Modeling - Seoul Bike Sharing Data

Istruzioni: lanciare tutti i code chunks con Ctrl+Alt+R in modo da evitare di lanciare i codici che stimano i modelli (i modelli sono già stati salvati nella cartella models)

Carico pacchetti:

Carico dati:

Modelli "all features"

Random forest 1

Random forest stimata per feature selection iniziale

Codice per stima modello (non necessario eseguire)

Save/Load model:

```
bike_rf1 <- readRDS("models/bike_rf1.rda")
```

```
paste("Validation RMSE: ", RMSE(round(predict(bike_rf1, bike_train)$predictions), bike_train$rented_bike
```

RMSE

```
## [1] "Validation RMSE:  86.7527482065492"
```

```
paste("Validation RMSE: ", RMSE(round(predict(bike_rf1, bike_valid)$predictions), bike_valid$rented_bike
```

```
## [1] "Validation RMSE:  194.680474112414"
```

```
paste("Testing RMSE: ", RMSE(round(predict(bike_rf1, bike_test)$predictions), bike_test$rented_bike_cou
```

```
## [1] "Testing RMSE:  198.944113443883"
```

Regressione lineare

Stima modello:

```
lm1 <- lm(rented_bike_count ~ ., bike_train_dummy)
```

Riassunto modello:

```
summary(lm1)
```

```
##
## Call:
## lm(formula = rented_bike_count ~ ., data = bike_train_dummy)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1283.86  -264.38   -49.44   203.05  1969.80
##
## Coefficients: (4 not defined because of singularities)
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -315.07194  117.92716  -2.672 0.007565 **
## hour                    26.67204    0.83715  31.861  < 2e-16 ***
## temperature             19.32989    4.21261   4.589 4.55e-06 ***
## humidity               -10.25038    1.17485  -8.725  < 2e-16 ***
## wind_speed              21.18798    5.74716   3.687 0.000229 ***
## visibility               0.05535    0.01300   4.258 2.10e-05 ***
## dew_point_temperature   12.14990    4.42625   2.745 0.006069 **
## solar_radiation        -84.84009    8.57337  -9.896  < 2e-16 ***
## rainfall               -53.06086    4.53112 -11.710  < 2e-16 ***
## snowfall                41.16752   12.52006   3.288 0.001014 **
## seasons_Spring         -65.30545   31.25898  -2.089 0.036732 *
## seasons_Summer         196.00481   27.02258   7.253 4.56e-13 ***
## seasons_Winter        -217.05726   46.03734  -4.715 2.47e-06 ***
## holiday_No.Holiday     151.05293   24.06710   6.276 3.70e-10 ***
## functioning_day_Yes    965.91655   30.58170  31.585  < 2e-16 ***
## weekday_Mon            -63.96823   19.35897  -3.304 0.000957 ***
## weekday_Sat            -77.65615   19.27358  -4.029 5.66e-05 ***
## weekday_Sun           -137.76117   19.35961  -7.116 1.24e-12 ***
## weekday_Thu            -35.18964   19.30786  -1.823 0.068418 .
## weekday_Tue            -41.57479   19.29923  -2.154 0.031261 *
## weekday_Wed            -18.75070   19.36517  -0.968 0.332947
## month_Aug             -554.20969   28.43136 -19.493  < 2e-16 ***
## month_Dec               61.93244   25.60520   2.419 0.015602 *
## month_Feb              -41.77876   26.24409  -1.592 0.111451
## month_Jan                     NA         NA      NA       NA
## month_Jul             -401.85129   27.19479 -14.777  < 2e-16 ***
## month_Jun                     NA         NA      NA       NA
## month_Mar              -73.45922   26.16311  -2.808 0.005004 **
## month_May              138.00549   26.82785   5.144 2.77e-07 ***
## month_Nov               78.94940   34.80561   2.268 0.023345 *
## month_Oct              170.50512   28.29167   6.027 1.77e-09 ***
## month_Sep                     NA         NA      NA       NA
## weekend_Yes                   NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 410.5 on 6278 degrees of freedom
## Multiple R-squared:  0.5947, Adjusted R-squared:  0.5929
## F-statistic:   329 on 28 and 6278 DF,  p-value: < 2.2e-16
```

Investighiamo le variabili per cui i coefficienti stimati della regressione lineare sono NA

Random forest dummy default

Modello applicato a dati con variabili dummy.

(Modello di default senza hyperparameter tuning)

Save/Load model:

```
bike_dummy_rf <- readRDS("models/bike_dummy_rf.rda")
```

```


default_train_rmse <- RMSE(round(predict(bike_dummy_rf, bike_train_dummy)$predictions),
                           bike_train_dummy$rented_bike_count)
default_valid_rmse <-
  RMSE(round(predict(bike_dummy_rf,
                     bike_valid_dummy)$predictions),
                           bike_valid_dummy$rented_bike_count)
default_test_rmse <-
  RMSE(round(predict(bike_dummy_rf, bike_test_dummy)$predictions),
                           bike_test_dummy$rented_bike_count)


paste("Training RMSE:", default_train_rmse)
```

RMSE

```
## [1] "Training RMSE: 85.0895861573404"
```

```
paste("Validation RMSE: ", default_valid_rmse)
```

```
## [1] "Validation RMSE:  193.3215407959"
```

```
paste("Testing RMSE: ", default_test_rmse)
```

```
## [1] "Testing RMSE:  195.569219548228"
```

Regressione lineare 2

Regressione lineare stimata escludendo le variabili che danno coefficienti stimati NA

Creiamo nuovo dataframe per training e test set:

```
bike_train_dummy2 <- bike_train_dummy %>%
  select(-names(which(is.na(lm1$coefficients))))

bike_valid_dummy2 <- bike_valid_dummy %>%
  select(-names(which(is.na(lm1$coefficients))))

bike_test_dummy2 <- bike_test_dummy %>%
  select(-names(which(is.na(lm1$coefficients))))
```

Stima regressione lineare:

```r
lm2 <- lm(rented_bike_count ~ ., bike_train_dummy2)
```

Riassunto modello:

```r
summary(lm2)
```

```
##
## Call:
## lm(formula = rented_bike_count ~ ., data = bike_train_dummy2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1283.86  -264.38   -49.44   203.05  1969.80
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -315.07194  117.92716  -2.672 0.007565 **
## hour                    26.67204    0.83715  31.861  < 2e-16 ***
## temperature             19.32989    4.21261   4.589 4.55e-06 ***
## humidity               -10.25038    1.17485  -8.725  < 2e-16 ***
## wind_speed              21.18798    5.74716   3.687 0.000229 ***
## visibility               0.05535    0.01300   4.258 2.10e-05 ***
## dew_point_temperature   12.14990    4.42625   2.745 0.006069 **
## solar_radiation        -84.84009    8.57337  -9.896  < 2e-16 ***
## rainfall               -53.06086    4.53112 -11.710  < 2e-16 ***
## snowfall                41.16752   12.52006   3.288 0.001014 **
## seasons_Spring         -65.30545   31.25898  -2.089 0.036732 *
## seasons_Summer         196.00481   27.02258   7.253 4.56e-13 ***
## seasons_Winter        -217.05726   46.03734  -4.715 2.47e-06 ***
## holiday_No.Holiday     151.05293   24.06710   6.276 3.70e-10 ***
## functioning_day_Yes    965.91655   30.58170  31.585  < 2e-16 ***
## weekday_Mon            -63.96823   19.35897  -3.304 0.000957 ***
## weekday_Sat            -77.65615   19.27358  -4.029 5.66e-05 ***
## weekday_Sun           -137.76117   19.35961  -7.116 1.24e-12 ***
## weekday_Thu            -35.18964   19.30786  -1.823 0.068418 .
## weekday_Tue            -41.57479   19.29923  -2.154 0.031261 *
## weekday_Wed            -18.75070   19.36517  -0.968 0.332947
## month_Aug             -554.20969   28.43136 -19.493  < 2e-16 ***
## month_Dec               61.93244   25.60520   2.419 0.015602 *
## month_Feb              -41.77876   26.24409  -1.592 0.111451
## month_Jul             -401.85129   27.19479 -14.777  < 2e-16 ***
## month_Mar              -73.45922   26.16311  -2.808 0.005004 **
## month_May              138.00549   26.82785   5.144 2.77e-07 ***
## month_Nov               78.94940   34.80561   2.268 0.023345 *
## month_Oct              170.50512   28.29167   6.027 1.77e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 410.5 on 6278 degrees of freedom
## Multiple R-squared:  0.5947, Adjusted R-squared:  0.5929
## F-statistic:   329 on 28 and 6278 DF,  p-value: < 2.2e-16
```

```
paste("Training RMSE: ", RMSE(round(predict(lm2, bike_train_dummy2)), bike_train_dummy2$rented_bike_cou
```

RMSE

```
## [1] "Training RMSE:  409.525471080192"
```

```
paste("Validation RMSE: ", RMSE(round(predict(lm2, bike_valid_dummy2)), bike_valid_dummy2$rented_bike_c
```

```
## [1] "Validation RMSE:  414.073145796431"
```

```
paste("Testing RMSE: ", RMSE(round(predict(lm2, bike_test_dummy2)), bike_test_dummy2$rented_bike_count)
```

```
## [1] "Testing RMSE:  416.657060619862"
```

Random forest 4

```
n_features <- length(bike_train_dummy) - 1

hyper_grid <- expand.grid(
  num.trees = c(100, n_features * 10, 500),
  mtry = floor(n_features * c(.05, .15, .25, .333, .4)),
  min.node.size = c(1, 3, 5, 10),
  replace = c(TRUE, FALSE),
  sample.fraction = c(.5, .63, .8),
  train_rmse = NA,
  valid_rmse = NA
)

# execute full cartesian grid search
for(i in seq_len(nrow(hyper_grid))) {
  # fit model for ith hyperparameter combination
  fit <- ranger(
    formula         = rented_bike_count ~ .,
    data            = bike_train_dummy,
    num.trees       = hyper_grid$num.trees[i],
    mtry            = hyper_grid$mtry[i],
    min.node.size   = hyper_grid$min.node.size[i],
    replace         = hyper_grid$replace[i],
    sample.fraction = hyper_grid$sample.fraction[i],
    verbose         = FALSE,
    respect.unordered.factors = 'order'
  )
  # export OOB error
  hyper_grid$train_rmse[i] <- sqrt(fit$prediction.error)

  pred <- round(predict(fit, bike_valid_dummy)$predictions)

  hyper_grid$valid_rmse[i] <- RMSE(pred, bike_valid_dummy$rented_bike_count)
}
```

Hyperparameter tuning Carica hyperparameter grid da csv:

```
hyper_grid <- read_csv("models/rf_hyper_grid.csv")[-1]
```

```
## Rows: 360 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (7): ...1, num.trees, mtry, min.node.size, sample.fraction, train_rmse, ...
## lgl (1): replace
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# assess top 10 models
hyper_grid %>%
  arrange(valid_rmse) %>%
  mutate(
    train_perc_gain = (default_train_rmse - train_rmse) /
      default_train_rmse * 100,
    valid_perc_gain = (default_valid_rmse - valid_rmse) /
      default_valid_rmse * 100) %>%
  head(10)
```

```
## # A tibble: 10 x 9
##    num.trees  mtry min.node.size replace sample.fraction train_rmse valid_rmse
##        <dbl> <dbl>         <dbl> <lgl>             <dbl>      <dbl>      <dbl>
##  1       500    12             1 FALSE               0.8       177.       182.
##  2       100    12             1 FALSE               0.8       182.       182.
##  3       320    12             1 FALSE               0.8       177.       182.
##  4       100    12             3 FALSE               0.8       181.       183.
##  5       500    12             3 FALSE               0.8       177.       183.
##  6       320    12             3 FALSE               0.8       179.       184.
##  7       320    12             5 FALSE               0.8       180.       184.
##  8       100    12             5 FALSE               0.8       184.       184.
##  9       500    12             5 FALSE               0.8       180.       186.
## 10       320    10             1 FALSE               0.8       183.       186.
## # ... with 2 more variables: train_perc_gain <dbl>, valid_perc_gain <dbl>
```

Fit Fittiamo modello con gli iperparametri del modello migliore:

```
bike_rf4 <- ranger(
  formula         = rented_bike_count ~ .,
  data            = bike_train_dummy,
  num.trees       = 500,
  mtry            = 12,
  min.node.size   = 1,
  replace         = FALSE,
  sample.fraction = 0.80,
  verbose         = FALSE,
  respect.unordered.factors = 'order',
)
```

Carico modello già stimato:

```
bike_rf4 <- readRDS("models/bike_rf4.rda")
```

```
paste("Training RMSE: ", RMSE(round(predict(bike_rf4, bike_train_dummy)$predictions), bike_train_dummy$r
```

RMSE

```
## [1] "Training RMSE:  35.9265784946732"
```

```
paste("Validation RMSE: ", RMSE(round(predict(bike_rf4, bike_valid_dummy)$predictions), bike_valid_dummy
```

```
## [1] "Validation RMSE:  182.408930031486"
```

```
paste("Testing RMSE: ", RMSE(round(predict(bike_rf4, bike_test_dummy)$predictions), bike_test_dummy$rent
```

```
## [1] "Testing RMSE:  186.777667991315"
```

Multilayer Perceptron

Vedi file "bike_MLP_nb.ipynb"

Carichiamo modello stimato:

Summary del modello:

```
summary(bike_mlp3)
```

```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  dense (Dense)                      (None, 256)                       8448
##  dense_1 (Dense)                    (None, 128)                       32896
##  dense_2 (Dense)                    (None, 32)                        4128
##  dense_3 (Dense)                    (None, 1)                         33
## ================================================================================
## Total params: 45,505
## Trainable params: 45,505
## Non-trainable params: 0
## ________________________________________________________________________________
```

Performance su validation set: RMSE = 171.2723

Modelli “selected features”

Regressione Lineare 3

Stimiamo ora un modello di regressione cosiderando solo alcune delle variabili più importanti individuate attraverso l’importance plot (8 variabili):

```
# no dew_point_temperature per correlazione con temperature
lm3 <- lm(rented_bike_count ~ hour + temperature + humidity + functioning_day_Yes + seasons_Winter + sol
```

```


paste("Training RMSE: ", RMSE(round(predict(lm3, bike_train_dummy2)), bike_train_dummy2$rented_bike_coun
```

RMSE

```
## [1] "Training RMSE:  435.765900375691"
```

```
paste("Validation RMSE: ", RMSE(round(predict(lm3, bike_valid_dummy2)), bike_valid_dummy2$rented_bike_co
```

```
## [1] "Validation RMSE:  437.195561256628"
```

```
paste("Testing RMSE: ", RMSE(round(predict(lm3, bike_test_dummy2)), bike_test_dummy2$rented_bike_count))
```

```
## [1] "Testing RMSE:  445.996639318174"
```

Random forest 5

```


n_features <- 8

hyper_grid <- expand.grid(
  num.trees = c(100, n_features * 10, 500),
  mtry = floor(n_features * c(.05, .15, .25, .333, .4)),
  min.node.size = c(1, 3, 5, 10),
  replace = c(TRUE, FALSE),
  sample.fraction = c(.5, .63, .8),
  train_rmse = NA,
  valid_rmse = NA
)

# execute full cartesian grid search
for(i in seq_len(nrow(hyper_grid))) {
  # fit model for ith hyperparameter combination
  fit <- ranger(
    formula = rented_bike_count ~ hour+temperature+humidity+functioning_day_Yes+seasons_Winter+
            dew_point_temperature+solar_radiation+rainfall,
    data            = bike_train_dummy,
    num.trees       = hyper_grid$num.trees[i],
```

```
    mtry            = hyper_grid$mtry[i],
    min.node.size   = hyper_grid$min.node.size[i],
    replace         = hyper_grid$replace[i],
    sample.fraction = hyper_grid$sample.fraction[i],
    verbose         = FALSE,
    respect.unordered.factors = 'order'
  )
  # export OOB error
  hyper_grid$train_rmse[i] <- sqrt(fit$prediction.error)

  pred <- round(predict(fit, bike_valid_dummy)$predictions)

  hyper_grid$valid_rmse[i] <- RMSE(pred, bike_valid_dummy$rented_bike_count)
}
```

Hyperparameter tuning Carica hyperparameter grid da csv:

```
hyper_grid2 <- read_csv("models/rf_hyper_grid2.csv")
```

```
## Rows: 360 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (6): num.trees, mtry, min.node.size, sample.fraction, train_rmse, valid_...
## lgl (1): replace
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# assess top 10 models
hyper_grid2 %>%
  arrange(valid_rmse) %>%
  mutate(
    train_perc_gain = (default_train_rmse - train_rmse) /
      default_train_rmse * 100,
    valid_perc_gain = (default_valid_rmse - valid_rmse) /
      default_valid_rmse * 100) %>%
  head(10)
```

```
## # A tibble: 10 x 9
##    num.trees  mtry min.node.size replace sample.fraction train_rmse valid_rmse
##        <dbl> <dbl>         <dbl> <lgl>             <dbl>      <dbl>      <dbl>
##  1       500     3             1 FALSE              0.63       241.       245.
##  2       500     3             1 TRUE               0.8        241.       245.
##  3       500     3             5 FALSE              0.8        241.       245.
##  4       100     3            10 FALSE              0.63       242.       245.
##  5        80     3             5 TRUE               0.8        245.       245.
##  6       500     3             3 FALSE              0.5        241.       245.
##  7       500     3            10 FALSE              0.8        241.       245.
##  8       500     3             3 FALSE              0.8        242.       245.
##  9       100     3             3 FALSE              0.8        245.       245.
## 10       500     3             1 FALSE              0.5        241.       245.
## # ... with 2 more variables: train_perc_gain <dbl>, valid_perc_gain <dbl>
```

Stimo il modello migliore:

```
bike_rf5 <- ranger(
    formula = rented_bike_count ~ hour+temperature+humidity+functioning_day_Yes+seasons_Winter+
            dew_point_temperature+solar_radiation+rainfall,
    data            = bike_train_dummy,
    num.trees       = 500,
    mtry            = 3,
    min.node.size   = 1,
    replace         = FALSE,
    sample.fraction = 0.63,
    verbose         = FALSE,
    respect.unordered.factors = 'order'
  )
```

Save/Load model:

```
# saveRDS(bike_rf5, "models/bike_rf5.rda")

bike_rf5 <- readRDS("models/bike_rf5.rda")
```

```
paste("Training RMSE: ", RMSE(round(predict(bike_rf5, bike_train_dummy)$predictions), bike_train_dummy$r
```

RMSE

```
## [1] "Training RMSE:  115.358932760696"
```

```
paste("Validation RMSE: ", RMSE(round(predict(bike_rf5, bike_valid_dummy)$predictions), bike_valid_dummy
```

```
## [1] "Validation RMSE:  244.518497250782"
```

```
paste("Testing RMSE: ", RMSE(round(predict(bike_rf5, bike_test_dummy)$predictions), bike_test_dummy$rent
```

```
## [1] "Testing RMSE:  251.574078798448"
```

MLP

Summary del modello:

```
summary(bike_mlp3)
```

```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  dense (Dense)                      (None, 256)                       8448
##  dense_1 (Dense)                    (None, 128)                       32896
```

```
##  dense_2 (Dense)                    (None, 32)                     4128
##  dense_3 (Dense)                    (None, 1)                      33
## ================================================================================
## Total params: 45,505
## Trainable params: 45,505
## Non-trainable params: 0
## ________________________________________________________________________________
```

MLP

Summary del modello:

```
summary(bike_mlp4)
```

```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                   Param #
## ================================================================================
##  dense (Dense)                      (None, 64)                     576
##  dense_1 (Dense)                    (None, 32)                     2080
##  dense_2 (Dense)                    (None, 8)                      264
##  dense_3 (Dense)                    (None, 1)                      9
## ================================================================================
## Total params: 2,929
## Trainable params: 2,929
## Non-trainable params: 0
## ________________________________________________________________________________
```