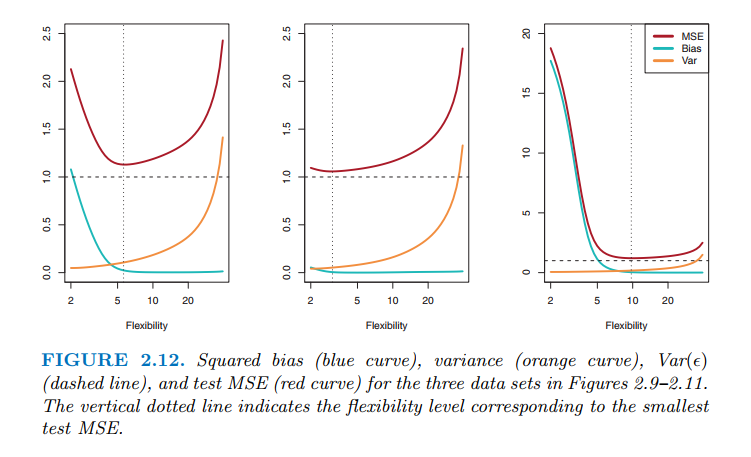
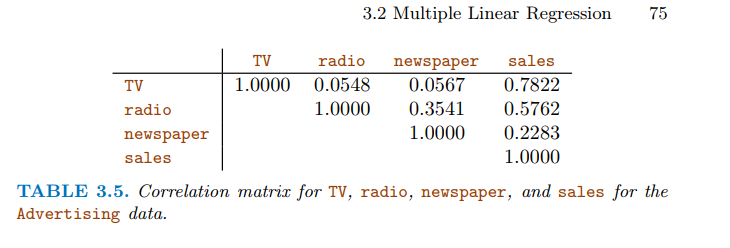
# Bias – variance rate of change

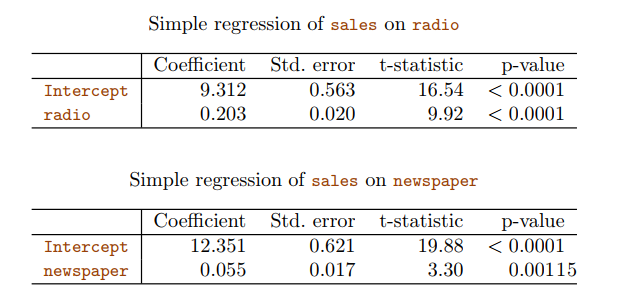


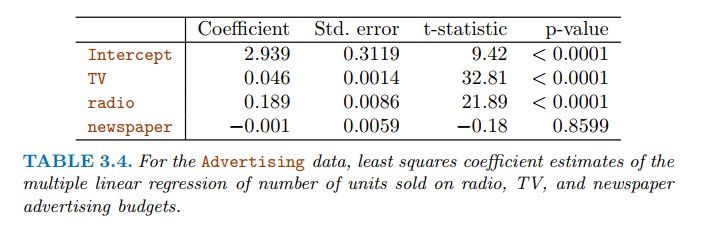
By observing the loss (MSE, redline) and how fast or slow it increases/decreases in relative to flexibility (model complexity), you can tell if the **true model** is simple (linear) or complicate (non-linear with high degree):

* Low decrease and fast increase (2nd graph): model is simple (linear), as such model (at the beginning where flexibility is low) is already good enough which results in low decrease in loss. More complexity => ran wild
* Fast decrease and low increase (3rd graph): complex model. As flexibility increases, model will fit better, hinting that simple model is not good enough. As it reaches its best loss, more complexity won’t help much, resulting in slow increase.

# High feature correlation but low importance for multiple regression: correlation ~~=>~~ causation







(dependent variable: **sales)**

(<http://www.statsmodels.org/stable/index.html> for python library to do this)

(Notebook: [https://github.com/tdpetrou/Machine-Learning-Books-With-Python/blob/master/Introduction%20to%20Statistical%20Learning/Chapter%2003.ipynb](https://github.com/tdpetrou/Machine-Learning-Books-With-Python/blob/master/Introduction to Statistical Learning/Chapter 03.ipynb))

Newspaper and radio have high correlation (0.35), and individually (simple regression), radio and newspaper’s coefficients are okay (2nd pic)

But for multiple regression, newspaper coefficient is really low (-0.001), and seems to be insignificant (high p-values)

* Explanation: maybe when radio budget increases, **newspaper budget also increases** **as a side effect, thus does not contribute to sale after all. Newspaper just take some “credit” for the effect of radio on sales**

Radios - sale

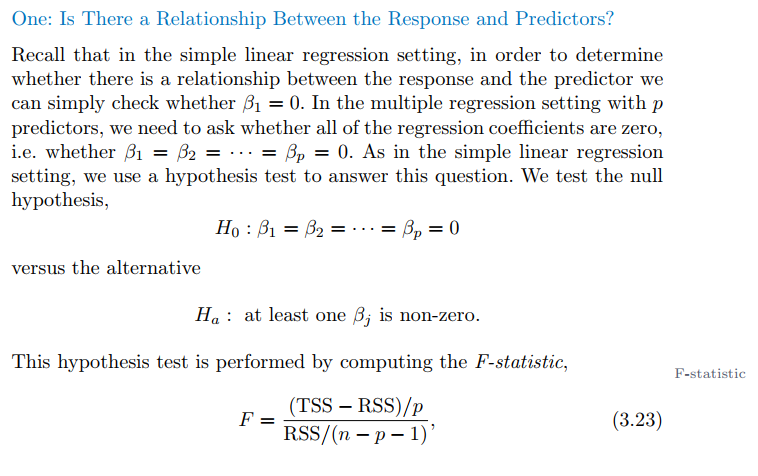
\ ~~|~~

newspaper

**This would explain the saying: correlation does not imply causation**

# F-statistics

To test hypothesis on a **list of variables to see if there is a relationship between AT LEAST ONE OF the features and class labels**



F >>1 => reject null hypothesis

In python, this will also be included when using statsmodels  
results1 = smf.ols('medv ~ lstat', data=boston).fit()

**Note**: F-statistics can also use to perform F hypothesis test COMPARING TWO MODELS

In python, use **from** **statsmodels.stats.anova** **import** anova\_lm

## Vs calculating p-values for each individual feature

Given these individual p-values for each individual feature (that provide info about whether this individual is related to the label), why do we need to look at the overall F-statistic? After all, it seems likely that *if any one of the p-values for the individual variables is very small* ***(e.g. p<.05 when we choose 95% CI)****, then at least one of the predictors is related to the response.*

consider an example in which p = 100 and H0 : β1 = β2 =. . . = βp = 0 is true, so no variable is truly associated with the response. In this situation, **about 5 % of the 100 p-values (think of these 100 p-values come from a normal distribution) associated with each variable will be below 0.05 by chance (because when we repeat the experiment, in 95% of the experiments, the value will fall within its range/CI, meaning the result is not significant**. Note that this conclusion does not drawn from the definition of p-values**)**.

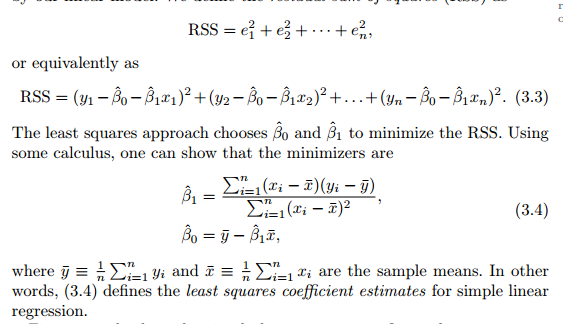
In other words, we expect to see approximately **five p-values small enough (<0.05, thus significant) even in the absence of any true association between the predictors and the response**.

* **Don’t look at small p-value of individual feature to determine association between that feature and response, ESPECIALLY WHEN # OF FEATURES ARE HIGH (>100)**

For **F-statistics, it adjusts for the number of predictor** (when you set up null and alternative hypotheses). Hence, if H0 is true, there is only a 5 % chance that the F-statistic will result in **ONE** p-value below 0.05, **regardless of the number of predictors or the number of observations.**

F-statistics works well with small number of features, favorably when p <<n

# Least square coefficient estimate (formula)



# Feature selection for LR

## Try out all combination of features: 2\*\*(# of features) model

## Forward selection (can be used with large p, as when p>>n)

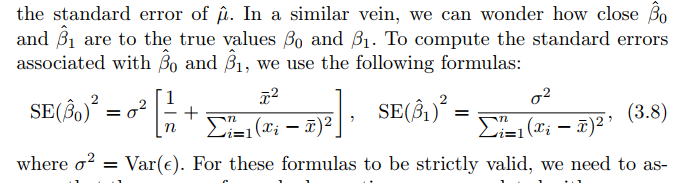
* Begin with null model
* Fit p simple LRs (one by one independently), and add a single feature **with lowest lost** to null model
* Fit p-1 simple LRs, add a single feature with lowest to above model
* Repeat until a stop rule is satisfied

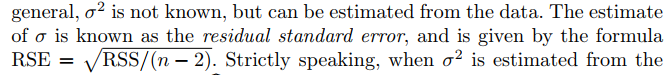
## Backward selection (cannot be used with large p as p-value is unreliable. See above)

* Start with LR and fit all variables + p features
* Remove feature with highest p-value, and fit LR using p-1 features
* Repeat until a stop rule is satisfied (when all p-values left are < threshold)

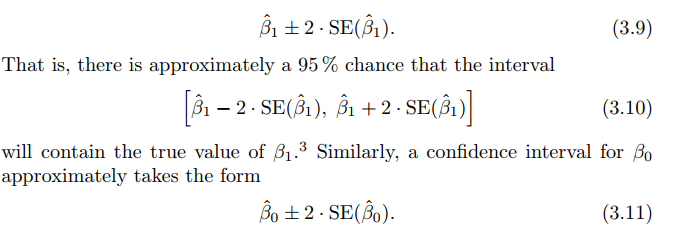
# Use confidence interval to evaluate LR coefficients/ quantify the uncertainty surrounding predicted labels

## Use CI to see how close the estimation B^0,B^1,B^2… to the real coefficient B0,B1,B2 …





The estimate error (residual standard error) based on the assumption: the true errors are Gaussian



## Revisit: How to interpret confidence interval

We use a **confidence interval** to **quantify the uncertainty surrounding the average sales over a large number of cities**.

For example, given that Xtv = $100,000 is spent on TV advertising and Xradio = $20,000 is spent on radio advertising in each city, the 95 % confidence interval for sale is **[10,985, 11,528]**. We interpret this to mean that 95 % of intervals of this form will contain the true value of f(Xtv,Xradio)

**In other words**, if we collect a large number of data sets like the Advertising dataset (taking a big sample from a population), and we construct a confidence interval for the **average sales** on the basis of each data set (given $100,000 in TV and $20,000 in radio advertising), then 95 % of these confidence intervals will contain the true value of average sales.

## Prediction interval

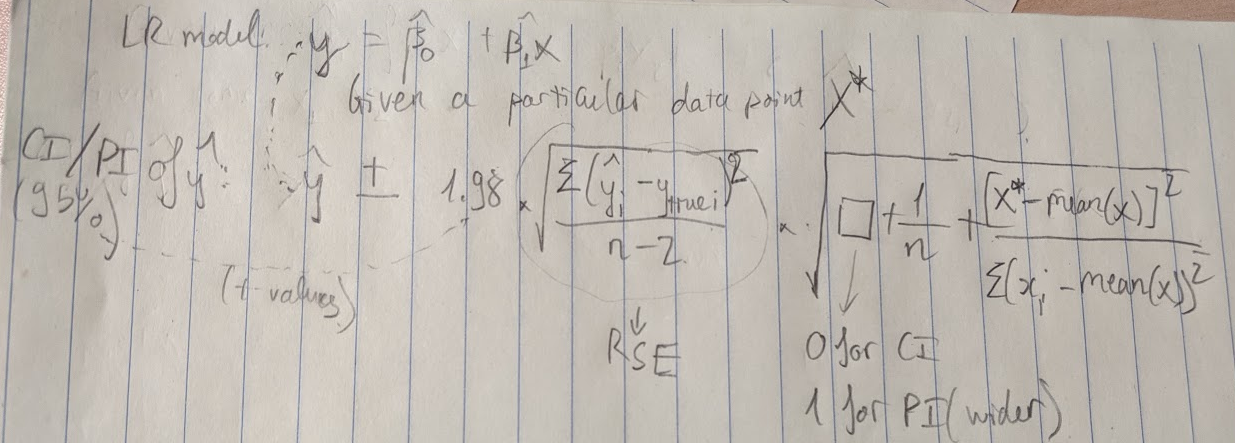
Since Y^ came from a linear model, **LR model cannot predict perfectly due to irreducible error**

* We use prediction intervals to see how much Y will vary from predicted Y^ including this irreducible error, **for a particular city (one sample)**

Prediction interval therefore **should be bigger than confidence interval above**

Given that $100,000 is interval spent on TV advertising and $20,000 is spent on radio advertising in that city the 95 % prediction interval is [7,930, 14,580]. We interpret this to mean that 95 % of intervals of this form will contain the true value of Y for this city

How is this calculated? (**only for 1 predictor X, so not for multi linear regression)**



# Assumptions/Problems of Linear Regression

## The errors are not correlated with one another

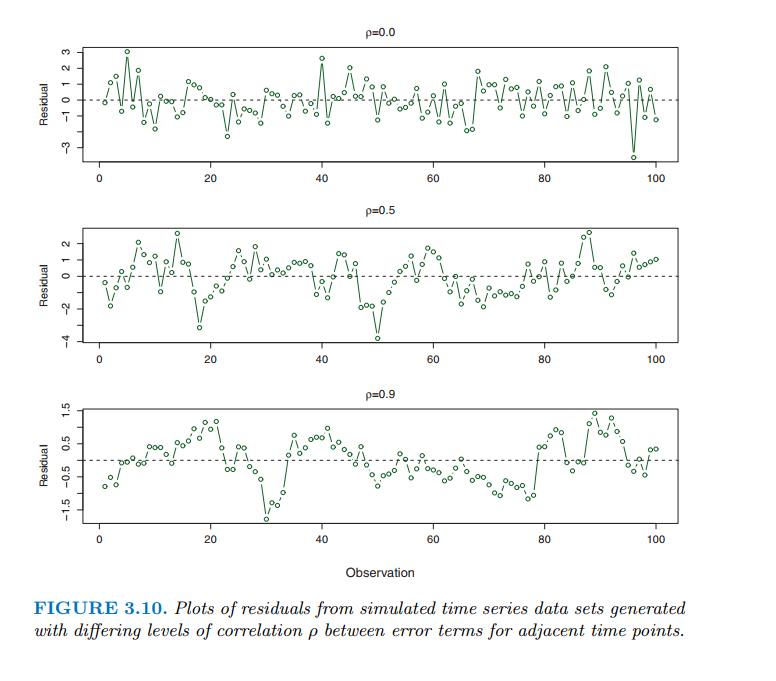
The standard errors that are computed for the estimated regression coeffs are based on the assumption of uncorrelated error terms. If there is correlation error terms, the estimated SE will < the true SE => smaller confidence interval => p-values that are not significant become significant => erroneous conclusion for statistical significance.

E.g of correlated errors:

* accidentally include a duplicate of the data,
* time series data (where observations obtained at adjacent time points **will have positively correlated errors,** due to the nature of data, or the bias/error from that one human collecting the data)
* Collecting data from people who subtly belong to some group (family, neighborhood)
* How to combat correlated error? **Good experimental design** (good AB test? Good data collection? Good survey strategy?)

### How to spot correlated error (specifically in time-series data?)

Use residual plot (data points vs y^ - y), to see if there is pattern in data using a fit model

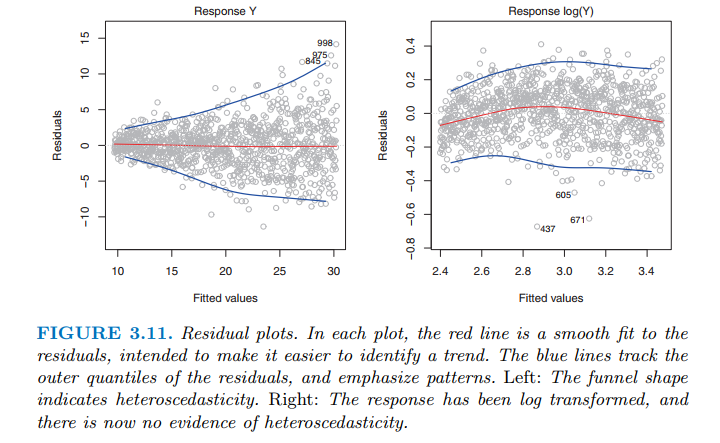


Note at p=0.9, clear pattern **of points above 0 or below 0 for a long adjacent time => adjacent residuals take on similar values!**

## The errors (residual of predicted and actual) are normally distributed and have constant variance (heteroscedasticity problem)

Constant Var(*i*) = *σ*2

If the errors are not normally distributed (skewed in some way), residual plot taking a funnel shape (small in one side, huge in the other)



Solution: log scale the y

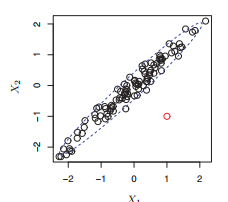
## Outliers (for response y)

Typically speaking, an **outlier** **label** won’t affect linear regression much. Though it will result in higher loss metric which does not really affect the real performance

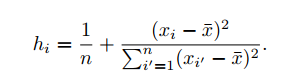
Spot outlier? Either plot data, or use **‘studentized residuals’** plot (divide each residual by its estimated standard error, which is sqrt(RSS/ (n-2)) ). Typical outliers will be >3 or <-3

## High leverage points (outliers for predictor X)

Special case: In case of multiple features e.g X1,X2 , an outlier can have reasonable X1 and X2 individually, but together it’s off. For > 2 features, it’s hard to plot and see these outliers



* Compute leverage statistic (TODO): calculate distance between a point and all the point’s mean (x\_bar)



Calculate h and plot leverage:

**from** **statsmodels.graphics.regressionplots** **import** plot\_leverage\_resid2

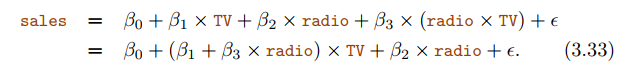
h will be between 1/n and 1. If h > (p+1)/n, then we might suspect high leverage

## **The predictor variables (features) are independent**. An increase in one won't result in an increase in another

So if we have two features: TV ad budget and radio ad budget, and one label: sales, a linear model that has B1 as coefficient of TV mean: **when TV increases by 1 unit, sale will (on average) increase by B1** (assume B1>0)**, regardless of radio**

We know it might not be true in real life, e.g. spending $ on radio ad might increases effectiveness of TV ad (**synergy, interaction effect)**

* Have to add this dependency/interaction effect in our model by adding an *interaction term* (which is **just a new feature which is a multiplication the other 2 interacting features)**



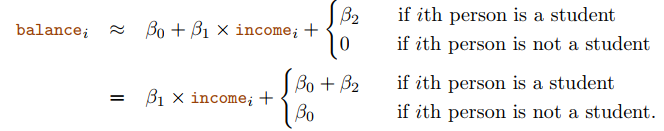
Now $1 increase in TV ad means that sales will increase by B1 + B3\*radio (and B2+B3\*TV for $1 increase in radio)

**Note: Tip: How to determine interaction effect (if there even is one?): Create this new multiplication feature and see either p-value is significant, or loss decreases significantly**

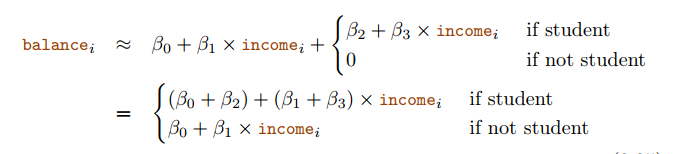
### Interaction term for categorical features

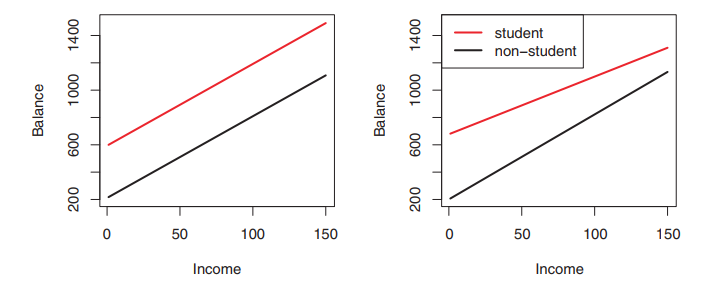
Reduced dummy/one-hot

Note that you only need n-1 new features for categorical features with n unique values (since we will utilize bias): 1 feature to determine girl or boy (0 or 1), 2 features to determine 3 things (10,01,00)



Interaction term between continuous and one-hot feature:





Left (no interaction term): so regardless of whether you are a student or not, an increase in income will lead the same discrepancy in credit card balance (terrible observation!)

Right (with interaction term): an increase in income will result in lower increase in balance for students than non-students

## Collinearity

Happens when two or more features are closely related to one another

* Make it hard to interpret their coefficient: how each one affects the label Y

When two features are highly collinear, what it does to your linear regression (specifically) model:

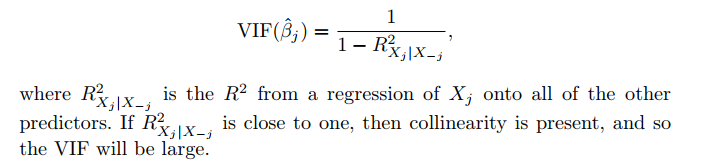
* It messes up the loss contour (RSS, as in the book), results in **multiple coefficient pairs of these 2 features having similar value for RSS**
* Cause standard error for each feature grow -> t-statistic to be smaller => higher p-values => coefficients aren’t statistically significant anymore, even though they can be

**Detect collinearity:**

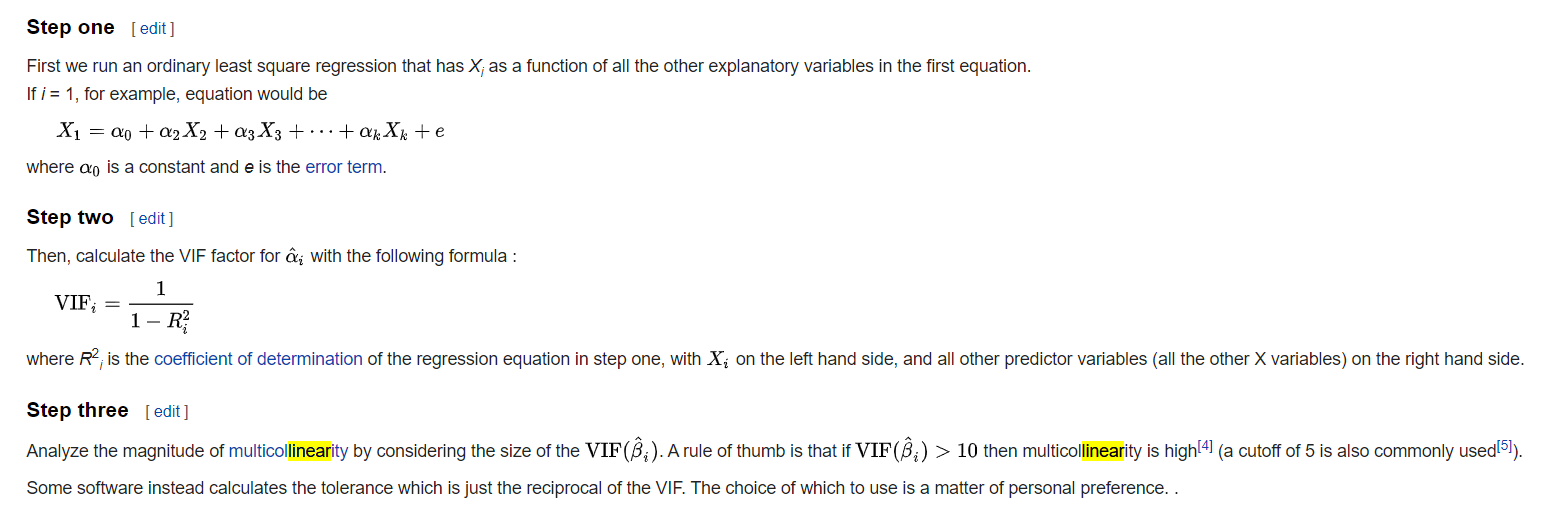
* Plot
* Correlation matrix

**Detect multicollinearity** (no pair has high correlation, but together a group >2 can have high correlation)

* Compute *variance inflation factor*
* The VIF is variance inflation factor the ratio of the **variance of** **βˆj when fitting the full model (with other features)** divided by the **variance of βˆj if fit on its own**
  + Min(VIF) = 1 (no collinearity)
  + VIF >5 or >10 means high collinearity



Calculating VIF: Set the suspected collinearity feature (e.g. X1) as label, and **try to predict X1 using a** **regression model** and other features

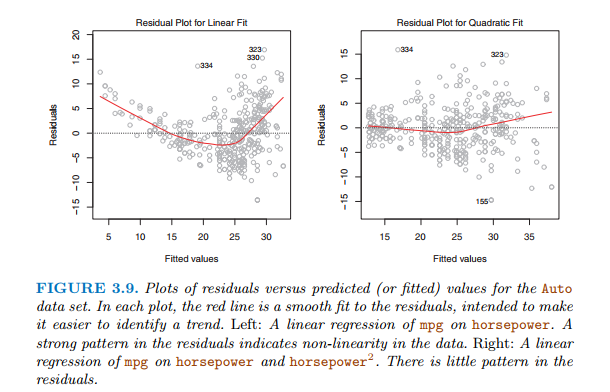


Note; statsmodels python package have this as “variance\_inflation\_factor” function

* What to do with high collinearity:
  + Drop one of them
  + Combine them (average them?) to create a new feature

# Check for non-linear relationships from a linear regression prediction

Use residual plots, and see if there is a strong pattern



# KNN vs Linear Regression

* LR: **parametric** approach

Easy to fit

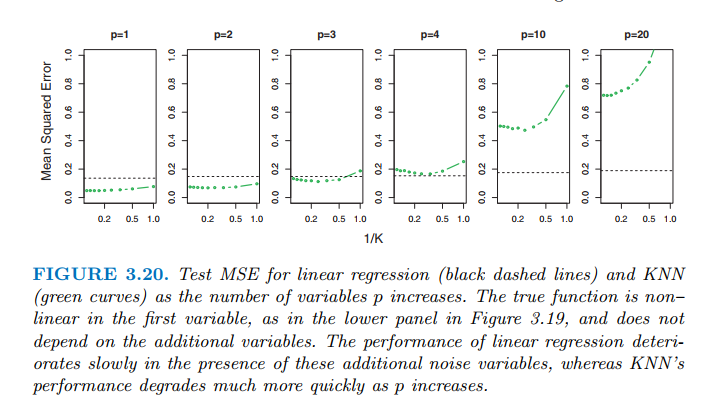
Simple interpretations

Statistical significance test (p-values) can be performed

But: strong assumption about data

* KNN: **non-parametric** approach

|  |  |
| --- | --- |
| **LR** | **KNN** |
| * **parametric** approach   Pros:  +Easy to fit  +Simple interpretations  +Statistical significance test (p-values) can be performed  Cons  - strong assumption about data and parametric form (linear) | **- Non parametric approach**  Pros:  +NO assumption  +Good when data true relationship is not linear  +Work better when data is strongly non-linear (curves all over), **BUT….**  Cons:  -Have none of the good stuff from LR  -**For strongly non-linear data with lots of noisy features, KNN performs worst than LR** (see below) |



When # of features (p) is large, and dataset size (n) is still the same, KNN performs worst **because curse of dimensionality:** That is, K observations that are nearest to a given test observation x0 (when p is small) **may be very far away from x0** in p-dimensional space (when p is increased) => meaning there will be **very few neighbor observations “near” any given observation when we increase # of features.**

**Note: “KNN performs worst because of curse of dimensionality” is only theoretically correct, not empirically correct. In practice KNN might still work very well.**

In practice, we might prefer LR than KNN if KNN metric score is only slightly better than LR, for the sake of interpretation.

## **Curse of dimensionality**:

The curse of dimensionality is this idea that **the more columns you have**, it creates a **space** that is **more and more empty,** the more **all of the points sit on the edge of that space**.

- If you just have a single dimension where things are random, then they are spread out all over.

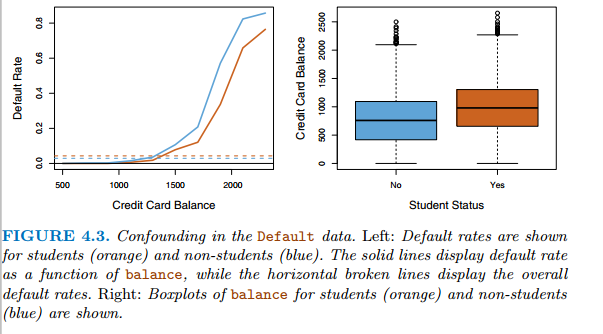
- If it is a square then the probability that they are in the middle means that they cannot be on the edge of either dimension so i**t is a little less likely that they are not on the edge**. **Each dimension you add**, it becomes multiplicatively **less likely that the point is not on the edge of at least one dimension,** so **in high dimensions, everything sits on the edge.**

=> in theory, the distance between points is much less meaningful

# Coefficient changes signs between single LR and multiple LR

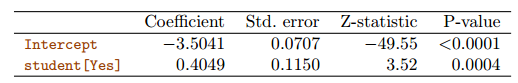
What this means: when using this feature ALONE to predict y, the coefficient is positive ( incr this B1 => prediction increase, whether as regression or as probability for classification), but when using this feature together with others, it changes to negative ( incr this B1 => prediction decrease). **This paradox is pronounced when feature is categorical**

Example: student feature to predict credit card default (binary classification)



Explanation: **feature interaction/correlation** (looking at left graph)

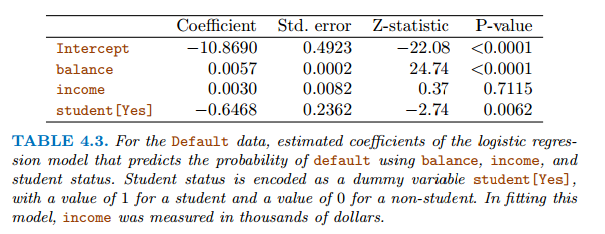
* **Since student is binary feature, when using student alone, the probability rate for default is going to be just 2 fixed values (one for student, one for non-students).** Reminder that student is either 1 or 0, so a change from nonstudent to student means an increase in probability, due to positive coeff. **At this point, other features don’t play a role.** In this case you can treat other features as noise or better, averages (some constant, like adding a constant column to df)



=> The **horizontal broken lines** near the base of the plot, which **show the default rates for students and non-students averaged over all values of balance and income.**

**THIS MEANS: ON AVERAGE (AS A WHOLE), A STUDENT (1) IS MORE LIKELY TO DEFAULT**

* On the other hands, when accounting other features, there will be some interaction, and thus a change of student from 0 to 1 (**for a fixed and specific value of other features)** might have different affect on the probability rate (in this case, **decrease**, **due to negative coeff**)

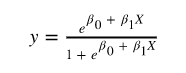


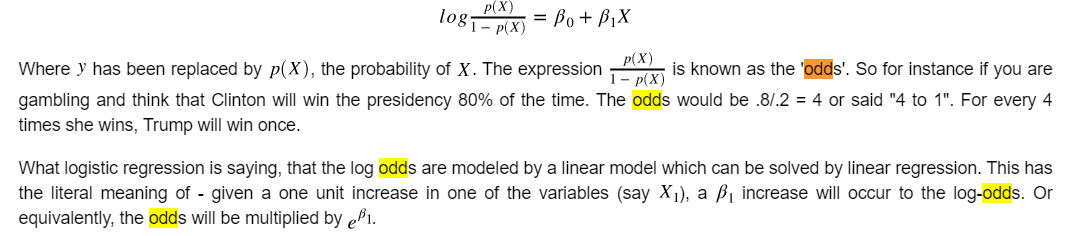
=> How to interpret this paradox: A student is **riskier** than a non-student i**f NO information about the student’s credit card balance (the other feature) is available**. However, **if two people having a same balance, a** student is **less risky** than a non-student to default

**Lesson from this**: aware of the dangers of **performing regressions with only 1 single features while other features may be relevant. 1-feature regression can be vastly different from multiple-feature regression, especially when there is feature correlation**

# Interpret Logistic Regression (sigmoid activation) in terms of ODD

Find function that always outputs number between 0 and 1. Many functions satisfy this condition. For logistic regression the ... logistic function! is used.



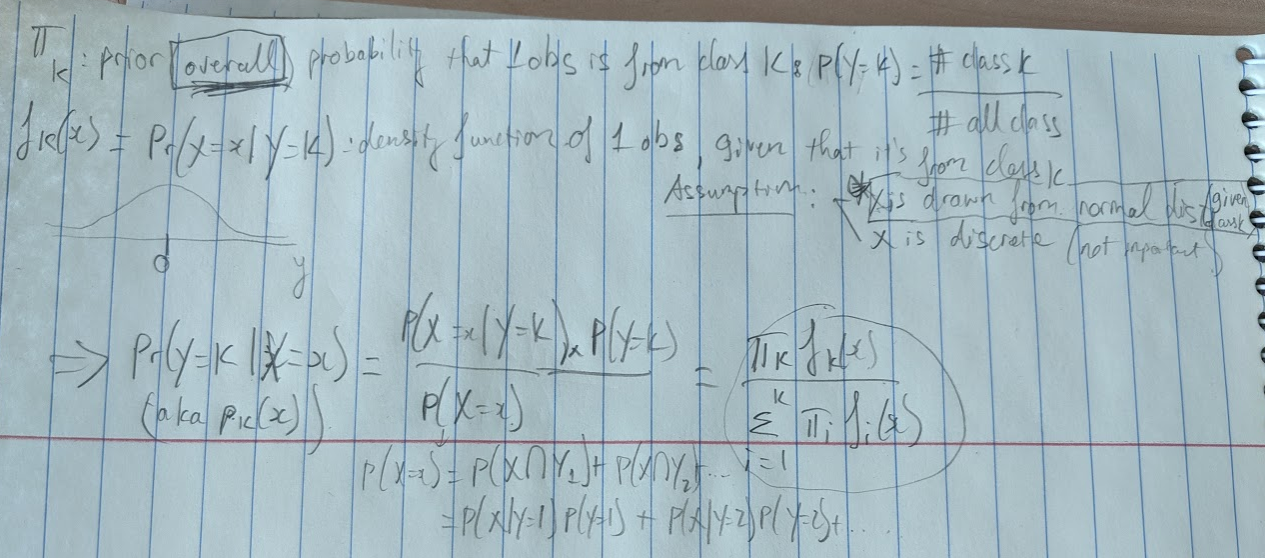


# Linear Discriminant Analysis

## Vs Logistic regression

* **When class are well-separated**, weight estimates for LR are unstable (wow? TODO: check)
* **When n is small**, and **features X are approximately normal** FOR EACH CLASS, LR is not as stable
* LDA is more popular when you have 2 response classes???
* Cons: overfitting if # of features >> # of samples n

## Bayes theorem formula

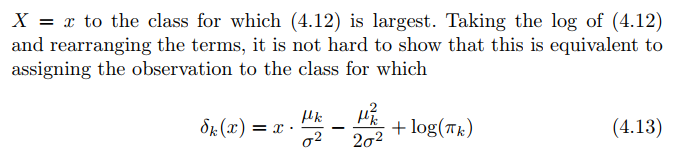
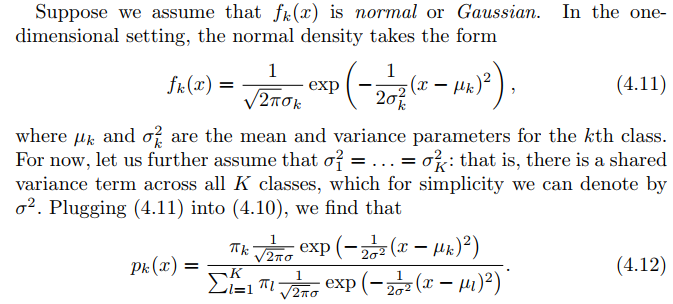


