```
!pip uninstall -y pandas pycaret google-colab
```

```
Found existing installation: pandas 2.2.2
Uninstalling pandas-2.2.2:
  Successfully uninstalled pandas-2.2.2
WARNING: Skipping pycaret as it is not installed.
Found existing installation: google-colab 1.0.0
Uninstalling google-colab-1.0.0:
  Successfully uninstalled google-colab-1.0.0
```

```
!pip install --ignore-installed blinker
```

```
Collecting blinker
  Downloading blinker-1.8.2-py3-none-any.whl.metadata (1.6 kB)
Downloading blinker-1.8.2-py3-none-any.whl (9.5 kB)
Installing collected packages: blinker
Successfully installed blinker-1.8.2
```

```
!pip install pandas numpy pycaret
```

```
# pip install imbalanced-learn
```

```
# Importing necessary libraries
import pandas as pd
import numpy as np
from pycaret.classification import *
from imblearn.under_sampling import RandomUnderSampler

# Display all columns and rows
pd.set_option('display.max_columns', None, 'display.max_rows', None)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
ls -al drive/MyDrive/home-credit-default-risk
```

```
total 2621352
-rw------- 1 root root  26567651 Dec 10  2019 application_test.csv
-rw------- 1 root root 166133370 Dec 10  2019 application_train.csv
-rw------- 1 root root 375592889 Dec 10  2019 bureau_balance.csv
-rw------- 1 root root 170016717 Dec 10  2019 bureau.csv
-rw------- 1 root root 424582605 Dec 11  2019 credit_card_balance.csv
-rw------- 1 root root     37383 Dec 10  2019 HomeCredit_columns_description.csv
-rw------- 1 root root 723118349 Dec 11  2019 installments_payments.csv
-rw------- 1 root root 392703158 Dec 10  2019 POS_CASH_balance.csv
-rw------- 1 root root 404973293 Dec 11  2019 previous_application.csv
-rw------- 1 root root    536202 Dec 11  2019 sample_submission.csv
```

```
# Load application data {train|test}
train_data = pd.read_csv('drive/MyDrive/home-credit-default-risk/application_train.csv')
test_data = pd.read_csv('drive/MyDrive/home-credit-default-risk/application_test.csv')

# TARGET variable distribution
train_data['TARGET'].value_counts(normalize=True)
```

	proportion
TARGET	
0	0.919271
1	0.080729

dtype: float64

```
null_percentage =  train_data.isnull().mean()*100

print(f"Train Data original shape: {train_data.shape}")
```

```
Train Data original shape: (307511, 122)
```

Resources ✕ •••

You are not subscribed. Learn more

You currently have zero compute units available. Resources offered free of charge are not guaranteed. Purchase more units here.

At your current usage level, this runtime may last up to 1 hour 40 minutes.

Manage sessions

Python 3 Google Compute Engine backend

Showing resources from 5:05 PM to 5:52 PM

System RAM
6.4 / 12.7 GB

Disk
36.9 / 107.7 GB

```
null_percentage =  test_data.isnull().mean()*100

print(f"Test Data original shape: {test_data.shape}")
```

```
Test Data original shape: (48744, 121)
```

```
test_null = null_percentage >= 35
```

```
#columns count with  >= 35% null values
print(f"Total columns with >= 35% null values: {test_null[test_null].count()}")
```

```
Total columns with >= 35% null values: 49
```

```
# Load bureau data
bureau = pd.read_csv('drive/MyDrive/home-credit-default-risk/bureau.csv')

# Load bureau balance data
bureau_balance = pd.read_csv('drive/MyDrive/home-credit-default-risk/bureau_balance.csv')

# Load POS_CASH_balance data
pos_cash_balance = pd.read_csv('drive/MyDrive/home-credit-default-risk/POS_CASH_balance.cs

# Load credit card balance data
credit_card_balance = pd.read_csv('drive/MyDrive/home-credit-default-risk/credit_card_bala

# Load installments payments data
installments_payments = pd.read_csv('drive/MyDrive/home-credit-default-risk/installments_p
```

```
# Sum and mean of credit amount
bureau_agg = bureau.groupby('SK_ID_CURR').agg({
    'AMT_CREDIT_SUM': ['sum', 'median'],
    'CREDIT_DAY_OVERDUE': ['max'],
    'DAYS_CREDIT': ['mean'],
}).reset_index()

# Rename the columns
bureau_agg.columns = ['SK_ID_CURR', 'CREDIT_SUM_TOTAL', 'CREDIT_SUM_MEAN', 'CREDIT_OVERDU
bureau_agg.head()
```

	SK_ID_CURR	CREDIT_SUM_TOTAL	CREDIT_SUM_MEAN	CREDIT_OVERDUE_MAX	CREDIT_DURATION_
0	100001	1453365.000	168345.00	0	-735.0
1	100002	865055.565	54130.50	0	-874.0
2	100003	1017400.500	92576.25	0	-1400.7
3	100004	189037.800	94518.90	0	-867.0
4	100005	657126.000	58500.00	0	-190.6

```
# Merge with train data
train_data = train_data.merge(bureau_agg, on='SK_ID_CURR', how='left')
test_data = test_data.merge(bureau_agg, on='SK_ID_CURR', how='left')
```

```
# Replace categorical values with numeric ones
bureau_balance['STATUS'] = bureau_balance['STATUS'].replace(['C', '0'], 0).replace(['1', 

# Convert the STATUS column to numeric to handle any unexpected non-numeric values
bureau_balance['STATUS'] = pd.to_numeric(bureau_balance['STATUS'], errors='coerce')

# Group by SK_ID_BUREAU and aggregate status as sum (missed payments) and count (total mo
bureau_balance_agg = bureau_balance.groupby('SK_ID_BUREAU').agg({
    'STATUS': ['sum', 'count']
}).reset_index()

# Rename the columns
bureau_balance_agg.columns = ['SK_ID_BUREAU', 'MISSED_PAYMENTS', 'TOTAL_MONTHS']

bureau_balance_agg.head()
```

	SK_ID_BUREAU	MISSED_PAYMENTS	TOTAL_MONTHS
0	5001709	0.0	86
1	5001710	0.0	53
2	5001711	0.0	3
3	5001712	0.0	19
4	5001713	0.0	0

```
# Count of active loans
pos_cash_agg = pos_cash_balance.groupby('SK_ID_CURR').agg({
    'MONTHS_BALANCE': 'count',  # Count of months with POS loans
    'SK_DPD': ['mean', 'sum'],  # Delay in payment (mean and total)
}).reset_index()

# Rename columns
pos_cash_agg.columns = ['SK_ID_CURR', 'POS_LOANS_COUNT', 'POS_DPD_MEAN', 'POS_DPD_TOTAL']

# Merge with train and test data
train_data = train_data.merge(pos_cash_agg, on='SK_ID_CURR', how='left')
test_data = test_data.merge(pos_cash_agg, on='SK_ID_CURR', how='left')


# Average balance over time
credit_card_agg = credit_card_balance.groupby('SK_ID_CURR').agg({
    'AMT_BALANCE': ['mean', 'max'],
    'MONTHS_BALANCE': 'count'
}).reset_index()

# Rename columns
credit_card_agg.columns = ['SK_ID_CURR', 'CREDIT_BALANCE_MEAN', 'CREDIT_BALANCE_MAX', 'CR

# Merge with train and test data
train_data = train_data.merge(credit_card_agg, on='SK_ID_CURR', how='left')
test_data = test_data.merge(credit_card_agg, on='SK_ID_CURR', how='left')


# Total and mean of payments
installments_agg = installments_payments.groupby('SK_ID_CURR').agg({
    'AMT_PAYMENT': ['sum', 'mean'],
    'DAYS_INSTALMENT': 'count'
}).reset_index()

# Rename columns
installments_agg.columns = ['SK_ID_CURR', 'PAYMENT_TOTAL', 'PAYMENT_MEAN', 'TOTAL_INSTALL

# Merge with train and test data
train_data = train_data.merge(installments_agg, on='SK_ID_CURR', how='left')
test_data = test_data.merge(installments_agg, on='SK_ID_CURR', how='left')
```

```
train_null_percentage =  train_data.isnull().mean()*100

train_null = train_null_percentage >= 35

#columns count with  >= 35% null values
print(f"Total columns with >= 35% null values: {train_null[train_null].count()}")
```

```
Total columns with >= 35% null values: 52
```

```
train_data.TARGET.value_counts()
```

	count
TARGET	
0	282686
1	24825

dtype: int64

```
X = train_data.drop(columns=['TARGET'])
y = train_data['TARGET']
```

```
# Initializing the RandomUnderSampler
rus = RandomUnderSampler(sampling_strategy='auto', random_state=42)

# Downsample the majority class
X_res, y_res = rus.fit_resample(X, y)

# Combine the resampled into a single DataFrame
train_data_resampled = pd.concat([X_res, y_res], axis=1)
```

```
train_data_resampled.TARGET.value_counts()
```

	count
TARGET	
0	24825
1	24825

dtype: int64

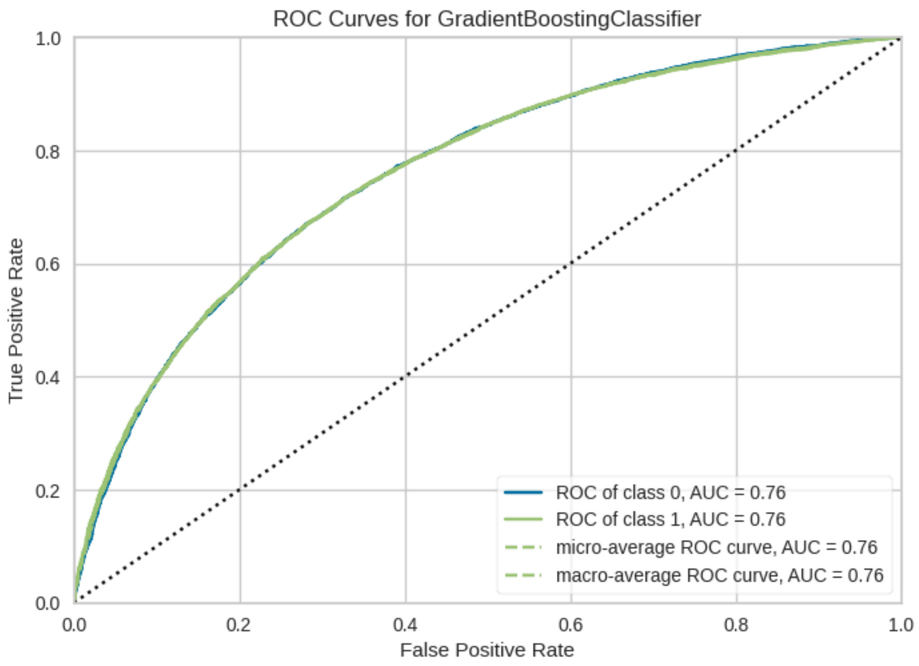
```
train = setup(data=train_data_resampled,target="TARGET")
```

	Description	Value
0	Session id	3230
1	Target	TARGET
2	Target type	Binary
3	Original data shape	(49650, 135)
4	Transformed data shape	(49650, 197)
5	Transformed train set shape	(34755, 197)
6	Transformed test set shape	(14895, 197)
7	Numeric features	118
8	Categorical features	16
9	Rows with missing values	99.2%
10	Preprocess	True
11	Imputation type	simple
12	Numeric imputation	mean
13	Categorical imputation	mode
14	Maximum one-hot encoding	25
15	Encoding method	None
16	Fold Generator	StratifiedKFold
17	Fold Number	10
18	CPU Jobs	-1
19	Use GPU	False
20	Log Experiment	False
21	Experiment Name	clf-default-name
22	USI	49d8

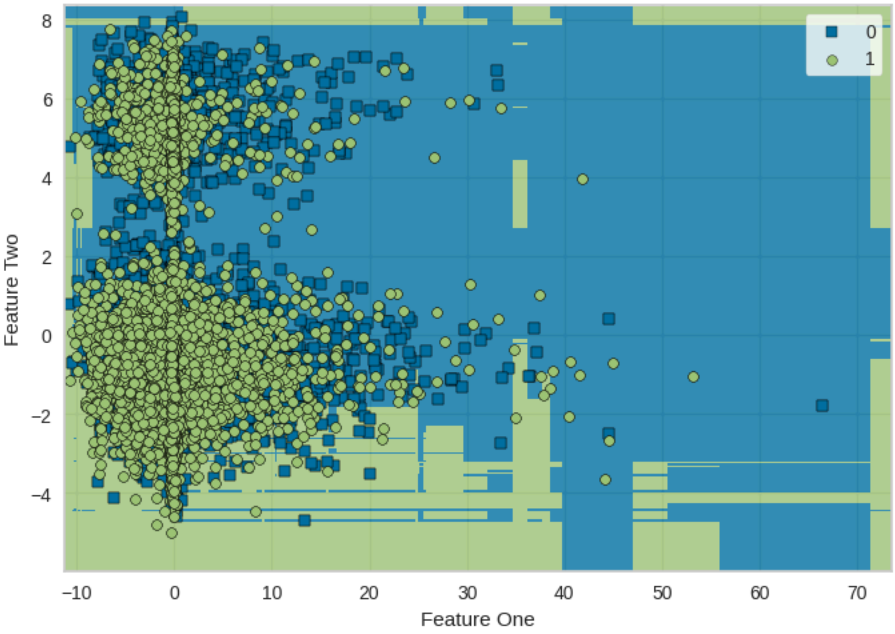
```
# compare baseline models
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.6884	0.7537	0.6864	0.6892	0.6878	0.3767	0.3768	44.8120
ridge	Ridge Classifier	0.6852	0.7485	0.6829	0.6862	0.6845	0.3705	0.3705	1.7920
lda	Linear Discriminant Analysis	0.6846	0.7484	0.6840	0.6850	0.6844	0.3693	0.3693	3.4950
ada	Ada Boost Classifier	0.6814	0.7457	0.6763	0.6834	0.6798	0.3629	0.3629	10.0270
xgboost	Extreme Gradient Boosting	0.6792	0.7406	0.6771	0.6801	0.6786	0.3584	0.3585	5.5920
rf	Random Forest Classifier	0.6779	0.7390	0.6568	0.6858	0.6709	0.3557	0.3561	19.7500
et	Extra Trees Classifier	0.6676	0.7253	0.6458	0.6754	0.6602	0.3353	0.3356	19.4360
lr	Logistic Regression	0.5979	0.6323	0.5853	0.6005	0.5927	0.1958	0.1959	16.3830
dt	Decision Tree Classifier	0.5874	0.5874	0.5899	0.5871	0.5884	0.1749	0.1749	4.7580
nb	Naive Bayes	0.5461	0.6156	0.2451	0.6299	0.3280	0.0923	0.1222	1.7950

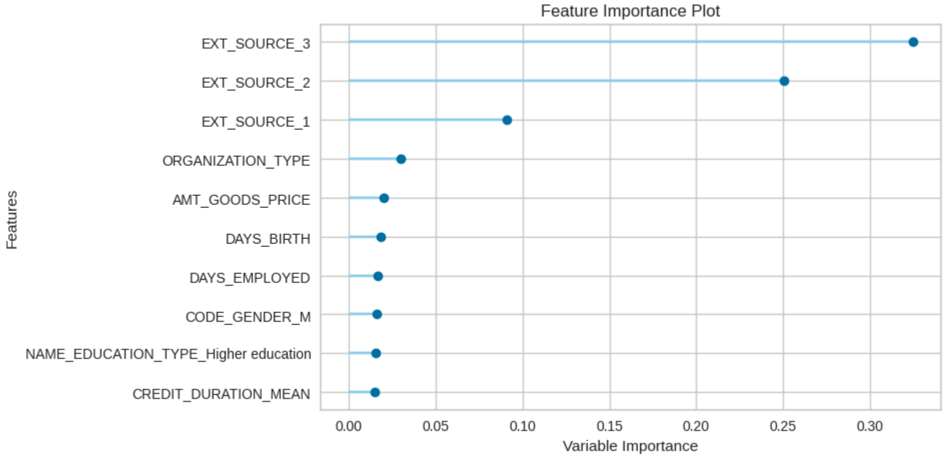
```
# Plot ROC Curve
plot_model(best_model, plot='auc')
```



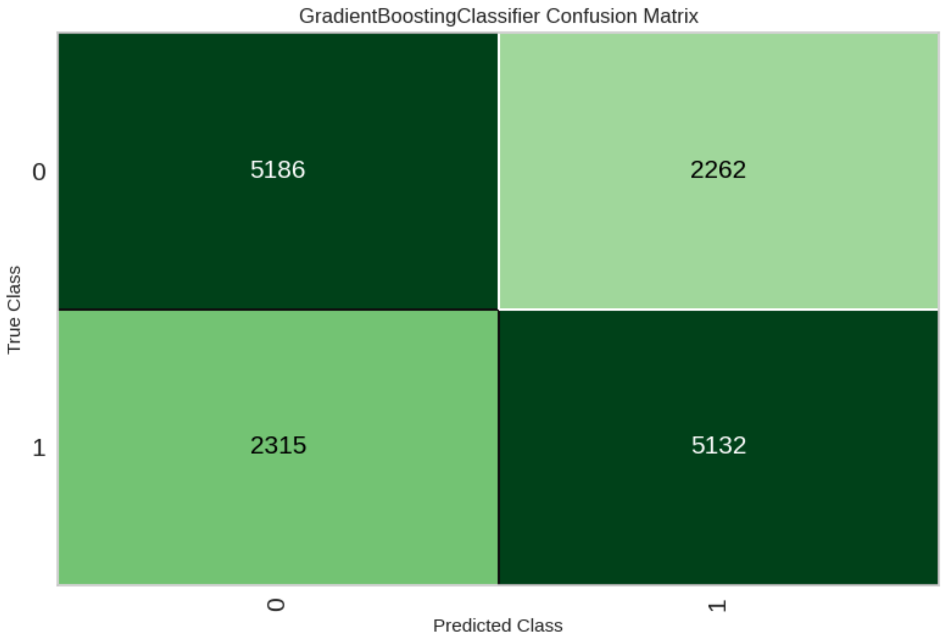
```
# Plot feature importance
plot_model(best_model, plot='boundary')
```



```
# Plot feature importance
plot_model(best_model, plot='feature')
```



```
# Plot feature importance
plot_model(best_model, plot='confusion_matrix')
```



```
# Retrieve feature importance as a DataFrame
importance = pull()
# print(importance)
importance
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
gbc	Gradient Boosting Classifier	0.6884	0.7537	0.6864	0.6892	0.6878	0.3767	0.3768	44.812
ridge	Ridge Classifier	0.6852	0.7485	0.6829	0.6862	0.6845	0.3705	0.3705	1.792
lda	Linear Discriminant Analysis	0.6846	0.7484	0.6840	0.6850	0.6844	0.3693	0.3693	3.495
ada	Ada Boost Classifier	0.6814	0.7457	0.6763	0.6834	0.6798	0.3629	0.3629	10.027
xgboost	Extreme Gradient Boosting	0.6792	0.7406	0.6771	0.6801	0.6786	0.3584	0.3585	5.592
rf	Random Forest Classifier	0.6779	0.7390	0.6568	0.6858	0.6709	0.3557	0.3561	19.750
et	Extra Trees Classifier	0.6676	0.7253	0.6458	0.6754	0.6602	0.3353	0.3356	19.436
lr	Logistic Regression	0.5979	0.6323	0.5853	0.6005	0.5927	0.1958	0.1959	16.383

```
predictions = predict_model(best_model, data=test_data)
```

```
predictions
```

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_
0	100001	Cash loans	F	N	Y	
1	100005	Cash loans	M	N	Y	
2	100013	Cash loans	M	Y	Y	
3	100028	Cash loans	F	N	Y	
4	100038	Cash loans	M	Y	N	
...	...	...	...	...	...	
48739	456221	Cash loans	F	N	Y	
48740	456222	Cash loans	F	N	N	
48741	456223	Cash loans	F	Y	Y	
48742	456224	Cash loans	M	N	N	
48743	456250	Cash loans	F	Y	N	

Change runtime type