**Missing values in R**

houseprice=read.csv(file.choose(), stringsAsFactors = F) # importing train data

head(houseprice)

str(houseprice)

sort(colSums(is.na(houseprice)),decreasing = T) #check for null values

housepricecat=houseprice[sapply(houseprice, is.character)] # creating subset of categorical attributes

str(housepricecat)

housepricenum=houseprice[sapply(houseprice, is.numeric)] # creating subset of numerical attributes

str(housepricenum)

values=c('MSSubClass','OverallQual','OverallCond','YearBuilt','YearRemodAdd','GarageYrBlt','YrSold','MoSold')

housepricevalues=housepricenum[values] # creating subset of values attributes

housepricenum=housepricenum[,!colnames(housepricenum) %in% values] #removing the attributes of values

from subset of numerical attributes

sort(colSums(is.na(housepricecat)),decreasing = T) #checking for null values in cat.subset

nonecols=c('PoolQC','MiscFeature','Alley','Fence','FireplaceQu') #creating vector nonecols for null values imputation

for (col in nonecols){ #for loop for impute the missing values for nonecols vector

housepricecat[sapply(housepricecat[col], is.na),col]='missing'

}

sort(colSums(is.na(housepricecat)),decreasing = T)

housepricecat=sapply(housepricecat, function(x) ifelse(is.na(x),names(which.max(table(x))),x))

#imputing null val. In cat.subset with idxmax

sort(colSums(is.na(housepricecat)),decreasing = T)

sort(colSums(is.na(housepricevalues)),decreasing = T)

housepricevalues=sapply(housepricevalues, function(x) ifelse(is.na(x),names(which.max(table(x))),x))

# imputing null val. In housepriceval.subset with idxmax

sort(colSums(is.na(housepricevalues)),decreasing = T)

sort(colSums(is.na(housepricenum)),decreasing = T)

housepricenum=sapply(housepricenum, function(x) ifelse(is.na(x),mean(x,na.rm = TRUE),x))

# imputing null val. In numerical subset with mean

sort(colSums(is.na(housepricenum)),decreasing = T)

houseprice1=data.frame(housepricenum,housepricevalues,housepricecat) #combining the complete train data

sort(colSums(is.na(houseprice1)),decreasing = T)

str(houseprice1)

housereg=lm(SalePrice~.,data = houseprice1[-1]) #applying lin reg on combined train data

summary(housereg)

housestep=step(housereg) #applying step reg

summary(housestep)

#Importing test data

housetest=read.csv(file.choose(), stringsAsFactors = F) #importing test data

sort(colSums(is.na(housetest)),decreasing = T)

housetestcat=housetest[sapply(housetest, is.character)] #creating cat. subset of test data

housetestnum=housetest[sapply(housetest, is.numeric)] #creating num. subset of test data

housetestvalues=housetestnum[values] #creating value. subset of test data

housetestnum=housetestnum[,!colnames(housetestnum) %in% values]

#removing attributes from num. subset which are included in value subset

sort(colSums(is.na(housetestcat)),decreasing = T)

for (col in nonecols){ #for loop for impute the missing values for nonecols vector

housetestcat[sapply(housetestcat[col], is.na),col]='missing'

}

sort(colSums(is.na(housetestcat)),decreasing = T)

housetestcat=sapply(housetestcat, function(x) ifelse(is.na(x),names(which.max(table(x))),x))

#imputing null val. In cat.subset with idxmax

sort(colSums(is.na(housetestcat)),decreasing = T)

sort(colSums(is.na(housetestvalues)),decreasing = T)

housetestvalues=sapply(housetestvalues, function(x) ifelse(is.na(x),names(which.max(table(x))),x))

#imputing null val. In value.subset with idxmax

sort(colSums(is.na(housetestvalues)),decreasing = T)

sort(colSums(is.na(housetestnum)),decreasing = T)

housetestnum=sapply(housetestnum, function(x) ifelse(is.na(x),mean(x,na.rm = TRUE),x))

#imputing null val. In num.subset with mean

sort(colSums(is.na(housetestnum)),decreasing = T)

housetest1=data.frame(housetestnum,housetestvalues,housetestcat) #combining the complete test data

sort(colSums(is.na(housetest1)),decreasing = T)

housetrainrpart=rpart(SalePrice~.,data = houseprice1[-1]) #applying dT on train data

rpart.plot(housetrainrpart,cex=0.5)

summary(housetrainrpart)

testrpartpred=predict(housetrainrpart,housetest1,type='vector') #predicting DT on test data

#matching levels in MSsubclass,Yearbuilt,Garage YrBlt attributes between test and train datasets

housetest1$MSSubClass=factor(housetest1$MSSubClass,levels = levels(houseprice1$MSSubClass))

housetest1$YearBuilt=factor(housetest1$YearBuilt,levels = levels(houseprice1$YearBuilt))

housetest1$GarageYrBlt=factor(housetest1$GarageYrBlt,levels = levels(houseprice1$GarageYrBlt))

write.csv(testrpartpred,"testrpartpred.csv")

houserandomforest=randomForest(SalePrice~.,data=houseprice1[-1],ntree=3000,importance=T,do.trace=200)

str(houseprice1)

#creating new variables of age, remod age and Garage Age both in test and train datasets

houseprice1$YearBuilt=as.Date(houseprice1$YearBuilt,"%Y")

houseprice1$YrSold=as.Date(houseprice1$YrSold,"%Y")

houseprice1$Age=(houseprice1$YrSold-houseprice1$YearBuilt)/365

houseprice1$YearRemodAdd=as.Date(houseprice1$YearRemodAdd,"%Y")

houseprice1$RemodAge=(houseprice1$YrSold-houseprice1$YearRemodAdd)/365

houseprice1$GarageYrBlt=as.Date(houseprice1$GarageYrBlt,"%Y")

houseprice1$GarAge=(houseprice1$YrSold-houseprice1$GarageYrBlt)/365

delcols=c("YearBuilt","YrSold","YearRemodAdd","GarageYrBlt")

houseprice2=houseprice1[!colnames(houseprice1) %in% delcols] #to drop columns by colnames

houserandomforest=randomForest(SalePrice~.,data=houseprice2[-1],ntree=3000,importance=T,do.trace=200)

print(houserandomforest)

sqrt(sum(houserandomforest$mse))

varImpPlot(houserandomforest)

housetest1$YearBuilt=as.Date(housetest1$YearBuilt,"%Y")

housetest1$YrSold=as.Date(housetest1$YrSold,"%Y")

housetest1$Age=(housetest1$YrSold-housetest1$YearBuilt)/365

housetest1$YearRemodAdd=as.Date(housetest1$YearRemodAdd,"%Y")

housetest1$RemodAge=(housetest1$YrSold-housetest1$YearRemodAdd)/365

housetest1$GarageYrBlt=as.Date(housetest1$GarageYrBlt,"%Y")

housetest1$GarAge=(housetest1$YrSold-housetest1$GarageYrBlt)/365

housetest2=housetest1[!colnames(housetest1) %in% delcols]

houseRFtestpred=predict(houserandomforest,housetest2[-1]

**Missing value handling with missForest package in R +Random Forest model:**

house=read.csv(file.choose()) #importing train data

testhouse=read.csv(file.choose()) #importing test data

totaldata=rbind(house[-81],testhouse) #row bind test and train data

for (f in 1:length(names(totaldata))){ #matching levels in binded data with the levels of train data

levels(house[,f])=levels(totaldata[,f])

}

sort(colSums(is.na(totaldata)),decreasing = T) #checking (na) missing values in binded data

totaldataimp=missForest(totaldata) #treating binded data with missforest imputation package

finaldata=totaldataimp$ximp #getting the imputed data matrix

sort(colSums(is.na(finaldata)),decreasing = T) #checking na in finaldata(imputed data matrix)

train=finaldata[1:1460,] #extracting train data from imputed data

test=finaldata [1461:2919,] #extracting test data from imputed data

train=cbind(train,SalePrice=house$SalePrice) #adding the y variable in train data

trainreg=lm(SalePrice~.,data=train[-1]) #applying lin reg on the train data

testpred=predict(trainreg,test[-1]) #predicting o/p of lin reg on test data

head(testpred) # checking head of predicted values

is.na(testpred) #checking na in predicted values

write.csv(testpred,"testpred.csv")

trainRF=randomForest(SalePrice~.,data=train[-1],ntree=3000,importance=T,do.trace=200)

#applying Random on train data

summary(trainRF) #o/p of Random forest model

testRFpred=predict(trainRF,test[-1]) #predicting test data with random forest model

write.csv(testRFpred,"testRFpred.csv")