UCLA Anderson
SCHOOL of MANAGEMENT

MGMTMFE 431:

Data Analytics and Machine Learning

Topic 8: Support Vector Machines

Spring 2025

Professor Lars A. Lochstoer

a. Support Vector Machines

Supervised learning technique

- Nonlinear classification
- We will cover two classes (e.g., default vs no default)
- Extension to multiple classes available

Developed in the Computer Science community

- Proved successful
- Applications spread to other fields
- Standard tool in the Machine Learning toolkit

a. Hyperplane

Consider P features X_1, …, X_P.

- A ***hyperplane*** is the *P-1* dimensional subspace:

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_P X_P = 0$$

If two features (two dimensions, the hyperplane is a one-dimensional line)

Use as classifier:

- If $\beta_0 + \beta_1 X_1 + \cdots + \beta_P X_P > 0$ classify as (say) 1
- If $\beta_0 + \beta_1 X_1 + \cdots + \beta_P X_P \leq 0$ classify as (say) -1

UCLA Anderson SCHOOL of MANAGEMENT

a. Separating hyperplanes

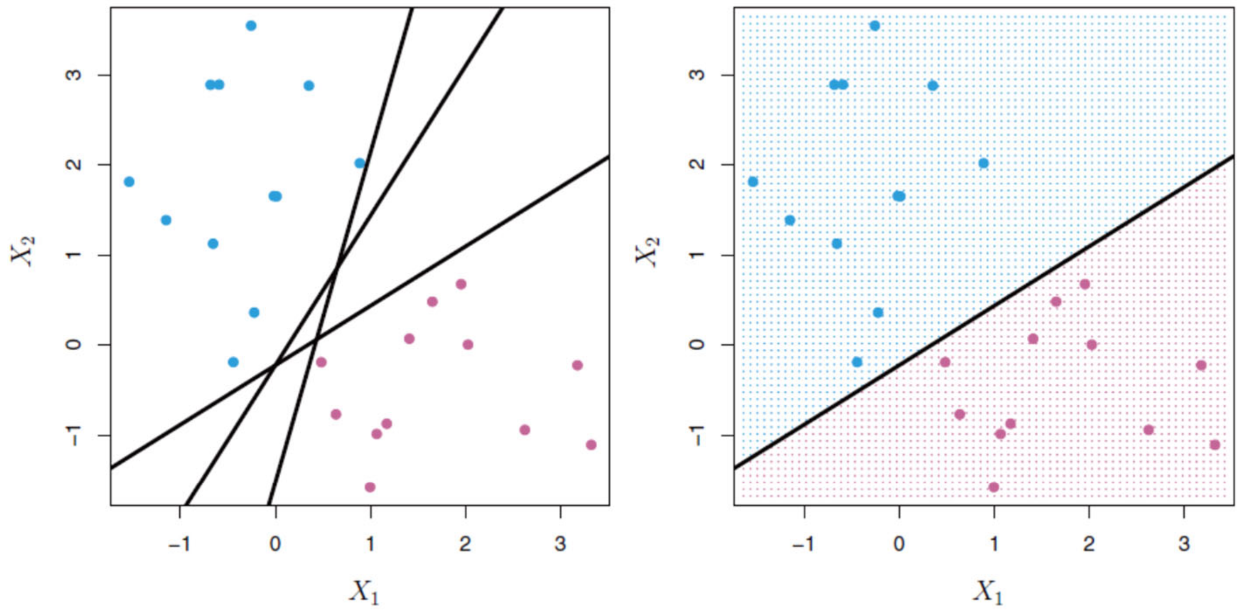


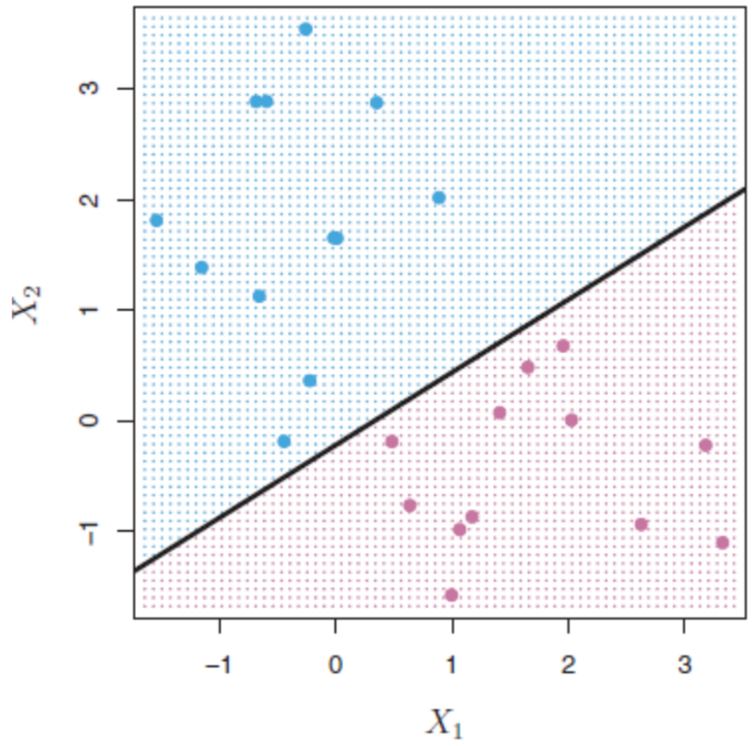
FIGURE 9.2. Left: *There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black.* Right: *A separating hyperplane is shown in black. The blue and purple grid indicates the decision rule made by a classifier based on this separating hyperplane: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.*

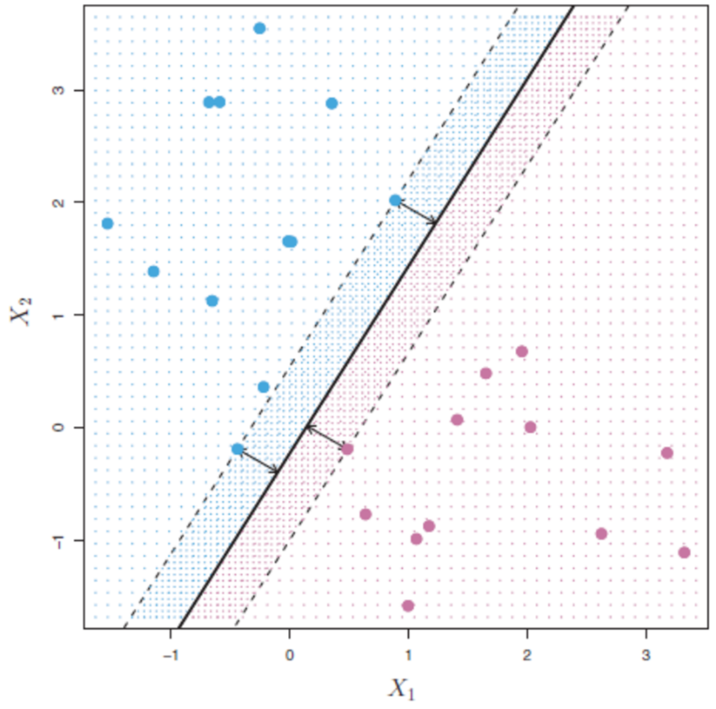
UCLA Anderson
SCHOOL of MANAGEMENT

a. Identification by Maximal Margin

In previous plot, infinite number of solutions

- Idea: Define classifier as the one with the maximal margin possible
- Max(min(distance to margin across points))
- ***Depends only on the few points at the margin***. These data points are call ***Support Vectors***





UCLA Anderson SCHOOL of MANAGEMENT

a. The Maximal Margin Problem

Let y_i = 1 or -1, x_{ij} are the features

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M \tag{9.9}$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1, \tag{9.10}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \ \ \forall\, i = 1, \ldots, n. \tag{9.11}$$

Note that (9.10) is not a constraint, but a normalization

(9.11) guarantees all observations will be on right side of hyperplane

M is the distance in the maximal margin

UCLAAnderson
SCHOOL of MANAGEMENT

a. Perfect linear separation may not exist

But, a separating hyperplane may not exist!

- Need a "***soft margin***" (allow some observations to be off)

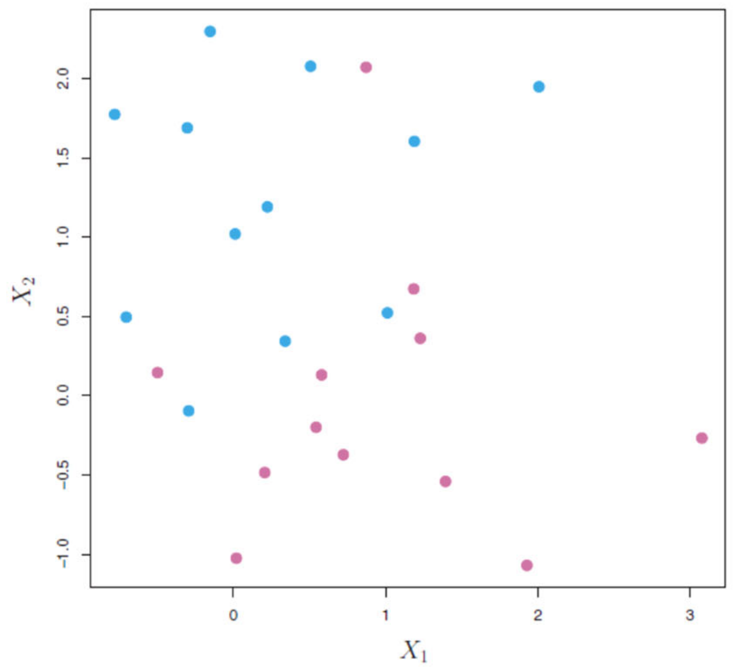


FIGURE 9.4. *There are two classes of observations, shown in blue and in purple. In this case, the two classes are not separable by a hyperplane, and so the maximal margin classifier cannot be used.*

a. Support Vector Classifier

Allow for ***soft margins*** where errors are tolerated

- The width of these margins is a tuning parameter found through cross-validation
- Draw the optimal hyperplane where we allow for misclassification within the margins.

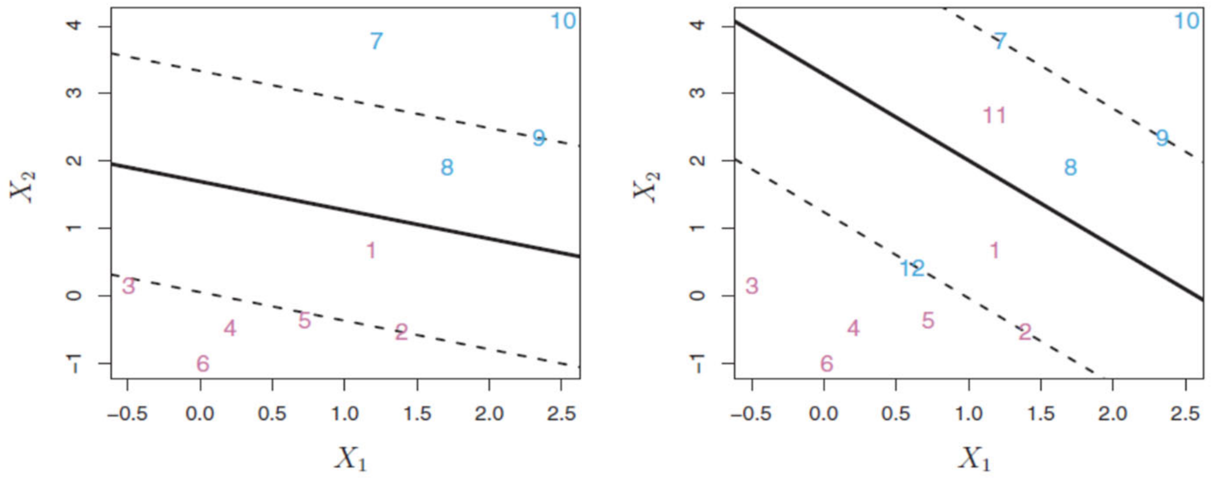


FIGURE 9.6. Left: *A support vector classifier was fit to a small data set. The hyperplane is shown as a solid line and the margins are shown as dashed lines.* Purple observations: *Observations* 3, 4, 5, *and* 6 *are on the correct side of the margin, observation* 2 *is on the margin, and observation 1 is on the wrong side of the margin.* Blue observations: *Observations* 7 *and* 10 *are on the correct side of the margin, observation* 9 *is on the margin, and observation* 8 *is on the wrong side of the margin. No observations are on the wrong side of the hyperplane.* Right: *Same as left panel with two additional points,* 11 *and* 12. *These two observations are on the wrong side of the hyperplane and the wrong side of the margin.*

UCLA Anderson SCHOOL of MANAGEMENT

a. Objective Function (linear classifier)

The objective function of a linear support vector classifier is:

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

What's new?

