Tanvi Amlani

Random Forest Project

For this project we will be exploring publicly available data from LendingClub.com. Lending Club connects people who need money (borrowers) with people who have money (investors). Hopefully, as an investor you would want to invest in people who showed a profile of having a high probability of paying you back. We will try to create a model that will help predict this.

Lending club had a very interesting year in 2016, so let's check out some of their data and keep the context in mind. This data is from before they even went public.

We will use lending data from 2007-2010 and be trying to classify and predict whether or not the borrower paid back their loan in full. I uploaded the data in the csv file.

Here are what the columns represent:

- credit.policy: 1 if the customer meets the credit underwriting criteria of LendingClub.com, and 0 otherwise.
- purpose: The purpose of the loan (takes values "credit_card", "debt_consolidation", "educational", "major_purchase", "small_business", and "all_other").
- int.rate: The interest rate of the loan, as a proportion (a rate of 11% would be stored as 0.11). Borrowers judged by LendingClub.com to be more risky are assigned higher interest rates.
- installment: The monthly installments owed by the borrower if the loan is funded.
- log.annual.inc: The natural log of the self-reported annual income of the borrower.
- dti: The debt-to-income ratio of the borrower (amount of debt divided by annual income).
- fico: The FICO credit score of the borrower.
- days.with.cr.line: The number of days the borrower has had a credit line.
- revol.bal: The borrower's revolving balance (amount unpaid at the end of the credit card billing cycle).
- revol.util: The borrower's revolving line utilization rate (the amount of the credit line used relative to total credit available).
- inq.last.6mths: The borrower's number of inquiries by creditors in the last 6 months.
- delinq.2yrs: The number of times the borrower had been 30+ days past due on a payment in the past 2 years.
- pub.rec: The borrower's number of derogatory public records (bankruptcy filings, tax liens, or judgments).

Import Libraries

Import the usual libraries for pandas and plotting.

```
@author: Tanvi
"""
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Get the Data

Use pandas to read loan_data.csv as a dataframe called loans.

In [2]:

Check out the info(), head(), and describe() methods on loans.

```
loans = pd.read_csv(r"C:\Users\Tanvi\OneDrive\Documents\Python\loan_data.csv")
loans.info()
loans.describe()
print(loans.head())
```

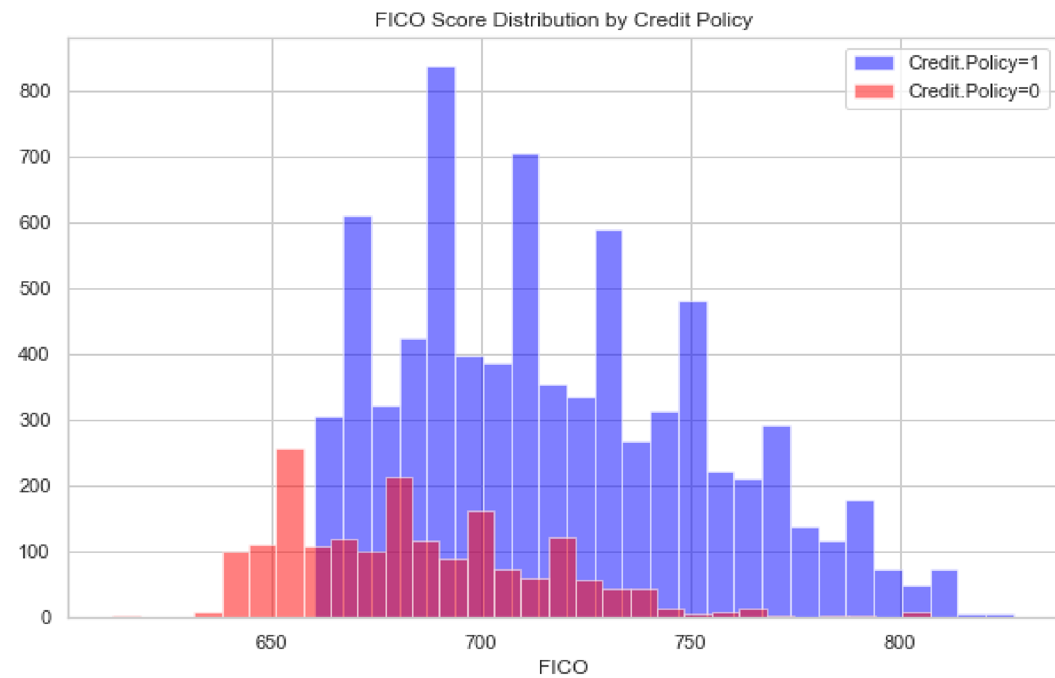
```
IPdb [6]: runfile('C:/Users/Tanvi/.spyder-py3/untitled1.py', wdir='C:/Users/Tanvi/.spyder-py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   credit.policy      9578 non-null   int64
 1   purpose            9578 non-null   object
 2   int.rate           9578 non-null   float64
 3   installment        9578 non-null   float64
 4   log.annual.inc     9578 non-null   float64
 5   dti                9578 non-null   float64
 6   fico               9578 non-null   int64
 7   days.with.cr.line  9578 non-null   float64
 8   revol.bal          9578 non-null   int64
 9   revol.util         9578 non-null   float64
 10  inq.last.6mths     9578 non-null   int64
 11  delinq.2yrs        9578 non-null   int64
 12  pub.rec            9578 non-null   int64
 13  not.fully.paid     9578 non-null   int64
dtypes: float64(6), int64(7), object(1)
memory usage: 1.0+ MB
```

