Letter Classification

Brent Mabey
David Lowe

Introduction

- Ancient, historic, and genealogical records have come to light in recent years.
- Transcribing handwritten records can be slow, and sometimes inaccurate.
- Digital scanning allows the digitization of handwritten records, but does not solve the problem of accurately and quickly translating each handwritten character to a recognized letter.
- Computers have the ability to track patterns of letters and words that can help quickly and accurately predict large texts.

Goal

Use computer generated characteristics of scanned handwritten letters to accurately predict the correct associated English letter.

Data

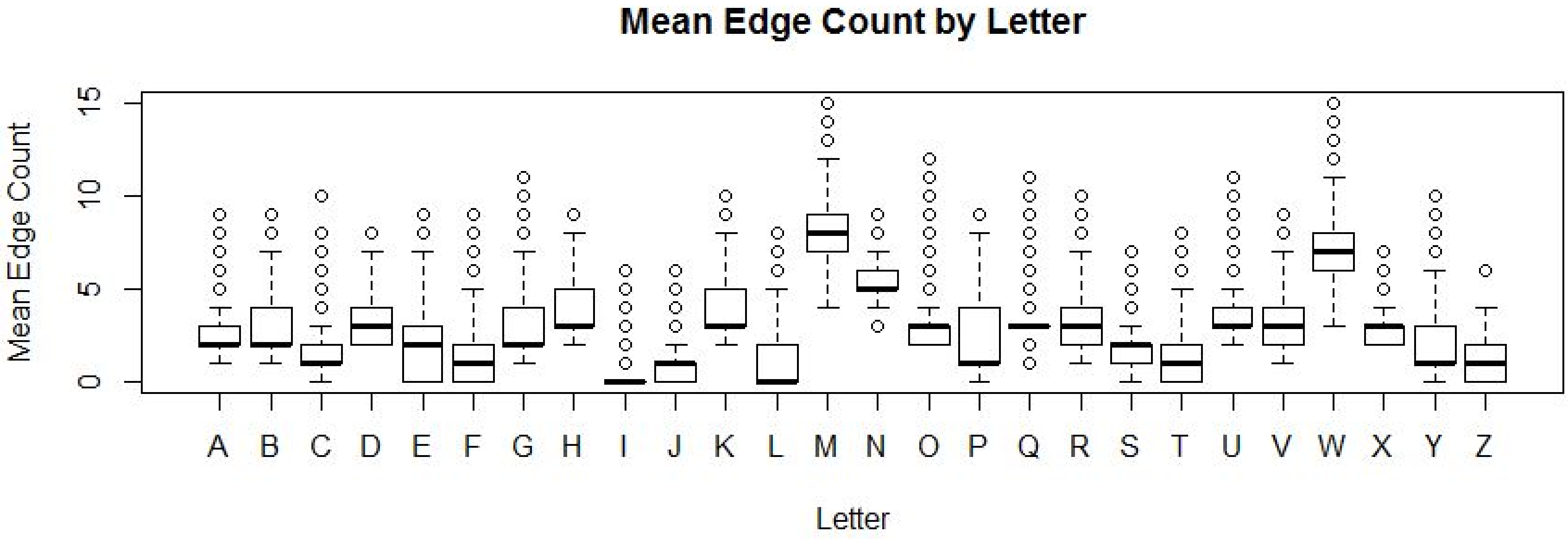
- 19,999 person-verified English letters
- 16 computerized character attributes (number of pixels, height of character, etc.)
- Counts are fairly even between the 26 English characters.

Letter Count

A	B	C	D	E	F	G	H	I	J	K	L	M
789	766	736	805	768	775	773	734	755	747	739	761	792

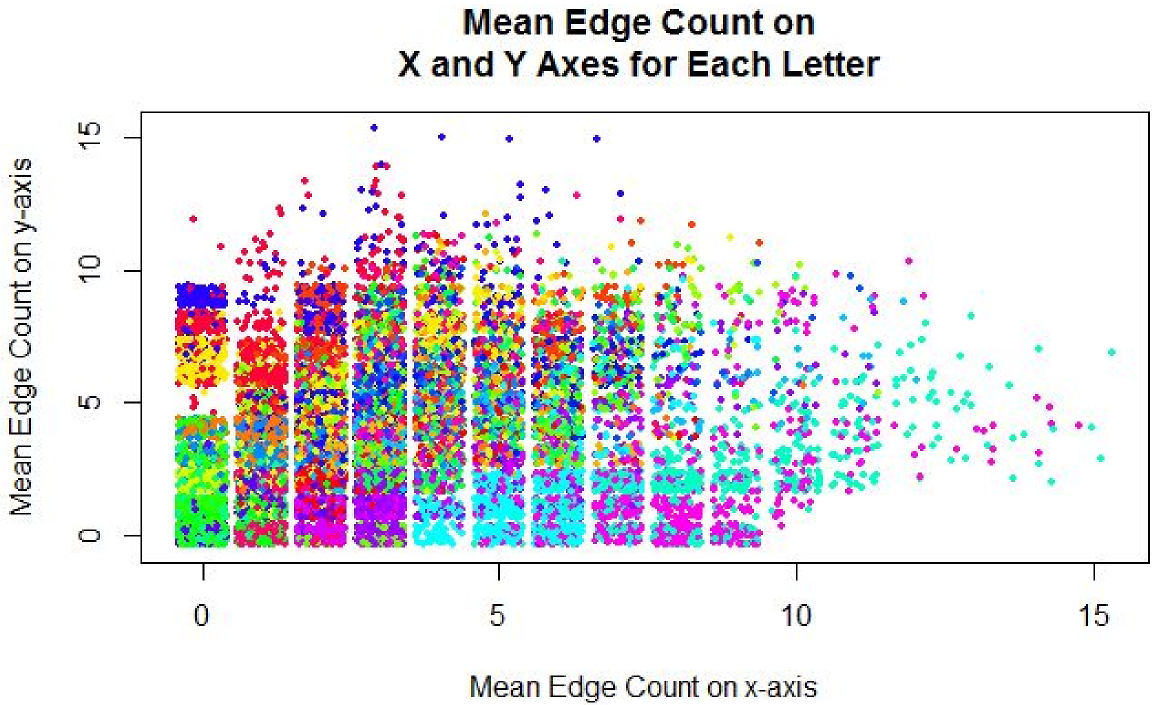
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
783	753	803	783	758	748	795	813	764	752	787	786	734

Data Exploration



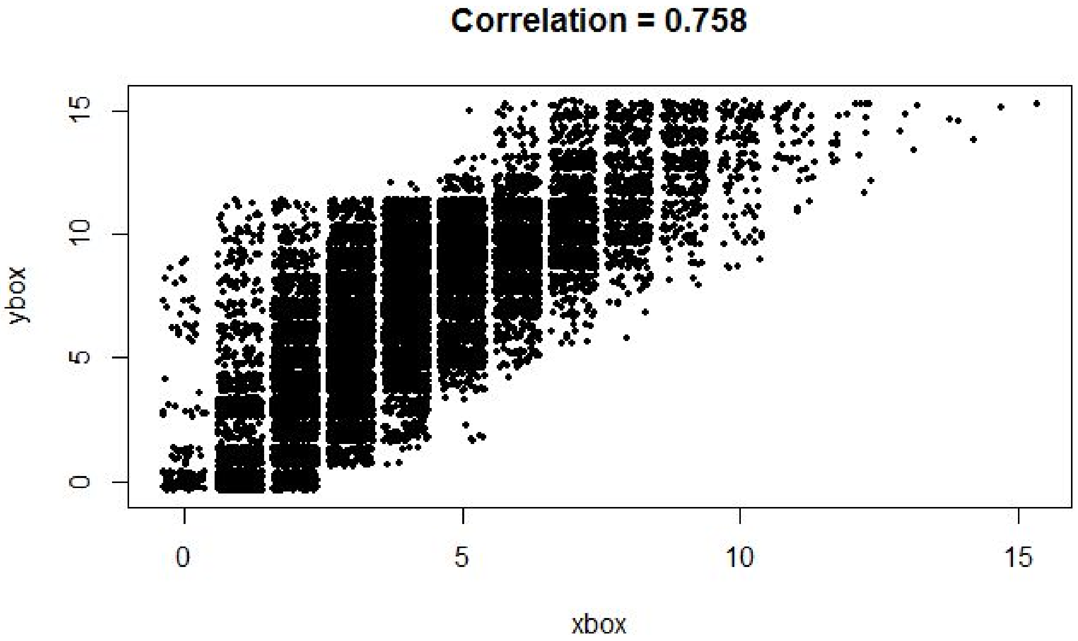
Data Exploration: Letter Clumping

- Some letter characteristics seem to clump letters together better than others
- Each letter is shown in a different color



Data Exploration: Collinearity

- Several variables show strong collinearity
- Many variables are convolutions of other variables
- Correlation between variables must be accounted for

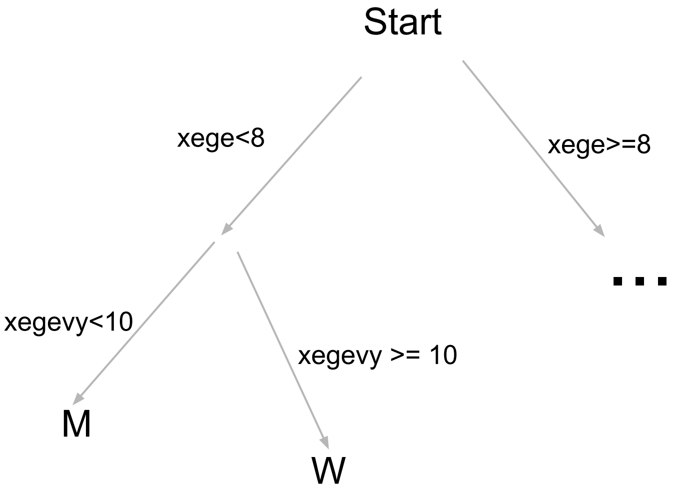


Methods

- Random forest classification algorithm
- Non-parametric ensemble method
- Cross-validation
  - Split data into two sets
  - Use first set to create random forest
  - Use second set to cross-validate, optimize parameters (number of trees, number of variables)
- Final model run with all of the data

Classification Trees

- Like a field guide for a data set
- Split explanatory variable into two groups
- Follow path (‘branches’) of splitting until you reach the end (‘terminal node’)
- Output prediction at the terminal node
- Splitting groups determined by best classification among all variables with fewest branches



Random Forest Algorithm

- Ensemble method: combine multiple model predictions
- Bagging: take bootstrapped sample, aggregate results
  - Accounts for importance of variables
- Make tree from bootstrapped data considering only a random sample of variables at a time
  - Keeps trees from being too similar
- Repeat many times
- Aggregate predictions from each tree
  - For regression, predict using mean/median
  - For classification, predict using mode

Justification

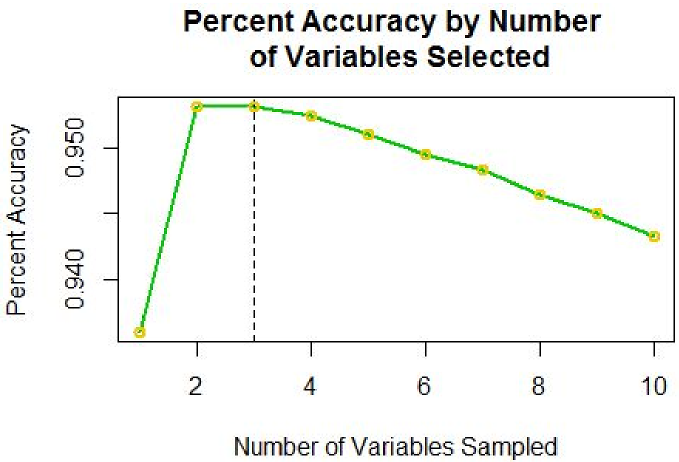
- No distributional assumptions are needed when using the random forest algorithm.
- By randomly selecting a subset of explanatory variables to use in each tree, the random forest algorithm “decorrelates” the variables, and the trees themselves.
- Bagging indicates the importance of each variable
- This allows us to classify the data into multiple categories (English letters)

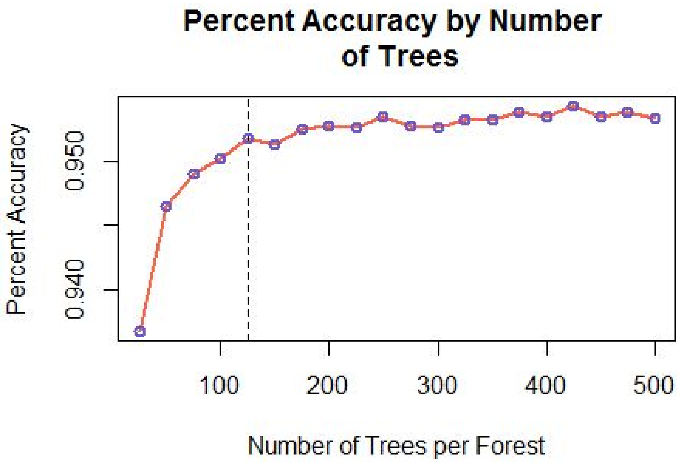
Model Tuning/ Performance Evaluation

- Bootstrap Settings:
  - 100 random forests built
  - 125 trees per forest
  - Three explanatory variables chosen at random for each tree

Avg. Accuracy	2.5%	97.5%
0.9529	0.9509	0.9547

Avg. OOB Error Rate	2.5%	97.5%
0.0546	0.0524	0.0570

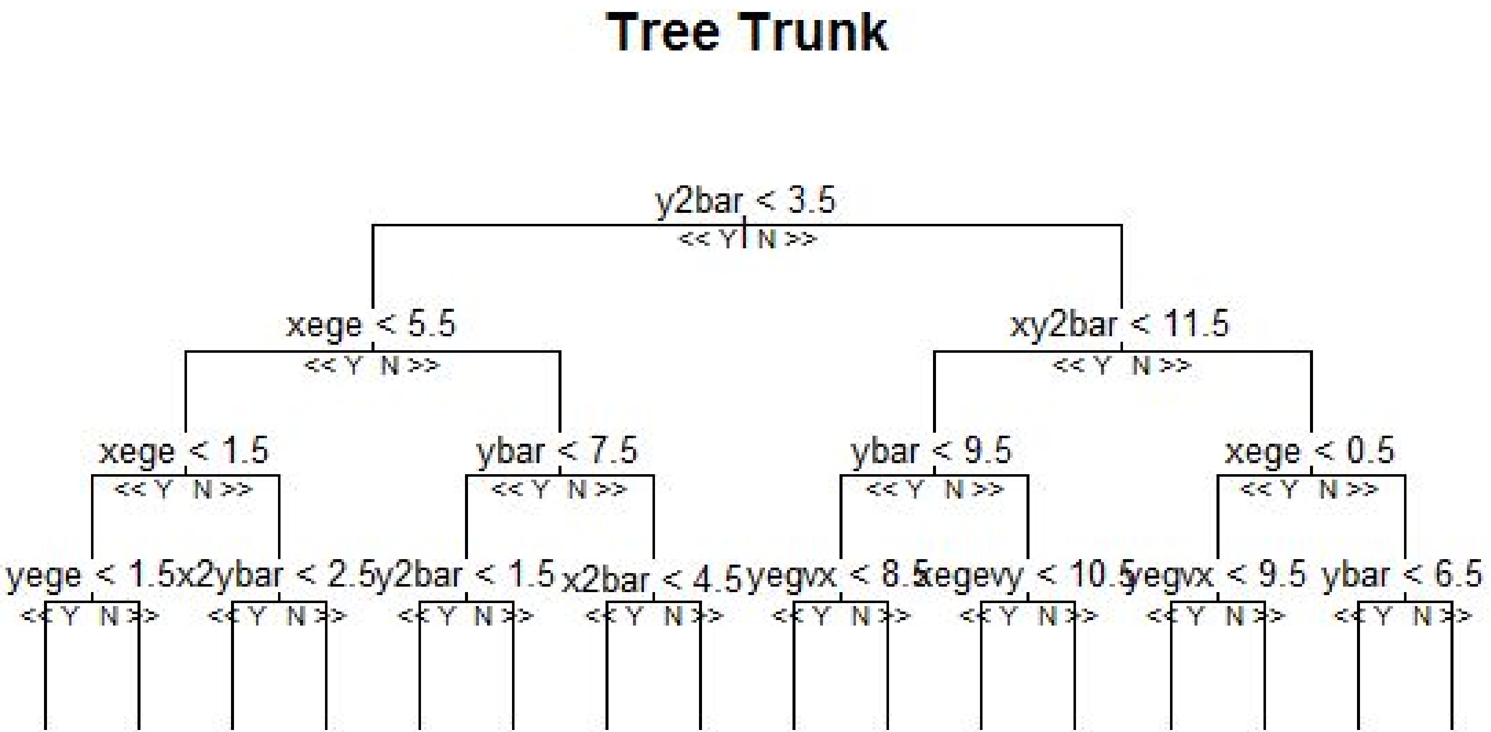




Confusion Matrix

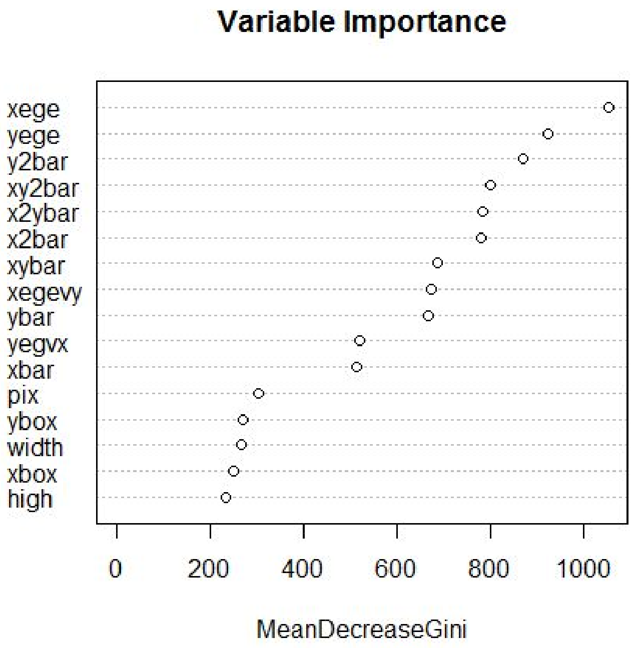
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	Error Rate
A	385	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	2	0	0.0153
B	0	354	0	2	1	0	1	1	0	0	0	0	0	2	0	0	1	7	0	0	1	4	0	2	1	1	0.0635
C	0	0	343	0	3	1	4	1	0	0	1	1	0	0	6	0	2	0	0	0	0	0	0	0	0	0	0.0525
D	0	2	0	396	0	0	0	4	0	0	0	0	0	1	2	0	0	1	1	0	0	0	0	0	0	0	0.0270
E	0	1	1	0	377	0	4	0	0	0	0	2	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0.0308
F	0	3	0	0	2	354	0	1	0	1	0	0	0	2	0	7	0	0	2	5	0	0	0	2	2	0	0.0709
G	0	2	1	1	3	0	392	1	0	0	0	0	1	0	2	0	1	1	1	0	0	0	1	1	0	0	0.0392
H	2	5	0	11	0	0	0	308	0	0	9	1	1	0	4	3	2	13	0	0	1	0	0	1	0	0	0.1468
I	0	2	0	0	0	1	0	0	355	10	0	0	0	0	0	2	0	0	0	1	0	0	0	1	0	0	0.0457
J	1	1	0	0	1	1	0	1	8	375	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	2	0.0434
K	0	0	1	1	3	1	0	8	0	0	316	0	0	0	0	0	0	9	0	0	1	0	0	5	0	0	0.0841
L	0	1	1	0	5	0	4	0	0	1	1	353	0	0	0	0	1	1	0	0	0	0	0	2	0	0	0.0459
M	1	0	0	0	0	0	0	1	0	0	1	0	362	6	0	0	0	0	0	0	0	0	2	0	0	0	0.0295
N	0	1	0	4	0	0	0	2	0	0	1	1	1	370	4	0	0	3	0	0	1	2	1	0	0	0	0.0537
O	0	1	2	7	0	0	1	0	0	0	0	0	0	0	363	1	5	1	0	0	1	0	1	0	0	0	0.0522
P	0	3	0	2	0	9	0	0	0	0	0	0	0	0	0	402	2	0	0	0	0	1	0	0	1	0	0.0429
Q	1	1	0	1	0	0	1	0	0	0	0	0	0	0	10	0	387	1	0	0	0	0	0	1	0	0	0.0397
R	0	7	0	1	0	0	0	2	0	0	4	1	0	2	0	0	2	358	0	0	0	0	0	0	0	0	0.0504
S	2	9	0	0	2	1	0	2	0	0	0	0	0	0	0	0	2	1	344	0	0	0	0	0	0	0	0.0523
T	0	0	2	0	0	3	0	0	0	0	1	0	0	0	0	0	0	0	1	378	2	1	0	2	5	1	0.0455
U	1	0	0	1	0	0	0	0	0	0	1	0	2	0	2	0	1	0	0	0	400	1	0	0	0	0	0.0220
V	0	8	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	0	0	0	0	380	3	0	2	0	0.0452
W	0	0	0	0	0	0	1	0	0	0	0	0	3	1	0	0	0	1	0	0	1	0	363	0	0	0	0.0189
X	0	2	0	1	1	0	0	0	0	0	5	0	0	0	0	0	0	2	0	0	0	0	0	379	0	0	0.0282
Y	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	3	2	2	0	0	390	0	0.0250
Z	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	5	1	0	0	0	0	0	0	0	362	0.0216

Example Tree



Results: Variable Importance

- Gini Index
  - Measures how well a tree splits data into ‘pure’ regions
  - Minimized when data are perfectly classified
  - Larger decrease in Gini=more important
- Mean edge counts
- X and Y variances



Results

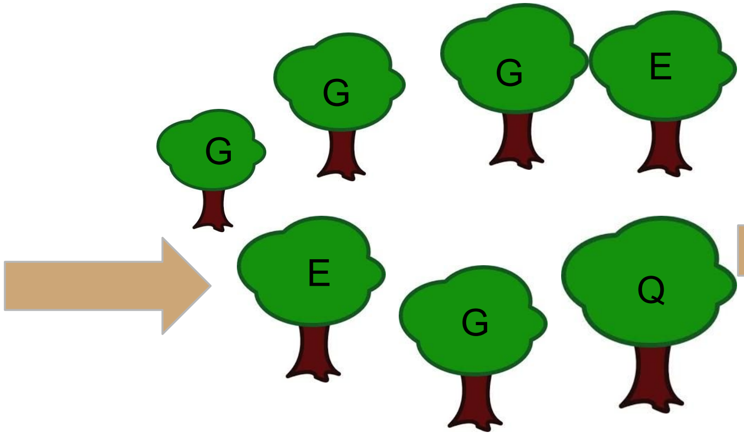
- I and J
- H, K, and R
- F and P
- O and Q,D

.	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	788	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	735	0	1	2	1	1	2	0	0	0	0	1	1	0	1	0	6	1	0	2	10	0	2	0	0
C	0	0	708	0	6	1	9	0	0	0	0	0	0	0	7	0	4	0	0	0	0	0	1	0	0	0
D	1	3	0	778	0	1	1	6	0	0	0	0	0	4	6	0	1	3	0	0	0	0	0	0	0	1
E	0	1	5	0	732	0	10	0	0	0	4	3	0	0	0	0	2	0	2	0	0	0	0	5	0	4
F	0	8	0	3	1	734	0	1	0	0	1	0	0	1	0	13	1	0	2	7	0	1	1	0	1	0
G	1	2	2	6	3	1	745	1	0	1	0	0	1	0	1	1	6	0	0	0	0	1	1	0	0	0
H	1	5	2	11	1	1	2	665	0	0	13	0	2	1	6	1	2	16	0	0	3	1	0	1	0	0
I	0	3	0	1	0	3	0	0	715	25	0	0	0	1	0	3	0	0	1	1	0	0	0	1	0	1
J	1	1	0	1	1	3	0	2	20	710	0	3	0	2	2	0	0	0	0	0	0	0	0	0	0	1
K	0	2	0	0	4	0	0	12	0	0	697	0	0	0	0	0	0	16	0	0	2	0	0	6	0	0
L	0	0	1	0	5	0	3	0	0	1	1	740	0	0	0	0	5	3	0	1	0	0	0	1	0	0
M	0	0	0	0	1	0	0	0	0	0	0	0	780	4	2	0	0	0	0	0	0	3	2	0	0	0
N	1	4	0	6	0	1	0	5	0	0	0	0	5	746	6	0	0	8	0	0	0	1	0	0	0	0
O	0	2	1	14	0	0	2	0	0	0	0	0	0	0	725	1	5	2	0	0	1	0	0	0	0	0
P	0	3	0	2	1	19	4	0	0	0	0	0	0	0	0	769	1	1	0	1	0	1	0	0	1	0
Q	1	2	0	1	0	0	1	0	0	0	0	1	0	0	14	0	758	2	1	0	0	0	1	0	0	1
R	0	12	1	1	0	0	0	3	0	0	9	0	0	6	0	1	3	721	0	0	0	0	0	1	0	0
S	0	6	0	1	1	2	0	3	0	1	0	0	0	0	1	0	2	2	729	0	0	0	0	0	0	0
T	0	2	2	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	778	0	1	0	1	7	1
U	1	1	0	0	0	0	0	2	0	0	1	0	5	0	3	0	1	0	0	0	799	0	0	0	0	0
V	0	8	0	0	0	1	0	0	0	0	0	0	2	1	0	2	1	3	0	0	0	740	4	0	2	0
W	0	0	0	0	0	0	1	0	0	0	0	0	3	1	1	0	1	0	0	0	1	1	743	0	0	0
X	0	1	0	1	4	0	0	0	0	0	4	0	0	0	0	0	1	0	0	0	0	0	0	776	0	0
Y	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	5	0	0	770	0
Z	0	1	0	1	3	1	0	0	0	0	0	0	0	0	0	0	8	0	2	0	0	0	0	0	0	718

Application

- Take characteristics of box with unknown letter
- Each tree uses characteristics to classify box as a letter
- Pick whichever letter gets chosen most frequently

xbox	4
ybox	9
width	6
high	7
pix	6
xbar	7
ybar	8
x2bar	6
y2bar	2
xybar	6
x2ybar	5
xy2bar	11
xege	4
xegevy	8
yege	7
yegvx	8



A	3
B	5
C	5
E	23
G	429
K	3
N	3
O	1
P	1
Q	12
R	3
S	5
U	3
V	1
W	2
Y	1

G

Conclusions

Strengths

- High predictive accuracy (~95%)
- More robust than single classification tree
- Everything cross-validated
- Easy implementation

Weaknesses

- Harder to interpret than in regression or single trees
  - More computationally intensive, too
- Difficulty distinguishing certain letters
  - H error: 14%
  - K error: 8%

Conclusions: Future Research

- Include lowercase letters, and characters written in different scripts and by different scribes so as to generalize the model.
- Incorporate surrounding letters to discern characters by probable letter association.