Mudcard

- **The muddiest part for me was classifying different problems (regression, binary, etc). It isn't always clear which approach is best.**
- **I struggled with the classification of which type of target variable to use in different situations**
  - Yes, that's a pretty tough question and there is no "one answer fits all" solution generally because a problem can be phrased as various different supervised ML problems.
  - Always consider what you plan to do with the ML model once it is deployed and figure out what sort of supervised ML problem best suits your needs.
- **I'm still a bit confused about the ML pipeline.**
- **I think the ML pipeline was the most muddiest just out out familiarity**
  - We will go through each step in detail so don't worry about it.
- **What to choose for the project? What are the stuffs I need to consider for it.**
  - I'll send out an announcement about this in a few days.
- **For coding, this course use Python, but my undergraduate courses all used R.**
  - We start slow so you could use the first few weeks of the term to pick up python.
  - If that's not something you'd like to do, consider dropping the course.
- **So for the final project, is it possible to show/give us an example of what is expected for us to show?**
  - Yes, there is a final reports folder in the course repo which contains a couple of great final reports from previous years.
- **I was struggling a bit with the feature matrix and understanding the difference between that and target variables.**
  - The target variable is always the variable you want to predict with the ML model. All other variables are usually in the feature matrix.
  - There are some exceptions to it like unique IDs or group IDs, we will cover those later.
- **About the target variable concept, i believe including more example will be better for us to understand**
  - I'd consider it a responsible use of GenAI to ask it for more examples of each type of supervised ML problem.

Lecture 2

Working with data (step 0)

Let's get started with Step 0!

The supervised ML pipeline

0. Data collection/manipulation: you might have multiple data sources and/or you might have more data than you need

- you need to be able to read in datasets from various sources (like csv, excel, SQL, parquet, etc)
- you need to be able to filter the columns/rows you need for your ML model
- you need to be able to combine the datasets into one dataframe

1. Exploratory Data Analysis (EDA): you need to understand your data and verify that it doesn't contain errors

- do as much EDA as you can!

2. Split the data into different sets: most often the sets are train, validation, and test (or holdout)

- practitioners often make errors in this step!
- you can split the data randomly, based on groups, based on time, or any other non-standard way if necessary to answer your ML question

3. Preprocess the data: ML models only work if X and Y are numbers! Some ML models additionally require each feature to have 0 mean and 1 standard deviation (standardized features)

- often the original features you get contain strings (for example a gender feature would contain 'male', 'female', 'non-binary', 'unknown') which needs to be transformed into numbers
- often the features are not standardized (e.g., age is between 0 and 100) but it needs to be standardized

4. Choose an evaluation metric: depends on the priorities of the stakeholders

- often requires quite a bit of thinking and ethical considerations

5. Choose one or more ML techniques: it is highly recommended that you try multiple models

- start with simple models like linear or logistic regression
- try also more complex models like nearest neighbors, support vector machines, random forest, etc.

6. Tune the hyperparameters of your ML models (aka cross-validation or hyperparameter tuning)

- ML techniques have hyperparameters that you need to optimize to achieve best performance
- for each ML model, decide which parameters to tune and what values to try
- loop through each parameter combination
  - train one model for each parameter combination
  - evaluate how well the model performs on the validation set
- take the parameter combo that gives the best validation score
- evaluate that model on the test set to report how well the model is expected to perform on previously unseen data

7. Interpret your model: black boxes are often not useful

- check if your model uses features that make sense (excellent tool for debugging)
- often model predictions are not enough, you need to be able to explain how the model arrived to a particular prediction (e.g., in health care)

Learning objectives

By the end of the lecture, you will be able to

- list main issues with data selection and collection
- use pandas/polars to read in a dataset
- filter rows of a dataframe

Learning objectives

By the end of the lecture, you will be able to

- **list main issues with data selection and collection**
- use pandas/polars to read in a dataset
- filter rows of a dataframe

Data selection and collection issues

- the field is called **DATA** science for a reason!
  - data is the most important part of data science
  - the quality and quantity of data determines if a project is feasible
  - it is usually much more valuable than algorithms
  - working with data takes up ~80% of a data scientist's time

- when you start working on a new project, approach your dataset with healthy skepticism!
- ask questions in two main categories:
  - is the data appropriate for the problem you are trying to solve?
  - is the data accurate?

Is the dataset appropriate for the purpose?

- Can you answer your question with your data?
  - medical studies based on white men only, can you use it to diagnose everyone?
  - sometimes the answer is no! some diseases can differently impact man vs women or various racial groups
  - sometimes the answer is yes! some diseases manifest the same way in everyone
- Is your dataset timely?
  - goal: predict covid cases today
    - covid changed a lot over the years: spreads easier but symptoms are milder
    - covid data from 2020 and 2021 might not be useable today
  - goal: predict severe weather events like hurricanes
    - there are not many hurricanes in each year so the temptation is to use data going back as far as possible
    - hurricanes became more severe and more frequent in recent years
    - is it OK to use data from e.g., 1960s to predict hurricanes today?
- What biases are there in your dataset?
  - your ML model will learn any biases your data has
  - gender bias and racial bias are the main things to worry about when dealing with human data
- is your dataset legal, ethical, and reliable?
  - can you use the dataset legally?
    - protected attributes (such as gender or race) often cannot be used especially in finance for example
  - ethical usage
    - if data is collected for one purpose, can you use it to solve another problem?
  - reliability
    - are there any conficts of interests that might make the dataset unreliable?
    - example: climate data from big oil companies

Is the dataset accurate?

- typos and errors
  - mistakes by humans inputting data are extremely common!

- Why are there missing values in the dataset?
  - could be fine
    - some respondents didn't answer all the survey questions
    - doctor didn't perform test on all patients
  - could be because of instrucment malfunction
  - changes in data collection process over time
- How are the missing values represented?
  - sometimes as np.nan
  - sometimes a string like 'missing' or '?'
  - sometimes unreasonable values are used
- Are the values valid?
  - sometimes you'll see incorrect or impossible values
  - 6 digit zip codes
  - people older than 200 years
  - negative numbers for a quantity that can only be non-negative
- Duplicate records
  - could be due to data entry error or data manipulation error (incorrect merge or append)

Documentation

- document your dataset!
- your future self and anyone else trying to reproduce your work will thank you!
- can be as simple as a text file
- describe each column in your dataset
  - what is described in the column?
  - what quantity is measued? does it have a unit?
  - what's the range of valid values?
  - what possible categories could there be? what does each category mean?
  - are there missing values? if there are, why?
  - how are the missing values represented?
- you will see examples of this already today!
- **if the dataset for your final project does not come with documentation, you need to write one!**

Quiz

Learning objectives

By the end of the lecture, you will be able to

- list main issues with data selection and collection
- **use pandas/polars to read in a dataset**
- filter rows of a dataframe

Pandas and Polars - why should you care?

- **when you work on an ML problem, you might work with data from various sources**
  - healthcare data might come from hospitals, insurance companies, state/federal agencies, etc.
  - finance data could come from banks, brokerage accounts, social security office, etc.
  - **you will need to pull data from all of these different sources and create one combined dataset ready for ML**
- **you might also have more data than you need to solve the ML problem**
  - in healthcare, you might be interested in people who have a certain symptom, or maybe you are interested in people who visited the ER multiple times
  - in finance, you might be required by law to not use sensitive or protected attributes eventhough you have access to them
  - **you need to filter out the rows and columns you need**
- packages like pandas and polars make this easy for you

Pandas and polars intro

Similarities:

- both are packages used for data manipulation and analysis
- both use the concept of data frames and series - we will talk about this more later today!
- both support reading and writing data in various formats (like CVS, excel, SQL, JSON, etc.)
- syntax is similar -- polars uses syntax similar to pandas to make it easier for pandas users to switch over :)

Differences:

- pandas is more established (released in 2008) so it has a large userbase, great manuals, large community to help with issues
- polars is a pretty new package, it has only been around since 2020 but the userbase is rapidly growing
- pandas integrates with more packages than polars (although that's changing)
- polars is much faster than pandas on large datasets

When to use pandas?

- Most companies have exensive code bases in pandas and it is unlikely they will switch over anytime soon, it's too costly.
- If you work with small to medium datasets and computational speed and memory efficiency are not that critical, pandas is fine.

When to use polars?

- If you work on large datasets.
- If computational speed and memory efficiency are mission-critical

Keep in mind that you can use both packages to make the best of both worlds!

- Use polars to perform the heavy computations!
- Use polars.to_pandas() and polars.from_pandas() to convert dataframes as needed!

Check out this site to see how you can convert pandas code to polars.

Check out this site for a list of libraries and tools that support polars.

In [1]:
```
# how to read in a database into a dataframe and basic dataframe structure
import pandas as pd

# load data from a csv file
df_pd = pd.read_csv('../data/adult_data.csv') # there are also pd.read_excel

print(df_pd)
#help(df_pd.head)
#print(df_pd.head()) # by default, shows the first five rows but check help(
#print(df_pd.shape) # the shape of your dataframe (number of rows, number of
#print(df_pd.shape[0]) # number of rows
#print(df_pd.shape[1]) # number of columns
```

```
       age          workclass  fnlwgt    education  education-num  \
0       39          State-gov   77516    Bachelors             13
1       50   Self-emp-not-inc   83311    Bachelors             13
2       38            Private  215646      HS-grad              9
3       53            Private  234721         11th              7
4       28            Private  338409    Bachelors             13
...    ...                ...     ...          ...            ...
32556   27            Private  257302   Assoc-acdm             12
32557   40            Private  154374      HS-grad              9
32558   58            Private  151910      HS-grad              9
32559   22            Private  201490      HS-grad              9
32560   52       Self-emp-inc  287927      HS-grad              9

            marital-status          occupation    relationship    race  \
0            Never-married        Adm-clerical   Not-in-family   White
1       Married-civ-spouse     Exec-managerial         Husband   White
2                 Divorced   Handlers-cleaners   Not-in-family   White
3       Married-civ-spouse   Handlers-cleaners         Husband   Black
4       Married-civ-spouse      Prof-specialty            Wife   Black
...                    ...                 ...             ...     ...
32556   Married-civ-spouse        Tech-support            Wife   White
32557   Married-civ-spouse   Machine-op-inspct         Husband   White
32558              Widowed        Adm-clerical       Unmarried   White
32559        Never-married        Adm-clerical       Own-child   White
32560   Married-civ-spouse     Exec-managerial            Wife   White

           sex  capital-gain  capital-loss  hours-per-week  native-country
\
0         Male          2174             0              40   United-States
1         Male             0             0              13   United-States
2         Male             0             0              40   United-States
3         Male             0             0              40   United-States
4       Female             0             0              40            Cuba
...        ...           ...           ...             ...             ...
32556   Female             0             0              38   United-States
32557     Male             0             0              40   United-States
32558   Female             0             0              40   United-States
32559     Male             0             0              20   United-States
32560   Female         15024             0              40   United-States

      gross-income
0            <=50K
1            <=50K
2            <=50K
3            <=50K
4            <=50K
...            ...
32556        <=50K
32557         >50K
32558        <=50K
32559        <=50K
32560         >50K

[32561 rows x 15 columns]
```

In [ ]:
```
# how to read in a database into a dataframe and basic dataframe structure
import polars as pl

# load data from a csv file
df_pl = pl.read_csv('../data/adult_data.csv') # there are also pd.read_excel
# check out pl.scan_csv() too! It is useful if you know you only need to wor
# it will only read in the necessary columns!

print(df_pl)
#help(df_pl.head)
#print(df_pl.head()) # by default, shows the first five rows but check help(
#print(df_pl.shape) # the shape of your dataframe (number of rows, number of
#print(df_pl.shape[0]) # number of rows
#print(df_pl.shape[1]) # number of columns
```

Packages

A package is a collection of classes and functions.

- a dataframe (pd.DataFrame()) is a pandas class
  - a class is the blueprint of how the data should be organized
  - classes have methods which can perform operations on the data (e.g., .head(), .shape)
- df is an object, an instance of the class.
  - when we put data into a class, it becomes an object
  - methods are attached to objects
    - you cannot call pd.head(), you can only call df.head()
- read_csv is a function
  - functions are called from the package
  - you cannot call df.read_csv, you can only call pd.read_csv()

DataFrame structure: both rows and columns are indexed!

- index column, no name
  - contains the row names
  - by default, index is a range object from 0 to number of rows - 1
  - any column can be turned into an index, so indices can be non-number, and also non-unique. more on this later.
- polars dataframes do not have an index column by default! rows are indexed by their integer position in the table
  - you can add an index column if you'd like though
- columns with column names on top in both polars and pandas

Always print your dataframe to check if it looks good!

Most common reasons it might not look ok:

- the first row is not the column name
  - there are rows above the column names that need to be skipped
  - there is no column name but by default, pandas assumes the first row is the column name. as a result, the values of the first row end up as column names.
- character encoding is off
- separator is not comma but some other charachter

In [ ]:
```
# check the help to find the solution
help(pd.read_csv)
```

Quiz

The adult_test.csv file is located in the data folder as well. It is a test set of the adult dataset so you would expect the same column names and generally a similar-looking structure. Read in the file using pandas or polars in the cell below. Make sure the dataframe looks good.

In [ ]:
```
# add your pandas code below


# add your polars code below
```

Learning objectives

By the end of the lecture, you will be able to

- list main issues with data selection and collection
- use pandas/polars to read in a dataset
- **filter rows of a dataframe**

Filter rows in pandas

- let's assume you have one dataframe to work with but you have too much data and you need to filter out some rows
- there are several ways to do that

1) Integer-based indexing, numpy arrays are indexed the same way.

2) Select rows based on the value of the index column

3) select rows based on column condition

1) Integer-based indexing, numpy arrays are indexed the same way.

In [ ]:
```
# df_pd.iloc[] - for more info, see https://pandas.pydata.org/pandas-docs/st
# iloc is how numpy arrays are indexed (non-standard python indexing)

# [start:stop:step] -  general indexing format

# start stop step are optional
#print(df_pd)
#print(df_pd.iloc[:])
#print(df_pd.iloc[::])
#print(df_pd.iloc[::1])

# select one row - 0-based indexing
#print(df_pd.iloc[0])

# indexing from the end of the data frame
#print(df_pd.iloc[-2])
```

In [ ]:
```
# select a slice - stop index not included
#print(df_pd.iloc[3:7])

# select every second element of the slice - stop index not included
#print(df_pd.iloc[3:7:2])

#print(df_pd.iloc[3:7:-2]) # return empty dataframe
#print(df_pd.iloc[7:3:-2])#  return rows with indices 7 and 5. 3 is the stop

# can be used to reverse rows
#print(df_pd.iloc[::-1])

# here is where indexing gets non-standard python
# select the 2nd, 5th, and 10th rows
#print(df_pd.iloc[[1,4,9]]) # such indexing doesn't work with lists but it w
```

2) Select rows based on the value of the index column

In [ ]:
```
# df_pd.loc[] - for more info, see https://pandas.pydata.org/pandas-docs/sta

#print(df_pd.index) # the default index when reading in a file is a range in
                # .loc and .iloc works ALMOST the same.
# one difference:
#print(df_pd.loc[3:9:2]) # this selects the 4th, 6th, 8th, 10th rows - the s

#help(df_pd.set_index)
```

In [ ]:
```
df_index_age = df_pd.set_index('age',drop=False)

#print(df_index_age.head())
#print(df_index_age.index)
#print(df_index_age.head())

print(df_index_age.loc[30].head()) # collect everyone with age 30 - the inde
```

3) select rows based on column condition

In [ ]:
```
# one condition
print(df_pd[df_pd['age']==30].head())
# here is the condition: it's a boolean series - series is basically a datat
#print(df_pd['age']==30)

# multiple conditions can be combined with & (and) | (or)
#print(df_pd[(df_pd['age']>30)&(df_pd['age']<35)].head())
#print(df_pd[(df_pd['age']==90)|(df_pd['native-country']==' Hungary')])
```

Filter rows in polars

1) Integer-based indexing

- there are no .loc[] or .iloc[] methods in polars but you can still select rows as you would with numpy arrays
- df_pl[start:stop:step]

In [ ]:
```
# example:
df_pl[1:10:2]
```

2) Select rows based on column conditions

- syntax is similar expect use .filter()

In [ ]:
```
df_pl.filter((df_pl['age']==30)&(df_pl['native-country']==' India'))
```

Quiz

How many people in adult_data.csv work at least 60 hours a week and have a doctorate?

In [ ]:
```
# add your pandas code below


# add your polars code below
```

Mud card