Handout_V2

Pierre Bauche

2018-09-03

Contents

Chapter 1

Information

1.1 Usefull ressource

Feature engenering : http://www.feat.engineering/review-predictive-modeling-process.html Semi supervized learning : https://www.analyticsvidhya.com/blog/2017/09/pseudo-labelling-semi-supervised-learning-technique/ - Machine Learning wth unlabbeled data : psuedo labeling

- Book
  - Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig.

1.2 todo

- create Blogdown
- tydiverse package
- Tensorflow for deep learning : open source software library developed at google for complex computation

1.3 interesting stuff

- polynomial regression using kernel smoothing
- Logistic regression using 5 fold stratified cross validation
- blockchain data science
- reinforcement learning
- adversarial training -Deep Learning
  - microsoft service
  - Tensorflow : read exemple
  - RKWard : free and open source Graphical User Interface for the R software
- ML with H2O, lime, Keras
- spark R
- image recognizing with R
- sentiment analysis = microsoft azure

- data privacy
- data quality control
- Web developpement
- Shiny : (microsoft azure serveur pour upload
- Rmarckodw :
- blogdown (Hugo)
- Insurance
- fraud
- underwriting models : predict somebody's insurance risk
- marketing predition
- customer segmentation

1.4 Hint

Chapter 2

Feature engeneering

2.1 Input feature

A feature is a numeric representation of raw data. Feature engineering is the process of formulating the most appropriate features given the data, the model, and the task. If features are't good enought then model canot be good.

Features selection is important. If they are to many fearture, model use noise or irrelevant information or redundant. IF they are not enought feature, model don't have the information .

- **Create new input :**
  - Combine feature
    - reduire dimention
    - reduire colinéarité
    - predictor qui ont du sens
    - use the input interaction
  - kmeans clustering as feature : attetion pas inclure la target risque overfitting
    - a data point can also be represented by a dense vector of its inverse distance to each cluster center. This retains more information than simple binary cluster assignment
  - n-day average (in time series) : peut reduire la variabilite etle noise
  - ratio
- **Tips**
  - Use knowledge to construct a better set of features (business)
  - Visualizing the correlation and check de relation
    - between input and output when output is numeric
    - between different input
  - Normalize the feature if metrics differt or unknow

2.1.1 Numeric Data

Predictors that are on a continuous scale are subject to somes issues that can be mitigated through the choose of model. Models that are smooth functions of input features or model hat use euclidian distance (regression, clustering, ...) are sensitive to the scale. Models based on space-partitioning trees (decision trees, gradient boosted machines, random forests) are not sensitive to scale.

There are a variety of modifications that can be made to an individual predictor that might improve its utility in a model.

- **scaling** : not change the shape of the distribution $/fracx - min(x)max(x) - min(x)$

  - Feature scaling is useful in situations where a set of input features differs wildly in scale.
- **standardization on N(0,1)** : $/fracx - mean(x)sqrt(var())$
  - essential when the distance or dot products between predictors are used (such as K-nearest neighbors or support vector machines)
  - essential when the variables are required to be a a common scale in order to apply a penalty (e.g. the lasso or ridge regression)
- **normalisation** : divide by the euclienne l^2 norme (=sums the squares of the values of the features across data points). SO the feature column has norm = 1
- **Discretization** :
  - fixed width
  - quantile binning

  Variables scaled and standardized are comparable Some models need gaussian input : scale + transform

- **Power transforms** : variance-stabilizing transformations** Power transforms change the distribution of the variable to more symetric distribution
  - log
  - sqrt
  - inverse
  - boxcox : generalisation : Only work for positive variable
  - johnson transform
  - logit transformations : This transformation changes the scale from zero and one to values between negative and positive infinity

2.1.2 count data

Raw counts that span several orders of magnitude are problematic for many models.In a linear model, the same linear coefficient would have to work for all possible values of the count. Large counts could also wreak havoc in unsupervised learning methods such as k-means clustering, which uses Euclidean distance as a similarity function to measure the similarity between data points. A large count in one element of the data vector would outweigh the similarity in all other elements, which could throw off the entire similarity measurement.

- **count transform**
  - binarise 0/1 if value
  - quantizing the count or group the counts
    * fixed-width binning, each bin contains a specific numeric range (ex age)
    * If count have multiple magnitudes, group by powers of 10 (0–9, 10–99, 100–999, 1000–9999, etc)
  - Quantile binning : adaptively positioning the bins based on the distribution of the data
  - log transform

2.1.3 categorical data

Use Dummy or keep factors with somes levels is same for most modeling. It suggest using the predictors without converting to dummy variables and, if the model appears promising, to also try refitting using dummy variables.

