**Final Exam - DSC 80, Winter 2026**

Full Name:

PID:

Section: ◯ A (9:30AM) ◯ B (11:00AM)

---

**Instructions:**

- This exam consists of 13 questions, worth a total of 180 points.
- Questions 1, 2, 8, 9, and 10 are marked with an M. These questions test material from the Midterm Exam and will be used for the redemption opportunity.
- Write your PID in the top right corner of each page in the space provided.
- For code questions, a correct implementation is not necessarily sufficient for full credit; we will also consider aspects such as readability, simplicity, and efficiency.
- Please write **clearly** in the provided answer boxes; we will not grade work that appears elsewhere. Completely fill in bubbles and square boxes; if we cannot tell which option(s) you selected, you may lose points.

  ◯ A bubble means that you should only **select one choice**.

  ☐ A square box means you should **select all that apply**.
- You may use two pages of double-sided handwritten notes. Aside from this, you may not refer to any other resources or technology during the exam. No calculators!

---

By signing below, you are agreeing that you will behave honestly and fairly during and after this exam.

Signature:

# Version A

Please do not open your exam until instructed to do so.

**Important**: Before proceeding, make sure to **rip off the last two pages** of this exam packet and read the data description.

## Question 1 (M) - 14 pts

To start, we need to calculate patient wait times, which are not provided in our data. Suppose we execute the line of code below to add a `"WaitTime"` column to `med`.

```
med["WaitTime"] = (med["StartTime"] - med[["ArrivalTime",
                   "AppointmentTime"]].max(axis=1)).dt.seconds / 60
```

Note that when we subtract two `pd.Timestamp` objects, the result is a `pd.Timedelta` object, whose `.seconds` attribute gives the time difference in seconds. There is no way to access the time difference in minutes directly.

**a)** (4 pts) Fill in the blanks in the code below so that the `"WaitTime"` column remains exactly the same as calculated above. In other words, the code below should give an equivalent way to calculate `"WaitTime"`.

```
def wait_time(x):
    return __(a)__
med["WaitTime"] = med.apply(__(b)__)
```

(a):

(b):

**b)** (2 pts) What is the data type of the `"WaitTime"` column?

◯ `pd.Timestamp` ◯ `pd.Timedelta` ◯ `int` ◯ `float`

**c)** (5 pts) Determine the value of the following expression.

```
list(med["WaitTime"].iloc[:5])
```

**d)** (3 pts) What kind of values *can* appear in the `"WaitTime"` column? Select all that apply.

☐ Negative ☐ Zero ☐ Positive

Now that we've added a `"WaitTime"` column to `med`, we'll also add a `"Wait"` column containing `int` values, defined as follows.

```
med["Wait"] = (med["WaitTime"] > 0).astype(int)
```

For the rest of the exam, med has `"WaitTime"` and `"Wait"` columns.

## Question 2 (M) - 10 pts

Doctors love having letters after their names! These letters usually represent degrees, titles, or certifications. We'll refer to them collectively as credentials. In the `"Provider"` column of `med`, each provider has at least one credential. Credentials appear after the name and are separated by commas. For example, the preview of `med` shows that Dr. Takashi Hirase has two credentials (MD and MPH).

**a)** (5 pts) Write **one line** of code that evaluates to a Series containing the number of credentials for each provider in the `"Provider"` column of `med`. You **must** use `.split()` and you may **not** define any lambda functions.

**b)** (5 pts) Write a different **single line** of code that evaluates to the same Series. This time, you may **not** use `.split()` and you may **not** define any lambda functions.

Finally, we'll add this Series to `med` as a new column called `"Credentials"`. This column is included in `med` for the rest of the exam.

## Question 3 - 16 pts

Consider the small subset of `med` shown in full at right. Recall that the `"Wait"` column was added after Question 1. The data is sorted by `"Age"`.

| "Age" | "NumProviders" | "Wait" |
|---|---|---|
| 6 | 6 | 0 |
| 11 | 3 | 0 |
| 17 | 3 | 1 |
| 19 | 6 | 1 |
| 23 | 2 | 0 |
| 31 | 4 | 1 |
| 38 | 4 | 1 |
| 44 | 2 | 1 |
| 44 | 3 | 0 |
| 60 | 5 | 1 |
| 60 | 5 | 0 |
| 79 | 3 | 0 |

**a)** (4 pts) If we train a decision tree on this data to predict `"Wait"` based on `"Age"` and `"NumProviders"`, what is the **maximum possible accuracy** the decision tree could achieve? Give your answer as an exact decimal or simplified fraction.

