TEAM 16

Aravind Reddy Sheru

Email: asheru@iu.edu (mailto:asheru@iu.edu)

Sai Charan Chintala

Email: sachin@iu.edu (mailto:sachin@iu.edu)

Seongbo Sim

Email: simseo@iu.edu (mailto:simseo@iu.edu)

Yun Joo An

Email: yunjooan@iu.edu (mailto:yunjooan@iu.edu)

In [1]:

```
import warnings
warnings.simplefilter('ignore')

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import re
from time import time
from scipy import stats
import json

from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder

from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

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.pipeline import Pipeline, FeatureUnion
from sklearn.metrics import make_scorer, roc_auc_score, log_loss, accuracy_score
from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import confusion_matrix
from IPython.display import display, Math, Latex


from sklearn.linear_model import Lasso,Ridge,LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb

# Read data from application_train dataset.
data = pd.read_csv('application_train.csv')
data.head()
```

Out[1]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_O
0	100002	1	Cash loans	M	N	
1	100003	0	Cash loans	F	N	

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_O
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

In [2]:
```
y = data['TARGET']
X = data.drop(['SK_ID_CURR','TARGET'], axis = 1)
```

In [3]:

```
experimentLog = pd.DataFrame(columns=["ExpID", "Cross fold train accuracy", "Tes

def pct(x):
    return round(100*x,1)


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


def returnModel(x,y,experimentLog,description_text):
    num_attribs = []
    cat_attribs = []

    for col in x.columns.tolist():
        if x[col].dtype in (['int','float']):
            num_attribs.append(col)
        else:
            cat_attribs.append(col)

    le_dict = {}
    for col in x.columns.tolist():
        if X[col].dtype == 'object':
            le = LabelEncoder()
            x[col] = x[col].fillna("NULL")
            x[col] = le.fit_transform(x[col])
            le_dict['le_{}'.format(col)] = le

    num_pipeline =Pipeline([('selector',DataFrameSelector(num_attribs)),
                           ('scaler', StandardScaler()),
                          ('imputer', SimpleImputer(strategy = 'median'))
                           ])


    cat_pipeline = Pipeline([
        ('selector', DataFrameSelector(cat_attribs)),
        ('imputer', SimpleImputer(strategy='most_frequent'))
    ])


    full_pipeline = FeatureUnion(transformer_list=[
        ("num_pipeline", num_pipeline),
        ("cat_pipeline", cat_pipeline),
    ])

    np.random.seed(42)
    full_pipeline_with_predictor = Pipeline([
            ("preparation", num_pipeline),
            ("linear", LogisticRegression(random_state=42))
        ])
    # split 20% test data with random seed set to 42 for correct experimentLog
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20, ra
    x_train, x_valid, y_train, y_valid = train_test_split(x_train, y_train, test_
    print("train data set: ")
    print(x_train.shape,y_train.shape)
    print("test data set: ")
    print(x_test.shape,y_test.shape)
    print("validation data set: ")
    print(x_valid.shape,y_valid.shape)


    start = time()
    full_pipeline_with_predictor.fit(x_train, y_train)
    np.random.seed(42)

    cv30Splits = ShuffleSplit(n_splits = 30, test_size = 0.3, random_state = 0)
    logit_scores = cross_val_score(full_pipeline_with_predictor, x_train, y_trai
    logit_score_train = logit_scores.mean()
    train_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)

    start = time()
    logit_score_valid  = full_pipeline_with_predictor.score(x_valid, y_valid)
    valid_time = np.round(time() - start, 4)

    AUC = roc_auc_score(y_test,full_pipeline_with_predictor.predict(x_test))
    print("AUC is {}".format(AUC))
    print("\n............\n")
    print("Confusion Matrix: {}".format(confusion_matrix(y_test, full_pipeline_w

    no_of_inputs = x.shape[1]

    temp_df = pd.DataFrame()
    temp_df = temp_df.append(pd.Series(["Baseline with {} inputs".format(no_of_i
                  AUC, train_time, test_time, "{} - Baseline LogisticRegression"
    temp_df.columns = experimentLog.columns

    experimentLog = experimentLog.append(temp_df,ignore_index=True)

    return le_dict, full_pipeline_with_predictor, experimentLog
```

In [4]:
```
le_dict, full_pipeline_with_predictor, experimentLog = returnModel(X,y,experimen
```

```
train data set:
(196806, 120) (196806,)
test data set:
(61503, 120) (61503,)
validation data set:
(49202, 120) (49202,)
AUC is 0.5043329019692199

............

Confusion Matrix: [[56507    47]
 [ 4902    47]]
```

In [5]:
```
experimentLog
```

Out[5]:

	ExpID	Cross fold train accuracy	Test Accuracy	Validation Accuracy	AUC	Train Time(s)	Test Time(s)	Experiment description
0	Baseline with 120 inputs	92.0	92.0	91.8	0.504333	110.2988	0.0994	All features Dataset - Baseline LogisticRegres...

Additional EDA

Discarding features with null values more than 30%

In [6]:

```
null_data = X.isna().sum().reset_index().rename(columns={'index':'col_name',0:'nu
null_data['count_%'] = null_data['null_count']/len(X)*100
null_data = null_data[null_data['count_%'] <= 30]
null_data
```

Out[6]:

	col_name	null_count	count_%
0	NAME_CONTRACT_TYPE	0	0.000000
1	CODE_GENDER	0	0.000000
2	FLAG_OWN_CAR	0	0.000000
3	FLAG_OWN_REALTY	0	0.000000
4	CNT_CHILDREN	0	0.000000
...	...	...	...
115	AMT_REQ_CREDIT_BUREAU_DAY	41519	13.501631
116	AMT_REQ_CREDIT_BUREAU_WEEK	41519	13.501631
117	AMT_REQ_CREDIT_BUREAU_MON	41519	13.501631
118	AMT_REQ_CREDIT_BUREAU_QRT	41519	13.501631
119	AMT_REQ_CREDIT_BUREAU_YEAR	41519	13.501631

75 rows × 3 columns

In [7]:

```
selected_columns = null_data['col_name'].tolist() + ['TARGET']
print(selected_columns)
```

```
['NAME_CONTRACT_TYPE', 'CODE_GENDER', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'CNT_C
HILDREN', 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE',
'NAME_TYPE_SUITE', 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_FAMILY_STAT
US', 'NAME_HOUSING_TYPE', 'REGION_POPULATION_RELATIVE', 'DAYS_BIRTH', 'DAYS_EMP
LOYED', 'DAYS_REGISTRATION', 'DAYS_ID_PUBLISH', 'FLAG_MOBIL', 'FLAG_EMP_PHONE',
'FLAG_WORK_PHONE', 'FLAG_CONT_MOBILE', 'FLAG_PHONE', 'FLAG_EMAIL', 'OCCUPATION_
TYPE', 'CNT_FAM_MEMBERS', 'REGION_RATING_CLIENT', 'REGION_RATING_CLIENT_W_CIT
Y', 'WEEKDAY_APPR_PROCESS_START', 'HOUR_APPR_PROCESS_START', 'REG_REGION_NOT_LI
VE_REGION', 'REG_REGION_NOT_WORK_REGION', 'LIVE_REGION_NOT_WORK_REGION', 'REG_C
ITY_NOT_LIVE_CITY', 'REG_CITY_NOT_WORK_CITY', 'LIVE_CITY_NOT_WORK_CITY', 'ORGAN
IZATION_TYPE', 'EXT_SOURCE_2', 'EXT_SOURCE_3', 'FONDKAPREMONT_MODE', 'HOUSETYPE
_MODE', 'WALLSMATERIAL_MODE', 'EMERGENCYSTATE_MODE', 'OBS_30_CNT_SOCIAL_CIRCL
E', 'DEF_30_CNT_SOCIAL_CIRCLE', 'OBS_60_CNT_SOCIAL_CIRCLE', 'DEF_60_CNT_SOCIAL_
CIRCLE', 'DAYS_LAST_PHONE_CHANGE', 'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3', 'FLAG_
DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7', 'FLAG_DOC
UMENT_8', 'FLAG_DOCUMENT_9', 'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCU
MENT_12', 'FLAG_DOCUMENT_13', 'FLAG_DOCUMENT_14', 'FLAG_DOCUMENT_15', 'FLAG_DOC
UMENT_16', 'FLAG_DOCUMENT_17', 'FLAG_DOCUMENT_18', 'FLAG_DOCUMENT_19', 'FLAG_DO
CUMENT_20', 'FLAG_DOCUMENT_21', 'AMT_REQ_CREDIT_BUREAU_HOUR', 'AMT_REQ_CREDIT_B
UREAU_DAY', 'AMT_REQ_CREDIT_BUREAU_WEEK', 'AMT_REQ_CREDIT_BUREAU_MON', 'AMT_REQ
_CREDIT_BUREAU_QRT', 'AMT_REQ_CREDIT_BUREAU_YEAR', 'TARGET']
```

In [8]:
```
null_data['col_type'] = null_data['col_name'].apply(lambda x: X[x].dtype)
null_data[null_data['count_%'] > 0]
```

Out[8]:

	col_name	null_count	count_%	col_type
7	AMT_ANNUITY	12	0.003902	float64
8	AMT_GOODS_PRICE	278	0.090403	float64
27	CNT_FAM_MEMBERS	2	0.000650	float64
40	EXT_SOURCE_2	660	0.214626	float64
41	EXT_SOURCE_3	60965	19.825307	float64
89	OBS_30_CNT_SOCIAL_CIRCLE	1021	0.332021	float64
90	DEF_30_CNT_SOCIAL_CIRCLE	1021	0.332021	float64
91	OBS_60_CNT_SOCIAL_CIRCLE	1021	0.332021	float64
92	DEF_60_CNT_SOCIAL_CIRCLE	1021	0.332021	float64
93	DAYS_LAST_PHONE_CHANGE	1	0.000325	float64
114	AMT_REQ_CREDIT_BUREAU_HOUR	41519	13.501631	float64
115	AMT_REQ_CREDIT_BUREAU_DAY	41519	13.501631	float64
116	AMT_REQ_CREDIT_BUREAU_WEEK	41519	13.501631	float64
117	AMT_REQ_CREDIT_BUREAU_MON	41519	13.501631	float64
118	AMT_REQ_CREDIT_BUREAU_QRT	41519	13.501631	float64
119	AMT_REQ_CREDIT_BUREAU_YEAR	41519	13.501631	float64

Filling null values in NAME_TYPE_SUITE column with "Other_C"

In [9]:
```
X_feature = data[selected_columns]
X_feature['NAME_TYPE_SUITE'].fillna('Other_C', inplace=True)
X_feature.head()
```

Out[9]:

	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDRE
0	Cash loans	M	N	Y	
1	Cash loans	F	N	N	
2	Revolving loans	M	Y	Y	
3	Cash loans	F	N	Y	
4	Cash loans	M	N	Y	

5 rows × 76 columns

Filling null values in the columns containing keyword AMT_REQ_CREDIT column with 0

In [10]:

```
temp_col_reqd = null_data[null_data['null_count'] != 0].reset_index(drop=True)['
for col in temp_col_reqd:
    if 'AMT_REQ_CREDIT' in col:
        print("columns to be filled with 0 is: {}".format(col))
        X_feature[col].fillna(0,inplace=True)
```

```
columns to be filled with 0 is: AMT_REQ_CREDIT_BUREAU_HOUR
columns to be filled with 0 is: AMT_REQ_CREDIT_BUREAU_DAY
columns to be filled with 0 is: AMT_REQ_CREDIT_BUREAU_WEEK
columns to be filled with 0 is: AMT_REQ_CREDIT_BUREAU_MON
columns to be filled with 0 is: AMT_REQ_CREDIT_BUREAU_QRT
columns to be filled with 0 is: AMT_REQ_CREDIT_BUREAU_YEAR
```

Filling null values in the column containing keyword CNT_SOCIAL_CIRCLE with 0

In [11]:

```
for col in temp_col_reqd:
    if 'CNT_SOCIAL_CIRCLE' in col:
        print("columns to be filled with 0 is: {}".format(col))
        X_feature[col].fillna(0,inplace=True)
```

```
columns to be filled with 0 is: OBS_30_CNT_SOCIAL_CIRCLE
columns to be filled with 0 is: DEF_30_CNT_SOCIAL_CIRCLE
columns to be filled with 0 is: OBS_60_CNT_SOCIAL_CIRCLE
columns to be filled with 0 is: DEF_60_CNT_SOCIAL_CIRCLE
```

Filling null values in the column CNT_FAM_MEMBERS with median

In [12]:

```
for col in temp_col_reqd:
    if 'CNT_FAM_MEMBERS' in col:
        print("columns to be filled with median is: {}".format(col))
        X_feature[col].fillna(X_feature[col].median(),inplace=True)
```

```
columns to be filled with median is: CNT_FAM_MEMBERS
```

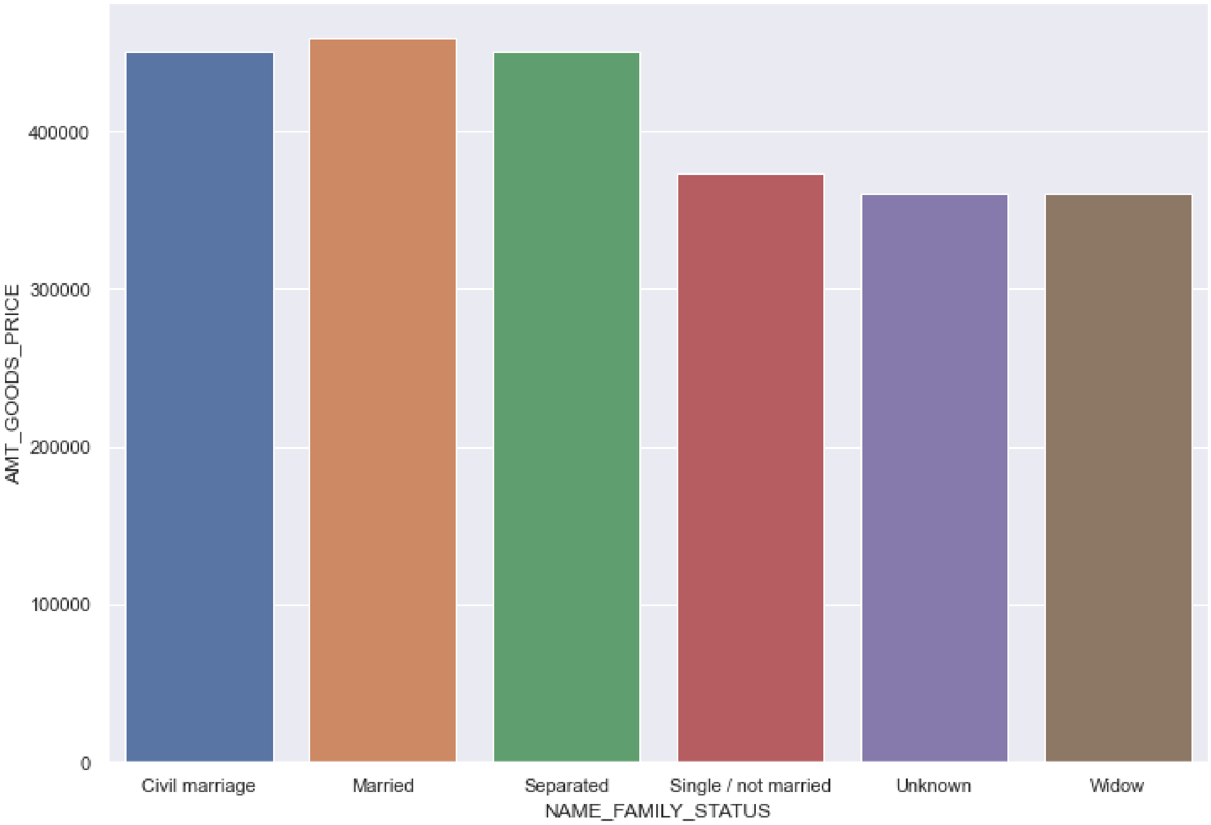
Filling null values in the column AMT_GOODS_PRICE with median fopr the respective category

In [13]:
```
temp_plt_data = X_feature[['AMT_GOODS_PRICE','NAME_FAMILY_STATUS']]
temp_plt_data = temp_plt_data.groupby('NAME_FAMILY_STATUS')['AMT_GOODS_PRICE'].m
temp_plt_data['AMT_GOODS_PRICE'] = temp_plt_data['AMT_GOODS_PRICE'].fillna(temp_
temp_plt_data.head()
```

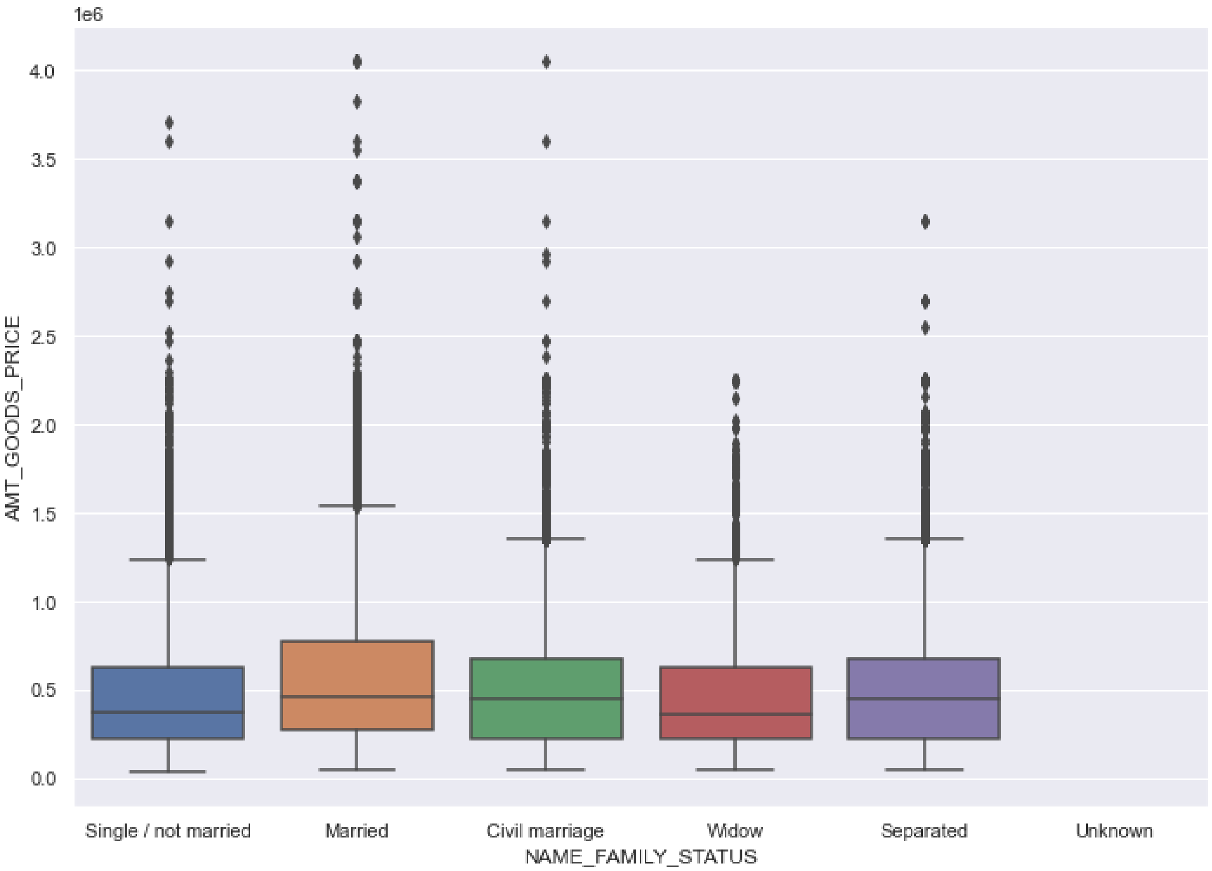
Out[13]:

	NAME_FAMILY_STATUS	AMT_GOODS_PRICE
0	Civil marriage	450000.0
1	Married	459000.0
2	Separated	450000.0
3	Single / not married	373500.0
4	Unknown	360000.0

In [14]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
ax = sns.barplot(x="NAME_FAMILY_STATUS", y="AMT_GOODS_PRICE", data=temp_plt_data
```



```
In [15]: sns.set(rc={'figure.figsize':(11.7,8.27)})
         ax = sns.boxplot(x="NAME_FAMILY_STATUS", y="AMT_GOODS_PRICE", data=X_feature)
```



In [16]:
```
def fill_category_value(a):
    if a['AMT_GOODS_PRICE'] == np.inf:
        return temp_plt_data[temp_plt_data['NAME_FAMILY_STATUS']==a['NAME_FAMILY_
    else:
        return a['AMT_GOODS_PRICE']

for col in temp_col_reqd:
    X_feature['AMT_GOODS_PRICE'] = X_feature['AMT_GOODS_PRICE'].fillna(np.inf)
    if 'AMT_GOODS_PRICE' in col:
        print("columns to be filled with category median is: {}".format(col))
        X_feature['AMT_GOODS_PRICE'] = X_feature.apply(lambda a: fill_category_va
```

```
columns to be filled with category median is: AMT_GOODS_PRICE
```

Dropping one single row with column DAYS_LAST_PHONE_CHANGE as null

In [17]:
```
X_feature.dropna(subset=['DAYS_LAST_PHONE_CHANGE'], inplace=True)
```

Dropping 12 rows with column AMT_ANNUITY as null

In [18]:
```
X_feature.dropna(subset=['AMT_ANNUITY'], inplace=True)
```

In [19]:
```
X_feature = X_feature.reset_index(drop=True)
```

Checking highest correlated features with External Source to replace the null values with

Check for EXT_SOURCE_2

In [20]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

temp_corr_df = pd.DataFrame()

for col in X_feature.columns.tolist():
    if X_feature[col].dtype == 'int':
        l = [col, X_feature['EXT_SOURCE_2'].corr(X_feature[col])]
    else:
        l = [col, X_feature['EXT_SOURCE_2'].corr(pd.DataFrame(LabelEncoder().f
    temp_corr_df = temp_corr_df.append(pd.Series(l),ignore_index=True)
temp_corr_df = temp_corr_df.rename(columns={0:'col_name',1:'correlation_with_E
temp_corr_df['correlation_with_EXT_2'] = abs(temp_corr_df['correlation_with_EX
temp_corr_df.sort_values(by='correlation_with_EXT_2',ascending=False).head(6).
```

Out[20]:

	col_name	correlation_with_EXT_2
27	REGION_RATING_CLIENT	0.292903
28	REGION_RATING_CLIENT_W_CITY	0.288306
48	DAYS_LAST_PHONE_CHANGE	0.195766
5	AMT_INCOME_TOTAL	0.170547
75	TARGET	0.160471

REGION_RATING_CLIENT : Our rating of the region where our client lives

In [21]:

```
region_rating_grouped = X_feature.groupby('REGION_RATING_CLIENT')['EXT_SOURCE_2'


def fill_external_source2(a):
    if a['EXT_SOURCE_2'] == np.inf:
        return region_rating_grouped[region_rating_grouped['REGION_RATING_CLIENT
    else:
        return a['EXT_SOURCE_2']

X_feature['EXT_SOURCE_2'] = X_feature['EXT_SOURCE_2'].fillna(np.inf)
X_feature['EXT_SOURCE_2'] = X_feature.apply(lambda a: fill_external_source2(a),a
```

Check for EXT_SOURCE_3

In [22]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

temp_corr_df = pd.DataFrame()

for col in X_feature.columns.tolist():
    if X_feature[col].dtype == 'int':
        l = [col, X_feature['EXT_SOURCE_3'].corr(X_feature[col])]
    else:
        l = [col, X_feature['EXT_SOURCE_3'].corr(pd.DataFrame(LabelEncoder().fit_
    temp_corr_df = temp_corr_df.append(pd.Series(l),ignore_index=True)
temp_corr_df = temp_corr_df.rename(columns={0:'col_name',1:'correlation_with_EXT_
temp_corr_df['correlation_with_EXT_3'] = abs(temp_corr_df['correlation_with_EXT_
temp_corr_df = temp_corr_df.sort_values(by='correlation_with_EXT_3',ascending=Fa
temp_corr_df
```

Out[22]:

	col_name	correlation_with_EXT_3
15	DAYS_BIRTH	0.205474
75	TARGET	0.178929
18	DAYS_ID_PUBLISH	0.131598
20	FLAG_EMP_PHONE	0.115284
16	DAYS_EMPLOYED	0.113426
38	EXT_SOURCE_2	0.109728
17	DAYS_REGISTRATION	0.107570
5	AMT_INCOME_TOTAL	0.088906
37	ORGANIZATION_TYPE	0.087994

In [23]:

```
ext_source_data = X_feature[temp_corr_df['col_name'].tolist()+['EXT_SOURCE_3']]

for col in ext_source_data.columns.tolist():
    if col != 'EXT_SOURCE_3':
        ext_source_data[col] = LabelEncoder().fit_transform(X_feature[[col]])

ext_source3_train = ext_source_data[ext_source_data['EXT_SOURCE_3'].notnull()]
ext_source3_test = ext_source_data[ext_source_data['EXT_SOURCE_3'].isnull()]
ext_source3_train.shape, ext_source3_test.shape
```

Out[23]:

```
((246535, 10), (60963, 10))
```

In [24]:

```
exs3_y_train = ext_source3_train[['EXT_SOURCE_3']]
exs3_X_train = ext_source3_train.drop(columns=['EXT_SOURCE_3'])

exs3_X_test = ext_source3_test.drop(columns=['EXT_SOURCE_3'])
```

In [25]:
```
from sklearn.linear_model import LinearRegression

model = LinearRegression().fit(exs3_X_train, exs3_y_train)
y_pred_exs3 = model.predict(exs3_X_test)

exs3_output = exs3_X_test
exs3_output['exs3_y'] = y_pred_exs3
exs3_output
```

Out[25]:

	DAYS_BIRTH	TARGET	DAYS_ID_PUBLISH	FLAG_EMP_PHONE	DAYS_EMPLOYED	EXT_SC
1	8382	0	5876	1	11384	
3	6142	0	3730	1	9533	
4	5215	0	2709	1	9534	
9	10676	0	2175	1	10553	
14	10562	0	4111	1	12369	
...	...	...	...	...	...	
307471	12298	0	6132	1	12244	
307488	12184	0	2387	1	11526	
307491	8442	0	5908	1	5318	
307493	15818	0	4185	1	12336	
307494	4372	0	2077	0	12573	

60963 rows × 10 columns

In [26]:
```
exs3_output = exs3_output.reset_index().rename(columns={'index':'index_to_be_upda
for i in exs3_output['index_to_be_updated'].tolist():
    X_feature['EXT_SOURCE_3'].iloc[i] = exs3_output[exs3_output['index_to_be_upda
```

Checking the null values in the Train dataset

In [27]:
```
new_null_data = X_feature.isna().sum().reset_index().rename(columns={'index':'co
new_null_data['count_%'] = new_null_data['null_count']/len(X_feature)*100
new_null_data = new_null_data[new_null_data['count_%'] <= 30]
new_null_data['col_type'] = new_null_data['col_name'].apply(lambda x: X_feature[
new_null_data[new_null_data['count_%'] > 0]
```

Out[27]:

	col_name	null_count	count_%	col_type

In [28]:
```
X_feature.isna().sum()
```

Out[28]:
```
NAME_CONTRACT_TYPE            0
CODE_GENDER                   0
FLAG_OWN_CAR                  0
FLAG_OWN_REALTY               0
CNT_CHILDREN                  0
                             ..
AMT_REQ_CREDIT_BUREAU_WEEK    0
AMT_REQ_CREDIT_BUREAU_MON     0
AMT_REQ_CREDIT_BUREAU_QRT     0
AMT_REQ_CREDIT_BUREAU_YEAR    0
TARGET                        0
Length: 76, dtype: int64
```

Training and testing with the selected columns

Adding Additional relevant features felt

In [29]:
```
X_feature['AMT_CREDIT_TO_ANNUITY_RATIO'] = X_feature['AMT_CREDIT'] / X_feature['
X_feature['Tot_EXTERNAL_SOURCE'] = X_feature['EXT_SOURCE_2'] + X_feature['EXT_SOU
X_feature['Salary_to_credit'] = X_feature['AMT_INCOME_TOTAL']/X_feature['AMT_CRE
X_feature['Annuity_to_salary_ratio'] = X_feature['AMT_ANNUITY']/X_feature['AMT_I
```

In [30]:
```
y = X_feature[['TARGET']]
X = X_feature.drop(['TARGET'], axis = 1)
```

In [31]:
```
le_dict, full_pipeline_with_predictor, experimentLog = returnModel(X,y,experimen
```

```
train data set:
(196798, 79) (196798, 1)
test data set:
(61500, 79) (61500, 1)
validation data set:
(49200, 79) (49200, 1)
AUC is 0.5055202479077585

............

Confusion Matrix: [[56482    60]
 [ 4898    60]]
```

In [32]:

```
experimentLog
```

Out[32]:

	ExpID	Cross fold train accuracy	Test Accuracy	Validation Accuracy	AUC	Train Time(s)	Test Time(s)	Experiment description
0	Baseline with 120 inputs	92.0	92.0	91.8	0.504333	110.2988	0.0994	All features Dataset - Baseline LogisticRegres...
1	Baseline with 79 inputs	91.9	91.9	91.8	0.505520	96.4481	0.0506	Selected features Dataset - Baseline LogisticR...

Hyperparameter tuning(Grid Search)

In [33]:

```
train = X_feature
train.shape
```

Out[33]:

```
(307498, 80)
```

In [34]:

```
features = train.columns.tolist()
features.remove('TARGET')
len(features)
```

Out[34]:

```
79
```

In [35]:

```
le_dict = {}

for col in features:
    if train[col].dtype == 'object':
        le = LabelEncoder()
        train[col] = le.fit_transform(train[col])
        le_dict['le_{}'.format(col)] = le
```

In [36]:

```
X = train[features]
y = train['TARGET']
```

In [37]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_siz
```

Decision Making Tree

In [38]:

```
DMT_pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('regressor', DecisionTreeRegressor(random_state=100))
])

param_grid = [{
    'regressor__max_depth': [2, 3],
    'regressor__min_samples_split': [2, 3],
}]

start = time()
dmt_search = GridSearchCV(estimator=DMT_pipe, param_grid=param_grid, cv=2)
dmt_search.fit(X_train, y_train)
train_time = np.round(time() - start, 4)

trainAcc  = dmt_search.score(X_train, y_train)
validAcc  = dmt_search.score(X_valid, y_valid)
start = time()
testAcc   = dmt_search.score(X_test, y_test)
test_time = np.round(time() - start, 4)

number_of_inputs = X_train.shape[1]
AUC = roc_auc_score(y_test,dmt_search.predict(X_test))

try: experimentLog
except : experimentLog = pd.DataFrame(columns=["ExpID", "Cross fold train accura
                                               "AUC", "Train Time(s)", "Test Tim
experimentLog.loc[len(experimentLog)] =[f"Gridsearch Decision Making Tree with {
                                        f"{trainAcc*100:8.2f}%", f"{testAcc*100:
                                        train_time, test_time,
                                        "Decision Making Tree GridSearch with se
```

In [39]:

```
experimentLog
```

Out[39]:

	ExpID	Cross fold train accuracy	Test Accuracy	Validation Accuracy	AUC	Train Time(s)	Test Time(s)	Experiment description
0	Baseline with 120 inputs	92.0	92.0	91.8	0.504333	110.2988	0.0994	All features Dataset - Baseline LogisticRegres...
1	Baseline with 79 inputs	91.9	91.9	91.8	0.505520	96.4481	0.0506	Selected features Dataset - Baseline LogisticR...
2	Gridsearch Decision Making Tree with 79 inputs	7.84%	7.56%	7.35%	0.738725	4.4617	0.0129	Decision Making Tree GridSearch with selected ...

Lasso Regression

In [40]:

```
lasso_pipeline = Pipeline([
                     ('scaler',StandardScaler()),
                     ('model',Lasso())
])


lasso_pipeline = GridSearchCV(lasso_pipeline,
                      {'model__alpha':[0.0001,0.001,0.01,0.1,0.2,0.3,0.4,0.5,0.6,
                      cv = 5, scoring="neg_mean_squared_error",verbose=3
                      )

start = time()
lasso_pipeline.fit(X_train, y_train)
train_time = np.round(time() - start, 4)

trainAcc  = lasso_pipeline.score(X_train, y_train)
validAcc  = lasso_pipeline.score(X_valid, y_valid)
start = time()
testAcc  = lasso_pipeline.score(X_test, y_test)
test_time = np.round(time() - start, 4)

AUC = roc_auc_score(y_test,lasso_pipeline.predict(X_test))
number_of_inputs = X_train.shape[1]

#del experimentLog
try: experimentLog
except : experimentLog = pd.DataFrame(columns=["ExpID", "Cross fold train accura
                                               "AUC", "Train Time(s)", "Test Tim
experimentLog.loc[len(experimentLog)] =[f"Lasso Reg with {number_of_inputs} input
                                        f"{trainAcc*100:8.2f}%", f"{testAcc*100::
                                        train_time, test_time,
                                        "Lasso Regression for feature selection"
```

```
Fitting 5 folds for each of 22 candidates, totalling 110 fits
[CV 1/5] END ..............model__alpha=0.0001;, score=-0.070 total time=  1
5.2s
[CV 2/5] END ..............model__alpha=0.0001;, score=-0.070 total time=  1
5.5s
[CV 3/5] END ..............model__alpha=0.0001;, score=-0.067 total time=  1
5.7s
[CV 4/5] END ..............model__alpha=0.0001;, score=-0.069 total time=  1
5.5s
[CV 5/5] END ..............model__alpha=0.0001;, score=-0.069 total time=  1
5.3s
[CV 1/5] END ...............model__alpha=0.001;, score=-0.070 total time=
2.1s
[CV 2/5] END ...............model__alpha=0.001;, score=-0.070 total time=
2.2s
[CV 3/5] END ...............model__alpha=0.001;, score=-0.067 total time=
2.2s
[CV 4/5] END ...............model__alpha=0.001;, score=-0.070 total time=
2.2s
[CV 5/5] END ...............model__alpha=0.001;, score=-0.069 total time=
In [41]: experimentLog
```

Out[41]:

	ExpID	Cross fold train accuracy	Test Accuracy	Validation Accuracy	AUC	Train Time(s)	Test Time(s)	Experiment description
0	Baseline with 120 inputs	92.0	92.0	91.8	0.504333	110.2988	0.0994	All features Dataset - Baseline LogisticRegres...
1	Baseline with 79 inputs	91.9	91.9	91.8	0.505520	96.4481	0.0506	Selected features Dataset - Baseline LogisticR...
2	Gridsearch Decision Making Tree with 79 inputs	7.84%	7.56%	7.35%	0.738725	4.4617	0.0129	Decision Making Tree GridSearch with selected ...
3	Lasso Reg with 79 inputs	-6.89%	-6.87%	-6.89%	0.755827	131.6072	0.0224	Lasso Regression for feature selection

```
In [42]: print(lasso_pipeline.best_params_)
         coefficients = lasso_pipeline.best_estimator_.named_steps['model'].coef_
         importance = np.abs(coefficients)
         len(np.array(features)[importance > 0])
```

```
{'model__alpha': 0.0001}
```

```
Out[42]: 74
```

Ridge Regression

In [43]:

```
ridge_pipeline = Pipeline([
                     ('scaler',StandardScaler()),
                     ('model',Ridge())
])


ridge_pipeline = GridSearchCV(ridge_pipeline,
                      {'model__alpha':[0.0001,0.001,0.01,0.1,0.2,0.3,0.4,0.5,0.6,
                      cv = 5, scoring="neg_mean_squared_error",verbose=3
                      )

start = time()
ridge_pipeline.fit(X_train, y_train)
train_time = np.round(time() - start, 4)

trainAcc  = ridge_pipeline.score(X_train, y_train)
validAcc  = ridge_pipeline.score(X_valid, y_valid)
start = time()
testAcc  = ridge_pipeline.score(X_test, y_test)
test_time = np.round(time() - start, 4)

AUC = roc_auc_score(y_test,ridge_pipeline.predict(X_test))
number_of_inputs = X_train.shape[1]

try: experimentLog
except : experimentLog = pd.DataFrame(columns=["ExpID", "Cross fold train accura
                                          "AUC", "Train Time(s)", "Test Tim
experimentLog.loc[len(experimentLog)] =[f"Ridge Reg with {number_of_inputs} inpu
                                        f"{trainAcc*100:8.2f}%", f"{testAcc*100:
                                        train_time, test_time,
                                        "Ridge Regression for feature selection"
```

```
Fitting 5 folds for each of 22 candidates, totalling 110 fits
[CV 1/5] END .............model__alpha=0.0001;, score=-0.070 total time=
0.3s
[CV 2/5] END .............model__alpha=0.0001;, score=-0.070 total time=
0.3s
[CV 3/5] END .............model__alpha=0.0001;, score=-0.067 total time=
0.3s
[CV 4/5] END .............model__alpha=0.0001;, score=-0.069 total time=
0.3s
[CV 5/5] END .............model__alpha=0.0001;, score=-0.069 total time=
0.3s
[CV 1/5] END ..............model__alpha=0.001;, score=-0.070 total time=
0.3s
[CV 2/5] END ..............model__alpha=0.001;, score=-0.070 total time=
0.3s
[CV 3/5] END ..............model__alpha=0.001;, score=-0.067 total time=
0.2s
[CV 4/5] END ..............model__alpha=0.001;, score=-0.069 total time=
0.3s
[CV 5/5] END ..............model__alpha=0.001;, score=-0.069 total time=
```

In [44]:

```
experimentLog
```

Out[44]:

	ExpID	Cross fold train accuracy	Test Accuracy	Validation Accuracy	AUC	Train Time(s)	Test Time(s)	Experiment description
0	Baseline with 120 inputs	92.0	92.0	91.8	0.504333	110.2988	0.0994	All features Dataset - Baseline LogisticRegres...
1	Baseline with 79 inputs	91.9	91.9	91.8	0.505520	96.4481	0.0506	Selected features Dataset - Baseline LogisticR...
2	Gridsearch Decision Making Tree with 79 inputs	7.84%	7.56%	7.35%	0.738725	4.4617	0.0129	Decision Making Tree GridSearch with selected ...
3	Lasso Reg with 79 inputs	-6.89%	-6.87%	-6.89%	0.755827	131.6072	0.0224	Lasso Regression for feature selection
4	Ridge Reg with 79 inputs	-6.89%	-6.87%	-6.89%	0.756776	30.4069	0.0134	Ridge Regression for feature selection

In [45]:

```
print(ridge_pipeline.best_params_)
coefficients = ridge_pipeline.best_estimator_.named_steps['model'].coef_
importance = np.abs(coefficients)
len(np.array(features)[importance > 0.005])
```

```
{'model__alpha': 3}
```

Out[45]:

```
22
```

Logistic Regression

In [46]:

```
logistic = LogisticRegression(max_iter=10000, tol=0.1)
clf_pipe = Pipeline(steps=[("logistic", logistic)])
param_grid = {
    "logistic__C": np.logspace(-4, 4, 10)
}


# Time and score test predictions
start = time()

clf_search = GridSearchCV(clf_pipe, param_grid, n_jobs=-1)
clf_search.fit(X_train, y_train)
train_time = np.round(time() - start, 4)

trainAcc  = clf_search.score(X_train, y_train)
validAcc  = clf_search.score(X_valid, y_valid)
start = time()
testAcc  = clf_search.score(X_test, y_test)
test_time = np.round(time() - start, 4)

number_of_inputs = X_train.shape[1]
AUC = roc_auc_score(y_test,clf_search.predict(X_test))

try: experimentLog
except : experimentLog = pd.DataFrame(columns=["ExpID", "Cross fold train accura
                                               "AUC", "Train Time(s)", "Test Tim
experimentLog.loc[len(experimentLog)] =[f"Gridsearch LogReg with {number_of_inpu
                                        f"{trainAcc*100:8.2f}%", f"{testAcc*100:
                                        train_time, test_time,
                                        "LogReg GridSearch with selected feature
```

In [47]:

```
experimentLog
```

Out[47]:

	ExpID	Cross fold train accuracy	Test Accuracy	Validation Accuracy	AUC	Train Time(s)	Test Time(s)	Experiment description
0	Baseline with 120 inputs	92.0	92.0	91.8	0.504333	110.2988	0.0994	All features Dataset - Baseline LogisticRegres...
1	Baseline with 79 inputs	91.9	91.9	91.8	0.505520	96.4481	0.0506	Selected features Dataset - Baseline LogisticR...
2	Gridsearch Decision Making Tree with 79 inputs	7.84%	7.56%	7.35%	0.738725	4.4617	0.0129	Decision Making Tree GridSearch with selected ...
3	Lasso Reg with 79 inputs	-6.89%	-6.87%	-6.89%	0.755827	131.6072	0.0224	Lasso Regression for feature selection
4	Ridge Reg with 79 inputs	-6.89%	-6.87%	-6.89%	0.756776	30.4069	0.0134	Ridge Regression for feature selection
5	Gridsearch LogReg with 79 inputs	91.91%	91.98%	91.96%	0.500000	55.1705	0.0168	LogReg GridSearch with selected features

Xgboost

In [ ]:

```
xgboost_pipe = Pipeline([
    ('standard_scaler', StandardScaler()),
    ('model', xgb.XGBClassifier())
])

param_grid = {
    'model__max_depth': [2, 3, 5, 7, 10],
    'model__n_estimators': [10, 100, 500]
}

xgboost_search = GridSearchCV(xgboost_pipe, param_grid, scoring="roc_auc",cv=5)

# Time and score test predictions
start = time()
xgboost_search.fit(X_train, y_train)
train_time = np.round(time() - start, 4)

trainAcc  = xgboost_search.score(X_train, y_train)
validAcc  = xgboost_search.score(X_valid, y_valid)
start = time()
testAcc   = xgboost_search.score(X_test, y_test)
test_time = np.round(time() - start, 4)

number_of_inputs = X_train.shape[1]
AUC = roc_auc_score(y_test,xgboost_search.predict(X_test))

try: experimentLog
except : experimentLog = pd.DataFrame(columns=["ExpID", "Cross fold train accura
                                               "AUC", "Train Time(s)", "Test Tim
experimentLog.loc[len(experimentLog)] =[f"Gridsearch Xgboost with {number_of_inpu
                                        f"{trainAcc*100:8.2f}%", f"{testAcc*100:
                                        train_time, test_time,
                                        "Xgboost GridSearch with selected featur
```

In [ ]:

```
experimentLog
```