ECON293_Final_Project

2022-05-25

Role of Default in Probability of Tipping: An ML Approach

Yuejun Chen, Bruno Escobar, Christine Kim, Ella Mao

June 1, 2022

Introduction

Our study examines the default effects by looking at the setting of tipping taxi drivers in New York City. Specifically, we seek to understand whether a higher default tip suggestion would change customer's probability of tipping. This setting offers two unique benefits. First, it provides a setting where consumers are facing the choice sets regularly. Second, the high-frequency nature of tipping results in a large dataset, which enables researchers to apply machine learning methods.

Data Cleaning

```
# Step 1: Cleaning -----
## Read in the stata version of the data
tips2009_vendor_1218 <- read_dta(paste0(Raw_Data,"tips2009_clean.dta"))

## Subset the data according to the following rules:
## 1. Rides from the vendors only
## 2. Fare between 12 to 18 dollars
tips2009_vendor_1218 <- tips2009_vendor_1218 %>% filter(vendor == 1 & fare >= 12 & fare <= 18)

# Step 2: Create additional categorical variables for the covariates  -----
classify_cov <- function(data){
  ## Categorize day of week
  data2 <- data %>%
    mutate(weekend = ifelse(pkp_dow >= 1 & pkp_dow <= 5, 0, 1))
  ## Categorize time of the day
  data3 <- data2 %>%
    mutate(pickup_time_group = ifelse(pkp_hour >= 6 & pkp_hour <= 12, 1,
                                      ifelse(pkp_hour >= 13 & pkp_hour <= 16, 2,
                                             ifelse(pkp_hour >= 17 & pkp_hour <= 20, 3, 0))))
  ## Convert the pick up locations to numeric, because the causal forest package does not support
  ## non-numeric values
  data4 <- data3 %>%
    mutate(Manhattan_pkp = ifelse(pkp_boro == "Manhattan", 1,0),
           Brooklyn_pkp = ifelse(pkp_boro == "Brooklyn", 1,0),
           Queens_pkp = ifelse(pkp_boro == "Queens", 1,0),
           Bronx_pkp = ifelse(pkp_boro == "The Bronx", 1,0),
           Staten_pkp = ifelse(pkp_boro == "Staten Island", 1,0),
           Other_pkp = ifelse(pkp_boro == "", 1,0))
```

```r
  ## Convert the drop off locations to numeric
  data5 <- data4 %>%
    mutate(Manhattan_drf = ifelse(drf_boro == "Manhattan", 1,0),
           Brooklyn_drf = ifelse(drf_boro == "Brooklyn", 1,0),
           Queens_drf = ifelse(drf_boro == "Queens", 1,0),
           Bronx_drf = ifelse(drf_boro == "The Bronx", 1,0),
           Staten_drf = ifelse(drf_boro == "Staten Island", 1,0),
           Other_drf = ifelse(drf_boro == "", 1,0))

  data_fnl <- data5
  return(data_fnl)
}

tips2009_1218_regroup <- classify_cov(tips2009_vendor_1218)

# Step 3: Save the data -----
write.csv(tips2009_1218_regroup,paste0(In_Data,"fare_1218_recoded_final.csv"))
```

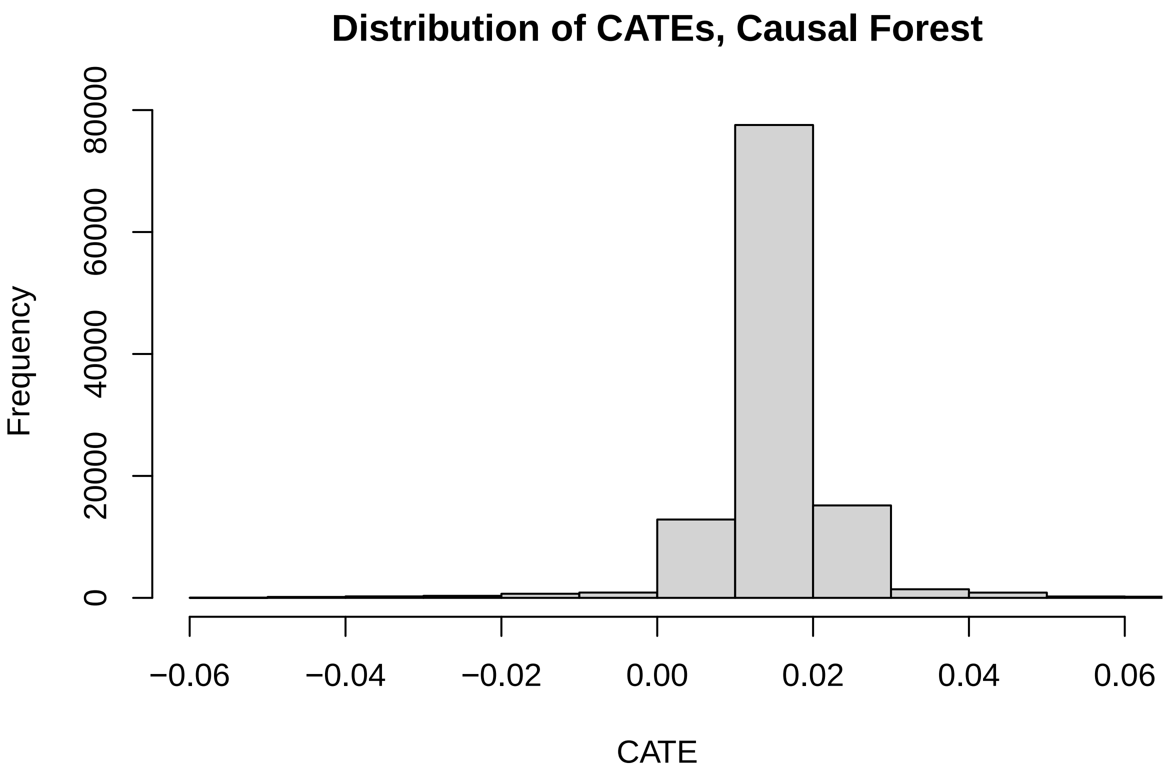
Methods

```r
# Step 1: Import the data -----
tips2009_1218 <- read.csv(paste0(In_Data,"fare_1218_recoded_final.csv"))
# Create subsets
reduced=1
if (reduced == 1){
  data = subset(tips2009_1218, fare>=14 & fare<=16)
  suffix = "r1"
}
if (reduced == 0){
  data = subset(tips2009_1218, fare>=12 & fare<=18)
  suffix = "r0"
}

# Step 2: Set up variables -----
n <- nrow(data)
# Treatment: Whether the fare amount is above or below 15 dollars
treatment <- "dsc_15"
# Outcome: Whether someone tips 0. 1 for yes, 0 for no.
outcome <- "tip_zero"
running = "fare"
# Additional covariates
covariates <- c("weekend", "pickup_time_group", "gr_inc10_All", "Manhattan_pkp", "Brooklyn_pkp", "Queens

# Step 3: Split the data into training and testing -----
split1<- sample(c(rep(0, 0.7 * nrow(data)), rep(1, 0.3 * nrow(data))))
data.train <- data[split1 == 0,]
data.test <- data[split1 == 1,]
# Training code
W_train <- data.train[,treatment]
Y_train <- data.train[,outcome]
X_train <- data.train[,covariates]
Z1_train = (data.train[,running]-15)*W_train
Z0_train = (data.train[,running]-15)*(1-W_train)
```

```r
# Test code
W_test <- data.test[,treatment]
Y_test <- data.test[,outcome]
X_test <- data.test[,covariates]
Z1_test = (data.test[,running]-15)*W_test
Z0_test = (data.test[,running]-15)*(1-W_test)
```

Causal Forest

```r
# Implement causal forest in grf
cf.priority = causal_forest(X_train, Y_train, W_train, num.trees = 100)
priority.cate <- predict(cf.priority, X_test)$predictions

## Estimate the causal forest on the test data
cf.eval <- causal_forest(X_test, Y_test, W_test)

## Estimate the ATE for the whole population
cf.ATE.mean <- mean(priority.cate)
cf.ATE.SD <- sd(priority.cate)
ATE_stats <- c(cf.ATE.mean,cf.ATE.SD)
write.csv(ATE_stats,paste0(Output, "Causal_Forest_sumstats.csv"),row.names=F)

# Show the CATE distribution
hist(priority.cate, main = "Distribution of CATEs, Causal Forest", xlab = "CATE", xlim=c(-0.06,0.06))
```




```r
png(file=paste0(Output, "Causal_Forest_CATE.png"),width=595, height=545)

## Plot QINI curve - using rank_average_treatment_effect
rate <- rank_average_treatment_effect(cf.eval, priority.cate, target = "QINI")
```

```r
png(file=paste0(Output, "Causal_Forest_QINI.png"),width=595, height=545)
plot(rate)
print(rate)
```

```
##       estimate      std.err            target
##  -0.0002170781 0.0003422336 priorities | QINI
```

```r
## Create quartile plots
cf.tau.hat <- priority.cate
strata <- 10
quintiles <- quantile(cf.tau.hat, prob=seq(from=0,to=1,by=1/strata))
cf.tau.hat.quartiles <- cut(cf.tau.hat, breaks = quintiles, labels = 1:strata, include.lowest = TRUE)
## Combine all the data together
combined_data <- as.data.frame(cbind(data.test,cf.tau.hat,cf.tau.hat.quartiles))
## Store tau estimates by quartiles
CATE_test <- combined_data %>% group_by(cf.tau.hat.quartiles) %>%
  summarise(tau.mean = mean(cf.tau.hat,na.rm = TRUE),
            tau.sd = sd(cf.tau.hat,na.rm = TRUE))
## Estimate ATE for each split
ATE.df <- data.frame(ATE=double(),
                     SE=integer())

## Create the calibration plot
for (i in 1:strata){
  temp_data <- combined_data %>% filter(cf.tau.hat.quartiles == i)
  rd_lm <- temp_data %$%
    lm(tip_zero ~ dsc_15 + I(fare - 15) + dsc_15:I(fare - 15))
  ATE.df[i,"ATE"] <- rd_lm$coefficients["dsc_15"]
  ATE.df[i,"SE"] <- summary(rd_lm)$coefficients[2,2]
}

## CI
ATE.df2 <- ATE.df %>% mutate(CI_max = ATE + 1.96*SE, CI_min = ATE - 1.96*SE)
## Combine the dataset
ATE.df.fnl <- cbind(ATE.df2,CATE_test)


cali_plot <- ggplot(ATE.df.fnl, aes(tau.mean, ATE)) +
  geom_point() +
  geom_errorbar(aes(ymin = CI_max, ymax = CI_min)) +
  theme_light() + labs(x = "Tau hat", y = "ATE", title = "")

ggsave(file=paste0(Output, "Calibration_by_quartile.png"), plot = cali_plot, width=8, height=4.94)

cali_plot
```

X Learner

```r
# Implement the x learner on the testing data
tf0 = regression_forest(X_train[W_train==0,], Y_train[W_train==0], num.trees = 100)
yhat0 = predict(tf0, X_train[W_train==1,])$predictions
xf1 = regression_forest(X_train[W_train==1,], Y_train[W_train==1]-yhat0, num.trees = 100)
xf.preds.1 = predict(xf1, X_test)$predictions
xf.preds.1[W_test==1] = predict(xf1,X_test[W_test==1,])$predictions
```

```r
tf1 = regression_forest(X_train[W_train==1,], Y_train[W_train==1], num.trees = 100)
yhat1 = predict(tf1, X_train[W_train==0,])$predictions
xf0 = regression_forest(X_train[W_train==0,], yhat1-Y_train[W_train==0], num.trees = 100)
xf.preds.0 = predict(xf0, X_test)$predictions
xf.preds.0[W_test==0] = predict(xf0,X_test[W_test==0,])$predictions
propf = regression_forest(X_test, W_test)
ehat = predict(propf)$predictions
preds.xf = (1 - ehat) * xf.preds.1 + ehat * xf.preds.0

## Estimate the ATE for the whole population
xf.ATE.mean <- mean(preds.xf)
xf.ATE.SD <- sd(preds.xf)
ATE_stats <- c(xf.ATE.mean,xf.ATE.SD)
write.csv(ATE_stats,paste0(Output, "X_learner_sumstats.csv"),row.names=F)

## Plot QINI curve - using rank_average_treatment_effect
rate <- rank_average_treatment_effect(cf.eval, preds.xf, target = "QINI")
png(file=paste0(Output, "X_Learner_QINI.png"),width=595, height=545)
plot(rate)
print(rate)
```

```
##       estimate      std.err            target
##  -0.0001572807 0.0003460642 priorities | QINI
```

```r
dev.off()
```

