Logistic Regression

부산대학교 정보·의생명공학대학
정보컴퓨터공학부

Various Algorithms/Methods for Data Analysis

❖ Time to wrap-up

- We've learned some algorithms/methods for data analysis. kNN, Naïve Bayes, Linear Regression, K-means, t-test, χ^2 test, …
- Need to understand the purpose, assumptions and applicable data sets/types of an algorithm/method

❖ 2 Main Categories : Regression and Prediction, Classification

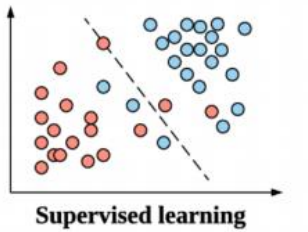
- 1) A certain keywords are included in an email. Is it an attempt at phishing ? We've changed the headline of our webpage, Is the web user likely to click on advertisement ?
- 2) Is the variable X (or more likely, $X_1, \ldots, X_p$) associated with a variable Y, and if so, what is the relationship and can we use it to predict Y ?

Supervised Learning	Classification	Naïve Bayes
		kNN
		Logistic Regression
	Regression and Prediction	Linear Regression
Unsupervised Learning	Clustering	K Means

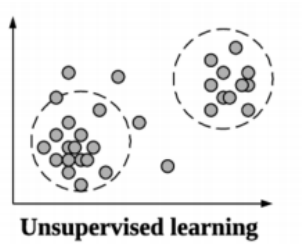
Machine Learning

- Automated means to learn from data and solve tasks

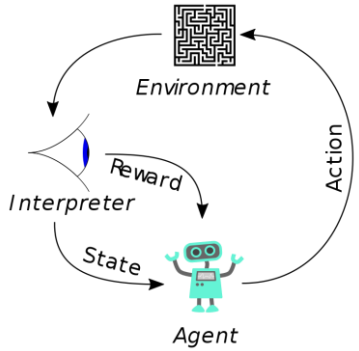
- Supervised Learning
  - Labeled data
  - Example : Classification

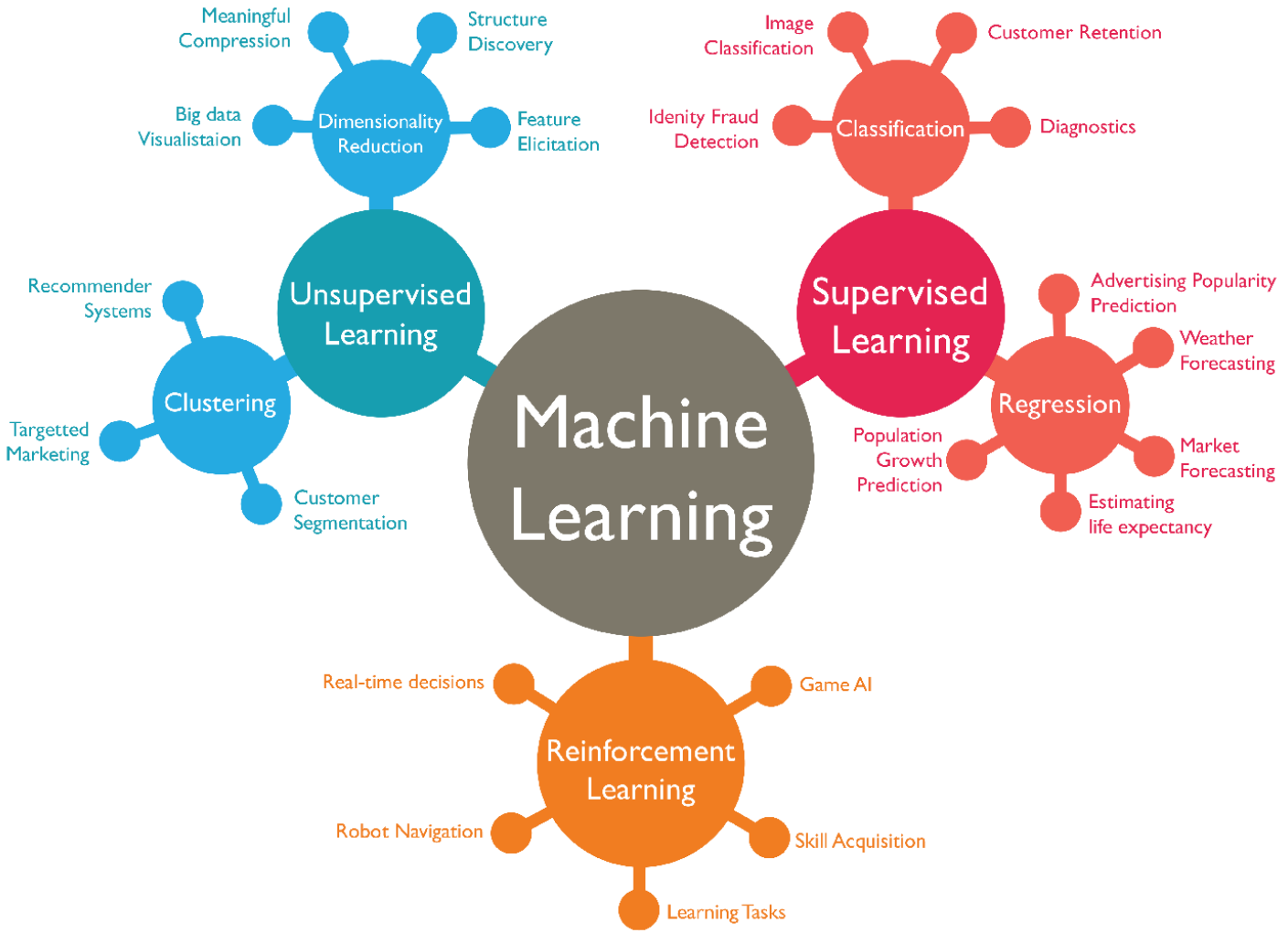


- Unsupervised Learning
  - No labeled data
  - Example : Feature Extraction



- Reinforcement Learning





Variable Types and Methods

❖ Variables

- Ex) The height of father and The height of son
- Ex) The score of final exam and the final grade
- Ex) The grade of Discrete mathematics and the grade of Linear Algebra

❖ Many (Confusing) Names for Independent variables

- Explanatory, **Predictor**, Control, Feature, Factor
  - Feature : Used in ML and Predictive Model, Factor : A categorical predictor variable

❖ Data Type - Categorical Data vs Numerical Data

- Categorical Data – Nominal, Ordinal

❖ Simple Linear Regression

- For Regression & Predication
- Numerical → **Numerical**

❖ Naïve Bayes

- For Classification
- Categorical/Numerical → **Categorical**

❖ Logistic Regression

- For Classification
- Categorical/Numerical → **Categorical**

Analysis Process

❖ Data → Model → Train/Fit → Evaluation

❖ Example)

- A group of 20 students spends between 0 and 6 hours studying for an exam.

  How does the number of hours spent studying affect the probability of the student passing the exam?

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

  - the number of hours each student spent studying, and whether they passed (1) or failed (0).
- Q) For a student who studies 2 hours, will he/she pass the exam ?
- Q) Can you estimate the probability of passing the exam ?
- Model
  - Simple Linear Regression
  - Naïve Bayes
  - Logistic Regression
- Evaluation ?

Colab Package Update

- Some features introduced in the slide will not work in Colab as it is
  - requires higher versions for some packages
- You can update packages in the Colab
  - You need to restart/resume your colab VM instance.
  - After disconnection, the updated instance will be removed.

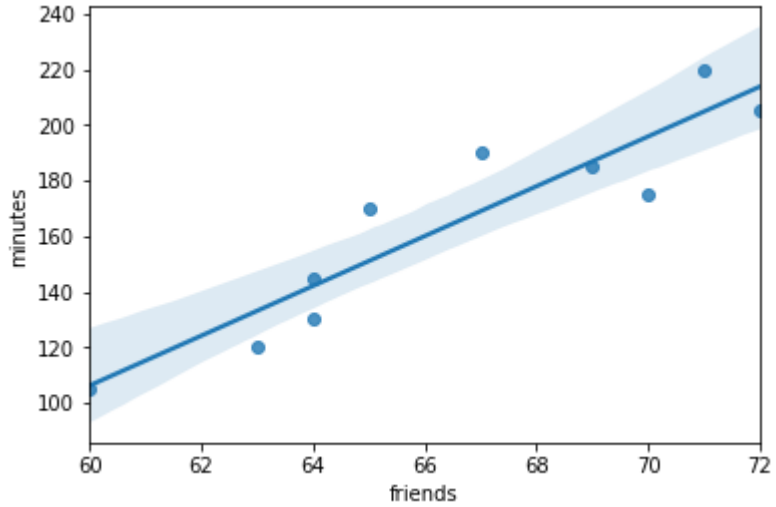
    so you have to update again.
- Packages needed to be updated
  - numpy : v1.19.5 → v1.20.3
  - pandas : v1.1.5 → 1.2.4
  - scipy v1.4.1 → v1.6.3
  - **scikit-learn** v0.22.2.post1 → 0.24.2
  - statsmodels v0.10.2 → 0.12.2

```
import numpy
import pandas
import scipy
import sklearn
import statsmodels

print(numpy.__version__)
print(pandas.__version__)
print(scipy.__version__)
print(sklearn.__version__)
print(statsmodels.__version__)

#!pip uninstall numpy -y
#!pip install -U numpy
#!pip uninstall pandas -y
#!pip install -U pandas
#!pip uninstall scipy -y
#!pip install -U scipy
#!pip uninstall scikit-learn -y
#!pip install -U scikit-learn
#!pip uninstall statsmodels -y
#!pip install -U statsmodels
```

Simple Linear Regression Revisited (1)

```
import numpy as np
import pandas as pd
from sklearn import linear_model
from sklearn import metrics
import seaborn as sn

# Prepare Data
data = pd.read_csv('https://raw.githubusercontent.com/inetguru/IDS-CB35533/main/datum.csv')
print(data.head())

X = data[['friends']]
y = data['minutes']

sn.regplot(x='friends',y='minutes', data=data[['friends','minutes']])
plt.show()

# Select a Model
model = linear_model.LinearRegression()

# Fit/Train
model.fit(X,y)
print('Intercept =', model.intercept_, 'coef =', model.coef_)

# Evaluation
X_test = X
y_test = y
y_pred = model.predict(X_test)

print('MSE =', metrics.mean_squared_error(y_test, y_pred))
print('Coefficient of determination : ', metrics.r2_score(y_test, y_pred))
```

	X	Y
name	friends	minutes
Hero	70	175
Dunn	65	170
Sue	72	205
Chi	63	120
Thor	71	220
Clive	64	130
Hicks	60	105
Devin	64	145
Kate	67	190
Klein	69	185



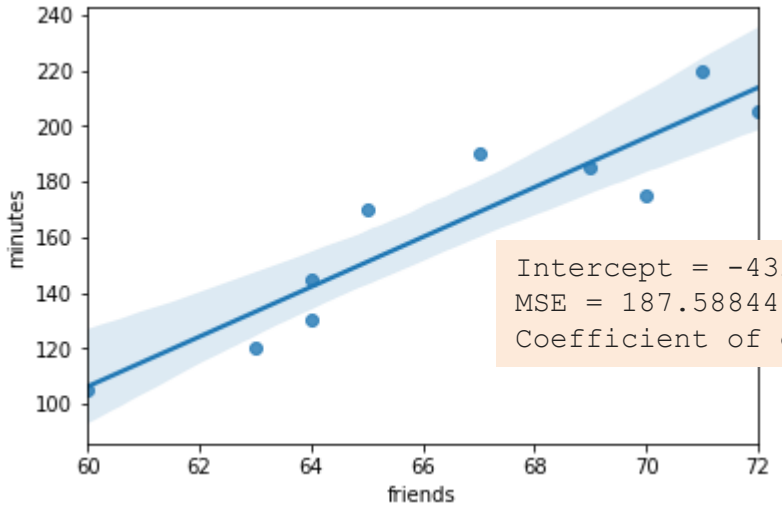
Simple Linear Regression Revisited (2)

- Use “Train Data” as “Test Data”

- Metrics for Evaluation
  - MSE = $E[(\hat{y}_i - y_i)^2]$
  - Coefficient of Determination, r^2 score

- Sklearn.metrics functions
  - mean_squared_error(y_test, y_pred)
  - r2_score(y_test, y_pred)

```
# Select a Model
model = linear_model.LinearRegression()

# Fit/Train
model.fit(X,y)
print('Intercept =', model.intercept_, 'coef =', model.coef_)

# Evaluation
X_test = X
y_test = y
y_pred = model.predict(X_test)

print('MSE =', metrics.mean_squared_error(y_test, y_pred))
print('Coefficient of determination : ', metrics.r2_score(y_test, y_pred))
```




```
Intercept = -432.0794223826715 coef = [8.97111913]
MSE = 187.5884476534296
Coefficient of determination : 0.8559505105368174
```

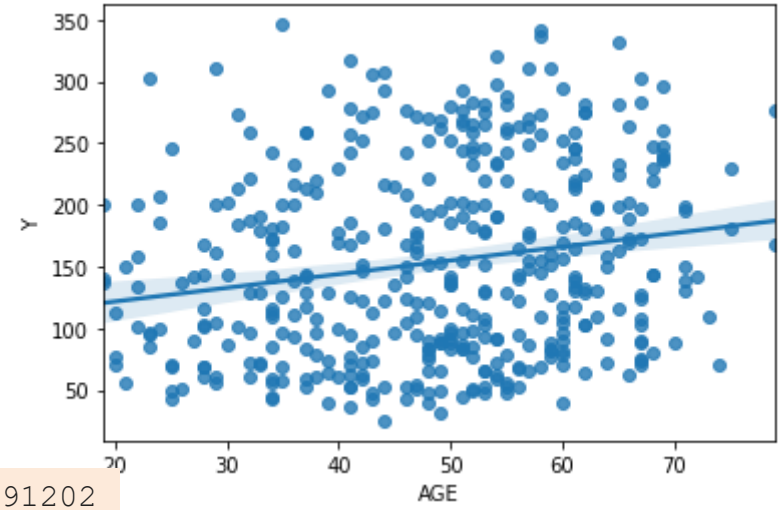
Multiple Linear Regression

❖ $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \epsilon$

- Multiple predictor variables

AGE	SEX	BMI	BP	S1	S2	S3	S4	S5	S6	Y
59	2	32.1	101	157	93.2	38	4	4.8598	87	151
48	1	21.6	87	183	103.2	70	3	3.8918	69	75
72	2	30.5	93	156	93.6	41	4	4.6728	85	141
24	1	25.3	84	198	131.4	40	5	4.8903	89	206
50	1	23	101	192	125.4	52	4	4.2905	80	135
23	1	22.6	89	139	64.8	61	2	4.1897	68	97
36	2	22	90	160	99.6	50	3	3.9512	82	138
66	2	26.2	114	255	185	56	4.55	4.2485	92	63
60	2	32.1	83	179	119.4	42	4	4.4773	94	110
29	1	30	85	180	93.4	43	4	5.3845	88	310
22	1	18.6	97	114	57.6	46	2	3.9512	83	101
56	2	28	85	184	144.8	32	6	3.5835	77	69

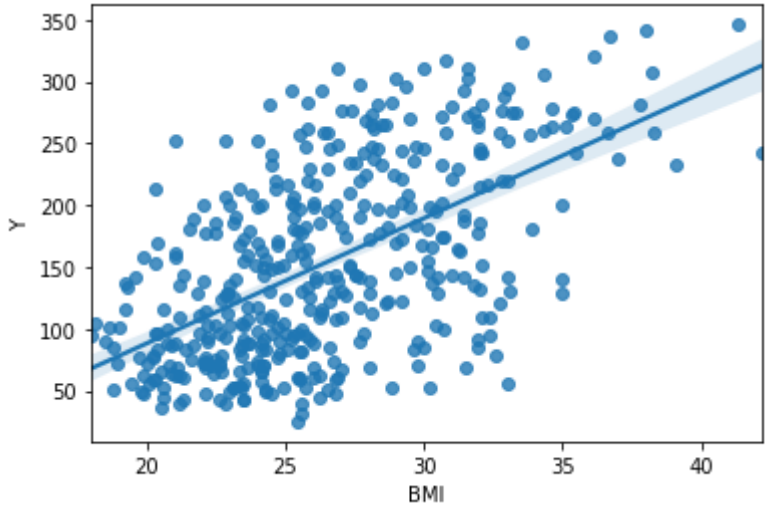
```
data = pd.read_csv('https://raw.githubusercontent.com/
inetguru/IDS-CB35533/main/diabetes.csv')
data.head()

train_data = data[:420]
test_data = data[421:442]

predictors = ['AGE']
target = 'Y'

X = train_data[predictors]
y = train_data[target]

sn.regplot(x=predictors[0],y=target, data=train_data[[predictors[0],target]])
plt.show()

model = linear_model.LinearRegression()
model.fit(X,y)

print('Intercept =', model.intercept_)
for name, coef in zip(predictors, model.coef_) :
  print(f' {name} : {coef}')
```




```
Intercept = 99.37114017191202
AGE : 1.108748462599086
```

Assign multiple predictors

```
data = pd.read_csv('https://raw.githubusercontent.com/inetguru/IDS-CB35533/main/diabetes.csv')
data.head()

train_data = data[:420]
test_data = data[421:442]

#predictors = ['AGE']
predictors = ['BMI', 'BP']

X = train_data[predictors]
y = train_data[target]

sn.regplot(x=predictors[0],y=target, data=train_data[[predictors[0],target]])
plt.show()

model = linear_model.LinearRegression()
model.fit(X,y)

print('Intercept =', model.intercept_)
for name, coef in zip(predictors, model.coef_) :
  print(f' {name} : {coef}')
```




```
Intercept = -202.8237181361084
BMI : 8.426293227749216
BP : 1.4083510337859113
```

Multiple Linear Regression - Evaluation

❖ Divide train data / test data

```
train_data = data[:420]
test_data = data[421:442]

#predictors = ['AGE']
predictors = ['BMI', 'BP']
target = 'Y'

X = train_data[predictors]
y = train_data[target]

sn.regplot(x=predictors[0],y=target, data=train_data[[predictors[0],target]])
plt.show()

model = linear_model.LinearRegression()
model.fit(X,y)

print('Intercept =', model.intercept_)
for name, coef in zip(predictors, model.coef_) :
  print(f' {name} : {coef}')

X_test = test_data[predictors]
y_test = test_data[target]
y_pred = model.predict(X_test)

print('MSE =', metrics.mean_squared_error(y_test, y_pred))
print('R2 Score =', metrics.r2_score(test_data[['Y']], y_pred))
```

```
MSE = 2521.597070760351
R2 Score = 0.4890345591616515
```

Naïve Bayes Revisited

❖ Will he play tennis if …

- PlayTennis : Yes/No
- Estimate the probability of “PlayTennis”
  - Pr(No | <Sunny, Cool, High, Strong>) = ?

$$P(y|x_1, x_2, \dots, x_k) = P(x_1, x_2, \dots, x_k|y) \cdot \frac{P(y)}{P(x_1, x_2, \dots, x_k)}$$

$$P(x_1, x_2, \dots, x_k) = P(y) \cdot P(x_1, x_2, \dots, x_k|y) + P(\bar{y}) \cdot P(x_1, x_2, \dots, x_k|\bar{y})$$

- There is no <sunny, cool, high, strong> case for “PlayTennis = no”.
- **Conditional Independence** → **Naïve** (assumption) but practical
- $P(x_1, x_2, \dots, x_k|y) = P(x_1|y) \cdot P(x_2|y) \cdot \cdots \cdot P(x_k|y)$
  - P(Yes)=9/14, P(No) = 5/14
  - P(Sunny|Yes) = 2/9, P(Cool|Yes)=3/9, P(High|Yes)=3/9, P(Strong|Yes) = 3/9
  - P(Sunny|No) = 3/5, P(Cool|No)=1/5, P(High|No) = 4/5, P(Strong|No) = 3/5
- Smoothing : what happen if $P(x_i|y) = 0$

	X_1	X_2	X_3	X_4	Y
Day	**Outlook**	**Temperature**	**Humidity**	**Wind**	**PlayTennis**
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$$P(yes)\ P(sunny|yes)\ P(cool|yes)\ P(high|yes)\ P(strong|yes) = .0053$$
$$P(no)\ P(sunny|no)\ P(cool|no)\ P(high|no)\ P(strong|no) = .0206$$

P(No | <Sunny, Cool, High, Strong>)=$\frac{0.0206}{0.0206+0.0053} = 0.795$

Naïve Bayes Revisited

- Data preprocessing is necessary
  - Categorical data in string → Categorical data in value
- Sklearn.preprocessing – Encoding categorical features
  - LabelEncoder()
    - OrdinalEncoder(), OneHotEncoder()

	X_1	X_2	X_3	X_4	Y
Day	**Outlook**	**Temperature**	**Humidity**	**Wind**	**PlayTennis**
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- Label Encoder
  - transforms each categorical feature to one new feature of integers (0 to n_categories - 1):
  - Alphabetical Order
  - Ex) Outlook – ‘Sunny’, ‘Overcast’, ‘Rain’ → ‘Overcast’ → 0, ‘Rain’ → 1, ‘Sunny’ → 2

```
import numpy as np
import pandas as pd
from sklearn import naive_bayes
from sklearn import metrics
from sklearn import preprocessing

data = pd.read_csv('https://raw.githubusercontent.com/inetguru/IDS-CB35533/main/tennis_play.csv')

columns = {}
label_encoders = {}
for feature in data.columns[1:] :
  label_encoders[feature] = preprocessing.LabelEncoder()
  columns[feature] = label_encoders[feature].fit_transform(data[feature])
  print(label_encoders[feature].classes_)

print(label_encoders['Outlook'].classes_)
print(columns['Outlook'])
print(label_encoders['Outlook'].inverse_transform(columns['Outlook']))

edata = pd.DataFrame(columns)
print(edata.head())
```

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes

```
['Overcast' 'Rain' 'Sunny']
['Cool' 'Hot' 'Mild']
['High' 'Normal']
['Strong' 'Weak']
['No' 'Yes']

['Overcast' 'Rain' 'Sunny']
[2 2 0 1 1 1 0 2 2 1 2 0 0 1]
['Sunny' 'Sunny' 'Overcast' 'Rain' 'Rain' 'Rain'
'Overcast' 'Sunny' 'Sunny' 'Rain' 'Sunny' 'Overcast'
'Overcast' 'Rain']

   Outlook  Temperature  Humidity  Wind  PlayTennis
0        2            1         0     1           0
1        2            1         0     0           0
2        0            1         0     1           1
3        1            2         0     1           1
4        1            0         1     1           1
```

Naïve Bayes - sklearn

❖ Sklearn Naïve_Bayes

- The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i|y)$
- Gaussian Naïve Bayes
- Multinomial Naïve Bayes
- **Categorical Naïve Bayes**

```
edata = pd.DataFrame(columns)
#print(edata.head())

predictors = ['Outlook', 'Temperature', 'Humidity', 'Wind']
target = 'PlayTennis'

X = edata[predictors]
y = edata[target]

model = naive_bayes.CategoricalNB(alpha=0.001)
model.fit(X,y)

print(model.predict([[2,0,0,0]]))
print(model.predict_proba([[2,0,0,0]]))
```

```
['Overcast' 'Rain' 'Sunny']
['Cool' 'Hot' 'Mild']
['High' 'Normal']
['Strong' 'Weak']
['No' 'Yes']
```

<Sunny, Cool, High, Strong>

<2,0,0,0>

```
[0]
[[0.7953405 0.2046595]]
```

MOTIVATING EXAMPLE – LOGISTIC REGRESSION

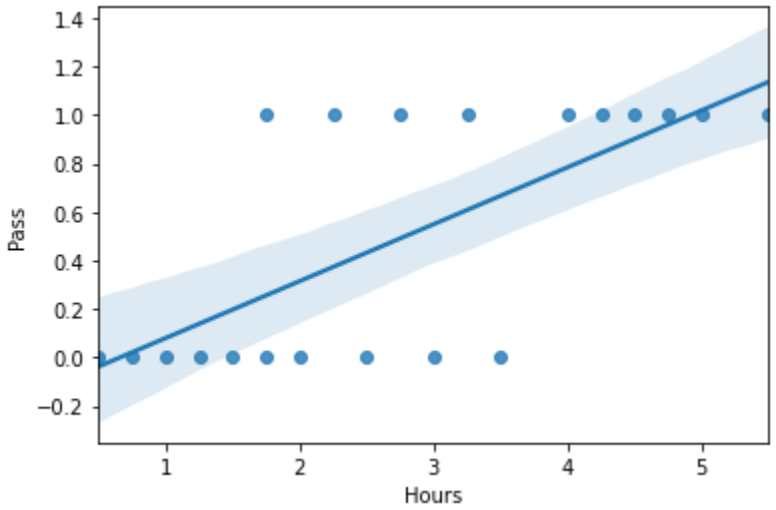
Pass/Fail Classification/Decision

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

- Q) For a student who studies 2 hours, will he/she pass the exam ?
- Q) Can you estimate the probability of passing the exam ?
- Model
  - Simple Linear Regression
  - Naïve Bayes
  - Logistic Regression
- Sklearn function
- model.predict()
- model.predict_proba()

```
import numpy as np
import pandas as pd
from sklearn import naive_bayes
from sklearn import linear_model
from sklearn import metrics
import seaborn as sn

data = pd.read_csv('https://raw.githubusercontent.com/
inetguru/IDS-CB35533/main/study_pass.csv')

predictors = ['Hours']
target = 'Pass'
X = data[predictors]
y = data[target]

sn.regplot(x=predictors[0],y=target, data=data[[target,predictors[0]]])

model1 = linear_model.LinearRegression()
model2 = naive_bayes.CategoricalNB(alpha=0.001)
model3 = linear_model.LogisticRegression()
```

Will he/she pass the exam ?




```
model1.fit(X,y)
model2.fit(X,y)
model3.fit(X,y)

X_test = X
y_test = y
y_pred1 = model1.predict(X_test)
y_pred2 = model2.predict(X_test)
y_pred3 = model3.predict(X_test)

print(y_test.values)
print(y_pred1)
print(y_pred2)
print(y_pred3)
```

Test data/Labelled data

```
[0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1]
```

Linear Regression

```
[-0.03663746 0.02201144 0.08066034 0.13930925
0.19795815 0.25660705 0.25660705 0.31525596
0.37390486 0.43255376 0.49120266 0.54985157
0.60850047 0.66714937 0.78444718 0.84309608
0.90174499 0.96039389 1.01904279 1.1363406 ]
```

Naïve Bayes

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1]
```

Logistic Regression

```
[0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1]
```

Can you estimate the probability of passing the exam ?

```
model1.fit(X,y)
model2.fit(X,y)
model3.fit(X,y)

X_test = X
y_test = y
y_pred1 = model1.predict(X_test)
y_pred2 = model2.predict(X_test)
y_pred3 = model3.predict(X_test)

print(y_test.values)
print(y_pred1)
print(y_pred2)
print(y_pred3)

#y_pred1_p = model1.predict_proba(X_test)
# Not Available

y_pred2_p = model2.predict_proba(X_test)
y_pred3_p = model3.predict_proba(X_test)

print(y_pred2_p[:,1])
print(y_pred3_p[:,1])
```

```
[0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1]

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1]

[0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1]

[4.99500500e-04 4.99500500e-04 2.00119952e-01
2.00119952e-01 2.00119952e-01 2.00119952e-01
2.00119952e-01 5.00000000e-01 5.00000000e-01
5.00000000e-01 5.00000000e-01 3.33444370e-01
3.33444370e-01 3.33444370e-01 9.99750125e-01
9.99750125e-01 9.99750125e-01 9.99750125e-01
9.99500500e-01 9.99500500e-01]

[0.07141048 0.09295545 0.12015954 0.15397362 0.19519227
0.24425941 0.24425941 0.30104715 0.36466683 0.43339461
0.50478398 0.57597882 0.6441537 0.70694435 0.81075008
0.85094629 0.88382827 0.91022165 0.93108619 0.95999014]
```

Evaluation

❖ Confusion Matrix (혼동 행렬)

- In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm.

Actual class \ Predicted class	Cat	Dog
Cat	**6**	2
Dog	1	**3**

Actual class \ Predicted class	P (Positive)	N (Negative)
P	**TP (True Positive) Hit**	FN (False Negative) Type II error overestimation
N	FP Type I error False alarm underestimation	**TN Correct rejection**

❖ Metrics

- Precision (정밀도) = $\frac{TP}{TP+FP} = \frac{6}{6+1} = \frac{6}{7}$
- Accuracy (정확도) = $\frac{TP+TN}{TP+FP+TN+FN} = \frac{6+3}{6+1+3+2} = \frac{9}{12}$
- Recall (재현율) = $\frac{TP}{TP+FN} = \frac{6}{6+2} = \frac{6}{8}$
- Fall out (위양성율) = $\frac{FP}{FP+TN} = \frac{1}{1+3} = \frac{1}{4}$
- F1 Score = $2 \times \frac{Precision \times Recall}{Precision+Recall}$

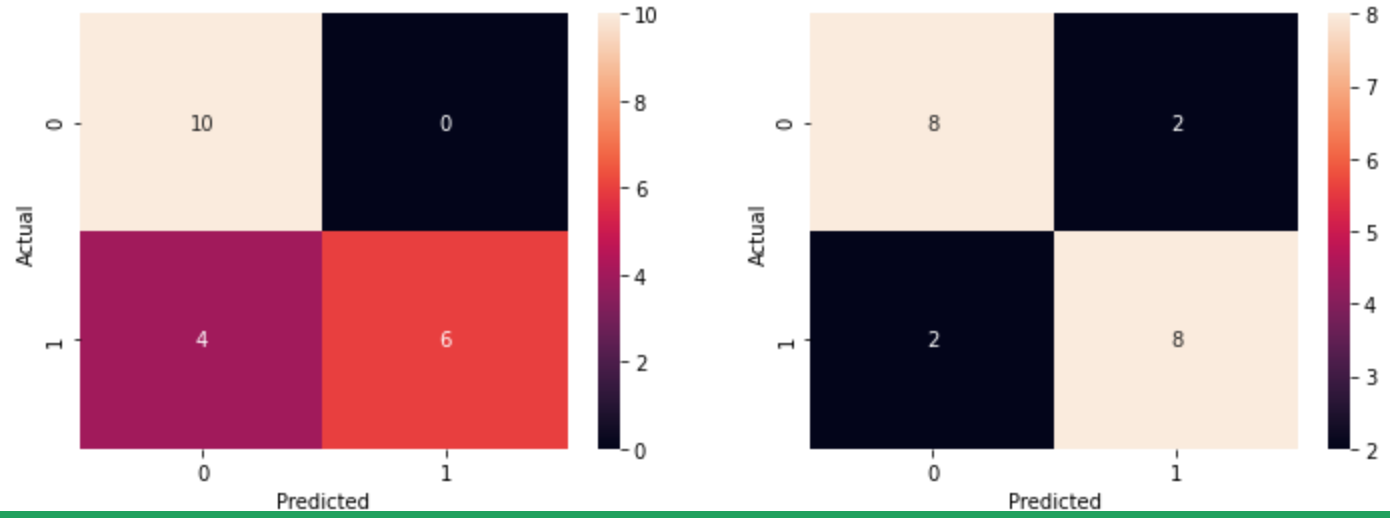
Building confusion matrices for evaluation

- Pandas crosstab() – Compute a frequency table of the factors
  - Use seaborn heatmap() for Visualization

```
confusion_matrix2 = pd.crosstab(y_test, y_pred2, rownames=['Actual'], colnames=['Predicted'])
confusion_matrix3 = pd.crosstab(y_test, y_pred3, rownames=['Actual'], colnames=['Predicted'])

plt.rcParams['figure.figsize'] = [12,4]
fig = plt.figure()
ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)
sn.heatmap(confusion_matrix2, annot=True, ax=ax1)
sn.heatmap(confusion_matrix3, annot=True, ax=ax2)
plt.show()

print('Accuracy (model2) : ', metrics.accuracy_score(y_test, y_pred2))
print('Accuracy (model3) : ', metrics.accuracy_score(y_test, y_pred3))
```

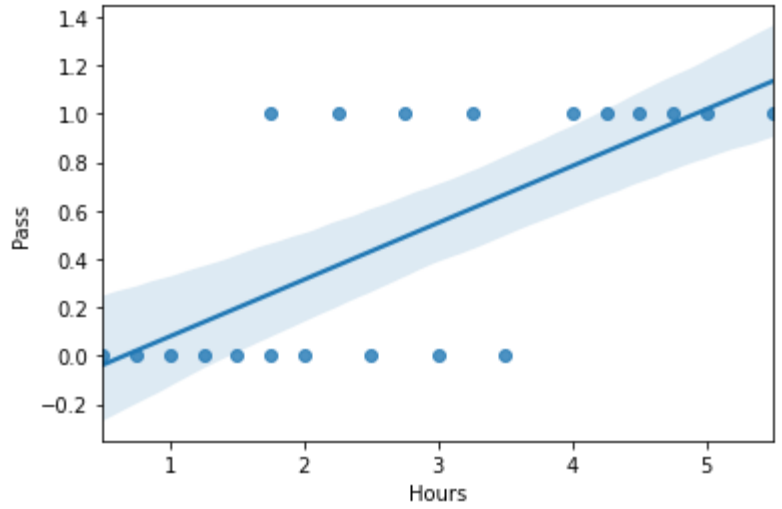
```
Accuracy (model2) : 0.8
Accuracy (model3) : 0.8
```

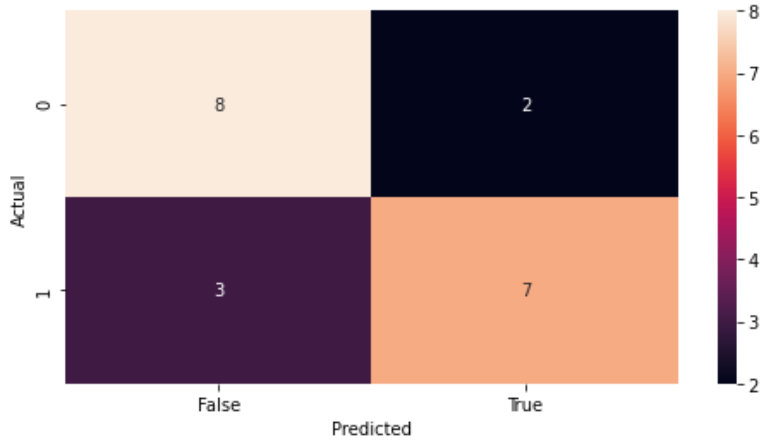
- Metrics – accuracy_score()



Linear Regression for Classification ?

- Threshold value based classification ?
  - If (predicted value > 0.5) then pass else fail ?





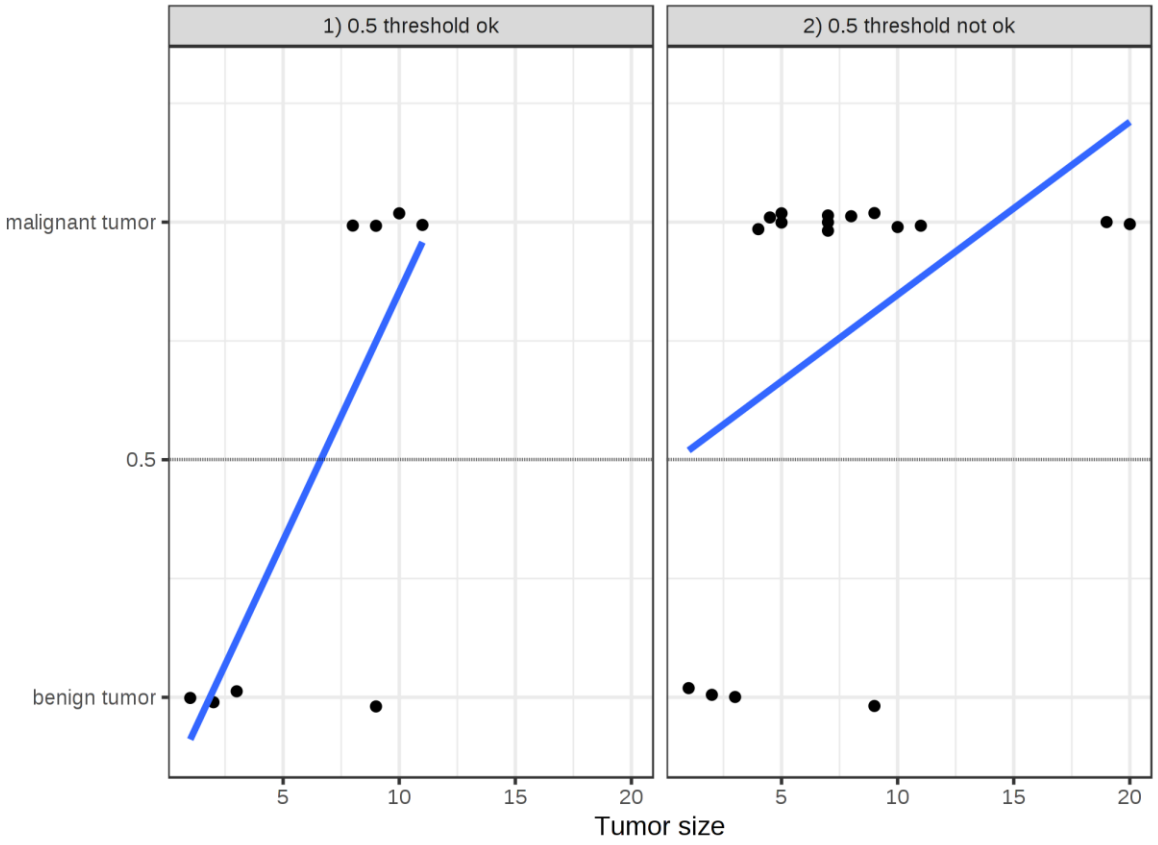

```
[False False False False False False False False False
False False True True True True True True True True True]

Accuracy (model1) : 0.75
```

```
y_pred1m = (y_pred1 > 0.5)
print(y_pred1m)

confusion_matrix1 = pd.crosstab(y_test, y_pred1m, rownames=['Actual'], colnames=['Predicted'])
sn.heatmap(confusion_matrix1, annot=True)
plt.show()

print('Accuracy (model1) : ', metrics.accuracy_score(y_test, y_pred1m))
```

LOGISTIC REGRESSION

Why Logistic Regression ?

❖ The limit of Linear Regression in Classification



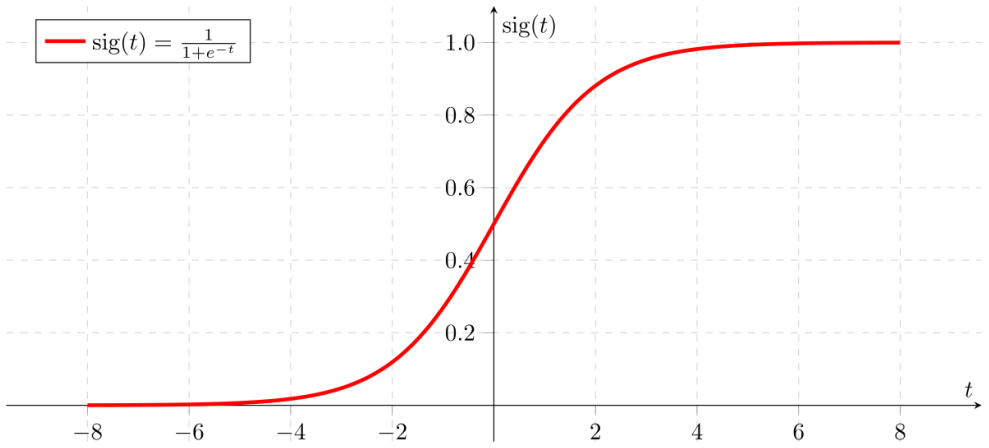
Source : https://christophm.github.io/interpretable-ml-book/logistic.html

❖ Solution - Logistic regression

- Instead of fitting a straight line or hyperplane, the logistic regression model uses the **logistic function** to squeeze the output of a linear equation between 0 and 1.
- The logistic function is defined as:

$$\text{logistic}(x) = \frac{1}{1 + e^{-x}}$$

  - Sigmoid curve/function



Logistic Regression

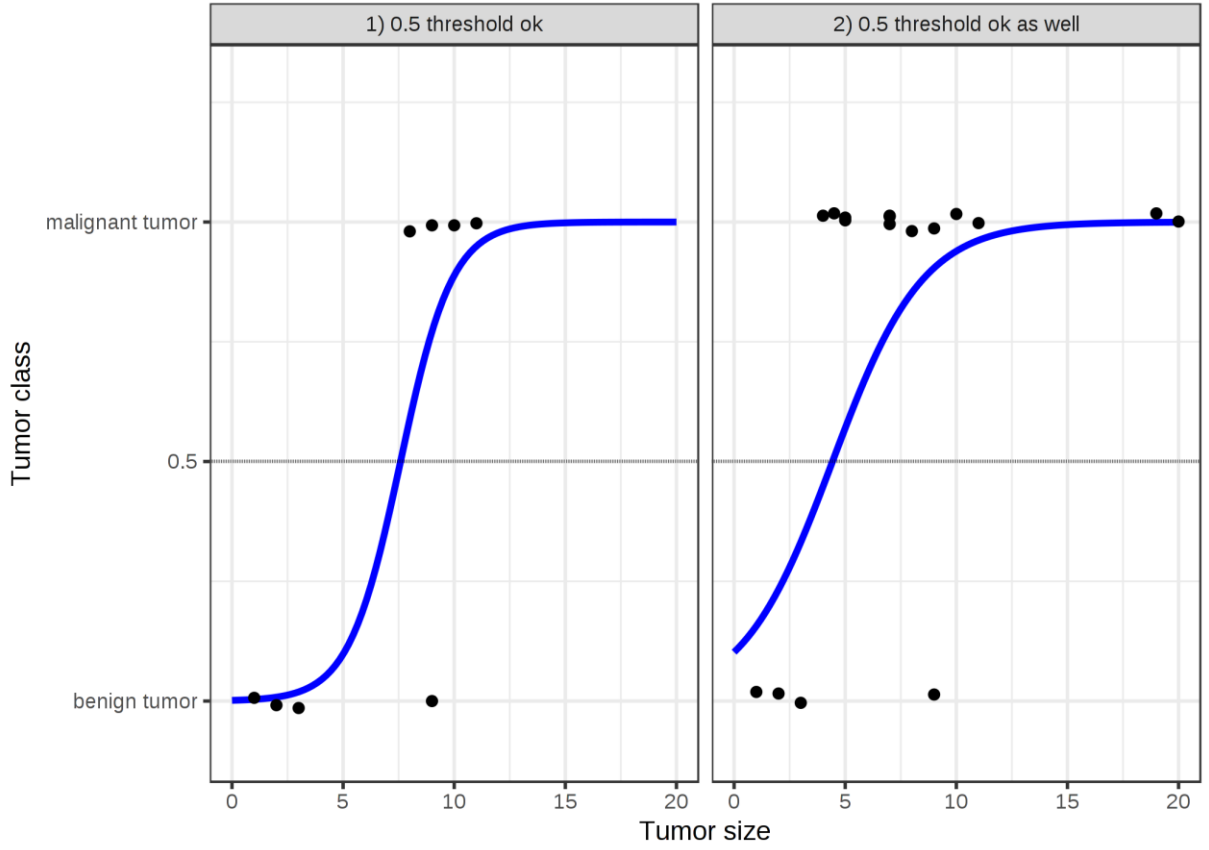
❖ (Multiple) Linear Regression

$$\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k$$

❖ Logistic Regression

$$P(y = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k)}}$$

- The weights do not influence the probability linearly any longer.
  - The interpretation of the weights in logistic regression differs from the interpretation of the weights in linear regression, since the outcome in logistic regression is a probability between 0 and 1..



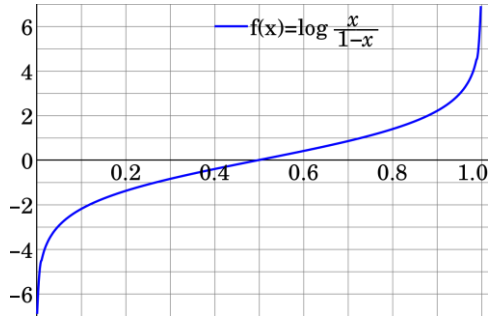
Logistic Regression

❖ Logistic regression vs. multiple linear regression

- Logistic regression is analogous to multiple linear regression except the outcome is binary.
- logistic regression is a structured model approach rather than a data-centric approach.
  - Due to its fast computational speed and its output of a model that lends itself to rapid scoring of new data, it is a popular method.

❖ Key Terms

- Odds (승산) : The ratio of "success" (1) to "not success" (0)
  - Let p be the probability of "success", Odds = $p/(1-p)$
    - Odds = 1 if $p = 0.5$, Odds = 1/4 if $p = 0.2$, Odds = 4 if $p = 0.8$
- Logit : a logarithmic function of odds that maps probability value from (0, 1) to $(-\infty, +\infty)$.
  - Also called Log-odds
  - $\text{logit}(p) = \log\left(\frac{p}{1-p}\right) = \log(p) - \log(1-p) = -\log\left(\frac{1}{p} - 1\right)$
  - The "logistic" function of p is given by the inverse-logit : $\text{logit}^{-1}(p) = \frac{1}{1+\exp(-p)} = \frac{1}{1+e^{-p}} = \frac{e^p}{1+e^p}$



Logistic Regression

- Consider a model with a predictor x_1 and x_2, and one binary response variable Y, which we denote $p = P(Y = 1)$.

❖ We assume a **linear relationship** between the **predictor** variables and **the log-odds of the event that $Y = 1$**.

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

- We can recover p from the inverse of log-odds

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}}$$

- The logistic distribution constrains the estimated probabilities p to lie between 0 and 1.
- If you let $\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0$ then $p = 0.5$
- As $\beta_0 + \beta_1 x_1 + \beta_2 x_2$ gets very big, p approaches 1.
- As $\beta_0 + \beta_1 x_1 + \beta_2 x_2$ gets very small, p approaches 0.

Fitting Equation to the Data

❖ Linear regression: Least squares

❖ https://en.wikipedia.org/wiki/Logistic_regression

❖ Logistic regression: Maximum likelihood

- Unlike linear regression with normally distributed residuals, it is not possible to find a closed-form expression for the coefficient values that maximize the likelihood function, so that an iterative process must be used instead.

❖ Likelihood function

- Measures the goodness of fit of a statistical model to a sample of data for given values of the unknown parameters
- Estimates parameters βs
- Practically easier to work with log-likelihood

❖ Maximum Likelihood

- Iterative computing
  - Choice of an arbitrary value for the coefficients (usually 0)
  - Computing of log-likelihood
  - Variation of coefficients' values
  - Reiteration until maximisation (plateau)
- Results
  - Maximum Likelihood Estimates (MLE) for βs
  - Estimates of $P(y)$ for a given value of x
- ref.
  - 최대우도법(Maximum Likelihood Estimation) 소개 – YouTube
  - Maximum Likelihood, clearly explained!!! - YouTube

Interpreting Coefficients of Logistic Regression

❖ Study Hour Example

- $\beta_0 = -3.13952411$
- $\beta_1 = 1.14860386$

❖ For a student who studies 3.0 hours, what is the probability of passing the exam ?

- We know

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1)}}$$

- Compare it with the output of predict_proba()

```
model3 = linear_model.LogisticRegression()

model3.fit(X,y)

X_test = X
y_test = y

y_pred3 = model3.predict(X_test)

print('Intercept =', model3.intercept_)
for name, coef in zip(predictors, model3.coef_) :
  print(f' {name} : {coef}')


import math
p_3hours = 1/(1+math.exp(-
(model3.intercept_+model3.coef_*3.0)))
print(p_3hours)

print(model3.predict_proba([[3.0]]))
```

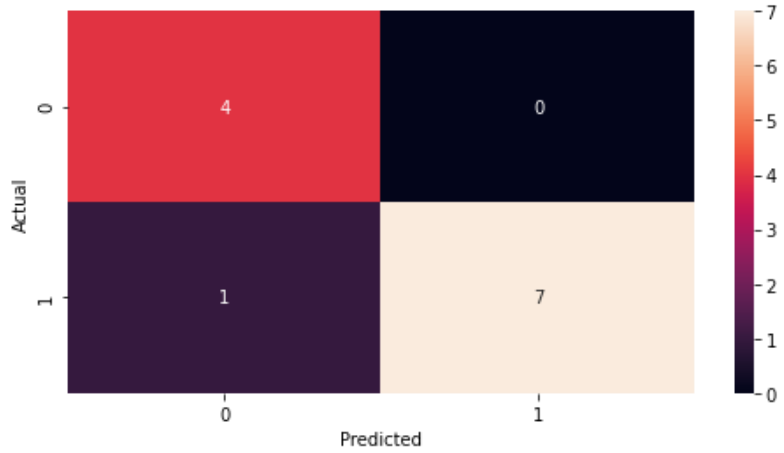
```
Intercept = [-3.13952411]
Hours : [1.14860386]
0.575978821074236 8
[[0.42402118 0.57597882]]
```

Logistic Regression Example

- Virus Infection Data
  - Y : 0 – Not Infected, 1 – Infected
  - X1 – Mask Wearing Score, X2 – Hand Washing Score, X3 – Exercise Score
  - 30 records
- Perform a logistic regression
  - Split data into train data and test data using sklearn.model_selection.train_test_split()
  - Evaluate your model

y	x1	x2	x3
0	3	6	5
0	4	10	4
1	3	5	3
1	3	1	5
1	3	1	5
0	5	11	4
1	3	6	5
1	2	10	4
1	3	2	3
0	4	9	5
0	4	10	6
1	3	3	1
1	4	3	2
1	3	1	2
0	5	11	5
0	5	10	6
0	4	9	5
1	4	7	2
0	5	8	4
1	3	6	3
1	5	6	5
0	3	9	3
1	4	5	2
1	3	5	2
0	2	11	6
1	3	4	5
1	3	5	4
0	5	11	5
0	4	8	4
0	3	8	5

```
import numpy as np
import pandas as pd
from sklearn import linear_model
from sklearn import metrics
from sklearn.model_selection import train_test_split
import seaborn as sn
import matplotlib.pyplot as plt

data = pd.read_csv('https://raw.githubusercontent.com/inetguru/IDS-CB35533/main/virus.csv')
train_data, test_data = train_test_split(data, train_size=0.6, random_state=1)

predictors = ['x1', 'x2', 'x3']
target='y'
X = train_data[predictors]
y = train_data[target]
X_test = test_data[predictors]
y_test = test_data[target]

model = linear_model.LogisticRegression()

model.fit(X,y)

y_pred=model.predict(X_test)

confusion_matrix = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted'])
sn.heatmap(confusion_matrix, annot=True)
plt.show()

print('Accuracy: ',metrics.accuracy_score(y_test, y_pred))
```




```
Accuracy: 0.9166666666666666
```