REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

META ANALYSIS AND MSS/MSI CLASSIFICATION ON COLORECTAL CANCER TUMOR SPECIMENS

19011070 – Ali Emre Nebiler

SENIOR PROJECT

Advisor
Res.Asst. Sultan Sevgi TURGUT

May, 2023

ACKNOWLEDGEMENTS

I would like to acknowledge and give my warmest thanks to my supervisor Res.Asst. Sultan Sevgi TURGUT who made this work possible. Her help and advices carried me through all the stages of writing my project. I am grateful for every brilliant comment and suggestion which carried out this project, thanks to you. Your help was what sustained me this far.

I would also like to give special thanks to my advisor Lect.PhD Ahmet ELBİR for his guidance before starting the research. Without your support, I could not have started the research and came to this point.

Finally, I would like to thank Yıldız Technical University and its Rectorate, for providing all material and moral opportunities for the use of all students.

Ali Emre Nebiler

TABLE OF CONTENTS

LIST OF SYMBOLS

%	percent
s, sec	second

LIST OF ABBREVIATIONS

CRC	Colorectal Cancer
AI	Artificial Intelligence
ML	Machine Learning
CNN	Convolutional Neural Network
SVM	Support Vector Machine
k-NN	K-Nearest Neighbors
NCBI	National Center for Biotechnology Information
GEO	Gene Expression Omnibus
DGE	Differential Gene Expression
KVKK	Personal Data Protection Authority
IDE	Integrated Development Environment
MSI	Microsatellite Instability
MSS	Microsatellite Stability
MMR	Mismatch Repair
MRI	Magnetic Resonance Imaging
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve
GDA	General Data Anonymization

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Meta Analysis and MSS/MSI Classification on Colorectal Cancer Tumor Specimens

Ali Emre Nebiler

Department of Computer Engineering
Senior Project

Advisor: Res.Asst. Sultan Sevgi TURGUT

Cancer is a disease that is seen in millions of people worldwide and whose definitive treatment has not been found yet. In addition to the increasing number of patients reaching millions, the risk of cancer in new generations is also increasing. There are more than 200 types of cancer, the third most common type is colorectal cancer.

Although there is no definite cure, there are various treatments for cancer. Beside treatment, early diagnosis is very important for such disease. It can be said that most of the diagnose methods are expensive. Apart from standard methods, the development of software technologies and artificial intelligence has begun to show its effect in medical field. Instead of classic tests and examinations, it became possible to produce predictions and classify different diseases by analyzing the existing samples.

In this project, considering the possibilities offered by technology and the demand of the medical field, it was desired to develop a cancer classification software which is cheaper, based on machine learning and classification methods. By using various cancer samples collected from different patients, models were trained with six different classification methods and the prediction results of each model were compared. In addition, the effect of different genes on colorectal cancer was also examined by Survival Analysis.

Keywords: Cancer, gene, classification, machine learning, analysis

ÖZET

Kolorektal Kanser Tümör Örnekleri Üzerinde Meta Analizi ve MSS/MSI Sınıflandırması

Ali Emre Nebiler

Bilgisayar Mühendisliği Bölümü
Bitirme Projesi

Danışman: Arş.Gör. Sultan Sevgi TURGUT

Günümüzde kanser, dünyada milyonlarca insanda görülen ve kesin tedavisi henüz bulunamamış bir hastalıktır. Milyonları bulan hasta sayısının gittikçe artmasının yanı sıra, yeni nesillerdeki kanser riski de gittikçe artmaktadır. Kanserin 200'den fazla türü bulunmaktadır, bunlardan en sık rastlanan 3. türü de kolorektal kanserdir.

Kesin bir tedavisi olmamakla birlikte, kanser için uygulanan ve sonuç alınan çeşitli tedaviler mevcuttur. Tedavinin yanı sıra, kanserde erken tetkik çok önemlidir. Tetkik için uygulanan metotların bir çoğu pahalı olarak nitelendirilebilir. Standart yöntemlerin dışında, yazılım teknolojileri ve yapay zekanın gelişimi de sağlık alanında etkisini göstermeye başlamıştır. Klasik testler ve incelemeler yerine, bu zamana kadar elde edilen örneklerin analizi ile farklı hastalıklar için tahmin üretmek ve sınıflandırma yapmak mümkün hale gelmiştir.

Bu projede teknolojinin sunduğu olanaklar ve sağlık sektörünün talebi doğrultusunda, daha az maliyetli ve makine öğrenmesi ile sınıflandırma metotlarına dayanan bir kanser sınıflandırma yazılımı geliştirilmek istenmiştir. Bu zamana kadar farklı hastalardan toplanmış çeşitli kanser örnekleri kullanılarak, altı farklı sınıflandırma yöntemi ile model eğitimi yapılmış ve her modelin tahmin sonuçları karşılaştırılmıştır. Ayrıca işlenen kanser örneklerinden elde edilen farklı genlerin, kolorektal kanser üzerindeki etkisi de Sağkalım Analizi ile incelenmiştir.

Anahtar Kelimeler: Kanser, gen, sınıflandırma, makine öğrenmesi, analiz

1 Introduction

1.1 Project Motivation

CRC is the third most common cancer diagnosed in both men and women in the country excluding skin cancers, according to analysis in United States [1]. The recovery period of such a disease is so long that early diagnosis is very important.

Therefore, the motivation of this project is to analyze available samples, detect any anomaly and fasten the examination process of CRC.

1.2 Project Goal

The main project goals are to analyze available CRC datasets, train an model with different machine learning classification methods which can classify samples accurately, and find new survival markers (biomarkers) to contribute to the diagnosis.

During the development, it was focused on to find the best method among different binary classification methods. Samples will be classified with their gene expressions and MSI/MMR statuses. The MSI/MMR status indicates the mutation rate of the repetitive DNA, which gives an idea about the possibility of a cancer.

1.3 Project Scope

The project is especially focused on CRC, not the other cancer types. The project's main scope is to use/combine existing sample datasets, develop a classification project with R and Python languages and find new possible survival markers.

1.4 Information About Sections

The project will be explained in more detail in the next sections.

Preliminary Examination: Similar studies in this field will be examined and the differences between these studies and this project will be explained.

Feasibility: The feasibility of the project and the reasons for choosing the resources used will be explained, and the economic time planning of the project will be examined.

System Analysis: The requirements and resources of the project will be determined, and the most appropriate solution will be finalized with diagrams for the design phase of the application.

System Design: Software, database and input-output designs will be examined.

Implementation: Intermediate outputs and final output obtained while the project is running will be shown with screenshots.

Experimental Results: Results obtained with different parameters, methods and datasets will be shown.

Performance Analysis: Experimental results will be compared and interpreted.

Conclusion: All project will be summarized, suggestions for future studies and final success will be shared.

2 Preliminary Examination

2.1 Similar Studies

In a similar study, a CNN-based approach was evaluated to automatically differentiate healthy tissues and tubulovillous adenomas from cancerous samples [2]. Another study's aim was to determine models for CRC identification that involve minimally invasive, affordable, portable, and accurate screening variables, including laboratory detection data [3]. In another study, an algorithm was developed for predicting MSI from the expression profiling of a gene panel in cancer [4]. The studies were mainly focused on gene expressions and sample MSI/MMR statuses, similar to this research.

2.2 Necessity

There are many diagnose methods for cancer. But still, classification with machine learning is not considered as successful as the other methods. Besides, some suggested diagnose methods (like MRI) are expensive in general [5]. As a result, machine learning success must be increased in cancer classification for the necessity of affordable diagnose methods.

2.3 Differences

The classification of MSI/MMR will be held as a Meta Analysis. Different from the other projects, the discovery of new survival markers will be attempted with Survival Analysis. The gene-disease relationships of the found survival markers will be investigated on the DisGeNet platform to confirm whether they are associated with CRC.

2.4 Methods

It was decided to use GEO database store from NCBI, a well known biotechnology information source, to get datasets. The main reason for this is that it is a widespread and extensive dataset resource. In GEO, the datasets of "GPL570" platform is selected to be used in this project, which is a complete coverage of the Human Genome U133 Set plus 6,500 additional genes for analysis of over 47,000 transcripts [6].

R language will be used to merge different datasets, perform Differential Gene Expression (DGE) and Survival Analysis on the merged dataset. R was prefered because it is a language and environment for statistical computing and graphics [7], which is suitable for the need.

Python will be used to perform binary classification on the pre-processed and merged dataset because of its wide range of library options for machine learning.

3 Feasibility

3.1 Technical Feasibility

3.1.1 Feasibility Of Software

As mentioned before in 2.4, R and Python languages were selected to use in the project. Since both languages' syntax is similar, it is considered to be easy to develop the project in two different languages.

R includes many features like an effective data handling and storage facility, a suite of operators for calculations on arrays and matrices, an integrated collection of intermediate tools for data analysis [7]. It is sufficient to manipulate multiple datasets. Python has a wide range of libraries. It has the capacity to run binary classification methods on the merged dataset.

CSV type was prefered for the storage of datasets. It has a simple structure and it is easy to work on. Both Python and R have the necessary libraries to read and write to CSV files.

3.1.2 Feasibility Of Equipment

During the development, Visual Studio Code for Python and RStudio for R were chosen as IDE, for compatibility reasons. The project will be developed in a device with MacOS, M1 chip and 16GB of RAM. Nevertheless, it can be run on a a device with Windows or Linux device with at least 8GB of RAM, as long as the environments were installed. 5GB of free storage space is enough for all processes and intermediate/final outputs.

3.1.3 Feasibility Of Communication

GEO datasets are available online and can be accessed with an R library called "GEOquery". Internet connection is neccessary while getting the datasets to the local device. For other processes, internet connection is not needed, locally stored CSV files will be used.

3.2 Time Planning

Considering the importance of the accuracy of the data, necessary time has been calculated and the time planning was completed, as shown in 3.1.

Table 3.1 Gantt Diagram

Tasks			*Months*			
Name	**Start**	**End**	**March**	**April**	**May**	**June**
Research About The Topic	10.03.2023	12.03.2023	▒			
Planning	13.03.2023	19.03.2023	▓			
Finding Datasets	13.03.2023	19.03.2023	▒			
Merging Datasets	20.03.2023	02.04.2023	▓			
DGE On Merged Dataset	27.03.2023	16.04.2023	▒	▒		
Training & Testing	17.04.2023	15.05.2023		▓	▓	
Fine-Tuning	01.05.2023	14.05.2023			▒	
Survival Analysis	15.05.2023	28.05.2023			▓	
DisGeNet Verification	22.05.2023	28.05.2023			▒	
Reporting	29.05.2023	04.06.2023				▓

3.3 Legal Feasibility

Since every sample data in the dataset is a personal data, confidentiality of the samples must be considered. As in many countries, Turkiye also has a regulation about personal data, which is coordinated by Personal Data Protection Authority (KVKK) [8].

According to KVKK, any health data is also an personal data, and must be protected. Without the related person's permission, the data cannot be shared or used completely in any field. The data can be used only if its privacy can be provided anonymously. In GEO datasets, every sample has an anonymous name starting with "GSM", which is identifying it as a GEO sample. Therefore, the usage of these datasets is legal.

3.4 Economic Feasibility

All datasets, IDEs and languages are free to use. There were no expense for this project.

To diagnose CRC, there are many tests like Blood Test, Diagnostic Colonoscopy, Proctoscopy, Biopsy and Imaging Tests. MSI/MMR Testing, whose results were used in this project, is a lab test of Biopsy samples. Furthermore, MRI is a type of Imaging Test [9]. When comparing MRI and Biopsy, MRI costs twice as much as Biopsy (approx. \$2,300 vs. \$1,150 in U.S. [10]). The success of the project can save up to \$1,000 for per test, which can lead to \$150,000,000 per year (only for U.S.) considering approx. 150,000 new cases [1].

4
System Analysis

4.1 Performance Metrics

- Accuracy
- Precision
- Recall
- F1 Score
- P-Value (in Survival Analysis)

4.2 Measurable Goals

The AI model's score must be minimum 80% when testing with an external dataset. With Survival Analysis and DisGeNet search, it is aimed to find at least one new survival marker associated with CRC.

4.3 Requirement & Available Resources

- **CRC Datasets:** GEO database store from NCBI will be used to get datasets.
- **Pre-Processes And DGE Analysis:** R language was selected to merge datasets and perform DGE, because it works fast on large datasets.
- **Classification With Machine Learning:** Python is a great choice to train an ML model with binary classification. It has lots of libraries and enough memory capacity to store great amount of data.
- **IDE:** VSCode and RStudio are compatible with the related languages and free to use.
- **Computer:** A computer which is compatible with R and Python is required. A MacOS device with M1 chip and 16GB of RAM is accessible to develop the project.

4.4 Modules

- **Dataset Merging Module:** It is an R module. It eliminates and merges genes in different datasets. It does this both for expressions data and metadata.
- **Remove Batch-Effect From Dataset Module:** It is an R module. After merging, expression ranges differ for each merged dataset. It calculates the difference and equalize the expression values in the same range.
- **DGE Analysis Module:** It is an R module. It follows the DGE pipeline and calculates up and down genes, required to train a model.
- **Classification With Machine Learning Module:** It is a Python module. It trains the model and test it with different binary classification methods. It will be improved with fine-tuning.
- **Survival Analysis Module:** It is a R module. It tries to find new potential survival markers.

4.5 Work-Flow

The work-flow includes three main parts, Feature Selection, Training & Testing and Survival Analysis, as shown in 4.1.

