Untitled

Rafael

06/01/2021

```
#' Analise e replicacao de alguns resultados em Fellner et all (2013)
#' Autor: Rafael Felipe Bressan
#' Arquivo: script_R.R
#' Script R para o trabalho monografico: Inferência Causal com Machine Learning
#' uma aplicacao para evasao fiscal
#' Pós-Graduacao Lato Sensu em Ciencia de Dados e Big Data PUC-MG
#' Ano: 2021
#'
#' Carrega as bibliotecas
library(sandwich)
library(fixest)
library(tidyverse)
library(glue)
library(skimr)
library(knitr)
library(kableExtra)
library(stargazer)
library(texreg)

#' Tabela 1 com a descricao das variaveis
desc_tbl <- read_csv("descricao.csv")

kbl(desc_tbl, booktabs = TRUE, longtable = TRUE, format = "latex",
    col.names = c("Variável", "Descrição"),
    caption = "Variáveis e descrições",
    label = "descricao") %>%
  kable_styling(full_width = FALSE,
                latex_options = c("repeat_header"),
                repeat_header_text = "(continuação)") %>%
  column_spec(2, width = "30em") %>%
  save_kable(file = "./Tables/table_descricao.tex")

#' Carregando os dados
data <- haven::read_dta("data_final.dta") %>%
  as.data.frame()

# Descrevendo o desenho do experimento ------------------------------

#' Tabela contando o numero de recipientes de cada tratamento
#'
data$treatment <- factor(data$treatment)
t_sizes <- as.vector(table(data$treatment))
```

```r
buckets <- data.frame(treatment = sort(unique(data$treatment)),
                      Buckets = c("T0", "T1", "T2", "T3", "T4", "T5", "T6"),
                      Description = c("Sem Correio", "Correio", "Ameaça", "Info",
                                      "Info&Ameaça", "Moral", "Moral&Ameaça"),
                      Size = t_sizes) %>%
  mutate(Prop = Size / nrow(data))

#' Tabela 2 recipientes de cada tratamento
kbl(buckets[-1], format = "latex", booktabs = TRUE, label = "descritivas1",
    col.names = c("Tratamento", "Descrição", "Observações", "Proporção"),
    caption = "Distribuição dos tratamentos na amostra.") %>%
  kable_styling(latex_options = c("HOLD_position")) %>%
  kable_classic(full_width = FALSE) %>%
  save_kable("./Tables/table_descritivas1.tex")

#' Junta de volta as inforamacoes de bucket e descricao para os dados
data <- data %>%
  left_join(buckets[, c("treatment", "Buckets", "Description")], by = "treatment")
#' Estatisticas descritivas da base
#' Variaveis com valor NA
skim_df <- data %>%
  skim_without_charts()

na_df <- skim_df %>%
  filter(n_missing > 0) %>%
  select(skim_variable, n_missing, complete_rate)

#' Tabela 3 dados faltantes
kbl(na_df, digits = 2, format = "latex", booktabs = TRUE, label = "missings",
    col.names = c("Variável", "No. Faltantes", "Completude"),
    caption = "Dados faltantes na amostra.") %>%
  kable_styling(latex_options = c("HOLD_position")) %>%
  kable_classic(full_width = FALSE) %>%
  footnote(general_title = "Nota:",
           general = "Completude refere-se a proporção de linhas preenchidas contra faltantes, e varia
           threeparttable = TRUE) %>%
  save_kable("./Tables/table_missings.tex")

#' Analise Exploratoria
#' Problema de atrito
attrition_level <- data %>%
  filter(mailing == 1) %>%
  group_by(treatment, Buckets, Description) %>%
  summarise(mail_count = n(),
            deliv_na_count = sum(is.na(delivered)),
            deliv_0_count = sum(delivered == 0),
            attr_rate = (deliv_na_count + deliv_0_count) / nrow(cur_data()))
chi_test <- chisq.test(attrition_level$deliv_0_count,
                       p = attrition_level$mail_count, rescale.p = TRUE)
atrito_foot <- glue("Na média total a taxa de atrito foi de {format(mean(attrition_level$attr_rate), di
#' Tabela 5 detalhando o nivel de atrito por tratamento
kbl(attrition_level[-1], digits = 4, format = "latex", booktabs = TRUE,
    label = "atrito-level",
```

```
    col.names = c("Tratamento", "Descrição", "Cartas", "Entregues NA",
                  "Não Entregues", "Taxa Atrito"),
    caption = "Taxa de atrito por tratamento.") %>%
  kable_styling(latex_options = c("HOLD_position")) %>%
  kable_classic(full_width = FALSE) %>%
  footnote(general_title = "Nota:", general = atrito_foot,
           threeparttable = TRUE) %>%
  save_kable("./Tables/table_atrito_level.tex")
#' Todos os NAs se referem ao grupo de controle, mas houve um pouco de atrito.
