Multivariate Data Analysis - Rice Grain Classification

Joshua Buchanan

Finished on June 6, 2025

Background

Accurate identification of rice grain type is important in agricultural production, quality control, and food processing. Different rice varieties vary in nutritional content, texture, market value, and suitability for specific dishes or industrial applications. Previously, classification would have relied on manual inspection, which is time-consuming, labour-intensive, and prone to human error.

Recent advances allow for the extraction of detailed features from digital images—such as shape, size, texture, and colour—which may be used to automatically classify rice grains. Automating this process increases efficiency and enables scalability in large agricultural operations.

High classification accuracy is particularly important because misclassification can lead to economic losses, compromised product quality, and reduced consumer trust. For example, mixing high-quality basmati rice with lower-quality grains can affect export standards and violate regulatory requirements. Therefore, developing models that can reliably predict rice grain type from image-derived features is of significant importance.

Research Question

Can multivariate classification techniques accurately predict the type of rice grain based on features extracted from digital images?

This study aims to evaluate the performance of several multivariate methods, including Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Random Forest, and XGBoost, in classifying rice grain types. By comparing these models, we seek to identify which approach offers the highest predictive accuracy. Striving for optimal accuracy is essential not only for practical deployment in real-world agricultural and industrial settings, but also to ensure reliability in quality control, reduce misclassification costs, and support data-driven decision-making in rice production and distribution systems.

Methods

To classify rice grain types based on image-derived features, we applied four multivariate machine learning models: Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Random Forest, and XGBoost. These models were selected to compare both linear and non-linear approaches, as well as interpretable versus ensemble-based techniques. Given the decent sample size of 75,000, a 70-30 training-validation split we be used to ensure the model evaluation is accurate to future predictive capabilities. To improve interpretability and potentially enhance performance in linear methods like LDA, we will also test the use of Principal

Component Analysis (PCA) to reduce the dimensionality of the color feature set.

Linear Discriminant Analysis (LDA) is efficient and interpretable but may not work well if the assumptions of equal covariance or multivariate normality are violated.
Quadratic Discriminant Analysis (QDA) relaxes the assumption of equal covariance, allowing more flexibility by fitting a non-linear (quadratic) decision boundary. This makes it more adaptable but also more prone to overfitting.
Random Forest is an ensemble method that builds multiple decision trees using bootstrap samples and random feature selection. It is robust to overfitting, handles non-linear relationships well, and provides variable importance measures. However, it can be less interpretable than linear models and computationally intensive for large datasets.
XGBoost (Extreme Gradient Boosting) is a powerful boosting algorithm that builds trees sequentially to correct previous errors. It often achieves high predictive performance through regularization and efficient optimization. Its drawbacks include longer training times and higher model complexity, which may reduce interpretability.

To evaluate and compare the predictive performance of these models, we will use their confusion matrices to calculate key classification metrics, including average accuracy, total accuracy, precision, recall, and F1 score, providing a comprehensive assessment of both overall and class-specific performance. Evaluating these models side by side will enable us to determine which model offers the best trade-off between accuracy, interpretability, and practical use.

Dataset Details

In this study, a total of 106 features were extracted from 75,000 rice grain images (an equal split of 5 different types: 15,000 of each) using MATLAB:
Morphological Features (12 total) — These include basic geometric properties of each grain, such as area, perimeter, major and minor axis lengths, eccentricity, and solidity.
Shape Features (4 total) — Derived from the morphological features, these include shape factors based on the ratios and combinations of area and axis lengths to capture overall grain form more precisely.
Color Features (90 total) — RGB images were converted into five color spaces (RGB, HSV, L*a*b*, YCbCr, XYZ). For each color channel, six statistical descriptors were computed: mean, standard deviation, skewness, kurtosis, entropy, and wavelet coefficients, capturing both intensity and texture information.

The 5 different types of rice grains are: Arborio, Basmati, Ipsala, Jasmine, and Karacadag.

Data Preprocessing

There were 22 missing values. The variables and how many missing values they had are shown in Table 1. It was decided to remove any rows with missing values of which there were 8 (0.01% of the dataset).

Variable	Number of Missing Values (NAs)
skewB	6
kurtosisB	6
skewCb	3
skewCr	2
kurtosisCb	3
kurtosisCr	2

Table 1: Variables with missing values and their respective NA counts

In order to identify any outliers, PCA is used on the entire dataset and first two principal components are plotted against each other in Figure 1. No clear outliers are seen but separation between classes is very clear here which indicates that our prediction quality should be good - especially for Ipsala which is the most separated group.

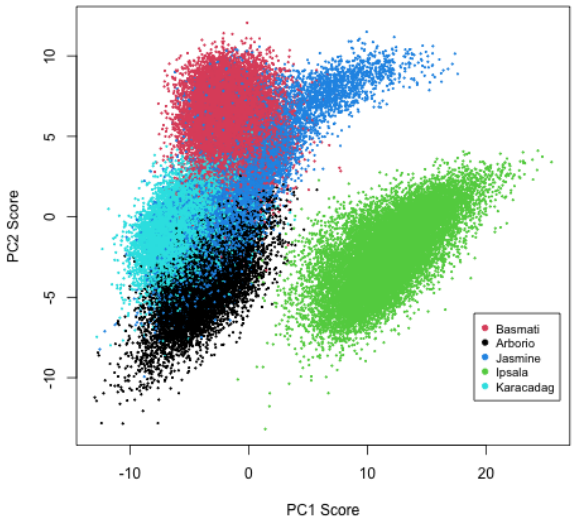

Figure 1: First two principal components of the dataset against each other, coloured by class

Next, PCA was used on the colour variables only in order to decrease dimensionality of them and thus increase interpretability of later models. Six principal components were selected to represent the 90 colour variables - these six account for 85.8% of the variability in the scaled data and will be compared on prediction accuracy with the full model later in the analysis. The pairs plot of these six pricipal components against each other is shown in Figure 2.

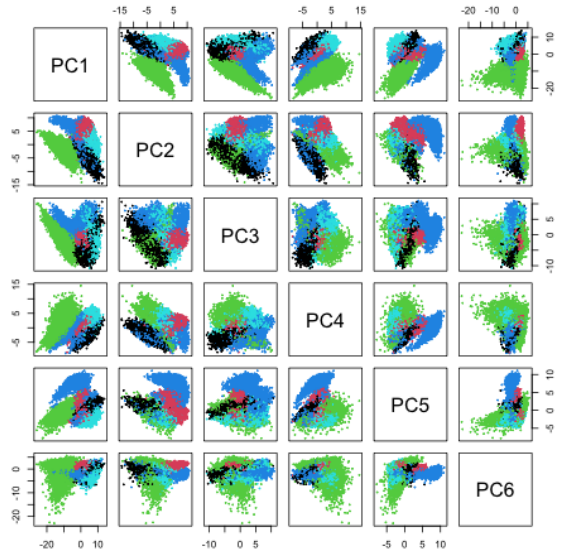

Figure 2: Pairs plot of the first six principal components of the colour variables

Replacing the colour variables with the 6 PC scores (only created using the Training 70% to avoid information leakage), it might now be useful to see how correlated all of our variables are with each other. This can be seen in Figure 3 - there are some variables that are highly correlated with each other. This is likely to affect the effectiveness of QDA and it is likely that LDA will better predict classes compared to QDA on the full dataset because of this.

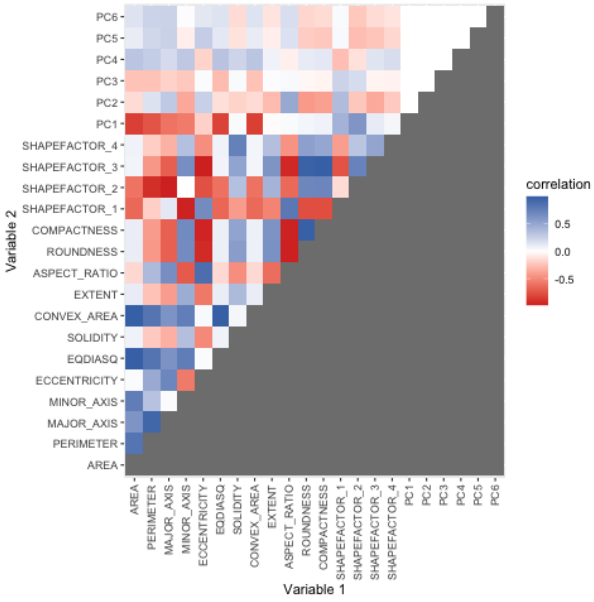

Figure 3: Correlation heatmap of all non-colour variables with six principal components of colours

Results

LDA

Normality checking is shown in Figure 4 which may suggest using QDA as an alternative as it can be more robust to deviation from normality. Equality of covariance was also checked using

Box's M Test which returned a very small p-value (< 0.001) – this means that the covariances matrices between groups are not the same – this also suggests using QDA as an alternative. The confusion matrix for LDA prediction on the 30% validation group using the first six principal component scores for colour variables is shown in Table 4 and with all 90 colour variables in Table 5. The total accuracy using LDA with the PCA-reduced colour variables was 99.32% vs. 99.80% for using all variables.

