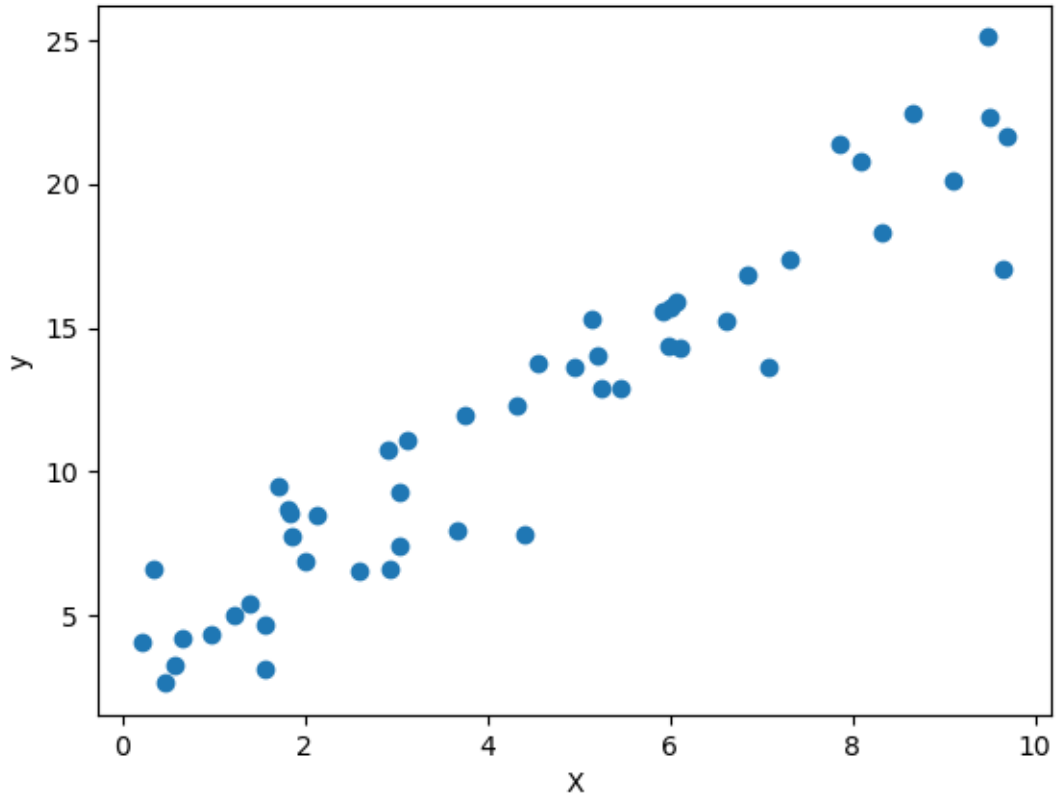
# INFO5002: Intro to Python for Info Sys

## Regression

Slides created by: Zachary Doucet


Northeastern University

# What if our data looks like this?




Looks like a linear relationship!

