MCIT ML Training

Use Cases – Default of Credit Card Clients

Amgad Zaki, AWS Senior Solutions Architect

amgadz@amazon.com

Use Case

- Input: excel sheet with historical customers data
- Required: develop a model to tell if client will default the payment or not.

- Data is labeled for the sake of training and learning. However in real life test data might be different, so split your data for better learning.

- This is a simple binary classification problem, and simple imputations are required.

- Model can be KNN, Random Forests, XGBoost, Logistic Regressions, NN, etc.

Use Case

- Sample Notebook for similar problem is provided.

- You can run it on your local machine or Amazon SageMaker.

- Scikit Learn is used here, and it is good framework for feature engineering in general.

Use Case – Metadata info

X1: Amount of the given credit.

X2: Gender (male; female).

X3: Education (graduate school; university; high school; others).

X4: Marital status (married; single; others).

X5: Age (years).

X6 – X11: History of past payments. Tracked past monthly payment records (from April to September, 2005) are displayed as follows: X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005… X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months… 8 = payment delay for eight months; 9 = payment delay for nine months and above.

X12-X17: Amount of bill statement X12 = amount of bill statement in September, 2005; X13 = amount of bill statement in August, 2005… X17 = amount of bill statement in April, 2005.

X18-X23: Amount of previous payment. X18 = amount paid in September, 2005; X19 = amount paid in August, 2005…. X23 = amount paid in April, 2005.

Y: Did the person default? (Yes = 1, No = 0)

Use Case – Metadata info

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14
ID	LIMIT_BA	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AM	BILL_AM	BILL_AM
1	20000	female	university	married	24	2	2	-1	-1	-2	-2	3913	3102	689
2	120000	female	university	single	26	-1	2	0	0	0	2	2682	1725	2682
3	90000	female	university	single	34	0	0	0	0	0	0	29239	14027	13559
4	50000	female	university	married	37	0	0	0	0	0	0	46990	48233	49291
5	50000	male	university	married	57	-1	0	-1	0	0	0	8617	5670	35835
6	50000	male	graduate school	single	37	0	0	0	0	0	0	64400	57069	57608
7	500000	male	graduate school	single	29	0	0	0	0	0	0	367965	412023	445007
8	100000	female	university	single	23	0	-1	-1	0	0	-1	11876	380	601
9	140000	female	others	married	28	0	0	2	0	0	0	11285	14096	12108
10	20000	male	high school	single	35	-2	-2	-2	-2	-1	-1	0	0	0
11	200000	female	high school	single	34	0	0	2	0	0	-1	11073	9787	5535
12	260000	female	graduate school	single	51	-1	-1	-1	-1	-1	2	12261	21670	9966
13	630000	female	university	single	41	-1	0	-1	-1	-1	-1	12137	6500	6500
14	70000	male	university	single	30	1	2	2	0	0	2	65802	67369	65701
15	250000	male	graduate school	single	29	0	0	0	0	0	0	70887	67060	63561
16	50000	female	high school		23	1	2	0	0	0	0	50614	29173	28116
17	20000	male	graduate school	single	24	0	0	2	2	2	2	15376	18010	17428
18	320000	male	graduate school	married	49	0	0	0	-1	-1	-1	253286	246536	194663
19	360000	female	graduate school	married	49	1	-2	-2	-2	-2	-2	0	0	0
20	180000	female	graduate school	single	29	1	-2	-2	-2	-2	-2	0	0	0
21	130000	female	high school	single	39	0	0	0	0	0	-1	38358	27688	24489
22	120000	female	university	married	39	-1	-1	-1	-1	-1	-1	316	316	316
23	70000	female	university	single	26	2	0	0	2	2	2	41087	42445	45020
24	450000	female	graduate school	married	40	-2	-2	-2	-2	-2	-2	5512	19420	1473
25	90000	male	graduate school	single	23	0	0	0	-1	0	0	4744	7070	0

Hands-on Machine Learning

Python

- Everyone is using it in machine learning & data science

Jupyter notebooks

- So much easier to create and share documentation that contains both code and rich text elements, such as equations

(Amazon SageMaker or Anaconda.com)

Sklearn (scikit-learn)

- Great selection of ML algorithms and data processing methods

Apache MXNet Gluon

- Scalability & ease of use

Amazon SageMaer Free Tier https://aws.amazon.com/free

**** Please Review The Rules and Conditions as per the link*

- Two months free
  - Notebook usage
    - Monthly free tier of 250 hours of t2.medium or t3.medium
  - Training
    - 50 hours of m4.xlarge or m5.xlarge for training
  - Inferencing
    - 125 hours of m4.xlarge or m5.xlarge

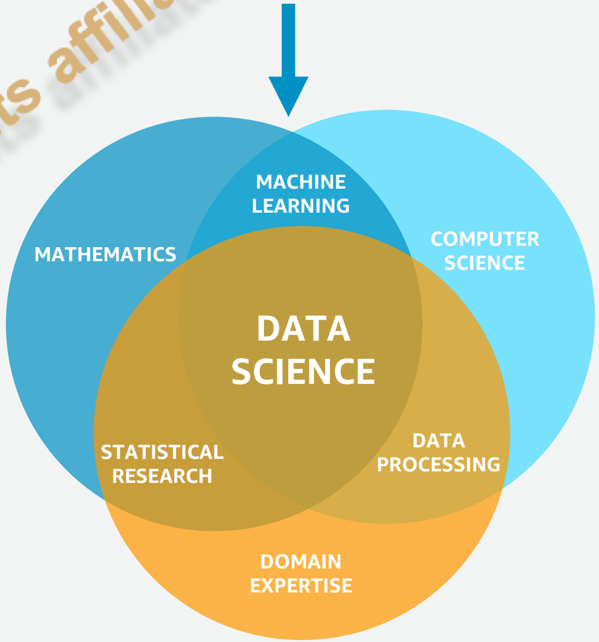
* The free tier does not cover the storage volume usage.

What is Data Science?

Wikipedia describes **Data Science** as:

"a multi-disciplinary field that uses scientific methods, processes, algorithms and systems to **extract knowledge and insights** from **structured** and **unstructured** data."

Machine Learning



https://en.wikipedia.org/wiki/Data_science

What is Machine Learning?

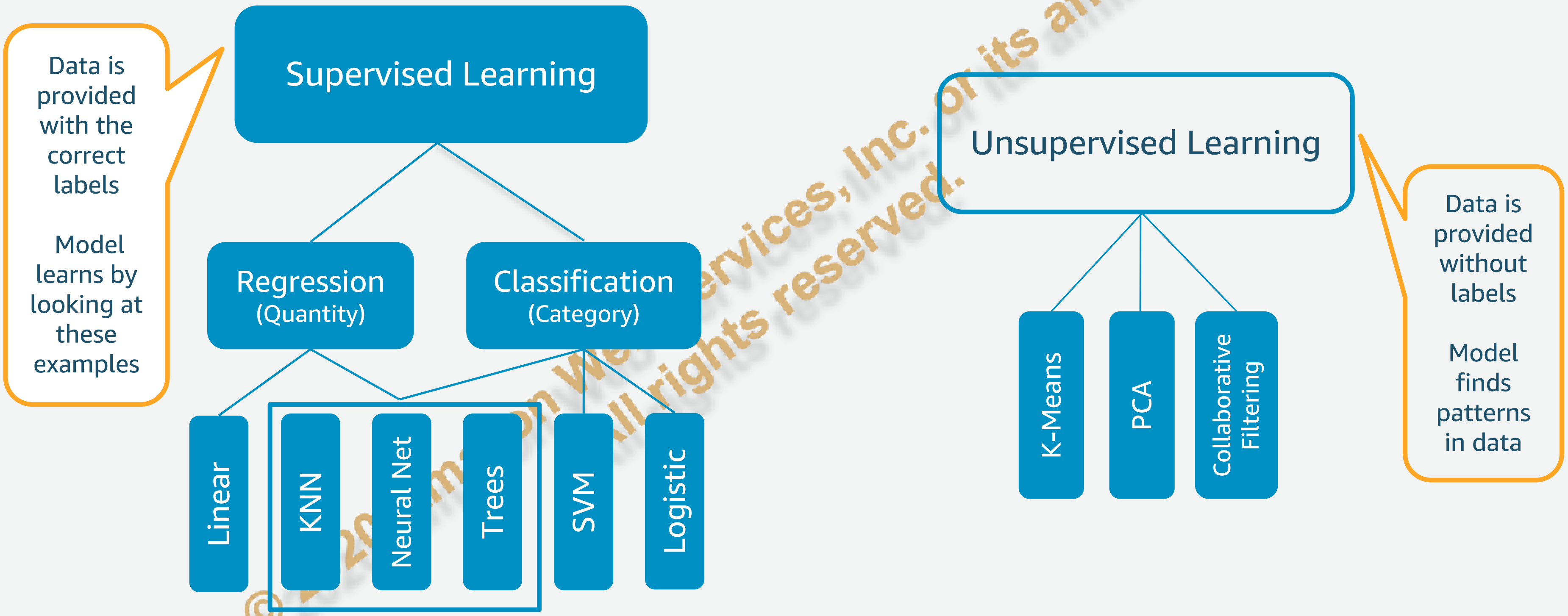
> “Programming computers to **learn from experience** should eventually **eliminate the need for much of this detailed programming effort.**”
>
> *Arthur Samuel (1959)*
> Pioneer of AI research

Problem ➤ Rules ➤ Classical Programming (If/else, etc.) ➤ Answers

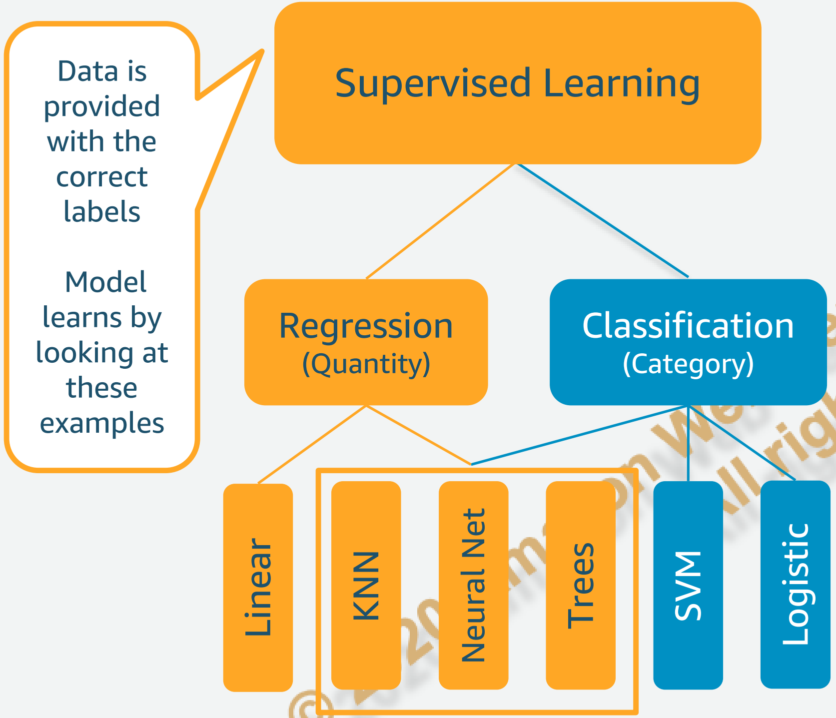
Problem ➤ Answers ➤ ML Algorithms ➤ Trained ML Models (Rules) ➤ Answers

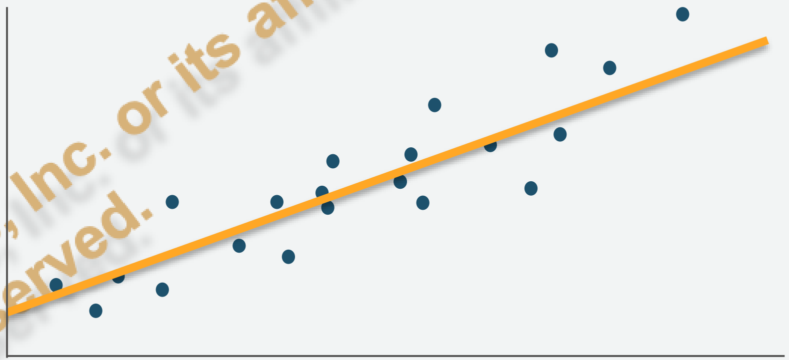
New Similar Problem

Supervised vs. Unsupervised Learning



Supervised Learning: Regression

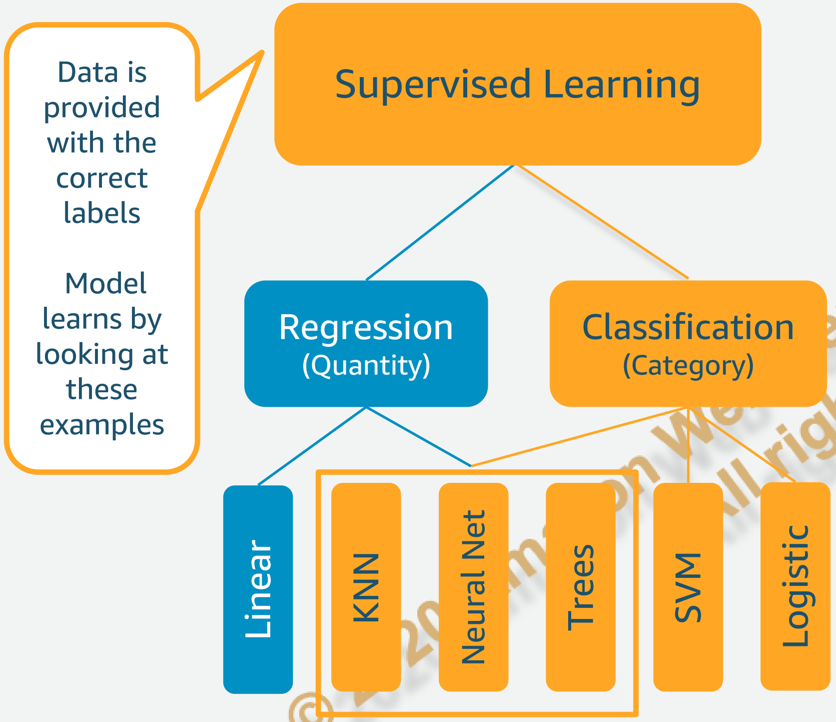




Label | Features

Price	Bedrooms	SqFootage	Age
280.000	3	3292	14
210.030	2	2465	6

Supervised Learning: Classification

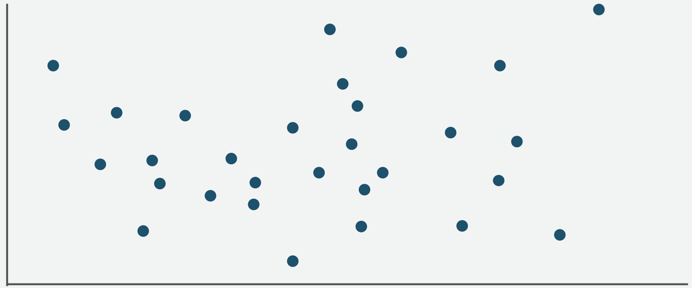


Label | Features

Star	Points	Edges	Size
1	5	10<	750
0	0	>9	150

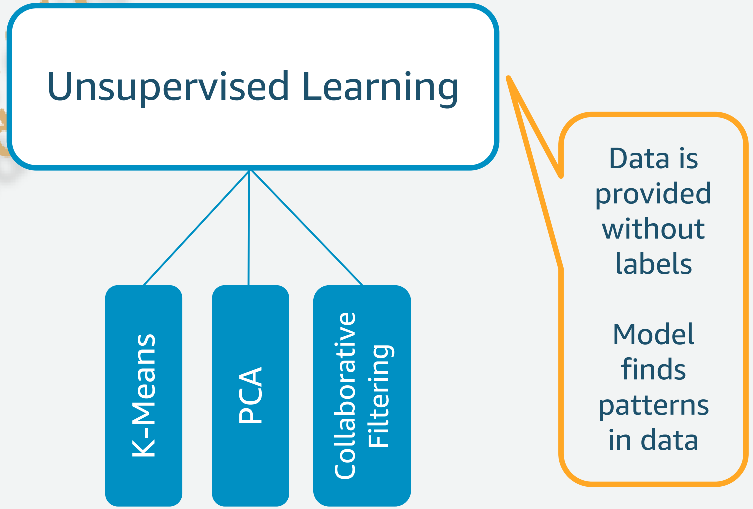
© 2020 Amazon Web Services, Inc. or its affiliates. All rights reserved.

