y_pred_1, '-r')
_ = plt.plot(x, line_point_5,'-g')
```

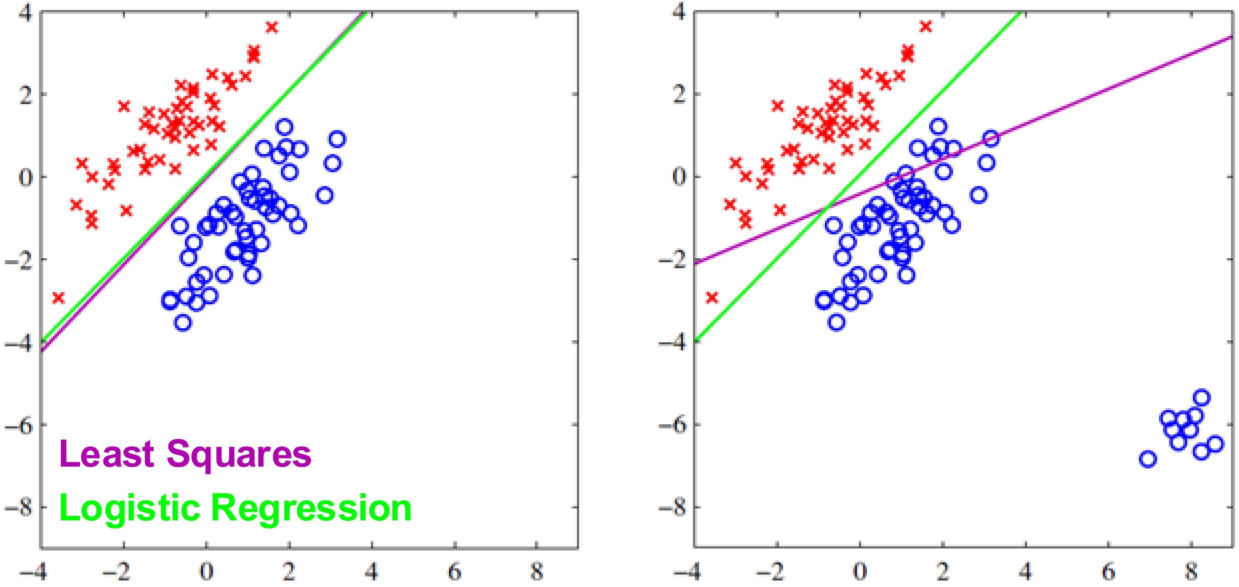


Function `predict_proba(X)` returns prediction of class assignment probabilities for each class. It returns n by C matrix if n data points were provided as argument.

(C=number classes)

https://towardsdatascience.com/why-linear-regression-is-not-suitable-for-binary-classification-c64457be8e28

# Least Squares vs. Logistic Regression



Least squares regression may also be (ab)used for classification

- To predict a class, threshold $w \cdot x + b$ against 0.5
- Not designed for classification though -- sensitive to outliers

[Source: Bishop “PRML”]

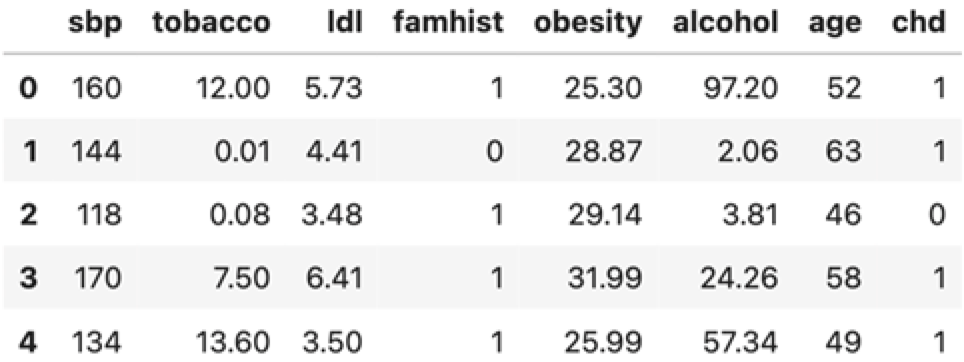
# Using Logistic Regression

**Logistic Regression have two main usages**

- building **predictive** classification models
- **understanding** how features relate to data classes / categories

**Example** South African Heart Disease (Hastie et al. 2001)
Data result from Coronary Risk-Factor Study in 3 rural areas of South Africa. Data are from white men 15-64yrs. Label is presence/absence of *myocardial infarction (MI).*

# Example: African Heart Disease

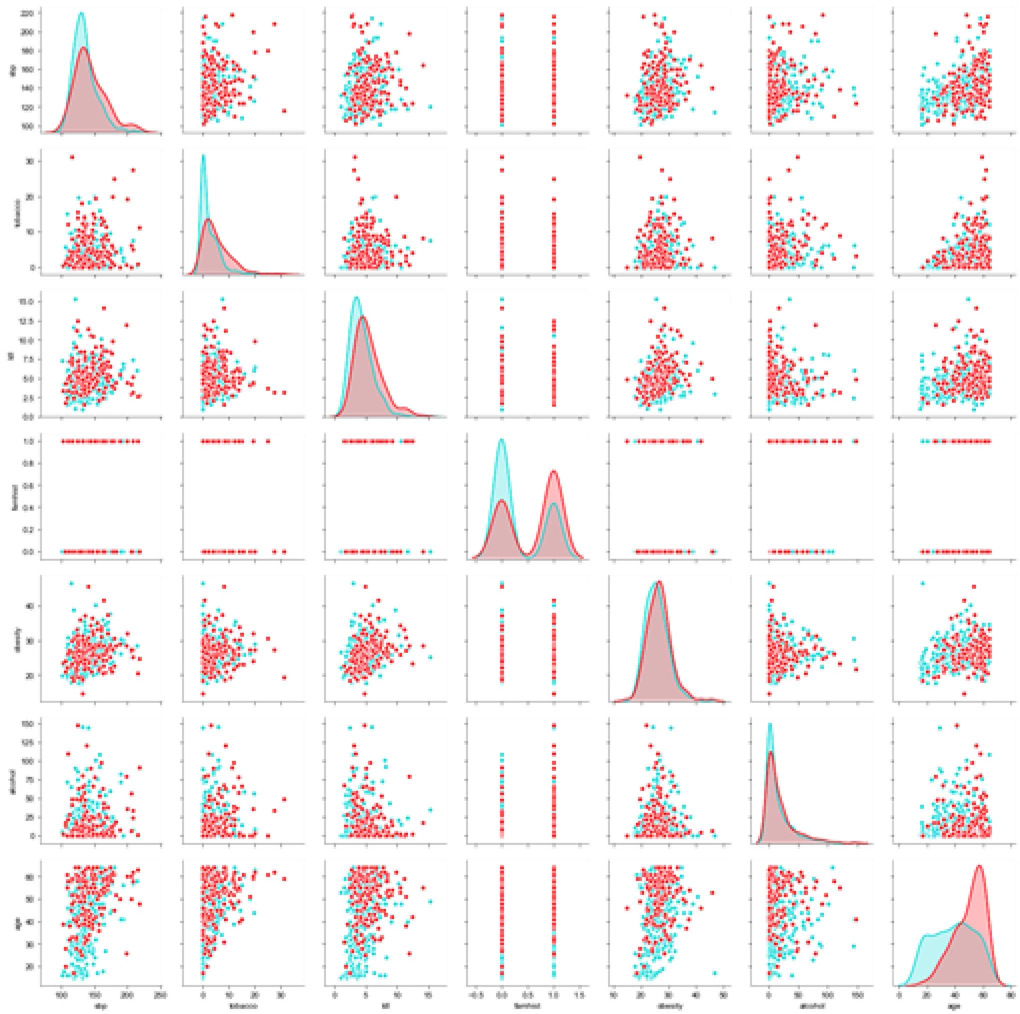


| | sbp | tobacco | ldl | famhist | obesity | alcohol | age | chd |
|---|---|---|---|---|---|---|---|---|
| **0** | 160 | 12.00 | 5.73 | 1 | 25.30 | 97.20 | 52 | 1 |
| **1** | 144 | 0.01 | 4.41 | 0 | 28.87 | 2.06 | 63 | 1 |
| **2** | 118 | 0.08 | 3.48 | 1 | 29.14 | 3.81 | 46 | 0 |
| **3** | 170 | 7.50 | 6.41 | 1 | 31.99 | 24.26 | 58 | 1 |
| **4** | 134 | 13.60 | 3.50 | 1 | 25.99 | 57.34 | 49 | 1 |

**Features**

- Systolic blood pressure
- Tobacco use
- Low density lipoprotein (ldl)
- Family history (discrete)
- Obesity
- Alcohol use
- Age

Q: How predictive is each of the features to myocardial infarction?



# Looking at Data

Each scatterplot shows pair of risk factors.