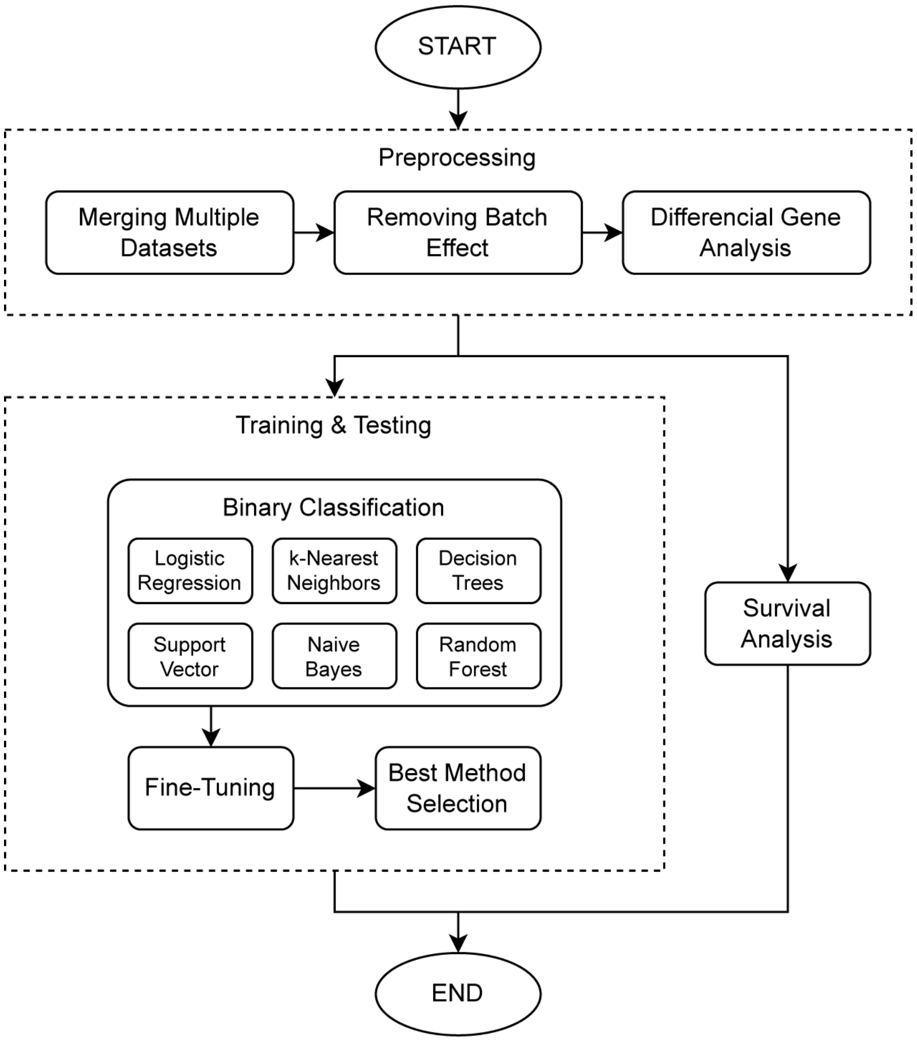


Figure 4.1 Work-Flow Diagram

4.6 Data-Flow

There are two main usage for this project. The first one is to train the model with given datasets, and the second one is to find new potential survival markers with Survival Analysis, as shown in 4.2 and detailed in 4.3. The sub-modules of the Preprocessing, Training & Testing and Survival Analysis modules were shown in 4.4, 4.5 and 4.6.

