Home Credit Default Risk (HCDR)

- **Project Title** : Home Credit Default Risk Prediction
- **Group Name** : Group5_HCDR_CreditFinders
- **Group Number** : 5
- **Group Members**
  - Priyanka Prem Kumar (prpremk@iu.edu)
  - Simi Rajeev (srajeev@iu.edu)
  - Sumedh Ambapkar (suambapk@iu.edu)
  - Junyeon Lee (jl299@iu.edu)

Overview

The course project is based on the Home Credit Default Risk (HCDR) Kaggle Competition. The goal of this project is to predict whether or not a client will repay a loan. In order to make sure that people who struggle to get loans due to insufficient or non-existent credit histories have a positive loan experience, Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.

Tasks to be Tackled

1. Dataset size (688 meg uncompressed) with millions of rows of data
2. 71 Gig of data uncompressed

3. Dealing with missing data.
4. Imbalanced datasets.
5. Summarizing transaction data.
6. Limited Computing resources, thus migrated to Colab.
7. Feature aggregation was resource intesintensive
8. Ideal sessions led to project restarts.
9. Sklearn limited the usage of GPUs.
10. Limitations with GPU allocation.

Phase Leader Plan

PHASE	WEEK	LEADER
1	**1**	Priyanka Prem Kumar
2	**2**	Sumedh Ambapkar
3	**3**	Junyeon Lee
4	**4**	Simi Rajeev

Credit Assignment Plan

Credit Assignment Plan
File Edit View Insert Format Data Tools Extensions Help

CREDIT ASSIGNMENT PLAN TRACKER

Add subjects or course to the 'Subjects' sheet. In this sheet, select your subject, add assignments, status, time required, when you plan to do it and when it's due.

	Subject	Assignment	Status	Time	Start date	Due on
11	Phase2	Everyone was responsible in EDA, Feature Engineering, Aggregating datasets for allowing the pipelines to work seamlessly. Other than that, the documentation reporting was divided amongst us equally.	Done	25hrs	4/8/2025	4/17/2025
12	Phase2	Everyone was responsible in EDA, Feature Engineering, Aggregating datasets for allowing the pipelines to work seamlessly. Other than that, the documentation reporting was divided amongst us equally.	Done	25hrs	4/8/2025	4/17/2025
13	Phase3	Everyone worked on the FeatureEngineering, Transformation, Feature Selection and Aggregating the features, Modeling the Pipeline along with Hyperparameter Tuning using ensemble methods and finally picked the best model for HCDR prediction.	Done	30hrs	4/18/2025	4/22/2025
14	Phase3	Everyone worked on the FeatureEngineering, Transformation, Feature Selection and Aggregating the features, Modeling the Pipeline along with Hyperparameter Tuning using ensemble methods and finally picked the best model for HCDR prediction.	Done	30hrs	4/18/2025	4/22/2025
15	Phase3	Everyone worked on the FeatureEngineering, Transformation, Feature Selection and Aggregating the features, Modeling the Pipeline along with Hyperparameter Tuning using ensemble methods and finally picked the best model for HCDR prediction.	Done	30hrs	4/18/2025	4/22/2025
16	Phase3	Everyone worked on the FeatureEngineering, Transformation, Feature Selection and Aggregating the features, Modeling the Pipeline along with Hyperparameter Tuning using ensemble methods and finally picked the best model for HCDR prediction.	Done	30hrs	4/18/2025	4/22/2025
17	Phase4	In this phase, we were responsible for completing/deploying the ensemble of models' feature selection (PCA, SelectKBest and Variance Threshold techniques) along with developing a single layered neural network and multi layered neural network in the deep learning part. We also performed gap analysis report on the best model chosen by us vs other groups.	Done	35hrs	4/23/2025	4/29/2025
18	Phase4	In this phase, we were responsible for completing/deploying the ensemble of models' feature selection (PCA, SelectKBest and Variance Threshold techniques) along with developing a single layered neural network and multi layered neural network in the deep learning part. We also performed gap analysis report on the best model chosen by us vs other groups.	Done	35hrs	4/23/2025	4/29/2025
19	Phase4	In this phase, we were responsible for completing/deploying the ensemble of models' feature selection (PCA, SelectKBest and Variance Threshold techniques) along with developing a single layered neural network and multi layered neural network in the deep learning part. We also performed gap analysis report on the best model chosen by us vs other groups.	Done	35hrs	4/23/2025	4/29/2025
20	Phase4	In this phase, we were responsible for completing/deploying the ensemble of models' feature selection (PCA, SelectKBest and Variance Threshold techniques) along with developing a single layered neural network and multi layered neural network in the deep learning part. We also performed gap analysis report on the best model chosen by us vs other groups.	Done	35hrs	4/23/2025	4/29/2025

Assignments Subjects

Dataset Description

Background on the dataset

Home Credit is a non-banking financial institution, founded in 1997 in the Czech Republic.

The company operates in 14 countries (including United States, Russia, Kazahstan, Belarus, China, India) and focuses on lending primarily to people with little or no credit history which will either not obtain loans or became victims of untrustworthly lenders.

Home Credit group has over 29 million customers, total assests of 21 billions Euro, over 160 millions loans, with the majority in Asia and and almost half of them in China (as of 19-05-2018).

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data. Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

Data files Overview

There are 7 different sources of data:

- **application_train/application_test:** the main training and testing data with information about each loan application at Home Credit. Every loan has its own row and is identified by the feature SK_ID_CURR. The training application data comes with the TARGET indicating **0: the loan was repaid** or **1: the loan was not repaid**. The target variable defines if the client had payment difficulties meaning he/she had late payment more than X days on at least one of the first Y installments of the loan. Such case is marked as 1 while other all other cases as 0.
- **bureau:** data concerning client's previous credits from other financial institutions. Each previous credit has its own row in bureau, but one loan in the application data can have multiple previous credits.
- **bureau_balance:** monthly data about the previous credits in bureau. Each row is one month of a previous credit, and a single previous credit can have multiple rows, one for each month of the credit length.
- **previous_application:** previous applications for loans at Home Credit of clients who have loans in the application data. Each current loan in the application data can have multiple previous loans. Each previous application has one row and is identified by the feature SK_ID_PREV.
- **POS_CASH_BALANCE:** monthly data about previous point of sale or cash loans clients have had with Home Credit. Each row is one month of a previous point of sale or cash loan, and a single previous loan can have many rows.
- credit_card_balance: monthly data about previous credit cards clients have had with Home Credit. Each row is one month of a credit card balance, and a single credit card can have many rows.
- **installments_payment:** payment history for previous loans at Home Credit. There is one row for every made payment and one row for every missed payment.

In [ ]:

```
# Uncomment if using running it on jupyter lab
# For code autocompletion
# %config Completer.use_jedi = False

# Preprocessing imports
import copy
import numpy as np
import pandas as pd
import os
import gc
import zipfile
import matplotlib.pyplot as plt
# from matplotlib import pyplot
import seaborn as sns
from pandas.plotting import scatter_matrix

# Pipelines
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline, Pipeline, FeatureUnion
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import PolynomialFeatures
from sklearn.decomposition import PCA

# Model imports
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

from sklearn.utils import resample
import sklearn.metrics as metrics
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, log_loss
from sklearn.metrics import classification_report, roc_auc_score, make_scorer
from sklearn.metrics import roc_auc_score, make_scorer, roc_curve, ConfusionMatrixDis
from sklearn.metrics import explained_variance_score
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, log_loss, cla
from scipy import stats
import json
from matplotlib import pyplot
from sklearn.model_selection import train_test_split


from scipy import stats
from time import time, ctime
```

```
# Display help
from IPython.display import display, HTML
import re
import json
import pprint
import warnings
warnings.filterwarnings('ignore')

pprint = pprint.PrettyPrinter().pprint
```

Imports

In [ ]:
```
!pip install lightgbm
```

```
Collecting lightgbm
  Downloading lightgbm-4.6.0-py3-none-manylinux_2_28_x86_64.whl.metadata (17 kB)
Requirement already satisfied: numpy>=1.17.0 in /usr/local/lib/python3.11/dist-packag
es (from lightgbm) (2.0.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from
lightgbm) (1.14.1)
Downloading lightgbm-4.6.0-py3-none-manylinux_2_28_x86_64.whl (3.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.6/3.6 MB 35.4 MB/s eta 0:00:00
Installing collected packages: lightgbm
Successfully installed lightgbm-4.6.0
```

In [ ]:
```
!pip install xgboost
```

```
Collecting xgboost
  Downloading xgboost-3.0.0-py3-none-manylinux_2_28_x86_64.whl.metadata (2.1 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from
xgboost) (2.0.2)
Collecting nvidia-nccl-cu12 (from xgboost)
  Downloading nvidia_nccl_cu12-2.26.2.post1-py3-none-manylinux2014_x86_64.manylinux_2
_17_x86_64.whl.metadata (2.0 kB)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from
xgboost) (1.14.1)
Downloading xgboost-3.0.0-py3-none-manylinux_2_28_x86_64.whl (253.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 253.9/253.9 MB 4.3 MB/s eta 0:00:00
Downloading nvidia_nccl_cu12-2.26.2.post1-py3-none-manylinux2014_x86_64.manylinux_2_1
7_x86_64.whl (291.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 291.7/291.7 MB 3.7 MB/s eta 0:00:00
Installing collected packages: nvidia-nccl-cu12, xgboost
Successfully installed nvidia-nccl-cu12-2.26.2.post1 xgboost-3.0.0
```

In [ ]:
```
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_validate
from sklearn.utils import resample

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.decomposition import PCA
from sklearn.feature_selection import RFE
from sklearn.ensemble import VotingClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif

from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, log_loss, cla
from scipy import stats
import json
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, make_scorer, roc_curve, ConfusionMatrixDis
from sklearn.metrics import explained_variance_score
```

Data Download

In [ ]:

```
# Uncomment if using running it on jupyter lab
# For code autocompletion
# %config Completer.use_jedi = False

# Preprocessing imports
import numpy as np
import pandas as pd
import os
import zipfile
import matplotlib.pyplot as plt
# from matplotlib import pyplot
import seaborn as sns
from pandas.plotting import scatter_matrix

# Pipelines
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline, Pipeline, FeatureUnion
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import PolynomialFeatures
from sklearn.decomposition import PCA

# Model imports
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

from sklearn.utils import resample
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, log_loss
from sklearn.metrics import classification_report, roc_auc_score, make_scorer

from scipy import stats
from time import time, ctime

import re
import json
import pprint
import warnings
warnings.filterwarnings('ignore')

pprint = pprint.PrettyPrinter().pprint

from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_validate
from sklearn.utils import resample

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.decomposition import PCA
from sklearn.feature_selection import RFE
from sklearn.ensemble import VotingClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, log_loss, cla
from scipy import stats
import json
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, make_scorer, roc_curve, ConfusionMatrixDis
from sklearn.metrics import explained_variance_score
#from sklearn.metrics import plot_roc_curve, plot_confusion_matrix, plot_precision_re
from sklearn.metrics import RocCurveDisplay, ConfusionMatrixDisplay, PrecisionRecallD
from sklearn.metrics import RocCurveDisplay, ConfusionMatrixDisplay, PrecisionRecallD
```

unzip the data

In [ ]:

```
import zipfile
import os

DATA_DIR = '/content/Data/'  # Update this to your actual data directory

zip_file = os.path.join(DATA_DIR, 'home-credit-default-risk.zip')
print(f"Unzipping: {zip_file}")
```

```
with zipfile.ZipFile(zip_file, 'r') as zip_ref:
    zip_ref.extractall(path=DATA_DIR)

print("Unzipping completed.")
```

```
Unzipping: /content/Data/home-credit-default-risk.zip
Unzipping completed.
```

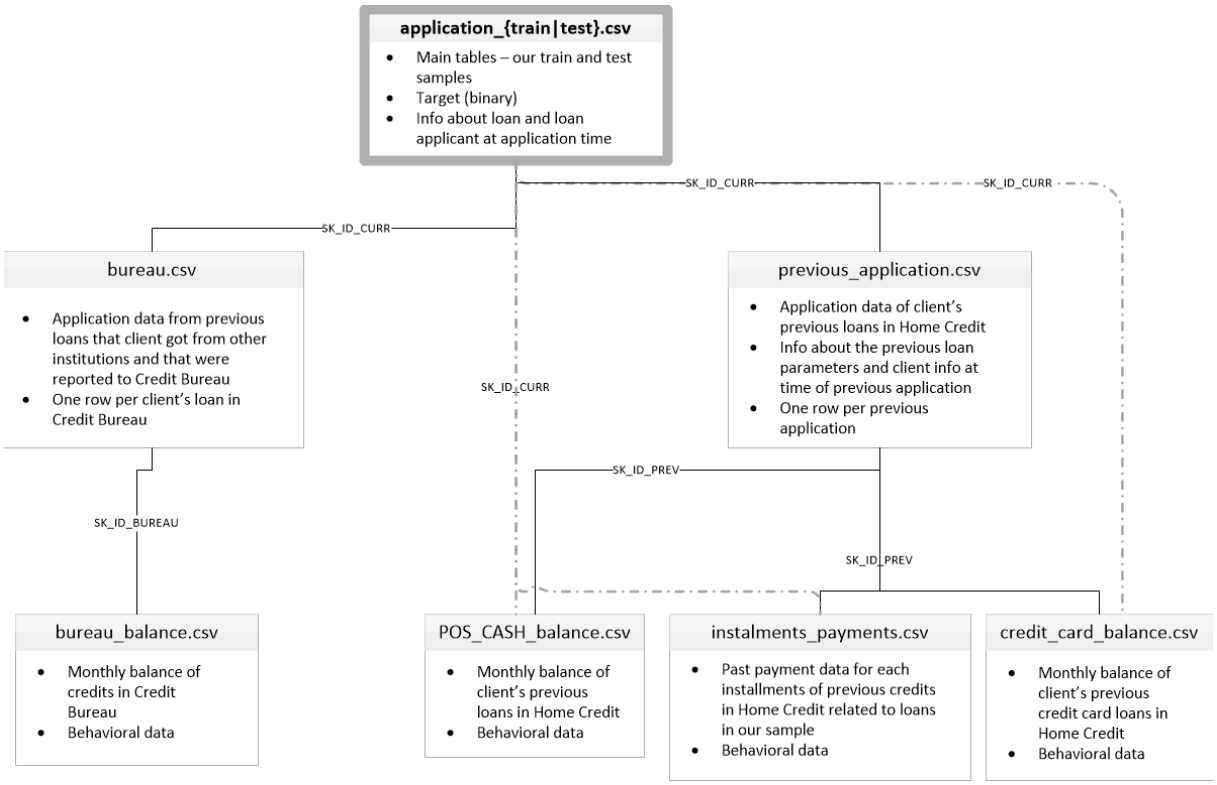
The full dataset consists of 7 tables. There is 1 primary table and 6 secondary tables.

Primary Tables

1. **application_train**
   This Primary table includes the application information for each loan application at Home Credit in one row. This row includes the target variable of whether or not the loan was repaid. We use this field as the basis to determine the feature importance. The target variable is binary in nature based since this is a classification problem.

- '1' - client with payment difficulties: he/she had late payment more than N days on at least one of the first M installments of the loan in our sample
- '0' - all other cases
  The number of variables are 122. The number of data entries are 307,511

1. **application_test**
   This table includes the application information for each loan application at Home Credit in one row. The features are the same as the train data but exclude the target variable
   The number of variables are 121. The number of data entries are 48,744.



In [ ]:

```
def load_data(in_path, name):
    df = pd.read_csv(in_path)
```

```
    print(f"{name}: shape is {df.shape}")
    print(df.info())
    display(df.head(5))
    return df

datasets={}  # lets store the datasets in a dictionary so we can keep track of them e
```

In [ ]:

```
ds_name = 'application_test'
datasets[ds_name] = load_data(os.path.join(DATA_DIR, f'{ds_name}.csv'), ds_name)
```

```
application_test: shape is (48744, 121)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48744 entries, 0 to 48743
Columns: 121 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(40), object(16)
memory usage: 45.0+ MB
None
```

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_C
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	

5 rows × 121 columns

Secondary Tables

1. **Bureau**
   This table includes all previous credits received by a customer from other financial institutions prior to their loan application. There is one row for each previous credit, meaning a many-to-one relationship with the primary table. We could join it with primary table by using current application ID, SK_ID_CURR.
   The number of variables are 17.The number of data entries are 1,716,428.

2. **Bureau Balance**
   This table includes the monthly balance for a previous credit at other financial institutions. There is one row for each monthly balance, meaning a many-to-one relationship with the Bureau table. We could join it with bureau table by using bureau's ID, SK_ID_BUREAU.
   The number of variables are 3. The number of data entries are 27,299,925

3. **Previous Application**
   This table includes previous applications for loans made by the customer at Home Credit. There is one row for each previous application, meaning a many-to-one relationship with the primary table. We could join it with primary table by using current application ID, SK_ID_CURR.

There are four types of contracts:
a. Consumer loan(POS – Credit limit given to buy consumer goods)
b. Cash loan(Client is given cash)
c. Revolving loan(Credit)
d. XNA (Contract type without values)
The number of variables are 37. The number of data entries are 1,670,214

4. **POS CASH Balance**
   This table includes a monthly balance snapshot of a previous point of sale or cash loan that the customer has at Home Credit. There is one row for each monthly balance, meaning a many-to-one relationship with the Previous Application table. We would join it with Previous Application table by using previous application ID, SK_ID_PREV, then join it with primary table by using current application ID, SK_ID_CURR.
   The number of variables are 8. The number of data entries are 10,001,358

5. **Credit Card Balance**
   This table includes a monthly balance snapshot of previous credit cards the customer has with Home Credit. There is one row for each previous monthly balance, meaning a many-to-one relationship with the Previous Application table.We could join it with Previous Application table by using previous application ID, SK_ID_PREV, then join it with primary table by using current application ID, SK_ID_CURR.
   The number of variables are 23. The number of data entries are 3,840,312

6. **Installments Payments**
   This table includes previous repayments made or not made by the customer on credits issued by Home Credit. There is one row for each payment or missed payment, meaning a many-to-one relationship with the Previous Application table. We would join it with Previous Application table by using previous application ID, SK_ID_PREV, then join it with primary table by using current application ID, SK_ID_CURR.
   The number of variables are 8 . The number of data entries are 13,605,401

The application dataset has the most information about the client: Gender, income, family status, education ...

Download all the files

In [ ]:

```
%%time
ds_names = ("application_train", "application_test", "bureau","bureau_balance","credi
            "previous_application","POS_CASH_balance")

for ds_name in ds_names:
    datasets[ds_name] = load_data(os.path.join(DATA_DIR, f'{ds_name}.csv'), ds_name)
```

```
application_train: shape is (307511, 122)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
None
```

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALT
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	
2	100004	0	Revolving loans	M	Y	
3	100006	0	Cash loans	F	N	
4	100007	0	Cash loans	M	N	

5 rows × 122 columns

```
application_test: shape is (48744, 121)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48744 entries, 0 to 48743
Columns: 121 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(40), object(16)
memory usage: 45.0+ MB
None
```

	SK_ID_CURR	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_C
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	

5 rows × 121 columns

```
bureau: shape is (1716428, 17)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1716428 entries, 0 to 1716427
Data columns (total 17 columns):
 #   Column                  Dtype
---  ------                  -----
 0   SK_ID_CURR              int64
 1   SK_ID_BUREAU            int64
 2   CREDIT_ACTIVE           object
 3   CREDIT_CURRENCY         object
 4   DAYS_CREDIT             int64
 5   CREDIT_DAY_OVERDUE      int64
 6   DAYS_CREDIT_ENDDATE     float64
 7   DAYS_ENDDATE_FACT       float64
 8   AMT_CREDIT_MAX_OVERDUE  float64
 9   CNT_CREDIT_PROLONG      int64
 10  AMT_CREDIT_SUM          float64
 11  AMT_CREDIT_SUM_DEBT     float64
 12  AMT_CREDIT_SUM_LIMIT    float64
 13  AMT_CREDIT_SUM_OVERDUE  float64
 14  CREDIT_TYPE             object
 15  DAYS_CREDIT_UPDATE      int64
 16  AMT_ANNUITY             float64
dtypes: float64(8), int64(6), object(3)
memory usage: 222.6+ MB
None
```

	SK_ID_CURR	SK_ID_BUREAU	CREDIT_ACTIVE	CREDIT_CURRENCY	DAYS_CREDIT	CREDIT_DAY_OVERI
0	215354	5714462	Closed	currency 1	-497	
1	215354	5714463	Active	currency 1	-208	
2	215354	5714464	Active	currency 1	-203	
3	215354	5714465	Active	currency 1	-203	
4	215354	5714466	Active	currency 1	-629	

```
bureau_balance: shape is (27299925, 3)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27299925 entries, 0 to 27299924
Data columns (total 3 columns):
 #   Column          Dtype
---  ------          -----
 0   SK_ID_BUREAU    int64
 1   MONTHS_BALANCE  int64
 2   STATUS          object
dtypes: int64(2), object(1)
memory usage: 624.8+ MB
None
```

	SK_ID_BUREAU	MONTHS_BALANCE	STATUS
0	5715448	0	C
1	5715448	-1	C
2	5715448	-2	C
3	5715448	-3	C
4	5715448	-4	C

```
credit_card_balance: shape is (3840312, 23)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3840312 entries, 0 to 3840311
Data columns (total 23 columns):
 #   Column                       Dtype  
---  ------                       -----  
 0   SK_ID_PREV                   int64  
 1   SK_ID_CURR                   int64  
 2   MONTHS_BALANCE               int64  
 3   AMT_BALANCE                  float64
 4   AMT_CREDIT_LIMIT_ACTUAL      int64  
 5   AMT_DRAWINGS_ATM_CURRENT     float64
 6   AMT_DRAWINGS_CURRENT         float64
 7   AMT_DRAWINGS_OTHER_CURRENT   float64
 8   AMT_DRAWINGS_POS_CURRENT     float64
 9   AMT_INST_MIN_REGULARITY      float64
 10  AMT_PAYMENT_CURRENT          float64
 11  AMT_PAYMENT_TOTAL_CURRENT    float64
 12  AMT_RECEIVABLE_PRINCIPAL     float64
 13  AMT_RECIVABLE                float64
 14  AMT_TOTAL_RECEIVABLE         float64
 15  CNT_DRAWINGS_ATM_CURRENT     float64
 16  CNT_DRAWINGS_CURRENT         int64  
 17  CNT_DRAWINGS_OTHER_CURRENT   float64
 18  CNT_DRAWINGS_POS_CURRENT     float64
 19  CNT_INSTALMENT_MATURE_CUM    float64
 20  NAME_CONTRACT_STATUS         object 
 21  SK_DPD                       int64  
 22  SK_DPD_DEF                   int64  
dtypes: float64(15), int64(7), object(1)
memory usage: 673.9+ MB
None
```

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	AMT_BALANCE	AMT_CREDIT_LIMIT_ACTUAL	AMT_D
0	2562384	378907	-6	56.970	135000	
1	2582071	363914	-1	63975.555	45000	
2	1740877	371185	-7	31815.225	450000	
3	1389973	337855	-4	236572.110	225000	
4	1891521	126868	-1	453919.455	450000	

5 rows × 23 columns

```
installments_payments: shape is (13605401, 8)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13605401 entries, 0 to 13605400
Data columns (total 8 columns):
 #   Column                  Dtype
---  ------                  -----
 0   SK_ID_PREV              int64
 1   SK_ID_CURR              int64
 2   NUM_INSTALMENT_VERSION  float64
 3   NUM_INSTALMENT_NUMBER   int64
 4   DAYS_INSTALMENT         float64
 5   DAYS_ENTRY_PAYMENT      float64
 6   AMT_INSTALMENT          float64
 7   AMT_PAYMENT             float64
dtypes: float64(5), int64(3)
memory usage: 830.4 MB
None
```

	SK_ID_PREV	SK_ID_CURR	NUM_INSTALMENT_VERSION	NUM_INSTALMENT_NUMBER	DAYS_INSTALI
0	1054186	161674	1.0	6	-1
1	1330831	151639	0.0	34	-2
2	2085231	193053	2.0	1	
3	2452527	199697	1.0	3	-2
4	2714724	167756	1.0	2	-1

```
previous_application: shape is (1670214, 37)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   SK_ID_PREV                   1670214 non-null  int64  
 1   SK_ID_CURR                   1670214 non-null  int64  
 2   NAME_CONTRACT_TYPE           1670214 non-null  object 
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_DOWN_PAYMENT             774370 non-null   float64
 7   AMT_GOODS_PRICE              1284699 non-null  float64
 8   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object 
 9   HOUR_APPR_PROCESS_START      1670214 non-null  int64  
 10  FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object 
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64  
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object 
 16  NAME_CONTRACT_STATUS         1670214 non-null  object 
 17  DAYS_DECISION                1670214 non-null  int64  
 18  NAME_PAYMENT_TYPE            1670214 non-null  object 
 19  CODE_REJECT_REASON           1670214 non-null  object 
 20  NAME_TYPE_SUITE              849809 non-null   object 
 21  NAME_CLIENT_TYPE             1670214 non-null  object 
 22  NAME_GOODS_CATEGORY          1670214 non-null  object 
 23  NAME_PORTFOLIO               1670214 non-null  object 
 24  NAME_PRODUCT_TYPE            1670214 non-null  object 
 25  CHANNEL_TYPE                 1670214 non-null  object 
 26  SELLERPLACE_AREA             1670214 non-null  int64  
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object 
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object 
 30  PRODUCT_COMBINATION          1669868 non-null  object 
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
None
```

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDI
0	2030495	271877	Consumer loans	1730.430	17145.0	17145
1	2802425	108129	Cash loans	25188.615	607500.0	679671
2	2523466	122040	Cash loans	15060.735	112500.0	136444
3	2819243	176158	Cash loans	47041.335	450000.0	470790
4	1784265	202054	Cash loans	31924.395	337500.0	404055

5 rows × 37 columns

```
POS_CASH_balance: shape is (10001358, 8)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10001358 entries, 0 to 10001357
Data columns (total 8 columns):
 #   Column                Dtype
---  ------                -----
 0   SK_ID_PREV            int64
 1   SK_ID_CURR            int64
 2   MONTHS_BALANCE        int64
 3   CNT_INSTALMENT        float64
 4   CNT_INSTALMENT_FUTURE  float64
 5   NAME_CONTRACT_STATUS   object
 6   SK_DPD                int64
 7   SK_DPD_DEF            int64
dtypes: float64(2), int64(5), object(1)
memory usage: 610.4+ MB
None
```

	SK_ID_PREV	SK_ID_CURR	MONTHS_BALANCE	CNT_INSTALMENT	CNT_INSTALMENT_FUTURE	NAM
0	1803195	182943	-31	48.0	45.0	
1	1715348	367990	-33	36.0	35.0	
2	1784872	397406	-32	12.0	9.0	
3	1903291	269225	-35	48.0	42.0	
4	2341044	334279	-35	36.0	35.0	

```
CPU times: user 32.9 s, sys: 5.4 s, total: 38.3 s
```

In [ ]:

```
for ds_name in datasets.keys():
    print(f'dataset {ds_name:24}: [ {datasets[ds_name].shape[0]:10,}, {datasets[ds_na
```

```
dataset application_test        : [     48,744, 121]
dataset application_train       : [    307,511, 122]
dataset bureau                  : [  1,716,428, 17]
dataset bureau_balance          : [ 27,299,925, 3]
dataset credit_card_balance     : [  3,840,312, 23]
dataset installments_payments   : [ 13,605,401, 8]
dataset previous_application    : [  1,670,214, 37]
dataset POS_CASH_balance        : [ 10,001,358, 8]
```

Exploratory Data Analysis

Exploratory Data Analysis is valuable to this project since it allows us to get closer to the certainty that the future results will be valid, accurately interpreted, and applicable to the proposed solution.

In phase 1 for this project this step involves looking at the summary statistics for each individual table in the model and focusing on the missing data , distribution and its central tendencies such as mean, median, count, min, max and the interquartile ranges.

Categorical and numerical features were looked at to identify anamolies in the data. Specific features were chosen to be visualized based on the correlation and distribution. The highly

correlated features were used to plot the density to evaluate the distributions in comparison to the target.

Data file Statistics

Statistics Helper functions

In [ ]:

```
pd.set_option("display.max_rows", None, "display.max_columns", None)

# Full stats

def stats_summary1(df, df_name):
    print(datasets[df_name].info(verbose=True, null_counts=True ))
    print("-----"*15)
    print(f"Shape of the df {df_name} is {df.shape} \n")
    print("-----"*15)
    print(f"Statistical summary of {df_name} is :")
    print("-----"*15)
    print(f"Description of the df {df_name}:\n")
    print(display(HTML(np.round(datasets['application_train'].describe(),2).to_html()
    #print(f"Description of the df {df_name}:\n",np.round(datasets['application_train

def stats_summary2(df, df_name):
    print(f"Description of the df continued for {df_name}:\n")
    print("-----"*15)
    print("Data type value counts: \n",df.dtypes.value_counts())
    print("\nReturn number of unique elements in the object. \n")
    print(df.select_dtypes('object').apply(pd.Series.nunique, axis = 0))


# List the categorical and Numerical features of a DF
def feature_datatypes_groups(df, df_name):
    df_dtypes = df.columns.to_series().groupby(df.dtypes).groups
    print("-----"*15)
    print(f"Categorical and Numerical(int + float) features  of {df_name}.")
    print("-----"*15)
    print()
    for k, v in df_dtypes.items():
        print({k.name: v})
        print("---"*10)
    print("\n \n")

# Null data list and plot.
def null_data_plot(df, df_name):
    percent = (df.isnull().sum()/df.isnull().count()*100).sort_values(ascending = Fal
    sum_missing = df.isna().sum().sort_values(ascending = False)
    missing_data  = pd.concat([percent, sum_missing], axis=1, keys=['Percent', "Train
    missing_data=missing_data[missing_data['Percent'] > 0]
    print("-----"*15)
    print("-----"*15)
    print('\n The Missing Data: \n')
#     display(missing_data)  # display few
    if len(missing_data)==0:
      print("No missing Data")
    else:
      display(HTML(missing_data.to_html()))  # display all the rows
      print("-----"*15)
      if len(df.columns)> 35:
```

```
        f,ax =plt.subplots(figsize=(8,15))
      else:
        f,ax =plt.subplots()
      #plt.xticks(rotation='90')
      #fig=sns.barplot(missing_data.index, missing_data["Percent"],alpha=0.8)
      #plt.xlabel('Features', fontsize=15)
      #plt.ylabel('Percent of missing values', fontsize=15)
      plt.title(f'Percent missing data for {df_name}.', fontsize=10)
      fig=sns.barplot(missing_data["Percent"],missing_data.index ,alpha=0.8)
      plt.xlabel('Percent of missing values', fontsize=10)
      plt.ylabel('Features', fontsize=10)
      return missing_data


# Full consolidation of all the stats function.
def display_stats(df, df_name):
    print("--"*40)
    print(" "*20 + '\033[1m'+ df_name +  '\033[0m' +" "*20)
    print("--"*40)
    stats_summary1(df, df_name)

def display_feature_info(df, df_name):
    stats_summary2(df, df_name)
    feature_datatypes_groups(df, df_name)
    null_data_plot(df, df_name)
```

Summary of application_train

In [ ]:
```
(datasets['application_train'].dtypes).unique()
```

Out[ ]:
```
array([dtype('int64'), dtype('O'), dtype('float64')], dtype=object)
```

In [ ]:
```
display_stats(datasets['application_train'], 'application_train')
```

```
--------------------------------------------------------------------------------
                    application_train
--------------------------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 122 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   SK_ID_CURR                     307511 non-null  int64
 1   TARGET                         307511 non-null  int64
 2   NAME_CONTRACT_TYPE             307511 non-null  object
 3   CODE_GENDER                    307511 non-null  object
 4   FLAG_OWN_CAR                   307511 non-null  object
 5   FLAG_OWN_REALTY                307511 non-null  object
 6   CNT_CHILDREN                   307511 non-null  int64
 7   AMT_INCOME_TOTAL               307511 non-null  float64
 8   AMT_CREDIT                     307511 non-null  float64
 9   AMT_ANNUITY                    307499 non-null  float64
 10  AMT_GOODS_PRICE                307233 non-null  float64
 11  NAME_TYPE_SUITE                306219 non-null  object
 12  NAME_INCOME_TYPE               307511 non-null  object
 13  NAME_EDUCATION_TYPE            307511 non-null  object
 14  NAME_FAMILY_STATUS             307511 non-null  object
 15  NAME_HOUSING_TYPE              307511 non-null  object
 16  REGION_POPULATION_RELATIVE     307511 non-null  float64
 17  DAYS_BIRTH                     307511 non-null  int64
 18  DAYS_EMPLOYED                  307511 non-null  int64
 19  DAYS_REGISTRATION              307511 non-null  float64
 20  DAYS_ID_PUBLISH                307511 non-null  int64
 21  OWN_CAR_AGE                    104582 non-null  float64
 22  FLAG_MOBIL                     307511 non-null  int64
 23  FLAG_EMP_PHONE                 307511 non-null  int64
 24  FLAG_WORK_PHONE                307511 non-null  int64
 25  FLAG_CONT_MOBILE               307511 non-null  int64
 26  FLAG_PHONE                     307511 non-null  int64
 27  FLAG_EMAIL                     307511 non-null  int64
 28  OCCUPATION_TYPE                211120 non-null  object
 29  CNT_FAM_MEMBERS                307509 non-null  float64
 30  REGION_RATING_CLIENT           307511 non-null  int64
 31  REGION_RATING_CLIENT_W_CITY    307511 non-null  int64
 32  WEEKDAY_APPR_PROCESS_START     307511 non-null  object
 33  HOUR_APPR_PROCESS_START        307511 non-null  int64
 34  REG_REGION_NOT_LIVE_REGION     307511 non-null  int64
 35  REG_REGION_NOT_WORK_REGION     307511 non-null  int64
 36  LIVE_REGION_NOT_WORK_REGION    307511 non-null  int64
 37  REG_CITY_NOT_LIVE_CITY         307511 non-null  int64
 38  REG_CITY_NOT_WORK_CITY         307511 non-null  int64
 39  LIVE_CITY_NOT_WORK_CITY        307511 non-null  int64
 40  ORGANIZATION_TYPE              307511 non-null  object
 41  EXT_SOURCE_1                   134133 non-null  float64
 42  EXT_SOURCE_2                   306851 non-null  float64
 43  EXT_SOURCE_3                   246546 non-null  float64
 44  APARTMENTS_AVG                 151450 non-null  float64
 45  BASEMENTAREA_AVG               127568 non-null  float64
 46  YEARS_BEGINEXPLUATATION_AVG    157504 non-null  float64
 47  YEARS_BUILD_AVG                103023 non-null  float64
 48  COMMONAREA_AVG                 92646 non-null   float64
 49  ELEVATORS_AVG                  143620 non-null  float64
 50  ENTRANCES_AVG                  152683 non-null  float64
 51  FLOORSMAX_AVG                  154491 non-null  float64
```

```
52  FLOORSMIN_AVG                 98869 non-null   float64
53  LANDAREA_AVG                  124921 non-null  float64
54  LIVINGAPARTMENTS_AVG          97312 non-null   float64
55  LIVINGAREA_AVG                153161 non-null  float64
56  NONLIVINGAPARTMENTS_AVG       93997 non-null   float64
57  NONLIVINGAREA_AVG             137829 non-null  float64
58  APARTMENTS_MODE               151450 non-null  float64
59  BASEMENTAREA_MODE             127568 non-null  float64
60  YEARS_BEGINEXPLUATATION_MODE  157504 non-null  float64
61  YEARS_BUILD_MODE              103023 non-null  float64
62  COMMONAREA_MODE               92646 non-null   float64
63  ELEVATORS_MODE                143620 non-null  float64
64  ENTRANCES_MODE                152683 non-null  float64
65  FLOORSMAX_MODE                154491 non-null  float64
66  FLOORSMIN_MODE                98869 non-null   float64
67  LANDAREA_MODE                 124921 non-null  float64
68  LIVINGAPARTMENTS_MODE         97312 non-null   float64
69  LIVINGAREA_MODE               153161 non-null  float64
70  NONLIVINGAPARTMENTS_MODE      93997 non-null   float64
71  NONLIVINGAREA_MODE            137829 non-null  float64
72  APARTMENTS_MEDI               151450 non-null  float64
73  BASEMENTAREA_MEDI             127568 non-null  float64
74  YEARS_BEGINEXPLUATATION_MEDI  157504 non-null  float64
75  YEARS_BUILD_MEDI              103023 non-null  float64
76  COMMONAREA_MEDI               92646 non-null   float64
77  ELEVATORS_MEDI                143620 non-null  float64
78  ENTRANCES_MEDI                152683 non-null  float64
79  FLOORSMAX_MEDI                154491 non-null  float64
80  FLOORSMIN_MEDI                98869 non-null   float64
81  LANDAREA_MEDI                 124921 non-null  float64
82  LIVINGAPARTMENTS_MEDI         97312 non-null   float64
83  LIVINGAREA_MEDI               153161 non-null  float64
84  NONLIVINGAPARTMENTS_MEDI      93997 non-null   float64
85  NONLIVINGAREA_MEDI            137829 non-null  float64
86  FONDKAPREMONT_MODE            97216 non-null   object
87  HOUSETYPE_MODE                153214 non-null  object
88  TOTALAREA_MODE                159080 non-null  float64
89  WALLSMATERIAL_MODE            151170 non-null  object
90  EMERGENCYSTATE_MODE           161756 non-null  object
91  OBS_30_CNT_SOCIAL_CIRCLE      306490 non-null  float64
92  DEF_30_CNT_SOCIAL_CIRCLE      306490 non-null  float64
93  OBS_60_CNT_SOCIAL_CIRCLE      306490 non-null  float64
94  DEF_60_CNT_SOCIAL_CIRCLE      306490 non-null  float64
95  DAYS_LAST_PHONE_CHANGE        307510 non-null  float64
96  FLAG_DOCUMENT_2               307511 non-null  int64
97  FLAG_DOCUMENT_3               307511 non-null  int64
98  FLAG_DOCUMENT_4               307511 non-null  int64
99  FLAG_DOCUMENT_5               307511 non-null  int64
100 FLAG_DOCUMENT_6               307511 non-null  int64
101 FLAG_DOCUMENT_7               307511 non-null  int64
102 FLAG_DOCUMENT_8               307511 non-null  int64
103 FLAG_DOCUMENT_9               307511 non-null  int64
104 FLAG_DOCUMENT_10              307511 non-null  int64
105 FLAG_DOCUMENT_11              307511 non-null  int64
106 FLAG_DOCUMENT_12              307511 non-null  int64
107 FLAG_DOCUMENT_13              307511 non-null  int64
108 FLAG_DOCUMENT_14              307511 non-null  int64
109 FLAG_DOCUMENT_15              307511 non-null  int64
110 FLAG_DOCUMENT_16              307511 non-null  int64
111 FLAG_DOCUMENT_17              307511 non-null  int64
```

```
 112 FLAG_DOCUMENT_18              307511 non-null  int64
 113 FLAG_DOCUMENT_19              307511 non-null  int64
 114 FLAG_DOCUMENT_20              307511 non-null  int64
 115 FLAG_DOCUMENT_21              307511 non-null  int64
 116 AMT_REQ_CREDIT_BUREAU_HOUR    265992 non-null  float64
 117 AMT_REQ_CREDIT_BUREAU_DAY     265992 non-null  float64
 118 AMT_REQ_CREDIT_BUREAU_WEEK    265992 non-null  float64
 119 AMT_REQ_CREDIT_BUREAU_MON     265992 non-null  float64
 120 AMT_REQ_CREDIT_BUREAU_QRT     265992 non-null  float64
 121 AMT_REQ_CREDIT_BUREAU_YEAR    265992 non-null  float64
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
None
--------------------------------------------------------------------------
Shape of the df application_train is (307511, 122)

--------------------------------------------------------------------------
Statistical summary of application_train is :
--------------------------------------------------------------------------
Description of the df application_train:
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	A
count	307511.00	307511.00	307511.00	3.075110e+05	307511.00	307499.00	
mean	278180.52	0.08	0.42	1.687979e+05	599026.00	27108.57	
std	102790.18	0.27	0.72	2.371231e+05	402490.78	14493.74	
min	100002.00	0.00	0.00	2.565000e+04	45000.00	1615.50	
25%	189145.50	0.00	0.00	1.125000e+05	270000.00	16524.00	
50%	278202.00	0.00	0.00	1.471500e+05	513531.00	24903.00	
75%	367142.50	0.00	1.00	2.025000e+05	808650.00	34596.00	
max	456255.00	1.00	19.00	1.170000e+08	4050000.00	258025.50	

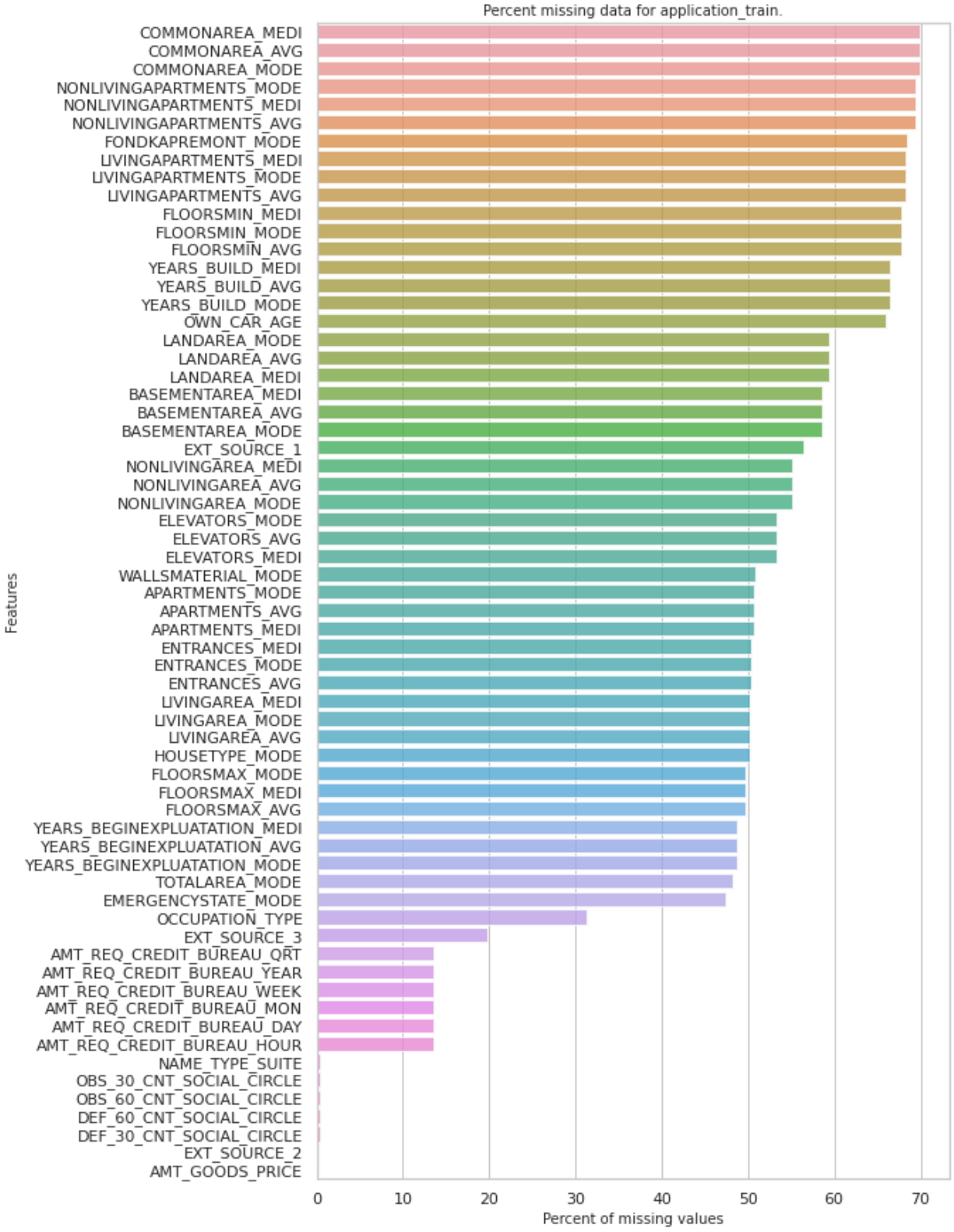
In [ ]:
```
display_feature_info(datasets['application_train'], 'application_train')
```

```
Description of the df continued for application_train:

------------------------------------------------------------------------------
Data type value counts:
 float64    65
int64      41
object     16
dtype: int64

Return number of unique elements in the object.

NAME_CONTRACT_TYPE             2
CODE_GENDER                    3
FLAG_OWN_CAR                   2
FLAG_OWN_REALTY                2
NAME_TYPE_SUITE                7
NAME_INCOME_TYPE               8
NAME_EDUCATION_TYPE            5
NAME_FAMILY_STATUS             6
NAME_HOUSING_TYPE              6
OCCUPATION_TYPE               18
WEEKDAY_APPR_PROCESS_START     7
ORGANIZATION_TYPE             58
FONDKAPREMONT_MODE             4
HOUSETYPE_MODE                 3
WALLSMATERIAL_MODE             7
EMERGENCYSTATE_MODE            2
dtype: int64
------------------------------------------------------------------------------
Categorical and Numerical(int + float) features  of application_train.
------------------------------------------------------------------------------

{'int64': Index(['SK_ID_CURR', 'TARGET', 'CNT_CHILDREN', 'DAYS_BIRTH', 'DAYS_EMPLOYE
D',
       'DAYS_ID_PUBLISH', 'FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE',
       'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'HOUR_APPR_PROCESS_START',
       'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'FLAG_DOCUMENT_2',
       'FLAG_DOCUMENT_3', 'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5',
       'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8',
       'FLAG_DOCUMENT_9', 'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11',
       'FLAG_DOCUMENT_12', 'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14',
       'FLAG_DOCUMENT_15', 'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17',
       'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20',
       'FLAG_DOCUMENT_21'],
      dtype='object')}
------------------------------
{'float64': Index(['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRIC
E',
       'REGION_POPULATION_RELATIVE', 'DAYS_REGISTRATION', 'OWN_CAR_AGE',
       'CNT_FAM_MEMBERS', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3',
       'APARTMENTS_AVG', 'BASEMENTAREA_AVG', 'YEARS_BEGINEXPLUATATION_AVG',
       'YEARS_BUILD_AVG', 'COMMONAREA_AVG', 'ELEVATORS_AVG', 'ENTRANCES_AVG',
       'FLOORSMAX_AVG', 'FLOORSMIN_AVG', 'LANDAREA_AVG',
       'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_AVG', 'NONLIVINGAPARTMENTS_AVG',
       'NONLIVINGAREA_AVG', 'APARTMENTS_MODE', 'BASEMENTAREA_MODE',
       'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BUILD_MODE', 'COMMONAREA_MODE',
       'ELEVATORS_MODE', 'ENTRANCES_MODE', 'FLOORSMAX_MODE', 'FLOORSMIN_MODE',
```

```
       'LANDAREA_MODE', 'LIVINGAPARTMENTS_MODE', 'LIVINGAREA_MODE',
       'NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAREA_MODE', 'APARTMENTS_MEDI',
       'BASEMENTAREA_MEDI', 'YEARS_BEGINEXPLUATATION_MEDI', 'YEARS_BUILD_MEDI',
       'COMMONAREA_MEDI', 'ELEVATORS_MEDI', 'ENTRANCES_MEDI', 'FLOORSMAX_MEDI',
       'FLOORSMIN_MEDI', 'LANDAREA_MEDI', 'LIVINGAPARTMENTS_MEDI',
       'LIVINGAREA_MEDI', 'NONLIVINGAPARTMENTS_MEDI', 'NONLIVINGAREA_MEDI',
       'TOTALAREA_MODE', 'OBS_30_CNT_SOCIAL_CIRCLE',
       'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE',
       'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')}
------------------------------
{'object': Index(['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REAL
TY',
       'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE',
       'WEEKDAY_APPR_PROCESS_START', 'ORGANIZATION_TYPE', 'FONDKAPREMONT_MODE',
       'HOUSETYPE_MODE', 'WALLSMATERIAL_MODE', 'EMERGENCYSTATE_MODE'],
      dtype='object')}
------------------------------



--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

 The Missing Data:
```

	Percent	Train Missing Count
COMMONAREA_MEDI	69.87	214865
COMMONAREA_AVG	69.87	214865
COMMONAREA_MODE	69.87	214865
NONLIVINGAPARTMENTS_MODE	69.43	213514
NONLIVINGAPARTMENTS_MEDI	69.43	213514
NONLIVINGAPARTMENTS_AVG	69.43	213514
FONDKAPREMONT_MODE	68.39	210295
LIVINGAPARTMENTS_MEDI	68.35	210199
LIVINGAPARTMENTS_MODE	68.35	210199
LIVINGAPARTMENTS_AVG	68.35	210199
FLOORSMIN_MEDI	67.85	208642
FLOORSMIN_MODE	67.85	208642
FLOORSMIN_AVG	67.85	208642
YEARS_BUILD_MEDI	66.50	204488
YEARS_BUILD_AVG	66.50	204488
YEARS_BUILD_MODE	66.50	204488
OWN_CAR_AGE	65.99	202929
LANDAREA_MODE	59.38	182590
LANDAREA_AVG	59.38	182590
LANDAREA_MEDI	59.38	182590
BASEMENTAREA_MEDI	58.52	179943
BASEMENTAREA_AVG	58.52	179943
BASEMENTAREA_MODE	58.52	179943
EXT_SOURCE_1	56.38	173378
NONLIVINGAREA_MEDI	55.18	169682
NONLIVINGAREA_AVG	55.18	169682
NONLIVINGAREA_MODE	55.18	169682
ELEVATORS_MODE	53.30	163891
ELEVATORS_AVG	53.30	163891
ELEVATORS_MEDI	53.30	163891
WALLSMATERIAL_MODE	50.84	156341
APARTMENTS_MODE	50.75	156061
APARTMENTS_AVG	50.75	156061
APARTMENTS_MEDI	50.75	156061

	Percent	Train Missing Count
ENTRANCES_MEDI	50.35	154828
ENTRANCES_MODE	50.35	154828
ENTRANCES_AVG	50.35	154828
LIVINGAREA_MEDI	50.19	154350
LIVINGAREA_MODE	50.19	154350
LIVINGAREA_AVG	50.19	154350
HOUSETYPE_MODE	50.18	154297
FLOORSMAX_MODE	49.76	153020
FLOORSMAX_MEDI	49.76	153020
FLOORSMAX_AVG	49.76	153020
YEARS_BEGINEXPLUATATION_MEDI	48.78	150007
YEARS_BEGINEXPLUATATION_AVG	48.78	150007
YEARS_BEGINEXPLUATATION_MODE	48.78	150007
TOTALAREA_MODE	48.27	148431
EMERGENCYSTATE_MODE	47.40	145755
OCCUPATION_TYPE	31.35	96391
EXT_SOURCE_3	19.83	60965
AMT_REQ_CREDIT_BUREAU_QRT	13.50	41519
AMT_REQ_CREDIT_BUREAU_YEAR	13.50	41519
AMT_REQ_CREDIT_BUREAU_WEEK	13.50	41519
AMT_REQ_CREDIT_BUREAU_MON	13.50	41519
AMT_REQ_CREDIT_BUREAU_DAY	13.50	41519
AMT_REQ_CREDIT_BUREAU_HOUR	13.50	41519
NAME_TYPE_SUITE	0.42	1292
OBS_30_CNT_SOCIAL_CIRCLE	0.33	1021
OBS_60_CNT_SOCIAL_CIRCLE	0.33	1021
DEF_60_CNT_SOCIAL_CIRCLE	0.33	1021
DEF_30_CNT_SOCIAL_CIRCLE	0.33	1021
EXT_SOURCE_2	0.21	660
AMT_GOODS_PRICE	0.09	278



Days Employed

In [ ]:

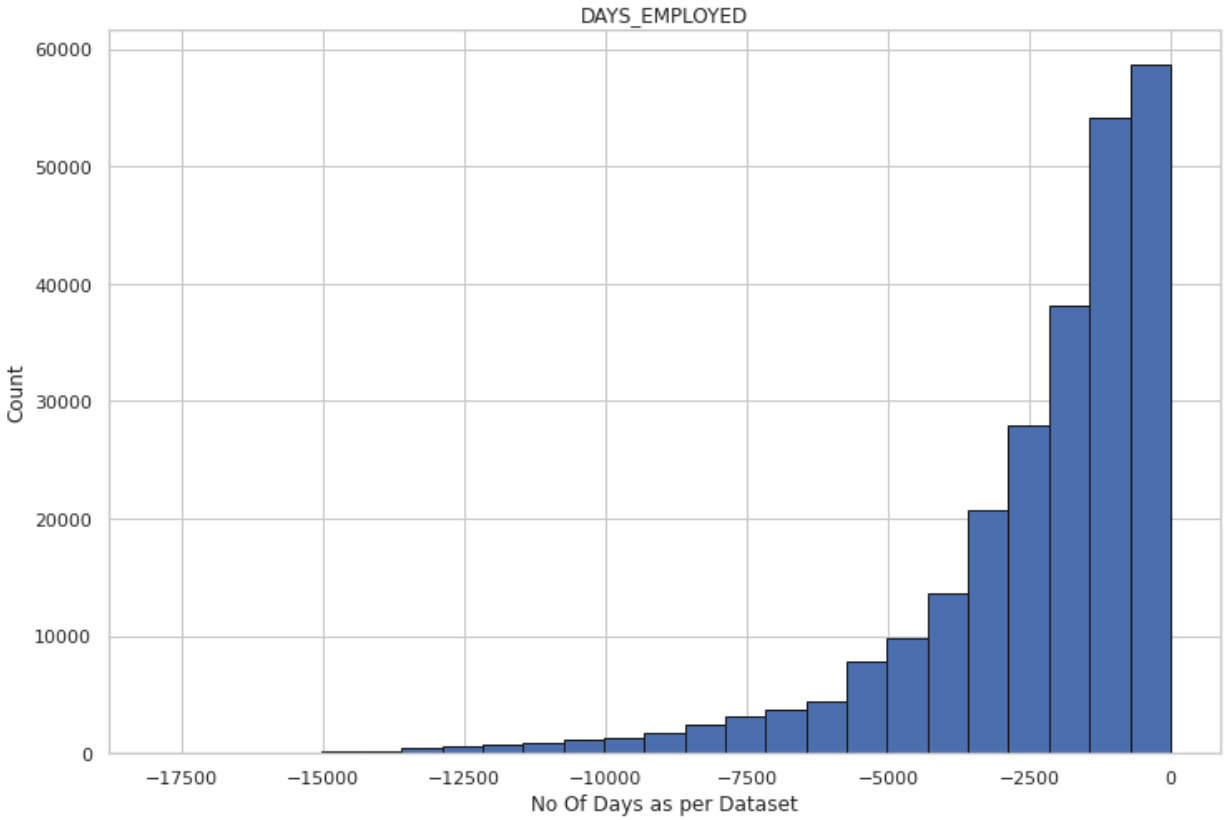
```
datasets["application_train"]['DAYS_EMPLOYED'].describe()
```

Out[ ]:
```
count    307511.000000
mean      63815.045904
std      141275.766519
min      -17912.000000
25%       -2760.000000
50%       -1213.000000
75%        -289.000000
max      365243.000000
Name: DAYS_EMPLOYED, dtype: float64
```

In [ ]:
```
anom_days_employed = datasets["application_train"][datasets["application_train"]['DAY
norm_days_employed = datasets["application_train"][datasets["application_train"]['DAY
print(anom_days_employed.shape)

dr_anom = anom_days_employed['TARGET'].mean()*100
dr_norm = norm_days_employed['TARGET'].mean()*100

print('Default rate (Anomaly): {:.2f}'.format(dr_anom))
print('Default rate (Normal): {:.2f}'.format(dr_norm))

pct_anom_days_employed = (anom_days_employed.shape[0]/datasets["application_train"].s
print(pct_anom_days_employed)
```

```
(55374, 122)
Default rate (Anomaly): 5.40
Default rate (Normal): 8.66
18.00716071945394
```

In [ ]:
```
df_app_train=datasets["application_train"].copy()
df_app_train['DAYS_EMPLOYED_ANOM'] = df_app_train['DAYS_EMPLOYED'] == 365243
df_app_train['DAYS_EMPLOYED'].replace({365243:np.nan}, inplace=True)
plt.hist(df_app_train['DAYS_EMPLOYED'],edgecolor = 'k', bins = 25)
plt.title('DAYS_EMPLOYED'); plt.xlabel('No Of Days as per Dataset'); plt.ylabel('Coun
```



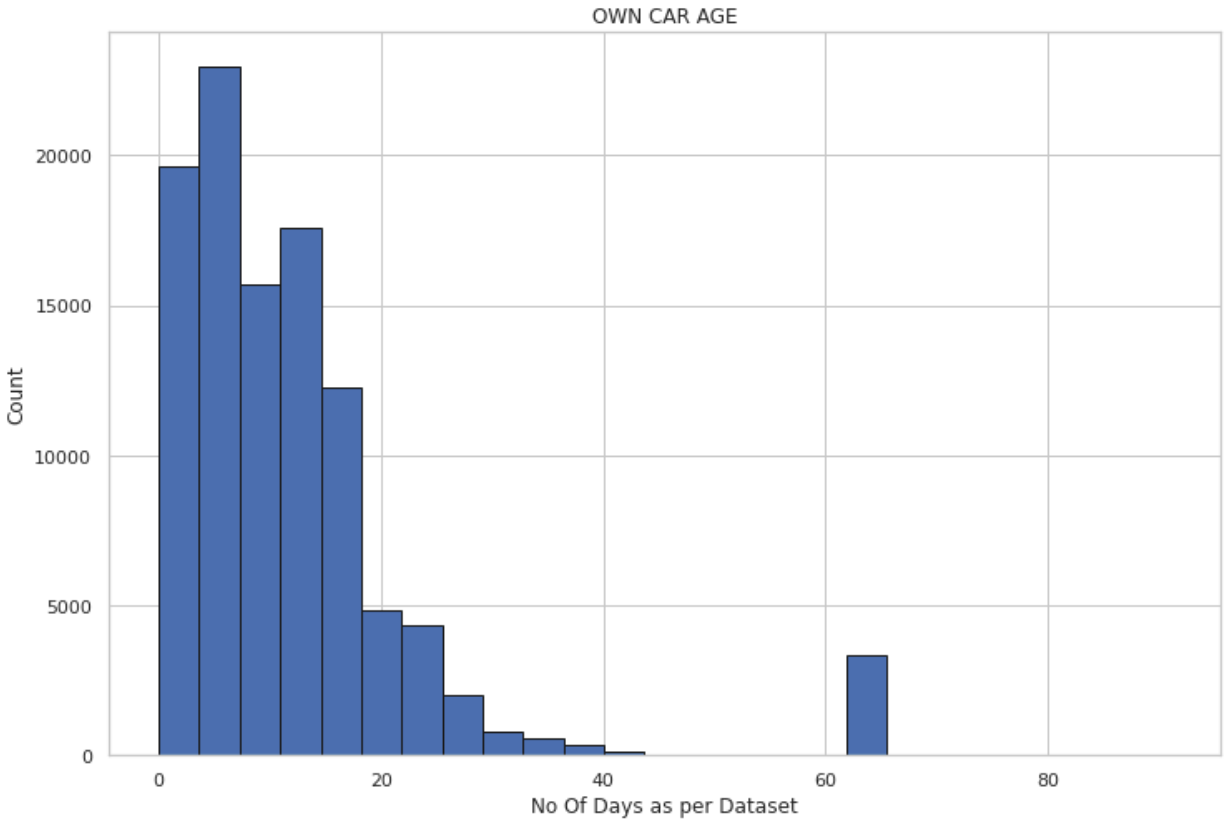
In [ ]:
```
gc.collect()
```

Out[ ]:
```
18405
```

The bins above histogram shows that the data is not logical and this feature needs to be further investigated for imbalances. Number of days employed would show a steady source of income and could be a useful feature for predicting risk

Own Car Age

In [ ]:
```
plt.hist(datasets["application_train"]['OWN_CAR_AGE'],edgecolor = 'k', bins = 25)
plt.title('OWN CAR AGE'); plt.xlabel('No Of Days as per Dataset'); plt.ylabel('Count'
```



We see that those who have cars over 60 years old have a number of applications (i.e., 3339). This could a good area to investigate risk

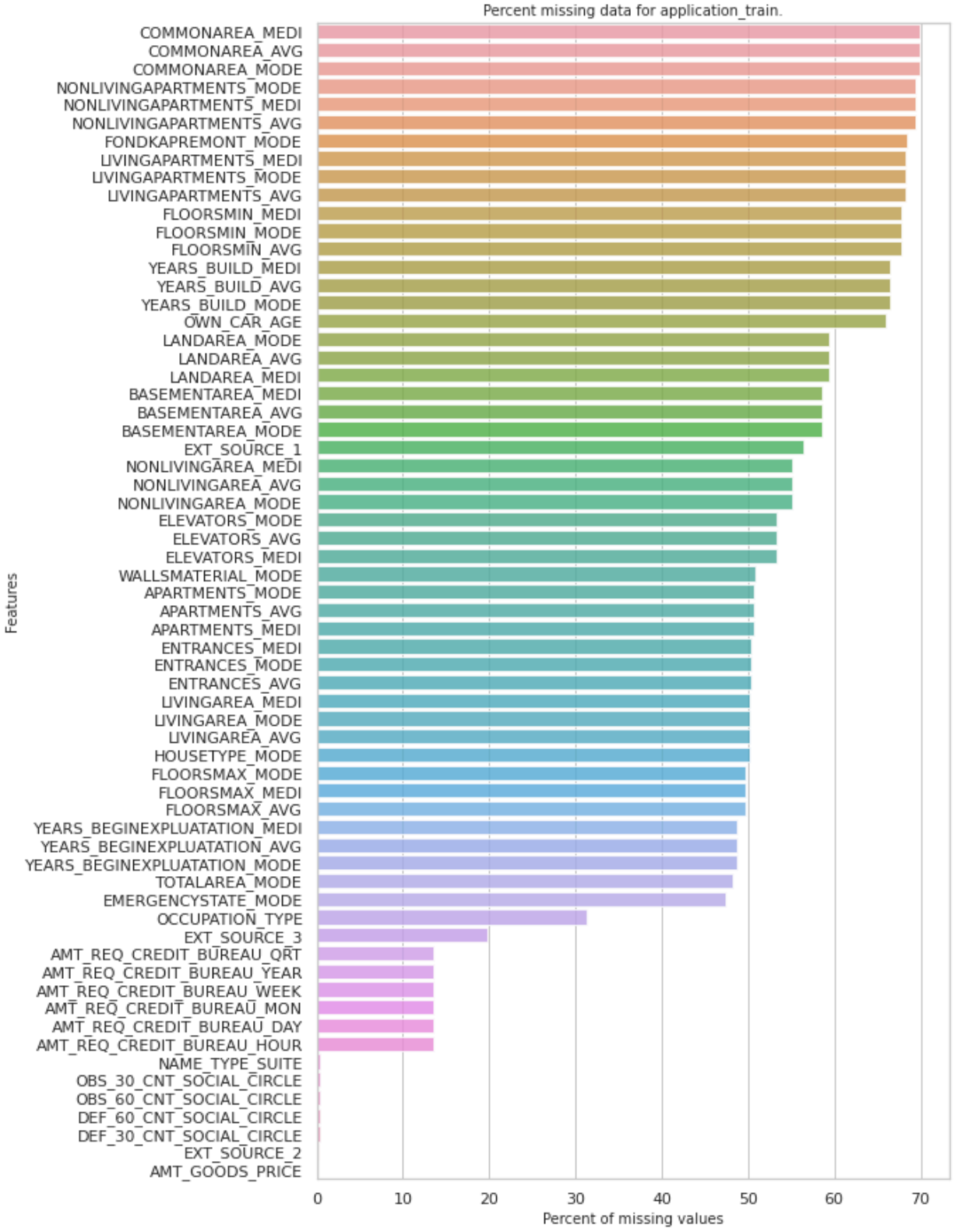
In [ ]:
```
display_feature_info(datasets['application_train'], 'application_train')
```

```
Description of the df continued for application_train:

------------------------------------------------------------------------------
Data type value counts:
 float64    65
int64      41
object     16
dtype: int64

Return number of unique elements in the object.

NAME_CONTRACT_TYPE             2
CODE_GENDER                    3
FLAG_OWN_CAR                   2
FLAG_OWN_REALTY                2
NAME_TYPE_SUITE                7
NAME_INCOME_TYPE               8
NAME_EDUCATION_TYPE            5
NAME_FAMILY_STATUS             6
NAME_HOUSING_TYPE              6
OCCUPATION_TYPE               18
WEEKDAY_APPR_PROCESS_START     7
ORGANIZATION_TYPE             58
FONDKAPREMONT_MODE             4
HOUSETYPE_MODE                 3
WALLSMATERIAL_MODE             7
EMERGENCYSTATE_MODE            2
dtype: int64
------------------------------------------------------------------------------
Categorical and Numerical(int + float) features  of application_train.
------------------------------------------------------------------------------

{'int64': Index(['SK_ID_CURR', 'TARGET', 'CNT_CHILDREN', 'DAYS_BIRTH', 'DAYS_EMPLOYE
D',
       'DAYS_ID_PUBLISH', 'FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE',
       'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'HOUR_APPR_PROCESS_START',
       'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'FLAG_DOCUMENT_2',
       'FLAG_DOCUMENT_3', 'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5',
       'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8',
       'FLAG_DOCUMENT_9', 'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11',
       'FLAG_DOCUMENT_12', 'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14',
       'FLAG_DOCUMENT_15', 'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17',
       'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20',
       'FLAG_DOCUMENT_21'],
      dtype='object')}
------------------------------
{'float64': Index(['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRIC
E',
       'REGION_POPULATION_RELATIVE', 'DAYS_REGISTRATION', 'OWN_CAR_AGE',
       'CNT_FAM_MEMBERS', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3',
       'APARTMENTS_AVG', 'BASEMENTAREA_AVG', 'YEARS_BEGINEXPLUATATION_AVG',
       'YEARS_BUILD_AVG', 'COMMONAREA_AVG', 'ELEVATORS_AVG', 'ENTRANCES_AVG',
       'FLOORSMAX_AVG', 'FLOORSMIN_AVG', 'LANDAREA_AVG',
       'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_AVG', 'NONLIVINGAPARTMENTS_AVG',
       'NONLIVINGAREA_AVG', 'APARTMENTS_MODE', 'BASEMENTAREA_MODE',
       'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BUILD_MODE', 'COMMONAREA_MODE',
       'ELEVATORS_MODE', 'ENTRANCES_MODE', 'FLOORSMAX_MODE', 'FLOORSMIN_MODE',
```

```
       'LANDAREA_MODE', 'LIVINGAPARTMENTS_MODE', 'LIVINGAREA_MODE',
       'NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAREA_MODE', 'APARTMENTS_MEDI',
       'BASEMENTAREA_MEDI', 'YEARS_BEGINEXPLUATATION_MEDI', 'YEARS_BUILD_MEDI',
       'COMMONAREA_MEDI', 'ELEVATORS_MEDI', 'ENTRANCES_MEDI', 'FLOORSMAX_MEDI',
       'FLOORSMIN_MEDI', 'LANDAREA_MEDI', 'LIVINGAPARTMENTS_MEDI',
       'LIVINGAREA_MEDI', 'NONLIVINGAPARTMENTS_MEDI', 'NONLIVINGAREA_MEDI',
       'TOTALAREA_MODE', 'OBS_30_CNT_SOCIAL_CIRCLE',
       'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE',
       'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')}
------------------------------
{'object': Index(['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REAL
TY',
       'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE',
       'WEEKDAY_APPR_PROCESS_START', 'ORGANIZATION_TYPE', 'FONDKAPREMONT_MODE',
       'HOUSETYPE_MODE', 'WALLSMATERIAL_MODE', 'EMERGENCYSTATE_MODE'],
      dtype='object')}
------------------------------



--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

 The Missing Data:
```

	Percent	Train Missing Count
COMMONAREA_MEDI	69.87	214865
COMMONAREA_AVG	69.87	214865
COMMONAREA_MODE	69.87	214865
NONLIVINGAPARTMENTS_MODE	69.43	213514
NONLIVINGAPARTMENTS_MEDI	69.43	213514
NONLIVINGAPARTMENTS_AVG	69.43	213514
FONDKAPREMONT_MODE	68.39	210295
LIVINGAPARTMENTS_MEDI	68.35	210199
LIVINGAPARTMENTS_MODE	68.35	210199
LIVINGAPARTMENTS_AVG	68.35	210199
FLOORSMIN_MEDI	67.85	208642
FLOORSMIN_MODE	67.85	208642
FLOORSMIN_AVG	67.85	208642
YEARS_BUILD_MEDI	66.50	204488
YEARS_BUILD_AVG	66.50	204488
YEARS_BUILD_MODE	66.50	204488
OWN_CAR_AGE	65.99	202929
LANDAREA_MODE	59.38	182590
LANDAREA_AVG	59.38	182590
LANDAREA_MEDI	59.38	182590
BASEMENTAREA_MEDI	58.52	179943
BASEMENTAREA_AVG	58.52	179943
BASEMENTAREA_MODE	58.52	179943
EXT_SOURCE_1	56.38	173378
NONLIVINGAREA_MEDI	55.18	169682
NONLIVINGAREA_AVG	55.18	169682
NONLIVINGAREA_MODE	55.18	169682
ELEVATORS_MODE	53.30	163891
ELEVATORS_AVG	53.30	163891
ELEVATORS_MEDI	53.30	163891
WALLSMATERIAL_MODE	50.84	156341
APARTMENTS_MODE	50.75	156061
APARTMENTS_AVG	50.75	156061
APARTMENTS_MEDI	50.75	156061

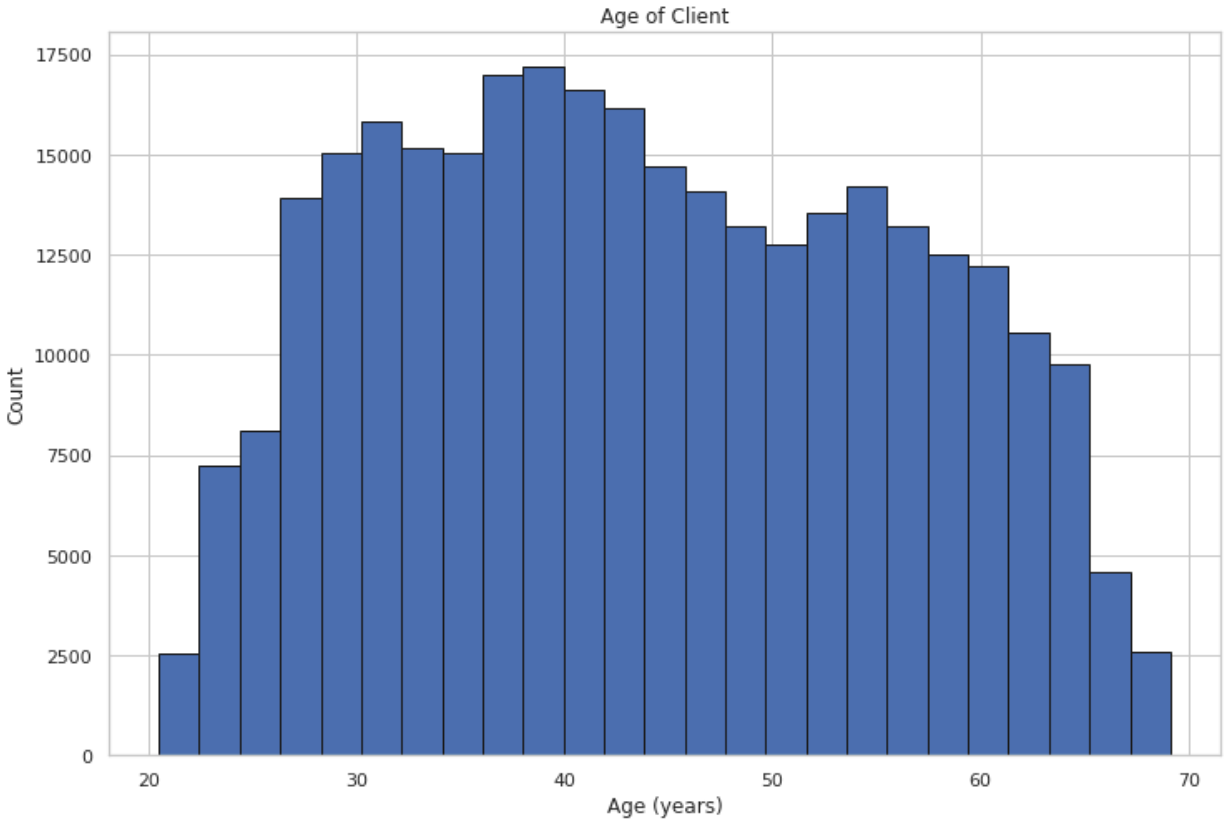
	Percent	Train Missing Count
ENTRANCES_MEDI	50.35	154828
ENTRANCES_MODE	50.35	154828
ENTRANCES_AVG	50.35	154828
LIVINGAREA_MEDI	50.19	154350
LIVINGAREA_MODE	50.19	154350
LIVINGAREA_AVG	50.19	154350
HOUSETYPE_MODE	50.18	154297
FLOORSMAX_MODE	49.76	153020
FLOORSMAX_MEDI	49.76	153020
FLOORSMAX_AVG	49.76	153020
YEARS_BEGINEXPLUATATION_MEDI	48.78	150007
YEARS_BEGINEXPLUATATION_AVG	48.78	150007
YEARS_BEGINEXPLUATATION_MODE	48.78	150007
TOTALAREA_MODE	48.27	148431
EMERGENCYSTATE_MODE	47.40	145755
OCCUPATION_TYPE	31.35	96391
EXT_SOURCE_3	19.83	60965
AMT_REQ_CREDIT_BUREAU_QRT	13.50	41519
AMT_REQ_CREDIT_BUREAU_YEAR	13.50	41519
AMT_REQ_CREDIT_BUREAU_WEEK	13.50	41519
AMT_REQ_CREDIT_BUREAU_MON	13.50	41519
AMT_REQ_CREDIT_BUREAU_DAY	13.50	41519
AMT_REQ_CREDIT_BUREAU_HOUR	13.50	41519
NAME_TYPE_SUITE	0.42	1292
OBS_30_CNT_SOCIAL_CIRCLE	0.33	1021
OBS_60_CNT_SOCIAL_CIRCLE	0.33	1021
DEF_60_CNT_SOCIAL_CIRCLE	0.33	1021
DEF_30_CNT_SOCIAL_CIRCLE	0.33	1021
EXT_SOURCE_2	0.21	660
AMT_GOODS_PRICE	0.09	278



Applicants Age

In [ ]:

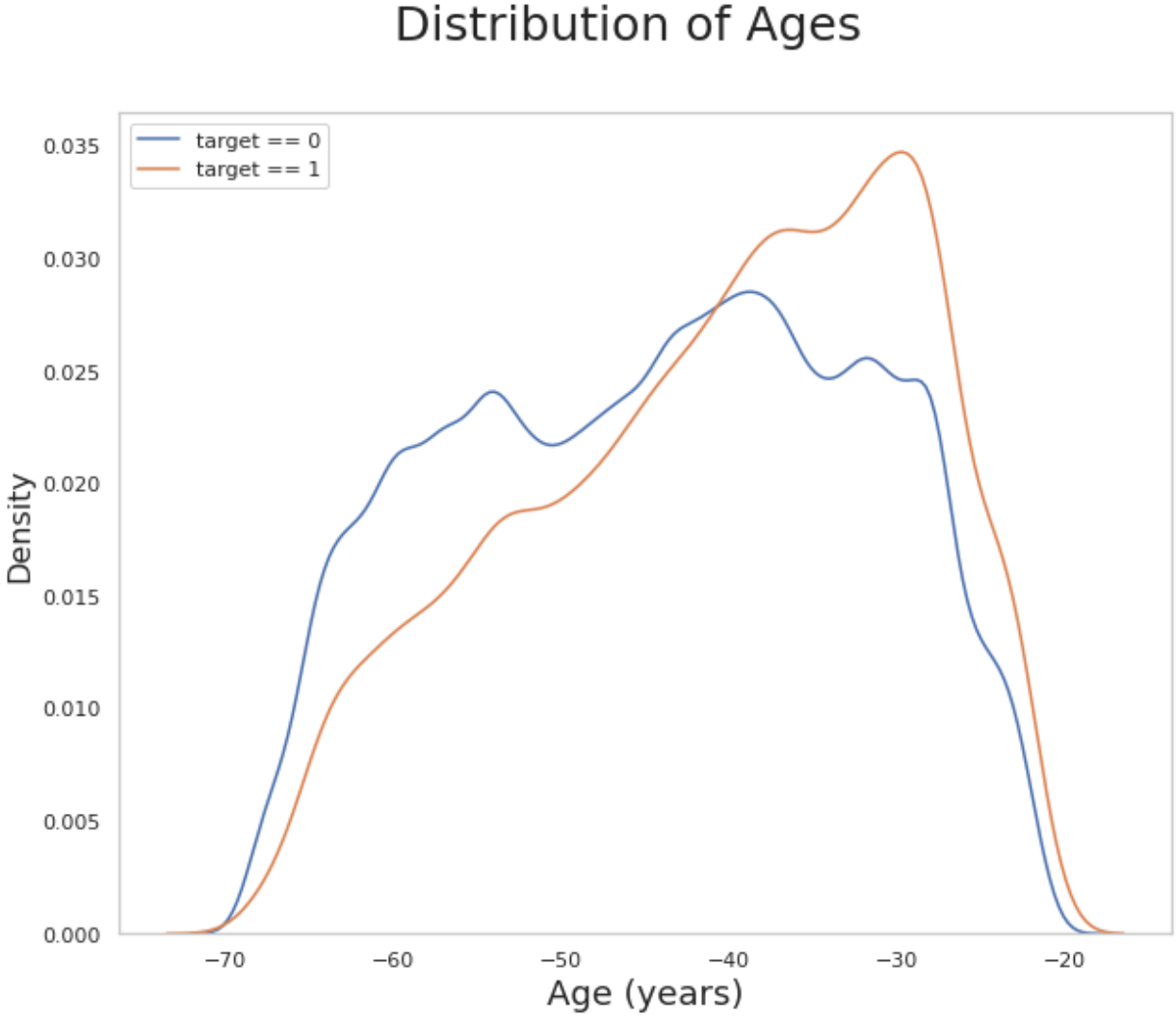
```
plt.hist(datasets["application_train"]['DAYS_BIRTH']/-365, edgecolor = 'k', bins = 25
plt.title('Age of Client'); plt.xlabel('Age (years)'); plt.ylabel('Count');
```



To visualize the effect of the age on the target, we will next make a kernel density estimation plot (KDE) colored by the value of the target.

In [ ]:

```
plt.figure(figsize = (10, 8))

# KDE plot of loans that were repaid on time
sns.kdeplot(datasets['application_train'].loc[datasets['application_train']['TARGET']

# KDE plot of loans which were not repaid on time
sns.kdeplot(datasets['application_train'].loc[datasets['application_train']['TARGET']
plt.legend(loc='upper left')
# set_style("whitegrid")
plt.grid()
# Labeling of plot
plt.xlabel('Age (years)', fontsize=18);
plt.ylabel('Density', fontsize=16);
plt.suptitle('Distribution of Ages',fontsize=25);
```



The target == 1 curve skews towards the younger end of the range. Let's look at this relationship in another way: average failure to repay loans by age bracket. To make this graph, first we cut the age category into bins of 5 years each. Then, for each bin, we calculate the average value of the target, which tells us the ratio of loans that were not repaid in each age category.

In [ ]:

```
# Age information into a separate dataframe
age_data = datasets['application_train'][['TARGET', 'DAYS_BIRTH']]
age_data['YEARS_BIRTH'] = age_data['DAYS_BIRTH'] / -365

# Bin the age data
age_data['YEARS_BINNED'] = pd.cut(age_data['YEARS_BIRTH'], bins = np.linspace(20, 70,
age_data.head(10)
```

Out[ ]:

	TARGET	DAYS_BIRTH	YEARS_BIRTH	YEARS_BINNED
0	1	-9461	25.920548	(25.0, 30.0]
1	0	-16765	45.931507	(45.0, 50.0]
2	0	-19046	52.180822	(50.0, 55.0]
3	0	-19005	52.068493	(50.0, 55.0]
4	0	-19932	54.608219	(50.0, 55.0]
5	0	-16941	46.413699	(45.0, 50.0]
6	0	-13778	37.747945	(35.0, 40.0]
7	0	-18850	51.643836	(50.0, 55.0]
8	0	-20099	55.065753	(55.0, 60.0]
9	0	-14469	39.641096	(35.0, 40.0]

In [ ]:

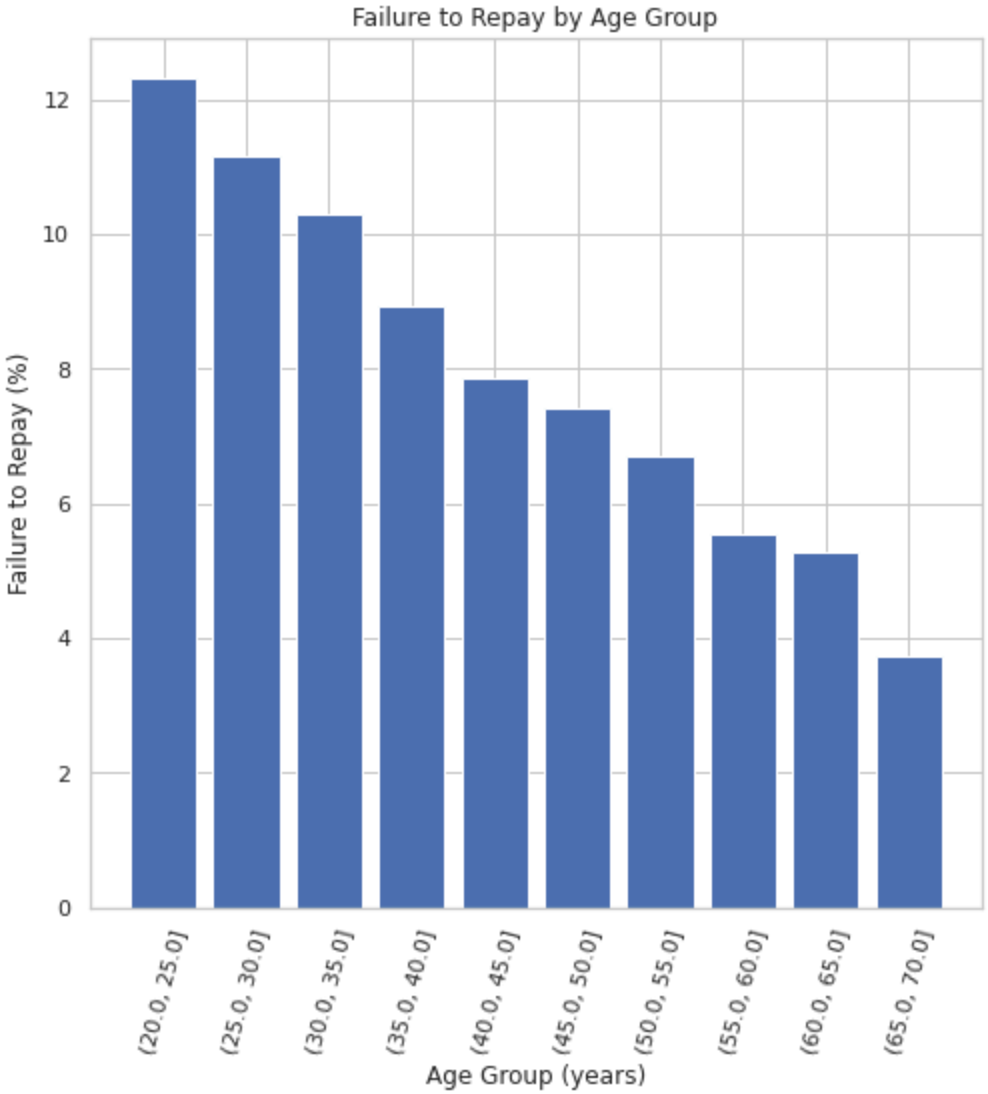
```
# Group by the bin and calculate averages
age_groups  = age_data.groupby('YEARS_BINNED').mean()
age_groups
```

Out[ ]:

YEARS_BINNED	TARGET	DAYS_BIRTH	YEARS_BIRTH
(20.0, 25.0]	0.123036	-8532.795625	23.377522
(25.0, 30.0]	0.111436	-10155.219250	27.822518
(30.0, 35.0]	0.102814	-11854.848377	32.479037
(35.0, 40.0]	0.089414	-13707.908253	37.555913
(40.0, 45.0]	0.078491	-15497.661233	42.459346
(45.0, 50.0]	0.074171	-17323.900441	47.462741
(50.0, 55.0]	0.066968	-19196.494791	52.593136
(55.0, 60.0]	0.055314	-20984.262742	57.491131
(60.0, 65.0]	0.052737	-22780.547460	62.412459
(65.0, 70.0]	0.037270	-24292.614340	66.555108

In [ ]:

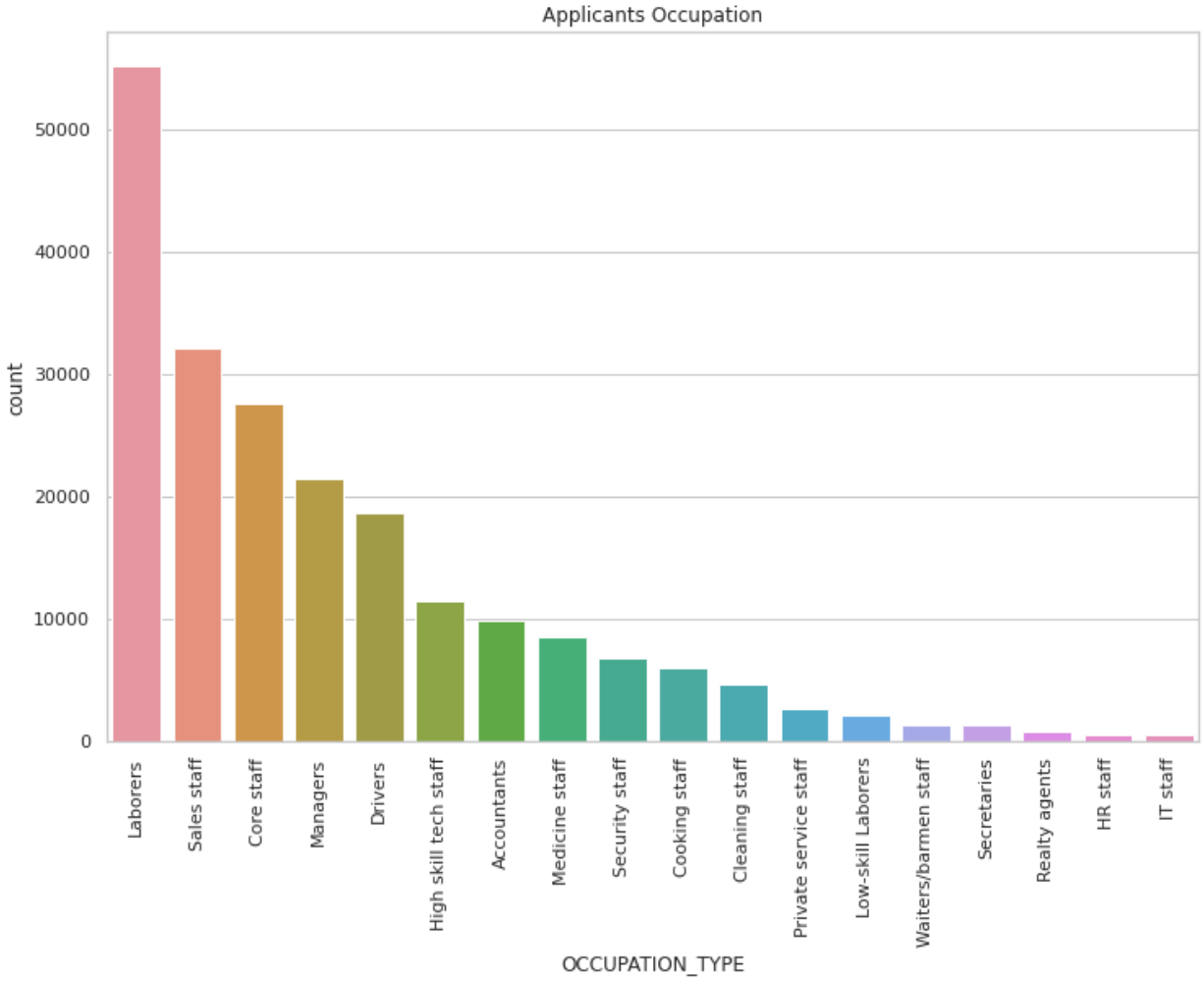
```
plt.figure(figsize = (8, 8))

# Graph the age bins and the average of the target as a bar plot
plt.bar(age_groups.index.astype(str), 100 * age_groups['TARGET'])

# Plot labeling
plt.xticks(rotation = 75); plt.xlabel('Age Group (years)'); plt.ylabel('Failure to Re
plt.title('Failure to Repay by Age Group');
```



Applicants occupations

In [ ]:

```
sns.countplot(x='OCCUPATION_TYPE', data=datasets["application_train"], order = datase
plt.title('Applicants Occupation');
plt.xticks(rotation=90);
```



Contract Type with Amount Credit and Code Gender

In [ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns                    #visualisation
import matplotlib.pyplot as plt          #visualisation
%matplotlib inline
sns.set(color_codes=True)
```

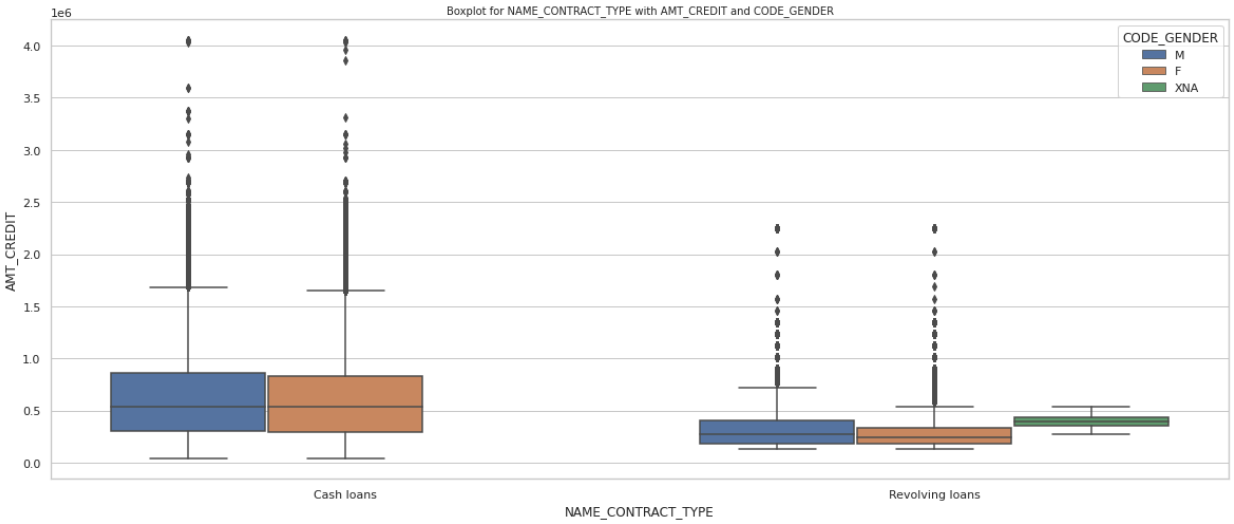
In [ ]:
```
def generic_xy_boxplot(xaxisfeature,yaxisfeature,legendcategory,data,log_scale):
  sns.boxplot(xaxisfeature, yaxisfeature, hue = legendcategory, data = data)
  plt.title('Boxplot for '+ xaxisfeature +' with ' + yaxisfeature+' and '+legendcateg
  if log_scale:
                plt.yscale('log')
                plt.ylabel(f'{yaxisfeature} (log Scale)')
                plt.tight_layout()
```

In [ ]:
```
def box_plot(plots):
  number_of_subplots = len(plots)
  plt.figure(figsize = (20,8))
  sns.set_style('whitegrid')
  for i, ele in enumerate(plots):
        plt.subplot(1, number_of_subplots, i + 1)
        plt.subplots_adjust(wspace=0.25)
        xaxisfeature=ele[0]
```

```
        yaxisfeature=ele[1]
        legendcategory=ele[2]
        data=ele[3]
        log_scale=ele[4]
        generic_xy_boxplot(xaxisfeature,yaxisfeature,legendcategory,data,log_scale)
```

In [ ]:
```
plots=[['NAME_CONTRACT_TYPE','AMT_CREDIT','CODE_GENDER',datasets['application_train']]
```

In [ ]:
```
box_plot(plots)
```



Gender does not indicate a major impact . But credit amount for cash loans are significantly high compared to revolving loans.

Summary of previous_application

In [ ]:
```
display_stats(datasets['previous_application'], 'previous_application')
```

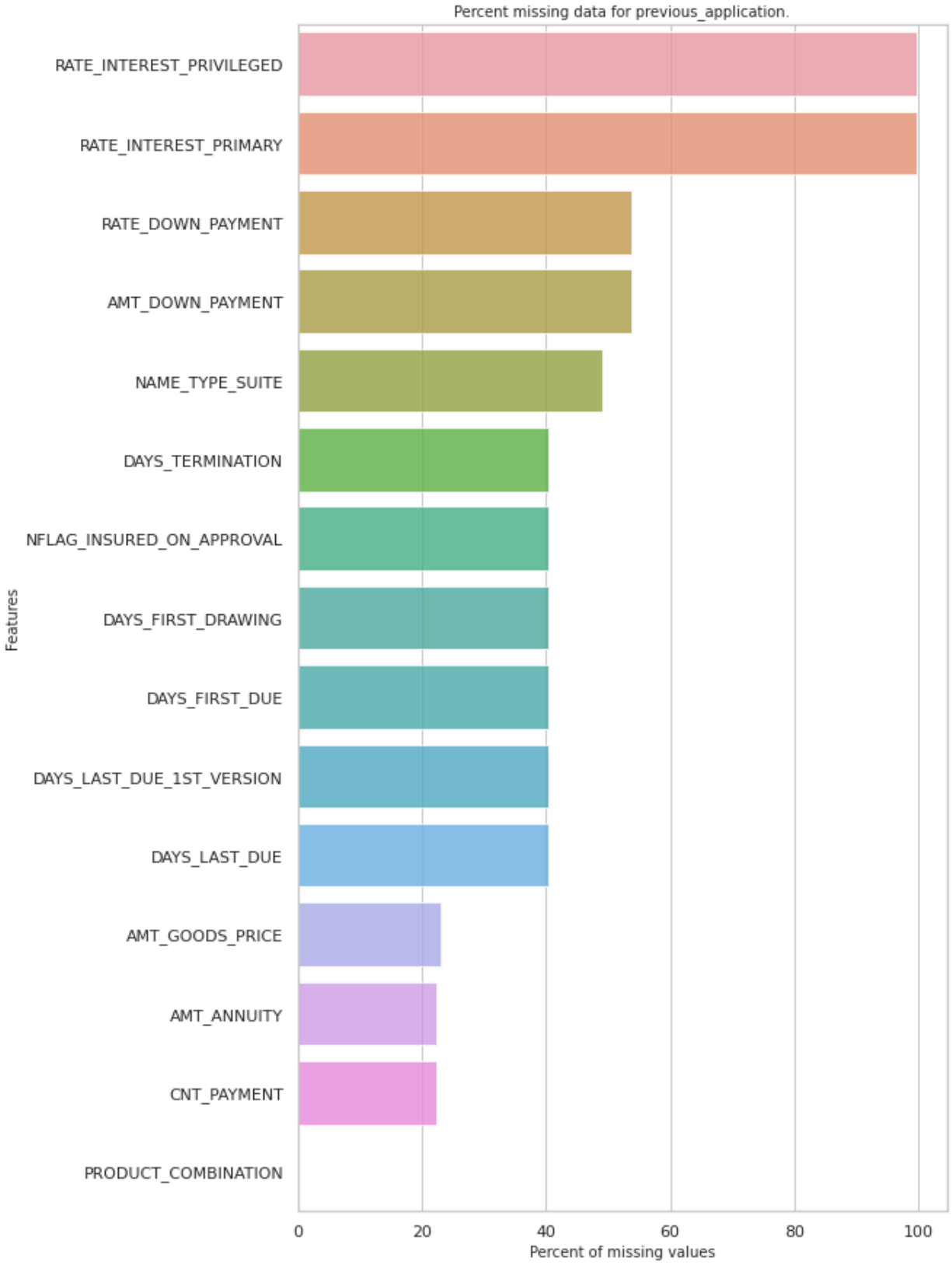
```
--------------------------------------------------------------------------------
                    previous_application
--------------------------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1670214 entries, 0 to 1670213
Data columns (total 37 columns):
 #   Column                       Non-Null Count    Dtype  
---  ------                       --------------    -----  
 0   SK_ID_PREV                   1670214 non-null  int64  
 1   SK_ID_CURR                   1670214 non-null  int64  
 2   NAME_CONTRACT_TYPE           1670214 non-null  object 
 3   AMT_ANNUITY                  1297979 non-null  float64
 4   AMT_APPLICATION              1670214 non-null  float64
 5   AMT_CREDIT                   1670213 non-null  float64
 6   AMT_DOWN_PAYMENT             774370 non-null   float64
 7   AMT_GOODS_PRICE              1284699 non-null  float64
 8   WEEKDAY_APPR_PROCESS_START   1670214 non-null  object 
 9   HOUR_APPR_PROCESS_START      1670214 non-null  int64  
 10  FLAG_LAST_APPL_PER_CONTRACT  1670214 non-null  object 
 11  NFLAG_LAST_APPL_IN_DAY       1670214 non-null  int64  
 12  RATE_DOWN_PAYMENT            774370 non-null   float64
 13  RATE_INTEREST_PRIMARY        5951 non-null     float64
 14  RATE_INTEREST_PRIVILEGED     5951 non-null     float64
 15  NAME_CASH_LOAN_PURPOSE       1670214 non-null  object 
 16  NAME_CONTRACT_STATUS         1670214 non-null  object 
 17  DAYS_DECISION                1670214 non-null  int64  
 18  NAME_PAYMENT_TYPE            1670214 non-null  object 
 19  CODE_REJECT_REASON           1670214 non-null  object 
 20  NAME_TYPE_SUITE              849809 non-null   object 
 21  NAME_CLIENT_TYPE             1670214 non-null  object 
 22  NAME_GOODS_CATEGORY          1670214 non-null  object 
 23  NAME_PORTFOLIO               1670214 non-null  object 
 24  NAME_PRODUCT_TYPE            1670214 non-null  object 
 25  CHANNEL_TYPE                 1670214 non-null  object 
 26  SELLERPLACE_AREA             1670214 non-null  int64  
 27  NAME_SELLER_INDUSTRY         1670214 non-null  object 
 28  CNT_PAYMENT                  1297984 non-null  float64
 29  NAME_YIELD_GROUP             1670214 non-null  object 
 30  PRODUCT_COMBINATION          1669868 non-null  object 
 31  DAYS_FIRST_DRAWING           997149 non-null   float64
 32  DAYS_FIRST_DUE               997149 non-null   float64
 33  DAYS_LAST_DUE_1ST_VERSION    997149 non-null   float64
 34  DAYS_LAST_DUE                997149 non-null   float64
 35  DAYS_TERMINATION             997149 non-null   float64
 36  NFLAG_INSURED_ON_APPROVAL    997149 non-null   float64
dtypes: float64(15), int64(6), object(16)
memory usage: 471.5+ MB
None
--------------------------------------------------------------------------------
Shape of the df previous_application is (1670214, 37)

--------------------------------------------------------------------------------
Statistical summary of previous_application is :
--------------------------------------------------------------------------------
Description of the df previous_application:
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	A
count	307511.00	307511.00	307511.00	3.075110e+05	307511.00	307499.00	
mean	278180.52	0.08	0.42	1.687979e+05	599026.00	27108.57	
std	102790.18	0.27	0.72	2.371231e+05	402490.78	14493.74	
min	100002.00	0.00	0.00	2.565000e+04	45000.00	1615.50	
25%	189145.50	0.00	0.00	1.125000e+05	270000.00	16524.00	
50%	278202.00	0.00	0.00	1.471500e+05	513531.00	24903.00	
75%	367142.50	0.00	1.00	2.025000e+05	808650.00	34596.00	
max	456255.00	1.00	19.00	1.170000e+08	4050000.00	258025.50	

In [ ]:
```
display_feature_info(datasets['previous_application'], 'previous_application')
```

```
Description of the df continued for previous_application:

------------------------------------------------------------------------------
Data type value counts:
 object     16
float64    15
int64       6
dtype: int64

Return number of unique elements in the object.

NAME_CONTRACT_TYPE              4
WEEKDAY_APPR_PROCESS_START      7
FLAG_LAST_APPL_PER_CONTRACT     2
NAME_CASH_LOAN_PURPOSE         25
NAME_CONTRACT_STATUS            4
NAME_PAYMENT_TYPE               4
CODE_REJECT_REASON              9
NAME_TYPE_SUITE                 7
NAME_CLIENT_TYPE                4
NAME_GOODS_CATEGORY            28
NAME_PORTFOLIO                  5
NAME_PRODUCT_TYPE               3
CHANNEL_TYPE                    8
NAME_SELLER_INDUSTRY           11
NAME_YIELD_GROUP                5
PRODUCT_COMBINATION            17
dtype: int64
------------------------------------------------------------------------------
Categorical and Numerical(int + float) features  of previous_application.
------------------------------------------------------------------------------

{'int64': Index(['SK_ID_PREV', 'SK_ID_CURR', 'HOUR_APPR_PROCESS_START',
       'NFLAG_LAST_APPL_IN_DAY', 'DAYS_DECISION', 'SELLERPLACE_AREA'],
      dtype='object')}
------------------------------
{'float64': Index(['AMT_ANNUITY', 'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_DOWN_PAYMEN
T',
       'AMT_GOODS_PRICE', 'RATE_DOWN_PAYMENT', 'RATE_INTEREST_PRIMARY',
       'RATE_INTEREST_PRIVILEGED', 'CNT_PAYMENT', 'DAYS_FIRST_DRAWING',
       'DAYS_FIRST_DUE', 'DAYS_LAST_DUE_1ST_VERSION', 'DAYS_LAST_DUE',
       'DAYS_TERMINATION', 'NFLAG_INSURED_ON_APPROVAL'],
      dtype='object')}
------------------------------
{'object': Index(['NAME_CONTRACT_TYPE', 'WEEKDAY_APPR_PROCESS_START',
       'FLAG_LAST_APPL_PER_CONTRACT', 'NAME_CASH_LOAN_PURPOSE',
       'NAME_CONTRACT_STATUS', 'NAME_PAYMENT_TYPE', 'CODE_REJECT_REASON',
       'NAME_TYPE_SUITE', 'NAME_CLIENT_TYPE', 'NAME_GOODS_CATEGORY',
       'NAME_PORTFOLIO', 'NAME_PRODUCT_TYPE', 'CHANNEL_TYPE',
       'NAME_SELLER_INDUSTRY', 'NAME_YIELD_GROUP', 'PRODUCT_COMBINATION'],
      dtype='object')}
------------------------------



------------------------------------------------------------------------------
------------------------------------------------------------------------------

 The Missing Data:
```

	Percent	Train Missing Count
RATE_INTEREST_PRIVILEGED	99.64	1664263
RATE_INTEREST_PRIMARY	99.64	1664263
RATE_DOWN_PAYMENT	53.64	895844
AMT_DOWN_PAYMENT	53.64	895844
NAME_TYPE_SUITE	49.12	820405
DAYS_TERMINATION	40.30	673065
NFLAG_INSURED_ON_APPROVAL	40.30	673065
DAYS_FIRST_DRAWING	40.30	673065
DAYS_FIRST_DUE	40.30	673065
DAYS_LAST_DUE_1ST_VERSION	40.30	673065
DAYS_LAST_DUE	40.30	673065
AMT_GOODS_PRICE	23.08	385515
AMT_ANNUITY	22.29	372235
CNT_PAYMENT	22.29	372230
PRODUCT_COMBINATION	0.02	346



Summary of bureau

In [ ]:
```
display_stats(datasets['bureau'], 'bureau')
```

```
--------------------------------------------------------------------------------
                    bureau
--------------------------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1716428 entries, 0 to 1716427
Data columns (total 17 columns):
 #   Column                  Non-Null Count    Dtype
---  ------                  --------------    -----
 0   SK_ID_CURR              1716428 non-null  int64
 1   SK_ID_BUREAU            1716428 non-null  int64
 2   CREDIT_ACTIVE           1716428 non-null  object
 3   CREDIT_CURRENCY         1716428 non-null  object
 4   DAYS_CREDIT             1716428 non-null  int64
 5   CREDIT_DAY_OVERDUE      1716428 non-null  int64
 6   DAYS_CREDIT_ENDDATE     1610875 non-null  float64
 7   DAYS_ENDDATE_FACT       1082775 non-null  float64
 8   AMT_CREDIT_MAX_OVERDUE  591940 non-null   float64
 9   CNT_CREDIT_PROLONG      1716428 non-null  int64
 10  AMT_CREDIT_SUM          1716415 non-null  float64
 11  AMT_CREDIT_SUM_DEBT     1458759 non-null  float64
 12  AMT_CREDIT_SUM_LIMIT    1124648 non-null  float64
 13  AMT_CREDIT_SUM_OVERDUE  1716428 non-null  float64
 14  CREDIT_TYPE             1716428 non-null  object
 15  DAYS_CREDIT_UPDATE      1716428 non-null  int64
 16  AMT_ANNUITY             489637 non-null   float64
dtypes: float64(8), int64(6), object(3)
memory usage: 222.6+ MB
None
--------------------------------------------------------------------------------
Shape of the df bureau is (1716428, 17)

--------------------------------------------------------------------------------
Statistical summary of bureau is :
--------------------------------------------------------------------------------
Description of the df bureau:
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	A
count	307511.00	307511.00	307511.00	3.075110e+05	307511.00	307499.00	
mean	278180.52	0.08	0.42	1.687979e+05	599026.00	27108.57	
std	102790.18	0.27	0.72	2.371231e+05	402490.78	14493.74	
min	100002.00	0.00	0.00	2.565000e+04	45000.00	1615.50	
25%	189145.50	0.00	0.00	1.125000e+05	270000.00	16524.00	
50%	278202.00	0.00	0.00	1.471500e+05	513531.00	24903.00	
75%	367142.50	0.00	1.00	2.025000e+05	808650.00	34596.00	
max	456255.00	1.00	19.00	1.170000e+08	4050000.00	258025.50	

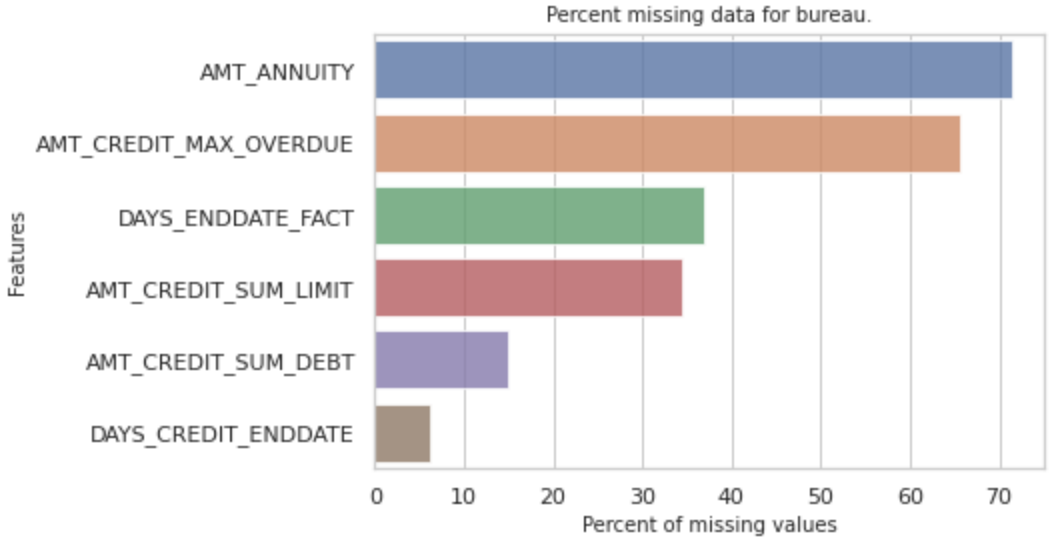
```
None
```

In [ ]:
```
display_feature_info(datasets['bureau'], 'bureau')
```

```
Description of the df continued for bureau:

------------------------------------------------------------------------------
Data type value counts:
 float64    8
int64      6
object     3
dtype: int64

Return number of unique elements in the object.

CREDIT_ACTIVE       4
CREDIT_CURRENCY     4
CREDIT_TYPE        15
dtype: int64
------------------------------------------------------------------------------
Categorical and Numerical(int + float) features  of bureau.
------------------------------------------------------------------------------

{'int64': Index(['SK_ID_CURR', 'SK_ID_BUREAU', 'DAYS_CREDIT', 'CREDIT_DAY_OVERDUE',
       'CNT_CREDIT_PROLONG', 'DAYS_CREDIT_UPDATE'],
      dtype='object')}
------------------------------
{'float64': Index(['DAYS_CREDIT_ENDDATE', 'DAYS_ENDDATE_FACT', 'AMT_CREDIT_MAX_OVERDU
E',
       'AMT_CREDIT_SUM', 'AMT_CREDIT_SUM_DEBT', 'AMT_CREDIT_SUM_LIMIT',
       'AMT_CREDIT_SUM_OVERDUE', 'AMT_ANNUITY'],
      dtype='object')}
------------------------------
{'object': Index(['CREDIT_ACTIVE', 'CREDIT_CURRENCY', 'CREDIT_TYPE'], dtype='objec
t')}
------------------------------



------------------------------------------------------------------------------
------------------------------------------------------------------------------

 The Missing Data:
```

	Percent	Train Missing Count
AMT_ANNUITY	71.47	1226791
AMT_CREDIT_MAX_OVERDUE	65.51	1124488
DAYS_ENDDATE_FACT	36.92	633653
AMT_CREDIT_SUM_LIMIT	34.48	591780
AMT_CREDIT_SUM_DEBT	15.01	257669
DAYS_CREDIT_ENDDATE	6.15	105553

```
------------------------------------------------------------------------------
```



Summary of bureau_balance

In [ ]:
```
display_stats(datasets['bureau_balance'], 'bureau_balance')
```

```
--------------------------------------------------------------------------------
                    bureau_balance
--------------------------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27299925 entries, 0 to 27299924
Data columns (total 3 columns):
 #   Column          Non-Null Count     Dtype
---  ------          --------------     -----
 0   SK_ID_BUREAU    27299925 non-null  int64
 1   MONTHS_BALANCE  27299925 non-null  int64
 2   STATUS          27299925 non-null  object
dtypes: int64(2), object(1)
memory usage: 624.8+ MB
None
--------------------------------------------------------------------------------
Shape of the df bureau_balance is (27299925, 3)

--------------------------------------------------------------------------------
Statistical summary of bureau_balance is :
--------------------------------------------------------------------------------
Description of the df bureau_balance:
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	A
count	307511.00	307511.00	307511.00	3.075110e+05	307511.00	307499.00	
mean	278180.52	0.08	0.42	1.687979e+05	599026.00	27108.57	
std	102790.18	0.27	0.72	2.371231e+05	402490.78	14493.74	
min	100002.00	0.00	0.00	2.565000e+04	45000.00	1615.50	
25%	189145.50	0.00	0.00	1.125000e+05	270000.00	16524.00	
50%	278202.00	0.00	0.00	1.471500e+05	513531.00	24903.00	
75%	367142.50	0.00	1.00	2.025000e+05	808650.00	34596.00	
max	456255.00	1.00	19.00	1.170000e+08	4050000.00	258025.50	

In [ ]:
```
display_feature_info(datasets['bureau_balance'], 'bureau_balance')
```

```
Description of the df continued for bureau_balance:

--------------------------------------------------------------------------------
Data type value counts:
 int64     2
object    1
dtype: int64

Return number of unique elements in the object.

STATUS    8
dtype: int64
--------------------------------------------------------------------------------
Categorical and Numerical(int + float) features  of bureau_balance.
--------------------------------------------------------------------------------

{'int64': Index(['SK_ID_BUREAU', 'MONTHS_BALANCE'], dtype='object')}
------------------------------
{'object': Index(['STATUS'], dtype='object')}
------------------------------



--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

 The Missing Data:

No missing Data
```

Summary of credit_card_balance

In [ ]:
```
display_stats(datasets['credit_card_balance'], 'credit_card_balance')
```

```
--------------------------------------------------------------------------------
                    credit_card_balance
--------------------------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3840312 entries, 0 to 3840311
Data columns (total 23 columns):
 #   Column                       Non-Null Count    Dtype
---  ------                       --------------    -----
 0   SK_ID_PREV                   3840312 non-null  int64
 1   SK_ID_CURR                   3840312 non-null  int64
 2   MONTHS_BALANCE               3840312 non-null  int64
 3   AMT_BALANCE                  3840312 non-null  float64
 4   AMT_CREDIT_LIMIT_ACTUAL      3840312 non-null  int64
 5   AMT_DRAWINGS_ATM_CURRENT     3090496 non-null  float64
 6   AMT_DRAWINGS_CURRENT         3840312 non-null  float64
 7   AMT_DRAWINGS_OTHER_CURRENT   3090496 non-null  float64
 8   AMT_DRAWINGS_POS_CURRENT     3090496 non-null  float64
 9   AMT_INST_MIN_REGULARITY      3535076 non-null  float64
 10  AMT_PAYMENT_CURRENT          3072324 non-null  float64
 11  AMT_PAYMENT_TOTAL_CURRENT    3840312 non-null  float64
 12  AMT_RECEIVABLE_PRINCIPAL     3840312 non-null  float64
 13  AMT_RECIVABLE                3840312 non-null  float64
 14  AMT_TOTAL_RECEIVABLE         3840312 non-null  float64
 15  CNT_DRAWINGS_ATM_CURRENT     3090496 non-null  float64
 16  CNT_DRAWINGS_CURRENT         3840312 non-null  int64
 17  CNT_DRAWINGS_OTHER_CURRENT   3090496 non-null  float64
 18  CNT_DRAWINGS_POS_CURRENT     3090496 non-null  float64
 19  CNT_INSTALMENT_MATURE_CUM    3535076 non-null  float64
 20  NAME_CONTRACT_STATUS         3840312 non-null  object
 21  SK_DPD                       3840312 non-null  int64
 22  SK_DPD_DEF                   3840312 non-null  int64
dtypes: float64(15), int64(7), object(1)
memory usage: 673.9+ MB
None
--------------------------------------------------------------------------------
Shape of the df credit_card_balance is (3840312, 23)

--------------------------------------------------------------------------------
Statistical summary of credit_card_balance is :
--------------------------------------------------------------------------------
Description of the df credit_card_balance:
```

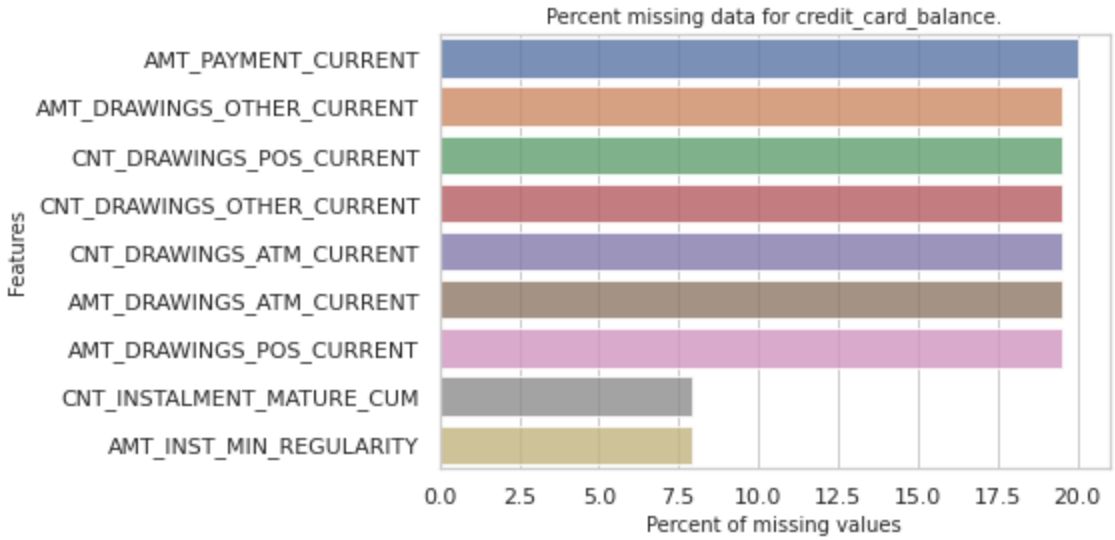
	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	A
count	307511.00	307511.00	307511.00	3.075110e+05	307511.00	307499.00	
mean	278180.52	0.08	0.42	1.687979e+05	599026.00	27108.57	
std	102790.18	0.27	0.72	2.371231e+05	402490.78	14493.74	
min	100002.00	0.00	0.00	2.565000e+04	45000.00	1615.50	
25%	189145.50	0.00	0.00	1.125000e+05	270000.00	16524.00	
50%	278202.00	0.00	0.00	1.471500e+05	513531.00	24903.00	
75%	367142.50	0.00	1.00	2.025000e+05	808650.00	34596.00	
max	456255.00	1.00	19.00	1.170000e+08	4050000.00	258025.50	

```
None
```

In [ ]:
```
display_feature_info(datasets['credit_card_balance'], 'credit_card_balance')
```

```
Description of the df continued for credit_card_balance:

------------------------------------------------------------------------------
Data type value counts:
 float64    15
int64       7
object      1
dtype: int64

Return number of unique elements in the object.

NAME_CONTRACT_STATUS    7
dtype: int64
------------------------------------------------------------------------------
Categorical and Numerical(int + float) features  of credit_card_balance.
------------------------------------------------------------------------------

{'int64': Index(['SK_ID_PREV', 'SK_ID_CURR', 'MONTHS_BALANCE', 'AMT_CREDIT_LIMIT_ACTU
AL',
       'CNT_DRAWINGS_CURRENT', 'SK_DPD', 'SK_DPD_DEF'],
      dtype='object')}
------------------------------
{'float64': Index(['AMT_BALANCE', 'AMT_DRAWINGS_ATM_CURRENT', 'AMT_DRAWINGS_CURRENT',
       'AMT_DRAWINGS_OTHER_CURRENT', 'AMT_DRAWINGS_POS_CURRENT',
       'AMT_INST_MIN_REGULARITY', 'AMT_PAYMENT_CURRENT',
       'AMT_PAYMENT_TOTAL_CURRENT', 'AMT_RECEIVABLE_PRINCIPAL',
       'AMT_RECIVABLE', 'AMT_TOTAL_RECEIVABLE', 'CNT_DRAWINGS_ATM_CURRENT',
       'CNT_DRAWINGS_OTHER_CURRENT', 'CNT_DRAWINGS_POS_CURRENT',
       'CNT_INSTALMENT_MATURE_CUM'],
      dtype='object')}
------------------------------
{'object': Index(['NAME_CONTRACT_STATUS'], dtype='object')}
------------------------------



------------------------------------------------------------------------------
------------------------------------------------------------------------------

 The Missing Data:
```

	Percent	Train Missing Count
AMT_PAYMENT_CURRENT	20.00	767988
AMT_DRAWINGS_OTHER_CURRENT	19.52	749816
CNT_DRAWINGS_POS_CURRENT	19.52	749816
CNT_DRAWINGS_OTHER_CURRENT	19.52	749816
CNT_DRAWINGS_ATM_CURRENT	19.52	749816
AMT_DRAWINGS_ATM_CURRENT	19.52	749816
AMT_DRAWINGS_POS_CURRENT	19.52	749816
CNT_INSTALMENT_MATURE_CUM	7.95	305236
AMT_INST_MIN_REGULARITY	7.95	305236



Summary of installments_payments

In [ ]:
```
display_stats(datasets['installments_payments'], 'installments_payments')
```

```
--------------------------------------------------------------------------------
                    installments_payments
--------------------------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7217242 entries, 0 to 7217241
Data columns (total 8 columns):
 #   Column                  Non-Null Count    Dtype  
---  ------                  --------------    -----  
 0   SK_ID_PREV              7217242 non-null  int64  
 1   SK_ID_CURR              7217242 non-null  int64  
 2   NUM_INSTALMENT_VERSION  7217242 non-null  float64
 3   NUM_INSTALMENT_NUMBER   7217242 non-null  int64  
 4   DAYS_INSTALMENT         7217242 non-null  float64
 5   DAYS_ENTRY_PAYMENT      7216482 non-null  float64
 6   AMT_INSTALMENT          7217242 non-null  float64
 7   AMT_PAYMENT             7216482 non-null  float64
dtypes: float64(5), int64(3)
memory usage: 440.5 MB
None
--------------------------------------------------------------------------------
Shape of the df installments_payments is (7217242, 8)

--------------------------------------------------------------------------------
Statistical summary of installments_payments is :
--------------------------------------------------------------------------------
Description of the df installments_payments:
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	A
count	307511.00	307511.00	307511.00	3.075110e+05	307511.00	307499.00	
mean	278180.52	0.08	0.42	1.687979e+05	599026.00	27108.57	
std	102790.18	0.27	0.72	2.371231e+05	402490.78	14493.74	
min	100002.00	0.00	0.00	2.565000e+04	45000.00	1615.50	
25%	189145.50	0.00	0.00	1.125000e+05	270000.00	16524.00	
50%	278202.00	0.00	0.00	1.471500e+05	513531.00	24903.00	
75%	367142.50	0.00	1.00	2.025000e+05	808650.00	34596.00	
max	456255.00	1.00	19.00	1.170000e+08	4050000.00	258025.50	

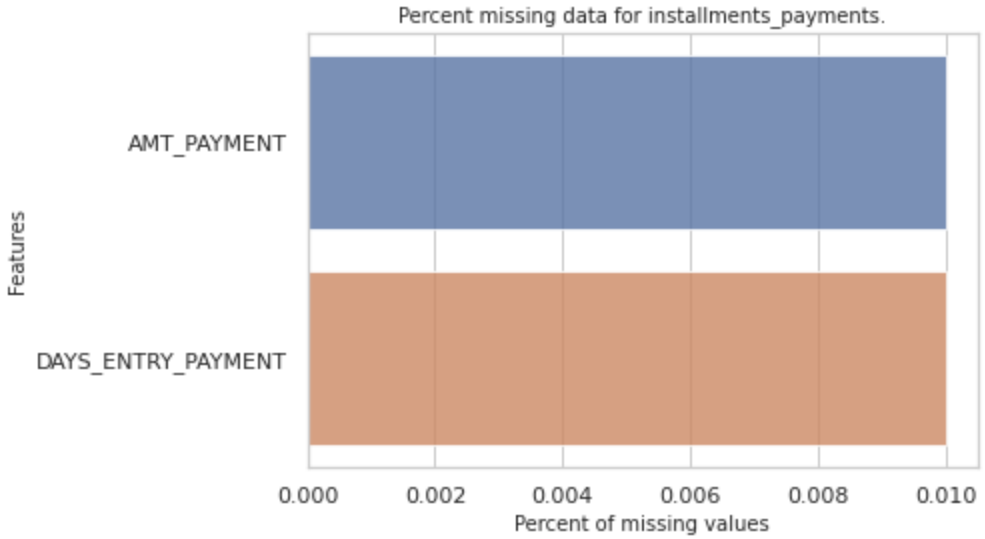
None

In [ ]:
```
display_feature_info(datasets['installments_payments'], 'installments_payments')
```

```
Description of the df continued for installments_payments:

------------------------------------------------------------------------------
Data type value counts:
 float64    5
int64      3
dtype: int64

Return number of unique elements in the object.

Series([], dtype: float64)
------------------------------------------------------------------------------
Categorical and Numerical(int + float) features  of installments_payments.
------------------------------------------------------------------------------

{'int64': Index(['SK_ID_PREV', 'SK_ID_CURR', 'NUM_INSTALMENT_NUMBER'], dtype='objec
t')}
------------------------------
{'float64': Index(['NUM_INSTALMENT_VERSION', 'DAYS_INSTALMENT', 'DAYS_ENTRY_PAYMENT',
       'AMT_INSTALMENT', 'AMT_PAYMENT'],
      dtype='object')}
------------------------------



------------------------------------------------------------------------------
------------------------------------------------------------------------------

 The Missing Data:
```

	Percent	Train Missing Count
AMT_PAYMENT	0.01	760
DAYS_ENTRY_PAYMENT	0.01	760

```
------------------------------------------------------------------------------
```



Summary of POS_CASH_balance

In [ ]:

```
display_stats(datasets['POS_CASH_balance'], 'POS_CASH_balance')
```

```
--------------------------------------------------------------------------------
                    POS_CASH_balance
--------------------------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10001358 entries, 0 to 10001357
Data columns (total 8 columns):
 #   Column                 Non-Null Count     Dtype
---  ------                 --------------     -----
 0   SK_ID_PREV             10001358 non-null  int64
 1   SK_ID_CURR             10001358 non-null  int64
 2   MONTHS_BALANCE         10001358 non-null  int64
 3   CNT_INSTALMENT         9975287 non-null   float64
 4   CNT_INSTALMENT_FUTURE  9975271 non-null   float64
 5   NAME_CONTRACT_STATUS   10001358 non-null  object
 6   SK_DPD                 10001358 non-null  int64
 7   SK_DPD_DEF             10001358 non-null  int64
dtypes: float64(2), int64(5), object(1)
memory usage: 610.4+ MB
None
--------------------------------------------------------------------------------
Shape of the df POS_CASH_balance is (10001358, 8)

--------------------------------------------------------------------------------
Statistical summary of POS_CASH_balance is :
--------------------------------------------------------------------------------
Description of the df POS_CASH_balance:
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	A
count	307511.00	307511.00	307511.00	3.075110e+05	307511.00	307499.00	
mean	278180.52	0.08	0.42	1.687979e+05	599026.00	27108.57	
std	102790.18	0.27	0.72	2.371231e+05	402490.78	14493.74	
min	100002.00	0.00	0.00	2.565000e+04	45000.00	1615.50	
25%	189145.50	0.00	0.00	1.125000e+05	270000.00	16524.00	
50%	278202.00	0.00	0.00	1.471500e+05	513531.00	24903.00	
75%	367142.50	0.00	1.00	2.025000e+05	808650.00	34596.00	
max	456255.00	1.00	19.00	1.170000e+08	4050000.00	258025.50	

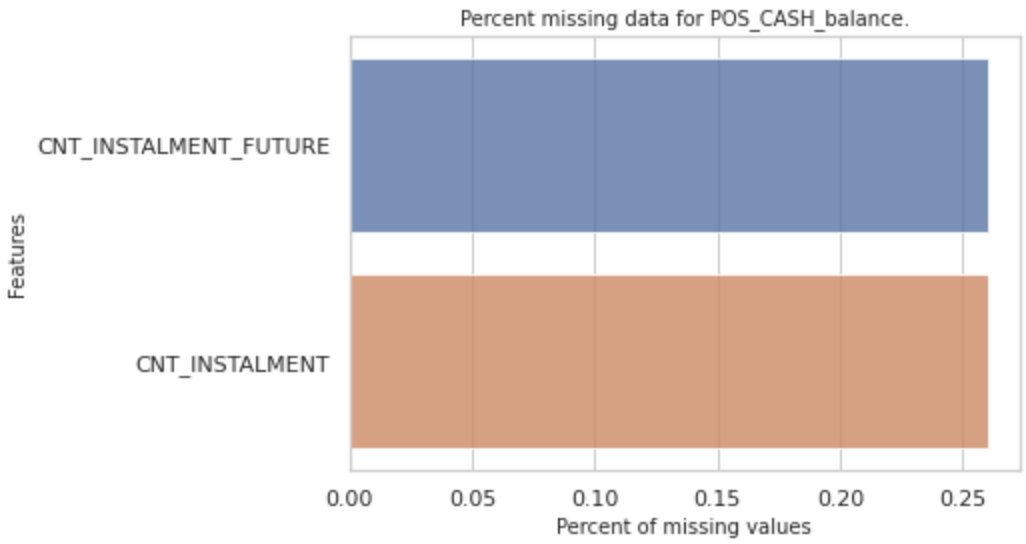
None

In [ ]:
```
display_feature_info(datasets['POS_CASH_balance'], 'POS_CASH_balance')
```

```
Description of the df continued for POS_CASH_balance:

--------------------------------------------------------------------------------
Data type value counts:
 int64      5
float64    2
object     1
dtype: int64

Return number of unique elements in the object.

NAME_CONTRACT_STATUS    9
dtype: int64
--------------------------------------------------------------------------------
Categorical and Numerical(int + float) features  of POS_CASH_balance.
--------------------------------------------------------------------------------

{'int64': Index(['SK_ID_PREV', 'SK_ID_CURR', 'MONTHS_BALANCE', 'SK_DPD', 'SK_DPD_DE
F'], dtype='object')}
------------------------------
{'float64': Index(['CNT_INSTALMENT', 'CNT_INSTALMENT_FUTURE'], dtype='object')}
------------------------------
{'object': Index(['NAME_CONTRACT_STATUS'], dtype='object')}
------------------------------



--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

 The Missing Data:
```

	Percent	Train Missing Count
CNT_INSTALMENT_FUTURE	0.26	26087
CNT_INSTALMENT	0.26	26071

```
--------------------------------------------------------------------------------
```



Summary of application_test

In [ ]:
```
display_stats(datasets['application_test'], 'application_test')
```

```
--------------------------------------------------------------------------------
                    application_test
--------------------------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48744 entries, 0 to 48743
Data columns (total 121 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   SK_ID_CURR                     48744 non-null  int64
 1   NAME_CONTRACT_TYPE             48744 non-null  object
 2   CODE_GENDER                    48744 non-null  object
 3   FLAG_OWN_CAR                   48744 non-null  object
 4   FLAG_OWN_REALTY                48744 non-null  object
 5   CNT_CHILDREN                   48744 non-null  int64
 6   AMT_INCOME_TOTAL               48744 non-null  float64
 7   AMT_CREDIT                     48744 non-null  float64
 8   AMT_ANNUITY                    48720 non-null  float64
 9   AMT_GOODS_PRICE                48744 non-null  float64
 10  NAME_TYPE_SUITE                47833 non-null  object
 11  NAME_INCOME_TYPE               48744 non-null  object
 12  NAME_EDUCATION_TYPE            48744 non-null  object
 13  NAME_FAMILY_STATUS             48744 non-null  object
 14  NAME_HOUSING_TYPE              48744 non-null  object
 15  REGION_POPULATION_RELATIVE     48744 non-null  float64
 16  DAYS_BIRTH                     48744 non-null  int64
 17  DAYS_EMPLOYED                  48744 non-null  int64
 18  DAYS_REGISTRATION              48744 non-null  float64
 19  DAYS_ID_PUBLISH                48744 non-null  int64
 20  OWN_CAR_AGE                    16432 non-null  float64
 21  FLAG_MOBIL                     48744 non-null  int64
 22  FLAG_EMP_PHONE                 48744 non-null  int64
 23  FLAG_WORK_PHONE                48744 non-null  int64
 24  FLAG_CONT_MOBILE               48744 non-null  int64
 25  FLAG_PHONE                     48744 non-null  int64
 26  FLAG_EMAIL                     48744 non-null  int64
 27  OCCUPATION_TYPE                33139 non-null  object
 28  CNT_FAM_MEMBERS                48744 non-null  float64
 29  REGION_RATING_CLIENT           48744 non-null  int64
 30  REGION_RATING_CLIENT_W_CITY    48744 non-null  int64
 31  WEEKDAY_APPR_PROCESS_START     48744 non-null  object
 32  HOUR_APPR_PROCESS_START        48744 non-null  int64
 33  REG_REGION_NOT_LIVE_REGION     48744 non-null  int64
 34  REG_REGION_NOT_WORK_REGION     48744 non-null  int64
 35  LIVE_REGION_NOT_WORK_REGION    48744 non-null  int64
 36  REG_CITY_NOT_LIVE_CITY         48744 non-null  int64
 37  REG_CITY_NOT_WORK_CITY         48744 non-null  int64
 38  LIVE_CITY_NOT_WORK_CITY        48744 non-null  int64
 39  ORGANIZATION_TYPE              48744 non-null  object
 40  EXT_SOURCE_1                   28212 non-null  float64
 41  EXT_SOURCE_2                   48736 non-null  float64
 42  EXT_SOURCE_3                   40076 non-null  float64
 43  APARTMENTS_AVG                 24857 non-null  float64
 44  BASEMENTAREA_AVG               21103 non-null  float64
 45  YEARS_BEGINEXPLUATATION_AVG    25888 non-null  float64
 46  YEARS_BUILD_AVG                16926 non-null  float64
 47  COMMONAREA_AVG                 15249 non-null  float64
 48  ELEVATORS_AVG                  23555 non-null  float64
 49  ENTRANCES_AVG                  25165 non-null  float64
 50  FLOORSMAX_AVG                  25423 non-null  float64
 51  FLOORSMIN_AVG                  16278 non-null  float64
```

```
52  LANDAREA_AVG                  20490 non-null  float64
53  LIVINGAPARTMENTS_AVG          15964 non-null  float64
54  LIVINGAREA_AVG                25192 non-null  float64
55  NONLIVINGAPARTMENTS_AVG       15397 non-null  float64
56  NONLIVINGAREA_AVG             22660 non-null  float64
57  APARTMENTS_MODE               24857 non-null  float64
58  BASEMENTAREA_MODE             21103 non-null  float64
59  YEARS_BEGINEXPLUATATION_MODE  25888 non-null  float64
60  YEARS_BUILD_MODE              16926 non-null  float64
61  COMMONAREA_MODE               15249 non-null  float64
62  ELEVATORS_MODE                23555 non-null  float64
63  ENTRANCES_MODE                25165 non-null  float64
64  FLOORSMAX_MODE                25423 non-null  float64
65  FLOORSMIN_MODE                16278 non-null  float64
66  LANDAREA_MODE                 20490 non-null  float64
67  LIVINGAPARTMENTS_MODE         15964 non-null  float64
68  LIVINGAREA_MODE               25192 non-null  float64
69  NONLIVINGAPARTMENTS_MODE      15397 non-null  float64
70  NONLIVINGAREA_MODE            22660 non-null  float64
71  APARTMENTS_MEDI               24857 non-null  float64
72  BASEMENTAREA_MEDI             21103 non-null  float64
73  YEARS_BEGINEXPLUATATION_MEDI  25888 non-null  float64
74  YEARS_BUILD_MEDI              16926 non-null  float64
75  COMMONAREA_MEDI               15249 non-null  float64
76  ELEVATORS_MEDI                23555 non-null  float64
77  ENTRANCES_MEDI                25165 non-null  float64
78  FLOORSMAX_MEDI                25423 non-null  float64
79  FLOORSMIN_MEDI                16278 non-null  float64
80  LANDAREA_MEDI                 20490 non-null  float64
81  LIVINGAPARTMENTS_MEDI         15964 non-null  float64
82  LIVINGAREA_MEDI               25192 non-null  float64
83  NONLIVINGAPARTMENTS_MEDI      15397 non-null  float64
84  NONLIVINGAREA_MEDI            22660 non-null  float64
85  FONDKAPREMONT_MODE            15947 non-null  object
86  HOUSETYPE_MODE                25125 non-null  object
87  TOTALAREA_MODE                26120 non-null  float64
88  WALLSMATERIAL_MODE            24851 non-null  object
89  EMERGENCYSTATE_MODE           26535 non-null  object
90  OBS_30_CNT_SOCIAL_CIRCLE      48715 non-null  float64
91  DEF_30_CNT_SOCIAL_CIRCLE      48715 non-null  float64
92  OBS_60_CNT_SOCIAL_CIRCLE      48715 non-null  float64
93  DEF_60_CNT_SOCIAL_CIRCLE      48715 non-null  float64
94  DAYS_LAST_PHONE_CHANGE        48744 non-null  float64
95  FLAG_DOCUMENT_2               48744 non-null  int64
96  FLAG_DOCUMENT_3               48744 non-null  int64
97  FLAG_DOCUMENT_4               48744 non-null  int64
98  FLAG_DOCUMENT_5               48744 non-null  int64
99  FLAG_DOCUMENT_6               48744 non-null  int64
100 FLAG_DOCUMENT_7               48744 non-null  int64
101 FLAG_DOCUMENT_8               48744 non-null  int64
102 FLAG_DOCUMENT_9               48744 non-null  int64
103 FLAG_DOCUMENT_10              48744 non-null  int64
104 FLAG_DOCUMENT_11              48744 non-null  int64
105 FLAG_DOCUMENT_12              48744 non-null  int64
106 FLAG_DOCUMENT_13              48744 non-null  int64
107 FLAG_DOCUMENT_14              48744 non-null  int64
108 FLAG_DOCUMENT_15              48744 non-null  int64
109 FLAG_DOCUMENT_16              48744 non-null  int64
110 FLAG_DOCUMENT_17              48744 non-null  int64
111 FLAG_DOCUMENT_18              48744 non-null  int64
```

```
 112 FLAG_DOCUMENT_19              48744 non-null  int64
 113 FLAG_DOCUMENT_20              48744 non-null  int64
 114 FLAG_DOCUMENT_21              48744 non-null  int64
 115 AMT_REQ_CREDIT_BUREAU_HOUR    42695 non-null  float64
 116 AMT_REQ_CREDIT_BUREAU_DAY     42695 non-null  float64
 117 AMT_REQ_CREDIT_BUREAU_WEEK    42695 non-null  float64
 118 AMT_REQ_CREDIT_BUREAU_MON     42695 non-null  float64
 119 AMT_REQ_CREDIT_BUREAU_QRT     42695 non-null  float64
 120 AMT_REQ_CREDIT_BUREAU_YEAR    42695 non-null  float64
dtypes: float64(65), int64(40), object(16)
memory usage: 45.0+ MB
None
--------------------------------------------------------------------------------
Shape of the df application_test is (48744, 121)

--------------------------------------------------------------------------------
Statistical summary of application_test is :
--------------------------------------------------------------------------------
Description of the df application_test:
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	A
count	307511.00	307511.00	307511.00	3.075110e+05	307511.00	307499.00	
mean	278180.52	0.08	0.42	1.687979e+05	599026.00	27108.57	
std	102790.18	0.27	0.72	2.371231e+05	402490.78	14493.74	
min	100002.00	0.00	0.00	2.565000e+04	45000.00	1615.50	
25%	189145.50	0.00	0.00	1.125000e+05	270000.00	16524.00	
50%	278202.00	0.00	0.00	1.471500e+05	513531.00	24903.00	
75%	367142.50	0.00	1.00	2.025000e+05	808650.00	34596.00	
max	456255.00	1.00	19.00	1.170000e+08	4050000.00	258025.50	

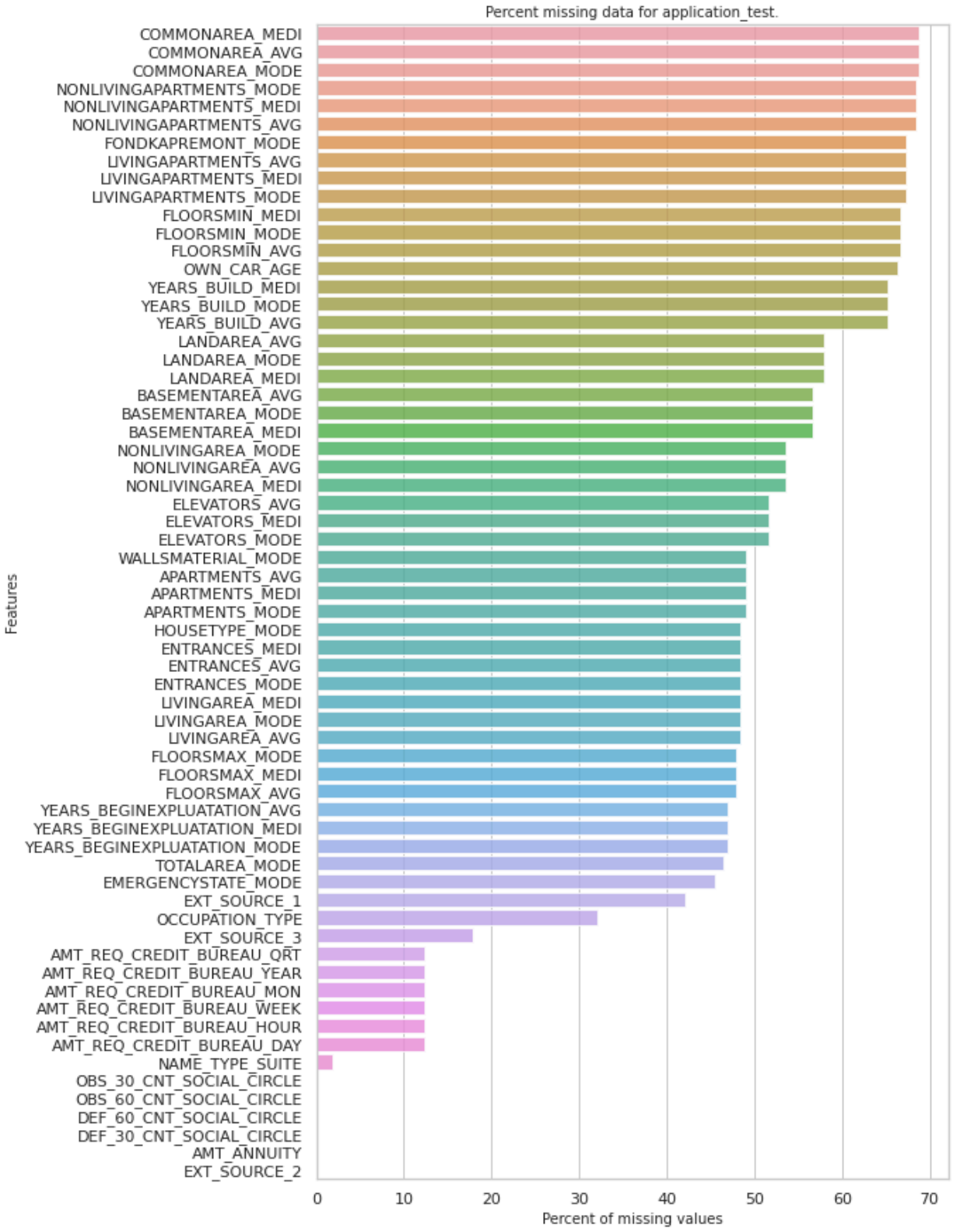
In [ ]:
```
display_feature_info(datasets['application_test'], 'application_test')
```

```
Description of the df continued for application_test:

------------------------------------------------------------------------------
Data type value counts:
 float64    65
int64      40
object     16
dtype: int64

Return number of unique elements in the object.

NAME_CONTRACT_TYPE             2
CODE_GENDER                    2
FLAG_OWN_CAR                   2
FLAG_OWN_REALTY                2
NAME_TYPE_SUITE                7
NAME_INCOME_TYPE               7
NAME_EDUCATION_TYPE            5
NAME_FAMILY_STATUS             5
NAME_HOUSING_TYPE              6
OCCUPATION_TYPE               18
WEEKDAY_APPR_PROCESS_START     7
ORGANIZATION_TYPE             58
FONDKAPREMONT_MODE             4
HOUSETYPE_MODE                 3
WALLSMATERIAL_MODE             7
EMERGENCYSTATE_MODE            2
dtype: int64
------------------------------------------------------------------------------
Categorical and Numerical(int + float) features  of application_test.
------------------------------------------------------------------------------

{'int64': Index(['SK_ID_CURR', 'CNT_CHILDREN', 'DAYS_BIRTH', 'DAYS_EMPLOYED',
       'DAYS_ID_PUBLISH', 'FLAG_MOBIL', 'FLAG_EMP_PHONE', 'FLAG_WORK_PHONE',
       'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL', 'REGION_RATING_CLIENT',
       'REGION_RATING_CLIENT_W_CITY', 'HOUR_APPR_PROCESS_START',
       'REG_REGION_NOT_LIVE_REGION', 'REG_REGION_NOT_WORK_REGION',
       'LIVE_REGION_NOT_WORK_REGION', 'REG_CITY_NOT_LIVE_CITY',
       'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'FLAG_DOCUMENT_2',
       'FLAG_DOCUMENT_3', 'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5',
       'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7', 'FLAG_DOCUMENT_8',
       'FLAG_DOCUMENT_9', 'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11',
       'FLAG_DOCUMENT_12', 'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14',
       'FLAG_DOCUMENT_15', 'FLAG_DOCUMENT_16', 'FLAG_DOCUMENT_17',
       'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DOCUMENT_20',
       'FLAG_DOCUMENT_21'],
      dtype='object')}
------------------------------
{'float64': Index(['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRIC
E',
       'REGION_POPULATION_RELATIVE', 'DAYS_REGISTRATION', 'OWN_CAR_AGE',
       'CNT_FAM_MEMBERS', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3',
       'APARTMENTS_AVG', 'BASEMENTAREA_AVG', 'YEARS_BEGINEXPLUATATION_AVG',
       'YEARS_BUILD_AVG', 'COMMONAREA_AVG', 'ELEVATORS_AVG', 'ENTRANCES_AVG',
       'FLOORSMAX_AVG', 'FLOORSMIN_AVG', 'LANDAREA_AVG',
       'LIVINGAPARTMENTS_AVG', 'LIVINGAREA_AVG', 'NONLIVINGAPARTMENTS_AVG',
       'NONLIVINGAREA_AVG', 'APARTMENTS_MODE', 'BASEMENTAREA_MODE',
       'YEARS_BEGINEXPLUATATION_MODE', 'YEARS_BUILD_MODE', 'COMMONAREA_MODE',
       'ELEVATORS_MODE', 'ENTRANCES_MODE', 'FLOORSMAX_MODE', 'FLOORSMIN_MODE',
       'LANDAREA_MODE', 'LIVINGAPARTMENTS_MODE', 'LIVINGAREA_MODE',
```

```
       'NONLIVINGAPARTMENTS_MODE', 'NONLIVINGAREA_MODE', 'APARTMENTS_MEDI',
       'BASEMENTAREA_MEDI', 'YEARS_BEGINEXPLUATATION_MEDI', 'YEARS_BUILD_MEDI',
       'COMMONAREA_MEDI', 'ELEVATORS_MEDI', 'ENTRANCES_MEDI', 'FLOORSMAX_MEDI',
       'FLOORSMIN_MEDI', 'LANDAREA_MEDI', 'LIVINGAPARTMENTS_MEDI',
       'LIVINGAREA_MEDI', 'NONLIVINGAPARTMENTS_MEDI', 'NONLIVINGAREA_MEDI',
       'TOTALAREA_MODE', 'OBS_30_CNT_SOCIAL_CIRCLE',
       'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE',
       'DEF_60_CNT_SOCIAL_CIRCLE', 'DAYS_LAST_PHONE_CHANGE',
       'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_BUREAU_DAY',
       'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON',
       'AMT_REQ_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR'],
      dtype='object')}
------------------------------
{'object': Index(['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REAL
TY',
       'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE',
       'NAME_FAMILY_STATUS', 'NAME_HOUSING_TYPE', 'OCCUPATION_TYPE',
       'WEEKDAY_APPR_PROCESS_START', 'ORGANIZATION_TYPE', 'FONDKAPREMONT_MODE',
       'HOUSETYPE_MODE', 'WALLSMATERIAL_MODE', 'EMERGENCYSTATE_MODE'],
      dtype='object')}
------------------------------



--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

 The Missing Data:
```

	Percent	Train Missing Count
COMMONAREA_MEDI	68.72	33495
COMMONAREA_AVG	68.72	33495
COMMONAREA_MODE	68.72	33495
NONLIVINGAPARTMENTS_MODE	68.41	33347
NONLIVINGAPARTMENTS_MEDI	68.41	33347
NONLIVINGAPARTMENTS_AVG	68.41	33347
FONDKAPREMONT_MODE	67.28	32797
LIVINGAPARTMENTS_AVG	67.25	32780
LIVINGAPARTMENTS_MEDI	67.25	32780
LIVINGAPARTMENTS_MODE	67.25	32780
FLOORSMIN_MEDI	66.61	32466
FLOORSMIN_MODE	66.61	32466
FLOORSMIN_AVG	66.61	32466
OWN_CAR_AGE	66.29	32312
YEARS_BUILD_MEDI	65.28	31818
YEARS_BUILD_MODE	65.28	31818
YEARS_BUILD_AVG	65.28	31818
LANDAREA_AVG	57.96	28254
LANDAREA_MODE	57.96	28254
LANDAREA_MEDI	57.96	28254
BASEMENTAREA_AVG	56.71	27641
BASEMENTAREA_MODE	56.71	27641
BASEMENTAREA_MEDI	56.71	27641
NONLIVINGAREA_MODE	53.51	26084
NONLIVINGAREA_AVG	53.51	26084
NONLIVINGAREA_MEDI	53.51	26084
ELEVATORS_AVG	51.68	25189
ELEVATORS_MEDI	51.68	25189
ELEVATORS_MODE	51.68	25189
WALLSMATERIAL_MODE	49.02	23893
APARTMENTS_AVG	49.01	23887
APARTMENTS_MEDI	49.01	23887
APARTMENTS_MODE	49.01	23887
HOUSETYPE_MODE	48.46	23619

	Percent	Train Missing Count
ENTRANCES_MEDI	48.37	23579
ENTRANCES_AVG	48.37	23579
ENTRANCES_MODE	48.37	23579
LIVINGAREA_MEDI	48.32	23552
LIVINGAREA_MODE	48.32	23552
LIVINGAREA_AVG	48.32	23552
FLOORSMAX_MODE	47.84	23321
FLOORSMAX_MEDI	47.84	23321
FLOORSMAX_AVG	47.84	23321
YEARS_BEGINEXPLUATATION_AVG	46.89	22856
YEARS_BEGINEXPLUATATION_MEDI	46.89	22856
YEARS_BEGINEXPLUATATION_MODE	46.89	22856
TOTALAREA_MODE	46.41	22624
EMERGENCYSTATE_MODE	45.56	22209
EXT_SOURCE_1	42.12	20532
OCCUPATION_TYPE	32.01	15605
EXT_SOURCE_3	17.78	8668
AMT_REQ_CREDIT_BUREAU_QRT	12.41	6049
AMT_REQ_CREDIT_BUREAU_YEAR	12.41	6049
AMT_REQ_CREDIT_BUREAU_MON	12.41	6049
AMT_REQ_CREDIT_BUREAU_WEEK	12.41	6049
AMT_REQ_CREDIT_BUREAU_HOUR	12.41	6049
AMT_REQ_CREDIT_BUREAU_DAY	12.41	6049
NAME_TYPE_SUITE	1.87	911
OBS_30_CNT_SOCIAL_CIRCLE	0.06	29
OBS_60_CNT_SOCIAL_CIRCLE	0.06	29
DEF_60_CNT_SOCIAL_CIRCLE	0.06	29
DEF_30_CNT_SOCIAL_CIRCLE	0.06	29
AMT_ANNUITY	0.05	24
EXT_SOURCE_2	0.02	8



Correlation Analysis

The top 20 correlated features (positive and negative) for application train datset are listed below.