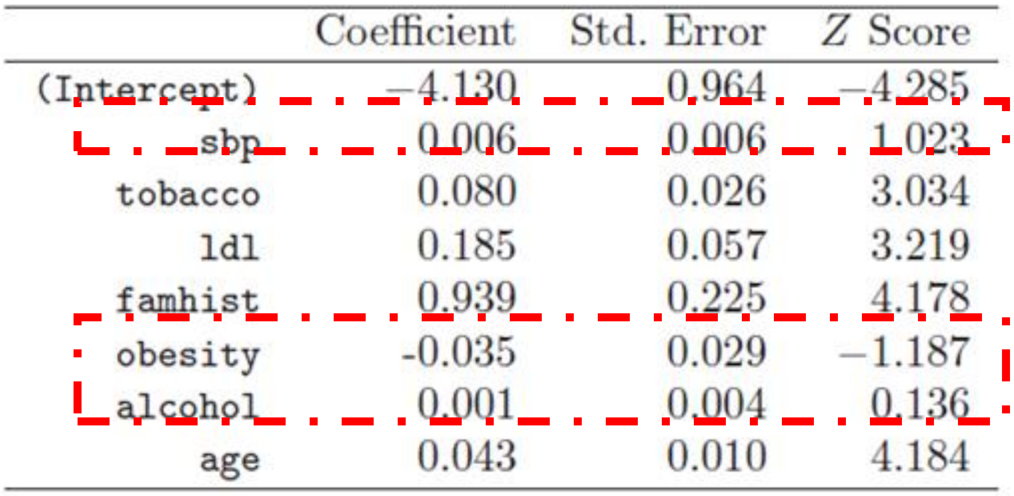
Cases **with** MI (**red**) and **without** (**cyan**)

### Features

- Systolic blood pressure
- Tobacco use
- Low density lipoprotein (ldl)
- Family history (discrete)
- Obesity
- Alcohol use
- Age

[Source: Hastie et al. (2001)]

# Example: African Heart Disease



| | Coefficient | Std. Error | Z Score |
|---|---|---|---|
| (Intercept) | −4.130 | 0.964 | −4.285 |
| sbp | 0.006 | 0.006 | 1.023 |
| tobacco | 0.080 | 0.026 | 3.034 |
| ldl | 0.185 | 0.057 | 3.219 |
| famhist | 0.939 | 0.225 | 4.178 |
| obesity | -0.035 | 0.029 | −1.187 |
| alcohol | 0.001 | 0.004 | 0.136 |
| age | 0.043 | 0.010 | 4.184 |

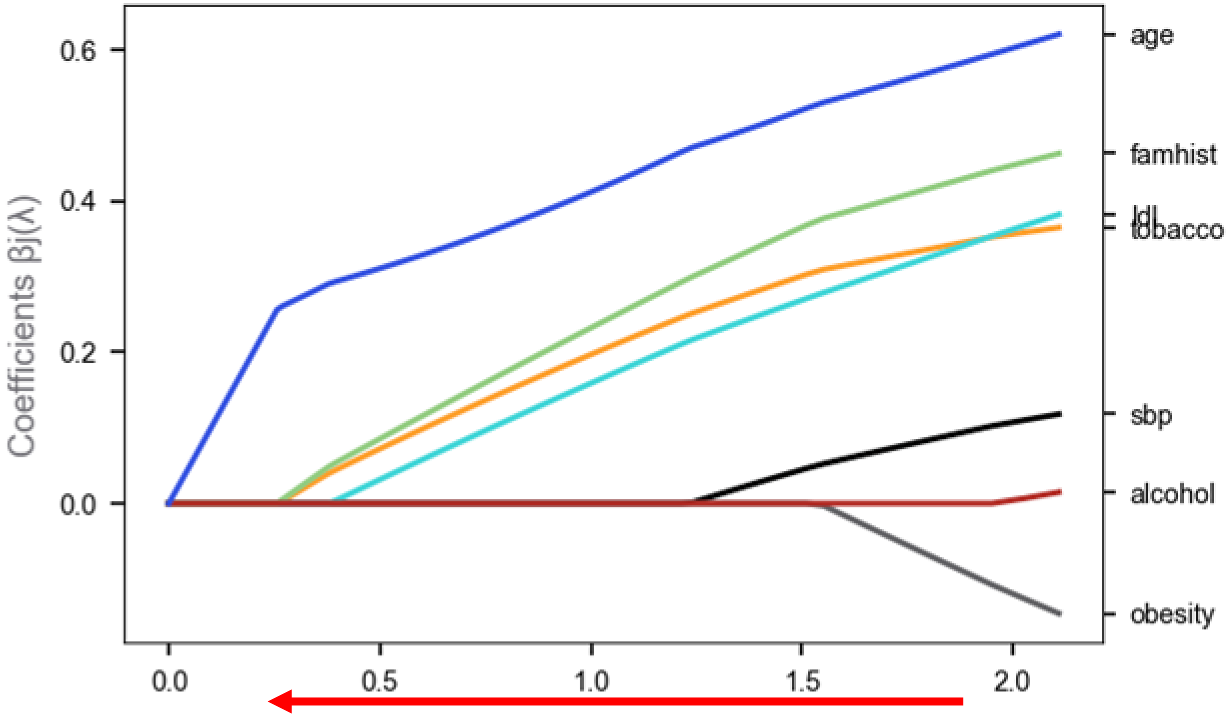
Z-score: score indicating the significance of each feature

|Z-score| > 2: feature is significant (we will learn more in *statistics*)

**Finding** Systolic blood pressure (sbp), obesity, and alcohol are **not significant predictors**

# L1 regularized logistic regression coefficients

$$\mathrm{argmin}_w \ \mathrm{Logistic\_Loss}(w) + \lambda |w|_1$$



**Regularization strength $\lambda$**

https://github.com/empathy87/The-Elements-of-Statistical-Learning-Python-Notebooks/blob/master/examples/South%20African%20Heart%20Disease.ipynb

# Evaluation measures for classification

# Pipeline for machine learning & data analysis

| | |
|---|---|
| 1. Choose a model | How should we represent the world? |
| 2. Choose a loss function | How do we quantify prediction error? |
| 3. Fit the model | How do we choose the best parameters of our model given our data? |
| 4. Evaluate model performance | How do we evaluate whether this process gave rise to a good model? |

# Model evaluation: baseline models

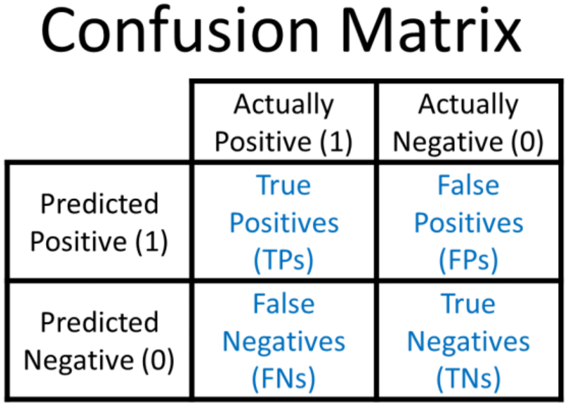
Baseline models: simplest reasonable models to compare against

- Uniform classifier: predict with one label consistently (i.e., Most common label classifier, Least common label classifier)
- Blind classifier: predict random labels

| | Features | | | | Label | Most common Prediction | Least common Prediction | Random Prediction |
|---|---|---|---|---|---|---|---|---|
