Created by Jeffrey Chan

This list is specifically for Machine Learning related information.

Library list

tydyverse

caret

MASS

Splinesfor splines function bs()

mgcv for GAM function

broom for augment()

modelr for rsquare() rmse() mae()

glance() computes r2, adjusted r2, sigma / RSE, AIC, BIC ***important***

To read the following list, when there is an -> like this, which means im commenting what is it all about. For full detail information about that specific functions, please use ?function() on your Rstudio.

Function list

sample_n(data, sample size)

-> it will ramdonly pick # of sample size you intput

set.seed()

-> create random number without repeating

training.samples <- data$col %>%

createDataPartition(p = 0.8, list = F)

train.data <- data[training.samples,]

-> using 80% of the data

test.data <- data[-training.samples,]

-> using 20% of the data

```
model <- lm(col ~., data = train.data)
```

-> lm means linear regression model

```
summary(model)
```

-> summarize the data with all statistical data

```
summary(model)$coef
```

-> list all the parameters coefficients

```
predictions <- model %>%
    predict(test.data)
```

-> feed the test data to the model

```
RMSE(predictions, test.data$col)
```

-> small # = good performance

```
R2(predictions, test.data$col)
```

-> close to ± 1 = good performance

```
(RMSE(predictions, test.data$col) / mean(test.data$col))*100
```

-> prediction error small # = good performance

```
ggplot(data, aes(x,y)) +
    geom_point() +
    stat_smooth()
```

-> geom_point() = scatter, stat_smooth() regression line

```
Model <- lm(data ~ x*y, data = train.data)
```

->interaction effects x * y with linear regression model

Contrasts(data$col)

> show you what 0, 1 is representing what category values for category regression

Variable <- data %>%

mutate(col = relevel(col, ref = "level_name")

->reconfig the level, start with 0. Contarsts is to read the level in #

Levels(data$col)

->check the factor levels

Variable <- model.matrix(~data, data = col)

->create a matrix with dummy variables

Anova(model)

-> Anova will take care of the unbalanced designs

Str(data)

->read the data info

Model <- lm(col ~ col + I(col^2), data = training.data)

->this is how you create polynomial regression

Model <- lm(y ~ poly(col , degree, raw = T), data = training.data)

->this is how you create polynomial regression

ggplot(training.data, aes(x,y)) +

geom_point() +

stat_smooth(moethod = lm, formula = y ~ plot(x, degree, raw = T)

->plot the polynomial regression model

This list is specifically for Machine Learning related information.

```
anova(model,model1,model2, … , modeln)
                    ->analysis of variance

Model <- lm(y ~ log(x), data = training.data)
                    ->log transformation linear regression

ggplot(training.data, aes(x,y)) +
    geom_point() +
    stat_smooth(method = lm, formula = y ~ log(x))
                    ->plot the log regression model

Knots <- quantile(training.data$col, p = c(.25, .5, .75))
                    ->create the knots for the splines model

Model <-lm(y ~ bs(col, knots = knots), data =training.data)
                    ->splines model

ggplot( training.data, aes(x,y)) +
    geom_point() +
    stat_smooth(method = lm, formula = y ~ splines::bs(x,df=#)
                    ->splines plot


Model <- gam(y ~s(x), data =training.data)
                    ->create Generalized additive model

ggplot(training.data, aes(x,y)) +
    geom_point() +
    stat)smooth(method = gam, formula = y~ s(x))
                    ->plot the gam regression function
```

```
ggplot(data, aes(x,y)) +
    geom_point() +
    stat_smooth(method = lm) +
    geom_line(aes(y = lwr), color = "red", linetype = "dashed") +
    geom_line(aes(y = upr), color = "red", linetype = "dashed")
```

->geom_line are the prediction interval.

```
Varaible <- augment(model)
```

->add performance data to the variable

```
ggplot(model, aes(x,y)) +
    geom_point() +
    stat_smooth(method = lm, se = F) +
    geom_segment(aes(xend = x, yend = col of performance data), color =
"red",size=0.3))
```

->This will visualize the residuals from the model

```
Plot(model)
```

->gives you diagnostic plot

Plot1 = fitted values (x) , residuals (y)

Plot 3= fitted values (x), root standardized residuals (y)

Plot 2= Theoretical Quantiles (x), standardized residuals(y)

Plot 5= leverage (x), standardized residuals(y) cooks dist

Plot 4 = obs # (x), cook's distance (y) cooks dist

```
AIC(model)
```

->small # = good performance

This list is specifically for Machine Learning related information.

BIC(model)

->small # = good performance

MAE(model)

->small # = good performance

glance(model)

->R2, adj.R2, sigma(RSE), AIC, BIC, P.value

Pretty much we want all small numbers except the R^2

train.control <-trainControl(method = "LOOCV")

->apply Levae one out cross validation

Loocv.model <- train(Col ~., data = data, dethod = "lm", trControl = train.control)

->train with LOOCV method

Train.control.k <- trainControl(method = "cv", number = #)

->apply K fold cross validation, # = k times usually 5/10

k.model <- train(col ~., data = data, method = "lm", trControl = train.control.k)

-> train with k fold.

Train.control.k.repeat <- trainControl(method = "repeatedcv", number = #, repeat = #)

->repeated with k fold at #

Train.control.bs <- trainControl(method = "boot", number = #)

->bootstrap traincontrol setup, number means sample size / repeatation

Model.bs <- train(col ~., data = data, methold = "lm", trControl = train.control.bs)

->train with bs train contorl