[71]:
```
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

new_review_embedding = embedding_model.encode(["It works!!!"])
new_review_embedding = new_review_embedding.reshape(1, -1)
similarities = cosine_similarity(new_review_embedding, embeddings)
most_similar_indices = similarities.argsort()[0][-5:][::-1]
most_similar_reviews = X.iloc[most_similar_indices]['review'].tolist()

for review in most_similar_reviews:
    print(review)
```

```
"Worked for Me !!!!"
"DOESN&#039;T WORK! Only gave me the worst night of my life. I had the worst cramps ever! It felt like labor an
d theres no way you can lay/ bend to fix the pain. I was throwing up while  trying to go to the bathroom. Could
n&#039;t go to the bathroom, couldn&#039;t breathe, sweating, and in  panic. It works for some but  if your not
sure if it works for you, don&#039;t try it, its not worth it!"
"It works. But as soon as I stop taking this.. little bumps on my hands always comes back. Still struggling wit
h this and I don&#039;t know what to do?? I had use virgin coconut oil but nothing worked..
Any one can help please"
"This pill does work! Trust me. I took this pill literally 20 minutes after having sex and I had bad cramping a
nd nausea and breast tenderness for about a week. I didn&#039;t get my period on time, I got it 5 days late. Bu
t hey, I got it! And I&#039;m so glad. This pill really does work."
"Worked for me, had mild side effects my virus was cleared. Very glad for this medicine, taken with Pegasys int
erferon"
```

This is because I'm using `cosine_similarity` on the embedding of the query "It works!!!" and the embeddings of all reviews to calculate how similar each review is to the query. Higher values indicate greater similarity. Then, I'm sorting the similarity scores and selecting the indices of the top 5 reviews, representing the reviews most similar in meaning to "It works!!!".

Question 10 (1 point)

Which pair of reviews (with non-identical text) have the highest similarity? Why do you think this is?

[74]:
```
similarities = cosine_similarity(embeddings)

np.fill_diagonal(similarities, -1)

for i in range(len(X)):
    for j in range(i + 1, len(X)):
        if X.iloc[i]['review'] == X.iloc[j]['review']:
            similarities[i, j] = -1
            similarities[j, i] = -1

max_index = np.unravel_index(np.argmax(similarities), similarities.shape)

review1 = X.iloc[max_index[0]]['review']
review2 = X.iloc[max_index[1]]['review']

print("Most similar reviews:")
print(f"Review 1: {review1}")
print(f"Review 2: {review2}")
```

```
Most similar reviews:
Review 1: "Works beautifully."
Review 2: "Works extremely well."
```

this is because im using cosine_similarity on the embeddings of the reviews to calculate a similarity score between every pair of reviews. Higher values indicate greater similarity. and im finding the indices of the maximum value in the similarity matrix, representing the two most similar reviews.

Question 7 (2 points)

Add the condition being treated as a feature. Does the model's accuracy improve?

[58]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

drug_reviews = drug_reviews.iloc[:len(y)]
embeddings_df = embeddings_df.iloc[:len(y)]

label_encoder = LabelEncoder()
embeddings_df['condition'] = label_encoder.fit_transform(drug_reviews['condition'])

X_train, X_test, y_train, y_test = train_test_split(embeddings_df, y, test_size=0.2, random_state=211)

model = LogisticRegression(random_state=211)
model.fit(X_train, y_train)

y_test_pred = model.predict(X_test)
new_accuracy = accuracy_score(y_test, y_test_pred)

print(f"Accuracy with condition as a feature: {new_accuracy:.4f}")
```

```
Accuracy with condition as a feature: 0.6500
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

[61]:
```
test_set_accuracy_with_condition = new_accuracy

the_model_performance_improves = test_set_accuracy_with_condition > test_set_accuracy
print(f"Model performance improves: {the_model_performance_improves}")
```

```
Model performance improves: False
```

Question 8 (1 points)

Is the condition feature more important than any single embedding dimension? Show your work.

[65]:
```
from sklearn.preprocessing import LabelEncoder

drug_reviews = drug_reviews.iloc[:len(y)]
embeddings_df = embeddings_df.iloc[:len(y)]

label_encoder = LabelEncoder()
embeddings_df['condition'] = label_encoder.fit_transform(drug_reviews['condition'])

X_train, X_test, y_train, y_test = train_test_split(embeddings_df, y, test_size=0.2, random_state=211)

# Train the model with the 'condition' feature:
model_with_condition = LogisticRegression(random_state=211)  # Make sure variable name is correct
model_with_condition.fit(X_train, y_train)

feature_importance = model_with_condition.coef_[0]

condition_feature_index = X_train.columns.get_loc('condition')

condition_importance = feature_importance[condition_feature_index]

embedding_importance = feature_importance[:384]

is_condition_more_important = condition_importance > embedding_importance.max()

print(is_condition_more_important)
```

```
False
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Question 9 (2 points)

Suppose you've got a new review: `"It works!!!"`.

Which reviews are most similar to this one? Why do you think this is?

[67]:
```
query_review = "It works!!!"
```

[10]:

	0	1	2	3	4	5	6	7	8	9	...	
0	-0.077628	-0.009143	-0.052208	0.010759	0.052049	-0.049303	0.066050	0.080704	-0.023603	-0.083210	...	0.0
1	-0.005510	0.009365	0.055555	0.103062	0.119525	-0.017933	0.036441	0.084298	-0.039507	-0.068830	...	-0.0
2	-0.074144	0.014378	0.017467	0.071874	0.036747	-0.016754	0.048345	0.113569	-0.061490	0.026435	...	0.0
3	-0.045093	-0.046545	0.001021	0.035365	-0.023364	0.023979	-0.016711	0.110397	-0.047272	-0.028503	...	-0.0
4	-0.068067	-0.020258	-0.015050	0.041240	-0.021989	-0.045596	0.053590	0.110229	-0.045288	0.028947	...	0.0
...	...	...	...	...	...	...	...	...	...	...	...	
1995	-0.058401	-0.043922	0.034079	0.108284	-0.006316	0.020449	0.034140	0.009383	-0.008800	-0.042013	...	-0.0
1996	-0.034792	0.010808	0.020451	0.071020	0.093329	0.038090	0.087002	0.107750	-0.106024	-0.080413	...	-0.(
1997	-0.069412	0.160490	-0.041720	0.029015	0.052835	0.096528	0.107080	-0.059257	0.010744	-0.061954	...	0.(
1998	-0.051795	0.022936	0.047044	0.045080	0.004642	0.024470	0.090476	0.140079	-0.041225	-0.073251	...	-0.(
1999	-0.035902	-0.054053	0.015757	0.059161	-0.057710	-0.011454	0.008595	0.092324	-0.033208	0.004964	...	0.(

2000 rows × 384 columns

Question 5 (1 points)

Using UMAP, produce a scatterplot of these embeddings.

[ ]:
```
!pip install umap-learn
```

[ ]:
```
!pip install dask[dataframe]
```

[ ]:
```
!pip install --upgrade numba
```

[ ]:
```
!pip install umap-learn[plot]
```

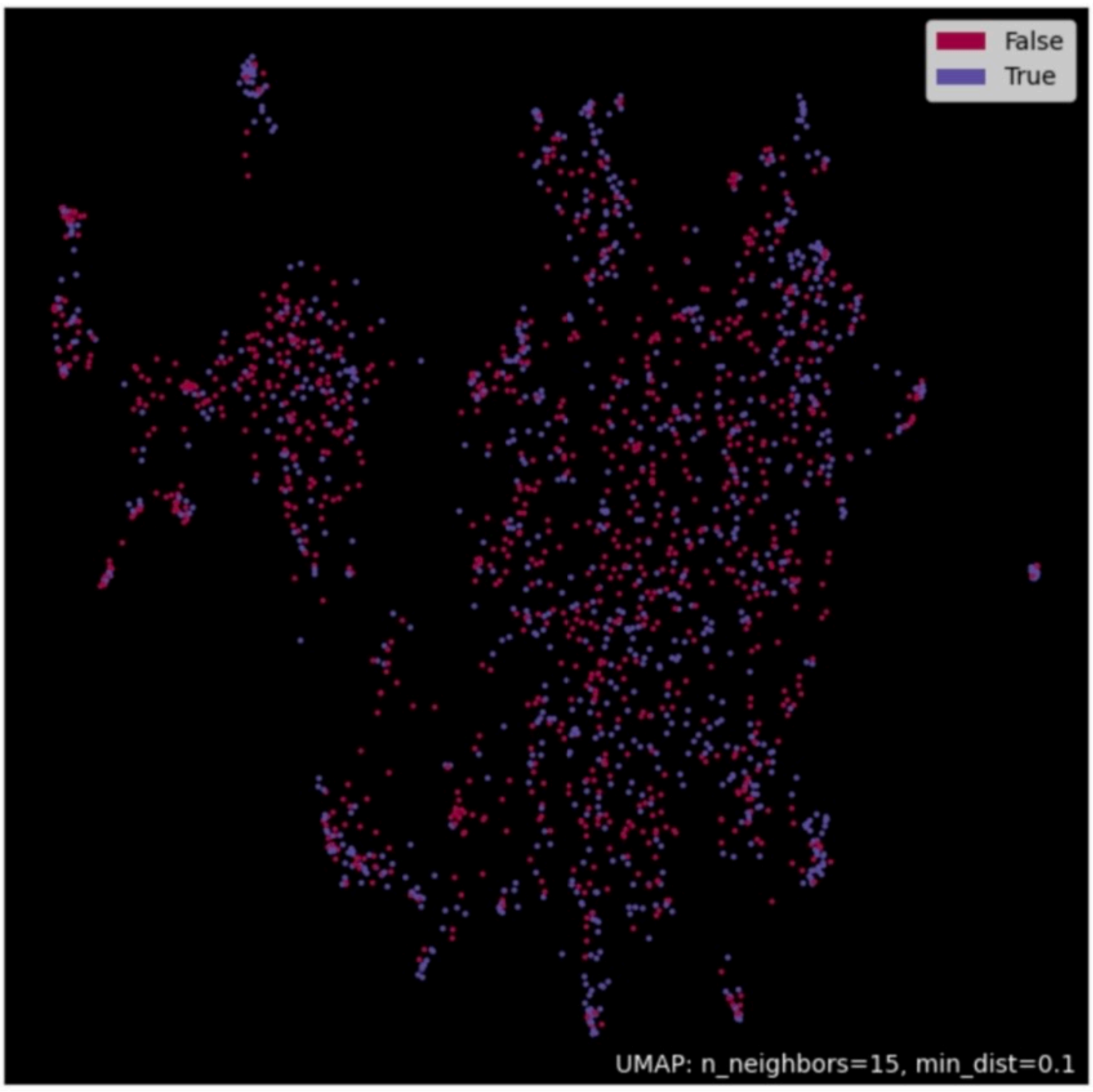
[25]:
```
import umap
import umap.plot
import matplotlib.pyplot as plt

reducer = umap.UMAP(n_neighbors=15, min_dist=0.1, metric='euclidean')

reducer.fit(embeddings_df)


umap.plot.points(reducer, labels=y, theme='viridis', cmap='viridis')
```

[25]:
```
<Axes: >
```



Question 6 (2 points)

Find a group of points on the scatterplot which seems well-separated from the others. What is distinctive about these reviews?

[37]:
```
import numpy as np
import pandas as pd


center_point = np.mean(reducer.embedding_, axis=0)
```

Question 3 (1 point)

What are the names of the 10 conditions which have the largest number of drugs used to treat them?

[16]:
```
top_drugs = drug_reviews.groupby('condition')['drugName'].nunique().sort_values(ascending=False)

# As a list
names_of_conditions_with_the_largest_number_of_drugs = top_drugs.head(10).index.tolist()
names_of_conditions_with_the_largest_number_of_drugs
```

[16]:
```
['Not Listed / Othe',
 'Pain',
 'Birth Control',
 'High Blood Pressure',
 'Acne',
 'Depression',
 'Rheumatoid Arthritis',
 'Diabetes, Type 2',
 'Allergic Rhinitis',
 'Insomnia']
```

Question 4 (1 points)

Prepare the data for further investigation and model training:

- Create a new column called `good_rating` in the DataFrame which indicates whether the rating is above the median rating
- Take a random sample of 2000 reviews, and use this `X` DataFrame for the following steps
- Create a y variable which contains the `good_rating` column
- Remove the `good_rating` and `rating` columns from `X`
- Create a new DataFrame called `embedding_df` which contains text embeddings for the reviews

[17]:
```
median_rating = drug_reviews['rating'].median()
drug_reviews['good_rating'] = drug_reviews['rating'] > median_rating
drug_reviews['good_rating']
```

[17]:

	good_rating
0	True
1	False
2	False
3	False
4	True
...	...
215058	True
215059	True
215060	False
215061	False
215062	True

215063 rows × 1 columns

dtype: bool

[18]:
```
# We're going to use a sample of the data for speed

X = drug_reviews.sample(2000, random_state=211).reset_index(drop=True)

# Create a y variable which contains the `good_rating` column
y = X['good_rating']
#  Remove the `rating` and `good_rating` from the X data frame
X = X.drop(columns=['good_rating', 'rating'])
```

[10]:
```
embedding_model = SentenceTransformer("all-MiniLM-L6-v2")
# Use `embedding_model` to create an embedding for each text

embeddings = embedding_model.encode(X['review'].tolist())
embeddings_df = pd.DataFrame(embeddings)
embeddings_df
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:104: UserWarning:
Error while fetching `HF_TOKEN` secret value from your vault: 'Requesting secret HF_TOKEN timed out. Secrets ca
n only be fetched when running from the Colab UI.'.
You are not authenticated with the Hugging Face Hub in this notebook.
If the error persists, please let us know by opening an issue on GitHub (https://github.com/huggingface/hugging
face_hub/issues/new).
  warnings.warn(
modules.json:   0%|          | 0.00/349 [00:00<?, ?B/s]
config_sentence_transformers.json:   0%|          | 0.00/116 [00:00<?, ?B/s]
README.md:   0%|          | 0.00/10.7k [00:00<?, ?B/s]
sentence_bert_config.json:   0%|          | 0.00/53.0 [00:00<?, ?B/s]
config.json:   0%|          | 0.00/612 [00:00<?, ?B/s]
model.safetensors:   0%|          | 0.00/90.9M [00:00<?, ?B/s]
tokenizer_config.json:   0%|          | 0.00/350 [00:00<?, ?B/s]
vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]
special_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]
1_Pooling/config.json:   0%|          | 0.00/190 [00:00<?, ?B/s]
```

```
[ ]: # Initialize Otter
import otter
grader = otter.Notebook("4_81bd.ipynb")
```

COMPSS211 Problem Set 4 ¶

Note: complete this problem set on Google Colab on a T4 machine for fastest performance.

```
[ ]: %pip install --quiet ucimlrepo sentence_transformers umap-learn
```

In this problem set, we're working with the drugs.com review dataset.

```
[2]: from ucimlrepo import fetch_ucirepo
from sentence_transformers import SentenceTransformer
import pandas as pd
import umap
```

```
/usr/local/lib/python3.10/dist-packages/sentence_transformers/cross_encoder/CrossEncoder.py:13: TqdmExperimentalWarning: Using `tqdm.autonotebook.tqdm` in notebook mode. Use `tqdm.tqdm` instead to force console mode (e.g. in jupyter console)
  from tqdm.autonotebook import tqdm, trange
```

```
[3]: drug_reviews_drugs_com = fetch_ucirepo(id=462)
drug_reviews = drug_reviews_drugs_com.data.features
drug_reviews
```

[3]:

	drugName	condition	review	rating	date	usefulCount
0	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combinati...	9	20-May-12	27
1	Guanfacine	ADHD	"My son is halfway through his fourth week of ...	8	27-Apr-10	192
2	Lybrel	Birth Control	"I used to take another oral contraceptive, wh...	5	14-Dec-09	17
3	Ortho Evra	Birth Control	"This is my first time using any form of birth...	8	3-Nov-15	10
4	Buprenorphine / naloxone	Opiate Dependence	"Suboxone has completely turned my life around...	9	27-Nov-16	37
...	...	...	...	...	...	...
215058	Tamoxifen	Breast Cancer, Prevention	"I have taken Tamoxifen for 5 years. Side effe...	10	13-Sep-14	43
215059	Escitalopram	Anxiety	"I've been taking Lexapro (escitaploprgra...	9	8-Oct-16	11
215060	Levonorgestrel	Birth Control	"I'm married, 34 years old and I have no ...	8	15-Nov-10	7
215061	Tapentadol	Pain	"I was prescribed Nucynta for severe neck/shou...	1	28-Nov-11	20
215062	Arthrotec	Sciatica	"It works!!!"	9	13-Sep-09	46

215063 rows × 6 columns

Question 1 (1 point)

Which drug with more than 100 reviews has the highest average rating?

```
[13]: review_counts = drug_reviews.groupby('drugName')['rating'].count()
drugs_with_more_than_100_reviews = review_counts[review_counts > 100].index
reviews_for_popular_drugs = drug_reviews[drug_reviews['drugName'].isin(drugs_with_more_than_100_reviews)]
average_ratings = reviews_for_popular_drugs.groupby('drugName')['rating'].mean()
drug_with_highest_average_rating = average_ratings.idxmax()
drug_with_highest_average_rating
```

```
[13]: 'Librium'
```

Question 2 (1 point)

On average, are reviews with high ratings considered more useful? Show your work.

```
[14]: median_rating = drug_reviews['rating'].median()

average_usefulness_high_rating = drug_reviews[drug_reviews['rating'] > median_rating]['usefulCount'].mean()
average_usefulness_low_rating = drug_reviews[drug_reviews['rating'] <= median_rating]['usefulCount'].mean()
```

```
[15]: # The value should be True or False
reviews_with_high_ratings_are_considered_more_useful = average_usefulness_high_rating > average_usefulness_low_
reviews_with_high_ratings_are_considered_more_useful
```

```
[15]: True
```