| Training data | $x_{11}$ | $x_{12}$ | $\cdots$ | $x_{1p}$ | **Yes** | **Yes** | **No** | **Yes** |
| | $x_{21}$ | $x_{22}$ | $\cdots$ | $x_{2p}$ | **No** | **Yes** | **No** | **No** |
| | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ | | | |
| Test data | $x_{n1}$ | $x_{n2}$ | $\cdots$ | $x_{np}$ | **Yes** | **Yes** | **No** | **Yes** |

# Classification: model evaluation

## Confusion matrix



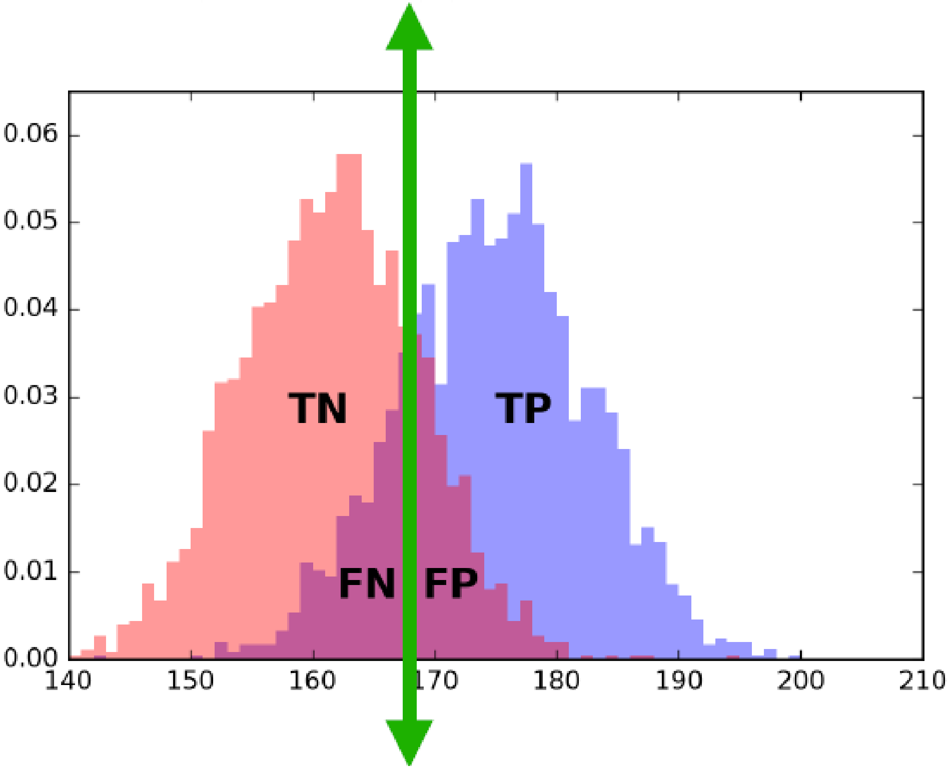
Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | True Positives (TPs) | False Positives (FPs) |
| Predicted Negative (0) | False Negatives (FNs) | True Negatives (TNs) |

- True positives (TP): + labeled as +
- True negatives (TN): - labeled as –
- False positives (FP): - labeled as +
- False negatives (FN): + labeled as -

# Classification: model evaluation

- Example: classify everyone of height ≥168cm as male (1)



# Fill out the tables: TP, TN, FP, FN

Cancer diagnosis. Positive class 5%, consider 100 data points.

- Left: Blind classifier (random label)
- Middle: Most common label classifier (always negative)
- Right: Least common label classifier (always )

## Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | 2.5 | 47.5 |
| Predicted Negative (0) | 2.5 | 47.5 |

## Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | 0 | 0 |
| Predicted Negative (0) | 5 | 95 |

## Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | 5 | 95 |
| Predicted Negative (0) | 0 | 0 |

# Accuracy

**Accuracy** ratio of correct predictions over all predictions

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + TN + FN} = P(f(x) = y)$$

Case 1: classes balanced, Positive class 50%

- Blind classifier accuracy = 50%
  - Predict random label
- Most common label classifier accuracy = 50%
  - Predict majority class (doesn't matter 1 or 0)

Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | True Positives (TPs) | False Positives (FPs) |
| Predicted Negative (0) | False Negatives (FNs) | True Negatives (TNs) |

# Fill out the tables: TP, TN, FP, FN

Cancer diagnosis. Positive class 50%, consider 100 data points.

- Left: Blind classifier (random label)
- Middle: Most common label classifier (always negative)

## Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | 25 | 25 |
| Predicted Negative (0) | 25 | 25 |

## Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | 0 | 0 |
| Predicted Negative (0) | 50 | 50 |

# Accuracy

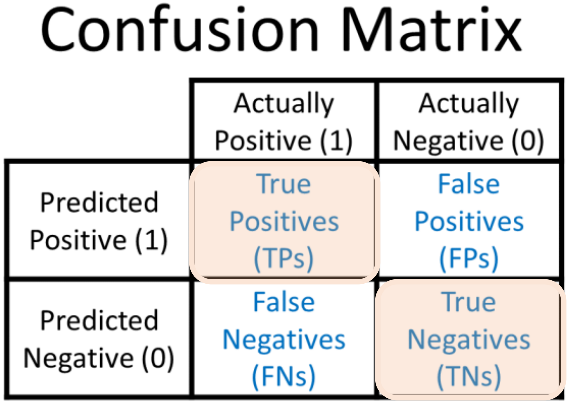
**Accuracy** ratio of correct predictions over all predictions

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + TN + FN} = P(f(x) = y)$$

Case 2: classes are imbalanced, Positive class 5%

- Blind classifier accuracy = 50%
- Most common label classifier accuracy = 95%
  - Always predict 0

Accuracy can be misleading in the presence of class imbalance!



Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | True Positives (TPs) | False Positives (FPs) |
| Predicted Negative (0) | False Negatives (FNs) | True Negatives (TNs) |

# Fill out the tables: TP, TN, FP, FN

Cancer diagnosis. Positive class 5%, consider 100 data points.

- Left: Blind classifier (random label)
- Middle: Most common label classifier (always negative)

## Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | 2.5 | 47.5 |
| Predicted Negative (0) | 2.5 | 47.5 |

## Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | 0 | 0 |
| Predicted Negative (0) | 5 | 95 |

# Precision

**Precision** ratio of correct positive predictions over positive predictions

$$\text{Precision} = \frac{TP}{TP + FP} = P(y = 1 \mid f(x) = 1)$$

**Example** cancer diagnosis. Positive class 5%

Blind classifier precision = $P(y = 1) =$ 5%

Most common label classifier precision

= $P(y = 1 \mid f(x) = 1) =$ undefined

Least common label classifier precision

= $P(y = 1) =$ 5%

## Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | True Positives (TPs) | False Positives (FPs) |
| Predicted Negative (0) | False Negatives (FNs) | True Negatives (TNs) |

# Precision

Cancer diagnosis. Positive class 5%

- Left: Blind classifier
- Middle: Most common label classifier
- Right: Least common label classifier

Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | 2.5 | 47.5 |
| Predicted Negative (0) | 2.5 | 47.5 |

Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | 0 | 0 |
| Predicted Negative (0) | 5 | 95 |

Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | 5 | 95 |
| Predicted Negative (0) | 0 | 0 |

# Recall

**Recall** ratio of correct positive predictions over all positive instances

$$\text{Recall} = \frac{TP}{P} = \frac{TP}{TP + FN} = P(f(x) = 1 \mid y = 1)$$

**Example** cancer diagnosis. Positive class 5%

Blind classifier recall $= P(f(x) = 1) = 50\%$

f(x) and y are independent

Most common label classifier recall

$= P(f(x) = 1 \mid y = 1) = 0$

Least common label classifier recall

$= P(f(x) = 1 \mid y = 1) = 100\%$

## Confusion Matrix

| | Actually Positive (1) | Actually Negative (0) |
|---|---|---|
| Predicted Positive (1) | True Positives (TPs) | False Positives (FPs) |
| Predicted Negative (0) | False Negatives (FNs) | True Negatives (TNs) |

# Precision vs Recall

$$\text{Precision} = \frac{TP}{TP+FP} \qquad \text{Recall} = \frac{TP}{TP+FN}$$

Are false positives more important than false negatives?

Diagnostic screening for cancer:

Recall is more important

Might be OK to have FPs but not OK to have FNs

Spam filtering:

Precision is more important

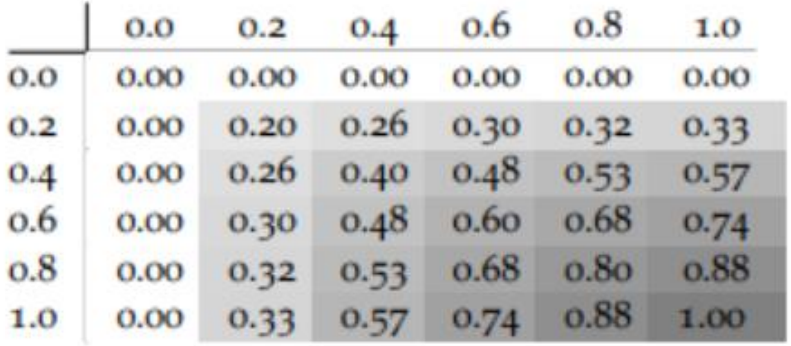
Might be OK to have FNs but not OK to have FPs

# F1-score

**F1-score** Harmonic mean of precision and recall

$$\mathrm{F1} = \frac{2\ \mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

F1 score high => need both precision and recall to be high



| | 0.0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 |
|---|---|---|---|---|---|---|
| 0.0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.2 | 0.00 | 0.20 | 0.26 | 0.30 | 0.32 | 0.33 |
| 0.4 | 0.00 | 0.26 | 0.40 | 0.48 | 0.53 | 0.57 |
| 0.6 | 0.00 | 0.30 | 0.48 | 0.60 | 0.68 | 0.74 |
| 0.8 | 0.00 | 0.32 | 0.53 | 0.68 | 0.80 | 0.88 |
| 1.0 | 0.00 | 0.33 | 0.57 | 0.74 | 0.88 | 1.00 |

Table 5.2: Table of f-measures when varying precision and recall values.

# Summary of recall & precision

- Class imbalanced ⇒ accuracy is misleading.

- High recall is easy to achieve
  - just predict + always

- High precision is hard to achieve in imbalanced classification

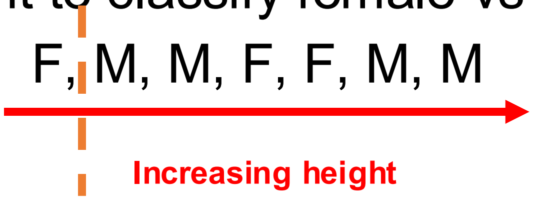
- F-score does the best job of any single statistic, but all four work together to describe the performance of a classifier.

# Receiver Operating Characteristic (ROC) curve

Class prediction often done by thresholding some ‘score’

- Changing the threshold results in a family of classifiers
- How to measure the performance of them altogether?

**Example** Use height to classify female vs male:



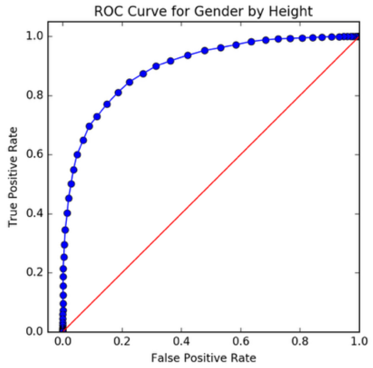
F, M, M, F, F, M, M

Increasing height

For each threshold, calculate

True positive rate $TPR = \frac{TP}{P}$ 4 4

False positive rate $FPR = \frac{FP}{N}$ 2 3

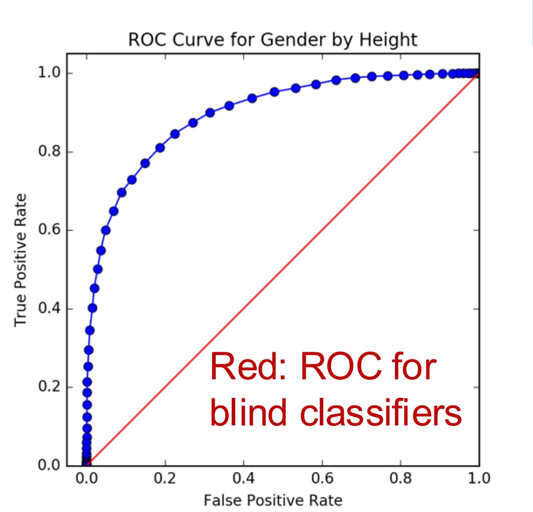


# ROC Curve

For each threshold, calculate

True positive rate $TPR = \frac{TP}{P}$

False positive rate $FPR = \frac{FP}{N}$



Red: ROC for blind classifiers

The higher the curve, the better the classifiers & scores are

Area under ROC curve (AUROC): how good is the score when used for classification

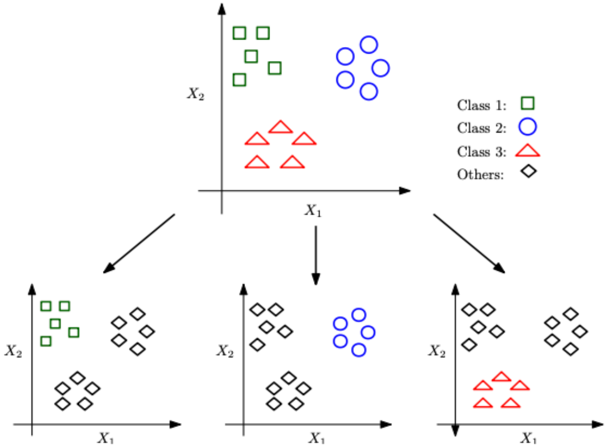
# Multiclass classification

# Multiclass classification

- Multiclass classification naturally happens in many real-world applications
  - E.g. movies: “drama”, “action”, “documentary”
  - E.g. objects: “automobile”, “airplanes”, “trees”
- Some of our methods are designed for binary classification, e.g. logistic regression
  - How to extend them to multiclass prediction?

# Approach 1: reducing multiclass to binary

The “One-vs-rest” approach:



**Training**

For each class i:

Train $f_i$ that classifies class i against the rest

**Test**

For test example x:

Use each $f_i$ to predict its probability of belonging class i

Pick class i that has the highest probability

Alternative: “all pairs” approach

# Approach 2: modifying the learning algorithm

**Case study** How to extend logistic regression to multiclass?

The multinomial logistic regression model:

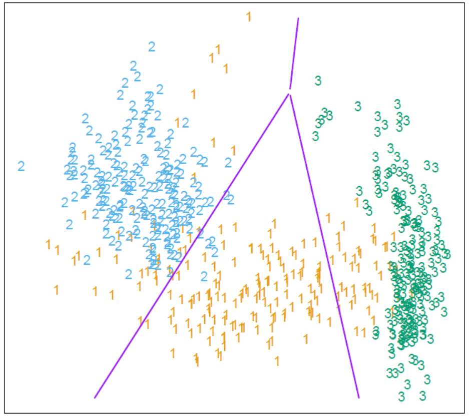
$p(C = i \mid x)$ 's sum to 1

$$\log \frac{p(C=1 \mid x)}{p(C=K \mid x)} = w_1^T x$$

$$\log \frac{p(C=2 \mid x)}{p(C=K \mid x)} = w_2^T x$$

$$\vdots$$

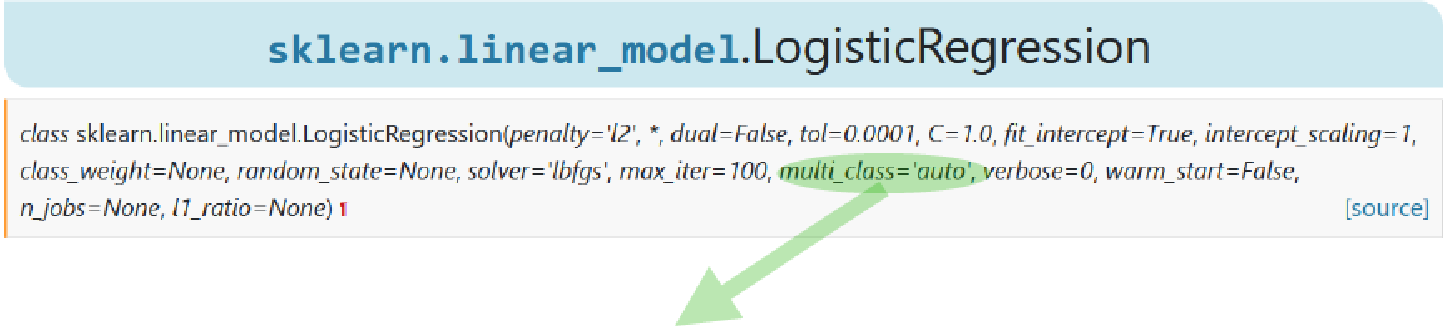
$$\log \frac{p(C=K-1 \mid x)}{p(C=K \mid x)} = w_{K-1}^T x$$



Can define multiclass cross entropy loss accordingly, and minimize it

# Comparison

- Reducing multiclass to binary
  - Pros: simple & general – binary classification algorithm as ‘black box’
- Modifying the learning algorithm
  - Pros: can exploit the structure of the problem more



## 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]

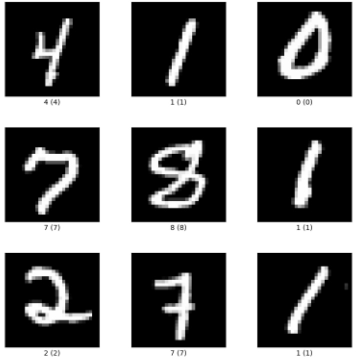
multi_class=‘ovr’ implements one-vs-rest

multi_class=‘multinomial’ implements multinomial

# Multiclass classification: performance metrics

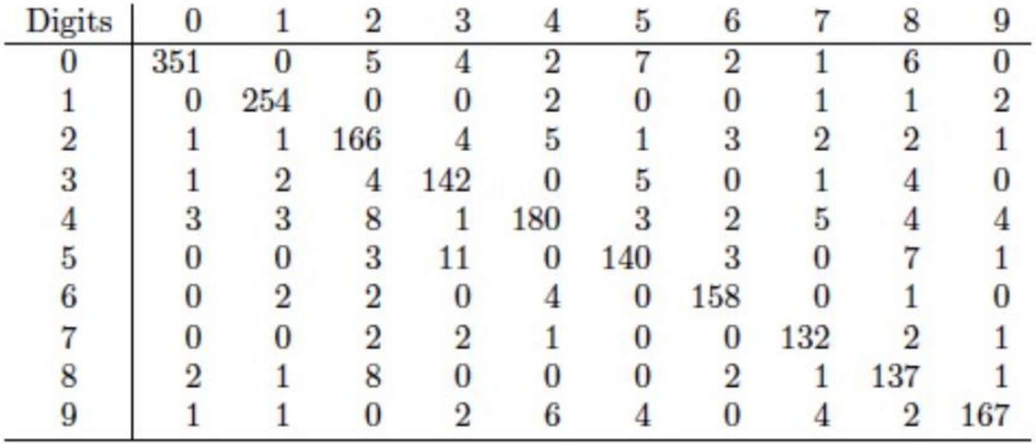
## Multiclass confusion matrix

**Example** Optical Character Recognition



Predicted label $f(x)$

Actual label $y$



| Digits | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 351 | 0 | 5 | 4 | 2 | 7 | 2 | 1 | 6 | 0 |
| 1 | 0 | 254 | 0 | 0 | 2 | 0 | 0 | 1 | 1 | 2 |
| 2 | 1 | 1 | 166 | 4 | 5 | 1 | 3 | 2 | 2 | 1 |
| 3 | 1 | 2 | 4 | 142 | 0 | 5 | 0 | 1 | 4 | 0 |
| 4 | 3 | 3 | 8 | 1 | 180 | 3 | 2 | 5 | 4 | 4 |
| 5 | 0 | 0 | 3 | 11 | 0 | 140 | 3 | 0 | 7 | 1 |
| 6 | 0 | 2 | 2 | 0 | 4 | 0 | 158 | 0 | 1 | 0 |
| 7 | 0 | 0 | 2 | 2 | 1 | 0 | 0 | 132 | 2 | 1 |
| 8 | 2 | 1 | 8 | 0 | 0 | 0 | 2 | 1 | 137 | 1 |
| 9 | 1 | 1 | 0 | 2 | 6 | 4 | 0 | 4 | 2 | 167 |

# Multiclass classification: performance metrics

We can precision & recall separately for each class $i$

$$\text{Precision}_i = P(\, y = i \mid f(x) = i \,) = \frac{C[i,i]}{\sum_j C[j,i]}$$

$$\text{Recall}_i = P(\, f(x) = i \mid y = i \,) = \frac{C[i,i]}{\sum_j C[i,j]}$$

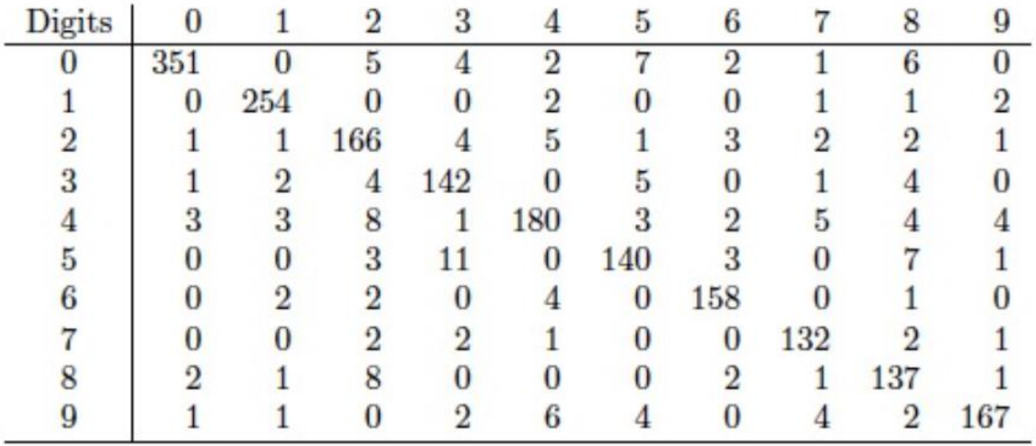
**Example**

$$\text{Precision}_0 = \frac{351}{359}$$

$$\text{Recall}_0 = \frac{351}{378}$$

Predicted label $f(x)$

Actual label $y$



| Digits | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 351 | 0 | 5 | 4 | 2 | 7 | 2 | 1 | 6 | 0 |
| 1 | 0 | 254 | 0 | 0 | 2 | 0 | 0 | 1 | 1 | 2 |
| 2 | 1 | 1 | 166 | 4 | 5 | 1 | 3 | 2 | 2 | 1 |
| 3 | 1 | 2 | 4 | 142 | 0 | 5 | 0 | 1 | 4 | 0 |
| 4 | 3 | 3 | 8 | 1 | 180 | 3 | 2 | 5 | 4 | 4 |
| 5 | 0 | 0 | 3 | 11 | 0 | 140 | 3 | 0 | 7 | 1 |
| 6 | 0 | 2 | 2 | 0 | 4 | 0 | 158 | 0 | 1 | 0 |
| 7 | 0 | 0 | 2 | 2 | 1 | 0 | 0 | 132 | 2 | 1 |
| 8 | 2 | 1 | 8 | 0 | 0 | 0 | 2 | 1 | 137 | 1 |
| 9 | 1 | 1 | 0 | 2 | 6 | 4 | 0 | 4 | 2 | 167 |