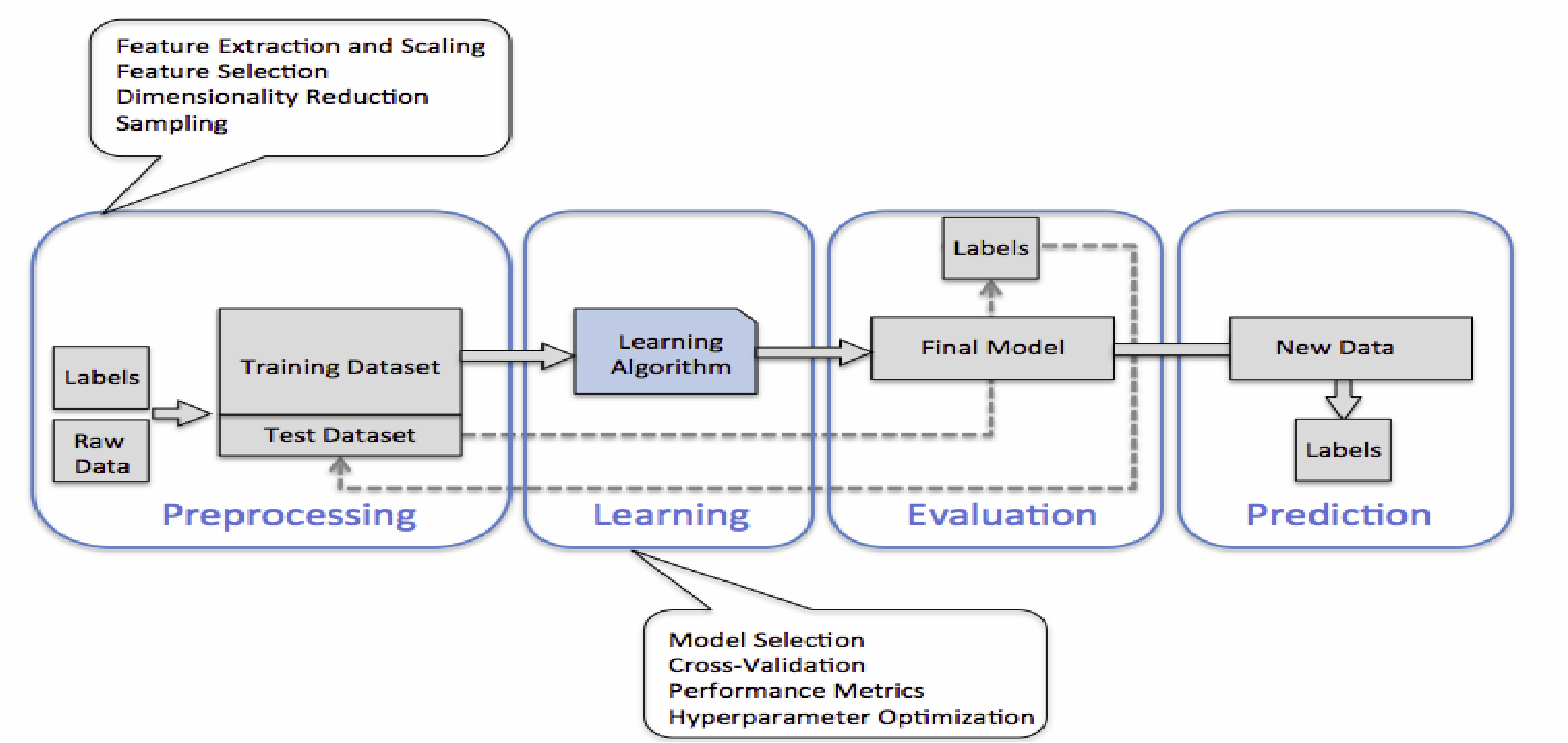
NLP for SE and AI Techniques

Agenda

- Advanced Evaluation Techniques (cont.)
- Supervised learning
  - KNNs
  - Linear Regression (OLS)
  - Ridge Regression
  - Lasso Regression

A roadmap for building machine learning systems

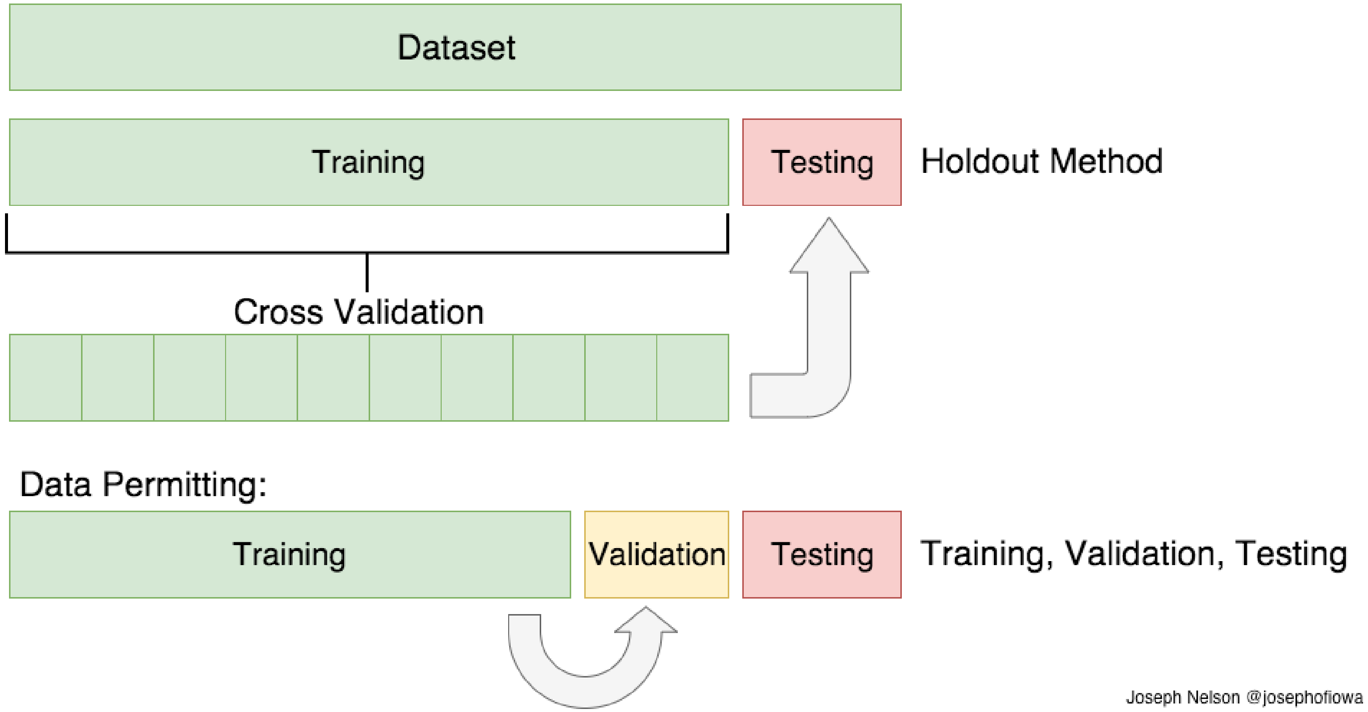


ADV. Evaluation Techniques (when we don’t have enough or quality dataset)

- Cross Validation
- Stratified Cross Validation
- Leave-One-Out Cross validation
- Bootstrapping

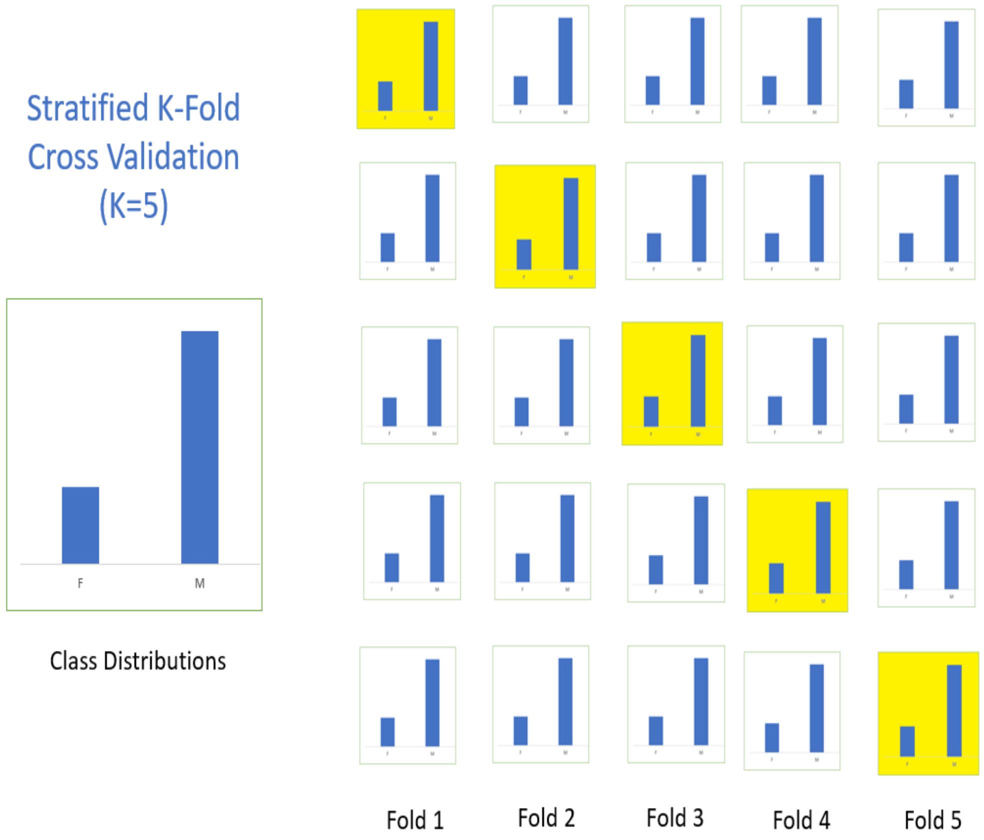
K-folds Cross-validation Method

- AKA. Rotation estimation
- Use to estimate a performance of the mode (i.e. mean of accuracy rate)



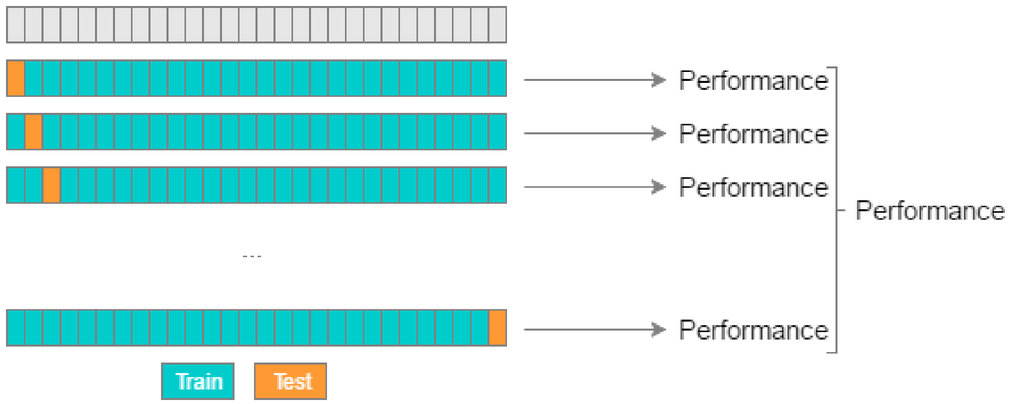
Stratified Cross-validation Method

- Same as Cross-validation but here we ensure that each fold is representative of all strata of the class.

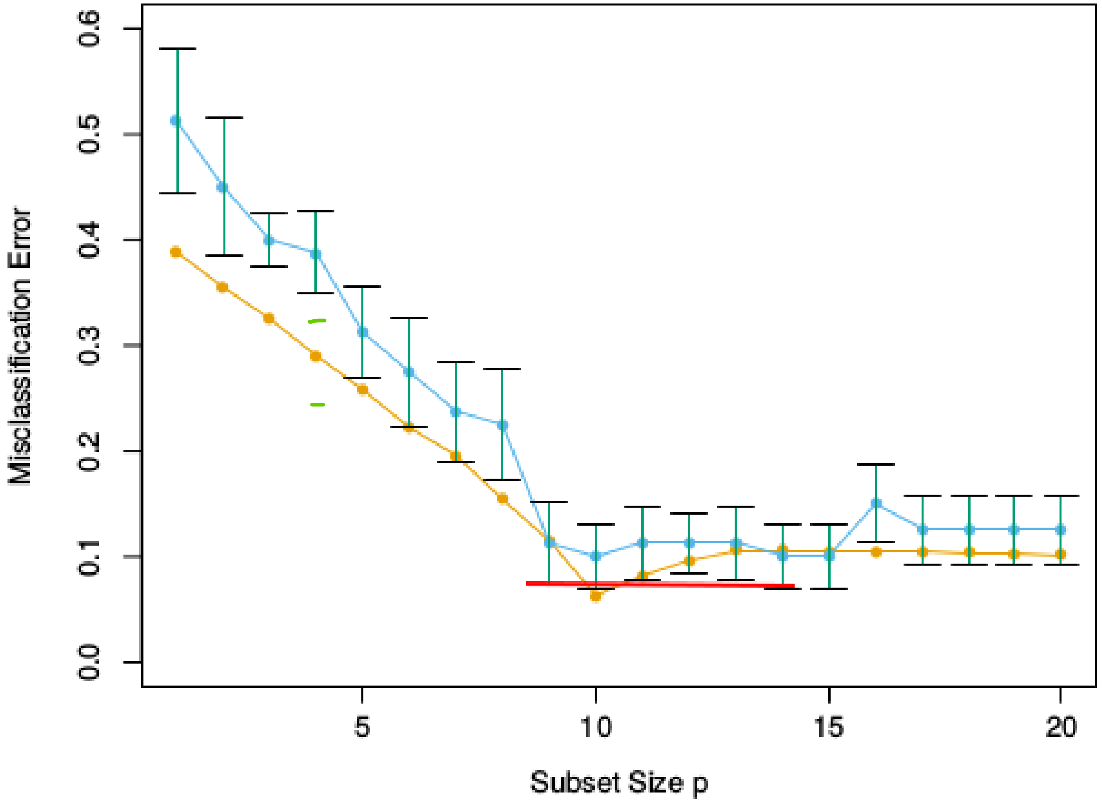


Leave-one-out Cross-validation Method

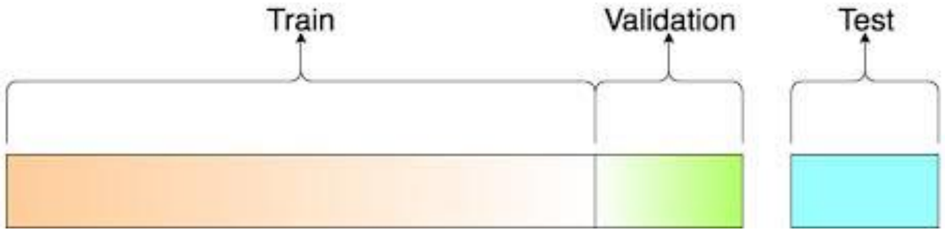
- Cross-validation for small sample size.
- The number of folds is the same as the number of training instances.
- Advantages:
  - Makes the best use of the data
  - Involve no random sampling
- Disadvantages:
  - Took long time to run, computationally expensive



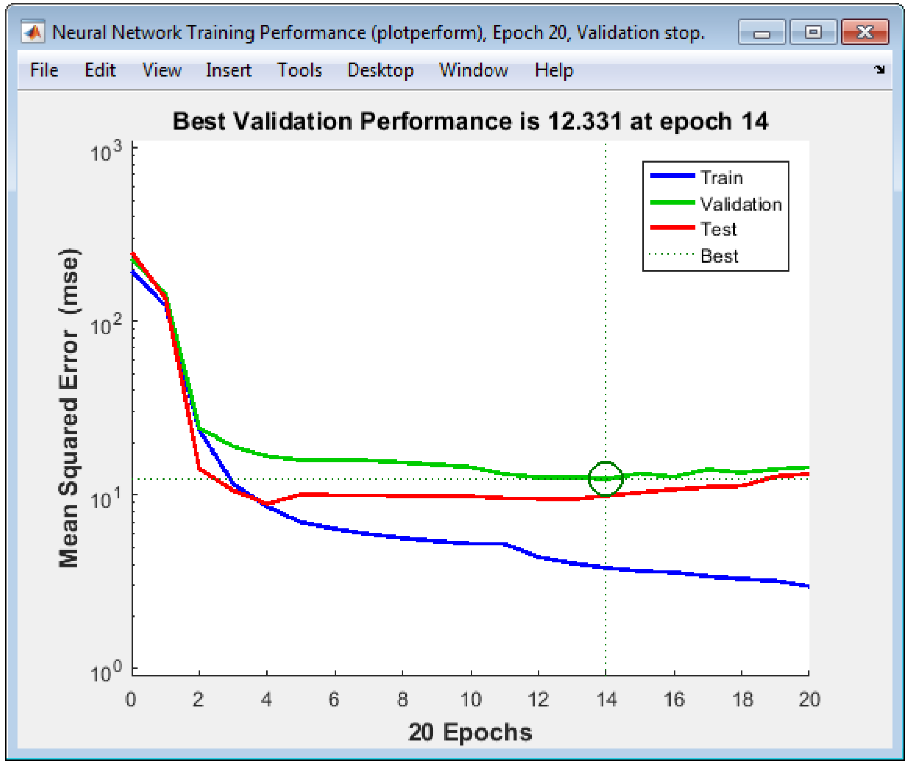
K-fold cross validation (cont.)



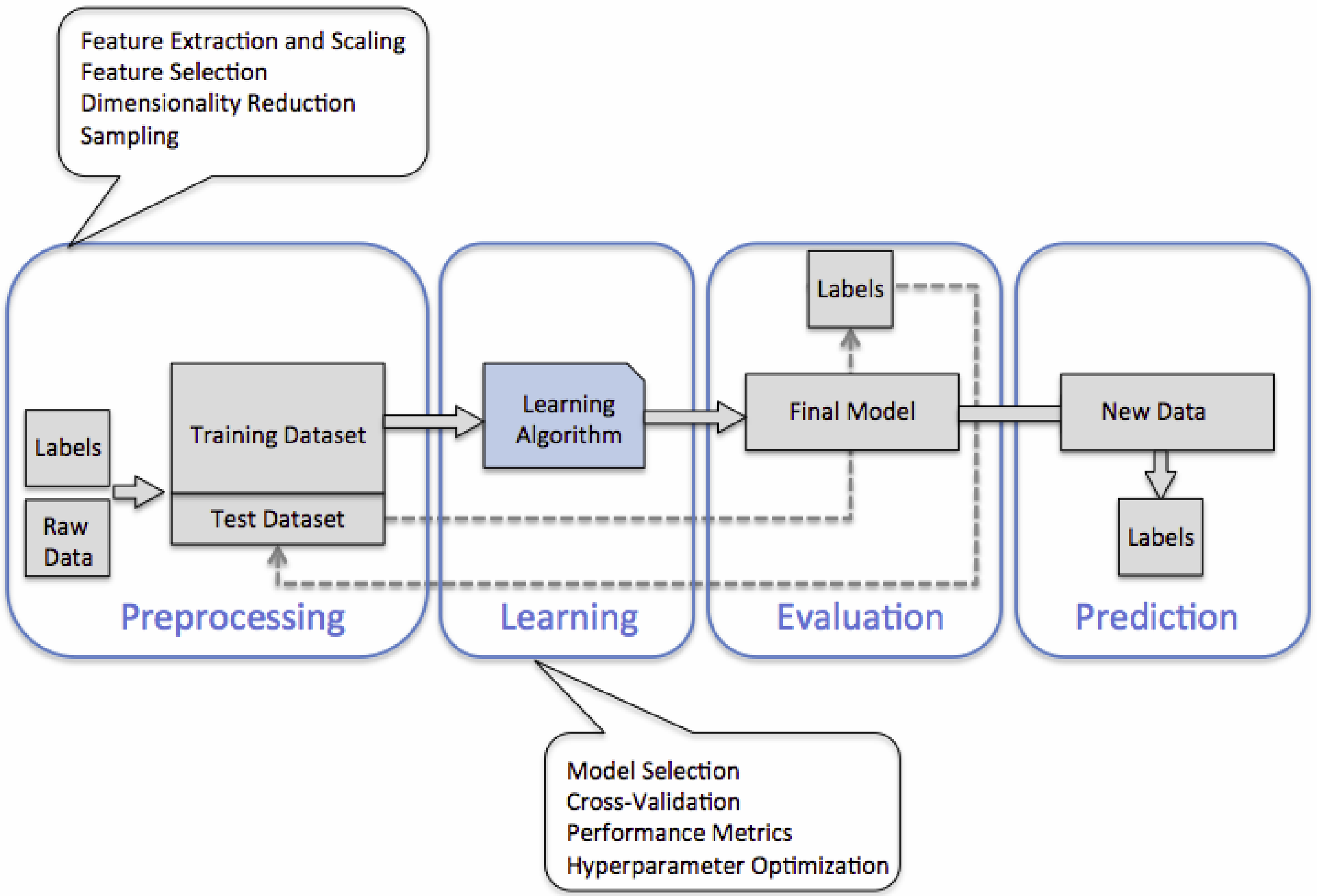
Three-way sampling method



Validation set is use for tuning parameters of your Model.



Recall: Road map to Data Mining/ Machine learning



Intro to ML

- Previous, Intelligent” applications, many systems used hand coded rules of “if ” and “else” decisions to process data or adjust to user input.
  - Changing the task even slightly might require a rewrite of the whole system.
  - Designing rules requires a deep understanding of how a decision should be made by a human expert.

AI vs. machine learning vs. deep learning

	AI	Machine learning	Deep learning
Optimal data volumes	Varying data volumes	Thousands of data points	Big data: millions of data points
Outputs	Anything from predictions to recommendations to decision-making	Numerical value, like a classification or score	Anything from numerical values to free-form elements, like free text and sound
How it works	Machines are programmed to mimic human activity with human-like accuracy	Uses various types of automated algorithms that learn to model functions and predict future actions from data	Uses neural networks that pass data through many processing layers to interpret data features and relationships
How it's managed	Algorithms require human oversight in order to function properly	Algorithms are directed by data analysts to examine specific variables in data sets	Algorithms are largely self-directed on data analysis once they're put into production

Machine learning algorithm	Data processing tasks	Section	Representative references
K-Nearest Neighbors	Classification	5.1.1	[58] [59]
Naive Bayes	Classification	5.1.2	[60] [61]
Support Vector Machine	Classification	5.1.3	[62] [63] [64] [65]
Linear Regression	Regression	5.2.1	[66] [66] [67] [68]
Support Vector Regression	Regression	5.2.2	[69] [70]
Classification and Regression Trees	Classification/Regression	5.3.1	[71] [72] [73]
Random Forests	Classification/Regression	5.3.2	[74]
Bagging	Classification/Regression	5.3.3	[75]
K-Means	Clustering	5.4.1	[76] [77] [78]
Density-Based Spatial Clustering of Applications with Noise	Clustering	5.4.2	[79] [80] [81]
Principal Component Analysis	Feature extraction	5.5.1	[82] [83] [84] [85] [86]
Canonical Correlation Analysis	Feature extraction	5.5.2	[87] [88]
Feed Forward Neural Network	Regression/Classification/ Clustering/Feature extraction	5.6.1	[89] [90] [91] [92] [93] [57]
One-class Support Vector Machines	Anomaly detection	5.8.1	[94] [95]

(Mahdavinejad et al., 2018)

The real challenge in using ML is to find the algorithm whose learning bias is the best match for a particular data set.

Predictive Model (in short)

- Construct a model from historical data to make a prediction on unseen data (e.g. we don’t know the answer)

- The job of machine learning functions is to find the optimal mapping function.
- There are two ways
  - Classification
  - Regression

Classification vs. Regression

Classification is the task of predicting a discrete class label.

Regression is the task of predicting a continuous quantity.

Classification vs. Regression

However,

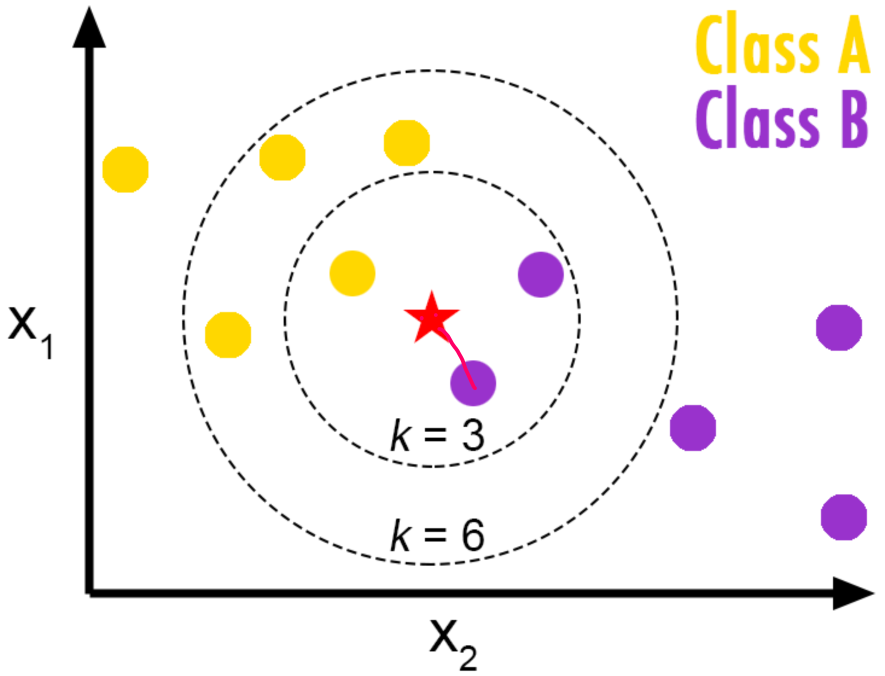
- A classification algorithm **may/can** predict a continuous value, **but** the continuous value is in the form of a probability for a class label.
- A regression algorithm **may/can** predict a discrete value, **but** the discrete value in the form of an integer quantity.

K-Nearest Neighbors

- One of the **most simplest** classification algorithm
- Its belonged to **supervised learning** algorithm.
- One of the most used and produce good performance
- Use similarity measurements
- Can be use in **regression** and **classification** problems
- Non-parametric (don’t need to follow specific distribution of data)

K-Nearest Neighbors (cont.)

The object is assigned a class of its nearest neighbor



K-Nearest Neighbors (cont.)

1. Compute a distance value between the item to be classified and every item in the training data-set

2. Pick the ***k*** closest data points (the items with the ***k*** lowest distances)

3. Conduct a “majority vote” among those data points — the dominating classification in that pool is decided as the final classification

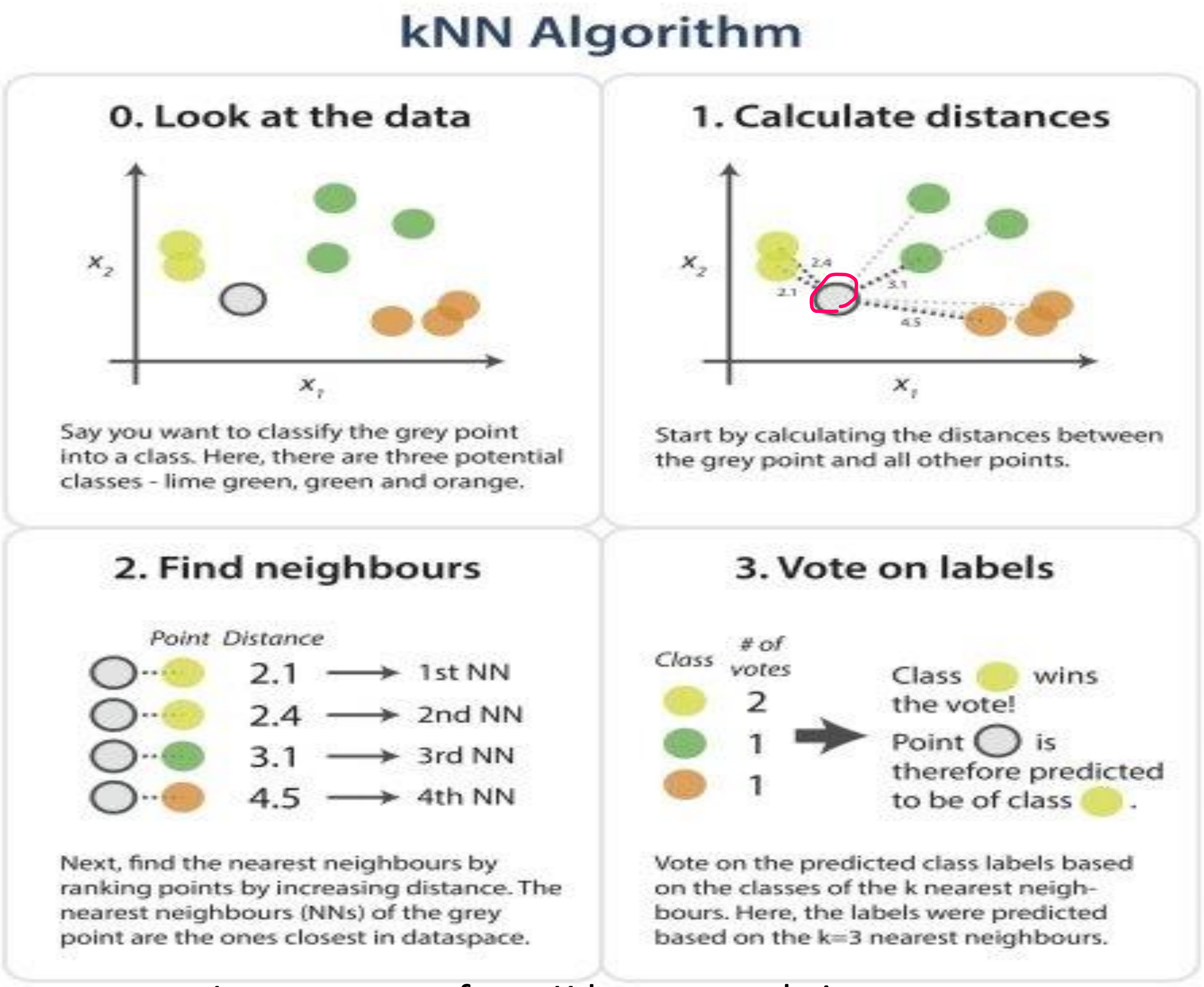
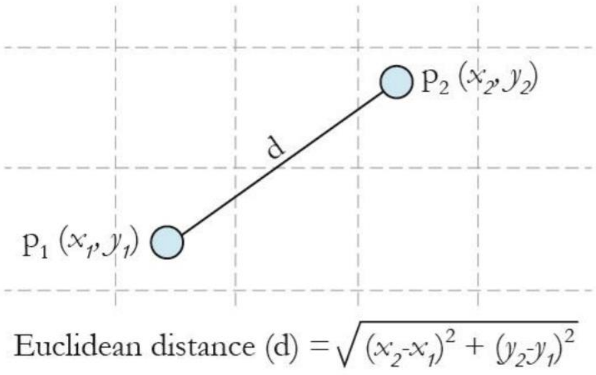


Image source from Kdnugget websites

How to get the k value?

- Euclidean distance

$$E(x,y) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}$$



- Cosine similarity

  Measurement between two non-zero vectors

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

K-Nearest Neighbors Algorithm ImpL.(cont.)

1. **Load** the data
2. Initialize the value of ***k***
3. To get the predicted class, iterate from 1 to total number of training data points
    1. **Calculate the distance** to all training data. Here, we will use Euclidean distance as our distance metric since it’s the most popular method.
    2. **Sort** the calculated distances in **ascending order** based on distance values
    3. Get **top *k* rows** from the sorted array
    4. Get **the most frequent class (Majority vote)** of these rows
    5. Return the predicted class

Scikit-learn library for python

- **Install Scikit-learn from (http://scikit-learn.org/)**
- **from** sklearn.neighbors **import** KNeighborsClassifier
- **from** sklearn.metrics **import** accuracy_sc
- **from** sklearn.cross_validation **import** train_test_split

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

KNN from scratch

```python
# a Counter is a collection where elements are stored as dictionary keys,
# and the key's counts are stored as dictionary values. The example below illustrates this.
from collections import Counter

import numpy as np


def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))

class KNN:
    def __init__(self, k=3):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y

    def predict(self, X):
        y_pred = [self._predict(x) for x in X]
        return np.array(y_pred)

    def _predict(self, x):
        # Compute distances between x and all examples in the training set
        distances = [euclidean_distance(x, x_train) for x_train in self.X_train]

        # Sort by distance and return indices of the first k neighbors
        k_idx = np.argsort(distances)[: self.k]

        # Extract the labels of the k nearest neighbor training samples
        k_neighbor_labels = [self.y_train[i] for i in k_idx]

        # return the most common class label
        most_common = Counter(k_neighbor_labels).most_common(1)
        return most_common[0][0]
```

```python
if __name__ == "__main__":

    from sklearn import datasets
    from sklearn.model_selection import train_test_split


    def accuracy(y_true, y_pred):
        accuracy = np.sum(y_true == y_pred) / len(y_true)
        return accuracy

    iris = datasets.load_iris()
    X, y = iris.data, iris.target

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=1234
    )

    k = 3
    clf = KNN(k=k)
    clf.fit(X_train, y_train)
    predictions = clf.predict(X_test)
    print("KNN classification accuracy", accuracy(y_test, predictions))
```

KNN Demo with sklearn

- https://github.com/preenet/961701_65/blob/main/Classification_demo_with_KNN.ipynb

Workshop (Term project Proposal(mini))

- Maximum number of 3 people
- Maxmum number 3-5 pages
  - NLP related problem
  - (dataset should contain at least one feature that is textual)
- Proposal content
  - Introduction to your problem
  - Activities, output(s), outcome, identify
  - Stakeholders
  - Data sources
  - DS tasks
  - Solution as software
- For master students (Replicate conference paper that is related to NLP to get 100%)

Agenda

- Term project due a week before final exam.
- One more workshop (ML topic)

Pros and Cons

- Pros
  - Simple
  - Can apply to both regression and classification
  - No data assumption needs
- Cons
  - Sensitive to scale of data
  - Curse of demission
  - Outlier sensitive
  - Missing value treatment

More KNN

- How to handle categorical variables?
  - Use dummy variables
- Optimal k value?
  - Perform model selection

KNN for regression

- KneighborsRegressor
- Use the same distance function like the classifier version

Mean absolute error

Overview | Formula

Formula

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

MAE = mean absolute error

y_i = prediction

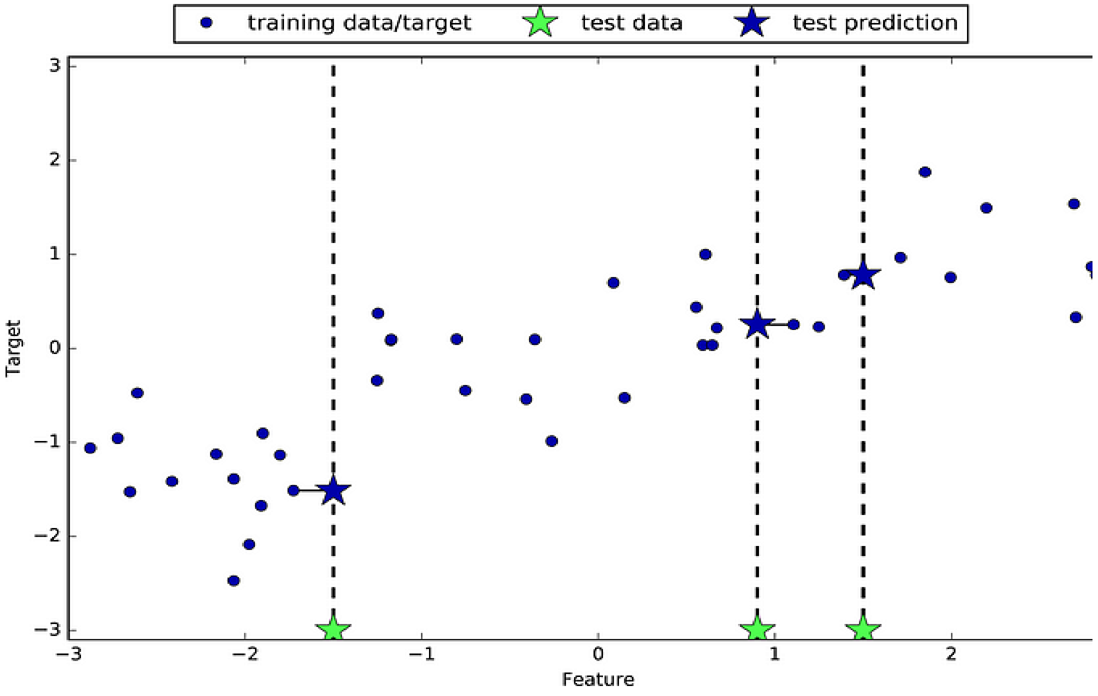
x_i = true value

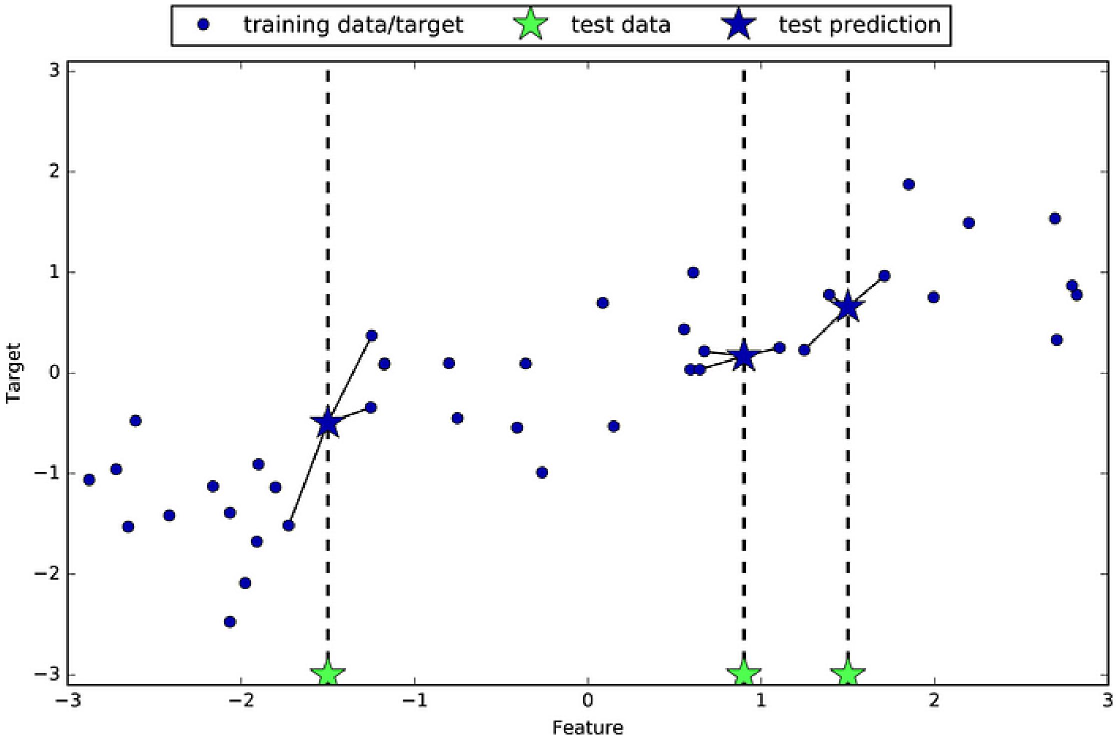
n = total number of data points

KNN regressor example

```python
# Import the necessary libraries
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Generate some sample data
X = np.random.rand(100, 1)  # Feature
y = 2 * X + np.random.randn(100, 1)  # Target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a KNN regressor with a specified number of neighbors (e.g., 5)
k = 5
knn_regressor = KNeighborsRegressor(n_neighbors=k)

# Fit the model to the training data
knn_regressor.fit(X_train, y_train)

# Make predictions on the test data
y_pred = knn_regressor.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

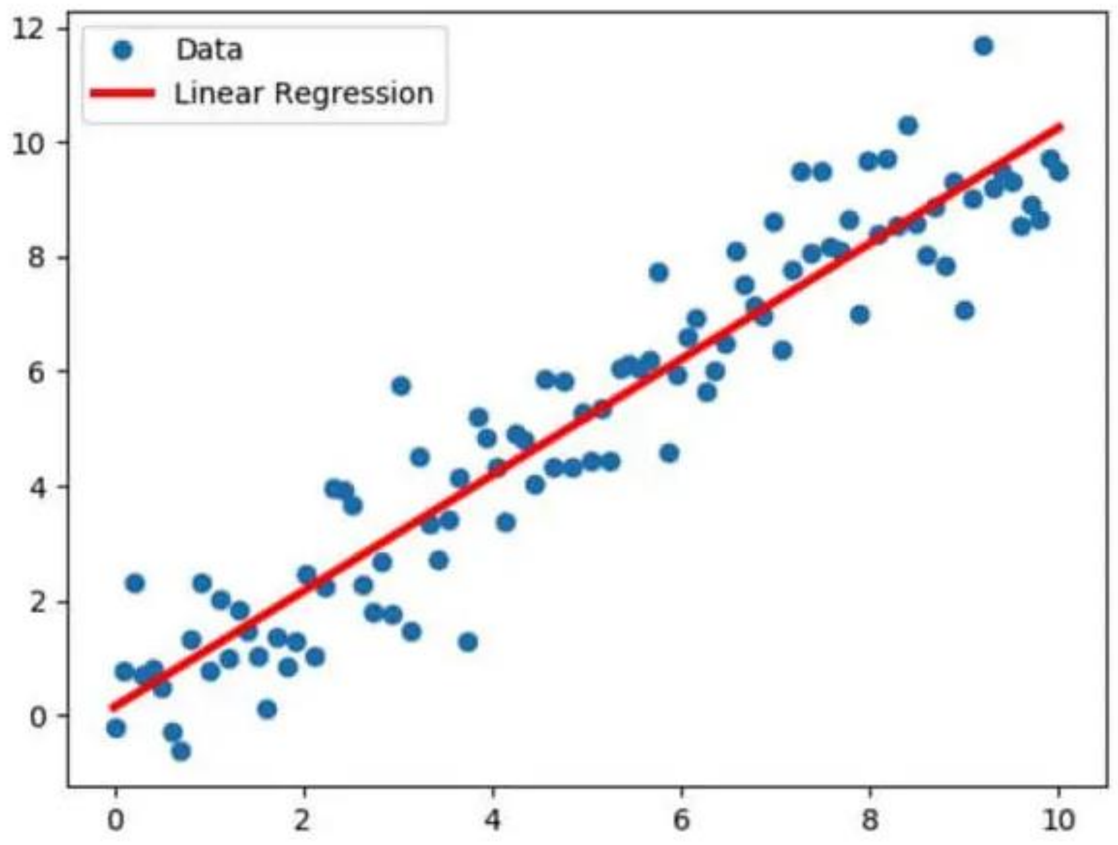
KNN regressor





Linear regression

- When we need model interpretability
- When the relationship tends to be linear



Regression introduction

- Regression – a task of approximating a mapping function (f) from input variables (X) to a continuous output variable (y).

- A continuous output variable real number(R), (e.g. int or float values.)
    - These are often quantities (e.g. amounts and sizes.)

- A problem with multiple input variables aka. multivariate regression problem.

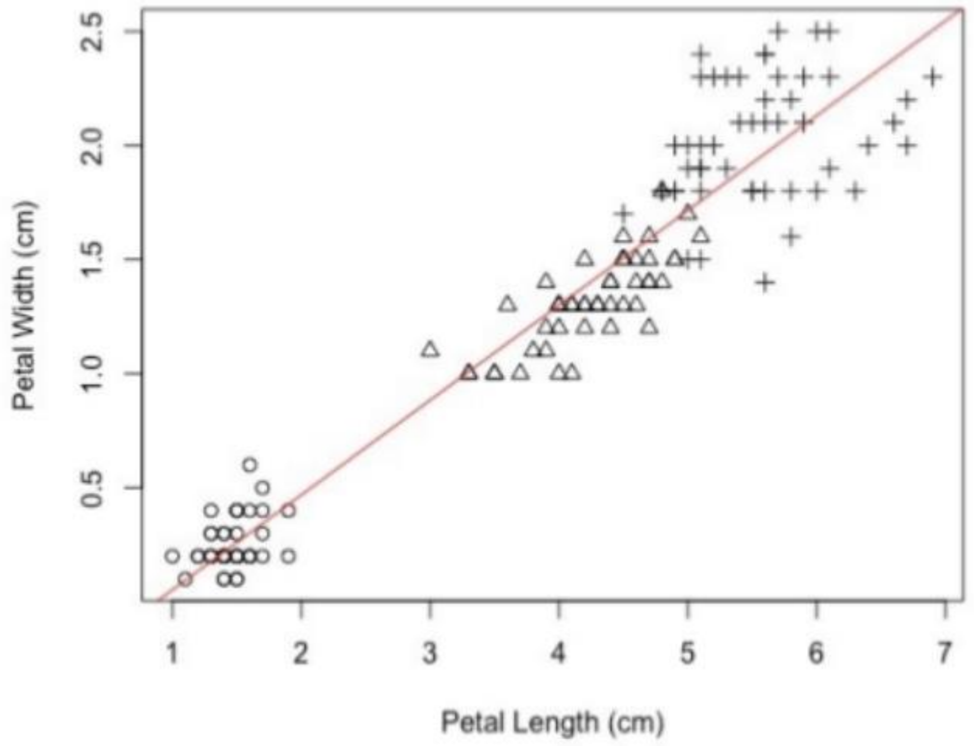
Regression introduction

- Classification predictions can be evaluated using accuracy, whereas regression predictions cannot.
- Accuracy = correct_pred / total_pred x 100

- Regression predictions can be evaluated using root mean squared error, whereas classification predictions cannot.
- RMSE = *sqrt*(avg(error^2))

Linear Regression for Iris dataset

- We can use linear regression as machine learning algorithm to predict petal_width given petal_length.



Least Squares - Linear Regression

- Linear regression function
  - $\hat{y} = b_0 + b_1 x$ or you may know from basic math such as (y = mx+b)
- Slope or Gradient (how steep the line is)
  - $b_1 = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}$
- Y-intercept (where the line cut the Y-axis)
  - $b_0 = \bar{y} - b_1 \bar{x}$

Linear regression- intro

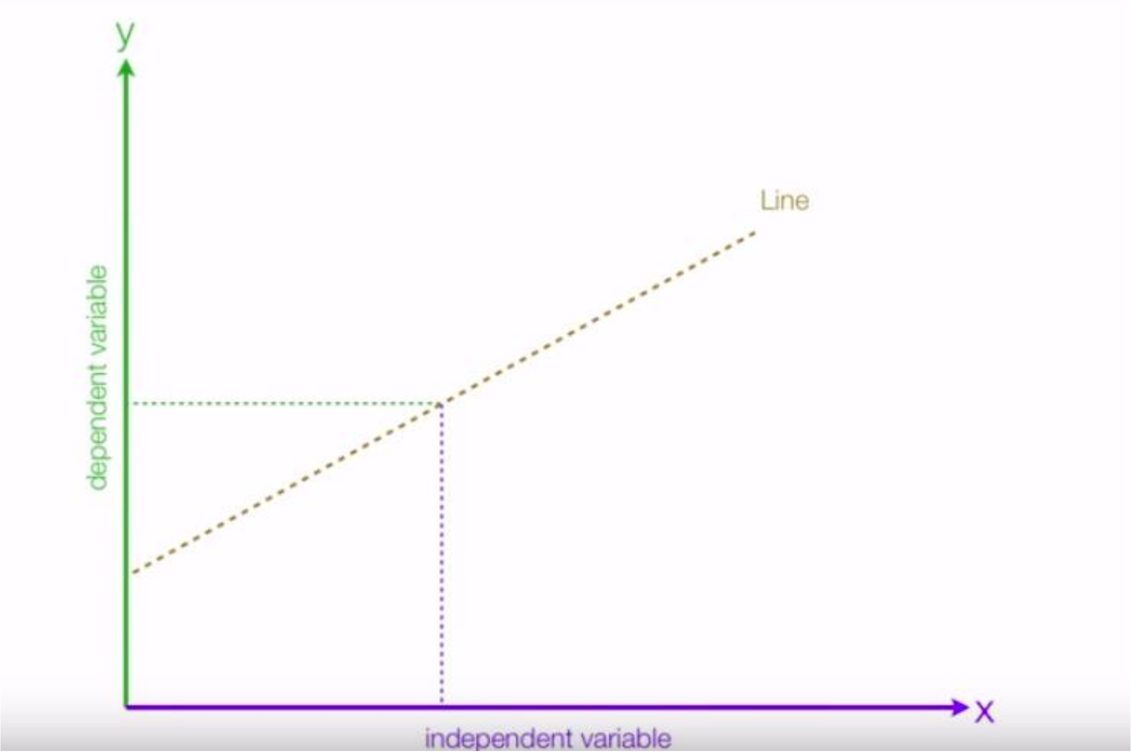
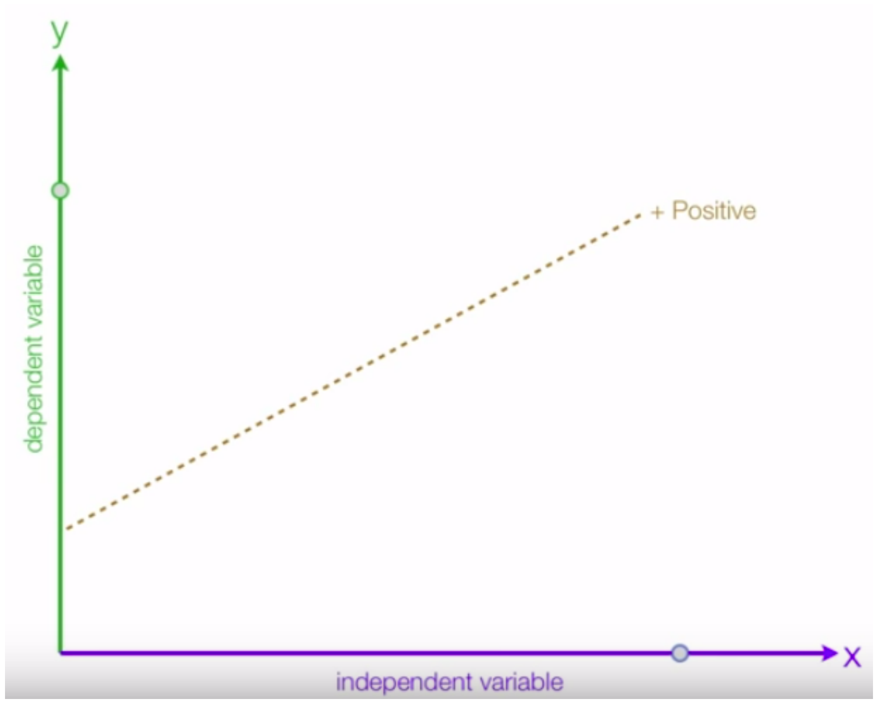


Image source: David Longstreet, Professor of the Universe, MyBookSucks, (Accessed, August 2018)

Linear regression- intro2



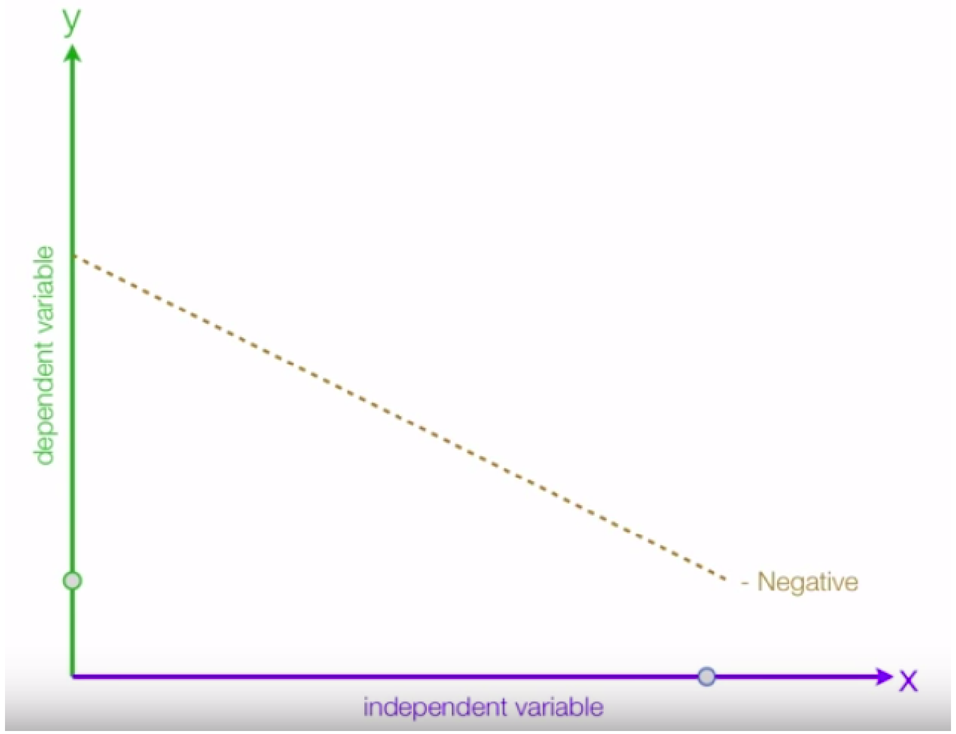
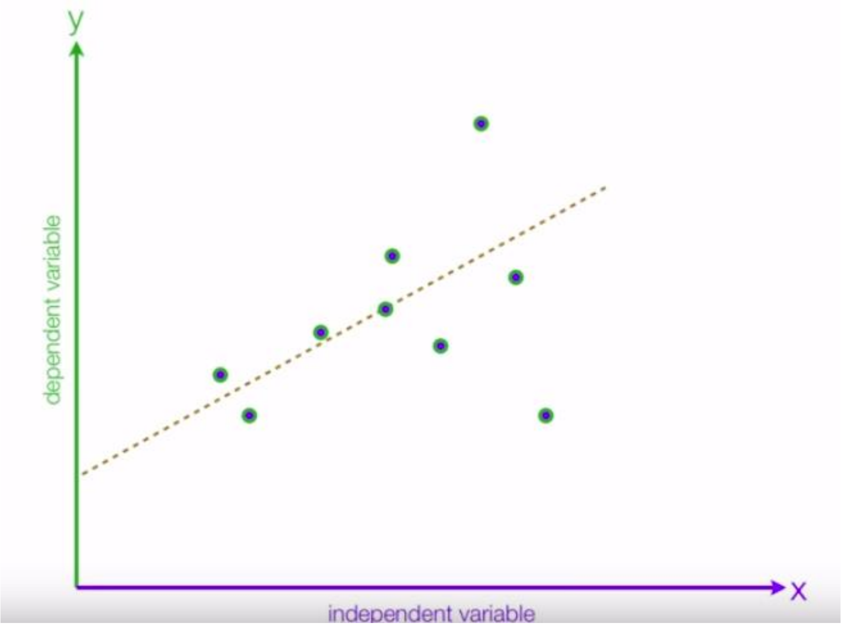


Image source: David Longstreet, Professor of the Universe, MyBookSucks, (Accessed, August 2018)

Linear regression- intro3

- We try to find the line that can fit the observation points
the line is called regression line

- We try to draw the line that minimize the errors



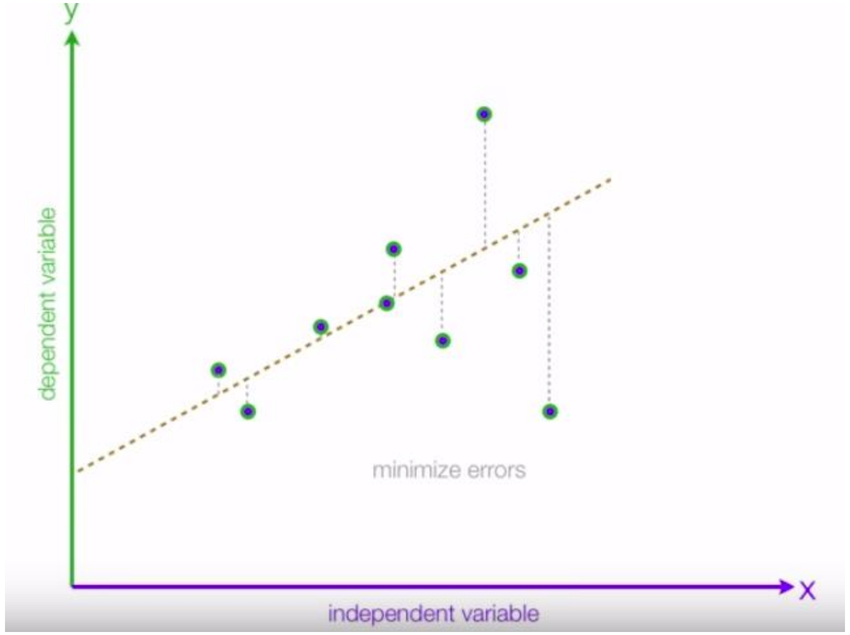
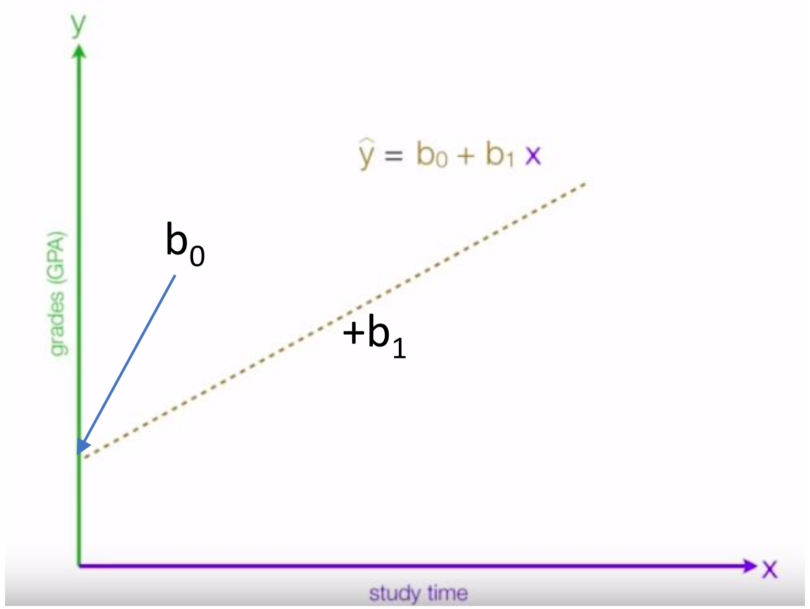


Image source: David Longstreet, Professor of the Universe, MyBookSucks, (Accessed, August 2018)

Linear regression- intro4

- $\hat{y}$ is the estimated value (dependent variable or output)
- x is the independent value (input, we can control)
- b_0 is the y-intercept
- b_1 is the slop of the regression line



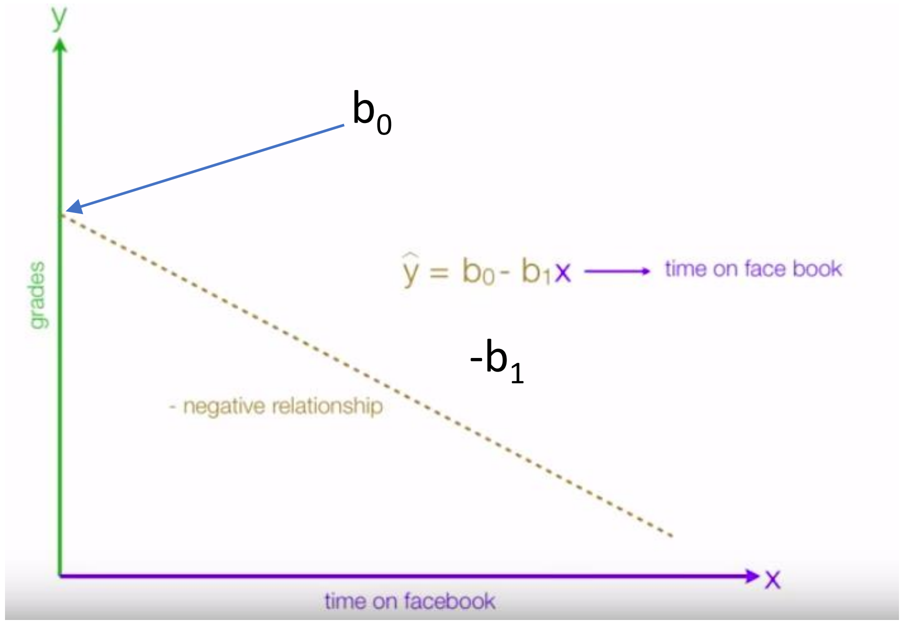
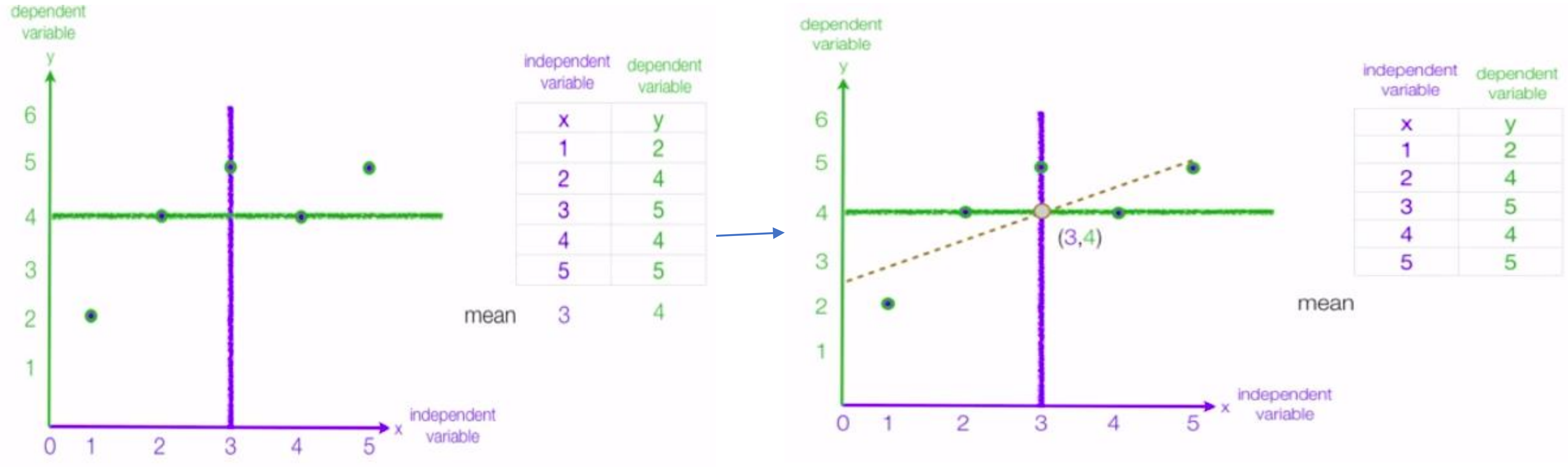


Image source: David Longstreet, Professor of the Universe, MyBookSucks, (Accessed, August 2018)

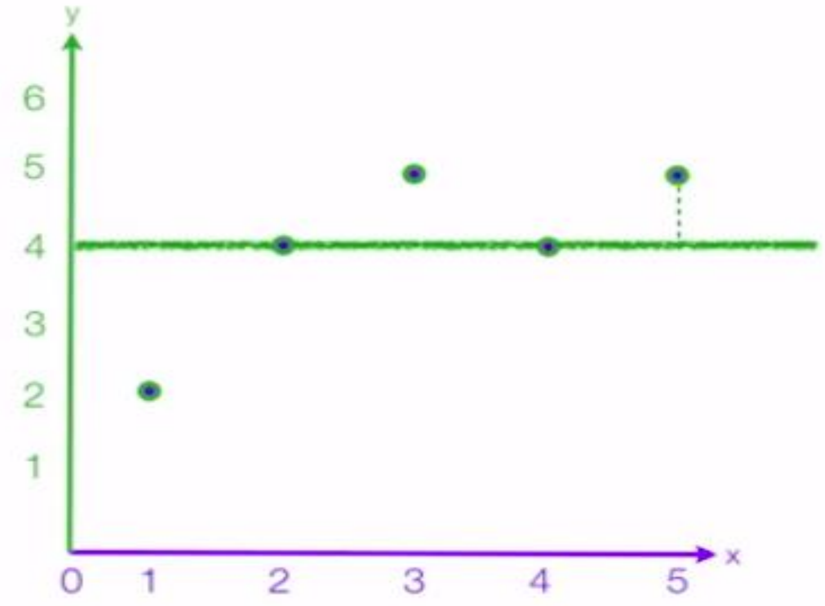
Regression using Least Square method



- Find $\bar{x}$ and $\bar{y}$, then plot and draw the line pass through the point
- All the possible regression line has to go through the point

Image source: David Longstreet, Professor of the Universe, MyBookSucks, (Accessed, August 2018)

Regression using Least Square method (cont.)

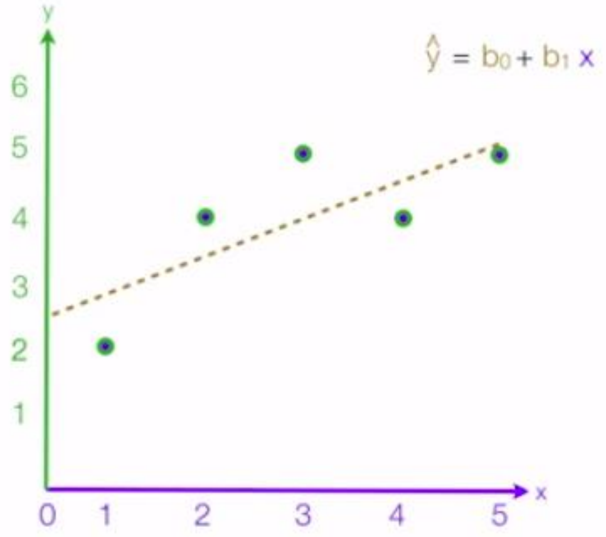


	x	y	$x - \bar{x}$	$y - \bar{y}$
	1	2	-2	-2
	2	4	-1	0
	3	5	0	1
	4	4	1	0
	5	5	2	
mean	3	4		

- Find the distance of observation points of x and y from the mean

Image source: David Longstreet, Professor of the Universe, MyBookSucks, (Accessed, August 2018)

Regression using Least Square method (cont.)



	x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$	$(x - \bar{x})(y - \bar{y})$
	1	2	-2	-2	4	4
	2	4	-1	0	1	0
	3	5	0	1	0	0
	4	4	1	0	1	0
	5	5	2	1	4	2
					10	6
mean	3	4				

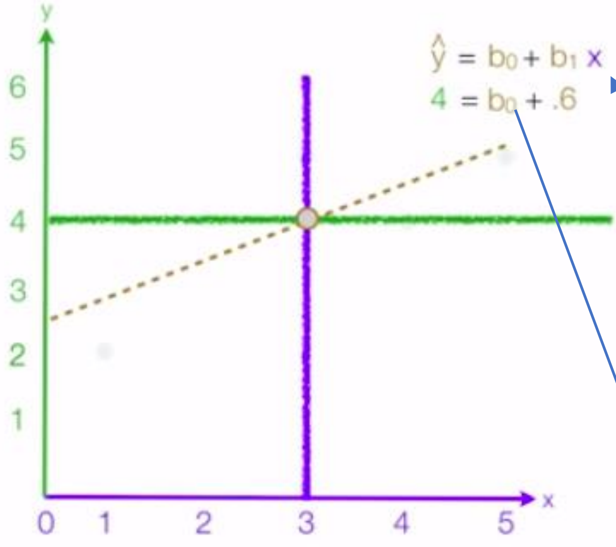
sum

$$b_1 = \frac{6}{10} = .6 = \frac{\Sigma(x - \bar{x})(y - \bar{y})}{\Sigma(x - \bar{x})^2}$$

- Now, we want to find the slope of the line by compute the square of $(x - \bar{x})$and $(x - \bar{x})(x - \bar{x})$,
- Hence, we get b_1 = 0.6

Image source: David Longstreet, Professor of the Universe, MyBookSucks, (Accessed, August 2018)

Regression using Least Square method (cont.)



x = 3

x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})^2$	$(x - \bar{x})(y - \bar{y})$
1	2	-2	-2	4	4
2	4	-1	0	1	0
3	5	0	1	0	0
4	4	1	0	1	0
5	5	2	1	4	2
				10	6
mean 3	4				

$$b_1 = \frac{6}{10} = .6 = \frac{\Sigma(x - \bar{x})(y - \bar{y})}{\Sigma(x - \bar{x})^2}$$

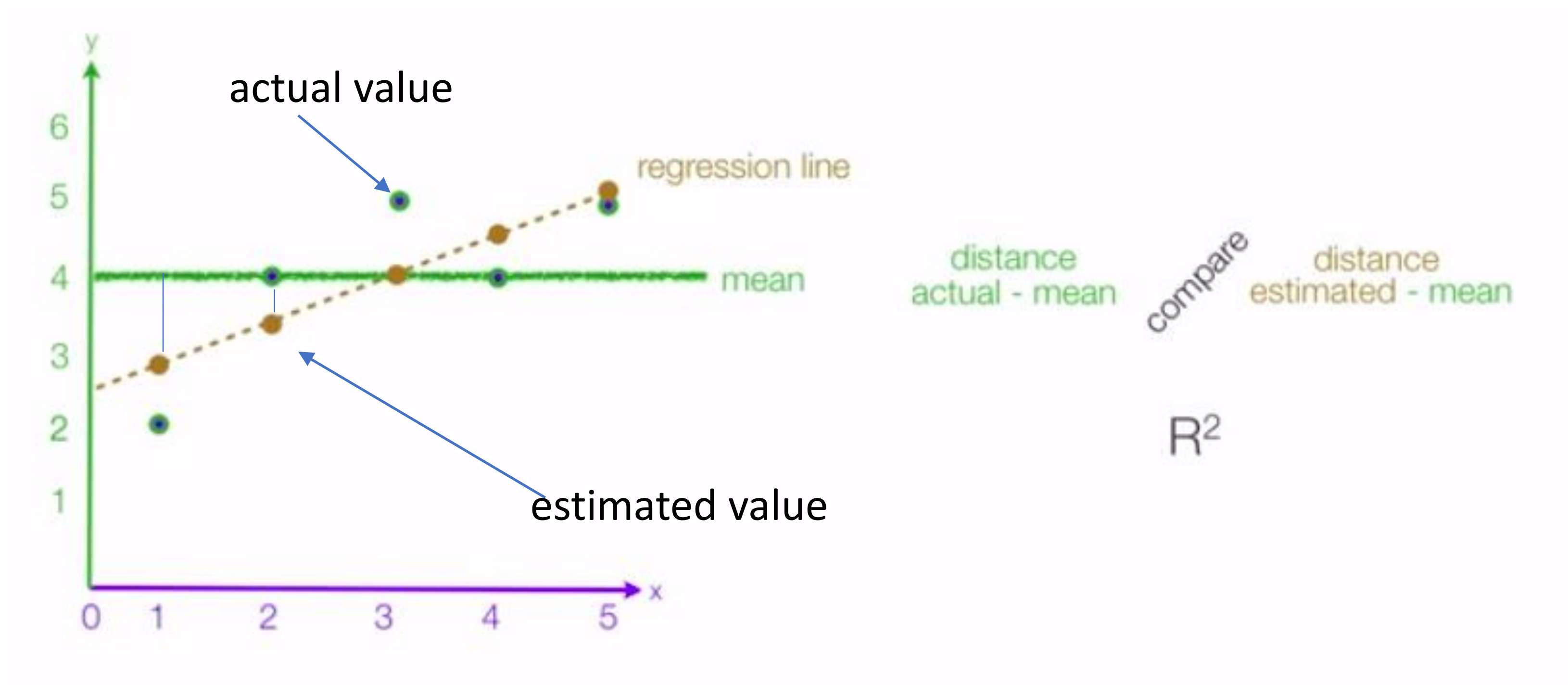
$b_0 = 2.2$

Now, we need to find the b_0

- We know that the regression line will always cross the p(3,4), so finding y-intercept is simple.
- Hence, we have $b_0 = 2.2$ and $b_1 = 0.6$
- $\bar{y}$= 2.2+0.6x

Image source: David Longstreet, Professor of the Universe, MyBookSucks, (Accessed, August 2018)

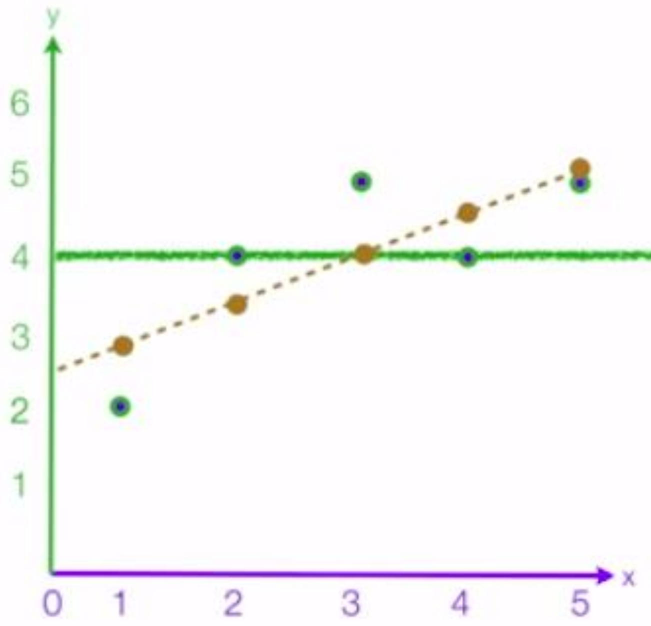
Evaluate regression model with R square



- R^2 = (R square)
- R^2 tells us how well our regression line can estimate predict value.

Image source: David Longstreet, Professor of the Universe, MyBookSucks, (Accessed, August 2018)

Evaluate regression model with R square



x	y	$y - \bar{y}$	$(y - \bar{y})^2$	$\hat{y}$	$\hat{y} - \bar{y}$	$(\hat{y} - \bar{y})^2$
1	2	-2	4	2.8	-1.2	1.44
2	4	0	0	3.4	-.6	.36
3	5	1	1	4	0	0
4	4	0	0	4.6	.6	.36
5	5	1	1	5.2	1.2	1.44
			6			3.6
mean	4					

$$R^2 = \frac{3.6}{6} = .6$$

R^2 ranges [0,1], the more closer 1 the better our regression line

Image source: David Longstreet, Professor of the Universe, MyBookSucks, (Accessed, August 2018)

Steps

- Find x^2 and xy
- Find slope
  - $\frac{\sum(x-\bar{x})(y-\bar{y})}{\sum(x-\bar{x})^2}$
- Get the y-intercept
- Create the equation

Multicollinearity

- two or more independent variables in a multiple regression model are highly correlated with each other.
- Lead to overfitting

Multicollinearity

- How to check for Multicollinearity
  - Use Variance Inflation Factor (VIF)

```
from statsmodels.stats.outliers_influence import
variance_inflation_factor
vif = pd.DataFrame()
vif["features"] = house_selected.columns
vif["vif_Factor"] =
[variance_inflation_factor(house_selected.values, i)
for i in range(house_selected.shape[1])]
vif
```

	features	vif_value
0	OverallQual	50.558204
1	YearBuilt	7015.226148
2	TotalBsmtSF	23.974354
3	1stFlrSF	700.064200
4	2ndFlrSF	141.592671
5	GrLivArea	1141.861313
6	PoolArea	1.046374
7	MoSold	6.529697
8	YrSold	6542.476374

Multicollinearity

- How to remedy?
  - Collect more data (Very expensive, but most effective)
  - Feature selection, transformation, PCA
  - Try ridge regression

Ridge regression

- Aka (L2 regularization), an alpha value to be tuned.
- Present regularization term which is not included in the linear regression
- Use when you have large number of feature and multicollinearity is happened
- Try with OLS first as based line and move to more complex like ridge

Ridge regression

- Loss function = OLS + alpha * summation (squared coefficient values)

$$\text{Ridge} = \frac{1}{N} \sum_{i=1}^{N} (y_i - (mx_i + z))^2 + \lambda \sum_{i=1}^{p} (mx_i + z)^2$$

Lasso regression

- Least Absolute Shrinkage Selector operator
- Aka. L1 regularization
- For the penalty term, take abs instead of square
- Can be used a feature selection

Discussion and Class activities

Sklearn for linear regresssion

```python
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Load the diabetes dataset
diabetes = datasets.load_diabetes()

# Use only one feature for simplicity
X = diabetes.data[:, np.newaxis, 2]

# Split the data into training/testing sets
X_train, X_test, y_train, y_test = train_test_split(X, diabetes.target, test_size=0.2, random_state=42)

# Create linear regression object
regr = LinearRegression()

# Train the model using the training set
regr.fit(X_train, y_train)

# Make predictions using the testing set
y_pred = regr.predict(X_test)
```

Sklearn for Ridge and how to tune

```python
ridge_regr = Ridge(alpha=1.0)

# Train the model using the training set
ridge_regr.fit(X_train, y_train)


# Split the data into training/testing sets
X_train, X_test, y_train, y_test = train_test_split(X, diabetes.target, test_size=0.2, random_state=42)

# Define a range of alpha values
alphas = np.logspace(-4, 4, 100)

# Create a RidgeCV regressor object
ridge_regr = RidgeCV(alphas=alphas, store_cv_values=True)

# Train the model using the training set
ridge_regr.fit(X_train, y_train)

# Get the best alpha value
best_alpha = ridge_regr.alpha_
print(f"Best alpha value: {best_alpha}")

# Make predictions using the testing set
y_pred_ridge = ridge_regr.predict(X_test)

# Print the coefficients
print('Coefficients: \n', ridge_regr.coef_)

# The mean squared error
print('Mean squared error: %.2f' % mean_squared_error(y_test, y_pred_ridge))

# The coefficient of determination: 1 is perfect prediction

```

Linear regression (by hand)

	Number of Chimpanzees	Percent Successful Hunts
0	1	30
1	2	45
2	3	51
3	4	57
4	5	60
5	6	65
6	7	70
7	8	71


CMU
CHIANG MAI
MARATHON
Run the legacy
60 years of CMU
NOV 26 , 2023
MARATHON - HALF MARATHON
MINI MARATHON - FUN RUN - VR RACE
เข้าสู่เว็บไซต์


CMU
CHIANG MAI
MARATHON
Run the legacy
60 years of CMU
NOV 26 , 2023
MARATHON - HALF MARATHON
MINI MARATHON - FUN RUN - VR RACE
RUNNING SHIRT
VR RACE
Front
Back
ติดต่อสอบถามได้ที่ CMU Marathon

KNN: Now's it your turn

- Let k = 3
- normalize the data in range.
- Plot a figure to support your result

Height (cms)	**Weight (kgs)**	**Size**
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	?

Term project

- Workshop :
- Please fill out the term project roaster under MSTeam

References

- Mahdavinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., & Sheth, A. P. (2018). Machine learning for internet of things data analysis: A survey. *Digital Communications and Networks*, *4*(3), 161–175. https://doi.org/10.1016/j.dcan.2017.10.002
- (*Data Science (The MIT Press Essential Knowledge Series): Kelleher, John D., Tierney, Brendan: 0000262535432: Amazon.Com: Books*, n.d.)
- Andreas C.Muller and Sarah Guido. 2017. Introduction to machine learning with pyhton
- Busse, C. D. (1978). Do Chimpanzees Hunt Cooperatively? *The American Naturalist*, *112*(986), 767–770. https://doi.org/10.1086/283318