




USING CHEMICAL ANALYSIS TO DETERMINE THE ORIGINS OF WINES

BRANDON SHEEHAN

ERIC SCHUBERT

CSE 5160

- Chemical analysis to determine wine origins in regions around Italy
- Dataset has 3 different classes and 13 predictors
- Multi-class classification problem
 - Class 1, 2, 3
- Results are not binary
- No missing values
- Created in 1991
- Used to test learning algorithms for functionality
- Great tool for finding wines in historically significant regions



Wine

Donated on 6/30/1991

Using chemical analysis to determine the origin of wines

Dataset Characteristics	Subject Area	Associated Tasks
Tabular	Physics and Chemistry	Classification
Feature Type	# Instances	# Features
Integer, Real	178	13

WINE DATASET FROM UCI MACHINE LEARNING REPOSITORY

- Predicting the origins of wines could be beneficial for sommeliers
- Data set could be expanded upon to predict other regions in Italy or other places in the world
- Could include other features such as price to further increase the dimensions of the data set
 - Price per bottle could be used to help classify historically significant producers



Wine

Donated on 6/30/1991

Using chemical analysis to determine the origin of wines

Dataset Characteristics

Tabular

Feature Type

Integer, Real

Subject Area

Physics and Chemistry

Instances

178

Associated Tasks

Classification

Features

13

REAL WORLD APPLICATION

Given the historical significance of the wine dataset with regards to training models, how accurate are the results of testing machine learning methods on it?

RESEARCH QUESTION

Considering this dataset is used for testing machine learning methods, our classification methods should yield extremely accurate results with the low number of data points present in the dataset.

HYPOTHESIS

- 13 predictors were used to classify the output variable
- Each row was assigned a class of either 1, 2, or 3 based upon the predictors

	Y	Alcohol	MalicAcid	Ash	Alcalinity	Magnesium	TotalPhenols	Flavanoids	Nonflavanoids	Proanthocyanins	ColorIn
1	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	
2	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	
3	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	
4	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	
5	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	
6	1	14.20	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	
7	1	14.39	1.87	2.45	14.6	96	2.50	2.52	0.30	1.98	
8	1	14.06	2.15	2.61	17.6	121	2.60	2.51	0.31	1.25	
9	1	14.83	1.64	2.17	14.0	97	2.80	2.98	0.29	1.98	
10	1	13.86	1.35	2.27	16.0	98	2.98	3.15	0.22	1.85	
11	1	14.10	2.16	2.30	18.0	105	2.95	3.32	0.22	2.38	
12	1	14.12	1.48	2.32	16.8	95	2.20	2.43	0.26	1.57	
13	1	13.75	1.73	2.41	16.0	89	2.60	2.76	0.29	1.81	
14	1	14.75	1.73	2.39	11.4	91	3.10	3.69	0.43	2.81	
15	1	14.38	1.87	2.38	12.0	102	3.30	3.64	0.29	2.96	
16	1	13.63	1.81	2.70	17.2	112	2.85	2.91	0.30	1.46	
17	1	14.30	1.92	2.72	20.0	120	2.80	3.14	0.33	1.97	

- 1) Alcohol
- 2) Malic acid
- 3) Ash
- 4) Alcalinity of ash
- 5) Magnesium
- 6) Total phenols
- 7) Flavanoids
- 8) Nonflavanoid phenols
- 9) Proanthocyanins
- 10) Color intensity
- 11) Hue
- 12) OD280/OD315 of diluted wines
- 13) Proline

PREDICTORS AND CLASS

- Before implementing any machine learning method, the dataset had to be prepped
- Each column had to be labeled
- Wine data had to be normalized
- Output variable removed
- Normalized data frame created
- Data set had to be randomized because data was organized in respect to the output variable Y

```
#load the dataset
wine <- read.csv("~/CSE 5160/wine/wine.data", header=FALSE)

#view dataset to check for accuracy
View(wine)

#rename columns
colnames(wine) <- c("Y", "Alcohol", "MalicAcid", "Ash", "Alcalinity", "Magnesium", "TotalPhenols", "Flav

#view column names
names(wine)
summary(wine)
str(wine)
dim(wine)

#randomize rows of the data before knn
set.seed(42)
rows <- sample(nrow(wine))
wine_random <- wine[rows, ]
View(wine_random)

#normalize dataset
normalize <- function(x) {
  return ((x-min(x)) / (max(x) - min(x)))
}

#view data after normalization
summary(wine_random)

#remove output variable
wine.x = subset(wine_random, select = -Y)
summary(wine.x)
dim(wine.x)

#create normalized dataframe
wine_x.normalized = as.data.frame(lapply(wine.x, normalize))
summary(wine_x.normalized)

#start preparation for cross validation of data
```

DATA PRE-PROCESSING

- After processing the data, the rows were randomized which indicates the classes are ready for the machine learning implementations.
 - Randomization was extremely important for securing accurate results
- All columns are now appropriately labeled
- In the bottom table, the data is now normalized and output y is removed
 - This prevents variables for carrying different weights

	Y	Alcohol	MalicAcid	Ash	Alcalinity	Magnesium	TotalPhenols	Flavanoids	Nonflavanoids	Proanthocyanins	ColorIntensity	Hue	DilutedWines	Proline
49	1	14.10	2.02	2.40	18.8	103	2.75	2.92	0.32	2.38	6.200000	1.070	2.75	1060
65	2	12.17	1.45	2.53	19.0	104	1.89	1.75	0.45	1.03	2.950000	1.450	2.23	355
153	3	13.11	1.90	2.75	25.5	116	2.20	1.28	0.26	1.56	7.100000	0.610	1.33	425
74	2	12.99	1.67	2.60	30.0	139	3.30	2.89	0.21	1.96	3.350000	1.310	3.50	985
146	3	13.16	3.57	2.15	21.0	102	1.50	0.55	0.43	1.30	4.000000	0.600	1.68	830
122	2	11.56	2.05	3.23	28.5	119	3.18	5.08	0.47	1.87	6.000000	0.930	3.69	465
178	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.200000	0.610	1.60	560
128	2	11.79	2.13	2.78	28.5	92	2.13	2.24	0.58	1.76	3.000000	0.970	2.44	466
47	1	14.38	3.59	2.28	16.0	102	3.25	3.17	0.27	2.19	4.900000	1.040	3.44	1065
24	1	12.85	1.60	2.52	17.8	95	2.48	2.37	0.26	1.46	3.930000	1.090	3.63	1015
71	2	12.29	1.61	2.21	20.4	103	1.10	1.02	0.37	1.46	3.050000	0.906	1.82	870
100	2	12.29	3.17	2.21	18.0	88	2.85	2.99	0.45	2.81	2.300000	1.420	2.83	406
89	2	11.64	2.06	2.46	21.6	84	1.95	1.69	0.48	1.35	2.800000	1.000	2.75	680
165	3	13.78	2.76	2.30	22.0	90	1.35	0.68	0.41	1.03	9.580000	0.700	1.68	615
110	2	11.61	1.35	2.70	20.0	94	2.74	2.92	0.29	2.49	2.650000	0.960	3.26	680
20	1	13.64	3.10	2.56	15.2	116	2.70	3.03	0.17	1.66	5.100000	0.960	3.36	845
154	3	13.23	3.30	2.28	18.5	98	1.80	0.83	0.61	1.87	10.520000	0.560	1.51	675
114	2	11.41	0.74	2.50	21.0	88	2.48	2.01	0.42	1.44	3.080000	1.100	2.31	434
111	2	11.46	3.74	1.82	19.5	107	3.18	2.58	0.24	3.58	2.900000	0.750	2.81	562

	Alcohol	MalicAcid	Ash	Alcalinity	Magnesium	TotalPhenols	Flavanoids	Nonflavanoids	Proanthocyanins	ColorIntensity	Hue	DilutedWines	Proline
1	0.8078947	0.25296443	0.5561497	0.42268041	0.35869565	0.61034483	0.54430380	0.35849057	0.621451104	0.41979522	0.47967480	0.542124542	0.557774608
2	0.3000000	0.14031621	0.6256684	0.43298969	0.36956522	0.31379310	0.29746835	0.60377358	0.195583596	0.14249147	0.78861789	0.351648352	0.054921541
3	0.5473684	0.22924901	0.7433155	0.76804124	0.50000000	0.42068966	0.19831224	0.24528302	0.362776025	0.49658703	0.10569106	0.021978022	0.104850214
4	0.5157895	0.18379447	0.6631016	1.00000000	0.75000000	0.80000000	0.53797468	0.15094340	0.488958991	0.17662116	0.67479675	0.816849817	0.504279601
5	0.5605263	0.55928854	0.4224599	0.53608247	0.34782609	0.17931034	0.04430380	0.56603774	0.280757098	0.23208191	0.09756098	0.150183150	0.393723252
6	0.1394737	0.25889328	1.0000000	0.92268041	0.53260870	0.75862069	1.00000000	0.64150943	0.460567823	0.40273038	0.36585366	0.886446886	0.133380884
7	0.8157895	0.66403162	0.7379679	0.71649485	0.28260870	0.36896552	0.08860759	0.81132075	0.296529968	0.67576792	0.10569106	0.120879121	0.201141227
8	0.2000000	0.27470356	0.7593583	0.92268041	0.23913043	0.39655172	0.40084388	0.84905660	0.425867508	0.14675768	0.39837398	0.428571429	0.134094151
9	0.8815789	0.56324111	0.4919786	0.27835052	0.34782609	0.78275862	0.59704641	0.26415094	0.561514196	0.30887372	0.45528455	0.794871795	0.561340942
10	0.4789474	0.16996047	0.6203209	0.37113402	0.27173913	0.51724138	0.42827004	0.24528302	0.331230284	0.22610922	0.49593496	0.864468864	0.525677603
11	0.3315789	0.17193676	0.4545455	0.50515464	0.35869565	0.04137931	0.14345992	0.45283019	0.331230284	0.15102389	0.34634146	0.201465201	0.422253923
12	0.3315789	0.48023715	0.4545455	0.38144330	0.19565217	0.64482759	0.55907173	0.60377358	0.08703072	0.76422764	0.571428571	0.091298146	
13	0.1605263	0.26086957	0.5882353	0.56701031	0.15217391	0.33448276	0.28481013	0.66037736	0.296529968	0.12969283	0.42276423	0.542124542	0.286733238
14	0.7236842	0.39920949	0.5026738	0.58762887	0.21739130	0.12758621	0.07172996	0.52830189	0.195583596	0.70819113	0.17886179	0.150183150	0.240370899
15	0.1526316	0.12055336	0.7165775	0.48453608	0.26086957	0.60689655	0.54430380	0.30188679	0.656151420	0.11689420	0.39024390	0.728937729	0.286733238
16	0.6868421	0.46640316	0.6417112	0.23711340	0.50000000	0.59310345	0.56751055	0.07547170	0.394321767	0.32593857	0.39024390	0.765567766	0.404422254
17	0.5789474	0.50592885	0.4919786	0.40721649	0.30434783	0.28275862	0.10337553	0.90566038	0.460567823	0.78839590	0.06504065	0.087912088	0.283166904
18	0.1000000	0.00000000	0.6096257	0.53608247	0.19565217	0.51724138	0.35232068	0.54716981	0.324921136	0.15358362	0.50406504	0.380952381	0.111269615
19	0.1131579	0.59288538	0.2459893	0.45876289	0.40217391	0.75862069	0.47257384	0.20754717	1.000000000	0.13822526	0.21951220	0.564102564	0.202567760

DATA AFTER PRE-PROCESSING

- The first machine learning method we chose to apply was K-Nearest-Neighbors (KNN)
- Before applying the method, 5-fold cross validation was applied
- Each chunk had approximately 35 data points because the total dataset had 178 rows
- The training set had 144 entries and the test set had 34 entries
- KNN was expanded to account for k-values between 1 and 21

```
#start preparation for cross validation of data
samplesize <- nrow(wine_random)

chunksize <- round(samplesize/5)

cv.k <- 5
cv.error.k <- rep(0, cv.k)

#function for cross validation for knn
for (n in 1:cv.k) {
  vset.start <- (n-1)*chunksize + 1
  if (n == cv.k){
    vset.end <- samplesize
  }
  else {
    vset.end <- n*chunksize
  }
  vset.range <- vset.start:vset.end

  #training and test sets
  train.x = wine_x.normalized[-(vset.range), ]
  test.x = wine_x.normalized[vset.range, ]
  train.y = wine_random[-(vset.range), 1]
  test.y = wine_random[vset.range, 1]

  #set seed
  set.seed(123)

  #knn function
  for (x in 1:21) {
    wine.pred <- knn(train.x, test.x, train.y, k = x)
    cv.error.k[x] <- mean(wine.pred != test.y)
  }
}

#error rates
cv.error.k
error = mean(cv.error.k)
```

Confusion Matrix

	Test		
Prediction	1	2	3
1	9	0	0
2	0	12	0
3	0	0	13

K-NEAREST-NEIGHBORS IMPLEMENTATION

- Based upon the confusion matrix, appropriate learning metrics were applied
- Accuracy
- Precision
- Sensitivity
- Specificity
- Error Rate
- ROC Curves cannot be applied to KNN

Confusion Matrix

Prediction	Test		
	1	2	3
1	9	0	0
2	0	12	0
3	0	0	13

K-NEAREST-NEIGHBORS ANALYSIS

- Based upon the metrics, we can conclude that our KNN implementation was extremely accurate
- This was to be expected based upon our initial hypothesis that this data is used for testing new machine learning methods
- It makes sense that the KNN method would yield superior results
- There were no incorrect predictions in the confusion matrix that was output
- Test data set was 100% accurate in predicting true positives and true negatives based upon the sensitivity and specificity

Performance Metrics

Accuracy	100%
Error Rate	0%
Precision – Class 1	100%
Precision – Class 2	100%
Precision – Class 3	100%
Sensitivity – Class 1	100%
Sensitivity – Class 2	100%
Sensitivity – Class 3	100%
Specificity – Class 1	100%
Specificity – Class 2	100%
Specificity – Class 3	100%

K-NEAREST-NEIGHBORS ANALYSIS

- The second machine learning method we chose to apply was Logistic Regression
- Before applying the method, the data was normalized, randomized, and separated into training and testing sets.
- Using sample there was a rough 60 40 split between training and testing sets
- Training had 120 objects, while testing had 58 objects.
- Since our dataset had 3 classes, we had to go with a multinomial approach rather than a binomial approach.
- Class 1 of our dataset was the reference level that was used as the response variable.

```
# Rename columns for better readability
colnames(wine) <- c("Y", "Alcohol", "MalicAcid", "Ash", "Alcalinity", "Mg")
wine$Y <- as.factor(wine$Y)

# Extracting numerical columns for normalization (commented out for now)
numerical_cols <- names(wine)[sapply(wine, is.numeric)]

# Normalize numerical columns (commented out for now)
wine[, numerical_cols] <- scale(wine[, numerical_cols])

# Randomize rows of the data before cross-validation
set.seed(42)
rows <- sample(nrow(wine))
wine_random <- wine[rows, ]
ind <- sample(2, nrow(wine_random), replace = TRUE, prob = c(0.6, 0.4))
# Create training and testing sets
training <- wine_random[ind == 1,]
testing <- wine_random[ind == 2,]

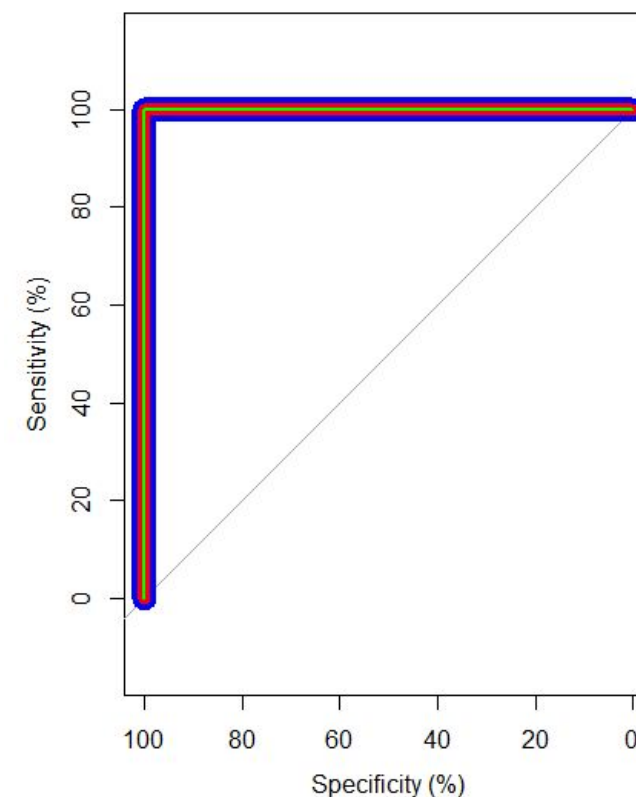
# Set reference level for the response variable in the training set
training$Y <- relevel(training$Y, ref="1")
testing$Y <- relevel(testing$Y, ref="1")
# Build multinomial logistic regression model
mymodel <- multinom(Y ~ ., data = training)
summary(mymodel)
```

MULTINOMIAL LOGISTIC REGRESSION IMPLEMENTATION

- Based upon the confusion matrix, appropriate learning metrics were applied
- Accuracy
- Precision
- Sensitivity
- Specificity
- Error Rate
- ROC Curves included with Logistic Regression

```
Training Set Metrics:  
> cat("Accuracy: ", accuracy_train, "\n")  
Accuracy: 1  
> cat("Precision: ", mean(precision_train), "\n")  
Precision: 1  
> cat("Recall: ", mean(recall_train), "\n")  
Recall: 1  
> cat("F1-score: ", mean(f1_train), "\n")  
F1-score: 1
```

```
Testing Set Metrics:  
> cat("Accuracy: ", accuracy_test, "\n")  
Accuracy: 0.9827586  
> cat("Precision: ", mean(precision_test), "\n")  
Precision: 0.9777778  
> cat("Recall: ", mean(recall_test), "\n")  
Recall: 0.9876543  
> cat("F1-score: ", mean(f1_test), "\n")  
F1-score: 0.9822164
```



MULTINOMIAL LOGISTIC REGRESSION ANALYSIS

- Based upon the metrics, we can conclude that our Multinomial Logistic Regression implementation was extremely accurate
- This was to be expected based upon our initial hypothesis that this data is used for testing new machine learning methods.
- The website we got the data from stated that Logistic Regression typically falls between 80% - 100% accurate
- There were no incorrect predictions in the Training set's confusion matrix, and there was only a single inaccuracy in the Testing set's Confusion matrix.

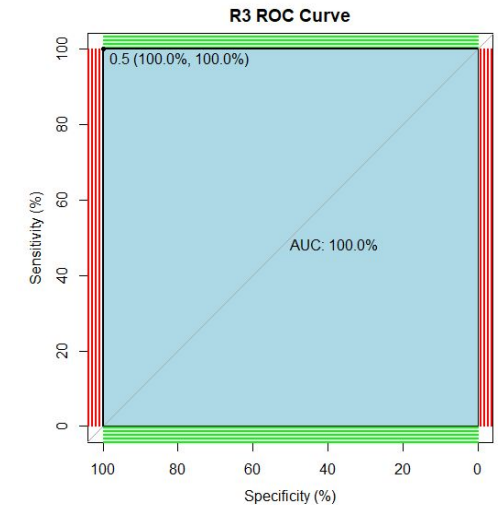
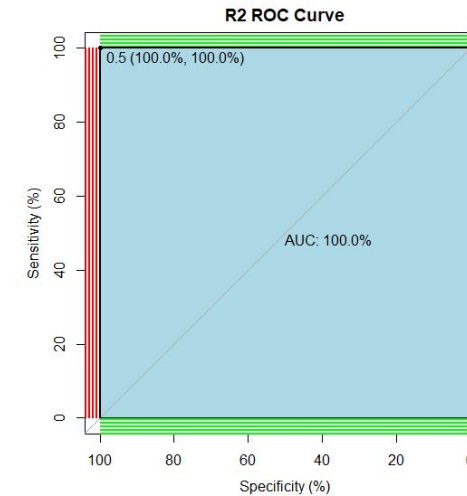
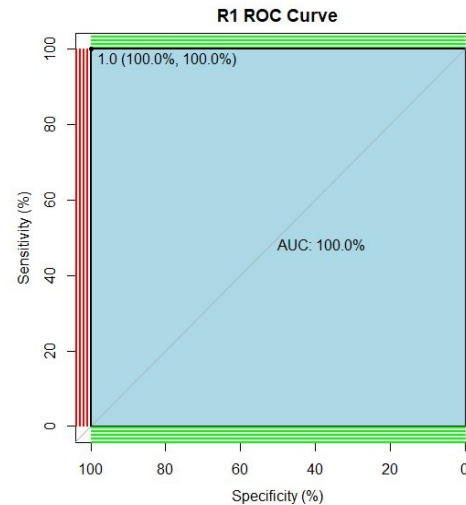
Performance Metrics

Training Confusion Matrix

p	1	2	3
1	44	0	0
2	0	45	0
3	0	0	31

Testing Confusion Matrix

p1	1	2	3
1	14	0	0
2	1	26	0
3	0	0	17



MULTINOMIAL LOGISTIC REGRESSION ANALYSIS

- There were multiple limitations to our machine learning implementations with respect to the wine dataset
 - The dataset only has 178 entries
 - With 80% of the data going to the training set, this only leaves 34 data points for the test set
 - If there were more rows in the data, there would likely be a greater error rate
 - We could of also tried different methods for cross-validation
 - We only tested 2 machine learning methods. For improvement in the future, we could try more methods to try to yield better results

LIMITATIONS

-
- KNN vs Logistic Regression
 - KNN yielded better results than Logistic Regression, but only slightly.
 - KNN was able to get perfect accuracy for both the training and the testing sets compared to Logistic regression, which had one inaccuracy in its test set.
 - Overall, both performed well and could be considered as reasonable machine learning methods for this dataset.
 - If the dataset had more entries and/or more classes both would most likely perform very similarly once more.
 - The last key difference was the test and train splits for KNN being 20/80 compared to Logistic Regression 40/60 which could impact accuracy.

MODEL COMPARE

In conclusion, our exploration of the UCI wine dataset through the application of K-Nearest Neighbors (KNN) and Multinomial Logistic Regression has yielded promising results, shedding light on the efficacy of these machine learning methods within the context of this historical dataset. The remarkable 100% accuracy achieved by the KNN model underscores its robust performance in capturing intricate patterns inherent in the dataset, while the 98% accuracy attained by Multinomial Logistic Regression further attests to the method's capability in discerning complex relationships. These outcomes not only showcase the suitability of these models for the wine dataset but also emphasize the dataset's enduring relevance as a reliable benchmark for training and evaluating machine learning algorithms.

CONCLUSION



THANK YOU!

DECEMBER 11, 2023