






Pattern Recognition

Final Project Presentation

Team 03

NCR + CatBoost
+ k-fold cross validation

F1 score : 0.5007
in test set

01. Data Exploration

Key Strategy

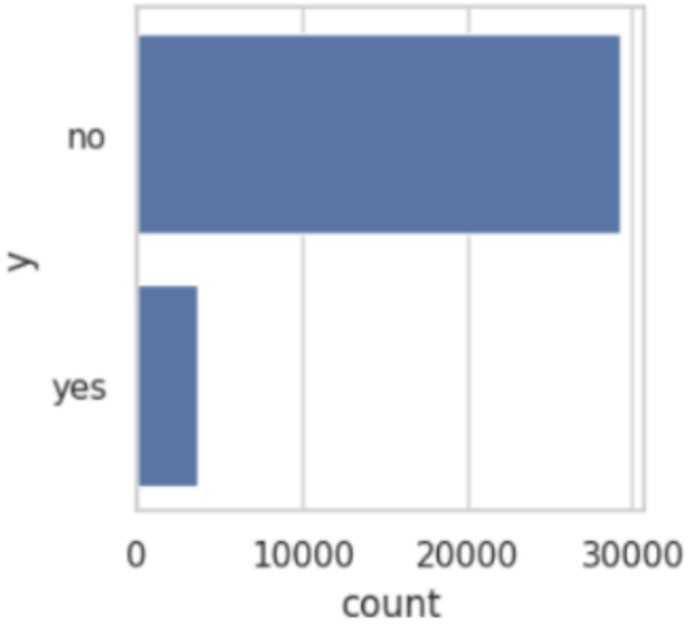
derivative
variable
"year"

Additional Attempt

preprocessing
duplicated
columns

excluding "id" and "y"

01. Data Exploration



What we got:
Highly
Imbalanced
data

02.Data Examination

```
[12] df_train.describe().T
```

	count	mean	std	min	25%	50%	75%	max
id	32950.0	20569.615569	11895.520420	1.000	10258.250	20571.000	30846.750	41188.000
age	32950.0	40.023703	10.401749	17.000				
campaign	32950.0	2.567830	2.766994	1.000				
pdays	32950.0	962.415964	187.054556	0.000				
previous	32950.0	0.172838	0.498098	0.000				
emp.var.rate	32950.0	0.083129	1.571951	-3.400				
cons.price.idx	32950.0	93.576610	0.578725	92.201				
cons.conf.idx	32950.0	-40.500091	4.632363	-50.800				
euribor3m	32950.0	3.622516	1.734791	0.634				
nr.employed	32950.0	5167.036495	72.250873	4963.600				

