BAN 210: Workshop 1

Import data

In [1]:
```
!pip install pyreadstat
```

```
Requirement already satisfied: pyreadstat in c:\users\admin\anaconda3\lib\site-packages (1.2.7)
Requirement already satisfied: pandas>=1.2.0 in c:\users\admin\anaconda3\lib\site-packages (from pyreadstat) (2.2.2)
Requirement already satisfied: numpy>=1.26.0 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (1.2
6.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadst
at) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (2024.
1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=1.2.0->pyreadstat) (202
3.3)
Requirement already satisfied: six>=1.5 in c:\users\admin\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas>=1.2.0
->pyreadstat) (1.16.0)
```

In [5]:
```
import pyreadstat as pyds
df1, meta = pyds.read_sas7bdat('../BAN 210/retention.sas7bdat')
df1
```

Out[5]:

	Att_hrs_fall	GENDER	HIGH_SCHOOL_PERCENTILE	AGE	Att_hrs_spr	Avg_income	Distance	Dropped_course	Major_rate	SAT
0	13.0	F	97.0	18.910335	15.0	30573.0	5.083094	0	0.821782	1160.0
1	14.0	M	97.0	18.036961	15.0	27305.0	2.198764	0	0.795455	1050.0
2	15.0	M	83.0	18.162902	15.0	30573.0	5.083094	0	0.843750	1140.0
3	14.0	M	92.0	18.590007	12.0	35865.0	3.245574	0	NaN	1090.0
4	16.0	F	80.0	18.880219	16.0	40125.0	19.486585	0	0.840391	1150.0
...	...	...	...	...	...	...	...	...	...	...
2621	13.0	M	42.0	18.151951	12.0	48581.0	179.444678	0	0.838384	1080.0
2622	13.0	M	74.0	18.650240	18.0	NaN	110.993592	0	0.831511	1110.0
2623	16.0	F	0.6	18.102669	12.0	73510.0	356.674286	0	0.828571	1150.0
2624	12.0	F	71.0	18.318960	13.0	NaN	NaN	0	0.829268	1060.0
2625	16.0	F	95.0	18.666667	16.0	59599.0	91.022950	0	0.829515	1140.0

2626 rows × 21 columns

Q1:

How many observations (rows) exist in the Retention table? How many variables (columns) exist in the Retention table? How are the initial roles and levels of the variables determined? Can we change the default settings? Explain by example

In [13]:
```
df1.shape
```

Out[13]: (2626, 21)

In [18]:
```
print("Number of observations (rows):", df1.shape[0])
print("Number of variables (columns):", df1.shape[1])
```

```
Number of observations (rows): 2626
Number of variables (columns): 21
```

Python does not give us the initial roles and levels of the variables determined. We have to tag each variable if it is predictor or a target variable. Besides, there is no default setting for python

Q2:

Based on the default settings are variables with missing values rejected? Explain. Based on the default settings, which numeric variables are assigned as nominal variables? Is there a maximum number of levels for character variables in the default settings? Explain.

In [74]:
```
# based on the default settings are variables with missing values rejected? Explain
# this code will display how many missing values are present per column
print(df1.isnull().sum())
```

```
Att_hrs_fall               0
GENDER                     0
HIGH_SCHOOL_PERCENTILE   263
AGE                        0
Att_hrs_spr                0
Avg_income               157
Distance                  98
Dropped_course             0
Major_rate                79
SAT                        0
Extra_curr                 0
Legacynum                  0
Stu_worker_ind             0
Need_pct_met               0
Perc_hrs_comp_fall         0
Hs_rate                  805
Dorm_rate                  0
Instate                    0
Transcrip                  0
Fall_GPA                   0
Target                     0
dtype: int64
```

Therefore the code above demonstrates that based on the default settings, variables with missing values are not rejected. The code above proves that most columns have 0 missing values but high_school_percentile, avg_income, distance, major_rate, and Hs_rate all have missing values. Hs_rate has the highest amount, with 805 values missing. Therefore, if a variable has missing values it is not rejected.

In Python, if there are any variables having missing values, the model will drop those missing values itself automatically. However, in SAS, those will be rejected.

In [166…

```
# is there a maximum number of levels of character variables in the default setting? Explain
# Check data types - to determine which variables are character variables (object)
df1.info()

# this code looks only at the variables that are 'object' and then
# counts the amount of unique levels (categorical variables have a few unique values)
unique_levels = df1.select_dtypes(include='object').agg('nunique')
print(unique_levels)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2626 entries, 0 to 2625
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Att_hrs_fall            2626 non-null   float64
 1   GENDER                  2626 non-null   object
 2   HIGH_SCHOOL_PERCENTILE  2363 non-null   float64
 3   AGE                     2626 non-null   float64
 4   Att_hrs_spr             2626 non-null   float64
 5   Avg_income              2469 non-null   float64
 6   Distance                2528 non-null   float64
 7   Dropped_course          2626 non-null   object
 8   Major_rate              2547 non-null   float64
 9   SAT                     2626 non-null   float64
 10  Extra_curr              2626 non-null   float64
 11  Legacynum               2626 non-null   object
 12  Stu_worker_ind          2626 non-null   object
 13  Need_pct_met            2626 non-null   float64
 14  Perc_hrs_comp_fall      2626 non-null   float64
 15  Hs_rate                 1821 non-null   float64
 16  Dorm_rate               2626 non-null   float64
 17  Instate                 2626 non-null   object
 18  Transcrip               2626 non-null   object
 19  Fall_GPA                2626 non-null   float64
 20  Target                  2626 non-null   int32
dtypes: float64(14), int32(1), object(6)
memory usage: 420.7+ KB
GENDER            2
Dropped_course    6
Legacynum         2
Stu_worker_ind    2
Instate           2
Transcrip         2
dtype: int64
```

The code above analyzes the amount of levels in the 7 character variables - shown as 'object'. Most of the character variables have two levels but the Dropped_course variable has 6 levels. Therefore this proves that based on the default setting there is no maximum number of levels for character variables.

The more categories we have in table, the more complex our model will be. Thus, there is no maximum number of levels for character variables. We can create as many categories as we want in python.

Q3:

How many input variables in the DataFrame? How many target variables? How many of the input variables are nominal? Are there any rejected variables? (Compare with your initial answer on the number of variables in this table.) What is a typical value for a sample size? Create a BAR plot of the GENDER variable. What percentage of the students are female? Highlight the students with Target=1 on the above GENDER bar plot. Copy and then paste the plot here. Create a HISTOGRAM plot of the Avg_income variable with 20 bins. Paste here. Take a sample of the last 10 rows of the Retention dataset Take a screenshot (PtrScr on your keyboard) and paste here.

In [30]:
```
# How many input variables in the DataFrame?

input_variables = df1.drop(columns=['Target']) #not include Target variable
num_inp_var = input_variables.shape[1]
print("Number of input variables", num_inp_var)
```

```
Number of input variables 20
```

In [38]:
```
df1.head()
```

Out[38]:

	Att_hrs_fall	GENDER	HIGH_SCHOOL_PERCENTILE	AGE	Att_hrs_spr	Avg_income	Distance	Dropped_course	Major_rate	SAT	...
0	13.0	F	97.0	18.910335	15.0	30573.0	5.083094	0	0.821782	1160.0	...
1	14.0	M	97.0	18.036961	15.0	27305.0	2.198764	0	0.795455	1050.0	...
2	15.0	M	83.0	18.162902	15.0	30573.0	5.083094	0	0.843750	1140.0	...
3	14.0	M	92.0	18.590007	12.0	35865.0	3.245574	0	NaN	1090.0	...
4	16.0	F	80.0	18.880219	16.0	40125.0	19.486585	0	0.840391	1150.0	...

5 rows × 21 columns

In [40]:
```
# How many target variables?
target_variables = df1['Target']

print("Number of target variables", 1)
```

```
Number of target variables 1
```

In [70]:
```
#Number of nominal variables
nominal_variables = input_variables.select_dtypes(include='object').shape[1] #not include Target variable
print("Number of nominal variables", nominal_variables)
```

```
Number of nominal variables 6
```

In [98]:
```
# Are there any rejected variables? including missing values and values having 1 unique value
# Columns with all values missing
all_missing = df1.columns[df1.isnull().all()]

# Columns with only one unique value (constant)
constant_cols = [col for col in df1.columns if df1[col].nunique() <= 1]

# Combine
rejected_vars = list(set(all_missing).union(constant_cols))

# Output
print("Number of rejected variables:", len(rejected_vars))
print("Rejected variable names:", rejected_vars)
```

```
Number of rejected variables: 0
Rejected variable names: []
```

There are no rejected variables, which is similar to what has mentioned on Q2 before.

In [ ]:
```
# What is a typical value for a sample size?
```

Small: < 100 observations — often too small unless it's a very controlled experiment.

Medium: 100–500 observations — usable, but may limit model complexity.

Good: 500–2000 observations — enough for solid statistical analysis and machine learning models.

Large: > 2000 observations — great for complex models and generalization

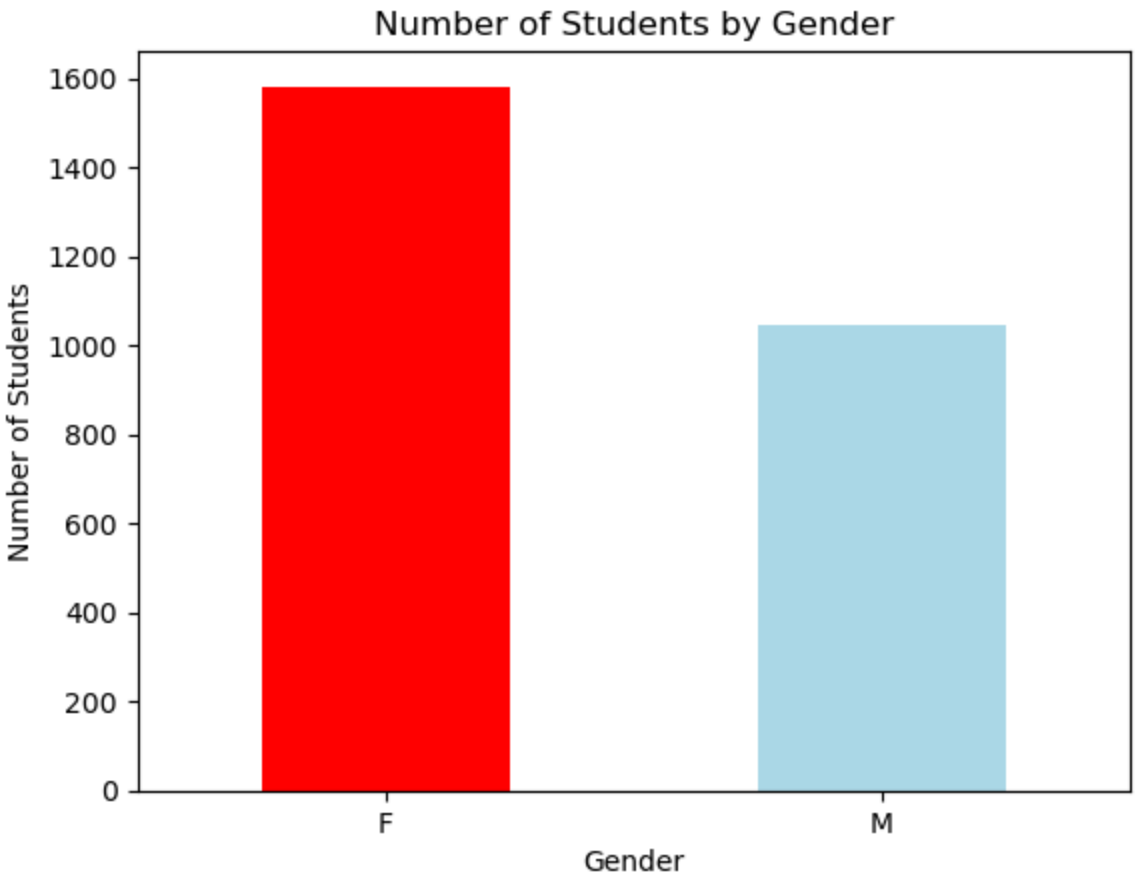
In [106…
```
# Create a BAR plot of the GENDER variable. What percentage of the students are female?
import matplotlib.pyplot as plt
import seaborn as sns

# Clean up Gender and Target variables
df1['GENDER'] = df1['GENDER'].astype(str)
df1['Target'] = df1['Target'].astype(int)

# Count how many students in each gender
gender_counts = df1['GENDER'].value_counts()
gender_counts
```

Out[106…
```
GENDER
F    1582
M    1044
Name: count, dtype: int64
```

In [120…
```python
# Calculate the percentage of students who are female
total_students = gender_counts.sum()
female_count = gender_counts.get('F',0)
female_percentage = (female_count / total_students) * 100
print ('the percentage of female students:', female_percentage, '%')
```

```
the percentage of female students: 60.24371667936025 %
```

In [130…
```python
# Create a bar plot of Gender counts
plt.figure
gender_counts.plot(kind='bar', color = ['red','lightblue'])
plt.title('Number of Students by Gender')
plt.xlabel ('Gender')
plt.ylabel ('Number of Students')
plt.xticks(rotation=0)
plt.show()
```



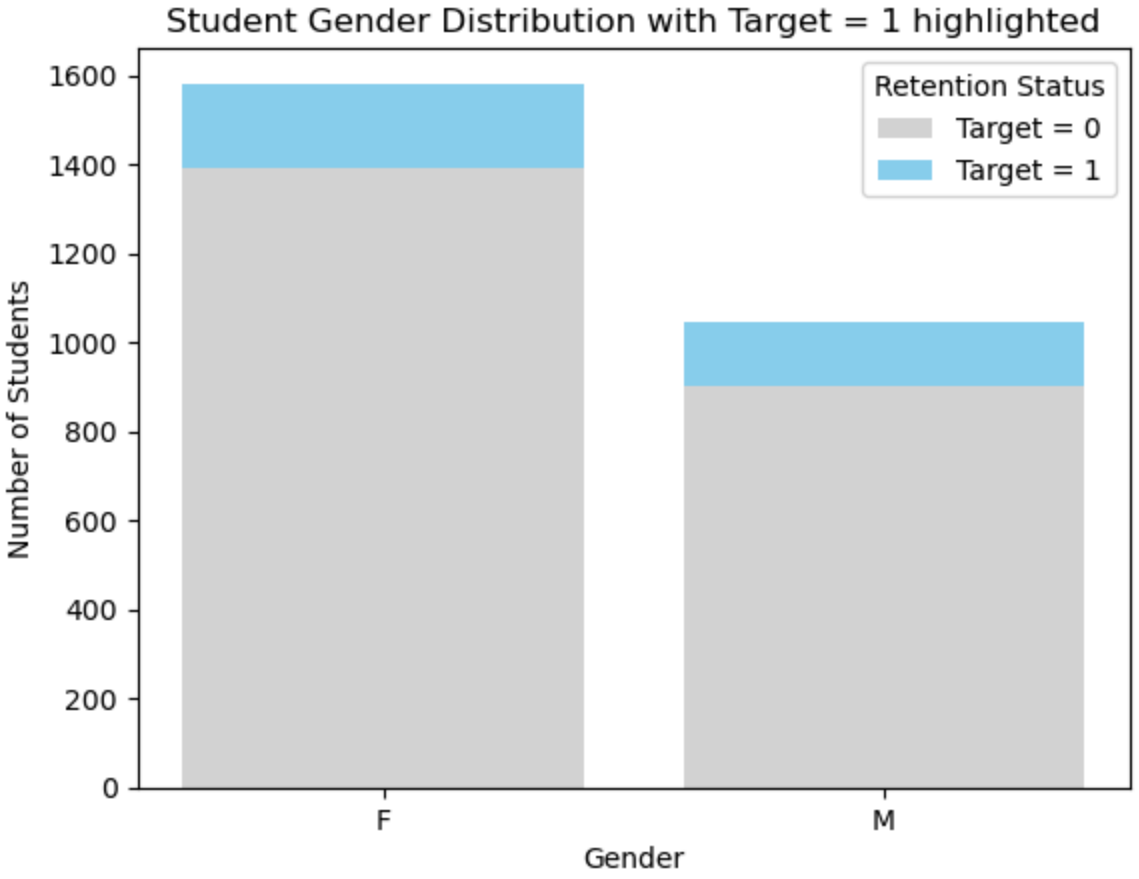
In [134…
```
#Highlight the students with Target=1 on the above GENDER bar plot
#Count only students with Target = 1 (retained) by Gender
gender_retained = df1[df1['Target'] == 1] ['GENDER'].value_counts()
gender_retained
```

Out[134…
```
GENDER
F    192
M    142
Name: count, dtype: int64
```

In [144…
```
genders = list(gender_counts.index)
```

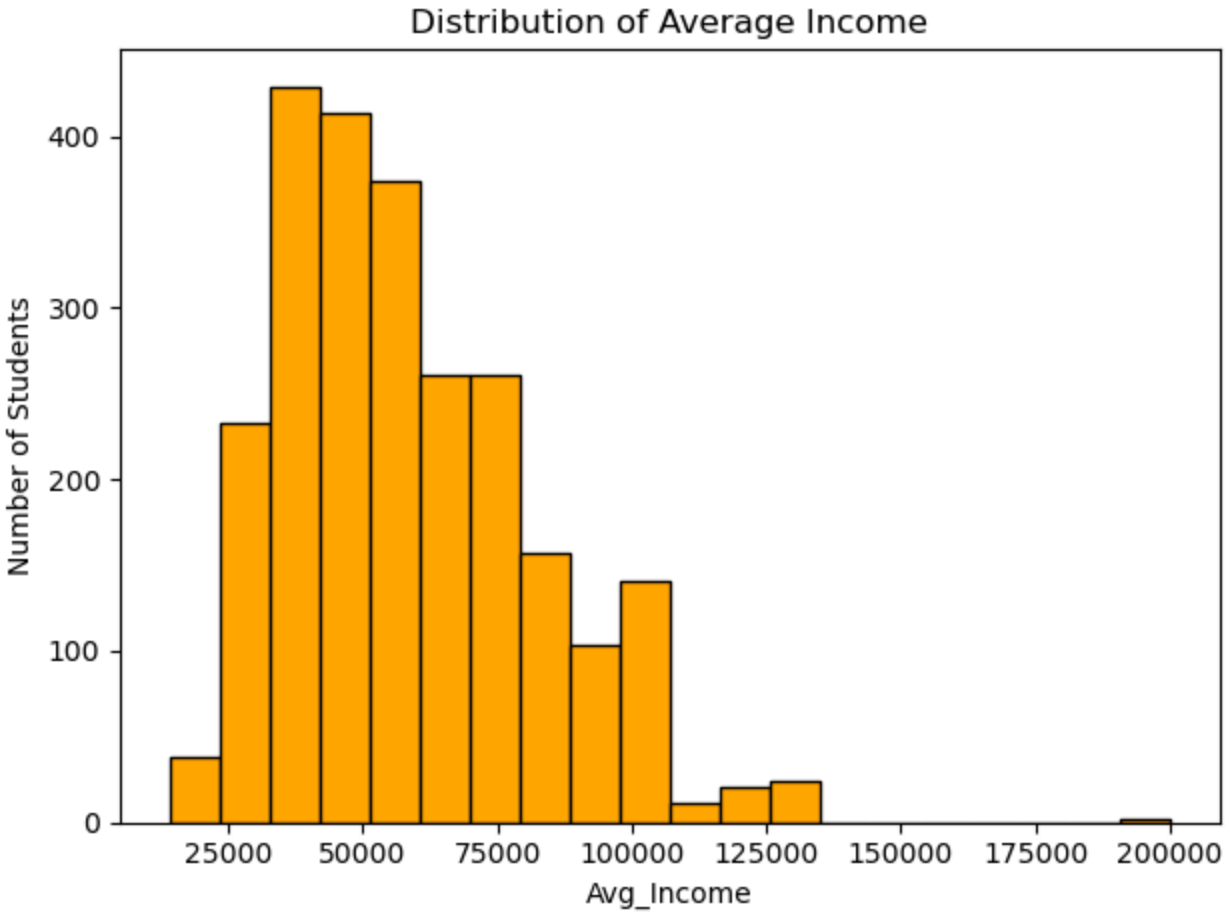
In [158…
```
#Plot the stacked bar
plt.figure
# base bar
plt.bar(genders, gender_counts - gender_retained, label = "Target = 0", color = 'lightGray')
# top bar
plt.bar(genders, gender_retained, bottom = gender_counts - gender_retained, label = "Target = 1", color = 'skyblue')
```

```
#Chart lables
plt.title('Student Gender Distribution with Target = 1 highlighted')
plt.xlabel('Gender')
plt.ylabel('Number of Students')
plt.xticks(rotation = 0)
plt.legend(title = 'Retention Status')
plt.show()
```



In [162…

```
#Create a HISTOGRAM plot of the Avg_income variable with 20 bins
plt.figure
plt.hist(df1['Avg_income'].dropna(), bins = 20, color = 'orange', edgecolor='black')
plt.title('Distribution of Average Income')
plt.xlabel('Avg_Income')
plt.ylabel('Number of Students')
plt.tight_layout()
plt.show()
```



In [164…

```
#Take a sample of the last 10 rows of the Retention dataset
last_10rows = df1.tail(10)
last_10rows
```

Out[164...

	Att_hrs_fall	GENDER	HIGH_SCHOOL_PERCENTILE	AGE	Att_hrs_spr	Avg_income	Distance	Dropped_course	Major_rate	SA
2616	15.0	M	59.0	18.606434	16.0	36760.0	156.742238	0	0.840391	1290.
2617	12.0	M	13.0	18.880219	18.0	26725.0	167.681727	0	0.795455	1060.
2618	13.0	F	51.0	18.488706	13.0	53190.0	1186.851867	0	0.831585	980.
2619	14.0	M	62.0	18.280630	16.0	60898.0	1154.751207	0	0.831511	1070.
2620	14.0	M	0.6	18.967830	13.0	NaN	NaN	0	0.829268	1140.
2621	13.0	M	42.0	18.151951	12.0	48581.0	179.444678	0	0.838384	1080.
2622	13.0	M	74.0	18.650240	18.0	NaN	110.993592	0	0.831511	1110.
2623	16.0	F	0.6	18.102669	12.0	73510.0	356.674286	0	0.828571	1150.
2624	12.0	F	71.0	18.318960	13.0	NaN	NaN	0	0.829268	1060.
2625	16.0	F	95.0	18.666667	16.0	59599.0	91.022950	0	0.829515	1140.

10 rows × 21 columns