Unsupervised Learning: Clustering

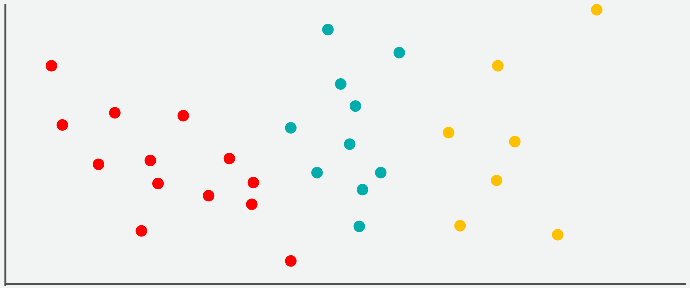


Features

Age	Music	Books
21	Jazz	Practical Magic
47	Classical	The Great Gatsby

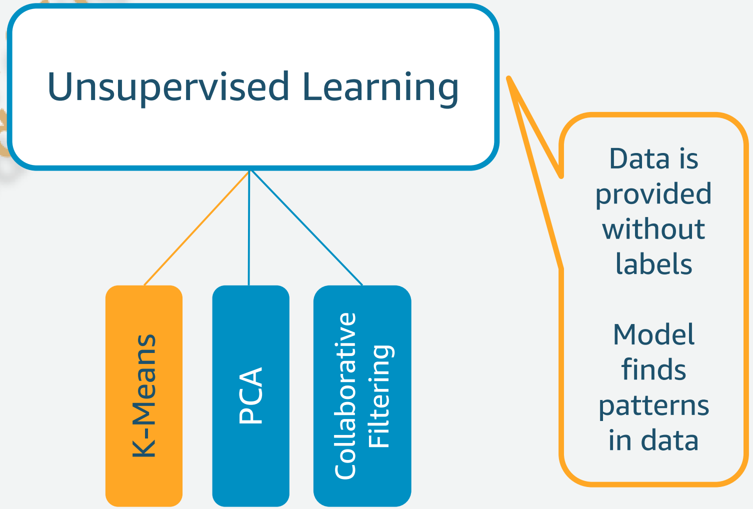


Unsupervised Learning: Clustering

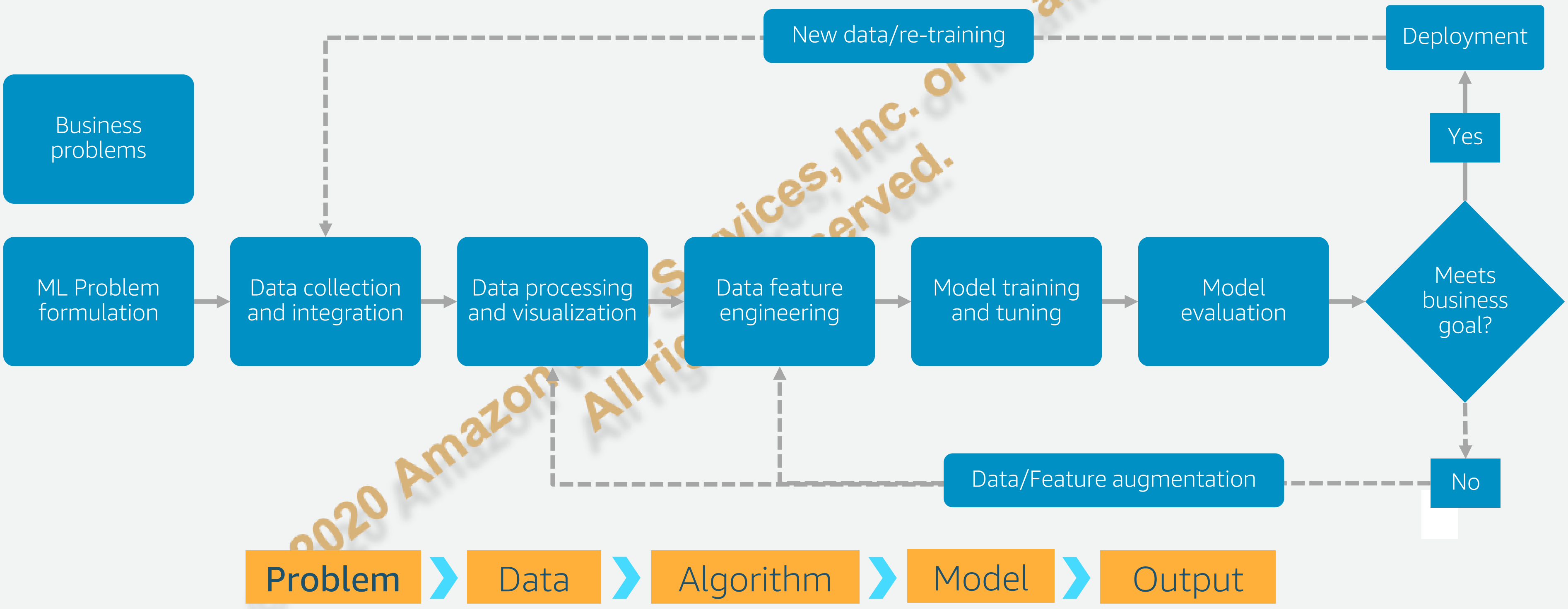


Features

Age	Music	Books
21	Jazz	Practical Magic
47	Classical	The Great Gatsby



Machine Learning Pipeline



Machine Learning Jargons

ML optimizes on **predictive** performance, while statistics places importance on **interpretability** and parsimony/simplicity.

ML	Statistics	Simply Put
Label/Target/"y"	Dependent/Response/Output Variable	The thing you're trying to predict.
Feature/"x"	Independent/Explanatory/Input Variable	Data that help you make predictions.
Feature Engineering	Transformation	Reshaping data to get more value.
1d, 2d,… nd	Dimensionality	Number of feature
Model Weights	Parameters	A set of numbers embedded in a model that can predict the labels.

Tools and Libraries

Machine Learning Tools and Libraries

- **Notebooks:** Sagemaker
  - 100+ Examples: https://github.com/awslabs/amazon-sagemaker-examples
- **Libraries:**
  - NumPy: http://www.numpy.org/
  - pandas: http://pandas.pydata.org/
  - scikit-learn: http://scikit-learn.org
  - Gluon: https://gluon.mxnet.io/

Problem and Data: Food Delivery

John loves to order his food online for home and work.

He wants to predict whether his order will be on time or late beforehand.

He logged his previous 45 orders like this:

IsBadWeather?	IsRushHour?	MilesDistanceFromRestaurant	IsUrbanAddress?	Late
0	1	5	1	0
1	0	7	0	1
0	1	2	1	0
1	1	4.2	1	1
0	0	7.8	0	0
...	...	...	...	...

Two classes: 0/on time, and 1/late

KNN Model

ML Model: K Nearest Neighbors (KNN)

Two classes: ▲ (on time/0) and ● (late/1)

KNN Algorithm classifies a record by comparing it to its nearest neighbors.

K is the number of nearest neighbors we will consider to classify a new record ?.

In this context:

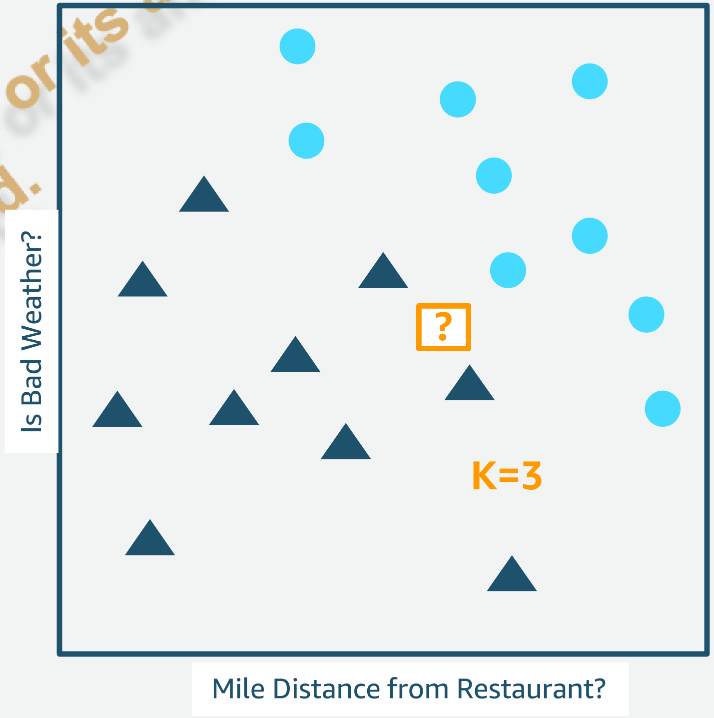
Physically closer ≈ Similar Record (same class)

ML Model: K Nearest Neighbors (KNN)

Two classes: ▲(on time/0) and ●(late/1)

KNN Algorithm classifies a record by comparing it to its nearest neighbors.

K is the number of nearest neighbors we will consider to classify a new record ? .

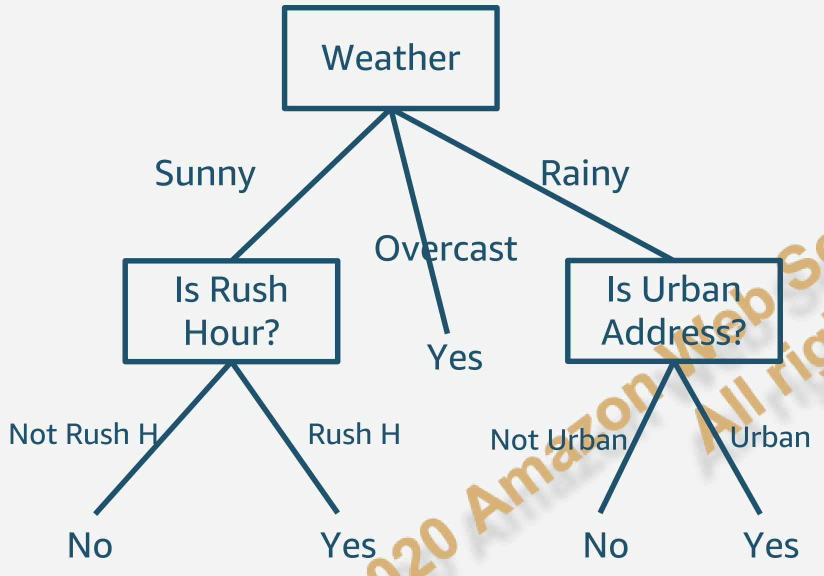


In this context:

Physically closer ≈ Similar Record (same class)

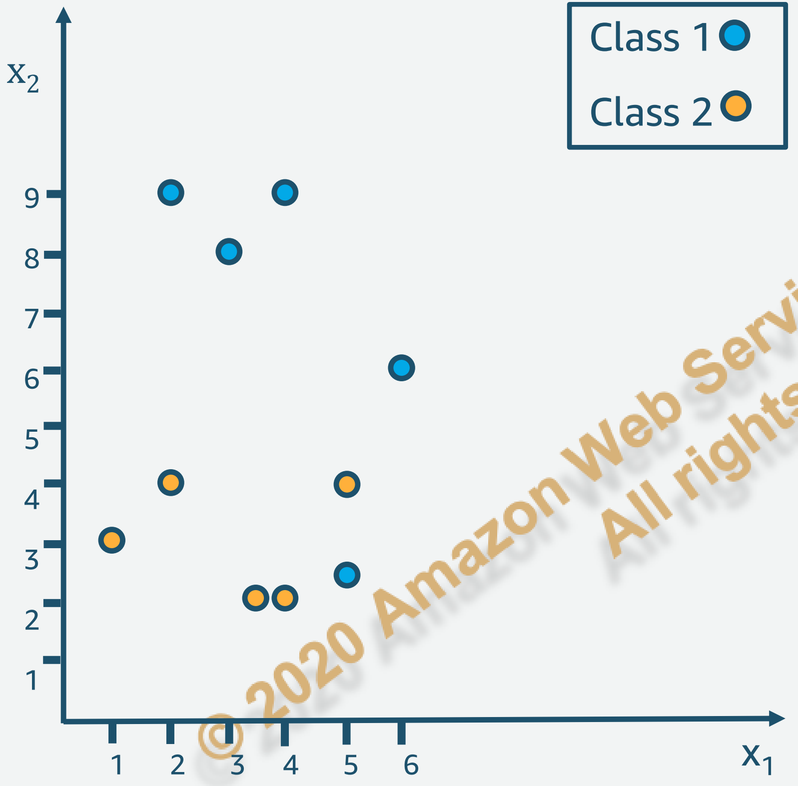
Decision Trees Models

ML Model: Decision Tree



IsBadWeather?	IsRushHour?	MilesDistanceFromRestaurant	IsUrbanAddress?	Late
0	1	5	1	0
1	0	7	0	1
0	1	2	1	0
1	1	4.2	1	1
0	0	7.8	0	0
…	…	…	…	…

Decision Trees: Numerical Example



x_1	x_2	y
3.5	2	1
5	2.5	2
1	3	1
2	4	1
4	2	1
6	6	2
2	9	2
4	9	2
5	4	1
3	8	2

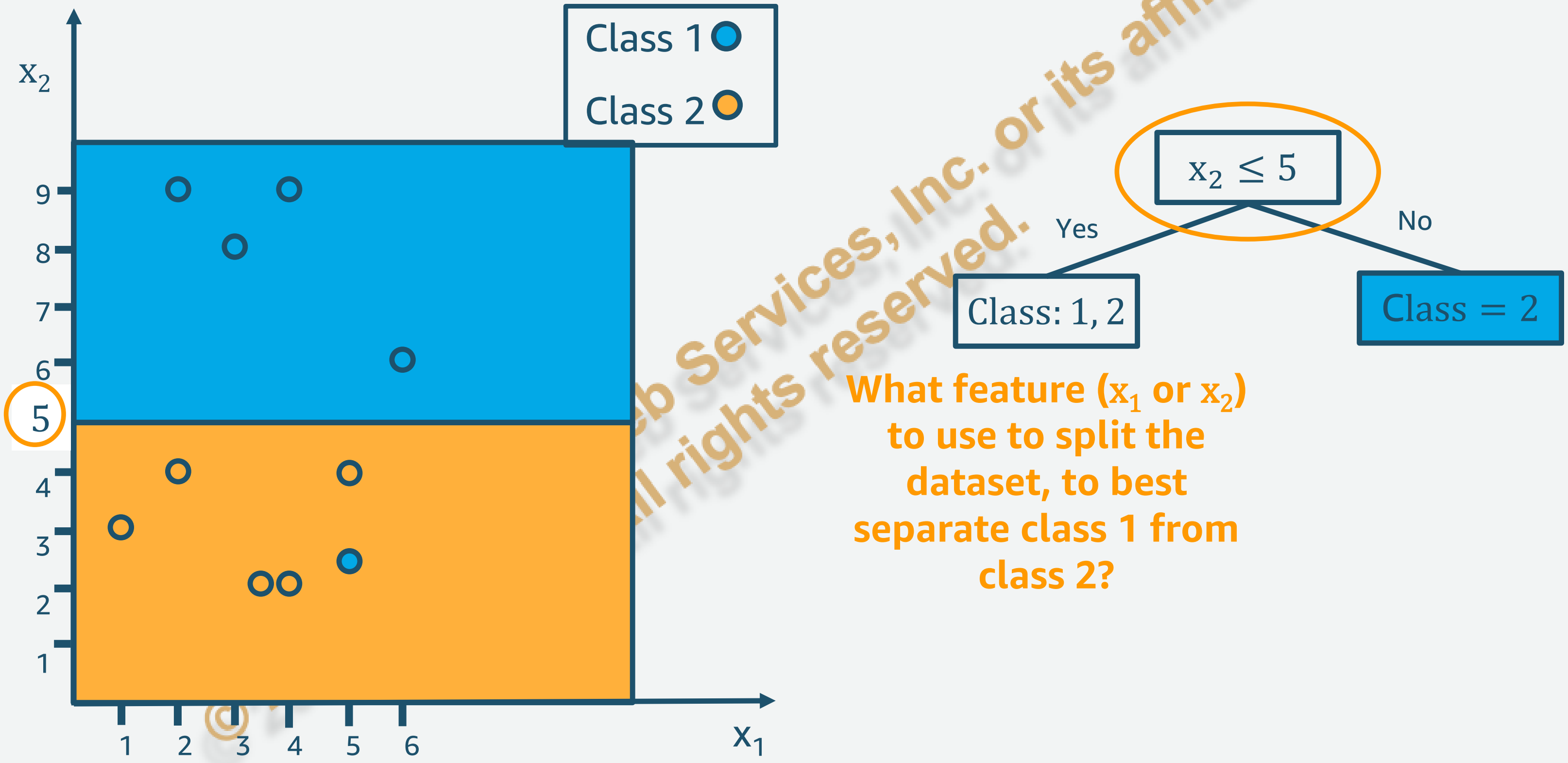
Decision Trees: Numerical Example

Class: 1,2

What feature (x_1 or x_2) to use to split the dataset, to best separate class 1 from class 2?

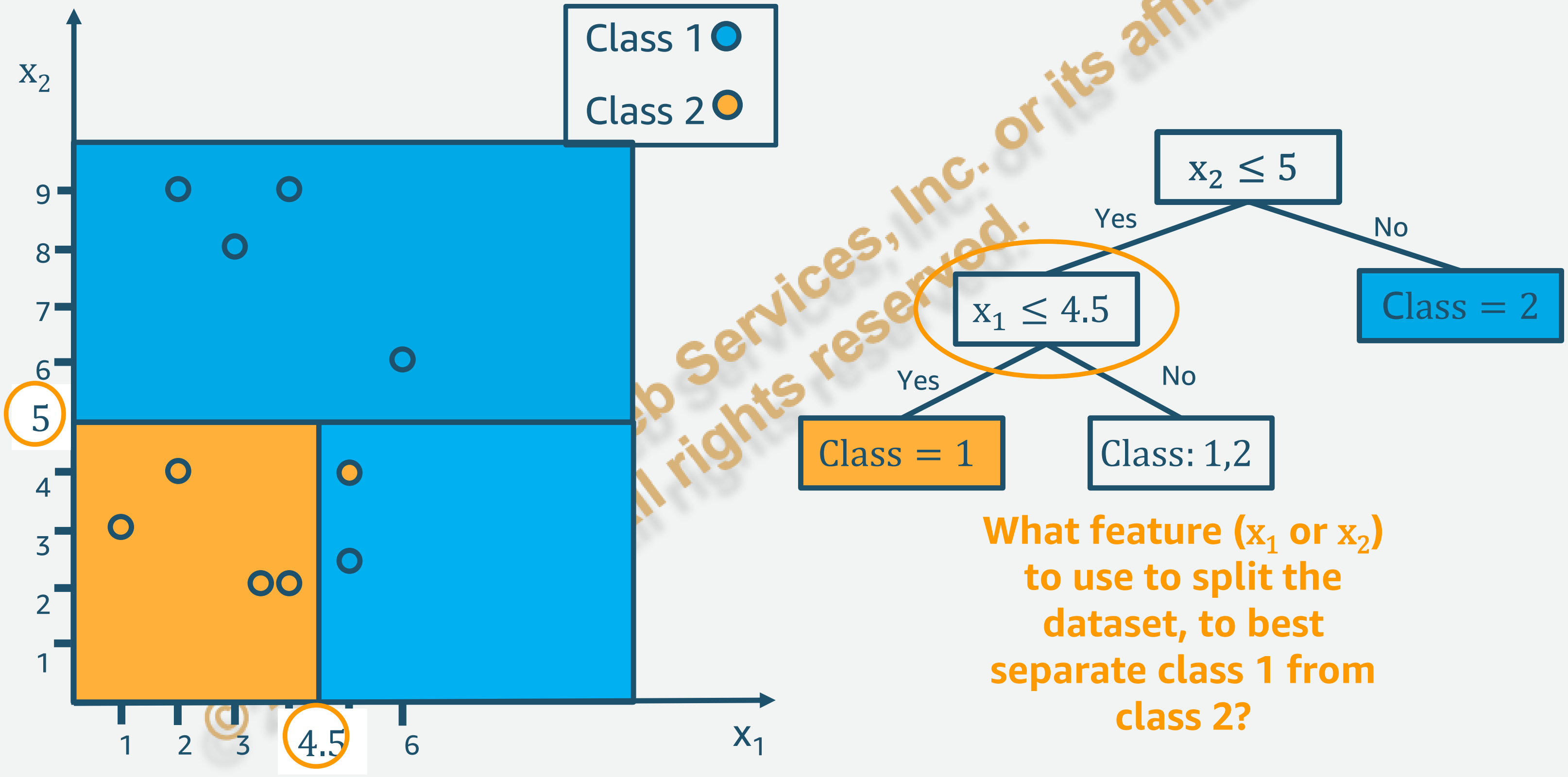
[select the splits such that the descendent subsets are "purer" than their parents]

Decision Trees: Numerical Example



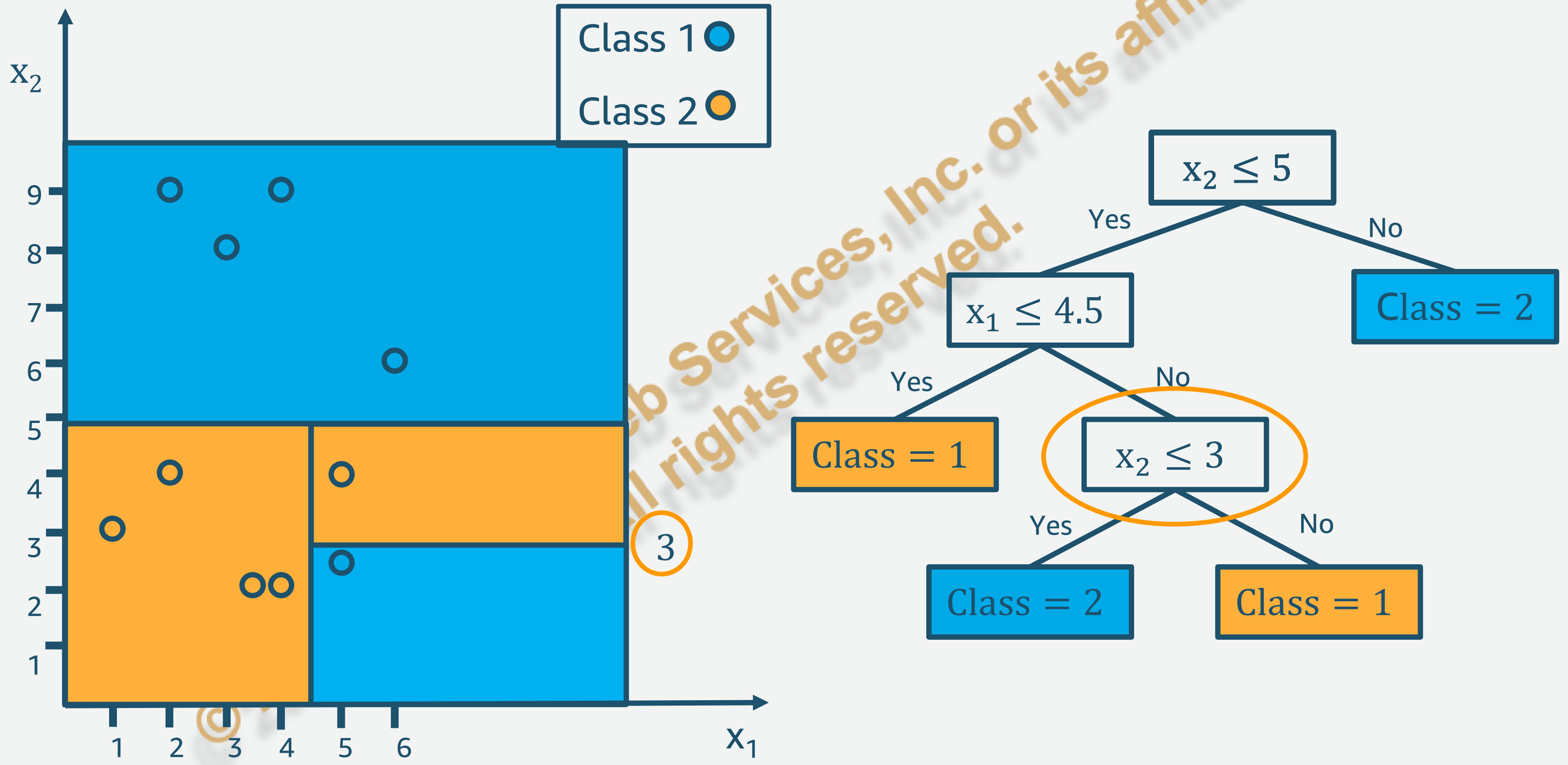
What feature (x_1 or x_2) to use to split the dataset, to best separate class 1 from class 2?

Decision Trees: Numerical Example



What feature (x_1 or x_2) to use to split the dataset, to best separate class 1 from class 2?

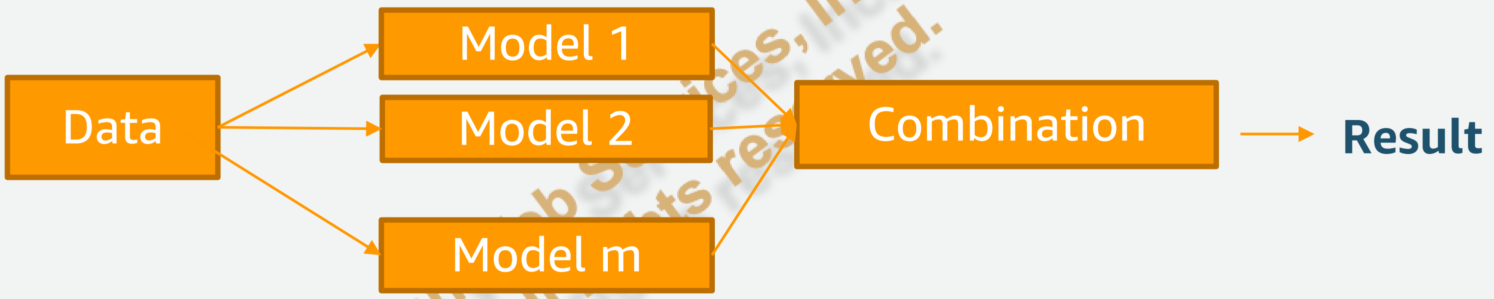
Decision Trees: Numerical Example



Ensemble Models

Ensemble Learning

Ensemble models create a **strong model** from **multiple weak models**.



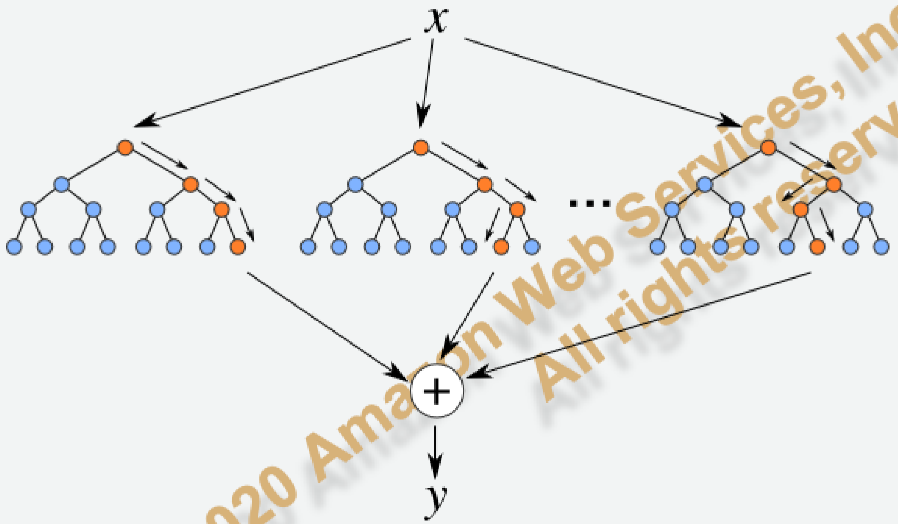
Bootstrap Aggregating (Bagging)

Bagging (Bootstrap Aggregating) methods

- Build several independent estimators on randomly drawn samples from the training set (with replacement) **- bootstrap technique**
  - For example, given dataset: [1, 2, 4, 5, 7, 9], potential samples are: [1, 1, 2, 4, 9, 9]; [2, 4, 5, 5, 7, 7]; [1, 1, 1, 1, 1, 1]; [1, 2, 4, 5, 7, 9]
- Majority vote or average the predictions from all estimators
- On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced
- Bagging Trees e.g., **Random Forest**

Bagging trees: Random Forests

Combination of multiple trees.

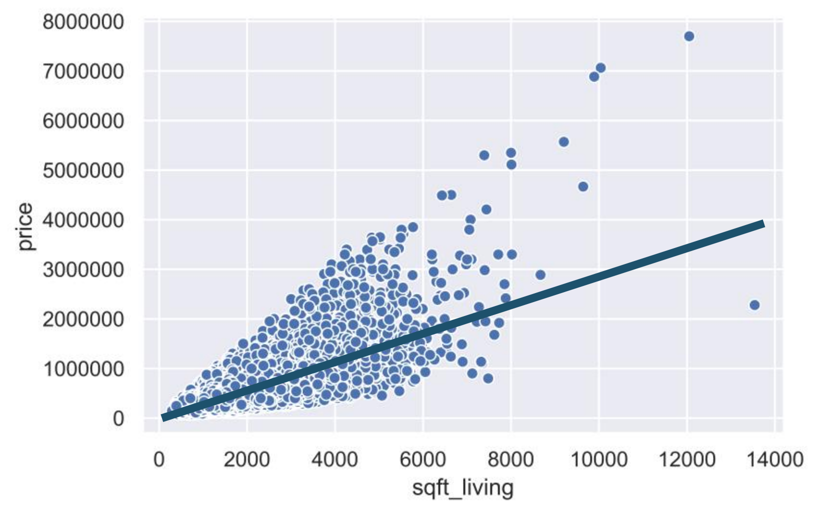


Train a decision tree for each sampled data.

Combine end results of each tree by voting.

Regression Models

Linear Regression



* Data source: King County, WA Housing Info.

We use regression for numerical value prediction.

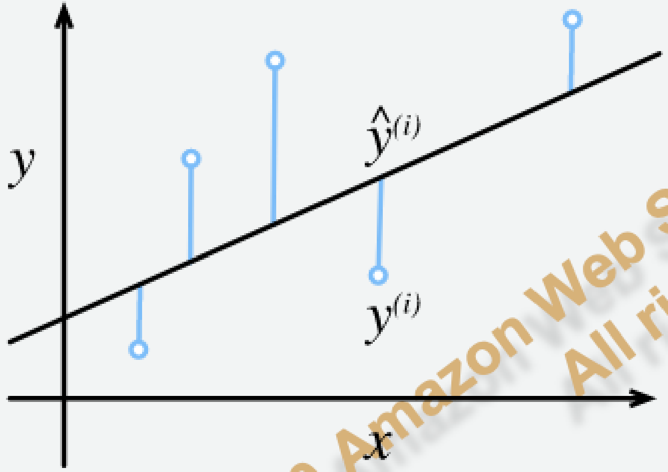
Example: How does the **price of a house** (target variable, y) change relate to its **square footage living** (explanatory variable, x)?

$$price = w_0 + w_1 * sqft\_living$$

For $sqft\_living = 6000$,

$$price = w_0 + w_1 * 6000$$

Linear Regression



Regression line $y = w_0 + w_1 x$ is defined by: w_0 (intercept), w_1 (slope).

The **vertical offset** for each data point from the line is the **error** between y (the true label) and $\hat{y}$ (the prediction based on x).

Best "line" (best w_0, w_1) minimizes the sum of squared errors (SSE):

$$\sum (y^{(i)} - \hat{y}^{(i)})^2$$

Linear Regression

Multiple linear regression includes q features with $q \geq 2$

$$y = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_q x_q$$

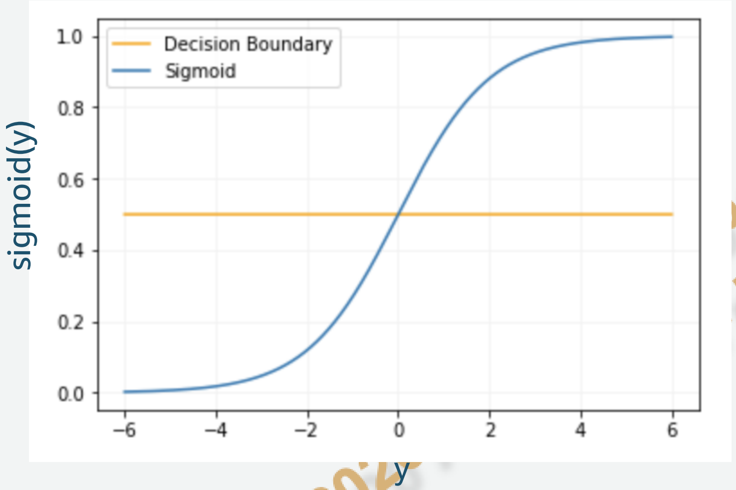
Example: How does the **price of a house** (target variable y) change relate to its **square footage living** (explanatory variable x_1), its **number of bedrooms** (explanatory variable x_2), its **zip code** (x_3),…?

$$price = w_0 + w_1 * sqft\_living + w_2 * bedrooms + w_3 * zip\_code + \ldots$$

- Sensitive to correlation between features.
- Features are allowed to have interactions and higher order terms.

Logistic Regression

Idea: We can apply the **Sigmoid function** to $y = w_0 + w_1 x_1 + \ldots + w_q x_q$



- The **Sigmoid (Logistic) function**

$$sigmoid(y) = \frac{1}{1 + e^{-y}}$$

"squishes" y values to the 0–1 range.

- Can define a "**Decision boundary**" at 0.5
  - if $sigmoid(y) < 0.5$, round down (class 0)
  - if $sigmoid(y) \geq 0.5$, round up (class 1)
- Our regression equation becomes:

$$sigmoid(w_0 + w_1 x_1 + \ldots + w_q x_q)$$

Model Evaluation

Evaluating Classification Models

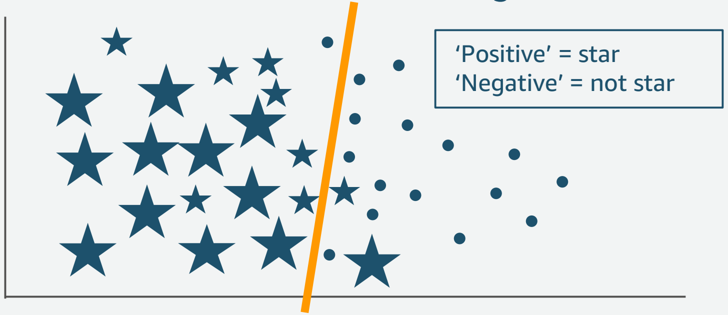
		Prediction	
		Positive	Negative
True State	Positive	True Positive 18	False Negative 2
	Negative	False Positive 1	True Negative 15

True Positive: Predicted 'Positive' when the actual is 'Positive'

False Positive: Predicted 'Positive' when the actual is 'Negative'

False Negative: Predicted 'Negative' when the actual is 'Positive'

True Negative: Predicted 'Negative' when the actual is 'Negative'



Classification: Accuracy

True State \ Prediction	Positive	Negative
Positive	True Positive 18	False Negative 2
Negative	False Positive 1	True Negative 15

Accuracy*: The percent (ratio) of cases classified correctly:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$Accuracy = \frac{18 + 15}{18 + 1 + 2 + 15} = 0.92$$

* (bad) $0 \leq Accuracy \leq 1$ (good)

Classification: Accuracy

		Prediction	
		Positive	Negative
True State	Positive	True Positive 2	False Negative 8
	Negative	False Positive 2	True Negative 88

High Accuracy Paradox:
Accuracy is misleading when dealing with imbalanced datasets: few true state positives (the 'rare' class), many true state negatives (the 'dominant' class), **high accuracy even when few True Positives**!

$$Accuracy = \frac{2 + 88}{2 + 2 + 8 + 88} = 0.90$$

Classification: Precision

		Prediction: Positive	Prediction: Negative
True State	Positive	True Positive 2	False Negative 8
	Negative	False Positive 2	True Negative 88

Precision*: Accuracy of a predicted positive outcome:

$$Precision = \frac{TP}{TP + FP}$$

$$Precision = \frac{2}{2+2} = 0.50$$

*(bad) $0 \leq Precision \leq 1$ (good)

Classification: Recall

		Prediction	
		Positive	Negative
True State	Positive	True Positive 2	False Negative 8
	Negative	False Positive 2	True Negative 88

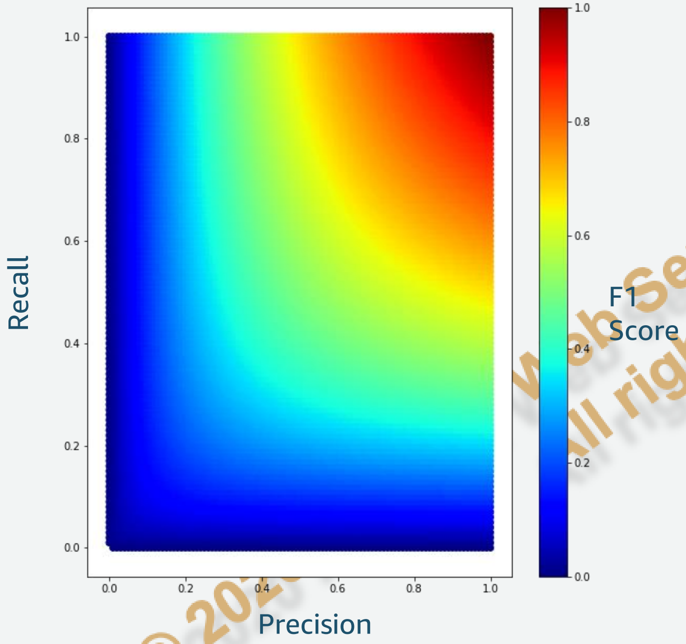
Recall*: : Measures model's ability to predict a positive outcome:

$$Recall = \frac{TP}{TP + FN}$$

$$Recall = \frac{2}{2+8} = 0.20$$

*(bad)$0 \leq Recall \leq 1$ (good)

Classification: F1 Score



F1 Score*: It is a combined metric, the harmonic mean of precision and recall.

$$F1\,Score = \frac{2 * (Precision * Recall)}{Precision + Recall}$$

$*$(bad) $0 \leq F1\,Score \leq 1$ (good)

Evaluating Regression Models

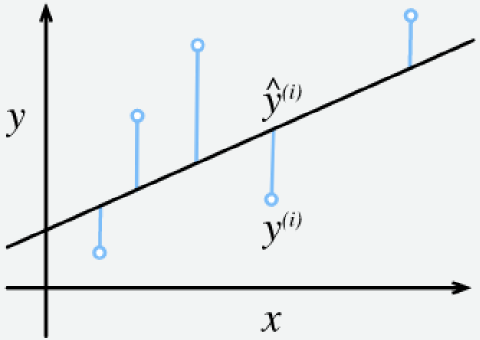
Metrics	Equations
Mean Squared Error (MSE)	$MSE = \frac{1}{n}\sum_{i=0}^{n}(y^{(i)} - \hat{y}^{(i)})^2$
Root Mean Squared Error (RMSE)	$RMS = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(y^{(i)} - \hat{y}^{(i)})^2}$
Absolute Mean Error (AME)	$AME = \frac{1}{n}\sum_{i=0}^{n}\lvert y^{(i)} - \hat{y}^{(i)}\rvert$
R Squared (R^2)	$R^2 = 1 - \frac{\sum_{i=0}^{n}(y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=0}^{n}(y^{(i)} - \bar{y})^2}$

$y^{(i)}$: Data values

$\hat{y}^{(i)}$: Predicted values

$\bar{y}$: Mean value of data values, $\bar{y} = \frac{1}{n}\sum_{i=0}^{n} y^{(i)}$

n : Number of data records



Model Evaluation: Overfitting

Overfitting: Model memorizes or imitates training data and doesn't generalize well on new "unseen" data (test data).

- Model is **too complex,** for simple problems can cause overfitting.
- Model picks up the noise instead of the underlying relationship.
- Model will **perform well on training, but poorly on test**.

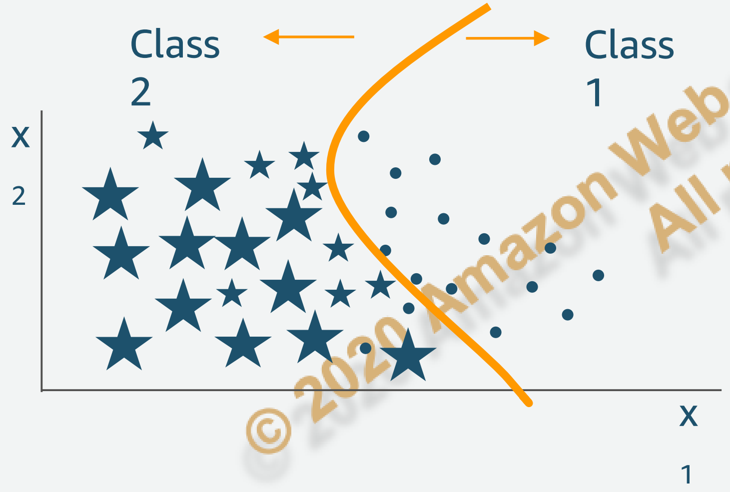
Model Evaluation: Underfitting

Underfitting: Model is not good enough to describe the relationship between the input data (x_1, x_2) and output y: {Class 1, Class 2}.

- Model is **too simple** to capture important patterns of training set.
- Model will **perform poorly on training and test.**

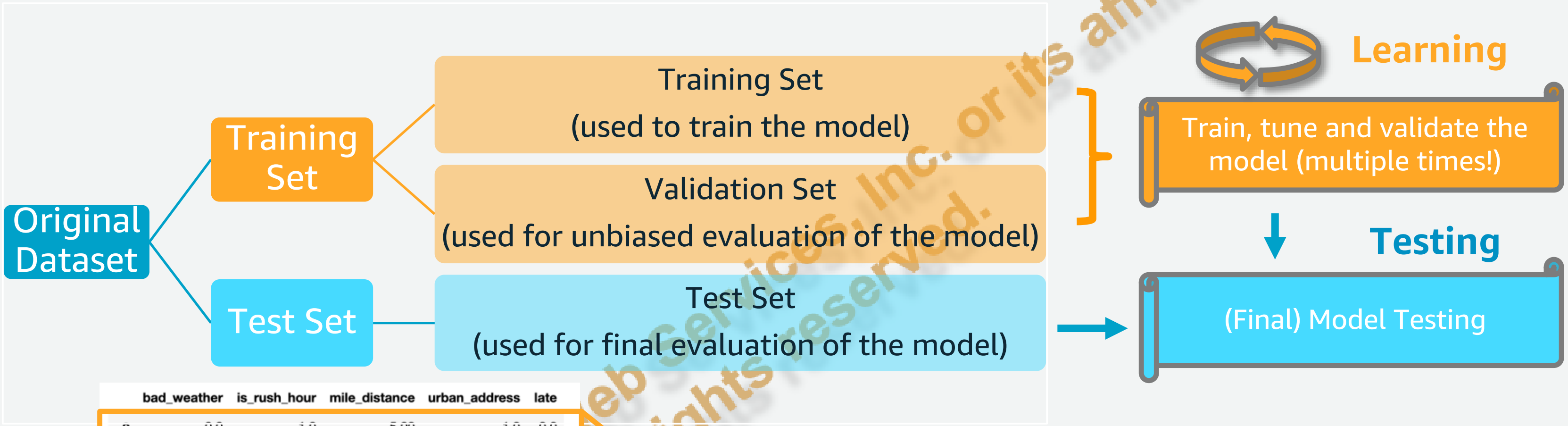
Model Evaluation: Appropriate Fitting

Appropriate fitting: Model captures the general relationship between the input data (x_1, x_2) and output y: {Class 1, Class 2}.



- Model not too simple, not too complex.
- Model picks up the underlying relationship rather than the noise in the training.
- Model will **perform good enough on training and test**.

Training - Validation - Test Datasets



	bad_weather	is_rush_hour	mile_distance	urban_address	late
0	0.0	1.0	5.00	1.0	0.0
1	1.0	0.0	7.00	0.0	1.0
2	0.0	1.0	2.00	1.0	0.0
3	1.0	1.0	4.20	1.0	0.0
4	0.0	0.0	7.80	0.0	1.0
5	1.0	0.0	3.90	1.0	0.0
6	0.0	1.0	4.00	1.0	0.0
7	1.0	1.0	2.00	0.0	0.0
8	0.0	0.0	3.50	0.0	1.0
9	1.0	0.0	2.60	1.0	0.0
10	0.0	0.0	4.10	0.0	1.0

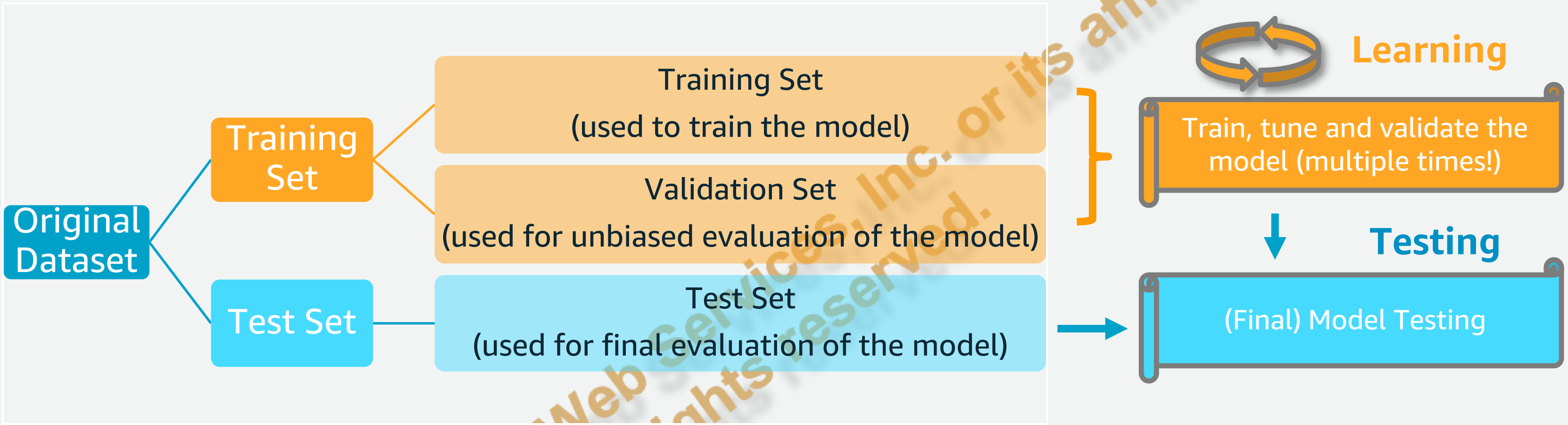
Training Set

Validation Set

Test Set

It is good practice to shuffle the dataset before the split to avoid bias in the resulting sets.

Training - Validation - Test Datasets



- Why do we divide the data into these three sub-datasets?
- How do we make sure our model generalizes well?

- The test set is not available to the model for learning, it is only used to ensure that the model generalizes well on new “unseen” data.

Exploratory Data Analysis

Exploratory Data Analysis

Exploratory Data Analysis **(EDA)** is an approach to analyze a dataset and capture main characteristics of it.

We usually use visual methods such as plots and histograms of data points.

Descriptive Statistics

Overall statistics df.head(), df.shape, df.info()

- Number of instances (i.e. number of rows)
- Number of features (i.e. number of columns)

Univariate statistics (single feature)

- Statistics for numerical features (mean, variance, histograms) -- df.describe(), hist(df[feature])
- Statistics for categorical features (histograms, mode, most/least frequent values, percentage, number of unique values)
  - Histogram of values -- df[feature].value_counts() or seaborn's distplot()
- Target statistics
  - Class distribution -- df[target].value_counts() or np.bincount(y)

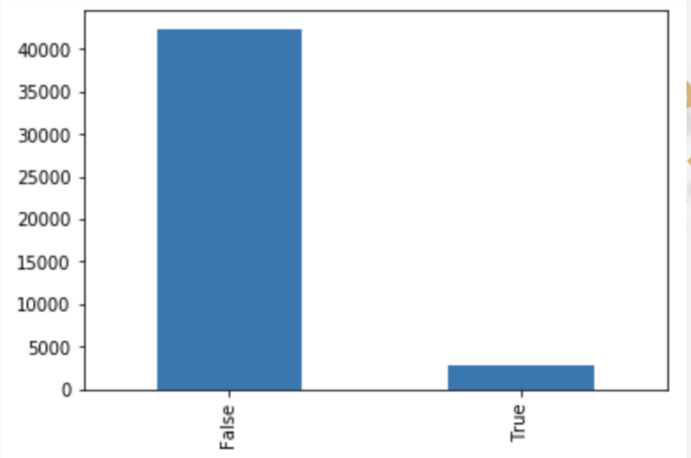
Multivariate statistics (more than one feature)

- Correlations -- df.plot.scatter(feature1, feature2), df[[feature1, feature2]].corr()

Univariate Statistics: Histograms

```
import matplotlib.pyplot as plt

df['SOME_FEATURE'].value_counts().plot.bar()
plt.show()
```



Numerical features:

- `.hist(df[feature]), .boxplot(df[feature])`

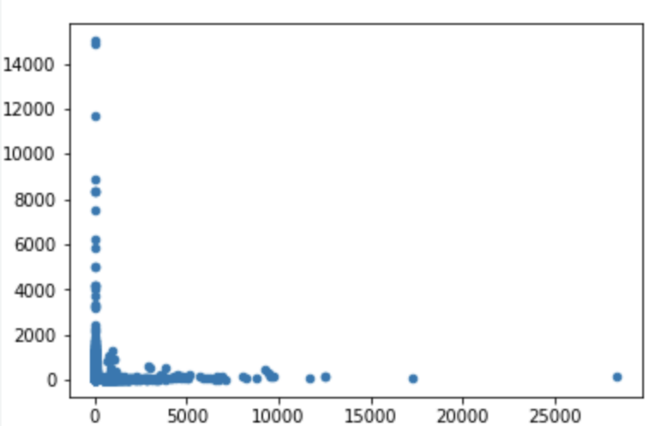
Categorical features:

- `df[feature].value_counts().plot.bar()`

Correlations: Scatterplot

Correlations: How strongly pairs of features are related.

```
df.plot.scatter(x='SOME_FEATURE ',
y='Some_TARGET')
plt.show()
```



Scatterplot matrices visualize attribute-target and attribute-attribute pairwise relationships.

Correlation matrices measure the linear dependence between features; can be visualized with heat-maps.

Correlations: Correlation Matrix

Correlation matrix: usually easier to read than scatterplots.

Correlation **values** are between -1 and 1:

- +1 means perfect positive correlation, while -1 shows perfect negative correlation.
- 0 means there is no relationship between the two variables.

```
cols = ['SOME_FEATURE 1', ' SOME_FEATURE2']
df[cols].corr().style.background_gradient(cmap='tab20c')
```

_WEIGHT	1	0.0128493
ST_PRICE	0.0128493	1

Correlations: Correlation Matrix

- **Highly correlated** (positive or negative) features usually degrade performance of linear ML models such as linear and logistic regression models - we should select one of the correlated features and discard the other(s).
- Decision Trees are immune to this problem.

```
cols = ['SOME_FEATURE1', 'SOME_FEATURE2']
df[cols].corr().style.background_gradient(cmap='tab20c')
```

WEIGHT	1	0.0128493
T_PRICE	0.0128493	1

Handling Missing Values

How to handle the missing values?

SEX	EDUCATION	MARRIAGE	PAY_0	PAY_2	BILL_AMT1	BILL_AMT2	BILL_AMT3
female	university	married	0	0	20344	21705	22537
female	university	?	1	2	23132	22474	28067
female	?	single	0	0	221590	227397	230302
female	graduate school	married	?	?	19161	20544	22704
male	?	married	-2	-2	?	?	?
male	graduate school		-1	-1	396	396	396
female	university	?	-1	-1	3959	?	285138
female	university	single	?	?	42238	38741	36696
female	university	married	?	?	21507	13207	13997

Handling Missing Values

Drop rows and/or columns with missing values: Remove those rows and/or columns from the dataset.

- Less training data samples and/or less features can lead to overfitting/underfitting

Imputation: Fill-in the missing values

- **Average imputation:** Replace missing values with the average value in the column. Useful for numeric variables – `df['col'].fillna((df['col'].mean()))`
- **Common point imputation:** Use the most common value for that column to replace missing values. Useful for categorical variables `df['col'].fillna((df['col'].mode()))`
- **Placeholder:** Assign a common value for missing data location
- **Advanced imputation:** Learn to predict missing values from complete samples using some machine learning techniques; for example: **AWS Datawig** tool uses neural networks to predict missing values in tabular data. https://github.com/awslabs/datawig

Feature Engineering

Feature Engineering

Feature engineering: Use domain and data knowledge to create novel features as inputs for ML models. Often more art than science.

- Use intuition: "What information would **a human** use to predict?"
- Generate many features, then apply dimensionality reduction if needed.
- Consider transformations of features and/or labels (e.g., squaring).
- Consider combinations of features (e.g., multiplication).
- Do not overthink or include too much manual logic.

sklearn.feature_extraction

Encoding Categorical Features

Categorical (also called **discrete**): These features don't have a natural numerical representation.

- Example: color ∈ {green, red, blue}, isFraud ∈ {false, true}
- Most Machine Learning models require converting categorical features to numerical ones.

Encode/define a mapping: Assign a number to each category.

- **Ordinal:** Categories are ordered, e.g., size ∈ {L > M > S}. We can assign L->3, M->2, S->1.
- **Nominal:** Categories are unordered, e.g., color. We can assign the numbers randomly.

Encoding Categorical Features

LabelEncoder: sklearn encoder, encodes target labels with value between 0 and n_classes-1 -- .fit(), .transform()

- Encodes target labels values, y (or **one feature only!**), and not the input X.
- Can be used to transform non-numerical labels or numerical labels.

	color	size	price	classlabel
0	green	S	10.1	shirt
1	red	M	13.5	pants
2	blue	L	15.3	shirt

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['color'] = le.fit_transform(df['color'])
print(df)
```

```
   color size  price classlabel
0      1    S   10.1      shirt
1      2    M   13.5      pants
2      0    L   15.3      shirt
```

Let's encode **one feature**, e.g. the **color** field.

Encoding Categorical Features

OrdinalEncoder: sklearn encoder, encodes categorical features as an integer array .fit(), .transform()

- Encodes (**two or more**) categorical features (**doesn't work on one feature only**!)
- Returns a single column of integers (0 to n_categories - 1) per feature.

	color	size	price	classlabel
0	green	S	10.1	shirt
1	red	M	13.5	pants
2	blue	L	15.3	shirt

Let's encode all **categorical** fields.

```
from sklearn.preprocessing import OrdinalEncoder
oe = OrdinalEncoder()
df[['color','size','classlabel']] =
        oe.fit_transform(df[['color','size','classlabel']])
print(df)
```

	color	size	price	classlabel
0	1.0	2.0	10.1	1.0
1	2.0	1.0	13.5	0.0
2	0.0	0.0	15.3	1.0

Encoding Categorical Features

Problem: Encoding categorical features with integers is wrong because the ordering and size of the integers is meaningless.

One-hot-encoding: Explode the categorical features into many binary features (as many categories per feature).

OneHotEncoder: sklearn one-hot encoder, encodes categorical features as a one-hot numeric array .fit(), .transform()

- Does not automatically name the new binary features.
- Works on **two or more features** (for one-hot encoding of one feature alone should use LabelBinarizer instead!)

get_dummies: pandas one-hot encoder

Encoding Categorical Features

get_dummies: **pandas** one-hot encoder, converts categorical features into new "dummy"/indicator features.

- Automatically names the new binary features.

```
pd.get_dummies(df, columns=['color'])
```

	color	size	price	classlabel
0	green	S	10.1	shirt
1	red	M	13.5	pants
2	blue	L	15.3	shirt

	size	price	classlabel	color_blue	color_green	color_red
0	S	10.1	shirt	0	1	0
1	M	13.5	pants	0	0	1
2	L	15.3	shirt	1	0	0

Encoding with many categories

Define a hierarchy structure:

Example: For a **zip code** feature, can try to use regions -> states -> city as the hierarchy,

and can choose a specific level to encode the zip code feature.

Group/bin the categories into **fewer groups** by similarity:

- Example: For some user demographics dataset, create age groups: 1-15, 16-22, 23-30, and so forth.

Model Development

K Nearest Neighbors (KNN)

K Nearest Neighbors (KNN) model predicts new data points based on the similar other records in a dataset.

Algorithm:

- Find "K" similar records
- For **classification** (predict class):
  - Take the majority class of those K records
- For **regression** (predict numerical value):
  - Take the average value of those K records

Assumptions:

- Similarity can be measured by the distance between features of data points.
- Points that are close to each other in space are also considered similar to each other.

K Nearest Neighbors (KNN) Classifier

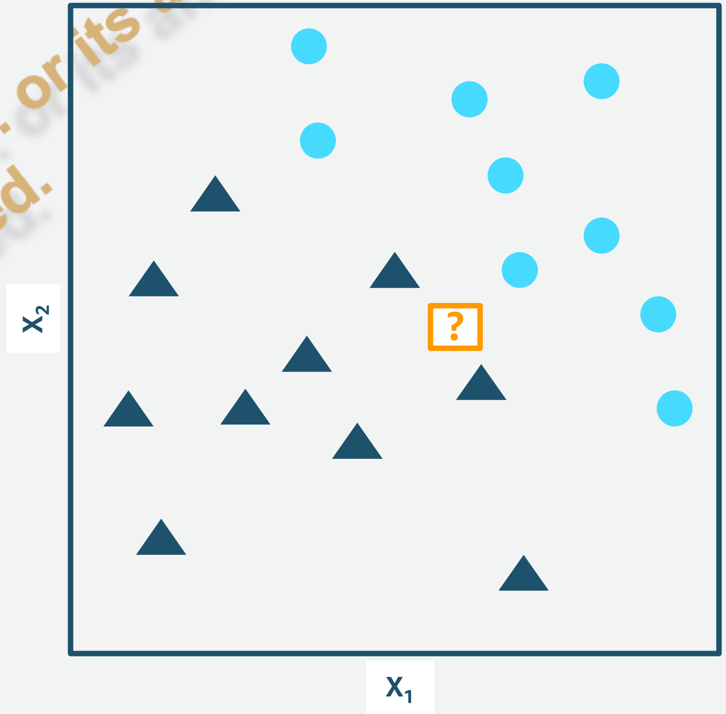
Assume a dataset:

- Two classes: ● and ▲
- Two features X_1 and X_2

K Nearest Neighbors Model:

- K=3

What class does ? belong?



K Nearest Neighbors (KNN) Classifier

Assume a dataset:

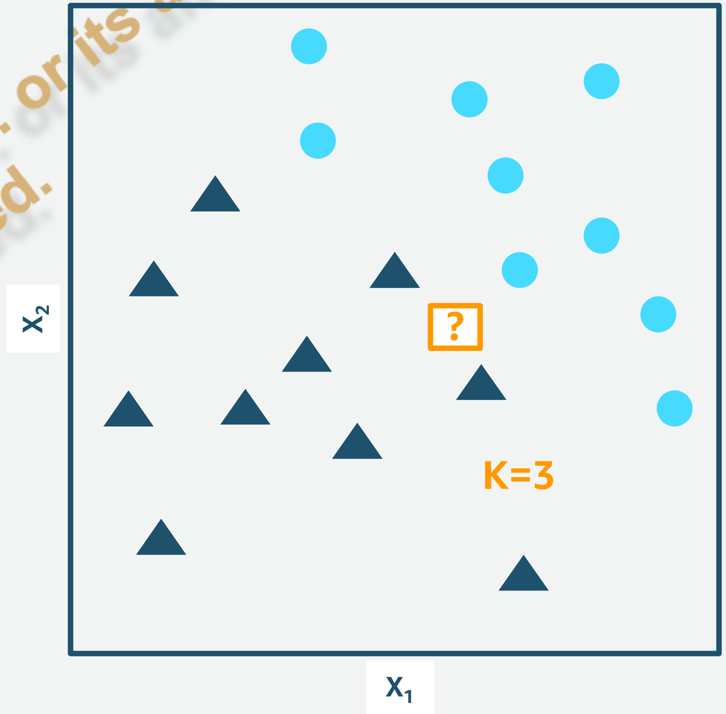
- Two classes: and
- Two features X_1 and X_2

K Nearest Neighbors Model:

- K=3

What class does ? belong?

- Look at the closest K points
- Pick the majority class:



K Nearest Neighbors Best Practices

Scaling

- Scale the features to values between 0-1.
- Otherwise, the model can rely on the features with large spreads.

K value

- Try and select an appropriate K number.
- Use a validation set or apply cross-validation for selecting K.

Curse of Dimensionality

- Suffers from high dimensional data (too many features).
- Spaces between points can get very large and our closer points ≈ similar records assumption may not hold very well.

We apply this method to our review data, and final project.

Feature Scaling

Motivation: Many algorithms are sensitive to features being on different scales, e.g., gradient descent and KNN

Solution: Bring features on to the same scale

Note: Some algorithms like decision trees and random forests aren't sensitive to features on different scales

Common choices (both linear):

- Mean/variance standardization
- MinMax scaling

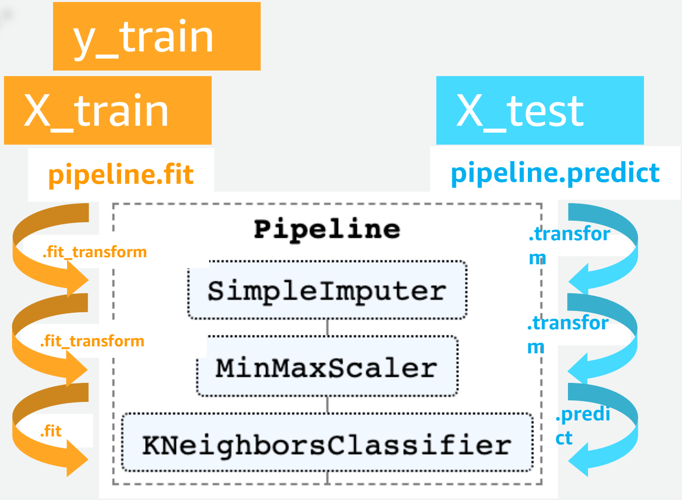
Pipeline in sklearn

Pipeline: sklearn sequential data transforms with a final estimator (prevents data leakage) – .fit(), .predict()

Pipeline(*steps*, *verbose=False*)

```
pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', MinMaxScaler()),
    ('clf', KNeighborsClassifier(n_neighbors = 3))
    ])

pipeline.fit(X_train, y_train)
predictions = pipeline.predict(X_test)
```



Thank you!