Data Science and Visualization (DSV, F23)

6. Regression

Hua Lu
https://luhua.ruc.dk; luhua@ruc.dk
PLIS, IMT, RUC

Agenda

- Regression Fundamentals
- Linear Regression
- Polynomial Regression
- Decision Tree Regression
- Logistic Regression

Classification vs. Regression

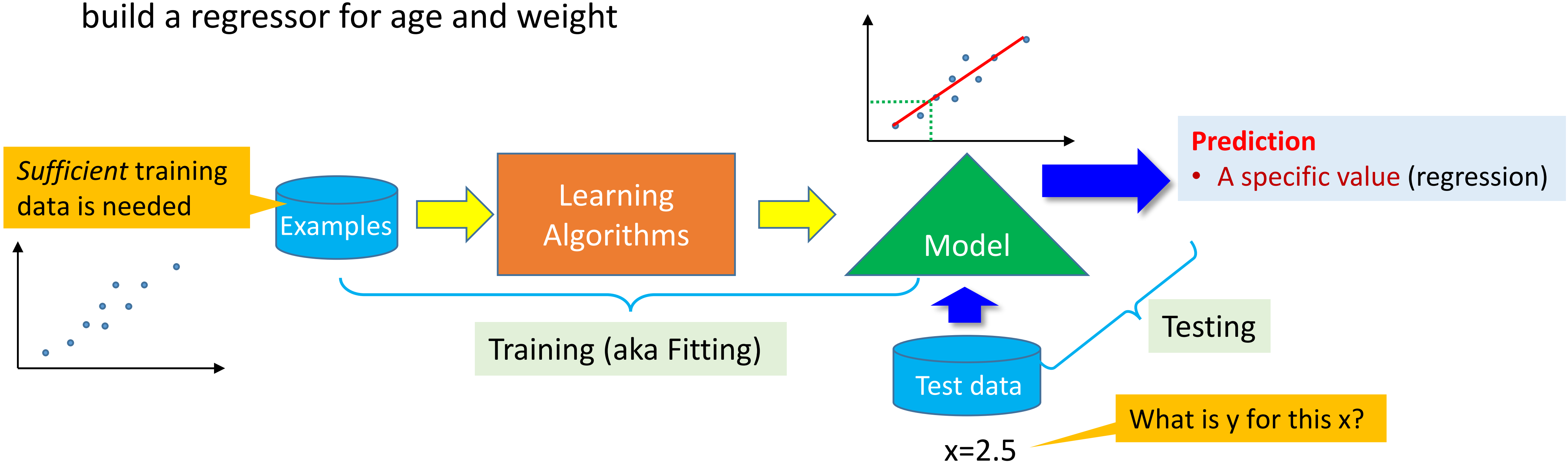
- **Classification**
  - Predict a discrete value (*class label*) from a pre-defined set (all classes)
    - E.g., given a loan applicant, predict if she/he is a *good* or *bad* client.
  - Models: Rule-based, Decision tree, Random forest, KNN, SVM, Bayes…
- **Regression**
  - To predict a value from a continuous range
    - E.g., to predict a stock's price.
  - We want to predict y for unseen X, based on the training data of known (X, y) pairs.
  - From the training data, we learn a function ***f***(.) *s.t.* ***f***(X) **approximates** the real y for each X.
    - Different types of ***f***, and different ways to learn it.
    - For an X in the training data, ***f***(X) may not be the same as the corresponding y!
  - For an unseen X, we predict its corresponding y value is ŷ = ***f***(X)
    - ŷ is the predicted y value for the given X

Regression can *also* be evaluated using CV!

Regression in General

- Supervised learning
- Training data is needed for regression
  - (x, y) pairs for 2D regression
  - E.g., (age, weight) value pairs if you want to build a regressor for age and weight

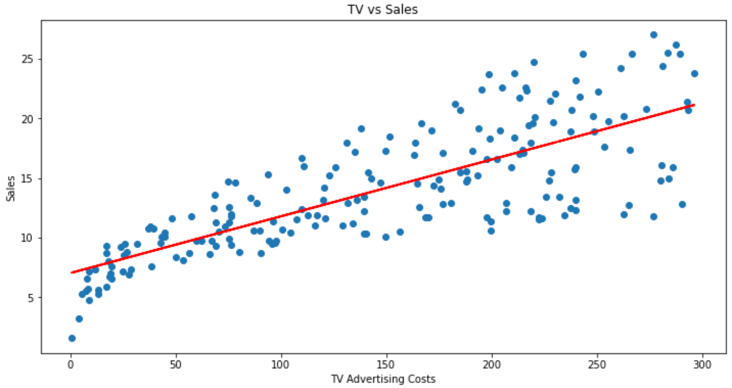
Notation:
- Ground truth: y
- Predicted value: ŷ

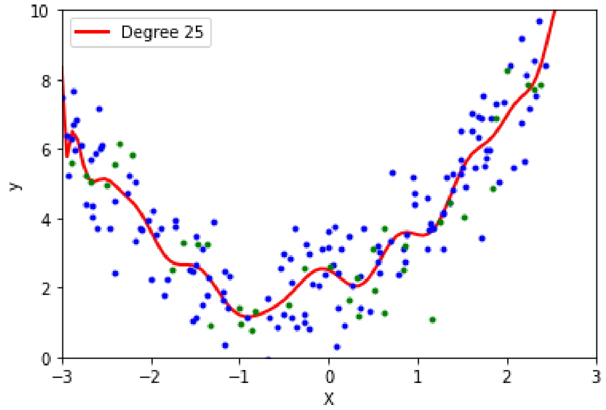


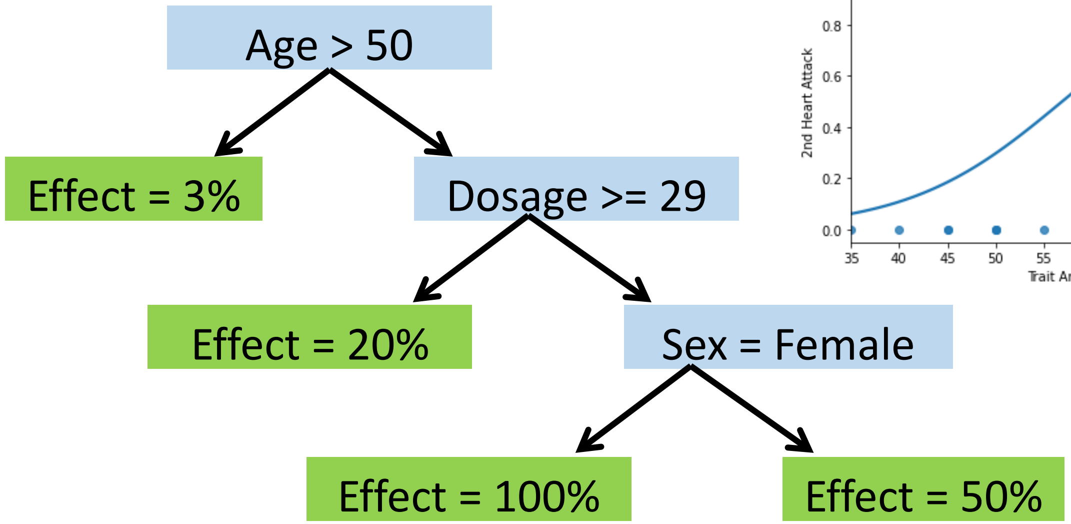
Regression Types

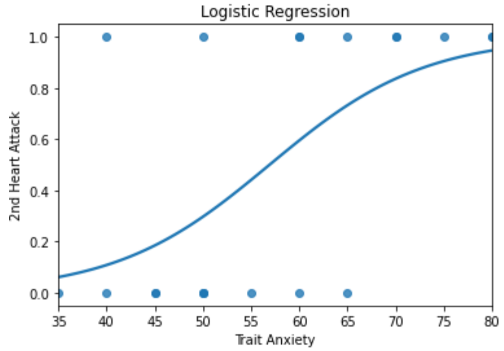
- Linear regression
- Polynomial regression
- Decision tree regression
- Logistic Regression
  - For classification!

- Python libraries
  - statsmodels
  - scikit-learn.linear_model
  - sklearn.preprocessing
  - sklearn.tree









Evaluating Regression Model $f(.)$

- Ground truth: y_i for X_i
- Predicted value: $\hat{y}_i = f(X_i)$

- **Mean Absolute Error** (MAE): The mean of the absolute value of the errors.
  - $\frac{1}{n}\Sigma|y_i - \hat{y}_i|$
- **Mean Squared Error** (MSE): The mean of the squared errors.
  - $\frac{1}{n}\Sigma(yi - \hat{y}_i)^2$
- **Root Mean Squared Error** (**RMSE**): The square root of the mean of the squared errors
  - $\sqrt{\frac{1}{n}\Sigma(yi - \hat{y}_i)^2}$
- Notes:
  - These metrics are absolute, not normalized w.r.t. the range of groundtruth y
  - You may compare different models' metrics on the same basis

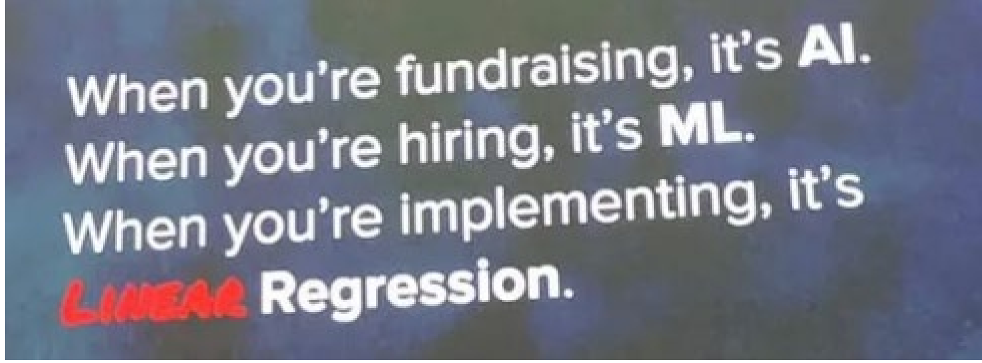
```
from sklearn import metrics
import numpy as np

metrics.mean_absolute_error(y_test, y_pred)
metrics.mean_squared_error(y_test, y_pred)
np.sqrt(metrics.mean_squared_error(y_test, y_pred))
```

Agenda

- Regression Fundamentals
- Linear Regression
- Polynomial Regression
- Decision Tree Regression
- Logistic Regression

