Regression- kNN, Linear, Decision tree

Sai Gonuguntla

https://www.kaggle.com/datasets/budincsevity/szeged-weather

10/05/2022

in linear regression the target variable y is quantitative and x is the predictor which can be quantitative or qualitative. It displays the relationship between x and y. Strengths are,it works well when there is a linear pattern in the data,it has low variance,and is a pretty simple algorithm. Weaknesses are, it has high bias because its looking for a linear relation in the data, so it doesn’t perform well when there are non-linear relationships.

#read in the data set

```
#options(stringsAsFactors = FALSE)
df <- read.csv("weatherHistory-2.csv", header=TRUE)

whist <- df[,c(4,5,6,7,8,9,11)]
attach(whist)

str(whist)
```

```
## 'data.frame':    96453 obs. of  7 variables:
##  $ Temperature..C.         : num  9.47 9.36 9.38 8.29 8.76 ...
##  $ Apparent.Temperature..C.: num  7.39 7.23 9.38 5.94 6.98 ...
##  $ Humidity                : num  0.89 0.86 0.89 0.83 0.83 0.85 0.95 0.89 0.82 0.7
2 ...
##  $ Wind.Speed..km.h.       : num  14.12 14.26 3.93 14.1 11.04 ...
##  $ Wind.Bearing..degrees.  : num  251 259 204 269 259 258 259 260 259 279 ...
##  $ Visibility..km.         : num  15.8 15.8 15 15.8 15.8 ...
##  $ Pressure..millibars.    : num  1015 1016 1016 1016 1017 ...
```

a. Divide into train and test

```
set.seed(1234)
i <- sample(1:nrow(df), 0.80*nrow(df), replace=FALSE)
train <- whist[i,]
test <- whist[-i,]
```

b. 5 R functions for data exploration using the training data.

```
tail(train,n=2)
```

	Temperature..C. <dbl>	Apparent.Temperature..C. <dbl>	Humidity <dbl>	Wind.Speed..km.h. <dbl>	Wir
57481	13.894444	13.894444	0.69	3.1234	
31612	1.111111	-2.272222	0.85	11.2700	

2 rows | 1-6 of 8 columns

```
dim(train)
```

```
## [1] 77162     7
```

```
str(train)
```

```
## 'data.frame':    77162 obs. of  7 variables:
##  $ Temperature..C.         : num  8 13.4 8.89 17.83 1.31 ...
##  $ Apparent.Temperature..C.: num  6.76 13.4 7.71 17.83 1.31 ...
##  $ Humidity                : num  0.99 0.83 0.93 0.96 1 0.73 0.93 0.62 0.76 0.93 .
..
##  $ Wind.Speed..km.h.       : num  7.63 17.11 8.05 12.41 3.53 ...
##  $ Wind.Bearing..degrees.  : num  121 138 130 220 247 179 271 341 138 41 ...
##  $ Visibility..km.         : num  0.612 15.875 3.059 8.372 0.37 ...
##  $ Pressure..millibars.    : num  1021 1009 1009 1012 1032 ...
```

```
summary(train)
```

```
##  Temperature..C.   Apparent.Temperature..C.    Humidity      Wind.Speed..km.h.
##  Min.   :-21.822   Min.   :-27.717          Min.   :0.0000   Min.   : 0.000
##  1st Qu.:  4.614   1st Qu.:  2.283          1st Qu.:0.6000   1st Qu.: 5.796
##  Median : 11.978   Median : 11.978          Median :0.7800   Median : 9.918
##  Mean   : 11.933   Mean   : 10.855          Mean   :0.7348   Mean   :10.799
##  3rd Qu.: 18.839   3rd Qu.: 18.839          3rd Qu.:0.8900   3rd Qu.:14.120
##  Max.   : 39.589   Max.   : 38.661          Max.   :1.0000   Max.   :63.853
##  Wind.Bearing..degrees. Visibility..km. Pressure..millibars.
##  Min.   :  0.0          Min.   : 0.00   Min.   :   0
##  1st Qu.:116.0          1st Qu.: 8.34   1st Qu.:1012
##  Median :180.0          Median :10.05   Median :1016
##  Mean   :187.6          Mean   :10.34   Mean   :1003
##  3rd Qu.:290.0          3rd Qu.:14.81   3rd Qu.:1021
##  Max.   :359.0          Max.   :16.10   Max.   :1046
```

```
head(train,n=5)
```

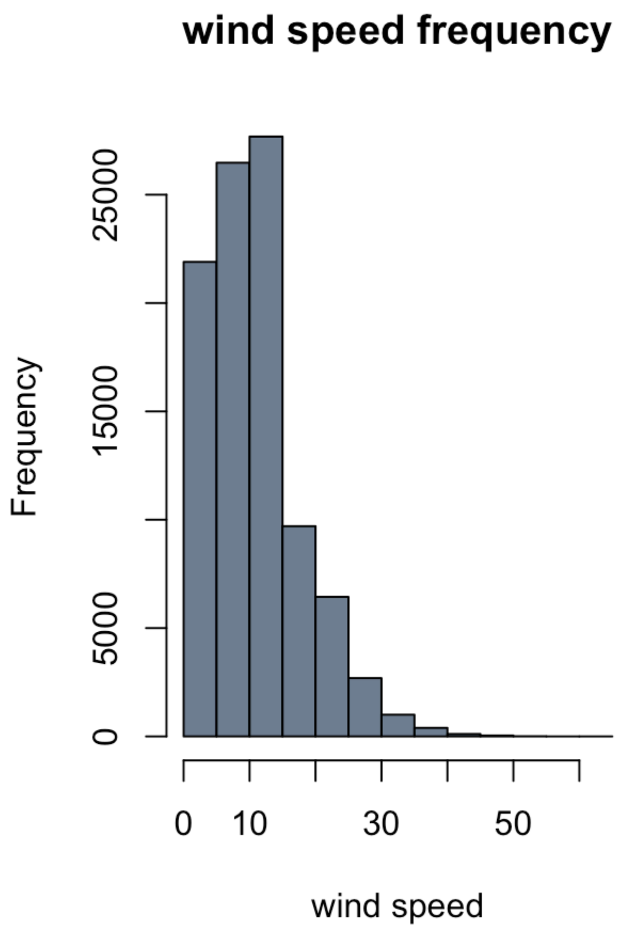
	Temperature..C. <dbl>	Apparent.Temperature..C. <dbl>	Humidity <dbl>	Wind.Speed..km.h. <dbl>	Wir
41964	8.000000	6.761111	0.99	7.6314	
15241	13.400000	13.400000	0.83	17.1143	
33702	8.888889	7.705556	0.93	8.0500	
83023	17.827778	17.827778	0.96	12.4131	
80756	1.311111	1.311111	1.00	3.5259	

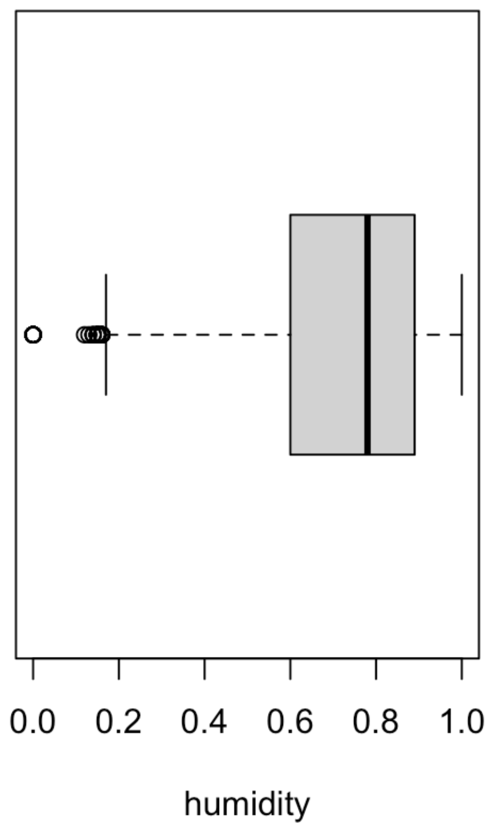
5 rows | 1-6 of 8 columns

c. 2 informative graphs, using the training data.