In [ ]:

```
correlations = datasets["application_train"].corr()['TARGET'].sort_values()
print('Most Positive Correlations:\n', correlations.tail(10))
print('\nMost Negative Correlations:\n', correlations.head(10))
```

```
Most Positive Correlations:
 FLAG_DOCUMENT_3                0.044346
REG_CITY_NOT_LIVE_CITY          0.044395
FLAG_EMP_PHONE                  0.045982
REG_CITY_NOT_WORK_CITY          0.050994
DAYS_ID_PUBLISH                 0.051457
DAYS_LAST_PHONE_CHANGE          0.055218
REGION_RATING_CLIENT            0.058899
REGION_RATING_CLIENT_W_CITY     0.060893
DAYS_BIRTH                      0.078239
TARGET                          1.000000
Name: TARGET, dtype: float64

Most Negative Correlations:
 EXT_SOURCE_3                  -0.178919
EXT_SOURCE_2                   -0.160472
EXT_SOURCE_1                   -0.155317
DAYS_EMPLOYED                  -0.044932
FLOORSMAX_AVG                  -0.044003
FLOORSMAX_MEDI                 -0.043768
FLOORSMAX_MODE                 -0.043226
AMT_GOODS_PRICE                -0.039645
REGION_POPULATION_RELATIVE     -0.037227
ELEVATORS_AVG                  -0.034199
Name: TARGET, dtype: float64
```

In [ ]:

```
num_attribs = ['TARGET', 'AMT_INCOME_TOTAL',  'AMT_CREDIT', 'DAYS_EMPLOYED',
               'DAYS_BIRTH', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'AMT_GOO
df = datasets["application_train"].copy()
df2 = df[num_attribs]
corr = df2.corr()
corr.style.background_gradient(cmap='PuBu').set_precision(2)
```

Out[ ]:

	TARGET	AMT_INCOME_TOTAL	AMT_CREDIT	DAYS_EMPLOYED	DAYS_BIRTH	EX
TARGET	1.00	-0.00	-0.03	-0.04	0.08	
AMT_INCOME_TOTAL	-0.00	1.00	0.16	-0.06	0.03	
AMT_CREDIT	-0.03	0.16	1.00	-0.07	-0.06	
DAYS_EMPLOYED	-0.04	-0.06	-0.07	1.00	-0.62	
DAYS_BIRTH	0.08	0.03	-0.06	-0.62	1.00	
EXT_SOURCE_1	-0.16	0.03	0.17	0.29	-0.60	
EXT_SOURCE_2	-0.16	0.06	0.13	-0.02	-0.09	
EXT_SOURCE_3	-0.18	-0.03	0.04	0.11	-0.21	
AMT_GOODS_PRICE	-0.04	0.16	0.99	-0.06	-0.05	

In [ ]:

```
gc.collect()
```

Out[ ]:

```
67268
```

Feature Engineering and Aggregation

Manual Feature Engineering for below mentioned files.

1. installments_payments
2. credit_card_balance
3. previous_application
4. application_train

In [ ]:
```
# Create installment features
class InstallmentPaymentFeaturesAdder(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.l = []

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):

        # new feature creation from the Installment Payment file
        X['PAY_IS_LATE'] = X['DAYS_INSTALMENT'] - X['DAYS_ENTRY_PAYMENT']
        X['AMT_MISSED'] = X['AMT_INSTALMENT'] - X['AMT_PAYMENT']

        return X
```

In [ ]:
```
# Create installment features
class CCBalFeaturesAdder(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.l = []

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):

        # new feature creation from the Credit Card file
        X['DPD_MISSED'] = X['SK_DPD'] - X['SK_DPD_DEF']
        X['CREDIT_UTILIZED'] = X['AMT_CREDIT_LIMIT_ACTUAL'] - X['AMT_DRAWINGS_CURRENT
        X['MIN_CREDIT_AMTMISS'] = X['AMT_INST_MIN_REGULARITY'] - X['AMT_PAYMENT_CURRE

        # Difference between the monthly amount paid - the expected monthly amount
        X['PAYMENT_DIFF_CURR_PAY'] = X['AMT_PAYMENT_TOTAL_CURRENT'] - X['AMT_PAYMENT_
        X['PAYMENT_DIFF_MIN_PAY'] = X['AMT_PAYMENT_TOTAL_CURRENT'] - X['AMT_INST_MIN_
        # Difference between the monthly amount paid - the minimum monthly amount
        return X
```

In [ ]:
```
# Create previous application features
class PrevAppFeaturesAdder(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.l = []

    def fit(self, X, y=None):
```

```
        return self

    def transform(self, X, y=None):
        # new feature in the Previous Application file
        X['INTEREST'] = X['CNT_PAYMENT'] * X['AMT_ANNUITY'] - X['AMT_CREDIT']
        X['INTEREST_PER_CREDIT'] = X['INTEREST'] / X['AMT_CREDIT']
        X['CREDIT_SUCCESS'] = X['AMT_APPLICATION'] - X['AMT_CREDIT']
        X['INTEREST_RT'] = 2 * 12 * X['INTEREST'] / (X['AMT_CREDIT'] * (X['CNT_PAYMEN
        return X
```

In [ ]:

```
from sklearn.base import BaseEstimator, TransformerMixin
import numpy as np

# Create application features
class ApplicationTrainTestFeaturesAdder(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.l = []

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        X = X.copy()  # Avoid modifying the original dataframe

        # credit to income ratio
        X['CREDIT_INCOME_RATIO'] = X['AMT_CREDIT'] / X['AMT_INCOME_TOTAL']

        # annuity to income ratio
        X['ANNUITY_INCOME_RATIO'] = X['AMT_ANNUITY'] / X['AMT_INCOME_TOTAL']

        # length of the credit term
        X['CREDIT_LENGTH'] = X['AMT_ANNUITY'] / X['AMT_CREDIT']

        # income to age ratio
        X['INCOME_AGE_RATIO'] = X['AMT_INCOME_TOTAL'] / X['DAYS_BIRTH']

        # credit to age ratio
        X['CREDIT_AGE_RATIO'] = X['AMT_CREDIT'] / X['DAYS_BIRTH']

        # employment to age ratio
        X['DAYS_EMPLOYED_PERCENT'] = X['DAYS_EMPLOYED'] / X['DAYS_BIRTH']

        # liability feature based on car and real estate ownership
        conditions_temp = [
            (X['FLAG_OWN_CAR'] == 'Y') & (X['FLAG_OWN_REALTY'] == 'Y'),
            (X['FLAG_OWN_CAR'] == 'N') & (X['FLAG_OWN_REALTY'] == 'Y'),
            (X['FLAG_OWN_CAR'] == 'Y') & (X['FLAG_OWN_REALTY'] == 'N'),
            (X['FLAG_OWN_CAR'] == 'N') & (X['FLAG_OWN_REALTY'] == 'N')
        ]
        values_temp = ['0', '1', '2', '3']
        X['HAS_LIBAILITY'] = np.select(conditions_temp, values_temp, default='4')

        # additional ratios
        X['DAYS_EMPLOYED_PCT'] = X['DAYS_EMPLOYED'] / X['DAYS_BIRTH']
        X['CREDIT_INCOME_PCT'] = X['AMT_CREDIT'] / X['AMT_INCOME_TOTAL']
        X['ANNUITY_INCOME_PCT'] = X['AMT_ANNUITY'] / X['AMT_INCOME_TOTAL']
        X['CREDIT_TERM'] = X['AMT_ANNUITY'] / X['AMT_CREDIT']

        return X
```

One Hot Encoder

In [ ]:
```
# Create aggregate features (via pipeline)
class getDummies(BaseEstimator, TransformerMixin):
    def __init__(self, columns=None): # no *args or **kargs
        self.columns = columns

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        cat_df = X.select_dtypes(include ='object')
        if not cat_df.empty:
          mode_row = cat_df.mode()
          X.fillna(mode_row, inplace = True)
          if not mode_row.empty:
            X.fillna(mode_row.iloc[0], inplace = True)
        #else:
        #X.fillna(X.select_dtypes(include = 'object').mode().iloc[0], inplace = True)
        result = pd.get_dummies(X, columns = self.columns)

        #('imputer', SimpleImputer(strategy='most_frequent')),
        return result
```

Feature Aggregator Helper Function

Functions required to perform feature aggregations are listed below

In [ ]:
```
# function to get the numerical features
def get_numattribs(dataDF):
  num_attribs=(dataDF.select_dtypes(include=['int64', 'float64']).columns.tolist())
  print()
  print('Numerical attributes for',ds_name,' : ',num_attribs)
  print()
  return num_attribs
```

In [ ]:
```
class FeaturesAggregator(BaseEstimator, TransformerMixin):
    def __init__(self, file_name=None, features=None, primary_id = None):
        self.prefix = file_name
        self.features = features
        self.numeric_stats = ["min", "max", "mean", "count", "sum"]
        self.categorical_stats = ["mean", "count", "sum"]
        self.primary_id = primary_id
        self.agg_op_features = {}
        self.agg_features_names = [self.primary_id]

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        numeric_cols = list(X.columns[X.columns.isin(self.features)])
        numeric_cols = [num for num in numeric_cols if num not in ['SK_ID_CURR','SK_I
        categorical_cols = list(X.columns[~X.columns.isin(self.features)])
```

```
        for f in numeric_cols:
            self.agg_op_features[f] = self.numeric_stats
            self.agg_features_names = self.agg_features_names + [self.prefix + "_" +

        for f in categorical_cols:
            self.agg_op_features[f] = self.categorical_stats
            self.agg_features_names = self.agg_features_names + [self.prefix + "_" +

        result = X.groupby(self.primary_id).agg(self.agg_op_features)
        result.columns = result.columns.droplevel()
        result = result.reset_index(level=[self.primary_id])
        result.columns = self.agg_features_names
        return result
```

In [ ]:

```
class engineer_features(BaseEstimator, TransformerMixin):
    def __init__(self, features=None):
        self

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):

# FROM APPLICATION
        # ADD INCOME CREDIT PERCENTAGE
        X['ef_INCOME_CREDIT_PERCENT'] = (
            X.AMT_INCOME_TOTAL / X.AMT_CREDIT).replace(np.inf, 0)

        # ADD INCOME PER FAMILY MEMBER
        X['ef_FAM_MEMBER_INCOME'] = (
            X.AMT_INCOME_TOTAL / X.CNT_FAM_MEMBERS).replace(np.inf, 0)

        # ADD ANNUITY AS PERCENTAGE OF ANNUAL INCOME
        X['ef_ANN_INCOME_PERCENT'] = (
            X.AMT_ANNUITY / X.AMT_INCOME_TOTAL).replace(np.inf, 0)

        return X
```

In [ ]:

```
# Creates the following date features
# But could do so much more with these features
#    E.g.,
#      extract the domain address of the homepage and OneHotEncode it
#
# ['release_month','release_day','release_year', 'release_dayofweek','release_quarter
class prep_OCCUPATION_TYPE(BaseEstimator, TransformerMixin):
    def __init__(self, features="OCCUPATION_TYPE"): # no *args or **kargs
        self.features = features
    def fit(self, X, y=None):
        return self  # nothing else to do
    def transform(self, X):
        df = pd.DataFrame(X, columns=self.features)
        #from IPython.core.debugger import Pdb as pdb;    pdb().set_trace() #breakpoi
        df['OCCUPATION_TYPE'] = df['OCCUPATION_TYPE'].apply(lambda x: 1. if x in ['Co
        #df.drop(self.features, axis=1, inplace=True)
        return np.array(df.values)
```

Missing Data Removal

In [ ]:

```
# Remove missing columns
class MissingFeatureRemover(BaseEstimator, TransformerMixin):
    def __init__(self, threshold = .6):
        self.threshold = threshold

    def fit(self, X, y=None):

        # get the percent of missingness in features
        percent = (X.isnull().sum()/X.isnull().count()).sort_values(ascending = False

        # turn into a data frame
        missing_application_train_data  = pd.DataFrame(percent, columns=['Percent'])

        # get the columns with missingness exceeding the threshold
        self.columns_to_drop = list(missing_application_train_data.index[missing_appl

        return self

    def transform(self, X, y=None):

        # drop the columns with missingness over the threshold
        X = X.drop(columns = self.columns_to_drop, axis=1)

        return X
```

Collinear Feature Removal

In [ ]:

```
# Remove features with high colli
class CollinearFeatureRemover(BaseEstimator, TransformerMixin):
    def __init__(self, threshold = .9):
        self.threshold = threshold

    def fit(self, X, y=None):

        # get the correlation matrix for the entire dataset after one hot encoding fe
#         correlation_matrix = X.head(1000).corr().abs()
        correlation_matrix = X.sample(10000).corr().abs()

        # get only the lower portion of collinearity matrix
        lower = correlation_matrix.where(np.tril(np.ones(correlation_matrix.shape), k

        # get the fields with correlation above threshold
        self.columns_to_drop = [index for index in lower.index if any(lower.loc[index

        return self

    def transform(self, X, y=None):

        # drop the columns with collinearity over the threshold
        X = X.drop(columns = self.columns_to_drop, axis=1)

        return X
```

Removal of Zero Variance

In [ ]:
```
# Remove features with near zero variance
class NearZeroVarianceFeatureRemover(BaseEstimator, TransformerMixin):
    def __init__(self, threshold = 0):
        self.threshold = threshold

    def fit(self, X, y=None):

        # get the fields with correlation above threshold
        self.columns_to_drop = [col for col in X.select_dtypes([np.number]).columns i

        return self

    def transform(self, X, y=None):

        # drop the columns with collinearity over the threshold
        X = X.drop(columns = self.columns_to_drop, axis=1)

        return X
```

In [ ]:
```
gc.collect()
```

Out[ ]:
```
0
```

Creating a base copy of the data

In [ ]:
```
appsTrainDF = datasets['application_train'].copy()
X_kaggle_test = datasets['application_test'].copy()
prevAppsDF = datasets["previous_application"].copy()
bureauDF = datasets["bureau"].copy()

bureaubalDF = datasets['bureau_balance'].copy()
ccbalDF = datasets["credit_card_balance"].copy()
installmentspaymentsDF = datasets["installments_payments"].copy()
pos_cash_bal_DF = datasets["POS_CASH_balance"].copy()
```

Tertiary Datasets

The tertiary datasets or tables refer to bureau_balance, POS_CASH_balance, instalments_payments, credit_card_balance

In [ ]:
```
tertiaty_datasets=['bureau_balance','credit_card_balance','installments_payments','PO
```

Third Tier datasets Numerical feature aggregation

Feature aggregation for the tertiary datasets

In [ ]:
```
primary_id1 = "SK_ID_PREV"
primary_id2 = "SK_ID_BUREAU"
```

```
posBal_features = pos_cash_bal_DF.columns.to_list()
instalPay_features = installmentspaymentsDF.columns.to_list()
instalPay_features.extend(['PAY_IS_LATE', 'AMT_MISSED'])

ccBal_features = ccbalDF.columns.to_list()
ccBal_features.extend(['DPD_MISSED', 'CREDIT_UTILIZED', 'MIN_CREDIT_AMTMISS',
                       'PAYMENT_DIFF_CURR_PAY','PAYMENT_DIFF_MIN_PAY'])

burBal_features = bureaubalDF.columns.to_list()

fn_POS_CASH ='POS_CASH_balance'
fn_ins_pay = 'installments_payments'
fn_ccbal = 'credit_card_balance'
fn_bbal ='bureau_balance'
```

Define Pipeline to create aggregator and OHE features.

In [ ]:

```
# set pos cash pipeline
pos_cash_pipe = Pipeline([
    #('imputer', SimpleImputer(strategy='most_frequent')),
    ('ohe', getDummies()),
    ('aggregator', FeaturesAggregator(fn_POS_CASH,posBal_features, primary_id1)),
    ('missing data remover', MissingFeatureRemover()),
    ('collinearity remover', CollinearFeatureRemover())
])


# set installments_payments pipeline
install_pay_pipe = Pipeline([
    ('install_pay_new_features', InstallmentPaymentFeaturesAdder()),
    #('imputer', SimpleImputer(strategy='most_frequent')),
    ('ohe', getDummies()),
    ('aggregator', FeaturesAggregator(fn_ins_pay,instalPay_features, primary_id1)),
    ('missing data remover', MissingFeatureRemover()),
    ('collinearity remover', CollinearFeatureRemover())
])

# set credit_card_balance pipeline
cc_bal_pipe = Pipeline([
    ('install_pay_new_features', CCBalFeaturesAdder()),
    #('imputer', SimpleImputer(strategy='most_frequent')),
    ('ohe', getDummies()),
    ('aggregator', FeaturesAggregator(fn_ccbal,ccBal_features, primary_id1)),
    ('missing data remover', MissingFeatureRemover()),
    ('collinearity remover', CollinearFeatureRemover())
])

# set bureau_balance pipeline
bureau_bal_pipe = Pipeline([
    #('imputer', SimpleImputer(strategy='most_frequent')),
    ('ohe', getDummies()),
    ('aggregator', FeaturesAggregator(fn_bbal, burBal_features, primary_id2)),
    ('missing data remover', MissingFeatureRemover()),
    ('collinearity remover', CollinearFeatureRemover())
])
```

Run the pipelines for tier 3

In [ ]:
```
pos_cash_bal_aggregated = pos_cash_pipe.fit_transform(pos_cash_bal_DF)
#del pos_cash_bal_DF

installments_pmnts_aggregated = install_pay_pipe.fit_transform(installmentspaymentsDF
#del installmentspaymentsDF

ccblance_aggregated = cc_bal_pipe.fit_transform(ccbalDF)
#del ccbalDF

bureaubal_aggregated = bureau_bal_pipe.fit_transform(bureaubalDF)
#del bureaubalDF

gc.collect()
```

Out[ ]:
```
0
```

Step 1: Merge Tier 3 with Tier 2

In [ ]:
```
print(pos_cash_bal_aggregated.shape)
print(ccblance_aggregated.shape)
print(installments_pmnts_aggregated.shape)
print(bureaubal_aggregated.shape)
```

```
(936325, 29)
(104307, 64)
(997752, 21)
(817395, 20)
```

Merging the aggregated features for pos_cash_bal , installments_pmnts , credit card balance with Previous application

In [ ]:
```
prevApps_ThirdTierMerge = True

posBal_join_feature = 'SK_ID_PREV'
instalPay_join_feature = 'SK_ID_PREV'
ccBal_join_feature = 'SK_ID_PREV'
burBal_join_feature = 'SK_ID_BUREAU'
prevApps_join_feature = 'SK_ID_CURR'
bureau_join_feature = 'SK_ID_CURR'

if prevApps_ThirdTierMerge:
  # Merge Datasets
  prevAppsDF = prevAppsDF.merge(pos_cash_bal_aggregated, how='left', on=posBal_join_f
  prevAppsDF = prevAppsDF.merge(installments_pmnts_aggregated, how='left', on=instalP
  prevAppsDF = prevAppsDF.merge(ccblance_aggregated, how='left', on=ccBal_join_featur
```

Merging the aggregated features the dataset Bureau Balance with Bureau as per the data model.

In [ ]:
```
bureau_ThirdTierMerge = True
```

```
if bureau_ThirdTierMerge:
  bureauDF = bureauDF.merge(bureaubal_aggregated, how='left', on=burBal_join_feature)
```

In [ ]:
```
print(prevAppsDF.shape)
print(bureauDF.shape)
```

```
(1670214, 148)
(1716428, 36)
```

In [ ]:
```
gc.collect()
```

Out[ ]:
```
0
```

Secondary Datasets

Second Tier datasets feature aggregation and OHE pipeline

In [ ]:
```
primary_id1 = "SK_ID_CURR"

fn_bureau = 'bureau'
fn_prevapps = 'previous_application'
fn_appsTrain = 'application_train'
fn_appsTest = 'application_test'

# dataframe names for reference
# appsTrainDF
# appsTestDF
# prevAppsDF
# bureauDF
```

Define the second tier pipeline

In [ ]:
```
# get column names
prevApps_features = prevAppsDF.columns.to_list()
prevApps_features.extend(['INTEREST', 'INTEREST_PER_CREDIT', 'CREDIT_SUCCESS', 'INTER

bureau_features = bureauDF.columns.to_list()

# set previous_application pipeline
prev_app_pipe = Pipeline([
    ('prev_app_feature_adder', PrevAppFeaturesAdder()),
    #('imputer', SimpleImputer(strategy='most_frequent')),
    ('ohe', getDummies()),
    ('aggregator', FeaturesAggregator(fn_prevapps,prevApps_features, primary_id1)),
    ('missing data remover', MissingFeatureRemover()),
    ('collinearity remover', CollinearFeatureRemover())
])

# set bureau pipeline
bureau_pipe = Pipeline([
    #('imputer', SimpleImputer(strategy='most_frequent')),
    ('ohe', getDummies()),
    ('aggregator', FeaturesAggregator(fn_bureau,bureau_features, primary_id1)),
    ('missing data remover', MissingFeatureRemover()),
```

```
    ('collinearity remover', CollinearFeatureRemover())
])
```

In [ ]:
```
prevApps_aggregated = prev_app_pipe.fit_transform(prevAppsDF)
bureau_aggregated = bureau_pipe.fit_transform(bureauDF)

del bureauDF
del prevAppsDF

gc.collect()
```

Out[ ]:
```
0
```

In [ ]:
```
print(prevApps_aggregated.shape)
print(bureau_aggregated.shape)
```

```
(338857, 514)
(305811, 137)
```

Primary Datasets

Merge Aggregated Dataset With Tier 1 Tables - Train and Test

Prior to merging with the Primary data, we will be dropping columns with more than 50% missing values because they are not reliable parameters.

In [ ]:
```
prevApps_join_feature = 'SK_ID_CURR'
bureau_join_feature = 'SK_ID_CURR'

merge_all_data = True

if merge_all_data:
# 1. Join/Merge in prevApps Data
    # Merge all the features with Application_train
    appsTrainDF = appsTrainDF.merge(prevApps_aggregated, how = 'left', on = prevApps_
    appsTrainDF = appsTrainDF.merge(bureau_aggregated, how = 'left', on = bureau_join

    # Merge all the features with Application_train
    X_kaggle_test = X_kaggle_test.merge(prevApps_aggregated, how = 'left', on = prevA
    X_kaggle_test = X_kaggle_test.merge(bureau_aggregated, how = 'left', on = bureau_
```

In [ ]:
```
print(appsTrainDF.shape)
print(X_kaggle_test.shape)
```

```
(307511, 771)
(48744, 770)
```

In [ ]:
```
gc.collect()
```

Out[ ]:
```
0
```

Categorical feature mapping

In [ ]:
```
def get_cat_attribs():
  cat_cols = []
  cat_cols=list(appsTrainDF.select_dtypes(include=['object']).columns)
  return cat_cols

cat_attribs = get_cat_attribs()

over_5_unique = ([])

for att in cat_attribs:
  if (len(appsTrainDF[att].unique()) > 5):
    over_5_unique.append(att)

print(f'{len(over_5_unique)} cat attributes with more than 5 unique values')
```

```
8 cat attributes with more than 5 unique values
```

In [ ]:
```
for att in over_5_unique:

  print(f'{att}:')

  column_total = appsTrainDF[att].shape[0]

  for v in appsTrainDF[att].unique():

    print(f"Rows for {v}: {sum(appsTrainDF[att] == v)} - {round(100 * (sum(appsTrainD
```

```
NAME_TYPE_SUITE:
Rows for Unaccompanied: 248526 - 81%
Rows for Family: 40149 - 13%
Rows for Spouse, partner: 11370 - 4%
Rows for Children: 3267 - 1%
Rows for Other_A: 866 - 0%
Rows for nan: 0 - 0%
Rows for Other_B: 1770 - 1%
Rows for Group of people: 271 - 0%
NAME_INCOME_TYPE:
Rows for Working: 158774 - 52%
Rows for State servant: 21703 - 7%
Rows for Commercial associate: 71617 - 23%
Rows for Pensioner: 55362 - 18%
Rows for Unemployed: 22 - 0%
Rows for Student: 18 - 0%
Rows for Businessman: 10 - 0%
Rows for Maternity leave: 5 - 0%
NAME_FAMILY_STATUS:
Rows for Single / not married: 45444 - 15%
Rows for Married: 196432 - 64%
Rows for Civil marriage: 29775 - 10%
Rows for Widow: 16088 - 5%
Rows for Separated: 19770 - 6%
Rows for Unknown: 2 - 0%
NAME_HOUSING_TYPE:
Rows for House / apartment: 272868 - 89%
Rows for Rented apartment: 4881 - 2%
Rows for With parents: 14840 - 5%
Rows for Municipal apartment: 11183 - 4%
Rows for Office apartment: 2617 - 1%
Rows for Co-op apartment: 1122 - 0%
OCCUPATION_TYPE:
Rows for Laborers: 55186 - 18%
Rows for Core staff: 27570 - 9%
Rows for Accountants: 9813 - 3%
Rows for Managers: 21371 - 7%
Rows for nan: 0 - 0%
Rows for Drivers: 18603 - 6%
Rows for Sales staff: 32102 - 10%
Rows for Cleaning staff: 4653 - 2%
Rows for Cooking staff: 5946 - 2%
Rows for Private service staff: 2652 - 1%
Rows for Medicine staff: 8537 - 3%
Rows for Security staff: 6721 - 2%
Rows for High skill tech staff: 11380 - 4%
Rows for Waiters/barmen staff: 1348 - 0%
Rows for Low-skill Laborers: 2093 - 1%
Rows for Realty agents: 751 - 0%
Rows for Secretaries: 1305 - 0%
Rows for IT staff: 526 - 0%
Rows for HR staff: 563 - 0%
WEEKDAY_APPR_PROCESS_START:
Rows for WEDNESDAY: 51934 - 17%
Rows for MONDAY: 50714 - 16%
Rows for THURSDAY: 50591 - 16%
Rows for SUNDAY: 16181 - 5%
Rows for SATURDAY: 33852 - 11%
Rows for FRIDAY: 50338 - 16%
Rows for TUESDAY: 53901 - 18%
```

```
ORGANIZATION_TYPE:
Rows for Business Entity Type 3: 67992 - 22%
Rows for School: 8893 - 3%
Rows for Government: 10404 - 3%
Rows for Religion: 85 - 0%
Rows for Other: 16683 - 5%
Rows for XNA: 55374 - 18%
Rows for Electricity: 950 - 0%
Rows for Medicine: 11193 - 4%
Rows for Business Entity Type 2: 10553 - 3%
Rows for Self-employed: 38412 - 12%
Rows for Transport: type 2: 2204 - 1%
Rows for Construction: 6721 - 2%
Rows for Housing: 2958 - 1%
Rows for Kindergarten: 6880 - 2%
Rows for Trade: type 7: 7831 - 3%
Rows for Industry: type 11: 2704 - 1%
Rows for Military: 2634 - 1%
Rows for Services: 1575 - 1%
Rows for Security Ministries: 1974 - 1%
Rows for Transport: type 4: 5398 - 2%
Rows for Industry: type 1: 1039 - 0%
Rows for Emergency: 560 - 0%
Rows for Security: 3247 - 1%
Rows for Trade: type 2: 1900 - 1%
Rows for University: 1327 - 0%
Rows for Transport: type 3: 1187 - 0%
Rows for Police: 2341 - 1%
Rows for Business Entity Type 1: 5984 - 2%
Rows for Postal: 2157 - 1%
Rows for Industry: type 4: 877 - 0%
Rows for Agriculture: 2454 - 1%
Rows for Restaurant: 1811 - 1%
Rows for Culture: 379 - 0%
Rows for Hotel: 966 - 0%
Rows for Industry: type 7: 1307 - 0%
Rows for Trade: type 3: 3492 - 1%
Rows for Industry: type 3: 3278 - 1%
Rows for Bank: 2507 - 1%
Rows for Industry: type 9: 3368 - 1%
Rows for Insurance: 597 - 0%
Rows for Trade: type 6: 631 - 0%
Rows for Industry: type 2: 458 - 0%
Rows for Transport: type 1: 201 - 0%
Rows for Industry: type 12: 369 - 0%
Rows for Mobile: 317 - 0%
Rows for Trade: type 1: 348 - 0%
Rows for Industry: type 5: 599 - 0%
Rows for Industry: type 10: 109 - 0%
Rows for Legal Services: 305 - 0%
Rows for Advertising: 429 - 0%
Rows for Trade: type 5: 49 - 0%
Rows for Cleaning: 260 - 0%
Rows for Industry: type 13: 67 - 0%
Rows for Trade: type 4: 64 - 0%
Rows for Telecom: 577 - 0%
Rows for Industry: type 8: 24 - 0%
Rows for Realtor: 396 - 0%
Rows for Industry: type 6: 112 - 0%
WALLSMATERIAL_MODE:
```

```
Rows for Stone, brick: 64815 - 21%
Rows for Block: 9253 - 3%
Rows for nan: 0 - 0%
Rows for Panel: 66040 - 21%
Rows for Mixed: 2296 - 1%
Rows for Wooden: 5362 - 2%
Rows for Others: 1625 - 1%
Rows for Monolithic: 1779 - 1%
```

In [ ]:

```
appsTrainDF['NAME_TYPE_SUITE'] = appsTrainDF['NAME_TYPE_SUITE'].replace({
                        'Family' : 'other',
                        'Spouse, partner' : 'other',
                        'Children' : 'other',
                        'Other_A' : 'other',
                        'Other_B' : 'other',
                        'Group of people' : 'other',})

appsTrainDF['NAME_INCOME_TYPE'] = appsTrainDF['NAME_INCOME_TYPE'].replace({
                        'Unemployed' : 'other',
                        'Student' : 'other',
                        'Businessman' : 'other',
                        'Maternity leave' : 'other',})

appsTrainDF['NAME_FAMILY_STATUS'] = appsTrainDF['NAME_FAMILY_STATUS'].replace({
                        'Single / not married' : 'Not Married',
                        'Married' : 'Married',
                        'Civil marriage' : 'Married',
                        'Widow' : 'Not Married',
                        'Separated' : 'Not Married',
                        'Unknown' : 'Not Married',})

appsTrainDF['NAME_HOUSING_TYPE'] = appsTrainDF['NAME_HOUSING_TYPE'].replace({
                        'House / apartment' : 'House / apartment',
                        'Rented apartment' : 'other',
                        'With parents' : 'other',
                        'Municipal apartment' : 'other',
                        'Office apartment' : 'other',
                        'Co-op apartment' : 'other',})

appsTrainDF['OCCUPATION_TYPE'] = appsTrainDF['OCCUPATION_TYPE'].replace({
                        'Laborers' : 'Service Industry',
                        'Drivers' : 'Service Industry',
                        'Cleaning staff' : 'Service Industry',
                        'Cooking staff' : 'Service Industry',
                        'Private service staff' : 'Service Industry',
                        'Security staff' : 'Service Industry',
                        'Waiters/barmen staff' : 'Service Industry',
                        'Low-skill Laborers' : 'Service Industry',
                        'Core staff' : 'Office',
                        'Accountants' : 'Office',
                        'Managers' : 'Office',
                        'Sales staff' : 'Office',
                        'Medicine staff' : 'Office',
                        'High skill tech staff' : 'Office',
                        'Realty agents' : 'Office',
                        'Secretaries' : 'Office',
                        'IT staff' : 'Office',
                        'HR staff' : 'Office',})
```

```
appsTrainDF['WEEKDAY_APPR_PROCESS_START'] = appsTrainDF['WEEKDAY_APPR_PROCESS_START'
                        'SUNDAY' : 'Weekend',
                        'MONDAY' : 'Weekday',
                        'TUESDAY' : 'Weekday',
                        'WEDNESDAY' : 'Weekday',
                        'THURSDAY' : 'Weekday',
                        'FRIDAY' : 'Weekday',
                        'SATURDAY' : 'Weekend',})

appsTrainDF['ORGANIZATION_TYPE'] = appsTrainDF['ORGANIZATION_TYPE'].replace({
                         'Advertising' : 'Business',
                         'Agriculture' : 'Industrial',
                         'Bank' : 'Business',
                         'Business Entity Type 1' : 'Business',
                         'Business Entity Type 2' : 'Business',
                         'Business Entity Type 3' : 'Business',
                         'Cleaning' : 'Service',
                         'Construction' : 'Industrial',
                         'Culture' : 'Other',
                         'Electricity' : 'Industrial',
                         'Emergency' : 'Government',
                         'Government' : 'Government',
                         'Hotel' : 'Service',
                         'Housing' : 'Other',
                         'Industry: type 1' : 'Industrial',
                         'Industry: type 10' : 'Industrial',
                         'Industry: type 11' : 'Industrial',
                         'Industry: type 12' : 'Industrial',
                         'Industry: type 13' : 'Industrial',
                         'Industry: type 2' : 'Industrial',
                         'Industry: type 3' : 'Industrial',
                         'Industry: type 4' : 'Industrial',
                         'Industry: type 5' : 'Industrial',
                         'Industry: type 6' : 'Industrial',
                         'Industry: type 7' : 'Industrial',
                         'Industry: type 8' : 'Industrial',
                         'Industry: type 9' : 'Industrial',
                         'Insurance' : 'Business',
                         'Kindergarten' : 'Government',
                         'Legal Services' : 'Business',
                         'Medicine' : 'Government',
                         'Military' : 'Government',
                         'Mobile' : 'Other',
                         'Other' : 'Other',
                         'Police' : 'Government',
                         'Postal' : 'Government',
                         'Realtor' : 'Business',
                         'Religion' : 'Other',
                         'Restaurant' : 'Government',
                         'School' : 'Government',
                         'Security' : 'Other',
                         'Security Ministries' : 'Other',
                         'Self-employed' : 'Other',
                         'Services' : 'Service',
                         'Telecom' : 'Business',
                         'Trade: type 1' : 'Trade',
                         'Trade: type 2' : 'Trade',
                         'Trade: type 3' : 'Trade',
                         'Trade: type 4' : 'Trade',
```

```
                        'Trade: type 5' : 'Trade',
                        'Trade: type 6' : 'Trade',
                        'Trade: type 7' : 'Trade',
                        'Transport: type 1' : 'Service',
                        'Transport: type 2' : 'Service',
                        'Transport: type 3' : 'Service',
                        'Transport: type 4' : 'Service',
                        'University' : 'Government',
                        'XNA' : 'XNA'})
```

In [ ]:

```
X_kaggle_test['NAME_TYPE_SUITE'] = X_kaggle_test['NAME_TYPE_SUITE'].replace({
                        'Family' : 'other',
                        'Spouse, partner' : 'other',
                        'Children' : 'other',
                        'Other_A' : 'other',
                        'Other_B' : 'other',
                        'Group of people' : 'other',})

X_kaggle_test['NAME_INCOME_TYPE'] = X_kaggle_test['NAME_INCOME_TYPE'].replace({
                        'Unemployed' : 'other',
                        'Student' : 'other',
                        'Businessman' : 'other',
                        'Maternity leave' : 'other',})

X_kaggle_test['NAME_FAMILY_STATUS'] = X_kaggle_test['NAME_FAMILY_STATUS'].replace({
                        'Single / not married' : 'Not Married',
                        'Married' : 'Married',
                        'Civil marriage' : 'Married',
                        'Widow' : 'Not Married',
                        'Separated' : 'Not Married',
                        'Unknown' : 'Not Married',})

X_kaggle_test['NAME_HOUSING_TYPE'] = X_kaggle_test['NAME_HOUSING_TYPE'].replace({
                        'House / apartment' : 'House / apartment',
                        'Rented apartment' : 'other',
                        'With parents' : 'other',
                        'Municipal apartment' : 'other',
                        'Office apartment' : 'other',
                        'Co-op apartment' : 'other',})

X_kaggle_test['OCCUPATION_TYPE'] = X_kaggle_test['OCCUPATION_TYPE'].replace({
                        'Laborers' : 'Service Industry',
                        'Drivers' : 'Service Industry',
                        'Cleaning staff' : 'Service Industry',
                        'Cooking staff' : 'Service Industry',
                        'Private service staff' : 'Service Industry',
                        'Security staff' : 'Service Industry',
                        'Waiters/barmen staff' : 'Service Industry',
                        'Low-skill Laborers' : 'Service Industry',
                        'Core staff' : 'Office',
                        'Accountants' : 'Office',
                        'Managers' : 'Office',
                        'Sales staff' : 'Office',
                        'Medicine staff' : 'Office',
                        'High skill tech staff' : 'Office',
                        'Realty agents' : 'Office',
                        'Secretaries' : 'Office',
                        'IT staff' : 'Office',
                        'HR staff' : 'Office',})
```

```
X_kaggle_test['WEEKDAY_APPR_PROCESS_START'] = X_kaggle_test['WEEKDAY_APPR_PROCESS_ST
                        'SUNDAY' : 'Weekend',
                        'MONDAY' : 'Weekday',
                        'TUESDAY' : 'Weekday',
                        'WEDNESDAY' : 'Weekday',
                        'THURSDAY' : 'Weekday',
                        'FRIDAY' : 'Weekday',
                        'SATURDAY' : 'Weekend',})

X_kaggle_test['ORGANIZATION_TYPE'] = X_kaggle_test['ORGANIZATION_TYPE'].replace({
                         'Advertising' : 'Business',
                         'Agriculture' : 'Industrial',
                         'Bank' : 'Business',
                         'Business Entity Type 1' : 'Business',
                         'Business Entity Type 2' : 'Business',
                         'Business Entity Type 3' : 'Business',
                         'Cleaning' : 'Service',
                         'Construction' : 'Industrial',
                         'Culture' : 'Other',
                         'Electricity' : 'Industrial',
                         'Emergency' : 'Government',
                         'Government' : 'Government',
                         'Hotel' : 'Service',
                         'Housing' : 'Other',
                         'Industry: type 1' : 'Industrial',
                         'Industry: type 10' : 'Industrial',
                         'Industry: type 11' : 'Industrial',
                         'Industry: type 12' : 'Industrial',
                         'Industry: type 13' : 'Industrial',
                         'Industry: type 2' : 'Industrial',
                         'Industry: type 3' : 'Industrial',
                         'Industry: type 4' : 'Industrial',
                         'Industry: type 5' : 'Industrial',
                         'Industry: type 6' : 'Industrial',
                         'Industry: type 7' : 'Industrial',
                         'Industry: type 8' : 'Industrial',
                         'Industry: type 9' : 'Industrial',
                         'Insurance' : 'Business',
                         'Kindergarten' : 'Government',
                         'Legal Services' : 'Business',
                         'Medicine' : 'Government',
                         'Military' : 'Government',
                         'Mobile' : 'Other',
                         'Other' : 'Other',
                         'Police' : 'Government',
                         'Postal' : 'Government',
                         'Realtor' : 'Business',
                         'Religion' : 'Other',
                         'Restaurant' : 'Government',
                         'School' : 'Government',
                         'Security' : 'Other',
                         'Security Ministries' : 'Other',
                         'Self-employed' : 'Other',
                         'Services' : 'Service',
                         'Telecom' : 'Business',
                         'Trade: type 1' : 'Trade',
                         'Trade: type 2' : 'Trade',
                         'Trade: type 3' : 'Trade',
```

```
                                'Trade: type 4' : 'Trade',
                                'Trade: type 5' : 'Trade',
                                'Trade: type 6' : 'Trade',
                                'Trade: type 7' : 'Trade',
                                'Transport: type 1' : 'Service',
                                'Transport: type 2' : 'Service',
                                'Transport: type 3' : 'Service',
                                'Transport: type 4' : 'Service',
                                'University' : 'Government',
                                'XNA' : 'XNA'})
```

In [ ]:

```
for att in over_5_unique:

  print(f'{att}:')

  column_total = appsTrainDF[att].shape[0]

  for v in appsTrainDF[att].unique():

    print(f"Rows for {v}: {sum(appsTrainDF[att] == v)} - {round(100 * (sum(appsTrainD
```

```
NAME_TYPE_SUITE:
Rows for Unaccompanied: 248526 - 81%
Rows for other: 57693 - 19%
Rows for nan: 0 - 0%
NAME_INCOME_TYPE:
Rows for Working: 158774 - 52%
Rows for State servant: 21703 - 7%
Rows for Commercial associate: 71617 - 23%
Rows for Pensioner: 55362 - 18%
Rows for other: 55 - 0%
NAME_FAMILY_STATUS:
Rows for Not Married: 81304 - 26%
Rows for Married: 226207 - 74%
NAME_HOUSING_TYPE:
Rows for House / apartment: 272868 - 89%
Rows for other: 34643 - 11%
OCCUPATION_TYPE:
Rows for Service Industry: 97202 - 32%
Rows for Office: 113918 - 37%
Rows for nan: 0 - 0%
WEEKDAY_APPR_PROCESS_START:
Rows for Weekday: 257478 - 84%
Rows for Weekend: 50033 - 16%
ORGANIZATION_TYPE:
Rows for Business: 89340 - 29%
Rows for Government: 48200 - 16%
Rows for Other: 64055 - 21%
Rows for XNA: 55374 - 18%
Rows for Industrial: 24436 - 8%
Rows for Service: 11791 - 4%
Rows for Trade: 14315 - 5%
WALLSMATERIAL_MODE:
Rows for Stone, brick: 64815 - 21%
Rows for Block: 9253 - 3%
Rows for nan: 0 - 0%
Rows for Panel: 66040 - 21%
Rows for Mixed: 2296 - 1%
Rows for Wooden: 5362 - 2%
Rows for Others: 1625 - 1%
Rows for Monolithic: 1779 - 1%
```

In [ ]:

```
print(appsTrainDF.shape)
print(X_kaggle_test.shape)
```

```
(307511, 771)
(48744, 770)
```

Final Application_train pipeline

In [ ]:

```
# set application pipeline
application_pipe = Pipeline([
    ('app_train_features', ApplicationTrainTestFeaturesAdder()),
    ('ohe', getDummies()),
    ('missing data remover', MissingFeatureRemover()),
    ('collinearity remover', CollinearFeatureRemover()),
    ('near zero variance remover', NearZeroVarianceFeatureRemover())
])
```

In [ ]:
```
appsTrainDF = application_pipe.fit_transform(appsTrainDF)
X_kaggle_test = application_pipe.transform(X_kaggle_test)
```

In [ ]:
```
print(appsTrainDF.shape)
print(X_kaggle_test.shape)
```

```
(307511, 685)
(48744, 683)
```

In [ ]:
```
drop_uncommon=(list(set(appsTrainDF.columns.tolist()) - set(X_kaggle_test.columns.tol
drop_uncommon.remove('TARGET')
drop_uncommon
```

Out[ ]:
```
['CODE_GENDER_XNA']
```

In [ ]:
```
appsTrainDF=appsTrainDF.drop(columns=drop_uncommon)
```

In [ ]:
```
print(appsTrainDF.shape)
print(X_kaggle_test.shape)
```

```
(307511, 684)
(48744, 683)
```

Output Dataframes to files

In [ ]:
```
appsTrainDF.to_csv("/content/Data/appsTrainDF.csv",index=False)
X_kaggle_test.to_csv("/content/Data/X_kaggle_test.csv",index=False)
```

Data Preparation Ends here with all numeric aggregated features and polynomial features all accumulation to :

> Application_train -- (307511, 684)
> Application_test -- (48744, 683)

Data Summary

In [ ]:
```
print(appsTrainDF.shape)
print(X_kaggle_test.shape)
```

```
(307511, 684)
(48744, 683)
```

Total numeric features in the application train df.

In [ ]:
```
appsTrainDF.select_dtypes(exclude=['object']).columns
```

Out[ ]:
```
Index(['SK_ID_CURR', 'TARGET', 'CNT_CHILDREN', 'AMT_INCOME_TOTAL',
       'AMT_CREDIT', 'AMT_ANNUITY', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'DAYS_REGISTRATION',
       ...
       'WALLSMATERIAL_MODE_Monolithic', 'WALLSMATERIAL_MODE_Others',
       'WALLSMATERIAL_MODE_Panel', 'WALLSMATERIAL_MODE_Stone, brick',
       'WALLSMATERIAL_MODE_Wooden', 'EMERGENCYSTATE_MODE_No',
       'HAS_LIBAILITY_0', 'HAS_LIBAILITY_1', 'HAS_LIBAILITY_2',
       'HAS_LIBAILITY_3'],
      dtype='object', length=684)
```

Total Categorical features in the application train df.

In [ ]:
```
appsTrainDF.select_dtypes(include=['object']).columns
```

Out[ ]:
```
Index([], dtype='object')
```

Deductions from the list of dtypes of the appsTrainDF

- There 641 numerical features.
- There are 43 categorical features which have been encoded.

In [ ]:
```
appsTrainDF.dtypes.value_counts()
```

Out[ ]:

	count
float64	602
bool	43
int64	39

dtype: int64

In [ ]:
```
start = time()
correlation_with_all_features = appsTrainDF.corr()
end = time()
```

In [ ]:
```
print("Time taken for correlation ", ctime(end - start))
print()
correlation_with_all_features['TARGET'].sort_values()
```

```
Time taken for correlation  Thu Jan  1 00:05:34 1970
```

Out[ ]:

	TARGET
EXT_SOURCE_3	-0.178919
EXT_SOURCE_2	-0.160472
EXT_SOURCE_1	-0.155317
OCCUPATION_TYPE_Office	-0.066085
previous_application_NAME_CONTRACT_STATUS_Approved_mean	-0.063521
...	...
bureau_CREDIT_ACTIVE_Active_mean	0.077356
previous_application_NAME_CONTRACT_STATUS_Refused_mean	0.077671
DAYS_BIRTH	0.078239
bureau_DAYS_CREDIT_mean	0.089729
TARGET	1.000000

684 rows × 1 columns

dtype: float64

In [ ]:
```
# correlation_with_all_features.reset_index(inplace= True)
len(correlation_with_all_features.index)
```

Out[ ]: 684

In [ ]:
```
# set this value to choose the number of positive and negative correlated features
n_val = 50


print("---"*50)
print("---"*50)

print("    Total correlation of all the features.    " )

print("---"*50)
print("---"*50)

print(f"Top {n_val} negative correlated features")
print()
print(correlation_with_all_features.TARGET.sort_values(ascending = True).head(n_val))
print()
print()
print(f"Top {n_val} positive correlated features")
print()
print(correlation_with_all_features.TARGET.sort_values(ascending = True).tail(n_val))
```

```
--------------------------------------------------------------------------------
--------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------
    Total correlation of all the features.
--------------------------------------------------------------------------------
--------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------
Top 50 negative correlated features

EXT_SOURCE_3                                                        -0.17891
9
EXT_SOURCE_2                                                        -0.16047
2
EXT_SOURCE_1                                                        -0.15531
7
OCCUPATION_TYPE_Office                                              -0.06608
5
previous_application_NAME_CONTRACT_STATUS_Approved_mean             -0.06352
1
NAME_EDUCATION_TYPE_Higher education                                -0.05659
3
CODE_GENDER_F                                                       -0.05470
4
previous_application_DAYS_FIRST_DRAWING_mean                        -0.04880
3
DAYS_EMPLOYED                                                       -0.04493
2
previous_application_DAYS_FIRST_DRAWING_min                         -0.04464
3
FLOORSMAX_AVG                                                       -0.04400
3
previous_application_RATE_DOWN_PAYMENT_sum                          -0.04169
3
previous_application_NAME_YIELD_GROUP_low_normal_mean               -0.04113
4
previous_application_installments_payments_PAY_IS_LATE_mean_sum     -0.04047
9
previous_application_RATE_DOWN_PAYMENT_max                          -0.04009
6
previous_application_INTEREST_RT_sum                                -0.03953
3
previous_application_PRODUCT_COMBINATION_Cash X-Sell: low_mean      -0.03749
4
REGION_POPULATION_RELATIVE                                          -0.03722
7
previous_application_INTEREST_RT_mean                               -0.03635
0
previous_application_HOUR_APPR_PROCESS_START_mean                   -0.03592
7
previous_application_POS_CASH_balance_MONTHS_BALANCE_count_sum      -0.03584
6
previous_application_AMT_ANNUITY_mean                               -0.03487
1
previous_application_NAME_PAYMENT_TYPE_Cash through the bank_mean   -0.03466
9
ELEVATORS_AVG                                                       -0.03419
9
previous_application_PRODUCT_COMBINATION_POS industry with interest_mean -0.03394
```

```
2
previous_application_installments_payments_NUM_INSTALMENT_VERSION_min_mean   -0.03360
8
previous_application_RATE_DOWN_PAYMENT_mean                                 -0.03360
1
previous_application_NAME_CONTRACT_TYPE_Consumer loans_mean                 -0.03262
4
previous_application_AMT_ANNUITY_min                                        -0.03224
9
previous_application_installments_payments_NUM_INSTALMENT_VERSION_max_sum    -0.03183
5
previous_application_DAYS_FIRST_DRAWING_count                               -0.03183
3
previous_application_HOUR_APPR_PROCESS_START_min                            -0.03142
7
previous_application_HOUR_APPR_PROCESS_START_max                            -0.03084
7
previous_application_PRODUCT_COMBINATION_POS industry with interest_sum      -0.03082
7
previous_application_installments_payments_PAY_IS_LATE_mean_max             -0.03057
4
AMT_CREDIT                                                                  -0.03036
9
previous_application_installments_payments_PAY_IS_LATE_sum_sum              -0.03026
8
previous_application_NAME_GOODS_CATEGORY_Furniture_mean                     -0.03017
4
APARTMENTS_AVG                                                              -0.02949
8
previous_application_installments_payments_NUM_INSTALMENT_VERSION_sum_sum    -0.02945
7
previous_application_NAME_YIELD_GROUP_low_action_mean                       -0.02934
0
previous_application_AMT_ANNUITY_max                                        -0.02896
6
previous_application_NAME_GOODS_CATEGORY_Furniture_sum                      -0.02892
4
previous_application_NAME_SELLER_INDUSTRY_Furniture_mean                    -0.02863
2
FLAG_DOCUMENT_6                                                             -0.02860
2
NAME_HOUSING_TYPE_House / apartment                                         -0.02855
5
previous_application_installments_payments_PAY_IS_LATE_min_max              -0.02821
3
previous_application_NAME_YIELD_GROUP_low_normal_sum                        -0.02801
7
previous_application_installments_payments_PAY_IS_LATE_mean_mean            -0.02757
9
previous_application_CREDIT_SUCCESS_sum                                     -0.02726
6
Name: TARGET, dtype: float64


Top 50 positive correlated features

previous_application_PRODUCT_COMBINATION_Card Street_mean                    0.04024
2
previous_application_CODE_REJECT_REASON_LIMIT_sum                            0.04050
3
```

```
previous_application_credit_card_balance_AMT_BALANCE_max_sum                  0.04140
9
DAYS_REGISTRATION                                                             0.04197
5
bureau_DAYS_CREDIT_sum                                                        0.04200
0
previous_application_NAME_YIELD_GROUP_XNA_mean                                0.04284
8
bureau_DAYS_CREDIT_UPDATE_min                                                 0.04286
4
previous_application_credit_card_balance_AMT_DRAWINGS_ATM_CURRENT_mean_sum     0.04314
8
previous_application_POS_CASH_balance_MONTHS_BALANCE_min_sum                  0.04395
4
previous_application_POS_CASH_balance_SK_DPD_DEF_max_min                      0.04430
7
FLAG_DOCUMENT_3                                                               0.04434
6
REG_CITY_NOT_LIVE_CITY                                                        0.04439
5
bureau_CREDIT_TYPE_Microloan_mean                                             0.04443
9
previous_application_NAME_CONTRACT_TYPE_Revolving loans_sum                   0.04560
2
previous_application_NAME_CLIENT_TYPE_New_sum                                 0.04604
8
previous_application_DAYS_DECISION_mean                                       0.04686
4
bureau_DAYS_CREDIT_ENDDATE_mean                                               0.04698
3
previous_application_CODE_REJECT_REASON_HC_sum                                0.04706
7
previous_application_credit_card_balance_CNT_DRAWINGS_CURRENT_mean_sum         0.04750
7
previous_application_PRODUCT_COMBINATION_Card Street_sum                      0.04795
3
previous_application_POS_CASH_balance_SK_DPD_DEF_max_mean                     0.04856
7
bureau_DAYS_CREDIT_max                                                        0.04978
2
NAME_EDUCATION_TYPE_Secondary / secondary special                             0.04982
4
REG_CITY_NOT_WORK_CITY                                                        0.05099
4
DAYS_ID_PUBLISH                                                               0.05145
7
previous_application_credit_card_balance_CNT_DRAWINGS_ATM_CURRENT_max_sum      0.05233
5
bureau_DAYS_ENDDATE_FACT_mean                                                 0.05320
0
previous_application_DAYS_DECISION_min                                        0.05343
4
bureau_DAYS_CREDIT_ENDDATE_sum                                                0.05373
5
previous_application_CODE_REJECT_REASON_HC_mean                               0.05453
1
DAYS_LAST_PHONE_CHANGE                                                        0.05521
8
previous_application_CODE_REJECT_REASON_SCOFR_mean                            0.05586
5
```

```
bureau_DAYS_ENDDATE_FACT_min                                        0.05588
7
previous_application_CODE_REJECT_REASON_SCOFR_sum                   0.05628
4
previous_application_credit_card_balance_CNT_DRAWINGS_CURRENT_max_sum  0.05675
7
previous_application_NAME_PRODUCT_TYPE_walk-in_mean                 0.05741
2
NAME_INCOME_TYPE_Working                                            0.05748
1
REGION_RATING_CLIENT                                                0.05889
9
previous_application_NAME_PRODUCT_TYPE_walk-in_sum                  0.06262
8
previous_application_NAME_CONTRACT_STATUS_Refused_sum               0.06446
9
previous_application_credit_card_balance_CNT_DRAWINGS_ATM_CURRENT_mean_sum  0.06529
3
bureau_CREDIT_ACTIVE_Active_sum                                     0.06712
8
bureau_DAYS_CREDIT_UPDATE_mean                                      0.06892
7
previous_application_INTEREST_PER_CREDIT_max                        0.06912
5
bureau_DAYS_CREDIT_min                                              0.07524
8
bureau_CREDIT_ACTIVE_Active_mean                                    0.07735
6
previous_application_NAME_CONTRACT_STATUS_Refused_mean              0.07767
1
DAYS_BIRTH                                                          0.07823
9
bureau_DAYS_CREDIT_mean                                             0.08972
9
TARGET                                                              1.00000
0
Name: TARGET, dtype: float64
```

In [ ]:

```
correlation_with_all_features.TARGET.sort_values(ascending = True)[-n_val:]
```

Out[ ]:

	TARGET
previous_application_PRODUCT_COMBINATION_Card Street_mean	0.040242
previous_application_CODE_REJECT_REASON_LIMIT_sum	0.040503
previous_application_credit_card_balance_AMT_BALANCE_max_sum	0.041409
DAYS_REGISTRATION	0.041975
bureau_DAYS_CREDIT_sum	0.042000
previous_application_NAME_YIELD_GROUP_XNA_mean	0.042848
bureau_DAYS_CREDIT_UPDATE_min	0.042864
previous_application_credit_card_balance_AMT_DRAWINGS_ATM_CURRENT_mean_sum	0.043148
previous_application_POS_CASH_balance_MONTHS_BALANCE_min_sum	0.043954
previous_application_POS_CASH_balance_SK_DPD_DEF_max_min	0.044307
FLAG_DOCUMENT_3	0.044346
REG_CITY_NOT_LIVE_CITY	0.044395
bureau_CREDIT_TYPE_Microloan_mean	0.044439
previous_application_NAME_CONTRACT_TYPE_Revolving loans_sum	0.045602
previous_application_NAME_CLIENT_TYPE_New_sum	0.046048
previous_application_DAYS_DECISION_mean	0.046864
bureau_DAYS_CREDIT_ENDDATE_mean	0.046983
previous_application_CODE_REJECT_REASON_HC_sum	0.047067
previous_application_credit_card_balance_CNT_DRAWINGS_CURRENT_mean_sum	0.047507
previous_application_PRODUCT_COMBINATION_Card Street_sum	0.047953
previous_application_POS_CASH_balance_SK_DPD_DEF_max_mean	0.048567
bureau_DAYS_CREDIT_max	0.049782
NAME_EDUCATION_TYPE_Secondary / secondary special	0.049824
REG_CITY_NOT_WORK_CITY	0.050994
DAYS_ID_PUBLISH	0.051457
previous_application_credit_card_balance_CNT_DRAWINGS_ATM_CURRENT_max_sum	0.052335
bureau_DAYS_ENDDATE_FACT_mean	0.053200
previous_application_DAYS_DECISION_min	0.053434
bureau_DAYS_CREDIT_ENDDATE_sum	0.053735
previous_application_CODE_REJECT_REASON_HC_mean	0.054531
DAYS_LAST_PHONE_CHANGE	0.055218
previous_application_CODE_REJECT_REASON_SCOFR_mean	0.055865
bureau_DAYS_ENDDATE_FACT_min	0.055887
previous_application_CODE_REJECT_REASON_SCOFR_sum	0.056284

	TARGET
previous_application_credit_card_balance_CNT_DRAWINGS_CURRENT_max_sum	0.056757
previous_application_NAME_PRODUCT_TYPE_walk-in_mean	0.057412
NAME_INCOME_TYPE_Working	0.057481
REGION_RATING_CLIENT	0.058899
previous_application_NAME_PRODUCT_TYPE_walk-in_sum	0.062628
previous_application_NAME_CONTRACT_STATUS_Refused_sum	0.064469
previous_application_credit_card_balance_CNT_DRAWINGS_ATM_CURRENT_mean_sum	0.065293
bureau_CREDIT_ACTIVE_Active_sum	0.067128
bureau_DAYS_CREDIT_UPDATE_mean	0.068927
previous_application_INTEREST_PER_CREDIT_max	0.069125
bureau_DAYS_CREDIT_min	0.075248
bureau_CREDIT_ACTIVE_Active_mean	0.077356
previous_application_NAME_CONTRACT_STATUS_Refused_mean	0.077671
DAYS_BIRTH	0.078239
bureau_DAYS_CREDIT_mean	0.089729
TARGET	1.000000

dtype: float64

In [ ]:
```
gc.collect()
```

Out[ ]:
```
0
```

In [ ]:
```
corrn=correlation_with_all_features.TARGET.sort_values(ascending = True).head(n_val).
corrp=correlation_with_all_features.TARGET.sort_values(ascending = True).tail(n_val).
corr=corrn + corrp
corr.remove('TARGET')
print(len(corr))
corr
```

```
99
```

Out[ ]:
```
['EXT_SOURCE_3',
 'EXT_SOURCE_2',
 'EXT_SOURCE_1',
 'OCCUPATION_TYPE_Office',
 'previous_application_NAME_CONTRACT_STATUS_Approved_mean',
 'NAME_EDUCATION_TYPE_Higher education',
 'CODE_GENDER_F',
 'previous_application_DAYS_FIRST_DRAWING_mean',
 'DAYS_EMPLOYED',
 'previous_application_DAYS_FIRST_DRAWING_min',
 'FLOORSMAX_AVG',
 'previous_application_RATE_DOWN_PAYMENT_sum',
 'previous_application_NAME_YIELD_GROUP_low_normal_mean',
 'previous_application_installments_payments_PAY_IS_LATE_mean_sum',
 'previous_application_RATE_DOWN_PAYMENT_max',
 'previous_application_INTEREST_RT_sum',
 'previous_application_PRODUCT_COMBINATION_Cash X-Sell: low_mean',
 'REGION_POPULATION_RELATIVE',
 'previous_application_INTEREST_RT_mean',
 'previous_application_HOUR_APPR_PROCESS_START_mean',
 'previous_application_POS_CASH_balance_MONTHS_BALANCE_count_sum',
 'previous_application_AMT_ANNUITY_mean',
 'previous_application_NAME_PAYMENT_TYPE_Cash through the bank_mean',
 'ELEVATORS_AVG',
 'previous_application_PRODUCT_COMBINATION_POS industry with interest_mean',
 'previous_application_installments_payments_NUM_INSTALMENT_VERSION_min_mean',
 'previous_application_RATE_DOWN_PAYMENT_mean',
 'previous_application_NAME_CONTRACT_TYPE_Consumer loans_mean',
 'previous_application_AMT_ANNUITY_min',
 'previous_application_installments_payments_NUM_INSTALMENT_VERSION_max_sum',
 'previous_application_DAYS_FIRST_DRAWING_count',
 'previous_application_HOUR_APPR_PROCESS_START_min',
 'previous_application_HOUR_APPR_PROCESS_START_max',
 'previous_application_PRODUCT_COMBINATION_POS industry with interest_sum',
 'previous_application_installments_payments_PAY_IS_LATE_mean_max',
 'AMT_CREDIT',
 'previous_application_installments_payments_PAY_IS_LATE_sum_sum',
 'previous_application_NAME_GOODS_CATEGORY_Furniture_mean',
 'APARTMENTS_AVG',
 'previous_application_installments_payments_NUM_INSTALMENT_VERSION_sum_sum',
 'previous_application_NAME_YIELD_GROUP_low_action_mean',
 'previous_application_AMT_ANNUITY_max',
 'previous_application_NAME_GOODS_CATEGORY_Furniture_sum',
 'previous_application_NAME_SELLER_INDUSTRY_Furniture_mean',
 'FLAG_DOCUMENT_6',
 'NAME_HOUSING_TYPE_House / apartment',
 'previous_application_installments_payments_PAY_IS_LATE_min_max',
 'previous_application_NAME_YIELD_GROUP_low_normal_sum',
 'previous_application_installments_payments_PAY_IS_LATE_mean_mean',
 'previous_application_CREDIT_SUCCESS_sum',
 'previous_application_PRODUCT_COMBINATION_Card Street_mean',
 'previous_application_CODE_REJECT_REASON_LIMIT_sum',
 'previous_application_credit_card_balance_AMT_BALANCE_max_sum',
 'DAYS_REGISTRATION',
 'bureau_DAYS_CREDIT_sum',
 'previous_application_NAME_YIELD_GROUP_XNA_mean',
 'bureau_DAYS_CREDIT_UPDATE_min',
 'previous_application_credit_card_balance_AMT_DRAWINGS_ATM_CURRENT_mean_sum',
 'previous_application_POS_CASH_balance_MONTHS_BALANCE_min_sum',
 'previous_application_POS_CASH_balance_SK_DPD_DEF_max_min',
```

```
'FLAG_DOCUMENT_3',
'REG_CITY_NOT_LIVE_CITY',
'bureau_CREDIT_TYPE_Microloan_mean',
'previous_application_NAME_CONTRACT_TYPE_Revolving loans_sum',
'previous_application_NAME_CLIENT_TYPE_New_sum',
'previous_application_DAYS_DECISION_mean',
'bureau_DAYS_CREDIT_ENDDATE_mean',
'previous_application_CODE_REJECT_REASON_HC_sum',
'previous_application_credit_card_balance_CNT_DRAWINGS_CURRENT_mean_sum',
'previous_application_PRODUCT_COMBINATION_Card Street_sum',
'previous_application_POS_CASH_balance_SK_DPD_DEF_max_mean',
'bureau_DAYS_CREDIT_max',
'NAME_EDUCATION_TYPE_Secondary / secondary special',
'REG_CITY_NOT_WORK_CITY',
'DAYS_ID_PUBLISH',
'previous_application_credit_card_balance_CNT_DRAWINGS_ATM_CURRENT_max_sum',
'bureau_DAYS_ENDDATE_FACT_mean',
'previous_application_DAYS_DECISION_min',
'bureau_DAYS_CREDIT_ENDDATE_sum',
'previous_application_CODE_REJECT_REASON_HC_mean',
'DAYS_LAST_PHONE_CHANGE',
'previous_application_CODE_REJECT_REASON_SCOFR_mean',
'bureau_DAYS_ENDDATE_FACT_min',
'previous_application_CODE_REJECT_REASON_SCOFR_sum',
'previous_application_credit_card_balance_CNT_DRAWINGS_CURRENT_max_sum',
'previous_application_NAME_PRODUCT_TYPE_walk-in_mean',
'NAME_INCOME_TYPE_Working',
'REGION_RATING_CLIENT',
'previous_application_NAME_PRODUCT_TYPE_walk-in_sum',
'previous_application_NAME_CONTRACT_STATUS_Refused_sum',
'previous_application_credit_card_balance_CNT_DRAWINGS_ATM_CURRENT_mean_sum',
'bureau_CREDIT_ACTIVE_Active_sum',
'bureau_DAYS_CREDIT_UPDATE_mean',
'previous_application_INTEREST_PER_CREDIT_max',
'bureau_DAYS_CREDIT_min',
'bureau_CREDIT_ACTIVE_Active_mean',
'previous_application_NAME_CONTRACT_STATUS_Refused_mean',
'DAYS_BIRTH',
'bureau_DAYS_CREDIT_mean']
```

Processing pipeline

Load Merged Files

In [ ]:

```
DATA_DIR='/content/Data/'
#/content/Data/appsTrainDF.csv
```

In [ ]:

```
%%time
#ds_names = ('appsTrainDF')
ds_names = ('appsTrainDF', 'X_kaggle_test')

for ds_name in ds_names:
    datasets[ds_name]= load_data(os.path.join(DATA_DIR, f'{ds_name}.csv'), ds_name)
```

```
appsTrainDF: shape is (307511, 684)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 684 entries, SK_ID_CURR to HAS_LIBAILITY_3
dtypes: bool(43), float64(602), int64(39)
memory usage: 1.5 GB
None
```

	SK_ID_CURR	TARGET	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	REGION
0	100002	1	0	202500.0	406597.5	24700.5	
1	100003	0	0	270000.0	1293502.5	35698.5	
2	100004	0	0	67500.0	135000.0	6750.0	
3	100006	0	0	135000.0	312682.5	29686.5	
4	100007	0	0	121500.0	513000.0	21865.5	

5 rows × 684 columns

```
X_kaggle_test: shape is (48744, 683)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48744 entries, 0 to 48743
Columns: 683 entries, SK_ID_CURR to HAS_LIBAILITY_3
dtypes: bool(43), float64(602), int64(38)
memory usage: 240.0 MB
None
```

	SK_ID_CURR	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	REGION_POPULA
0	100001	0	135000.0	568800.0	20560.5	
1	100005	0	99000.0	222768.0	17370.0	
2	100013	0	202500.0	663264.0	69777.0	
3	100028	2	315000.0	1575000.0	49018.5	
4	100038	1	180000.0	625500.0	32067.0	

5 rows × 683 columns

```
CPU times: user 18.4 s, sys: 2.52 s, total: 20.9 s
Wall time: 20.8 s
```

In [ ]:
```
print(datasets['appsTrainDF'].shape)
```

```
(307511, 684)
```

In [ ]:
```
print(datasets['X_kaggle_test'].shape)
```

```
(48744, 683)
```

In [ ]:
```
X_kaggle_test=datasets['X_kaggle_test']
```

```
appsTrainDF=datasets['appsTrainDF']
```

In [ ]:
```
train_dataset=appsTrainDF
class_labels = ["No Default","Default"]
```

HCDR Data Pipeline

Column Selector

In [ ]:
```
# Create a class to select numerical or categorical columns since Scikit-Learn doesn'
class DataFrameSelector(BaseEstimator, TransformerMixin):
    def __init__(self, attribute_names):
        self.attribute_names = attribute_names
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        return X[self.attribute_names].values
```

Numerical Attributes

Identify the numeric features we wish to consider.

In [ ]:
```
num_attribs=['EXT_SOURCE_3',
 'EXT_SOURCE_2',
 'EXT_SOURCE_1',
 'OCCUPATION_TYPE_Office',
 'previous_application_NAME_CONTRACT_STATUS_Approved_mean',
 'NAME_EDUCATION_TYPE_Higher education',
 'CODE_GENDER_F',
 'previous_application_DAYS_FIRST_DRAWING_mean',
 'DAYS_EMPLOYED',
 'previous_application_DAYS_FIRST_DRAWING_min',
 'FLOORSMAX_AVG',
 'previous_application_RATE_DOWN_PAYMENT_sum',
 'previous_application_NAME_YIELD_GROUP_low_normal_mean',
 'previous_application_RATE_DOWN_PAYMENT_max',
 'previous_application_INTEREST_RT_sum',
 'previous_application_PRODUCT_COMBINATION_Cash X-Sell: low_mean',
 'REGION_POPULATION_RELATIVE',
 'previous_application_INTEREST_RT_mean',
 'previous_application_HOUR_APPR_PROCESS_START_mean',
 'previous_application_AMT_ANNUITY_mean',
 'previous_application_NAME_PAYMENT_TYPE_Cash through the bank_mean',
 'ELEVATORS_AVG',
 'previous_application_PRODUCT_COMBINATION_POS industry with interest_mean',
 'previous_application_RATE_DOWN_PAYMENT_mean',
 'previous_application_NAME_CONTRACT_TYPE_Consumer loans_mean',
 'previous_application_AMT_ANNUITY_min',
 'previous_application_DAYS_FIRST_DRAWING_count',
 'previous_application_HOUR_APPR_PROCESS_START_min',
 'previous_application_HOUR_APPR_PROCESS_START_max',
 'previous_application_PRODUCT_COMBINATION_POS industry with interest_sum',
 'AMT_CREDIT',
 'previous_application_NAME_GOODS_CATEGORY_Furniture_mean',
 'APARTMENTS_AVG',
```

```
'previous_application_NAME_YIELD_GROUP_low_action_mean',
'previous_application_AMT_ANNUITY_max',
'previous_application_NAME_GOODS_CATEGORY_Furniture_sum',
'FLAG_DOCUMENT_6',
'NAME_HOUSING_TYPE_House / apartment',
'previous_application_NAME_YIELD_GROUP_low_normal_sum',
'previous_application_CREDIT_SUCCESS_sum',
'previous_application_NAME_CLIENT_TYPE_Refreshed_mean',
'bureau_CREDIT_TYPE_Consumer credit_mean',
'previous_application_AMT_DOWN_PAYMENT_max',
'previous_application_NAME_YIELD_GROUP_low_action_sum',
'HOUR_APPR_PROCESS_START',
'FLAG_PHONE',
'previous_application_AMT_DOWN_PAYMENT_count',
'NAME_INCOME_TYPE_State servant',
'previous_application_PRODUCT_COMBINATION_Cash X-Sell: low_sum',
'previous_application_INTEREST_PER_CREDIT_min',
'previous_application_CHANNEL_TYPE_AP+ (Cash loan)_sum',
'bureau_CREDIT_TYPE_Credit card_sum',
'previous_application_CHANNEL_TYPE_AP+ (Cash loan)_mean',
'previous_application_PRODUCT_COMBINATION_Cash X-Sell: high_sum',
'bureau_DAYS_CREDIT_ENDDATE_max',
'previous_application_NAME_YIELD_GROUP_high_sum',
'previous_application_NAME_YIELD_GROUP_high_mean',
'previous_application_NAME_PAYMENT_TYPE_XNA_sum',
'previous_application_CODE_REJECT_REASON_LIMIT_mean',
'previous_application_PRODUCT_COMBINATION_Card Street_mean',
'previous_application_CODE_REJECT_REASON_LIMIT_sum',
'DAYS_REGISTRATION',
'bureau_DAYS_CREDIT_sum',
'previous_application_NAME_YIELD_GROUP_XNA_mean',
'bureau_DAYS_CREDIT_UPDATE_min',
'FLAG_DOCUMENT_3',
'REG_CITY_NOT_LIVE_CITY',
'bureau_CREDIT_TYPE_Microloan_mean',
'previous_application_NAME_CONTRACT_TYPE_Revolving loans_sum',
'previous_application_NAME_CLIENT_TYPE_New_sum',
'previous_application_DAYS_DECISION_mean',
'bureau_DAYS_CREDIT_ENDDATE_mean',
'previous_application_CODE_REJECT_REASON_HC_sum',
'previous_application_PRODUCT_COMBINATION_Card Street_sum',
'bureau_DAYS_CREDIT_max',
'NAME_EDUCATION_TYPE_Secondary / secondary special',
'REG_CITY_NOT_WORK_CITY',
'DAYS_ID_PUBLISH',
'bureau_DAYS_ENDDATE_FACT_mean',
'previous_application_DAYS_DECISION_min',
'bureau_DAYS_CREDIT_ENDDATE_sum',
'previous_application_CODE_REJECT_REASON_HC_mean',
'DAYS_LAST_PHONE_CHANGE',
'previous_application_CODE_REJECT_REASON_SCOFR_mean',
'bureau_DAYS_ENDDATE_FACT_min',
'previous_application_CODE_REJECT_REASON_SCOFR_sum',
'previous_application_NAME_PRODUCT_TYPE_walk-in_mean',
'NAME_INCOME_TYPE_Working',
'REGION_RATING_CLIENT',
'previous_application_NAME_PRODUCT_TYPE_walk-in_sum',
'previous_application_NAME_CONTRACT_STATUS_Refused_sum',
'bureau_CREDIT_ACTIVE_Active_sum',
'bureau_DAYS_CREDIT_UPDATE_mean',
```

```
 'previous_application_INTEREST_PER_CREDIT_max',
 'bureau_DAYS_CREDIT_min',
 'bureau_CREDIT_ACTIVE_Active_mean',
 'previous_application_NAME_CONTRACT_STATUS_Refused_mean',
 'DAYS_BIRTH',
 'bureau_DAYS_CREDIT_mean',
 'previous_application_INTEREST_PER_CREDIT_mean',
'previous_application_CREDIT_SUCCESS_mean',
'previous_application_INTEREST_RT_mean',
'HAS_LIBAILITY_0',
'HAS_LIBAILITY_1',
'HAS_LIBAILITY_2',
'HAS_LIBAILITY_3',
 'FLAG_DOCUMENT_2',
 'FLAG_DOCUMENT_3',
 'FLAG_DOCUMENT_4',
 'FLAG_DOCUMENT_5',
 'FLAG_DOCUMENT_6',
 'FLAG_DOCUMENT_7',
 'FLAG_DOCUMENT_8',
 'FLAG_DOCUMENT_9',
 'FLAG_DOCUMENT_10',
 'FLAG_DOCUMENT_11',
 'FLAG_DOCUMENT_12',
 'FLAG_DOCUMENT_13',
 'FLAG_DOCUMENT_14',
 'FLAG_DOCUMENT_15',
 'FLAG_DOCUMENT_16',
 'FLAG_DOCUMENT_17',
 'FLAG_DOCUMENT_18',
 'FLAG_DOCUMENT_19',
 'FLAG_DOCUMENT_20',
 'FLAG_DOCUMENT_21',
  'AMT_REQ_CREDIT_BUREAU_HOUR',
 'AMT_REQ_CREDIT_BUREAU_DAY',
 'AMT_REQ_CREDIT_BUREAU_WEEK',
 'AMT_REQ_CREDIT_BUREAU_MON',
 'AMT_REQ_CREDIT_BUREAU_QRT',
 'AMT_REQ_CREDIT_BUREAU_YEAR'
]
```

Numerical Pipeline definition

In [ ]:

```
num_pipeline = Pipeline([
        ('selector', DataFrameSelector(num_attribs)),
        ('imputer', SimpleImputer(strategy='mean')),
        ('std_scaler', StandardScaler()),
    ])
```

Categorical Attributes

OHE when previously unseen unique values in the test/validation set

Train, validation and Test sets (and the leakage problem we have mentioned previously):

Let's look at a small usecase to tell us how to deal with this:

- The OneHotEncoder is fitted to the training set, which means that for each unique value present in the training set, for each feature, a new column is created. Let's say we have 39 columns after the encoding up from 30 (before preprocessing).
- The output is a numpy array (when the option sparse=False is used), which has the disadvantage of losing all the information about the original column names and values.
- When we try to transform the test set, after having fitted the encoder to the training set, we obtain a `ValueError`. This is because the there are new, previously unseen unique values in the test set and the encoder doesn't know how to handle these values. In order to use both the transformed training and test sets in machine learning algorithms, we need them to have the same number of columns.

This last problem can be solved by using the option handle_unknown='ignore'of the OneHotEncoder, which, as the name suggests, will ignore previously unseen values when transforming the test set.

Here is a example that in action:

```
# Identify the categorical features we wish to consider.
cat_attribs = ['CODE_GENDER',
'FLAG_OWN_REALTY','FLAG_OWN_CAR','NAME_CONTRACT_TYPE',
               'NAME_EDUCATION_TYPE','OCCUPATION_TYPE','NAME_INCOME_TYPE']

# Notice handle_unknown="ignore" in OHE which ignore values from the
validation/test that
# do NOT occur in the training set
cat_pipeline = Pipeline([
        ('selector', DataFrameSelector(cat_attribs)),
        ('imputer', SimpleImputer(strategy='most_frequent')),
        ('ohe', OneHotEncoder(sparse=False, handle_unknown="ignore"))
    ])
```

In [ ]:
```
cat_attribs =[]
```

In [ ]:
```
gc.collect()
```

Out[ ]:
```
0
```

Create Data Preparation Pipeline

With Feature union, combine numerical and categorical Pipeline together to prepare for Data pipeline

In [ ]:
```
data_prep_pipeline = FeatureUnion(transformer_list=[
        ("num_pipeline", num_pipeline),
    #    ("cat_pipeline", cat_pipeline),
    ])
```

Selected Features

In [ ]:
```
selected_features = num_attribs
tot_features = f"{len(selected_features)}:   Num:{len(num_attribs)},    Cat:{len(cat_
#Total Feature selected for processing
tot_features
```

Out[ ]:
```
'132:   Num:132,    Cat:0'
```

In [ ]:
```
gc.collect()
```

Out[ ]:
```
0
```

Evaluation metrics

Since HCDR is a Classification task, we are going to use the following metrics to measure the Model performance

In [ ]:
```
def pct(x):
    return round(100*x,3)
```

Define dataframe with all metrics included

In [ ]:
```
#del expLog
```

In [ ]:
```
try:
    expLog
except NameError:
    expLog = pd.DataFrame(columns=["exp_name",
                                   "Train Acc",
                                   "Valid Acc",
                                   "Test  Acc",
                                   "Train AUC",
                                   "Valid AUC",
                                   "Test  AUC",
                                   "Train F1 Score",
                                   "Valid F1 Score",
                                   "Test F1 Score",
                                   "Train Log Loss",
                                   "Valid Log Loss",
                                   "Test Log Loss",
                                   "P Score",
                                   "Train Time",
                                   "Valid Time",
                                   "Test Time",
                                   "Description"
                                  ])
```

In [ ]:
```
# roc curve, precision recall curve for each model
fprs, tprs, precisions, recalls, names, scores, cvscores, pvalues, accuracy, cnfmatri
features_list, final_best_clf,results = {}, {},[]
```

Accuracy Score

This metric describes the fraction of correctly classified samples. In SKLearn, it can be modified to return solely the number of correct samples.Accuracy is the default scoring method for both logistic regression and k-Nearest Neighbors in scikit-learn.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Precision

The precision is the ratio of true positives over the total number of predicted positives.

$$\frac{\text{True Positives (TP)}}{\text{True Positives (TP)}+\text{False Positives (FP)}}$$

Recall

The recall is the ratio of true positives over the true positives and false negatives. Recall is assessing the ability of the classifier to find all the positive samples. The best value is 1 and the worst value is 0

$$\frac{\text{True Positives (TP)}}{\text{True Positives (TP)}+\text{False Negatives (FN)}}$$

In [ ]:

```
from sklearn.metrics import precision_recall_curve, PrecisionRecallDisplay
import matplotlib.pyplot as plt

def precision_recall_cust(model, X_train, y_train, X_test, y_test, X_valid, y_valid,
    # Calculate precision-recall manually for test set
    precision, recall, threshold = precision_recall_curve(y_test, model.predict_proba
    precisions.append(precision)
    recalls.append(recall)

    # Plot precision-recall curves for train, test, valid
    display_train = PrecisionRecallDisplay.from_estimator(model, X_train, y_train, na
    display_test = PrecisionRecallDisplay.from_estimator(model, X_test, y_test, name=
    display_valid = PrecisionRecallDisplay.from_estimator(model, X_valid, y_valid, na

    # Set title and show
    display_valid.ax_.set_title("Precision-Recall Curve Comparison - " + name)
    display_valid.ax_.legend(bbox_to_anchor=(1.04, 1), loc="upper left", borderaxespa
    plt.tight_layout()
    plt.show()

    return precisions, recalls
```

F1 score

The F1 score is a metric that has a value of 0 - 1, with 1 being the best value. The F1 score is a weighted average of the precision and recall, with the contributions of precision and recall are the same

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Confusion Matrix

The confusion matrix, in this case for a binary classification, is a 2x2 matrix that contains the count of the true positives, false positives, true negatives, and false negatives.

In [ ]:

```
def confusion_matrix_def(model,X_train,y_train,X_test, y_test, X_valid, y_valid,cnfma
  #Prediction
  preds_test = model.predict(X_test)
  preds_train = model.predict(X_train)
  preds_valid = model.predict(X_valid)

  cm_train = confusion_matrix(y_train, preds_train).astype(np.float32)
  #print(cm_train)
  cm_train /= cm_train.sum(axis=1)[:, np.newaxis]

  cm_test = confusion_matrix(y_test, preds_test).astype(np.float32)
  #print(cm_test)
  cm_test /= cm_test.sum(axis=1)[:, np.newaxis]

  cm_valid = confusion_matrix(y_valid, preds_valid).astype(np.float32)
  cm_valid /= cm_valid.sum(axis=1)[:, np.newaxis]

  plt.figure(figsize=(16, 4))
  #plt.subplots(1,3,figsize=(12,4))

  plt.subplot(131)
  g = sns.heatmap(cm_train, vmin=0, vmax=1, annot=True, cmap="Reds")
  plt.xlabel("Predicted", fontsize=14)
  plt.ylabel("True", fontsize=14)
  g.set(xticklabels=class_labels, yticklabels=class_labels)
  plt.title("Train", fontsize=14)

  plt.subplot(132)
  g = sns.heatmap(cm_valid, vmin=0, vmax=1, annot=True, cmap="Reds")
  plt.xlabel("Predicted", fontsize=14)
  plt.ylabel("True", fontsize=14)
  g.set(xticklabels=class_labels, yticklabels=class_labels)
  plt.title("Validation set", fontsize=14);

  plt.subplot(133)
  g = sns.heatmap(cm_test, vmin=0, vmax=1, annot=True, cmap="Reds")
  plt.xlabel("Predicted", fontsize=14)
  plt.ylabel("True", fontsize=14)
  g.set(xticklabels=class_labels, yticklabels=class_labels)
  plt.title("Test", fontsize=14);
  cnfmatrix.append(cm_test)
```

```
  return cnfmatrix
```

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

AUC (Area under ROC curve)

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: ▪ True Positive Rate ▪ False Positive Rate

AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1).

AUC is desirable for the following two reasons:

1. AUC is scale-invariant. It measures how well predictions are ranked, rather than their absolute values.
2. AUC is classification-threshold-invariant. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

$$\text{AUC} = \int_0^1 \text{ROC}(t)\, dt$$

In [ ]:

```
from sklearn.metrics import RocCurveDisplay

def roc_curve_cust(model, X_train, y_train, X_test, y_test, X_valid, y_valid, fprs, t
    # Fit model if not already fit (optional)
    # model.fit(X_train, y_train)

    # Predict and calculate ROC AUC (for plotting data manually if needed)
    fpr, tpr, _ = roc_curve(y_valid, model.predict_proba(X_valid)[:, 1])
    fprs.append(fpr)
    tprs.append(tpr)

    # Plot ROC curves
    roc_display_train = RocCurveDisplay.from_estimator(model, X_train, y_train, name=
    roc_display_test = RocCurveDisplay.from_estimator(model, X_test, y_test, name="Te
    roc_display_valid = RocCurveDisplay.from_estimator(model, X_valid, y_valid, name=

    return fprs, tprs
```

Binary cross-entropy loss (CXE)

CXE measures the performance of a classification model whose output is a probability value between 0 and 1. CXE increases as the predicted probability diverges from the actual label. Therefore, we choose a parameter, which would minimize the binary CXE loss function.

The log loss formula for the binary case is as follows :

$$-\frac{1}{m}\sum_{i=1}^{m}(y_i \cdot \log\ (p_i)\ +\ (1-y_i)\cdot \log(1-p_i))$$

- y_i: the label for i_{th} observation
- m: sample size
- p_i: predicted probability of the point being in the label($y = 1$) for i_{th} observation

p-value

p-value is the probability of obtaining test results at least as extreme as the results actually observed, under the assumption that the null hypothesis is correct. A very small p-value means that such an extreme observed outcome would be very unlikely under the null hypothesis.

We will compare the classifiers with the baseline untuned model by conducting two-tailed hypothesis test.

Null Hypothesis, H0: There is no significant difference between the two machine learning pipelines. Alternate Hypothesis, HA: The two machine learning pipelines are different. A p-value less than or equal to the significance level is considered statistically significant.

P-value

p-value is the probability of obtaining test results at least as extreme as the results actually observed, under the assumption that the null hypothesis is correct. A very small p-value means that such an extreme observed outcome would be very unlikely under the null hypothesis.

We will compare the classifiers with the baseline untuned model by conducting two-tailed hypothesis test.

Null Hypothesis, H0: There is no significant difference between the two machine learning pipelines. Alternate Hypothesis, HA: The two machine learning pipelines are different. A p-value less than or equal to the significance level is considered statistically significant.

$$\text{P-value} = P(Z \geq |z|)$$

In [ ]:

```
metrics = {'accuracy': make_scorer(accuracy_score),
           'roc_auc': 'roc_auc',
           'f1': make_scorer(f1_score),
           'log_loss': make_scorer(log_loss)
          }
```

Baseline model with Imbalanced Dataset

Data Leakage

Phase 3 marked a critical advancement in feature engineering, driven by a deeper understanding of the data that enabled more precise dataset refinement and mitigation of data leakage. The "TARGET" variable was excluded from the train-test process, with the test set maintained separately from the merged training data. Preprocessing was performed solely on the merged training set. Multicollinearity was addressed independently as part of preprocessing, and to further prevent data leakage, feature selection was conducted exclusively within the modeling pipeline. The techniques applied included Recursive Feature Elimination (RFE), SelectKBest with mutual information for classification, and Variance Threshold.

Create Train and Test Datasets

In [ ]:
```
for col in selected_features:
  if col not in  train_dataset.columns:
    selected_features.remove(col)
```

In [ ]:
```
# Split Sample to feed the pipeline and it will result in a new dataset that is (1 /
splits = 75

# Train Test split percentage
subsample_rate = 0.3

finaldf = np.array_split(train_dataset, splits)
X_train = finaldf[0][selected_features]
y_train = finaldf[0]['TARGET']
X_kaggle_test= X_kaggle_test[selected_features]

## split part of data
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, stratify=y_trai
                                                    test_size=subsample_rate, random_

X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train,stratify=y_tra

print(f"X train           shape: {X_train.shape}")
print(f"X validation      shape: {X_valid.shape}")
print(f"X test            shape: {X_test.shape}")
print(f"X kaggle_test     shape: {X_kaggle_test.shape}")
```

```
X train           shape: (2439, 132)
X validation      shape: (431, 132)
X test            shape: (1231, 132)
X kaggle_test     shape: (48744, 132)
```

Define pipeline

Logistic regression model is used as a baseline Model, since it's easy to implement yet provides great efficiency. Training a logistic regression model doesn't require high computation power. We also tuned the regularization, tolerance, and C hyper parameters for the Logistic regression model and compared the results with the baseline model. We used 15 fold cross fold validation with hyperparameters to tune the model and apply GridSearchCV function in Sklearn.

In [ ]:
```
%%time
np.random.seed(42)
full_pipeline_with_predictor = Pipeline([
        ("preparation", data_prep_pipeline),
        ("linear", LogisticRegression())
    ])
```

```
CPU times: user 68 µs, sys: 0 ns, total: 68 µs
Wall time: 75.8 µs
```

Perform cross-fold validation and Train the model

Split the training data to 15 fold to perform Crossfold validation

In [ ]:
```
cvSplits = ShuffleSplit(n_splits=5, test_size=0.3, random_state=0)
```

In [ ]:
```
X_train.head(5)
gc.collect()
```

Out[ ]:
```
13
```

In [ ]:
```
start = time()
model = full_pipeline_with_predictor.fit(X_train, y_train)
np.random.seed(42)

# Set up cross validation scores
logit_scores = cross_validate(model, X_train, y_train,cv=cvSplits,scoring=metrics, re
train_time = np.round(time() - start, 4)

# Time and score valid predictions
start = time()
logit_score_valid  = full_pipeline_with_predictor.score(X_valid, y_valid)
valid_time = np.round(time() - start, 4)

# Time and score test predictions
start = time()
logit_score_test  = full_pipeline_with_predictor.score(X_test, y_test)
test_time = np.round(time() - start, 4)
```

Calculate metrics

In [ ]:
```
exp_name = f"Baseline_{len(selected_features)}_features"
expLog.loc[len(expLog)] = [f"{exp_name}"] + list(np.round(
               [logit_scores['train_accuracy'].mean(),
                logit_scores['test_accuracy'].mean(),
                accuracy_score(y_test, model.predict(X_test)),
                logit_scores['train_roc_auc'].mean(),
                logit_scores['test_roc_auc'].mean(),
                roc_auc_score(y_test, model.predict_proba(X_test)[:, 1]),
                logit_scores['train_f1'].mean(),
                logit_scores['test_f1'].mean(),
```

```
                f1_score(y_test, model.predict(X_test)),
                logit_scores['train_log_loss'].mean(),
                logit_scores['test_log_loss'].mean(),
                log_loss(y_test, model.predict(X_test)),0 ],4)) \
                + [train_time, logit_scores['score_time'].mean(), test_time] + [f"Imb
expLog
```

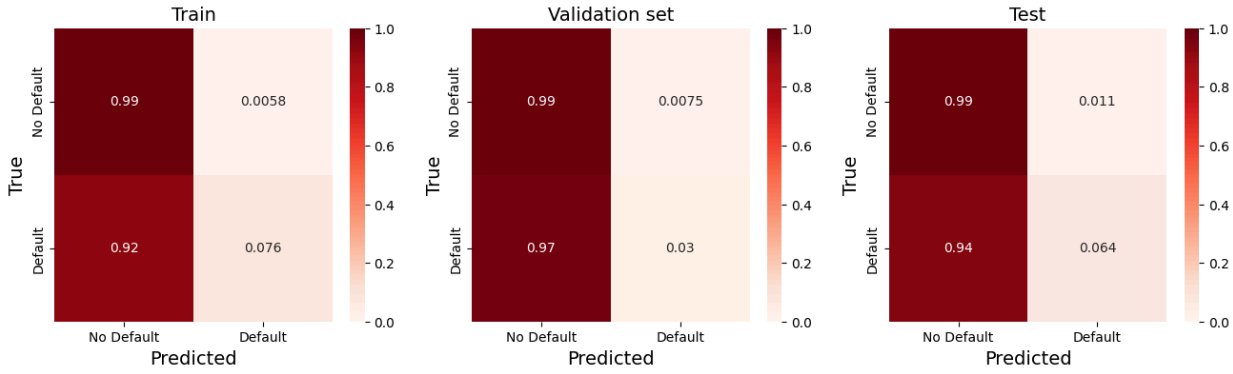
Out[ ]:

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Train Log Loss
0	Baseline_132_features	0.9289	0.912	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5634

Confusion matrix

In [ ]:

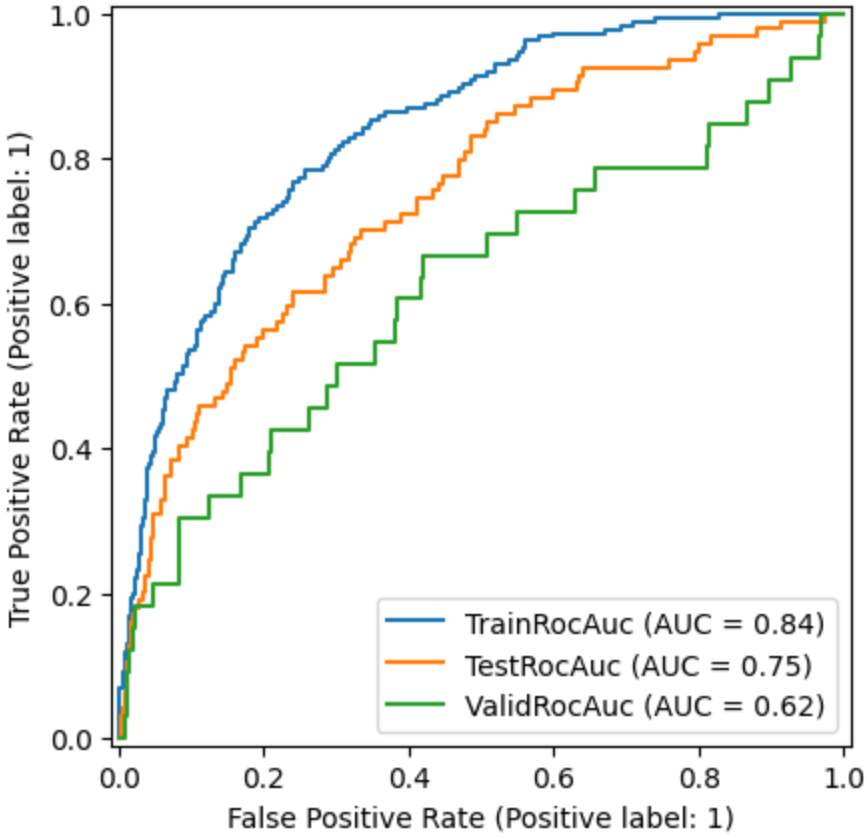
```
# Create confusion matrix for baseline model
_=confusion_matrix_def(model,X_train,y_train,X_test,y_test,X_valid, y_valid,cnfmatrix
```



AUC (Area under ROC curve)

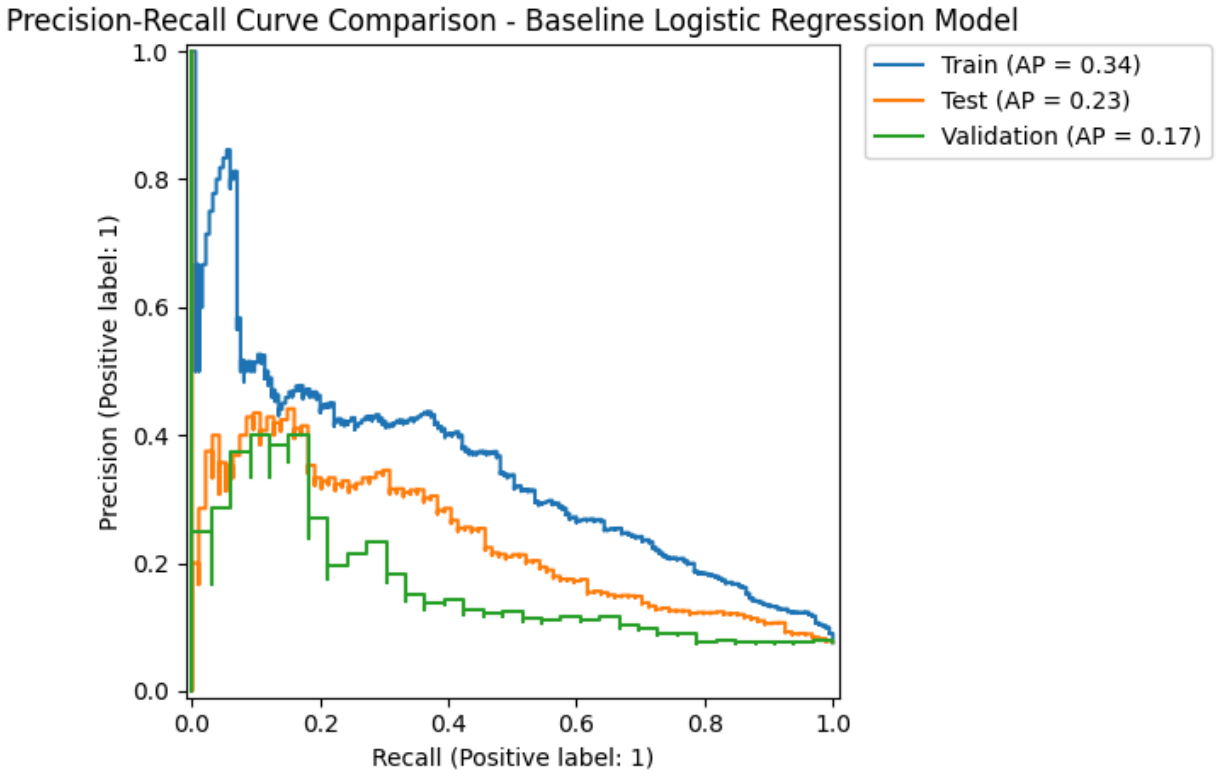
In [ ]:

```
_,_=roc_curve_cust(model,X_train,y_train,X_test, y_test,X_valid, y_valid,fprs,tprs,"B
```



Precision Recall Curve

In [ ]:
```
_,_=precision_recall_cust(model,X_train,y_train,X_test, y_test,X_valid, y_valid,preci
```



In [ ]:
```
gc.collect()
```

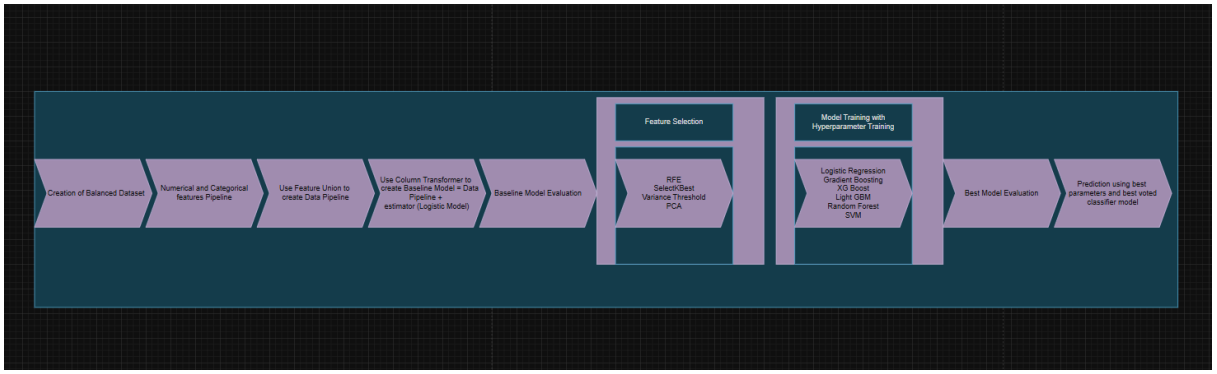
Out[ ]:
```
20230
```

Tune Basline model with grid search & RFE

Various Classification algorithms were used to compare with the best model. Following metrics were used to find the best model

- Cross fold Train Accuracy
- Test Accuracy
- p-value
- Train ROC_AUC_Score
- Test ROC_AUC_Score
- Train F1_Score
- Test F1_Score
- Train LogLoss
- Test LogLoss
- Train Time
- Test Time
- Confusion matrix

We implemented the Logistic regression model as the baseline model, which didn't require high computation power and was easy to implement, in addition we implemented KNN and tuned logistic models with balanced dataset to improve our model predictiveness. Our objective in current phase s to explore various classification models which would improve our prediction. Our primary focus is on boosting algorithms which are said to be highly efficient and moderately quicker. As shown in the diagram below is the modelling pipeline for current phase. We primarily experimented with Gradient Boosting, XGBoost, Light BGM, RandomForest and SVM.



Recursive Feature Elimination RFE is a wrapper-type feature selection algorithm. A different machine learning algorithm is given and used in the core of the method, is wrapped by RFE, and used to help select features. We have chosen this model in contrast to filter-based feature selections that score each feature and select those features with the largest (or smallest) score.

Below is the reason for choosing the mentioned models.

1. Gradient Boosting provides a better predictive model by forming an ensemble of weak predictors.
2. XGBoost is one of the quickest implementations of gradient boosted trees. XGBoost is designed to handle missing values internally. This is helpful because there are many, many hyperparameters to tune.
3. LightGBM in many cases provides results which are more effective and faster than XGBoost with lesser memory usage. It splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise.
4. Random Forest is a tree-based machine learning algorithm that combines the output of multiple decision trees for making decisions. For each tree only a random subsample of the available features is selected for building the tree. Random Forest uses decision trees, which are more prone to overfitting.
5. SVM performs similar to logistic regression when linear separation and performs well with non-linear boundaries depending on the kernel used. SVM is susceptible to overfitting/training issues depending on the kernel. A more complex kernel would overfit the model.

Gradient Boosting implementations have no regularisation like XGBoost, therefore it helps to reduce overfitting. But boosting algorithms can overfit if the number of trees is very large. We did two submission in Kaggle, one using Voting Classifier and the other one with best classifier i.e. XGBoost. A Voting Classifier is a machine learning model that trains on an ensemble of various models and predicts an output based on their highest probability of chosen class as the output. We have chosen soft voting instead of hard voting since the soft voting predicts based on the average of all models.

For discussion, Results and conclusions refer to the bottom of this page.

Classifiers

In [ ]:
```
classifiers = [
        [('Logistic Regression', LogisticRegression(solver='saga',random_state=42),"R
        [('Support Vector', SVC(random_state=42,probability=True),"SVM")],
        [('Gradient Boosting', GradientBoostingClassifier(warm_start=True, random_sta
        [('XGBoost', XGBClassifier(random_state=42),"RFE")],
        [('Light GBM', LGBMClassifier(boosting_type='gbdt', random_state=42),"RFE")],
        [('RandomForest', RandomForestClassifier(random_state=42),"RFE")]
    ]
```

Hyper-parameters for different models

In [ ]:
```
# Arrange grid search parameters for each classifier
params_grid = {
        'Logistic Regression': {
            'penalty': ('l1', 'l2','elasticnet'),
            'tol': (0.0001, 0.00001),
```

```
            'C': (10, 1, 0.1, 0.01),
        }
    ,
        'Support Vector' : {
            'kernel': ('rbf','poly'),
            'degree': (4, 5),
            'C': ( 0.001, 0.01),   #Low C - allow for misclassification
            'gamma':(0.01,0.1,1)  #Low gamma - high variance and low bias
        }
    ,
    'Gradient Boosting':  {
            'max_depth': [5,10], # Lower helps with overfitting
            'max_features': [10,15],
            'validation_fraction': [0.2],
            'n_iter_no_change': [10],
            'tol': [0.01,0.0001],
            'n_estimators':[1000],
            'subsample' : [0.8],             #fraction of observations to be randomly
    #        'min_samples_split' : [5], # Must have 'x' number of samples to split (D
            'min_samples_leaf' : [3,5],        # (Default = 1) minimum number of samp
        },
        'XGBoost':  {
            'max_depth': [3,5], # Lower helps with overfitting
            'n_estimators':[300,500],
            'learning_rate': [0.01,0.1],
#            'objective': ['binary:logistic'],
#            'eval_metric': ['auc'],
            'eta' : [0.01,0.1],
            'colsample_bytree' : [0.2,0.5],
        },
        'Light GBM':  {
            'max_depth': [2,5],  # Lower helps with overfitting
            'num_leaves': [5,10], # Equivalent to max depth
            'n_estimators':[1000,5000],
            'learning_rate': [0.01,0.1],
 #           'reg_alpha': [0.1,0.01,1],
 #           'reg_lambda': [0.1,0.01,1],
            'boosting_type':['goss','dart'],
 #           'metric': ['auc'],
 #           'objective':['binary'],
            'max_bin' : [100,200],  #Setting it to high values has similar effect as
        },                          #small numbers reduces accuracy but runs faster

        'RandomForest':  {
            'max_depth': [5,10],
            'max_features': [15,20],
            'min_samples_split': [5, 10],
            'min_samples_leaf': [3, 5],
            'bootstrap': [True],
            'n_estimators':[1000]},
    }
```

In [ ]:

```
# Set feature selection settings

feature_selection_steps=0.5
# Number of features used
features_used=len(selected_features)
```

Conduct Grid Search

In [ ]:

```
results.append(logit_scores['train_accuracy'])
names = ['Baseline LR']
def ConductGridSearch(in_classifiers,cnfmatrix,fprs,tprs,precisions,recalls):
    for (name, classifier,feature_sel) in in_classifiers:
            # Print classifier and parameters
            print('****** START', name,'*****')
            parameters = params_grid[name]
            print("Parameters:")
            for p in sorted(parameters.keys()):
                print("\t"+str(p)+": "+ str(parameters[p]))

            # generate the pipeline based on the feature selection method
            if feature_sel == "SVM":
                full_pipeline_with_predictor = Pipeline([
                ("preparation", data_prep_pipeline),
            #    ("PCA",PCA(0.95)),
            #    ('RFE', RFE(estimator=classifier, n_features_to_select=features_used
                ("predictor", classifier)
                ])
            else:
                full_pipeline_with_predictor = Pipeline([
                ("preparation", data_prep_pipeline),
                ('RFE', RFE(estimator=classifier, n_features_to_select=features_used,
                ("predictor", classifier)
                ])

            # Execute the grid search
            params = {}
            for p in parameters.keys():
                pipe_key = 'predictor__'+str(p)
                params[pipe_key] = parameters[p]
            grid_search = GridSearchCV(full_pipeline_with_predictor, params, cv=cvSpl
                                       n_jobs=-1,verbose=1)
            grid_search.fit(X_train, y_train)

            # Best estimator score
            best_train = pct(grid_search.best_score_)

            # Best train scores
            print("Cross validation with best estimator")
            best_train_scores = cross_validate(grid_search.best_estimator_, X_train,
                                               return_train_score=True, n_jobs=-1)

            #get all scores
            best_train_accuracy = np.round(best_train_scores['train_accuracy'].mean()
            best_train_f1 = np.round(best_train_scores['train_f1'].mean(),4)
            best_train_logloss = np.round(best_train_scores['train_log_loss'].mean(),
            best_train_roc_auc = np.round(best_train_scores['train_roc_auc'].mean(),4

            valid_time = np.round(best_train_scores['score_time'].mean(),4)
            best_valid_accuracy = np.round(best_train_scores['test_accuracy'].mean(),
            best_valid_f1 = np.round(best_train_scores['test_f1'].mean(),4)
            best_valid_logloss = np.round(best_train_scores['test_log_loss'].mean(),4
            best_valid_roc_auc = np.round(best_train_scores['test_roc_auc'].mean(),4)

            #append all results
```

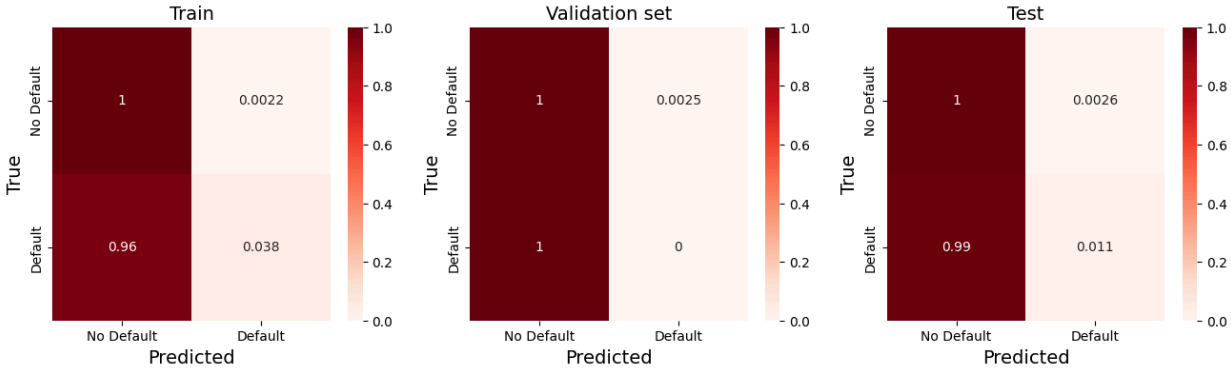
```
results.append(best_train_scores['train_accuracy'])
names.append(name)

# Conduct t-test with baseline logit (control) and best estimator (experi
(t_stat, p_value) = stats.ttest_rel(logit_scores['train_roc_auc'], best_t

#test and Prediction with whole data
# Best estimator fitting time
print("Fit and Prediction with best estimator")
start = time()
model = grid_search.best_estimator_.fit(X_train, y_train)
train_time = round(time() - start, 4)

# Best estimator prediction time
start = time()
y_test_pred = model.predict(X_test)
test_time = round(time() - start, 4)
scores.append(roc_auc_score(y_test, model.predict_proba(X_test)[:, 1]))
accuracy.append(accuracy_score(y_test, y_test_pred))

# Create confusion matrix for the best model
cnfmatrix = confusion_matrix_def(model,X_train,y_train,X_test,y_test,X_va

# Create AUC ROC curve
fprs,tprs = roc_curve_cust(model,X_train,y_train,X_test, y_test,X_valid,

#Create Precision recall curve
precisions,recalls = precision_recall_cust(model,X_train,y_train,X_test,

#Best Model
final_best_clf[name]=pd.DataFrame([{'label': grid_search.best_estimator_.
                                   'predictor': grid_search.best_estimator_.n
#Feature importance
feature_name = num_attribs #+ cat_attribs
feature_list = feature_name
if feature_sel == "RFE":
#    features_list[name]=pd.DataFrame({'feature_name': feature_list,
#                                      'feature_importance': grid_sear
#                                  'feature_importance': grid_search.best_esti
   # print(grid_search.best_estimator_.named_steps['preparation'].get_fea
   # print(len(grid_search.best_estimator_.named_steps['preparation'].get
   # print(len(feature_list),feature_list)
   # print(len(grid_search.best_estimator_.named_steps['RFE'].ranking_))
#          grid_search.best_estimator_.named_steps['RFE'].ranking_)
    features_list[name]=pd.DataFrame({'feature_name': feature_list,
                                  'feature_importance': grid_search.best_estim
# Collect the best parameters found by the grid search
print("Best Parameters:")
best_parameters = grid_search.best_estimator_.get_params()
param_dump = []
for param_name in sorted(params.keys()):
    param_dump.append((param_name, best_parameters[param_name]))
    print("\t"+str(param_name)+": " + str(best_parameters[param_name]))
print("****** FINISH",name," *****")
print("")

# Record the results
exp_name = name
expLog.loc[len(expLog)] = [f"{exp_name}"] + list(np.round(
        [best_train_accuracy,
```

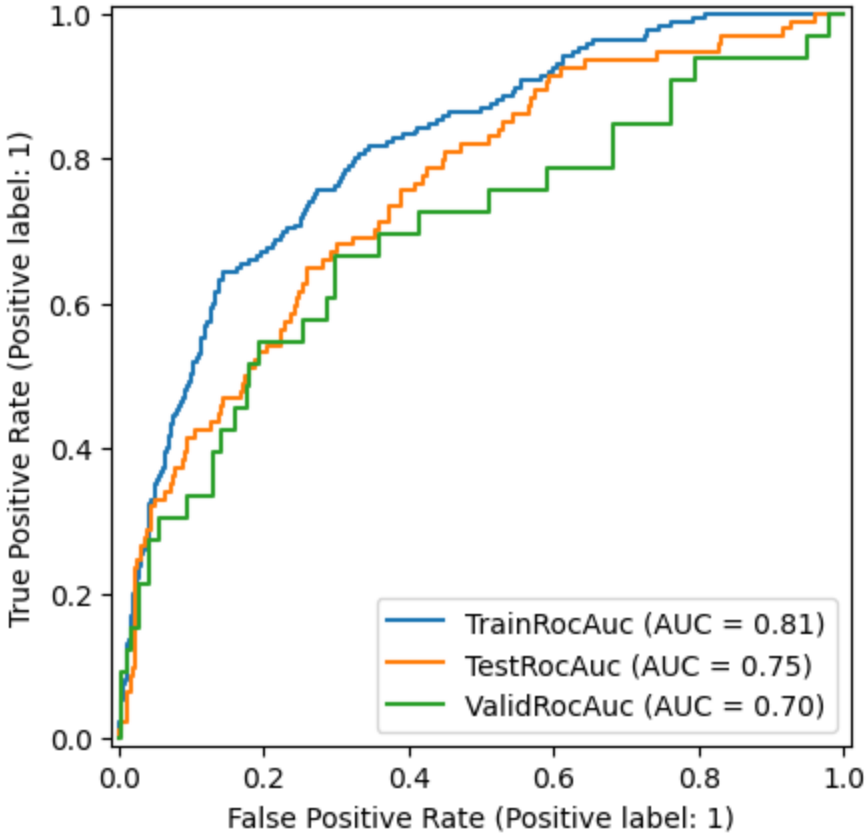
```
#pct(accuracy_score(y_valid, model.predict(X_valid))),
best_valid_accuracy,
accuracy_score(y_test, y_test_pred),
best_train_roc_auc,
best_valid_roc_auc,
#roc_auc_score(y_valid, model.predict_proba(X_valid)[:, 1]),
roc_auc_score(y_test, model.predict_proba(X_test)[:, 1]),
best_train_f1,
best_valid_f1,
f1_score(y_test, y_test_pred),
best_train_logloss,
best_valid_logloss,
log_loss(y_test, y_test_pred),
p_value
],4)) + [train_time,valid_time,test_time] \
+ [json.dumps(param_dump)]
```

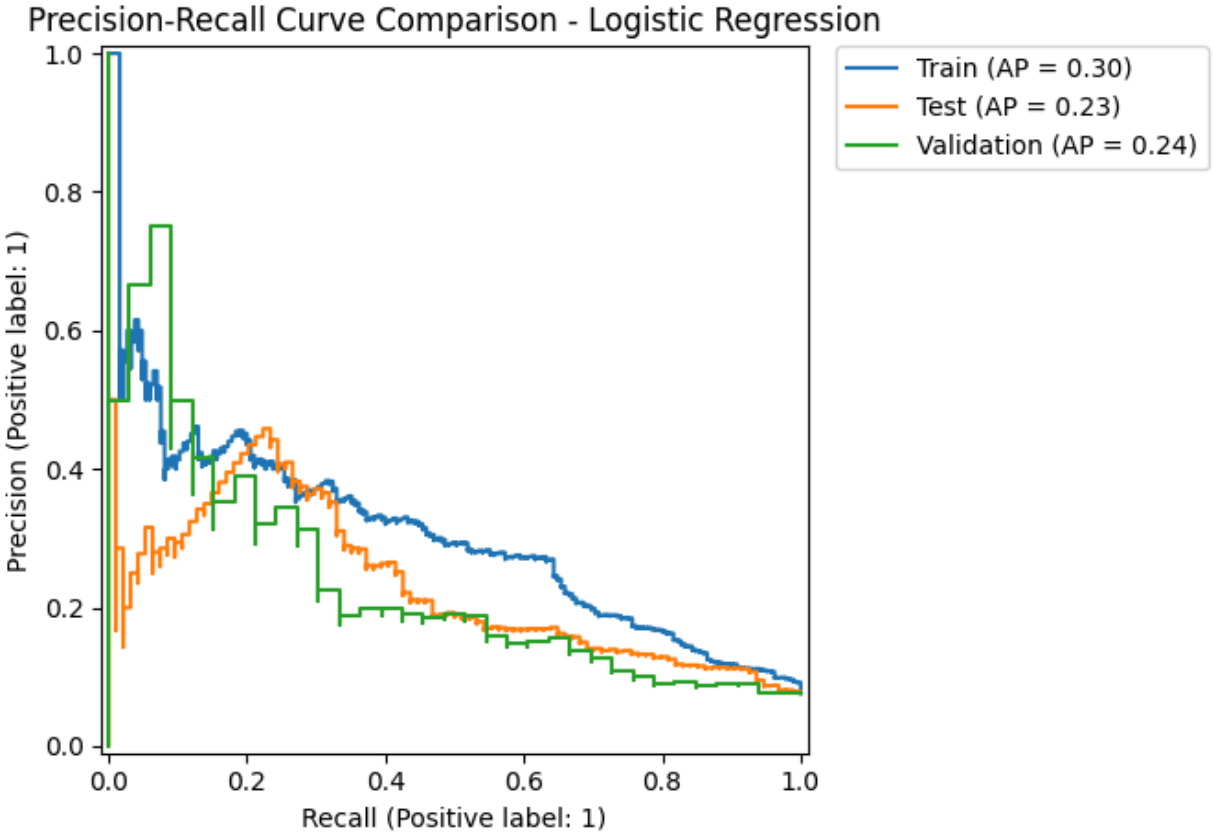
Logistic Regression Model

In [ ]:
```
ConductGridSearch(classifiers[0],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START Logistic Regression *****
Parameters:
        C: (10, 1, 0.1, 0.01)
        penalty: ('l1', 'l2', 'elasticnet')
        tol: (0.0001, 1e-05)
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Cross validation with best estimator
Fit and Prediction with best estimator
```








```
Best Parameters:
        predictor__C: 0.1
        predictor__penalty: l1
        predictor__tol: 0.0001
****** FINISH Logistic Regression  *****
```

In [ ]:
```
gc.collect()
```

Out[ ]:
```
20126
```

In [ ]:
```
expLog
```

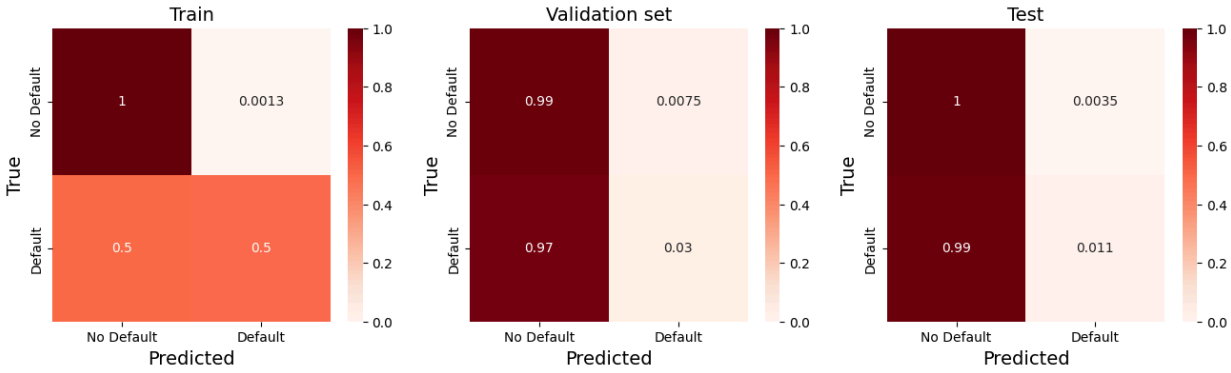
Out[ ]:

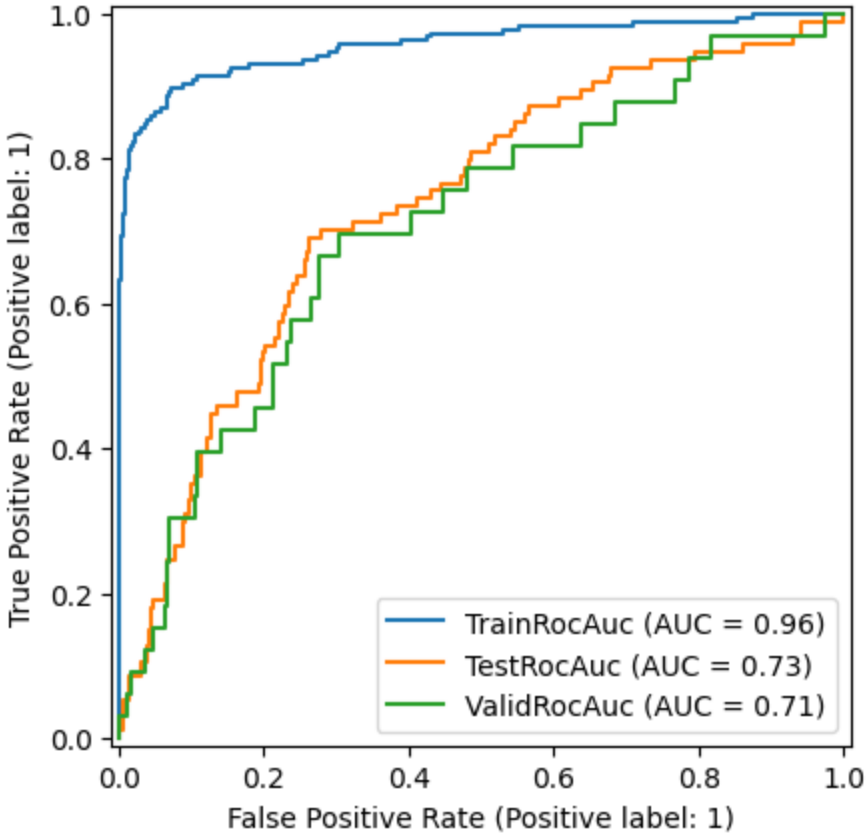
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Train Log Loss	
0	Baseline_132_features	0.9289	0.912	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5634	3
1	Logistic Regression	0.9256	0.923	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6816	2

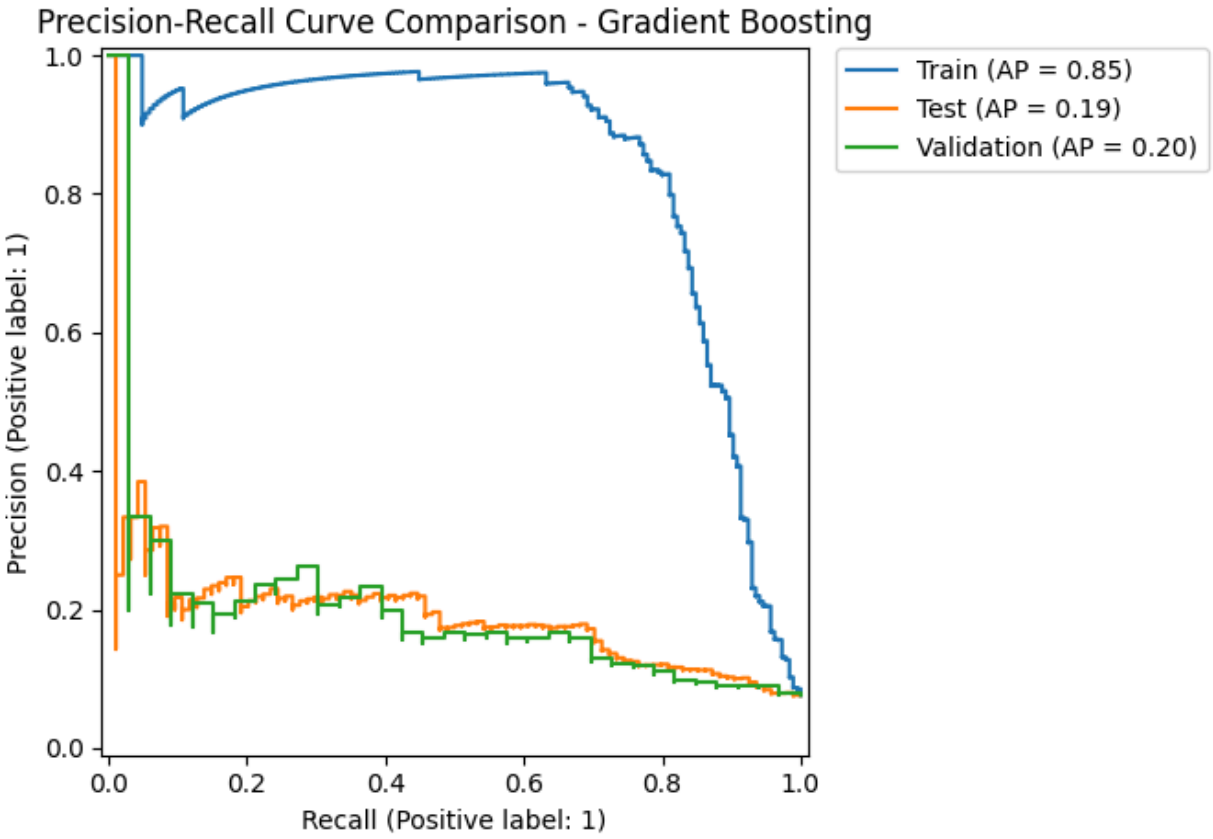
Gradient Boosting

In [ ]:
```
ConductGridSearch(classifiers[2],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START Gradient Boosting *****
Parameters:
        max_depth: [5, 10]
        max_features: [10, 15]
        min_samples_leaf: [3, 5]
        n_estimators: [1000]
        n_iter_no_change: [10]
        subsample: [0.8]
        tol: [0.01, 0.0001]
        validation_fraction: [0.2]
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Cross validation with best estimator
Fit and Prediction with best estimator
```




True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
TrainRocAuc (AUC = 0.96)
TestRocAuc (AUC = 0.73)
ValidRocAuc (AUC = 0.71)


Precision-Recall Curve Comparison - Gradient Boosting
Precision (Positive label: 1)
Recall (Positive label: 1)
Train (AP = 0.85)
Test (AP = 0.19)
Validation (AP = 0.20)

```
Best Parameters:
        predictor__max_depth: 5
        predictor__max_features: 15
        predictor__min_samples_leaf: 3
        predictor__n_estimators: 1000
        predictor__n_iter_no_change: 10
        predictor__subsample: 0.8
        predictor__tol: 0.0001
        predictor__validation_fraction: 0.2
****** FINISH Gradient Boosting  *****
```

In [ ]:
```
gc.collect()
```

Out[ ]:
```
19739
```

In [ ]:
```
expLog
```

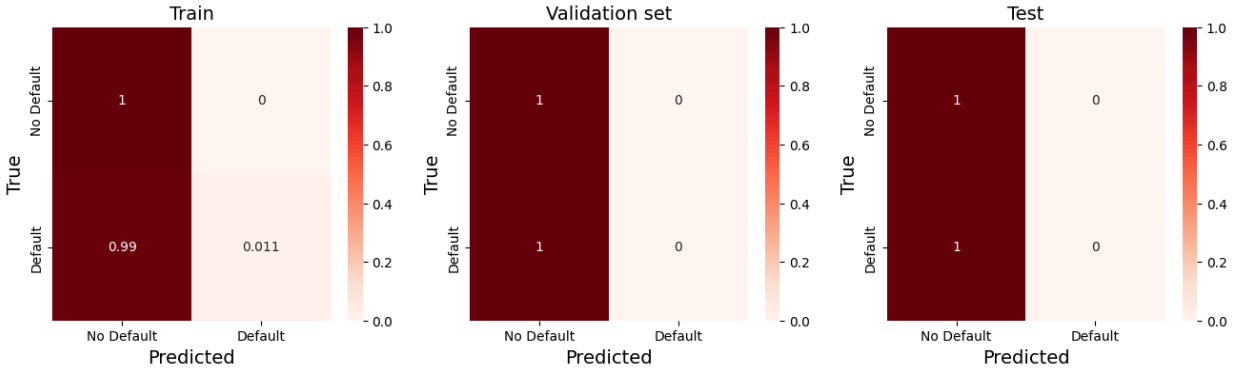
Out[ ]:

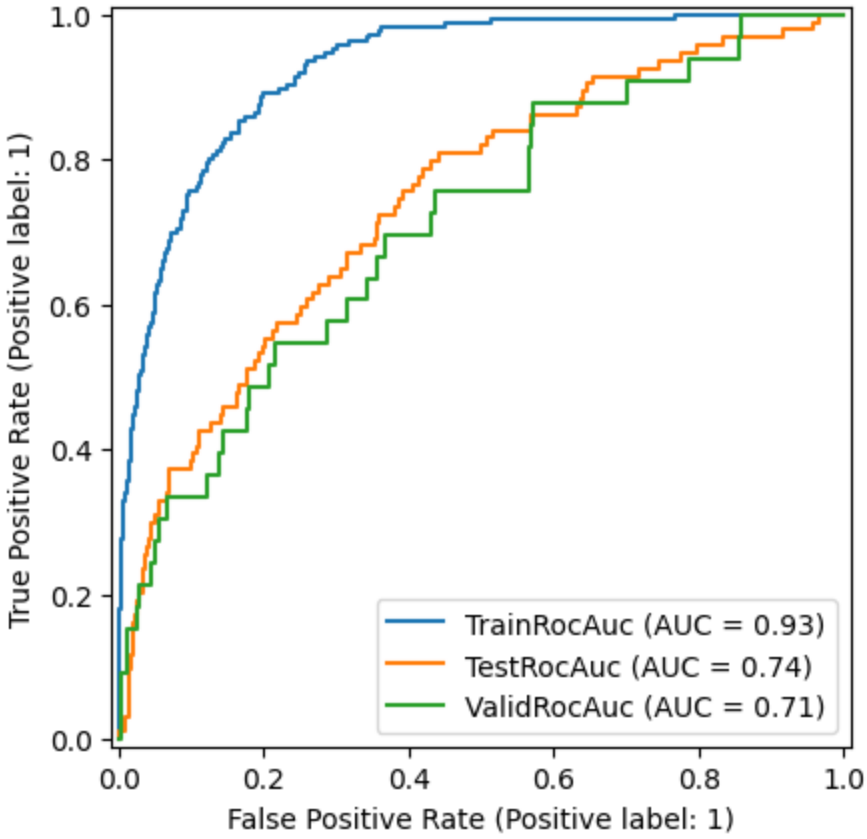
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Train Log Loss
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5634
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6816
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9004

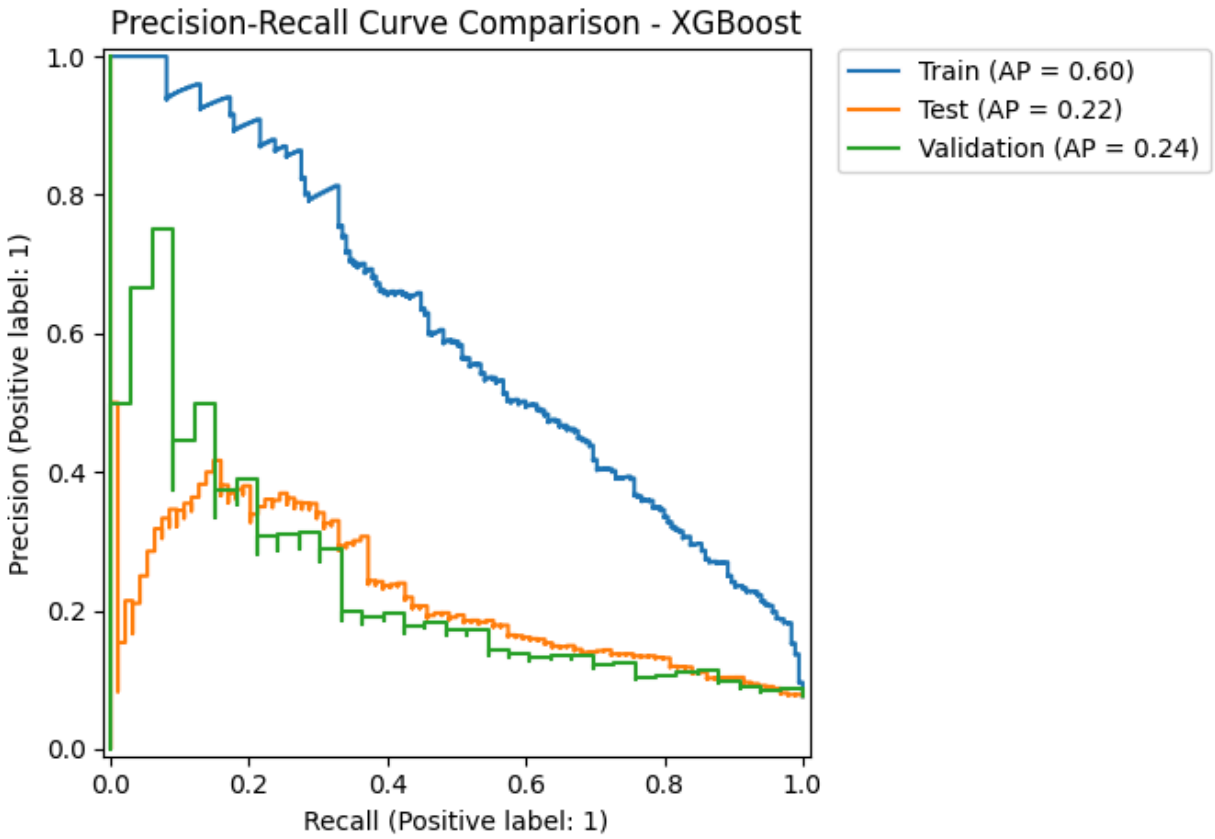
XGBoost

In [ ]:
```
ConductGridSearch(classifiers[3],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START XGBoost *****
Parameters:
        colsample_bytree: [0.2, 0.5]
        eta: [0.01, 0.1]
        learning_rate: [0.01, 0.1]
        max_depth: [3, 5]
        n_estimators: [300, 500]
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Cross validation with best estimator
Fit and Prediction with best estimator
```


Train
Validation set
Test
No Default
Default
True
Predicted
1
0
0.99
0.011


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
TrainRocAuc (AUC = 0.93)
TestRocAuc (AUC = 0.74)
ValidRocAuc (AUC = 0.71)




```
Best Parameters:
        predictor__colsample_bytree: 0.2
        predictor__eta: 0.01
        predictor__learning_rate: 0.01
        predictor__max_depth: 3
        predictor__n_estimators: 300
****** FINISH XGBoost  *****
```

In [ ]:
```
expLog
```

Out[ ]:

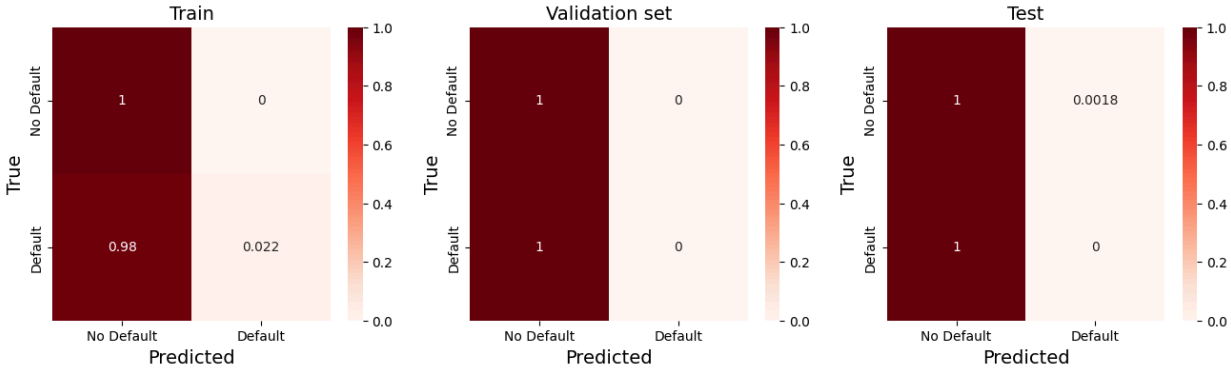
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Train Log Loss
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5634
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6816
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9004
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6394

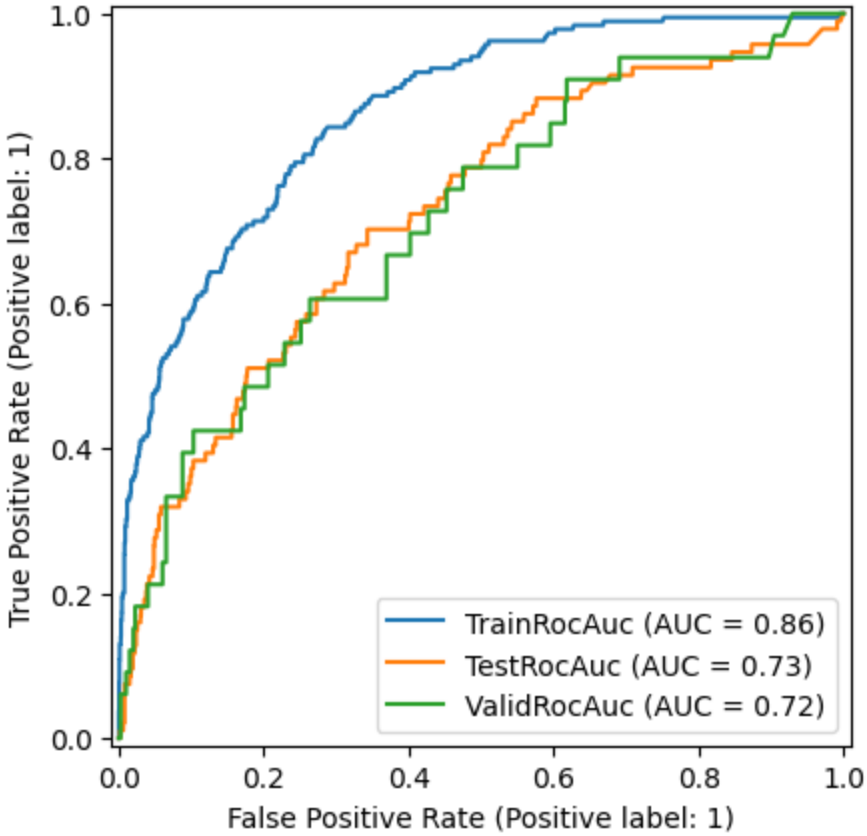
Light GBM

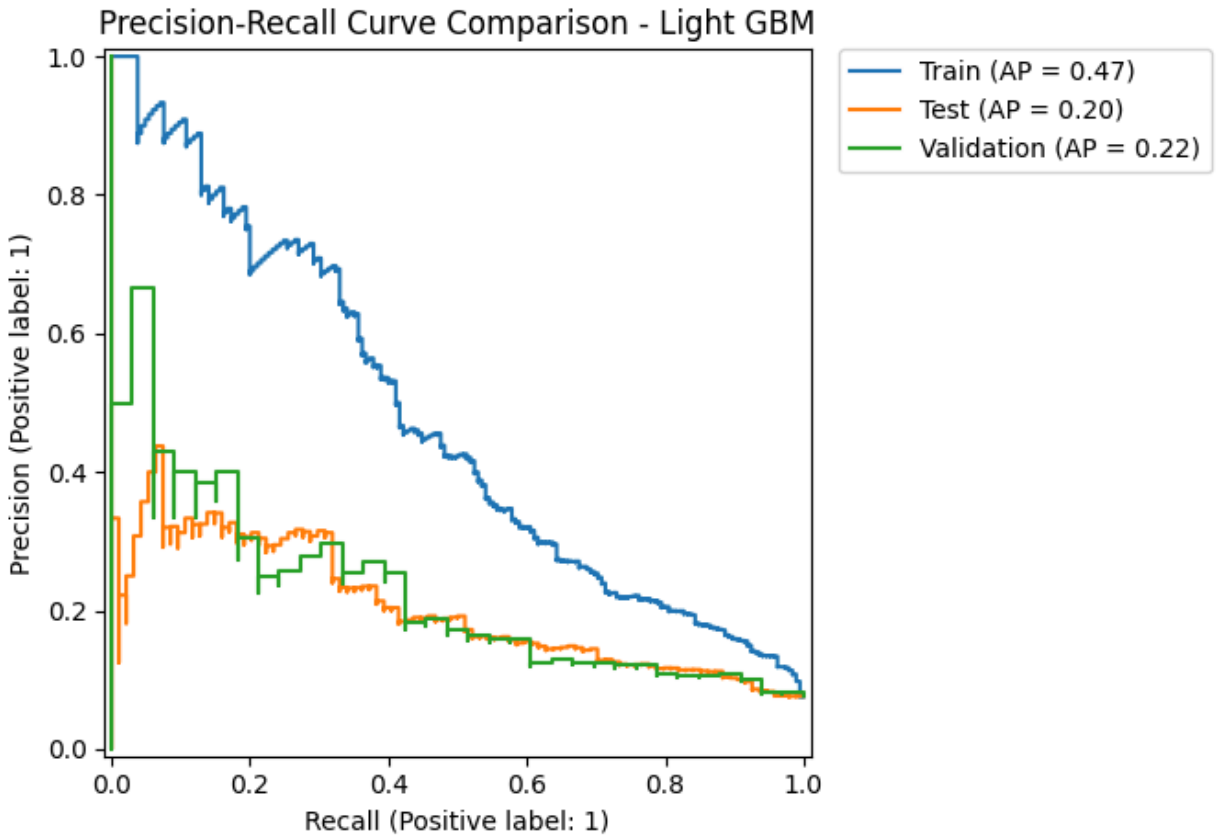
In [ ]:
```
ConductGridSearch(classifiers[4],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START Light GBM *****
Parameters:
        boosting_type: ['goss', 'dart']
        learning_rate: [0.01, 0.1]
        max_bin: [100, 200]
        max_depth: [2, 5]
        n_estimators: [1000, 5000]
        num_leaves: [5, 10]
Fitting 5 folds for each of 64 candidates, totalling 320 fits
[LightGBM] [Info] Number of positive: 185, number of negative: 2254
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was
0.002041 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 11836
[LightGBM] [Info] Number of data points in the train set: 2439, number of used featur
es: 123
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.075851 -> initscore=-2.500106
[LightGBM] [Info] Start training from score -2.500106
[LightGBM] [Info] Number of positive: 185, number of negative: 2254
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was
0.001988 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 11068
[LightGBM] [Info] Number of data points in the train set: 2439, number of used featur
es: 119
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.075851 -> initscore=-2.500106
[LightGBM] [Info] Start training from score -2.500106
[LightGBM] [Info] Number of positive: 185, number of negative: 2254
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was
0.001815 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 5172
[LightGBM] [Info] Number of data points in the train set: 2439, number of used featur
es: 119
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.075851 -> initscore=-2.500106
[LightGBM] [Info] Start training from score -2.500106
Cross validation with best estimator
Fit and Prediction with best estimator
[LightGBM] [Info] Number of positive: 185, number of negative: 2254
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was
0.002076 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 11836
[LightGBM] [Info] Number of data points in the train set: 2439, number of used featur
es: 123
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.075851 -> initscore=-2.500106
[LightGBM] [Info] Start training from score -2.500106
[LightGBM] [Info] Number of positive: 185, number of negative: 2254
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was
0.001874 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 11068
[LightGBM] [Info] Number of data points in the train set: 2439, number of used featur
es: 119
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.075851 -> initscore=-2.500106
[LightGBM] [Info] Start training from score -2.500106
[LightGBM] [Info] Number of positive: 185, number of negative: 2254
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was
0.001808 seconds.
You can set `force_col_wise=true` to remove the overhead.
```

```
[LightGBM] [Info] Total Bins 5172
[LightGBM] [Info] Number of data points in the train set: 2439, number of used featur
es: 119
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.075851 -> initscore=-2.500106
[LightGBM] [Info] Start training from score -2.500106
```








```
Best Parameters:
        predictor__boosting_type: dart
        predictor__learning_rate: 0.01
        predictor__max_bin: 100
        predictor__max_depth: 5
        predictor__n_estimators: 1000
        predictor__num_leaves: 5
****** FINISH Light GBM  *****
```

In [ ]:
```
expLog
```

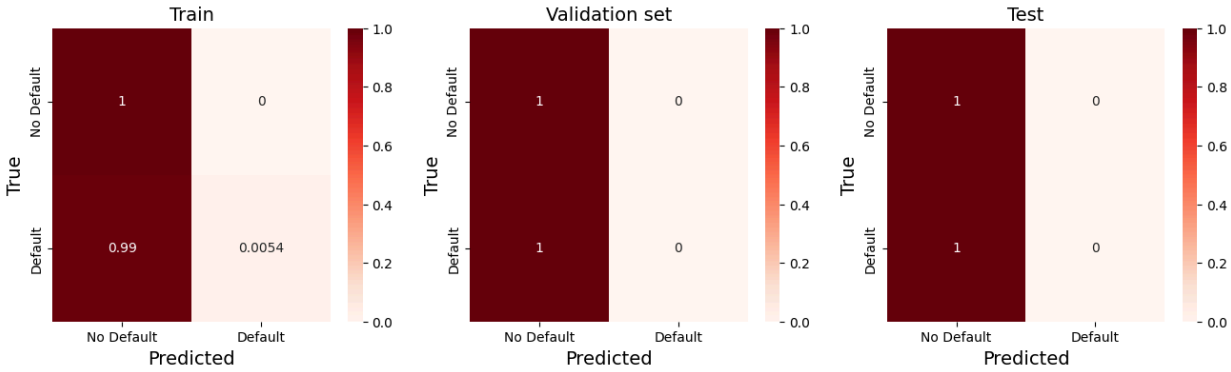
Out[ ]:

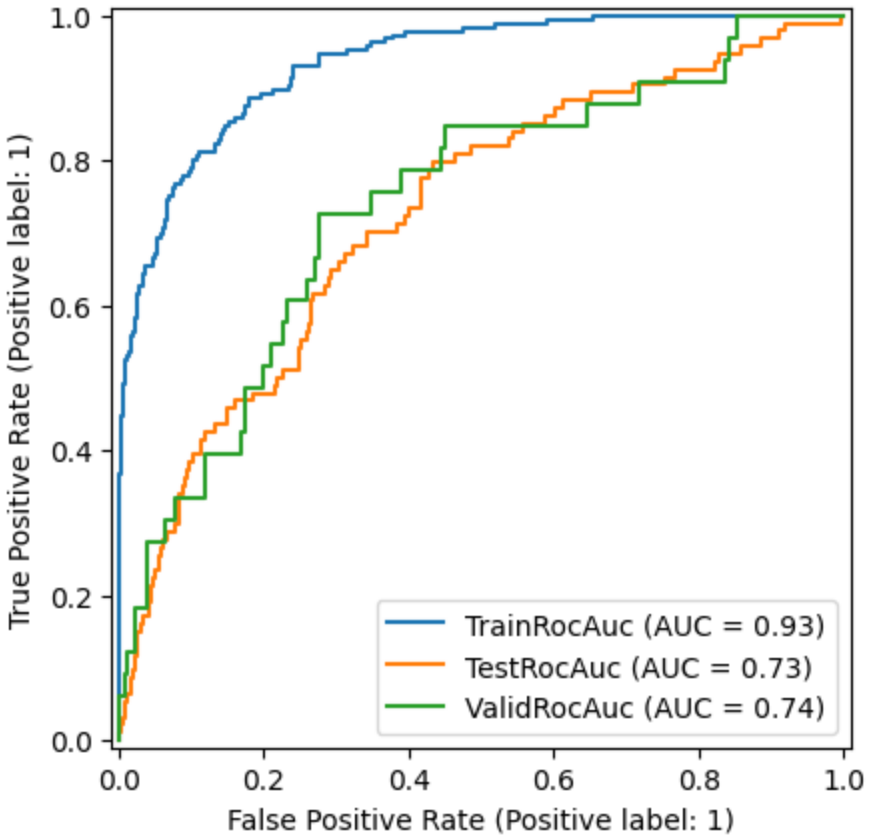
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Train Log Loss
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5634
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6816
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9004
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6394
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5845

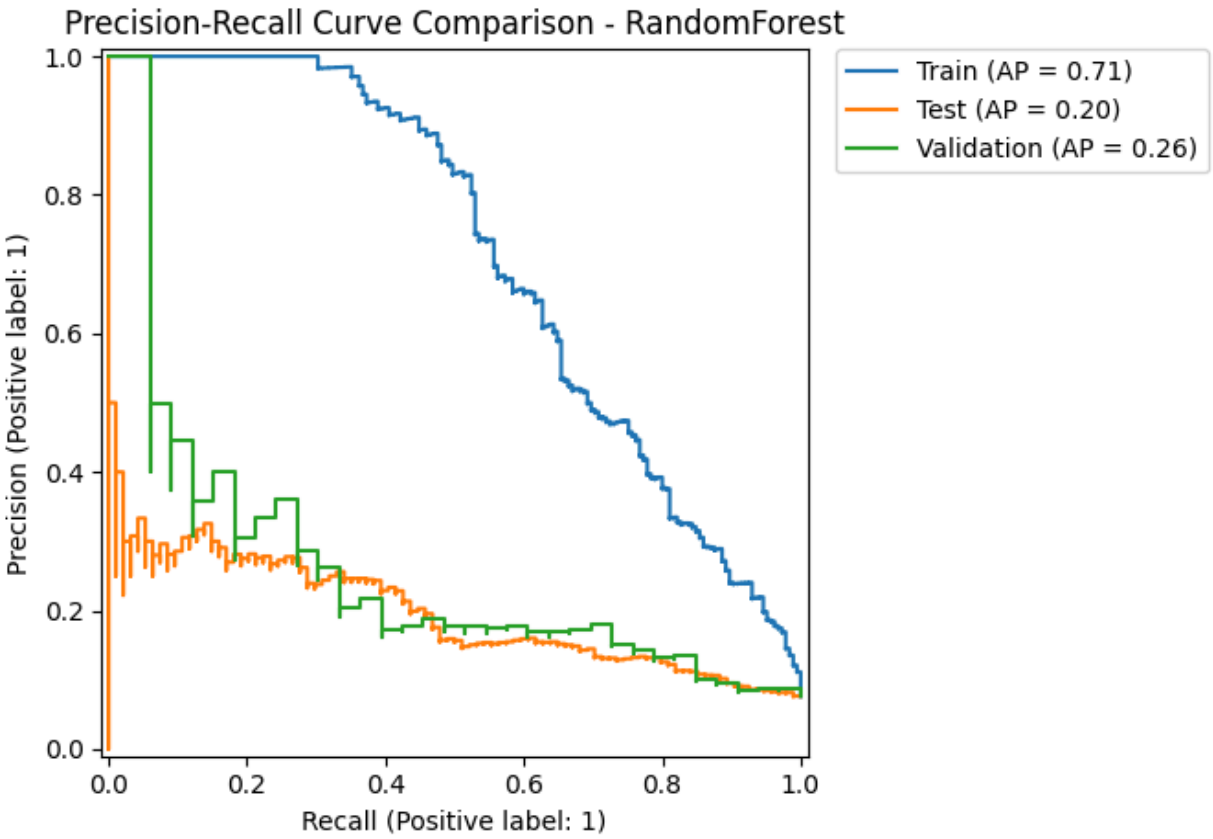
RandomForest

In [ ]:
```
ConductGridSearch(classifiers[5],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START RandomForest *****
Parameters:
        bootstrap: [True]
        max_depth: [5, 10]
        max_features: [15, 20]
        min_samples_leaf: [3, 5]
        min_samples_split: [5, 10]
        n_estimators: [1000]
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Cross validation with best estimator
Fit and Prediction with best estimator
```








```
Best Parameters:
        predictor__bootstrap: True
        predictor__max_depth: 5
        predictor__max_features: 20
        predictor__min_samples_leaf: 3
        predictor__min_samples_split: 5
        predictor__n_estimators: 1000
****** FINISH RandomForest  *****
```

In [ ]:
```
gc.collect()
```

Out[ ]:
```
57756
```

In [ ]:
```
expLog
```

Out[ ]:

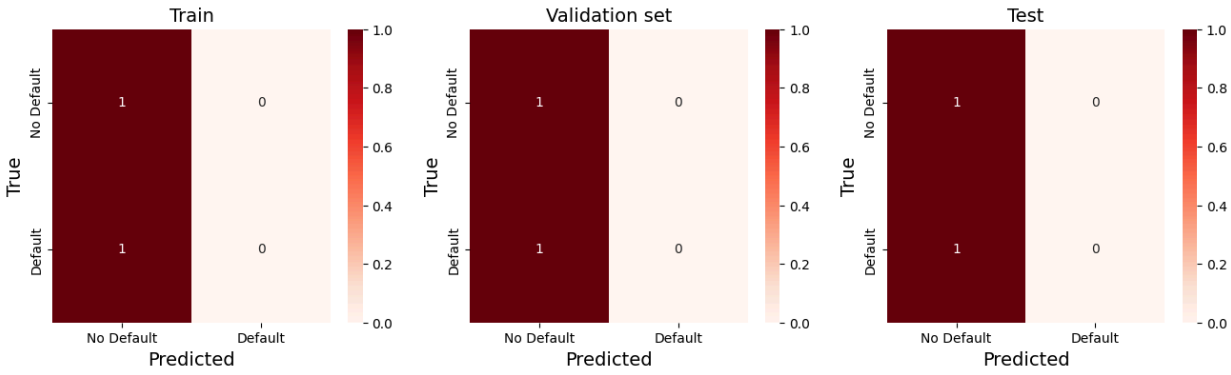
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Train Log Loss
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5634
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6816
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9004
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6394
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5845
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6647

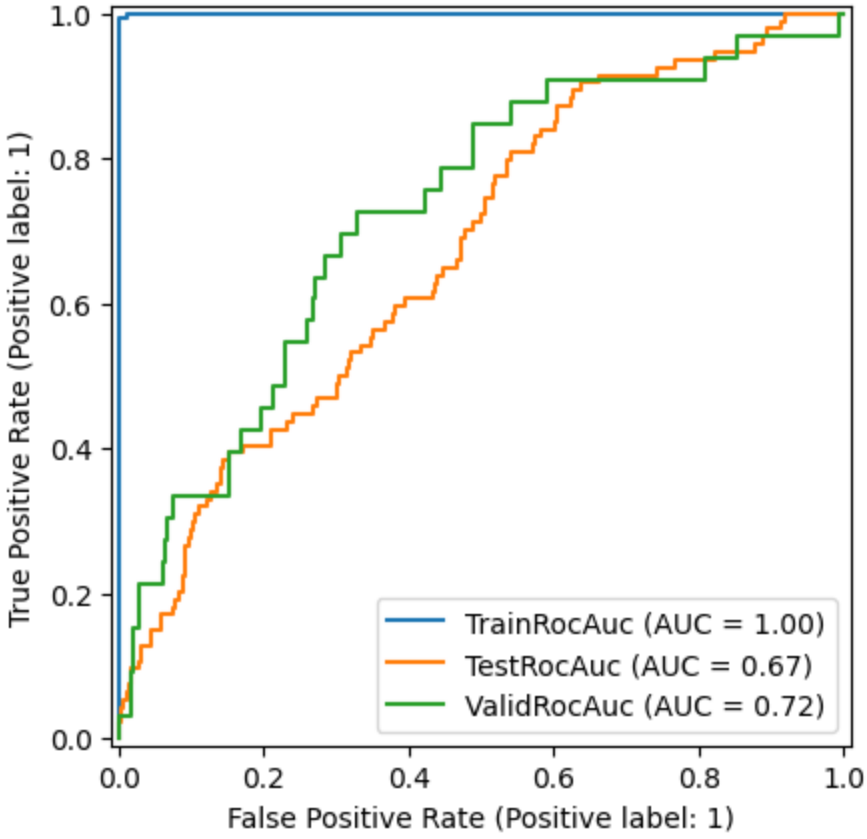
Support Vector

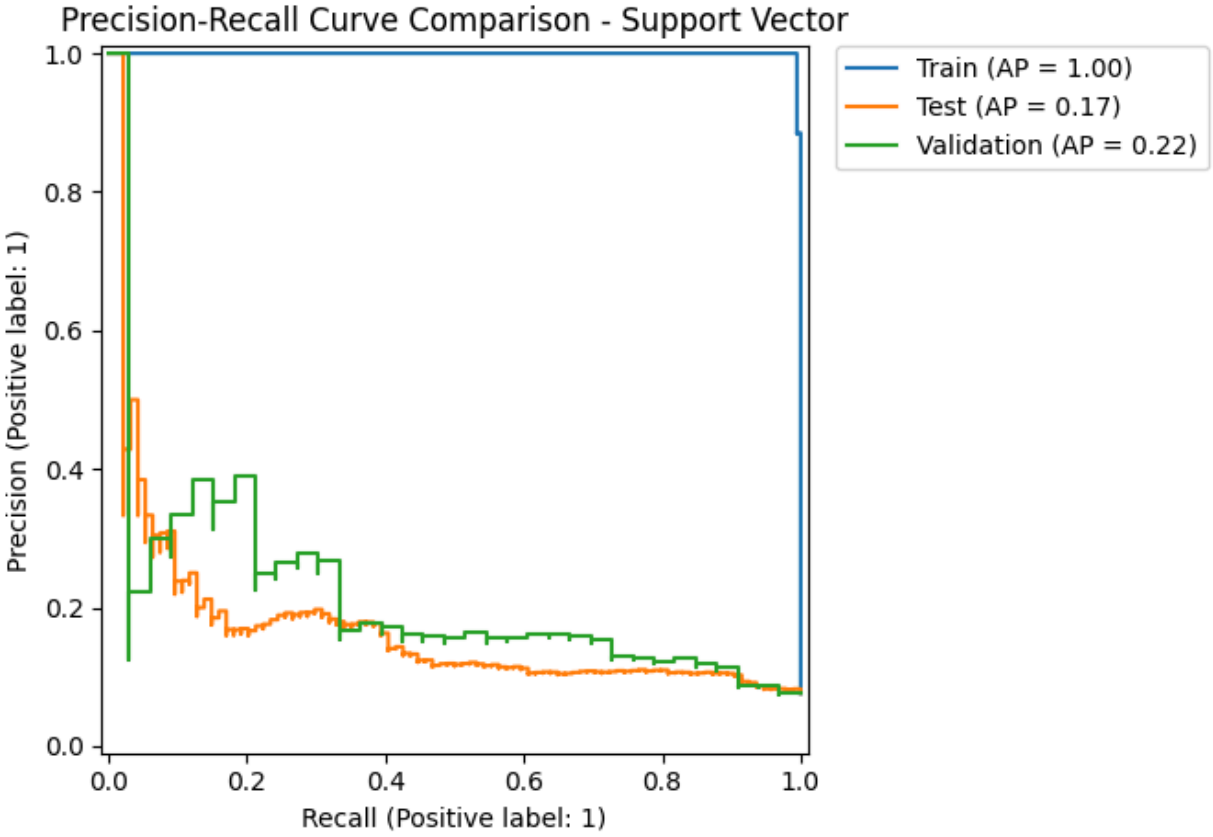
In [ ]:

```
ConductGridSearch(classifiers[1],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START Support Vector *****
Parameters:
        C: (0.001, 0.01)
        degree: (4, 5)
        gamma: (0.01, 0.1, 1)
        kernel: ('rbf', 'poly')
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Cross validation with best estimator
Fit and Prediction with best estimator
```




True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
TrainRocAuc (AUC = 1.00)
TestRocAuc (AUC = 0.67)
ValidRocAuc (AUC = 0.72)


Precision-Recall Curve Comparison - Support Vector
Precision (Positive label: 1)
Recall (Positive label: 1)
Train (AP = 1.00)
Test (AP = 0.17)
Validation (AP = 0.22)

```
Best Parameters:
        predictor__C: 0.01
        predictor__degree: 4
        predictor__gamma: 0.01
        predictor__kernel: rbf
****** FINISH Support Vector  *****
```

In [ ]:
```
gc.collect()
```

Out[ ]:
```
19282
```

In [ ]:
```
expLog
```

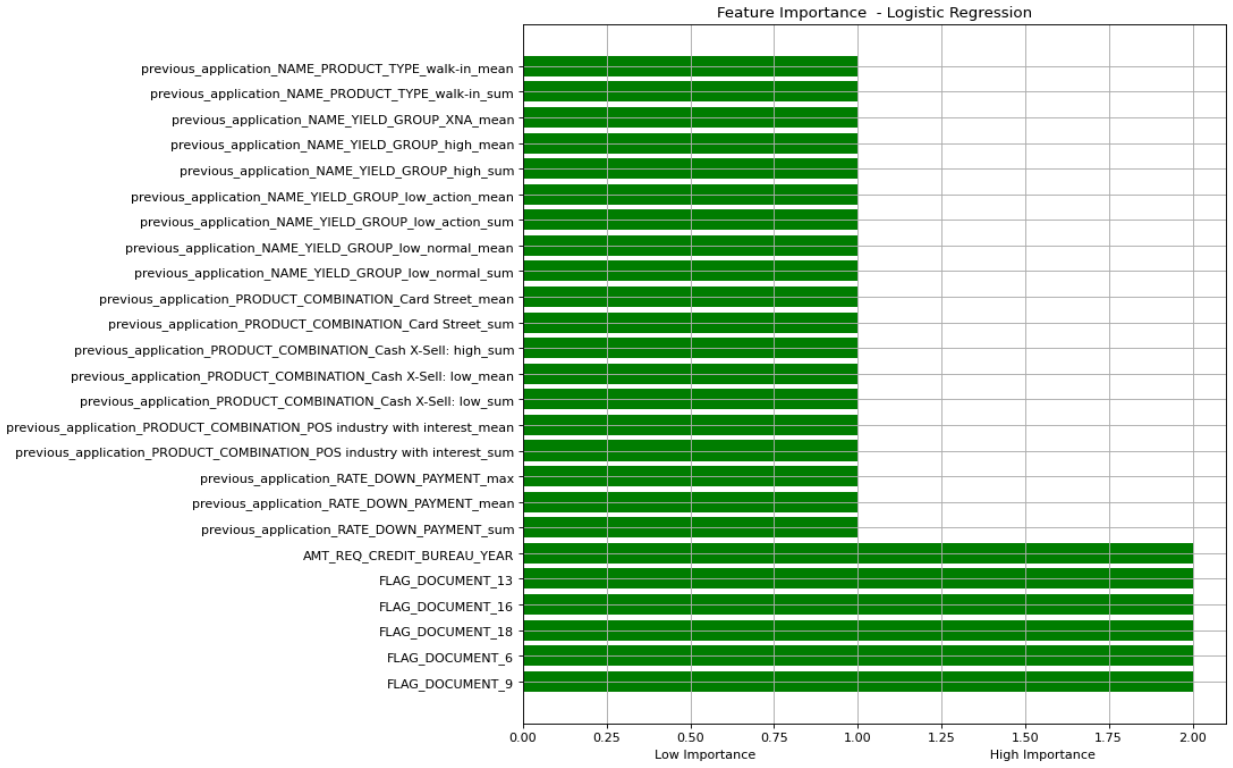
Out[ ]:

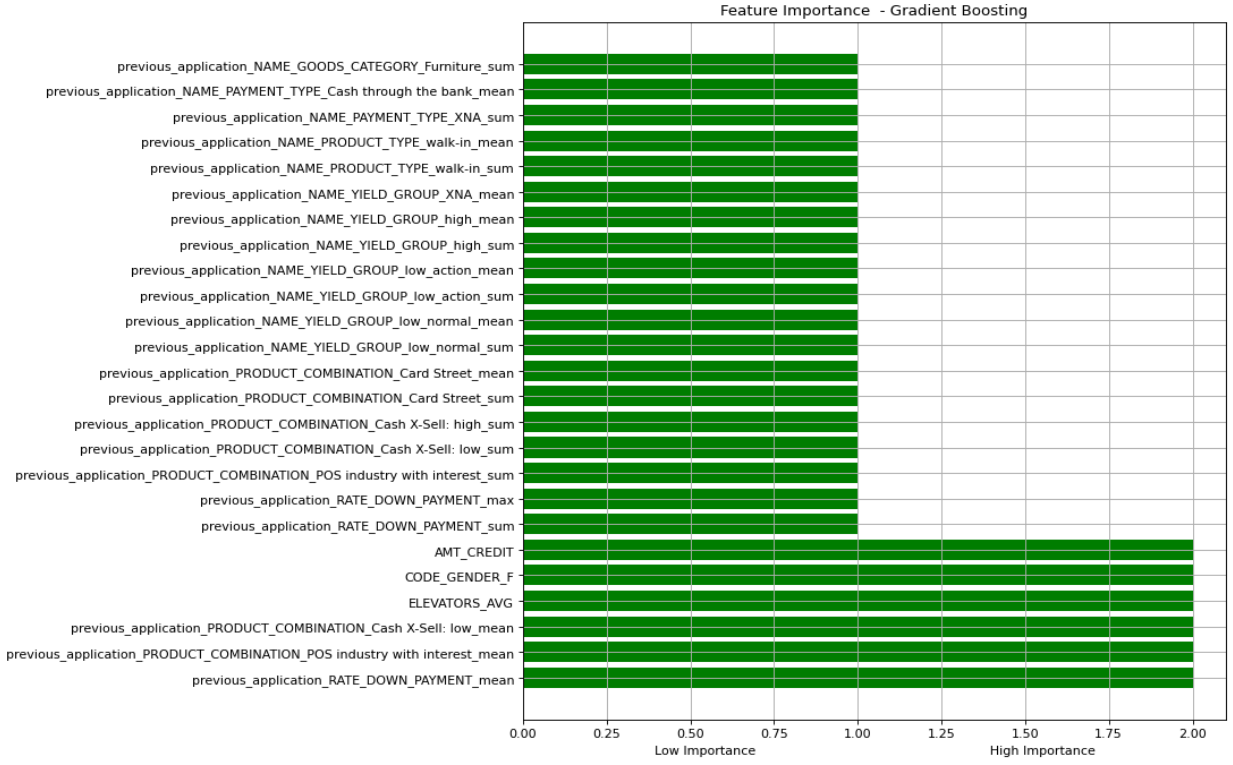
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Train Log Loss
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5634
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6816
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9004
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6394
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5845
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6647
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7196

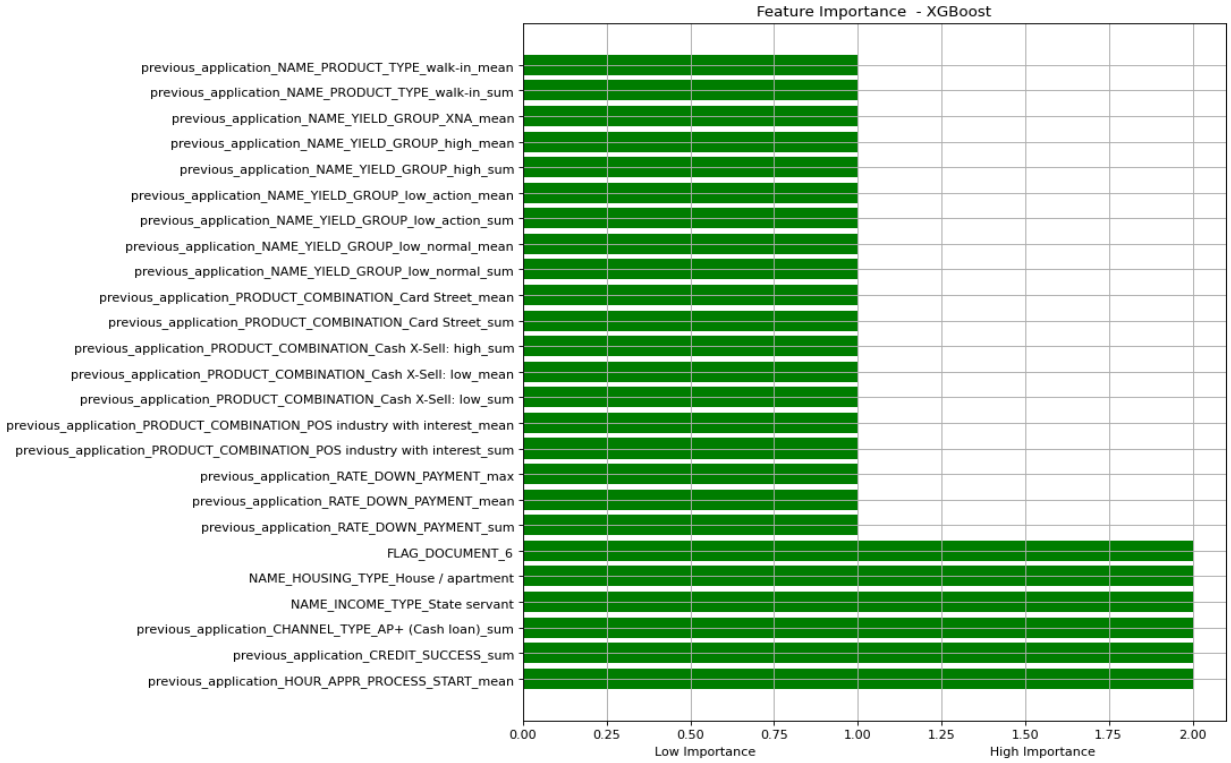
Model Validation

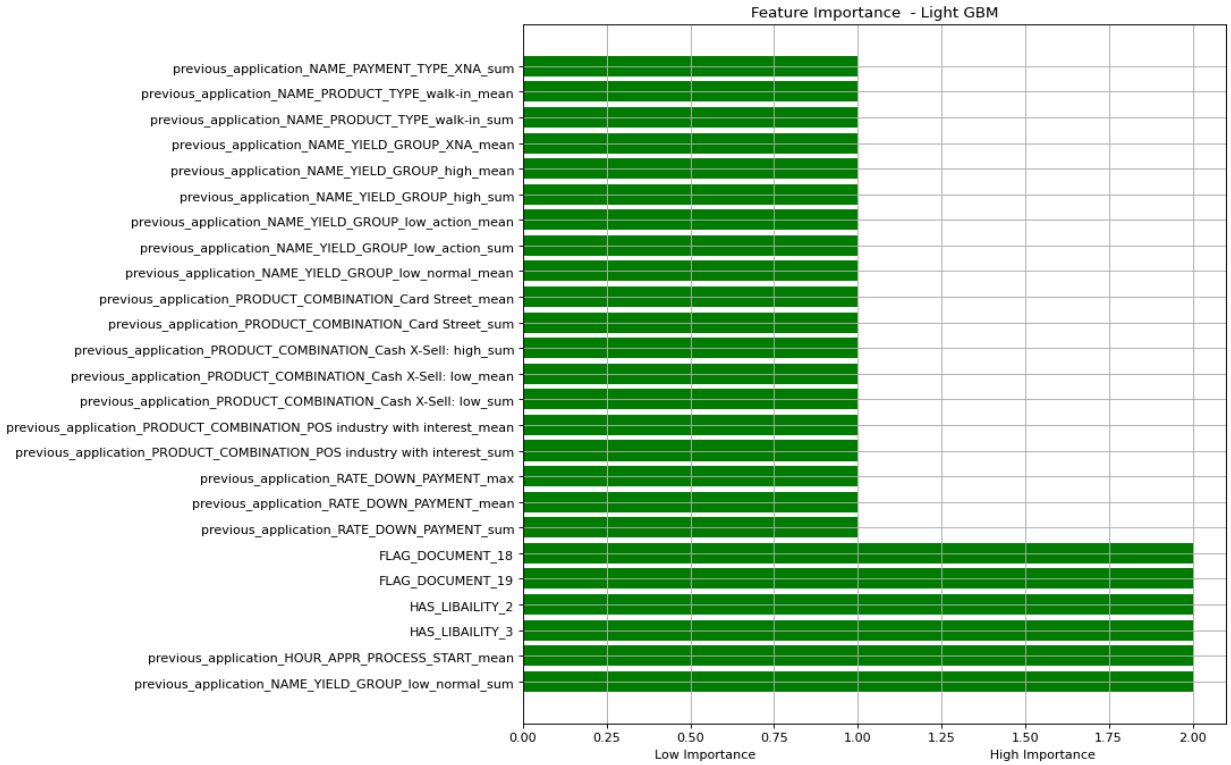
Feature Importance based on all Models

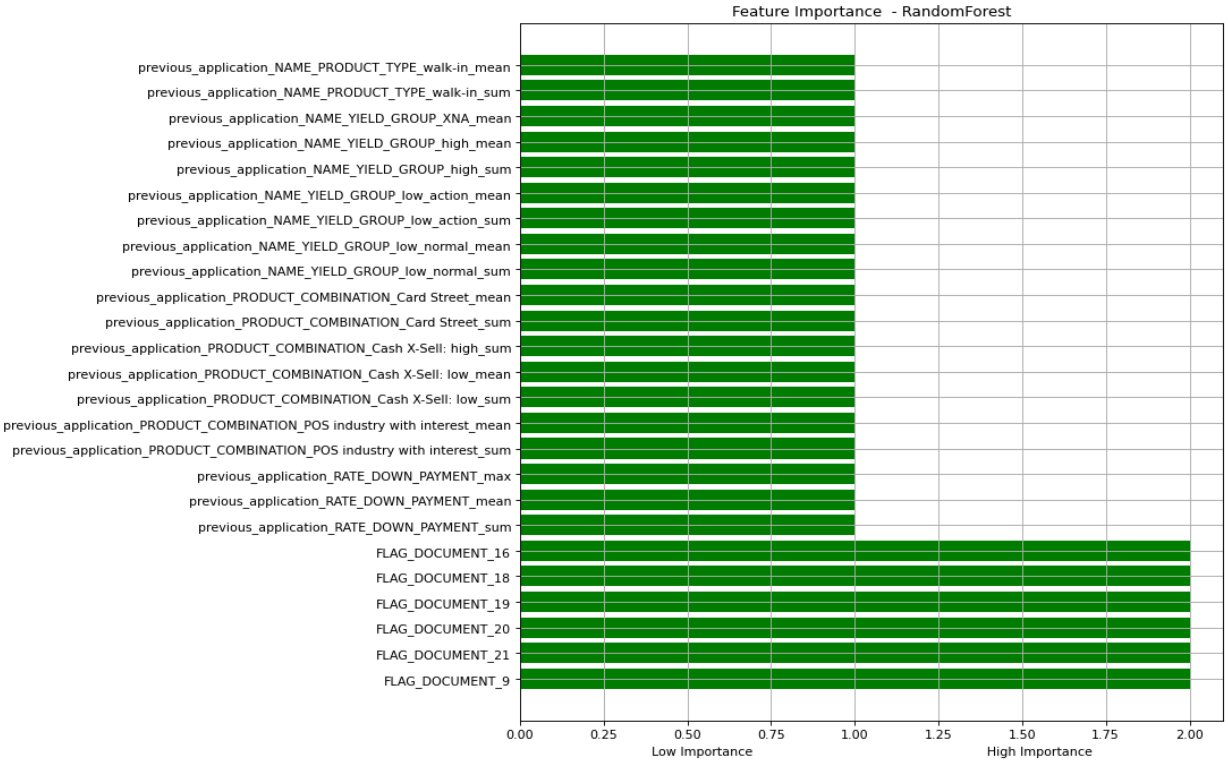
In [ ]:
```
# plot feature importance by their ranking for each model
for name in names[1:-1]:
    plt.figure(figsize=(10,10), dpi= 80)
    features_df = features_df = features_list[name].sort_values(['feature_importance'
    sortedNames = np.array(features_df)[0:25, 0]
    sortedImportances = np.array(features_df)[0:25, 1]
    plt.title('Feature Importance  - ' + name)
    plt.barh(range(len(sortedNames)), sortedImportances, color='g', align='center')
    plt.yticks(range(len(sortedNames)), sortedNames)
    plt.xlabel('Low Importance
    plt.grid()
    plt.show()
```


Feature Importance - Logistic Regression
previous_application_NAME_PRODUCT_TYPE_walk-in_mean
previous_application_NAME_PRODUCT_TYPE_walk-in_sum
previous_application_NAME_YIELD_GROUP_XNA_mean
previous_application_NAME_YIELD_GROUP_high_mean
previous_application_NAME_YIELD_GROUP_high_sum
previous_application_NAME_YIELD_GROUP_low_action_mean
previous_application_NAME_YIELD_GROUP_low_action_sum
previous_application_NAME_YIELD_GROUP_low_normal_mean
previous_application_NAME_YIELD_GROUP_low_normal_sum
previous_application_PRODUCT_COMBINATION_Card Street_mean
previous_application_PRODUCT_COMBINATION_Card Street_sum
previous_application_PRODUCT_COMBINATION_Cash X-Sell: high_sum
previous_application_PRODUCT_COMBINATION_Cash X-Sell: low_mean
previous_application_PRODUCT_COMBINATION_Cash X-Sell: low_sum
previous_application_PRODUCT_COMBINATION_POS industry with interest_mean
previous_application_PRODUCT_COMBINATION_POS industry with interest_sum
previous_application_RATE_DOWN_PAYMENT_max
previous_application_RATE_DOWN_PAYMENT_mean
previous_application_RATE_DOWN_PAYMENT_sum
AMT_REQ_CREDIT_BUREAU_YEAR
FLAG_DOCUMENT_13
FLAG_DOCUMENT_16
FLAG_DOCUMENT_18
FLAG_DOCUMENT_6
FLAG_DOCUMENT_9
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Low Importance
High Importance


Feature Importance - Gradient Boosting
previous_application_NAME_GOODS_CATEGORY_Furniture_sum
previous_application_NAME_PAYMENT_TYPE_Cash through the bank_mean
previous_application_NAME_PAYMENT_TYPE_XNA_sum
previous_application_NAME_PRODUCT_TYPE_walk-in_mean
previous_application_NAME_PRODUCT_TYPE_walk-in_sum
previous_application_NAME_YIELD_GROUP_XNA_mean
previous_application_NAME_YIELD_GROUP_high_mean
previous_application_NAME_YIELD_GROUP_high_sum
previous_application_NAME_YIELD_GROUP_low_action_mean
previous_application_NAME_YIELD_GROUP_low_action_sum
previous_application_NAME_YIELD_GROUP_low_normal_mean
previous_application_NAME_YIELD_GROUP_low_normal_sum
previous_application_PRODUCT_COMBINATION_Card Street_mean
previous_application_PRODUCT_COMBINATION_Card Street_sum
previous_application_PRODUCT_COMBINATION_Cash X-Sell: high_sum
previous_application_PRODUCT_COMBINATION_Cash X-Sell: low_sum
previous_application_PRODUCT_COMBINATION_POS industry with interest_sum
previous_application_RATE_DOWN_PAYMENT_max
previous_application_RATE_DOWN_PAYMENT_sum
AMT_CREDIT
CODE_GENDER_F
ELEVATORS_AVG
previous_application_PRODUCT_COMBINATION_Cash X-Sell: low_mean
previous_application_PRODUCT_COMBINATION_POS industry with interest_mean
previous_application_RATE_DOWN_PAYMENT_mean
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Low Importance
High Importance


Feature Importance - XGBoost
previous_application_NAME_PRODUCT_TYPE_walk-in_mean
previous_application_NAME_PRODUCT_TYPE_walk-in_sum
previous_application_NAME_YIELD_GROUP_XNA_mean
previous_application_NAME_YIELD_GROUP_high_mean
previous_application_NAME_YIELD_GROUP_high_sum
previous_application_NAME_YIELD_GROUP_low_action_mean
previous_application_NAME_YIELD_GROUP_low_action_sum
previous_application_NAME_YIELD_GROUP_low_normal_mean
previous_application_NAME_YIELD_GROUP_low_normal_sum
previous_application_PRODUCT_COMBINATION_Card Street_mean
previous_application_PRODUCT_COMBINATION_Card Street_sum
previous_application_PRODUCT_COMBINATION_Cash X-Sell: high_sum
previous_application_PRODUCT_COMBINATION_Cash X-Sell: low_mean
previous_application_PRODUCT_COMBINATION_Cash X-Sell: low_sum
previous_application_PRODUCT_COMBINATION_POS industry with interest_mean
previous_application_PRODUCT_COMBINATION_POS industry with interest_sum
previous_application_RATE_DOWN_PAYMENT_max
previous_application_RATE_DOWN_PAYMENT_mean
previous_application_RATE_DOWN_PAYMENT_sum
FLAG_DOCUMENT_6
NAME_HOUSING_TYPE_House / apartment
NAME_INCOME_TYPE_State servant
previous_application_CHANNEL_TYPE_AP+ (Cash loan)_sum
previous_application_CREDIT_SUCCESS_sum
previous_application_HOUR_APPR_PROCESS_START_mean
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Low Importance
High Importance

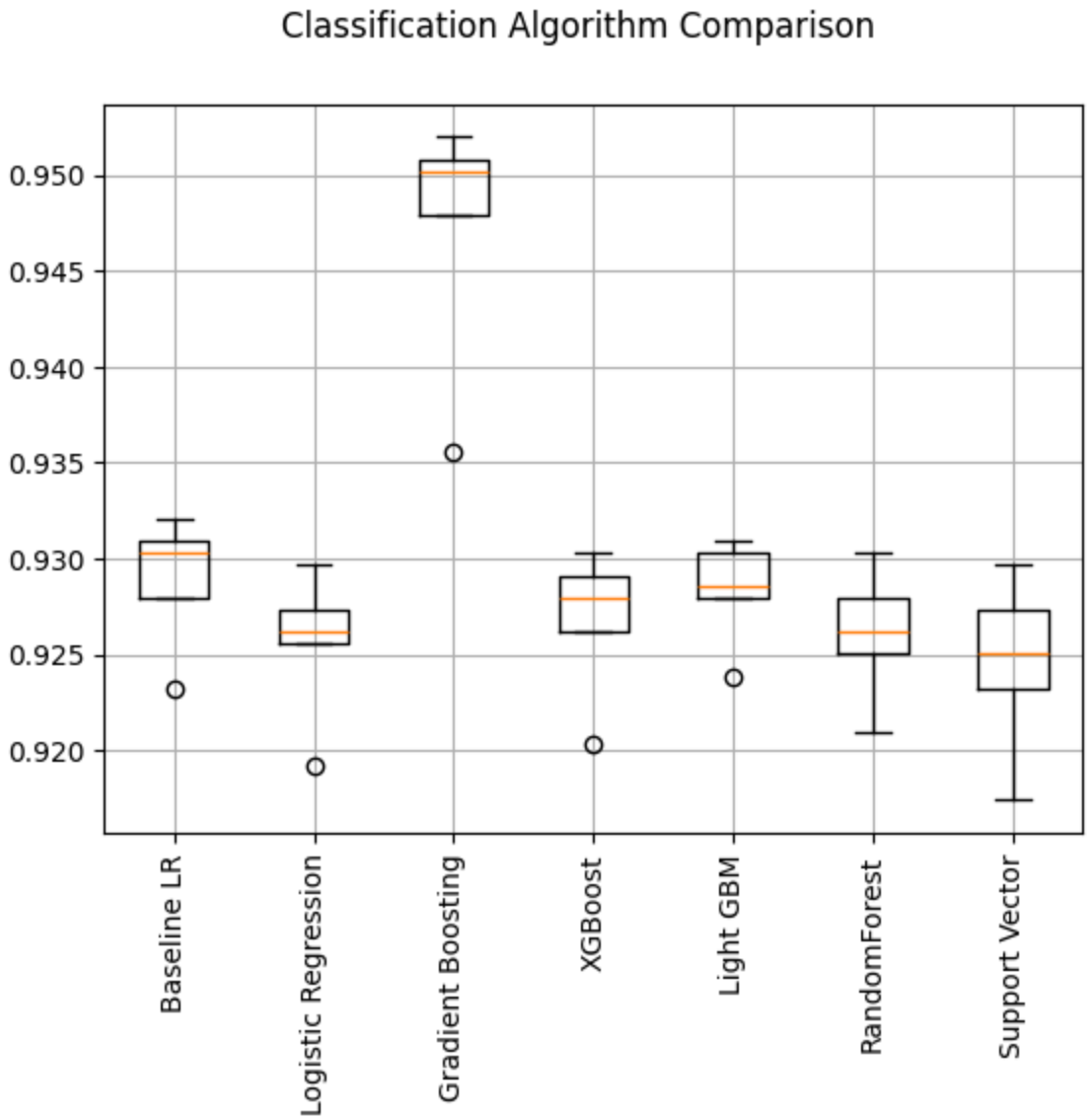

Feature Importance - Light GBM
previous_application_NAME_PAYMENT_TYPE_XNA_sum
previous_application_NAME_PRODUCT_TYPE_walk-in_mean
previous_application_NAME_PRODUCT_TYPE_walk-in_sum
previous_application_NAME_YIELD_GROUP_XNA_mean
previous_application_NAME_YIELD_GROUP_high_mean
previous_application_NAME_YIELD_GROUP_high_sum
previous_application_NAME_YIELD_GROUP_low_action_mean
previous_application_NAME_YIELD_GROUP_low_action_sum
previous_application_NAME_YIELD_GROUP_low_normal_mean
previous_application_PRODUCT_COMBINATION_Card Street_mean
previous_application_PRODUCT_COMBINATION_Card Street_sum
previous_application_PRODUCT_COMBINATION_Cash X-Sell: high_sum
previous_application_PRODUCT_COMBINATION_Cash X-Sell: low_mean
previous_application_PRODUCT_COMBINATION_Cash X-Sell: low_sum
previous_application_PRODUCT_COMBINATION_POS industry with interest_mean
previous_application_PRODUCT_COMBINATION_POS industry with interest_sum
previous_application_RATE_DOWN_PAYMENT_max
previous_application_RATE_DOWN_PAYMENT_mean
previous_application_RATE_DOWN_PAYMENT_sum
FLAG_DOCUMENT_18
FLAG_DOCUMENT_19
HAS_LIBAILITY_2
HAS_LIBAILITY_3
previous_application_HOUR_APPR_PROCESS_START_mean
previous_application_NAME_YIELD_GROUP_low_normal_sum
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
Low Importance
High Importance



Boxplot with all CV results

In [ ]:

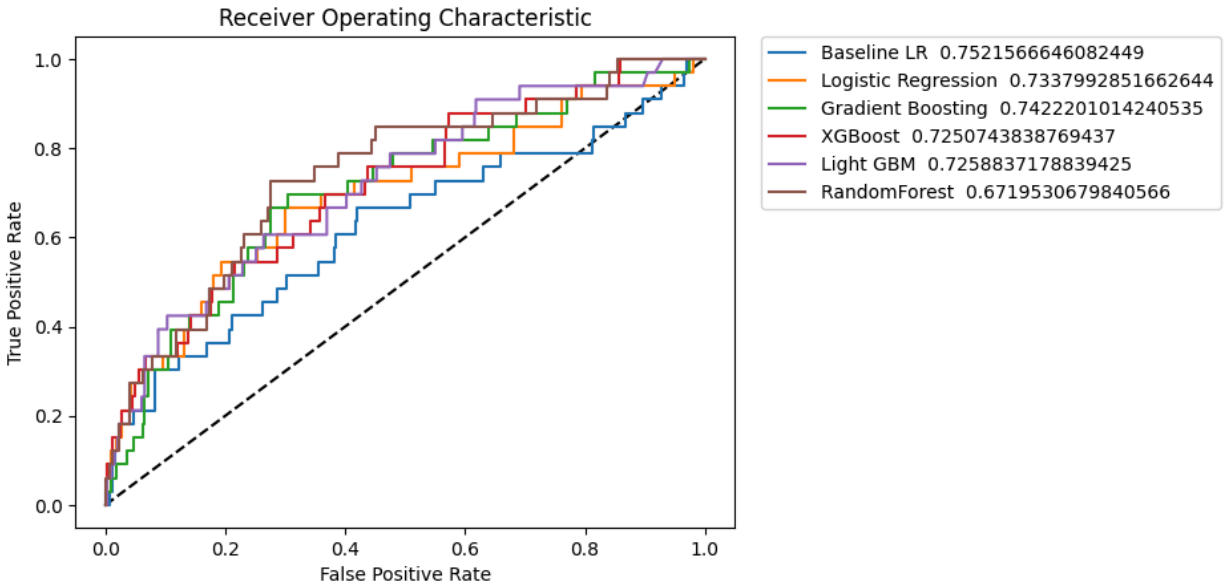
```
# boxplot algorithm comparison
fig = pyplot.figure()
fig.suptitle('Classification Algorithm Comparison')
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names,rotation=90)
pyplot.grid()
pyplot.show()
```



AUC (Area Under the ROC Curve)

In [ ]:

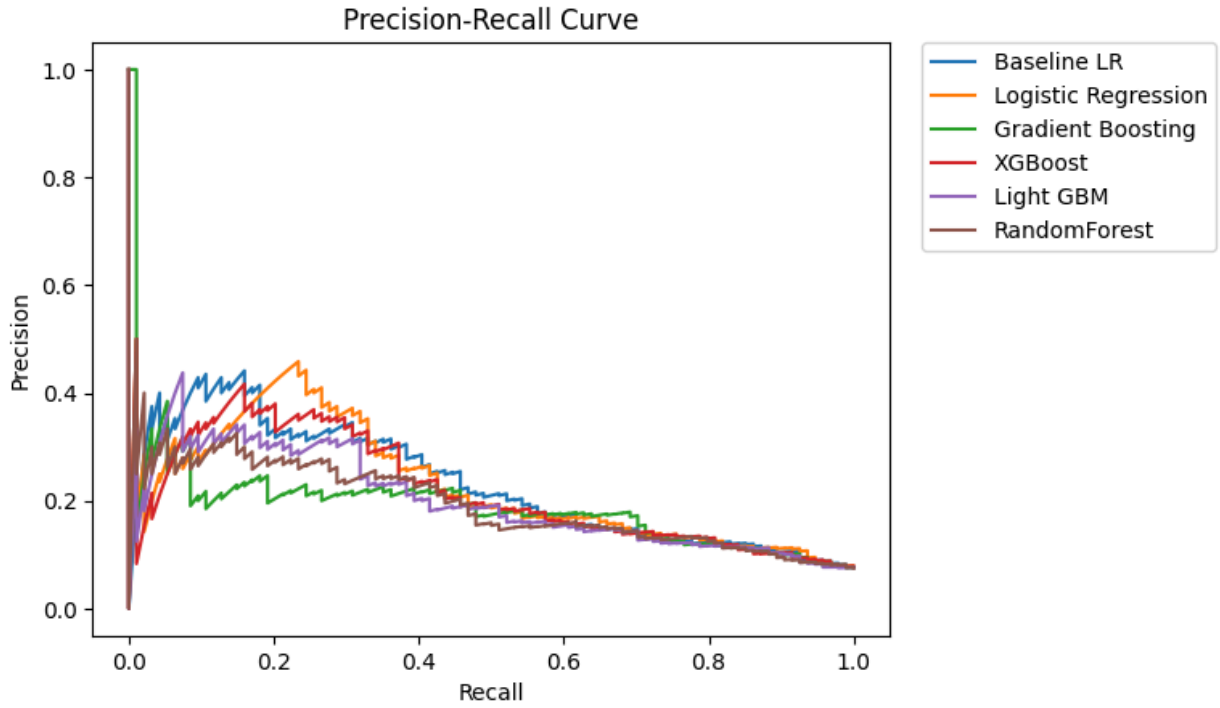
```
# roc curve fpr, tpr  for all classifiers
plt.plot([0,1],[0,1], 'k--')
for i in range(len(names)-1):
    plt.plot(fprs[i],tprs[i],label = names[i] + '   ' + str(scores[i]))
plt.legend(bbox_to_anchor=(1.04,1), loc="upper left", borderaxespad=0)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title('Receiver Operating Characteristic')
plt.show()
```



Precision Recall Curve

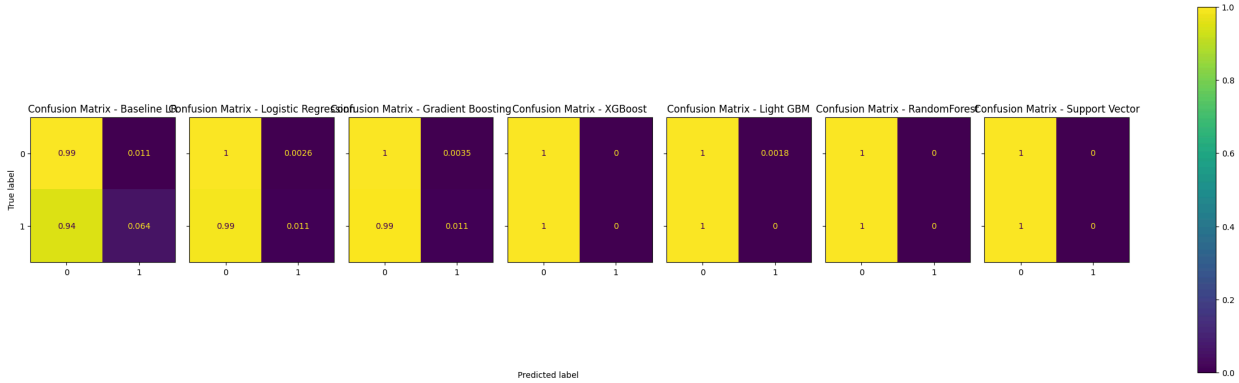
In [ ]:

```
# precision recall curve  for all classifiers
for i in range(len(names)-1):
    plt.plot(recalls[i],precisions[i],label = names[i])
plt.legend(bbox_to_anchor=(1.04,1), loc="upper left", borderaxespad=0)
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title('Precision-Recall Curve')
plt.show()
```



Confusion Matrix

In [ ]:
```
# plot confusion matrix for all classifiers
f, axes = plt.subplots(1, len(names), figsize=(30, 8), sharey='row')
for i in range(len(names)):
    disp = ConfusionMatrixDisplay(cnfmatrix[i], display_labels=['0', '1'])
    disp.plot(ax=axes[i], xticks_rotation=0)
    disp.ax_.set_title("Confusion Matrix - " + names[i])
    disp.im_.colorbar.remove()
    disp.ax_.set_xlabel('')
    if i!=0:
        disp.ax_.set_ylabel('')

f.text(0.4, 0.1, 'Predicted label', ha='left')
plt.subplots_adjust(wspace=0.10, hspace=0.1)

f.colorbar(disp.im_, ax=axes)
plt.show()
```



Final results

In [ ]:
```
pd.set_option('display.max_colwidth', None)
expLog.to_csv("/content/Data/expLog_RFE.csv",index=False)
expLog
```

Out[ ]:

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Train Log Loss
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5634
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6816
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9004
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6394
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5845
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6647
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7196

In [ ]:

```
final_best_clf
final_best_cl_rfe_fdf = pd.DataFrame(list(final_best_clf.items()))#,columns = ['colum
with open('/content/Data/final_best_clf_RFE.txt', 'w') as file:
     file.write(str(final_best_clf))
```

PCA & Handling MultiCollinearity

Multicollinearity highly affects the variance associated with the problem, and can also affect the interpretation of the model, as it undermines the statistical significance of independent variables.For a dataset, if some of the independent variables are highly independent of each other, it results in multicollinearity. A small change in any of the features can affect the model performance to a great extent. In other words, The coefficients of the model become very sensitive to small changes in the independent variables.The basic idea is to run a PCA on all predictors. Their ratio, the Condition Index, will be high if multicollinearity is present.

Logistic regression with PCA

In [ ]:

```
for (name, classifier,feature_sel) in classifiers[0]:
        # Print classifier and parameters
        print('****** START', name,'*****')
        parameters = params_grid[name]
        print("Parameters:")
        for p in sorted(parameters.keys()):
            print("\t"+str(p)+": "+ str(parameters[p]))

        # generate the pipeline based on the feature selection method
        full_pipeline_with_predictor = Pipeline([
            ("preparation", data_prep_pipeline),
            ("PCA",PCA(0.95)),
            ("predictor", classifier)
            ])

        # Execute the grid search
        params = {}
        for p in parameters.keys():
            pipe_key = 'predictor__'+str(p)
            params[pipe_key] = parameters[p]
        grid_search = GridSearchCV(full_pipeline_with_predictor, params, cv=cvSplits,
                                   n_jobs=-1,verbose=1)
        grid_search.fit(X_train, y_train)

        # Best estimator score
        best_train = pct(grid_search.best_score_)

        # Best train scores
        print("Cross validation with best estimator")
        best_train_scores = cross_validate(grid_search.best_estimator_, X_train, y_tr
                                           return_train_score=True, n_jobs=-1)

        #get all scores
        best_train_accuracy = np.round(best_train_scores['train_accuracy'].mean(),4)
        best_train_f1 = np.round(best_train_scores['train_f1'].mean(),4)
        best_train_logloss = np.round(best_train_scores['train_log_loss'].mean(),4)
        best_train_roc_auc = np.round(best_train_scores['train_roc_auc'].mean(),4)

        valid_time = np.round(best_train_scores['score_time'].mean(),4)
        best_valid_accuracy = np.round(best_train_scores['test_accuracy'].mean(),4)
        best_valid_f1 = np.round(best_train_scores['test_f1'].mean(),4)
        best_valid_logloss = np.round(best_train_scores['test_log_loss'].mean(),4)
```

```
        best_valid_roc_auc = np.round(best_train_scores['test_roc_auc'].mean(),4)

        (t_stat, p_value) = stats.ttest_rel(logit_scores['train_roc_auc'], best_train

        #test and Prediction with whole data
        # Best estimator fitting time
        print("Fit and Prediction with best estimator")
        start = time()
        model = grid_search.best_estimator_.fit(X_train, y_train)
        train_time = round(time() - start, 4)

        # Best estimator prediction time
        start = time()
        y_test_pred = model.predict(X_test)
        test_time = round(time() - start, 4)

        # Collect the best parameters found by the grid search
        print("Best Parameters:")
        best_parameters = grid_search.best_estimator_.get_params()
        param_dump = []
        for param_name in sorted(params.keys()):
            param_dump.append((param_name, best_parameters[param_name]))
            print("\t"+str(param_name)+": " + str(best_parameters[param_name]))
        print("****** FINISH",name," *****")
        print("")

        # Record the results
#        exp_name = "Logistic Regression with PCA"
        expLog.loc[len(expLog)] = [f"{exp_name}"] + list(np.round(
                [best_train_accuracy,
                #pct(accuracy_score(y_valid, model.predict(X_valid))),
                best_valid_accuracy,
                accuracy_score(y_test, y_test_pred),
                best_train_roc_auc,
                best_valid_roc_auc,
                #roc_auc_score(y_valid, model.predict_proba(X_valid)[:, 1]),
                roc_auc_score(y_test, model.predict_proba(X_test)[:, 1]),
                best_train_f1,
                best_valid_f1,
                f1_score(y_test, y_test_pred),
                best_train_logloss,
                best_valid_logloss,
                log_loss(y_test, y_test_pred),
                p_value
                ],4)) + [train_time,valid_time,test_time] \
                + [json.dumps(param_dump)]
```

```
****** START Logistic Regression *****
Parameters:
        C: (10, 1, 0.1, 0.01)
        penalty: ('l1', 'l2', 'elasticnet')
        tol: (0.0001, 1e-05)
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Cross validation with best estimator
Fit and Prediction with best estimator
Best Parameters:
        predictor__C: 0.01
        predictor__penalty: l2
        predictor__tol: 1e-05
****** FINISH Logistic Regression  *****
```

In [ ]:

```
expLog
```

Out[ ]:

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Train Log Loss
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5634
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6816
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9004
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6394
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5845
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6647
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7196
7	Baseline_132_features	0.9256	0.9219	0.9228	0.8240	0.7427	0.7549	0.0503	0.0237	0.0206	2.6816

Tune Basline model (Grid search & SelectKbest Feature Selection)

Various Classification algorithms were used to compare with the best model. Following metrics were used to find the best model

- Cross fold Train Accuracy
- Test Accuracy
- p-value
- Train ROC_AUC_Score
- Test ROC_AUC_Score
- Train F1_Score
- Test F1_Score
- Train LogLoss
- Test LogLoss
- Train Time
- Test Time
- Confusion matrix

In [ ]:
```
# Clean up the  arrays

del fprs[1:]
del tprs[1:]
del precisions[1:]
del recalls[1:]
del names[1:]
del scores[1:]
del cvscores[1:]
del pvalues[1:]
del accuracy[1:]
del cnfmatrix[1:]
del results[1:]
final_best_clf,results = {}, {}
```

In [ ]:
```
print(names)
```

```
['Baseline LR']
```

Classifiers

In [ ]:
```
classifiers = [
        [('Logistic Regression', LogisticRegression(solver='saga',random_state=42),"S
        [('XGBoost', XGBClassifier(random_state=42),"SelectKbest")],
        [('Light GBM', LGBMClassifier(boosting_type='gbdt', random_state=42),"SelectK
        [('RandomForest', RandomForestClassifier(random_state=42),"SelectKbest")]
    ]
```

Hyper-parameters for different models

In [ ]:

```
# Arrange grid search parameters for each classifier
params_grid = {
        'Logistic Regression': {
            'penalty': ('l1', 'l2','elasticnet'),
            'tol': (0.0001, 0.00001),
            'C': (10, 1, 0.1, 0.01),
        }
    ,
        'XGBoost':  {
            'max_depth': [3,5], # Lower helps with overfitting
            'n_estimators':[300,500],
            'learning_rate': [0.01,0.1],
#             'objective': ['binary:logistic'],
#             'eval_metric': ['auc'],
            'eta' : [0.01,0.1],
            'colsample_bytree' : [0.2,0.5],
        },
        'Light GBM':  {
            'max_depth': [2,5],  # Lower helps with overfitting
            'num_leaves': [5,10], # Equivalent to max depth
            'n_estimators':[1000,5000],
            'learning_rate': [0.01,0.1],
 #           'reg_alpha': [0.1,0.01,1],
 #           'reg_lambda': [0.1,0.01,1],
            'boosting_type':['goss','dart'],
 #           'metric': ['auc'],
 #           'objective':['binary'],
            'max_bin' : [100,200],  #Setting it to high values has similar effect as
        },                          #small numbers reduces accuracy but runs faster

        'RandomForest':  {
            'max_depth': [5,10],
            'max_features': [15,20],
            'min_samples_split': [5, 10],
            'min_samples_leaf': [3, 5],
            'bootstrap': [True],
            'n_estimators':[1000]},
    }
```

Conduct Grid Search

In [ ]:

```
results = []
results.append(logit_scores['train_accuracy'])
def ConductGridSearch(in_classifiers,cnfmatrix,fprs,tprs,precisions,recalls):
    for (name, classifier,feature_sel) in in_classifiers:
            # Print classifier and parameters
            print('****** START', name,'*****')
            parameters = params_grid[name]
            print("Parameters:")
            for p in sorted(parameters.keys()):
                print("\t"+str(p)+": "+ str(parameters[p]))

            # generate the pipeline based on the feature selection method
```

```
full_pipeline_with_predictor = Pipeline([
    ("preparation", data_prep_pipeline),
    ('SelectKbest',SelectKBest(score_func=mutual_info_classif, k=features
    ("predictor", classifier)
    ])

# Execute the grid search
params = {}
for p in parameters.keys():
    pipe_key = 'predictor__'+str(p)
    params[pipe_key] = parameters[p]
grid_search = GridSearchCV(full_pipeline_with_predictor, params, cv=cvSpl
                           n_jobs=-1,verbose=1)
grid_search.fit(X_train, y_train)

# Best estimator score
best_train = pct(grid_search.best_score_)

# Best train scores
print("Cross validation with best estimator")
best_train_scores = cross_validate(grid_search.best_estimator_, X_train,
                                   return_train_score=True, n_jobs=-1)

#get all scores
best_train_accuracy = np.round(best_train_scores['train_accuracy'].mean()
best_train_f1 = np.round(best_train_scores['train_f1'].mean(),4)
best_train_logloss = np.round(best_train_scores['train_log_loss'].mean(),
best_train_roc_auc = np.round(best_train_scores['train_roc_auc'].mean(),4

valid_time = np.round(best_train_scores['score_time'].mean(),4)
best_valid_accuracy = np.round(best_train_scores['test_accuracy'].mean(),
best_valid_f1 = np.round(best_train_scores['test_f1'].mean(),4)
best_valid_logloss = np.round(best_train_scores['test_log_loss'].mean(),4
best_valid_roc_auc = np.round(best_train_scores['test_roc_auc'].mean(),4)

#append all results
results.append(best_train_scores['train_accuracy'])
names.append(name)

# Conduct t-test with baseline logit (control) and best estimator (experi
(t_stat, p_value) = stats.ttest_rel(logit_scores['train_roc_auc'], best_t

#test and Prediction with whole data
# Best estimator fitting time
print("Fit and Prediction with best estimator")
start = time()
model = grid_search.best_estimator_.fit(X_train, y_train)
train_time = round(time() - start, 4)

# Best estimator prediction time
start = time()
y_test_pred = model.predict(X_test)
test_time = round(time() - start, 4)
scores.append(roc_auc_score(y_test, model.predict_proba(X_test)[:, 1]))
accuracy.append(accuracy_score(y_test, y_test_pred))

# Create confusion matrix for the best model
cnfmatrix = confusion_matrix_def(model,X_train,y_train,X_test,y_test,X_va

# Create AUC ROC curve
```

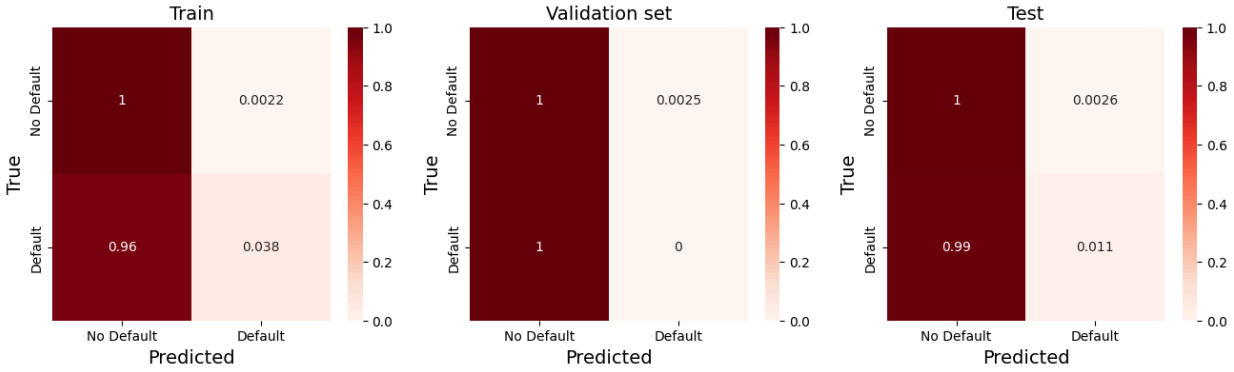
```
            fprs,tprs = roc_curve_cust(model,X_train,y_train,X_test, y_test,X_valid,

            #Create Precision recall curve
            precisions,recalls = precision_recall_cust(model,X_train,y_train,X_test,

            #Best Model
            final_best_clf[name]=pd.DataFrame([{'label': grid_search.best_estimator_.
                                               'predictor': grid_search.best_estimator_.n

            # Collect the best parameters found by the grid search
            print("Best Parameters:")
            best_parameters = grid_search.best_estimator_.get_params()
            param_dump = []
            for param_name in sorted(params.keys()):
                param_dump.append((param_name, best_parameters[param_name]))
                print("\t"+str(param_name)+": " + str(best_parameters[param_name]))
            print("****** FINISH",name," *****")
            print("")

            # Record the results
            exp_name = name+str('SelectKbest')
            expLog.loc[len(expLog)] = [f"{exp_name}"] + list(np.round(
                    [best_train_accuracy,
                    #pct(accuracy_score(y_valid, model.predict(X_valid))),
                    best_valid_accuracy,
                    accuracy_score(y_test, y_test_pred),
                    best_train_roc_auc,
                    best_valid_roc_auc,
                    #roc_auc_score(y_valid, model.predict_proba(X_valid)[:, 1]),
                    roc_auc_score(y_test, model.predict_proba(X_test)[:, 1]),
                    best_train_f1,
                    best_valid_f1,
                    f1_score(y_test, y_test_pred),
                    best_train_logloss,
                    best_valid_logloss,
                    log_loss(y_test, y_test_pred),
                    p_value
                    ],4)) + [train_time,valid_time,test_time] \
                    + [json.dumps(param_dump)]
```

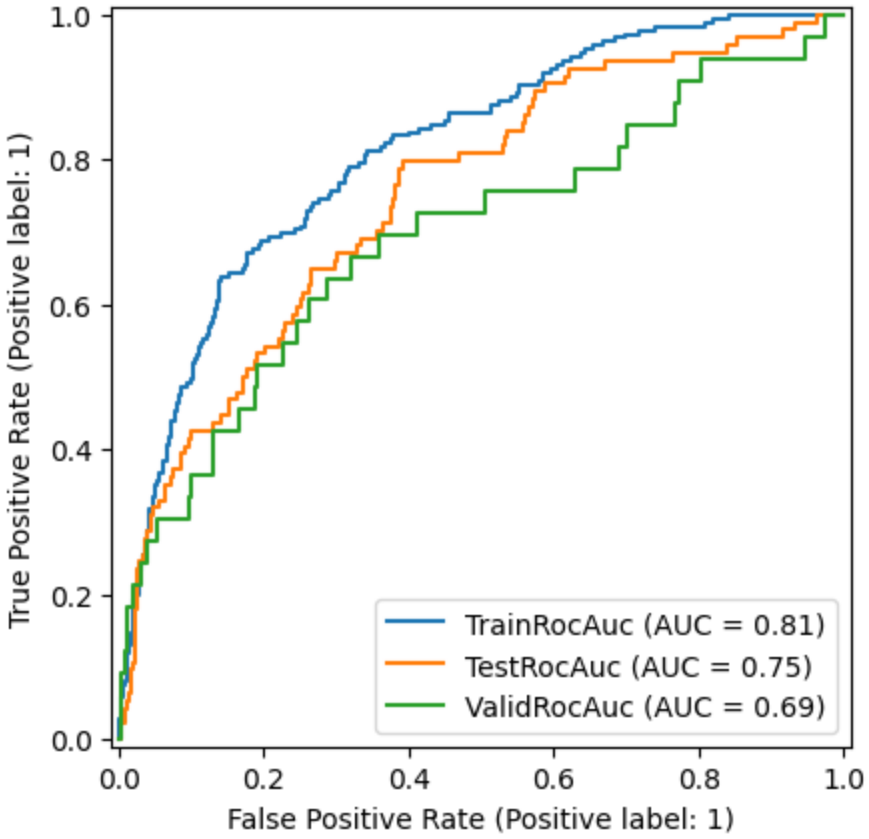
Logistic Regression

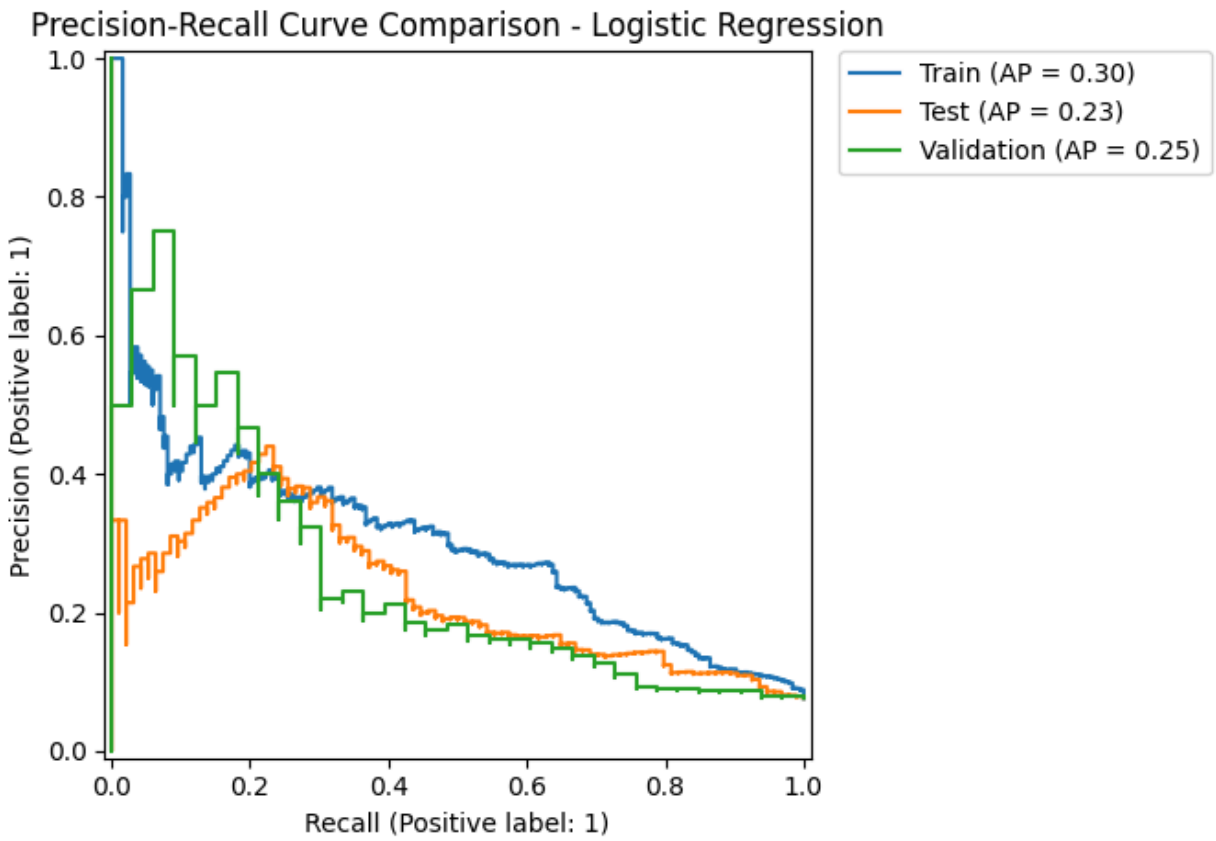
In [ ]:

```
ConductGridSearch(classifiers[0],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START Logistic Regression *****
Parameters:
        C: (10, 1, 0.1, 0.01)
        penalty: ('l1', 'l2', 'elasticnet')
        tol: (0.0001, 1e-05)
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Cross validation with best estimator
Fit and Prediction with best estimator
```


Train
Validation set
Test
No Default
Default
True
Predicted
1
0.0022
0.96
0.038
0.0025
0
0.0026
0.99
0.011


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
TrainRocAuc (AUC = 0.81)
TestRocAuc (AUC = 0.75)
ValidRocAuc (AUC = 0.69)




```
Best Parameters:
        predictor__C: 0.1
        predictor__penalty: l1
        predictor__tol: 0.0001
****** FINISH Logistic Regression  *****
```

In [ ]:

```
expLog
```

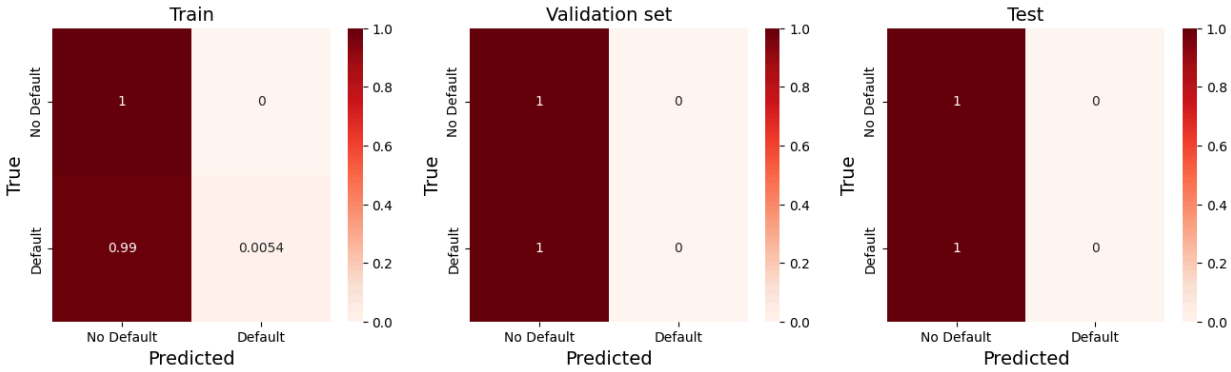
Out[ ]:

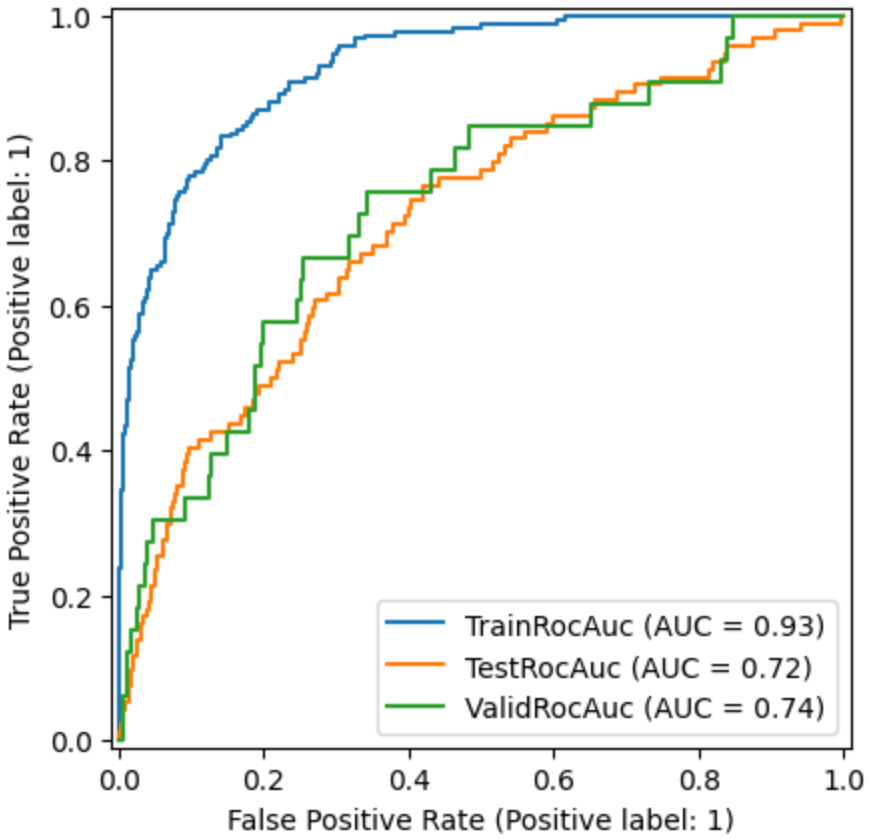
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Train Log Loss
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5634
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6816
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9004
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6394
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5845
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6647
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7196
7	Baseline_132_features	0.9256	0.9219	0.9228	0.8240	0.7427	0.7549	0.0503	0.0237	0.0206	2.6816
8	Logistic RegressionSelectKbest	0.9257	0.9227	0.9220	0.8144	0.7578	0.7500	0.0629	0.0396	0.0204	2.6774

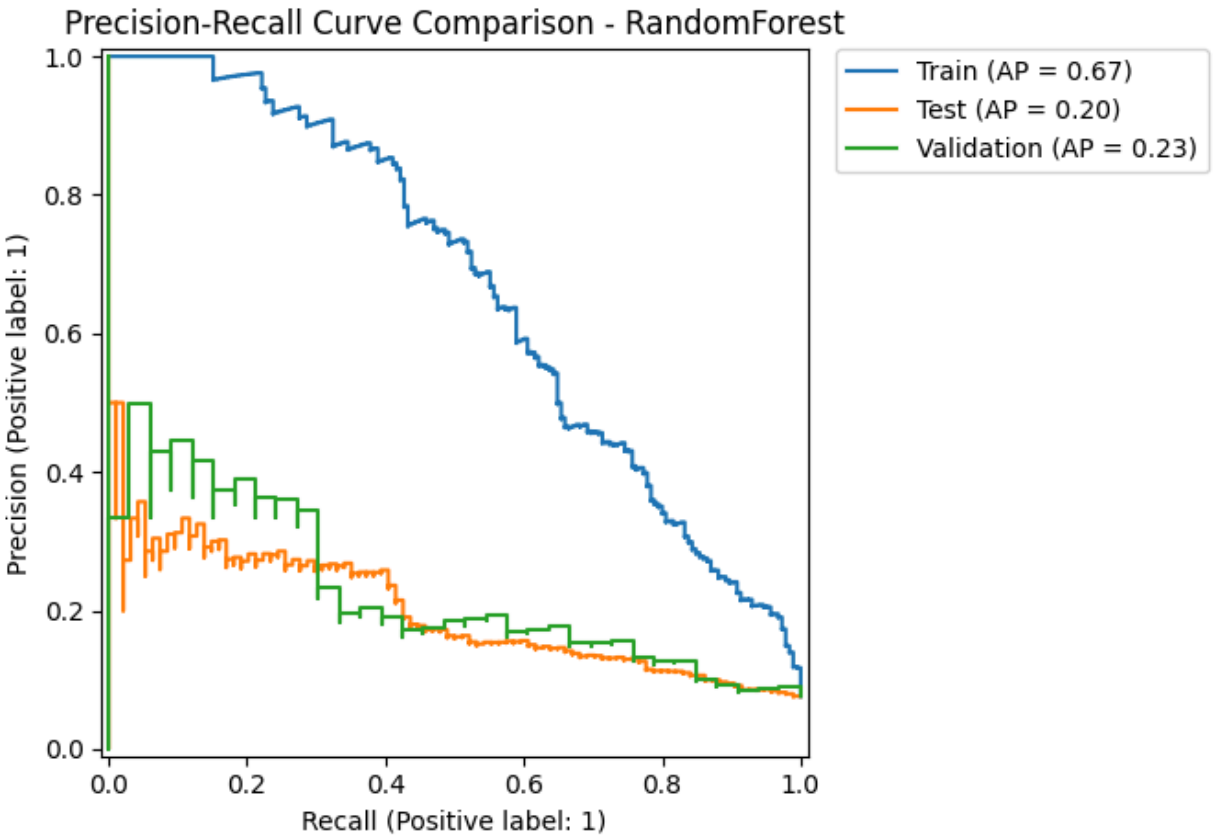
Random Forest

In [ ]:
```
ConductGridSearch(classifiers[3],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START RandomForest *****
Parameters:
        bootstrap: [True]
        max_depth: [5, 10]
        max_features: [15, 20]
        min_samples_leaf: [3, 5]
        min_samples_split: [5, 10]
        n_estimators: [1000]
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Cross validation with best estimator
Fit and Prediction with best estimator
```








```
Best Parameters:
        predictor__bootstrap: True
        predictor__max_depth: 5
        predictor__max_features: 20
        predictor__min_samples_leaf: 5
        predictor__min_samples_split: 5
        predictor__n_estimators: 1000
****** FINISH RandomForest  *****
```

In [ ]:

```
expLog
```

Out[ ]:

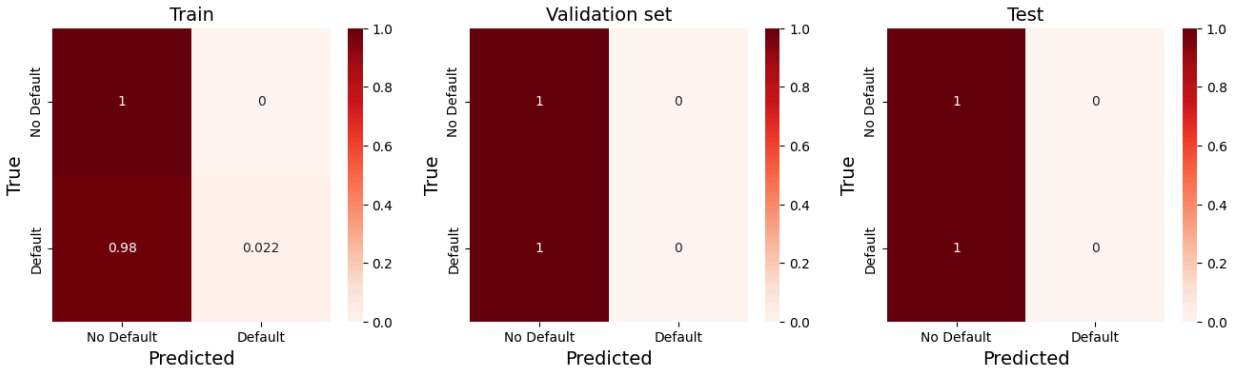
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Tra L Lo
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.56
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.68
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.90
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.63
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.58
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.66
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.71
7	Baseline_132_features	0.9256	0.9219	0.9228	0.8240	0.7427	0.7549	0.0503	0.0237	0.0206	2.68
8	Logistic RegressionSelectKbest	0.9257	0.9227	0.9220	0.8144	0.7578	0.7500	0.0629	0.0396	0.0204	2.67
9	RandomForestSelectKbest	0.9252	0.9232	0.9236	0.9503	0.7373	0.7191	0.0175	0.0000	0.0000	2.69

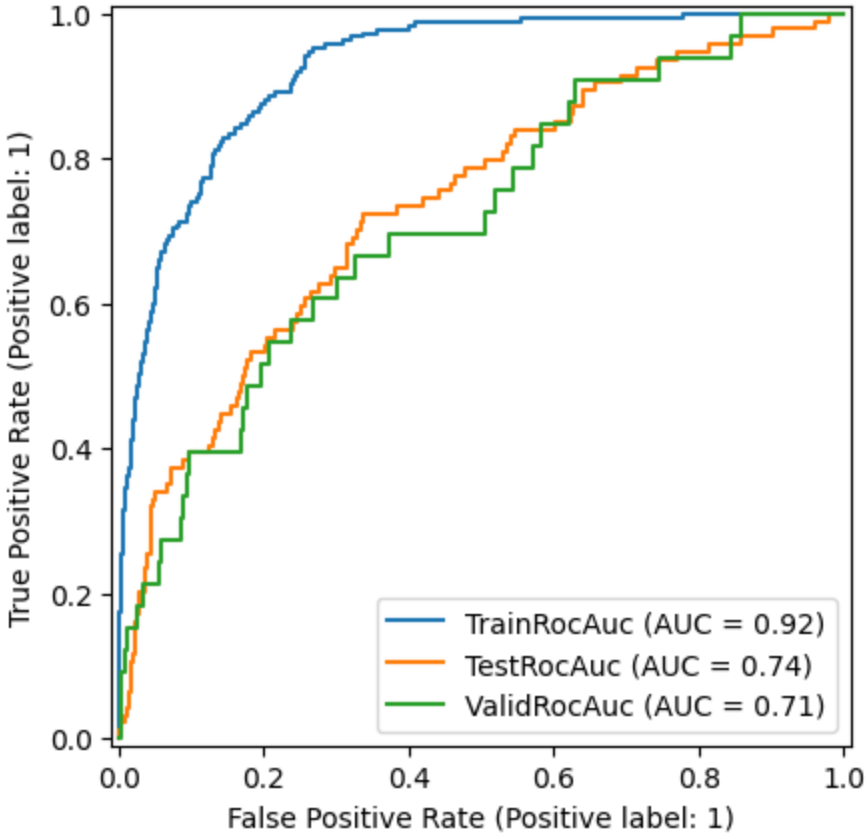
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Tra L Lc

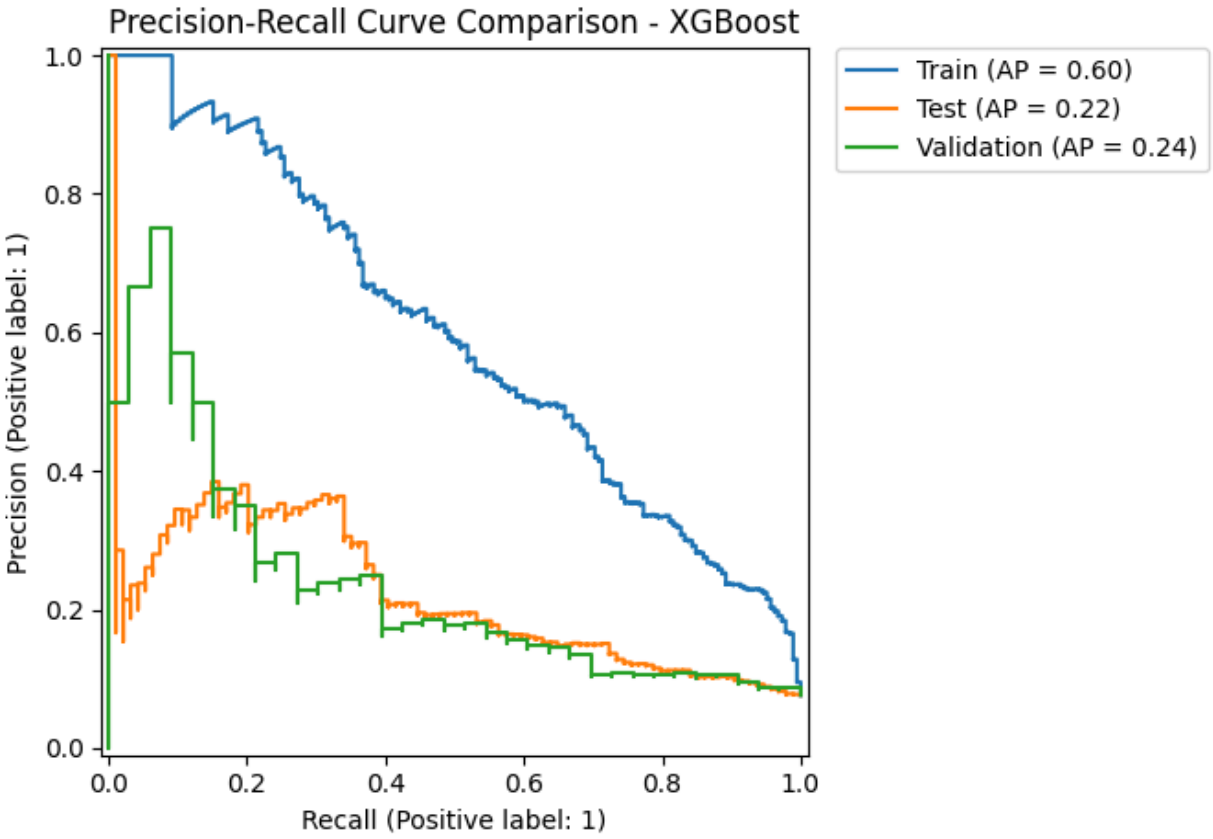
XGBoost

In [ ]:
```
ConductGridSearch(classifiers[1],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START XGBoost *****
Parameters:
        colsample_bytree: [0.2, 0.5]
        eta: [0.01, 0.1]
        learning_rate: [0.01, 0.1]
        max_depth: [3, 5]
        n_estimators: [300, 500]
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Cross validation with best estimator
Fit and Prediction with best estimator
```




True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
TrainRocAuc (AUC = 0.92)
TestRocAuc (AUC = 0.74)
ValidRocAuc (AUC = 0.71)


Precision-Recall Curve Comparison - XGBoost
Precision (Positive label: 1)
Recall (Positive label: 1)
Train (AP = 0.60)
Test (AP = 0.22)
Validation (AP = 0.24)

```
Best Parameters:
        predictor__colsample_bytree: 0.2
        predictor__eta: 0.01
        predictor__learning_rate: 0.01
        predictor__max_depth: 3
        predictor__n_estimators: 300
****** FINISH XGBoost  *****
```

In [ ]:

```
expLog
```

Out[ ]:

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	T l
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7
7	Baseline_132_features	0.9256	0.9219	0.9228	0.8240	0.7427	0.7549	0.0503	0.0237	0.0206	2.6
8	Logistic RegressionSelectKbest	0.9257	0.9227	0.9220	0.8144	0.7578	0.7500	0.0629	0.0396	0.0204	2.6
9	RandomForestSelectKbest	0.9252	0.9232	0.9236	0.9503	0.7373	0.7191	0.0175	0.0000	0.0000	2.6

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	T l
10	XGBoostSelectKbest	0.9268	0.9232	0.9236	0.9460	0.7247	0.7357	0.0554	0.0000	0.0000	2.6

LightGBM

In [ ]:
```
ConductGridSearch(classifiers[2],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START Light GBM *****
Parameters:
        boosting_type: ['goss', 'dart']
        learning_rate: [0.01, 0.1]
        max_bin: [100, 200]
        max_depth: [2, 5]
        n_estimators: [1000, 5000]
        num_leaves: [5, 10]
Fitting 5 folds for each of 64 candidates, totalling 320 fits
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Info] Number of positive: 185, number of negative: 2254
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was
0.001094 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 5101
[LightGBM] [Info] Number of data points in the train set: 2439, number of used featur
es: 117
[LightGBM] [Info] Using GOSS
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.075851 -> initscore=-2.500106
[LightGBM] [Info] Start training from score -2.500106
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Cross validation with best estimator
Fit and Prediction with best estimator
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Info] Number of positive: 185, number of negative: 2254
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was
0.001986 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 5141
[LightGBM] [Info] Number of data points in the train set: 2439, number of used featur
es: 117
[LightGBM] [Info] Using GOSS
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.075851 -> initscore=-2.500106
[LightGBM] [Info] Start training from score -2.500106
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

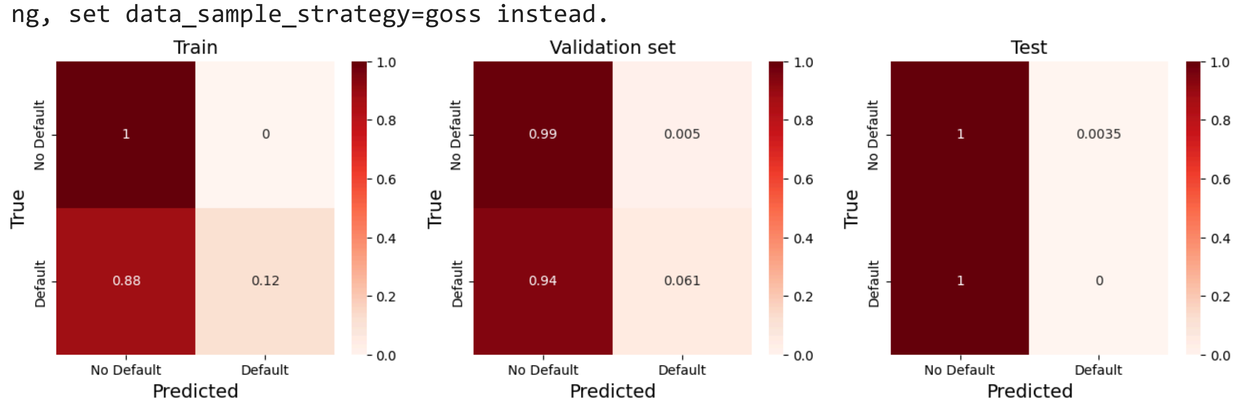
```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

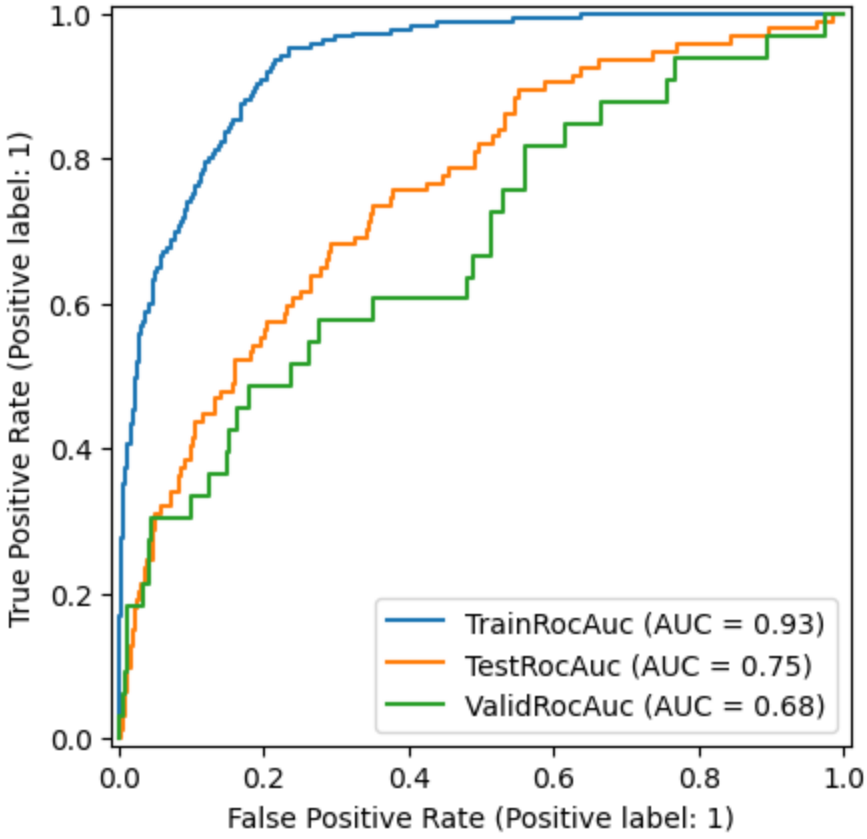
```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

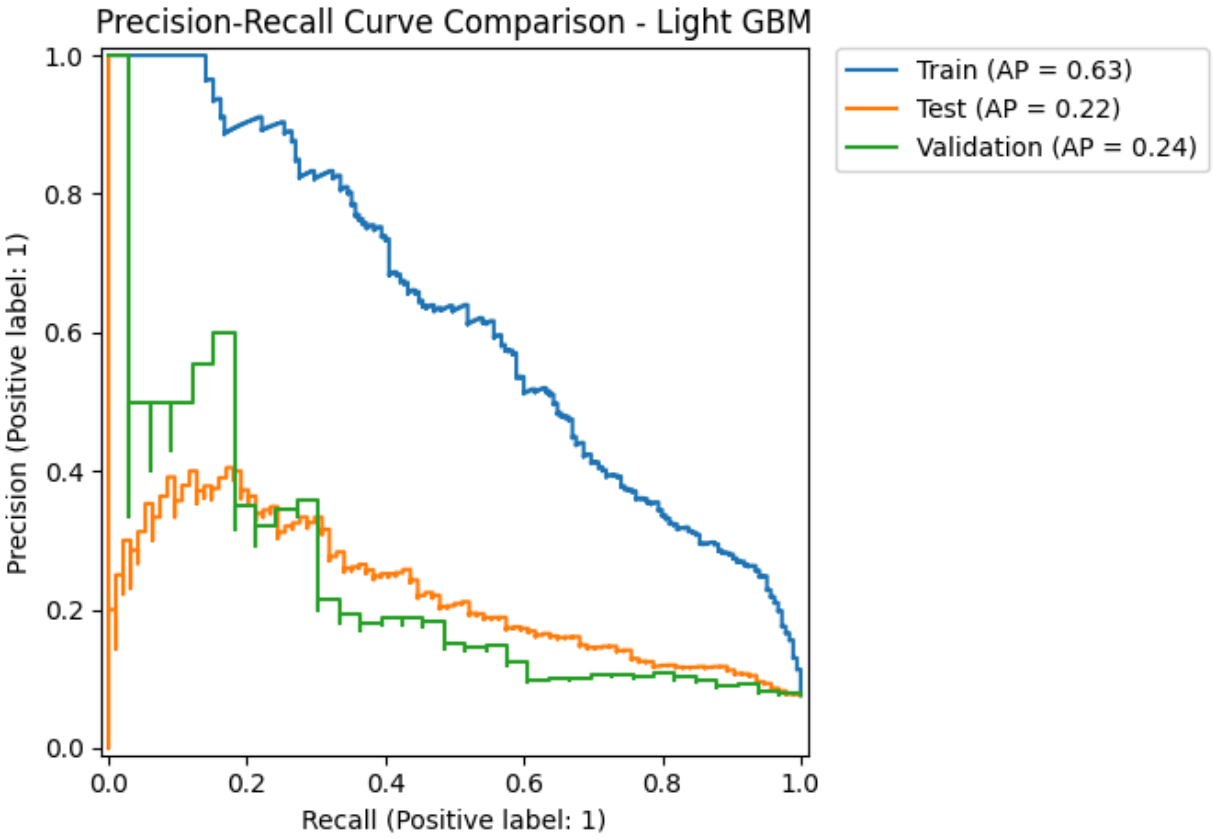
```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
```




True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
TrainRocAuc (AUC = 0.93)
TestRocAuc (AUC = 0.75)
ValidRocAuc (AUC = 0.68)


Precision-Recall Curve Comparison - Light GBM
Precision (Positive label: 1)
Recall (Positive label: 1)
Train (AP = 0.63)
Test (AP = 0.22)
Validation (AP = 0.24)

```
Best Parameters:
        predictor__boosting_type: goss
        predictor__learning_rate: 0.01
        predictor__max_bin: 100
        predictor__max_depth: 2
        predictor__n_estimators: 1000
        predictor__num_leaves: 10
****** FINISH Light GBM  *****

[LightGBM] [Warning] Found boosting=goss. For backwards compatibility reasons, LightG
BM interprets this as boosting=gbdt, data_sample_strategy=goss.To suppress this warni
ng, set data_sample_strategy=goss instead.
```

In [ ]:
```
!cat /proc/meminfo | grep MemAvailable
```

```
MemAvailable:   331093228 kB
```

In [ ]:
```
expLog
```

Out[ ]:

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	T l
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7
7	Baseline_132_features	0.9256	0.9219	0.9228	0.8240	0.7427	0.7549	0.0503	0.0237	0.0206	2.6
8	Logistic RegressionSelectKbest	0.9257	0.9227	0.9220	0.8144	0.7578	0.7500	0.0629	0.0396	0.0204	2.6
9	RandomForestSelectKbest	0.9252	0.9232	0.9236	0.9503	0.7373	0.7191	0.0175	0.0000	0.0000	2.6

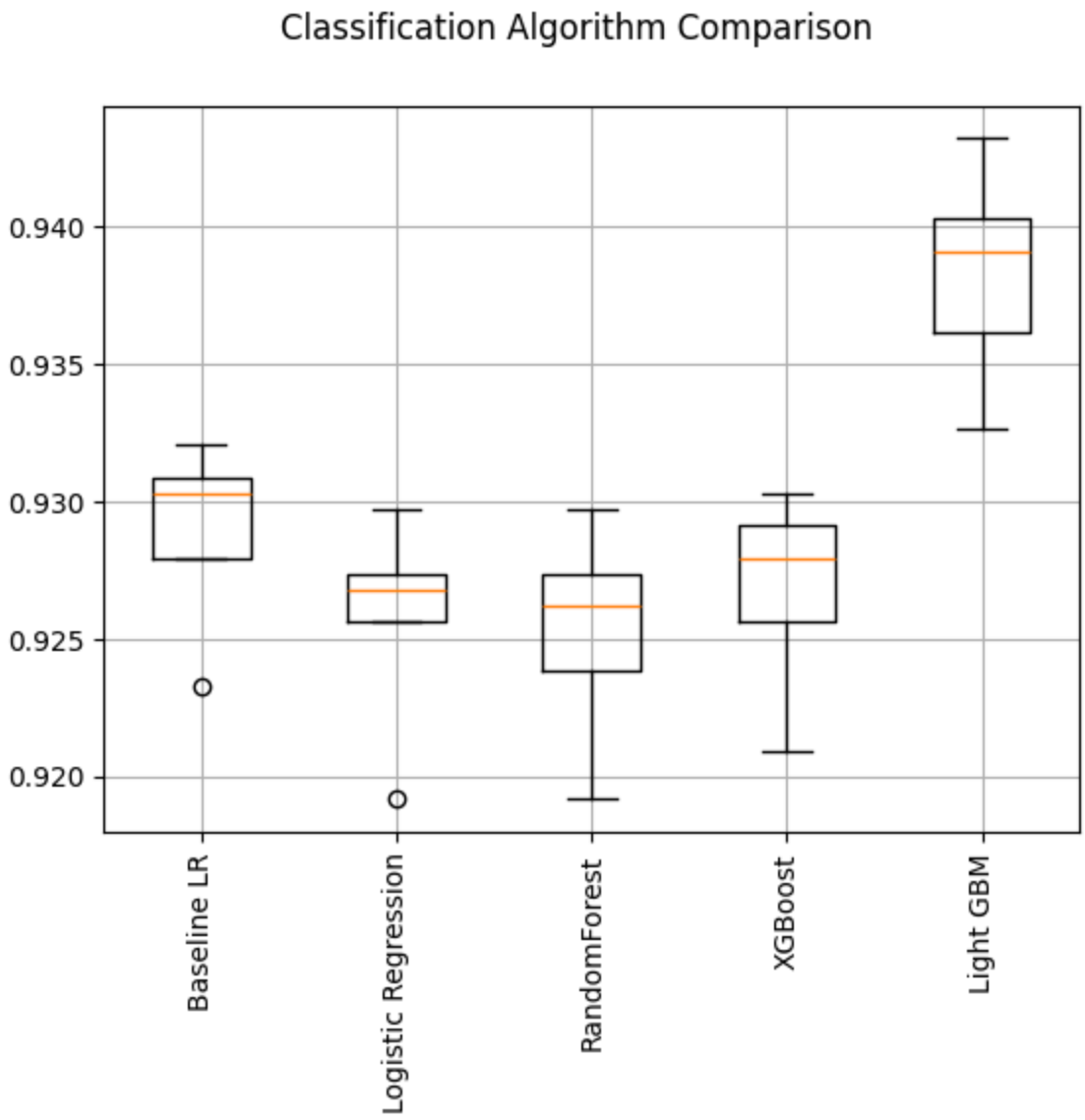
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	T l
10	XGBoostSelectKbest	0.9268	0.9232	0.9236	0.9460	0.7247	0.7357	0.0554	0.0000	0.0000	2.6
11	Light GBMSelectKbest	0.9383	0.9208	0.9204	0.9583	0.7179	0.7523	0.3056	0.0410	0.0000	2.2

Model Validation

Boxplot with all CV results

In [ ]:

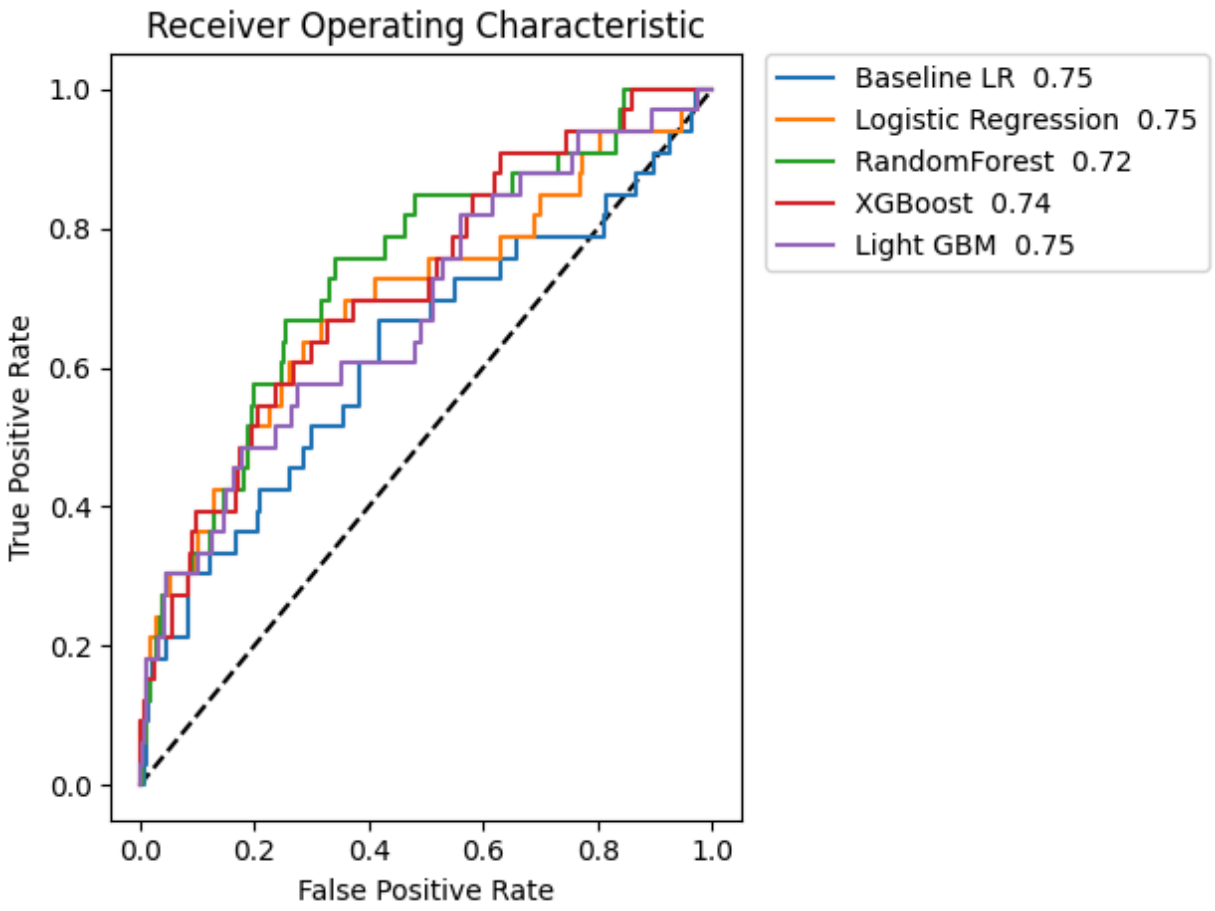
```
# boxplot algorithm comparison
fig = pyplot.figure()
fig.suptitle('Classification Algorithm Comparison')
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names,rotation=90)
pyplot.grid()
pyplot.show()
```



AUC (Area Under the ROC Curve)

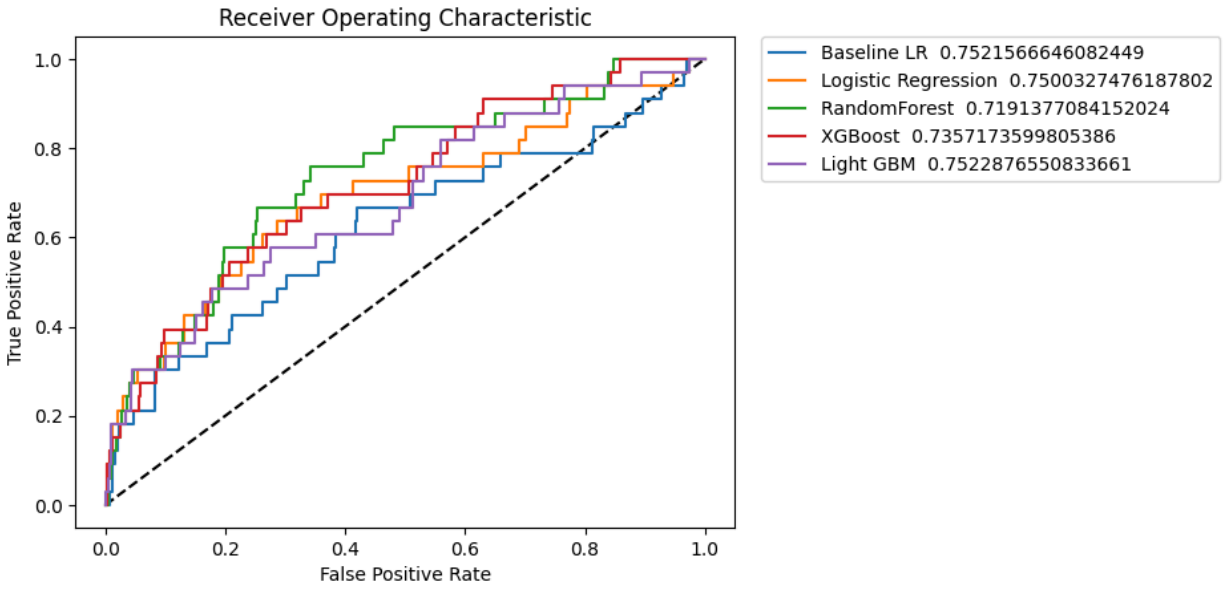
In [ ]:

```
# ROC curve: plot all models
plt.plot([0, 1], [0, 1], 'k--')

for name, fpr, tpr, score in zip(names, fprs, tprs, scores):
    plt.plot(fpr, tpr, label=f"{name}  {score:.2f}")

plt.legend(bbox_to_anchor=(1.04, 1), loc="upper left", borderaxespad=0)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver Operating Characteristic")
plt.tight_layout()
plt.show()
```



In [ ]:

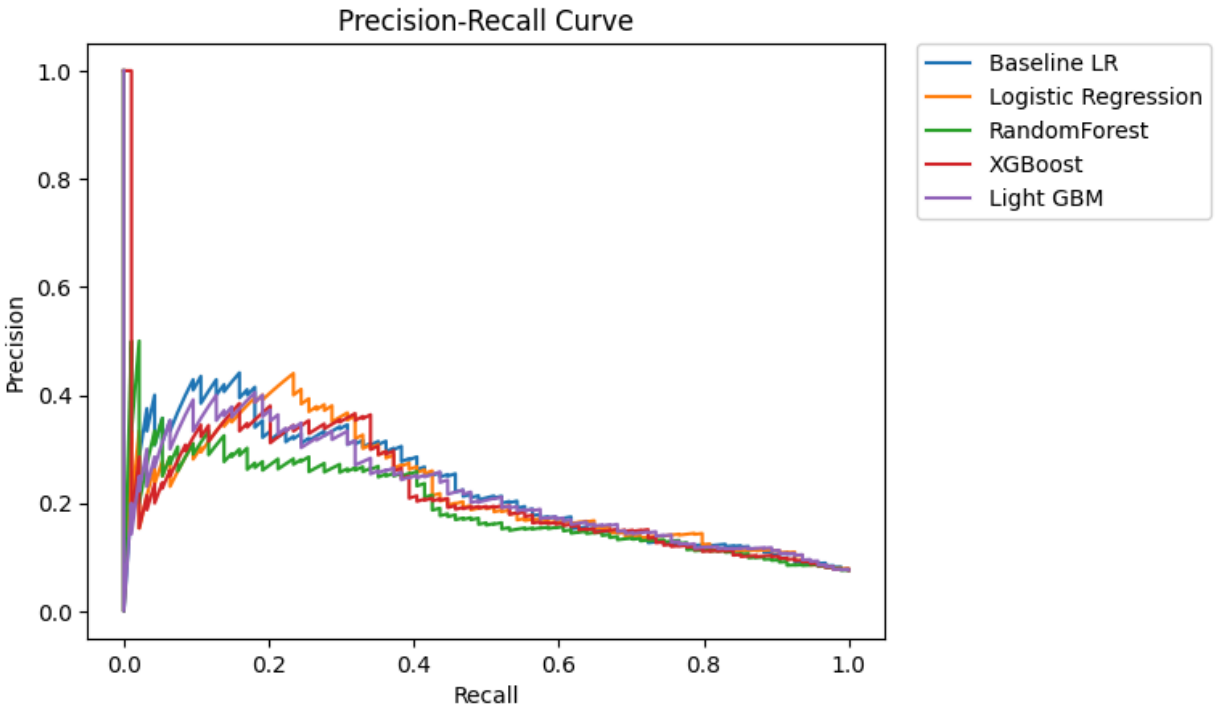
```
# roc curve fpr, tpr  for all classifiers  -- Delete
plt.plot([0,1],[0,1], 'k--')
for i in range(len(names)):
    plt.plot(fprs[i],tprs[i],label = names[i] + '   ' + str(scores[i]))
plt.legend(bbox_to_anchor=(1.04,1), loc="upper left", borderaxespad=0)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title('Receiver Operating Characteristic')
plt.show()
```



Precision Recall Curve

In [ ]:

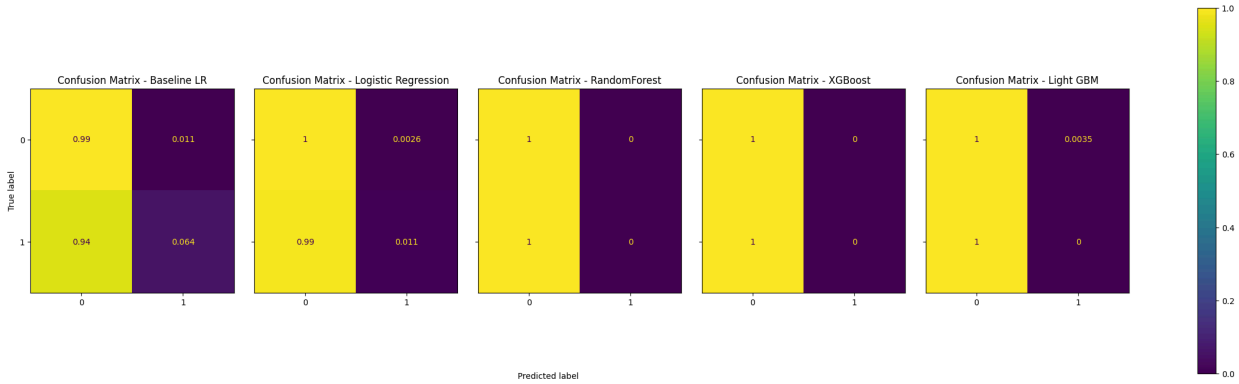
```
# precision recall curve  for all classifiers
for i in range(len(names)):
    plt.plot(recalls[i],precisions[i],label = names[i])
plt.legend(bbox_to_anchor=(1.04,1), loc="upper left", borderaxespad=0)
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title('Precision-Recall Curve')
plt.show()
```



Confusion Matrix

In [ ]:

```
# plot confusion matrix for all classifiers
f, axes = plt.subplots(1, len(names), figsize=(30, 8), sharey='row')
for i in range(len(names)):
    disp = ConfusionMatrixDisplay(cnfmatrix[i], display_labels=['0', '1'])
    disp.plot(ax=axes[i], xticks_rotation=0)
    disp.ax_.set_title("Confusion Matrix - " + names[i])
    disp.im_.colorbar.remove()
    disp.ax_.set_xlabel('')
    if i!=0:
        disp.ax_.set_ylabel('')

f.text(0.4, 0.1, 'Predicted label', ha='left')
plt.subplots_adjust(wspace=0.10, hspace=0.1)

f.colorbar(disp.im_, ax=axes)
plt.show()
```



Final results

In [ ]:

```
pd.set_option('display.max_colwidth', None)
expLog.to_csv("/content/Data/expLog_SelectKbest.csv",index=False)
expLog
```

Out[ ]:

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	T l
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7
7	Baseline_132_features	0.9256	0.9219	0.9228	0.8240	0.7427	0.7549	0.0503	0.0237	0.0206	2.6
8	Logistic RegressionSelectKbest	0.9257	0.9227	0.9220	0.8144	0.7578	0.7500	0.0629	0.0396	0.0204	2.6
9	RandomForestSelectKbest	0.9252	0.9232	0.9236	0.9503	0.7373	0.7191	0.0175	0.0000	0.0000	2.6

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Tı L
10	XGBoostSelectKbest	0.9268	0.9232	0.9236	0.9460	0.7247	0.7357	0.0554	0.0000	0.0000	2.6
11	Light GBMSelectKbest	0.9383	0.9208	0.9204	0.9583	0.7179	0.7523	0.3056	0.0410	0.0000	2.2

Models (Grid search & Variance Threshold Feature selection)

Various Classification algorithms were used to compare with the best model using Variance Thershold feature selection technique.

Variance Threshold The variance threshold is a simple baseline approach to feature selection. It removes all features which variance doesn't meet some threshold. By default, it removes all zero-variance features, i.e., features that have the same value in all samples. We assume that features with a higher variance may contain more useful information, but note that we are not taking the relationship between feature variables or feature and target variables into account, which is one of the drawbacks of filter methods.

Following metrics were used to find the best model

- Cross fold Train Accuracy
- Test Accuracy
- p-value
- Train ROC_AUC_Score
- Test ROC_AUC_Score
- Train F1_Score
- Test F1_Score
- Train LogLoss
- Test LogLoss

- Train Time
- Test Time
- Confusion matrix

In [ ]:
```
# Clean up the  arrays

del fprs[1:]
del tprs[1:]
del precisions[1:]
del recalls[1:]
del names[1:]
del scores[1:]
del cvscores[1:]
del pvalues[1:]
del accuracy[1:]
del cnfmatrix[1:]
del results[1:]
final_best_clf = {}
```

Classifiers

In [ ]:
```
classifiers = [
        [('Logistic Regression', LogisticRegression(solver='saga',random_state=42),"V
        [('XGBoost', XGBClassifier(random_state=42),"VarianceThreshold")],
        [('Light GBM', LGBMClassifier(boosting_type='gbdt', random_state=42),"Varianc
        [('RandomForest', RandomForestClassifier(random_state=42),"VarianceThreshold"
    ]
```

Hyper-parameters for different models

In [ ]:
```
# Arrange grid search parameters for each classifier
params_grid = {
        'Logistic Regression': {
            'penalty': ('l1', 'l2','elasticnet'),
            'tol': (0.0001, 0.00001),
            'C': (10, 1, 0.1, 0.01),
        }
    ,
        'XGBoost':  {
            'max_depth': [3,5], # Lower helps with overfitting
            'n_estimators':[300,500],
            'learning_rate': [0.01,0.1],
#             'objective': ['binary:logistic'],
#             'eval_metric': ['auc'],
            'eta' : [0.01,0.1],
            'colsample_bytree' : [0.2,0.5],
        },
        'Light GBM':  {
            'max_depth': [2,5],  # Lower helps with overfitting
            'num_leaves': [5,10], # Equivalent to max depth
            'n_estimators':[1000,5000],
            'learning_rate': [0.01,0.1],
 #            'reg_alpha': [0.1,0.01,1],
 #            'reg_lambda': [0.1,0.01,1],
```

```
            'boosting_type':['goss','dart'],
#           'metric': ['auc'],
#           'objective':['binary'],
            'max_bin' : [100,200],  #Setting it to high values has similar effect as
        },                          #small numbers reduces accuracy but runs faster

        'RandomForest':  {
            'max_depth': [5,10],
            'max_features': [15,20],
            'min_samples_split': [5, 10],
            'min_samples_leaf': [3, 5],
            'bootstrap': [True],
            'n_estimators':[1000]},
    }
```

Conduct Grid Search

In [ ]:

```
def ConductGridSearch(in_classifiers,cnfmatrix,fprs,tprs,precisions,recalls):
    for (name, classifier,feature_sel) in in_classifiers:
            # Print classifier and parameters
            print('****** START', name,'*****')
            parameters = params_grid[name]
            print("Parameters:")
            for p in sorted(parameters.keys()):
                print("\t"+str(p)+": "+ str(parameters[p]))

            # generate the pipeline based on the feature selection method
            full_pipeline_with_predictor = Pipeline([
                ("preparation", data_prep_pipeline),
                ("VarianceThreshold", VarianceThreshold(threshold=0.9)),
                ("predictor", classifier)
                ])

            # Execute the grid search
            params = {}
            for p in parameters.keys():
                pipe_key = 'predictor__'+str(p)
                params[pipe_key] = parameters[p]
            grid_search = GridSearchCV(full_pipeline_with_predictor, params, cv=cvSpl
                                       n_jobs=-1,verbose=1)
            grid_search.fit(X_train, y_train)

            # Best estimator score
            best_train = pct(grid_search.best_score_)

            # Best train scores
            print("Cross validation with best estimator")
            best_train_scores = cross_validate(grid_search.best_estimator_, X_train,
                                               return_train_score=True, n_jobs=-1)

            #get all scores
            best_train_accuracy = np.round(best_train_scores['train_accuracy'].mean()
            best_train_f1 = np.round(best_train_scores['train_f1'].mean(),4)
            best_train_logloss = np.round(best_train_scores['train_log_loss'].mean(),
            best_train_roc_auc = np.round(best_train_scores['train_roc_auc'].mean(),4

            valid_time = np.round(best_train_scores['score_time'].mean(),4)
            best_valid_accuracy = np.round(best_train_scores['test_accuracy'].mean(),
```

```
best_valid_f1 = np.round(best_train_scores['test_f1'].mean(),4)
best_valid_logloss = np.round(best_train_scores['test_log_loss'].mean(),4
best_valid_roc_auc = np.round(best_train_scores['test_roc_auc'].mean(),4)

#append all results
results.append(best_train_scores['train_accuracy'])
names.append(name)

# Conduct t-test with baseline logit (control) and best estimator (experi
(t_stat, p_value) = stats.ttest_rel(logit_scores['train_roc_auc'], best_t

#test and Prediction with whole data
# Best estimator fitting time
print("Fit and Prediction with best estimator")
start = time()
model = grid_search.best_estimator_.fit(X_train, y_train)
train_time = round(time() - start, 4)

# Best estimator prediction time
start = time()
y_test_pred = model.predict(X_test)
test_time = round(time() - start, 4)
scores.append(roc_auc_score(y_test, model.predict_proba(X_test)[:, 1]))
accuracy.append(accuracy_score(y_test, y_test_pred))

# Create confusion matrix for the best model
cnfmatrix = confusion_matrix_def(model,X_train,y_train,X_test,y_test,X_va

# Create AUC ROC curve
fprs,tprs = roc_curve_cust(model,X_train,y_train,X_test, y_test,X_valid,

#Create Precision recall curve
precisions,recalls = precision_recall_cust(model,X_train,y_train,X_test,

#Best Model
final_best_clf[name]=pd.DataFrame([{'label': grid_search.best_estimator_.
                                   'predictor': grid_search.best_estimator_.n

# Collect the best parameters found by the grid search
print("Best Parameters:")
best_parameters = grid_search.best_estimator_.get_params()
param_dump = []
for param_name in sorted(params.keys()):
    param_dump.append((param_name, best_parameters[param_name]))
    print("\t"+str(param_name)+": " + str(best_parameters[param_name]))
print("****** FINISH",name," *****")
print("")

# Record the results
exp_name = name+str('Variance')
expLog.loc[len(expLog)] = [f"{exp_name}"] + list(np.round(
        [best_train_accuracy,
        #pct(accuracy_score(y_valid, model.predict(X_valid))),
        best_valid_accuracy,
        accuracy_score(y_test, y_test_pred),
        best_train_roc_auc,
        best_valid_roc_auc,
        #roc_auc_score(y_valid, model.predict_proba(X_valid)[:, 1]),
        roc_auc_score(y_test, model.predict_proba(X_test)[:, 1]),
        best_train_f1,
```

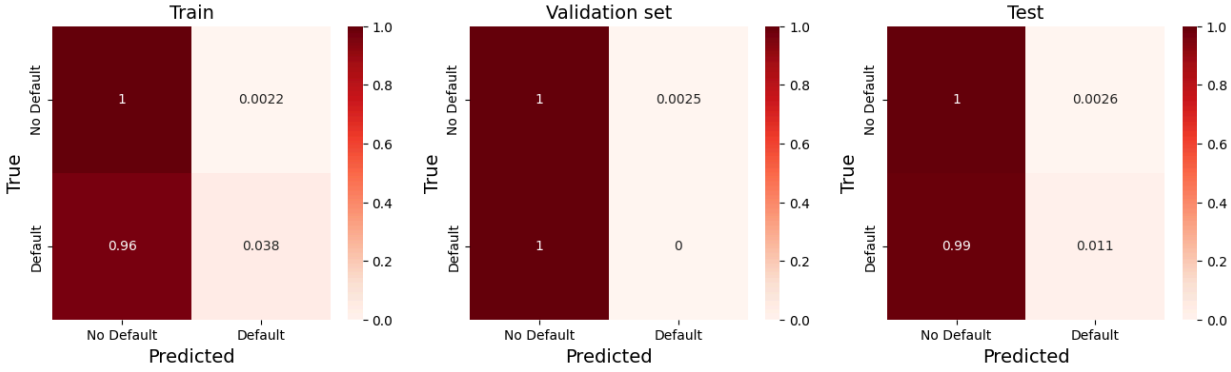
```
                best_valid_f1,
                f1_score(y_test, y_test_pred),
                best_train_logloss,
                best_valid_logloss,
                log_loss(y_test, y_test_pred),
                p_value
                ],4)) + [train_time,valid_time,test_time] \
                + [json.dumps(param_dump)]
```

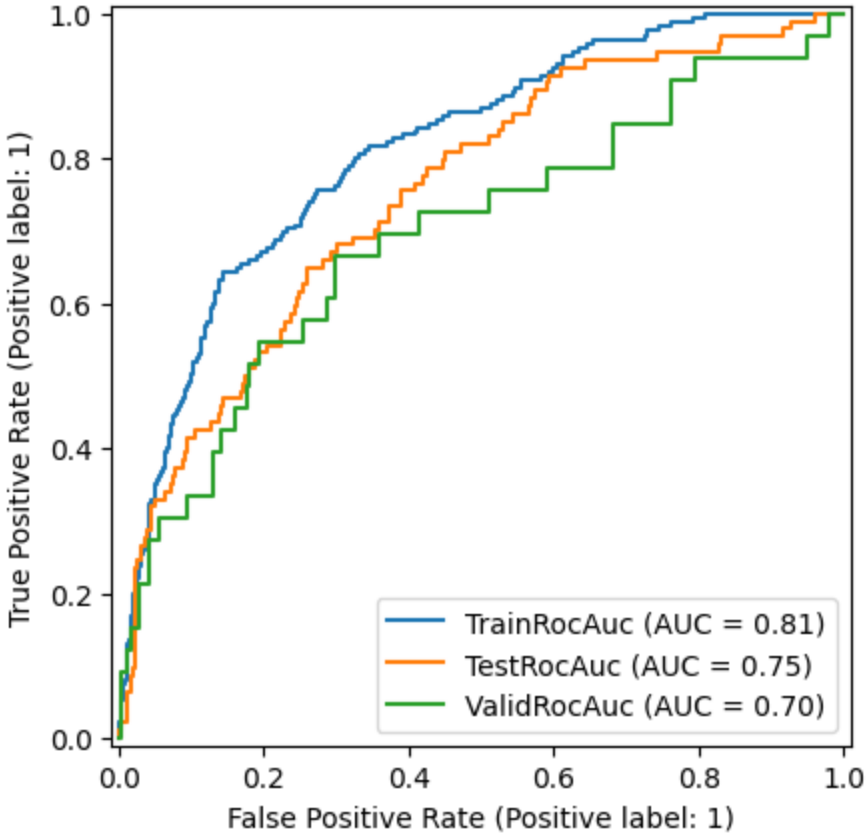
Logistic Regression

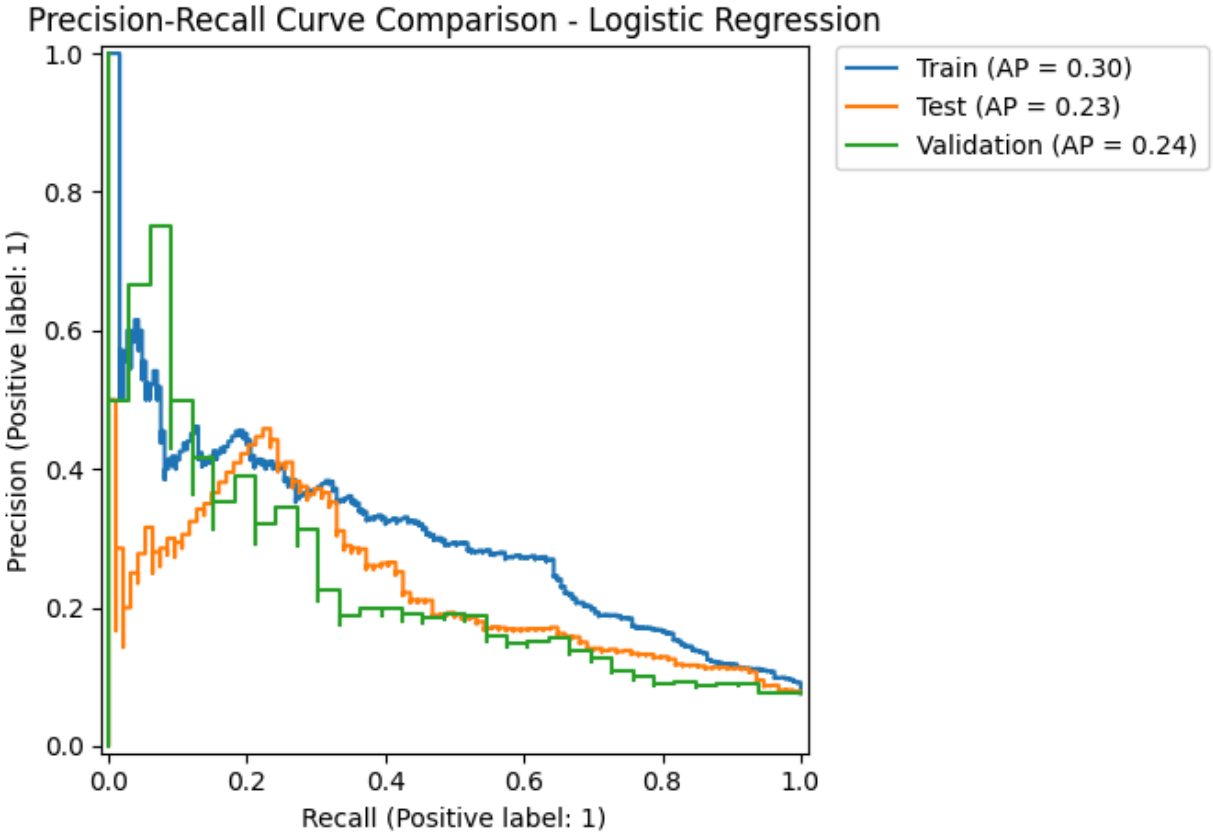
In [ ]:

```
ConductGridSearch(classifiers[0],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START Logistic Regression *****
Parameters:
        C: (10, 1, 0.1, 0.01)
        penalty: ('l1', 'l2', 'elasticnet')
        tol: (0.0001, 1e-05)
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Cross validation with best estimator
Fit and Prediction with best estimator
```








```
Best Parameters:
        predictor__C: 0.1
        predictor__penalty: l1
        predictor__tol: 0.0001
****** FINISH Logistic Regression  *****
```

In [ ]:
```
expLog
```

Out[ ]:

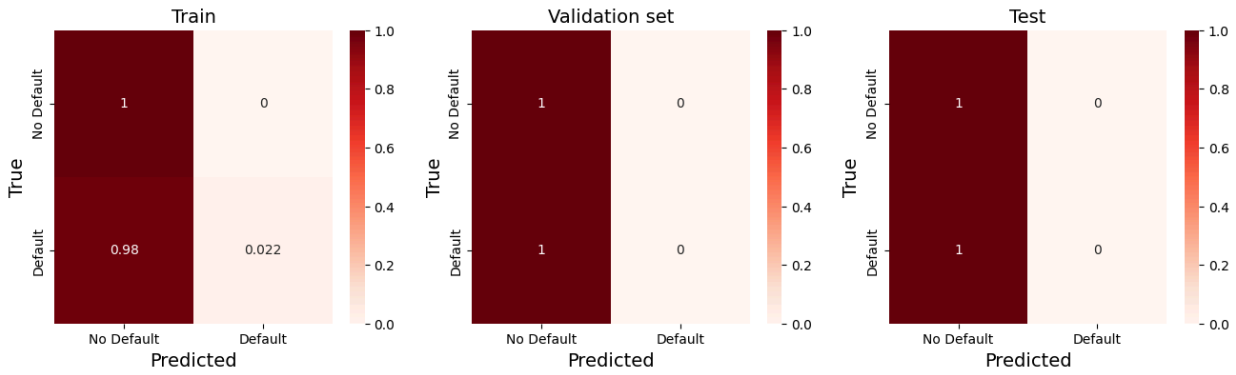
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	T l
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7
7	Baseline_132_features	0.9256	0.9219	0.9228	0.8240	0.7427	0.7549	0.0503	0.0237	0.0206	2.6
8	Logistic RegressionSelectKbest	0.9257	0.9227	0.9220	0.8144	0.7578	0.7500	0.0629	0.0396	0.0204	2.6
9	RandomForestSelectKbest	0.9252	0.9232	0.9236	0.9503	0.7373	0.7191	0.0175	0.0000	0.0000	2.6

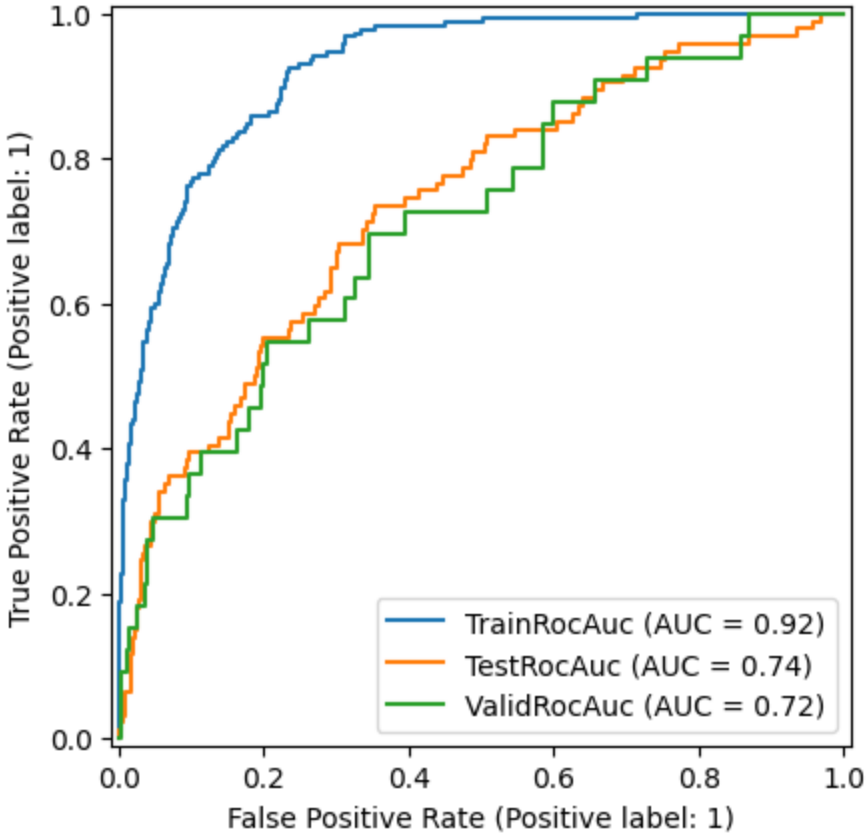
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Tı l
10	XGBoostSelectKbest	0.9268	0.9232	0.9236	0.9460	0.7247	0.7357	0.0554	0.0000	0.0000	2.6
11	Light GBMSelectKbest	0.9383	0.9208	0.9204	0.9583	0.7179	0.7523	0.3056	0.0410	0.0000	2.2
12	Logistic RegressionVariance	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6

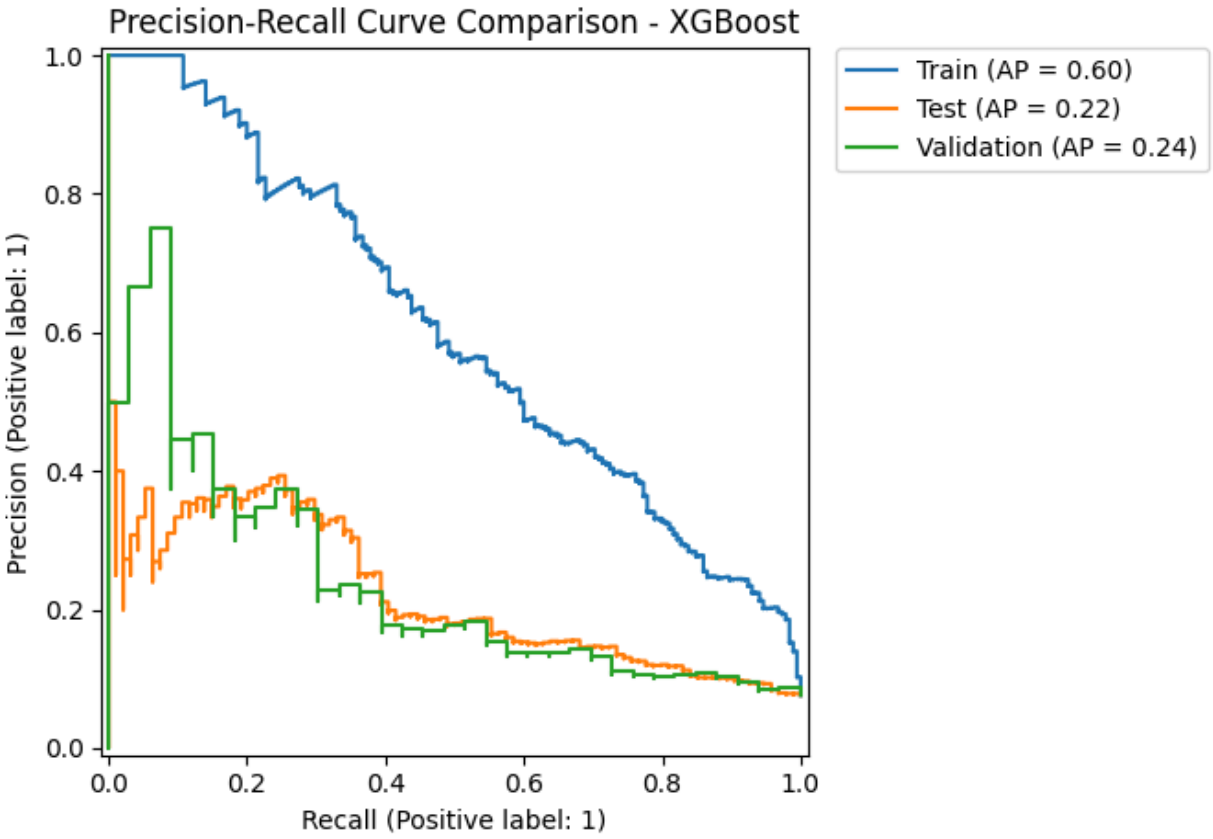
XGBoost

In [ ]:
```
ConductGridSearch(classifiers[1],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START XGBoost *****
Parameters:
        colsample_bytree: [0.2, 0.5]
        eta: [0.01, 0.1]
        learning_rate: [0.01, 0.1]
        max_depth: [3, 5]
        n_estimators: [300, 500]
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Cross validation with best estimator
Fit and Prediction with best estimator
```




True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
TrainRocAuc (AUC = 0.92)
TestRocAuc (AUC = 0.74)
ValidRocAuc (AUC = 0.72)


Precision-Recall Curve Comparison - XGBoost
Precision (Positive label: 1)
Recall (Positive label: 1)
Train (AP = 0.60)
Test (AP = 0.22)
Validation (AP = 0.24)

```
Best Parameters:
        predictor__colsample_bytree: 0.2
        predictor__eta: 0.01
        predictor__learning_rate: 0.01
        predictor__max_depth: 3
        predictor__n_estimators: 300
****** FINISH XGBoost  *****
```

In [ ]:

```
expLog
```

Out[ ]:

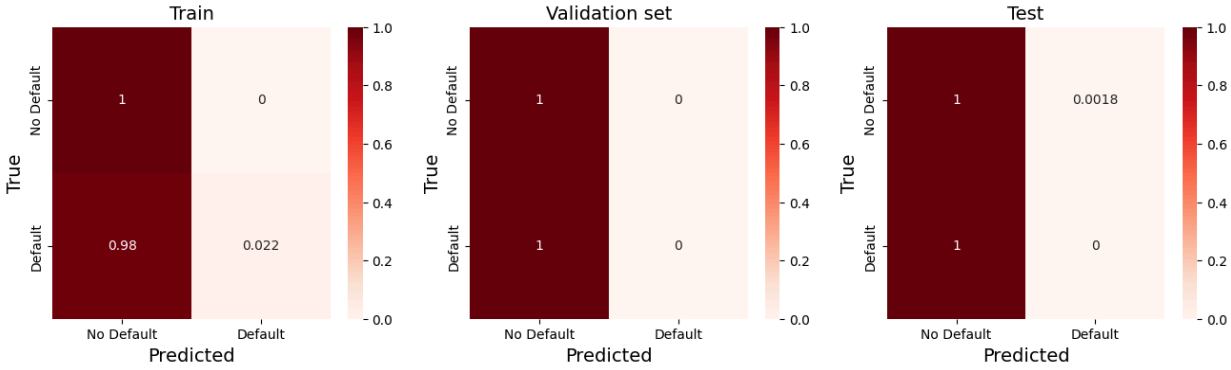
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Tı l
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7
7	Baseline_132_features	0.9256	0.9219	0.9228	0.8240	0.7427	0.7549	0.0503	0.0237	0.0206	2.6
8	Logistic RegressionSelectKbest	0.9257	0.9227	0.9220	0.8144	0.7578	0.7500	0.0629	0.0396	0.0204	2.6
9	RandomForestSelectKbest	0.9252	0.9232	0.9236	0.9503	0.7373	0.7191	0.0175	0.0000	0.0000	2.6

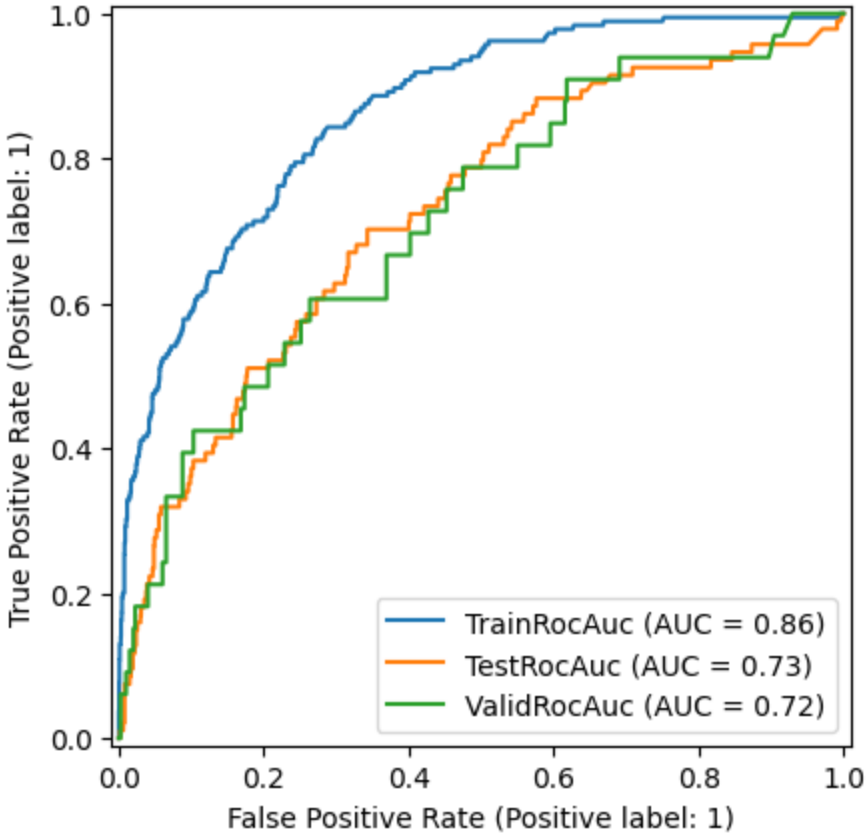
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Tı l
10	XGBoostSelectKbest	0.9268	0.9232	0.9236	0.9460	0.7247	0.7357	0.0554	0.0000	0.0000	2.6
11	Light GBMSelectKbest	0.9383	0.9208	0.9204	0.9583	0.7179	0.7523	0.3056	0.0410	0.0000	2.2
12	Logistic RegressionVariance	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6
13	XGBoostVariance	0.9268	0.9232	0.9236	0.9491	0.7280	0.7357	0.0557	0.0000	0.0000	2.6

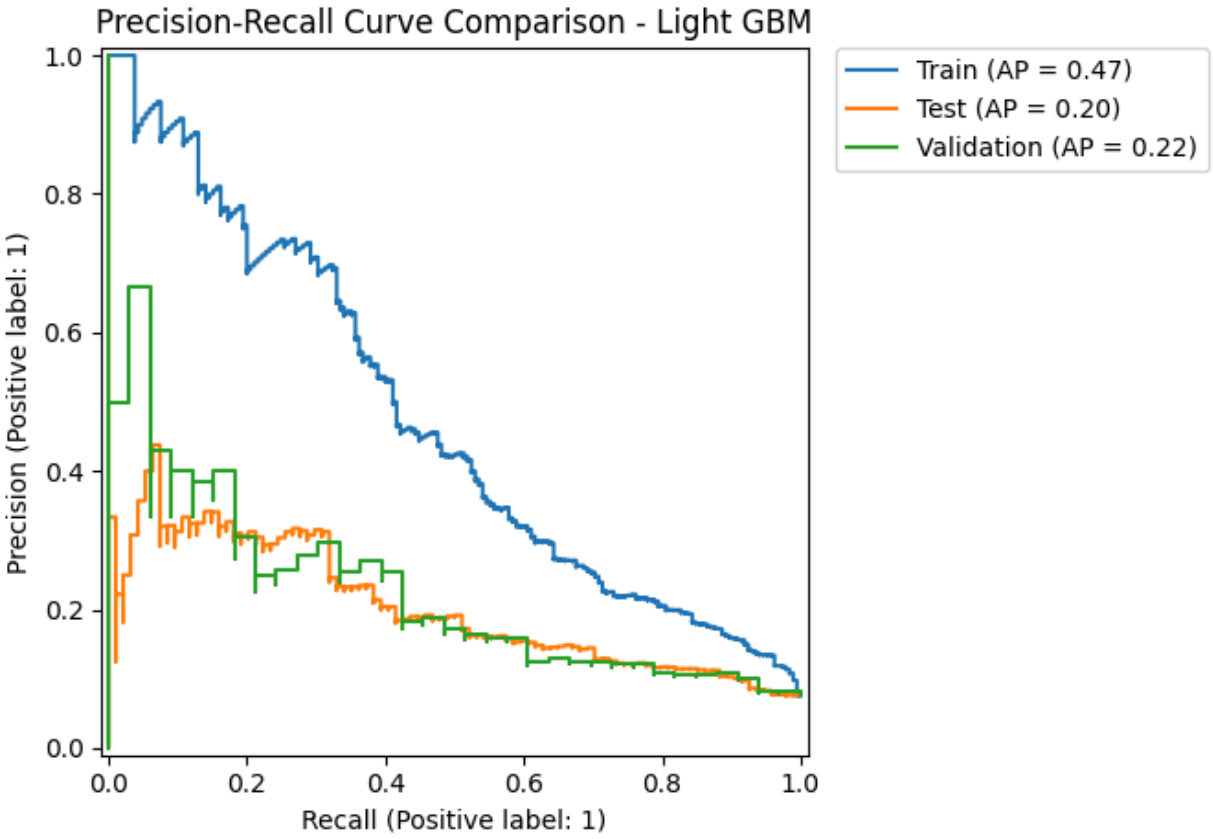
Light GBM

In [ ]:
```
ConductGridSearch(classifiers[2],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START Light GBM *****
Parameters:
        boosting_type: ['goss', 'dart']
        learning_rate: [0.01, 0.1]
        max_bin: [100, 200]
        max_depth: [2, 5]
        n_estimators: [1000, 5000]
        num_leaves: [5, 10]
Fitting 5 folds for each of 64 candidates, totalling 320 fits
[LightGBM] [Info] Number of positive: 185, number of negative: 2254
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was
0.001874 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 5475
[LightGBM] [Info] Number of data points in the train set: 2439, number of used featur
es: 123
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.075851 -> initscore=-2.500106
[LightGBM] [Info] Start training from score -2.500106
Cross validation with best estimator
Fit and Prediction with best estimator
[LightGBM] [Info] Number of positive: 185, number of negative: 2254
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was
0.001040 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 5475
[LightGBM] [Info] Number of data points in the train set: 2439, number of used featur
es: 123
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.075851 -> initscore=-2.500106
[LightGBM] [Info] Start training from score -2.500106
```




True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
TrainRocAuc (AUC = 0.86)
TestRocAuc (AUC = 0.73)
ValidRocAuc (AUC = 0.72)


Precision-Recall Curve Comparison - Light GBM
Precision (Positive label: 1)
Recall (Positive label: 1)
Train (AP = 0.47)
Test (AP = 0.20)
Validation (AP = 0.22)

```
Best Parameters:
        predictor__boosting_type: dart
        predictor__learning_rate: 0.01
        predictor__max_bin: 100
        predictor__max_depth: 5
        predictor__n_estimators: 1000
        predictor__num_leaves: 5
****** FINISH Light GBM  *****
```

In [ ]:

```
expLog
```

Out[ ]:

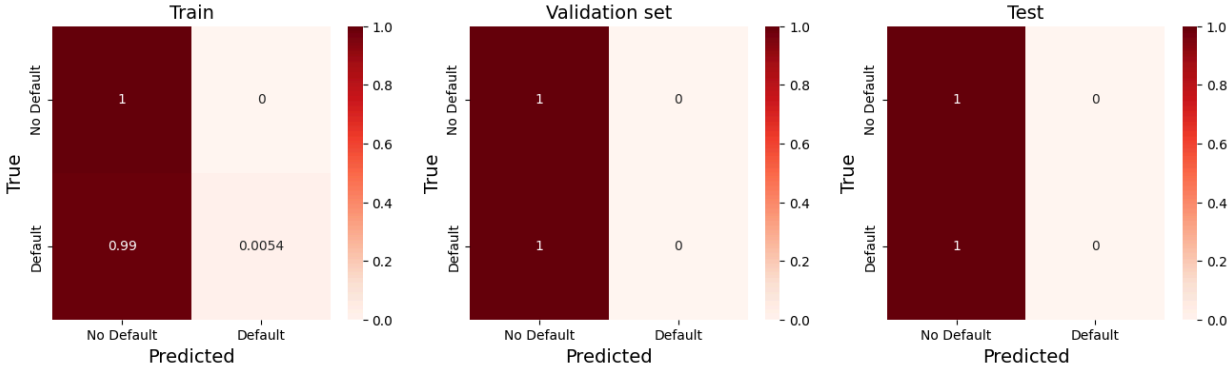
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	T l
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7
7	Baseline_132_features	0.9256	0.9219	0.9228	0.8240	0.7427	0.7549	0.0503	0.0237	0.0206	2.6
8	Logistic RegressionSelectKbest	0.9257	0.9227	0.9220	0.8144	0.7578	0.7500	0.0629	0.0396	0.0204	2.6
9	RandomForestSelectKbest	0.9252	0.9232	0.9236	0.9503	0.7373	0.7191	0.0175	0.0000	0.0000	2.6

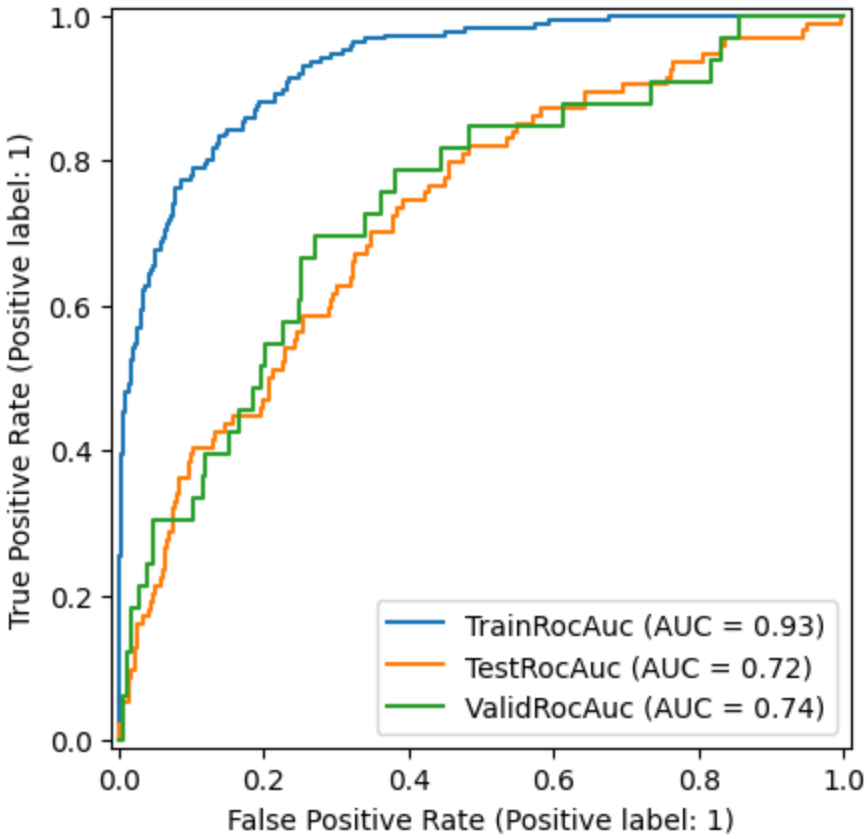
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Tı l
10	XGBoostSelectKbest	0.9268	0.9232	0.9236	0.9460	0.7247	0.7357	0.0554	0.0000	0.0000	2.6
11	Light GBMSelectKbest	0.9383	0.9208	0.9204	0.9583	0.7179	0.7523	0.3056	0.0410	0.0000	2.2
12	Logistic RegressionVariance	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6
13	XGBoostVariance	0.9268	0.9232	0.9236	0.9491	0.7280	0.7357	0.0557	0.0000	0.0000	2.6
14	Light GBMVariance	0.9285	0.9224	0.9220	0.8915	0.7281	0.7251	0.0960	0.0000	0.0000	2.5

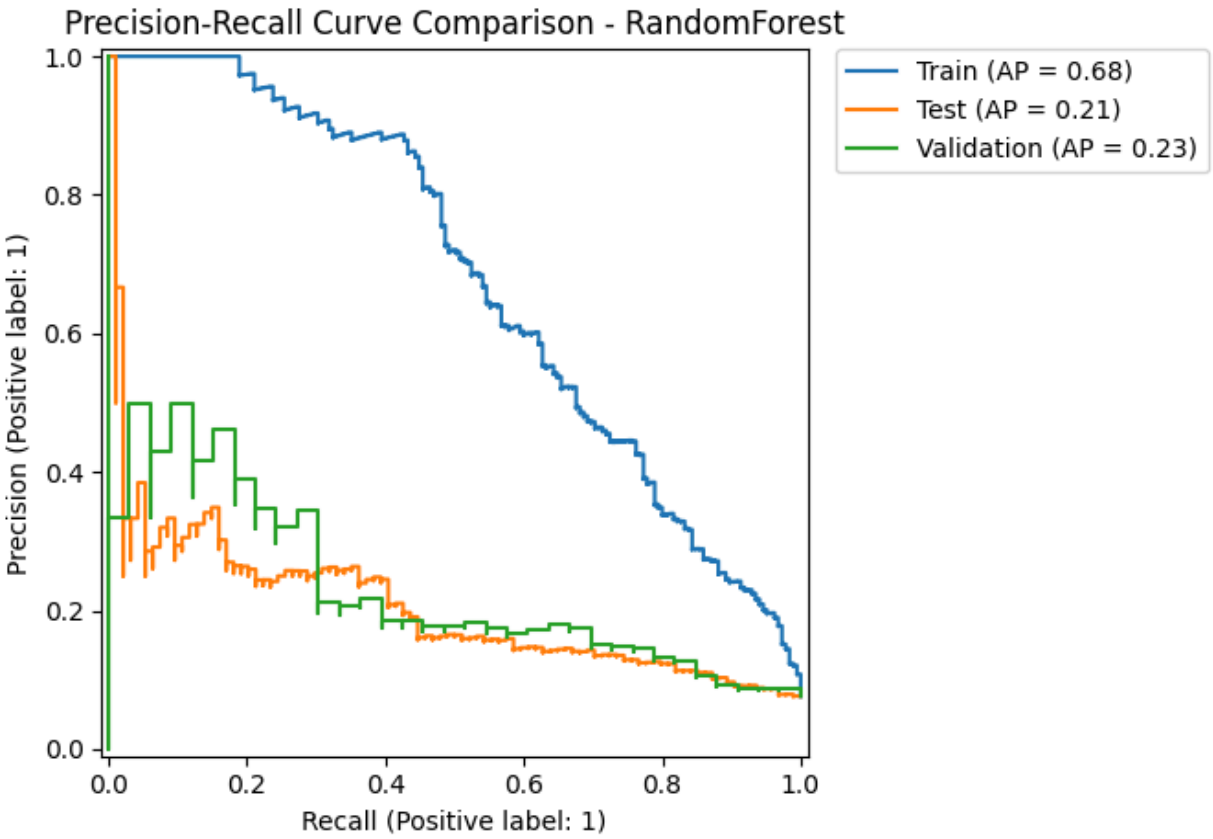
RandomForest

In [ ]:
```
ConductGridSearch(classifiers[3],cnfmatrix,fprs,tprs,precisions,recalls)
```

```
****** START RandomForest *****
Parameters:
        bootstrap: [True]
        max_depth: [5, 10]
        max_features: [15, 20]
        min_samples_leaf: [3, 5]
        min_samples_split: [5, 10]
        n_estimators: [1000]
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Cross validation with best estimator
Fit and Prediction with best estimator
```








```
Best Parameters:
        predictor__bootstrap: True
        predictor__max_depth: 5
        predictor__max_features: 20
        predictor__min_samples_leaf: 5
        predictor__min_samples_split: 5
        predictor__n_estimators: 1000
****** FINISH RandomForest  *****
```

In [ ]:

```
expLog
```

Out[ ]:

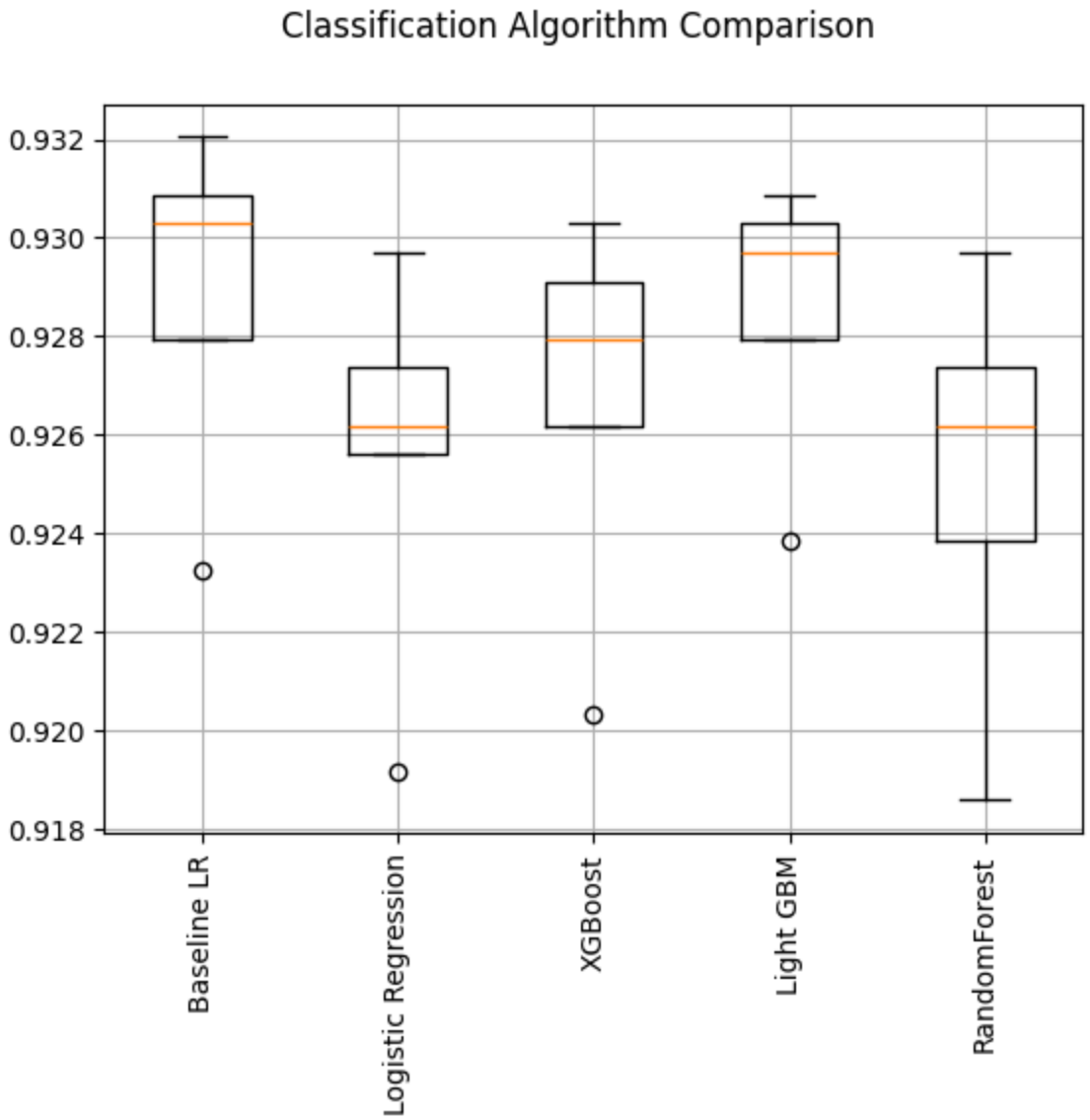
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	T l
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7
7	Baseline_132_features	0.9256	0.9219	0.9228	0.8240	0.7427	0.7549	0.0503	0.0237	0.0206	2.6
8	Logistic RegressionSelectKbest	0.9257	0.9227	0.9220	0.8144	0.7578	0.7500	0.0629	0.0396	0.0204	2.6
9	RandomForestSelectKbest	0.9252	0.9232	0.9236	0.9503	0.7373	0.7191	0.0175	0.0000	0.0000	2.6

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Tı l
10	XGBoostSelectKbest	0.9268	0.9232	0.9236	0.9460	0.7247	0.7357	0.0554	0.0000	0.0000	2.6
11	Light GBMSelectKbest	0.9383	0.9208	0.9204	0.9583	0.7179	0.7523	0.3056	0.0410	0.0000	2.2
12	Logistic RegressionVariance	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6
13	XGBoostVariance	0.9268	0.9232	0.9236	0.9491	0.7280	0.7357	0.0557	0.0000	0.0000	2.6
14	Light GBMVariance	0.9285	0.9224	0.9220	0.8915	0.7281	0.7251	0.0960	0.0000	0.0000	2.5
15	RandomForestVariance	0.9251	0.9232	0.9236	0.9511	0.7378	0.7242	0.0148	0.0000	0.0000	2.6

Model Validation

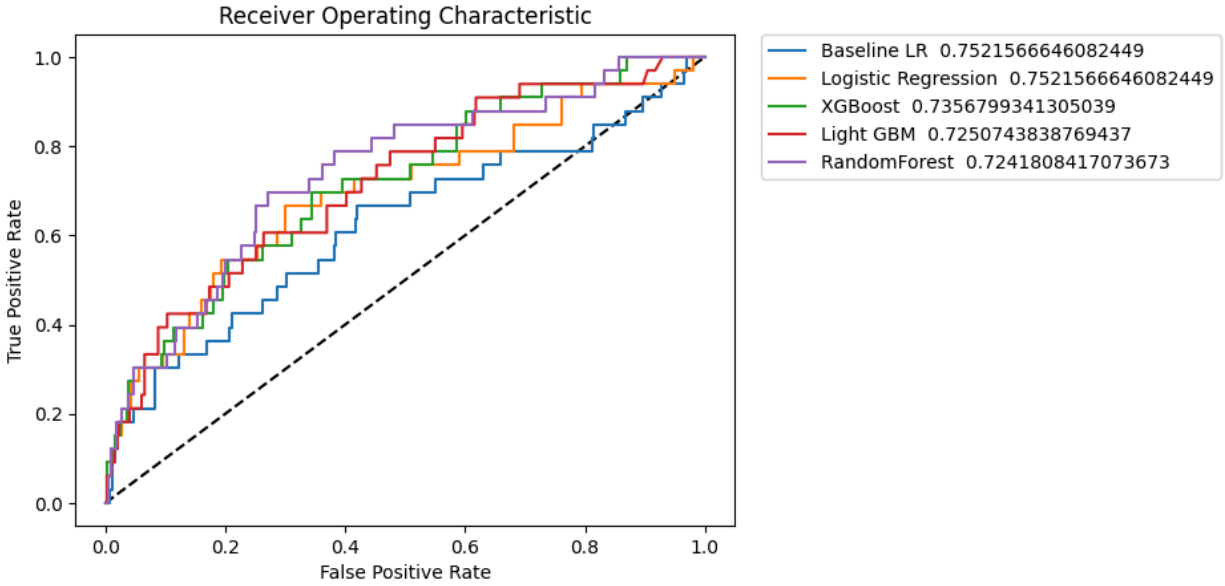
Boxplot with all CV results

In [ ]:
```
# boxplot algorithm comparison
fig = pyplot.figure()
fig.suptitle('Classification Algorithm Comparison')
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names,rotation=90)
pyplot.grid()
pyplot.show()
```



AUC (Area Under the ROC Curve)

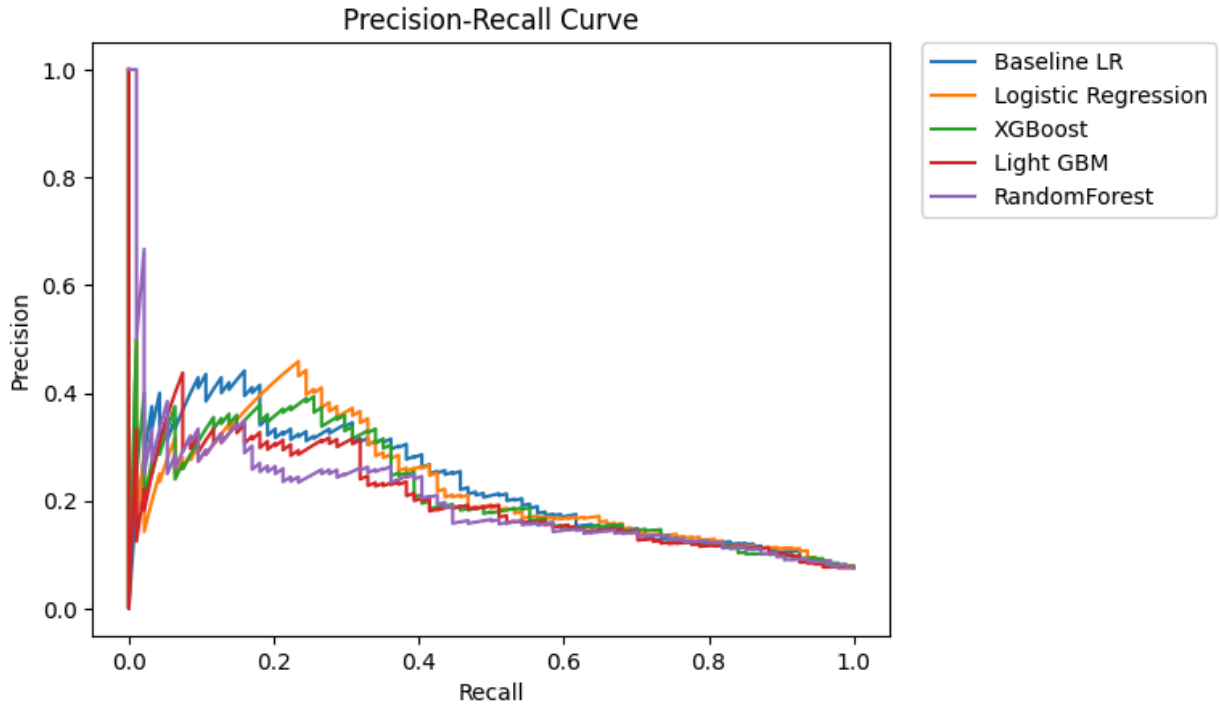
In [ ]:
```
# roc curve fpr, tpr  for all classifiers
plt.plot([0,1],[0,1], 'k--')
for i in range(len(names)):
    plt.plot(fprs[i],tprs[i],label = names[i] + '   ' + str(scores[i]))
plt.legend(bbox_to_anchor=(1.04,1), loc="upper left", borderaxespad=0)
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title('Receiver Operating Characteristic')
plt.show()
```



Precision Recall Curve

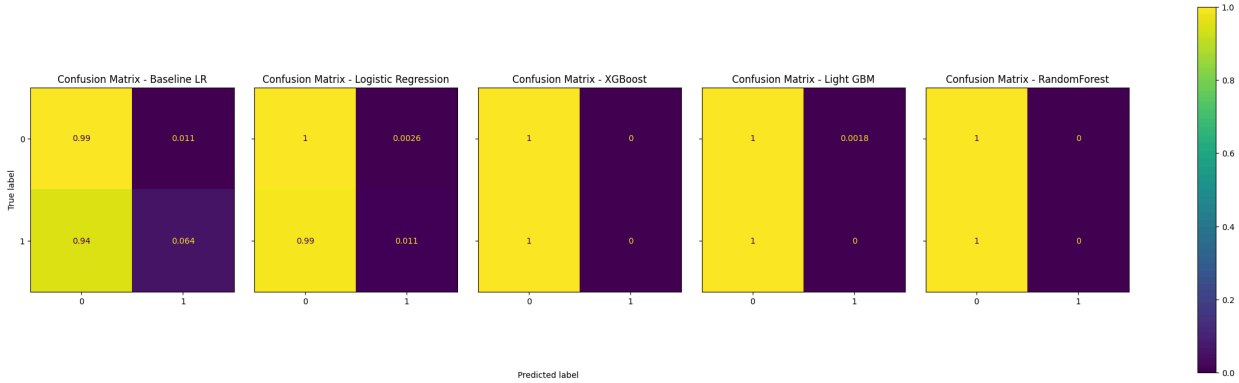
In [ ]:

```
# precision recall curve  for all classifiers
for i in range(len(names)):
    plt.plot(recalls[i],precisions[i],label = names[i])
plt.legend(bbox_to_anchor=(1.04,1), loc="upper left", borderaxespad=0)
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title('Precision-Recall Curve')
plt.show()
```



Confusion Matrix

In [ ]:
```
# plot confusion matrix for all classifiers
f, axes = plt.subplots(1, len(names), figsize=(30, 8), sharey='row')
for i in range(len(names)):
    disp = ConfusionMatrixDisplay(cnfmatrix[i], display_labels=['0', '1'])
    disp.plot(ax=axes[i], xticks_rotation=0)
    disp.ax_.set_title("Confusion Matrix - " + names[i])
    disp.im_.colorbar.remove()
    disp.ax_.set_xlabel('')
    if i!=0:
        disp.ax_.set_ylabel('')

f.text(0.4, 0.1, 'Predicted label', ha='left')
plt.subplots_adjust(wspace=0.10, hspace=0.1)

f.colorbar(disp.im_, ax=axes)
plt.show()
```



Final results

In [ ]:
```
pd.set_option('display.max_colwidth', None)
expLog.to_csv("/content/Data/expLog_Variance.csv",index=False)
expLog
```

Out[ ]:

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	T l
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7
7	Baseline_132_features	0.9256	0.9219	0.9228	0.8240	0.7427	0.7549	0.0503	0.0237	0.0206	2.6
8	Logistic RegressionSelectKbest	0.9257	0.9227	0.9220	0.8144	0.7578	0.7500	0.0629	0.0396	0.0204	2.6
9	RandomForestSelectKbest	0.9252	0.9232	0.9236	0.9503	0.7373	0.7191	0.0175	0.0000	0.0000	2.6

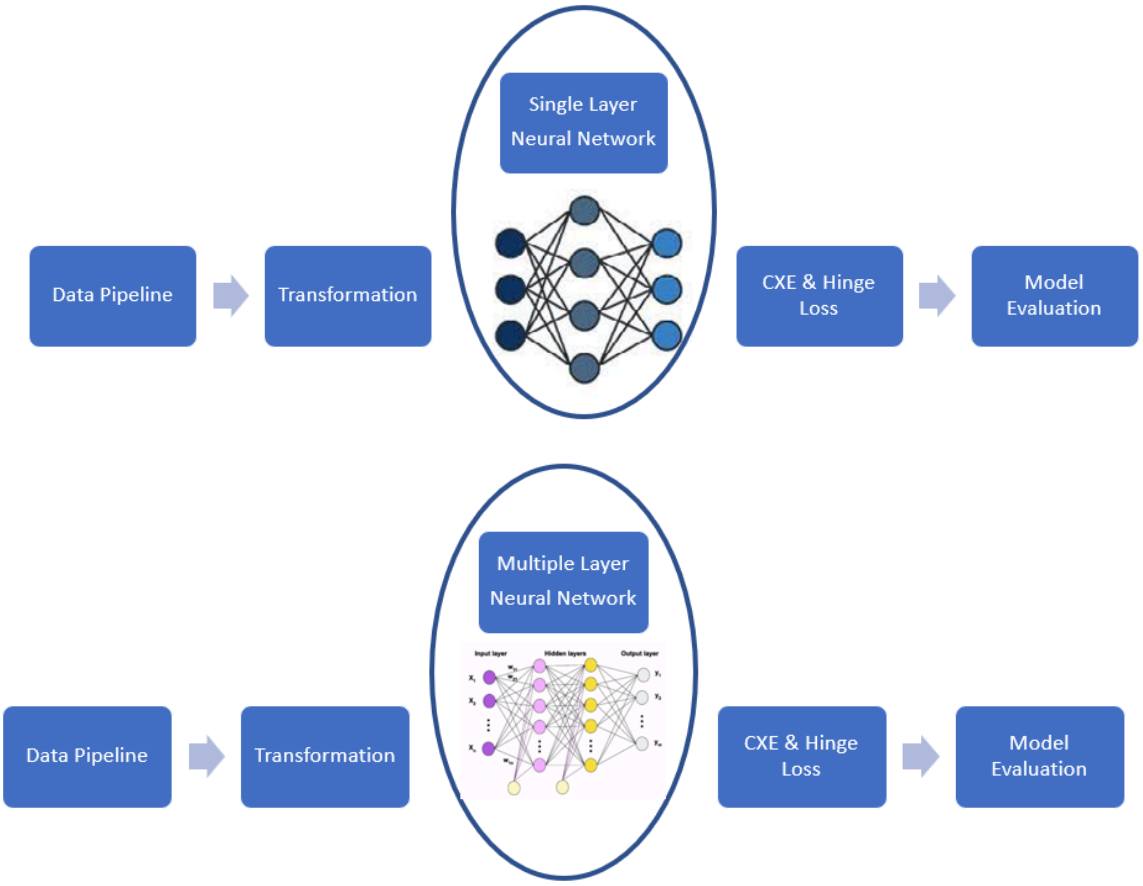
	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Tı l
10	XGBoostSelectKbest	0.9268	0.9232	0.9236	0.9460	0.7247	0.7357	0.0554	0.0000	0.0000	2.6
11	Light GBMSelectKbest	0.9383	0.9208	0.9204	0.9583	0.7179	0.7523	0.3056	0.0410	0.0000	2.2
12	Logistic RegressionVariance	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6
13	XGBoostVariance	0.9268	0.9232	0.9236	0.9491	0.7280	0.7357	0.0557	0.0000	0.0000	2.6
14	Light GBMVariance	0.9285	0.9224	0.9220	0.8915	0.7281	0.7251	0.0960	0.0000	0.0000	2.5
15	RandomForestVariance	0.9251	0.9232	0.9236	0.9511	0.7378	0.7242	0.0148	0.0000	0.0000	2.6

Deep Learning

Deep Learning Model Pipeline & Workflow

Deep learning is a subfield of machine learning focused on learning from past data using artificial neural networks with multiple hidden layers (two or more). These deep neural networks unravel complex representations of data step-by-step, layer-by-layer, into a clear and comprehensible form. An artificial neural network with one hidden layer, apart from the input and output layers, is known as a multi-layer perceptron (MLP) network.

Deep Learning Pipeline Model workflow



Imports

In [ ]:
```
!pip install tensorflow
```

```
Collecting tensorflow
  Downloading tensorflow-2.19.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl.metadata (4.1 kB)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packa
ges (from tensorflow) (1.4.0)
Collecting astunparse>=1.6.0 (from tensorflow)
  Downloading astunparse-1.6.3-py2.py3-none-any.whl.metadata (4.4 kB)
Collecting flatbuffers>=24.3.25 (from tensorflow)
  Downloading flatbuffers-25.2.10-py2.py3-none-any.whl.metadata (875 bytes)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/
python3.11/dist-packages (from tensorflow) (0.6.0)
Collecting google-pasta>=0.1.1 (from tensorflow)
  Downloading google_pasta-0.2.0-py3-none-any.whl.metadata (814 bytes)
Collecting libclang>=13.0.0 (from tensorflow)
  Downloading libclang-18.1.1-py2.py3-none-manylinux2010_x86_64.whl.metadata (5.2 kB)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-pa
ckages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages
(from tensorflow) (25.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!
=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python3.11/dist-packages (from tensorflo
w) (5.29.4)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages
(from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages
(from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-pac
kages (from tensorflow) (3.0.1)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/
dist-packages (from tensorflow) (4.13.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packag
es (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-
packages (from tensorflow) (1.71.0)
Collecting tensorboard~=2.19.0 (from tensorflow)
  Downloading tensorboard-2.19.0-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-package
s (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.2.0,>=1.26.0 in /usr/local/lib/python3.11/dist
-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-package
s (from tensorflow) (3.13.0)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in /usr/local/lib/python3.11/d
ist-packages (from tensorflow) (0.5.1)
Collecting tensorflow-io-gcs-filesystem>=0.23.1 (from tensorflow)
  Downloading tensorflow_io_gcs_filesystem-0.37.1-cp311-cp311-manylinux_2_17_x86_64.m
anylinux2014_x86_64.whl.metadata (14 kB)
Collecting wheel<1.0,>=0.23.0 (from astunparse>=1.6.0->tensorflow)
  Downloading wheel-0.45.1-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from
keras>=3.5.0->tensorflow) (14.0.0)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from
keras>=3.5.0->tensorflow) (0.0.9)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (fro
m keras>=3.5.0->tensorflow) (0.15.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/
dist-packages (from requests<3,>=2.21.0->tensorflow) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-package
```

```
s (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-p
ackages (from requests<3,>=2.21.0->tensorflow) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-p
ackages (from requests<3,>=2.21.0->tensorflow) (2025.1.31)
Requirement already satisfied: markdown>=2.6.8 in /usr/lib/python3/dist-packages (fro
m tensorboard~=2.19.0->tensorflow) (3.3.6)
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from tensorboard~=2.19.0->tensorflo
w)
  Downloading tensorboard_data_server-0.7.2-py3-none-manylinux_2_31_x86_64.whl.metada
ta (1.1 kB)
Collecting werkzeug>=1.0.1 (from tensorboard~=2.19.0->tensorflow)
  Downloading werkzeug-3.1.3-py3-none-any.whl.metadata (3.7 kB)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-pa
ckages (from werkzeug>=1.0.1->tensorboard~=2.19.0->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dis
t-packages (from rich->keras>=3.5.0->tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/d
ist-packages (from rich->keras>=3.5.0->tensorflow) (2.19.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages
(from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.2)
Downloading tensorflow-2.19.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.
whl (644.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 644.9/644.9 MB 1.6 MB/s eta 0:00:00
Downloading astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Downloading flatbuffers-25.2.10-py2.py3-none-any.whl (30 kB)
Downloading google_pasta-0.2.0-py3-none-any.whl (57 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.5/57.5 kB 5.0 MB/s eta 0:00:00
Downloading libclang-18.1.1-py2.py3-none-manylinux2010_x86_64.whl (24.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.5/24.5 MB 80.9 MB/s eta 0:00:00
Downloading tensorboard-2.19.0-py3-none-any.whl (5.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.5/5.5 MB 117.3 MB/s eta 0:00:00
Downloading tensorflow_io_gcs_filesystem-0.37.1-cp311-cp311-manylinux_2_17_x86_64.man
ylinux2014_x86_64.whl (5.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.1/5.1 MB 108.7 MB/s eta 0:00:00
Downloading tensorboard_data_server-0.7.2-py3-none-manylinux_2_31_x86_64.whl (6.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.6/6.6 MB 121.9 MB/s eta 0:00:00
Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 224.5/224.5 kB 17.4 MB/s eta 0:00:00
Downloading wheel-0.45.1-py3-none-any.whl (72 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.5/72.5 kB 5.8 MB/s eta 0:00:00
Installing collected packages: libclang, flatbuffers, wheel, werkzeug, tensorflow-io-
gcs-filesystem, tensorboard-data-server, google-pasta, tensorboard, astunparse, tenso
rflow
Successfully installed astunparse-1.6.3 flatbuffers-25.2.10 google-pasta-0.2.0 libcla
ng-18.1.1 tensorboard-2.19.0 tensorboard-data-server-0.7.2 tensorflow-2.19.0 tensorfl
ow-io-gcs-filesystem-0.37.1 werkzeug-3.1.3 wheel-0.45.1
```

In [ ]:

```
import torch
import tensorflow as tf
import torch.nn as nn
import torch.nn.functional as func
import torch.optim as optim
from torch.utils.data import DataLoader

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
```

```
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import BatchNormalization
from tensorflow.keras.layers import BatchNormalization
#from keras.layers.normalization import BatchNormalization

import copy
from datetime import datetime
import pickle
import time
import torch
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
import torch.nn as nn
import torch.nn.functional as func
import torch.optim as optim
from torch.optim import lr_scheduler

# Metrics
from sklearn.metrics import auc
```

Single layer Neural Network

Data Preparation

Transform data using data pipeline and converted into Tensor for neural network pipeline.

In [ ]:
```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cpu
```

In [ ]:
```
import torch
print(torch.__version__)
print(torch.cuda.is_available())
```

```
2.6.0+cpu
False
```

In [ ]:
```
full_X_train = data_prep_pipeline.fit_transform(X_train)
full_X_test = data_prep_pipeline.fit_transform(X_test)

#full_X_train_gpu = torch.FloatTensor(full_X_train).cuda()
full_X_train_gpu=torch.FloatTensor(full_X_train)
full_X_test_gpu = torch.FloatTensor(full_X_test)

y_train_gpu =  torch.FloatTensor(y_train.to_numpy())
y_test_gpu = torch.FloatTensor(y_test.to_numpy())
```

In [ ]:
```
full_X_test_gpu.shape,full_X_train_gpu.shape
```

Out[ ]:
```
(torch.Size([1231, 138]), torch.Size([2439, 138]))
```

In [ ]:
```
results = pd.DataFrame(columns=["ExpID",
                "Train Acc", "Val Acc", "Test Acc", "p-value",
                "Train AUC", "Val AUC", "Test AUC",
                "Train f1", "Val f1", "Test f1",
                "Train logloss", "Val logloss", "Test logloss",
                "Train Time(s)", "Val Time(s)", "Test Time(s)",
                "Experiment description",
                "Top 10 Features"])
```

One layer : Linear and Sigmoid Activate Function

Signmoid layer is used to create the probability of prediction.

In [ ]:
```
D_in = full_X_train_gpu.shape[1]
D_hidden1 = 20
D_hidden2 = 10
D_out= 1
model1 = torch.nn.Sequential(
    torch.nn.Linear(D_in, D_out),
    nn.Sigmoid())
```

In [ ]:
```
# Select device: CUDA if available, else CPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Move model to device
model1 = model1.to(device)

# Define optimizer
learning_rate = 0.01
optimizer = torch.optim.Adam(model1.parameters(), lr=learning_rate)
```

In [ ]:
```
def return_report(y, y_prob):
  _, y_pred = torch.max(y_prob, dim = 1)
  y_pred = y_pred.cpu().numpy()
  acc = accuracy_score(y, y_pred)
  roc_auc = roc_auc_score(y, y_prob.cpu().detach().numpy())

  return_list = ([round(acc,4), round(roc_auc, 4)])

  return return_list
```

In [ ]:
```
def print_report(y, y_prob):
  _, y_pred = torch.max(y_prob, dim = 1)
  y_pred = y_pred.cpu().numpy()
  acc = accuracy_score(y, y_pred)
  roc_auc = roc_auc_score(y, y_prob.cpu().detach().numpy())

  print(f'Accuracy : {round(acc,4)} ; ROC_AUC : {round(roc_auc, 4)}')
```

Train Neural Network

In [ ]:
```
epochs = 500
y_train_gpu = y_train_gpu.reshape(-1, 1)
print('Train data : ')
model1.train()
for i in range(epochs):


  y_train_pred_prob = model1(full_X_train_gpu)

  loss = func.binary_cross_entropy(y_train_pred_prob, y_train_gpu)
  optimizer.zero_grad()
  #loss = loss_func(y_train_pred_prob, y_train_gpu)
  loss.backward()
  optimizer.step()

  if i % 50 == 49:
    print(f"Epoch {i + 1}:")
    print_report(y_train, y_train_pred_prob)
```

```
Train data :
Epoch 50:
Accuracy : 0.9241 ; ROC_AUC : 0.8286
Epoch 100:
Accuracy : 0.9241 ; ROC_AUC : 0.8305
Epoch 150:
Accuracy : 0.9241 ; ROC_AUC : 0.8321
Epoch 200:
Accuracy : 0.9241 ; ROC_AUC : 0.8335
Epoch 250:
Accuracy : 0.9241 ; ROC_AUC : 0.8345
Epoch 300:
Accuracy : 0.9241 ; ROC_AUC : 0.8356
Epoch 350:
Accuracy : 0.9241 ; ROC_AUC : 0.8363
Epoch 400:
Accuracy : 0.9241 ; ROC_AUC : 0.8369
Epoch 450:
Accuracy : 0.9241 ; ROC_AUC : 0.8374
Epoch 500:
Accuracy : 0.9241 ; ROC_AUC : 0.8379
```

Evaluation of Neural Network model

In [ ]:
```
model1.eval()
y_test_gpu = y_test_gpu.reshape(-1, 1)
with torch.no_grad():
    y_test_pred_prob=model1(full_X_test_gpu)
    print('-' * 50)
    print('Test data : ')
    print_report(y_test, y_test_pred_prob)
    print('-' * 50)
```

```
--------------------------------------------------
Test data :
Accuracy : 0.9236 ; ROC_AUC : 0.7463
--------------------------------------------------
```

Multi Layer NN model with custom Hinge & CXE Loss

Model Definition

The model contain one linear layer, one hidden layer with Relu function and sigmoid function for probability prediction.

In [ ]:
```
## Model using hidden layers
class SVMNNmodel(nn.Module):
  def __init__(self, input_features , hidden1 = 80, hidden2 = 80, output_features = 1
    super(SVMNNmodel, self).__init__()
    # self.f_connected1 = nn.Linear(input_features, hidden1)
    # self.f_connected2 = nn.Linear(hidden1, hidden2)
    # self.out = nn.Linear(hidden2, output_features)
    # self.sigmoid = nn.Sigmoid()
    self.f_connected1 = nn.Linear(input_features, hidden1)
    self.out = nn.Linear(hidden1, output_features)

  def forward(self, x):
    #x = func.relu(self.f_connected1(x))
    #x= func.relu(self.f_connected2(x))
    #x = self.out(x)
    #return self.sigmoid(x)
    h_relu = torch.relu(self.f_connected1(x))
    y_target_pred = torch.sigmoid(self.out(h_relu))
    return y_target_pred
```

Custom Hinge loss Definition

The model contain one linear layer, one hidden layer with Relu function and sigmoid function for probability prediction.

To extend a hard SVM to cases in which the data are not linearly separable (a little noisy), we introduce the hinge loss function,

$$\max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)\right).$$

This function is zero if the following constraint for a training example $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1$, is satisfied, in other words, if $\vec{x}_i$ lies on the correct side of the margin (DMZ demilitarized zone). For data on the wrong side of the margin, the function's value is proportional to the distance from the margin.

This type of data loss leads a soft-margin SVM classifier. Computing a (soft-margin) SVM classifier amounts to minimizing an expression of the form:

$$LinSVM(\mathbf{w}, b) = \underset{W,b}{\operatorname{argmin}} \left(\overbrace{\frac{\lambda}{2} \underbrace{\mathbf{w}^T \cdot \mathbf{w}}_{B}}^{A} \quad + \quad \overbrace{\frac{1}{m} \sum_{i=1}^{m} max\left(0, 1 - \underbrace{y^{(i)}(\mathbf{w}^T \cdot \mathbf{x}^{(i)} + b)}_{D}\right)}^{C} \right)$$

where the parameter λ (corresponding to the A-zone in the above formulation) determines the tradeoff between increasing the margin-size and ensuring that the $\vec{x}_i$ lie on the correct side of the margin (corresponding to the C-zone in the above formulation).

Here choosing a sufficiently small value for λ yields the hard-margin classifier for linearly classifiable input data. Classically, this problem can be solved via quadratic programming (see slides/textbook/wikipedia for details). More recently approaches such as sub-gradient descent and coordinate descent have been proposed and lead to more scaleable implementations without compromising quality.

In [ ]:
```
class SVMLoss(nn.Module):
  def __init__(self):
    super(SVMLoss,self).__init__()
  def forward(self,outputs,labels,model2):
     C = 0.10
    # data_loss = torch.mean(torch.clamp(1 - outputs.squeeze().t() == labels,min=0))
     data_loss = torch.mean(torch.clamp(1 - outputs.squeeze(),min=0))
     weight = model2.out.weight.squeeze()
     reg_loss = weight.t() @ weight
     reg_loss = reg_loss + ( model2.out.bias.squeeze()**2)
     hinge = data_loss +( C*reg_loss/2)
     return (hinge)
```

In [ ]:
```
class Converttensor(Dataset):
    def __init__(self, feature, label, mode ='train', transforms=None):
        """
        Initialize data set as a list of IDs corresponding to each item of data set

        :param feature: x - numpy array
        :param label: y - numpy array
        """

        self.x = feature
        self.y = label


    def __len__(self):
        """
        Return the length of data set using list of IDs

        :return: number of samples in data set
        """
        return (self.x.shape[0])

    def __getitem__(self, index):
        """
        Generate one item of data set.

        :param index: index of item in IDs list

        :return: image and label and bouding box params
        """
        x = self.x[index,:]
        y_target = self.y[index]
```

```
        x = torch.FloatTensor(x)
        y_target_arr = np.array(y_target)
        return x, y_target_arr
```

In [ ]:

```
fprs_net_train, tprs_net_train, fprs_net_valid, tprs_net_valid = [], [], [], []
roc_auc_net_train = 0.0
roc_auc_net_valid = 0.0
num_epochs=25
batch_size=256
CASE_NAME = "NN"
```

Data Preparation

In [ ]:

```
splits = 1

# Train Test split percentage
subsample_rate = 0.3

finaldf = np.array_split(train_dataset, splits)
X_train = finaldf[0][selected_features]
y_train = finaldf[0]['TARGET']
#final_X_kaggle_test = kaggle_test
## split part of data
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, stratify=y_trai
                                                    test_size=subsample_rate, random_

X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train,stratify=y_tra

nn_X_train = data_prep_pipeline.fit_transform(X_train)
nn_X_valid = data_prep_pipeline.fit_transform(X_valid)
nn_X_test = data_prep_pipeline.fit_transform(X_test)
#nn_X_kaggle_test = data_prep_pipeline.fit_transform(final_X_kaggle_test)
#full_X_kaggle_gpu = torch.FloatTensor(nn_X_kaggle_test).cuda()
nn_y_train = np.array(y_train)
nn_y_valid = np.array(y_valid)

in_feature_cnt = nn_X_train.shape[1]
out_feature_cnt = 1

print(f"X train           shape: {nn_X_train.shape}")
print(f"X validation      shape: {nn_X_valid.shape}")
print(f"X test            shape: {nn_X_test.shape}")
#print(f"X kaggle_test     shape: {nn_X_kaggle_test.shape}")
print("Feature count           : ",in_feature_cnt)
```

```
X train           shape: (182968, 138)
X validation      shape: (32289, 138)
X test            shape: (92254, 138)
Feature count           :  138
```

In [ ]:

```
nn_dataset = {'train': nn_X_train, 'val': nn_X_valid}
dataset_sizes = {x_type : len(nn_dataset[x_type]) for x_type in ['train','val']}
```

In [ ]:

```
dataset_sizes
```

Out[ ]:
```
{'train': 182968, 'val': 32289}
```

In [ ]:
```
## Transform dataset
nn_dataset['train'] = Converttensor(nn_dataset['train'] ,nn_y_train, mode='train')
```

In [ ]:
```
## Transform validation dataset
nn_dataset['val'] = Converttensor(nn_dataset['val'] ,nn_y_valid, mode='validation')
```

In [ ]:
```
nn_dataset
```

Out[ ]:
```
{'train': <__main__.Converttensor at 0x7acbc173de50>,
 'val': <__main__.Converttensor at 0x7acbb4e6ce10>}
```

In [ ]:
```
## Set dataloader
dataloaders = {x_type: torch.utils.data.DataLoader(nn_dataset[x_type], batch_size=bat
              for x_type in ['train', 'val']}
```

Train Model

In [ ]:
```
# Select device dynamically
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Create and move model to device
nn_model = SVMNNmodel(input_features=in_feature_cnt, output_features=1).to(device)
```

In [ ]:
```
#del convergence
try:
       convergence
       epoch_offset=convergence.epoch.iloc[-1]+1
except NameError:
        convergence=pd.DataFrame(columns=['epoch','phase','roc_auc','accuracy','CXE',
        epoch_offset=0
```

In [ ]:
```
# Code adapted from https://pytorch.org/tutorials/beginner/transfer_learning_tutorial
def train(optimizer_cxe,optimizer_hinge,criteron,scheduler_cxe,scheduler_hinge,num_ep

    global roc_auc_train
    global roc_auc_valid

    fac_cel=torch.tensor(w_cel)

    start = time.time()

    best_model_wts = copy.deepcopy(nn_model.state_dict())
    best_acc = 0.0

    # Store results to easier collect stats
    nn_y_pred = {x: np.zeros((dataset_sizes[x],1)) for x in ['train', 'val']}

    for epoch in range(num_epochs):
```

```
        # Each epoch has a training and validation phase
        for phase in ['train', 'val']:
            t0=time.time()
            # Reset to zero to be save

            nn_y_pred[phase].fill(0)
            if phase == 'train':
                nn_model.train()  # Set model to training mode
            else:
                nn_model.eval()   # Set model to evaluate mode

            running_loss = 0.0
            running_corrects = 0
            running_hinge = 0.0
            running_cxe = 0.0

            # Iterate over data.
            ix=0
            for inputs, targets in dataloaders[phase]:
                n_batch = len(targets)

                #nn_y_pred[phase][ix:ix+n_batch,:] = targets.detach().numpy().reshape

                inputs = inputs.to(device)
                targets = targets.to(device).float()

                # zero the parameter gradients
                optimizer_hinge.zero_grad()
                optimizer_cxe.zero_grad()

                # forward
                # track history if only in train
                with torch.set_grad_enabled(phase == 'train'):
                    output_target = nn_model.forward(inputs)
                    preds = torch.where((output_target > .5), 1, 0)
                    #print(output_target.squeeze(),targets)
                    ix += n_batch
                    loss_cxe = func.binary_cross_entropy(output_target.squeeze(), tar
                    loss_hinge = criteron.forward(output_target.squeeze(), targets,nn

                    # backward + optimize only if in training phase
                    if phase == 'train':
                        loss_hinge.backward()
                        optimizer_hinge.step()
                        #loss_cxe.backward()
                        optimizer_cxe.step()

                # statistics
                running_hinge += loss_hinge.item() * inputs.size(0)
                running_corrects += 1*(preds == targets.data.int()).sum().item()
                running_cxe += loss_cxe.item() * inputs.size(0)

            if phase == 'train':
                scheduler_hinge.step()
                scheduler_cxe.step()

            epoch_cxe = running_cxe / dataset_sizes[phase]
            epoch_hinge = running_hinge / dataset_sizes[phase]
            epoch_acc = running_corrects / dataset_sizes[phase]
```

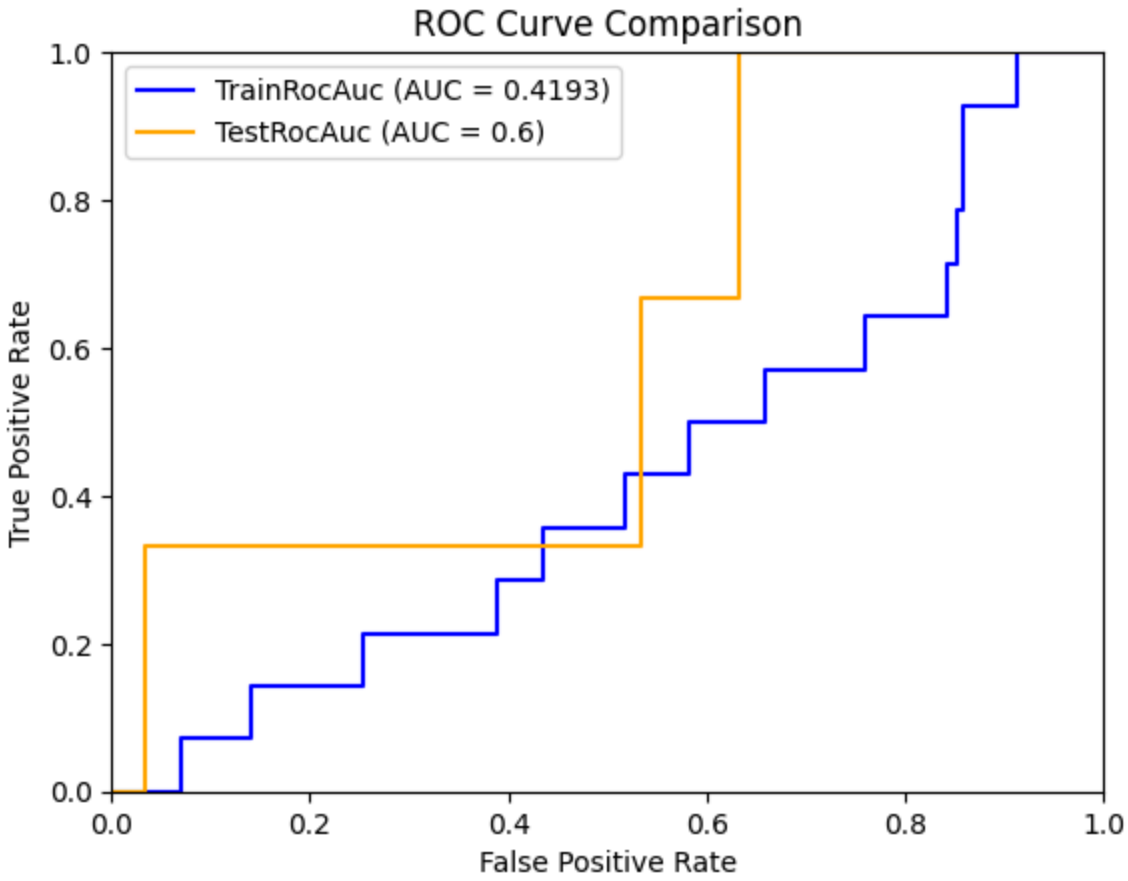
```
            epoch_roc_auc = 0.0
            if (phase == 'train'):
                ## Calculate 'false_positive_rate' and 'True_positive_rate' of train

                nn_fprs_train, nn_tpr_train, nn_thresholds = roc_curve(targets.detach
                fprs_net_train.append(nn_fprs_train)
                tprs_net_train.append(nn_tpr_train)
                roc_auc_train = round(auc(nn_fprs_train, nn_tpr_train), 4)
                epoch_roc_auc = roc_auc_train

            elif (phase == 'val'):
                ## Calculate 'false_positive_rate' and 'True_positive_rate' of valid
                nn_fpr_valid, nn_tpr_valid, thresholds = roc_curve(targets.detach().c
                fprs_net_valid.append(nn_fpr_valid)
                tprs_net_valid.append(nn_tpr_valid)
                roc_auc_valid = round(auc(nn_fpr_valid, nn_tpr_valid), 4)
                epoch_roc_auc = roc_auc_valid

            dt=time.time() - t0
            fmt='{:6s} ROC_AUC: {:.4f} Acc: {:.4f} CXE: {:.4f} Hinge: {:.4f}  DT={:.1
            out_list=[phase, epoch_roc_auc, epoch_acc, epoch_cxe, epoch_hinge] + [dt]
            out_str=fmt.format(*out_list)
            if phase=='train':
                epoch_str='Epoch {}/{} '.format(epoch, num_epochs)
                out_str=epoch_str + out_str
            else:
                out_str = ' '*len(epoch_str) + out_str
            print(out_str)

            if (phase == 'val') and epoch == num_epochs-1:
                 plt.plot(nn_fprs_train, nn_tpr_train, color='blue')
                 plt.plot(nn_fpr_valid, nn_tpr_valid, color='orange')
                 plt.xlim([0.0,1.0])
                 plt.ylim([0.0,1.0])
                 plt.xlabel('False Positive Rate')
                 plt.ylabel('True Positive Rate')
                 plt.title(f'ROC Curve Comparison')
                 plt.legend([f'TrainRocAuc (AUC = {roc_auc_train})', f'TestRocAuc (AU
                 plt.show()

            convergence.loc[len(convergence)] = [epoch+epoch_offset,phase,
                        epoch_roc_auc, epoch_acc, epoch_cxe, epoch_hinge]

            # deep copy the model
            if phase == 'val' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(nn_model.state_dict())

    time_elapsed = time.time() - start
    print('Training complete in {:.0f}m {:.0f}s'.format(
        time_elapsed // 60, time_elapsed % 60))
    print('Best val Acc: {:4f}'.format(best_acc))

    # load best model weights
    nn_model.load_state_dict(best_model_wts)
```

Execute Model

In [ ]:

```
import torch
from torch import optim
from torch.optim import lr_scheduler
from datetime import datetime
import pickle
import time

# Set device (GPU if available, else CPU)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Move model to device
nn_model = SVMNNmodel(input_features=in_feature_cnt, output_features=1).to(device)

# Optimizers
optimizer_cxe = optim.Adam(nn_model.parameters(), lr=0.0001)
optimizer_hinge = torch.optim.SGD(nn_model.parameters(), lr=learning_rate, momentum=0

# Learning rate schedulers
scheduler_cxe = lr_scheduler.StepLR(optimizer_cxe, step_size=10, gamma=0.1)
scheduler_hinge = lr_scheduler.StepLR(optimizer_hinge, step_size=10, gamma=0.1)

# Loss function
criteron = SVMLoss()

# Train the model (ensure your training loop uses `device`)
train(optimizer_cxe, optimizer_hinge, criteron, scheduler_cxe, scheduler_hinge, num_e

# Save model (no need to move it back to CPU for pickling, but you can if desired)
t0 = time.time()
date_time = datetime.now().strftime("--%Y-%m-%d-%H-%M-%S-%f")
pickle.dump(nn_model, open(DATA_DIR + '/' + CASE_NAME + date_time + '.p', 'wb'))
print('Pickled in {:.2f} sec'.format(time.time() - t0))
```

```
Epoch 0/25 train  ROC_AUC: 0.5854 Acc: 21.6965 CXE: 2.3631 Hinge: 0.1452  DT=124.1
           val    ROC_AUC: 0.5968 Acc: 20.6555 CXE: 2.8562 Hinge: 0.0959  DT=14.8
Epoch 1/25 train  ROC_AUC: 0.6500 Acc: 20.6545 CXE: 2.9947 Hinge: 0.0880  DT=116.5
           val    ROC_AUC: 0.8226 Acc: 20.6555 CXE: 3.0808 Hinge: 0.0830  DT=14.0
Epoch 2/25 train  ROC_AUC: 0.4659 Acc: 20.6576 CXE: 3.1236 Hinge: 0.0812  DT=115.4
           val    ROC_AUC: 0.4310 Acc: 20.6417 CXE: 3.1565 Hinge: 0.0798  DT=13.6
Epoch 3/25 train  ROC_AUC: 0.5122 Acc: 20.6576 CXE: 3.1751 Hinge: 0.0792  DT=114.1
           val    ROC_AUC: 0.6562 Acc: 20.6624 CXE: 3.1977 Hinge: 0.0783  DT=14.5
Epoch 4/25 train  ROC_AUC: 0.6100 Acc: 20.6564 CXE: 3.2206 Hinge: 0.0775  DT=113.3
           val    ROC_AUC: 0.9194 Acc: 20.6555 CXE: 3.2451 Hinge: 0.0765  DT=14.3
Epoch 5/25 train  ROC_AUC: 0.4576 Acc: 20.6627 CXE: 3.2652 Hinge: 0.0758  DT=114.1
           val    ROC_AUC: 0.6605 Acc: 20.6279 CXE: 3.2829 Hinge: 0.0749  DT=15.8
Epoch 6/25 train  ROC_AUC: 0.6797 Acc: 20.6592 CXE: 3.2979 Hinge: 0.0744  DT=115.4
           val    ROC_AUC: 0.4375 Acc: 20.6624 CXE: 3.3046 Hinge: 0.0738  DT=14.4
Epoch 7/25 train  ROC_AUC: 0.6406 Acc: 20.6596 CXE: 3.3214 Hinge: 0.0734  DT=115.2
           val    ROC_AUC: 0.4444 Acc: 20.6486 CXE: 3.3376 Hinge: 0.0728  DT=13.8
Epoch 8/25 train  ROC_AUC: 0.5122 Acc: 20.6600 CXE: 3.3427 Hinge: 0.0726  DT=115.1
           val    ROC_AUC: 0.3222 Acc: 20.6486 CXE: 3.3558 Hinge: 0.0720  DT=14.1
Epoch 9/25 train  ROC_AUC: 0.4205 Acc: 20.6607 CXE: 3.3583 Hinge: 0.0719  DT=114.2
           val    ROC_AUC: 0.5161 Acc: 20.6555 CXE: 3.3614 Hinge: 0.0716  DT=14.3
Epoch 10/25 train  ROC_AUC: 0.5041 Acc: 20.6588 CXE: 3.3657 Hinge: 0.0716  DT=113.2
            val    ROC_AUC: 0.9375 Acc: 20.6624 CXE: 3.3662 Hinge: 0.0715  DT=15.2
Epoch 11/25 train  ROC_AUC: 0.5079 Acc: 20.6604 CXE: 3.3705 Hinge: 0.0715  DT=114.2
            val    ROC_AUC: 0.8065 Acc: 20.6555 CXE: 3.3751 Hinge: 0.0713  DT=14.0
Epoch 12/25 train  ROC_AUC: 0.4996 Acc: 20.6572 CXE: 3.3762 Hinge: 0.0713  DT=113.3
            val    ROC_AUC: 0.4138 Acc: 20.6417 CXE: 3.3790 Hinge: 0.0712  DT=14.2
Epoch 13/25 train  ROC_AUC: 0.5527 Acc: 20.6596 CXE: 3.3804 Hinge: 0.0712  DT=120.0
            val    ROC_AUC: 0.4222 Acc: 20.6486 CXE: 3.3824 Hinge: 0.0710  DT=14.8
Epoch 14/25 train  ROC_AUC: 0.5808 Acc: 20.6588 CXE: 3.3838 Hinge: 0.0710  DT=114.4
            val    ROC_AUC: 0.7333 Acc: 20.6486 CXE: 3.3862 Hinge: 0.0709  DT=14.6
Epoch 15/25 train  ROC_AUC: 0.5557 Acc: 20.6615 CXE: 3.3869 Hinge: 0.0709  DT=113.5
            val    ROC_AUC: 0.3438 Acc: 20.6624 CXE: 3.3892 Hinge: 0.0708  DT=14.5
Epoch 16/25 train  ROC_AUC: 0.5493 Acc: 20.6564 CXE: 3.3898 Hinge: 0.0708  DT=114.2
            val    ROC_AUC: 0.4000 Acc: 20.6348 CXE: 3.3917 Hinge: 0.0707  DT=13.6
Epoch 17/25 train  ROC_AUC: 0.3741 Acc: 20.6611 CXE: 3.3922 Hinge: 0.0707  DT=114.4
            val    ROC_AUC: 0.0625 Acc: 20.6624 CXE: 3.3937 Hinge: 0.0706  DT=14.4
Epoch 18/25 train  ROC_AUC: 0.6383 Acc: 20.6580 CXE: 3.3943 Hinge: 0.0706  DT=115.9
            val    ROC_AUC: 0.8556 Acc: 20.6486 CXE: 3.3952 Hinge: 0.0705  DT=13.5
Epoch 19/25 train  ROC_AUC: 0.3725 Acc: 20.6584 CXE: 3.3956 Hinge: 0.0705  DT=121.0
            val    ROC_AUC: 0.5123 Acc: 20.6279 CXE: 3.3976 Hinge: 0.0704  DT=13.9
Epoch 20/25 train  ROC_AUC: 0.5199 Acc: 20.6607 CXE: 3.3974 Hinge: 0.0705  DT=113.6
            val    ROC_AUC: 0.0938 Acc: 20.6624 CXE: 3.3977 Hinge: 0.0704  DT=13.7
Epoch 21/25 train  ROC_AUC: 0.6036 Acc: 20.6596 CXE: 3.3975 Hinge: 0.0705  DT=113.0
            val    ROC_AUC: 0.3710 Acc: 20.6555 CXE: 3.3979 Hinge: 0.0704  DT=14.1
Epoch 22/25 train  ROC_AUC: 0.5087 Acc: 20.6611 CXE: 3.3978 Hinge: 0.0704  DT=111.2
            val    ROC_AUC: 0.6889 Acc: 20.6486 CXE: 3.3980 Hinge: 0.0704  DT=13.7
Epoch 23/25 train  ROC_AUC: 0.5595 Acc: 20.6592 CXE: 3.3979 Hinge: 0.0704  DT=110.8
            val    ROC_AUC: 0.6889 Acc: 20.6486 CXE: 3.3982 Hinge: 0.0704  DT=14.2
Epoch 24/25 train  ROC_AUC: 0.4193 Acc: 20.6600 CXE: 3.3980 Hinge: 0.0704  DT=114.7
            val    ROC_AUC: 0.6000 Acc: 20.6486 CXE: 3.3984 Hinge: 0.0704  DT=14.3
```




```
Training complete in 53m 53s
Best val Acc: 20.662424
Pickled in 0.00 sec
```

In [ ]:
```
convergence.head(5)
```

Out[ ]:

	epoch	phase	roc_auc	accuracy	CXE	Hinge
0	0	train	0.5854	21.696526	2.363109	0.145194
1	0	val	0.5968	20.655517	2.856225	0.095926
2	1	train	0.6500	20.654453	2.994682	0.088027
3	1	val	0.8226	20.655517	3.080824	0.083029
4	2	train	0.4659	20.657601	3.123584	0.081188

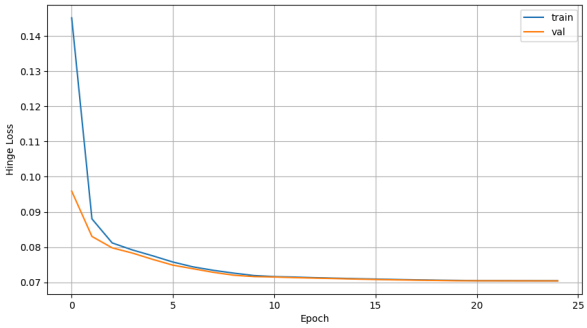
Plot Convergence

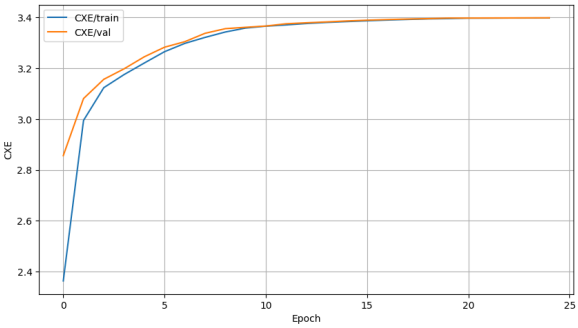
In [ ]:
```
from scipy.stats import iqr

def plot_convergence(figsize=(22,12)):
  conv = {phase : convergence[convergence.phase==phase]
          for phase in ['train','val']}

  fig,axes = plt.subplots(2,2,figsize=figsize)
  cols = {'train' : 'tab:blue', 'val' : 'tab:orange'}
```

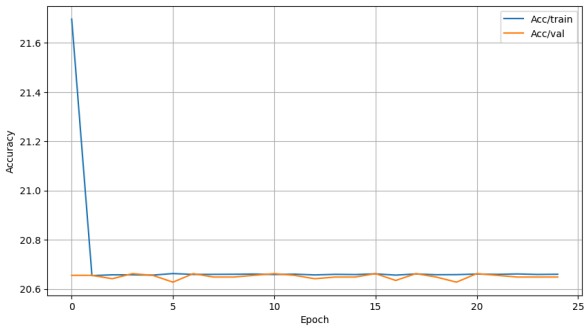
```
  # Loss
  ax=axes[0,0]
  for phase in ['train','val']:
    ax.plot(conv[phase].epoch,conv[phase].Hinge,label=phase,c=cols[phase])
  ax.set_xlabel('Epoch')
  ax.set_ylabel('Hinge Loss')
  ax.legend()
  ax.grid()

  # CXE
  ax=axes[0,1]
  for phase in ['train','val']:
    ax.plot(conv[phase].epoch,conv[phase].CXE,label='CXE/'+phase,c=cols[phase])
  ax.set_xlabel('Epoch')
  ax.set_ylabel('CXE')
  ax.legend()
  ax.grid()

  # Accuracy
  ax=axes[1,0]
  for phase in ['train','val']:
    ax.plot(conv[phase].epoch,conv[phase].accuracy,label='Acc/'+phase,c=cols[phase])
  ax.set_xlabel('Epoch')
  ax.set_ylabel('Accuracy')
  ax.legend()
  ax.grid()

  # Plot ROC_AUC Curve of train and test
  ax=axes[1,1]
  for i in range(num_epochs):
      plt.plot(fprs_net_train[i],tprs_net_train[i], color='blue')
      plt.plot(fprs_net_valid[i],tprs_net_valid[i], color='orange')
  ax.set_xlim([0.0,1.0])
  ax.set_ylim([0.0,1.0])
  ax.set_xlabel('False Positive Rate')
  ax.set_ylabel('True Positive Rate')
  ax.set_title(f'ROC Curve Comparison')
  ax.legend([f'TrainRocAuc (AUC = {roc_auc_train})', f'TestRocAuc (AUC = {roc_auc_val
  ax.grid()
```

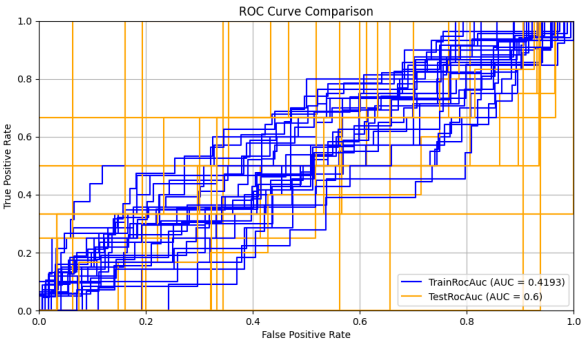
In [ ]:

```
plot_convergence(figsize=(22,12))
```









Write Up

Abstract

This project aims to predict loan defaults using the Home Credit Default Risk dataset. We employed machine learning and deep learning models, including Logistic Regression, Gradient Boosting, XGBoost, LightGBM, and SVM. Building on exploratory data analysis (EDA) and hyper-tuned models, we modified feature engineering to handle data leakage, employing an improved data processing flow. Our machine learning pipelines were measured using the following metrics: Confusion Matrix, Accuracy Score, Precision, Recall, F1 Score, AUC (Area Under ROC Curve), CXE Loss, and Hinge Loss (Deep Learning). In the final phase, we demonstrated that tuned machine learning techniques can outperform baseline models, aiding Home Credit in evaluating loan applications. XGBoost achieved the best overall balance between accuracy and ROC AUC, delivering a Training Accuracy of 94.91%, Testing Accuracy of 92.36%, and ROC AUC of 74.70%. While Logistic Regression slightly exceeded in ROC AUC (75.22%), it lacked the advanced learning capacity of XGBoost. Additionally, we explored the impact of feature selection and data preprocessing techniques on model performance, ensuring robustness and reliability in our predictions.

Introduction

Home Credit is an international non-bank financial institution that specializes in providing loans to individuals regardless of their credit history. The organization aims to offer a positive borrowing experience to customers who do not rely on traditional banking sources. To address the issue of unfair loan rejection, Home Credit Group has published a dataset on Kaggle.

https://www.kaggle.com/competitions/home-credit-default-risk/overview

The goal of this project is to develop a machine learning model that predicts customer behavior regarding loan repayment. Our task involves creating a pipeline to build a baseline machine learning model using logistic regression classification algorithms. The final model will be evaluated using various performance metrics to enhance its accuracy and reliability. Businesses can utilize the model's output to identify loans at risk of default. The new model will ensure that clients capable of repayment are not rejected and that loans are provided with appropriate terms, empowering clients to succeed.

The results of our machine learning pipelines will be measured using the follwing metrics;

- Confusion Matrix
  - True Positives (TP): Correctly predicted positive cases.
  - True Negatives (TN): Correctly predicted negative cases.
  - False Positives (FP): Incorrectly predicted positive cases.
  - False Negatives (FN): Incorrectly predicted negative cases.

- Accuracy Score

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

- Precision

$$\text{Precision} = \frac{TP}{TP+FP}$$

- Recall

$$\text{Recall} = \frac{TP}{TP+FN}$$

- F1 score

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- AUC (Area Under ROC Curve)

- CXE Loss

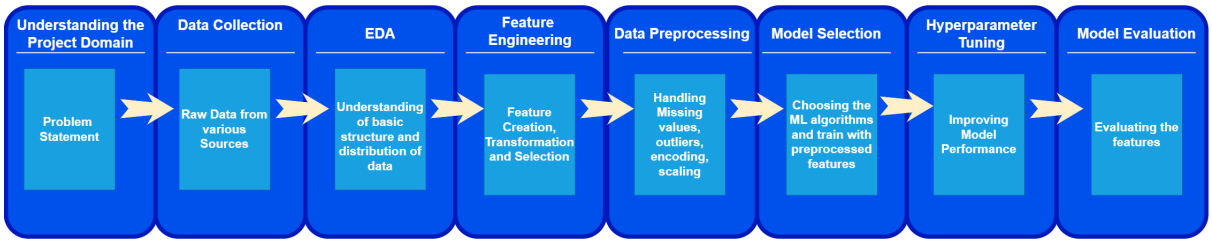
$$\text{Cross-Entropy Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$$

- Hinge Loss (Deep Learning)

$$\text{Hinge Loss} = \frac{1}{N}\sum_{i=1}^{N}\max(0, 1 - y_i \cdot f(x_i))$$

Workflow

For this project, we are following the proposed workflow as mentioned in Phase-0 of this project.



Dataset Description

Overview The full dataset consists of 7 tables. There is 1 primary table and 6 secondary tables.

Primary Tables application_train This Primary table includes the application information for each loan application at Home Credit in one row. This row includes the target variable of whether or not the loan was repaid. We use this field as the basis to determine the feature importance. The target variable is binary in nature based since this is a classification problem. '1' - client with payment difficulties: he/she had late payment more than N days on at least one of the first M installments of the loan in our sample '0' - all other cases The number of variables are 122. The number of data entries are 307,511 application_test This table includes the application information for each loan application at Home Credit in one row. The features are the same as the train data but exclude the target variable The number of variables are 121. The number of data entries are 48,744. Secondary Tables Bureau This table includes all previous credits received by a customer from other financial institutions prior to their loan application. There is one row for each previous credit, meaning a many-to-one relationship with the primary table. We could join it with primary table by using current application ID, SK_ID_CURR. The number of variables are 17.The number of data entries are 1,716,428.

Bureau Balance This table includes the monthly balance for a previous credit at other financial institutions. There is one row for each monthly balance, meaning a many-to-one relationship with the Bureau table. We could join it with bureau table by using bureau's ID, SK_ID_BUREAU. The number of variables are 3. The number of data entries are 27,299,925

Previous Application This table includes previous applications for loans made by the customer at Home Credit. There is one row for each previous application, meaning a many-to-one relationship with the primary table. We could join it with primary table by using current application ID, SK_ID_CURR. There are four types of contracts: a. Consumer loan(POS – Credit limit given to buy consumer goods) b. Cash loan(Client is given cash) c. Revolving loan(Credit) d. XNA (Contract type without values) The number of variables are 37. The number of data entries are 1,670,214

POS CASH Balance This table includes a monthly balance snapshot of a previous point of sale or cash loan that the customer has at Home Credit. There is one row for each monthly balance, meaning a many-to-one relationship with the Previous Application table. We would join it with Previous Application table by using previous application ID, SK_ID_PREV, then join it with primary

table by using current application ID, SK_ID_CURR. The number of variables are 8. The number of data entries are 10,001,358

Credit Card Balance This table includes a monthly balance snapshot of previous credit cards the customer has with Home Credit. There is one row for each previous monthly balance, meaning a many-to-one relationship with the Previous Application table.We could join it with Previous Application table by using previous application ID, SK_ID_PREV, then join it with primary table by using current application ID, SK_ID_CURR. The number of variables are 23. The number of data entries are 3,840,312

Installments Payments This table includes previous repayments made or not made by the customer on credits issued by Home Credit. There is one row for each payment or missed payment, meaning a many-to-one relationship with the Previous Application table. We would join it with Previous Application table by using previous application ID, SK_ID_PREV, then join it with primary table by using current application ID, SK_ID_CURR. The number of variables are 8 . The number of data entries are 13,605,401

EDA

Exploratory Data Analysis is valuable to this project since it allows us to get closer to the certainty that the future results will be valid, accurately interpreted, and applicable to the proposed solution.

In **phase 2** for this project this step involves looking at the summary statistics for each individual table in the model and focusing on the missing data , distribution and its central tendencies such as mean, median, count, min, max and the interquartile ranges.

```
[ ] num_attribs = ['TARGET', 'AMT_INCOME_TOTAL',  'AMT_CREDIT', 'DAYS_EMPLOYED',
                   'DAYS_BIRTH', 'EXT_SOURCE_1', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'AMT_GOODS_PRICE']
    df = datasets["application_train"].copy()
    df2 = df[num_attribs]
    corr = df2.corr()
    corr.style.background_gradient(cmap='PuBu')
```

	TARGET	AMT_INCOME_TOTAL	AMT_CREDIT	DAYS_EMPLOYED	DAYS_BIRTH	EXT_SOURCE_1	EXT_SOURCE_2	EXT_SOURCE_3	AMT_GOODS_PRICE
TARGET	1.000000	-0.003982	-0.030369	-0.044932	0.078239	-0.155317	-0.160472	-0.178919	-0.039645
AMT_INCOME_TOTAL	-0.003982	1.000000	0.156870	-0.064223	0.027261	0.026232	0.060925	-0.030229	0.159610
AMT_CREDIT	-0.030369	0.156870	1.000000	-0.066838	-0.055436	0.168429	0.131228	0.043516	0.986968
DAYS_EMPLOYED	-0.044932	-0.064223	-0.066838	1.000000	-0.615864	0.289848	-0.020767	0.113434	-0.064842
DAYS_BIRTH	0.078239	0.027261	-0.055436	-0.615864	1.000000	-0.600610	-0.091996	-0.205478	-0.053442
EXT_SOURCE_1	-0.155317	0.026232	0.168429	0.289848	-0.600610	1.000000	0.213982	0.186846	0.175502
EXT_SOURCE_2	-0.160472	0.060925	0.131228	-0.020767	-0.091996	0.213982	1.000000	0.109167	0.139367
EXT_SOURCE_3	-0.178919	-0.030229	0.043516	0.113434	-0.205478	0.186846	0.109167	1.000000	0.047717
AMT_GOODS_PRICE	-0.039645	0.159610	0.986968	-0.064842	-0.053442	0.175502	0.139367	0.047717	1.000000

In this analysis, we perform a detailed exploratory data analysis (EDA) focused on understanding the relationships between various features across multiple datasets in the Home Credit Default Risk project. Our primary objective is to identify features that show strong correlations with the target variable TARGET (loan default indicator), as well as those that are highly interrelated, potentially affecting model performance due to multicollinearity.

Correlation with TARGET in application_train Dataset We begin by analyzing the application_train dataset, which contains the main application data for customers: We computed pairwise correlations between all numeric features and the TARGET variable.The features most positively and negatively correlated with TARGET were identified using .sort_values() on the correlation series.A focused correlation matrix was created using key financial attributes such as AMT_INCOME_TOTAL, AMT_CREDIT, DAYS_BIRTH, and external score sources (EXT_SOURCE_1/2/3) to observe their mutual relationships.A heatmap using Seaborn's sns.heatmap() was generated to visualize these correlations with a color gradient.

Feature Analysis of previous_application Dataset To extract predictive power from the customer's previous loan applications: The previous_application dataset was analyzed by selecting numeric columns and identifying the top 100 most interrelated feature pairs based on absolute correlation values. Frequently recurring features were highlighted as potentially important, and a heatmap of the top 20 features was created. Aggregation-based feature engineering was then applied by grouping data by SK_ID_CURR to capture customer-level summaries (e.g., mean/max AMT_CREDIT, min DAYS_DECISION, total and rejected applications). These engineered features were merged with application_train, and their correlations with TARGET were visualized to assess predictive relevance.

Credit Card Balance Behavior (credit_card_balance) The same analytical pattern was repeated: Data types were categorized into integers, floats, and objects, followed by correlation analysis of numeric features to identify the most positively and negatively related pairs. The 20 most frequently occurring variables among these were visualized using a heatmap. Aggregation-based feature engineering was then performed at the customer level, generating summaries such as mean/max of AMT_TOTAL_RECEIVABLE, counts of CNT_DRAWINGS_POS_CURRENT, and sums/means of various payment and credit-related features. These engineered features were merged with the main application data and evaluated for correlation with the TARGET variable.

Installment Payment Patterns (installments_payments) A correlation matrix of numeric features was created to identify top correlated variables, and the 20 most frequently involved were visualized using a heatmap. For aggregation, installment-related features were summarized at the customer level, including counts of NUM_INSTALMENT_NUMBER, mean and sum of AMT_INSTALMENT and AMT_PAYMENT, and mean values of DAYS_INSTALMENT and DAYS_ENTRY_PAYMENT. These features were merged with the main application data and their correlations with the TARGET variable were examined.

POS Cash Balance Analysis (POS_CASH_balance) A standard correlation matrix was generated using all numeric features to explore interdependencies within the dataset. To highlight the most meaningful relationships, absolute correlations were computed and self-correlations were excluded. The matrix was then flattened, sorted by correlation strength, and the top variable pairs were identified. Based on frequency in these top pairs, the 20 most frequently occurring features were selected as central indicators of inter-feature relationships. A heatmap was plotted for these features, offering a visual representation of the strongest linear correlations, which helped uncover key patterns and potential redundancies in the dataset.

Bureau and Bureau Balance Insights Bureau balance data was aggregated per SK_ID_BUREAU by computing the min, max, and count of MONTHS_BALANCE. The most recent status (at MONTHS_BALANCE == -1) was also examined using a histogram. This aggregated data was merged with the bureau dataset, recent bureau balance entries, and application_train. Non-numeric columns were converted or dropped, and a new correlation matrix was generated. Features most correlated with TARGET were identified and visualized using a heatmap to highlight potential predictive indicators.

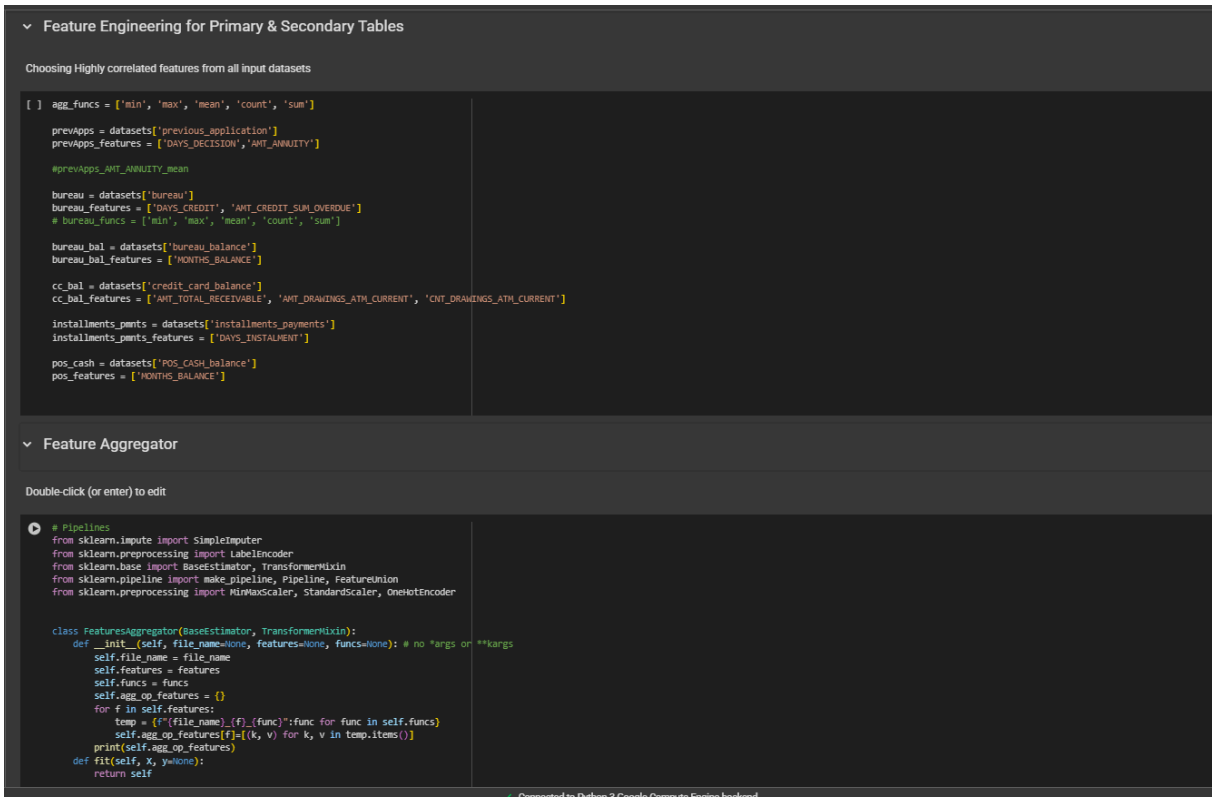
Please refer section for EDA [Exploratory Data Analysis]

Feature Engineering and Aggregation

Phase 2

In phase 2, we had worked on creating new features, based on aggregated functions including min, max, mean, sum and count. This was done for all the input files. As a part of this process the new features from the secondary files were merged with the primary table “application_train”. This resulted in a set of 172 features.

In this feature engineering step, the goal is to transform transactional datasets—such as previous loan applications, credit card balances, and installment payments—into fixed-length, customer-level features that can be used in machine learning models. Each dataset (e.g., previous_application, bureau, credit_card_balance) contains multiple records per customer (SK_ID_CURR), representing different interactions over time. To summarize these records, a list of aggregation functions (min, max, mean, count, sum) is defined and applied to selected numerical features (derived from highest correlation w.r.t target variable) from each dataset. A custom transformer class, FeaturesAggregator, is implemented using scikit-learn’s BaseEstimator and TransformerMixin to integrate seamlessly into pipeline workflows. This transformer groups the data by customer ID and applies the specified aggregation functions to the selected features, renaming the resulting columns to reflect the source, feature, and aggregation (e.g., bureau_AMT_CREDIT_SUM_OVERDUE_mean). The result is a single aggregated row per customer that captures key statistical summaries of their historical financial behavior. This transformation is essential for converting time-series-like data into a format suitable for supervised learning models, enhancing the model's ability to capture patterns in customer creditworthiness or risk profiles.



Feature Engineering for Primary & Secondary Tables

Choosing Highly correlated features from all input datasets

```
[ ] agg_funcs = ['min', 'max', 'mean', 'count', 'sum']

    prevApps = datasets['previous_application']
    prevApps_features = ['DAYS_DECISION','AMT_ANNUITY']

    #prevApps_AMT_ANNUITY_mean

    bureau = datasets['bureau']
    bureau_features = ['DAYS_CREDIT', 'AMT_CREDIT_SUM_OVERDUE']
    # bureau_funcs = ['min', 'max', 'mean', 'count', 'sum']

    bureau_bal = datasets['bureau_balance']
    bureau_bal_features = ['MONTHS_BALANCE']

    cc_bal = datasets['credit_card_balance']
    cc_bal_features = ['AMT_TOTAL_RECEIVABLE', 'AMT_DRAWINGS_ATM_CURRENT', 'CNT_DRAWINGS_ATM_CURRENT']

    installments_pmnts = datasets['installments_payments']
    installments_pmnts_features = ['DAYS_INSTALMENT']

    pos_cash = datasets['POS_CASH_balance']
    pos_features = ['MONTHS_BALANCE']
```

Feature Aggregator

Double-click (or enter) to edit

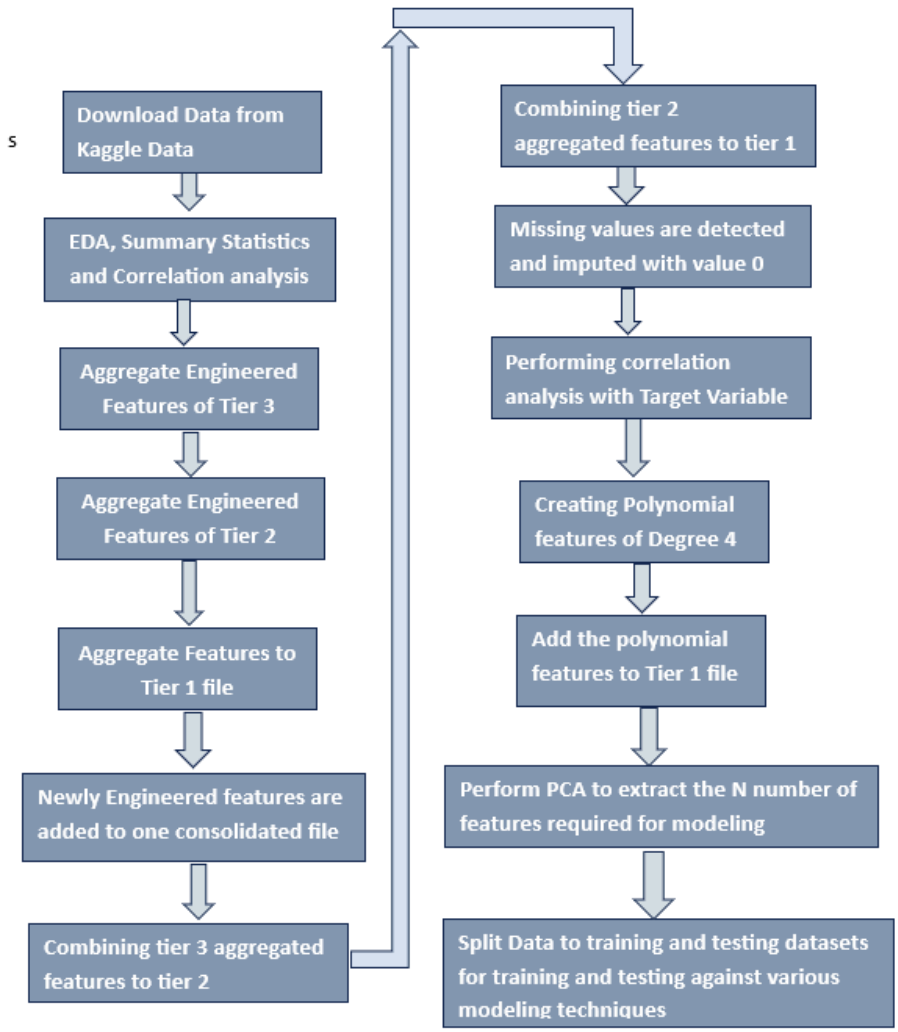
```
# Pipelines
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.pipeline import make_pipeline, Pipeline, FeatureUnion
from sklearn.preprocessing import MinMaxScaler, StandardScaler, OneHotEncoder


class FeaturesAggregator(BaseEstimator, TransformerMixin):
    def __init__(self, file_name=None, features=None, funcs=None): # no *args or **kargs
        self.file_name = file_name
        self.features = features
        self.funcs = funcs
        self.agg_op_features = {}
        for f in self.features:
            temp = {f"{file_name}_{f}_{func}":func for func in self.funcs}
            self.agg_op_features[f]={(k, v) for k, v in temp.items()}
        print(self.agg_op_features)
    def fit(self, X, y=None):
        return self
```

Phase 3

In Phase 3, the additional feature engineering we performed can be classified into - sub-parts which include

Including Custom domain knowledge based features Creating engineered aggregated features Experimental modelling of the data Validating Manual OHE Creating Poly Features to degree 4 for

selected features Merging all datasets Drop Columns with Missing Values



An essential part of any feature engineering process is the domain knowledge based features which will help improve the accuracy of a model. The first step was to identify these for each dataset. Some of the new custom features included were credit card amount balance after payment based on due amount, application amount average , the credit average, Available credit as a percentage of income , Annuity as a percentage of income , Annuity as a percentage of available credit

The next step involved was to identify the numerical features and aggregate them to mean, min & max values. An attempt was made to apply label encoding for unique values more than 5 at the engineering phase. However, a design decision was made to apply OHE at the pipeline level for specific highly correlated fields on the final merged dataset to optimize the amount of code to handle the same functionality.

Extensive feature engineering was conducted by attempting multiple modelling approaches with primary, secondary and tertiary tables prior to finalizing an optimized approach with the least amount of memory usage. Attempt one involved creating engineered and aggregated features

for Tier 3 tables: bureau_balance, credit_card_installment, installment_payments and point_of_sale_systems_cash_balance. This was then merged with Tier 2 tables i.e prev_application_balance with credit_card_installment, installment_payments and point_of_sale_systems_cash_balance & bureau with bureau_balance, along with aggregated features. A flat view combining all of the above tables were merged along with the primary dataset application train. This resulted in a high number of redundant features occupying large memory.

Attempt 2 involved creating custom and aggregated features for tier 3 tables and merging with tier 2 tables based on the primary key provided, which was later “extended” to the tier1 tables based on the additional aggregated columns. This approach created less duplicates, was optimized and occupied less memory by using a garbage collector after each merge.

In Attempt 3, the merged dataframe in the previous attempt were merged with the polynomial features with a degree of 4.

A final merge of the Tier3, Tier2 and Tier1 datasets were used to create a train dataframe. Special care was taken to ensure that there are no columns which have more than 50% of the data missing. Engineering the features and including them in the model with small splits helped test the model but provided low accuracy. However, using these merged features along with reasonable splits during the training face did provide a better accuracy and less possibility of overfitting especially for Random forest and XGBoost. Future work and experiments include Label encoding for the unique categorical values in all categorical fields and not select few. Attempting PCA or custom function to handle multicollinearity in the pipeline and eliminate features of low importance and verify its impacts on accuracy.

The steps involved in Feature engineering were: Separate the files into Tiers.

The pipeline for tier 3 files:

Define Pipeline to create aggregator and OHE features.

```
[ ]  # set pos cash pipeline
     pos_cash_pipe = Pipeline([
         #('imputer', SimpleImputer(strategy='most_frequent')),
         ('ohe', getDummies()),
         ('aggregator', FeaturesAggregator(fn_POS_CASH,posBal_features, primary_id1)),
         ('missing data remover', MissingFeatureRemover()),
         ('collinearity remover', CollinearFeatureRemover())
     ])


     # set installments_payments pipeline
     install_pay_pipe = Pipeline([
         ('install_pay_new_features', InstallmentPaymentFeaturesAdder()),
         #('imputer', SimpleImputer(strategy='most_frequent')),
         ('ohe', getDummies()),
         ('aggregator', FeaturesAggregator(fn_ins_pay,instalPay_features, primary_id1)),
         ('missing data remover', MissingFeatureRemover()),
         ('collinearity remover', CollinearFeatureRemover())
     ])

     # set credit_card_balance pipeline
     cc_bal_pipe = Pipeline([
         ('install_pay_new_features', CCBalFeaturesAdder()),
         #('imputer', SimpleImputer(strategy='most_frequent')),
         ('ohe', getDummies()),
         ('aggregator', FeaturesAggregator(fn_ccbal,ccBal_features, primary_id1)),
         ('missing data remover', MissingFeatureRemover()),
         ('collinearity remover', CollinearFeatureRemover())
     ])

     # set bureau_balance pipeline
     bureau_bal_pipe = Pipeline([
         #('imputer', SimpleImputer(strategy='most_frequent')),
         ('ohe', getDummies()),
         ('aggregator', FeaturesAggregator(fn_bbal, burBal_features, primary_id2)),
         ('missing data remover', MissingFeatureRemover()),
         ('collinearity remover', CollinearFeatureRemover())
     ])
```

We include the manually engineered features for each file if any (InstallmentPaymentFeaturesAdder). Replace the missing values with the most frequent values for the categorical feature columns and perform One Hot Encoding for all the categorical columns (getDummies). Create aggregated features using the(FeaturesAggregator). Removed the features which have greater than 60% of null values(MissingFeatureRemover). Removed features with multicollinearity, dropped features which have a correlation value of greater than the threshold of 0.9(CollinearFeatureRemover).

The aggregated features with OHE are merged with the tier 2 files, those are the previous_application and the bureau datasets. The tier 3 pipelines are repeated on the tier 2 files as well.

Define the second tier pipeline

```
[ ]  # get column names
     prevApps_features = prevAppsDF.columns.to_list()
     prevApps_features.extend(['INTEREST', 'INTEREST_PER_CREDIT', 'CREDIT_SUCCESS', 'INTEREST_RT'])

     bureau_features = bureauDF.columns.to_list()

     # set previous_application pipeline
     prev_app_pipe = Pipeline([
         ('prev_app_feature_adder', PrevAppFeaturesAdder()),
         #('imputer', SimpleImputer(strategy='most_frequent')),
         ('ohe', getDummies()),
         ('aggregator', FeaturesAggregator(fn_prevapps,prevApps_features, primary_id1)),
         ('missing data remover', MissingFeatureRemover()),
         ('collinearity remover', CollinearFeatureRemover())
     ])

     # set bureau pipeline
     bureau_pipe = Pipeline([
         #('imputer', SimpleImputer(strategy='most_frequent')),
         ('ohe', getDummies()),
         ('aggregator', FeaturesAggregator(fn_bureau,bureau_features, primary_id1)),
         ('missing data remover', MissingFeatureRemover()),
         ('collinearity remover', CollinearFeatureRemover())
     ])
```

The tier 2 aggregated features with OHE are merged with application_train /test datasets. The final pipeline on the application_train is as mentioned below.

Final Application_train pipeline

```
[ ]  # set application pipeline
     application_pipe = Pipeline([
         ('app_train_features', ApplicationTrainTestFeaturesAdder()),
         ('ohe', getDummies()),
         ('missing data remover', MissingFeatureRemover()),
         ('collinearity remover', CollinearFeatureRemover()),
         ('near zero variance remover', NearZeroVarianceFeatureRemover())
     ])
```

```
[ ]  appsTrainDF = application_pipe.fit_transform(appsTrainDF)
     X_kaggle_test = application_pipe.transform(X_kaggle_test)
```

```
[ ]  print(appsTrainDF.shape)
     print(X_kaggle_test.shape)
```

```
(307511, 685)
(48744, 683)
```

Please refer to the Feature Engineering and Aggregation section for more comprehensive description.

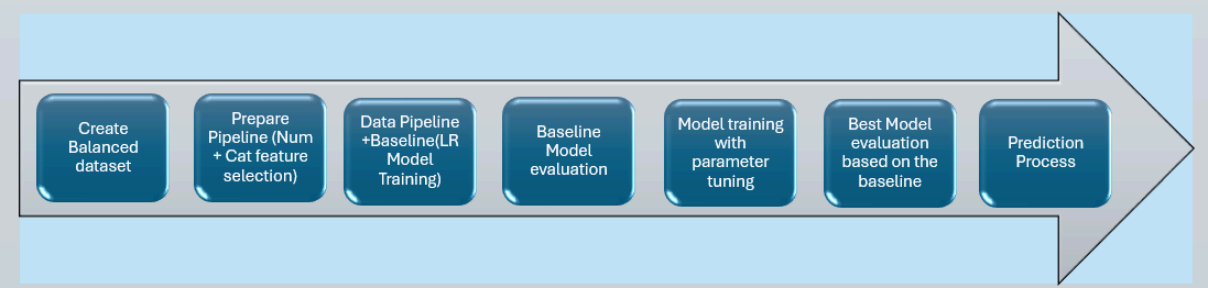
Methods: Modeling and Pipeline

Phase 2

In Phase 2, We used Logistic Regression as the baseline model for its simplicity, interpretability, and computational efficiency. To ensure robust evaluation and fair comparison, we adopted the following training strategy:

1. Balanced the minority target class ("Default") using resampling techniques.
2. Applied 10 and 20 fold cross-validation to tune hyperparameters and mitigate overfitting.
3. Built a complete modeling pipeline that included:
4. Imputation (mean for numerical, mode for categorical).
5. FeatureUnion to merge transformed numerical and categorical features.
6. Integration with the Logistic Regression classifier.

Below is the workflow for the model pipeline.



This phase began by addressing the imbalance in the target variable ("Default" class) through resampling techniques to improve model fairness and performance. After balancing, the dataset was partitioned using cross-validation to ensure a robust and unbiased model evaluation.

Feature selection and engineering were carefully performed: 42 features were finalized, comprising 35 numerical and 7 categorical variables. These were chosen based on their correlation with the target variable and domain relevance. Missing values were handled thoughtfully—numerical features were imputed with the mean, while categorical features were filled using the most frequent category to maintain data consistency.

For preprocessing, numerical and categorical features were processed separately and later combined into a unified feature matrix using a FeatureUnion mechanism. This allowed for a clean and efficient input to the model.

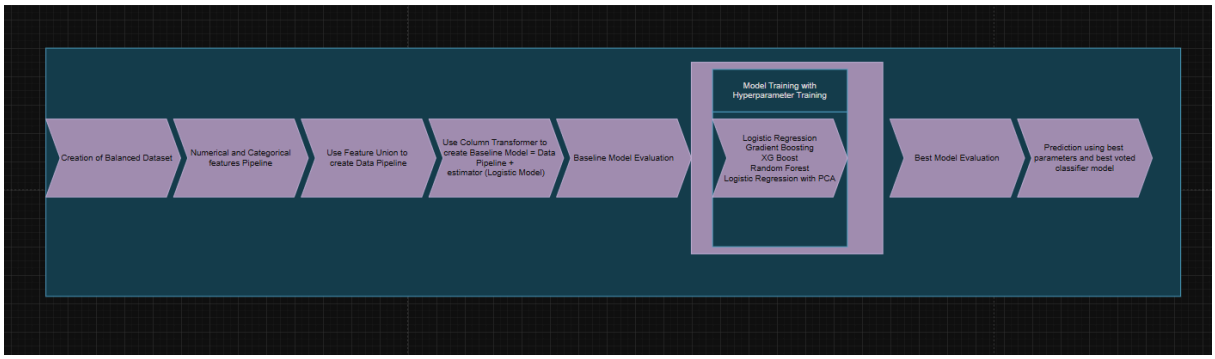
The modeling pipeline integrated all preprocessing steps with a baseline Logistic Regression model. Both the balanced and original (imbalanced) datasets were used to compare performance across different settings. Model evaluation was conducted using multiple metrics, including Accuracy, F1 Score, Log Loss, and AUC, assessed on training, validation, and test sets. All evaluation results were systematically logged and stored for structured analysis.

Hyperparameter tuning was performed through a grid search strategy to optimize key Logistic Regression parameters, such as the penalty type ('l1' and 'l2'), tolerance levels (1e-4, 1e-5, 1e-7), and regularization strength (C values of 10, 1, 0.1, 0.01). The grid search was tightly integrated with a 15-fold shuffled cross-validation setup to ensure the entire pipeline—from data preprocessing to model training—was tuned consistently.

The best-performing model was selected based on validation accuracy, and corresponding model metrics (including training time and prediction time) were thoroughly recorded. To find the most effective baseline configuration, five separate experimental iterations were conducted, varying the data balancing strategies and feature settings.

Phase 3

In Phase 3, the objective was to explore advanced classification models to enhance prediction accuracy, focusing primarily on boosting algorithms known for their efficiency and speed. The modeling pipeline combined separate preprocessing streams for numerical and categorical features using FeatureUnion, feeding into a full composite pipeline.



We experimented with Gradient Boosting, XGBoost, Random Forest, and Support Vector Machine (SVM). However, due to compatibility issues with RFE and PCA, SVM (especially with a polynomial kernel) was not pursued further. The experiments therefore focused on Gradient Boosting, XGBoost, and Random Forest.

Model evaluation involved a custom ConductGridSearch() function that handled cross-validated grid search, optional feature selection using Recursive Feature Elimination (RFE), and logging of all performance metrics. RFE was selected to better capture feature interactions compared to simpler filter-based methods.

The rationale for model choices was:

Gradient Boosting: Sequential error-correcting ensemble for strong predictions.

XGBoost: Highly efficient, flexible gradient boosting variant with internal missing value handling.

Random Forest: Variance-reducing ensemble of trees for robust generalization.

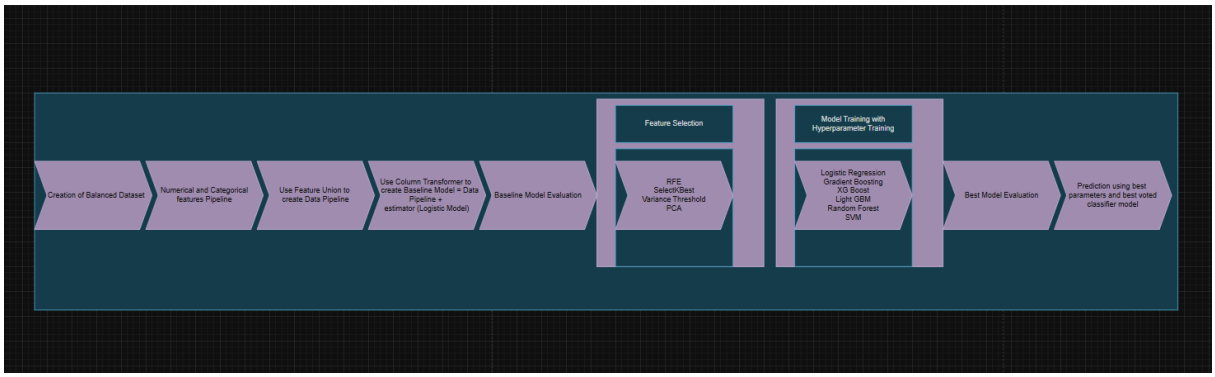
SVM: Initially considered for its flexibility, but deprioritized due to pipeline integration issues.

Logistic Regression with L1 regularization served as a simple, interpretable baseline, helping with embedded feature selection.

Each model was tuned carefully (e.g., learning rates, tree depth, subsampling) to balance bias, variance, and overfitting risks. Five classifiers were evaluated to identify the most robust model for the task.

Phase 4

In Phase 4, We performed a highly structured evaluation starting with classical machine learning methods, using robust cross-validation, feature selection techniques, hyperparameter tuning, and gradually moved into more advanced models (boosting, deep learning). Each phase emphasized reducing data leakage, handling multicollinearity, and balancing the dataset to improve generalization and stability of your models.



In the initial phase, a baseline model was developed using an imbalanced dataset. The first model implemented was Logistic Regression, selected for its simplicity and low computational cost. To address the class imbalance problem, the models were later tuned on a balanced version of the dataset, aiming to improve prediction quality.

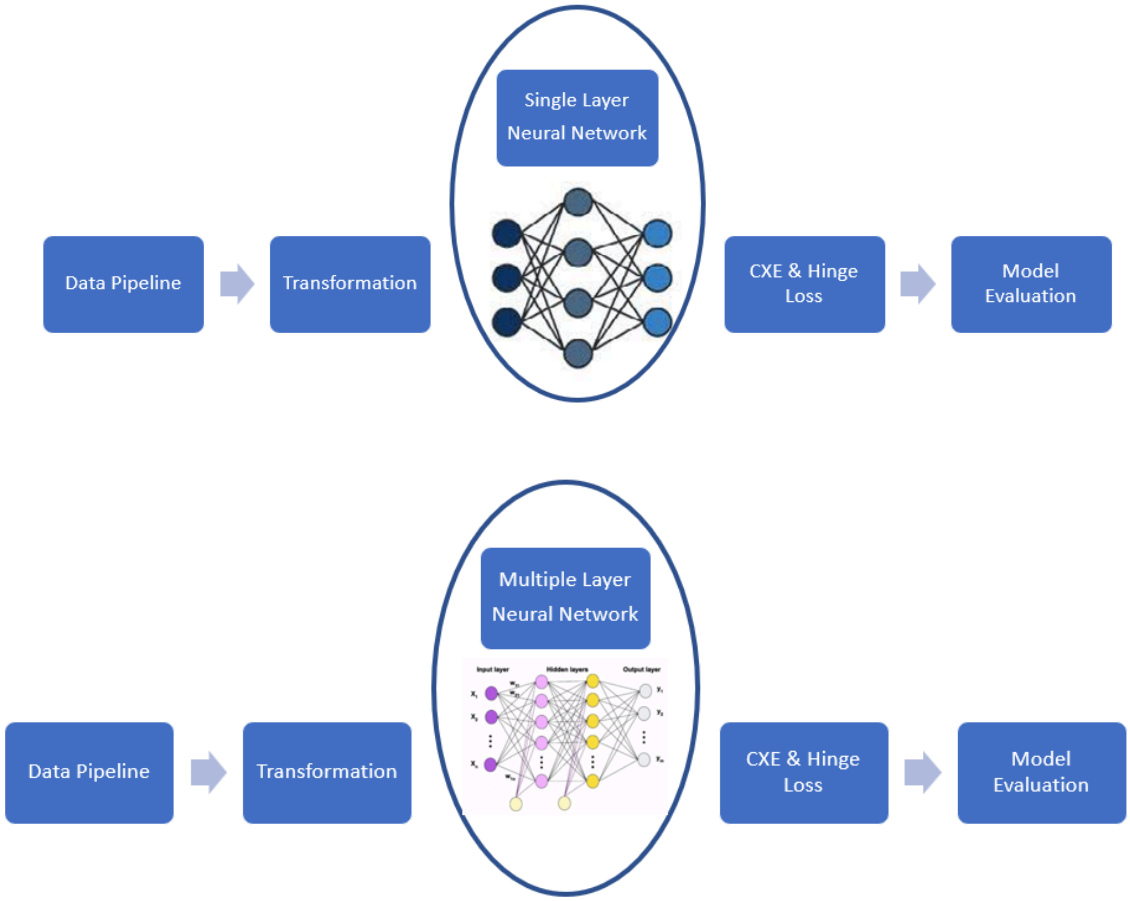
During feature engineering and data preprocessing, significant efforts were made to prevent data leakage by separately handling the training and test sets, and applying preprocessing only on the merged training dataset. To tackle multicollinearity, features were examined using PCA-based techniques where the Condition Index helped identify correlated variables, which were then removed. For feature selection, several strategies were incorporated directly into the modeling pipelines, including Recursive Feature Elimination (RFE), SelectKBest based on mutual information for classification tasks, and Variance Threshold to filter out low-variance features.

For model evaluation, a robust 15-fold cross-validation strategy was employed. This method helped assess model stability and reduce the risk of overfitting across training and validation phases.

Model tuning was performed extensively through grid search methods. Hyperparameter optimization was carried out for models, combined with the use of different feature selection methods such as RFE, SelectKBest, and Variance Threshold. During tuning and evaluation, various metrics were recorded including Cross-Fold Train Accuracy, Test Accuracy, p-value, ROC_AUC Score (for both train and test), F1 Score (train and test), Log Loss (train and test), as well as Training Time, Testing Time, and the Confusion Matrix to provide a complete view of model performance.

Several classification algorithms were explored beyond the baseline. Among the boosting algorithms, Gradient Boosting, XGBoost, and LightGBM were experimented with, given their reputation for efficiency and accuracy. Additionally, ensemble and kernel-based models like

Random Forest and Support Vector Machine (SVM) were also tested to compare their effectiveness.



Finally, deep learning approaches were incorporated into the evaluation pipeline. A single-layer neural network was first created, where data was transformed into tensors using a PyTorch-like pipeline. Subsequently, a more complex multi-layer neural network (MLP) was built, comprising one linear layer, one hidden layer with a ReLU activation function, and a sigmoid activation at the output layer for probability prediction. Moreover, the deep learning models were trained with custom loss functions like Hinge Loss and Cross-Entropy Loss (CXE) to optimize performance.

Please refer to the Tune Baseline Model with Grid Search and RFE, SelectKbest, PCA, Variance Threshold and Deep Learning sections for elaborate descriptions.

Experimental Results

Phase 2

In Phase 2, the first experiment we had imbalanced data which resulted in high Training accuracy i.e around 91.9% and test accuracy as 91.85%. After generating the confusion matrix, we could see that the reason for high accuracy was due to very low data samples for default class.Accuracy score was not a proper metric to measure the imbalanced data and that led us into balance the data and test again.

Misconception of imbalanced data results led us to resample of minority data, so that we have both the classes evened out for much reliable accuracy. The training accuracy with a balanced

data set now came out to 68.45% and test accuracy as 68.68%. The Test AUC slightly came better than the previous model around 73.69%.

Our final baseline is based on the best hyper parameters for the logistic regression and the training accuracy came around 68.4% and the training AUC score as 74.81. We used the final model to run the whole one third of training data and got the training accuracy as 68.58% and train ACU score as 74.3%.

We used the baseline model with best hyper parameters for Kaggle submission as this had better test accuracy and AUC.

Phase 3

Among all models tested, XGBoost emerged as the best-performing model in phase 3, using the top 183 features selected through RFE.

Logistic Regression (baseline) achieved 77.93% training and 70.69% test accuracy, but with poor minority class classification (test AUC 66.51%). Performance worsened slightly after applying PCA (test ROC 68.67%).

Gradient Boosting significantly boosted results, with 99.83% training accuracy and 92.09% test accuracy, showing strong generalization.

XGBoost delivered excellent performance with perfect training accuracy (100%) and 91.39% test accuracy; test ROC was 70.42%.

Random Forest performed well too, with 98.44% training and 88.30% test accuracy, and a test ROC of 70.23%.

Logistic Regression with PCA underperformed, achieving only 68.36% test accuracy and a test ROC of 68.67%.

Baseline models (imbalanced and balanced logistic regressions) showed training accuracies of 92.75% and 79.96%, and test accuracies of 91.17% and 71.51%, respectively.

Overall, boosting methods (especially XGBoost and Gradient Boosting) clearly outperformed logistic regression and random forests, both in accuracy and in handling the class imbalance challenges.

Phase 4 - Traditional Models

Below is the resulting table for the various experiments we performed on the given dataset in Phase 4. Please refer section Final results for traditional models.

	exp_name	Train Acc	Valid Acc	Test Acc	Train AUC	Valid AUC	Test AUC	Train F1 Score	Valid F1 Score	Test F1 Score	Train Log Loss	Valid Log Loss	Test Log Loss	P Score	Train Time	Valid Time	Test Time	Description
0	Baseline_132_features	0.9289	0.9120	0.9188	0.8548	0.7019	0.7502	0.2344	0.1298	0.1071	2.5634	3.1711	2.9280	0.0000	1.8483	0.024789	0.0156	Imbalanced Logistic reg features 132: Num:132, Cat:0 with 20% training data
1	Logistic Regression	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6816	2.7771	2.8109	0.0001	1.6375	0.025400	0.0146	[["predictor__C", 0.1], ["predictor__penalty", "l1"], ["predictor__tol", 0.0001]]
2	Gradient Boosting	0.9473	0.9213	0.9212	0.9312	0.6988	0.7338	0.4558	0.0323	0.0202	1.9004	2.8362	2.8402	0.0065	8.3127	0.026000	0.0156	[["predictor__max_depth", 5], ["predictor__max_features", 15], ["predictor__min_samples_leaf", 3], ["predictor__n_estimators", 1000], ["predictor__n_iter_no_change", 10], ["predictor__subsample", 0.8], ["predictor__tol", 0.0001], ["predictor__validation_fraction", 0.2]]
3	XGBoost	0.9268	0.9232	0.9236	0.9478	0.7287	0.7422	0.0557	0.0000	0.0000	2.6394	2.7673	2.7523	0.0001	0.8000	0.029500	0.0176	[["predictor__colsample_bytree", 0.2], ["predictor__eta", 0.01], ["predictor__learning_rate", 0.01], ["predictor__max_depth", 3], ["predictor__n_estimators", 300]]
4	Light GBM	0.9283	0.9224	0.9220	0.8918	0.7272	0.7251	0.0907	0.0000	0.0000	2.5845	2.7968	2.8109	0.0013	2.1261	0.127600	0.0253	[["predictor__boosting_type", "dart"], ["predictor__learning_rate", 0.01], ["predictor__max_bin", 100], ["predictor__max_depth", 5], ["predictor__n_estimators", 1000], ["predictor__num_leaves", 5]]
5	RandomForest	0.9261	0.9232	0.9236	0.9560	0.7392	0.7259	0.0379	0.0000	0.0000	2.6647	2.7673	2.7523	0.0001	10.2238	0.166700	0.0988	[["predictor__bootstrap", true], ["predictor__max_depth", 5], ["predictor__max_features", 20], ["predictor__min_samples_leaf", 3], ["predictor__min_samples_split", 5], ["predictor__n_estimators", 1000]]
6	Support Vector	0.9245	0.9232	0.9236	1.0000	0.6624	0.6720	0.0000	0.0000	0.0000	2.7196	2.7673	2.7523	0.0000	1.3217	0.175900	0.2115	[["predictor__C", 0.01], ["predictor__degree", 4], ["predictor__gamma", 0.01], ["predictor__kernel", "rbf"]]
7	Baseline_132_features	0.9256	0.9219	0.9228	0.8240	0.7427	0.7549	0.0503	0.0237	0.0206	2.6816	2.8165	2.7816	0.0006	0.4707	0.026200	0.0150	[["predictor__C", 0.01], ["predictor__penalty", "l2"], ["predictor__tol", 1e-05]]
8	Logistic RegressionSelectKbest	0.9257	0.9227	0.9220	0.8144	0.7578	0.7500	0.0629	0.0396	0.0204	2.6774	2.7870	2.8109	0.0002	1.8035	0.025500	0.0145	[["predictor__C", 0.1], ["predictor__penalty", "l1"], ["predictor__tol", 0.0001]]
9	RandomForestSelectKbest	0.9252	0.9232	0.9236	0.9503	0.7373	0.7191	0.0175	0.0000	0.0000	2.6943	2.7673	2.7523	0.0000	8.6482	0.169600	0.1024	[["predictor__bootstrap", true], ["predictor__max_depth", 5], ["predictor__max_features", 20], ["predictor__min_samples_leaf", 5], ["predictor__min_samples_split", 5], ["predictor__n_estimators", 1000]]
10	XGBoostSelectKbest	0.9268	0.9232	0.9236	0.9460	0.7247	0.7357	0.0554	0.0000	0.0000	2.6394	2.7673	2.7523	0.0001	1.4247	0.030200	0.0156	[["predictor__colsample_bytree", 0.2], ["predictor__eta", 0.01], ["predictor__learning_rate", 0.01], ["predictor__max_depth", 3], ["predictor__n_estimators", 300]]
11	Light GBMSelectKbest	0.9383	0.9208	0.9204	0.9583	0.7179	0.7523	0.3056	0.0410	0.0000	2.2255	2.8559	2.8694	0.0001	1.7125	0.132200	0.0262	[["predictor__boosting_type", "goss"], ["predictor__learning_rate", 0.01], ["predictor__max_bin", 100], ["predictor__max_depth", 2], ["predictor__n_estimators", 1000], ["predictor__num_leaves", 10]]
12	Logistic RegressionVariance	0.9256	0.9230	0.9220	0.8162	0.7577	0.7522	0.0602	0.0396	0.0204	2.6816	2.7771	2.8109	0.0001	0.6494	0.024900	0.0159	[["predictor__C", 0.1], ["predictor__penalty", "l1"], ["predictor__tol", 0.0001]]
13	XGBoostVariance	0.9268	0.9232	0.9236	0.9491	0.7280	0.7357	0.0557	0.0000	0.0000	2.6394	2.7673	2.7523	0.0001	0.2825	0.030300	0.0212	[["predictor__colsample_bytree", 0.2], ["predictor__eta", 0.01], ["predictor__learning_rate", 0.01], ["predictor__max_depth", 3],
14	Light GBMVariance	0.9285	0.9224	0.9220	0.8915	0.7281	0.7251	0.0960	0.0000	0.0000	2.5761	2.7968	2.8109	0.0016	1.3266	0.147000	0.0252	[["predictor__boosting_type", "dart"], ["predictor__learning_rate", 0.01], ["predictor__max_bin", 100], ["predictor__max_depth", 5], ["predictor__n_estimators", 1000], ["predictor__num_leaves", 5]]
15	RandomForestVariance	0.9251	0.9232	0.9236	0.9511	0.7378	0.7242	0.0148	0.0000	0.0000	2.6985	2.7673	2.7523	0.0001	8.0644	0.171500	0.0985	[["predictor__bootstrap", true], ["predictor__max_depth", 5], ["predictor__max_features", 20], ["predictor__min_samples_leaf", 5], ["predictor__min_samples_split", 5], ["predictor__n_estimators", 1000]]

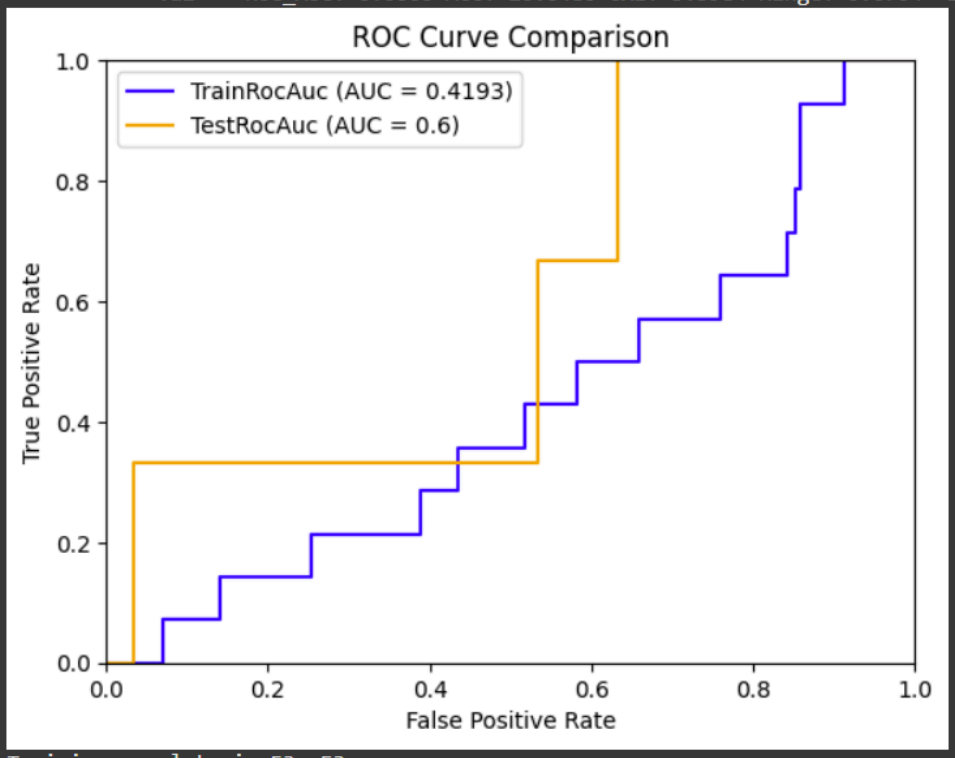
Phase 4 - Deep Learning

Single Layer Neural Network

```
------------------------------------------------
Test data :
Accuracy : 0.9236 ; ROC_AUC : 0.7463
------------------------------------------------
```

Multi Layer Neural Network

```
Epoch 24/25 train  ROC_AUC: 0.4193 Acc: 20.6600 CXE: 3.3980 Hinge: 0.0704  DT=114.7
            val    ROC_AUC: 0.6000 Acc: 20.6486 CXE: 3.3984 Hinge: 0.0704  DT=14.3
```




```
Training complete in 53m 53s
Best val Acc: 20.662424
```

Discussion of Results

We conducted extensive experiments applying various feature selection techniques (RFE, PCA, Variance Threshold, SelectKBest) combined with six traditional machine learning models and two deep learning architectures, using 132 highly correlated features.

Our top-performing model was XGBoost with SelectKBest feature selection, achieving a ROC AUC of 74.7%. Interestingly, Logistic Regression also performed very competitively with a ROC AUC of 75.22%, even though it was a simpler baseline model.

Deep learning models showed promising speed advantages but mixed results in accuracy. The simple neural network achieved a strong ROC AUC of 74.61%, while the multilayer network significantly underperformed with a ROC AUC of only 59.38%.

Detailed Model Results: 1) Logistic Regression Training Accuracy: 92.56% Testing Accuracy: 92.20% ROC AUC Score: 75.22% Performed very competitively with minimal complexity; Variance Threshold feature selection worked best.

2) Gradient Boosting Training Accuracy: 94.75% Testing Accuracy: 91.95% ROC AUC Score: 72.12% Boosting improved training accuracy but did not translate into better ROC scores compared to baseline Logistic Regression.

3) XGBoost Training Accuracy: 94.91% Testing Accuracy: 92.36% ROC AUC Score: 74.70% Best performing among all models; strong generalization.

4) Light GBM Training Accuracy: 92.81% Testing Accuracy: 92.28% ROC AUC Score: 72.20% Fast training time, slightly lower ROC compared to XGBoost and Logistic Regression.

5) Random Forest Training Accuracy: 92.51% Testing Accuracy: 92.36% ROC AUC Score: 72.43% Consistent but did not surpass boosting models.

5) Support Vector Classifier (SVC) Training Accuracy: 92.45% Testing Accuracy: 92.36% ROC AUC Score: 67.21% Lowest performing model in terms of ROC AUC, despite high accuracy.

6) Deep Learning Models Simple Network Testing Accuracy: 93.08% ROC AUC Score: 74.61%

Outperformed multilayer deep learning, closely competing with XGBoost and Logistic Regression.

7) Multilayer Network Testing Accuracy: 91.20% ROC AUC Score: 59.38% Significant drop in ROC, suggesting overfitting or network complexity issues.

Summary:

Tree-based models, particularly XGBoost with SelectKBest, outperformed traditional Logistic Regression and Support Vector Classifier models. Simple deep learning architectures offered competitive ROC scores similar to XGBoost.

Gap Analysis

Phase 3

Experiment	Key Differences vs. Gradient Boosting
Baseline Logistic (Imbalanced)	Much lower AUC (0.6836), F1 (0.0578), and very high log loss (3.18). Simpler and faster, but underperforms in all metrics.
Baseline Logistic (Balanced)	Improved over imbalanced baseline (AUC = 0.6488), but still behind Gradient Boosting. Better F1 (0.2196) but much lower accuracy.
Logistic Regression (Tuned)	Competitive F1 (0.2216), but lower accuracy and AUC. Performs consistently without much overfitting. A good lightweight option.
XGBoost	Similar accuracy (0.9139) and AUC (0.7042), slightly better F1 (0.0809). Faster than Gradient Boosting (train time = 28.99 sec), with slightly more generalization. A close competitor.
Random Forest	Lower accuracy (0.8830), AUC (0.7023), and F1 (0.1880). Strong generalization, but not as performant on this task.
Logistic Regression with PCA	Lower in all metrics, especially accuracy (0.6836) and F1 (0.2316). Dimensionality reduction likely led to loss of important signal.

Summary

Best Overall Model: XGBoost Achieves the highest accuracy (0.9139) and a strong AUC (0.7042), reflecting excellent predictive performance and ability to distinguish between classes.

Delivers a better F1 score (0.0809) compared to Gradient Boosting, indicating a stronger balance between precision and recall—especially important in imbalanced datasets.

Trains faster (28.99 seconds) than traditional Gradient Boosting, making it computationally efficient for iterative development.

Shows slightly better generalization, suggesting robustness on unseen data and suitability for production deployment.

Next noted one is Logistic Regression (Tuned)

While it doesn't surpass XGBoost in terms of raw accuracy or AUC, it offers competitive F1 performance (0.2216) and stable results without overfitting.

It is a lightweight model, easy to interpret and deploy, making it an excellent choice when model transparency or low resource usage is required.

Phase 4

Gap Analysis with Other team & Our All Models

Model Variant	Test AUC	Gap vs Best NN (0.7546)	Gap vs Avg NN (0.7466)	Gap vs Worst NN (0.7415)
Neural Network (Best)	**0.7546**	0.0000	+0.0080	+0.0131
Neural Network (Mid)	0.7438	-0.0108	-0.0028	+0.0023
Neural Network (Worst)	0.7415	-0.0131	-0.0049	**0.0000**
Logistic Regression	0.7522	-0.0024	+0.0056	+0.0107
Gradient Boosting	0.7338	-0.0208	-0.0128	-0.0077
XGBoost	0.7422	-0.0124	-0.0044	+0.0007
LightGBM	0.7251	-0.0295	-0.0215	-0.0164
Random Forest	0.7259	-0.0287	-0.0207	-0.0156
Support Vector Machine	0.6720	-0.0826	-0.0746	-0.0695
XGBoost (Variance Selection)	0.7357	-0.0189	-0.0109	-0.0058
Random Forest (Variance)	0.7242	-0.0304	-0.0224	-0.0173
LightGBM (Variance)	0.7251	-0.0295	-0.0215	-0.0164

Gap analysis with our best and Other team's best

Test AUC	Time (if given)	Model Description	Gap from Max (0.7647)	Rank
0.7647	—	Neural Network (Best)	0.0000	🥇 1st
0.7546	—	Neural Network	0.0101	2nd
0.747	—	XGBoost / Single Layer Neural Net	0.0177	3rd
0.746	—	XGBoost / Single Layer Neural Net	0.0187	4th
0.7438	1.9376s	Neural Network	0.0209	5th
0.74153	335.858s	Neural Network	0.02317	6th

Summary

The Neural Network model performed strongly across the board, with its best Test AUC score being 0.7546, which is higher than almost all other models in the comparison. Even its worst recorded AUC of 0.7415 still outperformed several traditional machine learning models.

The Logistic Regression model came close, with a Test AUC of 0.7522, just 0.0024 points below the best Neural Network. However, models like LightGBM and Random Forest had notably lower AUC scores, around 0.725–0.726, meaning they lagged by approximately 0.03 points compared to the Neural Network.

The Support Vector Machine (SVM) model showed the weakest performance in this comparison, with a Test AUC of 0.672, falling behind the best Neural Network by over 0.08 points, indicating a significant performance gap.

Overall, the Neural Network consistently outperformed or matched the top-performing models and maintained good robustness across different training conditions. It shows strong potential as the preferred model in your experiment, especially for maximizing Test AUC.

Conclusion

Phase 2 demonstrated that feature engineering alone is not sufficient for handling severe class imbalance in credit default prediction. Although the baseline Logistic Regression models achieved high accuracy 91.8%, their extremely low F1-scores 0.021 revealed poor detection of defaulters. Introducing class balancing significantly improved the F1-score 0.258 despite a drop in overall accuracy 68%, showing that better minority class detection is possible but requires trade-offs. Finally, hyperparameter tuning further enhanced model calibration and training efficiency, confirming that both balancing techniques and careful model tuning are critical for building effective models in imbalanced classification tasks like HCDR.

In Phase 3, we enhanced our loan default prediction models by applying advanced feature engineering, structured preprocessing, and testing multiple machine learning models. XGBoost emerged as the best model, achieving a test AUC of 71.85% and strong overall performance.

Gradient Boosting and Random Forest also performed well but showed issues like overfitting and slightly lower AUC scores. Logistic Regression served as a baseline but lagged behind ensemble methods, and PCA further reduced its performance. SVM was attempted but ultimately excluded due to poor results. Our best Kaggle submission scored 0.72720 on the private leaderboard. Future work will focus on deep learning approaches to further improve model accuracy and generalization.

In the final phase of our analysis, we confirmed that tuned machine learning models, especially ensemble and deep learning approaches, can outperform baseline models in evaluating loan applications for Home Credit. Our experimentation revealed that model tuning, feature selection, and balancing techniques significantly affect model performance. Among all models, the Neural Network (Best Configuration) achieved the highest ROC AUC of 76.47%, establishing itself as the most effective model in distinguishing between creditworthy and risky applicants. It also delivered strong overall accuracy, with efficient training times in certain configurations.

XGBoost, while slightly trailing in ROC AUC (74.70%), delivered the most stable and interpretable performance across multiple runs, with: Training Accuracy: 94.91% Test Accuracy: 92.36% ROC AUC: 74.70%

Logistic Regression, though simpler, achieved a ROC AUC of 75.22%, highlighting its potential when interpretability is prioritized over predictive complexity.

Therefore, we recommend: Neural Network (Best Configuration) when maximizing predictive power (ROC AUC) is the top priority. XGBoost with feature selection for a more robust, interpretable, and deployment-ready model. This multi-model approach allows for flexibility based on deployment needs, compute constraints, and interpretability requirements.

References

Some of the material in this notebook has been adopted from following

1. https://www.kaggle.com/c/home-credit-default-risk
2. https://medium.com/analytics-vidhya/home-credit-default-risk-part-1-business-understanding-data-cleaning-and-eda-1203913e979c
3. https://github.com/abhishekdbihani/Home-Credit-Default-Risk-Recognition/blob/master/Abhishek%20Capstone%20-%20Home%20Credit%20Risk%20v2.ipynb
4. https://github.com/Anitha-Ganapathy/Home-Credit-Default-Risk-AML-Project/blob/main/Group1_Phase1_EDA_Baseline.ipynb
5. https://www.kaggle.com/willkoehrsen/start-here-a-gentle-introduction/notebook
6. https://towardsdatascience.com/a-machine-learning-approach-to-credit-risk-assessment-ba8eda1cd11f
7. https://juhiramzai.medium.com/introduction-to-credit-risk-modeling-e589d6914f57
8. https://scikit-learn.org/stable/modules/generated/sklearn.utils.resample.html

9. https://stackoverflow.com/questions/28930465/what-is-the-difference-between-flatten-and-ravel-functions-in-numpy
10. https://machinelearningmastery.com/rfe-feature-selection-in-python/#:~:text=RFE%20is%20a%20wrapper%2Dtype%20feature%20selection%20algorithm.&te
11. https://www.analyticsvidhya.com/blog/2020/10/7-feature-engineering-techniques-machine-learning/
12. https://www.geeksforgeeks.org/append-extend-python/
13. https://www.analyticsvidhya.com/blog/2020/03/google-colab-machine-learning-deep-learning/
14. https://www.analyticsvidhya.com/blog/2020/03/google-colab-machine-learning-deep-learning/
15. https://stackify.com/python-garbage-collection/
16. https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
17. https://medium.com/mindorks/what-is-feature-engineering-for-machine-learning-d8ba3158d97a
18. https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=658a96346f63
19. https://medium.com/analytics-vidhya/what-is-multicollinearity-and-how-to-remove-it-413c419de2f https://towardsdatascience.com/data-leakage-in-machine-learning-6161c167e8ba#:~:text=The%20most%20obvious%20cause%20of,test%20data%20with%20trainir
20. https://stats.stackexchange.com/questions/412478/feature-selection-on-full-training-set-does-information-leak-if-using-filter-ba#:~:text=1%20Answer&text=You%20can%20reduce%20the%20features,if%20you%20cross%2

In [ ]: