Exploratory Data Analysis and Visualization

5. Supervised learning

Li Xinke

DS

City University of Hong Kong

Supervised learning

Statistical machine learning

- Machine learning constructs **algorithms** that can learn from data , especially for **prediction**

- Statistical learning is a branch of Statistics that was developed in response to Machine learning, emphasizing building statistical models, drawing **inferences** and assessing uncertainty

- Data Science is the extraction of knowledge from data, using ideas from mathematics, statistics, machine learning, computer programming, data engineering . . .

Structured vs. unstructured data

- Structured data: a flat file with a fixed number of measurements, e.g., patient response to a drug under consideration of certain conditions (such as age, weight, size, nutrition intake)

- Unstructured data: doctor's notes, Twitter feeds, broker reports

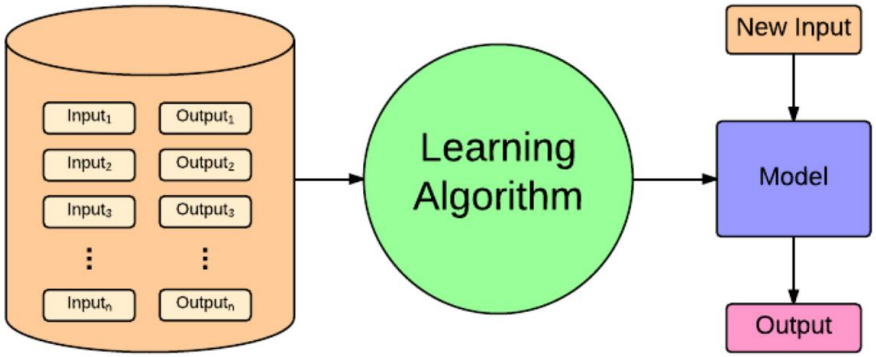
- We will focus on structured data.

21st century statistical learning examples

- Use **classification techniques** to classify which accounts are the most likely to upgrade their service contract (this helps the salesforce to know which accounts to focus on to sell more). . .
- Use **regression techniques** to improve how sales attract prospective clients and what features of the company's services they should highlight. . .
- **Regression and classification** are examples of **Supervised Learning** techniques (see next slide)

- Use **optimization techniques** to maximize the number of views of company promotional material a prospective customer sees for a given dollar amount of promotional spend

Supervised learning paradigm

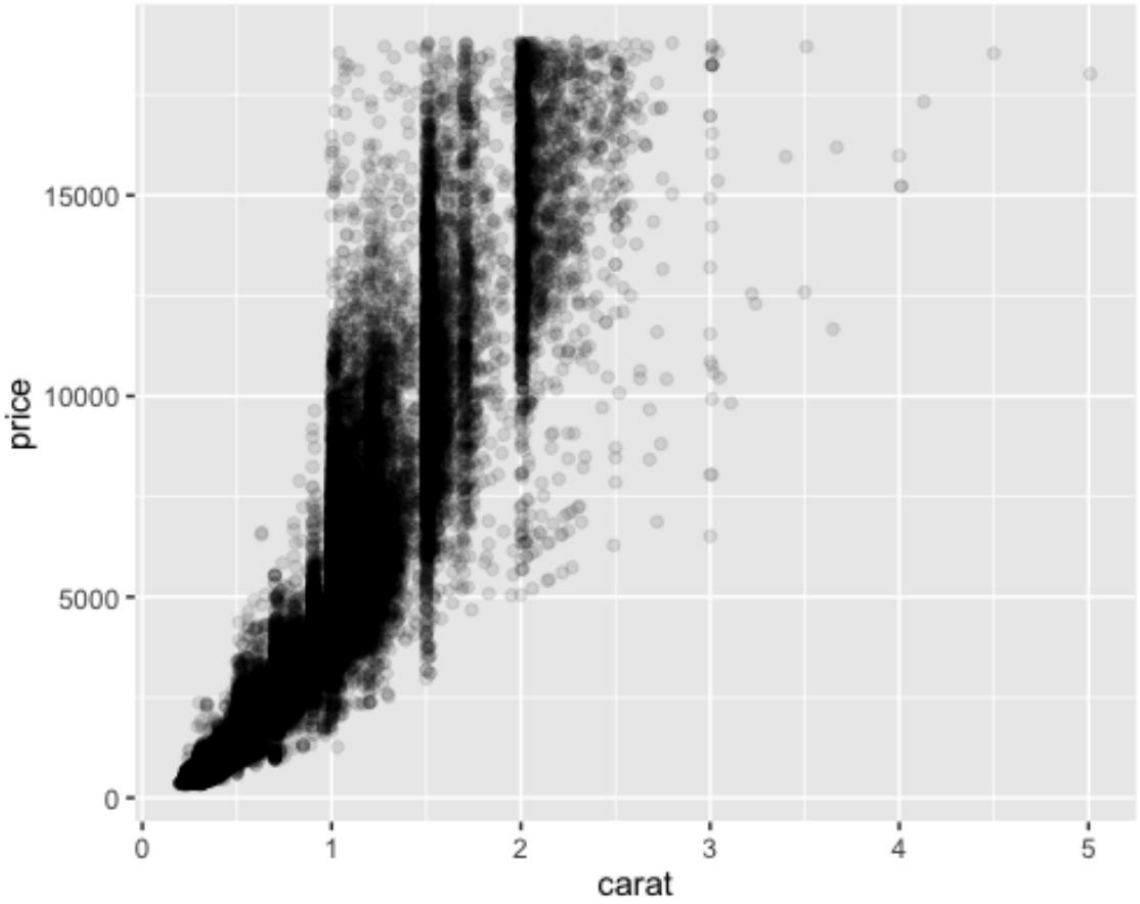


Traditional statistics: domain experts work for many years to learn good **features**; they bring statisticians a small clean dataset

Today's scenario: Domain knowledge is limited in new fields and large data sets are readily available. we (are sometimes forced to) start with **a large dataset** with many features

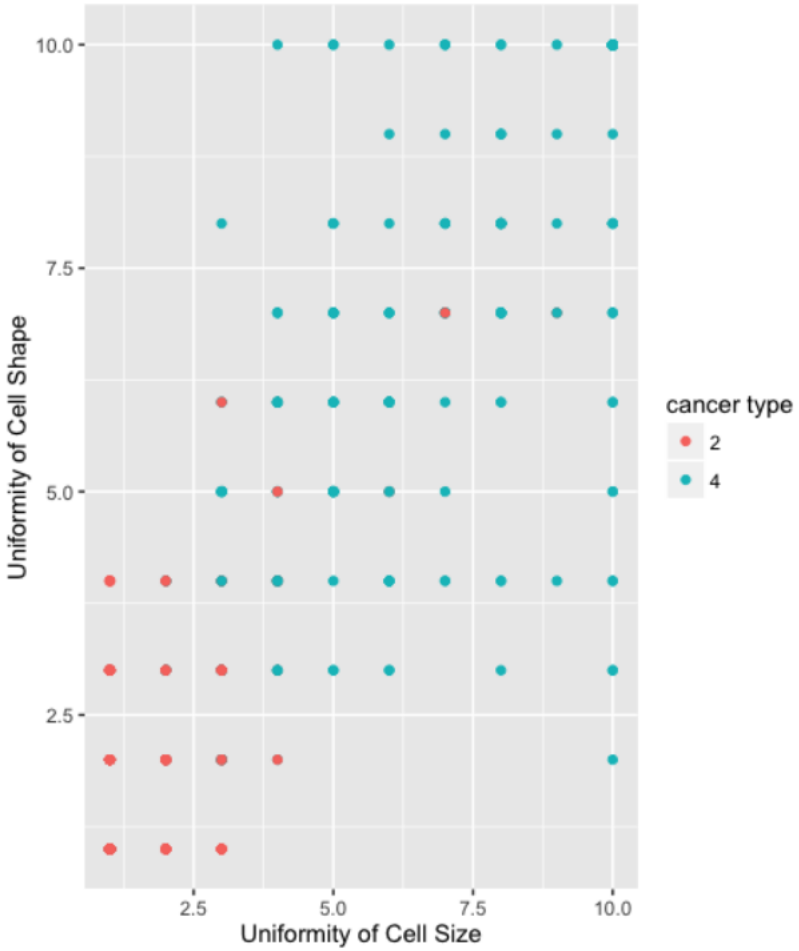
First supervised learning example: diamond price prediction

- Task: predict diamond price based on weight (regression)



Second example: cancer diagnosis (benign, malignant)

- This breast cancer dataset was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. (2 codes benign and 4 codes malignant). A classification example



More examples: Netflix challenge

- October 2006: Netflix offers $1M for an improved recommender algorithm.

https://en.wikipedia.org/wiki/Netflix_Prize

- **Training data**:

  100M ratings
  480K users
  17,770 movies
  6 years of data: 2000-2005

- **Test data**:

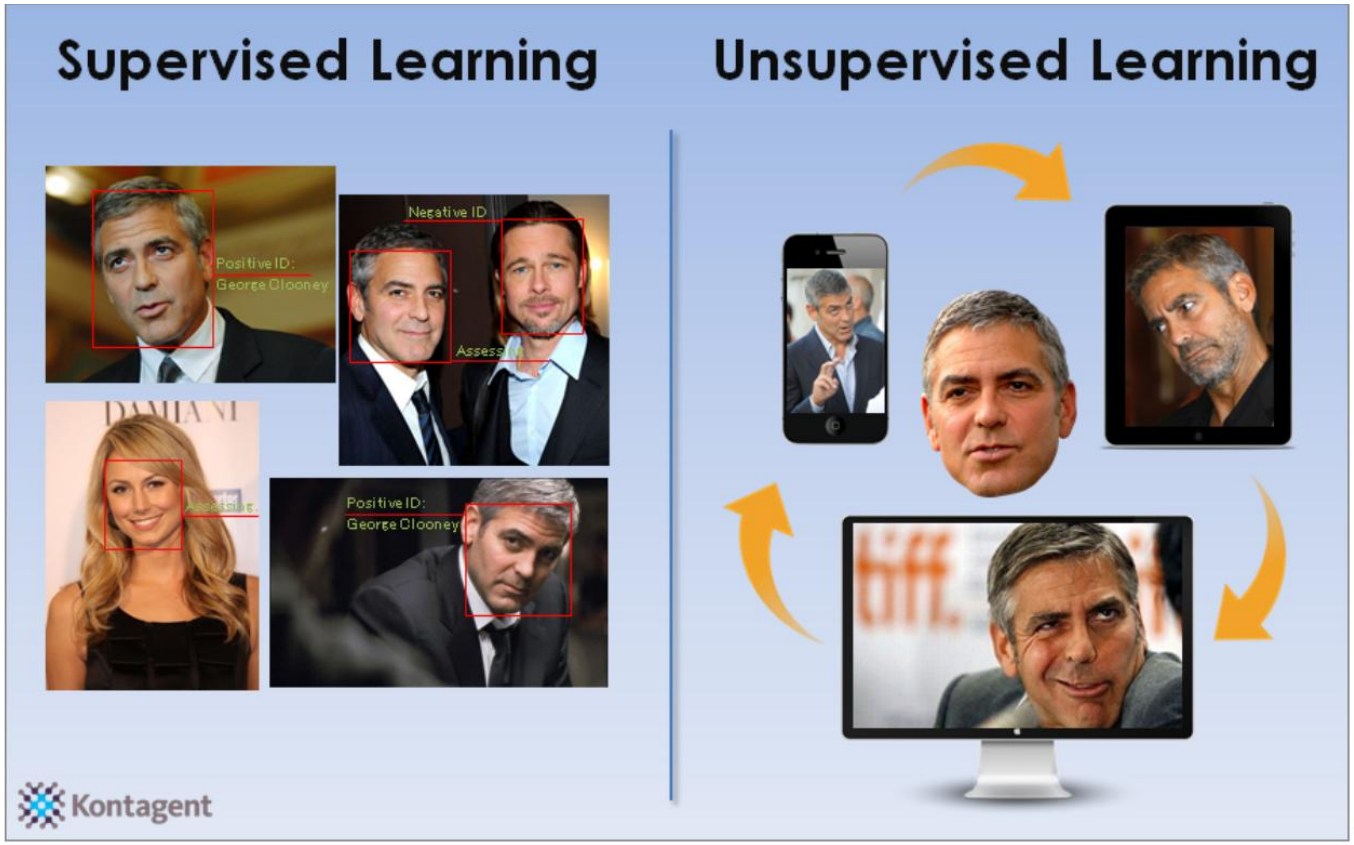
  Last few ratings of each user (2.8M)
  Evaluation via RMSE: root mean squared error
  Netflix Cinematch RMSE: 0.9514

- Competition:

  $1M grand prize for 10% improvement
  If 10% not met, $50K annual “Progress Prize” for best improvement

Supervised vs Unsupervised Learning

- **Supervised**: Both inputs (features, a.k.a. covariates, a.k.a. independent variables) and outputs (labels, a.k.a. response, a.k.a. dependent variable) available in training set.



Supervised Learning

- Outcome measurement Y
  - dependent variable, response, target
- Vector of p predictor measurements X
  - inputs, regressors, covariates, features, independent variables
- In regression problem, Y is quantitative
  - price, blood pressure
- In classification problem, Y takes values in a finite, unordered set
  - survived/died, digit 0-9, cancer class of tissue sample
- We have training data $(x_1, y_1), \ldots, (x_N, y_N)$.
- These are observations (examples, instances) of these measurements.

Unsupervised Learning

- No outcome variable, just a set of predictors (features) measured on a set of samples.
- Difficult to know how well you are doing.
- Different from supervised learning but can be useful as a pre-processing step for supervised learning.

An exercise

Are the following problems supervised or unsupervised problems?

If they are supervised problems, are they regression (output is numerical) or classification (output is categorical) problems?

- Problem 1: You have a large inventory of identical items. You want to predict how many of these items will sell over the next 3 months.
- Problem 2: You'd like software to examine individual customer accounts, and for each account decide if it has been hacked/compromised.
- Problem 3: Given a database of customer data, automatically discover market segments and group customers into different market segments.

An old example

Galton's Early Considerations of Regression

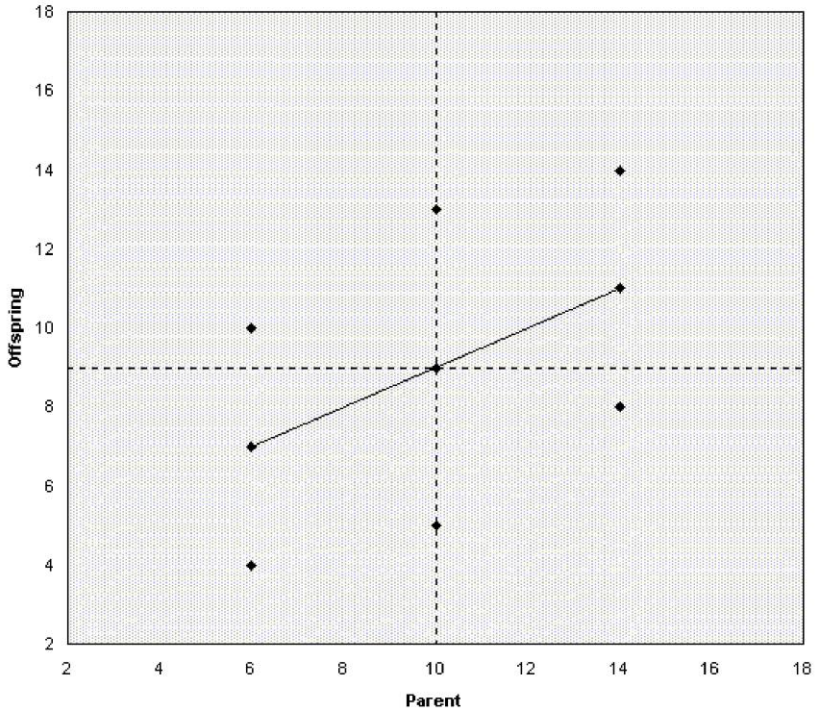
In 1875, Galton had distributed packets of sweet pea seeds to seven friends; each friend received seeds of uniform weight but there was substantial variation across different packets. Galton's friends harvested seeds from the new generations of plants and returned them to him.

