안녕하세요. 이번 프로젝트의 발표를 맡은 남재현이라고 합니다.
우선, 진행은 EDA, 전처리, 모델링 및 평가, 결론으로 진행하겠습니다.
시행착오는 각 슬라이드를 살펴볼 때 방향과 함께 말씀을 드리겠습니다.

< EDA >
데이터를 info 하고 나서 컬럼의 의미를 먼저 정의하였습니다. 하지만 버섯에 관련된 도메인 지식을 보유하고 있지 않아 정량적인 분석 후에도 의미있는 결과가 나오지 않는다면 간단한 논문 리딩 후 파생 변수를 만들어야겠다는 생각을 가진 채 시작하였습니다.

인포에서 인스턴스가 매우 많음을 확인하고 결측치를 우선 확인했습니다. 결측치가 50%이상인 컬럼을 우선 제거 후 진행하고, 모델의 성능이 부족하다면 위의 파생 변수 고려 시 삭제한 결측치 컬럼을 같이 고려 할 생각으로 드랍했습니다.

또 target 컬럼의 데이터 분포를 살펴보았습니다. 1:1에 거의 수렴하므로 SMOTE등을 사용하지 않고 그대로 진행하였습니다.

info를 봐주세요이제 본격적으로 인코딩 과정을 선택해야 했고 압도적으로 object 타입이 많기 때문에 이들을 전부 숫자형으로 변경할 수 없다면 '트리 기반 모델'을 사용해야겠다고 생각하였습니다. 트리 기반 모델은 데이터의 분포나 스케일에 민감한 모델이 아니므로 정규화/스케일링도 우선 제외 후 진행하겠다는 방향을 설정하였습니다.

전 슬라이드에서 말씀드린 50% 이상이 결측치인 컬럼을 제거한 후에 우선적으로 계절성을 가지고 있는 season컬럼에 레이블 인코딩을 진행하고 혹시 모를 타 모델 사용을 대비하여 class도 같이 진행하였습니다.

중복행들을 제거한 후에 모든 컬럼의 고유값을 살펴보며 상관 관계를 어떻게 분석할 지 고민하였고, 많은 object 타입을 숫자형으로 인코딩하기 위한

방법들의 장단점을 비교해보았습니다.

대부분의 컬럼이 수십개 이상의 고유치를 가지고 있으므로 원-핫 인코딩을 시행하면 차원 축소가 너무 힘들 것이라 판단하여 기각하였습니다.

또한, 남은 컬럼들이 순서를 가지고 있다는 정성적인 판단 하에 Ordinal 인코딩도 제외한 후

초반에 저희 팀이 설정한 트리 기반 모델과 적합하고, 고유치가 많은 상황에 사용하기 편한 인코딩으로
범주형 값마다 평균값으로 인코딩하는 타겟 인코딩과
각 범주를 이진 벡터로 변환하는 바이너리 인코딩을 고려하고 이제 결측치로 넘어갑니다.

일단 여기까지, 고려하고 있는 인코딩 방법은 2가지, 사용 할 모델은 트리 기반이며, 성능 지표가 좋지 않다면 드랍했던 결측치가 많은 컬럼과 파생 변수를 고려한다는 생각을 가지고 있었습니다.

다음 슬라이드부터는 여기에 보이는 대로 본격적으로 시각화를 하기 위해 드랍한 컬럼 제외 남은 컬럼의 결측치를 채움과 동시에 고윳값을 줄이는 과정을 시작합니다.

---

target 컬럼 제외 float형과 object형만 남았기 때문에 각 컬럼 타입끼리 묶어 비교하였습니다. train 데이터 중에 결측치가 존재하는 float형은 cap-diameter 컬럼이 유일했으므로 데이터의 분포를 왜도를 통해 살펴보았습니다. 왜도가 양수이므로 오른쪽 꼬리 분포를 가집니다. 오른쪽 꼬리 분포는 주로 평균>중앙값>최빈값순으로 기술 통계량이 분포하므로 결측치를 중앙값 대체를 진행했습니다.

후에 object 타입 컬럼들의 결측치를 채우기 위한 방법을 찾던 도중 일정 범주 이하의 데이터들을 Unknown으로 묶어 과소적합을 방지하는 방법을 찾게 되었습니다. 이 방법과 결측치를 다른 통계량으로 채우는 것도 원본 데이터의 손실이며 컬럼의 의미를 바꾸는 건 매한가지지만
저희가 생각하기에 Unknown으로 변경하는 방법이 기존의 데이터들의 영향력, 다시 말해서 예측 시 작용하는 가중치같은 역할을 덜 방해한다고 생각하였습니다.

그래서 누적백분율 99% 이상의 데이터 타입을 가진 컬럼들의 class 분포를 살펴보았고 전체 독버섯의 3%정도 만을 차지하기 때문에 unknown으로 묶어도 예측의 큰 차질이 없을 거라 생각하였습니다.

만약 약 8만개 중 독버섯의 비율이 굉장히 높았다면 이렇게 진행하지 않았겠지만
하지만 이 지점에서 약간의 데이터 손실을 감수하고 진행하였습니다.

이 슬라이드는 방금 설명드린 실행 결과입니다. 잘 보면 일부 데이터가 Unknown으로 대체된 것을 알 수 있습니다.

운이 좋게도 별 다른 처리를 하지 않았는데 object 타입의 결측치가 전부 사라진 것을 볼 수 있습니다. 이게 정말 날것의 공정 데이터가 아니기때문에 사실 이 지점에서 의심없이 이 방법이 독버섯 data 셋에서 가장 맞는 결측치 대체라는 확신을 하였습니다.

이제부턴 이상치입니다. 결측치와 마찬가지로 float와 object 나눠서 진행하겠습니다.
이상치의 존재 유무를 확인하기 위해 시각화를 진행하였고 위쪽의 꼬리가 긴 것으로 보아 이상치가 존재함을 알 수 있었습니다.

단순히 눈으로 판단한 것이기때문에 Z점수를 관찰해보았고 일반적으로 3시그마를 넘어가는 데이터를 이상치로 처리하는데, 훨씬 넘어가는 max 값을 볼 수 있습니다. 여기서 저는 3시그마를 넘기는 데이터들을 전부 제거해볼까 생각하였지만 남은 약 300백만개 데이터 중의 30%이상을 차지하기 때문에 다른 방법을 찾았습니다.

그 방법은 isolation forest 입니다. 이는 트리 기반의 이상치 탐색 방법으로 대규모 데이터셋과 다양한 특성의 데이터가 섞인 지금 사용하기 좋다고 판단하였습니다. 결과적으로 float 타입의 이상치를 처리하였습니다. 줄어든 투플의 수가 전체의 1%도 안되므로 데이터 손실을 적게 감수하였다고 생각하였습니다.

다음 과정은 당연히 object타입을 진행해야 하고, 각 컬럼의 데이터 별로 어느 데이터가 독버섯을 가장 많이 담고 있는지를 시각화 해보았습니다. 만약 한 컬럼에 a부터 e까지 고유치가 존재하는데 e라는 값을 가진 버섯들이 높은 비율로 독버섯이라면? 이를 학습에 반영하기 위해서입니다.

한 개를 제외한 모든 컬럼들은 방금 설명드린 비율이 비슷하였습니다. 하지만..! 다음 슬라이드가 저희 팀의 가장 key 입니다.

독버섯은 봄과 가을에 even하게 익는 걸 관찰하였습니다. 독버섯 중의 약 76%가 봄과 가을에 있었습니다. 이거 뭐 차원 증가의 위험을 감소해서라도 새로운 변수를 만들어야겠다고 생각하고 봄 또는 가을.. 낭만 넘치는 이진 변수를 생성하였습니다.

앞서 말씀드린 모든 전처리를 진행한 결과입니다. 별다른 인코딩을 진행하지 않았기 때문에 타입은 그대로지만 고유값은 많이 줄어든 것을 볼 수 있죠. 천개 이상의 고유값을 가진 컬럼은 전부 float 형입니다.
사실 pca같은 차원 축소도 진행되지 않은 상황입니다. 당연히 object형이 많으니까요? 이제 모델링 후의 평가 지표가 부족하면 돌아와서 인코딩을 진행하겠습니다.

저희 팀원들은 트리기반모델 중 light GBM, cat boost, XGboost 모델을 선정하였습니다. 각 모델의 장단점은 추후 비교 때 진행을 하기로 하고, 진행 결과를 살펴보겠습니다.

light GBM의 특징은 범주형 범수를 전부 카테고리형으로 변환해야 한다는 점? 또한 light GBM을 위한 전용 객체가 존재한다는 것이 있습니다.
검증 데이터 셋은 일반적으로 8:2로 분류하므로 20%를 가져갑니다.

처음 선정한 모델의 파라미터 값은 기본값 수준의 파라미터였습니다. 그래서 학습 후 평가지표를 산출하였더니 정확도, F1스코어가 약 0.97에서 머물렀고 이때만해도 사실 과적합을 의심하고 있었습니다. 하지만 다행히 혼동행렬을 산출해봤을 때 전체 검증 데이터 셋 60만개중의 False Positive는 4891 False negative는 5206으로 합쳐서 약 1.7%를 차지했으므로 하이퍼 파라미터 튜닝을 진행하였습니다. 아래에 튜닝을 통해 얻은 인잣값으로 대체를 하였습니다.

이게 검증 셋을 통한 스코어들인데 너무 작아서 안보이죠? 앞에서 설명드렸던 것과 동일합니다. 하이퍼 파라미터 튜닝을 거친 해당 모델로 submission 파일을 작성한 후에 캐글에서 최종 스코어 0.96을 얻었고 검증 스코어와 많은 차이가 나지 않으므로 모델의 일반화를 잘 거쳤다고 생각합니다.

다음은 Catboost 모델입니다. 저희 팀이 가장 처음에 시도했던 모델로 light GBM과 마찬가지로 오브젝트 컬럼을 카테고리형으로 변환해야 하고 데이터셋 Pool을 생성하는 특성이 있습니다. 기본적으로 이 모델은 앞의 데이터만 이용하여 과적합을 방지하므로 낮은 스코어가 나와도 일반화가 잘 되었다고 볼 수 있습니다. 이 모델의 스코어는 ~~~이고 캐글의 제출 스코어는 0.93으로 lightGBM보다 낮은 스코어를 보입니다.

개인적으로는 데이터 셋의 순서가 존재한다고 생각하지 않아 일반화가 잘 되었다고는 단정지을 수 없지만, 높은 스코어를 얻었음을 알 수 있습니다.