

# Plots of the restaurants analysis

## Contents

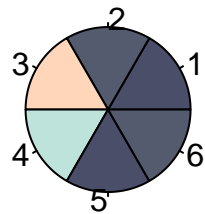
<b>1</b>	<b>Setup the enviroment</b>	<b>2</b>
<b>2</b>	<b>Definition of necessary functions</b>	<b>4</b>
2.1	Formalizing labels . . . . .	4
2.2	Getting general percentage improvement . . . . .	4
2.3	Estimating regression lines and getting confidence intervals . . . . .	4
2.4	Generate a ranking performance plot . . . . .	5
2.5	Generate an AUC-n plot . . . . .	6
2.6	Generate a within-opening performance plot . . . . .	7
2.7	Generate a basic AUC plot . . . . .	7
<b>3</b>	<b>Creating the paper's figures</b>	<b>9</b>
3.1	Figure XX: AUC comparison . . . . .	9
3.2	Figure XX: AUC-n comparison . . . . .	10
3.3	Figure XX: Ranking performance . . . . .	10
3.4	Figure XX: Performance within openings . . . . .	11
3.5	Figure XX: AUC comparison – many assessment . . . . .	12

# 1 Setup the enviroment

```
setwd("basicUtils/")
devtools::document()
devtools::install()
devtools::load_all()
setwd("../")
```

```
library(gridExtra)
library(cowplot)
library(grid)
library(basicUtils)
library(tidyverse)
library(ggthemes)
```

```
to_knit = T
#levander blueSky greyGreen green purple grey red blueGrey yellow sparklingGreen orrange beige
l = getColors("greenbeige")
```



```
l
```

```
FALSE $baseColor
FALSE [1] "#4a4e69"
FALSE
FALSE $darkColor
FALSE [1] "#555b6e"
FALSE
FALSE $highlightColor
FALSE [1] "#ffd6ba"
FALSE
FALSE $lightColor
FALSE [1] "#bee3db"
FALSE
FALSE $ultraLight
FALSE NULL
FALSE
FALSE $extraColor
FALSE NULL
```

```

baseColor = l$baseColor
bgColor = l$darkColor
if (to_knit==T){
baseSize = 9
shape_size=2
annot_size =2
line_size=0.5
}else{
baseSize = 21
shape_size=5
annot_size=4
line_size=2
}
l$lightColor

```

```
FALSE [1] "#bee3db"
```

```

highlightColor = l$highlightColor
baseShape = 7
highlightShape = 15
fontTheme = 'sans'
serror = function(x) sqrt(var(x,na.rm = T) / length(x))
bgColor = l$highlightColor

```

## 2 Definition of necessary functions

### 2.1 Formalizing labels

```
update_model_names <- function(k1) {  
  k1$model <- factor(k1$model)  
  levels(k1$model)[levels(k1$model)=="svdbinary"]="SVD\n(implicit)"  
  levels(k1$model)[levels(k1$model)=="svdtrinary"]="SVD\n(p-aware)"  
  levels(k1$model)[levels(k1$model)=="svdtrinaryexplicit"]="SVD\n(explicit)"  
  levels(k1$model)[levels(k1$model)=="explicit_SVD"]="SVD\n(explicit)"  
  levels(k1$model)[levels(k1$model)=="cnn"]="CNN\n(implicit)"  
  levels(k1$model)[levels(k1$model)=="lstm"]="LSTM"  
  levels(k1$model)[levels(k1$model)=="svm"]="SVM"  
  levels(k1$model)[levels(k1$model)=="xg"]="XGBoost"  
  levels(k1$model)[levels(k1$model)=="trinary_SVD"]="SVD\n(p-aware)"  
  levels(k1$model)[levels(k1$model)=="sahoo"]="HMM-CF\n(implicit)"  
  levels(k1$model)[levels(k1$model)=="logit"]="Lgistic\nregression"  
  levels(k1$model)[levels(k1$model)=="rf"]="Random\nforest"  
  levels(k1$model)[levels(k1$model)=="hmm"]="HMM"  
  levels(k1$model)[levels(k1$model)=="fb"]="Reputation\n(implicit)"  
  return(k1)  
}
```

### 2.2 Getting general percentage improvement

```
get_imprvement = function(d,algorithm){  
  new_d = ((d$hmm - d[,algorithm])/d[,algorithm]) * 100  
  return(new_d)  
}
```

### 2.3 Estimating regression lines and getting confidence intervals

```
get_labels = function(r,level){ #regression labels  
  t = tibble()  
  avg = se = 1  
  for(model in unique(r$model)){  
    r[r$model==model,] -> cur_r  
    res = lm(avg ~ n,cur_r)  
    from = confint(res, 'n', level = level)[1]  
    to = confint(res, 'n', level = level)[2]  
    intercept = res$coefficients[1]  
    #position to print the conf-int  
    yposition = ifelse(intercept > 1, 0.9 * intercept,
```

```

      ifelse(intercept > 0, -20 * intercept, 2 * intercept))#intercept
    t %>% bind_rows(tibble(from,to,model,yposition,avg,se)) -> t
  }
return(t)
}

```

## 2.4 Generate a ranking performance plot

```

get_ranking_performance_plot = function() {
  get_focal_imprvs = function(algorithm){
    new_d = get_imprvement(d,algorithm) %>% bind_cols(d %>% select(prc,fold))
    new_d %>% rename(improvement = algorithm) %>% mutate(model = algorithm) %>%
    group_by(model,prc) %>% summarise(avg=mean(improvement),
                                         se = serror(improvement))-> new_d

    return(new_d)
  }
  d = read.csv(paste("../data/evaluation_results/ranking_performance.csv",
                     sep=""))
  baselines = unique(d$algorithm)
  baselines = baselines[baselines != 'hmm']
  d %>% pivot_wider(names_from = algorithm, values_from=score)-> d
  baselines %>% map_dfr(get_focal_imprvs) -> r
  r %>% update_model_names %>% ggplot( aes(x = prc, y = avg,
                                           ymin = (avg - 1.645 * se), ymax = (avg + 1.645 * se),
                                           color = baseColor, fill = '1')) +
    geom_col(alpha=0.4,size=line_size)+
    facet_wrap(~model, ncol = 6, scales = "free_y") +
    geom_errorbar(width = 0.3) +
    scale_color_manual(values = c(baseColor), name = "Improvement over") +
    scale_fill_manual(values = c(bgColor), name = "Improvement over") +
    xlab("Ranked cohort (p)") +
    scale_x_continuous(breaks = c(0.5,1),labels = c('Bottom 50%', 'Top 50%')) +
    geom_hline(yintercept = 0, linetype = "dashed") +
    geom_vline(xintercept = 0.5, linetype = "dashed", alpha = 0.4) +
    ylab(" Performance lift (%)") +
    theme_hc(base_size=baseSize)+
    theme(strip.background =element_rect(fill=alpha(bgColor,0.4)))+
    theme(legend.position = "none")+
    theme(axis.title = element_text())->p
  return(p)
}

```

## 2.5 Generate an AUC-n plot

```
get_auc_n_plot = function() {  
  get_focal_imprvs = function(algorithm){  
    new_d = get_imprvement(d,algorithm) %>% bind_cols(d %>% select(n,fold))  
    new_d %>% rename(improvement = algorithm) %>% mutate(model = algorithm) %>%  
    group_by(model,n) %>% summarise(avg=mean(improvement),  
                                     se = serror(improvement))-> new_d  
  
    return(new_d)  
  }  
  d = read.csv("../data/evaluation_results/auc_single_assessment.csv")  
  baselines = unique(d$algorithm)  
  baselines = baselines[baselines != 'hmm']  
  d %>% pivot_wider(names_from = algorithm, values_from=score)-> d  
  baselines %>% map_dfr(get_focal_imprvs) -> r  
  get_labels(r,level=0.9) %>% update_model_names ->t  
  r %>% update_model_names %>% ggplot(aes(x = n, y = avg,  
                                           ymin = (avg -1.645 * se), ymax = (avg + 1.645 * se),  
                                           color = baseColor, shape = '1')) +  
  geom_smooth(aes(fill = bgColor), method = 'lm', alpha = 0.4,size=line_size)+  
  scale_color_manual(values = c(baseColor), name = "Improvement over") +  
  scale_fill_manual(values = c(bgColor), name = "Improvement over") +  
  scale_shape_manual(values = c(baseShape), name = "Improvement over") +  
  xlab("") +  
  geom_text(t, mapping = aes(x = 10, y =yposition,  
                             group = model,  
                             label = paste("Slope 90% CI: [", round(from, 2), ", ", "  
                                           round(to, 2), "]", sep = "")),  
                             size = annot_size, color = baseColor),  
            check_overlap = T,inherit.aes = FALSE, size=annot_size) +  
  scale_x_continuous(breaks = c(1, 5,10, 15,20)) +  
  facet_wrap(~model, ncol = 5, scales = "free_y")+  
  geom_hline(yintercept = 0, linetype = "dashed") +  
  ylab("AUC Improvement (%)") +  
  theme_hc(base_size=baseSize)+  
  theme(strip.background =element_rect(fill=alpha(bgColor,0.4)))+  
  theme(legend.position = "none")+  
  theme(axis.title = element_text())->p  
  return(p)  
}
```

## 2.6 Generate a within-opening performance plot

```
get_within_opening_perf_plot <- function() {
  topN = 17 #median number of applications over 4 (55/4)
           ##-- top 25% of options within each opening.
  d <- read.csv(paste("../data/evaluation_results/within_openings.csv",
                      sep = "")) %>% filter(n==topN)
  get_focal_imprvs = function(algorithm){
    new_d = get_imprvement(d,algorithm) %>% bind_cols(d %>% select(fold))
    new_d %>% rename(improvement = algorithm) %>% mutate(model = algorithm) %>%
    group_by(model) %>% summarise(avg=mean(improvement),
                                   se = serror(improvement))-> new_d

    return(new_d)
  }
  baselines = unique(d$algorithm)
  baselines = baselines[baselines != 'hmm']
  d %>% pivot_wider(names_from = algorithm, values_from=score)-> d
  baselines %>% map_dfr(get_focal_imprvs) -> r
  r %>% update_model_names %>% ggplot(aes(x = model, y = avg,
                                           ymin = avg - 1.645 * se, ymax = avg + 1.645 * se,
                                           color = '1', shape = '1')) +

  geom_point(size = shape_size) +
  geom_errorbar(width = 0.2) +
  ylab("Improvement (%)") +
  xlab(expression(paste(""))) +
  theme_hc(base_size=baseSize)+
  theme(strip.background =element_rect(fill=alpha(bgColor,0.4)))+
  scale_color_manual(values = c(baseColor, baseColor, baseColor)) +
  scale_shape_manual(name = "", values = c(baseShape, baseShape, baseShape)) +
  theme(legend.position = "none") +
  geom_abline(slope = 0, intercept = 0, linetype = 'dashed')+
  theme(axis.title = element_text())+
  facet_wrap(~"Within-task improvement of the HMM framework over alternative recommenders",str
}
```

## 2.7 Generate a basic AUC plot

```
get_auc_plot<- function(inputfile) {
  d = read_csv(paste("../data/evaluation_results/",inputfile,sep="")) %>%
    filter(n==0)
  get_focal_imprvs = function(algorithm){
    new_d = get_imprvement(d,algorithm) %>% bind_cols(d %>% select(fold))
    new_d %>% rename(improvement = algorithm) %>% mutate(model = algorithm) %>%
```

```

    group_by(model) %>% summarise(avg=mean(improvement),
                                   se = serror(improvement))-> new_d

    return(new_d)
}

baselines = unique(d$algorithm)
baselines = baselines[baselines != 'hmm']
d %>% select(fold,algorithm,score) %>% pivot_wider(names_from = algorithm,
                                                    values_from=score)->d

baselines %>% map_dfr(get_focal_imprvs) -> r
r %>% update_model_names %>% ggplot(aes(x = model, y = avg,
                                         ymin = avg - 1.645 * se,
                                         ymax = avg + 1.645 * se,
                                         color = 'r', shape = 'r')) +
  geom_point(size = shape_size) +
  geom_errorbar(width = 0.2) +
  ylab("Improvement (%)") +
  xlab(expression(paste(""))) +
  geom_vline(xintercept = c(10.5), linetype = "dashed") +
  theme_hc(base_size=baseSize)+
  theme(strip.background =element_rect(fill=alpha(bgColor,0.4)))+
  scale_color_manual(values = c(baseColor, baseColor, baseColor)) +
  scale_shape_manual(name = "", values = c(baseShape, baseShape, baseShape)) +
  theme(legend.position = "none") +
  geom_abline(slope = 0, intercept = 0, linetype = 'dashed')+
  theme(axis.title = element_text()+
  facet_wrap(~"AUC improvement of the HMM framework over alternative recommenders",strip.posit
return(p)
}

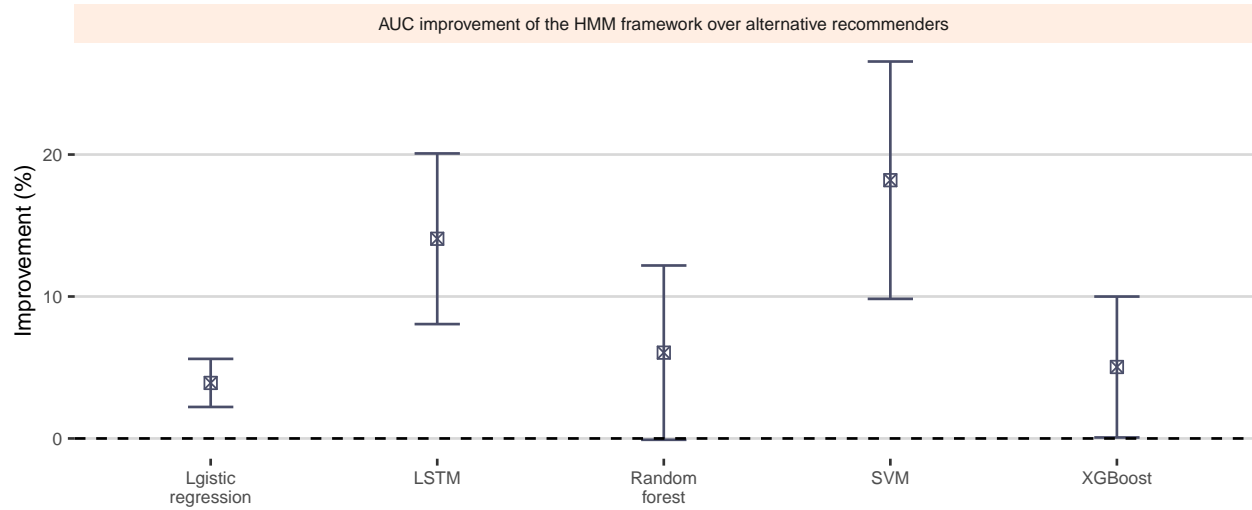
```



### 3 Creating the paper's figures

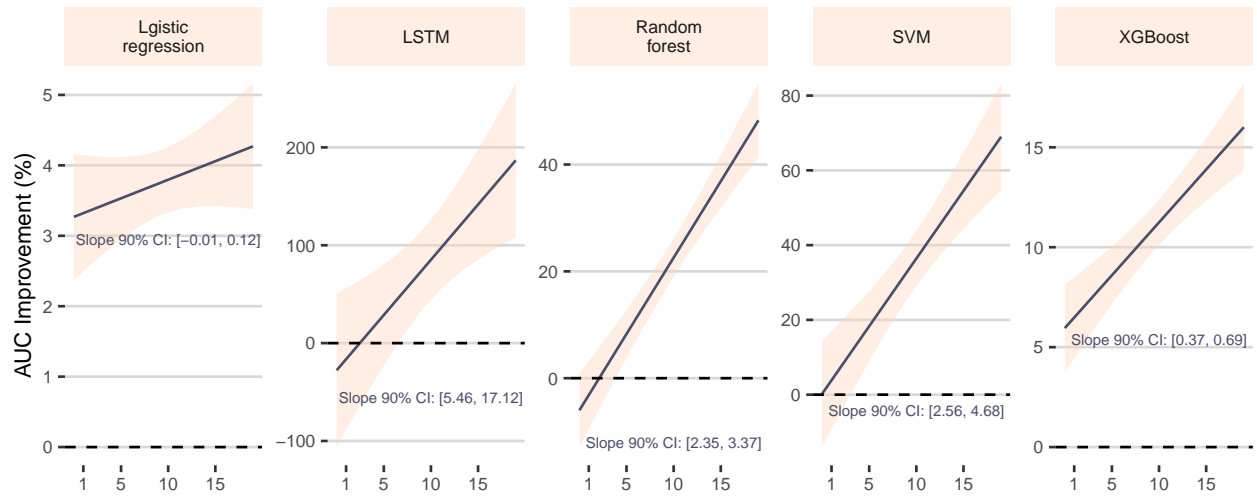
#### 3.1 Figure XX: AUC comparison

```
get_auc_plot('auc_single_assessment.csv') ->p
ggsave(file =
  paste("../..../Apps/Overleaf/gbu/ms/r2/figures/rest_auc.pdf", sep = ""),
  width = 14, height = 5, dpi = 300)
p
```



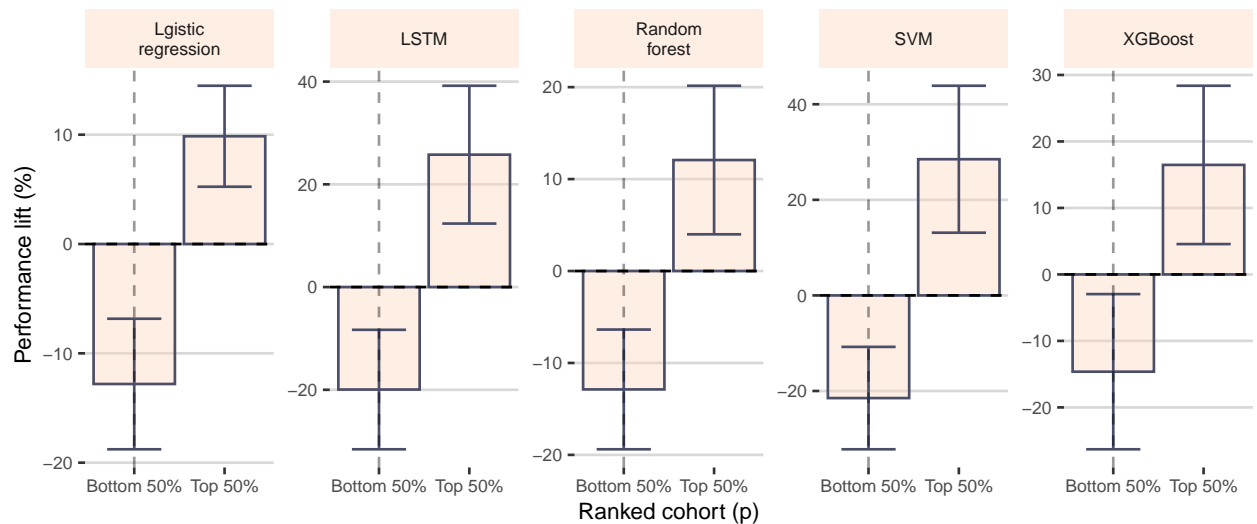
### 3.2 Figure XX: AUC-n comparison

```
p = get_auc_n_plot()
ggsave(file =
  paste("../..../Apps/Overleaf/gbu/ms/r2/figures/rest_aucn.pdf",
    sep = ""), width = 16, height = 5, dpi = 300)
p
```



