In [19]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler, LabelEncoder
import numpy as np
```

# Preprocessing: Converting Categorical Data to Numeric

In [11]:
```
# Load the two CSVs and separate target columns from train and test data
train_df = pd.read_csv('2019loans.csv')
test_df = pd.read_csv('2020Q1loans.csv')
y_train = train_df["loan_status"]
X_train = train_df.drop(columns = "loan_status")
y_test = test_df["loan_status"]
X_test = test_df.drop(columns = "loan_status")
```

In [12]:
```
X_train.head(5)
```

Out[12]:

| | Unnamed: 0 | index | loan_amnt | int_rate | installment | home_ownership | annual_inc | verification |
|---|---|---|---|---|---|---|---|---|
| 0 | 57107 | 57107 | 13375.0 | 0.1797 | 483.34 | MORTGAGE | 223000.0 | Not |
| 1 | 141451 | 141451 | 21000.0 | 0.1308 | 478.68 | MORTGAGE | 123000.0 | Source |
| 2 | 321143 | 321143 | 20000.0 | 0.1240 | 448.95 | MORTGAGE | 197000.0 | Source |
| 3 | 11778 | 11778 | 3000.0 | 0.1240 | 100.22 | RENT | 45000.0 | Not |
| 4 | 169382 | 169382 | 30000.0 | 0.1612 | 1056.49 | MORTGAGE | 133000.0 | Source |

5 rows × 85 columns

In [13]:
```
X_test.head(5)
```

Out[13]:

| | Unnamed: 0 | index | loan_amnt | int_rate | installment | home_ownership | annual_inc | verification_ |
|---|---|---|---|---|---|---|---|---|
| 0 | 67991 | 67991 | 40000.0 | 0.0819 | 814.70 | MORTGAGE | 140000.0 | Not \ |
| 1 | 25429 | 25429 | 6000.0 | 0.1524 | 208.70 | RENT | 55000.0 | Not \ |
| 2 | 38496 | 38496 | 3600.0 | 0.1695 | 128.27 | RENT | 42000.0 | Not \ |
| 3 | 19667 | 19667 | 20000.0 | 0.1524 | 478.33 | RENT | 100000.0 | Not \ |
| 4 | 37505 | 37505 | 3600.0 | 0.1240 | 120.27 | RENT | 50000.0 | Not \ |

5 rows × 85 columns

In [16]:
```
# one-hot encoding
X_dummies_train = pd.get_dummies(X_train)
X_dummies_test = pd.get_dummies(X_test)
```

```
y_label_train = LabelEncoder().fit_transform(train_df['loan_status'])
y_label_test = LabelEncoder().fit_transform(test_df['loan_status'])
print(f"Train: {X_dummies_train.shape}, Test: {X_dummies_test.shape}")
```

```
Train: (12180, 94), Test: (4702, 93)
```

In [18]:

```
# the training data has 94 columns and the testing data has 93 columns, so we ne
for col in X_dummies_train.columns:
    if col not in X_dummies_test.columns:
        X_dummies_test[col]=0
print(f"Train: {X_dummies_train.shape}, Test: {X_dummies_test.shape}")
# now both data sets have 94 features and we can proceed
```

```
Train: (12180, 94), Test: (4702, 94)
```

## Personal Prediction

Because Logistic Regression is based off of Linear Regression, it performs best with a linearly separable type of training set. However, our training data set has many categories and is not linearly separable, so I predict that the Random Forest Classifier will perform more accurately.

# Fit a LogisticRegression model and RandomForestClassifier model

### Logistic Regression

In [23]:

```
# Create a Logistic Regression model
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
# Fit to training data
classifier.fit(X_dummies_train, y_label_train)
# Print model score
print(f"Training Data Score: {classifier.score(X_dummies_train, y_label_train)}"
print(f"Testing Data Score: {classifier.score(X_dummies_test, y_label_test)}")
```

```
Training Data Score: 0.648440065681445
Testing Data Score: 0.5250957039557635
/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:76
3: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressio
n
  n_iter_i = _check_optimize_result(
```

### Random Forest Classifier

In [24]:

```
# Create a Random Forest Classifier model
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier()
# Fit to training data
classifier.fit(X_dummies_train, y_label_train)
```

```
# Print model score
print(f"Training Data Score: {classifier.score(X_dummies_train, y_label_train)}"
print(f"Testing Data Score: {classifier.score(X_dummies_test, y_label_test)}")
```

```
Training Data Score: 1.0
Testing Data Score: 0.6069757549978733
```

## Personal Prediction for Scaled Data

I predict the LR scores will improve, but the RFC scores will not as this type of classifier is independent of feature scaling.

In [29]:

```
# Scale data
scaler = StandardScaler().fit(X_dummies_train)
X_train_scaled = scaler.transform(X_dummies_train)
X_test_scaled = scaler.transform(X_dummies_test)
scaled_LR = LogisticRegression().fit(X_train_scaled, y_label_train).score(X_test
scaled_RFC = RandomForestClassifier().fit(X_train_scaled, y_label_train).score(X
print(f' The new Logistic Regression test score is: {scaled_LR}')
print(f' The new Random Forest Classifier test score is: {scaled_RFC}')
```

```
/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:76
3: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressio
n
  n_iter_i = _check_optimize_result(
 The new Logistic Regression test score is: 0.7203317737133135
 The new Random Forest Classifier test score is: 0.6135686941726924
```

### Results:

The Logistic Regression score improved with feature scaling as predicted, but the Random Forest Classifier also improved very slightly, and I was expecting no change. Perhaps it was due to combing the fit and score operations into one line rather than separate like I did pre-scaling (rounding differences).

In [ ]: