xgboost Data-Rate Prediction

```
library(tidyverse)
library(ggplot2)

library(mlr3)
library(mlr3learners)
library(mlr3pipelines)
library(mlr3tuning)
library(mlr3filters)
library(paradox)
```

```
future::plan("multiprocess")
```

```
## Warning: Strategy 'multiprocess' is deprecated in future (>= 1.20.0). Instead,
## explicitly specify either 'multisession' or 'multicore'. In the current R
## session, 'multiprocess' equals 'multisession'.
```

Upload-Rate Prediction

Reading the Data

```
data_dir = "../datasets/"

dataset_ul = read_csv(
  str_c(data_dir, "dataset_ul.csv"),
  col_types = cols(
    drive_id = col_integer(),
    scenario = col_factor(),
    provider = col_factor(),
    ci = col_factor(),
    enodeb = col_factor()
  )
) %>% select(
  drive_id,
  timestamp,
  scenario,
  provider,
  velocity_mps,
  acceleration_mpss,
  rsrp_dbm,
  rsrq_db,
  rssnr_db,
  cqi,
  ta,
  enodeb,
  f_mhz,
  payload_mb,
```

```
  throughput_mbits
) %>% drop_na() %>% rowid_to_column(var="row_id_original")
```

```
dataset_ul_o2 = filter(dataset_ul, provider=="o2")
glimpse(dataset_ul_o2)
```

```
## Rows: 2,039
## Columns: 16
## $ row_id_original   <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15...
## $ drive_id          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ timestamp         <dttm> 2018-12-10 09:08:57, 2018-12-10 09:09:08, 2018-1...
## $ scenario          <fct> campus, campus, campus, campus, campus, campus, c...
## $ provider          <fct> o2, o2, o2, o2, o2, o2, o2, o2, o2, o2, o2, o2, o...
## $ velocity_mps      <dbl> 11.80, 11.49, 7.93, 10.44, 10.92, 12.02, 10.28, 0...
## $ acceleration_mpss <dbl> 0.13, -0.26, 0.23, 0.06, 0.56, 0.09, -1.25, 0.00,...
## $ rsrp_dbm          <dbl> -99, -97, -96, -82, -101, -106, -112, -99, -98, -...
## $ rsrq_db           <dbl> -9, -12, -12, -11, -14, -13, -18, -15, -15, -14, ...
## $ rssnr_db          <dbl> -1, -2, 5, 11, -3, -3, -6, -4, -6, -4, -6, -3, -2...
## $ cqi               <dbl> 8, 9, 5, 15, 6, 6, 3, 4, 7, 4, 4, 5, 6, 5, 1, 4, ...
## $ ta                <dbl> 9, 7, 7, 7, 7, 7, 7, 12, 13, 13, 13, 13, 11, 13, ...
## $ enodeb            <fct> 54016, 52410, 52410, 52410, 52410, 52410, 52410, ...
## $ f_mhz             <dbl> 1750, 1750, 1750, 1750, 1750, 1750, 1750, 880, 88...
## $ payload_mb        <dbl> 1.0, 6.0, 5.0, 7.0, 5.0, 8.0, 9.0, 7.0, 10.0, 2.0...
## $ throughput_mbits  <dbl> 4.66, 3.97, 6.52, 1.37, 0.80, 1.04, 2.34, 4.09, 2...
```

```
dataset_ul_tmobile = filter(dataset_ul, provider=="tmobile")
glimpse(dataset_ul_tmobile)
```

```
## Rows: 2,301
## Columns: 16
## $ row_id_original   <int> 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2...
## $ drive_id          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ timestamp         <dttm> 2018-12-10 09:08:57, 2018-12-10 09:09:07, 2018-1...
## $ scenario          <fct> campus, campus, campus, campus, campus, campus, c...
## $ provider          <fct> tmobile, tmobile, tmobile, tmobile, tmobile, tmob...
## $ velocity_mps      <dbl> 11.83, 11.45, 8.15, 9.42, 10.61, 11.84, 9.75, 0.0...
## $ acceleration_mpss <dbl> 0.03, -0.32, 0.24, 0.43, 0.38, -0.37, -2.15, 0.00...
## $ rsrp_dbm          <dbl> -85, -84, -74, -92, -90, -101, -93, -94, -94, -94...
## $ rsrq_db           <dbl> -5, -6, -5, -6, -6, -10, -8, -11, -11, -10, -9, -...
## $ rssnr_db          <dbl> 22, 11, 29, 13, 16, 13, 7, 0, 8, 2, 24, 10, 22, 1...
## $ cqi               <dbl> 10, 13, 15, 12, 9, 15, 10, 9, 9, 7, 10, 9, 12, 15...
## $ ta                <dbl> 7, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3...
## $ enodeb            <fct> 103068, 114809, 114809, 114809, 114809, 114809, 1...
## $ f_mhz             <dbl> 1720, 1720, 1720, 1720, 1720, 1720, 1720, 1720, 1...
## $ payload_mb        <dbl> 4.0, 2.0, 4.0, 9.0, 8.0, 6.0, 5.0, 4.0, 3.0, 2.0,...
## $ throughput_mbits  <dbl> 24.52, 14.86, 16.27, 12.68, 14.59, 13.13, 16.37, ...
```

```
dataset_ul_vodafone = filter(dataset_ul, provider=="vodafone")
glimpse(dataset_ul_vodafone)
```

```
## Rows: 1,828
## Columns: 16
## $ row_id_original   <int> 4341, 4342, 4343, 4344, 4345, 4346, 4347, 4348, 4...
## $ drive_id          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ timestamp         <dttm> 2018-12-10 09:09:03, 2018-12-10 09:09:21, 2018-1...
```

```
## $ scenario          <fct> campus, campus, campus, campus, campus, campus, c...
## $ provider          <fct> vodafone, vodafone, vodafone, vodafone, vodafone,...
## $ velocity_mps      <dbl> 11.70, 8.22, 8.00, 10.30, 12.28, 0.00, 0.00, 0.00...
## $ acceleration_mpss <dbl> 0.06, 0.32, 0.53, 0.36, 0.12, 0.00, 0.00, 0.00, -...
## $ rsrp_dbm          <dbl> -121, -108, -111, -106, -110, -94, -95, -92, -98,...
## $ rsrq_db           <dbl> -15, -9, -13, -8, -9, -7, -7, -8, -6, -10, -7, -8...
## $ rssnr_db          <dbl> -8, 2, 6, 5, 9, 23, 23, 24, 14, 1, 14, 12, 14, 7,...
## $ cqi               <dbl> 4, 2, 6, 11, 10, 15, 12, 15, 12, 6, 15, 10, 11, 7...
## $ ta                <dbl> 63, 21, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 1...
## $ enodeb            <fct> 51044, 52316, 50026, 50026, 50026, 50026, 50026, ...
## $ f_mhz             <dbl> 1770, 1770, 1770, 1770, 1770, 1770, 1770, 1770, 1...
## $ payload_mb        <dbl> 6.0, 10.0, 0.1, 2.0, 6.0, 0.1, 0.1, 0.5, 7.0, 0.1...
## $ throughput_mbits  <dbl> 1.29, 3.18, 0.05, 2.93, 8.79, 5.16, 4.73, 10.13, ...
```

