Team 5 Final Project - Manikandan Ramalingam

Credit Default Analysis

In machine learning, the algorithms are just the tools, the raw material is the data - it's the ore that makes the gold. Thus, to build useful models, one needs to get intimate with data — it's strengths, flaws, nuances, patterns, cycles, etc. Graphical data analysis is much more than mere visualization.

Feel the Credit Default Risk Data set

There are multiple ways to analyze the data like human judgement based on the experience in the domain, utilizing various statistical and graphical analysis tools or picking features based on popular existing well defined models like Random forest classification, Principal component analysis etc. But, for all, the preliminary step would be to get the feel of the data set. The shape would provide the number of rows and columns (or features). This would enable us to use appropriate techniques for data cleansing. The below code does check the shape and type of parameters by printing the top 5 rows.

Get the Credit Default Dataset

By convention, seaborn is imported with the shorthand 'sns'. Seaborn includes a few example datasets. Let's import seaborn and load a dataset to start plotting. spellling

In [1]:

```
# Import seaborn
import seaborn as sns
import matplotlib.pyplot as plt #to allow subplot creation
import pandas as pd

# Fetch the train data into the data frame
df = pd.read_csv('/Users/manikanr/Downloads/assignment/train_data.csv')
# Apply the seaborn theme
sns.set_theme() #overwrite default Matplotlib styling parameters

shape = df.shape
print("Shape of the dataframe (row, col):",shape,"\r\n")

# Show the dataframe
df.head()
df.shape
```

```
Shape of the dataframe (row, col): (153755, 122)
```

Out[1]:

```
(153755, 122)
```

Analyze a Target variable without Feature Engineering

First, use all 121 features to analyze the target variable. Use GBC Tree classifier to predict the values and test for accuracy. Since we are not using Feature Engineering, we can select only numeric columns. So, first select all numeric columns from the data frame. Otherwise, we cannot apply the model classification with the combination of strings and numeric values. Also, drop the entire row when null values are present. Although this a feature enginnering step, without this basic data cleansing, we cannot predict the results.

In [4]:

```
import numpy as np
```

```
# Select only numeric columns
numeric_df = df.select_dtypes(include=['number'])

# Function to impute NaN with mean and floor the result
def impute_and_floor(df):
    # Select numeric columns
    numeric_cols = df.select_dtypes(include=[np.number])

    # Impute NaN with mean and floor the values
    for col in numeric_cols.columns:
        mean_value = numeric_cols[col].mean()
        df[col].fillna(mean_value, inplace=True)
        df[col] = np.floor(df[col])

    return df

# Apply the function to the DataFrame
df_cleaned = impute_and_floor(numeric_df)

df_cleaned.head()
df_cleaned.shape
```

Out[4]:

```
(153755, 106)
```

Check for Precision, Recall, F1-score and Accuracy without Feature Engineering Using GradientBoostingClassifier

This data will provide the baseline.

In [5]:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import classification_report

# Just drop the target column from 106 numeric feature columns
x_train, x_test, y_train, y_test = train_test_split(df_cleaned.drop(['TARGET'], axis='co
lumns'), df_cleaned.TARGET, test_size=0.2)

# Initialize and train the model
xgb_clf = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, r
andom_state=42)
xgb_clf.fit(x_train, y_train)

# Make predictions
y_pred = xgb_clf.predict(x_test)

# Evaluate the model
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

         0.0       0.92      1.00      0.96     28301
         1.0       0.00      0.00      0.00      2450

    accuracy                           0.92     30751
   macro avg       0.46      0.50      0.48     30751
weighted avg       0.85      0.92      0.88     30751
```

Feature Engineering Techniques

Applying Human Judgement First

There are 122 columns (or features) in this credit risk default file. Since we are predicting whether to provide loan or not based on the credit profile, this is a classification task in Machine Learning Paradigm. The loan re-payment depends on various factors like income, number of children in family, family members, type of occupation, assets, previous credits, previous credit account defaults, desperate to get loan (credit enquiries in past few months), instances of 30/60 day past due or earlier credits etc. So, before applying any statistical, graphical or Machine learning models, some important features are selected based on experience (application for earlier credits would also be considered experience) in the given domain.

The top features based on human judgement and reasons is below.

1. **AMT_INCOME_TOTAL - Income of Client**
2. **AMT_CREDIT - Loan Amount**
3. **AMT_ANNUITY - Loan Annuity**
4. **AMT_GOODS_PRICE - For Consumer loans**
5. **NAME_INCOME_TYPE - Income through family business, working salaried professional or Other)**
6. **NAME_EDUCATION_TYPE - This is important because well educated individuals tend to get more salaries over time**

```
                          and experience.
```

7. **NAME_HOUSING_TYPE - Rent or Own plays a role.**
8. **NAME_FAMILY_STATUS - Married, separated and paying alimony matters.**
9. **CNT_CHILDREN - Number of children if a person has to do child support.**
10. **FLAG_OWN_CAR - Do you own a car**
11. **FLAG_OWN_REALTY - Own any Realty**
12. **DAYS_BIRTH - How many since the client is born. The more in the range (>21 < 37), the better.**
13. **DAYS_EMPLOYED - Employment days. The more years results in higher salary.**
14. **FLAG_CONT_MOBILE - Mobile phone reachable to call in case of default**
15. **CNT_FAM_MEMBERS - Number of family members**
16. **REG_REGION_NOT_LIVE_REGION - If permanent address matches with contact address**
17. **LIVE_REGION_NOT_WORK_REGION - If work address not closer to contact address**
18. **ORGANIZATION_TYPE - Type of organization where client works. This is important to judge future growth on**

```
                        client's salary.
```

19. **DEF_60_CNT_SOCIAL_CIRCLE - How many observation of client's social surroundings defaulted on 60 DPD (days past due)**
20. **AMT_REQ_CREDIT_BUREAU_MON - Number of enquiries to Credit Bureau about the client one month before application**

20 Features Extraction

Extract the features in a data frame. Also, do some data cleansing with null values populated with a mean value of columns. This will make the analysis of the feature set easier.

In [6]:

```
# Extract these 21 variables
df_extract_21 = df[['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_INCOME_TYPE',
                    'NAME_EDUCATION_TYPE', 'NAME_HOUSING_TYPE', 'NAME_FAMILY_STATUS', 'CNT_CHILDREN', 'FLAG_OWN_CAR',
                    'FLAG_OWN_REALTY', 'DAYS_BIRTH', 'DAYS_EMPLOYED', 'FLAG_CONT_MOBILE', 'CNT_FAM_MEMBERS',
                    'REG_REGION_NOT_LIVE_REGION', 'LIVE_REGION_NOT_WORK_REGION', 'ORGANIZATION_TYPE',
                    'DEF_60_CNT_SOCIAL_CIRCLE', 'AMT_REQ_CREDIT_BUREAU_MON']]
df_extract_21.head()
```

Out[6]:

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE
0	157500.0	900000.0	26446.5	900000.0	Working	Secondary / secondary special
1	90000.0	733176.0	21438.0	612000.0	Working	Higher education
2	189000.0	1795500.0	62541.0	1795500.0	Pensioner	Secondary / secondary special
3	175500.0	494550.0	45490.5	450000.0	Pensioner	Higher education
4	270000.0	1724688.0	54283.5	1575000.0	Working	Higher education

Feature Engineering Technique1 - Mean Imputation and Normalization

For all the numeric values in 20 features set, populate the mean. Also, select the minimum value when selecting the mean as some features might not be floating point values.

In [7]:

```
import numpy as np

# Extract these 21 variables
df_extract_21 = df[['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_INCOME_TYPE',
                    'NAME_EDUCATION_TYPE', 'NAME_HOUSING_TYPE', 'NAME_FAMILY_STATUS', 'CNT_CHILDREN', 'FLAG_OWN_CAR',
                    'FLAG_OWN_REALTY', 'DAYS_BIRTH', 'DAYS_EMPLOYED', 'FLAG_CONT_MOBILE', 'CNT_FAM_MEMBERS',
                    'REG_REGION_NOT_LIVE_REGION', 'LIVE_REGION_NOT_WORK_REGION', 'ORGANIZATION_TYPE',
                    'DEF_60_CNT_SOCIAL_CIRCLE', 'AMT_REQ_CREDIT_BUREAU_MON']]

# Replace NaN values in specific columns with mean
columns_to_fill = ['AMT_ANNUITY', 'AMT_GOODS_PRICE', 'CNT_FAM_MEMBERS', 'DEF_60_CNT_SOCIAL_CIRCLE', 'AMT_REQ_CREDIT_BUREAU_MON']

# Calculate the mean of specific columns and round down to the nearest integer
mean_values = df_extract_21[columns_to_fill].mean().apply(np.floor)

# Fill NaN values in df_extract_21 with the calculated mean values
df_extract_21[columns_to_fill] = df_extract_21[columns_to_fill].fillna(mean_values)
df_extract_21.head()
```

```
<ipython-input-7-fe77968494d2>:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_extract_21[columns_to_fill] = df_extract_21[columns_to_fill].fillna(mean_values)
```

Out[7]:

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE
0	157500.0	900000.0	26446.5	900000.0	Working	Secondary / secondary special
1	90000.0	733176.0	21438.0	612000.0	Working	Higher education
2	189000.0	1795500.0	62541.0	1795500.0	Pensioner	Secondary / secondary special
3	175500.0	494550.0	45490.5	450000.0	Pensioner	Higher education
4	270000.0	1724688.0	54283.5	1575000.0	Working	Higher education

Feature Engineering Technique 2 - Encoding Categorical Variables

Feature Engineering Technique 2 - Encoding Categorical Variables

Convert the categorical variables to numeric values using encoding tenchnique. Also, convert few columns selected with negative values to positive values.

In [8]:

```
from sklearn.preprocessing import LabelEncoder

columns_to_encode = ['NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'NAME_HOUSING_TYPE', 'NAM
E_FAMILY_STATUS',
                     'ORGANIZATION_TYPE', 'FLAG_OWN_CAR', 'FLAG_OWN_REALTY', 'FLAG_CONT_
MOBILE']

# Encode categorical columns
label_encoder = LabelEncoder()
for col in columns_to_encode:
    df_extract_21[col] = label_encoder.fit_transform(df_extract_21[col])
df_extract_21.head()
```

```
<ipython-input-8-bc6de7c0550c>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  df_extract_21[col] = label_encoder.fit_transform(df_extract_21[col])
<ipython-input-8-bc6de7c0550c>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  df_extract_21[col] = label_encoder.fit_transform(df_extract_21[col])
<ipython-input-8-bc6de7c0550c>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  df_extract_21[col] = label_encoder.fit_transform(df_extract_21[col])
<ipython-input-8-bc6de7c0550c>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  df_extract_21[col] = label_encoder.fit_transform(df_extract_21[col])
<ipython-input-8-bc6de7c0550c>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  df_extract_21[col] = label_encoder.fit_transform(df_extract_21[col])
<ipython-input-8-bc6de7c0550c>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  df_extract_21[col] = label_encoder.fit_transform(df_extract_21[col])
<ipython-input-8-bc6de7c0550c>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  df_extract_21[col] = label_encoder.fit_transform(df_extract_21[col])
<ipython-input-8-bc6de7c0550c>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  df_extract_21[col] = label_encoder.fit_transform(df_extract_21[col])
```

Out[8]:

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE
0	157500.0	900000.0	26446.5	900000.0	7	4
1	90000.0	733176.0	21438.0	612000.0	7	1
2	189000.0	1795500.0	62541.0	1795500.0	3	4
3	175500.0	494550.0	45490.5	450000.0	3	1
4	270000.0	1724688.0	54283.5	1575000.0	7	1

Feature Engineering Technique 3 - Data Transformation

Note that Days birth and days employed are negative values. It is transformed to positive values for getting good prediction.

In [9]:

```python
# Convert negative to positive values
columns_to_convert_positive = ['DAYS_BIRTH', 'DAYS_EMPLOYED']
for col in columns_to_convert_positive:
    df_extract_21[col] = df_extract_21[col].abs()

df_extract_21.head()
```

```
<ipython-input-9-8bdef085ec8d>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  df_extract_21[col] = df_extract_21[col].abs()
```

Out[9]:

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE
0	157500.0	900000.0	26446.5	900000.0	7	4
1	90000.0	733176.0	21438.0	612000.0	7	1
2	189000.0	1795500.0	62541.0	1795500.0	3	4
3	175500.0	494550.0	45490.5	450000.0	3	1
4	270000.0	1724688.0	54283.5	1575000.0	7	1

Feature Engineering Technique 4 - Dimensionality Reduction to Extract 10 Most Important Features from 21

Extract the 10 most important features in a data frame out of 21. There are many techniques that can be used for this.

1. **Use Random Forest classifier and select top 10.**
2. **Use Principical Component Analysis.**
3. **SelectKBest an univariate method to select K=10 best features.**

SelectKBest Classification with Chi2

In [10]:

```
from sklearn.datasets import make_classification
from sklearn.feature_selection import SelectKBest, chi2

# Generate a sample regression dataset
#X, y = chi2(n_samples=df_extract_21.shape[0], n_features=df_extract_21.shape[1], random_
state=42)
# Perform feature selection using chi-squared test
selector = SelectKBest(score_func=chi2, k=10)  # Select top 10 features
y= df[['TARGET']]
X_new = selector.fit_transform(df_extract_21, y)
# Print the selected features
selected_features = df_extract_21.columns[selector.get_support()]
print("\nSelected features from SelectKBest with chi2 classification:\n")
print(selected_features)
```

```
Selected features from SelectKBest with chi2 classification:

Index(['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE',
       'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'DAYS_BIRTH',
       'DAYS_EMPLOYED', 'ORGANIZATION_TYPE', 'DEF_60_CNT_SOCIAL_CIRCLE'],
      dtype='object')
```

Random Forest Classifier to identify top features

In [11]:

```
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier

# Train Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
y= df[['TARGET']]
rf_model.fit(df_extract_21, y)

# Get feature importances
importances = rf_model.feature_importances_

# Get indices of top 10 features
top10_indices = np.argsort(importances)[::-1][:10]
print("\nSelected Features from Random Forest classification:\n")
for i, idx in enumerate(top10_indices):
    print(df_extract_21.columns[idx])
```

```
/Users/manikanr/anaconda3/lib/python3.8/site-packages/sklearn/base.py:1152: DataConversio
nWarning: A column-vector y was passed when a 1d array was expected. Please change the sh
ape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

```
Selected Features from Random Forest classification:

DAYS_BIRTH
AMT_ANNUITY
DAYS_EMPLOYED
AMT_CREDIT
AMT_INCOME_TOTAL
AMT_GOODS_PRICE
ORGANIZATION_TYPE
NAME_FAMILY_STATUS
CNT_FAM_MEMBERS
CNT_CHILDREN
```

PCA Analysis (unsupervised) to identify top 10 features

In [12]:

```
from sklearn.decomposition import PCA
```

```
# Initialize PCA with desired number of components (e.g., 10 for selecting top 10 compone
nts)
n_components = 10
pca = PCA(n_components=n_components)

# Fit PCA on the data and transform it
X_pca = pca.fit_transform(df_extract_21)

# Optionally, you can also access the principal components (eigenvectors)
principal_components = pca.components_

# Get the indices of the top 10 principal components with the largest explained variance
top10_indices = np.argsort(pca.explained_variance_ratio_)[::-1][:10]

# Get the names of the top 10 features corresponding to the top principal components
top10_features = []
for idx in top10_indices:
    component = principal_components[idx]
    relevant_features = df_extract_21.columns[np.abs(component) > 0.1]
    top10_features.extend(relevant_features)

# Remove duplicates (if any)
top10_features = list(set(top10_features))

# Print the names of the top 10 features
print("Top 10 features:")
print(top10_features)
```

```
Top 10 features:
['NAME_EDUCATION_TYPE', 'CNT_FAM_MEMBERS', 'CNT_CHILDREN', 'AMT_ANNUITY', 'AMT_GOODS_PRIC
E', 'NAME_HOUSING_TYPE', 'AMT_INCOME_TOTAL', 'NAME_FAMILY_STATUS', 'DAYS_BIRTH', 'ORGANIZ
ATION_TYPE', 'NAME_INCOME_TYPE', 'AMT_CREDIT', 'DAYS_EMPLOYED']
```

Conclusion on Top 10 features from above analysis

Top 10 features based on mode from above 3 supervised/unsupervised analysis results:-

'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE' 'NAME_INCOME_TYPE', 'NAME_EDUCATION_TYPE', 'DAYS_BIRTH', 'DAYS_EMPLOYED' 'ORGANIZATION_TYPE', 'CNT_FAM_MEMBERS'

In [13]:

```
df_extract_10 = df_extract_21[['AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOOD
S_PRICE', 'NAME_INCOME_TYPE',
                   'NAME_EDUCATION_TYPE', 'DAYS_BIRTH', 'DAYS_EMPLOYED', 'ORGANIZATION_T
YPE',
                   'CNT_FAM_MEMBERS']]
df_extract_10.head()
#df_extract_10.shape
```

Out[13]:

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE
0	157500.0	900000.0	26446.5	900000.0	7	4
1	90000.0	733176.0	21438.0	612000.0	7	1
2	189000.0	1795500.0	62541.0	1795500.0	3	4
3	175500.0	494550.0	45490.5	450000.0	3	1
4	270000.0	1724688.0	54283.5	1575000.0	7	1

Prediction After Applying Feature Engineering Technique

Used RandomForestClassifier on the top 10 features after Feature Engineering techniques.

In [14]:

```
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# Train Random Forest model
# Split the data
x_train, x_test, y_train, y_test = train_test_split(df_extract_10, df['TARGET'], test_si
ze=0.2, random_state=42)

# Initialize the classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_classifier.fit(x_train, y_train)

# Make predictions
y_pred = rf_classifier.predict(x_test)

# Evaluate the model
report = classification_report(y_test, y_pred)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.92      1.00      0.96     28294
           1       0.21      0.00      0.00      2457

    accuracy                           0.92     30751
   macro avg       0.57      0.50      0.48     30751
weighted avg       0.86      0.92      0.88     30751
```

Conclusion

From the analysis above with human judgement, different models and graphical analysis it seems the top 10 features out of 122 features in the train_set.csv seems to be 'AMT_INCOME_TOTAL', 'AMT_CREDIT', 'AMT_ANNUITY', 'AMT_GOODS_PRICE', 'NAME_INCOME_TYPE’, 'NAME_EDUCATION_TYPE’, 'DAYS_BIRTH’, 'DAYS_EMPLOYED' 'ORGANIZATION_TYPE’, 'CNT_FAM_MEMBERS'. This will enable the credit decision when applied with different machine learning models and hyper parameter tuning. On classification report analysis, the report is similar to the one we have at the top with 106 features. But, it took long time to train those compared to reduced dimensions of 10 values. So, it shouldn't be interpreted that we can use entire 106 features. It boils down to just using top 10 features perform with accuracy of 92%. This shows the importance of Feature Engineering.

Disclosure This is based on only 122 features and not the data in other csv files. There might be appropriate data in other files which might be more relevant. This exercise is focused on train_test.csv file.

In [16]:

```
# The Cleansed data frame with 10 features after applying Feature Engineering is shown be
low.
# Further model fits will be done using this
df_extract_10 = df_extract_10.join(df[['TARGET']])
df_extract_10.head()
```

Out[16]:

	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE
0	157500.0	900000.0	26446.5	900000.0	7	4
1	90000.0	733176.0	21438.0	612000.0	7	1
2	189000.0	1795500.0	62541.0	1795500.0	3	4
3	175500.0	494550.0	45490.5	450000.0	3	1

4	AMT_INCOME_TOTAL	270000.0	AMT_CREDIT	1724688.0	AMT_ANNUITY	54283.5	AMT_GOODS_PRICE	1575000.0	NAME_INCOME_TYPE	7	NAME_EDUCATION_TYPE	1	

In [ ]: