Data Mining: BeeViva Challenges

MARCO BELLÒ, University of Padua, Italy

All R source code, images, LaTeX sources and datasets (in csv format), can be found at the following repository: https://github.com/mhetacc/DataMiningChallenges/.

ACM Reference Format:
Marco Bellò. 2025. Data Mining: BeeViva Challenges. 1, 1 (December 2025), 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 RICE VARIETIES

1.1 Preliminary Observations

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).

Listing 1. Merge datasets

```
df1 = pd.read_csv("rice_test.csv").round(10)
df2 = pd.read_csv("Rice_Cammeo_Osmancik.csv").round(10)

keys = [col for col in df1.columns]
merged = df1.merge(df2, on=keys, how="inner")
```

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

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

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

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

Manuscript submitted to ACM

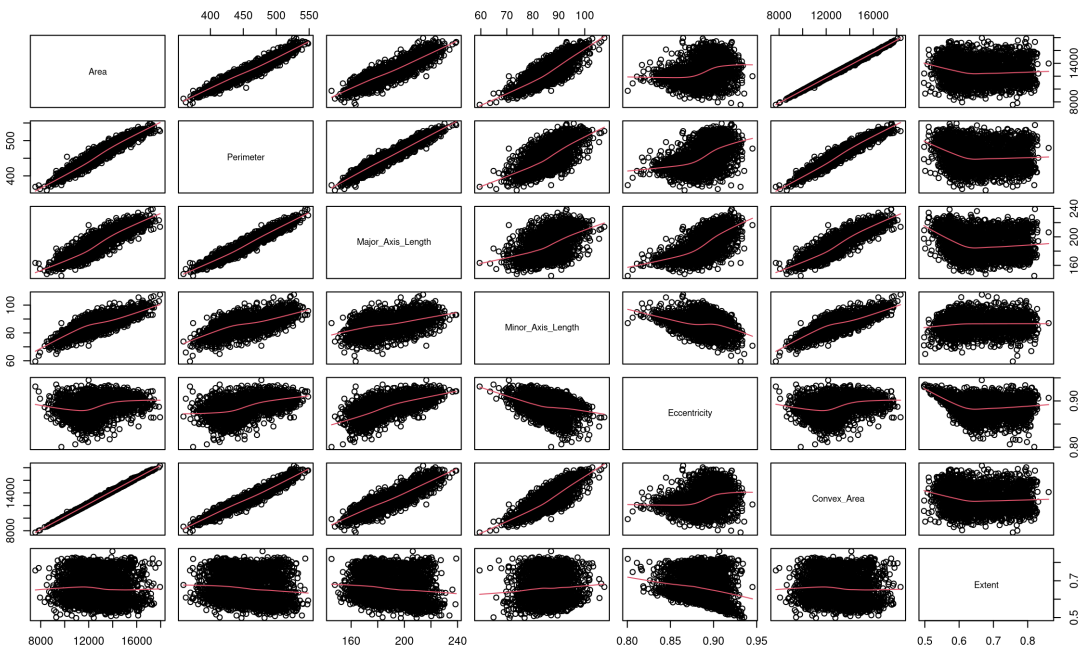

Fig. 1. Scatter Plot for the *rice_train* dataset with LOESS smoothing lines for each class.

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

```
rice.train$Combined <- rowMeans(rice.train[, c("Area","Perimeter","Major_Axis_Length","Convex_Area")])

linear_fit = lm(Class ~
                Minor_Axis_Length
                + Eccentricity
                + Extent
                + Combined,
                data=rice_train,
                subset = train
                )

```

```r
pred <- predict(linear_fit, newdata = rice_train[test, ])
mean((pred - y.test)^2)     # [1] 0.07439239
```

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

Listing 3. Ridge regression on rice dataset

```r
library(glmnet)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 0)
plot(cv.out)
bestlam <- cv.out$lambda.min
cv_mse <- min(cv.out$cvm)     # [1] 0.07924703
cv_rmse <- sqrt(cv_mse)       # [1] 0.2815085
```

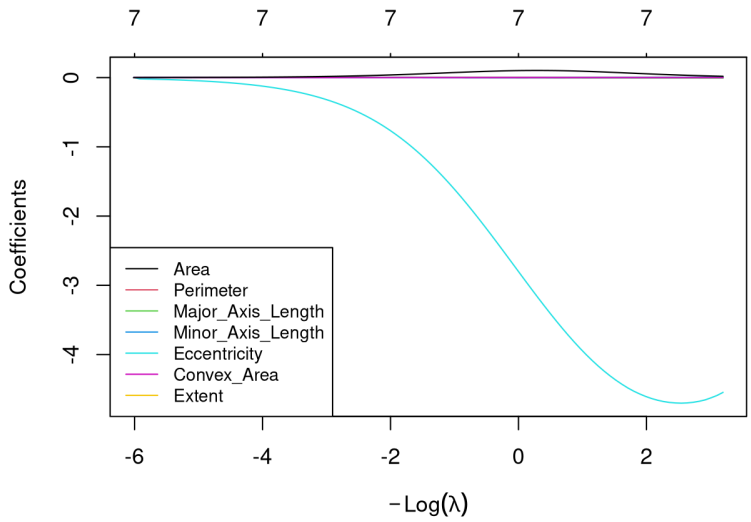

Fig. 3. Ridge coefficients for *rice_train* dataset.

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

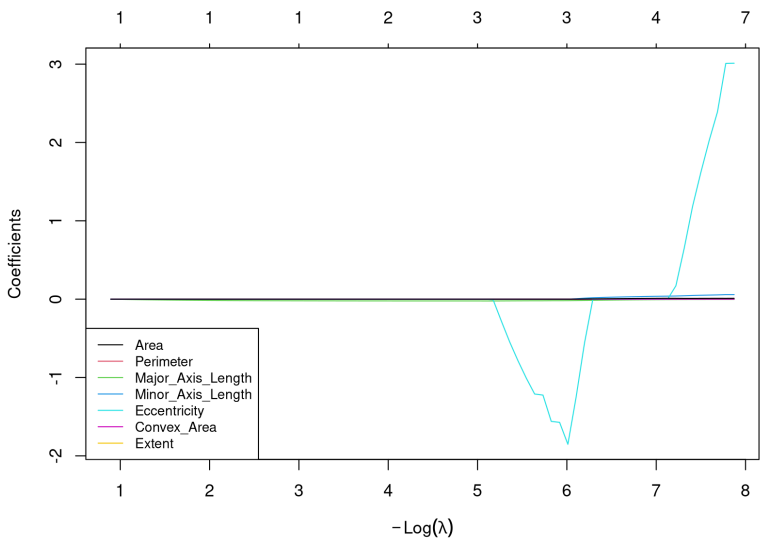

Fig. 4. Lasso coefficients for *rice_train* dataset.

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

```
pcr.fit <- pcr(Class ~ .,
               data = rice_train,
               subset = train,
               scale = TRUE,
               validation = "CV"
               )
```

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 6), 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 $span = 1$

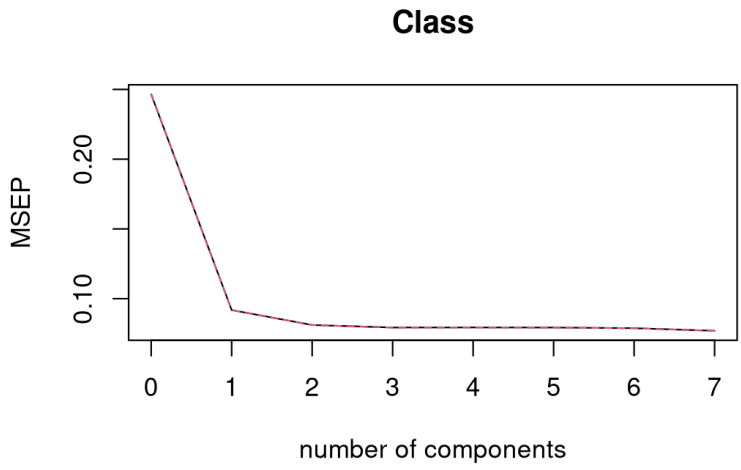

Fig. 5. PCR cross-validation MSE error for target *Class* on *rice_train* dataset.

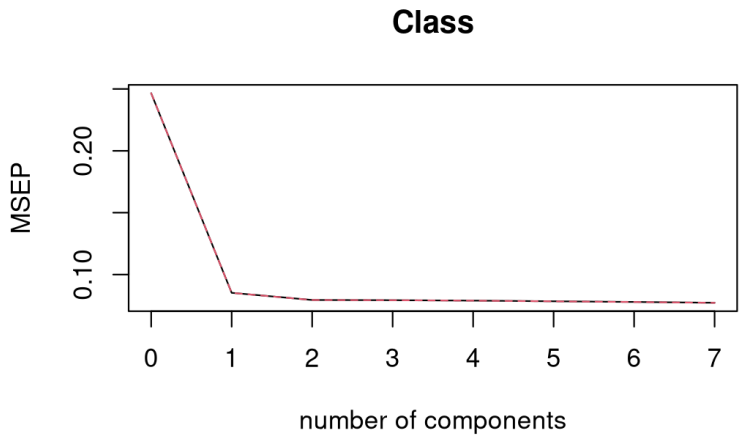

Fig. 6. PLS cross-validation MSE error for target *Class* on *rice_train* dataset.

```r
loess_fit1 <- loess(Class ~ Area
                    + Perimeter
                    + Convex_Area
                    + Major_Axis_Length,
                    data = rice_train,
                    subset = train,
                    span = 0.1
                    )

```

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

```
pred1 <- predict(loess_fit1, newdata = rice_train[test, ])
mean((pred1 - y.test)^2, na.rm = TRUE)    # [1] 46.0226
```

1.2.7 *K-Nearest Neighbors.*