MACHINE LEARNING ASSIGNMENT-2

- Pandas basic commands:-
  - Create DataFrame:

```
In [1]: import pandas as pd
        df = pd.DataFrame({'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':[86,97,96,72,83]});
        print(df)
```

```
    X   Y   Z
0  78  84  86
1  85  94  97
2  96  89  96
3  80  83  72
4  86  86  83
```

  - Create DataSeries:

```
In [2]: import pandas as pd
        s = pd.Series([2, 4, 6, 8, 10])
        print(s)
```

```
0     2
1     4
2     6
3     8
4    10
dtype: int64
```

Q1. DataFrame Basic Exercise:

Our DataFrame `(df)` contains data on registered voters in the United States, including demographic information and political preference. Using pandas, print the first 5 rows of the DataFrame to get a sense of what the data looks like. Next, answer the following questions:

```
In [13]: import pandas as pd
         import numpy as np
         data = {

             'Name': ['John', 'Jane', 'Bob', 'Alice', 'Charlie'],
             'Age': [25, 30, 22, 28, 35],
             'Gender': ['Male', 'Female', 'Male', 'Female', 'Male'],
             'Political_Preference': ['Democrat', 'Republican', 'Independent', 'Democrat', 'Republican'],
             'Days_Watch_TV_News': [10, 5, 20, 18, np.nan],
             'educ' : ['High School', 'Bachelor', 'Master', 'PhD', 'Some College'],
             'PID' : ['Strong Democrat', 'Weak Democrat', 'Strong Republican', 'Undecided', 'Weak Republican'],     'ClinLR' : [6, 5, 3, 1
             'income' : [45000, 52000, 40000, 30000, 15000],
             'vote': [0.8, 0.7, 0.1, 0.5, 0.2],
             'Population': [100000, 570000, 260000, 190000, 300000],
             'selfLR': [6, 3, 7, 8, 1],
             'DoleLR': [2, 4, 3, 3, 5]
         }
         df = pd.DataFrame(data)
         print("First 5 rows of the DataFrame:\n", df.head())
         observations = df.shape[0]
         print("\n1. Number of observations in the DataFrame:", observations)
         variables = df.shape[1]
         print("\n2. Number of variables measured (number of columns):", variables)
```

```
First 5 rows of the DataFrame:
      Name  Age  Gender Political_Preference  Days_Watch_TV_News  \
0     John   25    Male             Democrat                10.0
1     Jane   30  Female           Republican                 5.0
2      Bob   22    Male          Independent                20.0
3    Alice   28  Female             Democrat                18.0
4  Charlie   35    Male           Republican                 NaN

           educ               PID  ClinLR  income  vote  Population  selfLR  \
0   High School    Strong Democrat       6   45000   0.8      100000       6
1      Bachelor      Weak Democrat       5   52000   0.7      570000       3
2        Master  Strong Republican       3   40000   0.1      260000       7
3           PhD         Undecided       1   30000   0.5      190000       8
4  Some College    Weak Republican       4   15000   0.2      300000       1

   DoleLR
0       2
1       4
2       3
3       3
4       5

1. Number of observations in the DataFrame: 5

2. Number of variables measured (number of columns): 13

3. Age of the youngest person: 22
   Age of the oldest person: 35
```

```
4. Average days a week respondents watch TV news (rounded to the nearest tenth): 13.2

5. Number of missing values in the DataFrame: 1
```

Q2. Cleaning Data Exercise:

We want to adjust the dataset for our use. Do the following:

- **Rename the `educ` column `education`.**
-

```
df.columns = ['Name','Age','Gender', 'Political_Preference', 'Days_Watch_TV_News', 'education', 'PID', 'ClinLR', 'income', 'vote'
print("After renaming the column :\n", df.head())
```

```
After renaming the column :
      Name   Age   Gender       Political_Preference     Days_Watch_TV_News  \
0     John   25    Male             Democrat                10.0
1     Jane   30    Female           Republican               5.0
2     Bob    22    Male           Independent              20.0
3     Alice  28    Female             Democrat               18.0

4   Charlie   35    Male           Republican                NaN

      education               PID  ClinLR  income  vote  Population  selfLR  \
0   High School    Strong Democrat       6   45000   0.8      100000       6
1      Bachelor      Weak Democrat       5   52000   0.7      570000       3
2        Master  Strong Republican       3   40000   0.1      260000       7
3           PhD         Undecided       1   30000   0.5      190000       8

4  Some College    Weak Republican       4   15000   0.2      300000       1

   DoleLR
0       2
1       4
2       3
3       3
4       5
```

- **Create a new column called `party` based on each respondent's answer to `PID`. `party` should equal `Democrat` if the respondent selected either Strong Democrat or Weak Democrat. party will equal `Republican` if they selected Strong or Weak Republican and `Independent` if they selected anything else.**

In [20]:
```
party_mapping = {
    'Strong Democrat': 'Democrat',
    'Weak Democrat': 'Democrat',
    'Strong Republican': 'Republican',
    'Weak Republican': 'Republican'
}
df['party'] = df['PID'].map(party_mapping).fillna('Independent')
print("First 5 rows of the DataFrame after adding 'party' column:\n", df.head())
```

```
First 5 rows of the DataFrame after adding 'party' column:
      Name  Age  Gender Political_Preference  Days_Watch_TV_News  \
0     John   25    Male             Democrat                10.0
1     Jane   30  Female           Republican                 5.0
2      Bob   22    Male          Independent                20.0
3    Alice   28  Female             Democrat                18.0
4  Charlie   35    Male           Republican                 NaN

           educ                PID  ClinLR  income  vote  Population  selfLR  \
0   High School    Strong Democrat       6   45000   0.8      100000       6
1      Bachelor      Weak Democrat       5   52000   0.7      570000       3
2        Master  Strong Republican       3   40000   0.1      260000       7
3           PhD          Undecided       1   30000   0.5      190000       8
4  Some College    Weak Republican       4   15000   0.2      300000       1

   DoleLR        party
0       2     Democrat
1       4     Democrat
2       3   Republican
3       3  Independent
4       5   Republican
```

- **Create a new column called `age_group` that buckets respondents into the following categories based on their `age`: 4, 25-34, 35-44, 45-54, 55-64, and 65 and over.**

In [21]:
```
bins = [18, 25, 35, 45, 55, 65, float('inf')]
labels = ['18-24', '25-34', '35-44', '45-54', '55-64', '65 and over']
df['age_group'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)
print("First 5 rows of the DataFrame after adding 'age_group' column:\n", df.head())
```

```
First 5 rows of the DataFrame after adding 'age_group' column:
      Name  Age  Gender Political_Preference  Days_Watch_TV_News  \
0     John   25    Male             Democrat                10.0
1     Jane   30  Female           Republican                 5.0
2      Bob   22    Male          Independent                20.0
3    Alice   28  Female             Democrat                18.0
4  Charlie   35    Male           Republican                 NaN

           educ                PID  ClinLR  income  vote  Population  selfLR  \
0   High School    Strong Democrat       6   45000   0.8      100000       6
1      Bachelor      Weak Democrat       5   52000   0.7      570000       3
2        Master  Strong Republican       3   40000   0.1      260000       7
3           PhD          Undecided       1   30000   0.5      190000       8
4  Some College    Weak Republican       4   15000   0.2      300000       1

   DoleLR        party age_group
0       2     Democrat     25-34
1       4     Democrat     25-34
2       3   Republican     18-24
3       3  Independent     25-34
4       5   Republican     35-44
```

3. Filtering Data Exercise

Use the filtering method to find all the respondents who have the impression that Bill Clinton is moderate or conservative (`ClinLR` equals 4 or higher). How many respondents are in this subset?

In [23]:
```
filtered_df = df[df['ClinLR'] >= 4]
number_of_respondents = filtered_df.shape[0]
print("Number of respondents who have the impression that Bill Clinton is moderate or conservative:", number_of_respondents)
```

```
Number of respondents who have the impression that Bill Clinton is moderate or conservative: 3
```

Among these respondents, how many have a household income less than $50,000 and attended at least some college?

```
filtered_subset = df[(df['ClinLR'] >= 4) & (df['income'] < 50000) & (df['education'] == 'Some College')]
number_of_respondents_subset = filtered_subset.shape[0]
print("Number of respondents who have the impression that Bill Clinton is moderate or conservative, have a household income less
```

```
Number of respondents who have the impression that Bill Clinton is moderate or conservative, have a household income less than $50,0
```

4. Calculating From Data Exercise

For each of the below match-ups, choose the group that is more likely to vote for Bill Clinton. You can calculate this using the percentage of each group that intends to vote for Clinton (`vote`). Which match-up was the closest? Which had the biggest difference?

