ML_Group_Project

2024-11-20

Data Exploration

ADD SAMARA'S part

```
#cor(df_train[,-c(1,10)])
```

It is worth to note that "decane_toluene" is not in the test data set

Missing Data

Missing data was found in the 'parentspecies' attribute. According to the definition of the 'parentspecies' attribute, missing values imply a meaning.They are not missing randomly but due to difficulty in retrieving the 'parentspecies'. Therefore, a new level was created as 'Unknown'.

```
test_data$parentspecies[test_data$parentspecies == ""] <- "Unknown"
train_data$parentspecies[train_data$parentspecies == ""] <- "Unknown"
```

Dummy model

A dummy model is a supervised learning model that gives the same constant output regardless of the values of the covariates. We first built a dummy model and calculated the training error and the cross validation error.

```
set.seed(123)
dummy_model = glm(log_pSat_Pa ~ 1, data = train_data[,-1])
dummy_cv_error_5 = cv.glm(train_data[,-1] , dummy_model , K = 10)$delta[1]
dummy_error_train = mean((train_data$log_pSat_Pa - predict(dummy_model, train_data[,-1]))^2)
```

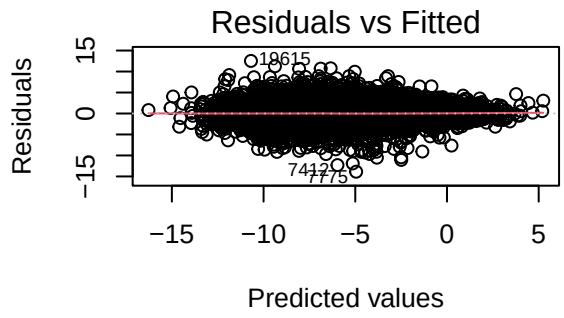
model	train	cv	kaggle_score
Dummy	9.735229	9.736679	-1e-04

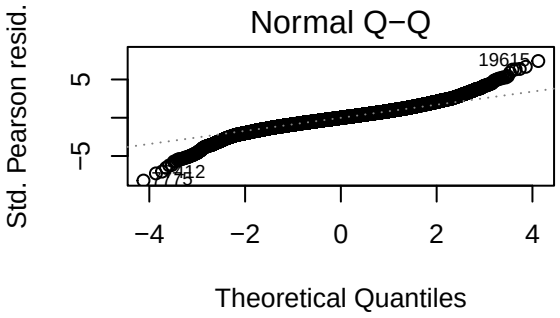
OLS as a baseline model: kaggle_score = 0.7163

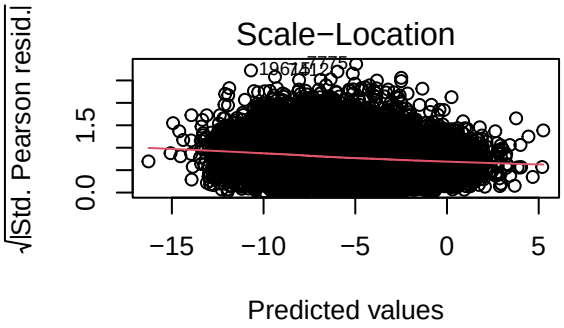
ten fold cross validation was done to get the cross validation error

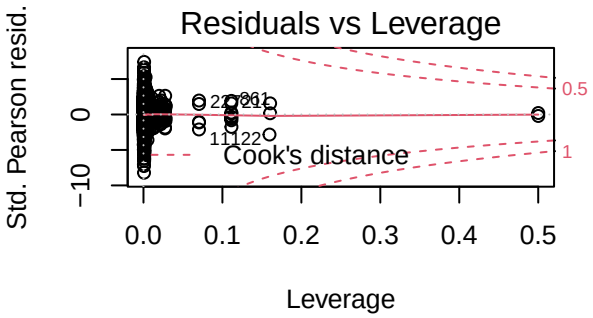
```r
set.seed(123)
ols_fit = glm(log_pSat_Pa ~ ., data = train_data[,-1])
cv_error_10 = cv.glm(train_data[,-1] , ols_fit , K = 10)$delta[1]
error_train = mean((train_data$log_pSat_Pa - predict(ols_fit, train_data[,-1]))^2)
```

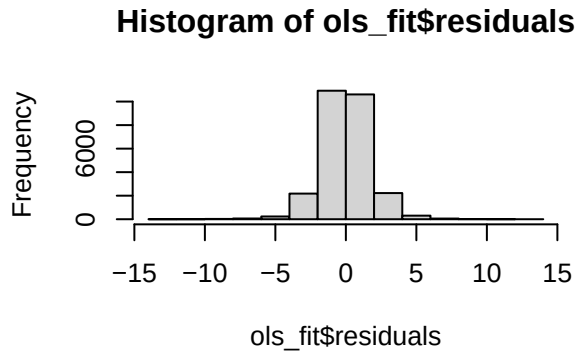
Plotting


Residuals vs Fitted
Residuals
Predicted values
19615
7412
7775


Normal Q−Q
Std. Pearson resid.
Theoretical Quantiles
19615
7412
7775


Scale−Location
√|Std. Pearson resid.|
Predicted values


Residuals vs Leverage
Std. Pearson resid.
Cook's distance
0.5
1
Leverage


Histogram of ols_fit$residuals
Frequency
ols_fit$residuals

model	train	cv	kaggle_score
Dummy	9.735229	9.736679	-0.0001
ols	2.855265	2.862134	0.7163

Then we analyze the correlation between each column and the target, selecting columns with a correlation above 0.5 or 0.3, and then using these for the model prediction. However, even after trying these two methods, the prediction score didn't change,

Lasso 0.7160

Before moving to non-linear models we tried regularization using Lasso

"decane_toluene" is not in the test data set. so it was handled in a way so that both train and test dataset has the same number of levels

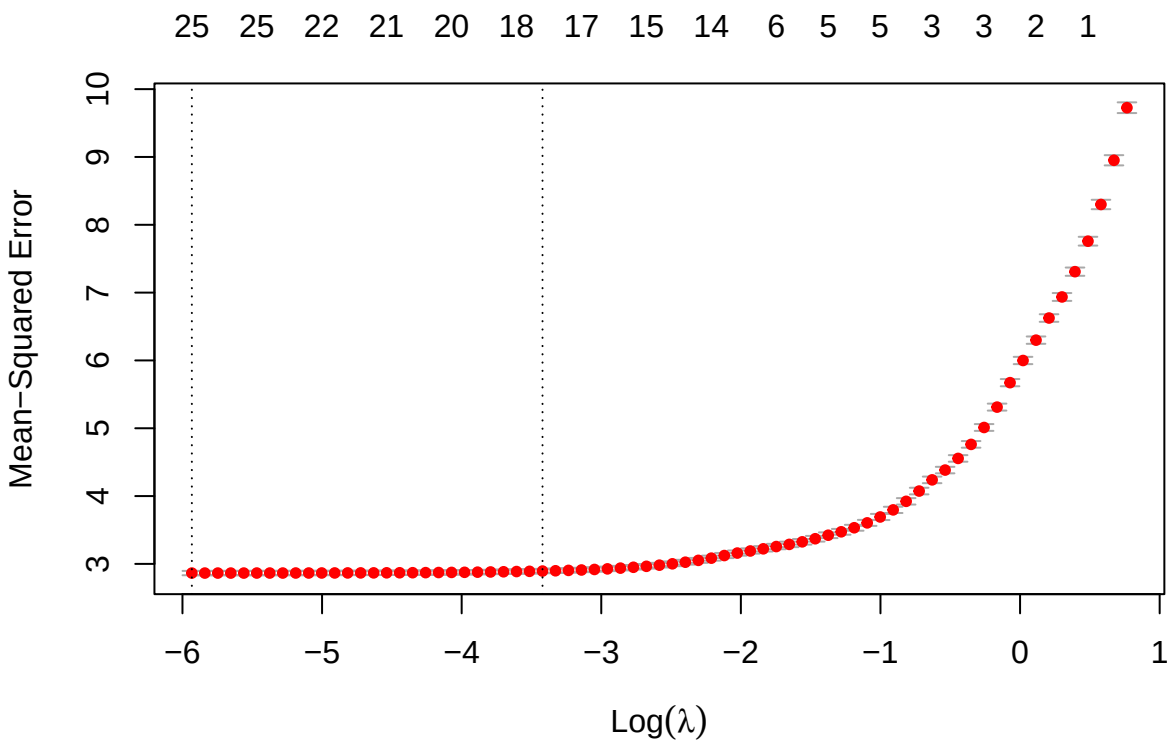
Level	Freq_train	Freq_test
apin	6165	1195
apin_decane	46	5
apin_decane_toluene	9	2
apin_toluene	37	5
decane	2218	381
decane_toluene	2	0
toluene	17950	3379
Unknown	210	33

Cross validation to obtain the best lambda(tuning parameter)

best lambda obtained was 0.002651092

Plot

```
plot(cvfit)
```



WE can use lasso as a variable selection method

```
## <sparse>[ <logic> ]: .M.sub.i.logical() maybe inefficient
```

	x
(Intercept)	6.2437263
MW	-0.0026108
NumOfAtoms	0.0000000
NumOfC	-0.7950384
NumOfO	-0.0008376
NumOfN	0.0000000
NumHBondDonors	-1.7561728
NumOfConf	-0.0022532
NumOfConfUsed	-0.0000226
parentspeciesapin	0.0000000
parentspeciesapin_decane	-0.4952347
parentspeciesapin_decane_toluene	0.0766192
parentspeciesapin_toluene	-0.2154127
parentspeciesdecane	-0.8308856
parentspeciesdecane_toluene	-0.4904175
parentspeciestoluene	0.0000000
parentspeciesUnknown	-0.8785030
C.C..non.aromatic.	-0.8968921
C.C.C.O.in.non.aromatic.ring	0.0000000

	x
hydroxyl..alkyl.	0.0000000
aldehyde	-0.2922533
ketone	0.0637678
carboxylic.acid	-1.0328902
ester	-0.2404483
ether..alicyclic.	-0.6441586
nitrate	0.0000000

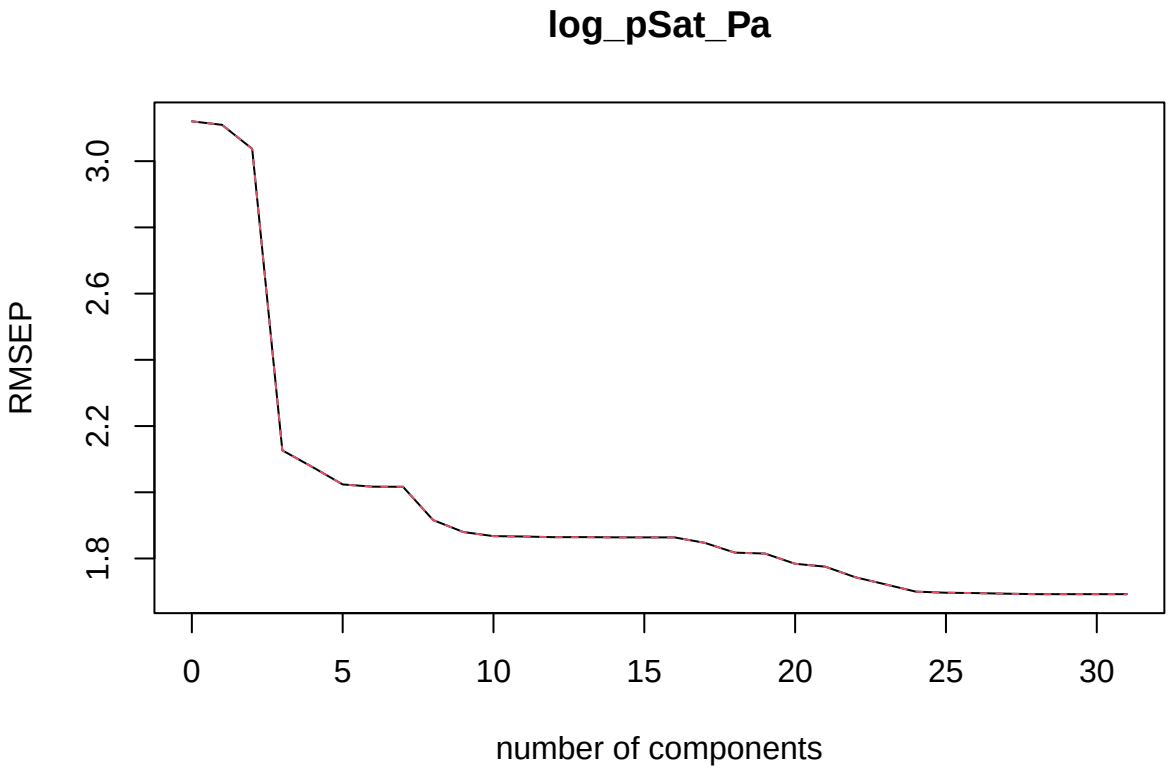
model	train	cv	kaggle_score
Dummy	9.735229	9.736679	-0.0001
ols	2.855265	2.862134	0.7163
Lasso	3.780056	2.857745	0.7160

PCR: Kaggel Score: 0.7163

```
set.seed(9)
pcr_fit = pcr(log_pSat_Pa ~ ., data = train_data[,-1], scale = TRUE, validation = "CV")
```

```
#summary(pcr_fit)

validationplot(pcr_fit)
```




```
min.pcr = which.min(MSEP(pcr_fit)$val[1,1, ] ) - 1
min.pcr
```

```
## 28 comps
##       28
```

```
summary(pcr_fit)
```

```
## Data:    X dimension: 26637 31
##  Y dimension: 26637 1
## Fit method: svdpc
## Number of components considered: 31
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            3.12     3.11    3.037    2.127    2.076    2.024    2.017
## adjCV         3.12     3.11    3.037    2.126    2.076    2.023    2.017
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       2.017    1.916     1.88     1.867     1.866     1.864     1.865
## adjCV    2.017    1.916     1.88     1.867     1.866     1.864     1.864
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV        1.864     1.864     1.864     1.847     1.818     1.815     1.784
## adjCV     1.864     1.864     1.864     1.847     1.818     1.815     1.784
```

```
##        21 comps  22 comps  23 comps  24 comps  25 comps  26 comps  27 comps
## CV        1.775     1.743     1.722       1.7     1.697     1.695     1.694
## adjCV     1.775     1.743     1.722       1.7     1.696     1.695     1.694
##        28 comps  29 comps  30 comps  31 comps
## CV        1.692     1.692     1.692     1.692
## adjCV     1.692     1.692     1.692     1.692
##
## TRAINING: % variance explained
##              1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X            14.5454   24.860    31.80    37.31    42.08    46.47    50.70
## log_pSat_Pa   0.6911    5.276    53.57    55.76    57.97    58.26    58.29
##              8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X              54.71    58.54     62.22     65.71     68.99     72.24     75.47
## log_pSat_Pa    62.35    63.75     64.24     64.28     64.37     64.37     64.40
##              15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## X               78.69     81.92     85.01     87.76     90.27     92.55
## log_pSat_Pa     64.40     64.40     65.01     66.13     66.23     67.37
##              21 comps  22 comps  23 comps  24 comps  25 comps  26 comps
## X               94.59     96.39     97.80     99.05     99.67     99.97
## log_pSat_Pa     67.69     68.85     69.62     70.39     70.51     70.55
##              27 comps  28 comps  29 comps  30 comps  31 comps
## X               99.99    100.00    100.00    100.00    100.00
## log_pSat_Pa     70.61     70.67     70.67     70.67     70.67
```

predicting using 28 PCs (Lowest cross validation error occurs when using 28 PCs)

```
pcr_fit_28PCs = pcr(log_pSat_Pa ~ ., data = train_data[,-1], scale = TRUE, ncomp = 28)
```

Regression tree

```
set.seed(123)
rt_model = rpart(log_pSat_Pa ~., data = train_data[,-1])
```

```
printcp(rt_model)
```

```
##
## Regression tree:
## rpart(formula = log_pSat_Pa ~ ., data = train_data[, -1])
##
## Variables actually used in tree construction:
## [1] NumHBondDonors NumOfAtoms     NumOfC
##
## Root node error: 259317/26637 = 9.7352
##
## n= 26637
##
##         CP nsplit rel error  xerror      xstd
## 1 0.322456      0   1.00000 1.00005 0.0091677
## 2 0.089325      1   0.67754 0.67761 0.0064662
## 3 0.042422      2   0.58822 0.58833 0.0058630
```

```
## 4 0.037709      3   0.54580 0.54598 0.0054090
## 5 0.037317      4   0.50809 0.51173 0.0053037
## 6 0.016344      5   0.47077 0.47100 0.0050751
## 7 0.012279      6   0.45443 0.45468 0.0049913
## 8 0.011859      7   0.44215 0.44525 0.0049292
## 9 0.010000      8   0.43029 0.43158 0.0048350
```

```
error_train_rt = mean((train_data$log_pSat_Pa - predict(rt_model, train_data[,-1]))^2)
error_train_rt
```

```
## [1] 4.188949
```

High testing error

Tree pruning check . I think this is wrong cp = 0.01

```
Pruned_rt = prune(rt_model, cp = 0.01)

error_val_rt = mean((train_data$log_pSat_Pa - predict(Pruned_rt, train_data[,-1]))^2)
error_val_rt
```

```
## [1] 4.188949
```

Boosting

gradient Boosting by coco : Kaggle Score:0.7485

```
y_train <- train_data$log_pSat_Pa
x_train <- model.matrix(~ . - 1, data = train_data[,-c(1,2)])


dtrain <- xgb.DMatrix(data = x_train, label = y_train)
#dtest <- xgb.DMatrix(data = as.matrix(X_test))

params <- list(
  booster = "gbtree",
  objective = "reg:squarederror",
  eta = 0.1,
  max_depth = 6,
  subsample = 0.8,
  colsample_bytree = 0.8
)

xgb_model <- xgb.train(
  params = params,
  data = dtrain,
  nrounds = 100,
  watchlist = list(train = dtrain),
  verbose = 1
)
```

```
save(xgb_model, file = "xgb_model.RData")
```

I cannot run this

```
grid <- expand.grid(
  nrounds = c(100, 200),
  max_depth = c(3, 6, 9),
  eta = c(0.01, 0.1, 0.2),
  gamma = c(0, 1, 5),
  colsample_bytree = c(0.7, 0.8, 1),
  min_child_weight = c(1, 3, 5),
  subsample = c(0.7, 0.8, 1)
)

control <- trainControl(
  method = "cv",
  number = 5,
  verboseIter = TRUE
)

xgb_tuned <- train(
  x = x_train, y = y_train,
  method = "xgbTree",
  trControl = control,
  tuneGrid = grid
)

print(xgb_tuned$bestTune)
```

```
x_test <- model.matrix(~ . - 1, data = test_data[,-1])
y_hat = predict(xgb_tuned, newdata = x_test)
ID = test_data$ID
TARGET = as.vector(y_hat)
df = data.frame(ID, TARGET)
write.csv(df[, c("ID", "TARGET")], "dummy_submission.csv", row.names = FALSE)
```

Random forest (Kaggle score: 0.7440)

Default mtry is P/3, Here p = 24

Have to try with P = 8

Kaggle score: 0.7440

```
set.seed(123)

bag.rf = randomForest(log_pSat_Pa ~ ., data = train_data[,-1], mtry = 5, importance = TRUE)
```

```
yhat.bag = predict(bag.rf , newdata = test_data[,-1])
```

```
importance(bag.rf)
```

```
                                %IncMSE IncNodePurity
MW                           45.897727   12795.57215
NumOfAtoms                   41.912149   17391.89092
NumOfC                       69.995928   15087.04646
NumOfO                       49.467413    6320.67865
NumOfN                       25.628846    5277.43812
NumHBondDonors               79.439765   67922.64607
NumOfConf                    55.789496   41825.99036
NumOfConfUsed                64.000651   12644.26327
parentspecies                36.420462    3064.42500
C.C..non.aromatic.           54.002075    2389.86048
C.C.C.O.in.non.aromatic.ring 12.587809     243.31435
hydroxyl..alkyl.             32.789366    8853.75687
aldehyde                     53.453905    3019.51319
ketone                       55.952484    4482.04815
carboxylic.acid              48.594367   11361.55240
ester                        36.431927    1739.50999
ether..alicyclic.            44.172666    1569.99315
nitrate                      34.873295    2493.70582
nitro                        25.110952    1215.78542
aromatic.hydroxyl             5.418516      69.27385
carbonylperoxynitrate        50.057204    7195.35574
peroxide                     56.805588    1857.80184
hydroperoxide                28.198106    6782.46597
carbonylperoxyacid           33.511581    3311.32168
nitroester                   12.733955     211.22329
```

SVR Selected features from lasso : Kaggle score: 0.5195

Selected features from lasso is used : Removed "NumOfAtoms" , "NumOfN","C.C.C.O.in.non.aromatic.ring","hydroxyl..alkyl.' , "nitrate"

```
y_train <- train_data$log_pSat_Pa

x_train <- model.matrix(~ . - 1, data = train_data[,-c(1,2,4,7,13,14,20)])

svm_model <- svm(
  x = x_train, y = y_train,
  type = "eps-regression",
  kernel = "radial",
  cost = 1,
  gamma = 1 / ncol(x_train)
)
```

```
x_test <- model.matrix(~ . - 1, data = test_data[,-c(1,2,6,12,13,19)])
y_hat = predict(svm_model,newdata = x_test)
ID = test_data$ID
```

```
TARGET = as.vector(y_hat)
df = data.frame(ID, TARGET)
write.csv(df[, c("ID", "TARGET")], "dummy_submission.csv", row.names = FALSE)
```

Kaggle score: 0.5195

SVR using all the features: Kaggle score: 0.7554

```
y_train <- train_data$log_pSat_Pa

x_train <- model.matrix(~ . - 1, data = train_data[,-c(1,2)])

svm_model <- svm(
  x = x_train, y = y_train,
  type = "eps-regression",
  kernel = "radial",
  cost = 1,
  gamma = 1 / ncol(x_train)
)

save(svm_model, file = "svm_model.RData")
```

Kaggle score: 0.7554

Why “eps-regression” ??

SVR Without “eps-regression” same Kaggle score: 0.7554

```
y_train <- train_data$log_pSat_Pa

x_train <- model.matrix(~ . - 1, data = train_data[,-c(1,2)])

svm_model_test <- svm(
  x = x_train, y = y_train,
  kernel = "radial",
  cost = 1,
  gamma = 1 / ncol(x_train)
)

save(svm_model_test, file = "svm_model_test.RData")
```

check

In the R e1071 package, if gamma is not specified explicitly when using an RBF kernel, it is often set to 1 / ncol(x_train). This value is commonly used as a heuristic to start with, providing a reasonable balance between overfitting and underfitting.

#SVM Tuning : Run later

```
scaled_test_data = scale(test_data[,-c(1,10)])

test_data[,-c(1,10)] <- scale(test_data[,-c(1,10)])
```

```
y_train <- train_data$log_pSat_Pa

x_train <- model.matrix(~ . - 1, data = train_data[,-c(1,2)])

tune_grid <- expand.grid(
  sigma = c(0.001, 0.01, 0.1),
  C = c(0.1, 1, 10)
)

control <- trainControl(method = "cv", number = 5)

svm_tuned <- train(
  x = x_train, y = y_train,
  method = "svmRadial",
  tuneGrid = tune_grid,
  trControl = control
  #preProcess = c("center", "scale")
)

#print(svm_tuned$bestTune)
save(model, file = "svm_tuned")
```

Next step : Perform PCA for varible selection and then running SVM