**<u>שמות ות"ז:</u>**

1) פאדי אמון, 212472542

2) רשיד אבו מדג'ם, 212555650

3) ערין אבו כף, 212654719

תיאור של מבנה הפרויקט, ותפקיד של כל קובץ, מחלקה, שיטות, ושדות:

הפרויקט שלנו מחולק לכמה מחלקות, ומכיל שני חלונות (Frames) אחד משמש כמסך לקלט מהמשתמש והמסך השני משמש להצגת התוצאות.

**<u>המחלקות</u>**:

1) **Binning-** מחלקה זו אחראית על הדיסקריטיזציה, והיא מכילה את ה- data ומחלקת אותם גם למספרים הדליים (bins) המבוקש.

2) **EntropyTree-** מחלקה המתארת עץ בינארי, ומכילה את הנתונים והאנטרופיה של כל נקודת פיצול.

3) **Pre-** המחקלה מטפלת בעיבוד המקדים.

4) **Process-** המחלקה יורשת מ- Pre ומעבדת את הנתונים בעזרת האלגוריתמים שמומשו על ידינו.

5) **BuildAlgorithm-** המחלקה יורשת מ- Pre ומעבדת את הנתונים בעזרת האלגוריתמים של הספריות המוכנות.

**תיאור סדר התלויות של הקבצים השונים בפרויקט:**

הקובץ main מכיל את ה- main ואת ה-GUI של הפרויקט, ומשתמש בקובץ Process בלבד.

הקובץ Binning משתמש בקובץ Entropy בלבד.

הקובץ Process משתמש בקובץ NaiveBayes, וקובץ Binning וקובץ ID3.

הקובץ Entropy משתמש בקובץ EntropyTree בלבד.

בקבצים ID3, NaiveBayes, EntropyTree אין שום שימוש בקבצים אחרים.

תפקידי הקבצים, ומה הם מכילים:

1) **main.py**- הקובץ מכיל את כל ה-GUI של המערכת ומקבל את כל הקלט כגון סוג אלגוריתם וכמה Bins ולבסוף מציג את הפלט למשתמש.

פונקציות מוכלות:

-**MainFrame**: המסך הראשי שמקבל את כל הקלט מהמשתמש ובודק את תקינותם ולאחר מכן עובר למסך של הפלט.

-**Neighbours_Clusters**: כאשר KNN או KMEANS נבחרים אז יש למשתמש אפשרות לבחור מספר שכנים או קלאסטרים.

-**Check_Input**: מעבר בין המסך הראשי של הקלט למסך של הפלט, וזאת לאחר בדיקת תקינות של כל הקלט, והצגת הודעה מתאימה במידה שהקלט לא תקין.

-**Back_To_MainFrame**: חזרה למסך הראשי לאחר הצגת התוצאות.

-**Update_Frame_Results**: הצגת התוצאות בפני המשתמש.

-**Apply_Algorithm**: מריץ את האלגוריתם על הנתונים ומחזיר תוצאות לפי הקלטים של המשתמש.

2) **Process.py**- הקובץ מכיל שלוש מחלקות: Pre, Process, BuildAlgorithm, תפקיד המחלקות האלה הוא הרצת האלגוריתמים וביצוע העיבוד המקדים לפי הדרישות של המשתמש בקלט.

**מטודות מוכלות במחלקות:**

**מחלקת Pre:**

**Is_Empty**: בודקת אם הקובץ מכיל תוכן.

**Delete_Nan_Class_Row**: מוחקת השורות שבהם ה CLASS ריק.

**Fill_Nan_Values**: השלמת ערכים חסרים.

**Normalize**: נירמול נתונים.

**Binning**: דיסקריטיזציה לפי השיטה וכמות bins שנבחרו.

**read_structure**: החזרת קובץ structure כמילון.

**Clean_Data**: עיבוד מקדים על נתונים בקובץ מסויים.

**Save_Data**: שמירת קובץ CSV לאחר העיבוד המקדים.

**מחלקת BuildAlgorithm:**

**Convert_Strings_To_Numbers**: בשיטה זו השתמשנו בכלי מוכן על מנת להמיר מחרוזות למספרים.

**Run**: בשיטה זו השתמשנו בכלי מוכן על מנת להפעיל המודל על הקבצים והחזרת התוצאות.

**מחלקת Process:**

**Build_Model**: בניית מודל על פי האלגוריתם שהתקבל בקלט.

**Save_Model**: שמירת קובץ SAV של המודל.

**Load_Model**: טעינת המודל.

**Running_Algorithm**: מחזיר תוצאות לאחר הפעלת המודל.

3) **ID3.py-** הקובץ מכיל מימוש עצמי לאלגוריתם ID3.

**<u>פונקציות מוכלות:</u>**

**read_structure**: החזרת קובץ structure כמילון.

**Get_Decision_Tree**: בניית מודל ה-ID3.

**Classification_Row**: החזרת סיווג של שורה מסויימת.

**ID3**: קוראת לשיטה Classification_Row ומעבירה לה קובץ אימון ומחזירה את המודל.

**Testing_model**: מתודה שמסווגת קובץ מסויים.

4) **<u>NaiveBayes.py</u>-** יש בקובץ מימוש עצמי של אלגוריתם naive bayes.

**<u>פונקציות מוכלות:</u>**

**ReadCsv**: קריאת קובת CSV.

**ReadStructure**: החזרת קובץ structure כמילון.

**make_prod**: החזרת מכפלה קארטזית של שני tuples.

**Build_provavility_For_One_Column**: בנייה והחזרת מילון של הסתברויות לעמודה.

**Probability**: החזרת הסתברות למאפיין מתוך טבלת ההסתברויות.

**conditional_probability**: החזרת הסתברות מותנית.

**NaiveBayes**: בניית המודל על קובץ ה- train.

**Testing_model**: מתודה שמסווגת קובץ מסויים.

5) **<u>Entropy.py</u>-** בקובץ זה מתבצעים חישובי אינטרופיה.

**<u>פונקציות מוכלות:</u>**

**entropy**: פונקציה המחשבת אנטרופיה.

**InfoGain**: פונקציה שמחשבת את ה- info gain של מערך נתונים ביחס למערך אחר.

**Conditional_Entropy:** פונקציה שמחשבת אנטרופיה מותנית.

**Mutual_Information**: פונקציה שמחשבת אנטרופיה משותפת.

6) **<u>Binning.py</u>**: קובץ האחראי על דיסקריטיזציות.

**<u>מחלקת Binning:</u> מחלקה זו אחראית על דיסקרטיזציה של כל הסוגים**,
מכילה גם את ה-

data frame ואת כמות ה- bins שהתקבלו בקלט.

**<u>מטודות מוכלות:</u>**

**Equal_Frequency**: מבצעת דיסקריטיזציה לפי תדר שווה.

**Equal_Width**: מבצעת דיסקריטיזציה לפי רוחב שווה.

**Entropy_Discretization**: מבצעת אנטרופיה לפי הדיסקרטיזציה.

**built_Enropy_Discretization**: מבצעת אנטרופיה בעזרת הפונקציה
built_EntropyBased.

7) **EntropyTree.py**: בקובץ זה נבנה עץ האנטרופיה.

**מחלקת EntropyTree:**

**מטודות מוכלות:**

**Getroot**: החזרת השורש של העץ.

**getSplit**: החזרת נקודת פיצול.

**getLeft**: החזרת בן שמאלי.

**getRight**: החזרת בן ימני.

**getLeafs:** פונקציה שמחזירה את כל העלים.

**getNodes**: החזרת צמתים מבלי העלים.

**getLevel_h**: החזרת רמה אחת לפני האחרונה.

**SetLeft:** מציבים אובייקט של EntropyTree בצומת שמאלית.

**setRight:** מציבים אובייקט של EntropyTree בצומת שמאלית.

**setSplit:** נקודת פיצול של המידע.

**setEntropy:** הגדרת האנטרופיה של המידע.

**isLeaf**: החזרת קודקוד במידה והוא עלה.

In [5]:

```
#EDA
import pandas as pd
df= pd.read_csv("train.csv")
df.head(10)
```

Out[5]:

| | age | job | marital | education | default | balance | housing | loan | contact | day | month | duration | campaign | previous | poutcome | class |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 58 | management | married | tertiary | no | 2143.0 | yes | no | unknown | 5 | may | 261.0 | 1 | 0 | unknown | no |
| **1** | 44 | technician | single | secondary | no | 29.0 | yes | no | unknown | 5 | may | 151.0 | 1 | 0 | unknown | no |
| **2** | 33 | entrepreneur | married | secondary | no | 2.0 | yes | yes | unknown | 5 | may | 76.0 | 1 | 0 | unknown | no |
| **3** | 47 | blue-collar | married | unknown | no | 1506.0 | yes | no | unknown | 5 | may | 92.0 | 1 | 0 | unknown | no |
| **4** | 33 | unknown | single | unknown | no | 1.0 | no | no | unknown | 5 | may | 198.0 | 1 | 0 | unknown | no |
| **5** | 35 | management | married | tertiary | no | 231.0 | yes | no | unknown | 5 | may | 139.0 | 1 | 0 | unknown | no |
| **6** | 28 | management | single | tertiary | no | 447.0 | yes | yes | unknown | 5 | may | 217.0 | 1 | 0 | unknown | no |
| **7** | 42 | entrepreneur | divorced | tertiary | yes | 2.0 | yes | no | unknown | 5 | may | 380.0 | 1 | 0 | unknown | no |
| **8** | 58 | retired | married | primary | no | 121.0 | yes | no | unknown | 5 | may | 50.0 | 1 | 0 | unknown | no |
| **9** | 43 | technician | single | secondary | no | 593.0 | yes | no | unknown | 5 | may | 55.0 | 1 | 0 | unknown | no |

In [14]:

```
#removing useless columns
df = df.drop(['day','month','campaign','previous','poutcome','balance',], axis=1)
df.head(10)
```

Out[14]:

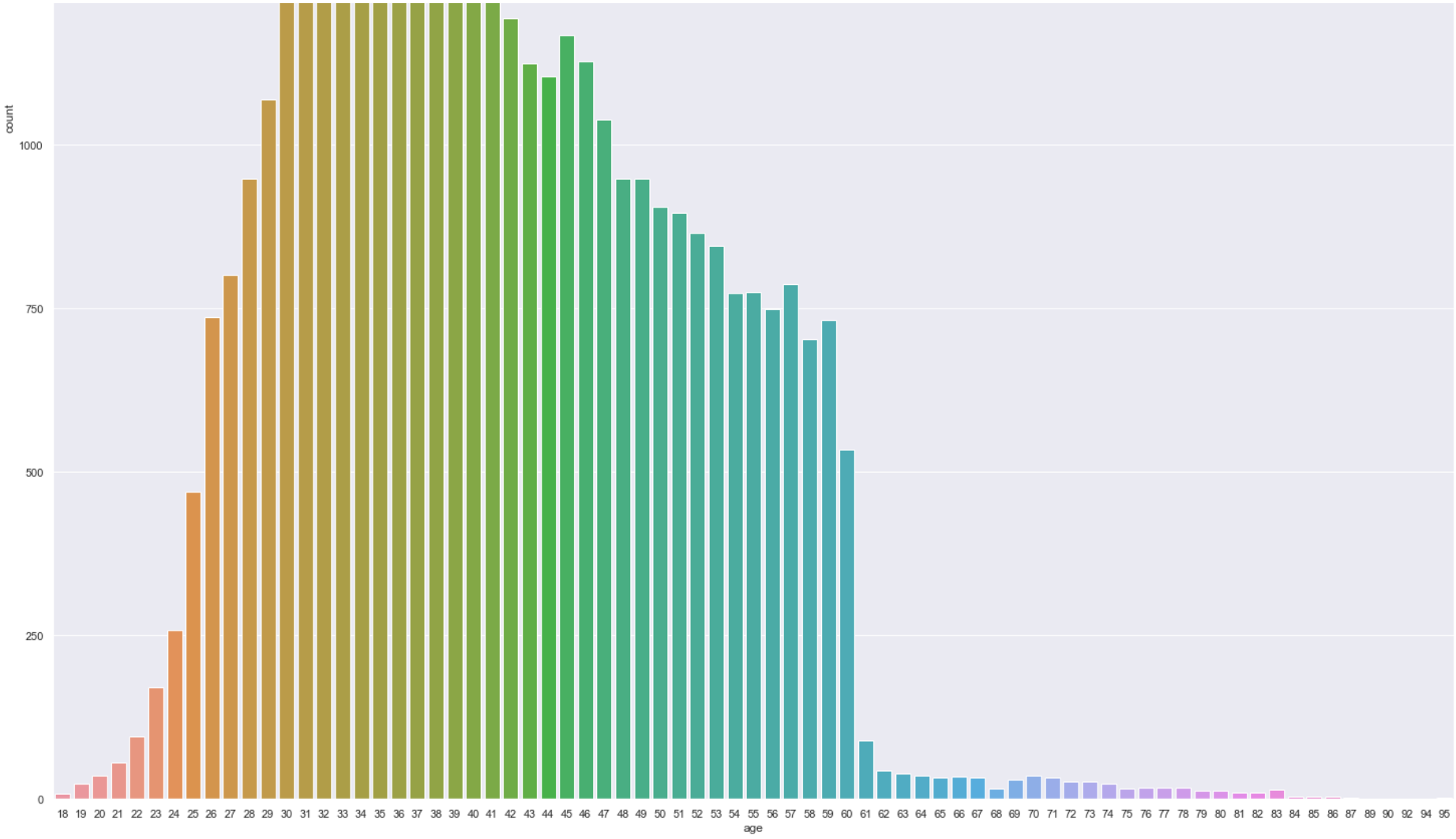
| | age | job | marital | education | default | housing | loan | contact | duration | class |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 58 | management | married | tertiary | no | yes | no | unknown | 261.0 | no |
| **1** | 44 | technician | single | secondary | no | yes | no | unknown | 151.0 | no |
| **2** | 33 | entrepreneur | married | secondary | no | yes | yes | unknown | 76.0 | no |
| **3** | 47 | blue-collar | married | unknown | no | yes | no | unknown | 92.0 | no |
| **4** | 33 | unknown | single | unknown | no | no | no | unknown | 198.0 | no |
| **5** | 35 | management | married | tertiary | no | yes | no | unknown | 139.0 | no |

| | age | job | marital | education | default | housing | loan | contact | duration | class |
|---|---|---|---|---|---|---|---|---|---|---|
| **6** | 28 | management | single | tertiary | no | yes | yes | unknown | 217.0 | no |
| **7** | 42 | entrepreneur | divorced | tertiary | yes | yes | no | unknown | 380.0 | no |
| **8** | 58 | retired | married | primary | no | yes | no | unknown | 50.0 | no |
| **9** | 43 | technician | single | secondary | no | yes | no | unknown | 55.0 | no |

In [3]:
```
import seaborn as sns
sns.set(color_codes=True)
import matplotlib.pyplot as plt
%matplotlib inline
```

In [125…
```
plt.figure(figsize=(25,25))
sns.countplot(x='age',data=df)
```

Out[125…
```
<AxesSubplot:xlabel='age', ylabel='count'>
```

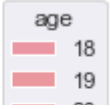


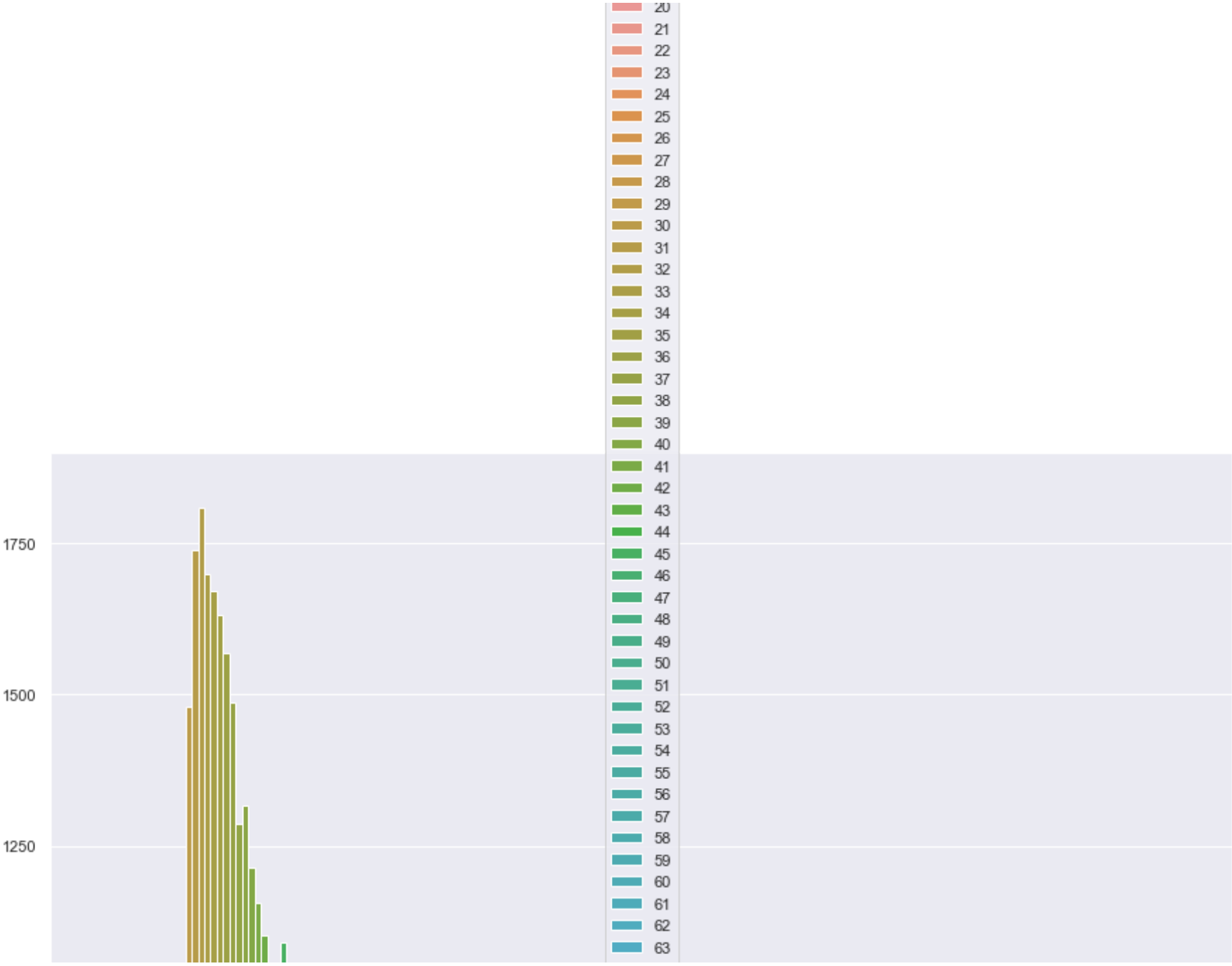

In [123…

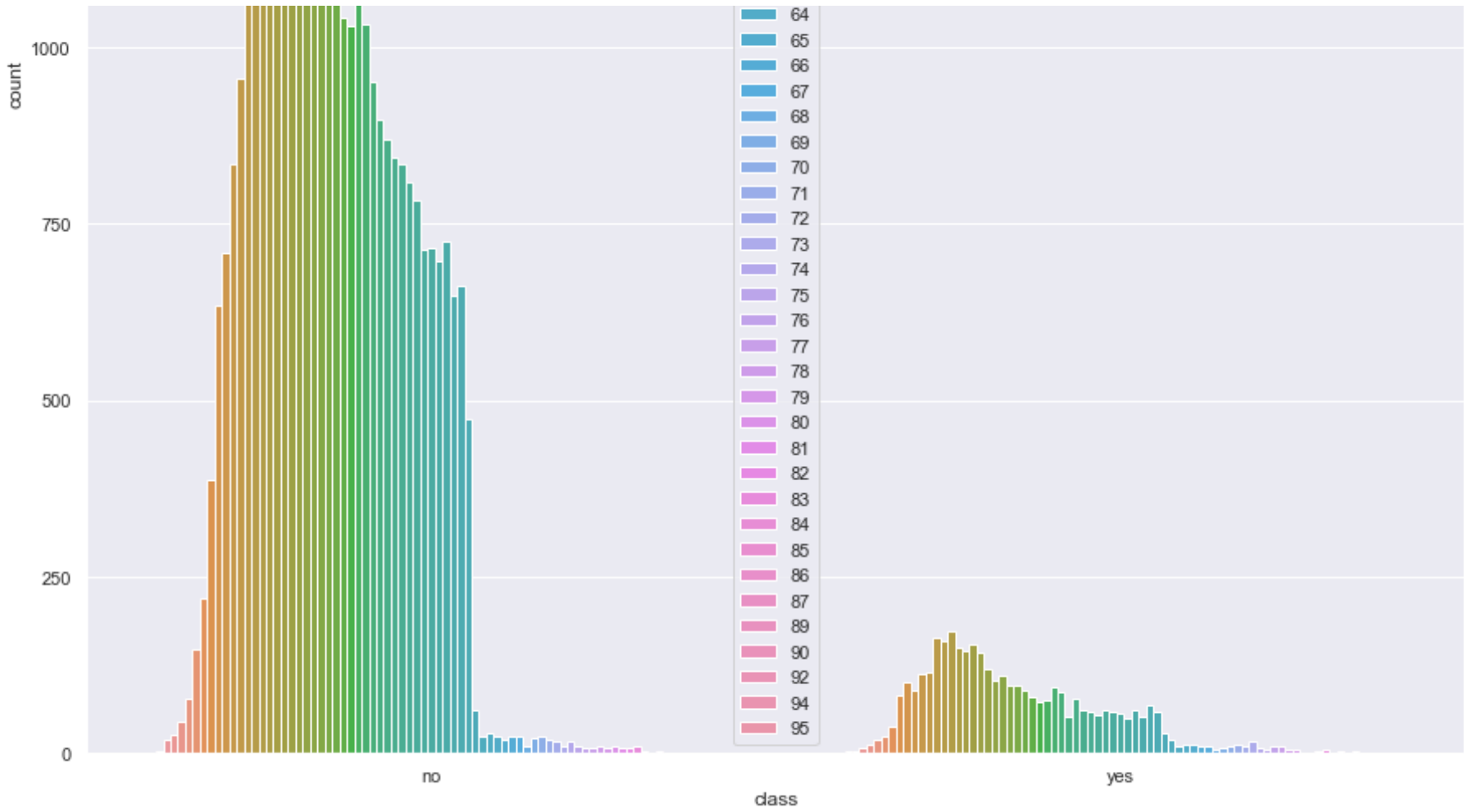
```
plt.figure(figsize=(15,15))
sns.countplot(x='class',hue='age',data=df)
```