14/12

**b)** (4 pts) Select the expression below that gives the weighted entropy associated with using `"Age" >= 15` as the root node of the decision tree.

- ○ $\frac{1}{6}\left(-\frac{2}{5}\log_2\left(\frac{2}{5}\right)-\frac{3}{5}\log_2\left(\frac{3}{5}\right)\right)$
- ○ $-\frac{2}{5}\log_2\left(\frac{2}{5}\right)-\frac{3}{5}\log_2\left(\frac{3}{5}\right)$
- ○ $\frac{1}{6}\left(-\frac{1}{2}\log_2\left(\frac{1}{2}\right)-\frac{1}{2}\log_2\left(\frac{1}{2}\right)\right)$
- ○ $-\frac{1}{2}\log_2\left(\frac{1}{2}\right)-\frac{1}{2}\log_2\left(\frac{1}{2}\right)$
- ● $\frac{5}{6}\left(-\frac{2}{5}\log_2\left(\frac{2}{5}\right)-\frac{3}{5}\log_2\left(\frac{3}{5}\right)\right)$

**c)** (4 pts) All but one of the following questions splits the data in such a way that the weighted entropy is the same. Which question yields a **different weighted entropy** than the others?

- ○ `"NumProviders" <= 2`
- ● `"NumProviders" <= 3`
- ○ `"NumProviders" <= 4`
- ○ `"NumProviders" <= 5`
- ○ `"NumProviders" <= 6`

**d)** (4 pts) What is the weighted entropy associated with any one of the questions you did **not** pick in part (c)? Give your answer as an exact decimal or simplified fraction.

1

PID: _____

## Question 4 - 13 pts

`sklearn` is considering adding a new hyperparameter to its `DecisionTreeClassifier` class. The new hyperparameter, `min_entropy`, is used to determine when a node should be split. A node will be split when its entropy is greater than or equal to `min_entropy`. Otherwise, the node will be a leaf node.

Suppose we create training and testing datasets as follows.

```
X = med.drop(columns=["Wait"])
y = med["Wait"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

**a)** (5 pts) The function below selects a value for `min_entropy` based on an input list of candidate values.

```
def find_min_entropy(candidates):
    highest_score = -1
    out = -1
    for min_e in candidates:
        dt = DecisionTreeClassifier(min_entropy=min_e)
        dt.fit(X_train, y_train)
        if dt.score(X_train, y_train) >= highest_score:
            highest_score = dt.score(X_train, y_train)
            out = min_e
    return out
```

What should the function return on an input list of `[0, 0.2, 0.4, 0.6, 0.8, 1]`?

0

**b)** (4 pts) **Circle one word in each box:** If we train a decision tree with the value selected in part (a) for `min_entropy`, the test accuracy will likely be higher / lower than the train accuracy due to underfitting / overfitting.

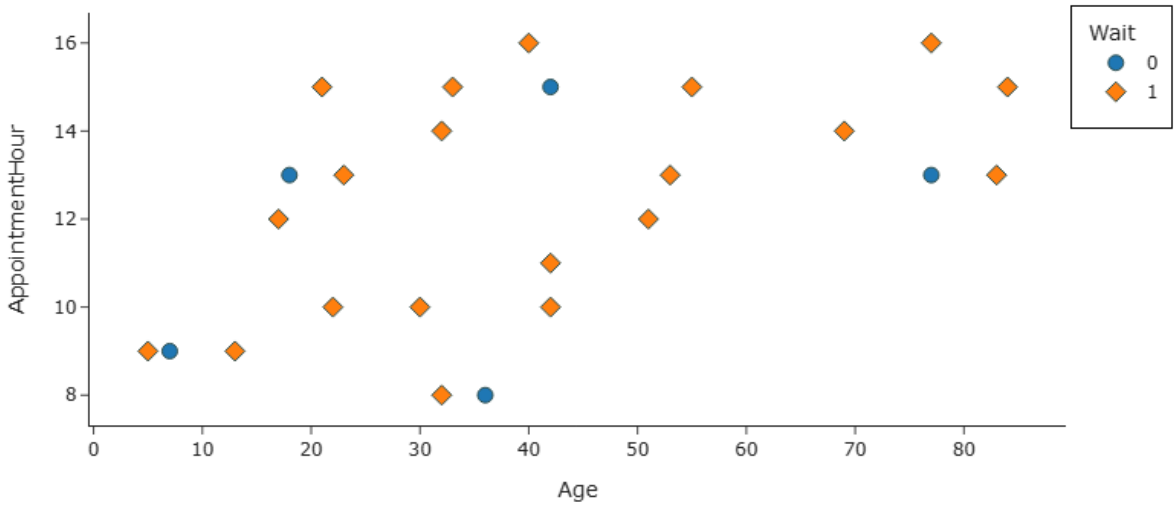
**c)** (4 pts) **Circle one word in each box:** In general, increasing the value of `min_entropy` increases / decreases bias and increases / decreases variance.

## Question 5 - 14 pts

In Lab 9, you learned about $k$-nearest neighbors regression. A related machine learning algorithm is $k$-nearest neighbors classification, in which predictions are made by finding the $k$ points in the training data that are nearest to the point we are trying to classify. We predict the class that the **majority** of those $k$ points belong to (similar to the way in which decision trees in a random forest vote on a prediction). In this problem, we'll use the standard Euclidean ($L_2$) distance to measure the distance between points.

In this problem, we'll try to predict `"Wait"` based on `"Age"` and `"AppointmentHour"`, where `"AppointmentHour"` is the hour from the `"AppointmentTime"` column.

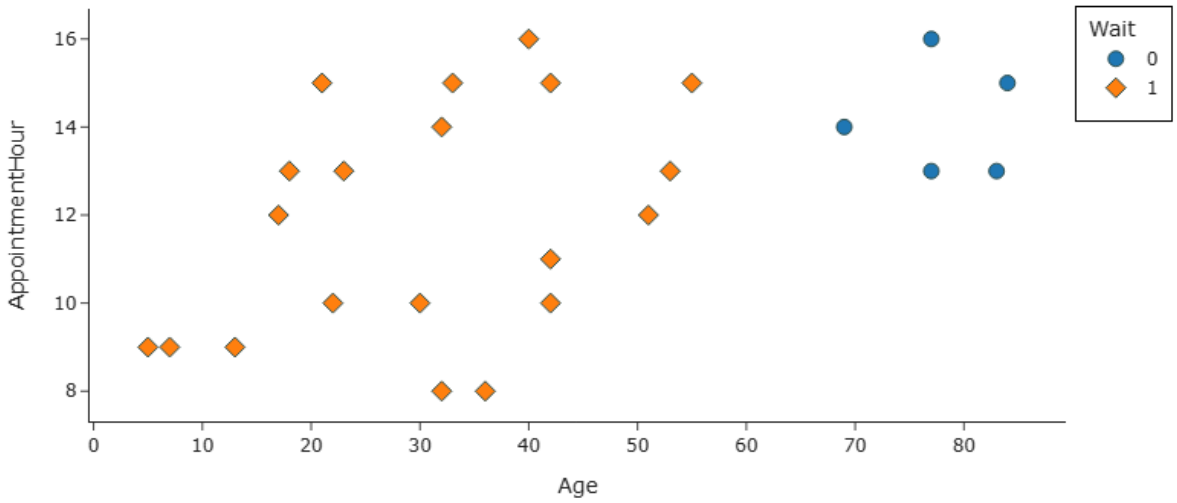
**a)** (6 pts) Suppose the training data consists only of the 25 points shown below. Determine the accuracy, precision, and recall of a 3-nearest neighbors classifier on this data.




Accuracy: ____ Precision: ____ Recall: ____

PID: _____

**b)** (6 pts) Now suppose the training data consists only of the 25 points shown below. Determine the accuracy, precision, and recall of a 3-nearest neighbors classifier on this data.




Accuracy: | Precision: | Recall:

**c)** (2 pts) Finally, consider a 15-nearest neighbor classifier, which has the same accuracy on both datasets. What is that accuracy?

## Question 6 - 11 pts

We want to use linear regression to predict `"WaitTime"` based on

- `"NumProviders"`,
- `"Credentials"` (from Question 2),
- `"Department"`,
- and whether `"AppointmentTime"` is in the morning (before 12:00) or afternoon (12:00 or later).

We want to ensure that the coefficients are **interpretable and can be used to determine the most impactful single feature in the model's predictions**.

Fill in the code below to fit an appropriate Pipeline to the data in `med`, which we will think of as our training data for this problem.

```
def hour(df):
    df.iloc[:, 0] = df.iloc[:, 0].dt.hour
    return df

X = med[["NumProviders", "Credentials", "Department", "AppointmentTime"]]
y = med["WaitTime"]

pl = [          ]

pl.fit(X, y)
```

There is only one blank in the code above, which should be filled with a capital letter corresponding to one of the answer choice options given on the next page. This answer choice will have blanks of its own, which you should also fill in. Every time you use an answer choice, fill in the blanks in that answer choice with one of the following:

- a capital letter corresponding to an answer choice option
- a string, or
- an integer.

Some answer choices will be unused. You should leave any blanks in those answer choices empty.

**Suggestion**: The best way to go about this problem is to first write code to define the Pipeline `pl`. Then try to match your code to the answer choices given. Drawing a picture of your Pipeline is also useful.

PID: _____

A. `drop = "first"`

B. `remainder = "drop"`

C. `remainder = "passthrough"`

D. `PolynomialFeatures(` [ ] `)`

E. `StandardScaler()`

F. `Binarizer(threshold =` [ ] `)`

G. `CountVectorizer()`

H. `FunctionTransformer(hour)`

I. `OneHotEncoder(` [ ] `)`

J. `LinearRegression()`

K. `ColumnTransformer([("one",` [ ] `, [` [ ] `]),`

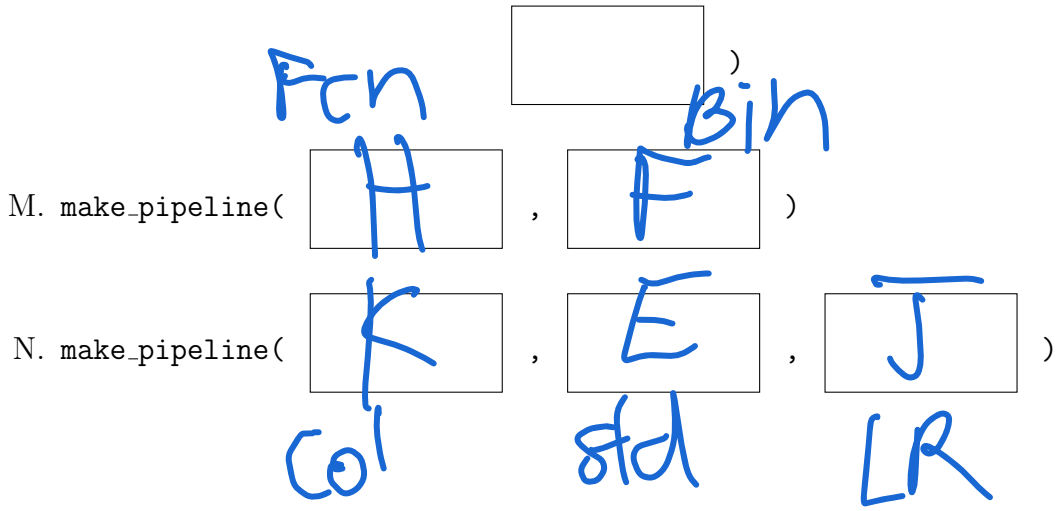
`("two",` [ ] `, [` [ ] `])],

[ ] `)`

L. `ColumnTransformer([("one",` [ ] `, [` [ ] `]),`

`("two",` [ ] `, [` [ ] `]),`

`("three",` [ ] `, [` [ ] `])],


)
M. make_pipeline(
,
)
N. make_pipeline(
,
,
)

PID: _____

## Question 7 - 15 pts

Suppose we derive a numerical feature `"AppointmentTimeSeconds"` which measures the `"AppointmentTime"` in seconds since midnight. Then we use linear regression to fit a prediction rule of the form:

predicted `"WaitTime"` $= w_0 + w_1 \cdot$ `"Age"` $+ w_2 \cdot$ `"NumProviders"` $+ w_3 \cdot$ `"AppointmentTimeSeconds"`

Consider each of the following changes to the model above, and determine which coefficients in the fit model may change. **Select all** coefficients that may change. Note that we are changing the original model each time, not stacking changes on top of one another.

**a)** (4 pts) Change `"AppointmentTimeSeconds"` to `"AppointmentTimeMinutes"`, which is measured in minutes since midnight.

☐ $w_0$ ☐ $w_1$ ☐ $w_2$ ☒ $w_3$

**b)** (3 pts) Remove the intercept term $w_0$.

☒ $w_1$ ☒ $w_2$ ☒ $w_3$

**c)** (4 pts) Add a new feature, which is $3 \cdot$ `"Age"` $+$ `"NumProviders"`.

☐ $w_0$ ☒ $w_1$ ☒ $w_2$ ☐ $w_3$

**d)** (4 pts) Add a new feature, which is `"Age"`/`"NumProviders"`.

☒ $w_0$ ☒ $w_1$ ☒ $w_2$ ☒ $w_3$

## Question 8 (M) - 15 pts

**a)** (6 pts) We suspect that some `"Provider"`s have longer `"WaitTime"`s than others. Fill in the blanks below to add a column to `med` called `"EstimatedWaitTime"` which contains the median `"WaitTime"` for appointments with the same `"Provider"`.

```
med["EstimatedWaitTime"] = (med.groupby(__(a)__)[__(b)__]
                               .__(c)__(__(d)__))
```

(a): _____ (b): _____

(c): _____ (d): _____

**b)** (9 pts) We suspect that some `"Department"`s are frequently running behind schedule and may occasionally have very high wait times. Fill in the blanks below so the result is a Series, indexed by `"Department"`, containing the 95th percentile of `"WaitTime"` for each `"Department"` in which at least 75 percent of appointments have a `"Wait"`. If less than 75 percent of appointments in a given `"Department"` have a `"Wait"`, the `"Department"` should not appear in the Series. Recall that `np.percentile(x, 95)` calculates the 95th percentile of `x`.

```
(med.groupby(__(a)__).__(b)__(__(c)__)
    .groupby(__(d)__)[__(e)__].__(f)__(__(g)__))
```

(a): _____

(b): _____

(c): _____

(d): _____

(e): _____

(f): _____

(g): _____

PID: _____

## Question 9 (M) - 16 pts

Suppose we have access to another DataFrame that contains billing information. The rows are the same as in `med`, but there are only three columns, `"MRN"`, `"AppointmentTime"`, and `"Billed"`. The `"Billed"` column contains the amount that the patient was billed for their medical services at the time of the appointment.

**a)** (4 pts) Suppose patients are billed for services at the time of their appointment, unless the services are very complex (such as a surgical procedure). For complex procedures, the `"Billed"` column is left empty, and patients are charged for services at a later date. In this scenario, what is the most likely missingness mechanism of the `"Billed"` column?

◯ missing by design (MD)
◯ missing not at random (MNAR)
◯ missing at random (MAR)
◯ missing completely at random (MCAR)

**b)** (4 pts) Now suppose we merge the billing DataFrame with `med` on `"MRN"` and `"AppointmentTime"`. We want to do a permutation test at the 0.05 significance level to decide if the missingness mechanism of the `"Billed"` column is more likely MCAR or MAR dependent on `"Department"`. Which of the following test statistics could be used for this permutation test? Select all that apply.

☐ difference of means
☐ absolute difference of means
☐ total variation distance (TVD)
☐ K-S statistic
☐ none of these

**c)** (4 pts) Suppose the p-value comes out to 0.03. What can we conclude? Select all that apply.

☐ The missingness mechanism is more likely MCAR than MAR.
☐ The missingness mechanism is more likely MAR than MCAR.
☐ The missingness mechanism is not MD.
☐ The missingness mechanism is not MNAR.
☐ None of the above is a valid conclusion.

**d)** (4 pts) Suppose additionally that on March 1, 2026, UC San Diego Health experienced a technical outage and all the billing data for that day was lost. Which imputation strategy is most appropriate if we want to make sure the mean and standard deviation don't change much as a result of the imputation?

◯ mean imputation
◯ probabilistic imputation
◯ mean imputation, conditional on `"Department"`
◯ probabilistic imputation, conditional on `"Department"`

## Question 10 (M) - 12 pts

Dr. Zheng and Dr. Golder are two medical doctors at UC San Diego Health. They each create a DataFrame of patients they have seen in the last year. Suppose that these DataFrames are called `dr_z` and `dr_g` and that each DataFrame includes a `"MRN"` column, which uniquely identifies patients.

Consider each of the following scenarios describing the overlap of `dr_z` and `dr_g`, and in each scenario, determine the number of rows in the DataFrame created by merging `dr_z` with `dr_g` using inner, outer, left, and right joins.

```
dr_z.merge(dr_g, on="MRN", how = ???)
```

**a)** (4 pts)

- `dr_z` has 100 rows, all representing distinct patients.
- `dr_g` has 80 rows, all representing distinct patients.
- 20 patients appear in both DataFrames.

`how = "inner"`: ☐ `how = "outer"`: ☐

`how = "left"`: ☐ `how = "right"`: ☐

**b)** (4 pts)

- `dr_z` has 50 rows, all representing distinct patients.
- `dr_g` has 15 rows, all representing distinct patients.
- All patients appearing in `dr_g` also appear in `dr_z`.

`how = "inner"`: ☐ `how = "outer"`: ☐

`how = "left"`: ☐ `how = "right"`: ☐

**c)** (4 pts)

- `dr_z` has 60 rows, representing 30 patients each appearing twice.
- `dr_g` has 80 rows, representing 40 patients each appearing twice.
- There are 10 patients that appear in both `dr_z` and `dr_g`.

`how = "inner"`: ☐ `how = "outer"`: ☐

`how = "left"`: ☐ `how = "right"`: ☐

PID: _____

## Question 11 - 12 pts

Suppose we train a unigram, bigram, and trigram model on the following corpus.

```
corpus = "Patient is ill. Patient is in pain. Ill patient will recover
          in time."
```

We tokenized the corpus as follows before training our models.

```
corpus.lower().split()
```

Now, we'd like to determine the probability of generating the sentence below, according to each model.

```
"Patient is in time."
```

For each part, give your answer as a simplified fraction (preferred) or a product of simplified fractions.

**a)** (4 pts) Determine the probability of the sentence above, according to the unigram model.

**b)** (4 pts) Determine the probability of the sentence above, according to the bigram model.

**c)** (4 pts) Determine the probability of the sentence above, according to the trigram model.

## Question 12 - 20 pts

On the website for UC San Diego Health, each provider has their own page. We've scraped the HTML from one provider's web page, which you can find at the end of this exam. We then instantiated a `BeautifulSoup` object, `soup`, from this HTML.

**a)** (3 pts) Consider the DOM tree for this document. How many children does the root node have?

◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ none of these

**b)** (3 pts) What does the following line of code evaluate to?

```
len(soup.find_all("script"))
```

◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ none of these

**c)** (6 pts) The latitude and longitude for the provider's office location are included in the document. Write one line of code that uses `soup.find()` (**not** `soup.find_all()`) to extract the latitude from `soup`, as a string (`"32.875663"`).

**d)** (4 pts) You'll notice that the HTML includes some JSON-formatted data. Locate the JSON object with keys `"@context"` and `"@graph"`. Fill in the blank in the code below to read this JSON object in as a Python dictionary, `dr_m`.

```
dr_m_string = ___________
dr_m = json.loads(dr_m_string)
dr_m
```

**e)** (4 pts) Write one line of code that extracts the street address from `dr_m`, as a string.

PID: _____

## Question 13 - 12 pts

The function `re.match(pat, s)` checks for the regular expression `pat` only at the beginning of string `s`. For example, `re.match("o", "hello")` does not find a match, but `re.match("h", "hello")` does.

**a)** (6 pts) The string `"UC San Diego Health"` has exactly two lowercase `a`'s. Write a regular expression pattern, `pat`, so that `re.match(pat, s)` finds a match if and only if `s` has exactly two lowercase `a`'s. **Write clearly!**

**b)** (6 pts) ICD-10-CM codes (International Classification of Diseases, Tenth Revision, Clinical Modification) are codes used in the medical field to classify diagnoses, symptoms, and causes of death. Below are a few examples of ICD-10-CM codes and their associated meanings:

- `J45.909`: Unspecified asthma, uncomplicated
- `F32.9`: Major depressive disorder, single episode, unspecified
- `G20`: Parkinson's disease
- `H43.391`: Eye floaters, right eye
- `S80.01XS`: Contusion of right knee, initial encounter

ICD-10-CM codes consist of 3 to 8 characters following a certain format:

- Codes begin with a capital letter followed by two digits, which together specify the broad category of the medical condition.
- This may be followed by a decimal point and additional capital letters and numbers, which give more specific details about the medical condition.

Write a regular expression pattern, `pat`, so that `re.match(pat, s)` finds a match if and only if `s` is formatted like an ICD-10-CM code.

The following are some examples of incorrectly formatted codes that should not be matched.

- `F27.`
- `TX3.120`
- `M27.56829`
- `L220.9`

**Write clearly!**

Dear students,

I so enjoyed teaching you and learning with you this quarter. I had a blast! While your enthusiasm may not be quite as strong, I hope you found something motivating and exciting in this course. Congratulations on completing this milestone and leveling up as a data scientist!

Feel free to leave a picture or note below, if you'd like.

**Before turning in your exam, please make sure that your PID is on every page.**