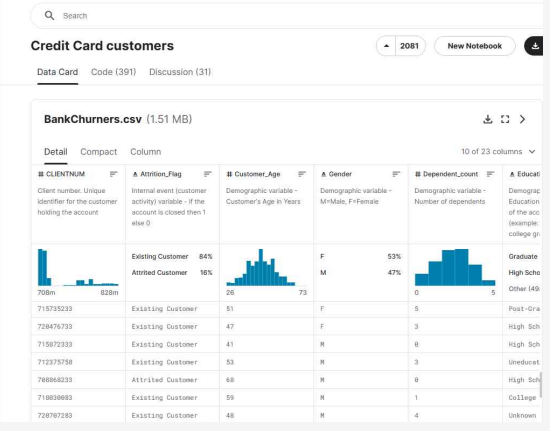
Bank Churn Data Exploration And Churn Prediction

은행 이탈자 데이터 탐색과 예측

신용카드 고객 데이터

Credit Card customers (kaggle.com)

Kaggle 내 신용카드고객 데이터를 사용



개요

1. 라이브러리 및 데이터 불러오기
2. Exploratory Data Analysis(EDA ; 탐색적 데이터 분석)
3. Data Preprocessing(데이터 전처리)
   - Data Upsampling Using SMOTE
   - Principal Component Analysis
4. Model Selection And Evaluation
   - Cross Validation(교차검증)
   - Model Evaluation(모델평가)
   - Model Evaluation On Original Data (Before Upsampling)
5. Results

라이브러리 및 데이터 불러오기

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as ex
import plotly.graph_objs as go
import plotly.figure_factory as ff
from plotly.subplots import make_subplots
import plotly.offline as pyo
pyo.init_notebook_mode()
sns.set_style('darkgrid')
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score as f1
from sklearn.metrics import confusion_matrix
import scikitplot as skplt
plt.rc('figure',figsize=(18,9))
%pip install imbalanced-learn
from imblearn.over_sampling import SMOTE
```

라이브러리 및 데이터 불러오기

```
c_data = pd.read_csv('BankChurners.csv')
c_data = c_data[c_data.columns[:-2]]
c_data.head(3)
```

	CLIENTNUM	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book
0	768805383	Existing Customer	45	M	3	High School	Married	$60K - 80K	Blue	39
1	818770008	Existing Customer	49	F	5	Graduate	Single	Less than $40K	Blue	44
2	713982108	Existing Customer	51	M	3	Graduate	Married	$80K - 120K	Blue	36

Exploratory Data Analysis(EDA ; 탐색적 데이터 분석)

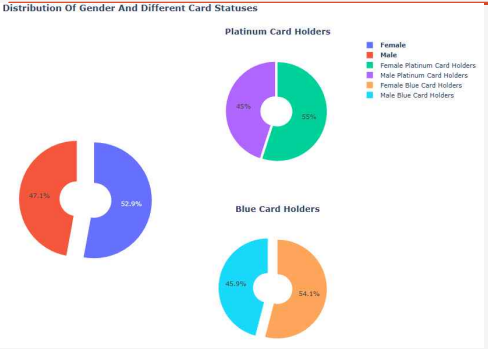
고객 나이의 분포

데이터 내에서 고객 연령의 분포는 정규분포에 근사하다.

연령의 분포는 정규성을 가정할 수 있다.


```
fig = make_subplots(rows=2, cols=1)

tr1=go.Box(x=c_data['Customer_Age'],name='Age Box Plot',boxmean=True)
tr2=go.Histogram(x=c_data['Customer_Age'],name='Age Histogram')

fig.add_trace(tr1,row=1,col=1)
fig.add_trace(tr2,row=2,col=1)

fig.update_layout(height=700, width=1200, title_text="Distribution of Customer Ages")
fig.show()
```

Exploratory Data Analysis(EDA ; 탐색적 데이터 분석)




```
fig = make_subplots(
    rows=2, cols=2,subplot_titles=('','<b>Platinum Card Holders','<b>Blue Card Holders<b>','Residuals'),
    vertical_spacing=0.05,
    specs=[[{'type': 'pie','rowspan': 2}        ,{'type': 'pie'}] ,
           [None                                ,{'type': 'pie'}]            ,
          ]
)

fig.add_trace(
    go.Pie(values=c_data.Gender.value_counts().values,labels=['<b>Female<b>','<b>Male<b>'],hole=0.3,pull=[0,0.3]),
    row=1, col=1
)

fig.add_trace(
    go.Pie(
        labels=['Female Platinum Card Holders','Male Platinum Card Holders'],
        values=c_data.query('Card_Category=="Platinum"').Gender.value_counts().values,
        pull=[0,0.05,0.5],
        hole=0.3

    ),
    row=1, col=2
)

fig.add_trace(
    go.Pie(
        labels=['Female Blue Card Holders','Male Blue Card Holders'],
        values=c_data.query('Card_Category=="Blue"').Gender.value_counts().values,
        pull=[0,0.2,0.5],
        hole=0.3
    ),
    row=2, col=2
)


fig.update_layout(
    height=600,
    showlegend=True,
    title_text="<b>Distribution Of Gender And Different Card Statuses<b>",
)

fig.show()
```

고객의 성별의 분포

데이터 내에서 여성 표본이 남성보다 더 많다.

하지만 비율이 그렇게 크지 않으므로 성별이 균일하게 분포되어 있다고 볼 수 있다.

Exploratory Data Analysis(EDA ; 탐색적 데이터 분석)

부양가족 수의 분포

부양 가족의 수의 분포는 약간 오른쪽으로 치우쳐진 정규 분포를 따른다.

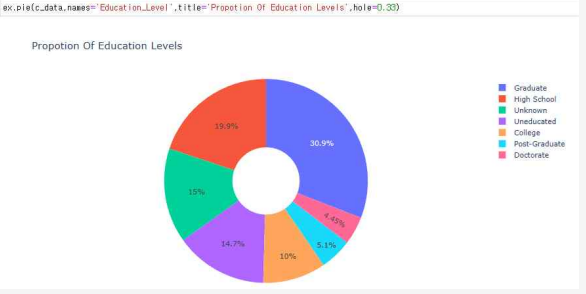

```
fig = make_subplots(rows=2, cols=1)

tr1=go.Box(x=cc_data['Dependent_count'],name='Dependent count Box Plot',boxmean=True)
tr2=go.Histogram(x=cc_data['Dependent_count'],name='Dependent count Histogram')

fig.add_trace(tr1,row=1,col=1)
fig.add_trace(tr2,row=2,col=1)

fig.update_layout(height=700, width=1200, title_text="Distribution of Dependent counts (close family size)")
fig.show()
```

Exploratory Data Analysis(EDA ; 탐색적 데이터 분석)

교육 수준

70%이상의 고객이 정식 교육 수준을 가지고 있다.

```
px.pie(c_data,names='Education_Level',title='Propotion Of Education Levels',hole=0.33)
```

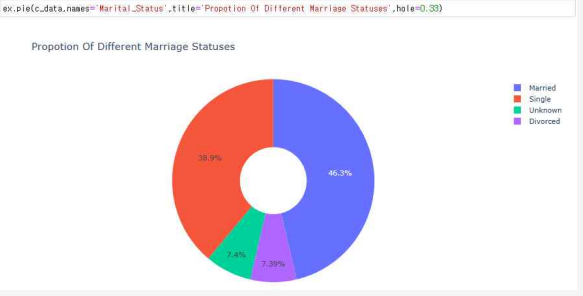


Exploratory Data Analysis(EDA ; 탐색적 데이터 분석)

결혼 상태

고객의 절반정도가 기혼자이고 나머지 절반은 미혼자이다.

```
ex.pie(c_data,names='Marital_Status',title='Propotion Of Different Marriage Statuses',hole=0.33)
```



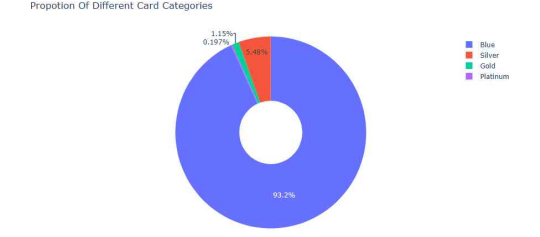
Exploratory Data Analysis(EDA ; 탐색적 데이터 분석)

소득의 수준

```
ex.pie(c_data,names='Income_Category',title='Propotion Of Different Incoae Levels',hole=0.33)
```

카드 카테고리

```
ex.pie(c_data,names='Card_Category',title='Propotion Of Different Card Categories',hole=0.33)
```



Exploratory Data Analysis(EDA ; 탐색적 데이터 분석)

은행에 속해 있던 기간

위 분포는 평평하고 낮은 첨도를 가지고 있으므로 정규성을 가정할 수 없다


```
Kurtosis of Months on book features is : 0.4001001201999707
```

```
fig = make_subplots(rows=2, cols=1)

tr1=go.Box(x=c_data['Months_on_book'],name='Months on book Box Plot',boxmean=True)
tr2=go.Histogram(x=c_data['Months_on_book'],name='Months on book Histogram')

fig.add_trace(tr1,row=1,col=1)
fig.add_trace(tr2,row=2,col=1)

fig.update_layout(height=700, width=1200, title_text="Distribution of months the customer is part of the bank")
fig.show()
print('Kurtosis of Months on book features is : {}'.format(c_data['Months_on_book'].kurt()))
```

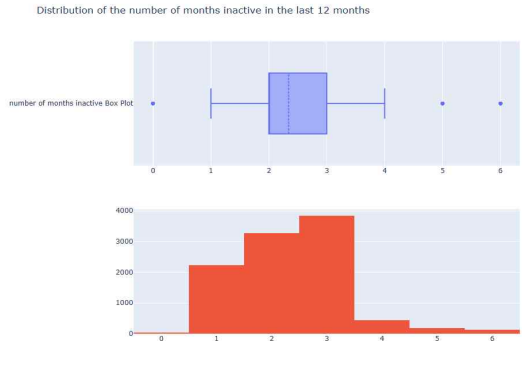
Exploratory Data Analysis(EDA ; 탐색적 데이터 분석)

고객이 가입한 은행의 상품 수

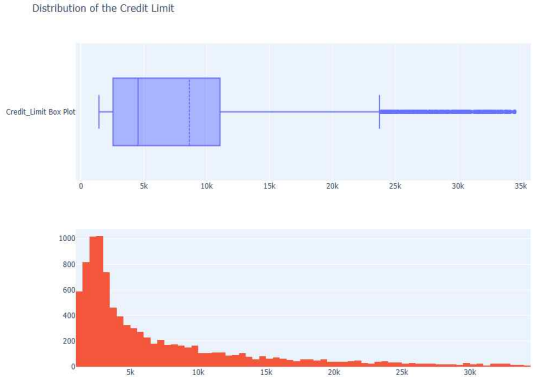
균일분포에 가깝기 때문에 이탈자 예측 변수로써 유용하지 않다

Exploratory Data Analysis(EDA ; 탐색적 데이터 분석)

카드를 사용하지 않은 개월의 수



카드한도

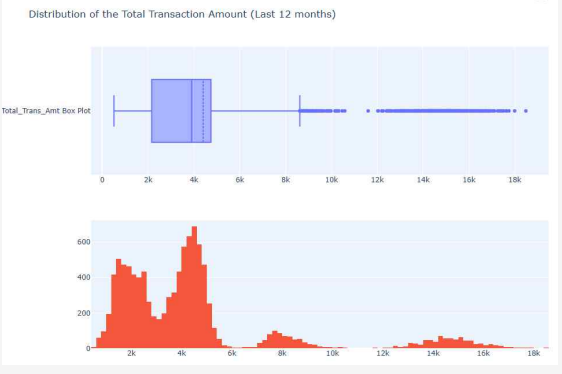


Exploratory Data Analysis(EDA ; 탐색적 데이터 분석)

12개월간의 거래량의 분포

위 분포는 다봉분포를 보여준다.

데이터 내 거래량의 크기에 따라 군집화할 수 있다.



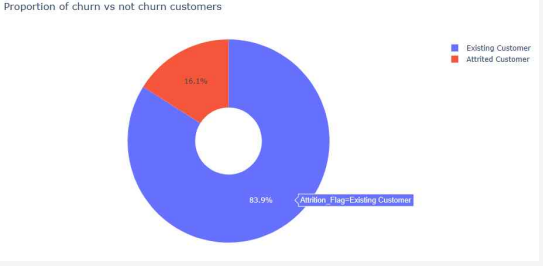
Exploratory Data Analysis(EDA ; 탐색적 데이터 분석)

이탈자 고객의 비율

데이터 내 16%만이 이탈자 고객이다.

다음단계에서는 SMOTE를 사용하여 이탈 샘플을 일반 고객 샘플 크기와 일치시킨다.

크기 차이로 놓칠 수 있는 작은 세부 사항을 파악할 수 있다.



Data Preprocessing(데이터 전처리)

```
c_data.Attrition_Flag = c_data.Attrition_Flag.replace({'Attrited Customer':1,'Existing Customer':0})
c_data.Gender = c_data.Gender.replace({'F':1,'M':0})
c_data = pd.concat([c_data,pd.get_dummies(c_data['Education_Level']).drop(columns=['Unknown'])],axis=1)
c_data = pd.concat([c_data,pd.get_dummies(c_data['Income_Category']).drop(columns=['Unknown'])],axis=1)
c_data = pd.concat([c_data,pd.get_dummies(c_data['Marital_Status']).drop(columns=['Unknown'])],axis=1)
c_data = pd.concat([c_data,pd.get_dummies(c_data['Card_Category']).drop(columns=['Platinum'])],axis=1)
c_data.drop(columns = ['Education_Level','Income_Category','Marital_Status','Card_Category','CLIENTNUM'],inplace=True)
```

LINE 1 : 기존 고객은 1, 이탈자 고객은 0으로 변수를 변환한다.

LINE 2 : 남자 고객은 1, 여자 고객은 0으로 변수를 변환한다.

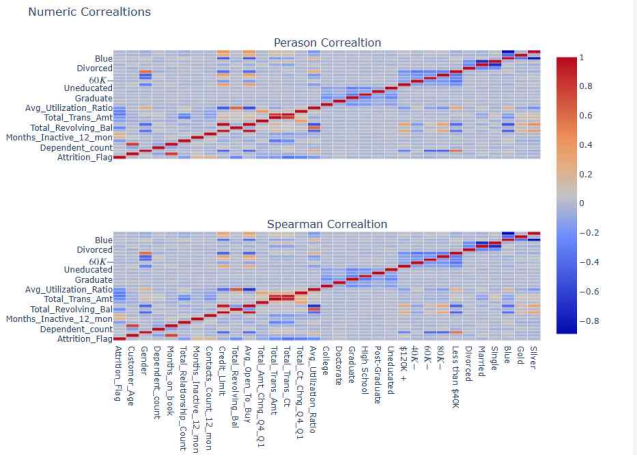
LINE 3~6 : 'Education_Level', 'Income_Category', Marital_Status', 'Card_Category' 열들에 대해 더미 변수를 생성하고, 'Unknown' 값이 있는 열들은 제거한다.

LINE 7 : 필요 없는 열들을 삭제한다.

Data Preprocessing(데이터 전처리)

lonths_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit	Total_Revolving_Bal	...	40K – 60K	60K – 80K	80K – 120K	Less than $40K	Divorced	Married	Single	Blue	Gold	Silver
1	3	12691.0	777	...	0	1	0	0	0	1	0	1	0	0
1	2	8256.0	864	...	0	0	0	1	0	0	1	1	0	0
1	0	3418.0	0	...	0	0	1	0	0	1	0	1	0	0
4	1	3313.0	2517	...	0	0	0	1	0	0	0	1	0	0
1	0	4716.0	0	...	0	1	0	0	0	1	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2	3	4003.0	1851	...	1	0	0	0	0	0	1	1	0	0
2	3	4277.0	2186	...	1	0	0	0	1	0	0	1	0	0
3	4	5409.0	0	...	0	0	0	1	0	1	0	1	0	0
3	3	5281.0	0	...	1	0	0	0	0	0	0	1	0	0
2	4	10388.0	1961	...	0	0	0	1	0	1	0	0	0	1

Data Preprocessing(데이터 전처리)



Data Preprocessing - Data Upsampling Using SMOTE

```
oversample = SMOTE()
X, y = oversample.fit_resample(c_data[c_data.columns[1:]], c_data[c_data.columns[0]])
usampled_df = X.assign(Churn = y)
```

