KBLMM - Homework 2

SVM for regression on Yacht Hydrodynamics dataset

Mikołaj Małkiński

09 October, 2019

Required libraries

```
library(kernlab)
library(ggplot2)
library(patchwork)
library(tidyr)
library(dplyr)
library(tidyverse)
library(reshape2)
library(kernlab)
library(caret)
library(modelr)
library(broom)
```

Exploratory Data Analysis

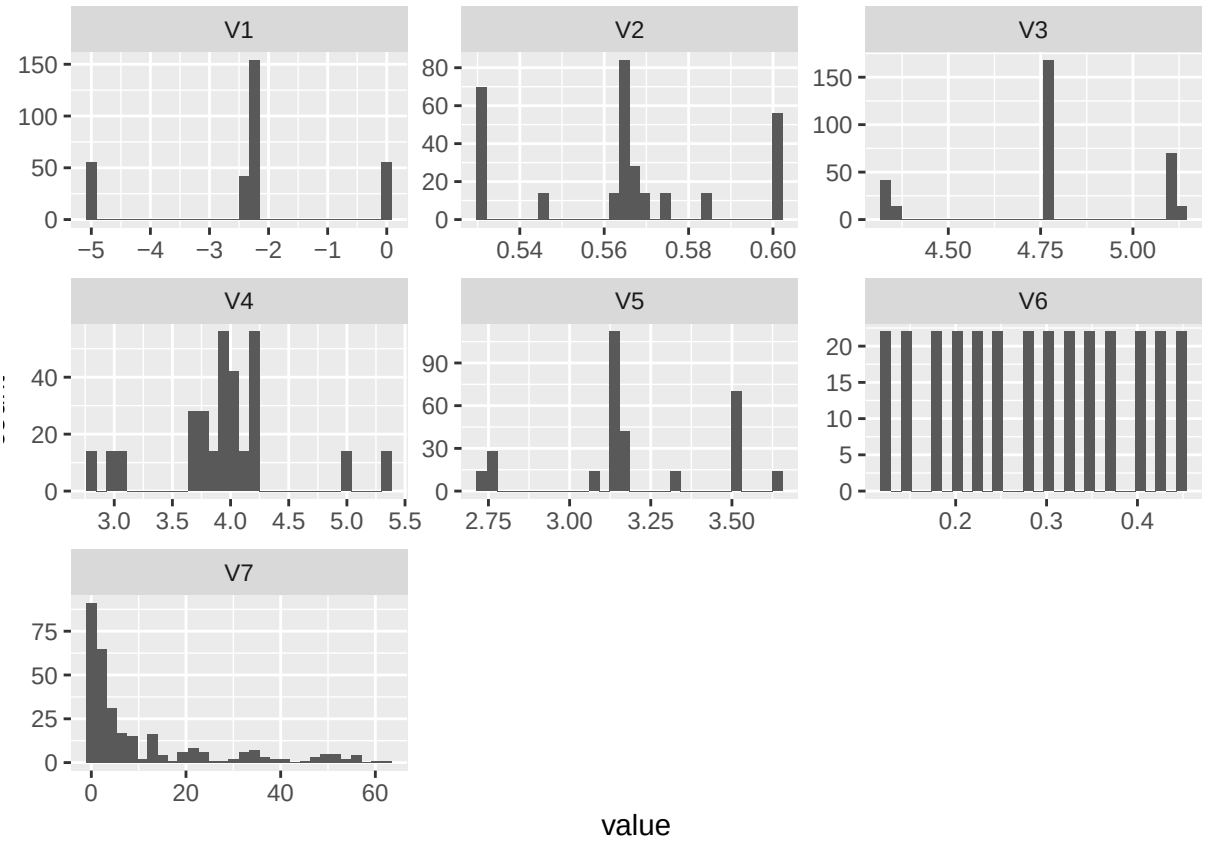
```
yachts <- read.table(yacht_hydrodynamics_path)
head(yachts)
```

```
##     V1    V2   V3   V4   V5    V6   V7
## 1 -2.3 0.568 4.78 3.99 3.17 0.125 0.11
## 2 -2.3 0.568 4.78 3.99 3.17 0.150 0.27
## 3 -2.3 0.568 4.78 3.99 3.17 0.175 0.47
## 4 -2.3 0.568 4.78 3.99 3.17 0.200 0.78
## 5 -2.3 0.568 4.78 3.99 3.17 0.225 1.18
## 6 -2.3 0.568 4.78 3.99 3.17 0.250 1.82
```

Firstly, lets load the dataset and analyse some basic statistics. Yacht Hydrodynamics dataset has 308 rows and 7 columns, with names: V1, V2, V3, V4, V5, V6, V7. Lets plot count of each unique value for every feature:

```
yachts %>%
    gather(key='var', value='value') %>%
    ggplot(aes(x = value)) + geom_histogram() + facet_wrap(~ var, scales = 'free')
```

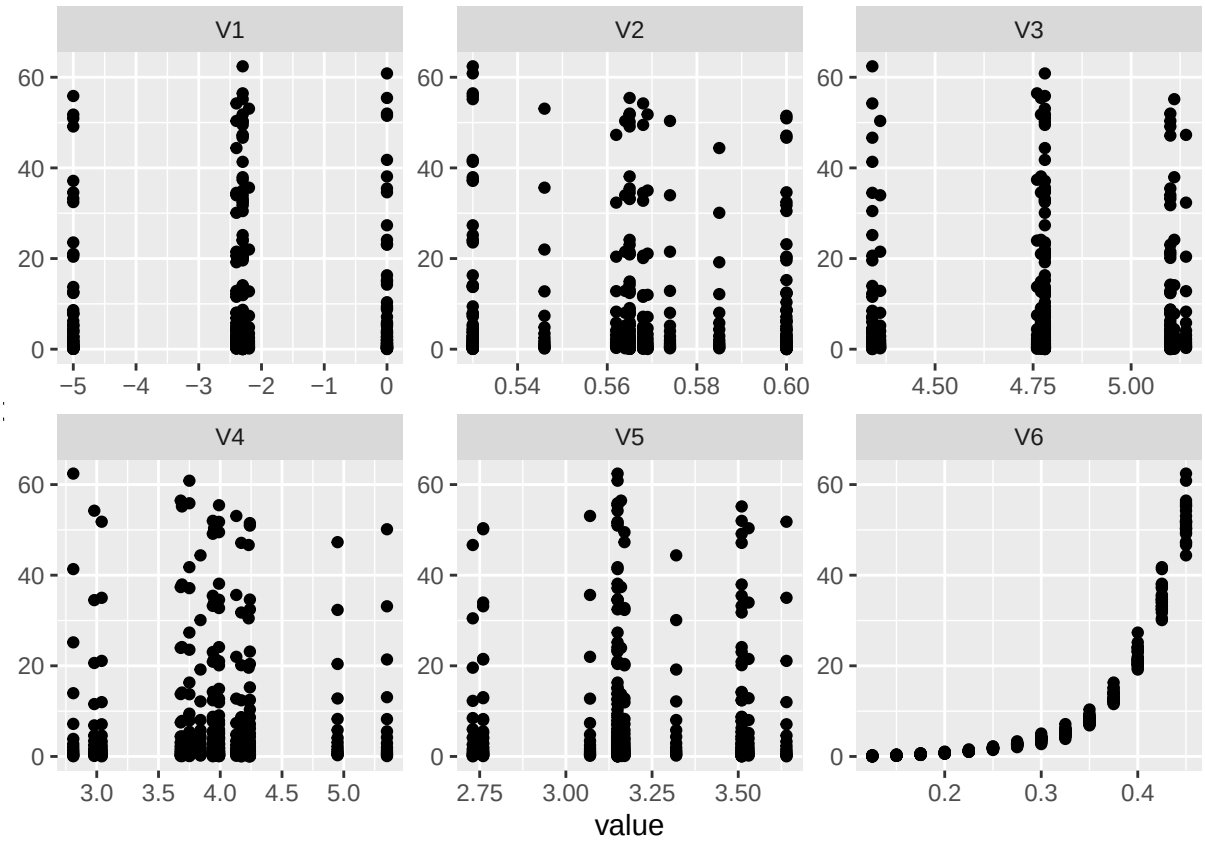
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Variables V1, . . . , V5 don't seem to have any particular regularities. They all seem to be centered around their mean with several outliers equally distributed at a given distance from the mean. However, V6 has 14 unique values, where each appears exactly 22 times. The only possible values for V6 belong to a sequence: 0.125, 0.15, . . . , 0.45. Lastly, V7 seems to have most values in the interval [0, 10], with several others going up to 62.42.

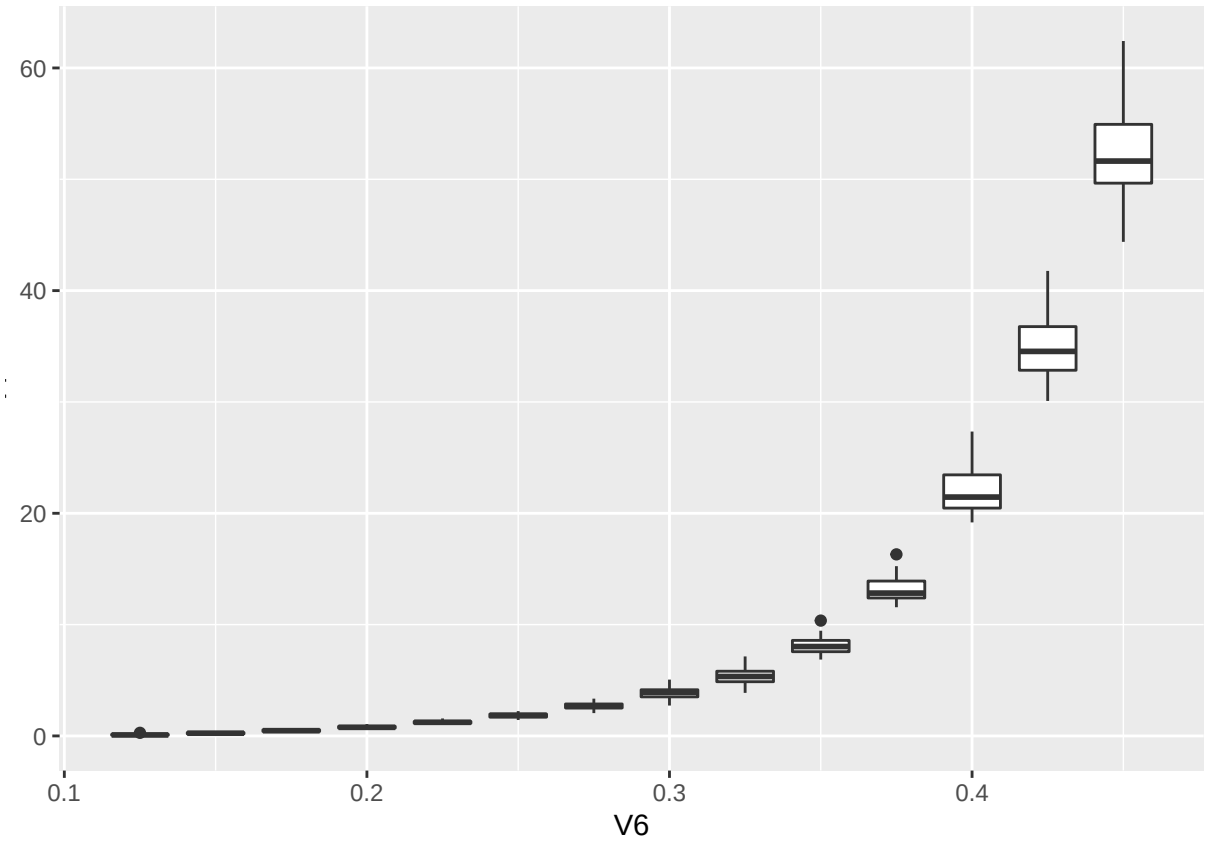
Now, let's see if there is any correlation between each individual feature and V7:

```
yachts %>%
    gather(-V7, key='var', value='value') %>%
    ggplot(aes(x = value, y = V7)) + geom_point() + facet_wrap(~ var, scales = 'free')
```