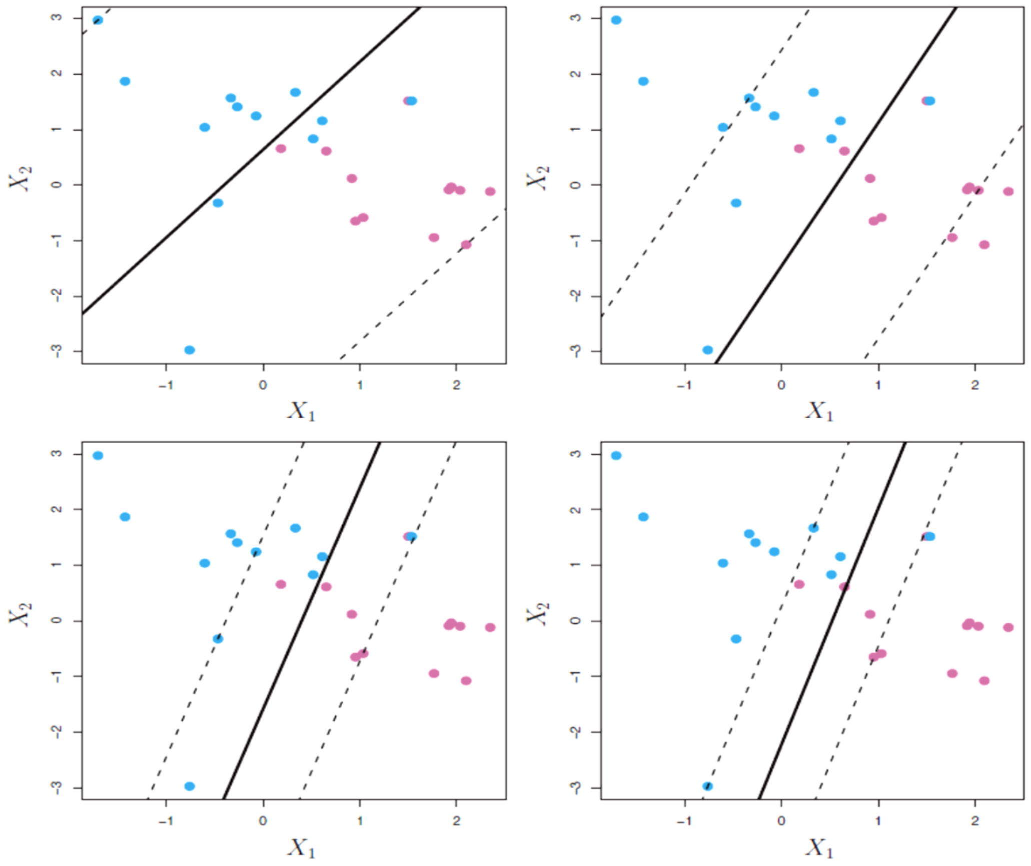
- The ***regularization*** implicit in the fuzzy boundary (C > 0)
- This is the main tuning parameter, C = 0 no errors allowed
- Note we are allowing for misclassification, just "*not too much*"

UCLAAnderson
SCHOOL of MANAGEMENT

a. *C* and the Bias-Variance Trade-Off

C large: high bias (errors in classification), but low variance (adding a point or two won’t change margin hardly at all

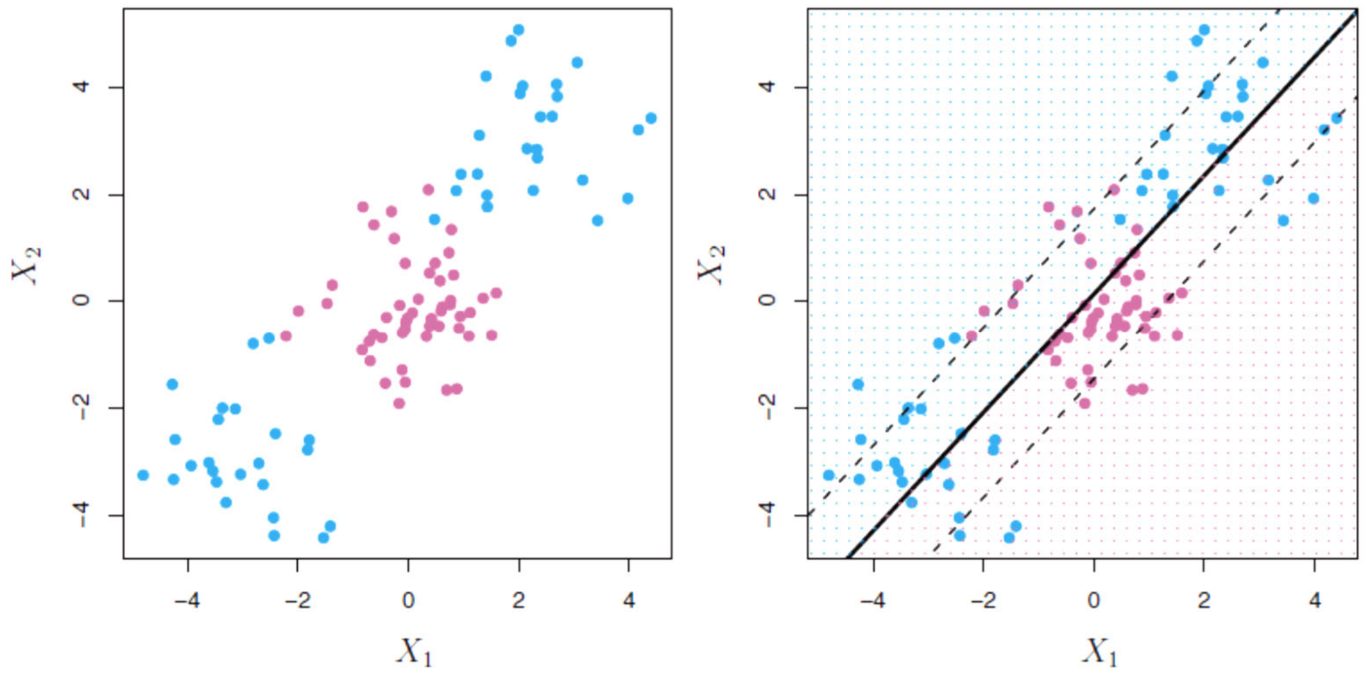
C small: low bias (mostly correct classification), but high variance (adding a data point could greatly affect the classifier, ie the lines would change a lot potentially)



UCLAAnderson
SCHOOL *of* MANAGEMENT

a. What about nonlinearity?