- **unordered categorical data**
  - dummy coding : in Feature engineering, il recommande de flag chaque variable categorielle en varible binaire
  - effect coding : -1 0 1 : -1 si different de categorie de reference. Effect coding is very similar to dummy coding, but results in linear regression models that are even simpler to interpret.

- **Dealing with Large Categorical Variables**
  - do nothing
  - dummy : create many variable with zero value for rare categories and add zero-variance predictor(computentional intencive)
  - delete rare value
  - recode and regroup categorical data
  - Compress the features. There are two choices:
    * Feature hashing, popular with linear models. A hash function is a deterministic function that maps a potentially unbounded integer to a finite integer range [1, m]. Feature hashing compresses the original feature vector into an m-dimensional vector. It Converte large cat var into small hash feature (but hashing feature are uninterpretable)
    * Bin counting, popular with linear models as well as trees. Rather than using the value of the categorical variable as the feature, use the conditional probability of the target under that value. In other words, instead of encoding the identity of the categorical value, we compute the association statistics between that value and the target that we wish to predict
- **Ordered data**
  - how measure de force to pass between each categorie ?
    * linear
    * quadratic

2.1.4 Date Time : Lubridate package

- Use as.POSIXct() and UTC (universal coordinated time)in time zone.
- create new variables : weekend (0/1), bankholiday (0/1), ...

2.2 Missing Value

- Do nothing
- remove
- impute
  - by mean : doesn't impact analysis
  - by singular value decomposition : approximate true value
  - by regression :approximate true value
  - Check lien 5 methode impute missing value

2.3 Outlier Detection

2.4 Sampling and resampling

Modern statistical methods assume that the underlying data comes from a random distribution. The performance measurements of models derived from data are also subject to random noise. the sample can be generalized for the population with statistical confidence. Is an approximatation.

Weak law of large numbers : $\bar{X_n} => \mu$
Central limit theorem : distribution standardis? tend vers une normale asymptotiquement

- **model sampling** : population data is already collected and you want to reduce time and the computational cost of analysis, along with improve the inference of your models
- **survey sampling** : create a sample design and then survey the population only to collect sample to save data collection costs.

Type of sampling methods :

- Boostrap sampling : sampling with replacement
- Jackknife = leave one out sampling + calculate average of the estimation
- Vfold crossvalidation : Resampling methods that can generate V different versions of the training set (same size) that can be used to evaluate model on test set. Each of the V assessment sets contains 1/V of the training set and each of these exclude different data points. Suppose V = 10, then there are 10 different versions of 90% of the data and also 10 versions of the remaining 10% for each corresponding resample. in the end, there are V estimates of performance for the model and each was calculated on a different assessment set. The cross-validation estimate of performance is computed by averaging the V individual metrics.
- Monte Carlo : Produces splits that are likely to contain overlap. For each resample, a random sample is taken with proportion of the training set going into the analysis set and the remaining samples allocated to the assessment set
- bootstrap : A bootstrap resample of the data is defined to be a simple random sample that is the same size as the training set where the data are sampled with replacement

2.5 variables selections

How do we cleanly separate the signal from the noise?

First Filter

- Na filter : column with to many NA
- Variance filter : Column with not enought variance to explain dataset
- corrélation filter : e will remove predictors that are highly correlated ($r2 > 0.9$) with other predictors. see corrplot
- Variance treshold : Variable with high variability also have higher information in them. We remove all variables havant variance less than a treshold.

2.5.1 Filter methods :

Select variables sans modélisation. Methode univariée. Order feature by importance. Methode robust contre overfitting mais peut selectionner variables redondantes. It is best to do prefiltering conservatively, so as not to inadvertently eliminate useful features before they even make it to the model training step

- Chi square test
- Correlation coefficients
- information gain metrics
- fisher score
- variance treshold

2.5.2 Wrapper Methods:

Test differentes combinaisons de feature selon crit?re de performance. Predictive model is used to evaluate the set of feature by accurancy metric. Méthode efficace pour la mod?lisation. Peut causé de l'overfitting.

- forward/backward selection
- recursive feature elimation algorithm
- ...

2.5.3 Embedded Methods :

Next step to wrapper methods. Introduce a penalty factor to the evaluation criteria of the model to bias the model toward lower complexity. Balance between complexity and accurancy. Less computationally expensive than Wrapper. Less prone to overfitting. These methods perform feature selection as part of the model training process

- Lasso
- Ridge regression
- …
- Decision tree
- Gradiant descent methods

2.5.4 Dimension reduction :

See unsuppervized section

- PCA see unsupervised analysis : Due to the orthogonality constraint in the objective function, PCA transformation produces a nice side effect: the transformed features are no longer correlated.
- svd
- k-means as a featurization procedure, a data point can be represented by its cluster membership

2.6 Example

2.6.1 Credit risk modeling

- **Feature ranking**
  - Fit logistic model
  - Calculate Gini coefficient
  - rearrange variables ? combine, weighted sums, etc
  - Need to understand variable individually ? use Filtering method
  - data dirty ? detect outlier
  - Data selection? use first ranking, forward selection and last Embedded method. Compare with crit?rion (misclassi, MSE, AIC, etc)
  - improve performance? bootstrap : subsample your data et redo analysis

```r
### Data Prep ###
#################
library(MLmetrics)

data = get(load("C:/Users/007/Desktop/Data science with R/R/Dataset/LoanDefaultPred.RData"))

#Create the default variable
data[,"default"]=ifelse(data$loss ==0, 0,1)
print(table(data$default)*100/nrow(data))
```

```
##
##      0      1
## 90.635  9.365
```

```r
# Without prior kwowledge : if more than 30 variable is continuous
continuous <-character()
categorical <-character()
i = names(data)[1]
```

```
p<-1
q<-1

for (i in names(data)){
unique_levels =length(unique(data[,i]))

  if(i %in% c("id","loss","default")){
next;
      }else if (unique_levels <=30 |is.character(data[,i])){
            categorical[p] <-i
            p=p+1
            data[[i]] <-factor(data[[i]])
  }else{
            continuous[q] <-i
            q=q+1
  }}

cat("\nTotal number of continuous variables in feature set ",length(continuous) -1)
```

```
##
## Total number of continuous variables in feature set  714
```

```
cat("\nTotal number of categorical variable in feature set ",length(categorical) -2)
```

```
##
## Total number of categorical variable in feature set  52
```

```
# Gini coef
performance_metric_gini <-data.frame(feature =character(), Gini_value =numeric())

# for (feature in names(data)){
#     if(feature %in%c("id","loss","default")) {
#         next
#       } else {
# tryCatch(
#   {glm_model <-glm(default ~get(feature),data=data,family=binomial(link="logit"));
#   predicted_values <-predict.glm(glm_model,newdata=data,type="response");
#   Gini_value <-Gini(predicted_values,data$default);
#   performance_metric_gini <-rbind(performance_metric_gini,cbind(feature,Gini_value));},error=function
# }
# }
#
# saveRDS(performance_metric_gini, "performance_metric_gini.rds")
performance_metric_gini <- readRDS("./save/performance_metric_gini.rds")

performance_metric_gini$Gini_value <-as.numeric(as.character(performance_metric_gini$Gini_value))

Ranked_Features <-performance_metric_gini[order(-performance_metric_gini$Gini_value),]
head(Ranked_Features)
```

```
##     feature Gini_value
## 389    f404  0.2579189
## 710    f766  0.2578312
## 585    f630  0.2415352
## 584    f629  0.2354368
## 321    f333  0.2352707
## 56      f64  0.2348747
```

```
# Note  : When you are running loops over large datasets, it is possible that the loop might stop due t

###################################################
### Try logistic regression with top 5 features ###
###################################################

glm_model <-glm(default ~f766 +f404 +f629 +f630 +f281 +f322,data=data,family=binomial(link="logit"))
predicted_values <-predict.glm(glm_model,newdata=data,type="response")
Gini_value <-Gini(predicted_values,data$default)
summary(glm_model)
```

```
##
## Call:
## glm(formula = default ~ f766 + f404 + f629 + f630 + f281 + f322,
##     family = binomial(link = "logit"), data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.6928  -0.4946  -0.4102  -0.3329   3.0013
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.502550   4.939282  -0.304   0.7610
## f766        -0.010228   4.916519  -0.002   0.9983
## f404        -1.395602   4.908606  -0.284   0.7762
## f629        -0.306456   0.172632  -1.775   0.0759 .
## f630        -0.165047   0.128300  -1.286   0.1983
## f281         0.007759   0.019386   0.400   0.6890
## f322         0.264196   0.128472   2.056   0.0397 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 12415  on 19938  degrees of freedom
## Residual deviance: 12040  on 19932  degrees of freedom
##   (61 observations deleted due to missingness)
## AIC: 12054
##
## Number of Fisher Scoring iterations: 5
```

```
Gini_value
```

```
## [1] 0.2697868
```

```
# Every features aren't always significant. Indication that features themselves are highly correlated. 
# Variable ranking method is univariate and lead to the selection of a redundant variables.

top_6_feature <-data.frame(data$f766,data$f404,data$f629,data$f630,data$f281,data$f322)
cor(top_6_feature, use="complete")
```

```
##            data.f766  data.f404  data.f629  data.f630  data.f281
## data.f766  1.0000000  0.9996754  0.6777553  0.6378040  0.8205665
## data.f404  0.9996754  1.0000000  0.6774434  0.6374457  0.8204153
## data.f629  0.6777553  0.6774434  1.0000000  0.9155376  0.6628148
## data.f630  0.6378040  0.6374457  0.9155376  1.0000000  0.6202698
## data.f281  0.8205665  0.8204153  0.6628148  0.6202698  1.0000000
## data.f322 -0.7706228 -0.7707861 -0.5450001 -0.5048133 -0.7371242
##             data.f322
## data.f766 -0.7706228
## data.f404 -0.7707861
## data.f629 -0.5450001
## data.f630 -0.5048133
## data.f281 -0.7371242
## data.f322  1.0000000
```

