exploring_pandas

January 20, 2025

1 Exploring pandas

The goal of this notebook is start exploring the `pandas` package, which we will use to handle data in Python. As usual, we start by importing the packages we'll need.

[1]:
```
#import pandas, numpy, seaborn, and matplotlib.pyplot
import pandas as pd #for data manipulation
import numpy as np #for numerical computation
import seaborn as sns #for data visualization and loading datasets
import matplotlib.pyplot as plt #for data visualization
```

The basic datastructure provided by `pandas` is the "Dataframe", which is basically a table of data. We will explore the following:

1. Creating dataframes in three ways
2. Getting basic info about the dataset
3. Locating data
4. Filtering data
5. Modifying and adding data
6. Saving data

1.0.1 1. Creating Dataframes

There are three main ways to create a pandas DataFrame:

1. From scratch using lists/dictionaries
2. Reading from a CSV file
3. Loading from a package like seaborn

Exercise 1: Creating dataframes

Running the cell below will create 3 dataframes. Display the top 5 rows of each using the command `df.head()`.

[2]:
```
# Method 1

# Generate two lists with hundred random numbers each, chosen in different ways.
x = np.random.normal(0, 1, 100)  # from normal distribution
y = np.random.uniform(-1, 1, 100)  # from uniform distribution

#Use a dictionary with column names as keys and lists as columns.
```

```
df = pd.DataFrame({'normal': x, 'uniform': y})

# Method 2
tips_df = pd.read_csv('../data/tips.csv')

#Method 3
titanic_df = sns.load_dataset('titanic')
```

[3]:
```
# Solution to exercise 1 here; create new cells for each dataframe.
```

1.0.2 2. Getting basic info about the dataset

When exploring a new DataFrame, these four commands are essential for understanding its structure:

```
df.shape        # display dimensions as (rows,columns)
df.columns      # display column names
df.info()       # Display data types and non-null counts for each column
df.describe()   # Generate statistical summary (count, mean, std, min, quantiles, max) for num
```

Exercise 2: Exploring DataFrame Properties

Using the Titanic and Tips datasets, perform the following tasks (don't hesitate to check the documentation for each command to undertand what its output represents):

1. Display the shape of both datasets. How many rows and columns are in each?
2. List all columns in both datasets. What types of information do they contain?
3. Use `info()` to examine the data types and missing values in each dataset.
4. Generate statistical summaries using `describe()`.

Compare your outputs. What are the main differences between these datasets in terms of: - Size (rows and columns) - Types of data (numerical vs categorical) - Missing (null) values

[4]:
```
# Solution to exercise 2; make new cells below if needed.
```

1.0.3 3. Locating data

Pandas provides several ways to access data in a DataFrame:

```
tips_df['total_bill']

# Select multiple columns (returns a DataFrame)
tips_df[['total_bill', 'tip']]

# Select by position using iloc[row, column]
tips_df.iloc[0]             # First row
tips_df.iloc[0:5]           # First 5 rows
titanic_df.iloc[:, 0]       # First column of Titanic dataset
titanic_df.iloc[0:5, 0:2]   # First 5 rows and 2 columns of Titanic dataset

# Select by label using loc[row_label, column_label]
```

```
tips_df.loc[0, 'tip']      # Value in first row, 'tip' column
tips_df.loc[0:4, 'tip']    # 'tip' values for first 5 rows
titanic_df.loc[0, 'age']   # Value in first row, 'age' column of Titanic dataset
titanic_df.loc[0:4, 'age'] # 'age' values for first 5 rows of Titanic dataset
```

Exercise 3: Locating data to answer questions

Tips Dataset: 1. What was the total bill for the first customer? 2. Display the tips given by the first 5 customers. 3. Find all bills that were more than $40. 4. Find all instances where a party of 4 people left a tip greater than $5. 5. Display the bills and tips for all Sunday dinner customers.

Titanic Dataset: 1. How many passengers were in first class? (Hint: Use `pclass == 1`) 2. What was the age of the youngest passenger? 3. Display the names of all female passengers who survived. 4. Find all passengers who paid more than $100 for their fare. 5. Show the age and class for all passengers between 20 and 25 years old.

[5]:
```
#Solution to exercise 3; make new cells below if needed.
```

1.0.4 4. Filtering data

```
tips_df[tips_df['total_bill'] > 50]                # Select rows where 'total_bill' is greater

# Multiple conditions using & (and), | (or)
tips_df[(tips_df['size'] > 3) & (tips_df['time'] == 'Dinner')]    # Select rows where 'size' i
titanic_df[(titanic_df['age'] < 18) | (titanic_df['fare'] > 100)]  # Rows where 'age' is less

# Using isin() for multiple values
tips_df[tips_df['day'].isin(['Sat', 'Sun'])]    # 'day' is a weekend
titanic_df[~titanic_df['embarked'].isin(['C', 'Q'])]   # 'embarked' is not 'C' or 'Q' (using ~

# String methods with str accessor
tips_df[tips_df['sex'].str.contains('Male')]    # 'sex' containing 'Male'
titanic_df[titanic_df['name'].str.endswith('Smith')]      # 'name' ending with 'Smith'
tips_df[tips_df['day'].str.lower() == 'friday']  # 'day' is 'friday' (case-insensitive matchin

# Null value filtering
titanic_df[titanic_df['age'].isna()]           # Rows where 'age' is missing a value
tips_df[tips_df['tip'].notna()]          # Rows where 'tip' is not missing a value

# between() for range filtering
titanic_df[titanic_df['fare'].between(20, 30)]  # Rows where 'fare' is at least 20 and at most
```

Exercise 4: Answering more complex questions by filtering

Tips Dataset: 1. What percentage of dinner bills had a tip greater than 20% of the total bill? 2. On which day of the week do smokers leave higher tips on average? 3. Find all instances where: - The party size was larger than 3 - The total bill was over $50 - It was dinner time 4. Compare the average tip percentage between: - Male vs Female customers - Smokers vs Non-smokers - Lunch vs Dinner times

Titanic Dataset: 1. What percentage of first-class female passengers survived? 2. Find all children

(age < 18) who: - Were traveling in third class - Had siblings/spouses aboard - Survived the disaster 3. Compare survival rates between: - Different passenger classes - Different age groups (child/adult/elderly) 4. Find the average fare paid by: - Survivors vs non-survivors - Males vs females in each passenger class

[6]:
```
#Starter code for exercise 4

# Tips dataset filters
expensive_bills = tips_df[tips_df['total_bill'] > 40]
large_parties = tips_df[(tips_df['size'] > 4) & (tips_df['time'] == 'Dinner')]

# Titanic dataset filters
first_survivors = titanic_df[(titanic_df['pclass'] == 1) &␣
 ↪(titanic_df['survived'] == 1)]
expensive_tickets = titanic_df[titanic_df['fare'] > 100]
young_third = titanic_df[(titanic_df['age'] < 10) & (titanic_df['pclass'] == 3)]
```

1.0.5 5. Modifying data

Common operations for modifying pandas DataFrames in place. If you don't want to alter the original dataframe, first make a copy with code like `df_copy = tips_df.copy()`.

```
# Basic modifications
tips_df.drop(['sex', 'smoker'], axis=1, inplace=True)      # Drop columns
tips_df.drop([0, 1], axis=0, inplace=True)                 # Drop rows by index
tips_df.drop_duplicates(inplace=True)                      # Drop duplicate rows

# Handle missing values
titanic_df.fillna(0, inplace=True)                           # Fill NA with constant
titanic_df['age'].fillna(titanic_df['age'].mean(), inplace=True) # Fill NA with mean
titanic_df.dropna(subset=['fare'], inplace=True)             # Drop rows with NA

# Vectorized operations (modify columns in place)
tips_df['total_bill'] *= 1.1                               # Increase by 10%
tips_df['total_bill'] = tips_df['total_bill'].clip(0, 100) # Limit range 0-100
tips_df['day'] = tips_df['day'].str.strip()                # Remove whitespace
tips_df.loc[tips_df['tip'] < 0, 'tip'] *= -1               # Make negatives positive
tips_df['total_bill'] /= tips_df['total_bill'].mean()      # Normalize a column
```

Exercise 5: Modifying data:

Tips Dataset: 1. What's the difference between average tip percentages on weekends vs weekdays? 2. What percentage of customers who spent over $50 left tips above 20%? 3. What's the average per-person cost difference between dinner and lunch? 4. Which day of the week had the highest average bill per person? 5. What percentage of large parties (>6 people) left tips above 15%?

Titanic Dataset: 1. Did passengers traveling with family have a higher survival rate than solo travelers? 2. What percentage of passengers who paid above twice the average fare survived? 3. What was the survival rate for children (under 18) in first class? 4. Which combination of gender and class had the highest survival rate? 5. What was the average age difference between survivors

and non-survivors?

[7]:
```
#Start code for exercise 5
# Tips dataset columns
tips_df['tip_pct'] = (tips_df['tip'] / tips_df['total_bill']) * 100
tips_df['per_person'] = tips_df['total_bill'] / tips_df['size']
tips_df['high_spender'] = tips_df['total_bill'] > 50
tips_df['large_party'] = tips_df['size'] > 6

# Titanic dataset columns
titanic_df['family_size'] = titanic_df['sibsp'] + titanic_df['parch']
titanic_df['solo'] = titanic_df['family_size'] == 0
titanic_df['child'] = titanic_df['age'] < 18
titanic_df['premium'] = titanic_df['fare'] > (2 * titanic_df['fare'].mean())
```

1.0.6 6. Grouping data

It is very important and useful to know how to group data together to make computations.

```
# Basic grouping by one column
tips_by_time = tips_df.groupby('time').size()                    # Count meals by time
surv_by_class = titanic_df.groupby('pclass')['survived'].mean()  # Survival rate by class

# Group by multiple columns
tips_day_time = tips_df.groupby(['day', 'time'])['tip'].mean()   # Mean tip by day & time
surv_class_sex = titanic_df.groupby(['pclass', 'sex'])['survived'].mean()  # Survival by class

# Common aggregation methods
tips_summary = tips_df.groupby('day').agg({
    'total_bill': 'mean',     # Mean bill per day
    'tip': 'sum',             # Total tips per day
    'size': ['min', 'max']    # Min and max party size per day
})

# Count and size
smoker_counts = tips_df.groupby('smoker').size()                 # Count customers by smoker s
class_counts = titanic_df.groupby('pclass')['survived'].count()  # Count passengers by class

# Group and filter
large_groups = tips_df.groupby('day').filter(lambda x: x['total_bill'].mean() > 20)  # Days wi
high_survival = titanic_df.groupby('pclass').filter(lambda x: x['survived'].mean() > 0.5)  # C
```

Exercise 6: Computing things by grouping data:

Tips dataset 1. What is the average tip amount for each day of the week? 2. Which time (lunch/dinner) has the highest total bills? 3. What is the average party size for smokers vs non-smokers? 4. What is the total amount spent by males vs females? 5. For each day, what percentage of customers were smokers?

Titanic dataset 1. What was the average fare paid in each class? 2. How many passengers of each

gender embarked from each port? 3. What was the average age of survivors vs non-survivors in each class? 4. What percentage of passengers survived in each age group (child/adult)? 5. What percentage of males vs females survived in each class?

Hint: Use `df.groupby()` followed by an aggregation function like `.mean()`, `.count()`, or `.sum()`.

[8]:
```
#Start code for exercise 6.

print("Average tips by day:")
print(tips_df.groupby('day')['tip'].mean())

print("\nSurvival rate by class:")
print(titanic_df.groupby('pclass')['survived'].mean())

print("\nPassenger counts by gender and port:")
print(titanic_df.groupby(['sex', 'embarked']).size())
```

```
Average tips by day:
day
Fri     2.734737
Sat     2.993103
Sun     3.255132
Thur    2.771452
Name: tip, dtype: float64

Survival rate by class:
pclass
1    0.629630
2    0.472826
3    0.242363
Name: survived, dtype: float64

Passenger counts by gender and port:
sex     embarked
female  C            73
        Q            36
        S           203
male    C            95
        Q            41
        S           441
dtype: int64
```

1.0.7 7. Creating new data

Very often, we need to compute things from a given dataset and need to keep track of them in new columns.

```
# Basic arithmetic with columns
tips_df['cost_per_person'] = tips_df['total_bill'] / tips_df['size']         # Per-person cost
```

```
tips_df['pct_of_max'] = tips_df['total_bill'] / tips_df['total_bill'].max() # Normalize by max
tips_df['total_amount'] = tips_df['total_bill'] + tips_df['tip']             # add two columns

# Creating columns with Boolean values
tips_df['high_tip'] = tips_df['tip_percent'] > 20                            # True if tip > 20%
titanic_df['is_child'] = titanic_df['age'] < 18                              # True if age < 18

# Creating columns by using the apply function
# Simple custom function
def age_group(age):
    if pd.isna(age): return 'unknown'
    elif age < 18: return 'child'
    else: return 'adult'
titanic_df['age_category'] = titanic_df['age'].apply(age_group)          # Categorize ages

# Lambda functions
tips_df['bill_category'] = tips_df['total_bill'].apply(lambda x: 'high' if x > 40 else 'low')

# Transformations using groupby
tips_df['tip_vs_day_avg'] = tips_df['tip'] / tips_df.groupby('day')['tip'].transform('mean')
```

Exercise 7: Computing things by creating new columns:

Tips Dataset: 1. Calculate the tip percentage for each bill. What's the average tip percentage? 2. Create a column for cost per person. Which day has highest per-person cost? 3. Label tips above 20% as 'high', others as 'normal'. How many high tips were there? 4. Add total amount (bill + tip) column. What's the highest total amount? 5. Flag bills above average as 'expensive'. What percentage of bills were expensive?

Titanic Dataset: 1. Create age groups: child (<18), adult (18-65), senior (>65). How many of each? 2. Add fare per family member column. What's the highest per-person fare? 3. Label passengers who paid above average fare as 'premium'. How many premium passengers survived? 4. Create male/female child flags. What percentage of female children survived? 5. Calculate each passenger's total family size (siblings + parents). What's the largest family?

[9]:
```
#Starter code for exercise 7; make new cells below if needed.
tips_df['tip_pct'] = (tips_df['tip'] / tips_df['total_bill']) * 100
tips_df['per_person'] = tips_df['total_bill'] / tips_df['size']
tips_df['tip_category'] = np.where(tips_df['tip_pct'] > 20, 'high', 'normal')

titanic_df['family_size'] = titanic_df['sibsp'] + titanic_df['parch']
titanic_df['fare_per_person'] = titanic_df['fare'] / (titanic_df['family_size']␣
 ↪+ 1)
titanic_df['child'] = titanic_df['age'] < 18
```

1.0.8 8. Saving datasets as csv files

Very similar to reading csv files:

```
df.to_csv('path/to/filename.csv')
```