Create the Prediction Tasks for Each Provider

```
make_task = function(dataset, task_id) {
  task = TaskRegr$new(
    id = task_id,
    backend = dataset %>% select(-drive_id, -timestamp, -provider, -scenario),
    target = "throughput_mbits"
  )

  task$col_roles$name = "row_id_original"
  task$col_roles$feature = setdiff(task$col_roles$feature, "row_id_original")

  return(task)
}

task_ul_o2 = make_task(dataset_ul_o2, "task_ul_o2")
task_ul_o2
```

```
## <TaskRegr:task_ul_o2> (2039 x 11)
## * Target: throughput_mbits
## * Properties: -
## * Features (10):
##   - dbl (9): acceleration_mpss, cqi, f_mhz, payload_mb, rsrp_dbm,
##     rsrq_db, rssnr_db, ta, velocity_mps
##   - fct (1): enodeb
```

```
task_ul_tmobile = make_task(dataset_ul_tmobile, "task_ul_tmobile")
task_ul_tmobile
```

```
## <TaskRegr:task_ul_tmobile> (2301 x 11)
## * Target: throughput_mbits
## * Properties: -
## * Features (10):
##   - dbl (9): acceleration_mpss, cqi, f_mhz, payload_mb, rsrp_dbm,
##     rsrq_db, rssnr_db, ta, velocity_mps
##   - fct (1): enodeb
```

```
task_ul_vodafone = make_task(dataset_ul_vodafone, "task_ul_vodafone")
task_ul_vodafone
```

```
## <TaskRegr:task_ul_vodafone> (1828 x 11)
```

```
## * Target: throughput_mbits
## * Properties: -
## * Features (10):
##   - dbl (9): acceleration_mpss, cqi, f_mhz, payload_mb, rsrp_dbm,
##     rsrq_db, rssnr_db, ta, velocity_mps
##   - fct (1): enodeb
```

Create Data Splitting Strategies for Testing and Validation

The outer resampling is used for the train/validation split.

```
get_row_ids_by_drive_ids = function(task, drive_ids) {
  result = (tibble(task$row_names) %>%
    inner_join(dataset_ul, by=c("row_name"="row_id_original")) %>%
    filter(drive_id %in% drive_ids))$row_id
  return(result)
}

make_outer_resampling = function(task, drive_ids_train, drive_ids_test) {
  row_ids_train = get_row_ids_by_drive_ids(task, drive_ids_train)
  row_ids_test = get_row_ids_by_drive_ids(task, drive_ids_test)

  result = rsmp("custom")
  result$instantiate(task, train_sets=list(row_ids_train), test_sets=list(row_ids_test))

  return(result)
}
```

The inner resampling is used for the parameter tuning on the training set.

```
make_inner_resampling = function(task, last_drive_id) {
  train_sets = list()
  test_sets = list()

  for (cur_last_drive_id_train in 2:(last_drive_id-1)) {
    drive_ids_train = 1:cur_last_drive_id_train
    drive_ids_test = cur_last_drive_id_train + 1

    row_ids_train = get_row_ids_by_drive_ids(task, drive_ids_train)
    row_ids_test = get_row_ids_by_drive_ids(task, drive_ids_test)

    train_sets[[length(train_sets)+1]] = row_ids_train
    test_sets[[length(test_sets)+1]] = row_ids_test
  }

  result = rsmp("custom")
  result$instantiate(task, train_sets=train_sets, test_sets=test_sets)

  return(result)
}
```

Create the Prediction Pipeline for Each Provider

```
make_learner = function(nrounds=100, eta=NULL, gamma=NULL, lambda=NULL) {
  factor_encoding = po(
```

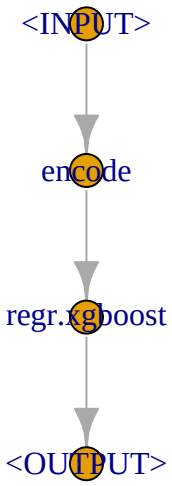
```
    "encode",
    method = "one-hot",
    affect_columns = selector_type("factor")
  )
  xgboost = lrn("regr.xgboost")

  if (!is.null(nrounds)) {
    xgboost$param_set$values = mlr3misc::insert_named(
      xgboost$param_set$values,
      list(nrounds=nrounds)
    )
  }
  if (!is.null(eta)) {
    xgboost$param_set$values = mlr3misc::insert_named(
      xgboost$param_set$values,
      list(eta=eta)
    )
  }
  if (!is.null(gamma)) {
    xgboost$param_set$values = mlr3misc::insert_named(
      xgboost$param_set$values,
      list(gamma=gamma)
    )
  }
  if (!is.null(lambda)) {
    xgboost$param_set$values = mlr3misc::insert_named(
      xgboost$param_set$values,
      list(lambda=lambda)
    )
  }

  pipe = factor_encoding %>>% PipeOpLearner$new(xgboost)
  learner = GraphLearner$new(pipe)
  return(learner)
}
```

Here we can see the prediction pipeline:

```
make_learner()$graph$plot()
```



Parameter Tuning

```
parameter_space = ParamSet$new(list(
  ParamInt$new("regr.xgboost.nrounds", lower=100, upper=1000),
  ParamDbl$new("regr.xgboost.eta", lower=0.01, upper=1),
  ParamDbl$new("regr.xgboost.gamma", lower=0, upper=10),
  ParamDbl$new("regr.xgboost.lambda", lower=0, upper=10)
))
```

```
get_tuning_result = function(task, grid_resolution, n_evals) {
  tuning_instance = TuningInstanceSingleCrit$new(
    task = task,
    learner = make_learner(),
    resampling = make_inner_resampling(task, last_drive_id=7),
    measure = msr("regr.mae"),
    terminator = trm("evals", n_evals=n_evals),
    search_space = parameter_space,
    store_benchmark_result = TRUE,
    check_values = TRUE
  )

  tuner = tnr("grid_search", resolution = grid_resolution)
  tuner$optimize(tuning_instance)

  return(tuning_instance)
}
```

```
tuning_result_o2 = get_tuning_result(task_ul_o2, grid_resolution = 20, n_evals = 10)

tuning_result_tmobile = get_tuning_result(task_ul_tmobile, grid_resolution = 20, n_evals = 10)

tuning_result_vodafone = get_tuning_result(task_ul_vodafone, grid_resolution = 20, n_evals = 10)
```

```
tuning_result_o2$result
```

```
##    regr.xgboost.nrounds regr.xgboost.eta regr.xgboost.gamma regr.xgboost.lambda
## 1:                  194       0.06210526           1.578947                  10
##    learner_param_vals  x_domain regr.mae
## 1:          <list[7]> <list[4]> 2.309917
```

```
tuning_result_tmobile$result
```

```
##    regr.xgboost.nrounds regr.xgboost.eta regr.xgboost.gamma regr.xgboost.lambda
## 1:                  574        0.2705263           8.421053            3.157895
##    learner_param_vals  x_domain regr.mae
## 1:          <list[7]> <list[4]> 3.173148
```

```
tuning_result_vodafone$result
```

```
##    regr.xgboost.nrounds regr.xgboost.eta regr.xgboost.gamma regr.xgboost.lambda
## 1:                  431        0.2184211           7.368421            5.263158
##    learner_param_vals  x_domain regr.mae
## 1:          <list[7]> <list[4]> 2.588257
```

Create Learners with Tuned Hyperparameters

```
learner_ul_o2 = make_learner(
  nrounds = tuning_result_o2$result$regr.xgboost.nrounds,
  eta = tuning_result_o2$result$regr.xgboost.eta,
  gamma = tuning_result_o2$result$regr.xgboost.gamma,
  lambda = tuning_result_o2$result$regr.xgboost.lambda
)

learner_ul_tmobile = make_learner(
  nrounds = tuning_result_tmobile$result$regr.xgboost.nrounds,
  eta = tuning_result_tmobile$result$regr.xgboost.eta,
  gamma = tuning_result_tmobile$result$regr.xgboost.gamma,
  lambda = tuning_result_tmobile$result$regr.xgboost.lambda
)

learner_ul_vodafone = make_learner(
  nrounds = tuning_result_vodafone$result$regr.xgboost.nrounds,
  eta = tuning_result_vodafone$result$regr.xgboost.eta,
  gamma = tuning_result_vodafone$result$regr.xgboost.gamma,
  lambda = tuning_result_vodafone$result$regr.xgboost.lambda
)
```

Validation Results

```
resampling_result_ul_o2 = resample(
  task = task_ul_o2,
  learner = learner_ul_o2,
  resampling = make_outer_resampling(task_ul_o2, drive_ids_train=1:7, drive_ids_test=8:10),
  store_models = TRUE
)

resampling_result_ul_tmobile = resample(
  task = task_ul_tmobile,
  learner = learner_ul_tmobile,
  resampling = make_outer_resampling(task_ul_tmobile, drive_ids_train=1:7, drive_ids_test=8:10),
  store_models = TRUE
)

resampling_result_ul_vodafone = resample(
  task = task_ul_vodafone,
  learner = learner_ul_vodafone,
  resampling = make_outer_resampling(task_ul_vodafone, drive_ids_train=1:7, drive_ids_test=8:10),
  store_models = TRUE
)
```

```
predictions_ul_o2 = as.data.table(resampling_result_ul_o2$prediction())
glimpse(tibble(predictions_ul_o2))
```

```
## Rows: 615
## Columns: 3
## $ row_id   <int> 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148...
## $ truth    <dbl> 4.47, 2.59, 2.26, 1.09, 0.77, 0.19, 0.26, 0.65, 1.45, 1.12...
## $ response <dbl> 2.9530549, 2.0047722, 3.8707159, 2.7057931, 3.3690865, 1.1...
```

```
predictions_ul_tmobile = as.data.table(resampling_result_ul_tmobile$prediction())
predictions_ul_vodafone = as.data.table(resampling_result_ul_vodafone$prediction())
```

```
validation_results_ul = bind_rows(
  tibble(predictions_ul_o2) %>%
    inner_join(tibble(task_ul_o2$row_names), by="row_id") %>%
    inner_join(dataset_ul, by=c("row_name"="row_id_original")),
  tibble(predictions_ul_tmobile) %>%
    inner_join(tibble(task_ul_tmobile$row_names), by="row_id") %>%
    inner_join(dataset_ul, by=c("row_name"="row_id_original")),
  tibble(predictions_ul_vodafone) %>%
    inner_join(tibble(task_ul_vodafone$row_names), by="row_id") %>%
    inner_join(dataset_ul, by=c("row_name"="row_id_original"))
)
glimpse(validation_results_ul)
```

```
## Rows: 1,840
## Columns: 19
## $ row_id            <int> 137, 138, 139, 140, 141, 142, 143, 144, 145, 146,...
## $ truth             <dbl> 4.47, 2.59, 2.26, 1.09, 0.77, 0.19, 0.26, 0.65, 1...
## $ response          <dbl> 2.9530549, 2.0047722, 3.8707159, 2.7057931, 3.369...
## $ row_name          <int> 137, 138, 139, 140, 141, 142, 143, 144, 145, 146,...
## $ drive_id          <int> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 9, 9...
## $ timestamp         <dttm> 2018-12-11 09:04:11, 2018-12-11 09:04:22, 2018-1...
## $ scenario          <fct> campus, campus, campus, campus, campus, campus, c...
```

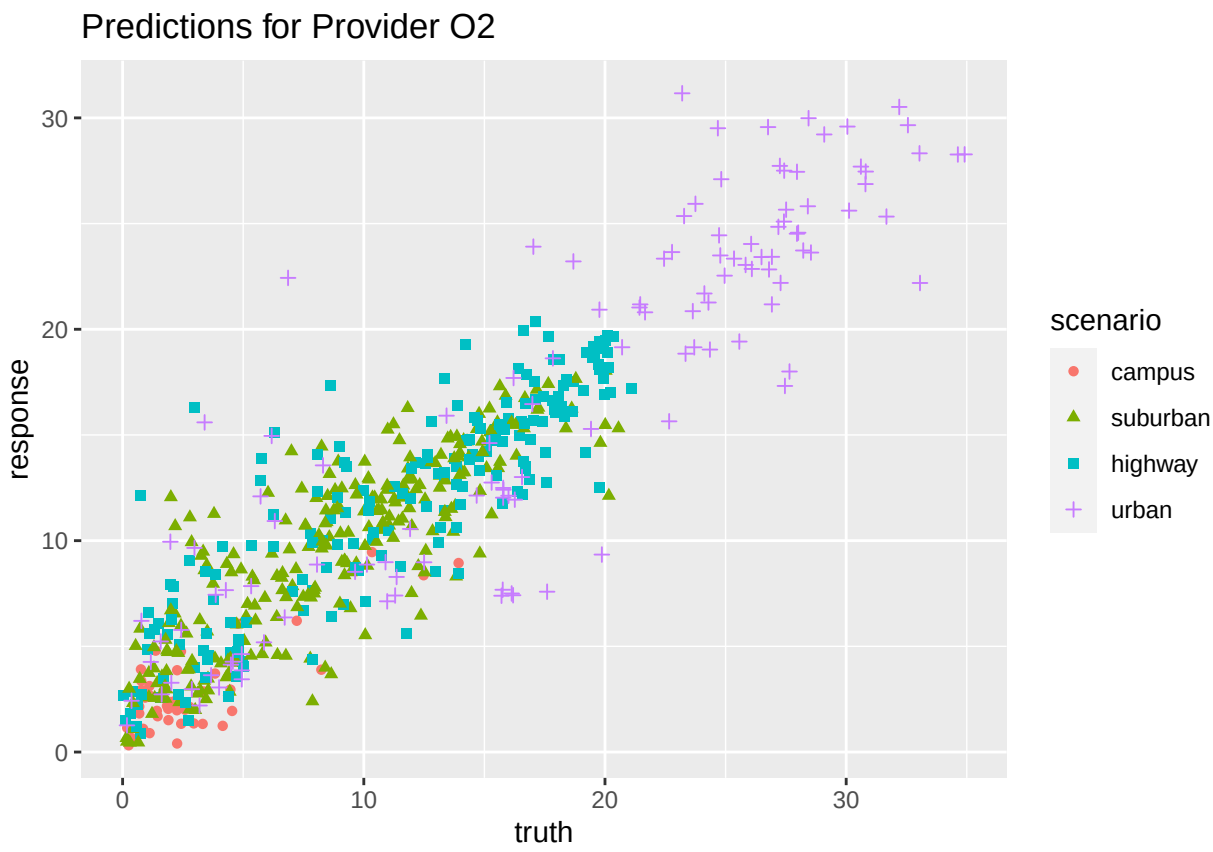
```
## $ provider          <fct> o2, o2, o2, o2, o2, o2, o2, o2, o2, o2, o2, o2, o...
## $ velocity_mps      <dbl> 0.00, 6.11, 9.39, 8.45, 11.68, 0.00, 0.00, 0.00, ...
## $ acceleration_mpss <dbl> 0.00, 0.35, 0.18, 0.39, 1.59, 0.00, 0.00, 0.00, 0...
## $ rsrp_dbm          <dbl> -89, -92, -94, -98, -102, -100, -101, -101, -100,...
## $ rsrq_db           <dbl> -9, -12, -14, -15, -16, -17, -16, -16, -17, -14, ...
## $ rssnr_db          <dbl> 13, 3, -1, -3, -5, -7, -6, -5, -8, 1, -7, -1, -2,...
## $ cqi               <dbl> 11, 5, 5, 4, 2, 3, 4, 4, 4, 6, 3, 5, 5, 2, 4, 6, ...
## $ ta                <dbl> 7, 7, 7, 7, 7, 12, 12, 12, 12, 12, 12, 12, 12, 12...
## $ enodeb            <fct> 52410, 52410, 52410, 52410, 52410, 52900, 52900, ...
## $ f_mhz             <dbl> 880, 880, 880, 880, 880, 880, 880, 880, 880, 880,...
## $ payload_mb        <dbl> 0.1, 0.5, 3.0, 9.0, 7.0, 3.0, 2.0, 2.0, 6.0, 3.0,...
## $ throughput_mbits  <dbl> 4.47, 2.59, 2.26, 1.09, 0.77, 0.19, 0.26, 0.65, 1...
```

```r
all(validation_results_ul$truth == validation_results_ul$throughput_mbits)
```