```
ohe_data =usampled_df[usampled_df.columns[15:-1]].copy()

usampled_df = usampled_df.drop(columns=usampled_df.columns[15:-1])
```

1. SMOTE를 이용한 오버샘플링

SMOTE()를 사용하여 소수 클래스(이탈한 고객)를 오버샘플링한다.

fit_resample 함수를 사용하여 샘플링을 수행하고, 오버샘플링된 특성 변수 X와 샘플링된 목표 변수 y를 반환한다.

새로운 데이터 usampled_df를 생성하고 X의 특성을 할당하면서 목표변수 y를 Churn열로 추가한다.

2. ohe_data를 이용해 usampled_df 내 더미 변수를 제거하였다.

Data Preprocessing - Data Upsampling Using SMOTE

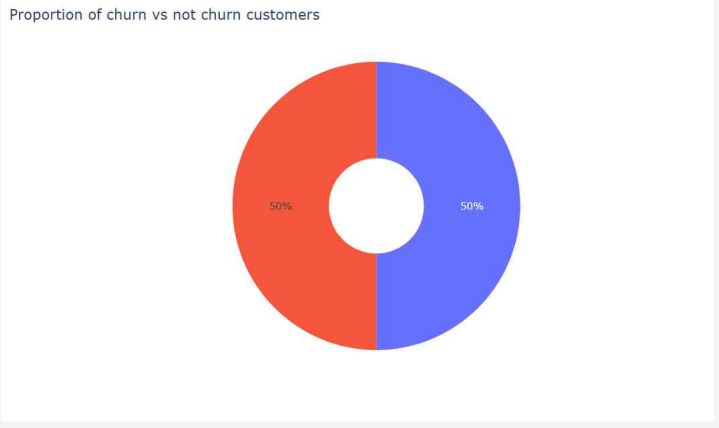
	Customer_Age	Gender	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit
0	45	0	3	39	5	1	3	12691.000000
1	49	1	5	44	6	1	2	8256.000000
2	51	0	3	36	4	1	0	3418.000000
3	40	1	4	34	3	4	1	3313.000000
4	40	0	3	21	5	1	0	4716.000000
...	...	...	...	...	...	...	...	...
16995	38	0	2	30	2	2	3	4879.841489
16996	45	1	3	34	2	3	1	5598.840274
16997	52	0	1	40	1	2	2	2317.213783
16998	46	0	2	33	3	3	2	1469.837539
16999	46	1	1	31	3	3	2	2186.411027

17000 rows × 16 columns

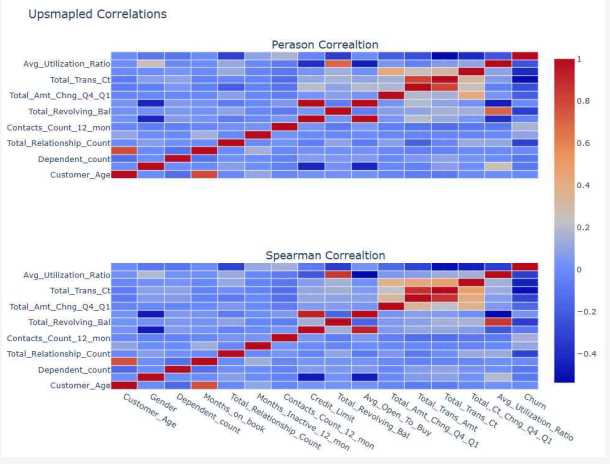
	College	Doctorate	Graduate	High School	Post-Graduate	Uneducated	$120K +	$40K — 60K	$60K — 80K	$80K — 120K	Less than $40K	Divorced	Married	Single	Blue	Gold	Silver
0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1	0	0
1	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1	0	0
2	0	0	1	0	0	0	0	0	0	1	0	0	1	0	1	0	0
3	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
4	0	0	0	0	0	1	0	0	1	0	0	0	1	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16995	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
16996	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
16997	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
16998	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
16999	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0

17000 rows × 17 columns

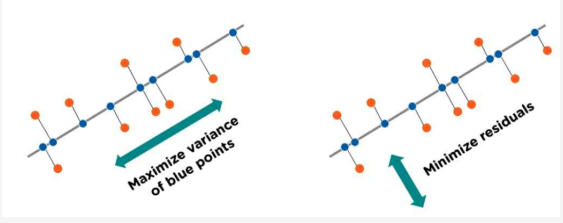
Data Preprocessing - Data Upsampling Using SMOTE



Data Preprocessing - Data Upsampling Using SMOTE



Principal Component Analysis (PCA ; 주성분분석)



주성분 분석은 고차원의 데이터를 저차원의 데이터로 환원시키는 기법을 말한다.

주성분 분석을 사용하여 범주형 자료의 차원을 줄여 변수를 일부 손실한다.

하지만 표본의 차이를 가장 잘 나타내는 성분들로 분해함으로써 데이터 분석에 여러 가지 이점을 제공한다.

Principal Component Analysis (PCA ; 주성분분석)

LINE 1~2 : 4개의 주성분을 사용한다.

PCA 클래스를 사용하여 주성분 분석을 수행할 모델을 설정한다.

LINE 3 : fit_transform를 사용하여 PCA를 적용하고, 각 샘플에 대한 주성분들의 행렬을 얻습니다.

LINE 4~5 : 각 주성분이 설명하는 분산의 비율을 얻습니다.

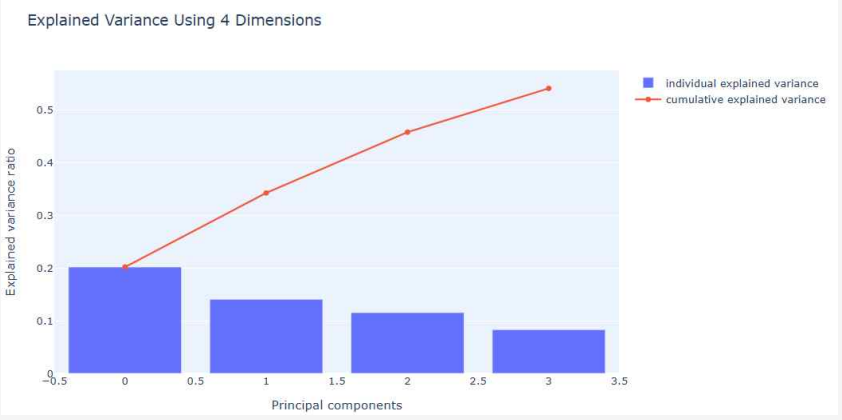
LINE 6 : cumsum 함수를 사용하여 누적 분산 비율을 계산합니다.

usampled_df에 주성분 데이터를 추가한다.

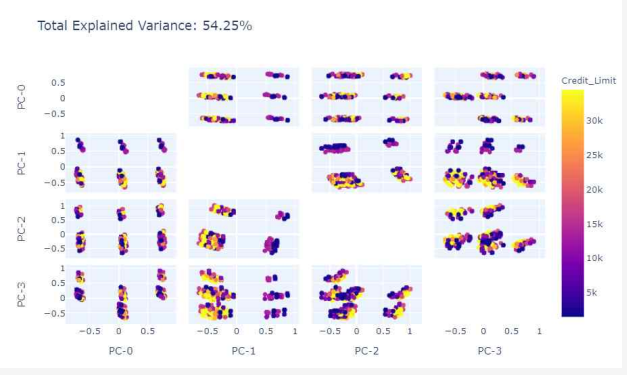
```
N_COMPONENTS = 4

pca_model = PCA(n_components = N_COMPONENTS )

pc_matrix = pca_model.fit_transform(ohe_data)

evr = pca_model.explained_variance_ratio_
total_var = evr.sum() * 100
cumsum_evr = np.cumsum(evr)

trace1 = {
    "name": "individual explained variance",
    "type": "bar",
    'y':evr}
trace2 = {
    "name": "cumulative explained variance",
    "type": "scatter",
    'y':cumsum_evr}
data = [trace1, trace2]
layout = {
    "xaxis": {"title": "Principal components"},
    "yaxis": {"title": "Explained variance ratio"},
  }
fig = go.Figure(data=data, layout=layout)
fig.update_layout(     title='Explained Variance Using {} Dimensions'.format(N_COMPONENTS))
fig.show()
```

```
usampled_df_with_pcs = pd.concat([usampled_df,pd.DataFrame(pc_matrix,columns=['PC-{}'.format(i) for
i in range(0,N_COMPONENTS)])],axis=1)
```

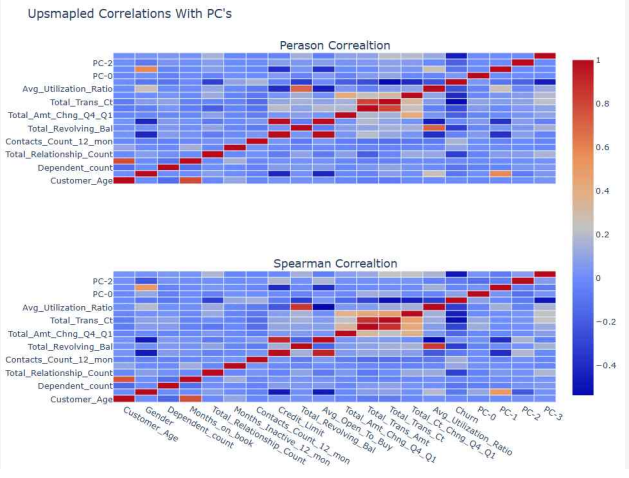
Principal Component Analysis (PCA ; 주성분분석)



Principal Component Analysis (PCA ; 주성분분석)



Principal Component Analysis (PCA ; 주성분분석)



Model Selection And Evaluation

```
X_features = ['Total_Trans_Ct','PC-3','PC-1','PC-0','PC-2','Total_Ct_Chng_Q4_Q1','Total_Relationship_Coun

X = usampled_df_with_pcs[X_features]
y = usampled_df_with_pcs['Churn']
```

```
train_x,test_x,train_y,test_y = train_test_split(X,y,random_state=42)
```

분석에 사용할 특성들을 리스트로 정의합니다.

선택한 특성들을 사용하여 X를 생성한다.

목표 변수를 y에 할당한다. 여기서는 이탈자 데이터의 열이 목표 변수로 사용했다.

train_test_split 함수를 사용하여 데이터를 훈련 데이터와 테스트 데이터로 나눈다.

Cross Validation(교차검증)

```
rf_pipe = Pipeline(steps =[ ('scale',StandardScaler()), ("RF",RandomForestClassifier(random_state=42)) ])
ada_pipe = Pipeline(steps =[ ('scale',StandardScaler()), ("RF",AdaBoostClassifier(random_state=42,learning_rate=0.7)) ])
svm_pipe = Pipeline(steps =[ ('scale',StandardScaler()), ("RF",SVC(random_state=42,kernel='rbf')) ])


f1_cross_val_scores = cross_val_score(rf_pipe,train_x,train_y,cv=5,scoring='f1')
ada_f1_cross_val_scores=cross_val_score(ada_pipe,train_x,train_y,cv=5,scoring='f1')
svm_f1_cross_val_scores=cross_val_score(svm_pipe,train_x,train_y,cv=5,scoring='f1')
```

1. Pipeline 클래스를 사용하여 세 가지 다른 분류 모델을 포함한 파이프라인을 정의한다.

; Random Forest 모델(RandomForestClassifier), AdaBoost 모델(AdaBoostClassifier), SVM 모델(SVC)

2. cross_val_score 함수를 사용하여 각 모델에 대한 교차 검증(F1 점수)을 수행한다.

cv=5는 5-폴드 교차 검증을 의미하며, 모델을 훈련하고 평가하는데 5개의 폴드가 사용된다.

scoring='f1'은 F1 점수를 평가 지표로 사용함을 나타낸다.

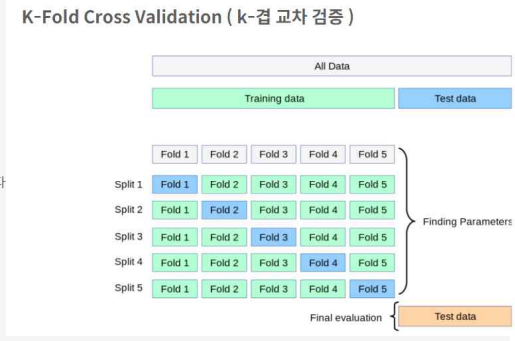
Cross Validation(교차검증)

•K-Fold는 가장 일반적으로 사용되는 교차 검증 방법이다.

•보통 회귀 모델에 사용되며, 데이터가 독립적이고 동일한 분포를 가진 경우에 사용된다.

•자세한 K-Fold 교차 검증 과정은 다음과 같다.

- 전체 데이터셋을 Training Set과 Test Set으로 나눈다.
- Training Set를 Traing Set + Validation Set으로 사용하기 위해 k개의 폴드로 나눈다
- 첫 번째 폴드를 Validation Set으로 사용하고 나머지 폴드들을 Training Set으로 사용한다.
- 모델을 Training한 뒤, 첫 번 째 Validation Set으로 평가한다.
- 차례대로 다음 폴드를 Validation Set으로 사용하며 3번을 반복한다.
- 총 k 개의 성능 결과가 나오며, 이 k개의 평균을 해당 학습 모델의 성능이라고 한다.



Cross Validation(교차검증)

$$Precision = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Positives}$$

$$Recall = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Negatives}$$

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

Precision(정밀도) : 모델이 양성으로 예측한 샘플 중에서 실제로 양성인 샘플의 비율

Recall(재현율) : 실제로 양성인 샘플 중에서 모델이 양성으로 정확히 예측한 샘플의 비율

F1 score : Precision과 Recall의 조화평균

Cross Validation(교차검증)

5개의 폴드로 나누어 교차검증을 진행했다.

Model Evaluation

훈련 데이터인 **train_x**와 해당하는 레벨인 **train_y**를 사용하여 모델을 훈련시킨다.

훈련된 모델을 이용해 테스트 데이터에 대한 예측을 수행한다.

각 모델의 예측 값을 F1 스코어로 나타내었다.

Random Forest 모델이 가장 높다.

```
rf_pipe.fit(train_x,train_y)
rf_prediction = rf_pipe.predict(test_x)

ada_pipe.fit(train_x,train_y)
ada_prediction = ada_pipe.predict(test_x)

svm_pipe.fit(train_x,train_y)
svm_prediction = svm_pipe.predict(test_x)
```

Model Results On Test Data

Model	F1 Score On Test Data
Random Forest	0.91
AdaBoost	0.89
SVM	0.89

Model Evaluation On Original Data (Before Upsampling)

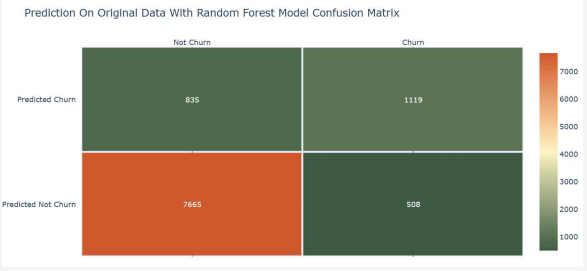
오버샘플링하기 전 원래의 데이터를 분류모델에 훈련 시켜보았다.

Model Result On Original Data (Without Upsampling)

Model	F1 Score On Original Data (Before Upsampling)
Random Forest	0.62
AdaBoost	0.54
SVM	0.55

```
ohe_data =c_data[c_data.columns[16:]].copy()
pc_matrix = pca_model.fit_transform(ohe_data)
original_df_with_pcs = pd.concat([c_data,pd.DataFrame(pc_matrix,columns=['PC-{}'.format(i) for i in range(0,N_COMPONENTS)])],axis=1)

unsampled_data_prediction_RF = rf_pipe.predict(original_df_with_pcs[X_features])
unsampled_data_prediction_ADA = ada_pipe.predict(original_df_with_pcs[X_features])
unsampled_data_prediction_SVM = svm_pipe.predict(original_df_with_pcs[X_features])
```

Results



Random Forest Model을 사용하여 실제 고객의 이탈 여부와 예측된 고객의 이탈자 여부를 확인했다.

실제로는 이탈하지 않았으나 모델이 이탈했다고 예측한 고객이 **508**명 있다.

이는 잠정적으로 이탈할 가능성이 있는 고객임을 알 수 있다.

Results

predict_proba를 사용하여 Random Forest 모델이 예측한 클래스별 확률을 얻는다.

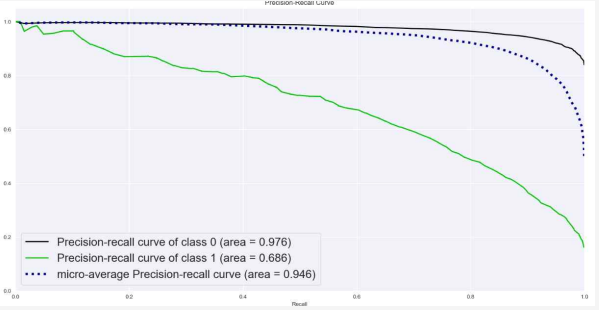
plot_precision_recall 함수를 사용하여 Precision-Recall 곡선을 시각화한다.

RC 곡선의 면적에 대한 해석

1.0: 완벽한 성능 (Precision과 Recall이 항상 1)

0.5: 랜덤한 모델과 비슷한 성능

0.0: 최악의 성능 (Precision이나 Recall이 항상 0)




```
unsampled_data_prediction_RF = rf_pipe.predict_proba(original_df_with_pcs[X_features])
skplt.metrics.plot_precision_recall(original_df_with_pcs['Attrition_Flag'], unsampled_data_prediction_RF)
plt.legend(prop={'size': 20})
```