In [1]:
```
import numpy as np
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt

import seaborn as sns
sns.set()

%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
loan_df = pd.read_csv('Data/Loan_Data.csv')
```

In [3]:
```
loan_df.head()
```

Out[3]:

	customer_id	credit_lines_outstanding	loan_amt_outstanding	total_debt_outstanding	income	years_employed	fico_score
0	8153374	0	5221.545193	3915.471226	78039.38546	5	605
1	7442532	5	1958.928726	8228.752520	26648.43525	2	572
2	2256073	0	3363.009259	2027.830850	65866.71246	4	602
3	4885975	0	4766.648001	2501.730397	74356.88347	5	612
4	4700614	1	1345.827718	1768.826187	23448.32631	6	631

Requirement: Input an FICO score and accurately return a categorical value. Given a set number of buckets corresponding to the number of input labels for the model, find out the boundaries that best summarize the data.

You need to create a rating map that maps the FICO score of the borrowers to a rating where a lower rating signifies a better credit score.

In [21]:
```
# The problem in its simplest form example!

def bucket_customers_by_fico_score(fico_scores):
```

```
fico_score_buckets = {"Excellent": [],
                      "Great": [],
                      "Very Good": [],
                      "Good": [],
                      "Fair": [],
                      "Not Good": [],
                      "Poor": [],
                      "Bad": [],
                      "Horrible": [],
                      "Terrible": []
                     }

for fico_score in fico_scores:
    if fico_score[0] < 120:
        fico_score_buckets["Excellent"].append(fico_score)

    elif fico_score[0] <= 120 and fico_score[0] < 240:
        fico_score_buckets["Great"].append(fico_score)

    elif fico_score[0] <= 240 and fico_score[0] < 360:
        fico_score_buckets["Very Good"].append(fico_score)

    elif fico_score[0] <= 360 and fico_score[0] < 480:
        fico_score_buckets["Good"].append(fico_score)

    elif fico_score[0] <= 480 and fico_score[0] < 600:
        fico_score_buckets["Fair"].append(fico_score)

    elif fico_score[0] <= 600 and fico_score[0] < 650:
        fico_score_buckets["Not Good"].append(fico_score)

    elif fico_score[0] <= 650 and fico_score[0] < 700:
        fico_score_buckets["Poor"].append(fico_score)

    elif fico_score[0] <= 700 and fico_score[0] < 750:
        fico_score_buckets["Bad"].append(fico_score)

    elif fico_score[0] <= 750 and fico_score[0] < 800:
        fico_score_buckets["Horrible"].append(fico_score)
```

```
        else:
            fico_score_buckets["Terrible"].append(fico_score)

    return fico_score_buckets
```

In [8]:
```
from scipy.optimize import minimize
```

In [17]:
```
# temp = list(loan_df[["fico_score", "default"]].itertuples(index=False, name=None))
```

In [22]:
```
fico_tuple = list(loan_df[["fico_score", "default"]].itertuples(index=False, name=None))

buckets = bucket_customers_by_fico_score(fico_tuple)
```

In [109…
```
ll_df = loan_df[["fico_score", "default"]]
```

In [124…
```
def log_likelihood(buckets):

    LL_sum = 0

    for key in buckets:
        n = len(buckets[key])

        if n != 0:
#             print("n: ", n)
            k = 0
            for i in range(n):
                k += buckets[key][i][1]
#             print("k: ", k)

            p = k/n
#             print("p: ", p)
            LL_sum += k*np.log(p) + (n-k)*np.log(1-p)

#         print("LL: ", LL_sum)

    return LL_sum
```

In [126…
```
minimize(log_likelihood, [])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[126], line 1
----> 1 minimize(log_likelihood)

TypeError: minimize() missing 1 required positional argument: 'x0'
```

Below is the Example Answer!

In [127…
```
x = loan_df['default'].to_list()
y = loan_df['fico_score'].to_list()
n = len(x)
print (len(x), len(y))
```

```
10000 10000
```

In [131…
```
y[0]
```

Out[131]: 605

In [140…
```
# [[[-10**18, 0] for i in range(551)] for j in range(10+1)]
```

In [128…
```
# Initialising list
default = [0 for i in range(851)]
total = [0 for i in range(851)]

for i in range(n):
    y[i] = int(y[i])
    default[y[i]-300] += x[i]  # Why do we need to -300 from the fico_score value y[i]? Trying to randomise??
    total[y[i]-300] += 1

for i in range(0, 551):
    default[i] += default[i-1]
    total[i] += total[i-1]

def log_likelihood(n, k):
    p = k/n
```

```
    if (p==0 or p==1):
        return 0

    return k*np.log(p)+ (n-k)*np.log(1-p)

r = 10 #rank 10 => 10 buckets

# The code then initializes a three-dimensional array, dp,
# that is used to store the calculated log-likelihood values for different sets of observations.
# The first dimension represents the number of iterations performed,
# the second dimension represents the rank of the observation,
# and the third dimension represents the log-likelihood and the index of the previous observation.

dp = [[[-10**18, 0] for i in range(551)] for j in range(r+1)]


for i in range(r+1):
    for j in range(551):
        if (i==0):
            dp[i][j][0] = 0
        else:
            for k in range(j):
                if (total[j]==total[k]):
                    continue
                if (i==1):
                    dp[i][j][0] = log_likelihood(total[j], default[j])
                else:
                    if (dp[i][j][0] < (dp[i-1][k][0] + log_likelihood(total[j]-total[k], default[j] - default[k]))):
                        dp[i][j][0] = log_likelihood(total[j]-total[k], default[j]-default[k]) + dp[i-1][k][0]
                        dp[i][j][1] = k

print (round(dp[r][550][0], 4))

k = 550
l = []
while r >= 0:
    l.append(k+300)
    k = dp[r][k][1]
    r -= 1
```

```
print(l)
```

```
-4217.8245
[850, 753, 752, 732, 696, 649, 611, 580, 552, 520, 300]
```