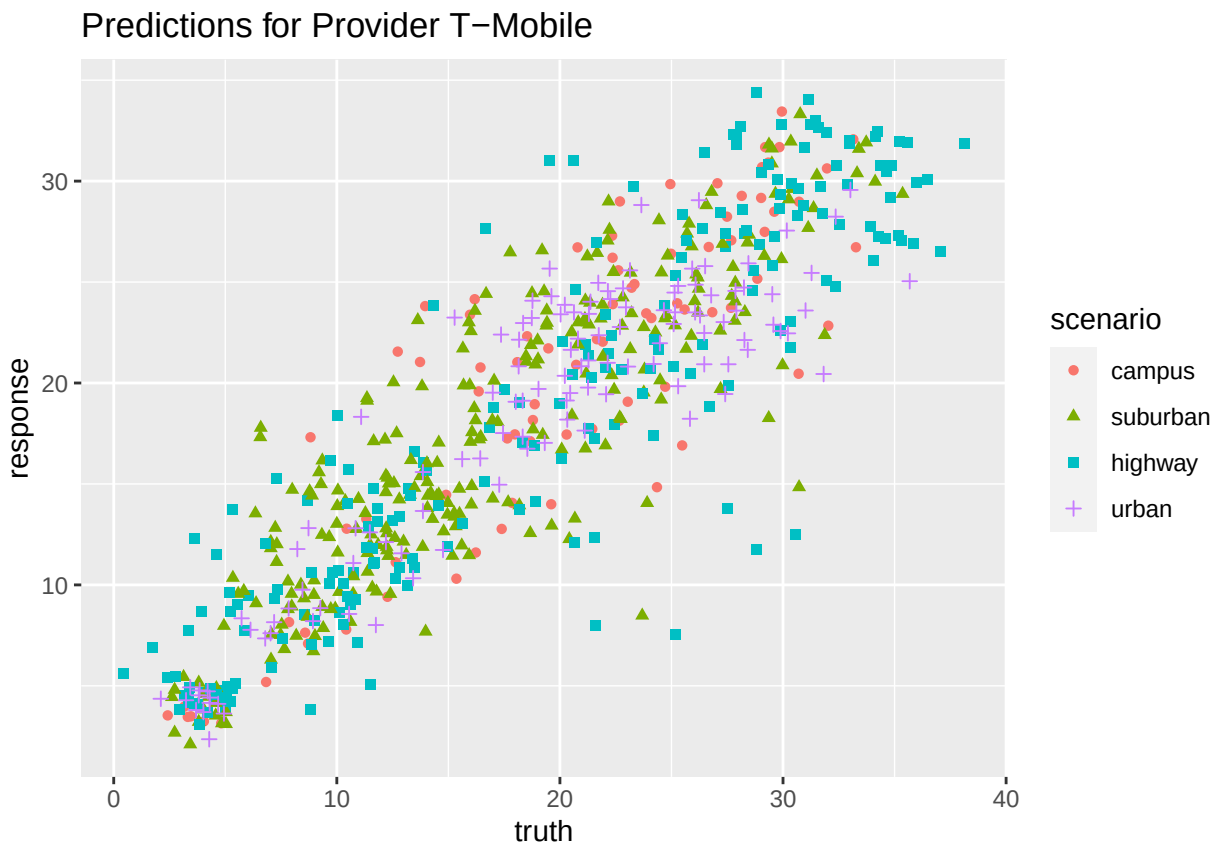
```
## [1] TRUE
```

Scatter Plots

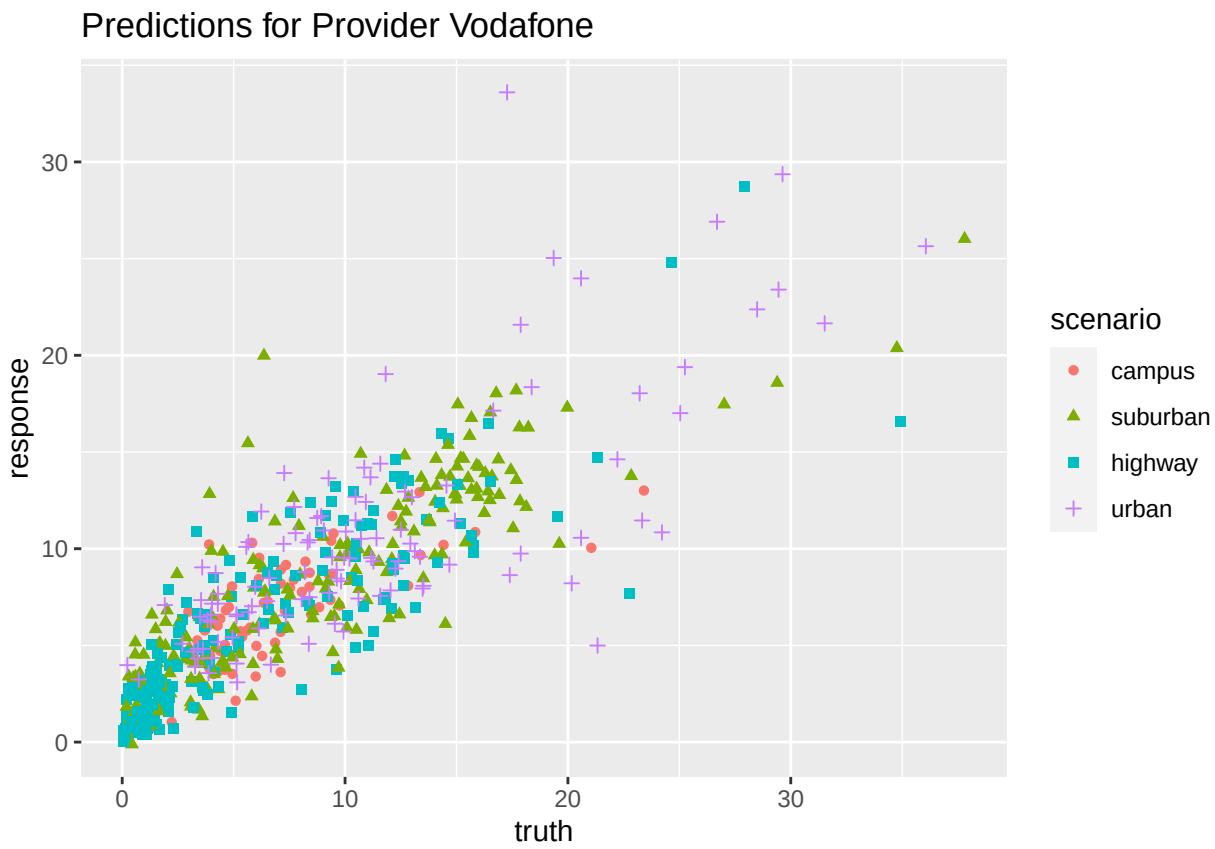
```r
ggplot(filter(validation_results_ul, provider=="o2"), aes(x=truth, y=response)) +
  geom_point(aes(color=scenario, shape=scenario)) +
  ggtitle("Predictions for Provider O2")
```




```r
ggplot(filter(validation_results_ul, provider=="tmobile"), aes(x=truth, y=response)) +
  geom_point(aes(color=scenario, shape=scenario)) +
  ggtitle("Predictions for Provider T-Mobile")
```




```
ggplot(filter(validation_results_ul, provider=="vodafone"), aes(x=truth, y=response)) +
  geom_point(aes(color=scenario, shape=scenario)) +
  ggtitle("Predictions for Provider Vodafone")
```



Performance Measures

```
compute_measure = function(measure, dataset, truth_column, response_column, providers, scenarios) {
  data_subset = filter(dataset, provider %in% providers, scenario %in% scenarios)
  truth = data_subset[[truth_column]]
  response = data_subset[[response_column]]

  tmp_prediction = PredictionRegr$new(
    row_ids = seq_along(truth),
    truth = truth,
    response = response
  )

  result = measure$score(tmp_prediction)
  return(result)
}
```

```
scenarios = c("campus", "suburban", "highway", "urban")
providers = c("o2", "tmobile", "vodafone")

results_by_provider = tibble()
for (cur_provider in providers) {
  cur_rsq = compute_measure(
    measure = msr("regr.rsq"),
    dataset = validation_results_ul,
    truth_column = "truth",
```

```
    response_column = "response",
    providers = cur_provider,
    scenarios = scenarios
  )
  cur_mae = compute_measure(
    measure = msr("regr.mae"),
    dataset = validation_results_ul,
    truth_column = "truth",
    response_column = "response",
    providers = cur_provider,
    scenarios = scenarios
  )

  results_by_provider = bind_rows(results_by_provider, list("provider"=cur_provider, "rsq"=cur_rsq, "mae
}

results_by_scenario = tibble()
for (cur_scenario in scenarios) {
  cur_rsq = compute_measure(
    measure = msr("regr.rsq"),
    dataset = validation_results_ul,
    truth_column = "truth",
    response_column = "response",
    providers = providers,
    scenarios = cur_scenario
  )
  cur_mae = compute_measure(
    measure = msr("regr.mae"),
    dataset = validation_results_ul,
    truth_column = "truth",
    response_column = "response",
    providers = providers,
    scenarios = cur_scenario
  )

  results_by_scenario = bind_rows(results_by_scenario, list("scenario"=cur_scenario, "rsq"=cur_rsq, "mae
}

results_by_provider_and_scenario = tibble()
for (cur_provider in providers) {
  for (cur_scenario in scenarios) {
    cur_rsq = compute_measure(
      measure = msr("regr.rsq"),
      dataset = validation_results_ul,
      truth_column = "truth",
      response_column = "response",
      providers = cur_provider,
      scenarios = cur_scenario
    )
    cur_mae = compute_measure(
      measure = msr("regr.mae"),
      dataset = validation_results_ul,
      truth_column = "truth",
```

```
      response_column = "response",
      providers = cur_provider,
      scenarios = cur_scenario
    )

    results_by_provider_and_scenario = bind_rows(
      results_by_provider_and_scenario,
      list(
        provider = cur_provider,
        scenario = cur_scenario,
        rsq=cur_rsq,
        mae=cur_mae
      )
    )
  }
}
```

```
knitr::kable(results_by_provider)
```

provider	rsq	mae
o2	0.8234817	2.332614
tmobile	0.7898664	3.044785
vodafone	0.7012584	2.473583

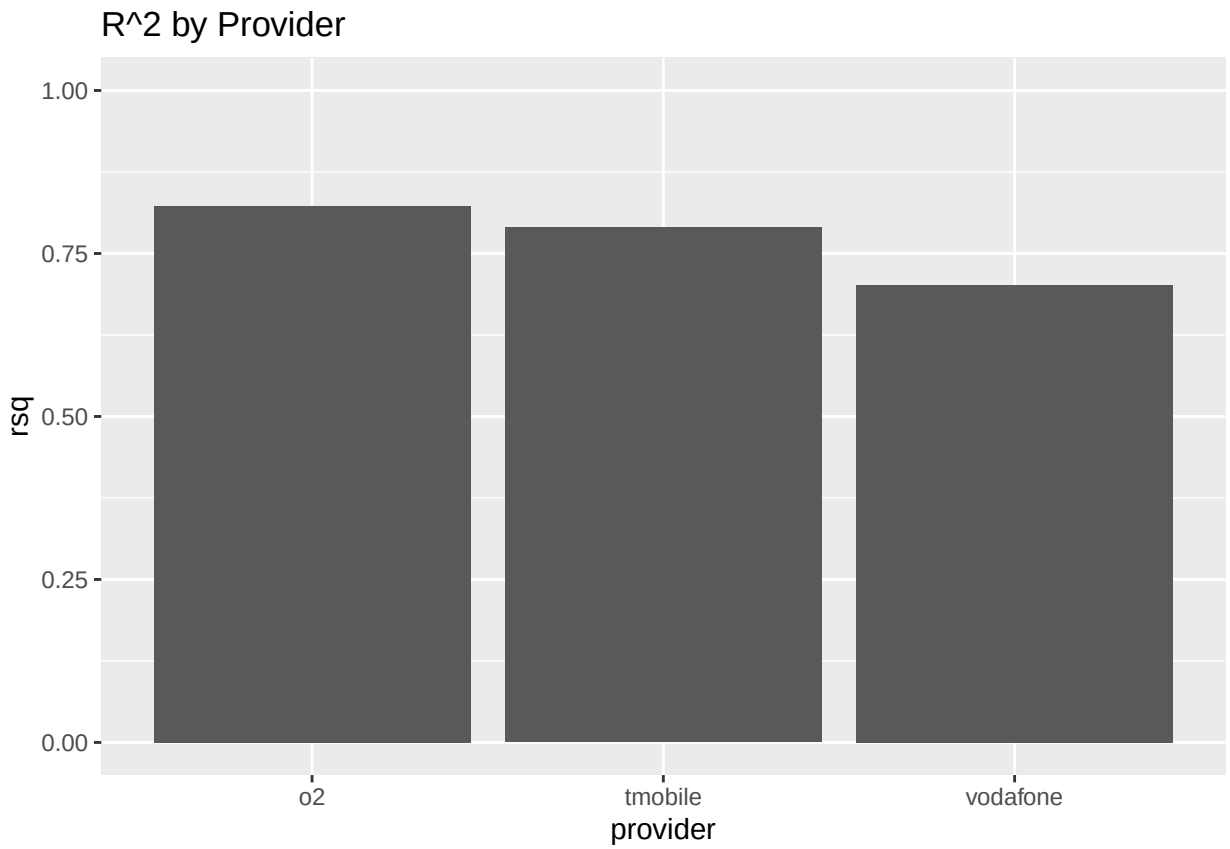
```
knitr::kable(results_by_scenario)
```

scenario	rsq	mae
campus	0.8887159	2.213314
suburban	0.8008482	2.504192
highway	0.8490269	2.572402
urban	0.7822949	3.234876

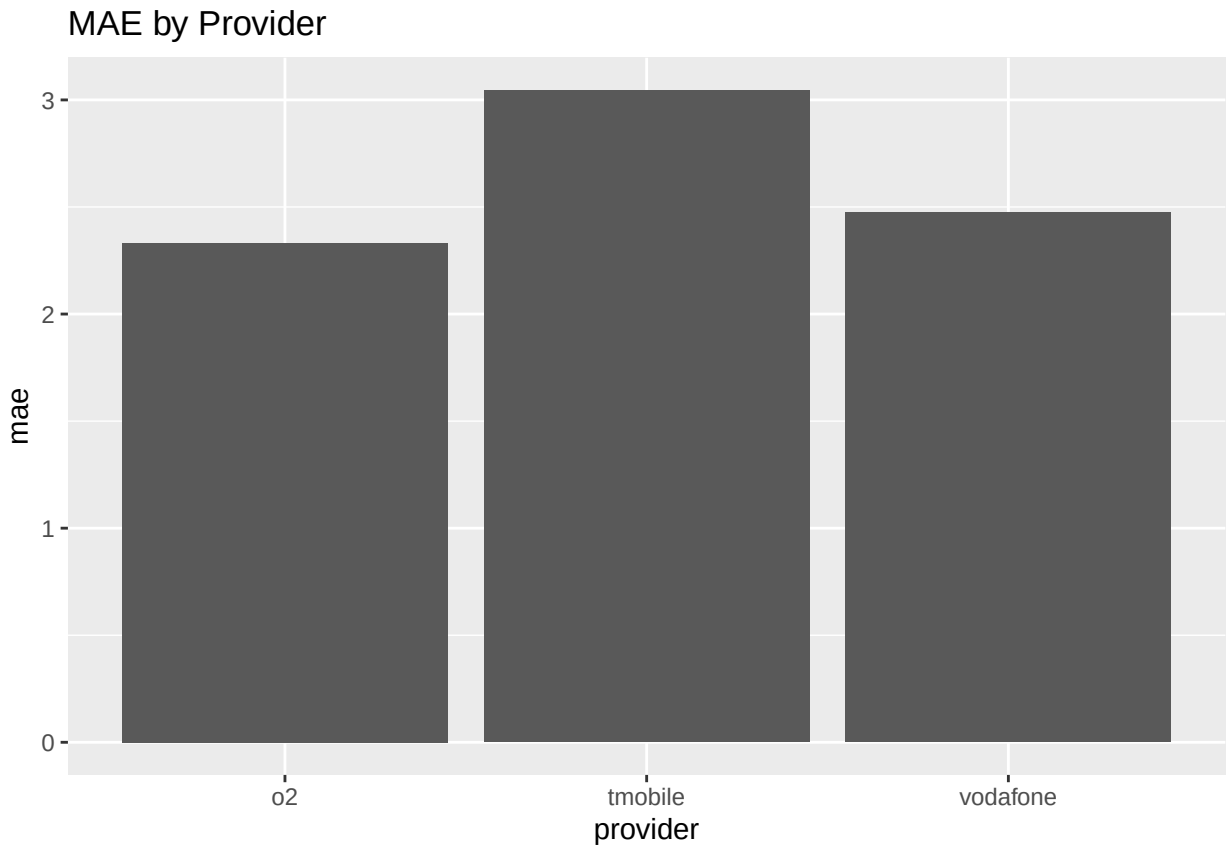
```
knitr::kable(results_by_provider_and_scenario)
```

provider	scenario	rsq	mae
o2	campus	0.6404435	1.315110
o2	suburban	0.6960392	2.028823
o2	highway	0.7515078	2.217786
o2	urban	0.7861956	3.556171
tmobile	campus	0.7434809	3.046734
tmobile	suburban	0.7519611	2.968127
tmobile	highway	0.8075956	3.324886
tmobile	urban	0.8146823	2.722458
vodafone	campus	0.5604603	1.841027
vodafone	suburban	0.7239832	2.511947
vodafone	highway	0.7244344	2.071376
vodafone	urban	0.5762142	3.441359

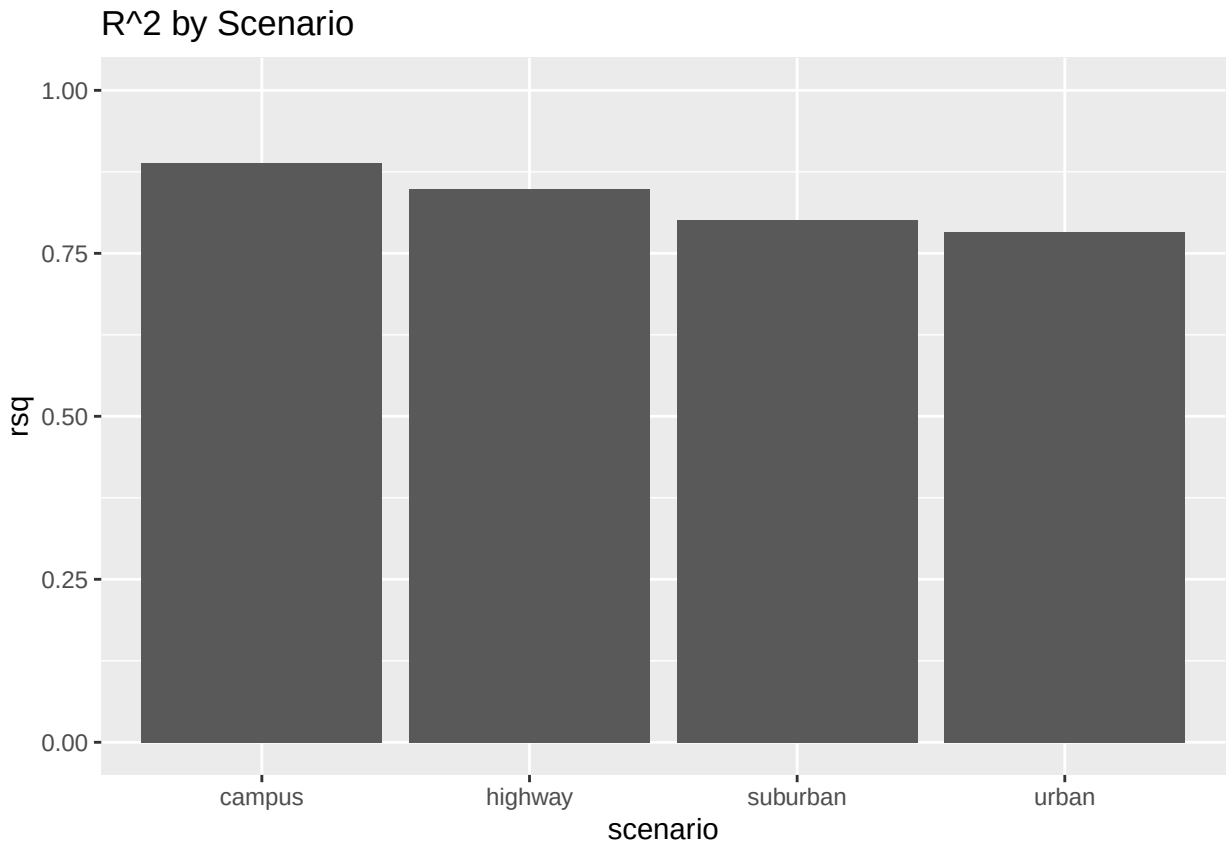
```r
ggplot(results_by_provider, aes(x=provider, y=rsq)) +
  geom_bar(stat = "identity") +
  ylim(c(0, 1)) +
  ggtitle("R^2 by Provider")
```



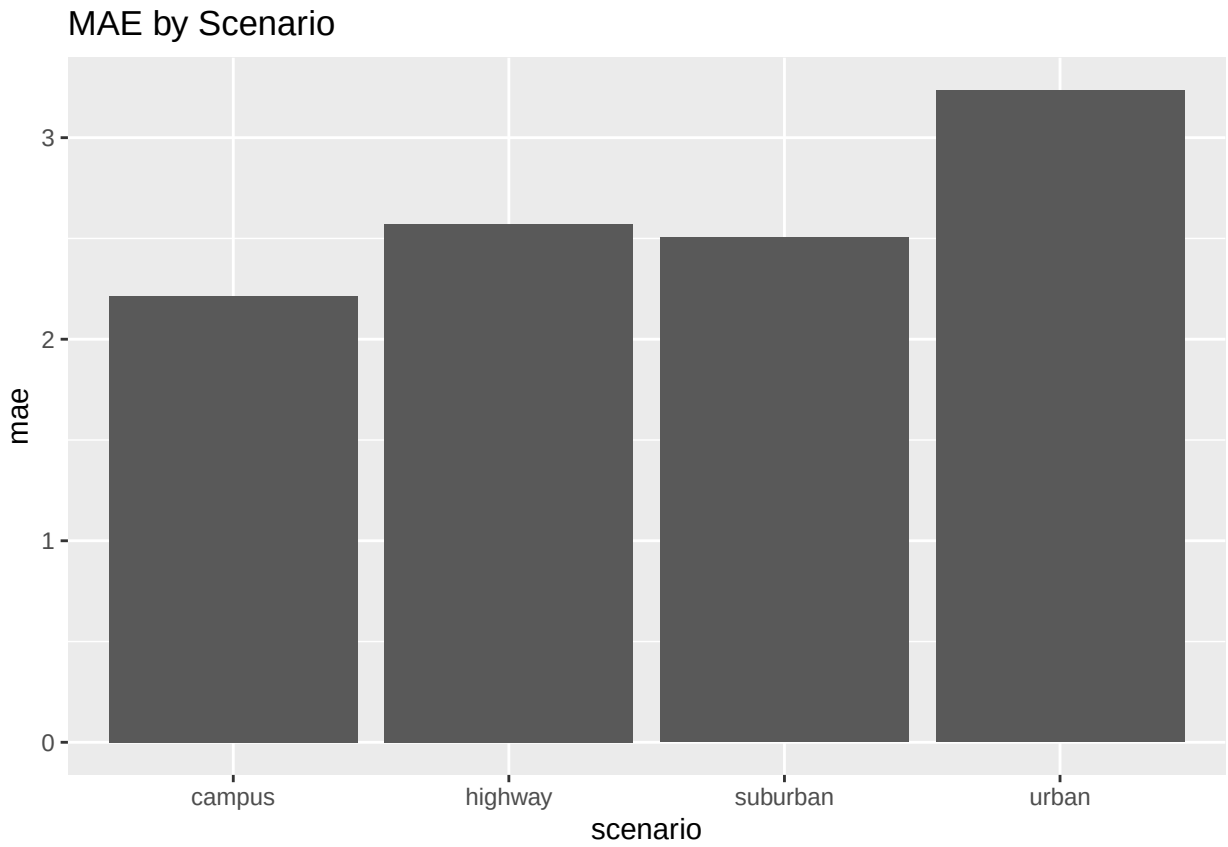

```r
ggplot(results_by_provider, aes(x=provider, y=mae)) +
  geom_bar(stat = "identity") +
  ggtitle("MAE by Provider")
```



