These slides: https://github.com/WinVector/Examples/blob/main/BarugROCday/ROCUtility.pdf
Rehearsal recording: https://youtu.be/US9EW7OB870

How to Pick an Optimal Utility Threshold Using the ROC Plot

John Mount
Win-Vector LLC
BARUG ROC Day
November 10, 2020

These slides: https://github.com/WinVector/Examples/blob/main/BarugROCday/ROCUtility.pdf
Rehearsal recording: https://youtu.be/US9EW7OB870

Outline

- In this talk I would like to show you the classic methods for picking the point on the ROC plot that maximizes utility with respect to some stated business goals.
- We are going to show code-snippets illustrating how to calculate using ROC concepts.
- Our example problem will be a probability model for a classification problem.
  - Why one should use probability models for classification problems instead of using "classifiers", "hard classifiers", or "classification rules" is discussed in "Don't Use Classification Rules for Classification Problems" https://win-vector.com/2020/08/07/dont-use-classification-rules-for-classification-problems/ .
  - Professor Norm Matloff also has also shared important criticisms of related mal-patterns in classification frameworks.
- We will assume we can specify some utilities to give us an objective to solve for.
- I'll include some new results and an open research direction in this talk.

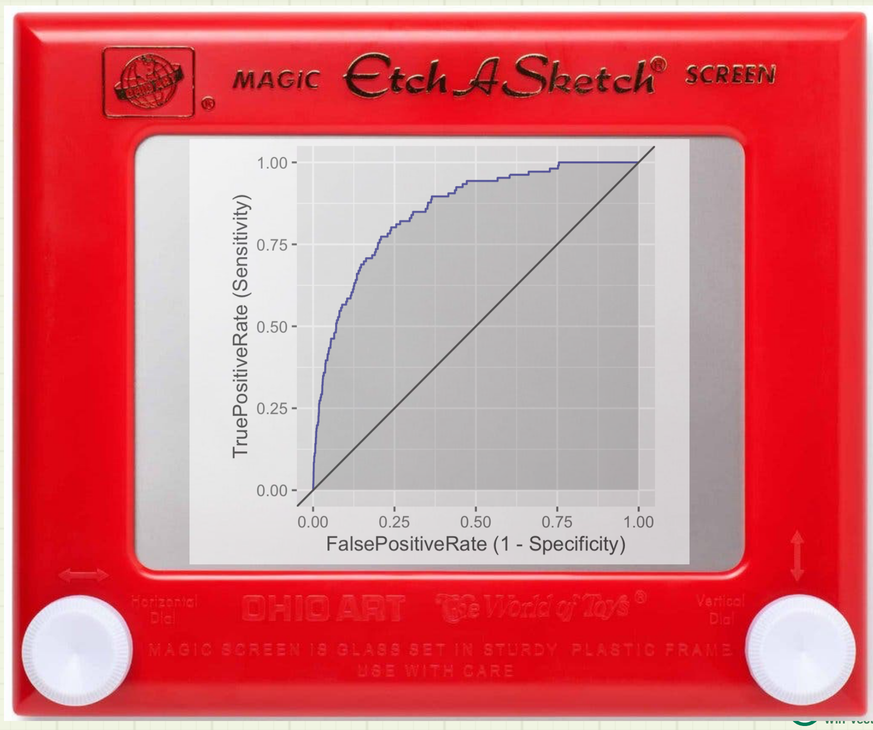
Win-Vector LLC

Let’s define the ROC plot yet again

Win-Vector LLC

Let’s define the ROC plot yet again

- It is exactly the set of shapes we can draw on an EtchASketch from the bottom left to top right while turning each dial only clockwise.



