Module 6 Assignment

December 1, 2024

1 Module 5 - Deployment of predictive models, dissemination of knowledge and additional topics

1.0.1 Assignment overview

For this assignment, it is possible to work in **groups of up to 2 students**. Read the instructions carefully, as they may assign tasks to specific students.

1.0.2 Group members

Leave blanks if group has less than 2 members: - Student 1: Yanxin Liang 50798412 - Student 2: Yelia Ye 89657605

1.0.3 Learning Goals:

After completing this week's lecture and tutorial work, you will be able to: 1. Provide examples of what happens when poorly designed models are deployed
2. Recognize the impact of technologies on at risk populations 3. Recognize and discuss the impacts of addictive online environments that optimize on metrics that do not benefit its users 4. Discuss how data science techniques can be used to mislead 5. Identify how one's identity connects to interpretation of results (e.g., confirmation bias) 6. Discuss the ethics of displaying and reporting findings
7. Explain Goodhart's Law: when the metric becomes the target 8. Understand and assess the environmental impact of data science and AI applications

2 Goodhart's law - scenario

In class, we cited Goodhart's law:

*

When a measure becomes a target it ceases to be a good measure.

-

Let's see how this plays out in a real-life scenario. Start by reading this article by Cathy O'Neil, based on her book *Weapons of Math Destruction*: "How Big Data Transformed Applying to College".

This article describes the importance of a particular metric, the *U.S. News & World Reports* college rankings, for colleges and students applying to those colleges. These scores are supposed to provide a quantitative, objective measure of the quality of an institution. Colleges know that students and their parents believe in the importance of these rankings, and will go to great lengths to make sure their score is high. This system, however, presents several issues and has had negative consequences on students.

Question 1 Let's start by looking at the algorithm used to produced the rankings. How would you evaluate this algorithm using our FATE lenses? Do you see issues of fairness, accountability, and/or transparency?

From a fairness perspective, the ranking algorithms clearly disadvantage less affluent or poor students. In the article, it was mentioned that the college admissions process has become increasingly dominated by competing algorithms that continue to drive up tuition costs. This has resulted in poorer students having a harder time paying expensive tuition. It was also mentioned in the article that colleges use algorithms to predict, among other things, how much financial aid a student will need, and that they will be more likely to want to admit wealthier students or those who are less in need of financial aid. This puts poorer students at a disadvantage. In order to compete for high-end students, universities will spend money on luxury dormitories and athletic facilities, but these luxury campus configurations do not enhance academics.

From an accountability perspective, the ranking methodology mentioned in the article clearly lacks accountability. Some colleges blatantly falsify data, such as faking SAT scores, graduation rates, retention rates, and acceptance rates, in order to improve their rankings. The fact that these falsifications can be carried out also reflects the lack of oversight by the relevant authorities in the collection of data related to the calculation of rankings.

From a transparency perspective, the ranking process is opaque, making it difficult for students and parents to understand how decisions are made. As mentioned in the article article, 'The rankings rely on a dubious and opaque scoring algorithm that changes over time and may or may not reflect someone's preferences.' It is difficult for students and parents to know how much weight different factors carry in the decision. This lack of transparency results in applicants not choosing colleges that are better suited for them on a case-by-case basis.

From an ethical standpoint, overall this idea that college rankings are prioritised over quality education is ethically problematic. The existence of this viewpoint focuses schools on how to improve their reputation rather than how to provide a better quality education and how to improve the school's academic The behaviours that universities engage in to improve their reputation can lead to higher costs for students or families, perpetuating the inequalities mentioned previously.

Question 2 List at least 3 ways, cited in the articles, in which colleges are trying to inflate their ranking, and explain why these actions, while boosting a college's score, are not effective at improving the college's quality or the students' experience.

1. Universities intentionally misrepresent relevant data, such as SAT scores and graduation rates, in order to improve their rankings. The article mentions that Bucknell University, Emory University, and Iona College have falsified average SAT scores over the years to improve their rankings. While this has improved their rankings, the rankings do not represent the true academic as well as educational level of the school. They spend their resources on how to lie and cheat instead of taking steps to improve their academics and the quality of their education.

2. In order to achieve higher rankings, schools will also falsify the employment rates of recent graduates. The article mentions that 'law schools will hire some graduates on a temporary basis and even count students who do not respond to employment surveys as "employed" '. It is still possible that a student may not find a job after the contract of temporary employment expires. The falsified employment rate data does not really reflect a school's ability to cultivate students' vocational ability and cannot reflect the real employment status of college students.

3. colleges will try to attract high-end students in order to improve their rankings. For this purpose, schools will build luxurious hostels and sports facilities. The construction of these facilities has no academic significance and consumes a lot of money that could have been spent on educational resources such as laboratory equipment. Expensive and luxurious facilities mean higher tuition fees, which is a huge financial burden for ordinary or poor families. This does not help to improve the academic standards and educational capacity of the college.

Overall, these practices have neglected the academic and educational functions that should be the most important concern of colleges, and spent a lot of resources on improving rankings. At the same time, it has increased inequality and the financial burden on students' families. It doesn't really care about the needs and experiences of students.

Question 3 The article mentions that metrics boosting a college's ranking include low acceptance rates (a sign of exclusivity) and high enrollment rates from accepted students (a sign of high desirability). It also explains that this resulted in an increase in applications to different schools (*"the number of students applying to at least seven has tripled since 1990, from 9 percent to 29 percent"*) and in the *death of the concept of safety school.* What do you think the author means by this? What is the connection between the U.S. News rankings and the chances of being invited to apply to a "safety school"?

For clarity, here is a definition of "safety school", according to the Princeton Review: *"A safety school is one where your academic credentials exceed the school's range for the average first-year student. You should be reasonably certain that you will be admitted to your safety schools".*

According to the article, low acceptance rates and high acceptance student enrollment are important indicators of college rankings. It is mentioned in the article that high acceptance rates are detrimental to rankings and in order to get better rankings, universities go through various tactics in order to lower the acceptance rate and increase the proportion of accepted applicants actual enrolling. this increases the competition in the admissions process. This has led to a significant increase in anxiety among applicants awaiting admission results. In order to alleviate this anxiety, the number of schools each person applies to becomes larger to ensure a higher probability of acceptance. This is also reflected in the article, 'the number of students applying to at least seven has tripled since 1990, from 9 percent to 29 percent'. Originally, safe schools would definitely or with a high probability admit those applicants who met the requirements, which is a very secure guarantee for the applicants. Safe schools exist to give applicants the opportunity and peace of mind to pursue offers from better schools, and because rankings have become so important, the original safe schools need to lower their acceptance rates in order to achieve better rankings, so they will reject some qualified applicants. Safe schools no longer provide a reliable guarantee for qualified applicants, so the concept of safe schools died out. With the demise of safety schools, the pressure on students and parents is increasing. The U.S. News rankings pressure colleges, including safety schools, to reject more students in order to lower acceptance rates to improve their rankings. Safety schools encourage students to apply more often, but the likelihood of being accepted is greatly reduced. They will reject applicants who already meet the requirements. This gives them a lower acceptance rate. This severely reduces the likelihood that a student will feel confident that they will be accepted to a safe school.

3 Addictive Online Environments

The addictive power of some technologies and online environments is becoming widely known. In class, we talked about some of the typical features a developer may include in their app or website to make them more addictive. The list includes: - endless scrolling - pull-to-refresh - never-ending auto-play videos - push notifications - temporarily available stories - likes - read-receipts

As part of this exercise, we would like each member of the group to go through their phone and find one app with addictive features and one without, and describe them here. Make sure to pick apps you feel comfortable talking about. Answer the question by editing the templates below (one for each student - if you are working alone, you can delete the second template or leave it blank).

3.0.1 Student 1: - Yanxin Liang -

App with addictive features **App name:** Red

App main functionality/goal: Red is a social media and user-generated content platform where users can share their lives by posting short videos, text, or images. It combines features of Pinterest, Instagram, and Amazon, offering a space to explore life guides, travel tips, product reviews, and topics of interest such as news, handmade crafts, fashion, cars, and more.

Decription of addictive features: Red presents information in a Pinterest-style layout enhanced with video and live-streaming features. I find its endless scrolling, pull-to-refresh, and personalized content recommendations particularly addictive. Using artificial intelligence and machine learning, Red analyzes user searches and interactions to tailor highly relevant content. This creates a seamless and engaging experience that often turns casual browsing into an unconscious habit.

App frequency of use: (weekly? daily? hourly?)

If you are comfortable doing so, describe your use of the app. Do you open it only when you need to complete a specific task? Or also in an unplanned, more spontaneous manner? How long do you stay on the app after opening it? Do you find some app features compelling to open it or to use it for a longer time (e.g. push notifications)?

I use the app hourly, both for specific tasks and spontaneous browsing. When I need to search for useful tutorials, such as cooking tips or product reviews, I turn to Red for suggestions. Also, its personalized recommendations make it easy to discover content that matches my interests. This often leads me to spend a lot of time to scroll through engaging posts as long as I open this app. #### App without addictive features **App name:** Google map

App main functionality/goal: Google map provides detailed information about geographical regions and sites worldwide. It shows directions and uses real-time traffic information to find the best route to destinations.

App frequency of use: (weekly? daily? hourly?)

If you are comfortable doing so, describe your use of the app. Do you open it only when you need to complete a specific task? Or also in an unplanned, more spontaneous manner? How long do you stay on the app after opening it? Do you find some app features compelling to open it or to use it for a longer time, which were not listed as addictive features?

I use Google Maps daily, primarily to check bus schedules and find the most efficient route to my destination. I stay on the route page until I am confident about the information, which usually takes around five minutes. While its features are generally task-oriented, the location photos, especially images of restaurant dishes, sometimes compel me to stay longer and explore more about it.

3.0.2 Student 2: - Yelia Ye -

App with addictive features **App name:** Red

App main functionality/goal: A short video and social media platform for sharing lifestyle tips, personal experiences, and shopping recommendations.

Decription of addictive features: Red uses endless scrolling and never-ending auto-play videos to keep users engaged. Its algorithm curates personalized content based on my interests, making it hard to stop scrolling. Push notifications also frequently prompt me to open the app, adding to its addictive nature. Red is very similar to TikTok in its design and user engagement strategies.

App frequency of use: (weekly? daily? hourly?) Hourly

If you are comfortable doing so, describe your use of the app. Do you open it only when you need to complete a specific task? Or also in an unplanned, more spontaneous manner? How long do you stay on the app after opening it? Do you find some app features compelling to open it or to use it for a longer time (e.g. push notifications)?

I often open Red both intentionally and impulsively, especially when I am trying to procrastinate during homework or tasks. The app easily distracts me and disrupts my plans. Sometimes, I check it every few hours, and each session often lasts much longer than I intend as I lose track of time while scrolling.

App without addictive features **App name:** iClicker

App main functionality/goal: A classroom engagement tool used for responding to polls or quizzes during lectures.

App frequency of use: (weekly? daily? hourly?) Weekly or only during class sessions when required.

If you are comfortable doing so, describe your use of the app. Do you open it only when you need to complete a specific task? Or also in an unplanned, more spontaneous manner? How long do you stay on the app after opening it? Do you find some app features compelling to open it or to use it for a longer time, which were not listed as addictive features?

I use iClicker solely for academic purposes, opening it during lectures when my professor asks us to respond to a poll or quiz. The app is highly task-focused and lacks features like notifications or curated content, which makes it non-addictive. Once I complete the required task, I close the app and don't use it until it is needed again.

4 Misleading Data Science Techniques

There are many ways in which data science may lead to the wrong conclusions, either by error or intentionally. A very good summary of such techniques is offered by Carl Bergstrom and Jevin West in their course "Calling Bullshit in the Age of Big Data", offered at the University of Washington and with recordings available online.

We are going to review some of these techniques here (most of them have to do with visualization):

- Dataviz in the popular media.
- Misleading axes.
- Manipulating bin sizes.
- Dataviz ducks.
- Glass slippers.
- The principle of proportional ink.
- Correlation vs. Causation

Now that you have familiarized with some misleading data science techniques, let's try to evaluate a real scenario. Here is an article, published on Nature in 2004, claiming that trends in the winning times of the men's and women's Olympic 100-meters sprint show that women are closing the time gap, and will one day (some time around 2156, to be exact) complete the race faster than men athletes.

Question 4 Do you think that the claim made by the article is true? If not, what mistake(s) did the authors commit that led them to this wrong conclusion?

We think that the claim made by the article is false. Firstly, Figure 1 in the article shows that the graph plots the winning times of the men's and women's Olympic finals over the past 100 years against the competition date and based on this graph the conclusion is obtained that women finished the games faster than men athletes around the year 2156. But in fact this conclusion is not reasonable, since the date of the competition and the time of finished the game do not constitute a causal relationship, we can only say that during this 100-year period, the date of the competition and the time of completion of the competition have a certain correlation. The conclusion of the article ignores all other factors that could potentially affect the time an athlete takes to complete a race, such as training methods and age of entry. We can also notice that during the analysis, the researcher continued a long way back based on the regression line found. Along the regression line it can be seen that the predicted race completion times get faster and faster, but this is not scientific. Human athletic performance is subject to physiological limitations and it is not possible to run faster and faster with no upper limit, so the subsequent trend should gradually level out. Furthermore, the graph is potentially misleading and violates the principle of proportional ink, as the regression lines extend well beyond the range of available data by more than a factor of three and result in an unscientific intersection, which is a visual emphasis on future predictions that is not supported by reliable evidence.

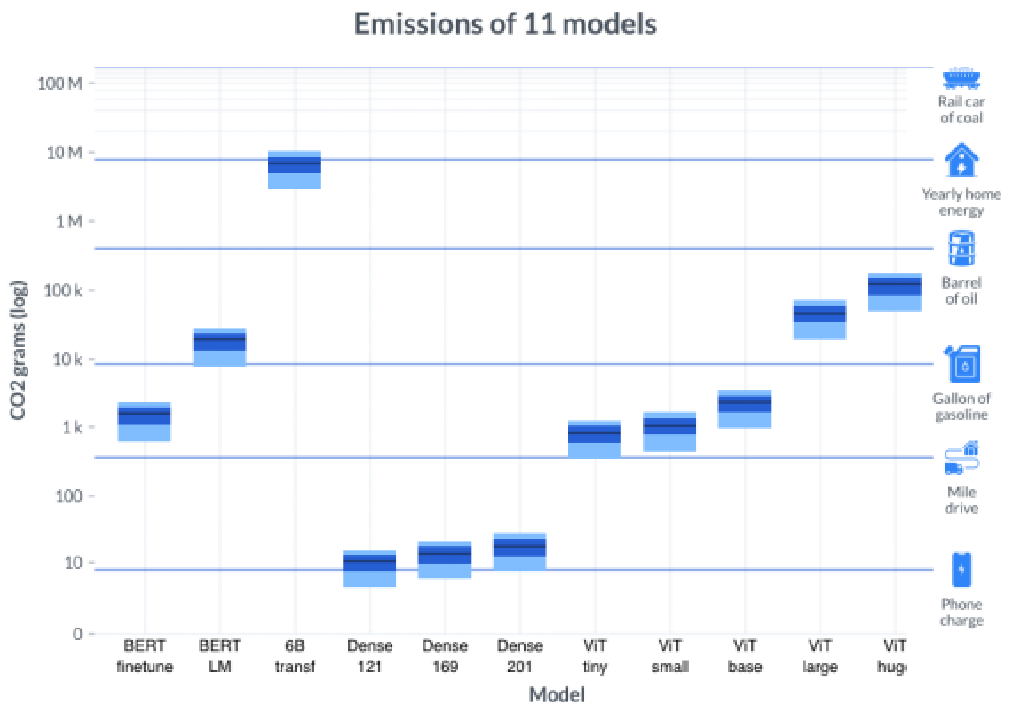
5 AI and the environment

Source: The Carbon Footprint of Artificial Intelligence

AI applications are becoming ubiquitous in society, from recommendation algorithms on streaming services and social media, to semi-autonomous vehicles, large language models and more. Few people, however, including few of the developers of these systems, spend time thinking about their environmental impact. Even fewer try to evaluate this impact.

The cost of training and maintaining AI algorithms is not immediately evident. The most obvious is the massive amount of electricity that they require. Depending on how that electricity is produced, there is also a variable cost in terms of CO2 released in the athmosphere. And we should not forget the cost of building and maintaining the hardware on which AI algorithms run, from the extraction of the raw materials, to the manifacturing and transportation of this hardware, concluding with the cost of disposing of it once it reached the end of its lifespan. Recent estimates of AI's carbon footprint range from 2.1% to 3.9% of the total greenhouse gas emissions. Still a small fraction of what is produced by more polluting industries such as manufacturing (24%) and transportation (27%), but not insignificant.

In April 2022, AI startup Hugging Face released an article with an estimate of the carbon footprint of one of its own AI models, comprehensive of all emissions produced during the model's whole life cycle. They estimated that the training alone of their model resulted in 25 metric tons of CO2 emissions. When they included the emissions produced by the manufacturing of the hardware and the broader computing infrastructure, and the energy required to run the trained model, the esitmate doubled to 50 metric tons of CO2 emissions, similar to the amount of CO2 produced by 11 gasoline-powered cars driven for one year.



Training neural network models takes more time and energy than simply using them for a single prediction. Here are some estimates for the emissions of 11 popular NLP and computer vision models.

Jesse Dodge, a research scientist at the Allen Institute for AI, said: "When AI developers create new systems, they're trying to push performance, accuracy, or the model's capabilities, and they focus less on efficiency". And it turns out that this focus on accuracy above everything else is costly. Here is a quote from a 2022 paper by Mill et al.:

There is a recognised trade-off between model accuracy and energy efficiency. In fact, the relationship has been shown to be logarithmic. That is, in order to achieve a linear improvement in accuracy, an exponentially larger model is required. A recent study […] confirmed the existence of the accuracy-energy trade off […] and indicated a 30-50% saving in energy for training related to a 1% reduction in accuracy.

In this portion of the assignment, we will increase our awareness of the carbon footprint of our algorithms, thanks to a software tool called CodeCarbon. Let's start by installing the required packages in your environment (you can delete or comment out this cell after you are done).

[2]:
```
# !pip install codecarbon
# !pip install transformers
```

This next cell is for importing CodeCarbon, as well as other libraries we will need for this exercise.

[3]:
```
from codecarbon import EmissionsTracker
from transformers import pipeline

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import xgboost as xgb
from sklearn.compose import ColumnTransformer, make_column_transformer
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer
from xgboost import XGBClassifier
from scipy.stats import lognorm, loguniform, randint
from sklearn.model_selection import (
    GridSearchCV,
    RandomizedSearchCV,
    cross_val_score,
    cross_validate,
    train_test_split,
)
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.svm import SVC
```

```
c:\Users\27790\miniconda3\envs\DSCI430\lib\site-packages\tqdm\auto.py:21:
TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
c:\Users\27790\miniconda3\envs\DSCI430\lib\site-
packages\torchvision\io\image.py:13: UserWarning: Failed to load image Python
extension: '[WinError 127]      'If you don't plan on using image
functionality from `torchvision.io`, you can ignore this warning. Otherwise,
there might be something wrong with your environment. Did you have `libjpeg` or
```

```
`libpng` installed before building `torchvision` from source?
  warn(
c:\Users\27790\miniconda3\envs\DSCI430\lib\site-
packages\torchvision\datapoints\__init__.py:12: UserWarning: The
torchvision.datapoints and torchvision.transforms.v2 namespaces are still Beta.
While we do not expect major breaking changes, some APIs may still change
according to user feedback. Please submit any feedback you may have in this
issue: https://github.com/pytorch/vision/issues/6753, and you can also check out
https://github.com/pytorch/vision/issues/7319 to learn more about the APIs that
we suspect might involve future changes. You can silence this warning by calling
torchvision.disable_beta_transforms_warning().
  warnings.warn(_BETA_TRANSFORMS_WARNING)
c:\Users\27790\miniconda3\envs\DSCI430\lib\site-
packages\torchvision\transforms\v2\__init__.py:54: UserWarning: The
torchvision.datapoints and torchvision.transforms.v2 namespaces are still Beta.
While we do not expect major breaking changes, some APIs may still change
according to user feedback. Please submit any feedback you may have in this
issue: https://github.com/pytorch/vision/issues/6753, and you can also check out
https://github.com/pytorch/vision/issues/7319 to learn more about the APIs that
we suspect might involve future changes. You can silence this warning by calling
torchvision.disable_beta_transforms_warning().
  warnings.warn(_BETA_TRANSFORMS_WARNING)
```

Let's see how the tool works in a toy example. In this cell, we are first creating an EmissionTracker. Then, we can call start() to start tracking the power required by following operations, thus producing an estimate of the associated carbon emissions. Stop tracking by calling stop() at the end of the operations you want to produce an estimate for. In this case, we are estimating the cost of 100,000 simple additions.

[4]:
```python
tracker = EmissionsTracker(log_level='error')
tracker.start()
try:
    for i in range(100000):
        _ = 1 + 1
finally:
     tracker.stop()
```

```
c:\Users\27790\miniconda3\envs\DSCI430\lib\site-
packages\codecarbon\output_methods\file.py:52: FutureWarning: The behavior of
DataFrame concatenation with empty or all-NA entries is deprecated. In a future
version, this will no longer exclude empty or all-NA columns when determining
the result dtypes. To retain the old behavior, exclude the relevant entries
before the concat operation.
  df = pd.concat([df, pd.DataFrame.from_records([dict(total.values)])])
```

[5]:
```python
# Visualize the results
tracker.final_emissions_data
```

[5]:
```
EmissionsData(timestamp='2024-11-28T12:52:46', project_name='codecarbon',
run_id='ea2661d6-2017-4725-b4de-62563263fcc6',
experiment_id='5b0fa12a-3dd7-45bb-9766-cc326314d9f1',
duration=0.7922214999562129, emissions=1.3339333908308522e-07,
emissions_rate=1.6837884239503476e-07, cpu_power=17.5,
gpu_power=20.605496693742644, ram_power=5.77703332901001,
cpu_energy=3.850777291437327e-06, gpu_energy=4.512503610002577e-06,
ram_energy=1.2339993527299668e-06, energy_consumed=9.59728025416987e-06,
country_name='Canada', country_iso_code='CAN', region='british columbia',
cloud_provider='', cloud_region='', os='Windows-10-10.0.22631-SP0',
python_version='3.10.14', codecarbon_version='2.8.0', cpu_count=16,
cpu_model='AMD Ryzen 9 5900HS with Radeon Graphics', gpu_count=1, gpu_model='1 x
NVIDIA GeForce RTX 3060 Laptop GPU', longitude=-122.84, latitude=49.4645,
ram_total_size=15.40542221069336, tracking_mode='machine', on_cloud='N',
pue=1.0)
```

Let's now try a more complex task: question answering. Question answering is a task in Natural Language Processing (NLP) which, as the name suggests, aims at producing an answer to a question asked in natural language. For this test will use the framework created by Hugging Face (transformers).

[6]:
```
tracker = EmissionsTracker(log_level='error')
tracker.start()  # Starting the tracker
try:
    nlp = pipeline('question-answering')  # transformer; for more info, check␣
 ↪https://huggingface.co/docs/transformers/task_summary#question-answering
    print(nlp({
    'question': 'What was on the steps?',
    'context': "He walked down the steps from the train station in a bit of a␣
 ↪hurry knowing the secrets in the briefcase must be secured as quickly as␣
 ↪possible. Bounding down the steps, he heard something behind him and quickly␣
 ↪turned in a panic. There was nobody there but a pair of old worn-out shoes␣
 ↪were placed neatly on the steps he had just come down. Had he past them␣
 ↪without seeing them? It didn't seem possible. He was about to turn and be on␣
 ↪his way when a deep chill filled his body."}))
finally:
     tracker.stop()  # Stopping the tracker
```

```
No model was supplied, defaulted to distilbert/distilbert-base-cased-distilled-
squad and revision 564e9b5 (https://huggingface.co/distilbert/distilbert-base-
cased-distilled-squad).
Using a pipeline without specifying a model name and revision in production is
not recommended.
c:\Users\27790\miniconda3\envs\DSCI430\lib\site-
packages\huggingface_hub\file_download.py:139: UserWarning: `huggingface_hub`
cache-system uses symlinks by default to efficiently store duplicated files but
your machine does not support them in
C:\Users\27790\.cache\huggingface\hub\models--distilbert--distilbert-base-cased-
```

```
distilled-squad. Caching files will still work but in a degraded version that
might require more space on your disk. This warning can be disabled by setting
the `HF_HUB_DISABLE_SYMLINKS_WARNING` environment variable. For more details,
see https://huggingface.co/docs/huggingface_hub/how-to-cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to
run Python as an administrator. In order to activate developer mode, see this
article: https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-
device-for-development
  warnings.warn(message)
c:\Users\27790\miniconda3\envs\DSCI430\lib\site-
packages\transformers\pipelines\question_answering.py:391: FutureWarning:
Passing a list of SQuAD examples to the pipeline is deprecated and will be
removed in v5. Inputs should be passed using the `question` and `context`
keyword arguments instead.
  warnings.warn(
```

```
{'score': 0.6361868381500244, 'start': 258, 'end': 286, 'answer': 'a pair of old
worn-out shoes'}
```

[7]:
```
tracker.final_emissions_data
```

[7]:
```
EmissionsData(timestamp='2024-11-28T12:54:25', project_name='codecarbon',
run_id='9d7af2f8-ecd1-4de8-baec-199d472117c8',
experiment_id='5b0fa12a-3dd7-45bb-9766-cc326314d9f1',
duration=76.46770329994615, emissions=1.4520833322191457e-05,
emissions_rate=1.8989498436004005e-07, cpu_power=17.5,
gpu_power=32.28260733956501, ram_power=5.77703332901001,
cpu_energy=0.0003717142838886098, gpu_energy=0.0005620890607819992,
ram_energy=0.0001109302820238845, energy_consumed=0.0010447336266944936,
country_name='Canada', country_iso_code='CAN', region='british columbia',
cloud_provider='', cloud_region='', os='Windows-10-10.0.22631-SP0',
python_version='3.10.14', codecarbon_version='2.8.0', cpu_count=16,
cpu_model='AMD Ryzen 9 5900HS with Radeon Graphics', gpu_count=1, gpu_model='1 x
NVIDIA GeForce RTX 3060 Laptop GPU', longitude=-122.84, latitude=49.4645,
ram_total_size=15.40542221069336, tracking_mode='machine', on_cloud='N',
pue=1.0)
```

Question 5 Report here the CO2 emissions generated by the two tasks, in grams (`emissions` in the output): - Additions: 0.00013339333908308522 grams - Question answering: 0.014520833322191457 grams

Question 6 In a 2009 estimate, Google said each query on its site generates 0.2 g of carbon dioxide, much higher than the toy examples above. In 2023, it was estimated that there are 99000 search queries per second run on the site. Assuming these numbers are accurate, how many **kilograms of carbon dioxide** are generated in a year by Google searches?

[11]:
```
99000 * 0.2 * 365 * 24 * 3600 / 1000
```

[11]:
```
624412800.0
```

624412800 kilograms of carbon dioxide are generated in a year by Google searches.

Question 7 The result of the previous question may not be very meaningful to a non-expert in environmental sciences. Thankfully, the U.S. Environmental Protection Agency (EPA) has released a tool, called Greenhouse Gas Equivalencies Calculator, to convert these numbers in more understandable equivalencies. Try inserting your result from the previous question in the tool, and answer this question: **how many pounds of coal should be burned to produce the same amount of CO2 as a year of Google searches?**

693,607,040 pounds of coal should be burned to produce the same amount of CO2 as a year of Google searches.

Finally, let's evaluate the carbon footprint of one of the models used in this course. In the cells below, you can find the code used to train a random forest model on the NIJ recidivism data used in Assignment 3. Look at this code and add two EmissionTracker, one to track the emissions produced when using the random forest model without fine tuning, and another one to track the emissions produced when fine-tuning the random forest model using 20 combinations of parameters. Include training and scoring in your tracking (but not preprocessing). Comments in the code should help you find the right place to insert the trackers.

[13]:
```
# Importing the training and test sets (you may need to adjust this code to␣
↪match your folder location)

train_df = pd.read_csv("training_set.csv")
test_df = pd.read_csv("testing_set.csv")

# Separating features and targets

X_train, y_train = (
    train_df.drop(columns=["Recidivism_Within_3years"]),
    train_df["Recidivism_Within_3years"],
)
X_test, y_test = (
    test_df.drop(columns=["Recidivism_Within_3years"]),
    test_df["Recidivism_Within_3years"],
)
```

[14]:
```
# Feature preprocessing

numeric_feats = [
    "Residence_PUMA",
    "Supervision_Risk_Score_First",
    "Avg_Days_per_DrugTest",
    "DrugTests_THC_Positive",
    "DrugTests_Cocaine_Positive",
    "DrugTests_Meth_Positive",
    "DrugTests_Other_Positive",
    "Percent_Days_Employed",
    "Jobs_Per_Year"
]  # apply scaling and imputation
categorical_feats = ["Gender",
                     "Race",
                     "Age_at_Release",
                     "Gang_Affiliated",
                     "Supervision_Level_First",
                     "Education_Level",
                     "Dependents",
                     "Prison_Offense",
                     "Prison_Years",
                     "Prior_Arrest_Episodes_Felony",
                     "Prior_Arrest_Episodes_Misd",
                     "Prior_Arrest_Episodes_Violent",
                     "Prior_Arrest_Episodes_Property",
                     "Prior_Arrest_Episodes_Drug",
                     "Prior_Arrest_Episodes_PPViolationCharges",
                     "Prior_Conviction_Episodes_Felony",
                     "Prior_Conviction_Episodes_Misd",
                     "Prior_Conviction_Episodes_Prop",
                     "Prior_Conviction_Episodes_Drug",
                     "Delinquency_Reports",
                     "Program_Attendances",
                     "Program_UnexcusedAbsences",
                     "Residence_Changes",
                     ]  # apply one-hot encoding
passthrough_feats = ["Prior_Arrest_Episodes_DVCharges",
                "Prior_Arrest_Episodes_GunCharges",
                "Prior_Conviction_Episodes_Viol",
                "Prior_Conviction_Episodes_PPViolationCharges",
                "Prior_Conviction_Episodes_DomesticViolenceCharges",
                "Prior_Conviction_Episodes_GunCharges",
                "Prior_Revocations_Parole",
                "Prior_Revocations_Probation",
                "Condition_MH_SA",
                "Condition_Cog_Ed",
                "Condition_Other",
                "Violations_ElectronicMonitoring",
                "Violations_Instruction",
                "Violations_FailToReport",
                "Violations_MoveWithoutPermission"
                ]  # Already binary
drop_feats = ["ID",
              "Recidivism_Arrest_Year1",
              "Recidivism_Arrest_Year2",
              "Recidivism_Arrest_Year3",
              "Training_Sample"]   # features not of interest
target = "Recidivism_Within_3years"

ct = make_column_transformer(
    (
        make_pipeline(SimpleImputer(), StandardScaler()),
        numeric_feats,
    ),  # scaling on numeric features
    (
        ␣
 ↪make_pipeline(SimpleImputer(strategy="most_frequent"),OneHotEncoder(handle_unknown="ignore"
        categorical_feats,
    ),  # OHE on categorical features
    ("passthrough", passthrough_feats),  # no transformations on the binary␣
 ↪features
    ("drop", drop_feats),  # drop the drop features
)
```

[15]:
```
# Fitting feature transformer and transforming training features

X_train_transformed = ct.fit_transform(X_train)
column_names = numeric_feats + list(
    ct.named_transformers_["pipeline-2"].get_feature_names_out(
        categorical_feats
    )
) + passthrough_feats
```

[16]:
```
results = {}  # Dictionary to collect results
emissions = []  # List to collect emission results from the tracker
                # You can add a result to this list after the tracker has␣
 ↪stopped by calling
                # emissions.append(tracker.final_emissions_data.emissions)

# We will use this function to fit and evaluate the models using␣
 ↪cross-validation
def mean_std_cross_val_scores(model, X_train, y_train, **kwargs):
    """
    Returns mean and std of cross validation
    """
    scores = cross_validate(model, X_train, y_train, **kwargs)

    mean_scores = pd.DataFrame(scores).mean()
    std_scores = pd.DataFrame(scores).std()
    out_col = []

    for i in range(len(mean_scores)):
        out_col.append((f"%0.3f (+/- %0.3f)" % (mean_scores[i], std_scores[i])))
    return pd.Series(data=out_col, index=mean_scores.index)
```

[17]:
```
tracker1 = EmissionsTracker(log_level='error')
tracker1.start()
try:
    # Pipeline including the column transformer and the random forest classifier

     pipe_forest = Pipeline([
          ('ct', ct),
          ('rf', RandomForestClassifier(class_weight="balanced",␣
 ↪random_state=2))])


     # Calling mean_std_cross_val_scores to fit and evaluate the model, and add␣
 ↪the results to our dictionary

     results["random_forest"] = mean_std_cross_val_scores(
     pipe_forest, X_train, y_train, return_train_score=True
     )
finally:
     tracker1.stop()

emissions.append(tracker1.final_emissions_data.emissions)
```

```
C:\Users\27790\AppData\Local\Temp\ipykernel_15348\1274509884.py:18:
FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a
future version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  out_col.append((f"%0.3f (+/- %0.3f)" % (mean_scores[i], std_scores[i])))
```

[18]:
```
# Now, we are going to tune the random forest model across 2 parameters:␣
 ↪max_depth and n_estimators
tracker = EmissionsTracker(log_level='error')
tracker.start()
try:
    param_grid_rf = {
        "rf__n_estimators": randint(low=10, high=100),
        "rf__max_depth": randint(low=2, high=20),
    }

    random_search_rf = RandomizedSearchCV(
        pipe_forest,
        param_grid_rf,
        n_iter=20,   # This parameter specifies the number of combinations to␣
 ↪try (20, in this case)
        verbose=1,
        n_jobs=-1,
        random_state=123,
        return_train_score=True,
    )

    # Fine-tuning the random forest model
    random_search_rf.fit(X_train, y_train);

    # Final fit and evaluation of the random forest model using the best␣
 ↪paramaters found during tuning
    best_rf_model = random_search_rf.best_estimator_
    results["random forest (tuned)"] = mean_std_cross_val_scores(
        best_rf_model, X_train, y_train, return_train_score=True
    )
finally:
    tracker.stop()

emissions.append(tracker.final_emissions_data.emissions)
```

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
```

```
C:\Users\27790\AppData\Local\Temp\ipykernel_15348\1274509884.py:18:
FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a
future version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  out_col.append((f"%0.3f (+/- %0.3f)" % (mean_scores[i], std_scores[i])))
```

[27]:
```
# Run this cell to visualize some information on the models (including training␣
 ↪and testing accuracy), and their emissions

results_df = pd.DataFrame(results)
results_df.loc['emissions (g of CO2)'] = [term*1000 for term in emissions]
results_df
```

[27]:
```
                          random_forest random forest (tuned)
fit_time              5.501 (+/- 0.097)     3.428 (+/- 0.017)
score_time            0.232 (+/- 0.013)     0.156 (+/- 0.003)
test_score            0.723 (+/- 0.010)     0.725 (+/- 0.009)
train_score           1.000 (+/- 0.000)     0.966 (+/- 0.002)
emissions (g of CO2)           0.006748              0.013435
```

Question 8

- What was the improvement in accuracy (on the test set) after fine tuning?
- How many more grams of CO2 were produced for this improvement? You can answer in the form "Fine tuning produces X many times more CO2"

1. After fine-tuning, the test accuracy increased from 0.723 to 0.725. This implies that fine-tuning can improve the performance of the prediction model, but the improvement might not

be significant.

2. Fine tuning produces 2 times more CO2.

6 Final thoughts

1) If you have completed this assignment in a group, please write a detailed description of how you divided the work and how you helped each other completing it:

We discussed our thoughts on the questions and shared our ideas with each other. And we divide the work so that each person could focus on specific parts of the assignment. We used GitHub to share our progress and make sure everything stayed organized. Once we completed all the questions, we came together to review the answers as a group, making sure that everything was clear and correct.

2) Have you used ChatGPT or a similar Large Language Model (LLM) to complete this homework? Please describe how you used the tool. We will never deduct points for using LLMs for completing homework assignments, but this helps us understand how you are using the tool and advise you in case we believe you are using it incorrectly.

We used ChatGPT to help us correct some grammar mistakes in some parts of our written responses.

3) Have you struggled with some parts (or all) of this homework? Do you have pending questions you would like to ask? Write them down here!

We've struggled to find the units used for measuring CO2 emissions. Additionally, we would like to explore the costs and resources involved in fine-tuning technology to reduce these emissions. Are there any established methods or guidelines for balancing technological development with CO2 reduction to achieve sustainable progress.