0. Mounts (Case of using Colaboratory) and Import modules

```
import os
import shutil
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/cont
```

```
os.getcwd()
```

```
'/content/drive/My Drive/AIhack_TH2022/00_Prepare'
```

```
os.chdir('/content/drive/MyDrive/AIhack_TH2022/00_Prepare')
!ls
```

```
answer.csv  Explanation.gslides  Titanic_example.gsheet
AUC.gsheet  sample.ipynb         titanic.zip
Data        test.csv             train.csv
```

1. Import data

Data list

- train.csv (This dataset has Survived information)
- test.csv (This dataset hasn't Survived information)
- answer.csv (This dataset has Survived information of test.csv)

Check the following:

(1) *train.csv* **has** Survived column

(2) *test.csv* **hasn't** Survived column

(3) *answer.csv* **has** Survived column

```
import pandas as pd
```

```
train_data = pd.read_csv("train.csv")
test_data = pd.read_csv("test.csv")
answer_data = pd.read_csv("answer.csv")
```

train data

```
train_data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th	female	38.0	1	0	PC 17599	71

```
train_data.shape # Number of rows and columns
```

```
(722, 12)
```

Definitions of column

Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd (1st = Upper, 2nd = Middle, 3rd = Lower)
sex	Sex	
Age	Age in years	
sibsp	# of siblings / spouses aboard the Titanic	
parch	# of parents / children aboard the Titanic	
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

test data

```
test_data.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	723	2	Gillespie, Mr. William Henry	male	34.0	0	0	12233	13.0000	
1	724	2	Hodges, Mr.	male	50.0	0	0	250643	13.0000	

```
test_data.shape # Number of rows and columns
```

```
(169, 11)
```

Campbell

answer_data

```
answer_data.head()
```

	PassengerId	Survived
0	723	0
1	724	0
2	725	1
3	726	0
4	727	1

```
answer_data.shape # Number of rows and columns
```

```
(169, 2)
```

2. Merge

```
# Set Flag
train_data["train_flag"] = 1
test_data["train_flag"] = 0
test_data["Survived"] = -99 # temporary

# merge
data = pd.concat([train_data, test_data], axis = 0)
```

```
data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7
1	2	1	1	Cumings, Mrs. John Bradley (Florence	female	38.0	1	0	PC 17599	71

```
data.shape
```

```
(891, 13)
```

3. Seeing data

```
# check columns
pd.DataFrame(data.columns)
```

	0
0	PassengerId
1	Survived
2	Pclass
3	Name
4	Sex
5	Age
6	SibSp
7	Parch
8	Ticket
9	Fare
10	Cabin
11	Embarked
12	train_flag

```
# check type of columns
pd.DataFrame(data.dtypes)
# another function -> data.info()
```

	0
PassengerId	int64
Survived	int64
Pclass	int64
Name	object
Sex	object
Age	float64
SibSp	int64
Parch	int64
Ticket	object
Fare	float64
Cabin	object
Embarked	object
train_flag	int64

```
# check to int and float data
data.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	89
mean	446.000000	-18.461279	2.308642	29.699118	0.523008	0.381594	3
std	257.353842	38.989791	0.836071	14.526497	1.102743	0.806057	4
min	1.000000	-99.000000	1.000000	0.420000	0.000000	0.000000	
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	1
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	3
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	51

4. Make a model (A very simple example)

```
# conversion to dummy data about "Sex" and "Embarked" colmuns
# If you want more information, look up "get_dummies()" in the google engine.
data = pd.get_dummies(data, columns=['Sex', 'Embarked'])
```

data

	PassengerId	Survived	Pclass	Name	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	35.0	0	0	373450	8.0500
...	...	...	...	...	...	...	...	...	...
				Montvila,					

```
from sklearn.model_selection import train_test_split

# Split into train_data and test_data.
train_data = data[data["train_flag"] == 1]
test_data = data[data["train_flag"] == 0]

# Stores variables(= Feature value) used for making AI (Feel free to choose!)
cols = ["Fare", "Sex_female", "Embarked_C", "Embarked_Q", "Embarked_S"]

# Separation of feature values and Survived
df_x = train_data[cols]
df_y = train_data['Survived']

# Split for training / validation
# If you want more information, look up "train_test_split()" in the google engine.
train_x, val_x, train_y, val_y = train_test_split(df_x, df_y, test_size=0.2)


# check of data size
print(train_x.shape)
print(val_x.shape)
print(train_y.shape)
print(val_y.shape)
```

```
(577, 5)
(145, 5)
(577,)
(145,)
```

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(train_x, train_y)
```

```
LogisticRegression()
```

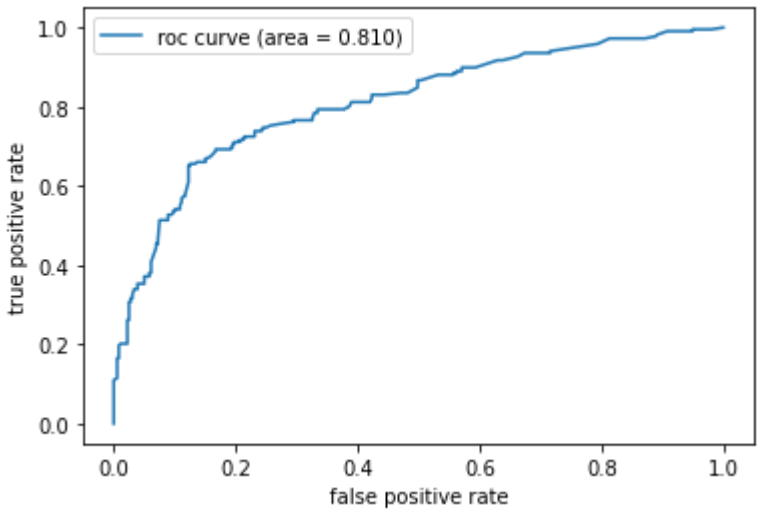
```
# predict(to Train)
y_train_pred = lr.predict_proba(train_x)

# predict(to val)
y_val_pred = lr.predict_proba(val_x)


from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
```

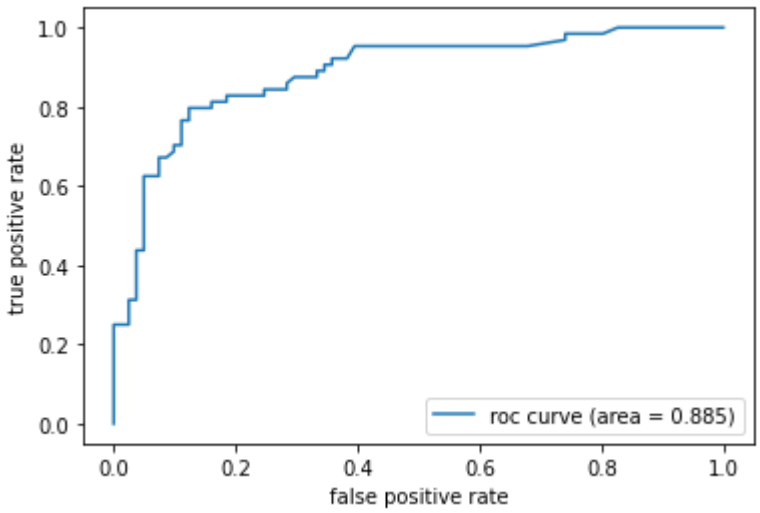
```
# AUC (Train)
y_train_pred = lr.predict_proba(train_x)[:, 1]
fpr, tpr, thresholds = roc_curve(y_true = train_y, y_score = y_train_pred)

plt.plot(fpr, tpr, label='roc curve (area = %0.3f)' % auc(fpr, tpr))
plt.legend()
plt.xlabel('false positive rate')
plt.ylabel('true positive rate')
plt.show()
```




```
# AUC (Val)
y_val_pred = lr.predict_proba(val_x)[:, 1]
fpr, tpr, thresholds = roc_curve(y_true = val_y, y_score = y_val_pred)

plt.plot(fpr, tpr, label='roc curve (area = %0.3f)' % auc(fpr, tpr))
plt.legend()
plt.xlabel('false positive rate')
plt.ylabel('true positive rate')
plt.show()
```




```
test_x = test_data[cols]
```

```
test_y = answer_data['Survived']

# AUC(test data)
y_test_pred = lr.predict_proba(test_x)[:, 1]
fpr, tpr, thresholds = roc_curve(y_true = test_y, y_score = y_test_pred)

plt.plot(fpr, tpr, label='roc curve (area = %0.3f)' % auc(fpr, tpr))
plt.legend()
plt.xlabel('false positive rate')
plt.ylabel('true positive rate')
plt.show()
```