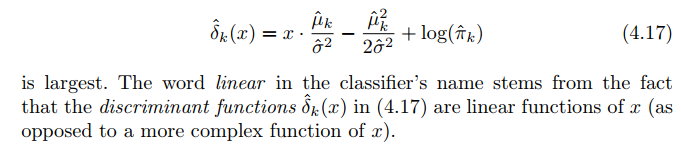
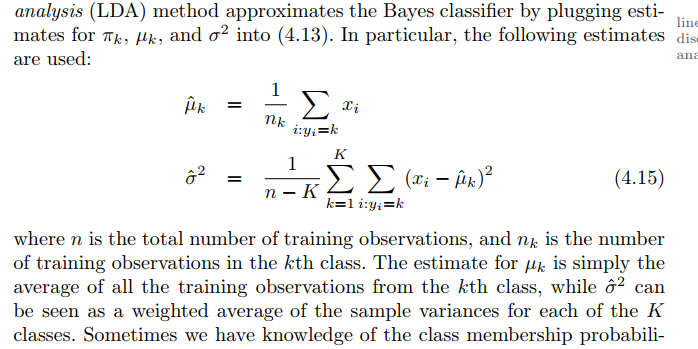
x

Mean(x)

Note: if X is continuous, then fk(x)**dx** would correspond to the probability of X **falling in a small region dx around x**

## Estimation of Bayes classifier (when there is only 1 feature X, aka p=1)

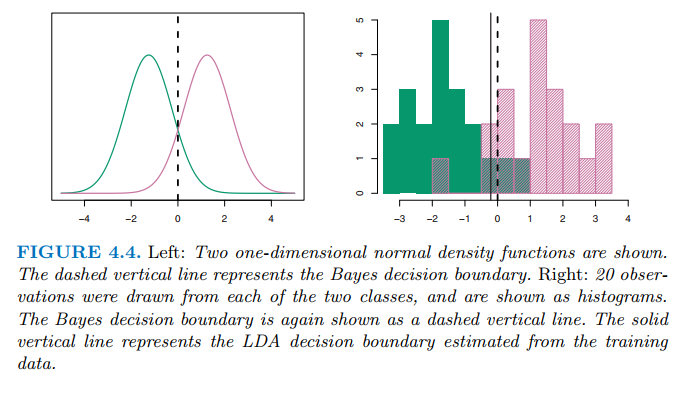




(Where the ‘linear’ in LDA came from)

**How to use formula 4.17**:

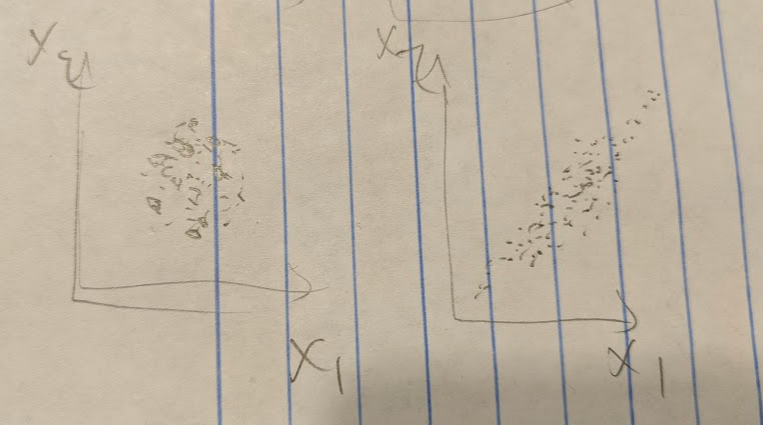
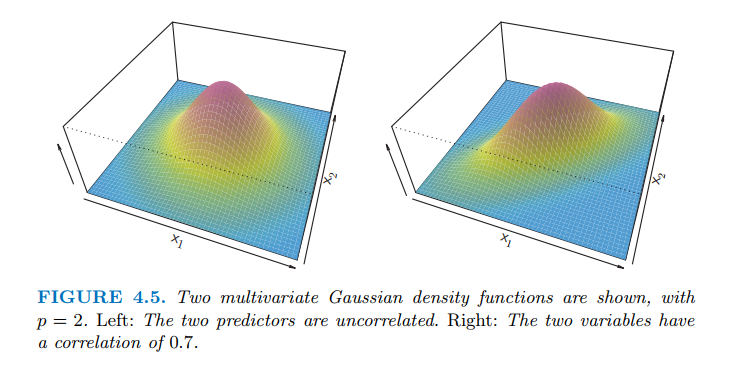
* Estimate *μk*, and *σ*2 using 4.15 (which is mean of kth class and weighted average of variance for each of K class (assuming normal distribution))
* Estimate *πk* from the note screenshot
* Compute decision boundary results from assigning an x to the class which 4.17 **is largest** (**so that if x > this decision boundary, class 0, else class 1**). For a binary classification, the easiest way is to take δ1(x) = δ2(x) (same x), and solve for x.



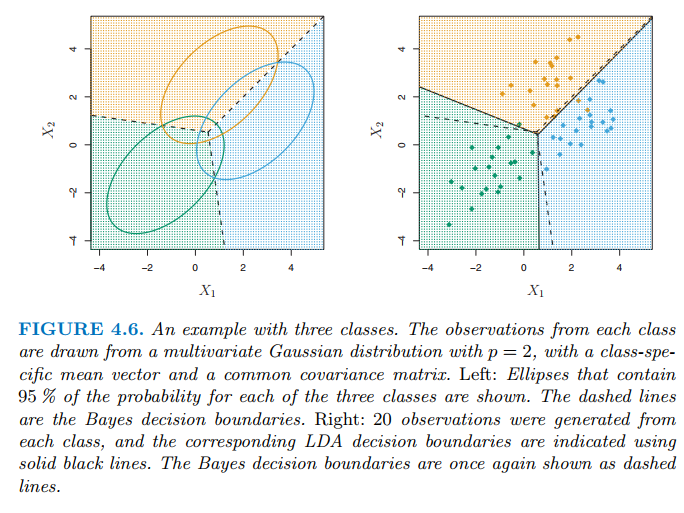
## When there are >=2 features (p>2)

We will have **multivariate normal distribution** (each individual feature has 1D normal distribution, and **there’s some correlation between each pair of predictors**). We also have **covariance matrix**, and we will turn formula (4.13) into a vector/matrix form

LDA assumes that the observations within each class are drawn from a multivariate Gaussian distribution with a **class-specific mean vector** and a **covariance matrix that is common to all K classes (**when class = k1, features correlate the same to each other. Same for class=k2 or k3 … This is similar to ‘shared variance term across all K classes’ when p=1**)**

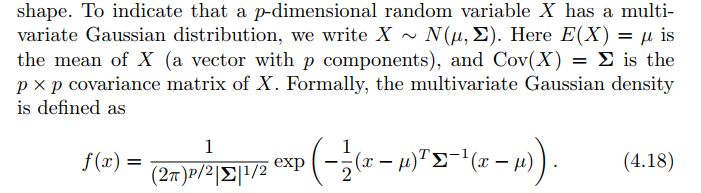
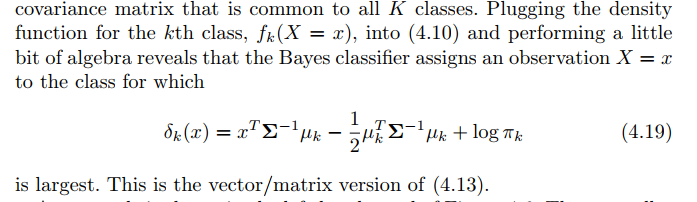


(Example of density functions for multivariate normal distribution). Here we have 2 variables, X1 and X2, **for one specific class k1** (hence only 1 peak node). Left one: the base will look like a circle (in 2D perspective, it’s like a messy circle) , right one: an eclipse (in 2D, it sorta forms a line to signal correlation between 2 features X1 and X2)



(Example of 3 class k1, k2 ,k3. Graph is 2D, so like looking from above)

(See book for formulas)

## Probability threshold using LDA for binary classification (imbalance class)

LDA is trying to approximate Bayes classifier, **which has the lowest TOTAL error rate out of all other classifiers** => it doesn’t care about individual class error (false + or false -)

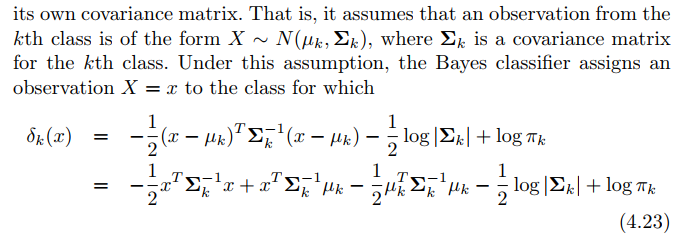
Bayes classifier works by assigning some observation to a class where the pk(X) is greatest. In binary classification, it means **pk(X) > 0.5** (It uses a threshold of 50%). You can lower (more lenient) /raise (more strict) this threshold depending on the class imbalance or domain knowledge (e.g. the focus of the project)

# Quadratic Discriminant Analysis

Reminder: LDA assumes that the observations within each class are drawn from a multivariate Gaussian distribution with a **class-specific mean vector** and a **covariance matrix that is common to all K classes**

QDA: same as LDA (obs from each class are from multivariate Gaussian distribution, and to plug parameters into Bayes’ theorem to perform prediction), **but QDA assumes each class has its own covariance matrix**

As a result, **QDA ends up with a quadratic function in term of x**



# LDA vs QDA

|  |  |
| --- | --- |
| LDA | QDA |
| Lower variance but high bias  **Lower variance**: Because LDA estimates **one** covariance matrix (estimating p(p+1)/2 parameters, or half of a covariance matrix square). Also LDA model is **linear** (# of class \* # of features coefficients to estimate), so ends up with a **linear decision boundary**  **High bias**: LDA’s assumption that K classes share a common covariance matrix is badly off  **Uses**: best use when # of observation (n) is small (we don’t want to overfit a small dataset, so reducing variance is crucial) | **High variance low bias**: QDA estimates a covariance **matrix for EACH CLASS** => Kp(p+1)/2 parameters => **lots of parameters.**  Also QDA is **quadratic,** which ends up with a **quadratic decision boundary (nonlinear => more complex)**  **Uses:** good when training set is large, or assumption of a common covariance matrix for K classes is not suitable |

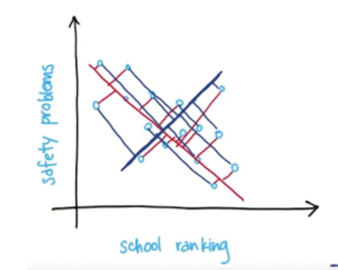
# When to use Logistic Regression, LDA, QDA or KNN

* (Bayes) decision boundary is linear
  + If for each class, features are **uncorrelated** and random-**normally distributed** => **1st:** LDA (can use 1 covariance matrix for all k class, and features are normally dist). **2nd:**Logistic Regression
  + Same as above, but features had **some correlation, same in every class** => **1st** : LDA (can still use 1 covariance matrix). **2nd :** LR
  + Features are generated from **t-distribution** (**not normally distributed anymore**) => **1st:** Logistic Regression. **2nd:** LDA (violates normally distribution assumption)
* Non-linear decision boundary
  + **Normal distribution**. For class k1, there is correlation between features, but in class k2 these features have a different correlation => **1st:** QDA (have to use different covariance matrix for each class now)
  + **Normal distribution** and **uncorrelated features** for every class, but somewhat non-linear => **1st:** QDA, **2nd:** KNN (with k chosen from CV)
  + Same as above, but **extremely non-linear** => **1st:** KNN-CV, **2nd:** QDA (QDA is not powerful/flexible enough)

# Principal Component Analysis

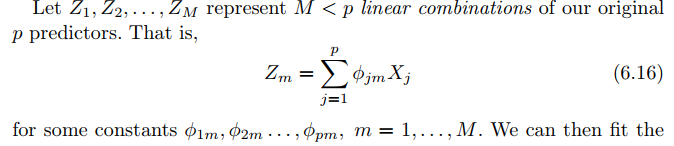
The idea is to come up with **PC** **direction** that **maximize variance** (or **minimize sum of squared distances between the original point and PC direction**/information loss) from old data point (from higher dimension) to its new transformed value in a lower dimension

What does it mean by “maximize variance”? When you project all data into this PC direction, the resulting observations would have the largest possible variance



* More variance of data along a PC direction = higher that PC is ranked
* The next-highest PC should be uncorrelated with the highest PC (in 2 dimensions, the 2nd PC is perpendicular to 1st PC)

After reduction, the new feature Z should be a **linear combination of old features:**



## Pros of PCA:

* Good when data have clear **general patterns** that could be simplified into smaller feature subsets (e.g. for human faces: eyes, mouths …)
* Visualize high dimensional data
* Reduce noise by ignoring PCs that doesn’t capture variance of data well enough
  + Especially useful in high dimensional data situation where # of features >= # of observations
* Fewer inputs for others ML models