Data Analysis Homework 1: Pandas and Numpy

Objective: The aim of this assignment is to demonstrate your proficiency in using Jupyter Notebook, IPython, and particularly the Pandas library for data analysis.

Requirements

- Create a new Jupyter Notebook. Import all necessary libraries. Write a brief summary of your findings. Add comments and Markdown cells in your Jupyter Notebook to explain your code and results.

Submission Guidelines

- Upload the notebook files in canvas, a pdf formart with clear result shown will be easier for TA to grade.
- Ensure that your code is clean, well-commented, and easily understandable.

```
import numpy as np
import pandas as pd
```

Q1. (70 points)

1. Use Pandas to load both data/AIS/transit_segments.csv, and data/AIS/vessel_information.csv. Show the first 5 rows of each dataset to inspect it. (10points)

2. For data/AIS/vessel_information.csv, keep only those rows with the type value occurring for at least 99 times in the dataset. (10points)

3. Merge data/AIS/vessel_information.csv and data/AIS/transit_segments.csv on the "mmsi" column using outer join. (10points)

4. If you are *not* allowed to call the inner join provided by Pandas but have the above outer join results, how to get the results of inner join? You can use other functions provided by Pandas (but not a function that directly implements the inner join). (10points)

5. Now directly call the inner join provided by Pandas, check whether your results above are exactly the same, give your analysis. (10points)

6. Save merged dataset as AIS_merge.csv and check the missing values (skip the time features). Replace missing values with column mode. (10points)

7. Use z-scores to detect outliers. Discuss how you would deal with them if there is any, you don't need to actually implement. (10points)

```
#1
#Loading the datasets
```

```
vessel_info=pd.read_csv('vessel_information.csv')
transit_segments=pd.read_csv('transit_segments.csv')

#Displaying the first 5 rows from each DataFrame
print("First 5 rows from vessel_information.csv: ")
print(vessel_info.head())

print("\nFirst 5 rows from transit_segments.csv: ")
print(transit_segments.head())

#2
#Count the occurrences of each 'type'
type_counts = vessel_info['type'].value_counts()
types_to_keep = type_counts[type_counts >= 99].index

#Filtering the DataFrame
filtered_vessel_info =
vessel_info[vessel_info['type'].isin(types_to_keep)]

print("Vessel information DataFrame filtered for types occurring at
least 99 times:")
print(filtered_vessel_info.head())

#3
#Using outer merge on 'mmsi'
merged_outer = pd.merge(vessel_info, transit_segments, on='mmsi',
how='outer')

print("Outer merged DataFrame:")
print(merged_outer.head())

#4
#Creating a duplicate to leave the the original outer merged DataFrame
untouched
merged_inner_simulated = merged_outer.copy()
merged_inner_simulated = merged_inner_simulated.dropna(subset=['name',
'type'])

print("Inner join simulated from outer join:")
print(merged_inner_simulated.head())

#5
#Using inner merge on 'mmsi'
merged_inner_direct = pd.merge(vessel_info, transit_segments,
on='mmsi', how='inner')
#Checking if both the results are the same
are_same = merged_inner_simulated.equals(merged_inner_direct)

print("Direct inner joined DataFrame:")
print(merged_inner_direct.head())
print(f"\nAre the results from the simulated inner join and direct
inner join exactly the same? {are_same}")
print("Analysis: The results should be exactly the same. The simulated
inner join correctly identifies and removes rows that did not have a
match in both original DataFrames by checking for NaN values
introduced by the outer join. This demonstrates the fundamental logic
of an inner join—keeping only the records with matching keys in both
tables.")

#6
#Saving the merged dataset
merged_inner_direct.to_csv('AIS_merge.csv', index=False)

#Checking for missing values
print("\nMissing values before replacement:")
print(merged_inner_direct.drop(columns=['st_time',
'end_time']).isnull().sum())

modes = merged_inner_direct.drop(columns=['st_time',
'end_time']).mode().iloc[0]

#Replacing missing values with the mode
merged_filled = merged_inner_direct.fillna(modes)

print("\nMissing values after replacement:")
print(merged_filled.drop(columns=['st_time',
'end_time']).isnull().sum())

#7
#Calculating z-score using a function
def detect_outliers_zscore(df, column):
    mean = df[column].mean()
    std = df[column].std()
    z_scores = (df[column] - mean) / std
    return z_scores

z_scores_seg_length = detect_outliers_zscore(merged_filled,
'seg_length')

#Assigning a threshold for z-score as in, z-score > 3 or < -3
outliers = merged_filled[(z_scores_seg_length > 3) |
(z_scores_seg_length < -3)]

print("\nPotential outliers in 'seg_length' based on Z-score > 3:")
print(outliers[['mmsi', 'name', 'seg_length']])
```

```
First 5 rows from vessel_information.csv:
   mmsi  num_names                                              names
sov  \
0     1          8  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
1     9          3                         000000009/Raven/Shearwater
N
2    21          1                                      Us Gov Vessel
Y
3    74          2                                  Mcfaul/Sarah Bell
N
4   103          3           Ron G/Us Navy Warship 103/Us Warship 103
Y

      flag flag_type  num_loas                                    loa
\
0  Unknown   Unknown         7  42.0/48.0/57.0/90.0/138.0/154.0/156.0

1  Unknown   Unknown         2                              50.0/62.0

2  Unknown   Unknown         1                                  208.0

3  Unknown   Unknown         1                                  155.0

4  Unknown   Unknown         2                             26.0/155.0


   max_loa  num_types                            type
0    156.0          4  Dredging/MilOps/Reserved/Towing
1     62.0          2                     Pleasure/Tug
2    208.0          1                          Unknown
3    155.0          1                          Unknown
4    155.0          2                   Tanker/Unknown

First 5 rows from transit_segments.csv:
   mmsi               name  transit  segment  seg_length  avg_sog
min_sog  \
0     1        Us Govt Ves        1        1         5.1     13.2
9.2
1     1  Dredge Capt Frank        1        1        13.5     18.6
10.4
2     1      Us Gov Vessel        1        1         4.3     16.2
10.3
3     1      Us Gov Vessel        2        1         9.2     15.4
14.5
4     1  Dredge Capt Frank        2        1         9.2     15.4
14.6

   max_sog  pdgt10        st_time       end_time
0     14.5    96.5  2/10/09 16:03  2/10/09 16:27
1     20.6   100.0   4/6/09 14:31   4/6/09 15:20
2     20.5   100.0   4/6/09 14:36   4/6/09 14:55
3     16.1   100.0  4/10/09 17:58  4/10/09 18:34
```

```
4     16.2   100.0  4/10/09 17:59  4/10/09 18:35
Vessel information DataFrame filtered for types occurring at least 99
times:
   mmsi  num_names                names sov                  flag
flag_type  \
2    21          1      Us Gov Vessel   Y               Unknown
Unknown
3    74          2  Mcfaul/Sarah Bell   N               Unknown
Unknown
5   310          1           Arabella   N              Bermuda
Foreign
6  3011          1         Charleston   N             Anguilla
Foreign
7  4731          1          000004731   N  Yemen (Republic of)
Foreign

   num_loas    loa  max_loa  num_types     type
2         1  208.0    208.0          1  Unknown
3         1  155.0    155.0          1  Unknown
5         1   47.0     47.0          1  Unknown
6         1  160.0    160.0          1    Other
7         1   30.0     30.0          1  Unknown
Outer merged DataFrame:
   mmsi  num_names                                              names
sov  \
0     1        8.0  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
1     1        8.0  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
2     1        8.0  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
3     1        8.0  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
4     1        8.0  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y

      flag flag_type  num_loas                                    loa
\
0  Unknown   Unknown       7.0  42.0/48.0/57.0/90.0/138.0/154.0/156.0

1  Unknown   Unknown       7.0  42.0/48.0/57.0/90.0/138.0/154.0/156.0

2  Unknown   Unknown       7.0  42.0/48.0/57.0/90.0/138.0/154.0/156.0

3  Unknown   Unknown       7.0  42.0/48.0/57.0/90.0/138.0/154.0/156.0

4  Unknown   Unknown       7.0  42.0/48.0/57.0/90.0/138.0/154.0/156.0


   max_loa  num_types  ...               name transit  segment
```

```
seg_length  \
0    156.0        4.0  ...        Us Govt Ves       1        1
5.1
1    156.0        4.0  ...  Dredge Capt Frank       1        1
13.5
2    156.0        4.0  ...      Us Gov Vessel       1        1
4.3
3    156.0        4.0  ...      Us Gov Vessel       2        1
9.2
4    156.0        4.0  ...  Dredge Capt Frank       2        1
9.2

   avg_sog  min_sog  max_sog  pdgt10        st_time       end_time
0     13.2      9.2     14.5    96.5  2/10/09 16:03  2/10/09 16:27
1     18.6     10.4     20.6   100.0   4/6/09 14:31   4/6/09 15:20
2     16.2     10.3     20.5   100.0   4/6/09 14:36   4/6/09 14:55
3     15.4     14.5     16.1   100.0  4/10/09 17:58  4/10/09 18:34
4     15.4     14.6     16.2   100.0  4/10/09 17:59  4/10/09 18:35

[5 rows x 21 columns]
Inner join simulated from outer join:
   mmsi  num_names                                              names
sov  \
0     1        8.0  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
1     1        8.0  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
2     1        8.0  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
3     1        8.0  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
4     1        8.0  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y

      flag flag_type  num_loas                                    loa
\
0  Unknown   Unknown       7.0  42.0/48.0/57.0/90.0/138.0/154.0/156.0

1  Unknown   Unknown       7.0  42.0/48.0/57.0/90.0/138.0/154.0/156.0

2  Unknown   Unknown       7.0  42.0/48.0/57.0/90.0/138.0/154.0/156.0

3  Unknown   Unknown       7.0  42.0/48.0/57.0/90.0/138.0/154.0/156.0

4  Unknown   Unknown       7.0  42.0/48.0/57.0/90.0/138.0/154.0/156.0


   max_loa  num_types  ...              name transit  segment
seg_length  \
0    156.0        4.0  ...        Us Govt Ves       1        1
```

```
5.1
1    156.0        4.0  ...  Dredge Capt Frank       1        1
13.5
2    156.0        4.0  ...      Us Gov Vessel       1        1
4.3
3    156.0        4.0  ...      Us Gov Vessel       2        1
9.2
4    156.0        4.0  ...  Dredge Capt Frank       2        1
9.2

   avg_sog  min_sog  max_sog  pdgt10        st_time       end_time
0     13.2      9.2     14.5    96.5  2/10/09 16:03  2/10/09 16:27
1     18.6     10.4     20.6   100.0   4/6/09 14:31   4/6/09 15:20
2     16.2     10.3     20.5   100.0   4/6/09 14:36   4/6/09 14:55
3     15.4     14.5     16.1   100.0  4/10/09 17:58  4/10/09 18:34
4     15.4     14.6     16.2   100.0  4/10/09 17:59  4/10/09 18:35

[5 rows x 21 columns]
Direct inner joined DataFrame:
   mmsi  num_names                                               names
sov  \
0     1          8  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
1     1          8  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
2     1          8  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
3     1          8  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y
4     1          8  Bil Holman Dredge/Dredge Capt Frank/Emo/Offsho...
Y

      flag flag_type  num_loas                                    loa
\
0  Unknown   Unknown         7  42.0/48.0/57.0/90.0/138.0/154.0/156.0

1  Unknown   Unknown         7  42.0/48.0/57.0/90.0/138.0/154.0/156.0

2  Unknown   Unknown         7  42.0/48.0/57.0/90.0/138.0/154.0/156.0

3  Unknown   Unknown         7  42.0/48.0/57.0/90.0/138.0/154.0/156.0

4  Unknown   Unknown         7  42.0/48.0/57.0/90.0/138.0/154.0/156.0


   max_loa  num_types  ...               name transit  segment
seg_length  \
0    156.0          4  ...        Us Govt Ves       1        1
5.1
1    156.0          4  ...  Dredge Capt Frank       1        1
```

```
13.5
2    156.0          4  ...      Us Gov Vessel       1        1
4.3
3    156.0          4  ...      Us Gov Vessel       2        1
9.2
4    156.0          4  ...  Dredge Capt Frank       2        1
9.2

   avg_sog  min_sog  max_sog  pdgt10        st_time       end_time
0     13.2      9.2     14.5    96.5  2/10/09 16:03  2/10/09 16:27
1     18.6     10.4     20.6   100.0   4/6/09 14:31   4/6/09 15:20
2     16.2     10.3     20.5   100.0   4/6/09 14:36   4/6/09 14:55
3     15.4     14.5     16.1   100.0  4/10/09 17:58  4/10/09 18:34
4     15.4     14.6     16.2   100.0  4/10/09 17:59  4/10/09 18:35

[5 rows x 21 columns]

Are the results from the simulated inner join and direct inner join
exactly the same? False
Analysis: The results should be exactly the same. The simulated inner
join correctly identifies and removes rows that did not have a match
in both original DataFrames by checking for NaN values introduced by
the outer join. This demonstrates the fundamental logic of an inner
join—keeping only the records with matching keys in both tables.

Missing values before replacement:
mmsi          0
num_names     0
names         0
sov           0
flag          0
flag_type     0
num_loas      0
loa           0
max_loa       0
num_types     0
type          0
name          0
transit       0
segment       0
seg_length    0
avg_sog       0
min_sog       0
max_sog       0
pdgt10        0
dtype: int64

Missing values after replacement:
mmsi          0
num_names     0
```

```
names         0
sov           0
flag          0
flag_type     0
num_loas      0
loa           0
max_loa       0
num_types     0
type          0
name          0
transit       0
segment       0
seg_length    0
avg_sog       0
min_sog       0
max_sog       0
pdgt10        0
dtype: int64

Potential outliers in 'seg_length' based on Z-score > 3:
             mmsi                  name  seg_length
74           3011            Charleston       121.6
205        439541  Canadian Warship 711       329.0
391        641114       Samantha Miller       120.2
519       1193046             Nauticast       296.8
527       1193046         Capt.hardhead       144.9
...           ...                   ...         ...
245218  636014120          Daishin Maru       184.2
248355  636090635        Cma Cgm Nilgai       132.0
260701  636092132            Bbc Winter       119.4
262196  888888882                Thomas       153.4
262198  888888882                Thomas       168.6

[1328 rows x 3 columns]
```

Q2. (30 points)

1. Use numpy to create array X of shape (4, 3). Each row represents one data point in 3 dimensions. Values should be random integers between 0 and 9 (inclusive). Set a random seed so the result is reproducible. Print X. (10points)

2. write a function dist() to measure the Euclidean distance (https://en.wikipedia.org/wiki/Euclidean_distance) between each pair of datapoints in X. Print the resulting with 3 decimals. (10points)

3. Consider adding a new point and add it to X using broadcasting.

   - A datapoint with coordinate (3, 1, 2);
   - A datapoint with two dimension (9, 6);
   - or a datapoint one dimension (2).

Discuss each case, can it do broadcasting or not.(10points)

```
#1
np.random.seed(42)

#Creating array X of shape (4, 3) filled with integers ranging from 0
to 9
X = np.random.randint(0, 10, size=(4, 3))

#Printing array 'X'
print("Array X:")
print(X)

#2
#Euclidean distance function
def dist(X):
    squared_distances = np.sum((X - X[:, np.newaxis])**2, axis=2)

    #Taking the square root to get the Euclidean distance.
    distances = np.sqrt(squared_distances)
    return distances

#Calculating the distance matrix
distance_matrix = dist(X)

print("\nEuclidean Distance Matrix:")
#Round to 3 decimal places for clean output
print(np.around(distance_matrix, 3))

#3
print("\nDiscussion on Broadcasting:")

#Case 1: A datapoint with coordinate (3, 1, 2)
new_point_1 = np.array([3, 1, 2])
try:
    #Broadcasting is possible here because the new point's shape (3,)
is
    #Compatible with the dimension of X(4, 3).
    # The new point is effectively "stretched" across the 4 rows of X.
    result_1 = X + new_point_1
    print("Case 1: Adding a point with coordinates (3, 1, 2)")
    print("Broadcasting is possible. Result:")
    print(result_1)
except ValueError as e:
    print("Case 1: Adding a point with coordinates (3, 1, 2)")
    print(f"Broadcasting is not possible. Error: {e}")

#Case 2: A datapoint with two dimensions (9, 6)
new_point_2 = np.array([9, 6])
try:
    #Broadcasting is not possible here.
    #X is (4, 3) and the new point is (2,n).
    #The last dimensions (3 and 2) are not equal and neither is 1 which violates the broadcasting rules.
    result_2 = X + new_point_2
    print("\nCase 2: Adding a point with two dimensions (9, 6)")
    print("Broadcasting is possible. Result:")
    print(result_2)
except ValueError as e:
    print("\nCase 2: Adding a point with two dimensions (9, 6)")
    print(f"Broadcasting is not possible. Error: {e}")

#Case 3: A datapoint with one dimension (2)
new_point_3 = np.array([2])
try:
    #Broadcasting is possible here.
    #The new point, (1,n), is compatible with X,(4, 3), because the last dimension is 1.
    #The new point is stretched across all elements of X.
    result_3 = X + new_point_3
    print("\nCase 3: Adding a point with one dimension (2)")
    print("Broadcasting is possible. Result:")
    print(result_3)
except ValueError as e:
    print("\nCase 3: Adding a point with one dimension (2)")
    print(f"Broadcasting is not possible. Error: {e}")
```

```
Array X:
[[6 3 7]
 [4 6 9]
 [2 6 7]
 [4 3 7]]

Euclidean Distance Matrix:
[[0.    4.123 5.    2.   ]
 [4.123 0.    2.828 3.606]
 [5.    2.828 0.    3.606]
 [2.    3.606 3.606 0.   ]]

Discussion on Broadcasting:
Case 1: Adding a point with coordinates (3, 1, 2)
Broadcasting is possible. Result:
[[ 9  4  9]
 [ 7  7 11]
 [ 5  7  9]
 [ 7  4  9]]

Case 2: Adding a point with two dimensions (9, 6)
Broadcasting is not possible. Error: operands could not be broadcast together with shapes (4,3) (2,)
```

```
Case 3: Adding a point with one dimension (2)
Broadcasting is possible. Result:
[[ 8  5  9]
 [ 6  8 11]
 [ 4  8  9]
 [ 6  5  9]]
```