#' Verificar balanceamento de variaveis para aqueles que atritaram
atrito_controle <- data %>%
  filter(mailing == 0 | (mailing == 1 & delivered == 0))

attr_bal <- atrito_controle %>%
  group_by(treatment, Buckets) %>%
  summarise(across(c(gender, age_aver, inc_aver, pop2005, pop_density2005,
                     compliance),
                   mean, na.rm = TRUE))
#' Teste anova para diferenca de medias
attr_anova <- tibble(var = c("gender", "age_aver", "inc_aver", "pop2005",
                                    "pop_density2005", "compliance")) %>%
  rowwise() %>%
  mutate(anov = list(anova(lm(paste0(var, "~treatment"), data = atrito_controle))),
         pval = anov["treatment", "Pr(>F)"]) %>%
  select(var,pval) %>%
  pivot_wider(names_from = var, values_from = pval) %>%
  add_column(Buckets = "Anova p-valor", .before = 1)

atr_bal_foot <- "Gênero igual a zero para mulher. Demais variáveis são denominadas em nível municipal, p
#' Tabela 6 balanceamento com atrito
attr_bal[-1] %>%
  bind_rows(attr_anova) %>%
  kbl(digits = 4, format = "latex", booktabs = TRUE, label = "atrito-bal",
      col.names = c("Tratamento", "Gênero", "Idade", "Renda", "População",
                    "Dens. pop.", "Compliance"),
      caption = "Análise de atrito. Balanceamento de variáveis selecionadas") %>%
  kable_styling(latex_options = c("HOLD_position")) %>%
  kable_classic(full_width = FALSE) %>%
  footnote(general_title = "Nota:",
           threeparttable = TRUE,
           general = atr_bal_foot) %>%
  save_kable("./Tables/table_atrito_bal.tex")
#' Histograma com designacao de tratamento e atrito
hist1 <- data %>%
  filter(treatment %in% c("0", "6"), pop_density2005 < 30) %>%
  select(treatment, pop_density2005) %>%
  ggplot(aes(x = pop_density2005, y = ..density.., fill = treatment)) +
  geom_histogram(bins = 50, alpha = 0.8, position = "dodge") +
  labs(x = "",
       y = "Frequência relativa",
       title = "Tratamento designado") +
  guides(fill = guide_legend(title = "Tratamento")) +
  scale_fill_discrete(type = c("blue", "red")) +
```

```
  theme_classic()

hist2 <- atrito_controle %>%
  filter(treatment %in% c("0", "6"), pop_density2005 < 30) %>%
  select(treatment, pop_density2005) %>%
  ggplot(aes(x = pop_density2005, y = ..density.., fill = treatment)) +
  geom_histogram(bins = 50, alpha = 0.8, position = "dodge") +
  labs(x = "Densidade Populacional (hab/km2)",
       y = "Frequência relativa",
       title = "Atrito") +
  guides(fill = guide_legend(title = "Tratamento")) +
  scale_fill_discrete(type = c("blue", "red")) +
  theme_classic()
png("./Figs/fig_atr_hist.png")
gridExtra::grid.arrange(hist1, hist2)
dev.off()

#' Replicacao das tabelas 1 e 2 de Fellner et al.
#'
#' Tabela 1
gtab1 <- data %>%
  group_by(treatment, Buckets, Description) %>%
  summarise(across(c(gender, age_aver, inc_aver, pop2005,
                     pop_density2005, compliance),
                   mean, na.rm = TRUE))

anova_results <- data.frame(var = c("gender", "age_aver", "inc_aver", "pop2005",
                                    "pop_density2005", "compliance")) %>%
  rowwise() %>%
  mutate(anov = list(anova(lm(paste0(var, "~treatment"), data = data))),
         pval = anov["treatment", "Pr(>F)"])
anov_row <- anova_results %>%
  select(-anov) %>%
  pivot_wider(names_from = var, values_from = pval) %>%
  mutate(Buckets = "Anova: ",
         Description = "p-values") %>%
  select(Buckets, Description, everything())

#' Tabela 4 balanceamento. Table 1 de Fellner et. al
tbl1_cap <- "Balanceamento de características individuais e por município por tipo de tratamento."
