DS(ML)B:
Data Science (& Machine Learning) for Business

Profs. Anton Ovchinnikov, Theos Evgeniou, Spyros Zoumpoulis

Sessions 07-08

- Wrap-up of **Supervised** Learning
- Discussion of Assignment 2
- Advanced Classification, cont.: Support Vector Machines, Regularizations (LASSO/Ridge)
- [Optional / Time-permitting]: Deep Learning

Big Picture:
Structure of the course

SESSIONS 1-2: Data analytics process; from Excel to R

- Tutorial 1: Getting comfortable with R

SESSIONS 3-4: Time Series Models

SESSIONS 5-6: Intro to Classification, Logistic Regression and CART Trees

- Tutorial 2: Mid-term help with R

SESSIONS 7-8: Wrap-up of Supervised Learning, Discuss Assignment 2, Advanced Classification

SESSIONS 9-10: Unsupervised Learning: Clustering, Segmentation and Dimensionality Reduction

- Tutorial 3: automatic machine learning with DataRobot
- Assignment 3 and Project Proposals
- Hands-on help with projects

SESSIONS 11-12: Guest speaker

SESSIONS 13-14: Project presentations

Plan for the day
Learning objectives

- Main objective: wrap-up/cristalize our understanding of supervised learning
- How:
  - Discussion of Assignement 2
    - Peer-to-peer group presentations + Q&A, each pair selects one [30mins]
    - Round of 4 peer-to-peer presentations, each 4 selects one [20mins]
    - « Winners » of each 4 present to us all + Q&A [30mins]
    - We all vote to select one group « to invest in »
    - I then reveal their $$$ made, we all discuss the results + Q&A
  - Advanced classification:
    - Support Vector Machines
    - Regularizations: LASSO, Ridge
    - [Optional / Time-permitting]: Deep Learning with TensorFlow
  - Summary: « from regression to deep learning »

Assignment 2: process

- Two tasks on hand:
  1. How to predict default probabilities
  2. How to decide which threshold to use?

- Task 1 is easy, e.g., here is an MVP:

```
credit_data_24000<-read.csv(file.choose(), header=TRUE)
new_applicants<-read.csv(file.choose(), header=TRUE)
str(credit_data_24000)
credit_data_24000$default_0<-as.factor(credit_data_24000$default_0)
ctree_tree<-ctree(default_0 ~ . -ID, data=credit_data_24000)
ctree_probabilities<-predict(ctree_tree, newdata=new_applicants)
write.csv(ctree_probabilities, file="pred_default_probs_new_applicants.csv")
```

- Task 2 is harder: how to know which threshold will bring the most money on the new applicants?
  - How to replicate giving credit on the data we have? →**Holdout!**

Assignment 2: process
Estimating threshold

- How to replicate giving credit on the data we have? →**Holdout!**

- Split the data into training and testing

```
training<-subset(credit_data_24000,ID<=23000)
testing<-subset(credit_data_24000,ID>23000)
```

- Train on `training`, predict on `testing`:

```
ctree_tree<-ctree(default_0~. -ID,data=training)

ctree_probabilities<-predict(ctree_tree,newdata=testing)
write.csv(ctree_probabilities, file="pred_default_probs_ctree_testing.csv")
```

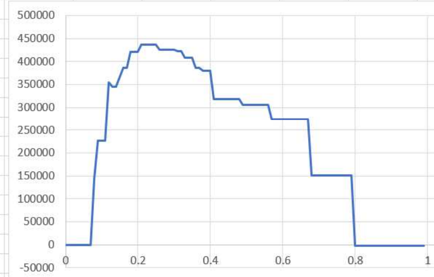
- Construct a profit curve, e.g., in Excel with a data table
  - In R this will require coding custom functions to loop and test all thresholds (beyond the scope of the course)

Assignment 2: process
Profit curve

- Construct a profit curve, e.g., in Excel with a data table
- Then repeat for different methods, models, data, etc.

N2 | {=TABLE(,G6)}

	A	B	D	F	G	I	J	K	M	N
1	ID	actual default_0	Predicted prob of default	principal	25000	Issue loan?	Revenue	Cost		437500
2	23001	0	0.081463583	interest	2%	1	1500	0	0	0
3	23002	0	0.078529278	recovery	20000	1	1500	0	0.01	0
4	23003	0	0.177814029	additional CLV	1000	1	1500	0	0.02	0
5	23004	0	0.15656963			1	1500	0	0.03	0
6	23005	0	0.081463583	offer threshold	0.21	1	1500	0	0.04	0
7	23006	0	0.177814029	revenue if repay	1500	1	1500	0	0.05	0
8	23007	0	0.081463583	loss if default	5000	1	1500	0	0.06	0
9	23008	0	0.078529278			1	1500	0	0.07	0
10	23009	1	0.676855895			0	0	0	0.08	145500
11	23010	0	0.089395267	Profit	437500	1	1500	0	0.09	226500
12	23011	0	0.089395267			1	1500	0	0.1	226500
13	23012	0	0.078529278	max profit	437500	1	1500	0	0.11	226500
14	23013	0	0.155172414	opt threshold	0.21	1	1500	0	0.12	355000
15	23014	1	0.144153226			1	0	5000	0.13	345000
16	23015	0	0.144153226			1	1500	0	0.14	345000
17	23016	0	0.078529278			1	1500	0	0.15	366500
18	23017	0	0.078529278			1	1500	0	0.16	386500
19	23018	0	0.081463583			1	1500	0	0.17	386500
20	23019	0	0.177814029			1	1500	0	0.18	420500
21	23020	1	0.676855895			0	0	0	0.19	420500
22	23021	0	0.081463583			1	1500	0	0.2	420500
23	23022	1	0.177814029			1	0	5000	0.21	437500
24	23023	0	0.081463583			1	1500	0	0.22	437500
25	23024	0	0.078529278			1	1500	0	0.23	437500
26	23025	0	0.177814029			1	1500	0	0.24	437500
27	23026	0	0.177814029			1	1500	0	0.25	437500
28	23027	0	0.155172414			1	1500	0	0.26	426500
29	23028	0	0.17791411			1	1500	0	0.27	426500
30	23029	0	0.144153226			1	1500	0	0.28	426500
31	23030	0	0.081463583			1	1500	0	0.29	426500
32	23031	0	0.089395267			1	1500	0	0.3	426500



Profit Curve:
"Implied" classification metric

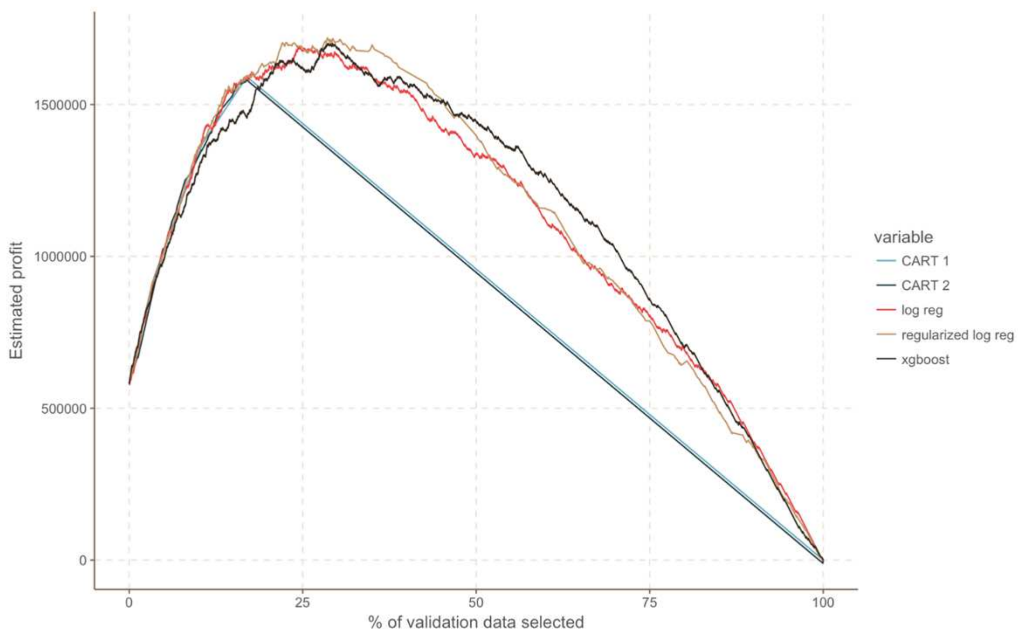
- Measure business profit if we only select the "top" cases in terms of the probability of "response"
- For this, we need to define values and costs of correct classifications and misclassifications

	Predicted: default	Predicted: no default
Actual: default	$0	-$5000
Actual: no default	$0	$1500

Profit = # of 1's correctly predicted * value of capturing a 1
+ # of 0's correctly predicted * value of capturing a 0
+ # of 1's incorrectly predicted as 0 * cost of missing a 1
+ # of 0's incorrectly predicted as 1 * cost of missing a 0

Profit Curve:
"Implied" classification metric

- Given a classifier, rank instances in the test data from highest predicted probability of belonging to class 1 (= default) to lowest
- Can put the cutoff for giving vs. not giving credit at any rank
- As you move the cutoff, calculate the corresponding profit… [here with 4000 in testing]



Moving beyond the MVP

Better models

on MVP data: [T*, π*]

- Logistic + `stepAIC`: 0.24, 423K
- `cart`: 0.21, 437.5K
- `randomforest`: 0.25, 510K
- “advanced blender” from DataRobot: 0.24, 581K

Better data

- Feature engineering
- Examples of “out of the box” impactful features:
  - NoOfDelay_Sum
  - NoOfDelay_Count
  - SumOfAll_PAYSTATUS
  - 2month_Delay_Count
  - SD_BILL
  - TotalPayment_6Months
  - TotalPayment_3Months
  - Single_and_male
  - Retiree_60

Feature Engineering

INSEAD

Key idea: Your data may have more information than what is contained in your existing variables

- Spend **lots of time** thinking of ways to combine your variables into new ones!
  - Or generally, where/how to get more information that would help your models learn better and make more accurate predictions
- "Engineering" good features may be more important than using a better method
- Requires contextual knowledge of the business
  - Can not be outsourced
  - Too many permutations to be fully automated

[**How hard**] have you thought about feature engineering?

S.No	Feature	Example	Definition	Formula
1	C_Feature_Noofdefaults_Count	2	Count of all pay status where pay status > 0	=IF(G2>0,1,0)+IF(H2>0,1,0)+IF(I2>0,1,0)+IF(J2>0,1,0)+IF(K2>0,1,0)+IF(L2>0,1,0)
2	Feature_Noofdefaults_Sum	4	Sum of all pay status where pay status > 0	=IF(G2>0,G2,0)+IF(H2>0,H2,0)+IF(I2>0,I2,0)+IF(J2>0,J2,0)+IF(K2>0,K2,0)+IF(L2>0,L2,0)
3	C_Feature_Improve_Pay1	No Change	If Pay Status in Pay X improved from PayX-1	=IF(G2=H2,"No Change",IF(G2<H2,"Improved","Decreased"))
4	C_Feature_Improve_Pay2	Decreased	If Pay Status in Pay X improved from PayX-1	=IF(H2=I2,"No Change",IF(H2<I2,"Improved","Decreased"))
5	C_Feature_Improve_Pay3	No Change	If Pay Status in Pay X improved from PayX-1	=IF(I2=J2,"No Change",IF(I2<J2,"Improved","Decreased"))
6	C_Feature_Improve_Pay4	Decreased	If Pay Status in Pay X improved from PayX-1	=IF(J2=K2,"No Change",IF(J2<K2,"Improved","Decreased"))
7	C_Feature_Improve_Pay5	No Change	If Pay Status in Pay X improved from PayX-1	=IF(K2=L2,"No Change",IF(K2<L2,"Improved","Decreased"))
8	C_Feature_Paystatus_Notchanged_6	FALSE	If Pay Status in Pay1 - Pay6 remains same	=SUM(G2:L2)/6=G2
9	C_Feature_Paystatus_Notchanged_3	FALSE	IF Pay Status in Pay1 - Pay3 remains same	=SUM(G2:I2)/3=G2
10	Feature_Limit_Log	4.301029996	Log of Limit Balance	=LOG(B2)
11	Feature_Limit_TANH	0.999632614	TANH of log of limit balance	=TANH(AI2)
12	Feature_Standarized_LimitBal	-1.138344866	Standarized Z value of limit balance	=STANDARDIZE(B2,AVERAGE($B$2:$B$24001),STDEV.P($B$2:$B$24001))
13	C_Feature_Sex_Marriage	Female AND Married	Interaction of Sex and Marital Status	=IF(C2=1,"Male",IF(C2=2,"Female","Other"))&" AND
14	C_Feature_Retired60	Not Retired	Capture Retirement assuming age 60	=IF(F2>60,"Retired","Not Retired")
15	C_Feature_Retired65	Not Retired	Capture Retirement assuming age 65	=IF(F2>65,"Retired","Not Retired")
16	C_Feature_Education_Missingflag	0	Surrogate variable for Education missing values	=IF(D2=0,1,0)
17	C_Feature_Marriage_Missingflag	0	Surrogate variable for Marital status missing value	=IF(E2=0,1,0)
18	C_Feature_Fullpayment_PayM1	Not Fully Paid	If Pay amount in month 1 is fully paid or not	=IF((N2-S2)<=0,"Fully Paid","Not Fully Paid")
19	C_Feature_Fullpayment_PayM2	Fully Paid	If Pay amount in month 2 is fully paid or not	=IF((O2-T2)<=0,"Fully Paid","Not Fully Paid")
20	C_Feature_Fullpayment_PayM3	Fully Paid	If Pay amount in month 3 is fully paid or not	=IF((P2-U2)<=0,"Fully Paid","Not Fully Paid")
21	C_Feature_Fullpayment_PayM4	Fully Paid	If Pay amount in month 4 is fully paid or not	=IF((Q2-V2)<=0,"Fully Paid","Not Fully Paid")
22	C_Feature_Fullpayment_PayM5	Fully Paid	If Pay amount in month 5 is fully paid or not	=IF((R2-W2)<=0,"Fully Paid","Not Fully Paid")
23	C_Feature_NospendMonths	3	Count of months where bill amount = 0	=IF(M2=0,1,0)+IF(N2=0,1,0)+IF(O2=0,1,0)+IF(P2=0,1,0)+IF(Q2=0,1,0)+IF(R2=0,1,0)
24	C_Feature_NoPaymentMonths	5	Count of months where payment amount = 0	=IF(S2=0,1,0)+IF(T2=0,1,0)+IF(U2=0,1,0)+IF(V2=0,1,0)+IF(W2=0,1,0)+IF(X2=0,1,0)
25	Feature_Ratio_NPM_NSM	1.666666667	Ratio of No payment months / No spend month	=IFERROR(AW2/AV2,0)
26	Feature_Payment_Percent_1	0	Payment 1 month ago / bill 2 month ago	=IFERROR(S2/N2,0)
27	Feature_Payment_Percent_2	1	Payment 2 month ago / bill 3 month ago	=IFERROR(T2/O2,0)
28	Feature_Payment_Percent_3	0	Payment 3 month ago / bill 4 month ago	=IFERROR(U2/P2,0)
29	Feature_Payment_Percent_4	0	Payment 4 month ago / bill 3 month ago	=IFERROR(V2/Q2,0)
30	Feature_Payment_Percent_5	0	Payment 5 month ago / bill 4 month ago	=IFERROR(W2/R2,0)
31	Feature_Average_Percent	0.2	Average of all Payment Ratio (M1 - M5)	=AVERAGE(AY2:BC2)
32	C_Feature_Bill_Adjustment1	0	Flag if the bill amount < 0 to identify adjustments	=IF(M2<0,1,0)
33	C_Feature_Bill_Adjustment2	0	Flag if the bill amount < 0 to identify adjustments	=IF(N2<0,1,0)
34	C_Feature_Bill_Adjustment3	0	Flag if the bill amount < 0 to identify adjustments	=IF(O2<0,1,0)
35	C_Feature_Bill_Adjustment4	0	Flag if the bill amount < 0 to identify adjustments	=IF(P2<0,1,0)
36	C_Feature_Bill_Adjustment5	0	Flag if the bill amount < 0 to identify adjustments	=IF(Q2<0,1,0)
37	C_Feature_Bill_Adjustment6	0	Flag if the bill amount < 0 to identify adjustments	=IF(R2<0,1,0)
38	C_Feature_AllMonth_minus2	FALSE	Flag if all the pay status is -2	=SUM(G2:L2)=-12
39	C_Feature_AllMonth_minus1	FALSE	Flag if all the pay status is -1	=SUM(G2:L2)=-6
40	C_Feature_AllMonth_zero	FALSE	Flag if all the pay status is 0	=SUM(G2:L2)=0
41	Feature_sumofall_paystatus	-2	Sum of all pay status	=SUM(G2:L2)
42	Feature_Averageborrowing	1284	Average spending per month	=AVERAGE(M2:R2)
43	Feature_exposure	0.0642	Average spendng / total limit to check exposure	=BO2/B2
44	C_Feature_Agebucket	20-25	Continious variable age divided into buckets of 5	=VLOOKUP(F2,Age!A:B,2,0)
45	Feature_totaloutstanding	7015	Total Spending - Total Payments	=SUM(M2:R2)-SUM(S2:X2)
46	C_Feature_outstanding_neg_flag	0	Flag to see if total payments > total spending	=IF(BR2<0,1,0)
47	C_Feature_1month_delay_count	0	Count of times payments were delayed by 1 month	=IF($G2=1,1,0)+IF($H2=1,1,0)+IF($I2=1,1,0)+IF($J2=1,1,0)+IF($K2=1,1,0)+IF($L2=1,1,0)
48	C_Feature_2month_delay_count	2	Count of times payments were delayed by 2 month	=IF($G2=2,1,0)+IF($H2=2,1,0)+IF($I2=2,1,0)+IF($J2=2,1,0)+IF($K2=2,1,0)+IF($L2=2,1,0)
49	C_Feature_3month_delay_count	0	Count of times payments were delayed by 3 month	=IF($G2=3,1,0)+IF($H2=3,1,0)+IF($I2=3,1,0)+IF($J2=3,1,0)+IF($K2=3,1,0)+IF($L2=3,1,0)
50	C_Feature_4month_delay_count	0	Count of times payments were delayed by 4 month	=IF($G2=4,1,0)+IF($H2=4,1,0)+IF($I2=4,1,0)+IF($J2=4,1,0)+IF($K2=4,1,0)+IF($L2=4,1,0)
51	C_Feature_5month_delay_count	0	Count of times payments were delayed by 5 month	=IF($G2=5,1,0)+IF($H2=5,1,0)+IF($I2=5,1,0)+IF($J2=5,1,0)+IF($K2=5,1,0)+IF($L2=5,1,0)
52	C_Feature_6month_delay_count	0	Count of times payments were delayed by 6 month	=IF($G2=6,1,0)+IF($H2=6,1,0)+IF($I2=6,1,0)+IF($J2=6,1,0)+IF($K2=6,1,0)+IF($L2=6,1,0)
53	C_Feature_6monthplus_delay_count	0	Count of times payments were delayed by >6 mon	=IF($G2>6,1,0)+IF($H2>6,1,0)+IF($I2>6,1,0)+IF($J2>6,1,0)+IF($K2>6,1,0)+IF($L2>6,1,0)
54	C_Feature_AbvAvg_Month1	TRUE	Flag to see if spend in Month 1 > Average	=M2>$BO2
55	C_Feature_AbvAvg_Month2	TRUE	Flag to see if spend in Month 2 > Average	=N2>$BO2
56	C_Feature_AbvAvg_Month3	FALSE	Flag to see if spend in Month 3 > Average	=O2>$BO2
57	C_Feature_AbvAvg_Month4	FALSE	Flag to see if spend in Month 4 > Average	=P2>$BO2
58	C_Feature_AbvAvg_Month5	FALSE	Flag to see if spend in Month 5 > Average	=Q2>$BO2
59	C_Feature_AbvAvg_Month6	FALSE	Flag to see if spend in Month 6 > Average	=R2>$BO2
60	C_Feature_PayStatus_Category	More than once	How many times payment were delayed in last 6	=IF(IF(G2>0,1,0)+IF(H2>0,1,0)+IF(I2>0,1,0)+IF(J2>0,1,0)+IF(K2>0,1,0)+IF(L2>0,1,0)=1,"Rare
61	C_Feature_old_but_not_married	0	Flag to check if client is over 40 and not yet marrie	=IF(AND(F2>40,E2>=2),1,0)
62	Feature_TotalPayment_6Months	689	Total Payment in last 6 months	=SUM(S2:X2)
63	Feature_TotBill_TotPymt	7015	Total Bill - Total Payments in last 6 months	=SUM(M2:R2)-SUM(S2:W2)

S.No	Feature	Example	Definition	Formula
80	Feature_overpayment_Amt_5	0	Amount of overpayment	=CU2*(W2-R2)
81	Feature_total_overpymt_Amt	0	Sum of all overpayments	=SUM(CW2:DA2)
82	C_Feature_no_overpymt	1	Flag to check if no overpa	IF(DB2=0,1,0)
83	C_Feature_Range_PayMonth	4	Range of pay status over	=MAX(G2:L2)-MIN(G2:L2)
84	C_Feature_minus2_count	2	Count of how many	=IF(G2=-2,1,0)+IF(H2=-2,1,0)+IF(I2=-2,1,0)+IF(J2=-2,1,0)+IF(K2=-2,1,0)+IF(L2=-2,1,0)
85	Feature_Exposure_Month1	0.19565	Bill amount 1 / total limi	=M2/B2
86	C_Feature_Limit_Bin	1	Continious variable	=IF(B2<100000,1,IF(B2<200000,2,IF(B2<300000,3,IF(B2<400000,4,IF(B2<500000,5,6)))))
87	C_Feature_Diminishing	0	Flag to see if the	=IF((G2-L2)=5,1,0)
88	C_Feature_Abnormal_Tx1	1	Flag to check if the	=IF(M2>AVERAGE($M$2:$R$2)*2,1,0)
89	C_Feature_Abnormal_Tx2	1	Flag to check if the	=IF(N2>AVERAGE($M$2:$R$2)*2,1,0)
90	C_Feature_Abnormal_Tx3	0	Flag to check if the	=IF(O2>AVERAGE($M$2:$R$2)*2,1,0)
91	C_Feature_Abnormal_Tx4	0	Flag to check if the	=IF(P2>AVERAGE($M$2:$R$2)*2,1,0)
92	C_Feature_Abnormal_Tx5	0	Flag to check if the	=IF(Q2>AVERAGE($M$2:$R$2)*2,1,0)
93	C_Feature_Abnormal_Tx6	0		=IF(R2>AVERAGE($M$2:$R$2)*2,1,0)
94	C_Feature_all_zeros	0	Count of pay status = 0 ir	=IF(G2=0,1,0)+IF(H2=0,1,0)+IF(I2=0,1,0)+IF(J2=0,1,0)+IF(K2=0,1,0)+IF(L2=0,1,0)
95	Feature_status_growth	-1	% increase in Pay Status	=IFERROR(G2/L2,0)
96	Feature_SD_growth	1.699673171	Standard Deviation of pa	=STDEV.P(G2:L2)
97	C_Feature_Pay_last2_status	2 AND 2	Pay Status of M1 and M2	=G2&" AND "&H2
98	Feature_SD_bill	1608.143754	Standard Deviation of bil	=STDEV.P(M2:R2)
99	Feature_SD_payment	256.7751394	Standard Deviation of pa	=STDEV.P(S2:X2)
100	Feature_SDB_div_SDP	6.262848333	Ratio of SD of bill over SI	=IFERROR(DS2/DT2,0)
101	Feature_Pay1	0	% of payment in M1 / Bil	=IFERROR(S2/N2,0)
102	Feature_Pay2	1	% of payment in M2 / Bil	=IFERROR(T2/O2,0)
103	Feature_Pay3	0	% of payment in M3 / Bil	=IFERROR(U2/P2,0)
104	Feature_Pay4	0	% of payment in M4 / Bil	=IFERROR(V2/Q2,0)
105	Feature_Pay5	0	% of payment in M5 / Bil	=IFERROR(W2/R2,0)
106	Feature_Avg_Pay1_to_5	0.2	Average % of payment	=AVERAGE(DV2:DZ2)
107	C_Feature_Limit_Method2	1-1	Alternative method to bir	=LEFT(A2,2)&"-"&LEN(A2)
108	Feature_Correl	-3014.188114	Correlation with limit bal	=-0.150709405692922*B2
109	Feature_Kurtosis_Spending	-1.382080958	Kurtosis of all bill amour	=IFERROR(KURT(M2,N2,O2,P2,Q2,R2),0)
110	Feature_Kurtosis_Payment	6	Kurtosis of all payment a	=IFERROR(KURT(S2,T2,U2,V2,W2,X2),0)
111	Feature_Kurtosis_Pay_mult_Spend	-8.29248575	Product of Kurtosis of bil	=EE2*ED2
112	C_Feature_Mode_PayStatus	2	Mode of pay status	=IFERROR(MODE.SNGL(G2:L2),0)
113	C_Feature_Sex_Education_Marriage	2 AND 2 AND 1	Interaction of Sex, Educa	=C2&" AND "&D2&" AND "&E2
114	C_Feature_Sex_Education_Marriage	2 AND 2 AND 1 AND 20-	Interaction of Sex, Educa	=C2&" AND "&D2&" AND "&E2&" AND "&BQ2
115	C_Feature_Sex_Education_Marriage	2 AND 2 AND 1 AND 20-	Interaction of Sex,	=C2&" AND "&D2&" AND "&E2&" AND "&BQ2&" AND "&DG2
116	C_Feature_Sex_Habit	2 AND More than once	Interaction of Sex and Ha	=C2&" AND "&CG2
117	C_Feature_Sex_Marriage_Habit	Female AND Married AN	Interaction of Sex, Marita	=AL2&" AND "&EK2
118	C_Feature_PayStatus_PayScore_Mor	Under AND 2	Interaction of Pay Status	=IF(S2=N2,"Equal Payment",IF(S2<N2,"Under","Over"))&" AND " &G2
119	C_Feature_PayStatus_PayScore_Mor	Equal Payment AND 2	Interaction of Pay Status	=IF(T2=O2,"Equal Payment",IF(T2<O2,"Under","Over"))&" AND " &H2
120	C_Feature_PayStatus_PayScore_Mor	Equal Payment AND -1	Interaction of Pay Status	=IF(U2=P2,"Equal Payment",IF(U2<P2,"Under","Over"))&" AND " &I2
121	C_Feature_PayStatus_PayScore_Mor	Equal Payment AND -1	Interaction of Pay Status	=IF(V2=Q2,"Equal Payment",IF(V2<Q2,"Under","Over"))&" AND " &J2
122	C_Feature_PayStatus_PayScore_Mor	Equal Payment AND -2	Interaction of Pay Status	=IF(W2=R2,"Equal Payment",IF(W2<R2,"Under","Over"))&" AND " &K2
123	Feature_NospendMonths_Num	3	Numeric Value of C_Feat	=AV2
124	Feature_NoPaymentMonths_Num	5	Numeric Value of C_Feat	=AW2
125	Feature_1month_delay_count_Num	0	Numeric Value of C_Feat	=BT2
126	Feature_2month_delay_count_Num	2	Numeric Value of C_Feat	=BU2
127	Feature_3month_delay_count_Num	0	Numeric Value of C_Feat	=BV2
128	Feature_4month_delay_count_Num	0	Numeric Value of C_Feat	=BW2
129	Feature_5month_delay_count_Num	0	Numeric Value of C_Feat	=BX2
130	Feature_6month_delay_count_Num	0	Numeric Value of C_Feat	=BY2
131	Feature_6monthplus_delay_count_N	0	Numeric Value of C_Feat	=BZ2
132	Feature_allM_overpayment_Num	0	Numeric Value of C_Feat	=CV2
133	Feature_Range_PayMonth_Num	4	Numeric Value of C_Feat	=DD2
134	Feature_Mode_PayStatus_Num	2	Numeric Value of C_Feat	=EG2
135	C_Feature_Increasing_Delays	0	Flag if increasing delay tr	=IF(AND(G2-H2=1,H2-I2=1,I2-J2=1,J2-K2=1,K2-L2=1),1,0)
136	C_Feature_Allzero_with_minus12	0	Flag if sum of bills and	=IF(AND(SUM(M2:X2)=0,SUM(G2:L2)>=-12),1,0)
137	C_Feature_Allzero_with_minus18	0	Flag if sum of bills and	=IF(AND(SUM(M2:X2)=0,SUM(G2:L2)>=-18),1,0)
138	C_Feature_PAY1_Weighted	20	Pay status weigher (on sc	=G2*10
139	C_Feature_PAY2_Weighted	18	Pay status weigher (on sc	=H2*9
140	C_Feature_PAY3_Weighted	-8	Pay status weigher (on sc	=I2*8
141	C_Feature_PAY4_Weighted	-7	Pay status weigher (on sc	=J2*7
142	C_Feature_PAY5_Weighted	-12	Pay status weigher (on sc	=K2*6

Profit = 663K

Additional/advanced classification methods

- CART-based tree ensembles
  - `randomforest`
  - gradient boosting machines (`xgboost`)
- "inbetween" regressions and trees:
  - support vector machines (`svm`)
- Regularizations (LASSO/Ridge)
  - SVM is also a regularization
- [Optional / Time-permitting] "Deep Learning" -- Artificial Neural Networks
  - ANN (Deep Learning) is also a regularization

Tree Ensemble Methods

Both **random forests** and **boosted trees** generate multiple random samples from the training set (with replacement), and train a different CART for each sample of the data. This is called "bagging."

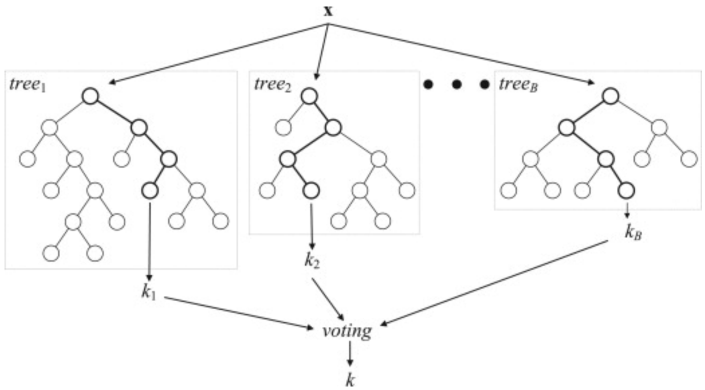
- Random Forests
  - The samples are completely random. No adaptiveness.
  - Use fully grown CARTs (each with low bias, high variance). Reduce variance by bagging together many uncorrelated trees.
  - Final prediction is the simple average

- Boosted trees
  - Based on small trees: weak learners with high bias, low variance
  - But adaptive: instances modeled poorly by the overall system before, have larger probability of being picked now → higher weight
  - Final prediction is a weighted average

Random Forest

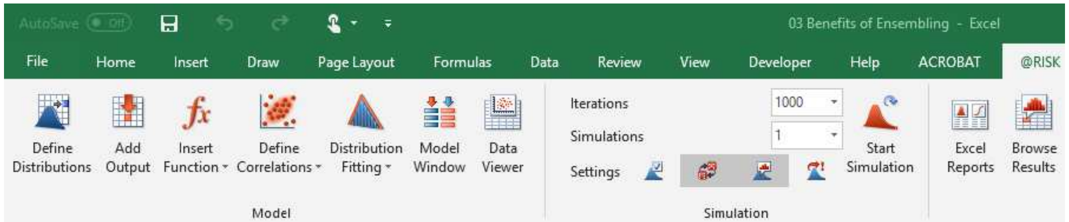
Main idea: Fit many trees to different samples of data, then ensemble them

- In addition to simply making a prediction, random forest provides an important insight into which variables show up in many trees - "important variables"



- Why does this work? “Wisdom of Crowds”
- Analogy to the decision making by a committee: Parliament vs Dictator, Board of a company vs sole proprietor, etc.
- Requires specifying hyper-parameters: number of trees and columns, depth of trees, voting rules, etc.
- Hyper-parameters are tuned via grid search

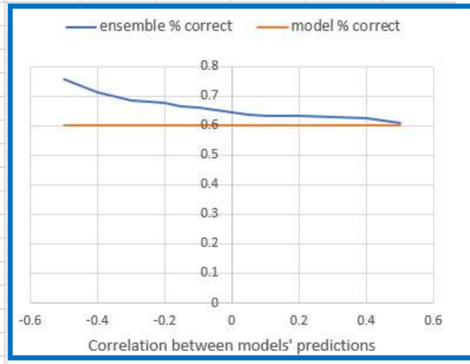
Illustration of Ensembling: @Risk "Monte Carlo" simulation



3 models;
each correct
with this prob →

	A	B	C	D	E	F	G	H
1								
2	model % correct		0.6		Correlation	0.5		
3								
4		model 1	1		@RISK Correlations	model 1 in $C	model 2 in $C	model 3 in $C$6
5		model 2	0		model 1 in $C$4	1		
6		model 3	1		model 2 in $C$5	0.5	1	
7					model 3 in $C$6	0.5	0.5	1
8		majority vote	1					
9								
10		Correlation	1	ensemble % correct	model % correct			
11		-0.5	1	0.758	0.6			
12		-0.4	1	0.713	0.6			
13		-0.3	0	0.685	0.6			
14		-0.2	1	0.676	0.6			
15		-0.15	0	0.665	0.6			
16		-0.1	1	0.662	0.6			
17		-0.05	1	0.654	0.6			
18		0	1	0.646	0.6			
19		0.05	0	0.637	0.6			
20		0.1	1	0.634	0.6			
21		0.15	1	0.633	0.6			
22		0.2	1	0.633	0.6			
23		0.3	1	0.629	0.6			
24		0.4	0	0.623	0.6			
25		0.5	1	0.609	0.6			

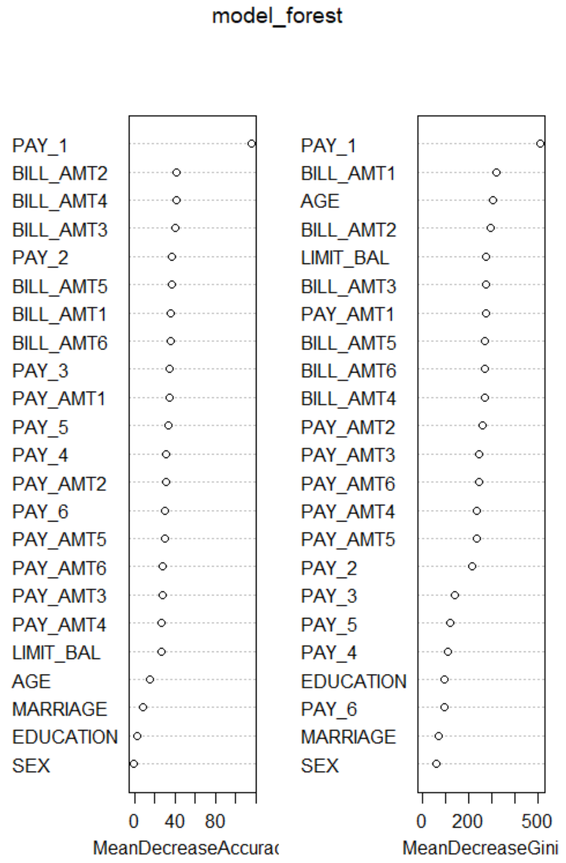
and predictions are correlated
← with this correlation coefficient/matrix



Majority vote ensemble is **ALWAYS** more accurate; more so when models have negatively correlated predictions

A2 implementation: randomforest

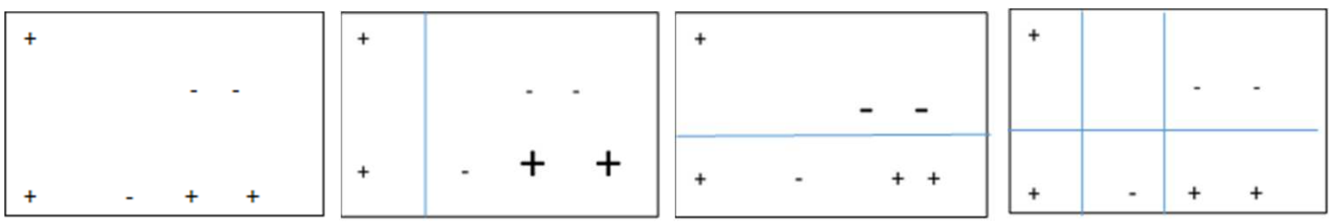
```
model_forest <- randomForest(default_0 ~ . -ID,
data=training,importance=TRUE, proximity=TRUE,
cutoff = c(0.5,0.5), type="classification")

varImpPlot(model_forest)


forest_probabilities<-predict(model_forest,
newdata=testing,type="prob")[,2]

write.csv(forest_probabilities,
file="predicted_default_probs_forest_testing.csv")


forest_probabilities<-predict(model_forest,
newdata=new_applicants,type="prob")[,2]

write.csv(forest_probabilities,
file="predicted_default_probs_forest_new_applicants
.csv")
```



Gradient Boosted Trees:
`xgboost`

Main idea: Notice which data points are not explained well by the existing tree, make those data points more important ("higher weight") and re-fit to describe them better

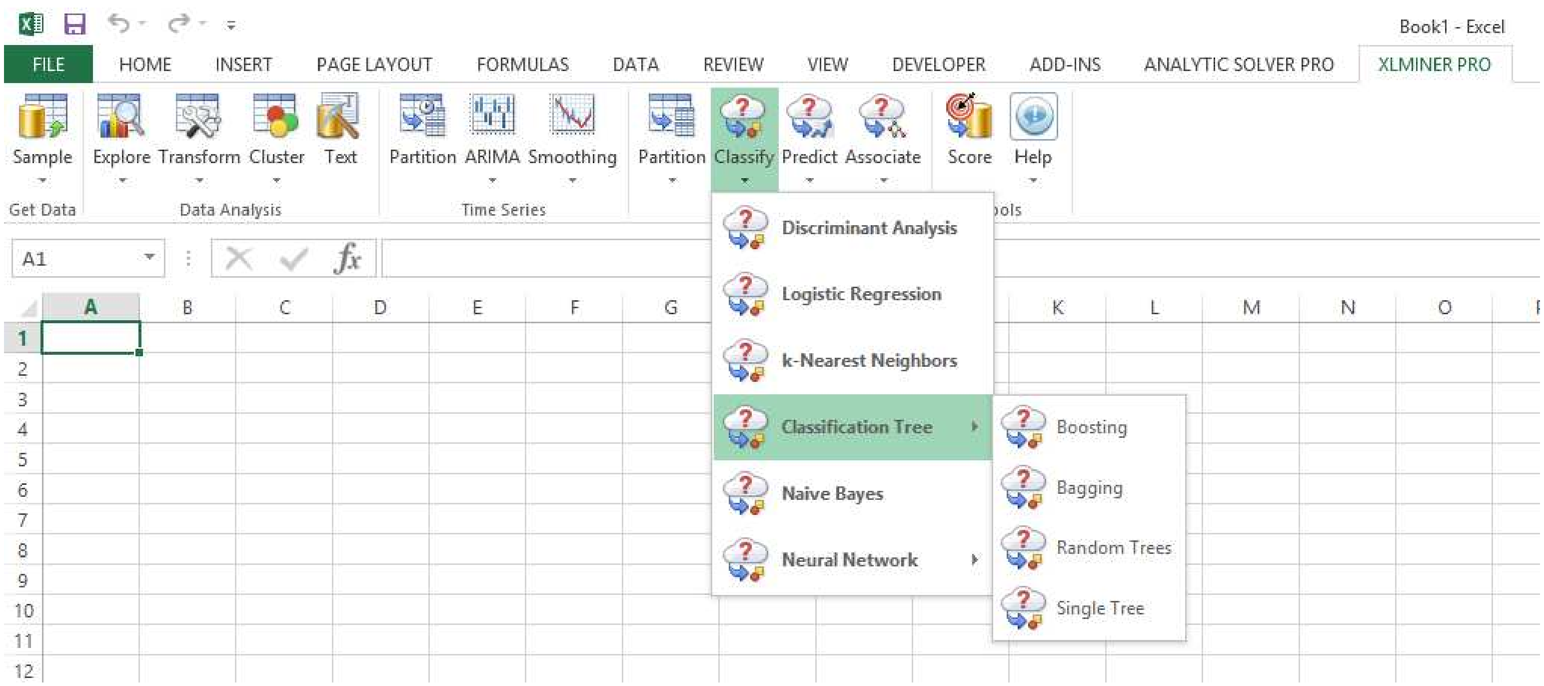
- Combine variables and add new splits to explain those higher weight data points



- Requires specifying additional hyper-parameters: number of trees, depth and learning/decay rate
- Hyper-parameters are tuned via grid search

CART-like Methods in Excel

Part of XLMiner Pro by Frontline Systems (same company that makes Solver)

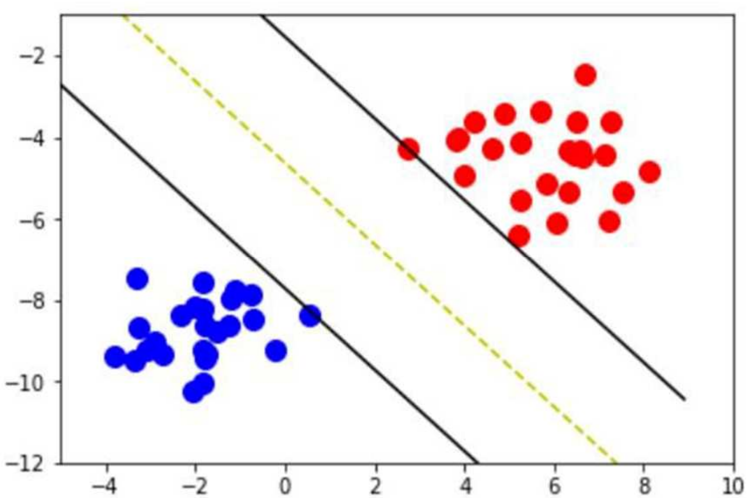


"In-between" Regressions and Trees: Support Vector Machines

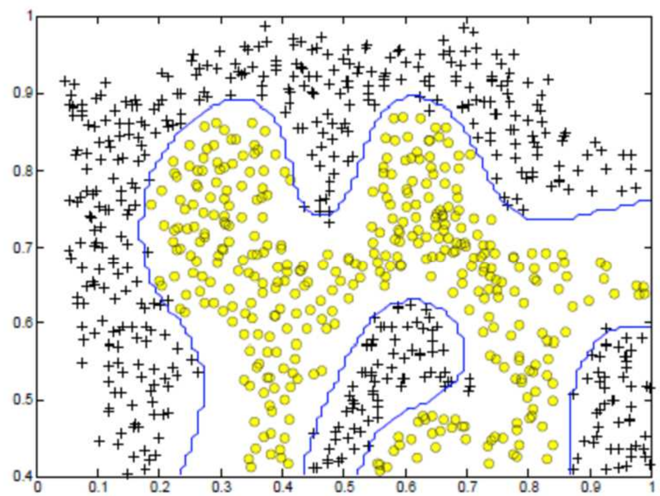
INSEAD

Main idea: Draw a line ("hyperplane", "kernel") dividing parameter space in two regions so that the margin on the sides is maximal

- Observe: CART-like methods draw vertical or horizontal lines only
- Why does this work? "Complicated" lines may work better than just vertical or horizontal ones



Linear kernel



Radial basis (Gaussian) kernel

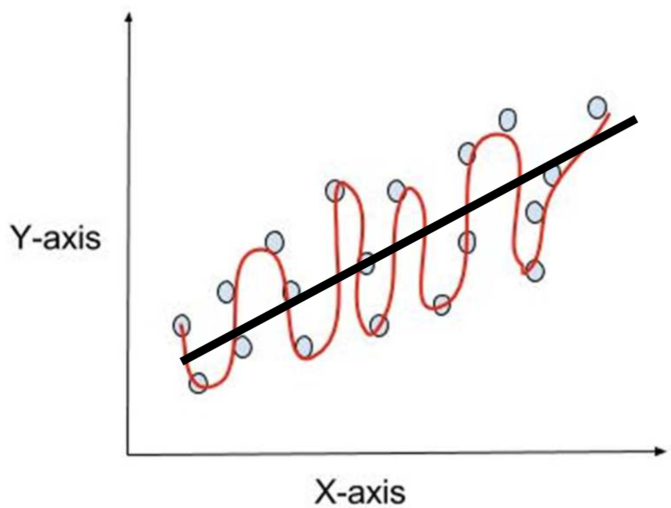
A2 implementation : svm

```
pacman::p_load("caret","ROCR","lift","glmnet","MASS","e1071")

model_svm <- svm(default_0 ~ . -ID, data=training, probability=TRUE)
summary(model_svm)

svm_probabilities<-attr(predict(model_svm, newdata=testing,
probability=TRUE), "prob")[,1]
write.csv(svm_probabilities, file="predicted_default_probs_SVM_testing.csv")

svm_probabilities<-attr(predict(model_svm,newdata=new_applicants,
probability=TRUE), "prob")[,1]
write.csv(svm_probabilities,
file="predicted_default_probs_SVM_new_applicants.csv")
```

Regularizations

[Idea Before] **Main Idea**:

- How can we further improve prediction accuracy? [Nonlinear] "Feature Engineering"



$$Y \ = a + \boldsymbol{b} * \boldsymbol{X} \ + \ error$$

versus

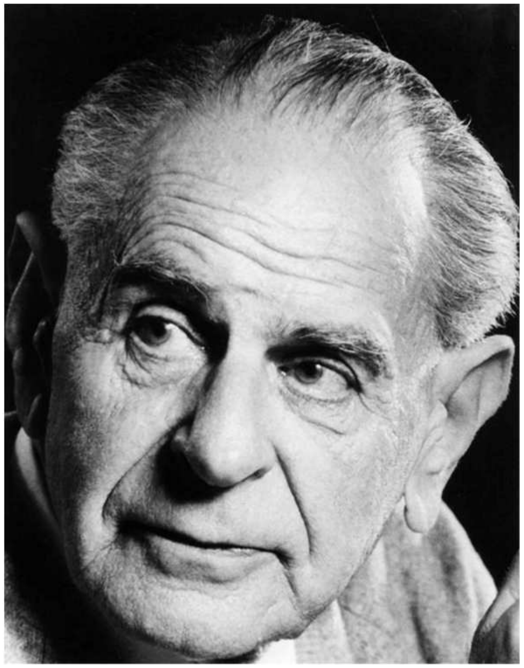
$$Y \ = a + \sum_{k=1}^{\infty} b_k * \varphi_k(x) + error$$

For example: $\varphi_k(x) = x^k$

$\varphi_k(x) = \cos(kx)$

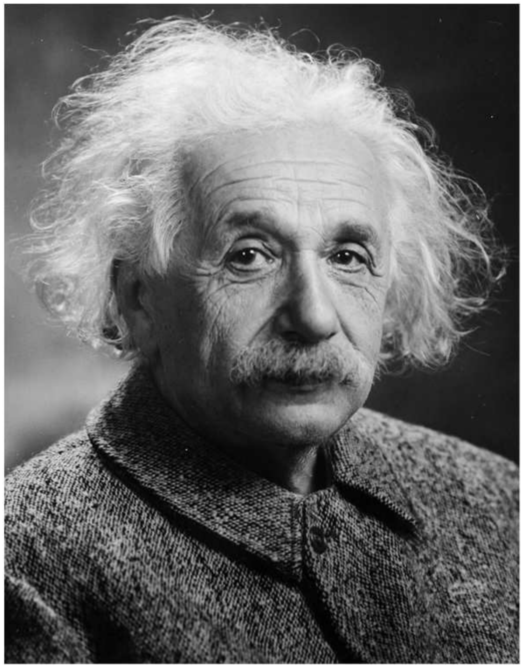
$\varphi_k(x) = IF[M\&single, \ldots$

Regularization:
Feature Engineering & Overfitting



Karl Popper

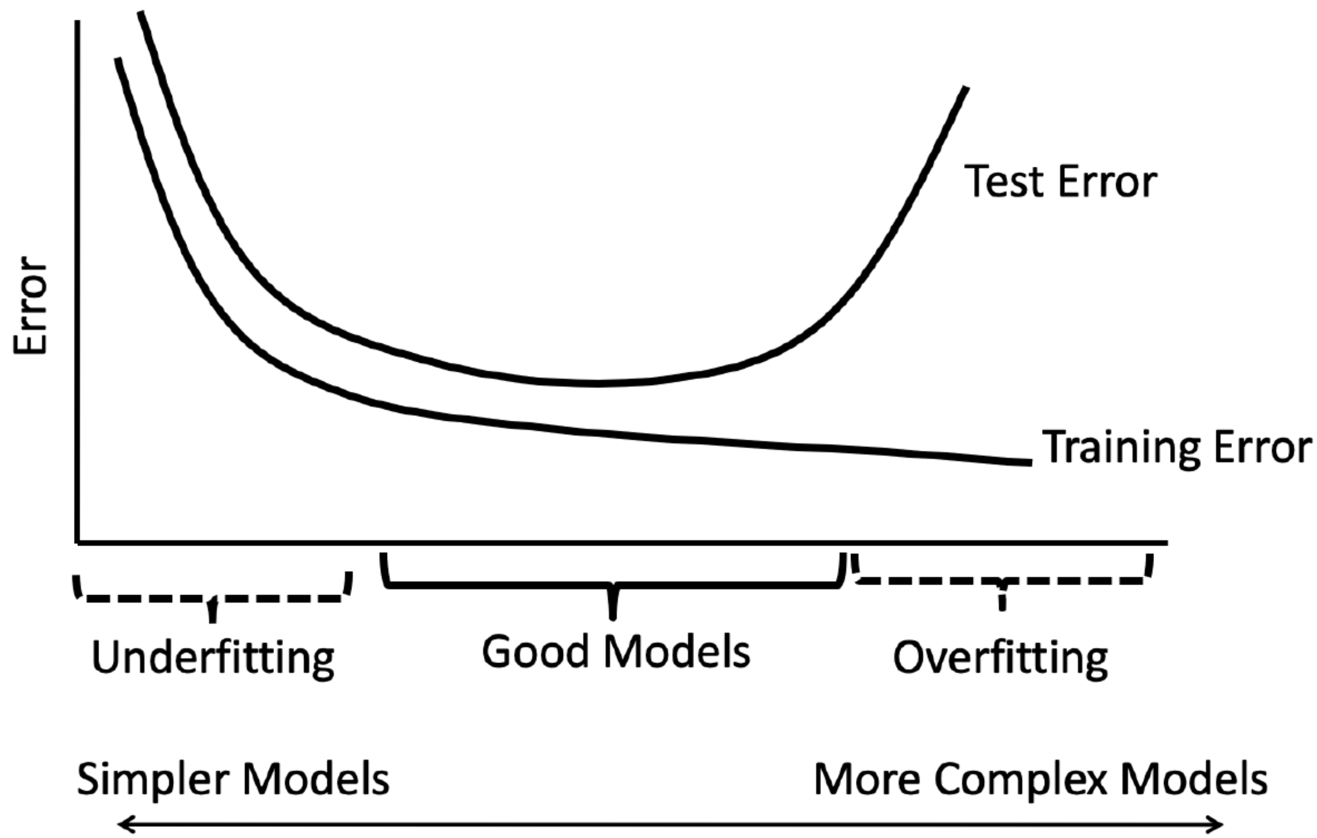
Theory of Knowledge: **Falsifiability**
"*All swans are white*"



Albert Einstein

Theory of Knowledge: **Complexity**
"*Everything Should Be Made as Simple as Possible, But Not Simpler*", (KISS) $E=mc^2$

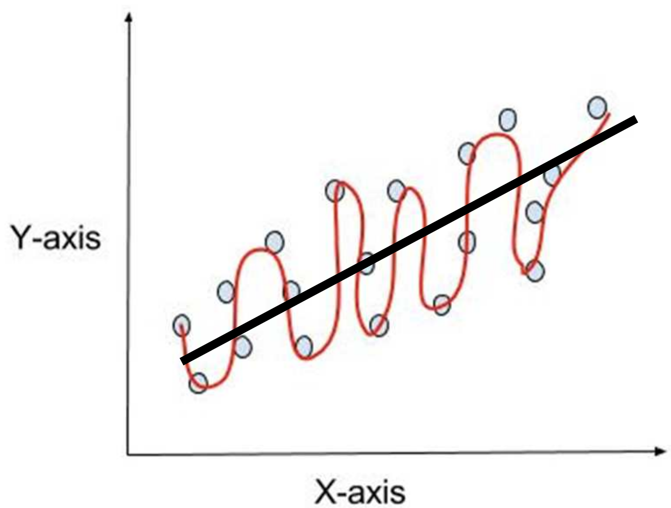
Complexity: Underfitting and Overfitting



Regularizations

[Idea Before] **Main Idea**:

- How can we further improve prediction accuracy? [Nonlinear] "Feature Engineering“



$$Y \ = a + \boldsymbol{b} * \boldsymbol{X} \ + \ error$$

versus

$$Y \ = a + \sum_{k=1}^{\infty} b_k * \varphi_k(x) + error$$

For example: $\varphi_k(x) = x^k$

$\varphi_k(x) = \cos(kx)$

$\varphi_k(x) = IF[M\&single, \ldots$

Main Idea: …but penalize for having too much complexity (too many variables) to avoid overfitting. Add penalty parameter, $\boldsymbol{\lambda}$, into the "regression" objective to control complexity

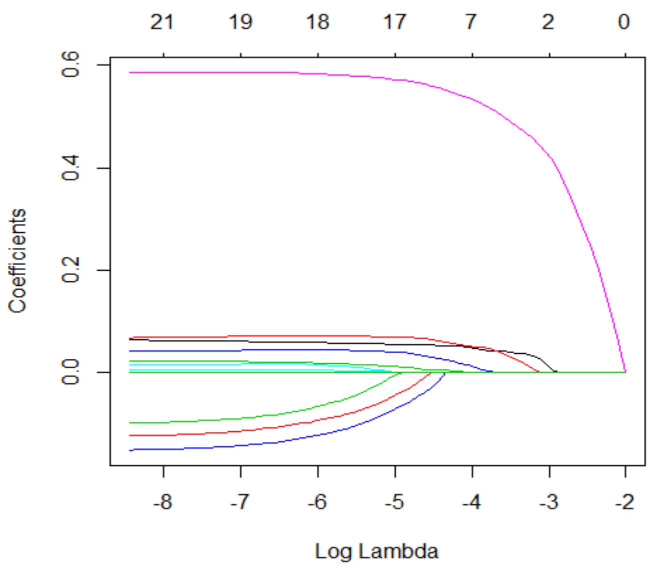
Examples of Complexity Control

“norm 1” of coefficients vector

LASSO regression, $y \equiv f(x_i) = a + bX$	$\min_f \sum_{i=1}^{m} (f(x_i) - y_i)^2 + \lambda \lVert \boldsymbol{b} \rVert_1$	Penalizes the absolute values of the coefficients: Most coefficients will be zero, only the most important coefficients will be non-zero ("concentrating the weights")
Linear Ridge regression, $y \equiv f(x_i) = a + bX$	$\min_f \sum_{i=1}^{m} (f(x_i) - y_i)^2 + \lambda \lVert \boldsymbol{b} \rVert_2^2$	Penalizes sum of squares of coefficients: will end-up having many small coefficients ("redistributing the weights")
Support Vector Machines $y \equiv f(x_i) = a + bX$	$\min_f \sum_{i=1}^{m} \lvert f(x_i) - y_i \rvert_e + \lambda \lVert \boldsymbol{b} \rVert_2^2$	
Generalized Ridge regression, $y \equiv f(x_i) = \sum_{k=1}^{\infty} b_k * \varphi_k(x)$	$\min_f \sum_{i=1}^{m} (f(x_i) - y_i)^2 + \lambda \lVert \boldsymbol{b} \rVert_K^2$	Most flexible, but hardest to interpret coefficients

Understanding LASSO/Ridge package `glmnet`

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \frac{1}{2N}\sum_{i=1}^{N}(y_i-\beta_0-x_i^T\beta)^2+\lambda\left[(1-\alpha)||\beta||_2^2/2+\alpha||\beta||_1\right]$$



βs are regression coefficients

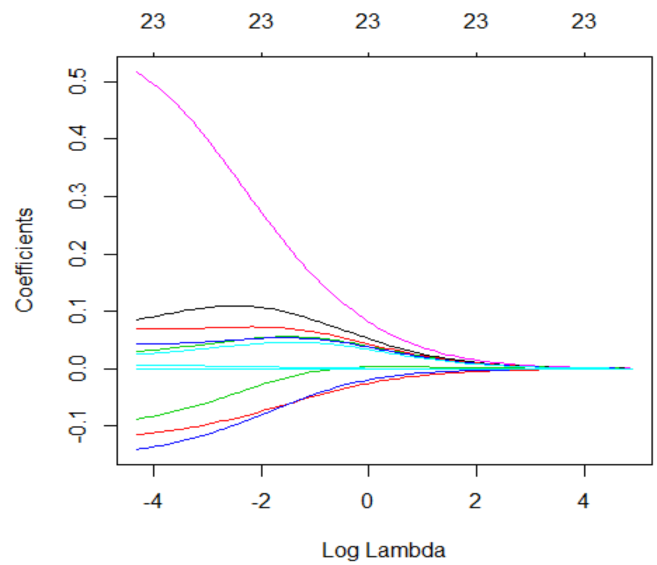
λ is the penalty parameter

$\leftarrow$

α:=1 LASSO

α=0: Ridge

$\rightarrow$

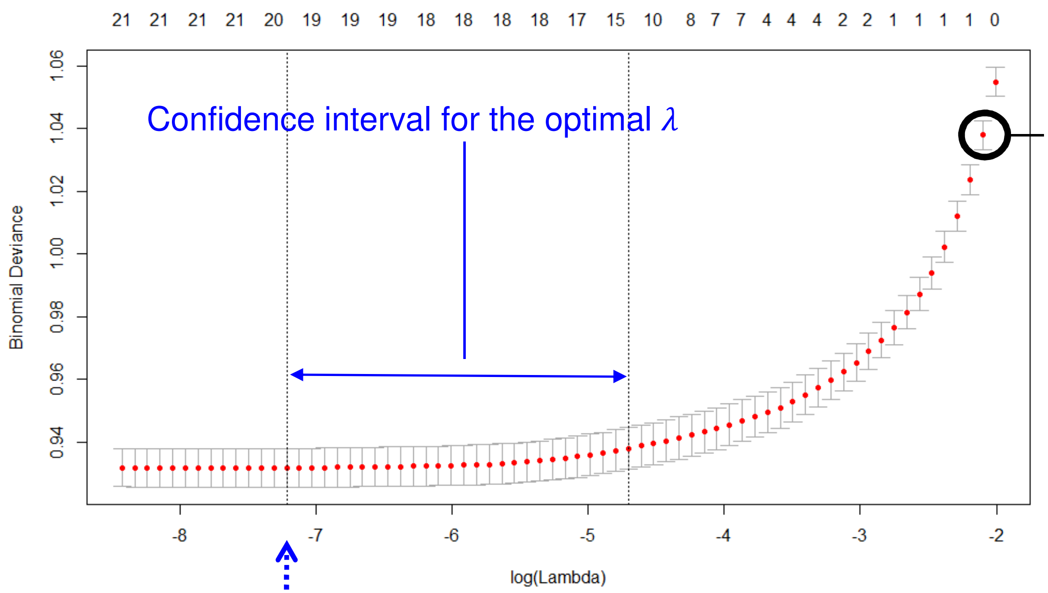


Understanding LASSO/Ridge: selecting λ

How to determine the [optimal] value of penalty, λ?

Through cross-validation (of course!)

Number of variables in the model



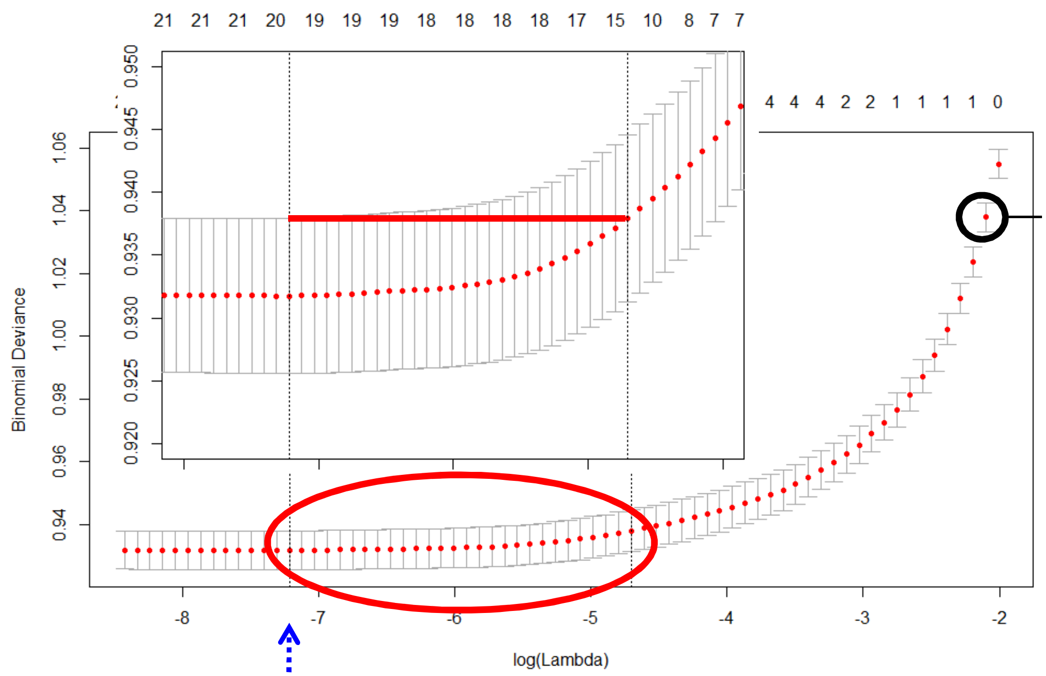
Confidence interval for MSE (from 5-fold cross validation)

```
> penalty.lasso <- crossval$lambda.min #determine optimal penalty parameter
> log(penalty.lasso) #see where it was on the graph
[1] -7.217623
```

Understanding LASSO/Ridge: selecting λ

How to determine the [optimal] value of penalty, λ?

Through cross-validation (of course!)



Confidence interval for MSE (from 5-fold cross validation)

```
> penalty.lasso <- crossval$lambda.min #determine optimal penalty parameter
> log(penalty.lasso) #see where it was on the graph
[1] -7.217623
```

LASSO/Ridge for A2 package `glmnet`

```
pacman::p_load("caret","ROCR","lift","glmnet","MASS","e1071")
#create the y variable and matrix (capital X) of x variables (will make the code below easier to read + will ensure that all levels exist)
y<-training$default_0
X<-model.matrix(ID ~. - default_0 , data=credit_data_24000)[,-1]
X<-cbind(credit_data_24000$ID,X)
# split X into testing, trainig/holdout and prediction as before
X.training<-subset(X,X[,1]<=23000)[,-1]
X.testing<-subset(X, X[,1]>=23001)[,-1]
# create model matrix for the new applicants, first adjust data for some minor "glitches" in formatting
new_applicants<-new_applicants[,-25]
new_applicants[,1]<-c(1:1000)
X.prediction<-model.matrix(ID ~. , data = new_applicants)[,-1]

#LASSO (alpha=1)
lasso.fit<-glmnet(x = X.training, y = y, alpha = 1, family="binomial")
plot(lasso.fit, xvar = "lambda")
#selecting the best penalty lambda
crossval <-  cv.glmnet(x = X.training, y = y, alpha = 1, family="binomial") #create cross-validation data
plot(crossval)
penalty.lasso <- crossval$lambda.min #determine optimal penalty parameter, lambda
log(penalty.lasso) #see where it was on the graph
plot(crossval,xlim=c(-8,-4),ylim=c(0.92,0.95)) # lets zoom-in
lasso.opt.fit <-glmnet(x = X.training, y = y, alpha = 1, lambda = penalty.lasso, family="binomial") #estimate the model with the optimal penalty
coef(lasso.opt.fit) #resultant model coefficients
# predicting the performance on the testing set
lasso_probabilities <- predict(lasso.opt.fit, s = penalty.lasso, newx =X.testing, family="binomial",type="response")
write.csv(lasso_probabilities, file="predicted_default_probs_LASSO_testing.csv")
```

#ridge (alpha=0)

[Optional/Time-permitting]
Artificial Neural Networks (Deep Learning)

Example: ImageNet

Deep Learning = Non-linear feature engineering + BIG Data + Regularizations

INSEAD

ImageNet classification with deep convolutional neural networks

Authors:
Alex Krizhevsky University of Toronto
Ilya Sutskever University of Toronto
Geoffrey E. Hinton University of Toronto

Published in:

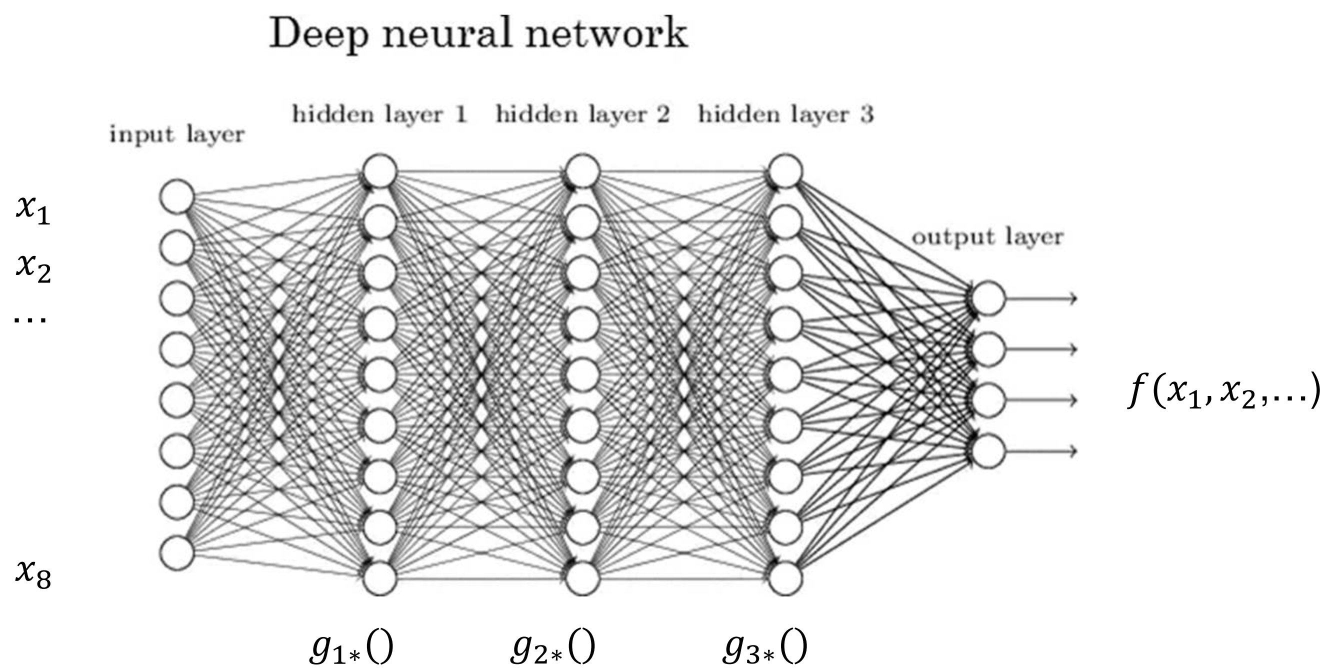
- Proceeding

NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1

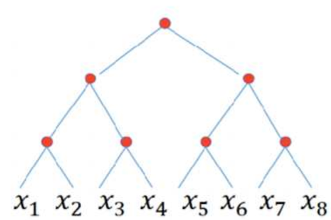
Pages 1097-1105

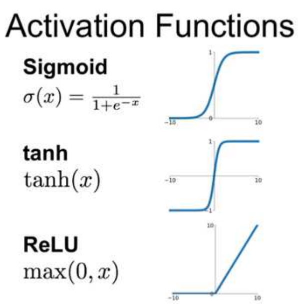
We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overriding in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

Deep Learning = “sexy” rebranding of Neural Networks (+ BIG Data)



$$f(x_1,x_2,\ldots,x_8)=g_3(g_{21}(g_{11}(x_1,x_2),g_{12}(x_3,x_4))g_{22}(g_{11}(x_5,x_6),g_{12}(x_7,x_8)))$$

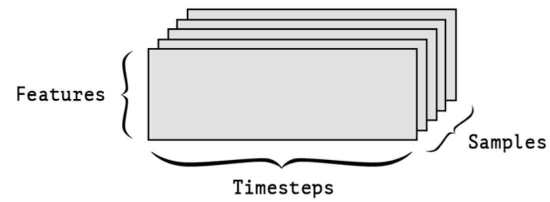


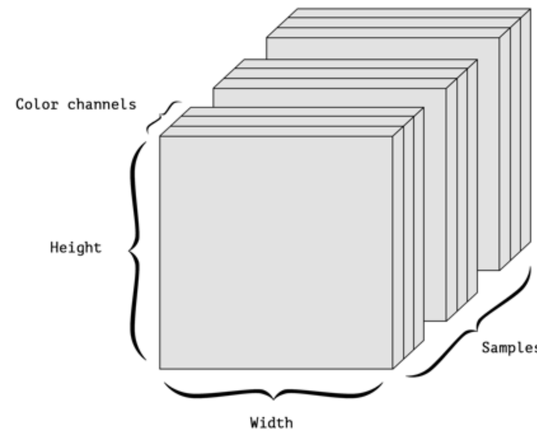


Deep Learning in R packages `TensorFlow` and `Keras`

- What is TensorFlow? https://tensorflow.rstudio.com/
  - Library for training Deep Learning models by Google (~90th percentile for R downloads = popular)
  - "Tensor" = matrix, "Flow" = process of training the neural network

Data	Tensor
Vector data	2D tensors of shape `(samples, features)`
Timeseries data	3D tensors of shape `(samples, timesteps, features)`
Images	4D tensors of shape `(samples, height, width, channels)`
Video	5D tensors of shape `(samples, frames, height, width, channels)`

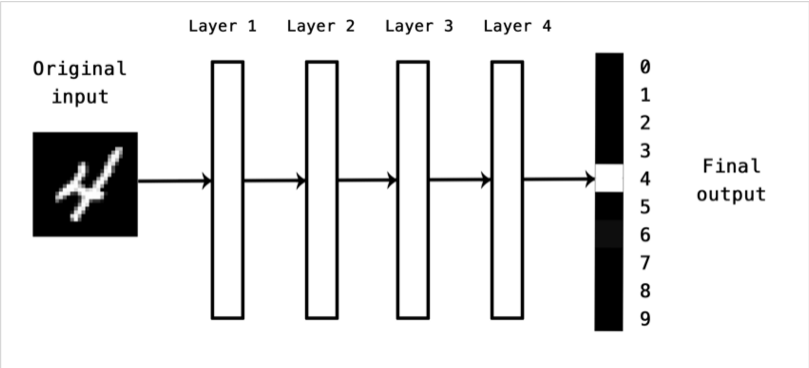


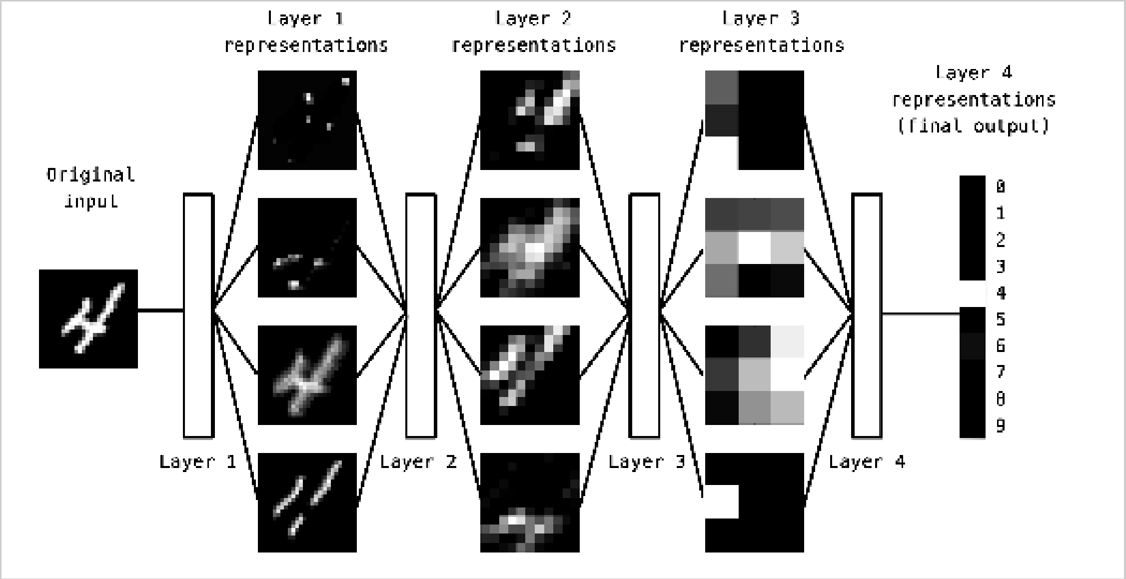


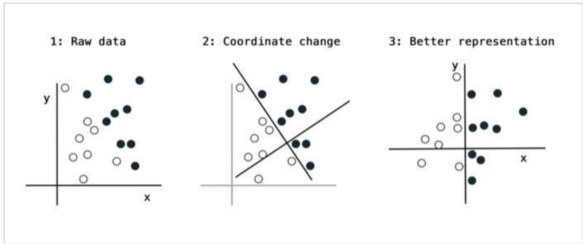
- What is Keras?
  - (Python-based) library for controlling TensorFlow at the "high-level" (without too much coding)

Key Concepts of Neural Networks (Deep Learning)

Layers, units, representations, weights, activation functions



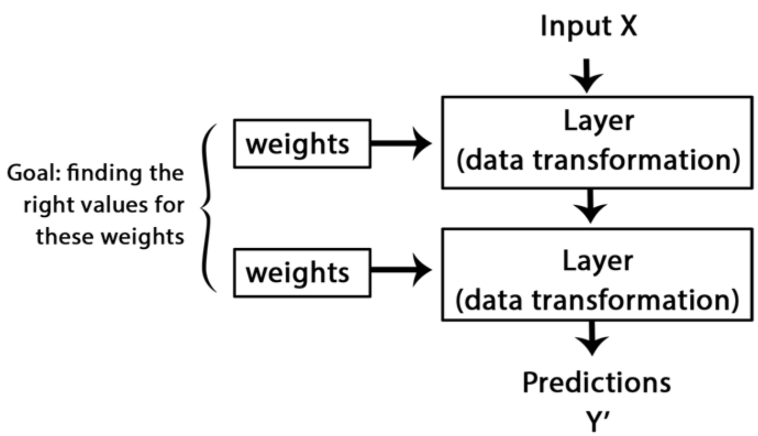


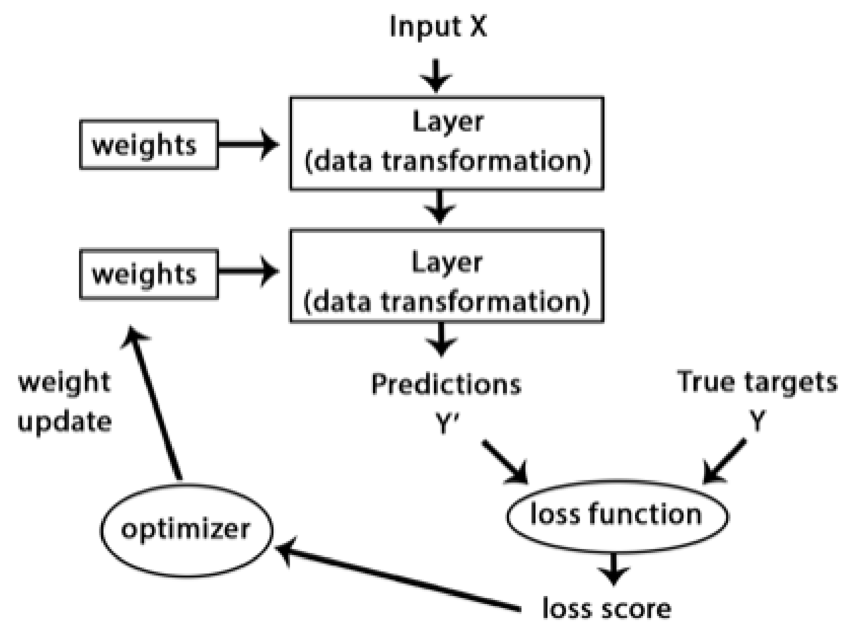


Units at Layers transform the data: “represent” it differently

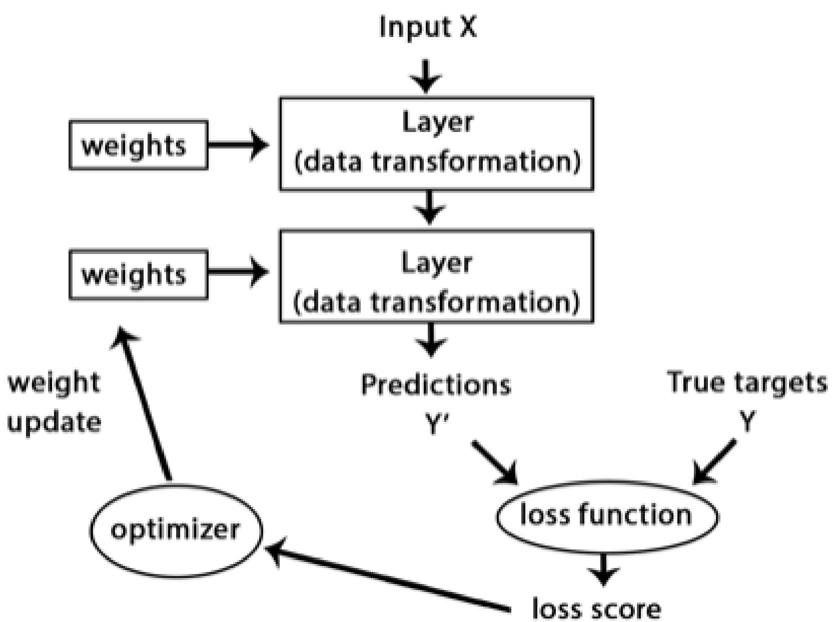
Weights and activation functions control how the transformations work

Key Concepts of Neural Networks (Deep Learning)





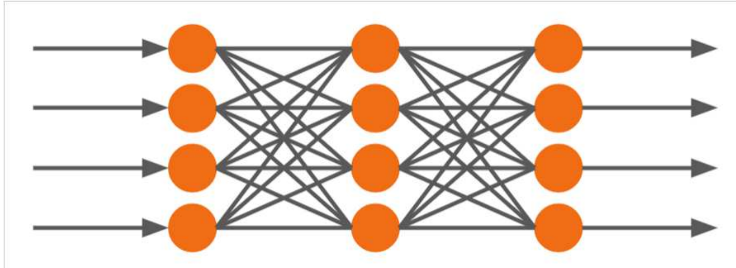
Key Concepts of Neural Networks (Deep Learning)



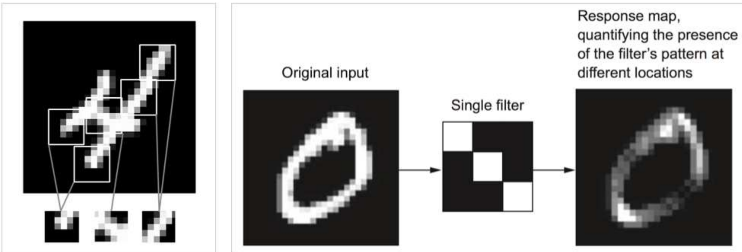

```
optimizer_adadelta()
optimizer_adagrad()
optimizer_adam()
optimizer_adamax()
optimizer_nadam()
optimizer_rmsprop()
optimizer_sgd()
```

```
loss_binary_crossentropy()
loss_categorical_crossentropy()
loss_categorical_hinge()
loss_cosine_proximity()
loss_hinge()
loss_kullback_leibler_divergence()
loss_logcosh()
loss_mean_absolute_error()
loss_mean_absolute_percentage_error()
```

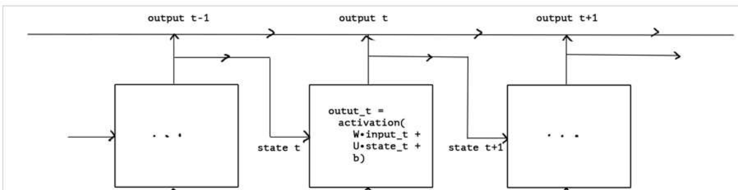
`layer_dense()` Dense layer (generically used) / "Dropout"



`layer_conv_2d()` Convolutional layer (images)



`layer_simple_rnn()`
`layer_gru()`
`layer_lstm()`
Recurrent layer (text, time-series) / LSTM



Awesome cheat-sheet: https://github.com/rstudio/cheatsheets/raw/master/keras.pdf

Deep Learning for A2 packages `TensorFlow` and `Keras`

Note: installing Tensorflow and Keras for the first time requires 2 extra steps:
`install_tensorflow()`
`install_keras()`

INSEAD

Main steps: **data pre-processing into Tensors** → defining the model (layers, activations, etc.) → compiling → training/fitting → ... [the rest is the same as with other methods]

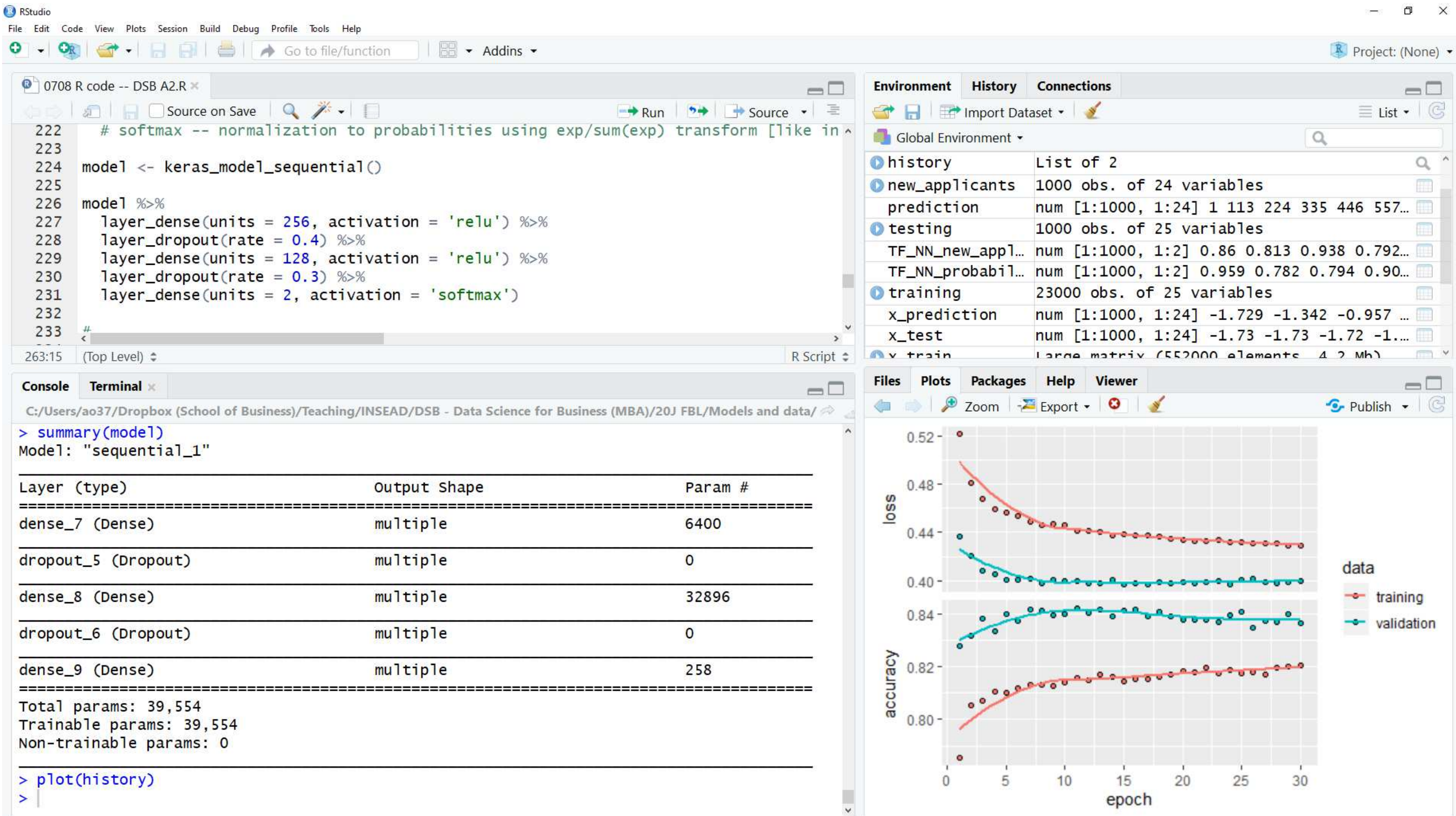
```
# Preprocessing data for inputting into Keras
# Tensors are matrices... hence the input data has to be in a form of a matrix

x_train <- data.matrix(training[,-25]) #matrix of features ("X variables") for
training; remove the "default_0" column number 25
y_train <- training$default_0 #target vector ("Y variable") for training

x_test <- data.matrix(testing[,-25])
y_test <- testing$default_0

x_train <- array_reshape(x_train, c(nrow(x_train), 24)) # Reshape -- Keras interprets
data using row-major semantics (as opposed to R's default column-major semantics)
x_test <- array_reshape(x_test, c(nrow(x_test), 24))

# final data preparation steps: scaling for X and converting to categorical for Y
x_train <- scale(x_train)
x_test <- scale(x_test)
y_train <- to_categorical(y_train, 2)
y_test <- to_categorical(y_test, 2)
```

Deep Learning for A2
packages `TensorFlow` and `Keras`

Main steps: data pre-processing into Tensors → **defining the model (layers, activations, etc.)** → **compiling** → **training/fitting** → … [the rest is the same as with other methods]

```
model <- keras_model_sequential()
model %>%
  layer_dense(units = 256, activation = 'relu') %>% # 1st layer and its parameters
  layer_dropout(rate = 0.4) %>%                     # 2nd layer and its parameters
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 2, activation = 'softmax')    # last layer for classification

model %>% compile(
  loss = 'binary_crossentropy',
  optimizer = 'adam',
  metrics = c('accuracy'))

history <- model %>% fit(
  x_train, y_train,         # on what data to train
  epochs = 30,              # how many repetitions to have
  batch_size = 256,         # how many datapoints are fed to the network at a time
  validation_split = 0.2) # percentage of training data to keep for cross-validation
summary(model)
plot(history)
```

Deep Learning for A2 packages `TensorFlow` and `Keras`



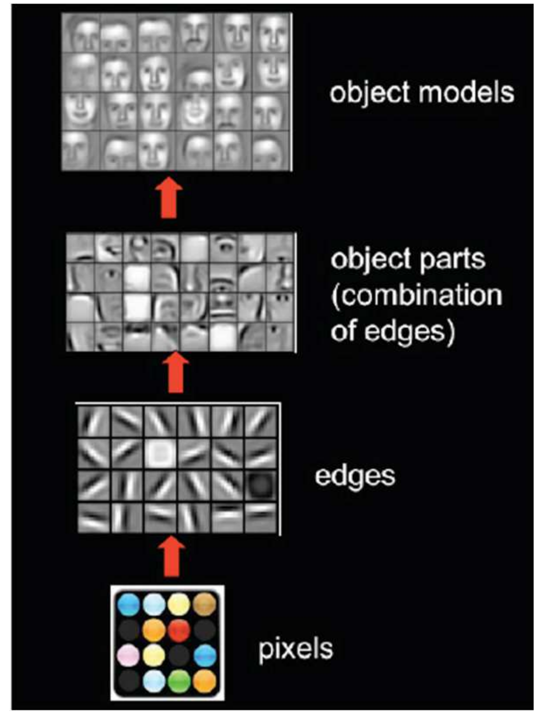
So what is Deep Learning?

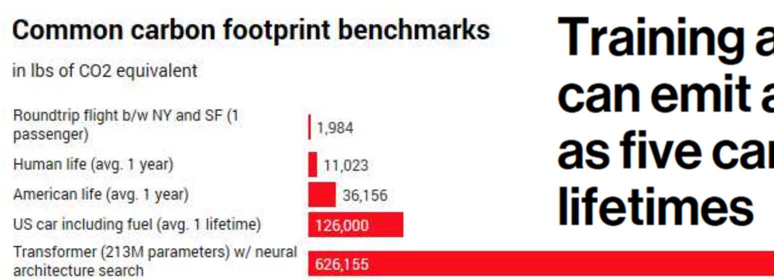
Deep Learning =

- Highly-non-linear feature engineered learning through representations
- + Big Data
- + Regularizations

In practice:

- LOTS of experimentation (many more hyper-parameters than other models)
- VERY computationally intensive
- … but allows for transfer learning
- … and achieves amazing results when truly BIG data is available (lots or it, and feature-rich / ”high-entropy”)



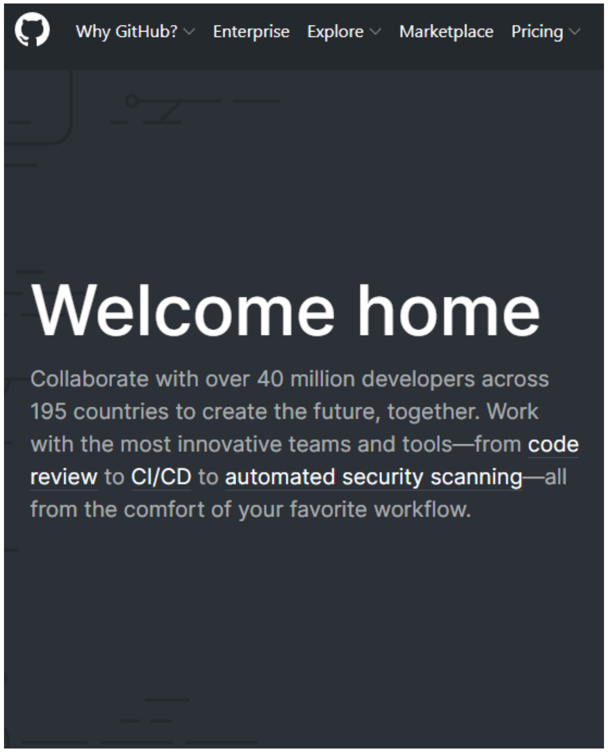


Training a single AI model can emit as much carbon as five cars in their lifetimes

Summarizing:
Supervised Learning from `lm` to Deep Learning

- Start by understanding your data
  - Data-generating (“business”) process, Data dictionary, Visualizations (Tableau, etc.), Structure/Summary of data
- Add feature engineering
  - Use many and highly **non-linear** functions
  - On **high dimensional** (“feature-rich”) data
- Tune hyper-parameters / “regularize”, i.e., **manage over-fitting**
  - The goal is not to build a model that explains past, but rather one that predicts future: Train/**Test**/Cross-validate
- All is improved using **Big Data** (more opportunities for feature engineering)
- And all is fueled by **open source** (R/python + packages, >18000 as of Feb 2020) and online collaboration (online communities, GitHub)

What is GitHub?
Leading soft for coding collaborations ("version control")




Microsoft to acquire GitHub for $7.5 billion

June 4, 2018 | Microsoft News Center

Acquisition will empower developers, accelerate GitHub's growth and advance Microsoft services with new audiences

Our DS(ML)B course is also on GitHub:
http://inseaddataanalytics.github.io/INSEADAnalytics/home.html [undergoing some updating]

Next…

- Sessions 9-10: **Unsupervised** Learning:
  - Clustering and Segmentation: K-means, hierarchical clustering
  - Dimensionality Reduction: Principle Component Analyses (PCA)
  - [Time-permitting]: Association rules, Anomaly detection

- Assignment 3 [due by the beginning of 11-12]: Market segmentation for Boats (A) case

- Proposal for final project due TBA
  - Feedback will be provided **ASAP** to give you more time for projects

- Tutorial 3: "Democratizing AI: auto-ML, AI PaaS – DataRobot demo/speaker
- Sessions 11-12: Guest speaker, Elias Baltassis, BCG: "AI in business: Data science process and use-cases"
- Sessions 13-14: Project presentations

INSEAD

The Business School for the World®

Europe | Asia | Middle East