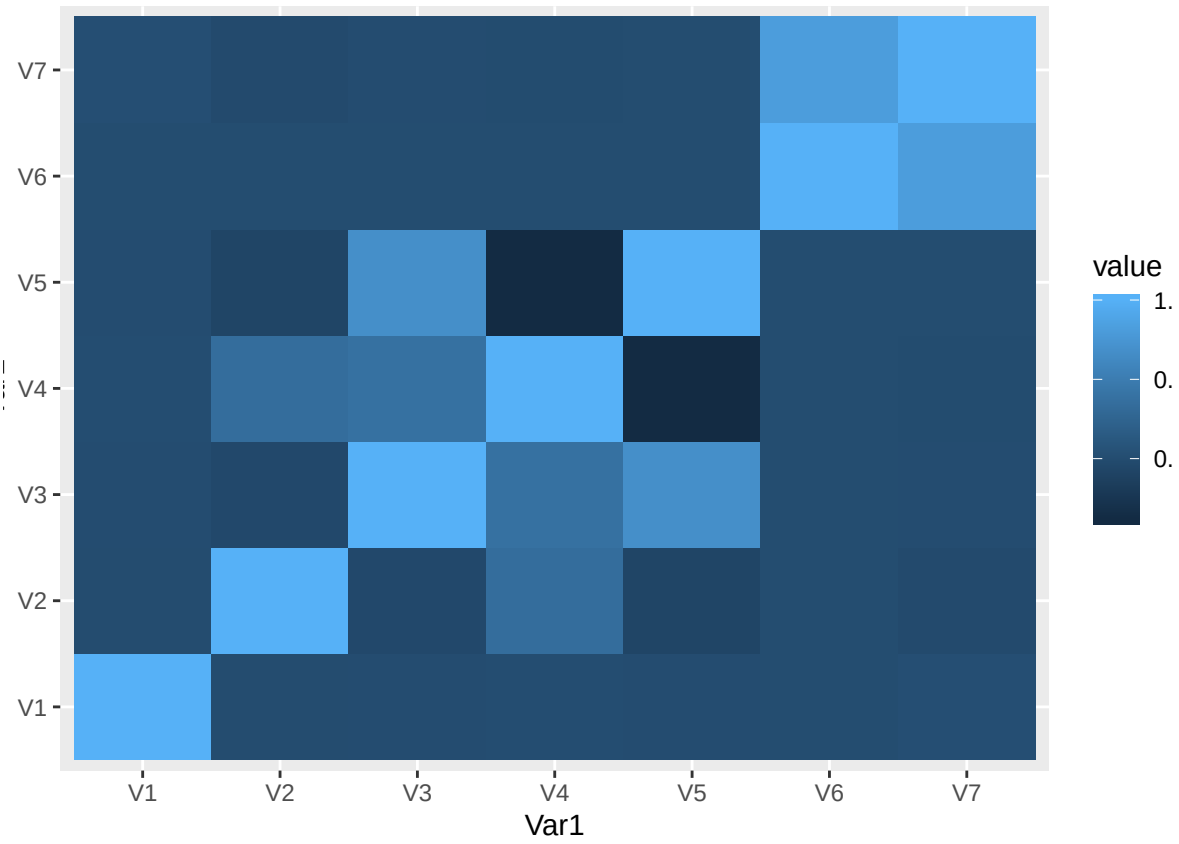
From these plots, we can clearly see that the last plot discovers a high correlation between values of V6 and V7, which resembles an exponential function. Linear growth in V6 is reflected by an exponential growth of the values of V7. This can be also visualised by drawing a boxplot for each unique value of V6:

```
ggplot(yachts, aes(x = V6, y = V7)) + geom_boxplot(aes(group=cut_number(V6, 14)))
```



