머신러닝 모델을 활용한

Titanic 사고 생존자 예측

목차

1. 주제선택 및 프로젝트 개요

가 . 주제선택 : 왜 하필 타이타닉 ?

1. 주제선택 및 프로젝트 개요

나 . 프로젝트 개요

	탑승자	2,224명
	사망자	1,514명
	생존자	710명

어떤 사람이 생존확률이 높았을까 ? 여자 , 아이 , 상류층

주어진 데이터를 바탕으로 생존에 영향을 미친 요읶을 파악

1. 주제선택 및 프로젝트 개요

나 . 프로젝트 개요

데이터

train_data: 승객번호 1~891 번

PassengerId 승객번호	Survived 생존여부	Pclass 객실등급	Name 이름	Sex 성별	Age 나이	SibSp 동승한 형제자매, 배우자	Parch 동승한 부모, 자식	Ticket 티켓번호	Fare 티켓요금	Cabin 객실번호	Embarked 승선지
1	0	3	Braund, M	male	22	1	0	A/5 21171	7.25		S
2	1	1	Cumings, I	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen,	female	26	0	0	STON/O2.	7.925		S
4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
6	0	3	Moran, Mr	male		0	0	330877	8.4583		Q

test_data: 승객번호 892~1309 번

PassengerId 승객번호	Pclass 객실등급	Name 이름	Sex 성별	Age 나이	SibSp 동승한 형제자매, 배우자	Parch 동승한 부모, 자식	Ticket 티켓번호	Fare 티켓요금	Cabin 객실번호	Embarked 승선지
892	3	Kelly, Mr. .	male	34.5	0	0	330911	7.8292		Q
893	3	Wilkes, Mr	female	47	1	0	363272	7		S
894	2	Myles, Mr.	male	62	0	0	240276	9.6875		Q
895	3	Wirz, Mr. A	male	27	0	0	315154	8.6625		S
896	3	Hirvonen,	female	22	1	1	3101298	12.2875		S
897	3	Svensson,	male	14	0	0	7538	9.225		S

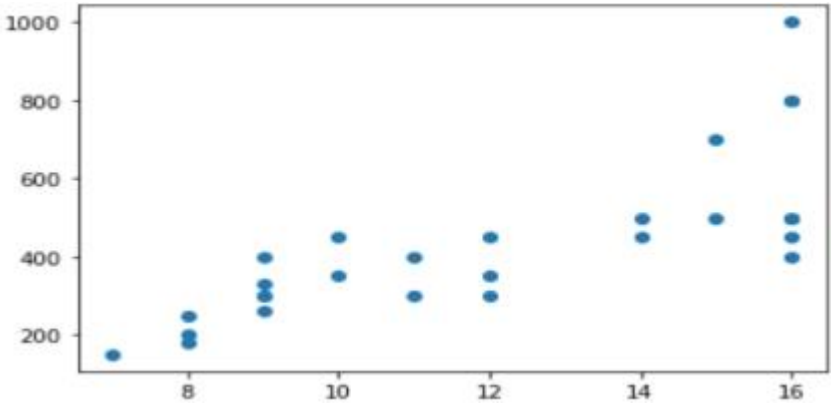
train_data 의 Survived 값 분포를 이용해서 test_data 의 Survived 값을 예측

2. 머신러닝이란 ?

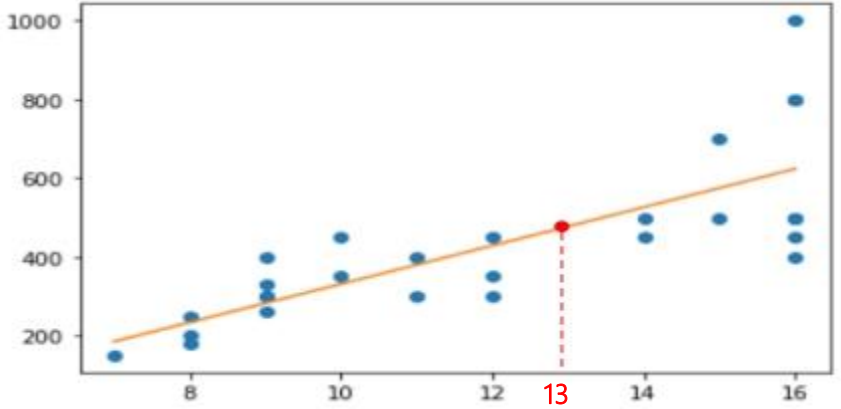
입력받은 데이터 (train_data) 에서 패턴을 찾아낸 후
그 패턴으로 모르는 데이터의 (test_data) 값 (target value) 을 예측하는 알고리즘

ex)

입력 데이터 분포



선형 회귀 후 값예측



Titanic 문제의 경우 test_data 의 Survived 값을 모른다고 가정

또는 앞으로의 사고를 예측한다고 생각

3. 프로젝트 구조

가 . 데이터 수집

나 . 데이터 전처리

다 . 머신러닝 학습 및 예측

라 . 필요시 나 , 다 단계 반복

- 결측값 , 오류값 처리
- 안 쓰는 열 제거 , 새로운 열 생성
- 문자데이터 -> 숫자로 변경
- 숫자데이터를 범주화
- 시각화

3. 프로젝트 구조

사용된 패키지와 모듈

```
# data analysis and wrangling
import pandas as pd
import numpy as np
import random as rnd
```

낯익은 친구들

```
# visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

플롯 (그래프 그리기)

```
# machine learning
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
```

머싞러닝

3. 프로젝트 구조

데이터전처리 (시각화 제외)

안 쓸 변수(Name, PassengerId) 제거

```
train_df = train_df.drop(['Name', 'PassengerId'], axis=1)
test_df = test_df.drop(['Name'], axis=1)
combine=[train_df,test_df]
train_df.shape, test_df.shape
```

((891, 9), (418, 9))

성별(sex) 변수를 숫자 범주형 변수로 변환

```
for dataset in combine:
    dataset['Sex'] = dataset['Sex'].map({'female':1, 'male':0}).astype(int)

train_df.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	0	3	0	22.0	1	0	7.2500	S	1
1	1	1	1	38.0	1	0	71.2833	C	3
2	1	3	1	26.0	0	0	7.9250	S	2
3	1	1	1	35.0	1	0	53.1000	S	3
4	0	3	0	35.0	0	0	8.0500	S	1

AgeBand를 바탕으로 Age를 범주형 변수로 바꿔준 후, AgeBand변수는 제거

```
for dataset in combine:
    dataset.loc[ dataset['Age'] <= 16, 'Age'] = 0
    dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1
    dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <= 48), 'Age'] = 2
    dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3
    dataset.loc[ dataset['Age'] > 64, 'Age']
train_df.head()
```

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title	AgeBand
0	0	3	0	1	1	0	7.2500	S	1	(16.0, 32.0]
1	1	1	1	2	1	0	71.2833	C	3	(32.0, 48.0]
2	1	3	1	1	0	0	7.9250	S	2	(16.0, 32.0]
3	1	1	1	2	1	0	53.1000	S	3	(32.0, 48.0]
4	0	3	0	2	0	0	8.0500	S	1	(32.0, 48.0]

연령(Age) 변수를 범주형 변수로 변환

```
# 임의로 5개 그룹을 지정
# cut이라는 method 이용하면 구간별로 나눌 수 있음
train_df['AgeBand']=pd.cut(train_df['Age'],5)
train_df[['AgeBand', 'Survived']].groupby(['AgeBand'],as_index=False).mean().sort_values(by='AgeBand', ascending=True)
```

	AgeBand	Survived
0	(-0.08, 16.0]	0.550000
1	(16.0, 32.0]	0.337374
2	(32.0, 48.0]	0.412037
3	(48.0, 64.0]	0.434783
4	(64.0, 80.0]	0.090909

SibSp와 Parch를 가족과의 동반여부를 알 수 있는 새로운 변수로 통합

```
for dataset in combine:
    dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1

train_df[['FamilySize', 'Survived']].groupby(['FamilySize'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

	FamilySize	Survived
3	4	0.724138
2	3	0.578431
1	2	0.552795
6	7	0.333333
0	1	0.303538
4	5	0.200000
5	6	0.136364
7	8	0.000000
8	11	0.000000

Age 변수와 Pclass를 곱한 Age*class 변수 생성

```
for dataset in combine: # 나이하고 좌석등급하고 합침
    dataset['Age*Class'] = dataset.Age * dataset.Pclass

train_df.loc[:, ['Age*Class', 'Age', 'Pclass']].head(10) ##loc첫번째 인수는 행이름, 두번째 인수는 열이름(들)
```

	Age*Class	Age	Pclass
0	66	22	3
1	38	38	1
2	78	26	3
3	35	35	1
4	105	35	3
5	75	25	3
6	54	54	1
7	6	2	3
8	81	27	3
9	28	14	2

그밖에도...

승선지 데이터를 범주형으로 변확

티켓요금 결측값 처리 후 범주형으로 변확

승선지(Ebmarked) 변수를 최빈값으로 대체

```
freq_port = train_df.Embarked.dropna().mode()[0]
freq_port
```

'S'

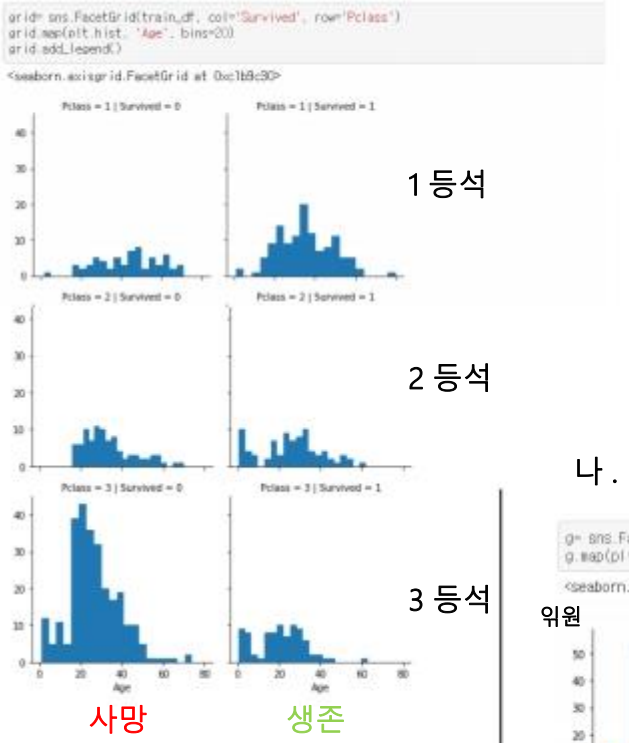
```
for dataset in combine: # 빈 값 2개는 최빈값인 S로 채운다음 승선지별 생존률을 구함
    dataset['Embarked'] = dataset['Embarked'].fillna(freq_port)

train_df[['Embarked', 'Survived']].groupby(['Embarked'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

	Embarked	Survived
0	C	0.553571
1	Q	0.389610
2	S	0.339009

3. 프로젝트 구조

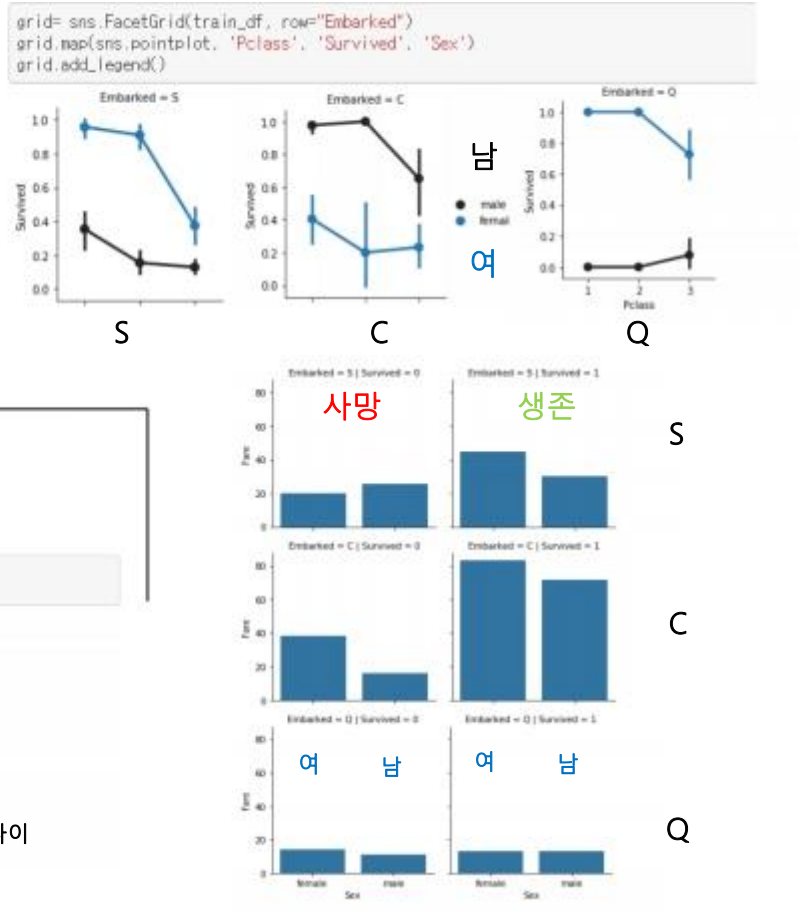
데이터전처리 (시각화)

가 . 좌석과 생존률 (train_data)



나 . 나이와 생존률 (train_data)

다 . 승선지·성별과 생존률 (train_data)



4. 결과 및 응용

가 . 결과

나이와 등급을 곱햐 값을 갖는 열을 만든 후

```
for dataset in combine:
    dataset['Age*Class'] = dataset.Age * dataset.Pclass

train_df.loc[:, ['Age*Class', 'Age', 'Pclass']].head(10)
```

	Age*Class	Age	Pclass
0	3	1	3
1	2	2	1
2	3	1	3
3	2	2	1
4	6	2	3
5	3	1	3
6	3	3	1
7	0	0	3
8	3	1	3
9	0	0	2

→

모든 열과 생존률 갂의 상관관계를 계산함

```
coeff_df = pd.DataFrame(train_df.columns.delete(0))
coeff_df.columns = ['Feature']
coeff_df["Correlation"] = pd.Series(logreg.coef_[0])

coeff_df.sort_values(by='Correlation', ascending=False)
```

	Feature	Correlation
1	Sex	2.201527
5	Title	0.398234
2	Age	0.287163
4	Embarked	0.261762
6	IsAlone	0.129140
3	Fare	-0.085150
7	Age*Class	-0.311201
0	Pclass	-0.749007

성별값이 클수록 (여성) 생존률 UP

좌석등급값 작으면 (1 등석) 생존률 UP

4. 결과 및 응용

가 . 결과

	Model	Score
3	Random Forest	86.76
8	Decision Tree	86.76
1	KNN	84.74
0	Support Vector Machines	83.84
2	Logistic Regression	80.36
7	Linear SVC	79.01
5	Perceptron	78.00
6	Stochastic Gradient Decent	77.55
4	Naive Bayes	72.28

다양한 머신러닝 모델을 적용한 결과 모델마다 target 예측 성공률이 달랐다

데이터마다 특성을 고려하여 적절한 모델을 선택한 필요가 있다 !

4. 결과 및 응용

나 . 응용 1: 새로운 열 (feature) 만들기

생존률에 영향을 미친 나이와 티켓요금을 곱해 Age*Fare 열을 만든 것과 비슷하게

성별과 티켓요금을 곱햐 Sex*Fare 열을 만든 후의 결과 비교

```
for dataset in combine:
    dataset['Sex*Fare']= dataset.Sex*dataset.Fare

train_df.head()
test_df.head()
train_df.tail()
```

	Survived	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class	Sex*Fare
886	0	2	0	1	1	0	5	1	2	0
887	1	1	1	1	2	0	2	1	1	2
888	0	3	1	1	2	0	2	0	3	2
889	1	1	0	1	2	1	1	1	1	0
890	0	3	0	1	0	2	1	1	3	0

< 기존 >

	Model	Score
3	Random Forest	86.76
8	Decision Tree	86.76
1	KNN	84.74
0	Support Vector Machines	83.84
2	Logistic Regression	80.36
7	Linear SVC	79.01
5	Perceptron	78.00
6	Stochastic Gradient Decent	74.19
4	Naive Bayes	72.28

<Sex*Fare 생성 후 >

	Model	Score
3	Random Forest	86.76
8	Decision Tree	86.76
1	KNN	84.85
0	Support Vector Machines	84.18
2	Logistic Regression	80.36
7	Linear SVC	79.01
5	Perceptron	77.78
6	Stochastic Gradient Decent	77.33
4	Naive Bayes	75.42

4. 결과 및 응용

나 . 응용 1: 새로운 열 (feature) 만들기

비슷한 방법으로 승선지와 성별을 곱한 feature, 나이와 성별을 곱한 feature 생성 후 각각 결과 비교

Embarked*Sex

```
In [22]: for dataset in combine:
             dataset['E*S']=dataset['Embarked']*dataset['Sex']
         display(train_df.head())
         display(test_df.head())
```

	PassengerId	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class	E*S
0	892	3	0	2	0	2	1	1	6	0
1	893	3	1	2	0	0	3	0	6	0
2	894	2	0	3	1	2	1	1	6	0
3	895	3	0	1	1	0	1	1	3	0
4	896	3	1	1	1	0	3	0	3	0

< 기존 >

	Model	Score
3	Random Forest	86.76
8	Decision Tree	86.76
1	KNN	84.74
0	Support Vector Machines	83.84
2	Logistic Regression	80.36
7	Linear SVC	79.01
5	Perceptron	78.00
6	Stochastic Gradient Decent	74.19
4	Naive Bayes	72.28

<E*S 생성 후 >

	Model	Score
3	Random Forest	86.76
8	Decision Tree	86.76
1	KNN	84.74
2	Logistic Regression	81.03
7	Linear SVC	79.57
0	Support Vector Machines	78.23
6	Stochastic Gradient Decent	77.89
4	Naive Bayes	74.75
5	Perceptron	66.78

Age*Sex

```
for dataset in combine:
    dataset['Age*Sex'] = dataset.Age * dataset.Sex

train_df.head()
```

	Survived	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class	Age*Sex
0	0	3	0	1	7.2500	S	1	0	3	0
1	1	1	1	2	71.2833	C	3	0	2	2
2	1	3	1	1	7.9250	S	2	1	3	1
3	1	1	1	2	53.1000	S	3	0	2	2
4	0	3	0	2	8.0500	S	1	1	6	0

< 기존 >

	Model	Score
3	Random Forest	86.76
8	Decision Tree	86.76
1	KNN	84.74
0	Support Vector Machines	83.84
2	Logistic Regression	80.36
7	Linear SVC	79.01
5	Perceptron	78.00
6	Stochastic Gradient Decent	74.19
4	Naive Bayes	72.28

<Age*Sex 생성 후 >

	Model	Score
3	Random Forest	86.76
8	Decision Tree	86.76
1	KNN	84.85
2	Logistic Regression	81.37
7	Linear SVC	79.12
6	Stochastic Gradient Decent	79.01
0	Support Vector Machines	78.34
4	Naive Bayes	75.20
5	Perceptron	74.41

4. 결과 및 응용

나 . 응용 2: 나이를 범주화 하지 않는다면 ?

연령범위를 5 부분으로 나누어 범주화

Let us create Age bands and determine correlations with Survived.

```
In [26]:
train_df['AgeBand'] = pd.cut(train_df['Age'], 5)
train_df[['AgeBand', 'Survived']].groupby(['AgeBand'], as_index=False).mean().sort_values(by='AgeBand', ascending=True)
```

Out[26]:

	AgeBand	Survived
0	(-0.08, 16.0]	0.550000
1	(16.0, 32.0]	0.337374
2	(32.0, 48.0]	0.412037
3	(48.0, 64.0]	0.434783
4	(64.0, 80.0]	0.090909

Out[51]:

< 범주화 후 >

	Model	Score
3	Random Forest	86.76
8	Decision Tree	86.76
1	KNN	84.74
0	Support Vector Machines	83.84
2	Logistic Regression	80.36
7	Linear SVC	79.12
6	Stochastic Gradient Decent	78.56
5	Perceptron	78.00
4	Naive Bayes	72.28

Out[51]:

< 범주화 전 >

	Model	Score
3	Random Forest	94.16
8	Decision Tree	94.16
1	KNN	87.21
7	Linear SVC	81.59
2	Logistic Regression	80.58
4	Naive Bayes	77.10
5	Perceptron	74.52
6	Stochastic Gradient Decent	73.40
0	Support Vector Machines	70.93

5. 느낀점

가 . 데이터전처리가 머쉬러닝을 이용한 데이터분석의 대부분을 차지한다 .

나 . 머쉬러닝 모델을 제대로 적용하기 위해서는 알고리즘의 작동방식에 대해 더 자세히 이해할 필요가 있다

다 . 이번 프로젝트에서는 다루지 않은 데이터 수집과정을 더 알아볼 필요가 있다 .