

USER GUIDE TO THE METHOD IN THE PAPER: MACHINE LEARNING APPROACH ON BREAST CANCER SUBTYPE CLASSIFICATION TARGETING KEY GENES IDENTIFICATION BASED ON RNA-SEQ DATA

STEP 1 : Data Cleaning

1. The first important step to be done is data cleaning. At this stage, the cleaning process is carried out on two main datasets, namely RNA-seq data (see **Supplementary_Table_1**) and clinical data (see **Supplementary_Table_2**). RNA-seq data cleaning is carried out to clean the dataset from missing values or unknown gene data. Meanwhile, in clinical data, the cleaning process is carried out by removing patient samples that have unknown subtypes and samples with the BRCA_Normal label, because they are not included in the breast cancer subtype category analyzed in this study.
2. After the RNA-seq data and clinical data have been cleaned, the next step is to align the two types of data to only include the same patient samples. This is important to ensure that the gene expression data (RNA-seq) analyzed has valid clinical subtype information. This alignment process is done by matching the patient ID from the clinical data with the column ID in the RNA data.

```
[ ] ID_subtype = df_klinis_clean["PATIENT_ID"].tolist()

[ ] #Used to match the front ID in clinical data with that in RNA data.
    patient_prefixes = set([pid[:12] for pid in ID_subtype])

[ ] metadata_columns = ['Hugo_Symbol']
    rna_columns = [col for col in df_RNA_cleaned.columns if col[:12] in patient_prefixes]

[ ] rna_filter_ID = df_RNA_cleaned[metadata_columns + rna_columns]

[ ] rna_filter_ID
```

The final output of this process is a subset of RNA-seq data that has been filtered to include only patients with valid breast cancer subtypes: BRCA_Basal, BRCA_HER2, BRCA_LumA, and BRCA_LumB. These subtypes are also documented in the clinical data (see **Supplementary_Table_3**). This aligned data will subsequently undergo a normalization process for further analysis. To perform this process, please use the **data_preprocessing.py** script in the data cleaning section.

STEP 2 : Data Normalization

The next step is to normalize the RNA-seq data using DESeq2. At this stage, there are two main processes that are carried out simultaneously, namely normalization using size factor normalization and Differential Expression Analysis (DEA). These two processes are carried out simultaneously because, by default, DESeq2 applies normalization first before performing differential expression analysis. To do this step, use the **DESeq2_normalization.py** script.

```
with ro.pandas2ri.converter.context():
    # Convert pandas DataFrame to R format using rpy2
    r_count_data = ro.pandas2ri.py2rpy(count_data)
    r_col_data = ro.pandas2ri.py2rpy(col_data)

    # Running DESeq2 in R via rpy2
    ro.r('''
        library(DESeq2)

        # count_data <- as.data.frame(count_data)
        # rownames(count_data) <- count_data$Hugo_Symbol

        dds <- DESeqDataSetFromMatrix(countData = round(%s), colData = %s, design = ~ Subtype)
        dds <- DESeq(dds)
        res <- results(dds)

        res_data <- as.data.frame(res)
        normalized_counts <- counts(dds, normalized = TRUE)

        normalized_counts_df <- as.data.frame(normalized_counts)
        ''' % (r_count_data.r_repr(), r_col_data.r_repr()))

    # Results of differential expression analysis
    res_df = ro.r('res_data')

    # Normalization results
    normalized_counts_df = ro.r('normalized_counts_df')
```

NB: At this stage, *count_data* refers to the cleaned and aligned RNA-seq data (see *Supplementary_Table_3*), while *col_data* contains patient ID information along with breast cancer subtype labels (see *Supplementary_Table_4*).

- **The results of the RNA-seq data normalization process will look like the image below:**

	TCGA-3C- AAAU-01	TCGA-3C- AALI-01	TCGA-3C- AALJ-01	TCGA-3C- AALK-01	TCGA-4H- AAAK-01	TCGA-5L- AAT0-01
1	12.615548	16.761288	8.925467	10.723870	13.806813	13.163633
2	50.462190	69.017070	152.724650	140.385202	82.840877	107.189587
3	0.000000	0.985958	0.000000	0.000000	0.000000	0.000000
4	0.000000	0.985958	0.000000	0.000000	0.000000	0.000000
5	1.940853	141.977973	0.000000	0.000000	0.000000	0.000000
...

- *The Differential Expression Analysis (DEA) process generates values for each gene, including baseMean, log2FoldChange, lfcSE, stat, p-value, and padj, as shown in the image below:*

	baseMean	log2FoldChange	lfcSE	stat	pvalue	padj
1	7.602400	-0.398812	0.102724	-3.882354	1.034502e-04	1.843706e-04
2	114.894678	-0.421191	0.060089	-7.009490	2.391887e-12	7.078077e-12
3	0.454947	0.275471	0.230491	1.195150	2.320284e-01	2.872963e-01
4	0.058038	-0.278620	2.173136	-0.128211	8.979819e-01	9.611395e-01
5	3.224059	-3.152354	0.508537	-6.198873	5.686902e-10	1.466940e-09


The results of data normalization are stored in (**Supplementary_Table_5**) and used for further analysis, including model training. Meanwhile, the results of differential gene expression analysis (DEA) are stored in (**Supplementary_Table_6**) and used in the selection and analysis of important genes.

STEP 3 : Data Selection

This stage aims to select genes that will be used as input for further analysis. Selection is done by filtering genes in (**Supplementary_Table_5**) based on the baseMean value, which is the average level of gene expression across all samples. Only genes with *baseMean* ≥ 10 are retained, as they are considered to have sufficient expression levels for further analysis. This baseMean value is obtained from the results of differential expression analysis using DESeq2 in the previous stage (**Supplementary_Table_6**). To perform this process, please use the **data_preprocessing.py** script in the data selection section.

```
[22] #Filter genes using basemean parameter >= 10
      filter_gen = DEA_result[DEA_result["baseMean"] >= 10]["Hugo_Symbol"]
```

```
[23] filter_data_RNA = RNA_normalized[RNA_normalized["Hugo_Symbol"].isin(filter_gen)]
      number_of_gene = len(filter_data_RNA)
      print("Number of genes with baseMean >= 10:", number_of_gene)
```

 Number of genes with baseMean >= 10: 15751

After the data selection process with the basemean parameter set ≥ 10 , the number of genes was reduced from 20,518 to 15,751. This total gene is the one used in the classification modeling process later (see **Supplementary_Table_7**).

STEP 4 : Data Labeling

Before the labeling process, the gene selection dataset (see **Supplementary_Table_7**) was first transformed so that its structure was suitable for classification analysis. Initially, the data structure was in wide format, where each row represents one gene (Hugo_Symbol) and each column represents a patient sample, as seen in the figure below.

	Hugo_Symbol	TCGA-3C- AAAU-01	TCGA-3C- AALI-01	TCGA-3C- AALJ-01	TCGA-3C- AALK-01	TCGA-4H- AAAK-01
0	HMGB1P1	50.462190	69.017070	152.724650	140.385202	82.840877
1	HSPB1P1	281.423753	443.681165	1091.882079	395.808278	91.716686
2	GTPBP6	444.455445	674.395370	651.559061	767.244126	696.257850
3	EFCAB12	39.787496	29.578744	22.809526	52.644451	53.254850
4	A1BG	191.174067	233.672080	419.496930	186.205372	265.288048

Transpose is done by swapping the positions of rows and columns, so that each row represents a patient sample, while each column represents a gene, as seen in the figure below. This process is done to ensure that the data fits the format required by the classification algorithm, where each patient sample has a set of features in the form of gene expression values.

	0	1	2	3	4	5
Id_Sample	HMGB1P1	HSPB1P1	GTPBP6	EFCAB12	A1BG	A2LD1
TCGA-3C- AAAU-01	50.46219	281.423753	444.455445	39.787496	191.174067	99.953954
TCGA-3C- AALI-01	69.01707	443.681165	674.39537	29.578744	233.67208	70.003028
TCGA-3C- AALJ-01	152.72465	1091.882079	651.559061	22.809526	419.49693	159.66668
TCGA-3C- AALK-01	140.385202	395.808278	767.244126	52.644451	186.205372	61.418526

