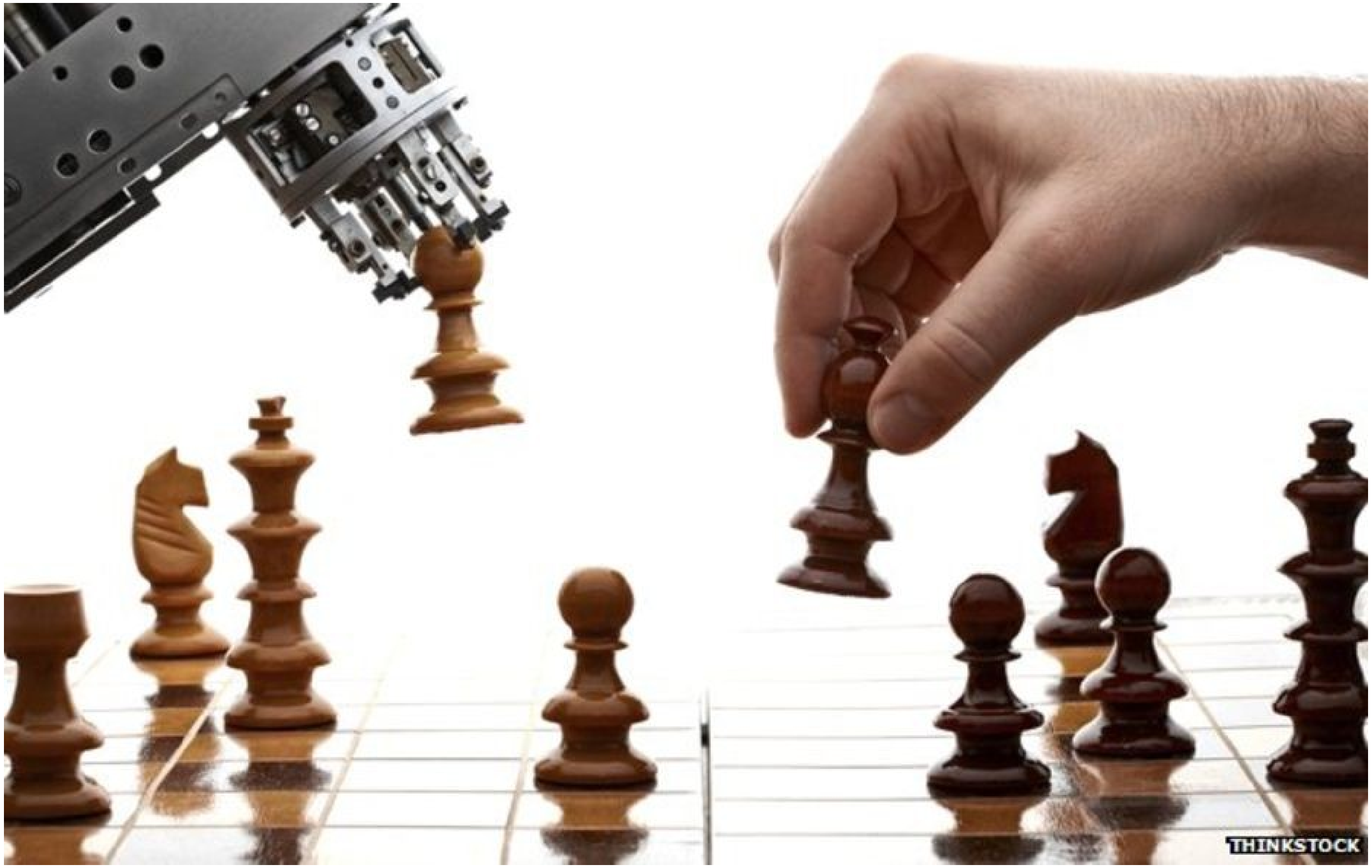


Introduction to Artificial Intelligence
COSC 4550 / COSC 5550

Professor Cheney
11/29/17

data pipelines

collecting and preprocessing data is as (if not more) important to getting good results in practical applications than knowing the underlying machine learning theory

cleaning data

the data that you collect could be messy/poor for a number of reasons, including missing or invalid attribute values due to data entry (or availability) problems

you can quickly search for:

blank attribute fields
nonsensical min/max values (or cardinality)
features with unusually high variance
features with zero or low variance
outliers

there are multiple options for how to treat bad data points:

just ignore that data point
(simplest and adds least errors,
but could be a problem if you're dataset is small)

add a new class for “unknown” values
(simple, but uninformative)

try to estimate the missing attribute
from other features of that data point:

assume the most common value of that attribute
in your dataset (overall mode)

assume the most common value of that attribute
dependent on some other attribute (subclass mode)

find the closest valid datapoint according to the other
attributes and match the missing entries (KNN)

try to estimate the missing attribute
from other features of that data point:

use the mean value of that attribute in your dataset
(or subset)

train a regression or classification model on the relationship
between the missing data and other attributes
(most informed method,
but still doesn't add a lot of new information)

normalizing data

ML methods that use computations in the feature space
(KNN, K-means, neural networks, SVM, …)
are sensitive to the mean and variance of attributes

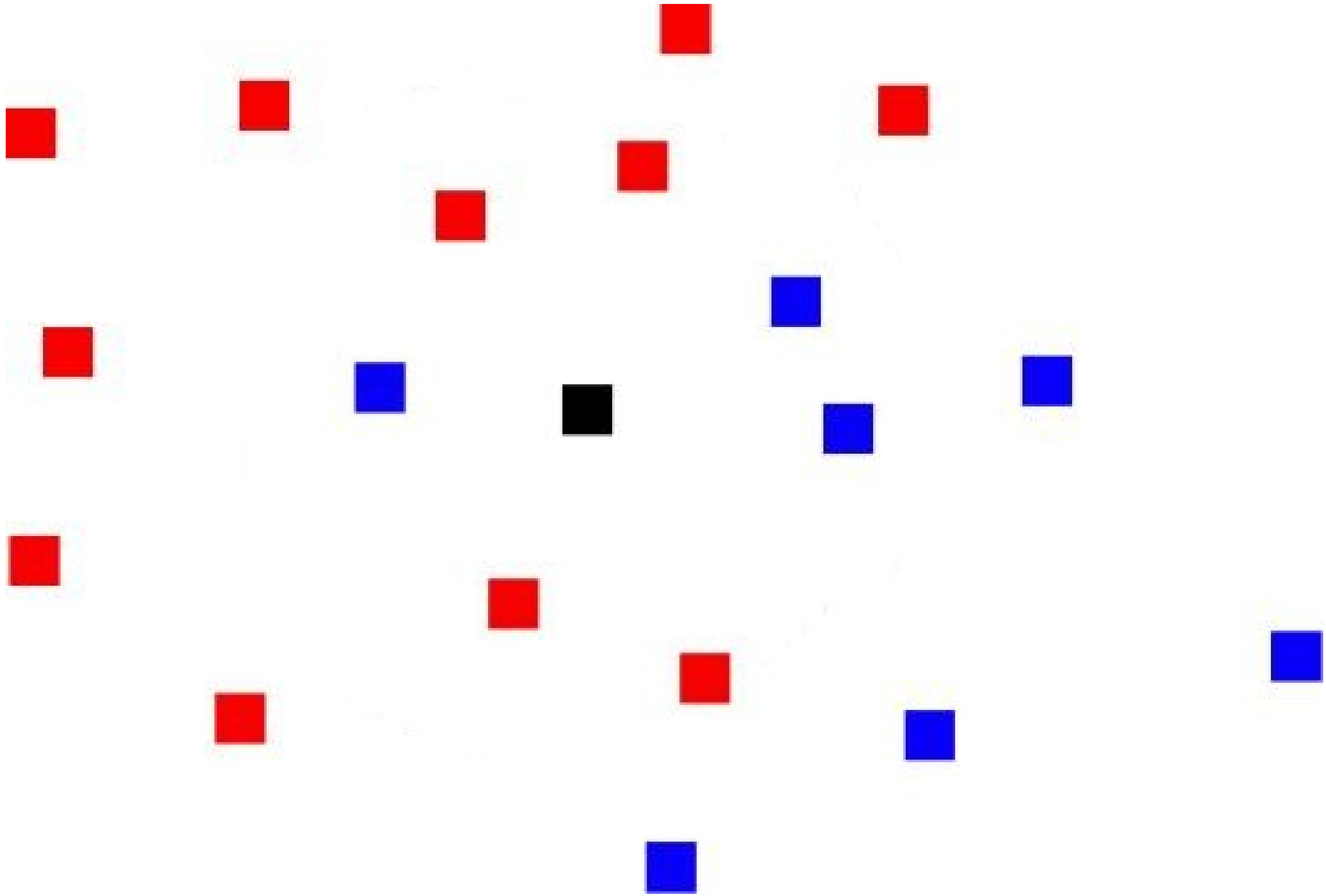
apply z-score transformation (on each attribute separately)
to normalize by mean and standard deviation:

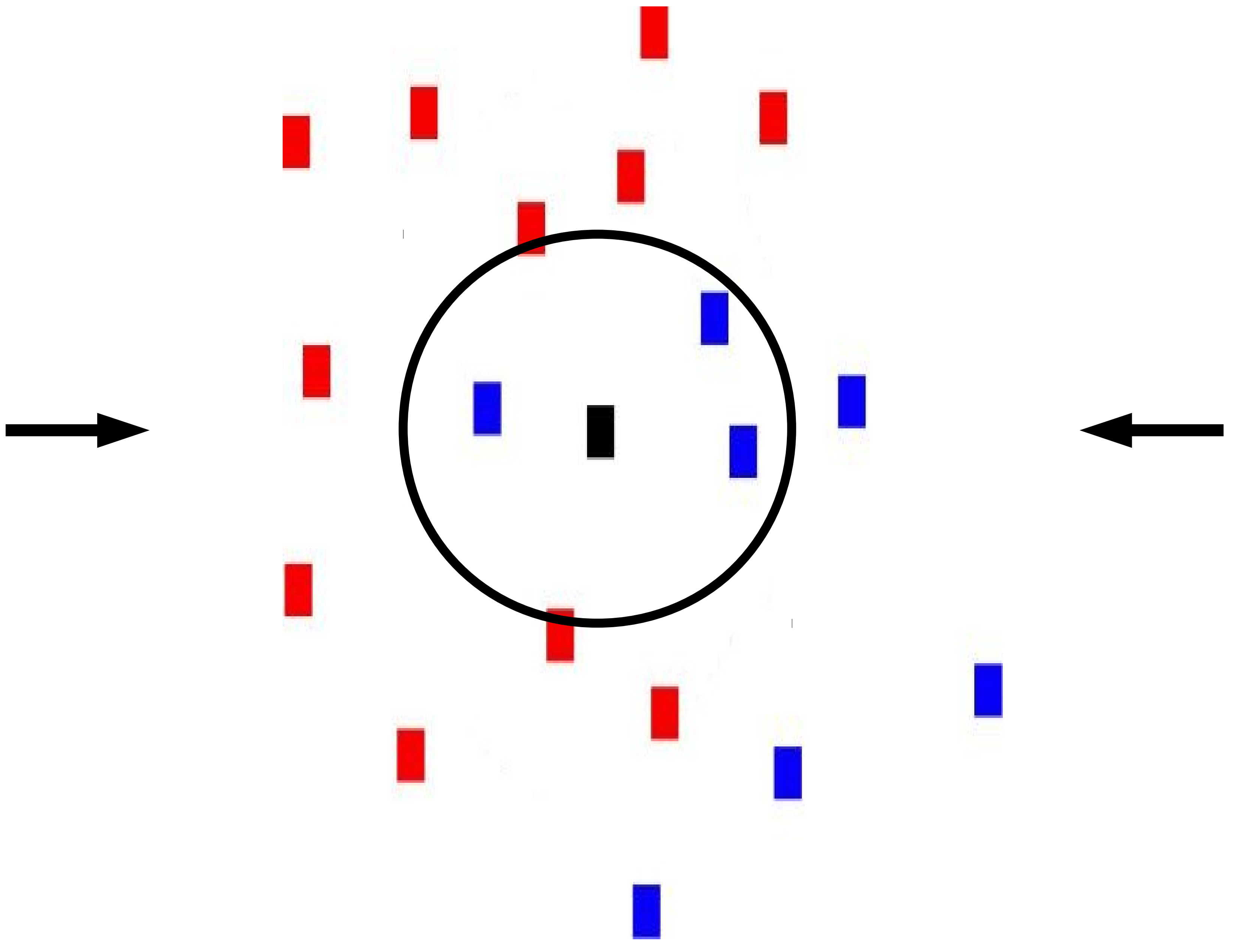
$$z = \frac{x - \mu}{\sigma}$$

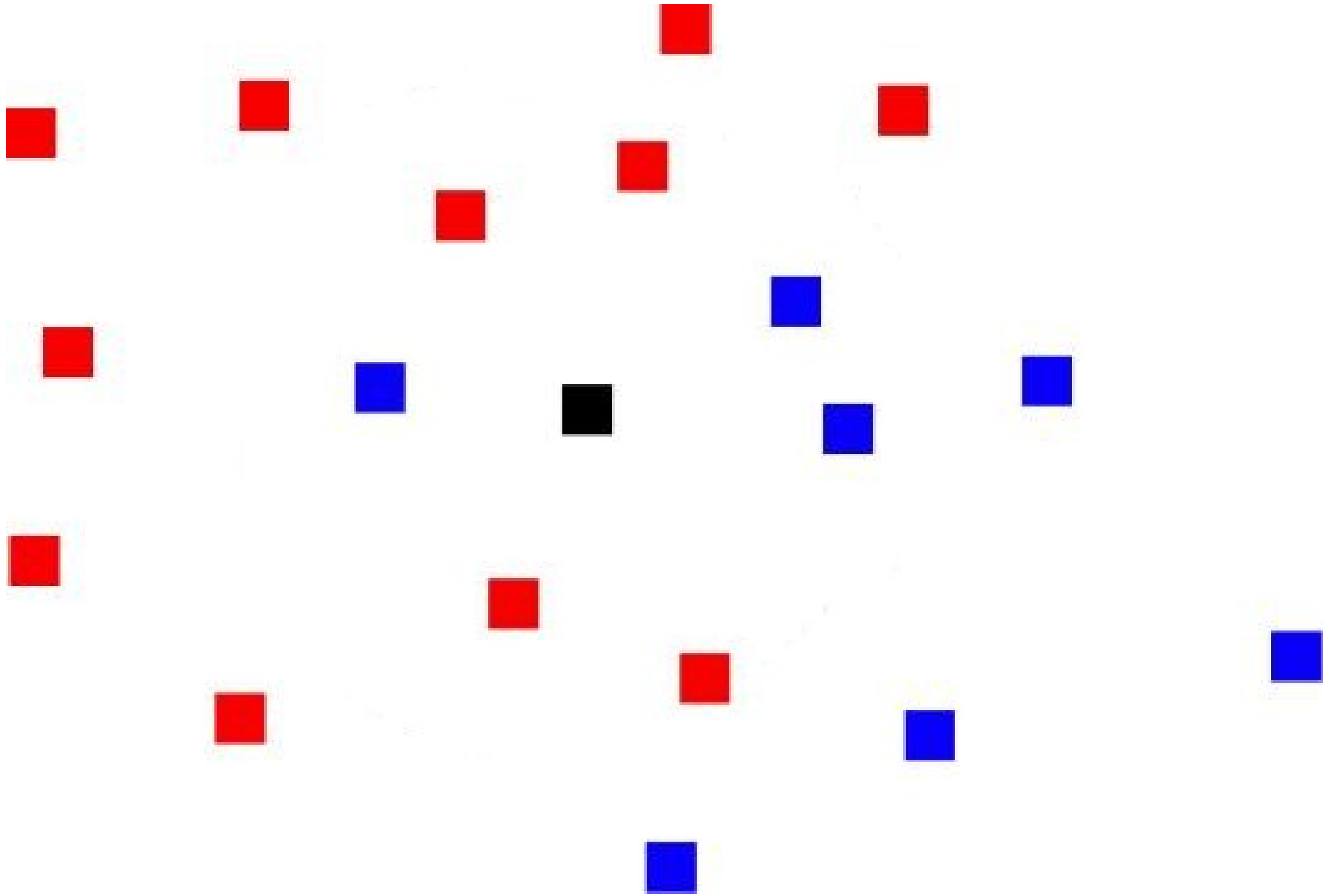
min-max scaling can also be used to map the values of an attribute to the range [0,1]