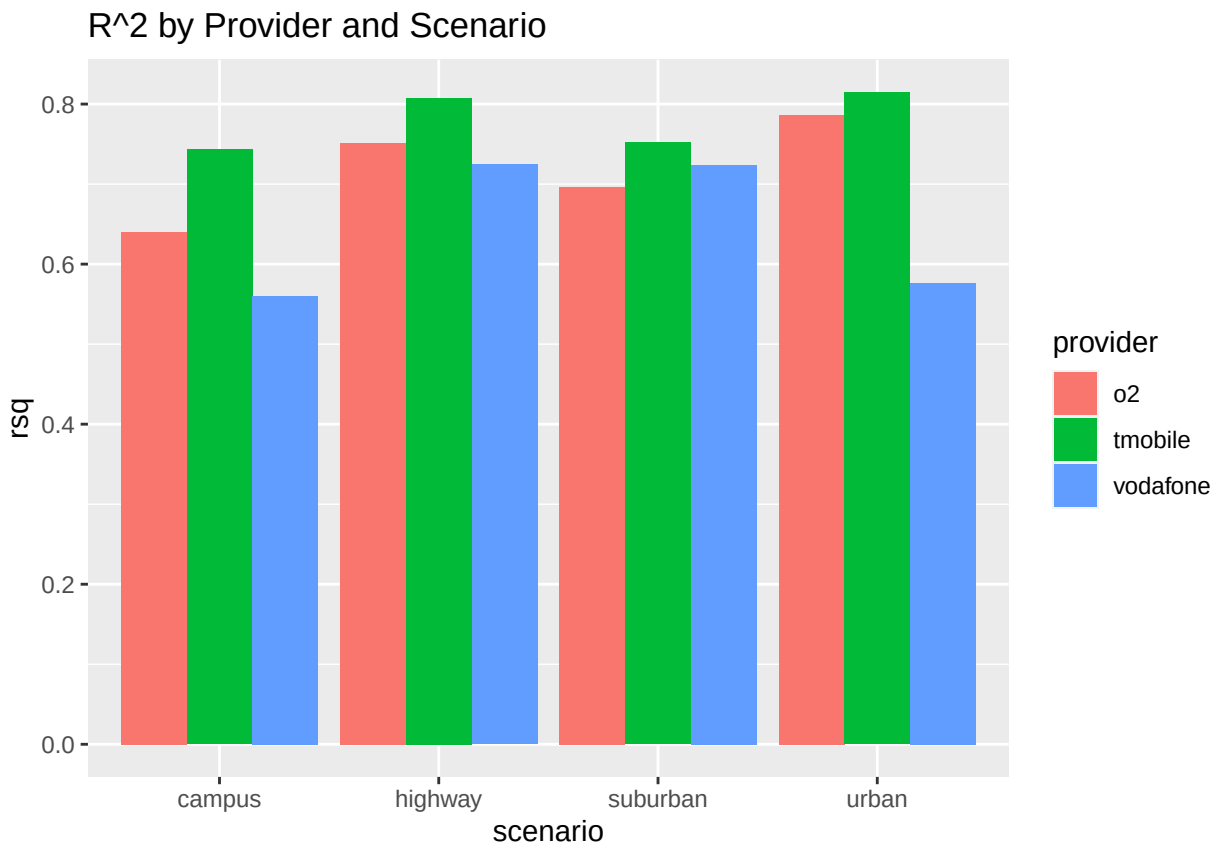

```
ggplot(results_by_scenario, aes(x=scenario, y=rsq)) +
  geom_bar(stat = "identity") +
  ylim(c(0, 1)) +
  ggtitle("R^2 by Scenario")
```



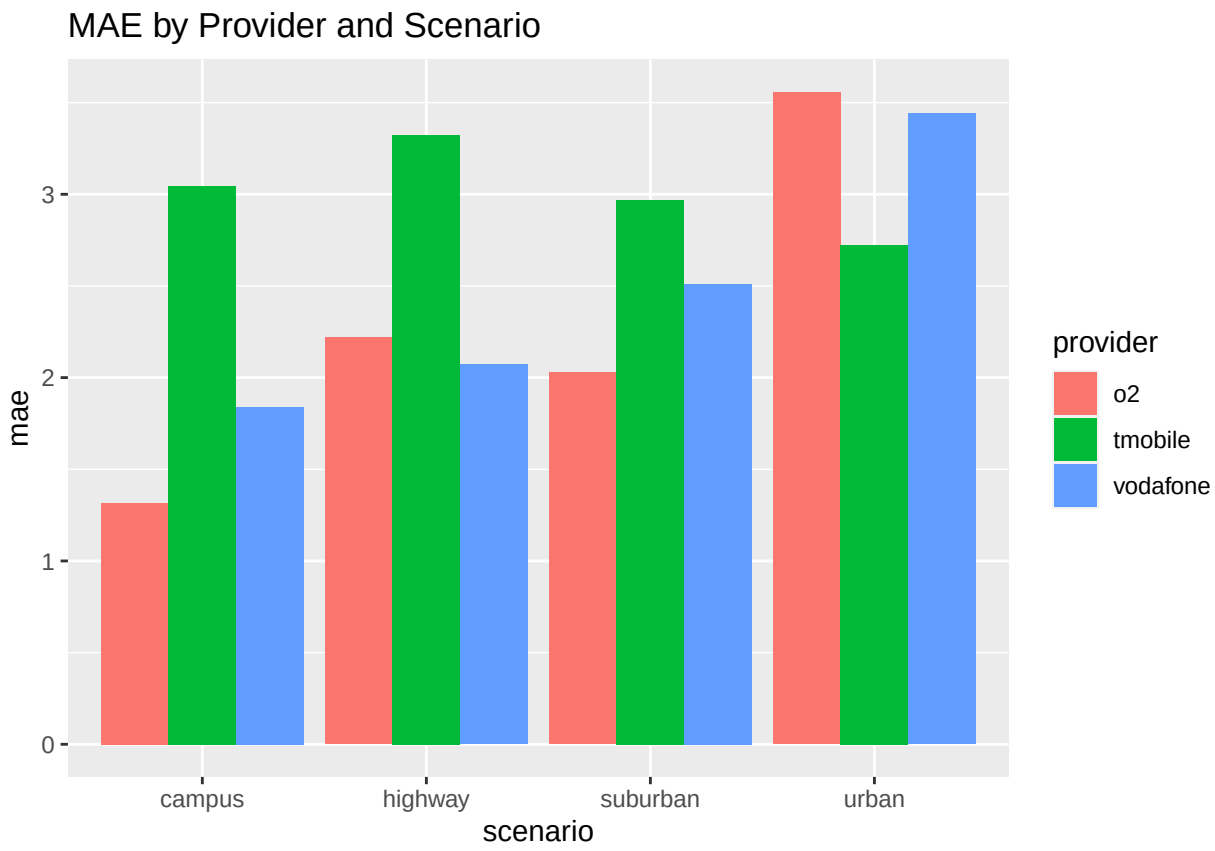

```
ggplot(results_by_scenario, aes(x=scenario, y=mae)) +
  geom_bar(stat = "identity") +
  ggtitle("MAE by Scenario")
```




```
ggplot(results_by_provider_and_scenario, aes(x=scenario, y=rsq, fill=provider)) +
  geom_bar(position = "dodge", stat = "identity") +
  ggtitle("R^2 by Provider and Scenario")
```




```
ggplot(results_by_provider_and_scenario, aes(x=scenario, y=mae, fill=provider)) +
  geom_bar(position = "dodge", stat = "identity") +
  ggtitle("MAE by Provider and Scenario")
```



Feature Importance

```
uninstantiate_resampling = function(resampling) {
  new_resampling = new.env()
  class(new_resampling) = class(resampling)
  for (val in ls(resampling)) {
    if (val != "is_instantiated") {
      assign(val, get(val, envir=resampling), envir = new_resampling)
    }
  }
  new_resampling$is_instantiated = FALSE

  return(new_resampling)
}
```

Provider O2

```
filter_permutation_o2 = flt("permutation",
  learner = make_learner(),
  resampling = uninstantiate_resampling(
    make_outer_resampling(task_ul_o2, drive_ids_train=1:7, drive_ids_test=8:10)
  ),
  measure = msr("regr.mae"),
  standardize = TRUE,
  nmc=10
```

```
)
filter_permutation_o2$calculate(task_ul_o2)
permutation_results_o2 = as.data.table(filter_permutation_o2)
```

```
ggplot(permutation_results_o2) +
  geom_bar(aes(x = reorder(feature, -score), y = score), stat="identity") +
  xlab("feature") +
  ylab("MAE difference") +
  scale_x_discrete(guide = guide_axis(angle = 20)) +
  ggtitle("Permutation Feature Importance")
```