

# 1 Data Mining: BeeViva Challenges

---

2  
3 MARCO BELLÒ, University of Padua, Italy  
4

5 All R source code, images and L<sup>A</sup>T<sub>E</sub>X sources, can be found at the following repository: <https://github.com/mhetacc/DataMiningChallenges/>.  
6 Datasets have been removed and can be found at <https://www.datachallenge.it/competitions/>.  
7

8 **ACM Reference Format:**

9 Marco Bellò. 2025. Data Mining: BeeViva Challenges. 1, 1 (December 2025), 20 pages. <https://doi.org/XXXXXX.XXXXXXXX>  
10

11 **1 RICE VARIETIES**

12 **1.1 Preliminary Observations**

13 *1.1.1 Dataset.* All following considerations are made using the datasets provided on the challenge page (*rice\_test.csv* and *rice\_train.csv*), but it is worth noting that they both stem from an original one that can be found at the [dataset source page](#). I used it to compute an alternative version of the test dataset that contains the true attribute for *Class* (listing 1).

20  
21 Listing 1. Merge datasets

```
22 1 df1 = pd.read_csv("rice_test.csv").round(10)
23 2 df2 = pd.read_csv("Rice_Cammeo_Osmancik.csv").round(10)
24 3
25 4 keys = [col for col in df1.columns]
26 5 merged = df1.merge(df2, on=keys, how="inner")
```

27  
28 *1.1.2 Scatter Plot.* From the *scatter plot* (figure 1) we can infer that there are four features (*Area*, *Perimeter*, *Convex\_Area* and *Major\_Axis\_Length*) that seem to be extremely correlated

29  
30 *1.1.3 Correlation Matrix.* We can use a correlation matrix (table 1) to see exactly how related to each other the features  
31 are. As suspected, all four features before-mentioned have a high degree of correlation (over ninety percent). Once way  
32 to handle this is to either use models robust against collinearity, or to either combine or remove some of the correlated  
33 features.

34 *1.1.4 Features' Importance.* To get a better idea on the importance of each feature, I trained a simple linear regression  
35 model, and looked at the result with *summary(fit)*, which can be seen in table 2. The significance stars tells us visually  
36 that, with the exception of *Minor\_Axis\_Length*, the features that are highly correlated to each other contributes strongly  
37 to the model, while the *Eccentricity* and *Extent* contribute very little.

---

38 Author's address: Marco Bellò, marco.bello.3@studenti.unipd.it, University of Padua, Padua, Italy.  
39

40  
41 Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not  
42 made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components  
43 of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on  
44 servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

45 © 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

46 Manuscript submitted to ACM

47  
48 Manuscript submitted to ACM

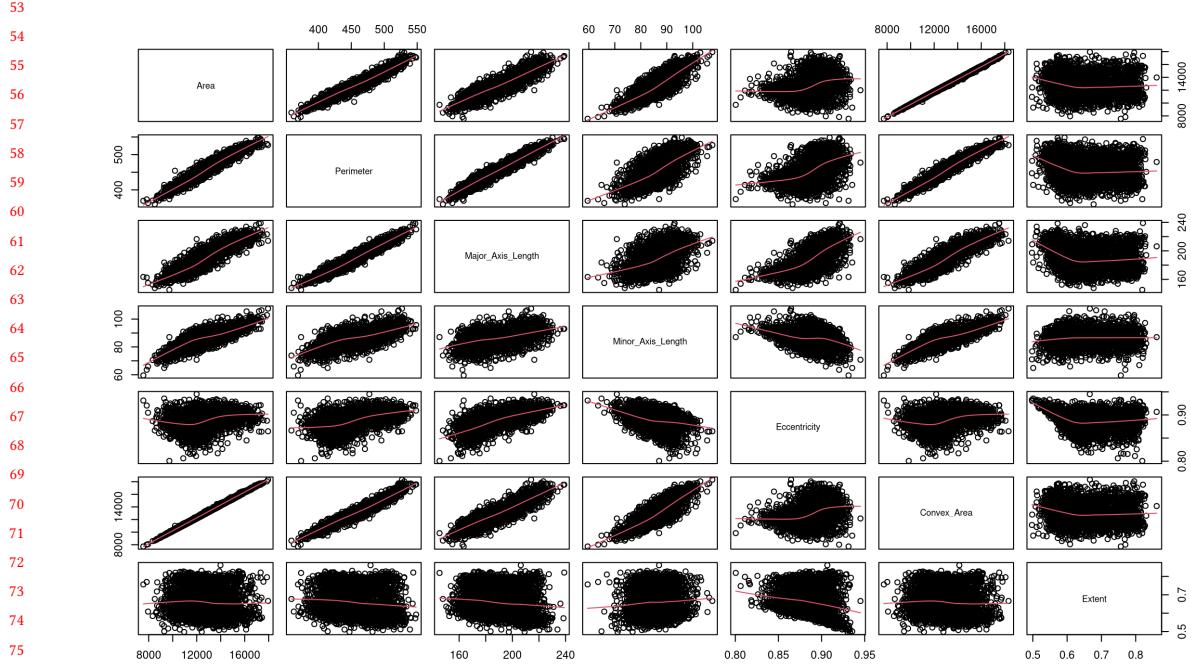


Fig. 1. Scatter Plot for the *rice\_train* dataset with LOESS smoothing lines for each class.

Table 1. Correlation matrix of features

	Area	Perimeter	Major Axis Length	Minor Axis Length	Eccentricity	Convex Area	Extent
Area	1.00000000	0.96704011	0.90356325	0.79022701	0.34653177	0.99895272	-0.06002223
Perimeter	0.96704011	1.00000000	0.97163180	0.63486482	0.53784650	0.97046788	-0.12718015
Major Axis Length	0.90356325	0.97163180	1.00000000	0.45675159	0.70625660	0.90390912	-0.13562592
Minor Axis Length	0.79022701	0.63486482	0.45675159	1.00000000	-0.29393931	0.78975480	0.06192558
Eccentricity	0.34653177	0.53784650	0.70625660	-0.29393931	1.00000000	0.34707340	-0.19301157
Convex Area	0.99895272	0.97046788	0.90390912	0.78975480	0.34707340	1.00000000	-0.06397213
Extent	-0.06002223	-0.12718015	-0.13562592	0.06192558	-0.19301157	-0.06397213	1.00000000

We can also measure the total variance explained by all predictors combined using  $R^2 = 0.6953849$ , and we can check the standardized coefficients to better compare predictors' importance, as shown in table 3. Once again, we can see that Extent and Eccentricity contribute very little to the overall model.

**1.1.5 Principal Components.** A good way to aggregate and simplify data is by decomposing it into its principal components (centered in zero and scaled to have unit variance). Components' summary can be seen in table 4, and it shows us that more than ninety nine percent of overall variance can be explained with just three components, so a 2D visualization with only the first two components should give us a good idea of the whole dataset (figure 2). The plot suggests that data is well separated, with similar classes clustering nicely. This should make classification easier.

Table 2. Regression coefficients and coefficients' importance

Parameter	Estimate	Std. Error	t value	Pr(> t )	Importance
(Intercept)	-2.152e+00	1.990e+00	-1.081	0.279633	
Area	5.601e-04	1.023e-04	5.474	4.79e-08	***
Perimeter	8.440e-03	2.221e-03	3.800	0.000148	***
Major_Axis_Length	-2.198e-02	5.709e-03	-3.851	0.000120	***
Minor_Axis_Length	4.669e-02	1.213e-02	3.850	0.000121	***
Eccentricity	4.534e+00	1.828e+00	2.481	0.013170	*
Convex_Area	-8.609e-04	9.418e-05	-9.141	< 2e-16	***
Extent	7.146e-02	7.144e-02	1.000	0.317237	

Table 3. Standardized regression coefficients

Parameter	Standardized Coefficient
(Intercept)	NA
Area	1.95970669
Perimeter	0.60605955
Major_Axis_Length	-0.77272038
Minor_Axis_Length	0.54256082
Eccentricity	0.18931314
Convex_Area	-3.09099064
Extent	0.01114328

Table 4. Summary of principal components

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Standard deviation	2.2924	1.2436	0.9562	0.51393	0.10621	0.07758	0.04519	0.02052
Proportion of Variance	0.6569	0.1933	0.1143	0.03302	0.00141	0.00075	0.00026	0.00005
Cumulative Proportion	0.6569	0.8502	0.9645	0.99753	0.99894	0.99969	0.99995	1.00000

## 1.2 Modeling

After exploring the dataset, I tried to find the best performing model for predicting the target.

**1.2.1 Linear Regression.** First I split the training dataset into training and validation sets, fitted the model and compute its *MSE*. From this basic regression I got  $MSE = 0.07354557$ . Since we know from the preliminary observations that some features are highly correlated (so the dataset suffers from collinearity), I tried to transform the dataset to reduce it. A straightforward solution is to compute the average of all of them (listing 2), but this yield a slightly worse result:  $MSE \approx 0.074$ .

Another approach would be to drop features altogether, but first we want to measure their variance inflation factor. As a rule of thumb, a *VIF* value above then indicates a problematic amount of collinearity, so I tired to drop features iteratively to see if the model improves, as shown in table 5. For each dropped variable I computed the *VIFs* and the *MSE* of the resulting model. As we can see, dropping *Area* resulted in performances similar to combining the variables, while dropping the other features worsened prediction performances even further.

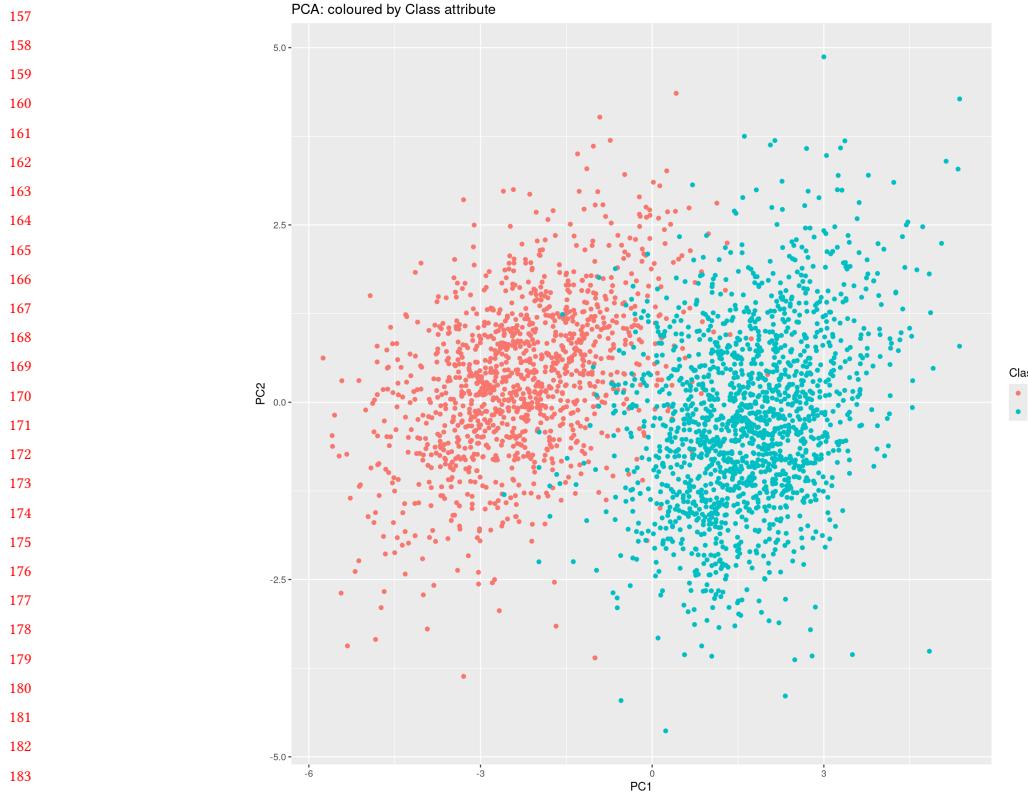


Fig. 2. Plot with first two principal components of *rice\_train* dataset.

Table 5. Effect of variable removal on variance inflation factors (VIF) and mean squared error (MSE)

Variable Dropped	VIF Perimeter	VIF Major_A_L	VIF Minor_A_L	VIF Ecc.	VIF Convex_A	VIF Extent	MSE
Area	114	206	133	53	332	1	0.0735629
Convex Area	108	134	40	49	-	1	0.07572877
Major Axis Length	47	-	37	29	-	1	0.07718435
Perimeter	-	-	1	1	-	1	0.0806868

Listing 2. Average of highly correlated features

```
1 rice.train$Combined <- rowMeans(rice.train[, c("Area", "Perimeter", "Major_Axis_Length", "Convex_Area")])
2
3 linear_fit = lm(
4   Class ~
5   Minor_Axis_Length
6   + Eccentricity
7   + Extent
8   + Combined,
9   data=rice_train,
10  subset = train
11 )
```

```

209 12
210 13 pred <- predict(linear_fit, newdata = rice_train[test, ])
211 14 mean((pred - y.test)^2)      # [1] 0.07439239
212

```

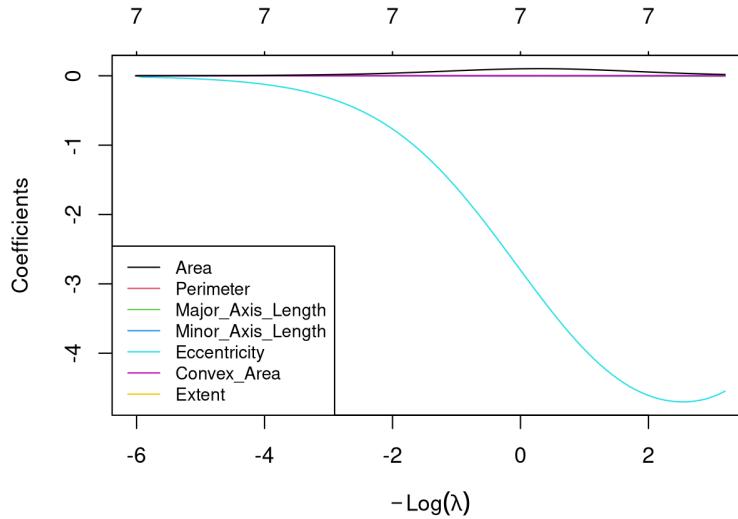
213  
214   1.2.2 *Ridge Regression.* I used the *glmnet* library, which takes care of the variable standardization for us, has a built-in  
215 cross-validation functionality to compute the best possible  $\lambda$ . With code 3 we get a result worse than linear regression,  
216 since  $MSE = 0.07924703$ . That being said, we can see in plot 3 an effective shrinkage of most coefficients towards zero,  
217 with the exception of Eccentricity, which we already recognized as not very impactful on the overall variance and  
218 prediction.  
219

Listing 3. Ridge regression on rice dataset

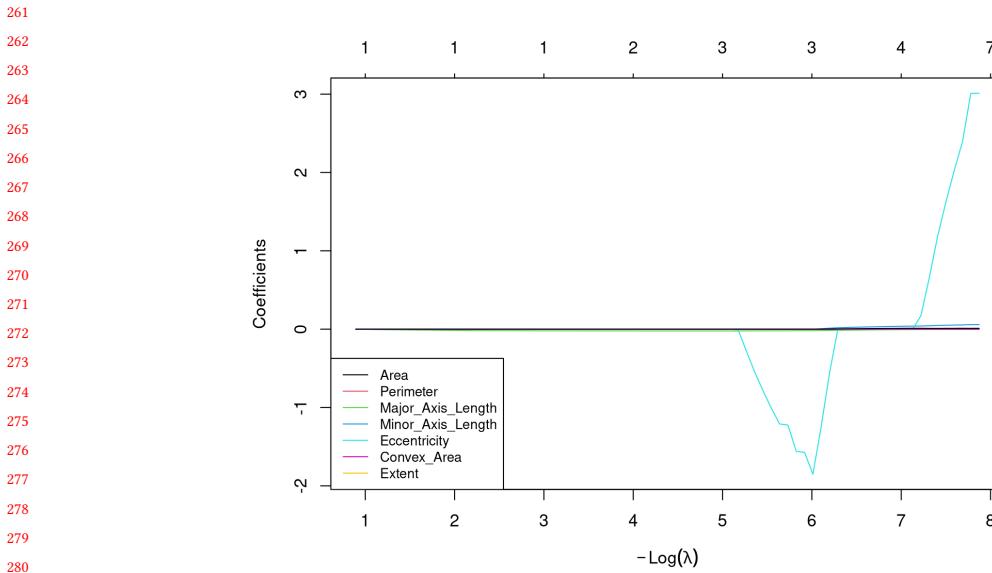
```

220
221
222 1 library(glmnet)
223 2 cv.out <- cv.glmnet(x[train, ], y[train], alpha = 0)
224 3 plot(cv.out)
225 4 bestlam <- cv.out$lambda.min
226 5 cv_mse <- min(cv.out$cvm)      # [1] 0.07924703
227 6 cv_rmse <- sqrt(cv_mse)       # [1] 0.2815085
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260

```

Fig. 3. Ridge coefficients for *rice\_train* dataset.

255   1.2.3 *Lasso Regression.* Very similar to ridge regression, except *glmnet* gets called with  $\alpha = 1$ . We get a slightly  
256 better result than ridge, with  $MSE = 0.07348723$ . From the plot (in figure 4), we can see that once again Eccentricity  
257 seems to resist shrinkage, but overall coefficient reduction is much more accentuated compare to ridge, since lasso can  
258 set them to zero.  
259

Fig. 4. Lasso coefficients for *rice\_train* dataset.

**1.2.4 Principal Components Regression.** I fitted the model with *pls* library built-in cross-validation, as seen in code 4. The lowest error (figure 5) is achieved with seven components:  $MSE = 0.07354557$ , slightly worse than lasso regression.

Listing 4. Principal components regression on rice dataset

```

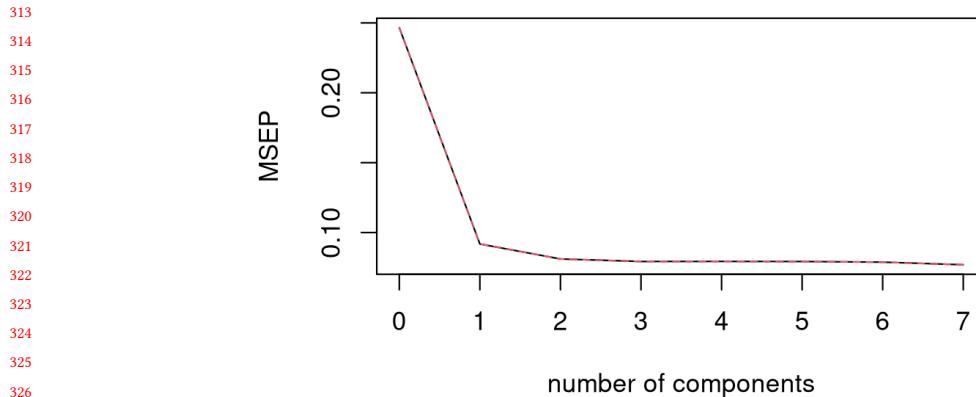
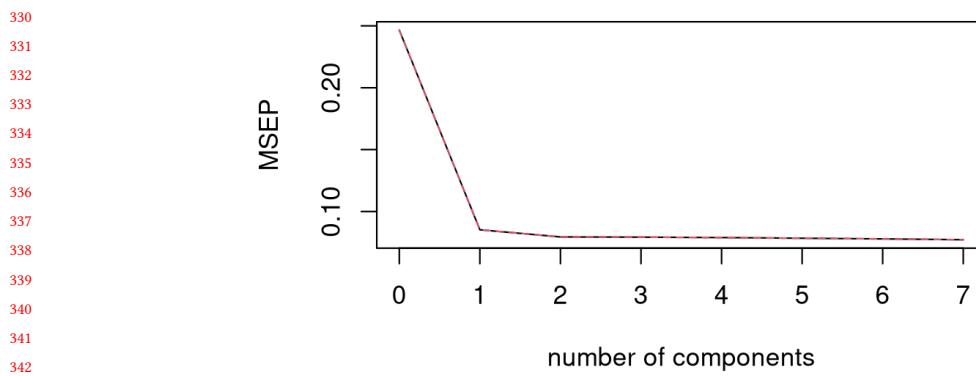
1 pcr.fit <- pcr(
2   Class ~ .,
3   data = rice_train,
4   subset = train,
5   scale = TRUE,
6   validation = "CV"
7 )

```

**1.2.5 Partial Least Squares.** I used the same *pls* library to fit the partial least squares model on the dataset. The best result is obtained with just two components (figure 8), yielding  $MSE = 0.07609556$ , which is slightly worse than PCR but needs a third of the components, potentially leading to less overfitting and computational cost.

**1.2.6 LOESS.** I then tried some non-parametric methods, starting with LOESS regression. This method allows up to four predictors, so I used the most useful ones previously identified: *Area*, *Perimeter*, *Convex\_Area* and *Major\_Axis\_Length*. Predictors are normalized by default. Then, fitted three models with three different values for span. Predictions must be sanitized since there some test points can fall outside the local neighborhood, meaning being Na. With  $span = 0.1$  we have five such predictions. The code can be seen at 5. The results were disastrous, so I tried to include only two predictors. This yielded the best results yet out of all models: with  $span = 0.1$ ,  $MSE = 0.06099749$ . All results are summarized in table 6

Listing 5. LOESS regression on rice dataset with four predictors and  $span = 1$

Fig. 5. PCR cross-validation MSE error for target *Class* on *rice\_train* dataset.Fig. 6. PLS cross-validation MSE error for target *Class* on *rice\_train* dataset.

```
348 1 loess_fit1 <- loess(  
349 2   Class ~ Area  
350 3   + Perimeter  
351 4   + Convex_Area  
352 5   + Major_Axis_Length,  
353 6   data = rice_train,  
354 7   subset = train,  
355 8   span = 0.1  
356 9 )  
357 10  
358 11 pred1 <- predict(loess_fit1, newdata = rice_train[test, ])  
359 12 mean((pred1 - y.test)^2, na.rm = TRUE) # [1] 46.0226
```

360 1.2.7 *K*-Nearest Neighbors. I used the *caret* library to fit KNN classification both for regression and classification. The  
361 former to get a comparable prediction with the other models, the latter to be more logically sound, since *Class* is a  
362 categorical target. The code for classification can be seen at 6, for regression the only difference is the *metric* parameter:  
363

Table 6. MSE for LOESS with different *span* values and number of predictors

Span	Number of Predictors	MSE
0.1	Four (4)	46.0226
0.5	Four (4)	1.799422
0.9	Four (4)	0.2276324
0.1	Two (2)	0.06099749
0.5	Two (2)	0.06514592
0.9	Two (2)	0.06836381

"Accuracy" for classification, "RMSE" for regression. For both methods I used Leave-One-Out-Cross-Validation, and both RMSE plots can be seen in figure 7.

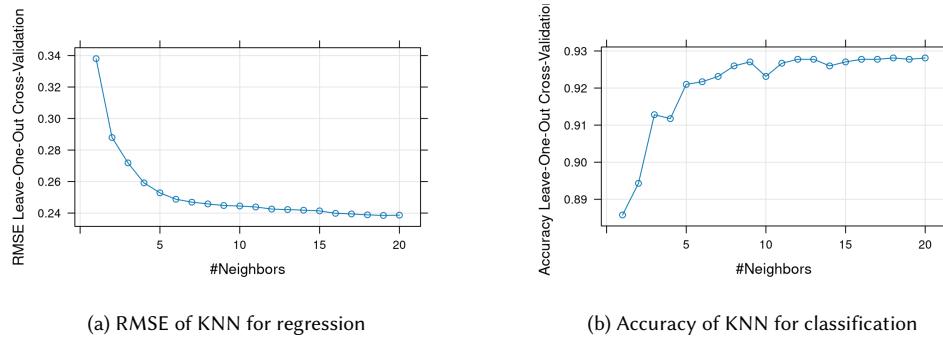
For regression, the best result is achieved with nineteen neighbors, yielding a mean square error even better than LOESS:  $MSE = 0.05688406$ . For classification, the best accuracy is achieved with twenty neighbors, yielding an accuracy of almost 93%. This last model is the one I ultimately used to predict the target values on the test dataset.

Listing 6. KNN regression on rice dataset with Leave-One-Out-Cross-Validation

```

385 1 train.control <- trainControl(method = "LOOCV")
386 2 train$Class <- as.factor(train$Class) # necessary for classification
387 3
388 4 knn_fit <- train(
389 5   Class ~ .,
390 6   method      = "knn",
391 7   tuneGrid    = expand.grid(k = 1:20),
392 8   trControl   = train.control,
393 9   preProcess  = c("center", "scale"),     # normalized
39410  metric       = "Accuracy",
39511  data         = train
39612 )

```

Fig. 7. RMSE and Accuracy respectively of KNN for regression and for classification with LOOCV on *rice\_train* dataset.

1.2.8 *Random Forest*. The last method I tried is *random forest* with Leave-One-Out-Cross-Validation, using the *caret* library still. Since this method is extremely computationally expensive, I had to leverage the library *doParallel* to use

Manuscript submitted to ACM

417 all available CPU cores, and I used the *ranger* method instead, which is a faster random forest implementation. I also  
 418 reduced the amount of predictors by combining the highly correlated ones (we already saw that dropping them yield  
 419 similar to worse results). The code can be seen at ??.

420 The optimal model, with *mtry* = 2, *splitrule* = *extratress* and *min.node.size* = 5, yielded a slightly worse result than  
 421 KNN trained on all features: *MSE* = 0.0589397. The plot for *RMSE* can be seen in figure ??

423 Listing 7. Random Forest regression on *rice\_train* dataset with Leave-One-Out-Cross-Validation

```
424
425 1 train.control <- trainControl(method = "LOOCV")
426 2
427 3 rf_fit <- train(
428 4   Class ~ Minor_Axis_Length + Eccentricity + Extent + Combined,
429 5   method      = "ranger",
430 6   tuneLength = 10,
431 7   trControl  = train.control,
432 8   metric      = "RMSE",
433 9   data        = rice_train,
434 10  allowParallel = TRUE
435 11 )
```

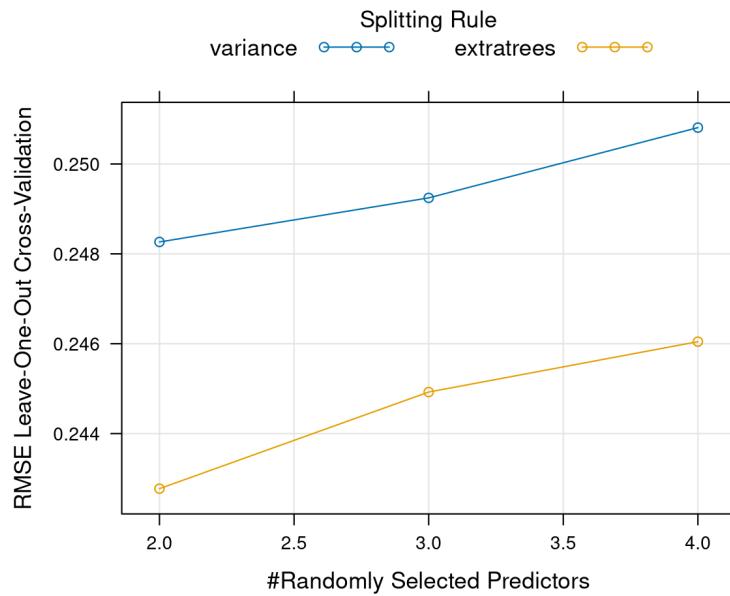


Fig. 8. RMSE error for random forest on *rice\_train* dataset.

### 1.3 Rice Varieties Conclusions

464 The best model so far is KNN with *K* = [19, 20]. While slightly more computationally expensive than other methods  
 465 (with the exception of random forest) it improves MSE by 0.00411343 over the second best method (LOESS with  
 466 *span* = 0.1). I decided to use predict the target with a KNN fit in classification mode, since I felt it more logically sound.  
 467

469 I decided not to try random forest with all predictors for time constraints: it required a 10-Cores machine (i7-13620H  
470 @ 4.90 GHz) twenty-five minutes to fit the model.  
471

## 472 2 PHONE USERS

### 473 2.1 Preliminary Observations

474 2.1.1 *Dataset.* Let's first briefly analyze the dataset.

- 475 • **For each user:**

- 476 – Plan type
- 477 – Payment method
- 478 – Sex
- 479 – Activation zone
- 480 – Activation channel
- 481 – Value-added service 1
- 482 – Value-added service 2

- 483 – **For each month:**

- 484 \* |expensive calls|
- 485 \* |cheap calls|
- 486 \* time(expensive calls)
- 487 \* time(cheap calls)
- 488 \* cost(expensive calls)
- 489 \* cost(cheap calls)
- 490 \* |incoming calls|
- 491 \* time(incoming calls)
- 492 \* |SMS|
- 493 \* |calls to call center|

494 Considering that the target is monthly call time we can make the hypothesis that some features do not interest us,  
495 for example any monthly feature not concerning with call time itself, leaving us with only *time(expensivecalls)* and  
496 *time(cheapcalls)*, which we could further assume can be combined since overall cost is not a concern. It goes without  
497 saying that all of the above will have to be proven empirically.

500 2.1.2 *Scatter Plot.* I filtered out all not call time-related monthly data. The result (figure 9) is a bit hard to read due to  
501 its sheer size. Looking at it more up close we can infer some things: first of all we see that call data looks positively  
502 skewed (figure 10), and that there are some activation channels that see more call time than others, while activation  
503 regions seem to differ lightly between one another (figure 11a). On the other hand, plan and sex seem to have an impact  
504 on call time (figure 11b), while age does not seem to have a big impact, with the exception of the very young and very  
505 old (figure 11c).

506 2.1.3 *Call Features Over Everything Else.* Let's try to see if calls amount and time are related to other features. First I  
507 plotted total calls taken and total calls time over nine months (figure 13). We can see that the graphs are almost identical  
508 from a trend standpoint, hinting at a strong correlation between number of calls and time spent calling.

509 Then I wanted to see whether sex plays any role in total call time. Using the library *dplyr* I computed all data in  
510 table 7. On average, men's calls seem to last 8% longer. To mitigate the effect of outliers I filtered out the top 1%, which  
511 still shows the same trend, albeit reduced to 5%. Cutting even further the top 10% shows a bigger increase in men's  
512 Manuscript submitted to ACM

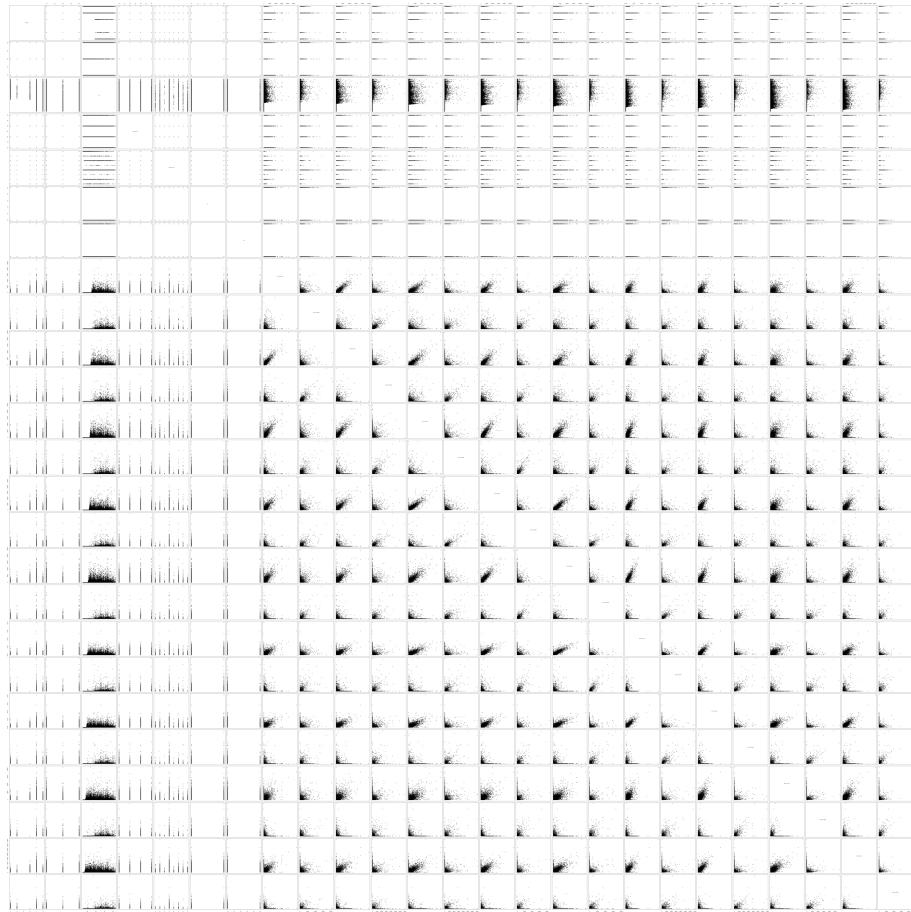
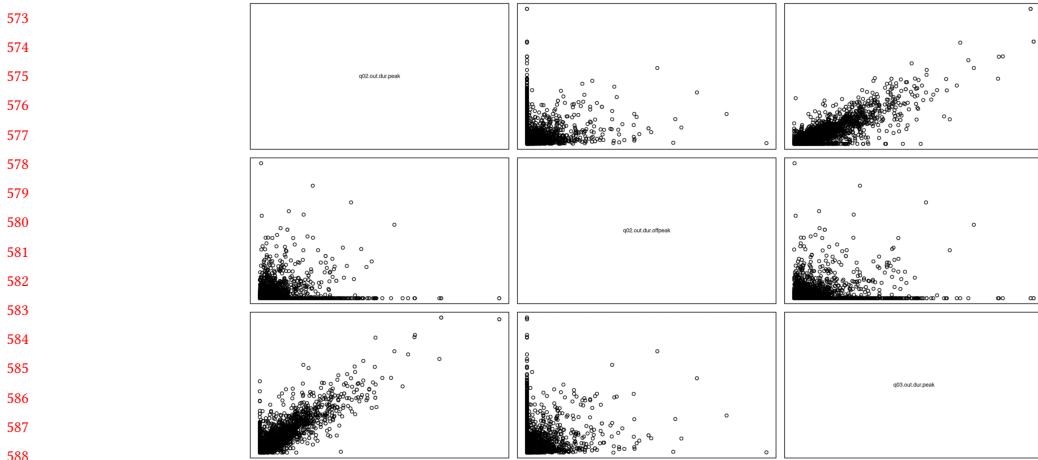
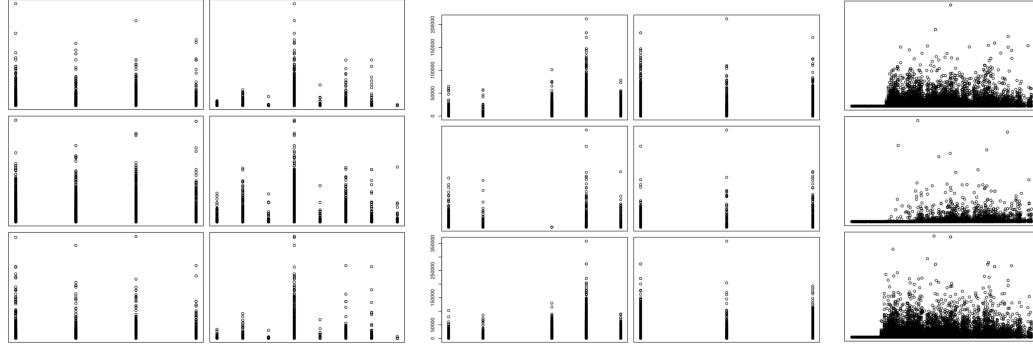


Fig. 9. Scatter Plot for the filtered *phone\_train* dataset.

average call times of about 10%. Lastly, I tried to log-transform the data. Even in this situation we see that men have longer calls, hinting that this predictor may be useful. I did exactly the same with payment methods, activation zones, activation channels and both value-added services, always cutting off the top 1%. All data can be seen in table 8. There seem to be quite a bit of variance in call time compared to the tariff plan, which would make sense: some plans may be geared toward calling, while others are more suited sending SMS. We can see some variance on payment method too. Activation zone and activation channel seem to have some influence on call times, with the exception of zone eight, whose call times are way lower. On the other hand, customers that have activated either first or second added-value services seem to do longer calls.

**2.1.4 Skewness.** Call time data is usually positively skewed, with many users having low usage and a small number of user with very high usage (e.g., 50.000 seconds). This is easily verifiable at a glance if we look at the density graph in figure 13a. This effect can be mitigated by log-transforming the data, as shown in figure 13b. Moreover, with the *e1071* library I computed a skewness matrix for all features (appendix A.1): we can see that all monthly-based data (calls, call

Fig. 10. Scatter Plot for call time for *phone\_train* dataset.

(a) Activation zone (X-axis, L) and activation channel (X-axis, R) over call time (Y axis) (b) Plan type (X-axis, L) and sex (X-axis, R) over call time (Y axis) (c) Age (X-axis) over call time (Y axis)

Fig. 11. Different features plotted over call time for the *phone\_train* dataset.

times, SMS, costs, etc) are highly positively skewed. To manage this problem we can either transform them (log, square root or Box-Cox transform), or use more robust models (such as random forest, Gamma/Poisson or quantile regression). Methods like linear, ridge or lasso regression, and even some non-parametric ones like KNN are not ideal.

**2.1.5 Correlation Matrix.** Since `cor` function only works on numerical data, I transformed all categorical with the `model.matrix` function. We can infer the following: age is correlated to all call-related monthly features by roughly 25%, first and second value-added services are correlated to some, not all, monthly features by roughly 10%, and the only features that share significant correlation are the monthly call-related ones, for example in month four, outgoing call time and call expenses share a correlation of 97%, while for the seventh month the correlation drop to 57%, which is still significant.

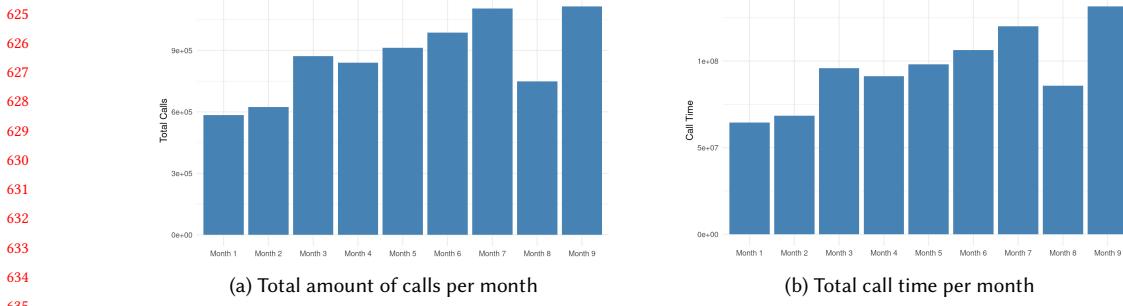
Fig. 12. Total calls and total call time per month for the *phone\_train* dataset.