```
       credit.policy     int.rate  ...      pub.rec  not.fully.paid
count    9578.000000  9578.000000  ...  9578.000000     9578.000000
mean        0.804970     0.122640  ...     0.062122        0.160054
std         0.396245     0.026847  ...     0.262126        0.366676
min         0.000000     0.060000  ...     0.000000        0.000000
25%         1.000000     0.103900  ...     0.000000        0.000000
50%         1.000000     0.122100  ...     0.000000        0.000000
75%         1.000000     0.140700  ...     0.000000        0.000000
max         1.000000     0.216400  ...     5.000000        1.000000

[8 rows x 13 columns]
   credit.policy             purpose  ...  pub.rec  not.fully.paid
0              1  debt_consolidation  ...        0               0
1              1         credit_card  ...        0               0
2              1  debt_consolidation  ...        0               0
3              1  debt_consolidation  ...        0               0
4              1         credit_card  ...        0               0

[5 rows x 14 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 19 columns):
```

Exploratory Data Analysis

Create a histogram of two FICO distributions on top of each other, one for each credit.policy outcome.

```
# Plot histograms
plt.figure(figsize=(10,6))
loans[loans['credit.policy']==1]['fico'].hist(alpha=0.5,color='blue',
                                              bins=30,label='Credit.Policy=1')
loans[loans['credit.policy']==0]['fico'].hist(alpha=0.5,color='red',
                                              bins=30,label='Credit.Policy=0')
plt.legend()
plt.xlabel('FICO')
plt.title('FICO Score Distribution by Credit Policy')
plt.grid(True)
plt.show()
```

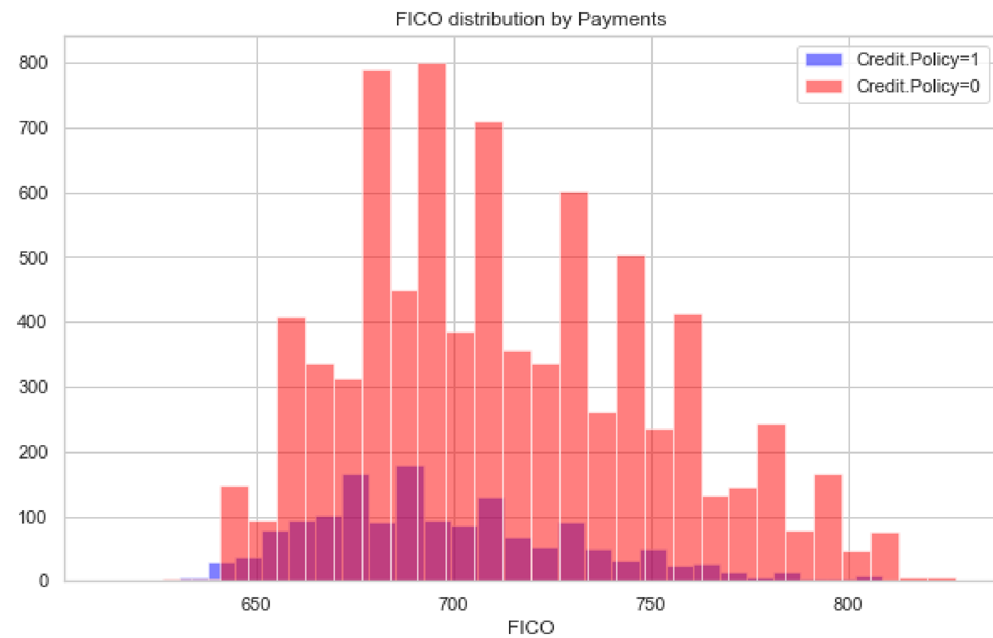


This histogram shows the distribution of **FICO scores** split by **Credit Policy** (1 = meets credit policy, 0 = does not meet credit policy). Here are the main insights:

- **Higher FICO Scores Are Linked to Meeting Credit Policy**: The blue bars (Credit.Policy=1) are more frequent in the higher FICO score ranges (around 680–800+), indicating that applicants with better credit scores are more likely to meet the lending institution's credit policy.
- **Lower FICO Scores Tend to Violate Credit Policy**: The red bars (Credit.Policy=0) dominate in the lower FICO ranges (around 640–700), suggesting that individuals with lower credit scores are often denied credit or flagged by the policy.
- **Clear Separation Trend**: There's a visible separation between the two groups—Credit.Policy=1 applicants are generally clustered toward the higher end of FICO scores, while Credit.Policy=0 applicants taper off significantly after 700, highlighting how creditworthiness is largely determined by FICO score thresholds.

Create a similar figure, except this time select by the not.fully.paid column

```
plt.figure(figsize=(10,6))
loans[loans['not.fully.paid']==1]['fico'].hist(alpha=0.5,color='blue',
                                              bins=30,label='Credit.Policy=1')
loans[loans['not.fully.paid']==0]['fico'].hist(alpha=0.5,color='red',
                                              bins=30,label='Credit.Policy=0')
plt.legend()
plt.xlabel('FICO')
plt.title('FICO distribution by Payments')
plt.grid(True)
plt.show()
```

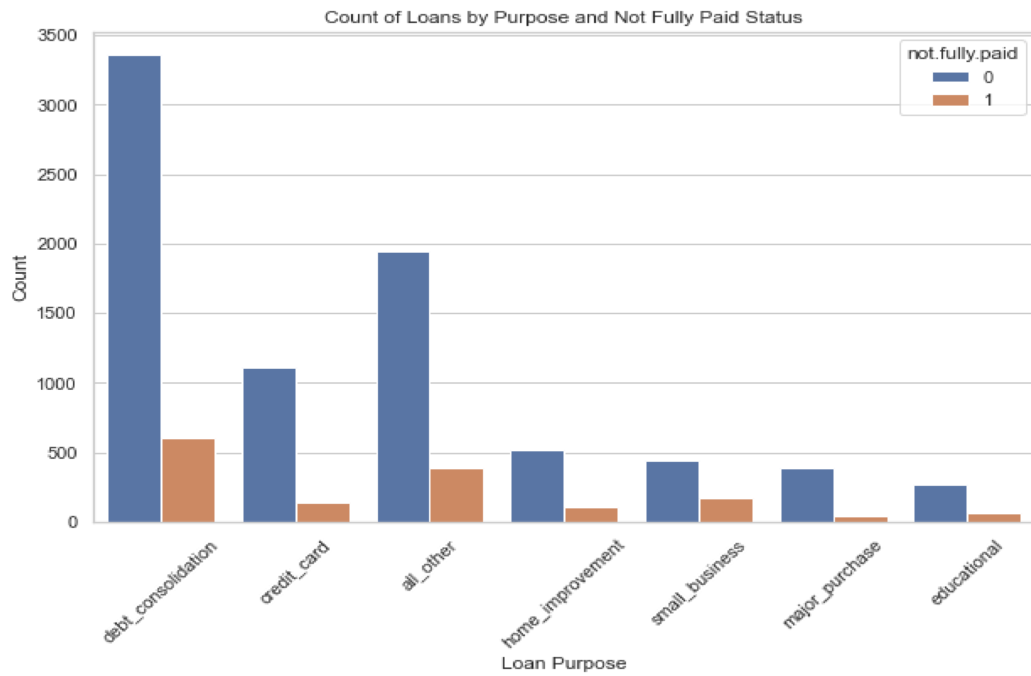


This histogram visualizes the **FICO score distribution by payment status**, with the color coding based on the **Credit Policy** (Credit.Policy=1 for approved applicants, Credit.Policy=0 for those who didn't meet the policy). Below are the key insights:

- The red bars (Credit.Policy=0) dominate across almost the entire range of FICO scores—from around 640 to 800. This suggests that many borrowers **did not meet the credit policy**, even if they had moderately good FICO scores.
- While we might expect that higher FICO scores mean better creditworthiness, the presence of red bars even at 750–800 FICO shows that **FICO alone is not a sufficient predictor** for meeting credit policy or for payment behavior.
- The purple bars (Credit.Policy=1) are much smaller, meaning **only a small portion of borrowers** meet the policy and pay as expected. This could reflect strict approval standards or risk-averse lending criteria.
- This chart highlights the **complexity of credit risk assessment**—even borrowers with decent FICO scores can fail to meet internal policies, possibly due to other factors like high debt-to-income ratios, employment status, or loan purpose.

Create a countplot using seaborn showing the counts of loans by purpose, with the color hue defined by not.fully.paid

```
plt.figure(figsize=(10,6))
sns.countplot(data=loans, x='purpose', hue='not.fully.paid')
plt.title('Count of Loans by Purpose and Not Fully Paid Status')
plt.xlabel('Loan Purpose')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

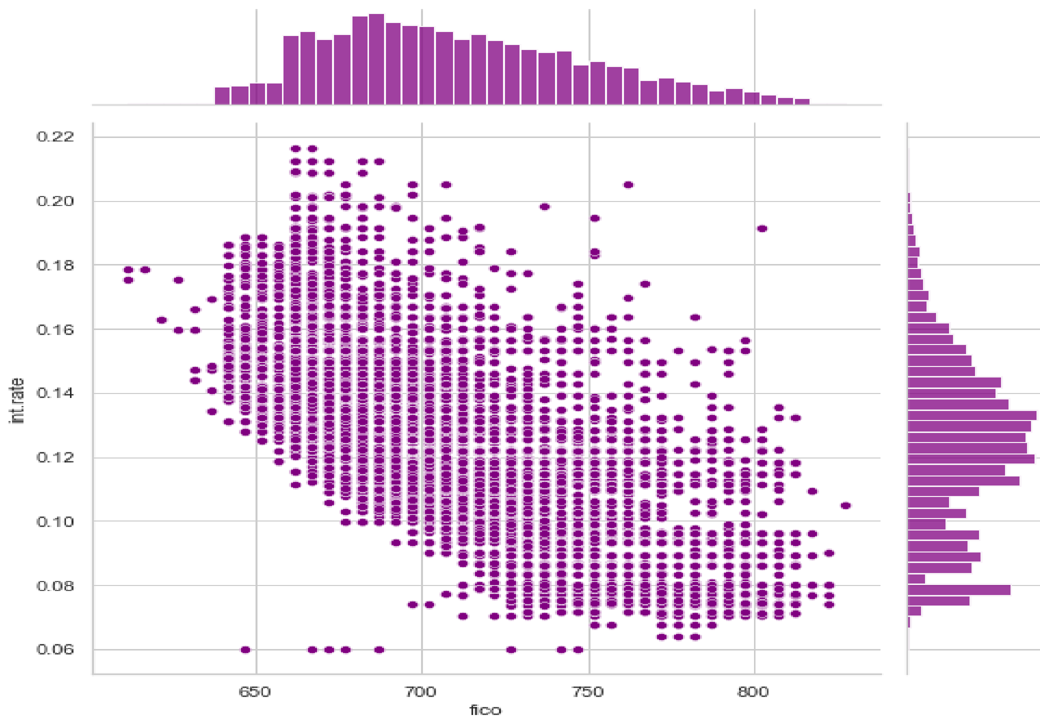


This bar chart shows the **count of loans by their purpose** and distinguishes whether they were **fully paid (not.fully.paid = 0)** or **not fully paid (not.fully.paid = 1)**. Here's what we can learn:

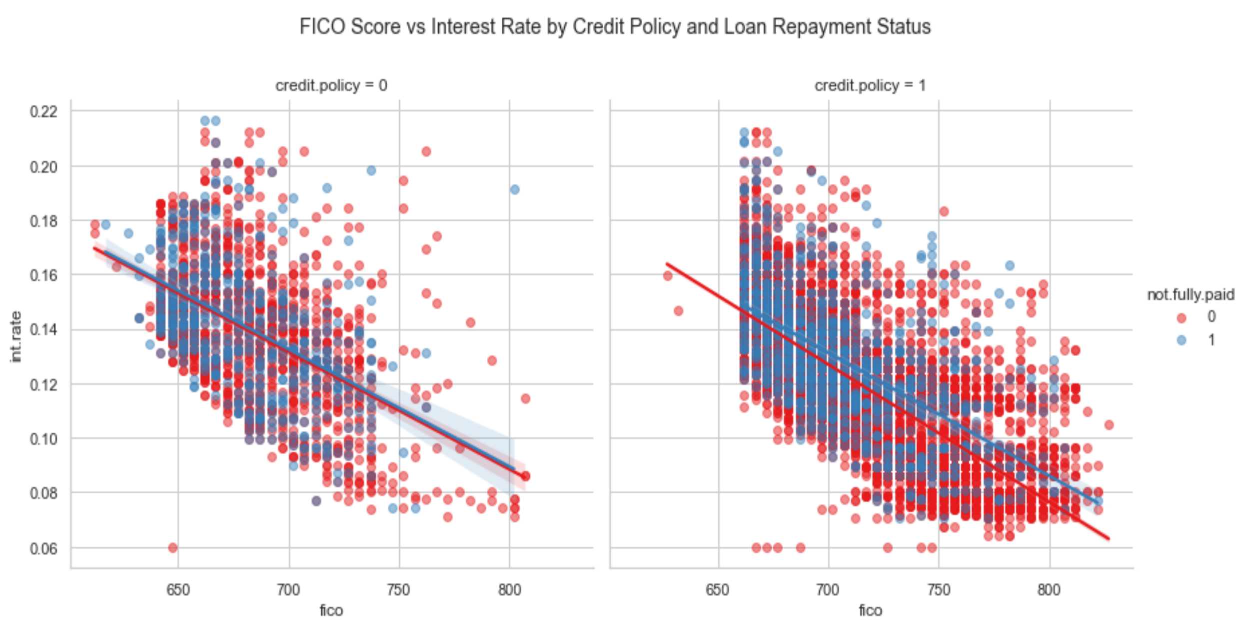
- **Debt Consolidation Is the Most Common Loan Purpose**: It has the highest loan count overall, with a significant portion not fully paid, indicating that borrowers seeking debt consolidation pose higher repayment risks.
- **Default Risk Varies by Purpose**: Categories like small business and major purchase have relatively high default proportions, even if the total number of loans is smaller. This suggests purpose-specific risk patterns, important for lenders when assessing applications.
- **Educational Loans Have the Fewest Defaults**: Although less frequent overall, educational loans show a smaller not-fully-paid count, indicating lower observed default rates in this category, which may reflect better borrower profiles or repayment structures.
-

the trend between FICO score and interest rate. Recreate the following jointplot

```
sns.jointplot(data=loans, x='fico', y='int.rate', kind='scatter', height=8, color='purple')

# Show the plot
plt.show()
```