Displays a histogram that shows windspeed frequency and a box plot which displays the humidity.

```
par(mfrow=c(1,2))
hist(Wind.Speed..km.h., col="slategray", main="wind speed frequency",xlab="wind speed
")
boxplot(Humidity,horizontal=TRUE, xlab="humidity")
```





Build a linear regression with all predictors except columns 1,2,3,10 and 12

```
lm1 <- lm(Apparent.Temperature..C.~., data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = Apparent.Temperature..C. ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3617 -0.7152 -0.1068  0.6824  5.3455
##
## Coefficients:
##                          Estimate Std. Error  t value Pr(>|t|)
## (Intercept)            -2.605e+00  4.439e-02  -58.684  < 2e-16 ***
## Temperature..C.         1.125e+00  5.472e-04 2056.862  < 2e-16 ***
## Humidity                1.052e+00  2.702e-02   38.955  < 2e-16 ***
## Wind.Speed..km.h.      -9.569e-02  5.910e-04 -161.925  < 2e-16 ***
## Wind.Bearing..degrees.  5.372e-04  3.642e-05   14.750  < 2e-16 ***
## Visibility..km.         1.105e-04  1.026e-03    0.108    0.914
## Pressure..millibars.    1.868e-04  3.317e-05    5.630 1.81e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.078 on 77155 degrees of freedom
## Multiple R-squared:  0.9899, Adjusted R-squared:  0.9899
## F-statistic: 1.255e+06 on 6 and 77155 DF,  p-value: < 2.2e-16
```

Evaluate

```
pred1 <- predict(lm1, newdata=test)
cor1 <- cor(pred1, test$Apparent.Temperature..C.)
mse1 <- mean((pred1-test$Apparent.Temperature..C.)^2)
print(paste('correlation:', cor1))
```

```
## [1] "correlation: 0.994878444357297"
```

```
print(paste('mse:', mse1))
```

```
## [1] "mse: 1.15909871906881"
```

kNN for regression

cor is 0.9907, mse is 4.638

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
# fit the model
fit <- knnreg(train[,1:7],train[,1],k=3)

# Evaluate
pred2 <- predict(fit, test[,1:7])
cor_knn1  <- cor(pred2, test$Apparent.Temperature..C.)
mse_knn1 <- mean((pred2- test$Apparent.Temperature..C.)^2)
rmse_knn1 <- (sqrt(mse_knn1))

print(paste('correlation:', cor_knn1))
```

```
## [1] "correlation: 0.990757671363682"
```

```
print(paste('mse:',  mse_knn1))
```

```
## [1] "mse: 4.63857430739458"
```

Scale the data

```
train_scaled <- train[,1:7]
means <- sapply(train_scaled,mean)
stdvs <- sapply(train_scaled,sd)
train_scaled <- scale(train_scaled, center=means, scale=stdvs)
test_scaled <- scale(test[,1:7], center=means, scale=stdvs)
```

kNN on scaled data

scaling the data performed better. cor before was 0.9908 and after is 0.9986. mse before was 4.6386, and now is 0.3296

```
fit <- knnreg(train_scaled, train$Apparent.Temperature..C., k=3)
pred3 <- predict(fit, test_scaled)
cor_knn2  <- cor(pred3, test$Apparent.Temperature..C.)
mse_knn2 <- mean((pred3 - test$Apparent.Temperature..C.)^2)

print(paste('correlation:', cor_knn2))
```

```
## [1] "correlation: 0.998561485590879"
```

```
print(paste('mse:', mse_knn2))
```

```
## [1] "mse: 0.329554309396625"
```

Find the best k

Try various values of k. K=9 gves the best values

```
cor_k <- rep(0,20)
mse_k <- rep(0,20)

i <- 1
for (k in seq(1, 10, 2)){
  fit_k <- knnreg(train_scaled,train$Apparent.Temperature..C., k=k)
  pred_k <-predict(fit_k, test_scaled)
  cor_k[i] <- cor(pred_k, test$Apparent.Temperature..C.)
  mse_k[i] <- mean((pred_k - test$Apparent.Temperature..C.)^2)
  print(paste("k=", k, cor_k[i], mse_k[i]))
  i <- i + i
}
```

```
## [1] "k= 1 0.997540182758721 0.557735966877665"
## [1] "k= 3 0.998561485590879 0.329554309396625"
## [1] "k= 5 0.998750024604675 0.290759920432771"
## [1] "k= 7 0.998800549217474 0.283070876112961"
## [1] "k= 9 0.998837244464568 0.278818882197203"
```

```
which.min(mse_k)
```

```
## [1] 3
```

```
which.max(cor_k)
```

```
## [1] 16
```

scaled knn regression

cor is 0.9988 and mse is 0.2788 which is better than on the scaled data using k=3

```
fit <- knnreg(train_scaled, train$Apparent.Temperature..C., k=9)
pred4 <- predict(fit, test_scaled)
cor_knn3 <- cor(pred4, test$Apparent.Temperature..C.)
mse_knn3 <- mean((pred4 - test$Apparent.Temperature..C.)^2)

print(paste("cor=", cor_knn3))
```

```
## [1] "cor= 0.998837244464568"
```

```
print(paste("mse=", mse_knn3))
```

```
## [1] "mse= 0.278818882197203"
```

Using tree

Correlation was 0.9786 and rmse was 2.19

```
#install.packages("tree")
library(tree)

tree1 <- tree(Apparent.Temperature..C.~., data=train)
summary(tree1)
```

```
##
## Regression tree:
## tree(formula = Apparent.Temperature..C. ~ ., data = train)
## Variables actually used in tree construction:
## [1] "Temperature..C."
## Number of terminal nodes:  7
## Residual mean deviance:  4.931 = 380400 / 77160
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -19.62000  -1.58500  -0.04461   0.00000   1.53800   8.96800
```

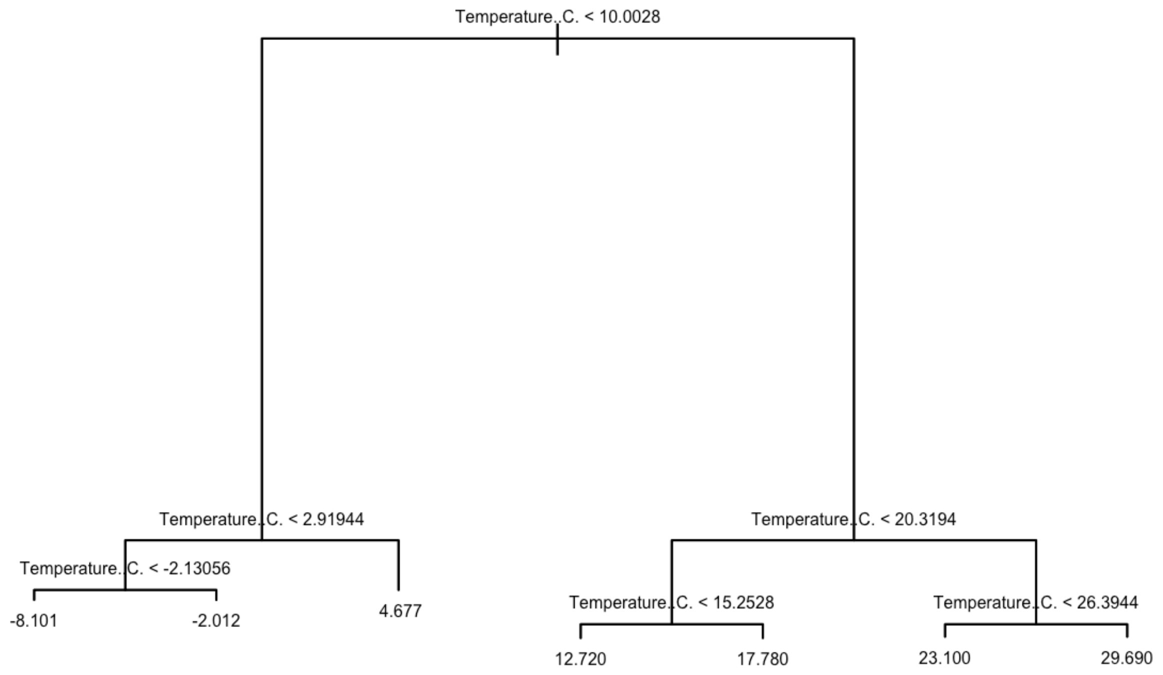
```
pred5 <- predict(tree1, newdata=test)
print(paste('correlation:', cor(pred5, test$Apparent.Temperature..C.)))
```

```
## [1] "correlation: 0.978610523738168"
```

```
rmse_tree <- sqrt(mean((pred5-test$Apparent.Temperature..C.)^2))
print(paste('rmse:', rmse_tree))
```

```
## [1] "rmse: 2.19112277024394"
```

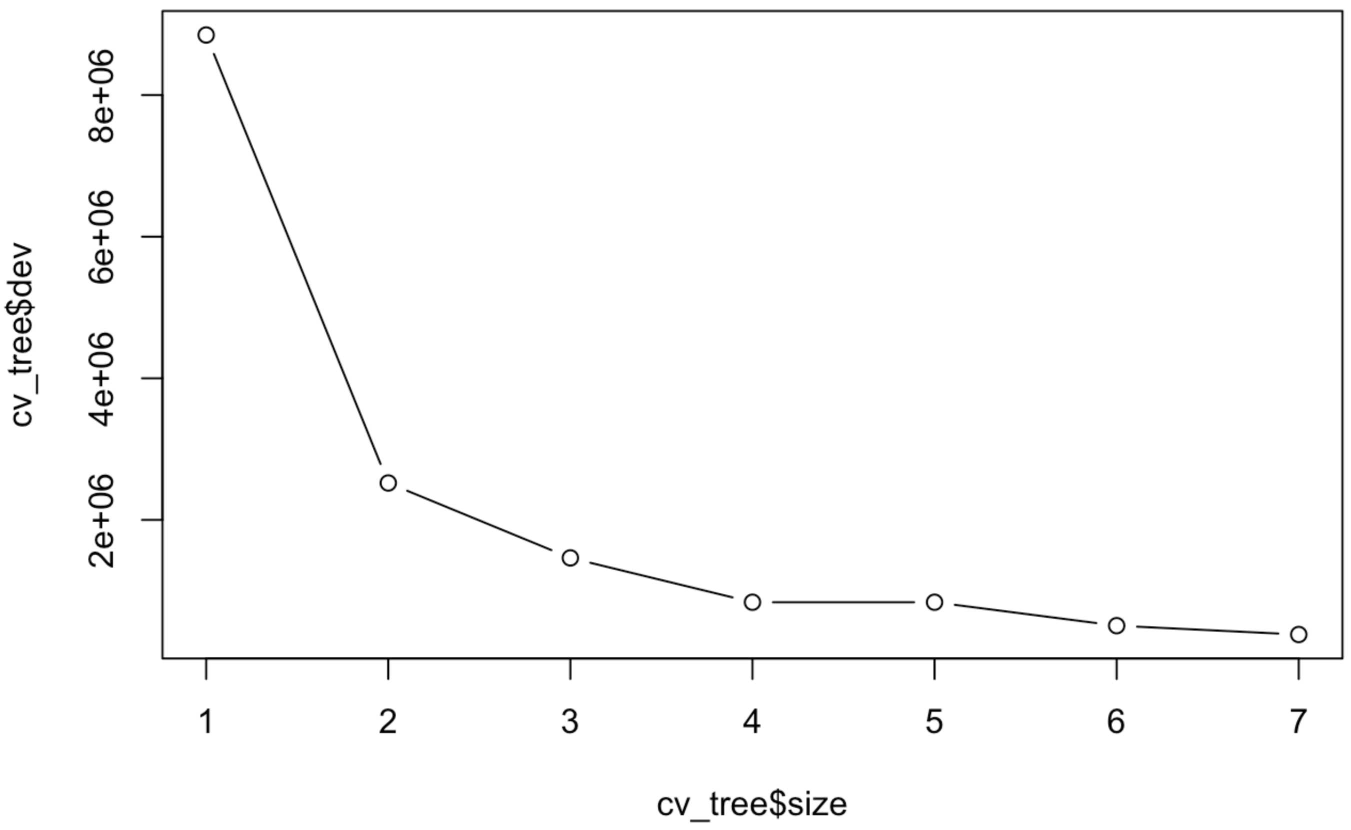
```
plot(tree1)
text(tree1, cex=0.5, pretty=0)
```


Temperature..C. < 10.0028
Temperature..C. < 2.91944
Temperature..C. < 20.3194
Temperature..C. < -2.13056
-8.101
-2.012
4.677
Temperature..C. < 15.2528
12.720
17.780
Temperature..C. < 26.3944
23.100
29.690

Cross validation

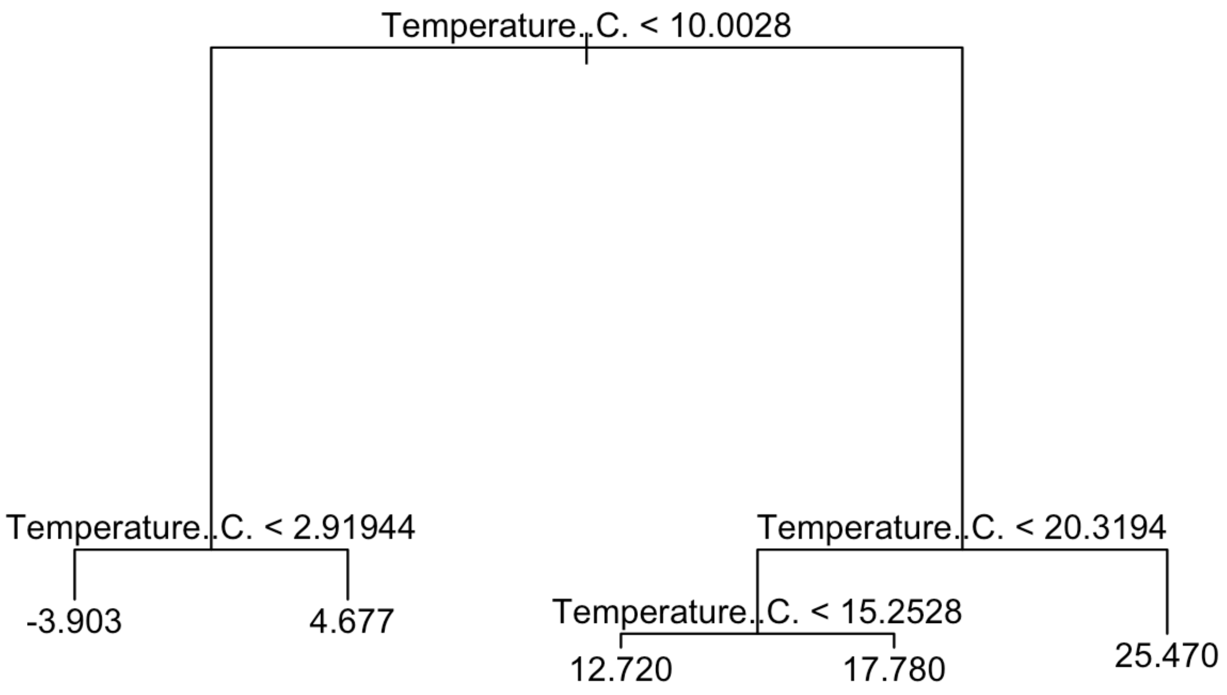
to try to get better results

```
cv_tree <- cv.tree(tree1)
plot(cv_tree$size, cv_tree$dev, type='b')
```



prune the tree

```
tree_pruned <- prune.tree(tree1, best=5)
plot(tree_pruned)
text(tree_pruned, pretty=0)
```


Temperature..C. < 10.0028
Temperature..C. < 2.91944
-3.903
4.677
Temperature..C. < 20.3194
Temperature..C. < 15.2528
12.720
17.780
25.470

test on pruned tree

For the unpruned tree, correlation was 0.9786 and rmse was 2.19. The cor for the pruned tree is 0.9625 which is slightly lower than the correlation for the unpruned tree, and the rmse is 2.8898 which is higher. So pruning didn't improve the result. Although decision tree didn't perform the best, random forest produced very good results with a correlation of 0.9996 and rmse of 0.3133. Bagging also performed really well with the correlation of 0.9999 and rmse of 0.1607. For linear regression the correlation was 0.9949 and mse was 1.1591 so it performed better than using decision trees. Knn regression had a cor of 0.9908 and mse of 4.6386 and after scaling using k=9 we got a cor of 0.9988 and mse of 0.2788 which is much better. So overall not taking random forest and bagging into account, knn performed the best, then linear regression, followed by decision tree regression.

```
pred_pruned <- predict(tree_pruned, newdata = test)
cor_pruned <- cor(pred_pruned, test$Apparent.Temperature..C.)
rmse_pruned <- rmse_pruned <- sqrt(mean((pred_pruned-test$Apparent.Temperature..C.)^2
))

print(paste('cor:', cor_pruned))
```

```
## [1] "cor: 0.962487132707812"
```

```
print(paste('rmse:', rmse_pruned))
```

```
## [1] "rmse: 2.88983347843392"
```

Random Forest

Will outperform the decision tree but will lose interpretability.

```
#install.packages("randomForest")
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(1234)
rf <- randomForest(Apparent.Temperature..C.~., data=train, importance=TRUE)
rf
```

```
##
## Call:
##  randomForest(formula = Apparent.Temperature..C. ~ ., data = train,      importanc
e = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 0.1098514
##                     % Var explained: 99.9
```

predict on the random forest

Correlation is slightly higher and rmse is much lower than what we got for linear regression.

```
pred_rf <- predict(rf, newdata = test)
cor_rf <- cor(pred_rf, test$Apparent.Temperature..C.)
print(paste('cor:', cor_rf))
```

```
## [1] "cor: 0.999605069000984"
```

```
rmse_rf <- sqrt(mean((pred_rf-test$Apparent.Temperature..C.)^2))
print(paste('rmse:',rmse_rf))
```

```
## [1] "rmse: 0.313284504282562"
```

bagging

```
bag <- randomForest(Apparent.Temperature..C.~., data=train, mtry = 6)
bag
```

```
##
## Call:
##  randomForest(formula = Apparent.Temperature..C. ~ ., data = train,      mtry = 6)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##           Mean of squared residuals: 0.02565725
##                     % Var explained: 99.98
```

predict on bagging

bagging also produced good results with a high correlation of 0.9999 and rmse of 0.1607. It also did best out of all the algorithms.

```
pred_bag <- predict(bag, newdata = test)
cor_bag <- cor(pred_bag, test$Apparent.Temperature..C.)
print(paste('cor:', cor_bag))
```

```
## [1] "cor: 0.999887006056601"
```

```
rmse_bag <- sqrt(mean((pred_bag-test$Apparent.Temperature..C.)^2))
print(paste('rmse:',rmse_bag))
```

```
## [1] "rmse: 0.16072651526701"
```

Linear regression outperformed decision tree regression likely due to the fact that linear regression usually does better than decision trees when the underlying function is linear. In the dataset used there is a strong linear relation hence the result. Decision trees suffers from high variance so it didn’t do that well. However, random forest produced very good results because it counteracts the high variance. knn regression after scaling performed the best out of all algorithms overall likely because knn works well on low dimensions and there were only 6 features.