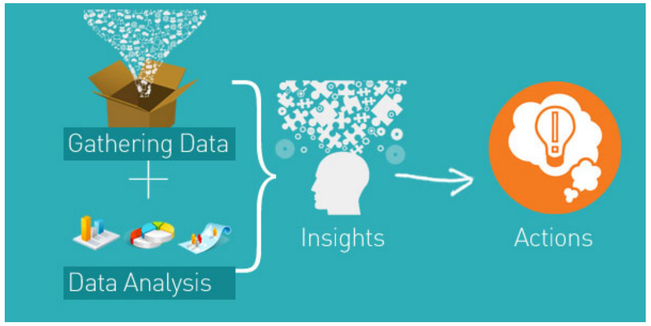
Machine Learning Essentials

Peerapon S.

Machine Learning (2/67)



Topics

Basic types of machine learning

Model training

Overfitting and underfitting

Model development process

What is Machine Learning ?

"FIELD OF STUDY THAT GIVES COMPUTERS THE ABILITY TO LEARN WITHOUT BEING EXPLICITLY PROGRAMMED."

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM* Journal of Research and Development, 3, 210–229.

"THE STUDY OF COMPUTER ALGORITHMS THAT ALLOWS COMPUTER PROGRAMS TO AUTOMATICALLY IMPROVE THROUGH EXPERIENCE."

Mitchell, T. M. (1997). Machine learning. McGraw-Hill.

Basic Types of Learning

Labeled Data

Example inputs

Example outputs

Supervised Learning Algorithms

Unseen inputs

Prediction Model

Classification model
Regression model

Predicted outputs

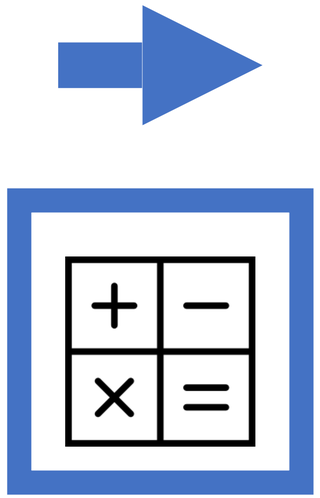
Input data

Unsupervised Learning Algorithms

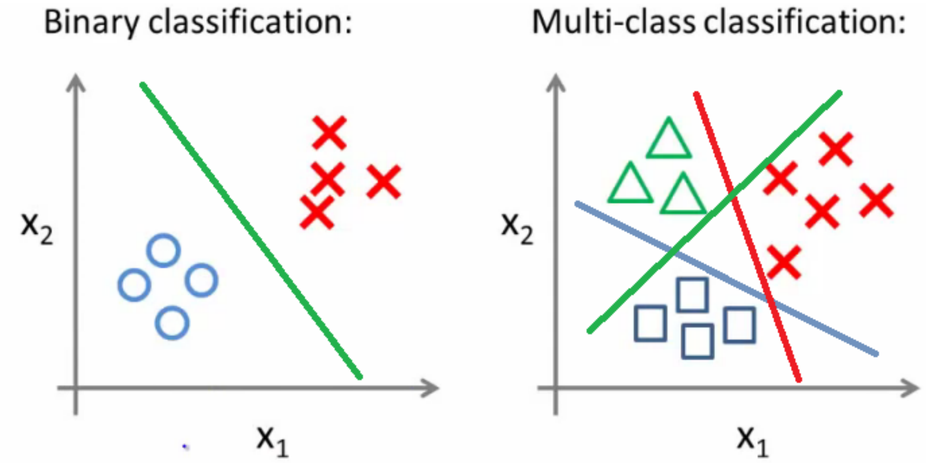
Hidden/Unusual patterns
Alternative representations

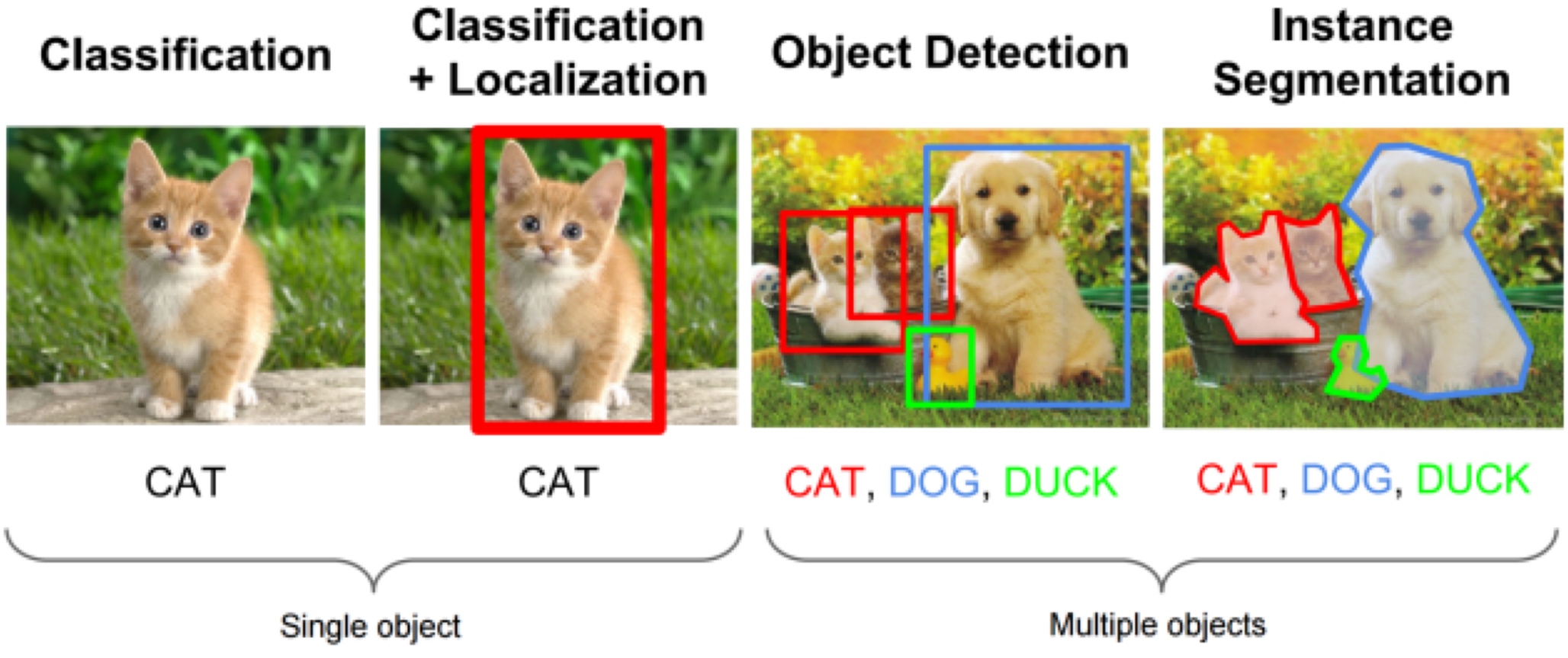
Classification Model

- Area
- Age
- Bedrooms
- Bathrooms
- Parking area

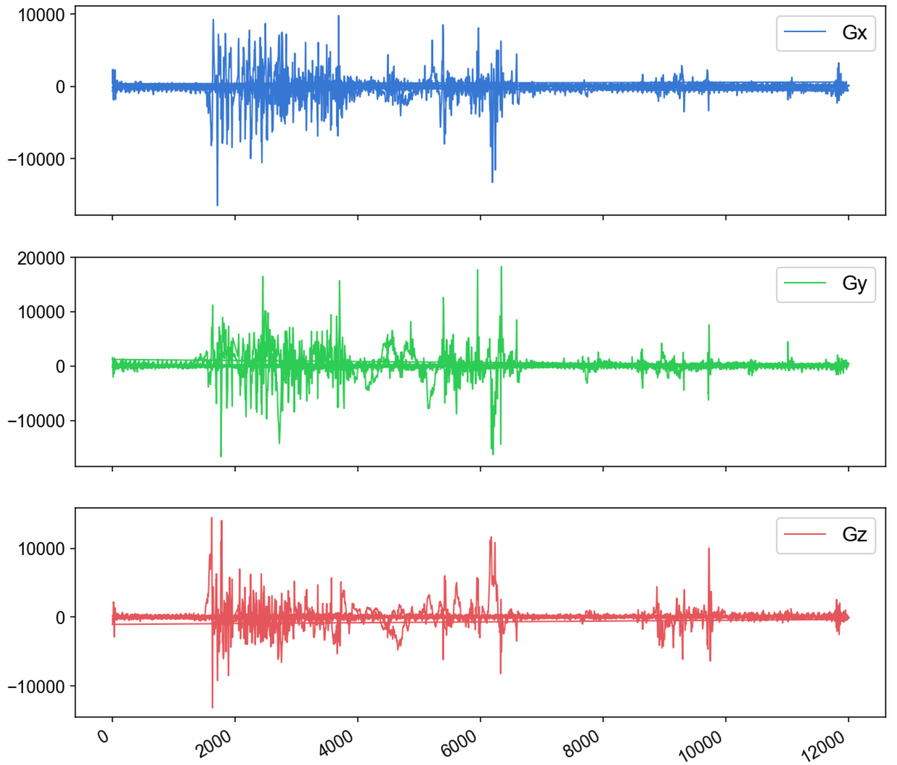


Kinds of properties (Residence or Commercial)




Classification
Classification + Localization
Object Detection
Instance Segmentation
CAT
CAT
CAT, DOG, DUCK
CAT, DOG, DUCK
Single object
Multiple objects

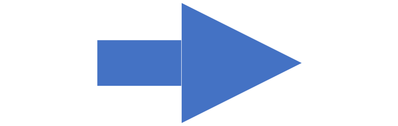
Classification Model from Time Series Data

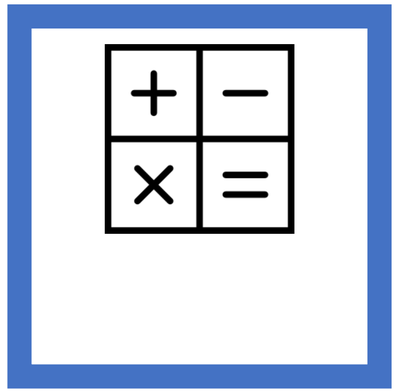


9:14:37 PM	Sleep
9:14:38 PM	Sleep
9:14:39 PM	Sleep
9:14:40 PM	Sleep
9:14:41 PM	Sleep
9:14:42 PM	Sleep
9:14:43 PM	Sleep
9:14:44 PM	Sleep
9:14:45 PM	Sleep
9:14:46 PM	Sleep
4:46:08 PM	Eat
4:46:09 PM	Eat
4:46:10 PM	Eat
4:46:11 PM	Eat
4:46:12 PM	Eat
4:46:13 PM	Eat
4:46:14 PM	Eat
4:46:15 PM	Eat

Regression Model

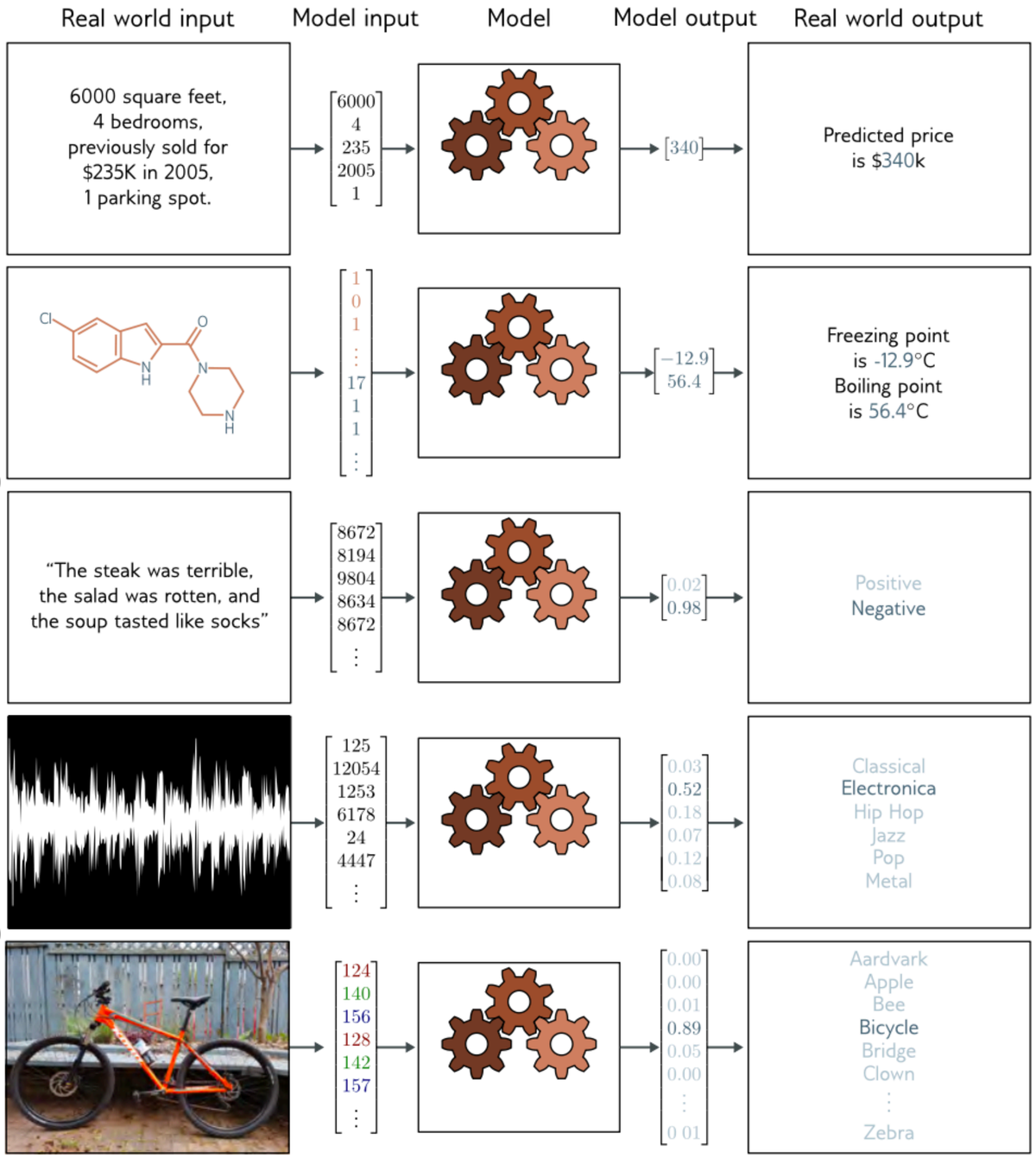
- Area
- Age
- Bedrooms
- Bathrooms
- Parking area



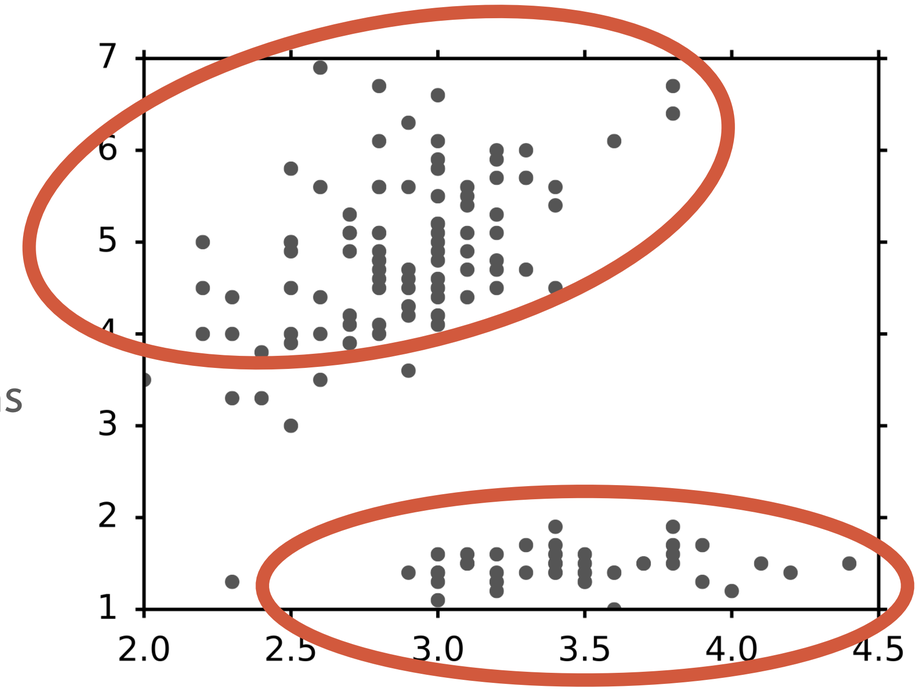


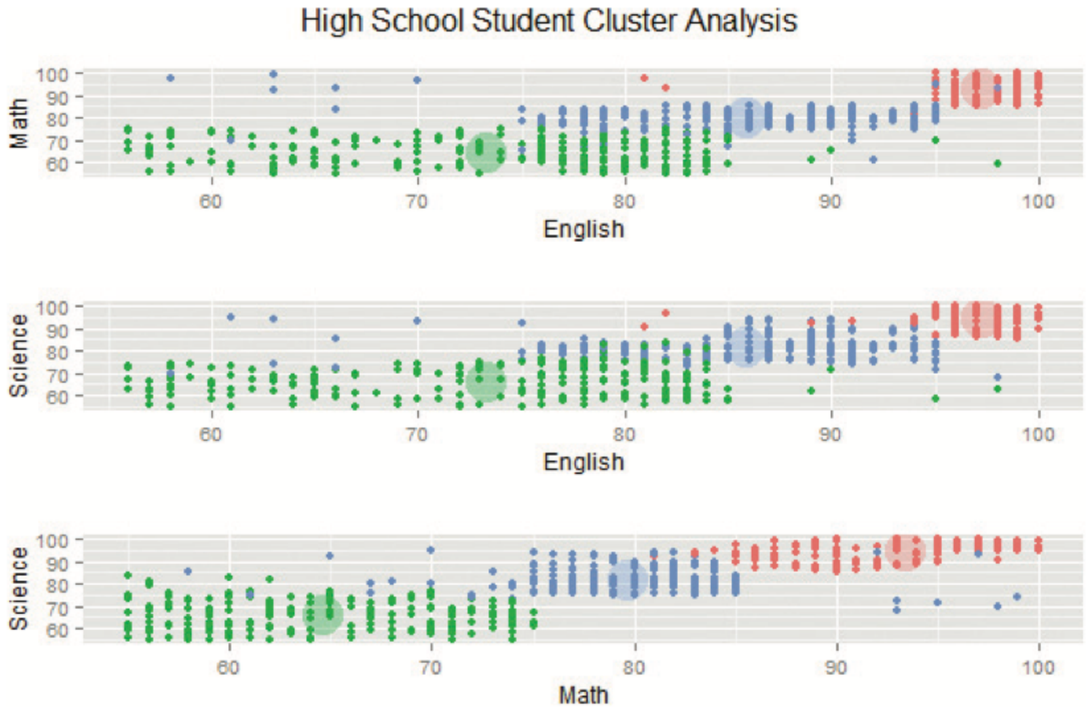
Price of property

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_2 x_3 + \beta_4 x_4 + \beta_5 x_5 + \epsilon$$


Real world input
Model input
Model
Model output
Real world output
6000 square feet,
4 bedrooms,
previously sold for
$235K in 2005,
1 parking spot.
6000
4
235
2005
1
340
Predicted price
is $340k
Cl
O
N
H
N
N
H
1
0
1
17
1
1
−12.9
56.4
Freezing point
is -12.9°C
Boiling point
is 56.4°C
"The steak was terrible,
the salad was rotten, and
the soup tasted like socks"
8672
8194
9804
8634
8672
0.02
0.98
Positive
Negative
125
12054
1253
6178
24
4447
0.03
0.52
0.18
0.07
0.12
0.08
Classical
Electronica
Hip Hop
Jazz
Pop
Metal
124
140
156
128
142
157
0.00
0.00
0.01
0.89
0.05
0.00
0 01
Aardvark
Apple
Bee
Bicycle
Bridge
Clown
Zebra

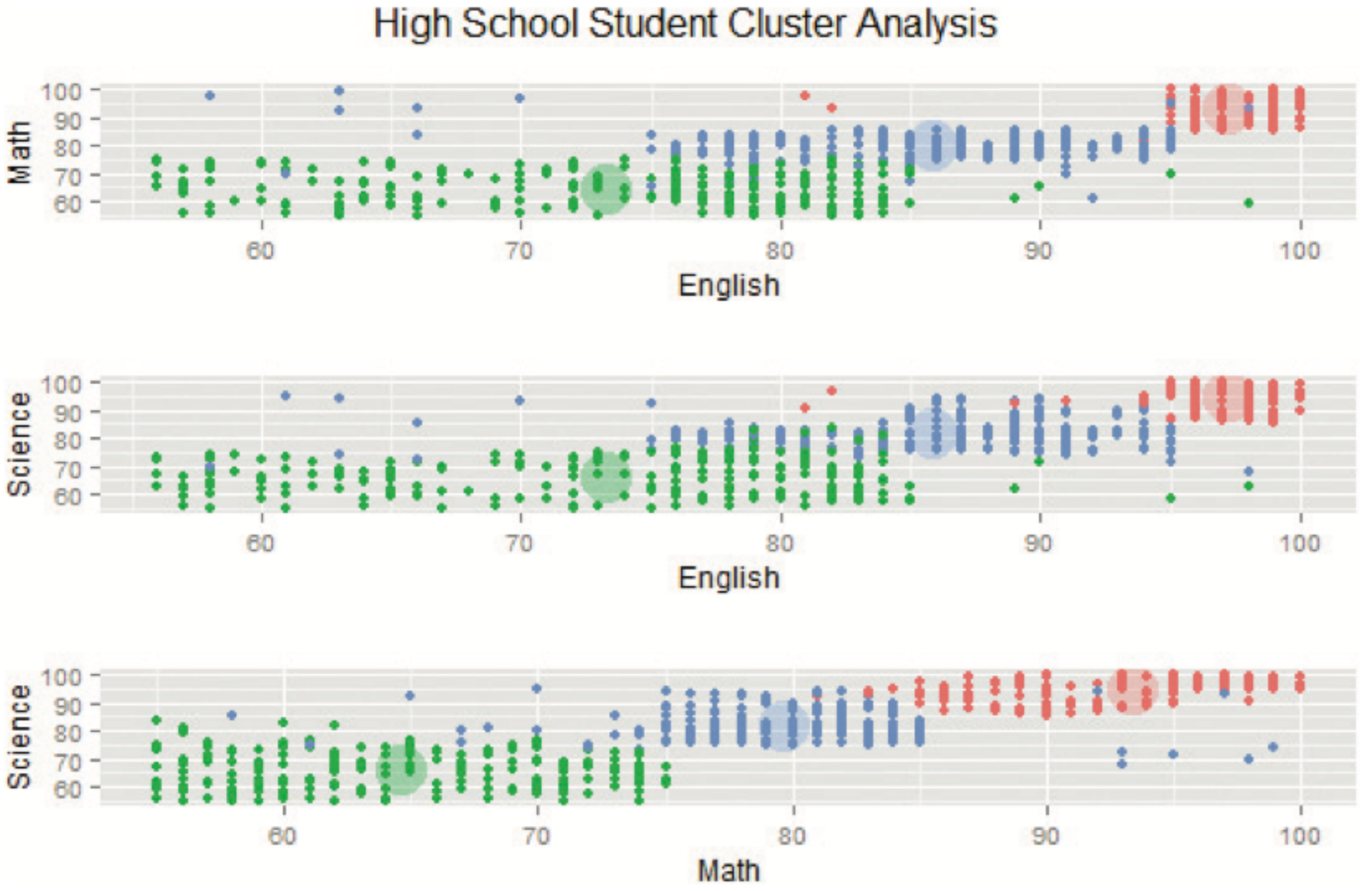
Cluster/Segmentation Analysis





English	Math	Science
99	96	97
99	96	97
98	97	97
95	100	95
95	96	96
96	97	96
100	96	97
95	98	98
98	96	96
99	99	95

English	Math	Science
99	96	97
99	96	97
98	97	97
95	100	95
95	96	96
96	97	96
100	96	97
95	98	98
98	96	96
99	99	95



Dimensionality Reduction

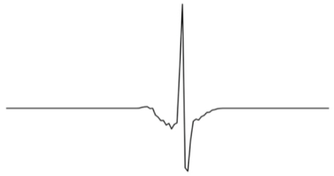
Lots of variables

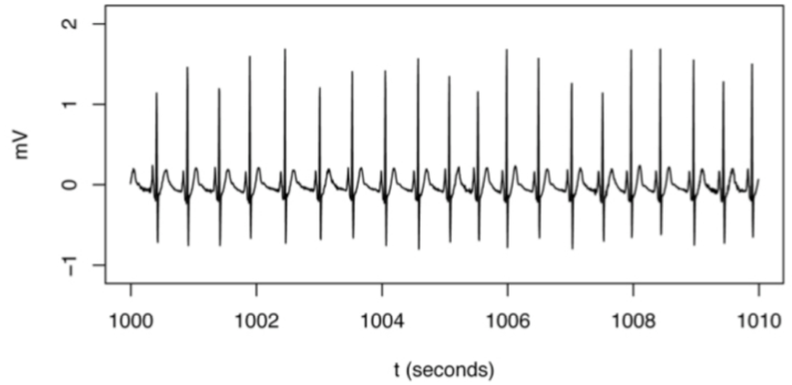
Row ID	Order ID	Order Date	Order Priority	Order Quantity	Sales	Discount	Ship Mode	Profit	Unit Price	Shipping Cost
1	3	10-13-10	Low	6	261.54	0.04	Regular Air	-213.25	38.94	35
49	293	10-1-12	High	49	10123.02	0.07	Delivery Truck	457.81	208.16	68.02
50	293	10-1-12	High	27	244.57	0.01	Regular Air	46.71	8.69	2.99
80	483	07-10-11	High	30	4965.7595	0.08	Regular Air	1198.97	195.99	3.99
85	515	08-28-10	Not Specified	19	394.27	0.08	Regular Air	30.94	21.78	5.94
86	515	08-28-10	Not Specified	21	146.69	0.05	Regular Air	4.43	6.64	4.95
97	613	06-17-11	High	12	93.54	0.03	Regular Air	-54.04	7.3	7.72
98	613	06-17-11	High	22	905.08	0.09	Regular Air	127.70	42.76	6.22
103	643	03-24-11	High	21	2781.82	0.07	Express Air	-695.26	138.14	35
107	678	02-26-10	Low	44	228.41	0.07	Regular Air	-226.36	4.98	8.33
127	807	11-23-10	Medium	45	196.85	0.01	Regular Air	-166.85	4.28	6.18
128	807	11-23-10	Medium	32	124.56	0.04	Regular Air	-14.33	3.95	2
134	868	06-8-12	Not Specified	32	716.84	0	Regular Air	134.72	21.78	5.94
135	868	06-8-12	Not Specified	31	1474.33	0.04	Regular Air	114.46	47.98	3.61
149	933	08-4-12	Not Specified	15	80.61	0.02	Regular Air	-4.72	5.28	2.99
160	995	05-30-11	Medium	46	1815.49	0.03	Regular Air	782.91	39.89	3.04
161	998	11-25-09	Not Specified	16	248.26	0.07	Regular Air	93.80	15.74	1.39
175	1154	02-14-12	Critical	44	4462.23	0.04	Delivery Truck	440.72	100.98	26.22
176	1154	02-14-12	Critical	11	663.784	0.25	Regular Air	-481.04	71.37	69
203	1344	04-15-12	Low	15	834.904	0.06	Regular Air	-11.68	65.99	5.26
204	1344	04-15-12	Low	18	2480.9205	0.01	Regular Air	313.58	155.99	8.99
213	1412	03-12-10	Not Specified	13	59.03	0.1	Express Air	26.92	3.69	0.5
214	1412	03-12-10	Not Specified	21	97.48	0.05	Regular Air	-5.77	4.71	0.7
229	1539	03-9-11	Low	33	511.83	0.1	Regular Air	-172.88	15.99	13.18
230	1539	03-9-11	Low	38	184.99	0.05	Regular Air	-144.55	4.89	4.93
231	1540	08-4-12	High	30	80.9	0.09	Regular Air	5.76	2.88	0.7

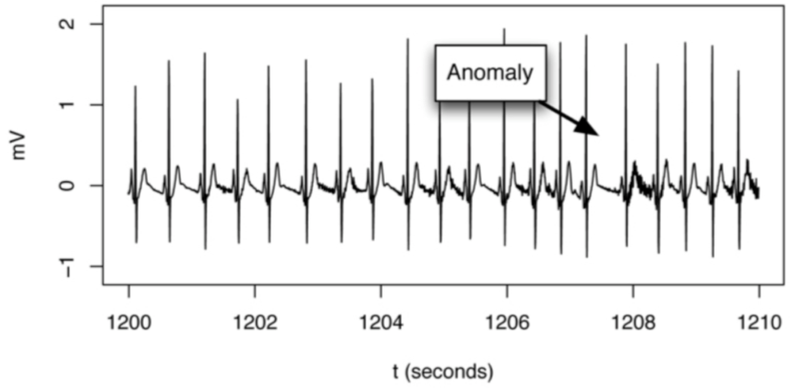
Principal component analysis
Manifold learning
Autoencoder
Clustering
Factor analysis

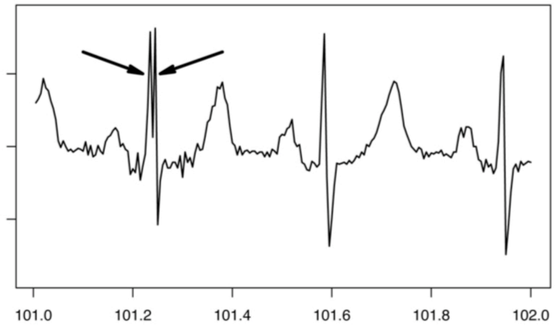
Few significant transformed features

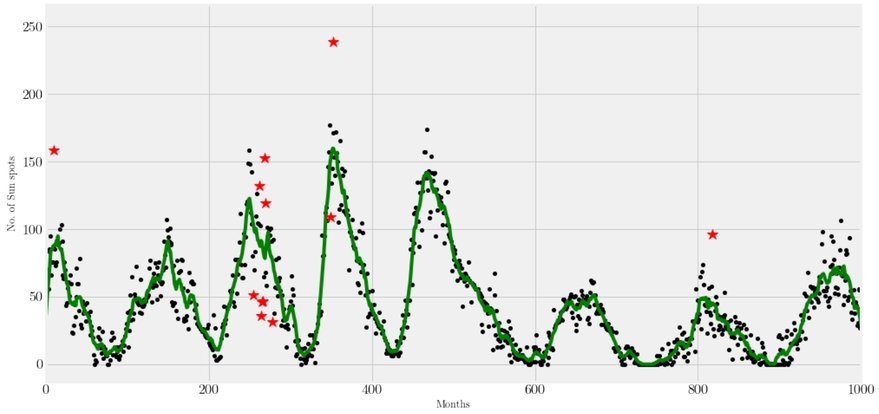
Anomaly Detection

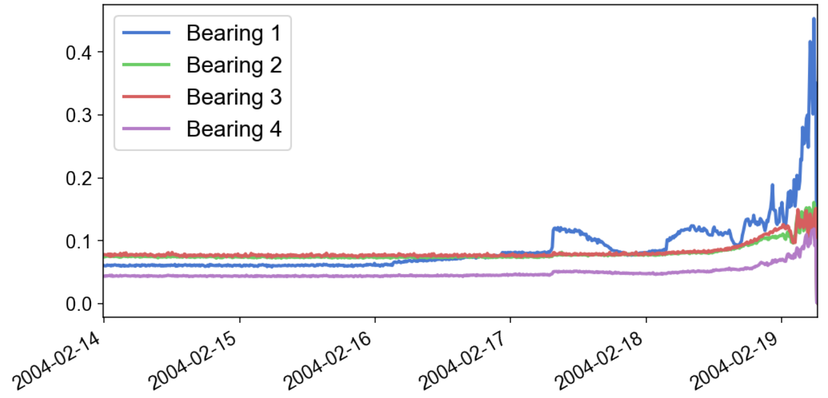


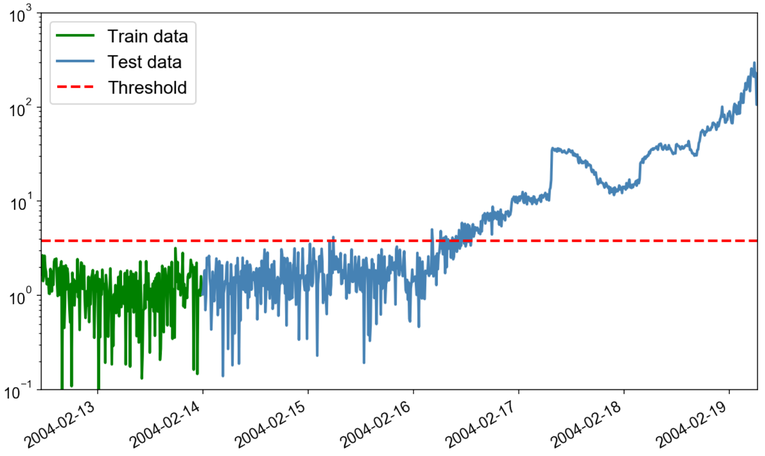












Most Basic Machine Learning

An agent has been selling 300 houses in the last years and want to be able to predict the price of a house by just knowing the size of the house.

In general,

x (size) called **feature** (input)

y (price) called **target** (output)

i: sample index

For a quantitative response y, we assume that y is related to x by an ***unknown function*** $f(\cdot)$ and ***zero-mean errors*** ϵ by

$$y = f(\boldsymbol{x}) + \epsilon$$

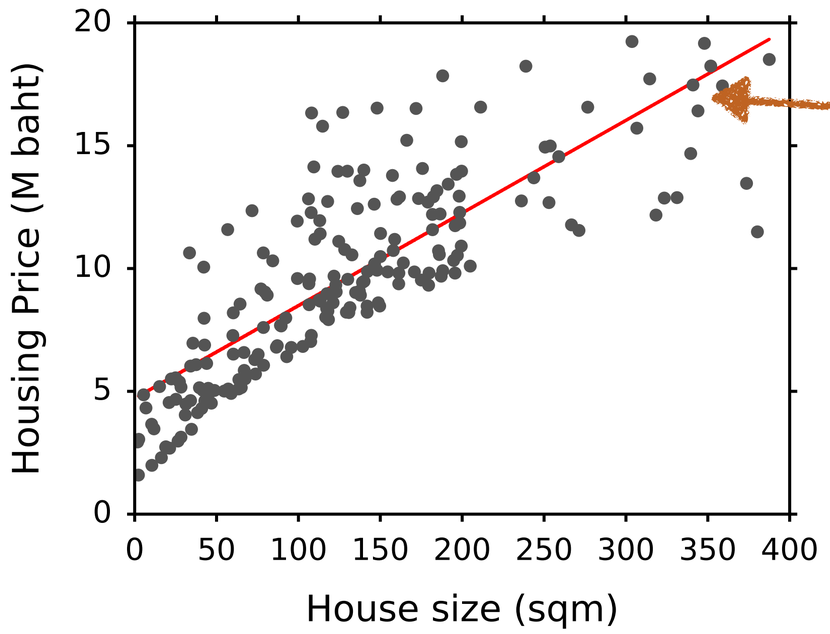
	x	y
i	Size (m^2)	Price (Mbaht)
1	50 $= x_1$	1.4 $= y_1$
2	128 $= x_2$	2.6 $= y_2$
3	24 $= x_3$	0.8 $= y_3$
4	78 $= x_4$	1.2 $= y_4$
i	... $= x_i$	... $= y_i$

Model-based Learning - Parametric Model

A ***model*** is a hypothesized function $h(\cdot)$, or ***hypothesis,*** that we use to approximate $f(\cdot)$ by using a ***learning algorithm***.

The learning task is to fit $h(\cdot)$ to obtain $\hat{f}(\cdot)$ from example data.

$\hat{f}$ can then be used for *prediction* or *inference*.



Parametric model

$$y \approx h_\theta(x) = \beta_0 + \beta_1 x, \quad \theta = (\beta_1, \beta_2)$$

(Trainable) Model parameters

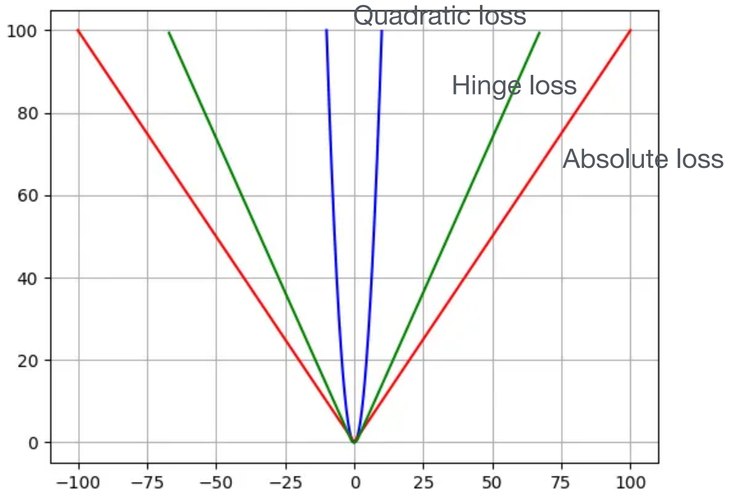
Linear model: $y(x)$ as a linear combinations of model parameters.

Loss Function

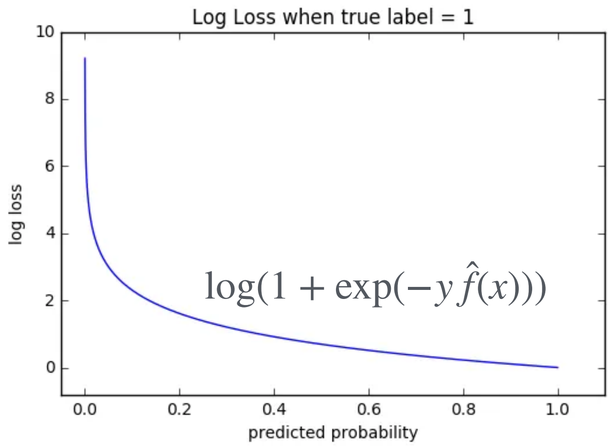
Function $L(\hat{y}, y) : (\hat{y}, y) \rightarrow R$ signifies the error between the predicted value $\hat{y} = \hat{f}(x)$ and the actual value y.

Measures the error from a *single training sample*.

Examples: Quadratic loss (Squared error), Absolute error, Log loss



$e = \hat{y} - y$



$\hat{y}$

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\,|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

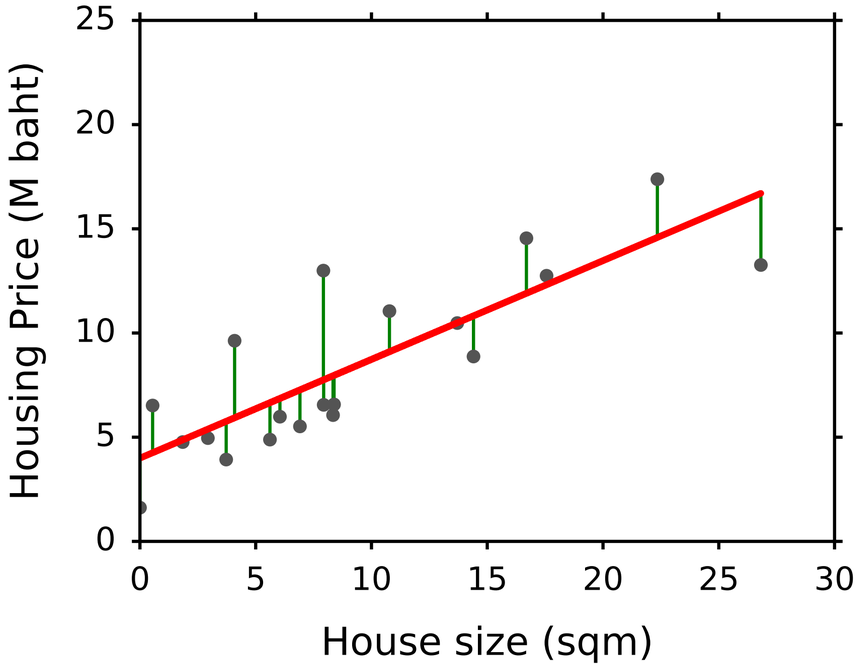
Cost Function

Cost function J is an *average loss* function for the dataset after model training.

