In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
application_record = pd.read_csv('application_record.csv')
credit_record = pd.read_csv('credit_record.csv')
```

In [3]:
```
application_record
```

Out[3]:

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOBIL	FLAG_WORK_PHONE	FLAG_PHONE	FLAG_EMAIL	OCCUPATION_TYPE	CNT_FAM_MEMBERS
0	5008804	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0
1	5008805	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0
2	5008806	M	Y	Y	0	112500.0	Working	Secondary / secondary special	Married	House / apartment	-21474	-1134	1	0	0	0	Security staff	2.0
3	5008808	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	-19110	-3051	1	0	1	1	Sales staff	1.0
4	5008809	F	N	Y	0	270000.0	Commercial associate	Secondary / secondary special	Single / not married	House / apartment	-19110	-3051	1	0	1	1	Sales staff	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
438552	6840104	M	N	Y	0	135000.0	Pensioner	Secondary / secondary special	Separated	House / apartment	-22717	365243	1	0	0	0	NaN	1.0
438553	6840222	F	N	N	0	103500.0	Working	Secondary / secondary special	Single / not married	House / apartment	-15939	-3007	1	0	0	0	Laborers	1.0
438554	6841878	F	N	N	0	54000.0	Commercial associate	Higher education	Single / not married	With parents	-8169	-372	1	1	0	0	Sales staff	1.0
438555	6842765	F	N	Y	0	72000.0	Pensioner	Secondary / secondary special	Married	House / apartment	-21673	365243	1	0	0	0	NaN	2.0
438556	6842885	F	N	Y	0	121500.0	Working	Secondary / secondary special	Married	House / apartment	-18858	-1201	1	0	1	0	Sales staff	2.0

438557 rows × 18 columns

In [4]:
```
credit_record
```

Out[4]:

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C
...	...	...	...
1048570	5150487	-25	C
1048571	5150487	-26	C
1048572	5150487	-27	C
1048573	5150487	-28	C
1048574	5150487	-29	C

1048575 rows × 3 columns

In [5]:
```
credit_record.replace(['X','C'], 0,inplace=True)
```

In [6]:
```
credit_record.STATUS = pd.to_numeric(credit_record.STATUS)
```

In [7]:
```
drop_ls = []
for i in range(len(credit_record)):
    if credit_record.STATUS[i] != 0:
        drop_ls.append(credit_record.ID[i])
```

In [8]:
```
len(drop_ls)
```

Out[8]:
```
14194
```

In [9]:
```
for i in range(len(credit_record)):
        if credit_record.ID[i] in drop_ls:
            credit_record.STATUS[i] = 1
```

In [10]:
```
credit_record.STATUS.value_counts()
```

Out[10]:
```
0    904764
1    143811
Name: STATUS, dtype: int64
```

In [11]:
```
credit_record.drop_duplicates(inplace=True)
credit_record
```

Out[11]:

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	0
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	0
...	...	...	...
1048570	5150487	-25	0
1048571	5150487	-26	0
1048572	5150487	-27	0
1048573	5150487	-28	0
1048574	5150487	-29	0

1048575 rows × 3 columns

In [12]:
```
print(f'No. of IDs in application_record = {len(application_record.ID)} No. of IDs in credit_record = {len(credit_record.ID)}')
```

```
No. of IDs in application_record = 438557 No. of IDs in credit_record = 1048575
```

In [13]:
```
dataset = application_record.merge(credit_record, on=['ID'], how='inner')
```

In [14]:
```
dataset.drop(['ID'],inplace=True,axis=1)
```

In [15]:
```
dataset.duplicated().sum()
```

Out[15]:
```
412393
```

In [16]:
```
dataset.drop_duplicates(inplace=True)
```

In [17]:
```
dataset
```

Out[17]:

	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE	NAME_FAMILY_STATUS	NAME_HOUSING_TYPE	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOBIL	FLAG_WORK_PHONE	FLAG_PHONE	FLAG_EMAIL	OCCUPATION_TYPE	CNT_FAM_MEMBERS	MONTHS_BALANCE	STATUS
0	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0	0	1
1	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0	-1	1
2	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0	-2	1
3	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0	-3	1
4	M	Y	Y	0	427500.0	Working	Higher education	Civil marriage	Rented apartment	-12005	-4542	1	1	0	0	NaN	2.0	-4	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
777710	M	N	Y	0	112500.0	Working	Secondary / secondary special	Single / not married	Rented apartment	-9188	-1193	1	0	0	0	Laborers	1.0	-9	1
777711	M	N	Y	0	112500.0	Working	Secondary / secondary special	Single / not married	Rented apartment	-9188	-1193	1	0	0	0	Laborers	1.0	-10	1
777712	M	N	Y	0	112500.0	Working	Secondary / secondary special	Single / not married	Rented apartment	-9188	-1193	1	0	0	0	Laborers	1.0	-11	1
777713	M	N	Y	0	112500.0	Working	Secondary / secondary special	Single / not married	Rented apartment	-9188	-1193	1	0	0	0	Laborers	1.0	-12	1
777714	M	N	Y	0	112500.0	Working	Secondary / secondary special	Single / not married	Rented apartment	-9188	-1193	1	0	0	0	Laborers	1.0	-13	1

365322 rows × 19 columns

In [18]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 365322 entries, 0 to 777714
Data columns (total 19 columns):
 #   Column               Non-Null Count   Dtype  
---  ------               --------------   -----  
 0   CODE_GENDER          365322 non-null  object 
 1   FLAG_OWN_CAR         365322 non-null  object 
 2   FLAG_OWN_REALTY      365322 non-null  object 
 3   CNT_CHILDREN         365322 non-null  int64  
 4   AMT_INCOME_TOTAL     365322 non-null  float64
 5   NAME_INCOME_TYPE     365322 non-null  object 
 6   NAME_EDUCATION_TYPE  365322 non-null  object 
 7   NAME_FAMILY_STATUS   365322 non-null  object 
 8   NAME_HOUSING_TYPE    365322 non-null  object 
 9   DAYS_BIRTH           365322 non-null  int64  
 10  DAYS_EMPLOYED        365322 non-null  int64  
 11  FLAG_MOBIL           365322 non-null  int64  
 12  FLAG_WORK_PHONE      365322 non-null  int64  
 13  FLAG_PHONE           365322 non-null  int64  
 14  FLAG_EMAIL           365322 non-null  int64  
 15  OCCUPATION_TYPE      252192 non-null  object 
 16  CNT_FAM_MEMBERS      365322 non-null  float64
 17  MONTHS_BALANCE       365322 non-null  int64  
 18  STATUS               365322 non-null  int64  
dtypes: float64(2), int64(9), object(8)
memory usage: 55.7+ MB
```

In [19]:
```
dataset.describe()
```

Out[19]:

	CNT_CHILDREN	AMT_INCOME_TOTAL	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOBIL	FLAG_WORK_PHONE	FLAG_PHONE	FLAG_EMAIL	CNT_FAM_MEMBERS	MONTHS_BALANCE	STATUS
count	365322.000000	3.653220e+05	365322.000000	365322.000000	365322.0	365322.000000	365322.000000	365322.000000	365322.00000	365322.000000	365322.000000
mean	0.425742	1.848982e+05	-16161.482656	60776.306365	1.0	0.221878	0.294214	0.089595	2.19825	-21.695310	0.203226
std	0.768540	1.017316e+05	4144.182785	139028.719425	0.0	0.415510	0.455689	0.285601	0.92849	15.016078	0.402400
min	0.000000	2.700000e+04	-25152.000000	-15713.000000	1.0	0.000000	0.000000	0.000000	1.00000	-60.000000	0.000000
25%	0.000000	1.170000e+05	-19614.000000	-3208.000000	1.0	0.000000	0.000000	0.000000	2.00000	-33.000000	0.000000
50%	0.000000	1.575000e+05	-15849.000000	-1566.000000	1.0	0.000000	0.000000	0.000000	2.00000	-20.000000	0.000000
75%	1.000000	2.250000e+05	-12676.000000	-378.000000	1.0	0.000000	1.000000	0.000000	3.00000	-9.000000	0.000000
max	19.000000	1.575000e+06	-7489.000000	365243.000000	1.0	1.000000	1.000000	1.000000	20.00000	0.000000	1.000000

In [20]:
```
dataset.isna().sum()
```

Out[20]:
```
CODE_GENDER               0
FLAG_OWN_CAR              0
FLAG_OWN_REALTY           0
CNT_CHILDREN              0
AMT_INCOME_TOTAL          0
NAME_INCOME_TYPE          0
NAME_EDUCATION_TYPE       0
NAME_FAMILY_STATUS        0
NAME_HOUSING_TYPE         0
DAYS_BIRTH                0
DAYS_EMPLOYED             0
FLAG_MOBIL                0
FLAG_WORK_PHONE           0
FLAG_PHONE                0
FLAG_EMAIL                0
OCCUPATION_TYPE      113130
CNT_FAM_MEMBERS           0
MONTHS_BALANCE            0
STATUS                    0
dtype: int64
```

In [21]:
```
dataset.isna().sum().sum()
```

Out[21]:
```
113130
```

In [22]:
```
dataset.OCCUPATION_TYPE
```

Out[22]:
```
0             NaN
1             NaN
2             NaN
3             NaN
4             NaN
            ...
777710    Laborers
777711    Laborers
777712    Laborers
777713    Laborers
777714    Laborers
Name: OCCUPATION_TYPE, Length: 365322, dtype: object
```

In [23]:
```
dataset.OCCUPATION_TYPE.value_counts()
```

Out[23]:
```
Laborers                62839
Core staff              34175
Sales staff             33786
Managers                31066
Drivers                 23349
High skill tech staff   14459
Medicine staff          11937
Accountants             11926
Security staff           6851
Cooking staff            6663
Cleaning staff           5201
Private service staff    2989
Low-skill Laborers       2000
Secretaries              1523
Waiters/barmen staff     1272
HR staff                  973
IT staff                  617
Realty agents             566
Name: OCCUPATION_TYPE, dtype: int64
```

In [24]:
```
dataset.OCCUPATION_TYPE.replace(np.nan, 'Other', inplace = True)
```

In [25]:
```
dataset.OCCUPATION_TYPE.value_counts()
```

Out[25]:
```
Other                  113130
Laborers                62839
Core staff              34175
Sales staff             33786
Managers                31066
Drivers                 23349
High skill tech staff   14459
Medicine staff          11937
Accountants             11926
Security staff           6851
Cooking staff            6663
Cleaning staff           5201
Private service staff    2989
Low-skill Laborers       2000
Secretaries              1523
Waiters/barmen staff     1272
HR staff                  973
IT staff                  617
Realty agents             566
Name: OCCUPATION_TYPE, dtype: int64
```

In [26]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [27]:
```
for col in dataset.columns:
    if dataset[col].dtype == 'object':
        dataset[col] = le.fit_transform(dataset[col])
```

In [28]:
```
dataset.drop_duplicates(inplace=True)
```

In [29]:
```
X = dataset.iloc[:, :-1]
y = dataset.iloc[:,-1]
```

In [30]:
```
from sklearn.model_selection import KFold,StratifiedKFold,train_test_split
kfold = StratifiedKFold(n_splits=8, shuffle=True, random_state=0)
```

In [31]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [32]:
```
for train_index, test_index in kfold.split(X,y):

    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
```

In [33]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(class_weight='balanced',max_depth=48,splitter='best',random_state=42,min_samples_split=48)
classifier.fit(X_train, y_train)
```

Out[33]:
```
DecisionTreeClassifier(class_weight='balanced', max_depth=48,
                       min_samples_split=48, random_state=42)
```

In [34]:
```
y_pred = classifier.predict(X_test)
```

In [35]:
```
y_pred_train= classifier.predict(X_train)
```

In [36]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
acc = accuracy_score(y_test,y_pred)
confusion_mat = confusion_matrix(y_test,y_pred)
pre_score = precision_score(y_test,y_pred)
recall = recall_score(y_test,y_pred)
f1 = f1_score(y_test,y_pred)
specificity_test = confusion_mat[0,0] / (confusion_mat[0,0] + confusion_mat[0,1])
```

In [37]:
```
print(f'Accuracy Score = {acc}\n Confusion Matrix = {confusion_mat}\n Precision Score = {pre_score}\n Recall Score = {recall}\n F1 Score = {f1}\n Specificity Test = {specificity_test}')
```

```
Accuracy Score = 0.8631336910106209
 Confusion Matrix = [[30484  5901]
 [  349  8931]]
 Precision Score = 0.6021440129449838
 Recall Score = 0.9623922413793103
 F1 Score = 0.7407929661579297
 Specificity Test = 0.8378177820530438
```

In [38]:
```
acc_train = accuracy_score(y_train,y_pred_train)
confusion_mat_train = confusion_matrix(y_train,y_pred_train)
pre_score_train = precision_score(y_train,y_pred_train)
recall_train = recall_score(y_train,y_pred_train)
f1_train = f1_score(y_train,y_pred_train)
specificity_train = confusion_mat_train[0,0] / (confusion_mat_train[0,0] + confusion_mat_train[0,1])
```

In [39]:
```
print(f'Accuracy Score = {acc_train}\n Confusion Matrix = {confusion_mat_train}\n Precision Score = {pre_score_train}\n Recall Score = {recall_train}\n F1 Score = {f1_train}\n Specificity Test = {specificity_train}')
```

```
Accuracy Score = 0.871684336648345
 Confusion Matrix = [[214176  40518]
 [   499  64464]]
 Precision Score = 0.6140481225352917
 Recall Score = 0.9923187044933269
 F1 Score = 0.7586454441142721
 Specificity Test = 0.8409149803293364
```

In [ ]:
```

```