In [116…

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score,f1_score,precision_score, recall_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import RandomOverSampler,SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler
from collections import Counter
import warnings
warnings.filterwarnings("ignore")
```

In [90]:

```
dataset=pd.read_csv('./capstone_application_data.csv')
dataset.head()
```

Out[90]:

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TC
0	100002	1	Cash loans	M	N	Y	0	202
1	100003	0	Cash loans	F	N	N	0	270
2	100004	0	Revolving loans	M	Y	Y	0	67
3	100006	0	Cash loans	F	N	Y	0	135
4	100007	0	Cash loans	M	N	Y	0	121

5 rows × 122 columns

In [91]:

```
dataset.shape
```

Out[91]:

```
(307511, 122)
```

In [92]:

```
dataset.info()
```

```
<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
```

In [93]:

```
dataset.drop(columns = ['SK_ID_CURR',
                        '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_MOBIL',
                           'FLAG_EMP_PHONE',
                           'FLAG_WORK_PHONE',
                           'FLAG_CONT_MOBILE',
                           'FLAG_PHONE',
                            'FLAG_EMAIL',
                       'NAME_TYPE_SUITE',
                        'REGION_POPULATION_RELATIVE',
                        'DAYS_REGISTRATION',
                        'DAYS_ID_PUBLISH',
                        'OWN_CAR_AGE',
                        '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',
                          'FONDKAPREMONT_MODE',
                          'HOUSETYPE_MODE',
                          'TOTALAREA_MODE',
                          'WALLSMATERIAL_MODE',
                          'EMERGENCYSTATE_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',
                          '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',
                        'WEEKDAY_APPR_PROCESS_START',
                          'HOUR_APPR_PROCESS_START',
                         'EXT_SOURCE_1',
                         'EXT_SOURCE_2',
                         'EXT_SOURCE_3'],inplace=True)
```

In [94]:
```
dataset.shape
```

Out[94]:
```
(307511, 24)
```

In [95]:
```
dataset.head()
```

Out[95]:

	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CR
0	1	Cash loans	M	N	Y	0	202500.0	406
1	0	Cash loans	F	N	N	0	270000.0	1293
2	0	Revolving loans	M	Y	Y	0	67500.0	135
3	0	Cash loans	F	N	Y	0	135000.0	312
4	0	Cash loans	M	N	Y	0	121500.0	513

5 rows × 24 columns

In [96]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Data columns (total 24 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   TARGET                       307511 non-null  int64  
 1   NAME_CONTRACT_TYPE           307511 non-null  object 
 2   CODE_GENDER                  307511 non-null  object 
 3   FLAG_OWN_CAR                 307511 non-null  object 
 4   FLAG_OWN_REALTY              307511 non-null  object 
 5   CNT_CHILDREN                 307511 non-null  int64  
 6   AMT_INCOME_TOTAL             307511 non-null  float64
 7   AMT_CREDIT                   307511 non-null  float64
 8   AMT_ANNUITY                  307499 non-null  float64
 9   AMT_GOODS_PRICE              307233 non-null  float64
 10  NAME_INCOME_TYPE             307511 non-null  object 
 11  NAME_EDUCATION_TYPE          307511 non-null  object 
 12  NAME_FAMILY_STATUS           307511 non-null  object 
 13  NAME_HOUSING_TYPE            307511 non-null  object 
 14  DAYS_BIRTH                   307511 non-null  int64  
 15  DAYS_EMPLOYED                307511 non-null  int64  
 16  OCCUPATION_TYPE              211120 non-null  object 
 17  CNT_FAM_MEMBERS              307509 non-null  float64
 18  REGION_RATING_CLIENT         307511 non-null  int64  
 19  REGION_RATING_CLIENT_W_CITY  307511 non-null  int64  
 20  ORGANIZATION_TYPE            307511 non-null  object 
 21  APARTMENTS_AVG               151450 non-null  float64
 22  BASEMENTAREA_AVG             127568 non-null  float64
 23  YEARS_BEGINEXPLUATATION_AVG  157504 non-null  float64
dtypes: float64(8), int64(6), object(10)
memory usage: 56.3+ MB
```

In [100…
```
#Percentage of null values in each column
col_with_nullvalues=dataset.isnull().sum()/len(dataset)*100
col_with_nullvalues.sort_values(ascending=False)
```

Out[100…
```
BASEMENTAREA_AVG               58.515956
APARTMENTS_AVG                 50.749729
YEARS_BEGINEXPLUATATION_AVG    48.781019
OCCUPATION_TYPE                31.345545
AMT_GOODS_PRICE                 0.090403
AMT_ANNUITY                     0.003902
CNT_FAM_MEMBERS                 0.000650
NAME_HOUSING_TYPE               0.000000
ORGANIZATION_TYPE               0.000000
REGION_RATING_CLIENT_W_CITY     0.000000
REGION_RATING_CLIENT            0.000000
DAYS_EMPLOYED                   0.000000
DAYS_BIRTH                      0.000000
TARGET                          0.000000
NAME_CONTRACT_TYPE              0.000000
NAME_EDUCATION_TYPE             0.000000
NAME_INCOME_TYPE                0.000000
AMT_CREDIT                      0.000000
AMT_INCOME_TOTAL                0.000000
CNT_CHILDREN                    0.000000
FLAG_OWN_REALTY                 0.000000
FLAG_OWN_CAR                    0.000000
CODE_GENDER                     0.000000
NAME_FAMILY_STATUS              0.000000
dtype: float64
```

In [101…
```
# List the columns having more than 50% missing values
nullvalues_50=col_with_nullvalues[col_with_nullvalues.values>50.0].sort_values(ascending=False)
nullvalues_50
```

Out[101…
```
BASEMENTAREA_AVG    58.515956
APARTMENTS_AVG      50.749729
dtype: float64
```

In [102…
```
#drop the columns with 50 % or more null values
col_with_nullvalues = list(col_with_nullvalues[col_with_nullvalues.values>=50.0].index)
dataset.drop(labels=col_with_nullvalues,axis=1,inplace=True)
```

```
print(len(col_with_nullvalues))
```

```
2
```

In [103…

```
def reduce_mem_usage(df):
    start_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))

    for col in df.columns:
        col_type = df[col].dtype

        if col_type != object:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)
        else:
            df[col] = df[col].astype('object')

    end_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage after optimization is: {:.2f} MB'.format(end_mem))
    print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))

    return df
dataset = reduce_mem_usage(dataset)
```

```
Memory usage of dataframe is 32.26 MB
Memory usage after optimization is: 32.26 MB
Decreased by 0.0%
```

In [104…

```
#convert all datatypes to integer
le = LabelEncoder()
dataset['NAME_CONTRACT_TYPE'] = le.fit_transform(dataset['NAME_CONTRACT_TYPE'])
dataset['CODE_GENDER'] = le.fit_transform(dataset['CODE_GENDER'])
dataset['FLAG_OWN_CAR'] = le.fit_transform(dataset['FLAG_OWN_CAR'])
dataset['FLAG_OWN_REALTY'] = le.fit_transform(dataset['FLAG_OWN_REALTY'])
dataset['NAME_INCOME_TYPE'] = le.fit_transform(dataset['NAME_INCOME_TYPE'])
dataset['NAME_EDUCATION_TYPE'] = le.fit_transform(dataset['NAME_EDUCATION_TYPE'])
dataset['NAME_FAMILY_STATUS'] = le.fit_transform(dataset['NAME_FAMILY_STATUS'])
dataset['NAME_HOUSING_TYPE'] = le.fit_transform(dataset['NAME_HOUSING_TYPE'])
dataset['OCCUPATION_TYPE'] = le.fit_transform(dataset['OCCUPATION_TYPE'].astype(str))
dataset['ORGANIZATION_TYPE'] = le.fit_transform(dataset['ORGANIZATION_TYPE'])
dataset['CNT_FAM_MEMBERS'] = le.fit_transform(dataset['CNT_FAM_MEMBERS'])
```

In [105…

```
#Label Encoding

le = LabelEncoder()
dataset['NAME_CONTRACT_TYPE'] = le.fit_transform(dataset['NAME_CONTRACT_TYPE'])
dataset['CODE_GENDER'] = le.fit_transform(dataset['CODE_GENDER'])
dataset['FLAG_OWN_CAR'] = le.fit_transform(dataset['FLAG_OWN_CAR'])
dataset['FLAG_OWN_REALTY'] = le.fit_transform(dataset['FLAG_OWN_REALTY'])
dataset['NAME_INCOME_TYPE'] = le.fit_transform(dataset['NAME_INCOME_TYPE'])
dataset['NAME_EDUCATION_TYPE'] = le.fit_transform(dataset['NAME_EDUCATION_TYPE'])
dataset['NAME_FAMILY_STATUS'] = le.fit_transform(dataset['NAME_FAMILY_STATUS'])
dataset['NAME_HOUSING_TYPE'] = le.fit_transform(dataset['NAME_HOUSING_TYPE'])
dataset['OCCUPATION_TYPE'] = le.fit_transform(dataset['OCCUPATION_TYPE'].astype(str))
dataset['ORGANIZATION_TYPE'] = le.fit_transform(dataset['ORGANIZATION_TYPE'])
dataset['YEARS_BEGINEXPLUATATION_AVG'] = le.fit_transform(dataset['YEARS_BEGINEXPLUATATION_AVG'])
dataset['CNT_FAM_MEMBERS'] = le.fit_transform(dataset['CNT_FAM_MEMBERS'])
```

In [106…

```
dataset.isnull().sum().sort_values(ascending=False)
```

Out[106…

```
AMT_GOODS_PRICE               278
AMT_ANNUITY                    12
TARGET                          0
```

```
NAME_FAMILY_STATUS               0
ORGANIZATION_TYPE                0
REGION_RATING_CLIENT_W_CITY      0
REGION_RATING_CLIENT             0
CNT_FAM_MEMBERS                  0
OCCUPATION_TYPE                  0
DAYS_EMPLOYED                    0
DAYS_BIRTH                       0
NAME_HOUSING_TYPE                0
NAME_EDUCATION_TYPE              0
NAME_CONTRACT_TYPE               0
NAME_INCOME_TYPE                 0
AMT_CREDIT                       0
AMT_INCOME_TOTAL                 0
CNT_CHILDREN                     0
FLAG_OWN_REALTY                  0
FLAG_OWN_CAR                     0
CODE_GENDER                      0
YEARS_BEGINEXPLUATATION_AVG      0
dtype: int64
```

In [108…

```
# missing value imputation
def mode_impute(dataset,col):
    return dataset[col].fillna(dataset[col].mode()[0])
dataset['AMT_GOODS_PRICE'] = mode_impute(dataset,'AMT_GOODS_PRICE')
dataset['AMT_ANNUITY'] = mode_impute(dataset,'AMT_ANNUITY')
missing(dataset.select_dtypes('object'))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_54348/1046487735.py in <module>
      4 dataset['AMT_GOODS_PRICE'] = mode_impute(dataset,'AMT_GOODS_PRICE')
      5 dataset['AMT_ANNUITY'] = mode_impute(dataset,'AMT_ANNUITY')
----> 6 missing(dataset.select_dtypes('object'))

NameError: name 'missing' is not defined
```

In [109…

```
dataset.isnull().sum().sort_values(ascending=False)
```

Out[109…

```
TARGET                       0
NAME_CONTRACT_TYPE           0
ORGANIZATION_TYPE            0
REGION_RATING_CLIENT_W_CITY  0
REGION_RATING_CLIENT         0
CNT_FAM_MEMBERS              0
OCCUPATION_TYPE              0
DAYS_EMPLOYED                0
DAYS_BIRTH                   0
NAME_HOUSING_TYPE            0
NAME_FAMILY_STATUS           0
NAME_EDUCATION_TYPE          0
NAME_INCOME_TYPE             0
AMT_GOODS_PRICE              0
AMT_ANNUITY                  0
AMT_CREDIT                   0
AMT_INCOME_TOTAL             0
CNT_CHILDREN                 0
FLAG_OWN_REALTY              0
FLAG_OWN_CAR                 0
CODE_GENDER                  0
YEARS_BEGINEXPLUATATION_AVG  0
dtype: int64
```

In [112…

```
#Create x and y variables
x=dataset.drop('TARGET', axis=1).to_numpy()
y=dataset['TARGET'].to_numpy()

#Create Training and Test Datasets
from sklearn.model_selection import train_test_split
x_train, x_test,y_train, y_test = train_test_split(x, y, stratify=y,test_size=0.2,random_state=100)

#Fix the imbalanced Classes
from imblearn.over_sampling import SMOTE
smt=SMOTE(random_state=100)
x_train_smt,y_train_smt = smt.fit_resample(x_train,y_train)

#Scale the Data
scaler = RobustScaler().fit(x_train)
x_train2 = scaler.transform(x_train)
```

```
x_test2 = scaler.transform(x_test)
print("Done")
```

```
Done
```

In [113…

```
#Current Class Balance - Test Data
print('Current - Class Split')
num_zeros = (y_train == 0).sum()
num_ones = (y_train == 1).sum()
print('TARGET 0 -',  num_zeros)
print('TARGET 1 -',  num_ones)
```

```
Current - Class Split
TARGET 0 - 226148
TARGET 1 - 19860
```

In [114…

```
#Class Balance - Test Data
print('Train Data - Class Split')
num_zeros = (y_train_smt == 0).sum()
num_ones = (y_train_smt == 1).sum()
print('TARGET 0 -',  num_zeros)
print('TARGET 1 -',  num_ones)
```

```
Train Data - Class Split
TARGET 0 - 226148
TARGET 1 - 226148
```

In [117…

```
#Script for Decision Tree
from sklearn.tree import DecisionTreeClassifier

for name,method in [('DT', DecisionTreeClassifier(random_state=100))]:
    method.fit(x_train2,y_train)
    predict = method.predict(x_test2)
    print('\nEstimator: {}'.format(name))
    print(confusion_matrix(y_test,predict))
    print(classification_report(y_test,predict))
```

```
Estimator: DT
[[51299  5239]
 [ 4290   675]]
              precision    recall  f1-score   support

           0       0.92      0.91      0.92     56538
           1       0.11      0.14      0.12      4965

    accuracy                           0.85     61503
   macro avg       0.52      0.52      0.52     61503
weighted avg       0.86      0.85      0.85     61503
```

In [118…

```
#Script for Random Forest
from sklearn.ensemble import RandomForestClassifier

for name,method in [('RF', RandomForestClassifier(random_state=100))]:
    method.fit(x_train2,y_train)
    predict = method.predict(x_test2)
    print('\nEstimator: {}'.format(name))
    print(confusion_matrix(y_test,predict))
    print(classification_report(y_test,predict))
```

```
Estimator: RF
[[56534     4]
 [ 4964     1]]
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     56538
           1       0.20      0.00      0.00      4965

    accuracy                           0.92     61503
   macro avg       0.56      0.50      0.48     61503
weighted avg       0.86      0.92      0.88     61503
```

In [119…
```
#Next Steps - Feature Selection using SelectFromModel
from sklearn.feature_selection import SelectFromModel
clf = DecisionTreeClassifier (class_weight='balanced',
                                                random_state=100)
clf.fit(x_train2,y_train)
model = SelectFromModel(clf, prefit=True)
feature_idx = model.get_support()
feature_name = dataset.drop('TARGET',axis=1).columns[feature_idx]
print('\nKey Features:',feature_name)
```

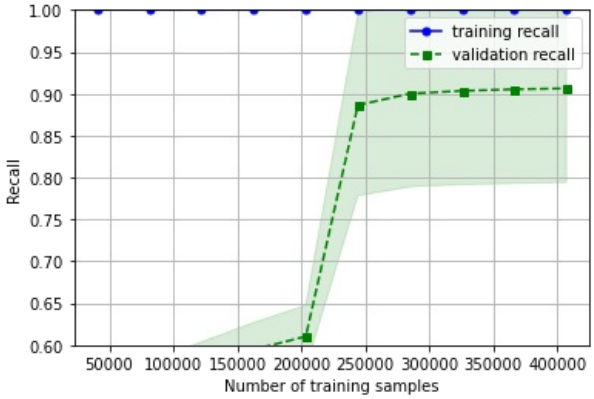
```
Key Features: Index(['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE',
       'DAYS_BIRTH', 'DAYS_EMPLOYED', 'ORGANIZATION_TYPE',
       'YEARS_BEGINEXPLUATATION_AVG'],
      dtype='object')
```

In [123…
```
#Construct some pipelines
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

#Create Pipeline

pipeline =[]

pipe_lda = Pipeline([('scl', StandardScaler()),
                     ('clf', DecisionTreeClassifier())])
pipeline.insert(0,pipe_lda)



pipe_logreg = Pipeline([('scl', StandardScaler())])

pipeline.insert(2,pipe_logreg)


# Set grid search params

modelpara =[]

param_gridlda = {'clf__solver':['svd','lsqr','eigen']}
modelpara.insert(2,param_gridlda)

param_gridqda = {}
modelpara.insert(4,param_gridqda)

param_gridlogreg = {'clf__C': [0.01, 0.1, 1, 10, 100],
                    'clf__penalty': ['l1', 'l2']}
modelpara.insert(2,param_gridlogreg)
```

In [124…
```
#Define Plot for learning curve

from sklearn.model_selection import learning_curve

def plot_learning_curves(model):
    train_sizes, train_scores, test_scores = learning_curve(estimator=model,
                                                            X=x_train_smt,
                                                            y=y_train_smt,
                                                            train_sizes= np.linspace(0.1, 1.0, 10),
                                                            cv=10,
                                                            scoring='recall_weighted',random_state=100)
    train_mean = np.mean(train_scores, axis=1)
    train_std = np.std(train_scores, axis=1)
    test_mean = np.mean(test_scores, axis=1)
    test_std = np.std(test_scores, axis=1)

    plt.plot(train_sizes, train_mean,color='blue', marker='o',
             markersize=5, label='training recall')
    plt.fill_between(train_sizes, train_mean + train_std, train_mean - train_std,
                     alpha=0.15, color='blue')

    plt.plot(train_sizes, test_mean, color='green', linestyle='--', marker='s', markersize=5,
             label='validation recall')
    plt.fill_between(train_sizes, test_mean + test_std, test_mean - test_std,
                     alpha=0.15, color='green')
    plt.grid(True)
    plt.xlabel('Number of training samples')
    plt.ylabel('Recall')
    plt.legend(loc='best')
    plt.ylim([0.6, 1.0])
    plt.show()
```

In [125…
```
#Plot Learning Curve
print('DecisionTree Learning Curve')
plot_learning_curves(pipe_lda)
```

DecisionTree Learning Curve



In [126…

```
#Next Steps - Feature Selection using SelectFromModel
from sklearn.feature_selection import SelectFromModel
clf = DecisionTreeClassifier (class_weight='balanced',
                                                random_state=100)
clf.fit(x_train2,y_train)
model = SelectFromModel(clf, prefit=True)
feature_idx = model.get_support()
feature_name = dataset.drop('TARGET',axis=1).columns[feature_idx]
print('\nKey Features:',feature_name)
```

```
Key Features: Index(['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE',
       'DAYS_BIRTH', 'DAYS_EMPLOYED', 'ORGANIZATION_TYPE',
       'YEARS_BEGINEXPLUATATION_AVG'],
      dtype='object')
```

In [ ]: