In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import gc
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, precision_recall_curve, roc_curve, auc, ave
from keras.models import Sequential, Model
from keras.layers import Dense, Activation,Reshape, Dropout, Input, Flatten, Concatena
from keras.layers import Embedding
from keras.callbacks import EarlyStopping
```

In [2]:
```
house_df=pd.read_csv('House Loan Data.csv')
house_df.head()
```

Out[2]:

	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

In [3]:
```
house_df.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
```

Checking Missing Values with Percentage

In [4]:
```
house_df.isnull().sum().sort_values(ascending = False).head(20)
```

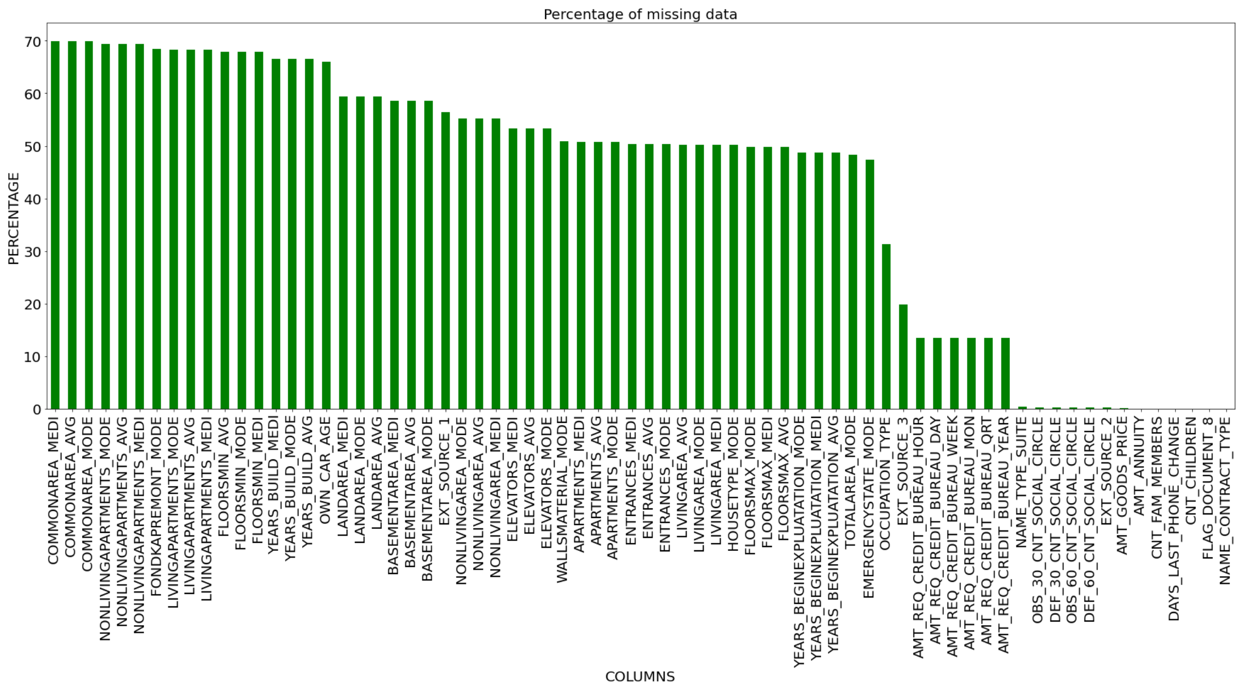
Out[4]:
```
COMMONAREA_MEDI             214865
COMMONAREA_AVG              214865
COMMONAREA_MODE             214865
NONLIVINGAPARTMENTS_MODE    213514
NONLIVINGAPARTMENTS_AVG     213514
NONLIVINGAPARTMENTS_MEDI    213514
FONDKAPREMONT_MODE          210295
LIVINGAPARTMENTS_MODE       210199
LIVINGAPARTMENTS_AVG        210199
LIVINGAPARTMENTS_MEDI       210199
FLOORSMIN_AVG               208642
FLOORSMIN_MODE              208642
FLOORSMIN_MEDI              208642
YEARS_BUILD_MEDI            204488
YEARS_BUILD_MODE            204488
YEARS_BUILD_AVG             204488
OWN_CAR_AGE                 202929
LANDAREA_MEDI               182590
LANDAREA_MODE               182590
LANDAREA_AVG                182590
dtype: int64
```

In [5]:
```python
missing_values_percentage = (house_df.isnull().sum()/len(house_df)*100).sort_values(as
missing_values_percentage.head(20)
```

Out[5]:
```
COMMONAREA_MEDI             69.872297
COMMONAREA_AVG              69.872297
COMMONAREA_MODE             69.872297
NONLIVINGAPARTMENTS_MODE    69.432963
NONLIVINGAPARTMENTS_AVG     69.432963
NONLIVINGAPARTMENTS_MEDI    69.432963
FONDKAPREMONT_MODE          68.386172
LIVINGAPARTMENTS_MODE       68.354953
LIVINGAPARTMENTS_AVG        68.354953
LIVINGAPARTMENTS_MEDI       68.354953
FLOORSMIN_AVG               67.848630
FLOORSMIN_MODE              67.848630
FLOORSMIN_MEDI              67.848630
YEARS_BUILD_MEDI            66.497784
YEARS_BUILD_MODE            66.497784
YEARS_BUILD_AVG             66.497784
OWN_CAR_AGE                 65.990810
LANDAREA_MEDI               59.376738
LANDAREA_MODE               59.376738
LANDAREA_AVG                59.376738
dtype: float64
```

In [6]:
```python
missing_values_percentage.head(70).plot(kind = 'bar', figsize = (30,10), color = 'gree
plt.title('Percentage of missing data',size = '20')
plt.xticks(fontsize =20)
plt.yticks(fontsize = 20)
plt.ylabel("PERCENTAGE", size = 20)
plt.xlabel("COLUMNS", size = 20)
```

Out[6]:
```
Text(0.5, 0, 'COLUMNS')
```



Percentage of Defaulter to Payer from Dataset for Target Column

```
In [7]: Percentage_Payer = house_df['TARGET'].value_counts()[0]/len(house_df)*100
        Percentage_Payer
```

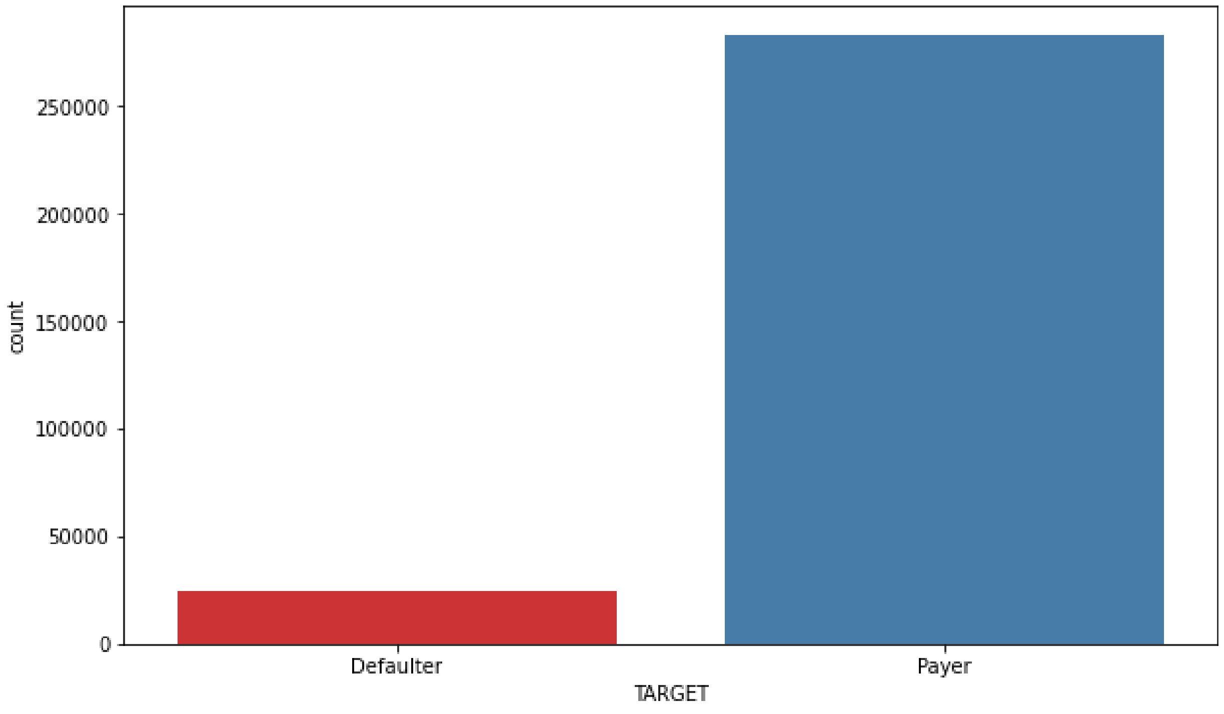
```
Out[7]: 91.92711805431351
```

```
In [8]: Percentage_Defaulter = house_df['TARGET'].value_counts()[1]/len(house_df)*100
        Percentage_Defaulter
```

```
Out[8]: 8.072881945686495
```

```
In [9]: house_df['TARGET'] = house_df['TARGET'].replace({0:'Payer',1:'Defaulter'})
        plt.figure(figsize = (10,6))
        sns.countplot(x = 'TARGET', data = house_df, palette = 'Set1')
```

```
Out[9]: <AxesSubplot:xlabel='TARGET', ylabel='count'>
```



Balancing the Dataset

```
In [10]: # Number of Payer
         house_df['TARGET'].value_counts()[0]
```

```
Out[10]: 282686
```

```
In [11]: house_df['TARGET'].value_counts()[1]
```

```
Out[11]: 24825
```

```
In [24]: from sklearn.utils import resample
         df_1 = house_df[house_df['TARGET'] == 1]
         df_0 = house_df[house_df['TARGET'] == 0]
```

```
In [34]: df_1_upsample = resample(df_1 , n_samples = 0, replace = True)
```

```
In [35]: house_df1 = pd.concat([df_0 , df_1_upsample])
         house_df1
```

Out[35]:

SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY

0 rows × 122 columns

```
In [ ]:
```