TRUMAN STATE UNIVERSITY

Professional Program in DATA SCIENCE

PDAT615G: Machine Learning

Module 3 – Support Vector Machines and Neural Networks

DISTINCT BY DESIGN

Support Vector Machines

So far, we've seen two classifiers that depend on a linear combination of inputs.

Logistic Regression

- $\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k \rightarrow$
- β's maximize likelihood of observed data/minimize a weighted least squared criterion.

Perceptron (w/ loss function from last time)

- $w_0 + w_1 x_1 + \cdots + w_k x_k \rightarrow$
- w's minimize distance of misclassified points to divider.
- Not unique if points linearly separable.

Maximum-Margin Classifier

- $\max_{|\vec{w}|=1} M$ such that $|\vec{w} \cdot \vec{x_i}| \geq M$ for all i.
- $\min_{\vec{w}} |\vec{w}|$ such that $|\vec{w} \cdot \vec{x_i}| \geq 1$ for all i.
- Not defined if groups aren't separable.

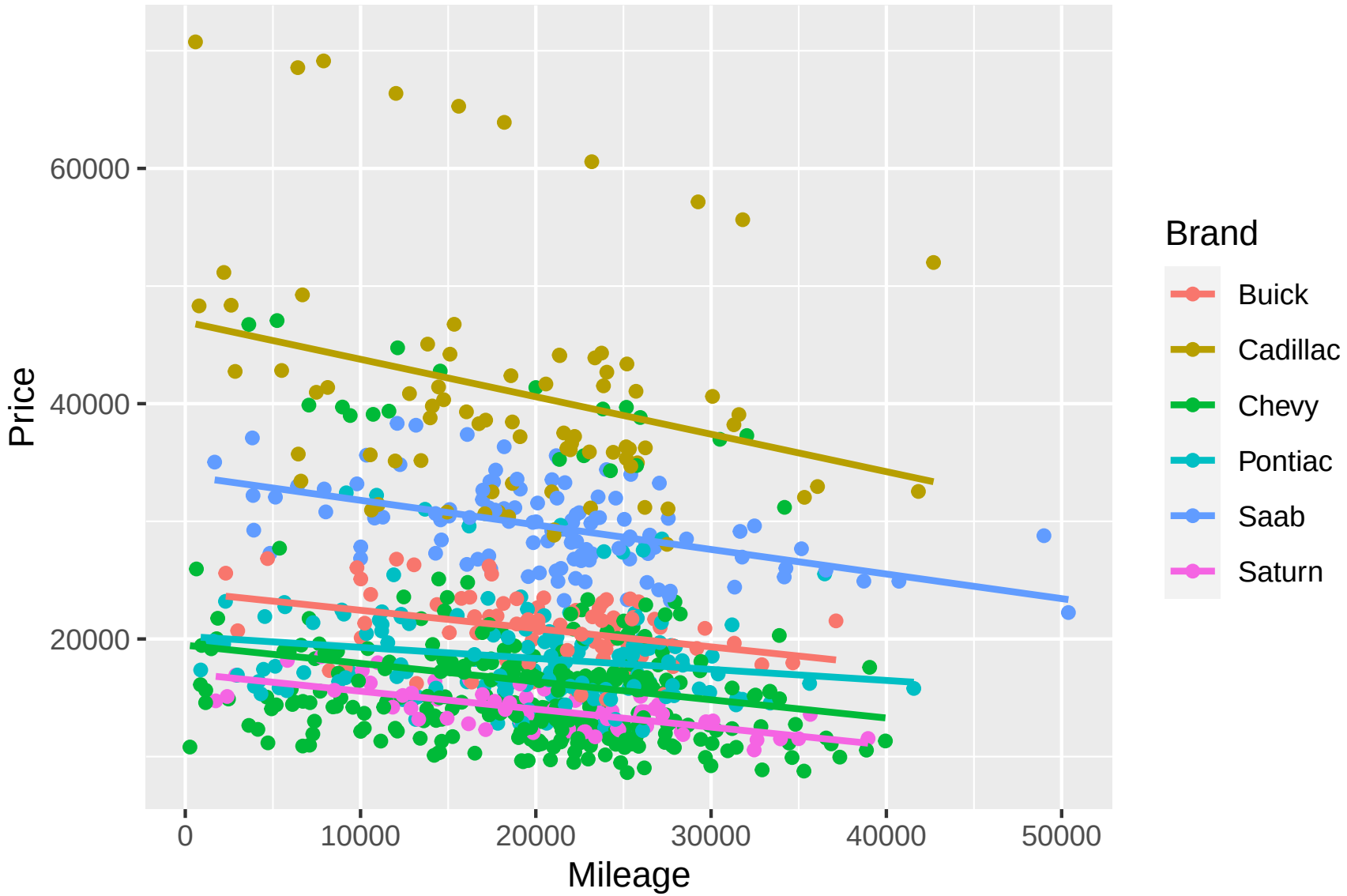
A *support vector classifier* maximizes margin between groups while allowing a "budget" for violations.

- Response is a $-1/1$ variable.
- $\max_{|\vec{w}|=1} M$ such that
  $y_i(\vec{w} \cdot \vec{x_i}) \geq M(1 - \varepsilon_i)$ for all i and $\sum_{i=1}^{n} \varepsilon_i \leq C$.
- $\min_{\vec{w}} |\vec{w}|$ such that
  $y_i(\vec{w} \cdot \vec{x_i}) \geq 1 - \varepsilon_i$ for all i and $\sum_{i=1}^{n} \varepsilon_i \leq C$.
- Note: For misclassified points, $\varepsilon \geq 1$.
- Can't have more than C misclassified points.
- Final classifier for a new point $\vec{x}$ is a sum of dot products with each original data point:

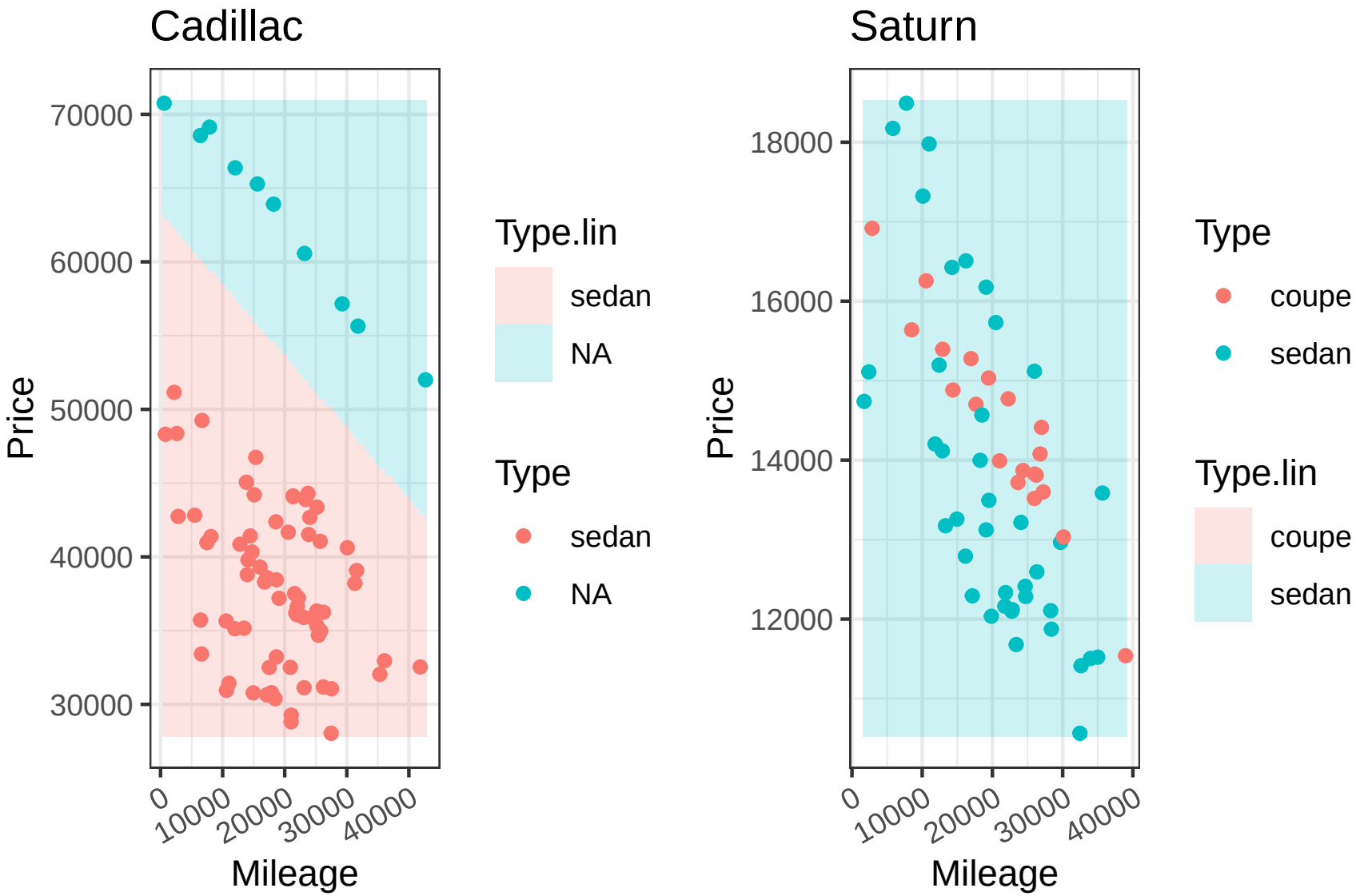
$$f(\vec{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i (\vec{x} \cdot \vec{x_i}).$$

- The α's are only non-zero on points within the margin–the classifier is not sensitive to changes in the easily-classified points.

Example: Predicting car type from mileage and resale value.

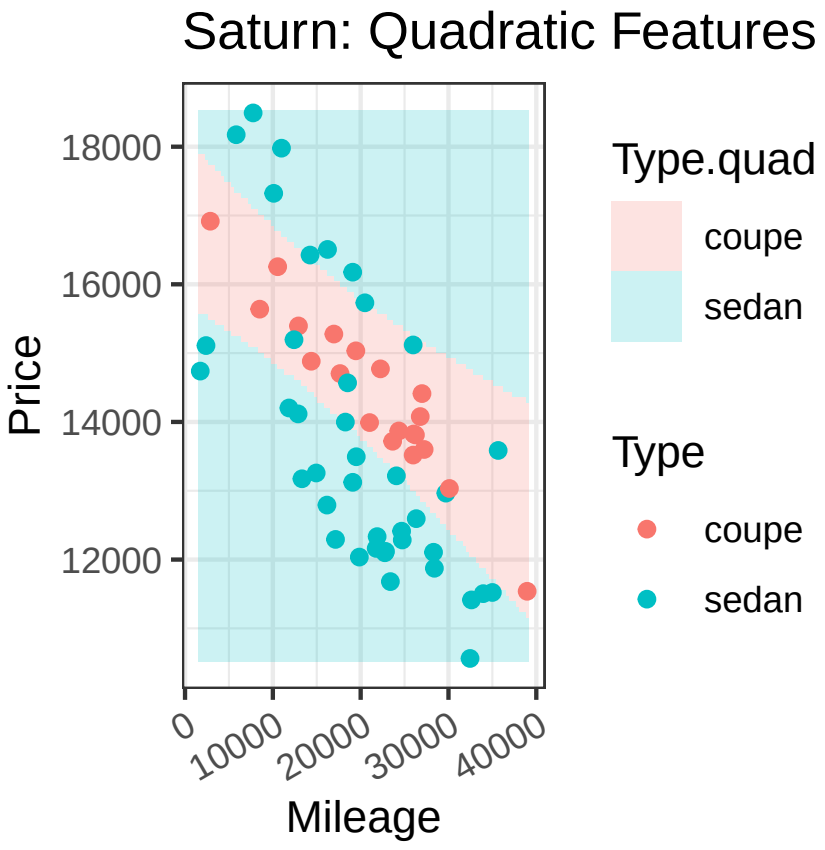


Linear support vector classifiers do well when a linear division exists, but can't handle other patterns.



Adding more features can create a classifier that's non-linear in the original variables.

Price	Mileage	Price^2	Mile^2	P*M
17.3	10.1	300.1	102.1	175.0
16.5	16.2	272.5	263.4	267.9
16.4	14.2	269.8	202.8	233.9
15.1	26.0	228.6	674.9	392.8
18.0	11.0	323.2	120.7	197.5
13.6	35.7	184.6	1271.8	484.5



The Big Idea: Choosing a different "inner product" can have the same effect as adding features.

- The classifier value depends on data points only through the dot product:

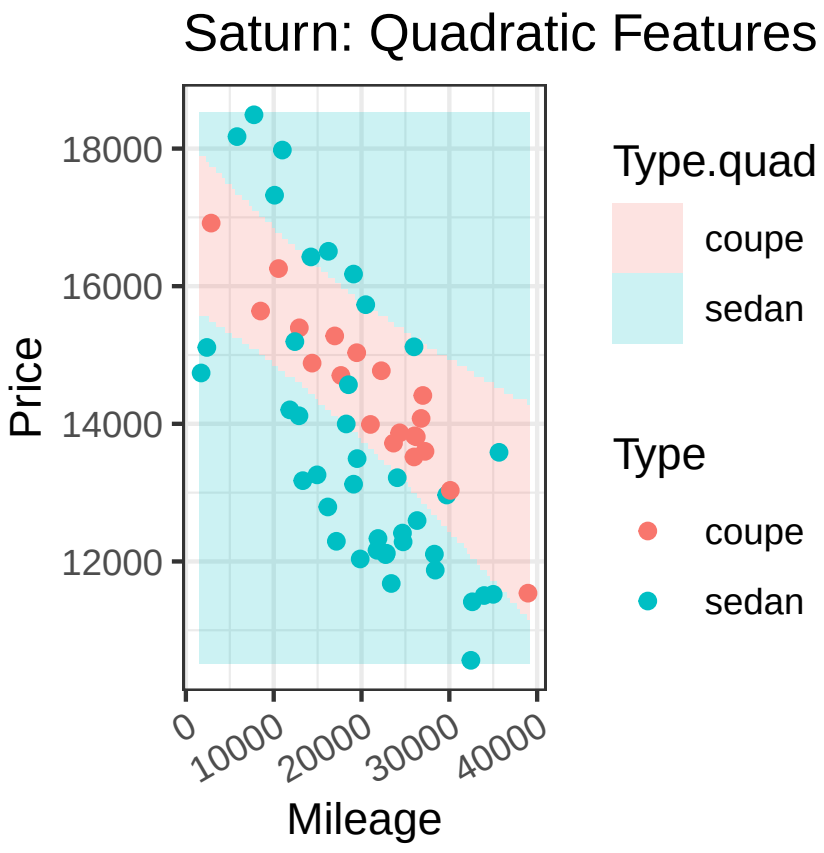
$$f(\vec{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i (\vec{x} \cdot \vec{x_i}).$$

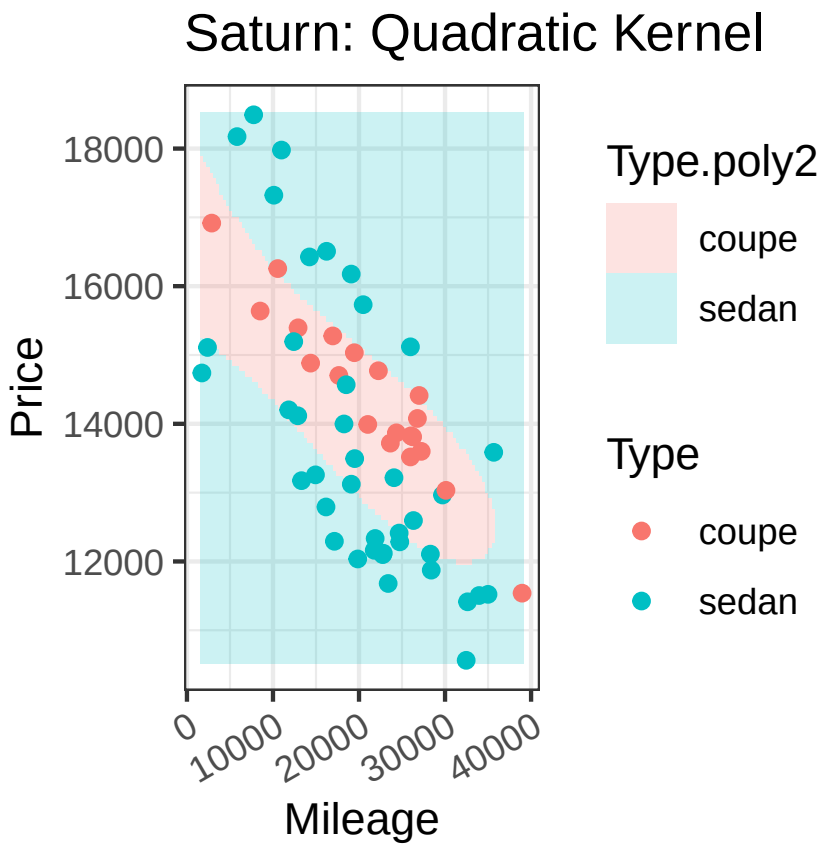
- It's possible (thanks, math!) to replace the "vanilla" dot product with another *kernel function* to create a new *inner product*.

$$f(\vec{x}) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(\vec{x}, \vec{x_i}).$$

- Kernel functions replicate the effect of embedding the points in a higher-dimensional space, where it might be easier to find a complete linear separation [e.g., $K(\vec{x}, \vec{x_i}) = (1 + \vec{x} \cdot \vec{x_i})^2$].

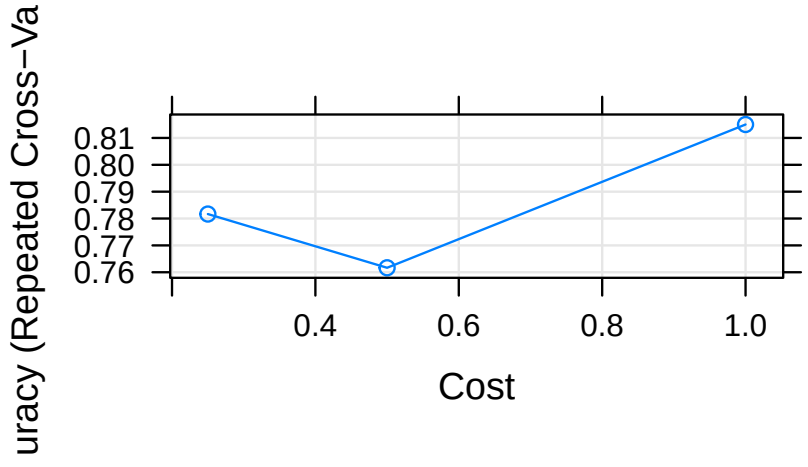
Example: Comparison of Quadratic Features and Quadratic Kernel

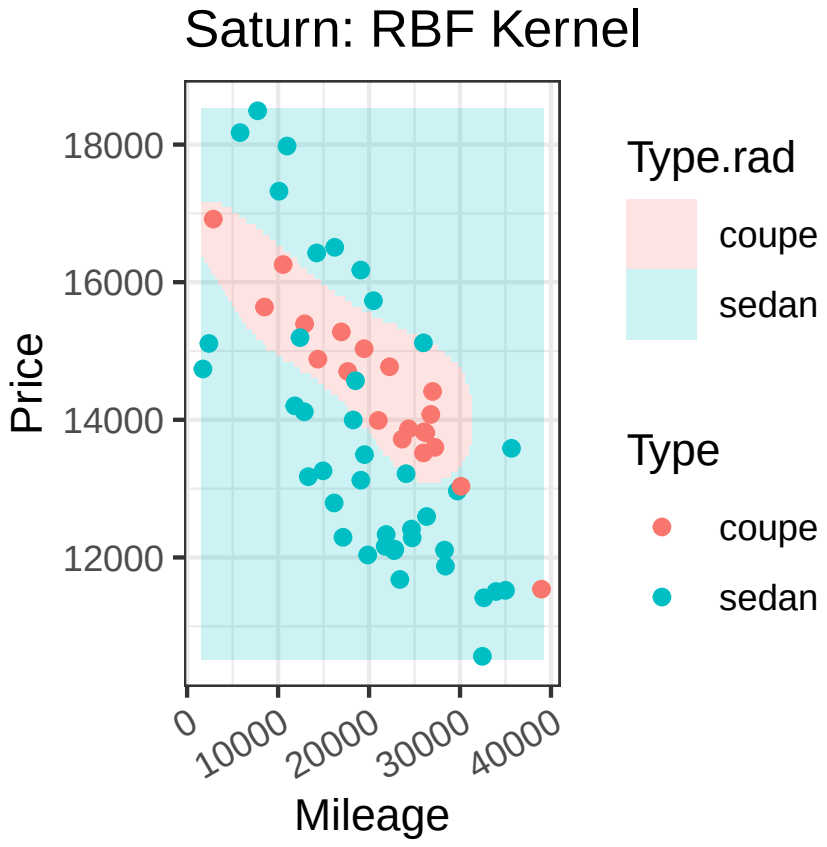




Example: The “radial basis function” kernel is often a good choice (using `train` with `svmRadial`).

```
svm.fit.rad <- train(
  Type ~ Price + Mileage, data=sat,
  method="svmRadial",
  trControl=trainControl(
    method="repeatedcv",
    number=10, repeats=10
  )
)
plot(svm.fit.rad)
```





For a response with $K > 2$ classes, multiple SVM's are created, using either one-vs-one or one-vs-all approaches.

One-vs-One classification creates $\frac{K(K-1)}{2}$ SVM's, each of which compares only two classes. Each SVM "votes" on which class should be the final classification. Default in `e1071` and `kernlab` implementations.

One-vs-All classification creates K SVM's, each of which compares one class to the rest. The final classification for a point $\vec{x}$ is determined by which of the K dividing hyperplane the point $\vec{x}$ is *farthest* from.