# Predicting Loan Grades Using Machine Learning Approaches: Decision Tree-Based Ensemble Models

## Question: What are the potential determinants that could impact the loan grading procedure?

## Introduction

Nowadays, with the progressive escalation of living expenditures, it has become increasingly prevalent for individuals to engage in excessive spending, which can be unmanageable and extend to activities such as frequently using credit cards and availing loans to fulfill their needs. Consequently, it is imperative for lending institutions to assess a borrower's eligibility for a loan based on their credit profile. However, by utilizing advanced machine learning techniques, we can accurately forecast the loan grade of borrowers, enabling lenders to make well-informed, data-driven decisions, hence reducing default rates and causing a change in the lending industry.

Loans are counted as a form of debt, often presented as mortgages, business loans, student loans, or credit card loans. They generally have one party being the lender, typically a lending institution such as a commercial bank. On the other side, the borrowers of the loans are charged with an interest rate higher than the risk-free rate depending varied on a set of different aspects of the borrower. Predicting loan grading is crucial for all lending institutions, regardless of size, as it enables them to minimize unnecessary expenses and improve their business profitability by utilizing their own loan grading system.

Granting a large amount of money to a borrower with a low-interest rate on a loan that is actually of low credit quality can result in a significant loss due to the high possibility of default. To mitigate such risks, commercial banks evaluate the creditworthiness of the borrower and determine the appropriate interest rate for the loan (Bai & Zha, 2022). Thus, precise assessment of loan grading is essential for ensuring accuracy in setting the interest rate and loan prices. It is also a key component of credit risk management, which is a major concern for lending institutions of all sizes worldwide.

## Background

Loan grading is similar in the sense of credit rating, but in the view of loans. Based on the article by Will Kenton, the determination of loan grades is dependent on various factors such as the likelihood of default, collateral value, and the borrower's credit history. Approaches to determining a borrower’s loan grade may vary across different lending institutions depending on the size and complexity of these institutions (Kenton, 2021). Loan scores are typically determined by the borrower's credit score as well as other credit risk indicators such as the support provided by the guarantor, repayment history, cash flow, and projected annual expenses (Kenton, 2021).

According to Ben Armstrong's essay, it is indicated that borrowers who receive an A grade on their loan have the least evaluated risk of loss and are therefore charged the lowest interest rates. Conversely, a G

grade indicates the highest anticipated risk of loss, and G loans carry a greater risk than A loans, therefore, they have the highest interest rates (Armstrong, 2019). A loan-to-value (LTV) ratio is assigned to each loan in relation to its grade. The loan grades can be determined by the LTV ratio, with grade A loans having the lowest LTV ratio whereas grade G loans have the highest (Armstrong, 2019). A lower LTV ratio indicates that the value of the loan is lower than that of the property (Pirgaip & Hepsen, 2018).

Our primary objective is to identify the potentially influential factors that may impact the prediction of loan grades. Given the nature of the problem as a classification task, it is logical to utilize decision trees owing to their interpretability and visualizability. Moreover, we have utilized ensemble models, such as the random forest and gradient boosting models, to enhance the performance of our models. In the subsequent sections, we shall expound upon the rationale for our model selection, provide pertinent information about the dataset, and elucidate the process of optimizing our models.

# Model

***Model Choice***

Given the categorical nature of our response variable with multiple categories, a decision tree model is a suitable initial approach. Decision trees have demonstrated considerable efficacy in dealing with categorical variables, particularly in cases where there exist noteworthy interactions among variables. In this particular dataset, several independent variables display interdependence, such as the Debt-to-income ratio evincing a strong association with annual income.

In order to enhance prediction accuracy and prevent the issue of overfitting, we intend to construct a random forest model. To facilitate a comparative analysis with random forest, we shall also investigate the gradient boosting approach, which typically yields better results than random forest. It should be emphasized, nonetheless, that this approach is more susceptible to overfitting, a phenomenon that happens when the model is very intricate and closely resembles the training data, leading to poor generalization and reduced accuracy on test data.

***Modelling Approach***

The fundamental concept behind decision tree models involves recursively partitioning the training data based on the values of the input features. To determine the best feature and corresponding value for splitting the data into two subsets at each node of the tree, we commonly use criteria such as the Gini index or cross-entropy. This process continues for each subset unless certain constraints are enforced, such as reaching a maximum depth or having too few observations at each nodes. Subsequently, the decision tree can be used to make predictions for the target variable on a test dataset by traversing the tree from the root node to a leaf node.

Random forest is an ensemble learning method that combines multiple decision trees to form a “forest” of trees. The fundamental objective of the algorithm is to improve model performance and reduce overfitting. It does this by randomly sampling the training data to create new datasets for each individual tree, and randomly selecting a subset of features at each node of the tree to determine the best split. The key idea behind random forest is to create a set of decision trees that are uncorrelated or have a low correlation with each other, which can reduce prediction errors and improve generalization performance. This is achieved by training each decision tree on a different subset of the

data and features. The ensemble approach allows the model to capture complex relationships between the predictors and the outcome, leading to accurate predictions for new data.

Another ensemble learning technique unlike the random forest that incorporates many decision trees but builds from a previous tree is called gradient boosting. In this method, each decision tree is built to correct the errors of the previous one, and the predictions from all trees are combined to produce the final result. This approach can achieve high accuracy on a variety of prediction tasks and can often outperform other traditional machine learning algorithms on a range of prediction problems.

# Data

***Collection***

The dataset obtained from LendingClub contains 96,779 observations and 122 variables documenting the lending transaction status for the first quarter of 2017. LendingClub is the largest American online lending institution, established in 2006, earning a high rating of 4.8 stars (Treece & Tarver, 2023). It mainly offers peer-to-peer lending with loan amounts ranging from $1,000 up to $40,000 and a limit to borrowers having at least 600 FICO credits(Treece & Tarver, 2023). The outcome variable is ‘grade’, with values rated from A to G. Each letter indicates the loan grade for each observation.

On average, the amount of loan requested by the borrowers is $14,858 USD, with a significant portion of loans carrying interest rates exceeding 10%. There is a marked variation in the applicable loan for each of the borrowers giving the standard deviation of $9,396 USD. The loan amount has identical descriptive statistics as the funded amount since each observation records one successful transaction between the lender and the borrower. Out of the total loan applications, 33,698 borrowers received a C loan grade, indicating that these applicants were assessed as moderate-risk borrowers. Conversely, loans with a G-grade classification were found to be extremely rare, with only 508 observations recorded. The lack of availability can be attributed to the fact that high-risk loans are more challenging to secure approval for.

***Construction and Cleaning***

We started by altering the data types of certain variables, for example, changing the type of interest rate from object to float. Upon examining the unprocessed data, it is apparent that a significant number of variables, such as 'id', 'member_id', and 'sec_app_earliest_cr_line', contain a conspicuous number of null values. Therefore, it is necessary to eliminate these columns from the data. Additionally, we removed the column that documented the applicants' occupations, as almost every observation had a distinct job title.

Afterwards, we recoded the remaining columns that contained string observations using dummy variables. Prior to this, we addressed the issue of missing observations, which were recoded with the value 99 to indicate their absence. Furthermore, we intend to randomly divide the observations into training and test sets, allocating 75% of the data to the former and 25% to the latter.

***Estimation - Optimization***

To enhance the efficiency of the decision tree, one approach involves the pre-processing of data, whereby missing observations are inputted and categorical variables are encoded. This has been done through the data cleaning process. Through such pre-processing, noise within the data can be minimized, leading to a more accurate decision tree. Additionally, it is prudent to restrict the maximum

depth of the tree to a reasonable level since the complexity of the model grows exponentially with increasing depth. This growth in complexity may result in overfitting issues, in which the model loses its predictive ability due to becoming overly specialized to the training data. Additionally, the combination of multiple decision trees can enhance overall performance, which can be accomplished through the use of random forest and gradient boosting.

Random forest contains numerous hyperparameters that can be optimized to increase model performance, such as the number of trees, the maximum depth of the trees, and the number of features to consider at each split. Grid search can be employed to determine the optimal hyperparameters and identify the best-performing model. Moreover, correlated features should be eliminated, as they will be assigned equal or similar importance, thereby reducing their overall importance compared to an equivalent tree constructed without correlated variables.

Similarly, multiple hyperparameters, such as the learning rate, the number of trees, and the maximum depth of the trees, can be adjusted to enhance the performance of a gradient boosting model. Employing the grid search technique can assist in identifying the optimal values for these hyperparameters, leading to the selection of the highest-performing model. Similar to what was mentioned above, removing correlated features should be considered since they can compromise the model performance. As gradient boosting models are prone to overfitting, early stopping techniques can be utilized to prevent this outcome.

## Results & Statistics

To start with, we constructed a simple classification tree on the training dataset and predicted the loan grades in the test set. However, given the model's high training and test scores of 1, there may be potential overfitting issues that need to be addressed.

Subsequently, we used the random forest model with 25 trees, which only decreased the training score by a negligible amount while reducing the test score to 0.72. In light of the significant difference between training scores and test scores, there are still issues regarding overfitting in the model. In contrast to test scores, the resulting out-of-bag (OOB) scores, which are derived from "left-over" training data that were not applied to train the model, usually provide an unbiased estimate of the model's performance on unseen data. OOB scores are often slightly lower than test scores since they are based on a smaller amount of data and are therefore more susceptible to randomness. In this case, the OOB score of 0.65 suggests that the model may not be as effective at predicting unseen data as it is at predicting the training data.

In order to enhance the performance of the random forest model, grid-search 5-fold cross-validation can be used. This involves defining a set of parameter values to be tested and determining the ideal combination of parameters that yields the best model performance. By using this technique, the optimal number of trees in the forest is found to be 100, with the maximum number of features considered at each split equal to the square root of the total number of features (max_features = sqrt(n_features)). Subsequently, fitting the new model with the set of best parameters results in a test score of 0.76. This represents a slight improvement over the model without implementing a grid search.

Lastly, a gradient boosting model was generated to be compared with the random forest model. The gradient boosting classifier model achieved a training score of 0.80 and a test score of 0.79 after performing 40 boosting stages and considering a maximum of 50 features at each split. Similar to the random forest model, the model performance was enhanced by optimizing the tuning parameters

through grid search. Compared to the previous model, the updated model had the same best set of parameters as the random forest model, yielding a training score of 0.95 and a test score of 0.94.

## Conclusion

Upon assessing all the implemented classification trees, the most optimal model is the gradient boosting classifier with 100 trees and a maximum number of features considered at each split that is equivalent to the square root of the total number of features. The small gap between the training and test scores suggests that this model is likely not overfitting the training data. This demonstrates that the model has successfully captured the underlying patterns and relationships in the data and can apply them to new data. In comparison to other models that were created, this gradient boosting model demonstrates superior performance, presenting a test score of 0.94.

To conclude, there are several significant factors that contribute to loan grade prediction. The most prominent predictor is the interest rate, the percentage of the principal that borrowers have to pay before the loan due date. Typically, the interest rate on the loan varies with the loan risk, which is evaluated by the loan grade. An individual with a low risk can borrow at a lower interest rate, and an individual with a higher risk can only borrow at a higher interest rate. Therefore, lenders charge a higher interest rate in order to compensate for the risk associated with lower loan grades. Aside from the interest rate of the loan amount, the length of the loan (36 months or 60 months) affects the loan grade as well. As a result, long-term loans tend to have a lower loan grade since lending institutions are likely to incur more losses in the event of default, making the loans riskier. Furthermore, some weaker determinants of loan grades, such as the ratio of balance to high credit/credit limit in the bank account, reflect the borrower's credit score and thus affect the loan evaluation.

## Code

In [59]:
```
import numpy as np
import pandas as pd

from sklearn import ensemble, metrics, model_selection, preprocessing, tree
from matplotlib import pyplot
```

In [60]:
```
data = pd.read_csv('loan.csv',
                   low_memory=False, header=1)
```

In [61]:
```
data = pd.DataFrame(data)
data
```

Out[61]:

| | id | member_id | loan_amnt | funded_amnt | funded_amnt_inv | term | int_rate | installment |
|---|---|---|---|---|---|---|---|---|
| **0** | NaN | NaN | 3600.0 | 3600.0 | 3600.0 | 36 months | 7.49% | 111.97 |
| **1** | NaN | NaN | 15000.0 | 15000.0 | 15000.0 | 60 months | 14.99% | 356.78 |
| **2** | NaN | NaN | 8400.0 | 8400.0 | 8400.0 | 36 months | 11.39% | 276.56 |
| **3** | NaN | NaN | 4000.0 | 4000.0 | 4000.0 | 36 months | 10.49% | 130.00 |
| **4** | NaN | NaN | 6000.0 | 6000.0 | 6000.0 | 36 months | 7.24% | 185.93 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **96776** | NaN | NaN | 10000.0 | 10000.0 | 10000.0 | 36 months | 8.24% | 314.48 |
| **96777** | NaN | NaN | 6325.0 | 6325.0 | 6325.0 | 36 months | 15.99% | 222.34 |
| **96778** | NaN | NaN | 15625.0 | 15625.0 | 15625.0 | 60 months | 28.69% | 493.03 |
| **96779** | Total amount funded in policy code 1: 1437969475 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **96780** | Total amount funded in policy code 2: 520780182 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

96781 rows × 122 columns

# Data Cleasing

In [62]:
```
data.drop(data.tail(2).index, inplace = True)
data.columns
```

Out[62]:
```
Index(['id', 'member_id', 'loan_amnt', 'funded_amnt', 'funded_amnt_inv',
       'term', 'int_rate', 'installment', 'grade', 'sub_grade',
       ...
       'sec_app_earliest_cr_line', 'sec_app_inq_last_6mths',
       'sec_app_mort_acc', 'sec_app_open_acc', 'sec_app_revol_util',
       'sec_app_open_il_6m', 'sec_app_num_rev_accts',
       'sec_app_chargeoff_within_12_mths',
       'sec_app_collections_12_mths_ex_med',
       'sec_app_mths_since_last_major_derog'],
      dtype='object', length=122)
```

In [63]:
```
len(data.columns)
```

Out[63]:
```
122
```

In [65]:
```
data['int_rate'] = data['int_rate'].str[:-1].astype(float)
y = pd.Series(data['grade'], name='loan_grade')
y
```

Out[65]:
```
0        A
1        C
2        B
3        B
4        A
       ..
96774    D
96775    C
96776    B
96777    C
96778    F
Name: loan_grade, Length: 96779, dtype: object
```

In [66]:
```
X = data.drop(columns = ['grade', 'sub_grade'])
```

In [75]:
```
drop_list = []
for i in data.columns:
    if data[i].isnull().sum() > 0:
        if data[i].isnull().sum() > 90000:
            drop_list.append(i)
```

In [68]:
```
X = X.drop(columns = drop_list)
X = X.drop(columns = 'emp_title')
X = X.fillna(99)
X = pd.get_dummies(X)
X.head()
```

Out[68]:

| | loan_amnt | funded_amnt | funded_amnt_inv | int_rate | installment | annual_inc | dti | delinq_2yrs | inq_la |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3600.0 | 3600.0 | 3600.0 | 7.49 | 111.97 | 120000.0 | 18.90 | 0.0 | |
| **1** | 15000.0 | 15000.0 | 15000.0 | 14.99 | 356.78 | 125000.0 | 17.25 | 0.0 | |
| **2** | 8400.0 | 8400.0 | 8400.0 | 11.39 | 276.56 | 50000.0 | 15.63 | 0.0 | |
| **3** | 4000.0 | 4000.0 | 4000.0 | 10.49 | 130.00 | 50000.0 | 33.61 | 1.0 | |
| **4** | 6000.0 | 6000.0 | 6000.0 | 7.24 | 185.93 | 125000.0 | 9.25 | 0.0 | |

5 rows × 2771 columns

# Descriptive Analysis

In [69]:
```
X.describe()
```

Out[69]:

| | loan_amnt | funded_amnt | funded_amnt_inv | int_rate | installment | annual_inc | |
|---|---|---|---|---|---|---|---|
| **count** | 96779.000000 | 96779.000000 | 96779.000000 | 96779.000000 | 96779.000000 | 9.677900e+04 | 96779 |
| **mean** | 14858.279947 | 14858.279947 | 14853.928022 | 13.370696 | 450.885696 | 8.276104e+04 | 2 |
| **std** | 9396.273231 | 9396.273231 | 9393.128515 | 5.069448 | 283.501239 | 2.149442e+05 | 185 |
| **min** | 1000.000000 | 1000.000000 | 1000.000000 | 5.320000 | 30.120000 | 0.000000e+00 | -1 |
| **25%** | 7800.000000 | 7800.000000 | 7800.000000 | 10.490000 | 244.300000 | 4.900000e+04 | 12 |
| **50%** | 12000.000000 | 12000.000000 | 12000.000000 | 12.740000 | 372.710000 | 7.000000e+04 | 17 |
| **75%** | 20000.000000 | 20000.000000 | 20000.000000 | 15.990000 | 602.300000 | 9.950000e+04 | 24 |
| **max** | 40000.000000 | 40000.000000 | 40000.000000 | 30.990000 | 1715.420000 | 6.100000e+07 | 9999 |

8 rows × 2771 columns

In [70]:
```
y.describe()
```

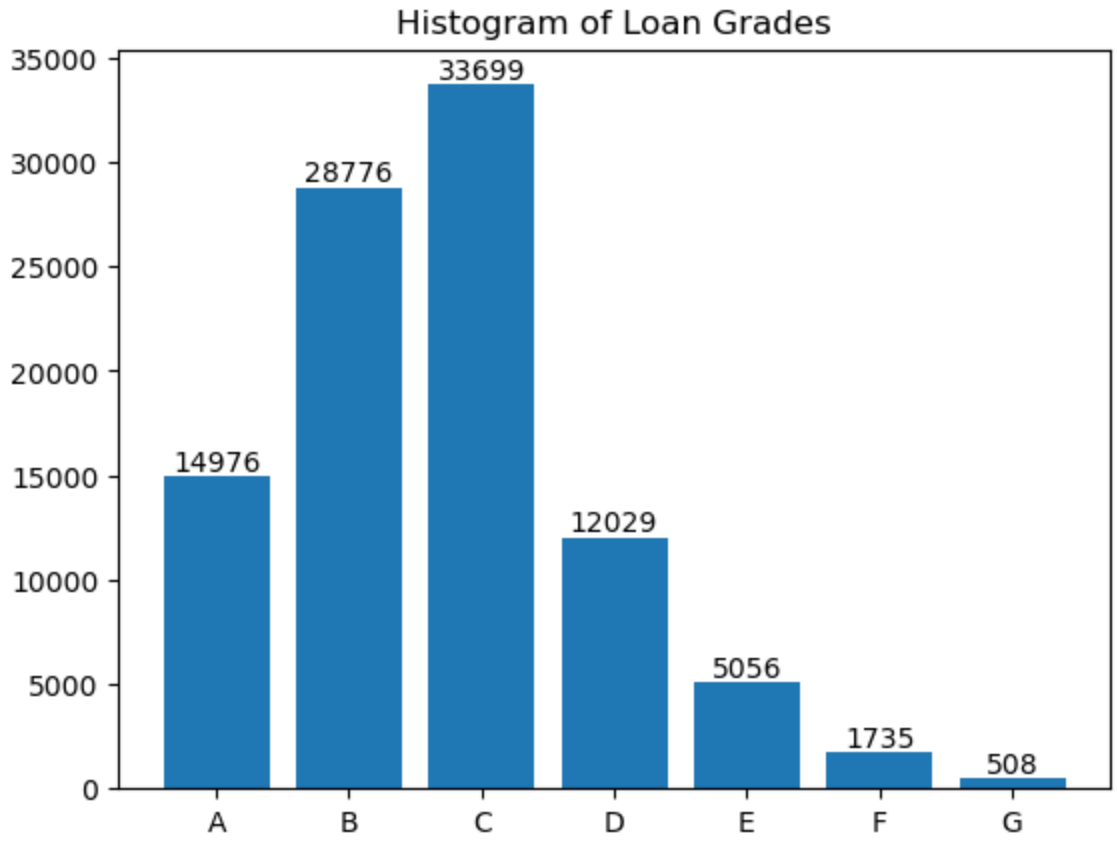
Out[70]:
```
count     96779
unique        7
top           C
freq      33699
Name: loan_grade, dtype: object
```

In [71]:
```
y.value_counts()
```

Out[71]:
```
C    33699
B    28776
```

```
A    14976
D    12029
E     5056
F     1735
G      508
Name: loan_grade, dtype: int64
```

In [72]:
```
d = {}
for i in ['A','B','C','D','E','F','G']:
    d[i] = sum(y == i)

grade = list(d.keys())
value = list(d.values())
fig, ax = pyplot.subplots()
bars = ax.bar(grade, value)
ax.bar_label(bars)
pyplot.title("Histogram of Loan Grades")
pyplot.show()
```




In [15]:
```
# Training and Test Splits
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y,
                                                                    train_size=.75,
                                                                    test_size=.25,
                                                                    shuffle=True,
                                                                    random_state=1)
```

# Model

## Decision Tree

In [16]:
```
tree_model = tree.DecisionTreeClassifier(random_state=1)
tree_model.fit(X_train, y_train)
```

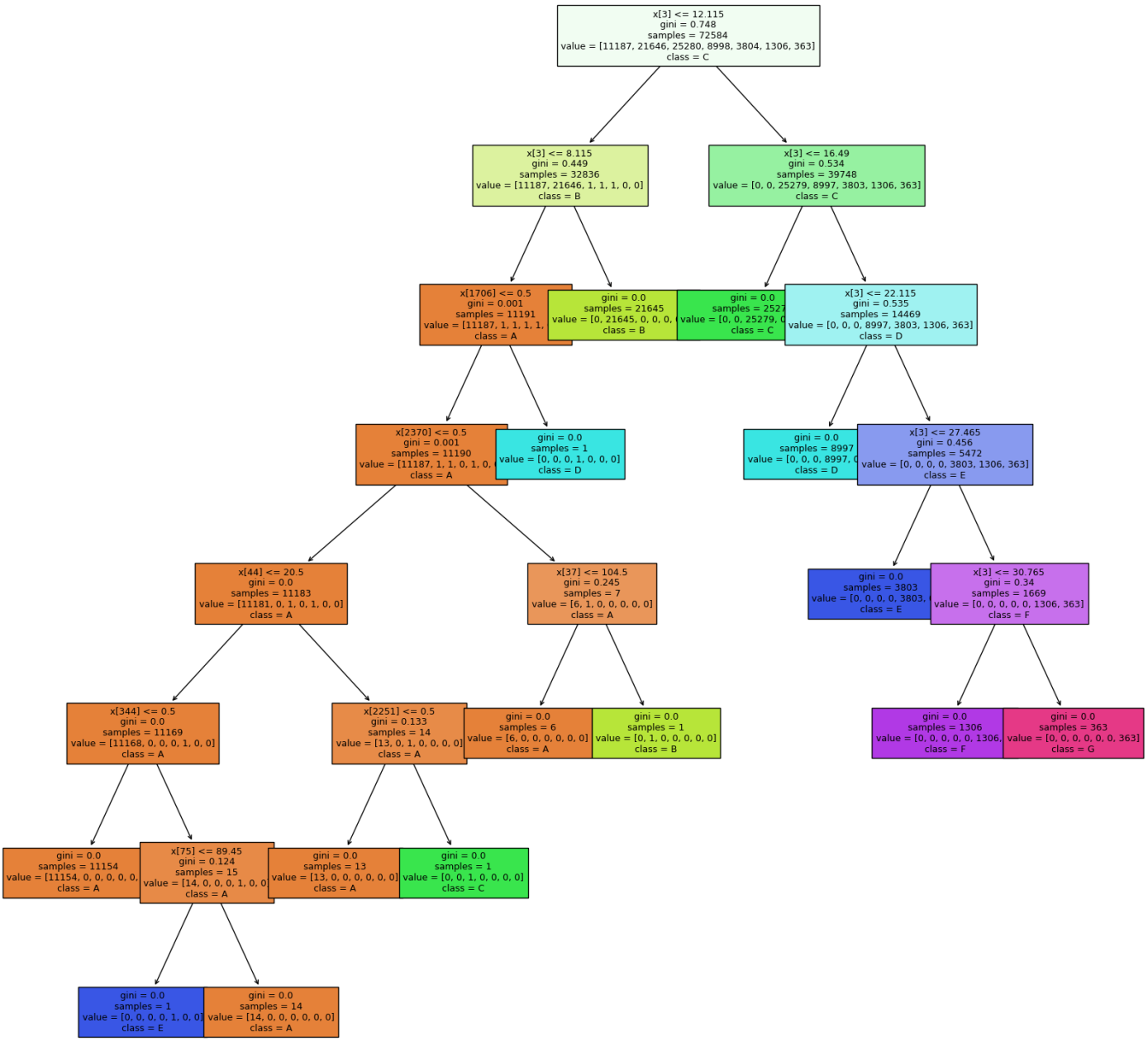
```
print('Training Score:', tree_model.score(X_train, y_train), "\n")
print('Test Score:', tree_model.score(X_test, y_test), "\n")
```

```
Training Score: 1.0

Test Score: 1.0
```

In [17]:
```
print(metrics.classification_report(y_test, tree_model.predict(X_test)))
```

```
              precision    recall  f1-score   support

           A       1.00      1.00      1.00      3789
           B       1.00      1.00      1.00      7130
           C       1.00      1.00      1.00      8419
           D       1.00      1.00      1.00      3031
           E       1.00      1.00      1.00      1252
           F       1.00      1.00      1.00       429
           G       1.00      1.00      1.00       145

    accuracy                           1.00     24195
   macro avg       1.00      1.00      1.00     24195
weighted avg       1.00      1.00      1.00     24195
```

In [18]:
```
class_names = ['A','B','C','D','E','F','G']
pyplot.figure(figsize=(20,20))
tree.plot_tree(
    tree_model,
    filled=True,
    class_names=class_names,
    fontsize=9)
pyplot.show()
```



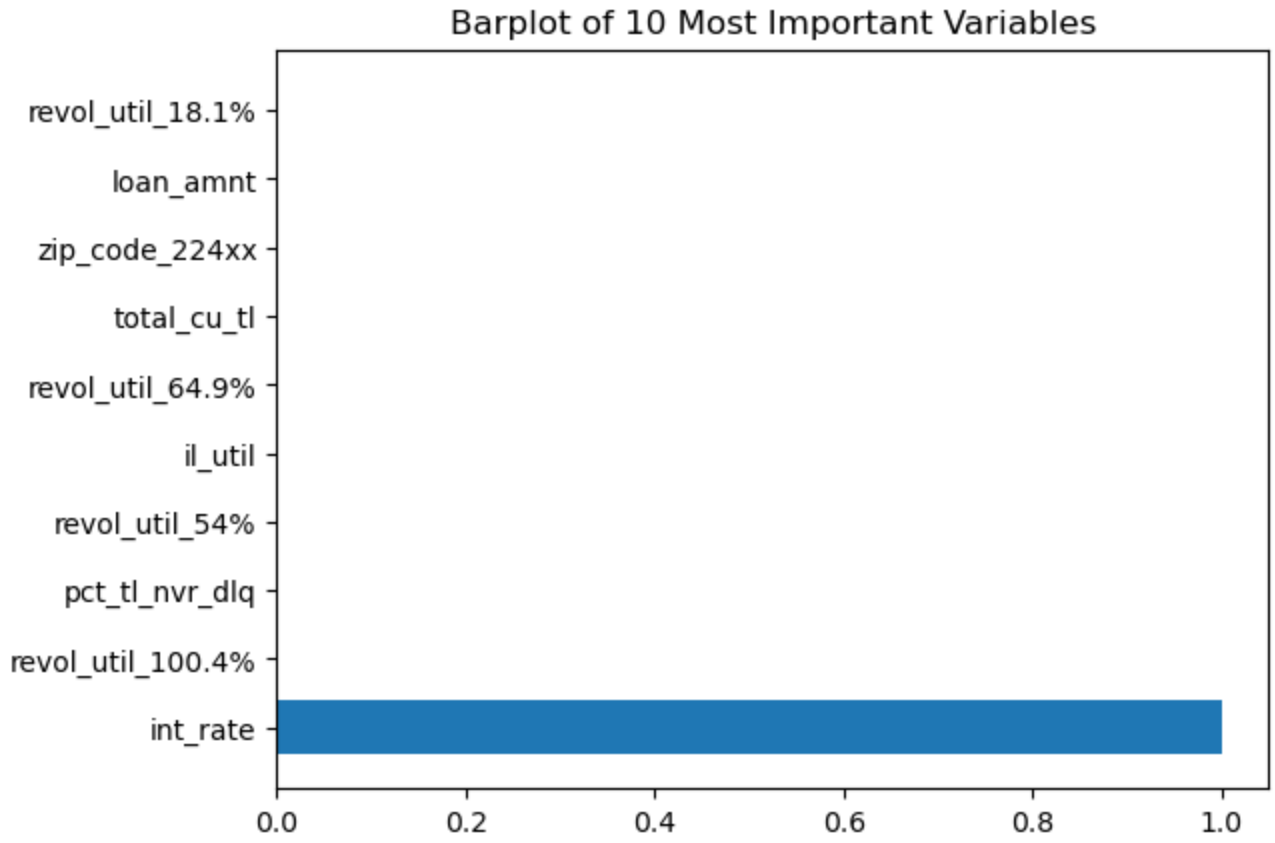

In [19]:
```
var = pd.DataFrame({"Importance": tree_model.feature_importances_.tolist(),
                    "Feature Name": tree_model.feature_names_in_.tolist()})

# 10 Most Important Variables
var.sort_values("Importance", ascending=False).head(10)
```

Out[19]:

| | Importance | Feature Name |
|---|---|---|
| **3** | 0.999853 | int_rate |
| **1706** | 0.000037 | revol_util_100.4% |
| **75** | 0.000034 | pct_tl_nvr_dlq |
| **2251** | 0.000034 | revol_util_54% |
| **37** | 0.000032 | il_util |
| **2370** | 0.000005 | revol_util_64.9% |
| **44** | 0.000003 | total_cu_tl |
| **344** | 0.000002 | zip_code_224xx |

| | | |
|---|---|---|
| **0** | 0.000000 | loan_amnt |
| **1852** | 0.000000 | revol_util_18.1% |

In [20]:
```
df = var.sort_values("Importance", ascending=False).head(10)
pyplot.barh(df["Feature Name"], df["Importance"])
pyplot.title("Barplot of 10 Most Important Variables")
pyplot.show()
```




## Random Forest

In [21]:
```
random_forest = ensemble.RandomForestClassifier(n_estimators=25,
                                                random_state=1,
                                                oob_score=True)
random_forest.fit(X_train, y_train)

print('Training Score:', random_forest.score(X_train, y_train), "\n")
print('Test Score:', random_forest.score(X_test, y_test), "\n")
```

```
Training Score: 0.9998622285903229

Test Score: 0.7243232072742302
```

