MACHINE LEARNING FUNDAMENTALS

Amir Ghaderi, Ph.D., P.Eng.
February – March 2020
Tillyard Auditorium

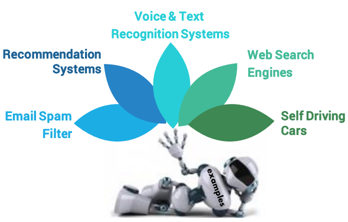
Session 1: Feb 19th

1

2

What is Machine Learning?

In its simplest definition **Machine Learning** is the science (and art) of programming computers so they can learn from data



“Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term “Machine Learning” in 1959 while at IBM ”

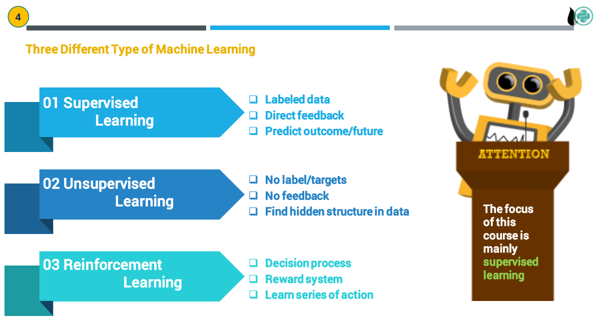
2

3

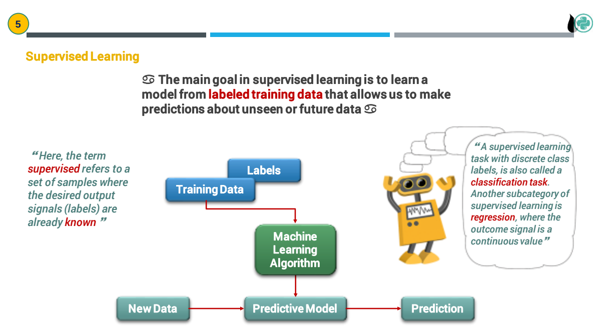
Any Application in Energy Sector? **Let's listen to Uncle Rob**

https://youtu.be/f7dhOHMX0js?t=457

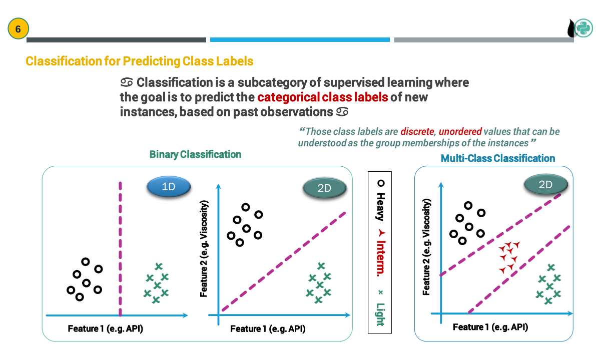
3


4
Three Different Type of Machine Learning
01 Supervised Learning
Labeled data
Direct feedback
Predict outcome/future
02 Unsupervised Learning
No label/targets
No feedback
Find hidden structure in data
03 Reinforcement Learning
Decision process
Reward system
Learn series of action
ATTENTION
The focus of this course is mainly supervised learning

4


5
Supervised Learning
♋ The main goal in supervised learning is to learn a model from labeled training data that allows us to make predictions about unseen or future data ♋
“Here, the term supervised refers to a set of samples where the desired output signals (labels) are already known ”
Labels
Training Data
Machine Learning Algorithm
New Data
Predictive Model
Prediction
“A supervised learning task with discrete class labels, is also called a classification task. Another subcategory of supervised learning is regression, where the outcome signal is a continuous value”

5

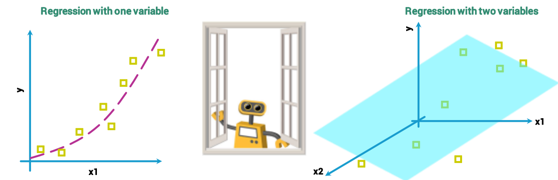

6
Classification for Predicting Class Labels
♋ Classification is a subcategory of supervised learning where the goal is to predict the categorical class labels of new instances, based on past observations ♋
“Those class labels are discrete, unordered values that can be understood as the group memberships of the instances”
Binary Classification
1D
Feature 1 (e.g. API)
2D
Feature 2 (e.g. Viscosity)
Feature 1 (e.g. API)
Heavy
Interm.
Light
Multi-Class Classification
2D
Feature 2 (e.g. Viscosity)
Feature 1 (e.g. API)

6

7

Regression for Predicting Continuous Outcomes

In regression analysis, we are given a number of predictor (explanatory) variables and a **continuous response** variable (outcome or target), and we try to find a relationship between those variables that allows us to predict an outcome

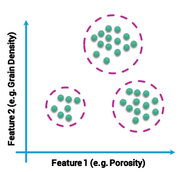


7

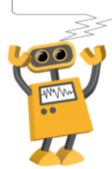
8

Discovering Hidden Structures with Unsupervised Learning

In unsupervised learning, unlabeled data or data of unknown structure should be dealt with. Using unsupervised learning techniques, it is possible to explore the structure of the data to extract meaningful information without the guidance of a known outcome variable or reward function



Sub-filed1: Finding Subgroups with Clustering

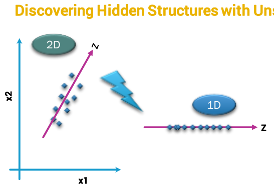


*"**Clustering** is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups"*

8

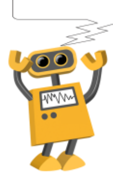
9

Discovering Hidden Structures with Unsupervised Learning



Isopach mapping in petroleum geology is an example of dimensionality reduction for easier visualization purpose

Sub-filed1: Dimensionality Reduction



*"**Dimensionality reduction** or dimension reduction is the process of reducing the number of random variables under consideration. by obtaining a set of principal variables. Approaches can be divided into feature selection and feature extraction."*

9

10

Basic Terminology and Notations

Rock Typing Data Set

Samples (instances, observations)

Core sample	Core poro	Core perm	Core grain size	Core grain density	Rock type
1	0.12	124.5	12	2489.9	Sandstone
2	0.03	na	9	1545.7	Dolomite
...	...	...	...	...	...
50	0.17	136.9	5	2333	Sandstone
...	...	...	...	...	...
303	0.091	9.5	8.5	na	Limestone

Missing Data

Features (attributes, measurements, explanatory variables)

Class Labels (targets, responses)

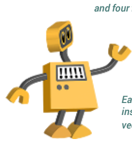
10

11

Basic Terminology and Notations

The "core samples" dataset consisting of 303 samples and four features can then be written as a $\boldsymbol{X} \in \mathbb{R}^{303\times4}$

$$\boldsymbol{X} = \begin{bmatrix} x_1^{(1)} & x_1^{(2)} & x_1^{(3)} & x_1^{(4)} \\ x_2^{(1)} & x_2^{(2)} & x_2^{(3)} & x_2^{(4)} \\ \vdots & \vdots & \vdots & \vdots \\ x_{303}^{(1)} & x_{303}^{(2)} & x_{303}^{(3)} & x_{303}^{(4)} \end{bmatrix}$$



*Superscript **i** to refer to the **i**th training sample, and the subscript **j** to refer to the **j**th dimension of the training dataset*

Each row in this feature matrix represents one rock instance and can be written as a four-dimensional row vector $\boldsymbol{x}^{(i)} \in \mathbb{R}^{1\times4}$

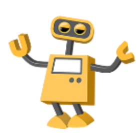
$$\boldsymbol{x}^{(i)} = \begin{bmatrix} x_1^{(i)} & x_2^{(i)} & x_3^{(i)} & x_4^{(i)} \end{bmatrix}$$

Each feature is a 150-dimensional column vector $\boldsymbol{x}_j \in \mathbb{R}^{303\times1}$

$$\boldsymbol{x}_j = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_{303}^{(1)} \end{bmatrix}$$

Similarly, we store the target variables (here, class labels) as a 303-dimensional column vector:

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \dots \\ y^{(303)} \end{bmatrix} (y \epsilon \{ss, dl, lm\})$$

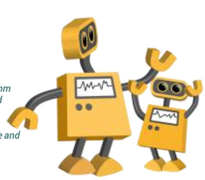


11

12

Training Data - Test Data

*"The observations in the **training set** comprise the experience that the algorithm uses to learn. In supervised learning problems, each observation consists of an observed response variable and one or more observed explanatory variables"*



*"The **test set** is a similar collection of observations that is used to evaluate the performance of the model using some performance metric. It is important that no observations from the training set are included in the test set."*

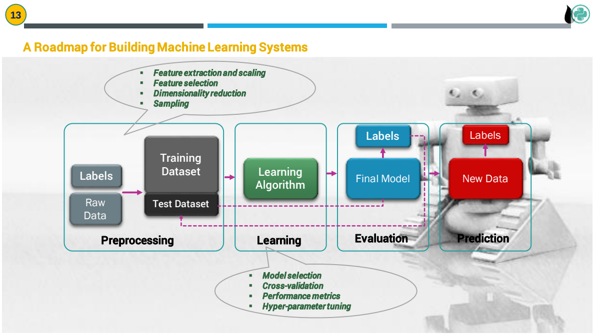
Memorization

Generalization

Over-fitting

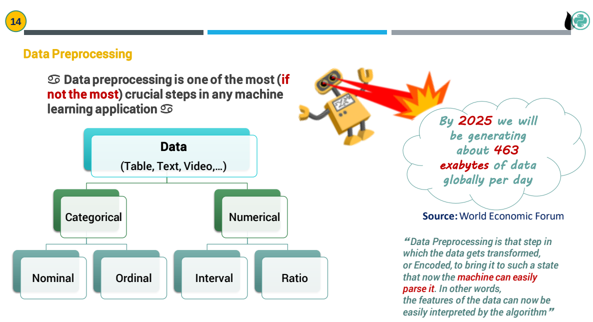
Under-fitting

12



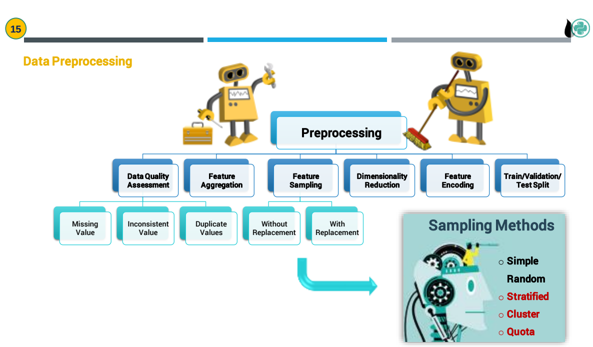
13
A Roadmap for Building Machine Learning Systems
Feature extraction and scaling
Feature selection
Dimensionality reduction
Sampling
Labels
Raw Data
Training Dataset
Test Dataset
Preprocessing
Learning Algorithm
Learning
Labels
Final Model
Evaluation
Labels
New Data
Prediction
Model selection
Cross-validation
Performance metrics
Hyper-parameter tuning

13



14
Data Preprocessing
♋ Data preprocessing is one of the most (if not the most) crucial steps in any machine learning application ♋
Data
(Table, Text, Video,...)
Categorical
Numerical
Nominal
Ordinal
Interval
Ratio
By 2025 we will be generating about 463 exabytes of data globally per day
Source: World Economic Forum
"Data Preprocessing is that step in which the data gets transformed, or Encoded, to bring it to such a state that now the machine can easily parse it. In other words, the features of the data can now be easily interpreted by the algorithm"

14



15
Data Preprocessing
Preprocessing
Data Quality Assessment
Feature Aggregation
Feature Sampling
Dimensionality Reduction
Feature Encoding
Train/Validation/ Test Split
Missing Value
Inconsistent Value
Duplicate Values
Without Replacement
With Replacement
Sampling Methods
Simple Random
Stratified
Cluster
Quota

15

16

Simple Linear Regression

♋ Simple linear regression can be used to model a **linear relationship** between **one response variable** and **one explanatory variable** (feature) ♋

Raw(Input) Data

Training Instance (scenarios)	CAPEX (MM$)	NPV (MM$)
1	46	70
2	48	90.2
3	50	130
4	54	175.1
5	58.5	180

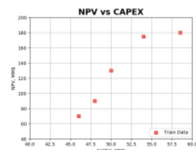
Python Code

Code 1

```
# code 1
import matplotlib.pyplot as plt
import numpy as np

actual_capex = np.array([[46], [48], [50], [54], [58.5]])
actual_npv = np.array([70, 90.2, 130, 175.1, 180])
fig, ax = plt.subplots()
ax.scatter(actual_capex, actual_npv,
           marker='s', s=50, c='#fa574b',
           label='Train Data')
font_dict = {'fontsize': 18, 'fontweight': 'bold'}
ax.set_title("NPV vs CAPEX", **font_dict)
ax.set_xlabel("CAPEX, MM$")
ax.set_ylabel("NPV, MM$")
ax.legend(loc='lower right')
ax.set_xlim((40, 60))
ax.set_ylim((40, 200))
ax.grid(True)
plt.show()
```

Matplotlib Output



16

17

Simple Linear Regression: Building Predictive Model Using Scikit-Learn Library

Code 2

```
# code 2
from sklearn.linear_model import LinearRegression
sl_model = LinearRegression() ❶
sl_model.fit(actual_capex, actual_npv) ❷
predicted_npv = sl_model.predict(np.array([[52]])) ❸
print(f"CAPEX of $MM 52 should bring about an NPV of $MM {predicted_npv[0]:.1f}")
```

❶ `LinearRegression` class is an ***estimator***. We create one object of this estimator class and name it model

❷ The `fit()` method of `LinearRegression` learns the parameters of the following model for simple linear regression

$$y = \underbrace{\beta_0}_{\text{Intercept}} + \underbrace{\beta_1}_{\text{coefficient(s)}} x$$

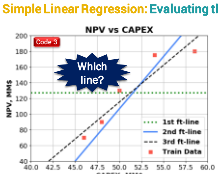
❸ After `LinearRegression` learned the parameters via fit method, `predict()` method can be used to predict the value of response for any explanatory variable

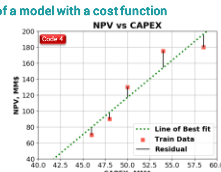
All estimators in scikit-learn implement fit() (to learn parameters of the model) and predict() (to predict the value of response) methods.

17

18

Simple Linear Regression: Evaluating the fitness of a model with a cost function





❶ A cost function, also called a loss function, is used to define and measure the error of a model.

❷ The differences between the model predicted by the model and the observed response in the training set are called **residuals** or **training errors**.

❸ the differences between the predicted and observed values in the test data are called **prediction errors** or **test errors.**

❶ Variety of cost functions can be defined to evaluate of the model's fitness; i.e., MAE, MSE

❷ A typical cost function for regression problem is residual sum of squares, which upon minimization will generate optimized values for fit function

18

19

Select a Performance Measure (Metric): MSE, RMSE, MAE, SSR, R^2, Adj. R^2,…

y_i Actual Expected Output

$\hat{y}_i$ Model's Prediction

Mean Squared Error

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$$

Mean Absolute Error

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|$$

MSE	RMSE	MAE	R^2
[0 +∞)	[0 +∞)	[0 +∞)	(-∞ 1]

When to use each?

$$\text{RMSE} = \sqrt{MSE}$$

Root Mean Squared Error

$$R^2 = 1.0 - \frac{MSE\ (model)}{MSE\ (baseline)}$$

R-Squared

$$\text{MSE(baseline)} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \bar{y})^2$$

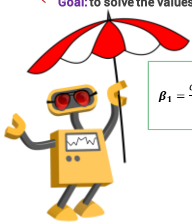
19

20

Simple Linear Regression: Solving ordinary least squares for simple linear regression

$$\boxed{y = \beta_0 + \beta x}$$

Goal: to solve the values of β_0 and β_1 that minimize the cost function



$$\beta_1 = \frac{cov(x,y)}{var(x)} = \frac{\frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}}{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

$$\beta_0 = \bar{y} - \beta_1\bar{x}$$

Code 5 ❶ Calculate β_0 and β_1 for training data set

Code 5 ❷ Compare the result from task1 with those of SKL model

20

21

Simple Linear Regression: Evaluating the model

So far: We have used a learning algorithm to estimate a model's parameters from the training data

How can we assess whether our model is a good representation of the real relationship?

❶

Test Instance	CAPEX (MM$)	NPV (MM$)	Predicted NPV (MM$)
1	48	110	
2	49	85	
3	51	150	
4	56	180	
5	52	110	

Code 6 ❶ Use fitted model to fill the column in

❷ Several measures can be used to assess our model's predictive capabilities. For example, **r-squared**, measures how well the observed values of the response variables are predicted by the model.

$$R^2 = 1.0 - \frac{SS_{res}}{SS_{tot}} = 1.0 - \frac{\sum_{i=1}^{n}(y_i - h(x_i))^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Code 6 ❷ Calculate R^2 for test data and compare with SKL results

21

22

Multiple Linear Regression: Upgrade of the Simple Linear Model

You might have some intuitions about the effect of time on the projects' NPV

Why R^2 is poor?

Training Instance	CAPEX (MM$)	CAPEX Spent Time (Year)	NPV (MM$)
1	46	5	70
2	48	4	90.2
3	50	3	130
4	54	5	175.1
5	58.5	3	180

Test Instance	CAPEX (MM$)	CAPEX Spent Time (Year)	NPV (MM$)
1	48	5	110
2	49	3	85
3	51	5	150
4	56	5	180
5	52	3	110

We cannot proceed with simple linear regression, but we can use a generalization of simple linear regression that can use multiple explanatory variables called multiple linear regression.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

multiple linear regression uses one coefficient per each explanatory variables (an arbitrary number of variables can be used)

22

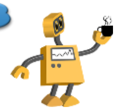
23

Multiple Linear Regression: Solve for coefficient using normal equation

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} \beta_0 + \beta_1 x_1^{(1)} + \beta_2 x_2^{(1)} \\ \beta_0 + \beta_1 x_1^{(2)} + \beta_2 x_2^{(2)} \\ \vdots \\ \beta_0 + \beta_1 x_1^{(n)} + \beta_2\, x_2^{(n)} \end{bmatrix} = ?$$

❶ Decompose the last matrix with summation terms into two matrices multiplication

X is called design matrix



$$\boxed{Y = X\beta}$$

$$\boxed{\beta = \left(X^T X\right)^{-1} X^T Y}$$

Closed-form solution for normal equation

❷ When obtaining β values might be a challenge using this approach?

Code 7

❸ Write a code to calculate β values for the train data (table in previous slide)

23

24

Multiple Linear Regression: SKL implementation of MLR

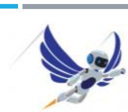
Code 8

```
import numpy as np
from sklearn.linear_model import LinearRegression

# from this point on we use the typical variables names for machine learning
x_train = np.array([[46, 5], [48, 4], [50, 3], [54, 5], [58.5, 3]])
y_train = np.array([70, 90.2, 130, 175.1, 180])

x_test = np.array([[48, 5], [49, 3], [51, 5], [56, 5], [52, 3]])
y_test = np.array([110, 85, 150, 180, 110])

ml_model = LinearRegression()
ml_model.fit(x_train, y_train)
beta0, beta = ml_model.intercept_, ml_model.coef_
print(f"SKL:\n Beta0 = {beta0:.4f}; Beta1 = {beta[0]:.4f}; Beta2 = {beta[1]:.4f}")
r2 = ml_model.score(x_test, y_test)
print(f"SKL:\n Model's R2 = {100 * r2:.2f}%")
```



It appears that adding the spent time as an explanatory variable has improved the performance of our model

Caution: Evaluating a model on a single test set can provide inaccurate estimates of the model's performance. We can estimate its performance more accurately by training and testing on many partitions of the data.

24

25

Polynomial Regression: Special form of MLR

So far, we assumed that the real relationship between the explanatory variables and the response variable is linear. This assumption is not always true.

Polynomial regression (a special case of multiple linear regression) can add terms with degrees greater than one to the model

Quadratic regression, or regression with a second order polynomial, is given by the following formula (one explanatory variable): $y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2$

Task ❶ What is the form of this equation in case of two explanatory variables?

Task ❷ What is the general form of this equation?

Note that the equation for polynomial regression is the same as the equation for multiple linear regression in vector notation

25

26

Polynomial Regression: SKL implementation of polynomial regression

Code 9

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

x_train = np.array([[6], [8], [10], [14], [18]])
y_train = np.array([7, 9, 13, 17.5, 18])
x_test = np.array([[6], [8], [11], [16]])
y_test = np.array([8, 12, 15, 18])

quadratic_featurizer = PolynomialFeatures(degree=2) ❶
x_train_quadratic = quadratic_featurizer.fit_transform(x_train) ❷
x_test_quadratic = quadratic_featurizer.transform(x_test) ❸

quadratic_regressor = LinearRegression()
quadratic_regressor.fit(x_train_quadratic, y_train)
r_sq = quadratic_regressor.score(x_test_quadratic, y_test)
print(f"Quadratic regression r-square: {r_sq:.4f}")
```

❶ `PolynomialFeatures` class is a transformer. We create one object of this transformer (here, quadratic_featurizer) to transform our explanatory variables.

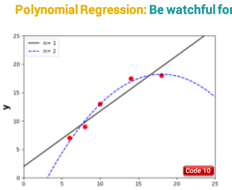
❷ `fit_transform()` method of the transformer is used to tune the transformer and transform the train data into proper shape

❸ `transform()` method of the transformer is used to *transform the test data* into proper shape

26

27

Polynomial Regression: Be watchful for over-fitting

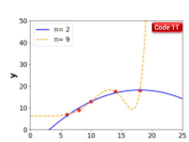


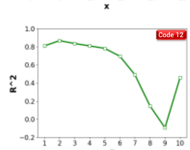
Task ❶ Try to regenerate this plot

n = 1 | R^2 = 81%

n = 2 | R^2 = 87%

- While quadratic and cubic regression models are the most common, we can add polynomials of any degree.
- The 9th polynomial regression model fits the training data almost exactly! The model's r-squared score, however, is -0.09. We created an extremely complex model that fits the training data exactly but fails to approximate the real relationship.
- This problem is called over-fitting.





27