


[Forums / Assignments](#)[Help](#)

# Course project - Pre processing, feature extraction and PCA

[Subscribe for email updates.](#)[UNRESOLVED](#) No tags yet. [+ Add Tag](#)Sort replies by: [Oldest first](#) [Newest first](#) [Most popular](#)[Christian Grant](#) · 14 days ago 

## Feature extraction and selection using PCA

I'm considering only focusing on the following 17 features ,that the authors of the paper identified as most relevant. See section 5.1 in

<http://groupware.les.inf.puc-rio.br/public/papers/2013.Velloso.QAR-WLE.pdf>

They used a feature selection algorithm proposed by:

M.A. Hall Correlation-based Feature Subset Selection for Machine Learning, Department of CS, New Zealand, 1999

### Location Feature

- |    |          |                                      |
|----|----------|--------------------------------------|
| 1  | belt     | mean of the roll                     |
| 2  | belt     | variance of the roll                 |
| 3  | belt     | maximum                              |
| 4  | belt     | range of the accelerometer vector    |
| 5  | belt     | variance of the accelerometer vector |
| 6  | belt     | variance of the gyro                 |
| 7  | belt     | variance of the magnetometer         |
| 8  | arm      | variance of the accelerometer vector |
| 9  | arm      | maximum of the magnetometer          |
| 10 | arm      | minimum of the magnetometer          |
| 11 | dumbbell | maximum of the acceleration          |
| 12 | dumbbell | variance of the gyro                 |

- 13 dumbbell maximum of the magnetometer
- 14 dumbbell minimum of the magnetometer
- 15 glove sum of the pitch
- 16 glove maximum of the gyro
- 17 glove minimum of the gyro

Should we try to extract the same 17 features using a feature extraction algorithm such as PCA?  
The code in the lecture for doing this is:

```
M <- abs(cor(training[, -160]))  
diag(M) <- 0  
which(M > 0.8, arr.ind=T)
```

Finding the correlation for 159 variables in a large data set could take some time on a lap top.

Has anyone tried? How much time will it take? Is it a necessary step?

If we do PCA, there's some work in terms of pre processing the data set (converting factor to numeric, ignoring some columns such as name etc.)

Or should we assume that the 17 features are given, and only focus on the subsequent training model, prediction, and accuracy?

Any comments?

The course project goal could be clearer!

"The goal of your project is to predict the manner in which they did the exercise."

Does this mean we should attempt to replicate the used approach?

Or should we simply try to predict "classe" based on our own selection of predictors, and algorithms?

## Update

I ended up using 52 predictor variables (see the following comments), and I have interpreted the ultimate goal as being able to predict the test cases correctly.

↑ 0 ↓ · flag

Anonymous · 14 days ago

As a manual step I cleaned the original data set by removing the columns filled with NA's. This reduced the data set by a third.

The following took less an a second on an old laptop:

```
M <- abs(cor(training[, -53]))
diag(M) <- 0
which(M > 0.8, arr.ind=T)
```



Using the built in R functions, the following also takes less than a second:

```
analysis = prcomp(training_set[, -53], scale. = TRUE);
```



```
variance_vector = analysis$sdev ^ 2;
```



```
variance_vector
```

```
relative_variance <- variance_vector / sum(variance_vector);
```



```
cumsum(relative_variance);
```



We only need ~37 features to capture 99% of the variance.

↑ 8 ↓ · flag

umair · 14 days ago 🔗

Can we use varImp function to figure out important features just like we did in last question of quiz 3?

↑ 0 ↓ · flag

Jae Lim · 7 days ago 🔗

I think capturing 99% variance is too much, because this potentially introduces overfitting later on.

↑ 0 ↓ · flag

Juraj Vyskočil Signature Track · 7 days ago 🔗

> I think capturing 99% variance is too much, because this potentially introduces overfitting later on.

I do not think so, overfitting is different issue. Capturing the variance is just a measure of compressing the data - how much we get rid of some data. It has nothing to do with overfitting, because then we would get overfitting each time we train our model on non-reduced original data set (= 100% variance).

And still you can capture 99% variance, and get very poor accuracy on training data (so certainly not overfitted model).

↑ 0 ↓ · flag

Jae Lim · 7 days ago 🔒

On contrary, I think it is in this case. The PCA is generally a feature selection method to reduce the number (high dimensionality) of explanatory variables (correlated predictors) to a few uncorrelated variables which still explains the data. If you let 100% variance in data for this case with a large number of predictors, you're no way to avoid overfitting the model. In short, 99% is 100% and it's just waste doing PCA here.

↑ -3 ↓ · flag

Juraj Vyskočil Signature Track · 7 days ago 🔒

PCA does not mean the feature selection, PCA is a method of creating NEW (lower dimensional) predictors with capturing as much variance as possible. So it does not mean discarding less-useful predictors, but creating the new ones in lesser count. At least AFAIK. And in addition, these new predictors are much more difficult to interpret.

Maybe you referred to the right singular vector analysis, by which we can determine predictors with maximum variance contribution.

In any case, I agree that PCA is just wasting here. If we want to avoid the overfitting, we can do so by implementing some kind of regularization of the criteria function, not by reducing the data dimension. But I haven't heard this so far in the class lectures yet.

↑ 5 ↓ · flag

---

[+ Comment](#)

Noah Peart · 14 days ago 🔒

Ill just add that you don't need to use 19000 rows. I got 90% accuracy with just 3000 rows. This will speed up your prediction considerably.

↑ 3 ↓ · flag

umair · 14 days ago

I selected 28 features and 3000 rows on training data,  
Then I tried to fit the model using modFit4 <-  
train(classe~.,data=mytraining\_data,method="rf",prox=TRUE)  
But I am getting following error  
final tuning parameters could not be determined, Please suggest

↑ 2 ↓ · flag

Noah Peart · 13 days ago

you are most likely choosing a subset that contains only one level of classe. Make sure that your subset has equal samples of classe variable and it should work, i.e.

```
trainInds <- sample(nrow(training), 3000)
train <- training[trainInds,]
```

↑ 2 ↓ · flag

umair · 13 days ago

Yes, You are right, Thanks.  
Is it necessary to normalize the selected features

↑ 0 ↓ · flag

Noah Peart · 13 days ago

Depends what you are doing. For PCA, you want to at least center the data, and probably normalize if the variables have a wide range of variation (unless you don't want to equally weight your variables). In general, if the algorithm you are using is sensitive to different variations in your variables you may want to normalize. You can always try your prediction both ways and just see which way works better.

↑ 1 ↓ · flag

Anonymous · 13 hours ago

Thanks Noah for that tip, saved me a big headache

0 · flag

[+ Comment](#)

Christian Grant · 13 days ago

First I removed columns with many na's or many empty spaces, timestamp, X, user\_name and the new\_window column.

Is there an easy way to reduce the dataset from 160 variables to 54 variables?

Result: 54 columns.

Then I tried 2 random forests with 3436 samples

1. PCA with 37 components ~ 0.99 of the variance

Training Result: 89% accuracy

2. 53 predictors

Training Result: 98% accuracy

**Update:** 52 predictor, 99% accuracy.

1 · flag



Tatiana Shingel Moody Signature Track · 13 days ago

Hi Christian,

I have tried pretty much the same pre-processing as you. How did you come up with 3436 samples? My problem is, if I use all 19000 rows even with a few principal components, it takes forever to run on my machine (1.8GHz, 4GB RAM, 64-bit operating system). How long did it take you to train your model? I would very much appreciate your input.

Thanks,

0 · flag

Noah Peart · 13 days ago

Random forest will take a while, try a support vector machine or some variation of logistic

regression which should only take a minute or so for a few thousand rows and give decent predictions. Neural nets also take a while if you are using a bunch of predictors.

↑ 0 ↓ · flag

[umair](#) · 13 days ago 🔗

I used around 4000 observations with 28 variables. It took almost 1 hour to train random forest. My machine is Intel i3 with 4 GB Ramm windows 7 64 bit

↑ 0 ↓ · flag



[Tatiana Shingel Moody](#) Signature Track · 12 days ago 🔗

One hour is not so bad. I waited for a couple of hours and just quit. Then I reduced the number of samples. It ran quickly, but the accuracy of prediction was low. Support vector machines also take a long time. It's impractical if you want to tune parameters, for example. Has anyone tried a "parallel" package which is supposed to enable multi-core processing in R on a windows machine?

↑ 2 ↓ · flag

[Noah Peart](#) · 12 days ago 🔗

some of the methods will use parallel processing by default. The boosted trees (gmb) uses multiple cores on my machine by default when called through caret.

↑ 1 ↓ · flag

[Juraj Vyskočil](#) Signature Track · 12 days ago 🔗

**Christian, how did you get such accuracy? I keep getting poor accuracy on training set and can't find out why:**

Here's my try:

```
# load data
trainRawData <- read.csv("pml-training.csv",na.strings=c("NA",""))
# discard NAs
NAs <- apply(trainRawData,2,function(x) {sum(is.na(x))})
validData <- trainRawData[,which(NAs == 0)]
# make training set
```

```

trainIndex <- createDataPartition(y = validData$classe, p=0.2,list=FALSE) # 39
27 rows
trainData <- validData[trainIndex,]
# discards unuseful predictors
removeIndex <- grep("timestamp|X|user_name|new_window",names(trainData))
trainData <- trainData[,-removeIndex]
# make model
modFit <- train(trainData$classe ~.,data = trainData,method="rpart")
modFit

```

And here is the outcome with really poor 54% accuracy on train set.

CART

3927 samples

53 predictors

5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing

Resampling: Bootstrapped (25 reps)

Summary of sample sizes: 3927, 3927, 3927, 3927, 3927, 3927, ...

Resampling results across tuning parameters:

cp	Accuracy	Kappa	Accuracy SD	Kappa SD
0.0358	0.541	0.407	0.037	0.0523
0.0621	0.371	0.131	0.0287	0.0442
0.114	0.342	0.0868	0.0356	0.0557

Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was cp = 0.0358.

I have not tried a random forest yet, but I suppose that basic decision tree should not give so low accuracy.

UPDATE:

After almost hour of random forest computing I got 97.4% accuracy, but I am astonished at how big the difference is.



↑ 4 ↓ · flag

Christian Grant · 12 days ago

I tried rpart. It's amazingly fast. I only got 51.3 % accuracy. Try using random forest (rf) .

↑ 1 ↓ · flag

Juraj Vyskočil Signature Track · 12 days ago

Yes, I've just tried it and with good result now, but I was suprised by the change in accuracy in respect of the basic rpart method, almost as if the basic decision tree was really unuseful until adding some boosting method.

Anyway, do we know some basic or general guidelines when analysing data with such many dimensions? We were just lucky here to get the easy reducible dataset with almost 2/3 empty space in it.

What do we know so far - we can do SVM analysis and identify the predictors with the most variance contribution... or we can do some hierarchical clustering to identify which classes are hard to identify.

Anything next?

↑ 0 ↓ · flag

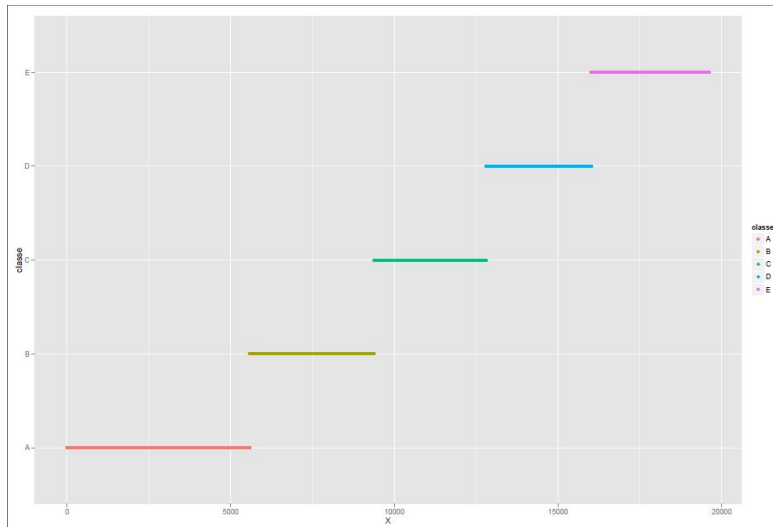
Christian Grant · 10 days ago

Since the study is a HAR (Human activity recognition study) using sensors, it makes sense to remove all variables that are not sensor data (6-7 variables). We were lucky that many variables contained NA's or empty spaces. Otherwise, we will have to wait until Apple launches a 16 core MacBook Pro, with built in parallel clustering capabilities :-)

↑ 3 ↓ · flag

John Poppelaars · 7 days ago

Hi Christian, Why did you leave out X? if you make a plot of X and Classe, you will see that there is a relation between X and Classe. See the attached plot



↑ 0 ↓ · flag

Navtej Singh Khandpur · 4 days ago

X is just a sequence number, no?

↑ 2 ↓ · flag

Christian Grant · 3 days ago

That's how I understand it.

↑ 0 ↓ · flag

John Poppelaars · 3 days ago

Yep, you're right

↑ 0 ↓ · flag

[+ Comment](#)

Christian Grant · 12 days ago

The random forest method with 3436 samples and 53 predictors took less than an hour on a MacBook Pro - 2010 (2.6 GHZ, 4 GB ram, 64 bit, Mac os x 10.9.3). I will time it more precisely next time. I spliced the training set into a training set 70% and a cross validation set 30%. For the training test run I used 25% of the new training set ( $100\% \times 70\% \times 25\% = 3436$  samples). A test set is not generated, since it's provided.

↑ 0 ↓ · flag

[michael kowalski](#) · 12 days ago 


My MacBook Air mid-2010 took 15 minutes to do the random forest using caret as described above. That was with parallel processing enabled - so 2 cores.

↑ 1 ↓ · flag

[Christian Grant](#) · 12 days ago 

My MacBook Pro mid-2010 with 4 gb took 20 minutes using random forest on 3436 samples.

↑ 0 ↓ · flag

[Alberto Odor](#) Signature Track · 2 days ago 

How do you select the 25% you use from the training set?

↑ 0 ↓ · flag



[Kevin Davenport](#) · a day ago 

Took less than 3 minutes on last years 8gb macbook pro Retina using Python.

↑ 0 ↓ · flag

---

[+ Comment](#)

[Christian Grant](#) · 12 days ago 

Has anyone tried using R in a Mac or Windows parallel cluster?

↑ 0 ↓ · flag

---

[+ Comment](#)

[Christian Grant](#) · 12 days ago 

How do we best calculate the out of sample error?

**Approach:**

1. I use predict to predict the values for classe in the cross validation set
2. I identified how many of the predictions match the actual values  
`predictedValue == crossValidateDataset$classe`
3. I calculated the the out of sample error using the following

$1 - (\text{sum}(\text{predictedValue} == \text{ActualValue}) / \text{"cross validate data set sample size"})$

**Result:**

In sample error: 0

Out of sample error: 0.006795787 ~ 0.68 %

method: rf

accuracy: 0.988

duration: 43 minutes

trainingSampleSize: 6869

crossValidationSample: 2943

Is there a function that makes this process easier?

I can't use mean squared error - MSE or RMSE since the variables are factor variables, not continuous variables.

Should we use a confusion matrix?

How is it calculated for the cross validation set?

↑ 3 ↓ · flag

michael kowalski · 11 days ago

Caret also has model evaluation functions. The ones I find most useful are `postResample()` and `confusionMatrix()`.

They are listed in the [caret documentation](#) under 'other functions' with examples.

↑ 4 ↓ · flag

---

+ Comment

Amirhossein Meisami · 12 days ago

`confusionMatrix$overall[1]` will give you the accuracy

↑ 2 ↓ · flag

---

+ Comment

Amirhossein Meisami · 12 days ago

Guys,

I am trying to do PCA on 4 different sets of features which I assume they may have some sort of similarities. I partitioned my training data into test and train data sets. After PCA my rf model gives 99% accuracy on test data partition using 30 principal components and 4 of the predictors. However, when I apply this model on the original test set given in assignment section, I get a result which is 40% different from the result of my rf method with 53 predictors (99% accuracy). Both rf methods are 99 percent accurate on the partitioned training dataset but the result on the test data is noticeably different. Any thoughts?!

↑ 0 ↓ · flag

umair · 12 days ago

My model gives 95% accuracy. When I tried it on 20 problems. I got only 11 correct answers.

↑ 0 ↓ · flag

Anonymous · 6 hours ago

testing file (20 samples) do not have the variable "classe"... maybe that's the error

↑ 0 ↓ · flag

---

[+ Comment](#)

Christian Grant · 12 days ago

I just submitted the 20 predictions using the test data set. They were all correct.

Training duration: 1.921951 hours (MacBook Pro - mid 2010, 2.6 GHZ, 4 GB ram, 64 bit, Mac OS X 10.9.3)

Model accuracy: 0.9954982

Predictors: 53

Training sample size: 13737

Method: "rf"

### Update

I reduced the # of predictors to 52.

I removed num\_windows as a predictor for various reasons, since it's not a sensor measurement.

Here are some of my reasons for removing predictors:

- When the variable doesn't appear to be a sensor reading. The goal of HAR (human activity

recognition) studies is to predict activities using sensors.

- When it's unclear how a variable is measured, and when a variable is related to the sequence of the experiments. Examples are new\_window, and num\_windows. If the experiments for the "classe" categories are not in a random sequence, it's no surprise that only using the num\_windows variable results in 100% accuracy.
- When a predictor is 100% correlated to what it's supposed to predict. It seems suspicious when one predictor can predict the result with 100% accuracy.
- A person's id/name is not a sensor measurement.
- Time should not impact the weight lifting activities, unless fatigue at late hours is an issue.

↑ 3 ↓ · flag



Simon Flannery Signature Track · 11 days ago 🔒

If you pass in a train control object to the training function, you should be able to decrease your training times to less than 10 minutes and still maintain the same accuracy. By default, the training model is using bootstrapping which is expensive. Try passing in, for example: `trControl = trainControl(method = "cv", number = 4)`.

↑ 14 ↓ · flag

TIMOLEON VAIDIS · 10 days ago 🔒

I followed your suggestion on `trControl`. I got a random forest model trained in just 7.72 mins. I also used `"allowParallel=T"`. It ran on a quad-core windows 7 desktop (i5, 8GB RAM). I got 99.6% accuracy on the cross validation set and 20/20 correct predictions on the test cases. I used 53 predictors and a training set of ~13800 rows.

However, the CPU Usage did not exceed 25% noticeably. I am wondering if this is due to the algorithm not being working in parallel mode except for a fraction of the time, or due to lack of some other configuration that I should have applied (to R?, system?). When I was running parallel in MATLAB I always observed usages close to 100%. Any thoughts on that point?

Thank you Simon:)

Tim

↑ 3 ↓ · flag

Christian Grant · 10 days ago 🔒

Thanks Simon. That's remarkable. I was able to reduce the time from 2.034711 hours to 13.74775 minutes, while increasing the accuracy from 0.981 to 0.991.

↑ 2 ↓ · flag

Jim Maas · 4 days ago

Thanks to all who contributed, has helped me a lot. Thus I thought I would share some info re timing and parallelization. I use R in parallel on a 500-core cluster for quite of bit of work so am moderately familiar with it, and highly recommend it. If you develop on your pc in parallel, you can then upload to a cluster, use lots of cores and get the job done in a small fraction of the time..

My computer is ~ four years old, dual core PC with hyperthreading, so it thinks it has four cores, and I run Linux. I use the "doMC" package on a single pc to run packages in parallel. When I go to the cluster I use "doMPI".

I just trained a random forest model, using 70% of the total training data set, i.e. 13737 records (rows) of 45 variables (columns) at it took ~ 44 minutes to complete. Incidentally the results got 99.6 % accuracy by using only 27 of the 45 variables. This same model then predicted 20/20 correctly for the final results.

I note several people have correctly suggested that you could get very high accuracy with a much smaller subset of the training data, and this is a logical step. However if you have the computational resources, and it doesn't take too much longer, or you can run it in batch in the background, while drinking coffee and/or gazing out the window, then why not?

Thanks for all the help. :-)

J

↑ 2 ↓ · flag

[+ Comment](#)

Amirhossein Meisami · 11 days ago

The 20 predictions can be made correctly just by 1 or 2 (num\_window) predictors in a 2-minute run with 0.99 accuracy. My problem is how to do a little fancier job by applying PCA on 4 different groups of predictors (i.e. arms, forearms etc.) to get fewer variables in the final model with a reasonable accuracy.

↑ 1 ↓

· flag

Christian Grant · 11 days ago

Amazing, which method are you using?

↑ 0 ↓ · flag

Amirhossein Meisami · 11 days ago

I am using "rf". There is a problem with this dataset. I believe num\_window is enough to accurately predict the outcome! I am still having issue in applying PCA correctly! it works well on the training data (which I partitioned into train and test sets), but on the original test data it is roughly 0.7 accurate!

↑ 0 ↓ · flag

Christian Grant · 11 days ago

My accuracy for num\_window as a predictor is 100%

Did anyone find out what num\_window is, and how it's measured?

Is num\_window a valid predictor, or is it a design flaw?

It could be related to the the sequence of experiments. If the sequence of experiments was arranged in an order that reflects classe and not randomly, then it's no surprise that the accuracy is 100%.

↑ 1 ↓ · flag

michael kowalski · 11 days ago

The link between all the time variables and the classe is discussed [here](#).

We are dealing with time series data with samples at intervals shorter than 1 second.

Whichever exercise version ( the classe ) was being performed in each row of our data was also very likely to be the same version in the period before and after. 'num\_window' appears to be a literal count of time windows over which summary data is gathered by the device.

↑ 0 ↓ · flag

Peter Mooshammer · 6 days ago

I used 52 variables on a reduced set of rows (using createFolds) and Random Forest which



results in a high accuracy. I used another set of the training data (a second fold) with the confusion Matrix, which confirms the high accuracy. But the result of prediction with the testing data makes me wonder, if I did something wrong. (Note I haven't tried to submit them yet ...)

↑ 0 ↓ · flag

[+ Comment](#)

Jay Rosen · 11 days ago 🔗

Hey there! I have a quick question and would be very appreciative if someone can point me in the right direction.

I have preprocessed my data using principal components (after eliminating the columns filled with missing/blank data). I've then "trained" a model using "random forest" as the method and it says I achieve approx 97% accuracy. Suppose my fitted model is contained in the variable called "modelFit." When I run 'modelFit\$coefnames' I see results which begin as

```
[1] "PC1" "PC2" "PC3" "PC4" "PC5" ...
```

...which may be my problem. I believe these should be the column names of my 'training' dataset? If so, can someone suggest what I may be doing incorrectly?

When I try to run

`pred_rf <- predict(modelFit, testing)`, I get the following error message:

```
Error in eval(expr, envir, enclos) : object 'PC1' not found
```

I have verified the colnames of my training and testing datasets are the same. I even get the same error message when I try to run '`predict(modelFit, training)`' (i.e. using the same dataset I just used in training the model.)

I would appreciate any insights on how I can solve this problem.

Thanks!

↑ 0 ↓ · flag

Anonymous · 11 days ago

Make sure you also apply your pre-processing step to the testing data before using it in `predict(modelFit, testing)`

↑ 0 ↓ · flag



Ivan Ngeow · 11 days ago

You'll have to call 'predict' *twice* in your prediction step:

```
# training
preproc <- preProcess(training, method='pca', thresh=0.9)
training.pca <- predict(preproc, training[,-1])      # make a reduced tra
ining set (exclude column 1)
modelFit <- train(training[[1]] ~ ., data=training.pca, method='rf')

# predicting
testing.pca <- predict(preproc, testing)           # make a reduced testing s
et
predictions <- predict(modelFit, testing.pca)
```

↑ 4 ↓ · flag

Jay Rosen · 11 days ago

Thanks, Ivan!

↑ 0 ↓ · flag

[+ Comment](#)



Alexander Kucherenko · 10 days ago

This is my approach. I removed columns with NA's, highly correlated columns, factor variables and timestamp columns and used only 49 numeric variables to build a predictive model. I used neither PCA nor scaling. The big dataset was divided into train set (70%) and validation set (30%). Accuracies of my Random Forest and SVM models on the validation set are about 0.99 (I tweaked sigma and cost of constraints violation to achieve such accuracy of SVM). Predictions from test set were 100% correct.

↑ 2 ↓

[· flag](#)[+ Comment](#)[Joseph Levy](#) · 9 days ago 

95% accuracy using random forest feels too good to be true. Worried about overfitting. Thoughts, anyone?

[↑ 0 ↓](#) · [flag](#)[Juraj Vyskočil](#) [Signature Track](#) · 9 days ago 

It does not seem to be overfitting when you get this accuracy for CV or test set.  
I'd rather believe that there are some predictors which shouldn't be there.  
Has anyone tried to discard num\_window variable?

[↑ 0 ↓](#) · [flag](#)[Christian Grant](#) · 9 days ago 

I discarded num\_windows. The predictions from the test set were 100% correct using 52 predictors. If I have time I will try to reduce the data set further using either PCA and/or the near zero variance function.

[↑ 1 ↓](#) · [flag](#)[michael kowalski](#) · 9 days ago 

yes, even without num\_window ( or any other time variables ) the accuracy is 98% for both training and validation sets. I don't think it is over-fitting there are just some strong predictors and a large amount of data to train an accurate model.

[↑ 1 ↓](#) · [flag](#)[Juraj Vyskočil](#) [Signature Track](#) · 9 days ago 

BTW in the [Machine Learning](#) course there was said that one should have adequate cause for reducing dimensions with PCA (e.g. when original dataset is too demanding for computing, or to visualize the data), otherwise we shouldn't add this step into the basic ML workflow automatically.

[↑ 1 ↓](#) · [flag](#)

Christian Grant · 9 days ago

In this course the professor said it's better to err on the side of having too many features.

↑ 1 ↓ · flag

Juraj Vyskočil Signature Track · 9 days ago

By the way, has anybody tried clustering on these data?

I supposed that when predictors were so strong, they should create well-recognizable clusters. I tried hierarchical clustering, but saw no clear cluster patterns.

↑ 0 ↓ · flag

---

[+ Comment](#)

Brandon Verkennes · 8 days ago

Quick question. If we remove part of the data from the training set, say new\_window, do we alter the test set as well? Or do we leave that unaltered when we run our prediction?

↑ 0 ↓ · flag

Christian Grant · 8 days ago

You have to remove the same columns from the test set as the ones you removed from the training and cross validation sets.

↑ 2 ↓ · flag

---

[+ Comment](#)

Daisy Wang Signature Track · 7 days ago

I hope someone can help me understand the data, since I cannot find codebook for it. I'm quite puzzled at the data structure. It seems that the derived features, e.g. mean, max, min, etc., are only populated when new\_window = "yes". Does it mean that those derived features are the summary statistics calculated from the same window\_num? If so, how could max\_yaw\_belt is less than yaw\_belt in raw form?

I was trying to remove the features with high missing, e.g. more than 95%, and found I ended up removing most of derived features, including the ones selected in authors' classification model in the paper, e.g.

```
sum(is.na(training$avg_roll_belt)/19622) #97.93%
```

I'm wondering if I read in the data incorrectly or the authors imputed the missing in order to keep the derived features in the model. Thanks.

↑ 1 ↓ · flag

[+ Comment](#)

[Stephen Evers](#) · 7 days ago 🔗

In my approach, admittedly targeted at getting the best score on the submission test set, I created 3 sets of data 60% training, 20% cross validation and 20% for out of sample error estimation. I removed all NA columns and did some feature selection based on columns provided in the final test set of 20 items. First round with PCA at .99 threshold got 95.7% Accuracy. I went back manually reduced the feature set to 7 features without PCA. Second round 99.8% (see below) and 20/20 on the submission test set. I'm not using num\_window, but I believe a number of the original columns can be used to almost perfectly predict the final submission set.

#### Overall Statistics

Accuracy : 0.9979

95% CI : (0.9959, 0.9991)

No Information Rate : 0.2848

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.9974

I don't think the test set in this assignment is representative of the real problem, the objective should be to predict A through E based on the accelerometer readings such that it is usable for more than just the 5 subjects in the test. The test set should provide a longer sequence of data as this is a time series ML problem and exclude columns which wouldn't be available or relevant in a real life application of the predictions. My approach although following all the guidelines in this class would not achieve an accurate score in a newly created test set (this would include using the num\_window column for you predictions).

↑ 0 ↓ · flag

[Tom Jackson](#) · 7 days ago 🔗

How did you use the 20% to calculate the out of sample error estimation?

↑ 0 ↓

· flag

Stephen Evers · 6 days ago

just use the confusionMatrix function as shown in week one of the class eg:  
confusionMatrix(crossValidation\$classe, predictions)

↑ 0 ↓ · flag

[+ Comment](#)

valerio orfano · 6 days ago

Hi guys i am quite confused about the out of sample error.

I understood that using the parameter `<<trainControl(method = "cv", number = 4)>>` the caret package automatically uses k-fold data slicing and therefore automatically uses cross validation. Am i right? So i jus have to pass the whole training data to the train function. Or as in my case a smaller training set as i have a 32bit windows OS, otherwise i get an out of memory error.

To calculate the error out of sample i could be using the function  
`missClass = function(values,prediction){sum((((prediction > 0.5)*1) != values)/length(values))}`  
passing test data and predictions and obtaining 100 % accuracy

Or the split between training and cross validation (70% and 30%) should be done manually? I think it is doable but not the best option?

Rgds valerio

↑ 0 ↓ · flag

Andreas Doersam · 6 days ago

As far as I understand it (and I might be totally wrong here...), the train method in the caret package does indeed use a k-fold cross validation strategy with your trControl parameter.

A random forest model (method = "rf") calculates the OOB value internally and stores it in the `<model-name>$finalModel$err.rate` variable. So for an random forest you could simply use the full dataset to train the model and retrieve the OOB value from the model.

For the other methods that I tried, I didn't find a corresponding OOB value for the final model, so I split the dataset into two parts, one for training and one for testing and OOB value

estimation. I also did this for the random forest and the OOB value from the model and my own value from predicting the test set were very similar.

↑ 0 ↓ · flag

[+ Comment](#)

Stuti Awasthi · 4 days ago

Hi all,

I am bit unclear about how Cross Validation is utilised. My doubt is, the Cross Validation breaks the data in different sample keeping one of the resampled dataset as validation set. Now once we build the model using `trainControl("cv")`, does it choses the best model automatically or does it apply same model to all the different folds to provide accuracy, and then its in the hands of the programmer to identify the best model and use that for prediction.

Please Suggest.

I have also submitted the project with 19/20 correct answers:

Training Data is cleaned, applied PCA and distributed in 75% : 25% ratio fro train and validation set.

While modelling used "RF" and also used `trainControl` with Cross Validation 3 Folds.

No of predictors : 36

Validation Accuracy : 97.8%

Thanks

Stuti Awasthi

↑ 0 ↓ · flag



Ivan Ngeow · 3 days ago

hi Stuti, it might help your understanding of train and `trainControl` to pass in these options to display additional progress messages:

```
tc <- trainControl(method = "repeatedcv", number = 3, repeats = 3, verboseIt
er=T, returnResamp='all')
fit <- train(..., do.trace=100, method='rf', trControl=tc)
```

You can also examine `fit$control$index` to see exactly which cases (from your dataset) have

been used in each CV iteration, and **fit\$**~~resample~~ for the parameters used and metrics for each iteration.

↑ 1 ↓ · flag

---

[+ Comment](#)

↓ scroll down for more ↓