Out[123…

```
<AxesSubplot:xlabel='class', ylabel='count'>
```

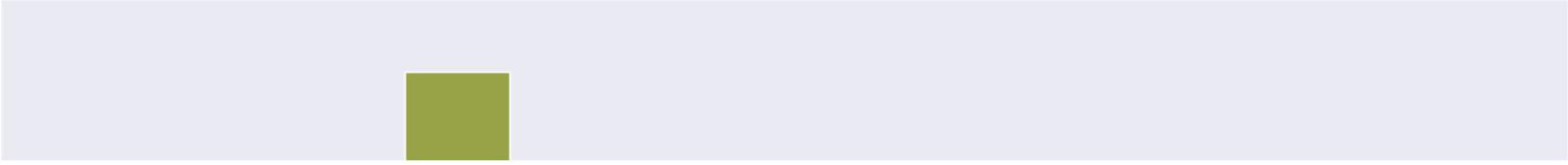


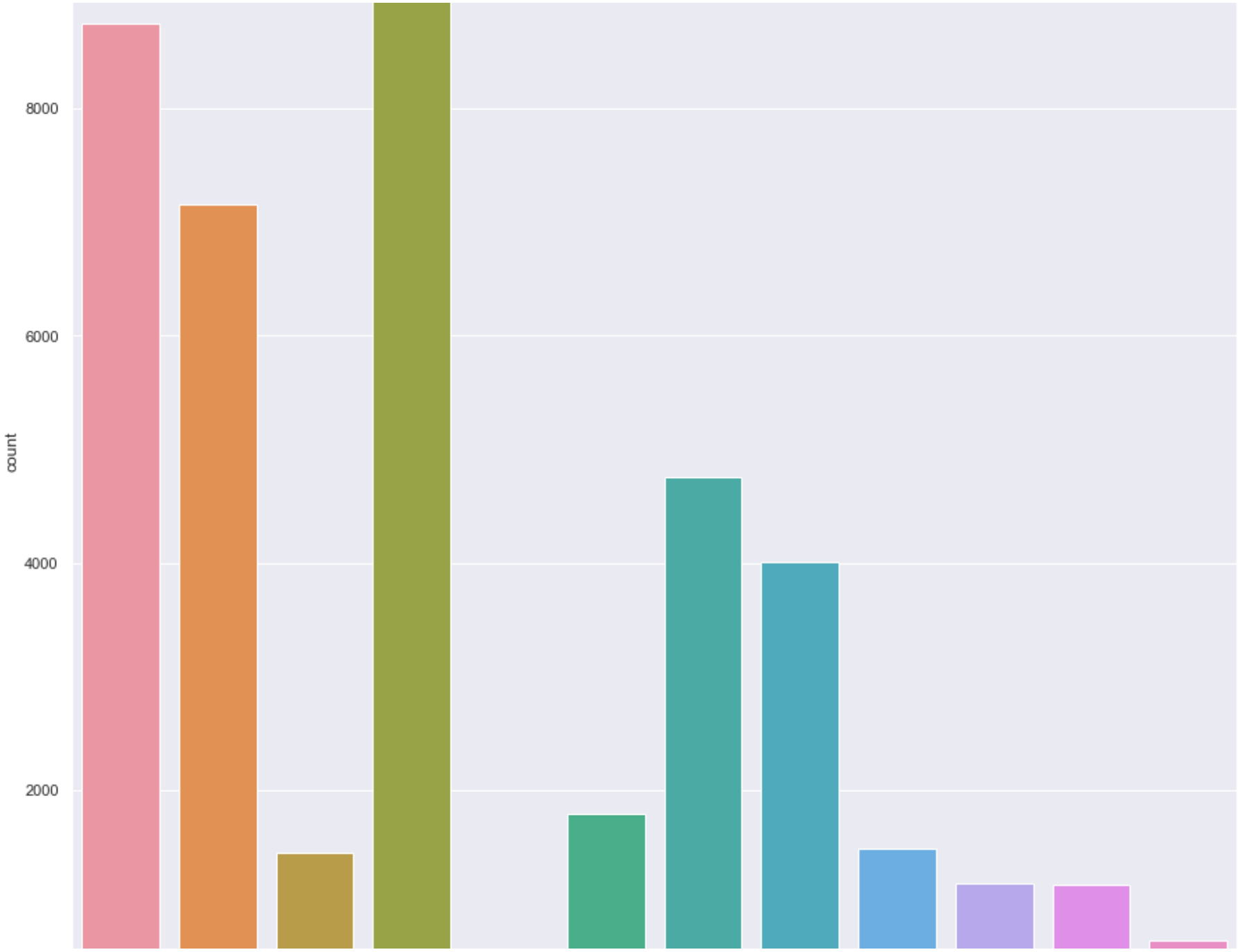


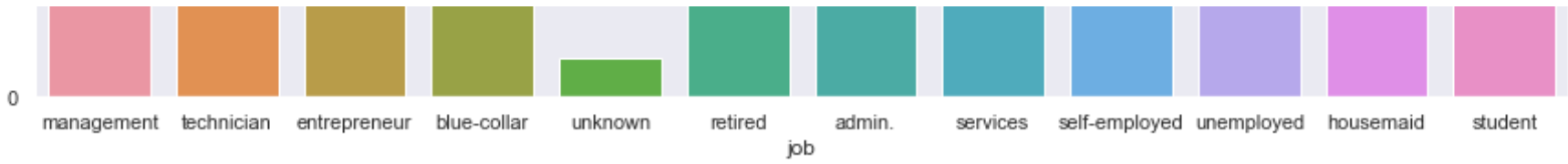



In [85]:
```
plt.figure(figsize=(15,15))
sns.countplot(x='job',data=df)
```

Out[85]: <AxesSubplot:xlabel='job', ylabel='count'>



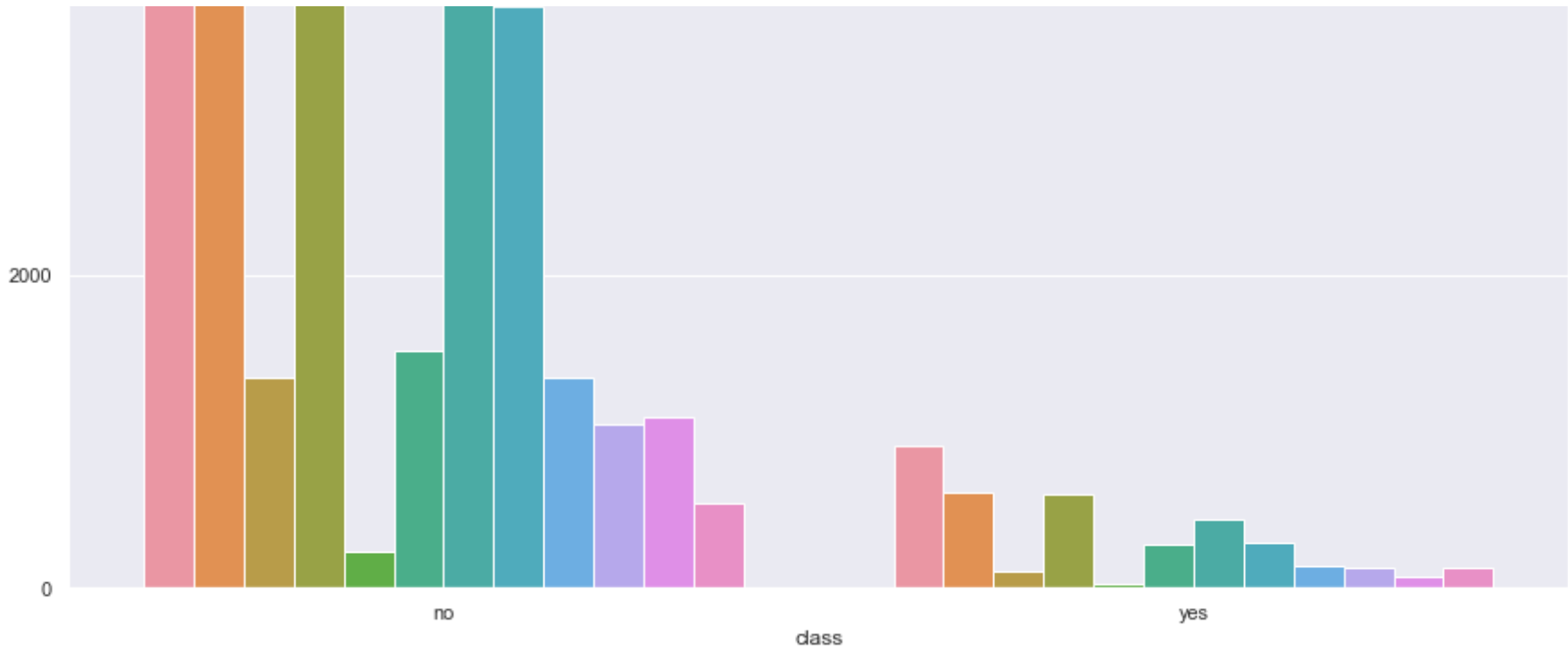





In [126…

```
plt.figure(figsize=(15,15))
sns.countplot(x='class',hue='job',data=df)
```

Out[126… <AxesSubplot:xlabel='class', ylabel='count'>

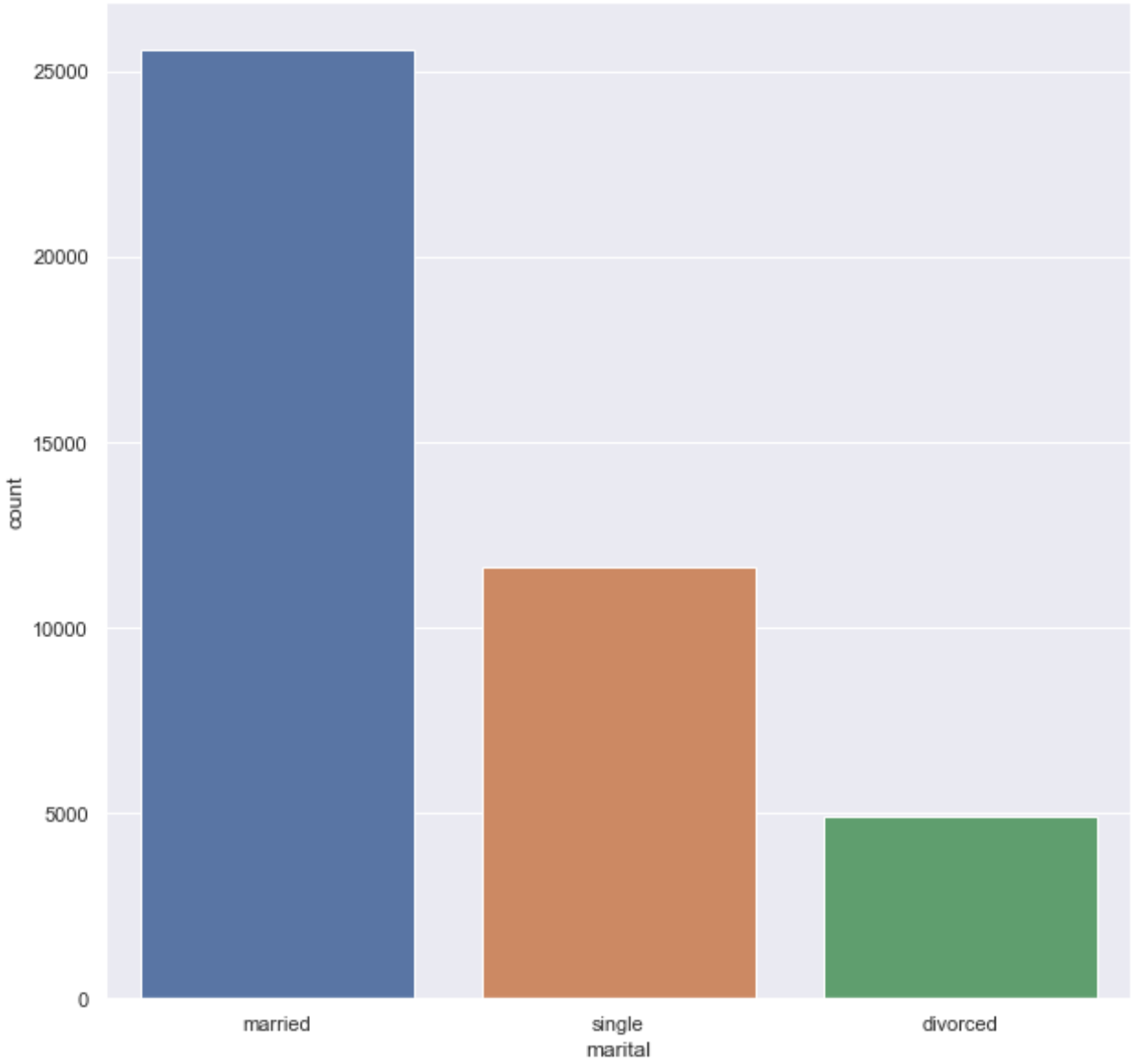



In [120…

```
plt.figure(figsize=(10,10))
sns.countplot(x='marital',data=df)
```

Out[120…

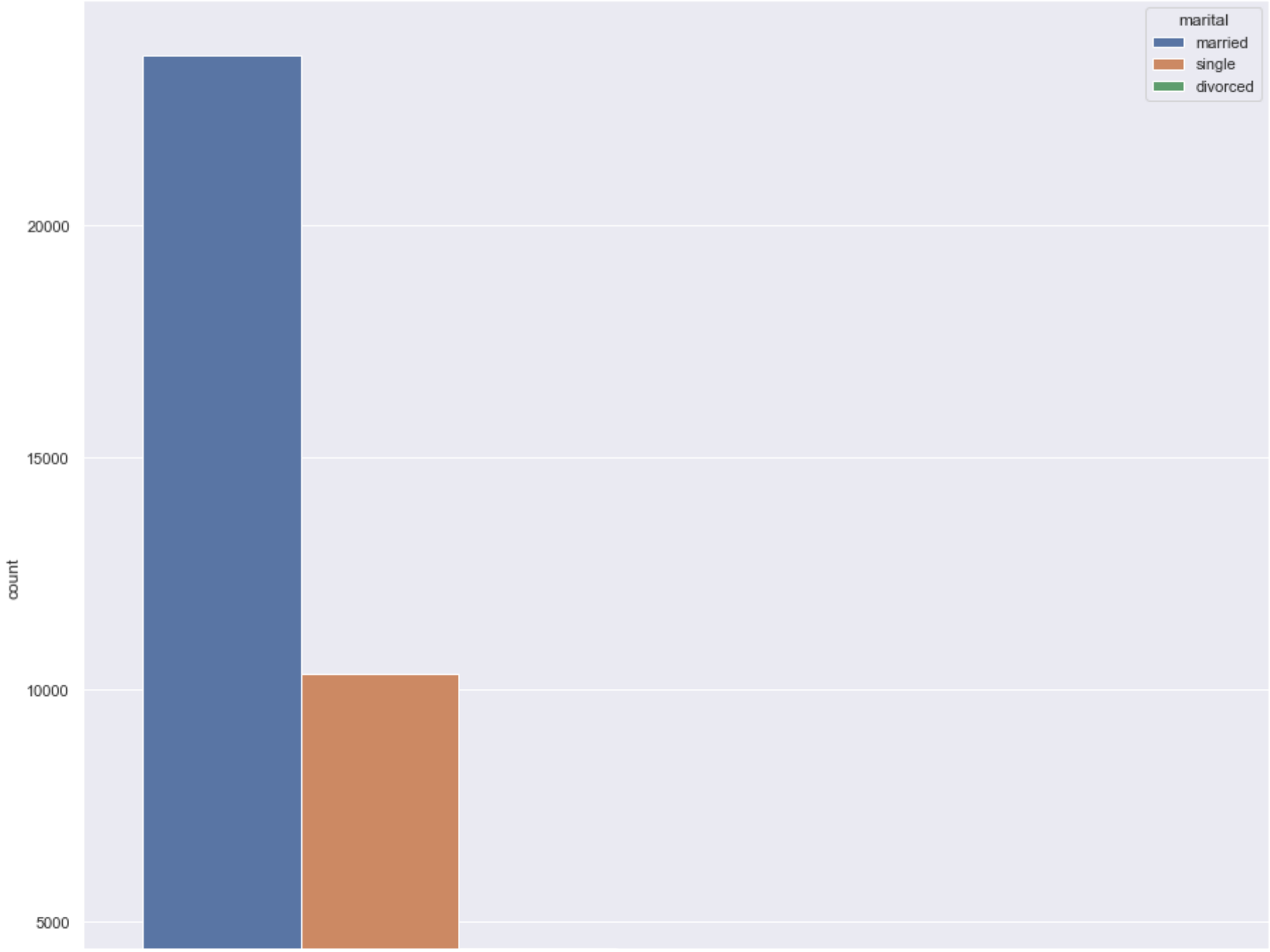
```
<AxesSubplot:xlabel='marital', ylabel='count'>
```

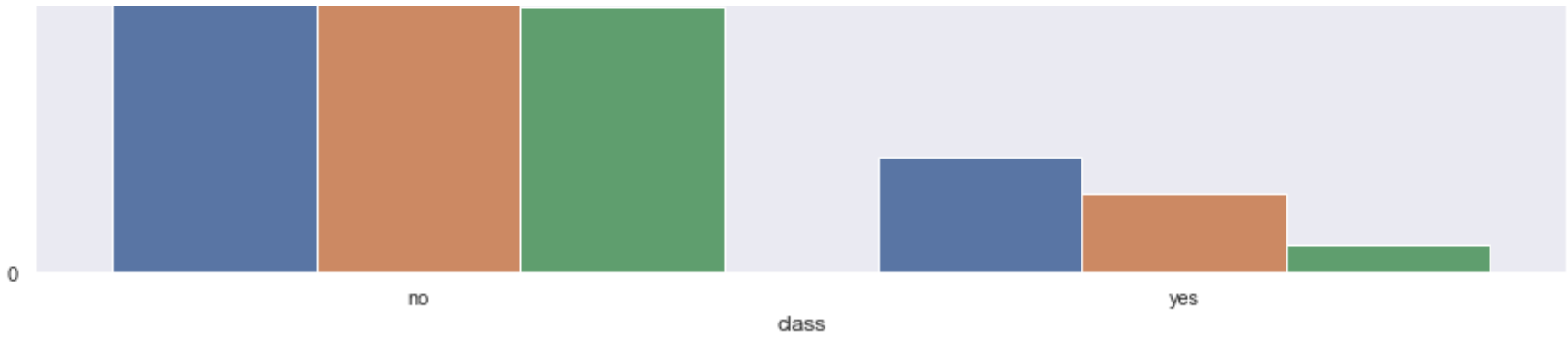



In [130…

```
plt.figure(figsize=(15,15))
sns.countplot(x='class',hue='marital',data=df)
```

Out[130… <AxesSubplot:xlabel='class', ylabel='count'>

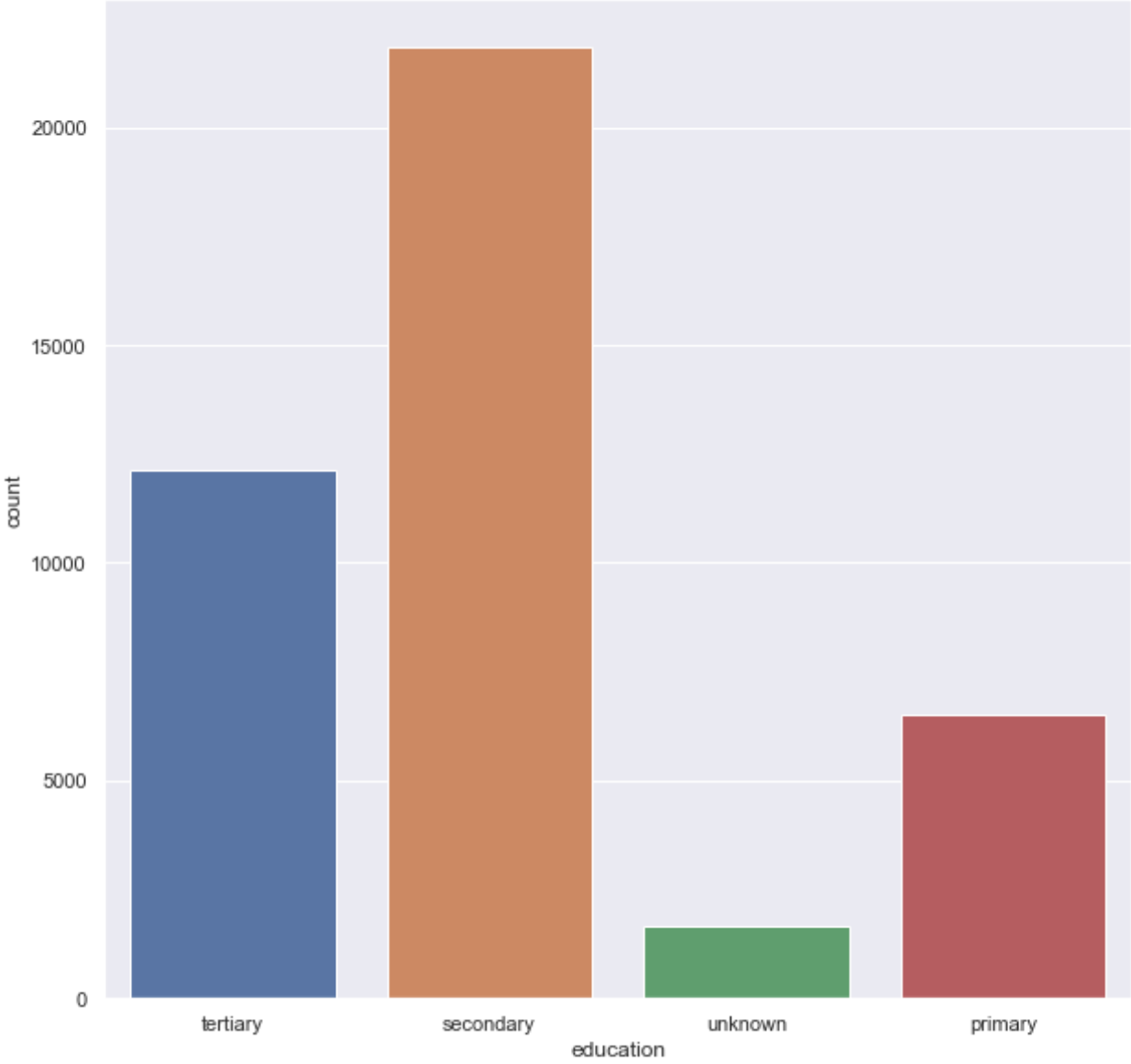





In [119…

```
plt.figure(figsize=(10,10))
sns.countplot(x='education',data=df)
```

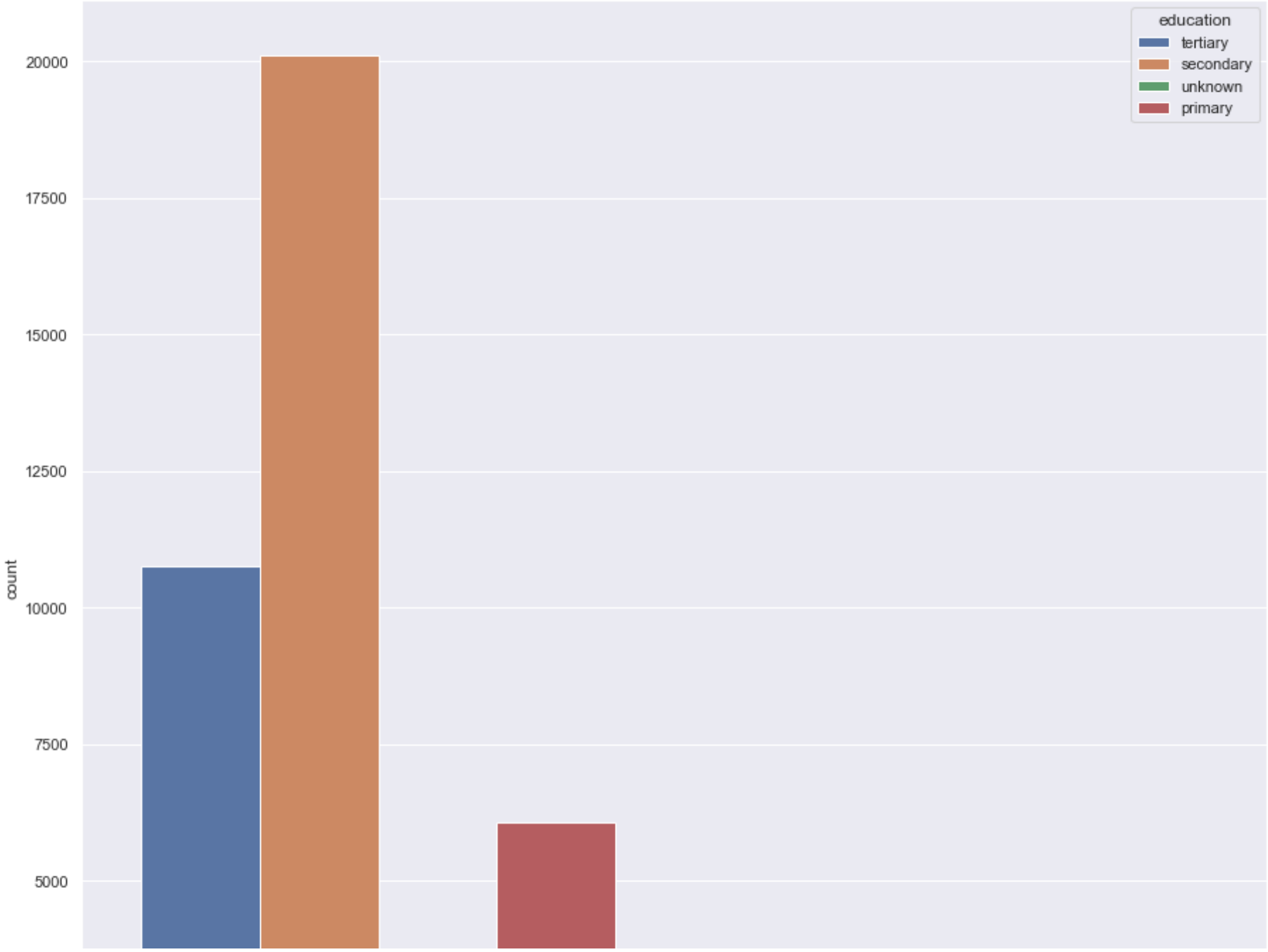
Out[119… <AxesSubplot:xlabel='education', ylabel='count'>

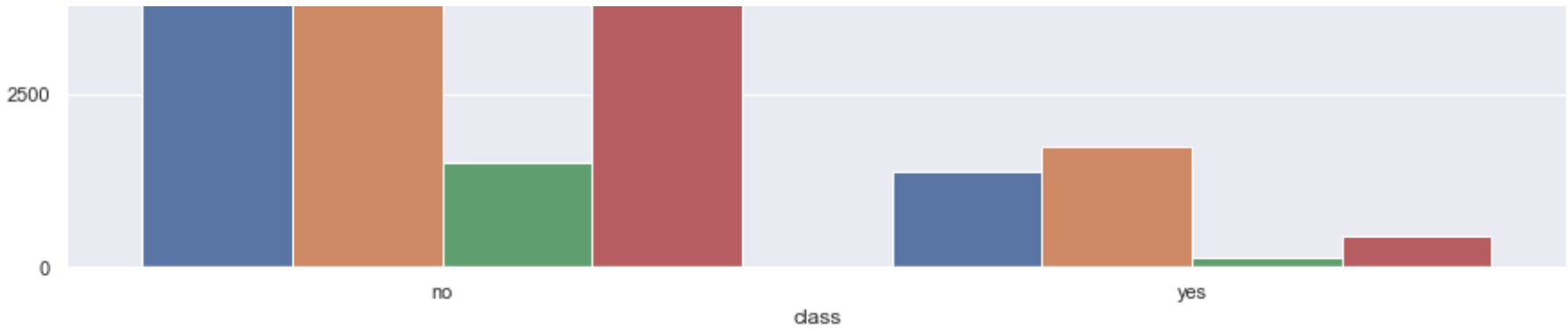



In [131…

```
plt.figure(figsize=(15,15))
sns.countplot(x='class',hue='education',data=df)
```

Out[131… <AxesSubplot:xlabel='class', ylabel='count'>

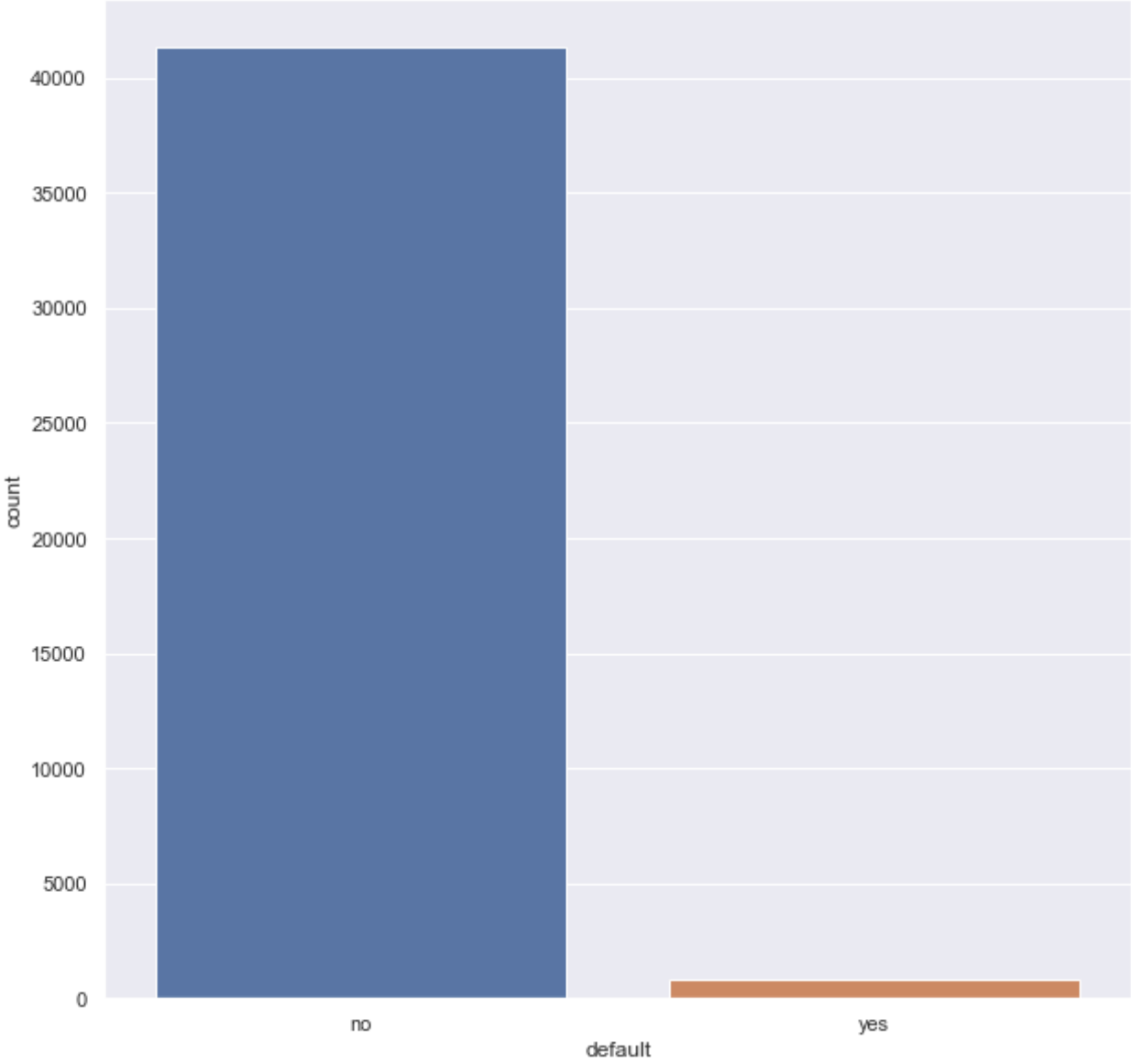





In [118…

```
plt.figure(figsize=(10,10))
sns.countplot(x='default',data=df)
```

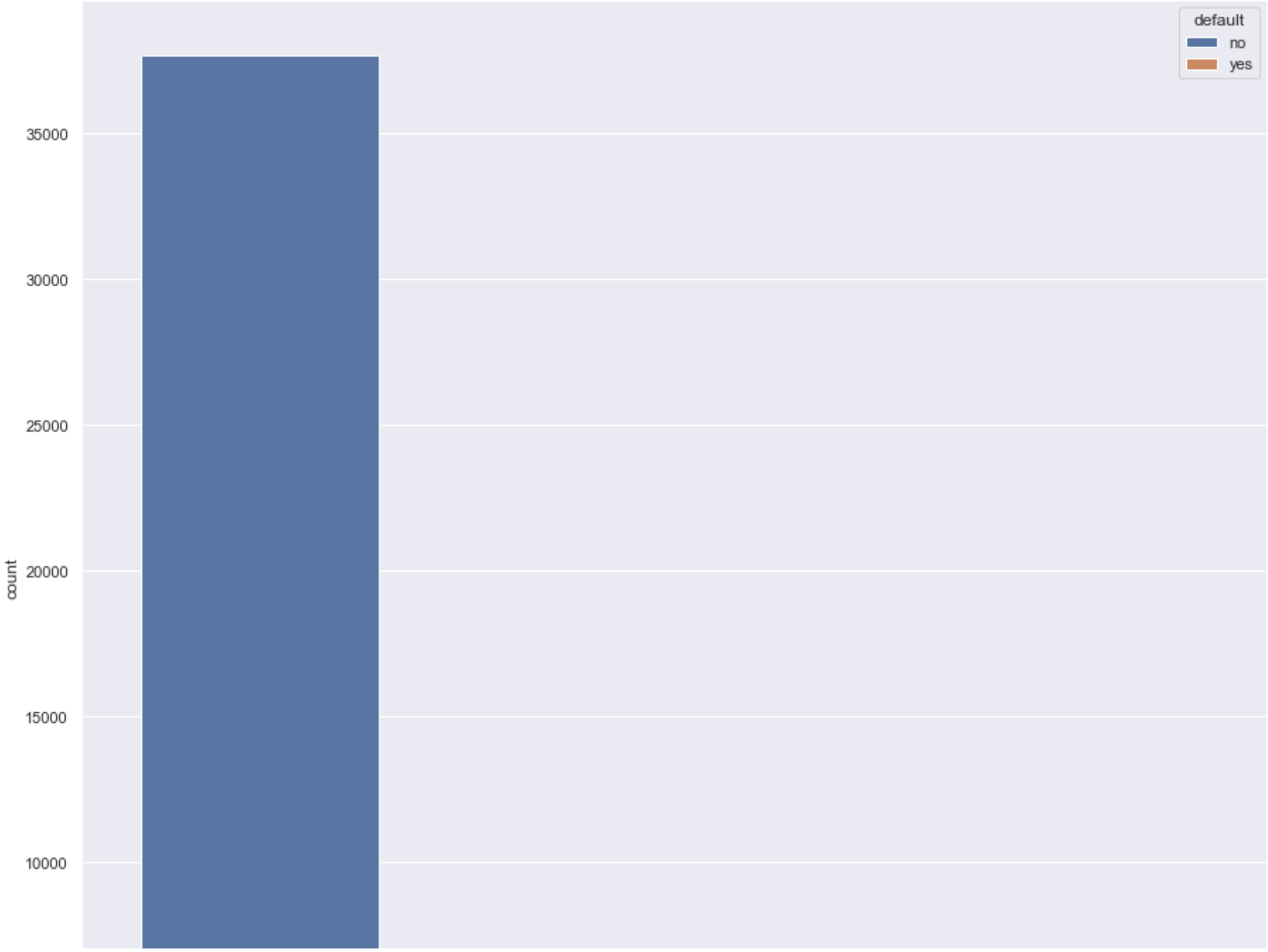
Out[118… <AxesSubplot:xlabel='default', ylabel='count'>

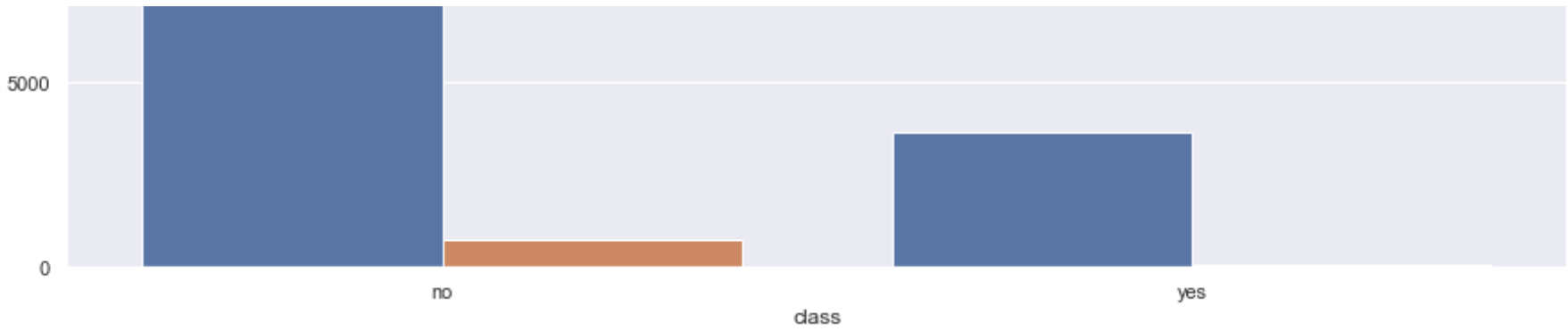



In [134…

```
plt.figure(figsize=(15,15))
sns.countplot(x='class',hue='default',data=df)
```

Out[134… <AxesSubplot:xlabel='class', ylabel='count'>

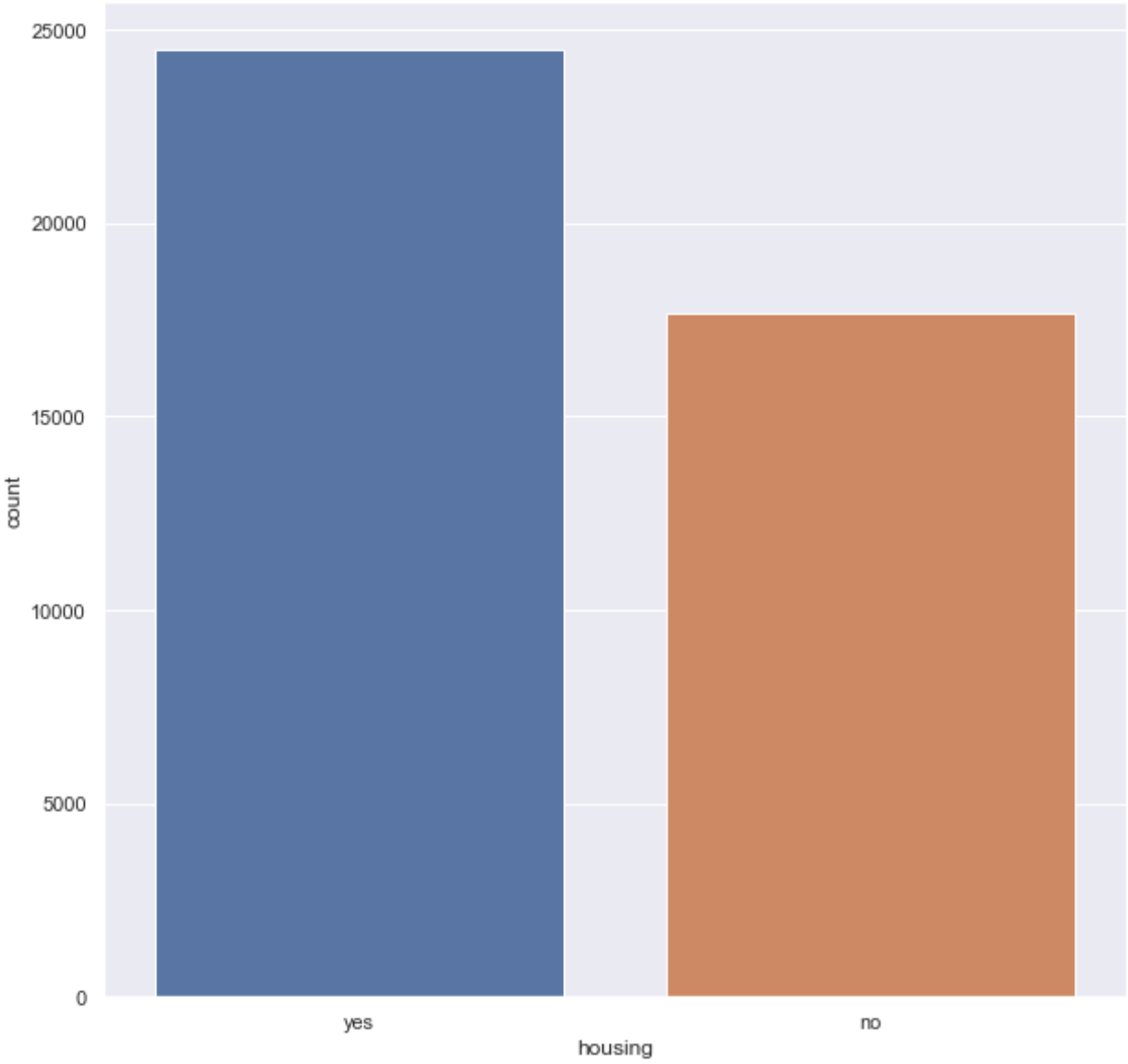





In [117…

```
plt.figure(figsize=(10,10))
sns.countplot(x='housing',data=df)
```

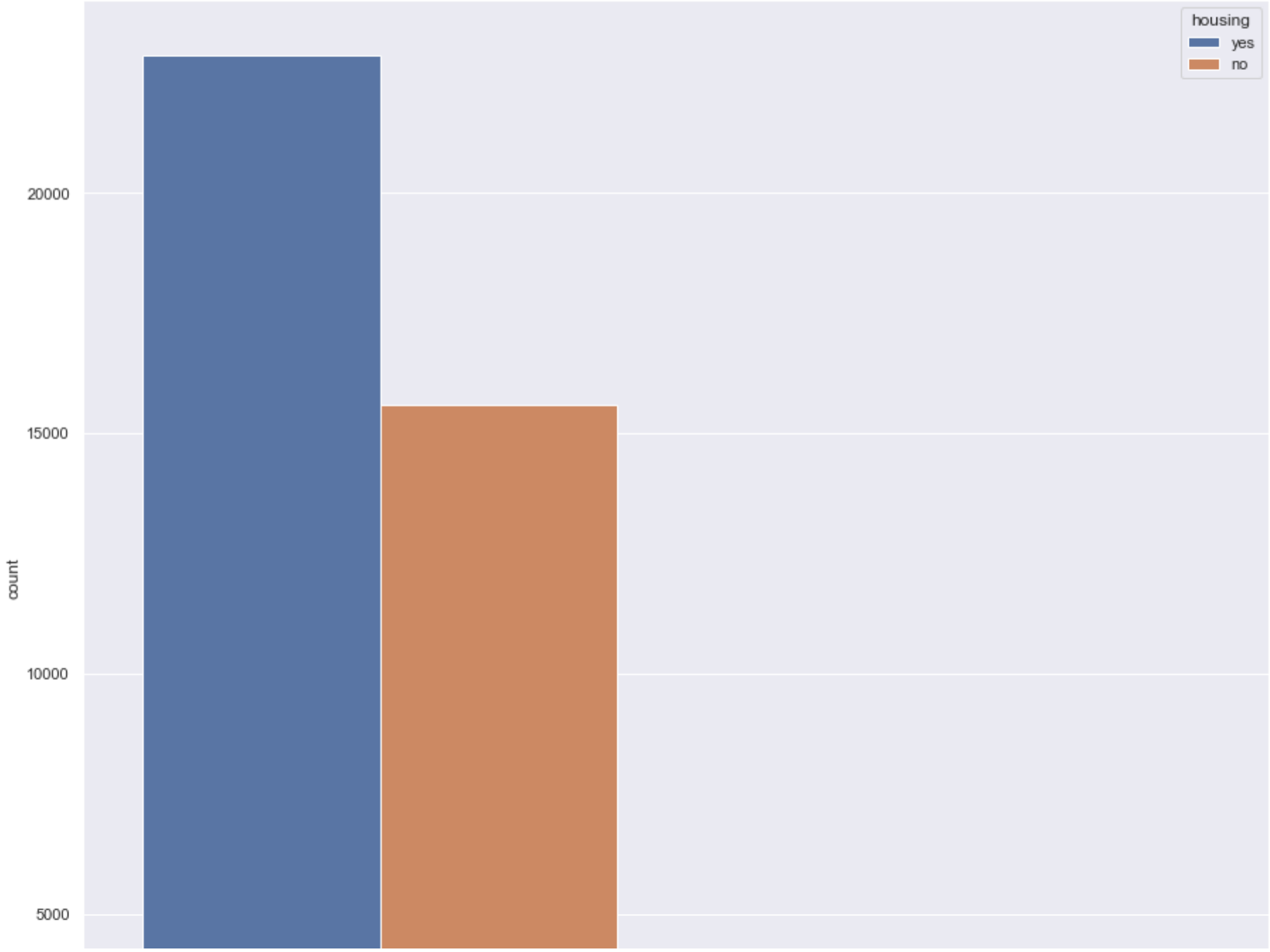
Out[117… <AxesSubplot:xlabel='housing', ylabel='count'>

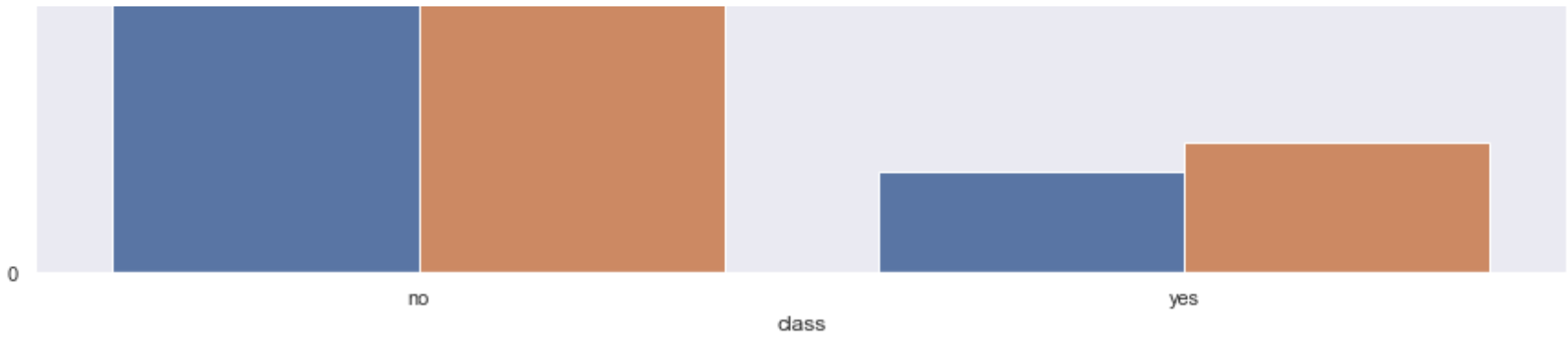



In [135…

```
plt.figure(figsize=(15,15))
sns.countplot(x='class',hue='housing',data=df)
```

