

```
[1]: import warnings
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns

In [2]: from scipy.stats import ttest_ind
import re
from category_encoders import TargetEncoder
from imblearn.over_sampling import RandomOverSampler
from sklearn.metrics import roc_auc_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import roc_curve

In [3]: from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OrdinalEncoder
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import statsmodels.discrete.discrete_model as sm
from statsmodels.api import add_constant as ac

In [4]: df = pd.read_csv("C:/Users/akash/Desktop/Scaler/Case_Study/LoanTap/LoanTap.csv")
df.head()
```

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_acc	pub_rec
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	...	16.0	0.0
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	...	17.0	0.0
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	...	13.0	0.0
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	...	6.0	0.0
4	24375.0	60 months	17.27	609.33	C	C5	Management Inc.	9 years	MORTGAGE	55000.0	...	13.0	0.0

5 rows x 27 columns

```
In [5]: df.info()
Out[5]: (39630, 27)

In [6]: df.info()
class 'pandas.core.frame.DataFrame':
RangeIndex: 39630 entries, 0 to 396029
Data columns (total 27 columns):
#      Column                Non-Null Count  Dtype
---  --
0     loan_amnt              39630 non-null  float64
1     term                  39630 non-null  object
2     int_rate              39630 non-null  object
3     installment          39630 non-null  float64
4     grade                 39630 non-null  object
5     sub_grade            39630 non-null  object
6     emp_title             373103 non-null object
7     emp_length           377729 non-null object
8     home_ownership        39630 non-null  object
9     annual_inc           39630 non-null  float64
10    verification_status   39630 non-null  object
11    issue_d              39630 non-null  object
12    loan_status          39630 non-null  object
13    purpose              394275 non-null object
14    title                394275 non-null object
15    dti                  39630 non-null  float64
16    earliest_cr_line      377729 non-null object
17    open_acc             39630 non-null  float64
18    pub_rec              39630 non-null  float64
19    revol_bal            39630 non-null  float64
20    revol_util           395794 non-null object
21    total_acc            39630 non-null  float64
22    initial_list_status   39630 non-null  object
23    application_type      39630 non-null  object
24    mort_acc             398235 non-null float64
25    pub_rec_bankruptcies 39630 non-null  float64
26    address              39630 non-null  object
dtypes: float64(12), object(15)
memory usage: 81.6+ MB

Checking the outliers
In [7]: df.describe()
Out[7]:
```

	loan_amnt	term	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util
count	14113.880809	13.690000	39630.000000	39630.000000	396300e+05	39630.000000	39630.000000	39630.000000	396300e+05	395754.000000
mean	14113.880809	13.690000	431.849698	74203.175790	17.379514	11.311153	0.178191	15844.539653	53.791749	25.147144
std	8357.441341	4.472157	250.727790	6.163762e+04	18.019092	5.137649	0.530671	2.059184e+04	24.452193	
min	500.000000	5.320000	16.080000	0.000000e+00	0.000000	0.000000	0.000000	6.020000e+03	0.000000	
25%	8000.000000	10.490000	250.330000	4.500000e+04	11.280000	8.000000	0.000000	6.020000e+03	35.800000	
50%	12000.000000	13.330000	375.430000	6.400000e+04	16.910000	10.000000	0.000000	1.118100e+04	54.800000	
75%	20000.000000	16.490000	567.300000	9.000000e+04	22.980000	14.000000	0.000000	1.962200e+04	72.900000	
max	40000.000000	30.990000	1533.810000	8.706582e+06	9999.000000	90.000000	86.000000	1.743266e+06	892.300000	

Lets check loan amount, annual income and revol balance - As these have some differenc in mean and median

```
In [8]: df.describe().loc[['mean','50%']]
Out[8]:
```

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util	total_acc	mort_acc	pub_rec
mean	14113.880809	13.6394	431.849698	74203.175790	17.379514	11.311153	0.178191	15844.539653	53.791749	25.147144	1.813991	
50%	12000.000000	13.3300	375.430000	64000.000000	16.910000	10.000000	0.000000	11181.000000	54.800000	24.000000	1.000000	

Univariate

```
In [9]: cols = ['loan_amnt', 'annual_inc', 'revol_bal']
fig,axs = plt.subplots(ncows = 2,ncols = 3,figsize = (15,7))
axs = axs.flat
index = 0
for col in cols:
    sns.boxplot(y = df[col],ax = axs[index])
    index += 1
plt.show()
```

What percentage of customers have fully paid their Loan Amount?

```
In [10]: (df['loan_status'].value_counts()[0])/len(df)*100
Out[10]: 80.3870918697825
```

Lets do % of outliers

- We have 4 and 5 percent of data as outliers

```
In [11]: limits = []
for col in cols:
    QQR = df[col].quantile(.75) - df[col].quantile(.25)
    limit = (1.5*QQR)/df[col].quantile(.75)
    limits.append(limit)
n = sum(df[col] > limit)
percent = round(n*100/len(df), 2)
print('Percentage of Outliers in the {} is {}'.format(col,percent))

Percentage of Outliers in the loan_amnt is 0.05
Percentage of Outliers in the annual_inc is 4.22
Percentage of Outliers in the revol_bal is 5.37

Lets check for common outlier
In [12]: len(df[(df['loan_amnt']>limits[0]) & (df['annual_inc']>limits[1]) & (df['revol_bal']>limits[2])])
Out[12]: 6.312653081837235e-05

As we see outliers came out of annual income and revol bal actually have 4774 data points in which case this
will very useful, So its better not to drop these, as these will even give a new meaning to our Model.
In [13]: len(df[(df['annual_inc']>limits[1]) & (df['revol_bal']>limits[2])])
Out[13]: 4799

0.0063 percent common outliers - Dropping these
In [14]: df.drop(df[(df['loan_amnt']>limits[0]) & (df['annual_inc']>limits[1]) & (df['revol_bal']>limits[2])].index, inplace = True)
Out[14]: (396005, 27)

Categorical Values
In [15]: df.describe(include = 'object')
Out[15]:
```

	term	grade	sub_grade	emp_title	emp_length	home_ownership	verification_status	issue_d	loan_status	purpose
count	390005	390005	396005	377078	377704	396005	398005	396005	396005	396005
unique	2	7	35	173101	11	6	3	115	2	14
top	36 months	B	B3	Teacher	10+ years	MORTGAGE	Verified	Oct-2014	Fully Paid	debt_consolidation
freq	301988	116009	26653	4389	126029	198329	139544	14846	318333	244494

No duplicates

```
In [16]: df[df.duplicated()]
Out[16]:
```

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_acc	pub_rec	revol_bal
--	-----------	------	----------	-------------	-------	-----------	-----------	------------	----------------	------------	-----	----------	---------	-----------

0 rows x 27 columns

Missing Values treatment using mean and mode to impute as there is no outliers in features which have missing values

```
In [17]: df.isna().sum()/len(df)*100
Out[17]:
```

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_acc	pub_rec	revol_bal
loan_amnt	0.000000													
term	0.000000													
int_rate	0.000000													
installment	0.000000													
grade	0.000000													
sub_grade	0.000000													
emp_title	5.789573													
emp_length	76.524177													
home_ownership	0.000000													
annual_inc	0.000000													
verification_status	0.000000													
issue_d	0.000000													
title	0.422924													
dti	0.000000													
earliest_cr_line	0.000000													
open_acc	0.000000													
pub_rec	0.000000													
revol_bal	78.524176													
revol_util	0.069696													
total_acc	0.000000													
initial_list_status	0.000000													
application_type	0.000000													
mort_acc	9.544071													
pub_rec_bankruptcies	0.133099													
address	0.000000													
dtype:	float64													

```
In [18]: categorical_missing = ['emp_title','emp_length','title']
numerical_missing = ['mort_acc','pub_rec_bankruptcies','revol_util']
cols = df.columns
df[col] = df[col].fillna(df[col].mode()[0])
for col in categorical_missing:
    df[col] = df[col].fillna(df[col].mean())

Bi-Variate
Categorical
In [19]: categorical = df.describe(include = 'object').columns
fig,ax = plt.subplots(ncows = 2,ncols = 3,figsize = (20,15))
ax = axs.flat
axs = ['sub_grade','emp_title','issue_d','title','earliest_cr_line','address','loan_status','purpose','emp_length']
index = 0
for col in categorical:
    if col in avoid:
        sns.complot(x = df[col],hue = df['loan_status'],ax = axs[index])
        index += 1
plt.show()
```

Many people take 36 Months loan

People in grade G and F have high possibility of not paying the loan

Very less number of people who owns the house take loan

Interestingly Verified people have high probability of charged off

```
In [20]: fig,ax = plt.subplots(ncows = 3,ncols = 1,figsize = (20,20))
ax = axs.flat
avoid = ['sub_grade','emp_title','issue_d','title','earliest_cr_line','address','loan_status','purpose','emp_length']
index = 0
for col in categorical:
    if col in avoid:
        sns.complot(x = df[col],hue = df['loan_status'],ax = axs[index])
        index += 1
plt.show()
```

Debt Consolidation and credit card is mostly a purpose for the loan

10+ and 2 year take more loans

Numerical

```
In [21]: numerical = df.describe().columns
fig,ax = plt.subplots(ncows = 4,ncols = 3,figsize = (20,25))
ax = axs.flat
avoid = ['sub_grade','emp_title','issue_d','title','earliest_cr_line','address','loan_status','purpose','emp_length']
index = 0
for col in numerical:
    sns.boxplot(y = df[col],x = df['loan_status'],ax = axs[index])
    index += 1
plt.show()
```

Loan amount, Interest Rate, Installments,Cti, pub_rec - When these have higher value there is a better chance the loan will be charged off

Annual Income and mort_acc have positive impact on payin off the loan

Other Features seems like doesn't have much affect on the loan status, Lets do Hypothesis testing to confirm.

Hypothesis Testing - T-Test

- Null Hypothesis - Feature have same mean for whether loan is paid or charged off.
- Alternative Hypothesis - Feature have different mean for whether loan is paid or charged off.
- Significance = 5 percent
- Features going to test = 'open_acc', 'revol_bal', 'total_acc'
- To make T-test powerful taking equal number of samples in each category

```
In [22]: test = ['open_acc', 'revol_bal', 'total_acc']
for col in test:
    df[df['loan_status'] == 'Charged Off'].dropna()[col]
    data = df[df['loan_status'] == 'Fully Paid'].dropna().sample(len(df[df['loan_status'] == 'Charged Off'])).reset_index(drop=True)
    print('T-Test Results for {} = {}'.format(a = col, b = ttest_ind(df[df['loan_status'] == 'Fully Paid'], df[df['loan_status'] == 'Charged Off'])).reset_index(drop=True))

T-Test Results for open_acc = Ttest_indResult(statistic=-13.493213173325426, pvalue=1.8096729513390234e-41)
T-Test Results for revol_bal = Ttest_indResult(statistic=-5.91379660052219, pvalue=3.354733420539526e-09)
T-Test Results for total_acc = Ttest_indResult(statistic=8.317480217529207, pvalue=9.0567665024447e-17)

P-value is significantly lesser than .05 in all the 4 cases which means we reject Null Hypothesis - All mean are not same

Not able to derive much as annual income got scaled to 1million
In [23]: df[df['loan_status'] == 'annual_inc', x = 'loan_amnt', data = df]
plt.show()
```

Lot of different interest rate for different people

```
In [24]: df['int_rate'].value_counts()
Out[24]:
```

int_rate	count
10.99	12410
12.99	9631
15.61	9350
11.99	8582
8.90	8019
17.34	1
14.38	1
14.62	1
14.70	1
22.64	1

Name: int_rate, Length: 566, dtype: float64

At all interest we have different loan amount, interest rate with 30 percent and 40,000 loan amount as well

```
In [25]: sns.scatterplot(y = 'int_rate', x = 'loan_amnt', data = df)
plt.show()
```

Loan Amount increases installment increases

```
In [26]: sns.scatterplot(y = 'installment', x = 'loan_amnt', data = df)
plt.show()
```

Multi - Variate

Not much can be derived from this

```
In [28]: sns.scatterplot(y = 'installment', x = 'loan_amnt', hue = 'loan_status', data = df)
plt.show()
```

Higher loan amount gave to verified people at an average

- In all different verification status, there is an equal possibility to paid or charged off.

```
In [29]: sns.boxplot(hue = 'loan_status', y = 'loan_amnt', x = 'verification_status', data = df)
plt.show()
```

Average Annual Income is high, then probability of paying loan back is high in verification status

```
In [30]: sns.boxplot(hue = 'loan_status', y = 'annual_inc', x = 'verification_status', data = df)
plt.show()
```

Intersting info from plot - grade G have the highest annual income

```
In [31]: sns.barplot(hue = 'loan_status', y = 'annual_inc', x = 'grade', data = df)
plt.show()
```

which grade people are more likely to pay the loan back

```
In [32]: (df[df['loan_status'] == 'Fully Paid']['grade'].value_counts()/df['grade'].value_counts())*100
Out[32]:
```

grade	percentage
B	87.425976
A	85.364001
C	78.524176
D	93.711731
F	71.131262
G	62.6328394
52.161100	

which verification_status people are more likely to pay the loan back

```
In [33]: (df[df['loan_status'] == 'Fully Paid']['verification_status'].value_counts()/df['verification_status'].value_counts())*100
Out[33]:
```

verification_status	percentage
Not Verified	85.364001
Source Verified	78.524176
Verified	77.676575

which purpose people are more likely to pay the loan back

```
In [34]: (df[df['loan_status'] == 'Fully Paid']['purpose'].value_counts()/df['purpose'].value_counts())*100
Out[34]:
```

purpose	percentage
debt_consolidation	86.523313
credit_card	83.286351
debt_consolidation	78.524176
educational	83.657588
home_improvement	82.591385
house	80.281490
major_purchase	83.524861
medical	78.288847
moving	76.524177
other	78.781256
renewable_energy	76.595745
small_business	70.549026
vacation	81.076702
wedding	87.913907

Heat Map

- Loan amount have positive correlation with Annual Income and revol balance
- Loan amount have very high positive correlation with installments
- Annual Income have positive correlation with installment
- pub_rec_bankruptcies have very high positive correlation with pub_rec
- revol_bal and pub_rec_bankruptcies have negative correlation
- open_acc and total_acc have high positive correlation
- pub_rec_bankruptcies and pub_rec have negative correlation with loan amount
- Annual Income and dti have negative correlation

```
In [36]: fig,ax = plt.subplots()
fig.set_size_inches(15, 10)
sns.heatmap(df.corr(method = 'spearmanr'), annot = True)
```

Pre-Processing

Convert issue_d to year

```
In [37]: df['issue_year'] = df['issue_d'].apply(lambda x: int(x[:4]))
df.drop(columns = ['issue_d'], inplace = True)
```

Converting Address to the given last 4 numbers to gain better understanding on whole area rather than having one address

```
In [38]: df['address'] = df['address'].apply(lambda x: x[-5:])
```

Convert earliest_cr_line to year

```
In [39]: df['earliest_cr_line_year'] = df['earliest_cr_line'].apply(lambda x: int(x[:4]))
df.drop(columns = ['earliest_cr_line'], inplace = True)
```

Encode

```
In [40]: def regex(x):
return int(re.findall('\d+', x)[0])

In [41]: df['emp_length'] = df['emp_length'].apply(lambda x: 0 if x == '< 1 year' else int(x))
df['emp_length'] = df['emp_length'].apply(lambda x: 10 if x == '10+ years' else int(x))
df['emp_length'] = df['emp_length'].apply(regex)
```

Encoding

- Fully Paid = 0
- Charged off = 1

One Hot Encoding


```
In [42]: df['loan_status'] = pd.get_dummies(df['loan_status'])['Charged Off']

In [43]: onehot = {'term','verification_status','initial_list_status', 'application_type'}
for col in onehot:
    temp = pd.get_dummies(df[col]).iloc[:,1:]
    df.drop(columns = [col], inplace = True)
    df = pd.merge(df, temp, left_index=True, right_index=True)
```

Ordinal Encoding

```
In [44]: enc = OrdinalEncoder()
df['grade'] = enc.fit_transform(np.array(df['grade']).reshape(-1,1))

In [45]: enc1 = OrdinalEncoder()
df['sub_grade'] = enc1.fit_transform(np.array(df['sub_grade']).reshape(-1,1))
```

Target Encoding

```
In [46]: target = ['emp_title','home_ownership','purpose','address','title']
alist = []
for col in target:
    enc = TargetEncoder()
    df[col] = enc.fit_transform(df[col], df['loan_status'])

In [47]: df.head()
```

	loan_amnt	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	loan_status	...	pub_rec	bankruptcies
0	10000.0	11.44	259.48	1.0	8.0	0.247191	1	0.226620	117000.0	0	...	0	0
1	8000.0	11.99	265.68	1.0	9.0	0.316979	4	0.169577	65000.0	0	...	0	0
2	15600.0	10.49	506.97	1.0	7.0	0.181819	1	0.226620	43057.0	0	...	0	0
3	7200.0	6.49	220.65	0.0	1.0	0.196139	6	0.226620	54000.0	0	...	0	0
4	24375.0	17.27	609.33	2.0	14.0	0.196139	9	0.169577	55000.0	1	...	0	0

5 rows × 29 columns

Imbalanced data - oversampling the minority

```
In [48]: df['loan_status'].value_counts()
0    318333
1     77672
Name: loan_status, dtype: int64

In [49]: oversample = RandomOverSampler(sampling_strategy='minority')
x,y = oversample.fit_resample(df.drop(columns = ['loan_status']), df['loan_status'])

In [50]: y.value_counts()
0    318333
1     318333
Name: loan_status, dtype: int64
```

Standardizing the Data

```
In [51]: scaling = StandardScaler()
scaling.fit(x)
scaled = scaling.transform(x)
df_scaled = pd.DataFrame(scaled, columns = x.columns)

In [52]: df_scaled.head()
```

	loan_amnt	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	purpose	...	pub_rec	bankruptcies
0	-0.533747	-0.666588	-0.440616	-0.783166	-0.644126	0.214728	-0.761234	1.066051	0.728663	-0.336095	...	0	-0.3
1	-0.771181	-0.546418	-0.695803	-0.783166	-0.498364	0.823078	0.426438	-1.028366	-0.106978	0.430700	...	0	0
2	0.131065	-0.874514	0.269309	-0.783166	-0.789888	0.823078	-0.761234	1.066051	-0.459602	-1.267692	...	0	-0.3
3	-0.866154	-1.748118	-0.875914	-1.503971	-1.664459	-0.230296	1.218220	1.066051	-0.283748	-1.267692	...	0	-0.3
4	1.172803	0.607215	0.678728	-0.062360	0.230446	-0.230296	2.405893	-1.028366	-0.267678	-1.267692	...	0	-0.3

5 rows × 28 columns

Train - Val - Test Split

```
In [53]: x_train,x_test,y_train,y_test = train_test_split(df_scaled,y,test_size =.2)

In [54]: x_train,x_val,y_train,y_val = train_test_split(x_train,y_train,test_size =.25)
```

Model - 1 Logistic Regression

- 83 percent accuracy is definitely not good

```
In [55]: model1 = LogisticRegression()
model1.fit(x_train,y_train)
print("Train score : (a), val score : (b)".format(a = model1.score(x_train,y_train), b = model1.score(x_val,y_train)))
Train score : 0.8295885591323537, val score : 0.8302953672653594

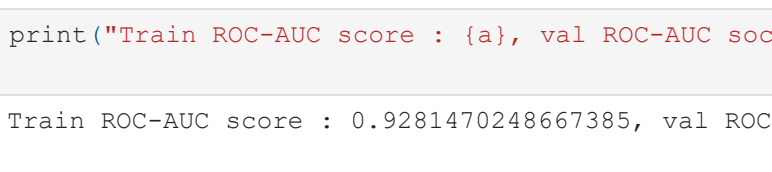
In [56]: print("Train f1_score score : (a), val f1_score score : (b)".format(a = f1_score(y_train,model1.predict(x_train)
b = f1_score(y_val,model1.predict(x_val))))
Train f1_score score : 0.8276986519643523, val f1_score score : 0.8289883010803752
```

AUC - ROC Score

- this score seems to be better in both train and val

```
In [57]: print("Train ROC-AUC score : (a), val ROC-AUC score : (b)".format(a = roc_auc_score(y_train,model1.predict_proba(x_train)[:,1])
b = roc_auc_score(y_val,model1.predict_proba(x_val)[:,1]))
Train ROC-AUC score : 0.9281668121361394, val ROC-AUC score : 0.9277162314242234

In [58]: fpr,tpr,th = roc_curve(y_train,model1.predict_proba(x_train)[:,1])
plt.plot(fpr,tpr)
plt.show()
```



Model - 2 Logistic Regression using elastic net not much improvement in this case

- 83 percent accuracy is definitely not good

```
In [59]: model2 = LogisticRegression(penalty='elasticnet', solver = 'saga', l1_ratio = 0.5)
model2.fit(x_train,y_train)
print("Train score : (a), val score : (b)".format(a = model2.score(x_train,y_train), b = model2.score(x_val,y_train)))
Train score : 0.8298681386600486, val score : 0.8290231126259493

In [60]: print("Train f1_score score : (a), val f1_score score : (b)".format(a = f1_score(y_train,model2.predict(x_train)
b = f1_score(y_val,model2.predict(x_val))))
Train f1_score score : 0.8256383705538238, val f1_score score : 0.826305837674823
```

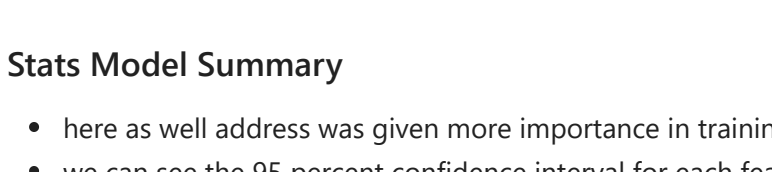
AUC - ROC Score

- this score seems to be better in both train and val

```
In [61]: print("Train ROC-AUC score : (a), val ROC-AUC score : (b)".format(a = roc_auc_score(y_train,model2.predict_proba(x_train)[:,1])
b = roc_auc_score(y_val,model2.predict_proba(x_val)[:,1]))
Train ROC-AUC score : 0.9281470248667385, val ROC-AUC score : 0.9277437502442812
```

There doesnt seem to be much difference between train and val score lets plot the ROC- AUC Curve

```
In [62]: fpr,tpr,th = roc_curve(y_train,model2.predict_proba(x_train)[:,1])
plt.plot(fpr,tpr)
plt.show()
```



Not much difference in both the models, so lets go with model1

Model Interpretation

- Below are the top 5 important feature as per our model

```
In [63]: coeff = pd.DataFrame({'Coeff': model1.coef_.reshape(len(df_scaled.columns),), 'Features': df_scaled.columns})
coeff.sort_values('Coeff', ascending=False).head()
```

Coeff	Features
19	13432821 address
5	0.899045 emp_title
11	0.858744 dti
4	0.549186 sub_grade
10	0.422012 title

Stats Model Summary

- here as well accuracy was given more importance in the model.
- we can see the 95 percent confidence interval for each features.

```
In [64]: x_sm = sm.Logit(x_train)
model = sm.Logit(y_train,x_sm).fit()
model.summary()
Optimization terminated successfully.
Current function value: inf
Iterations 14
```

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib\site-packages\statsmodels\base\model.py:592: ResessianInversionWarning: Inverting hessian failed, no bse or cov_params available

C:\Users\akash\Anaconda3\envs\fictional\lib