**Titanic: Machine Learning from Disaster**

**- ML 프로젝트 보고서**

의용생체공학과 201738413 김지연

**서론**

**1. Kaggle**

데이터 사이언티스들 및 개발자들이 데이터 사이언스, 머신러닝, 딥러닝 등과 같은 모든 예측 분석 태스크를 실무적으로 경험할 수 있는 공간이다. “Compete” 탭에서 진행되는 여러 유형의 주제의 챌린지에서는 개개인이 작성한 코드와 그에 대한 설명 그리고 데이터를 공유할 수 있다. 팀의 챌린지 결과를 제출하면 리더보드를 통해 순위를 확인할 수 있고, 좋은 성능을 낸 팀에게는 우승 보상을 주어 참가자들 간 경쟁력을 강화한다. 챌린지 주최지에 따라서 우승 보상은 상금이 될수도 있고, 일자리 제공이 될 수도 있으며 혹은 제품이 될 수도 있다.

**2. Titanic: Machine Learning from Disaster Challenge**

**챌린지 주제**

챌린지의 주제는 “ 타이타닉호 침몰 사고”이다. 1912년 4월 15일, 타이타닉호라는 이름의 여객선이 빙산과 충돌하여 침몰한 사고로, 역사적으로 남은 난파 사고이다. 승객과 승무원 총 2,224명이 승선해 있었으며, 사고로 인해 1,502명이 사망하였다.

**챌린지 목적**

이 챌린지의 목적은 사고에서 생존율이 높았던 유형의 사람들에 대한 예측 모델을 만드는 것이다.모델 구축에는 승선했던 사람들의 특징들을 담은 csv 파일을 학습데이터로 사용한다. 승객들의 특징들에는 이름, 연령, 성별, 사회경제적 클래스 등이 있다.

**챌린지 특징**

타이타닉호 침몰 사고 데이터로 만든 챌린지는 머신러닝계의 매우 기본적인 머신러닝이다. 이는 마치 여타 언어를 배울 때의 “Hello World”같은 존재이며. 딥러닝을 배울 때 MNIST 데이터로 처음 학습을 시작하는 것과 같은 맥락이다.

**3. 프로젝트 진행 환경**

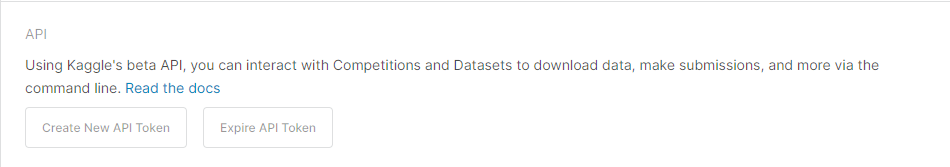
프로젝트 진행은 로컬에서 진행하였다. GPU를 사용할 수 있다는 점에서 코랩도 매우 좋은 프로젝트 진행 도구이다. 하지만 드라이브 마운트 및 데이터 다운로드가 복잡하며, 일정 시간 활성화 하지 않으면 런타임이 비활성화되는 점 때문에 로컬에서 진행하였다. 로컬에서는 주피터 노트북을 이용하였고, 머신러닝에 필요한 패키지를 설치한 적절한 가상환경을 구축하여 사용하였다.

**본론**

**1. 데이터셋 다운로드**

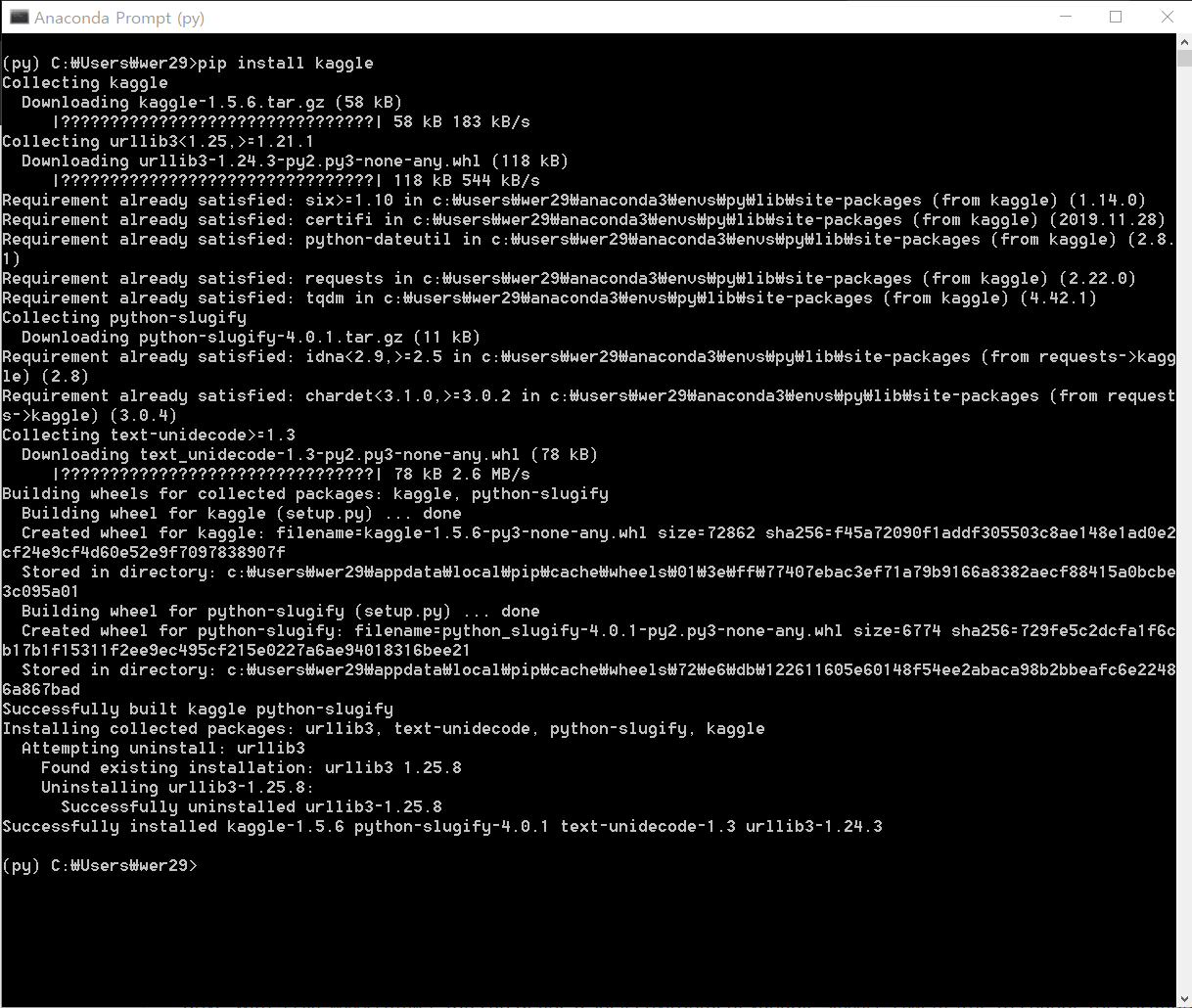
**kaggle.json 다운로드**

Kaggle 사이트에 로그인한 후, “My Account” 메뉴에 있는 API란에서 “Create New API Token”으로 kaggle.json을 다운로드한다. API를 사용하기 위해서는 kaggle.json 파일이 ~/.kaggle/kaggle.json 의 경로에 있어야 한다.

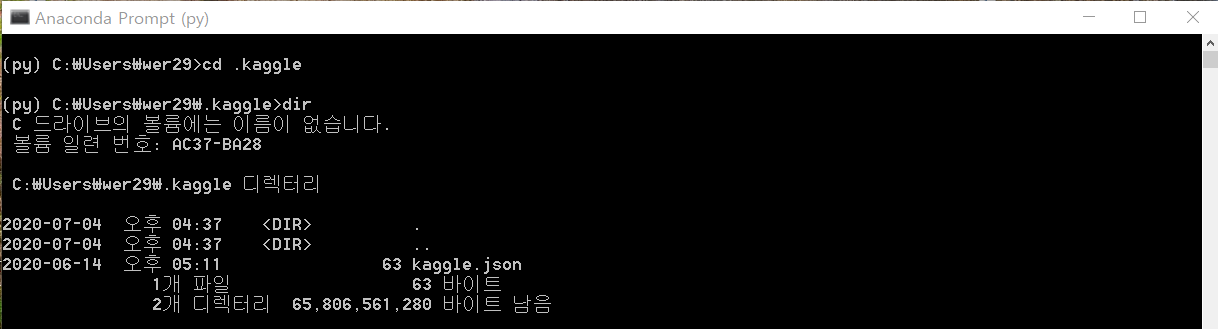


**가상환경에 kaggle 설치**

Kaggle API를 사용하기 위해 명령어 **pip install kaggle** 로 kaggle을 가상환경에 설치한다.

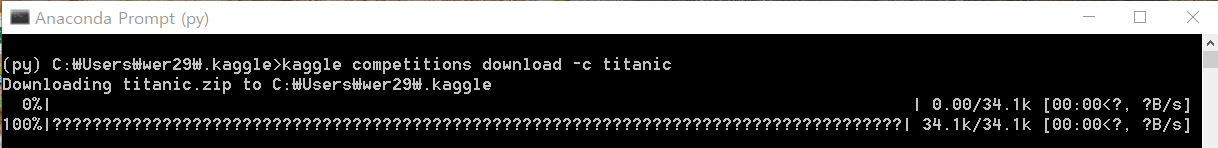


**kaggle.json이 있는 경로로 이동**



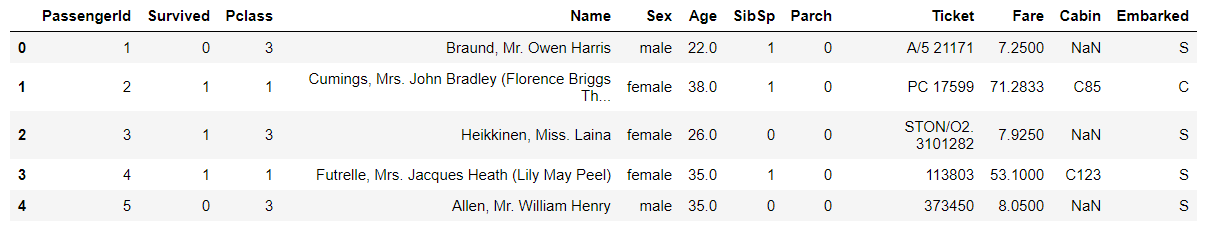
**데이터 다운로드**

**kaggle competitions download -c titanic** 명령어로 타이타닉 대회의 데이터셋을 다운로드한다. 데이터는 titanic.zip으로 다운로드 되며, gender\_submission.csv, test.csv, train.csv 의 세 파일로 구성되어 있다.



**2. 데이터셋 분석**

**Train 데이터 구성**

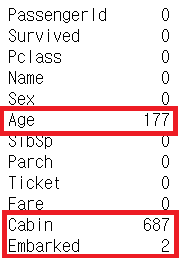


학습 데이터에는

* Survived: 생존 여부 => 0 = No, 1 = Yes
* Pclass: 객실 등급 => 1 = 1st, 2 = 2nd, 3 = 3rd
* Sex: 성별
* Age: 나이
* Sibsp: 함께 탑승한 형제자매, 배우자의 수
* Parch: 함께 탑승한 부모, 자식의 수
* Ticket: 티켓 번호
* Fare: 운임
* Cabin: 객실 번호
* Embarked: 탑승 항구 => C = Cherbourg, Q = Queenstown, S = Southampton

로 PassengerID를 제외한 총 11개의 feature가 있다.

**결측값 확인**

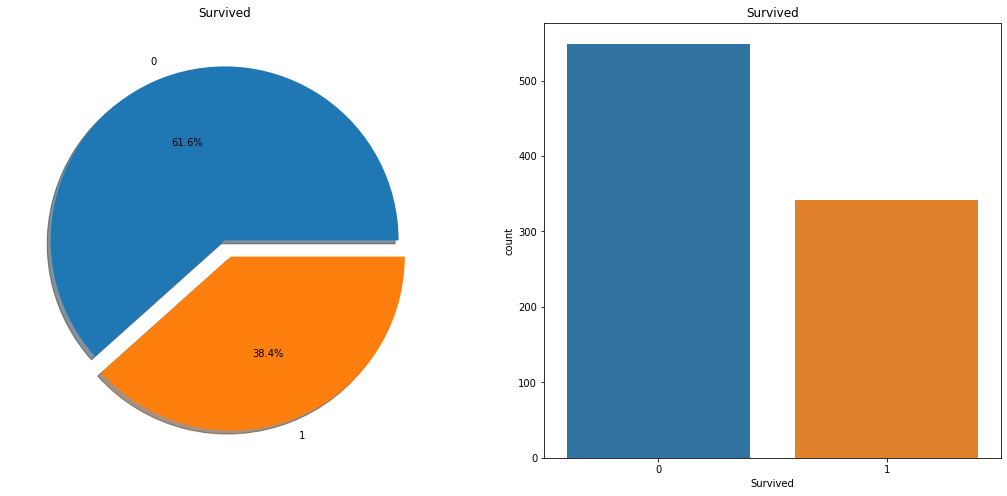


결측치가 많을수록 데이터 분석에서 오차가 발생할 위험이 있다. 즉, 결측치가 많은 feature는 모델 성능에 영향을 주므로, 모델 수립 이전에 결측치에 대한 확인을 해야한다. feature별 결측치 비율에 따라 각 feature의 결측치에 대한 처리방법이 달라진다.

Train 데이터에서 결측치를 확인한 결과,

* Cabin(687), Age(177), Embarked(2) 3개의 feature에 결측값이 존재한다.
* Cabin, Age는 많은 결측값을 가지며, Embarked는 아주 적은 결측값을 갖는다.

**전체 생존자 수 확인**



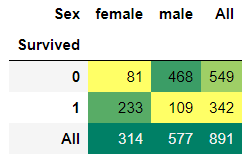
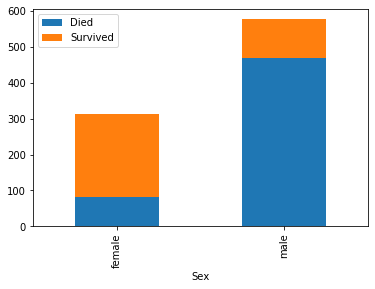
파이차트에서 레이블 0이 사망, 1이 생존을 의미한다. 차트에 따르면, 61.6%가 사망하였고, 38.4%가 생존하였다. Bar 그래프에 따르면,

* 총 891명 중에서 약 540 명이 사망하였고,
* 약 350명이 생존하였다.

즉, 전체 승객 중에서 사망자의 수가 생존자의 수에 비해 약 2배가량 많았다.

**성별에 따른 생존율 확인**

성별과 생존여부 사이의 관계를 확인하기 위한 목적으로 성별에 따른 생존한 샘플과 사망한 샘플의 수를 각각 확인한다.



여성 승객과 남성 승객의 생존여부 확인 결과 (Blue가 사망, Orange가 생존),

* 여성 승객은 81명이 사망, 233명이 생존하였고 (약 74% 생존율),
* 남성 승객은 468명이 사망, 109명이 생존하였다 (약 19% 생존율).

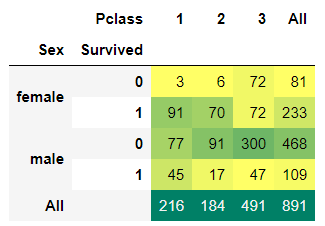
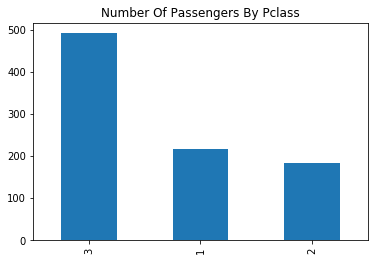
즉, 성별(Sex) 특성에서는 남성 승객에 비해 여성 승객의 생존율이 월등히 높다. 특히 승선한 승객 중 남성 승객의 수가 여성 승객의 수보다 2배가 많음에도 불구하고 남성 승객의 생존율이 여성 승객보다 훨씬 낮다.

* 여성 승객은 약 74%의 생존율을 가지고,
* 남성 승객은 약 19%의 생존율을 가진다.