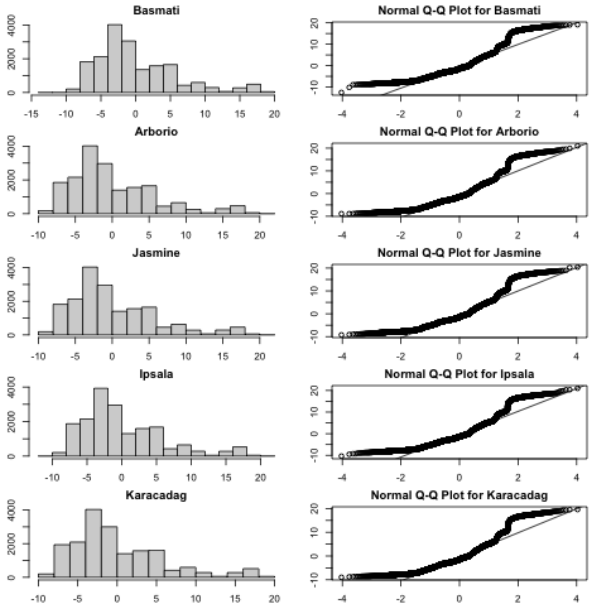

Figure 4: Histograms and Q-Q Plots checking normality

The correlation between the linear discriminant functions and the original variables is shown in Table 2.

Variable	LD1	LD2	LD3	LD4
AREA	-0.27	0.85	0.29	0.07
PERIMETER	0.26	0.79	0.39	0.17
MAJOR_AXIS	0.55	0.67	0.35	0.18
MINOR_AXIS	-0.80	0.49	0.19	0.00
ECCENTRICITY	0.89	0.36	-0.12	0.21
EQDIASQ	-0.27	0.83	0.31	0.11
SOLIDITY	-0.52	-0.13	0.07	-0.19
CONVEX_AREA	-0.27	0.85	0.29	0.07
EXTENT	-0.62	-0.09	-0.09	-0.09
ASPECT_RATIO	0.96	0.07	0.18	0.12
ROUNDNESS	-0.96	-0.18	-0.08	-0.13
COMPACTNESS	-0.96	-0.21	-0.02	-0.13
SHAPEFACTOR_1	0.88	-0.33	-0.14	-0.03
SHAPEFACTOR_2	-0.56	-0.71	-0.24	-0.24
SHAPEFACTOR_3	-0.95	-0.25	0.02	-0.15
SHAPEFACTOR_4	-0.53	-0.13	0.10	-0.05
PC1	0.09	-0.90	-0.12	0.12
PC2	0.49	-0.23	0.46	-0.23
PC3	0.13	-0.08	-0.25	-0.67
PC4	-0.14	0.08	0.53	-0.28
PC5	0.21	0.08	-0.40	0.15
PC6	0.12	0.07	0.32	0.25

Table 2: Correlations of linear discriminant functions with original variables

LD1 primarily distinguishes rice grain types based on shape characteristics. It is strongly influenced by variables such as `ASPECT_RATIO`, `ECCENTRICITY`, and `SHAPEFACTOR_1`, which have large positive loadings, and by `ROUNDNESS`, `COMPACTNESS`, and `SHAPEFACTOR_3`, which have large negative loadings. This indicates that LD1 separates elongated, narrow grains from rounder, more compact ones. **LD2** is dominated by size-related features, including `AREA`, `PERIMETER`, and `EQDIASQ`, with positive loadings, and by `SHAPEFACTOR_2` and the first principal component (`PC1`) with negative loadings. This axis appears to contrast larger grains against more compact or color-influenced profiles. **LD3** and **LD4** contribute less to class separation, with loadings primarily associated with color-derived principal components, suggesting they capture more subtle differences in texture or color intensity among the rice varieties.

QDA

Confusion matrix is shown in Table 6. The prediction accuracy is lower (but very similar) compared with the LDA implementations, 99.76% vs. 99.80%. This indicates that the prediction does not necessarily benefit from non-linear decision boundaries.

Random Forest

Random forest was used with 500 trees which took considerably longer to train compared to LDA and QDA. The confusion matrix can be seen in Table 7. The model prediction accuracy was

even better than LDA (99.91%) but it would need to be determined if this small improvement is worth the drastically increased training time.

XGBoost

XGBoost was used with 100 boosting rounds, a 0.1 learning rate, and a maximum depth of each tree of 4. With these parameters, the model did train slightly faster than the random forest model but as shown in Table 8, it did not perform as well. The total accuracy of this model on the validation set was 99.88%.

Comparison

Table 3 compares all 5 models on different evaluation statistics. The random forest model performed the best at correctly classifying the 30% validation set based on the training set - it achieved the highest overall performance with an average accuracy of 99.96%, precision, recall, and F1 score of 99.91%, and the lowest error rate of 0.04%.

Metric	**LDA (PCA)**	**LDA (Full)**	**QDA**	**Random Forest**	**XGBoost**
Average Accuracy	0.9973	0.9992	0.9990	0.9996	0.9995
Error Rate	0.0027	0.0008	0.0010	0.0004	0.0005
Precision	0.9933	0.9980	0.9976	0.9991	0.9988
Recall	0.9933	0.9980	0.9976	0.9991	0.9988
F1 Score	0.9933	0.9980	0.9976	0.9991	0.9988
Total Accuracy	0.9932	0.9980	0.9976	0.9991	0.9988

Table 3: Model evaluation metrics for rice grain classification

Conclusion

Among the five classification models evaluated (LDA with reduced color variables, LDA with all variables, QDA, Random Forest, and XGBoost) all achieved extremely high performance, with average and total accuracies above 99%. This shows that it is possible to accurately classify these rice grain types using these extracted image features. However, the ensemble methods, particularly **Random Forest** and **XGBoost**, slightly outperformed the linear and quadratic discriminant models. Random Forest achieved the highest overall performance with an average accuracy of 99.96%, precision, recall, and F1 score of 99.91%, and the lowest error rate of 0.04%.

The analysis demonstrates that while linear models like LDA are efficient and interpretable, the more complex tree-based models offer superior classification accuracy.

A key limitation of this analysis is that it assumes the dataset is clean, balanced, and representative. The results may not generalize to unseen or noisier data, and overfitting could still be a concern. Additionally, all models were evaluated using a fixed split rather than cross-validation, which may limit robustness.

For future work, I might apply k-fold cross-validation to better estimate generalization performance and exploring feature selection techniques to reduce redundancy of more highly correlated variables. Also, it might be worthwhile, given more time and scope to more thoroughly check model assumptions and delve deeper into parameter tuning in the tree-based methods.

References

IBM. (2023). *Random forest explained.* IBM. https://www.ibm.com/think/topics/random-forest

Köklü, M. (2020). *Rice dataset repository.* https://www.muratkoklu.com/datasets/

Köklü, M., & Taşdemir, S. (2021). Classification of rice varieties using machine learning algorithms. *Journal of Agricultural Informatics*, https://dergipark.org.tr/en/pub/ankutbd/issue/68966/862482

GeeksforGeeks. (2023a). *XGBoost in R programming.* https://www.geeksforgeeks.org/xgboost-in-r-programming/

GeeksforGeeks. (2023b). *Metrics for machine learning model.* https://www.geeksforgeeks.org/metrics-for-machine-learning-model/

Appendix

Relevant R Code

```
## Read dataset
rice <- read.csv("Rice_MSC_Dataset.csv")

## Where are the NA values
num.na <- apply(rice, 2, FUN = function(x) { sum(is.na(x))})
num.na[num.na > 0]

## Remove rows with NA
rice.filt <- na.omit(rice)

### PCA Plot - should any data points be removed
png(filename="PCAall.png")
all.vars.comp <- prcomp(rice.filt[,1:106], center=T, scale=T)
plot(all.vars.comp$x[,1],all.vars.comp$x[,2],
col=as.numeric(as.factor(rice.filt$CLASS)), pch=19, cex=0.2,
 xlab="PC1 Score", ylab="PC2 Score")
legend(19,-6, legend=unique(rice.filt$CLASS), pch=19, cex=0.8,
 col=unique(as.numeric(as.factor(rice.filt$CLASS))))

## Selecting only the 90 colour features
## and only using the training data to avoid data leakage.
rice.colours <- rice.filt[train.index,17:106]

## Perform PCA
rice.comp <- prcomp(rice.colours, center=T, scale=T)
screeplot(rice.comp)

rice.comp$sdev
## stops dropping much after the sixth one. Could make argument for six PCs.

## Decide number of components with screeplot
plot(rice.comp$sd^2, xlab='component', ylab='variance', main='Rice screeplot')
abline(h=1)

## Cumulative sum of the variance explained by each PC.
cumsum((rice.comp$sd^2)/sum(rice.comp$sd^2))
# The first six PCs account for 85.8% of the variability in the scaled data.

## Pairs plot
png("pcpairs.png")
par(mar = c(0, 0, 0, 0))
pairs(rice.comp$x[,1:6], col=as.numeric(as.factor(rice.filt$CLASS)), pch=19, cex=0.2)

library(ggplot2)
library(reshape2)

corr <- cor(rice.reduced[,1:22])
```

```
# correlation heatmap
png("corplot.png")
diag(corr) <- NA
corr[lower.tri(corr)] <- NA
melted_corr <- melt(corr)
names(melted_corr) = c('Variable 1', 'Variable 2', 'correlation')
ggplot(melted_corr, aes(x=`Variable 1`, y=`Variable 2`, fill=correlation)) +
  geom_tile() +
  scale_fill_gradient2(low="#d73027", high="#4575b4") +
  guides(x = guide_axis(angle=90))

## Let us add the PC variables to a new dataset with the other variables.
rice.reduced <- cbind(rice.filt[,1:16],
## Need PC scores for Test set also
                      predict(rice.comp, rice.filt[,17:106])[,1:6],
                      CLASS=rice.filt[,107])

## Try LDA
library(MASS)

## Checking equality of covariance using Box M's test:
library(biotools)
boxM_result <- boxM(rice.filt[,-107], rice.filt[,107])
boxM_result

## Create training and test indices - 70% training set will be used here.
set.seed(23)
train.index <- sample(1:nrow(rice.filt), size=nrow(rice.filt) * 0.7)

rice.reduced.lda <- lda(rice.reduced[train.index,1:22], rice.reduced[train.index,23])
## predict
rice.reduced.lda.predict <- predict(rice.reduced.lda, rice.reduced[-train.index,-23])

## Normality plot
png("norm.png")
rt <- unique(rice.filt$CLASS)
par(mar = rep(2,4))
par(mfrow=c(5,2))
for (i in 1:5) {
  hist(rice.lda.predict$x[rice.filt$CLASS == rt[i]], main=rt[i])
  qqnorm(rice.lda.predict$x[rice.filt$CLASS == rt[i]],
   main=paste("Normal Q-Q Plot for", rt[i]))
  qqline(rice.lda.predict$x[rice.filt$CLASS == rt[i]])
}

#### Cross-validation to determine number of functions to use
## Create matrix
cv.predict <- matrix(NA, nrow = nrow(rice.reduced), ncol = 4)
colnames(cv.predict) <- c("One dimension", "Two dimensions",
                          "Three dimensions", "Four dimensions")
```

```
## Create 10 folds
set.seed(767) ## so it is the same each time we run
folds <- sample(rep(1:10, length.out = nrow(rice.reduced)))

## 10-fold cross-validation loop
for (k in 1:10) {
  ## Training and test data for the k-th fold
  train_indices <- which(folds != k)
  test_indices <- which(folds == k)

  ## Fit LDA model for training data
  lda.temp <- lda(rice.reduced[train_indices,-23], rice.reduced[train_indices,23])

  ## Predict for test data
  for (i in 4:1) {
    cv.predict[test_indices, i] <- predict(lda.temp,
                                rice.reduced[test_indices,-23], dimen = i)$class
  }
}

# Print predictions for each number of LDA dimensions
colSums(cv.predict==as.numeric(as.factor(rice.reduced$CLASS)))

## Correct proportion classified
colSums(cv.predict==as.numeric(as.factor(rice.reduced$CLASS)))/nrow(rice.reduced)

## Adding fourth dimension is still a 0.27% increase on the third.
## I would suggest leaving all four for maximum accuracy.


## Function for evaluating the prediction quality
measure.classification <- function(cm){
  avg.accuracy <- 0
  err.rate <- 0
  ## Calculate average accuracy and error rate
  for (i in 1:ncol(cm)){
    avg.accuracy <- avg.accuracy + (cm[i,i] + sum(cm[-i,-i])) / sum(cm)
    err.rate <- err.rate + (sum(cm[i,-i]) + sum(cm[-i,i])) / sum(cm)
  }
  avg.accuracy <- avg.accuracy / i
  err.rate <- err.rate / i
  precision <- mean(diag(cm) / rowSums(cm))
  recall <- mean(diag(cm) / colSums(cm))
  F1.score <- 2 * precision * recall/(precision + recall)
  total.accuracy <- sum(diag(cm)) / sum(cm)
  return(list(avg.accuracy=round(avg.accuracy,4),
              err.rate=round(err.rate,4),
              precision=round(precision,4),
              recall=round(recall,4),
              F1.score=round(F1.score,4),
              tot.accuracy=round(total.accuracy,4)))
```

```
}

## LDA reduced Confusion matrix
lda.cm <- table(rice.reduced.lda.predict$class, rice.reduced$CLASS[-train.index],
 dnn=c('Pred','Actual'))
lda.cm
measure.classification(lda.cm)

## do it also for the non-reduced dataset
rice.lda <- lda(rice.filt[train.index,1:106], rice.filt[train.index,107])
rice.lda.predict <- predict(rice.lda, rice.filt[-train.index,-107])
lda.cm.2 <- table(rice.lda.predict$class, rice.filt$CLASS[-train.index],
 dnn=c('Pred','Actual'))
lda.cm.2
measure.classification(lda.cm.2)

## Trying QDA
rice.qda <- qda(rice.filt[train.index,1:106], rice.filt[train.index,107])
rice.qda.predict <- predict(rice.qda, rice.filt[-train.index,-107])
qda.cm.2 <- table(rice.qda.predict$class, rice.filt$CLASS[-train.index],
 dnn=c('Pred','Actual'))
qda.cm.2
measure.classification(qda.cm.2)

## Random Forest Model
library(randomForest)
rf.rice <- randomForest(x=rice.filt[train.index,-107],
 y=as.factor(rice.filt[train.index,107]), ntree=500, mtry=3, importance=T)
rice.rf.predict <- predict(rf.rice, rice.filt[-train.index,-107])
rf.cm <- table(rice.rf.predict, as.factor(rice.filt$CLASS[-train.index]),
 dnn=c('Pred','Actual'))
rf.cm
measure.classification(rf.cm)

#XGBoost Model
library(xgboost)
xg.rice <- xgboost(data=as.matrix(rice.filt[train.index,-107]),
                   label=as.numeric(as.factor(rice.filt[train.index,107]))-1,
                   objective="multi:softprob",
                   num_class=5,
                   nrounds=100,
                   eta=0.1,
                   max_depth=4, verbose=0)
xg.predict <- matrix(predict(xg.rice, as.matrix(rice.filt[-train.index,-107])), nrow=5)
xg.pred.class <- max.col(t(xg.predict))
xg.cm <- table(xg.pred.class, rice.filt$CLASS[-train.index], dnn=c("Pred","Actual"))
measure.classification(xg.cm)
```

Tables and Figures

Predicted \ Actual	Arborio	Basmati	Ipsala	Jasmine	Karacadag
Arborio	4442	0	1	1	44
Basmati	0	4429	0	4	0
Ipsala	0	0	4460	0	0
Jasmine	9	52	1	4489	0
Karacadag	40	0	0	0	4526

Table 4: Confusion matrix for LDA using PCA-reduced colour variables

Predicted \ Actual	Arborio	Basmati	Ipsala	Jasmine	Karacadag
Arborio	4474	0	0	0	3
Basmati	0	4455	0	0	0
Ipsala	0	0	4462	0	0
Jasmine	11	26	0	4494	0
Karacadag	6	0	0	0	4567

Table 5: Confusion matrix for LDA using all variables

Predicted \ Actual	Arborio	Basmati	Ipsala	Jasmine	Karacadag
Arborio	4466	0	1	4	7
Basmati	0	4466	0	1	0
Ipsala	0	0	4460	0	0
Jasmine	15	15	1	4489	0
Karacadag	10	0	0	0	4563

Table 6: Confusion matrix for QDA using all variables

Predicted \ Actual	Arborio	Basmati	Ipsala	Jasmine	Karacadag
Arborio	4482	0	1	1	4
Basmati	0	4477	0	2	0
Ipsala	0	0	4461	0	0
Jasmine	6	4	0	4491	0
Karacadag	3	0	0	0	4566

Table 7: Confusion matrix for random forest model

Predicted \ Actual	Arborio	Basmati	Ipsala	Jasmine	Karacadag
Arborio	4481	0	0	1	5
Basmati	0	4473	0	2	0
Ipsala	0	0	4462	0	0
Jasmine	5	8	0	4491	0
Karacadag	5	0	0	0	4565

Table 8: Confusion matrix for XGBoost