DSAI510 Assignment 01

September 27, 2025

1 Assignment 1 - Deadline: Oct 5, 2025, Sun 11pm

DSAI 510 Fall 2025 Complete the assignment below and upload both the .ipynb file AND its pdf to https://moodle.boun.edu.tr by the deadline. The submission page on Moodle will close automatically after this date and time.

To make a pdf, this may work: Hit CMD+P or CTRL+P, and save it as PDF. You may also use other options from the File menu.

1.1 Note about markdowns in Jupyter notebooks

Markdown cells contain headings, text, links, and images. Double-click this cell to see how it's written.

To create a Markdown cell, use the cell-type dropdown in the toolbar and select "Markdown" (you can also hover just below a cell and use the cell menu).

You can copy and paste images directly into Markdown cells. The images are embedded in the .ipynb file, so no separate image files are created.

When you're done editing a Markdown cell, press Shift+Enter to render it in a clean, human-readable form.

1.1.1 Problem 1 (10 pts)

a) Create a dataframe using pandas library for the table below and save it as df.

	Color	Number Label
0	red	1
1	green	2
2	blue	3

b) Show the dataframe so it's displayed as above.

c) Make both columns of categorical type (yes, I want Number Label also categorical). Then, check to verify that both columns are of categorical type.

(Use a different codeblock for each part a), b) and c). Run your codeblocks with Shift+Enter so the output shows under each cell.)

[ ]:
```
# Part a
import pandas as pd
df = pd.DataFrame({"Color": ['red', 'green', 'blue'], "Number Label": [1, 2,␣
↪3]})
```

[ ]:
```
# Part b
df
```

[ ]:
```
   Color  Number Label
0    red             1
1  green             2
2   blue             3
```

[ ]:
```
# Part c
df["Color"] = df["Color"].astype('category')
df["Number Label"] = df["Number Label"].astype('category')

df.dtypes
```

[ ]:
```
Color           category
Number Label    category
dtype: object
```

1.1.2 Problem 2 (10 pts)

Load the `breast_cancer` dataset from the sklearn library into a dataframe. Display the first five and last five records. Also, show the total number of columns and rows in the dataset.

[23]:
```
from sklearn.datasets import load_breast_cancer

bc_json = load_breast_cancer()
bc = pd.DataFrame(bc_json.data, columns=bc_json.feature_names)

print("Total rows:", bc.shape[0])
print("Total columns", bc.shape[1])
```

```
Total rows: 569
Total columns 30
```

[24]:
```
print("===== First 5 =====")
bc.head()
```

```
===== First 5 =====
```

[24]:
```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
0        17.99         10.38          122.80     1001.0          0.11840
```

```
1        20.57         17.77           132.90     1326.0          0.08474
2        19.69         21.25           130.00     1203.0          0.10960
3        11.42         20.38            77.58      386.1          0.14250
4        20.29         14.34           135.10     1297.0          0.10030

   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  …  worst radius  worst texture  worst perimeter  \
0                 0.07871  …         25.38          17.33           184.60
1                 0.05667  …         24.99          23.41           158.80
2                 0.05999  …         23.57          25.53           152.50
3                 0.09744  …         14.91          26.50            98.87
4                 0.05883  …         22.54          16.67           152.20

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]
```

[25]:
```
print("===== Last 5 =====")
bc.tail()
```

```
===== Last 5 =====
```

[25]:
```
     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
564        21.56         22.39          142.00     1479.0          0.11100
565        20.13         28.25          131.20     1261.0          0.09780
566        16.60         28.08          108.30      858.1          0.08455
567        20.60         29.33          140.10     1265.0          0.11780
568         7.76         24.54           47.92      181.0          0.05263

     mean compactness  mean concavity  mean concave points  mean symmetry  \
```

```
564          0.11590         0.24390              0.13890         0.1726
565          0.10340         0.14400              0.09791         0.1752
566          0.10230         0.09251              0.05302         0.1590
567          0.27700         0.35140              0.15200         0.2397
568          0.04362         0.00000              0.00000         0.1587

     mean fractal dimension  …  worst radius  worst texture  \
564                 0.05623  …        25.450          26.40
565                 0.05533  …        23.690          38.25
566                 0.05648  …        18.980          34.12
567                 0.07016  …        25.740          39.42
568                 0.05884  …         9.456          30.37

     worst perimeter  worst area  worst smoothness  worst compactness  \
564           166.10      2027.0           0.14100            0.21130
565           155.00      1731.0           0.11660            0.19220
566           126.70      1124.0           0.11390            0.30940
567           184.60      1821.0           0.16500            0.86810
568            59.16       268.6           0.08996            0.06444

     worst concavity  worst concave points  worst symmetry  \
564           0.4107                0.2216          0.2060
565           0.3215                0.1628          0.2572
566           0.3403                0.1418          0.2218
567           0.9387                0.2650          0.4087
568           0.0000                0.0000          0.2871

     worst fractal dimension
564                  0.07115
565                  0.06637
566                  0.07820
567                  0.12400
568                  0.07039

[5 rows x 30 columns]
```

1.1.3 Problem 3 (10 pts)

Use `yfinance` library to get and display silver prices for the last 10 days.

