indeed

Using Open Source Tools for Machine Learning

Samuel Taylor
Data Scientist

AMERICAN EXPRESS
BUSINESS
3759 876543 21
BGMC1014C
YOUR BUSINESS NAME
© AMEX
Master

Ryan Born

We help people get jobs.

THIS TALK IS

- An introduction to ML
- Friendly to newcomers
- Helpful to experienced folks
- Oriented toward application
- Respectful of theory

THIS TALK IS NOT

- A substitute for a Ph. D.
- The end-all, be-all
- A detailed tutorial

Agenda

Machine learning intro

UC0: Credit card applications

UC1: Teach a computer ASL

UC2: Forecasting energy load

UC3: Use ML to find your next job

Agenda

Machine learning intro

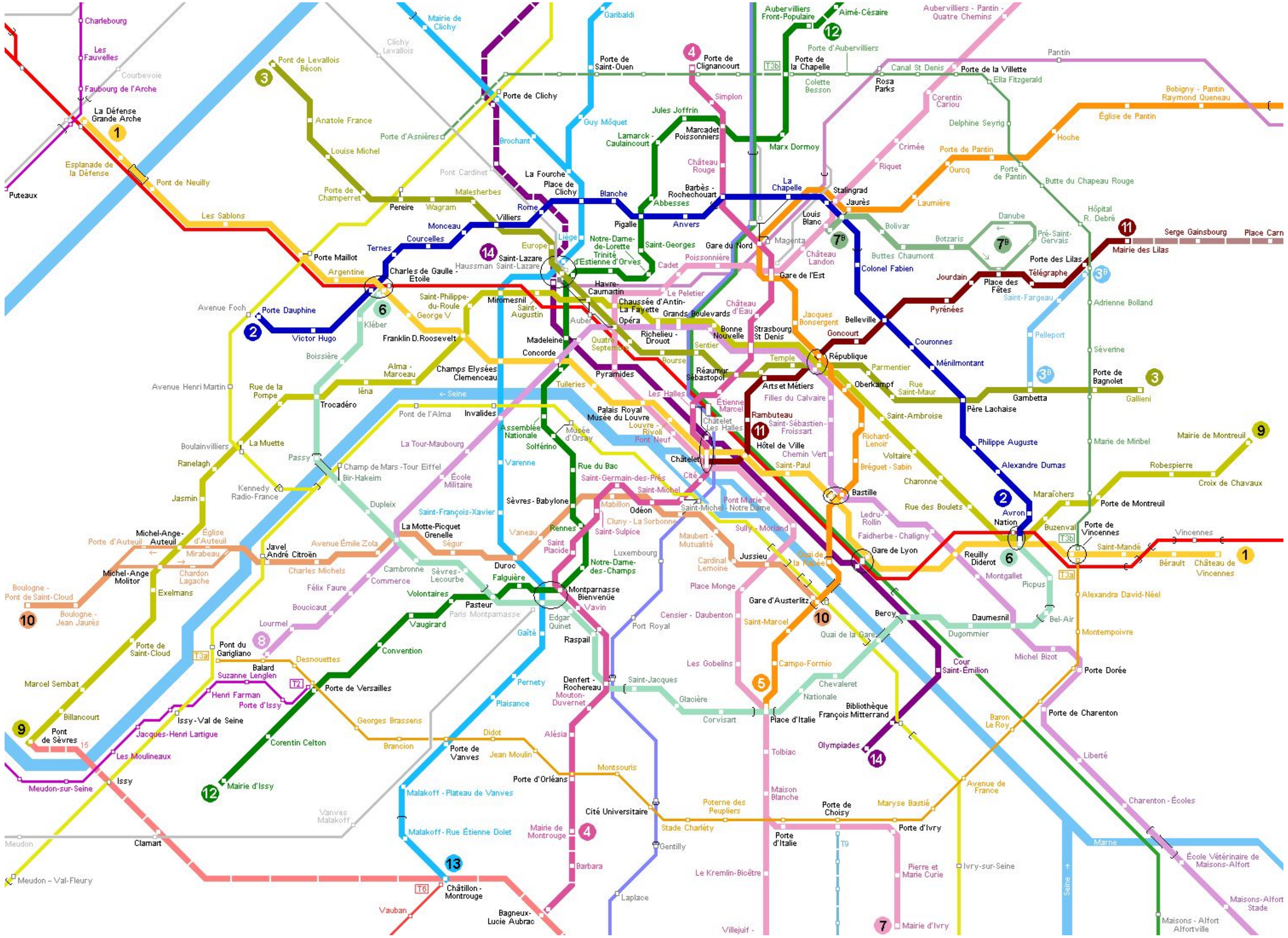
UC0: Credit card applications

UC1: Teach a computer ASL

UC2: Forecasting energy load

UC3: Use ML to find your next job

Machine learning?



urbanrail.net

Machine learning

Supervised	Unsupervised	Other stuff (lots)

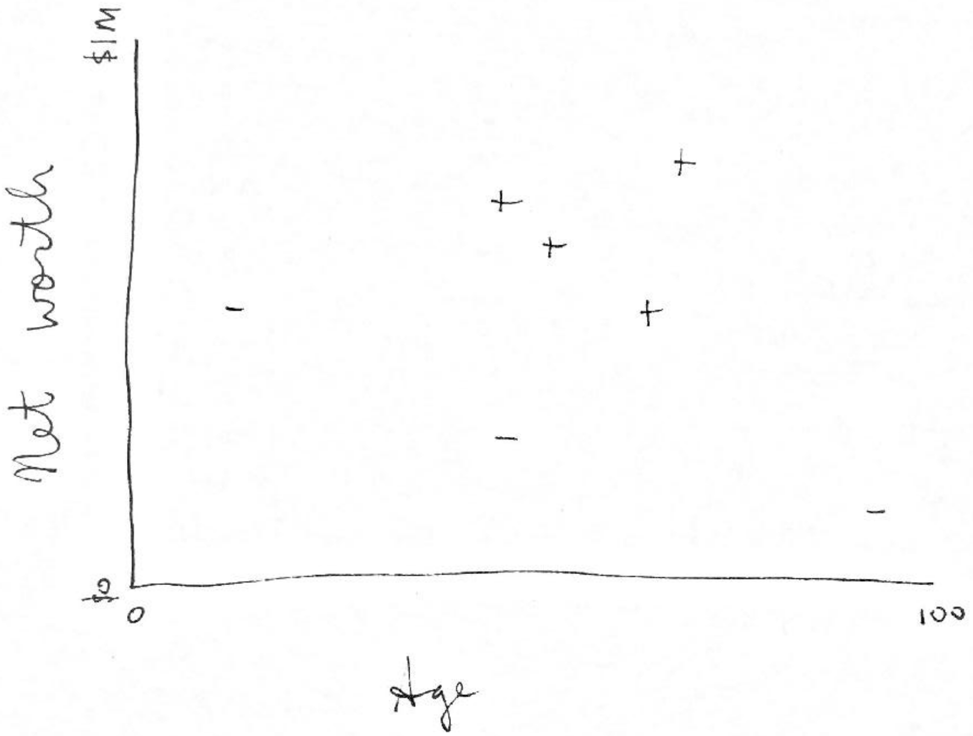
Machine learning

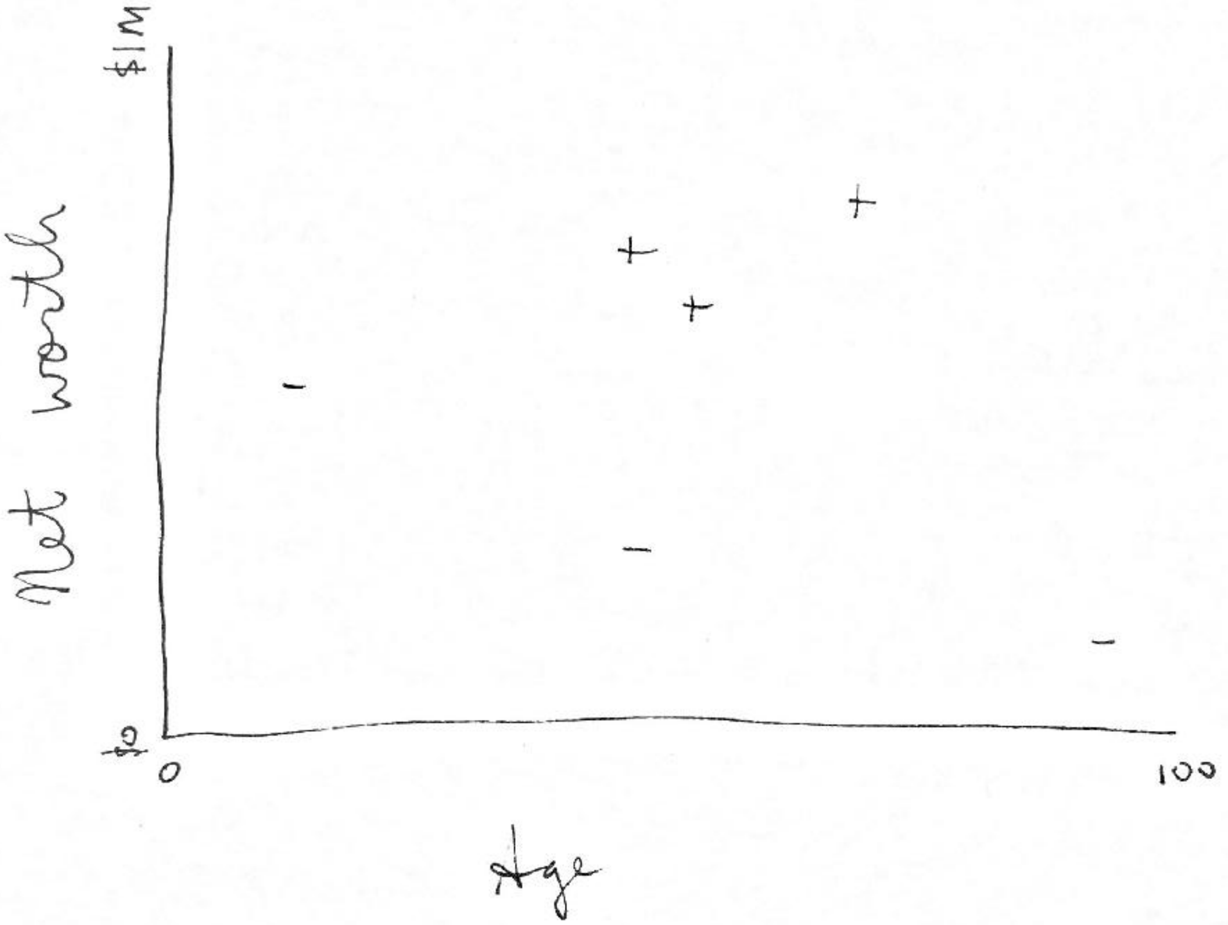
Supervised
Classification
Regression

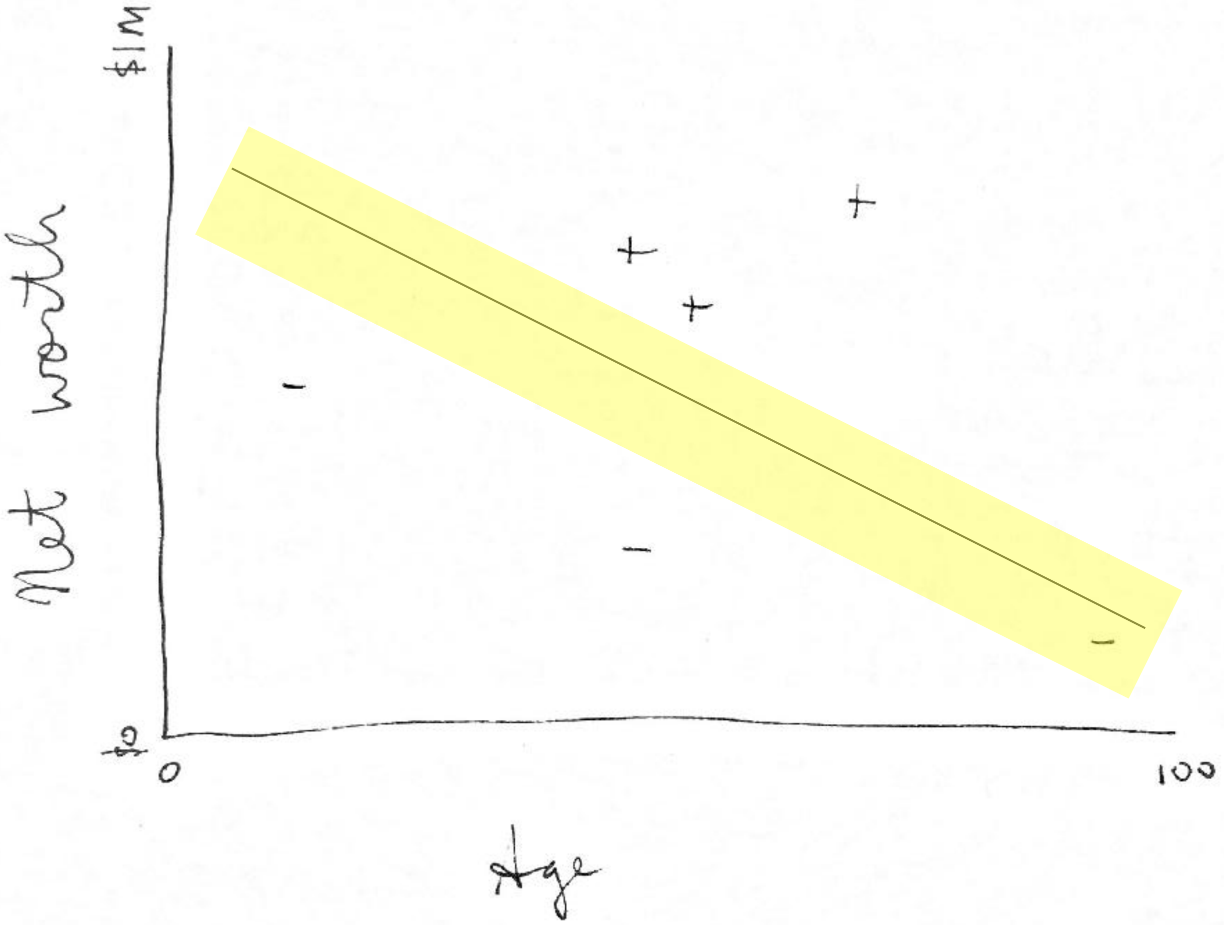
Unsupervised

Other stuff (lots)

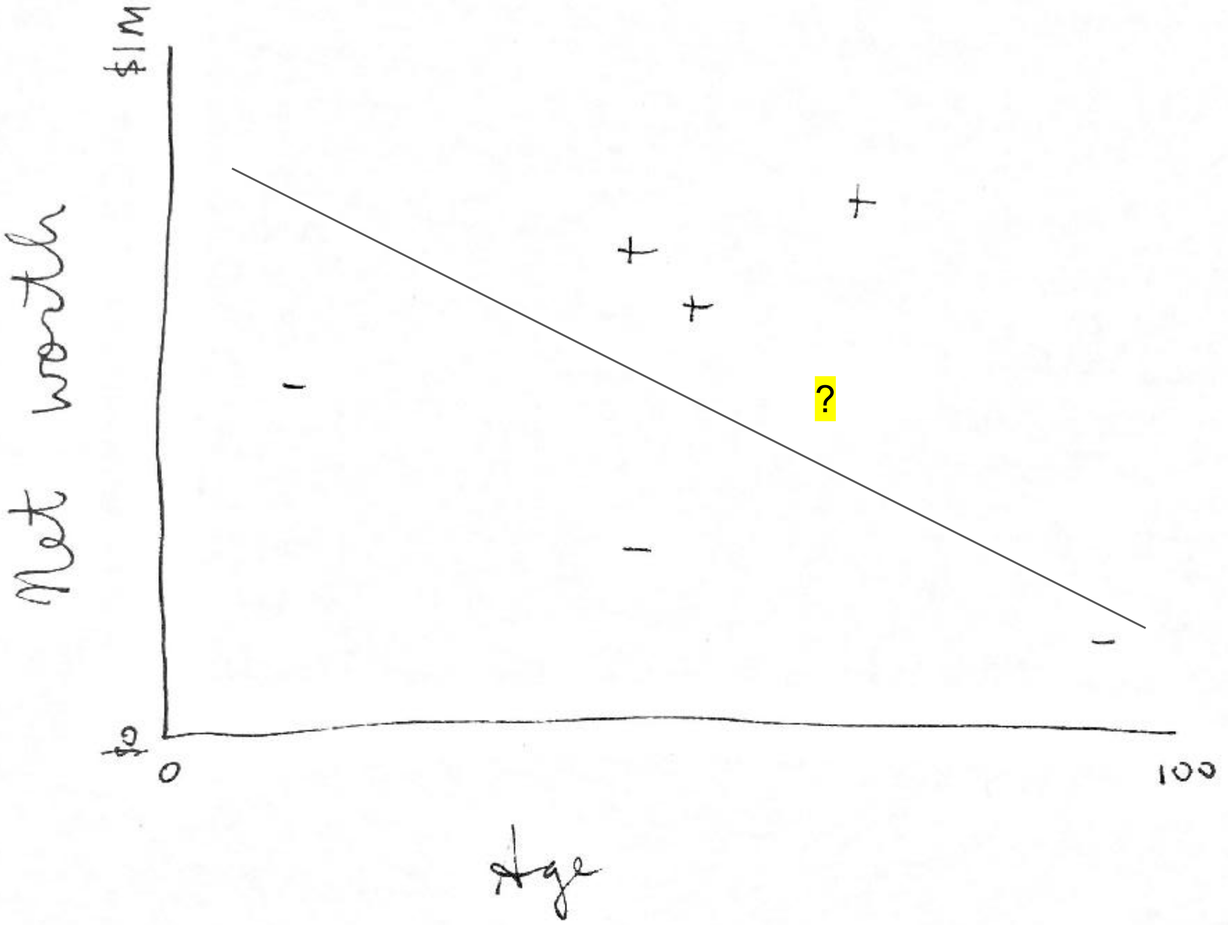
Age	Net worth	Given credit?
12.5	$500K	No
50	$250K	No
97	$90K	No
50	$750K	Yes
53	$650K	Yes
60	$500K	Yes
62	$800K	Yes

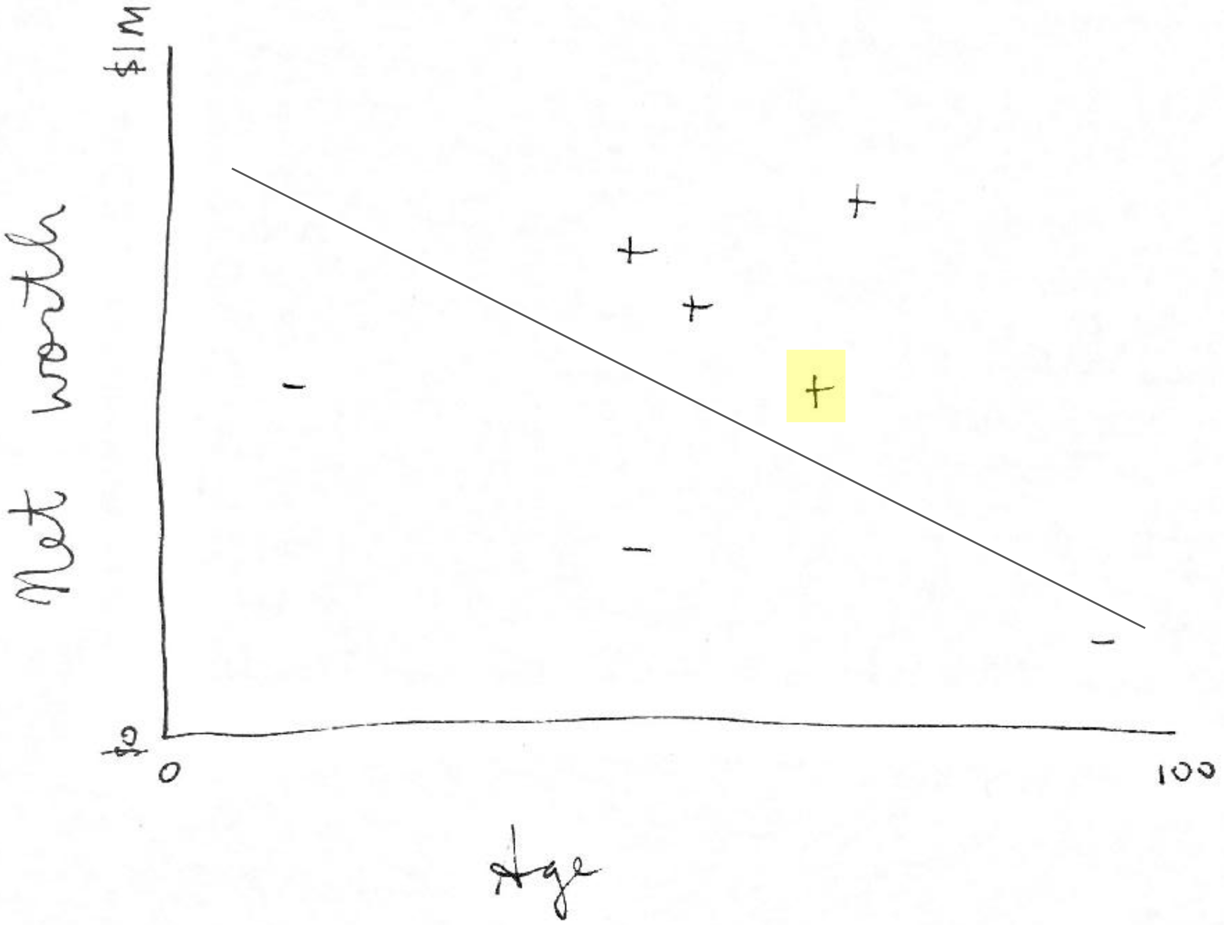



$1M
Net worth
$0
0
100
Age



$1M
Net worth
$0
0
100
Age
+
+
+
-
-
-


$1M
Net worth
$0
0
100
Age
?

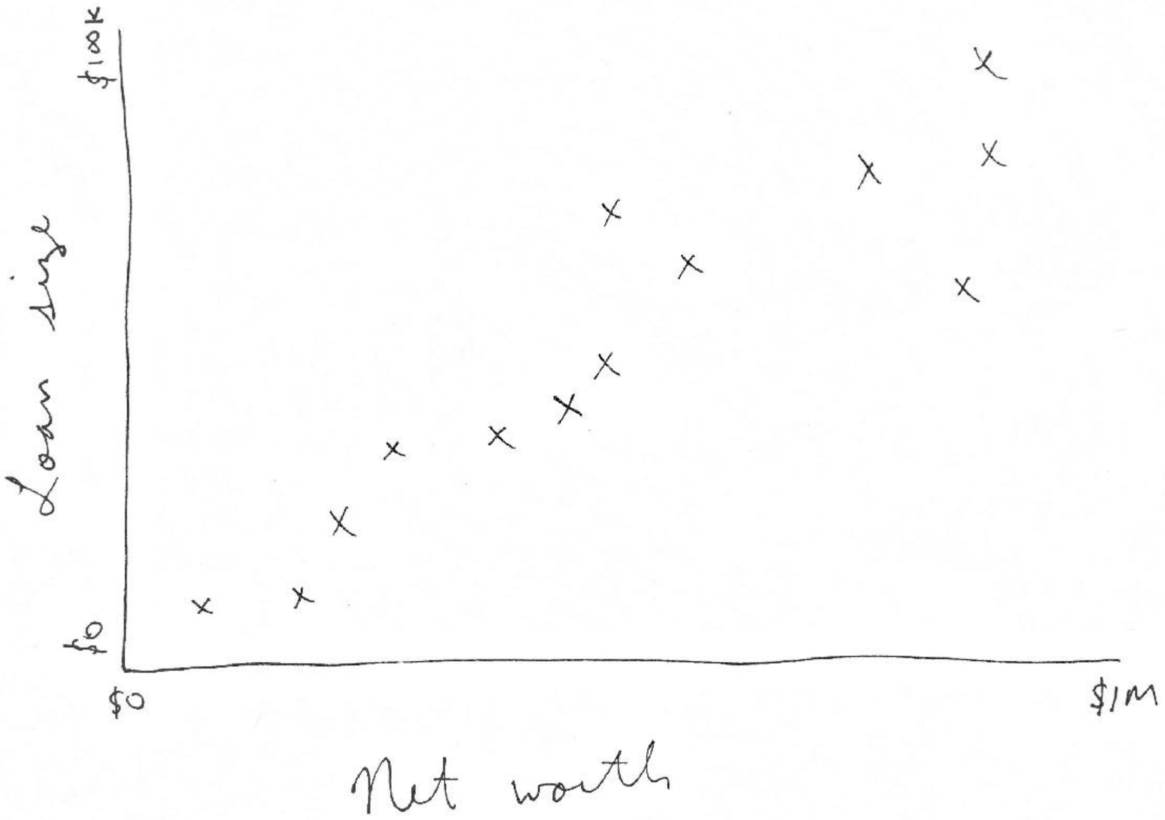

$1M
Net worth
$0
0
100
Age

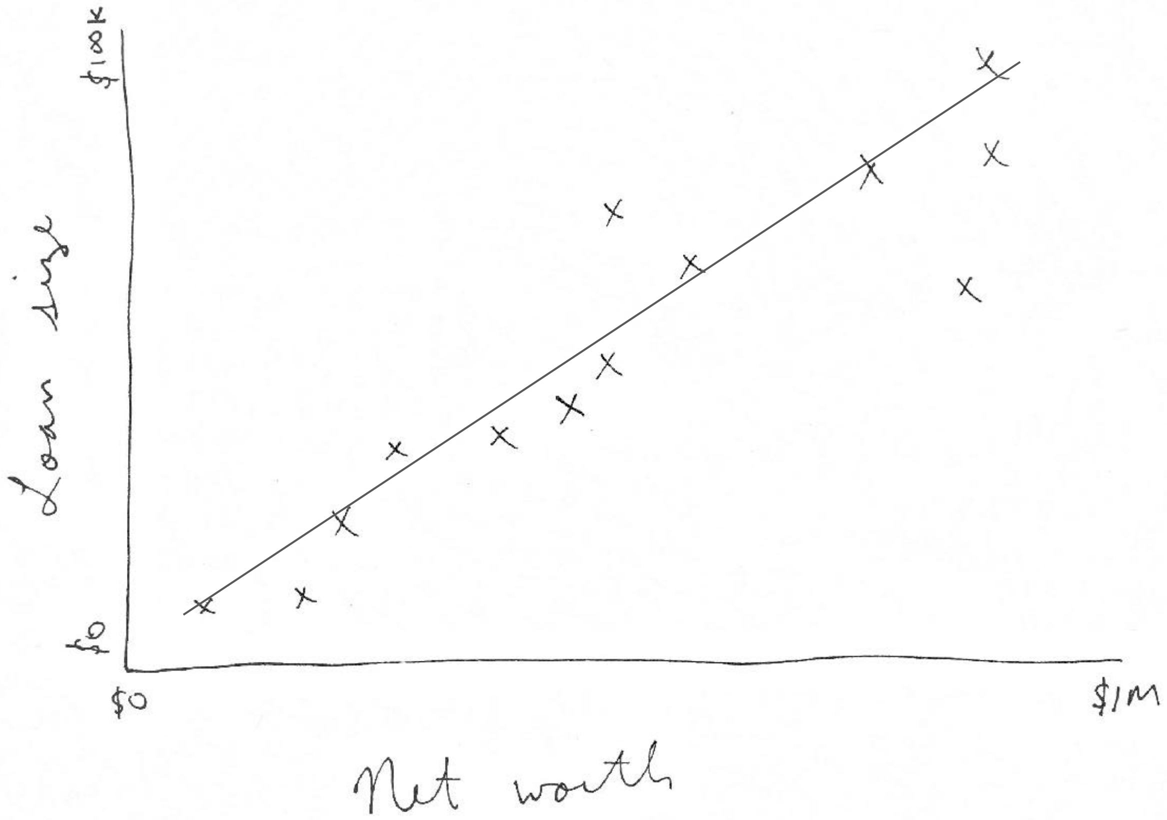
Machine learning

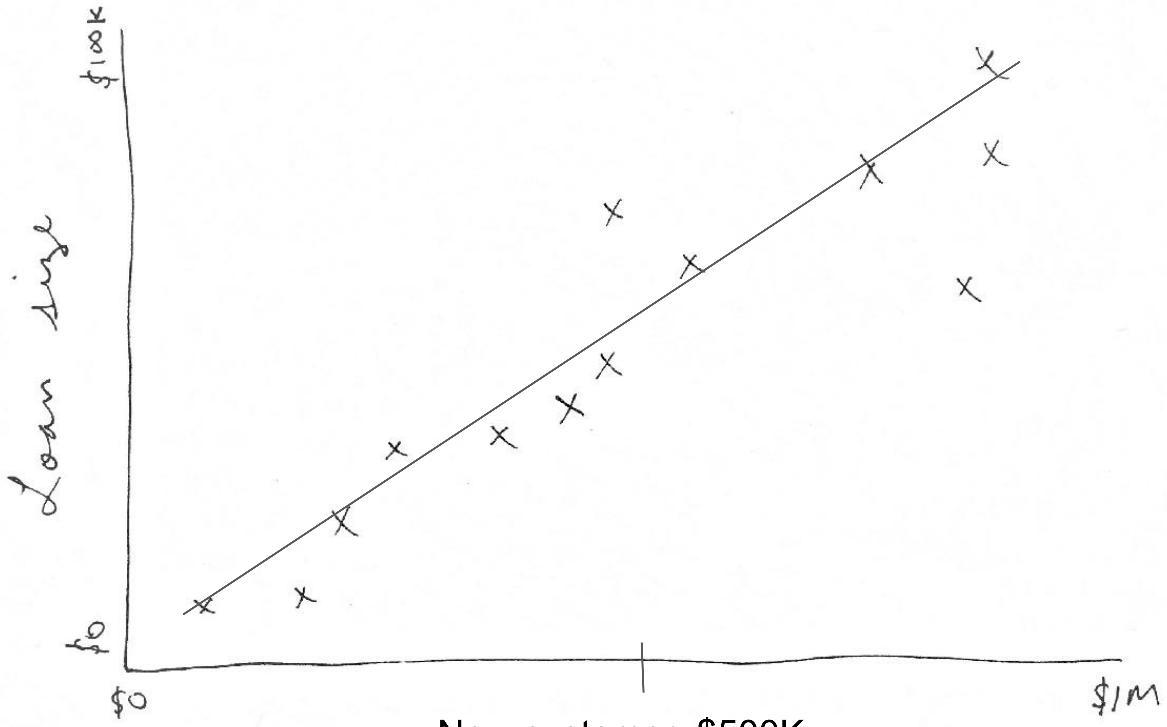
Supervised
Classification
Regression

Unsupervised

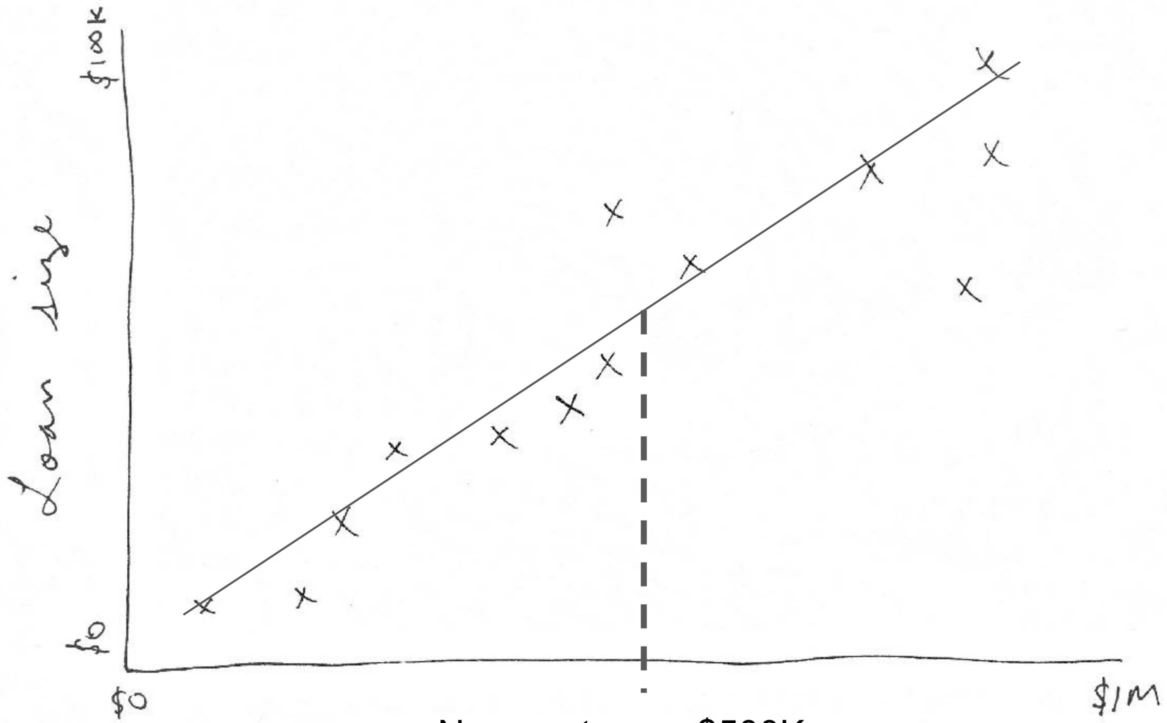
Other stuff (lots)


$100K
$0
Loan size
$0
$1M
Net worth

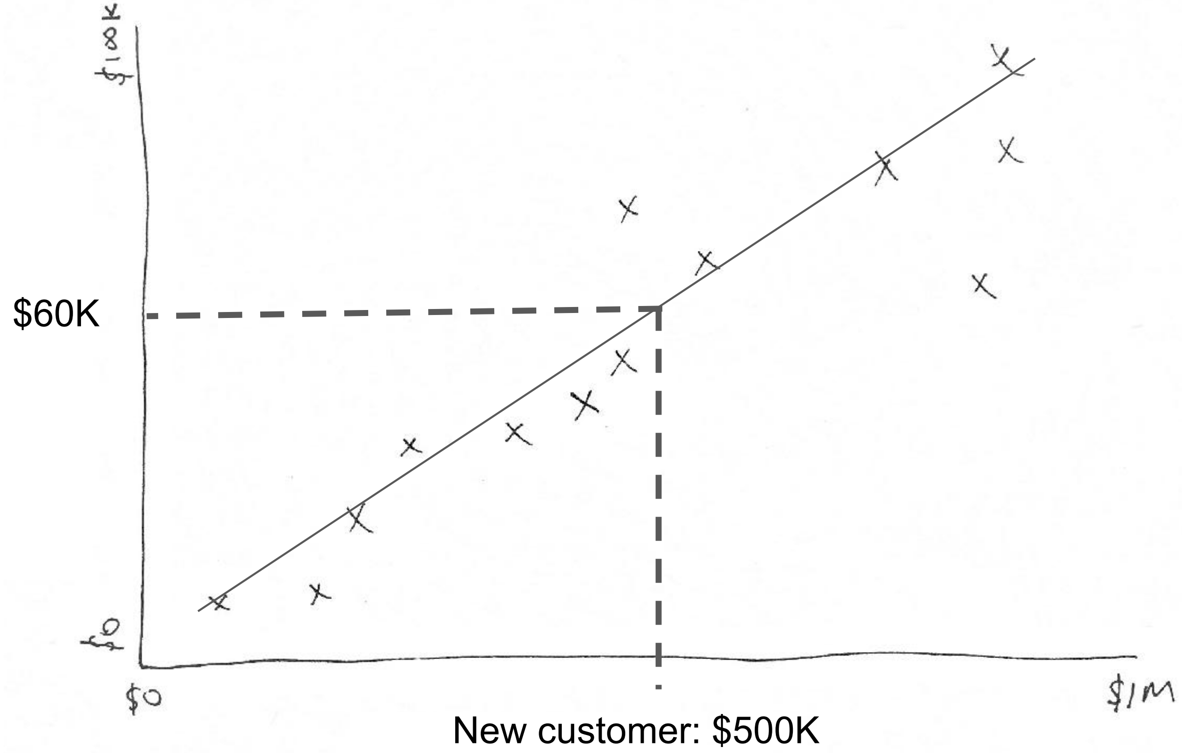

$100K
$0
Loan size
$0
$1M
Net worth



New customer: $500K



New customer: $500K

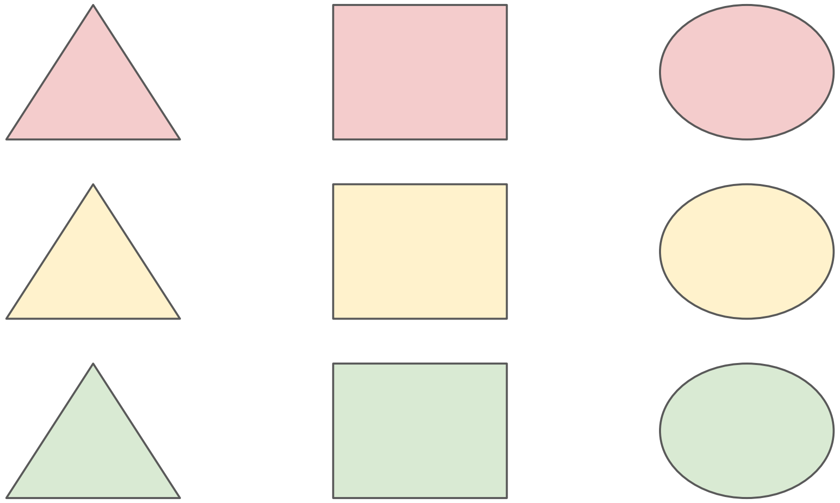

$100K
$0
$60K
$0
$1M
New customer: $500K

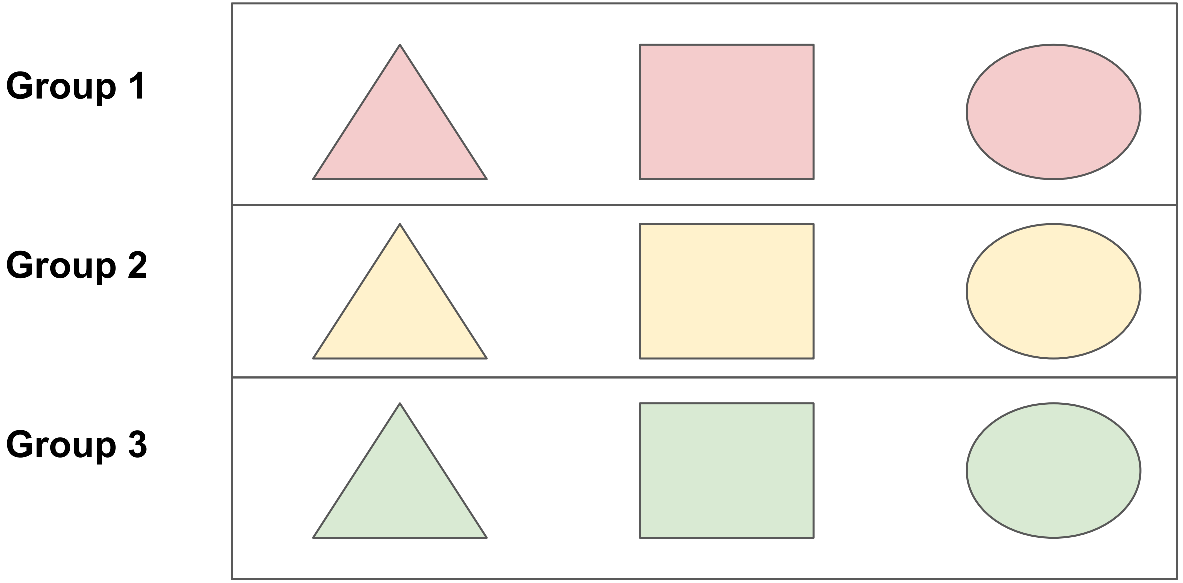
Machine learning

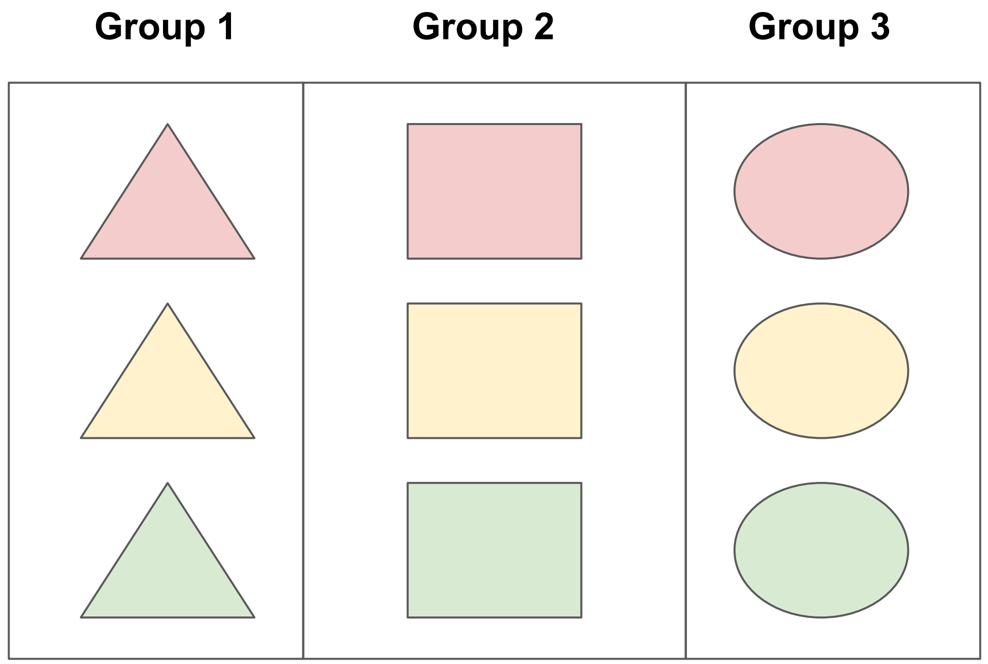
Supervised

Unsupervised
Clustering

Other stuff (lots)




Group 1
Group 2
Group 3


Group 1
Group 2
Group 3

Machine learning

Supervised

Unsupervised

Other stuff
(lots)

ML uses data to approximate something we care about

➔ Goal Find *f(x)*

ML uses data to approximate something we care about

- ➔ Goal Find *f(x)*
- ➔ **Problem** ***f(x)*** **is unknown**
  (perhaps unknowable)

ML uses data to approximate something we care about

- ➔ **Goal Find *f(x)***
- ➔ **Problem *f(x)* is unknown**
  (perhaps unknowable)
- ➔ **But we can measure points from *f(x)***
  (with noise)

ML uses data to approximate something we care about

- ➔ Goal Find *f(x)*
- ➔ Problem *f(x)* is unknown (perhaps unknowable)
- ➔ But we can measure points from *f(x)* (with noise)
- ➔ Algorithms to find a *g(x)* that approximates *f(x)*

Agenda

Machine learning intro

UC0: Credit card applications

UC1: Teach a computer ASL

UC2: Forecasting energy load

UC3: Use ML to find your next job

- What’s the problem?
- What does the data look like?
- What kind of ML problem is this?
- Solution
- Lessons learned

➔ What’s the problem?

Should we (the bank)
give this consumer a credit card?

→ What does the data look like?

Age	Net worth (K$)	Given credit?
12.5	500	No
50	250	No
62	800	Yes
50	750	Yes
53	650	Yes
60	500	Yes

→ **What kind of ML problem is this?**

➔ What kind of ML problem is this?

Classification

→ Solution

scikit learn

e1701, rpart, igraph, nnet, randomForest, caret, kernlab, ...

WEKA
The University of Waikato

➔ Solution

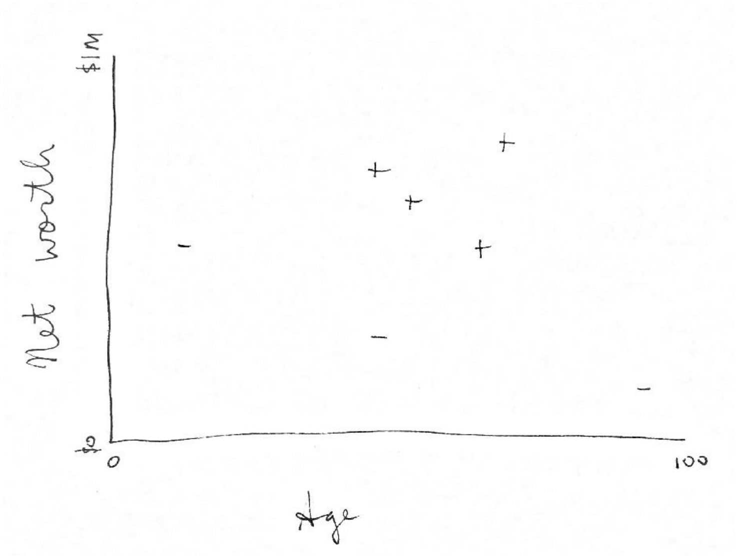
```
from sklearn.svm import LinearSVC

features = [(12.5, 500), (50, 250), (62, 800),
            (50, 750), (53, 650), (60, 500)]
given_credit = [False, False, True, True, True, True]

classifier = LinearSVC()
classifier.fit(features, given_credit)
classifier.predict([[63, 500]])
```

➔ Solution

```
from sklearn.svm import LinearSVC

features = [(12.5, 500), (50, 250), (62, 800),
            (50, 750), (53, 650), (60, 500)]
given_credit = [False, False, True, True, True, True]

classifier = LinearSVC()
classifier.fit(features, given_credit)
classifier.predict([[63, 500]])
```

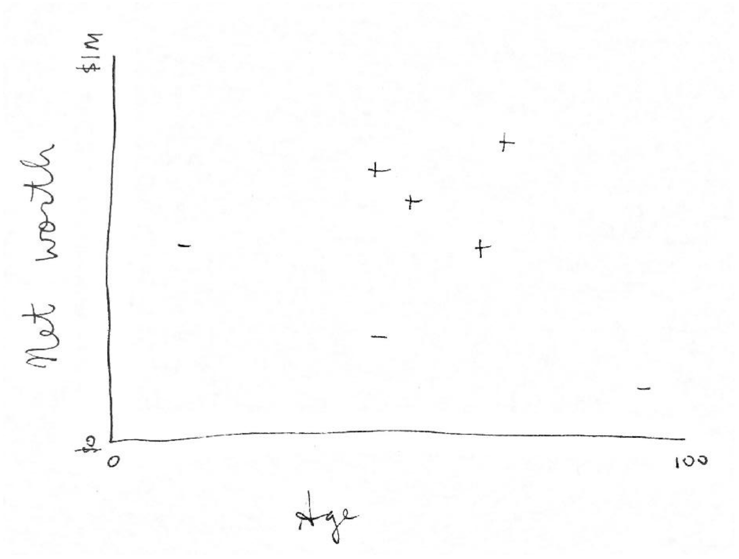
➔ Solution

```
from sklearn.svm import LinearSVC

features = [(12.5, 500), (50, 250), (62, 800),
            (50, 750), (53, 650), (60, 500)]
given_credit = [False, False, True, True, True, True]

classifier = LinearSVC()
classifier.fit(features, given_credit)
classifier.predict([[63, 500]])
```



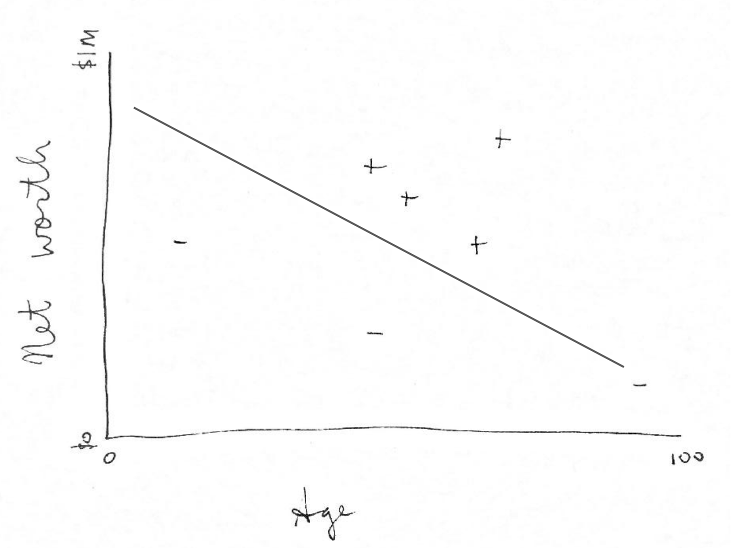
➔ Solution

```
from sklearn.svm import LinearSVC

features = [(12.5, 500), (50, 250), (62, 800),
            (50, 750), (53, 650), (60, 500)]
given_credit = [False, False, True, True, True, True]

classifier = LinearSVC()
classifier.fit(features, given_credit)
classifier.predict([[63, 500]])
```



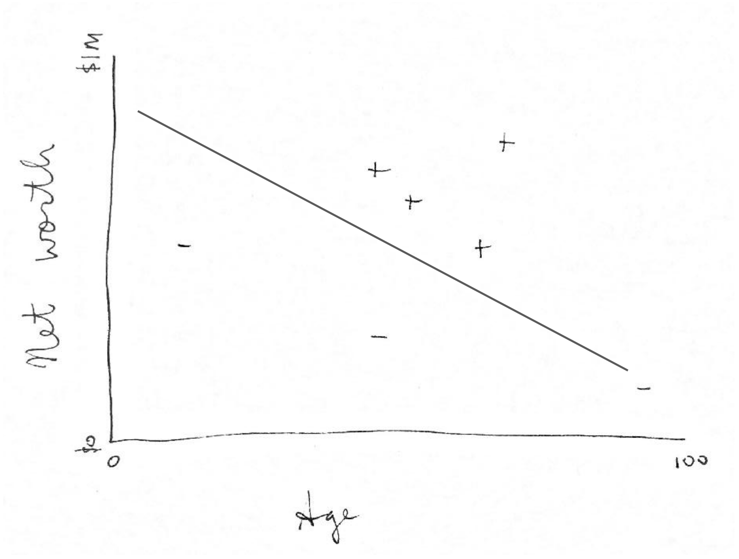
➔ Solution

```
from sklearn.svm import LinearSVC

features = [(12.5, 500), (50, 250), (62, 800),
            (50, 750), (53, 650), (60, 500)]
given_credit = [False, False, True, True, True, True]

classifier = LinearSVC()
classifier.fit(features, given_credit)
classifier.predict([[63, 500]])
```



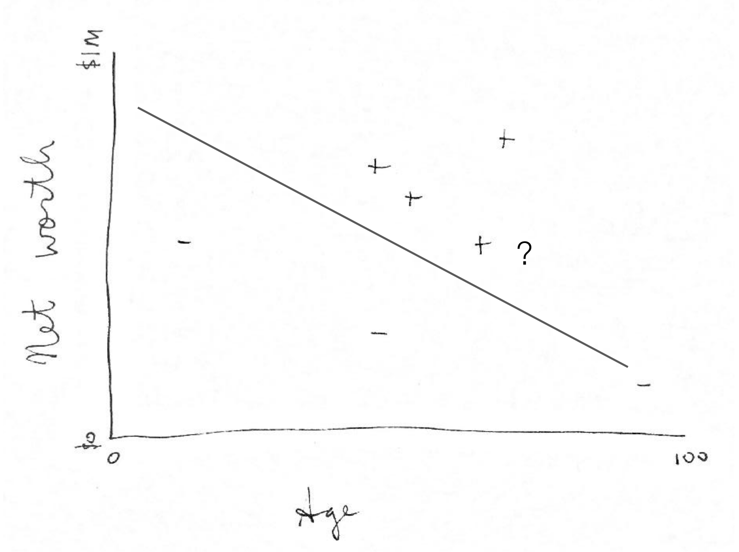
➔ Solution

```
from sklearn.svm import LinearSVC

features = [(12.5, 500), (50, 250), (62, 800),
              (50, 750), (53, 650), (60, 500)]
given_credit = [False, False, True, True, True, True]

classifier = LinearSVC()
classifier.fit(features, given_credit)
classifier.predict([[63, 500]])
```



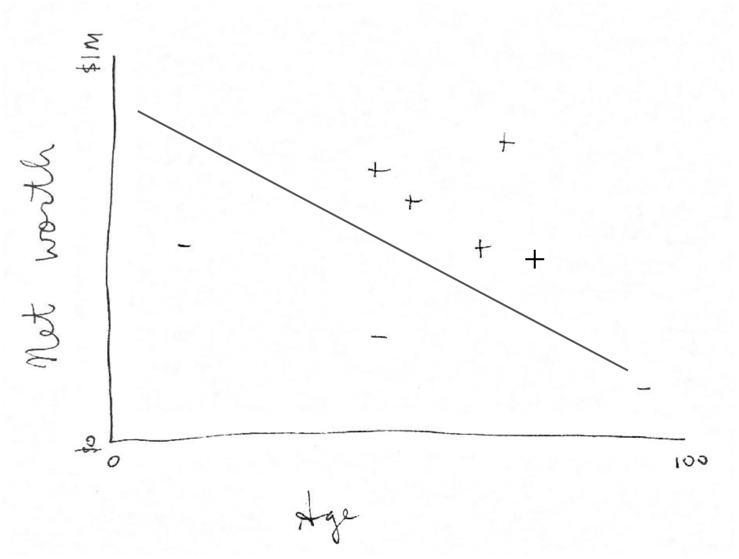
➔ Solution

```
from sklearn.svm import LinearSVC

features = [(12.5, 500), (50, 250), (62, 800),
            (50, 750), (53, 650), (60, 500)]
given_credit = [False, False, True, True, True, True]

classifier = LinearSVC()
classifier.fit(features, given_credit)
classifier.predict([[63, 500]])
```



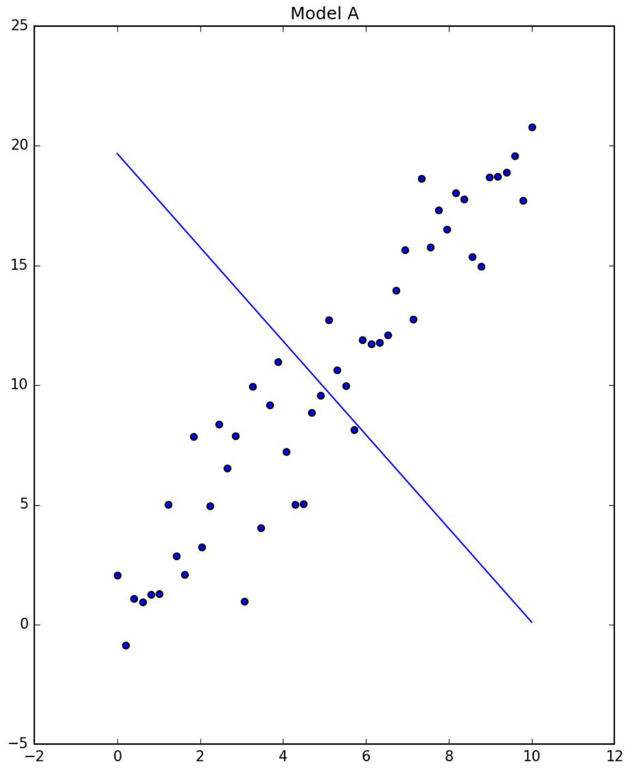
➔ Solution

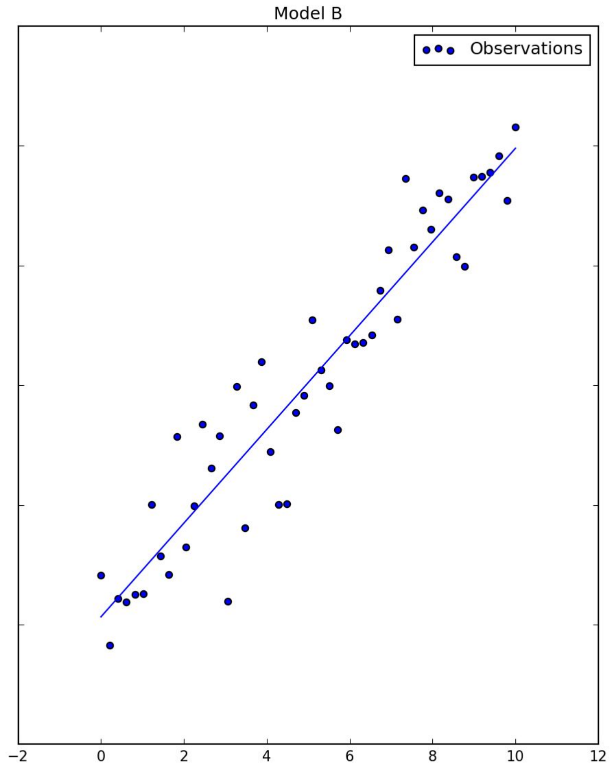
```python
from sklearn.svm import LinearSVC

features = [(12.5, 500), (50, 250), (62, 800),
            (50, 750), (53, 650), (60, 500)]
given_credit = [False, False, True, True, True, True]

classifier = LinearSVC()
classifier.fit(features, given_credit)
classifier.predict([[63, 500]])
```



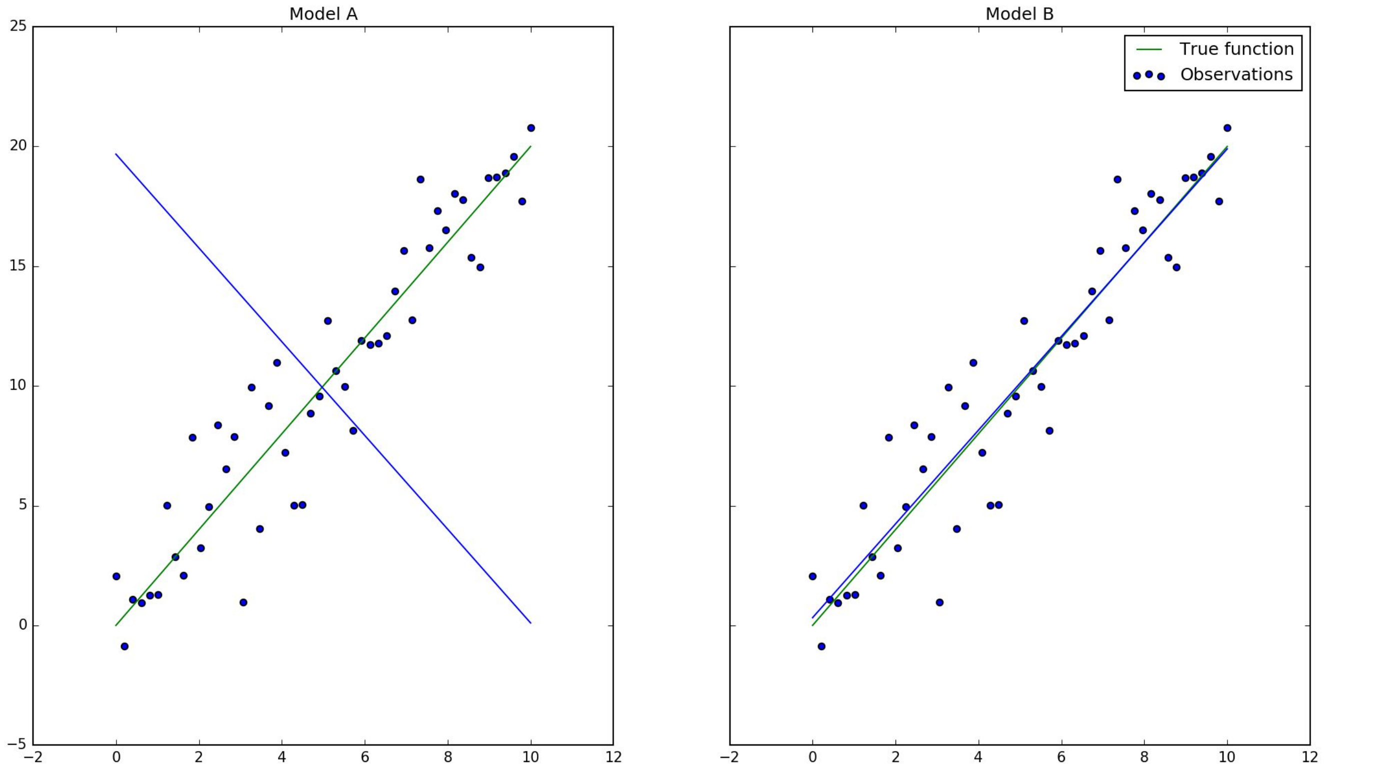
How accurate is it?

Measuring error

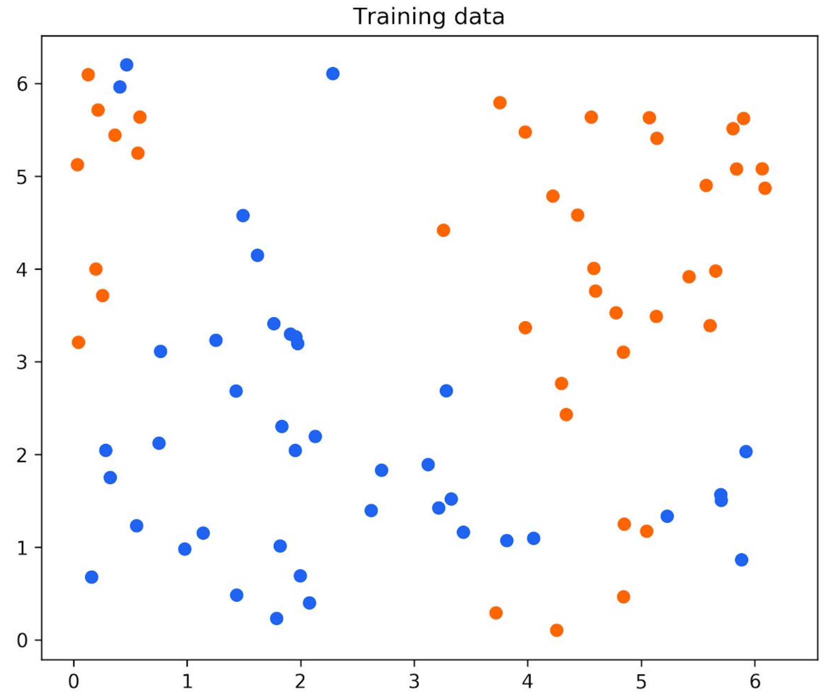


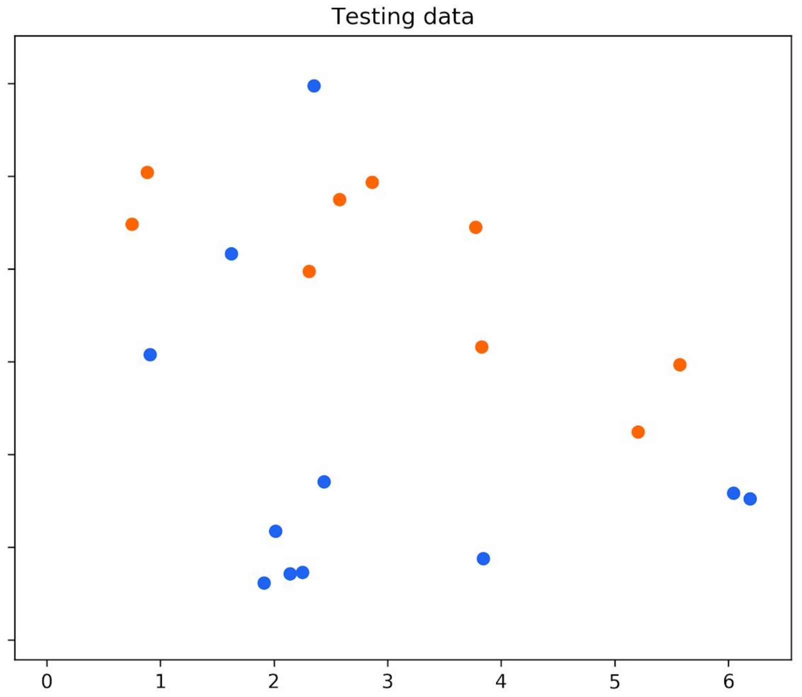


Measuring error



Randomly selected testing data allows for model evaluation



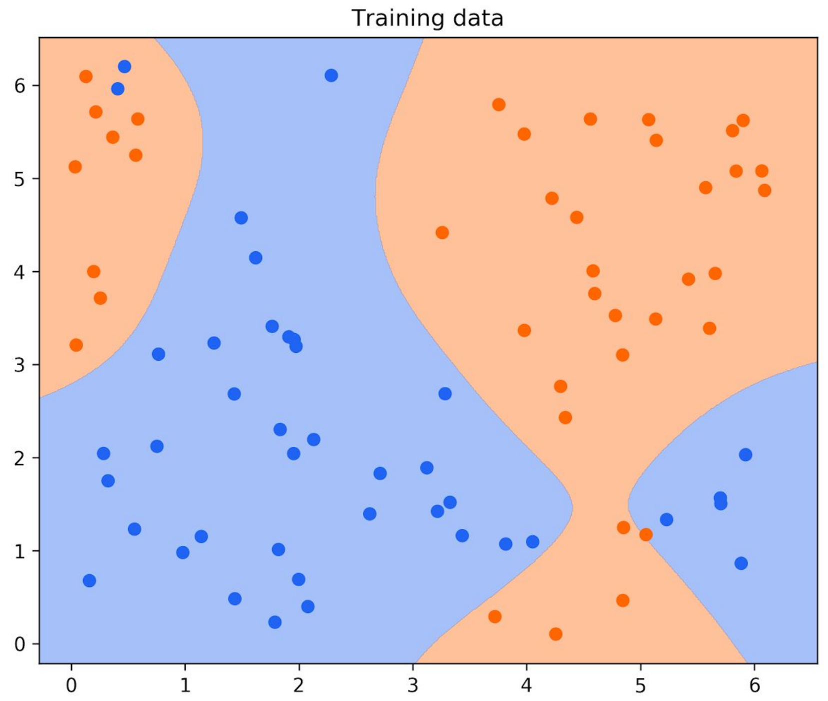


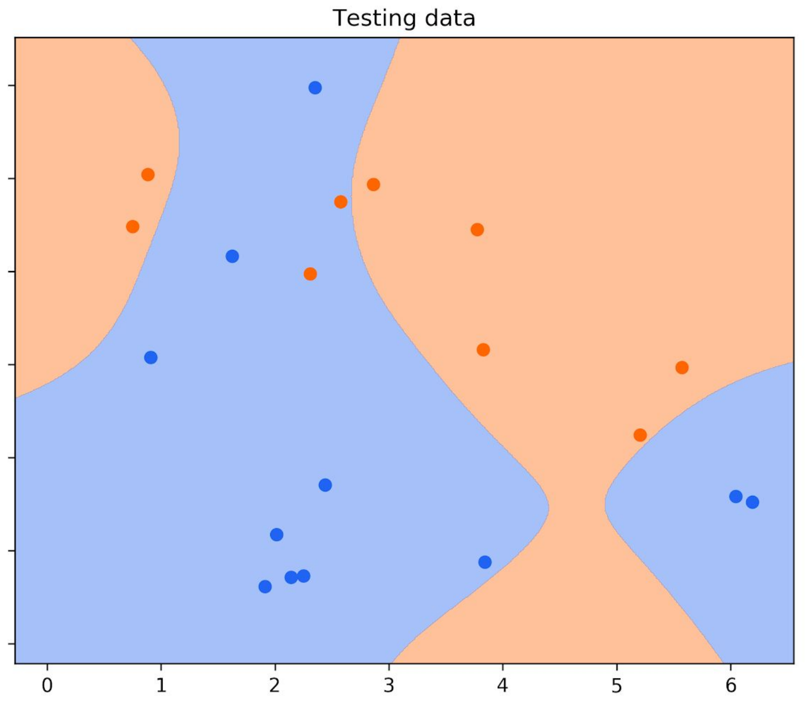
Randomly selected testing data allows for model evaluation

Training data
Testing data
0
1
2
3
4
5
6

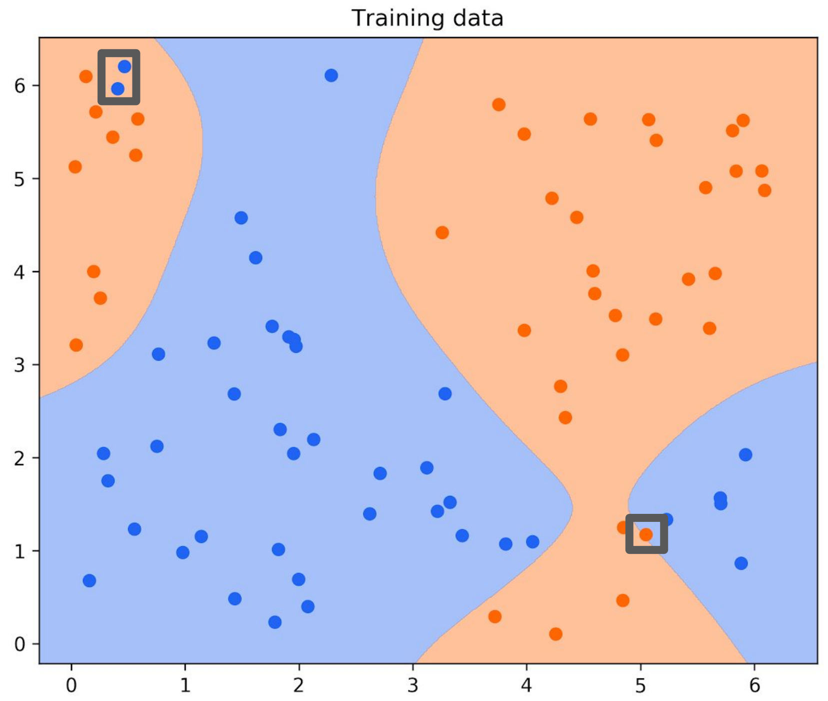
`sklearn.model_selection.train_test_split`

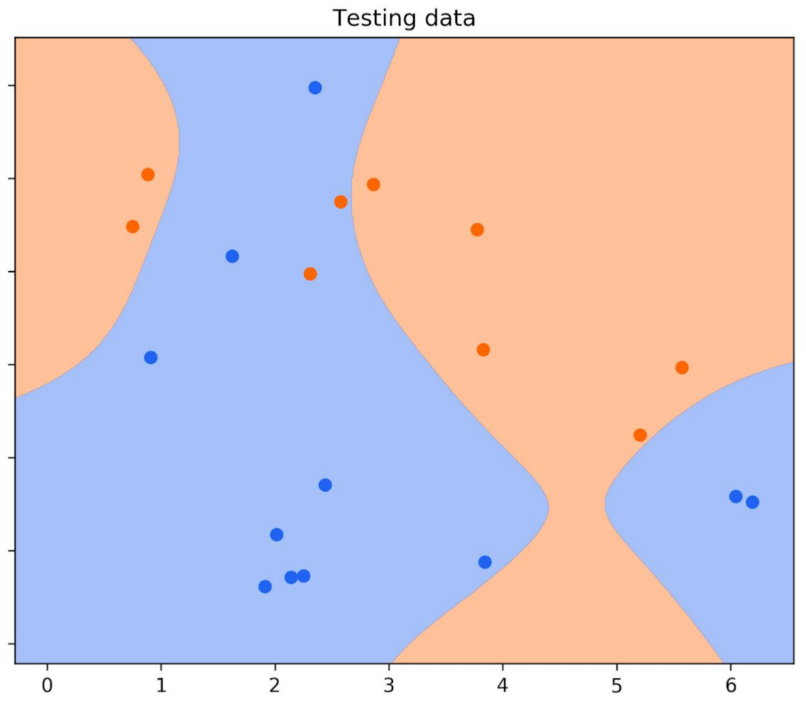
Train a model on the training data, see how it does on the testing data



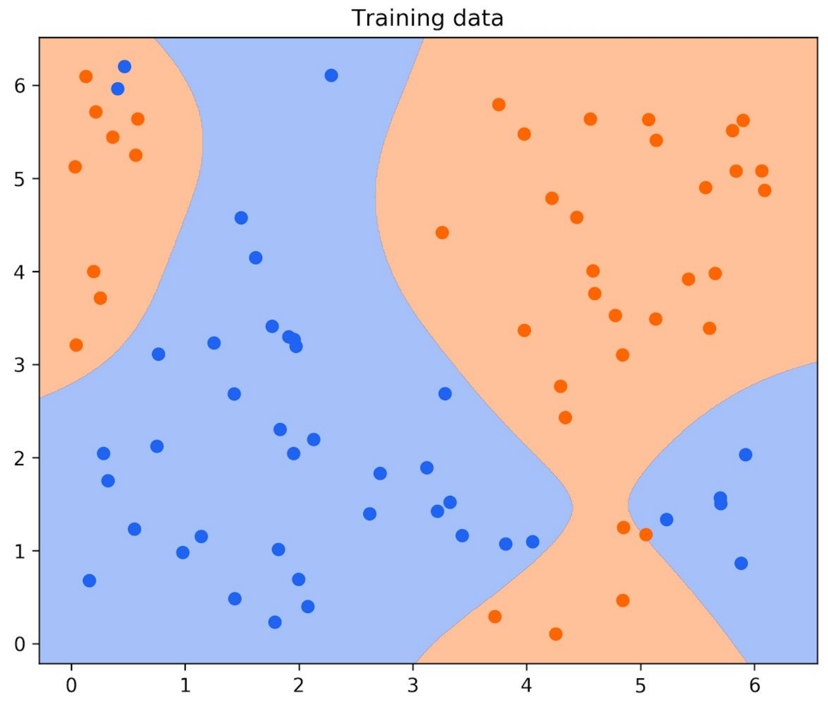


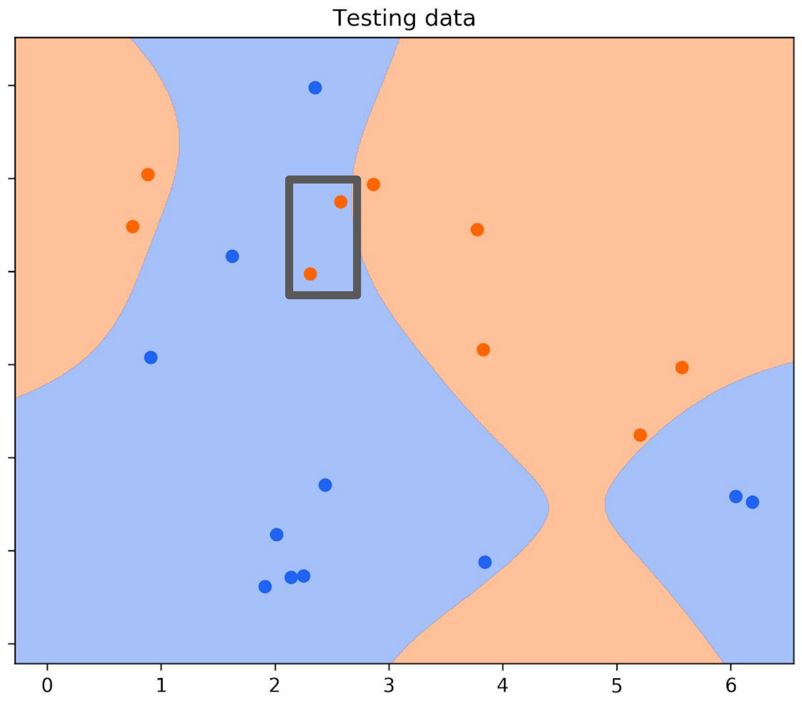
Train a model on the training data, see how it does on the testing data





Train a model on the training data, see how it does on the testing data





Measuring error

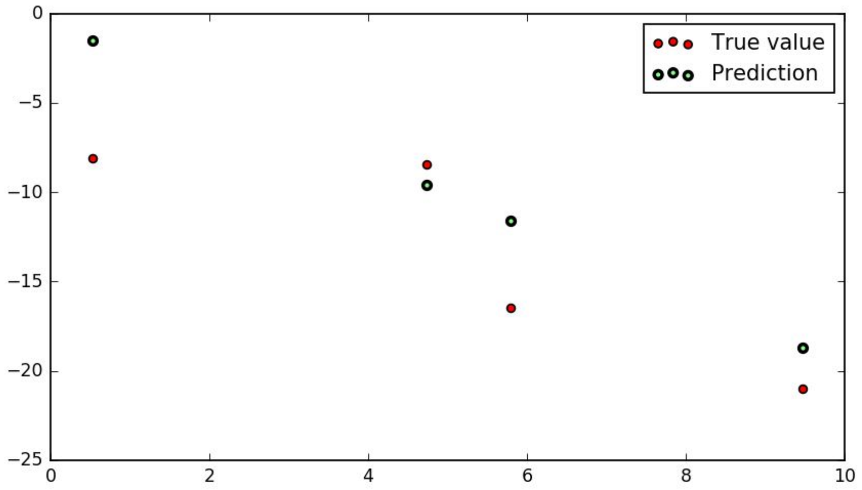
- Hold out some “testing data”
- Compare test data to prediction
- Ideally, calculate real cost

	Actually a warhead	**Actually not a warhead**
Predict a warhead	👍	Destruction of humanity
Predict not a warhead	Destruction of humanity	👍

	Actually me	**Actually not me**
Recognize fingerprint	👍	People criticize my memes for not being funny
Reject fingerprint	I get a little annoyed	👍

Defining real cost is not always possible

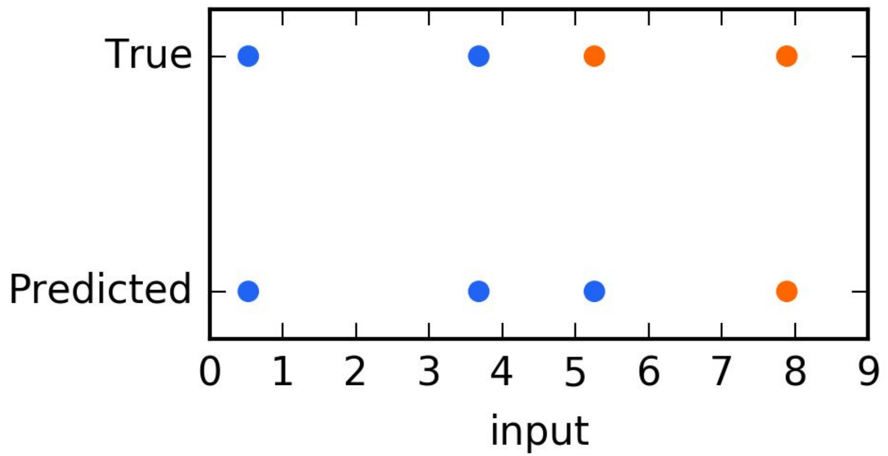
Mean squared error



Input	True	Predict	Diff	Sq. diff
0.53	-8.10	-1.51	-6.60	43.50
4.74	-8.47	-9.60	1.13	1.27
5.79	-16.45	-11.62	-4.83	23.30
9.47	-21.01	-18.70	-2.31	5.34

(43.5 + 1.27 + 23.3 + 5.34) / 4 = 18.35

Classification error



Input	True	Predict	Error?
0.53	1	1	0
3.68	1	1	0
5.26	0	1	1
7.89	0	0	0

UC0: LESSONS LEARNED

- This stuff is pretty neat
- Testing data enables evaluation

Agenda

Machine learning intro

UC0: Credit card applications

UC1: Teach a computer ASL

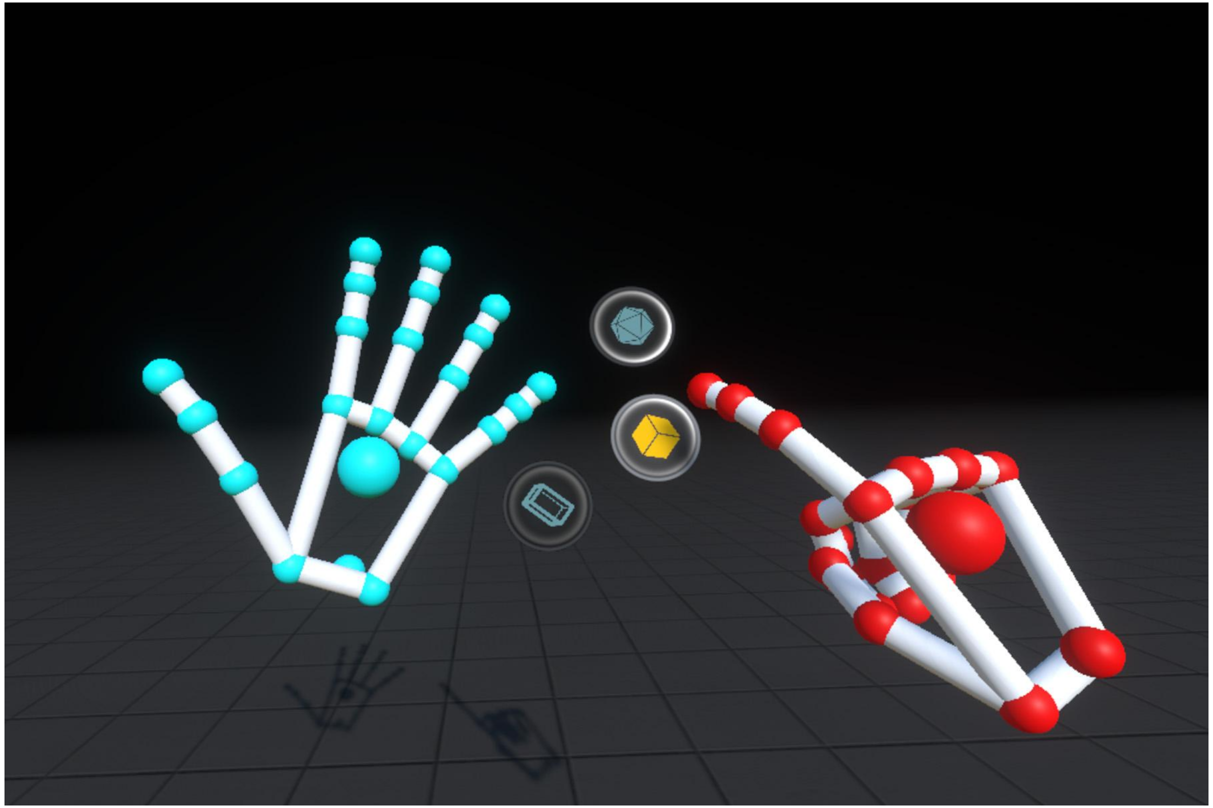
UC2: Forecasting energy load

UC3: Use ML to find your next job

➔ What’s the problem?

I don’t know sign language.

khawkins04



→ What does the data look like?

joint1_x	joint1_y	joint1_z	...	joint20_x	joint20_y	joint20_z	sign
-14.24845886	-11.23913574	47.79299927	...	39.12654877	-20.38291168	-67.37110138	a
-14.24845886	-11.23913574	47.79299927	...	39.12654877	-20.38291168	-67.37110138	a
-14.24845886	-11.23913574	47.79299927	...	39.12654877	-20.38291168	-67.37110138	a
-14.66805267	-12.86016846	47.25432587	...	39.19580078	-18.27232361	-68.12595367	a
-6.099303246	3.211929321	-21.70319366	...	1.87420845	11.96398926	-98.45552063	b
-5.093156815	2.45741272	-22.05827522	...	6.529464722	14.67698669	-97.91105652	b
32.73310089	-1.139434814	-12.70455551	...	8.51625061	18.76667786	-97.07907867	b
33.09098053	1.941070557	-11.63526344	...	10.23889732	31.46665955	-93.68971252	b
-23.29023552	-0.6312103271	-21.13870239	...	14.70001984	23.49594116	-95.80595398	b
32.82236862	-1.860855103	-12.38504791	...	10.76865768	19.6521759	-96.92489624	b

➔ **What kind of ML problem is this?**

➔ What kind of ML problem is this?

Classification

Solution

Choose a model

- Split data into training, testing
- Train a bunch of models on training data
- Evaluate them on test data
- Select the best one

→ Solution

Build an application

- → Keyboard… not so great
- → But! It’s good enough to make an educational game

Great job! Now try "x".

ssaamm/sign-language-tutor

UC1: LESSONS LEARNED

- Define the problem
- Limit scope
- Model selection
- More than the model

Agenda

Machine learning intro

UC0: Credit card applications

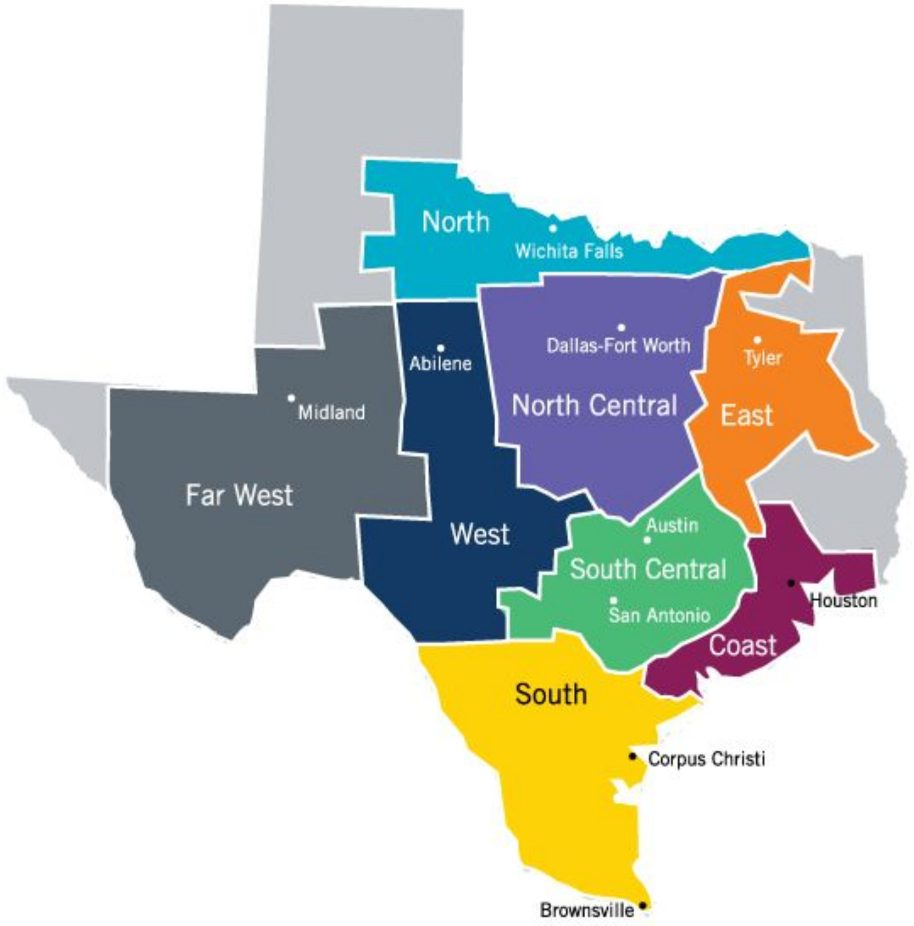
UC1: Teach a computer ASL

UC2: Forecasting energy load

UC3: Use ML to find your next job

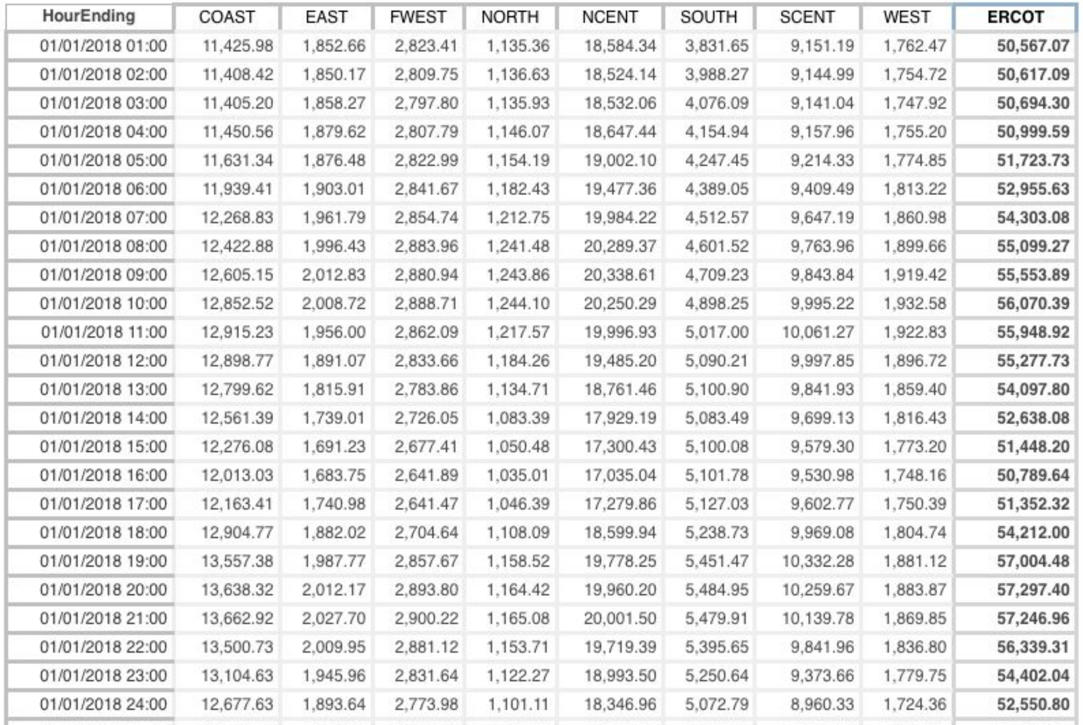
➔ What’s the problem?

Must know when to schedule energy production


North
Wichita Falls
Abilene
Dallas-Fort Worth
Tyler
Midland
North Central
East
Far West
West
Austin
South Central
San Antonio
Houston
Coast
South
Corpus Christi
Brownsville

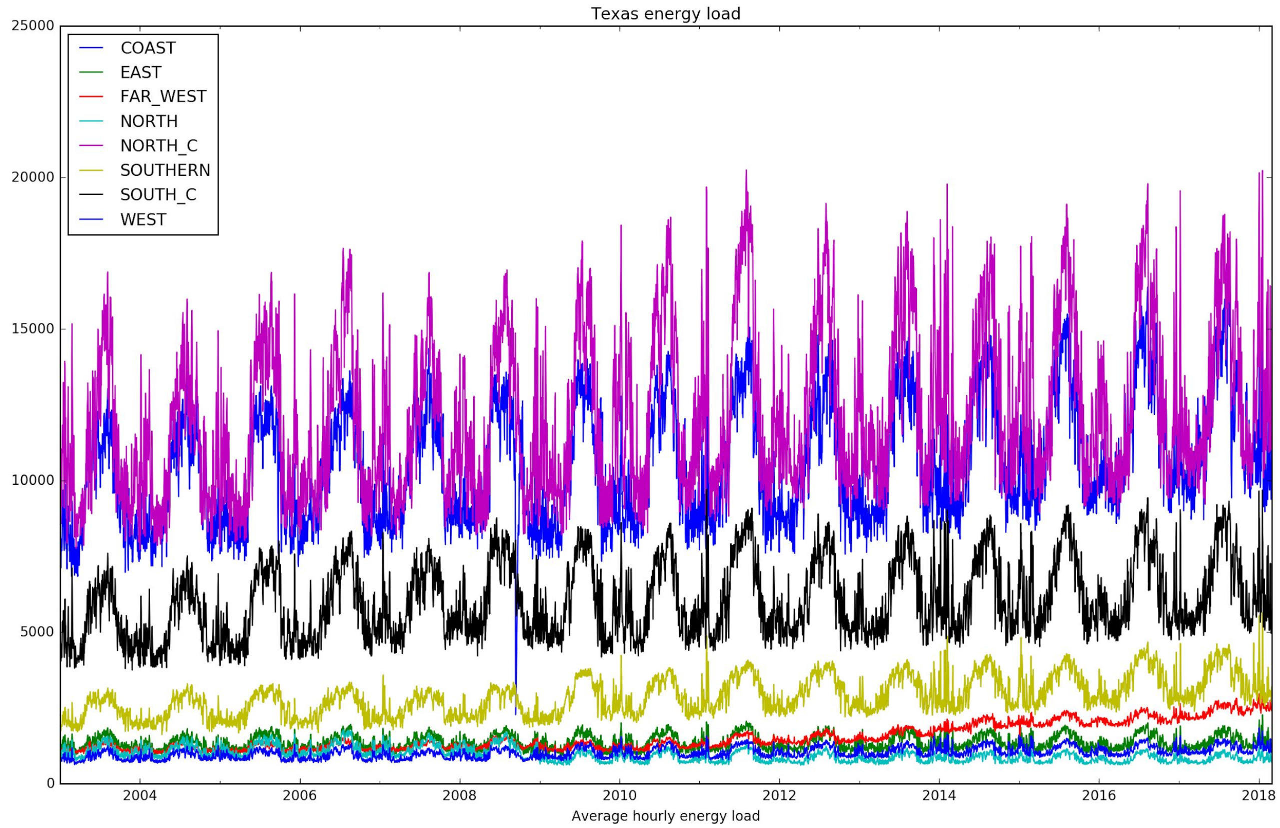
ercot

➔ What does the data look like?



HourEnding	COAST	EAST	FWEST	NORTH	NCENT	SOUTH	SCENT	WEST	ERCOT
01/01/2018 01:00	11,425.98	1,852.66	2,823.41	1,135.36	18,584.34	3,831.65	9,151.19	1,762.47	**50,567.07**
01/01/2018 02:00	11,408.42	1,850.17	2,809.75	1,136.63	18,524.14	3,988.27	9,144.99	1,754.72	**50,617.09**
01/01/2018 03:00	11,405.20	1,858.27	2,797.80	1,135.93	18,532.06	4,076.09	9,141.04	1,747.92	**50,694.30**
01/01/2018 04:00	11,450.56	1,879.62	2,807.79	1,146.07	18,647.44	4,154.94	9,157.96	1,755.20	**50,999.59**
01/01/2018 05:00	11,631.34	1,876.48	2,822.99	1,154.19	19,002.10	4,247.45	9,214.33	1,774.85	**51,723.73**
01/01/2018 06:00	11,939.41	1,903.01	2,841.67	1,182.43	19,477.36	4,389.05	9,409.49	1,813.22	**52,955.63**
01/01/2018 07:00	12,268.83	1,961.79	2,854.74	1,212.75	19,984.22	4,512.57	9,647.19	1,860.98	**54,303.08**
01/01/2018 08:00	12,422.88	1,996.43	2,883.96	1,241.48	20,289.37	4,601.52	9,763.96	1,899.66	**55,099.27**
01/01/2018 09:00	12,605.15	2,012.83	2,880.94	1,243.86	20,338.61	4,709.23	9,843.84	1,919.42	**55,553.89**
01/01/2018 10:00	12,852.52	2,008.72	2,888.71	1,244.10	20,250.29	4,898.25	9,995.22	1,932.58	**56,070.39**
01/01/2018 11:00	12,915.23	1,956.00	2,862.09	1,217.57	19,996.93	5,017.00	10,061.27	1,922.83	**55,948.92**
01/01/2018 12:00	12,898.77	1,891.07	2,833.66	1,184.26	19,485.20	5,090.21	9,997.85	1,896.72	**55,277.73**
01/01/2018 13:00	12,799.62	1,815.91	2,783.86	1,134.71	18,761.46	5,100.90	9,841.93	1,859.40	**54,097.80**
01/01/2018 14:00	12,561.39	1,739.01	2,726.05	1,083.39	17,929.19	5,083.49	9,699.13	1,816.43	**52,638.08**
01/01/2018 15:00	12,276.08	1,691.23	2,677.41	1,050.48	17,300.43	5,100.08	9,579.30	1,773.20	**51,448.20**
01/01/2018 16:00	12,013.03	1,683.75	2,641.89	1,035.01	17,035.04	5,101.78	9,530.98	1,748.16	**50,789.64**
01/01/2018 17:00	12,163.41	1,740.98	2,641.47	1,046.39	17,279.86	5,127.03	9,602.77	1,750.39	**51,352.32**
01/01/2018 18:00	12,904.77	1,882.02	2,704.64	1,108.09	18,599.94	5,238.73	9,969.08	1,804.74	**54,212.00**
01/01/2018 19:00	13,557.38	1,987.77	2,857.67	1,158.52	19,778.25	5,451.47	10,332.28	1,881.12	**57,004.48**
01/01/2018 20:00	13,638.32	2,012.17	2,893.80	1,164.42	19,960.20	5,484.95	10,259.67	1,883.87	**57,297.40**
01/01/2018 21:00	13,662.92	2,027.70	2,900.22	1,165.08	20,001.50	5,479.91	10,139.78	1,869.85	**57,246.96**
01/01/2018 22:00	13,500.73	2,009.95	2,881.12	1,153.71	19,719.39	5,395.65	9,841.96	1,836.80	**56,339.31**
01/01/2018 23:00	13,104.63	1,945.96	2,831.64	1,122.27	18,993.50	5,250.64	9,373.66	1,779.75	**54,402.04**
01/01/2018 24:00	12,677.63	1,893.64	2,773.98	1,101.11	18,346.96	5,072.79	8,960.33	1,724.36	**52,550.80**

➔ What does the data look like?

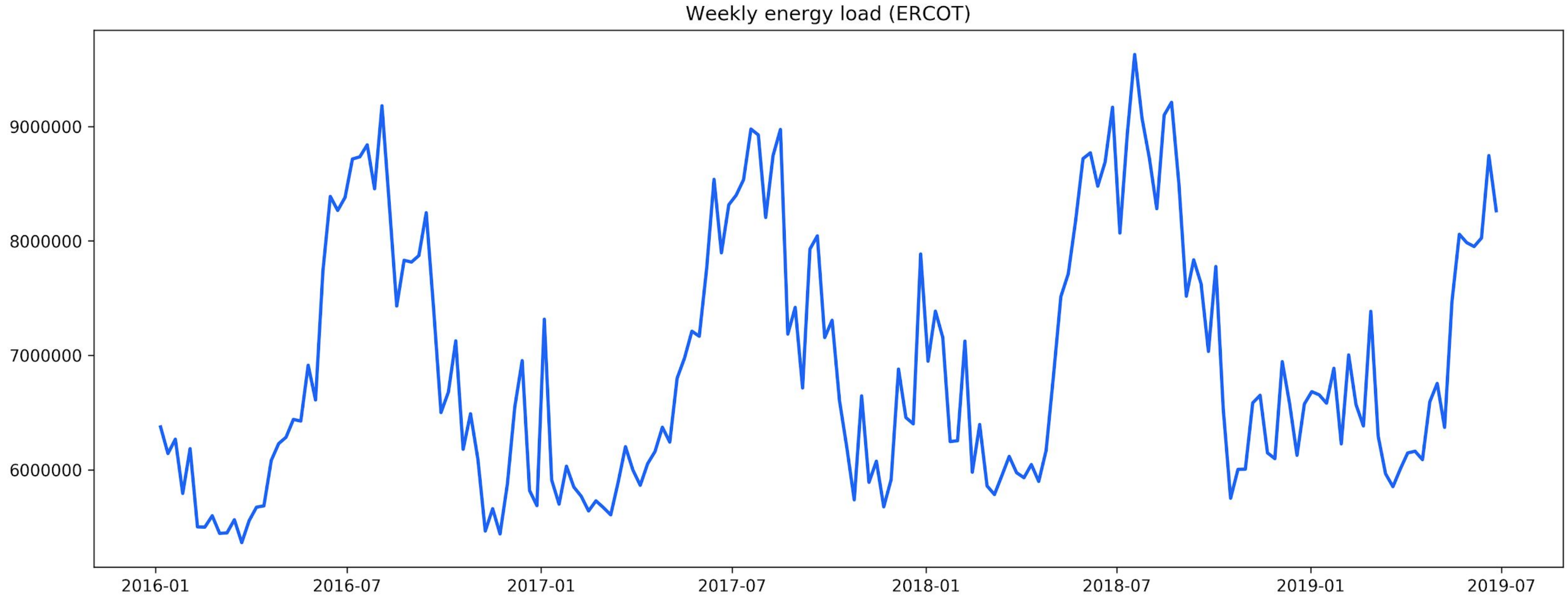


- ➔ **What kind of ML problem is this?**

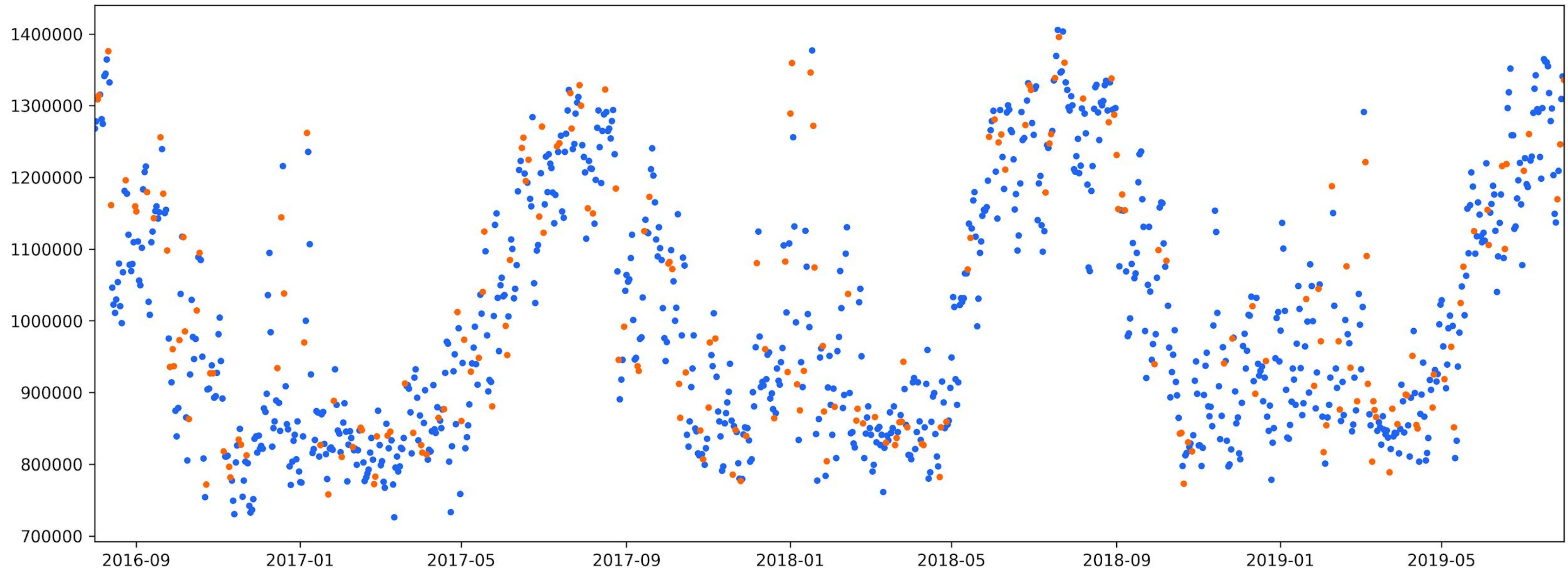
➔ What kind of ML problem is this?

Regression

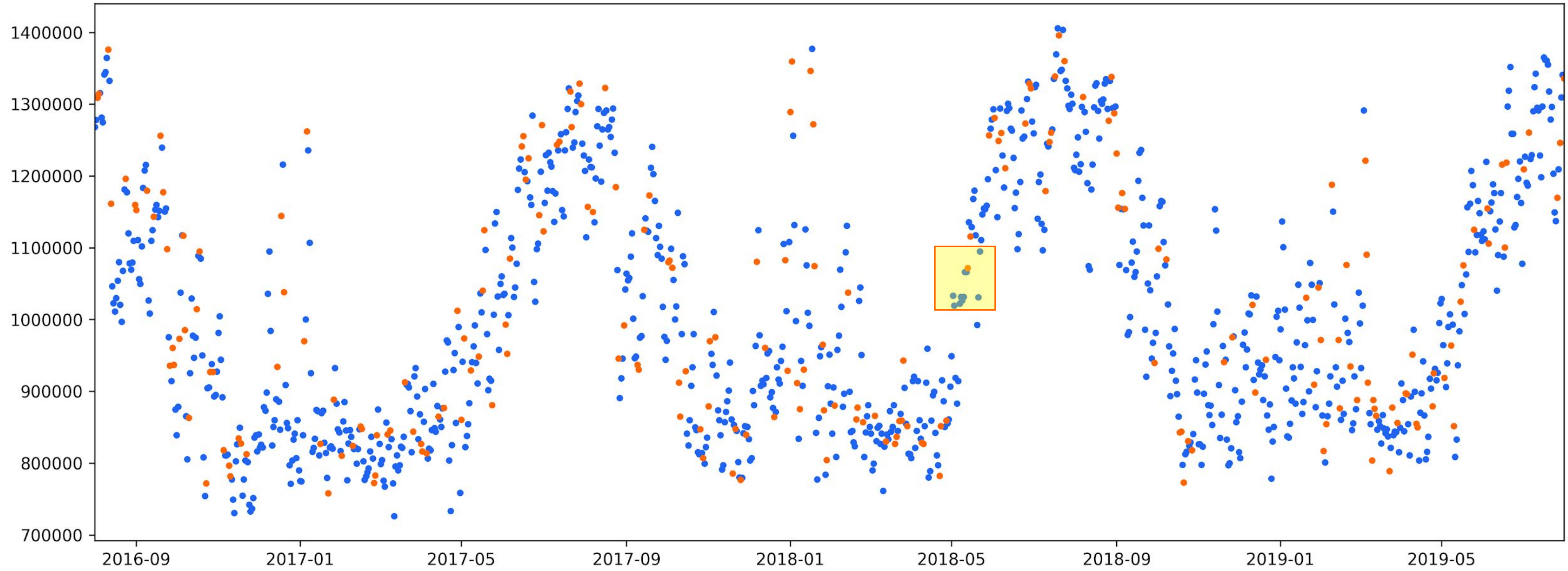
Time series data exhibits seasonality



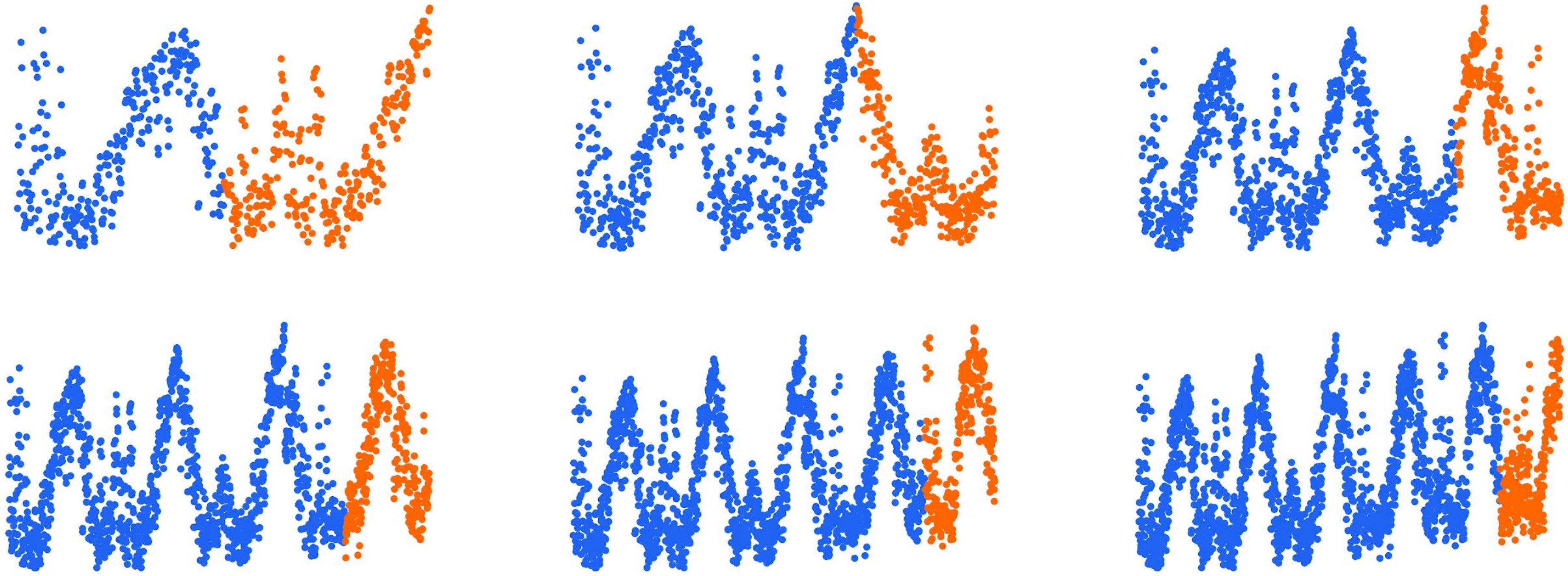
For time series data, random train/test splits leak information



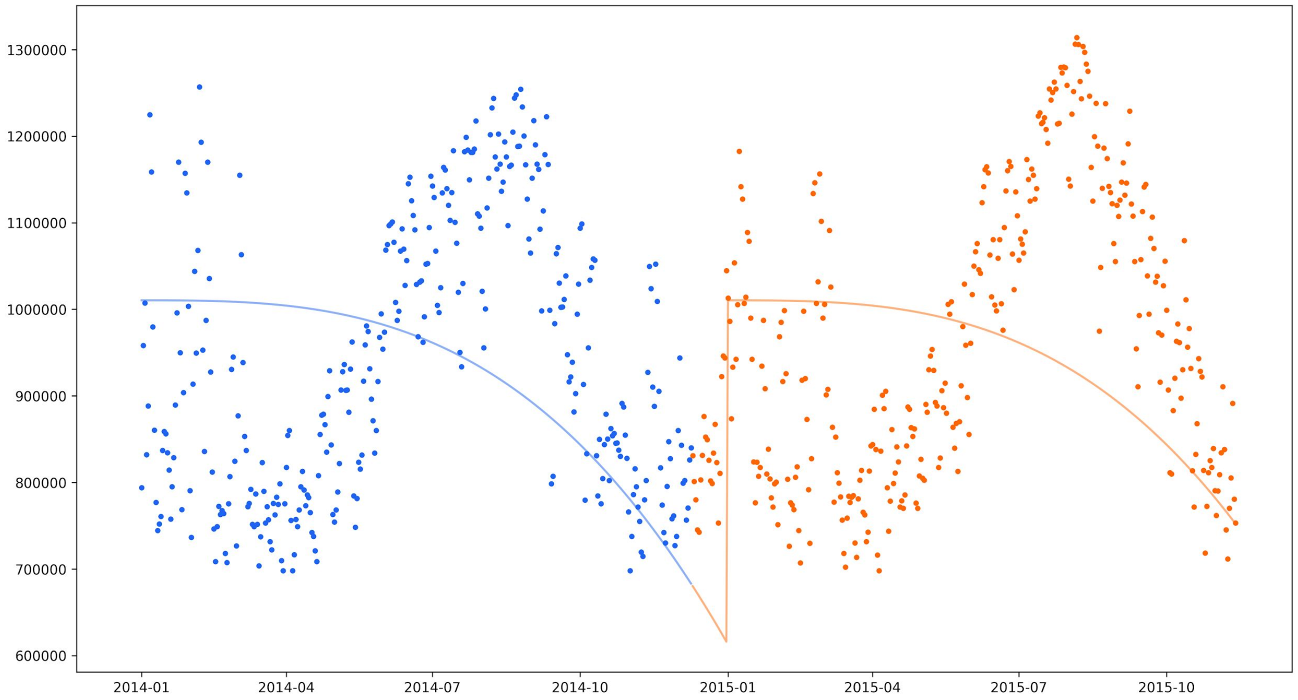
For time series data, random train/test splits leak information



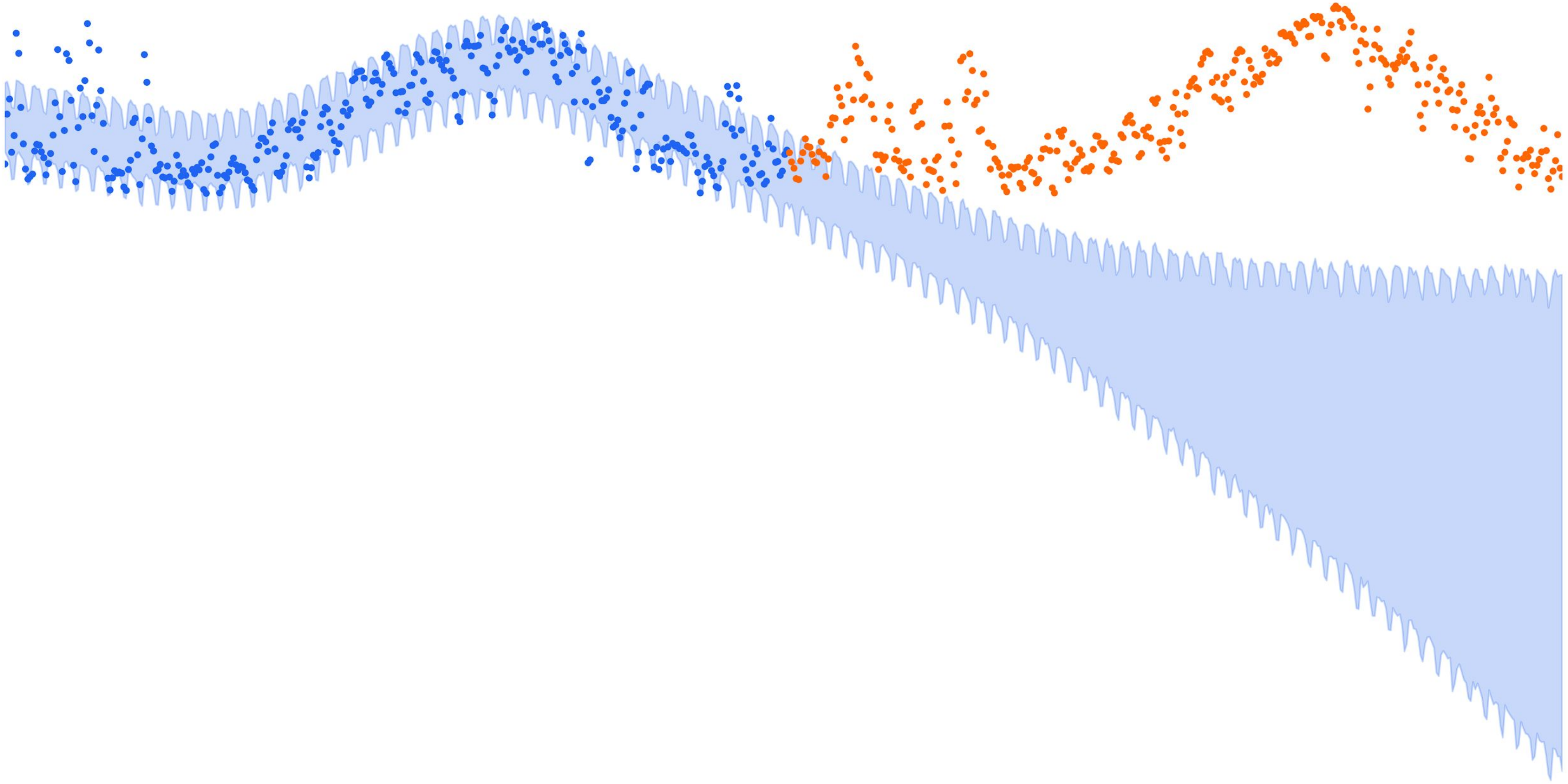
When using time series data, split based on time

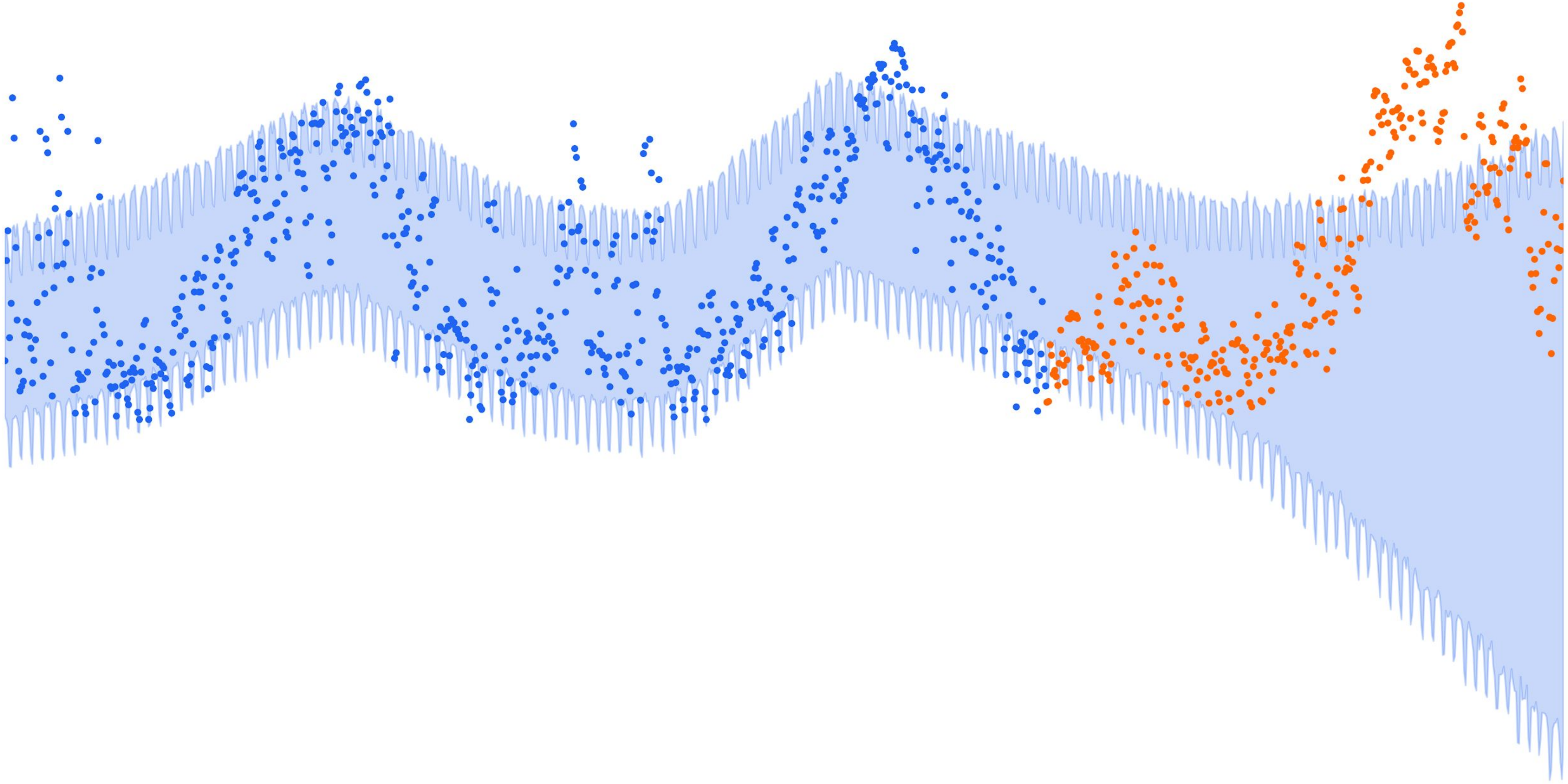


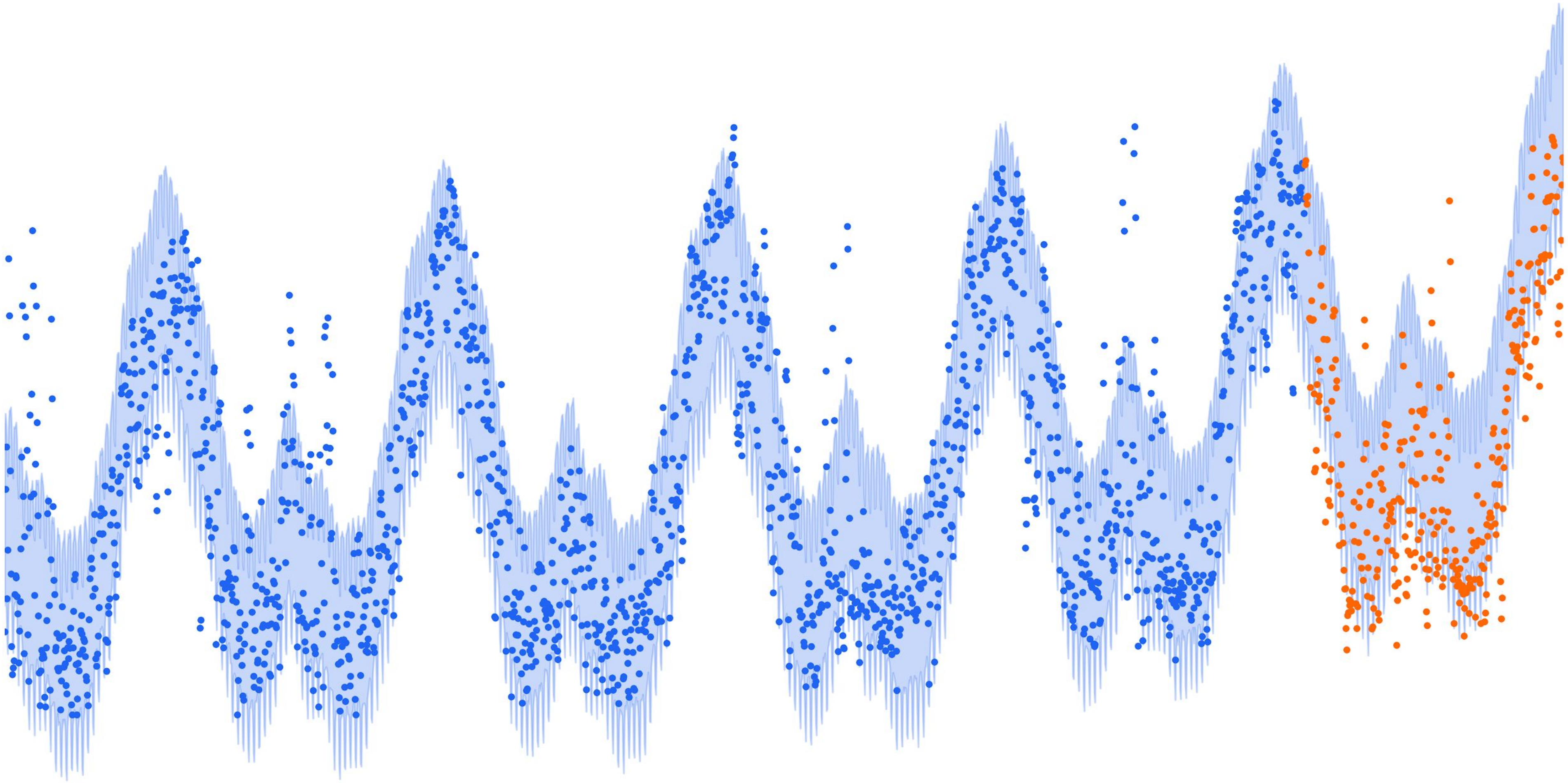
Some models learn poorly from time series data



https://facebook.github.io/prophet/







UC2: LESSONS LEARNED

- Time series data is special
- Seasonality
- Train/test split – Don’t use random!

Agenda

Machine learning intro

UC0: Credit card applications

UC1: Teach a computer ASL

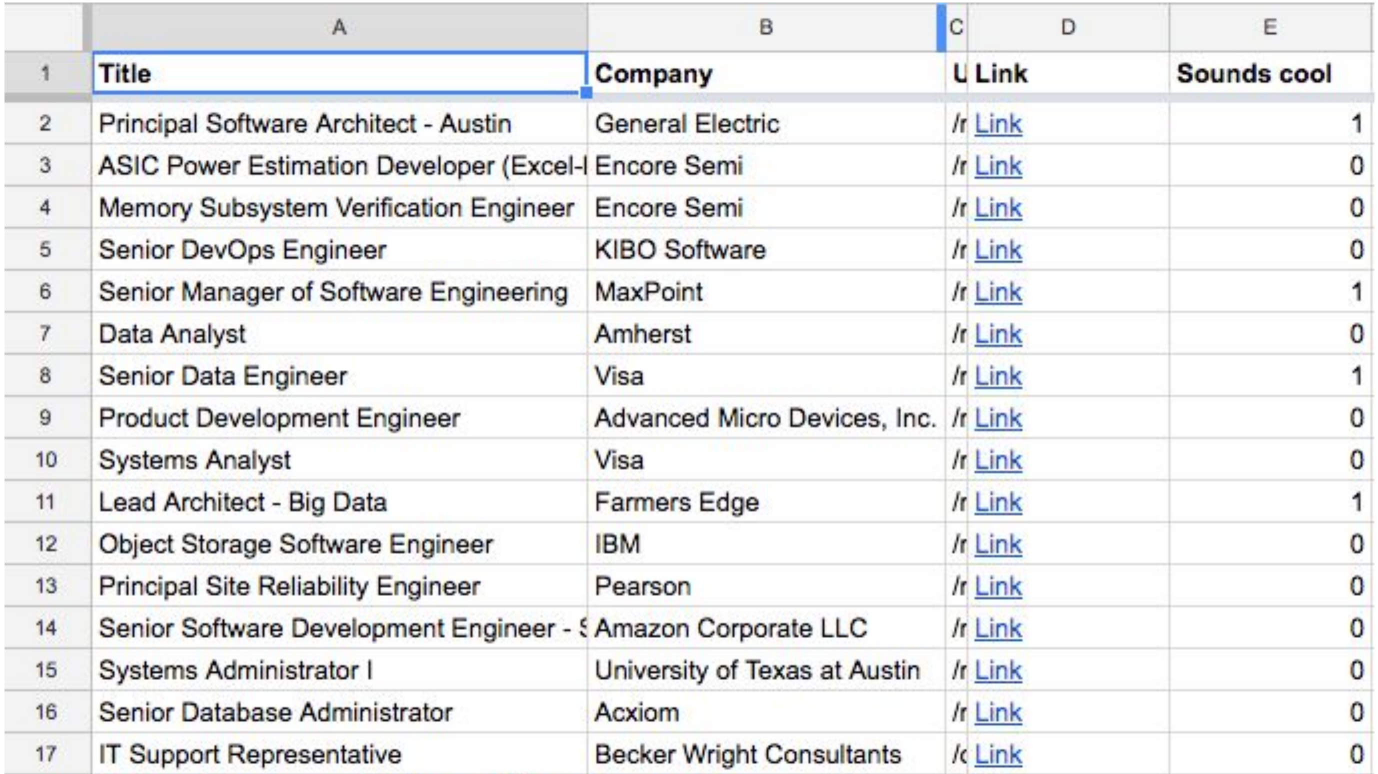
UC2: Forecasting energy load

UC3: Use ML to find your next job

➔ What’s the problem?

Passive job hunting

➔ What does the data look like?

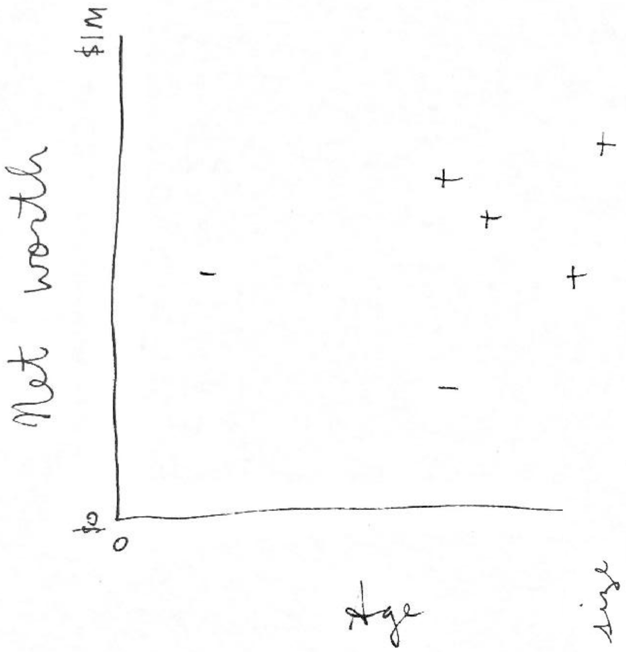


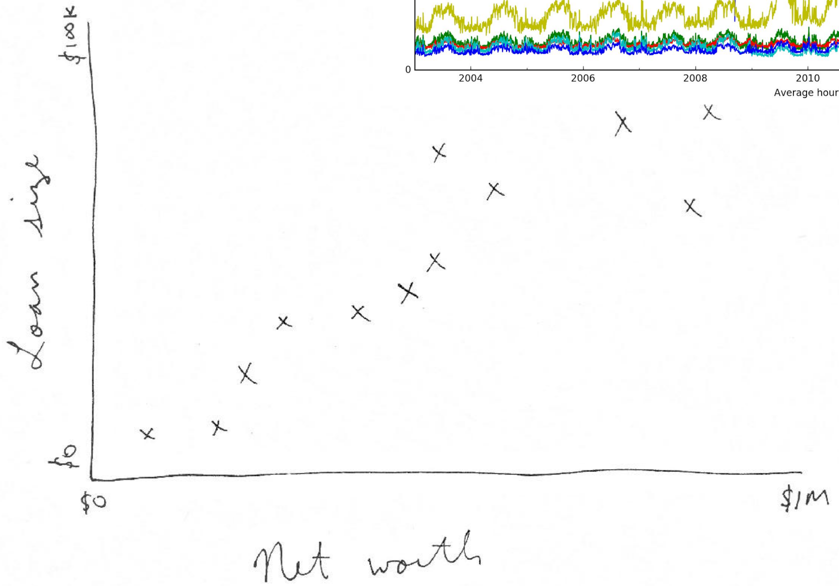
	A	B	C	D	E
1	**Title**	**Company**	**U**	**Link**	**Sounds cool**
2	Principal Software Architect - Austin	General Electric	/r	Link	1
3	ASIC Power Estimation Developer (Excel-I	Encore Semi	/r	Link	0
4	Memory Subsystem Verification Engineer	Encore Semi	/r	Link	0
5	Senior DevOps Engineer	KIBO Software	/r	Link	0
6	Senior Manager of Software Engineering	MaxPoint	/r	Link	1
7	Data Analyst	Amherst	/r	Link	0
8	Senior Data Engineer	Visa	/r	Link	1
9	Product Development Engineer	Advanced Micro Devices, Inc.	/r	Link	0
10	Systems Analyst	Visa	/r	Link	0
11	Lead Architect - Big Data	Farmers Edge	/r	Link	1
12	Object Storage Software Engineer	IBM	/r	Link	0
13	Principal Site Reliability Engineer	Pearson	/r	Link	0
14	Senior Software Development Engineer - S	Amazon Corporate LLC	/r	Link	0
15	Systems Administrator I	University of Texas at Austin	/r	Link	0
16	Senior Database Administrator	Acxiom	/r	Link	0
17	IT Support Representative	Becker Wright Consultants	/c	Link	0

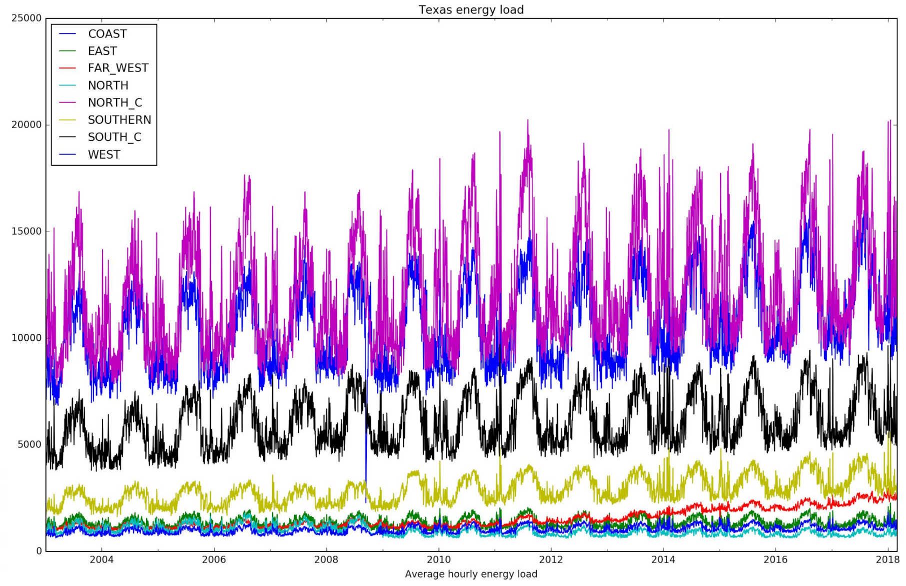
- **What kind of ML problem is this?**

➔ What kind of ML problem is this?

Classification


$1M
Net worth
$0
0
Age


$100K
Loan size
$0
$0
$1M
Net worth


Texas energy load
COAST
EAST
FAR_WEST
NORTH
NORTH_C
SOUTHERN
SOUTH_C
WEST
25000
20000
15000
10000
5000
0
2004
2006
2008
2010
2012
2014
2016
2018
Average hourly energy load

(Data Scientist, sounds_cool=True) $\longrightarrow$ **(5, 1)**

???

	Engi-neer	web	Applica-tions	sr	jr	analytics	software	data	developer
Sr. Web Applications Developer - Data Analytics	0	1	1	1	0	1	0	1	1
Jr. Software Developer	0	0	0	0	1	0	1	0	1
Sr. Data Engineer	1	0	0	1	0	0	0	1	0
Data data data data	0	0	0	0	0	0	0	4	0

(Data Scientist, sounds_cool=True)

⬇

(1, 0, 0, 1, 0, 0, 0, 0, 0, 1)

```
X = rated_jobs['title'].as_matrix()
y = rated_jobs['sounds_cool'].as_matrix()

vect = CountVectorizer()
Xp = vect.fit_transform(X).toarray()
clf = LogisticRegression().fit(Xp, y)

new_job_ratings = clf.predict(new_jobs)

# array([ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  1.])
```

```python
X = rated_jobs['title'].as_matrix()
y = rated_jobs['sounds_cool'].as_matrix()

vect = CountVectorizer()
Xp = vect.fit_transform(X).toarray()
clf = LogisticRegression().fit(Xp, y)

new_job_ratings = clf.predict(new_jobs)

# array([ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  1.])
```

```
X = rated_jobs['title'].as_matrix()
y = rated_jobs['sounds_cool'].as_matrix()

vect = CountVectorizer()
Xp = vect.fit_transform(X).toarray()
clf = LogisticRegression().fit(Xp, y)

new_job_ratings = clf.predict(new_jobs)

# array([ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  1.])
```

```
X = rated_jobs['title'].as_matrix()
y = rated_jobs['sounds_cool'].as_matrix()

vect = CountVectorizer()
Xp = vect.fit_transform(X).toarray()
clf = LogisticRegression().fit(Xp, y)

new_job_ratings = clf.predict(new_jobs)

# array([ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  1.])
```

```
X = rated_jobs['title'].as_matrix()
y = rated_jobs['sounds_cool'].as_matrix()

vect = CountVectorizer()
Xp = vect.fit_transform(X).toarray()
clf = LogisticRegression().fit(Xp, y)

new_job_ratings = clf.predict(new_jobs)

# array([ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  1.])
```

```
X = rated_jobs['title'].as_matrix()
y = rated_jobs['sounds_cool'].as_matrix()

vect = CountVectorizer()
Xp = vect.fit_transform(X).toarray()
clf = LogisticRegression().fit(Xp, y)

new_job_ratings = clf.predict(new_jobs)

# array([ 0.,  0.,  0.,  1.,  0.,  0.,  0.,  1.])
```

Accuracy

Classification error: 0.197

(awesome!)

Accuracy

Classification error: 0.197

(awesome!)

But wait…

Accuracy

Classification error: 0.197

(awesome!)

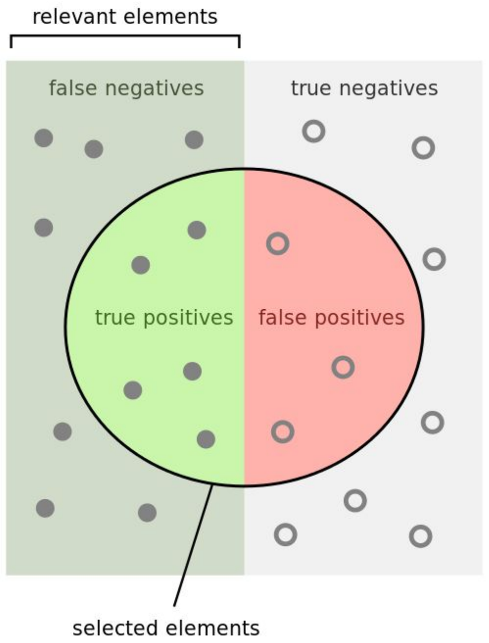
But wait…

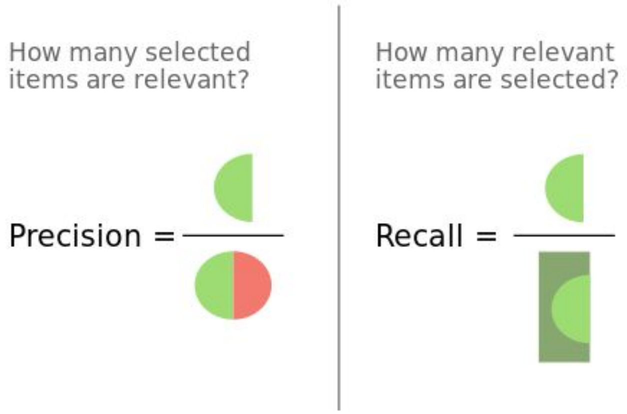
Base rate == 0.197

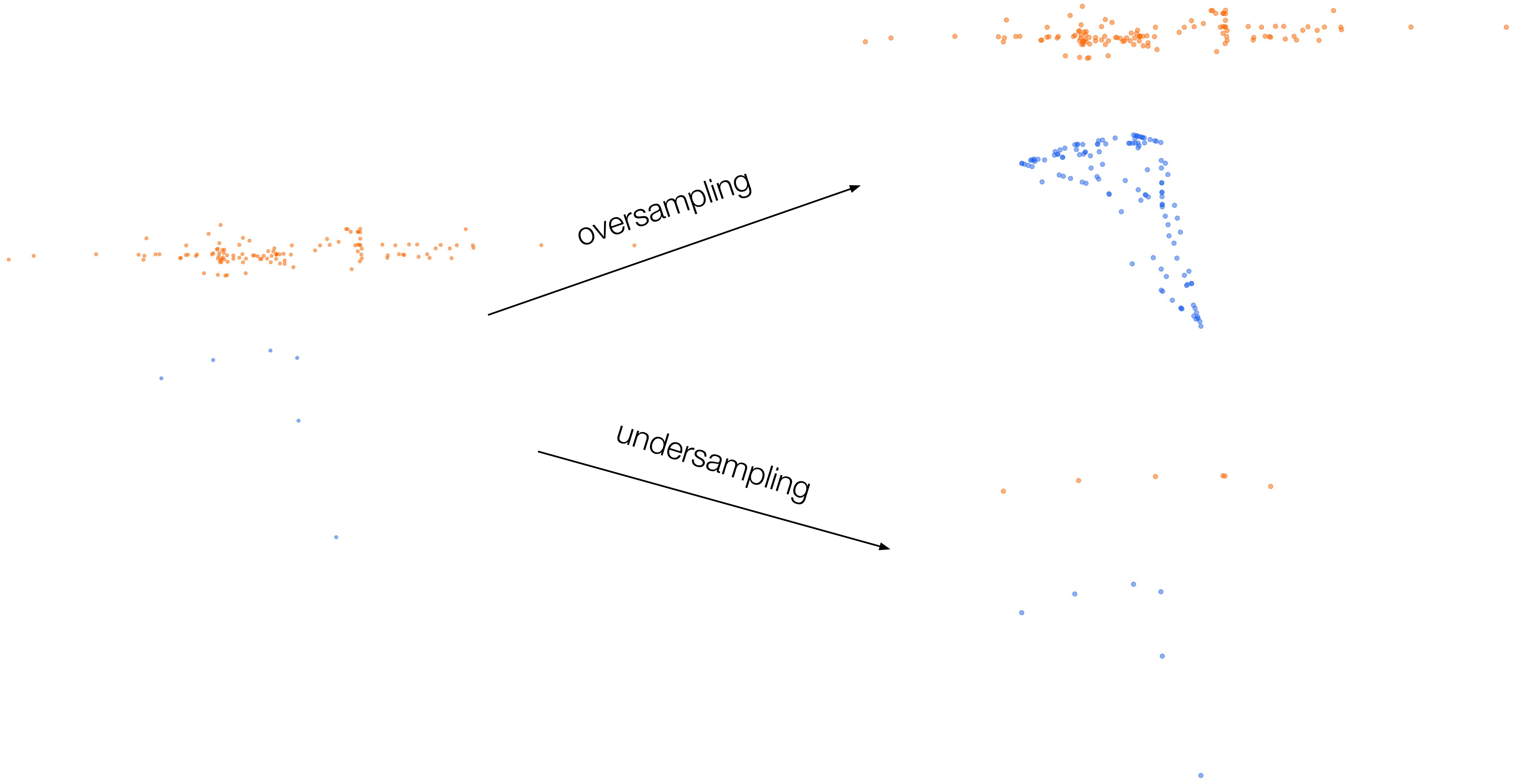
	Actual 0	Actual 1	Total
Predicted 0	400	100	500
Predicted 1	0	0	0
Total	400	100	

Imbalance

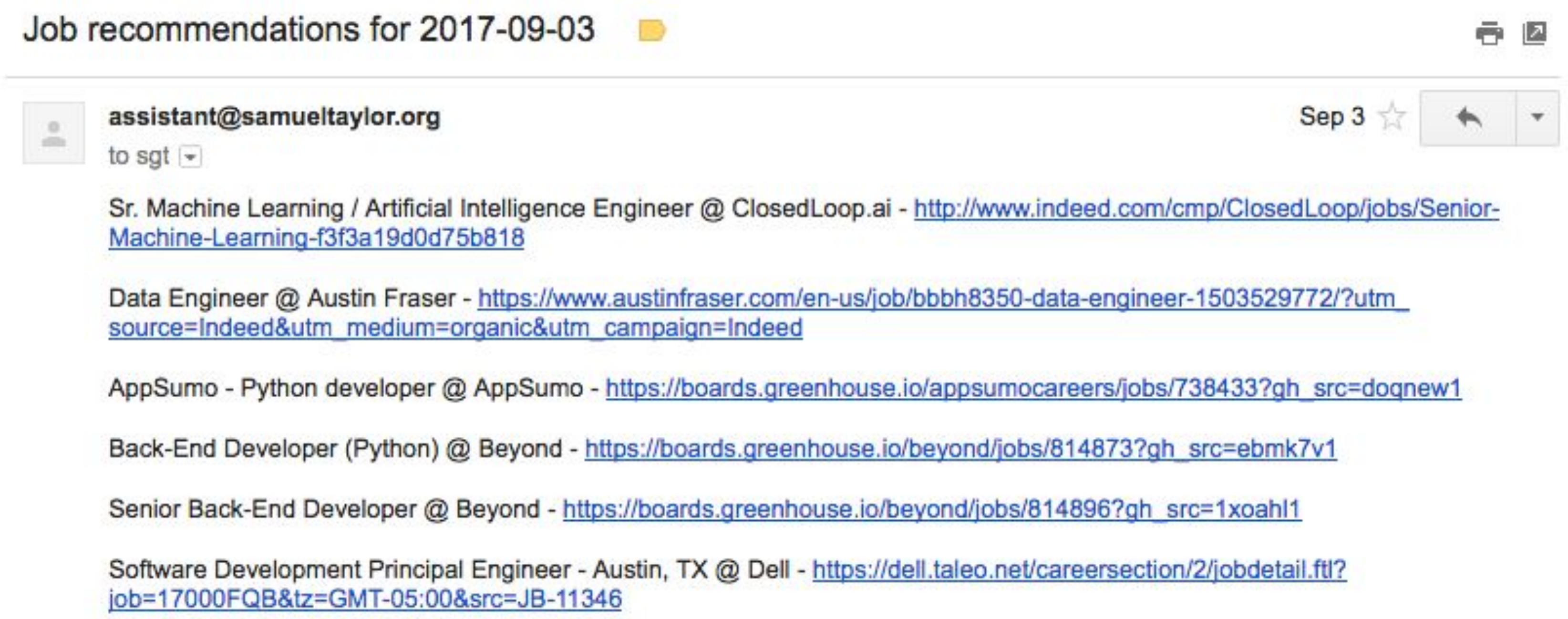
Better error metrics




How many selected items are relevant?
Precision =
How many relevant items are selected?
Recall =


oversampling
undersampling

End result



Job recommendations for 2017-09-03

assistant@samueltaylor.org
to sgt

Sep 3

Sr. Machine Learning / Artificial Intelligence Engineer @ ClosedLoop.ai - http://www.indeed.com/cmp/ClosedLoop/jobs/Senior-Machine-Learning-f3f3a19d0d75b818

Data Engineer @ Austin Fraser - https://www.austinfraser.com/en-us/job/bbbh8350-data-engineer-1503529772/?utm_source=Indeed&utm_medium=organic&utm_campaign=Indeed

AppSumo - Python developer @ AppSumo - https://boards.greenhouse.io/appsumocareers/jobs/738433?gh_src=doqnew1

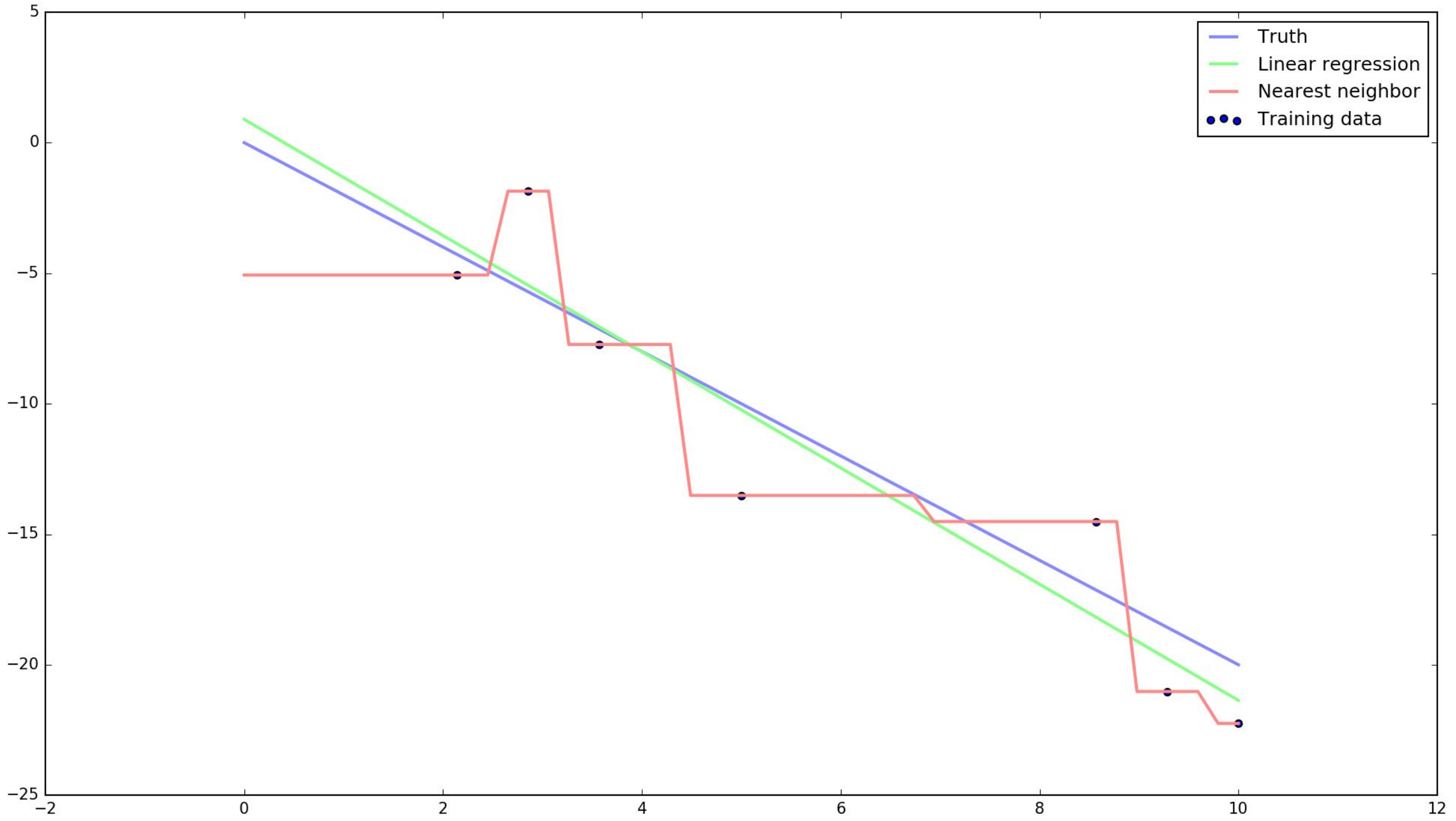
Back-End Developer (Python) @ Beyond - https://boards.greenhouse.io/beyond/jobs/814873?gh_src=ebmk7v1

Senior Back-End Developer @ Beyond - https://boards.greenhouse.io/beyond/jobs/814896?gh_src=1xoahl1

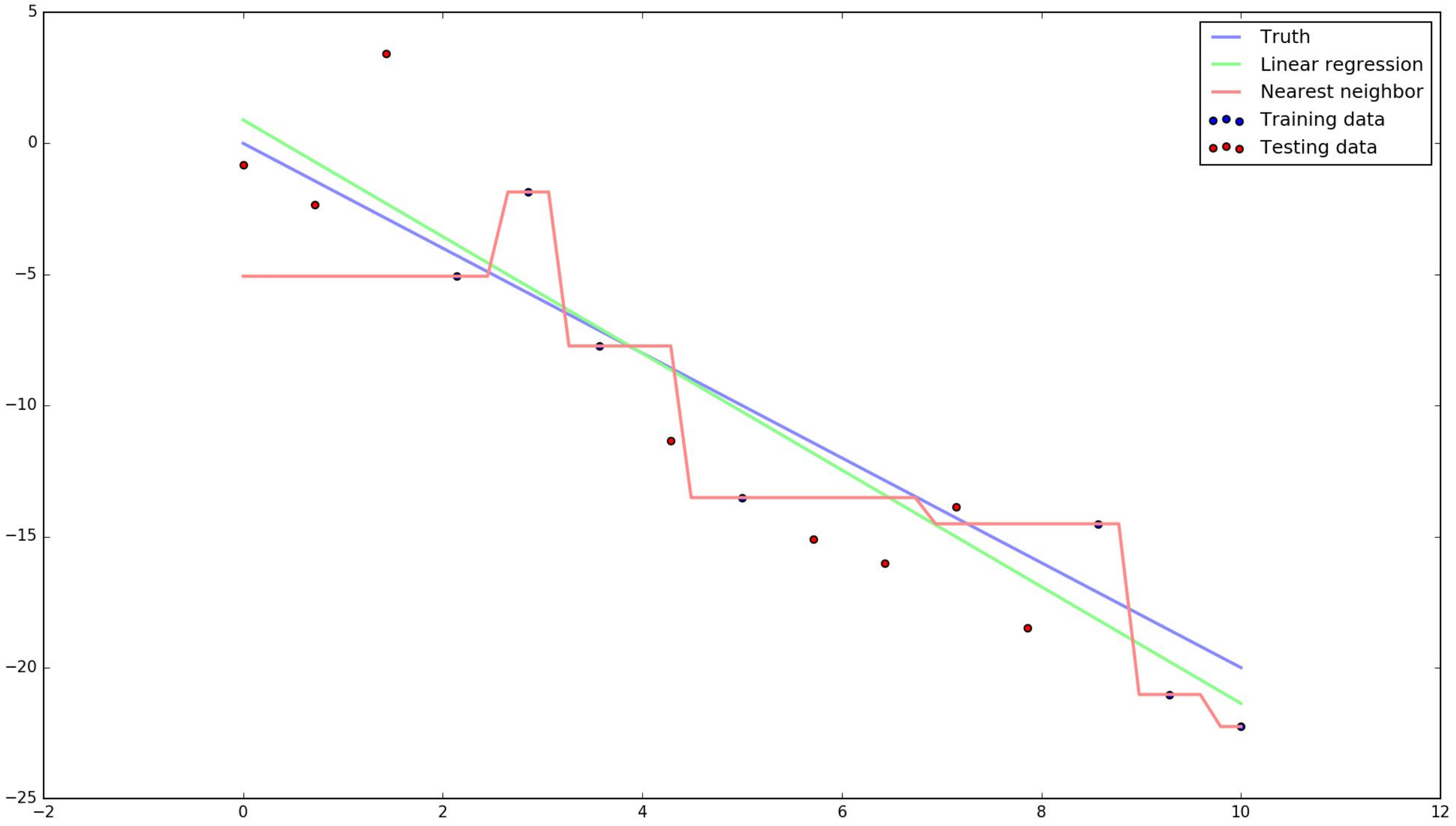
Software Development Principal Engineer - Austin, TX @ Dell - https://dell.taleo.net/careersection/2/jobdetail.ftl?job=17000FQB&tz=GMT-05:00&src=JB-11346

UC3: LESSONS LEARNED

- Understand the base rate
- Simple doesn’t mean ineffective
- Approximation-generalization tradeoff



5
0
−5
−10
−15
−20
−25
−2
0
2
4
6
8
10
12
Truth
Linear regression
Nearest neighbor
Training data


Truth
Linear regression
Nearest neighbor
Training data
Testing data
5
0
−5
−10
−15
−20
−25
−2
0
2
4
6
8
10
12

UC3: LESSONS LEARNED

- Understand the base rate
- Simple doesn't mean ineffective
- Approximation / generalization tradeoff

also, it's easier

Deep breath, everyone

Takeaways

→ **Supervised learning**

Using past examples to predict a continuous or discrete value

Takeaways

➔ **Supervised learning**

Using past examples to predict a continuous or discrete value

➔ **Measuring performance**

Split data into training and testing subsets

Takeaways

→ **Supervised learning**

Using past examples to predict a continuous or discrete value

→ **Measuring performance**

Split data into training and testing subsets

→ **K.I.S.S.**

Try the simplest thing that could possibly work

Takeaways

- ➔ **Supervised learning**

  Using past examples to predict a continuous or discrete value

- ➔ **Measuring performance**

  Split data into training and testing subsets

- ➔ **K.I.S.S.**

  Try the simplest thing that could possibly work

- ➔ **Test and iterate**

indeed

slides: **go.indeed.com/SamuelATO**
twitter: **@SamuelDataT**
email: **sgt@samueltaylor.org**