In [1]:

```
# Import necessary packages,Pandas is used for data manipulation
import pandas as pd
import numpy as np
# Read in data and display first 5 rows
features = pd.read_csv('loan_data_set2.csv')
features.head(5)
```

Out[1]:

	Loa0_ID	Ge0der	Married	Depe0de0ts	Self_Emplo1ed	Applica0tI0come	Coapplica0tI0com
0	1002	1.0	0.0	0.0	0.0	5849	0
1	1003	1.0	1.0	1.0	0.0	4583	1508
2	1005	1.0	1.0	0.0	1.0	3000	0
3	1006	1.0	1.0	0.0	0.0	2583	2358
4	1008	1.0	0.0	0.0	0.0	6000	0

In [2]:

```
features = features.dropna()
```

In [3]:

```
print('The shape of our features is:', features.shape)
# Descriptive statistics for each column
features.describe()
```

```
The shape of our features is: (480, 12)
```

Out[3]:

	Loa0_ID	Ge0der	Married	Depe0de0ts	Self_Emplo1ed	Applica0tI0come	C
count	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	
mean	2002.158333	0.820833	0.647917	0.777083	0.137500	5364.231250	
std	566.898488	0.383892	0.478118	1.020815	0.344734	5668.251251	
min	1003.000000	0.000000	0.000000	0.000000	0.000000	150.000000	
25%	1535.750000	1.000000	0.000000	0.000000	0.000000	2898.750000	
50%	1975.500000	1.000000	1.000000	0.000000	0.000000	3859.000000	
75%	2475.750000	1.000000	1.000000	2.000000	0.000000	5852.500000	
max	2990.000000	1.000000	1.000000	3.000000	1.000000	81000.000000	

In [4]:

```
# One-hot encode the data using pandas get_dummies
features = pd.get_dummies(features)
# Display the first 5 rows of the last 12 columns
features.iloc[:,5:].head(5)
```

Out[4]:

	ApplicaOtI0come	CoapplicaOtI0come	Loa0Amou0t	Loa0_Amou0t_Term	Credit_Histor1	Pro
1	4583	1508.0	128.0	360.0	1.0	
2	3000	0.0	66.0	360.0	1.0	
3	2583	2358.0	120.0	360.0	1.0	
4	6000	0.0	141.0	360.0	1.0	
5	5417	4196.0	267.0	360.0	1.0	

In [5]:

```
# Use numpy to convert to arrays
# Labels are the values we want to predict
labels = np.array(features['Loa0_Status'])
# Remove the labels from the features
# axis 1 refers to the columns
features= features.drop('Loa0_Status', axis = 1)
# Saving feature names for later use
feature_list = list(features.columns)
# Convert to numpy array
features = np.array(features)
```

In [22]:

```
# Using Skicit-learn to split data into training and testing sets
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
train_features, test_features, train_labels, test_labels = train_test_split
print('Training Features Shape:', train_features.shape)
print('Training Labels Shape:', train_labels.shape)
print('Testing Features Shape:', test_features.shape)
print('Testing Labels Shape:', test_labels.shape)
```

```
Training Features Shape: (360, 11)
Training Labels Shape: (360,)
Testing Features Shape: (120, 11)
Testing Labels Shape: (120,)
```

In [23]:

```
# The baseline predictions are the historical averages
baseline_preds = test_features[:, feature_list.index('Applica0tI0come')]
# Baseline errors, and display average baseline error
baseline_errors = abs(baseline_preds - test_labels)
print('Average baseline error: ', round(np.mean(baseline_errors), 2))
```

```
Average baseline error:  5681.95
```

In [24]:

```
# Import the model we are using
from sklearn.ensemble import RandomForestRegressor
# Instantiate model with 1000 decision trees
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
# Train the model on training data
rf.fit(train_features, train_labels);
```

In [25]:

```
# Use the forest's predict method on the test data
predictions = rf.predict(test_features)
# Calculate the absolute errors
errors = abs(predictions - test_labels)
# Print out the mean absolute error (mae)
print('Mean Absolute Error:', round(np.mean(errors), 2))
```

```
Mean Absolute Error: 0.3
```

In [28]:

```
# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors/test_labels)
# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Accuracy: -inf %.
```