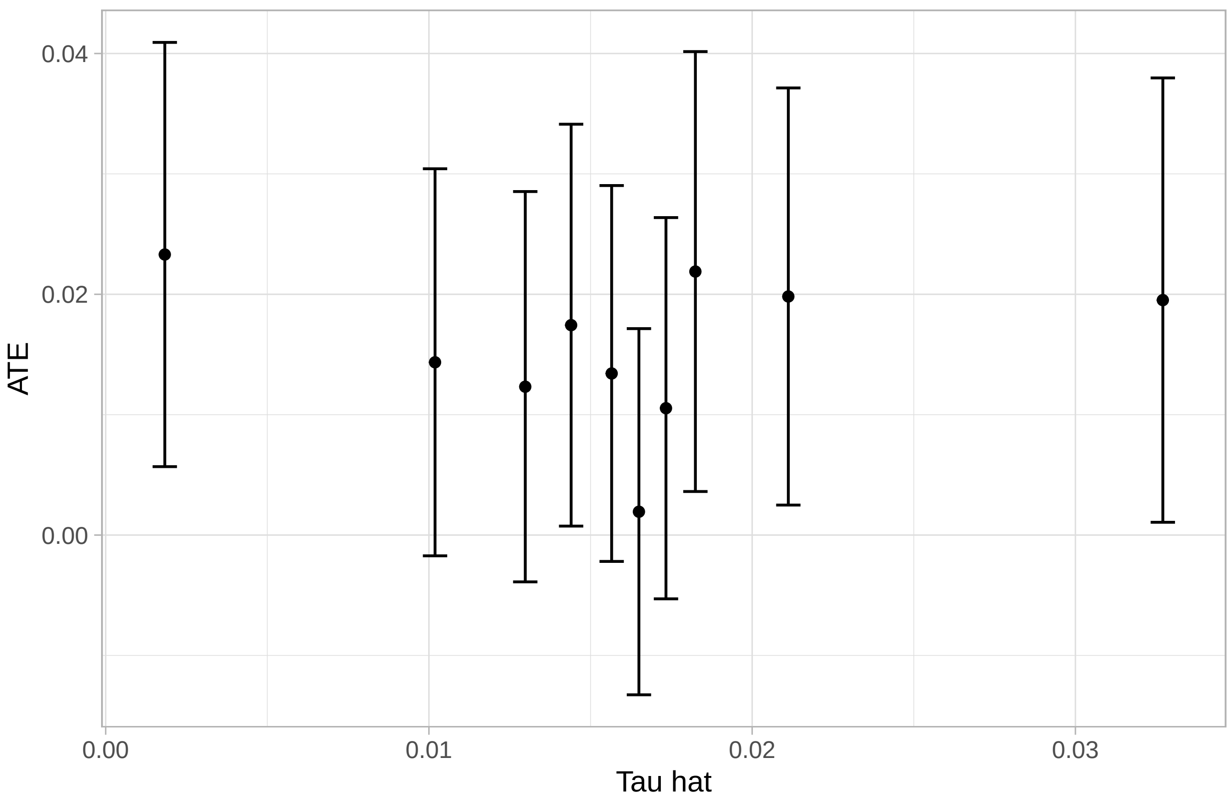
```
## pdf
##   2
```

```r
# Show the CATE distribution
png(file=paste0(Output, "X_Learner_CATE.png"),width=595, height=545)
hist(ehat, main = "", xlab = "CATE")
dev.off()
```

```
## pdf
##   2
```

```r
## Create quartile plots
xf.tau.hat <- preds.xf
strata <- 10
quintiles <- quantile(xf.tau.hat, prob=seq(from=0,to=1,by=1/strata))
xf.tau.hat.quartiles <- cut(xf.tau.hat, breaks = quintiles, labels = 1:strata, include.lowest = TRUE)
## Combine all the data together
combined_data <- as.data.frame(cbind(data.test,xf.tau.hat,xf.tau.hat.quartiles))
## Store tau estimates by quartiles
CATE_test <- combined_data %>% group_by(xf.tau.hat.quartiles) %>%
  summarise(tau.mean = mean(xf.tau.hat,na.rm = TRUE),
            tau.sd = sd(xf.tau.hat,na.rm = TRUE))
## Estimate ATE for each split
ATE.df <- data.frame(ATE=double(),
                  SE=integer())

## Create the calibration plot
for (i in 1:strata){
  temp_data <- combined_data %>% filter(xf.tau.hat.quartiles == i)
  rd_lm <- temp_data %$%
```

```
    lm(tip_zero ~ dsc_15 + I(fare - 15) + dsc_15:I(fare - 15))
  ATE.df[i,"ATE"] <- rd_lm$coefficients["dsc_15"]
  ATE.df[i,"SE"] <- summary(rd_lm)$coefficients[2,2]
}

## CI
ATE.df2 <- ATE.df %>% mutate(CI_max = ATE + 1.96*SE, CI_min = ATE - 1.96*SE)
## Combine the dataset
ATE.df.fnl <- cbind(ATE.df2,CATE_test)


cali_plot <- ggplot(ATE.df.fnl, aes(tau.mean, ATE)) +
  geom_point() +
  geom_errorbar(aes(ymin = CI_max, ymax = CI_min)) +
  theme_light() + labs(x = "Tau hat", y = "ATE", title = "")

ggsave(file=paste0(Output, "Calibration_by_quartile.png"), plot = cali_plot, width=8, height=4.94)

cali_plot
```



LM Forest

```
## LM forest
lmf = lm_forest(X_train, Y_train, cbind(W_train, Z0_train, Z1_train), num.tree = 100)
lm.tau.hat <- predict(lmf,X_test)$predictions[, 1, ]

## Estimate the ATE for the whole population
lm.ATE.mean <- mean(lm.tau.hat)
```

```r
lm.ATE.SD <- sd(lm.tau.hat)
ATE_stats <- c(lm.ATE.mean,lm.ATE.SD)
write.csv(ATE_stats,paste0(Output, "LM_forest_sumstats_", suffix,".csv"),row.names=F)

## Plot QINI curve - using rank_average_treatment_effect
rate <- rank_average_treatment_effect(cf.eval, lm.tau.hat, target = "QINI")
png(file=paste0(Output, "LM_forest_QINI_", suffix,".png"),width=595, height=545)
plot(rate)
print(rate)
```

```
##      estimate     std.err           target
##  0.0003823195 0.000368536 priorities | QINI
```

```r
dev.off()
```

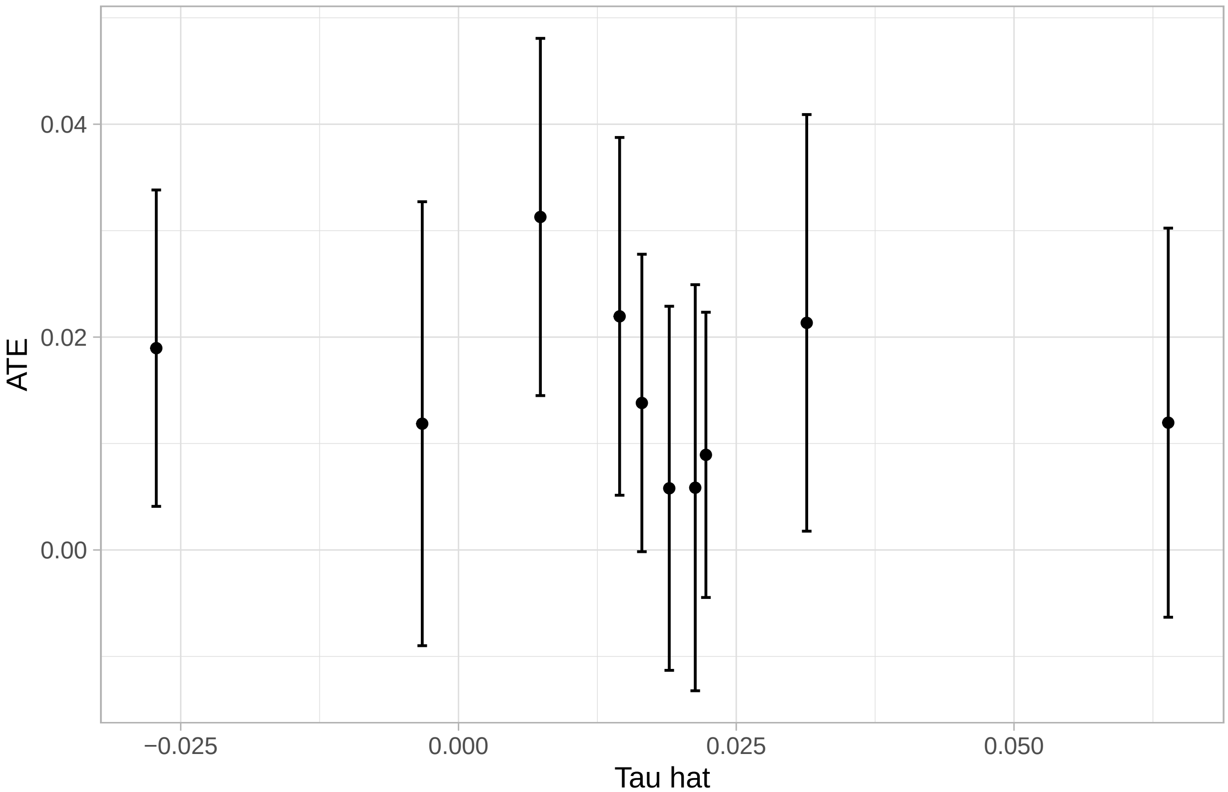
```
## pdf
##   2
```

```r
# Show the CATE distribution
png(file=paste0(Output, "LM_forest_CATE_", suffix,".png"),width=595, height=545)
hist(lm.tau.hat, main = "", xlab = "CATE")
dev.off()
```

```
## pdf
##   2
```

```r
## Create quantile plots ----
strata <- 10
quintiles <- quantile(lm.tau.hat, prob=seq(from=0,to=1,by=1/strata))
lm.tau.hat.quartiles <- cut(lm.tau.hat, breaks = quintiles, labels = 1:strata, include.lowest = TRUE)

## Combine all the data together
combined_data <- as.data.frame(cbind(data.test,lm.tau.hat,lm.tau.hat.quartiles))

## Store tau estimates by quartiles
CATE_test <- combined_data %>% group_by(lm.tau.hat.quartiles) %>%
  summarise(tau.mean = mean(lm.tau.hat,na.rm = TRUE),
            tau.sd = sd(lm.tau.hat,na.rm = TRUE))

## Estimate ATE for each split
ATE.df <- data.frame(ATE=double(),
                     SE=integer())

## Create the calibration plot
for (i in 1:strata){
  temp_data <- combined_data %>% filter(lm.tau.hat.quartiles == i)
  rd_lm <- temp_data %$%
    lm(tip_zero ~ dsc_15 + I(fare - 15) + dsc_15:I(fare - 15))
  ATE.df[i,"ATE"] <- rd_lm$coefficients["dsc_15"]
  ATE.df[i,"SE"] <- summary(rd_lm)$coefficients[2,2]
}

## CI
ATE.df2 <- ATE.df %>% mutate(CI_max = ATE + 1.96*SE, CI_min = ATE - 1.96*SE)

## Combine the dataset
```

```r
ATE.df.fnl <- cbind(ATE.df2,CATE_test)

## GGplot
cali_plot <- ggplot(ATE.df.fnl, aes(tau.mean, ATE)) +
  geom_point() +
  geom_errorbar(aes(ymin = CI_max, ymax = CI_min)) +
  theme_light() + labs(x = "Tau hat", y = "ATE", title = "")

ggsave(file=paste0(Output,"Calibration_by_quartile_LM_forest_", suffix,".png"), plot = cali_plot, width=

cali_plot
```