← train

↓ test

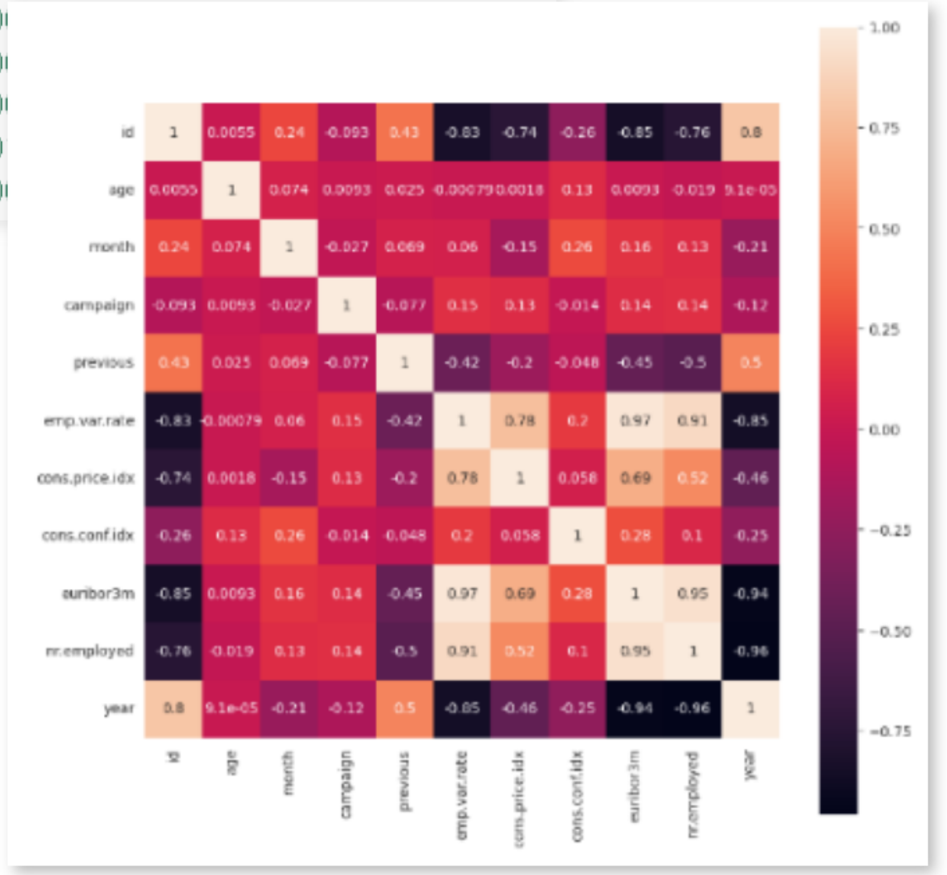
```
[13] df_test.describe().T
```

	count	mean	std	min	25%	50%	75%	max
id	8238.0	20694.031682	11868.573076	4.000	10457.750	20694.000	31056.250	41181.000
age	8238.0	40.025492	10.499522	17.000	32.000	38.000	47.000	94.000
campaign	8238.0	2.566642	2.782228	1.000	1.000	2.000	3.000	43.000
pdays	8238.0	962.713401	186.346378	0.000	999.000	999.000	999.000	999.000
previous	8238.0	0.173464	0.481933	0.000	0.000	0.000	0.000	5.000
emp.var.rate	8238.0	0.076912	1.567072	-3.400	-1.800	1.100	1.400	1.400
cons.price.idx	8238.0	93.571882	0.579322	92.201	93.075	93.444	93.994	94.767
cons.conf.idx	8238.0	-40.512637	4.611767	-50.800	-42.700	-41.800	-36.400	-26.900
euribor3m	8238.0	3.616390	1.733168	0.635	1.334	4.857	4.961	5.045
nr.employed	8238.0	5167.033576	72.258533	4963.600	5099.100	5191.000	5228.100	5228.100

show very similar statistics

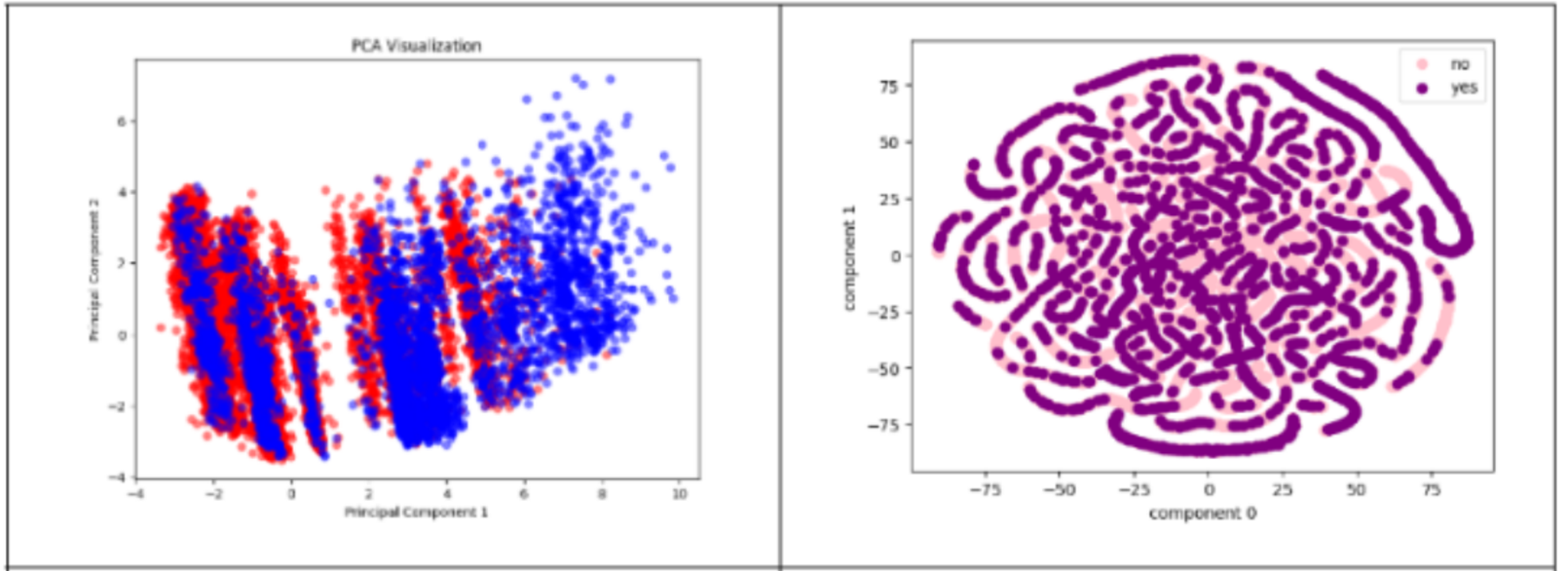
03.Data Preprocessing

```
df_test["year"] = df_test["cons.price.idx"]
df_test["year"] = df_test["cons.price.idx"].replace({93.918 : 2008, 92.963 : 2009, 93.444 : 2008, 92.893 : 2009,
                                                     94.027 : 2010, 92.431 : 2009, 94.199 : 2010, 94.465 : 2008,
                                                     92.649 : 2009, 93.994 : 20
                                                     94.601 : 2010, 92.469 : 20
                                                     93.876 : 2010, 92.713 : 20
                                                     92.379 : 2009, 94.055 : 20
                                                     94.767 : 2010, 92.756 : 20
```



create "year" column
& delete "euribor3m", "nr.employed"

03.Data Preprocessing



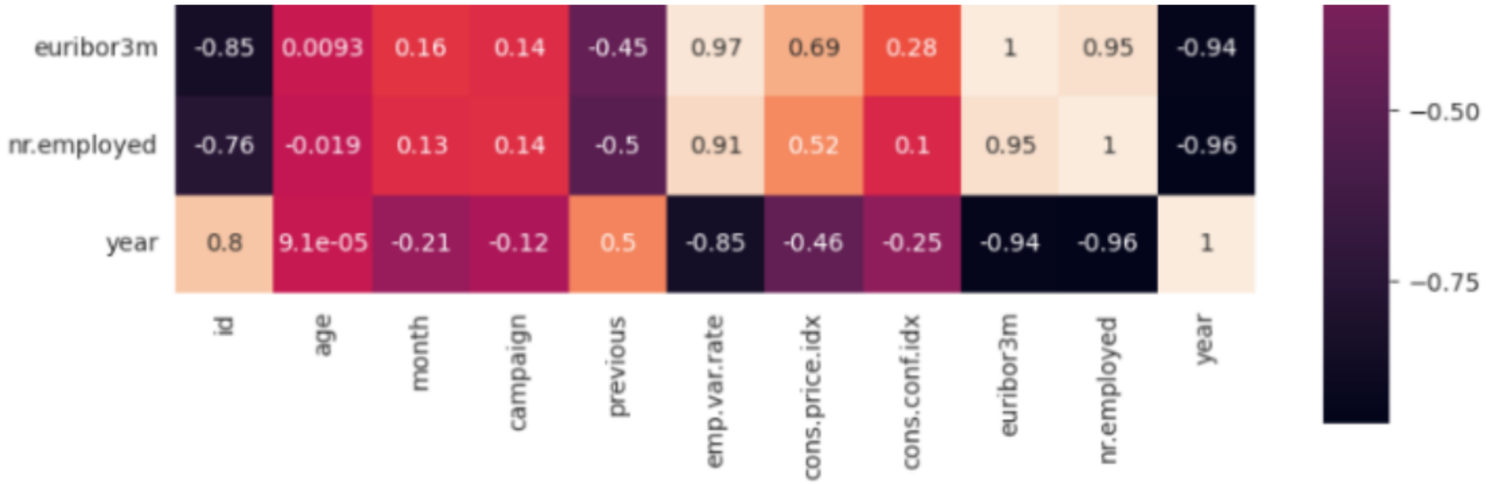
visualization result of PCA | visualization result of t-SNE

dimensionality reduction

(we tried)

03.Data Preprocessing

```
df_train = df_train.drop(labels=["euribor3m", "nr.employed"], axis=1)
```



delete "euribor3m", "nr.employed"

03.Data Preprocessing

```
df_train[columns_without_ynid].duplicated(keep=False).sum()
```

```
2504
```

```
train_duplicated = df_train.loc[df_train[columns_without_ynid].duplicated(keep=False), :]
```

```
train_duplicated['y'].value_counts()
```

```
y
no     2290
yes     214
Name: count, dtype: int64
```

preprocessing

duplicated columns

excluding "id" and "y"

(we tried)

03.Data Preprocessing

```
df_test['job'].replace('unknown', df_test['job'].mode().iloc[0], inplace=True)
```

```
df_test['loan'].replace('unknown', df_test['loan'].mode().iloc[0], inplace=True)
```

```
df_test['marital'].replace('unknown', df_test['marital'].mode().iloc[0], inplace=True)
```

```
df = df.drop('default_yes', axis = 1)
```

```
min_max_scaler=preprocessing.MinMaxScaler()
data_scaled=pd.DataFrame(min_max_scaler.fit_transform(df),columns=df.columns)
```

replacing all unknown missing values & MinMax Scaler

04.Data Sampling

Oversampling

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_cls))
```

	precision	recall	f1-score	support
0.0	0.92	0.97	0.94	5849
1.0	0.97	0.91	0.94	5966
accuracy			0.94	11815
macro avg	0.94	0.94	0.94	11815
weighted avg	0.94	0.94	0.94	11815

ADASYN

Over & Undersampling

	precision	recall	f1-score	support
0.0	0.36	0.49	0.42	743
1.0	0.93	0.89	0.91	5847
accuracy			0.84	6590
macro avg	0.65	0.69	0.66	6590
weighted avg	0.87	0.84	0.85	6590

SMOTE Tomek

	precision	recall	f1-score	support
0.0	0.95	0.97	0.96	3585
1.0	0.98	0.96	0.97	4369
accuracy			0.96	7954
macro avg	0.96	0.97	0.96	7954
weighted avg	0.97	0.96	0.96	7954

SMOTE ENN

04.Data Sampling

Undersampling

	precision	recall	f1-score	support
0.0	0.91	0.97	0.94	3162
1.0	0.81	0.57	0.67	713
accuracy			0.90	3875
macro avg	0.86	0.77	0.81	3875
weighted avg	0.89	0.90	0.89	3875

Tomek Links

	precision	recall	f1-score	support
0.0	0.91	0.98	0.94	5557
1.0	0.70	0.32	0.44	781
accuracy			0.90	6338
macro avg	0.81	0.65	0.69	6338
weighted avg	0.89	0.90	0.88	6338

One-Sided Selection

	precision	recall	f1-score	support
0.0	0.92	0.98	0.95	5589
1.0	0.71	0.33	0.45	715
accuracy			0.91	6304
macro avg	0.81	0.65	0.70	6304
weighted avg	0.90	0.91	0.89	6304

NCR

- for final model

Model Selection
+ Final Result

05.Model Selection

Utilizing Pycaret

```
model = compare_models(sort = 'F1', fold = 3, n_select = 5)
```

Initiated	22:47:45
Status	Loading Dependencies
Estimator	Compiling Library

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lda	Linear Discriminant Analysis	0.8896	0.7961	0.8896	0.8799	0.8839	0.3925	0.3960	0.2433
lightgbm	Light Gradient Boosting Machine	0.9004	0.0000	0.9004	0.8827	0.8826	0.3394	0.3745	0.5267
catboost	CatBoost Classifier	0.8992	0.0000	0.8992	0.8810	0.8823	0.3414	0.3717	4.1900
gbc	Gradient Boosting Classifier	0.9017	0.8059	0.9017	0.8845	0.8821	0.3316	0.3742	1.2633
rf	Random Forest Classifier	0.8959	0.0000	0.8959	0.8770	0.8804	0.3358	0.3595	0.4633
knn	K Neighbors Classifier	0.8920	0.0000	0.8920	0.8735	0.8785	0.3341	0.3503	0.5833
lr	Logistic Regression	0.9002	0.7961	0.9002	0.8823	0.8771	0.2947	0.3481	0.8600
ada	Ada Boost Classifier	0.8984	0.8015	0.8984	0.8787	0.8767	0.2970	0.3419	0.5533
ridge	Ridge Classifier	0.8994	0.7961	0.8994	0.8808	0.8763	0.2904	0.3426	0.2133
et	Extra Trees Classifier	0.8857	0.0000	0.8857	0.8674	0.8736	0.3144	0.3253	0.4767
nb	Naive Bayes	0.8376	0.0000	0.8376	0.8760	0.8529	0.3535	0.3664	0.2167
dt	Decision Tree Classifier	0.8419	0.0000	0.8419	0.8505	0.8460	0.2513	0.2519	0.2700
dummy	Dummy Classifier	0.8873	0.0000	0.8873	0.7874	0.8344	0.0000	0.0000	0.2400
svm	SVM - Linear Kernel	0.6908	0.6495	0.6908	0.8770	0.6858	0.1969	0.2580	0.3867
qda	Quadratic Discriminant Analysis	0.4256	0.7881	0.4256	0.8708	0.4233	0.1490	0.1890	0.2500

```
save_model(tuned_lda, './lda')
```

```
Transformation Pipeline and Model Successfully Saved

(Pipeline(memory=Memory(location=None),
          steps=[('label_encoding',
                  TransformerWrapperWithInverse(exclude=None, include=None,
                                                transformer=LabelEncoder())),
                 ('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['id', 'age', 'campaign', 'pdays',
                                              'previous', 'emp.var.rate',
                                              'cons.price.idx', 'cons.conf.idx',
                                              'euribor3m', 'nr.employed'],
                                     transformer=SimpleImpute...
                                                 return_df=True,
                                                 use_cat_names=True,
                                                 verbose=0))),
                 ('clean_column_names',
                  TransformerWrapper(exclude=None, include=None,
                                     transformer=CleanColumnNames(match='[\\]\\[\\,\\{\\}\\"\\:]+'))),
                 ('trained_model',
                  LinearDiscriminantAnalysis(covariance_estimator=None,
                                             n_components=None, priors=None,
                                             shrinkage=None, solver='svd',
                                             store_covariance=False,
                                             tol=0.0001))],
          verbose=False),
 './lda.pkl')
```

What we tried

Logistic regression

decision tree

randomforest

Naive Bayes

SGD (Stochastic Gradient Descent)

Gradient boosting classifier

LightGBM

BBC (balanced bagging classifier)

XGboost

Catboost

05.Model Selection

Results

	precision	recall	f1-score	support
0.0	0.90	0.98	0.94	5354
1.0	0.65	0.23	0.34	778
accuracy			0.89	6132
macro avg	0.77	0.60	0.64	6132
weighted avg	0.87	0.89	0.86	6132

Logistic Regression

	precision	recall	f1-score	support
0.0	0.91	0.96	0.93	5354
1.0	0.55	0.31	0.40	778
accuracy			0.88	6132
macro avg	0.73	0.64	0.67	6132
weighted avg	0.86	0.88	0.87	6132

Decision Tree

	precision	recall	f1-score	supp
0.0	0.89	0.99	0.94	5354
1.0	0.70	0.18	0.29	778
accuracy			0.89	6132
macro avg	0.80	0.59	0.61	6132
weighted avg	0.87	0.89	0.86	6132

Random Forest

	precision	recall	f1-score	support
0.0	0.92	0.87	0.89	5354
1.0	0.34	0.48	0.40	778
accuracy			0.82	6132
macro avg	0.63	0.67	0.65	6132
weighted avg	0.85	0.82	0.83	6132

Naive Bayes

	precision	recall	f1-score	support
0.0	0.87	1.00	0.93	5354
1.0	0.00	0.00	0.00	778
accuracy			0.87	6132
macro avg	0.44	0.50	0.47	6132
weighted avg	0.78	0.87	0.81	6132

Stochastic Gradient Descent

	precision	recall	f1-score	support
0.0	0.91	0.97	0.94	5354
1.0	0.60	0.30	0.40	778
accuracy			0.89	6132
macro avg	0.75	0.64	0.67	6132
weighted avg	0.87	0.89	0.87	6132

Gradient Boosting Classifier

	precision	recall	f1-score	support
0.0	0.90	0.97	0.94	5354
1.0	0.63	0.29	0.40	778
accuracy			0.89	6132
macro avg	0.77	0.63	0.67	6132
weighted avg	0.87	0.89	0.87	6132

LightGBM

	precision	recall	f1-score	support
0.0	0.91	0.97	0.94	5354
1.0	0.57	0.31	0.40	778
accuracy			0.88	6132
macro avg	0.74	0.64	0.67	6132
weighted avg	0.86	0.88	0.87	6132

XGBoost

	precision	recall	f1-score	support
0.0	0.91	0.97	0.94	5354
1.0	0.57	0.31	0.40	778
accuracy			0.88	6132
macro avg	0.74	0.64	0.67	6132
weighted avg	0.86	0.88	0.87	6132

CatBoost

	precision	recall	f1-score	support
0.0	0.93	0.87	0.90	5354
1.0	0.39	0.57	0.47	778
accuracy			0.83	6132
macro avg	0.66	0.72	0.68	6132
weighted avg	0.87	0.83	0.85	6132

Decision Tree + BBC

	precision	recall	f1-score	support
0.0	0.94	0.87	0.90	5354
1.0	0.40	0.62	0.48	778
accuracy			0.83	6132
macro avg	0.67	0.74	0.69	6132
weighted avg	0.87	0.83	0.85	6132

Random Forest + BBC

Finalized model

NCR + CatBoost
+ k-fold cross validation

	precision	recall	f1-score	support
0.0	0.91	0.97	0.94	3498
1.0	0.80	0.54	0.65	759
accuracy			0.89	4257
macro avg	0.86	0.76	0.79	4257
weighted avg	0.89	0.89	0.89	4257

06.Challenge & Solution

LOOCV

LOOCV

```
# CatBoost 모델 정의
model = CatBoostClassifier()

param_grid = {'iterations': [200, 300],'depth': [5], 'learning_rate': [0.01, 0.05, 0.1],'l2_leaf_reg': [3, 5, 7]}

from sklearn.model_selection import LeaveOneOut
loo = LeaveOneOut()
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, scoring='f1', cv=loo, verbose=1, n_jobs=-1 )

#model training
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
best_score =  grid_search.best_score_

print(best_params)
print(best_score)

best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

print(classification_report(y_test, y_pred))
```

```
Fitting 17024 folds for each of 18 candidates, totalling 306432 fits
```

6시간 39분 13초 오전 5:50에 완료됨

Runtime Problem of Colab

4시간 54분 19초 오전 8:07에 완료됨

5시간 11분 32초 오전 8:07에 완료됨

6시간 39분 13초 오전 5:50에 완료됨

```
function ClickConnect(){
console.log("Working");
document.querySelector("colab-toolbar-button#connect").click()

}setInterval(ClickConnect, 1800000)

// the code we used
```

07. Conclusion

F1 score : 0.648

Accuracy : 0.894

ROC AUC : 0.857

in train set

F1 score : 0.5007

in test set

Pattern Recognition Final Presentation

Thank You

Team 03