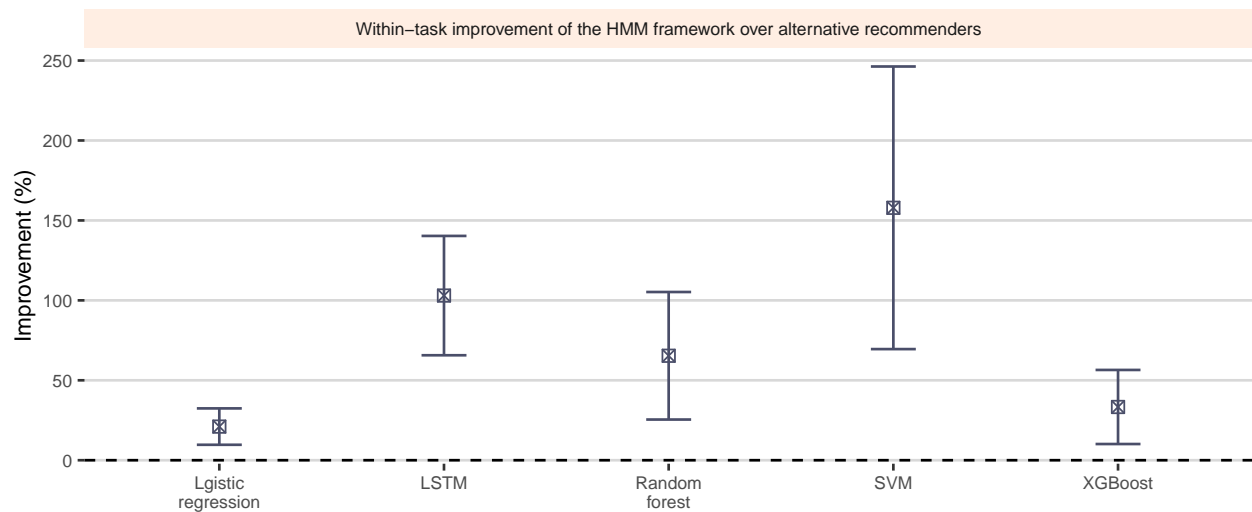
### 3.3 Figure XX: Ranking performance

```
p = get_ranking_performance_plot()
ggsave(file =
  paste("../..../Apps/Overleaf/gbu/ms/r2/figures/rest_ranking_performance.pdf",
    sep = ""), width = 16, height = 5, dpi = 300)
p
```



### 3.4 Figure XX: Performance within openings

```
p = get_within_opening_perf_plot()
ggsave(file =
paste("../..../Apps/Overleaf/gbu/ms/r2/figures/rest_withinOpenings.pdf",
      sep = ""),
      width = 14, height = 5, dpi = 300)
p
```



### 3.5 Figure XX: AUC comparison – many assessment

```
get_auc_plot('auc_many_assessment.csv')->p
ggsave(file =
paste("../..../Apps/Overleaf/gbu/ms/r2/figures/rest_many_assessment.pdf",
      sep = ""),
width = 14, height = 5, dpi = 300)
p
```

