**SECB3203 PROGRAMMING FOR BIOINFORMATICS – Lab 2 (12%)**

**Due: 2 Jan 2026 (Friday) 11.59 PM MYT**

**Note:**

- **Filename: L2_YourName.py/ipynb** (e.g., *L2_Ali.py or L2_Ali.ipynb*)
- **Put your full name and matric number at the beginning of the codes.**
- **For the last part, you just need to zip your figure and your source code file answer together in the submission.**

## PART A – MACHINE LEARNING (3%)

The breast cancer Wisconsin dataset includes two distinct classes of samples: malignant and benign. This dataset has been extensively studied in the field of machine learning (Ahmad, 2016). Feature selection techniques, such as univariate feature selection, have been employed to identify the most relevant features for classification based on univariate statistical tests like the chi-squared test and F-test (Vinaixa et al., 2012). Furthermore, various classification algorithms, including Bayesian network, naïve Bayes, decision trees, and multilayer neural network, have been compared to classify benign and malignant cancer from the Wisconsin Breast Cancer dataset (Omran et al., 2021). Additionally, the Wisconsin Diagnosis Breast Cancer (WDBC) and Wisconsin Breast Cancer (WBC) datasets have been utilized in research to develop breast cancer diagnosis methods (Huang & Chen, 2022). Moreover, a comprehensive analysis of machine learning classification algorithms with and without feature selection has been performed on the Wisconsin Breast Cancer Original (WBCO), Wisconsin Diagnosis Breast Cancer (WDBC), and Wisconsin Prognosis Breast Cancer (WPBC) datasets for breast cancer prediction (Hasan & Shafi, 2023). These studies demonstrate the significance of the Wisconsin breast cancer dataset and the application of feature selection techniques in developing accurate classification models for breast cancer diagnosis.

**Tasks:**

Write the correct Python commands in each activity below to fulfil the demonstration of feature selection using chi-squared test and F-test in breast cancer Wisconsin dataset from sklearn.

**[INPUT]**

The breast cancer Wisconsin dataset that loads from sklearn, consists of features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. These features are used to perform machine learning tasks such as classification and clustering. The dataset includes

various attributes, such as mean radius, mean texture, mean perimeter, mean area, mean smoothness, mean compactness, mean concavity, mean concave points, mean symmetry, mean fractal dimension, radius error, texture error, perimeter error, area error, smoothness error, compactness error, concavity error, concave points error, symmetry error, fractal dimension error, worst radius, worst texture, worst perimeter, worst area, worst smoothness, worst compactness, worst concavity, worst concave points, worst symmetry, and worst fractal dimension. These attributes are essential for training machine learning models to classify breast cancer tumors as malignant or benign.

**[PROCESS]**

1. Import the libraries: numpy, pandas, sklearn.
2. Import the datasets from sklearn.
3. Import SelectKBest, chi2, and f_classif from sklearn.feature_selection.
4. Load the dataset using load_breast_cancer().
5. Print the dimensions of the data frame for the dataset (samples vs features).
6. Print all features and the size of the features.
7. Print the size of malignant and benign samples respectively.
8. Print the first three samples with their feature values.
9. Use univariate feature selection with a chi-squared test is used for feature scoring to select the top five features according to the k highest scores.
10. Build the model using fit().
11. Print the selected top five features with the lowest p-values which achieved p-values <= 0.05 using pvalues_.
12. Use univariate feature selection with F-test is used for feature scoring to select the top five features according to the k highest scores.
13. Build the model using fit().
14. Print the selected top five features with the lowest p-values which achieved p-values <= 0.05 using pvalues_.
15. Print the same features from the p-values of the chi-squared test and F-test.

**Hint for the functions needed:**

append()

DataFrame()

sort_values()

head()

len()

iloc[]

**[EXAMPLE OUTPUT] (**not necessarily you will get the exact same answer**)**

```
dimension:  (569, 30)
30 Features:
 ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
ID for malignant with size 212  :
 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44, 45, 47, 53, 54, 56, 57, 62, 64, 65, 70,
72, 73, 75, 77, 78, 82, 83, 85, 86, 87, 91, 94, 95, 99, 100, 105, 108, 117, 118, 119, 121,
122, 126, 127, 129, 131, 132, 134, 135, 138, 141, 146, 156, 161, 162, 164, 167, 168, 171,
172, 177, 180, 181, 182, 184, 186, 190, 193, 194, 196, 197, 198, 199, 201, 202, 203, 205,
207, 210, 212, 213, 214, 215, 218, 219, 223, 229, 230, 233, 236, 237, 239, 244, 250, 252,
253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 272, 274, 277, 280, 282,
283, 297, 300, 302, 317, 321, 323, 328, 329, 330, 335, 337, 339, 343, 351, 352, 353, 365,
366, 368, 369, 370, 372, 373, 379, 385, 389, 392, 393, 400, 408, 414, 417, 430, 432, 433,
435, 441, 444, 446, 449, 451, 460, 461, 468, 479, 487, 489, 492, 498, 499, 501, 503, 509,
512, 514, 516, 517, 521, 533, 535, 536, 562, 563, 564, 565, 566, 567]
ID for benign with size 357  :
 [19, 20, 21, 37, 46, 48, 49, 50, 51, 52, 55, 58, 59, 60, 61, 63, 66, 67, 68, 69, 71, 74, 76, 79,
80, 81, 84, 88, 89, 90, 92, 93, 96, 97, 98, 101, 102, 103, 104, 106, 107, 109, 110, 111, 112,
113, 114, 115, 116, 120, 123, 124, 125, 128, 130, 133, 136, 137, 139, 140, 142, 143, 144,
145, 147, 148, 149, 150, 151, 152, 153, 154, 155, 157, 158, 159, 160, 163, 165, 166, 169,
```

170, 173, 174, 175, 176, 178, 179, 183, 185, 187, 188, 189, 191, 192, 195, 200, 204, 206, 208, 209, 211, 216, 217, 220, 221, 222, 224, 225, 226, 227, 228, 231, 232, 234, 235, 238, 240, 241, 242, 243, 245, 246, 247, 248, 249, 251, 266, 267, 268, 269, 270, 271, 273, 275, 276, 278, 279, 281, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 298, 299, 301, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 318, 319, 320, 322, 324, 325, 326, 327, 331, 332, 333, 334, 336, 338, 340, 341, 342, 344, 345, 346, 347, 348, 349, 350, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 367, 371, 374, 375, 376, 377, 378, 380, 381, 382, 383, 384, 386, 387, 388, 390, 391, 394, 395, 396, 397, 398, 399, 401, 402, 403, 404, 405, 406, 407, 409, 410, 411, 412, 413, 415, 416, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 431, 434, 436, 437, 438, 439, 440, 442, 443, 445, 447, 448, 450, 452, 453, 454, 455, 456, 457, 458, 459, 462, 463, 464, 465, 466, 467, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 480, 481, 482, 483, 484, 485, 486, 488, 490, 491, 493, 494, 495, 496, 497, 500, 502, 504, 505, 506, 507, 508, 510, 511, 513, 515, 518, 519, 520, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 534, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 568]

the first three samples with their features values:

```
[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.001e-01
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.534e+02
  6.399e-03 4.904e-02 5.373e-02 1.587e-02 3.003e-02 6.193e-03 2.538e+01
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.654e-01
  4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.690e-02
  7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.408e+01
  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.499e+01
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.860e-01
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.974e-01
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.403e+01
  6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.357e+01
  2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.430e-01
  3.613e-01 8.758e-02]]
```

the top 5 features from p-values of chi-squared test:

```
           features  p<0.05
13       worst area     0.0
2    mean perimeter     0.0
3         mean area     0.0
12  worst perimeter     0.0
9        area error     0.0


the top 5 features from p-values of F-test:
                features         p<0.05
22  worst concave points  1.969100e-124
17       worst perimeter  5.771397e-119
7    mean concave points  7.101150e-116
15          worst radius  8.482292e-116
2         mean perimeter  8.436251e-101


the same features from p-values of chi-squared test and F-test:
 ['mean perimeter', 'worst perimeter']
```

**References:**


https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html#sklearn.feature_selection.chi2

https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html#sklearn.feature_selection.f_classif


**(PUT YOUR ANSWER IN THE SOURCE CODE FILE)**

## PART B – DEEP LEARNING (3%)

**Note: If there is a problem installing TensorFlow at Visual Studio Code, recommend using https://colab.research.google.com/ to code for the deep learning.**

Artificial Neural Networks (ANNs) are computational systems inspired by the functioning of biological nervous systems, such as the human brain. They consist of interconnected computational nodes, or neurons, which collectively learn from input data to optimize their output. On the other hand, Convolutional Neural Networks (CNNs) are similar to traditional ANNs as they also comprise neurons that self-optimize through learning. Each neuron receives input and performs operations, such as scalar product followed by a non-linear function, from raw image vectors to the final output of the class score. The entire network expresses a single perceptive score function, with the last layer containing loss functions associated with the classes. Moreover, the tips and tricks developed for traditional ANNs are still applicable to CNNs. These characteristics highlight the fundamental principles of ANNs and CNNs, emphasizing their ability to process and learn from input data to produce meaningful outputs.

The architecture of the convolutional neural network is as follows:

- Input — height × width × depth
- Conv2D — Extract key features from images.
- MaxPooling2D — Dimensionality reduction by down-sampling
- Flatten — Flattens the input shape e.g. (4, 8) → 4 × 8 = 32
- Dropout — Prevent model overfitting.
- Dense — Fully connected layer to classify flattened input.

**Tasks:**

Write the correct Python commands in each activity below to fulfil the demonstration of CNNs model in order to get the following output.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_26 (Conv2D)          (None, 298, 298, 32)      608

 max_pooling2d_23 (MaxPoolin  (None, 149, 149, 32)     0
 g2D)

 conv2d_27 (Conv2D)          (None, 147, 147, 64)      18496

 max_pooling2d_24 (MaxPoolin  (None, 73, 73, 64)       0
 g2D)

 conv2d_28 (Conv2D)          (None, 71, 71, 96)        55392

 max_pooling2d_25 (MaxPoolin  (None, 35, 35, 96)       0
 g2D)

 conv2d_29 (Conv2D)          (None, 33, 33, 128)       110720

 max_pooling2d_26 (MaxPoolin  (None, 16, 16, 128)      0
 g2D)

 conv2d_30 (Conv2D)          (None, 14, 14, 224)       258272

 max_pooling2d_27 (MaxPoolin  (None, 7, 7, 224)        0
 g2D)

 flatten_8 (Flatten)         (None, 10976)             0

 dropout_5 (Dropout)         (None, 10976)             0

 dense_9 (Dense)             (None, 5)                 54885

=================================================================
Total params: 498,373
Trainable params: 498,373
Non-trainable params: 0
_________________________________________________________________
```

**[PROCESS]**

1. Import the libraries required as follows:

   import tensorflow as tf

   from tensorflow import keras

   from keras.models import Sequential

   from keras.layers import Dense, Flatten, Dropout

   from keras.layers.convolutional import Conv2D, MaxPooling2D

2. Create a sequential model.
3. Add a first convolutional layer with Conv2D() with Rectified Linear Unit (ReLU) activation function and 32 – 3 × 3 Filter. Also, include the input shape with 300 × 300 × 2. The ReLU activation function has proven to be more effective than the widely used logistic sigmoid function.
4. Add the max-pooling layer with MaxPooling2D() with 2 × 2.
5. Add the second convolutional layer with Conv2D() with ReLU activation function and 64 – 3 × 3 Filter.
6. Add the max-pooling layer with MaxPooling2D() with 2 × 2.
7. Add the third convolutional layer with Conv2D() with ReLU activation function and 96 – 3 × 3 Filter.
8. Add the max-pooling layer with MaxPooling2D() with 2 × 2.
9. Add the fourth convolutional layer with Conv2D() with ReLU activation function and 128 – 3 × 3 Filter.
10. Add the max-pooling layer with MaxPooling2D() with 2 × 2.
11. Add the fifth convolutional layer with Conv2D() with ReLU activation function and 224 – 3 × 3 Filter.
12. Add the max-pooling layer with MaxPooling2D() with 2 × 2.
13. Add the Flatten layer.
14. Add the Dropout layer with 0.5.
15. Add the last layer is a Dense layer that has a SoftMax activation function with 5 units, which is needed for this multi-class classification problem.
16. Show model summary.

References:

https://www.tensorflow.org/tutorials/images/cnn#create_the_convolutional_base
https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D
https://www.datacamp.com/tutorial/convolutional-neural-networks-python#the-network

**(PUT YOUR ANSWER IN THE SOURCE CODE FILE)**

## PART C – IDENTIFICATION OF TANDEM REPEATS (3%)

The enumeration of the occurrences of 'A', 'C', 'G', and 'T' in a DNA sequence constitutes a fundamental step in DNA sequence analysis, offering insights into the structure, function, and evolutionary aspects of genetic material. This process is an essential component of various bioinformatics and molecular biology applications. An illustration of identifying repeating patterns is the identification of tandem repeats. Short tandem repeats (STRs), also known as microsatellites, are repetitive DNA sequences characterized by short motifs, typically consisting of 1-6 base pairs, repeated in tandem. These microsatellites exhibit variability in the number of repeats among individuals, thereby conferring value for applications such as DNA fingerprinting and forensic analysis.

**[INPUT]** Three small part of the target DNA sequences from Human alpha-1 type XI collagen (COL11A1) mRNA (*COL11A1*) is provided as follows:

| |
|---|
| Part 1: **tttaga** |
| Part 2: **ttcgtg** |
| Part 3: **ttgtga** |

**[PROCESS]** The concepts used in Question 2 must include **CONTROL STRUCTURES**, **STRING**, and **FUNCTION**.

**Task 1:**

Write a Python code function to count the number of nucleotides in a DNA sequence, specifically the symbols **'A', 'C', 'G', and 'T'** (in order) to analyze the target DNA sequence obtained from *Homo sapiens* through GenBank (J04177.1).

**[OUTPUT]**

| |
|---|
| Part 1 Counts: {'A': 2, 'C': 0, 'G': 1, 'T': 3} |
| Part 2 Counts: {'A': 0, 'C': 1, 'G': 2, 'T': 3} |
| Part 3 Counts: {'A': 1, 'C': 0, 'G': 2, 'T': 3} |

(**PUT YOUR ANSWER IN THE SOURCE CODE FILE**)

**Task 2:**

Write a Python code function to compare microsatellite profiles based on the number of repeats in each part of the DNA sequence from three small parts of the target DNA sequence obtained from *Homo sapiens* through GenBank (J04177.1).

**Consider a single microsatellite motif "TT" for the comparison.**

Conditions:

- One repeat of {microsatellite_motif}: {Which Profile} has the most repeats of {microsatellite_motif}.
- At least two repeats of {microsatellite_motif}: {Which Profiles} have the same most repeats of {microsatellite_motif}.
- Otherwise: No profile has any repeats of {microsatellite_motif}.

**[OUTPUT]**

| |
|---|
| Microsatellite Profile Comparison Result:<br>The profiles Part 1, Part 2, Part 3 have the same most repeats of TT. |

(**PUT YOUR ANSWER IN THE SOURCE CODE FILE**)

## PART D – REPRESENTATION OF YOUR GROUP PROJECT DATA (3%)

Instruction: Attach your figure to the submission.