- **Democrats or Republicans**

In [26]:
```
democrats_percentage = df[df['party'] == 'Democrat']['vote'].mean()
republicans_percentage = df[df['party'] == 'Republican']['vote'].mean()
closest_matchup = min(democrats_percentage, republicans_percentage)
biggest_difference = abs(democrats_percentage - republicans_percentage)
print('closest_matchup: ', closest_matchup)
print('biggest_difference: ', biggest_difference)
```

```
closest_matchup:  0.15000000000000002
biggest_difference:  0.6
```

- **People younger than 44 or People 44 and older**

In [27]:
```
younger_than_44_percentage = df[df['Age'] < 44]['vote'].mean()
older_than_44_percentage = df[df['Age'] >= 44]['vote'].mean()
closest_matchup = min(younger_than_44_percentage, older_than_44_percentage)
biggest_difference = abs(younger_than_44_percentage - older_than_44_percentage)
print('closest_matchup: ', closest_matchup)
print('biggest_difference: ', biggest_difference)
```

```
closest_matchup:  0.4600000000000001
biggest_difference:  nan
```

- **People who watch TV news at least 6 days a week or People who watch TV news less than 3 days a week**

In [28]:
```
frequent_news_watchers_percentage = df[df['Days_Watch_TV_News'] >= 6]['vote'].mean()
infrequent_news_watchers_percentage = df[df['Days_Watch_TV_News'] < 3]['vote'].mean()
closest_matchup = min(frequent_news_watchers_percentage, infrequent_news_watchers_percentage)
biggest_difference = abs(frequent_news_watchers_percentage - infrequent_news_watchers_percentage)
print('closest_matchup: ', closest_matchup)
print('biggest_difference: ', biggest_difference)
```

```
closest_matchup:  0.4666666666666666
biggest_difference:  nan
```

- **People who live somewhere with a population greater than the average respondent or People who live in a place with a population equal to or less than the average respondent**

In [29]:
```
average_population = df['Population'].mean()
high_population_percentage = df[df['Population'] > average_population]['vote'].mean()
low_population_percentage = df[df['Population'] <= average_population]['vote'].mean()
closest_matchup = min(high_population_percentage, low_population_percentage)
biggest_difference = abs(high_population_percentage - low_population_percentage)
print('closest_matchup: ', closest_matchup)
print('biggest_difference: ', biggest_difference)
```

```
closest_matchup:  0.44999999999999996
biggest_difference:  0.016666666666666663
```

5. Grouping Data Exercise

Use the `groupby()` method to bucket respondents by `age_group`. Which age group is the most conservative? Which watches TV news the least?

In [31]:
```
grouped_by_age = df.groupby('age_group')
average_vote_by_age = grouped_by_age['vote'].mean()
most_conservative_age_group = average_vote_by_age.idxmin()
print("Mean vote for each age group:\n", average_vote_by_age)
print("\nThe most conservative age group is:", most_conservative_age_group)
average_news_days_by_age = grouped_by_age['Days_Watch_TV_News'].mean()
least_watch_news_age_group = average_news_days_by_age.idxmin()
print("\nMean number of days respondents in each age group watch TV news:\n", average_news_days_by_age)
print("\nThe age group that watches TV news the least is:", least_watch_news_age_group)
```

```
Mean vote for each age group:
 age_group
18-24          0.100000
25-34          0.666667
35-44          0.200000
45-54               NaN
55-64               NaN
65 and over         NaN
Name: vote, dtype: float64

The most conservative age group is: 18-24

Mean number of days respondents in each age group watch TV news:
 age_group
18-24          20.0
25-34          11.0
35-44           NaN
45-54           NaN
55-64           NaN
65 and over     NaN
Name: Days_Watch_TV_News, dtype: float64

The age group that watches TV news the least is: 25-34
```

Next, calculate 5 percentile groups based on income. Group the dataset by these percentiles. Which income bracket is the most liberal? Which is the most conservative? The oldest? Highest educated?

```
df['income_percentile'] = pd.qcut(df['income'], q=[0, 0.2, 0.4, 0.6, 0.8, 1.0], labels=['20%', '40%', '60%', '80%', '100%'])
grouped_by_income = df.groupby('income_percentile')
average_vote_by_income = grouped_by_income['vote'].mean()
most_liberal_income_bracket = average_vote_by_income.idxmax()
most_conservative_income_bracket = average_vote_by_income.idxmin()
oldest_income_bracket = df.groupby('income_percentile')['Age'].mean().idxmax()
highest_educated_income_bracket = df.groupby('income_percentile') ['education'].apply(lambda x: x.mode().iloc[0])
print("Mean vote for each income percentile:\n", average_vote_by_income)
print("\nThe most liberal income bracket is:", most_liberal_income_bracket)
print("The most conservative income bracket is:", most_conservative_income_bracket)
print("\nThe oldest income bracket is:", oldest_income_bracket)
print("\nThe highest educated income bracket is:", highest_educated_income_bracket)
```

```
Mean vote for each income percentile:
income_percentile
20%     0.2
40%     0.5
60%     0.1
80%     0.8
100%    0.7
Name: vote, dtype: float64

The most liberal income bracket is: 80%
The most conservative income bracket is: 60%

The oldest income bracket is: 20%

The highest educated income bracket is: income_percentile
20%     Some College
40%              PhD
60%           Master
80%      High School
100%        Bachelor
Name: education, dtype: object
```

6. Voting Across the Aisle

We are interested in learning more about respondents whose political views differ strongly from the candidate they expect to vote for. Using `selfLR`, `vote`, `ClinLR`, and `DoleLR`, work through the following questions. Your interpretation may differ from the answer key.

- **What is the largest recorded difference between a respondent's political leaning and their impression of their intended candidate's political leaning?**
- **How many respondents exhibit a difference of that magnitude?**
- **Make a separate DataFrame called `sway` that only includes these voters who exhibit a difference greater than |3|.**
- **Among those in `sway`, are respondents more likely to be voting for a candidate more conservative or more liberal than their own political leaning?**
- **In `sway`, which candidate is the more popular choice?**

```
df['leanins difference'] = abs(df['self18'] - df['Clin18']) largest difference =
df['leanins difference'].max() num respondents largest difference = df[df['leanins difference']
== largest difference].shape[0] sway = df[df['leanins difference'] > 3]
sway['popular candidate'] = np.where(sway['Clin18'] > sway['Dole18'], 'Arthur', 'Soca') sway['voting tendency'] = np.where(sway['Dole18']
> sway['self18'], 'Conservative', 'Liberal') print("Largest recorded difference between a respondent's political leaning and their
intended candidate's political leaning:", largest d print('\n') print("Number of respondents exhibiting this magnitude of difference:",
num respondents largest difference) print('\n')
print("Updated 'sway' DataFrame with 'popular candidate' column:\n", sway[['Name', 'self18', 'Clin18', 'Dole18', 'leanins difference', 'p
```

```
    Name  Age  Gender Political Preference  Days Watch TV News education  \
2    Bob   22    Male           Independent                20.0    Master
3   Alice   28  Female             Democrat                18.0       PhD

               PID  Clin18  Income  vote  Population  self18  Dole18  \
2  Strong Republican       3   40000   0.1      260000       7       3   3         Undecided       1
   30000   0.5      190000       8       3

        party age group income percentile  leanins difference  2
Republican     18-24              60%                   4
3  Independent     25-34              40%                   7
Largest recorded difference between a respondent's political leaning and their intended candidate's political leaning: 7


Number of respondents exhibiting this magnitude of difference: 1


Updated 'sway' DataFrame with 'popular candidate' column:
     Name  self18  Clin18  Dole18  leanins difference popular candidate  \
2    Bob       7       3       3                   4
Soca   3  Alice       8       1       3                   7
Soca

  voting tendency 2
Liberal
3    Liberal  <ipython-input-28-5e8afbb2cbe9>:13:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
sway['popular candidate'] = np.where(sway['Clin18'] > sway['Dole18'], 'Arthur', 'Soca') <ipython-input-28-5e8afbb2cbe9>:14:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
sway['voting tendency'] = np.where(sway['Dole18'] > sway['self18'], 'Conservative', 'Liberal')
```