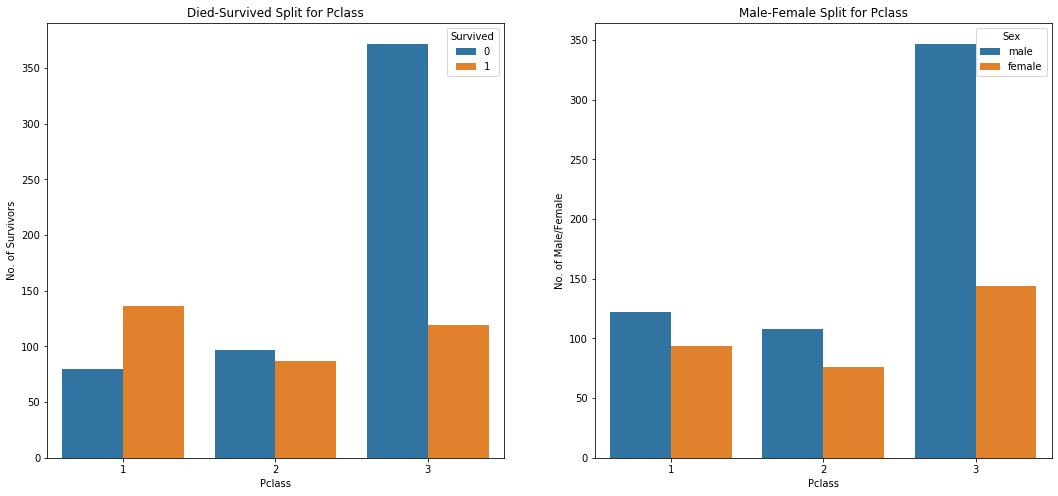
각 성별의 비율과 상관없이 일관되게 남성 승객은 생존율이 낮고, 여성 승객은 생존율이 높은 것은 유의미한 특성임을 나타낸다. 여기서 알 수 있는 것은, 승객 중 남성 승객 수가 여성 승객보다 약 2배 정도 많음에도 불구하고 생존율은 약 4배 낮은 것으로 보아 구조 시에 여성에게 먼저 구조가 우선되었음을 알 수 있다.

**객실 등급에 따른 생존율 확인**

객실 등급과 생존여부 사이의 관계를 확인하기 위한 목적으로 객실 등급에 따른 생존한 샘플과 사망한 샘플의 수를 각각 확인한다.



3등급 객실의 승객이 가장 많으며, 뒤이어 1등급, 2등급 순으로 승객이 많다. 이에 따른 객실 등급별 생존율을 구해봐야 한다.



객실 등급별 생존여부를 확인 결과,

* 1등급 객실의 샘플은 136명이 생존하였고, 80명이 사망하였고 (약 63% 생존율),
* 2등급 객실의 샘플은 87명이 생존하였고, 97명이 사망하였으며 (약 47% 생존율),
* 3등급 객실의 샘플은 119명이 생존하였고, 372명이 사망하였다 (약 24% 생존율).

1등급, 2등급, 3등급 객실 순으로 생존율이 높다. 즉, 객실등급(Pclass) 특성에서는 1등급 객실의 생존율이 가장 높다.

3등급 객실의 승객 수가 1등급 객실보다 약 2배가량 많음에도 불구하고, 생존율은 2.5배 낮다. 여기서 간접적으로 알 수 있는 사실은, 객실 별 사람 수와 관계없이 돈이 많고 지위가 높은 사람들에게는 구조에서 우선되었다는 점이다.

**나이에 따른 생존율 확인**

나이와 생존여부 사이의 관계를 확인하기 위한 목적으로 나이에 따른 생존한 샘플과 사망한 샘플의 수를 각각 확인한다. 그 전에 먼저 Age 특성의 177개 결측치를 처리해야 한다. 이 177개의 결측치를 처리하는 방법에는 여러가지가 있다.

1. 나이의 평균으로 대체:

가장 쉬운 방법이다. 하지만 데이터셋에는 다양한 연령대의 샘플들이 있기 때문에 평균으로 대체한다면 좋은 성능을 내는 모델을 만드는 데에 큰 도움이 되지 못한다. 가장 어린 승객의 나이와 가장 나이가 많은 승객의 나이는 outlier 데이터가 되기 때문이다.

1. 나이의 중앙값으로 대체:

평균과 마찬가지로 쉬운 방법이다. 다른 점이라면, 평균은 outlier 값들을 포함하기 때문에 일반적인 값을 얻기 어렵지만, 중앙값을 얻는다면 outlier 데이터들의 영향을 피해갈 수 있다는 점이다. 그러나 마찬가지로, 그런 이상치들을 피한다고 하더라도 177개의 데이터를 중앙값 하나로 대체하기에는 근거가 부족하다.

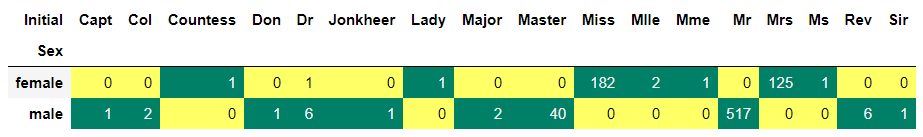
1. Name 특성을 이용하여 대체:

외국의 경우 나이와 성별에 따라 앞에 붙는 호칭이 다르다. 예를 들어 Mr, Mrs, Miss 등이 있는데, 이러한 특징들을 이용하면 각 호칭 별 나이대의 평균을 구할 수 있다. 이렇게 구한 평균들로 177개의 결측치의 Name 변수에 있는 호칭에 따라 알맞게 나이 변수를 채워 넣을 수 있게 된다.

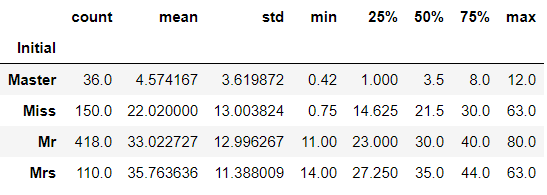
이에 따라 3번의 방법으로 Age의 177개 변수를 처리한다.

먼저 호칭 뒤에는 .(온점)이 온다는 점을 이용하여 호칭을 추출한다. Name 특성에서 온점이 오기 직전까지의 A-Z 사이의 문자열을 모두 추출하고, 새로운 데이터 특성인 "Initial"이라고 지정하여 정의한다.

이렇게 추출한 호칭들은 ['Mr' 'Mrs' 'Miss' 'Master' 'Don' 'Rev' 'Dr' 'Mme' 'Ms' 'Major' 'Lady' 'Sir' 'Mlle' 'Col' 'Capt' 'Countess' 'Jonkheer'] 이다. 이렇게 호칭이 다양한 이유는 Name 특성에서 작성 상의 오타이거나, 다양한 연령대로 인한 여러 호칭으로 작성되었기 때문이다. 이 호칭들은 모두 Mr, Mrs, Miss, Other로 구분하여 범주화 한다. 범주화 과정을 용이하게 하기 위해 성별과 호칭을 같이 출력한다.

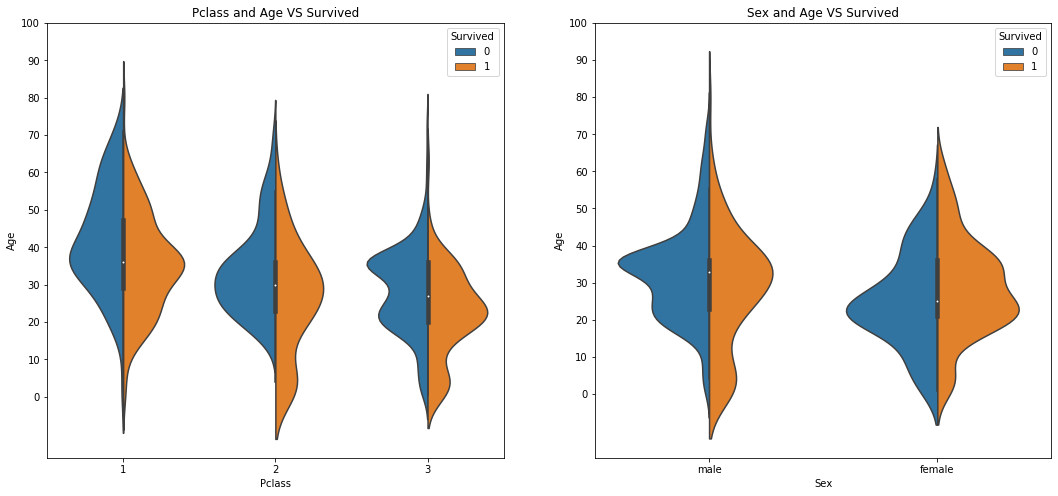


* Don : = sir(상류층), 성인남성
* Mme : Madame, Mrs, 기혼 여성
* Capt : Captain, 성인남성
* Lady : 상류층 숙녀, 미혼 여성
* Countess : 여자 백작, 기혼 여성
* mlle : Mademoiselle, Miss, 미혼 여성
* Jonkheer : 낮은 직급의 귀족, 젊은 남성 귀족, 성인남성
* Rev : 목사/신부에 대한 경칭, 성인남성
* Col : 군인 관련, 성인남성
* Master : Mr 보다 어린 남자아이, 어린아이
* Major : 소령, 성인남성

위를 바탕으로, 다음과 같이 Master, Mr, Mrs, Miss 4개로 호칭을 재정리 할 수 있다.

여기서 구해지는 4개의 범주 별 평균 나이를 기준으로, 데이터셋의 호칭에 따라 결측값을 채운다. 평균값은 반올림하여 Master는 5세, Miss는 22세, Mr는 33세, Mrs는 36세로 한다.

결측값을 채운 후, 나이와 성별에 따른 생존율을 시각화 한다. 나이는 범주형이 아닌 연속형이기 때문에 다른 시각화 방법을 사용해야 한다.

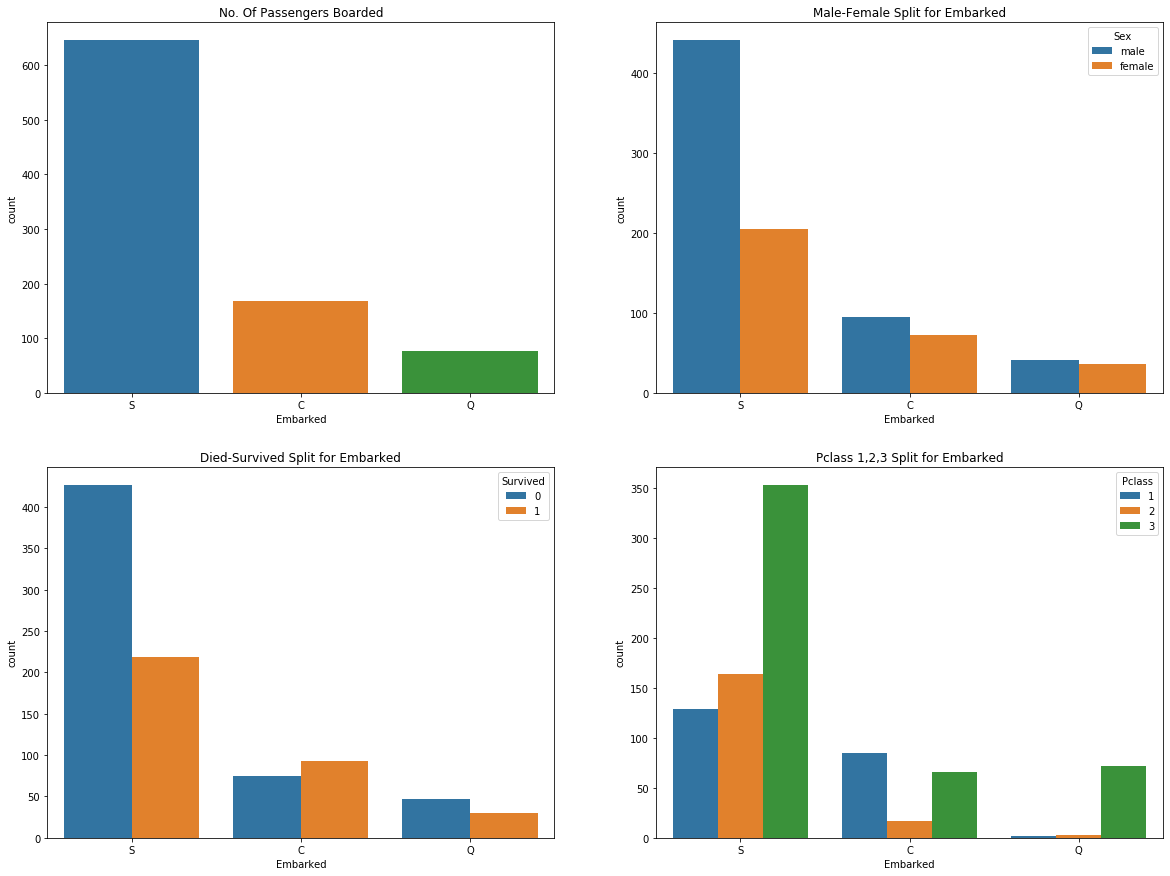
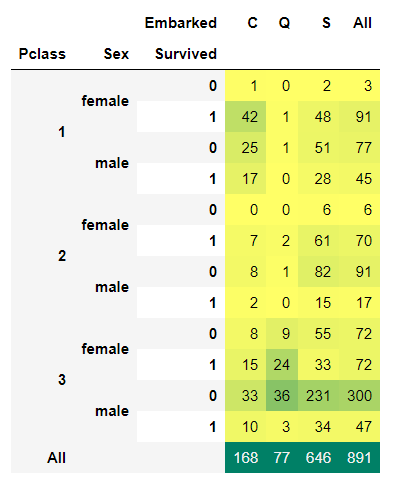


객실 등급 별로 본 나이별 생존율은 1등급의 30-40대가 가장 높다. 그러나 사망자 역시 30-40대가 가장 높다. 그리고 낮은 객실 등급일수록 10세 이하의 어린아이가 많다. 그러나 객실 등급 특성과 생존율의 관계에서 높은 객실일수록 생존율이 높다는 가정과는 상충되는 부분이 있다. 어린아이들은 객실 등급과 상관없이 생존율이 높다. 2등급과 3등급 객실의 10세 이하 생존율이 그 근거가 된다. 특히 5세 이하의 어린아이들의 생존율이 높은 것으로 보아 어린아이들이 우선 구조되었음을 알 수 있다.

성별 별로 본 나이별 생존율은, 남성 승객의 경우 나이가 많을수록 생존율이 낮고, 30대의 여성과 남성의 생존율이 가장 높다. 그러나 생존율을 보았을 때는 남성 승객보다 여성 승객의 생존율이 더 높게 나타난다. 즉, 여성 승객과 어린 아이들이 우선 구조되었다.

**탑승 항구에 따른 생존율 확인**

탑승 항구와 생존여부 사이의 관계를 확인하기 위한 목적으로 탑승 항구에 따른 생존한 샘플과 사망한 샘플의 수를 각각 확인한다. 그 전에 먼저 Embarked 특성의 2개 결측치를 처리해야 한다. 여기서 결측치는 2개뿐이므로 가장 간단한 방법으로 처리한다. 세 개의 항구에서는 각각 S항구 644명, C항구 168명, Q항구 77명이 탑승했다. S(Southampton) 항구에서 압도적으로 많은 승객들이 탑승했으므로, 2개의 결측치도 S값으로 채운다. 이는 2개의 샘플이 세 개의 항구 중에서 S 항구에서 탑승했을 확률이 높기 때문이다.



결과적으로 C항구에서 탑승한 승객의 생존율이 가장 높았으며, S항구에서 탑승한 승객의 생존율이 가장 낮았다. 이는 위에서 살펴본 바와 같이, 생존율이 가장 낮았던 3등급 객실의 샘플들이 대부분 S항구에서 탑승했기 때문이다. 여기서 특이한 점은 C항구의 경우 유일하게 사망자 수가 생존자 수보다 적은데, 이는 1등급 객실 사용자들이 S와 C 항구에 대부분 몰려있고, 3등급 객실 이용자들의 분포가 높은 S항구에 비해 C항구는 비슷한 분포를 가지기 때문이다. 간단히 말하자면, S항구에 비해 C항구는 상대적으로 1등급 객실 이용자들의 비율이 높고, 1등급 객실 이용자들을 우선 구조했기 때문에 C항구에서 탑승한 승객들의 생존율이 높게 나타났다. 이 특성 역시 부와 지위가 있을수록 우선 구조됨을 뒷받침한다.

**3. 특성 전처리**

머신러닝 모델을 만들기에 앞서, 연속형인 데이터들은 일정 기준을 만들어 범주화 해주어야 한다. "Pclass"와 같은 특성은 1,2,3으로 카테고리가 3개로 고정되어 필드값으로 들어갈 수 있지만, "Age"와 같은 특성은 최대는 샘플의 수만큼 변수가 생성될 수도 있다는 단점이 있다. 타이타닉 데이터셋에서는 Fare, Age가 연속형 데이터이므로 4개, 5개의 구간 등으로 나누어 범주화하여 새로운 특성으로 만들어 내었다.

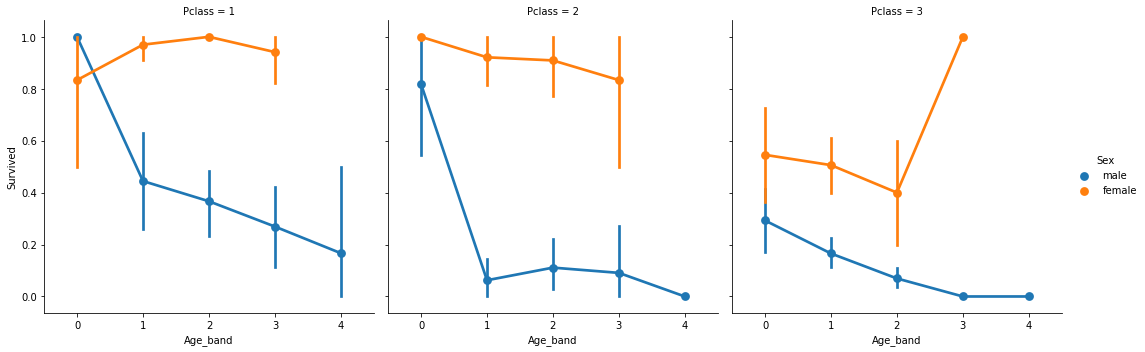
또 다른 특성 처리에는 상호 연관성이 있는 특성들을 바탕으로 새로운 유의미한 특성을 만드는 방법도 있다. 타이타닉 데이터셋에서는 Parch와 SibSp를 이용하여 가족 단위의 승객과 혼자 승선한 승객으로 나누는 하나의 새로운 특성을 만들어 낼 수 있다.

새로운 특성을 만들어내는 것과 반대되게 기존의 특성을 삭제하는 과정도 중요하다. 중복되거나 결측값이 많아서 무의미한 데이터인 특성들은 모델을 만드는 데에 도움이 되지 못하므로 제거하는게 현명하다. 따라서 Embarked나 Age 처럼 충분히 결측값 처리가 가능한 특성과 달리 결측값이 매우 많았던 Cabin 은 데이터에서 제외시키면 된다.

특성을 범주화하고 유의미한 특성을 새로 만든 후에는 마지막으로 문자열로 이루어진 특성을 수로 변환하는 것이다. 머신러닝 모델에는 문자가 아닌 숫자가 전달되어야 하기 때문에 Sex, Embarked, Initial 과 같은 문자열 데이터를 수로 변환하여야 한다.

**Age 특성의 범주화**

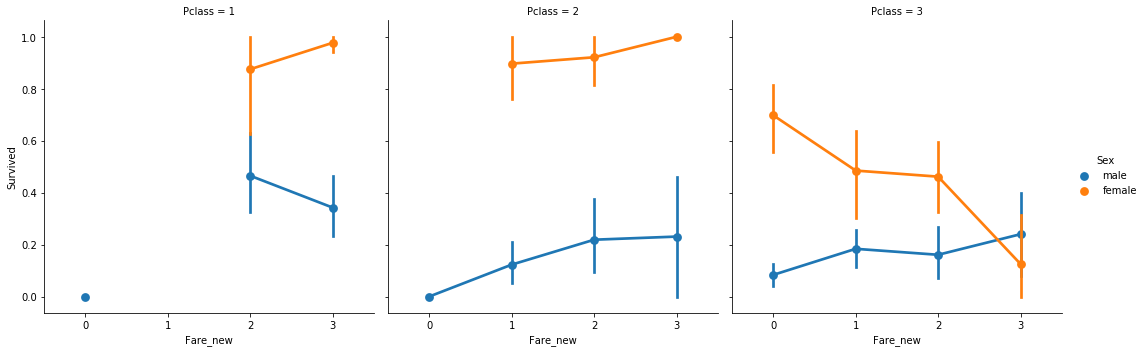
Age의 경우 최대 연령이 80임을 확인한 0부터 4까지의 5구간을 설정하여 "Age\_band"라는 새로운 특성으로 만든다. 즉, 0구간이 가장 나이가 적고, 4구간으로 갈수록 나이가 많은 집단이다. 그 결과 17-32세와 33-48세 구간이 가장 많은 샘플 분포 구간이었다. 또한 객실 등급별로 본 각 나이 구간 별 생존율은 다음과 같았다.



세 개의 그래프 모두 우하향을 향하는 동일한 패턴을 가지고 있다. 차이점이 있다면 낮은 등급 객실일 수록 그래프가 아래로 편향되어 있다는 점이다. 이를 통해서 객실 등급이 낮다면 이미 알아본 바처럼 생존율이 낮으며, 전반적으로는 객실 등급과 무관하게 나이가 많을수록 생존율이 낮고, 반대로 어릴수록 생존율이 높다. 남성 승객 데이터는 객실 등급과 나이 구간에 상관 없이 전부 생존율이 낮다.

**Fare 특성의 범주화**

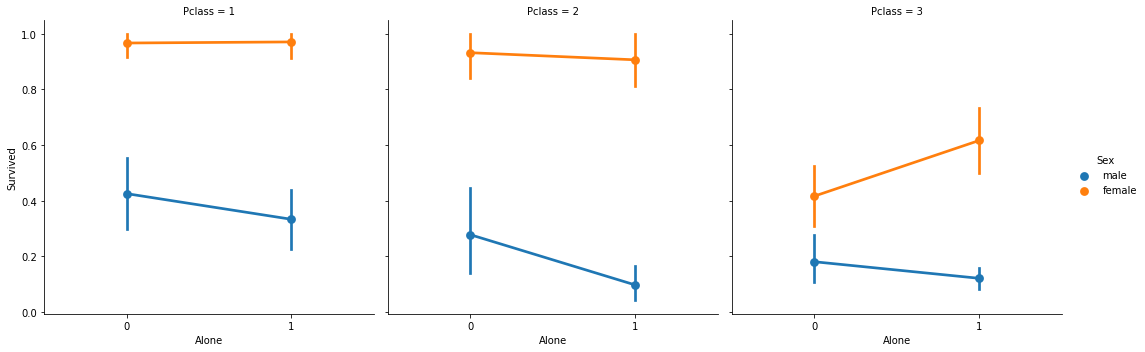
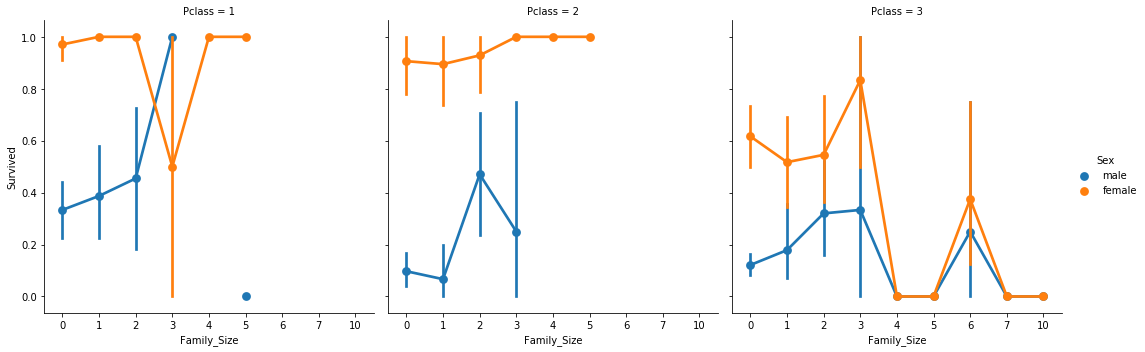
연속형 데이터인 Fare도 마찬가지의 방법을 적용한다. 요금을 0-3까지의 4등급으로 나누어 "Fare\_new"라는 범주화된 새로운 특성으로 만든다. 이렇게 나누면 저렴한 비용으로 이용한 저소득층과 비싼 값의 티켓을 이용한 부유층에 따른 생존율을 볼 수 있다. 티켓 값에 따른 데이터 분포는 균일했다. 또한 객실 등급 및 성별로 본 각 요금 구간 별 생존율은 다음과 같았다.



높은 등급의 객실을 이용한 사람들은 높은 생존율을 가짐을 이미 분석한 상태에서, 비싼 요금을 지불하는 집단 역시 대체로 높은 등급의 객실을 이용했을 것이다. 따라서 그래프에서 알 수 있듯이 1등급 객실에는 2,3 등급 요금을 지불한 데이터들이 치중되어 있고, 생존율 역시 높다. 2등급 객실의 경우 높은 이용 요금을 낸 데이터는 생존율이 높지만 그렇지 않은 경우는 생존율이 낮았다. 3등급 객실을 이용한 데이터들은 요금 등급에 관계없이 생존율이 평균적으로 낮은 편이다. 남성 승객 데이터는 객실 등급과 요금 구간에 상관 없이 전부 생존율이 낮다.

**Parch와 SibSp 특성의 결합**

Parch와 SibSp 각각 함께 탑승한 부모와 아이들의 수, 형제자매와 배우자의 수를 나타내는 특성이다. 부모와 아이, 형제자매와 배우자는 모두 가족으로 볼 수 있고 필드 값은 인원 수를 표시하므로, 이 두 특성의 조합은 가족의 크기를 나타내는 새로운 특성인 "Family\_Size" 가 될 수 있다. 두 특성을 더하면 0부터 10까지 다양한 크기의 가족이 나온다. 여기서 크기가 0이라면 가족이 없다는 뜻이므로 혼자 승선했음을 의미한다. 따라서 이러한 특수 경우는 "Alone" 이라는 새로운 특성으로 만든다. 데이터 분포를 살펴보면, 가족 단위의 경우 1명을 동반하는 경우가 가장 많았고, 혼자 온 승객은 가족과 함께 온 승객에 비해 1.5배 정도 많았다. 또한 객실 등급 및 성별로 본 각 가족 크기 구간 별 생존율은 다음과 같았다.

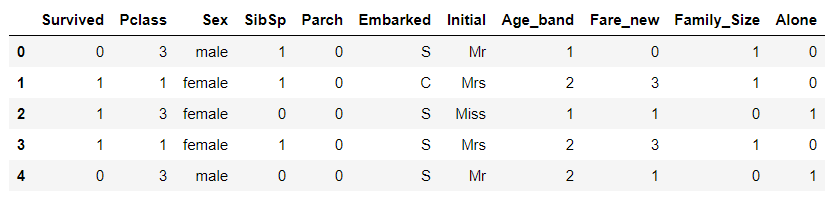


1행은 가족과 함께 승선한 데이터고, 2행은 혼자 승선한 데이터이다. 2행만 봐도 알 수 있듯이 여성 승객의 전형적인 특징인 높은 생존율이 있더라도 승객의 대부분은 남성이었다는 점을 생각하면 혼자 탄 승객들은 생존율이 굉장히 낮음을 알 수 있다. 남성 승객 데이터의 특징은 여성 승객에 비해 항상 생존율이 낮다는 점이다.

**무의미한 특성 삭제**

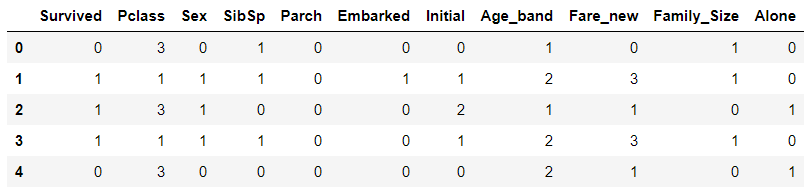
보통 결측값이 있는 특성에 대해서는 여러 방법을 강구하여 그 null 값을 채워 넣지만, Cabin과 같이 대부분의 값이 비어 있는 경우라면 값을 채우기보다는 해당 특성을 제거하는 것이 간단하고 더 좋은 모델링을 가능하게 한다. 결측 값이 많다는 것은 그만큼 정보가 없기 때문에 의미가 없는 특성임을 나타내기 때문이다. 이와 같은 이유로 Cabin 특성의 열은 삭제한다.

마찬가지로 다른 여러 특성들도 제거될 수 있다. 제거의 기준은 불필요한 특성이거나 범주화 할 수 없을 때이다. PassengerId, Name, Ticket 속성은 범주화가 불가능하여 제거한다. Age, Fare, Fare\_Range, Died는 생존율과 관계가 있는 속성이지만 이미 새로운 속성인 Age\_band, Fare\_new로 만들었고 Died는 Survived와 같은 속성이므로 불필요한 속성이다. 제거 후 남는 속성은 Survived, Pclass, Sex, SibSp, Parch, Embarked, Initial, Age\_band, Fare\_new, Family\_Size, Alone 이다.

****

**문자열 데이터 수로 변환**

Sex, Embarked, Initial 은 모두 문자열 필드 값을 갖는 속성이므로 숫자로 변환한다. 나머지 속성들은 모두 숫자를 데이터로 가지므로 상관없다. 문자열들은 각 속성의 카테고리 별로 0부터 변환한다.



**4. 모델 형성**

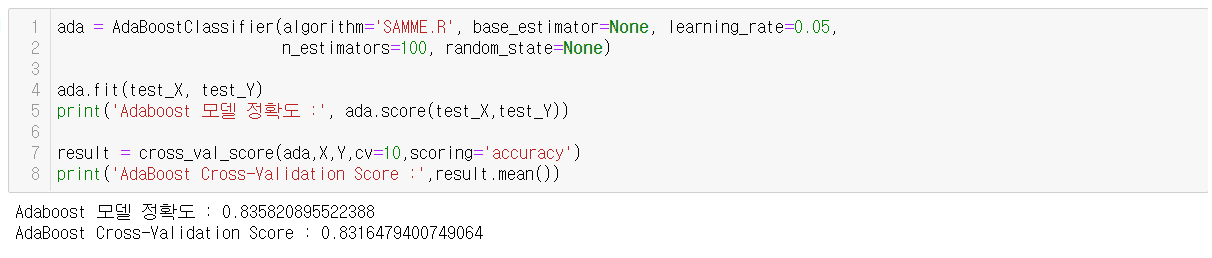
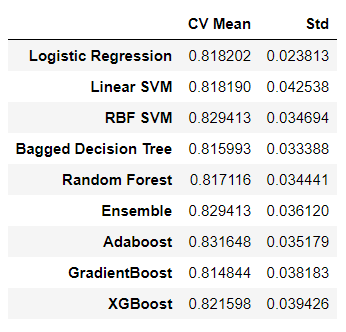
**학습 데이터 분할**

데이터는 Train.csv에서 train\_X가 70%, train\_Y가 30%가 되도록 나눈다. 이때 Survived는 레이블이므로 포함되지 않도록 유의한다. 또한 교차 검증용 데이터를 만들기 위해 Survived 특성을 포함하지 않는 전체 학습 데이터와 Survived 특성만을 포함하는 레이블을 X, Y로서 만든다.

**학습 알고리즘**

테스트 해본 학습 알고리즘은 'Logistic Regression', 'Linear SVM', 'RBF SVM', 'Bagged Decision Tree', 'Random Forest', 'Ensemble', 'Adaboost', 'GradientBoost', 'XGBoost'의 9개이다. 이 중 SVM, Random Forest, Adaboost, Gradient Boosting, SGBoost 모두 Grid search를 하여 하이퍼 파라미터 튜닝을 하였다. Decision Tree에는 Bagging을 적용하였고, 앙상블 모델도 있었다.

최적의 모델을 만들기 위해 전체 9개의 모델에 대하여 10 폴드 교차 검증으로 성능을 비교한 결과 다음과 같았다.



교차 검증 결과 9개의 모델 중 테스트 데이터에 대해 가장 좋은 성능을 보인 모델은 Adaboost 였다. 따라서 모델은 Adaboost로 선택하게 되었다. Adaboost는 Adaptive Boost의 줄임말로, 분류문제에만 적용 가능하고, 부스팅 계열 앙상블 알고리즘이며 그 원리는 다음과 같다.

1. 1번 약한 분류기가 자신의 기준으로 생존과 사망을 분류한다.
2. 2번 약한 분류기는 1번 분류기가 분류에 실패한 데이터에 대해서 더 높은 가중치를 두어 분류한다.
3. 3번 약한 분류기는 2번 약한 분류기가 잘못 분류한 데이터에 대해 더 가중치를 두어 분류한다.
4. 이와 같은 과정이 목표 정확도 등에 도달할 때까지 반복되며,
5. 최종적으로는 모든 분류기들을 합쳐 테스트 세트에 대한 예측을 한다.

**즉, Adaboost 알고리즘은 약한 분류기들을 하나씩 직렬로 학습하고, 각 분류기들은 이전 분류기의 데이터 분류 결과에 따라 예측을 한다. 구체적으로는, 이전 분류기의 잘못 분류된 데이터들에 대해 높은 가중치를 부여하여 그것을 하나씩 순차적으로 약한 분류기들이 고쳐 나가는 방식이다. 개별 약한 분류기들의 성능은 매우 좋지 않더라도, 이러한 방식으로 직렬 처리되어 하나의 알고리즘이 된다면, 모델의 성능은 매우 높아지게 된다. 여기서 약한 분류기는 Decision Tree로, 디폴트 값이다.**

Adaboost 알고리즘으로 모델을 만들 때 하이퍼 파라미터는 그리드 서치를 이용하여 설정하였다. 그리드 서치에 의존하여 하이퍼 파라미터를 설정한 이유는 다음과 같다.

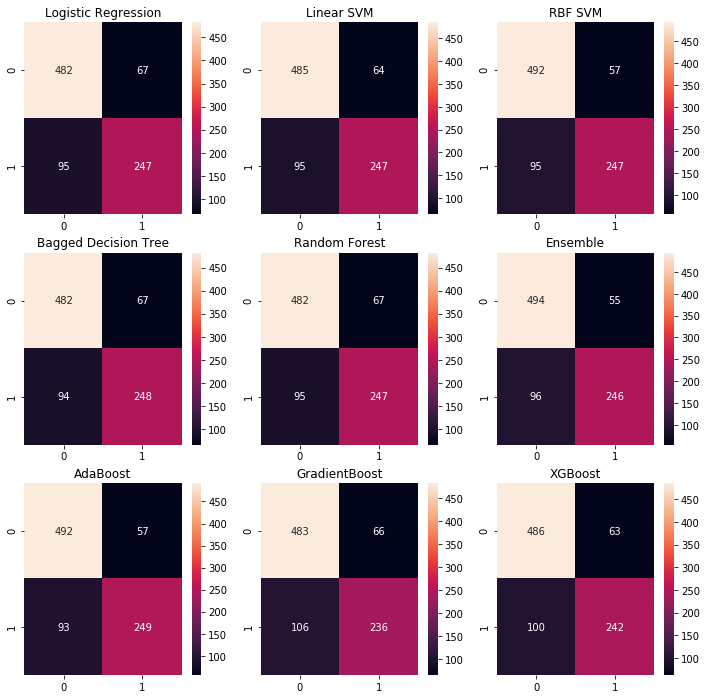
1. 최적의 하이퍼 파라미터를 찾기 위해서는 반복적인 작업이 필요하다.
2. 사실상 수많은 경험에 의거하지 않는 이상은 최적의 파라미터를 찾기 힘들며, 찾았다고 하더라도 신뢰하기 힘들다.

따라서 그리드 탐색을 여러 알고리즘에 적용하여 최대의 성능을 낼 수 있도록 돕고, 결과로 나온 하이퍼 파라미터를 바탕으로 테스트 세트를 평가하는 것이 효율적이다. 아다부스트에 관해서는 중요 하이퍼 파라미터인 n\_estimators와 learning\_rate을 설정하는 것을 그리드 탐색에 맡겼다. 여기서 각 주요 하이퍼 파라미터의 의미는 다음과 같다.

* n\_estimators :

학습에 사용한 약한 분류기의 개수이다. 디폴트 값은 50이다. 이 하이퍼 파라미터는 커질수록 decision boundary의 수도 같이 증가하기 때문에 모델이 점점 복잡해진다. 따라서 모델이 오버피팅된다면 약한 분류기의 수를 낮춰보는 것이 하나의 해결책이다. 그리드 탐색을 통해서 100이라는 값을 얻었다.

* learning\_rate :
* 약한 분류기가 학습하면서 이전 분류기의 오류들을 보정할 때의 계수인 학습률이다. 약한 분류기를 제어하는 하이퍼 파라미터와 trade-off 관계이다. 그리드 탐색을 통해서 0.05라는 값을 얻었다.

**5. 모델 성능평가**

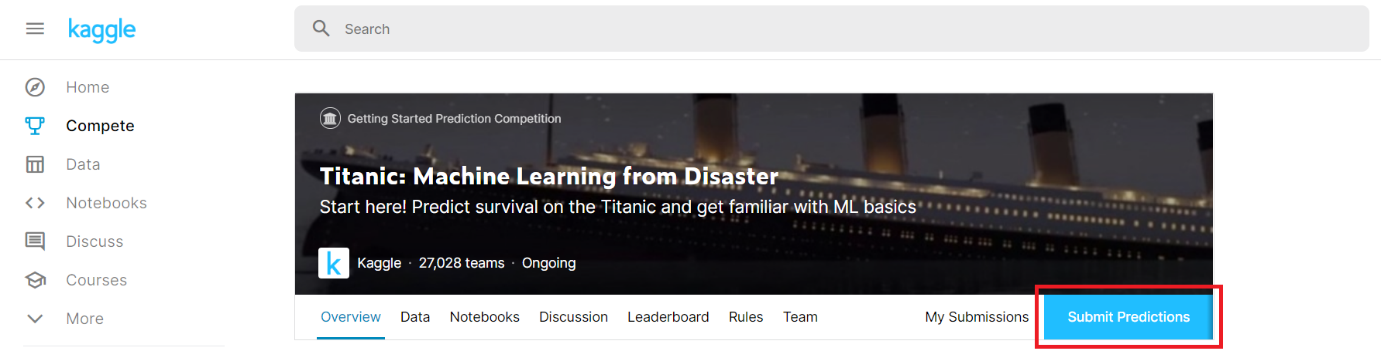
학습을 위해 모델을 선택하기 전 시도했던 모든 모델과 Adaboost 알고리즘을 이용한 모델로 테스트 세트에 대해 교차 검증으로 예측한 결과를 confusion matrix로 나타내었다. 그 결과 위에서 교차 검증을 통해 알아본 확률과 같이, Adaboost가 가장 좋은 성능을 보였고, 그와 유사하게 RBF 커널을 사용하는 SVM이 좋은 모델이었다.

대체로 모든 영역에 있어서 알고리즘별 차이는 없어 보인다. 그러나 Adaboost가 가장 좋은 성능을 낸 이유는, 사망자에 대한 예측 능력이 뛰어났으며, 생존자를 사망자로 잘못 예측하는 수가 다른 알고리즘에 비해 적었기 때문이다.

