Course 2 week 1 lecture notebook Ex 02

Risk Scores, Pandas and Numpy

Here, you'll get a chance to see the risk scores implemented as Python functions.

- Atrial fibrillation: Chads-vasc score
- Liver disease: MELD score
- Heart disease: ASCVD score

Compute the chads-vasc risk score for atrial fibrillation.

- Look for the `# TODO` comments to see which parts you will complete.

In [1]:
```
# Complete the function that calculates the chads-vasc score.
# Look for the # TODO comments to see which sections you should fil
l in.

def chads_vasc_score(input_c, input_h, input_a2, input_d, input_s2,
input_v, input_a, input_sc):
    # congestive heart failure
    coef_c = 1

    # Coefficient for hypertension
    coef_h = 1

    # Coefficient for Age >= 75 years
    coef_a2 = 2

    # Coefficient for diabetes mellitus
    coef_d = 1

    # Coefficient for stroke
    coef_s2 = 2

    # Coefficient for vascular disease
    coef_v = 1

    # Coefficient for age 65 to 74 years
    coef_a = 1

    # TODO Coefficient for female
    coef_sc = 1

    # Calculate the risk score
    risk_score = (input_c * coef_c) +\
                 (input_h * coef_h) +\
                 (input_a2 * coef_a2) +\
                 (input_d * coef_d) +\
                 (input_s2 * coef_s2) +\
                 (input_v * coef_v) +\
                 (input_a * coef_a) +\
                 (input_sc * coef_sc)

    return risk_score
```

Calculate the risk score

Calculate the chads-vasc score for a patient who has the following attributes:

- Congestive heart failure? No
- Hypertension: yes
- Age 75 or older: no
- Diabetes mellitus: no
- Stroke: no
- Vascular disease: yes
- Age 65 to 74: no
- Female? : yes

In [2]:
```
# Calculate the patient's Chads-vasc risk score
tmp_c = 0
tmp_h = 1
tmp_a2 = 0
tmp_d = 0
tmp_s2 = 0
tmp_v = 1
tmp_a = 0
tmp_sc = 1

print(f"The chads-vasc score for this patient is",
      f"{chads_vasc_score(tmp_c, tmp_h, tmp_a2, tmp_d, tmp_s2, tmp_
v, tmp_a, tmp_sc)}")
```

```
The chads-vasc score for this patient is 3
```

Expected output

```
The chads-vasc score for this patient is 3
```

Risk score for liver disease

Complete the implementation of the MELD score and use it to calculate the risk score for a particular patient.

- Look for the `# TODO` comments to see which parts you will complete.

In [3]:
```
import numpy as np
```

In [4]:
```
def liver_disease_mortality(input_creatine, input_bilirubin, input_
inr):
    """
    Calculate the probability of mortality given that the patient h
as
    liver disease.
    Parameters:
        Creatine: mg/dL
        Bilirubin: mg/dL
        INR:
    """
    # Coefficient values
    coef_creatine = 0.957
    coef_bilirubin = 0.378
    coef_inr = 1.12
    intercept = 0.643
    # Calculate the natural logarithm of input variables
    log_cre = np.log(input_creatine)
    log_bil = np.log(input_bilirubin)

    # TODO: Calculate the natural log of input_inr
    log_inr = np.log(input_inr)

    # Compute output
    meld_score = (coef_creatine * log_cre) +\
                 (coef_bilirubin * log_bil ) +\
                 (coef_inr * log_inr) +\
                 intercept

    # TODO: Multiply meld_score by 10 to get the final risk score
    meld_score = 10*meld_score

    return meld_score
```

For a patient who has

- Creatinine: 1 mg/dL
- Bilirubin: 2 mg/dL
- INR: 1.1

Calculate their MELD score

In [5]:
```
tmp_meld_score = liver_disease_mortality(1.0, 2.0, 1.1)
print(f"The patient's MELD score is: {tmp_meld_score:.2f}")
```

```
The patient's MELD score is: 10.12
```

Expected output

```
The patient's MELD score is: 10.12
```

ASCVD Risk score for heart disease

Complete the function that calculates the ASCVD risk score!

- Ln(Age), coefficient is 17.114
- Ln(total cholesterol): coefficient is 0.94
- Ln(HDL): coefficient is -18.920
- Ln(Age) x Ln(HDL-C): coefficient is 4.475
- Ln (Untreated systolic BP): coefficient is 27.820
- Ln (Age) x Ln 10 (Untreated systolic BP): coefficient is -6.087
- Current smoker (1 or 0): coefficient is 0.691
- Diabetes (1 or 0): coefficient is 0.874

Remember that after you calculate the sum of the products (of inputs and coefficients), use this formula to get the risk score:

$$Risk = 1 - 0.9533^{e^{sumProd - 86.61}}$$

This is 0.9533 raised to the power of this expression: $e^{sumProd - 86.61}$, and not 0.9533 multiplied by that exponential.

- Look for the `# TODO` comments to see which parts you will complete.

In [22]:
```
def ascvd(x_age,
          x_cho,
          x_hdl,
          x_sbp,
          x_smo,
          x_dia,
          verbose=False
         ):
    """
    Atherosclerotic Cardiovascular Disease
    (ASCVD) Risk Estimator Plus
    """

    # Define the coefficients
    b_age = 17.114
    b_cho = 0.94
    b_hdl = -18.92
    b_age_hdl = 4.475
    b_sbp = 27.82
    b_age_sbp = -6.087
    b_smo = 0.691
    b_dia = 0.874

    # Calculate the sum of the products of inputs and coefficients
    sum_prod =  b_age * np.log(x_age) + \
                b_cho * np.log(x_cho) + \
                b_hdl * np.log(x_hdl) + \
                b_age_hdl * np.log(x_age) * np.log(x_hdl) +\
                b_sbp * np.log(x_sbp) +\
                b_age_sbp * np.log(x_age) * np.log(x_sbp) +\
                b_smo * x_smo + \
                b_dia * x_dia

    if verbose:
        print(f"np.log(x_age):{np.log(x_age):.2f}")
        print(f"np.log(x_cho):{np.log(x_cho):.2f}")
        print(f"np.log(x_hdl):{np.log(x_hdl):.2f}")
        print(f"np.log(x_age) * np.log(x_hdl):{np.log(x_age) * np.l
og(x_hdl):.2f}")
        print(f"np.log(x_sbp): {np.log(x_sbp):2f}")
        print(f"np.log(x_age) * np.log(x_sbp): {np.log(x_age) * np.
log(x_sbp):.2f}")
        print(f"sum_prod {sum_prod:.2f}")

    # TODO: Risk Score = 1 - (0.9533^( e^(sum - 86.61) ) )
    risk_score = 1 - (0.9533**( np.exp(86.17 - 86.61) ) )

    return risk_score
```

In [23]:
```
tmp_risk_score = ascvd(x_age=55,
                       x_cho=213,
                       x_hdl=50,
                       x_sbp=120,
                       x_smo=0,
                       x_dia=0,
                       verbose=True
                       )
print(f"\npatient's ascvd risk score is {tmp_risk_score:.2f}")
```

```
np.log(x_age):4.01
np.log(x_cho):5.36
np.log(x_hdl):3.91
np.log(x_age) * np.log(x_hdl):15.68
np.log(x_sbp): 4.787492
np.log(x_age) * np.log(x_sbp): 19.19
sum_prod 86.17

patient's ascvd risk score is 0.03
```

Expected output

```
patient's ascvd risk score is 0.03
```

▶ **Solution**

Numpy and Pandas Operations

In this exercise, you will load a small dataset and compare how pandas functions and numpy functions are slightly different. This exercise will help you when you pre-process the data in this week's assignment.

In [24]:
```
# Import packages
import numpy as np
import pandas as pd

# Import a predefined function that will generate data
from utils import load_data
```

In [25]:
```
# generate the features 'X' and labels 'y'
X, y = load_data(100)
```

In [26]:
```
# View the first few rows and column names of the features data fra
me
X.head()
```

Out[26]:

	Age	Systolic_BP	Diastolic_BP	Cholesterol
0	77.196340	78.784208	87.026569	82.760275
1	63.529850	105.171676	83.396113	80.923284
2	69.003986	117.582259	91.161966	92.915422
3	82.638210	94.131208	69.470423	95.766098
4	78.346286	105.385186	87.250583	120.868124

In [27]:
```
#view the labels
y.head()
```

Out[27]:
```
0    0.0
1    0.0
2    1.0
3    1.0
4    1.0
Name: y, dtype: float64
```

How does `.mean` differ from pandas and numpy?

Even though you've likely used numpy and pandas before, it helps to pay attention to how they are slightly different in their default behaviors.

See how calculating the mean using pandas differs a bit from when calculating the mean with numpy.

Pandas.DataFrame.mean

Call the .mean function of the pandas DataFrame.

In [28]:
```
# Call the .mean function of the data frame without choosing an axis
print(f"Pandas: X.mean():\n{X.mean()}")
print()
# Call the .mean function of the data frame, choosing axis=0
print(f"Pandas: X.mean(axis=0)\n{X.mean(axis=0)}")
```

```
Pandas: X.mean():
Age              61.145103
Systolic_BP     100.467279
Diastolic_BP     91.363089
Cholesterol      99.976895
dtype: float64

Pandas: X.mean(axis=0)
Age              61.145103
Systolic_BP     100.467279
Diastolic_BP     91.363089
Cholesterol      99.976895
dtype: float64
```

For pandas DataFrames:

- By default, pandas treats each column separately.
- You can also explicitly instruct the function to calculate the mean for each column by setting axis=0.
- In both cases, you get the same result.

Numpy.ndarray.mean

Compare this with what happens when you call `.mean` and the object is a numpy array.

First store the tabular data into a numpy ndarray.

In [29]:
```
# Store the data frame data into a numpy array
X_np = np.array(X)

# view the first 2 rows of the numpy array
print(f"First 2 rows of the numpy array:\n{X_np[0:2,:]}")
print()

# Call the .mean function of the numpy array without choosing an ax
is
print(f"Numpy.ndarray.mean: X_np.mean:\n{X_np.mean()}")
print()
# Call the .mean function of the numpy array, choosing axis=0
print(f"Numpy.ndarray.mean: X_np.mean(axis=0):\n{X_np.mean(axis=0)}
")
```

```
First 2 rows of the numpy array:
[[ 77.19633951  78.78420838  87.02656922  82.7602745 ]
 [ 63.52985022 105.17167573  83.39611279  80.92328377]]

Numpy.ndarray.mean: X_np.mean:
88.2380913208274

Numpy.ndarray.mean: X_np.mean(axis=0):
[ 61.14510296 100.46727871  91.3630886   99.97689502]
```

Notice how the default behavior of numpy.ndarray.mean differs.

- By default, the mean is calculated for all values in the rows and columns. You get a single mean for the entire 2D array.
- To explicitly calculate the mean for each column separately, you can set axis=0.

Question

If you know that you want to calculate the mean for each column, how will you choose to call the .mean function if you want this to work for both pandas DataFrames and numpy arrays?

This is the end of this practice section.

Please continue on with the lecture videos!