Out[135… <AxesSubplot:xlabel='class', ylabel='count'>



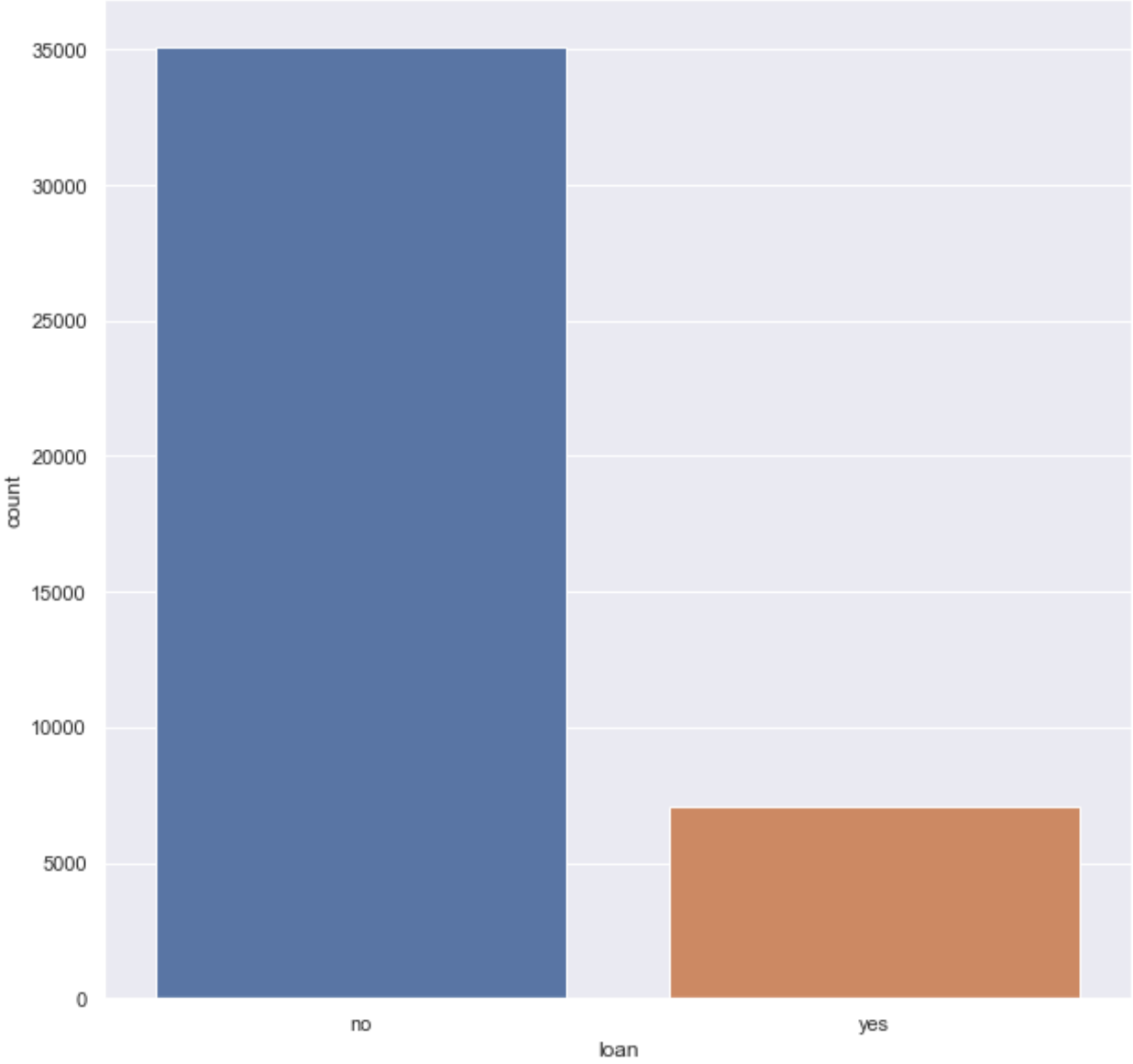



In [116…

```
plt.figure(figsize=(10,10))
sns.countplot(x='loan',data=df)
```

Out[116…

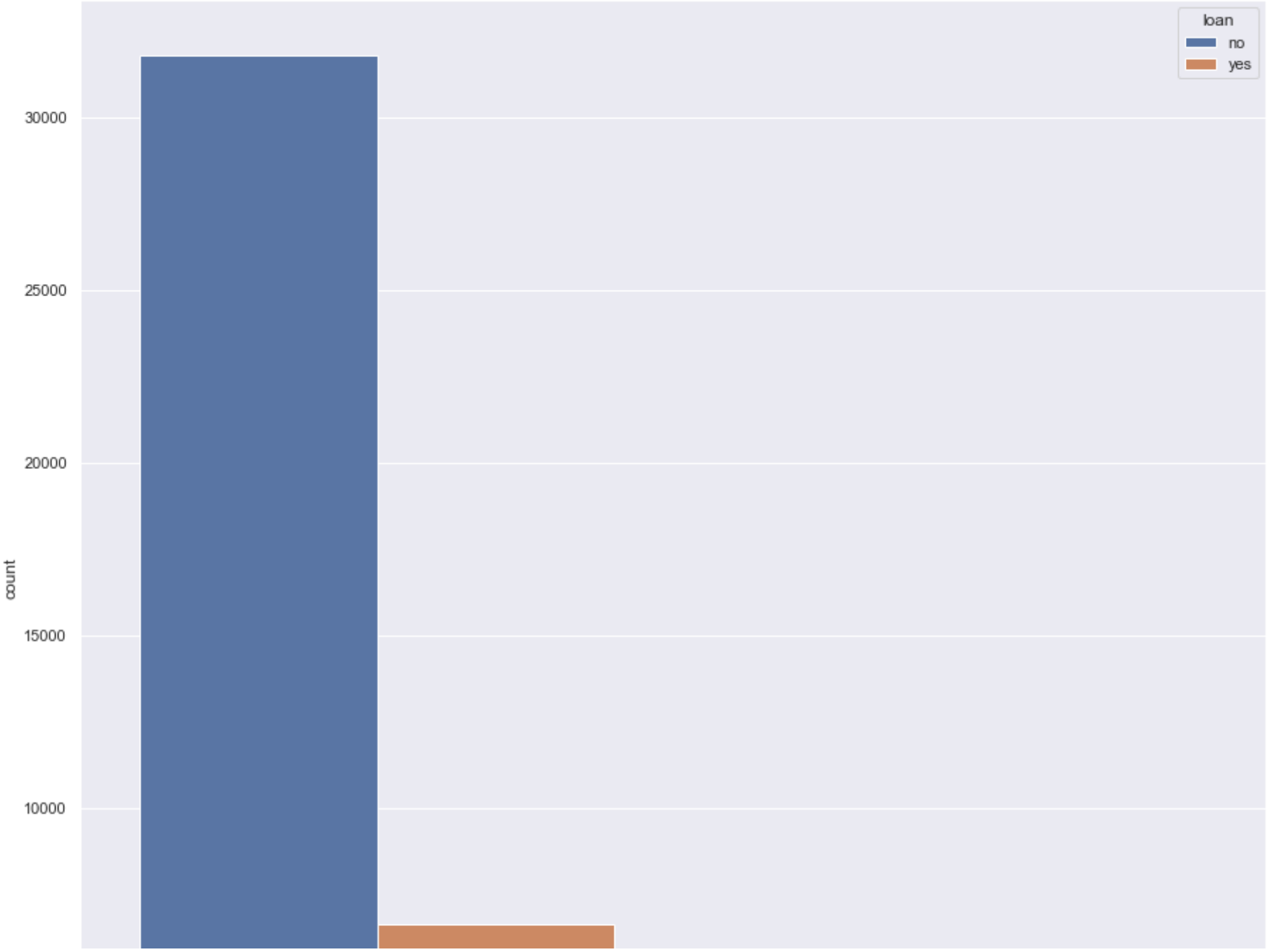
```
<AxesSubplot:xlabel='loan', ylabel='count'>
```

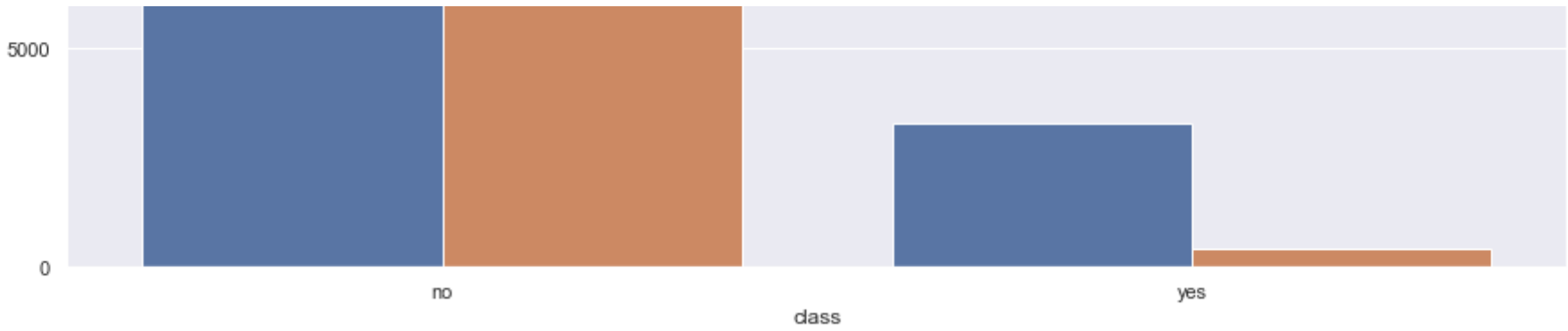



In [137…

```
plt.figure(figsize=(15,15))
sns.countplot(x='class',hue='loan',data=df)
```

Out[137… <AxesSubplot:xlabel='class', ylabel='count'>

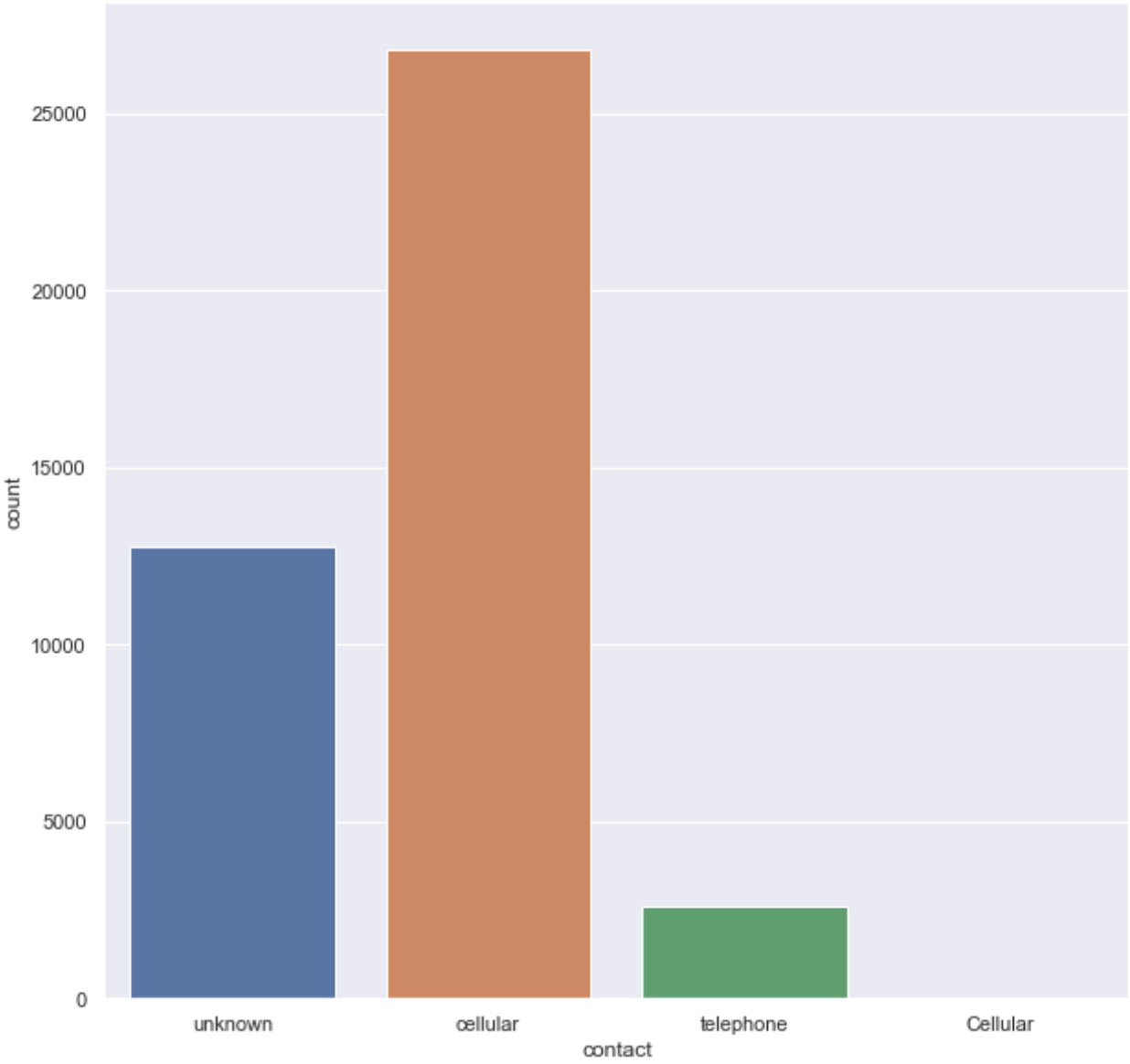





In [115…

```
plt.figure(figsize=(10,10))
sns.countplot(x='contact',data=df)
```

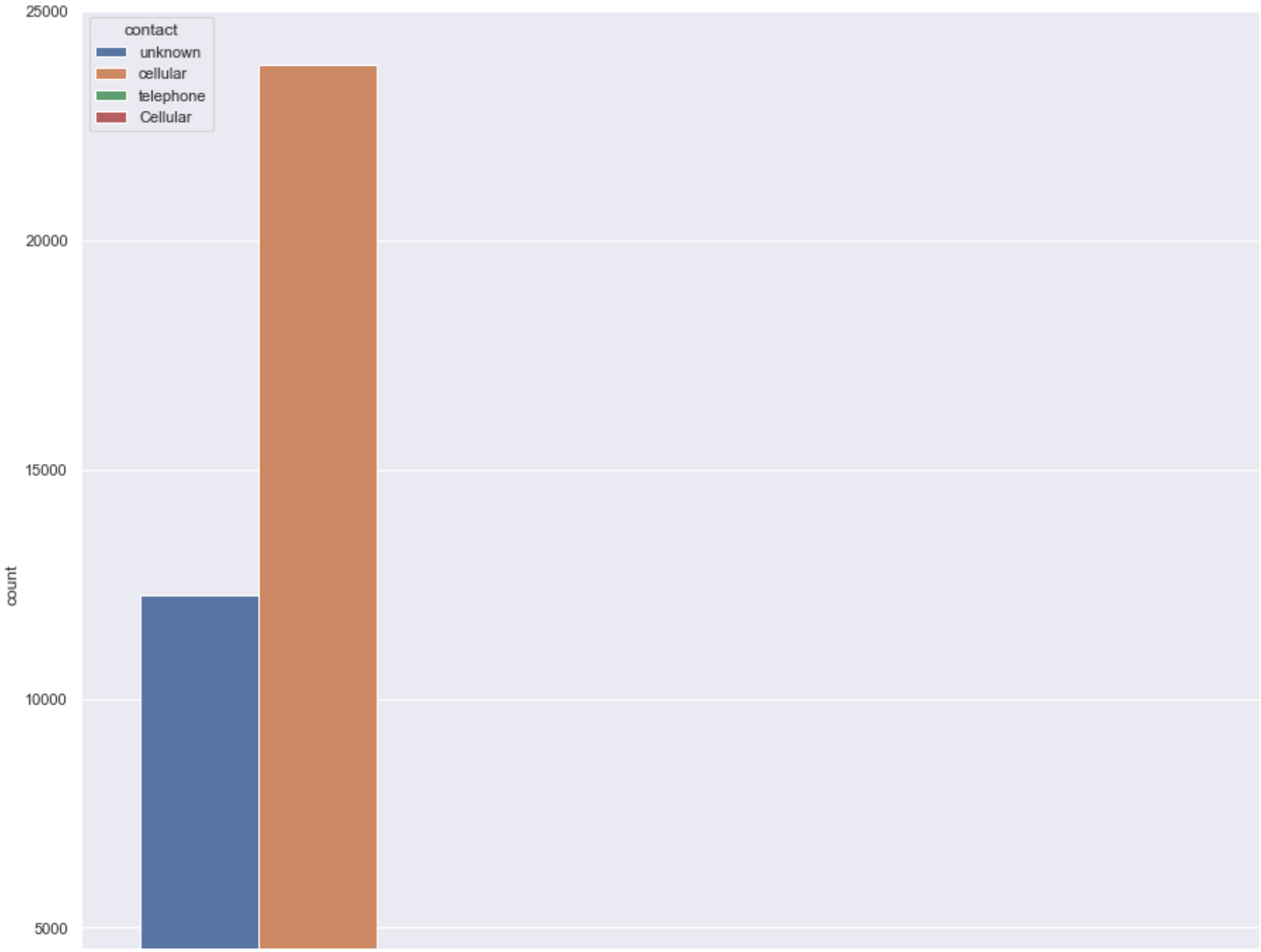
Out[115… <AxesSubplot:xlabel='contact', ylabel='count'>

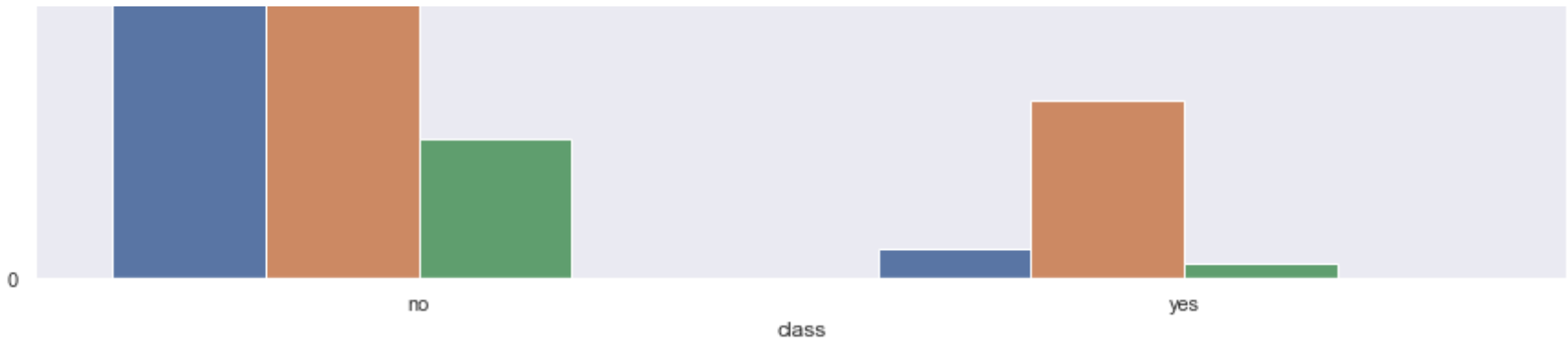



In [138…

```
plt.figure(figsize=(15,15))
sns.countplot(x='class',hue='contact',data=df)
```

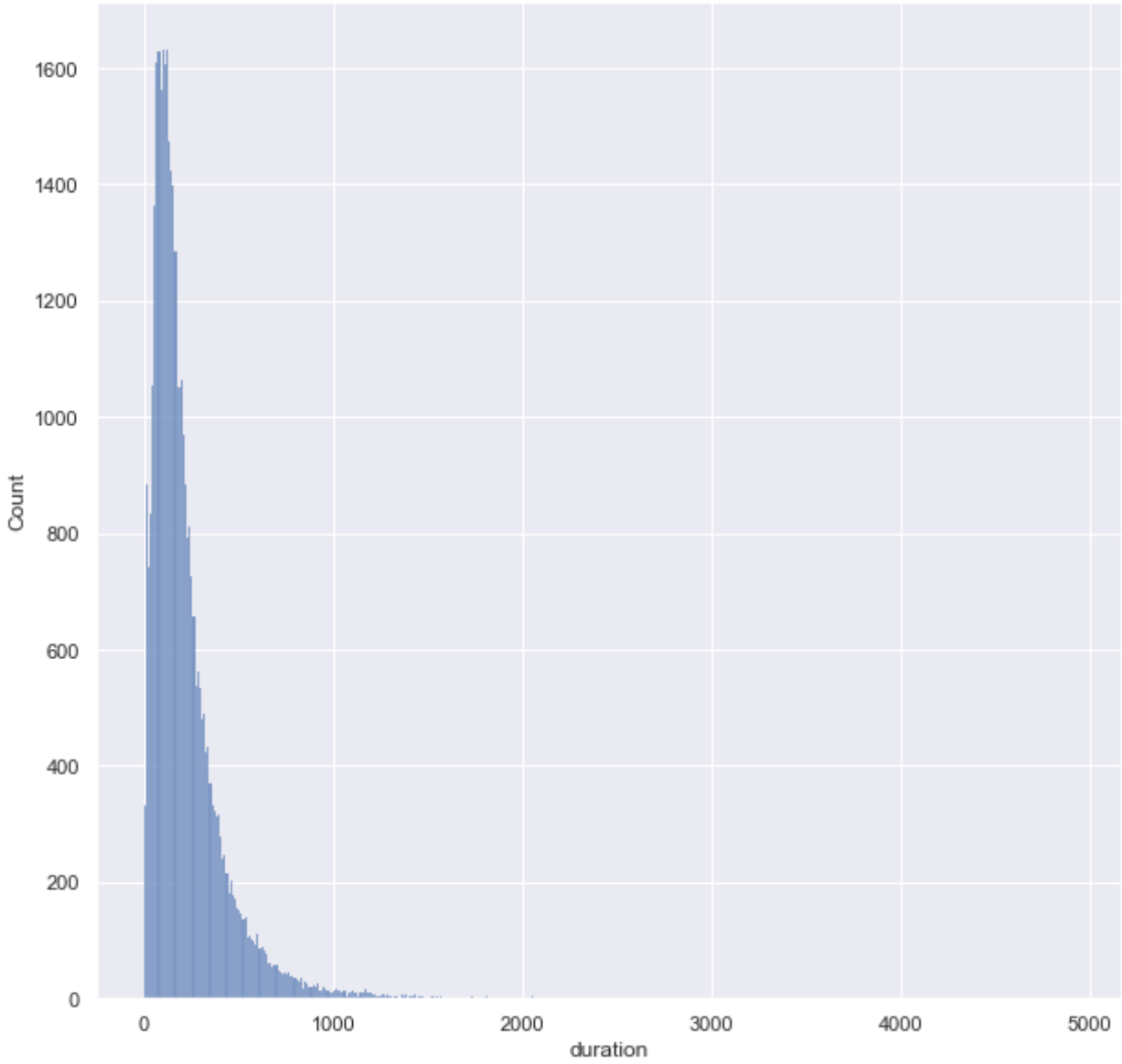
Out[138… <AxesSubplot:xlabel='class', ylabel='count'>






In [13]:
```
classification = df[df['class'] == 'no']
plt.figure(figsize=(10,10))
sns.histplot(classification['duration'])
```

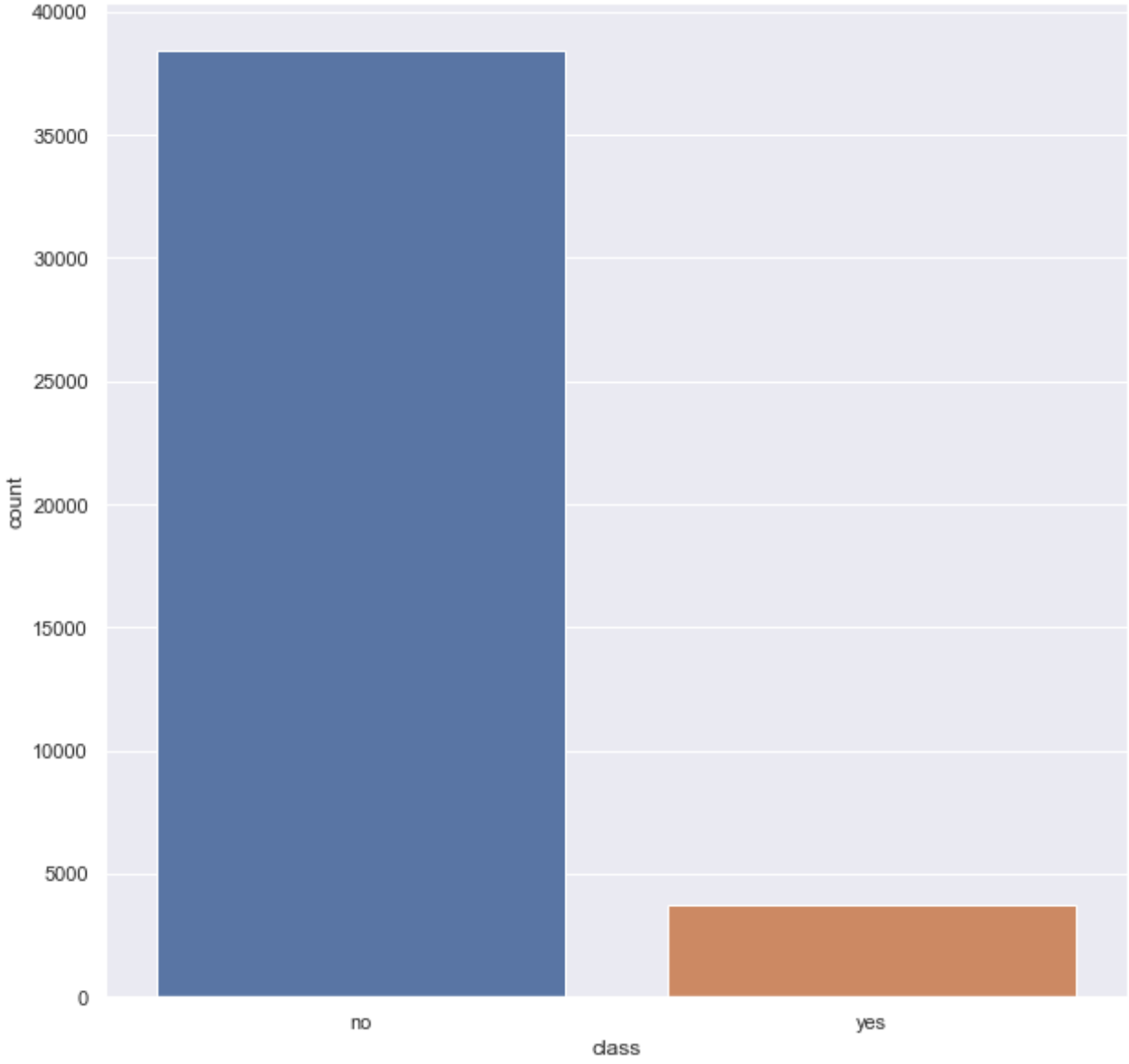
Out[13]:
```
<AxesSubplot:xlabel='duration', ylabel='Count'>
```




In [141…

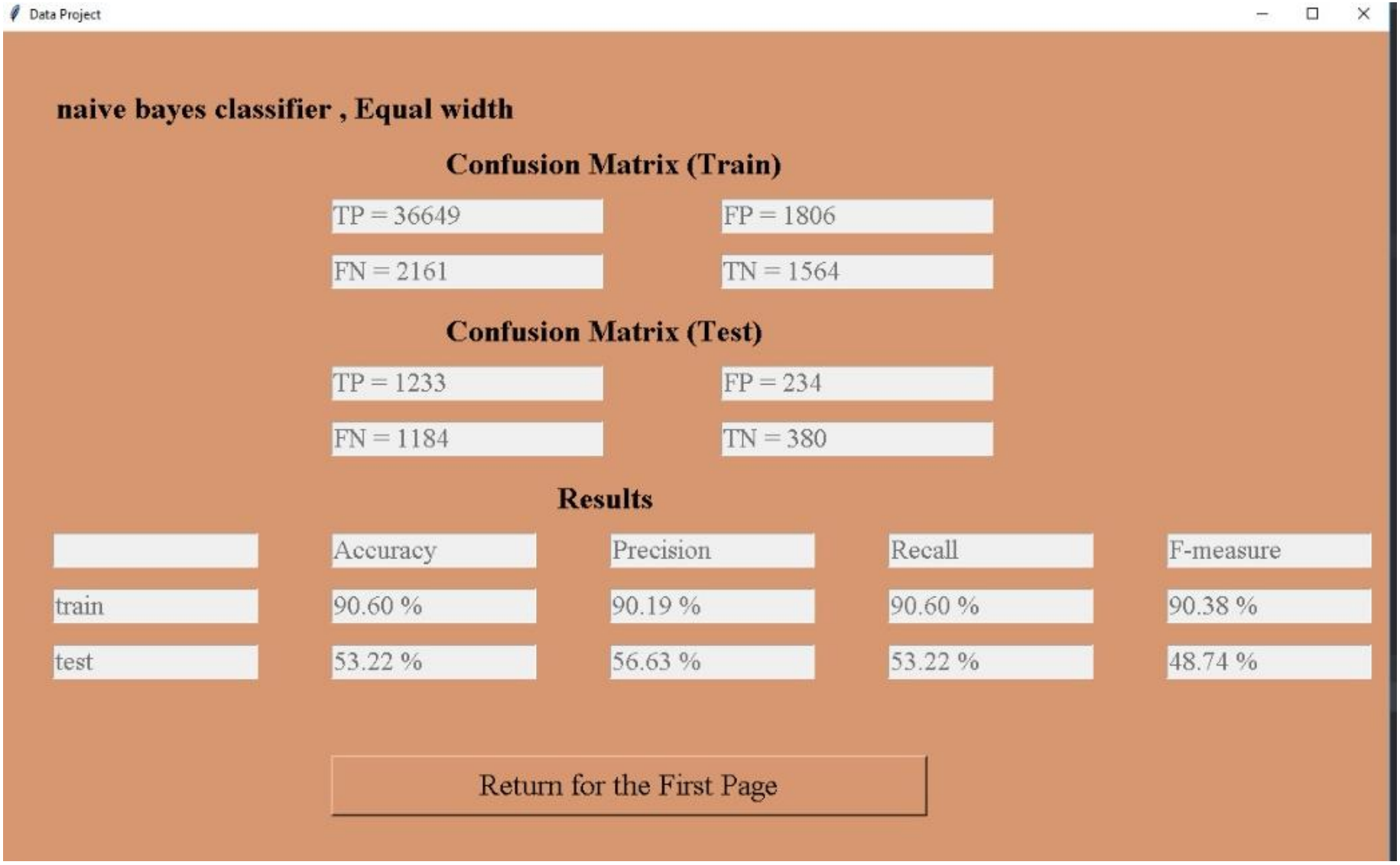
```
plt.figure(figsize=(10,10))
sns.countplot(x='class',data=df)
```

Out[141… <AxesSubplot:xlabel='class', ylabel='count'>




**מסקנות**:

נראה כי הגורמים שהכי משפיעים על המכירה זה, אופן יצירת הקשר, ברירת מחדל, הלוואה וזמן השיחה, כלומר שככל שאורך זמן השיחה ארוך יותר, כך הסיכוי לסיווג גדל, לעומת זאת הסיכויים פחות משתנים משתנים כל כך בנוגע לדיור, השכלה, זוגיות, גיל, עבודה ורוב סוגי העבודה.



**naive bayes classifier , Equal frequency**

**Confusion Matrix (Train)**

| | |
|---|---|
| TP = 36649 | FP = 1806 |
| FN = 2161 | TN = 1564 |

**Confusion Matrix (Test)**

| | |
|---|---|
| TP = 1233 | FP = 234 |
| FN = 1184 | TN = 380 |

**Results**

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 90.60 % | 90.19 % | 90.60 % | 90.38 % |
| test | 53.22 % | 56.63 % | 53.22 % | 48.74 % |

Return for the First Page

Data Project

**naive bayes classifier , Based entropy**

**Confusion Matrix (Train)**

| | |
|---|---|
| TP = 36649 | FP = 1806 |
| FN = 2161 | TN = 1564 |

**Confusion Matrix (Test)**

| | |
|---|---|
| TP = 1233 | FP = 234 |
| FN = 1184 | TN = 380 |

**Results**

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 90.60 % | 90.19 % | 90.60 % | 90.38 % |
| test | 53.22 % | 56.63 % | 53.22 % | 48.74 % |

Return for the First Page

Data Project

## K-MEANS , Equal width

### Confusion Matrix (Train)

| | |
|---|---|
| TP = 15578 | FP = 22877 |
| FN = 2094 | TN = 1631 |

### Confusion Matrix (Test)

| | |
|---|---|
| TP = 1149 | FP = 318 |
| FN = 1260 | TN = 304 |

### Results

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 40.80 % | 80.95 % | 40.80 % | 51.63 % |
| test | 47.94 % | 48.30 % | 47.94 % | 43.05 % |

Return for the First Page

Data Project

## K-MEANS , Equal frequency

### Confusion Matrix (Train)

| | |
|---|---|
| TP = 15578 | FP = 22877 |
| FN = 2094 | TN = 1631 |

### Confusion Matrix (Test)

| | |
|---|---|
| TP = 318 | FP = 1149 |
| FN = 304 | TN = 1260 |

### Results

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 40.80 % | 80.95 % | 40.80 % | 51.63 % |
| test | 52.06 % | 51.73 % | 52.06 % | 47.46 % |

Return for the First Page

Data Project

## K-MEANS , Based entropy

### Confusion Matrix (Train)

| | |
|---|---|
| TP = 15578 | FP = 22877 |
| FN = 2094 | TN = 1631 |

### Confusion Matrix (Test)

| | |
|---|---|
| TP = 318 | FP = 1149 |
| FN = 304 | TN = 1260 |

### Results

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 40.80 % | 80.95 % | 40.80 % | 51.63 % |
| test | 52.06 % | 51.73 % | 52.06 % | 47.46 % |

Return for the First Page

Data Project

## KNN , Equal width

### Confusion Matrix (Train)

| | |
|---|---|
| TP = 38455 | FP = 0 |
| FN = 2686 | TN = 1039 |

### Confusion Matrix (Test)

| | |
|---|---|
| TP = 1457 | FP = 10 |
| FN = 1530 | TN = 34 |

### Results

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 93.63 % | 94.05 % | 93.63 % | 91.94 % |
| test | 49.19 % | 63.48 % | 49.19 % | 33.85 % |

Return for the First Page

**KNN , Equal frequency**

**Confusion Matrix (Train)**

| | |
|---|---|
| TP = 38455 | FP = 0 |
| FN = 2686 | TN = 1039 |

**Confusion Matrix (Test)**

| | |
|---|---|
| TP = 1457 | FP = 10 |
| FN = 1530 | TN = 34 |

**Results**

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 93.63 % | 94.05 % | 93.63 % | 91.94 % |
| test | 49.19 % | 63.48 % | 49.19 % | 33.85 % |

Return for the First Page

Data Project

**KNN , Based entropy**

**Confusion Matrix (Train)**

| | |
|---|---|
| TP = 38455 | FP = 0 |
| FN = 2686 | TN = 1039 |

**Confusion Matrix (Test)**

| | |
|---|---|
| TP = 1457 | FP = 10 |
| FN = 1530 | TN = 34 |

**Results**

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 93.63 % | 94.05 % | 93.63 % | 91.94 % |
| test | 49.19 % | 63.48 % | 49.19 % | 33.85 % |

Return for the First Page

Data Project

## ID3 , Equal width

### Confusion Matrix (Train)

| | |
|---|---|
| TP = 38455 | FP = 0 |
| FN = 0 | TN = 3725 |

### Confusion Matrix (Test)

| | |
|---|---|
| TP = 1160 | FP = 307 |
| FN = 1126 | TN = 438 |

### Results

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 100.00 % | 100.00 % | 100.00 % | 100.00 % |
| test | 52.72 % | 54.90 % | 52.72 % | 49.50 % |

Return for the First Page

## ID3 , Equal frequency

### Confusion Matrix (Train)

| | |
|---|---|
| TP = 38455 | FP = 0 |
| FN = 0 | TN = 3725 |

### Confusion Matrix (Test)

| | |
|---|---|
| TP = 1156 | FP = 311 |
| FN = 1104 | TN = 460 |

### Results

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 100.00 % | 100.00 % | 100.00 % | 100.00 % |
| test | 53.32 % | 55.54 % | 53.32 % | 50.35 % |

Return for the First Page

Data Project

## ID3 , Based entropy

### Confusion Matrix (Train)

| | |
|---|---|
| TP = 38455 | FP = 0 |
| FN = 0 | TN = 3725 |

### Confusion Matrix (Test)

| | |
|---|---|
| TP = 1125 | FP = 342 |
| FN = 1092 | TN = 472 |

### Results

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 100.00 % | 100.00 % | 100.00 % | 100.00 % |
| test | 52.69 % | 54.48 % | 52.69 % | 50.04 % |

Return for the First Page

**ID3 (our) , Equal width**

**Confusion Matrix (Train)**

| | |
|---|---|
| TP = 38318 | FP = 137 |
| FN = 2486 | TN = 1239 |

**Confusion Matrix (Test)**

| | |
|---|---|
| TP = 1105 | FP = 362 |
| FN = 1028 | TN = 536 |

**Results**

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 93.78 % | 93.57 % | 93.78 % | 92.44 % |
| test | 54.14 % | 55.87 % | 54.14 % | 52.18 % |

Return for the First Page

Data Project

## ID3 (our) , Equal frequency

### Confusion Matrix (Train)

| | |
|---|---|
| TP = 38368 | FP = 87 |
| FN = 1551 | TN = 2174 |

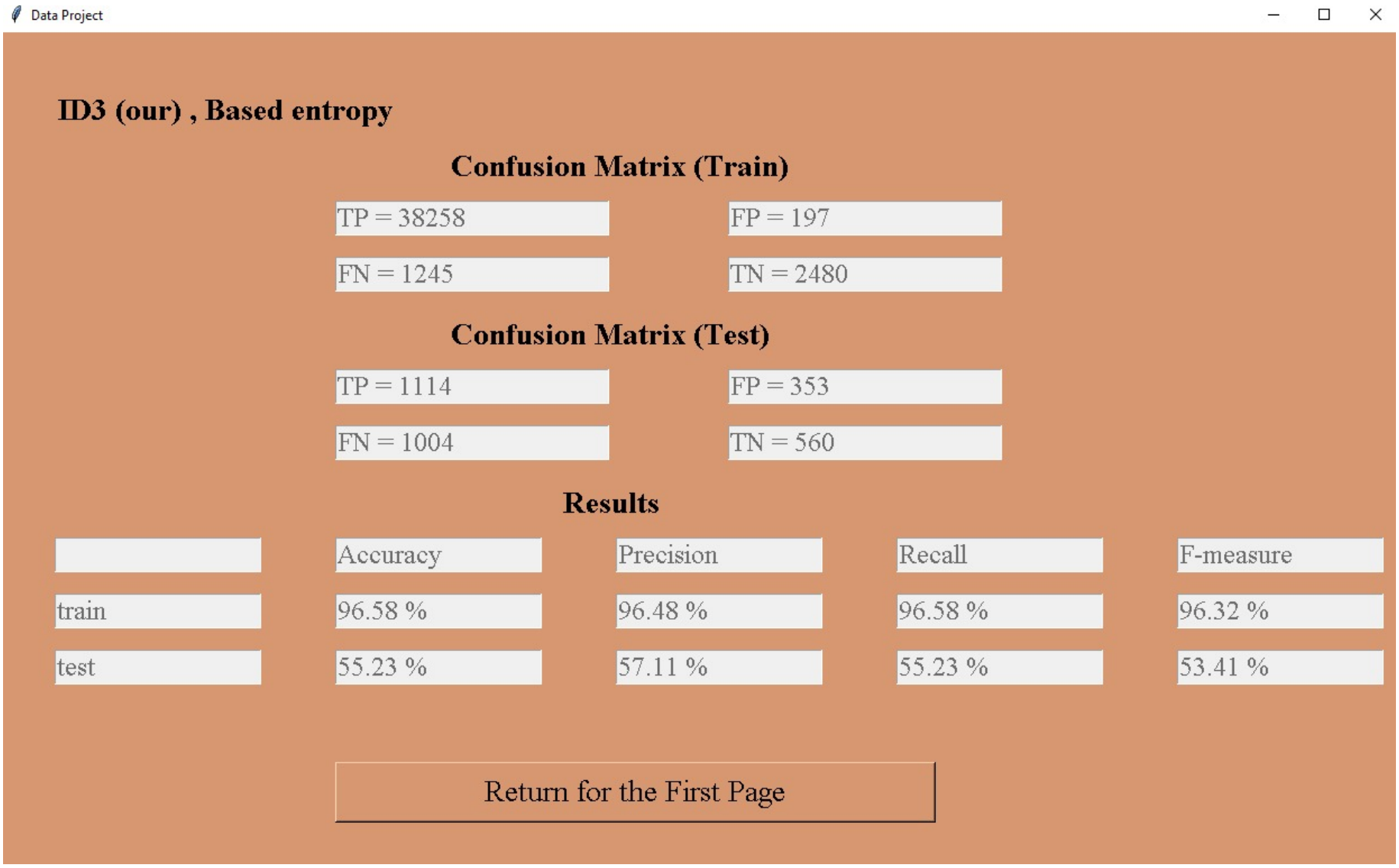
### Confusion Matrix (Test)

| | |
|---|---|
| TP = 1073 | FP = 394 |
| FN = 925 | TN = 639 |

### Results

| | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|
| train | 96.12 % | 96.12 % | 96.12 % | 95.68 % |
| test | 56.48 % | 57.91 % | 56.48 % | 55.37 % |

Return for the First Page




מסקנות:

יצא המודל הכי מדוייק naive bayes

האלגוריתמים ID3, K - MEANS, KNN המדדים כמעט זהים בשיטות הדיסקרטיזציה.

| אלגוריתם | שיטת דיסקריטיזציה | תוצאה % ( Accuracy) | תוצאה % (Precision) | תוצאה % (Recall) | תוצאה % (F-measure) |
|---|---|---|---|---|---|
| Naive Bayes | Equal width | 53.22 | 56.63 | 53.22 | 48.74 |
| Naive Bayes | Equal Frequency | 53.22 | 56.63 | 53.22 | 48.74 |
| Naive Bayes | *Based Entropy* | 53.22 | *56.63* | *53.22* | *48.74* |
| *K-means* | Equal width | *47.94* | *48.30* | *47.94* | *43.05* |
| *K-means* | Equal Frequency | *52.06* | *51.73* | *52.06* | *47.46* |
| *K-means* | *Based Entropy* | *52.06* | *51.73* | *52.06* | *47.46* |
| *KNN* | Equal width | *49.19* | *63.48* | *49.19* | *33.85* |
| *KNN* | Equal Frequency | *49.19* | *63.48* | *49.19* | *33.85* |
| *KNN* | Based Entropy | *49.19* | *63.48* | *49.19* | *33.85* |
| *ID3 ספריות* | Equal width | 52.72 | *54.90* | *52.72* | *49.50* |
| *ID3 ספריות* | Equal Frequency | 53.32 | *55.54* | *53.32* | *50.35* |
| *ID3 ספריות* | *Based Entropy* | *52.69* | *54.48* | *52.69* | *50.04* |
| *ID3 our* | Equal width | *54.14* | *55.87* | *54.14* | *52.18* |
| *ID3 our* | Equal Frequency | *56.48* | *57.91* | *56.48* | *55.37* |
| *ID3 our* | *Based Entropy* | *55.23* | *57.11* | *55.23* | *53.41* |