After the data is transposed, the labeling process is performed by matching the Patient_ID from the RNA-seq data (see **Supplementary_Table_8**) with the filtered clinical data. In this step, the clinical dataset is refined to include only the relevant columns: PATIENT_ID and SUBTYPE, using the following filter:

```
# Extracting the patient_id and subtype columns from clinical data
df_klinis_clean = df_klinis_clean[["PATIENT_ID", "SUBTYPE"]]
```

```
#Match patient_id from clinical data with sample_id in RNA data
RNA_transpose["PATIENT_ID"] = RNA_transpose["Id_Sample"].str[:12]
```

Each patient sample is then labeled according to the breast cancer subtype listed in the clinical data. The final output of this process is a structured dataset where each row contains gene expression values and a corresponding subtype label, as shown in the figure below. This labeled dataset is stored in **Supplementary_Table_9**. To perform this process, please use the **data_preprocessing.py** script, specifically the section for data labeling.

	Id_Sample	HMGB1P1	HSPB1P1	GTPBP6	EFCAB12	SUBTYPE
0	TCGA-3C-AAAU-01	50.462190	281.423753	444.455445	39.787496	BRCA_LumA
1	TCGA-3C-AALI-01	69.017070	443.681165	674.395370	29.578744	BRCA_Her2
2	TCGA-3C-AALJ-01	152.724650	1091.882079	651.559061	22.809526	BRCA_LumB
3	TCGA-3C-AALK-01	140.385202	395.808278	767.244126	52.644451	BRCA_LumA
4	TCGA-4H-AAAK-01	82.840877	91.716686	696.257850	53.254850	BRCA_LumA

STEP 5 : Data Transformation

Since the dataset used (see **Supplementary_Table_9**) contains categorical data, namely breast cancer subtype labels, a data transformation process is required. This transformation is carried out using label encoding techniques to convert subtype labels into a numeric format that can be processed by machine learning algorithms. To perform this process, please use the **data_preprocessing.py** script in the **data transformation** section.

	Id_Sample	HMGB1P1	HSPB1P1	GTPBP6	EFCAB12	SUBTYPE
0	TCGA-3C-AAAU-01	50.462190	281.423753	444.455445	39.787496	2
1	TCGA-3C-AALI-01	69.017070	443.681165	674.395370	29.578744	1
2	TCGA-3C-AALJ-01	152.724650	1091.882079	651.559061	22.809526	3
3	TCGA-3C-AALK-01	140.385202	395.808278	767.244126	52.644451	2
4	TCGA-4H-AAAK-01	82.840877	91.716686	696.257850	53.254850	2

Label Encoding Description:

- 0 = BRCA_Basal
- 1 = BRCA_Her2
- 2 = BRCA_LumA
- 3 = BRCA_LumB

The final results of this process are stored in **Supplementary_Table_10**, and will be used as input data for the classification model in the next stage.

STEP 6 : Modeling

At this stage, the modeling process is carried out using three main algorithms, namely Random Forest, Gradient Boosting, and XGBoost. All three models are built with the same approach, namely the One-vs-Rest technique to handle multiclass classification.

The data used as input is data that has gone through all stages of preprocessing and labeling, as stored in **Supplementary_Table_10**. This data is divided into a proportion of 80% for training and 20% for testing. The training and evaluation process of the model is carried out using the **classification_model.py** script.

STEP 7 : Identification of Key Genes

Key gene identification was performed using the feature importance approach, which aims to measure the extent to which each feature (in this case, gene) contributes to the model's prediction results. These feature importance values are generated from three previously trained classification models: Random Forest, Gradient Boosting, and XGBoost.

To run this process, users can use the **classification_model.py script in the feature importance section** . The results of this approach will display the weight or level of influence of each gene on the prediction of each breast cancer subtype, which is visualized in the image below.

	0	1	2	3
Gene				
FOXA1	0.398165	0.000000	0.000000	0.000000
MLPH	0.051657	0.000000	0.000000	0.000000
ABLIM3	0.036689	0.000041	0.000071	0.000000
FOXC1	0.035421	0.006657	0.000449	0.001565
FAM69C	0.030887	0.000000	0.000000	0.000000
...

A list of key genes generated from each model can be seen in **Supplementary_Table_11**.