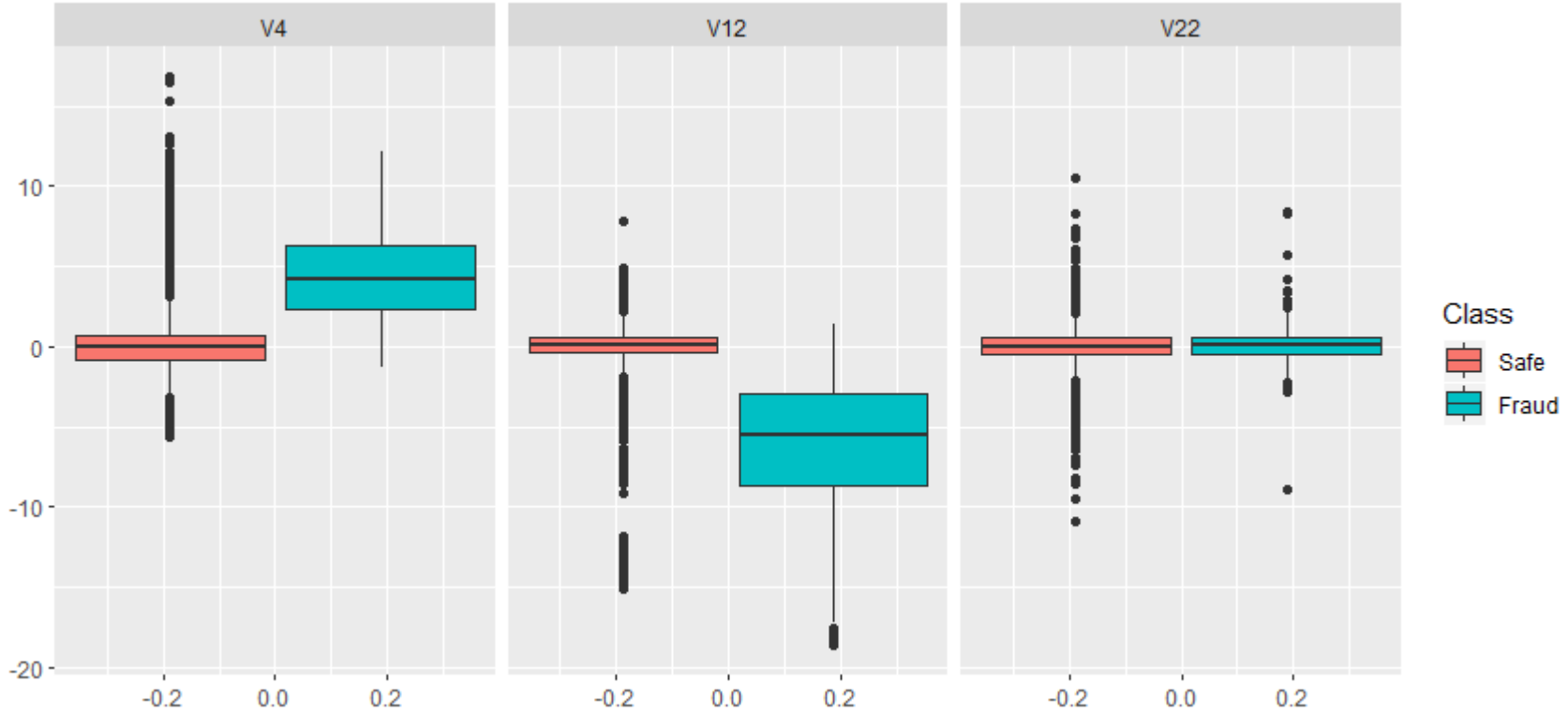
ML Methods for Credit Card Fraud Identification

Brandon Cook, Cason Wight, and Mitchell Wassom

Background

- Credit card fraud costs consumers an estimated $\$22$ billion per year.
- Due to the importance of this issue, credit card companies typically have teams of data scientists specifically tasked with identifying fraudulent transactions.
- We obtained data that contains information on about $300{,}000$ credit card transactions of which only 492 are known to be fraudulent (about 0.1%). The dataset also contains information on 29 different characteristics of the transaction.
- However, because such information is highly proprietary and confidential, most of the explanatory variables (labeled simply as V1-V28) are the principal component scores of the explanatory variables (borrowing 536 notation, these are the $\mathbf{Z}$s rather than the original $\mathbf{X}$s).
- Our task was to try to identify which transactions are fraudulent.

Data Visualization



Problems with Data

- Small proportion of transactions are fraudulent
  - Classification for unbalanced classes can be unreliable
- Fraud cases very similar to non-fraud cases
  - High variance
- Meaningful inference impossible due to principal-component variables

Goals

1. Apply machine learning (ML) methods to explore credit card fraud
    a. Classification Tree
    b. K-nearest neighbors (KNN)
    c. Neural Net
2. Assess prediction ability and “fit” of these three methods

Classification Trees

- Partition p-dimensional predictor space into distinct regions $R_1, \ldots, R_t$
- Predict class based on predictor region
  - Each region corresponds to one predicted class
  - Prediction is the most populous class in the region

$$\hat{y}|\mathbf{x}_0 = \sum_{t=1}^{T} \left(\underset{c}{\arg\max}\, \pi_c \right) \mathbb{I}(\mathbf{x}_0 \in R_t)$$

$$\pi_c = \text{Proportion of } \{y|\mathbf{x}_i : \mathbf{x} \in R_t\}$$

Classification Trees

- Split predictor space by predictor and cut point that leads to greatest reduction in Gini Index error
- Repeat for each of split “branches”

$$Error = \hat{\pi}_{0,t}\left(1 - \hat{\pi}_{0,t}\right) + \hat{\pi}_{1,t}\left(1 - \hat{\pi}_{1,t}\right)$$

$$\hat{\pi}_{k,t} = \text{Proportion of Obs in } R_t \text{ of class } k$$

Classification Trees

- Tree with lowest error will be overfit, unable to accurately predict new data
- Create pruned tree by reducing error with penalty
  - Penalty based on size of tree, T
  - Penalization parameter chosen via cross-validation

$$\text{Error} + \alpha T$$

$$\alpha = \text{Penalization parameter}$$

$$T = \text{Size of tree}$$

Classification Trees

Weaknesses:

- Homogeneity in regions lacks nuance
- Difficult to detect unbalanced classes
- No uncertainty for inference

Strengths:

- Outputs probabilities
- Decision tree is very interpretable

K-Nearest Neighbor

Model explanation

The general idea of K-Nearest Neighbor is to match each datapoint that we wish to predict up with their k "nearest neighbors," and look at what proportion of the k neighbors belong to each class. This is used as the predicted probability for the unknown datapoint.

The datapoint is usually then classified according to the highest predicted probability. (In this model, we set the cutoff probability value at $.22$ for classifying as a fraudulent transaction - increased correctly identified fraudulent transactions by 16, at the expense of 9 falsely identified non-fraudulent transactions).

How we find "nearest neighbors" is through Mahalanobis distance:

$$\text{Distance}_{M_{i,j}} = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)' \mathbf{S}^{-1} (\mathbf{x}_i - \mathbf{x}_j)}$$

K Nearest Neighbor Cont.

Greek letter definitions

K - number of nearest neighbors - found by cross-validation. (Lowest number of misclassifications in a testing set of 20% of the data.) In these data, 5 is the best value for K.

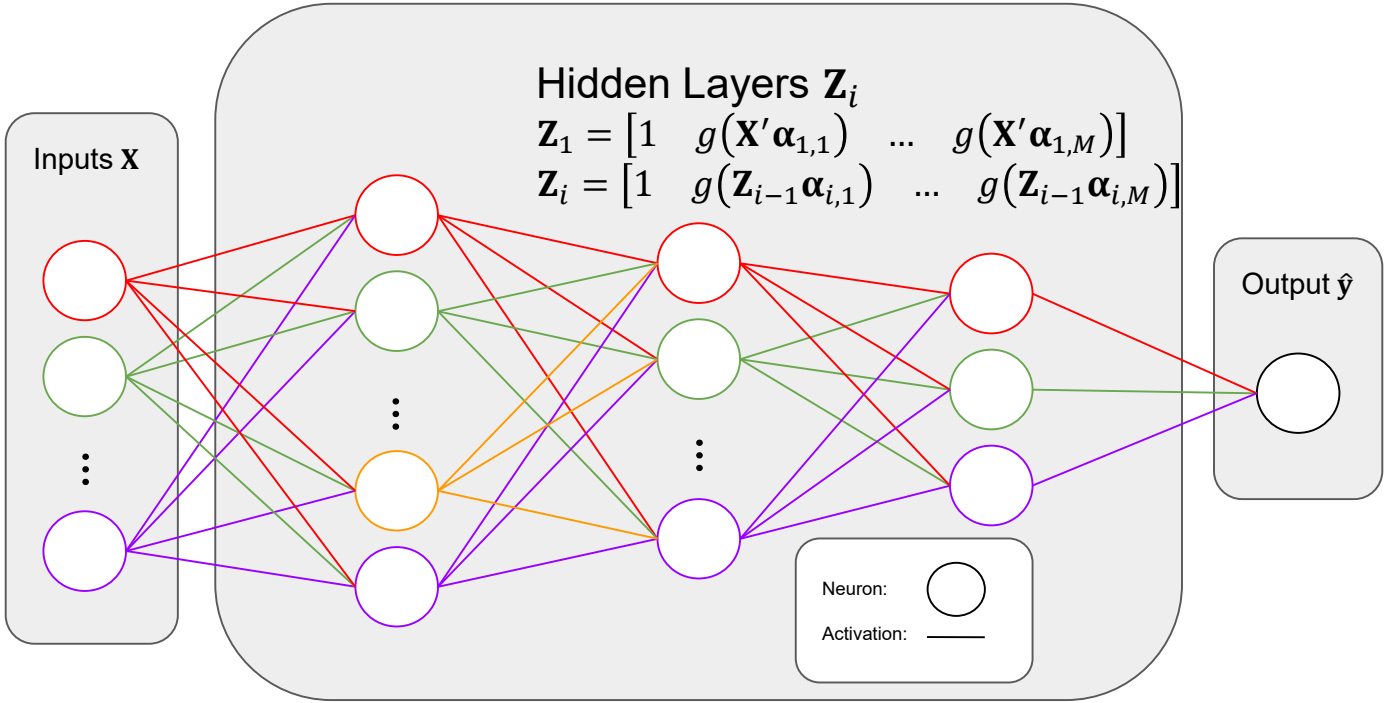
Assumptions

None

Strengths/weaknesses

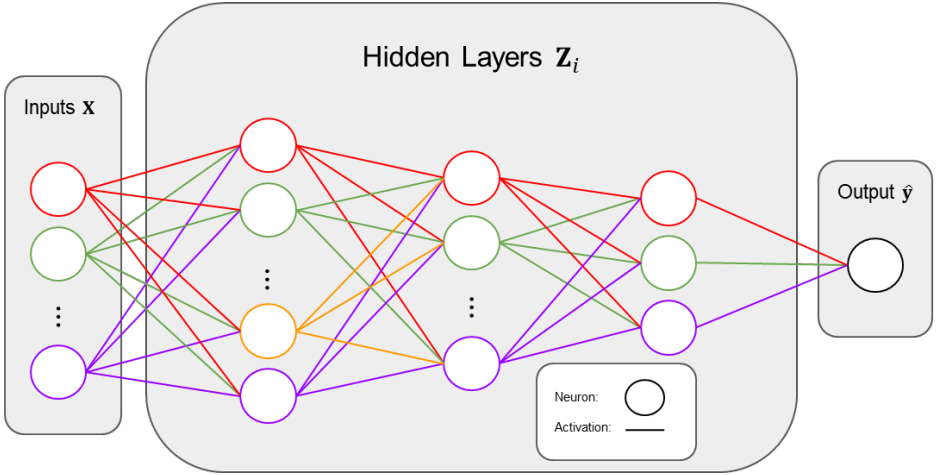
Strengths - Outputs estimated probabilities. Very easy concept to explain. Useful in difficult datasets where patterns are hard to explain. No assumptions.
Weaknesses - Not as good at classifying as, say, neural nets. Careful not to include useless variables.

Neural Nets



$g(\cdot)$ is the activation function (many available–such as sigmoidal, hyperbolic, etc.)
$\boldsymbol{\alpha}_{i,m}$ are the weights for each input given in the previous layer for the i^{th} layer on the m^{th} neuron of the level

Neural Nets



Steps for “fitting” a neural net

1. Center/scale the data

2. Construct the neural net framework

3. Update best parameters for the $\boldsymbol{\alpha}$s and $\mathbf{Z}$s via back-propagation by minimizing a loss function (regression/cross-entropy/etc.)
    a. For each step, calculate the $\mathbf{Z}$s
    b. Get derivatives of parameters with respect to the loss
    c. Use Newton-Raphson to minimize loss (gradient descent)

4. Tune the hyperparameters (number of layers/neurons/learning rate/etc.) via grid search and cross-validation

Neural Nets Cont.

Strengths:

- Highly flexible (linear, nonlinear, categorical, etc.)
- Can achieve excellent prediction
- No assumptions of any type needed (not a model)

Weaknesses:

- Inference is utterly impossible
- Tuning is a major challenge
- Computation time can be a burden

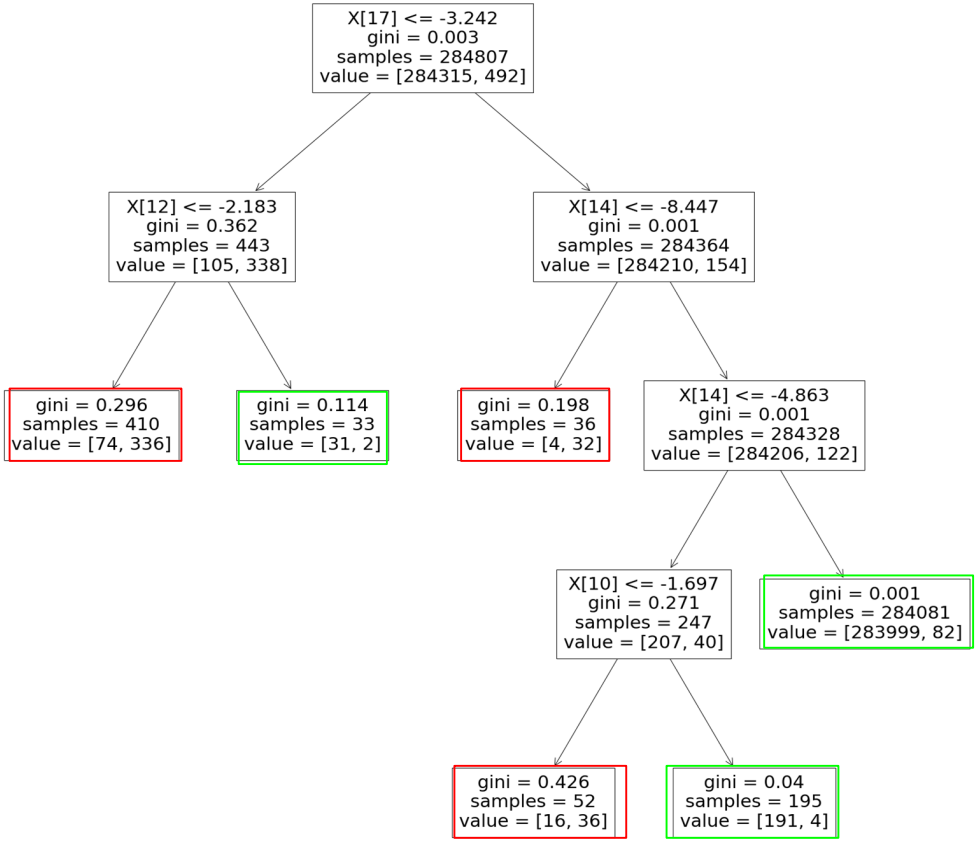
Classification Trees for Credit Card Fraud - Brandon

- All variables included; Pruning determines which variables are most important
- Pruning parameter chosen via cross-validation
  - Model with highest sensitivity chosen (thus AUC might be inferior)
  - Sensitivity prioritized because false negatives worse than false positives

	Sens	Spec	Prec	Neg Pred	F-score
Full Model	0.8211	0.9997	0.8112	0.9997	0.8162
CV	0.7760	0.9998	0.8584	0.9996	0.8151

Classification Tree

- Inputs 17, 12, 14, and 10 are the only variables that are in the final tree
- One large leaf with most of the non-fraud observations
- Five small leaves with varying fraud probabilities
- Cutoff: 6% chance of fraud



K-Nearest Neighbors for Credit Card Fraud - Mitch

Variables included/excluded - all included; Assumptions met - vacuously

Model fit (sensitivity, specificity, precision, negative predicted value, and F-score)

- Sensitivity = 410/492 = .8333
- Specificity = 284,283/284,315 = .9999
- Precision = 410/442 = .9276
- Negative Predictive Value = 284,283/284,365 = .9997

Self Fit	Predicted 0	Predicted 1	Sum
Actual 0	284,283	32	284,315
Actual 1	82	410	492
Sum	284,365	442	284,807

Cross validated values for prediction

- Sensitivity = 70/87 = .8046
- Specificity = 56,865/56,875 = .9998
- Precision = 70/80 = .8750
- Negative Predictive Value = 56,858/56,876 = .9997

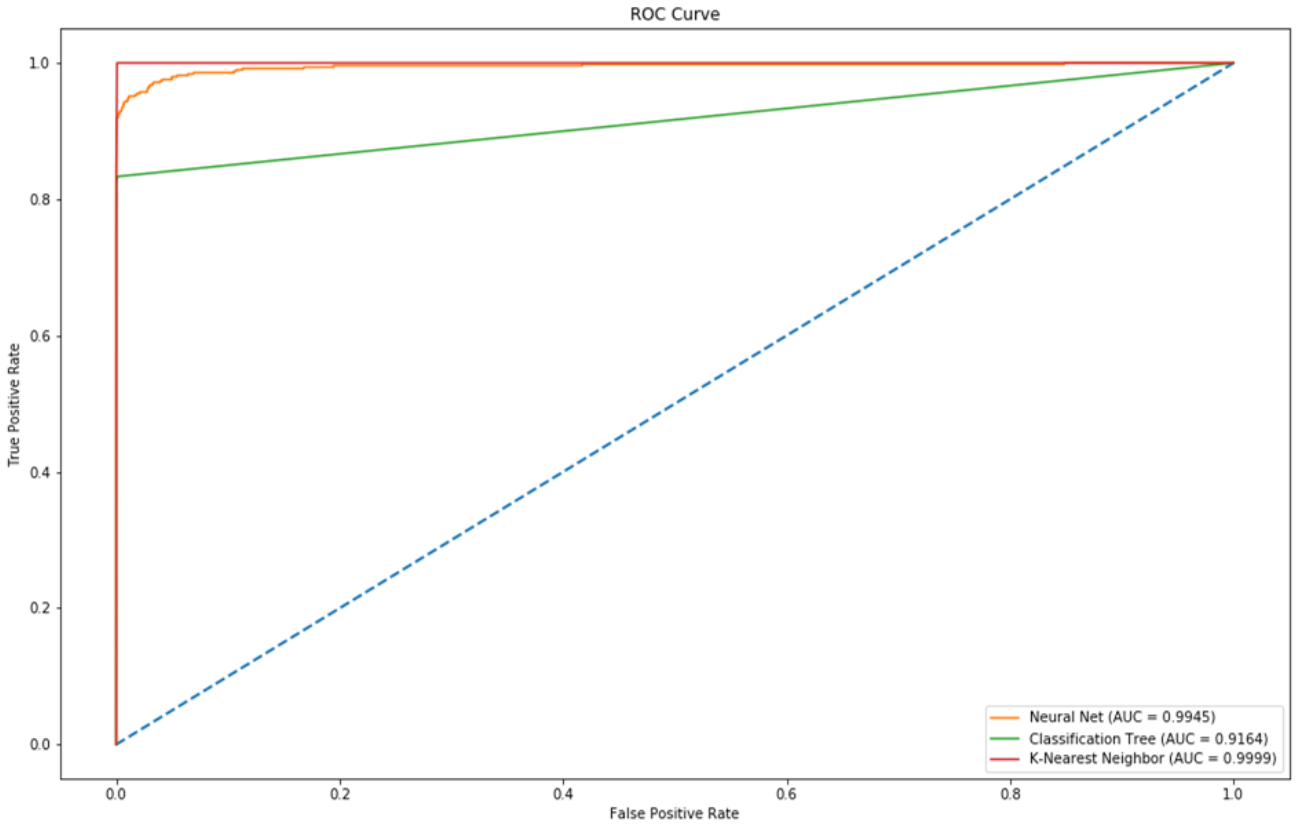
Training/Test	Predicted 0	Predicted 1	Sum
Actual 0	56,865	10	56,875
Actual 1	17	70	87
Sum	56,882	80	56,962

Neural Net for Credit Card Fraud

- Best neural net had 3 hidden layers, each with 30 neurons
- Best activation function was rectified linear unit ($ReLu$: $g(x) = \max(0, x)$)
- Best learning rate was 0.01
- Best cutoff probability was 36.98%
- All hyperparameters chosen via limited grid search (due to time constraints)

	Sens	Spec	Prec	Neg Pred	F-score
Full Model	0.8272	0.9999	0.9667	0.9997	0.8904
CV	0.8627	0.9997	0.8381	0.9998	0.8502

ROC curves and AUC



Summary of Results - Self Fit

Self - Fit	KNN	Trees	Neural Nets
Sensitivity	.8333	.8211	.8272
Specificity	.9999	.9997	.9999
Precision	.9276	.8112	.9667
NPV	.9997	.9997	.9997
F	.8779	.8162	.8904
AUC	.9999	.9164	.9945

Predictions - Testing / Training Set

80% Train/Test	KNN	Trees	Neural Nets
Sensitivity	.8046	.8529	.8627
Specificity	.9998	.9996	.9997
Precision	.8750	.8131	.8381
NPV	.9997	.9997	.9998
F	.8383	.8325	.8502

Conclusion

- About 80% of CC fraud can be detected
  - Add other variables to capture more variation
  - More cases of fraud could increase sensitivity
- All methods have difficulty identifying fraud (relatively low sensitivity)
- KNN fits the data best
  - Sensitivity of $.8333$
  - AUC of $.9999$, when constructed with full data
- Neural nets had the best sensitivity for prediction ($.8627$)
- Meanings of variables must be known to take action
- With financial costs of false positives/negatives known, methods could be optimized to minimize costs (better method for selecting cutoff probabilities)

Teamwork

Introduction, KNN, Summary of Results - Mitchell

Data visualization/issues, CT, Conclusion - Brandon

Model fit and prediction ability, NN, Goals/Outline - Cason