Statistical learning and deep learning: theoretical background and hands-on sessions

Tidymodels

S. Biffani & F. Biscarini

IBBA/CNR

27 gennaio, 2023

Tidymodels 1

- collezione di librerie per implementare modelli predittivi
- Obiettivo:
  - uniformare le molte procedure/librerie disponibili
  - dare la possibilità di creare delle *workflow*
  - utilizzare l'approccio `tidyverse` (operazioni in sequenza)

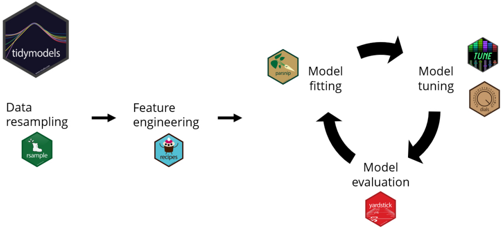


Figura 1: un pizzico di Tidymodels

Tidymodels 2

Overview of *tidymodels* Basics

Package	Step	Functions
rsample	**1.** Split into testing and training sets	initial_split() training() testing()
recipes	**2.** Create recipe + assign variable roles	recipe() update_role()
parsnip	**3.** Specify model, engine, and mode	parsnip function for specifying model (ex. decision_tree()) (https://www.tidymodels.org/find/parsnip/) set_engine() set_mode()
WORKFLOWS	**4.** Create workflow, add recipe, add model	workflow() add_recipe() add_model()
parsnip	**5.** Fit workflow	fit()
parsnip	**6.** Get predictions	predict()
yardstick	**7.** Use predictions to get performance metrics	rmse() (continuous outcome) accuracy() (categorical outcome) metrics() (either type of outcome)

Figura 2: un pizzico di Tidymodels

Tidymodels 3

I passi principali:

1. Data Resampling and Feature Engineering: `rsample`, `recipes`
2. Model Fitting and Tuning: `parsnip`, `tune`, `dials`
3. Model Evaluation: `yardstick`

rsample: Crea training e test datasets 1

```
library(ISLR2)

library(tidymodels)
tidymodels_prefer(quiet=F)

?initial_split()
```

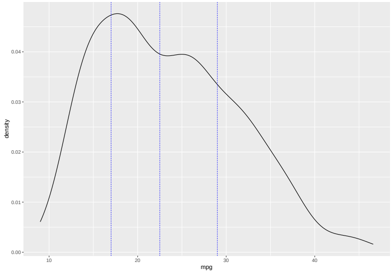
1. Create a data split object

```
set.seed(3)
auto_split <- initial_split(
  Auto,
  prop = 0.5,
  # stratification by outcome variable
  #strata = mpg
)
```

rsample: Crea training e test datasets 2

strata= stratificazione del campionamento (vale anche per le variabili continue)

```
Auto %>%
  mutate(Q=ntile(mpg,4))%>%
  group_by(Q) %>%
  mutate(Q1=ifelse(Q<4,max(mpg),min(mpg)))%>%
  ggplot(.,aes(mpg))+
  geom_line(stat = "density")+
  geom_vline(aes(xintercept=Q1),
             color="blue",
             linetype="dashed",
             size=.2)
```




```
set.seed(3)
auto_split <- initial_split(
  Auto,
  prop = 0.5,
  # stratification by outcome variable
  strata = mpg
)
```

rsample: Crea training e test datasets 3

1. Create the training data

```
auto_training <- auto_split %>%
  training()
```

2. Create the test data

```
auto_test <- auto_split %>%
  testing()
```

3. Check number of rows in each dataset

```
nrow(auto_training)
```

```
## [1] 194
```

Specify a model with parsnip

1. Specify model type (e.g. linear regression, random forest . . .)
   - linear_reg()
   - rand_forest()
2. Specify engine (i.e. package implementation of algorithm)
   - set_engine("some package's implementation")
3. declare mode (e.g. classification vs linear regression)
   - use this when model can do both classification & regression
     - set_mode("regression")
     - set_mode("classification")

All available models are listed at here.

Fit a model 1

```r
lm_spec <- linear_reg() %>%
  set_engine(engine = "lm") %>%
  set_mode(mode = "regression")

m <- fit(
  lm_spec, # parsnip model spec
  mpg ~ horsepower, # formula
  auto_training # data frame
)
```

```r
m
```

```
## parsnip model object
##
##
## Call:
## stats::lm(formula = mpg ~ horsepower,
##
## Coefficients:
## (Intercept)   horsepower
##     39.7476      -0.1532
```

Fit a model 2

obtain the estimated parameters

```
tidy(m) %>%
  knitr::kable(.,digits = 2,
               caption = 'Parametri Stimati')
```

Tabella 1: Parametri Stimati

term	estimate	std.error	statistic	p.value
(Intercept)	39.75	1.00	39.66	0
horsepower	-0.15	0.01	-17.32	0

generate predictions

```
# using fitting model to get prediction
lm_pred <- m %>%
  predict(new_data = auto_test)
```

```
head(lm_pred) %>%
  knitr::kable(caption = 'Predizioni',digits=.1)
```

Tabella 2: Predizioni

.pred
20
17
7
17
25
25

Fit a model 3

bind result

```
auto_test_res <- auto_test %>%
  select(mpg, horsepower) %>%
  bind_cols(lm_pred)

head(auto_test_res) %>%
  knitr::kable(caption='Stima e Variabile Originale',
               digits=c(0,0,1))
```

Tabella 3: Stima e Variabile Originale

mpg	horsepower	.pred
18	130	19.8
16	150	16.8
14	215	6.8
15	150	16.8
24	95	25.2
22	95	25.2

Measure the model performance with yardstick::rmse() 1

- Residui: valori predetti - valori originali $\hat{y}_i - y_i$
- Mean Absolute Error: $\frac{1}{n}\sum_i^n |(\hat{y}_i - y_i)|$
- Root Mean Absolute Error: $\frac{1}{n}\sqrt{\sum_1^n (\hat{y}_i - y_i)^2}$

sfrutto l'oggetto creato precedentemente

```
auto_test_res %>%
  rmse(truth = mpg, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        4.86
```

qui troviamo tutte le possibile *metriche*

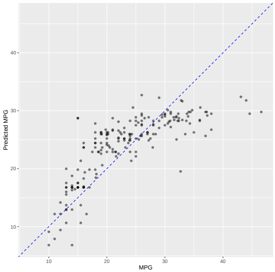
Measure the model performance with yardstick::rsq() 2

R-2

```
auto_test_res %>%
  rsq(truth = mpg, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.606
```

```
auto_test_res %>%
  ggplot(aes(x = mpg, y = .pred)) +
  geom_point(alpha = .5) +
  geom_abline(color = "blue", linetype = 2) +
  coord_obs_pred() +
  labs(x = "MPG", y = "Predicted MPG")
```



Cross Validation 1

- `loo_cv()`: leave-one-out (deprecated)
  - Leave-one-out methods are deficient compared to almost any other method. For anything but pathologically small samples, LOO is computationally excessive, and it may not have good statistical properties. Although the rsample package contains a loo_cv() function, these objects are not generally integrated into the broader tidymodels frameworks.
- `vfold_cv()`: k-fold
- `bootstraps`: test set con replacement

Cross Validation 2

```
vfold_cv(data, v = 10, repeats = 1, strata = NULL, breaks = 4, ...)
```

```
set.seed(123)

folds <- vfold_cv(Auto, v = 10,strata = mpg)
##
fit_resamples(lm_spec,mpg ~ horsepower,resamples=folds)-> res

res %>%
  collect_metrics() %>%
  knitr::kable(caption='Statistiche Cross-Validation')
```

Tabella 4: Statistiche Cross-Validation

.metric	.estimator	mean	n	std_err	.config
rmse	standard	4.889087	10	0.1409716	Preprocessor1_Model1
rsq	standard	0.610207	10	0.0208169	Preprocessor1_Model1

Cross Validation 3

- repliche di CV per diminuire la varianza dell'errore

```
vfold_cv(data, v = 10, repeats = 1, strata =
NULL, breaks = 4, ...)
```

```
set.seed(123)

resall<-NULL
for (i in seq(1:5)){

  folds <- vfold_cv(Auto, v = 10,
                    strata = mpg,
                    repeats =i)
  fit_resamples(lm_spec,
                mpg ~ horsepower,
                resamples=folds)-> res
  res %>%
  collect_metrics()-> res1
  bind_rows(resall,res1)-> resall
}
```

```
resall %>%
  filter(.metric=='rmse') %>%
  mutate(Repliche=n/10) %>%
  select(Repliche,std_err) %>%
  ggplot(aes(Repliche,std_err))+
  geom_line()
```