Cheyenne Bennmarie,

Mayuresh Budukh,

Edwin Jeyakumar

Hridayraj K Modi,

Yuanhui Jiang,

Abhishek Prataap Raghuvir

a.(i)

In [6]:
```
import pandas as pd

# Load the dataset
data = pd.read_csv("LendingClub_LoanStats3a_v12.csv")

# Filter rows where loan_status is "Fully Paid" or "Charged Off"
data = data[data['loan_status'].isin(['Fully Paid', 'Charged Off'])]

# Define the new variable Default
data['Default'] = (data['loan_status'] == 'Charged Off').astype(int)
```

```
/var/folders/rf/lmhdc33x4ys2xw1wmnlj6rfh0000gn/T/ipykernel_32082/3383880034.py:4:
DtypeWarning: Columns (21,24,29,31) have mixed types. Specify dtype option on impo
rt or set low_memory=False.
  data = pd.read_csv("LendingClub_LoanStats3a_v12.csv")
```

a.(ii)

In [4]:
```
# Calculate the total number of loans
total_loans = len(data)

# Calculate the total number of defaults
total_defaults = data['Default'].sum()

# Calculate the default rate
default_rate = total_defaults / total_loans

print("Average default rate in the sample:", round(default_rate*100,3),"%")
```

```
Average default rate in the sample: 14.353 %
```

b.(i)

In [ ]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from scipy.stats import chi2
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt

#read in data
data=pd.read_csv('C:\\Users\\Downloads\\LendingClub_LoanStats3a_v12.csv')
data=data[data["loan_status"].apply(lambda x:x=='Fully Paid' or x=='Charged Off')]
```

```
#convert loan_status to one hot coding
data['Default']=data["loan_status"].apply(lambda x:1 if x=='Charged Off' else 0)

#convert grade to float (A is 7,G is 1)
def grade_to_float(x):
    if x=='A':
        return 7
    elif x=='B':
        return 6
    elif x=='C':
        return 5
    elif x=='D':
        return 4
    elif x=='E':
        return 3
    elif x=='F':
        return 2
    elif x=='G':
        return 1
data['grade']=data["grade"].apply(grade_to_float)
data['cons']=1

model=sm.Logit(data['Default'],data[["grade",'cons']]).fit()
print(model.summary())
```

```
Regression Result:

                           Logit Regression Results
==============================================================================
Dep. Variable:                Default   No. Observations:
39412
Model:                          Logit   Df Residuals:
39410
Method:                           MLE   Df Model:
1
Date:                Mon, 22 Apr 2024   Pseudo R-squ.:
0.04304
Time:                        18:21:57   Log-Likelihood:
-15514.
converged:                       True   LL-Null:
-16211.
Covariance Type:            nonrobust   LLR p-value:
2.004e-305
==============================================================================
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------
------------
grade         -0.3666      0.010    -37.520      0.000      -0.386
-0.347
cons           0.1109      0.050      2.203      0.028       0.012
0.210
==============================================================================
```

From the results we can see: the coefficient of grade is -0.3666, it's significantly negative, meaning grade has a negative effect on default probability. It makes sense, since the higher grade is (more close to A), the lower default possibility.

b.(ii)

In [ ]:
```
#Likelihood Ratio Test
model1=sm.Logit(data['Default'],data[['cons']]).fit()
print(-2*(model1.llf-model.llf))
print(1-chi2.cdf(-2*(model1.llf-model.llf), 1))
```

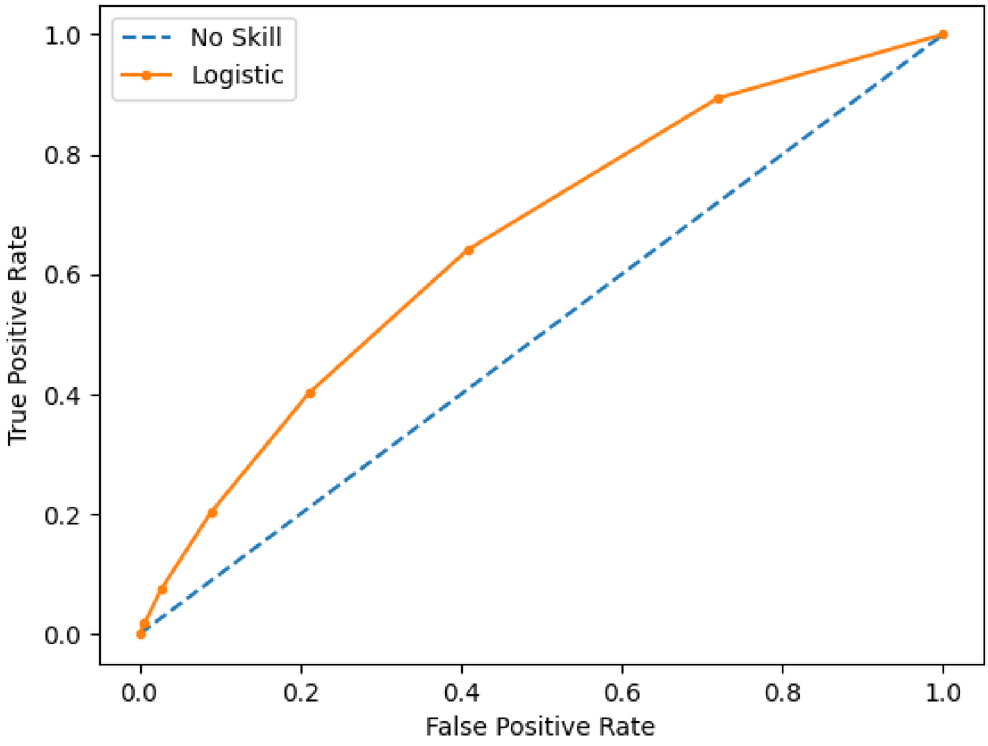
Result:
LR=1395.4930894519966
p value=0.0

From the result we can see: the p value is 0, meaning we reject the null hypothesis that the parameter of grade variable is zero. Meaning grade can help for the prediction of default.

b.(iii)

In [ ]:
```
lgt_fpr, lgt_tpr, _=roc_curve(data['Default'],model.predict(data[["grade",'cons']]
random_fpr, random_tpr, _ = roc_curve(data['Default'], [0 for i in range(len(data[

plt.plot(random_fpr, random_tpr, linestyle='--', label='No Skill')
plt.plot(lgt_fpr, lgt_tpr, marker='.', label='Logistic')
# axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the legend
plt.legend()
```

ROC Curve:



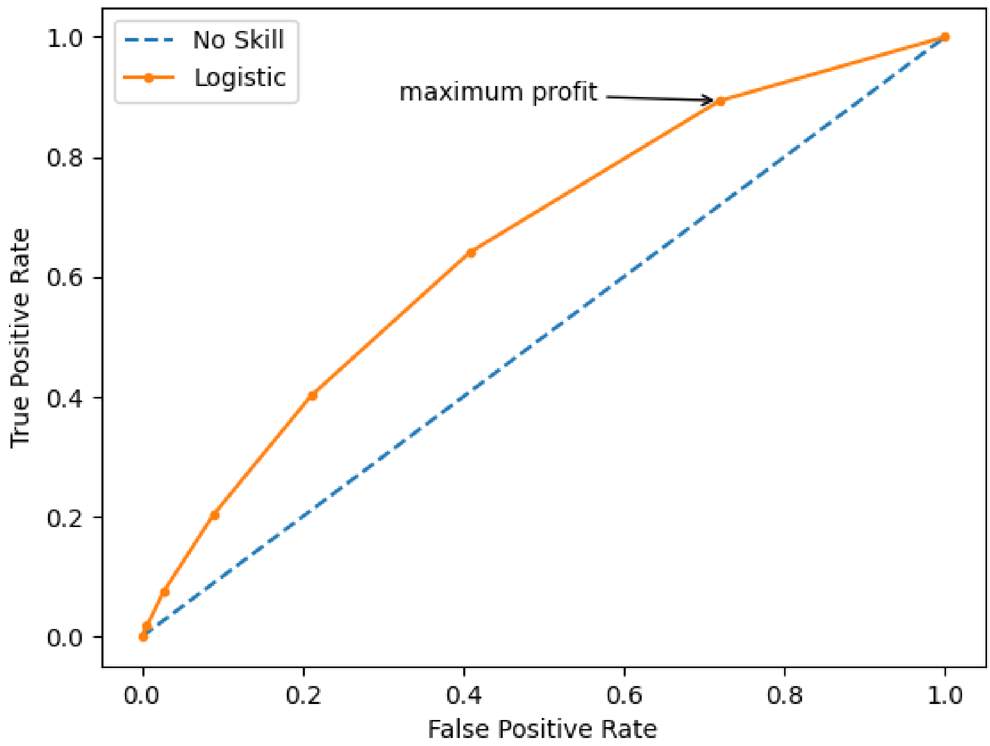
From the ROC Curve we can see: The logistic line is above the random guess line, which indicates that the model perform better than a random guess.

b.(iv)

In [ ]:
```
lgt_fpr, lgt_tpr, _=roc_curve(data['Default'],model.predict(data[["grade",'cons']]
random_fpr, random_tpr, _ = roc_curve(data['Default'], [0 for i in range(len(data[

