EODS Project 1

- Instagram Fake Users Identifier

By: Lu Yang, Hannah Mei, Jenny Li and Xiao Yin (Team 16)

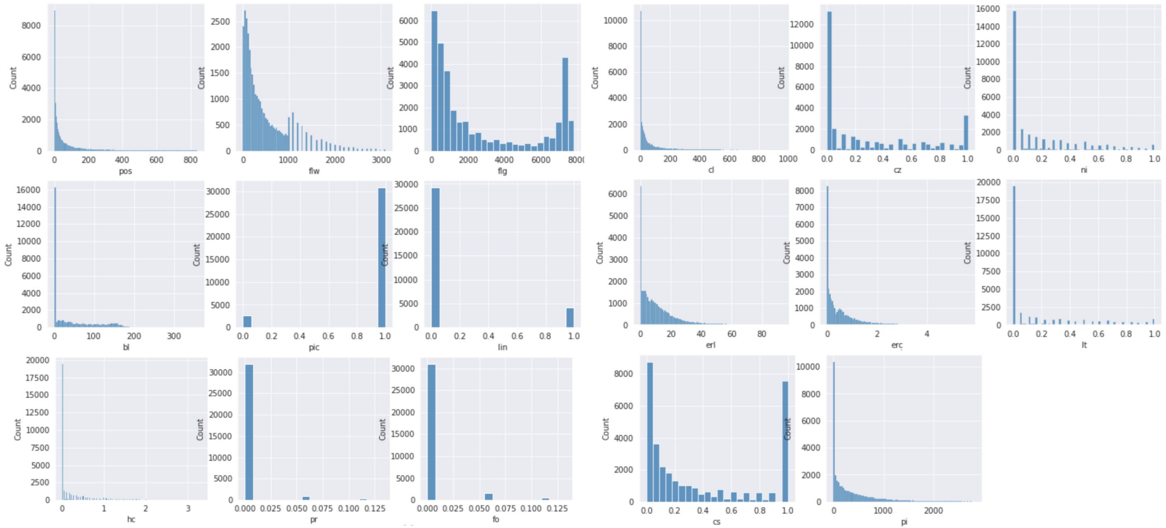
Contents

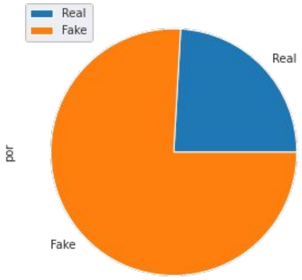
Questions:

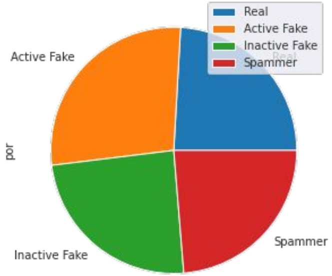
1. How to identify fake users,
2. Moreover, how to identify spammers among fake users?

EDA- Feature distributions

Kaggle: https://www.kaggle.com/datasets/krpurba/fakeauthentic-user-instagram







- 17 features in total
  - ‘pos’ | Num posts, ‘flg’ | Num following, ‘pi’ | Post interval, ‘cl’ | Average caption length…
  - 2 categorical: ‘pic’ | Profile pic availability, ‘lin’ | Link availability
- 2 Class: “Fake” vs “Real”
- 4 Class: "Real" vs "Active Fake" vs "Inactive Fake" vs "Spammer"
- Most of the features are very skewed with large outliers and many zeros
- Label proportion in pie chart

