

# Prediktiv analys

FÖRELÄSNING 5

# Dagens fråga



# Dagens agenda

- Regression i Python
- KNN
- Model evaluation för regression
  - Cross validation
  - Overfitting
  - Bias-variance tradeoff
  - Overfitting
  - Regularization
  - Feature selection
- Lasso regression
- Diamond prices
- Post popularity

# Förra föreläsning

- Regression
- The multiple regression model
- Ordinary least square
- Error metrics för regression
  - Mean squared error
  - Root mean squared error
  - Mean absolute error
  - R-squared
  - Explained variance
- Predicting crime

# Vi följer specifika steg för att bygga en Machine learning modell



Detta är generella steg och man kan få hoppa fram och tillbaka flera gånger innan det blir rätt.

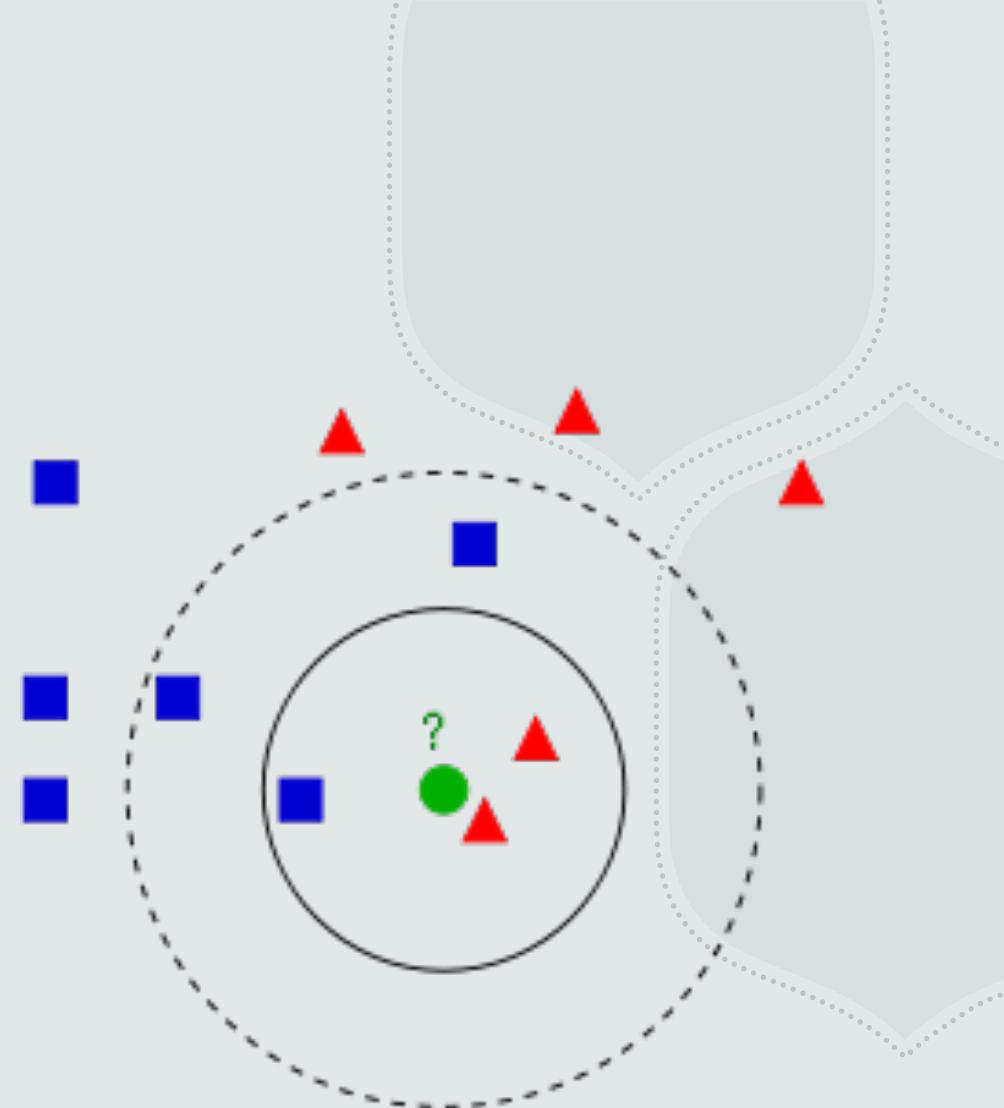
# K-Nearest Neighbor



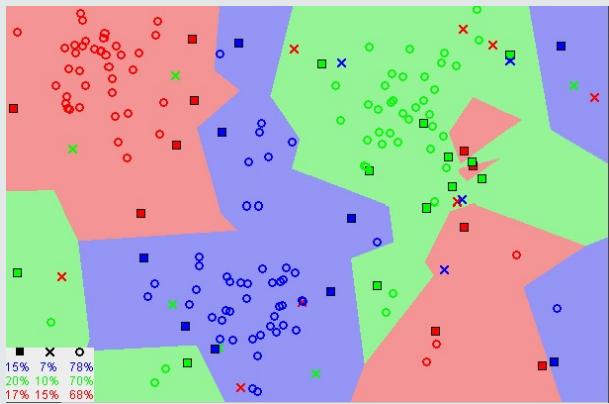
K-NEAREST NEIGHBOR  
(KNN) MODELLEN FÖR  
REGRESSION



K-NEAREST NEIGHBOR I  
SCIKIT-LEARN



# Intuitionen bakom K-Nearest Neighbor modellen



Baserad på en intuitiv idé:

- ”Observationer av egenskapers värden som ligger nära varandra, kommer att ha mål(target) värden som också ligger nära varandra”

Så principen bakom KNN modellen är att:

- För att kunna prediktera för en ny observation så måste man först hitta tillräckligt med fördefinierade träningsexempel ”K” som är nära i distans till den nya observationen och sedan använda target värden för KNN för att prediktera målet för den nya observationen.
- Tex om du försöker förutse bostadspriser så kan man utgå ifrån att lägenheter med liknande egenskaper och områden kommer ha liknade priser.

# Steg i K-Nearest Neighbor modellen

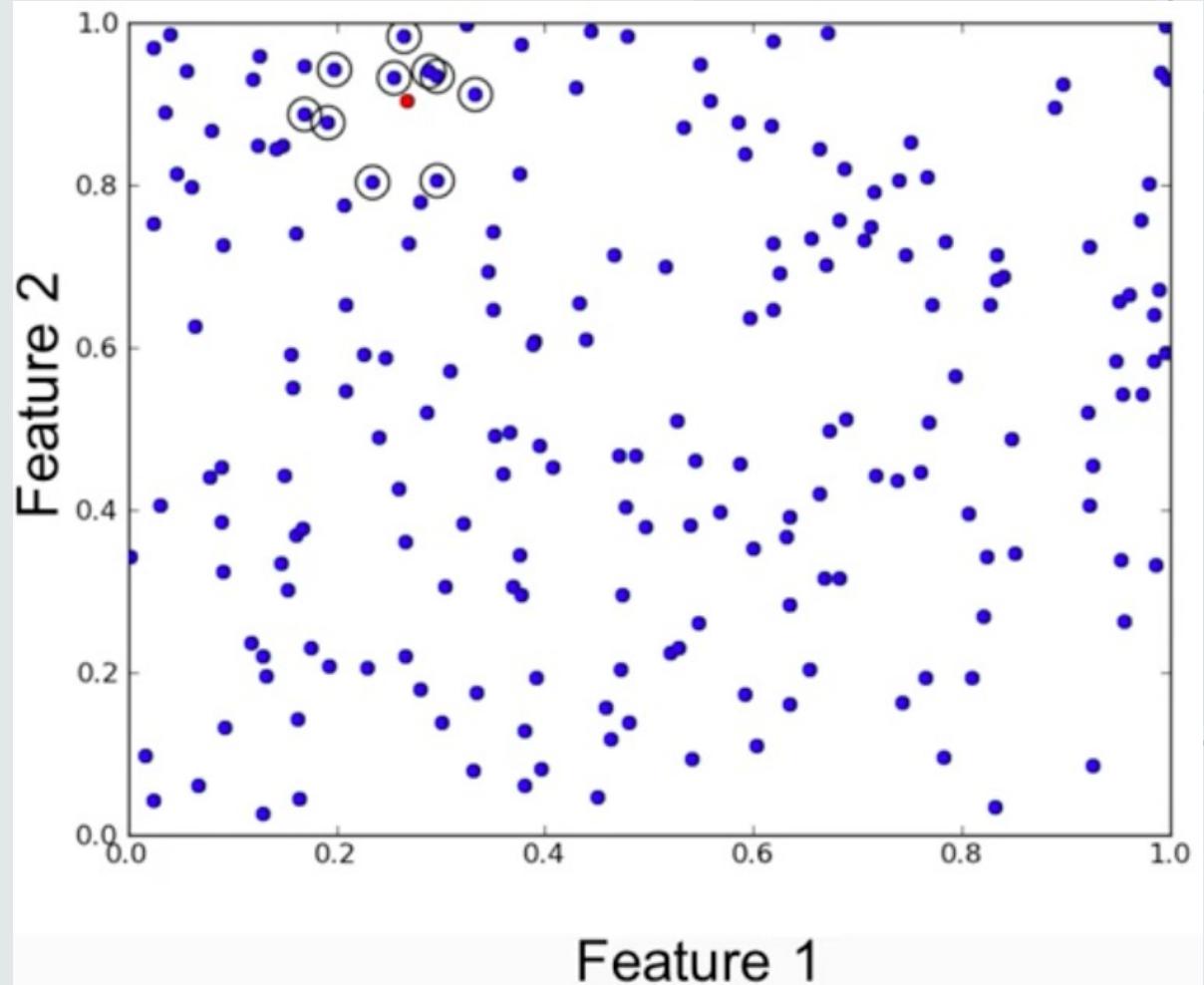
1. Välj ett antal "neighbors" K
2. Välj ett sätt att räkna distans (Euklidisk distans är vanligt)

För varje datapunkt i träningsdatan:

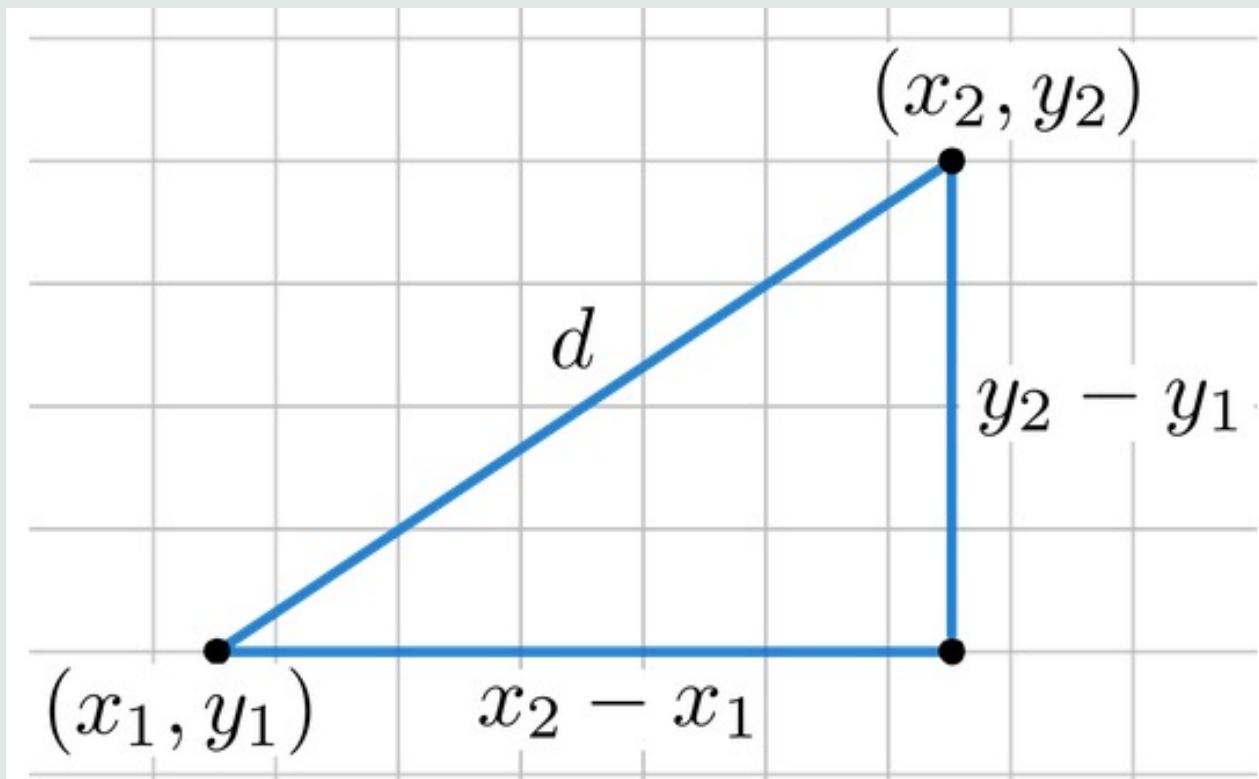
1. Beräkna distansen mellan datapunkten och alla andra observationer
2. Hitta de närmsta K observationer till datapunkten, dessa är K-neighbors
3. Hämta target(output) värdet för de K närmsta observationerna
4. Prediktera output värdet för datapunkten genom att beräkna medelvärdet av output värdet till K närmste grannarna

$$y_{pred} = \frac{1}{K} \sum_{i=1}^K y_i$$

Exempel med 2 egenskaper och K = 10



# Euclidisk distans



- Avstånd mellan två punkter i 2-dimensioner (från Pythagoras sats)

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



## Steg i ”K-Nearest Neighbor” modellen

- Säga för K=10
- Predikterat värde = medelvärdet = 0.161

Träningspunkt	Target värde
Granne 1	0.20
Granne 2	0.10
Granne 3	0.15
Granne 4	0.22
Granne 5	0.23
Granne 6	0.20
Granne 7	0.15
Granne 8	0.14
Granne 9	0.10
Granne 10	0.12

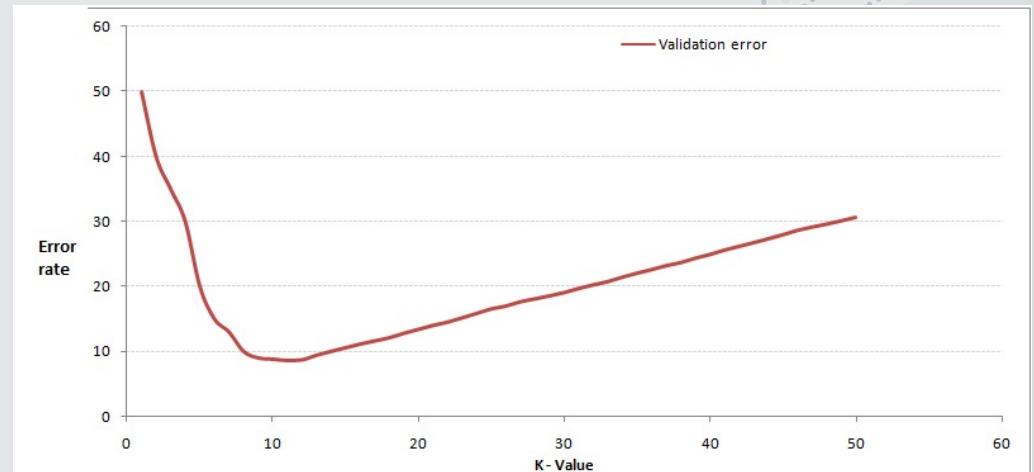
# Antaganden KNN

KNN gör inga antaganden om underliggande distributionen till datan

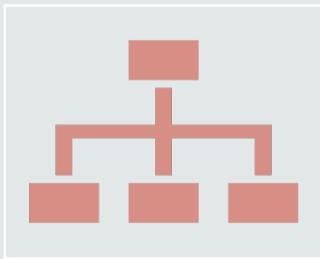
Datapunkterna finns i ett metrisk utrymme – det finns en avstånd som kan mäts mellan punkterna

# Saker att tänka på med KNN

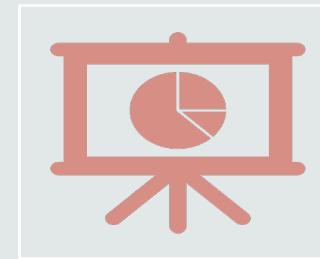
- Algoritmen är enkel och lätt att implementera
- Man kan använda KNN för regression och klassificering
- Algoritmen blir markant längsammare med ökat antal grannar K och datapunkter
- För att välja rätt antal grannar K kan man köra algoritmen flera gånger för olika värden på K. Välj det K som får lägst prediction error
- Lågt värde på K kan ge ostabil modell. *Tänk 1-2 grannar*
- För stort värde på K ger också hög andel fel. *Tänk för många datapunkter är med*



# K-Nearest Neighbor i scikit-leran



Använd "KNeighborRegression estimator" (från  
skleran.neighbor)



Viktiga parametrar

N\_neighbors: K = antal grannar

Weights: Viktningsfunktion för prediktering:

- "uniform"
- "distance"
- "user-defined"

Metric: Metrisk distans att använda (populära):

- "minkowski"
- "euclidian"

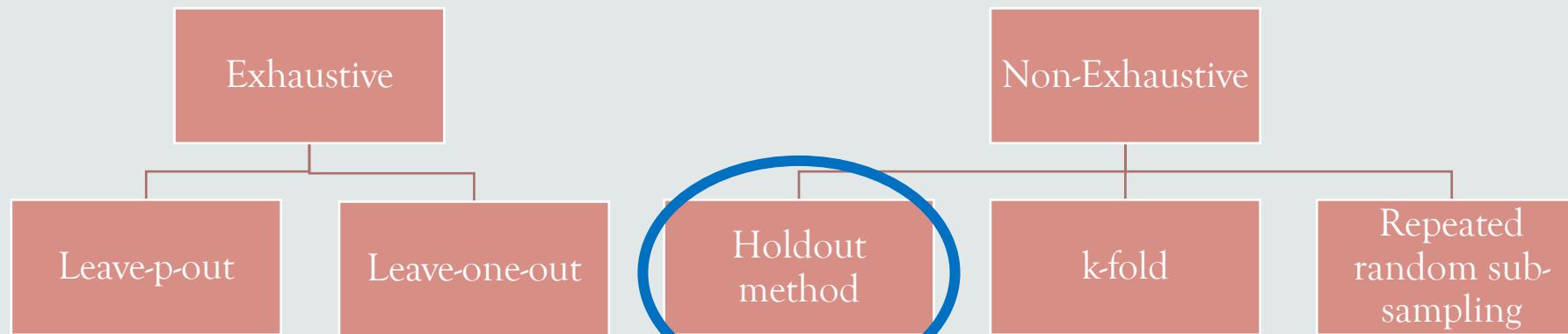
# Model Evaluation for Regression

Hur man evaluerar regressionsmodeller:

- Cross-validation
- Overfitting
- Regularization
- Feature selection

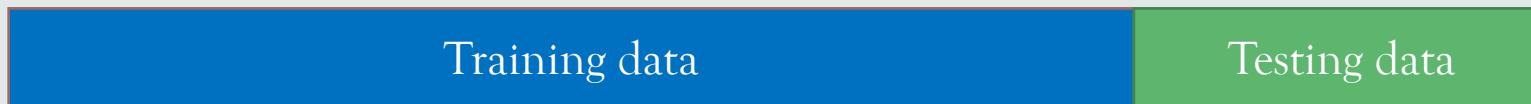
# Hur man evaluerar

- Målet med prediktiv analys är att få predikteringar om okända händelser
- Vi vill ha modeller som är generella mot den data som modellen inte sett tidigare
- För att estimera hur vår modell kommer prestera med data den inte sett tidigare använder vi en teknik kallad ”cross-validation”



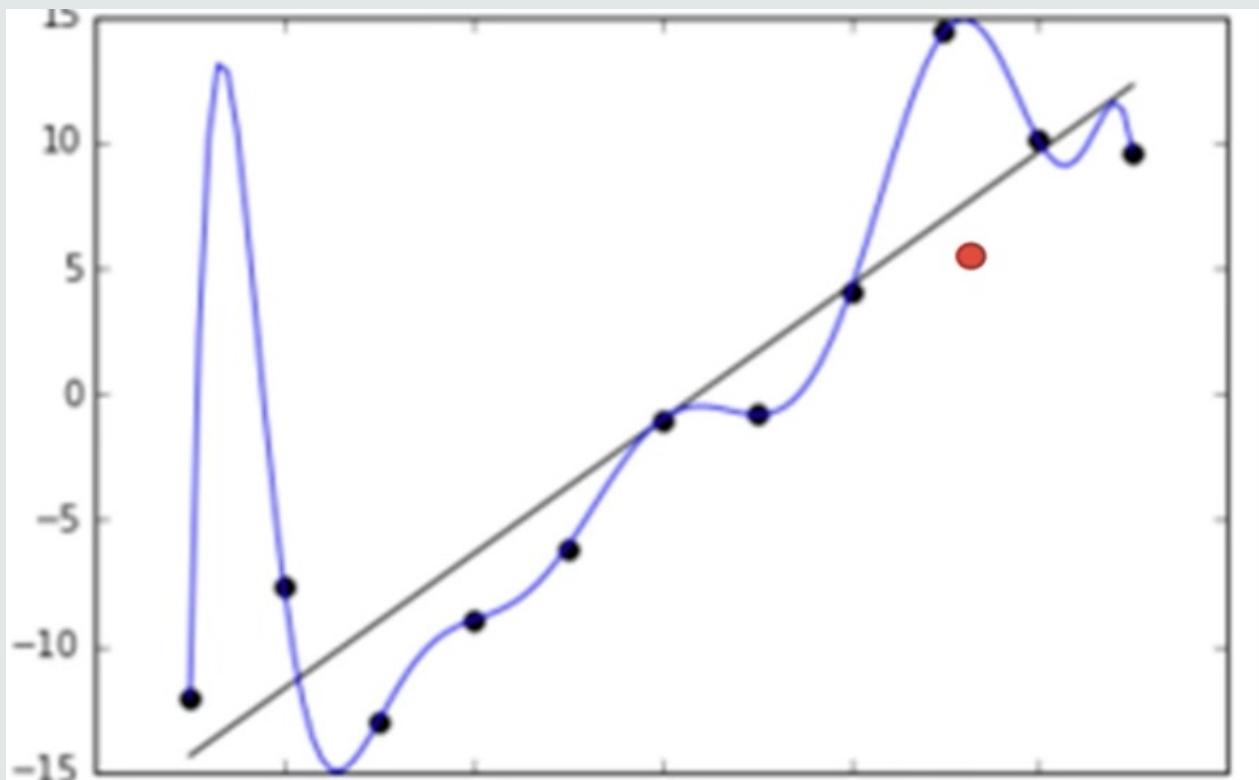
# Holdout Cross-Validation

- "Holdout" betyder att man undanhåller en viss procent av träningsdatan, vanligtvis mellan 10-40%, 20-25% är standard. Denna del kallas **testing** dataset ( $x_{\text{test}}$ ,  $y_{\text{test}}$ ). Vi evaluerar modellen med testdatan.
- Resten av datan används för att träna modellen för att hitta rätt parameter till modellen. Denna del kallas **training** dataset ( $x_{\text{train}}$ ,  $y_{\text{ttrain}}$ ).

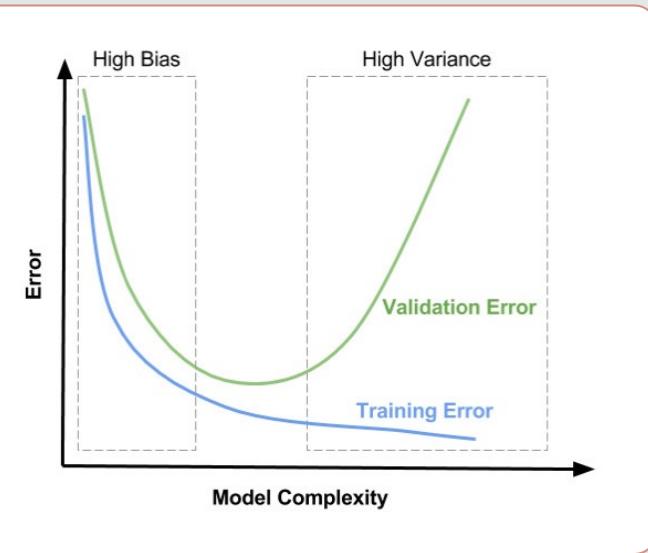


- Observationerna slumpas random in i respektive grupp

# Overfitting

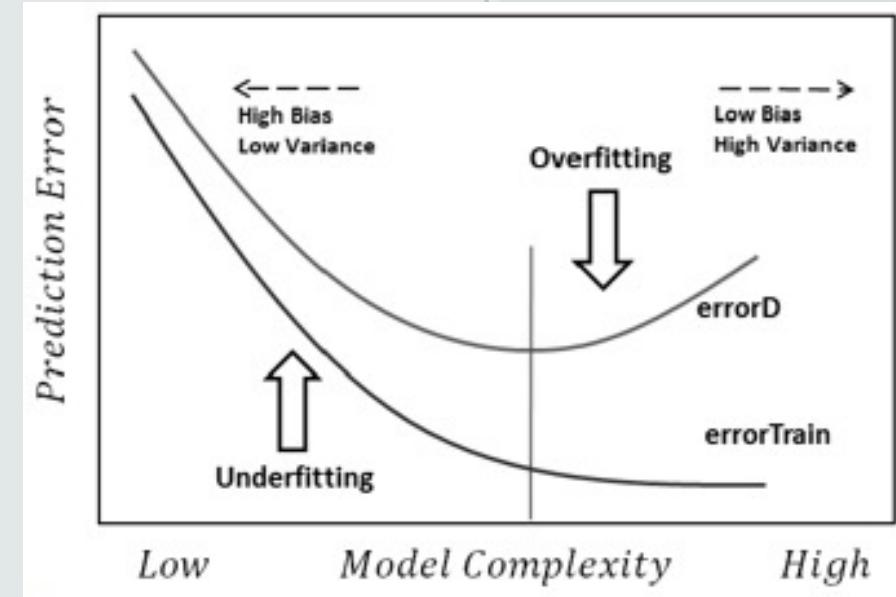
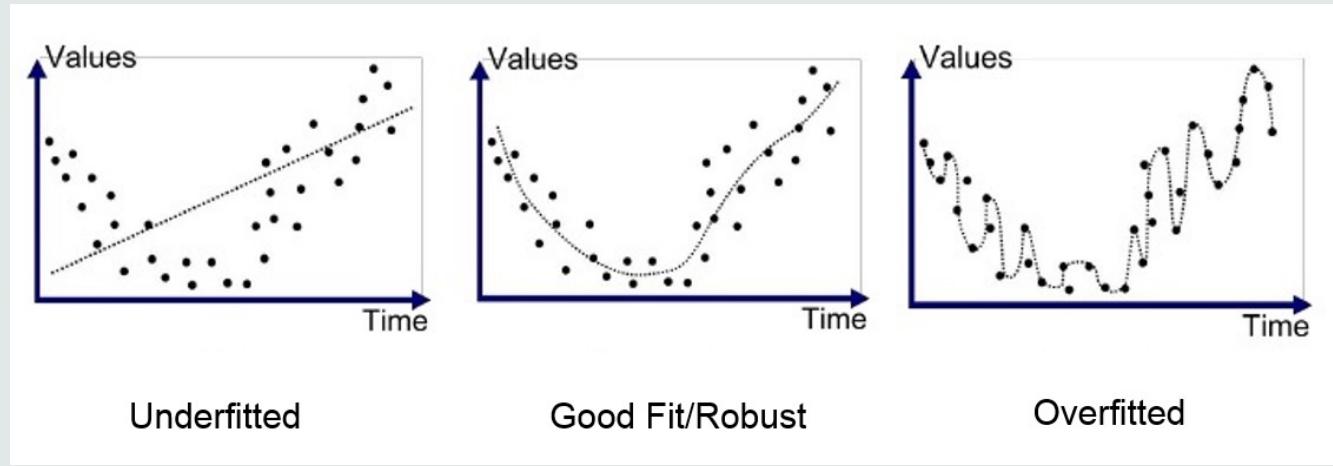


- Modellen lär sig alla aspekter av träningsdata inklusive slumpmässiga avvikelse.
- Som konsekvens av detta så kommer modellen ge dåliga predikteringar för osedd data.
- Modellen blir för komplex och kan inte prediktera verkligheten
- Den röda punkten kommer liga närmare verkligheten med den raka modellen än den mer komplexa predikterade blåa modellen.



- När man gör prediktion är det viktigt att förstå prediktion error med bias och varians
- Man vill minimera bias och varians, men det måste vara en balans mellan de två
- **Bias** – skillnad mellan genomsnitt prediktion och korrekt värde, alltså skillnad mellan  $\hat{y}$  och  $y$ . Modeller med hög bias anpassar inte modellen till träningsdatan. Vi har en för simpel modell. Resultat högt fel för träning- och testdata.
- **Varians** – variationen i modellprediktionen för ett givet datapunkt. Spridningen av datan. Modeller med hög varians anpassar till träningsdatan för mycket – inte generell nog för data den inte har sett tidigare. Resultat lågt fel träning men högt fel test.
- En bra modell hittar balansen mellan för hög bias och för hög varians för att varken overfit eller underfit modellen

## Bias-variance tradeoff



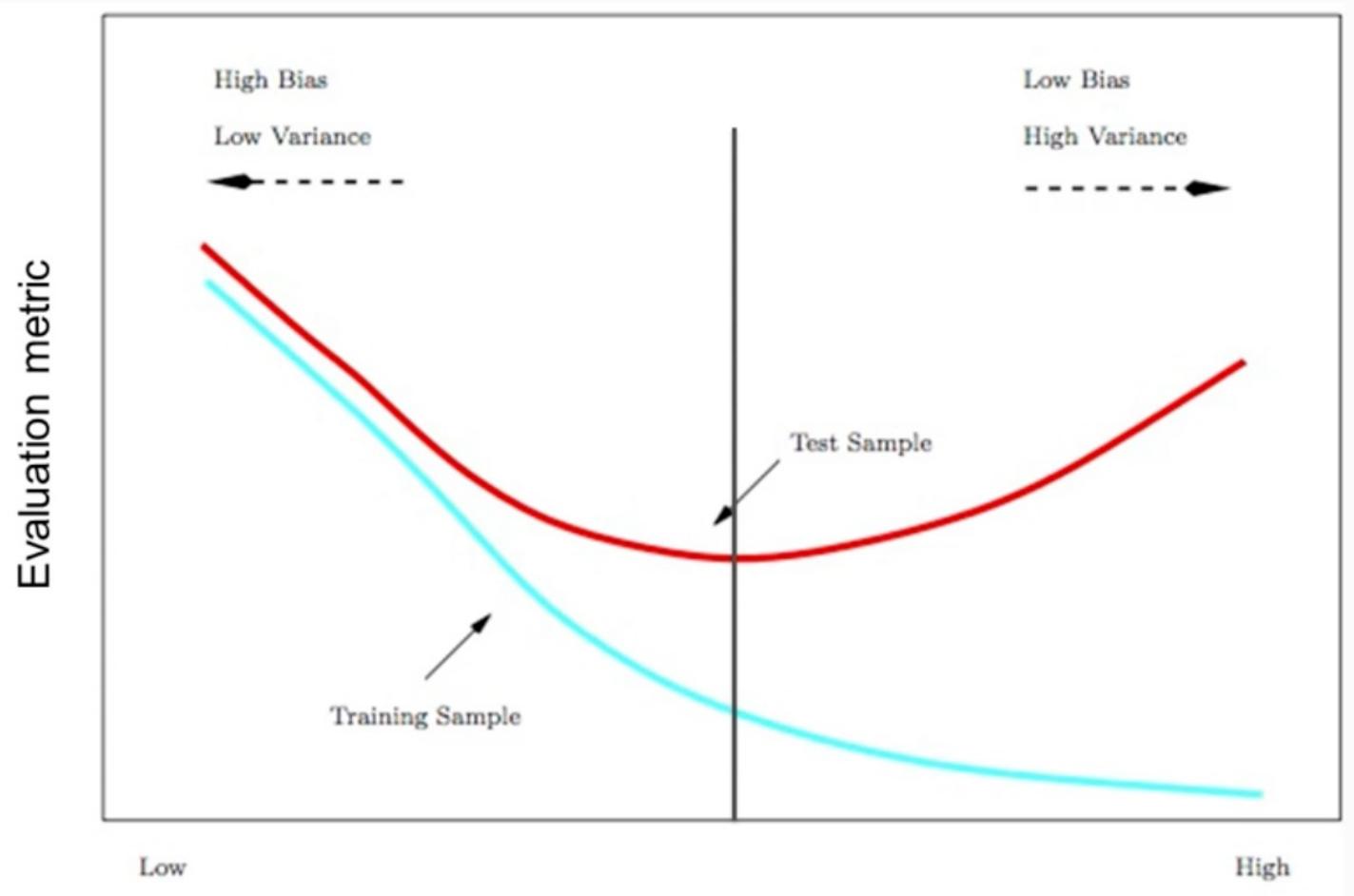
# Overfitting - Underfitting

Evaluering av modell – har den lärt ordentlig från sättet datan var delad i träning och test data?

1. **Underfitting** – high bias. Modellen lärde sig inte bra av datan och kan inte prediktera, även på träningsdata.
2. **Overfitting** – high variance. Modellen lärde för bra av träningsdata och är inte generell nog för ny data.
3. **Balans** rätt balans mellan bias och varians. Lärde bra under träning och kan också prediktera bra.

# Overfitting – Modell komplexitet

Det är större risk att overfitting  
händer i mer komplexa modeller



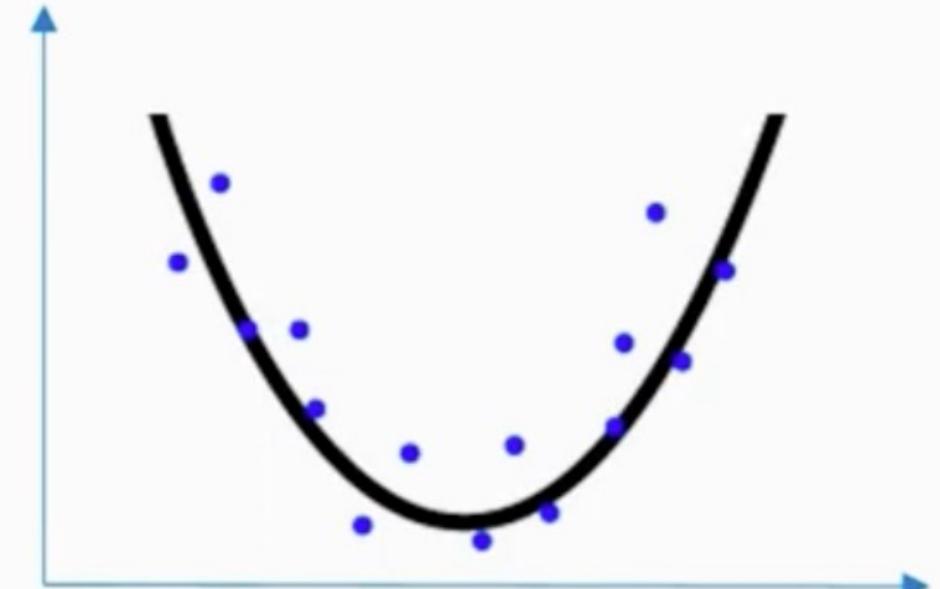
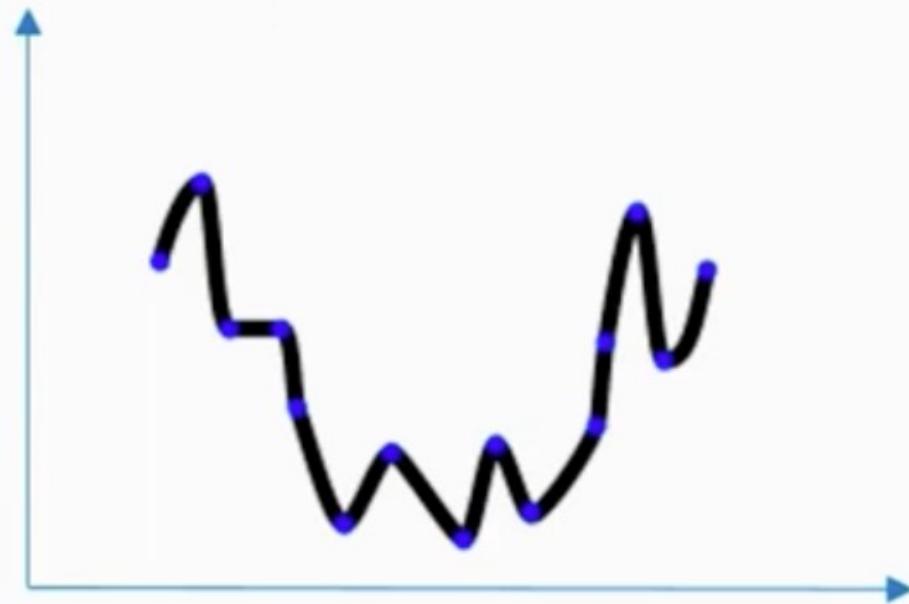
# Regularization

Regularization är en teknik  
för att förhindra Overfitting

Overfitted model

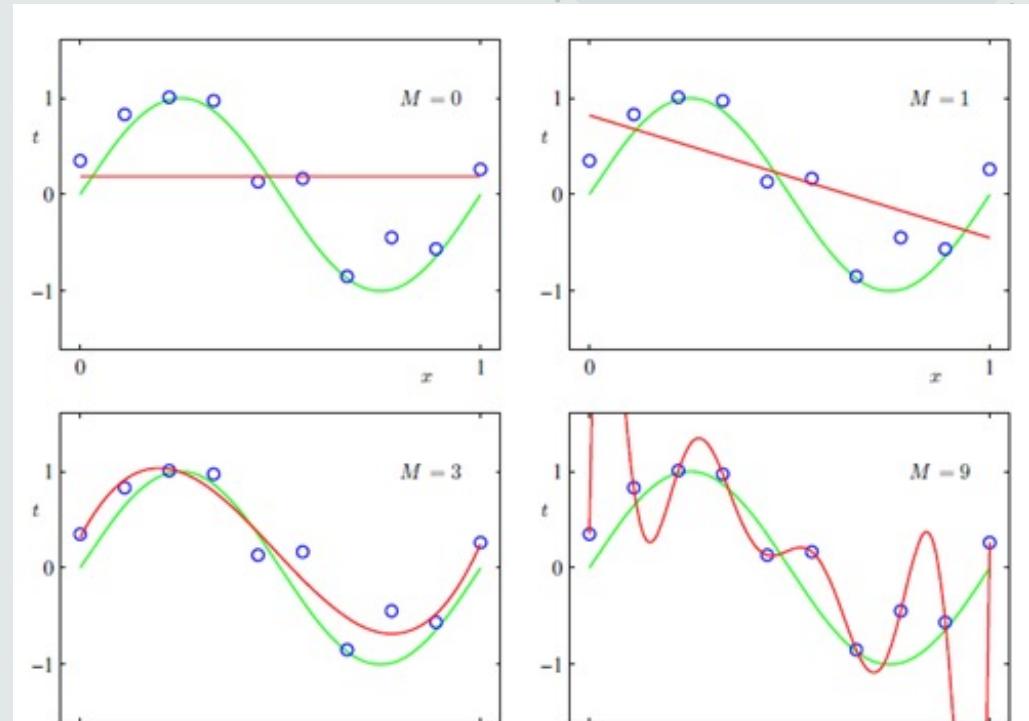


After regularization



# Regularization

- Regularization är tekniker för att reducera fel när man tränar en modell samtidigt som man undviker overfitting
- Fallet med bilden så har ett polynom av grad 9 fångat upp alla datapunkter men är för komplex till att prediktera ny data korrekt
- Regularization lägger till ett straff för att undvika detta

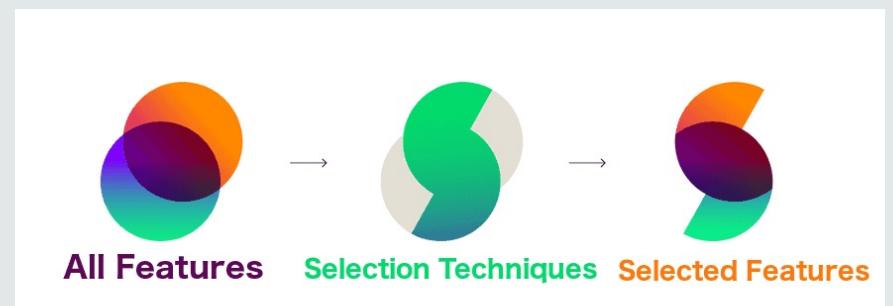


# Feature selection

- **Feature** är det samma som en *input variabel*. Vi kan också beteckna det som en *attribut* eller namnet på en *kolumn*.
- I machine learning använder man namnet features
- En feature är en mätbar egenskap hos något som observeras. Tex petal length, width
- Det är viktigt att välja **informativa** och **oberoende** features till sin modell
- **Informativa** betyder de ska förklara observationen, **output variabeln**, väl
- **Oberoende** betyder att man inte ska ha med lika features som förklarar outputen på samma sätt

# Feature selection

- Irrelevanta och delvis relevanta features kan ha **negativ** effekt på prediktionen!
- Vi behöver alltså välja vilka features (attribut) vi ska ha med i modellen
- Vid att ta bort irrelevanta och mindre viktiga features undviker vi overfitting
- Vi använder feature selection för att bestämma vilka features som är bra att använda
- Kan göras manuellt eller automtisk



# Lasso regression



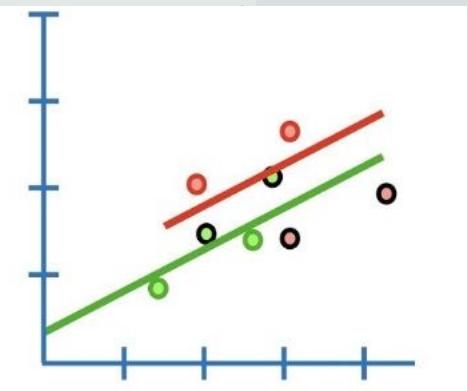
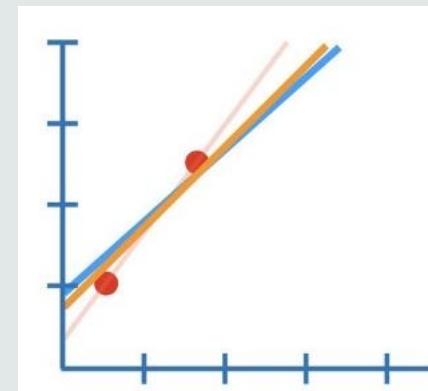
MODIFIKATION AV  
MULTIPLE  
REGRESSION  
MODEL SOM  
EXKLUDERAR  
EGENSKAPER SOM  
ÄR IRRELEVANTA  
FÖR MODELLEN



INTENTIONEN  
BAKOM



LASSO ESTIMATOR  
IN SCIKIT-LEARN



# Lasso: Least Absolute Shrinkage and Selection Operator

- Lasso är en multiple regression modell som i tillägg använder en metod som utför feature selection och regularization för att kunna förbättra noggrannheten för modellen

$$y_{pred} = w_0 + w_1x_1 + w_2x_2 + \cdots + w_px_p + \epsilon$$

Samma formel som i "Multiple Regression Model" men detta skiljer:

$$\sum_{i=1}^n (y_{pred_i} - y_i)^2 + \alpha \sum_{j=0}^p |w_j|$$

En "straff" tvingar vissa (irrelevanta) koefficienter att bli 0

# Lasso

- Väldigt användbar modell för "högre dimensionella" regressions problem. Alltså när vi har många variabler i datasetet
- Identifierar automatiskt de mest värdefulla egenskaperna (features)
- Bra prestanda när vi har få egenskaper som är viktiga och vi har många irrelevanta egenskaper

# Lasso Regression

- Lasso uppmaner till en simplare modell utan att ta bort features, men vikta ner påverkan
- Bra för modeller med hög multicollinearity
- Använder L1 regularisering som gör en automatisk feature selection
- L1 lägger till lett straff lika med absoluta värdet av koefficienternas  $w$  storlek

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 + \alpha \sum_{j=1}^p |w_j|$$

- Där  $y_i$  är observerad värde,  $\hat{y}_i$  är predikterat värde,  $n$  är antal observationer,  $\alpha$  är straff-parameter,  $w$  är vikterna till alla features och  $p$  är antal features i modellen
- $\alpha = 0$  betyder att alla features inkluderas och motsvarar linear regression
- $\alpha = \infty$  betyder att inga features inkluderas
- Bias ökar med större  $\alpha$
- Varians ökar med mindre  $\alpha$

# Lasso i scikit-learn



`sklearn.linear_model.Lasso(alpha=1.0)`



Alpha: konstant som multiplicerar L1-termen, sätts som standard till 1.0



Sätt högre värde om modellen ska välja egenskaper mer aggressivt (minne antal egenskaper)



Sätt mindre värde om modellen inte ska exkludera för många egenskaper

# Lasso i scikit-learn

$$\sum_{i=1}^n (y_{pred_i} - y_i)^2 + \underset{\text{Alpha}}{\alpha} \sum_{j=0}^p |w_j|$$

# Prediction Crime in US communitys

- Many variables are included so that algorithms that select or learn weights for attributes could be tested. However, clearly unrelated attributes were not included; attributes were picked if there was any plausible connection to crime (N=122), plus the attribute to be predicted (Per Capita Violent Crimes).
- Data is described below based on original values. All numeric data was normalized into the decimal range 0.00-1.00 using an unsupervised, equal-interval binning method. Attributes retain their distribution and skew (hence for example the population attribute has a mean value of 0.06 because most communities are small). E.g. An attribute described as 'mean people per household' is actually the normalized (0-1) version of that value.

# Prediction Crime in US communities

Hands on



Photo: Shutterstock/Photoshot

# Predicting Diamond Prices

1 **target** and 9 features

- Predicting **target** = price in US dollars
- carat: weight of the diamond (0.2~5.01)
- cut: quality of the cut (Fair, Good, Very Good, Premium, Ideal)
- color: diamond colour, from J (worst) to D (best)
- clarity: a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))
- x: length in mm (0~10.74)
- y: width in mm (0~58.9)
- z: depth in mm (0~31.8)
- depth: total depth percentage =  $z / \text{mean}(x, y) = 2 * z / (x + y)$  (43~79)
- table: width of top of diamond relative to widest point (43~95)



# Predicting Diamond Prices

Hands on

# Predicting Post Popularity

- Data Set Information:

This dataset summarizes a heterogeneous set of features about articles published by Mashable in a period of two years. The goal is to predict the number of shares in social networks (popularity).

- Features description

Number of Attributes: 61 (58 predictive attributes, 2 non-predictive, 1 target)

# Predicting Post Popularity

Hands on



# Improving Accuracy

- As this example shows, building accurate predictive models is not a straightforward task. There are many strategies to improve the accuracy of our models:
  - Collect more and/or better data
  - Feature engineering
  - Pre-processing: Deal with outliers, missing data and scale of features
  - Feature selection methods
  - Regularization
  - Parameter tuning
  - Use more complex models
- Not everything can be predicted accurately even with lots of data, highly complex models and best practices

# Övning

Testa runt med modellerna och parametraran och se hur det påverkar resultatet.

# Länkar

- KNN <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
- Overfitting - <https://blog.minitab.com/blog/understanding-statistics/how-to-avoid-overfitting-your-regression-model>
- Overfitting - <https://statisticsbyjim.com/regression/overfitting-regression-models/>
- Regularization - <https://towardsdatascience.com/regularization-an-important-concept-in-machine-learning-5891628907ea>
- Lasso regression <https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b>

# Vad har vi gjort idag?

- Regression i Python
- KNN
- Model evaluation för regression
  - Cross validation
  - Overfitting
  - Bias-variance tradeoff
  - Overfitting
  - Regularization
  - Feature selection
- Lasso regression
- Diamond prices
- Post popularity

# Nästa lektion

- Klassificering