Table 7. Call time statistics by sex under different data cuts and transformations

Type of cut	Sex	Number of customers	Total call time	Average call time
None (raw)	B	5266	557266412	105823
None (raw)	F	1030	62493290	60673
None (raw)	M	3704	242226228	65396
Top 1% cut	B	5201	498846876	95914
Top 1% cut	F	1025	57203392	55808
Top 1% cut	M	3674	215801665	58738
Top 10% cut	B	4595	276063673	60079
Top 10% cut	F	960	32400501	33751
Top 10% cut	M	3445	130747836	37953
Log-transformed	B	5266	557266412	10.00
Log-transformed	F	1030	62493290	8.24
Log-transformed	M	3704	242226228	8.58

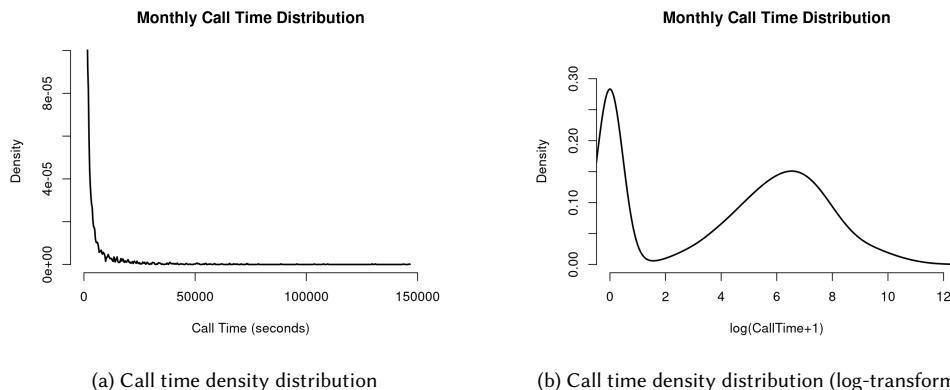
Fig. 13. Call time density distributions for the *phone\_train* dataset.

Table 8. Call time over different categorical features

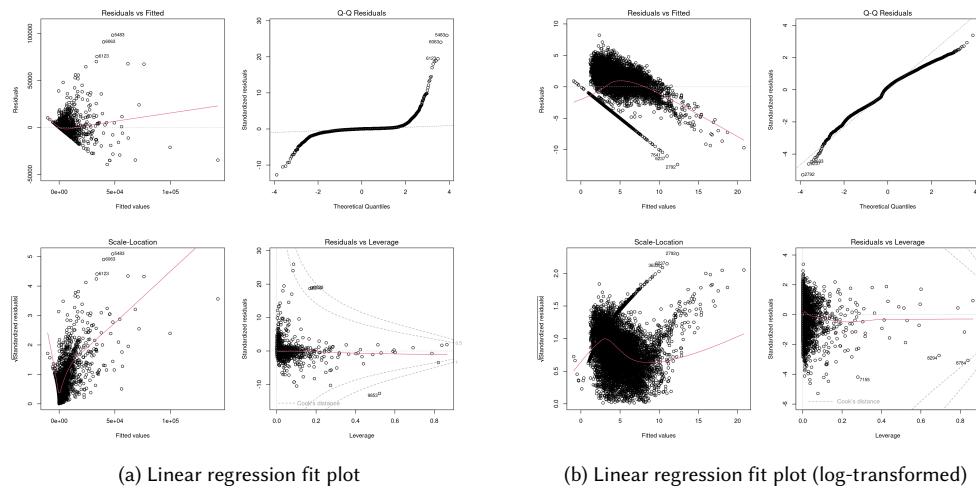
Feature	Number of customers	Total call time	Average call time
Tariff plan 3	781	44811041	57376
Tariff plan 4	83	10575009	127410
Tariff plan 6	724	92214268	127368
Tariff plan 7	3564	489635465	137384
Tariff plan 8	4748	134616150	28352
Activation zone1	3507	288670845	82313
Activation zone2	3130	213438509	68191
Activation zone3	2342	197941555	84518
Activation zone4	921	71801024	77960
Activation channel 2	130	10643060	81870
Activation channel 3	408	36663577	89862
Activation channel 4	31	2369822	76446
Activation channel 5	7135	623961079	87451
Activation channel 6	47	4352853	92614
Activation channel 7	652	62777153	96284
Activation channel 8	111	12074180	108776
Activation channel 9	1386	19010209	13716
Value-added one: No	7450	526620961	70687
Value-added one: Yes	2450	245230972	100094
Value-added two: No	9279	695404226	74944
Value-added two: Yes	621	76447707	123104

2.1.6 *Linear Regression Insights.* A preliminary analysis can be done by plotting a linear regression fit (figure 14a). We can see that the data presents high heteroskedasticity, skewness and in general it is not normally distributed. There are also potential outliers and influential points that could distort the model, such as point 9853. As for features relevance (appendix A.2), for the most part, relevant features are the one related to monthly calls. It is interesting to notice that the second month does not seem to count much, even though it presents a similar amount of call data compared to the others.

Log-transforming the target variable yielded better results (figure 14b), but the above mentioned problems did not disappear. Feature relevance held some surprises (appendix A.3), for example categorical features contributed much more to the model and only a small subset of monthly call data was relevant.

2.1.7 *Principal Components.* Data as-is can be transformed into 101 principal components, and the cumulative proportion with only two is approximately 0.45. To get to 99% of explained variance we need 69 components. Plotting the first two components shows us, once again, signs of heteroskedasticity and non-normal data distribution, as shown in figure 15

2.1.8 *Preliminary Observations Conclusions.* Data shows high skewness and heteroskedasticity, and some features do not seem to contribute much to the model. We probably should try to filter out some data, for example SMS amount, transform monthly call data, for example with log or Box-Cox transforms, and use some robust regression methods, such as quantile or tree-based approaches.

Fig. 14. Linear regression fit plots for *phone\_train* dataset.

## 2.2 Modeling

**2.2.1 Lasso Regression.** I used lasso regression with cross-validation as a preliminary diagnostic tool to check which features drop or transform. First, I made a baseline prediction without transforming any value, with the following mean square error:  $MSE = 18091358$ . Not great, but expected since we saw in the preliminary observations many signs of non-linearity.

I then tried log-transforming the target value (with all features), which resulted in a massive improvement:  $MSE = 6.200236$ .

Then, on the not transformed target, I tried different predictions with different data filtering, with the following results: filtering out incoming calls, SMS and calls to the call center yielded a 3% improvement over the all-features model ( $MSE = 18040253$ ). Putting calls to the call center back in worsened the model slightly ( $MSE = 18080968$ ). I thus decided to leave these features filtered out permanently.

Then, I tried to filter out either both or any between calls value and calls amount, which produced a worse performing model in all instances. Then, I filtered out non monthly features, and discovered in the end that the best result ( $MSE = 18019454$ ) can be obtained with the following features **filtered out**: payment method, value-added service two, activation zone and activation channel (plus, of course, the previously mentioned incoming calls, SMS and calls to the call center).

I then tried to pre-process the monthly features, such as adding together peak and off-peak values or predict on the average monthly call times. All yielded worse results.

Lastly, with the best performing filtering, I log-transformed only the target ( $MSE = 6.253648$ ), only the monthly features ( $MSE = 20415157$ ), and then both, which ultimately yielded the best result:  $MSE = 4.289544$ . It is worth noting that log-transforming the target on the filtered dataset produce a worse performing fit than a model fitted on all data and log-transformed target.

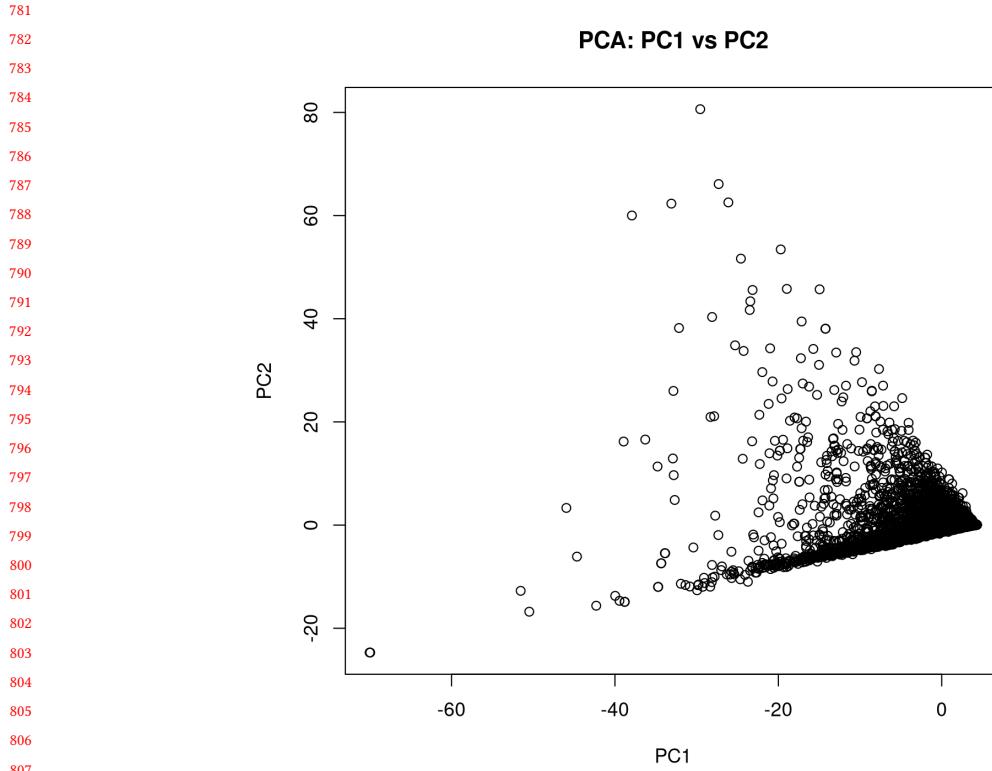


Fig. 15. Scatter Plot for call time for the *phone\_train* dataset.

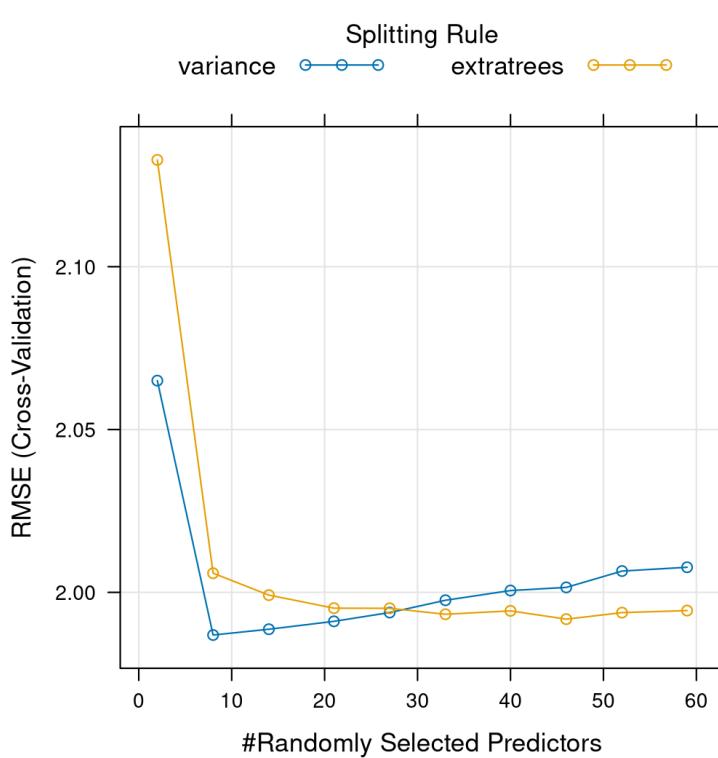
To summarize: the best performing model is obtained by filtering out payment method, value-added service two, activation zone, activation channel, incoming calls, SMS and calls to the call center, and by log-transforming the target and the remaining monthly call-related features.

The complete list of filtering and transformations can be seen in appendix A.4

**2.2.2 Random Forest.** Considering all of the above a model inherently robust to non-normally distributed data should yield better results. Random Forest is a good example of such a model. I still log-transforming both target and predictor since it showed good results, and fitting this type of model requires a lot of time. I run the model with the `ranger` method on a `tuneLength` of ten with cross-validation. Compared to the previous challenge I had to give up on LOOCV due to memory constraints. The resulting best fit (figure ??, with `mtry = 8`, variance as the split rule and minimum node size of five) yielded the best mean square error overall:  $MSE = 3.947784$ , which is why I ultimately used this model to predict the target value.

### 2.3 Phone Users Conclusions

The dataset was, in fact, highly not normally distributed and non-linear, so transforming the features greatly improved a linear fit such as lasso. Moreover, using a robust model yielded even better results, proving our hypothesis.

Fig. 16. RMSE for random forest for the *phone\_train* dataset.

## A APPENDICES

### A.1 Phone call time skewness

Listing 8. Skewness matrix

1	tariff.plan	activation.channel	q01.out.ch.peak	q01.out.dur.peak	q01.out.val.peak
2	-2.026605	1.073333	4.930308	5.883900	10.158610
3	q01.out.dur.offpeak	q01.out.val.offpeak	q01.in.ch.tot	q01.in.dur.tot	q01.ch.sms
4	9.336163	31.977903	4.322467	3.615376	37.220847
5	q02.out.ch.peak	q02.out.dur.peak	q02.out.val.peak	q02.out.ch.offpeak	q02.out.dur.offpeak
6	4.011558	5.026925	6.120357	9.636244	9.202850
7	q02.in.ch.tot	q02.in.dur.tot	q02.ch.sms	q02.ch.cc	q03.out.ch.peak
8	3.657302	3.444048	38.464828	10.697464	3.742071
9	q03.out.val.peak	q03.out.ch.offpeak	q03.out.dur.offpeak	q03.out.val.offpeak	q03.in.ch.tot
10	4.144170	8.709344	9.948054	25.074765	3.359255
11	q03.ch.sms	q03.ch.cc	q04.out.ch.peak	q04.out.dur.peak	q04.out.val.peak
12	35.231226	12.146903	3.726583	4.415209	3.988589
13	q04.out.dur.offpeak	q04.out.val.offpeak	q04.in.ch.tot	q04.in.dur.tot	q04.ch.sms
14	12.366396	10.517232	3.362859	3.660378	33.513288
15	q05.out.ch.peak	q05.out.dur.peak	q05.out.val.peak	q05.out.ch.offpeak	q05.out.dur.offpeak
16	3.578910	4.056863	4.242749	12.129119	12.598793

885	17	q05.in.ch.tot	q05.in.dur.tot	q05.ch.sms	q05.ch.cc	q06.out.ch.peak
886	18	3.450009	3.165951	38.804264	9.085293	3.364608
887	19	q06.out.val.peak	q06.out.ch.offpeak	q06.out.dur.offpeak	q06.out.val.offpeak	q06.in.ch.tot
888	20	4.037063	15.035554	12.511158	10.149751	3.386507
889	21	q06.ch.sms	q06.ch.cc	q07.out.ch.peak	q07.out.dur.peak	q07.out.val.peak
890	22	28.668984	13.747535	3.157588	4.088846	3.830199
891	23	q07.out.dur.offpeak	q07.out.val.offpeak	q07.in.ch.tot	q07.in.dur.tot	q07.ch.sms
892	24	12.907377	10.128761	3.237636	3.192192	28.628610
893	25	q08.out.ch.peak	q08.out.dur.peak	q08.out.val.peak	q08.out.ch.offpeak	q08.out.dur.offpeak
894	26	3.963771	4.131527	4.117853	9.256204	11.536043
895	27	q08.in.ch.tot	q08.in.dur.tot	q08.ch.sms	q08.ch.cc	q09.out.ch.peak
896	28	3.972012	3.953185	20.317293	8.946827	2.927636
897	29	q09.out.val.peak	q09.out.ch.offpeak	q09.out.dur.offpeak	q09.out.val.offpeak	q09.in.ch.tot
898	30	3.646469	10.344301	15.974234	11.674903	2.728376
899	31	q09.ch.sms	q09.ch.cc	y		
899	32	16.048910	13.034136	10.820427		