Linear classifiers will not always perform well



a. Support Vector Machine

Sometimes linear classification is not appropriate

- Support Vector Machines (SVMs) allow for ***nonlinear boundaries***
- Popular functional forms are ***polynomial*** and ***radial***
- These ***kernels*** require another tuning parameter (polynomial order of polynomial kernel or sensitivity in radial kernel)
- I will show graphically the outcomes of these two kernels and then present the math and an example

a. 2nd order polynomial classifier

We could consider classifier function as 2nd-order polynomial:

$$\underset{\beta_0,\beta_{11},\beta_{12}\ldots,\beta_{p1},\beta_{p2},\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to } y_i\left(\beta_0+\sum_{j=1}^{p}\beta_{j1}x_{ij}+\sum_{j=1}^{p}\beta_{j2}x_{ij}^2\right)\geq M(1-\epsilon_i),$$

$$\sum_{i=1}^{n}\epsilon_i\leq C,\quad \epsilon_i\geq 0,\quad \sum_{j=1}^{p}\sum_{k=1}^{2}\beta_{jk}^2=1.$$

where *M* is the width of the margin (the dashed line's distance from solid line)

- Quickly, however, the number of parameters would get high as we go to higher-order polynomials if we stick to this way of enlarging feature set

UCLA Anderson
SCHOOL of MANAGEMENT

a. Support Vector Machine

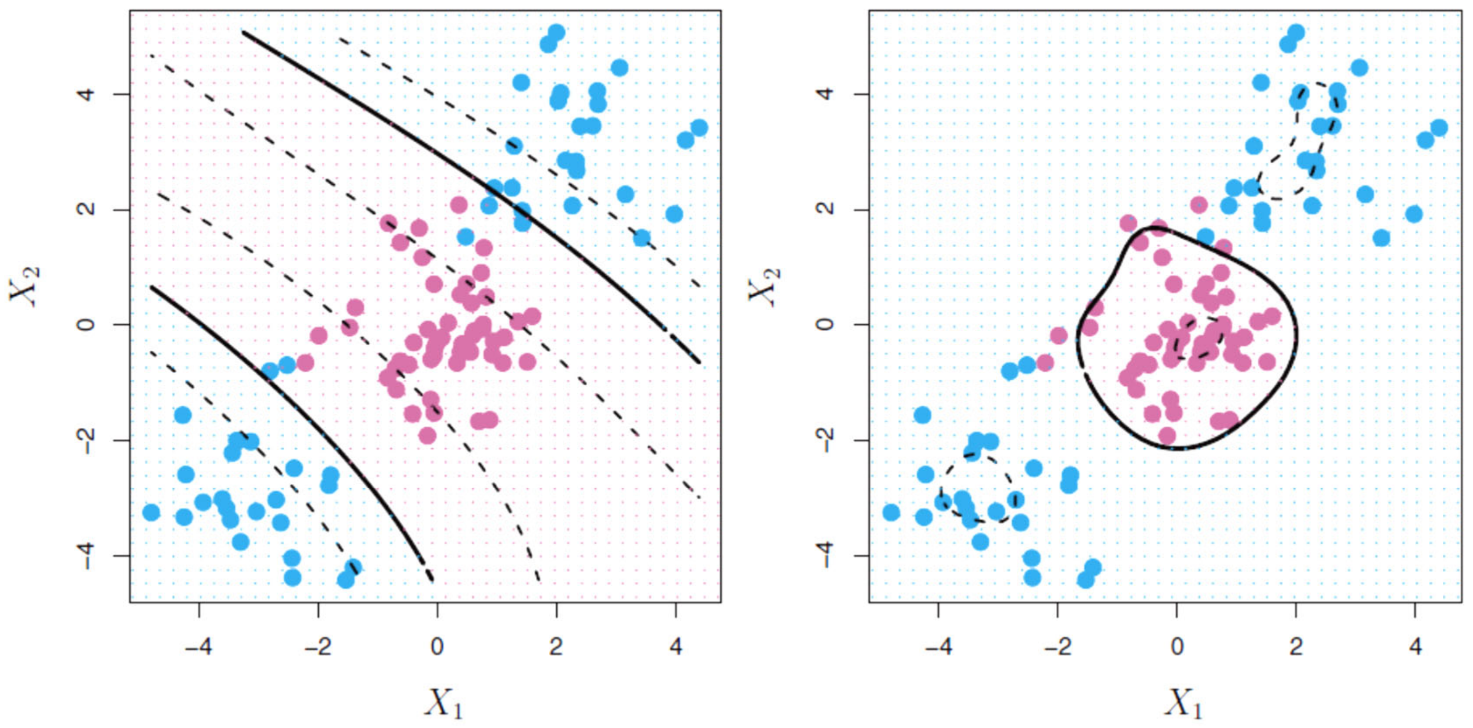


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

UCLA Anderson SCHOOL of MANAGEMENT

a. SVM Kernels

The ***nonlinear boundaries*** of SVM are formalized in "***Kernels***"

- Enlarging the feature space in a particular way that's more efficient than just adding polynomial terms in terms of number of parameters
- It turns out the solution to the linear classifier of the problem on Slide 9 is a function of the inner product between all observations (think *correlations*; here, x_i is vector of features for observation i, and *x* is the vector of features to be classified). The linear support vector classifier can therefore be represented as:

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$

- Here, the brackets denote inner product (the *p* features in observation *x* is multiplied by the p features in x_i)

UCLA Anderson
SCHOOL *of* MANAGEMENT

a. SVM Kernels (see Ch 9.3 in ISL book)

The SVM Kernel is a generalization of the inner product

- For instance, a *d*th -degree ***polynomial Kernel*** is:

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

- A ***radial Kernel*** is

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2).$$

- And the classifier function is then:

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$$

a. SVM Kernels (see Ch 9.3 in ISL book)

From last slide, classifier function is:

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$$

- Note that the sum is over *i* in ***S***.
- S is the set of observations that are within the margins. These are the only observations that matter for the objective function
- Thus, in practice there are < *n* parameters in the optimization
- Luckily, all of this is already coded up for us! We will continue to use the Python package sklearn in our worked example

a. Credit card data set

Use text data set: default_of_credit_card_clients.tsv

- Have billing amount and payment amount for 6 bills

```
   ID  LIMIT_BAL  SEX  ...  PAY_AMT5  PAY_AMT6  default payment next month
0   1      20000    2  ...         0         0                           1
1   2     120000    2  ...         0      2000                           1
2   3      90000    2  ...      1000      5000                           0
3   4      50000    2  ...      1069      1000                           0
4   5      50000    1  ...       689       679                           0
```

UCLAAnderson
SCHOOL of MANAGEMENT

a. Pre-process the data

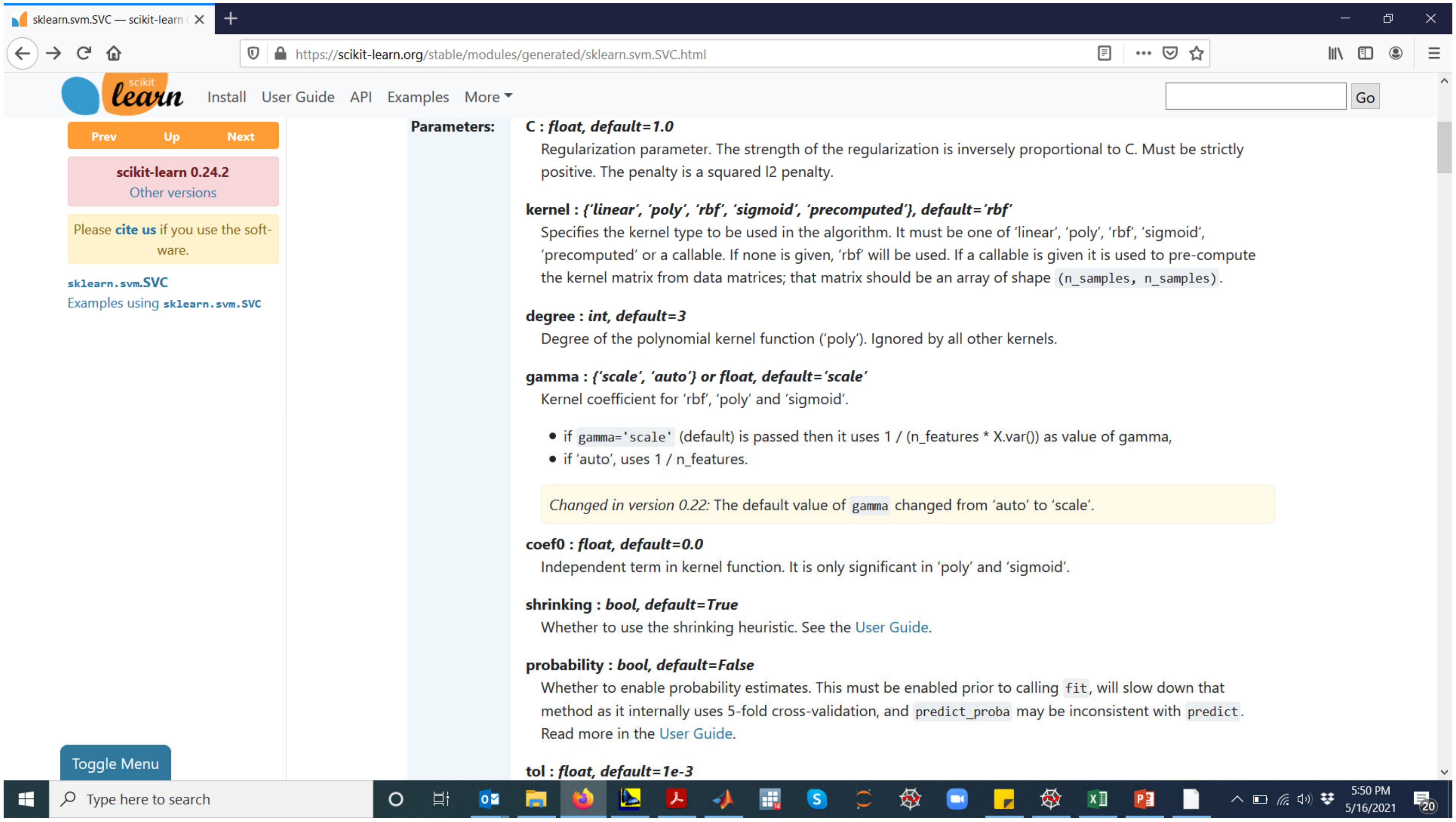
We make categorical variables into dummies, and scale all the features before sending to the Support Vector Machine classifier

```
# Split data into dependent and independent variables
X = df_downsample.drop('DEFAULT', axis=1).copy()
y = df_downsample['DEFAULT'].copy()

# One-hot encoding (i.e., creating dummy variables for each case)
# we only do so for categorical variables
X_encoded = pd.get_dummies(X, columns=['SEX', 'EDUCATION', 'MARRIAGE',
'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6'])

# Split data randomly into train/test
# since proportion not specified, defaults to 25% in test sample
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y,
random_state=42)

# scale X variables to mean zero and unit variance
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

UCLAAnderson
SCHOOL of MANAGEMENT

a. SVC function

The SVC function is the support vector machine classifier function in scikit-learn (sklearn)


sklearn.svm.SVC — scikit-learn
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
Install
User Guide
API
Examples
More
Go
Prev
Up
Next
scikit-learn 0.24.2
Other versions
Please cite us if you use the software.
sklearn.svm.SVC
Examples using sklearn.svm.SVC
Toggle Menu
Type here to search
5:50 PM
5/16/2021

Parameters:

C : *float, default=1.0*
Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

kernel : *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default='rbf'*
Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

degree : *int, default=3*
Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

gamma : *{'scale', 'auto'} or float, default='scale'*
Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features.

Changed in version 0.22: The default value of `gamma` changed from 'auto' to 'scale'.

coef0 : *float, default=0.0*
Independent term in kernel function. It is only significant in 'poly' and 'sigmoid'.

shrinking : *bool, default=True*
Whether to use the shrinking heuristic. See the User Guide.

probability : *bool, default=False*
Whether to enable probability estimates. This must be enabled prior to calling `fit`, will slow down that method as it internally uses 5-fold cross-validation, and `predict_proba` may be inconsistent with `predict`. Read more in the User Guide.

tol : *float, default=1e-3*

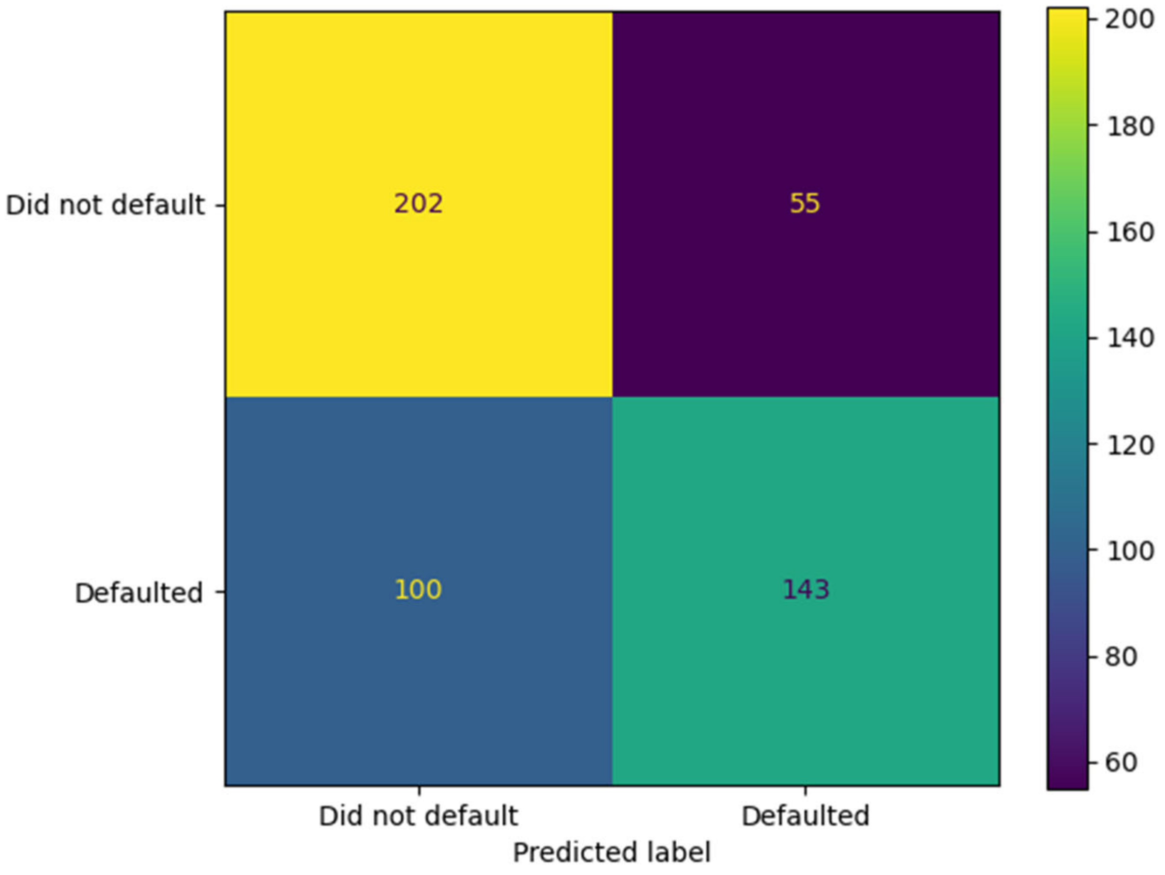
UCLAAnderson
SCHOOL of MANAGEMENT

a. Implement SVC

Classification (prediction): default or no default

```
# Implement SVC (support vector classification, classification using SVM)
clf_svm = SVC(random_state=42) # Untrained shell
clf_svm.fit(X_train_scaled, y_train) # Train!
```

```
# Confusion Matrix
plot_confusion_matrix(clf_svm,

X_test_scaled,
                      y_test,

values_format='d',

display_labels=["Did not
default",
"Defaulted"])family="binomial")
```

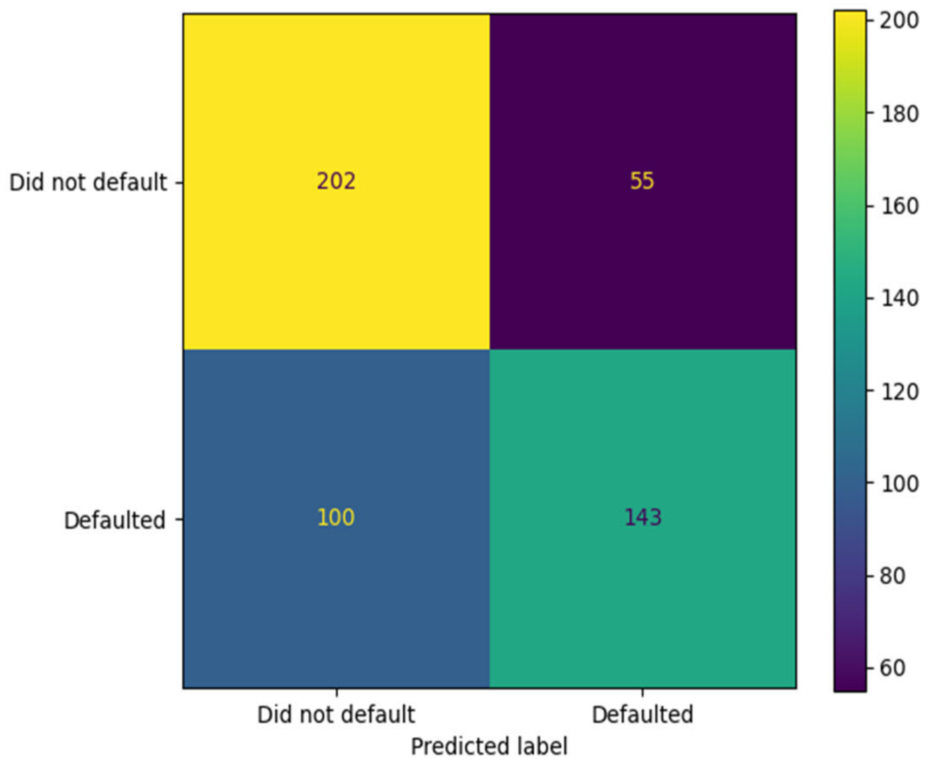


a. Cross-validation

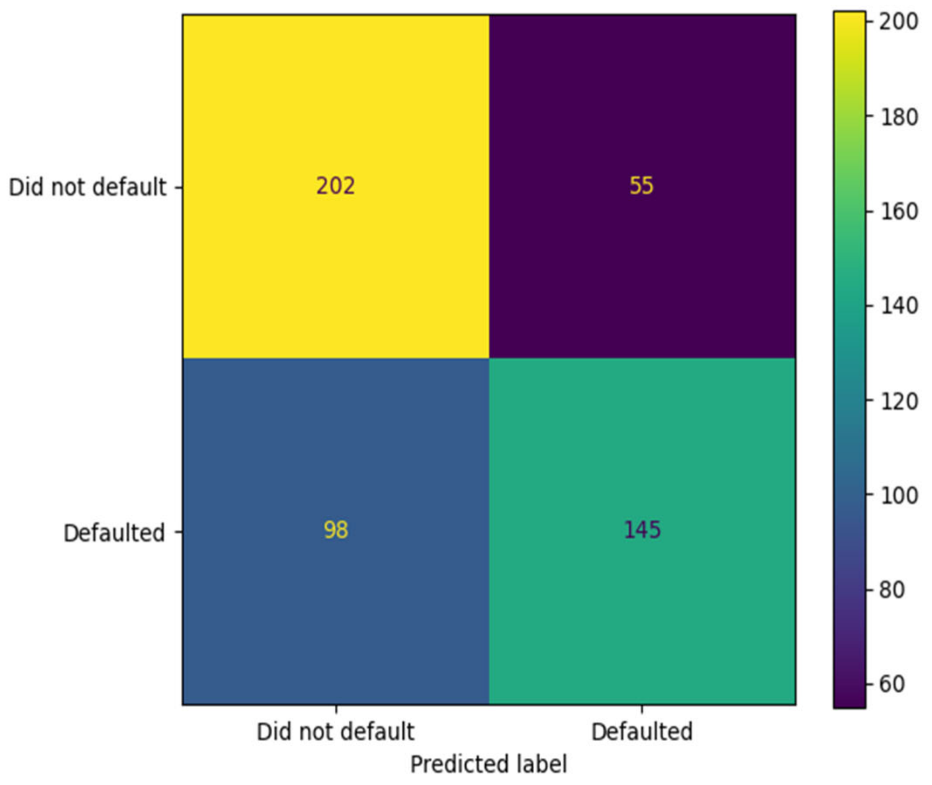
```
# Optimize parameters with cross-validation
# inverse of C governs strength of regularization, fuzzy boundary
# gamma is the kernel coefficient of the radial function since we choose rbf
# gamma = 'scale' implies 1/(nr features times variance of X variables)
param_grid = [
  {'C': [0.5, 1, 10, 100, 1000], # NOTE: Values for C must be > 0
   'gamma': ['scale', 1, 0.1, 0.01, 0.001, 0.0001],
   'kernel': ['rbf']},]

optimal_params = GridSearchCV(
        SVC(), # shell
        param_grid, # parameters
        cv=5, # number of folds to cross-validate
        scoring='accuracy', ## (accruacy is default scoring) Slightly improved, but hardly!!!
## For more scoring metrics see:
        ## https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter
        verbose=0 # NOTE: If you want to see what Grid Search is doing, set verbose=2
    )

optimal_params.fit(X_train_scaled, y_train)
print(optimal_params.best_params_)
-> C = 100
-> gamma = 0.001

# Reimplement SVM with optimized parameters
clf_svm = SVC(random_state=42, C=100, gamma=0.001) # specifiy the values
clf_svm.fit(X_train_scaled, y_train)
```

a. Cross-validation: slightly improved

Original



Cross-validated



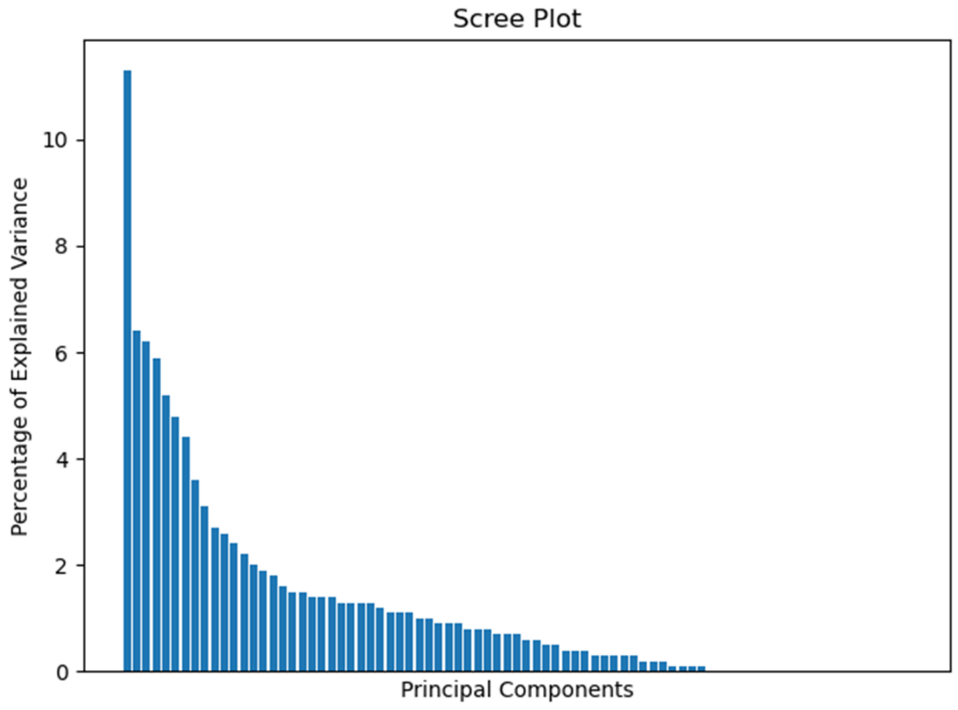
UCLA Anderson
SCHOOL of MANAGEMENT

a. Visualizing SVM

24 features = 24 dimensions (= head explodes)

- Classic trick: consider first couple of principal components of X
- Hope is: main variation in X also explains main variation in Y

```
pca = PCA() # NOTE: By default, PCA() centers the data, but does not scale it.
X_train_pca = pca.fit_transform(X_train_scaled)

per_var = np.round(pca.explained_variance_ratio_* 100, decimals=1)
labels = [str(x) for x in range(1, len(per_var)+1)]
```

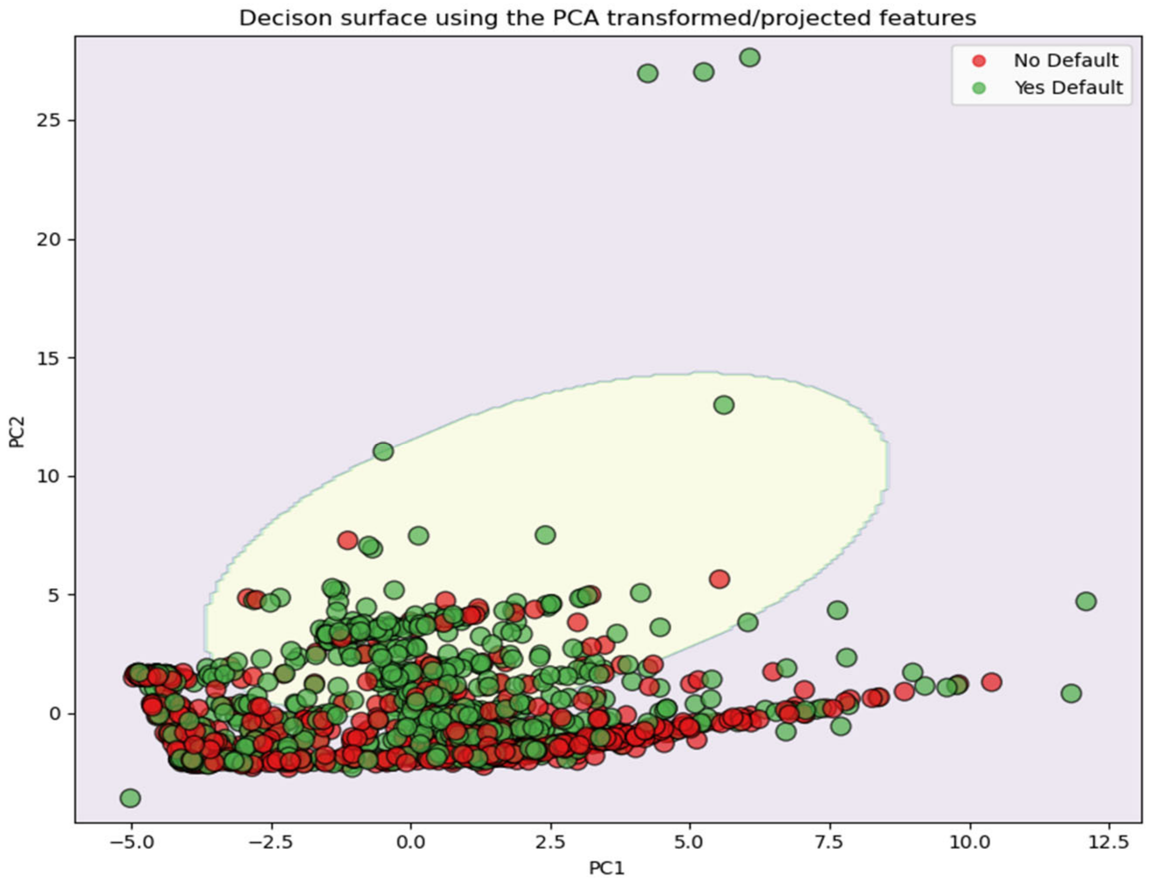


UCLA Anderson SCHOOL of MANAGEMENT

a. Visualizing SVM cont’d

Redo SVC fit exercise using only two X’s

- PC1 and PC2
- Plot data points (X’s) along with classification regions



UCLAAnderson
SCHOOL of MANAGEMENT

a. Oversampling

In the previous, we oversampled the default sample

- SVM sensitive to minority outcomes being ignored
  - Setting all observations as “No Default” may optimize the objective function if very few defaults in sample

- Oversampling ensures that each outcome gets equal (or at least more similar weights) in model training and testing
  - Issue: note that we are changing the unconditional probability of default in the sample we are looking at!

UCLA Anderson SCHOOL of MANAGEMENT

a. Oversampling: example

```
# Split data into two subgroups (default vs no default)
df_no_default = df_no_missing[df_no_missing['DEFAULT'] == 0]
df_default = df_no_missing[df_no_missing['DEFAULT'] == 1]

# Downsample the data (1000 random obs for each group)
df_no_default_downsampled = resample(df_no_default,
                                      replace=False,
                                      n_samples=1000,
                                      random_state=42)

df_default_downsampled = resample(df_default,
                                   replace=False,
                                   n_samples=1000,
                                   random_state=42)

# Merge two groups to have 2000 obs
df_downsample = pd.concat([df_no_default_downsampled,
df_default_downsampled])

# Split data into dependent and independent variables
X = df_downsample.drop('DEFAULT', axis=1).copy()
y = df_downsample['DEFAULT'].copy()
```

UCLA Anderson
SCHOOL of MANAGEMENT

a. Oversampling

Oversampling of defaults in your training sample will tend to lead to a model that overpredicts defaults, since defaults is the minority outcome

- Imagine a very simple model that just predicts the sample mean, with an equal sampling of the two outcomes (default and no default), this probability is 50/50

So how can we adjust for the bias we are introducing through oversampling?

- Think of unconditional (full sample) probability of default (22% in this sample) as a prior.
- Posterior probability of default is the model predicted probability times the prior probability of default, divided by the sum of the prior times the model probabilities
- Reference: Bayes rule discussion from earlier in class

$$PostProb_i = \frac{ModelProb_i \times Prior_i}{ModelProb_i \times Prior_i + (1 - ModelProb_i) \times (1 - Prior_i)}$$

a. SVM: Summary

Supervised learning technique

- Nonlinear classification
- Often quite good properties, but non-linearities (as with decision trees) often require more data in typical finance settings
- More flexibility = more parameters, can be harder to interpret findings
- A useful alternative to, say, logistic regressions