DSCI 430 - Fairness, Accountability, Transparency and Ethics (FATE) in Data Science

Module 1 - Introduction and Ethical foundations

Assignment overview

This assignment is composed of three parts:

- **Part 1 - The Black Mirror Writers' Room.** In this portion of the exercise, you will brainstorm near future technology and its possible drawbacks, and illustrate them in a futuristic cautionary tale. You will also be asked questions about ethical theories and how they apply to the scenario you have described. Credits: Casey Fiesler - The Black Mirror Writers Room: The Case (and Caution) for Ethical Speculation in CS Education
- **Part 2 - Python review.** As this course uses Python as the programming language for our exercises, a basic understanding of the fundamentals and the use of some libraries is necessary. This portion of the exercise will help you review useful Python syntax and/or fill the gap in your knowledge before tackling larger exercises. We recommend discussing with an instructor if you find this portion of the assignment too difficult to complete with a reasonable amount of effort.
- **Part 3 - Final thoughts.** Complete this section so that we can better understand how you completed the assignment and any issues you may have encountered.

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.

Group members

Leave Student 2 blank if group has less than 2 members:

- Student 1: Nicholas Tam (45695970)
- Student 2: Jingyuan Liu (69763183)

Learning Goals:

After completing this week's lecture and tutorial work, you will be able to:

1. Define ethics and describe what constitutes an ethical issue
2. Explain the need for ethics in data science
3. Identify common ethical issues in data science
4. Describe common ethical frameworks and how they can be applied to data science applications

5. Imagine scenarios in which current technology could be used in unethical ways
6. Evaluate and make arguments around data science scenarios using ethical theories (e.g., Kantianism, utilitarianism, virtue ethics etc.)

7. Compare and contrast different ethical theories and explain the case for and against each one as they apply to data science

The Black Mirror Writers' Room

Black Mirror is a Netflix series centered around the use of advanced technology and its possible unexpected (sometimes catastrophic) consequences. In this exercise, you will come up with your very own Black Mirror episode (or at least a synopsis)!

Warm up

Before jumping into the creative writing part, we should review the various elements of FATE in Data Science and make sure that they are clear:

FATE element	Definition
Fairness	The idea that every group or population that is affected by a technological application is being treated equally and not receiving a different outcome *solely because they belong to their group*.
Accountability	Clear definition of who should be held responsible of the outcome of the technological application and under what circumstances.
Transparency	The technical definition of transparency in Data Science refers to being able to understand why a technological application produced a specific outcome. This is also called *explainability*. But transparency can also refer to the demand of making the use of algorithms more transparent to the public, including informing the users about when they are used, where the data used was sourced from, and making algorithms available for auditing.
Ethics	Evaluation of whether or not a technology should be used based on the moral values of a group or society. Society may reject a technology because it is does not follow the principles of Fairness, Accountability or Transparency, but also for other reasons.

Question 1

Consider the following scenario:

In the country of Dataland, the police department uses an algorithm to assess the risk level of people reporting cases of domestic abuses and violence. Thanks to this algorithm, they can identify the most serious threats and intervene accordingly. The algorithm has had a positive impact, assessing cases with more accuracy than other prior strategies and allowing the police force to make an efficient use of their resources. However, it occasionally fails to correctly identify people at high risk of violence (*false negatives*), leaving them without the protection they need. It is also affected by other issues. For each issue outlined in this table, check whether it is a Fairness, Accountability or Transparency problem.

Issue	Fairness	Accountability	Transparency
When the algorithm fails to identify a high-risk case and violence occurs, it is unclear if the police department should shoulder any responsibility.		✔	
An analysis of the algorithm's results suggests that false negatives occur more frequently among victims with physical disabilities.	✔		
The majority of people reporting domestic abuse are not aware that their cases are being evaluated by an algorithm, or do not know the score they received.			✔
The police department receives a recommendation for each case, but does not know which characteristic(s) of the case have resulted in the final evaluation.			✔
The algorithm was trained using past cases filed by the police department, but the			✔

Issue	Fairness	Accountability	Transparency
people involved where not informed that their information was being used for this purpose.			

Question 2

Considere the issues outlined in the previous question, as well as the fact that the algorithm is the best system of appraisal available to the police forces so far for cases of domestic violence. Do you think that the use of this algorithm is *ethical*? Clearly state your thesis (opposed/favourable) and use one of the ethical perspectives listed in this reading to support it.

We would argue that we are in favor of the statement that the use of this algorithm is ethical, using the common good perspective. By the common good perspective of ethics, actions are ethical if they provide maximum increase in happiness, welfare, health, security, and sustainability for the groups and communities involved. In theory, the algorithm would reduce and discourage domestic violence, while allowing more efficiency within police force operations, and thus increase community happiness and security. However, the false negatives occuring more with disabled individuals could lead to those victims being less willing to rely on the police, and the police being unaware of the characteristics leading to the final evaluation could lead to misunderstandings that escalate issues further.

Despite these potential drawbacks, we would argue that the use of the algorithm would be ethical through the common good perspective of ethics.

Note: this case is fictional but inspired by a real algorithm, called VioGén, used in Spain to determine the risk level of victims of gender-based violence and assign protection measures. The algorithm has been recently going under severe scrutiny (Read more).

Question 3: Write your own Black Mirror episode

Now that you have acquired the necessary familiarity with some required knowledge and terminology, it is time to use your creativity!

Step 1: Brainstorm *one* near future technology based on a topic of your choice. It should be close enough that it seems like a plausible future. Describe it in the next cell.

Disease prediction using predictive machine learning models. For example, in the near future, a patient just needs to tell the machine learning model what symptoms they have and the AI doctor can carry out a bunch of testing (i.e. blooding testing, CT scanning, etc). Then the model would predict the disease the patient has and the potential treatments for the patient based on the symptoms and testing results. This model can be used in disease diagnosis (i.e. whether the tumor is benign or malignant) and treatment recommendations. Overall, the model is well-trained, and its misdiagnosis rate of most diseases is much lower, on average, than that is done by real doctors. This is the best-ever model we have so far for disease diagnosis and prediction.

Step 2: What are the potential social implications and/or ethical issues and/or regulatory challenges with this technology? Explain if and how they are connected to FATE (e.g. is it a Fairness issue? Or maybe a Transparency issue? It could be more than one option).

- Fairness: Disparities in disease effects and likelihoods between gender, racial and age groups (e.g. Sickle cell anemia), leading to potential differences in false positive and negatives.

Refined answer (Fairness):

- Fairness: The model is stated to be well-trained, but there is no guarantee that the dataset the model is trained on will have sufficient representation of certain portions of the population, leading to certain underrepresented minorities obtaining less effective diagnoses compared to the majority of the population. For instance, European populations are found to be more vulnerable to hemochromatosis, while African populations are found to be more vulnerable to sickle cell anemia, yet the model may not reflect that due to being trained in Vancouver and thus provide less effective diagnoses for such diseases.

- Accountability: Who claims responsibility upon misdiagnosis, especially with the risk of the data input being fudged deliberately or otherwise?

- Transparency: Need to inform previous patients on whether or not their medical information can be used for training the model; discern the disease the model is specifically trained to identify, and whether or not the characteristics important for model classification and their corresponding results are similar to those used by doctors.

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10764412/#:~:text=Defining%20fairness%20in%20healthcare&text=In%20the%20context%20of%20radiology,social%20status%20or%20ethnic%20differences.

Step 3: Time for storytelling! Write the summary of a new Black Mirror episode based on the technology of your choice. Try covering all of the following:

- What do you think might be a cautionary tale related to this technology?
- What fictional person in the future would best illustrate this caution? Provide a detailed description and explain what makes them the best character to carry your message.
- What is their story? Explain their background, their motivations, and their journey through your episode.

As part of your submission, please update the episode thumbnail slide (from Module 1 slide deck) using information from your episode. Don't forget to add a picture! Then, share it with the rest of the class on Canvas.

A machine used for all medical diagnoses in the nation fails to detect a patient’s rare form of cancer, leading to a treatment that escalates their condition. By the time the doctor is informed of the model’s failure, the sudden decline of their patient's health, and their staff’s inaction, they can do nothing but watch as their fatal mistake forces their patient to waste away.

- Complacency resulting from having a machine learning model doing diagnosis for medical staff, careless usage of medical treatments with drawbacks.
- A patient that is almost out of the window of treatment to be saved, as one could argue that this is the worst case scenario when medical treatment gets delayed.

- They intially start off hopeful as they get a potential second chance at life, raise their reservations about the effectiveness of the model while everyone else is dismisses their concerns, get increasingly uncomfortable during the treatment, break down in horror as they are informed that their health has taken for the worse because of the treatment while they had been misdiagnosed, and die in frustrated despair.

The episode thumbnail slide can also be found here.

Refined answer:

Dr. Dale had gotten complacent and dissatisfied as MeDCal, a machine learning model with a 99% accuracy in disease prediction, becomes the new standard for all medical diagnostics in the nation. However, a patient by the name of Casey has their condition consistently misdiagnosed for Lyme disease instead of lung cancer, leading to treatments that fail to alleviate their condition. By the time Dale is informed of the model's failure, the sudden decline of their patient's health, and their staff's inaction, they can do nothing but watch as their fatal mistake forces Casey to waste away.

- Complacency resulting from having a machine learning model doing diagnosis for medical staff, careless usage of medical treatments with drawbacks.
- Dale, the doctor involved in the patient's treatment. One could argue that the patient deteriorating past the point of no return from incorrect medical intervention is the worst case scenario when medical treatment gets delayed, and Dale will likely suffer the consequences of the patient's death due to potentially holding accountability for such, and the lack of transparency on the diagnosis means that more time is wasted on identifying the alternative conditions the patient is suffering.
- Dale started off highly comitted to helping others before the invention of the ML model, but slowly got demotivated and complacent as it took over most of their work. They and their medical staff dismissed Casey's reservations about the effectiveness of the model, and eventually hit the horrifying realisation that they missed the opportunity to prevent Casey's condition from becoming terminal. By the end, Dale can only collapse catatonically as the consequences of their mistakes finally hit them.

The episode thumbnail slide can also be found here.

Step 4: Let's take a step back, and imagine that you are (one of) an activist/legistlator/technology practitioner at the time the technology described in your episode is being developed and its use being discussed. Select *one* of the ethical perspectives listed in this reading, and use this perspective to argue against its deployment. Pay particular attention to the counter-arguments! People with interests in this technology will certainly argue against you, and you must anticipate and rebut their claims. Include at least 2 conter-arguments and how you would respond to them.

Ethical perspective chosen: Care ethics perspective.

Argument: Is it ethical to delegate medical diagnosis to machine learning models? The use of an AI predictive model for disease diagnosis may weaken the relationship and connections between a patient and a doctor. Patients are no longer facing real doctors, and the AI doctor reduces the level of care and comfort patients receive in a clinic.

First counter-argument (with rebuttal): Common good perspective: Since this model has a much lower misdiagnosis rate than doctors on average, it has direct health benefits for people in the community and society; their life expectancy is expected to increase with the implement of this new model. The overall well-being of the community is likely to improve.

Second counter-argument (with rebuttal): Doctors will likely remain involved with the patient's treatment, and will need to be able to contextualise the model's diagnosis to their patients for reassurance, mitigating the potential detatchment from using the diagnosis model.

Refined answer:

Ethical perspective chosen: Care ethics perspective.

Argument: Is it ethical to delegate medical diagnosis to machine learning models? The use of an AI predictive model for disease diagnosis may weaken the relationship and connections between a patient and a doctor. Patients are no longer facing real doctors, and the AI doctor reduces the level of care and comfort patients receive in a clinic.

First counter-argument (with rebuttal): Common good perspective: Since this model has a much lower misdiagnosis rate than doctors on average, it has direct health benefits for people in the community and society; their life expectancy is expected to increase with the implement of this new model. The overall well-being of the community is likely to improve as a consequence. Rebuttal: However, the model is likely to be relatively costly in terms of both purchase and maintenance, this does not account for reassuring people that are unwilling to entirely trust such a model, and it is possible that the doctors will charge more to diagnose with the model than without, so less wealthy individuals will have more trouble benefitting from the model's implementation.

Second counter-argument (with rebuttal): Doctors will likely remain involved with the patient's treatment, and will need to be able to contextualise the model's diagnosis to their patients for reassurance, mitigating the potential detatchment from using the diagnosis model. Rebuttal: However, since the model is the one doing the diagnosis for the doctor, the patient's initial impression of the doctor may be less positive regardless, and those that are unreceptive to the model entirely may think the doctor as incompetent for having to rely on it for medical diagnosis.

- Types of cancer & common misdiagnoses. Marciano Legal. (n.d.). https://marcianolegal.com/types-of-cancer-common-misdiagnoses/#:~:text=Many%20bacterial%20and%20fungal%20infections,poses%20its%20own%20significant%20problems.

Step 5: Finally, let's end on a more positive note and imagine a "Light Mirror" scenario, where the negative consequences of the technology you have described are averted and positive results are achieved in their stead. Try answering the following questions:

1. What kinds of solutions can be deployed in the immediate for addressing the harms of the technology you have described? What could we do to ensure that we don’t get to the negative consequences you imagined later in the future?

2. Could you imagine a scenario where the technology you have described is used with positive consequences, given the appropriate safe guards?

1. Depending on the condition of interest or even in all cases, informed consent could be required for usage of their data for model testing and evaluation. Having the doctor continue to be involved with the diagnosis by contextualising the model’s results could also mitigate issues with care ethics between the patient and the doctor. The model could also provide multiple potential diagnoses through the generated probability scores.

2. If the model is designed to work for multiple types of diseases, the model could diagnose a different medical condition that neither the patient nor the doctor were expecting, allowing them to potentially cure the patient before the condition escalates to become life-threatening. The model could also be used for the prediction of disease outbreaks, depending on the location of the populace of interest.

- Learn medical diagnosis with machine learning: Advantages and limitations. KnowledgeNile. (2023, July 3). https://www.knowledgenile.com/blogs/medical-diagnosis-with-machine-learning-advantages-and-limitations#:~:text=For%20Medical%20Diagnosis-,Helps%20In%20Identifying%20Diseases%20And%20Diagnosis,stages%20to%20other%20hereditary%20illnesses.
- Ursin, F., Timmermann, C., & Steger, F. (2021, March 4). Ethical implications of alzheimer’s disease prediction in asymptomatic individuals through artificial intelligence. MDPI. https://www.mdpi.com/2075-4418/11/3/440#:~:text=The%20ethical%20framework%20includes%20the,the%20opportunity%20for%20future%20planning.

Sources

The Black Mirror Writers' Room exercises was designed by Dr. Casey Fiesler. Links to her work and publications:

- The Black Mirror Writers Room: The Case (and Caution) for Ethical Speculation in CS Education
- "Run Wild a Little With Your Imagination": Ethical Speculation in Computing Education with Black Mirror

Python Review

In this section of the assignment, we will review useful Python functions and libraries that will allow you to read and analyze data, as well as training simple Machine Learning models.

Section 1: Exploring datasets with Pandas

First, we will need a dataset to work on. Let's use a weather type dataset, a good starting dataset (we will save more interesting cases for later!). Download this dataset from the link to use it for this exercise.

We also need to import the necessary library to read and manipulate our dataset, which is Pandas. The imports are given to you. Next, use the `read_csv()` function to import the data in your workspace. The documentation for this function can be found here.

```
import pandas as pd

weather_classification_data =
pd.read_csv("weather_classification_data.csv")
weather_classification_data.head()
```

```
   Temperature  Humidity  Wind Speed  Precipitation (%)    Cloud Cover
\
0         14.0        73         9.5               82.0  partly cloudy

1         39.0        96         8.5               71.0  partly cloudy

2         30.0        64         7.0               16.0          clear

3         38.0        83         1.5               82.0          clear

4         27.0        74        17.0               66.0       overcast


   Atmospheric Pressure  UV Index  Season  Visibility (km)
Location  \
0               1010.82         2  Winter              3.5    inland

1               1011.43         7  Spring             10.0    inland

2               1018.72         5  Spring              5.5  mountain

3               1026.25         7  Spring              1.0   coastal

4                990.67         1  Winter              2.5  mountain


  Weather Type
0        Rainy
1       Cloudy
2        Sunny
3        Sunny
4        Rainy
```

Now, let's use the `describe()` function of the Pandas library to get an overview of the dataset, and answer the following questions (you can write your answers in this box):

- What is the maximum temperature recorded in the dataset? *109 °C*
- What is the average wind speed? *9.832197 km/h*

Note: some of the values you will see may appear unrealistic (such as incredibly high temperatures). The dataset is artificially generated and purposefully includes outliers to practice detection and handling, but it is not something we will worry about it this exercise - we are just interested in getting some practice with useful commands.

```
# YOUR ANSWER HERE
weather_classification_data.describe()

        Temperature      Humidity    Wind Speed  Precipitation (%)  \
count  13200.000000  13200.000000  13200.000000       13200.000000
mean      19.127576     68.710833      9.832197          53.644394
std       17.386327     20.194248      6.908704          31.946541
min      -25.000000     20.000000      0.000000           0.000000
25%        4.000000     57.000000      5.000000          19.000000
50%       21.000000     70.000000      9.000000          58.000000
75%       31.000000     84.000000     13.500000          82.000000
max      109.000000    109.000000     48.500000         109.000000

       Atmospheric Pressure      UV Index  Visibility (km)
count          13200.000000  13200.000000     13200.000000
mean            1005.827896      4.005758         5.462917
std               37.199589      3.856600         3.371499
min              800.120000      0.000000         0.000000
25%              994.800000      1.000000         3.000000
50%             1007.650000      3.000000         5.000000
75%             1016.772500      7.000000         7.500000
max             1199.210000     14.000000        20.000000
```

The `describe()` function is helpful, but it does not answer all the questions we may have. For example, we did not get any idea about the class distribution in our dataset, that is, how many samples we have for each of the four classes (Rainy, Cloudy, Sunny, Snowy). Can you write a line of code to answer this question?

```
# YOUR ANSWER HERE
weather_classification_data["Weather Type"].value_counts()

Weather Type
Rainy     3300
Cloudy    3300
Sunny     3300
Snowy     3300
Name: count, dtype: int64
```

Thanks to `describe()`, we know that the minimum temperature recorded is -25 C, but we have no idea which sample it belongs to. Can you write a line of code to find the sample number and also the Weather Type associated to it?

```
# YOUR ANSWER HERE
weather_classification_data.loc[weather_classification_data['Temperatu
re'] ==

weather_classification_data['Temperature'].min()]
```

```
      Temperature  Humidity  Wind Speed  Precipitation (%) Cloud Cover
\
4609        -25.0       105        29.0              106.0    overcast


      Atmospheric Pressure  UV Index  Season  Visibility (km)
Location  \
4609                980.86         1  Winter              2.5
mountain

     Weather Type
4609        Snowy
```

The sample number is 4609, with `Weather Type == "Snowy"`.

Again thanks to `describe()`, we know that 25% of the samples in the dataset have a recorded Precipitation higher that 82 (you can verify this in the output table), but how many of these are Snowy? Answer this question in 1 line of code.

```
# YOUR ANSWER HERE
print(weather_classification_data.loc[(weather_classification_data["Pr
ecipitation (%)"] > 82) & (weather_classification_data['Weather Type']
== "Snowy")].shape)
```

```
(1243, 11)
```

Among all samples with `Precipitation (%) > 82`, 1243 of them have `Weather Type == "Snowy"`.

Finally, sometimes we may be interested in sorting the dataframe by the values in a column. In this cell, sort the dataframe by humidity in descending order, and check the results by printing the first 5 rows.

```
# YOUR ANSWER HERE
weather_data_by_humidity = weather_classification_data.sort_values(by
= "Humidity", ascending = False)
weather_data_by_humidity.head()
```

```
       Temperature  Humidity  Wind Speed  Precipitation (%)    Cloud
Cover  \
1303          29.0       109        21.0               93.0  partly
cloudy
8716          16.0       109        27.0              102.0
overcast
```

```
9707          51.0       109        17.0               98.0
overcast
2812          16.0       109        39.0               87.0  partly
cloudy
12566          4.0       109        16.0               93.0
overcast

       Atmospheric Pressure  UV Index  Season  Visibility (km)
Location  \
1303                1018.98         9  Winter              7.5
inland
8716                1007.30         1  Winter             11.5
inland
9707                 994.03         8  Spring              5.5
coastal
2812                1011.38        11  Spring              2.0
inland
12566                988.15        12  Winter              3.5
inland

      Weather Type
1303        Cloudy
8716        Cloudy
9707         Rainy
2812         Rainy
12566        Snowy
```

As last step of this section, save the sorted dataframe in a new csv file called "weather_data_by_humidity.csv"

```
# YOUR ANSWER HERE
weather_data_by_humidity.to_csv('weather_data_by_humidity.csv',
index=False)
```

Section 2: Training ML models with Scikit-learn

We are now interested in creating a model to predict the weather type based on the features available. Let's see how to do that using the python library Scikit-learn, while reviewing some important concepts about training and evaluating models. Simply run the cells below to see the output and answer the related questions.

First, we need to split our data set into training and testing set. The next cell shows how to do that. We will also separate the Weather Type column (target) from the other columns (features)

```
from sklearn.model_selection import train_test_split

train_df, test_df = train_test_split(weather_classification_data,
test_size=0.2, random_state=123)  # 80%-20% train test split on df
```

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

X_train.head()  # quick visual check on X_train, the features
dataframe

       Temperature  Humidity  Wind Speed  Precipitation (%)    Cloud
Cover  \
12987         26.0        45         3.5               10.0
clear
905           29.0        71        21.0               86.0  partly
cloudy
5590          38.0        63         5.5               11.0
clear
7269          17.0        66        18.0               63.0  partly
cloudy
1417          32.0        39         7.5                3.0
clear

       Atmospheric Pressure  UV Index  Season  Visibility (km)
Location
12987               1011.01         7  Autumn              5.0
inland
905                 1013.77        12  Winter              6.5
inland
5590                1013.87        11  Spring              7.5
mountain
7269                 992.22         1  Winter              2.5
inland
1417                1021.43         9  Autumn              5.5
inland
```

Question for you: creating a testing set is very important when training a model. **Why? How is it used? What would happen if we did not do this very important step?**

- After training the model on the training set, we need to evaluate and assess the performance of the model on an unseen dataset. Since we fit the model on the training set, it is inappropriate to evaluate the model performance on the training set again because the model has seen the datatset before. The testing set gives the model unseen data and lets the model perform on this new data, allowing for a more accurate assessment of model performance.
- After the original dataset is split into two sets and the model is fitted, we will use the model to predict the targets of observations in the testing set given their features in `X_test`. These predicted values can be used to compare with the true targets from the test set in `y_test`, along with potential application of formulas or metrics, to evaluate the performance of this model.

- Without a testing set, we will not be able to properly evaluate the performance of the model under new deployment data. We may overestimate the model performance, as the model will often perform well on the dataset that was trained on, but it might overfit onto the training data, causing it perform poorly on unseen datasets in the future.

Refined answer:

- After training the model on the training set, we need to evaluate and assess the performance of the model on an unseen dataset. Since we fit the model on the training set, it is inappropriate to evaluate the model performance on the training set again because the model has seen the datatset before. The testing set gives the model unseen data and lets the model perform on this new data, allowing for a more accurate assessment of model performance.
- After the original dataset is split into two sets and the model is fitted, we will use the model to predict the targets of observations in the testing set given their features in `X_test`. These predicted values can be used to compare with the true targets from the test set in `y_test`, along with potential application of formulas or metrics, to evaluate the performance of this model. More importantly, we can only use the testing set once to evaluate the final model performance at the last step after fitting the model. If we use the testing set multiple times, especially if we choose to refine the model in response to the given model performance, we violate the "Golden Rule" of machine learning (The testing set cannot influence training the model in any way) and cause data leakage, effectively making it a glorified validation set.
- Without testing sets, we will not be able to properly evaluate the performance of each model under new unseen deployment data in the real world compared to other similar models with modified modelling decisions within their pipelines. The test set allows for unbiased comparisons of model performance to evaluate which hyperparameters are most optimal for generalising to the overall population.

As you can see, the dataset includes categorical features. Most classifiers require categorical features to be transformed before they can be used for training and prediction. The code below uses One Hot Encoding to convert the categorical features Cloud Cover, Season and Location, while leaving the numberical features unchanged.

This is a simple example of data preprocessing. Preprocessing can be more extensive (for example, including scaling of numerical features), but we are only interested in an overview of the fundamentals, so we will just apply One Hot Encoding to make the data usable.

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer

passthrough = ['Temperature', 'Humidity', 'Wind Speed', 'Precipitation
(%)',
               'Atmospheric Pressure', 'UV Index', 'Visibility (km)']
categorical = ['Cloud Cover', 'Season', 'Location']


ct = make_column_transformer(
    (OneHotEncoder(), categorical),  # OHE on categorical features
    ("passthrough", passthrough),  # no transformations on the
numberical features
)

# Fit the encoder on the training data and transform
train_encoded = ct.fit_transform(X_train)

# Transform the test data
test_encoded = ct.transform(X_test)

# Convert the encoded data back to DataFrame for better readability

column_names = (

ct.named_transformers_["onehotencoder"].get_feature_names_out().tolist
() + passthrough
)

X_train_encoded = pd.DataFrame(train_encoded, columns=column_names)
X_test_encoded = pd.DataFrame(test_encoded, columns=column_names)

# Run this cell to see what the encoded data set looks like

X_train_encoded
```

```
       Cloud Cover_clear  Cloud Cover_cloudy  Cloud Cover_overcast  \
0                    1.0                 0.0                   0.0
1                    0.0                 0.0                   0.0
2                    1.0                 0.0                   0.0
3                    0.0                 0.0                   0.0
4                    1.0                 0.0                   0.0
...                  ...                 ...                   ...
10555                0.0                 0.0                   0.0
10556                0.0                 0.0                   1.0
10557                1.0                 0.0                   0.0
10558                1.0                 0.0                   0.0
10559                0.0                 0.0                   0.0

       Cloud Cover_partly cloudy  Season_Autumn  Season_Spring
Season_Summer  \
0                            0.0            1.0            0.0
0.0
1                            1.0            0.0            0.0
0.0
2                            0.0            0.0            1.0
0.0
3                            1.0            0.0            0.0
0.0
4                            0.0            1.0            0.0
0.0
...                          ...            ...            ...
```

```
...
10555                        1.0            0.0            1.0
0.0
10556                        0.0            0.0            0.0
1.0
10557                        0.0            0.0            0.0
1.0
10558                        0.0            0.0            0.0
1.0
10559                        1.0            0.0            0.0
1.0

       Season_Winter  Location_coastal  Location_inland
Location_mountain  \
0                0.0               0.0              1.0
0.0
1                1.0               0.0              1.0
0.0
2                0.0               0.0              0.0
1.0
3                1.0               0.0              1.0
0.0
4                0.0               0.0              1.0
0.0
...              ...               ...              ...
...
10555            0.0               0.0              1.0
0.0
10556            0.0               0.0              0.0
1.0
10557            0.0               0.0              1.0
0.0
10558            0.0               1.0              0.0
0.0
10559            0.0               0.0              0.0
1.0

       Temperature  Humidity  Wind Speed  Precipitation (%)  \
0             26.0      45.0         3.5               10.0
1             29.0      71.0        21.0               86.0
2             38.0      63.0         5.5               11.0
3             17.0      66.0        18.0               63.0
4             32.0      39.0         7.5                3.0
...            ...       ...         ...                ...
10555        -19.0      27.0         5.5               66.0
10556         27.0      63.0         8.0               73.0
10557         25.0      23.0         4.5               13.0
10558         38.0      39.0         9.0               10.0
10559         11.0      61.0         9.5               41.0
```

```
       Atmospheric Pressure  UV Index  Visibility (km)
0                   1011.01       7.0              5.0
1                   1013.77      12.0              6.5
2                   1013.87      11.0              7.5
3                    992.22       1.0              2.5
4                   1021.43       9.0              5.5
...                     ...       ...              ...
10555                929.72       9.0             14.5
10556               1016.64       2.0              1.0
10557               1014.45      11.0              6.5
10558               1018.10      10.0              7.0
10559               1008.30       3.0              6.5

[10560 rows x 18 columns]
```

Question for you: It appears that we applied the same One Hot Encoding transformation to both training and test set. Why did we bother doing this operation on the separate sets? Could have we just transformed the original dataframe `df`, and then split it in training and test set?

From the coding block above, we can see that we fit the encoder on the training data and transform at the same time by using the `ct.fit_transform()` function. It is possible to apply One Hot Encoding to the entire model then apply the split, but the encoding can access information from the testing data and thus may result in a different data transformation on the original dataframe. However, we want the testing set to remain unseen to evaluate the model performance more accurately (e.g. New `Cloud Cover` category that is in testing data but not training data, leads to different categorisation for new category). Therefore, we should avoid transforming the original dataframe first.

There are many classifiers we can choose from. We will use Decision Trees to start. Decision trees are very simple classification algorithms, although they have typically mediocre performance on complex classification problems.

A certain level of familiarity with Decision Trees is expected in this course. You may want to review the material from your previous courses, or this introduction.

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt


model = DecisionTreeClassifier(random_state=123) # Create a decision tree
model.fit(X_train_encoded, y_train) # Fit a decision tree
```

```
DecisionTreeClassifier(random_state=123)
```

Now that we have the tree, we want to see how well it performs. Let's first check the accuracy on the training set:

```
model.score(X_train_encoded, y_train) # Score the decision tree
```

```
1.0
```

100% accuracy!!!

...

This sounds too good to be true... let's check the test set to see how well the trees perform on unseen samples:

```
model.score(X_test_encoded, y_test) # Score the decision tree on test
set
```

```
0.9151515151515152
```

Accuracy dropped significantly when we moved to unseen samples!

This is because the Decision Tree, if left unsupervised, is very prone to **overfitting**.

Question for you: what does it mean for a model to overfit?

A model is overfitting when it fits onto the training set too well. In other words, it might understand and follow the patterns of the training set too closely, likely causing it to perform poorly with unseen data (e.g. testing set). The model is too specific for the training set, and thus not generalised enough for unseen data. It learns a mapping function that is very specific and sensitive to the training data and may even capture some quirks from the data.

Refined answer:

A model is overfitting when the model learns a mapping function that is too sensitive to the training data, and may even capture some random noise, quirks, and irrelevant features from the training data, which cannot generalize well on new examples, likely causing the model to perform very well on the training data, but poorly with unseen data (e.g. testing set, deployment data).

To prevent a model from overfitting, we tune its **hyperparameters.** Hyperparameters are like knobs that we can use to regulate the way a model learn.

Some hyperparameters for the scikit-learn DecisionTreeClassifier include:

- max_depth: the maximum distance between the root node and a leaf node
- min_samples_split: the minimum number of samples required to split an internal node
- min_samples_leaf; the minimum number of samples required to be at a leaf node

You can look up other hyperparameters and their default values in the DecisionTreeClassifier documentation. By default, the maximum depth value is set to *None*, that is, the tree is free to grow until it has parfectly classified all samples. As we have seen, this results in perfect accuracy on the training set, but much lower accuracy on unseen samples.

Run the cell below to see the depth of our overfitted tree:

```
model.get_depth()
```

```
19
```

If we could find the right depth for our tree, we could reduce the problem of overfitting.

Question for you: what would happen if we reduce the depth of the tree *too much*? What do you expect the accuracy on training and test set to look like in this case?

Reducing the depth of the tree too much will cause the model to underfit the data, not capturing enough patterns from the features provided for effective predictions when training the model. As a consequence, the predictions are likely to be too random to be useful, and the accuracy score on both the training and testing sets will be very low.

Refined answer:

Reducing the depth of the tree too much will cause the model to underfit the data, not capturing enough patterns from the features provided for effective predictions when training the model. As a consequence, the predictions are likely to be too random to be useful, and the accuracy score on both the training and testing sets will be very low. In addition, the training accuracy and testing accuracy are very close to each other; the difference between the training and testing scores becomes very small.

Hyperparameter tuning is typically done on a **validation set.** A validation set is a set of samples not used for training, like the test set, but unlike the test set, we are allowed to use this multiple times as we look for the best hyperparameter values.

Because our data set is rather small, it is not great to take more samples from the training set to create a validation set, because:

- We would have fewer samples (less information) to train our model
- The validation set would also be small, and result in a highly variable accuracy measure (meaning if we run the experiment again changing the samples in each set, we will likely get very different results)

There is a method that we can use to eliminate both problems, called ***k*-fold cross-validation.** Cross-validation iteratively separates training and validation set (*k* times), so we get multiple measures of accuracy on the validation sets, which can be averaged for a more stable result. A good understanding of how cross-validation works is important for any data scientist. I encourage you to review cross-validation from previous courses, or this introduction video (courtesy of Dr. Kolhatkar).

Scikit-learn has a great method that we can use to perform cross-validation and find the best hyperparameters for a model at the same time, called GridSearchCV. Let's use it to find the best depth for our Decision Tree:

```
from sklearn.model_selection import GridSearchCV
import numpy as np  # to create the array of values for depth

param_grid = {
    "max_depth": np.arange(1, 20, 1)  # testing all depths from 1 to
19
}

grid_search = GridSearchCV(
    model, param_grid, cv=10, n_jobs=-1, return_train_score=True
    # 10-fold cross-validation for all possible
    # depths
)
grid_search.fit(X_train_encoded, y_train)
```

```
GridSearchCV(cv=10,
estimator=DecisionTreeClassifier(random_state=123),
             n_jobs=-1,
             param_grid={'max_depth': array([ 1,  2,  3,  4,  5,  6,
7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19])},
             return_train_score=True)
```

```
grid_search.best_score_
```

```
0.9116477272727271
```

```
grid_search.best_params_
```

```
{'max_depth': 11}
```

Complete the sentence (replace --?--): Among all possible trees, GridSearchCV picked a tree of depth **11**, with an average validation accuracy of **around 0.912**.

The accuracy on the training set is no longer 100%, but we expect this tree to perform better on unseen samples. Let's try it on our test set:

```
best_tree = grid_search.best_estimator_

best_tree.score(X_test_encoded, y_test) # Score the decision tree on
test set
```

```
0.9143939393939394
```

The accuracy is similar to when the model was overfitting, but hyperparameter tuning brought us 2 advantages:

- We had a more realistic expectation of what our accuracy was going to be (closer to 91%, not 100%)
- We simplified the model and reduced its depth. This makes the model faster and easier to visualize.

Question for you: on what samples (or portion of samples) of `X_train_encoded` was the final model (`best_tree`) trained on?

All samples of `X_train_encoded` were used to train the final model. After GridSearchCV performs cross-validation and finds the best hyperparameter `max_depth` with the highest average validation accuracy, it refits this best model `best_tree` with this best hyperparameter `max_depth` on the entire training dataset `X_train_encoded`, not just a proportion of it.

The model can now be used to get predictions for unseen samples. For example:

```
random_sample = X_test_encoded.sample(n=1, random_state=42)

random_sample

      Cloud Cover_clear  Cloud Cover_cloudy  Cloud Cover_overcast  \
2005                1.0                 0.0                   0.0

      Cloud Cover_partly cloudy  Season_Autumn  Season_Spring
Season_Summer  \
2005                        0.0            0.0            0.0
0.0

      Season_Winter  Location_coastal  Location_inland
Location_mountain  \
2005            1.0               0.0              0.0
1.0

      Temperature  Humidity  Wind Speed  Precipitation (%)  \
2005         33.0      67.0         5.5                5.0

      Atmospheric Pressure  UV Index  Visibility (km)
2005               1015.42      10.0              8.0

best_tree.predict(random_sample)

array(['Sunny'], dtype=object)
```

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 started on separate sections within the assignment; Nicholas started with section 1, while Jingyuan started with section 2. Afterwards, we collaborated on modifying the other section that we did not complete to refine our responses. Finally, we each responded to the last two questions on final thoughts separately.

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.

- Jingyuan's response: In particular, I used ChatGPT to gather some information on "One Hot Encoding transformation". This is a new terminology I have never heard about, and I asked ChatGPT to provide a brief introduction on how it works, its working mechanism (how the data is transformed and how the data is split), and some examples. Besides, I did not use any generative AI tools to help with the assignment but did make use of some of the material and lectures from CPSC330, which is another course I am taking right now.
- Nicholas' response: I used ChatGPT for more medical information to add to Question 3 step 3.

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!

- Jingyuan's response: I am still confused about the two formulations of the Right Perspective.
- Nicholas' response: I struggled with Question 3, particularly steps 3, 4, and 5, primarily due to the creative writing aspect and research for existing methods to supplement the arguments.