Measure how well the model fits the training dataset for the specified model parameters θ and loss function L.

Ex: a common cost function for regression is the ***'mean squared error'*** (MSE) function.

Depending on a learning algorithm, the cost function can be ***local*** or ***global***.



$$J(\theta) = n^{-1} \sum_{i=1}^{n} L(h_\theta(x_i), y_i), \quad \theta = (\beta_1, \beta_2)$$

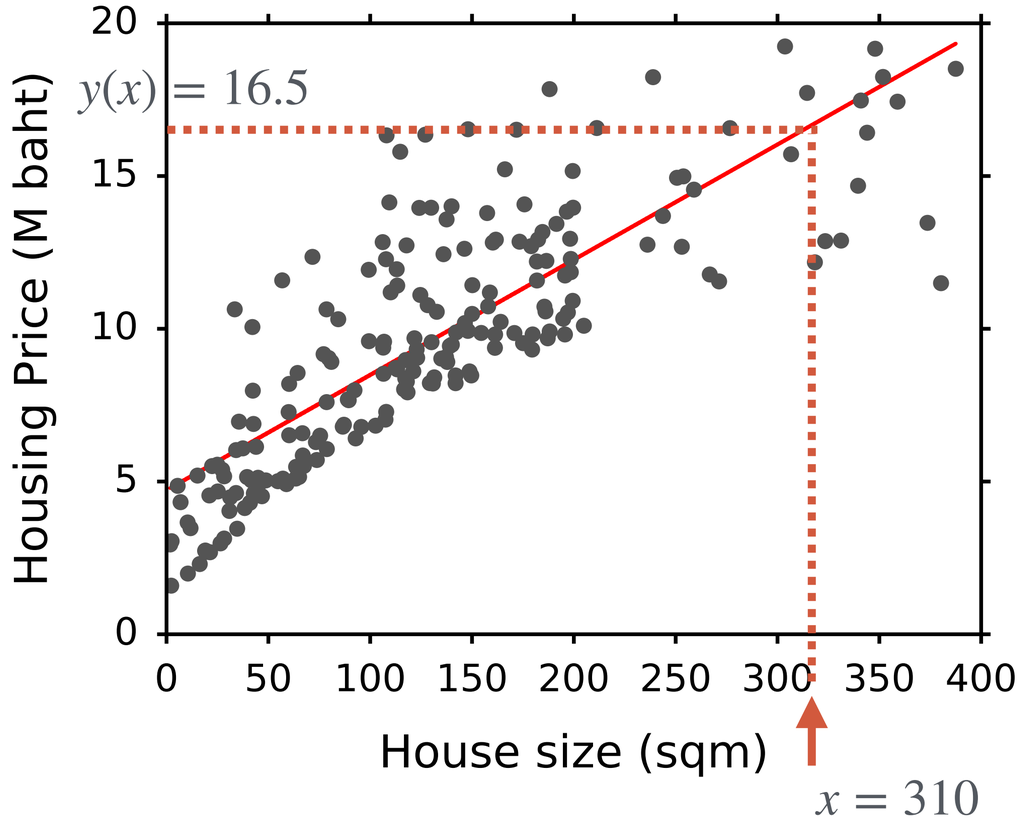
$$= n^{-1} \sum_{i=1}^{n} (h_\theta(x_i) - y_i)^2$$

$$= n^{-1} \sum_{i=1}^{n} (\beta_0 + \beta_1 x_i - y_i)^2$$

Model Training/Fitting

Training ML models = Finding values of the model parameters θ that minimize the cost function J by applying optimization algorithms.

Note that the term ***model*** may refer to as hypothesized function $h(\cdot)$ or the fitted model $\hat{f}(\cdot)$.



Hypothesized fn: $h(x; \beta_0, \beta_1) = \beta_0 + \beta_1 x$

Fitted model: $\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x = 4.717 + 0.038x$

In this case, the value of the cost function also quantifies how well the model predicts the data -- **Train error**

Mean Squared Error (MSE) : $J(\hat{\beta}_0, \hat{\beta}_1) = n^{-1} \sum_{i=1}^{n} (\hat{\beta}_0 + \hat{\beta}_1 x_i - y_i)^2$

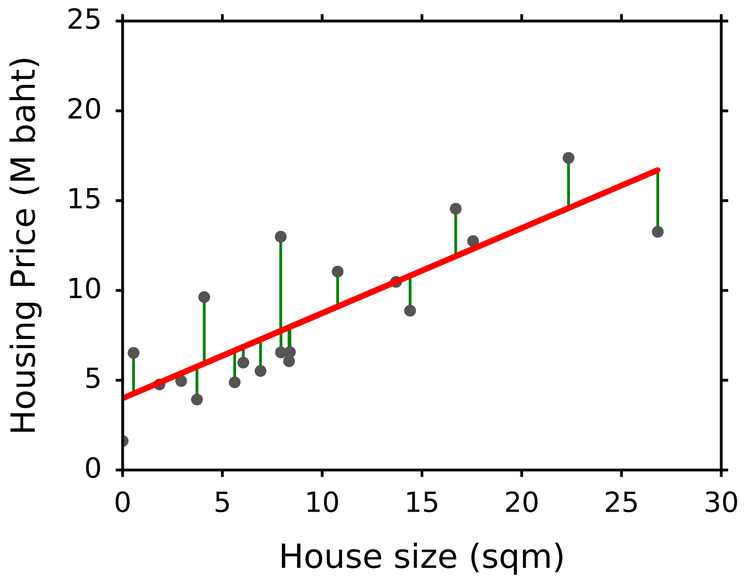
☐ The goal is to find $\hat{f}$ that minimizes

$$\begin{aligned}\mathbb{E}\{(y-\hat{y})^2\} &= \mathbb{E}\{(f(x)+\epsilon-\hat{f}(x))^2\} \\ &= \mathbb{E}\{(f(x)-\hat{f}(x))^2\} + \mathbb{V}\text{ar}\{\epsilon\}\end{aligned}$$

Fitted model: $\hat{y} = \hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$

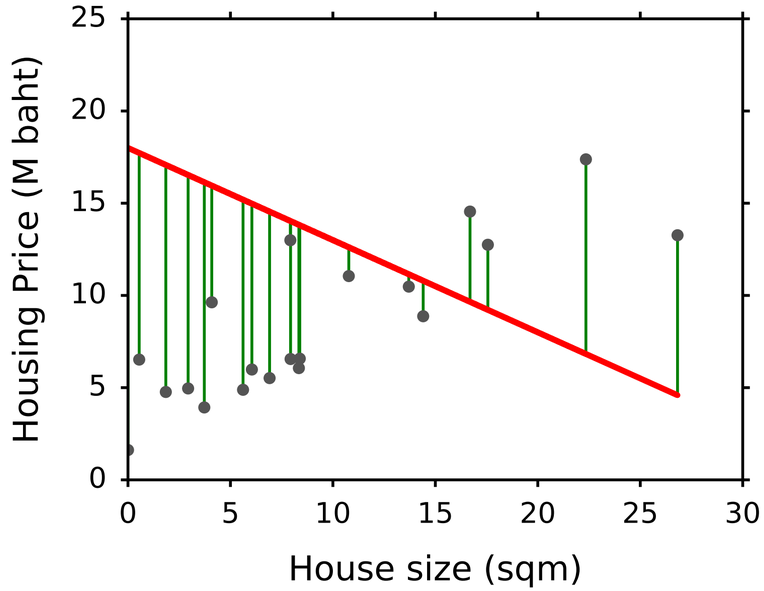
$\hat{\beta}_0 = 4.717$

$\hat{\beta}_1 = 0.038$



$\hat{\beta}_0 = 20$

$\hat{\beta}_1 = -0.05$



Learning Definition

How do you tell if a machine "learns" ?

> A computer program is said to learn from **experience E**
> with respect to some class of **tasks T** and **performance measure P**,
> if its performance at tasks in T, as measured by P, improves with experience E.

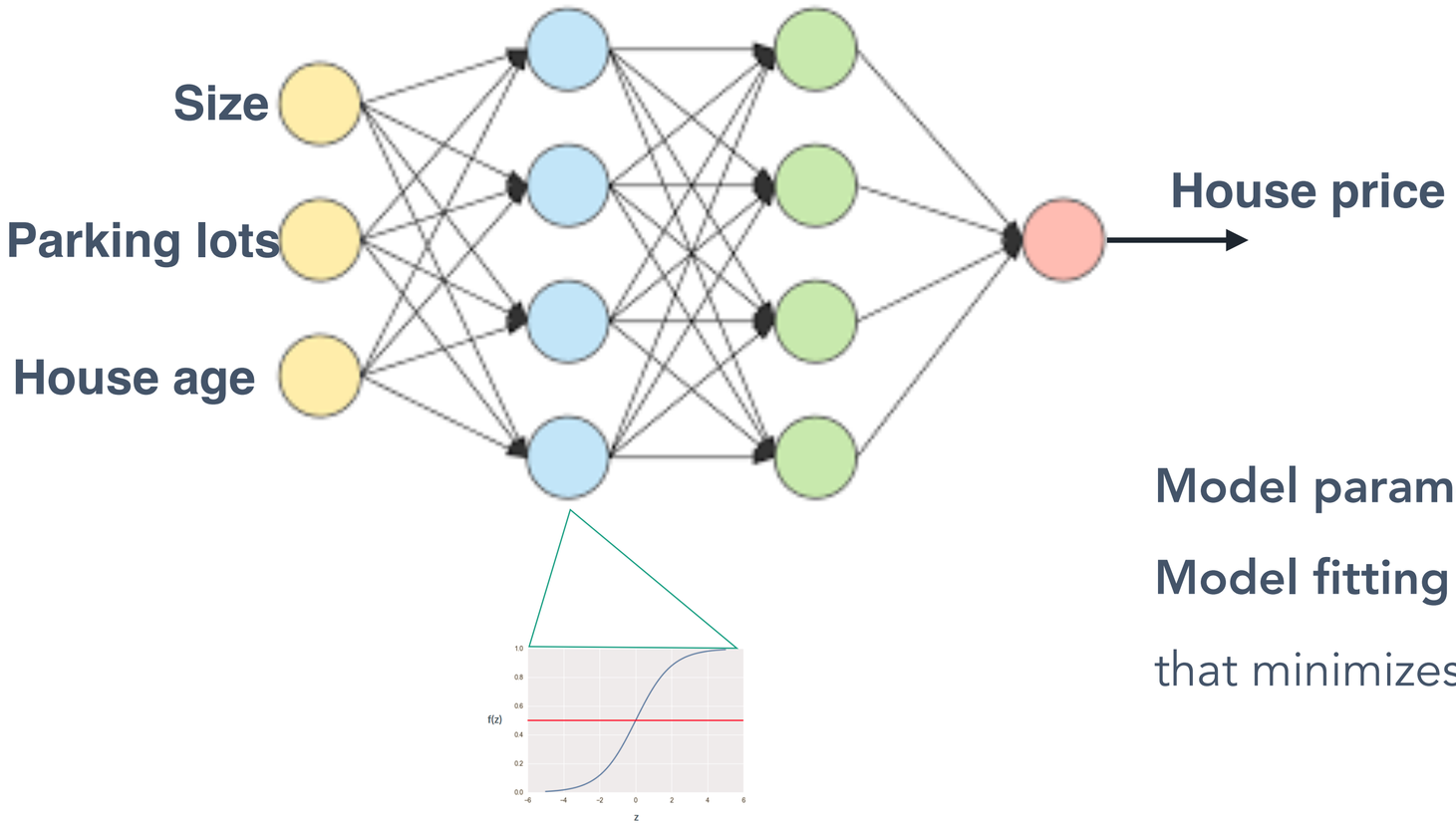
Ex: Linear regression

Task T Predict y from x using $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 = \boldsymbol{\beta}^T \boldsymbol{x}$

Measure P $J = n^{-1} \sum_i (\hat{y}_i - y_i)^2$

Experience E Use available data to determine $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$

More Complex (Non-linear) Model: Neural Networks



Model parameters = Connection weights

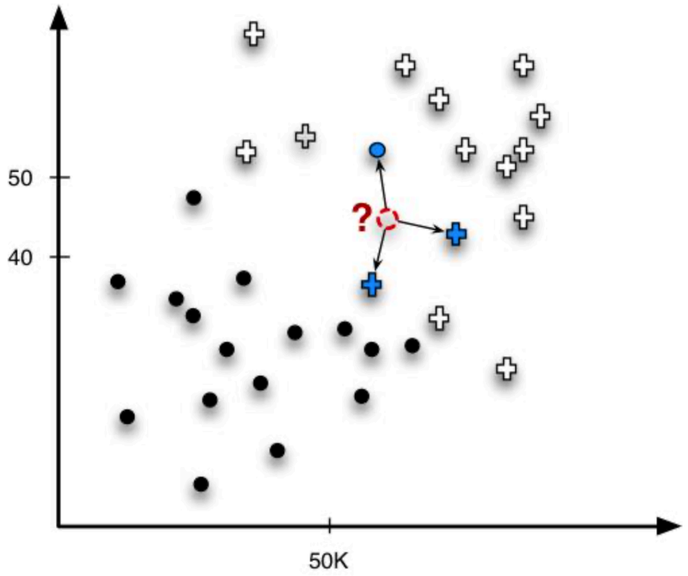
Model fitting = Find a set of weights that minimizes the cost function.

Instance-based / Memory-based Learning - Non-Parametric Model

Predict outputs by explicitly quantifying the relationship between input instances with those in the train data.

- No model training/fitting actually happens.
- Computation on the dataset is performed at the prediction time, hence "***lazy learning***".

Example: K-Nearest Neighbors (KNN) classification and regression.



$$\Pr(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j).$$

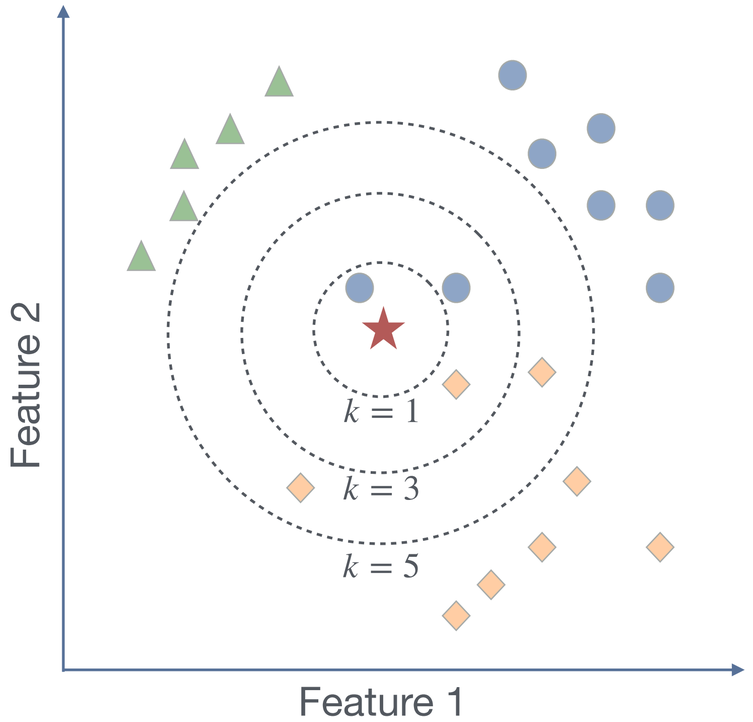
$$\hat{f}(x) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} y_i$$

Hyperparameters

Parameters that must be specified prior to model learning or usage:

- Ex; Max tree height, # layers in a neural network, K in the KNN algorithm.
- Proper values are critical to the model performance.

Best-performing model is determined from a set of models with different hyperparameter settings.

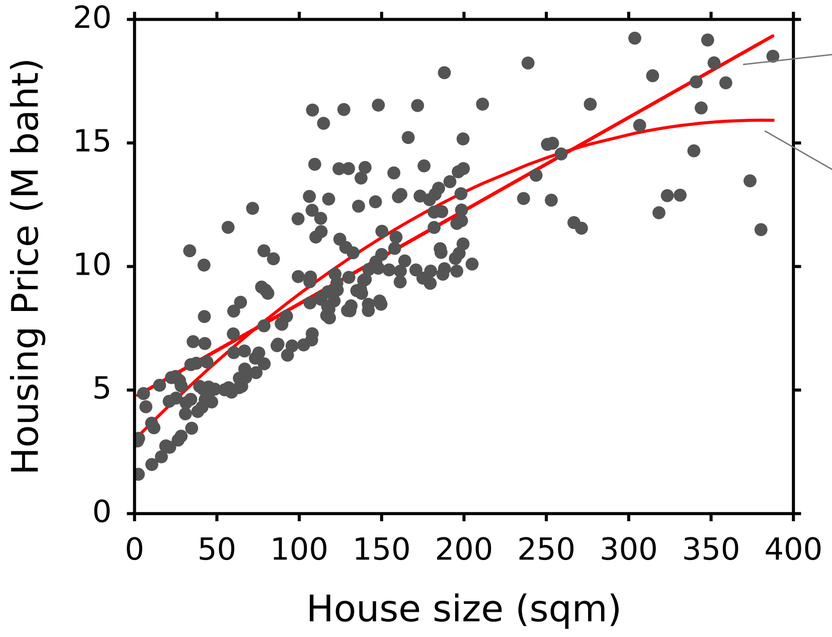


Model Complexity and Flexibility

Model complexity is increased by adding more terms or using complex forms/architectures, referred to as ***degrees of freedom*** available to the model.

A model is ***flexible*** if it has a capacity to learn a wide range of input patterns (e.g., has more shapes).

Model flexibility typically increases with its complexity.



$h_1(x) = \beta_0 + \beta_1 x$ (less complex model)

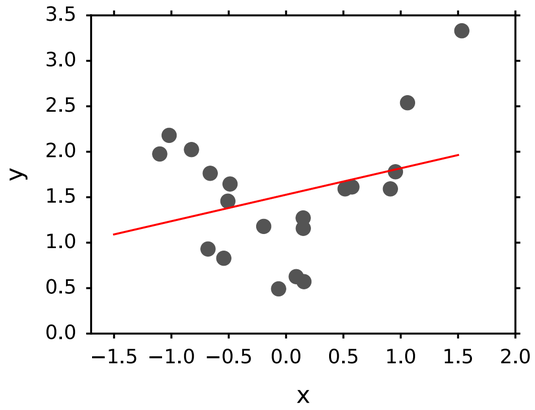
$h_2(x) = \beta_0 + \beta_1 x + \beta_2 x^2$ (more complex model)

Underfitting

Model is too simple and cannot learn the pattern generated by (true) f.

Underfit

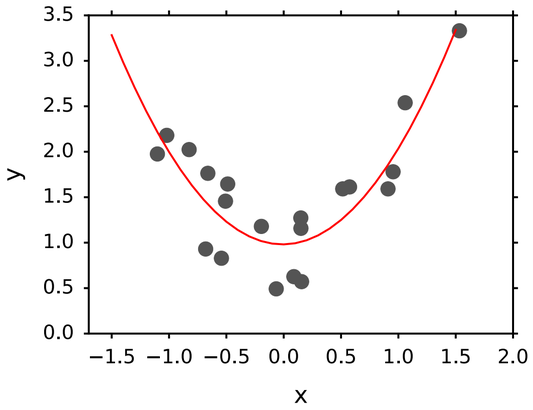
Train error = 0.21



$$h(x) = \beta_0 + \beta_1 x$$

Just right

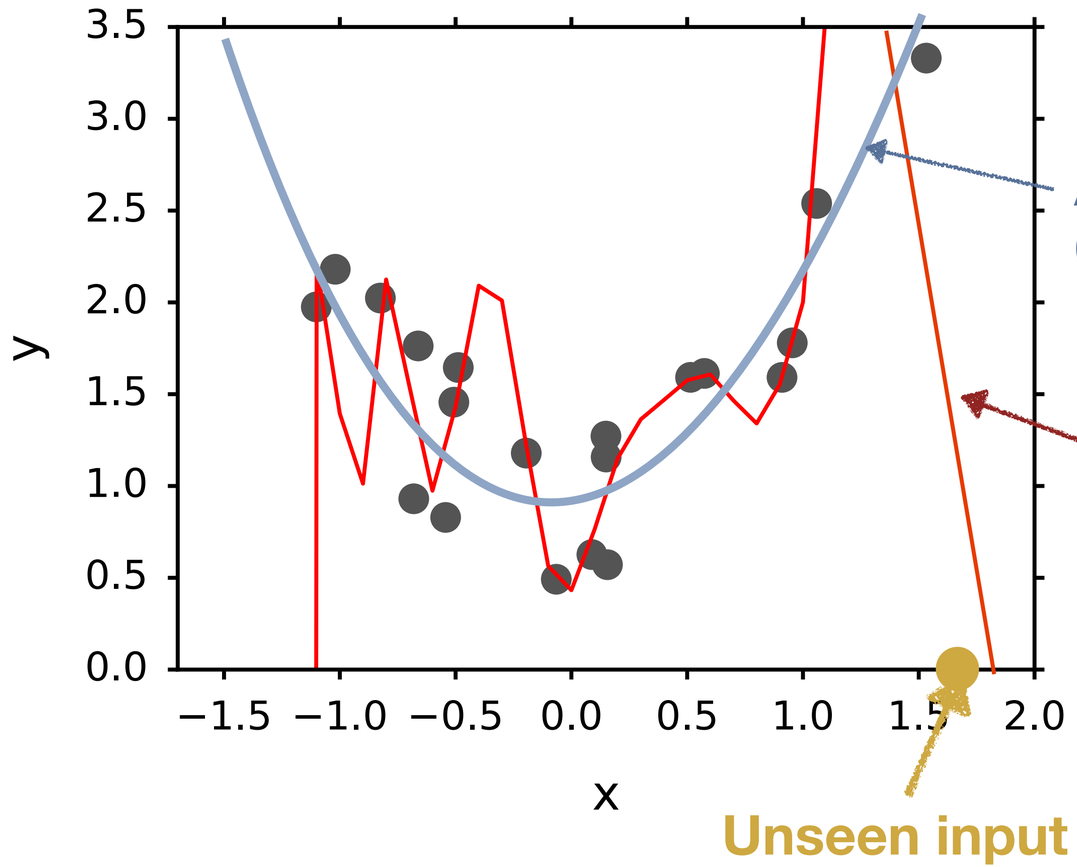
Train error = 0.05



$$h(x) = \beta_0 + \beta_1 x + \beta_2 x^2$$

Overfitting

Model is too flexible and learns the pattern too well to generalize !



Train error very small

Prediction error for unseen input very high !

$$h(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_{14} x^9 + \beta_{15} x^{10}$$

Model Development Goals

Model Evaluation: How do we credibly evaluate the true generalization performance ?

Model Selection: How do we find best performing model produced from a set of models produced by different hyperparameter settings, forms, or sets of features ?

Basic Model Evaluation -- Hold-out Validation Method

How do we quantify how well $\hat{f}$ is a good approximation of (true) f ?

Using train errors (***resubstitution validation/evaluation***) results in optimistic bias due to overfitting.

Train a model with one set of data (***train data***) and evaluate it on another set of data (***test data***).

Use performance metrics evaluated on the test set to represent the ***generalization performance.***

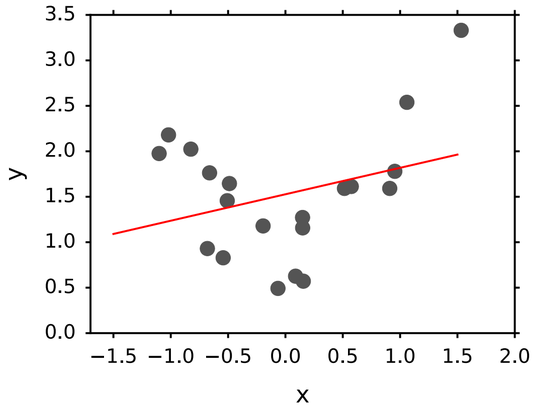
Train set	**Test set**
70-80% of data to training the model.	30-20% of data preserved for testing
Used to evaluate '**Train error**' or '**Train performance**' Prediction error from Train data	Used to evaluate '**Test error**', or '**Test performance**' Prediction error from Test data. (How well the model generalizes)

Evaluating Regression Model

For inputs (x, y) not used to train the model (***test observations***), we want to minimize the ***test MSE*** or ***test error***: Avg $(y - \hat{y})^2$

Underfit

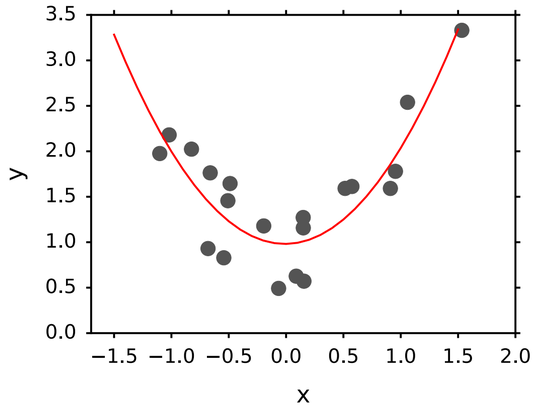
Train error = 0.21



$$h_1(x) = \beta_0 + \beta_1 x$$

Just right

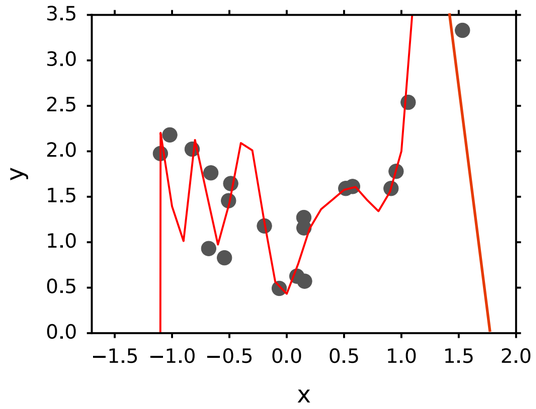
Train error = 0.05



$$h_2(x) = \beta_0 + \beta_1 x + \beta_2 x^2$$

Overfit

Train = 0.02



$$h(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_{14} x^9 + \beta_{15} x^{10}$$

Choosing Right Model Complexity - Bias and Variance

Model with lowest test error is desirable.

For ***fitted model*** $\hat{f}$ and ***unseen input*** x, the expected test MSE is given by

$$\mathbb{E}\{(y - \hat{f}(x))^2\} = \mathbb{V}\text{ar}\{\hat{f}(x)\} + \mathbb{E}^2\{y - \hat{f}(x)\}$$

Average test MSE obtained by repeatedly estimated $\hat{f}$ using different training sets.

Variance of predictions

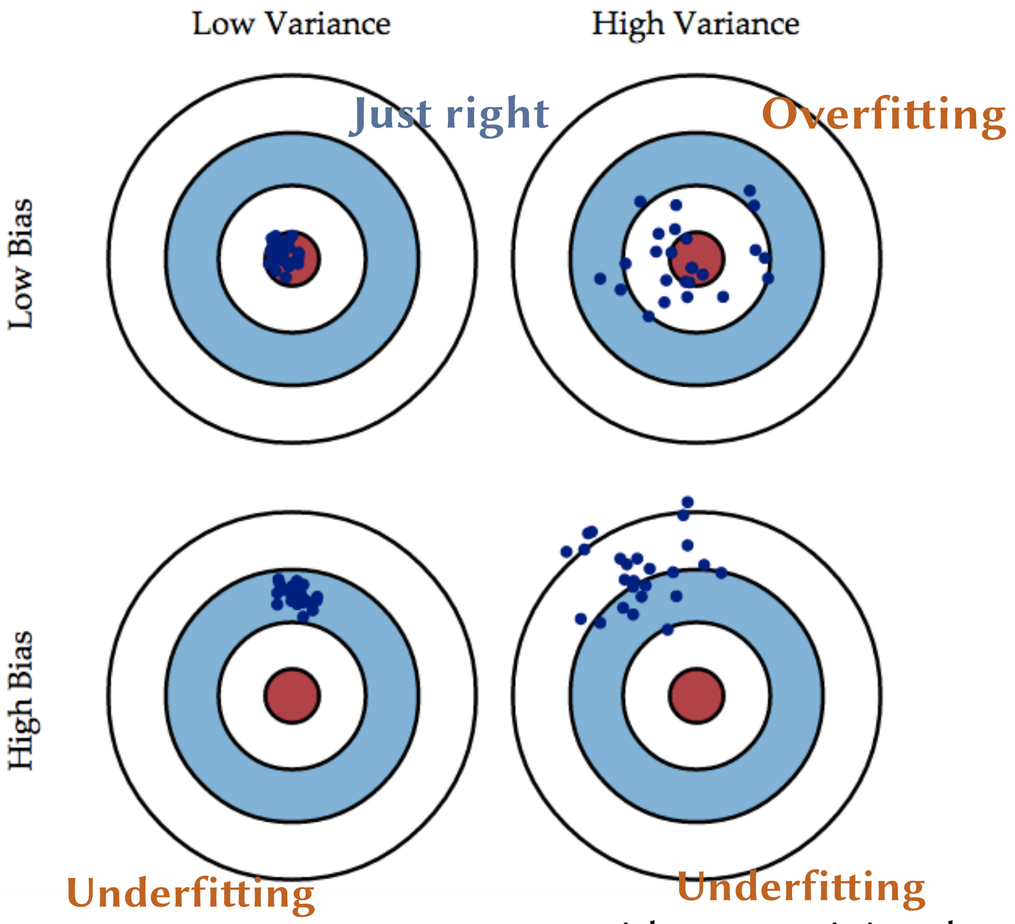
Squared **Bias** (average of test errors)

- Bias = Average of test errors from many $\hat{f}$.
- Variance = Average deviation of test errors from many $\hat{f}$
- Expected test MSE = Squared Bias + Variance

Bias-Variance Tradeoff

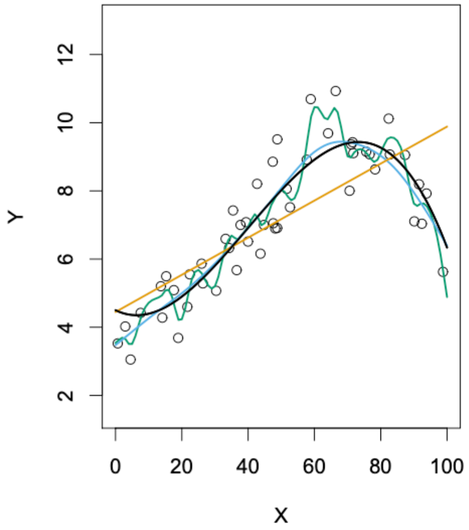
Fig. 7.1 Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2008

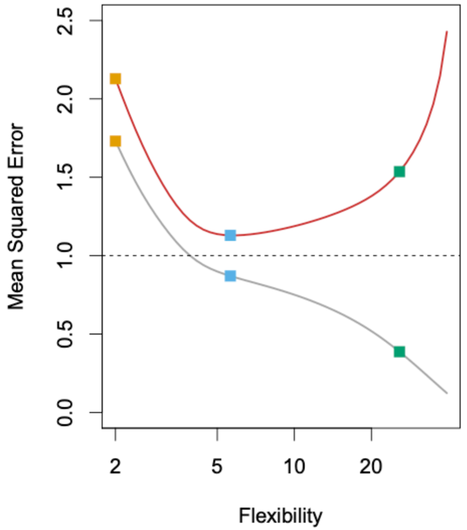
Blue points = Predictions of unseen input x_0 from $\hat{f}$ trained with different training sets.

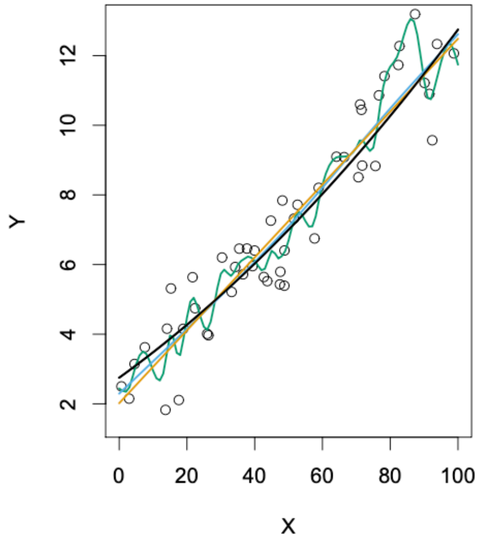


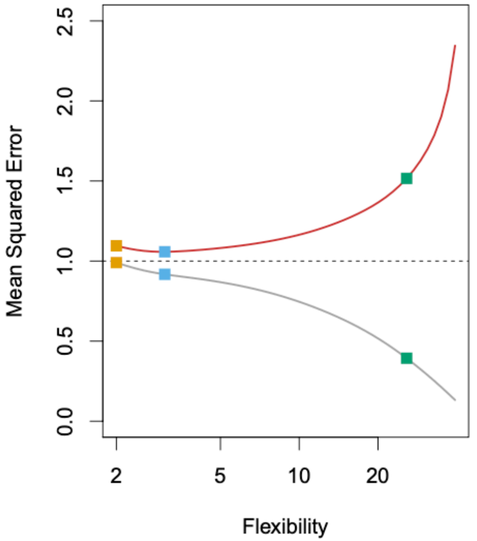
with poor training data

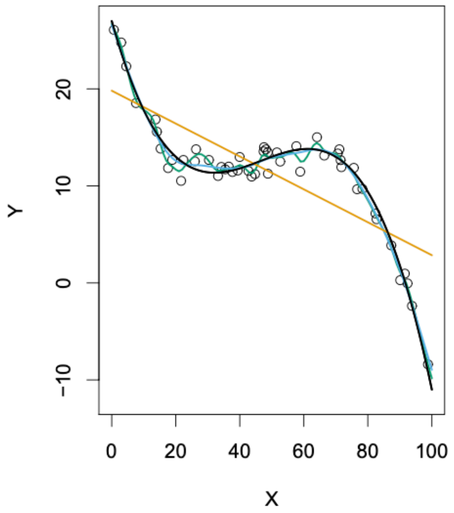
Train-Test Error Curves

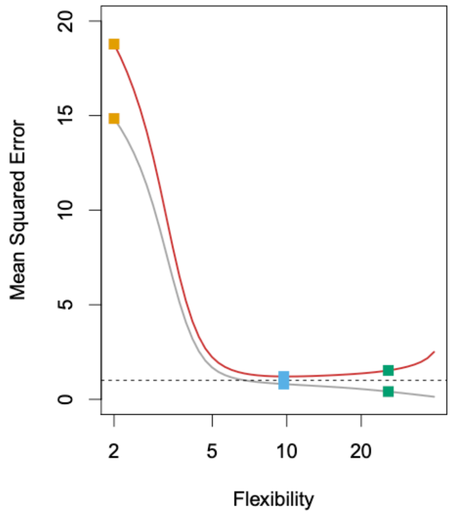




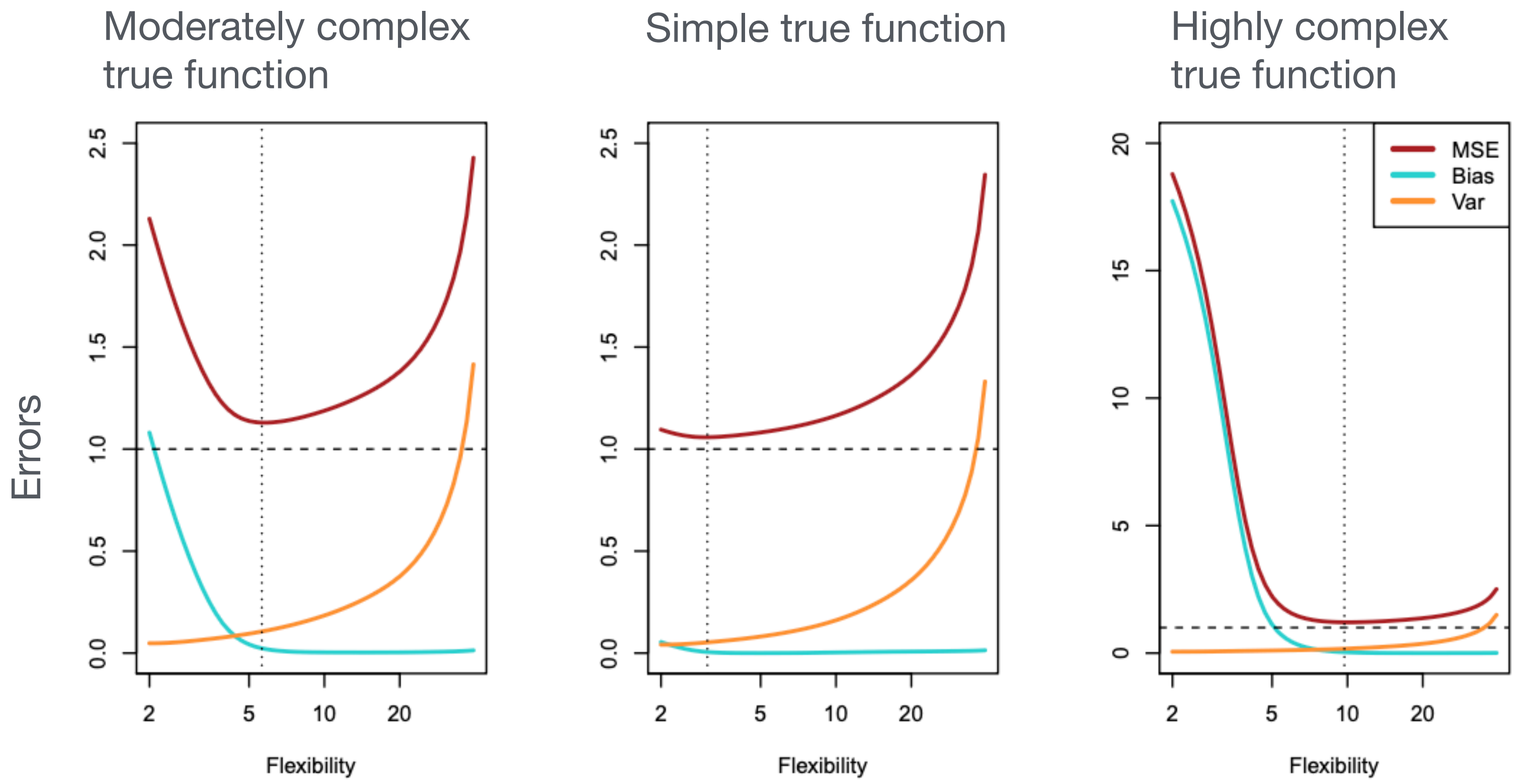








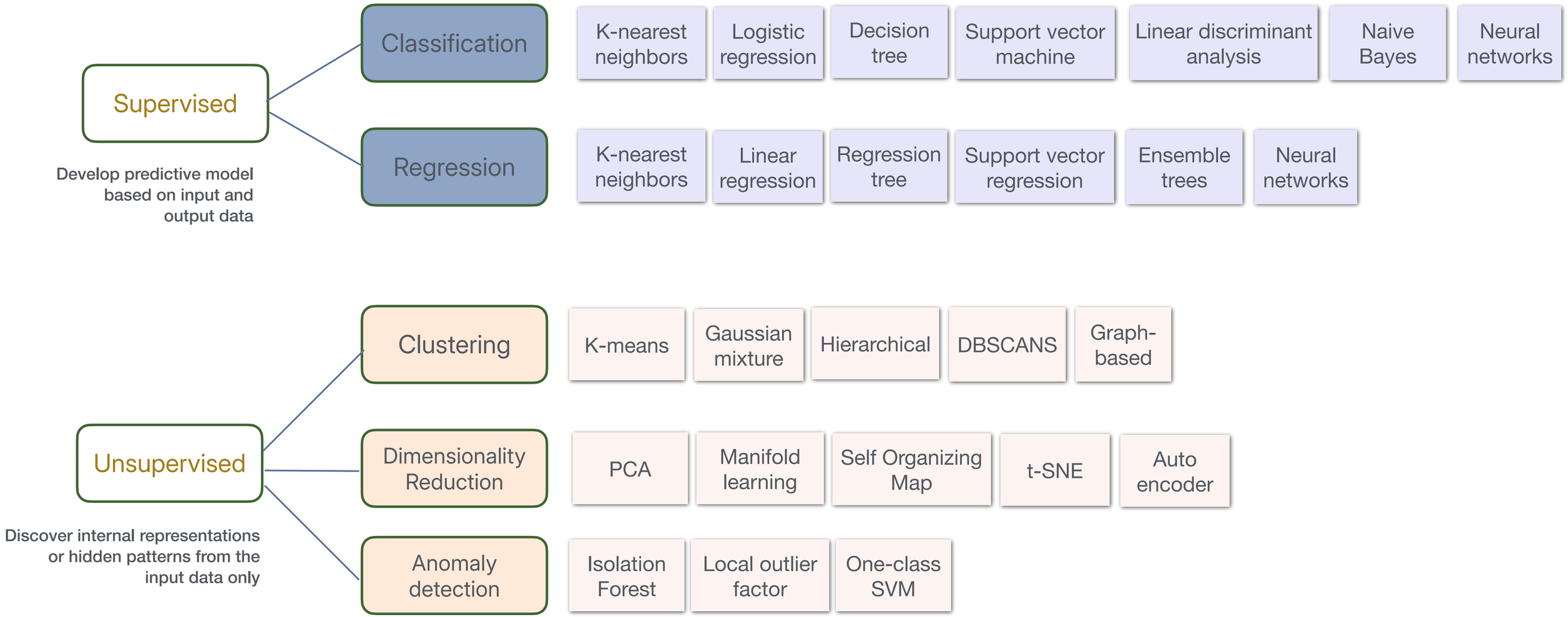
- Bias-variance tradeoff prevents supervised learning algorithms from generalizing beyond their training set.
- The challenge lies in finding a method with just right flexibility (and complexity) so that the variance and the squared bias are low.
- Minimizing both bias and variance is also conflicting, and requires additional techniques to treat the problem.



No-Free-Lunch Theorem

NO ONE METHOD DOMINATES ALL OTHERS OVER ALL POSSIBLE DATA SETS.

Some Basic ML Algorithms



Supplementary Material

Model Selection

Assess the model performance on different combinations of features and hyperparameters.

Select the combination with best performance, i.e., the best performing model.

Feature tuning

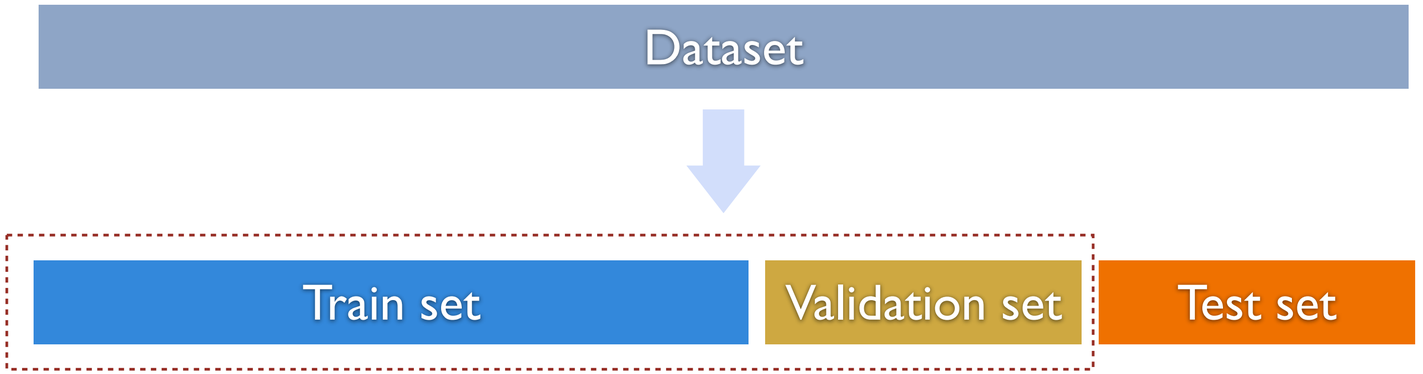
- Try various subsets of input features, e.g., hand-picked, correlation screening, subset enumeration.
- Transform existing features, e.g., Binning, scaling, squares.
- Extract new features from existing ones, e.g., PCA, auto-encoder.
- Create new features from existing ones, e.g., Ratio/product of variables (interaction terms), window statistics, clustering distance.

Hyperparameter tuning

- Ex: Loss function, tree depth, hidden layers, activation function, etc.
- Grid-search and random-search

Three-way Hold-out Method (Train-Validation-Test Split)

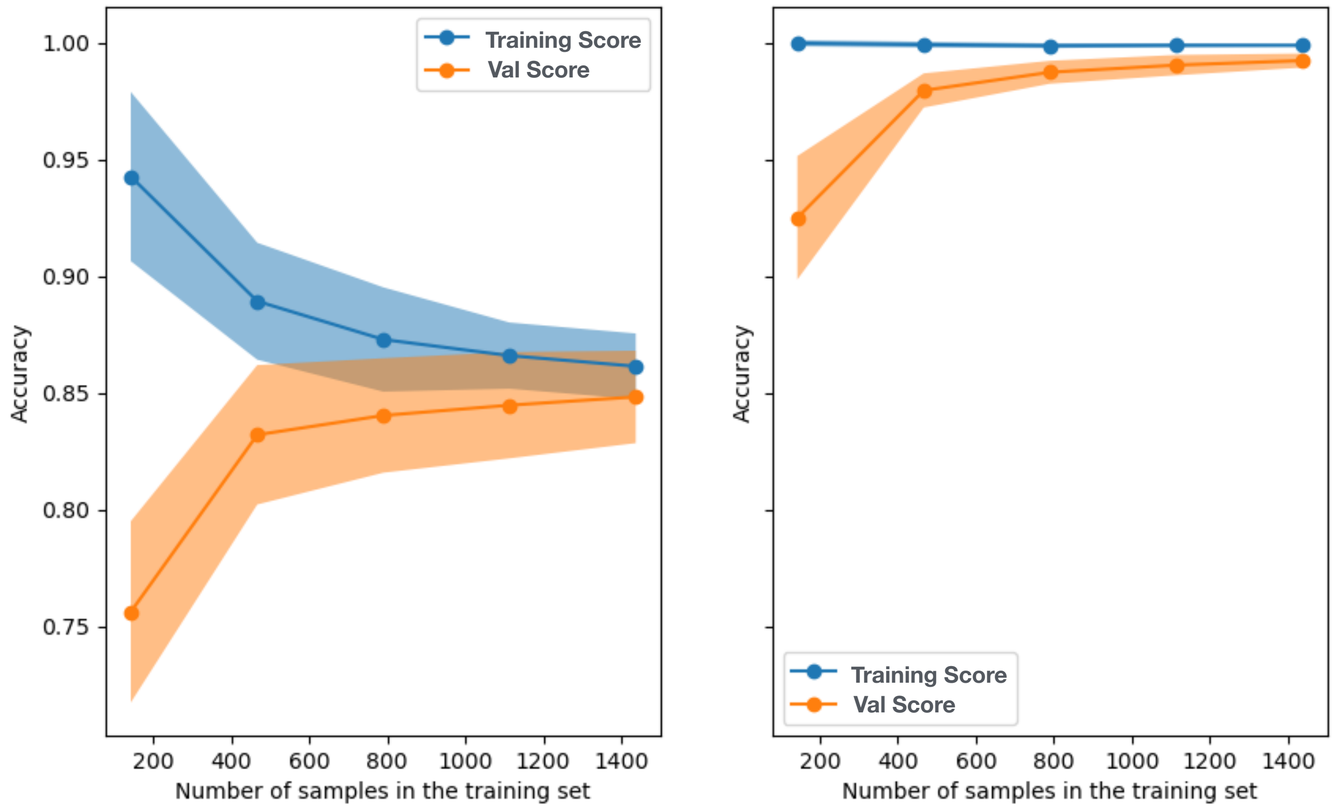
- **Train data**: Used to fit the model over different hyperparameter settings and feature sets.
- **Validation data**: Used to assess the fitted models and decide on the best-performing one.
- **Test data**: Held out until the final model (hypothesis+hyperparameters+features) has been decided.
  - Merge the train and validation data and fit the model.
  - Evaluate the generalization performance of the fitted model on the test set.
  - *Never touch it until arriving at the final model !*



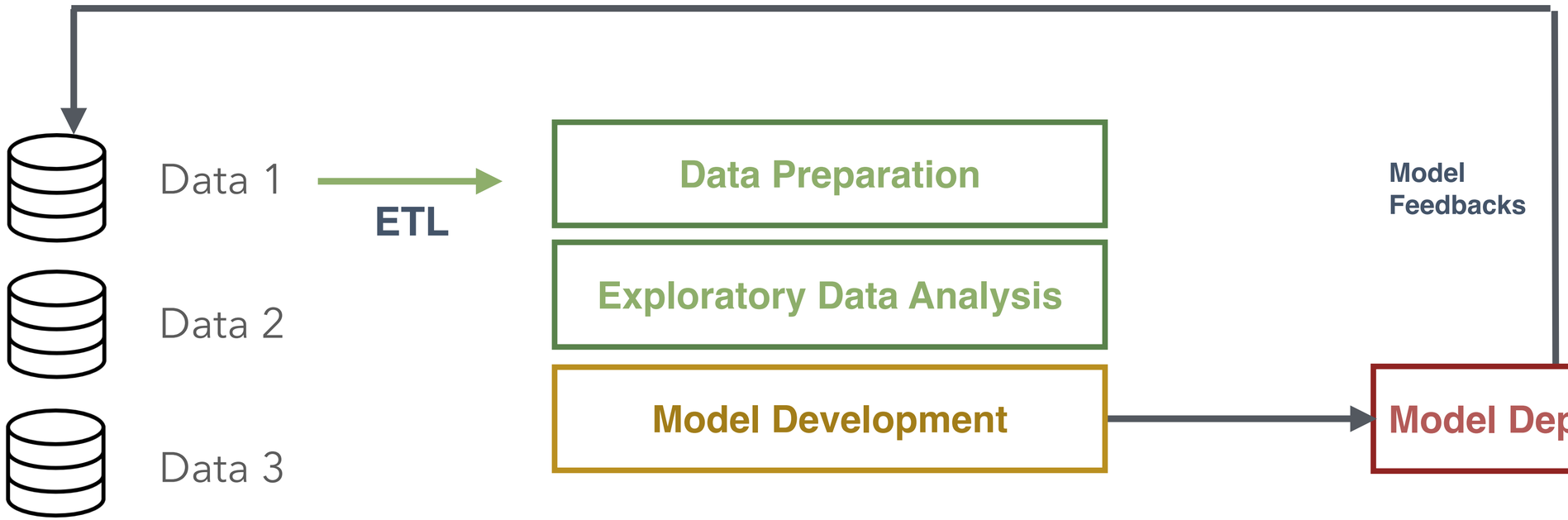
- Use the whole data set to fit the final model for deployment.

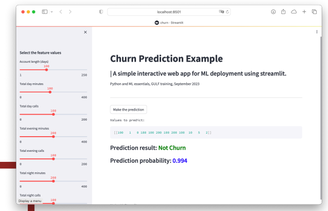
Learning Curve

Train and validation data can be used to assess if more data would help improving the model performance.



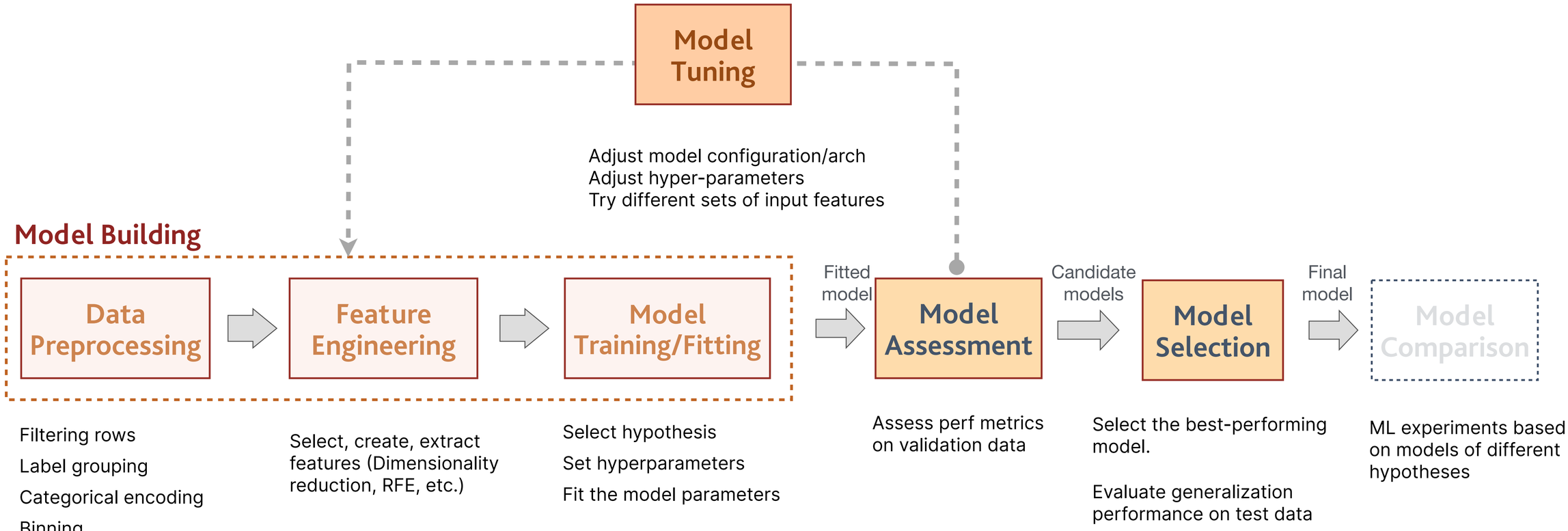
ML Project Life Cycle





- Model equation + parameters packaged
- Processing pipelining
- (Business) user interface
- APIs, load balancing, cloud server
- Model maintenance (monitoring, data updating, retraining)

Model Development Process



Model Comparison -- ML Experiments

Get best-performing models from different hypotheses (learning algorithms) and evaluate their generalization performance.

Compare their generalization performance to select the final model to deploy.

- Two models: Paired-t test, two-sample t test, z-test for proportion difference, McNemar test
- Several models: Cochran's Q test, F test, etc.

Hands-on Data Preprocessing

Data Preprocessing Activity

Activity: Loan Prediction

Column	Description	Type
income	Income of the user	int
age	Age of the user	int
experience	Professional experience of the user in years	int
profession	Profession	string
married	Whether married or single	string
house_ownership	Owned or rented or neither	string
car_ownership	Does the person own a car	string
risk_flag	Defaulted on a loan	string
current_job_years	Years of experience in the current job	int
current_house_years	Number of years in the current residence	int
city	City of residence	string
state	State of residence	string

Modeling Example: Basic Decision Tree in Python

Load and explore data

- Basic information (dimension, column meanings, etc.)
- Missing values, Correlations

Data preparation and preprocessing

- Drop unused or redundant columns
- Label grouping
- Data encoding
- Splitting data

Model fitting

Performance evaluation

Model saving

Age	Income	Jobsatisfaction	Desire	Enrolls
<=30	High	No	Fair	No
<=30	High	No	Excellent	No
31 to 40	High	No	Fair	Yes
>40	Medium	No	Fair	Yes
>40	Low	Yes	Fair	Yes
>40	Low	Yes	Excellent	No
31 to 40	Low	Yes	Excellent	Yes
<=30	Medium	No	Fair	No
<=30	Low	Yes	Fair	Yes
>40	Medium	Yes	Fair	Yes
<=30	Medium	Yes	Excellent	Yes
31 to 40	Medium	No	Excellent	Yes
31 to 40	High	Yes	Fair	Yes
>40	Medium	No	Excellent	No

Load and Explore Data

Basic information

```
df.head()
df.info()
df.describe(include='all')
```

	Age	Income	Jobsatisfaction	Desire	Enrolls
0	<=30	High	No	Fair	No
1	<=30	High	No	Excellent	No
2	31 to 40	High	No	Fair	Yes
3	>40	Medium	No	Fair	Yes
4	>40	Low	Yes	Fair	Yes

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Age              14 non-null     object
 1   Income           14 non-null     object
 2   Jobsatisfaction  14 non-null     object
 3   Desire           14 non-null     object
 4   Enrolls          14 non-null     object
dtypes: object(5)
memory usage: 692.0+ bytes
```

To do data profiling, run `pip install ydata-profiling`

```
import ydata_profiling

df.profile_report()
```

Data Preprocessing

One-hot encoding

```
onehot_columns = ['JobSat','Desire']
df=pd.get_dummies(df, columns=onehot_columns)
```

Label/Ordinal encoding

```
Age_cat = ['<=30', '31 to 40', '>40']
df['Age']=df['Age'].apply(lambda x: Age_cat.index(x))

Income_cat = ['Low', 'Medium', 'High']
df['Income']=df['Income'].apply(lambda x: Income_cat.index(x))

Target_cat = ['No', 'Yes']
df['Target']=df['Target'].apply(lambda x: Target_cat.index(x))
```

Model Fitting

Decision tree in scikit-learn implements Classification and Regression Tree (CART) algorithm.

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_decrease=0.0, class_weight=None, ccp_alpha=0.0*) [source]

```
from sklearn.tree import DecisionTreeClassifier

dt_clf = DecisionTreeClassifier()
dt_clf.fit(X_train, y_train)

# Predict results
dt_clf.predict(X_train[0:2])
dt_clf.predict_proba(X_train[0:2])
```

Performance Evaluation

Accuracy

```
from sklearn.metrics import accuracy_score

accuracy_score(y_train, dt_clf.predict(X_train))
```

Classification report

```
from sklearn.metrics import classification_report

print(classification_report(y_train, dt_clf.predict(X_train)))
```

Confusion matrix

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_train, dt_clf.predict(X_train))
```

K-Fold Cross Validation

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_validate

scoring_metrics = ['accuracy','precision_macro','recall_macro', 'f1_macro']
scores = cross_validate(DecisionTreeClassifier(), X_train, y_train,
                        scoring=scoring_metrics, cv = 5, return_train_score= False)
scores_df = pd.DataFrame(scores)
scores_df
```

	fit_time	score_time	test_accuracy	test_precision	test_recall
0	0.022441	0.008939	0.905782	0.687500	0.647059
1	0.015810	0.006936	0.890792	0.602410	0.735294
2	0.015521	0.007266	0.899358	0.636364	0.720588
3	0.014588	0.007181	0.905579	0.671429	0.691176
4	0.017157	0.007003	0.905579	0.662162	0.720588

Hyperparameter Tuning

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

param_grid = {
    'feature_name1':  [1, 2, 3, 4, 5],
    'feature_name2': [None, 'val1', 'val2', 0.5, 0.8, 0.9],
}

dt_clf_tuned = GridSearchCV(DecisionTreeClassifier(), param_grid, scoring='accuracy')
dt_clf_tuned.fit(X_train, y_train)

# Sort scores
pd.DataFrame(dt_clf_tuned.cv_results_).sort_values('mean_test_score', ascending=False)
```

Model Saving and Loading

```
dt_clf_tuned.fit(X, y)
```

Fit the tuned model to the whole dataset

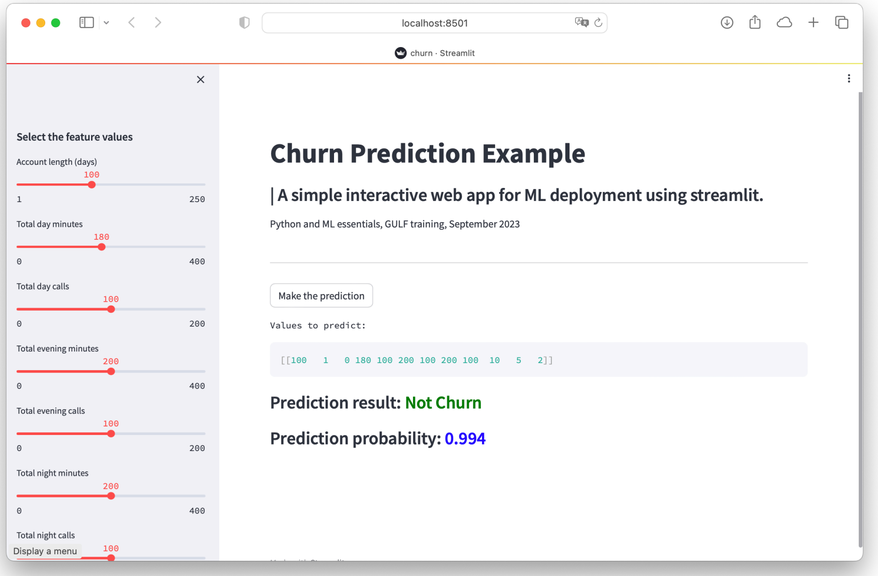
```
import pickle
import numpy as np

output = open('churn_model.pkl', 'wb')
pickle.dump(dt_clf_tuned, output)
output.close()
```

Save the tuned model

```
pkl_file = open('churn_model.pkl', 'rb')
churn_model = pickle.load(pkl_file)
```

Load the tuned model and use it to predict some value.

```
Xin = X_test.iloc[[20]]
churn_model.predict(Xin)[0]
churn_model.predict_proba(Xin).flatten()
```



Activity: Car Seat

Format

A data frame with 400 observations on the following 11 variables.

`Sales`
: Unit sales (in thousands) at each location

`CompPrice`
: Price charged by competitor at each location

`Income`
: Community income level (in thousands of dollars)

`Advertising`
: Local advertising budget for company at each location (in thousands of dollars)

`Population`
: Population size in region (in thousands)

`Price`
: Price company charges for car seats at each site

`ShelveLoc`
: A factor with levels `Bad`, `Good` and `Medium` indicating the quality of the shelving location for the car seats at each site

`Age`
: Average age of the local population

`Education`
: Education level at each location

`Urban`
: A factor with levels `No` and `Yes` to indicate whether the store is in an urban or rural location

`US`
: A factor with levels `No` and `Yes` to indicate whether the store is in the US or not

Supplementary

Model Selection from Nested K-fold CV

Nested K-fold CV gives the model with best test errors.

- Need to trade-off performance and complexity to choose model to deploy.

There are as many optimal hyperparameter sets as the number of outer loop iterations. Which one to use ?

Possible approaches:

- Apply K-fold CV on the whole data for each hyperparameter set and determine the best one.
- Create an ensemble model from models with different hyperparameter sets. The prediction is obtained by taking average, soft voting, or hard voting.

Model Selection in Practice

Sometimes the simplest hypothesis consistent with the data is less accurate for prediction than a more complicated one.

Simpler model may not be the most “accurate” one but it may be

- Computationally more efficient
- Easier to implement
- Easier to understand and reason compared to more complicated alternatives.

Non-optimized models are ok if they are within one standard error of the best performance model that is overly complex.

Challenges for ML Beginners

Turning business problems into ML problem

Framing the ML problem (input, output, model)

Having an intuition about what features should be relevant and available "before" the target.

Beware of data leakage

Improving on an already good model

When straightforward "default" model fitting yield terrible results,

- Tuning the model
- Trying new features
- Trying alternative models

Studying new models and actually understand them.

Data Leakage

got_pneumonia	age	weight	male	took_antibiotic_medicine	...
False	65	100	False	False	...
False	72	130	True	False	...
True	58	100	False	True	...

Target leakage - Feature contains information about the target.

- Ex: Antibiotic taken after the pneumonia has occurred.
- Model will be inaccurate because patients who will get pneumonia won't have received antibiotics yet when we need to make predictions about their future health.
- Any variable updated (or created) after the target value is realized should be excluded.
- Ex: Daily expense is a good indicator of transaction limit exceeding but it cannot be known in advance.

Train-test contamination

- Run preprocessing (like fitting an imputer for missing values, data scaling) before splitting the data.
- Information in test set leaks to the training, and hence too optimistic model performance.
- Although less significant, should avoid it by preprocessing train and test data separately.
- When the same model is deployed in real time, we tend to see a performance degradation because there is a lot of uncertainty and there is no prior information of the data, e.g., mean, std, which is as a result of data leakage during the testing phase.

☐ Do not apply SMOTE before cross-validation.

☐ Beware of train-test splitting images from the same person.

THE MACHINE LEARNING CANVAS

Designed for: | Designed by: | Date: | Iteration: .

PREDICTION TASK

Type of task? Entity on which predictions are made? Possible outcomes? Wait time before observation?

DECISIONS

How are predictions turned into proposed value for the end-user? Mention parameters of the process / application that does that.

VALUE PROPOSITION

Who is the end-user? What are their objectives? How will they benefit from the ML system? Mention workflow/interfaces.

DATA COLLECTION

Strategy for initial train set & continuous update. Mention collection rate, holdout on production entities, cost/constraints to observe outcomes.

DATA SOURCES

Where can we get (raw) information on entities and observed outcomes? Mention database tables, API methods, websites to scrape, etc.

IMPACT SIMULATION

Can models be deployed? Which test data to assess performance? Cost/gain values for (in)correct decisions? Fairness constraint?

MAKING PREDICTIONS

When do we make real-time / batch pred.? Time available for this + featurization + post-processing? Compute target?

BUILDING MODELS

How many prod models are needed? When would we update? Time available for this (including featurization and analysis)?

FEATURES

Input representations available at prediction time, extracted from raw data sources.

MONITORING

Metrics to quantify value creation and measure the ML system's impact in production (on end-users and business)?

OWNML.CO

Model Deployment Options