$$y_i = mx_i + b$$

# Performance is sum of squared errors

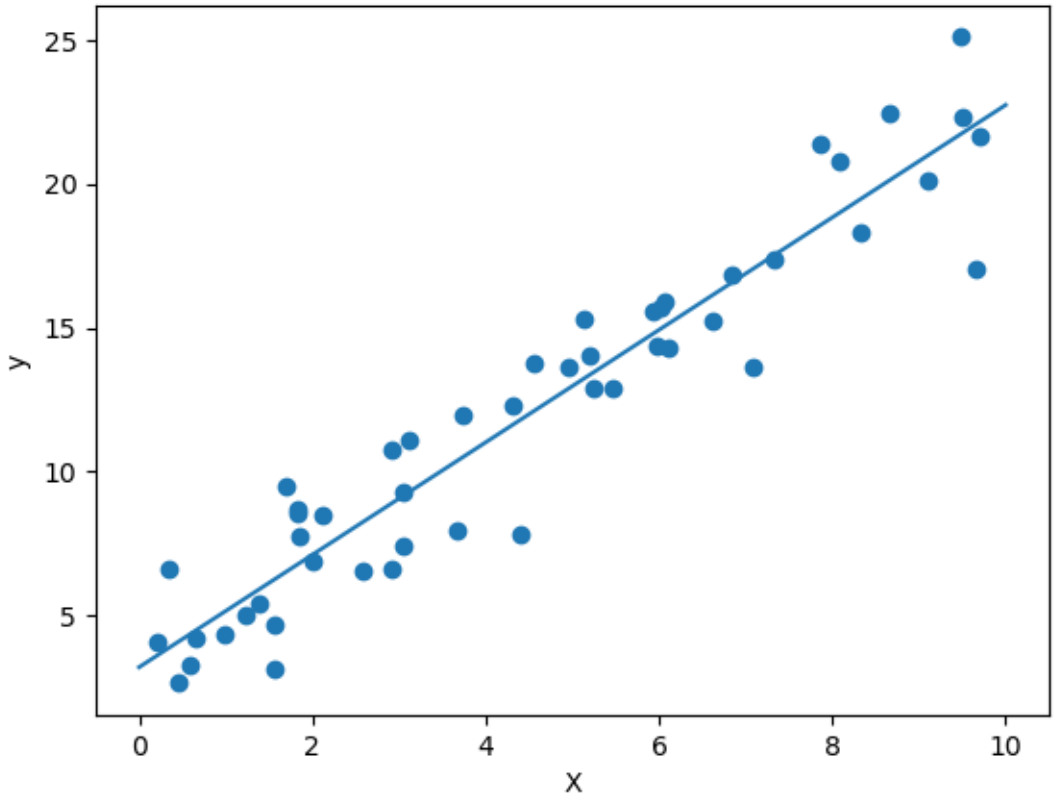
- Error is simply the difference between what we predict and the actual.

```
def error(y_pred, y):
    return y_pred - y
```

- We can maybe calculate our total error of our model!
  - But if one point has error -1 and the other +1 => they cancel.
- Instead we need to square the errors.

```
def sqr_error(x, y, m, b):
    return sum([error(predict(x_i, m, b), y_i)**2
      for x_i, y_i in zip(x, y)])
```

# The result




- Doesn’t that look good.
- We need a way to evaluate how well it does besides error.

# General training paradigm

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
# To train the model
model.fit(X_train, y_train)

# To predict some values on the trained model
y_pred = model.predict(X_test)

# To score the model
model.score(X_test, y_test)

# Get the coefficient(s)
model.coef_

# Get the intercept
model.intercept_
```

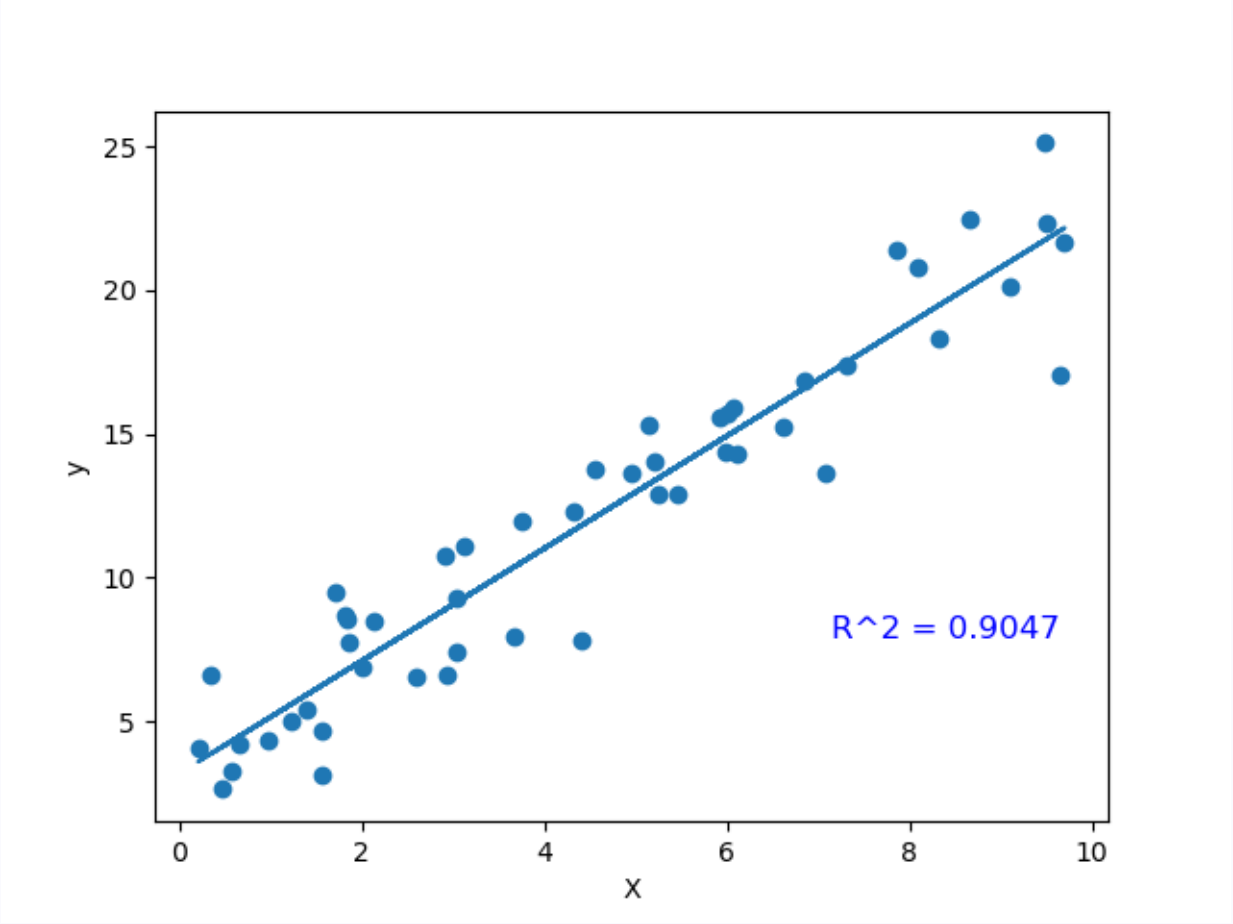
# Coefficient of determination (R-squared)

- Measures the total variation of the dependent variable that is captured by the model. This is what score returns.

$$R^2 = 1 - \frac{SSE}{SS_{tot}} = \frac{\sum (y_i - \hat{y})^2}{\sum (y_i - \bar{y})^2}$$

- Score of 1 means perfectly encapsulated.

# The result



# Multiple Regression

DSfS 191-202

# What if we have more than 1 feature?

- If we have more than 1 feature then we say we are performing multiple linear regression.

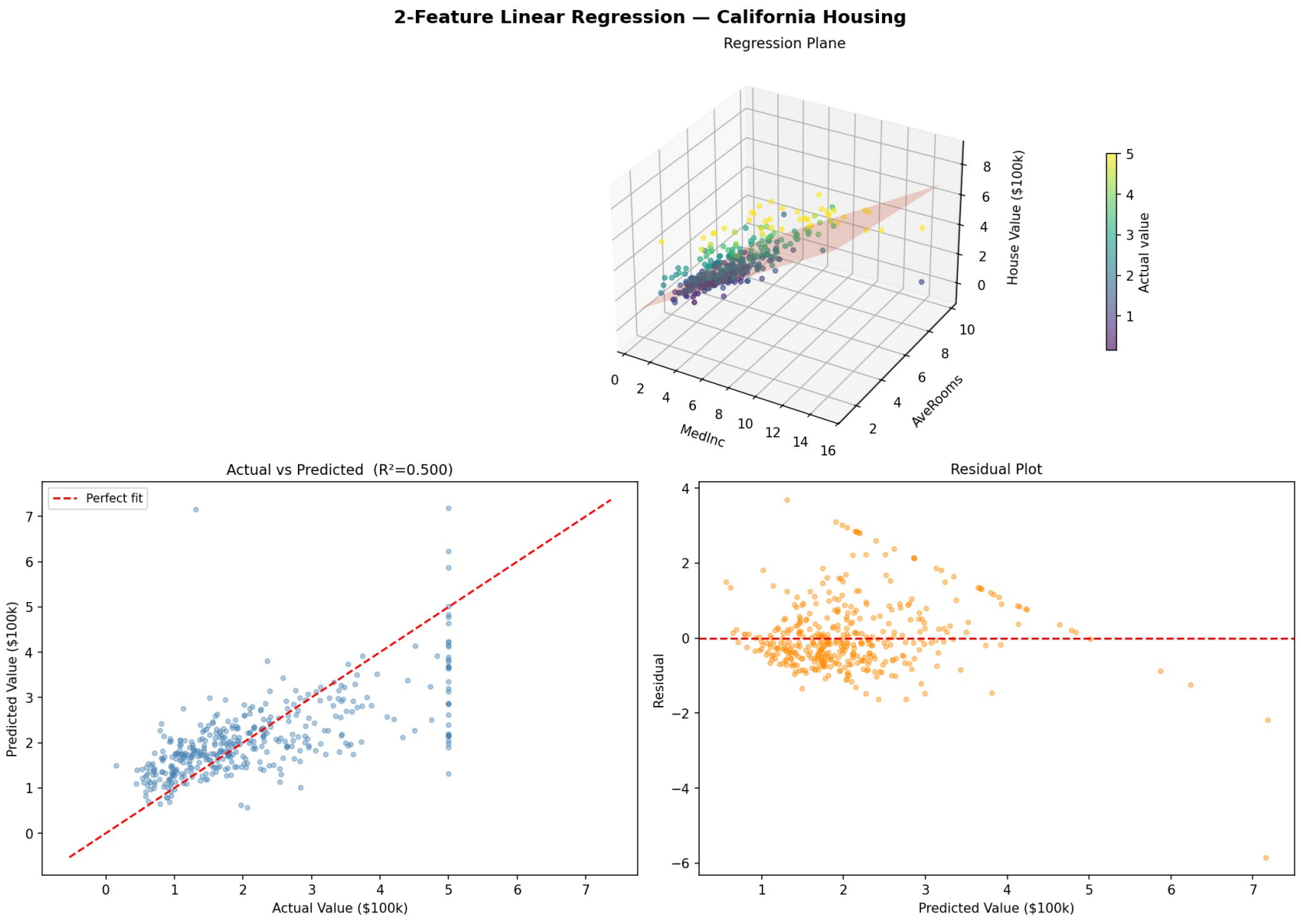
$$\hat{y} = \alpha + \beta_1 * x_1 + \beta_2 * x_2 + \ldots + \beta_k * x_k$$

Feature 1 Feature k

- Imagine we wanted to predict the grade a student gets based on number of hours studied AND number lectures attended.

# We can still use same code

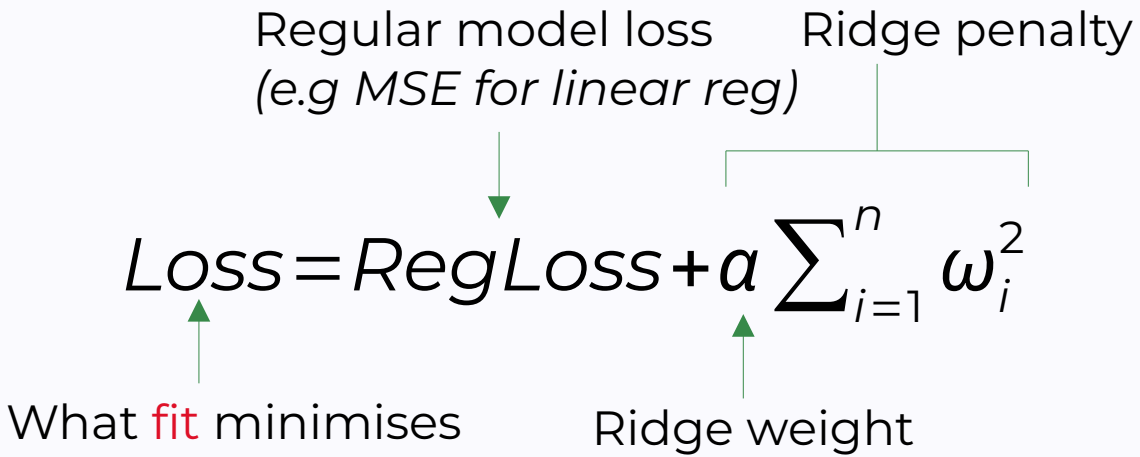
- But our input data needs to follow three restrictions:
  - Each feature should not be perfectly correlated to another (e.g Celsius and Fahrenheit).
  - You need to have more data than features.
  - All columns in matrix should be linearly independent.


2-Feature Linear Regression — California Housing
Regression Plane
House Value ($100k)
Actual value
MedInc
AveRooms
Actual vs Predicted (R²=0.500)
Perfect fit
Predicted Value ($100k)
Actual Value ($100k)
Residual Plot
Residual
Predicted Value ($100k)

# Penalties

# Ridge (L2 regularisation)

- Used to prevent overfitting by adding a penalty that is proportional to the square of the coefficient magnitudes

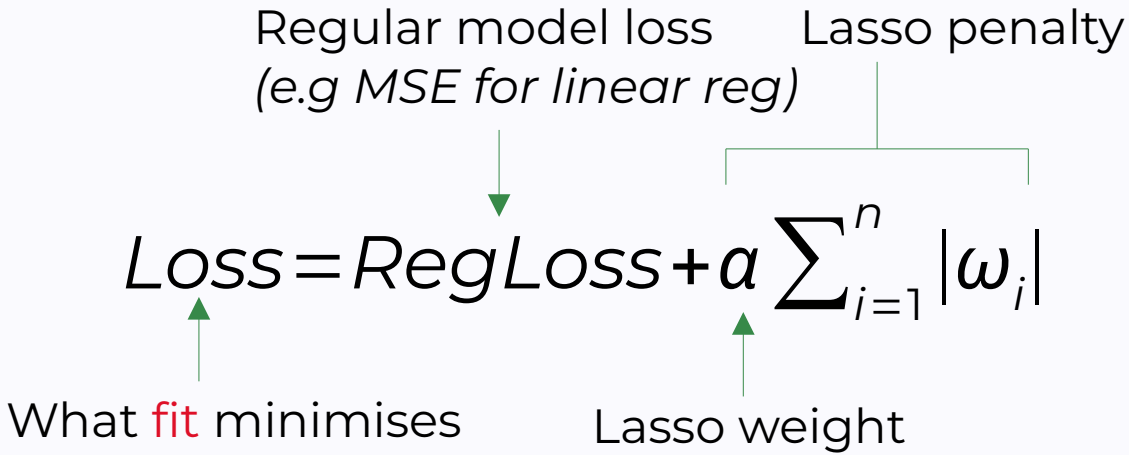


# Using Ridge

```
from sklearn.linear_model import Ridge

model = Ridge(alpha=1.0, fit_intercept=True, max_iter=None,
    tol=0.0001, solver="auto", random_state=None)

# To train the model
model.fit(X_train, y_train)


 ...
```

# Lasso (L1 regularisation)

- Used to prevent overfitting by adding a penalty that is proportional to the absolute value coefficient magnitudes



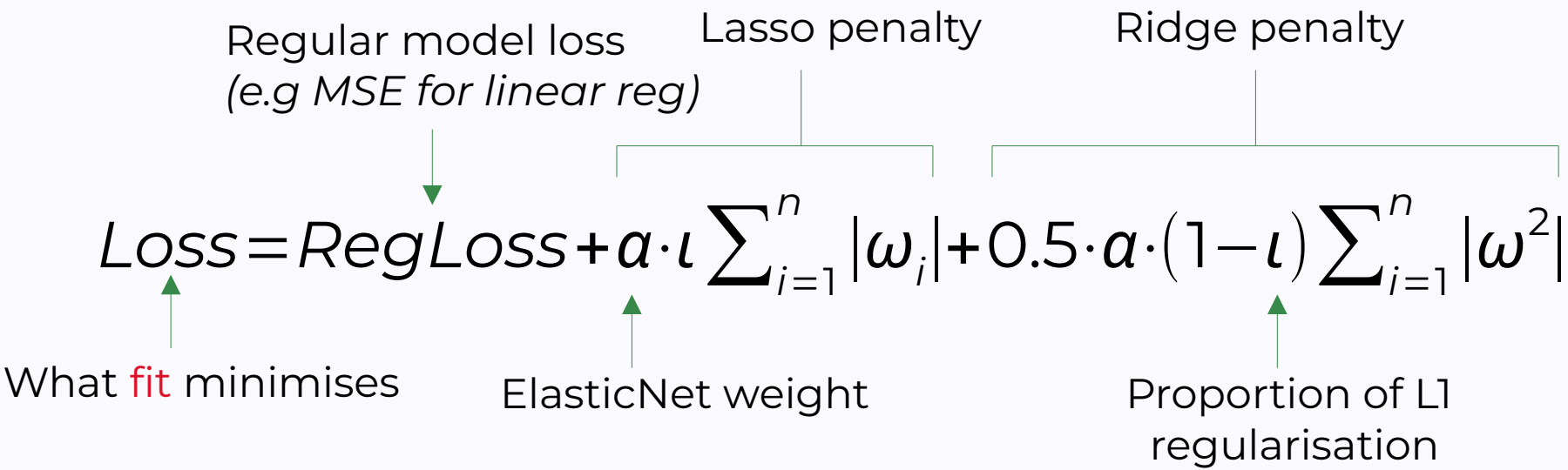
# Using Lasso

```
from sklearn.linear_model import Lasso

model = Lasso(alpha=1.0, fit_intercept=True, precompute=False,
    max_iter=None, tol=0.0001, warm_start=False,
    random_state=None, selection="cyclic")

# To train the model
model.fit(X_train, y_train)

...
```

Useful for feature selection. Less important features will go towards 0.

# ElasticNet

- Combines L1 and L2 regularisation



# Using ElasticNet

```
from sklearn.linear_model import ElasticNet

model = ElasticNet(alpha=1.0, l1_ratio=0.5,
    fit_intercept=True, precompute=False, max_iter=None,
    tol=0.0001, warm_start=False, random_state=None,
    selection="cyclic")

# To train the model
model.fit(X_train, y_train)

 ...
```

# Logistic Regression

DSfS 203-214

# What if we want our output

- To be a probability or between [0, 1]?
- Our simple and multiple linear regression can give us arbitrarily large and small values.
- We need to use a special function.

# Logistic Function

$$logistic(x) = \frac{1}{1+e^{-x}}$$

```
def logistic(x):
    return 1.0 / (1 + math.exp(-x))
```

- As x increases, e^-x gets smaller => closer to 1.
- As x decreases, e^-x gets bigger => closer to 0.

# Logistic Regression

$$\hat{y} = f(X\beta)$$

Where f is the logistic function

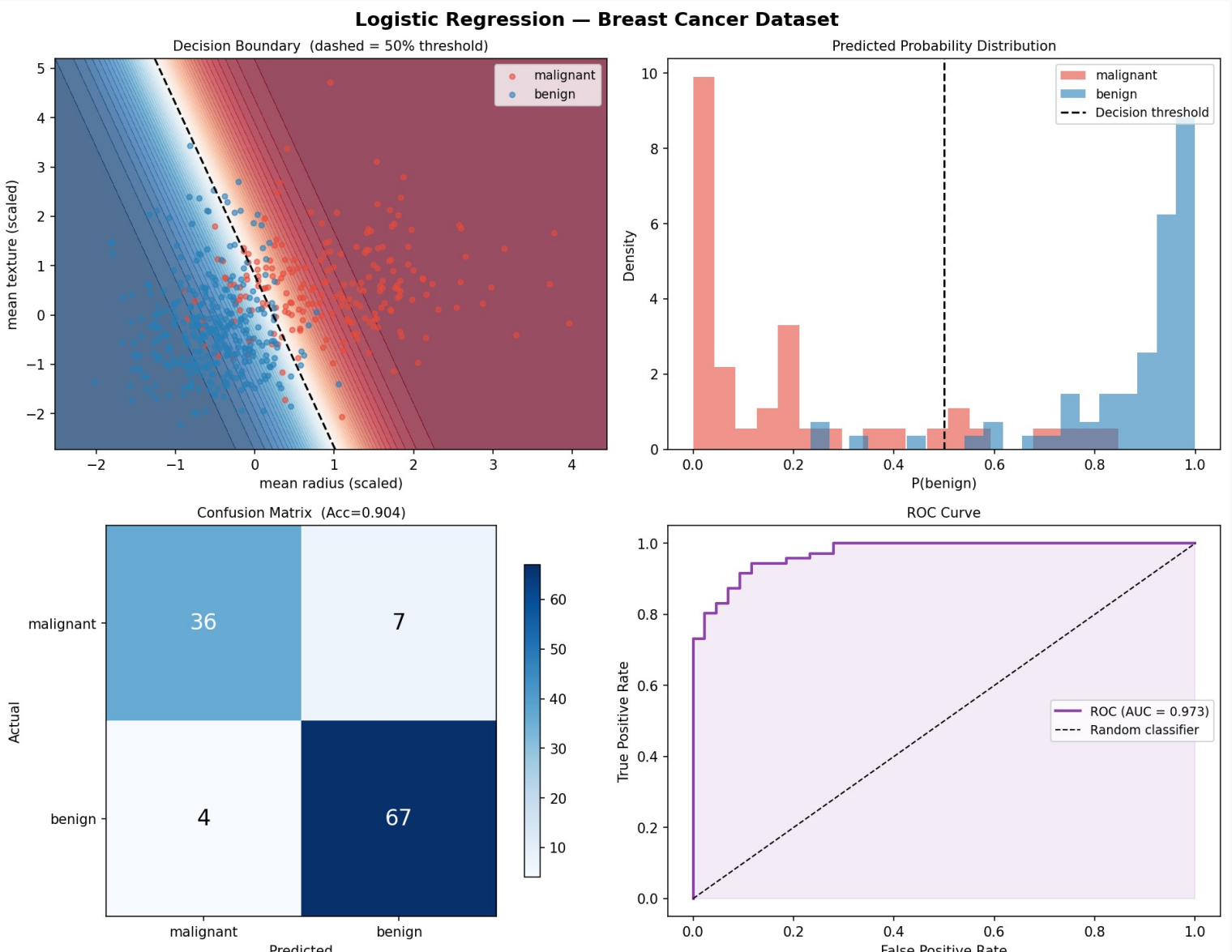
```
def predict(X, beta):
    m = np.matmul(X, beta)
    return logistic(m)
```

# Lingo

- Hyperparametres are variables that are not learned but defined by the user to the model or solver.
  - This happens when you create the sklearn model!
- Epoch: single pass-through all of the data.
- Iteration: single pass through a batch of data.

# Using Logistic regression

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=1000)

# To train the model
model.fit(X_train, y_train)

 ...
```


Logistic Regression — Breast Cancer Dataset
Decision Boundary (dashed = 50% threshold)
malignant
benign
mean texture (scaled)
mean radius (scaled)
Predicted Probability Distribution
malignant
benign
Decision threshold
Density
P(benign)
Confusion Matrix (Acc=0.904)
36
7
4
67
malignant
benign
Actual
Predicted
ROC Curve
ROC (AUC = 0.973)
Random classifier
True Positive Rate
False Positive Rate

# Confusion Matrix

- Remember back to our true positive and true negative box.

```
from sklearn.metrics import confusion_matrix

matrix = confusion_matrix(y_true, y_pred)
# matrix[i, j] indicates number samples with true label
# being i'th class and predicted label j'th class
```

# Receiver Operating Characteristic (ROC) Curve

- Way to show how FPR and TPR is affected as you change the cutt off to what is considered a positive result

```
from sklearn.metrics import roc_curve, roc_auc_score

fpr, tpr, thresholds = roc_curve(y_test, y_prob)
# List of false positive rate, true positive rate,
#  & varying thresholds
auc = roc_auc_score(y_test, y_prob)
# Gives area under roc curve (higher better)
```

# Hyperparametre Tuning

- Domain expert selection
- Manual Search (Trial and error)
- Grid search
- Random search
- Bayesian optimisation (advanced)

# Let’s practice

- Train a model to attempt to predict breast cancer diagnosis.
  - wdbc.data is a csv of data
  - wdbc.names gives information on the data
- Show performance on test set with confusion matrix and ROC curve with area under curve.
- **Bonus: can you use Streamlit to help show this!**