2.6.2 variance treshold approach

```
# Attention, les variables ne sont pas standardisées, on ne peut pas les comparer directement. On utili

# Calculate CV
coefficient_of_variance <-data.frame(feature =character(), cov =numeric())

for (feature in names(data)){
  if(feature %in%c("id","loss","default")){next
  }else if(feature %in% continuous){
    tryCatch({
      cov <-abs(sd(data[[feature]], na.rm =TRUE)/mean(data[[feature]],na.rm =TRUE));
      if(cov !=Inf){
coefficient_of_variance <-rbind(coefficient_of_variance,cbind(feature, cov));
      } else {next}
                },error=function(e){})
  }else{next}
}

coefficient_of_variance$cov <-as.numeric(as.character(coefficient_of_variance$cov))
Ranked_Features_cov <-coefficient_of_variance[order(-coefficient_of_variance$cov),]

head(Ranked_Features_cov)
```

```
##     feature       cov
## 294    f338 128.05980
## 377    f422 111.93083
## 664    f724  69.64913
## 349    f393  55.39446
## 712    f775  47.64456
## 350    f394  46.68719
```

```
## Logistic model

glm_model <-glm(default ~f338 +f422 +f724 +f636 +f775 +f723,data=data, family=binomial(link="logit"));
predicted_values <-predict.glm(glm_model,newdata=data,type="response")
Gini_value <-Gini(predicted_values,data$default)

cat("The Gini Coefficient for the fitted model is ",Gini_value);
```

```
## The Gini Coefficient for the fitted model is  0.1465253
```

Contrairement au Ranking avec Gini, les variables ne sont pas dominés par leur structure de correlation. Mais les variables ne sont pas toutes significatives individuellement et le coef GINI pas particuliérement amélioré. Avec variance treshlod on espére selectionné des variables indépendantes

2.7 Method Summary

	Variable quanti	Variable quali
Graph	Time series, barplot, boxplot, histographe, QQplot, scaterplot	barplot, boxplot
Test	t-test sur la moyenne, chi2 sur la variance, test normalité, corrélation, test F variance, test de levene	test proportion, test ajustement, test indépendance
Modélisation	Régression linéaire	régression logistique, analyse discriminante, abre décision

- **Parametric** : assume thaht sample data is drawn from a known probabilité distribution based on fixed set of parameters. For instance, linear regression assumes normal distribution, whereas logistic assumes binomial distribution, etc. This assumption allows the methods to be applied to small datasets as well.
  - involve a two-step model-based approach : Chose model (ex : linear) and estimate (ex: ols)
  - reduce the probleme of model estimation to a probleme of parameter estimation
  - but if the chosen model is too far from the true f, then the estimate will be poor
- **Non parametric** : not assume any probabilty distribution or prior. Contruct empirical distributions from data. (= Kernel regression, NPMR)

Models can also be evaluated in terms of variance and bias.

- A model has high variance if small changes to the underlying data used to estimate the parameters cause a sizable change in those parameters (or in the structure of the model)
- Model bias reflects the ability of a model to conform to the underlying theoretical structure of the data. A low bias model is one that can be highly flexible and has the capacity to fit a variety of different shapes and patterns. A high bias model would be unable to estimate values close to their true theoretical counterparts. Linear methods often have high bias since, without modification, cannot describe nonlinear patterns in the predictor variables. Tree-based models, support vector machines, neural networks, and others can be very adaptable to the data and have low bias.

2.8 tips

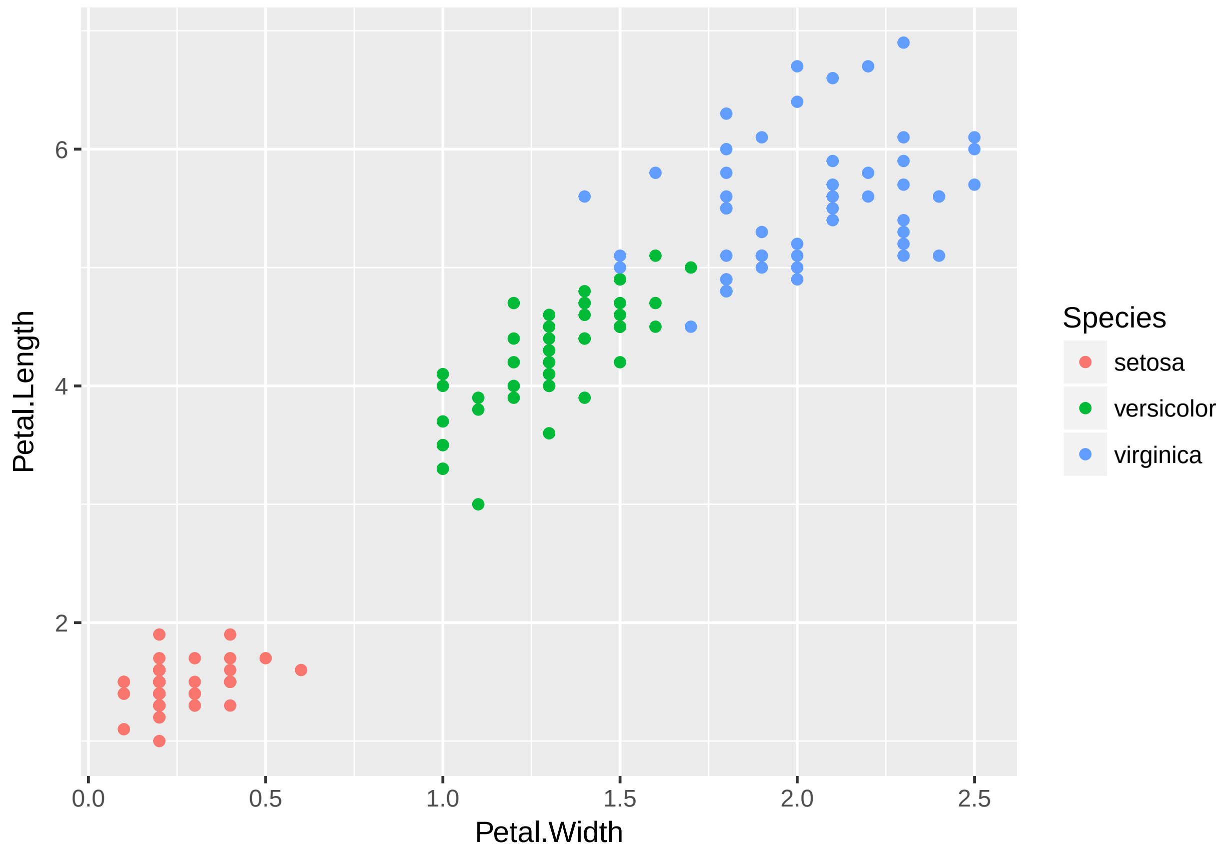
- Tidyverse package
- Given below are some of the rare feature engineering tricks implemented in the winning solutions of several data science competitions. - Transform data to Image - Meta-leaks - Representation learning features Mean encodings - Transforming target variable

Chapter 3

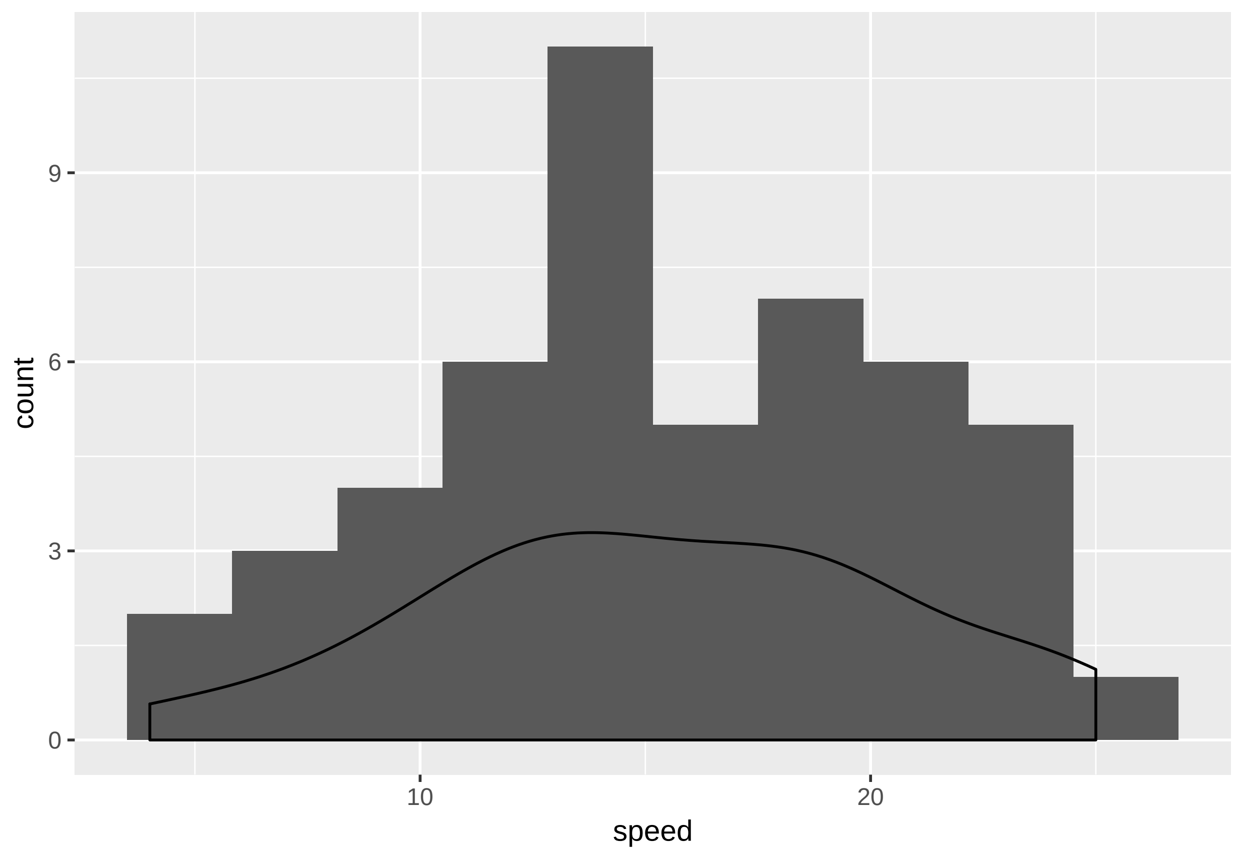
Data visualization

3.1 Descriptive

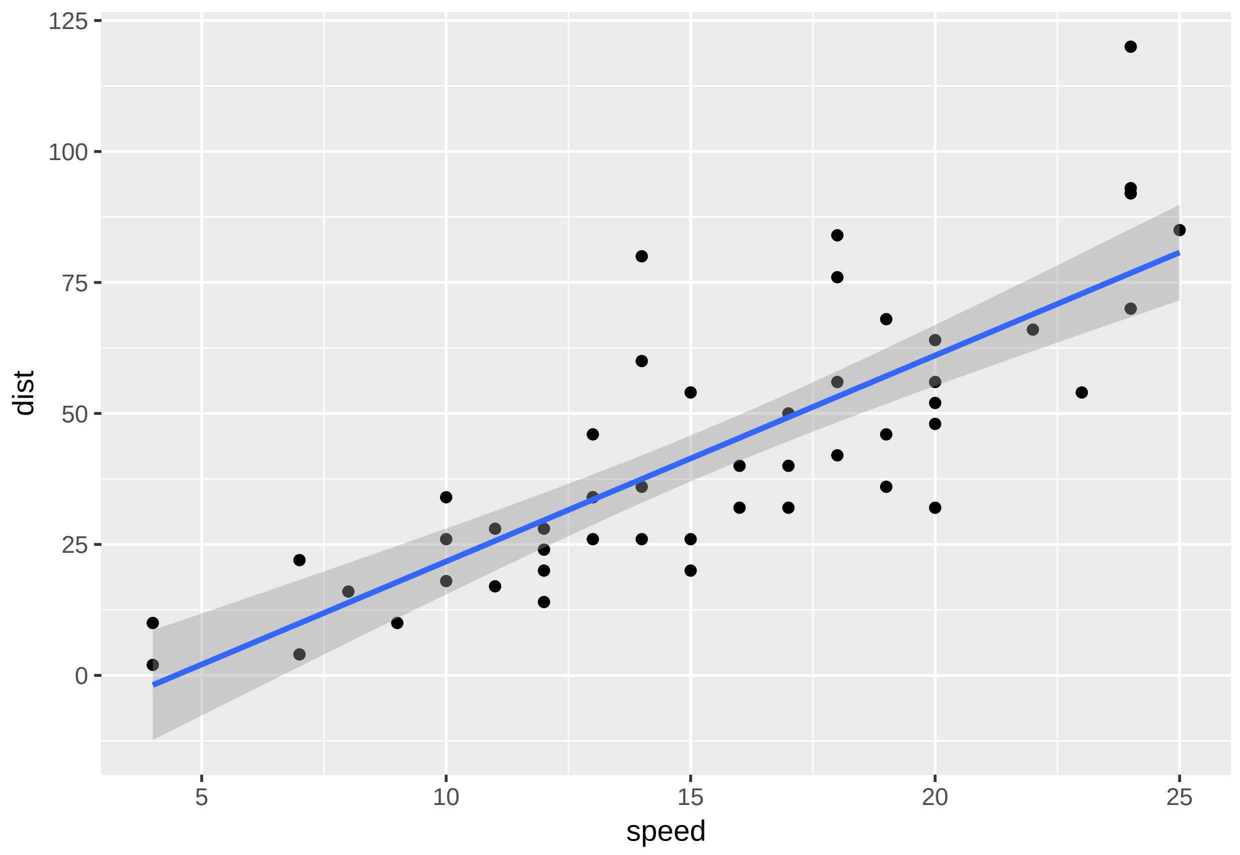
```
#change theme => + theme()

iris %>% qplot(Petal.Width, Petal.Length , color = Species, data = .)
```



```
cars %>% ggplot(aes(x = speed, y = ..count..)) + geom_histogram(bins = 10) + geom_density()
```



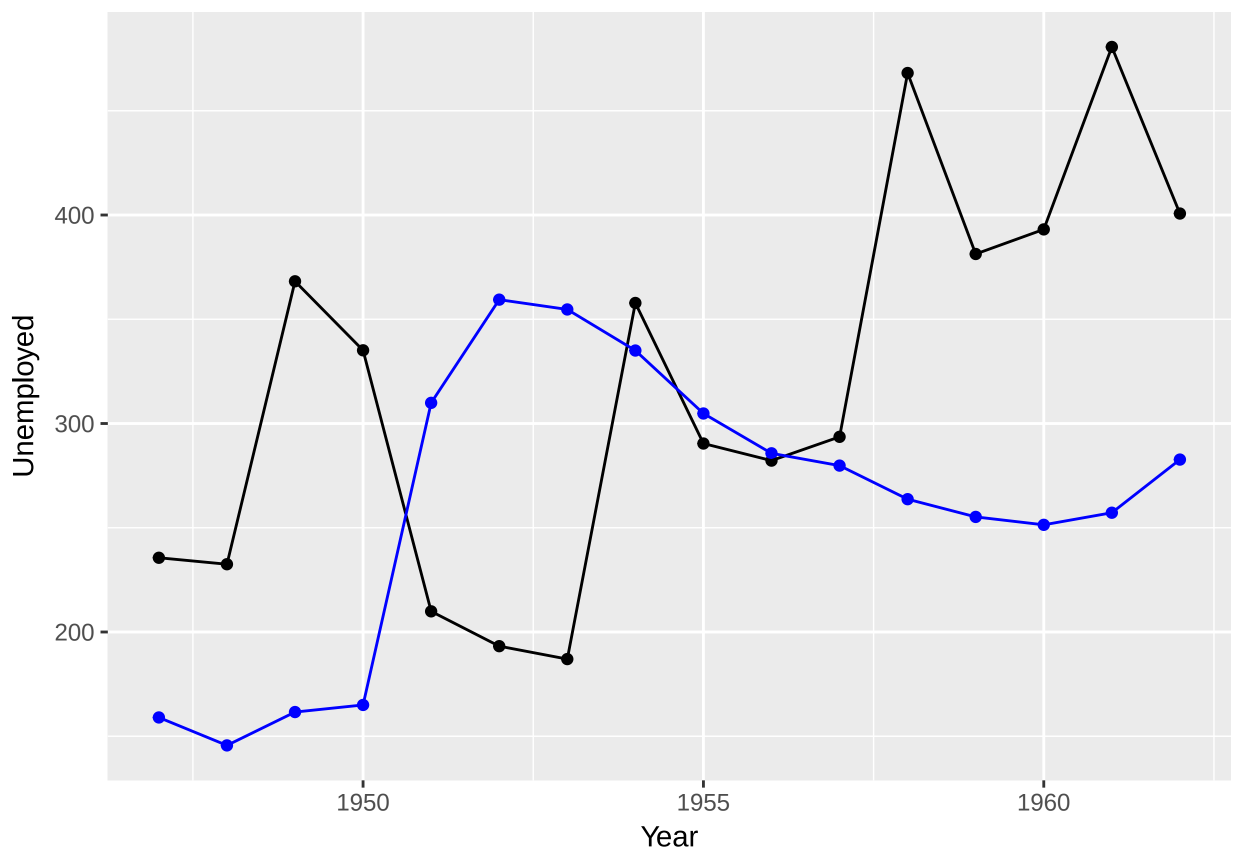
```r
cars %>% ggplot(aes(x = speed, y = dist)) + geom_point() + geom_smooth(method = "lm")
```



```
# If non linear smooth :  method = 'loess'
```

