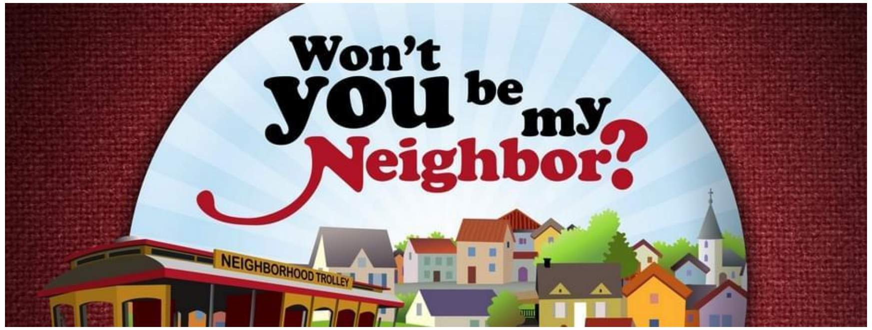
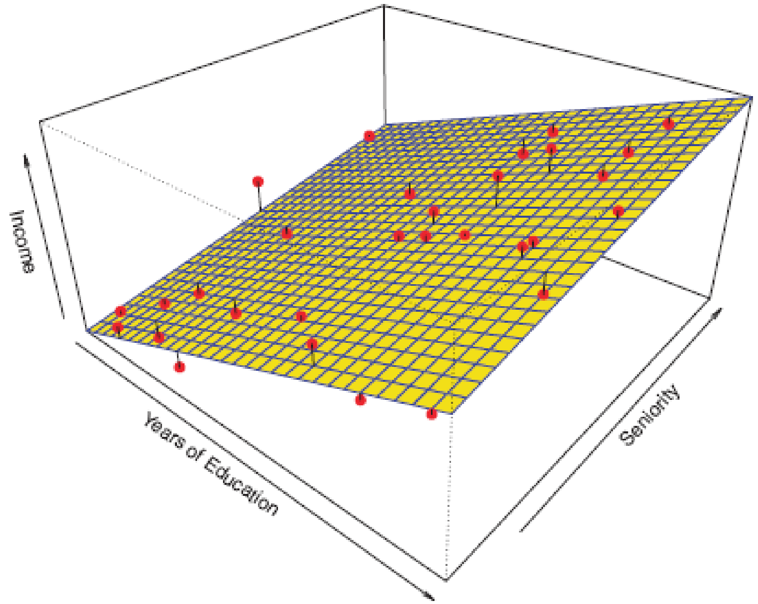
YData: Introduction to Data Science

Lecture 37: multiple regression

Overview



Conclusion on evaluating classifiers



Multiple regression

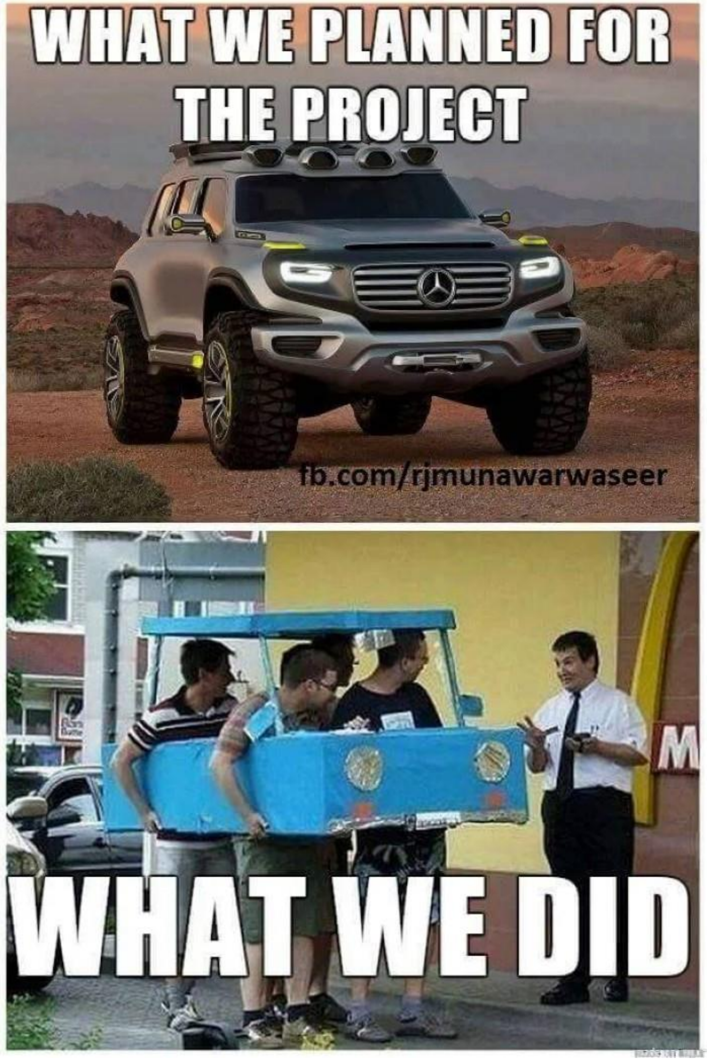
Announcements

Homework 11 has been posted

- It is due on Sunday May 1st

Project 3 is due Friday at 11pm

A practice final exam has been posted to Canvas



Prediction: regression and clasification

We “learn” a function f

- f(**x**) $\longrightarrow$ y

Input: **x** is a data vector of "features"

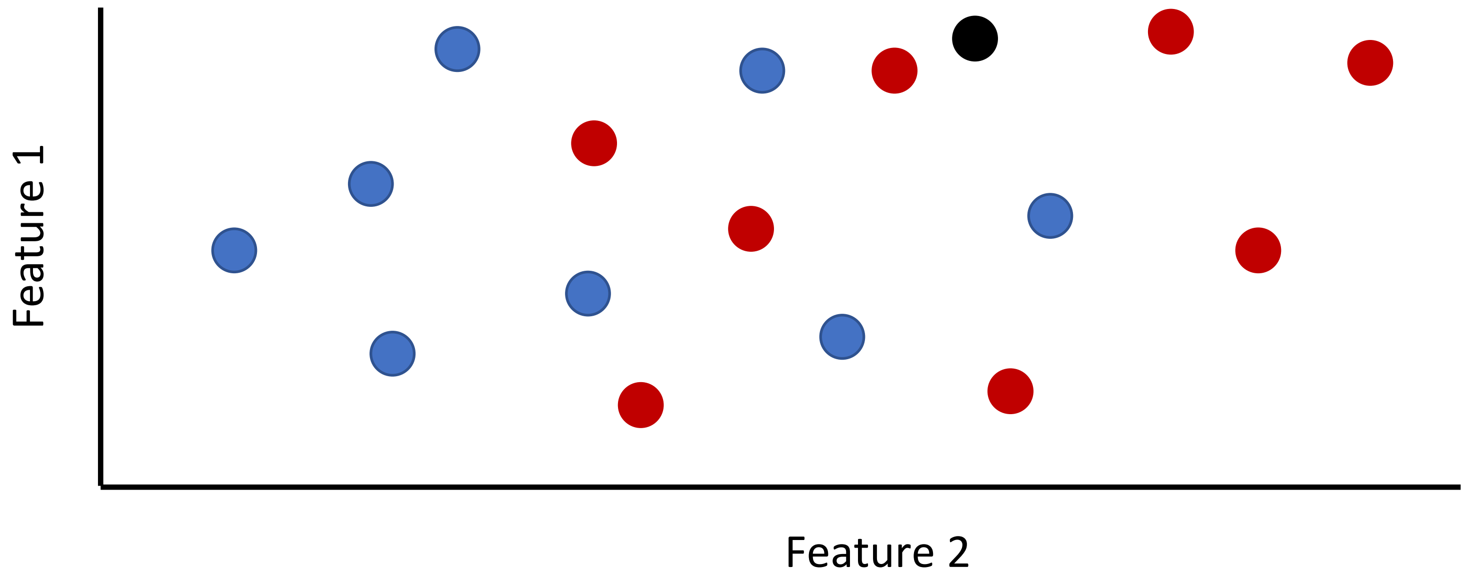
Output:

- Regression: output is a real number ($y \in R$)
- Classification: output is a categorical variable y_k

Finding the k Nearest Neighbors (k ≥ 1)

To classify a point:

- Find its k nearest neighbors
- Take a majority vote of the k nearest neighbors to see which of the two classes appears more often
- Assign the point the class that wins the majority vote



Accuracy of a classifier

The accuracy of a classifier on a labeled data set is the proportion of examples that are labeled correctly on the *test set*

If the labeled data set is sampled at random from a population, then we can infer accuracy on that population

Let's explore this in Jupyter!

Evaluation

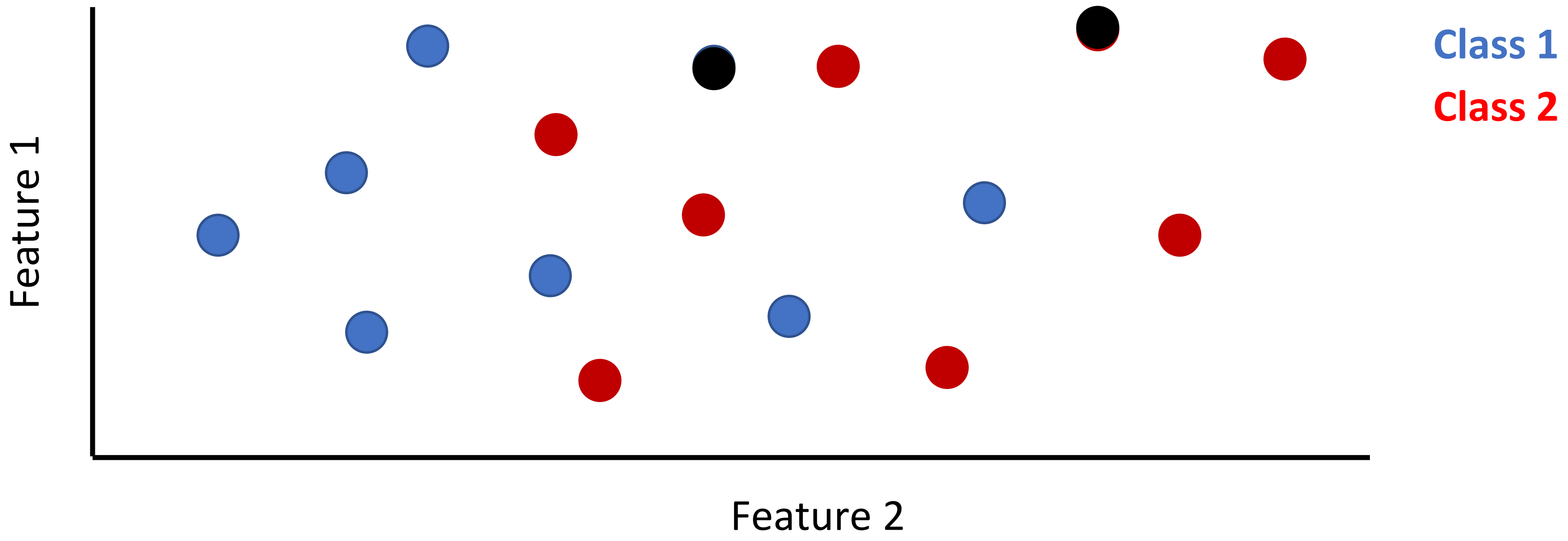
Training and test accuracy

Q: What would happen if we tested the classifier using the training data with k = 1?

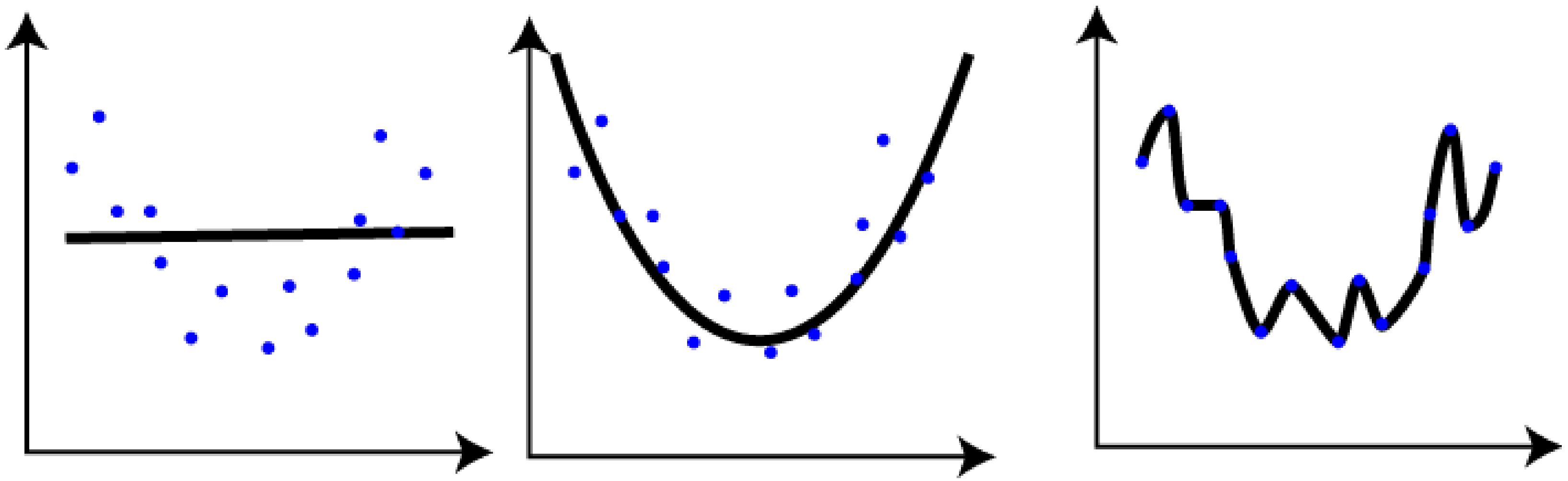
A: We would have 100% accuracy

Q: Would this indicate that the classifier is good?

- A: No!



Overfitting



Overfitting

If our classifier has over-fit to the training data then:

a. We might not have a realistic estimate of how accurate its predictions will be on new data

b. There might be a better classifier that would not over-fit to the data and thus can make better predictions

What we really want to estimate is how well the classifier will make predictions on new data, which is called the **generalization (or test) error**

Overfitting song…

Cross-validation

Training error rate (training accuracy): model predictions are made on using the same data that the model was fit with

Test error rate (test accuracy): model predictions are made on a separate set of data

Multiple regression

Prediction: regression and clasification

We “learn” a function f

- f(**x**) $\longrightarrow$ y

Input: **x** is a data vector of "features"

Output:

- Regression: output is a real number ($y \in R$)
- Classification: output is a categorical variable y_k

Multiple regression

In multiple regression we try to predict a quantitative response variable *y* using several features $x_1, x_2, \ldots, x_k$

For multiple linear regression, the underlying model is:

$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \ldots \beta_k \cdot x_k + \epsilon$$

We estimate coefficients using a data set to make predictions ŷ

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \hat{\beta}_2 \cdot x_2 + \ldots + \hat{\beta}_k \cdot x_k$$

Multiple regression

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \hat{\beta}_2 \cdot x_2 + \ldots + \hat{\beta}_k \cdot x_k$$

There are many uses for multiple regression models including:

- To make predictions as accurately as possible
- To understand which predictors (x) are related to the response variable (y)

Multiple regression

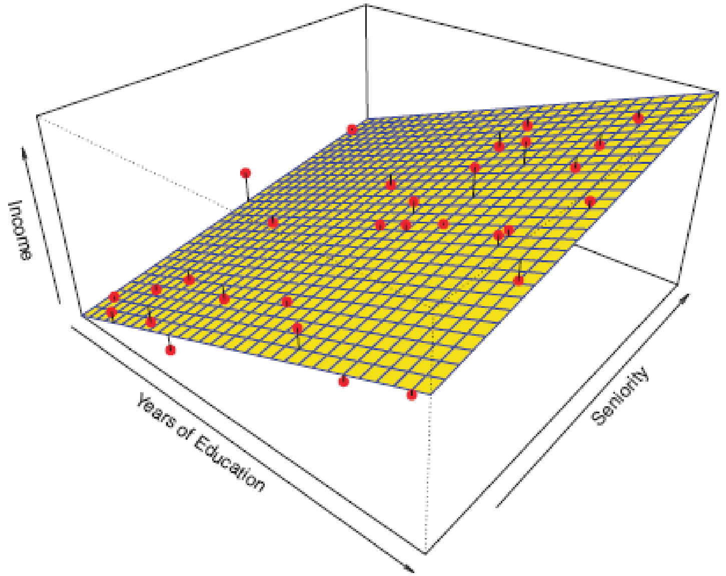
$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \hat{\beta}_1 \cdot x_2$$

$$\text{sales price} = \hat{\beta}_0 + \hat{\beta}_1 \cdot \text{square-footage} + \hat{\beta}_1 \cdot \text{year-built}$$

The coefficients ($\hat{\beta}_i$) are found by minimizing the RMSE

- i.e., we can use the minimize() function

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$



Let's explore this in Jupyter!