Create the lmplots to see if the trend differed between not.fully.paid and credit.policy



Now set up the data for classification for Random forest model and decision tree model by utilizing Sickit-learn libraries

"Purpose column" as categorical that means we have to transform them using dummy variables so sklearn will be able to understand them.

one clean step using pd.get_dummies

use pd.get_dummies(loans,columns=cat_Bank_loans,drop_first=True) to create a fixed larger dataframe that has new feature columns with dummy variables. Set this dataframe as final_data.

Train Test Split

split the data into a training set and a testing set!

Use sklearn to split the data into a training set and a testing set

```
Bank_loans = ['purpose']
final_data= pd.get_dummies(loans, columns = Bank_loans, drop_first = True)

final_data.info()

from sklearn.model_selection import train_test_split
X = final_data.drop('not.fully.paid',axis=1)
y = final_data['not.fully.paid']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, random_state = 101)

from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)

#Predictions and Evaluation of Decision Tree BY Classification and Confusion matrix
predictions = dtree.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test,predictions))
print(confusion_matrix(y_test,predictions))
```

Decision tree Classification Model and Confusion Matrix

```
              precision    recall  f1-score   support

           0       0.85      0.83      0.84      2831
           1       0.20      0.22      0.21       522

    accuracy                           0.74      3353
   macro avg       0.53      0.53      0.53      3353
weighted avg       0.75      0.74      0.75      3353

[[2363  468]
 [ 405  117]]
```

0 = Not Fully Paid = NO (the borrower **did** pay back the loan)

1 = Not Fully Paid = YES (the borrower **did NOT** fully pay back the loan)

Metric	**Paid (0)**	**Defaulted (1)**
Precision	0.85 → good	0.20 → poor
Recall	0.83 → good	0.22 → poor
F1-score	0.84	0.21
Support	2831 (paid loans)	522 (defaults)

Predicted | Paid | Defaulted

Actually Paid | 2363 | 468

Actually Defaulted | 405 | 117

2363 borrowers actually paid and were correctly predicted → True Negatives

468 borrowers paid but model wrongly predicted they would default → False Positives

405 borrowers defaulted but model predicted they would pay → False Negatives

117 borrowers defaulted and model correctly predicted it → True Positives

- **Model is good at predicting those who will repay loans** (class 0). That's why it shows high accuracy (0.74).
- **Model performs poorly in predicting defaults (class 1)** — only catches 22% of actual defaulters. That's dangerous for lenders!
- **Many defaults are missed (405 out of 522)**, which could result in significant financial loss.

```python
#Random Forest model
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 300)
rfc.fit(X_train, y_train)

rfc_pred = rfc.predict(X_test)
print(classification_report(y_test,rfc_pred))
print(confusion_matrix(y_test,rfc_pred))
```

Random Forest Classification Model and Confusion Matrix

```
              precision    recall  f1-score   support

           0       0.85      0.99      0.91      2831
           1       0.42      0.02      0.04       522

    accuracy                           0.84      3353
   macro avg       0.63      0.51      0.48      3353
weighted avg       0.78      0.84      0.78      3353

[[2816   15]
 [ 511   11]]
```

- High Accuracy, But Poor Recall for Class 1: Overall accuracy is 84%, but recall for class 1 (loans not fully paid) is only 0.01 (1%). That means the model only correctly identified 1% of the actual unpaid loans.
- Severe Class Imbalance Handling Issue: Precision for class 1 is 0.43, but due to extremely low recall, the F1-score is just 0.02. This is a classic case of a model being biased toward the majority class (fully paid).