
Kaggle - Titanic Team B

김민규, 김민주, 윤희준



Project Proposal

Topic | Kaggle Titanic Challenge

Description | 타이타닉에 탑승한 사람들의 신상정보를 활용하여
승선한 사람들의 생존여부를 예측하는 모델을 생성

인터넷에 올라와있는 **Reference**를 활용하여 파이썬을
이용하여 데이터를 다루는 법을 공부

**Expected
Duration** | 4 weeks

Team Member | 김민규, 김민주, 윤희준

Before Start

Planning
Finding reference

Week 6

Exploratory data analysis

- Survival rate, Sex, Age, Embarked, Family, and Cabin
- Relationship between

Week 8

Building machine learning
• model and prediction

Week 5

Present progress
Check DataSet

- Null data

Week 7

Present progress
Feature Engineering

- Fill Null data
- Change datatype
 - One hot encoding

Dataset Check

- `isnull()` function을 이용한 null data check
- Target의 distribution 확인 후 모델 적용 여부 확인

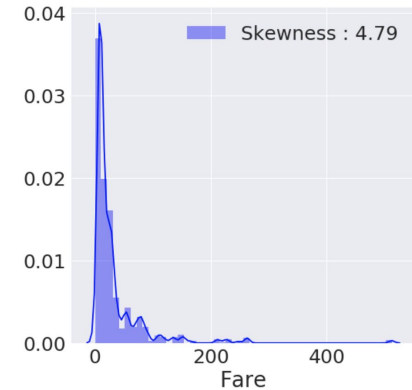
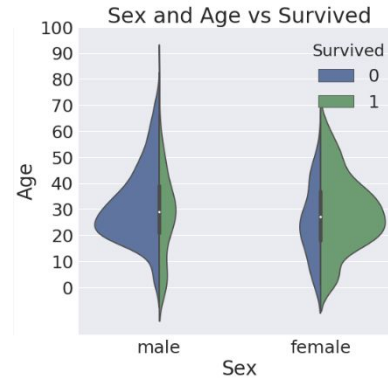
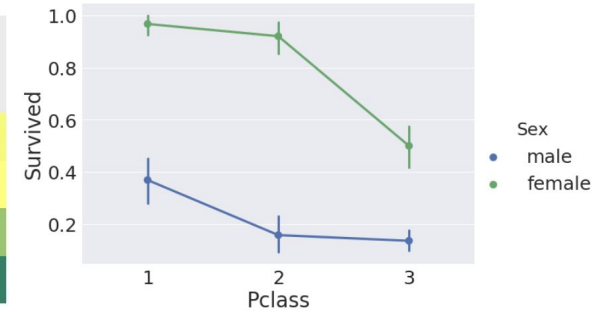
```
for col in df_train.columns:
    msg = 'column: {:>10}|t Percent of NaN value: {:.2f}%'.format(col, 100*(df_train[col].isnull().sum()/df_train[col].shape[0]))
    print(msg)
```

```
column: PassengerId|t Percent of NaN value: 0.00%
column:   Survived|t Percent of NaN value: 0.00%
column:    Pclass|t Percent of NaN value: 0.00%
column:      Name|t Percent of NaN value: 0.00%
column:       Sex|t Percent of NaN value: 0.00%
column:      Age|t Percent of NaN value: 19.87%
column:   SibSp|t Percent of NaN value: 0.00%
column:   Parch|t Percent of NaN value: 0.00%
column:  Ticket|t Percent of NaN value: 0.00%
column:     Fare|t Percent of NaN value: 0.00%
column:    Cabin|t Percent of NaN value: 77.10%
column: Embarked|t Percent of NaN value: 0.22%
```

Exploratory Data Analysis

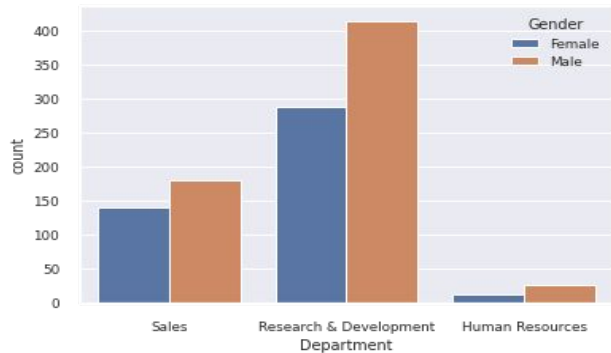
- Data Visualization을 통한 Data Analysis 과정
- Reference에서는 Table, CrossTab, Histogram, Pie Chart, Bar Graph, Factor Plot, Violin Plot 등을 이용

Survived	0	1	All
Pclass			
1	80	136	216
2	97	87	184
3	372	119	491
All	549	342	891

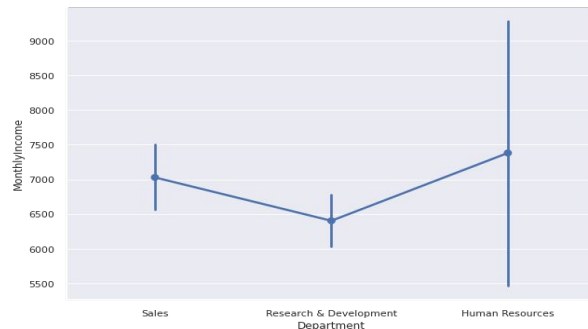


Exploratory Data Analysis

- **Table** - 정확한 값을 보여줄 수 있음, 개별 값 비교 조회에 능함, 한번에 여러 측정단위를 사용할때 보기 편함
- **Histogram**- 단일 변수의 분포 및 관계 비교에 용이함
- **CrossTab**- 다수의 변수의 관계성 파악에 용이
- **Violin Plot**-데이터 양이 많고 데이터 분포의 밀도를 시각화하는데 많이 쓰임
- **Scatter Plot**- 각 데이터의 정확한 값과 전체적인 분포를 시각화하기 용이함



IBM Employee Dataset



```
sns.countplot('Department', hue='Gender', data=df_train)  
plt.show()
```

```
[14] sns.factorplot('Department', 'MonthlyIncome', data=df_train,  
                size=6, aspect=1.5)
```

Feature Engineering

Definition: Feature Engineering은 **모델 정확도를 높이기 위해서** 주어진 데이터로 예측 모델의 문제를 **잘 표현할 수 있는 features로 변형시키는 과정**이다.

= 머신러닝 알고리즘을 작동하기 위해 데이터의 도메인 지식을 활용해 feature를 만드는 과정

Process: Brainstorm features -> Devise features -> Select features
-> Evaluate models -> Revise features -> 이 과정을 반복

Techniques: Imputation, Outliers, Binning, Log Transform, Mapping, One-hot encoding, Grouping Operations, Feature Split, Scaling, Extracting Data...

- **Techniques of Feature Engineering we learned from Titanic Project**

- 3.1 Fill Null Data -> Extracting Data

- 3.2 Change Age(continuous to categorical)-> Binning, Grouping Operations

- 3.3 Change Initial, Embarked and Sex (string to numerical) -> Mapping

Feature Engineering: 3.1 Fill Null data (3.1.1 Fill Null Age using title)

Titanic Project

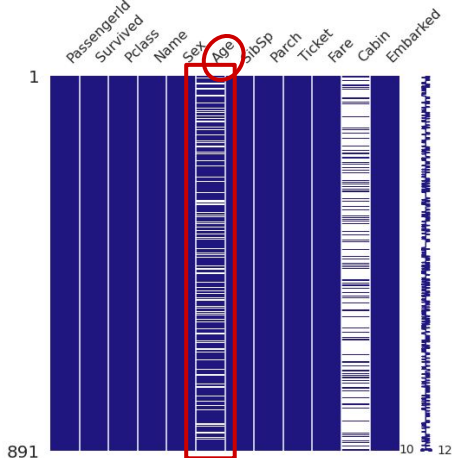
- Age의 Null Data 177개 존재
- 그래프 빈칸으로 Age의 Null Data 존재확인

```
[ ] df_train['Age'].isnull().sum()
```

177

```
msno.matrix(df=df_train.iloc[:, :], figsize=(8, 8), color=(0.1, 0.1, 0.5))
```

<matplotlib.axes._subplots.AxesSubplot at 0x7ff78612a8d8>



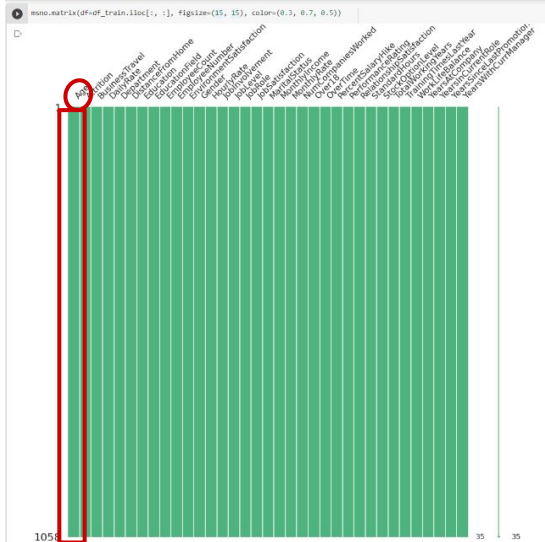
VS

IBM Employee Dataset

- Age의 Null Data 없음
- 그래프 빈칸이 없어 Age의 Null Data 없음을 확인

```
[4] df_train['Age'].isnull().sum()
```

0



Feature Engineering: 3.1 Fill Null data (3.1.1 Fill Null Age using title)

Titanic Project

- Age의 Null data 정리하기 - **keypoint!!**
- 각 이름 sample들의 title을 추출
- **extract('([A-Za-z]+)\.')**
: 정규표현식 사용 (특정 문자를 원하는 문자로 표현)
- **replace**: 다른값으로 치환 (ex. Mile-> Miss, Mme-> Miss 등)
- **loc** (location)

Ex. [:, :] -> 전체

[:1, :] -> 두번째 row까지

[1:3, :] -> 1부터 3까지 row가져와라

```
df_train['Name'].str
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
...
886  Montvila, Rev. Juozas
887  Graham, Miss. Margaret Edith
888  Johnston, Miss. Catherine Helen "Carrie"
889  Behr, Mr. Karl Howell
890  Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

```
[ ] df_train['Initial'] = df_train.Name.str.extract('([A-Za-z]+)\.')
```

```
df_test['Initial'] = df_test.Name.str.extract('([A-Za-z]+)\.')
```

extract('([A-Za-z]+)\.') <- 요기 괄호 안에 정규표현식 -> 의미: 대문자, 소문자 단어가 한개이상이고 점이 붙은 것 ex) Mr., Mrs. Miss.

```
[ ] pd.crosstab(df_train['Initial'], df_train['Sex']).T.style.background_gradient(cmap='summer_r')
```

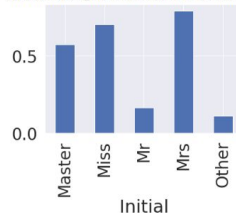
Initial	Capt	Col	Countess	Don	Dr	Jonkheer	Lady	Major	Master	Miss	Mlle	Mme	Mr	Mrs	Ms	Rev	Sir
Sex																	
female	0	0	1	0	1	0	1	0	0	182	2	1	0	125	1	0	0
male	1	2	0	1	6	1	0	2	40	0	0	0	517	0	0	6	

```
[ ] df_train['Initial'].replace(['Mlle','Mme','Ms','Dr','Major','Lady','Countess','Jonkheer','Col','Rev','Capt','Sir','Don'],
                               ['Miss','Miss','Miss','Mr','Mr','Mrs','Mrs','Other','Other','Other','Mr','Mr','Mr'],inplace=True)
df_test['Initial'].replace(['Mlle','Mme','Ms','Dr','Major','Lady','Countess','Jonkheer','Col','Rev','Capt','Sir','Don'],
                           ['Miss','Miss','Miss','Mr','Mr','Mrs','Mrs','Other','Other','Other','Mr','Mr','Mr'],inplace=True)
```

다른값으로 치환한다. Mile-> Miss / Mme-> Miss 등등

```
[ ] df_train.groupby('Initial')['Survived'].mean().plot.bar()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f8b5eb3c590>



```
df_train.loc[(df_train.Age.isnull())&(df_train.Initial=='Mr'),'Age'] = 33
df_train.loc[(df_train.Age.isnull())&(df_train.Initial=='Mrs'),'Age'] = 36
df_train.loc[(df_train.Age.isnull())&(df_train.Initial=='Master'),'Age'] = 5
df_train.loc[(df_train.Age.isnull())&(df_train.Initial=='Miss'),'Age'] = 22
df_train.loc[(df_train.Age.isnull())&(df_train.Initial=='Other'),'Age'] = 46

df_test.loc[(df_test.Age.isnull())&(df_test.Initial=='Mr'),'Age'] = 33
df_test.loc[(df_test.Age.isnull())&(df_test.Initial=='Mrs'),'Age'] = 36
df_test.loc[(df_test.Age.isnull())&(df_test.Initial=='Master'),'Age'] = 5
df_test.loc[(df_test.Age.isnull())&(df_test.Initial=='Miss'),'Age'] = 22
df_test.loc[(df_test.Age.isnull())&(df_test.Initial=='Other'),'Age'] = 46
```

Feature Engineering: 3.1 Fill Null data (3.1.2 Fill Null in Embarked)

Titanic Project

- Embarked의 null data 2개 존재
-> S 에서 가장 많은 탑승객이 있었으므로,
간단하게 Null 을 S로 채움.
- dataframe 의 fillna method 를 이용하면
쉽게 채울 수 있고, 여기서 inplace=True 로
하면 df_train 에 fillna 가 실제로 적용됨

```
[ ] df_train['Embarked'].isnull().sum()
```

2

```
[ ] df_train.shape
```

(891, 14)

```
[11] print('Embarked has ', sum(df_train['Embarked'].isnull()), ' Null values')
```

Embarked has 0 Null values

```
[12] df_train['Embarked'].fillna('S', inplace=True)
```

EDA했을때 S가 제일 많아서 S로 채움.

- The **fillna()** method replaces the NULL values with a specified value.
- Using the **inplace=True** keyword in a pandas method changes the default behaviour
such that the operation on the dataframe doesn't return anything, it instead 'modifies the underlying data

IBM Employee Dataset

- Null Data 뜨는게 거의 없음
- Null Data 정리가 거의 필요없어서 초보자가
연습할 때 이 파트를 공부하기에 썩 좋진 않았던
데이터였지만, 실제 분석에 사용할 때는 null이
거의 없으니 정확도 높은 좋은 라이브러리인듯

```
[34] df_train['Over18'].isnull().sum()
```

0

```
[35] df_train['Education'].isnull().sum()
```

0

```
[36] df_train['JobLevel'].isnull().sum()
```

0

```
[38] df_train['JobRole'].isnull().sum()
```

0

```
[39] df_train['Gender'].isnull().sum()
```

0

VS

Feature Engineering: 3.2 Change Age(continuous to categorical)

Titanic Project

```
[ ] df_train['Age_cat'] = 0
df_train.loc[df_train['Age'] < 10, 'Age_cat'] = 0
df_train.loc[(10 <= df_train['Age']) & (df_train['Age'] < 20), 'Age_cat'] = 1
df_train.loc[(20 <= df_train['Age']) & (df_train['Age'] < 30), 'Age_cat'] = 2
df_train.loc[(30 <= df_train['Age']) & (df_train['Age'] < 40), 'Age_cat'] = 3
df_train.loc[(40 <= df_train['Age']) & (df_train['Age'] < 50), 'Age_cat'] = 4
df_train.loc[(50 <= df_train['Age']) & (df_train['Age'] < 60), 'Age_cat'] = 5
df_train.loc[(60 <= df_train['Age']) & (df_train['Age'] < 70), 'Age_cat'] = 6
df_train.loc[70 <= df_train['Age'], 'Age_cat'] = 7

df_test['Age_cat'] = 0
df_test.loc[df_test['Age'] < 10, 'Age_cat'] = 0
df_test.loc[(10 <= df_test['Age']) & (df_test['Age'] < 20), 'Age_cat'] = 1
df_test.loc[(20 <= df_test['Age']) & (df_test['Age'] < 30), 'Age_cat'] = 2
df_test.loc[(30 <= df_test['Age']) & (df_test['Age'] < 40), 'Age_cat'] = 3
df_test.loc[(40 <= df_test['Age']) & (df_test['Age'] < 50), 'Age_cat'] = 4
df_test.loc[(50 <= df_test['Age']) & (df_test['Age'] < 60), 'Age_cat'] = 5
df_test.loc[(60 <= df_test['Age']) & (df_test['Age'] < 70), 'Age_cat'] = 6
df_test.loc[70 <= df_test['Age'], 'Age_cat'] = 7
```

VS

IBM Employee Dataset

```
[40] print("Maximum size of Age: ", df_train['Age'].max())
      print("Minimum size of Age: ", df_train['Age'].min())
```

Maximum size of Age: 60
Minimum size of Age: 18

```
[41] df_train['Age_cat'] = 0
df_train.loc[df_train['Age'] < 20, 'Age_cat'] = 0
df_train.loc[(20 <= df_train['Age']) & (df_train['Age'] < 30), 'Age_cat'] = 1
df_train.loc[(30 <= df_train['Age']) & (df_train['Age'] < 40), 'Age_cat'] = 2
df_train.loc[(40 <= df_train['Age']) & (df_train['Age'] < 50), 'Age_cat'] = 3
df_train.loc[(50 <= df_train['Age']) & (df_train['Age'] < 60), 'Age_cat'] = 4
df_train.loc[60 <= df_train['Age'], 'Age_cat'] = 5

df_test['Age_cat'] = 0
df_test.loc[df_test['Age'] < 20, 'Age_cat'] = 0
df_test.loc[(20 <= df_test['Age']) & (df_test['Age'] < 30), 'Age_cat'] = 1
df_test.loc[(30 <= df_test['Age']) & (df_test['Age'] < 40), 'Age_cat'] = 2
df_test.loc[(40 <= df_test['Age']) & (df_test['Age'] < 50), 'Age_cat'] = 3
df_test.loc[(50 <= df_test['Age']) & (df_test['Age'] < 60), 'Age_cat'] = 4
df_test.loc[60 <= df_test['Age'], 'Age_cat'] = 5
```

- dataframe 의 indexing 방법: **loc** 를 사용해 직접해주기 or 함수사용

Feature Engineering: 3.2 Change Age(continuous to categorical)

Titanic Project

```
[ ] def category_age(x):  
    if x < 10:  
        return 0  
    elif x < 20:  
        return 1  
    elif x < 30:  
        return 2  
    elif x < 40:  
        return 3  
    elif x < 50:  
        return 4  
    elif x < 60:  
        return 5  
    elif x < 70:  
        return 6  
    else:  
        return 7
```

The **elif** is short for **else if**. It allows us to **check for multiple expressions**.

If the condition for **if** is **False**, it checks the condition of the next **elif** block and so on.

If all the conditions are **False**, the body of **else** is executed.

```
[ ] df_train['Age_cat_2'] = df_train['Age'].apply(category_age)
```

```
df_train['Age'].apply(category_age)  
0      2  
1      3  
2      2  
3      3  
4      3  
..  
886    2  
887    1  
888    2  
889    2  
890    3  
Name: Age, Length: 891, dtype: int64
```

VS

IBM Employee Dataset

```
[42] def category_age(x):  
    if x < 20:  
        return 0  
    elif x < 30:  
        return 1  
    elif x < 40:  
        return 2  
    elif x < 50:  
        return 3  
    elif x < 60:  
        return 4  
    else:  
        return 5
```

```
[43] df_train['Age_cat_2'] = df_train['Age'].apply(category_age)
```

```
[44] df_train['Age'].apply(category_age)
```

```
0      3  
1      3  
2      2  
3      2  
4      1  
..  
1053   4  
1054   3  
1055   2  
1056   1  
1057   1  
Name: Age, Length: 1058, dtype: int64
```

- dataframe 의 indexing 방법: loc 를 사용해 직접해주기 or 함수사용

Feature Engineering - Correlation

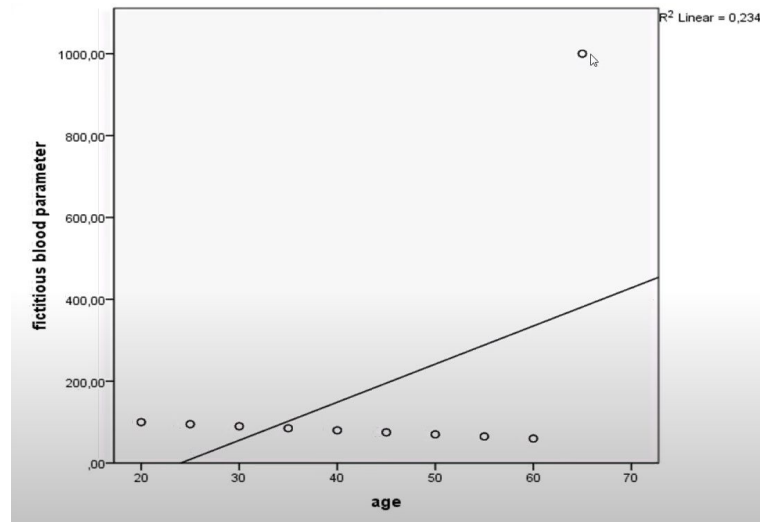
Pearson

- Requires normal distribution (정규성의 과정)

Spearman

- Lack of normal distribution
 - based on data's ranks (서열척도)
- Less sensitive to outliers in data (이상점에 덜 민감)

Pearson correlation과 Spearman correlation이 많이 다르다면 Pearson correlation에 큰 영향을 미치는 이상점이 데이터에 포함되어 있을 가능성이 높다



Feature Engineering - Correlation

Pearson

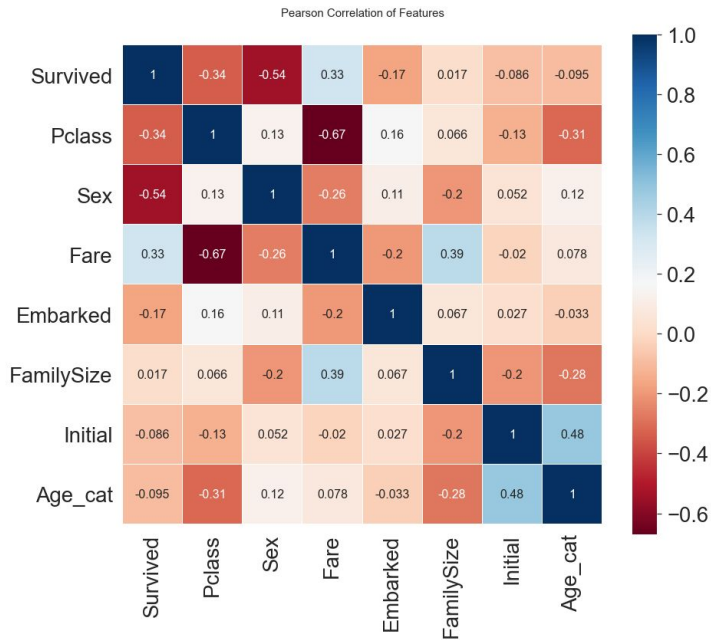
```
#데이터 가져오기
heatmap_data = df_train[['Survived', 'Pclass', 'Sex', 'Fare', 'Embarked', 'FamilySize', 'Initial', 'Age_cat']]
```

```
#색 정하기
colormap = plt.cm.RdBu
```

```
#크기 정하기
plt.figure(figsize=(14, 12))
```

```
#테이블 타이틀 정하기
plt.title('Pearson Correlation of Features', y=1.05, size=15)
```

```
#데이터 type 을 float 으로 바꾼뒤 correlation 찾기
sns.heatmap(heatmap_data.astype(float).corr(method="pearson"), linewidths=0.1, vmax=1.0,
            square=True, cmap=colormap, linecolor='white', annot=True, annot_kws={"size": 16})
```



Feature Engineering - Correlation

Spearman

#데이터 가져오기

```
heatmap_data = df_train[['Survived', 'Pclass', 'Sex', 'Fare', 'Embarked', 'FamilySize', 'Initial', 'Age_cat']]
```

#색 정하기

```
colormap = plt.cm.RdBu
```

#크기 정하기

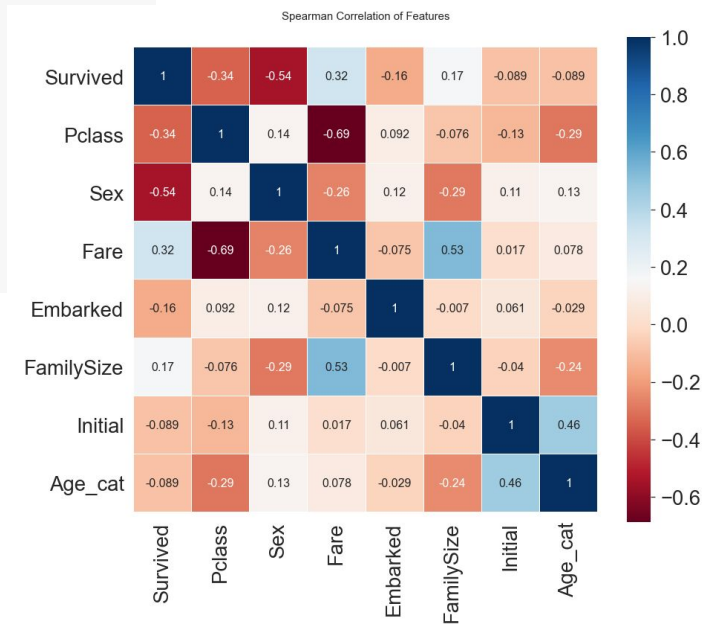
```
plt.figure(figsize=(14, 12))
```

#테이블 타이틀 정하기

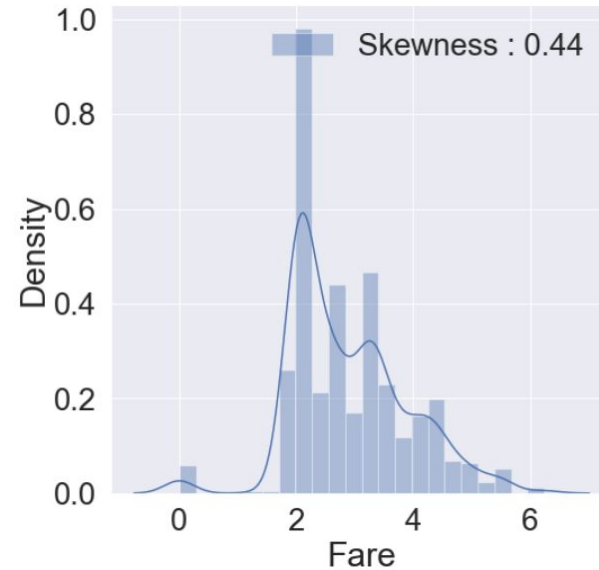
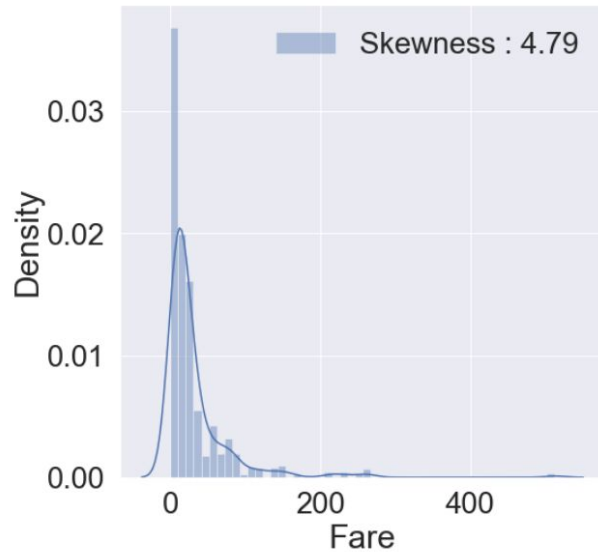
```
plt.title('Pearson Correlation of Features', y=1.05, size=15)
```

#데이터 type 을 float 으로 바꾼뒤 correlation 찾기

```
sns.heatmap(heatmap_data.astype(float).corr(method="spearman"), linewidths=0.1, vmax=1.0, square=True, cmap=colormap, linecolor='white', annot=True, annot_kws={"size": 16})
```



Feature Engineering - Correlation



```
df_train['Fare'] = df_train['Fare'].map(lambda i: np.log(i) if i > 0 else 0)
```

Next Step

Building machine learning 과정 진행

다른 kaggle들을 이용한 여러 종류의 데이터셋을 이용한 코드들 공부

<https://www.kaggle.com/code/m2skills/datasets-and-tutorial-kernels-for-beginners/notebook>

Dataset Check

Exploratory Data Analysis


Feature Engineering

4.1/4.2

```
#importing all the required ML packages
from sklearn.ensemble import RandomForestClassifier # 유명한 randomforestclassifier 입니다.
from sklearn import metrics # 모델의 평가를 위해서 씁니다
from sklearn.model_selection import train_test_split # training set을 원개 나눠주는 함수입니다.
```

```
X_train = df_train.drop('Survived', axis=1).values
target_label = df_train['Survived'].values
X_test = df_test.values
```

train 70 : test 30



```
X_tr, X_vld, y_tr, y_vld = train_test_split(X_train, target_label, test_size=0.3, random_state=2018)
```

Sklearn [train_test_split](#)

- 데이터셋이 하나 밖에 없을때 그 데이터를 편히 나누기 위해 쓰인다
- Data array 를 두가지로 나눈다: **training** 데이터 & **testing** 데이터
 - 만약 training 과 testing 에 같은 데이터 셋을 쓰면 계산 실수가 나올수있기 때문에
can lead to inaccurate predictions
- 보통 이상적인 비율은 training 80 : testing 20 이다.

4.1/4.2

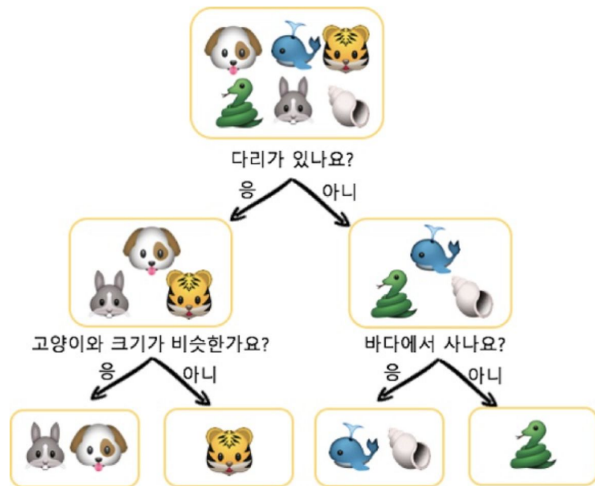
모델 이름

```
model = RandomForestClassifier()  
model.fit(X_tr, y_tr) ← 학습시키기  
prediction = model.predict(X_vld) ← 예측하기
```

```
print('총 {}명 중 {:.2f}% 정확도로 생존을 맞춤'.format(y_vld.shape[0], 100 * metrics.accuracy_score(prediction, y_vld)))
```

총 268명 중 83.21% 정확도로 생존을 맞춤

Decision Tree



- 스무고개처럼 계속 질문을 하여 데이터셋을 구분하는 머신러닝 방식
- 간소하게 보여주고 싶을때는 단일 디시전 트리로 보여주는게 제일 깔끔하게 보여줄수 있음

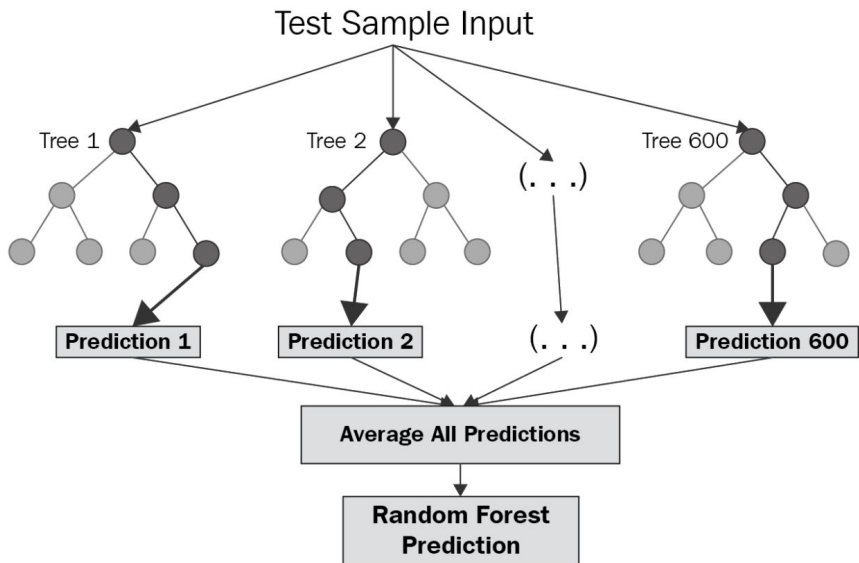
```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_tr, y_tr)
prediction = model.predict(X_vld)
```

```
print('총 {}명 중 {:.2f}% 정확도로 생존을 맞춤'.format(y_vld.shape[0], 100 * metrics.accuracy_score(prediction, y_vld)))
```

총 268명 중 76.49% 정확도로 생존을 맞춤

Random Forest

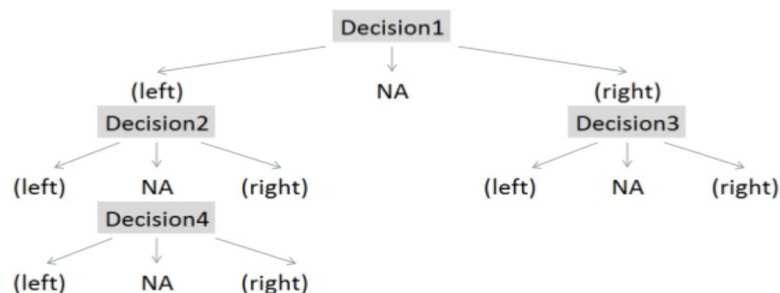
Decision Tree를 여러번 생성하여 그 결과를 평균내는 Machine Learning 방식



Decision Tree의 장점을 그대로 가져가고 일반화 성능을 올린 Model

Gradient Boost Machine

GBM's decision tree structure



Random Forest가 Decision Tree를 병렬식으로 연결하여 평균을 냈다면 GBM은 직렬로 연결하여 모델을 발전시키는 Machine Learning 방식

무작위성이 없는 대신 사전 가지치기를 강력하게 사용

```
from sklearn.ensemble import GradientBoostingClassifier
model = GradientBoostingClassifier()
model.fit(X_tr, y_tr)
prediction = model.predict(X_val)
```

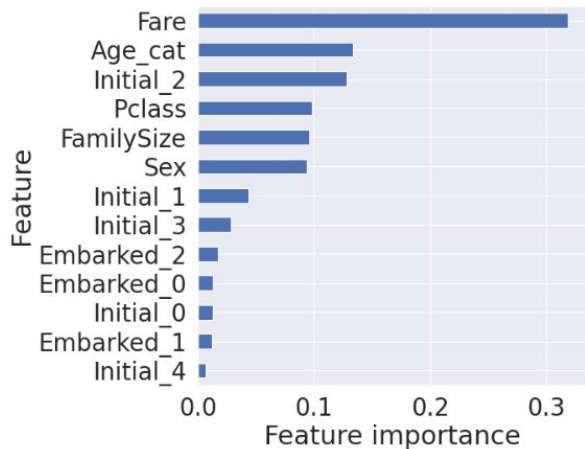
```
print('총 {}명 중 {:.2f}% 정확도로 생존을 맞춤'.format(y_val.shape[0], 100 * metrics.accuracy_score(prediction, y_val)))
```

총 268명 중 85.07% 정확도로 생존을 맞춤

4.3 Feature Importance

- 학습데이터 필터링을 최대한 해서 모델에게 학습하기 좋은 정보를 잘 만들어주는 것이 목표
- 모델이 판단을 내려 예측을 할 때 '어떤 feature에 가장 큰 영향을 받느냐'를 말해주는 것

```
[ ] plt.figure(figsize=(8, 8))
    Series_feat_imp.sort_values(ascending=True).plot.barh()
    plt.xlabel('Feature importance')
    plt.ylabel('Feature')
    plt.show()
```



- 위에 있는 feature부터 중요도가 높은 순위
- 가설도 세울 수 있다. "타이타닉의 생존을 예측은 Fare, Sex, Pclass가 가장 중요한 역할을 하고 있음"
- 중요도 순위 밑에 위치한 Initial은 중요하지 않다고 판단해 drop할수도 있음

4.4 Prediction on Test Set

- 저장 후 제출하는 타이타닉 프로젝트 최종 단계
- Yohan Lee씨 튜토리얼로는 0.75의 정확도
- 다른 **feature**를 쓰면 어떻게 결과가 나올지 등 바뀌서 생존율 예측 정확도를 높일 수 있음
-> 추가, 드랍, null 더 깨끗하게 거르기 - **feature engineering** 방식 바뀌서 등등

Submission and Description	Public Score
my_first_submission.csv a day ago by Min Kyu Kim First Submission	0.74401