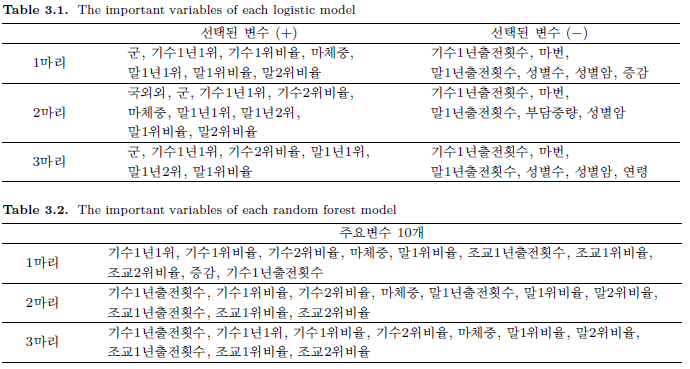
**데이터기반학습 Project 3 보고서**

(scikit-learn 0.21.0 버전 사용)

5조 정유민, 문현지, 한태구

1. Data Analyze / Background Search

초기에 주어진 데이터 변수들에 대해 단순 상관계수 분석을 한 결과, 대부분의 변수들은 순위와의 상관계수 절대값이 0.1보다 작은 것을 확인하였다. 상관계수를 이용한 변수 선택은 모델의 성능 개선에 유의미한 효과를 줄 것 같지 않아, 대신 본 프로젝트와 같은 데이터를 사용한 경마 예측에 관한 논문인 ‘Analysis of Horse Races: Prediction of Winning Horses in Horse Races Using Statistical Models’을 참고하여 데이터를 전처리하기로 하였다.



위의 도표에서 알 수 있듯이, 크게 말과 관련된 변수(군, 국외외, 마체중)와 통산 순위와 관련된 변수(기수1년1위, 기수1위비율 등)로 나누어졌다. 말 소유자에 대한 데이터는 아예 사용하지 않은 것으로 나타났다. 위의 도표에서 선택된 주요 변수들 및 그 외의 변수들을 heuristic하게 선정하고 2절의 전처리를 통해 만들었다.

2. Data Preprocessing

(1) Column Selection

Column의 종류가 많기 때문에 미리 실시한 통계적 분석과 관련 논문에서의 자료를 바탕으로 Heuristic하게 column을 선택하였다. 수치형 데이터의 경우, 위의 표에서 언급한 데이터를 사용하였고, birth와 age처럼 비슷한 역할을 하는 중복된 column들을 제외하였다. 범주형의 경우, 대부분 상관관계가 없었기에 위의 논문에서 주요변수로 언급된 horse의 Gender(성별)과 Home(출산지)를 추가로 사용하였다.

(2) Feature Preprocessing

첫째로, 비정량데이터인 Gender와 Home의 경우 범주형 데이터이므로 get\_dummies() 함수를 활용하여 one-hot encoding을 하였다. Gender는 암말(F), 수말(M), 거세말(G) 세 종류 각각을 분류하여 사용했지만, Home은 한국산이 대부분이고 그 외의 경우 각 나라마다 수가 적으므로 한국산(K)과 그렇지 않은 것 두 가지로만 분류하여 사용했다. 또한, 1위와 2위 횟수는 전체 경기 수에 의존적이므로 이를 **전체 경기 수 대비 비율로 바꾸어** ‘최근 1년 1&2위 비율’과 ‘통산 1&2위 비율’라는 변수로 사용하였다(‘horse\_1yr\_win 'horse\_total\_win’, 'jockey\_1yr\_win’, 'jockey\_total\_win’, 'trainer\_1yr\_win’, 'trainer\_total\_win'). 마찬가지로, horse\_money(총 상금)의 경우에도 경기 수가 많을수록 커지는 값이므로 전체 경기 수로 나눈 값을 사용하였다. 마지막으로, Jockey와 horse의 아주 최근 순위가 현재의 실력을 반영한다고 생각하여 **최근 3개의 경기에 대해 rank를 평균 낸 것을 사용**하였다(‘j\_rank’, ‘h\_rank’). 이를 반영한 input 변수 리스트는 다음과 같다.



(3) Normalization

위의 변수를 그대로 사용하면, 동일한 말(x)이 다른 순위결과(y)를 가지게 되어 학습을 어렵게 만든다. 따라서 수치형 변수('price’, 'horse\_money’, 'rating’, ’horse\_1yr\_win’, ‘horse\_total\_win’, ‘jockey\_1yr\_win’, 'jockey\_total\_win’, 'trainer\_1yr\_win’, 'trainer\_total\_win’, 'j\_rank’, 'h\_rank')에 대해서는 **경기 별로 z-normalization을 수행하여 ‘norm\_’ 전치사가 붙은 변수로 input에 추가**하였다.

(4) Data Cleaning

NaN 데이터의 경우 merge를 위해 필요한 말, 기수 등에 대한 데이터의 누락으로 발생한 것이 다수이므로, 이를 0으로 채우기 보다는 **다른 데이터들의 평균으로 채워주었다**.

3. Data Selection

이렇게 전처리된 변수들에 대해 l1-norm기반의 LinearSVC를 수행하고 scikit-learn에서 제공하는 **SelectFromModel() 모듈에 넣어주어 자동으로 변수를 선택**해주는 작업을 수행하였다. 최종 제출 파일에서, 총 29개 중 20개의 변수가 자동 선택됨을 확인할 수 있었다.