## A.2 Linear Regression Features Relevance

Listing 9. Linear regression features relevance

905	1	t	value	Pr(> t )
906	2	(Intercept)	15.136	< 2e-16 ***
907	3	tariff.plan	-23.320	< 2e-16 ***
908	4	payment.methodDomiciliazione Bancaria	2.029	0.042513 *
909	5	age	-3.290	0.001004 **
910	6	activation.channel	2.445	0.014485 *
911	7	vas1Y	2.637	0.008389 **
912	8	q01.out.ch.peak	-2.998	0.002725 **
913	9	q01.out.dur.offpeak	3.006	0.002652 **
914	10	q01.out.val.offpeak	-6.760	1.46e-11 ***
915	11	q03.out.val.peak	3.534	0.000411 ***
916	12	q03.in.dur.tot	-4.009	6.15e-05 ***
917	13	q04.out.dur.peak	-3.183	0.001460 **
918	14	q04.out.val.peak	2.320	0.020335 *
919	15	q04.out.ch.offpeak	-4.137	3.55e-05 ***
920	16	q05.out.ch.offpeak	2.194	0.028282 *
921	17	q05.out.dur.offpeak	-2.375	0.017556 *
922	18	q05.out.val.offpeak	3.723	0.000198 ***
923	19	q05.in.ch.tot	-3.401	0.000674 ***
924	20	q05.in.dur.tot	2.231	0.025735 *
925	21	q06.out.dur.peak	-3.182	0.001469 **
926	22	q06.out.val.peak	2.791	0.005260 **
927	23	q06.out.ch.offpeak	-4.226	2.40e-05 ***
928	24	q06.out.val.offpeak	4.772	1.85e-06 ***
929	25	q06.in.dur.tot	2.642	0.008249 **
930	26	q07.out.dur.peak	3.250	0.001160 **
931	27	q07.in.ch.tot	4.881	1.07e-06 ***
932	28	q07.in.dur.tot	-4.645	3.45e-06 ***
933	29	q08.out.ch.peak	2.856	0.004293 **
934	30	q08.out.val.peak	-4.434	9.33e-06 ***
935	31	q08.out.ch.offpeak	-4.099	4.19e-05 ***
936	32	q08.out.dur.offpeak	4.495	7.04e-06 ***
936	33	q08.out.val.offpeak	2.112	0.034720 *
936	34	q09.out.ch.peak	-6.673	2.64e-11 ***

937	35	q09.out.dur.peak	-0.825 0.409148
938	36	q09.out.val.peak	11.179 < 2e-16 ***
939	37	q09.out.ch.offpeak	10.026 < 2e-16 ***
940	38	q09.out.dur.offpeak	18.656 < 2e-16 ***
941	39	q09.out.val.offpeak	-4.800 1.61e-06 ***
942	40	q09.ch.cc	-2.097 0.036019 *

943

### A.3 Linear Regression Log-Transformed Features Relevance

945

946 Listing 10. Linear regression log-transformed features relevance

948	1		Pr(> t )
949	2	(Intercept)	< 2e-16 ***
950	3	tariff.plan	< 2e-16 ***
951	4	payment.methodCarta di Credito	0.007256 **
952	5	payment.methodDomiciliazione Bancaria	0.001023 **
953	6	sexF	0.000623 ***
954	7	sexM	6.81e-09 ***
955	8	age	< 2e-16 ***
956	9	activation.channel	4.42e-07 ***
957	10	vas1Y	2.43e-10 ***
958	11	q01.out.ch.peak	0.004972 **
959	12	q01.in.dur.tot	0.009498 **
960	13	q03.out.ch.peak	0.016320 *
961	14	q04.out.dur.peak	0.002052 **
962	15	q04.out.val.peak	0.012237 *
963	16	q04.out.ch.offpeak	0.031593 *
964	17	q04.ch.cc	0.036098 *
965	18	q05.in.ch.tot	0.041127 *
966	19	q06.out.val.offpeak	0.032902 *
967	20	q07.out.dur.offpeak	0.014165 *
968	21	q07.out.val.offpeak	0.004209 **
969	22	q08.out.ch.peak	0.009375 **
970	23	q09.out.ch.peak	< 2e-16 ***
971	24	q09.out.val.peak	0.000828 ***
972	25	q09.out.ch.offpeak	3.27e-08 ***
973	26	q09.in.ch.tot	0.000294 ***
974	27	q09.ch.sms	0.001544 **

973

### A.4 Lasso Regression Progressive Filtering And Transformations Results

974

- No filtering:  $MSE = 18091358$ ;
- Only log-transform target:  $MSE = 6.200236$ ;
- Filtered incoming calls, SMS and calls to the call center:  $MSE = 18040253$ ;
- Filtered incoming calls, SMS:  $MSE = 18080968$ ;
- Only Value of calls filtered out:  $MSE = 18067089$ ;
- Only amount of calls filtered out:  $MSE = 18118489$ ;
- Incoming calls, SMS, call-center calls, and payment information filtered out:  $MSE = 18035221$ ;
- Incoming calls, SMS, call-center calls, payment information, and sex filtered out:  $MSE = 18036839$ ;
- Incoming calls, SMS, call-center calls, payment information, sex, activation zone, and activation channel filtered out:  $MSE = 18034269$ ;

- 989     • Incoming calls, SMS, call-center calls, payment information, sex, activation zone, activation channel, and age  
 990        filtered out:  $MSE = 18034610$ ;
- 991     • Incoming calls, SMS, call-center calls, payment information, sex, activation zone, activation channel, age, and  
 992        plan information filtered out:  $MSE = 18975290$ ;
- 993     • Incoming calls, SMS, call-center calls, payment information, sex, activation zone, activation channel, age, plan  
 994        information, and value-added services filtered out:  $MSE = 18969265$ ;
- 995     • Incoming calls, SMS, call-center calls, payment information, value-added services, activation zone, and activation  
 996        channel filtered out:  $MSE = 18021270$ ;
- 997     • Incoming calls, SMS, call-center calls, payment information, value-added services, and activation zone filtered  
 998        out:  $MSE = 18022544$ ;
- 999     • Incoming calls, SMS, call-center calls, payment information, value-added services, and activation channel filtered  
 1000        out:  $MSE = 18024845$ ;
- 1001     • Incoming calls, SMS, call-center calls, payment information, value-added service 1, activation zone, and activation  
 1002        channel filtered out:  $MSE = 18035228$ ;
- 1003     • Incoming calls, SMS, call-center calls, payment information, value-added service 2, activation zone, and activation  
 1004        channel filtered out:  **$MSE = 18019454$  (best)**.

1005 Transformations:

- 1006     • Peak and off-peak values aggregated together:  $MSE = 32024591$ ;
- 1007     • Peak and off-peak values aggregated together, with prediction performed on average call time, number of calls,  
 1008        and value over nine months:  $MSE = 32174039$ .

1009 With best filtering:

- 1010     • Log-transform of the target variable only:  $MSE = 6.253648$ ;
- 1011     • Log-transform of the monthly predictors only:  $MSE = 20415157$ ;
- 1012     • Log-transform of both the target variable and the monthly predictors:  $MSE = 4.289544$ .