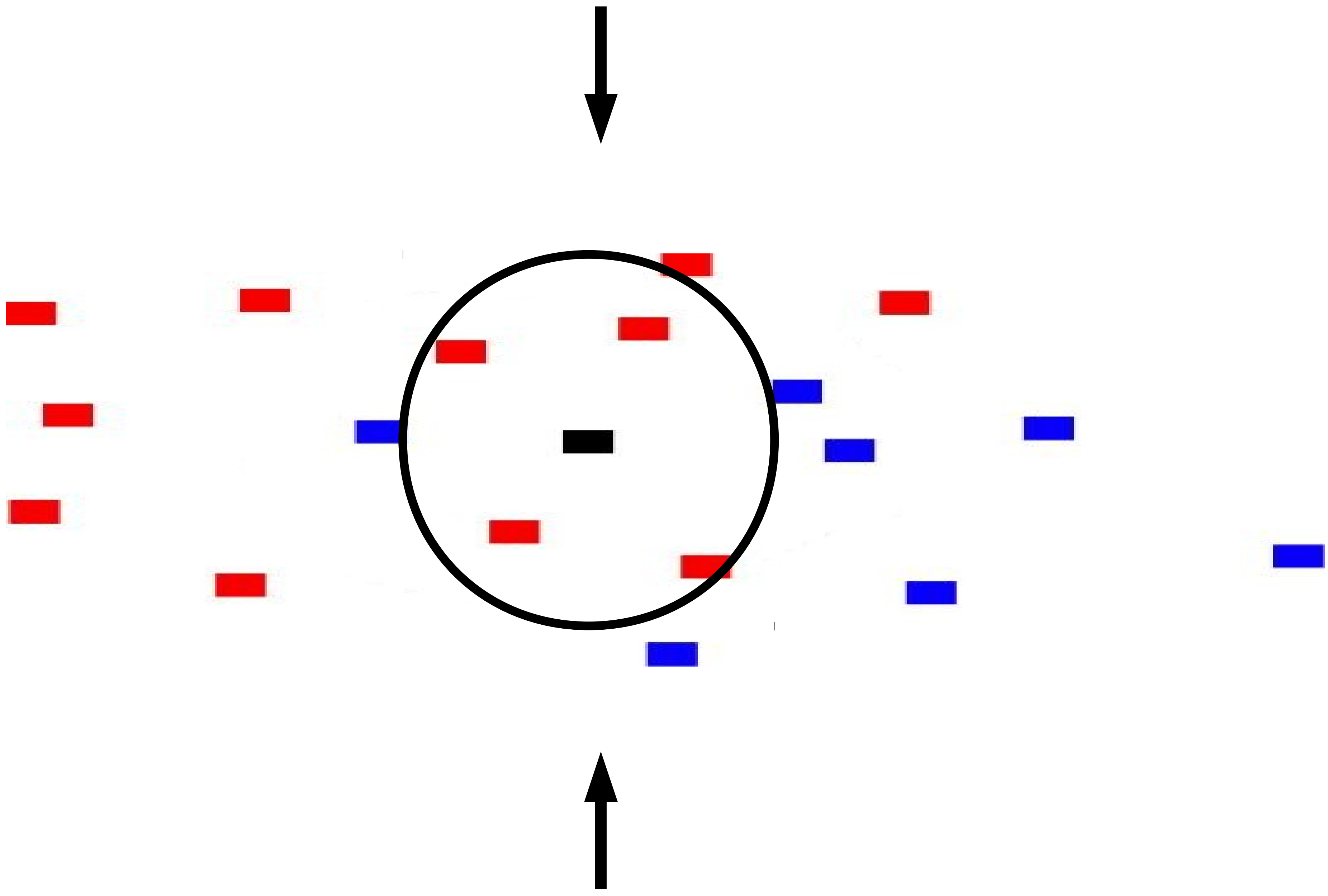
$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

ML methods that use computations in the feature space
(KNN, K-means, neural networks, SVM, …)
are sensitive to the mean and variance of attributes









feature selection

calculate correlations between features
to remove redundant attributes

subject to some threshold (e.g. $r > 0.75$)

ordering matter in what features get kept or removed..

or instead of removing features,
add or multiply features together to create higher-order
(e.g. polynomial) features

by combining (or removing) features,
the resulting dataset is of lower-dimension
and quicker/easier to learn

recursive feature elimination based on variable importance

train a model with all variables,
find the variable with the least importance to the model,
remove that variable and retrain with remaining set

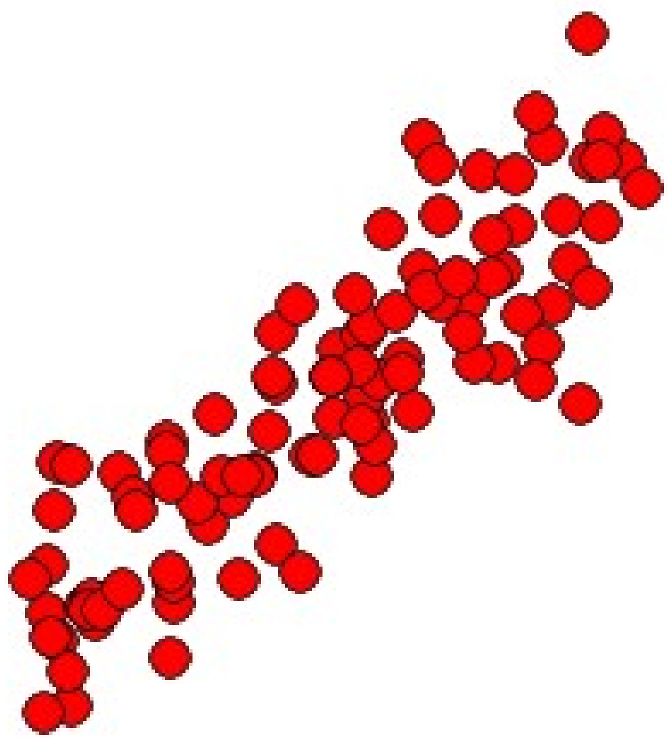
how do you measure variable importance?

for some methods, this is explicitly part of the solution
(e.g. parameter coefficients in linear regression)

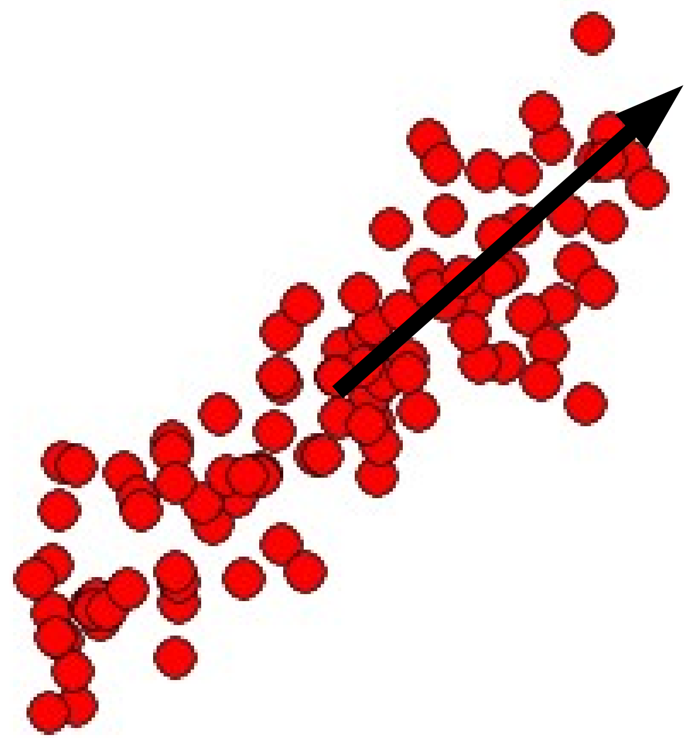
but you may need to randomly remove/perturb features
and retrain your model to find the effect of that variable
on the quality of your solution (exhaustively,
by a genetic algorithm, or simulated annealing)

Principal Component Analysis (PCA)

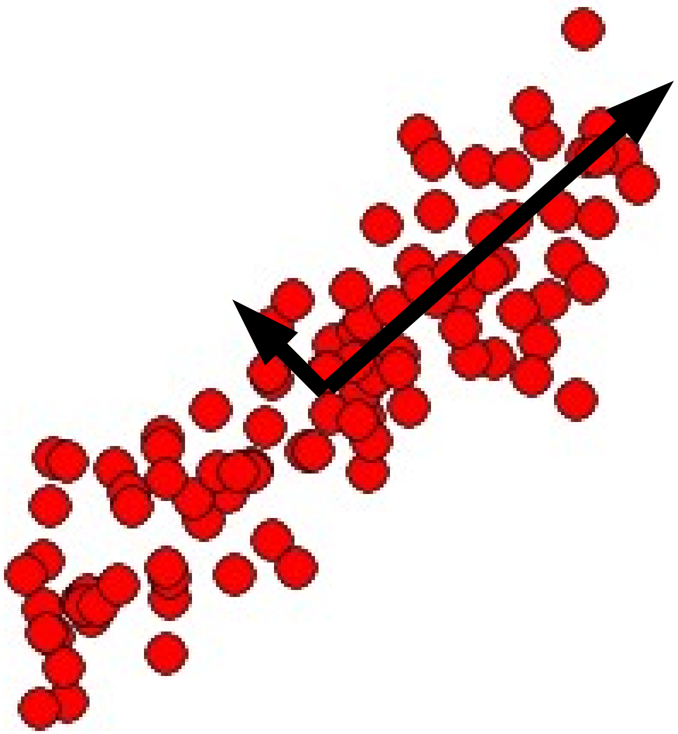
Principal Component Analysis (PCA) finds the basis vector (i.e. direction) that explain the most variance in the data



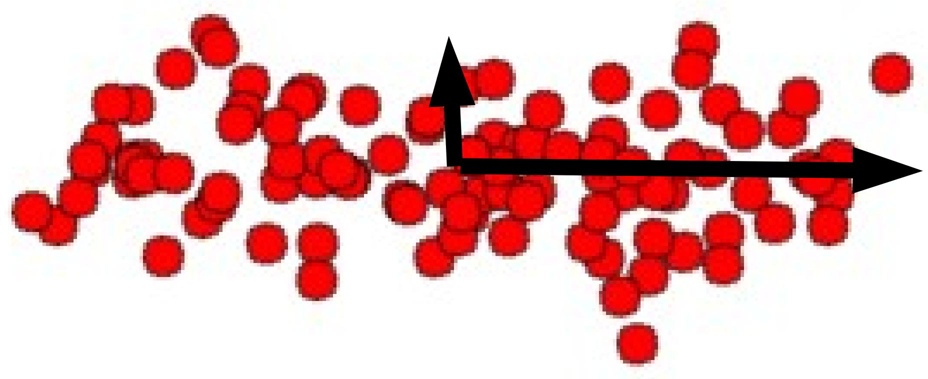
you can think of this similarly to finding the line that best fits of the data as a combination of the attributes of the data



PCA then finds the dimension that is orthogonal to the original basis vector that accounts for the next most variance



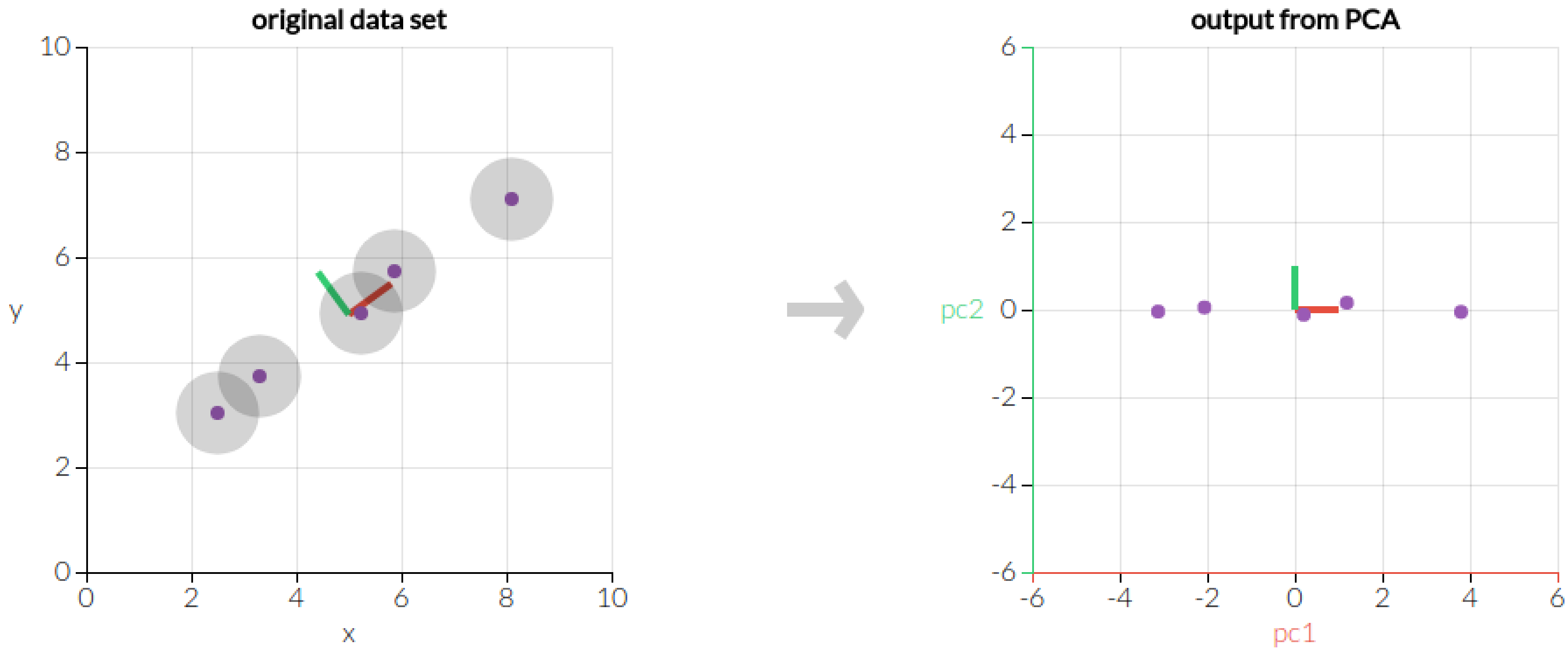
it then transforms the dataset, such that the data is described by these vectors (combinations of the original features)



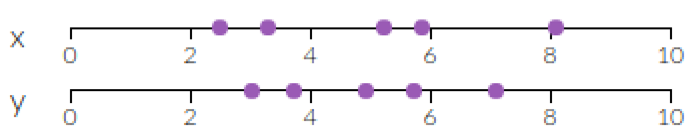
doing so creates new features that are completely uncorrelated each each other (they're orthogonal!)

and it ranks these new features in order of their importance in describing the variance of the data

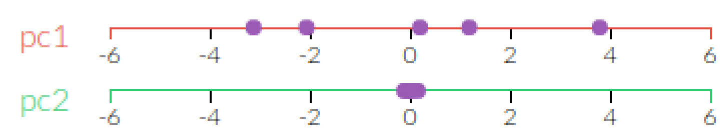
so later features may contribute little or nothing to the variance of the data (and can be removed – yay dimensionality reduction!)

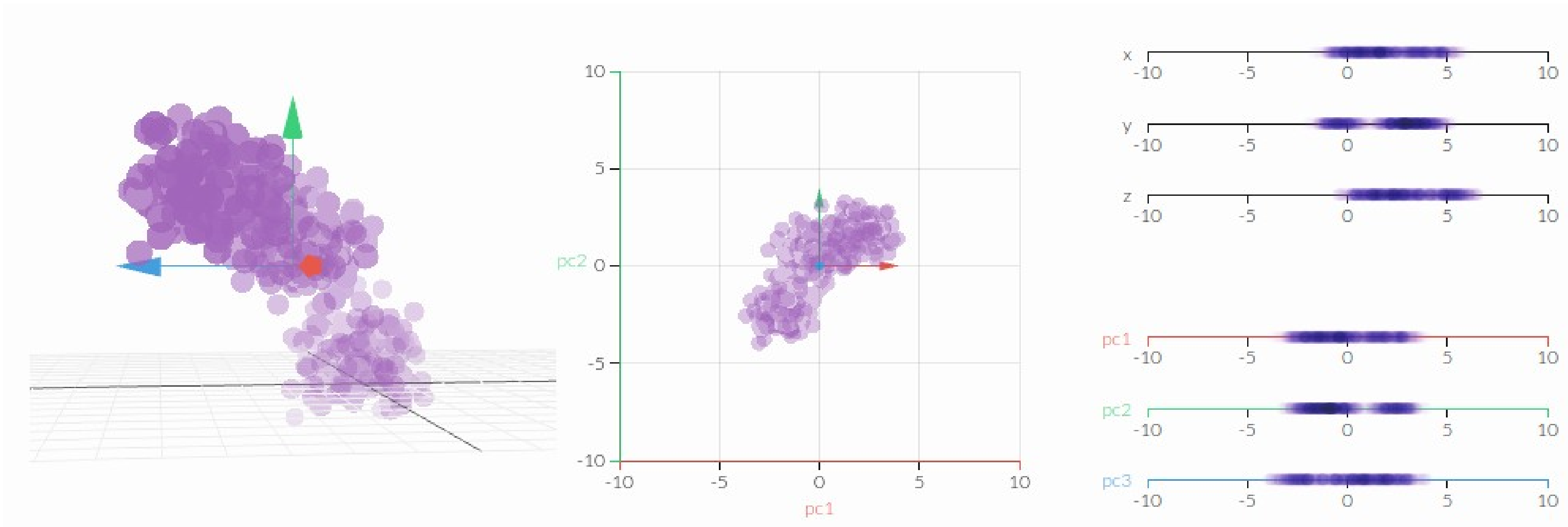


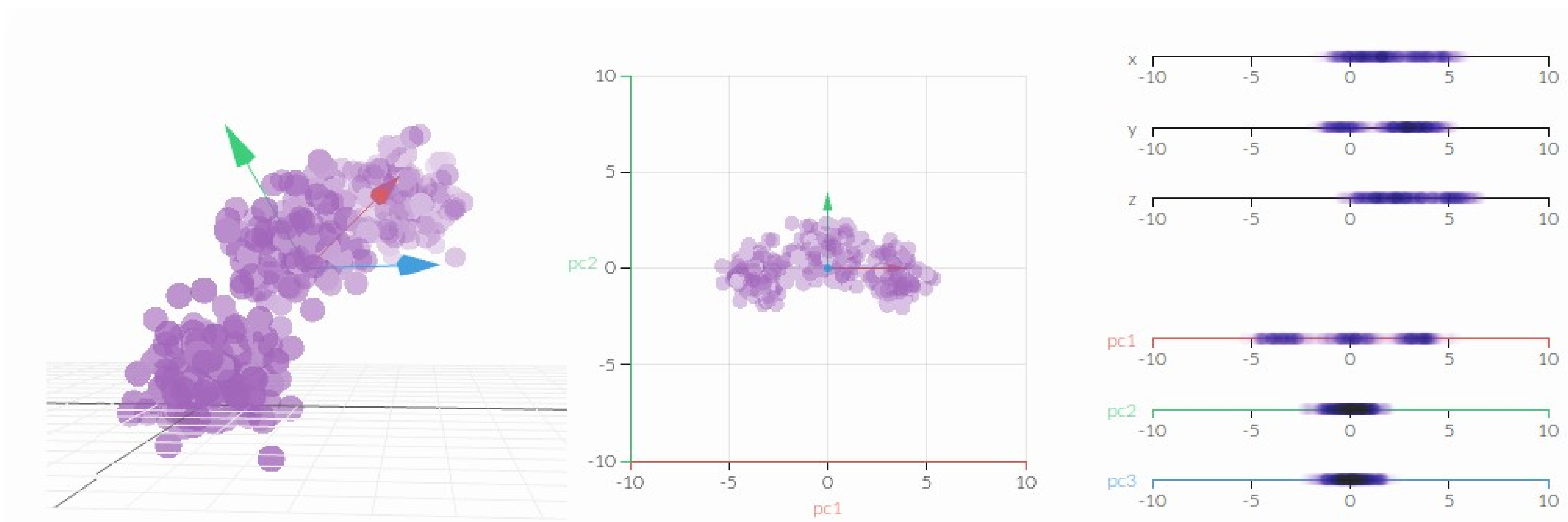
PCA is useful for eliminating dimensions. Below, we've plotted the data along a pair of lines: one composed of the x-values and another of the y-values.



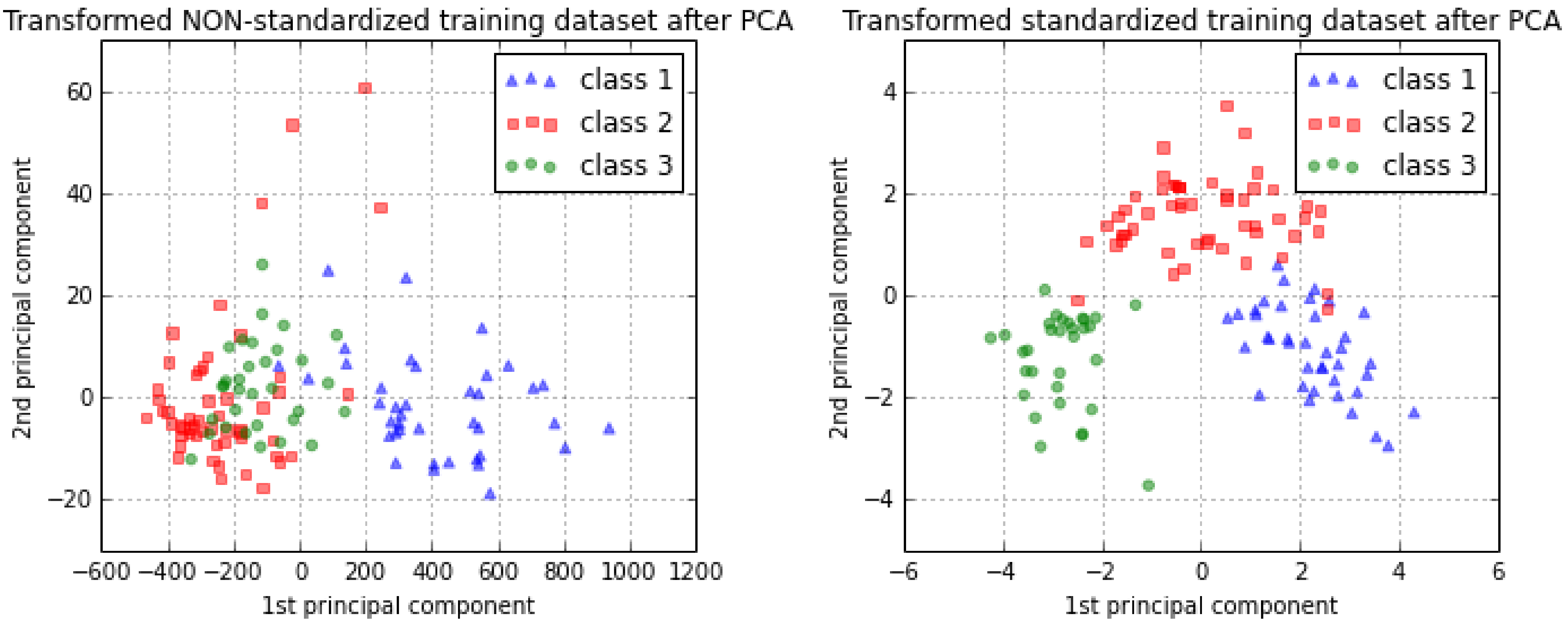
If we're going to only see the data along one dimension, though, it might be better to make that dimension the principal component with most variation. We don't lose much by dropping PC2 since it contributes the least to the variation in the data set.




10
5
pc2 0
-5
-10
-10 -5 0 5 10
pc1
x
y
z
pc1
pc2
pc3
-10 -5 0 5 10


10
5
pc2 0
-5
-10
-10 -5 0 5 10
pc1
x
y
z
pc1
pc2
pc3
-10 -5 0 5 10

side note: PCA also requires normalized data



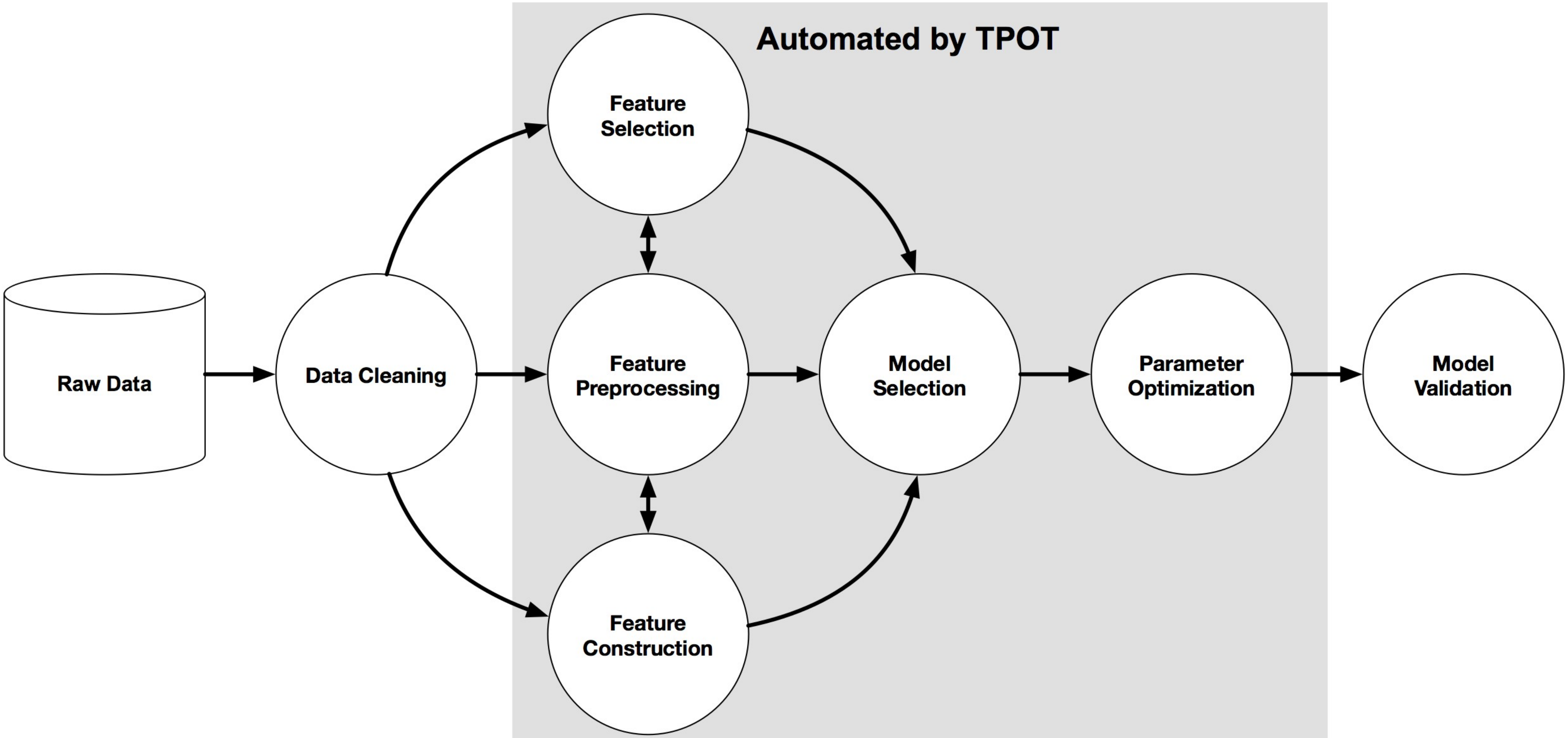
model selection

now we can finally start the machine learning
that we've been talking about this whole time…

choosing our model, training on data, validating on test sets this seems like a lot of work and tons of choice to make…

let's automate it all!

Tree-based Pipeline Optimization Tool


Automated by TPOT
Raw Data
Data Cleaning
Feature Selection
Feature Preprocessing
Feature Construction
Model Selection
Parameter Optimization
Model Validation

Pipeline Operators

Preprocessors. We implemented a standard scaling operator that uses the sample mean and variance to scale the features (StandardScaler), a robust scaling operator that uses the sample median and inter-quartile range to scale the features (RobustScaler), and an operator that generates interacting features via polynomial combinations of numerical features (PolynomialFeatures).

Pipeline Operators

Decomposition. We implemented RandomizedPCA, a variant of Principal Component Analysis that uses randomized Singular Value Decomposition (SVD) [13].

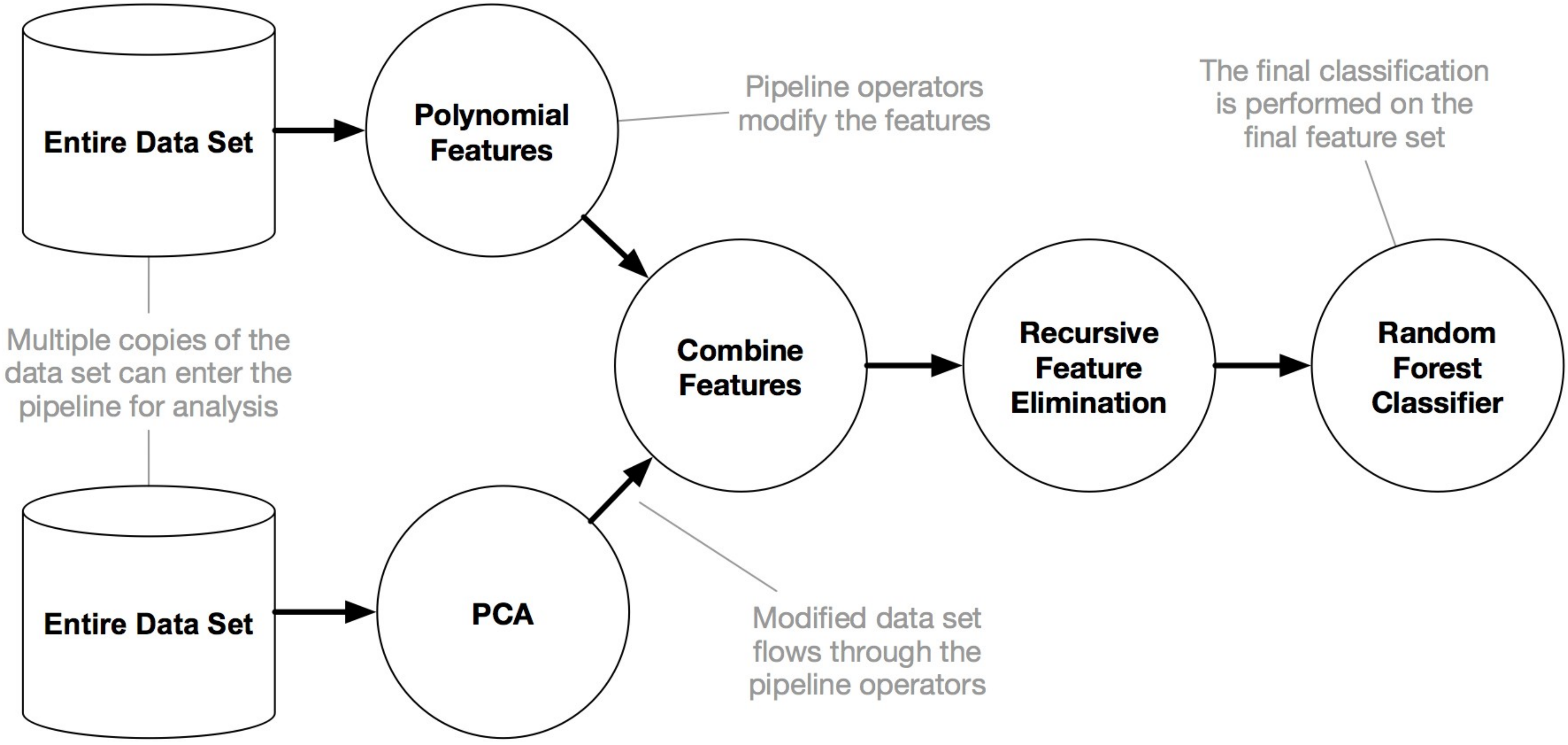
Pipeline Operators

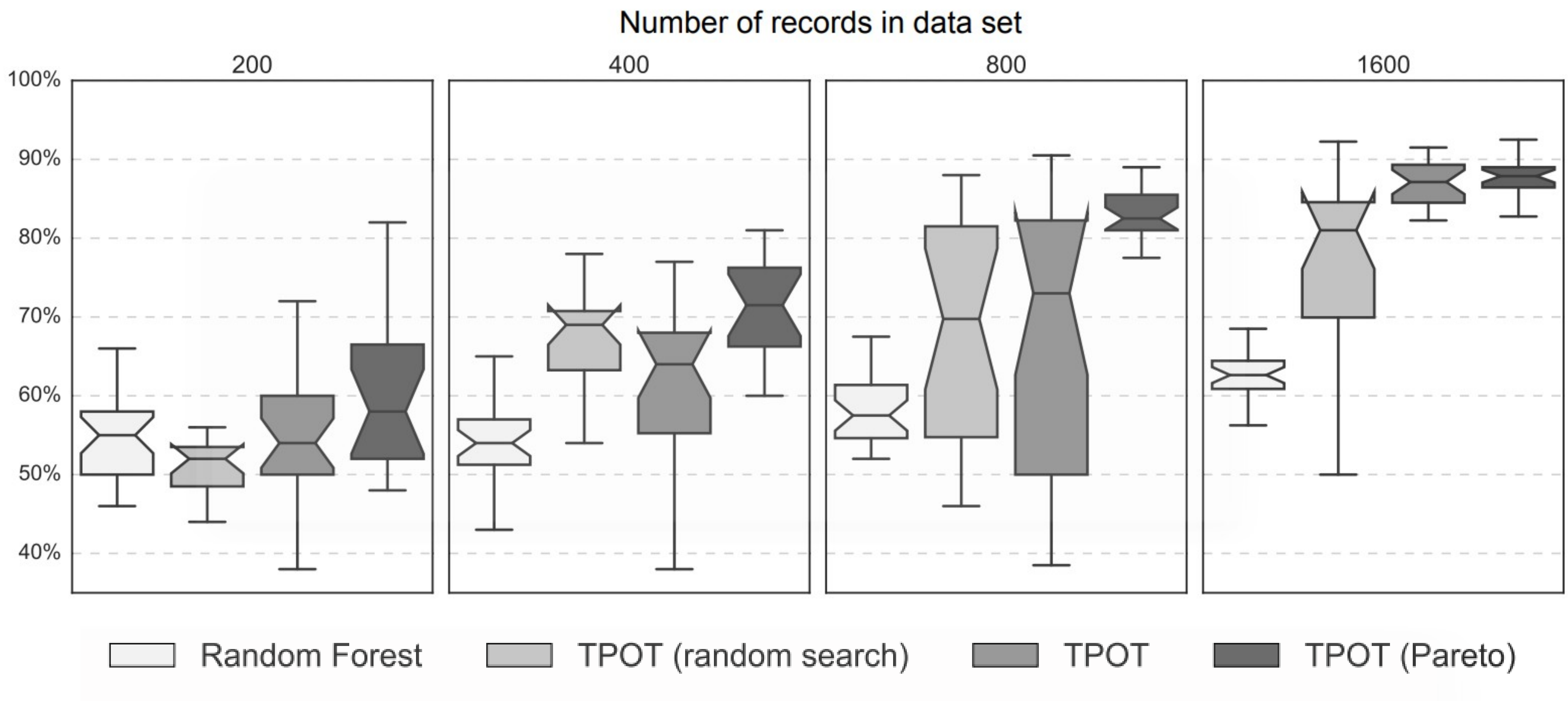
Feature Selection. We implemented a recursive feature elimination strategy (RFE), a strategy that selects the top k features (SelectKBest), a strategy that selects the top n percentile of features (SelectPercentile), and a strategy that removes features that do not meet a minimum variance threshold (VarianceThreshold).

Pipeline Operators

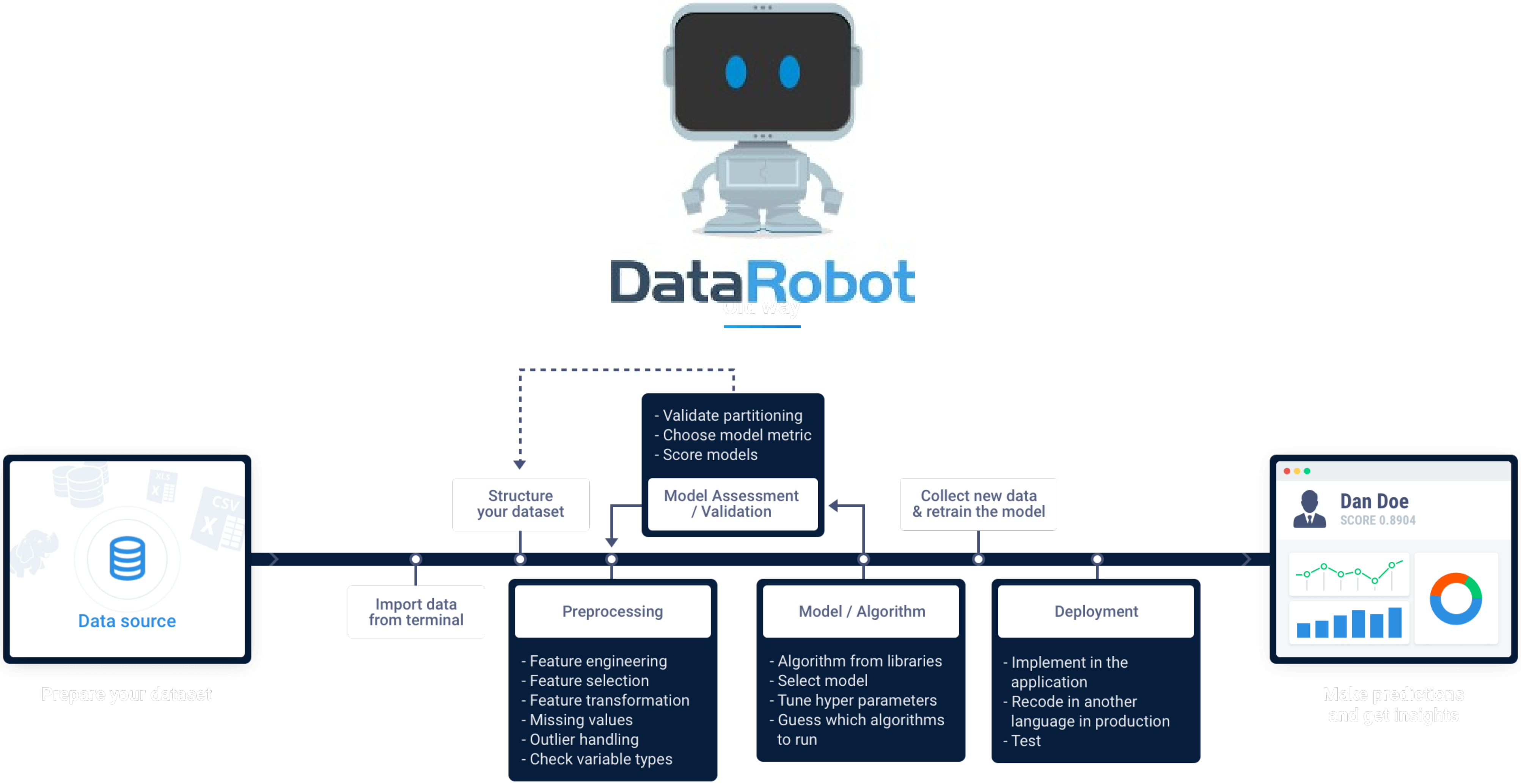
Models. In this paper, we focus on supervised learning models. We implemented both individual and ensemble tree-based models (DecisionTreeClassifier, RandomForestClassifier, and GradientBoostingClassifier), non-probabilistic and probabilistic linear models (SVM and LogisticRegression), and k-nearest neighbors (KNeighborsClassifier).

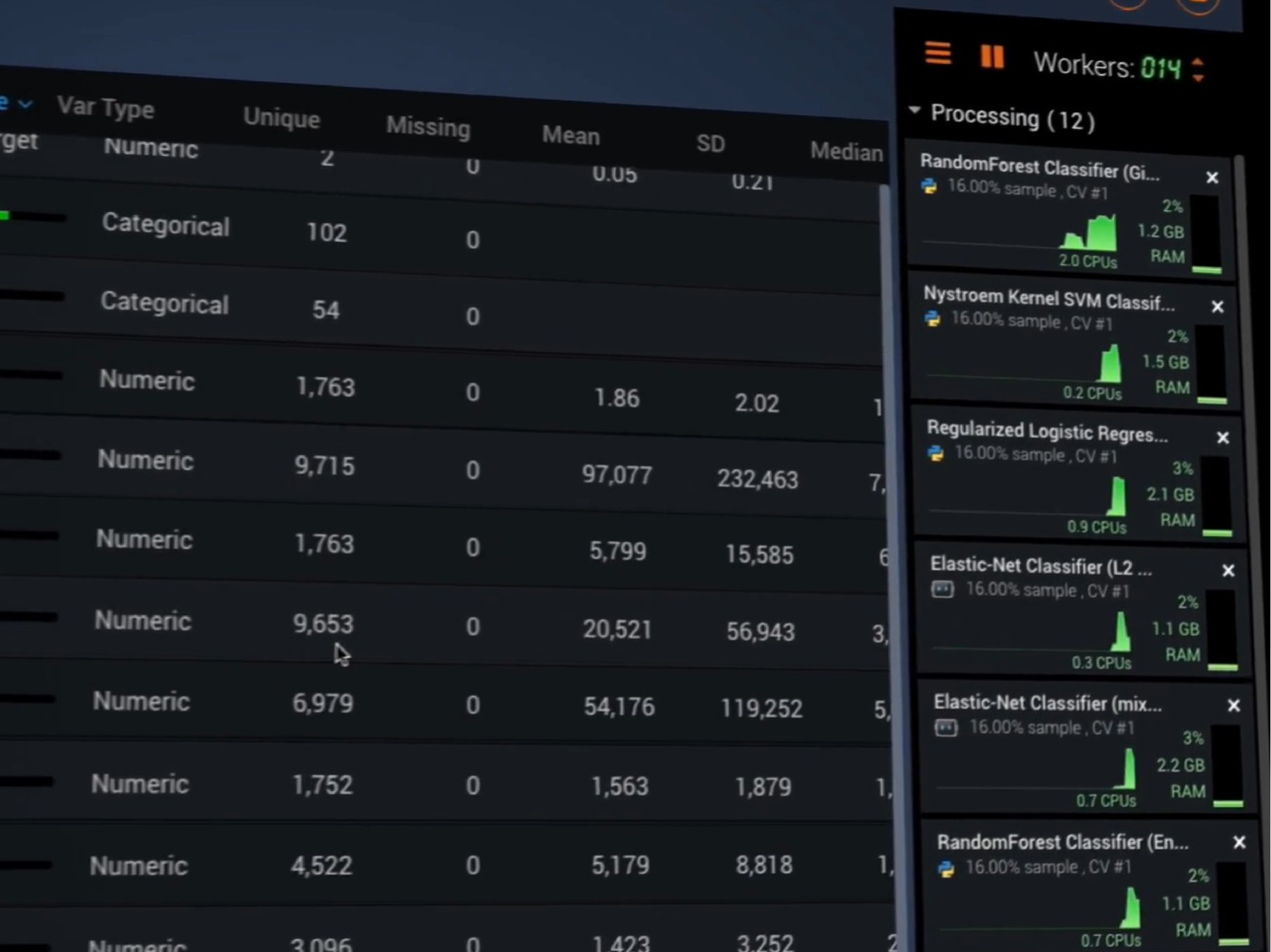
uses genetic programming (a genetic algorithm for trees) to build trees that represent data pipelines





Number of records in data set
200
400
800
1600
100%
90%
80%
70%
60%
50%
40%
Random Forest
TPOT (random search)
TPOT
TPOT (Pareto)


DataRobot
Data source
Prepare your dataset
Import data from terminal
Structure your dataset
Preprocessing
- Feature engineering
- Feature selection
- Feature transformation
- Missing values
- Outlier handling
- Check variable types
- Validate partitioning
- Choose model metric
- Score models
Model Assessment / Validation
Model / Algorithm
- Algorithm from libraries
- Select model
- Tune hyper parameters
- Guess which algorithms to run
Collect new data & retrain the model
Deployment
- Implement in the application
- Recode in another language in production
- Test
Dan Doe
SCORE 0.8904
Make predictions and get insights


Var Type
Unique
Missing
Mean
SD
Median
Numeric
Categorical
Workers: 014
Processing (12)
RandomForest Classifier (Gi...
16.00% sample , CV #1
2.0 CPUs
1.2 GB RAM
Nystroem Kernel SVM Classif...
16.00% sample , CV #1
0.2 CPUs
1.5 GB RAM
Regularized Logistic Regres...
16.00% sample , CV #1
0.9 CPUs
2.1 GB RAM
Elastic-Net Classifier (L2 ...
16.00% sample , CV #1
0.3 CPUs
1.1 GB RAM
Elastic-Net Classifier (mix...
16.00% sample , CV #1
0.7 CPUs
2.2 GB RAM
RandomForest Classifier (En...
16.00% sample , CV #1
0.7 CPUs
1.1 GB RAM

all of this is very easy to use/code with ML packages

we'll see that next class…