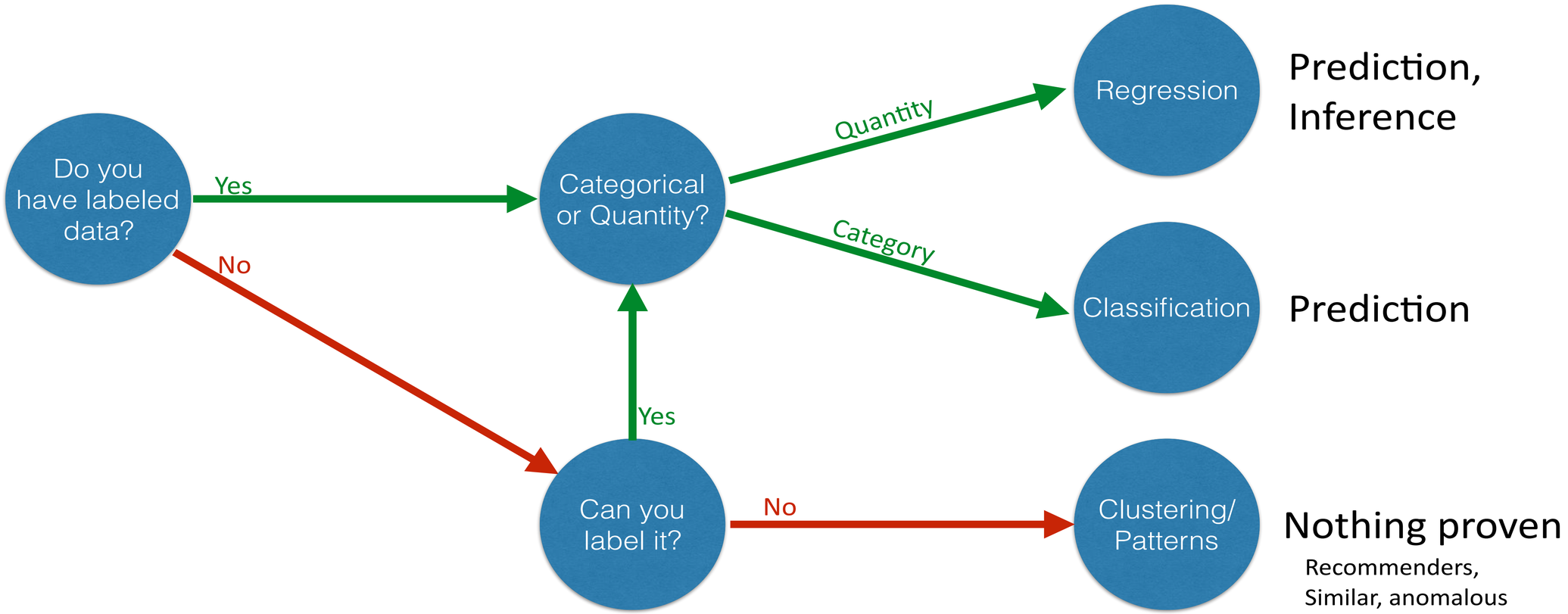

GTK CYBER
GET ON THE FAST TRACK

Data Science for Security Professionals - Day 3

All about Machine Learning


Do you have labeled data?
Yes
No
Categorical or Quantity?
Quantity
Category
Regression
Classification
Yes
Can you label it?
No
Clustering/ Patterns
Prediction, Inference
Prediction
Nothing proven
Recommenders,
Similar, anomalous

Machine Learning: Feature Engineering

Use Case: From URL strings to “Features”

URL Definition

https://www.google.com/search?q=URL&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjcl6ut-IDUAhVEPCYKHdJGDsYQ_AUIDCgD&biw=1215&bih=652

https://	protocol
www	subdomain
google.com	zone apex
google	domain
.com	top-level-domain (tld)
/search?q=URL…	path

DNS 101

- Domain Name Service (DNS) resolves domain names to IP addresses (like a phone book)
- Domain Registrars: authority that signs unique domain names
- State of Authority (SOA): Contains for example name of server for zone, administrator of zone, default time-to-live (ttl = time a DNS record is cached), seconds of secondary name server should wait before checking for updates
- Root Zone controlled by Internet Assigned Numbers Authority (IANA)
- Name Servers (NS Records): used by tld servers to direct traffic to DNS server (which contains authoritative DNS records)
- A records (part of DNS record): “A” stands for IP Address
- CNAME (part of DNS record): resolves one domain name to another
- Autonomous System (AS) and Border gateway Protocol (BGP) info

Python libraries: **`python-whois, dnspython, tldextract, ipaddress`**

Representation of URL Knowledge

- Come up with a representation/set of knowledge that has enough complexity to accurately describe the problem for the computer
- Knowledge here does not mean hard-coded knowledge or formal set of rules
- The computer rather uses the knowledge we provide to extract patterns and acquire own knowledge
- We should provide knowledge about reality that has high variance about the problem it describes (e.g. a feature that is high when it rains and low when it’s sunshine)

What makes them different?

URL BlackList

amazon-sicherheit.kunden-ueberpruefung.xyz

eclipsehotels.com/language/en-GB/eng.exe

bohicacapital.com/page

summerweb.net

ad.getfond.info

vdula.czystykod.pl/rxdjna2.html

svision-online.de/mgfi/administrator/components/com_babackup/classes/fx29id1.txt

URL WhiteList

gurufocus.com/stock/PNC

dvdtalk.ru/review

333cn.com/zx/zhxw.html

made-in-china.com/special/led-lighting

google.com/u/0/112261544981697332354/posts

youtube.com/watch?v=Qp8MQ4shN6U

unesco.org/themes/education-sustainable-developm

thisisthefirst.com/page/5

Malicious URL Detection Features (Literature)

1. **BlackList Features**: BlackLists suffer from a high false negative rate, but can still be useful as machine learning feature.
2. **Lexical Features**: Capture the property that malicious URLs tend to "look different" from benign URLs. Contextual information such as the length of the URL, number of digits, lengths of different parts, entropy of domain name.
3. **Host-based Features**: Properties of web site host. "Where" the site is hosted, "who" owns it and "how" it is managed. API queries are needed (WHOIS, DNS records). Examples: Date of registration, the geolocations, autonomous system (AS) number, connection speed or time-to-live (TTL).
4. **Content-based Features**: Less commonly used feature as it requires execution of web-page. Can be not only be not safe, but also increases the computational cost. Examples: HTML or JavaScript based.

Sahoo et al. 2017: Malicious URL Detection using Machine Learning: A Survey

Preparation In Class Exercise
ML Feature Engineering

Lexical Features

1. Length of URL
2. Length of domain
3. Count of digits
4. Entropy of domain
5. Position (or index) of the first digit
6. **Bag-of-words** for tld, domain and path parts of the URL

Host-based Features

1. Time delta between today's date and creation date
2. Check if it is an IP address

Bag-of-Words

- Bag-of-words model: (Frequency of) occurrence of each word is used as a feature
- Sklearn's **`CountVectorizer`**: Convert a collection of text documents to a matrix of token counts

Simple Example: Imposing a vocabulary of `top_tlds=['.com', '.de', '.uk']`

```
CountVectorizer_tlds = CountVectorizer(analyzer='word', vocabulary=top_tlds)
CountVectorizer_tlds = CountVectorizer_tlds.fit(tlds)
matrix_tlds = CountVectorizer_tlds.transform(tlds)
```

Bag-of-words model fitting

URL string
…google.ru…
…facebook.com…
…google.de…

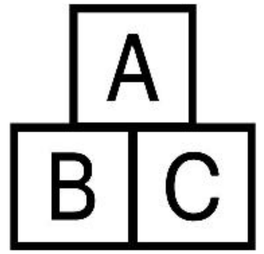
'.com'	'.de'	'.uk'
0	0	0
1	0	0
0	1	0

http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Preprocessing - an Art Work!

- Imputing missing values
- Scaling/Normalization
- One-Hot Encoding (Encoding categorical features)
- Embedding (e.g. word2vec)
- Binarizing (e.g. needed for Deep Learning multi-class target vector encoding)
- Encoding strings as int
- Dimensionality Reduction (e.g. PCA)
- Augmentation (e.g. tild/zoom images)
- Feature selection based on classifier
- Variance threshold

Data Types



Primary Python libraries: **`pandas, sklearn, scipy`**

http://scikit-learn.org/stable/modules/preprocessing.html

Imputing Missing Values

```
# using the most_frequent value
df.src_bytes = df.src_bytes.fillna
(df.src_bytes.value_counts().index[0])

# using the mean value
df.dst_bytes = df.dst_bytes.fillna(df.dst_bytes.mean())
```

One Hot Encoding

Color
Red
Red
Blue
Green
Yellow
Red

One Hot Encoding

4 Categories

4 Columns with 1 when Category is True and delete original column!

Color
Red
Red
Blue
Green
Yellow
Red

Color_Red	Color_Blue	Color_Yellow	Color_Green
1	0	0	0
1	0	0	0
0	1	0	0
0	0	0	1
0	0	1	0
1	0	0	0

One Hot Encoding

```
colors = ['Red', 'Red', 'Blue', 'Green', 'Yellow', ‘Red']
series_data = pd.Series( colors )
pd.get_dummies( series_data )

# df scenario
df=pd.get_dummies(df, prefix=None, prefix_sep='_',
dummy_na=False, columns=['protocol_type','flag'],
sparse=False)
```

Encoding strings as int

```
PROBE = ['portsweep.', 'satan.', 'nmap.', 'ipsweep.']
df = df.replace(to_replace = PROBE, value=1)
```

In Class Exercise

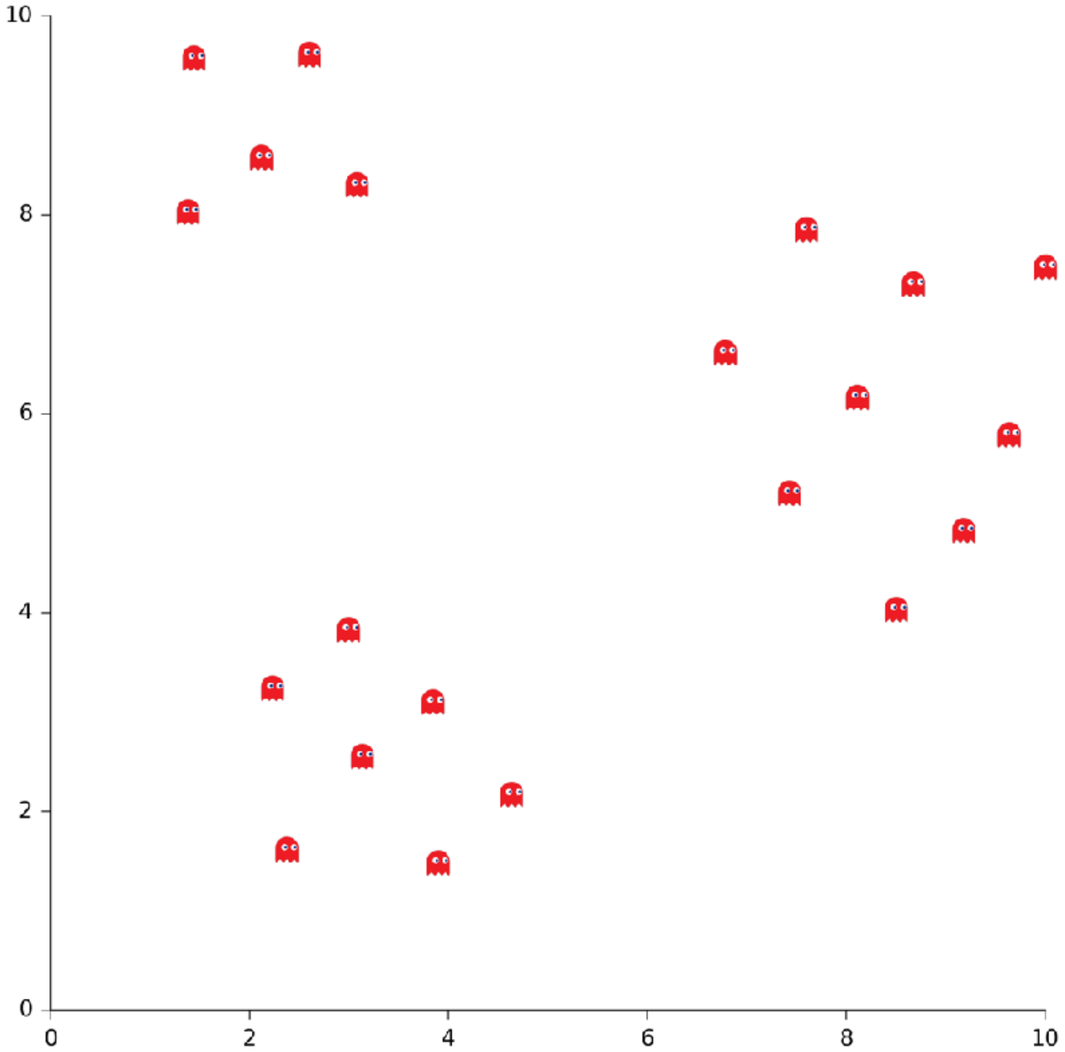
Please take 30-40 minutes and complete

Worksheet 5 - Feature Engineering

Unsupervised Machine Learning

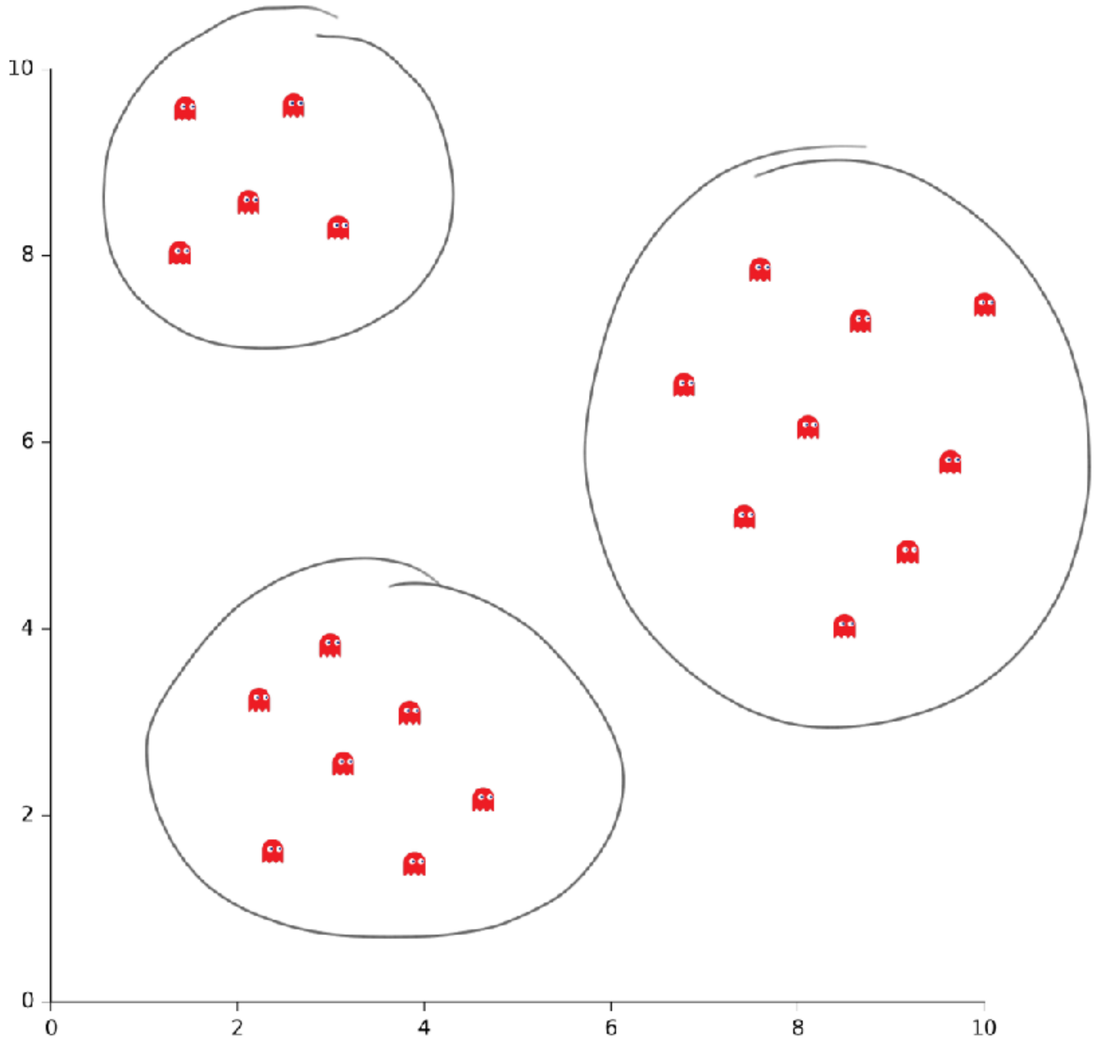
Clustering and Outliers

K-means Clustering



- All “same class”?
- Still some structure?
- Infer “different classes”?

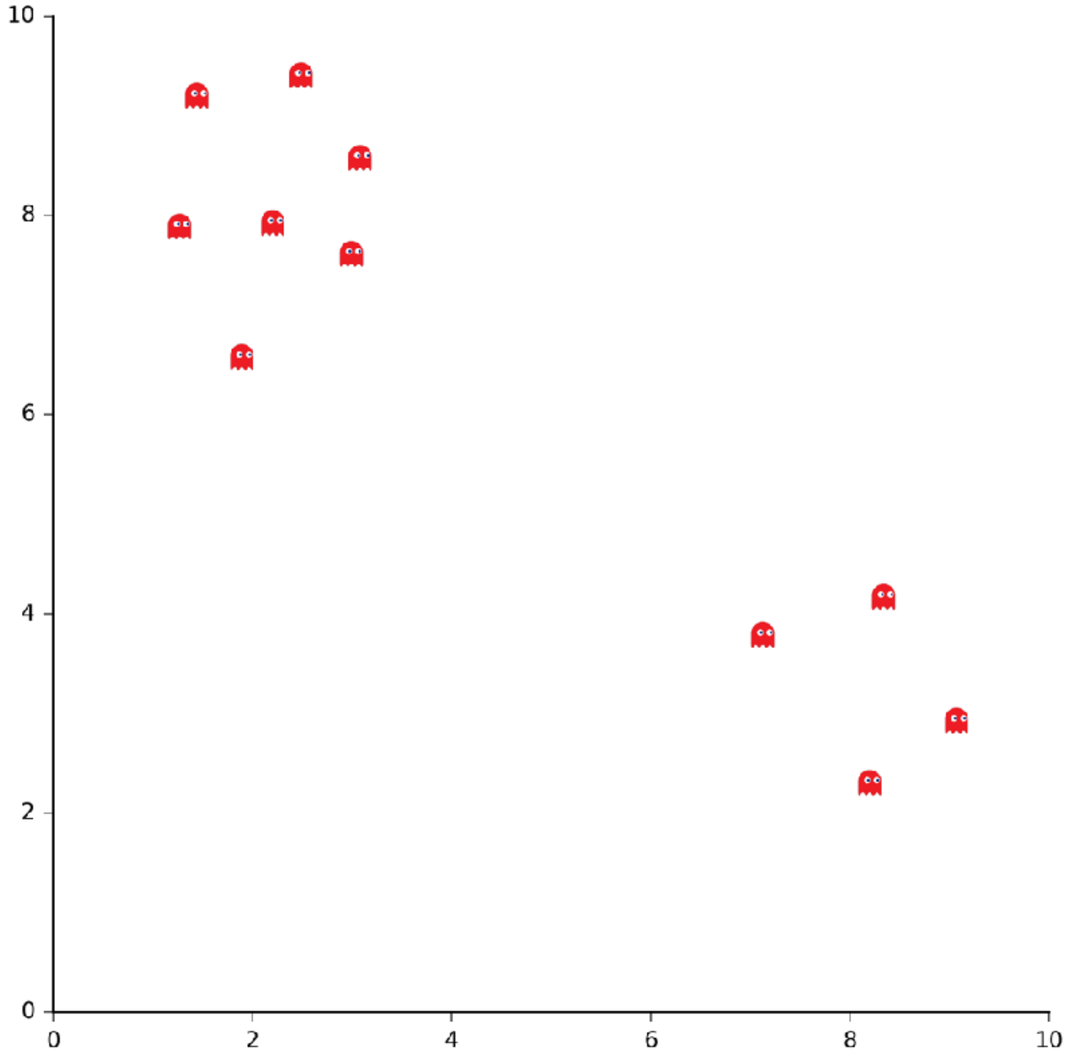
K-means Clustering



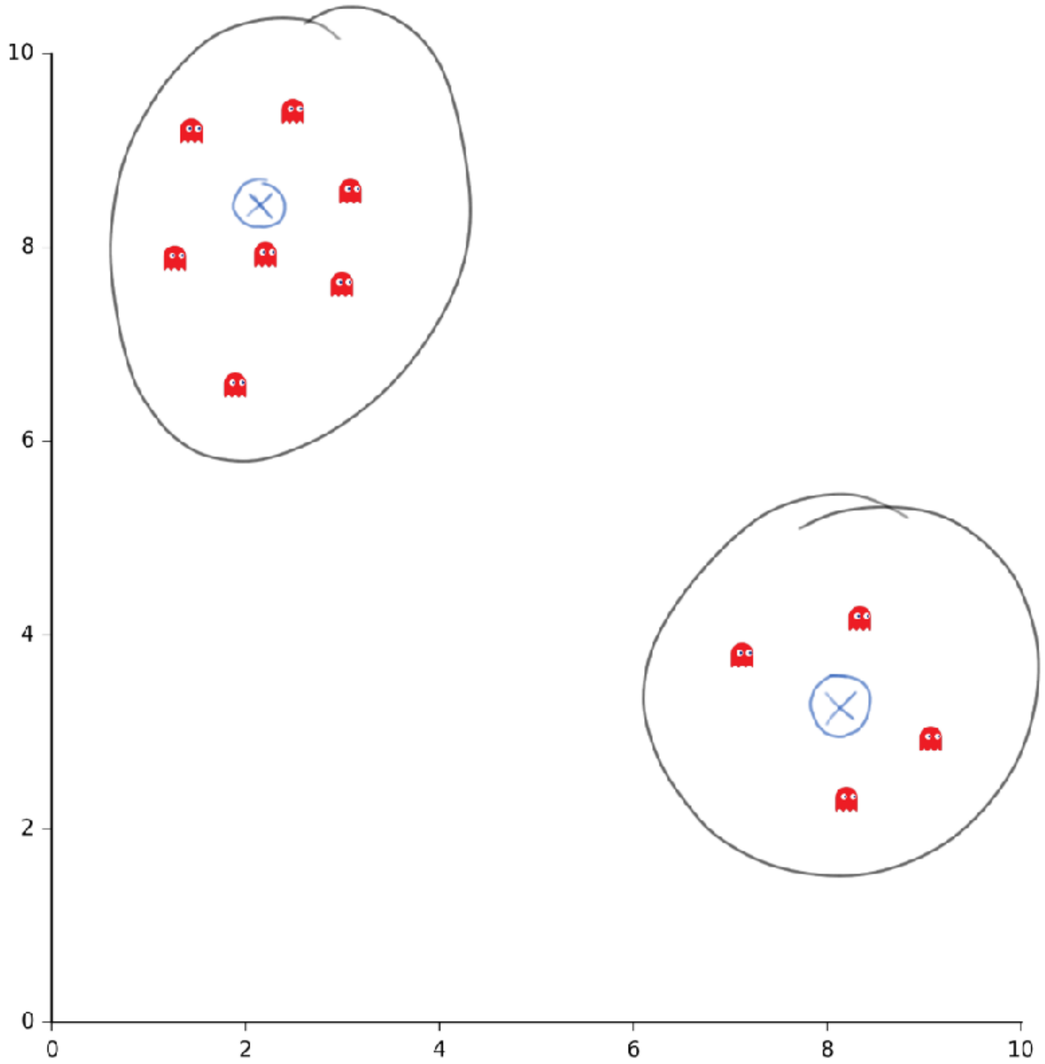
You have to define k
= number of clusters

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

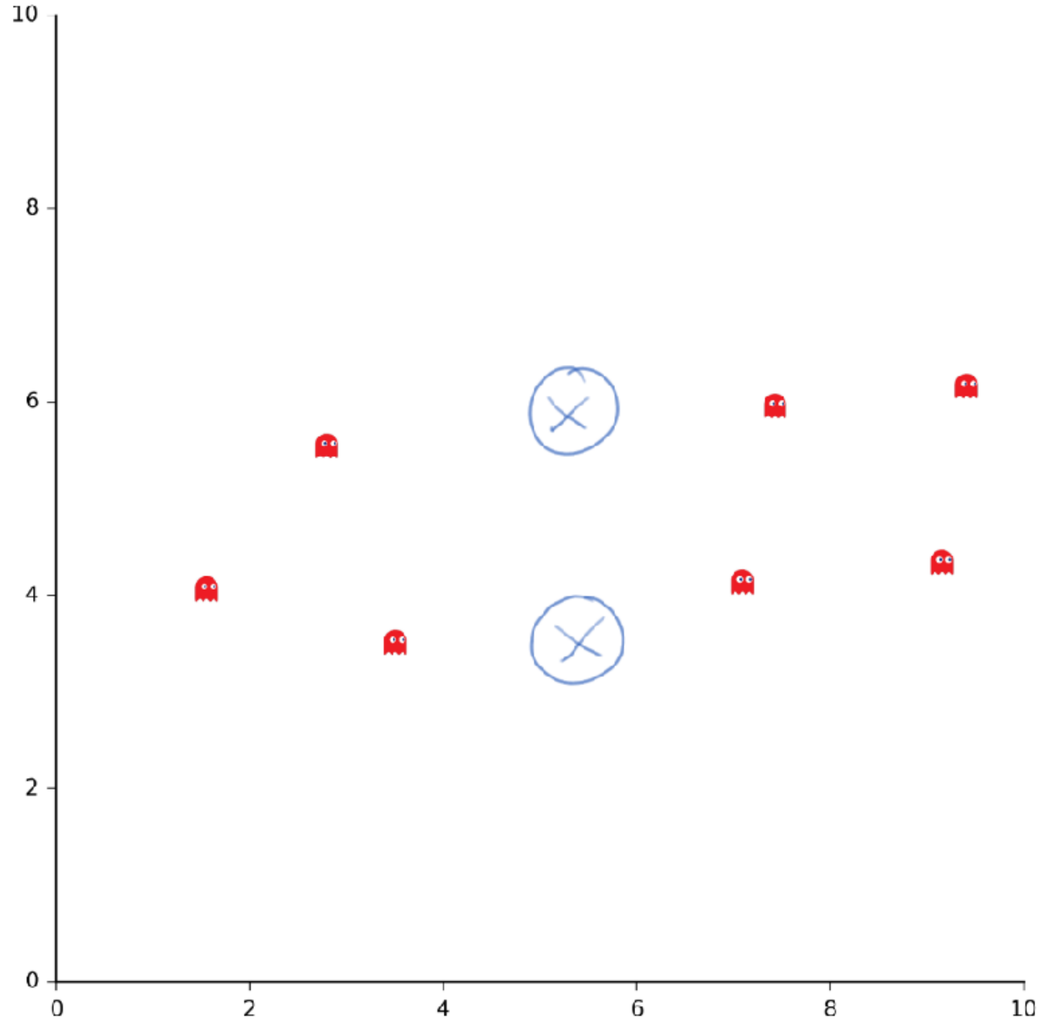
How to cluster?
What are cluster centers?



Optimized cluster centers

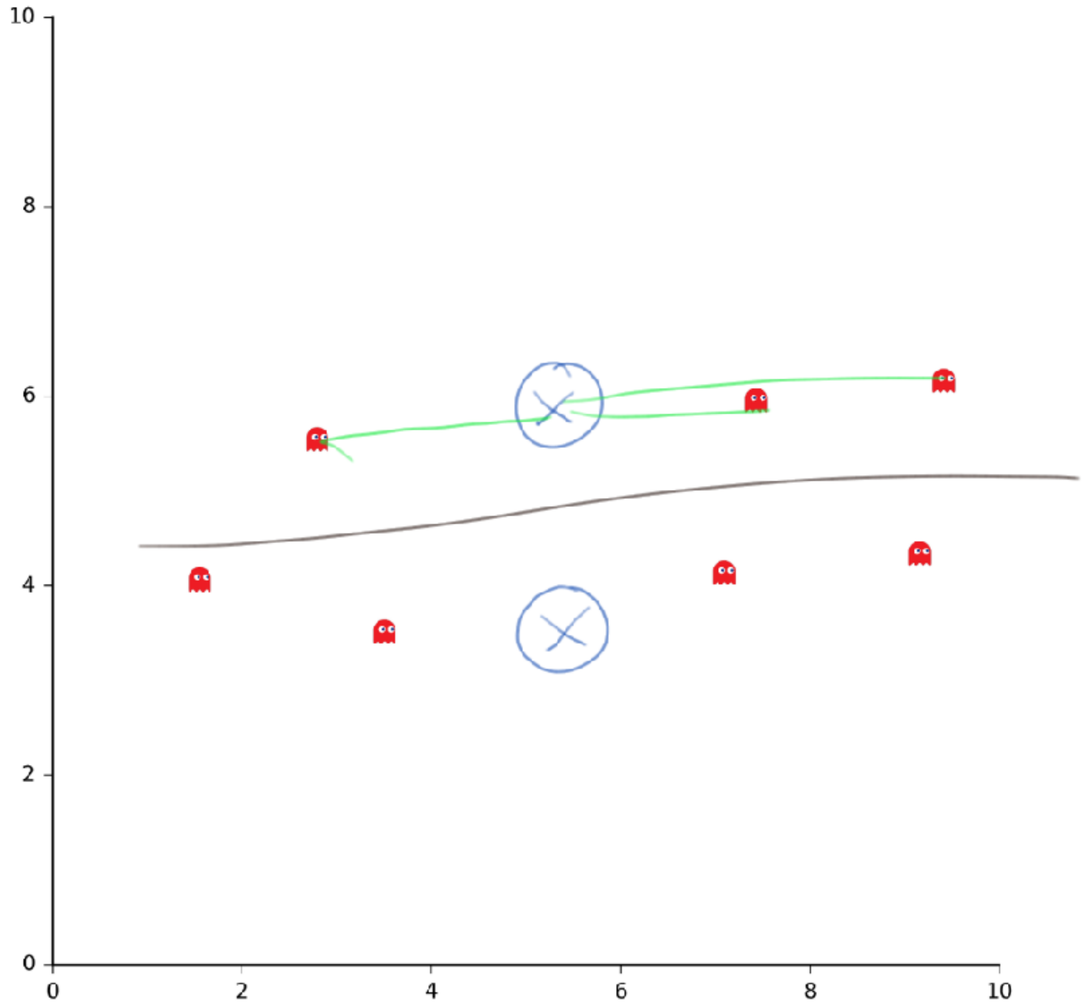


How to find cluster centers?



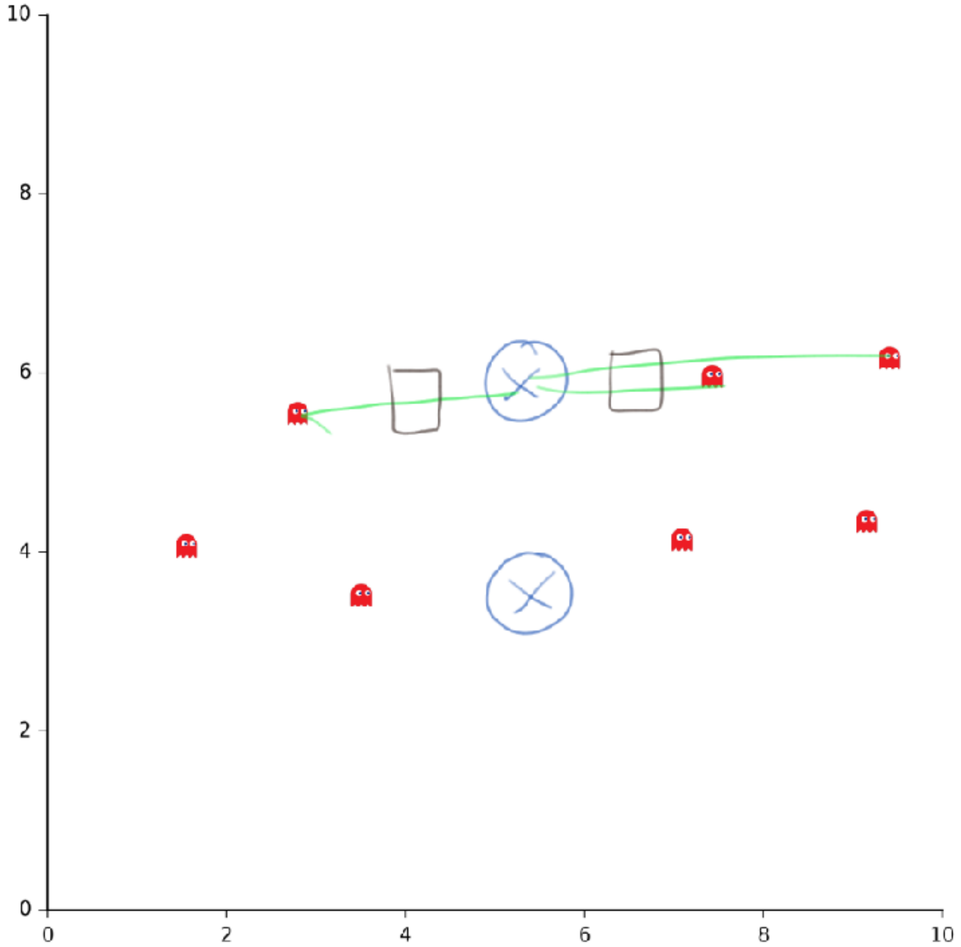
Initial guess!
Can be random
or farthest away

Iterative Optimization



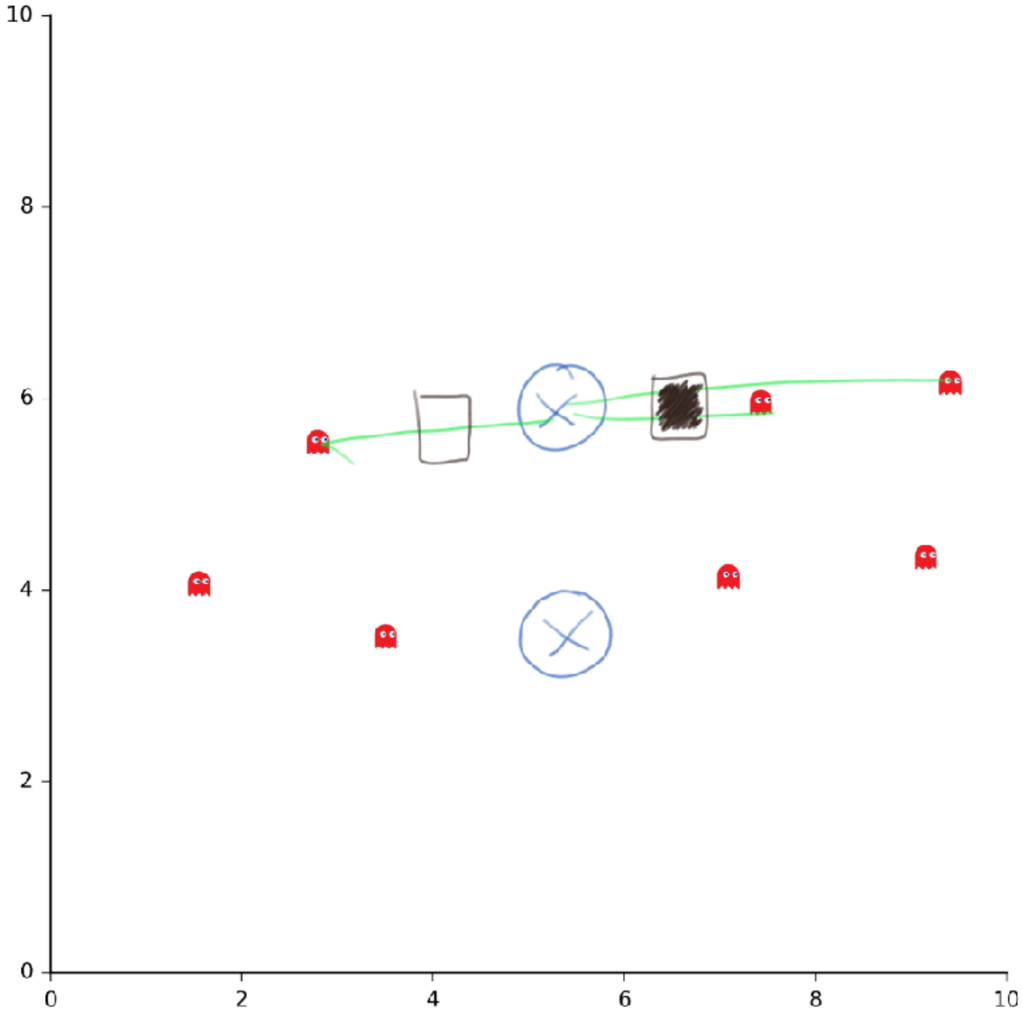
Step 1: Assign

Which black box would be the next best choice?

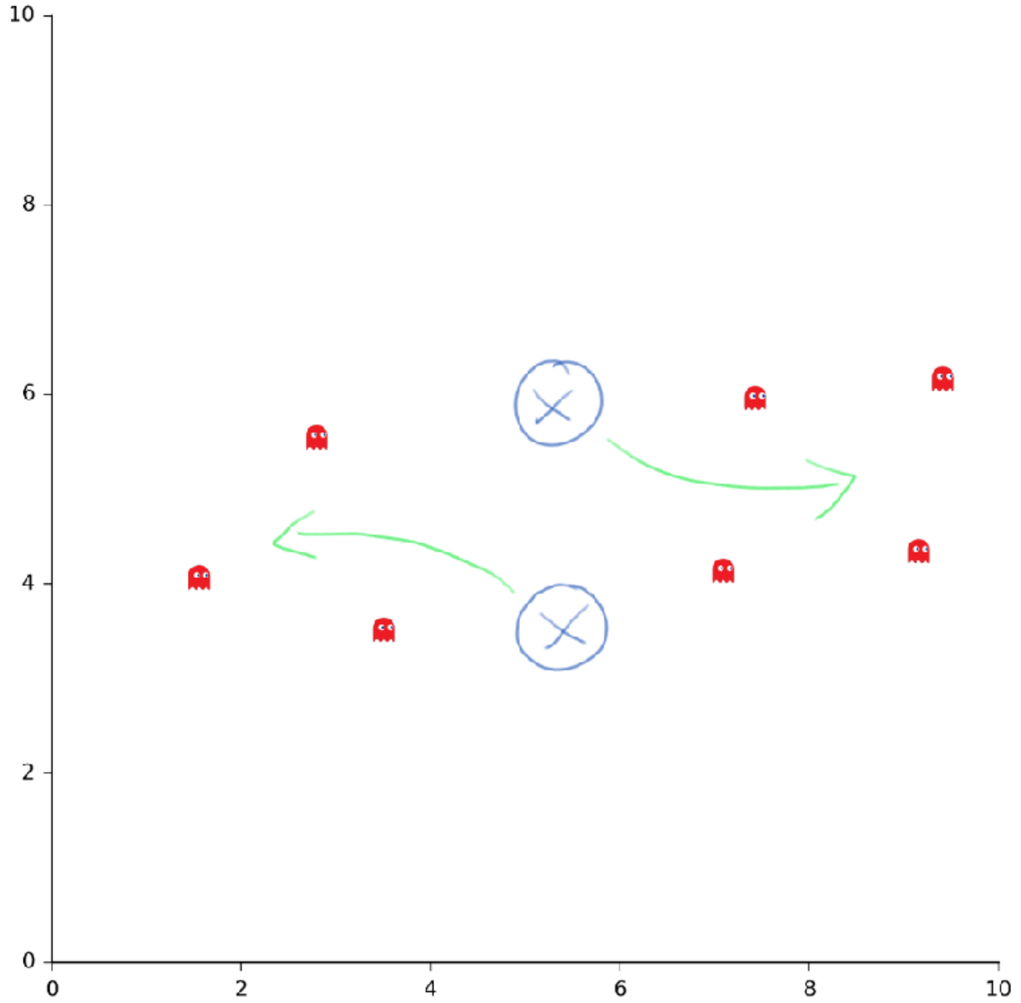


Step 2: Optimize

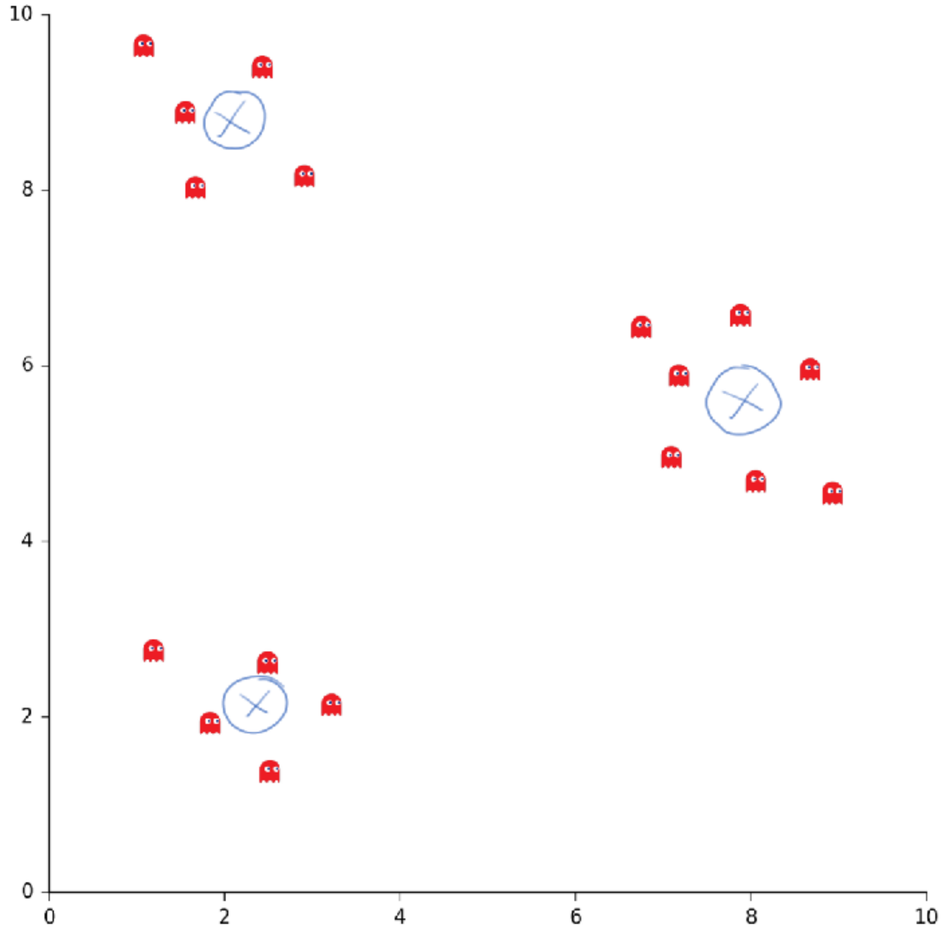
Answer



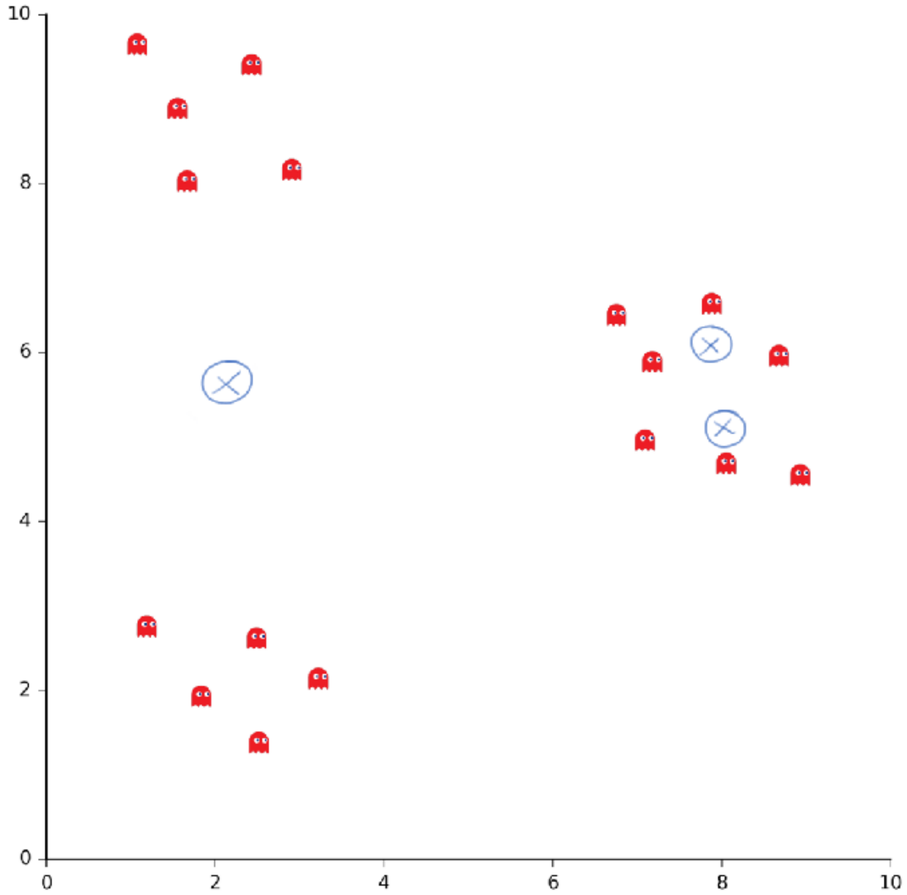
Iterative Optimization



Which one is correct?

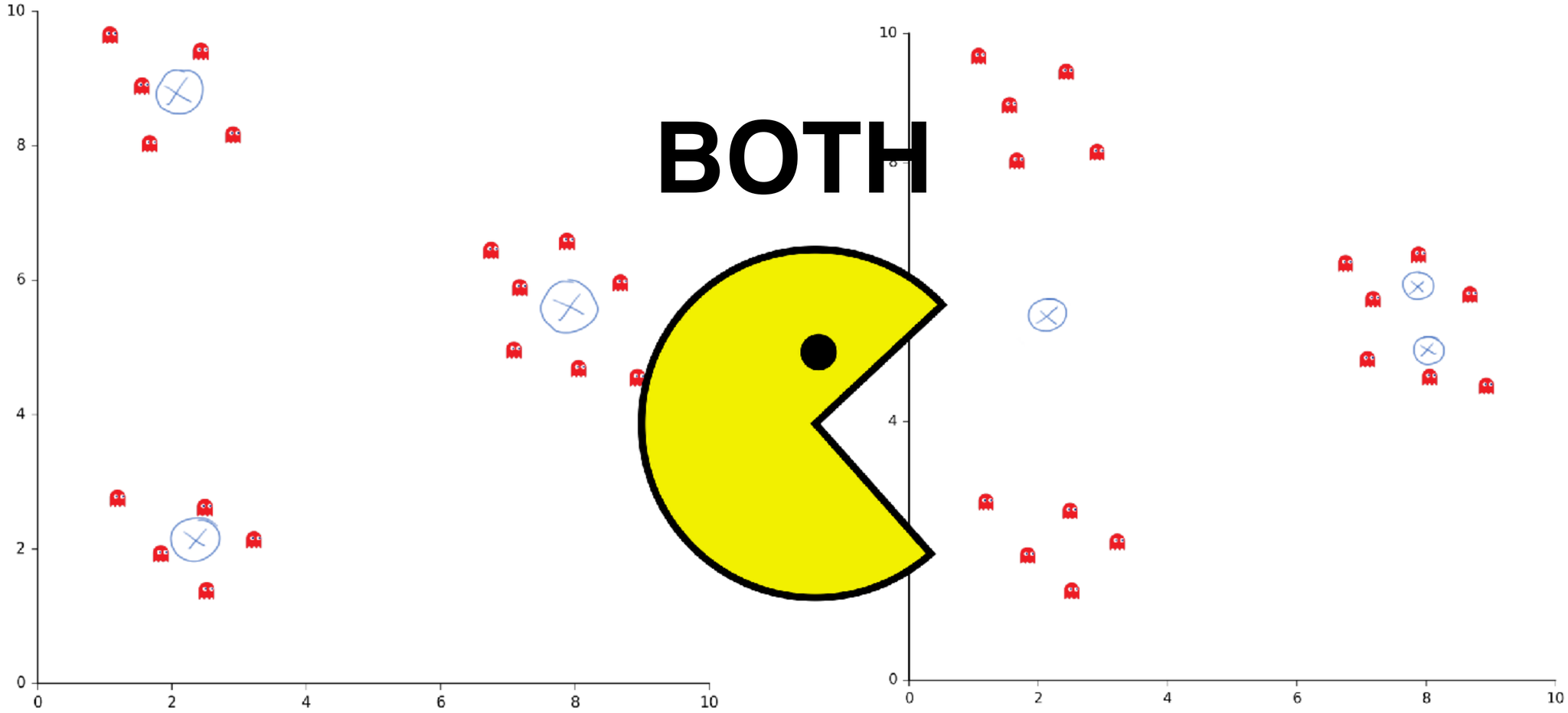


A

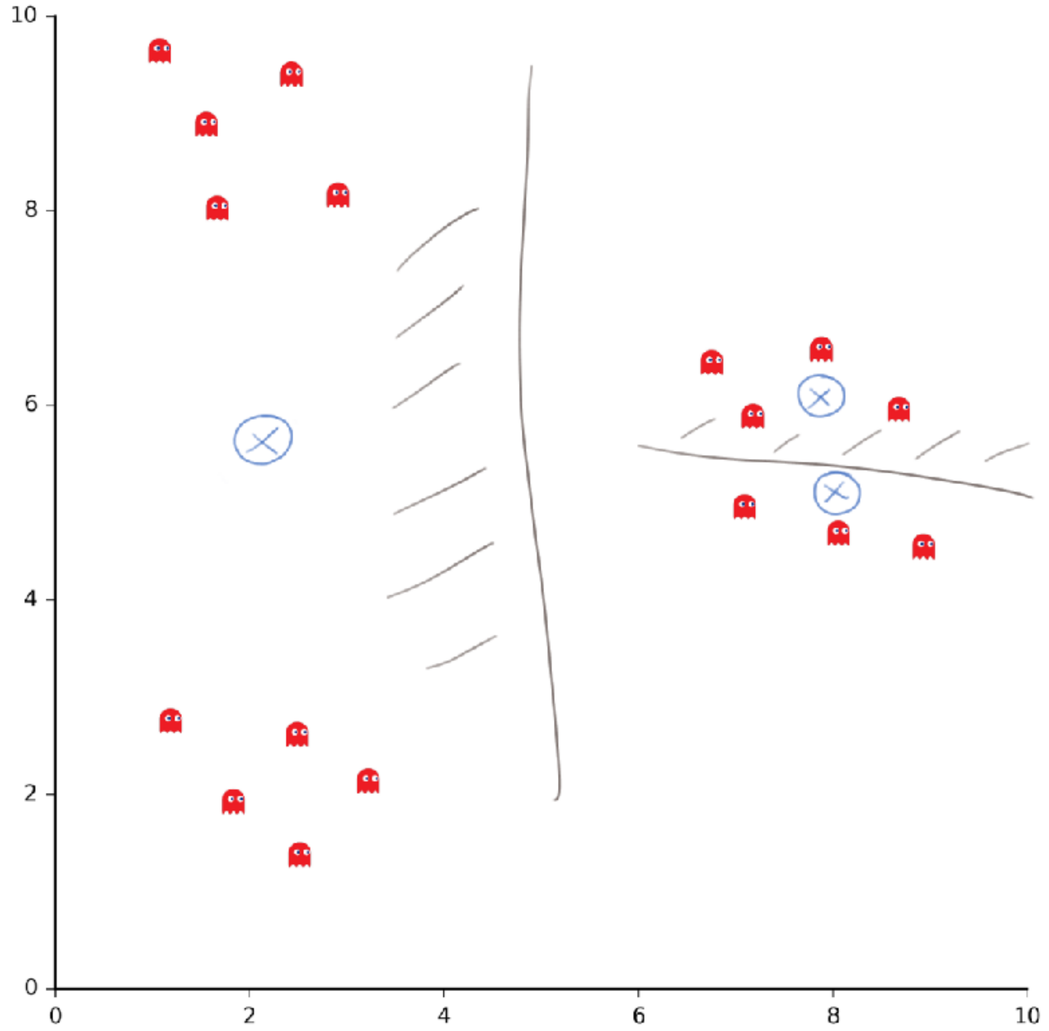


B

Which one is correct?



Pain of optimization…Being stuck at sub-optimal local minimum…



Initial guess matters!
Same outcome
cannot be guaranteed

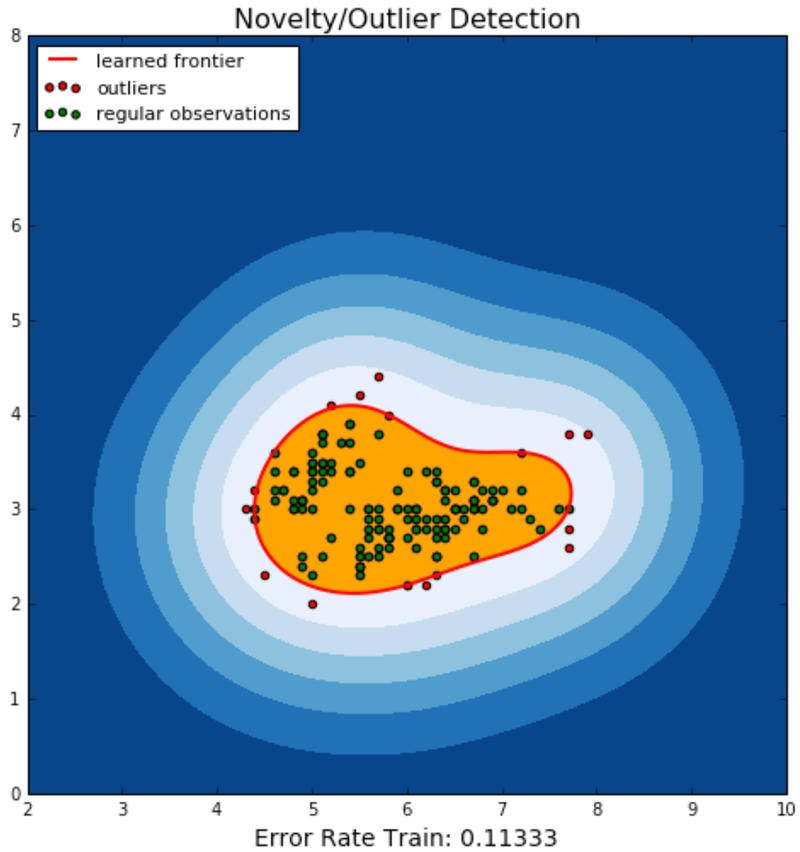
Sklearn's K-means Clustering

```
n_clusters=3
labels_km = cluster.KMeans(n_clusters=n_clusters).fit_predict(X)
print(labels_km)
```

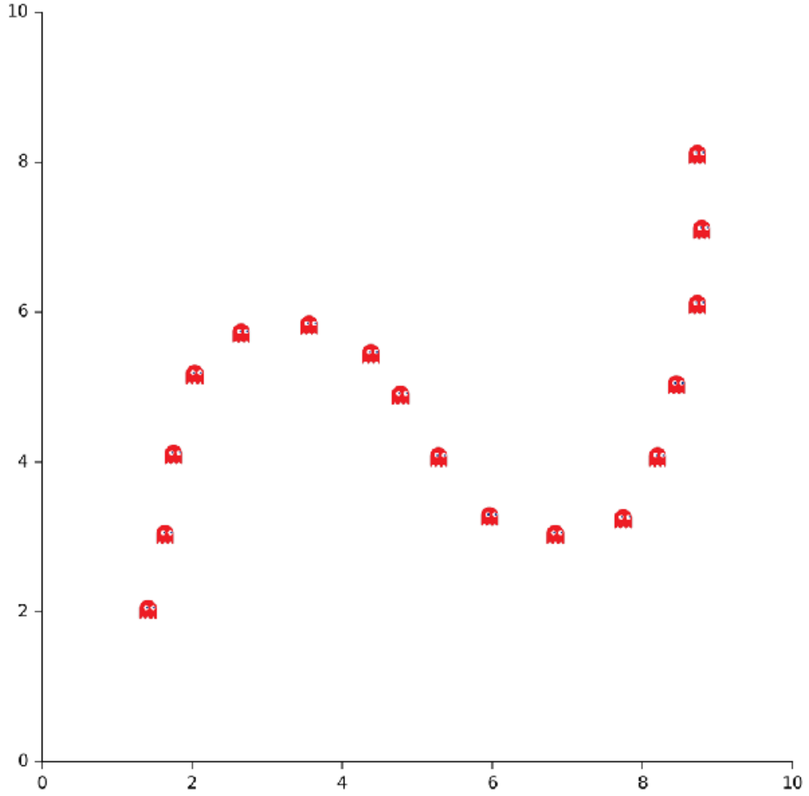
Outlier Detection

```
clf = svm.OneClassSVM(kernel='rbf', degree=3, gamma='auto', \
    coef0=0.0, tol=0.001, nu=0.1)

clf.fit(X)
target_pred_outliers=clf.predict(X)
```

Delete n% of “outlier data”,
here ~10%

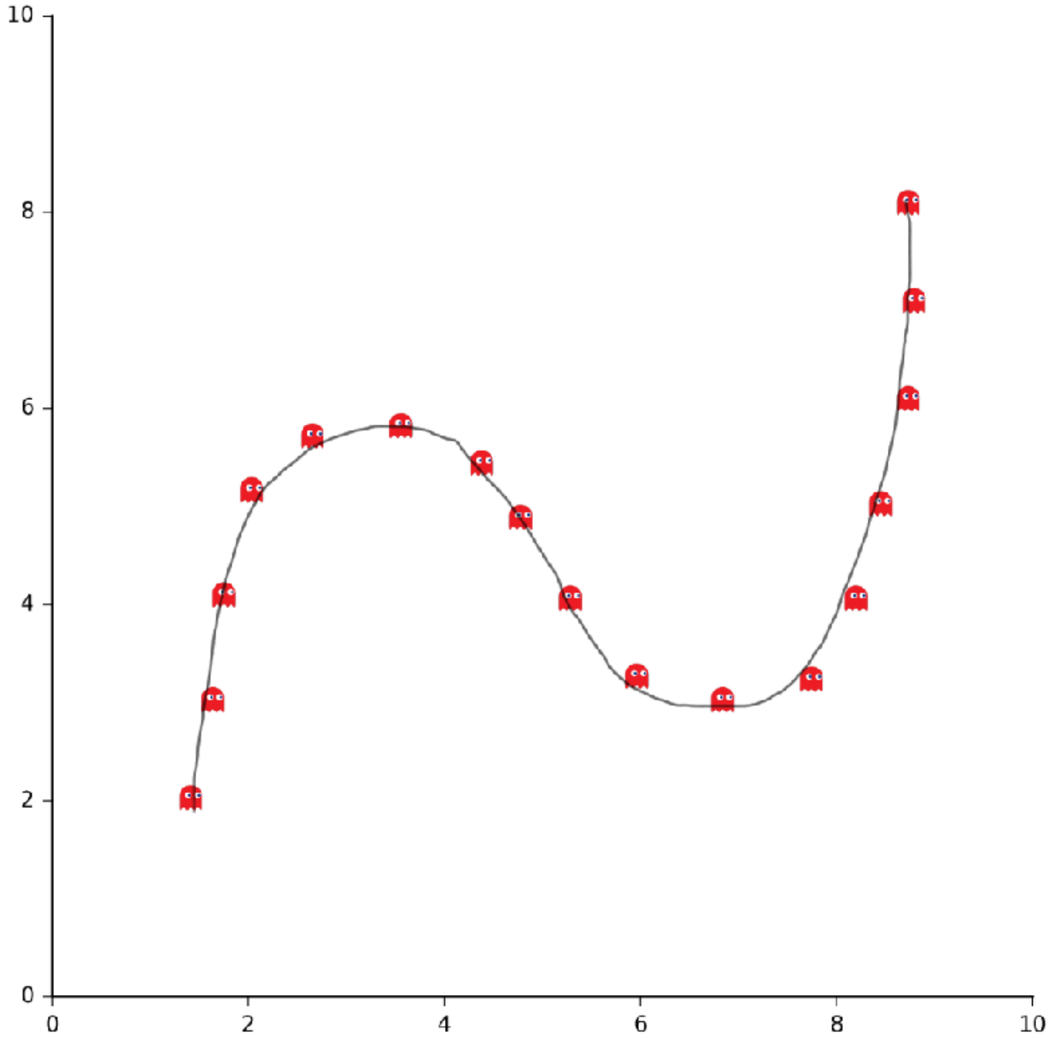


Dimensionality of Data



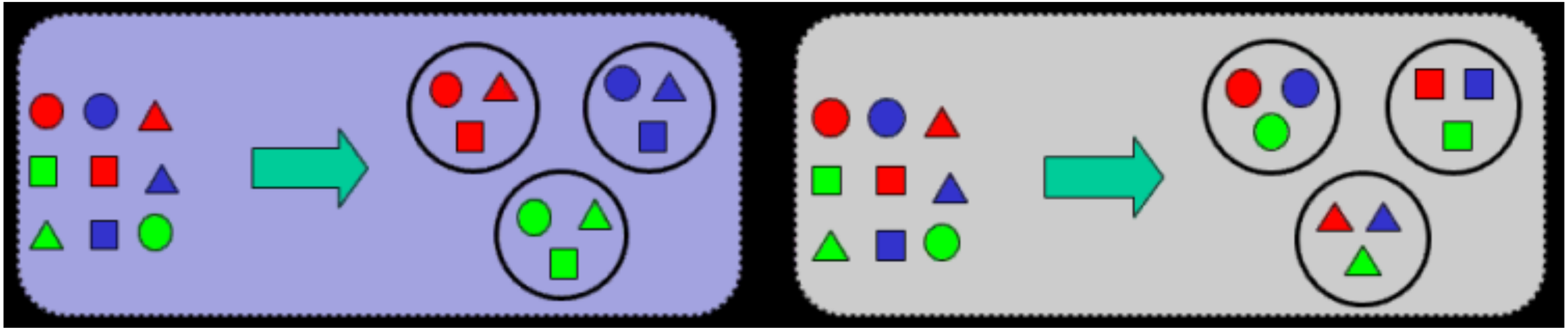
3D, 2D or 1D?

Dimensionality of Data



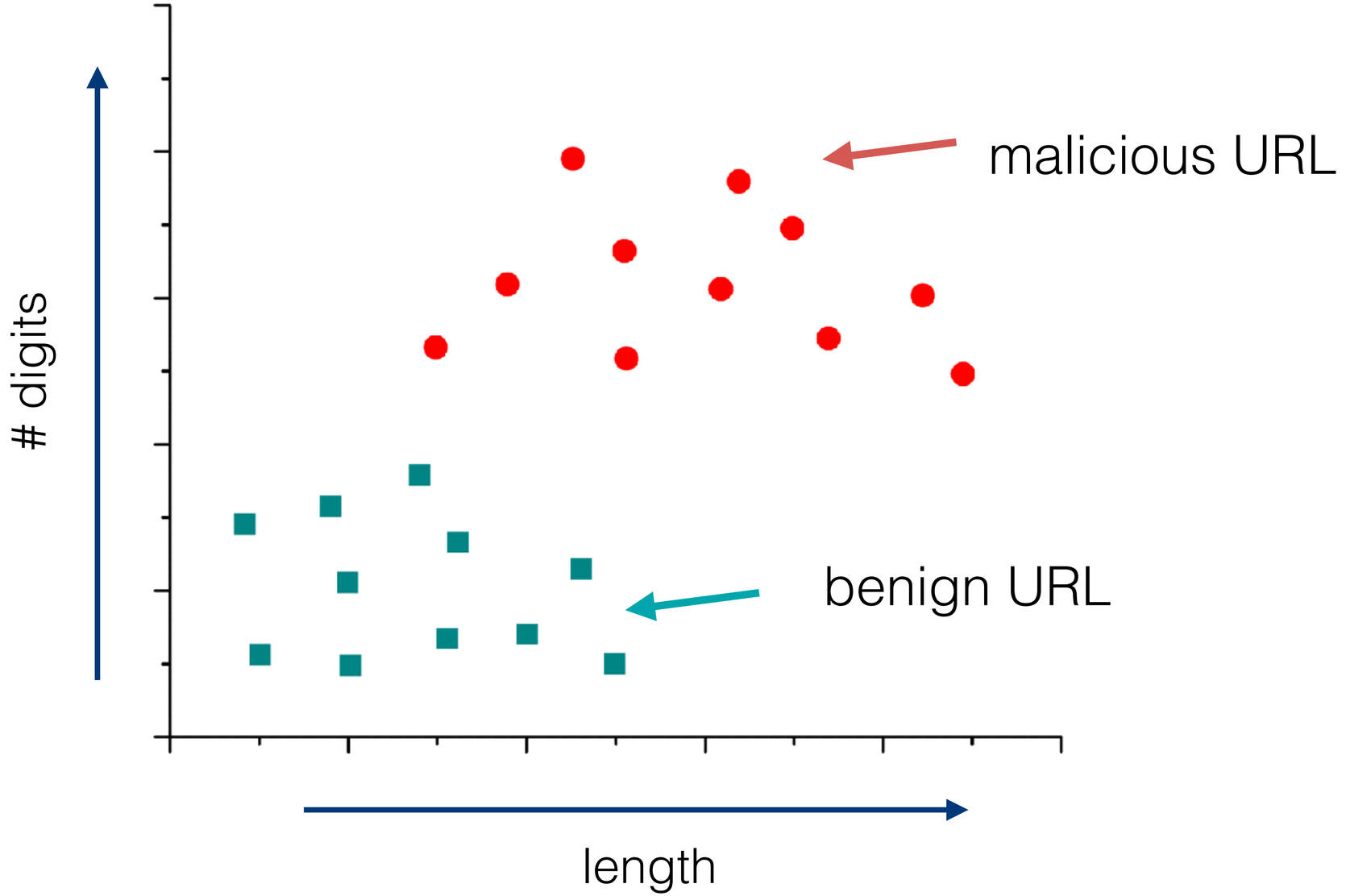
1D - line

Final Note:
Data Representation and Feature Selection

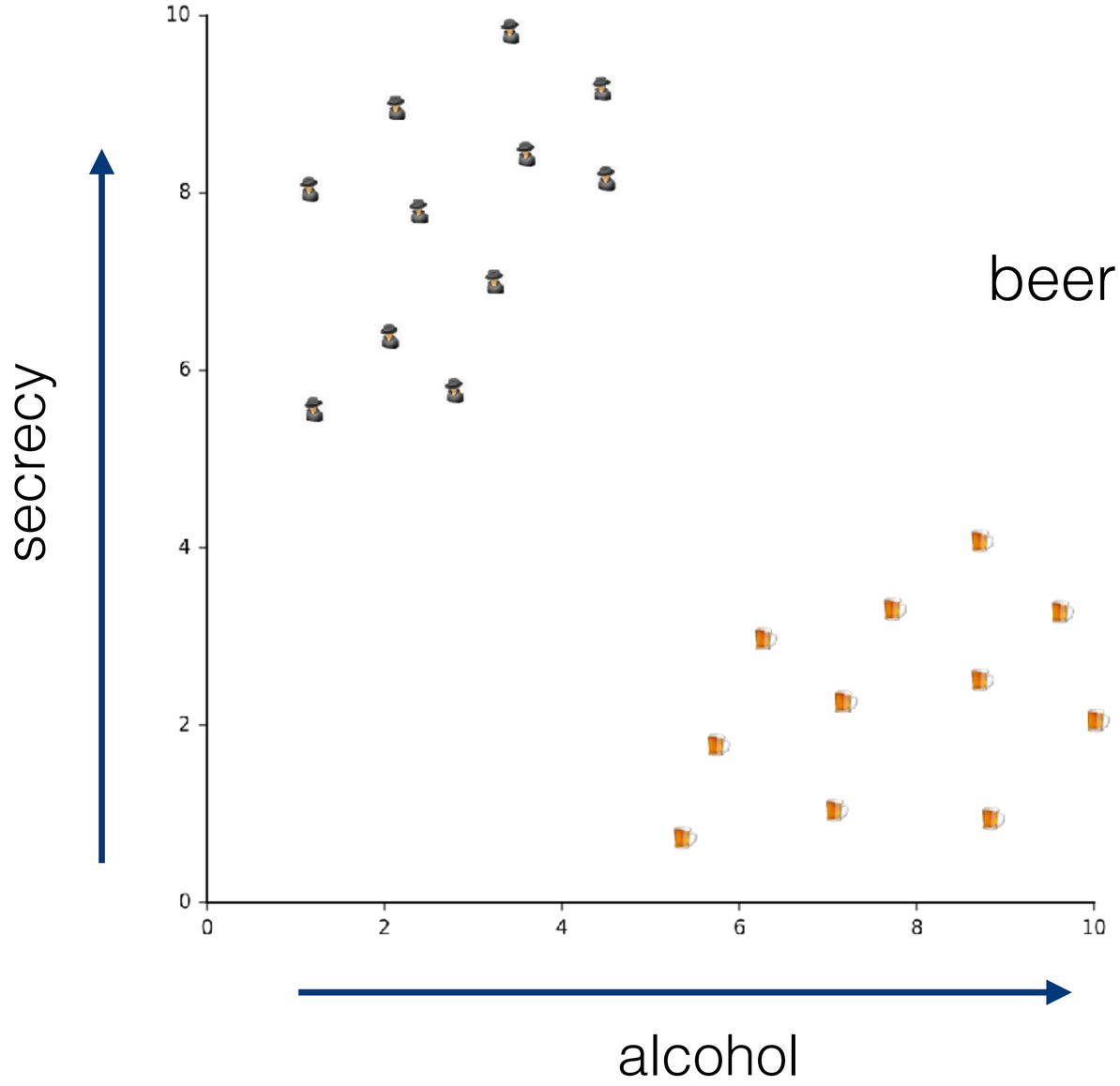


Supervised Machine Learning

Features and Labels

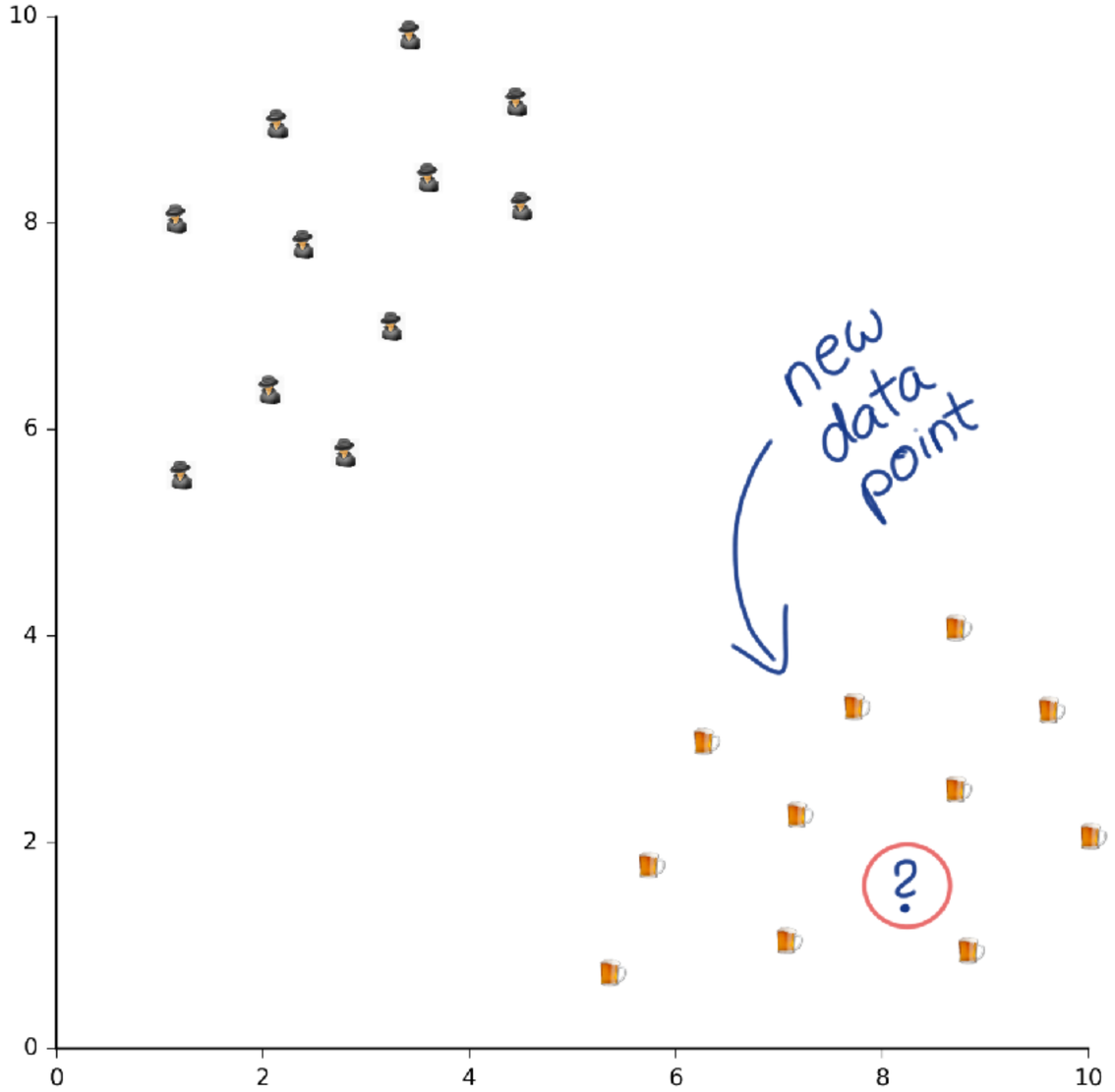


FUN Features and Labels

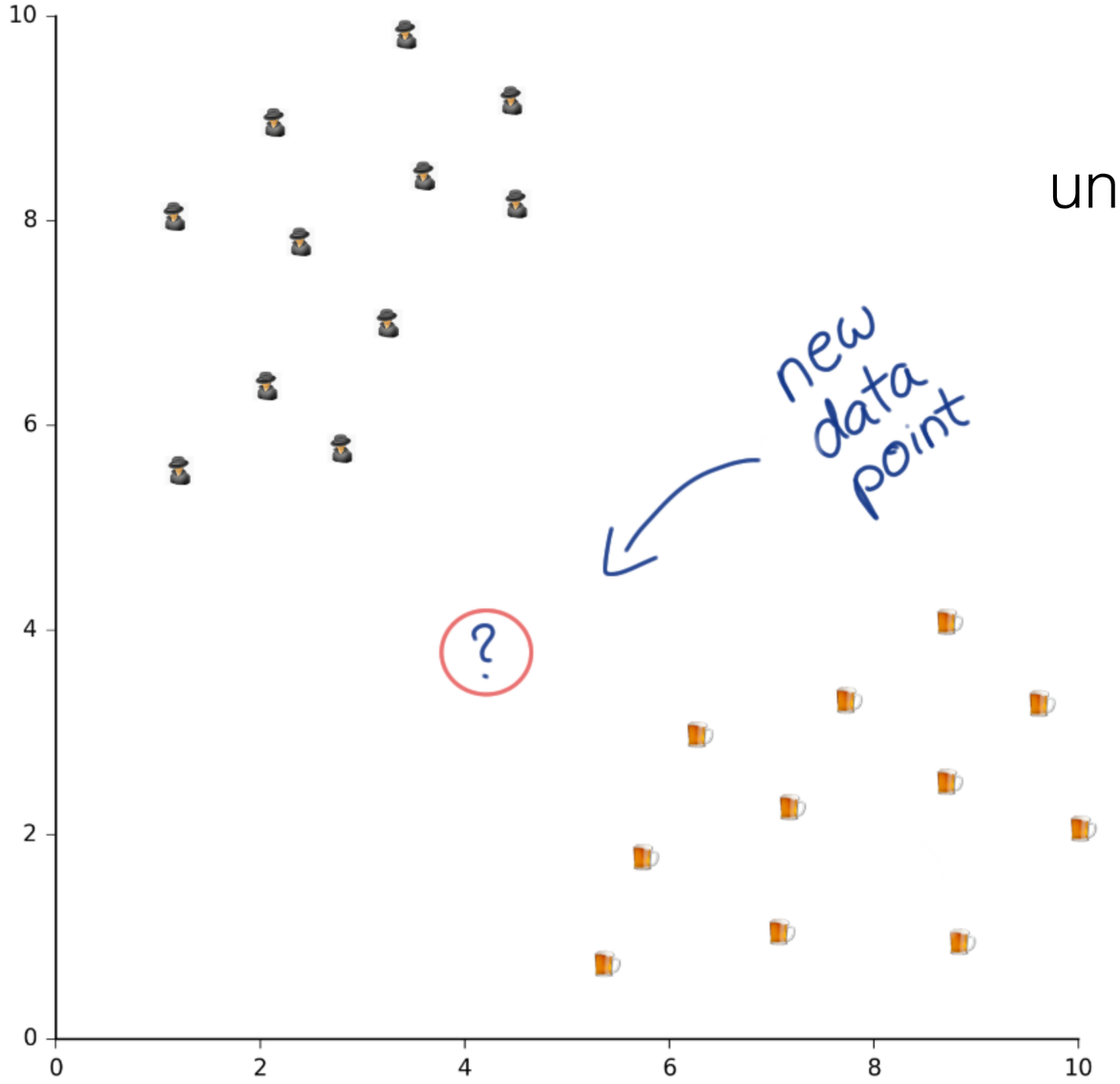


beer and cyber !!!

Making a new decision

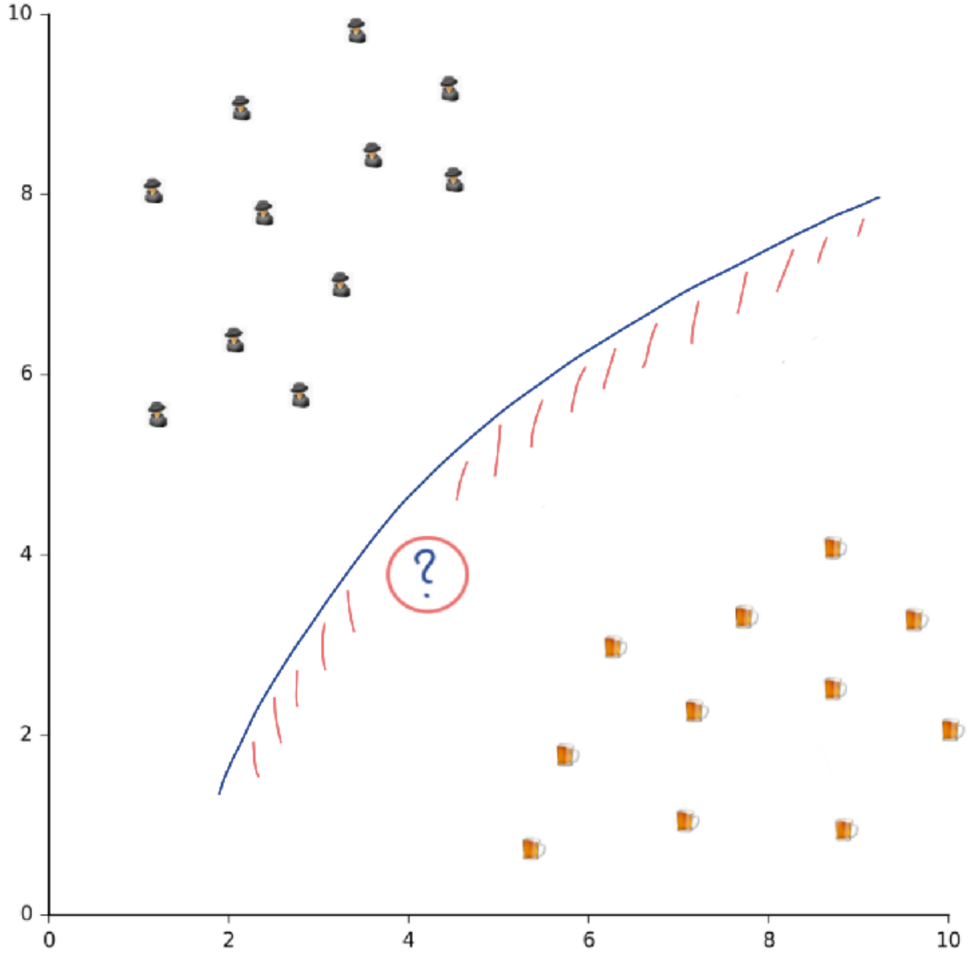


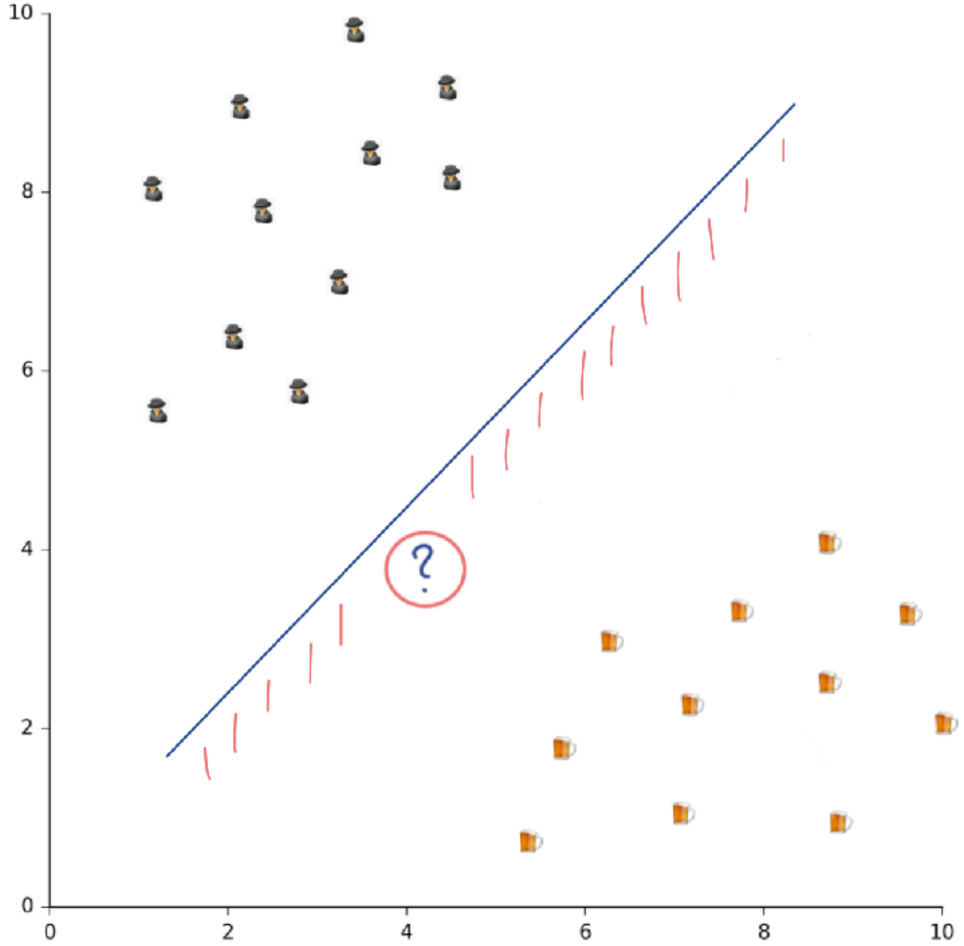
Making a new decision



unclear?

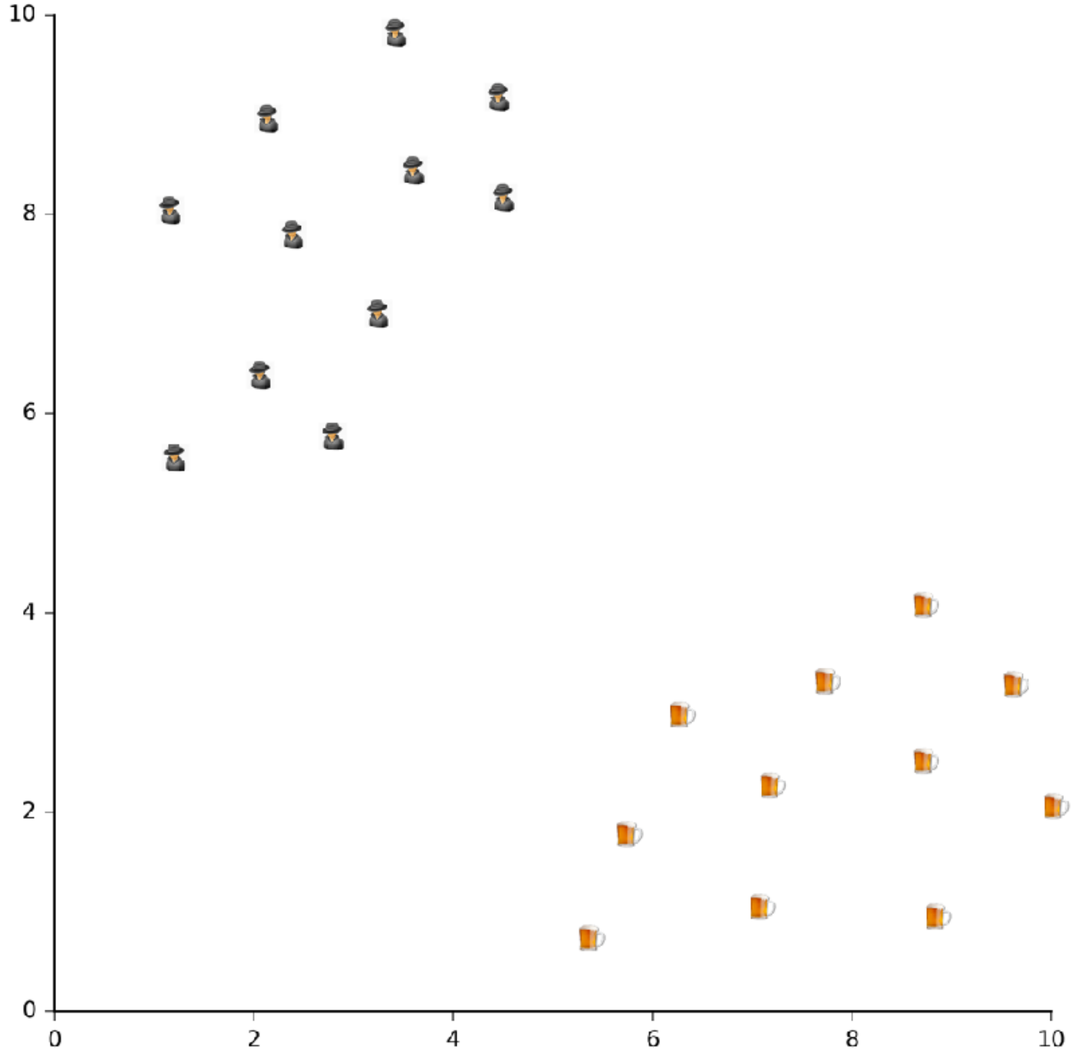
Decision surface concept



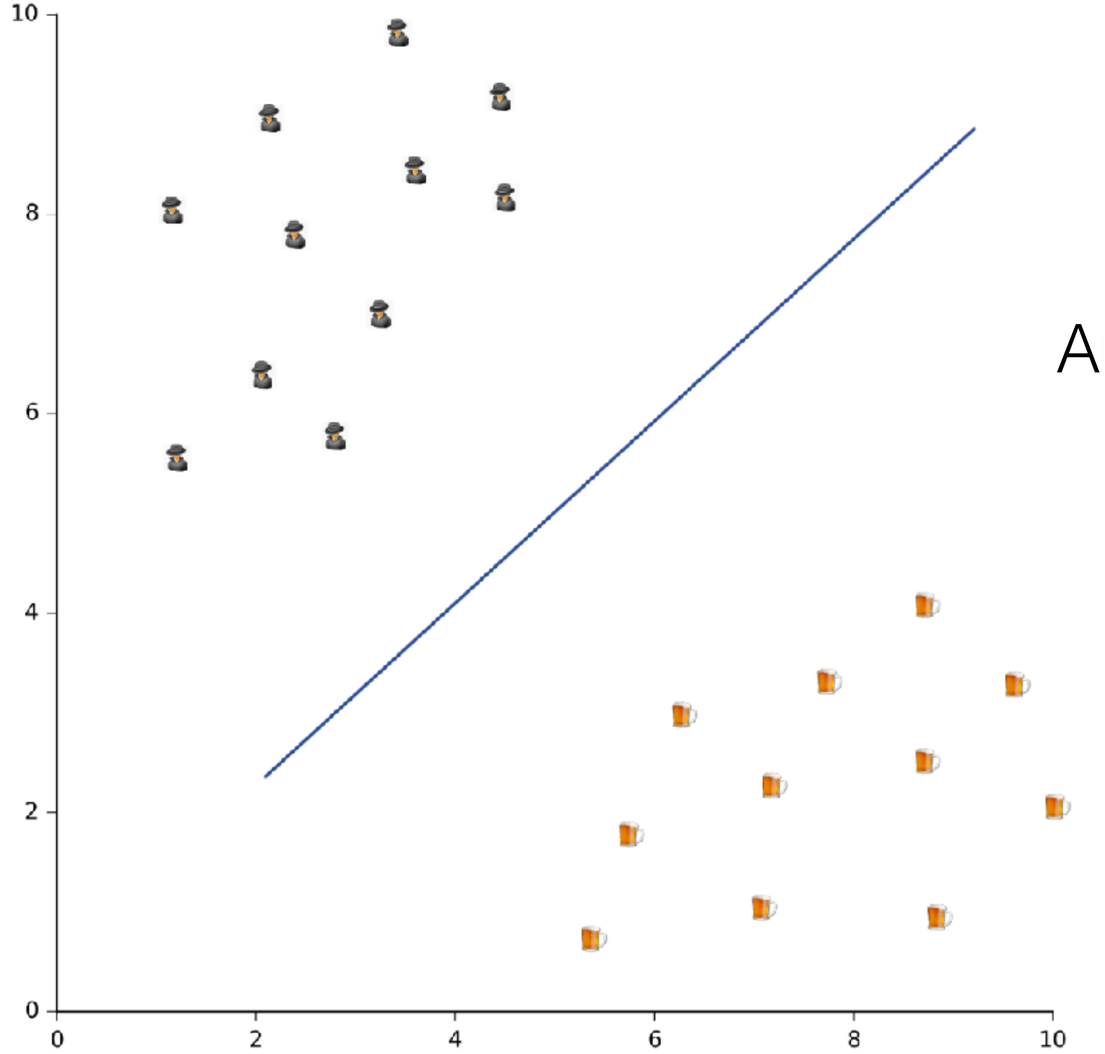


Support Vector Machines (SVM)

How can we **linearly** separate beer and cyber?

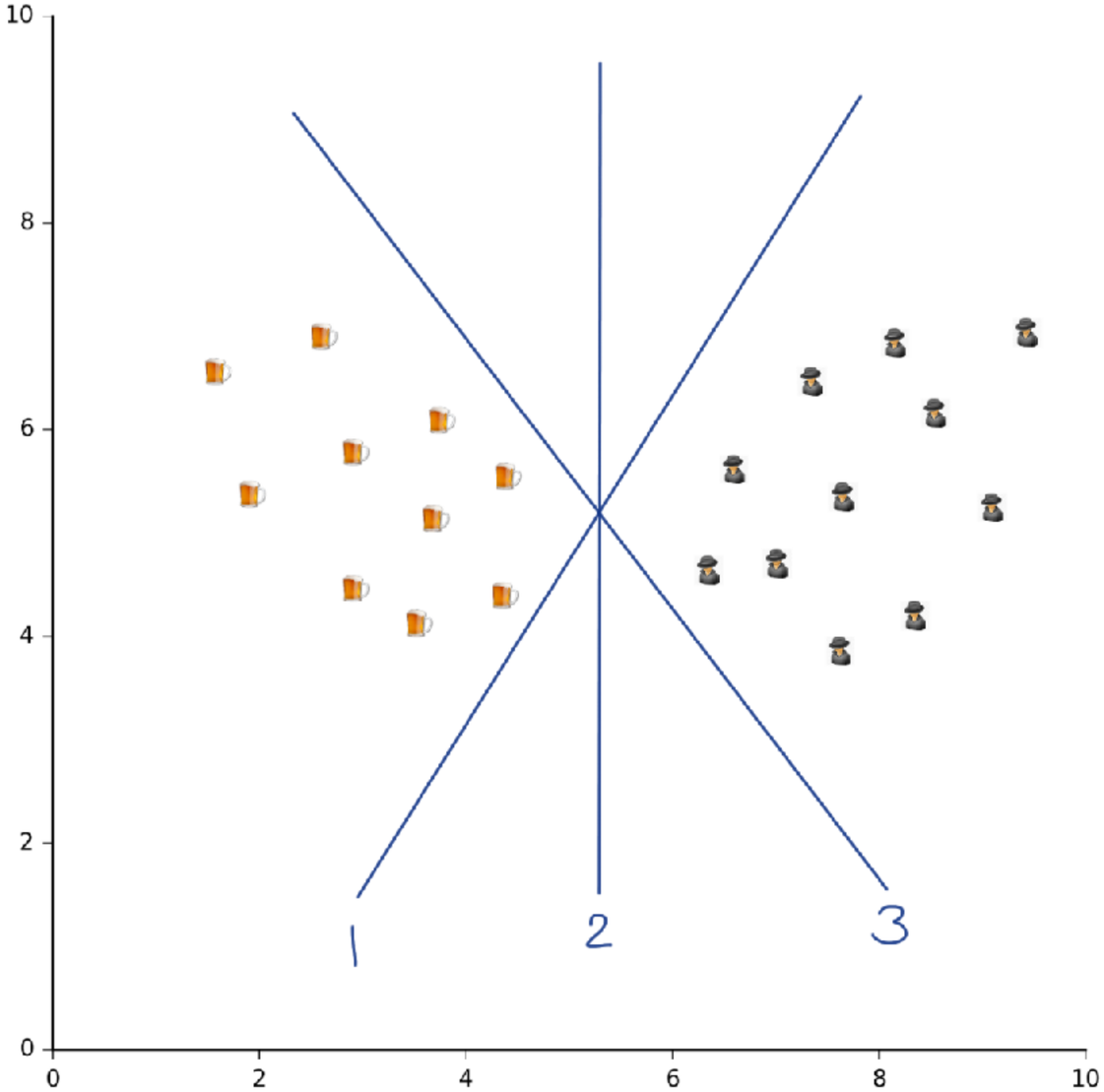


How can we **linearly** separate beer and cyber?

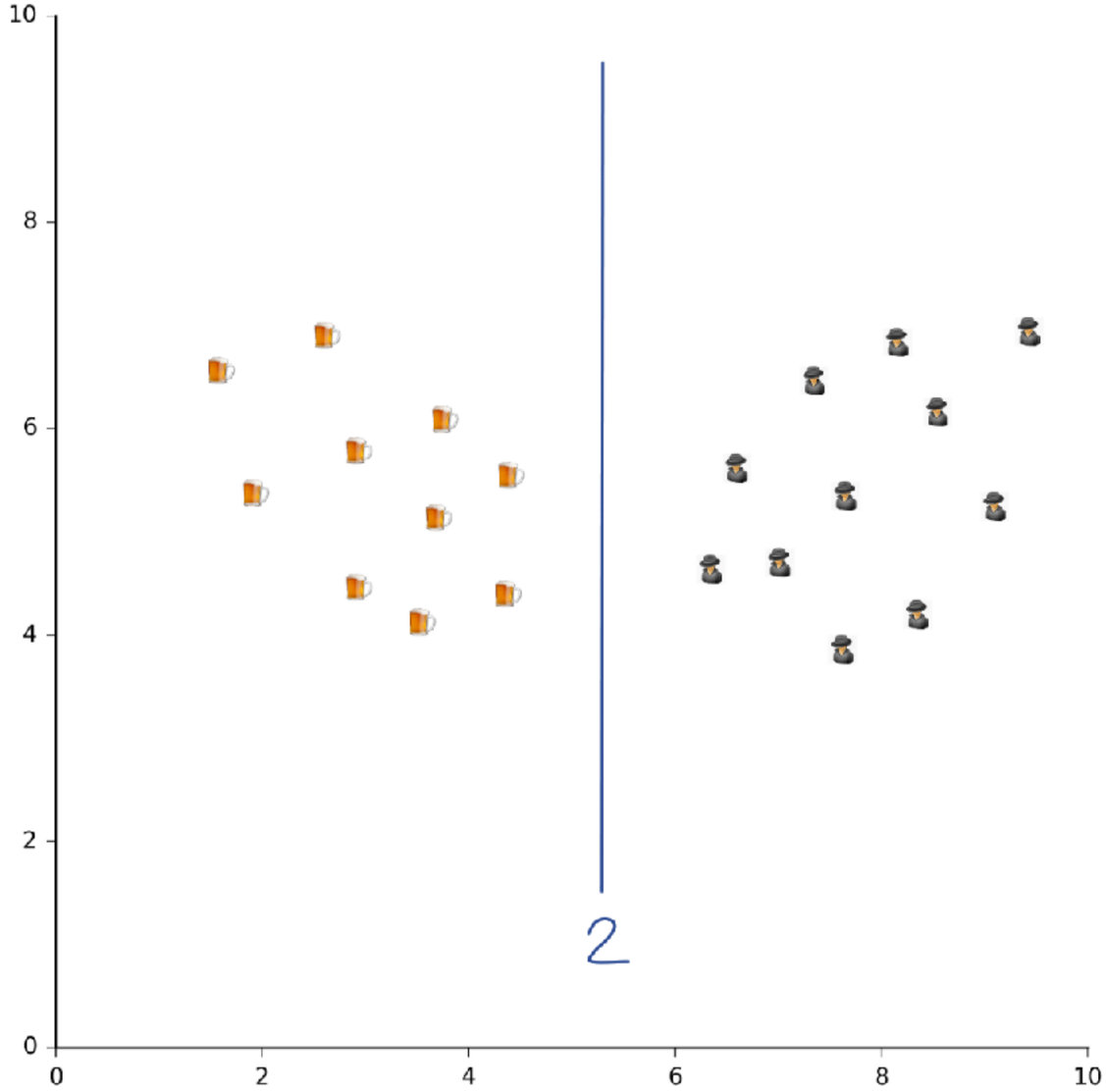


Answer: straight line

Same question, choose the best line!

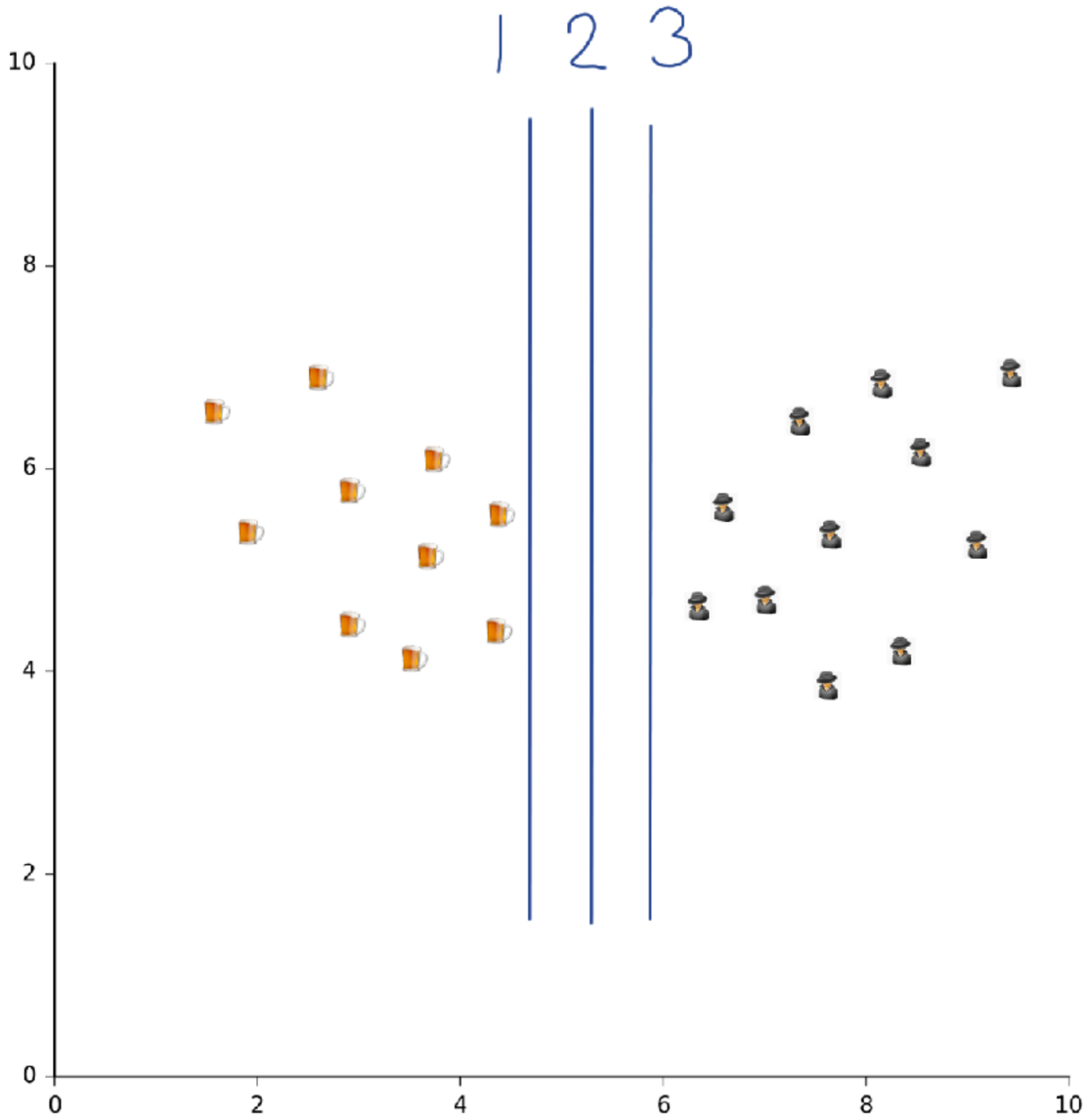


Same question, choose the best line!

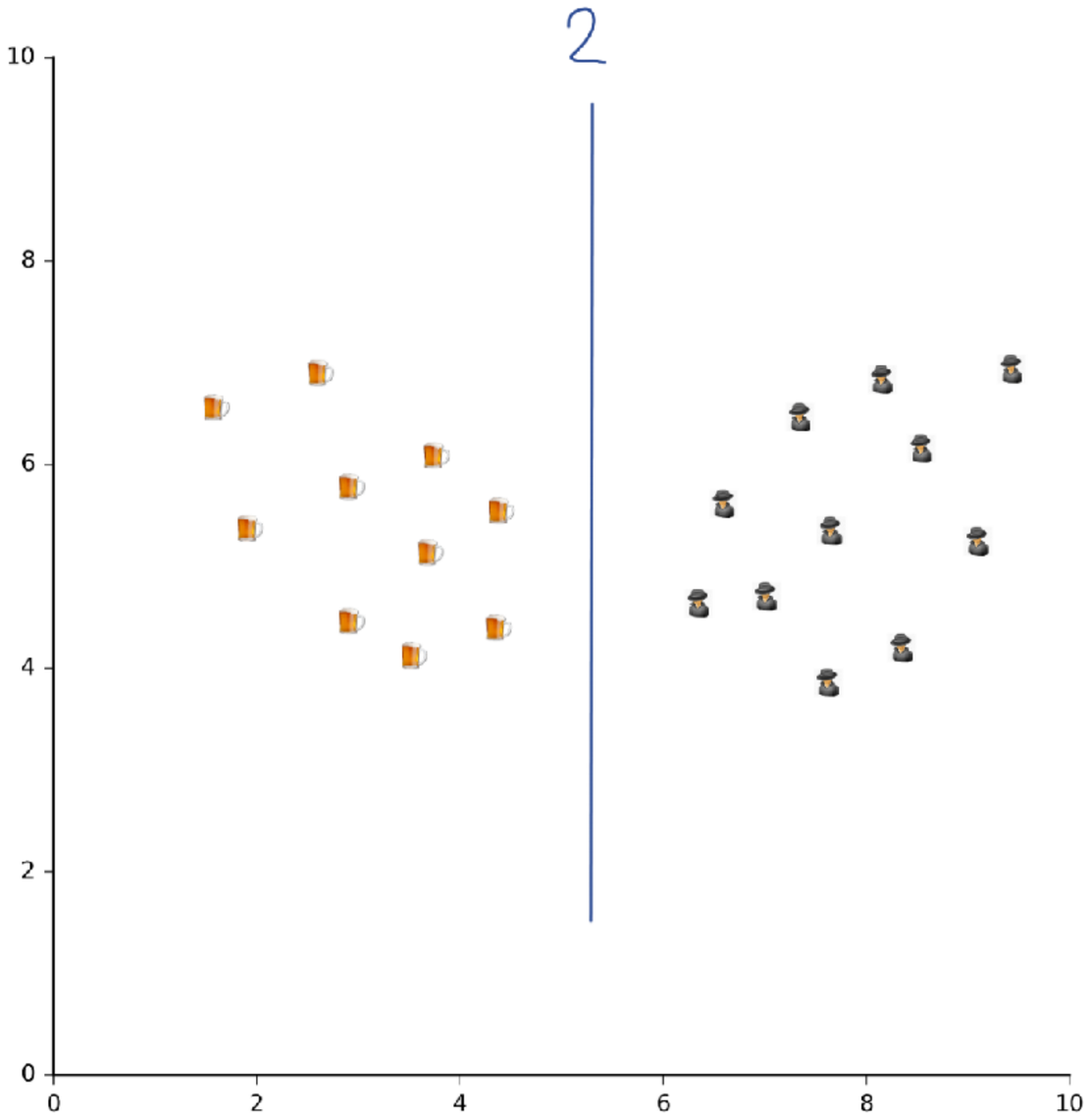


Answer: 2

How about now?

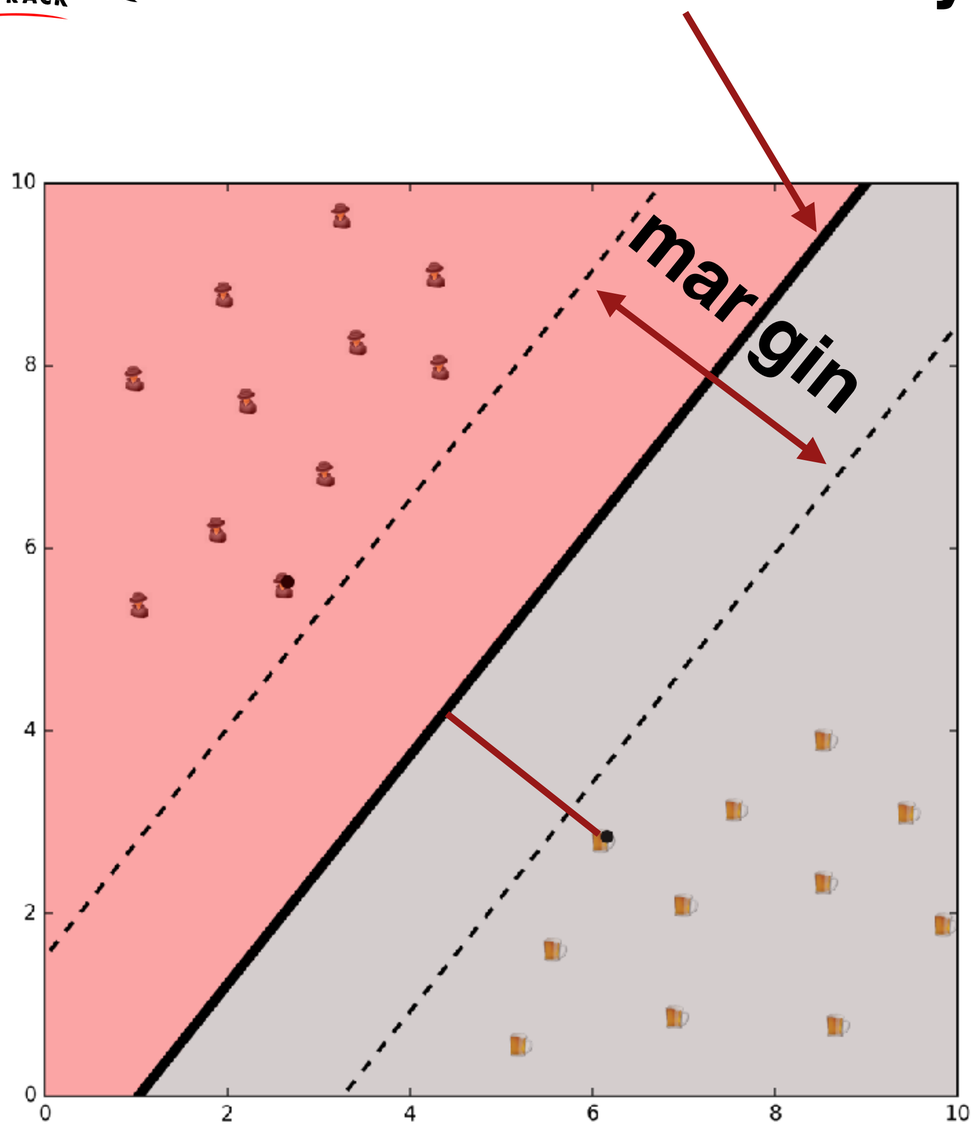


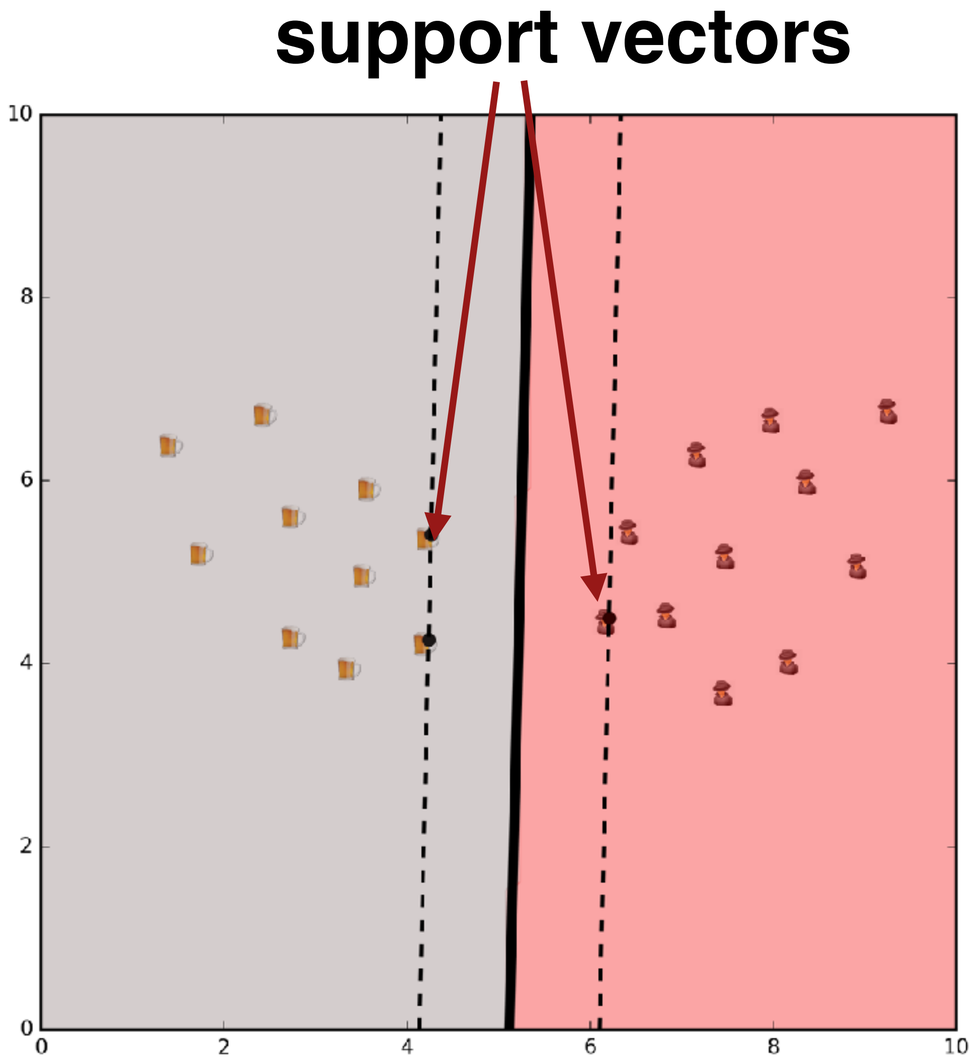
How about now?



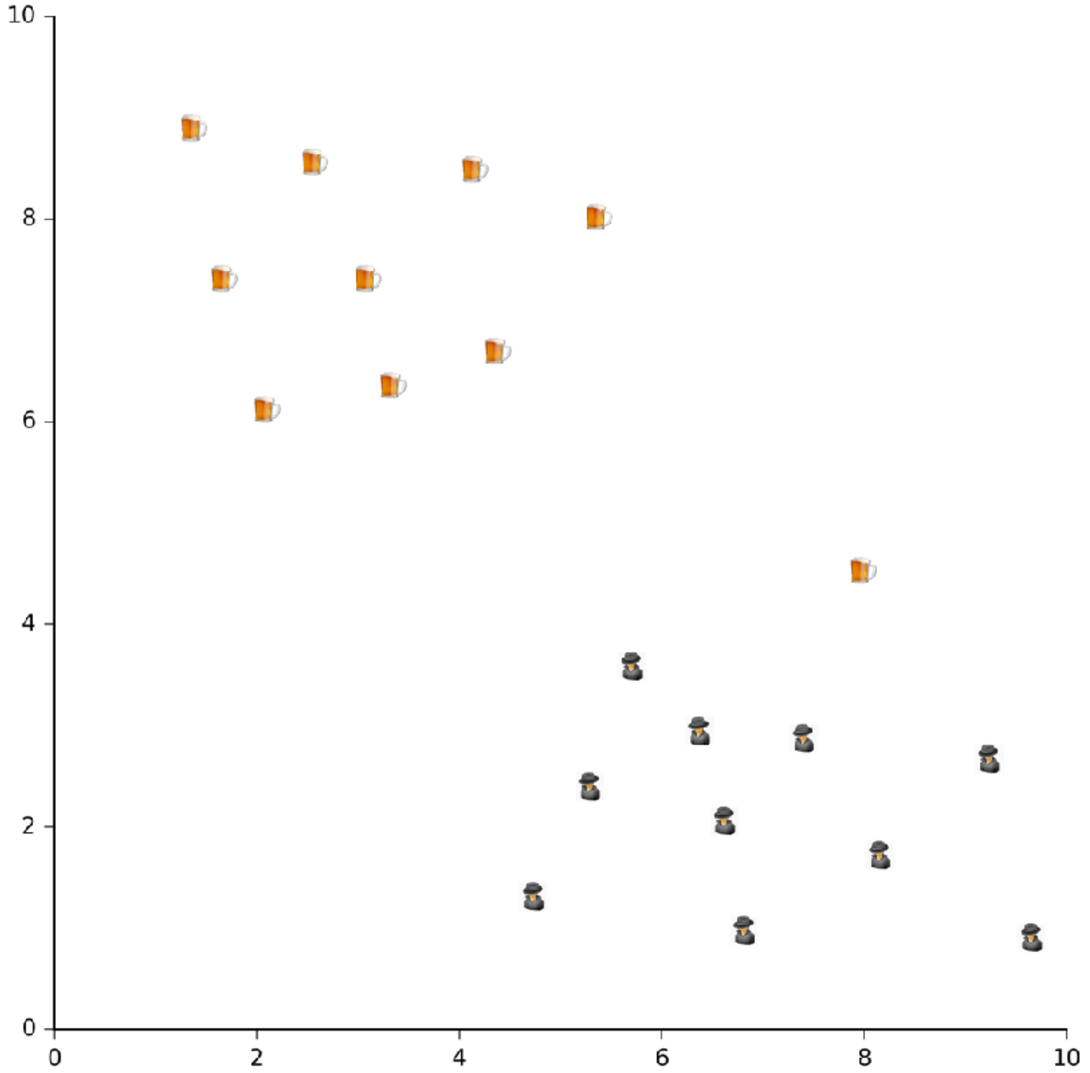
Answer: 2

Open BlackBox…

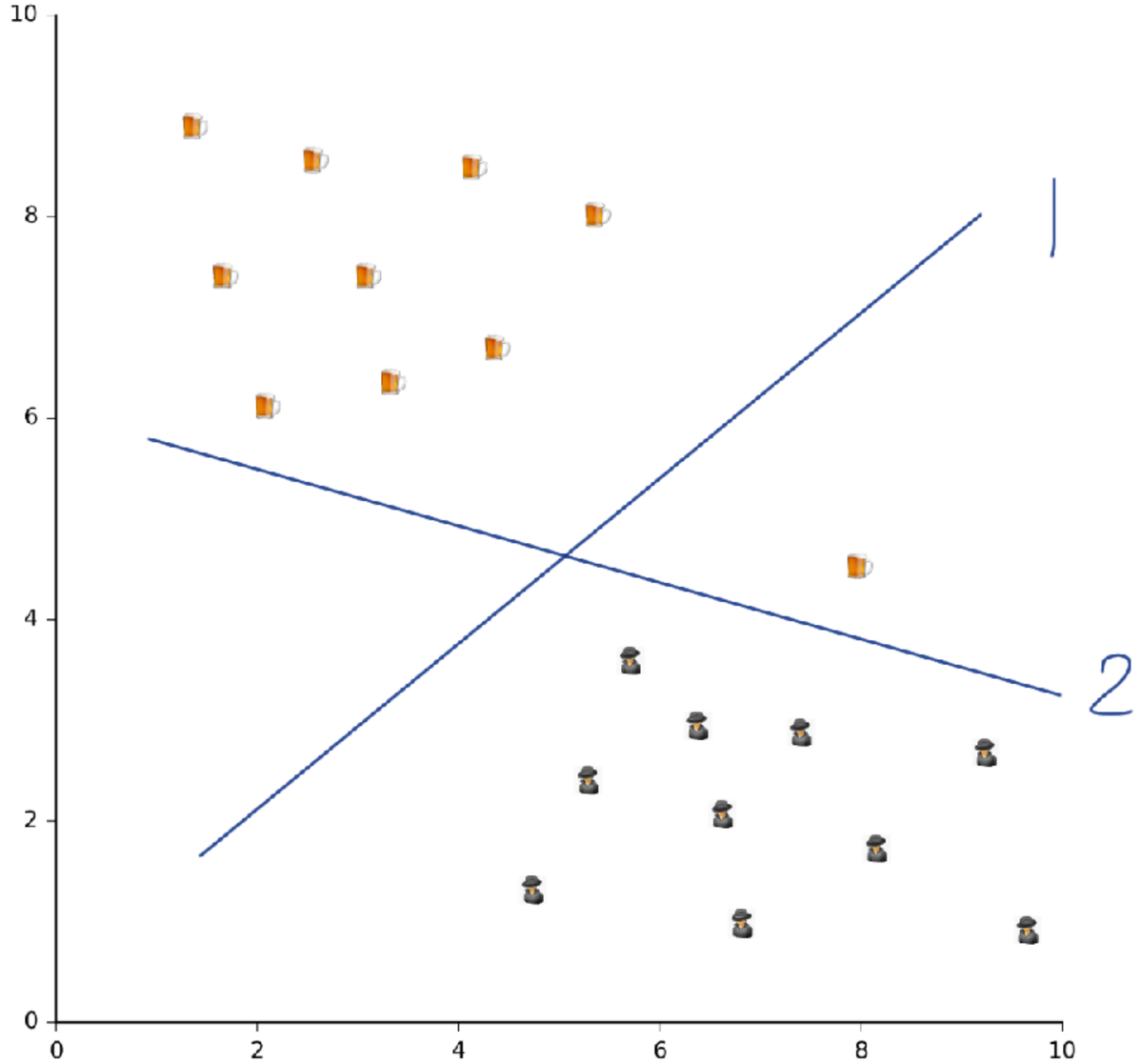




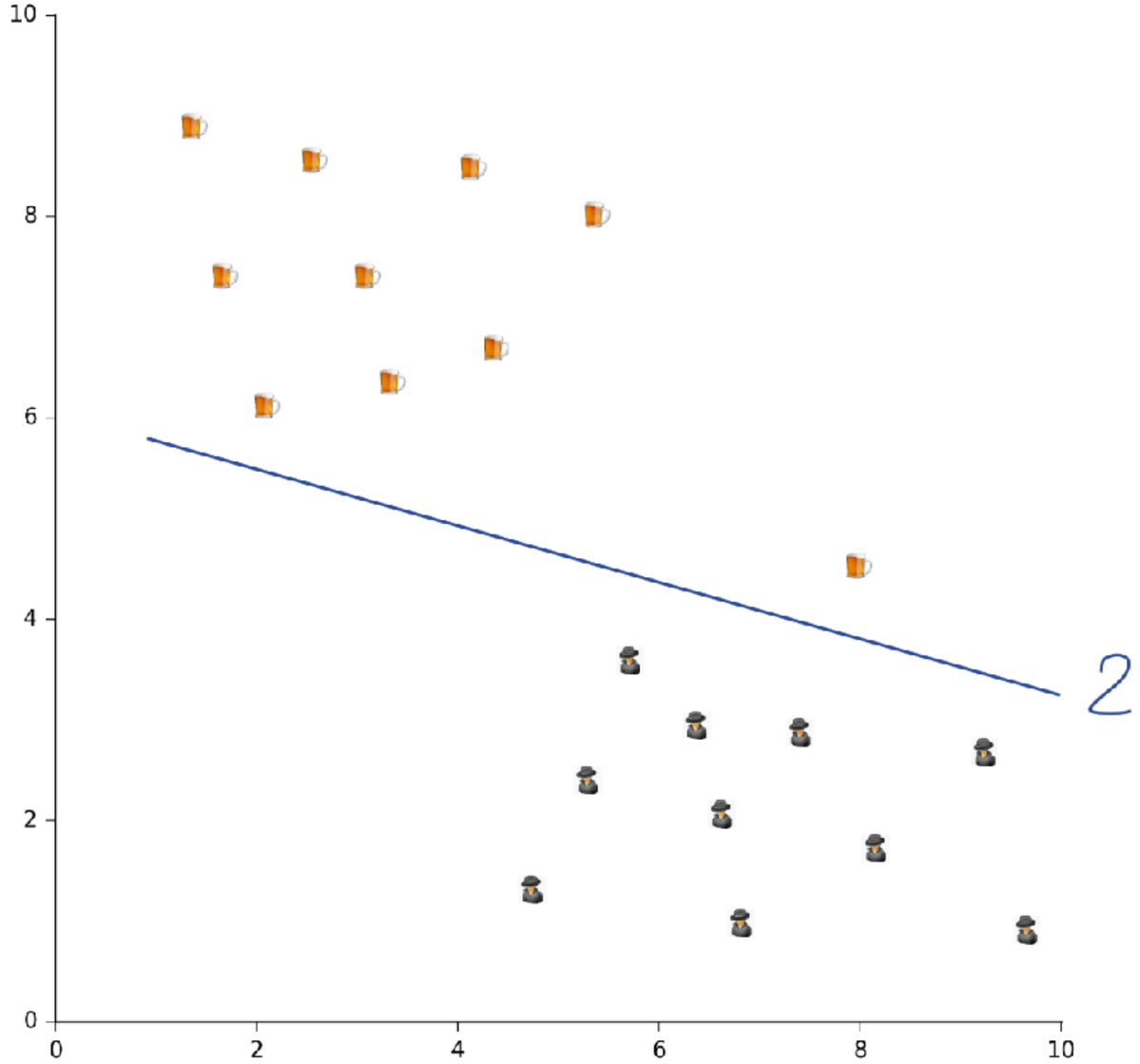
Another test, how do we best separate beer and cyber linearly?



Choose the best line!

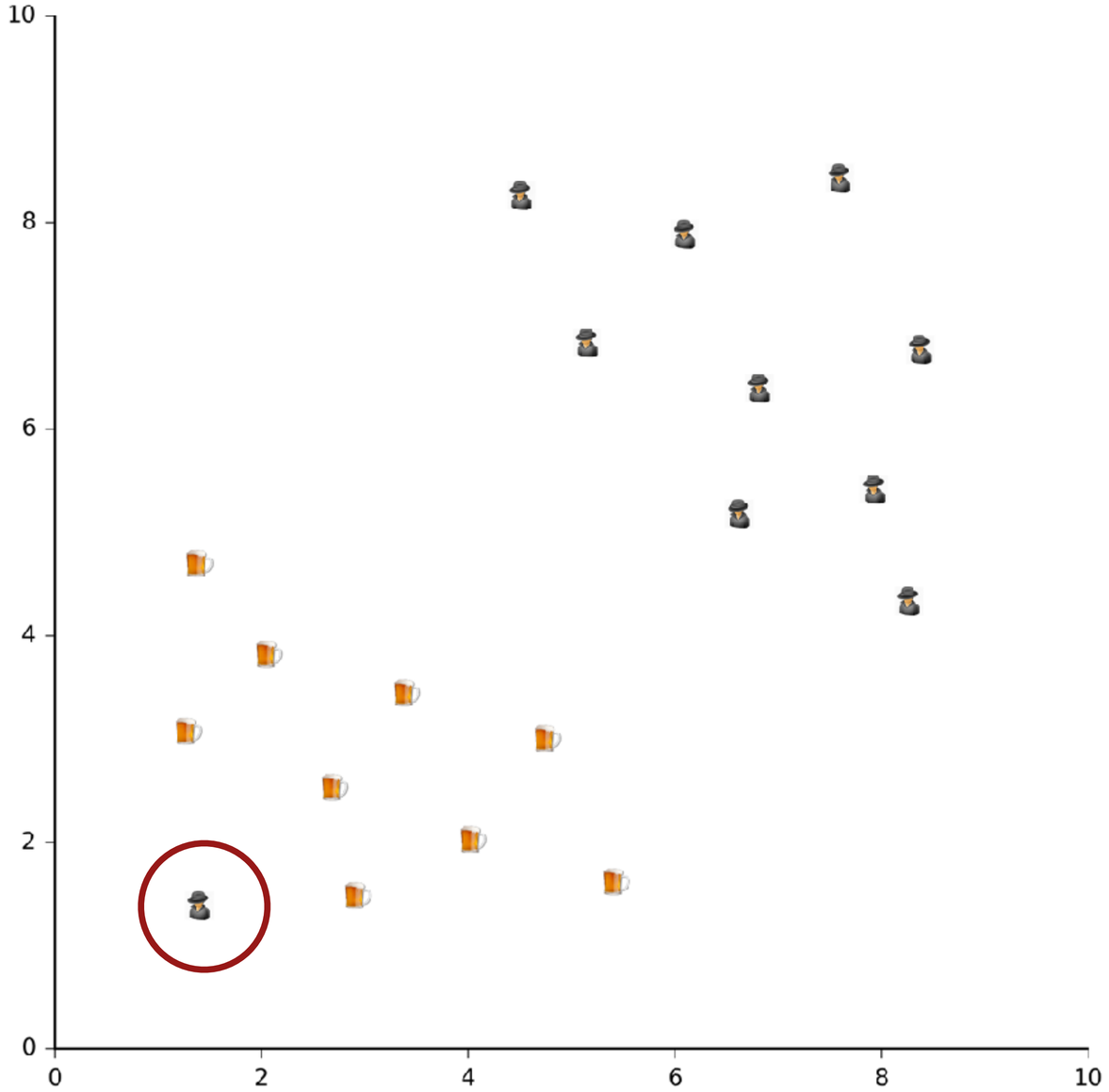


Choose the best line!

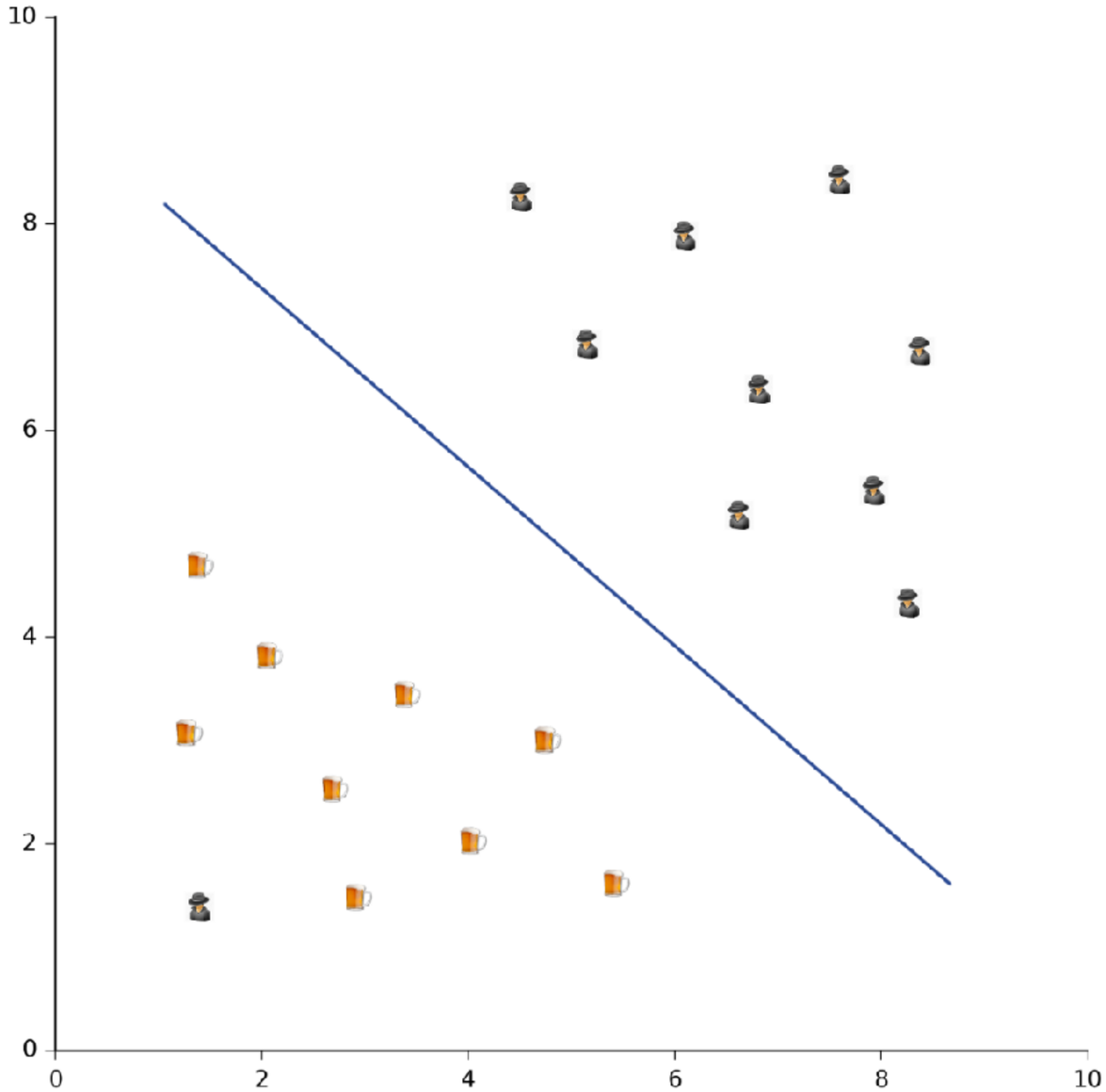


Answer: 2

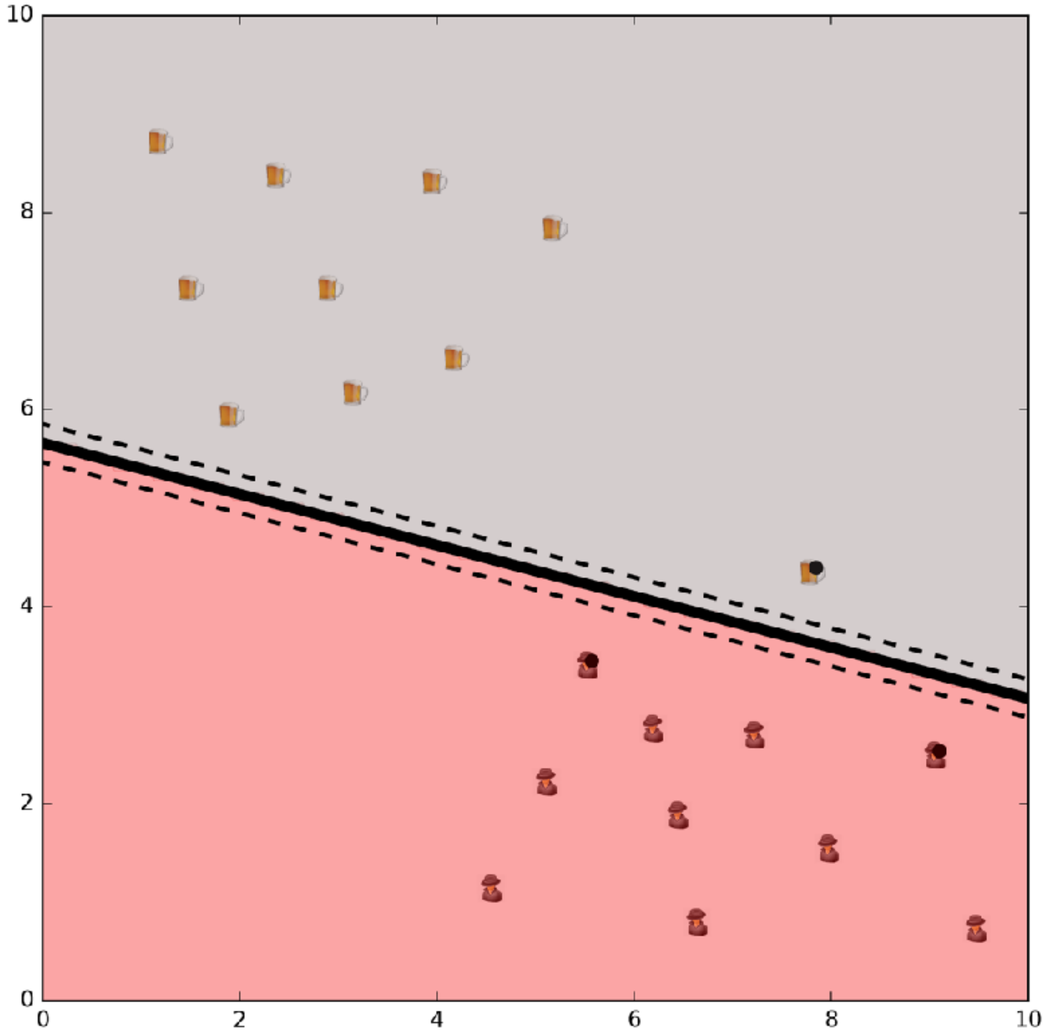
Wait and now?

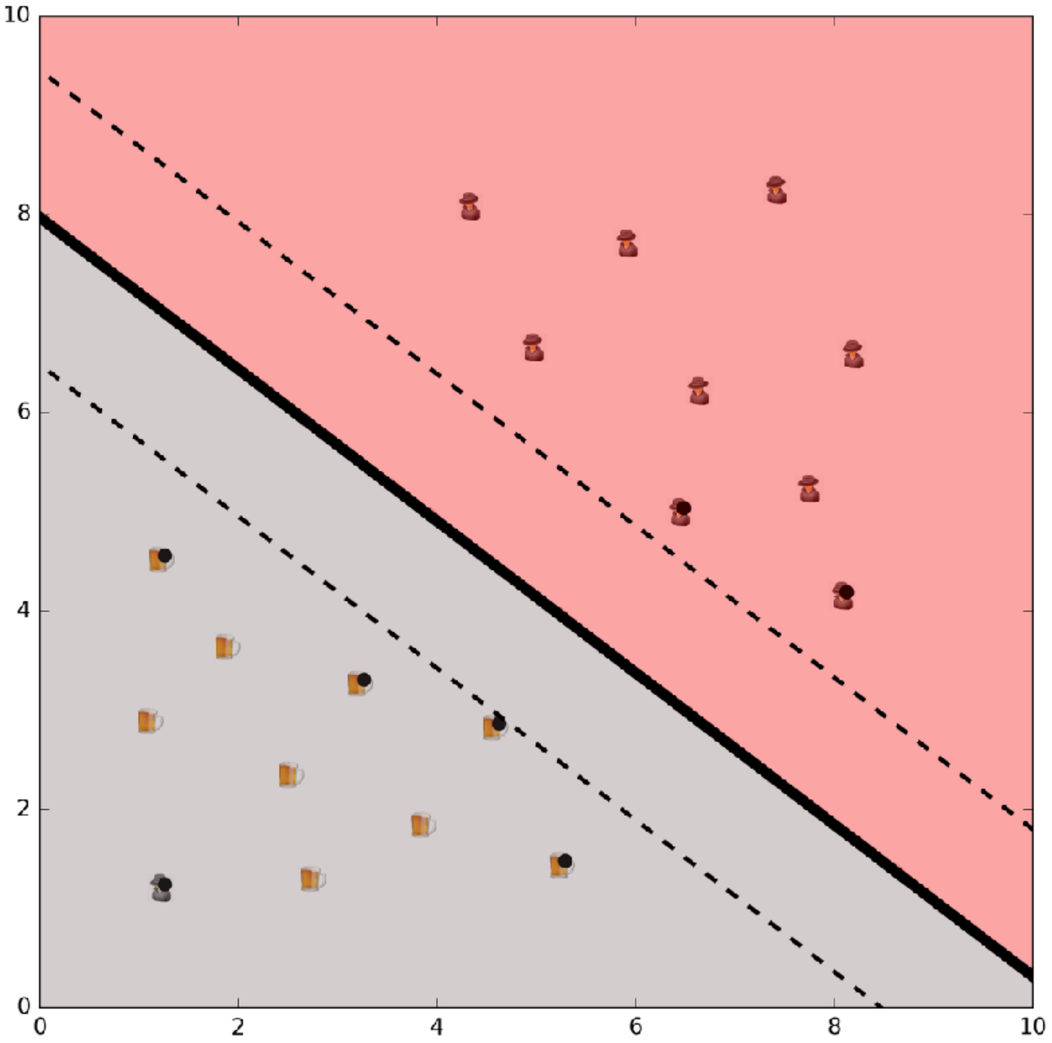


Just ignore?

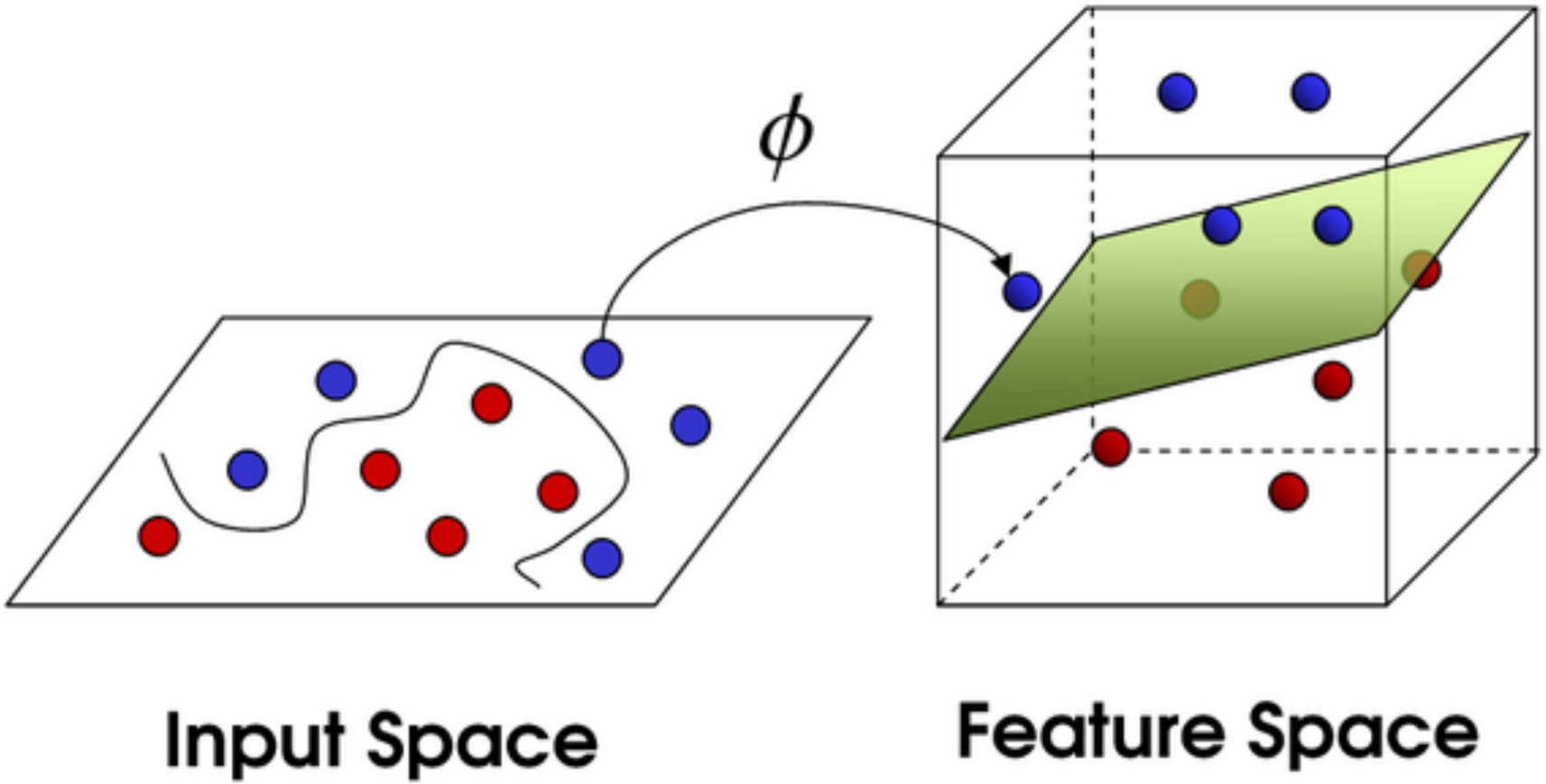


Open BlackBox…



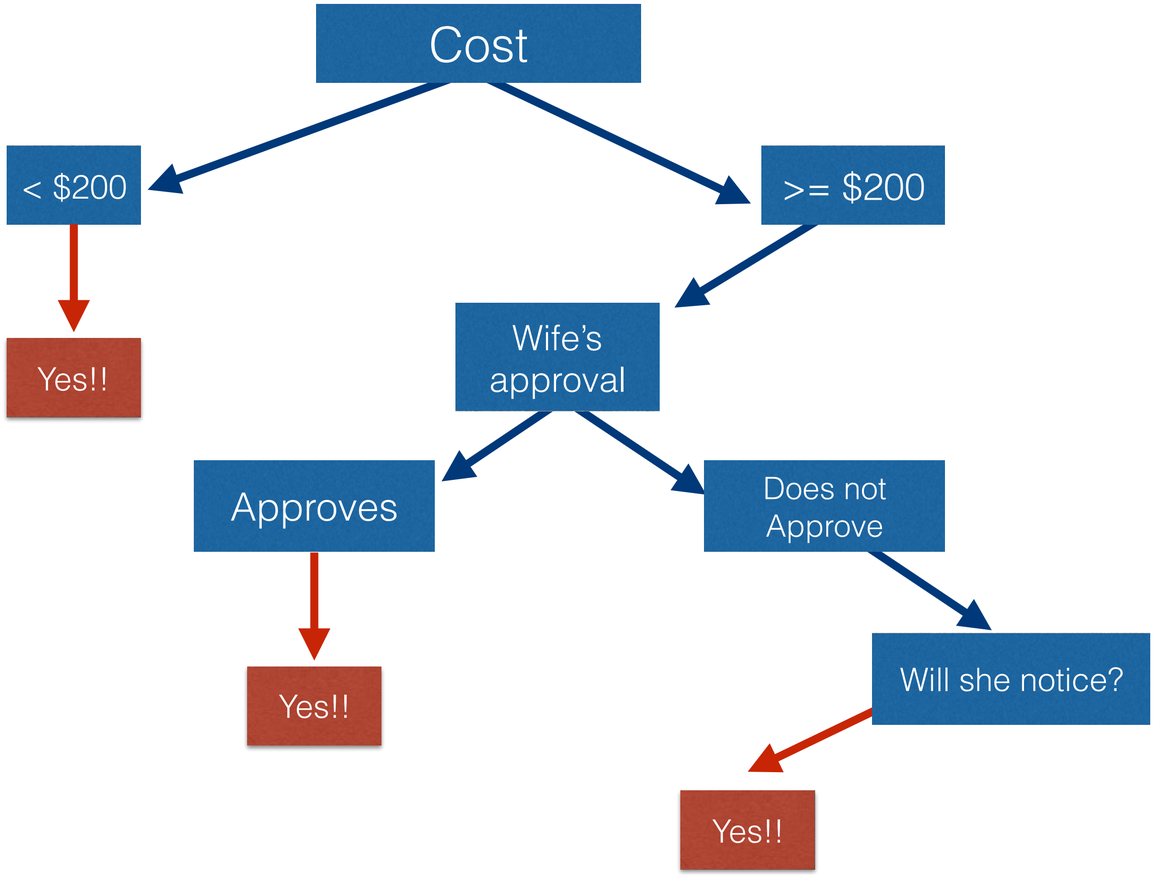


Kernel trick



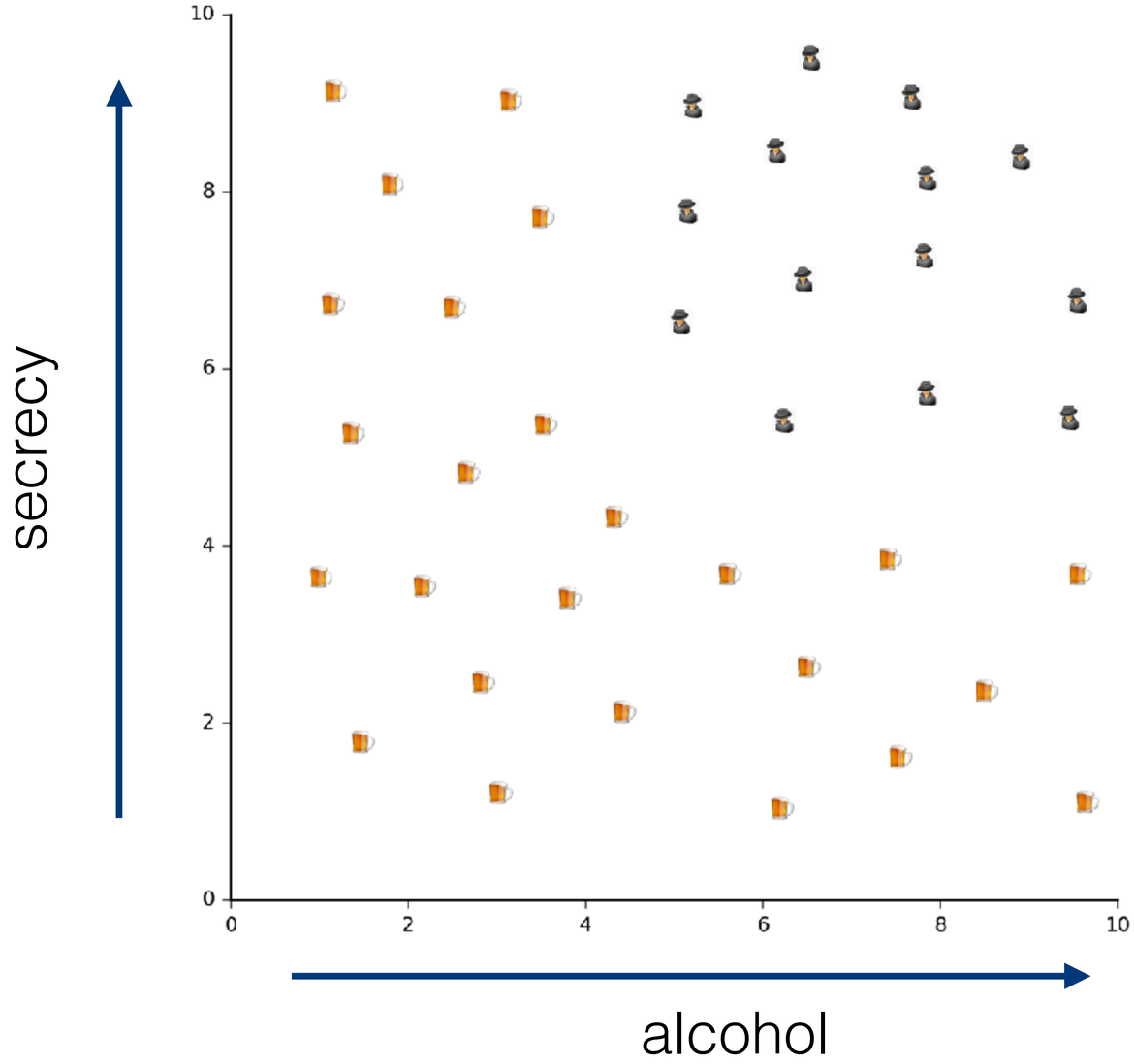
Simple Decision Tree (DT)

Should I buy a new tech gadget?

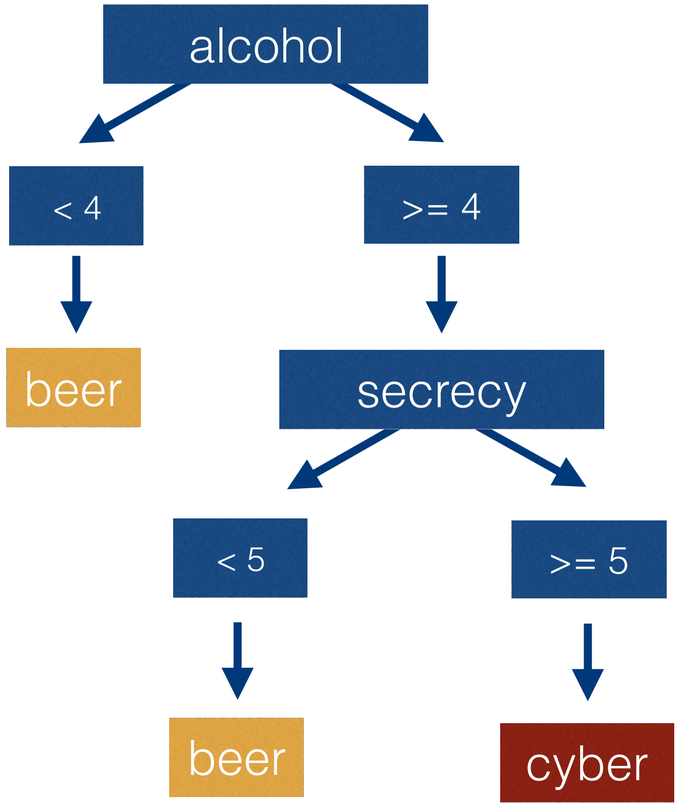


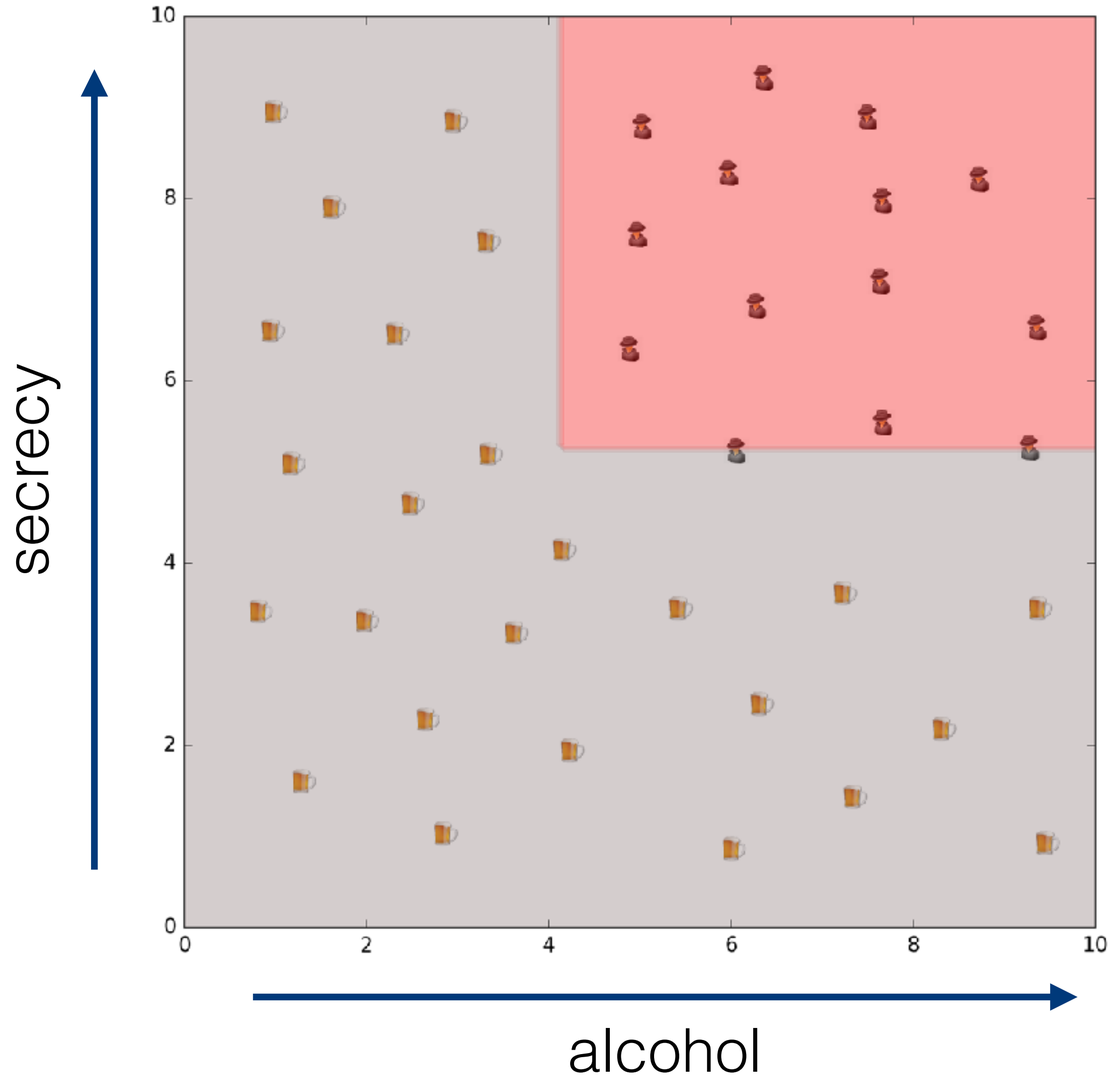
This is Charles' example!

How can we separate beer and cyber?

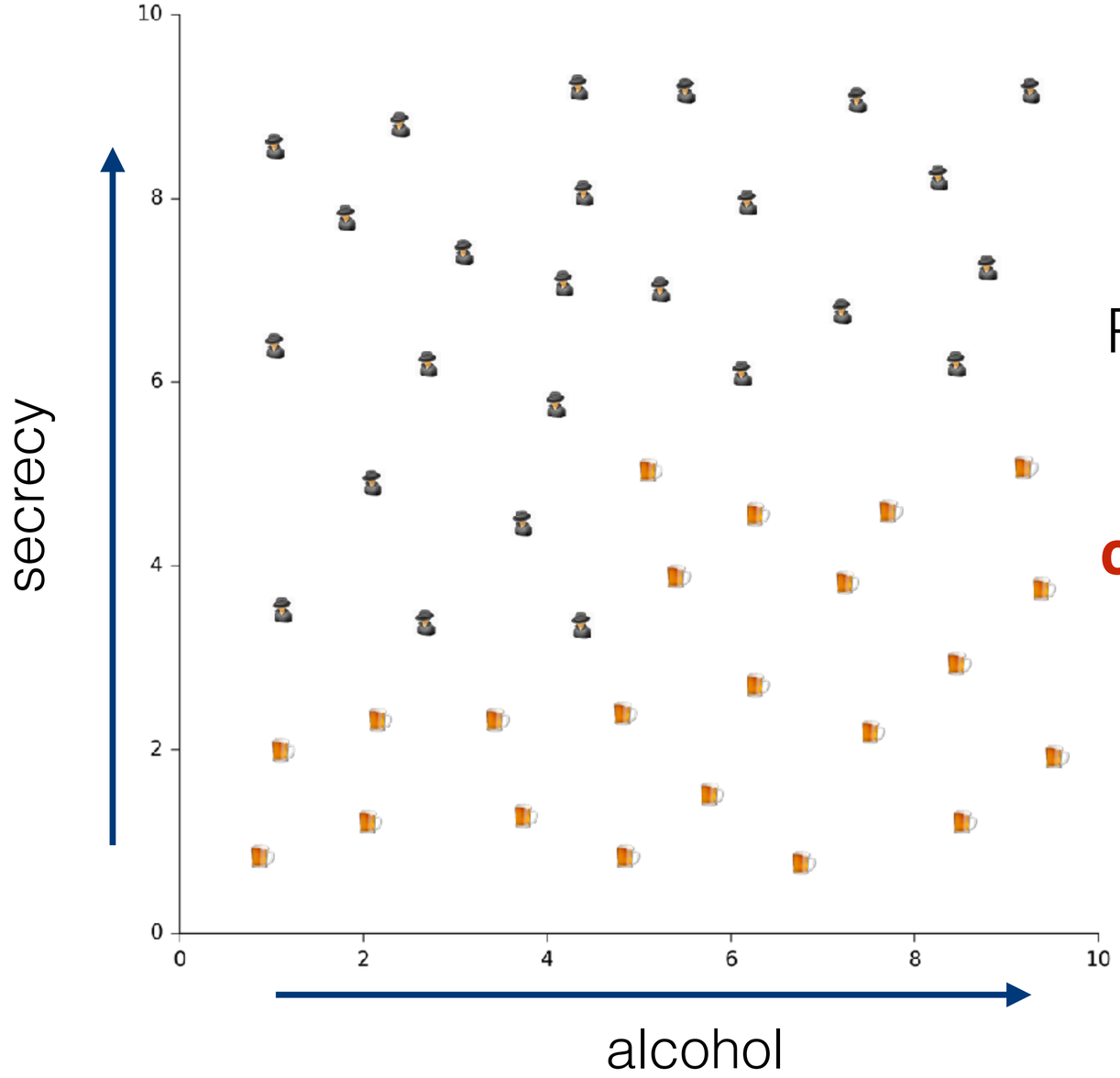


How can we separate beer and cyber?



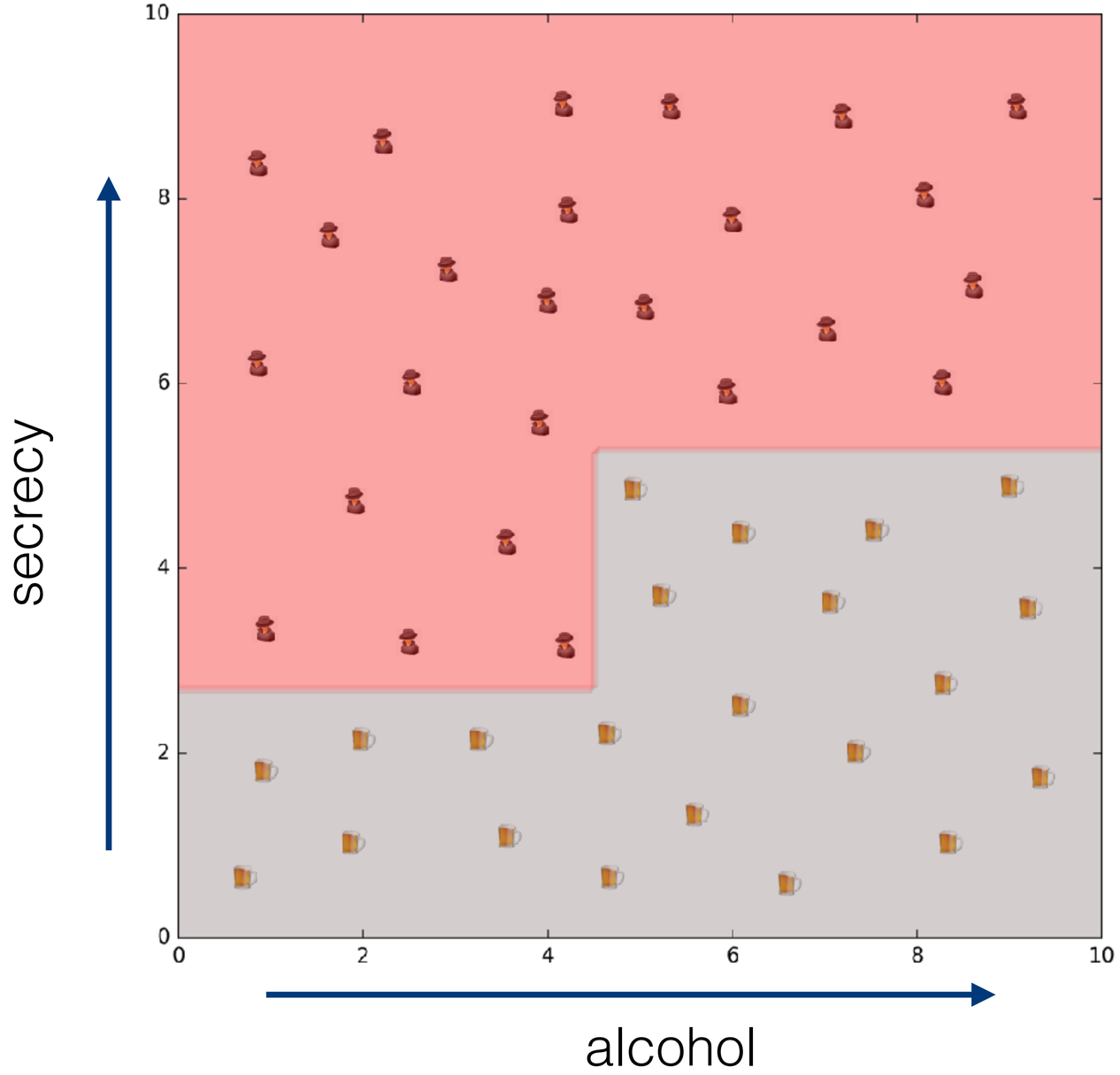


Pen and paper worksheet: solve!



Please take 10 minutes and complete **printed Worksheet - create a Decision Tree by hand!**

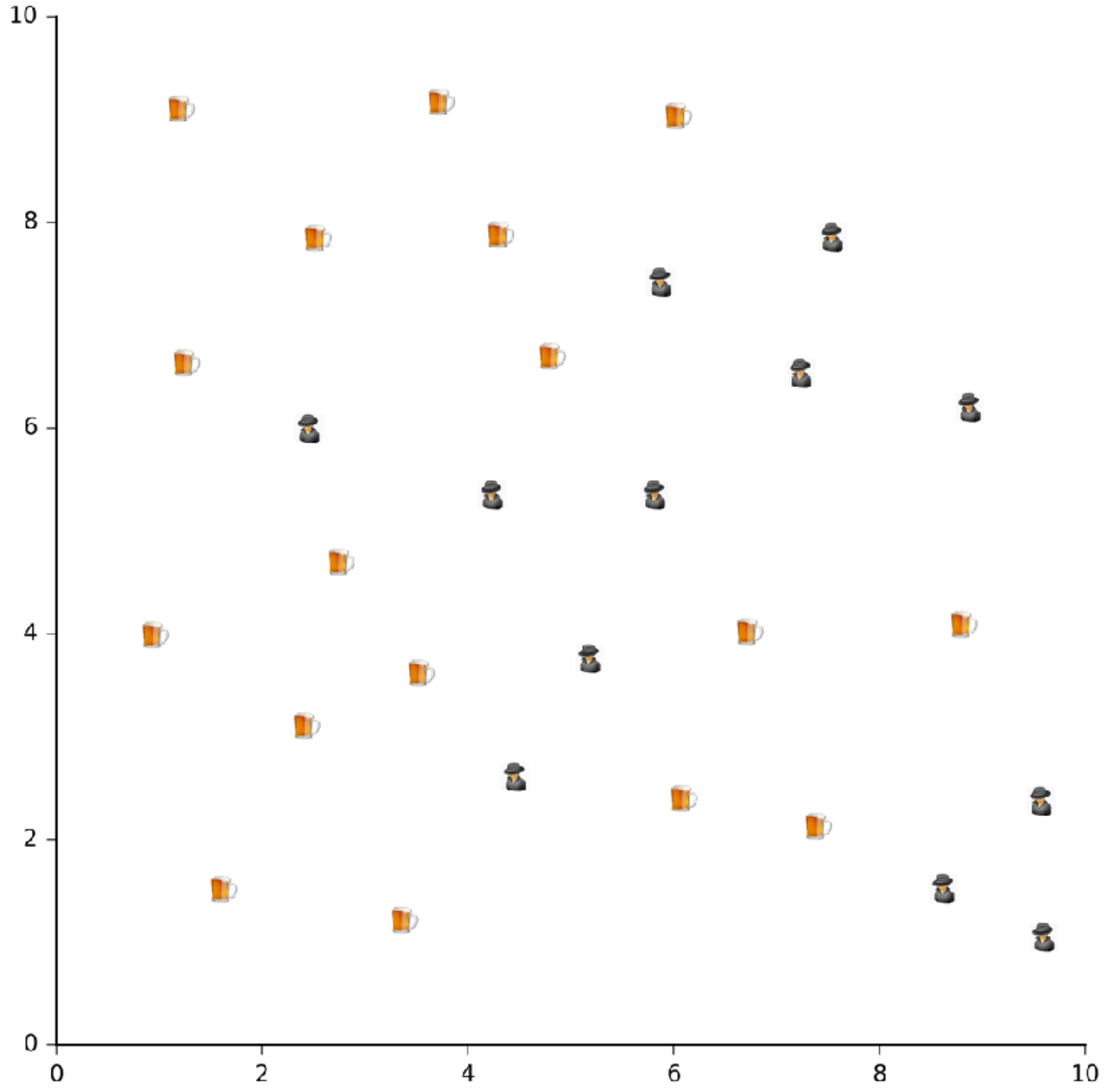
Pen and paper worksheet: solve!



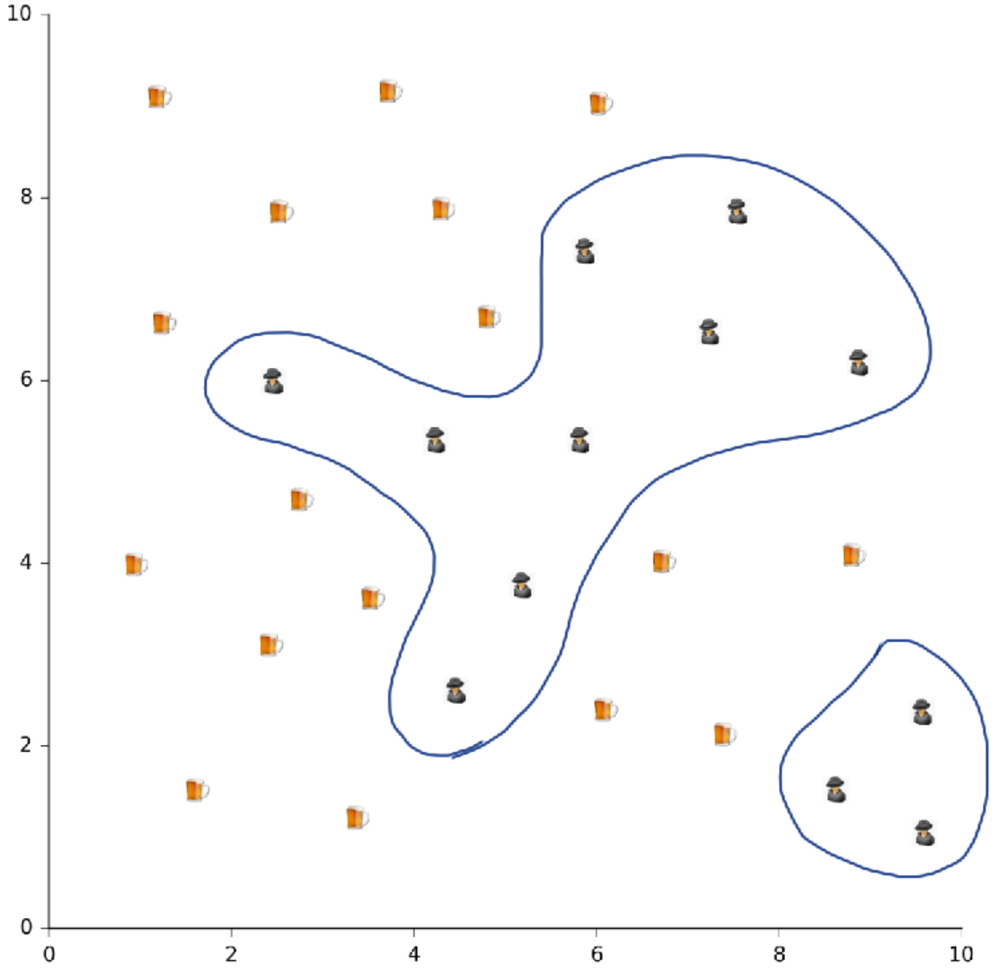
Answer:

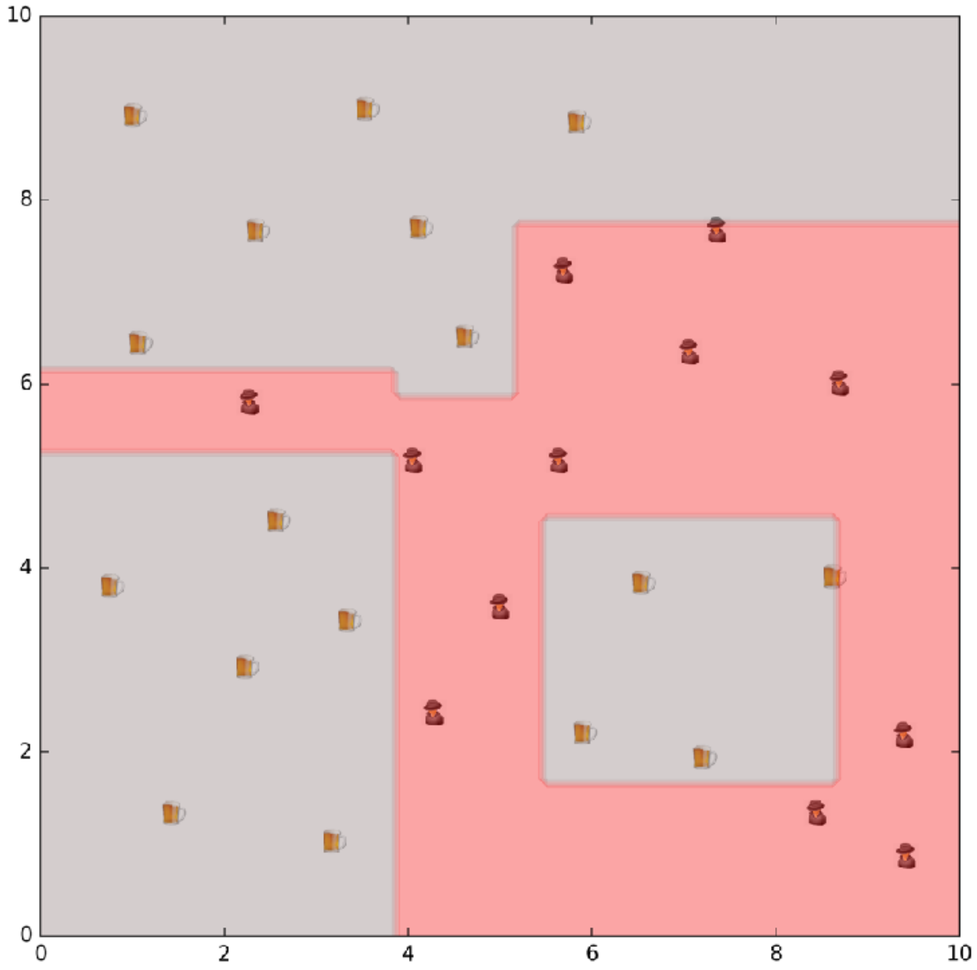
1. secrecy threshold 5.27
2. alcohol threshold 4.47
3. secrecy threshold 2.70

Ultimate last challenge! Who can solve it?



Such boundaries can be a sign of over-fitting!

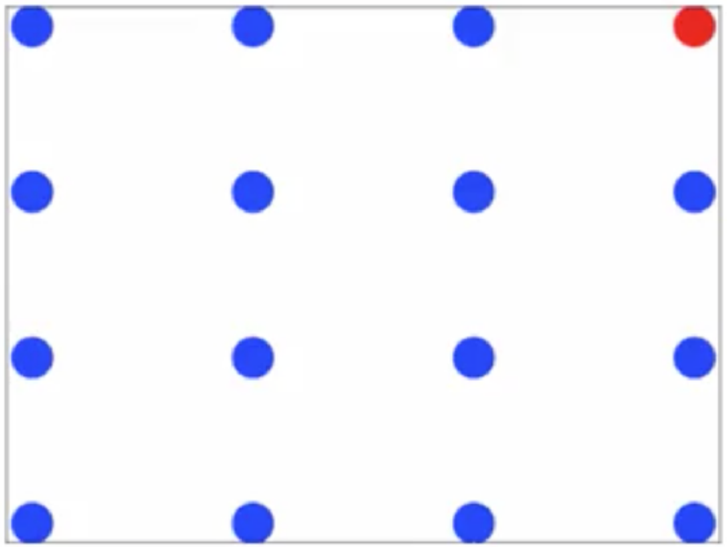




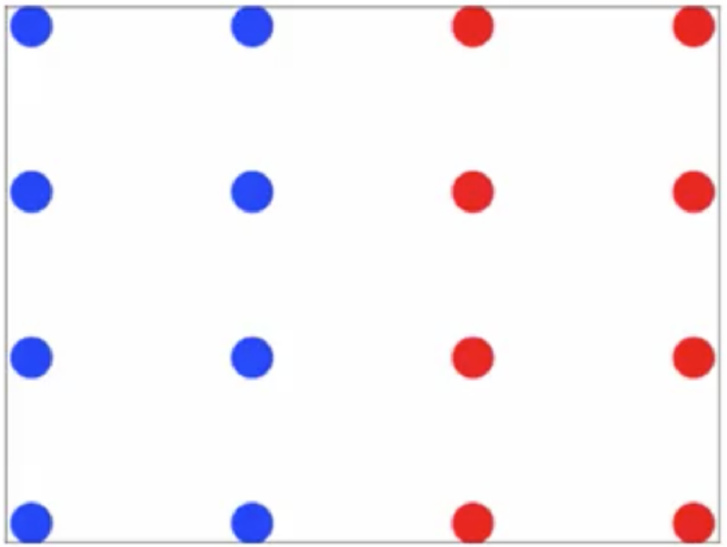
Lastly: How does a Decision Tree exactly decide to split?

Criterion for best splits: **Impurity**

A



B

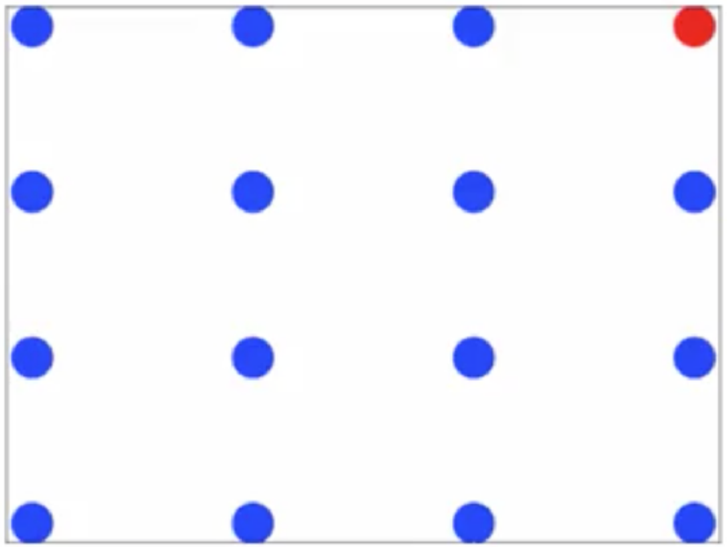


Quiz: By intuition which example appears to be more pure, that is, separates the two classes better?

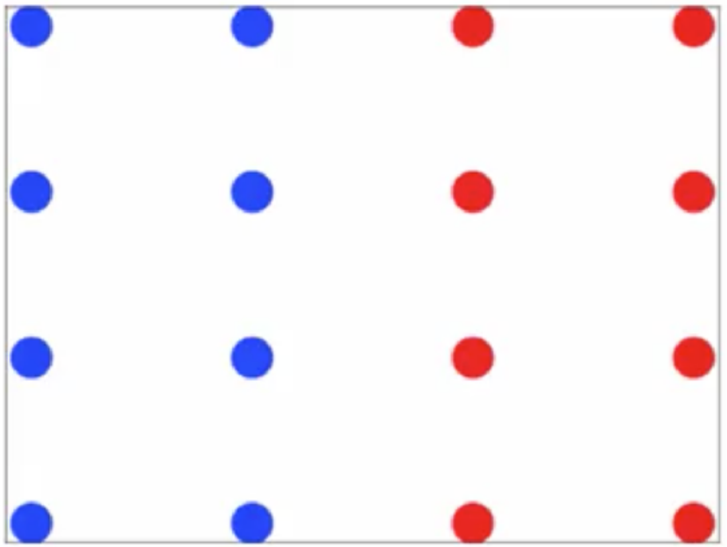
Lastly: How does a Decision Tree exactly decide to split?

Criterion for best splits: **Information Gain**

A



B



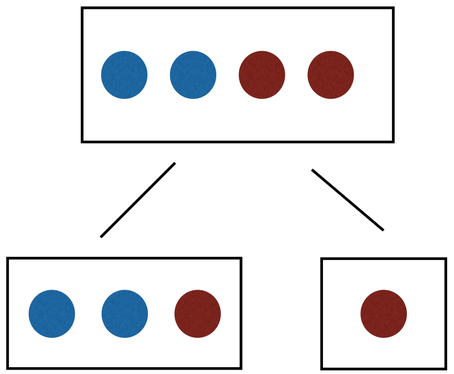
Answer: A

Which feature below gives the best split?

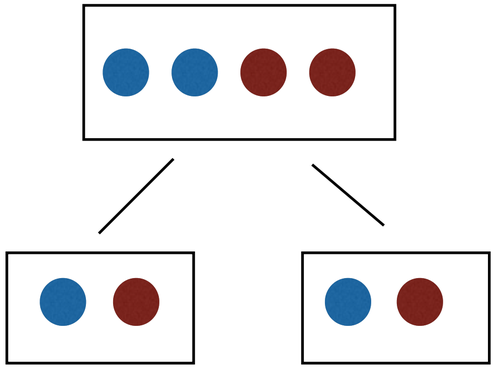
Information Gain = entropy(parent) - [weighted average]entropy(children)

entropy(parent)=1.0
most impure binary system

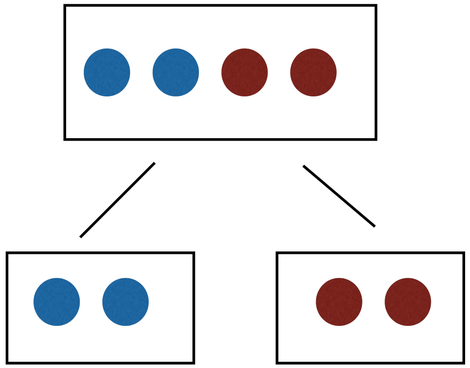
choosing next best feature to split next



feature 1



feature 2



feature 3

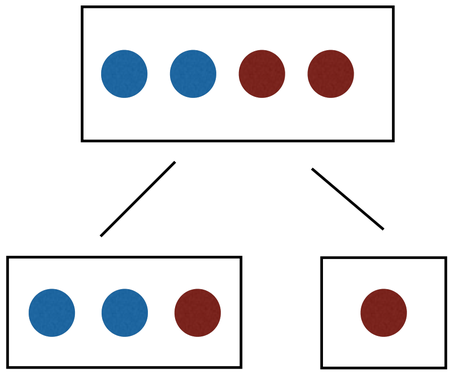
?

Which feature below gives the best split?

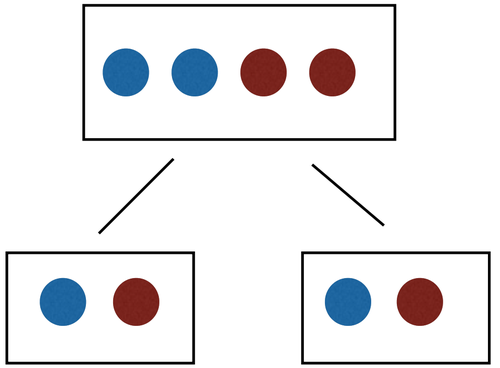
Information Gain = entropy(parent) - [weighted average]entropy(children)

entropy(parent)=1.0
most impure binary system

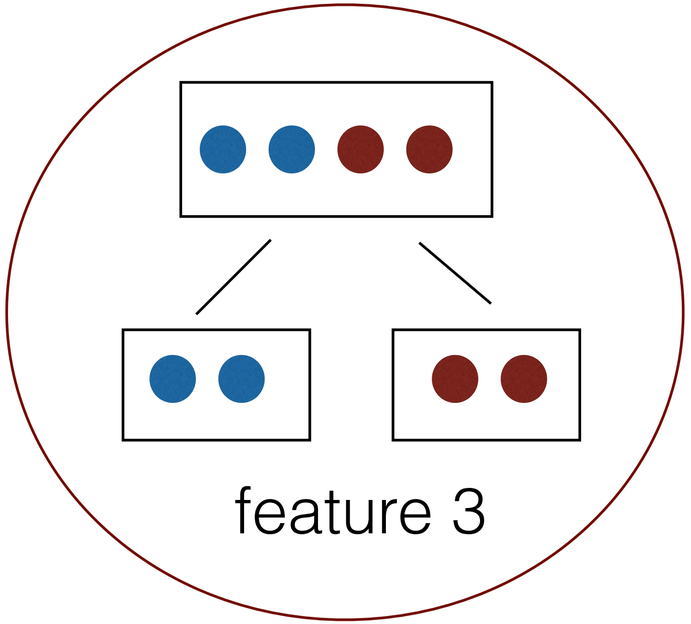
choosing next best feature to split next



feature 1



feature 2



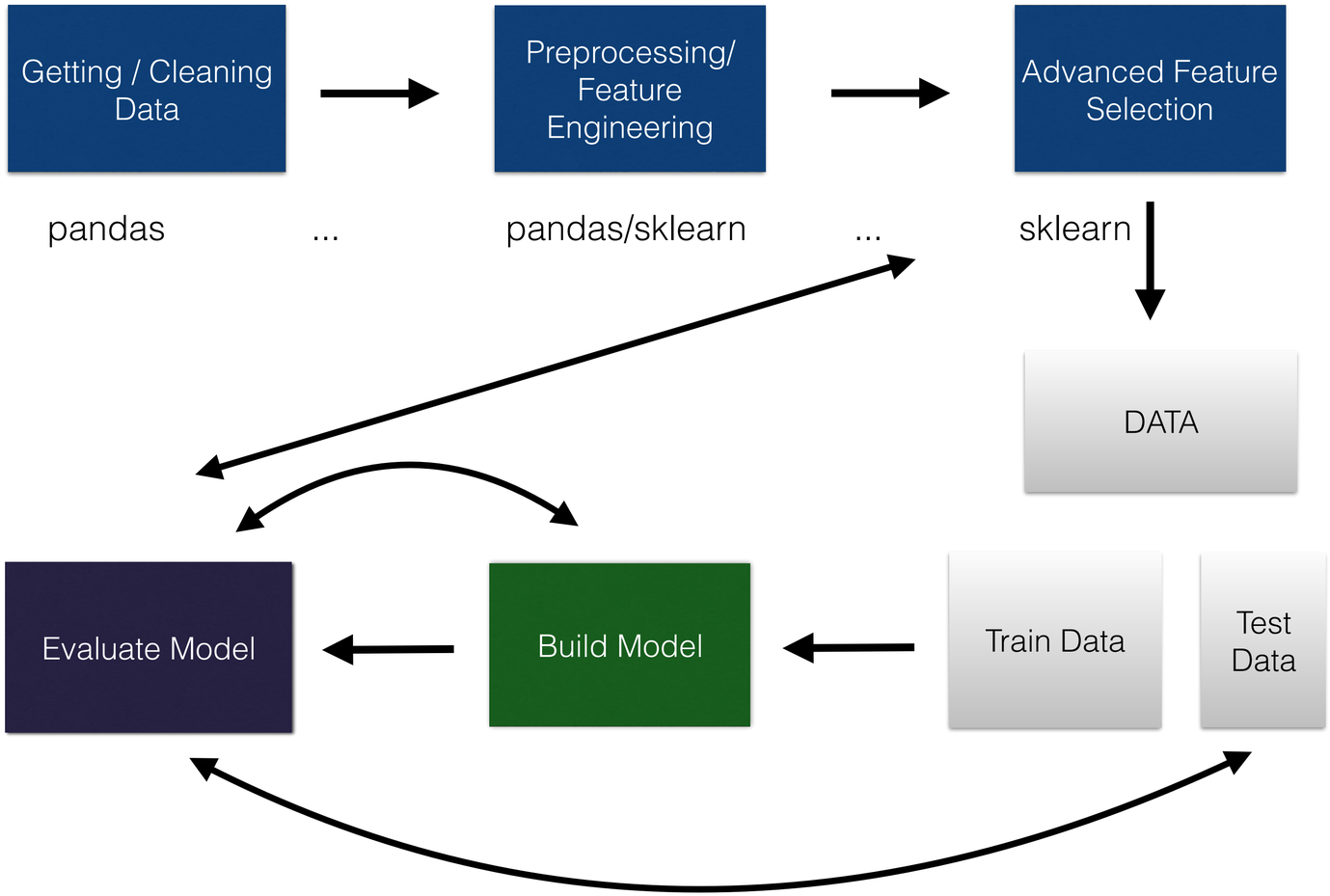
feature 3

Information Gain	**0.3112**	**0**	**1**

Supervised Machine Learning

Hands on train - predict with sklearn

Machine Learning Process



Train-Predict in sklearn

```
## Support Vector Machine Classification

clf = svm.SVC()
clf = clf.fit(X, target)
target_pred = clf.predict(X)
```

clf means classifier

Train-Predict in sklearn

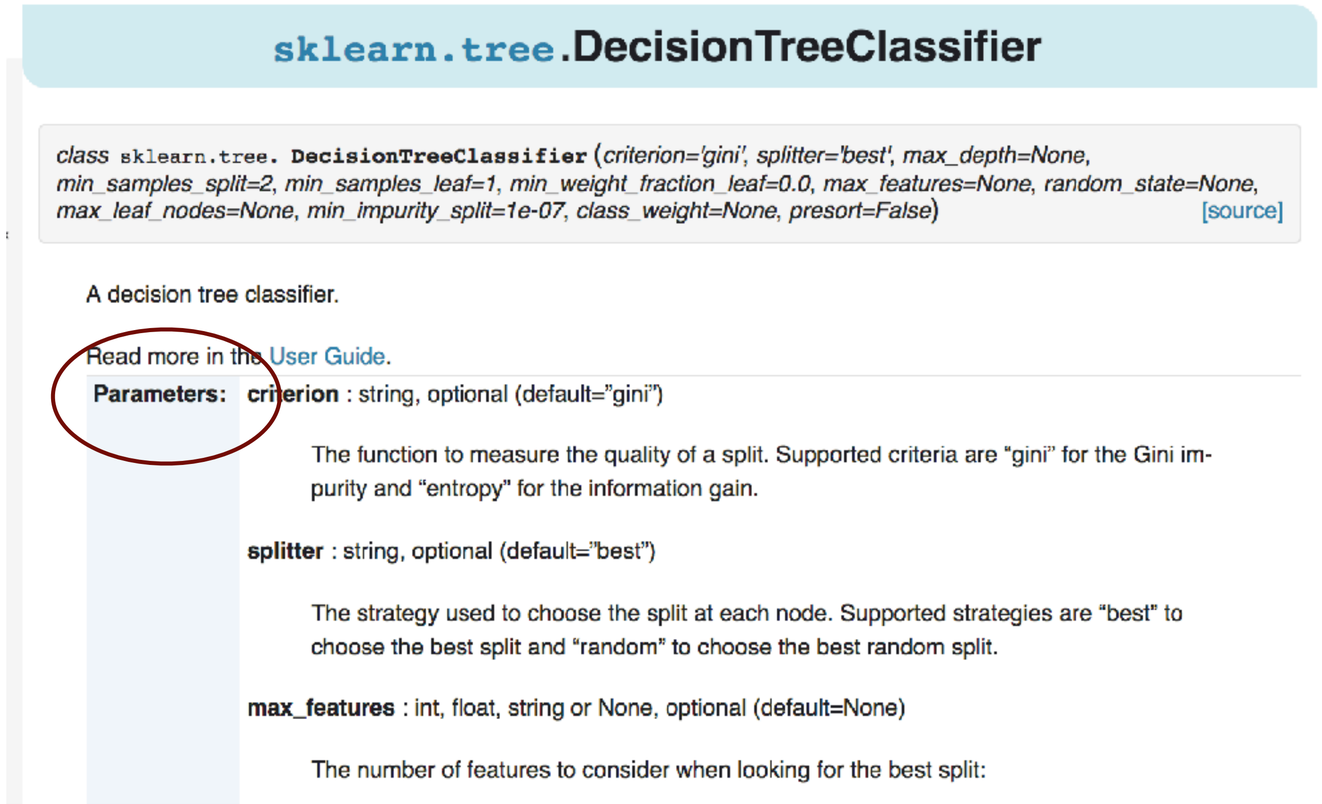
```
## Decision Tree Classification

clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, target)
target_pred = clf.predict(X)
```

clf means classifier

Python Sklearn Overview

http://scikit-learn.org/stable/

Sklearn Classifiers Navigation



sklearn.tree.DecisionTreeClassifier

class sklearn.tree. **DecisionTreeClassifier** (*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_split=1e-07, class_weight=None, presort=False*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters: **criterion** : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter : string, optional (default="best")

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

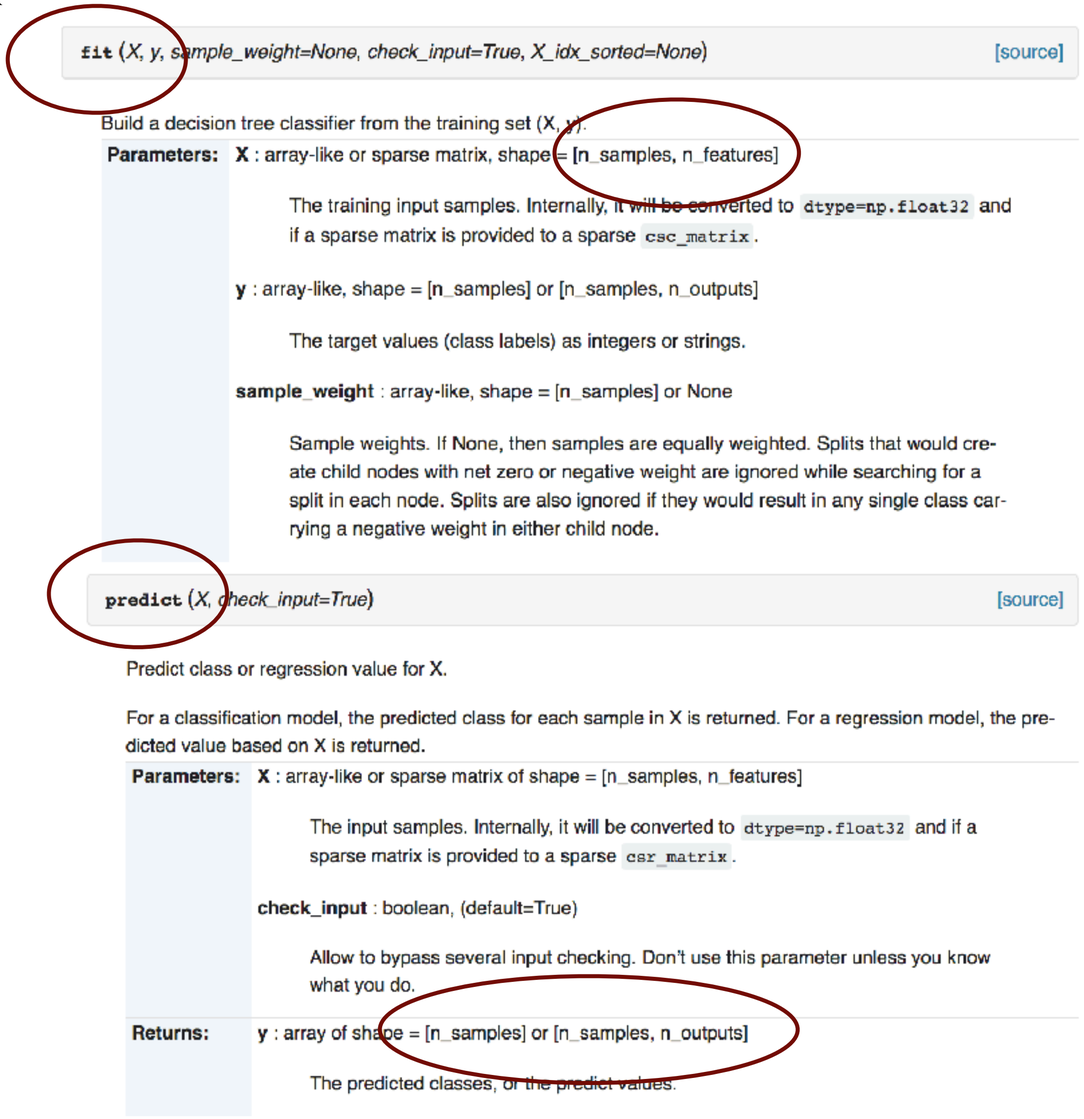
max_features : int, float, string or None, optional (default=None)

The number of features to consider when looking for the best split:

Check parameters for each classifier!

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

Sklearn Classifiers Navigation



fit (*X, y, sample_weight=None, check_input=True, X_idx_sorted=None*) [source]

Build a decision tree classifier from the training set (X, y).

Parameters: **X** : array-like or sparse matrix, shape = [n_samples, n_features]

The training input samples. Internally, it will be converted to `dtype=np.float32` and if a sparse matrix is provided to a sparse `csc_matrix`.

y : array-like, shape = [n_samples] or [n_samples, n_outputs]

The target values (class labels) as integers or strings.

sample_weight : array-like, shape = [n_samples] or None

Sample weights. If None, then samples are equally weighted. Splits that would create child nodes with net zero or negative weight are ignored while searching for a split in each node. Splits are also ignored if they would result in any single class carrying a negative weight in either child node.

predict (*X, check_input=True*) [source]

Predict class or regression value for X.

For a classification model, the predicted class for each sample in X is returned. For a regression model, the predicted value based on X is returned.

Parameters: **X** : array-like or sparse matrix of shape = [n_samples, n_features]

The input samples. Internally, it will be converted to `dtype=np.float32` and if a sparse matrix is provided to a sparse `csr_matrix`.

check_input : boolean, (default=True)

Allow to bypass several input checking. Don't use this parameter unless you know what you do.

Returns: **y** : array of shape = [n_samples] or [n_samples, n_outputs]

The predicted classes, or the predict values.

Check input and output dimensions of fit and predict methods for feature matrix X and target vector y!

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

Sklearn Classifiers Navigation

Attributes:

classes_ : array of shape = [n_classes] or a list of such arrays

The classes labels (single output problem), or a list of arrays of class labels (multi-output problem).

feature_importances_ : array of shape = [n_features]

The feature importances. The higher, the more important the feature. The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance [R245].

max_features_ : int,

The inferred value of max_features.

n_classes_ : int or list

The number of classes (for single output problems), or a list containing the number of classes for each output (for multi-output problems).

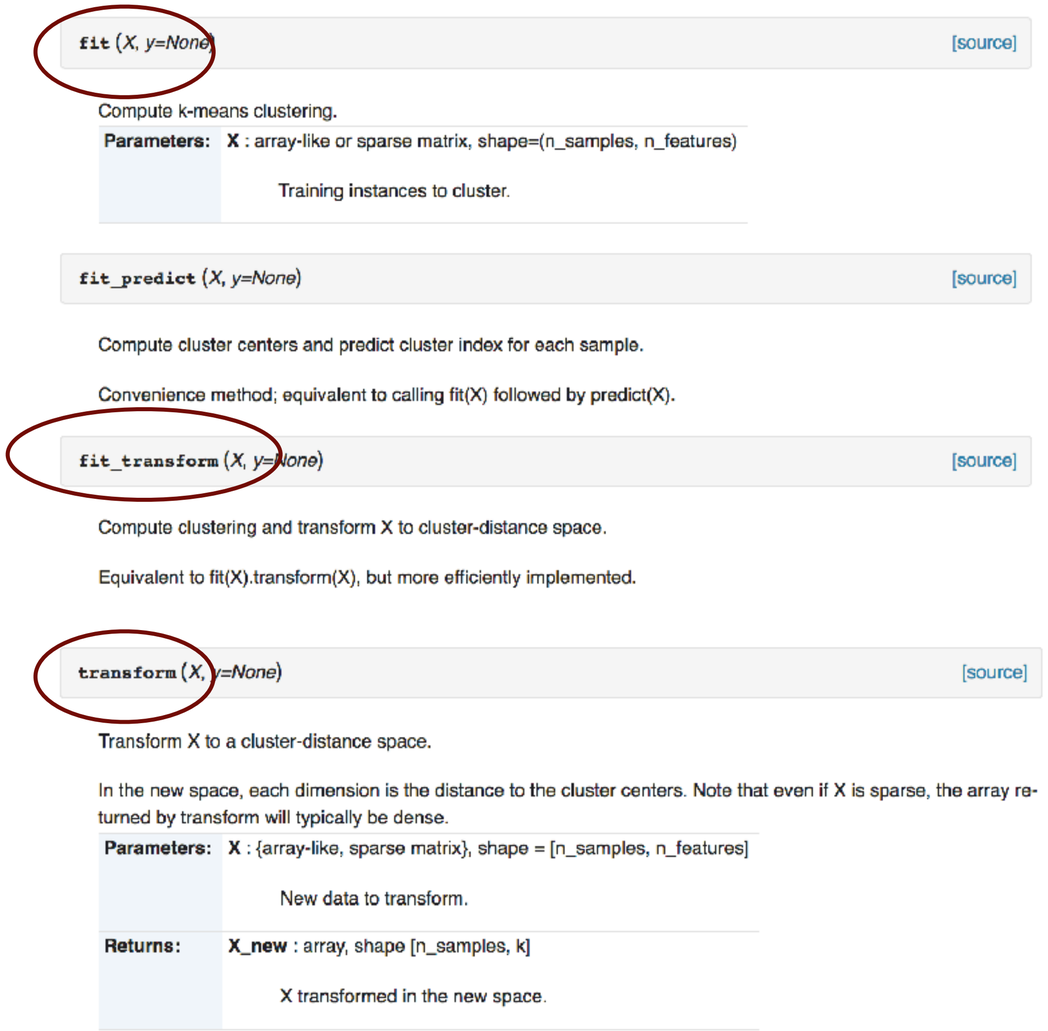
n_features_ : int

The number of features when `fit` is performed.

After calling fit method you can retrieve certain attributes of your clf object!

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

Sklearn Classifiers Navigation



fit (*X*, *y=None*) [source]

Compute k-means clustering.

Parameters: **X** : array-like or sparse matrix, shape=(n_samples, n_features)

Training instances to cluster.

fit_predict (*X*, *y=None*) [source]

Compute cluster centers and predict cluster index for each sample.

Convenience method; equivalent to calling fit(X) followed by predict(X).

fit_transform (*X*, *y=None*) [source]

Compute clustering and transform X to cluster-distance space.

Equivalent to fit(X).transform(X), but more efficiently implemented.

transform (*X*, *y=None*) [source]

Transform X to a cluster-distance space.

In the new space, each dimension is the distance to the cluster centers. Note that even if X is sparse, the array returned by transform will typically be dense.

Parameters: **X** : {array-like, sparse matrix}, shape = [n_samples, n_features]

New data to transform.

Returns: **X_new** : array, shape [n_samples, k]

X transformed in the new space.

Clustering and dimensionality reduction methods like PCA have more fit and transform rather than predict and often combine fit_transform in one method!

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Supervised Machine Learning

Metrics and cross-validation

Performance Metrics Classifier

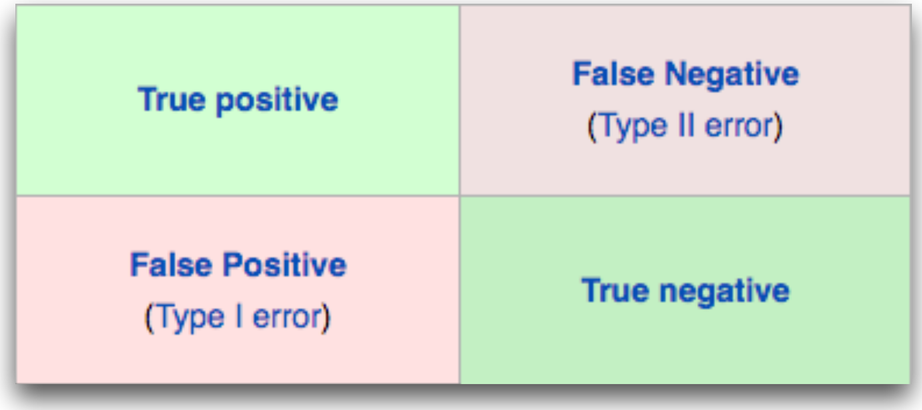
Confusion Matrix

```
target =      [1,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0]
target_pred = [0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0]
print(metrics.confusion_matrix(target, target_pred))
```

True target

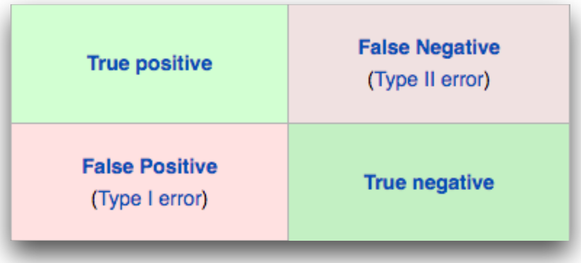
```
[[7 2]
 [3 4]]
```

Predicted target



Accuracy

```
accuracy = metrics.accuracy_score(target, target_pred)
```



True target

0 [[7 2]

1   [3 4]]

0 1

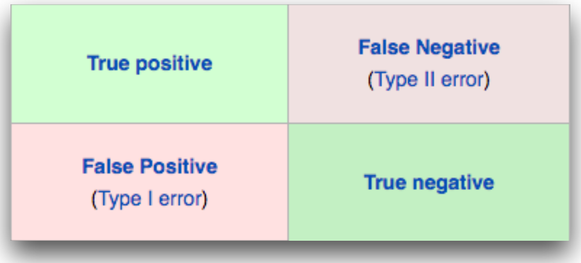
Predicted target

Accuracy = (TP + TN)/ N

Quiz: Calculate by hand!

Accuracy

```
accuracy = metrics.accuracy_score(target, target_pred)
```



True target

0 [[7 2]

1 [3 4]]

0 1

Predicted target

Accuracy = (TP + TN)/ N

= (7+4)/16

= 0.6875

Precision

Column-wise!

Precision

```
precision_class = metrics.precision_score(target, target_pred, average = None)
precision_avg = metrics.precision_score(target, target_pred, average = 'binary')
```

True positive	**False Negative** (Type II error)
False Positive (Type I error)	**True negative**

True target

0 1

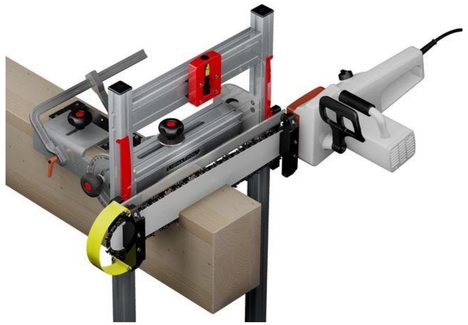
```
[ [ 7  2 ]
  [ 3  4 ] ]
```

0 1

Predicted target

Precision = TP/ (TP + FP)

Quiz: Calculate precision by hand for both classes!



Precision

```
precision_class = metrics.precision_score(target, target_pred, average = None)
precision_avg = metrics.precision_score(target, target_pred, average = 'binary')
```

True positive	**False Negative** (Type II error)
False Positive (Type I error)	**True negative**

True target

0 [[7 2]

1 [3 4]]

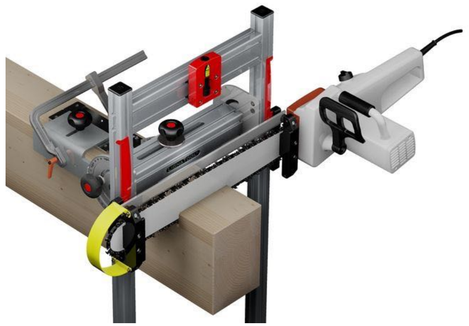
0 1

Predicted target

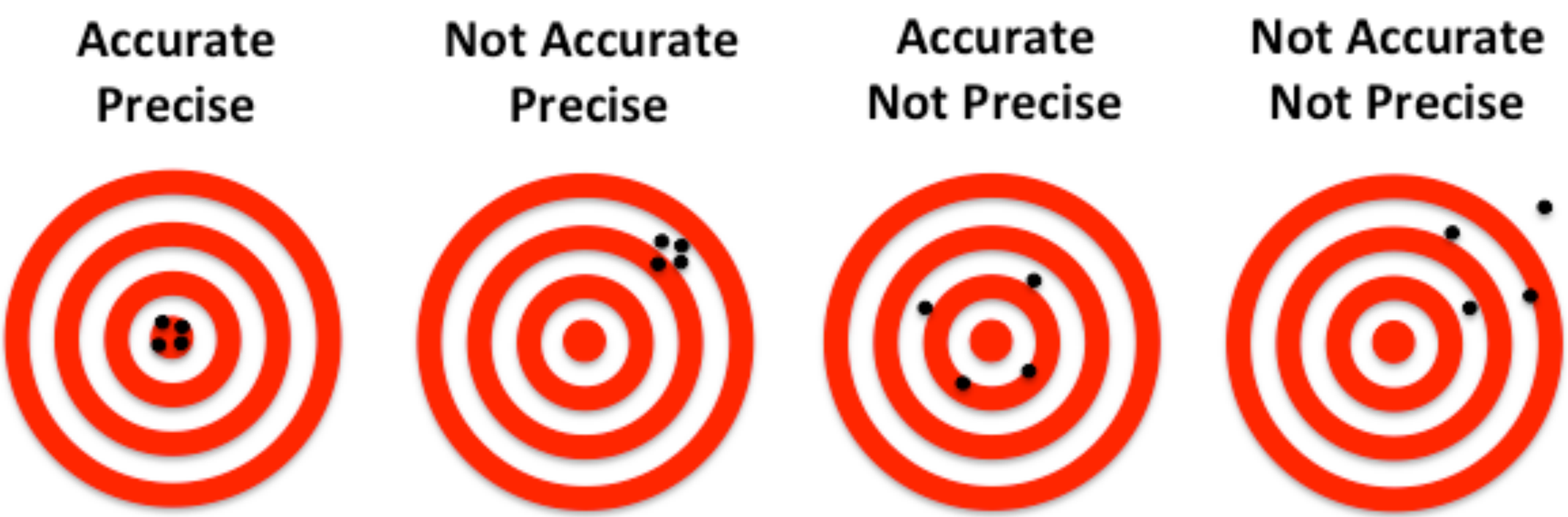
Precision = TP/ (TP + FP)

Prec_Class0 = 7/(7+3) = 0.7

Prec_Class1 = 4/(4+2) = 0.666



Precision

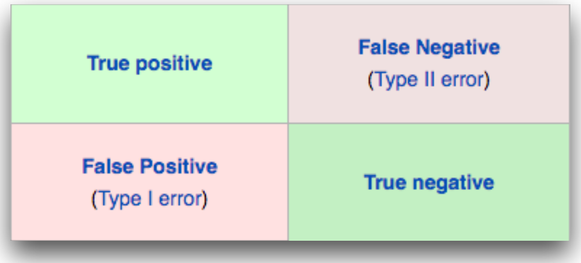

Accurate
Precise
Not Accurate
Precise
Accurate
Not Precise
Not Accurate
Not Precise

Recall

Row-wise!

Recall

```
recall_class = metrics.recall_score(target, target_pred, average = None)
recall_avg = metrics.recall_score(target, target_pred, average = 'binary')
```



True target

0 [[7 2]

1   [3 4]]

0 1

Predicted target

Recall = TP/ (TP + FN)
Quiz: Calculate recall by hand for both classes!

Recall

```
recall_class = metrics.recall_score(target, target_pred, average = None)
recall_avg = metrics.recall_score(target, target_pred, average = 'binary')
```

True positive	**False Negative** (Type II error)
False Positive (Type I error)	**True negative**

True target

0 1

```
[[7 2]
 [3 4]]
```

0 1

Predicted target

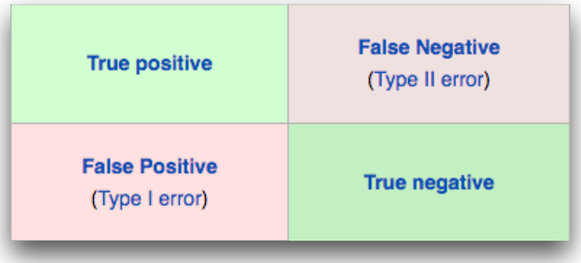
Recall = TP/ (TP + FN)

Rec_Class0 = 7/(7+2) = 0.777

Rec_Class1 = 4/(4+3) = 0.571

Quiz: compute metrics for each class of 3-class confusion matrix

```
target =      [1,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,1,1,2,0,0,0,0,0]
target_pred = [0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,2,2,1,2,2,2,2,2]
print(metrics.confusion_matrix(target, target_pred))
```



True target

```
[[7  2  5]
 [3  4  2]
 [0  1  4]]
```

Predicted target

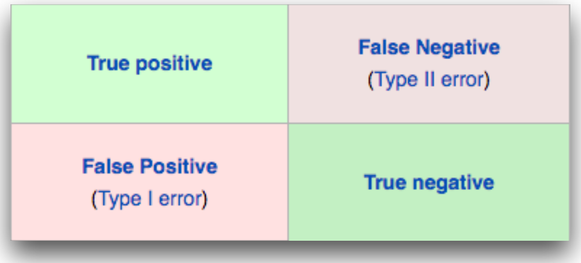
Accuracy = (Sum Diagonal)/ N

Precision = TP/ (TP + FP)

Recall = TP/ (TP + FN)

Quiz: compute metrics for each class of 3-class confusion matrix

```
target =      [1,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,1,1,2,0,0,0,0,0]
target_pred = [0,1,1,0,0,1,1,1,1,0,0,0,0,0,0,0,2,2,2,2,2,2,1,2,2,2,2,2]
print(metrics.confusion_matrix(target, target_pred))
```



True target

```
[[7  2  5]
 [3  4  2]
 [0  1  4]]
```

Predicted target

Accuracy = (7+4+4)/28 = 0.5357
Prec_Class2 = 4/(4 + 5 + 2) = 0.3636
Rec_Class2 = 4/(4 + 1 + 0) = 0.8
…

```
Precision:  [ 0.7  0.57142857  0.36363636]
Recall:  [ 0.5  0.44444444  0.8]
```

Where is the biggest crime scene?

Confusion matrices all with equal accuracy 0.6875!!!
How about precision and recall?

```
[[10  0]
 [ 5  1]]
```

A

```
[[7 2]
 [3 4]]
```

B

```
[[0  4]
 [1 11]]
```

C

Where is the biggest crime scene?

Confusion matrices all with equal accuracy 0.6875!!!
How about precision and recall?

```
Precision:  [ 0.7  1]
Recall:  [ 1  0.2]

[[10  0]
 [5  1]]
```

A

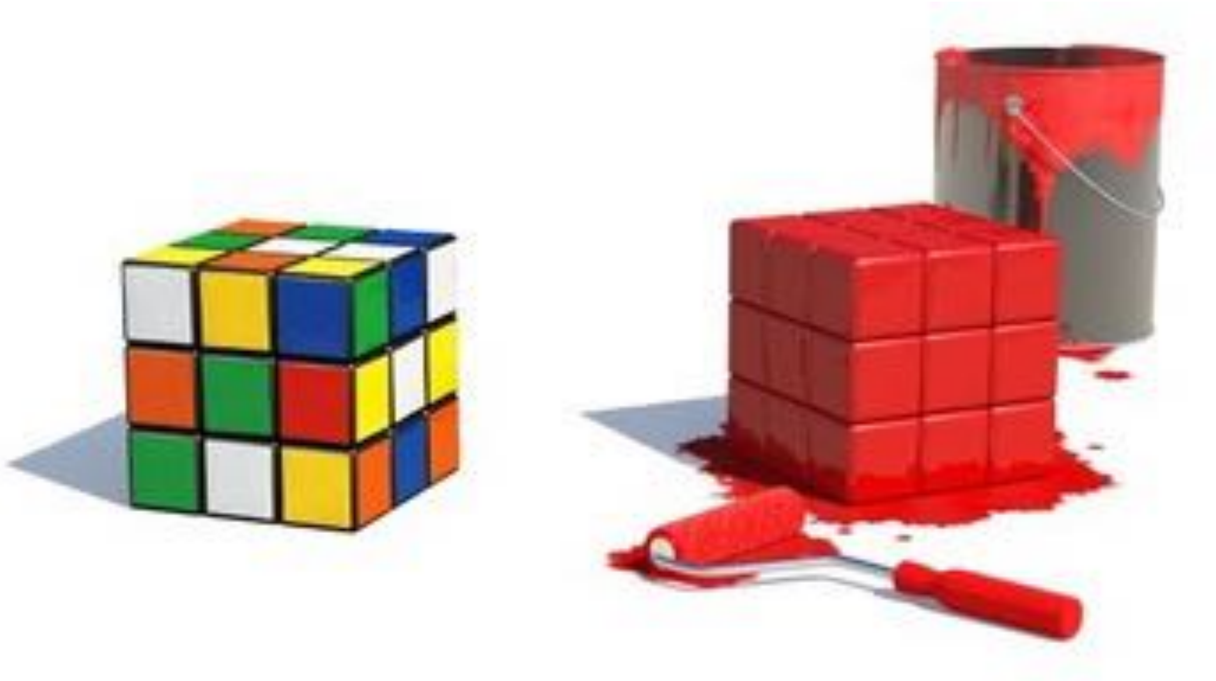
```
Precision:  [ 0.7  0.7]
Recall:  [ 0.8  0.6]

[[7  2]
 [3  4]]
```

B

```
Precision:  [ 0  0.7]
Recall:  [ 0  0.9]

[[0  4]
 [1  11]]
```

Cross-Validation

Why cross-validation?



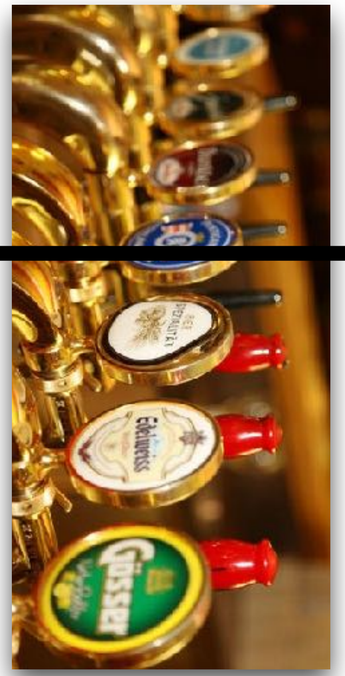
Because you don’t wanna be that guy!!!!

Split data for train and test

X

70%

target



Simple train test split!

```
# Simple Cross-Validation: Split the data set into training and test data

X_train, X_test, target_train, target_test =
model_selection.train_test_split(X, target, test_size=0.25)
```

Simple cross-validation!

```
## Train the classifier
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, target_train)

## Making predictions using test data
target_pred = clf.predict(X_test)

## Report metrics
accuracy = metrics.accuracy_score(target_test, target_pred)
```

K-fold cross-validation!

```
## Set-up generator for k subsets
cvKFold = model_selection.KFold(n_splits=3, shuffle=True,
random_state=33)
cvKFold.get_n_splits(X)

## List of k accuracies on test data
scores = model_selection.cross_val_score(clf, X, target,
cv=cvKFold)
```

In Class Exercise

Please take 45 minutes and complete

Worksheet 6 - DGA Detection ML Classification

Discussion In Class Exercise

- Save (e.g. pickle) fitted classifier and all models to transform raw data to features to make predictions on new URLs (in addition have all functions available)
- Feature Importances: “DurationCreated”, “Length” and “EntropyDomain” are top features. Downside is that when creation date of a new URL is not in data base or not available model becomes useless and most likely make wrong predictions if you just supply a random creation date.
- CountVectorizer and/or other Bag-of-Words methods result in a sparse matrix (=most elements are zeros) and a high-dimensional feature space. Consider word2vec embedding (e.g. https://www.tensorflow.org/tutorials/word2vec)
- Is accuracy of 0.86 acceptable for production? NO! Will result in high false positive rate!
- To achieve model accuracies of 0.999… you need much more data and/or better data, tweak feature engineering/selection and model selection (however don’t forget about the risk to overfit!)
- Consider Deep Learning! (You still need lots of data, but chances of catching “new” malicious URLs may increase due to the fact that neural network learn the feature space themselves!)

Feature Importances

0.524576	DurationCreated
0.092815	Length
0.080639	EntropyDomain
0.059617	DigitsCount
0.057386	FirstDigitIndex
0.057095	LengthDomain
0.008454	com
0.006945	net
0.005946	news.1
0.005724	org
0.004589	ru

Hyper-Parameter Tuning!

General Optimization: e.g. Gradient Descent!

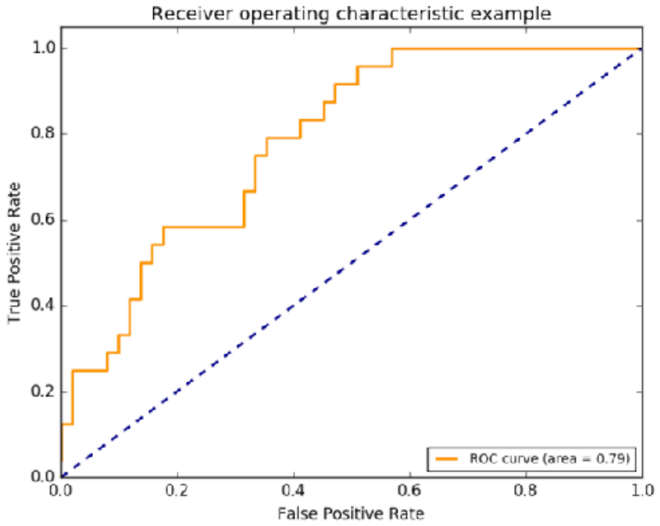
Hyper-Parameter Tuning!

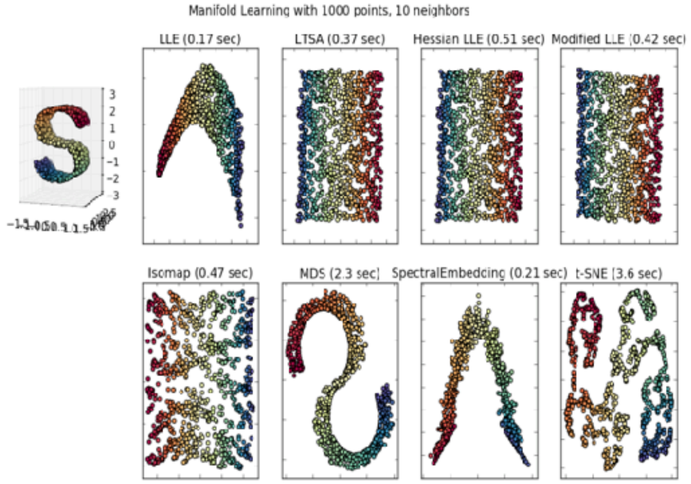
```
clf = ensemble.AdaBoostClassifier()
sss = model_selection.StratifiedShuffleSplit(n_splits=3, test_size=0.25,
random_state=33)
sss.get_n_splits(X, target)

# parameters for AdaBoost Classifier
param_dist = {'learning_rate': [.1, .2, .3, .4, .5, .6, .7, .8, .9, 1.0],
'n_estimators': [5, 10, 25, 50, 75, 100]}

random_search = model_selection.RandomizedSearchCV(clf,
param_distributions=param_dist,cv=sss.split(X, target), n_iter=12)
```

You don’t have to manually train predict all the time, while changing some parameter values !

Future Studies - sklearn

- Feature Selection using Classifiers
- Ranking of best features
- Compare multiple Classifiers and pick best!
- Model Stacking or Majority Voting
- Plotting (ROC curve, manifold learning)





Case Studies in Machine Learning & Cyber Security

Finding SQL Injection Attempts

What is SQL injection?

SQL Injection is a vulnerability, most commonly in web applications in which an attacker can execute malicious queries on **YOUR** server

A normal SQL Query

```
SELECT *
FROM users
WHERE username=charles AND
password=pass1234
```

Pseudo Code for Web App Authentication

```
username = <from user>
password = <from user>

query = “SELECT * FROM users WHERE username =
username AND password = password”

query_result = db.execute(query)
if len( query_result > 0 )
    //Authenticate user
else
    //Boot them out
```

Pseudo Code for Web App Authentication

```
username = “charles”
password = “12345”   //Combination an idiot would use on their luggage
query = “SELECT *
FROM users
WHERE username = charles AND
password = 12345”

query_result = db.execute(query)
if len( query_result > 0 )
    //Authenticate user
else
    //Boot them out
```

Pseudo Code for Web App Authentication

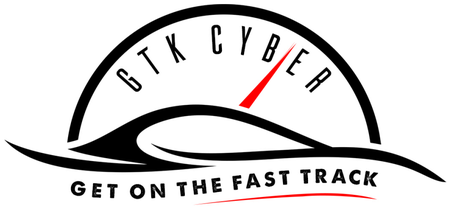
```
username = “charles”
password = “12345 OR 1=1”   //Combination an idiot would use on their luggage
query = “SELECT *
FROM users
WHERE username = charles AND
password = 12345 OR 1=1”

query_result = db.execute(query)
if len( query_result > 0 )
    //Authenticate user
else
    //Boot them out
```

Legit vs. Malicious

```
SELECT count(category) FROM Product WHERE price >
'$20'
SELECT ctid, xmin, * FROM lockdemo
SELECT current_date FROM dual
```

Legit vs. Malicious

```
' or 'unusual' = 'unusual'

' or 'something' = 'some'+'thing'

' or 'text' = n'text'

' or 'something' like 'some%'
```



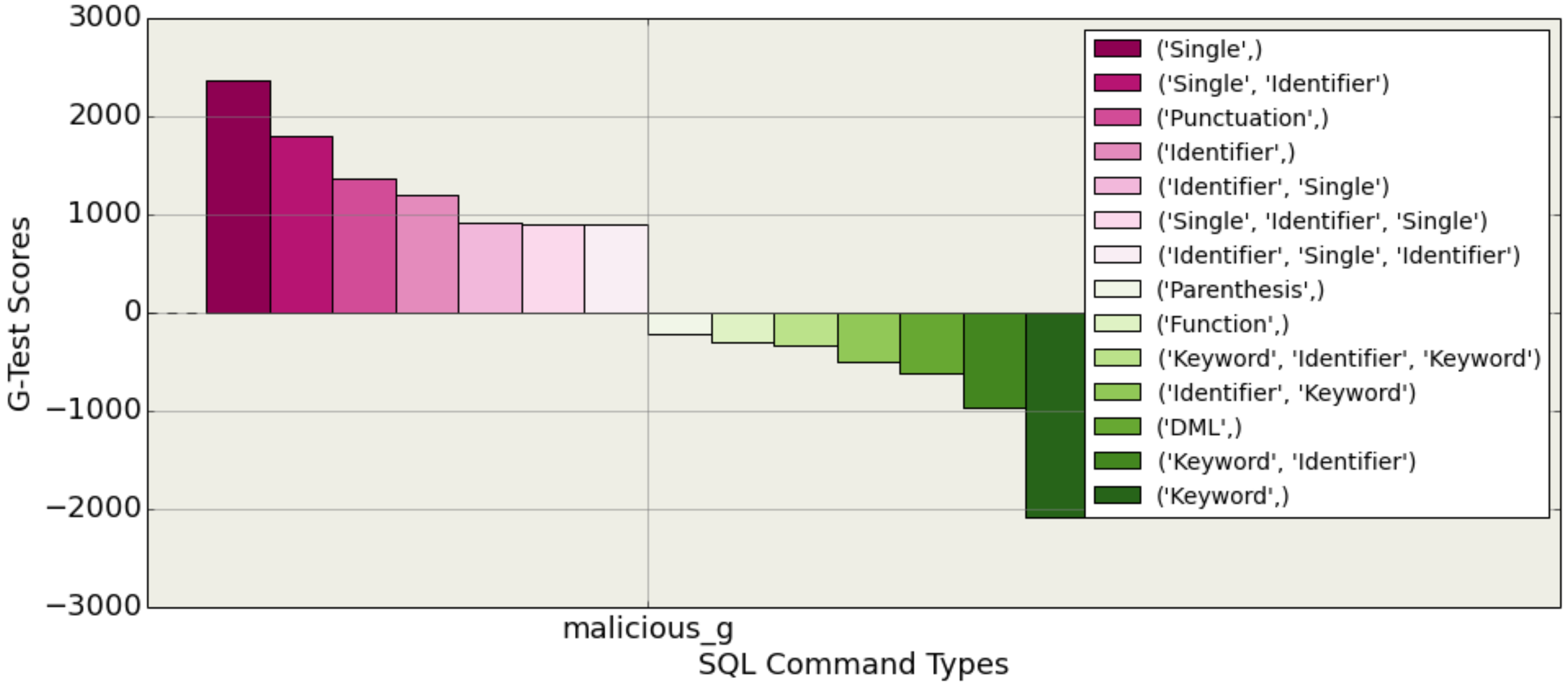
Step 1: Tokenize SQL

```
import sqlparse
def parse_it(raw_sql):
    parsed = sqlparse.parse(unicode(raw_sql,'utf-8'))
    return [token._get_repr_name() for parse in parsed for token in
parse.tokens if token._get_repr_name() != 'Whitespace']

dataframe['parsed_sql'] = dataframe['raw_sql'].map(lambda x: parse_it(x))
```

	raw_sql	type	parsed_sql
0	; exec master..xp_cmdshell 'ping	malicious	[Single, Identifier, Float, Float, Float,
1	create user name identified by	malicious	[DDL, Keyword, Identifier, Keyword,
2	create user name identified by pass123	malicious	[DDL, Keyword, Identifier, Keyword,
3	exec sp_addlogin 'name' , 'password'	malicious	[Keyword, Identifier, IdentifierList]
4	exec sp_addsrvrolemember 'name' ,	malicious	[Keyword, Identifier, IdentifierList]

https://github.com/ClickSecurity/data_hacking

Step 2: Create N-Grams



https://github.com/ClickSecurity/data_hacking

Step 2: Create N-Grams

	raw_sql	type	parsed_sql	sequences
0	; exec master..xp_cmdshell 'ping 10.10.1.2'--	malicious	[Single, Identifier, Float, Float, Float, Erro...	[('Single',), ('Identifier',), ('Float',), ('F...
44	anything' or 'x'='x	malicious	[Identifier, Single, Identifier, Single, Ident...	[('Identifier',), ('Single',), ('Identifier',)...
49	; exec master..xp_cmdshell 'ping aaa.bbb.ccc....	malicious	[Single, Identifier, Error, Single]	[('Single',), ('Identifier',), ('Error',), ('S...
54	; if not(select system_user) <> 'sa' waitfor ...	malicious	[Single, Identifier, Single, Integer, Placehol...	[('Single',), ('Identifier',), ('Single',), ('...
55	; if is_srvrolemember('sysadmin') > 0 waitfor...	malicious	[Single, Identifier, Single, Integer, Placehol...	[('Single',), ('Identifier',), ('Single',), ('...

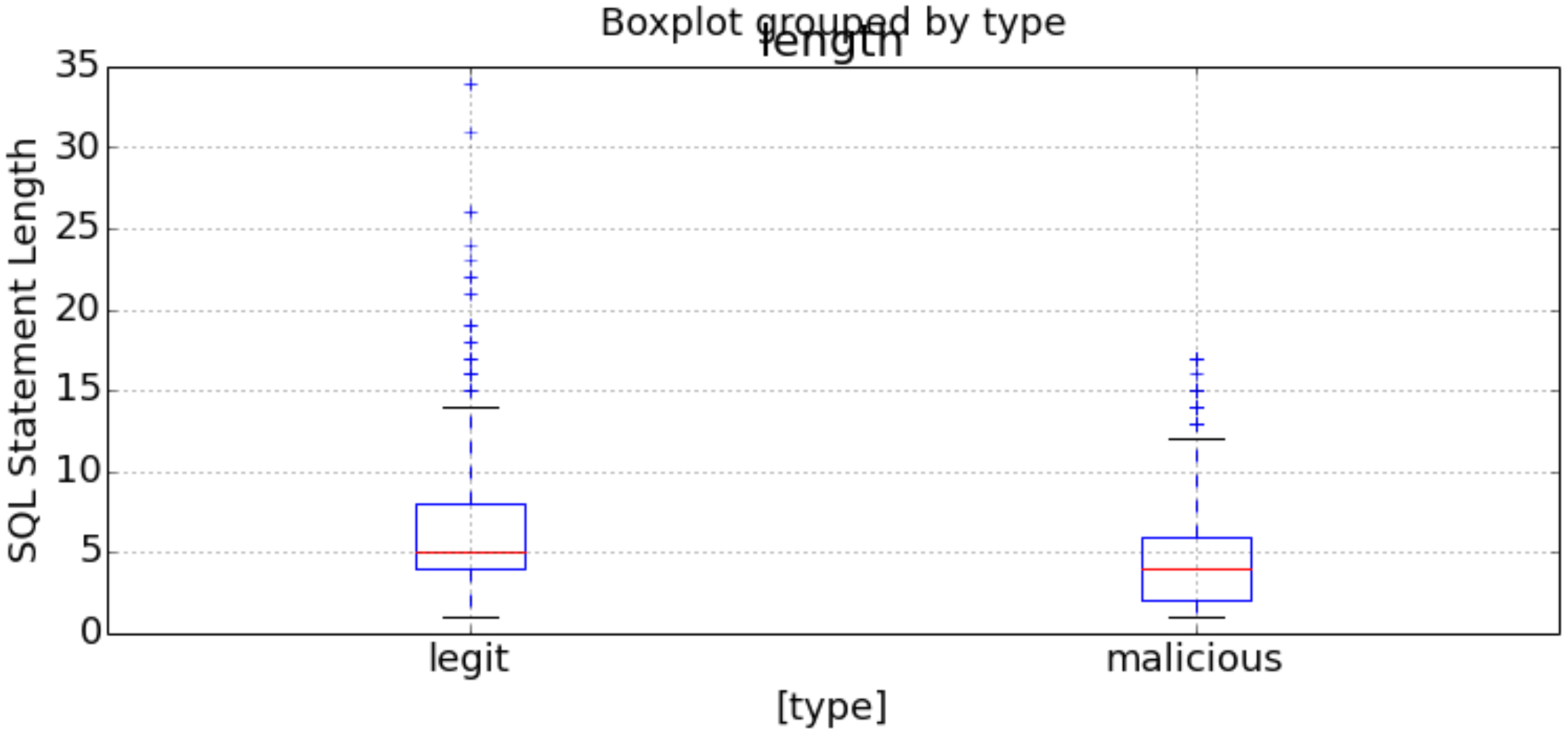
https://github.com/ClickSecurity/data_hacking

Step 3: Build Feature Vector

- Entropy
- Length
- G-Score Test

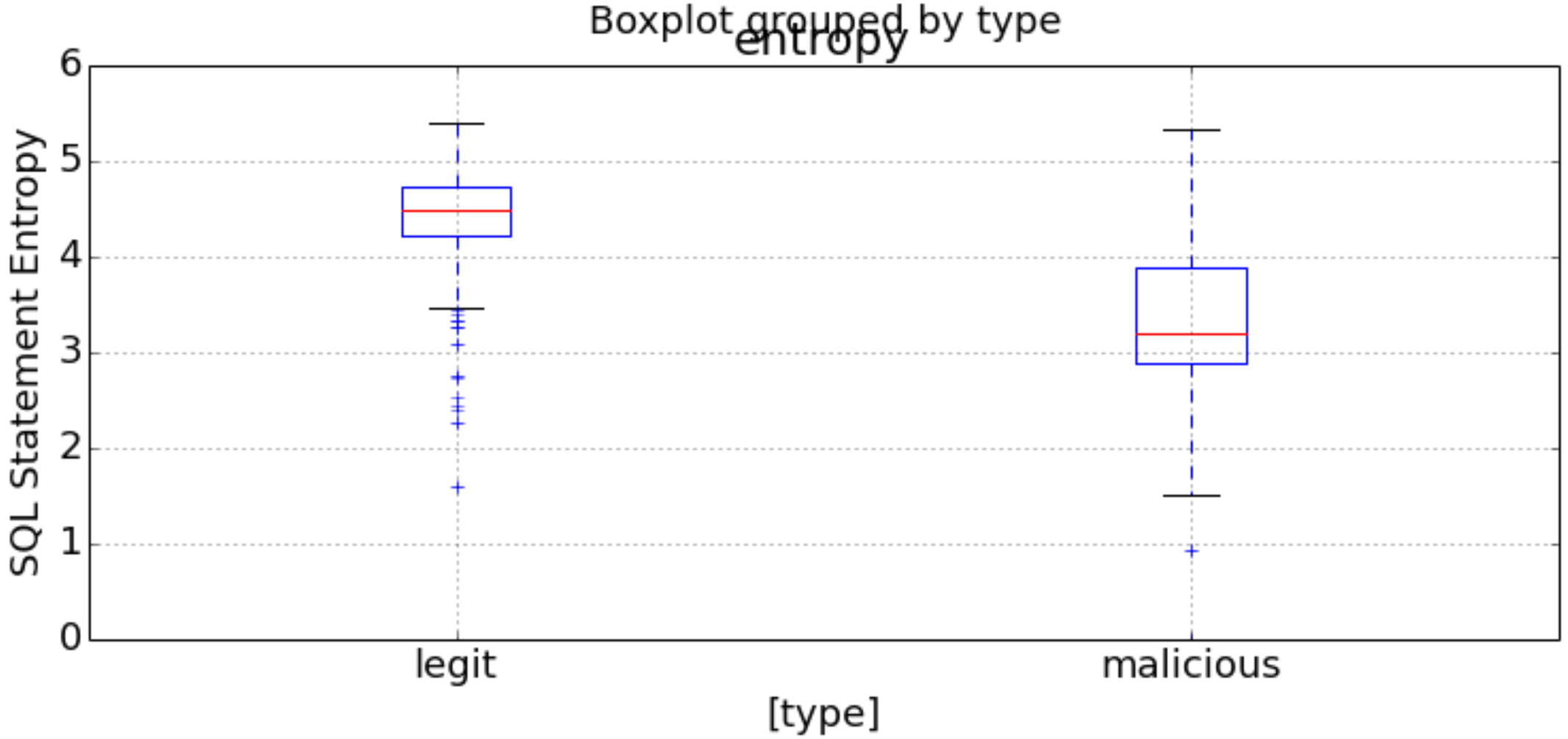
https://github.com/ClickSecurity/data_hacking

Step 3: Build Feature Vector



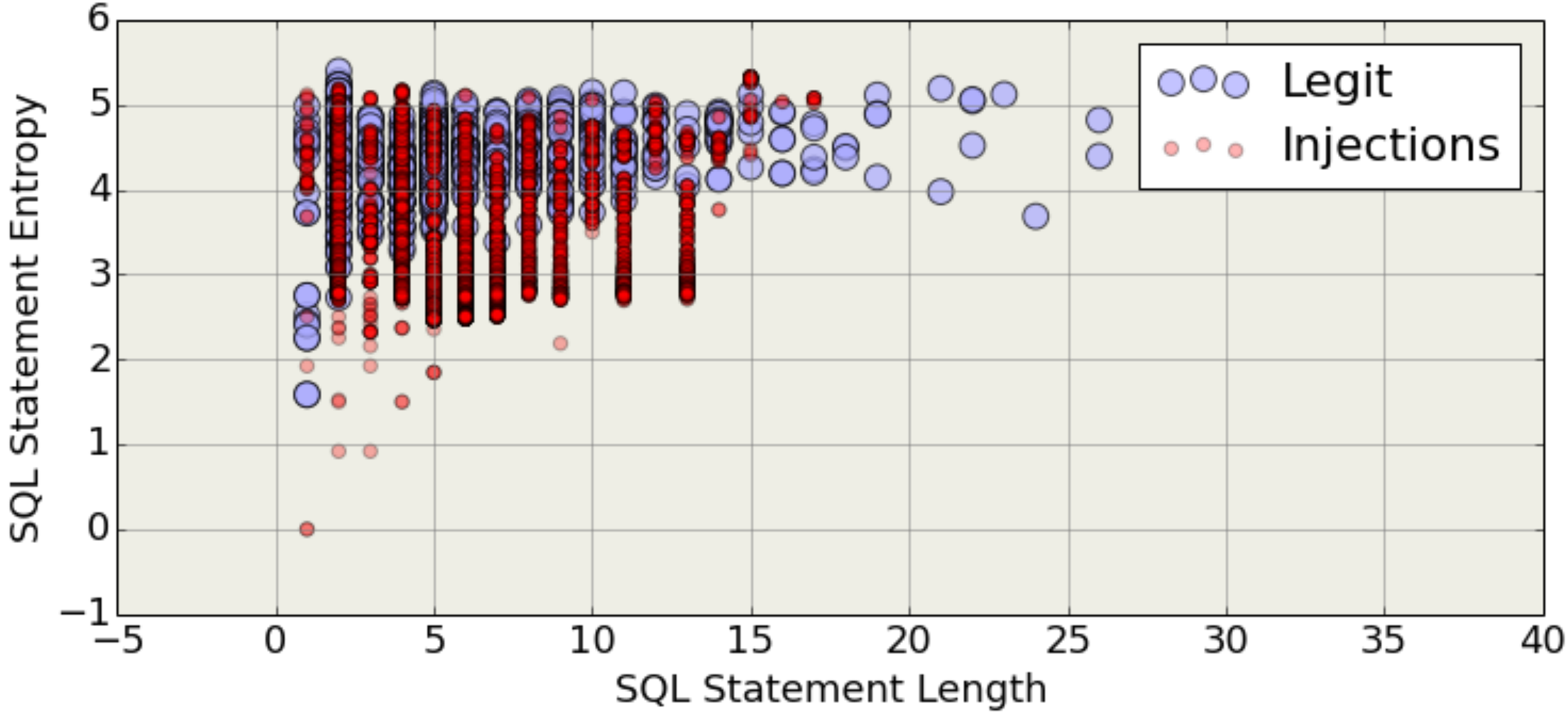
https://github.com/ClickSecurity/data_hacking

Step 3: Build Feature Vector



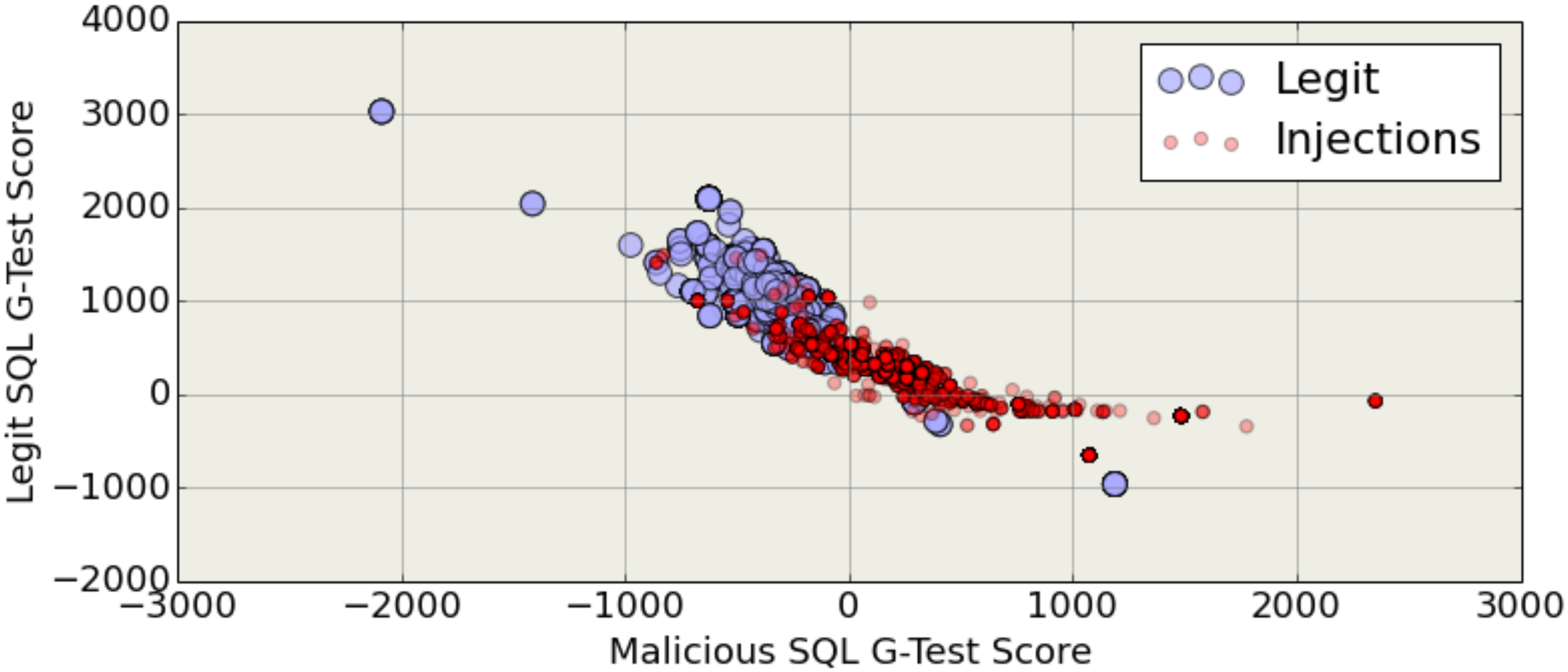
https://github.com/ClickSecurity/data_hacking

Step 3: Build Feature Vector



https://github.com/ClickSecurity/data_hacking

Step 3: Build Feature Vector



https://github.com/ClickSecurity/data_hacking

Step 4: Train Classifier

```
import sklearn.ensemble
clf =
sklearn.ensemble.RandomForestClassifier(n_estimators=20)

scores = sklearn.cross_validation.cross_val_score(clf, X, y,
cv=10, n_jobs=4)
print scores
```


https://github.com/ClickSecurity/data_hacking

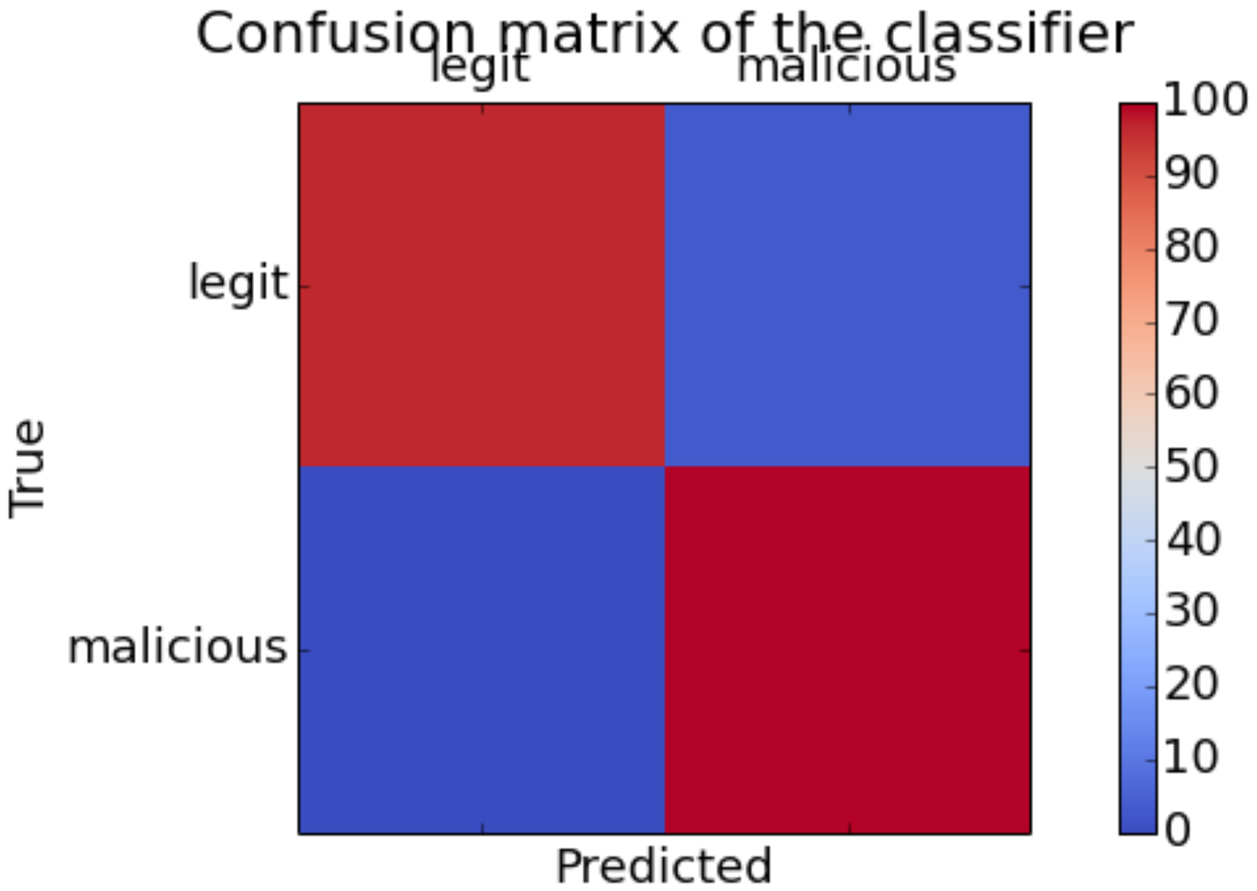
Step 5: Evaluate Model

```
scores = sklearn.cross_validation.cross_val_score(clf, X, y,
cv=10, n_jobs=4)
```

```
[ 0.99784173  0.99784173  1.          0.99784173
0.99856115  0.99784017
  0.99640029  0.99856012  0.99784017  0.99784017]
```


https://github.com/ClickSecurity/data_hacking

Step 5: Evaluate Model




```
Confusion Matrix Stats
legit/legit: 96.59% (170/176)
legit/malicious: 3.41% (6/176)
malicious/legit: 0.08% (2/2603)
malicious/malicious: 99.92% (2601/2603)
```


https://github.com/ClickSecurity/data_hacking

Questions?

Thank you!

gtkcyber.com