In [23]:
```
# Out-of-Bag Score
random_forest.oob_score_
```

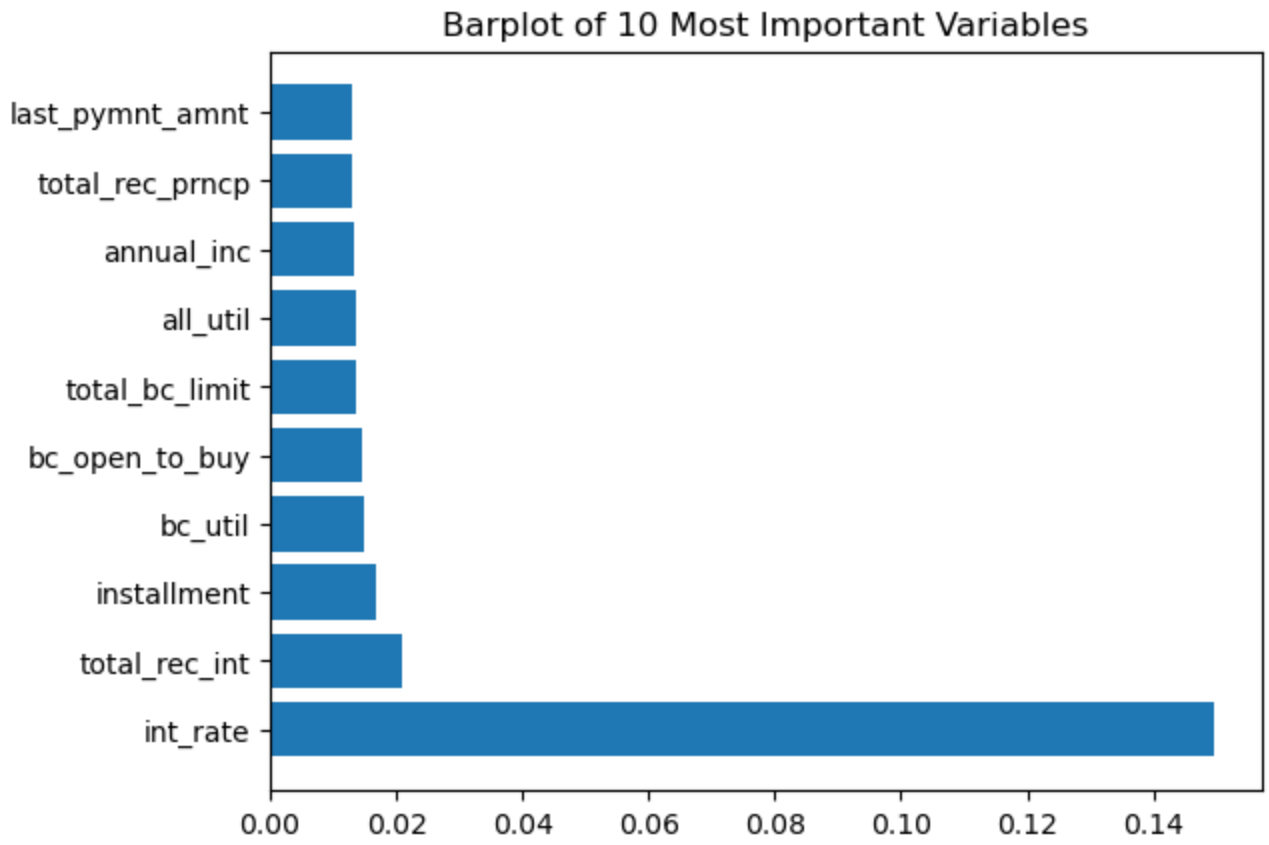
Out[23]:
```
0.6491375509754216
```

In [24]:
```
var2 = pd.DataFrame({"Importance": random_forest.feature_importances_.tolist(),
                     "Feature Name": random_forest.feature_names_in_.tolist()})

# 10 Most Important Variables
var2.sort_values("Importance", ascending=False).head(10)
```

Out[24]:

| | Importance | Feature Name |
|---|---|---|
| **3** | 0.149774 | int_rate |
| **20** | 0.020933 | total_rec_int |
| **4** | 0.016932 | installment |
| **49** | 0.014923 | bc_util |
| **48** | 0.014698 | bc_open_to_buy |
| **81** | 0.013639 | total_bc_limit |
| **41** | 0.013555 | all_util |
| **5** | 0.013223 | annual_inc |
| **19** | 0.013077 | total_rec_prncp |
| **24** | 0.012970 | last_pymnt_amnt |

In [25]:
```
df2 = var2.sort_values("Importance", ascending=False).head(10)
pyplot.barh(df2["Feature Name"], df2["Importance"])
pyplot.title("Barplot of 10 Most Important Variables")
pyplot.show()
```




In [26]:
```
parameters = {
    'n_estimators': [20,50,70,100],
    'max_features': [20,30,40,50,'sqrt'],
}

rf = ensemble.RandomForestClassifier(random_state=1)
grid = model_selection.GridSearchCV(rf, param_grid=parameters)
grid.fit(X_train, y_train)
grid.best_params_
```

Out[26]:
```
{'max_features': 'sqrt', 'n_estimators': 100}
```

In [27]:
```
random_forest2 = ensemble.RandomForestClassifier(max_features='sqrt',
                                                 n_estimators=100,
                                                 random_state=1)
random_forest2.fit(X_train, y_train)

print('Training Score:', random_forest2.score(X_train, y_train), "\n")
print('Test Score:', random_forest2.score(X_test, y_test), "\n")
```

```
Training Score: 1.0

Test Score: 0.7632568712543915
```

## Gradient Boosting

In [29]:
```
gb = ensemble.GradientBoostingClassifier(n_estimators=40,max_features=50,random_state=1)
gb.fit(X_train, y_train)

print('Training Score:', gb.score(X_train, y_train), "\n")
print('Test Score:', gb.score(X_test, y_test), "\n")
```

```
Training Score: 0.7997768103163232

Test Score: 0.7934283942963423
```

In [30]:
```
# Optimise the tuning parameters
gb2 = ensemble.GradientBoostingClassifier(random_state=1)

param_grid = {
              "n_estimators": [40,50,60,100],
              "max_features": [50,'sqrt','log2']}

grid_search = model_selection.GridSearchCV(gb2, param_grid=param_grid)
grid_search.fit(X_train, y_train)
grid_search.best_params_
```

Out[30]:
```
{'max_features': 'sqrt', 'n_estimators': 100}
```

In [31]:
```
gb2 = ensemble.GradientBoostingClassifier(n_estimators=100,
                                          max_features='sqrt',
                                          random_state=1)
gb2.fit(X_train, y_train)

print('Training Score:', gb2.score(X_train, y_train), "\n")
print('Test Score:', gb2.score(X_test, y_test), "\n")
```

```
Training Score: 0.9478672985781991

Test Score: 0.9380450506302955
```

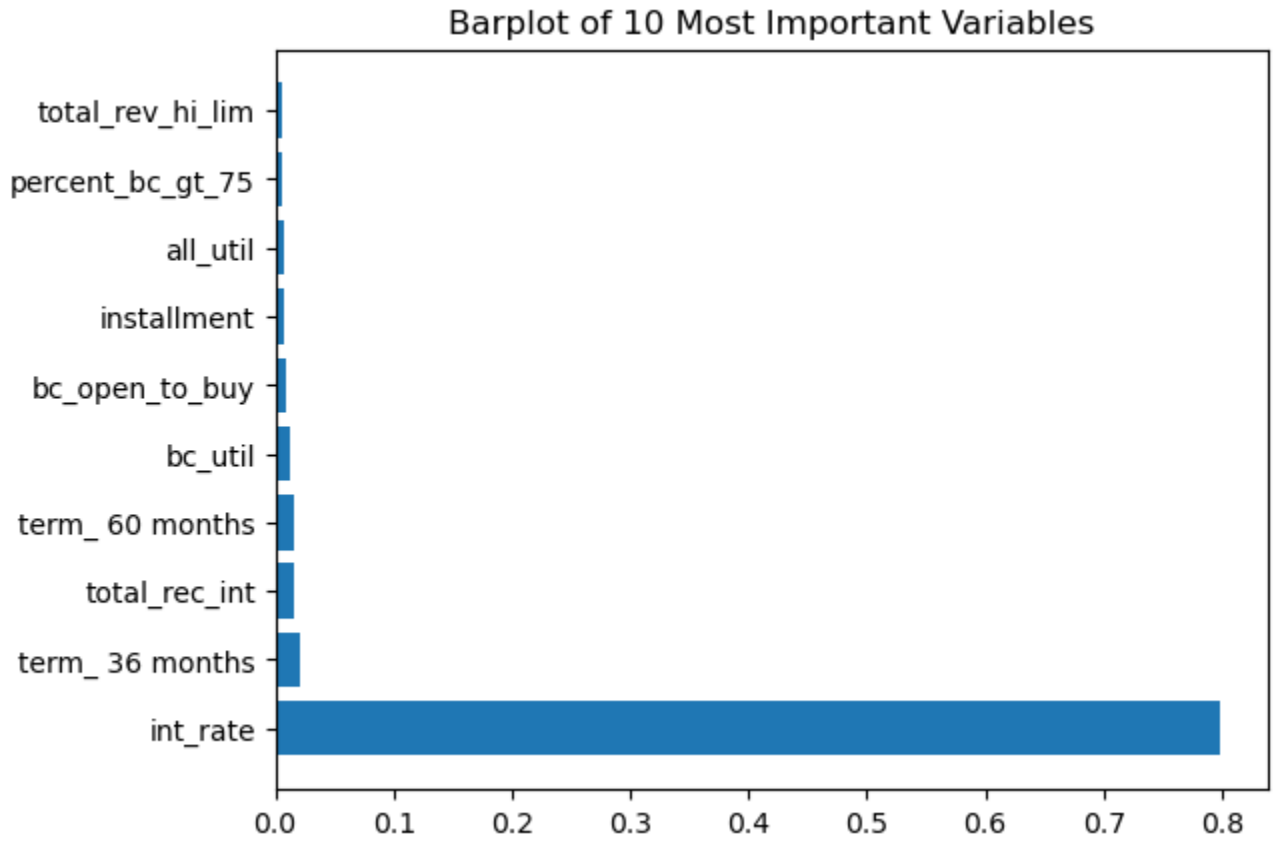
In [32]:
```
var3 = pd.DataFrame({"Importance": gb2.feature_importances_.tolist(),
                     "Feature Name": gb2.feature_names_in_.tolist()})

# 10 Most Important Variables
var3.sort_values("Importance", ascending=False).head(10)
```

Out[32]:

| | Importance | Feature Name |
|---|---|---|
| **3** | 0.798356 | int_rate |
| **83** | 0.020272 | term_ 36 months |
| **20** | 0.015526 | total_rec_int |

| | | |
|---|---|---|
| **84** | 0.015483 | term_ 60 months |
| **49** | 0.012779 | bc_util |
| **48** | 0.008987 | bc_open_to_buy |
| **4** | 0.007402 | installment |
| **41** | 0.006817 | all_util |
| **76** | 0.005300 | percent_bc_gt_75 |
| **42** | 0.005167 | total_rev_hi_lim |

In [33]:
```
df3 = var3.sort_values("Importance", ascending=False).head(10)
pyplot.barh(df3["Feature Name"], df3["Importance"])
pyplot.title("Barplot of 10 Most Important Variables")
pyplot.show()
```



# Reference

- Armstrong, B. (2019, November 6). About loan grading. Real Estate News & Investing Blog. Retrieved March 20, 2023, from https://blog.groundfloor.com/groundfloorblog/about-loan-grading
- Bai, Y., & Zha, D. (2022). Commercial Bank Credit Grading Model Using Genetic Optimization Neural Network and Cluster Analysis. Computational Intelligence and Neuroscience, 2022, 4796075–11. https://doi.org/10.1155/2022/4796075
- Kenton, W. (2021, May 19). Loan grading definition. Investopedia. Retrieved March 20, 2023, from https://www.investopedia.com/terms/l/loan-grading.asp
- Pirgaip, B., & Hepsen, A. (2018). Loan-to-value policy: Evidence from Turkish dual banking system. International Journal of Islamic and Middle Eastern Finance and Management, 11(4), 631–649. https://doi.org/10.1108/imefm-08-2017-0208
- Treece, K., & Tarver, J. (2023, February 27). LendingClub Personal Loans Review 2023. Forbes. Retrieved March 20, 2023, from https://www.forbes.com/advisor/personal-loans/lendingclub-personal-loans-review/#:~:text=LendingClub%20Corporation%20has%20a%204.43,and%20funds%20were%20received%20quickly.