[ ]:
```python
from datetime import datetime, timedelta
import yfinance as yf

today = datetime.now()
start_date = today - timedelta(days=10)
date_format = "%Y-%m-%d"
```

```
# is SLV correct?
silver = yf.download('SLV', start=start_date.strftime(date_format), end=today.
 ↪strftime(date_format), auto_adjust=True)
silver
```

```
[**********************100%**********************]  1 of 1 completed
```

[ ]:

Price	Close	High	Low	Open	Volume
Ticker	SLV	SLV	SLV	SLV	SLV
Date					
2025-09-17	37.790001	38.340000	37.349998	38.040001	31045500
2025-09-18	37.990002	38.000000	37.610001	37.849998	13289200
2025-09-19	39.040001	39.119999	38.189999	38.259998	36776300
2025-09-22	40.040001	40.049999	39.340000	39.500000	32055400
2025-09-23	39.959999	40.360001	39.790001	40.310001	28492600
2025-09-24	39.820000	40.160000	39.639999	40.130001	31533200
2025-09-25	41.029999	41.060001	40.119999	40.459999	36395800
2025-09-26	41.860001	42.330002	41.090000	41.230000	44540700

1.1.4 Problem 4 (10 pts)

Use `requests` library to pull from internet some information of your choosing.

[ ]:
```
import requests
import json
import os
from dotenv import load_dotenv

load_dotenv()

response = requests.post(

    url="https://openrouter.ai/api/v1/chat/completions",

    headers={
        "Authorization": f"Bearer {os.getenv('OPENROUTER_API_KEY')}",
        "Content-Type": "application/json"
    },

    data=json.dumps({
        "model": "qwen/qwen2.5-vl-32b-instruct:free",
        "messages": [
            {
                "role": "user",
                "content": [
                    {
                        "type": "text",
```

```
                        "text": "Can you summarize this poster in terms of␣
↪conference, first and last authors with their respective affiliations, and␣
↪abstract in two sentences?"
                    },
                    {
                        "type": "image_url",
                        "image_url": {
                            "url": "https://media.licdn.com/dms/image/v2/
↪D4E22AQEH_V_jA594fg/feedshare-shrink_2048_1536/B4EZkAojMrIIA4-/0/
↪1756652256867?
↪e=1761782400&v=beta&t=J3j1gg3bL1D4LMvGDjU8tmireUEjlTOe-aRN3m7-_V4"
                        }
                    }
                ]
            }
        ],
    })
)
```

[19]:
```
result = response.json()
# print(result)
# print("==="*15)
result['choices'][0]['message']['content']
```

[19]:
```
'### Summary of the Poster\n\n#### **Conference:**\nThe poster is from the *IEEE
MLSP 2025* conference held in Istanbul, Türkiye, from August 31 to September 3,
2025.\n\n#### **Authors and Affiliations:**\n- **First Author:** Yiğit
Ateş<sup>1</sup>\n  - Affiliation: TAM Finans R&D, Istanbul, Türkiye\n- **Last
Author:** Süayb S. Arslan<sup>2,3</sup>\n  - Affiliation:  \n    - <sup>2</sup>
Department of Computer Engineering and Institute for DSAI, Boğaziçi University,
Istanbul, Türkiye  \n    - <sup>3</sup> Department of Brain and Cognitive
Sciences, MIT, Cambridge, MA, USA\n\n#### **Abstract in Two Sentences:**\nThis
research introduces a novel methodology for enhancing RAG (Retrieval-Augmented
Generation) systems by combining semantic chunking with Chain-of-Thought (CoT)
reasoning. The proposed approach uses LLMs (large language models) to generate
enhanced document chunks with metadata, improving retrieval and response
quality. By integrating BM25 lexical matching with semantic vector search, the
method significantly outperforms traditional RAG systems, achieving high
accuracy and stability across various chunk sizes, all without requiring
unsupervised fine-tuning. The results demonstrate enhanced coherence and
retrieval performance, paving the way for more robust document processing
systems.'
```

1.1.5 Problem 5 (10 pts)

Install postgresql (use Youtube, ChatGPT, Google, PostgreSQL webpages etc.) to your own computer (not Google Colab!). It's very likely that you'll run into technical problems. Let's see if you can use the Internet to solve technical problems.

Then, load dvdrentar.tar database (don't open the tar) into your PostgreSQL server.

Make a SQL querry to get 10 records from the "actor" table and display the result.

[3]:
```
import psycopg2
import pandas as pd

conn = psycopg2.connect(
    host="localhost",
    database="dvdrental",
    user="yigitates",
    port="5432"
)

query = """
SELECT * FROM actor
LIMIT 10
"""

df = pd.read_sql_query(query, conn)
conn.close()
df
```

```
/var/folders/5d/9l6gb_c515qgg88vn1bg1y340000gn/T/ipykernel_25103/872611031.py:16
: UserWarning: pandas only supports SQLAlchemy connectable (engine/connection)
or database string URI or sqlite3 DBAPI2 connection. Other DBAPI2 objects are
not tested. Please consider using SQLAlchemy.
  df = pd.read_sql_query(query, conn)
```

[3]:

	actor_id	first_name	last_name	last_update
0	1	Penelope	Guiness	2013-05-26 14:47:57.620
1	2	Nick	Wahlberg	2013-05-26 14:47:57.620
2	3	Ed	Chase	2013-05-26 14:47:57.620
3	4	Jennifer	Davis	2013-05-26 14:47:57.620
4	5	Johnny	Lollobrigida	2013-05-26 14:47:57.620
5	6	Bette	Nicholson	2013-05-26 14:47:57.620
6	7	Grace	Mostel	2013-05-26 14:47:57.620
7	8	Matthew	Johansson	2013-05-26 14:47:57.620
8	9	Joe	Swank	2013-05-26 14:47:57.620
9	10	Christian	Gable	2013-05-26 14:47:57.620