README.md

Churn Prediction

The Data

Our principal data set come from a start up that offers memberships to users access online meditation, breathwork and coaching sessions with live professors.

We have a data set of classes taken by user, their plan, their plan type, when and if they have cancelled that plan/membership.

Attibutes/Features Description

- Class Category: class/session type (Breathwork; Meditation & Breathwork; Meditation, Breathwork & Movement; Body Awereness; Body Intelligence; Mindfulness Design);
- Instructor: who teaches the sessions;
- School Instructor: which instance inside the platform the instructor came from;
- Date: date of the session;
- Time: time of each session;
- Duration: duration of the session;
- Language: language which the session is given;
- Name: Name of the user/member/student;
- Phone Number: of the user (empty);
- Status: shows if the user took the class (NO_SHOW; PRESENT; CANCELED);
- Plan Type: Free == 1; Freemium (used for trial) == 2; Paid == 3;
- Currency: BRL, USD, EUR, GBP;
- Cancelled Date: dd/mm/yyyy if the user has cancelled the membership, NA if they are still active;
- Churn: 0 (no); 1 (yes).

Goal / Target

Try to predict churn by using as a basis the user behavior of taking classes.

Tools

- Excel,
- Python,
- Jupiter Notebook,
- VSCode,
- Numpy, Pandas,

- Matplotlib, SeaBorn
- and Scikit Learn.

Disclosure

All sensitive data has been replaced to not compromise user and company information.

Step 1: Preparing the Data

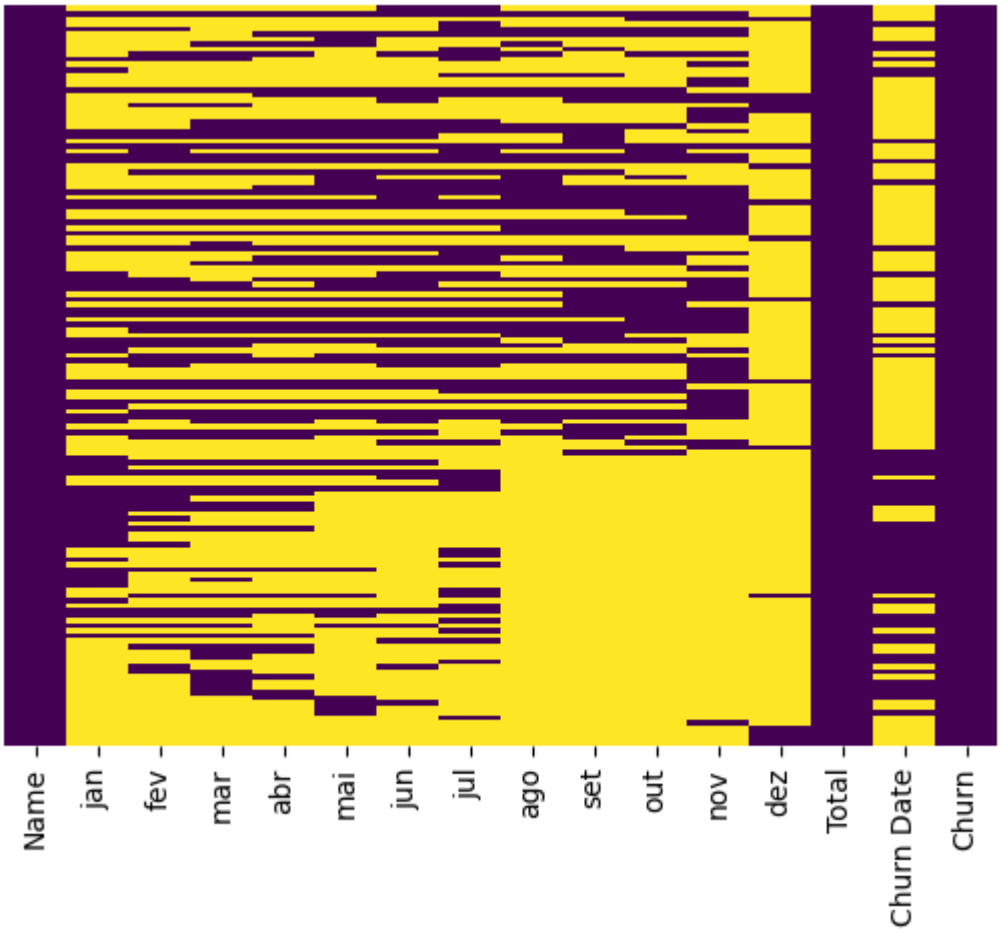
Load two data sets.The first one wth more complete information described above. And a second one just with the attendance of the users easily show by month.

```
df = pd.read_excel('Report_Class_Booking_Presence.xlsx', sheet_name='ClassBook')
df.head()
```

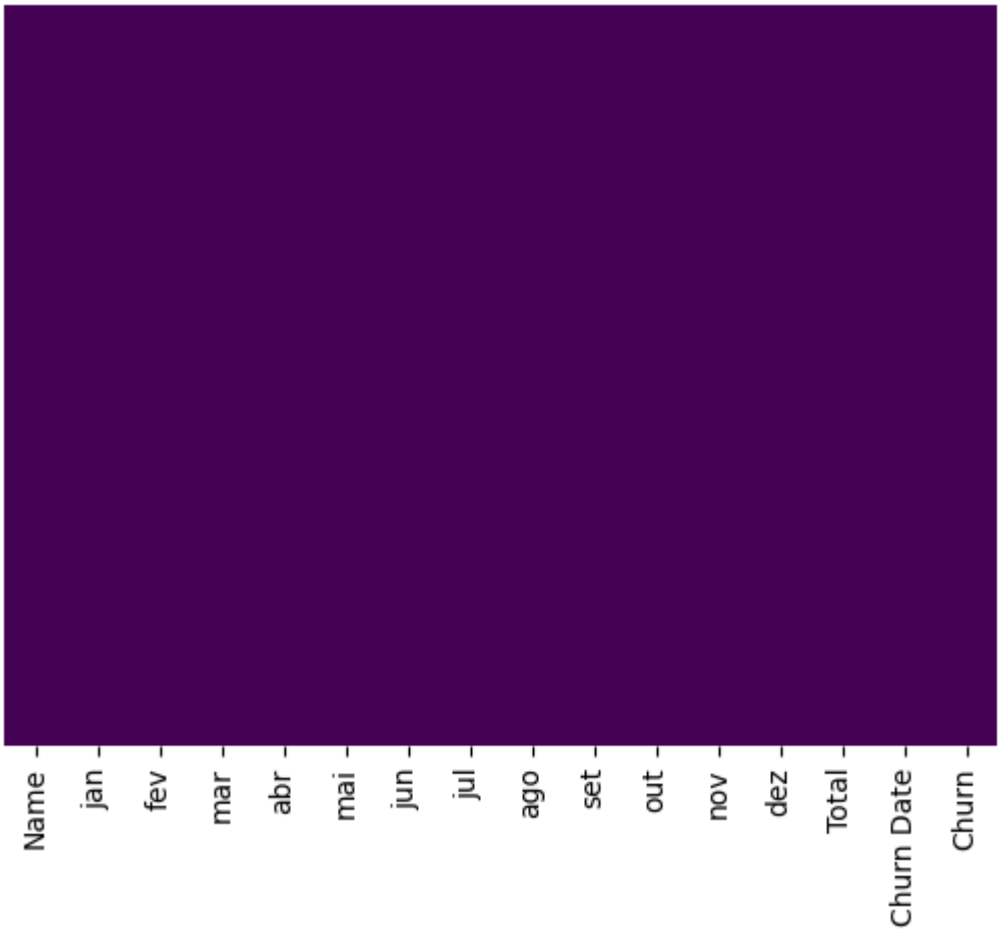
```
df_new = pd.read_excel('Report_Class_Booking_Presence.xlsx',
sheet_name='Attendance Paid Users')
df_new.head()
```

Some of the data was 'null' values. It was actual zeros in the real world. So we filled it.

```
sns.heatmap(df_new.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```




```
df_new = df_new.fillna(0)

sns.heatmap(df_new.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

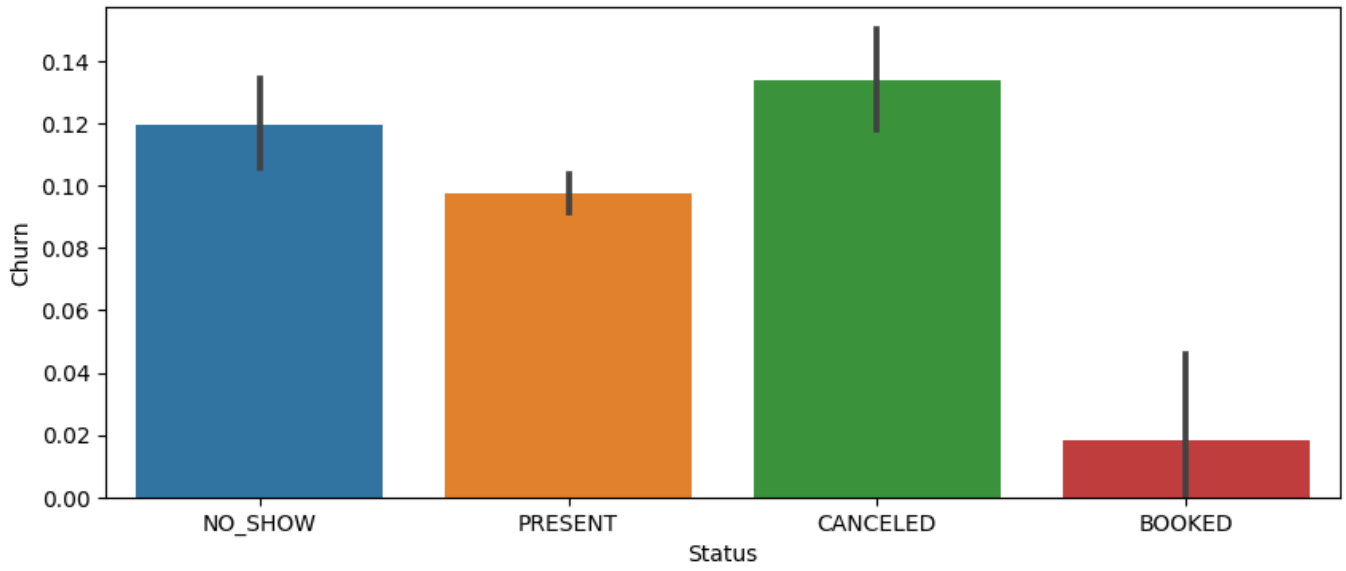


We had some outliers with 3x more monthly attendance than the average. So we removed them.

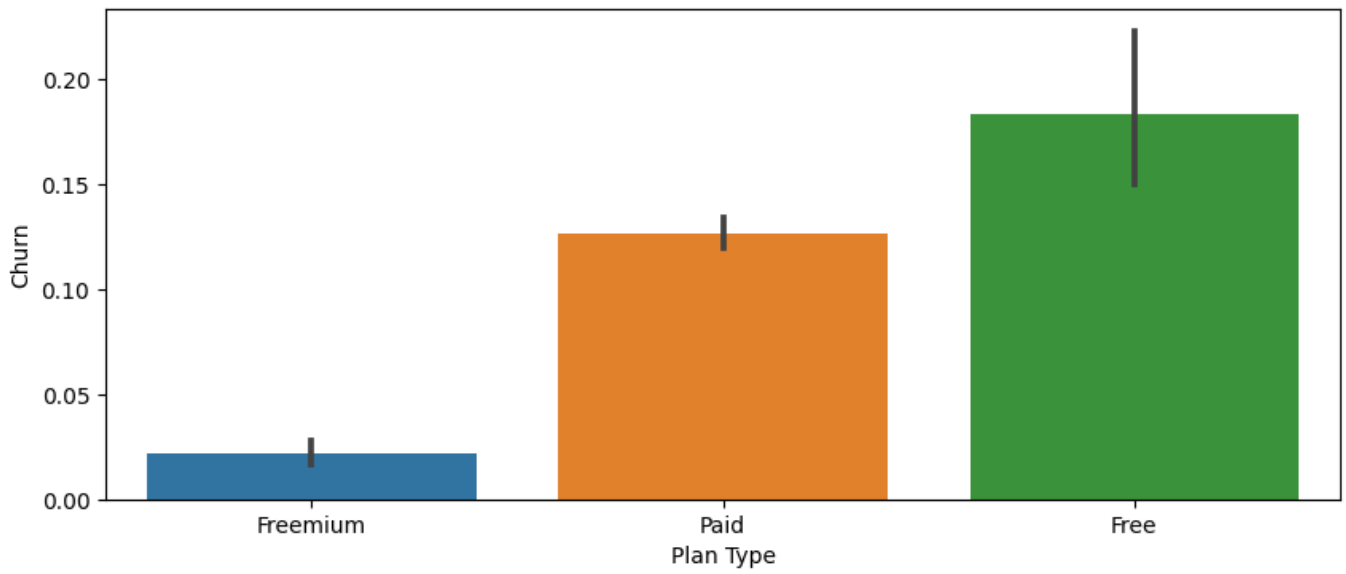
```
df_new = df_new[df_new.Total < 200]
```

Step 2: EDA - Exploratory Data Analysis

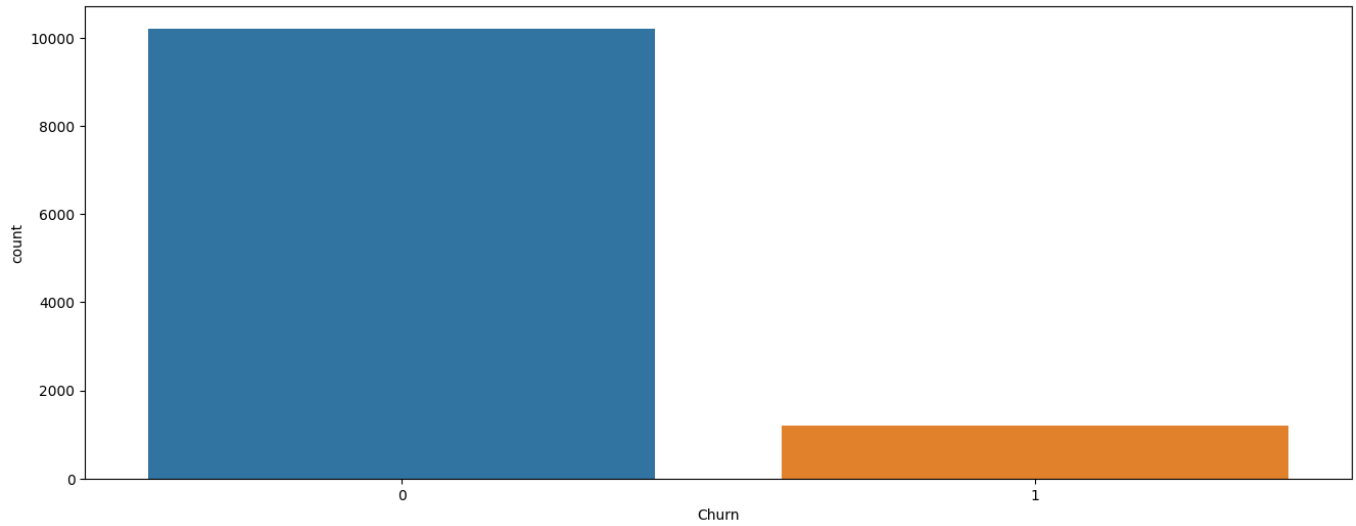
How 'Status' is correlated with 'Churn'. We can see that people who do the sessions has lower churn:



How 'Plan Type' is correlated with 'Churn'. We can see that 'Freemium' and 'Paid' users has the same churn:



Counting evasions(churn) of active users. We can see that is a unbalenced data set::

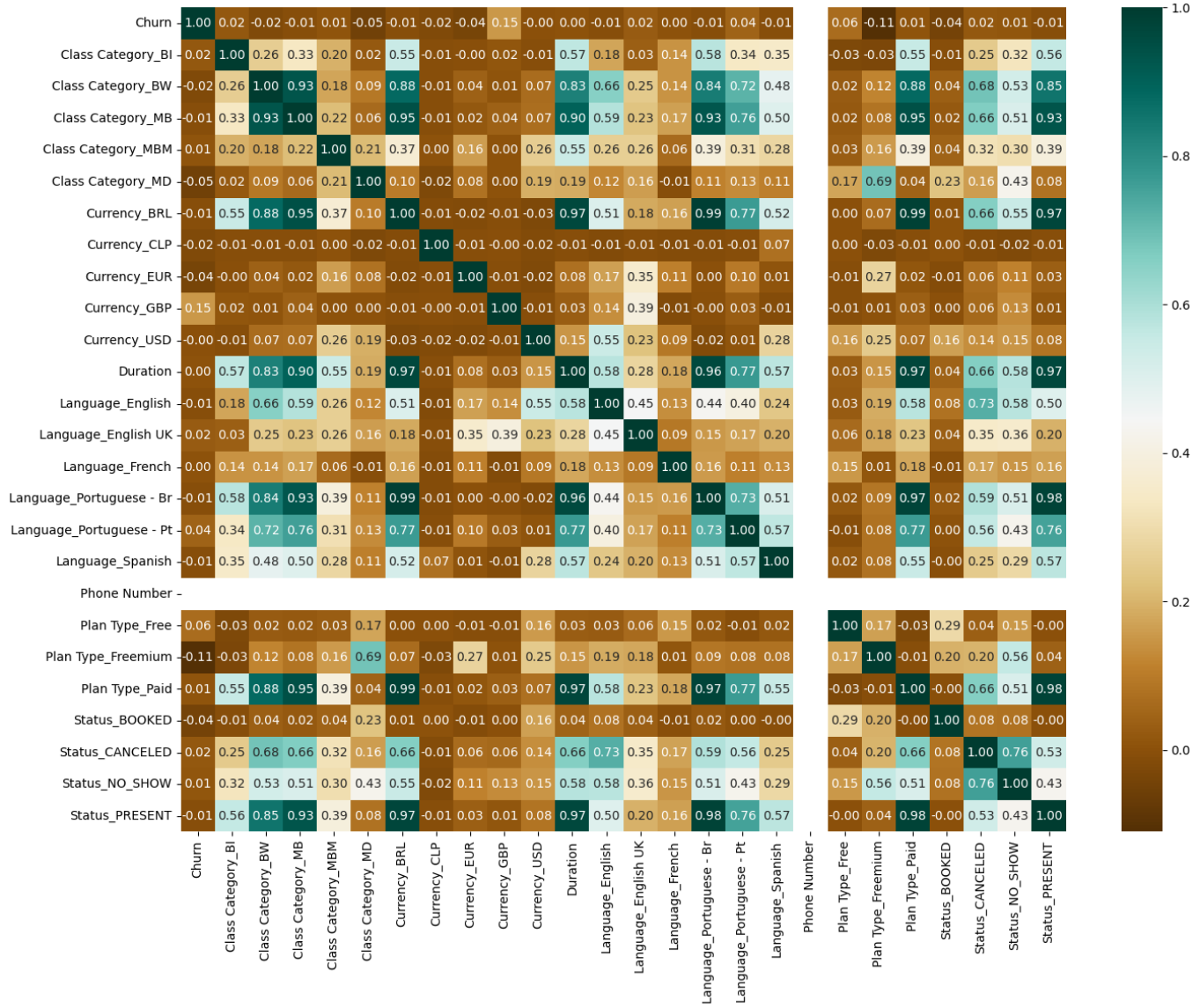


We have transform categorical variables to numerical variables and also pivot table by user to continue some analysis.

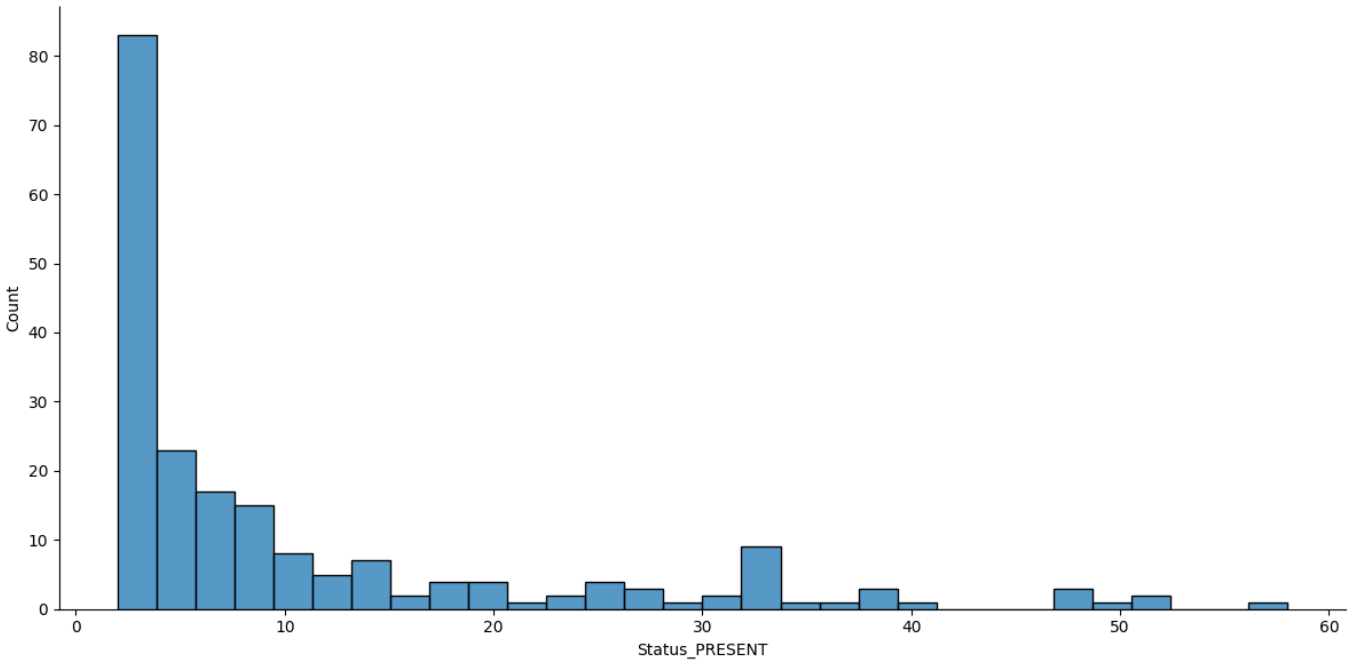
And how a user could be two times evasion we did a new 'Churn' column:

```
df = pd.get_dummies(df, columns=['Currency', 'Language', 'Plan Type', 'Class
Category', 'Status'])

df_pivoted = pd.pivot_table(df, index=['Name'], aggfunc=np.sum)

df_pivoted['Churn'] = np.where(df_pivoted['Churn'] != 0, 1, 0)
```

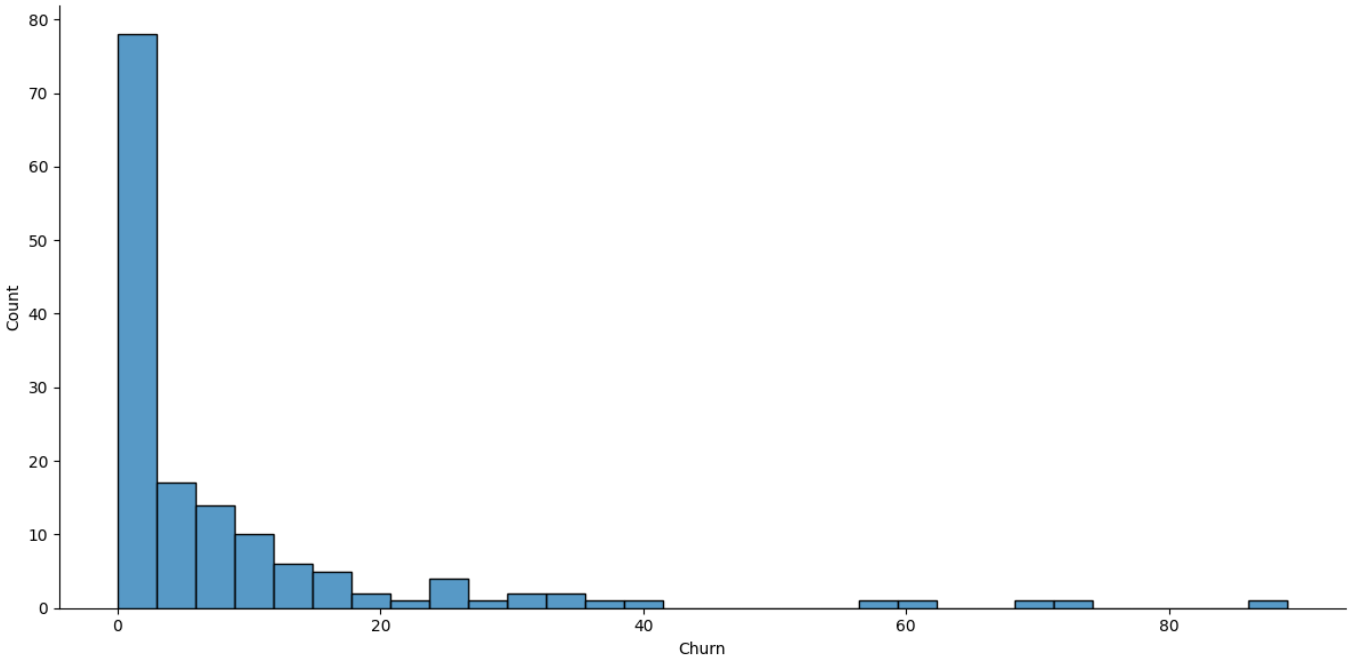
Checking correlations:



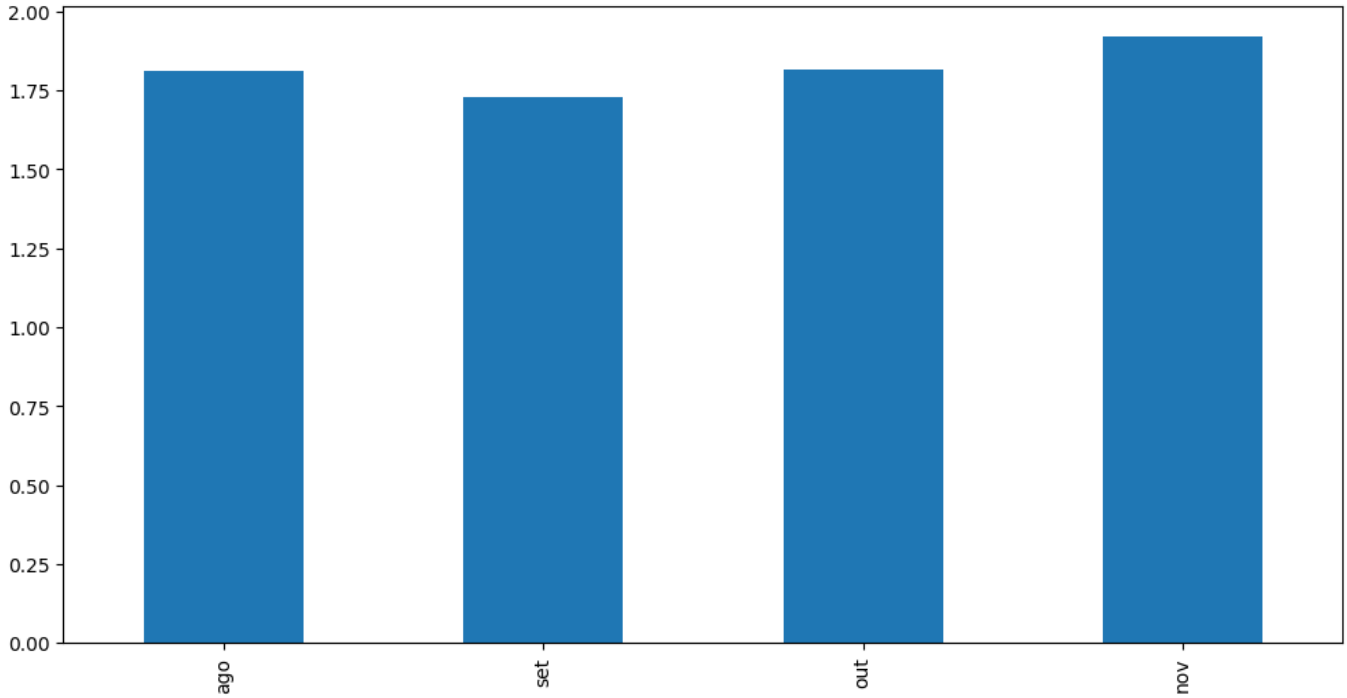
Counting Status_PRESENT without outliers (how many sessions each user took):



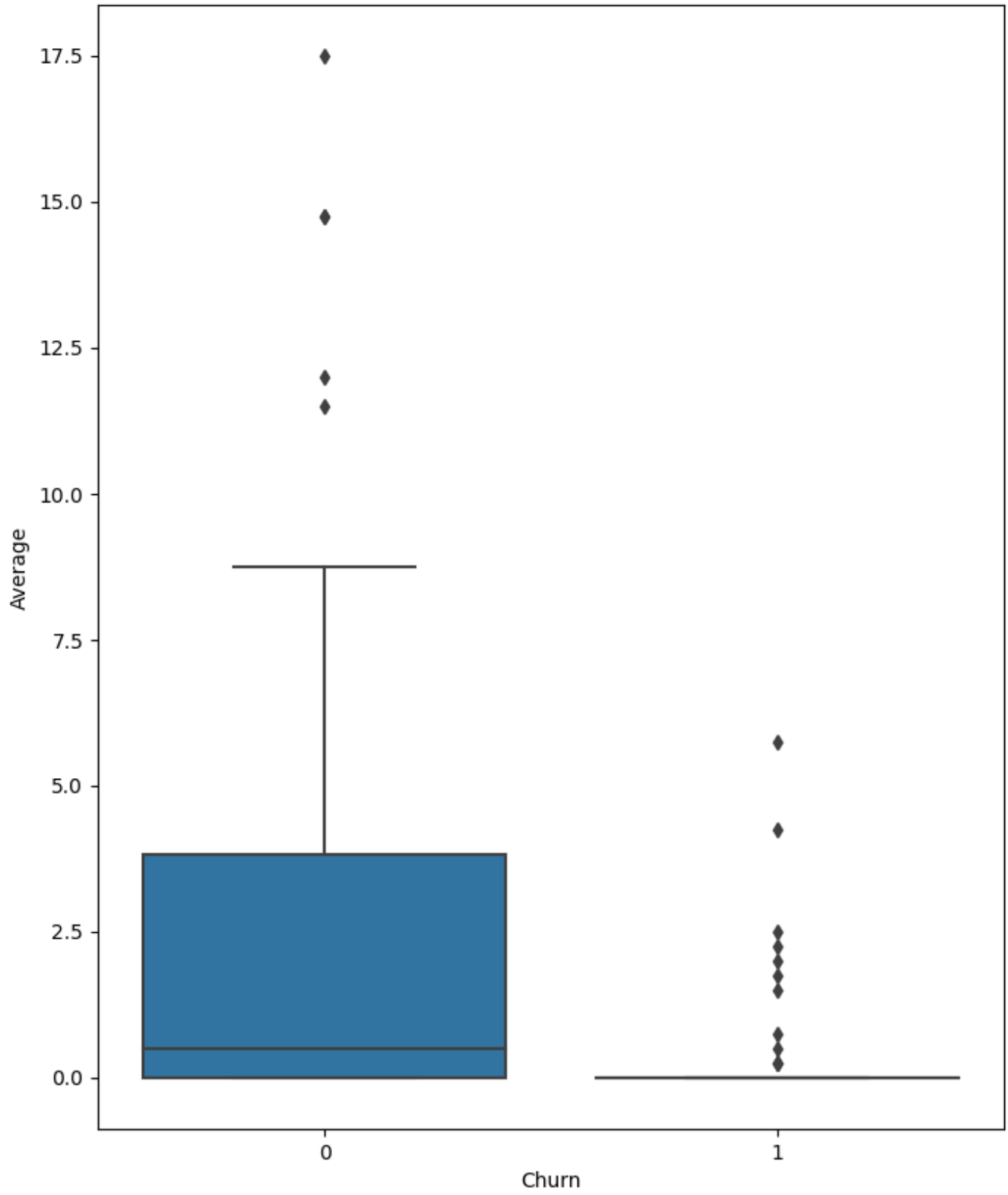
Churn by instructor:



Avg. monthly attendance (Paid Users):



Avg. attendance by churn (Paid Users):



EDA Conclusion

Since we want to predict churn, observing the heatmap we cannot see clearly correlations between any actual features on the general data set.

With the second data frame we can clearly see that user who have canceled their membership has took less sessions (avg. of 0.41) and who 'stay' did more sessiones (avg. of 2.70) each month.

We are asumming that if users do at least 4 sessions by month they engenders loyalty.

We should collect data about others things (posible new quality features) to improve future analysis.

Ex.:

- Who didn't even booked any session?
- Did the user took the assessment?
- Made the personalized follow-up?
- How many times did the user enter the platform?
- Overdue Payment?
- With who the user did the sessions?
- Quality of the transmission.

Step 3: Prediction

Let's try to do some predictions using student attendance as the main attribute.

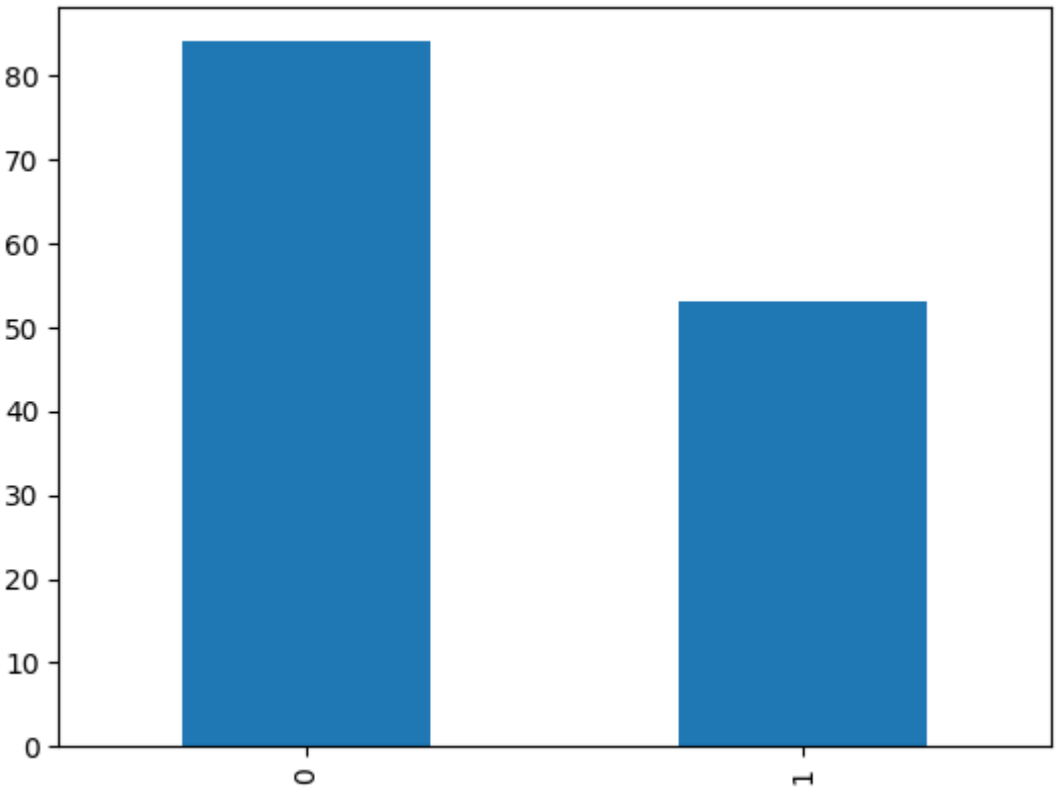
We are going to test the predictions using the first data set (with all attendances) and then the second data set (with just the paid users attendance).

In thrid place, as we also have an unbalanced data set, we are going to ajust using OverSampling of the minority.

Train test split:

```
from sklearn.model_selection import train_test_split

X = pred.drop('Churn',axis=1)
y = pred['Churn']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=101, stratify=y)
```

Unbalanced data:



% of each classification:

Churn % 0 61.31% 1 38.68%

Decision Tree

Training data using DECISION TREE:

```
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()

dtree.fit(X_train,y_train)
```

Predicting:

```
predictions = dtree.predict(X_test)
```

Evaluating the model:

```
from sklearn.metrics import classification_report,confusion_matrix

print(confusion_matrix(y_test,predictions))

print(classification_report(y_test,predictions))
```

```
[[17  9]
 [ 2 14]]


              precision    recall  f1-score   support

           0       0.89      0.65      0.76        26
           1       0.61      0.88      0.72        16

    accuracy                           0.74        42
   macro avg       0.75      0.76      0.74        42
weighted avg       0.79      0.74      0.74        42
```

Random Forest

Training data using RANDOM FOREST:

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100)

rf.fit(X_train,y_train)
```

Predicting:

```
predictions2 = rf.predict(X_test)
```

Evaluating the model:

```
print(classification_report(y_test,predictions2))

print(confusion_matrix(y_test,predictions2))
```

```
[[18  8]
 [ 3 13]]

              precision    recall  f1-score   support

           0       0.86      0.69      0.77        26
           1       0.62      0.81      0.70        16

    accuracy                           0.74        42
   macro avg       0.74      0.75      0.73        42
weighted avg       0.77      0.74      0.74        42
```

Doing it again using the fisrt data set (more complete):

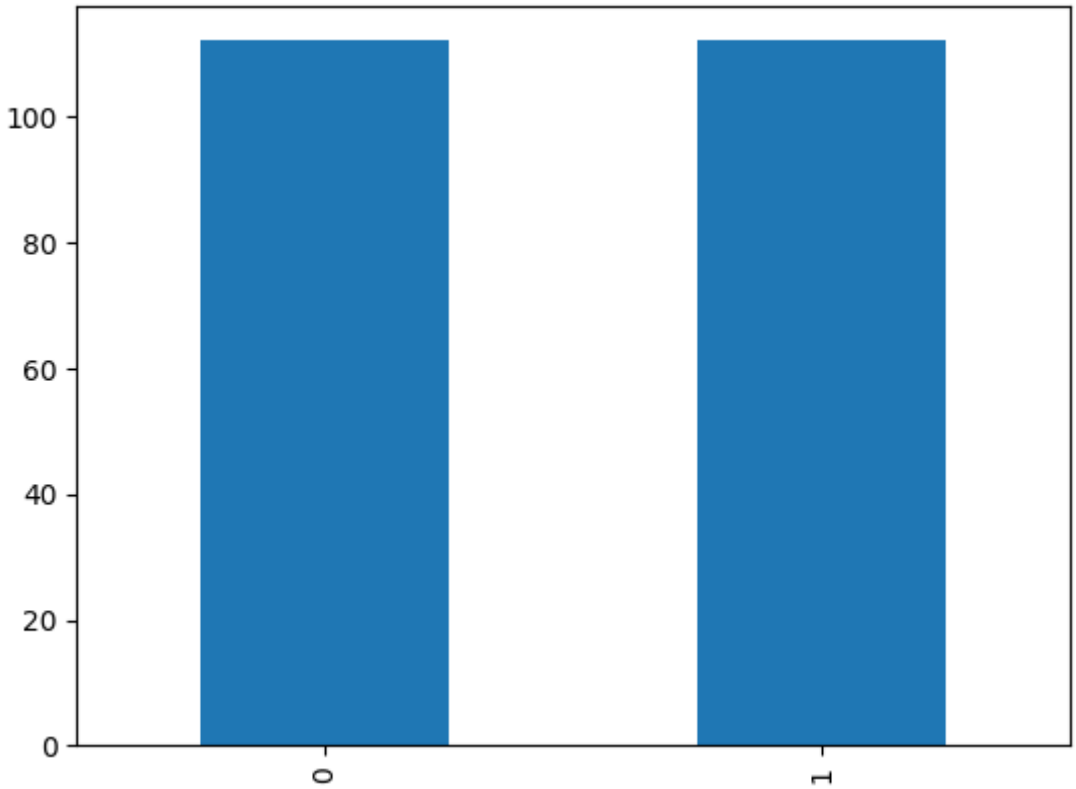
Random Forest:

```
[[46  2]
 [12  5]]

              precision    recall  f1-score   support

           0       0.79      0.96      0.87        48
           1       0.71      0.29      0.42        17

    accuracy                           0.78        65
   macro avg       0.75      0.63      0.64        65
weighted avg       0.77      0.78      0.75        65
```

Oversampling unbalanced data:

Balancing minority data:

```
from imblearn.over_sampling import RandomOverSampler

ros = RandomOverSampler(random_state=42)

X_res, y_res = ros.fit_resample(X_train, y_train)

y_res.value_counts().plot.bar()
```



Classifying again:

```
rf = RandomForestClassifier(n_estimators=100)

rf.fit(X_train,y_train)

predictions4 = rf.predict(X_test)
```

```
print(confusion_matrix(y_test,predictions4))

print(classification_report(y_test,predictions4))
```

```
[[46  2]
 [12  5]]
```

```
              precision    recall  f1-score   support

           0       0.79      0.96      0.87        48
           1       0.71      0.29      0.42        17

    accuracy                           0.78        65
   macro avg       0.75      0.63      0.64        65
weighted avg       0.77      0.78      0.75        65
```

There was no improvement in the model balancing the data.

ML Conclusion

Seems that our model using random forest can predict relatively well who will not be evasion but not so well who will be.

Others kind of data should be get from the platform to improve the model as well observed also in the EDA above.

Even when the data set is balanced, the results still bring low recall.

We have a little data set with less than 200 paid users.

Posible next steps:

- 1. Change the algorithm that could work better with little data sets like Naive Bayes;
- 2. Continue with the same algorithm and improve the parameters to obtain better results;
- 3. Improve training/test data;
- 4. Change the business question.