Data Processing

Feature Engineering on Y

1. Adding a 2 - class label ***y2*** indicating fake users (r, a/i/s)
2. One-hot-encoding on ‘class’ (***y4***)

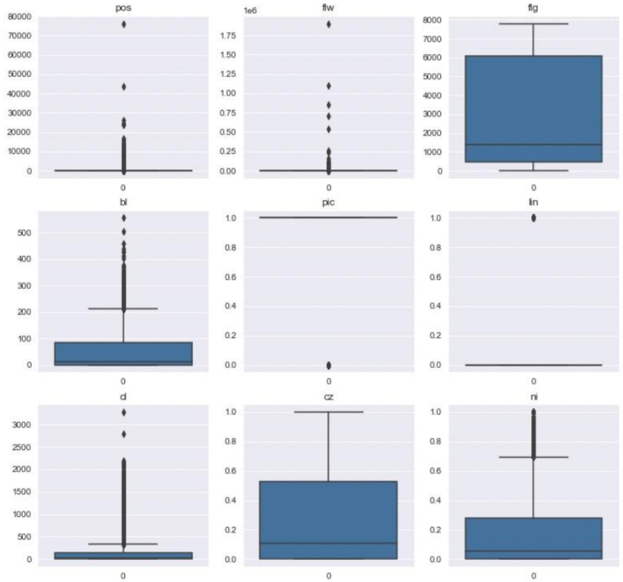
Feature Engineering on X

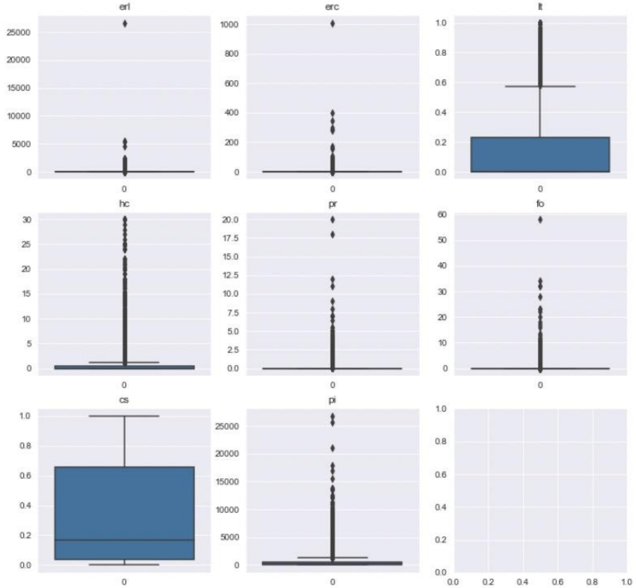
1. Checking repeated rows
2. Splitting into train set and test set
3. Standardizing

Feature Engineering on X

Detecting outliers

Before IQR



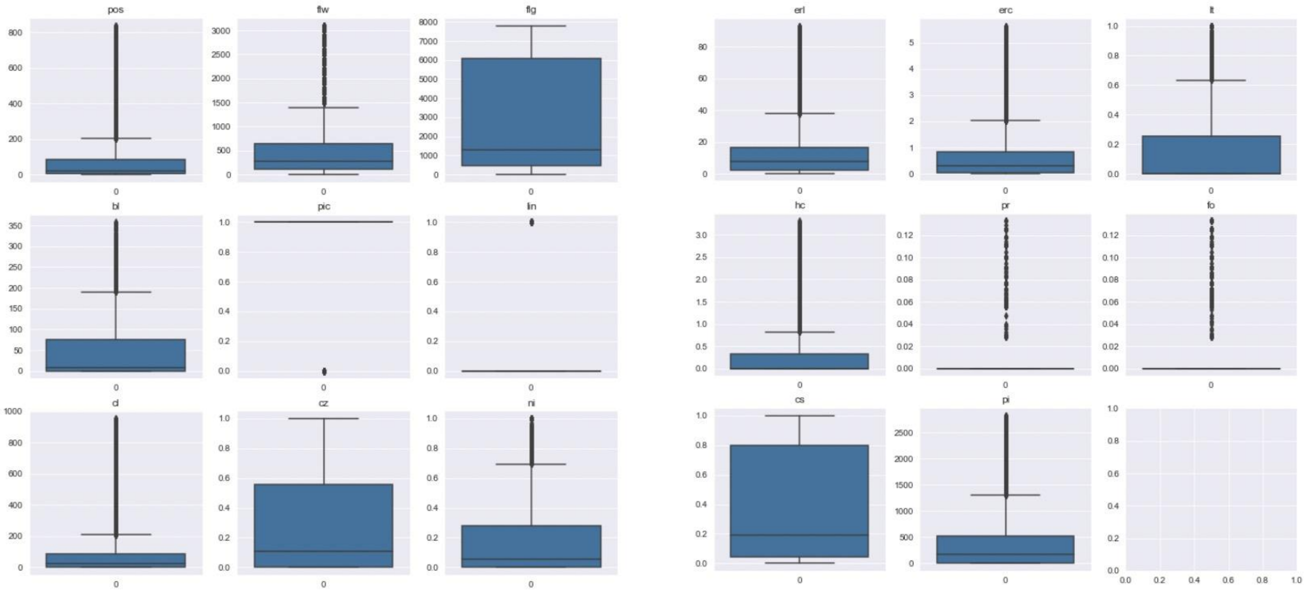


43307 rows × 17 columns

Feature Engineering on X

Detecting outliers

After IQR



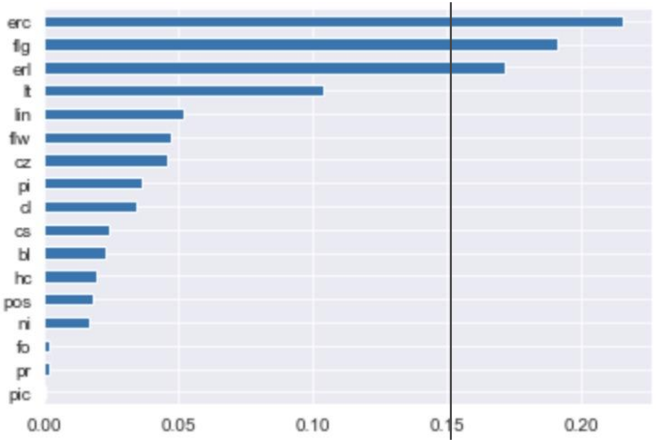
33512 rows × 17 columns

Feature Selection

Tree Based Model Feature Importance

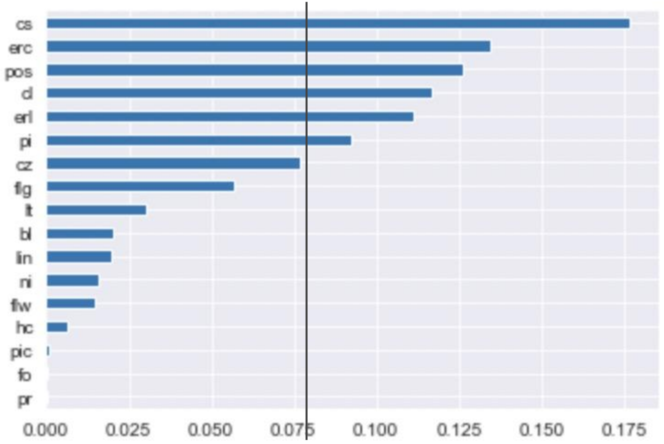
2 - class

{'max_depth': 9, 'n_estimators': 23}



4 - class

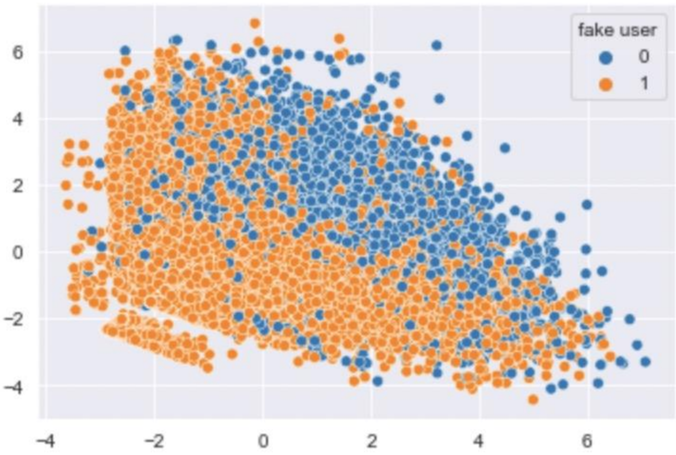
{'max_depth': 9, 'n_estimators': 29}



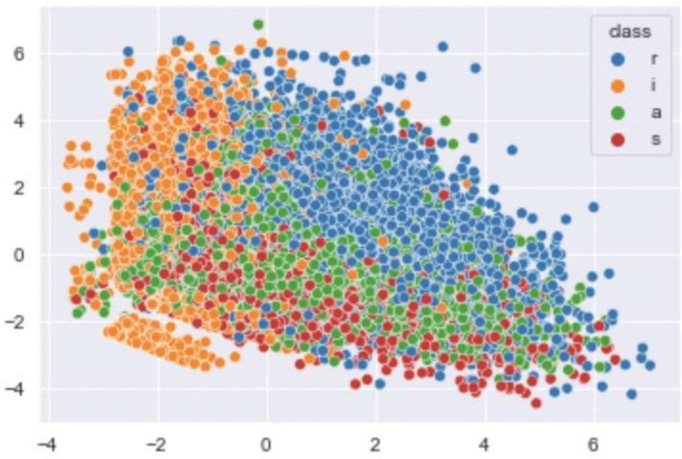
Feature Extraction

PCA (90% explained variance ratio — Dimensions: 17 -> 13)

2 - class

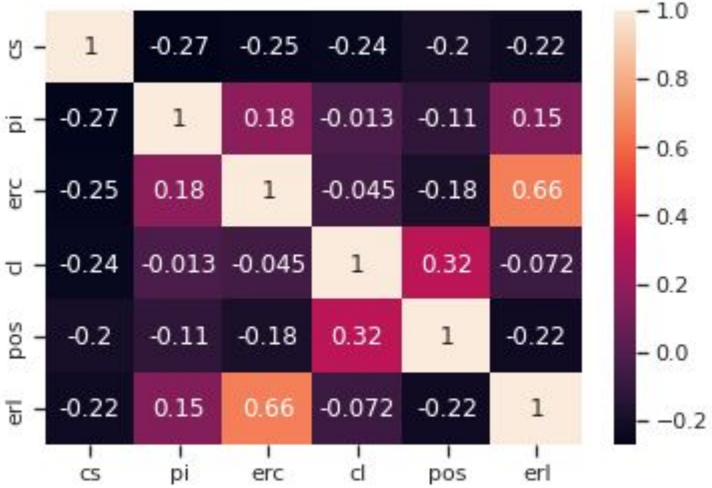


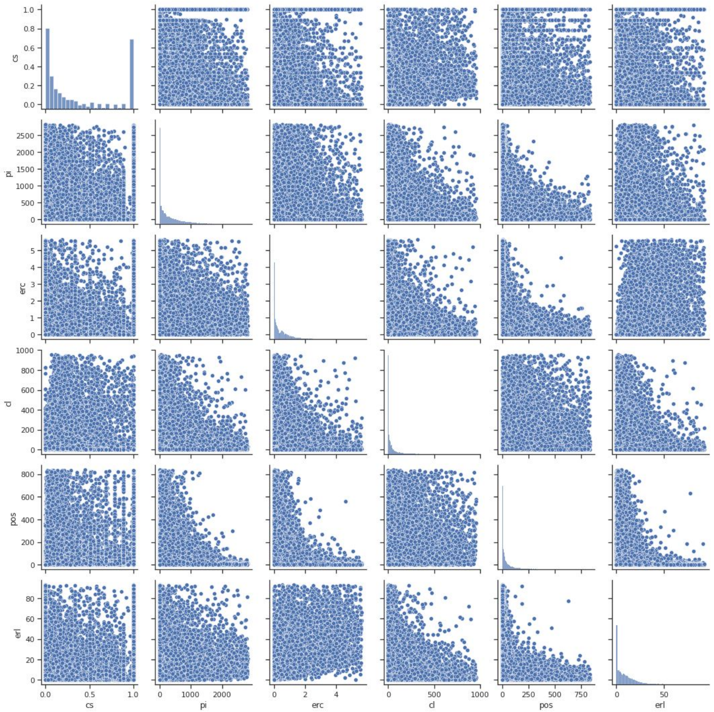
4 - class



Correlation Matrices

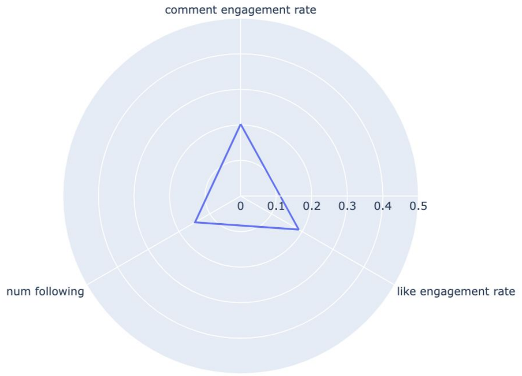
4-class selected features (2-class matrices are in the same format)

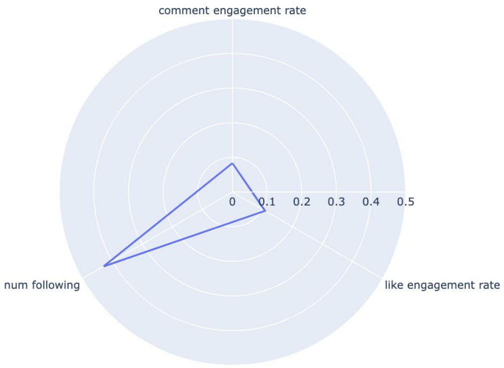




- Check feature correlations and multicollinearity
- No particularly high correlation between the features except between erc and erl
- Highest abs= 0.66
- Impact on Model selection → Non-parametric Models

Radar Plots by Class (2 Class)

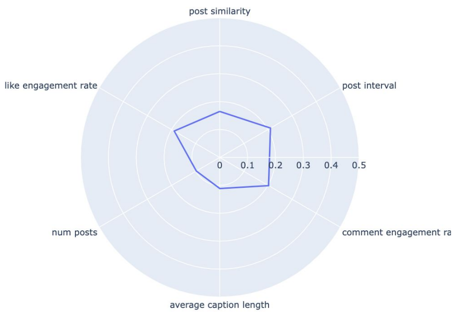




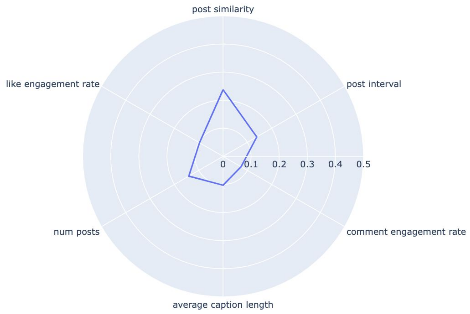
Scaled and averaged the 3 most important features for 2-class case

- Some pattern can be uncovered using radar graph
- Number of following is on average higher among fake users
- comment/like engagement rate are much low

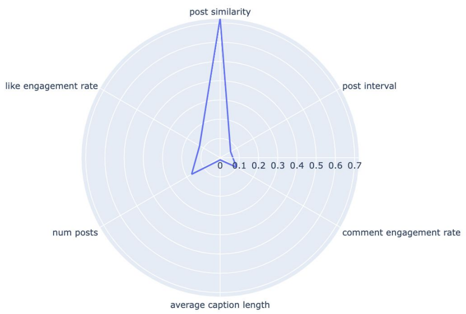
Radar Plots by Class (4 Class)



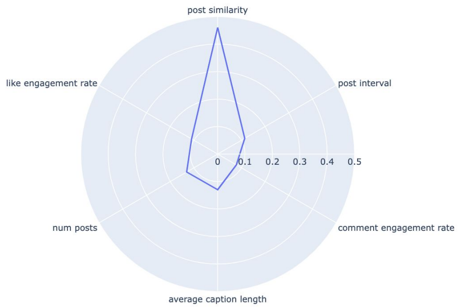
real users



active fake user



inactive fake user



spammer fake user

- Scaled and averaged the 6 most important features to compare
- Compare to the real users, all 3 categories of fake users have more similar pattern
- “post similarity” is very high among fake
- “Engagement rates”, “post interval” and “average caption length” are especially low among fake users

