Data Mining with Titanic dataset

Crisp-DM

1. **Business Understanding** เป็นช่วงของการทำความเข้าใจในธุรกิจ ในที่นี้คือ คนที่มี Feature แบบไหนถึงจะมีชีวิตรอด
2. **Data Understanding** เป็นช่วงทำความเข้าใจในข้อมูล โดยเป็นการทำความเข้าใจกับข้อมูลที่จัดเก็บ รวบรวมข้อมูล ศึกษาและทำความคุ้นเคยกับข้อมูล ตลอดจนประเมินคุณภาพของข้อมูลที่ได้มา
3. **Data Preparation Phase** ช่วงเตรียมข้อมูล โดยเตรียมข้อมูลดิบที่จะต้องใช้ในขั้นตอนที่เหลือ ตลอดจนเลือกตัวแปรที่ต้องการมาวิเคราะห์ให้เหมาะสม อีกทั้งแปลงรูปแบบของตัวแปรให้อยู่ในรูปแบบเดียวกัน เพื่อให้ข้อมูลพร้อมสำหรับการนำไปใช้ไปสร้างแบบจำลอง
4. **Model Phase** ช่วงสร้างแบบจำลอง โดยเป็นการคัดเลือก แบบจำลองที่เหมาะสม ปรับปรุงตัวแปร ลักษณะแบบจำลองเพื่อให้ได้ผลลัพธ์ที่ดีที่สุด อาจจะใช้เทคนิคหลายๆเทคนิคเข้ามาช่วยวิเคราะห์ได้ ถ้าจำเป็น ก็สามารถย้อนกลับไปช่วงการเตรียมข้อมูล เพื่อเตรียมข้อมูลให้เหมาะสมกับการสร้างแบบจำลองใหม่ได้
5. **Evaluation Phase** ช่วงประเมินผล โดยเป็นการประเมินแบบจำลองที่ใช้ในการวิเคราะห์ทั้งหมด ประเมินว่าแบบจำลองไหนตอบโจทย์ในขั้นตอนแรกได้ดีที่สุด ตรวจสอบความถูกต้องและสภาพแวดล้อมต่างๆ ตัดสินใจในการนำผลลัพธ์ไปใช้
6. **Deployment Phase** โดยเป็นการนำไปใช้งานจริง รวมถึงนำเสนอตัวอย่างจากการนำไปใช้จริง

Crisp-DM with Titanic dataset

1.1. Business Understanding

ให้แบ่งกลุ่มของคนที่มีอยู่ใน dataset ว่าลักษณะแบบไหนถึงจะมีชีวิตรอด หรือ ไม่มีชีวิตรอด

1.2 Data Understanding

import csv file ซึ่งเป็นเซทของข้อมูล ไททานิค โดยใช้ python tools ในการทำเหมืองข้อมูล พบว่ามีทั้งหมด 891 แถว และ มี 12 คอลัมน์ โดยข้อมูลแต่ละแถวคือ ข้อมูลของผู้โดยสารเรือแต่ละคน และข้อมูลของแต่ละคอลัมน์มีดังต่อไปนี้

1. 'PassengerId' คือ หมายเลขผู้โดยสาร
2. 'Survived' คือ ผู้รอดชีวิต โดย 1 เป็นการรอดชีวิต ส่วน 0 คือผู้เสียชีวิต
3. 'Pclass' คือ สถานะภาพทางเศรษฐกิจและสังคม แบ่งได้เป็น การศึกษา + อาชีพ + รายได้
4. 'Name' คือ ชื่อของผู้โดยสาร
5. 'Sex' คือ เพศ
6. 'Age' คือ อายุ
7. 'SibSp' คือ จำนวนความสัมพันธ์ของผู้โดยสารที่มาด้วยกัน เช่น พี่น้อง หรือ คู่สมรส
8. 'Parch' คือ จำนวนความสัมพันธ์ของผู้โดยสารที่มาด้วยกัน เช่น พ่อ แม่ ลูก
9. 'Ticket' คือ หมายเลขตั๋ว
10. 'Fare' คือ ค่าโดยสาร
11. 'Cabin' คือ หมายเลขห้องโดยสาร
12. 'Embarked' คือ ท่าเรือที่รับผู้โดยสารขึ้นเรือ

รายละเอียดจะเพิ่มเติมให้รูปต่อไป

1. import necessary library

In [1]:
```
# Imports

# pandas
import pandas as pd

# numpy, matplotlib, seaborn
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
import graphviz

%matplotlib inline
```

2. Read In and Explore the Data

In [2]:
```
# get titanic & test csv files as a DataFrame
df = pd.read_csv("./titanic_data_set/train.csv")

# preview the data
df.head(5)
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

3. Data Analysis

In [3]:
```
# columns
print(df.columns, '\n', df.shape)
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
 (891, 12)
```

In [4]:
```
# samples

df.sample(5)
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
831	832	1	2	Richards, Master. George Sibley	male	0.83	1	1	29106	18.7500	NaN	S
334	335	1	1	Frauenthal, Mrs. Henry William (Clara Heinshei...	female	NaN	1	0	PC 17611	133.6500	NaN	S
151	152	1	1	Pears, Mrs. Thomas (Edith Wearne)	female	22.00	1	0	113776	66.6000	C2	S
479	480	1	3	Hirvonen, Miss. Hildur E	female	2.00	0	1	3101298	12.2875	NaN	S
514	515	0	3	Coleff, Mr. Satio	male	24.00	0	0	349209	7.4958	NaN	S

In [5]:
```
# there are some null values that need to be cleaned

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
```

Type of Features

- **Numerical Features:** Age (Continuous), Fare (Continuous), SibSp (Discrete), Parch (Discrete)
- **Categorical Features:** Survived, Sex, Embarked, Pclass
- **Alphanumeric Features:** Ticket, Cabin

```
In [6]: 1 df.describe(include='all')
```

Out[6]:

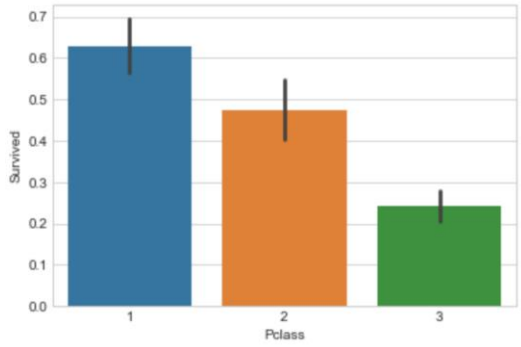
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
count	891.000000	891.000000	891.000000	891	891	714.000000	891.000000	891.000000	891	891.000000	204	889
unique	NaN	NaN	NaN	891	2	NaN	NaN	NaN	681	NaN	147	3
top	NaN	NaN	NaN	Collyer, Mr. Harvey	male	NaN	NaN	NaN	347082	NaN	C23 C25 C27	S
freq	NaN	NaN	NaN	1	577	NaN	NaN	NaN	7	NaN	4	644
mean	446.000000	0.383838	2.308642	NaN	NaN	29.699118	0.523008	0.381594	NaN	32.204208	NaN	NaN
std	257.353842	0.486592	0.836071	NaN	NaN	14.526497	1.102743	0.806057	NaN	49.693429	NaN	NaN
min	1.000000	0.000000	1.000000	NaN	NaN	0.420000	0.000000	0.000000	NaN	0.000000	NaN	NaN
25%	223.500000	0.000000	2.000000	NaN	NaN	20.125000	0.000000	0.000000	NaN	7.910400	NaN	NaN
50%	446.000000	0.000000	3.000000	NaN	NaN	28.000000	0.000000	0.000000	NaN	14.454200	NaN	NaN
75%	668.500000	1.000000	3.000000	NaN	NaN	38.000000	1.000000	0.000000	NaN	31.000000	NaN	NaN
max	891.000000	1.000000	3.000000	NaN	NaN	80.000000	8.000000	6.000000	NaN	512.329200	NaN	NaN

4. Data Visualization

Pclass Feature

```
In [7]: 1 sns.barplot(df['Pclass'], df['Survived'], data=df)
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x10be33208>
```



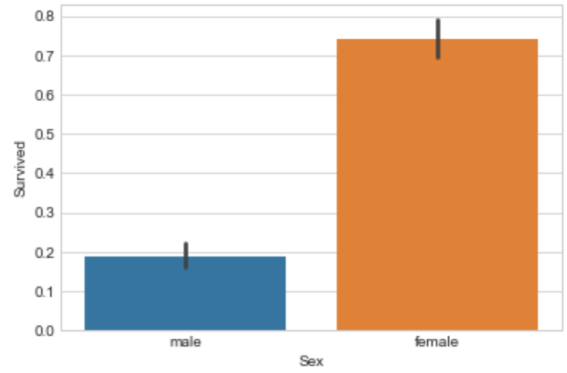
Higher class could survive more.

Pclass Feature ผู้ที่รอดชีวิตส่วนใหญ่คือ กลุ่มคนที่ไม่ได้มาเป็นครอบครัว

Sex Feature

```
In [8]: 1 sns.barplot(df['Sex'], df['Survived'], data=df)
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1067e9898>
```



Female are more likely to survive.

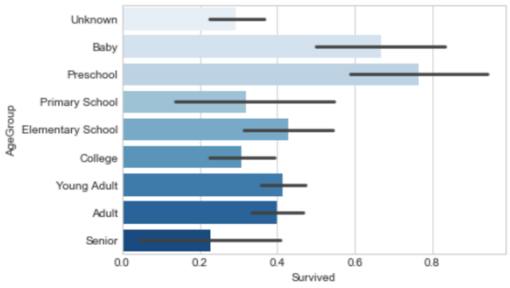
Sex Feature ผู้ที่รอดชีวิตส่วนใหญ่คือ ผู้หญิง

Age Feature

In [9]:
```
print(df[['Age']].info())
df['Age'] = df['Age'].astype(float).fillna(-0.5)
bins = [-1, 0, 3, 6, 12, 18, 23, 35, 60, np.inf]
labels = ['Unknown', 'Baby', 'Preschool', 'Primary School', 'Elementary School', 'College', 'Young Adult', 'Adu
df['AgeGroup'] = pd.cut(df['Age'], bins=bins, labels=labels)

# sns.barplot(df['AgeGroup'], df['Survived'], data=df)
#draw a bar plot of Age vs. survival
sns.barplot(df['Survived'], df['AgeGroup'], data=df, palette='Blues')
plt.show()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 1 columns):
Age    714 non-null float64
dtypes: float64(1)
memory usage: 7.0 KB
None
```



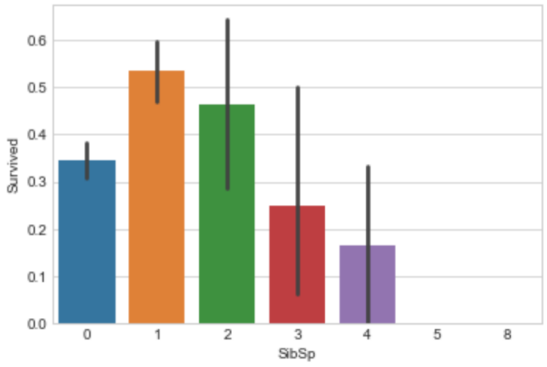
Baby and children are more likely to survive.

Age Feature ผู้ที่รอดชีวิตส่วนใหญ่คือ เด็ก

SibSp Feature

In [10]:
```
1 sns.barplot(df['SibSp'], df['Survived'], data=df)
```

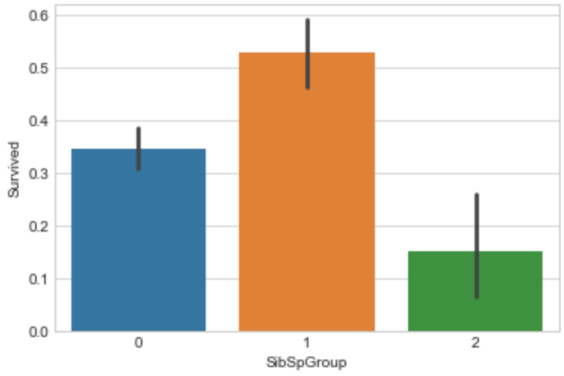
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x10bf1f7f0>



In [11]:
```
1 #map each of the sip groups to a numerical value
2 sipSp_mapping = {0: 0, 1: 1, 2: 1}
3
4 df['SibSpGroup'] = df['SibSp'].map(sipSp_mapping)
5 df['SibSpGroup'] = df['SibSpGroup'].fillna(2)
6 df['SibSpGroup'] = df['SibSpGroup'].astype('int')
```

In [12]:
```
1 sns.barplot(df['SibSpGroup'], df['Survived'], data=df)
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x10d5627b8>

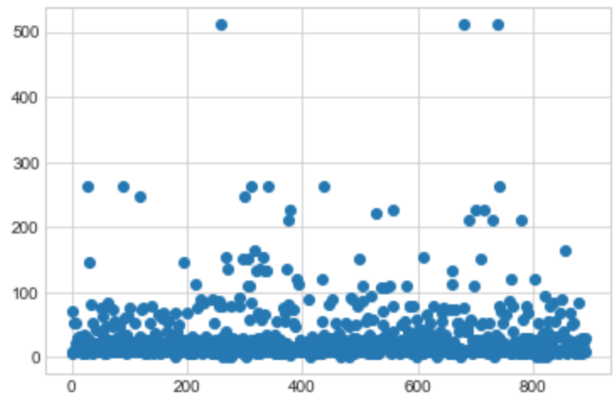


Combine SipSp to groups.

SibSp Feature ผู้ที่รอดชีวิตส่วนใหญ่คือ คนที่มาเป็นคู่จะมีโอกาสรอดสูงกว่า

Fare Feature

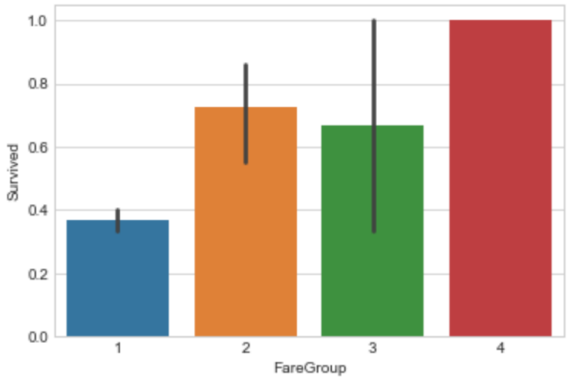
In [14]:
```
1 plt.scatter(df.index, df.Fare)
2 plt.show()
```



In [15]:
```
labels = [1, 2, 3, 4]
df['FareGroup'] = pd.cut(df['Fare'], 4, labels=labels)

sns.barplot(df['FareGroup'], df['Survived'], data=df)
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x10d7ba2b0>



Rich has higher chance to survive.

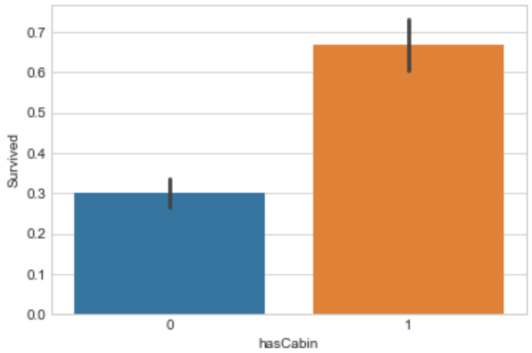
Fare Feature ผู้ที่รอดชีวิตส่วนใหญ่มาจากค่าโดยสารแพงมากสุด

Cabin Feature

In [16]:
```
df['hasCabin'] = df['Cabin'].notnull().astype(int)

sns.barplot(df['hasCabin'], df['Survived'], data=df)
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x10d87d0b8>



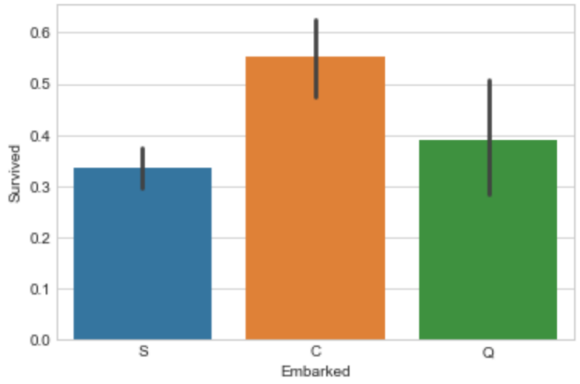
One who has a cabin might mean he is rich.

Cabin Feature ผู้ที่รอดชีวิตส่วนใหญ่คือ ผู้ที่มีหมายเลขห้องโดยสาร

Embark Feature

```
In [17]: 1 sns.barplot(df['Embarked'], df['Survived'], data=df)
```

```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x10d92aac8>
```



Embark Feature ผู้ที่รอดชีวิตส่วนใหญ่มาจากท่าเรือ Cherbourge

1.3 Data Preparation Phase

Cleaning data จากการกำจัดค่า null และจัดให้ข้อมูลอยู่ในรูปของตัวเลข integer เพราะ library python รับแต่ค่า integer

5. Cleaning Data

Find null values

```
In [18]: 1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 16 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            891 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
AgeGroup       891 non-null category
SibSpGroup     891 non-null int64
FareGroup      891 non-null category
hasCabin       891 non-null int64
dtypes: category(2), float64(2), int64(7), object(5)
memory usage: 99.8+ KB
```

Pclass Feature

Do nothing.

Age Feature

Fill null value by predicting from their titles.

In [19]:
```
df['Name'].head(10)
```

Out[19]:
```
0                              Braund, Mr. Owen Harris
1    Cumings, Mrs. John Bradley (Florence Briggs Th...
2                               Heikkinen, Miss. Laina
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                             Allen, Mr. William Henry
5                                     Moran, Mr. James
6                              McCarthy, Mr. Timothy J
7                       Palsson, Master. Gosta Leonard
8    Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
9                  Nasser, Mrs. Nicholas (Adele Achem)
Name: Name, dtype: object
```

Age เติมอายุที่หายไปโดยทำนายจากชื่อนำหน้าว่าควรจะอยู่ในกลุ่มคนอายุเท่าไหร่

In [20]:
```
df['Title'] = df.Name.str.extract(' ([A-Za-z]+)\.', expand=False)
pd.crosstab(df['Title'], df['Sex'])

# some of these are outlier.
```

Out[20]:

Sex	female	male
Title		
Capt	0	1
Col	0	2
Countess	1	0
Don	0	1
Dr	1	6
Jonkheer	0	1
Lady	1	0
Major	0	2
Master	0	40
Miss	182	0
Mlle	2	0
Mme	1	0
Mr	0	517
Mrs	125	0
Ms	1	0
Rev	0	6
Sir	0	1

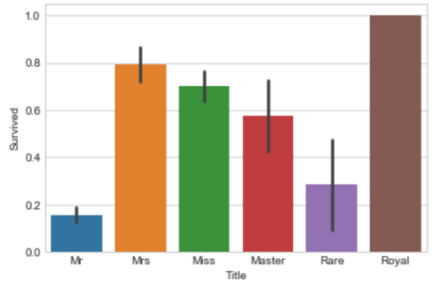
In [21]:
```
#replace various titles with more common names

df['Title'] = df['Title'].replace(['Lady', 'Capt', 'Col', 'Don', 'Dr', 'Major', 'Rev', 'Jonkheer', 'Dona'], 'Ra

df['Title'] = df['Title'].replace(['Countess', 'Lady', 'Sir'], 'Royal')
df['Title'] = df['Title'].replace('Mlle', 'Miss')
df['Title'] = df['Title'].replace('Ms', 'Miss')
df['Title'] = df['Title'].replace('Mme', 'Mrs')

sns.barplot(df['Title'], df['Survived'])
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x10da76860>



In [22]:
```
#map each of the title groups to a numerical value

title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Royal": 5, "Rare": 6}

df['Title'] = df['Title'].map(title_mapping)
df['Title'] = df['Title'].fillna(0)
```

In [23]:
```
# check results
df[['Title']].sample(5)
```

Out[23]:

	Title
456	1
760	1
525	1
164	4
240	2

In [24]:
```
# find age group for each title by mode

age_value = {}
for n in range(6):
    age_value[n+1] = df[df['Title'] == n+1]['AgeGroup'].mode()
    print(age_value[n+1].values)
```

```
[Young Adult]
Categories (9, object): [Unknown < Baby < Preschool < Primary School ... College < Young Adult < Adult < Senior]
[Young Adult]
Categories (9, object): [Unknown < Baby < Preschool < Primary School ... College < Young Adult < Adult < Senior]
[Adult]
Categories (9, object): [Unknown < Baby < Preschool < Primary School ... College < Young Adult < Adult < Senior]
[Baby]
Categories (9, object): [Unknown < Baby < Preschool < Primary School ... College < Young Adult < Adult < Senior]
[Young Adult, Adult]
Categories (9, object): [Unknown < Baby < Preschool < Primary School ... College < Young Adult < Adult < Senior]
[Adult]
Categories (9, object): [Unknown < Baby < Preschool < Primary School ... College < Young Adult < Adult < Senior]
```

In [25]:
```
# age title maping from previous result
age_title_mapping = {1: "Young Adult", 2: "Young Adult", 3: "Adult", 4: "Baby", 5: "Young Adult", 6: "Adult"}
# fill na with previous result
df.AgeGroup = df.AgeGroup.replace("Unknown", df["Title"].map(age_title_mapping))
```

In [53]:
```
df.AgeGroup.head()
```

Out[53]:
```
0    5
1    7
2    6
3    6
4    6
Name: AgeGroup, dtype: int64
```

In [27]:
```
# change agegroup's data type from category to int
age_mapping = {'Baby': 1, 'Preschool': 2, 'Primary School': 3, 'Elementary School': 4, 'College': 5, 'Young Adu
df.AgeGroup = df.AgeGroup.map(age_mapping)
```

Sex กำหนดให้เพศหญิงเป็น 0 และเพศชายเป็น 1

Sex Feature

In [28]:
```
# map Male = 1, Female = 0

df.Sex = (df.Sex.values == 'male').astype(int)
df['Sex'].head()
```

Out[28]:
```
0    1
1    0
2    0
3    0
4    1
Name: Sex, dtype: int64
```

SibSp, Parch ไม่ต้องทำอะไรเพราะเป็นเลขอยู่แล้ว

SibSp Feature

Use SibSpGroup instead.

Parch Feature

Do nothing.

fare feature จัดกลุ่มค่าโดยสารเป็น 4 กลุ่ม

Fare Feature

In [29]:
```
#map Fare values into groups of numerical values

df['FareGroup'] = pd.cut(df['Fare'], 4, labels = [1, 2, 3, 4]).astype(int)
df[['Fare', 'FareGroup']].sample(10)
```

Out[29]:

	Fare	FareGroup
593	7.7500	1
326	6.2375	1
527	221.7792	2
84	10.5000	1
724	53.1000	1
451	19.9667	1
601	7.8958	1
196	7.7500	1
54	61.9792	1
822	0.0000	1

cabin feature จัดกลุ่มว่ามีห้องโดยสารว่ามีหรือไม่มี โดยมีห้องโดยสารมีโอกาสรอดสูงกว่า

Cabin Feature

Use 'hasCabin' instead of 'Cabin'

Embark Feature

In [30]:
```
# map each Embarked value to a numerical value
embarked_mapping = {"S": 1, "C": 2, "Q": 3}
df['Embarked'] = df['Embarked'].map(embarked_mapping)
# fill na with Cherbourg as most people come from there
df['Embarked'] = df['Embarked'].fillna(2).astype(int)
df.sample(5)

# Southampton : 1
# Cherbourg : 2
# Queenstown : 3
```

Drop unneeded columns

In [31]:
```
# check our dataframe
df.sample(5)
```

Out[31]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	AgeGroup	SibSpGroup	FareGroup	hasC
219	220	0	2	Harris, Mr. Walter	1	30.0	0	0	W/C 14208	10.5000	NaN	1	6	0	1	
395	396	0	3	Johansson, Mr. Erik	1	22.0	0	0	350052	7.7958	NaN	1	5	0	1	
446	447	1	2	Mellinger, Miss. Madeleine Violet	0	13.0	0	1	250644	19.5000	NaN	1	4	0	1	
25	26	1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	0	38.0	1	5	347077	31.3875	NaN	1	7	1	1	
757	758	0	2	Bailey, Mr. Percy Andrew	1	18.0	0	0	29108	11.5000	NaN	1	4	0	1	

In [33]:
```
# drop unnecessary columns, these columns won't be useful in analysis and prediction
df = df.drop(['PassengerId','Name','Ticket', 'Fare', 'Cabin', 'Age', 'SibSp', 'Title'], axis=1)
```

In [34]:
```
df.head()
```

Out[34]:

	Survived	Pclass	Sex	Parch	Embarked	AgeGroup	SibSpGroup	FareGroup	hasCabin
0	0	3	1	0	1	5	1	1	0
1	1	1	0	0	2	7	1	1	1
2	1	3	0	0	1	6	0	1	0
3	1	1	0	0	1	6	1	1	1
4	0	3	1	0	1	6	0	1	0

1.4 Model Phase

การแบ่งข้อมูลในการเทรนตั้งแต่คนแรก - คนที่ 700 ส่วนที่เหลือเป็นค่าในการนำไปทดสอบ โดยแบ่งข้อมูลออกเป็น

1. กลุ่ม x_train คือ คอลัมน์ทั้งหมดยกเว้นคอลัมน์Survived ที่จะนำมาเทรน
2. กลุ่ม x_test คือ คอลัมน์ทั้งหมดยกเว้นคอลัมน์Survived ที่จะนำมาทดสอบ
3. y_train คือ คอลัมน์Survivedเท่านั้น ที่จะนำมาเทรน
4. y_test คือ คอลัมน์Survivedเท่านั้น ที่จะนำมาทดสอบ

6. Using ML Models

split data

In [35]:
```
X = df.drop(columns='Survived')
y = df['Survived']

X_train = X.loc[0:701]
X_test = X.loc[701:]
y_train = y.loc[0:701]
y_test = y.loc[701:]
```

เช็คข้อมูลว่าไม่มี null และเป็น integer ทุกคอลัมน์

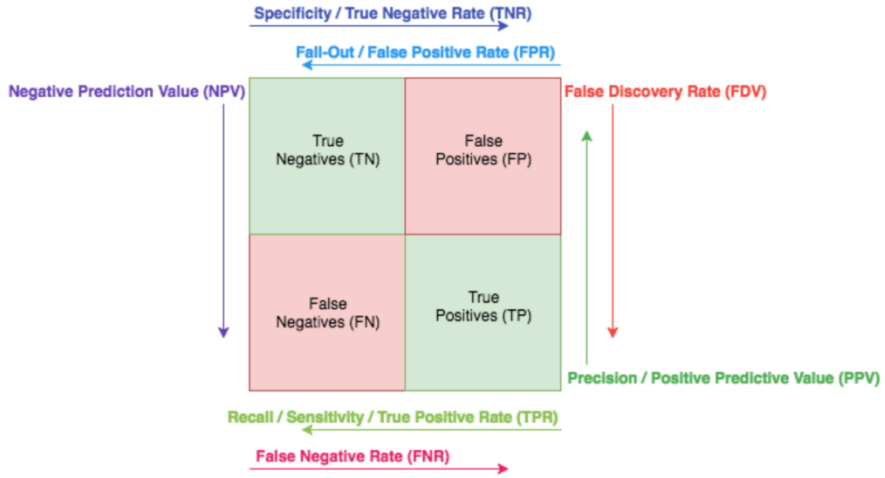
In [40]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 702 entries, 0 to 701
Data columns (total 8 columns):
Pclass        702 non-null int64
Sex           702 non-null int64
Parch         702 non-null int64
Embarked      702 non-null int64
AgeGroup      702 non-null int64
SibSpGroup    702 non-null int64
FareGroup     702 non-null int64
hasCabin      702 non-null int64
dtypes: int64(8)
memory usage: 44.0 KB
```

In [41]:
```
X_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 190 entries, 701 to 890
Data columns (total 8 columns):
Pclass        190 non-null int64
Sex           190 non-null int64
Parch         190 non-null int64
Embarked      190 non-null int64
AgeGroup      190 non-null int64
SibSpGroup    190 non-null int64
FareGroup     190 non-null int64
hasCabin      190 non-null int64
dtypes: int64(8)
memory usage: 12.0 KB
```

Confusion matrix เพื่อเทียบผลการทดลอง



3-Nearest Neighbor classification

with majority vote method

In [46]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
model_knn = knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

In [47]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
print('\nprecision = ' + str((cm[0][0] + cm[1][1])/ (cm[0][1] + cm[1][0] + cm[0][0
```

```
[[77 13]
 [22 43]]

precision = 0.7741935483870968
```

In [48]:
```
y_pred = knn.predict(X_real_test)
cm = confusion_matrix(y_real_test, y_pred)
print(cm)
print('\nprecision = ' + str((cm[0][0] + cm[1][1])/ (cm[0][1] + cm[1][0] + cm[0][0
```

```
[[104  16]
 [ 18  52]]

precision = 0.8210526315789474
```

3-NN ได้ความแม่นยำ 77.42 % จาก training data พอทดสอบข้อมูลใหม่ได้ 82.11 %

Naive Bayes classification

In [50]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
model_gnb = gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
```

In [51]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
print('\nprecision = ' + str((cm[0][0] + cm[1][1])/ (cm[0][1] + cm[1][0] + cm[0][0
```

```
[[76 14]
 [20 45]]

precision = 0.7806451612903226
```

In [52]:
```
y_pred = gnb.predict(X_real_test)
cm = confusion_matrix(y_real_test, y_pred)
print(cm)
print('\nprecision = ' + str((cm[0][0] + cm[1][1])/ (cm[0][1] + cm[1][0] + cm[0][0
```

```
[[103  17]
 [ 18  52]]

precision = 0.8157894736842105
```

Naïve Bayes ได้ความแม่นยำ 78.06 % จาก training data พอทดสอบข้อมูลใหม่ได้ 81.58 %

Random Forest classification

In [53]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
model_rf = rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

In [54]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
print('\nprecision = ' + str((cm[0][0] + cm[1][1])/ (cm[0][1] + cm[1][0] + cm[0][0
```

```
[[74 16]
 [23 42]]

precision = 0.7483870967741936
```

In [55]:
```
y_pred = rf.predict(X_real_test)
cm = confusion_matrix(y_real_test, y_pred)
print(cm)
print('\nprecision = ' + str((cm[0][0] + cm[1][1])/ (cm[0][1] + cm[1][0] + cm[0][0
```

```
[[100  20]
 [ 16  54]]

precision = 0.8105263157894737
```

Random Forest ได้ความแม่นยำ 74.84 % จาก training data พอทดสอบข้อมูลใหม่ได้ 81.05 %

หลังจากใช้ 3NN, Naïve Bayes, Random forest พบว่า 3NN มีความแม่นยำสูงที่สุดที่ 82.11% ทีนี้ลองใช้ 10 fold-cv ดูบ้าง

7. Using 10-Fold-CV

In [49]:
```
# Necessary imports:
from sklearn.cross_validation import cross_val_predict
```

Naive Bayes classification

In [72]:
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model_gnb, predictors, target, cv=10)
print(scores)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
[0.77464789 0.67605634 0.78873239 0.75714286 0.77142857 0.84285714
 0.71428571 0.72857143 0.85714286 0.7826087 ]
Accuracy: 0.77 (+/- 0.10)
```

In [73]:
```
predicted = cross_val_predict(model_gnb, X_real_test, y_real_test)
cm = confusion_matrix(y_real_test, predicted)
print(cm)
print('\nprecision = ' + str((cm[0][0] + cm[1][1])/ (cm[0][1] + cm[1][0] + cm[0][0
```

```
[[119   1]
 [ 61   9]]

precision = 0.6736842105263158
```

3-Nearest Neighbor classification

In [64]:
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model_knn, predictors, target, cv=10)
print(scores)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
[0.76056338 0.76056338 0.8028169  0.74285714 0.75714286 0.84285714
 0.78571429 0.77142857 0.77142857 0.75362319]
Accuracy: 0.77 (+/- 0.06)
```

In [71]:
```
predicted = cross_val_predict(model_knn, X_real_test, y_real_test)
cm = confusion_matrix(y_real_test, predicted)
print(cm)
print('\nprecision = ' + str((cm[0][0] + cm[1][1])/ (cm[0][1] + cm[1][0] + cm[0][0
```

```
[[104  16]
 [ 24  46]]

precision = 0.7894736842105263
```

Random Forest classification

In [126]:
```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model_rf, predictors, target, cv=10)
print(scores)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
[0.69014085 0.77464789 0.74285714 0.74285714 0.82857143 0.88571429
 0.81428571 0.77142857 0.82857143 0.73913043]
Accuracy: 0.78 (+/- 0.11)
```

In [127]:
```
predicted = cross_val_predict(model_rf, X_real_test, y_real_test)
cm = confusion_matrix(y_real_test, predicted)
print(cm)
print('\nprecision = ' + str((cm[0][0] + cm[1][1])/ (cm[0][1] + cm[1][0] + cm[0][0] + cm[1][1])))
```

```
[[109  11]
 [ 20  51]]

precision = 0.837696335078534
```

หลังจากใช้ 3NN, Naïve Bayes, Random forest พบว่า Random forest มีความแม่นยำสูงที่สุดที่ 83.77%

5. **Evaluation Phase** หลังจากการใช้โมเดล พบว่าการแบ่งข้อมูลธรรมดามีความแม่นยำสูงกว่าแบบ 10 fold cv ประมาณ 10% แต่อาจจะเกิด over fitting เราจึงเลือกใช้การแบ่งข้อมูลแบบ 10 fold cv โดยโมเดลที่ให้ค่าความแม่นยำสูงสุดคือ Random Forest classification ที่ 83.77 จึงสรุปได้ว่าควรใช้ Random forest แบบ 10fold-cv

6. **Deployment Phase** งานฉบับนี้ยังไม่ถึงช่วงนี้

References

http://digi.library.tu.ac.th/thesis/it/0729/03chapter2.pdf

https://www.kaggle.com/omarelgabry/a-journey-through-titanic?scriptVersionId=447802/notebook

http://scikit-learn.org/stable/modules/naive_bayes.html

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

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://www.sanyamkapoor.com/machine-learning/confusion-matrix-visualization/

https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.str.extract.html

http://www.ritchieng.com/machinelearning-one-hot-encoding/

https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html

http://scikit-learn.org/stable/modules/cross_validation.html

https://stackoverflow.com/questions/41458834/how-is-scikit-learn-cross-val-predict-accuracy-score-calculated