R Notebook

Code

Loading the libraries

LAB_9_20122065

Hide

```
library(ggplot2)
library(caret)
library(dplyr)
library(caTools)
library(corrplot)
```

Loading the data set

Hide

```
df = read.table('/home/thomaskutty/Documents/my_folders/mcs2sem/R_lab_msc/datafolder/
energy.csv', header = TRUE, sep = ',')
head(df)
```

	date <fctr>	Appliances <int>	lights <int>	T1 <dbl>	RH_1 <dbl>	T2 <dbl>	RH_2 <dbl>	T3 <dbl>	RH_3 <dbl>
1	2016-01-11 17:00:00	60	30	19.89	47.59667	19.2	44.79000	19.79	44.73000
2	2016-01-11 17:10:00	60	30	19.89	46.69333	19.2	44.72250	19.79	44.79000
3	2016-01-11 17:20:00	50	30	19.89	46.30000	19.2	44.62667	19.79	44.93333
4	2016-01-11 17:30:00	50	40	19.89	46.06667	19.2	44.59000	19.79	45.00000
5	2016-01-11 17:40:00	60	40	19.89	46.33333	19.2	44.53000	19.79	45.00000
6	2016-01-11 17:50:00	50	40	19.89	46.02667	19.2	44.50000	19.79	44.93333

6 rows | 1-10 of 29 columns

Note: We need to predict the random variable rv1 using the linear regression model with and with out Ridge Regularization ; So, Lets start creating our first linear regression model using highly correlated variables

Splitting the data set into train and test

Hide

```
# splitting the data set
df = subset(df,select = -c(date,rv2))
sample<-sample.split(df$rv1,SplitRatio =0.7)
train=subset(df,sample==TRUE)
test=subset(df,sample==FALSE)
```

Getting the structure of the train data

Hide

```
str(train)
```

```
'data.frame':   13814 obs. of  27 variables:
 $ Appliances : int  50 50 60 60 60 60 70 580 250 100 ...
 $ lights     : int  30 40 40 50 50 40 40 60 40 10 ...
 $ T1         : num  19.9 19.9 19.9 19.9 19.9 ...
 $ RH_1       : num  46.3 46.1 46.3 45.8 45.6 ...
 $ T2         : num  19.2 19.2 19.2 19.2 19.2 ...
 $ RH_2       : num  44.6 44.6 44.5 44.5 44.5 ...
 $ T3         : num  19.8 19.8 19.8 19.8 19.7 ...
$ RH_3       : num  44.9 45 45 44.9 44.9 ...
 $ T4         : num  18.9 18.9 18.9 18.9 18.9 ...
 $ RH_4       : num  45.9 45.7 45.5 45.8 45.9 ...
 $ T5         : num  17.2 17.2 17.2 17.1 17.1 ...
$ RH_5       : num  55.1 55.1 55.1 55 54.9 ...
 $ T6         : num  6.56 6.43 6.37 6.26 6.19 ...
 $ RH_6       : num  83.2 83.4 84.9 86.1 86.4 ...
 $ T7         : num  17.2 17.1 17.2 17.1 17.1 ...
 $ RH_7       : num  41.4 41.3 41.2 41.2 41.2 ...
 $ T8         : num  18.2 18.1 18.1 18.1 18.1 ...
 $ RH_8       : num  48.7 48.6 48.6 48.6 48.6 ...
$ T9         : num  17 17 17 17 17 ...
 $ RH_9       : num  45.5 45.4 45.4 45.3 45.3 ...
 $ T_out      : num  6.37 6.25 6.13 5.9 5.92 5.93 5.95 5.98 6 6 ...
 $ Press_mm_hg: num  734 734 734 734 734 ...
 $ RH_out     : num  92 92 92 92 91.8 ...
 $ Windspeed  : num  6.33 6 5.67 5 5.17 ...
 $ Visibility : num  55.3 51.5 47.7 40 40 ...
 $ Tdewpoint  : num  5.1 5 4.9 4.7 4.68 4.67 4.65 4.62 4.52 4.43 ...
 $ rv1        : num  28.6 45.4 10.1 47.2 33 ...
```

Note: We can see that except date all other features are numerical

Now we find the correlation between variables using heatmap and cor() function

Hide

```
correlation.train = cor(train)
```

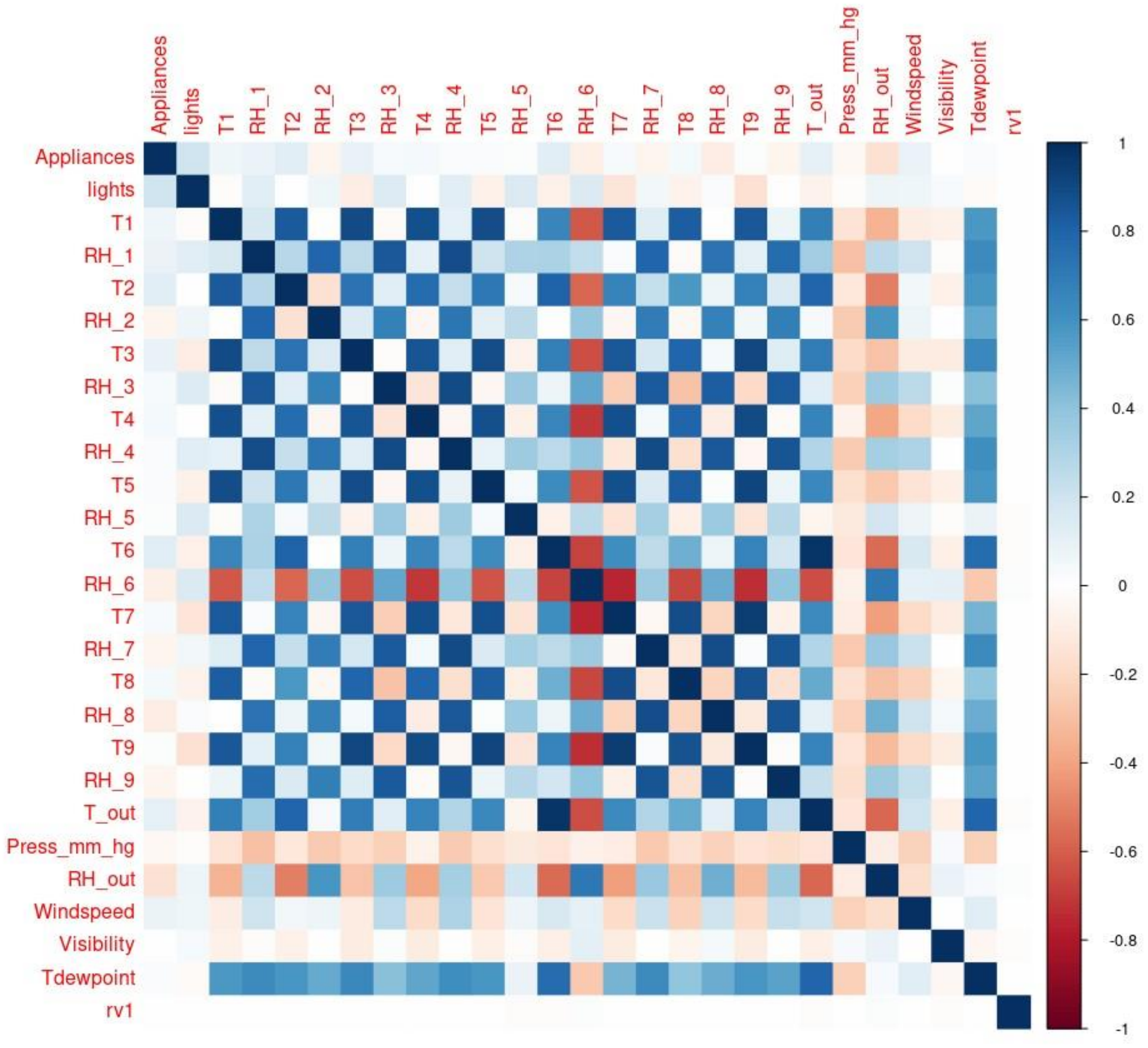
It is difficult to find which all the variables are highly correlated with the target variable

And there is a correlation of 1.0 with rv2 variable (because both are exactly the same feature so we have to remove that)

So, lets visualize the correlation with the heat map

Hide

```
corrplot(correlation.train,method ='color')
```



Note: We can see that all the variable are less corrrelated with the target variable. But lets try creating a linear model using all the variables.

Creating the model

Hide

```
# creating the model using all the features
model = lm(rv1~.,data = train)
# getting the summary of the model
summary(model)
```

```
 Call: lm(formula = rv1 ~ ., data
= train)

Residuals:
     Min       1Q   Median       3Q      Max
-26.4043 -12.5059  -0.0943  12.5652  26.5526

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.905575  17.470010  -0.052   0.9587
Appliances  -0.000279   0.001329  -0.210   0.8337
lights     -0.006356   0.017879  -0.356   0.7222
T1          0.263076   0.339178   0.776   0.4380
RH_1         0.075154   0.123104   0.610   0.5415
T2          -0.223466   0.302094  -0.740   0.4595
RH_2        -0.102595   0.142545  -0.720   0.4717
T3          -0.104766   0.196644  -0.533   0.5942
RH_3         0.041569   0.124443   0.334   0.7384
T4           0.206003   0.189934   1.085   0.2781
RH_4         0.098598   0.117865   0.837   0.4029
T5          -0.382413   0.217034  -1.762   0.0781 .
RH_5        -0.023014   0.016058  -1.433   0.1518
T6          -0.011412   0.117232  -0.097   0.9225
RH_6         0.009227   0.012530   0.736   0.4615
T7          -0.093587   0.244271  -0.383   0.7016
RH_7         0.019647   0.079220   0.248   0.8041
T8          -0.031227   0.179043  -0.174   0.8615
RH_8         0.083199   0.068997   1.206   0.2279
T9           0.599346   0.328248   1.826   0.0679 .
RH_9        -0.146592   0.076027  -1.928   0.0539 .
T_out        0.142087   0.284484   0.499   0.6175
Press_mm_hg  0.022486   0.019682   1.142   0.2533
RH_out       0.036094   0.058637   0.616   0.5382
Windspeed    0.018922   0.063942   0.296   0.7673
Visibility  -0.016503   0.010682  -1.545   0.1224
Tdewpoint   -0.244133   0.276965  -0.881   0.3781
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.49 on 13787 degrees of freedom
Multiple R-squared:  0.001618,  Adjusted R-squared:  -0.0002646
F-statistic: 0.8595 on 26 and 13787 DF,  p-value: 0.6695
```

If you have a statistically significant overall F-test, you can draw several other conclusions.

For the model with no independent variables, the intercept-only model, all of the model’s predictions equal the mean of the dependent variable. Consequently, if the overall F-test is statistically significant, your model’s predictions are an improvement over using the mean.

But here can see that the p value corresponding to the f statistic is very much greater thatn 0.05 so we must accept the null hypothesis that all beta coefficients are zero.

Hide

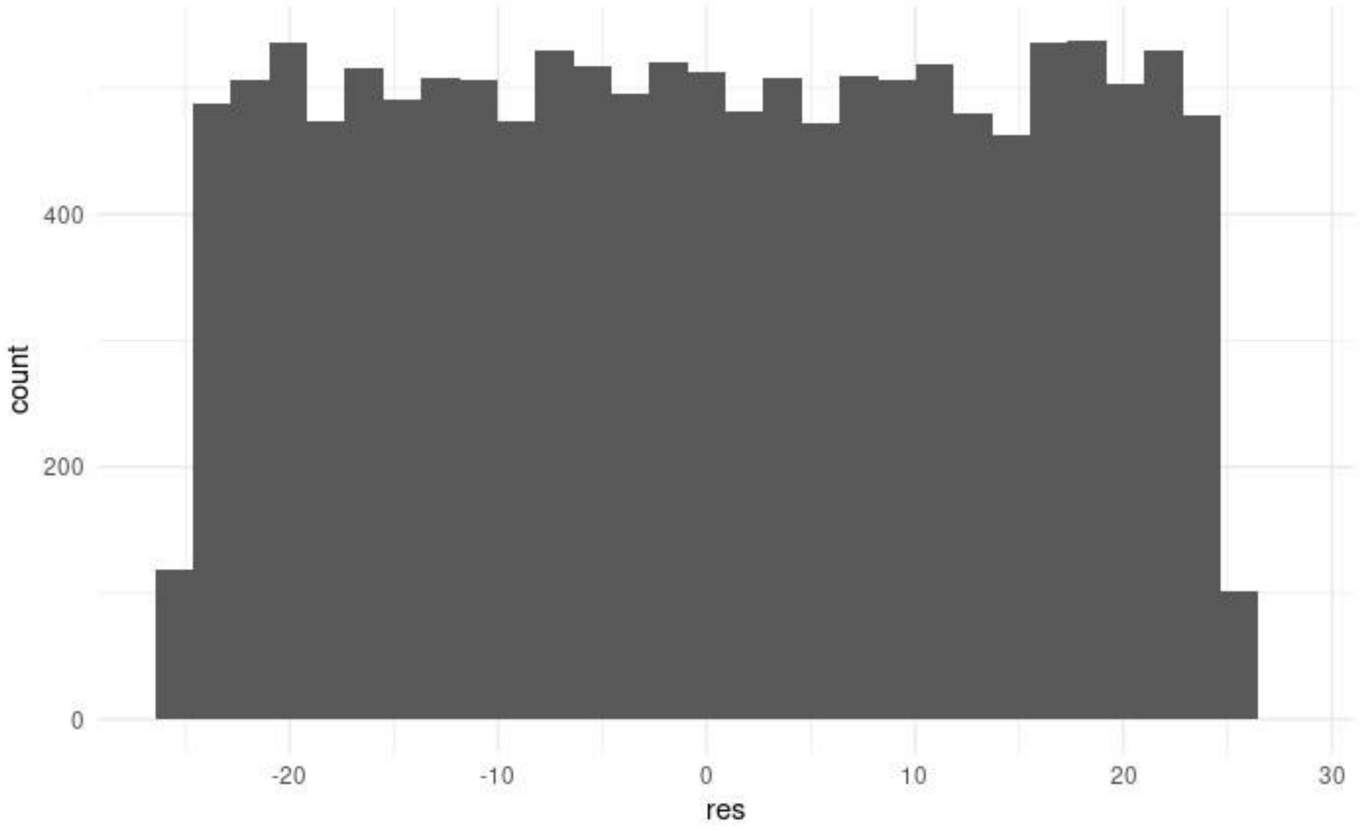
```
# saving the model residuals
res = residuals(model)
res = as.data.frame(res)
head(res)
```

	res <dbl>
3	3.937153
4	20.723399
5	-14.667963
7	22.353673
8	8.155655
9	6.547019

6 rows

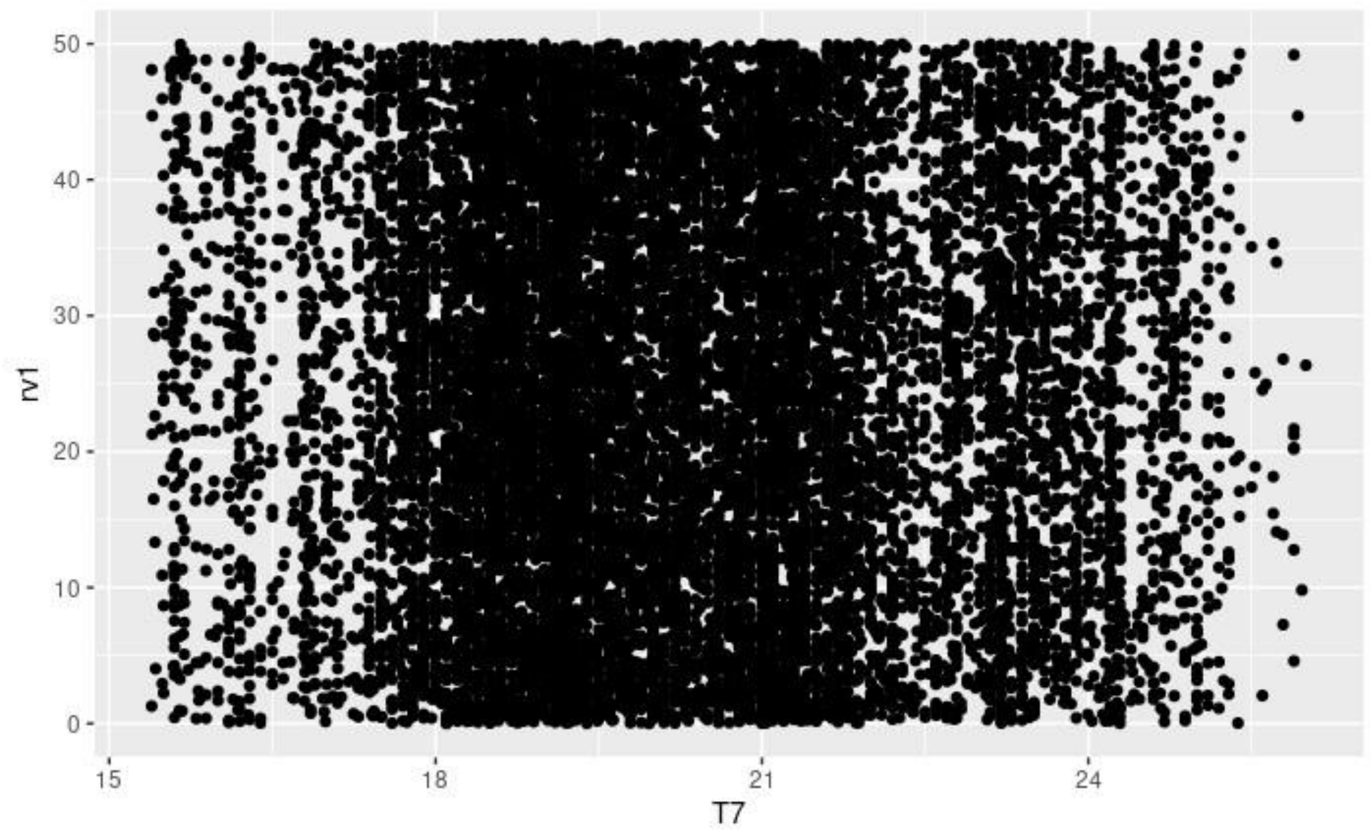
Hide

```
pl_residuals = ggplot(res,aes(res))+geom_histogram()+ theme_minimal()
pl_residuals
```



Hide

```
ggplot(data = train, aes(y = rv1, x = T7))+ geom_point()
```



rv1
50
40
30
20
10
0
15
18
21
24
T7

Ridge regression

Hide

```
# testing the model in data set
rv1.prediction = predict(model,test)
rv1.prediction = as.data.frame(rv1.prediction)
head(rv1.prediction)
```

```
   rv1.prediction
            <dbl>
1        24.65946
2        24.72118
6        24.87846
11       24.81796
13       25.05435
17       25.75085
6 rows
```

Hide

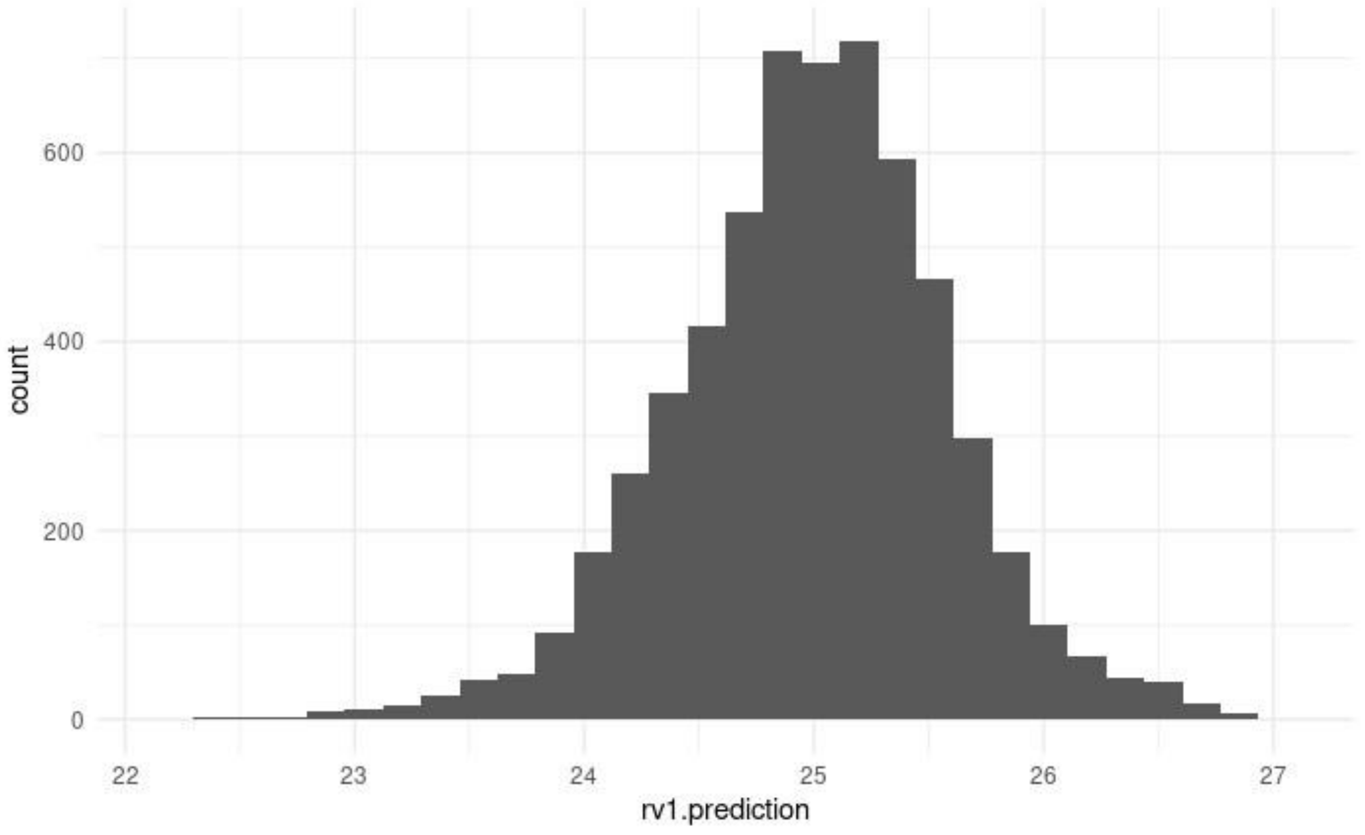
```
result = cbind(rv1.prediction, test$rv1)
colnames(result) = c('pred', 'real')
result = as.data.frame(result)
result
```

	pred <dbl>	real <dbl>
1	24.65946	13.27543316
2	24.72118	18.60619498
6	24.87846	44.91948425
11	24.81796	10.29872874
13	25.05435	34.35114233
17	25.75085	35.88092541
29	24.82262	43.48454229
31	24.82543	24.10400577
32	24.68393	29.97829127
33	24.70393	24.67706535

1-10 of 5,921 rows Previous **1** 2 3 4 5 6 ... 100 Next

Hide

```
pl_residuals_test = ggplot(rv1.prediction,aes(rv1.prediction))+geom_histogram()+ them
e_minimal()
pl_residuals_test
```



Hide

```
# getting the mean square value
mse  = mean((result$real - result$pred)2)
print(mse)
```

```
[1] 210.8825
```

Hide

```
# calculating the rmse
rmse = mse^0.5
print(rmse)
```

```
[1] 14.52179
```

Hide

```
sse = sum((result$pred - result$real)2)
sst = sum((mean(test$rv1) - result$real)2)
r2 = 1 - (sse/sst)
sse
```

```
[1] 1248635
```

Hide

```
sst
```

```
[1] 1247447
```

Hide

```
r2
```

```
[1] -0.0009522422
```

Hide

```


# ridge regression for overcoming overfitting
library(dplyr)
library(caret)

num.cols = sapply(df,is.numeric)
attach(df)
```

```
The following objects are masked from df (pos = 3):

    Appliances, lights, Press_mm_hg, RH_1, RH_2, RH_3, RH_4, RH_5, RH_6, RH_7, RH_8,
RH_9, RH_out, rv1,
    T_out, T1, T2, T3, T4, T5, T6, T7, T8, T9, Tdewpoint, Visibility, Windspeed
```

Hide

```
dummies = dummyVars(rv1 ~., data = df[,num.cols])

train_dummies = predict(dummies, newdata = train[,num.cols])
test_dummies = predict(dummies,newdata = test[,num.cols])
print(dim(train_dummies))
```

```
[1] 13814    26
```

Hide

```
print(dim(test_dummies))
```

```
[1] 5921   26
```

Hide

```
library(glmnet)
```

```
Loading required package: Matrix
Loaded glmnet 4.1-1
```

Hide

```
x = as.matrix(train_dummies)
x_test = as.matrix(test_dummies)


y_train = train$rv1
y_test = test$rv1

lambdas = 10^seq(2,-3,by = -.1)
ridge_reg  = glmnet(x,y_train, nlambda =25, alpha = 0, family= 'gaussian', lambda =
 lambdas)

summary(ridge_reg)
```

```
          Length Class     Mode
a0          51   -none-    numeric
beta      1326   dgCMatrix S4
df          51   -none-    numeric
dim          2   -none-    numeric
lambda      51   -none-    numeric
dev.ratio   51   -none-    numeric
nulldev      1   -none-    numeric
npasses      1   -none-    numeric
jerr         1   -none-    numeric
offset       1   -none-    logical
call         7   -none-    call
nobs         1   -none-    numeric
```

Hide

```
# getting the optimal value of lambda
cv_ridge <- cv.glmnet(x, y_train, alpha =0, lambda = lambdas)
optimal_lambda <- cv_ridge$lambda.min
optimal_lambda
```

```
[1] 100
```

Hide

```
# Compute R^2 from true and predicted values
eval_results <- function(true, predicted, df) {
    SSE <- sum((predicted - true)^2)
    SST <- sum((true - mean(true))^2)
    R_square <- 1 - SSE / SST
    RMSE = sqrt(SSE/nrow(df))
    # Model performance metrics
    data.frame(
        RMSE = RMSE,
        Rsquare = R_square
    )
}
```

Hide

```
# Prediction and evaluation on train data
predictions_train <- predict(ridge_reg, s = optimal_lambda, newx = x)
eval_results(y_train, predictions_train, train)
```

RMSE <dbl>	Rsquare <dbl>
14.48652	0.0002431454

1 row

Hide

```
NA
NA
NA
NA
```

Hide

```
# Prediction and evaluation on test data
predictions_test <- predict(ridge_reg, s = optimal_lambda, newx = x_test)
eval_results(y_test, predictions_test, test)
```

RMSE <dbl>	Rsquare <dbl>
14.51307	0.0002499472

1 row