Our Example Data Source

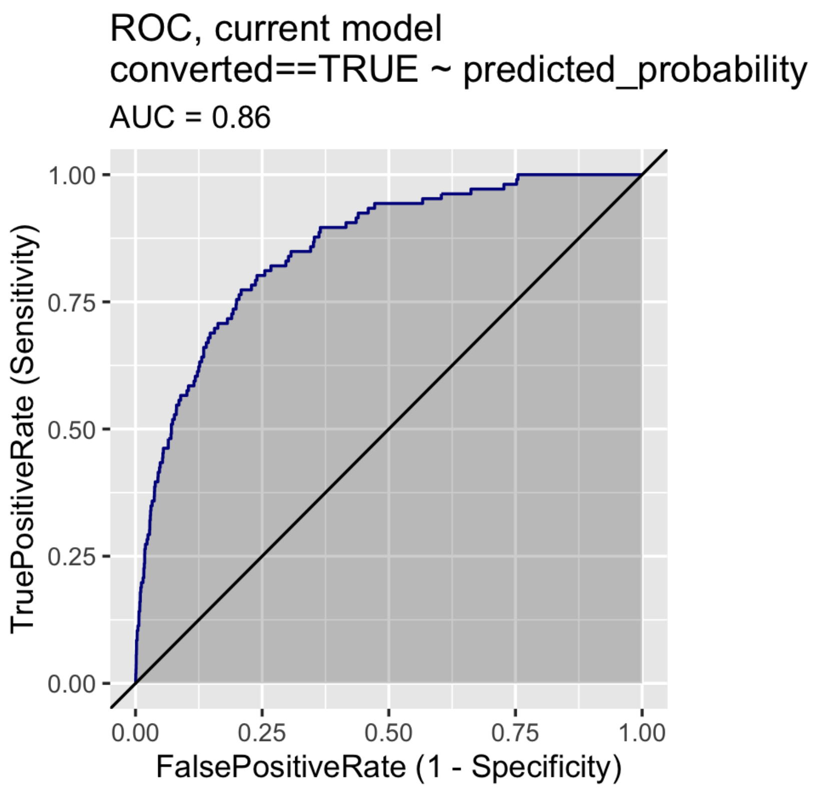
- Our example data is the synthetic example taken from https://github.com/WinVector/sigr/blob/main/extras/utility_modeling/ROC_optimization.Rmd , which is discussed here: https://win-vector.com/2020/10/10/how-to-pick-an-optimal-utility-threshold-using-the-roc-plot/ .

Looking At The Example Data

```
knitr::kable(head(d))

|converted | predicted_probability|
|:---------|---------------------:|
|FALSE     |             0.0040164|
|FALSE     |             0.0199652|
|FALSE     |             0.0132867|
|FALSE     |             0.0051605|
|FALSE     |             0.0038753|
|FALSE     |             0.0057591|
```

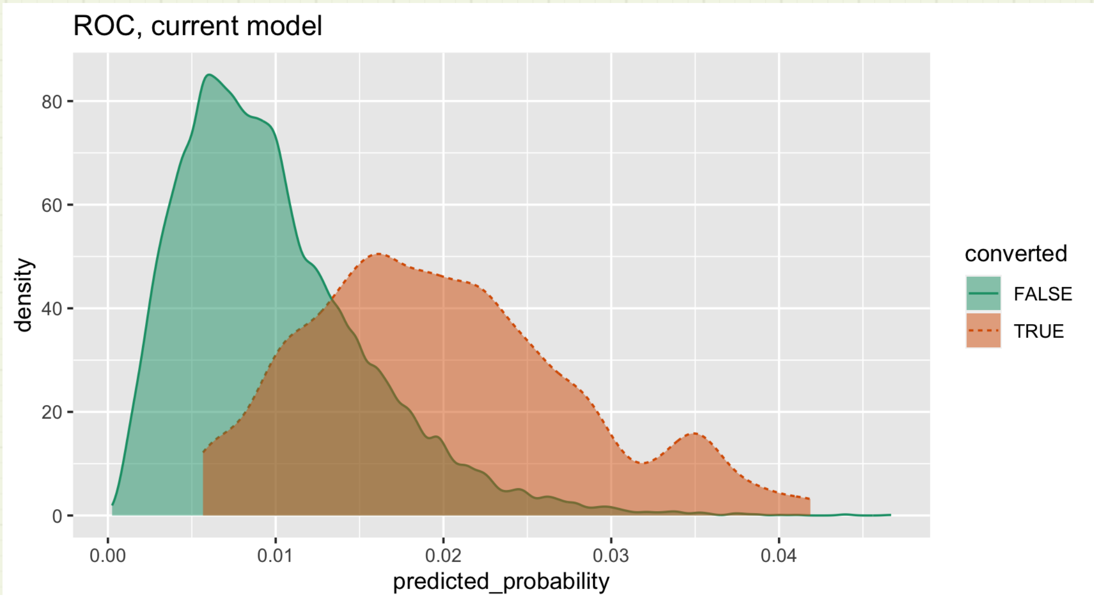
Graphing the Example Data

```
library(WVPlots)

ROCPlot(
  d,
  xvar = "predicted_probability",
  truthVar = "converted",
  truthTarget = TRUE,
  title = "ROC, current model")
```



Graphing the Example Data

```
DoubleDensityPlot(
  d,
  xvar = "predicted_probability",
  truthVar = "converted",
  title = "ROC, current model")
```



Some Observations

- For a calibrated probability model the double density plot encodes the outcome prevalence (please see https://win-vector.com/2020/10/27/the-double-density-plot-contains-a-lot-of-useful-information/).
- The position of the slope-1-point of the ROC plot *may* also contain such information, but only under distribution assumptions that seem not to be always met (some related notes here: https://win-vector.com/2020/10/29/a-single-parameter-family-characterizing-probability-model-performance/).

Stating our Utilities

- To use the ROC plot to optimize utility we must first state our value for each of:
    - True Positives (instances we thought would convert and did)
    - True Negatives (instances we thought would not convert and did not)
    - False Positives (instances we thought would convert and did not)
    - False Negatives (instances we thought would convert and did)
- And we must have the observed prevalence (rate of positive occurrences).

Win-Vector LLC

The Classic Solution

- Equation 1.18 of Section 1.4.1 "Decision Goal: Maximum Expected Value" of James P. Egan, *Signal Detection Theory and ROC Analysis*, Academic Press, 1975 state the optimum utility is found where:

$$\frac{\partial Sensitivity}{\partial(1 - Specificity)} = \frac{1 - Prevalence}{Prevalence} \frac{TNV - FPV}{TPV - FNV}$$

Win-Vector LLC

Let’s Derive That

- We re-derived that using **`sympy`** here: https://github.com/WinVector/sigr/blob/main/extras/utility_modeling/ROC_utility.ipynb .
- Solution outline:

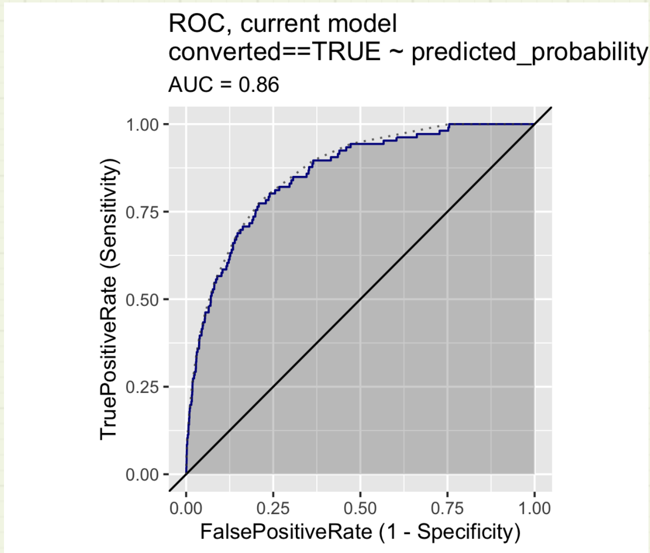
```
# enter relations
relations = [
    Population - (TP + FN + TN + FP),
    Prevalence - (TP + FN) / (TP + FN + TN + FP),
    sensitivity - (TP / (TP + FN)),
    specificity - (TN / (TN + FP)),
    Utility - (TP * TPV + FP * FPV + TN * TNV + FN * FNV)
]

# solve for utility
util = solve(relations, [TP, FP, TN, FN, Utility])[Utility]

# take the derivative
dutil_dspec = diff(util.subs(sensitivity, sens), specificity)
relation = dutil_dspec.subs(dSens_dSpec, - ROC_slope)

# solve for derivative equaling zero
soln = solve(relation, ROC_slope)
```

Win-Vector LLC

Using The Solution

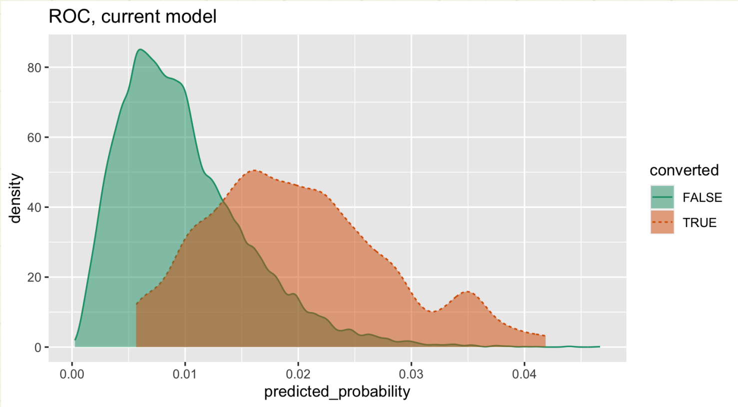
- We need to work on a *proper* ROC plot.
  - This is defined as an ROC plot that is convex
  - In this plot slope is monotone decreasing in **`1 - Specificity`**
  - This makes the slopes determine **`(1 - Specificity, Sensitivity)`** pairs in a well defined fashion.

```
ROCPlot(
  d,
  xvar = "predicted_probability",
  truthVar = "converted",
  truthTarget = TRUE,
  title = "ROC, current model",
  add_convex_hull = TRUE)
```

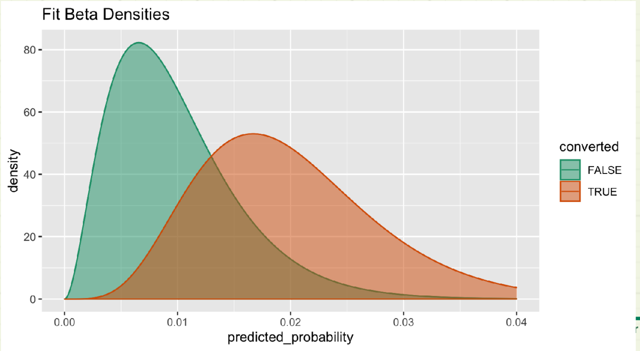


Step 1: Fit Conditional Beta Distributions

```
DoubleDensityPlot(
  d,
  xvar = "predicted_probability",
  truthVar = "converted",
  title = "ROC, current model")
```




```
library(wrapr)

unpack[
  shape1_pos,
  shape2_pos,
  shape1_neg,
  shape2_neg] <-
      sigr::find_ROC_matching_ab(
        modelPredictions =
           d$predicted_probability,
        yValues = d$converted)
```



This is not a Disadvantage

- A low complexity parametric fit may reduce variance (at a possible cost of some bias if we don't have the right parametric model family).
- Classically we wouldn't have the ability to evaluate the ROC plot at many points.
    - The ROC plot was not historically the performance of a single continuous model score evaluated at different thresholds.
    - It was instead point measured by experiment. Each point on the ROC curve might require executing a whole new empirical experiment. We might have as few as 2 to 5 evaluations available!

Some Comments on Working Parametrically

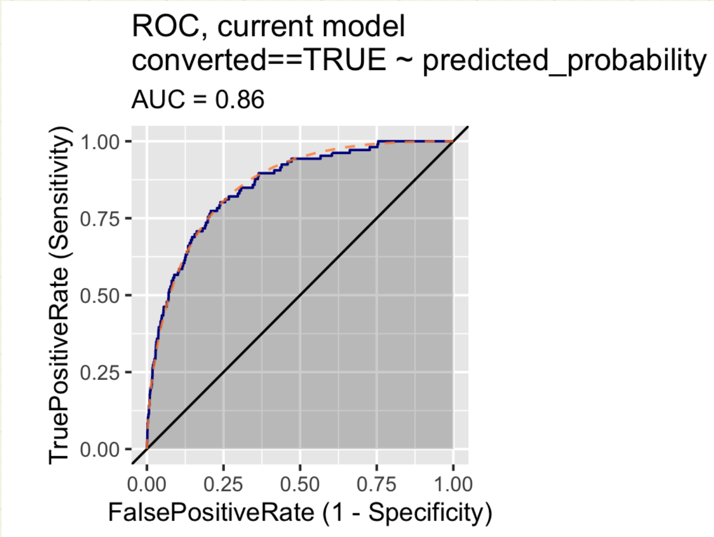
- The most popular parametric model was the logit-normal with matching variances.
  - Why the variances must match is discussed in “Your Lopsided Model is Out to Get You” https://win-vector.com/2020/10/26/your-lopsided-model-is-out-to-get-you/ .
  - For beta distributions the natural conditions are a bit different than matching variances. Some notes on this can be found in “A Single Parameter Family Characterizing Probability Model Performance” https://win-vector.com/2020/10/29/a-single-parameter-family-characterizing-probability-model-performance/ .

Step 2: Define Our Utility, or Determine Our Slope

```
true_positive_value <- 100 - 5   # net revenue - cost
false_positive_value <- -5       # the cost of a call
true_negative_value <-  0
false_negative_value <- -0.01    # a small penalty for having missed them
prevalence <- mean(d$converted)

target_slope <- ((1 - prevalence) / (prevalence)) *
  ((true_negative_value - false_positive_value)/(true_positive_value - false_negative_value))
print(target_slope)
# [1] 4.912095
```

Step 3: Plot the Idealized ROC plot

```
ROCPlot(
  d,
  xvar = "predicted_probability",
  truthVar = "converted",
  truthTarget = TRUE,
  title = "ROC, current model",
  add_beta_ideal_curve = TRUE)

ideal_roc <- sigr::sensitivity_and_specificity_s12p12n(
      seq(0, 1, by = 0.000001),
      shape1_pos = shape1_pos,
      shape2_pos = shape2_pos,
      shape1_neg = shape1_neg,
      shape2_neg = shape2_neg)

n <- nrow(ideal_roc)
ideal_roc$slope <-
  c(NA, ideal_roc$Sensitivity[-1] - ideal_roc$Sensitivity[-n]) /
  c(NA, (1 - ideal_roc$Specificity[-1]) - (1 - ideal_roc$Specificity[-n]))
```

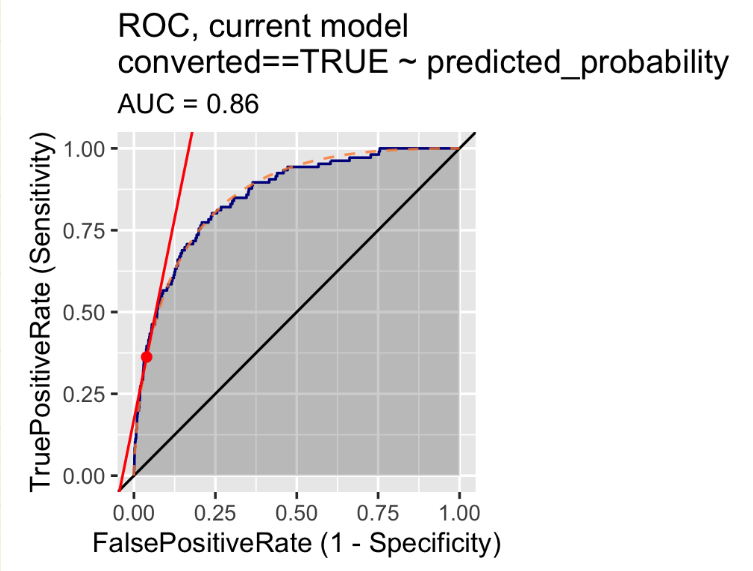


Win-Vector LLC

Step 4: Find the Matching Slope

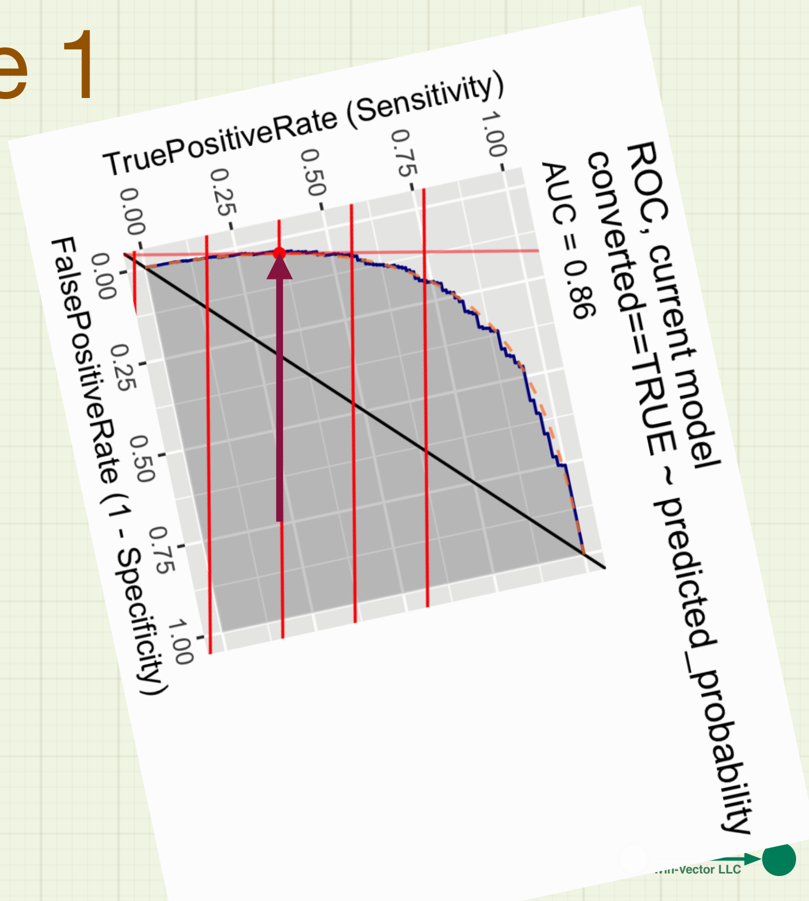
```
idx <- which.min(abs(ideal_roc$slope - target_slope))
ideal_roc[idx, ]
##          Score Specificity Sensitivity    slope
## 21740 0.021739   0.9610274   0.3627042 4.911909
```

Please remember this number



Alternative 1

- Could find same point by optimizing in a direction orthogonal to desired slope line.
- That is just a line with slope **`-1/s`**, where **`s`** was our original slope target. Optimize in this direction.
- This could in principle be done by laying a ruler at slope **`s`** against the boundary of the ROC plot.
    - The same plot could be re-used for different business trade-offs.



Win-Vector LLC

Alternative 2

- The slope is the derivative of **`Sensitivity`** with respect to **`1 - Specificity`**
- We know the derivatives of **`Sensitivity`** and **`1 - Specificity`**
- So we can numerically solve the following equation for **`x`** in the range **`(0, 1)`**.

$$x^{a_{positive} - a_{negative}} (1-x)^{b_{positive} - b_{negative}} = \frac{\beta(a_{negative}, b_{negative})}{\beta(a_{positive}, b_{positive})} \frac{1 - Prevalence}{Prevalence} \frac{TNV - FPV}{TPV - FNV}$$

Win-Vector LLC

The Fully Calibrated Case

- *If* the model score is fully calibrated (**`E[y|pred] = pred`** for all observed values of **`pred`**), and the outcome conditioned distributions are beta densities, then this simplifies to the following

$$\frac{x}{1-x} = \frac{\beta(a_{negative}, b_{negative})}{\beta(a_{positive}, b_{positive})} \frac{1 - Prevalence}{Prevalence} \frac{TNV - FPV}{TPV - FNV}$$

- These fully calibrated condition is *not* met for this example data, and *not* usually met for common classifiers. So a full calibration polishing step is perhaps a possible avenue for probability model improvement.

- Some notes on these ideas can be found here: https://win-vector.com/2020/10/29/a-single-parameter-family-characterizing-probability-model-performance/

Win-Vector LLC

Full Calibration

- Example from **`xgboost`** applied to the **`KDD2009`** data set. Notice the model is not fully calibrated on training data, but nearly fully calibrated on held-out test data!

(source https://github.com/WinVector/Examples/blob/main/density_shapes/PredPlot.md)

Win-Vector LLC

Conclusions / Take-Aways

- The ROC plot is a classic tool from signal detection theory used to characterize and optimize decision thresholds.

- In modern machine learning contexts the ROC is largely used as a step to the AUC (area under the curve) as a generic “goodness score” for a numeric model’s performance on a classification problem.

- To work on the ROC curve we have to idealize it (take the convex hull, or even perform a parametric fit).

- Maximizing utility remains an import point. However, if it is so important it may make more sense to solve for utility in a natural utility space. This leads us to our next speaker.

Win-Vector LLC

Thank You

These slides: https://github.com/WinVector/Examples/blob/main/BarugROCday/ROCUtility.pdf
Rehearsal recording: https://youtu.be/US9EW7OB870

Win-Vector LLC