4. Parameter Tuning

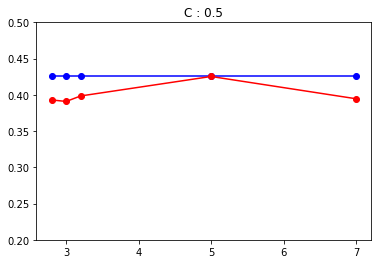
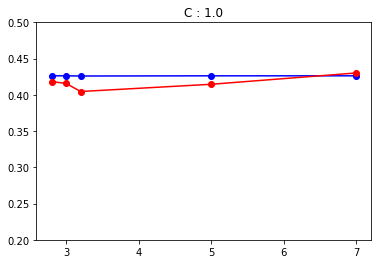
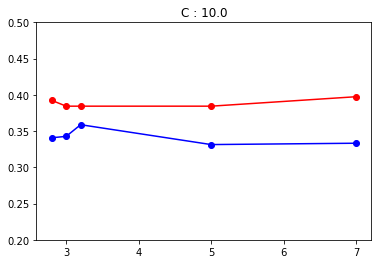
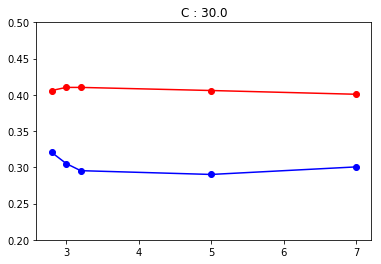
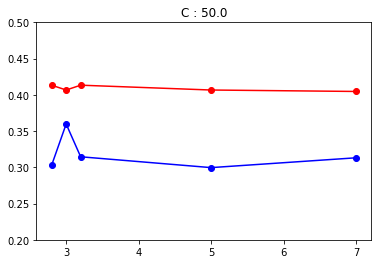
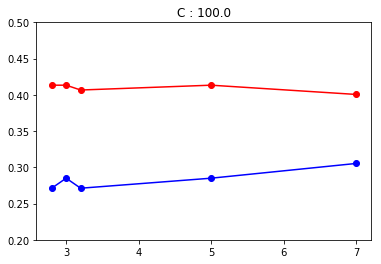
(1) Validation

Validation시 기본적으로 walk forward 방식을 사용하였다. 최종으로 테스트하는 set이 2주 단위이므로 validation set의 단위도 2주로 하였고, 총 validation을 너무 크게 하지 않기 위해 walk의 단위는 1주, fold의 개수는 10개로 설정하였다.

(2) Parameter Tuning

사전 테스트 결과 gamma는 ‘auto’가 제일 성능이 좋아 고정하였고, max\_iter의 경우 5000이 넘어가면 성능의 변동이 거의 없어 빠른 학습을 위해 5000으로 설정하였다. 또한 Kernel의 경우 poly는 비교적 성능이 좋지 않아 배제하였고, C의 경우 100을 넘어가면 100인 경우와 성능이 비슷해 100 이하인 경우에 대해서만 tuning을 진행하였다. 최종적으로 Kernel, C, class\_weight에 대해 다음과 같은 범위에서 grid search를 실시하였다.





위의 그래프 6개는 각 C에 대해 class\_weight(x축)의 변화에 따라 각 커널이 어떤 F1\_score을 보였는지 나타낸 그림이다. Kernel의 경우, 대체적으로 linear(빨간색)가 성능이 좋았으나 C가 작을 경우 rbf(파란색)의 성능이 더 좋거나 비슷하여 두 kernel 모두 사용하기로 했다. C의 경우, 10 이상에선 성능이 안정적이지 못해 1 이하로 설정하기로 하였다. Class\_weight의 경우, 변동이 미미하거나 일관적이지 않았다.

이를 확인하기 위해 predict\_proba() 함수를 통해 SVM이 리턴하는 클래스 확률을 보니 본 학습 데이터의 샘플 수가 너무 많아서(10800여 개)인지, 클래스 간의 불균형이 심해서인지 class\_weight가 잘 작동하지 않는 것으로 확인되었다. 관련하여 찾아보니, SVM은 샘플 수가 10000개 정도를 넘어갈 때는 학습이 어려우므로 사용을 권장하지 않는다고 한다. 따라서 클래스 불균형과 과잉 샘플 수를 해결하기 위해, **positive와 negative를 동일하게 샘플링하되 class\_weight를 None으로 설정하는 방법**을 고안하였다. 해당 방식으로 1000개씩 2000개를 샘플링하니, predict\_proba()와 predict()가 동일한 결과를 리턴하며 F1\_score가 0.55이상으로 높게 나오는 것을 확인하였다. 샘플링 수는 1000개 외에도 실험해보았으나, 1000개보다 적을 때는 성능이 떨어지며 1000개보다 많을 때는 1000개일 때와 성능 차이가 없는 것을 확인했다.

(3) Final Model(Ensemble)

샘플링 방식은 어떤 샘플이 선택되느냐에 따라 성능이 달라질 수 있으므로(안정성이 떨어지므로) 9개의 모델을 각기 다른 샘플로 학습하여 앙상블하기로 하였다. 위의 파라미터에서 다양한 앙상블 조합으로 테스트 해본 결과 **[‘rbf' C=0.8]을 3개, [‘linear’ C=1.0]을 6개 앙상블** 하는 것이 가장 robust한 것으로 확인되었다. 대부분의 모델은 accuracy가 낮고 recall이 지나치게 높으므로, 단순히 평균값을 사용하지 않고 각 모델이 예측하는 1에 대한 확률을 모두 더하여 **각 경주마다 그 값이 가장 큰 4마리 말만 1로 예측하는 방식으로 pred\_y를 생성**하였다. (‘4마리’는 실험적으로 결정)

4. Result

Test day=['2019-04-13’, '2019-04-14’, '2019-04-20’, '2019-04-21']에 대한 결과는 다음과 같다.