gtab1[-1] %>%
  bind_rows(anov_row) %>%
  kbl(digits = 4, booktabs = TRUE, format = "latex", label = "tab1",
      col.names = c("Tratamento", "Descrição", "Gênero", "Idade", "Renda",
                    "População", "Dens. pop.", "Compliance"),
      caption = tbl1_cap) %>%
  kable_styling(latex_options = "HOLD_position", font_size = 10) %>%
  kable_classic(full_width = FALSE) %>%
  footnote(general_title = "Nota:",
           threeparttable = TRUE,
           general = atr_bal_foot) %>%
  save_kable(file = "./Tables/table1.tex")
```

```r
#' Regressoes da Tabela 7
#'
reg_21 <- feols(resp_A~mailing+threat+appeal+info, data = data)
reg_22 <- feols(resp_A~mailing+threat+appeal+info+i_tinf+i_tapp, data = data)

delivered <- data %>%
  filter(delivered == 1)
reg_23 <- feols(resp_B~threat+appeal+info, data = delivered)
reg_24 <- feols(resp_B~threat+appeal+info+i_tinf+i_tapp, data = delivered)
reg_25 <- feols(resp_all~threat+appeal+info, data = delivered)
reg_26 <- feols(resp_all~threat+appeal+info+i_tinf+i_tapp, data = delivered)

#' Dicionário para o nome das variáveis nas tabelas
fixest::setFixest_dict(c(resp_A = "Registro",
                         resp_B = "Atual. Contratual",
                         resp_all = "Resposta Geral",
                         mailing = "Correio",
                         threat = "Ameaça",
                         appeal = "Moral",
                         info = "Info",
                         i_tinf = "Ameaça x Info",
                         i_tapp = "Ameaça x Moral",
                         threat_evasion_D1 = "Ameaça x Evasão",
                         appeal_evasion_D1 = "Moral x Evasão",
                         info_evasion_D1 = "Info x Evasão",
                         evasion_1 = "Evasão",
                         threat_evasion_D2 = "Ameaça x Evasão",
                         appeal_evasion_D2 = "Moral x Evasão",
                         info_evasion_D2 = "Info x Evasão",
                         evasion_2 = "Evasão",
                         "(Intercept)" = "Constante"))
#' Ajusta o estilo das tabelas
est_style = list(depvar = "title:Dep. Var.",
                 model = "title:Modelo",
                 var = "title:\\emph{Variáveis}",
                 stats = "title:\\emph{Estatísticas de diagnóstico}",
                 notes = "title:\\emph{\\medskip Notas:}")
#' Cria a Tabela 7. Table 2 in Fellner et. al
esttex(reg_21, reg_22, reg_23, reg_24, reg_25, reg_26,
       file = "./Tables/table2.tex",
       label = "tab:tab2",
       style = est_style,
       replace = TRUE,
       se = "White",
       digits = 3,
       fitstat = "",
       order = c("Correio", "^Ameaça$", "^Moral$", "^Info$", "Ameaça x Moral",
                 "Ameaça x Info", "Constante"),
       title = "Efeito do tratamento nos registros, atualizações contratuais, and resposta geral para o

#' Efeitos Heterogeneos
#' Replicacao da Table C1
#' Mediana da população dos municípios, da densidade, da renda e de votantes
```

```r
#' a direita
med_pop <- median(data$pop2005, na.rm = TRUE)
med_den <- median(data$pop_density2005, na.rm = TRUE)
med_rend <- median(data$inc_aver, na.rm = TRUE)
med_vot <- median(data$vo_cr + data$vo_r, na.rm = TRUE)
#' Dataframe com indicadores de acima da mediana
efeito_het_df <- data %>%
  mutate(pop_hi = pop2005 >= med_pop,
         popdens_hi = pop_density2005 >= med_den,
         rend_hi = inc_aver >= med_rend,
         vot_hi = (vo_cr + vo_r) >= med_vot)
#' Conjunto de variáveis de controle
Z_extended <- gsub("\\s+", "+",
                   "pop_density2005 pop2005 nat_EU nat_nonEU fam_marri fam_divor_widow
                   edu_hi edu_lo rel_evan rel_isla rel_orth_other rel_obk pers2 pers3 pers4
                   pers5more  vo_r vo_cl vo_l  j_unempl j_retire j_house j_studen
                   inc_aver age0_30 age30_60  bgld kaern noe ooe salzbg steierm
                   tirol vlbg wien schober")
#' Regressões para efeitos heterogêneos
#' formulas
form_str <- paste0("resp_A~threat+appeal+info+gender+compliance_t+", Z_extended)
#' Default do erro padrão é ser robusto a heterocedasticidade
fixest::setFixest_se(no_FE = "white")
het_reg_pop <- efeito_het_df %>%
  filter(delivered == 1) %>%
  group_by(pop_hi) %>%
  summarise(lm_model = list(feols(as.formula(form_str), data = cur_data())))

het_reg_den <- efeito_het_df %>%
  filter(delivered == 1) %>%
  group_by(popdens_hi) %>%
  summarise(lm_model = list(feols(as.formula(form_str), data = cur_data())))

het_reg_rend <- efeito_het_df %>%
  filter(delivered == 1) %>%
  group_by(rend_hi) %>%
  summarise(lm_model = list(feols(as.formula(form_str), data = cur_data())))

het_reg_vot <- efeito_het_df %>%
  filter(delivered == 1) %>%
  group_by(vot_hi) %>%
  summarise(lm_model = list(feols(as.formula(form_str), data = cur_data())))

#' Tabela 8
esttex(het_reg_pop$lm_model, het_reg_den$lm_model, het_reg_rend$lm_model,
       het_reg_vot$lm_model,
       file = "./Tables/tablec1.tex",
       label = "tab:tabc1",
       style = est_style,
       replace = TRUE,
       se = "White",
       digits = 3,
       fitstat = "",
```

```
       keep = c("^Ameaça$", "^Moral$", "^Info$"),
       order = c("^Ameaça$", "^Moral$", "^Info$"),
       title = "Efeito heterogêneo do tratamento. Modelo de regressão linear.")
```

```
# Metodos de Machine Learning para inferencia causal
# com dados de Fellner et all (2013)
# Autor: Rafael Felipe Bressan
# Arquivo: script_py.py
# Script Python para o trabalho monografico: Inferência Causal com Machine
# Learning uma aplicacao para evasao fiscal
# Pós-Graduacao Lato Sensu em Ciencia de Dados e Big Data PUC-MG
# Ano: 2021

# importa bibliotecas necessarias
import gc
from copy import deepcopy
import numpy as np
import pandas as pd

# Regressao linear e IV
import statsmodels.api as sm
from linearmodels import IV2SLS

# Modelos de ML
import lightgbm as lgb
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LogisticRegression, Lasso
from sklearn.ensemble import RandomForestRegressor

# EconML
from econml.ortho_iv import DMLATEIV, IntentToTreatDRIV
from econml.cate_interpreter import SingleTreeCateInterpreter
from econml.dml import ForestDML, DML, SparseLinearDML
from econml.causal_forest import CausalForest

# Graficos
import matplotlib.pyplot as plt

# Funcoes auxiliares
def stars(x, levels=[0.1, 0.05, 0.01]):
    assert (len(levels)==3), "Comprimento de levels deve ser 3."

    if x>levels[0]:
        return ''
    elif x>levels[1]:
        return '*'
    elif x>levels[2]:
        return '**'
    else:
        return '***'

def format_float(x, digits):
    return '{:.{dig}f}'.format(x, dig=digits)
```

```
def surr_parenthesis(x, digits):
    return '('+'{:.{dig}f}'.format(x, dig=digits)+')'


# Carregando os dados
data = pd.read_stata("data_final.dta").sort_values("treatment")
X_cols = ["gender", "pop_density2005",
    "compliance", "compliance_t", "vo_r", "vo_cr", "vo_cl", "vo_l",
    "inc_aver", "edu_aver", "edu_lo", "edu_mi", "edu_hi",
    "age_aver", "age0_30", "age30_60", "nat_A", "nat_EU", "nat_nonEU"]

######################################################################
# Ignorando o atrito e estimando os efeitos apenas para
# delivered == 1
######################################################################
deliv = data[data["delivered"]!=0].fillna(0)
# Para avaliacao de efeitos heterogeneos
X_eval = (deliv[X_cols]
    .describe()
    .loc[["mean", "min", "25%", "50%", "75%", "max"]]
)
# Tratamentos
t1_deliv = deliv[deliv["treatment"].isin([0,1])]
t2_deliv = deliv[deliv["treatment"].isin([0,2])]
t3_deliv = deliv[deliv["treatment"].isin([0,3])]
t5_deliv = deliv[deliv["treatment"].isin([0,5])]

# Variaveis de interesse
Y1 = t1_deliv["resp_A"].values
T1 = t1_deliv["delivered"].values
X1 = t1_deliv[X_cols].values
X1_treat=X1[T1 == 1]

Y2 = t2_deliv["resp_A"].values
T2 = t2_deliv["delivered"].values
X2 = t2_deliv[X_cols].values
X2_treat=X2[T2 == 1]

Y3 = t3_deliv["resp_A"].values
T3 = t3_deliv["delivered"].values
X3 = t3_deliv[X_cols].values
X3_treat=X3[T3 == 1]

Y5 = t5_deliv["resp_A"].values
T5 = t5_deliv["delivered"].values
X5 = t5_deliv[X_cols].values
X5_treat=X5[T5 == 1]

# ForestDML() = CausalForest()??
# ForestDML eh muito mais rapido que CausalForest com resultados
# semelhantes para a interpretacao via arvore

# DML com regressao logistica para E[T|X,W] e Floresta Aleatoria para
```

```
# E[Y|X,W]. Modelo final para Theta(X) eh escolhido por floresta
dml=ForestDML(
    model_t=LogisticRegression(),
    model_y=RandomForestRegressor(),
    discrete_treatment=True,
    n_estimators=1000,
    subsample_fr=0.7,
    min_samples_leaf=20,
    n_crossfit_splits=3,
    n_jobs=-1
)
# DML para T1
dml.fit(Y1, T1, X1, inference='auto')
dml1_eff=dml.effect(X1, T0=0, T1=1)
dml1_eff_treat=dml.effect(X1_treat, T0=0, T1=1)
print(f"ATE T1 por DML: {np.mean(dml1_eff)}\nATT T1 por DML: {np.mean(dml1_eff_treat)}")
dml1_inf=dml.effect_inference(X_eval.values)
dml1_summary=dml1_inf.summary_frame(alpha=0.05)[["point_estimate", "pvalue", "stderr"]]
dml1_summary.index=X_eval.index
dml1_summary["star"]=dml1_summary["pvalue"].apply(stars)
dml1_summary["point_estimate"]=dml1_summary["point_estimate"].apply(format_float, digits=4)
dml1_summary["point_estimate"]=dml1_summary["point_estimate"].str.cat(dml1_summary["star"])
dml1_summary["stderr"]=dml1_summary["stderr"].apply(surr_parenthesis, digits=4)
dml1_summary=dml1_summary[["point_estimate", "stderr"]].stack()
dml1_summary.name="Correio"

# DML para T2
dml.fit(Y2, T2, X2, inference='auto')
dml2_eff=dml.effect(X2, T0=0, T1=1)
dml2_eff_treat=dml.effect(X2_treat, T0=0, T1=1)
print(f"ATE T2 por DML: {np.mean(dml2_eff)}\nATT T2 por DML: {np.mean(dml2_eff_treat)}")
dml2_inf=dml.effect_inference(X_eval.values)
dml2_summary=dml2_inf.summary_frame(alpha=0.05)[["point_estimate", "pvalue", "stderr"]]
dml2_summary.index=X_eval.index
dml2_summary["star"]=dml2_summary["pvalue"].apply(stars)
dml2_summary["point_estimate"]=dml2_summary["point_estimate"].apply(format_float, digits=4)
dml2_summary["point_estimate"]=dml2_summary["point_estimate"].str.cat(dml2_summary["star"])
dml2_summary["stderr"]=dml2_summary["stderr"].apply(surr_parenthesis, digits=4)
dml2_summary=dml2_summary[["point_estimate", "stderr"]].stack()
dml2_summary.name="Ameaça"

# Interpretacao por arvore de decisao para T2
interp = SingleTreeCateInterpreter(
    include_model_uncertainty=False,
    max_depth=3,
    min_samples_leaf=10)
interp.interpret(dml, X2)
fig, ax1 = plt.subplots(figsize=(25,6))
interp.plot(feature_names=X_cols, fontsize=12, ax=ax1)
fig.savefig("Figs/fig_tree_dml.png")

# DML para T3
dml.fit(Y3, T3, X3, inference='auto')
```

```
dml3_eff=dml.effect(X3, T0=0, T1=1)
dml3_eff_treat=dml.effect(X3_treat, T0=0, T1=1)
print(f"ATE T3 por DML: {np.mean(dml3_eff)}\nATT T3 por DML: {np.mean(dml3_eff_treat)}")
dml3_inf=dml.effect_inference(X_eval.values)
dml3_summary=dml3_inf.summary_frame(alpha=0.05)[["point_estimate", "pvalue", "stderr"]]
dml3_summary.index=X_eval.index
dml3_summary["star"]=dml3_summary["pvalue"].apply(stars)
dml3_summary["point_estimate"]=dml3_summary["point_estimate"].apply(format_float, digits=4)
dml3_summary["point_estimate"]=dml3_summary["point_estimate"].str.cat(dml3_summary["star"])
dml3_summary["stderr"]=dml3_summary["stderr"].apply(surr_parenthesis, digits=4)
dml3_summary=dml3_summary[["point_estimate", "stderr"]].stack()
dml3_summary.name="Info"

# DML para T5
dml.fit(Y5, T5, X5, inference='auto')
dml5_eff=dml.effect(X5, T0=0, T1=1)
dml5_eff_treat=dml.effect(X5_treat, T0=0, T1=1)
print(f"ATE T5 por DML: {np.mean(dml5_eff)}\nATT T5 por DML: {np.mean(dml5_eff_treat)}")
dml5_inf=dml.effect_inference(X_eval.values)
dml5_summary=dml5_inf.summary_frame(alpha=0.05)[["point_estimate", "pvalue", "stderr"]]
dml5_summary.index=X_eval.index
dml5_summary["star"]=dml5_summary["pvalue"].apply(stars)
dml5_summary["point_estimate"]=dml5_summary["point_estimate"].apply(format_float, digits=4)
dml5_summary["point_estimate"]=dml5_summary["point_estimate"].str.cat(dml5_summary["star"])
dml5_summary["stderr"]=dml5_summary["stderr"].apply(surr_parenthesis, digits=4)
dml5_summary=dml5_summary[["point_estimate", "stderr"]].stack()
dml5_summary.name="Moral"

# Melhora consumo de memoria
del dml
collected=gc.collect()
# ATE
dml_ate=[np.mean(x) for x in [dml1_eff, dml2_eff, dml3_eff, dml5_eff]]
# ATT
dml_att=[np.mean(x) for x in
    [dml1_eff_treat, dml2_eff_treat, dml3_eff_treat, dml5_eff_treat]]

treatments_list=["Correio", "Ameaça", "Info", "Moral"]
dml_effect=pd.DataFrame({"ATE": dml_ate, "ATT": dml_att}, index=treatments_list)

# Quantis das variaveis utilizadas para aferir heterogeneidade
X_eval.transpose().to_latex(
    buf="Tables/tab_het_vars.tex",
    decimal=",",
    caption="Variáveis utilizadas para identificar heterogeneidade nos efeitos.",
    label="tab:het-vars"
)

# Sumario com os resultados para os 4 tratamentos
dml_summary=(
    pd.concat(
        [dml1_summary, dml2_summary, dml3_summary, dml5_summary],
        axis=1)
```

```
    .reset_index()
    .rename(columns={"level_0": "X"})
    .drop(columns="level_1")
)

dml_summary.to_latex(
    buf="Tables/tab_dml_summary.tex",
    decimal=",",
    caption="Efeitos heterogêneos do tratamento estimados por Double Machine Learning.",
    label="tab:dml-summary",
    index=False
)
# Nota: os estágios de previsão foram floresta aleatória para $E[Y|\bfx]$
# e regressão logística para $E[T|\bfx]$. O modelo final para o efeito
# condicional do tratamento, $\theta(\bfx)$, é uma floresta aleatória.

######################################################################
# Metodos com variaveis Instrumentais
######################################################################

######################################################################
# Considerando o atrito e estimando os efeitos
######################################################################
iv=data.copy()
iv["delivered"] = iv["delivered"].fillna(0)
# Para avaliacao de heterogeneidade
X_eval = (iv[X_cols]
    .describe()
    .loc[["mean", "min", "25%", "50%", "75%", "max"]]
)
# Tratamentos
t1_iv = iv[iv["treatment"].isin([0,1])]
t2_iv = iv[iv["treatment"].isin([0,2])]
t3_iv = iv[iv["treatment"].isin([0,3])]
t5_iv = iv[iv["treatment"].isin([0,5])]

# Definindo as variaveis
Z1 = t1_iv["treatment"]
T1 = t1_iv["delivered"]
Y1 = t1_iv["resp_A"]
X1 = t1_iv[X_cols]

Z2 = t2_iv["treatment"]
T2 = t2_iv["delivered"]
Y2 = t2_iv["resp_A"]
X2 = t2_iv[X_cols]

Z3 = t3_iv["treatment"]
T3 = t3_iv["delivered"]
Y3 = t3_iv["resp_A"]
X3 = t3_iv[X_cols]

Z5 = t5_iv["treatment"]
```

```
T5 = t5_iv["delivered"]
Y5 = t5_iv["resp_A"]
X5 = t5_iv[X_cols]

# Modelos para E[Y|X] e E[T|Z,X]
lgb_YX_par = {
    "metric": "rmse",
    "learning_rate": 0.1,
    "num_leaves": 30,
    "max_depth": 5
}

lgb_TXZ_par = {
    "objective": "binary",
    "metric": "auc",
    "learning_rate": 0.1,
    "num_leaves": 30,
    "max_depth": 5
}

lgb_theta_par = {
    "metric": "rmse",
    "learning_rate": 0.1,
    "num_leaves": 30,
    "max_depth": 3
}

modelTXZ = lgb.LGBMClassifier(**lgb_TXZ_par)
modelYX = lgb.LGBMRegressor(**lgb_YX_par)
modelZX = lgb.LGBMClassifier(**lgb_TXZ_par)
# Modelo inicial para o efeito heterogeneo
# Sera melhorado pelo algoritmo Double Robust IV
pre_theta = lgb.LGBMRegressor(**lgb_theta_par)

## Modelo 2-stages Least Squares

# Variaveis com amostra completa
Z = iv[["mailing", "threat", "info", "appeal", "i_tinf", "i_tapp"]]
T = Z.multiply(iv["delivered"], axis=0)
Y = iv["resp_A"]
model_2sls = IV2SLS(Y, exog=np.ones(len(Y)), endog=T, instruments=Z)
iv2sls_fit = model_2sls.fit(debiased=True)
iv2sls_params=iv2sls_fit.params[["mailing", "threat", "info", "appeal"]]
# Soma os valores para ter mesma interpretacao que DML e DRIV
iv2sls_params["threat"]=iv2sls_params["threat"]+iv2sls_params["mailing"]
iv2sls_params["info"]=iv2sls_params["info"]+iv2sls_params["mailing"]
iv2sls_params["appeal"]=iv2sls_params["appeal"]+iv2sls_params["mailing"]
treatments_list=["Correio", "Ameaça", "Info", "Moral"]
iv2sls_effect=pd.DataFrame({"LATE": iv2sls_params.values}, index=treatments_list)

## Modelo DRIV

# Treina o modelo DRIV mais flexivel. Theta(X) pode ser um modelo
```

```
# flexivel (nao parametrico, ie. floresta aleatoria) de X
# ATENCAO: leva bastante tempo para rodar
driv1 = IntentToTreatDRIV(
    model_Y_X=modelYX,
    model_T_XZ=modelTXZ,
    flexible_model_effect=pre_theta,
    n_splits=3,
    featurizer=None #PolynomialFeatures(degree=1, include_bias=False)
)
# Mesmo modelo para cada tratamento
driv2=deepcopy(driv1)
driv3=deepcopy(driv1)
driv5=deepcopy(driv1)

# DRIV para T1
print("Iniciando fit de DRIV T1\n")
driv1.fit(Y1, T1, Z=Z1, X=X1, inference="bootstrap")
print("Fim do fit de DRIV T1\n")
driv1_eff=driv1.effect(X1, T0=0, T1=1)
print(f"LATE T1 por DRIV: {np.mean(driv1_eff)}")
print("Iniciando inferencia de DRIV T1\n")
driv1_inf=driv1.effect_inference(X_eval)
print("Fim da inferencia de DRIV T1\n")
driv1_summary=driv1_inf.summary_frame(alpha=0.05)[["point_estimate", "pvalue", "stderr"]]
driv1_summary.index=X_eval.index
driv1_summary["star"]=driv1_summary["pvalue"].apply(stars)
driv1_summary["point_estimate"]=driv1_summary["point_estimate"].apply(format_float, digits=4)
driv1_summary["point_estimate"]=driv1_summary["point_estimate"].str.cat(driv1_summary["star"])
driv1_summary["stderr"]=driv1_summary["stderr"].apply(surr_parenthesis, digits=4)
driv1_summary=driv1_summary[["point_estimate", "stderr"]].stack()
driv1_summary.name="Correio"
# Melhora o consumo de memoria
del driv1
collected=gc.collect()

# DRIV para T2
print("Iniciando fit de DRIV T2\n")
driv2.fit(Y2, T2, Z=Z2, X=X2, inference="bootstrap")
print("Fim do fit de DRIV T2\n")
driv2_eff=driv2.effect(X2, T0=0, T1=1)
print(f"LATE T2 por DRIV: {np.mean(driv2_eff)}")
print("Iniciando inferencia de DRIV T2\n")
driv2_inf=driv2.effect_inference(X_eval)
print("Fim da inferencia de DRIV T2\n")
driv2_summary=driv2_inf.summary_frame(alpha=0.05)[["point_estimate", "pvalue", "stderr"]]
driv2_summary.index=X_eval.index
driv2_summary["star"]=driv2_summary["pvalue"].apply(stars)
driv2_summary["point_estimate"]=driv2_summary["point_estimate"].apply(format_float, digits=4)
driv2_summary["point_estimate"]=driv2_summary["point_estimate"].str.cat(driv2_summary["star"])
driv2_summary["stderr"]=driv2_summary["stderr"].apply(surr_parenthesis, digits=4)
driv2_summary=driv2_summary[["point_estimate", "stderr"]].stack()
driv2_summary.name="Ameaça"
# Interpretacao causal por arvore de decisao
```

```
interp = SingleTreeCateInterpreter(
    include_model_uncertainty=False,
    max_depth=3,
    min_samples_leaf=10
)
interp.interpret(driv2, X2)
fig, ax1 = plt.subplots(figsize=(25,6))
interp.plot(feature_names=X2.columns, fontsize=12, ax=ax1)
fig.savefig("Figs/fig_tree_driv.png")
# Melhora o consumo de memoria
del driv2
collected=gc.collect()

# DRIV para T3
driv3.fit(Y3, T3, Z=Z3, X=X3, inference="bootstrap")
driv3_eff=driv3.effect(X3, T0=0, T1=1)
print(f"LATE T3 por DRIV: {np.mean(driv3_eff)}")
driv3_inf=driv3.effect_inference(X_eval)
print("Fim da inferencia de DRIV T3\n")
driv3_summary=driv3_inf.summary_frame(alpha=0.05)[["point_estimate", "pvalue", "stderr"]]
driv3_summary.index=X_eval.index
driv3_summary["star"]=driv3_summary["pvalue"].apply(stars)
driv3_summary["point_estimate"]=driv3_summary["point_estimate"].apply(format_float, digits=4)
driv3_summary["point_estimate"]=driv3_summary["point_estimate"].str.cat(driv3_summary["star"])
driv3_summary["stderr"]=driv3_summary["stderr"].apply(surr_parenthesis, digits=4)
driv3_summary=driv3_summary[["point_estimate", "stderr"]].stack()
driv3_summary.name="Info"
# Melhora o consumo de memoria
del driv3
collected=gc.collect()

# DRIV para T5
driv5.fit(Y5, T5, Z=Z5, X=X5, inference="bootstrap")
driv5_eff=driv5.effect(X5, T0=0, T1=1)
print(f"LATE T5 por DRIV: {np.mean(driv5_eff)}")
driv5_inf=driv5.effect_inference(X_eval)
print("Fim da inferencia de DRIV T5\n")
driv5_summary=driv5_inf.summary_frame(alpha=0.05)[["point_estimate", "pvalue", "stderr"]]
driv5_summary.index=X_eval.index
driv5_summary["star"]=driv5_summary["pvalue"].apply(stars)
driv5_summary["point_estimate"]=driv5_summary["point_estimate"].apply(format_float, digits=4)
driv5_summary["point_estimate"]=driv5_summary["point_estimate"].str.cat(driv5_summary["star"])
driv5_summary["stderr"]=driv5_summary["stderr"].apply(surr_parenthesis, digits=4)
driv5_summary=driv5_summary[["point_estimate", "stderr"]].stack()
driv5_summary.name="Moral"
# Melhora o consumo de memoria
del driv5
collected=gc.collect()

# LATE
driv_late=[np.mean(x) for x in
    [driv1_eff, driv2_eff, driv3_eff, driv5_eff]]
treatments_list=["Correio", "Ameaça", "Info", "Moral"]
```

```
driv_effect=pd.DataFrame({"LATE": driv_late}, index=treatments_list)
# driv_effect.to_latex(
#     buf="Tables/tab_driv_late.tex",
#     decimal=",",
#     caption="LATE estimado por Doubly Robust IV para diferentes tratamentos.",
#     label="tab:driv-late"
# )

# Junta todos os efeitos em uma tabela unica
all_effects=(dml_effect
    .merge(iv2sls_effect, left_index=True, right_index=True)
    .merge(driv_effect, left_index=True, right_index=True)
)
mindex=pd.MultiIndex.from_tuples([
    ("ForestDML", "ATE"),
    ("ForestDML", "ATT"),
    ("IV2SLS", "LATE"),
    ("DRIV", "LATE")],
    names=["Modelo", "Efeito"])
all_effects.columns=mindex
all_effects.to_latex(
    buf="Tables/tab_all_effects.tex",
    decimal=",",
    caption="Efeitos médios dos tratamentos estimados pelos métodos de ForestDML, IV2SLS e DRIV.",
    label="tab:all-effects",
    float_format="%.4f",
    multirow=True,
    multicolumn=True,
    multicolumn_format="c"
)

# Sumario com os resultados para os 4 tratamentos
driv_summary=(
    pd.concat(
        [driv1_summary, driv2_summary, driv3_summary, driv5_summary],
        axis=1)
    .reset_index()
    .rename(columns={"level_0": "X"})
    .drop(columns="level_1")
)

driv_summary.to_latex(
    buf="Tables/tab_driv_summary.tex",
    decimal=",",
    caption="Efeitos heterogêneos do tratamento estimados por Doubly Robust IV.",
    label="tab:driv-summary",
    index=False
)
# Nota: os estágios de previsão foram gradient boosted tree (regressão) para $E[Y|\bfx]$
# e gbm (classificação) $E[T|\bfx]$. O modelo final para o efeito
# condicional do tratamento, $\theta(\bfx)$, também foi uma gbm-regressão, porém mais rasa,
# com apenas 3 níveis de profundidade.
```

```
# Salva objetos
# with open("modelos.pkl", "wb") as f:
#     pickle.dump([model_dml, model_ldriv, model_driv], f)

# Carrega objetos
# with open("modelos.pkl", "wb") as f:
#     model_dml, model_ldriv, model_driv = pickle.load(f)


# Floresta Causal
# cf=CausalForest(
#     n_trees=200,
#     min_leaf_size=15,
#     max_depth=8,
#     model_T=LogisticRegression(),
#     model_Y=RandomForestRegressor(),
#     discrete_treatment=True,
#     random_state=123
# )
# cf.fit(Y, T, X)
# cf_inf=cf.effect_inference(X_eval.values)
# cf_const_eff=cf.const_marginal_effect(X)
# print(f"ATE por Causal Forest {np.mean(cf_const_eff)}")
# # cf_eff=cf.effect(X_eval.values)
# # cf_lb, cf_ub=cf.effect_interval(X_eval.values)
# # cf_dict={"Estimate": cf_eff, "LB": cf_lb, "UB": cf_ub}
# # cf_df=pd.DataFrame(cf_dict, index=X_eval.index)
# cf_summary=cf_inf.summary_frame(alpha=0.05)
# cf_summary.index=X_eval.index
# cf_summary
# cf_summary.to_latex(
#     buf="Tables/tab_cf_summary.tex",
#     decimal=",",
#     caption="Efeitos heterogêneos do tratamento T1 estimados por Floresta Causal.",
#     label="tab:cf-summary"
# )
# # Floresta causal nao eh muito robusta a especificacao do modelo
# # E[Y|X,W] (Random Forest ou Lasso)

# # Interpretacao causal por arvore de decisao
# interp = SingleTreeCateInterpreter(
#     include_model_uncertainty=False,
#     max_depth=3, min_samples_leaf=10)
# interp.interpret(cf, X)
# fig, ax1 = plt.subplots(figsize=(25,6))
# interp.plot(feature_names=X_cols, fontsize=12, ax=ax1)
# fig.savefig("Figs/fig_tree_cf.png")

# ## Modelo DMLATEIV
# model_dml = DMLATEIV(
#     model_Y_W=modelYX,
#     model_T_W=modelTXZ,
#     model_Z_W=modelZX,
```

```
#     discrete_treatment=True,
#     discrete_instrument=True,
#     n_splits=5
# )
# model_dml.fit(Y, T, Z, W=None, inference="bootstrap")
# model_dml.const_marginal_effect_interval(alpha=0.05)
```