510-assignment01-ahmet-kasim-erbay

October 5, 2024

1 Assignment 1 - Deadline: Oct 2, 2024 11pm

DSAI 510 Fall 2024 Complete the assignment below and upload both the .ipynb file AND its pdf to https://moodle.boun.edu.tr by **Oct 2, 2024 11pm**. 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

Markdowns are cells that contain headings, text, links and images. Double click here and see how it's made.

To make a markdown cell, simply click on the cell and choose "Markdown" from the menu above (see the image below).

You can copy and paste images into markdown cells. The image will be embedded in the .ipynb file so there won't be any separate file for the images.

When you're done typing the markdown cell, simply press Shift+Enter and it will be rendered into a nice, human-readable form.

1.1.1 Problem 1

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

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

c) Make both columns of categorical type (yes, I want Number Label also categorical). Check at the end if 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.)

[1]:
```
# Part a
#your code here
import pandas as pd

data = {
    "Color":["red", "green", "blue"],
    "Number Label":[1,2,3]
```

```
}

df = pd.DataFrame(data)
```

[2]:
```
# Part b

#your code here
display(df)
```

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

[3]:
```
# Part c

#your code here

df.Color = df.Color.astype("category")
df["Number Label"] = df["Number Label"].astype("category")

print(df["Color"].dtype)
print(df["Number Label"].dtypes)
```

```
category
category
```

1.1.2 Problem 2

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.

[4]:
```
from sklearn.datasets import load_breast_cancer

breast_cancer = load_breast_cancer()

df = pd.DataFrame(breast_cancer.data, columns=breast_cancer.feature_names)
df["target"] = breast_cancer.target

print(f"Total # of Columns", len(df.columns))
print(f"Total # of Rows", len(df))

display(df.head())
display(df.tail())
```

```
Total # of Columns 31
Total # of Rows 569
```

```
   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 texture  worst perimeter  worst area  \
0                 0.07871  …          17.33           184.60      2019.0
1                 0.05667  …          23.41           158.80      1956.0
2                 0.05999  …          25.53           152.50      1709.0
3                 0.09744  …          26.50            98.87       567.7
4                 0.05883  …          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave points  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
     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 texture  worst perimeter  worst area  \
564                 0.05623  …          26.40           166.10      2027.0
565                 0.05533  …          38.25           155.00      1731.0
566                 0.05648  …          34.12           126.70      1124.0
567                 0.07016  …          39.42           184.60      1821.0
568                 0.05884  …          30.37            59.16       268.6

     worst smoothness  worst compactness  worst concavity  \
564           0.14100            0.21130           0.4107
565           0.11660            0.19220           0.3215
566           0.11390            0.30940           0.3403
567           0.16500            0.86810           0.9387
568           0.08996            0.06444           0.0000

     worst concave points  worst symmetry  worst fractal dimension  target
564                0.2216          0.2060                  0.07115       0
565                0.1628          0.2572                  0.06637       0
566                0.1418          0.2218                  0.07820       0
567                0.2650          0.4087                  0.12400       0
568                0.0000          0.2871                  0.07039       1

[5 rows x 31 columns]
```

1.1.3 Problem 3

Use quandl API to get and display silver prices for the last 10 days.

(See Lecture 02.ipynb about how to sign up and get your free API key.)

[5]:
```
from keys import *
import requests
from datetime import datetime, timedelta

def time_formatter():
    today_date = str(datetime.today()).split(" ")[0]
    _10_days_before = str(datetime.today() - timedelta(12)).split(" ")[0]
    return today_date, _10_days_before

today, _10_days_before = time_formatter()

url = f"""https://api.nasdaq.com/api/quote/SI:CMX/historical?
 ↪assetclass=commodities&fromdate={_10_days_before}&todate={today}"""

response = requests.get(url)
```

```
if response.status_code == 200:

    columns = list(response.json()["data"]["tradesTable"]["headers"].values())

    rows = [list(i.values()) for i in response.
 ↪json()["data"]["tradesTable"]["rows"]]

    df = pd.DataFrame(rows, columns=columns)

    display(df)
else:
    print(f"Status Code {response.status_code}")
```

	Date	Close/Last	Volume	Open	High	Low
0	10/04/2024	32.394	85,593	32.29	33.225	31.755
1	10/03/2024	32.464	55,702	32.10	32.485	31.65
2	10/02/2024	31.92	69,133	31.705	32.59	31.26
3	10/01/2024	31.742	60,171	31.42	32.145	31.38
4	09/30/2024	31.458	56,238	31.945	32.15	31.155
5	09/27/2024	31.816	73,451	32.31	32.59	31.635
6	09/26/2024	32.341	90,777	32.11	33.02	32.035
7	09/25/2024	32.018	84,038	32.445	32.61	31.87
8	09/24/2024	32.43	110,550	31.02	32.60	30.96
9	09/23/2024	31.085	60,524	31.535	31.55	30.67

1.1.4 Problem 4 (10 pts)

Download the dvdrental.tar PostgreSQL database from https://www.postgresqltutorial.com/postgresql-getting-started/postgresql-sample-database/ Don't open the tar, you'll use it as tar.

Install postgresql (use Youtube, ChatGPT, Google, PostgreSQL webpages etc.) and necessary Python libraries (see Lecture 02.ipynb notebook). It's very likely that you'll run into technical problems. Let's see if you can use the Internet to solve technical problems.

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

[6]:
```
import pandas as pd
from sqlalchemy import create_engine

username = "dvdrental"  # this is usually the default username
password = "dvdrental"

host = "localhost"
port = "5432"

# Create an engine instance
engine = create_engine(f'postgresql://{username}:{password}@{host}:{port}')

try:
```

```
    connection = engine.connect()
    print("Successfully connected to the dvdrental database!")
except Exception as e:
    print(f"Error: {e}")
finally:
    connection.close()
```

```
Successfully connected to the dvdrental database!
```

[7]:
```
query = "SELECT * FROM ACTOR LIMIT 10;"
tables = pd.read_sql(query, engine)
display(tables)
```

	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