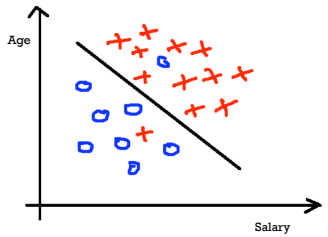
CHULA ENGINEERING
Foundation toward Innovation

COMPUTER



Logistic Regression

2110574: AI for Engineers

Peerapon Vateekul, Ph.D.

Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University
Peerapon.v@chula.ac.th

Outlines

- Introduction (Recap)
- Logistic Model (Model M1)
- Evaluation
- Non-numeric variables (Model M2)
- Feature Selection (Model M3) [optional]
- Demo

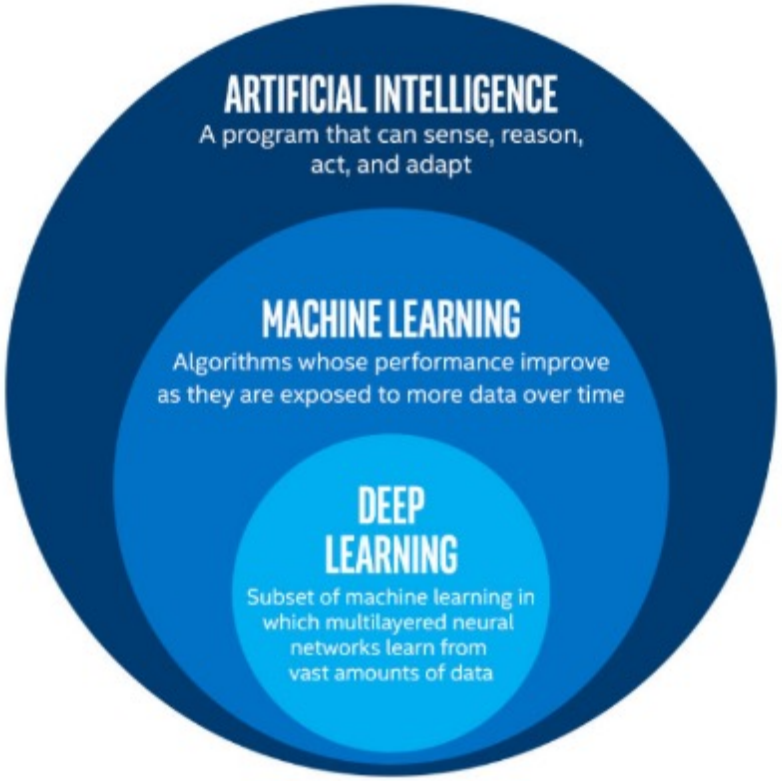
Introduction (Recap)

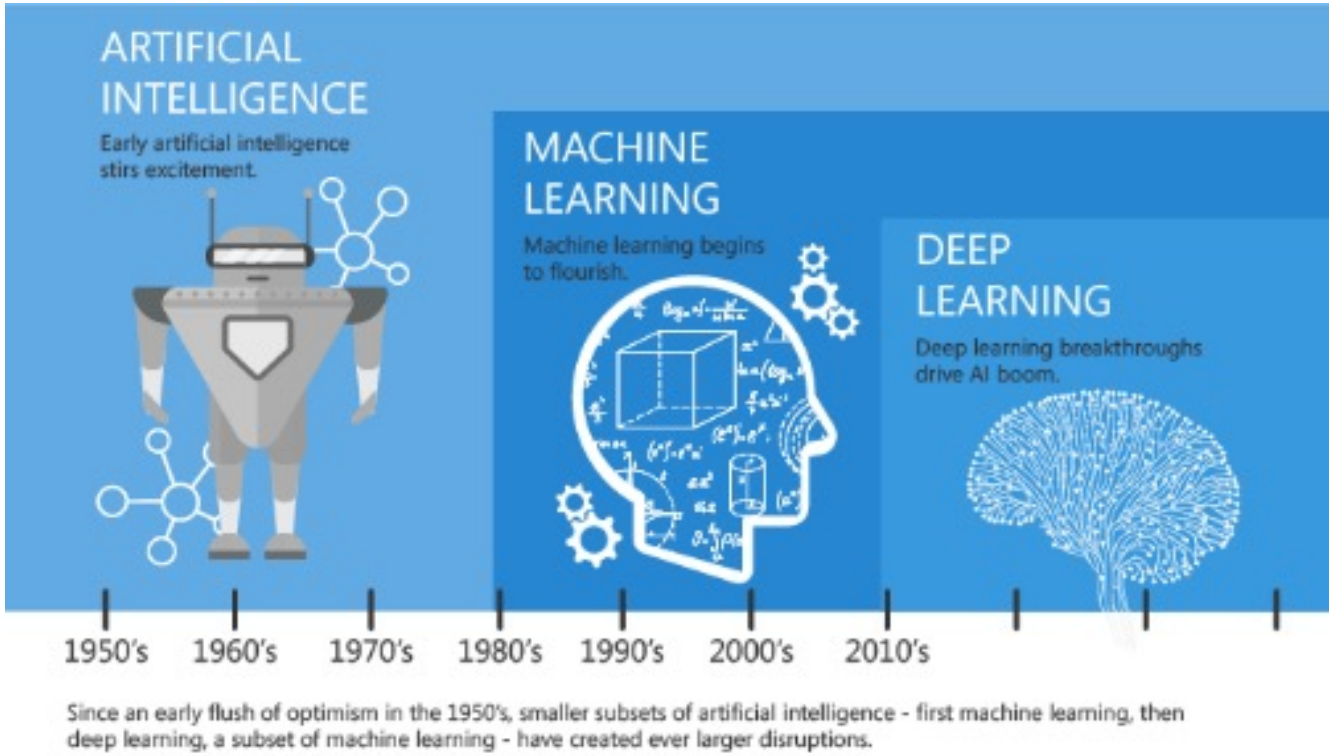
AI, Machine Learning, and Deep Learning (recap)

- Machine Learning (ML) is a subfield in AI focusing on making to learn by itself without human intervention.

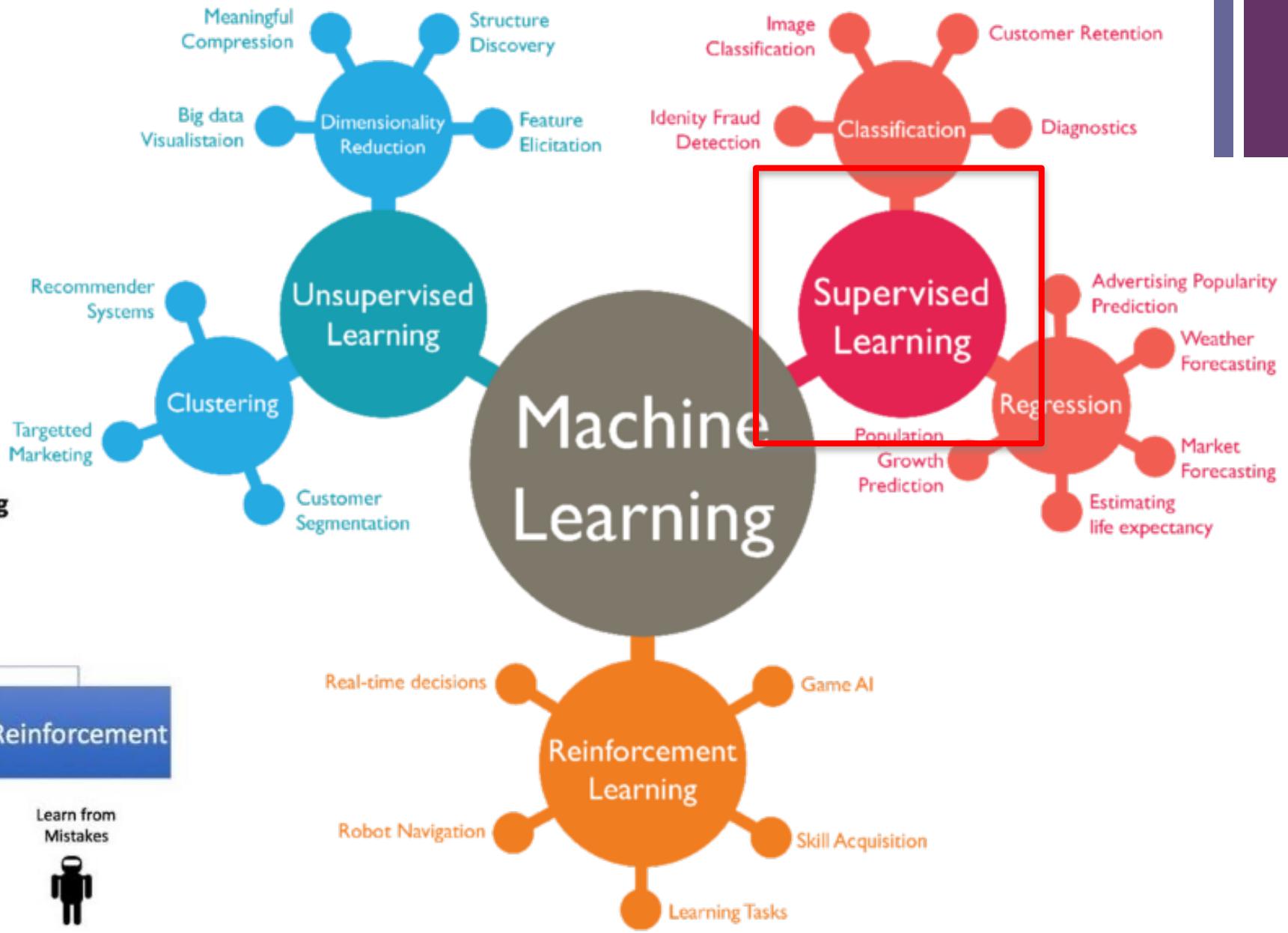
"Machine learning is the science *of getting computers to act without being explicitly programmed*." — *Stanford University*

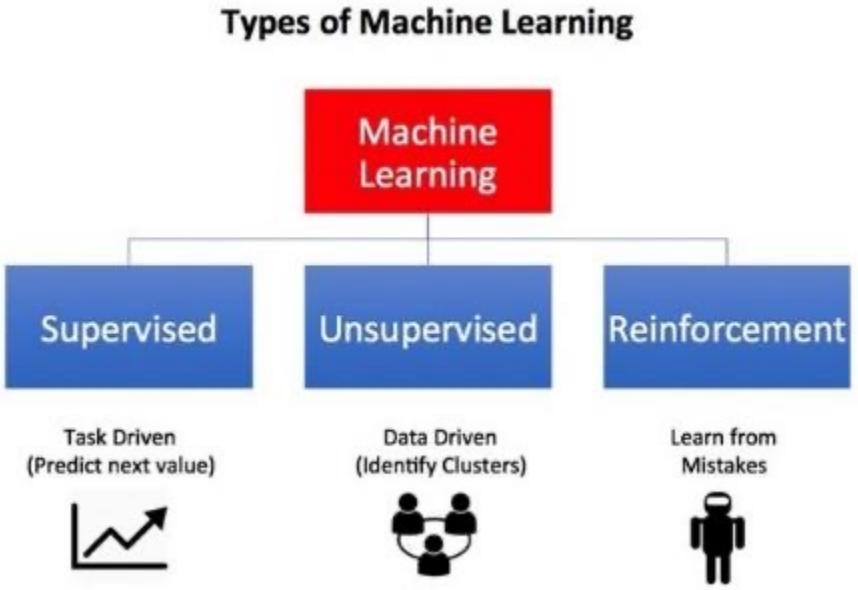
https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55



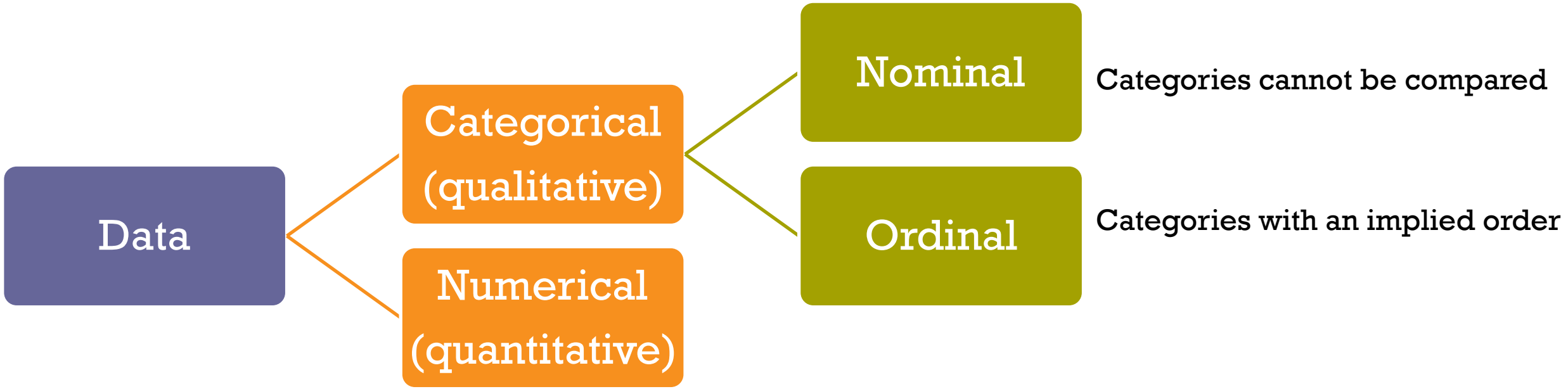


Machine Learning (recap)





Terminology: Kinds of data (recap)



Supervised learning (recap)

Training Data

	inputs			target
Age	**Income**	**Gender**	**Province**	**Purchase**
25	25,000	Female	Bangkok	Yes (1)
35	50,000	Female	Nontaburi	Yes (1)
32	35,000	Male	Bangkok	No (0)

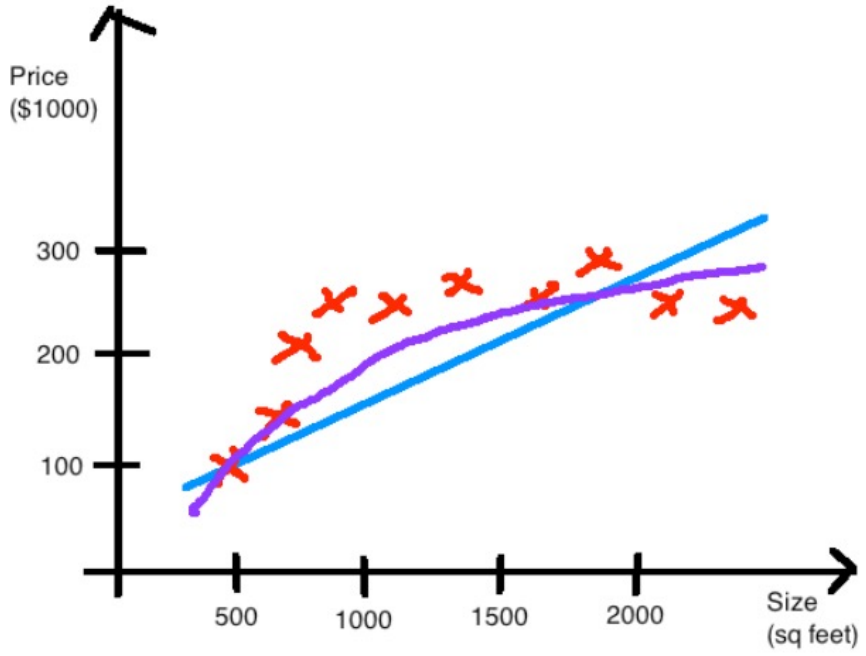
Testing Data

P	**Income**	**Gender**	**Province**	**Purchase**
25	25,000	Female	Bangkok	?

Application: Direct Target Customer

Regression: predict a continuous value

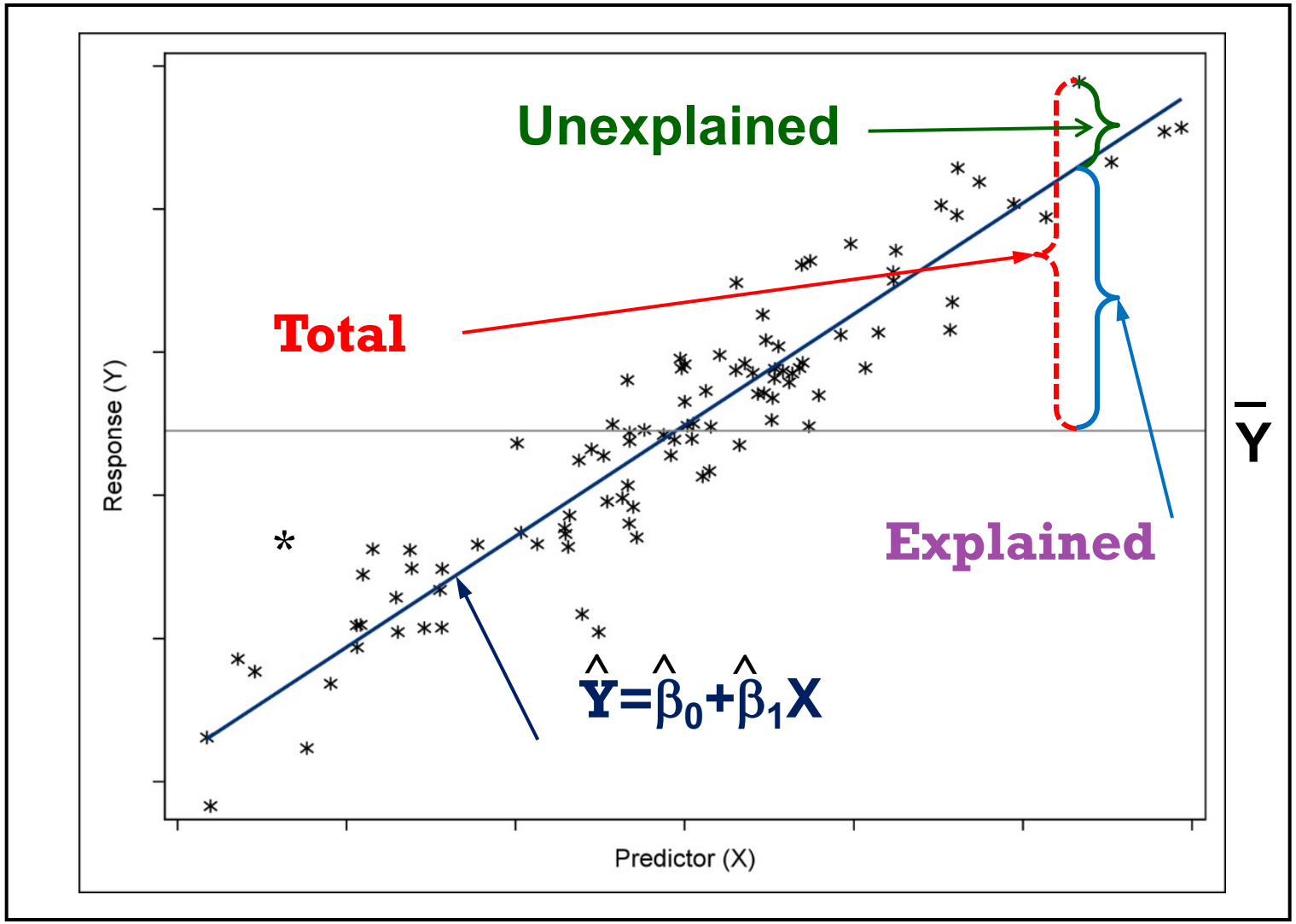
Linear Regression



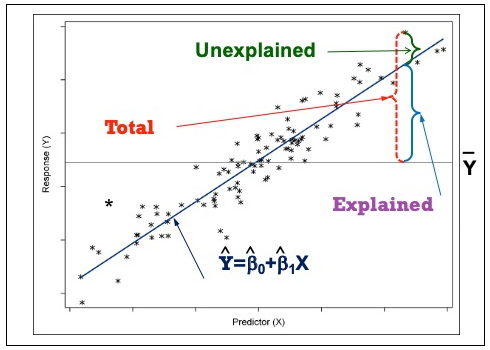
Predict a sale price of each house

- **Some techniques:**
  - Linear Regression / GLM
  - Decision Trees
  - Support vector regression
  - Neural Network
  - Ensembles
- **Sample Applications**
  - Financial risk management
  - Revenue forecasting

OLS: Explained versus Unexplained Variability



Coefficient of Determination: R^2



$$R^2 = \frac{SSM}{SST} = 1 - \frac{SSE}{SST}$$

id	chol (x)	bp (y)	predict	error	squred error (SE)	guess	(y - y_bar)	squred total (ST)
1	437	194	196.1897	(2.1897)	4.7948	143.4286	50.5714	2,557.4694
2	264	121	141.4179	(20.4179)	416.8906	143.4286	(22.4286)	503.0408
3	249	131	136.6689	(5.6689)	32.1364	143.4286	(12.4286)	154.4694
4	297	159	151.8657	7.1343	50.8982	143.4286	15.5714	242.4694
5	243	123	134.7693	(11.7693)	138.5164	143.4286	(20.4286)	417.3265
6	272	161	143.9507	17.0493	290.6786	143.4286	17.5714	308.7551
7	161	115	108.8081	6.1919	38.3396	143.4286	(28.4286)	808.1837
average	274.7143	143.4286		SSE	972.2548		SST	4,991.7143
				MSE	138.8935			
				RMSE	11.7853			
	R^2	1 - (SSE/SST)	0.8052					

Adjusted R^2

$$R^2 = \frac{SSM}{SST} = 1 - \frac{SSE}{SST}$$

$$R^2_{ADJ} = 1 - \frac{(n-i)(1-R^2)}{n-p}$$

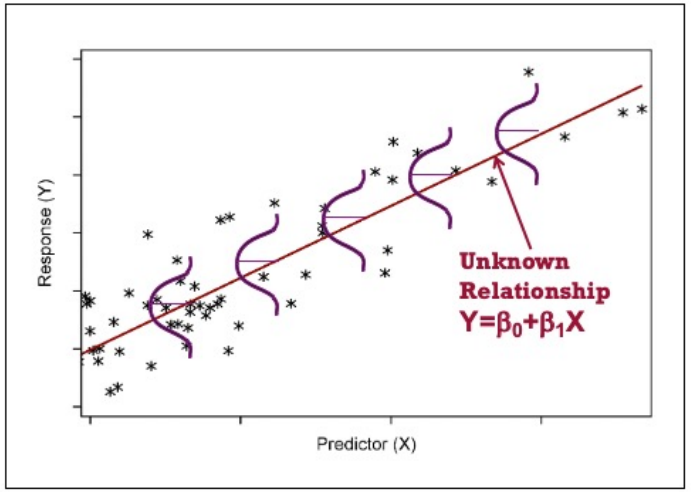
- i=1 if there is an intercept and 0 otherwise
- n=the number of observations used to fit the model
- p=the number of parameters in the model

Assumptions for Linear Regression

- The mean of the Ys is accurately modeled by a linear function of the X_i.
  - (y, x_i) = linear relationship (correlation)

- The random error term, ε, is assumed to have a normal distribution with a mean of zero.
- The random error term, ε, is assumed to have a constant variance, σ^2.
  - Not skew

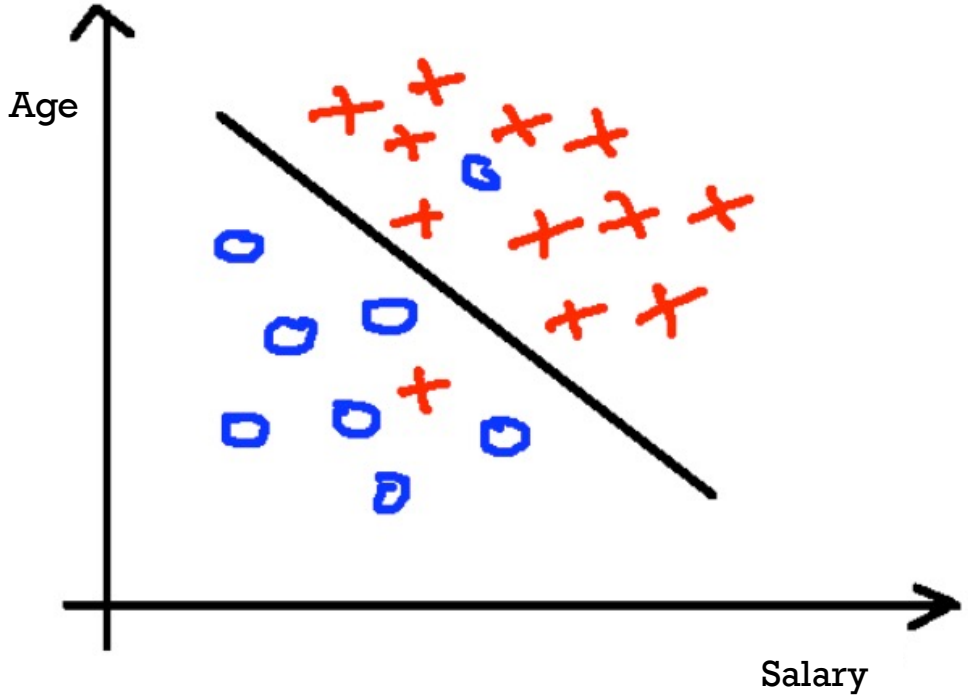
- The errors are independent.



Logistic Regression (Model M1)

Classification: predicting a category

Logistic Regression



Predict targeted customers who tend to buy our product (yes/no)

- **Some techniques:**
  - Naïve Bayes
  - Decision Tree
  - Logistic Regression
  - Support Vector Machines
  - Neural Network
  - Ensembles
- **Sample Applications**
  - Database marketing
  - Fraud detection
  - Pattern detection
  - Churn customer detection

Regression Task:
Linear Regression Prediction Formula

input measurement

$$\hat{y} = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

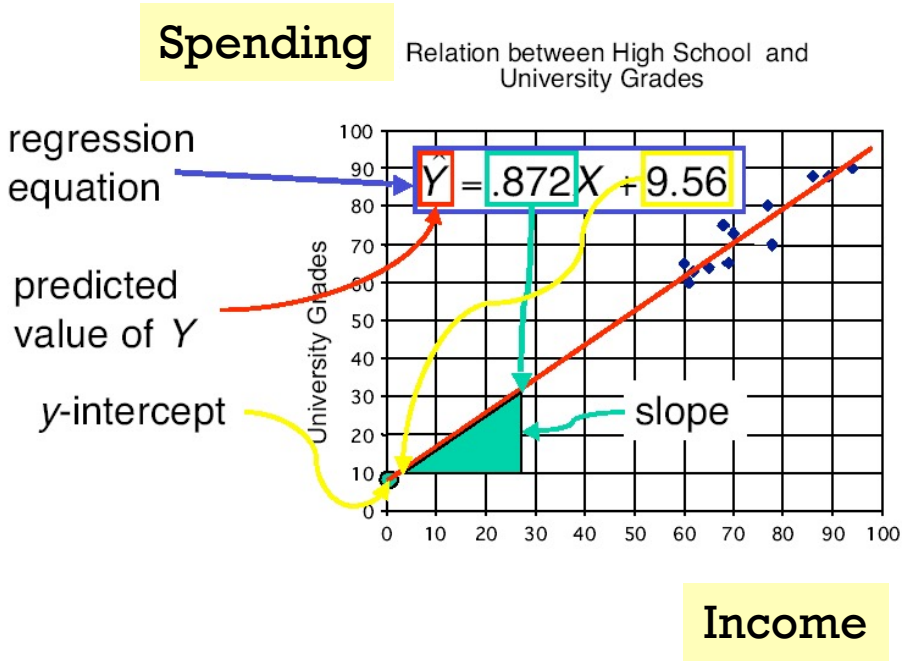
prediction estimate

intercept estimate *parameter estimate*

Choose intercept and parameter estimates to *minimize*.

squared error function

$$\sum_{\text{training data}} (y_i - \hat{y}_i)^2$$

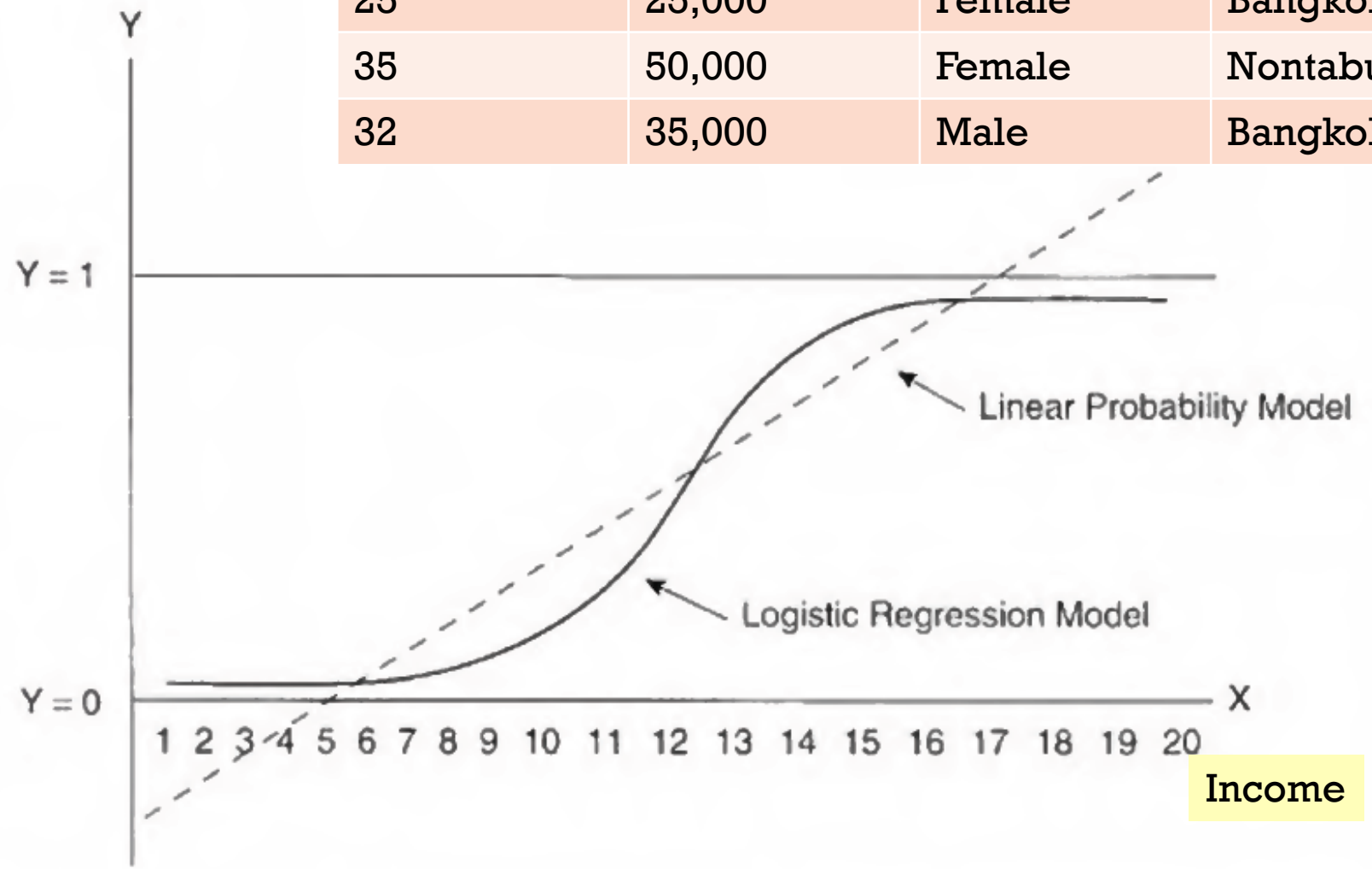


...

Classification Task

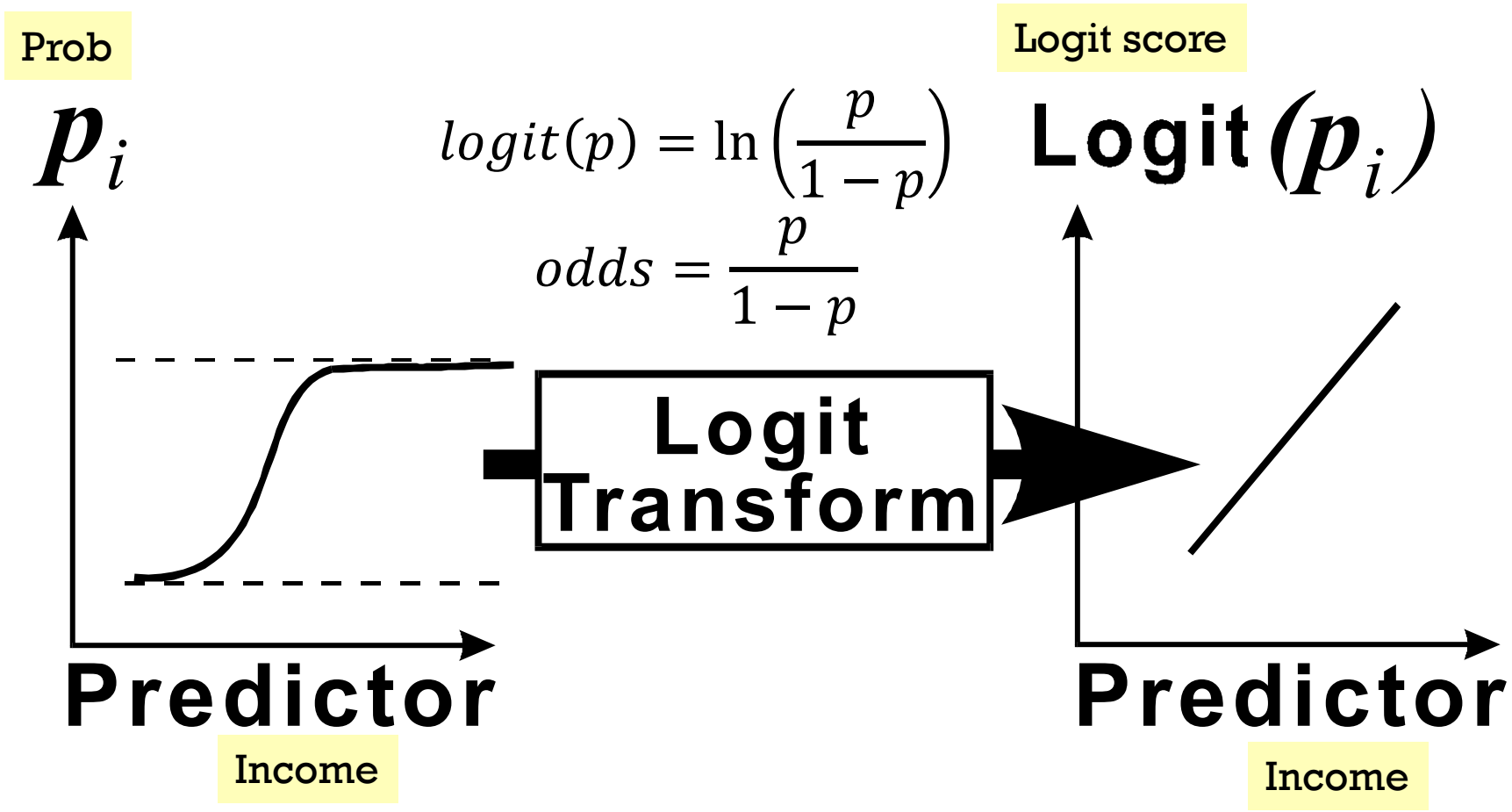
Prob. of purchase

Age	Income	Gender	Province	Purchase
25	25,000	Female	Bangkok	Yes (1)
35	50,000	Female	Nontaburi	Yes (1)
32	35,000	Male	Bangkok	No (0)



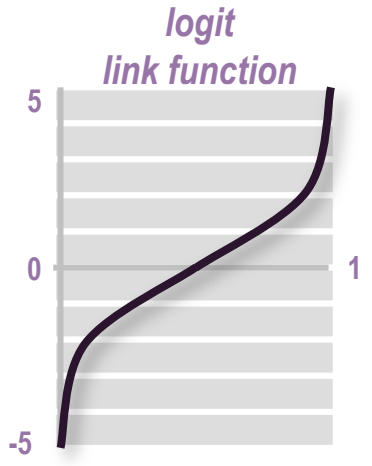
Logistic function
Sigmoid function

Logistic Regression
Logit link function: Non-linear to Linear

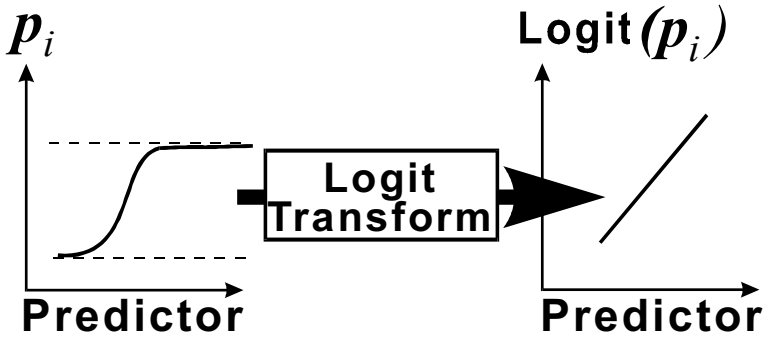


Logistic Regression Prediction Formula

$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$ *logit scores*



The logit link function transforms probabilities (between 0 and 1) to logit scores (between −∞ and +∞).



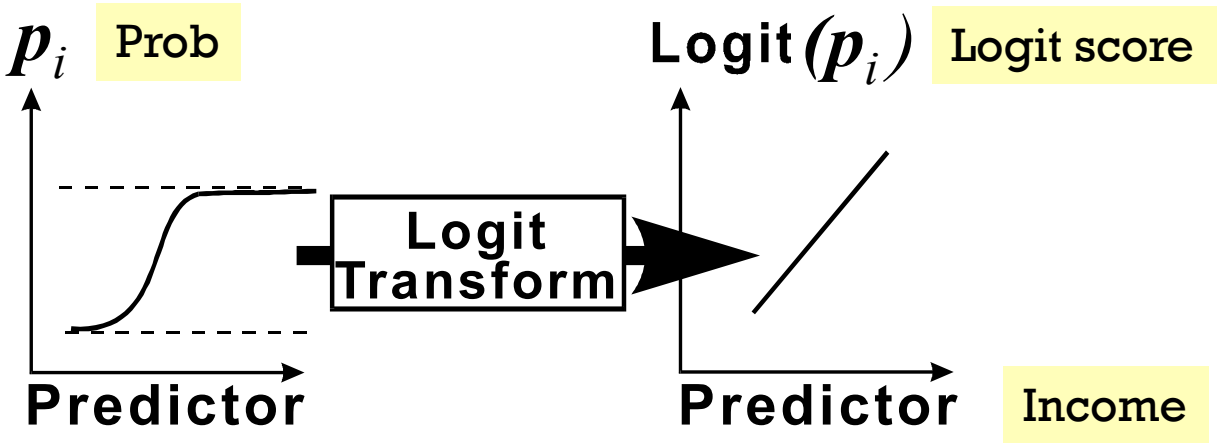
...

Logit Link Function

$$\log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2 = \text{logit}(\hat{p})$$

$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

To obtain prediction estimates, the logit equation is solved for $\hat{p}$.



...

Training Phase: Regressions
Maximum Log-Likelihood Estimation (MLE)

$$\text{logit}(\hat{p}) = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

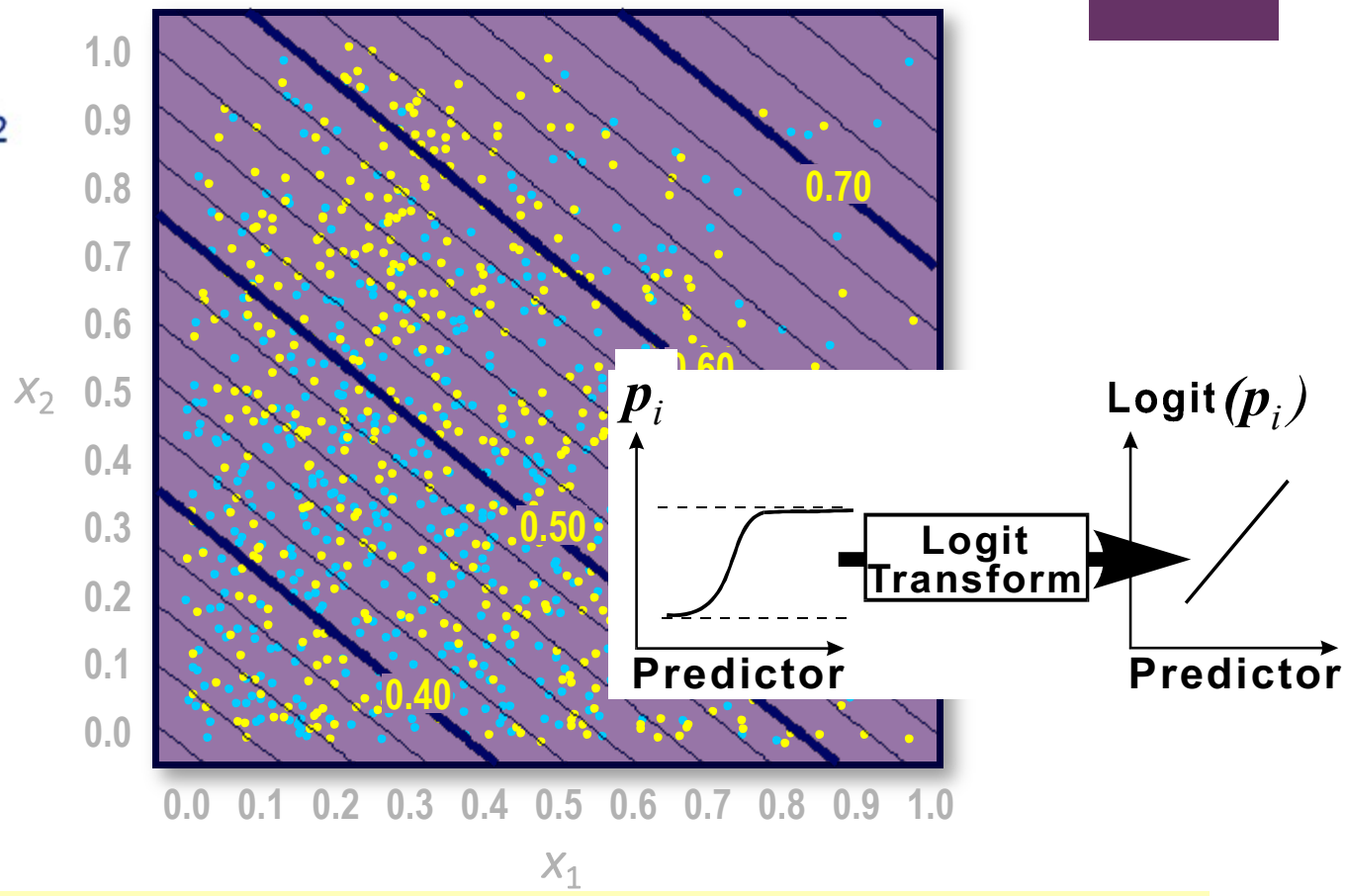
$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

Find parameter estimates by *maximizing*

$$\sum \log(\hat{p}_i) + \sum \log(1 - \hat{p}_i)$$

primary outcome training cases — *secondary outcome training cases*

log-likelihood function



For each set of parameters (w0, w1, w2), we can calculate log-likelihood (LL) of the whole training data.
So, pick the set of parameters with a maximum LL (MLE).

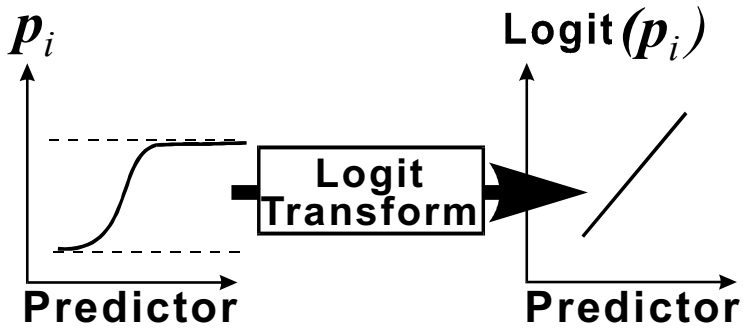
...

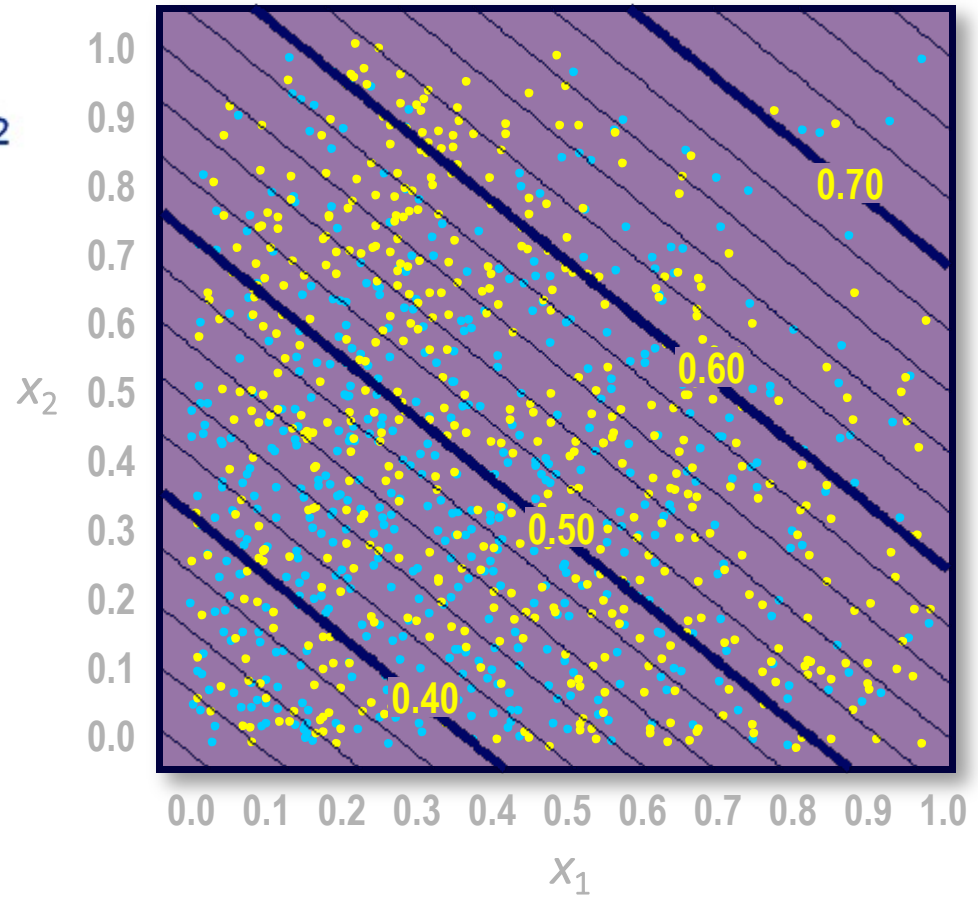
Testing Phase: Regressions

$$\text{logit}(\hat{p}) = -0.81 + 0.92\,x_1 + 1.11\,x_2$$

$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

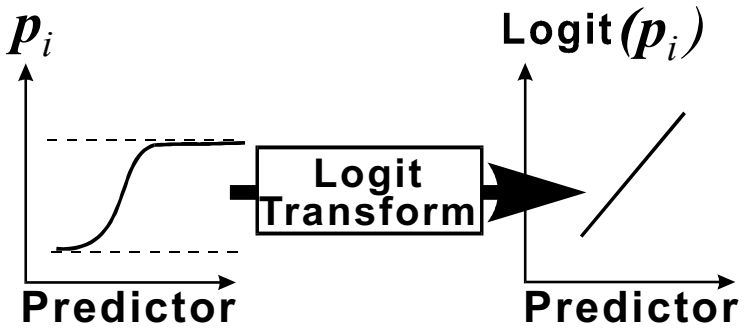
Using the maximum likelihood estimates, the prediction formula assigns a logit score to each x_1 and x_2.





...

Quiz



- What is the logistic regression prediction for the indicated point?

a. 0.243

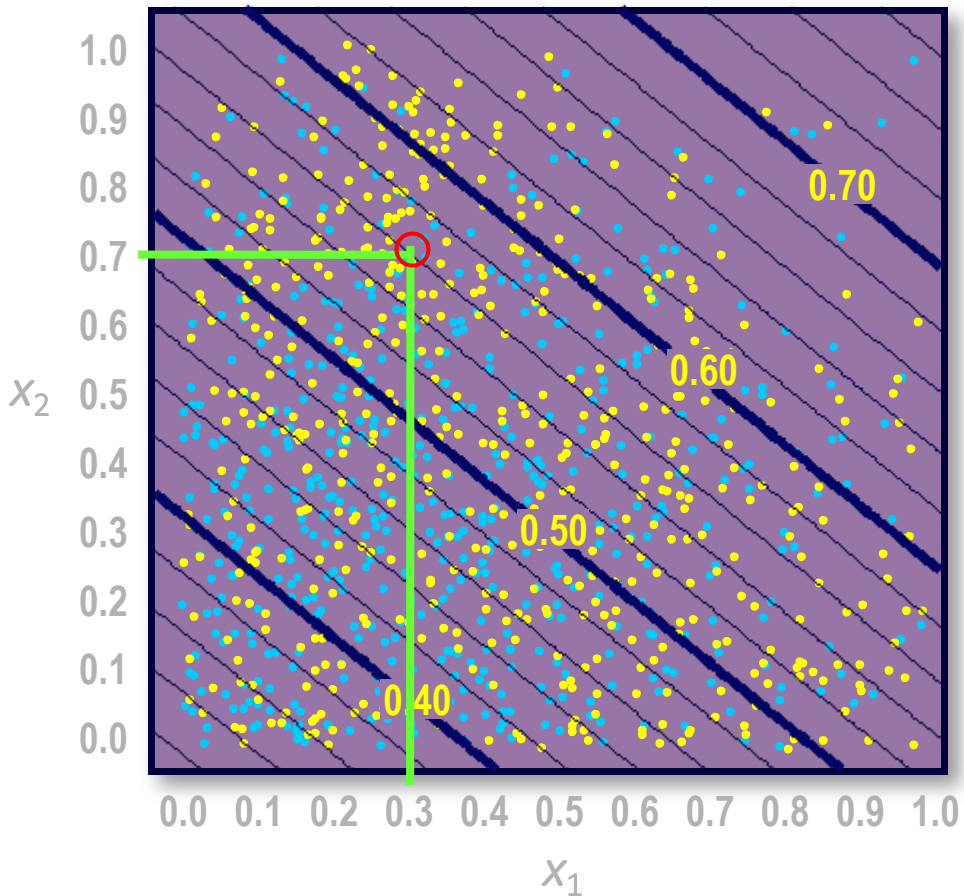
b. 0.56

c. yellow

d. It depends.

$$\text{logit}(\hat{p}) = -0.81 + 0.92\,x_1 + 1.11\,x_2$$

$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$



Training Phase: Regressions (Recap)
Maximum Log-Likelihood Estimation (MLE)

$$\text{logit}(\hat{p}) = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

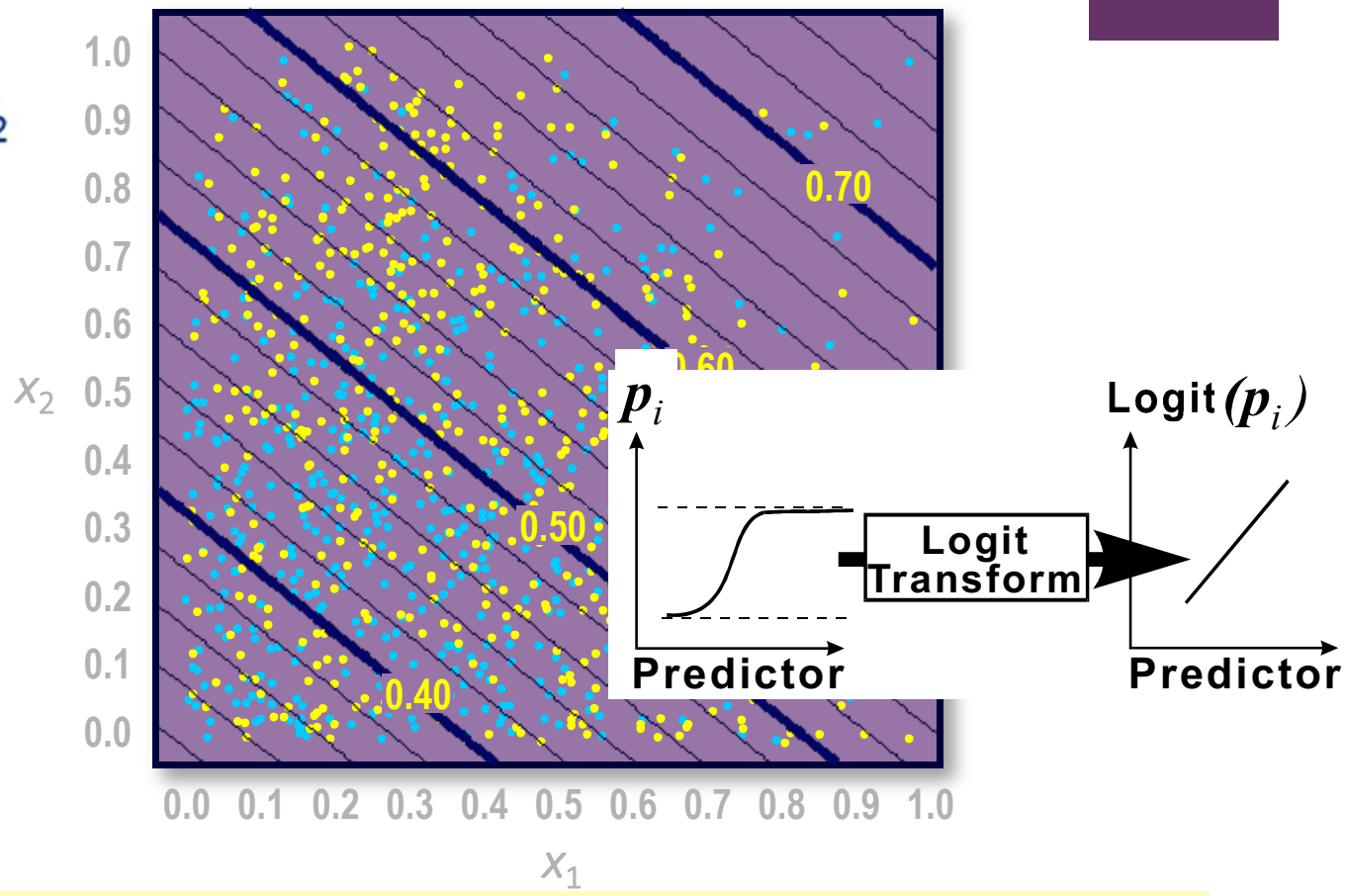
$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

Find parameter estimates by *maximizing*

$$\sum \log(\hat{p}_i) + \sum \log(1 - \hat{p}_i)$$

primary outcome training cases — *secondary outcome training cases*

log-likelihood function



For each set of parameters (w0, w1, w2), we can calculate log-likelihood (LL) of the whole training data.
So, pick the set of parameters with a maximum LL (MLE).

...

$$P(Y = y) = p^y(1-p)^{1-y}, y = 0,1$$

$$P(Y = 1) = p^1(1-p)^0 = p$$

$$P(Y = 0) = p^0(1-p)^{1-0} = (1-p)$$

$$Likelihood = \prod_{i=1}^{n} p^{y_i}(1-p)^{1-y_i}$$

$$Log - Likelihood\ (LL) = \sum_{i=1}^{n} y_i \ln(p_i) + (1-y_i) ln(1-p_i)$$

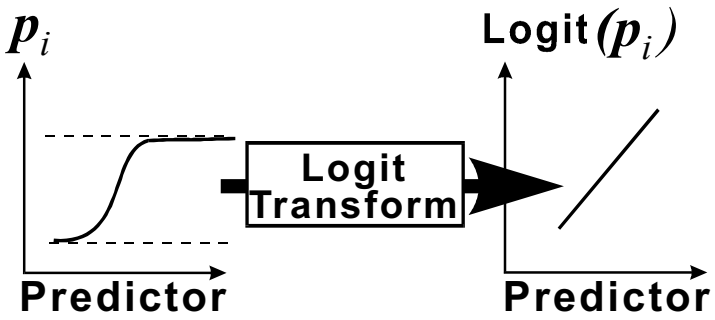
Outcome	Predicted Probability	
y_i	p_i	$(1-p_i)$
1	.899	.101
1	.540	.460
0	.317	.683
1	.516	.439
1	.698	.302
0	.457	.543
0	.234	.766
1	.899	.101
1	.451	.439
1	.764	.236
0	.457	.543
0	.561	.439
1	.698	.302
1	.457	.543
0	.899	.101

[a]–2 * log-likelihood = 17.48.

$$\text{logit}(\hat{p}) = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

$$\text{logit}(\hat{p}) = -0.81 + 0.92 x_1 + 1.11 x_2$$

$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

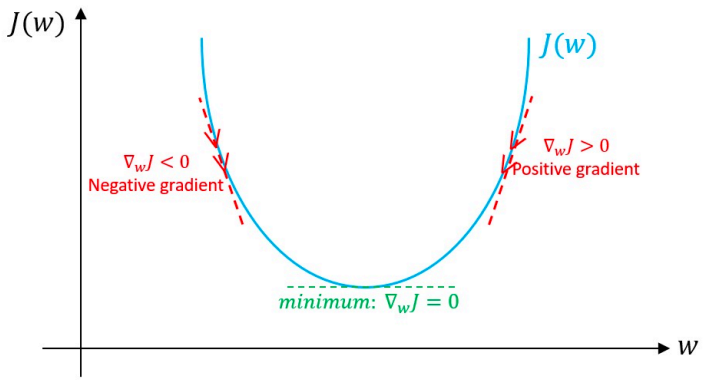


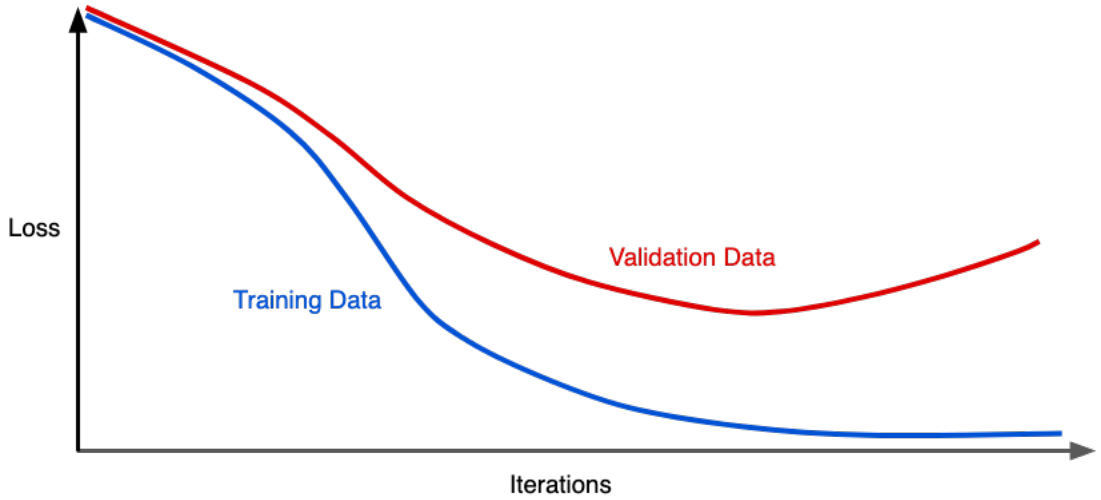
MLE Solver

$$\text{logit}(\hat{p}) = \hat{w}_0 + \hat{w}_1 x_1 + \hat{w}_2 x_2$$

$$\text{logit}(\hat{p}) = -0.81 + 0.92 x_1 + 1.11 x_2$$

- Maximum Log-Likelihood Estimation (MLE) refers to maximize LL
- Or minimize -2LL (loss)
- Gradient descent optimization
  - Stochastic Average Gradient solver





Prev Up Next

scikit-learn 0.22.1
Other versions

Please cite us if you use the software.

sklearn.linear_model.LogisticRegression
Examples using sklearn.linear_model.LogisticRe

solver : *{'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default='lbfgs'*

Algorithm to use in the optimization problem.

- For small datasets, 'liblinear' is a good choice, whereas 'sag' and 'saga' are faster for large ones.
- For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs' handle multinomial loss; 'liblinear' is limited to one-versus-rest schemes.
- 'newton-cg', 'lbfgs', 'sag' and 'saga' handle L2 or no penalty
- 'liblinear' and 'saga' also handle L1 penalty
- 'saga' also supports 'elasticnet' penalty
- 'liblinear' does not support setting `penalty='none'`

Note that 'sag' and 'saga' fast convergence is only guaranteed on features with approximately the same scale. You can preprocess the data with a scaler from sklearn.preprocessing.

New in version 0.17: Stochastic Average Gradient descent solver.

New in version 0.19: SAGA solver.

Changed in version 0.22: The default solver changed from 'liblinear' to 'lbfgs' in 0.22.

max_iter : *int, default=100*

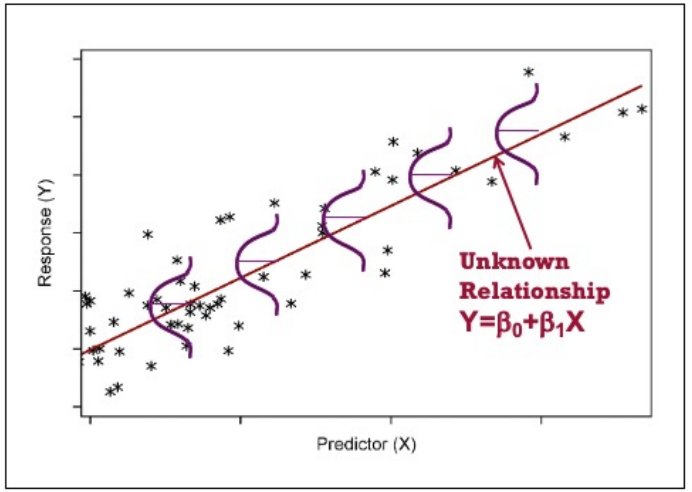
Maximum number of iterations taken for the solvers to converge.

Assumptions for Linear Regression (recap)

- The mean of the Ys is accurately modeled by a linear function of the X_i.
  - (y, x_i) = linear relationship (correlation)

- The random error term, ε, is assumed to have a normal distribution with a mean of zero.
- The random error term, ε, is assumed to have a constant variance, σ^2.
  - Not skew

- The errors are independent.

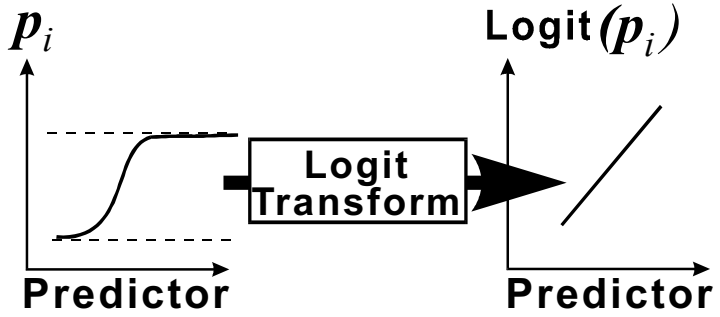


Assumptions for Logistic Regression

- The mean of the logit is accurately modeled by a linear function of the X_i.
  - (logit, x_i) = linear relationship

- ~~The random error term, ε, is assumed to have a normal distribution with a mean of zero.~~
- ~~The random error term, ε, is assumed to have a constant variance, σ^2.~~
  - ~~Not skew~~

- The errors are independent.



Beyond the prediction

1. Manage missing values
2. Interpret the model
3. Handle outliers
4. Use nonnumeric inputs

1) Missing Values and the Prediction Formula

$$\text{logit}(\hat{p}) = -0.81 + 0.92 \cdot x_1 + 1.11 \cdot x_2$$

Predict: (x1, x2) = (0.3, ?)

Problem 2: Prediction formulas cannot score cases with missing values.

Missing Values and the Prediction Formula

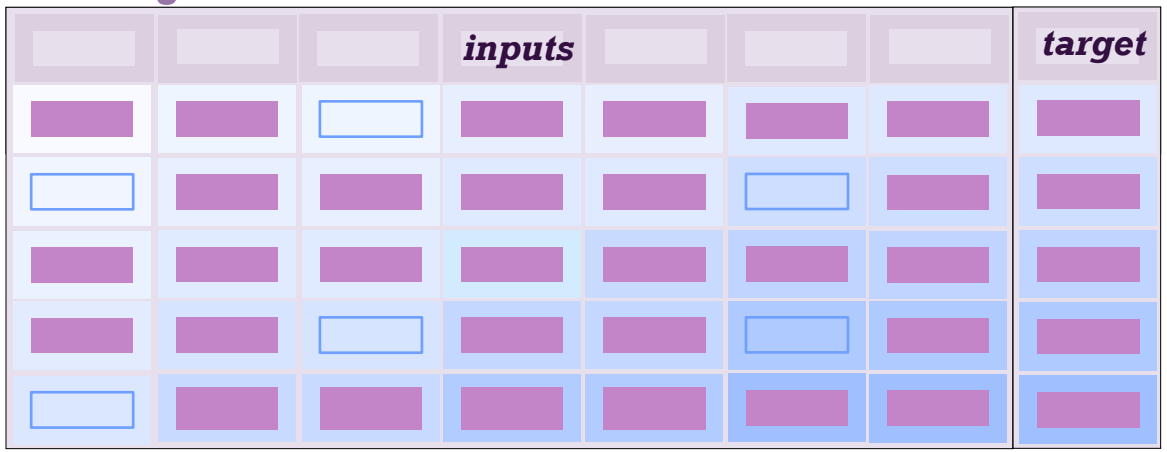
$$\text{logit}(\hat{p}) = -0.81 + 0.92 \cdot 0.3 + 1.11 \cdot ?$$

Predict: (x1, x2) = (0.3, ?)

Problem 2: **Prediction formulas cannot score cases with missing values.**

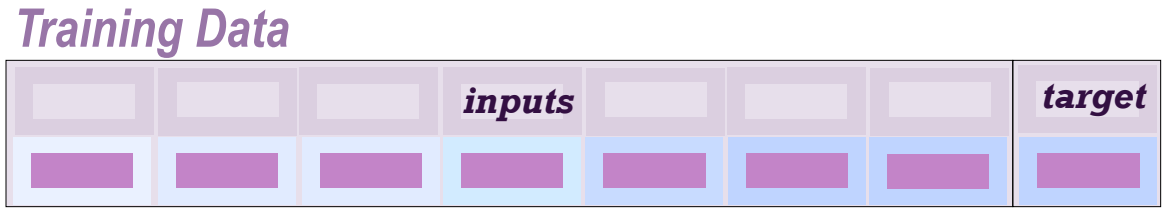
Missing Values and Regression Modeling

Training Data



Problem 1: Training data cases with missing values on inputs used by a regression model are ignored.

Missing Values and Regression Modeling



Consequence: Missing values can significantly reduce your amount of training data for regression modeling!

2) Interpret the model: Positive effect (income)

Recap how to interpret "regression"

$$logit(p) = 500 + 0.2 \times Income - 0.8 \times Age$$

$$\ln(odds) = 500 + 0.2 \times Income - 0.8 \times Age$$

$$\ln(odds(Income = 1)) = 0.2 \times 1 = 0.2$$

$$odds(Income = 1) = e^{0.2} = 1.22$$

$$\ln(odds(Income = 0)) = 0.2 \times 0 = 0$$

$$odds(Income = 0) = e^{0} = 1$$

$$Odss\ Ratio(Income) = \frac{odds(Income = 1)}{odds(Income = 0)} = \frac{1.22}{1} = 1.22\ (+22\%) = \exp(w_1)$$

- OR(Income) = 1.22 (22% increase)
- When income increases by 1THB, it has higher chance (odds) to purchase for 22%

Interpret the model: Negative effect (age)

Recap how to interpret "regression"

$$logit(p) = 500 + 0.2 \times Income - 0.8 \times Age$$

$$\ln(odds) = 500 + 0.2 \times Income - 0.8 \times Age$$

$$\ln(odds(Age = 1)) = -0.8 \times 1 = -0.8$$

$$odds(Age = 1) = e^{-0.8} = 0.45$$

$$\ln(odds(Age = 0)) = -0.8 \times 0 = 0$$

$$odds(Age = 0) = e^{0} = 1$$

$$Odss\ Ratio(Age) = \frac{odds(Age = 1)}{odds(Age = 0)} = \frac{0.45}{1} = 0.45\ (-55\%) = \exp(w_1)$$

- OR(Age) = 0.45 (55% decrease)
- When age increases by 1 year, it has lower chance (odds) to purchase for 55%.

sklearn.linear_model.LogisticRegression

class `sklearn.linear_model.` **`LogisticRegression`**(*penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*) ¶ [source]

Logistic Regression (aka logit, MaxEnt) classifier.

In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs', 'sag', 'saga' and 'newton-cg' solvers.)

This class implements regularized logistic regression using **regularization is applied by default**. It can handle both c 64-bit floats for optimal performance; any other input forn

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 solver supports both L1 and L2 regularization, with a dual supported by the 'saga' solver.

Read more in the User Guide.

Examples

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97...
```

Parameters:

penalty : *{'l1', 'l2', 'elasticnet', 'none'}*, default='l2'

Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only l2 penalties. 'elasticnet' is only supported by the 'saga' solver. If 'none' (not supported by the liblinear solver), no regularization is applied.

New in version 0.19: l1 penalty with SAGA solver (allowing 'multinomial' + L1)

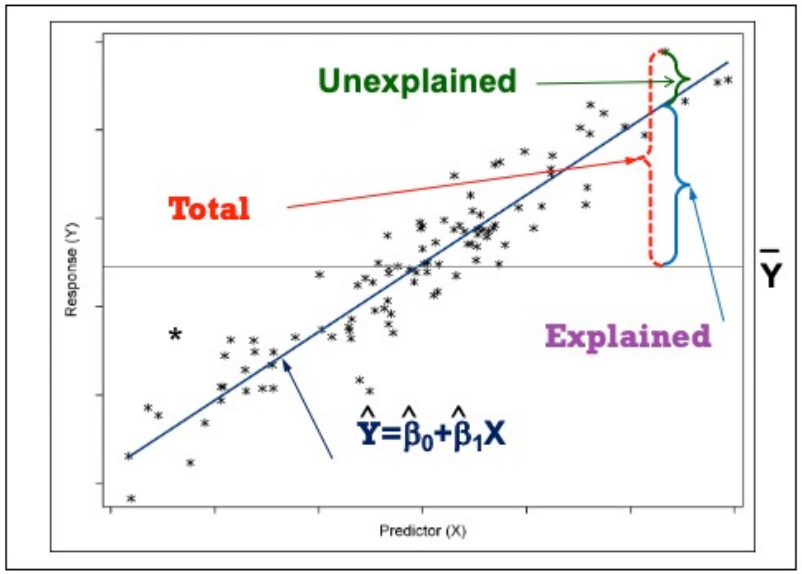
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Evaluation

Pseudo R^2

TP, TN, FP, FN

Precision, Recall, F1



R^2 (recap)

$$R^2 = \frac{SSM}{SST} = 1 - \frac{SSE}{SST}$$

id	chol (x)	bp (y)	predict	error	squred error (SE)	guess	(y - y_bar)	squred total (ST)
1	437	194	196.1897	(2.1897)	4.7948	143.4286	50.5714	2,557.4694
2	264	121	141.4179	(20.4179)	416.8906	143.4286	(22.4286)	503.0408
3	249	131	136.6689	(5.6689)	32.1364	143.4286	(12.4286)	154.4694
4	297	159	151.8657	7.1343	50.8982	143.4286	15.5714	242.4694
5	243	123	134.7693	(11.7693)	138.5164	143.4286	(20.4286)	417.3265
6	272	161	143.9507	17.0493	290.6786	143.4286	17.5714	308.7551
7	161	115	108.8081	6.1919	38.3396	143.4286	(28.4286)	808.1837
average	274.7143	143.4286		SSE	972.2548		SST	4,991.7143
				MSE	138.8935			
				RMSE	11.7853			
	R^2	1 - (SSE/SST)	0.8052					

Pseudo R^2

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

Outcome	Predicted Probability	
Y	$\hat{Y}$	$1 - \hat{Y}$
1	.899	.101
1	.540	.460
0	.317	.683
1	.516	.439
1	.698	.302
0	.457	.543
0	.234	.766
1	.899	.101
1	.451	.439
1	.764	.236
0	.457	.543
0	.561	.439
1	.698	.302
1	.457	.543
0	.899	.101

$\bar{y} = 0.6$

[a]–2 * log-likelihood = 17.48.

Confusion Matrix

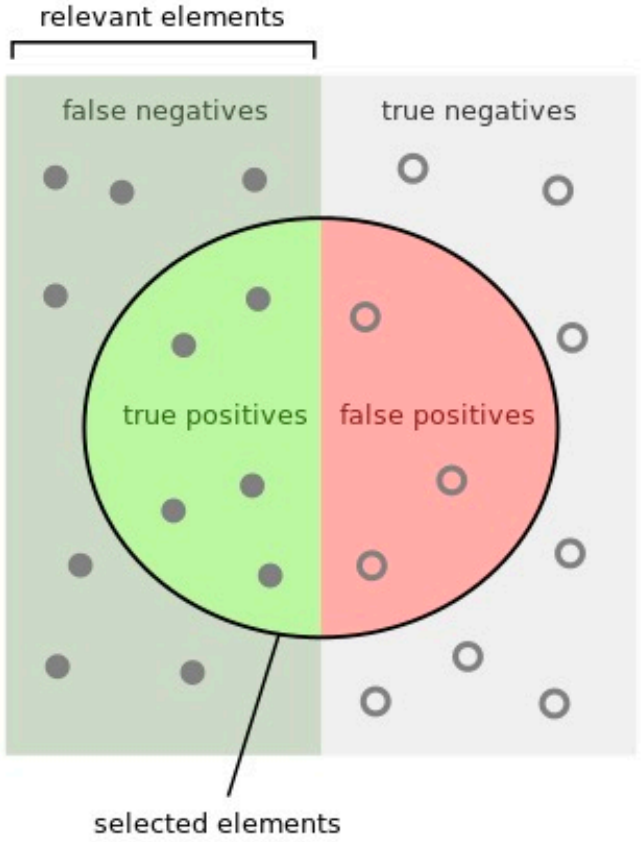
n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

- True Positive (TP)
- True Negative (TN)
- False Positive (FP)
- False Negative (FN)

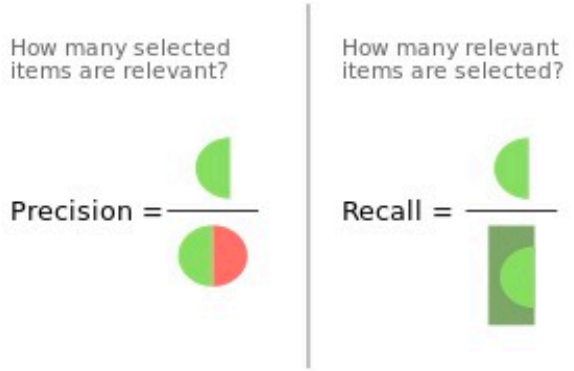
- Accuracy = (TP + TN) / total
- Misclassification = (FP + FN) / total

Precision, Recall, F1

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	



- Precision = correctly predict = TP / (TP + FP)
- Recall = coverage = TP / (TP + FN)
- F1 = (2*pre*rec) / (pre + rec)



Precision, Recall, F1: Average

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

- Precision = correctly predict = TP / (TP + FP)
- Recall = coverage = TP / (TP + FN)
- F1 = (2*pre*rec) / (pre + rec)

- What is a positive class?
  - 1) Direct target marketing? Yes
  - 2) Intelligent diagnosis to predict "Corona" or "Flu" – both are important.
- If there is no positive (all classes are important)
  - Macro Average
  - Weighted Average (Micro Average) by the amount of data in that class

Prev Up Next

scikit-learn 0.22.1
Other versions

Please cite us if you use the software.

sklearn.metrics.confusion_matrix
Examples using
sklearn.metrics.confusion_matri

sklearn.metrics.confusion_matrix

sklearn.metrics.**confusion_matrix**(*y_true*, *y_pred*, *labels=None*, *sample_weight=None*, *normalize=None*) [source]

Compute confusion matrix to evaluate the accuracy of a classification.

By definition a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i and predicted to be in group j.

Thus in binary classification, the count of true negatives is $C_{0,0}$, false negatives is $C_{1,0}$, true positives is $C_{1,1}$ and false positives is $C_{0,1}$.

Read more in the User Guide.

Parameters:

y_true : *array-like of shape (n_samples,)*
Ground truth (correct) target values.

f shape (n_samples,)
s returned by a classifier.

shape (n_classes), default=None
ex the matrix. This may be used to reorder or select a subset of labels. If `None` is given, those
once in `y_true` or `y_pred` are used in sorted order.

ay-like of shape (n_samples,), default=None

normalize : *{'true', 'pred', 'all'}, default=None*
Normalizes confusion matrix over the true (rows), predicted (columns) conditions or all the population. If None, confusion matrix will not be normalized.

```
>>> from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

sklearn.metrics.classification_report

sklearn.metrics.**classification_report**(*y_true*, *y_pred*, *labels=None*, *target_names=None*, *sample_weight=None*, *digits=2*, *output_dict=False*, *zero_division='warn'*) [source]

Build a text report showing the main classification metrics

Read more in the User Guide.

Parameters:

y_true : *1d array-like, or label in*
Ground truth (correct) target va

y_pred : *1d array-like, or label in*
Estimated targets as returned b

labels : *array, shape = [n_labels]*
Optional list of label indices to in

target_names : *list of strings*
Optional display names matchin

sample_weight : *array-like of sh*
Sample weights.

digits : *int*
Number of digits for formatting
the returned values will not be rounded.

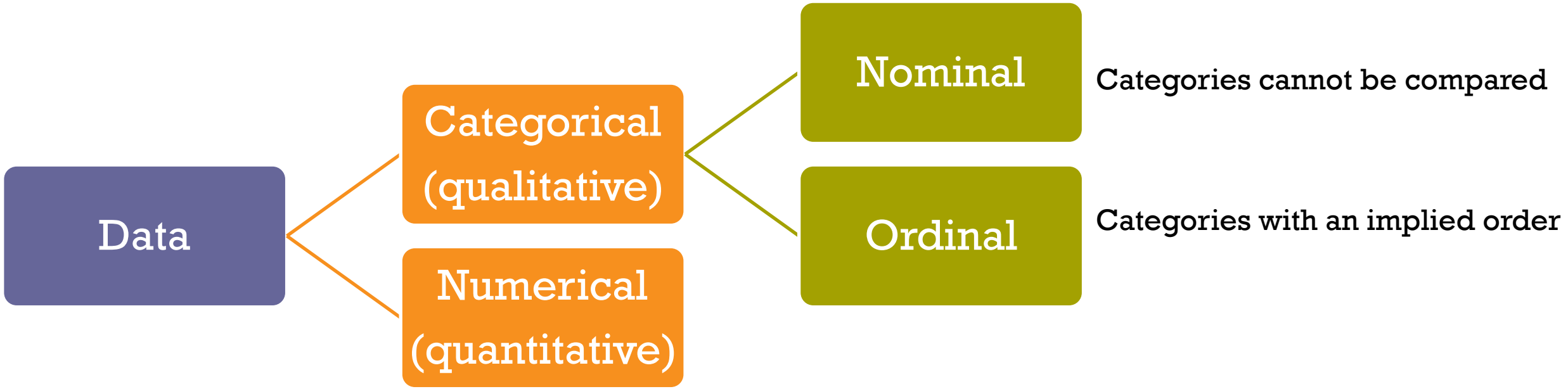
Examples

```
>>> from sklearn.metrics import classification_report
>>> y_true = [0, 1, 2, 2, 2]
>>> y_pred = [0, 0, 2, 2, 1]
>>> target_names = ['class 0', 'class 1', 'class 2']
>>> print(classification_report(y_true, y_pred, target_names=target_names))
              precision    recall  f1-score   support
<BLANKLINE>
     class 0       0.50      1.00      0.67         1
     class 1       0.00      0.00      0.00         1
     class 2       1.00      0.67      0.80         3
<BLANKLINE>
    accuracy                           0.60         5
   macro avg       0.50      0.56      0.49         5
weighted avg       0.70      0.60      0.61         5
<BLANKLINE>
>>> y_pred = [1, 1, 0]
>>> y_true = [1, 1, 1]
>>> print(classification_report(y_true, y_pred, labels=[1, 2, 3]))
              precision    recall  f1-score   support
<BLANKLINE>
           1       1.00      0.67      0.80         3
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
<BLANKLINE>
   micro avg       1.00      0.67      0.80         3
   macro avg       0.33      0.22      0.27         3
weighted avg       1.00      0.67      0.80         3
<BLANKLINE>
```

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

Non-numeric variables (Model M2)

Terminology: Kinds of data (recap)



Ordinal: Recode

Grade	GradeNum
A	4.00
B+	3.50
B	3.00
C+	2.50
C	2.00
D+	1.50
D	1.00
F	0.00

Size	SizeNum
XL	5
L	4
M	3
S	2
XS	1

Categorical

- Dummy coding = (n-1) dummy codes

Branch	BranchNum	D_BKK	B_Patum	D_Non	D_BKK	B_Patum	
BKK	1	1	0	0	1	0	
Patumtani	2	0	1	0	0	1	
Nontaburi	3	0	0	1	0	0	reference

Search the docs ...

Input/output
General functions
Series
DataFrame
Pandas arrays
Panel
Index objects
Date offsets
Window
GroupBy
Resampling
Style

pandas.get_dummies

pandas.**get_dummies**(*data, prefix=None, prefix_sep='_', dummy_na=False, columns=None, sparse=False, drop_first=False, dtype=None*) → 'DataFrame' [source]

Convert categorical variable into dummy/indicator variables.

Parameters: **data** : *array-like, Series, or DataFrame*

Data of which to get dummy indicators.

prefix : *str, list of str, or dict of str, default None*

String to append DataFrame column names. Pass a list with length equal to the number of columns when calling get_dummies on a DataFrame. Alternatively, *prefix* can be a dictionary mapping column names to prefixes.

prefix_sep : *str, default '_'*

If appending prefix, separator/delimiter to use. Or pass a list or dictionary as with *prefix*.

dummy_na : *bool, default False*

Add a column to indicate NaNs, if False NaNs are ignored.

ke, default None

s in the DataFrame to be encoded. If *columns* is None then all the columns with *object* or *category* converted.

efault False

dummy-encoded columns should be backed by a `SparseArray` (True) or a regular NumPy array (False).

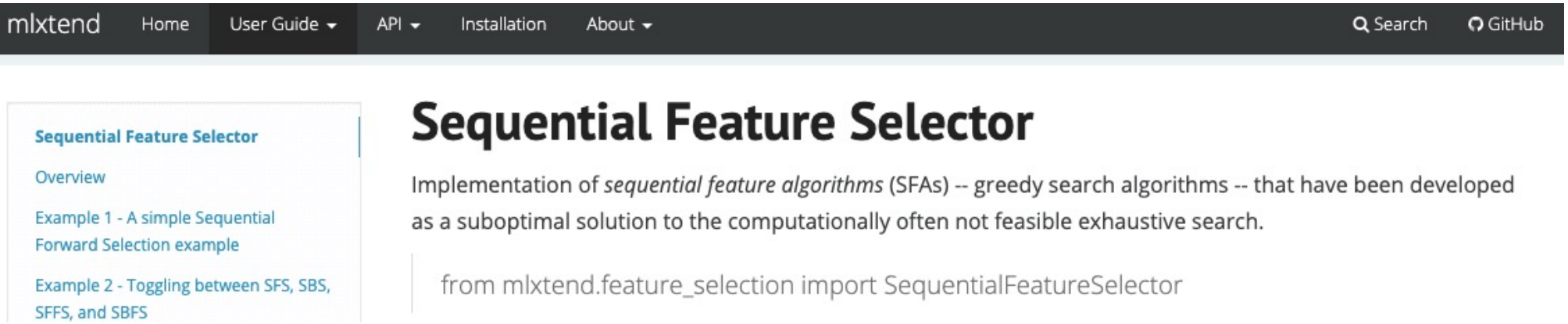
```
>>> pd.get_dummies(pd.Series(list('abcaa')), drop_first=True)
   b  c
0  0  0
1  1  0
2  0  1
3  0  0
4  0  0
```

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.get_dummies.html

Feature Selection (Model M3) - optional

Feature Selection

- Forward
- Backward
- Stepwise



- Sequential feature selection algorithms are a family of greedy search algorithms that are used to reduce an initial d-dimensional feature space to a k-dimensional feature subspace where $k < d$.
- In a nutshell, SFAs remove or add one feature at the time based on the classifier performance (such as, accuracy) until a feature subset of the desired size k is reached.
- Sequential Forward Selection (SFS)
- Sequential Backward Selection (SBS)
- Sequential Forward Floating Selection (SFFS)
- Sequential Backward Floating Selection (SBFS)

http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/

Seqeuntial Feature Selector (cont.)

- The ***floating*** variants, SFFS and SBFS, can be considered as extensions to the simpler SFS and SBS algorithms.
- The dimensionality of the subset during the search can be thought tobe "floating" up and down. (somewhat similar to stepwise)
  - Sequential Forward Floating Selection (SFFS)
  - Sequential Backward Floating Selection (SBFS)

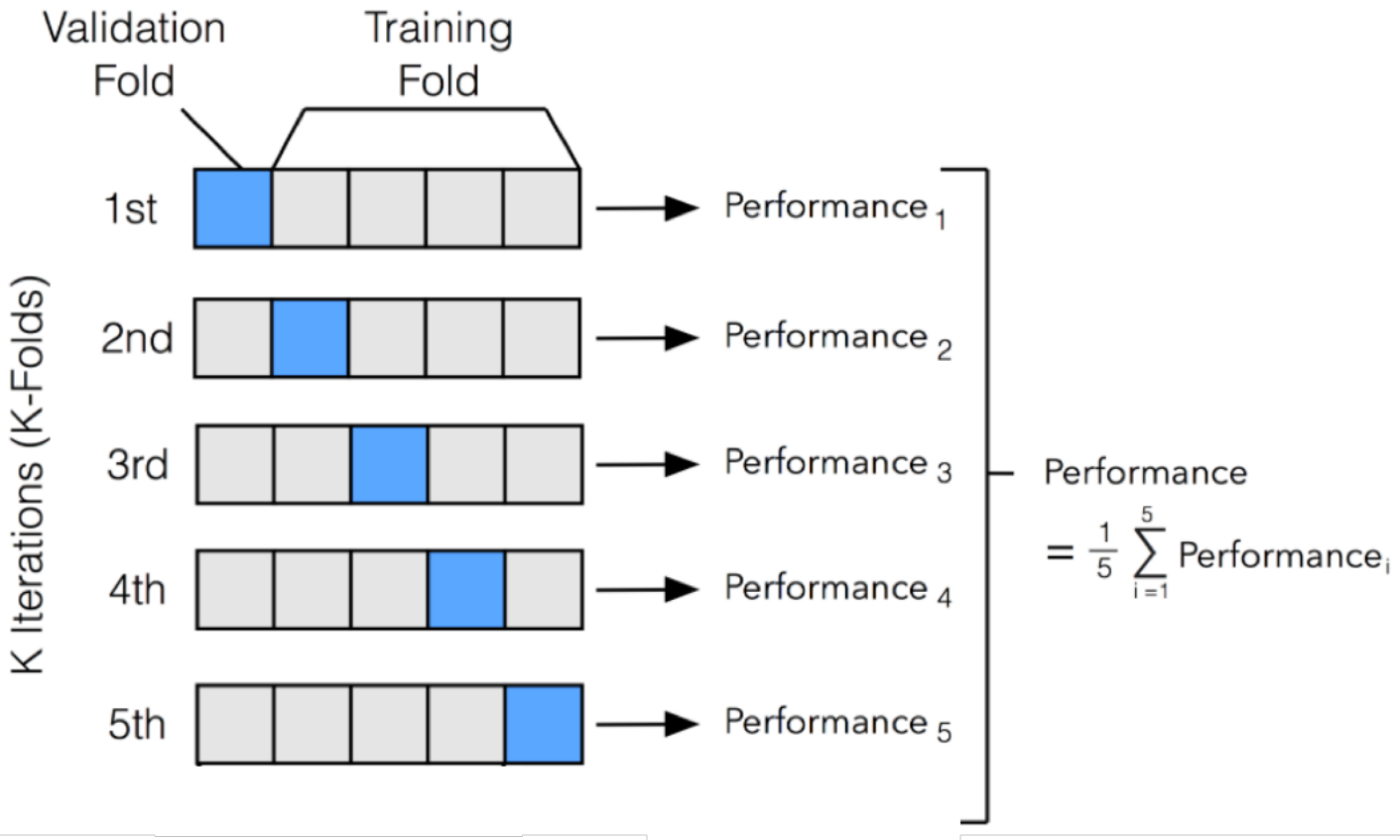
```
# Build RF classifier to use in feature selection
clf = LogisticRegression()

# Build step forward feature selection
sfs1 = sfs(clf,
           k_features=5,
           forward=True,
           floating=False,
           verbose=2,
           scoring='accuracy',
           cv=5)

# Perform SFFS
sfs1 = sfs1.fit(bank_final, y)
```

```
# Which features?
feat_cols = list(sfs1.k_feature_idx_)
print(feat_cols)
```

```
[6, 10, 15, 16, 17]
```

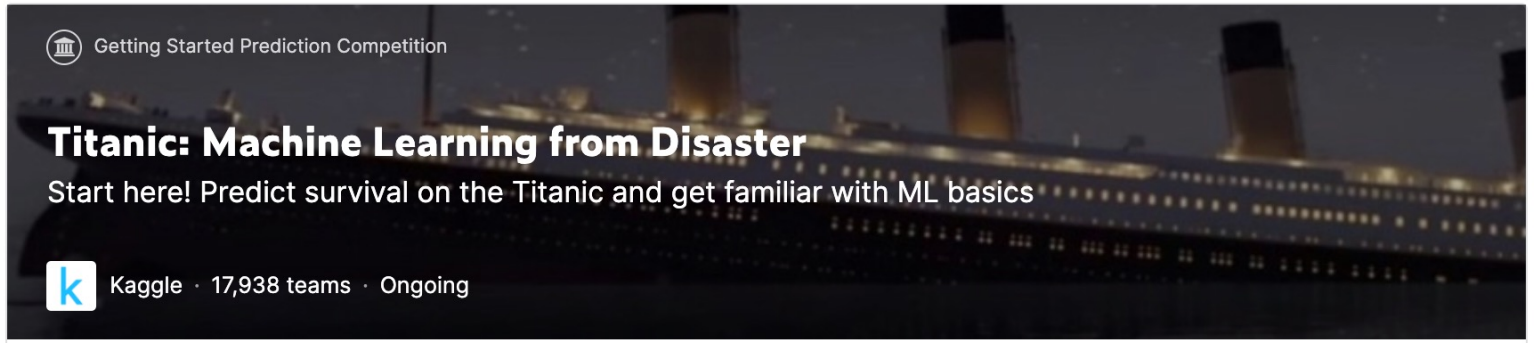


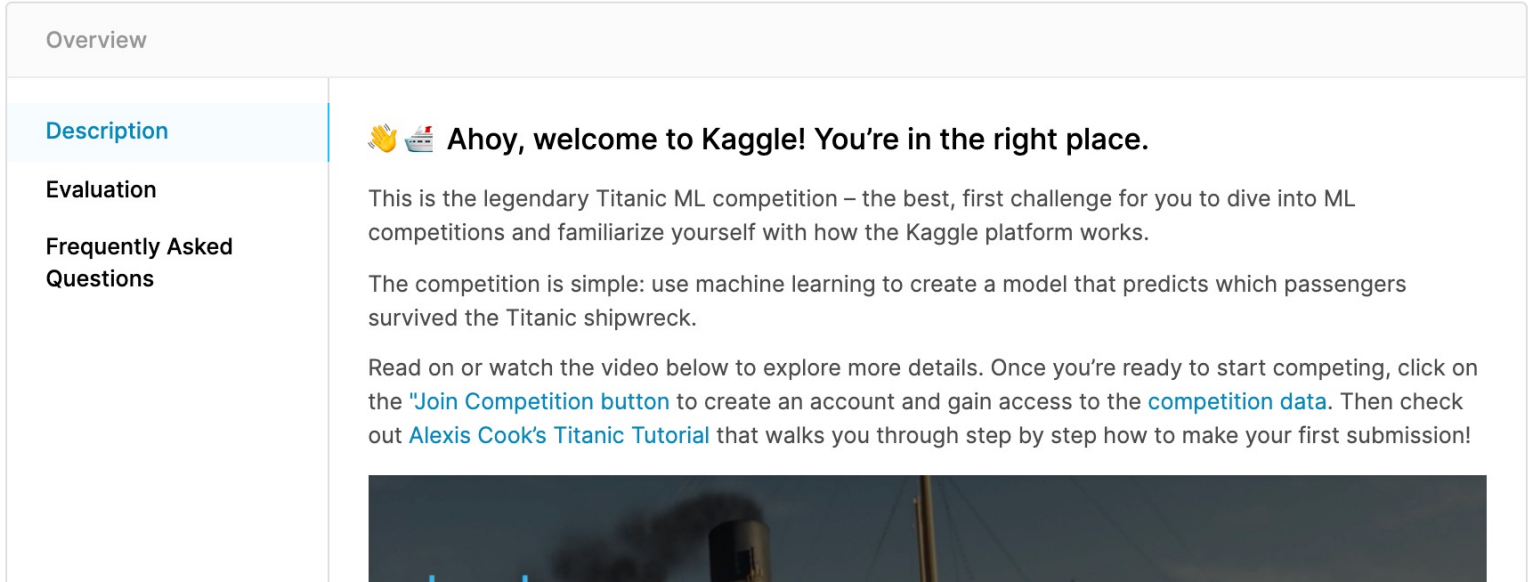
Demo

Titanic Survival Classification

Target='Survived' (binary: 0, 1)

https://github.com/davidjohnnn/all_datasets/raw/master/bay/titanic_train.csv





Titanic Survival Classification (cont.)

Target='Survived' (binary: 0, 1)

- Demo
  - Use only Age & Sex as inputs

- In-Class Activity
  - Use all variables
  - Impute missing values
    - Impute categorical with mode
    - Improve how to impute "Age". Rather than just using an average for the whole passengers, we should impute age differently for each passenger class (Pclass).
  - Some variables should be dropped manually
  - Evaluate and compare to the simple model in terms of Macro F1; which one is better?
  - Compute odds ratio for "Age" & explain it

Homework

https://github.com/kaopanboonyuen/Python-Data-Science/raw/master/Dataset/adult_data/adult.csv

Attribute Information:

Listing of attributes:

>50K, <=50K.

Target: income

age: continuous.
workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gc
fnlwgt: continuous.
education: Bachelors, Some-college, 11th, HS-grad, Prof-scho
education-num: continuous.
marital-status: Married-civ-spouse, Divorced, Never-married,
occupation: Tech-support, Craft-repair, Other-service, Sales,
relationship: Wife, Own-child, Husband, Not-in-family, Other-re
race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other,
sex: Female, Male.
capital-gain: continuous.
capital-loss: continuous.
hours-per-week: continuous.
native-country: United-States, Cambodia, England, Puerto-Ric
Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, G

train_test_split
- test = 0.30
- random_state = 101

Macro F1
- [0, 40) 5 points
- [40, 70) 8 points
- [70, 100] 10 points

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K