Lastly, a correlation heatmap between variables once more reveals that the biggest correlation is between V6 and V7.

```
yachts %>%
    cor() %>%
    melt() %>%
    ggplot(aes(x = Var1, y = Var2, fill = value)) + geom_tile()
```



Model training

Now, an SVM regressor will be trained on the dataset. Firstly, the dataset will be split into a training (80%) and a testing one (20%). Then, the training dataset will be used for hyper-parameter tuning using a 10-fold cross-validation. This way of splitting dataset will make sure that the model with tuned hyper-parameters is independent of the testing set. Finally, model performance will be reported on the testing set with the help of R^2 and $RMSE$ metrics.

```
train_indices <- yachts$V7 %>% createDataPartition(p = 0.8, list = FALSE)
yachts_train <- yachts[train_indices, ]
yachts_test <- yachts[-train_indices, ]
```

Define a function which predicts values of V7 for a test set and compares them to actual values.

```
plot_regression <- function(model) {
    y_hat <- predict(model, yachts_test)
    ggplot(mapping = aes(x = yachts_test$V7, y_hat)) + geom_point() + geom_abline(color='red')
}
```

Define a function to train svm regressors using 10-fold crossvalidation and return mean of all R^2 and $RMSE$.

```
svm_regression_r2 <- function(kernel, c, epsilon) {

    k = 10
    folds <- cut(seq(1, nrow(yachts_train)), breaks=k, labels=FALSE)
    r2s <- rep(0, k)
    rmses <- rep(0, k)
```

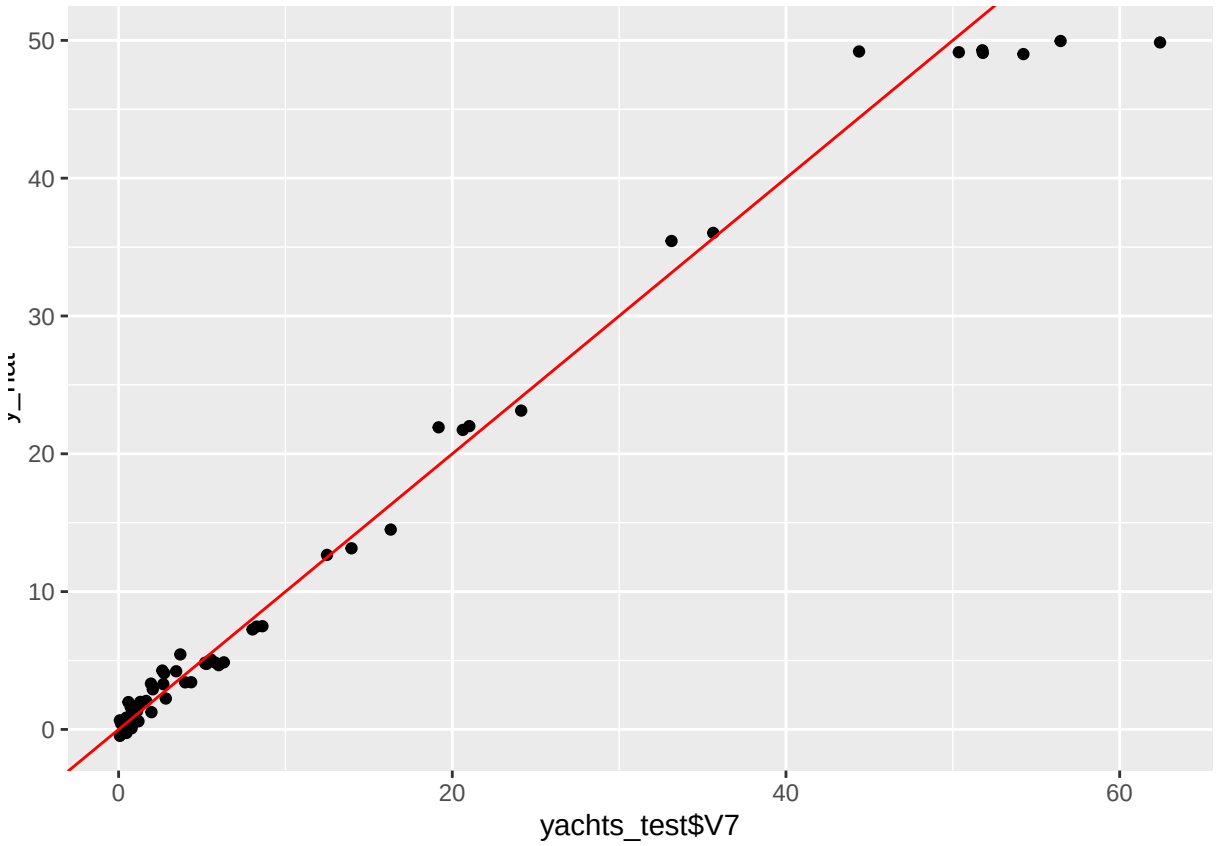
```
    for(i in 1:10) {
        test_indices <- which(folds == i, arr.ind = TRUE)
        test_dataset <- yachts_train[test_indices, ]
        train_dataset <- yachts_train[-test_indices, ]

        model <- ksvm(V7 ~ ., train_dataset, type = 'eps-svr', kernel = kernel, C = c, epsilon = epsilo
        predicted <- predict(model, test_dataset)
        r2s[i] <- R2(predicted, test_dataset$V7)
        rmses[i] <- RMSE(predicted, test_dataset$V7)
    }

    c(r2 = mean(r2s), rmse = mean(rmses))
}
```