Create HTE plots by categoircal variables

```r
HTE_plot <- function(lm.tau.hat,data.test,factor, factor_label){
  ## Combine all the data together
  combined_data <- as.data.frame(cbind(data.test,lm.tau.hat))
  ## Store tau estimates by drop off borough
  CATE_test <- combined_data %>% group_by(get(factor)) %>%
    summarise(tau.mean = mean(lm.tau.hat,na.rm = TRUE),
              tau.sd = sd(lm.tau.hat,na.rm = TRUE))
  ## ATE storage df
  ATE.df <- data.frame(ATE=double(),
                       SE=integer(),
                       strata = integer(),
                       method = c())
  ## Estimate Causal Forest for each split
```

```r
CausalF.df <- data.frame(ATE=double(),
                         SE=integer(),
                         strata = integer(),
                         method = c())
## LM forest storage df
LMforest.df <- data.frame(ATE=double(),
                          SE=integer(),
                          strata = integer(),
                          method = c())

## Create the calibration plot
column_of_interest <- combined_data[,colnames(combined_data) == factor]
strata <- length(unique(column_of_interest))

## Store tau hats from each method for plotting purposes
for (i in 1:strata){
  temp_data <- combined_data %>% filter(get(factor) == sort(unique(column_of_interest))[i])
  ## RD
  rd_lm <- temp_data %$%
    lm(tip_zero ~ dsc_15 + I(fare - 15) + dsc_15:I(fare - 15))
  ATE.df[i,"ATE"] <- rd_lm$coefficients["dsc_15"]
  ATE.df[i,"SE"] <- coeftest(rd_lm, vcov=vcovHC(rd_lm, "HC2"))[2,2]
  ATE.df[i,"strata"] <- i
  ATE.df[i,"method"] <- "Regression Discontinuity"
  ## Causal Forest
  forest <- causal_forest(X=temp_data[,covariates],W=temp_data[,treatment],Y=temp_data[,outcome],num.
  forest.ate <- average_treatment_effect(forest)
  CausalF.df[i,"ATE"] <- forest.ate[1]
  CausalF.df[i,"SE"] <- forest.ate[2]
  CausalF.df[i,"strata"] <- i
  CausalF.df[i,"method"] <- "Causal Forest"
  ## LM forest
  Wlm = temp_data[,treatment]
  Zlm = temp_data$fare - 15
  Z1lm= Zlm*Wlm
  Z0lm= Zlm*(1-Wlm)
  lmforest = lm_forest(X=temp_data[,covariates], Y=temp_data[,outcome], W=cbind(Wlm, Z0lm, Z1lm), num
  tau.temp <- predict(lmforest)$predictions[, 1, ]
  LMforest.df[i,"ATE"] <-mean(tau.temp)
  LMforest.df[i,"SE"] <- sqrt(var(tau.temp) / length(tau.temp))
  LMforest.df[i,"strata"] <- i
  LMforest.df[i,"method"] <- "LM Forest"
}

## Combine the tau hats from all methods
res <- rbind(ATE.df, CausalF.df, LMforest.df)
#res <- rbind(LMforest.df)

## Recode values in strata
for (i in 1:strata){
  res$strata[res$strata == i] <- factor_label[i]
}
```

```
  calibration_plot <- ggplot(res) +
    aes(x = strata, y = ATE, group=method, color=method) +
    geom_point(position=position_dodge(0.2)) +
    geom_errorbar(aes(ymin=ATE-2*SE , ymax=ATE+2*SE), width=.2, position=position_dodge(0.2)) +
    ylab("") + xlab("") +
    ggtitle("Average CATE within each ranking (as defined by predicted CATE)") +
    theme_light() +
    theme(legend.position="bottom", legend.title = element_blank())

  return(calibration_plot)
}

data.test.drfboro <- data.test %>% filter(drf_boro != "")
lm.tau.hat.drfboro <- lm.tau.hat[data.test$drf_boro != ""]

data.test.noNA <- data.test %>% filter(!is.na(gr_inc10_All))
lm.tau.hat2 <- lm.tau.hat[!is.na(data.test$gr_inc10_All)]

## Generate plots of HTE for known groups

# Weekends vs weekdays
sort(unique(data.test.drfboro$weekend))
```

```
## [1] 0 1
```

```
plot_weekend <- HTE_plot(lm.tau.hat,data.test,factor = "weekend", factor_label = c("Weekday", "Weekend"
ggsave(file=paste0(Output, "HTE_by_weekend_LM_forest_",suffix,".png"), plot = plot_weekend, width=8, he

# Morning vs afternoon
sort(unique(data.test.drfboro$pickup_time_group))
```

```
## [1] 1 2 3
```

```
plot_TimeofDay <- HTE_plot(lm.tau.hat,data.test,factor = "pickup_time_group", factor_label = c('Morning
ggsave(file=paste0(Output, "HTE_by_timeofday_LM_forest_",suffix,".png"), plot = plot_TimeofDay, width=8

# Income deciles
sort(unique(data.test.drfboro$gr_inc10_All))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
plot_Income <- HTE_plot(lm.tau.hat2,data.test.noNA,factor = "gr_inc10_All", factor_label = c("1","2","3
ggsave(file=paste0(Output, "HTE_by_income_LM_forest_",suffix,".png"), plot = plot_Income, width=8, heig

# Borough
sort(unique(data.test.drfboro$drf_boro))
```

```
## [1] "Brooklyn"      "Manhattan"     "Queens"        "Staten Island"
## [5] "The Bronx"
```

```
plot_drfboro <- HTE_plot(lm.tau.hat = lm.tau.hat.drfboro,data.test = data.test.drfboro,factor = "drf_bo
ggsave(file=paste0(Output, "HTE_by_dropoff_LM_forest_",suffix,".png"), plot = plot_drfboro, width=8, he
```