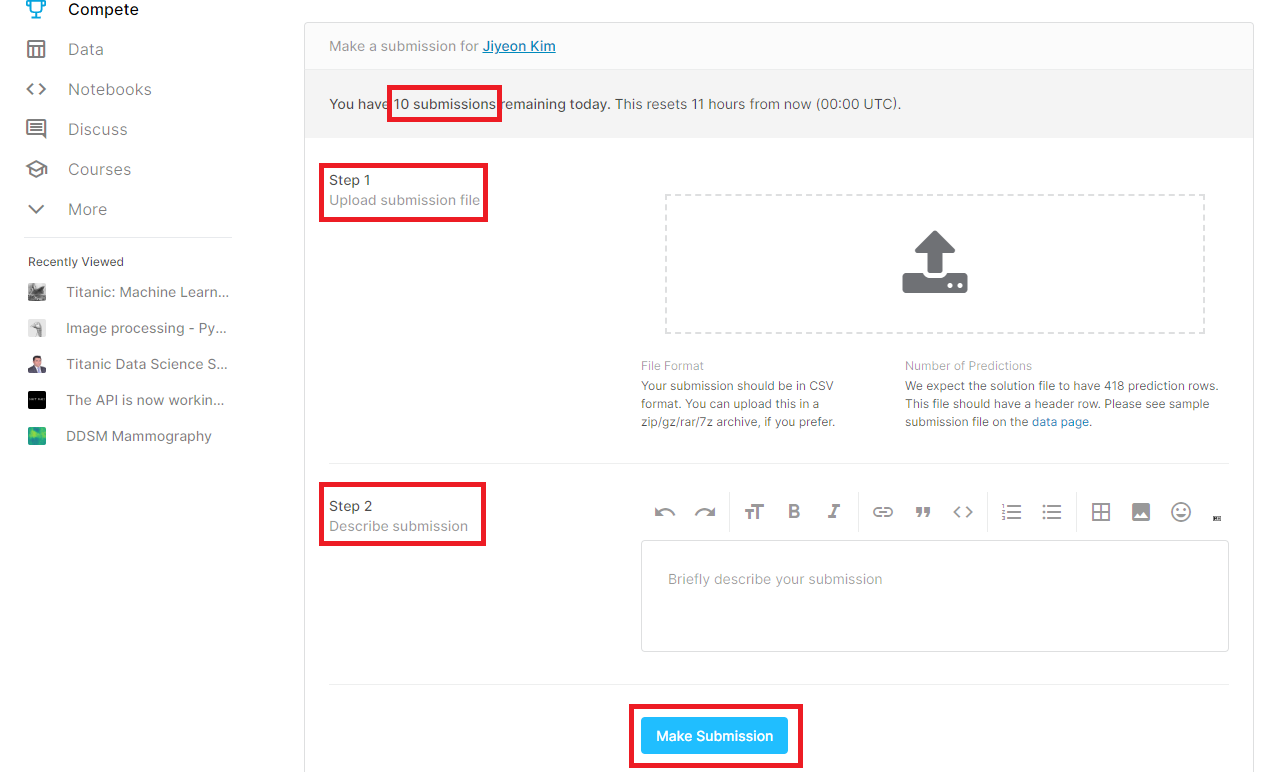
마지막으로는, Kaggle에서 제공한 퍼블릭한 Test.csv를 이용하여 예측을 한다. 이 데이터에 대한 예측은 레이블을 알 수는 없다. 하지만 Kaggle 측에 Test.csv에 대해 PassengerID별 Survived 예측을 한 파일을 업로드하면 성능을 평가받을 수 있고, 리더보드에 순위와 함께 기록된다.

**결론**

**kaggle에 결과 제출**

1. “Compete” 탭의 해당 챌린지 페이지에서 “Submit Predictions” 버튼을 누른다.

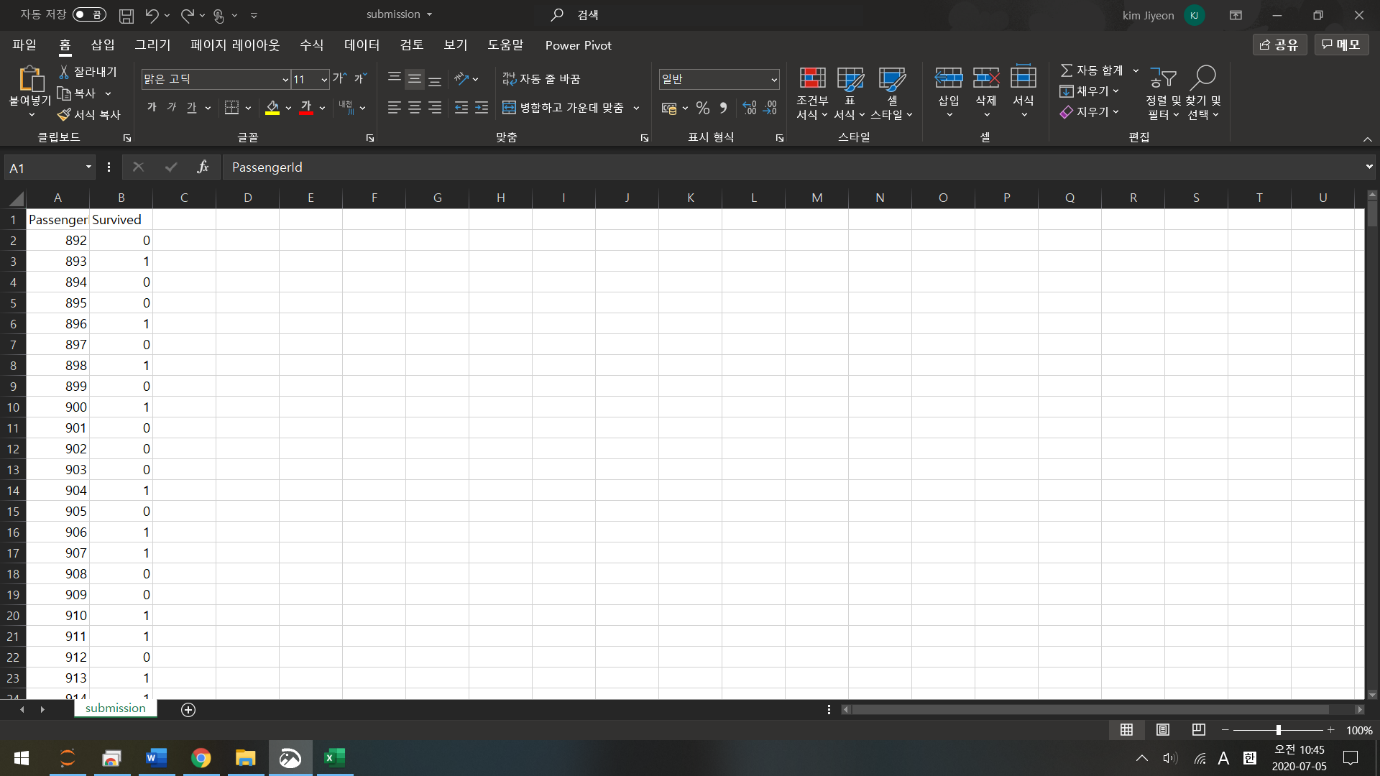
2. 제출 형식에 맞게 편집한 .csv 파일을 업로드한다.



**제출 시 유의사항**

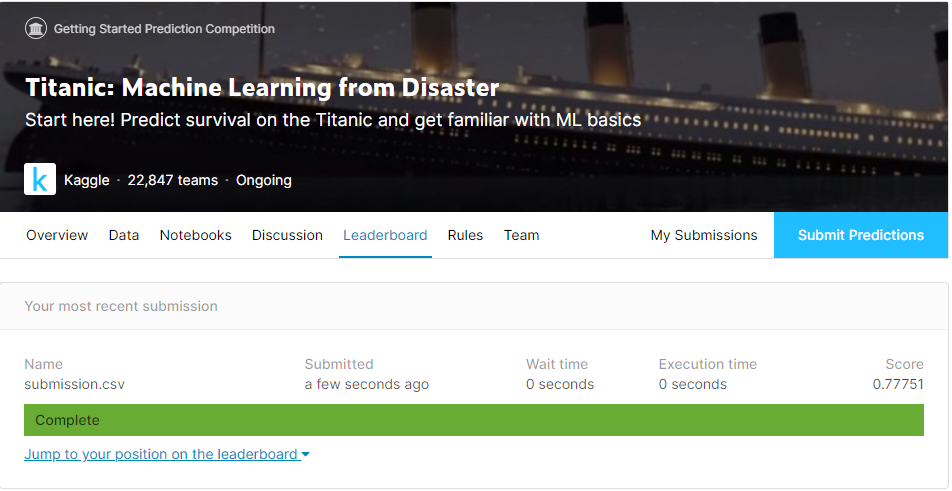
제출형식은 Header 행이 있어야 하며, 2개의 열만을 가져야 하고, 418개의 엔트리를 갖는 .csv 파일이어야 한다. 2개의 열은 각각 다음과 같다.

* PassengerID
* Survived

****

두번째 열은 0,1로 이루어진 binary prediction이어야 한다. 형식을 지키지 않은 제출파일은 제출시 에러가 발생한다.

**Submission 완료**



**Ref.**

<https://www.kaggle.com/c/titanic>

<https://developers.ascentnet.co.jp/2017/11/24/kaggle-process-review/>

<https://towardsdatascience.com/kaggle-titanic-machine-learning-model-top-7-fa4523b7c40>

<https://www.kangtaeho.com/91?category=768316>

<https://injo.tistory.com/31>