Now grid search will be used to find model with such hyperparameters, that R^2 and $RMSE$ are highest. The choice of kernels is based on manual experimentation - all kernels from *kernlab* package were tested and 5 best ones were chosen for hyperparameter tuning with different values of C and *epsilon*.

```
grid_search <- list(
    kernel = c(rbfdot(), anovadot(), polydot(), polydot(degree = 2), polydot(degree = 3)),
    c = seq(0.5, 1.5, 0.25),
    epsilon = c(0.01, 0.05, 0.1, 0.20, 0.40)
) %>%
    cross_df() %>%
    mutate(metrics = pmap(., svm_regression_r2)) %>%
    unnest_wider(metrics)
```

Now we can visualize the performance of different parameter configurations, depending on whether we want to focus on highest value of R^2, lowest $RMSE$ or try to pick a model which has good enough values of both metrics. Lets find a model with lowest $RMSE$ and then with highest R^2. Then, it will be trained on the whole training dataset and its performance will be shown on the test dataset.

```
best_row <- grid_search %>% arrange(rmse, -r2) %>% filter(row_number() == 1)
model <- ksvm(V7 ~ ., yachts_train, type = 'eps-svr', kernel = anovadot(), C = best_row$c, epsilon = bes
plot_regression(model)
```

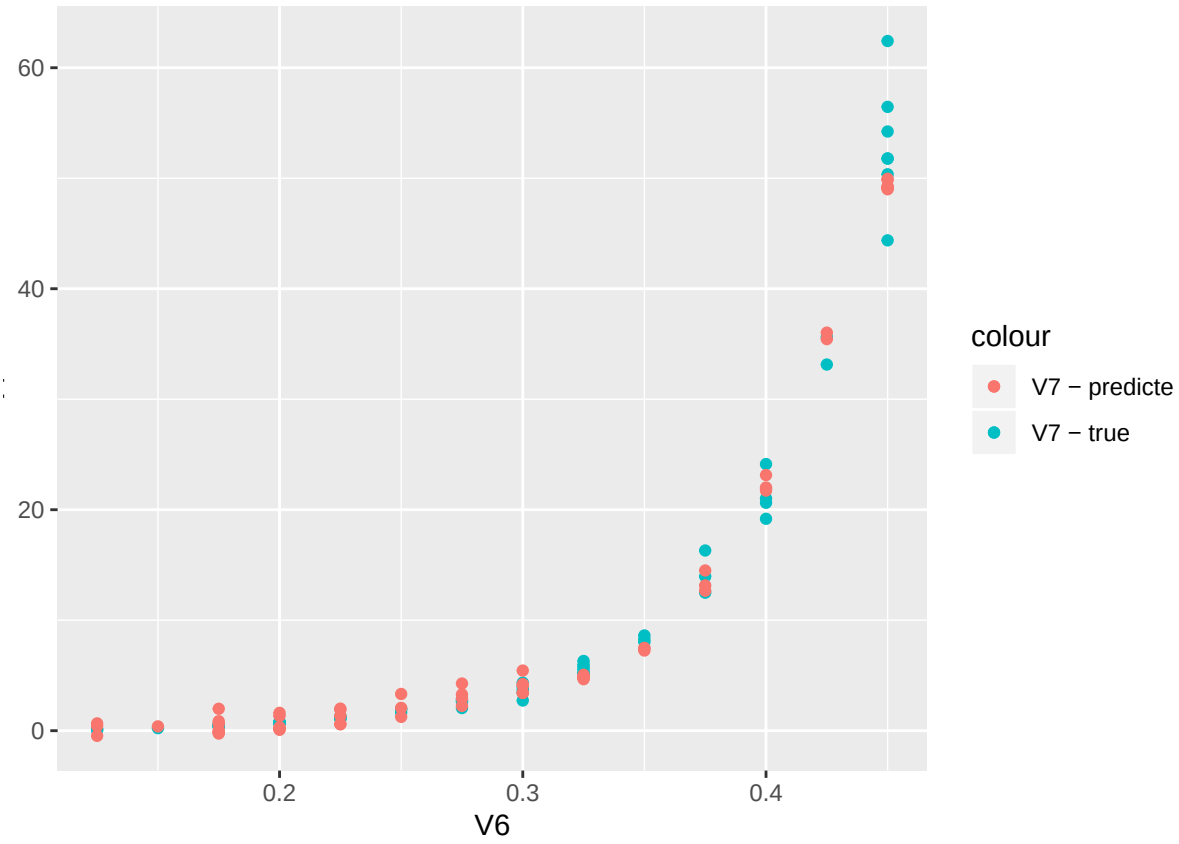


The plot presents actual values of V7 on the x axis and predicted values on the y axis. Perfect model would produce dots only on the red line. We can see that chosen model is a rather good fit for the data. Following hyperparametrs were used: kernel = *anovadot*, C = 1.5, epsilon = 0.05, and the obtained metrics are: $R^2 = 0.9936$, $RMSE = 1.5785$. High value of R^2 tells that the model is able to explain most of the variance in V7, and low value of $RMSE$ shows that in general the difference between actual and predicted value is small.

Feature selection

We can see the correlation between predicted values of V7, actual values of V7 and actual values of V6. It's visible that that the model relies mostly on the values of V6 - for every observation with given value of V6 the model seems to give nearly the same answer. This suggests that other variables aren't useful at all and could be safely removed from the model. Experimental verification using other feature selection methods (AIC, BIC) lead to the same conclusion.

```
predicted <- predict(model, yachts_test)
yachts_test %>%
    ggplot(aes(V6)) +
    geom_point(aes(y = V7, colour = 'V7 - true')) +
    geom_point(aes(y = predicted, colour = 'V7 - predicted'))
```



Finally, let's train the model only on the V6 variable and compare its performance.

```
model <- ksvm(V7 ~ V6, yachts_train, type = 'eps-svr', kernel = anovadot(), C = best_row$c, epsilon = b
predicted <- predict(model, yachts_test)
r2 <- R2(predicted, yachts_test$V7)
rmse <- RMSE(predicted, yachts_test$V7)
```

Using only V6: $R^2 = 0.9853$, $RMSE = 2.3583$, whereas using every feature: $R^2 = 0.9936$, $RMSE = 1.5785$. Feature selection helped to increase R^2 by -0.0083 and decrease $RMSE$ by 0.7797. This concludes that amongst all features, V6 is the only important one for the model.