R Scripts Used for statistical data Analysis

Loading neccessary packages and Libraries

*# library(extrafont) ## get more fonts*

*# font\_import() ## import more fonts*

*# loadfonts(device = "win") ## load more fonts*

install.packages("car") *# Install the car package*

library(car) *## Influence Index Plots, Summary for mixed models and more*

library(ggfortify) *## Data Visualization Tools for Statistical Analysis Results*

library(openxlsx) *## read xlsx files into R*

library(tidyverse)  *## collection of R packages designed for data science*

library(ggpubr) *## Combining multiple ggplot elements*

library(emmeans) *## Post-hoc testing*

library(knitr)

library(data.table)

library(reshape2)

library("readxl") *#install.packages("readxl")*

library(openxlsx) *#install.packages("openxlsx") # Install the openxlsx package*

library(ggfortify) *# install.packages("ggfortify")*

library(car)

library(ggfortify)

library(openxlsx)

library(tidyverse)

install.packages("emmeans")

# install.packages("tidyverse")

library(tidyverse)

# install.packages("knitr")

library(knitr)

1. Load data

setwd("C:/Users/XXXX/AI")

AI <- openxlsx::read.xlsx("Datasheet-stat20240923\_complete.xlsx", sheet = "MasterSheet")

# AI<- read.xlsx("Datasheet-stat20240923\_complete.xlsx", sheet="MasterSheet")

colnames (AI)

AI## Remove NA rows:

AI <- na.omit(AI)

## Relevel the Algorithm Variable

AI$Algorithm <- as.factor(AI$Algorithm)

AI$Algorithm <- factor(AI$Algorithm,

levels = c("Linear regression", "Elastic Net regressor", "Decision tree regressor", "Random forest","Gradient boost", "LightGBM", "XGBoost"))

*## Change data structure if needed*

AI$Sampling.Technique <- as.factor(AI$Sampling.Technique)

AI$Dataset <- as.factor(AI$Dataset)

AI$Sample.Size <- as.factor(AI$Sample.Size)

*## Functions:*

*# alternate vertical adjustment for x-axis labels*

alt\_label\_positions <- function(labels) { vjust\_values <- rep(c(0.9,1.1), length.out = length(labels))

return(vjust\_values)}

1. Scripts used for ANOVA, tukey test, pairwise comparisions and visualizations (Box plots etc)\*Note: here are the example scropts used for # Log Nautral (s\*CPU%) the simmiaör scropts were used for the other vaibales namely nRMSC and MAPE
2. ### Model Output for Sampling Techniques and Algorithms

```{r}

*# Calculate 'In' as the product of 'seconds.in.absolute.terms' and 'Average.CPU.usage.(%)'*

AI$In <- AI$`seconds.in.absolute.terms` \* AI$`Average.CPU.usage.(%)`

model <- lm(log(In) ~ Sampling.Technique \* Algorithm + Dataset, data = AI)

autoplot(model, which = c(1:4))

anova(model)

anova(model) %>% kable(caption = "ln(s\*CPU%)")

summary(model)$r.squared

```

1. ### Pairwise Comparisons

```{r}

emm.all.in<-emmeans(model, pairwise ~ Algorithm | Sampling.Technique)

pairs(emm.all.in)

pairs(emm.all.in) %>% kable()

```

1. ### Boxplot

```{r}

*## Create p-values for pairwise comparison*

*#### Extracting the p values from the e means #######*

*## Extract pairwise comparisons from*

pairwise\_results <- summary(emm.all.in$contrasts)

*## Split the contrast column into A1 and A2*

p.box <- pairwise\_results %>%

separate(contrast, into = c("A1", "A2"), sep = " - ") %>%

select(Sampling.Technique, A1, A2, p.value) %>%

rename(SampTV = Sampling.Technique, label = p.value)

*## only keep significant values*

p.box$label<-as.numeric(p.box$label)

p.box<-filter(p.box, label<=0.05)

*## create a datatable with each comparison and its p value*

data.table::setnames(p.box, c("Sampling.Technique", "A1", "A2", "label"))

p.box$label <- formatC(

signif(p.box$label, digits = 3), digits = 2, format = "g", flag = "#")

p.box <- as.data.table(p.box)

*## set specifications fo where it will appear in the plot*

p.box[, y := (0:(.N-1)) \* (2/.N)+19, by=Sampling.Technique]

*## add an incremental spacing based on the row index*

p.box[, y := y + (0:(.N-1)) \* 2, by=Sampling.Technique] # Increase y position incrementally for each row

```

```{r, fig.width=10, fig.height=8}

*## Named vector for the facet labels*

facet\_labels <- c("AbsoluteNumberSampling" = "Absolute Number Sampling", "PercentageSampling" = "Percentage Sampling")

*## new variable to show significance without numbers*

p.box<-mutate(p.box, significance = ifelse(label<=0.001, "\*\*\*", ifelse(label<=0.01,"\*\*", "\*")))

data.table::setDF(p.box)

ggplot(AI, aes(x = Algorithm, y = log(In))) +

stat\_boxplot(geom = 'errorbar', width = 0.2) + ## Add errorbar to get T whiskers

geom\_boxplot(size = 0.2, outlier.shape = 1, outlier.stroke = 0.8) + ## Boxplot with customized outleirs

geom\_hline(yintercept = 0, size = 0.5, color = "black", linetype=2) + ## highlighting the 0 intercept

stat\_summary(fun = mean, geom = "point", shape = 4, size = 2, stroke=0.8, color = "black") + ## add mean values to the plot for each group

facet\_grid(~Sampling.Technique, labeller = labeller(Sampling.Technique = facet\_labels))+

ggsignif::geom\_signif(data=p.box,

aes(xmin=A1, xmax=A2, annotations=significance, y\_position=y),

textsize = 3.5, vjust = -0.001,

tip\_length = 0.01,

manual=TRUE, size=0.6) + ## add result from pariwise comaprison

ylim(-10, 45) + ## y axis limits

scale\_y\_continuous(breaks = seq(-5, 15, by = 5), limits = c(-10, 45), expand = c(0, 0)) +

labs(y = "ln(s\*CPU%)") + ## plot title

theme\_minimal()+ ## general base theme for the plot

theme(#text = element\_text(family = "Gill Sans MT"), ## Font

panel.grid.major.x = element\_blank(), ## remove grid lines

panel.grid.minor.x = element\_blank(), panel.grid.minor.y = element\_blank(),

axis.text.x = element\_text(angle = 45, vjust = 1, hjust=1.1, color = "black", size=10),

axis.text.y = element\_text( size=10), plot.title = element\_text(hjust = 1, size=8), ## Center plot title

axis.title.x = element\_blank(), axis.title.y = element\_text(hjust = 0.25), ## Adjust y-axis title position

plot.margin = unit(c(1, 1, 1, 2), "cm"))

```

*### Pairwise Comparison Visualization*

```{r, fig.width=10, fig.height=8}

*## extract pairwise comparison*

*## Extract the contrasts result*

tukey\_result <- as.data.frame(summary(emm.all.in)$contrast)

tukey\_df <- summary(tukey\_result, by = "Sampling.Technique")[c(1:nrow(tukey\_result)),]

*## Calculate the lower and upper confidence limits*

tukey\_df$lower.CL <- tukey\_df$estimate - 1.96 \* tukey\_df$SE

tukey\_df$upper.CL <- tukey\_df$estimate + 1.96 \* tukey\_df$SE

*## Check the data frame structure*

print(tukey\_df)

*## Reorder the columns to match the desired structure*

tukey\_df <- tukey\_df[, c("contrast", "Sampling.Technique", "estimate", "SE", "df", "lower.CL", "upper.CL", "t.ratio", "p.value")]

*## Plot the results*

ggplot(tukey\_df, aes(x = contrast, y = estimate, color = p.value < 0.05)) +

geom\_point(size = 4) + geom\_errorbar(aes(ymin = lower.CL, ymax = upper.CL), width = 0.2) +

geom\_hline(yintercept = 0, linetype = "dashed", color = "black") +

facet\_grid(~Sampling.Technique, labeller = labeller(Sampling.Technique = facet\_labels))+

labs( title = "Post-Hoc Pairwise Comparisons", x = "Algorithm Comparisons", y = "Mean Difference (log-transformed)" ) + coord\_flip() + # Flip the axes for better readability

scale\_color\_manual(values = c("TRUE" = "chartreuse3", "FALSE" = "black")) + theme\_minimal()

```

1. ## Absolute Sampling

*### Model Output*

*#### Interaction*

```{r}

AI2 <- AI %>% dplyr::filter(Sampling.Technique == "AbsoluteNumberSampling") %>% droplevels()

model <- lm(log(In) ~ Algorithm \* Sample.Size + Dataset, data = AI2)

autoplot(model, which = c(1:4))

anova(model)

anova(model) %>% kable(caption = "Absolute Sampling: ln(s\*CPU%)")

summary(model)$r.squared

```

*### Pairwise Comaprison*

```{r}

emm.all.in<-emmeans(model, pairwise ~ Algorithm | Sample.Size)

pairs(emm.all.in)

pairs(emm.all.in) %>% kable()

```

*### Boxplot*

*## Create p-values for pairwise comparison*

*#### Extracting the p values*

*## Extract pairwise comparisons*

```{r}

pairwise\_results <- summary(emm.all.in$contrasts)

## Split the contrast column into A1 and A2

p.box <- pairwise\_results %>%

separate(contrast, into = c("A1", "A2"), sep = " - ") %>%

select(Sample.Size, A1, A2, p.value) %>%

rename(SampTV = Sample.Size, label = p.value)

## View the resulting data frame

## only keep significant values

p.box$label<-as.numeric(p.box$label)

p.box<-filter(p.box, label<=0.05)

### create a datatable with each comparison and its p value

data.table::setnames(p.box, c("Sample.Size", "A1", "A2", "label"))

p.box$label <- formatC( signif(p.box$label, digits = 3), digits = 2, format = "g", flag = "#")

p.box <- as.data.table(p.box)

## set specifications fo where it will appear in the plot

p.box[, y := (0:(.N-1)) \* (2/.N)+19, by=Sample.Size]

## add an incremental spacing based on the row index

p.box[, y := y + (0:(.N-1)) \* 2, by=Sample.Size] # Increase y position incrementally for each row

```

```{r, fig.width=10, fig.height=8}

*## Named vector for the facet labels*

facet\_labels <- c("100" = "100", "1000"= "1'000", "10000"= "10'000", "1e+05"= "100'000")

*## new variable to show significance without numbers*

p.box<-mutate(p.box, significance = ifelse(label<=0.001, "\*\*\*", ifelse(label<=0.01,"\*\*", "\*")))

data.table::setDF(p.box)

ggplot(AI2, aes(x = Algorithm, y = log(In))) +

stat\_boxplot(geom = 'errorbar', width = 0.2) + ## Add errorbar to get T whiskers

geom\_boxplot(size = 0.2, outlier.shape = 1, outlier.stroke = 0.8) + ## Boxplot with customized outleirs

geom\_hline(yintercept = 0, size = 0.5, color = "black", linetype=2) + ## highlighting the 0 intercept

stat\_summary(fun = mean, geom = "point", shape = 4, size = 2, stroke=0.8, color = "black") + ## add mean values to the plot for each group

facet\_grid(~Sample.Size, labeller = labeller(Sample.Size = facet\_labels))+

ggsignif::geom\_signif(data=p.box,

aes(xmin=A1, xmax=A2, annotations=significance, y\_position=y),

textsize = 3.5, vjust = -0.001, tip\_length = 0.01, manual=TRUE, size=0.6) + ## add result from pariwise comaprison

ylim(-10, 40) + ## y axis limits

scale\_y\_continuous(breaks = seq(-5, 15, by = 5), limits = c(-10, 40), expand = c(0, 0)) +

labs(y="ln(s\*CPU%)" ) + ## plot title

theme\_minimal()+ ## general base theme for the plot

theme(#text = element\_text(family = "Gill Sans MT"), ## Font

panel.grid.major.x = element\_blank(), ## remove grid lines

panel.grid.minor.x = element\_blank(),

panel.grid.minor.y = element\_blank(),

axis.text.x = element\_text(

angle = 45, vjust = 1, hjust=1.1, color = "black", size=10), axis.text.y = element\_text( size=10),

plot.title = element\_text(hjust = 1, size=8), ## Center plot title

axis.title.x = element\_blank(),

axis.title.y = element\_text(hjust = 0.25), ## Adjust y-axis title position

plot.margin = unit(c(1, 1, 1, 2), "cm"))

```

*### Pairwise Comparison Visualization*

```{r, fig.width=10, fig.height=8}

*## extract pairwise comparison*

*## Extract the contrasts results*

tukey\_result <- as.data.frame(summary(emm.all.in)$contrast)

tukey\_df <- summary(tukey\_result, by = "Sample.Size")[c(1:nrow(tukey\_result)),]

## Calculate the lower and upper confidence limits

tukey\_df$lower.CL <- tukey\_df$estimate - 1.96 \* tukey\_df$SE

tukey\_df$upper.CL <- tukey\_df$estimate + 1.96 \* tukey\_df$SE

*## Check the data frame structure*

print(tukey\_df)

*## Reorder the columns to match the desired structure*

tukey\_df <- tukey\_df[, c("contrast", "Sample.Size", "estimate", "SE", "df", "lower.CL", "upper.CL", "t.ratio", "p.value")]

## Plot the results

ggplot(tukey\_df, aes(x = contrast, y = estimate, color = p.value < 0.05)) +

geom\_point(size = 4) +

geom\_errorbar(aes(ymin = lower.CL, ymax = upper.CL), width = 0.2) +

geom\_hline(yintercept = 0, linetype = "dashed", color = "black") +

facet\_grid(~Sample.Size, labeller = labeller(Sample.Size = facet\_labels))+

labs( title = "Post-Hoc Pairwise Comparisons",

x = "Algorithm Comparisons", y = "Mean Difference (log-transformed)" ) +

coord\_flip() + # Flip the axes for better readability

scale\_color\_manual(values = c("TRUE" = "chartreuse3", "FALSE" = "black")) +

theme\_minimal()

```

*For completion, below is the model without the interaction term:*

*#### No Interaction*

```{r}

model <- lm(log(In) ~ Algorithm + Sample.Size + Dataset, data = AI2)

autoplot(model, which = c(1:4))

anova(model)

anova(model) %>% kable(caption = "Absolute Sampling without interaction: ln(s\*CPU%)")

summary(model)$r.squared

```

```{r}

*## Pairwise comparison*

emm.all.in<-emmeans(model, pairwise ~ Algorithm )

emm.all.in[1] %>% kable(caption = "Algorithm")

emm.all.in[2] %>% kable(caption = "Algorithm comparison")

emm.all.in<-emmeans(model, pairwise ~ Sample.Size)

emm.all.in[1] %>% kable(caption = "Sample Size")

emm.all.in[2] %>% kable(caption = "Sample Size comparison")

emm.all.in<-emmeans(model, pairwise ~ Dataset)

emm.all.in[1] %>% kable(caption = "Dataset")

emm.all.in[2] %>% kable(caption = "Dataset comparison")

```

1. ## Percentage Sampling

*### Model Output*

```{r}

AI3 <- AI %>% dplyr::filter(Sampling.Technique == "PercentageSampling") %>% droplevels()

model <- lm(log(In) ~ Algorithm \* Sample.Size + Dataset, data = AI3)

autoplot(model, which = c(1:4))

anova(model)

anova(model) %>% kable(caption = "Percentage Sampling: ln(s\*CPU%)")

summary(model)$r.squared

```

*### Pairwise comparison*

```{r}

emm.all.in<-emmeans(model, pairwise ~ Algorithm | Sample.Size)

pairs(emm.all.in)

pairs(emm.all.in) %>% kable()

```

*### Boxplot*

```{r}

*## Create p-values for pairwise comparison*

*#### Extracting the p values*

*## Extract pairwise comparisons*

pairwise\_results <- summary(emm.all.in$contrasts)

*## Split the contrast column into A1 and A2*

p.box <- pairwise\_results %>%

separate(contrast, into = c("A1", "A2"), sep = " - ") %>%

select(Sample.Size, A1, A2, p.value) %>%

rename(SampTV = Sample.Size, label = p.value)

*## only keep significant values*

p.box$label<-as.numeric(p.box$label)

p.box<-filter(p.box, label<=0.05)

*## create a datatable with each comparison and its p value*

data.table::setnames(p.box, c("Sample.Size", "A1", "A2", "label"))

p.box$label <- formatC( signif(p.box$label, digits = 3), digits = 2, format = "g", flag = "#")

p.box <- as.data.table(p.box)

*## set specifications of where it will appear in the plot*

p.box[, y := (0:(.N-1)) \* (2/.N)+19, by=Sample.Size]

*## add an incremental spacing based on the row index*

p.box[, y := y + (0:(.N-1)) \* 2, by=Sample.Size] # Increase y position incrementally for each row

```

```{r, fig.width=10, fig.height=8}

*## Named vector for the facet labels*

facet\_labels <- c("0.125" = "12.5%", "0.25" = "25%", "0.5"= "50%", "1"= "100%")

*## new variable to show significance without numbers*

p.box<-mutate(p.box, significance = ifelse(label<=0.001, "\*\*\*", ifelse(label<=0.01,"\*\*", "\*")))

data.table::setDF(p.box)

ggplot(AI3, aes(x = Algorithm, y = log(In))) +

stat\_boxplot(geom = 'errorbar', width = 0.2) + ## Add errorbar to get T whiskers

geom\_boxplot(size = 0.2, outlier.shape = 1, outlier.stroke = 0.8) + ## Boxplot with customized outleirs

geom\_hline(yintercept = 0, size = 0.5, color = "black", linetype=2) + ## highlighting the 0 intercept

stat\_summary(fun = mean, geom = "point", shape = 4, size = 2, stroke=0.8, color = "black") + ## add mean values to the plot for each group

facet\_grid(~Sample.Size, labeller = labeller(Sample.Size = facet\_labels))+

ggsignif::geom\_signif(data=p.box, aes(xmin=A1, xmax=A2, annotations=significance, y\_position=y),

textsize = 3.5, vjust = -0.001, tip\_length = 0.01, manual=TRUE, size=0.6) +

## add result from pariwise comaprison

ylim(-10, 45) + ## y axis limits

scale\_y\_continuous(breaks = seq(-5, 15, by = 5), limits = c(-10, 35), expand = c(0, 0)) +

labs(y="ln(s\*CPU%)") + ## plot title

theme\_minimal()+ ## general base theme for the plot

theme(#text = element\_text(family = "Gill Sans MT"), ## Font

panel.grid.major.x = element\_blank(), ## remove grid lines

panel.grid.minor.x = element\_blank(),

panel.grid.minor.y = element\_blank(),

axis.text.x = element\_text(angle = 45, vjust = 1, hjust=1.1, color = "black", size=10),axis.text.y = element\_text( size=10), plot.title = element\_text(hjust = 1, size=8),

*## Center plot title* axis.title.x = element\_blank(),

axis.title.y = element\_text(hjust = 0.25), ## Adjust y-axis title position

plot.margin = unit(c(1, 1, 1, 2), "cm"))

```

### Pairwise Comparison Visualization

```{r, fig.width=10, fig.height=8}

## extract pairwise comparison

## Extract the contrasts result

tukey\_result <- as.data.frame(summary(emm.all.in)$contrast)

tukey\_df <- summary(tukey\_result, by = "Sample.Size")[c(1:nrow(tukey\_result)),]

## Calculate the lower and upper confidence limits

tukey\_df$lower.CL <- tukey\_df$estimate - 1.96 \* tukey\_df$SE

tukey\_df$upper.CL <- tukey\_df$estimate + 1.96 \* tukey\_df$SE

## Check the data frame structure

print(tukey\_df)

## Reorder the columns to match the desired structure

tukey\_df <- tukey\_df[, c("contrast", "Sample.Size", "estimate", "SE", "df", "lower.CL", "upper.CL", "t.ratio", "p.value")]

## Plot the results

ggplot(tukey\_df, aes(x = contrast, y = estimate, color = p.value < 0.05)) +

geom\_point(size = 4) +

geom\_errorbar(aes(ymin = lower.CL, ymax = upper.CL), width = 0.2) +

geom\_hline(yintercept = 0, linetype = "dashed", color = "black") +

facet\_grid(~Sample.Size, labeller = labeller(Sample.Size = facet\_labels))+

labs( title = "Post-Hoc Pairwise Comparisons", x = "Algorithm Comparisons", y = "Mean Difference (log-transformed)" ) + coord\_flip() + # Flip the axes for better readability

scale\_color\_manual(values = c("TRUE" = "chartreuse3", "FALSE" = "black")) +

theme\_minimal()

```