Linear Regression

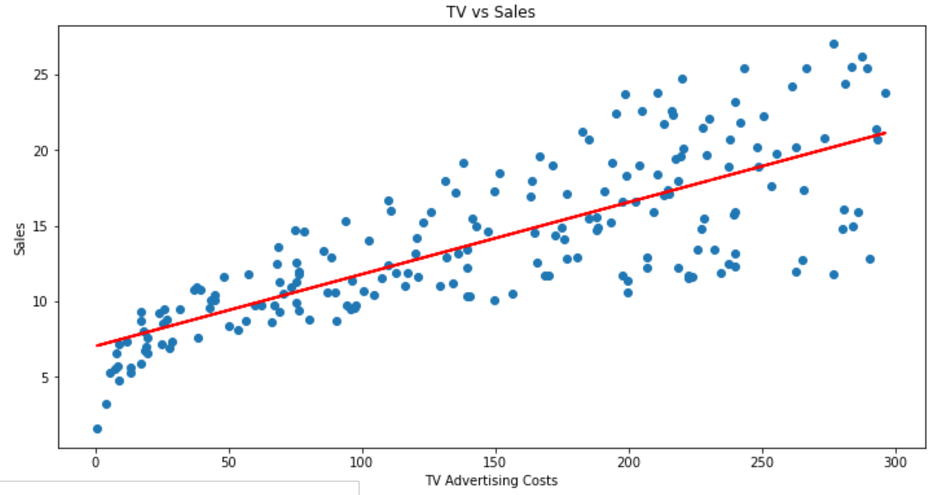


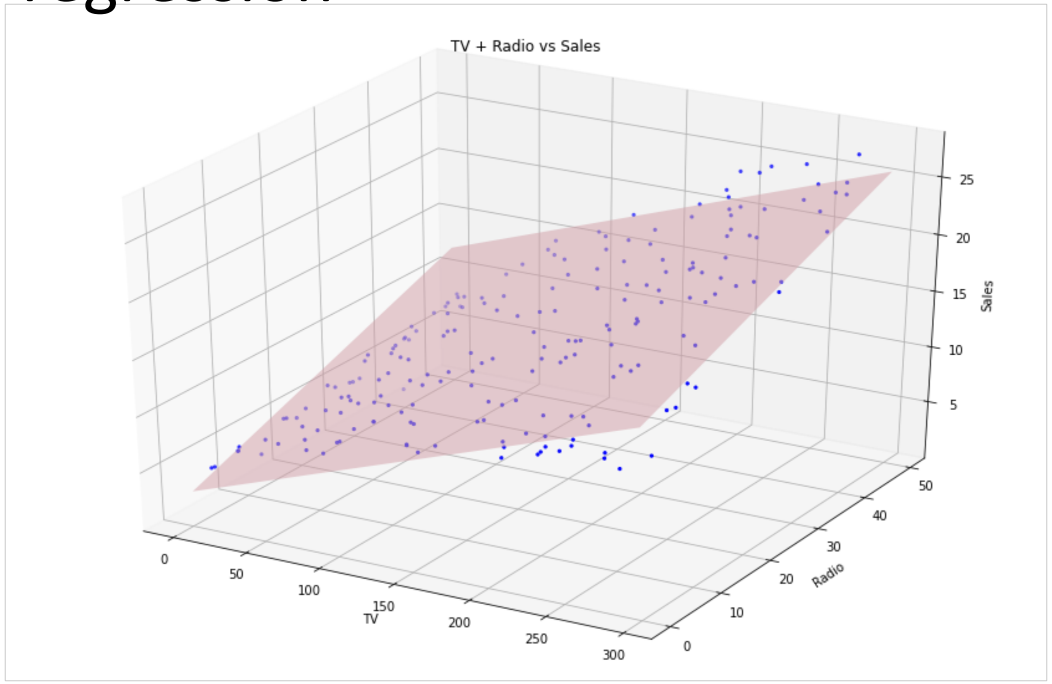
- Linear Regression is a basic yet popular predictive analytics technique that uses historical data to predict an output variable.
- **Assumption**: there exists a 'linear relationship' between input (independent) variables and their output (dependent) variables.
  - $\hat{y} = \boldsymbol{f}(X) = \alpha + \beta X$
  - X is the **input** or independent variable (scalar value or vector).
  - $\hat{y}$ is the **output variable** that we want to predict for a given X.
  - y is the groundtruth variable dependent on the X.
- A core step in Linear regression is to learn the coefficients α and β from training data, *s.t.* the difference between $\hat{y}$ and y is *minimized*.

Linear Regression Types

Linear function: $\hat{y} = \boldsymbol{f}(X) = \alpha + \beta X$

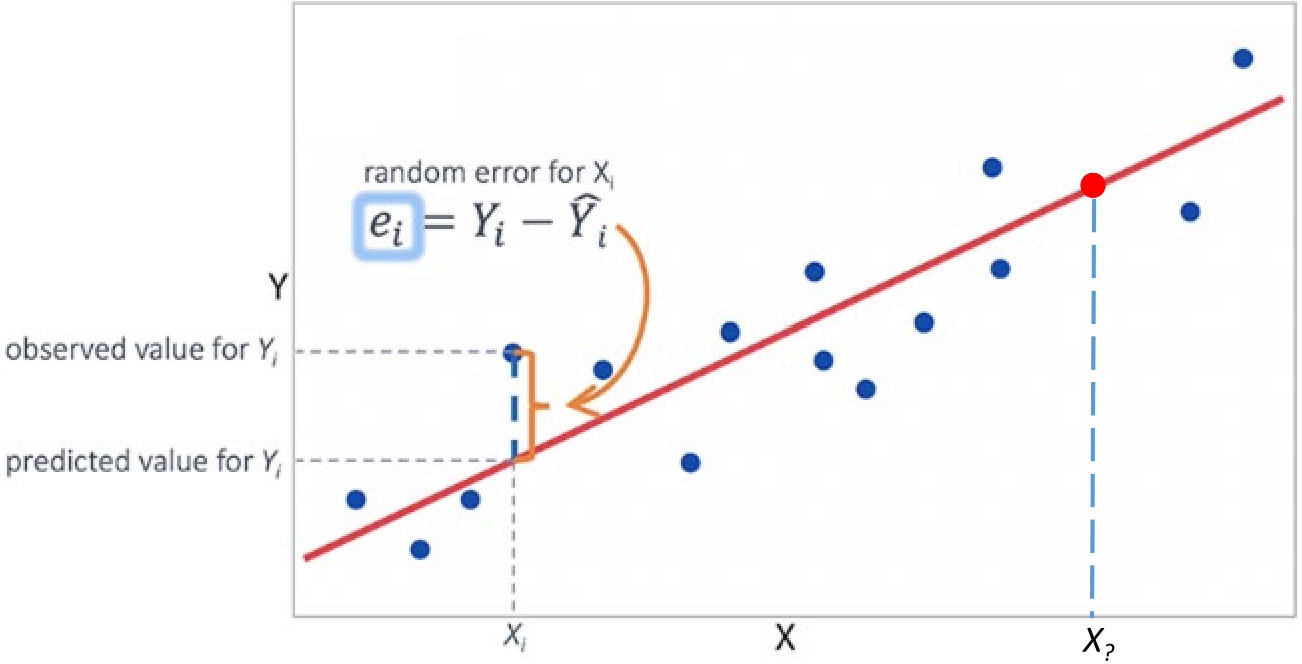
- Simple linear regression
  - X is a scaler value
- Multiple linear regression
  - X is a vector





Ordinary Least Squares (OLS)

- Linear function: $\hat{y} = \boldsymbol{f}(X) = \alpha + \beta X = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n$ (n>=1)
- OLS decides α and β according to **Principle of Least Squares**
  - To minimize **sum of squared residuals** (**SSR**): $\sum e^2 = \sum (y - \hat{y})^2$
  - I.e., the sum of the squares of the differences between the observed dependent variable (y) and the output variable (ŷ) predicted by the linear function of the independent variable (X).



- Training data object
- Learned linear function
- Prediction for the variable $X_?$

Adapted from http://shorturl.at/bvxyR

Example in Jupyter Notebook

- Advertising dataset
  - 200 data objects of 4 columns/attributes
  - Available in Moodle
- What is the relationship between advertising costs (on TV, Radio and Newspaper) and sales?
  - Simple linear regression
  - Multiple linear regression
- Lecture6_LR_advertising.ipynb
  - Statsmodels and scikit-learn libraries
  - Visualization of linear models

	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

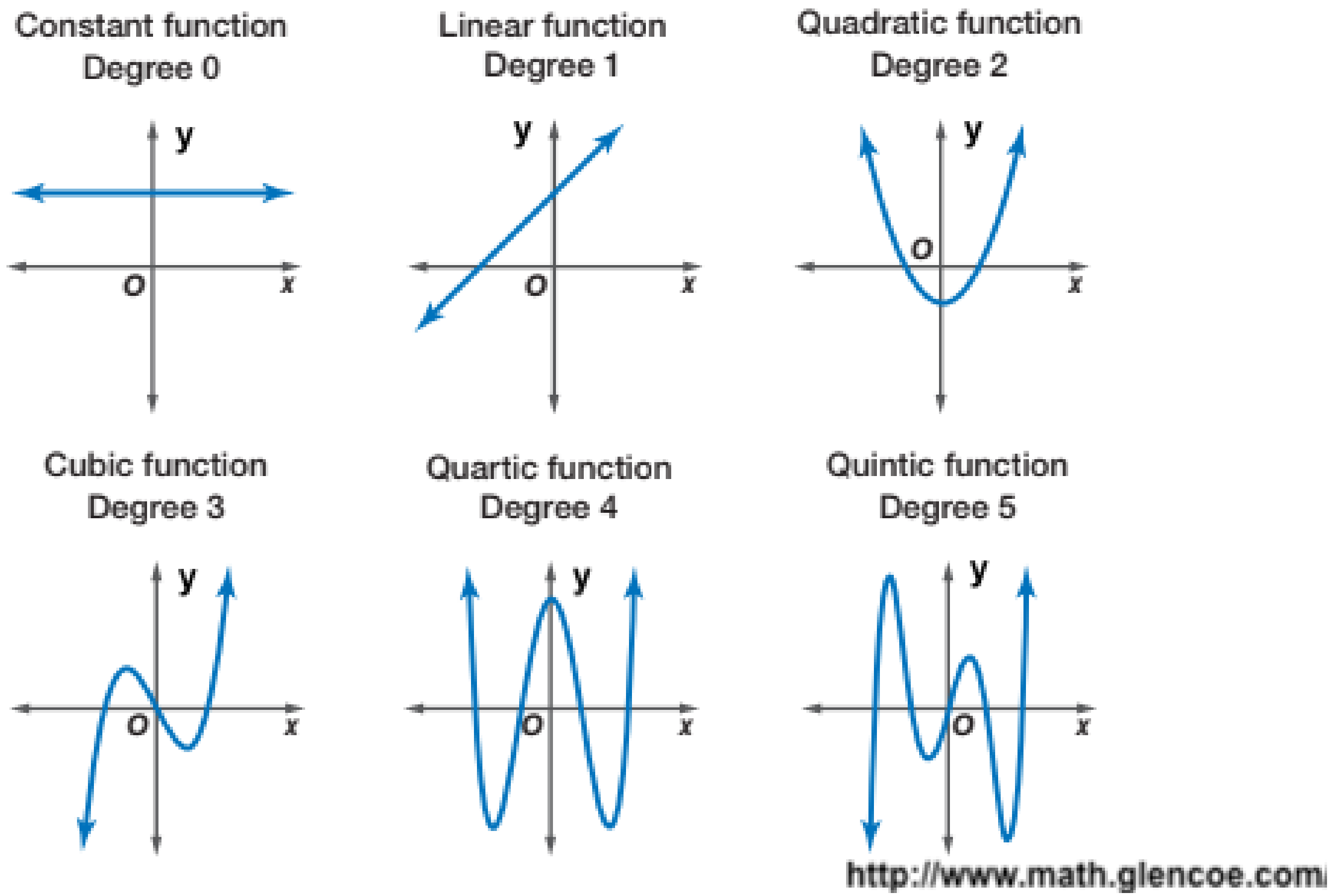
jupyter DEMO

Agenda

- Regression Fundamentals
- Linear Regression
- Polynomial Regression
- Decision Tree Regression
- Logistic Regression

Polynomial Regression Functions

- Polynomial function: $\hat{y} = \alpha + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots + \beta_n x^n$
  - Degree n
- A generalization of linear regression
- Find a curve that best fits a set of values
  - n-1 turning points



Back to Linear Regression

- Linear function: $\hat{y} = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n$
- Polynomial function: $\hat{y} = \alpha + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \ldots + \beta_n x^n$
- What can you see from these two equations?
- A polynomial function can be transformed into a Linear one!
  - $x_1 = x$, $x_2 = x^2$, $x_3 = x^3$, ..., $x_n = x^n$

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
lr_2 = LinearRegression()
lr_2.fit(X_poly, y)
y_poly = lr_2.predict(X_poly)
```

X_poly = $[x^0, x^1, x^2, x^3, x^4]$

$X = x$

Train a multiple linear regressor

Predict ŷ using the linear model

Example in Jupyter Notebook

- position_salaries dataset
  - 10 data objects of 3 columns/attributes
  - Available in Moodle
- What is the relationship between Level and Salary?
  - Try regressors with different degrees to fit the data
- Lecture6_PR_salaries.ipynb

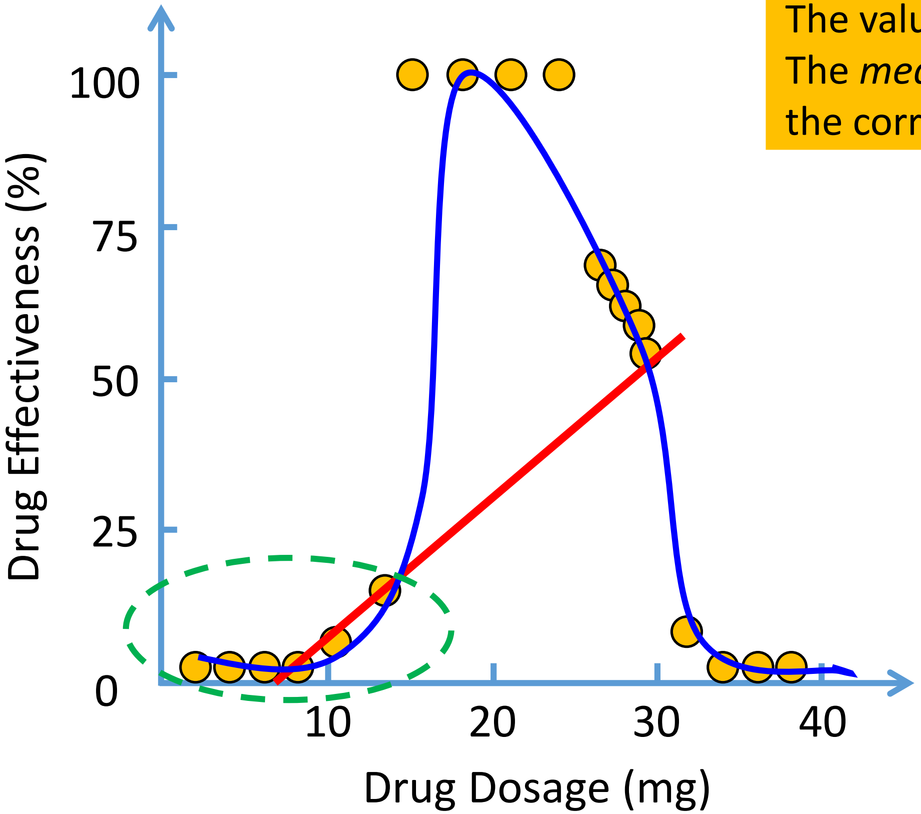
	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000
5	Region Manager	6	150000
6	Partner	7	200000
7	Senior Partner	8	300000
8	C-level	9	500000
9	CEO	10	1000000

Agenda

- Regression Fundamentals
- Linear Regression
- Polynomial Regression
- Decision Tree Regression
- Logistic Regression

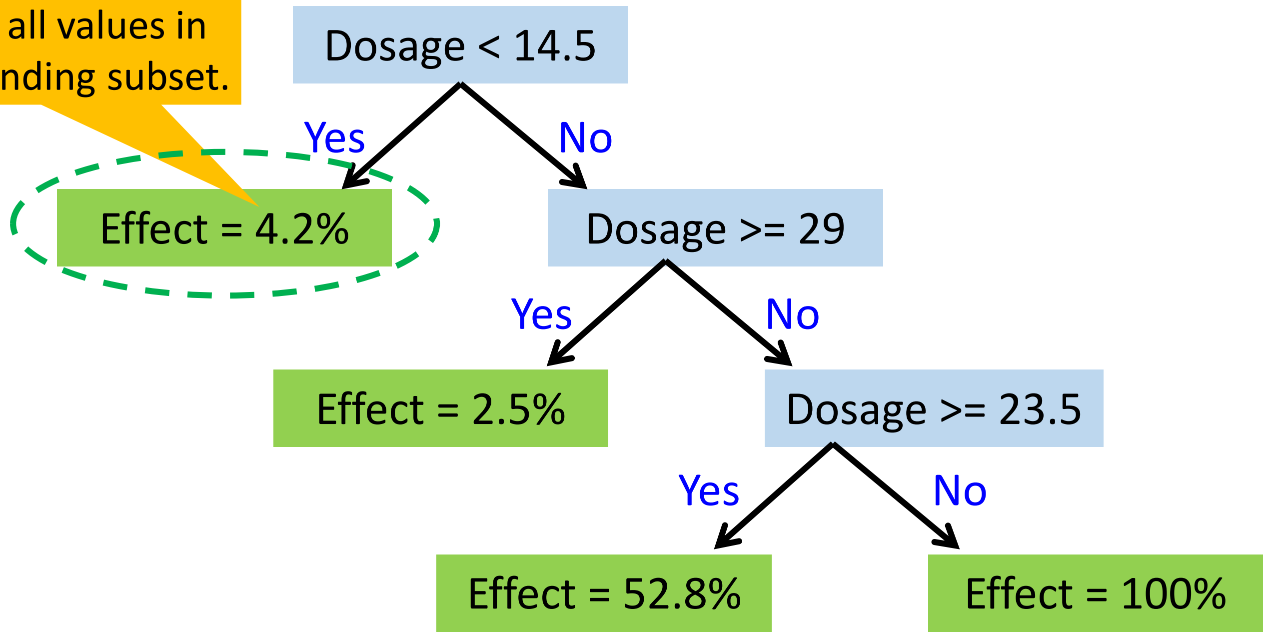
A Motivation Example

- Neither LR nor PR can work



Decisition Tree for Regression

The value in a leaf node: The *mean* of all values in the corresponding subset.

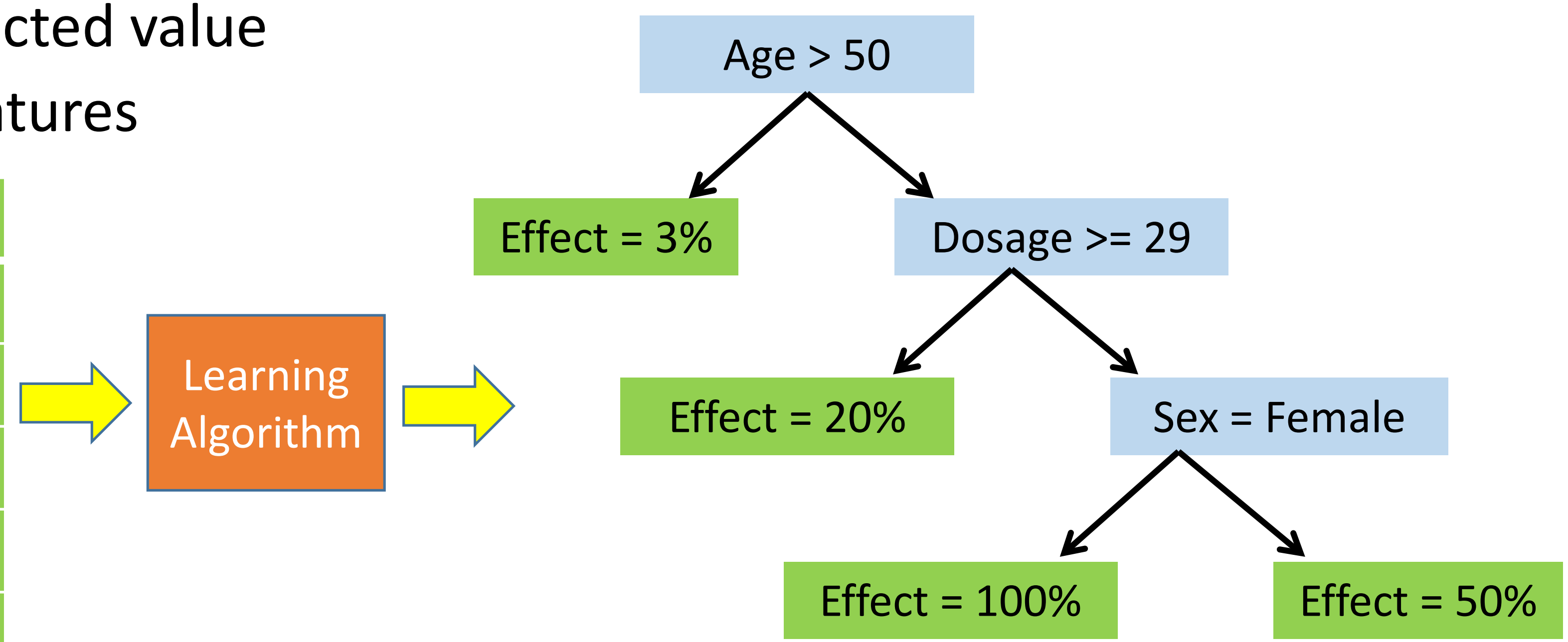


The leaf nodes do not contain class labels as a DT classifier does!

Decision Tree Regressor

- A special type of decision tree
- Each internal node asks an ‘Is’ question
  - More options can always be converted to binary
- Each leaf node gives a predicted value
- It also supports multiple features

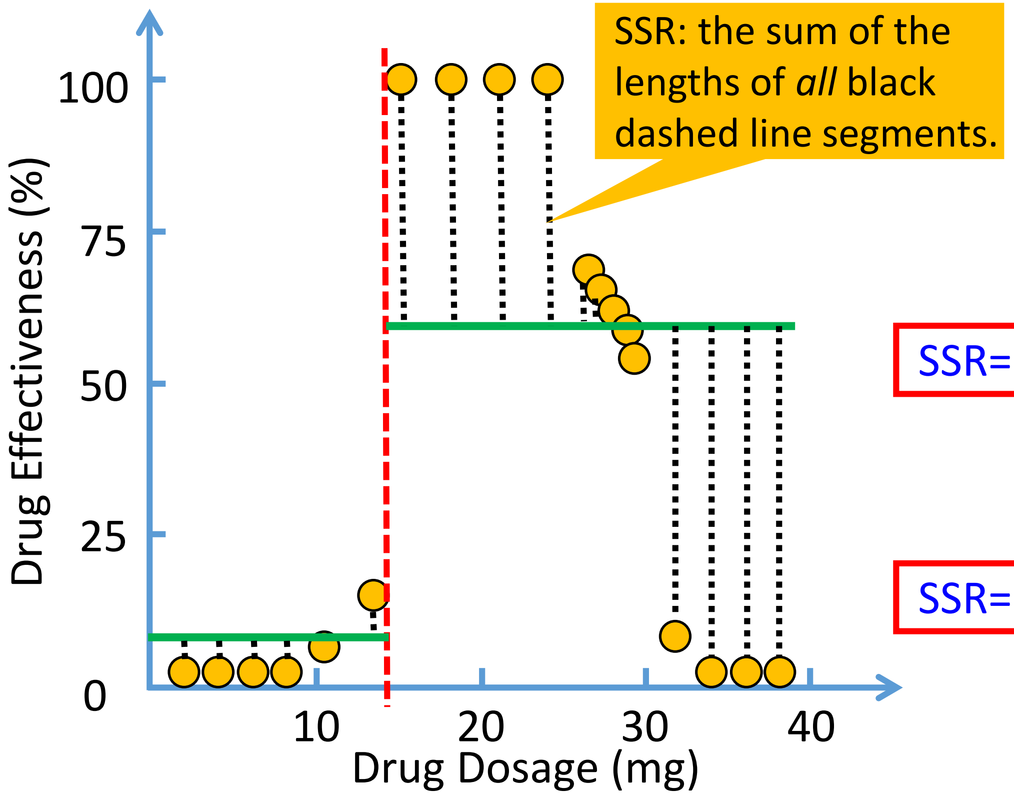
Dosage	Age	Sex	Effect
10	25	Female	98
20	73	Male	0
35	54	Female	6
5	12	Male	44
…	…	…	…

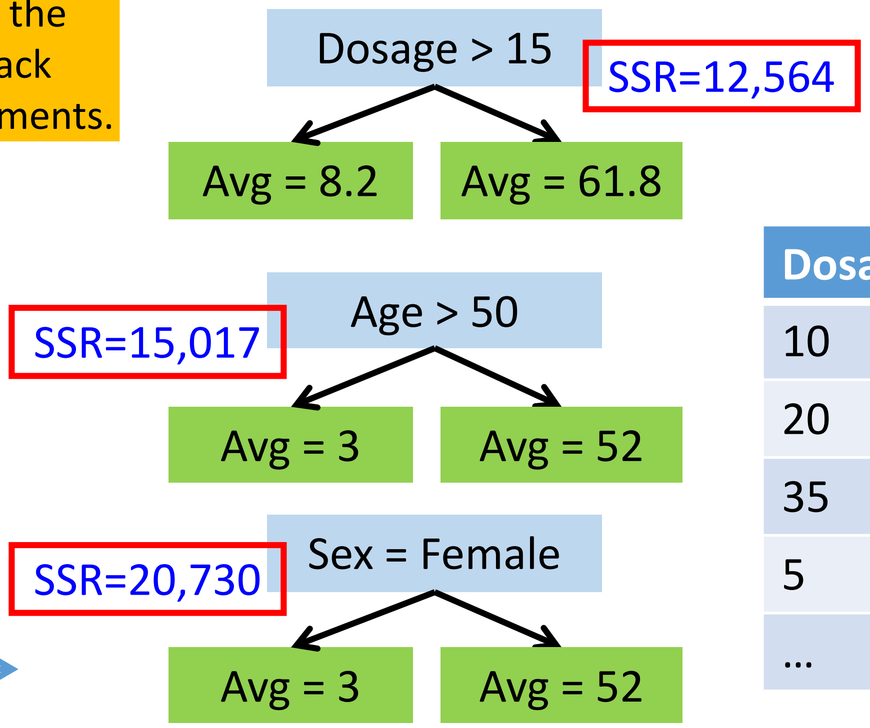


Decision Tree Regressor Generation

Advanced

1. On each column, find the best (binary) split that results in the smallest SSR.
2. Choose the column with the (globally) smallest SSR, and use its split as the root node.
3. Repeat 1 and 2 on each subset recursively, until no further split is needed or possible.





Dosage	Age	Sex	Effect
10	25	Female	98
20	73	Male	0
35	54	Female	6
5	12	Male	44
…	…	…	…

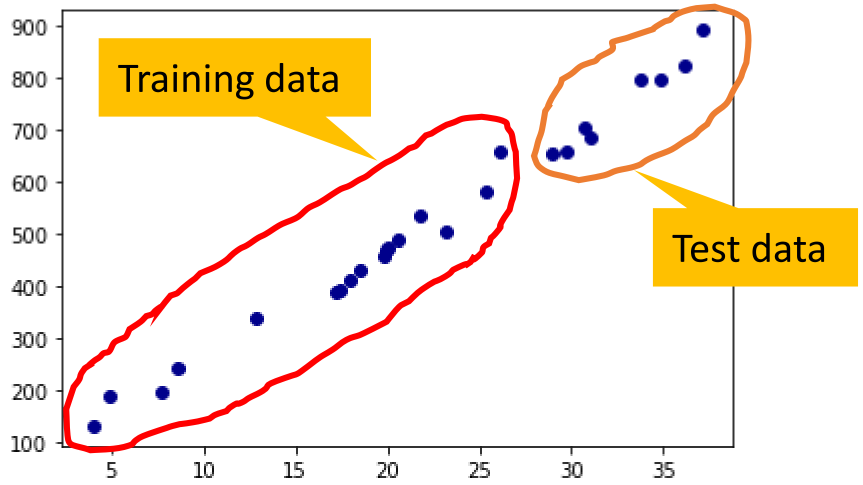
Example in Jupyter Notebook

- Boston Housing dataset
  - 506 data objects of 14 columns/attributes
  - We only focus on LSTAT and MEDV
  - Available in Moodle
- What is the relationship between LSTAT and MEDV?
  - Linear regression does not work well

- Lecture6_DTR_boston.ipynb

Notes on DT Regressors

- DT regressors in general are unable to *extrapolate* to any kind of data that they haven't seen before.
- DT regressors are used in regression problems *iff* the target variable is inside the range of values that have been seen in the training data.
- DT regressors are prone to overfitting.

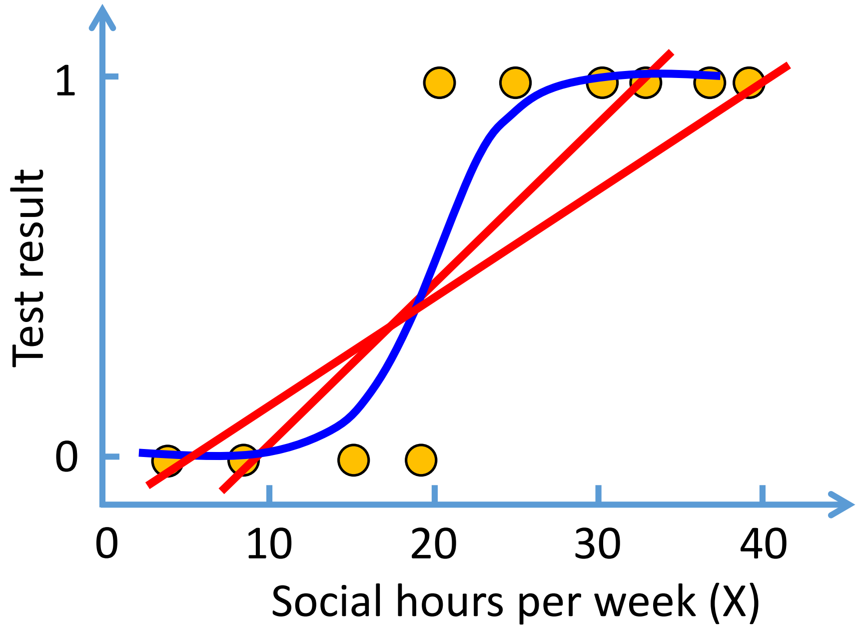


Agenda

- Regression Fundamentals
- Linear Regression
- Polynomial Regression
- Decision Tree Regression
- Logistic Regression

A Motivation Example

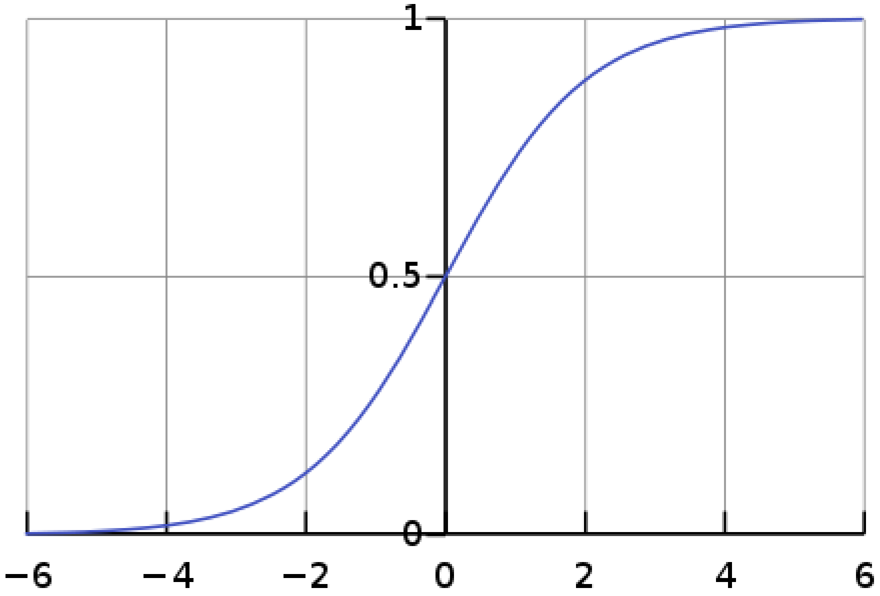
- A general case: To predict a binary, categorical dependent variable
  - 1 (yes, success, positive, …) or 0 (no, failure, negative, ...)
- A fictional dataset about social hours per week and COVID-19 test result
  - A linear regressor fails
  - So does a polynomial regressor
  - A decision tree regressor might work, but not when the two subsets of points overlap or cross each other
- A S-shape curve may fit such data



Sigmoid Function

- A sigmoid function is characterized by a S-shaped curve or sigmoid curve.
- A common example of a sigmoid function is the logistic function:
  - $S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} = 1 - S(-x)$
- S(x) returns y values in the range of 0 and 1
  - S(-∞) = 0; S(+∞) = 1
  - Probabilities!
- S(x) can predict probabilities continuously, but for convenience we stipulate
  - 1 if S(x) >= 0.5
  - 0 if S(x) < 0.5

Regression -> Classification!

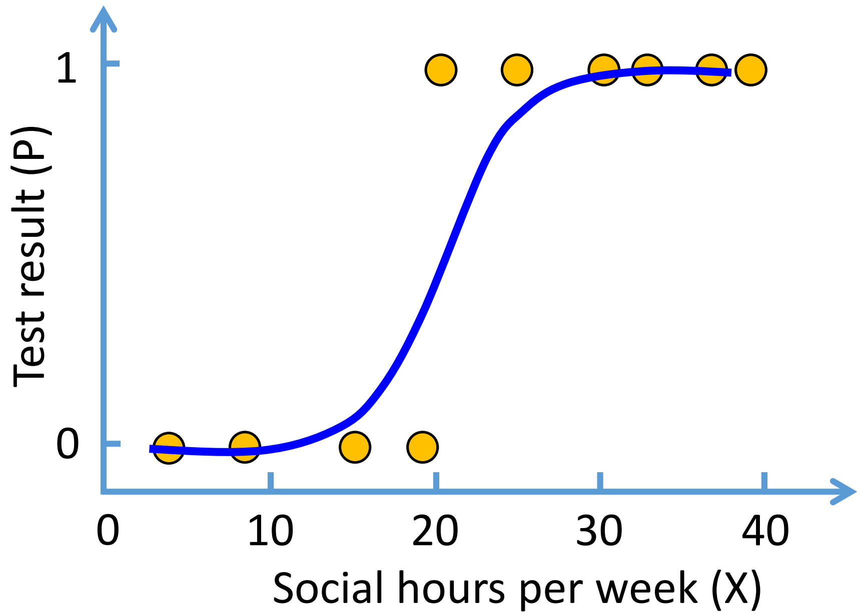


Logit and Logistic Regression

Advanced

- A distribution of 0s and 1s
  - The mean of the distribution is equal to the *proportion* of 1s in the distribution.
    - It is also the *probability* *P* of drawing a label of 1 at random from the distribution.
    - The proportion (and probability) of 0s is (1 - *P*).
  - The odds of being 1 is $odds = \frac{P}{1-P}$
- In logistic regression, the dependent variable y is a *logit*, the natural log of the odds:
  - $\log(odds) = logit(P) = \ln(\frac{P}{1-P})$
  - We find logit(P) = a + bX, *i.e.* the log(odds) or logit is assumed to be linearly related to X
  - $\ln\left(\frac{P}{1-P}\right) = a + bX, \frac{P}{1-P} = e^{a+bX}$
  - $\boldsymbol{P = \frac{1}{1+e^{-(a+bX)}}}$

Sigmoid/logistic function



Example in Jupyter Notebook

- Heart attack patient dataset
  - 20 data objects of 3 columns/attributes
  - Available in Moodle
- Will a patient have the 2^{nd} heart attack?

	2nd_Heart_Attack	Treatment_of_Anger	Trait_Anxiety
0	1	1	70
1	1	1	80
2	1	1	50
3	1	0	60
4	1	0	40

- Lecture6_Logit_patients.ipynb

DEMO

Notes on Logistic Regression

- Given a set of independent (continuous or categorical) variables, Logistic Regression predicts binary, categorical (not continuous) dependent variables.
- The goal of logistic regression is to estimate the *probability of occurrence of a value*, not the value of the variable itself.
  - It's for classification, not for regression ☺
- The range of values for the prediction is restricted to the range between 0 and 1.
  - This is ensured by using the logistic function.

- A generalization of the linear regression model

Summary

- Regression
  - General form
  - Evaluation
- Linear regression
  - Simple vs. multiple
- Polynomial regression
  - Conversion to linear regression
- Decision tree regression
  - Special decision tree structure
- Logistic Regression
  - For classification
  - Relation to linear regression

https://www.reddit.com/r/ProgrammerHumor/

References

- Mandatory reading
  - Andreas C. Muller and Sarah Guido: Introduction to Machine Learning with Python, O'Reilly, 2016
    - Chapter 2: Regression, Linear Models