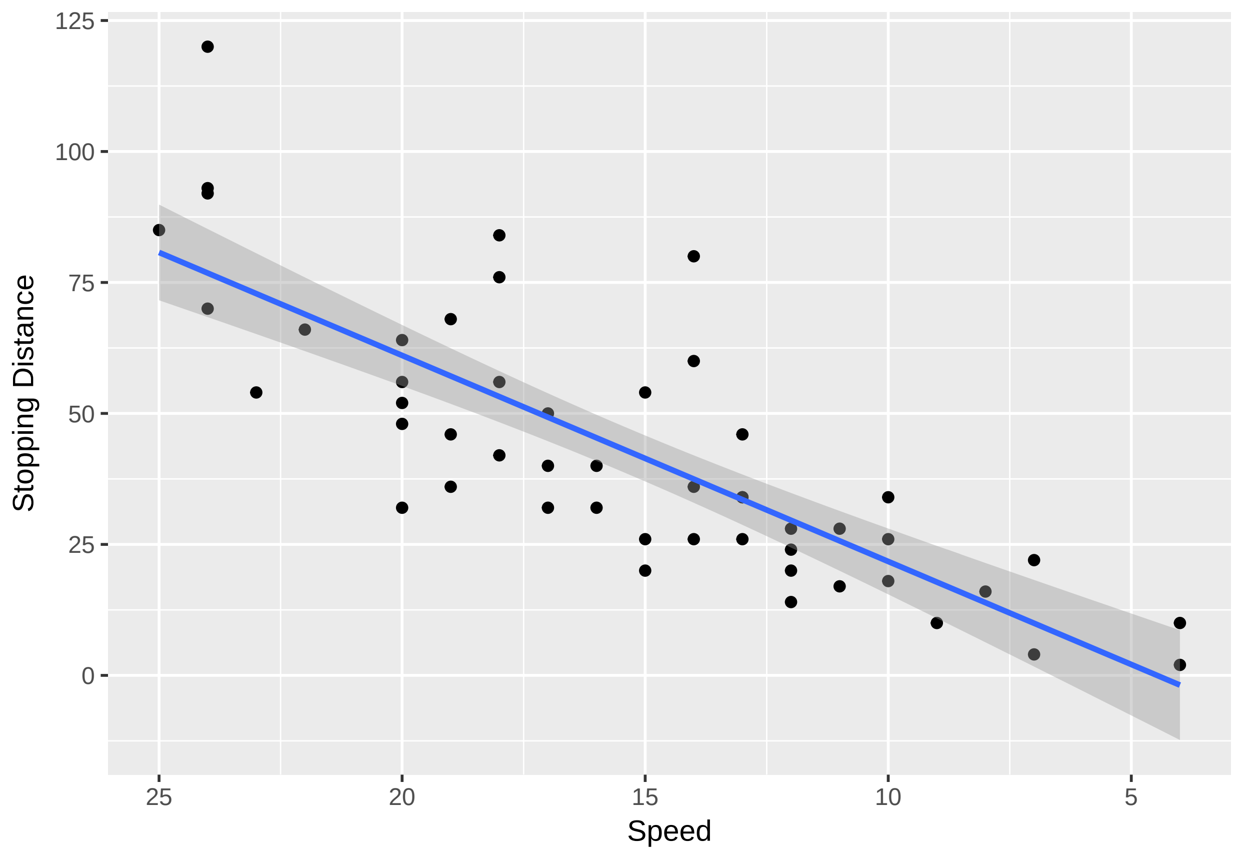
- **Multiple line**

```
longley %>% ggplot(aes(x = Year)) +
geom_point(aes(y = Unemployed)) +
geom_point(aes(y = Armed.Forces), color = "blue") +
geom_line(aes(y = Unemployed)) +
geom_line(aes(y = Armed.Forces), color = "blue")
```



- **Scaling**

```
cars %>% ggplot(aes(x = speed, y = dist)) +
geom_point() + geom_smooth(method = "lm") +
scale_x_reverse("Speed") +
scale_y_continuous("Stopping Distance")
```



```
iris %>% ggplot(aes(x = Species, y = Petal.Length)) +
geom_boxplot() + geom_jitter(width = 0.1, height = 0.1) +
scale_x_discrete(labels = c("setosa" = "Setosa",
"versicolor" = "Versicolor",
"virginica" = "Virginica"))
```