plt.plot(random_fpr, random_tpr, linestyle='--', label='No Skill')
plt.plot(lgt_fpr, lgt_tpr, marker='.', label='Logistic')
# axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the legend
plt.legend()



data['prob']=model.predict(data[["grade",'cons']])
cutoff_list=[0]+sorted(model.predict(data[["grade",'cons']]).unique())
profit_list=[]
for p in cutoff_list:
    data['predict_default']=data['prob'].apply(lambda x:0 if x<=p else 1)
    temp_data=data[data['predict_default']==0]
    profit_list.append(np.sum(temp_data['Default'].apply(lambda x:1 if x==0 else -

print(cutoff_list)
print(profit_list)


plt.annotate('maximum profit',(lgt_fpr[-2],lgt_tpr[-2]),xytext =(lgt_fpr[-2]-0.4,l

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

```
The best cutoff default probability is 0.0791, that is, when
P<=0.0791, it don't default, otherwise it defaults.
```

Plot:



3

In [13]:
```
# Import necessary libraries
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split

# Load the dataset
file_path = 'LendingClub_LoanStats3a_v12.csv'
loan_data = pd.read_csv(file_path)

# Define a function to create a default column based on loan status
def create_default_variable(status):
    if status in ['Charged Off', 'Default']:
        return 1
    else:
        return 0

# Creating the binary target variable 'default'
loan_data['default'] = loan_data['loan_status'].apply(create_default_variable)

# Extracting the features and target
features = loan_data[['loan_amnt', 'annual_inc']]
target = loan_data['default']

# Adding a constant to the feature set
features = sm.add_constant(features, has_constant='add')
```

```
# Splitting data into train and test sets for robust regression testing
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.:

# Logistic Regression
logistic_model = sm.Logit(y_train, X_train)
result = logistic_model.fit()

# Print the summary of the regression
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.406186
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                default   No. Observations:                31828
Model:                          Logit   Df Residuals:                    31825
Method:                           MLE   Df Model:                            2
Date:                Mon, 22 Apr 2024   Pseudo R-squ.:                 0.01055
Time:                        07:57:08   Log-Likelihood:                -12928.
converged:                       True   LL-Null:                       -13066.
Covariance Type:            nonrobust   LLR p-value:                 1.291e-60
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -1.7138      0.036    -47.986      0.000      -1.784      -1.644
loan_amnt   3.138e-05   2.31e-06     13.603      0.000    2.69e-05    3.59e-05
annual_inc -6.717e-06   5.16e-07    -13.007      0.000   -7.73e-06   -5.71e-06
==============================================================================
```

```
/var/folders/rf/lmhdc33x4ys2xw1wmnlj6rfh0000gn/T/ipykernel_32082/1913912243.py:8:
DtypeWarning: Columns (21,24,29,31) have mixed types. Specify dtype option on impo
rt or set low_memory=False.
  loan_data = pd.read_csv(file_path)
```

Intercept (const): The coefficient is -1.7138, indicating the log-odds of default when both loan_amnt and annual_inc are zero.

loan_amnt: The coefficient is 3*10^-5 . This suggests that as the loan amount increases, the likelihood of default slightly increases.

annual_inc: The coefficient is -6 * 10^-6. This indicates that higher annual income slightly decreases the likelihood of default.

In [ ]:

(min_score max_score mean_score count n_positives \ quantile

0 1.821416e-01 0.272032 0.200977 796 175

1 1.656216e-01 0.182027 0.172752 796 160

2 1.560407e-01 0.165592 0.160408 796 120

3 1.492429e-01 0.156000 0.152320 795 114

4 1.425707e-01 0.149239 0.145744 796 109

5 1.361999e-01 0.142571 0.139402 796 93

6 1.291833e-01 0.136187 0.132765 795 102

7 1.200830e-01 0.129183 0.124775 796 96

8 1.051574e-01 0.120079 0.113470 796 70

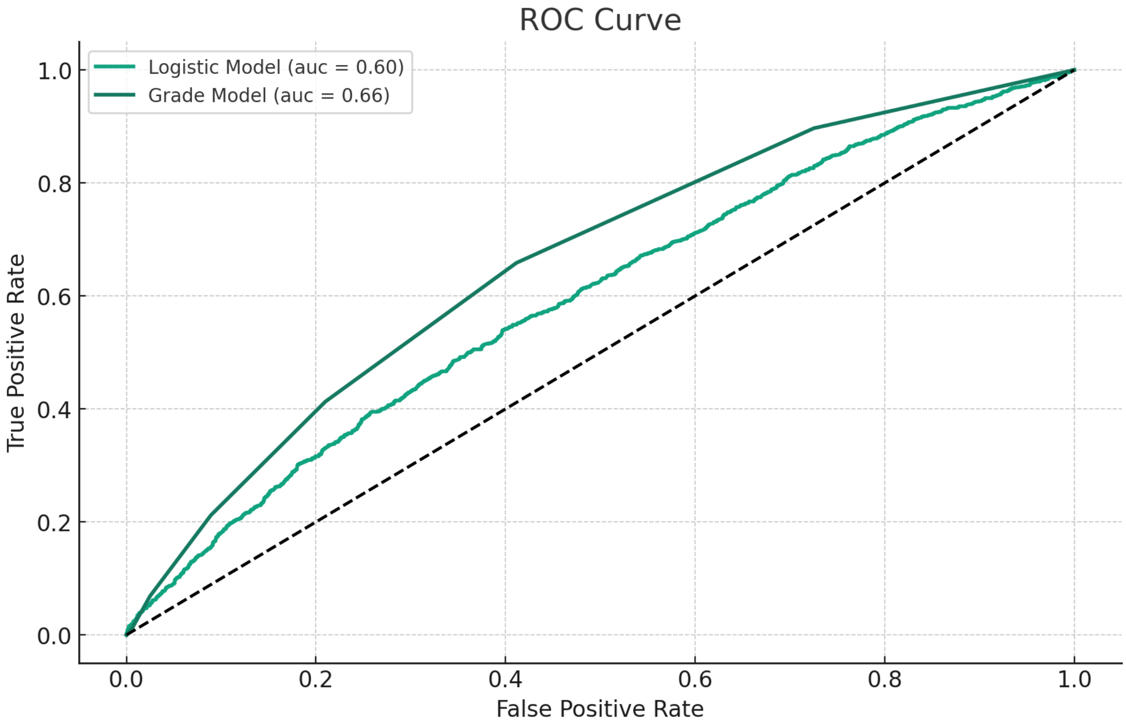
9 6.611422e-19 0.105157 0.084885 796 65

```
response_rate      lift
```

quantile
0 0.219849 1.584747
1 0.201005 1.448911
2 0.150754 1.086683
3 0.143396 1.033648
4 0.136935 0.987071
5 0.116834 0.842180
6 0.128302 0.924843
7 0.120603 0.869347
8 0.087940 0.633899
9 0.081658 0.588620 , min_score max_score mean_score count n_positives response_rate \
quantile
0 0.249558 0.333333 0.267518 796 225 0.282663
1 0.211930 0.249558 0.214246 796 164 0.206030
2 0.167518 0.211930 0.184424 796 147 0.184673
3 0.167518 0.167518 0.167518 795 131 0.164780
4 0.121285 0.167518 0.142427 796 110 0.138191
5 0.121285 0.121285 0.121285 796 92 0.115578
6 0.121285 0.121285 0.121285 795 95 0.119497
7 0.060344 0.121285 0.090202 796 47 0.059045
8 0.060344 0.060344 0.060344 796 50 0.062814
9 0.060344 0.060344 0.060344 796 43 0.054020

```
lift
```

quantile
0 2.037531
1 1.485134
2 1.331187
3 1.187788
4 0.996126
5 0.833124
6 0.861373
7 0.425618
8 0.452785
9 0.389395)



from the Roc curves, we see that the dummy variables for term strucutre and rates had improved the distinguishing power of the model between default and non defaults

In [26]:
```
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# Load the dataset
file_path = 'LendingClub_LoanStats3a_v12.csv'
loan_data = pd.read_csv(file_path)

# Define a function to create a default column based on loan status
def create_default_variable(status):
    if status in ['Charged Off', 'Default']:
        return 1
    else:
        return 0

# Creating the binary target variable 'default'
loan_data['default'] = loan_data['loan_status'].apply(create_default_variable)

# Preprocess 'term' to extract numeric values
loan_data['term_numeric'] = loan_data['term'].str.extract('(\d+)').astype(float)

# Prepare the updated feature set
features_updated = loan_data[['loan_amnt', 'annual_inc', 'term_numeric', 'int_rate
features_updated = sm.add_constant(features_updated, has_constant='add')

# Splitting data into train and test sets for the updated logistic regression mode
X_train_updated, X_test_updated, y_train_updated, y_test_updated = train_test_spli

# Train the updated logistic regression model
logistic_model_updated = sm.Logit(y_train_updated, X_train_updated)
logistic_result_updated = logistic_model_updated.fit()
```

```
print(logistic_result_updated.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.387066
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                default   No. Observations:                31828
Model:                          Logit   Df Residuals:                    31823
Method:                           MLE   Df Model:                            4
Date:                Mon, 22 Apr 2024   Pseudo R-squ.:                 0.05713
Time:                        08:26:35   Log-Likelihood:                -12320.
converged:                       True   LL-Null:                       -13066.
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const           -3.8268      0.078    -49.376      0.000      -3.979      -3.675
loan_amnt    -8.386e-07   2.57e-06     -0.327      0.744   -5.87e-06    4.19e-06
annual_inc   -5.785e-06   5.13e-07    -11.272      0.000   -6.79e-06   -4.78e-06
term_numeric     0.0165      0.002      9.942      0.000       0.013       0.020
int_rate        13.3283      0.505     26.385      0.000      12.338      14.318
================================================================================
```

```
/var/folders/rf/lmhdc33x4ys2xw1wmnlj6rfh0000gn/T/ipykernel_32082/119774756.py:9: D
typeWarning: Columns (21,24,29,31) have mixed types. Specify dtype option on impor
t or set low_memory=False.
  loan_data = pd.read_csv(file_path)
```

```
           response_rate      lift
```

quantile

0 0.312814 2.254868

1 0.222362 1.602858

2 0.183417 1.322131

3 0.162264 1.169654

4 0.136935 0.987071

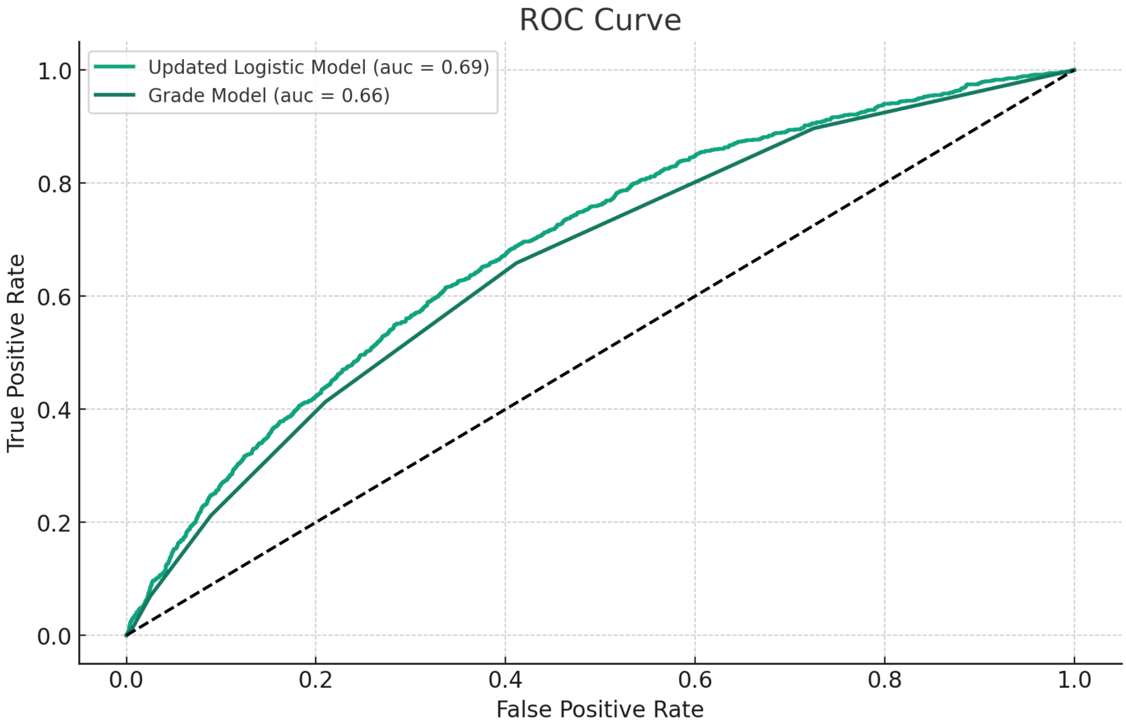
5 0.120603 0.869347

6 0.083019 0.598428

7 0.064070 0.461840

8 0.066583 0.479952

9 0.035176 0.253559)



The ROC curve plotted for the updated model and the original 'grade' model shows that the updated logistic model performs significantly better, as evidenced by the higher area under the curve. The lift table for the updated model also indicates improved performance across quantiles, especially in the highest risk quantiles.

In [27]:
```
# Create the squared term of the interest rate and add it to the features
loan_data['int_rate_squared'] = loan_data['int_rate'] ** 2

# Update the feature set to include the squared interest rate term
features_extended = loan_data[['loan_amnt', 'annual_inc', 'term_numeric', 'int_rat
features_extended = sm.add_constant(features_extended, has_constant='add')

# Split the data into train and test sets for the extended logistic regression mode
X_train_extended, X_test_extended, y_train_extended, y_test_extended = train_test_

# Train the extended logistic regression model
logistic_model_extended = sm.Logit(y_train_extended, X_train_extended)
logistic_result_extended = logistic_model_extended.fit()

# Display the summary of the extended logistic regression model
logistic_result_extended.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.386692
         Iterations 7
```

Out[27]:

Logit Regression Results

Dep. Variable:	default	**No. Observations:**	31828
Model:	Logit	**Df Residuals:**	31822
Method:	MLE	**Df Model:**	5
Date:	Mon, 22 Apr 2024	**Pseudo R-squ.:**	0.05804
Time:	08:32:41	**Log-Likelihood:**	-12308.
converged:	True	**LL-Null:**	-13066.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	-4.6723	0.193	-24.201	0.000	-5.051	-4.294
loan_amnt	3.642e-08	2.56e-06	0.014	0.989	-4.98e-06	5.06e-06
annual_inc	-5.652e-06	5.12e-07	-11.037	0.000	-6.66e-06	-4.65e-06
term_numeric	0.0171	0.002	10.388	0.000	0.014	0.020
int_rate	26.1873	2.722	9.620	0.000	20.852	31.523
int_rate_squared	-47.9251	9.935	-4.824	0.000	-67.398	-28.453

The inclusion of interest rate and its squared term indicates a nonlinear association between interest rates and the likelihood of default. This suggests that as interest rates rise, the probability of default also increases, but the pace of this increase slows down over time. This is evidenced by the negative coefficient of the squared interest rate term, which implies a diminishing impact at higher interest rates.