- Galton plotted the weights of the daughter seeds against the weights of the mother seeds.

- Galton realized that the mean weights of daughter seeds from a particular size of mother seed approximately described a straight line with positive slope less than 1.0.

An old example

Galton's Early Considerations of Regression



r = 0.51

The y-coordinates of most of the points are closer to the horizontal offspring mean than their x-coordinates.

“regression toward the mean” -- Galton

Linear regression with a single variable

- Nowadays, the package **scikit-learn** is a widely used Python library for machine learning, built on top of NumPy and some other packages. （recall miss data imputing)
- There are five basic steps when you are implementing linear regression:
  1. Import the packages and classes you need.
  2. Provide data to work with and do appropriate transformations if necessary.
  3. Create a regression model and fit it with existing data.
  4. Check the results of model fitting to know whether the model is satisfactory.
  5. Apply the model for predictions.

Import the packages and read in data

Dataset: galton.csv. father height v.s. son height.

```
: import numpy as np
  from sklearn.linear_model import LinearRegression
```

Figure 1: import NumPy and LinearRegression

```
import pandas as pd
import matplotlib.pyplot as plt
```

```
height = pd.read_csv("data/galton.csv");height.head()
```

	Unnamed: 0	child	parent
0	1	66.435917	70.851069
1	2	65.943364	69.858889
2	3	64.278858	65.278141
3	4	63.851914	64.032631
4	5	63.192294	63.418992

Figure 2: Do you see any undesirable elements in this DataFrame?

Set colume 0 as index column with index_col = 0

```
height = pd.read_csv("data/galton.csv", index_col=0); display(height.head())
#can also try print(height.head())
```

	child	parent
1	66.435917	70.851069
2	65.943364	69.858889
3	64.278858	65.278141
4	63.851914	64.032631
5	63.192294	63.418992

```
height.info() # the .info() give you some information about the DataFrame height
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 928 entries, 1 to 928
Data columns (total 2 columns):
child     928 non-null float64
parent    928 non-null float64
dtypes: float64(2)
memory usage: 21.8 KB
```

Some summary statistics about the variables in the dataframe

```
height.describe() # columnwise summary statistics
```

	child	parent
count	928.000000	928.000000
mean	68.086288	68.299524
std	1.527244	1.857460
min	63.192294	63.418992
25%	67.027340	67.233314
50%	68.115752	68.336807
75%	69.077425	69.485036
max	72.185332	73.355968

Visual inspection

```
plt.figure(); plt.scatter(height['parent'], height['child']); plt.show()
```

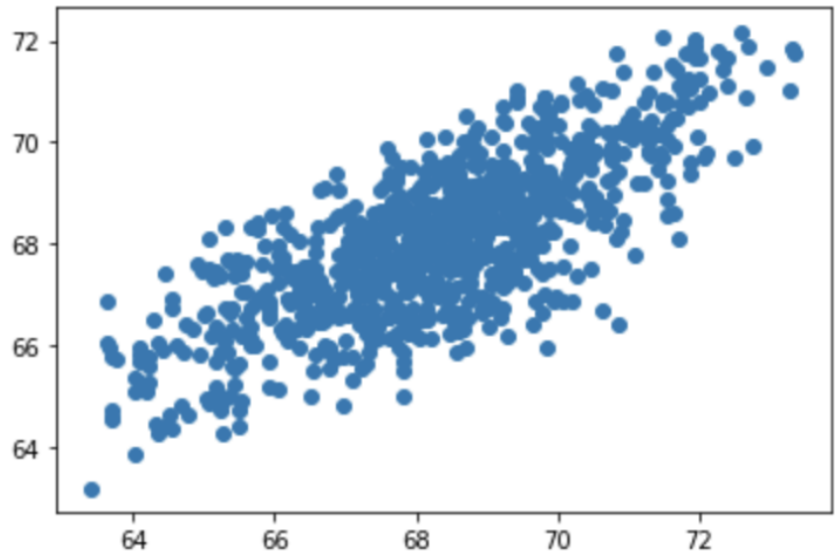


Figure 4: Scatter plot

- What information can be told by looking at this scatter plot?

The most naive prediction

- Take parent's height as the predicted value for child's height: $\hat{y} = x$

```
plt.figure(); plt.scatter(height['parent'], height['child'])
plt.xlabel('parent height'); plt.ylabel('child height'); plt.title('child vs. parent heights')
lineStart = height['parent'].min(); lineEnd = height['parent'].max()
plt.plot([lineStart, lineEnd], [lineStart, lineEnd], color = 'r'); plt.show()
```

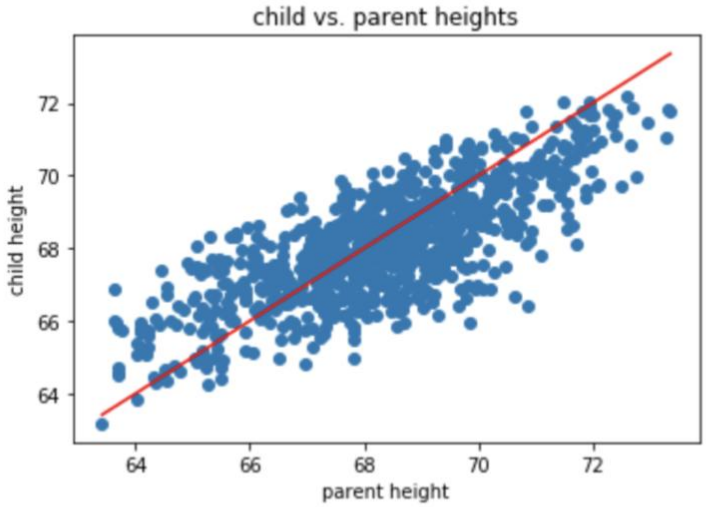


Figure 5: The red line is the 45 degree angle line

- Does the prediction look good? By what standard?

Linear regression model

- The simplest of all is a (simple) *linear* regression model; that is, the *response* or *target y* satisfies

$$y = \beta_0 + \beta_1 x + \varepsilon$$

in which

- β_0 is the *intercept* term
- β_1 is the *slope*
- β_0 and β_1 are *coefficients* or *parameters* of the linear model
- ε is a noise term, which is assumed to have **mean zero**.
- It is often assumed to be **normally distributed** in theoretical analysis.
- Often, we assume the standard deviation of ε, denoted by σ, is unknown. σ is another parameter of the simple linear regression model.

Linear regression (child heights v.s. parent heights)

$$\begin{aligned}\hat{y} &= \hat{\beta}_0 + \hat{\beta}_1(x) \\ &= \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_1\left(x - \bar{x}\right) \\ &= \bar{y} + \hat{\beta}_1\left(x - \bar{x}\right)\end{aligned}$$

Why did we decompose the equation this way?

Let me first tell you that the slope estimate $\hat{\beta}_1 \in (0, 1)$ in our example

- $\hat{\beta}_0 + \hat{\beta}_1 \bar{x}$ is average child height
- $\hat{\beta}_1 \cdot (x - \bar{x})$ is *regression to mean*, so taller parents' children shrink toward average

Fitting the regression model

- **Data**: Have n pairs (x_i, y_i), where x_i is the height of parent i and y_i is height of child i

- **Predictions**: $\hat{y}_i$ is our model's prediction of the height of child i

- **Loss function**: First, define a way to measure *error* or *residual* $e_i = y_i - \hat{y}_i$

$$Loss(y, \hat{y}) = (y - \hat{y})^2$$

Fitting the regression model

- **Data**: Have a lot of pairs (x_i, y_i), where x_i is the height of parent i and y_i is height of child i, and $i = 1, \cdots, n$
- **Predictions**: $\hat{y}_i = \beta_0 + \beta_1 x_i$ is model prediction of child height i
- **Loss on data**: Estimate parameters β_0 and β_1 by solving least squares problem (suppose the minimizers are $\hat{\beta}_0$ and $\hat{\beta}_1$)

$$\underset{\beta_0, \beta_1}{\text{minimize}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \left(= \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 \right)$$

- $e_i = y_i - \widehat{y_i}$ represents the i-th *residual*
- We define the *residual sum of squares* (RSS) as

$$\begin{aligned} RSS &= \textstyle\sum_{i=1}^{n} (y_i - \widehat{y_i})^2 \\ &= (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + \cdots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2 \end{aligned}$$

Assessing the performance of the Model

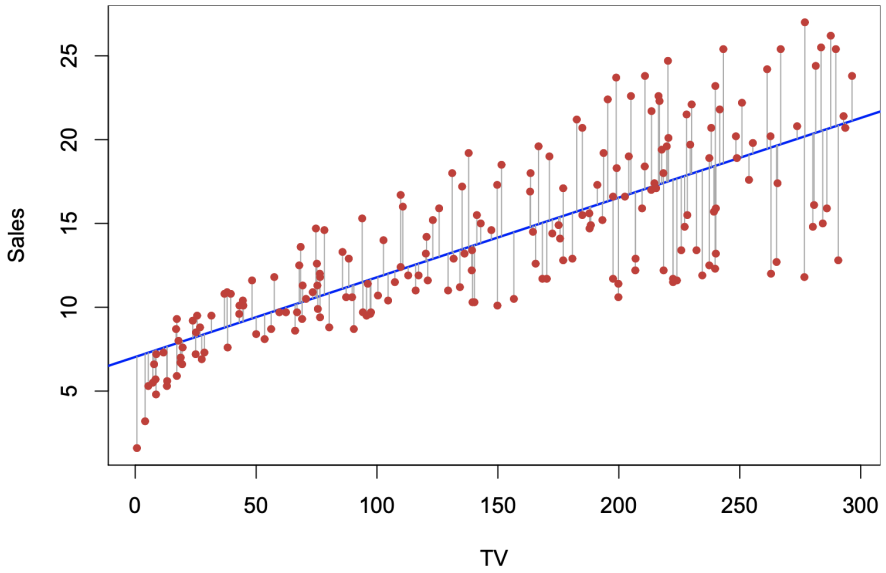
$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- *Residual standard error* (RSE)

$$RSE = \sqrt{\frac{1}{n-p-1} RSS}$$

p = # of features (p = 1 in simple linear regression)

- The RSE is an estimate of the standard deviation of ε



- RSE = 3.26
- Unit of sales: thousand USD
- Actual sales in each market deviate from the true regression line by approximately 3,260 USD, on average.
- Influenced by unit.

Assessing the performance of the Model

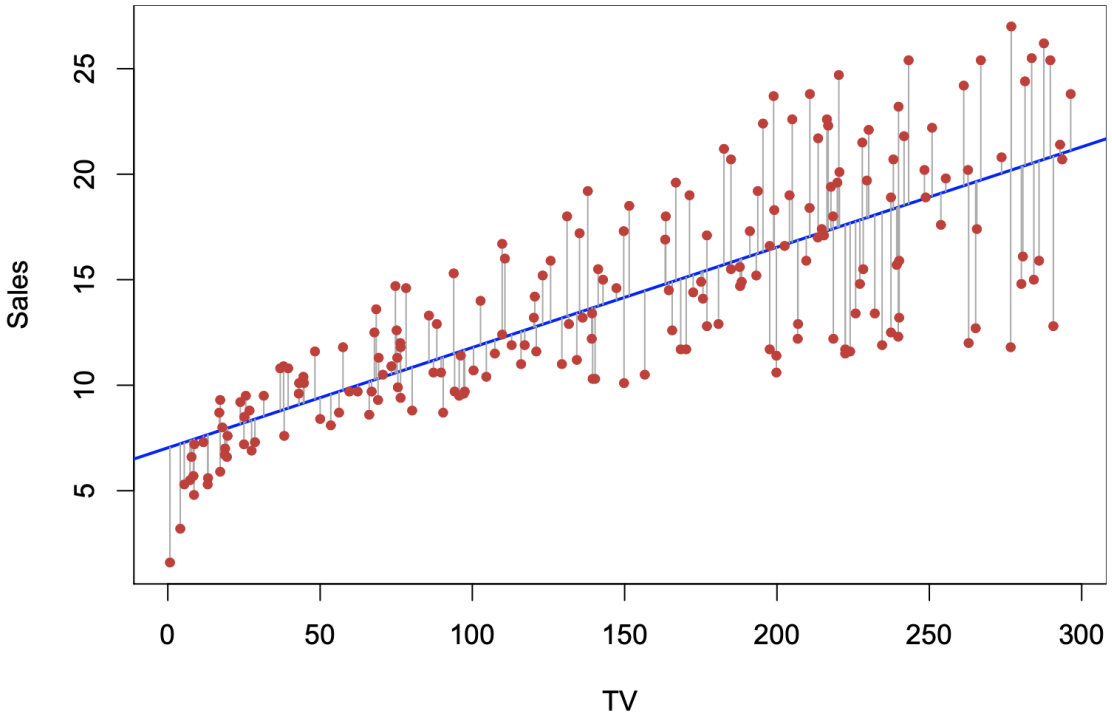
- R^2 (a.k.a. coefficient of determination)

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

where

- $RSS = \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$ is residual sum of squares. $e_i = y_i - \widehat{y}_i$ is the ith residual.
- $TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$ is total sum of squares
- $TSS - RSS = \sum_{i=1}^{n}(\widehat{y}_i - \bar{y})^2$ (note this is only true when one uses the least squares approach)
- This R^2 definition works for both simple linear regression and multiple linear regression
- R^2 is the percent of the variation in the response explained by the regression model; a common measure for how good a linear fit is.
- $0 \leq R^2 \leq 1$ Is a bigger R^2 better?

R^2 (a.k.a. coefficient of determination)



- R^2 = 0.61.
- 61% the variability in sales is explained by a linear regression on TV.
- Not affect by the units.

Correlation and R^2

- Recall (sample) correlation r between X and Y :

$$r(X, Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

- Measures the **linear** dependency between two numerical variables.
- For simple linear regression, $R^2 = r^2$. This relation does not extend to multiple linear regression.
- In different application fields, *good* R^2 values are vastly different.

Back to Python implementation on the height data

```
X = height.drop('child', axis =1).values
y = height['child'].values
## DataFrame.values returns the NumPy representation of the Data Frame, and the axis label will be removed.
## Equivalently, one can use .to_numpy().
print(type(X)); print(type(y)) ## try to remove the .values and see the types
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

Figure 6: Data preparation

```
linear_model = LinearRegression(); linear_model.fit(X, y)
## if X were created by X = height['parent'] it is necessary to transform X
## by.reshape(-1,1) before calling the fit function, for the instructor's way,
## X is already two dimensional.
r_sq = linear_model.score(X, y); print('coefficient of determination:', r_sq)
```

```
coefficient of determination: 0.5712707984937204
```

```
print('intercept:', linear_model.intercept_); print('slope:', linear_model.coef_)
## You can notice that .intercept_ is a scalar, while .coef_ is an array.
```

```
intercept: 25.641176413281514
slope: [0.62145545]
```

Figure 7: Fitting a simple linear regression model

Making Predictions

```
y_pred = linear_model.predict(X) ## making prediction on the training X.
```

```
x_new = np.arange(50, 60).reshape((-1, 1))
## Making prediction on some new points
## Here the .reshape(-1,1) function is necessary to make a 1-D array two dimensional
```

```
y_new = linear_model.predict(x_new); print(y_new)
```

```
[56.71394872 57.33540416 57.95685961 58.57831506 59.1997705  59.82122595
 60.4426814  61.06413684 61.68559229 62.30704773]
```

Figure 8: Prediction on training and new x

How do we find the minimizer in the loss function?

- We have used the Python LinearRegression as a blackbox to find the $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize the RSS.
- People in the precomputer age (or in exam settings) use an exact formula

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \text{ and } \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

  remark: every least squares regression line passes ($\bar{x}$, $\bar{y}$)

- Modern packages use optimization techniques in this class.

Multiple linear regression

In addition to scalar input $x \in \mathbb{R}$, we can consider vector input $x \in \mathbb{R}^p$

- p is feature dimension of input (sometimes use d for *dimension*)
- Inputs of form

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}$$

- Call x the *covariates*, *independent variables*, *explanatory variables*, *features*, *attributes* or *predictor variables*
  - Note that variables are usually NOT independent of one another

Example: housing data

- Output (response, target, dependent) variable is medv, the median home price in different neighborhoods
- covariates include
  - crim: per capita crime rate
  - rm: average number of rooms per dwelling
  - zn: proportion of large lots (zoned for $> 25,000$ feet)
  - river: whether a home is near a river ($x \in \{0, 1\}$)
  - ptratio: pupil-teacher ratio by town

```
housing = pd.read_csv("data/housing.csv"); display(housing.head())
```

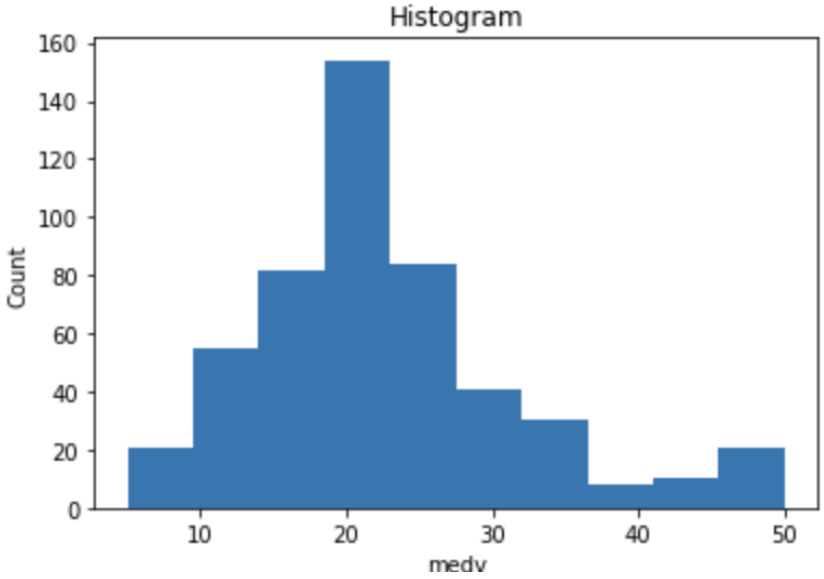
	crim	zn	river	rm	ptratio	medv
0	0.00632	18.0	0	6.575	15.3	24.0
1	0.02731	0.0	0	6.421	17.8	21.6
2	0.02729	0.0	0	7.185	17.8	34.7
3	0.03237	0.0	0	6.998	18.7	33.4
4	0.06905	0.0	0	7.147	18.7	36.2

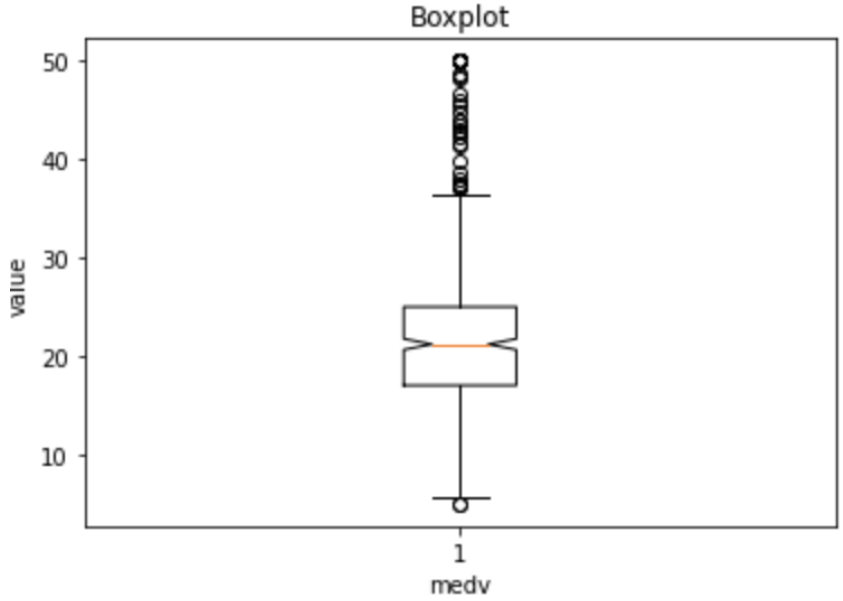
Check data

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 6 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   river    506 non-null    int64  
 3   rm       506 non-null    float64
 4   ptratio  506 non-null    float64
 5   medv     506 non-null    float64
dtypes: float64(5), int64(1)
memory usage: 23.8 KB
```

Check the distribution of Y

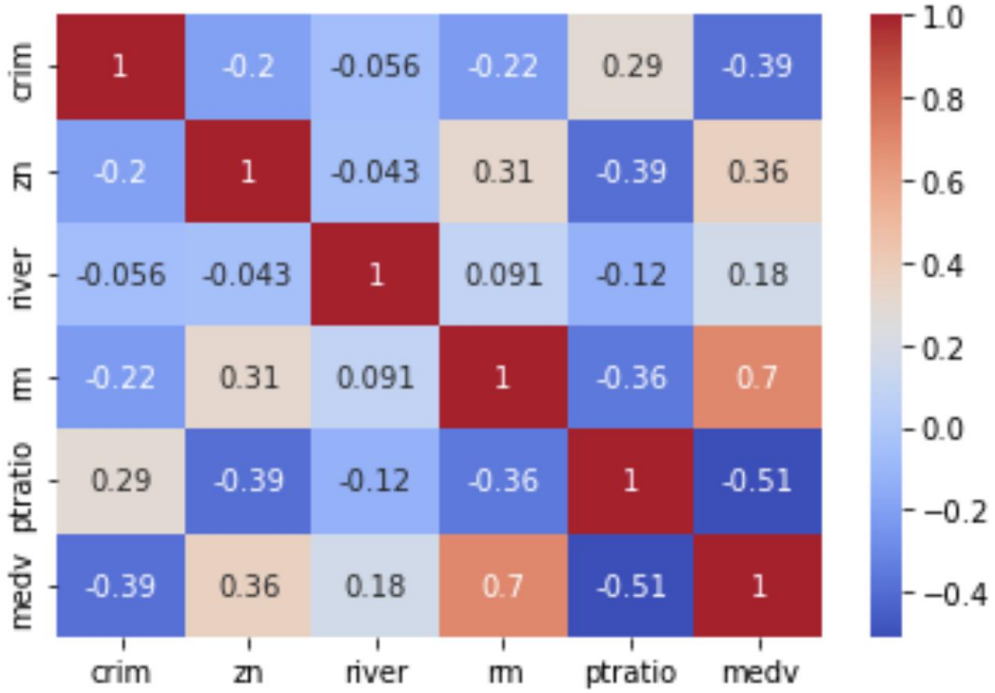




What can you see from the plot?

Check the correlation

	crim	zn	river	rm	ptratio	medv
crim	1.000000	-0.200469	-0.055892	-0.219247	0.289946	-0.388305
zn	-0.200469	1.000000	-0.042697	0.311991	-0.391679	0.360445
river	-0.055892	-0.042697	1.000000	0.091251	-0.121515	0.175260
rm	-0.219247	0.311991	0.091251	1.000000	-0.355501	0.695360
ptratio	0.289946	-0.391679	-0.121515	-0.355501	1.000000	-0.507787
medv	-0.388305	0.360445	0.175260	0.695360	-0.507787	1.000000

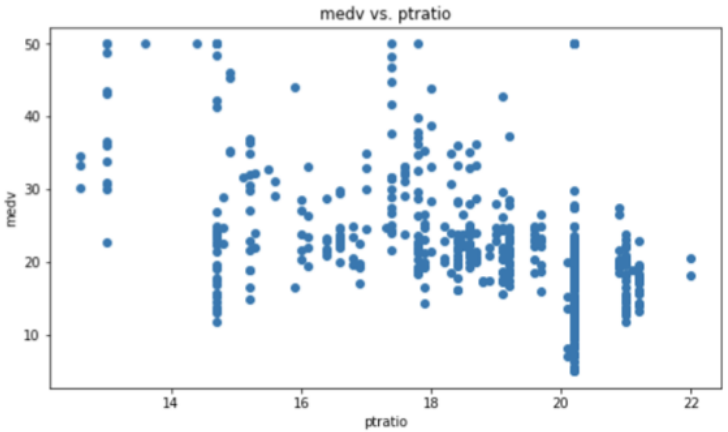


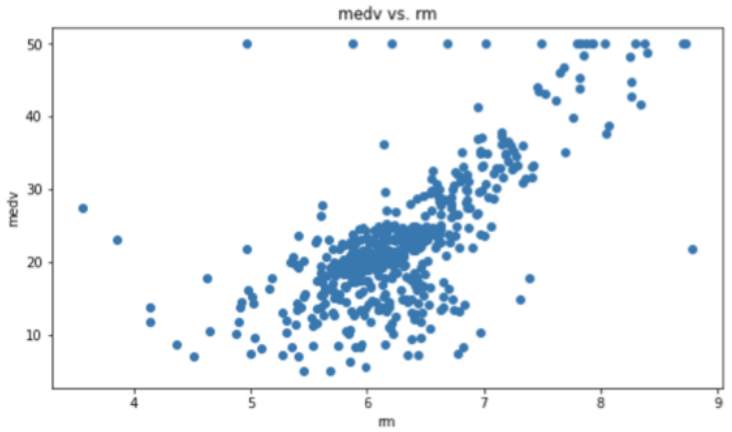
- medv: median home price in different neighborhoods
- crim: per capita crime rate
  rm: average number of rooms per dwelling
- zn: proportion of large lots (zoned for > 25, 000 feet)
- river: whether a home is near a river (0: No, 1: yes)
- ptratio: pupil-teacher ratio by town

Check covariates

- For illustration purpose we only use ptratio and rm as covariates
- It looks like one has negative correlation with medv and the other has positive correlation with medv
- If for some reason, we have to choose only one of these two predictors, which one would you like to choose?

```
plt.figure(figsize=(20, 5))
features = ['ptratio', 'rm']; target = housing['medv']
for i, col in enumerate(features):
    plt.subplot(1, len(features) , i+1) # subplot(nrows, ncols, index). Three separate integers describing
    #the position of the subplot. If the three integers are nrows, ncols, and index in order,
    #the subplot will take the index position on a grid with nrows rows and ncols columns.
    #index starts at 1 in the upper left corner and increases to the right.
    x = housing[col]; y = target
    plt.scatter(x, y)
    plt.title('medv' + " vs. " + col)
    plt.xlabel(col); plt.ylabel('medv')
```





```
X2 = housing[['ptratio','rm']].values
y2 = housing['medv'].values
linear_model_2 = LinearRegression(); linear_model_2.fit(X2, y2)
r_sq_house = linear_model_2.score(X2, y2); print('coefficient of determination:', r_sq_house)
```

```
coefficient of determination: 0.5612534621272915
```

```
print(linear_model_2.coef_); print(linear_model_2.intercept_)
```

```
[-1.2671614   7.71407021]
-2.5611648168335925
```

Figure 4: Fitting a multiple linear regression model

- The model fitting is not much different from fitting a simple linear regression model
- Except that one does not need to worry about the covariates being a 1D object
- Try regress medv on rm only, do you see a larger or smaller R square?
- The Python sklearn library focuses on prediction, is not yet fully developed for some typical statistical inferential tasks.

```
import statsmodels.api as sm
X3 = sm.add_constant(X2)
ols = sm.OLS(y2, X3)
ols_result = ols.fit()
ols_result.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.561
Model:	OLS	**Adj. R-squared:**	0.560
Method:	Least Squares	**F-statistic:**	321.7
Date:	Sun, 31 Jan 2021	**Prob (F-statistic):**	1.04e-90
Time:	20:23:15	**Log-Likelihood:**	-1631.8
No. Observations:	506	**AIC:**	3270.
Df Residuals:	503	**BIC:**	3282.
Df Model:	2		
Covariance Type:	nonrobust		

- continue from the previous page. The summary also contains:

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.5612	4.189	-0.611	0.541	-10.791	5.669
x1	-1.2672	0.134	-9.440	0.000	-1.531	-1.003
x2	7.7141	0.414	18.650	0.000	6.901	8.527

- Compare the coefficients fitted by statmodels and sklearn
- What are the second to the last columns about?
- We see some t-statistics and their p-values here. They indicates the significance of the each covariates.

Quick introduction to hypothesis test

- Hypothesis test: **null hypothesis** H_0 vs. **alternative hypothesis** H_a
- Null hypothesis is usually the status quo, old technology, ineffective new drugs. . . .
- What is a **p-value**? the probability of obtaining a result equal to or "more extreme" than what was actually observed, when the null hypothesis is true
- α is a user specified value called *level of significance*
- We should design a statistical test such that $P_{H_0}(\text{reject } H_0) \leq \alpha$
- Reject the null hypothesis if p-value is smaller than or equal to α

https://www.youtube.com/watch?app=desktop&v=zJ8e_wAWUzE&ab_channel=TheOrganicChemistryTutor

	coef	std err	t	P>|t|	[0.025	0.975]
const	-2.5612	4.189	-0.611	0.541	-10.791	5.669
x1	-1.2672	0.134	-9.440	0.000	-1.531	-1.003
x2	7.7141	0.414	18.650	0.000	6.901	8.527

- We see some t-statistics and their p-values here. They indicates the significance of the each covariates.

- H_0: coef = 0

When we use more than one input variables

- Make predictions

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j$$

- Model the data as

$$y = \beta_0 + \sum_{j=1}^{p} \beta_j x_j + \varepsilon$$

  where ε is noise
- As in simple linear regression, choose noise distribution based on
  - Convenience (assume it is normally distributed)
  - Prior knowledge (rarely)

Fitting a multiple regression

- Making predictions using

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j$$

- Fit as in single variable case. Solve (least squares criterion or OLS)

$$\underset{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p}{\text{minimize}} \sum_{i=1}^{n} (y_i - \beta_0 - \beta^T x_i)^2$$

- Since each observation has p values, $x_i = (x_{i1}, \cdots, x_{ip})^T$, where $i = 1, \cdots, n$. In matrix notation, let $y = (y_1 \cdots, y_n)^T$ and

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & x_{23} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & x_{n3} & \cdots & x_{np} \end{bmatrix}$$

where x_{ij} is the ith observation of the jth variable

Is there a relationship between the response and a set of predictors?

- $H_0 : \beta_1 = \cdots = \beta_p = 0$
- H_a: at least one of β_j is non-zero
- Did we miss β_0?
- This hypothesis test is performed by computing the F-statistic,

$$F = \frac{(TSS - RSS)/p}{RSS/(n - p - 1)}$$

- The larger the F-statistic, the stronger evidence against null hypothesis
- Why do we need F-statistic and its p-values? Because inspecting individual t-statistic and p-value is NOT enough to answer the question posted on the top of the slide
- F-statistic is used only when p is small

RSE and R^2 for multiple linear regression

- RSE and R^2 are common measures of model fit.
- R^2 is used most often.
- R^2 will always increase when more variables are added to the model
- Reason: adding another variable to the least squares equations allows us to fit the training data more accurately
- Residual standard error (RSE)

$$RSE = \sqrt{\frac{1}{n-p-1}RSS}$$

- Does RSE always increase/decrease when more variables are added to the model?
- RSE has the same unit as the response. So we cannot compare RSEs across different responses.
- $RSE/\bar{y}$ could be more interpretable than just RSE.

Making prediction

- There are three sorts of uncertainty associated with making prediction using linear model
  - the least squares plane

    $$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1, + \cdots + \hat{\beta}_p x_p$$

    is an estimate for the true population regression plane

    $$f(x) = \beta_0 + \beta_1 x_1, + \cdots + \beta_p x_p$$

  - assuming a linear model for $f(X)$ is almost always an *approximation* of reality
  - Even if you know $f(x)$, the response value cannot be determined perfectly because of the random error ε in the model ($y = f(x) + \varepsilon$)
  - The first two kind of errors are *reducible errors* and the last is *irreducible error*

Categorical (qualitative) predictors and interaction terms

Categorical predictors:

- Sometimes, categorical variables can be useful in making predictions.
- We usually code qualitative variables by *dummy variables* (variables taking values 0 and 1). Why are they called dummy variables?
- Suppose you have a categorical variable with three *levels*. We need to code it with two dummy variables.

```
df
```

	color	size	price	classlabel
0	green	1	10.1	class1
1	red	2	13.5	class2
2	blue	3	15.3	class1

```
# multicollinearity guard in get_dummies
pd.get_dummies(df[['price', 'color', 'size']], drop_first=True)
```

	price	size	color_green	color_red
0	10.1	1	1	0
1	13.5	2	0	1
2	15.3	3	0	0

Categorical (qualitative) predictors and interaction terms

Interaction terms:

- Sometimes, influence of the predictors can not be decoupled into additive terms. In addition to *main effects*, we need *interaction effect*:

$$sales = \beta_0 + \beta_1 \cdot TV + \beta_2 \cdot radio + \beta_3 \cdot (radio \cdot TV) + \varepsilon$$

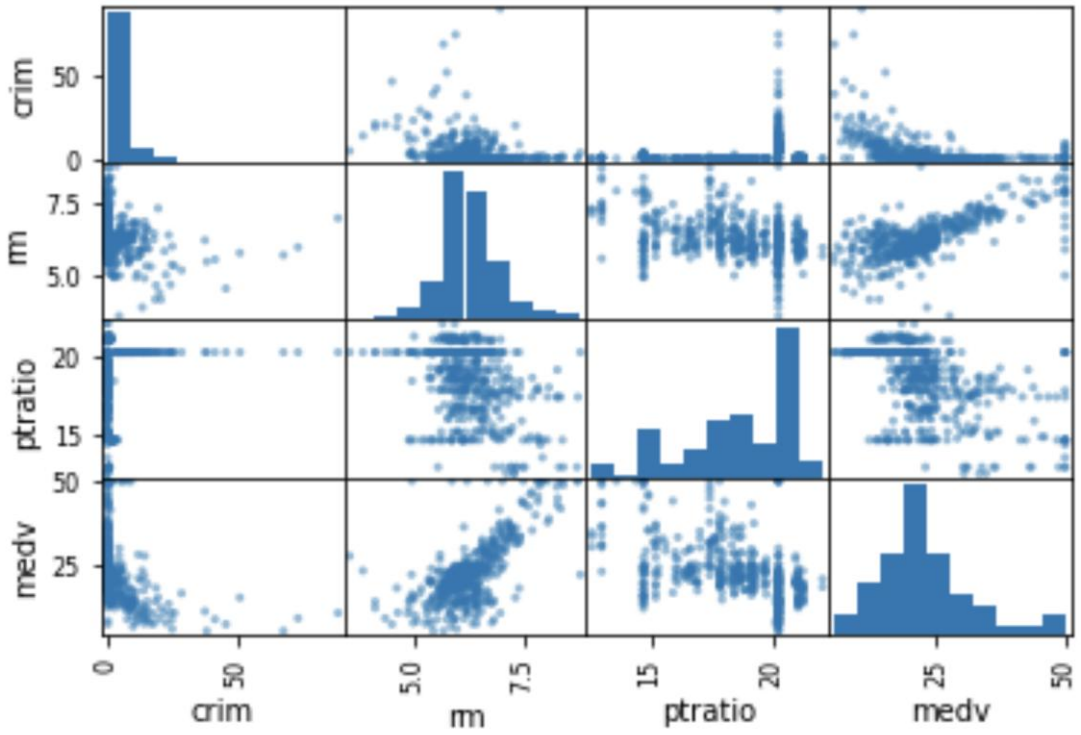
-Rewrite the above equation as:

$$sales = \beta_0 + (\beta_1 + \beta_3 \cdot radio) \cdot TV + \beta_2 \cdot radio + \varepsilon$$

- How do we interpret β_3? the increase in the effectiveness of `TV` advertising for a one unit increase in `radio` advertising (or vice-versa)

Collinearity

- Collinearity: two or more predictor variables are closely related to one another
- Collinearity causes problems to separate out the individual effects of collinear variables on the response



Scatter matrix: One-to-one relationship

Collinearity

- Collinearity: two or more predictor variables are closely related to one another
- Collinearity causes problems to separate out the individual effects of collinear variables on the response
- multicollinearity: collinearity that exists between three or more variables; hard to detect by looking at the correlation matrix
- variance inflation factor(VIF):

$$VIF(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}}$$

  where $R^2_{X_j|X_{-j}}$ is the R^2 from a regression of X_j onto all other predictors
- A VIF value larger than 5 or 10 indicates problematic amount of collinearity. Solution 1: drop variable with high VIF; solution 2: eliminate predictor with the highest p-value in its t-test
- Does collinearity matter for prediction? Not as much as for inference

Performance measure on the (training) dataset does not tell the whole story

- So far, all the performance evaluations are on training data
- They tell us how well the model performs on the SAME data that was used to fit it
- By overfitting the model (e.g., using lots of predictors), we can make the RSS arbitrarily low.
- But this model will not accurately predict future datasets. Prediction of future datasets is called "generalization".
- For fixed amount of data, the larger the model, the larger the R^2. Therefore, R^2 cannot be used to compare models with different sizes.
- There are two solutions
  - create model-size adjusted performance measures, such as adj-R^2.
  - evaluate the performance on data NOT used for fitting (training).

Classification

- **Classification**: supervised learning when outcomes are categorical (a.k.a. qualitative)
- The categorical outcomes (responses) are usually called *class labels*.
- Both classification and regression are supervised learning
- Classification is perhaps the most widely used machine learning methods. Examples include email spam filter, credit card fraud detection, automatic cancer diagnosis, etc.

- In applications where (acurate) "labels" in theory exist, but we do not have access to them, we cannot formulate them as classification problems. Name one example? How about medicare/medicaid fraud?
- There are many off-the-shelf classification methods. In this lecture, we will begin with the most common (basic) one:
  - logistic regression

What is the usual objective for classification?

- Binary classification is the most common classification scenario
- Features $X \in R^p$ and class labels $Y \in \{0, 1\}$
- A classifier h is some function (usually data-dependent function) that maps the feature space into the label space. One can think of a classifier as a data-dependent partition of the feature space

- The classification error (risk) is the probability of misclassification. In other words:

  $P(h(X) \neq Y)$, where P is regarding the joint distribution of (X, Y).

- $P(h(X) \neq Y)$ is usually denoted by $R(h)$
- *Often* (NOT always), we construct classifiers to minimize the classification error.

Binary classification

- Note that the classification error can be decomposed into two parts

$$P(h(X) \neq Y) = P(h(X) \neq Y | Y = 0) \cdot P(Y = 0) + P(h(X) \neq Y | Y = 1) \cdot P(Y = 1)$$

we will talk more about this decomposition in future lectures

How many parts can we have in a multi-class classification problem with 3 classes or more?

Why not linear regression?

- A general remark: before inventing new methods, we should ask why the existing ones do not suffice
- When the outcome variable has more than 2 categories. For example, an `income` variable has three levels: type A, type B, and type C
  - if we code *type* $A = 1$, *type* $B = 2$, *type* $C = 3$, and run linear regression, then
  - i). we have endorsed an ordering in the types
  - ii). we assumed the same difference between pairs
  - an equally reasonable coding *type* $C = 1$, *type* $B = 2$, *type* $A = 3$ will imply a totally different relationship among the three types
  - each of these codings will lead to different predictions
- When the outcome variable has 2 categories
  - we can introduce *dummy variables*
  - and cut the predicted y's at some level, i.e., declare prediction above that level of class 1, and 0 otherwise
  - but this approach is usually inferior to methods that specifically designed for classification

Logistic regression

- Model the conditional probability $P(Y = 1|X = x)$ (compare with linear model)
- The logistic (a.k.a. sigmoid) function

$$f(x) = \frac{e^{\beta_0+\beta_1 x}}{1 + e^{\beta_0+\beta_1 x}}$$

- Logistic regression model: $P(Y = 1|X = x) = f(x)$.

Logistic regression

- The sigmoid function f takes values between 0 and 1; perfect for modeling probability
- Under the logistic regression model, the *log-odds* or *logit* is linear in the input variable X:

$$\log\left(\frac{P(Y=1|X=x)}{P(Y=0|X=x)}\right) = \beta_0 + \beta_1 x$$

- In some books, the above equation is the definition of logistic regression model, or called *logit model*. These two definitions are equivalent
- For logit function: https://en.wikipedia.org/wiki/Logit

- β_1 can be interpreted as the average change in log-odds associated with a one-unit increase in X
- β_1 does NOT correspond to the change in $P(Y=1|X=x)$ associated with 1 one-unit increase in X

Fitting Logistic regression

- The coefficients β_0 and β_1 in the sigmoid function are unknown
- Need to estimate them from training data
- Q: recall linear regression, what criterion did we use to find the coefficient estimates?
- Given training data (pairs are independent of each other)

$$\{(x_1, y_1), \cdots, (x_n, y_n)\}$$

- And let $p(x) = P(Y = 1|X = x)$. We would like to find $\hat{\beta}_0$ and $\hat{\beta}_1$ such that they maximize the *likelihood function* $l(\beta_0, \beta_1)$:

$$l(\beta_0, \beta_1) = \Pi_{i:y_i=1} p(x_i) \Pi_{i':y_{i'}=0} (1 - p(x_{i'}))$$

- This is called a *maximum likelihood approach*
- The least squares method for linear regression is in fact a maximum likelihood approach

Example

Suppose we collect data for a group of students in a statistics class with variables X_1=hours studied, X_2= undergrad GPA, and Y=receive an A. We fit a logistic regression and produce estimated coefficients $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, $\hat{\beta}_2 = 1$. (Take $Y = 1$ to mean `receive an A`)

- Estimate the probability that a student who studies for 40 hours and has an undergrad GPA of 3.5 to get an A in the class.
- How many hours would the student in the previous part need to study to have a 50% chance of getting an A in the class?

Linear discriminant analysis

- Linear discriminant analysis (LDA) is a model that appears often in the statistics/ML literature for its nice mathematical properties to analyze
- We skip its details here because it usually has a similar empirical performance to logistic regression, and in practice, it has far less popular compared to logistic regression
- However, it is worth to know that from the modeling perspective, LDA and logistic regression are two different approaches.
- What is an LDA model:

$$X|(Y=0) \sim \mathcal{N}(\mu_0, \Sigma) \quad \text{vs.} \quad X|(Y=1) \sim \mathcal{N}(\mu_1, \Sigma)$$

- Key: two class normal; different means, but same covariance matrix
- How about model fitting? The means and covariance are unknown parameters. We use the so-called **plug-in approach** (not required).
- LDA gives linear decision boundary.

Bayes' Theorem

- Let A and B be two events. Then

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- An example involving Bayes' Theorem. Suppose in the population, there are 5% of the people using drug D. Scientists have developed a test to identify the users of this drug. This test has false negative rate 10% and false positive rate 20%. Then given that a person is tested positive, what is the probabilty that he/she is a drug D user?

Bayes' Theorem

- Suppose $Y \in \{0, 1\}$, and $X|(Y = 0)$ and $X|(Y = 1)$ are continous
- $P(Y = 1|X = x) = \frac{f_{X|Y=1}(x) \cdot P(Y=1)}{f_X(x)} = \frac{f_{X|Y=1}(x) \cdot P(Y=1)}{f_{X|Y=1}(x) \cdot P(Y=1) + f_{X|Y=0}(x) \cdot P(Y=0)}$
- The above allows one to compute $P(Y = 1|X = x)$ from the LDA assumption

K-nearest neighbors (KNN)

- 'KNN': to predict class label for an observation $X = x$, the K training observations that are closest to x are identified. Then x is assigned to the class to which the plurality of these observations belong
- Why did we say "plurality" instead of *majority*?
- Can predict the class label for any $x \in R^p$ (including the training observations) in this way
- Special cases: $K = 1$ and $K = n$ (n is the training sample size)

```
import numpy as np
import pandas as pd
url="https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
# Assign colum names to the dataset
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
# Read dataset to pandas dataframe
dataset = pd.read_csv(url, names=names); dataset.head()
```

```
##    sepal-length  sepal-width  petal-length  petal-width        Class
## 0           5.1          3.5           1.4          0.2  Iris-setosa
## 1           4.9          3.0           1.4          0.2  Iris-setosa
## 2           4.7          3.2           1.3          0.2  Iris-setosa
## 3           4.6          3.1           1.5          0.2  Iris-setosa
## 4           5.0          3.6           1.4          0.2  Iris-setosa
```

- This is perhaps the best known database to be found in the pattern recognition literature. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

```
np.unique(dataset["Class"])
```

```
## array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

```
dataset.info()
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 150 entries, 0 to 149
## Data columns (total 5 columns):
##  #   Column        Non-Null Count  Dtype
## ---  ------        --------------  -----
##  0   sepal-length  150 non-null    float64
##  1   sepal-width   150 non-null    float64
##  2   petal-length  150 non-null    float64
##  3   petal-width   150 non-null    float64
##  4   Class         150 non-null    object
## dtypes: float64(4), object(1)
## memory usage: 6.0+ KB
```

```
dataset.describe()
```

```
##        sepal-length  sepal-width  petal-length  petal-width
## count    150.000000   150.000000    150.000000   150.000000
## mean       5.843333     3.054000      3.758667     1.198667
## std        0.828066     0.433594      1.764420     0.763161
## min        4.300000     2.000000      1.000000     0.100000
## 25%        5.100000     2.800000      1.600000     0.300000
## 50%        5.800000     3.000000      4.350000     1.300000
## 75%        6.400000     3.300000      5.100000     1.800000
## max        7.900000     4.400000      6.900000     2.500000
```

```
from sklearn.model_selection import train_test_split
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values
X_train, X_test, y_train, y_test=\
  train_test_split(X, y, test_size=0.20, random_state=5, stratify=y)

from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

```
## KNeighborsClassifier()
```

```
y_pred = classifier.predict(X_test)
np.mean(y_pred != y_test)
```

```
## 0.03333333333333333
```

Understanding different errors

- Type I/II error from the confusion matrix
- Specificity = 1 – type I error
- Power (a.k.a. sensitivity; a.k.a. recall) = 1 – type II error
- Precision = d/(b+d)
- Sometimes, type I error is called false positive rate (fpr); 1- type II error is called true positive rate (tpr)