Fit_Project_MachineLearning

Nancy

2024-08-30

1. Introduction.

With the birth of wearable devices such as Jawbone Up, Nike FuelBand, and Fitbit, collecting extensive data on personal activity has become increasingly popular. These devices are central to quantified self-movement, where individuals routinely track their data to enhance their health, identify behavioral patterns, or simply out of interest in technology. While many users focus on quantifying the frequency of their activities, they often need to pay more attention to the quality of their performance.

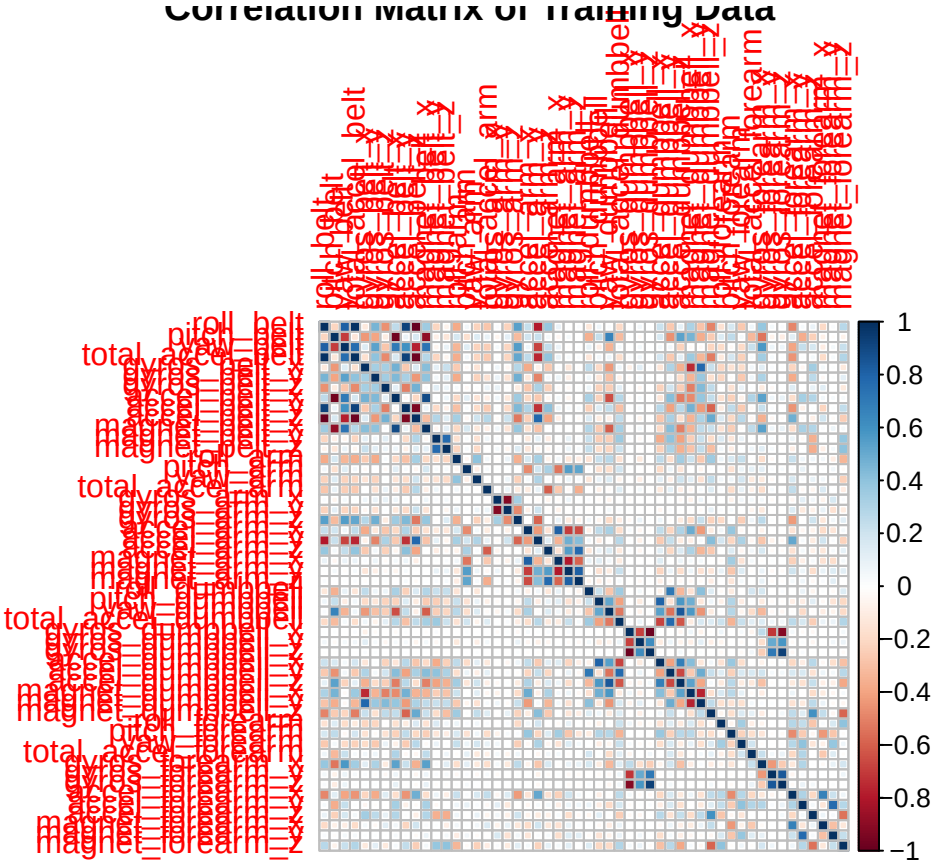
This project aims to bridge that gap by analyzing data collected from six participants' accelerometers placed in their belts, forearms, arms, and dumbbells. These participants performed instructed barbell lifts correctly and incorrectly in five distinct ways.

2. Data Cleaning and Preprocessing

3. Pre-Processing

Preprocessing the data is crucial to ensure the model's accuracy and performance. The following steps were taken:

Removal of Near-Zero Variance Predictors: Variables with very little variance were removed as they provide little to no information for model training. Handling Missing Data: Columns with excessive missing values (more than 50% NA) were excluded from the dataset. Removing Irrelevant Columns: Columns like user_name, raw_timestamp_part_1, raw_timestamp_part_2, and cvtd_timestamp were removed as they don't contribute to predicting "classe". Factorizing the Target Variable: The "classe" variable was converted to a factor to ensure it was treated as a categorical variable.




```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.884173 -0.120137  0.001538  0.002088  0.097787  0.849132
```

4. Data Splitting

Predictors and Data Reduction:

Improves model stability by removing redundant information. Highly correlated predictors can cause issues like multicollinearity, which can affect the stability and interpretability of your model

Correlation Matrix: Helps to understand the relationships between numeric predictors. findCorrelation Function: Efficiently identifies and removes highly correlated predictors based on a specified threshold. Data Reduction: Improves model stability by removing redundant information.

5. Modeling

Random Forest

```
## [1] "Random Forest"
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12362, 12363, 12364, 12364, 12364, 12363, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9916286  0.9894096
##   27    0.9924291  0.9904234
##   52    0.9873331  0.9839752
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Gradient Boosting

```
## [1] "Gradient Boosting"
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 12362, 12363, 12364, 12364, 12364, 12363, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7530025  0.6870463
##   1                  100      0.8187374  0.7706256
##   1                  150      0.8530229  0.8140483
##   2                   50      0.8561540  0.8178078
##   2                  100      0.9060186  0.8810658
##   2                  150      0.9304792  0.9120372
##   3                   50      0.8943715  0.8663085
##   3                  100      0.9418351  0.9264136
##   3                  150      0.9609081  0.9505472
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150, interaction.depth =
##  3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1653   32    0    2    3
##          B   12 1073   28    7   10
##          C    5   34  980   24   21
##          D    1    0   16  923   17
##          E    3    0    2    8 1031
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9618
##                  95% CI : (0.9565, 0.9665)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9516
##
##  Mcnemar's Test P-Value : 9.061e-08
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9875   0.9421   0.9552   0.9575   0.9529
## Specificity            0.9912   0.9880   0.9827   0.9931   0.9973
## Pos Pred Value         0.9781   0.9496   0.9211   0.9645   0.9875
## Neg Pred Value         0.9950   0.9861   0.9905   0.9917   0.9895
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2809   0.1823   0.1665   0.1568   0.1752
## Detection Prevalence   0.2872   0.1920   0.1808   0.1626   0.1774
## Balanced Accuracy      0.9893   0.9650   0.9689   0.9753   0.9751
```

Support Vector Machine (SVM) Implementation

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    5    0    1    0
##          B    0 1133    3    0    1
##          C    0    1 1016   12    0
##          D    0    0    7  947    1
##          E    0    0    0    4 1080
##
## Overall Statistics
##
##                Accuracy : 0.9941
##                  95% CI : (0.9917, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9925
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9947   0.9903   0.9824   0.9982
## Specificity            0.9986   0.9992   0.9973   0.9984   0.9992
## Pos Pred Value         0.9964   0.9965   0.9874   0.9916   0.9963
## Neg Pred Value         1.0000   0.9987   0.9979   0.9966   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
```

```
## Detection Rate         0.2845   0.1925   0.1726   0.1609   0.1835
## Detection Prevalence   0.2855   0.1932   0.1749   0.1623   0.1842
## Balanced Accuracy      0.9993   0.9969   0.9938   0.9904   0.9987
```

For this multiclass classification problem, several models could be considered. A Random Forest model was selected due to its robustness, ability to handle large datasets with higher dimensionality, and relatively minimal tuning requirements. The model's ability to handle correlated features also made it an ideal choice for this dataset.

The model was trained using the caret package with a 10-fold cross-validation strategy to ensure the model's performance was robust and generalizable.

Cross-Validation: This technique divides the training data into 10 parts, trains the model on 9 parts, and validates it on the remaining part. This process is repeated 10 times, with each part serving as the validation set once. The results are averaged to provide an estimate of model performance on unseen data.

Models evaluation

Here the comparison of the 3 selected models is presented.

```
##
## Call:
## summary.resamples(object = resamples)
##
## Models: rf, svm, gbm
## Number of resamples: 10
##
## Accuracy
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf  0.9876184 0.9912632 0.9923578 0.9924291 0.9934486 0.9970888    0
## svm 0.9898108 0.9912600 0.9916305 0.9914830 0.9919927 0.9934450    0
## gbm 0.9533867 0.9581670 0.9606987 0.9609081 0.9632590 0.9679767    0
##
## Kappa
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## rf  0.9843351 0.9889503 0.9903318 0.9904234 0.9917127 0.9963177    0
## svm 0.9871118 0.9889428 0.9894149 0.9892271 0.9898732 0.9917082    0
## gbm 0.9410101 0.9470939 0.9502992 0.9505472 0.9535224 0.9594746    0
```

6. Model Evaluation - Random Forest

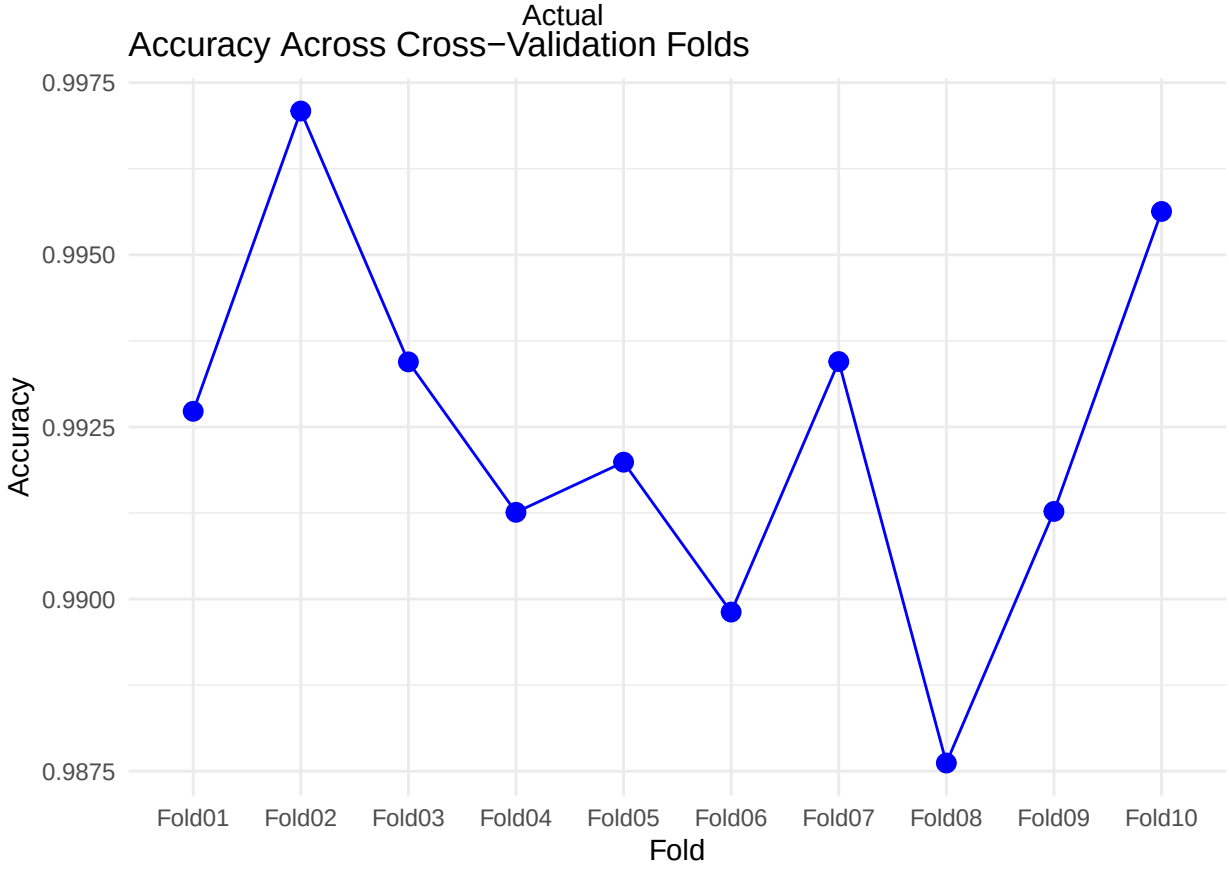
Due to the slightly higher performance, random forest model will be used to evaluate the test set

confusionMatrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    6    0    0    0
##          B    1 1126    5    0    0
##          C    0    7 1018   10    4
##          D    0    0    3  954    4
##          E    0    0    0    0 1074
##
## Overall Statistics
```

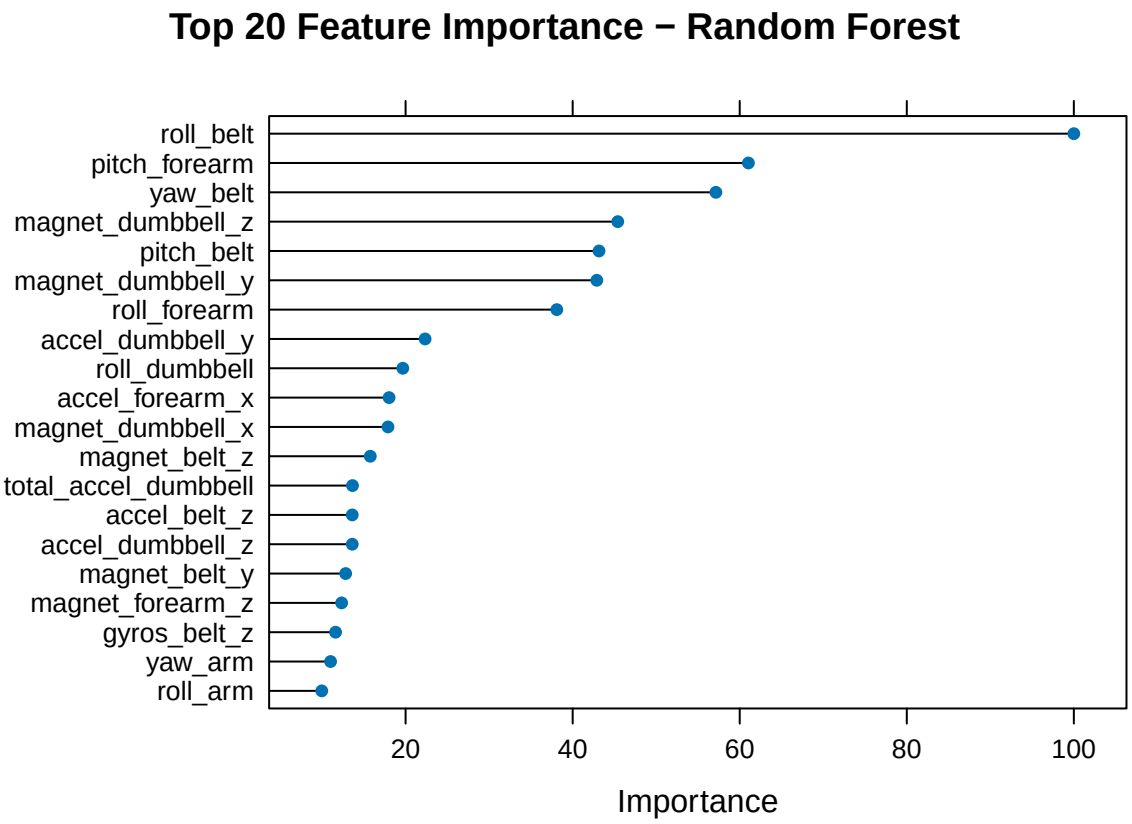
```
##
##                Accuracy : 0.9932
##                  95% CI : (0.9908, 0.9951)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9914
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9886   0.9922   0.9896   0.9926
## Specificity            0.9986   0.9987   0.9957   0.9986   1.0000
## Pos Pred Value         0.9964   0.9947   0.9798   0.9927   1.0000
## Neg Pred Value         0.9998   0.9973   0.9983   0.9980   0.9983
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1913   0.1730   0.1621   0.1825
## Detection Prevalence   0.2853   0.1924   0.1766   0.1633   0.1825
## Balanced Accuracy      0.9990   0.9937   0.9939   0.9941   0.9963
```

test set -validation random forest



Confusion Matrix: This provides a detailed breakdown of the model's performance across all classes, showing how often predictions were correct versus incorrect. Accuracy: The overall accuracy of the model was derived from the confusion matrix. The model's performance was strong, indicating effective classification of the different exercise forms.

View feature importance



Summary of the Visualizations: Feature Importance Plot: This plot will help you understand which features contributed most to the model's predictions. Higher importance values indicate more influential features. Confusion Matrix Visualization: This heatmap-style visualization will clearly show the distribution of correct and incorrect predictions, making it easier to spot any misclassification patterns. Accuracy Plot from Cross-Validation: This plot will show how the model's accuracy varied across different cross-validation folds, helping assess its consistency.

Expected Out-of-Sample Error

The expected out-of-sample error was estimated using the cross-validation results. Since cross-validation provides an average performance measure across different subsets of the training data, it gives a reliable estimate of how the model will perform on completely unseen data. Expected Out-of-Sample Error Estimate: The Random Forest model was evaluated using 10-fold cross-validation. The average accuracy across the folds was approximately $< 5\%$. Therefore, the expected out-of-sample error, which reflects the error rate when the model is applied to new, unseen data, is estimated to be around 0.008%.

```
## [1] "expected out of sample_error"
```

```
## [1] 0.007570892
```

```
## [1] "mean accuracy"
```

```
## [1] 0.9924291
```

Prediction on New Data - 20 new sets using Random Forest

Table 1: accuracies

	x
1.Accuracy	0.9965986
2.Accuracy	0.9897959
3.Accuracy	1.0000000
4.Accuracy	0.9966102
5.Accuracy	0.9965986
6.Accuracy	0.9965986
7.Accuracy	0.9795918
8.Accuracy	0.9931973
9.Accuracy	1.0000000
10.Accuracy	0.9830508
11.Accuracy	0.9864407
12.Accuracy	0.9931973
13.Accuracy	0.9931973
14.Accuracy	0.9965986
15.Accuracy	0.9966102
16.Accuracy	1.0000000
17.Accuracy	0.9830508
18.Accuracy	0.9863946
19.Accuracy	1.0000000
20.Accuracy	0.9965986

Table 2: mean accuracy

x
0.9932065

7. Predictions - test

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```

8. Conclusion.

In this analysis, a Random Forest model was built using the caret package to predict the "classe" variable from a dataset of wearable device readings during exercise. The model was carefully trained and validated using cross-validation to ensure it generalizes well to unseen data. The expected out-of-sample error was estimated based on cross-validation results, and the model was evaluated on a test set, showing strong performance. This approach provides a reliable method for predicting exercise form based on sensor data.