머신러닝으로 게임봇 탐지

백진우, 정희영, 최종문, 김정학

연세대학교 빅데이터 학회 “YBIGTA”

Ⅰ. 서론

참고문헌 (1) 논문의 컬럼을 많이 참고하였다. 그 외에도 봇 유저가 일반 유저보다 퀘스트 완료를 덜 했을 것이라는 생각으로 퀘스트 관련 변수를 새롭게 추가하였다. 데이터 크기가 커서 GCP에서 데이터 정제 작업을 진행했으며 필요한 로그를 뽑아낸 후에는 로컬에서 데이터 칼럼 수정 및 모델링 작업을 수행했다. 훈련데이터를 랜덤포레스트, XGBoost 등 여러 머신러닝 방법으로 학습했으며 그 중 f1 score가 가장 높은 모델을 채택하기로 한다.

Ⅱ. 방법론

생성한 변수들을 조정해가며 여러 머신러닝 모델을 실험했다. 가장 좋은 성능을 낸 데이터셋은 다음과 같다. 로그에서 계정 별로 추출한 변수들(아이템 획득 수, 경험치 획득 수, 포탈 사용 수, 앉은 횟수 등)에 일별 변수를 추가했다. 일별 변수는 ‘per_day’의 이름을 붙였으며 같은 시간 단위마다의 행위가 게임 플레이스타일을 더 특징짓는다고 생각했다. 그렇게 나온 변수들을 서로 나누어 n^2개의 새 변수 중 피어슨 상관관계가 0.1이 넘는 변수들만 추려 상관관계의 원인을 추론할 수 있는 컬럼만 새로 추가했다. 예를 들어 teleport_count/play_time이 상관관계가 높게 나온 변수 중 하나인데, 봇들은 맵 이동을 적게 하고 사냥에 집중할 했기에 상관관계가 있다고 추론했다. 그 후 다시 전체 변수에 대해 Lasso 분류기로 학습을 하여 model based feature selection을 해서 변수들을 뽑았고, 그와는 별개로 전체 변수에 대해 정

답과의 피어슨 상관계수 분석을 하여 랜덤 변수보다 높은 상관관계를 가진 변수들을 또 뽑았다. 그렇게 뽑힌 두 개의 변수 집단을 비교하여 양쪽에 모두 속한 변수는 사용하였고, 한 쪽에만 속한 경우에는 변수가 중첩되는 경우에는 제거해서 최종 32개의 변수를 얻었다.

사용한 모델은 랜덤포레스트와 로지스틱 회귀, XGBoost이며 Grid Search를 통해 최적의 파라미터를 찾아 학습을 진행했다. 또한 일반 유저와 봇의 라벨링이 불균형하게 분포하므로 훈련 데이터에 스모트 처리를 해주어야 한다고 생각해, 스모트 처리 여부를 적용해 f1 score를 측정했다.

III. 탐지결과 및 평가

1. 랜덤포레스트(스모트 처리 안함)

```
In [11]: from sklearn.metrics import classification_report

         print(classification_report(y_test, y_pred_rf, target_names=['Human', 'Bot']))
```

```
              precision    recall  f1-score   support

       Human       0.98      1.00      0.99      2244
         Bot       0.95      0.70      0.80       186

    accuracy                           0.97      2430
   macro avg       0.96      0.85      0.90      2430
weighted avg       0.97      0.97      0.97      2430
```

```
In [12]: from sklearn.metrics import accuracy_score, f1_score
         print("Accuracy: %.2f" %accuracy_score(y_test, y_pred_rf))
         print("F1 score: %.2f" %f1_score(y_test, y_pred_rf))
```

```
Accuracy: 0.97
F1 score: 0.80
```

2. 랜덤포레스트(스모트 처리)

```
In [21]: from sklearn.metrics import classification_report

         print(classification_report(y_test, y_pred_rf, target_names=['Human', 'Bot']))
```

```
              precision    recall  f1-score   support

       Human       0.99      0.99      0.99      2244
         Bot       0.83      0.83      0.83       186

    accuracy                           0.97      2430
   macro avg       0.91      0.91      0.91      2430
weighted avg       0.97      0.97      0.97      2430
```

```
In [22]: from sklearn.metrics import accuracy_score, f1_score
         print("Accuracy: %.2f" %accuracy_score(y_test, y_pred_rf))
         print("F1 score: %.2f" %f1_score(y_test, y_pred_rf))
```

```
Accuracy: 0.97
F1 score: 0.83
```

3. 로지스틱 회귀분석(스모트 처리)

```
[ ]  from sklearn.metrics import accuracy_score, f1_score
     print("Accuracy: %.2f" %accuracy_score(y_test, y_pred))
     print("F1 score: %.2f" %f1_score(y_test, y_pred))
```

```
Accuracy: 0.94
F1 score: 0.64
```

4. XGBoost(스모트 처리 안함)

```
[16]  from sklearn.metrics import classification_report

      print(classification_report(y_test, prediction, target_names=['Human', 'Bot']))
```

```
              precision    recall  f1-score   support

       Human       0.98      0.99      0.99      2259
         Bot       0.91      0.77      0.84       171

    accuracy                           0.98      2430
   macro avg       0.95      0.88      0.91      2430
weighted avg       0.98      0.98      0.98      2430
```

```
from sklearn.metrics import accuracy_score, f1_score
print("Accuracy: %.2f" %accuracy_score(y_test, prediction))
print("F1 score: %.2f" %f1_score(y_test, prediction))
```

```
Accuracy: 0.98
F1 score: 0.84
```

5. XGBoost(스모트 처리)

```
[21]  from sklearn.metrics import classification_report

      print(classification_report(y_test, prediction, target_names=['Human', 'Bot']))
```

```
              precision    recall  f1-score   support

       Human       0.98      0.99      0.99      2259
         Bot       0.91      0.77      0.84       171

    accuracy                           0.98      2430
   macro avg       0.95      0.88      0.91      2430
weighted avg       0.98      0.98      0.98      2430
```

```
[22]  from sklearn.metrics import accuracy_score, f1_score
      print("Accuracy: %.2f" %accuracy_score(y_test, prediction))
      print("F1 score: %.2f" %f1_score(y_test, prediction))
```

```
Accuracy: 0.98
F1 score: 0.84
```

Ⅳ. 결론

	랜덤포레스트 (스모트X)	랜덤포레스트 (스모트O)	로지스틱회귀분석 (스모트O)	XGBoost (스모트X)	XGBoost (스모트O)
Accuracy	0.97	0.97	0.94	0.98	0.98
F1_score	0.80	0.83	0.64	0.84	0.84

모델의 경우, 결과적으로 XGBoost의 매개변수를 조율해서 사용했다. XGBoost의 f1 score가 0.84로 가장 높았다. 파라미터 조율 과정에서는 GirdSearchCV를 사용했다. Xgboost을 통해서 변수를 한 번 더 걸러내려고 시도했지만, 모델에서 변수들을 비교적 균일하게 반영하였기에 따로 빼지는 않았다.

[참고문헌]

[1] Kang, A. R., Jeong, S. H., Mohaisen, A., & Kim, H. K. (2016). Multimodal game bot detection using user behavioral characteristics. *SpringerPlus,* 5(1), 1-19.