Two Class Classification: Random Forest Classifier

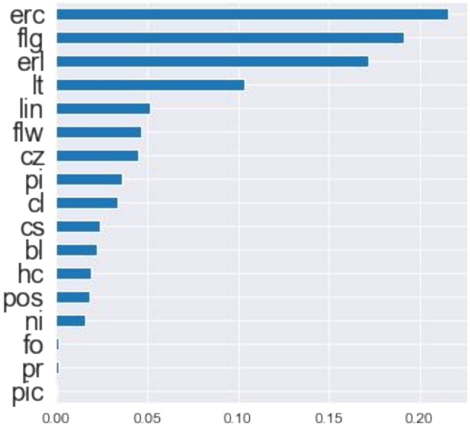
1. Tune Hyperparameters

```
params2_rf = {'n_estimators':[i for i in range(10,30)],
          'max_depth':[i for i in range(3,10)]}
gscv2_rf = GridSearchCV(RandomForestClassifier(random_state=123),
                        param_grid=params2_rf, cv=5, refit=True)
gscv2_rf.fit(x2_train, y2_train)
print(gscv2_rf.best_params_)
```

```
{'max_depth': 9, 'n_estimators': 23}
```

➔ Best (‘n_estimators’, ‘max_depth’) : (23, 9)

2. Select Features

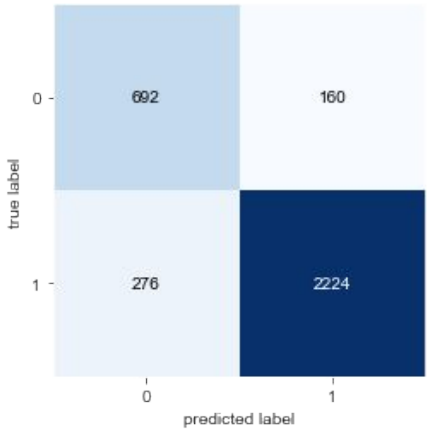


➔ Selected Features: erc, flg, erl

3. Calculate Accuracy Score

0.8699 on Test Dataset

4. Plot Confusion Matrix



Two Class Classification: Gradient Boosting Classifier

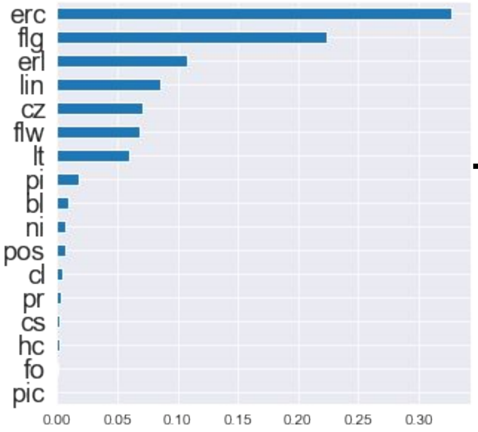
1. Tune Hyperparameters

```python
mean_scores = []
for rate in [0.05, 0.1, 0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.7,
            0.75, 0.8, 0.85, 0.9,0.95]:
    gbc = GradientBoostingClassifier(learning_rate = rate,random_state = 123)
    scores = cross_val_score(gbc, x2_train, y2_train, cv = 5)
    mean_scores.append((rate, scores.mean().round(4)))
sorted(mean_scores, key=lambda x:x[1],reverse=True)[0]
```

```
(0.5, 0.9302)
```

➔ Best learning_rate: 0.5

2. Select Features

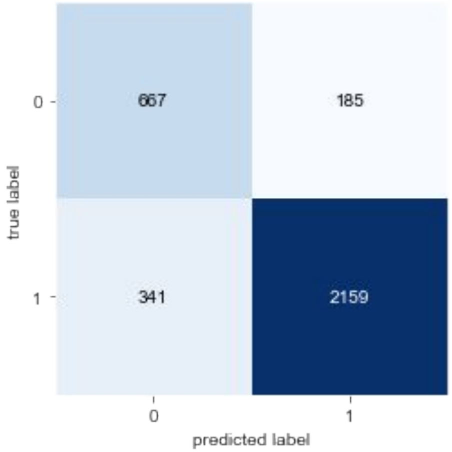


➔ Selected features: erc, flg

3. Calculate Accuracy Score

0.8431 on Test Dataset

4. Plot Confusion Matrix



Two Class Classification: KNN

1. Tune Hyperparameters

```
mean_scores = []
for n in [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]:
    knn = KNeighborsClassifier(n_neighbors = n)
    scores = cross_val_score(knn, x2_train, y2_train, cv = 5)
    mean_scores.append((n, scores.mean().round(4)))
sorted(mean_scores, key=lambda x:x[1],reverse=True)[0]
```

```
(11, 0.8705)
```

➔ Best n_neighbors:11

2. Calculate Accuracy Score

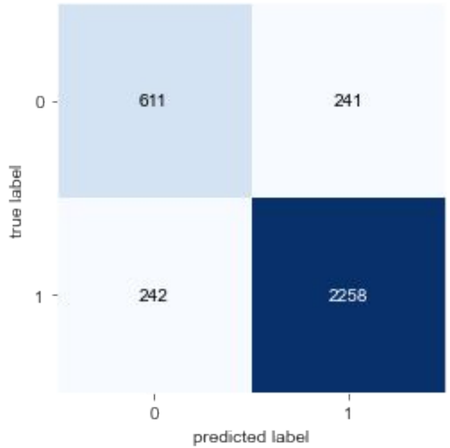
With Features Selected by Random Forest Classifier (erc, flg, erl) ➔ 0.8559 on Test Dataset

With Features Selected by Gradient Boosting Classifier (erc, flg) ➔ 0.8323 on Test Dataset

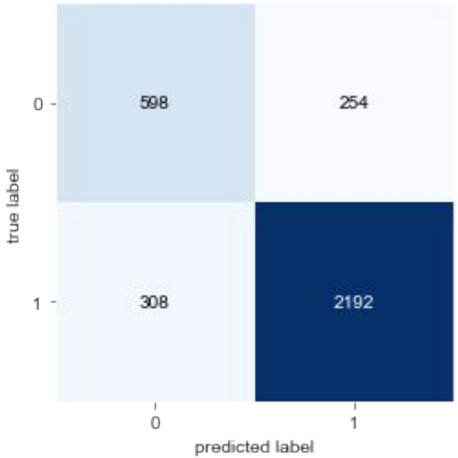
Two Class Classification: KNN Continued

3. Plot Confusion Matrix

With Features Selected by Random Forest Classifier (erc, flg, erl)



With Features Selected by Gradient Boosting Classifier (erc, flg)



Two Class Classification: Summary

	Accuracy_Score	Model	Hyperparameter	Feature	Recall	Precision
0	0.8699	RandomForestClassifier	max_depth=9, n_estimators = 23	erc,flg,erl	0.8896	0.9329
1	0.8431	GradientBoostingClassifier	learning_rate=0.5	erc,flg	0.8636	0.9211
2	0.8559	KNN	n_neighbors=11	erc,flg,erl	0.9032	0.9036
3	0.8323	KNN	n_neighbors=11	erc,flg	0.8768	0.8962

- If we want to best classify real and fake users in general (i.e. highest accuracy score), or increase the rate that we predict fake users correctly (i.e. highest precision)
  - Choose Random Forest Classifier
  - With max_depth=9, n_estimators=23
  - Use ‘erc’, ‘flg’, and ‘erl’ as features
- If we want to predict as many fake users among all fake users as possible (i.e. highest recall score)
  - Choose KNN
  - With n_neighbors=11
  - Use ‘erc’, ‘flg’, and ‘erl’ as features

Four-Class Classification

R: Authentic / real users

A: Active fake users.

I: Inactive fake users

S: Spammer fake users.

Business Goal:

- Identify as many spammers as possible;
- Give in-time alerts to users when users get requests or DM from spammers.

Four-Class Classification: Feature Analysis

Results of feature selection:

pos: Number of total posts that the user has ever posted.

cs: Average cosine similarity of between all pair of two posts a user has.

cl: The average number of character of captions in media.

erc: (num comments) divide by (num media) divide by (num followers).

erl: (num likes) divide by (num media) divide by (num followers).

Features of two-class model:

flg: Number of followers of the user.

erc: (num comments) divide by (num media) divide by (num followers).

erl: (num likes) divide by (num media) divide by (num followers).

Four-Class Classification: Model Comparison

Spammer' Recall	Model	Parameters
0.83	GradientBoosting	learning_rate=0.15
0.81	RandomForest	max_depth': 9, 'n_estimators': 29
0.67	KNN	n_neighbors=15

Aiming at detecting as many spammers as possible, we focus on the Recall of ‘S’, Spammer fake users:

Gradient Boosting Classifier;

learning_rate=0.15

Best "Recall" Model: Gradient Boosting Classifier

1. Tune Hyperparameters

```
mean_scores = []
for rate in np.linspace(0.05, 1, num=20):
    gbc = GradientBoostingClassifier(learning_rate=rate, random_state=123)
    scores = cross_val_score(gbc, x_train, y_train, cv=5)
    mean_scores.append((rate, scores.mean().round(4)))
sorted(mean_scores, key=lambda x:x[1],reverse=True)[0]
```

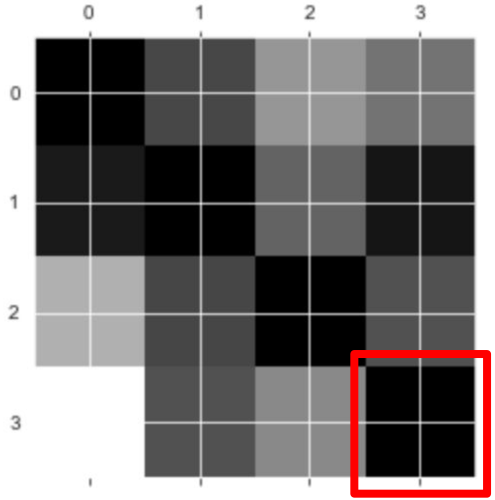
-> Best learning rate = 0.15

2. Fitting Results

-> Accuracy score = 0.54

-> Recall of 'S' = 0.83

Correctly identify 83% of spammers.



Summary & Future Work

Summary:

- First, detect as many fake users as possible
  - Use KNN
    - With n_neighbors=11; ‘erc’, ‘flg’, and ‘erl’ as features
    - Achieve 0.9032 recall score on test dataset
- Then, further detect as many spammer fake users as possible
  - Use Gradient Boosting Classifier
    - With learning_rate=0.15; ‘pos’, ‘cl’, ‘cs’, ‘erc’, ‘erl’, ‘flg’ as features
    - Achieve recall score for spammer = 0.83.

Future Work:

- Explore the impact of outliers
  - Rerun the experiment without dropping all outliers since outliers might represent intrinsic behavioral difference between real and fake users
- Variable correlation
  - Control correlated variables such as “erl” and “erc”
- Model improvement
  - Transform data
    - Satisfy the distributional assumptions of parametric models
    - Fit parametric models (e.g. Logistic Regression l1 penalty)
    - Test on new datasets

Q&A

Reference

- **Data source:**

K. R. Purba, D. Asirvatham and R. K. Murugesan, "Classification of instagram fake users using supervised machine learning algorithms," International Journal of Electrical and Computer Engineering (IJECE), vol. 10, no. 3, pp. 2763-2772, 2020.

K. R. Purba, D. Asirvatham and R. K. Murugesan, “Fake/Authentic User Instagram.” *Kaggle*, https://www.kaggle.com/datasets/krpurba/fakeauthentic-user-instagram?select=user_fake_authentic_4class.csv. Accessed 23 March 2022.

- **Code:**

https://colab.research.google.com/drive/1hzFwg63lsJsZXvdlHAxUAU5h3wXtml2P?usp=sharing