    - Chapter 4: Interactions and Polynomials
- Further readings
  - Linear regression
    - Tutorial: https://towardsdatascience.com/introduction-to-linear-regression-in-python-c12a072bedf0
  - Decision tree regression
    - https://gdcoder.com/decision-tree-regressor-explained-in-depth/
    - https://www.youtube.com/watch?v=g9c66TUylZ4
  - Logistic regression
    - https://www.youtube.com/watch?v=OCwZyYH14uw
    - https://www.youtube.com/watch?v=0m-rs2M7K-Y
    - http://faculty.cas.usf.edu/mbrannick/regression/Logistic.html (Advanced)

Exercises (1)

1. Using the Diamonds dataset (available in Moodle), do the following in Jupyter Notebook
    1. Select columns **carat**, **depth** and **table** as independent variables. Use them to predict column **price**.
    2. Split the data into training and test sets.
    3. Use the training set to construct a *linear regressor*.
    4. Use the same training set to construct a *decision tree regressor*.
    5. Apply the two regressors on the test set, and show their **MAE**, **MSE** and **RMSE**.
    6. Visualize the two regressors.
    7. Decide which regressor is better based on 5 and 6.

Exercises (2)

2. Using the Boston dataset (available in Moodle), do the following in Jupyter Notebook
    1. Select LSTAT (independent variable) and MEDV (dependent variable)
    2. Split the data into training and test sets.
    3. Use the same training set to construct a few *polynomial regressors* with degree of 2 to 8 respectively.
    4. Apply all these regressors on the test set, and show their **MAE**, **MSE** and **RMSE**.
    5. Decide which regressor is the best.
    6. Visualize the best polynomial regressor.

Exercises (3)

3. Using the **cleaned Titanic** dataset (available in Moodle), do the following in Jupyter Notebook
    1. Split the data into training and test sets
    2. Use the **Pclass, Sex**, and **Age** columns as the independent variable, build a logistic regressor to predict the **Survived** column, and validate the model