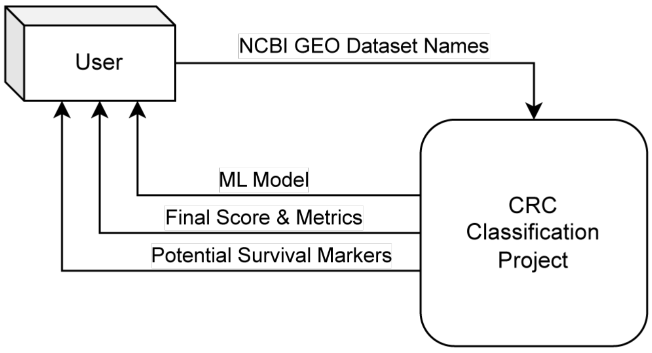


Figure 4.2 Data-Flow Diagram - Level 0

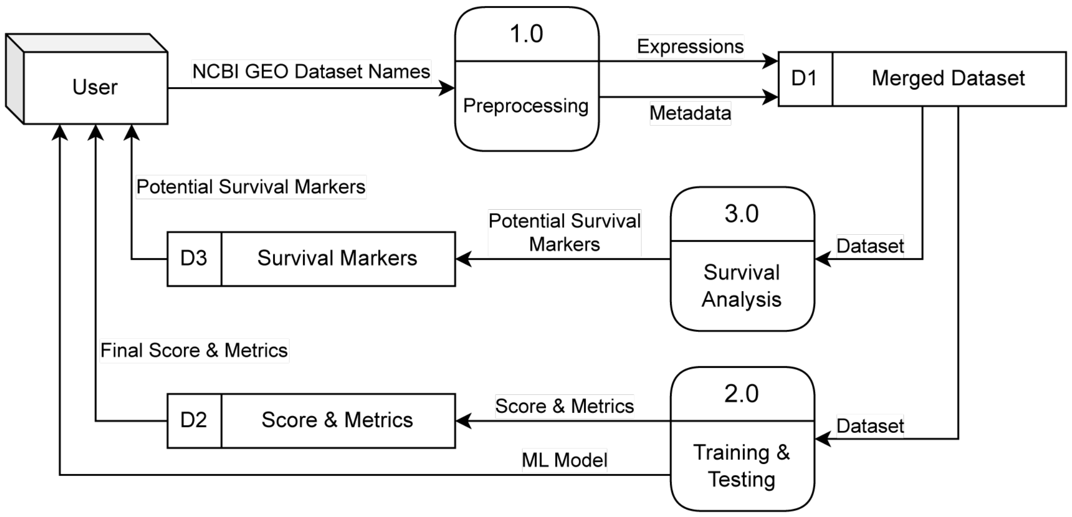


Figure 4.3 Data-Flow Diagram - Level 1

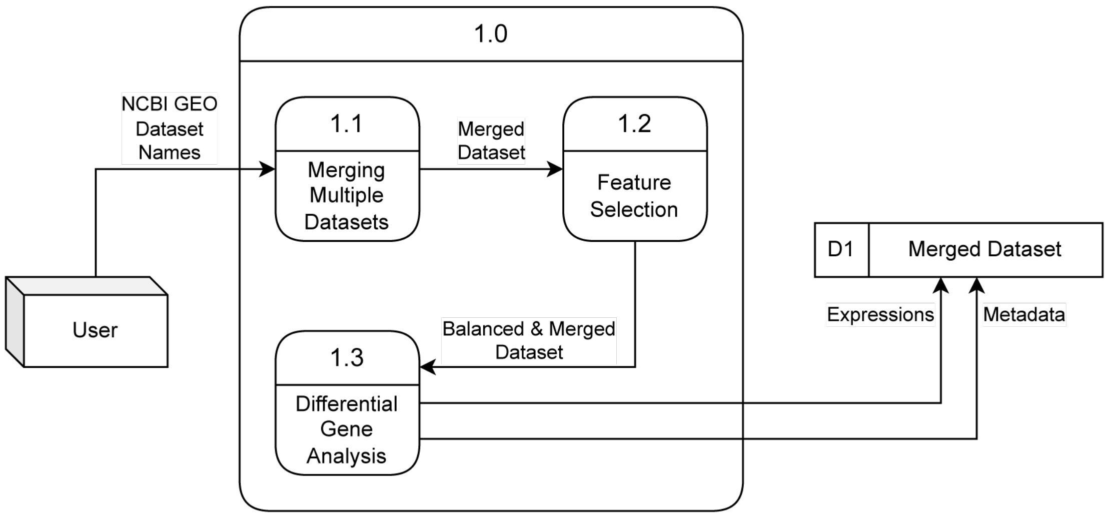


Figure 4.4 Data-Flow Diagram - Level 2 - Preprocessing

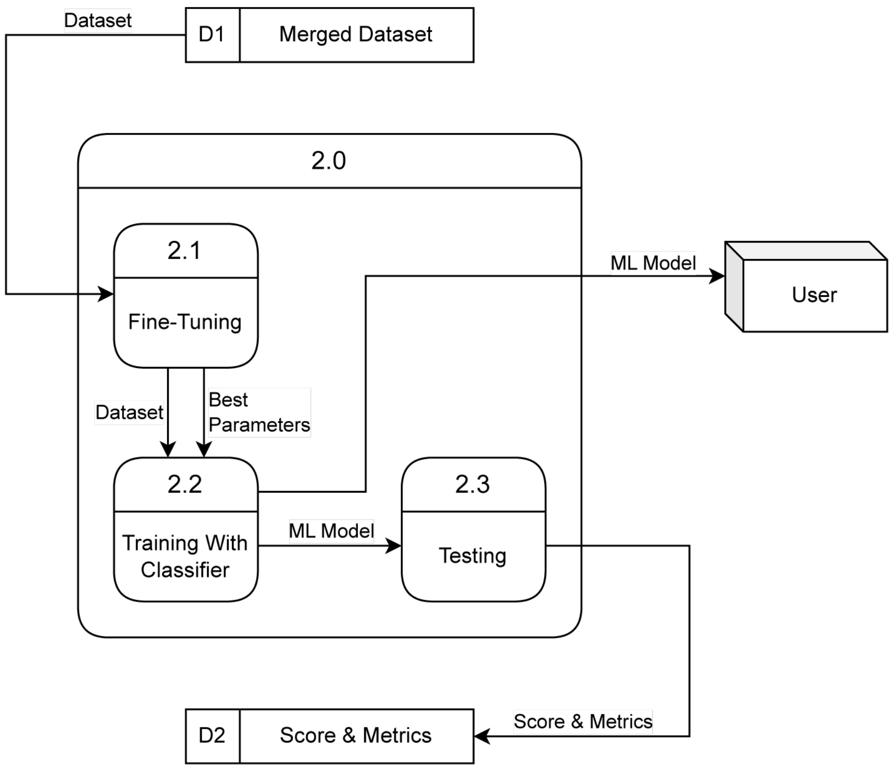


Figure 4.5 Data-Flow Diagram - Level 2 - Training & Testing

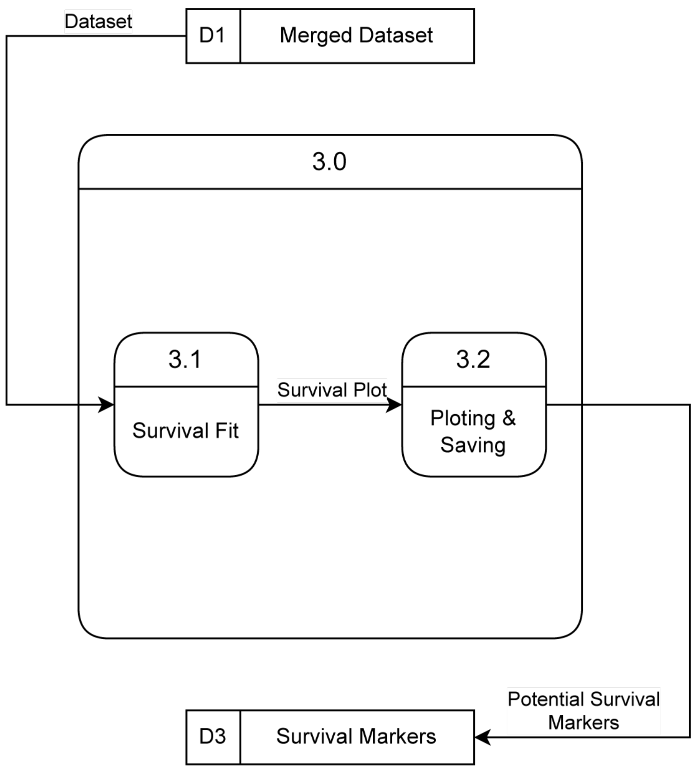


Figure 4.6 Data-Flow Diagram - Level 2 - Survival Analysis

5 System Design

5.1 Design Of Software

The modules were designed as shown in the block diagram 4.1. Each module was explained below.

5.1.1 Preprocesses

This module was developed with R. It merges multiple datasets into a single dataset, with only necessary genes. To decrease gene amount, only the joint genes were used and genes without external names were ignored. However, since the gene number was still large, DGE was performed.

– **Differential Gene Expression (DGE) Analysis:** It is the process by which information from a gene is used in the synthesis of a functional gene product, which may be proteins. A gene is declared differentially expressed if an observed difference or change in read counts or expression levels between two experimental conditions is statistically significant [11].

5.1.2 Training and Testing

This module was developed with Python. To find best hyperparameters for each classification method, fine-tuning was performed. Shuffle Split method was preferred to obtain different train and test groups, Grid Search was performed to try different parameters.

– **Shuffle Split:** It generates a user defined number of independent train-test dataset splits. Samples are first shuffled and then split into a pair of train and test sets. It is thus a good alternative to k-fold cross validation that allows a finer control on the number of iterations and the proportion of samples on each side of the train-test split [12].

– **Grid Search:** In scikit-learn hyperparameters are passed as arguments to the constructor of the estimator classes. Grid Search method searches the hyper-parameter space for the best cross validation score [13].

Six different classification methods were performed for each train-test group.

– **Logistic Regression:** It estimates the probability of an event occurring, such as MSS or MSI-H, based on a given dataset of independent variables (in our case expression values) [14].

– **k-Nearest Neighbors:** It is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point [15].

– **Decision Trees:** It is a non-parametric supervised learning algorithm. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes [16].

– **Support Vector Machine:** It works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable [17].

– **Naive Bayes:** It is also known as a probabilistic classifier since it is based on Bayes' Theorem. This theorem, also known as Bayes' Rule, allows us to "invert" conditional probabilities [18].

– **Random Forest:** It is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems [19].

Each classifier's best parameters were found with fine-tuning and tested with internal and external data. At the end, the classification report results were obtained and compared.

5.1.3 Survival Analysis

This module was developed with R. It uses the up/downregulation, MSI/MMR and gene expression data to find potential new survival markers, which indicates that if a gene has an effect on the disease. It compares overall survival (OS) time and event data with the expressions of a gene to infer the effect of the gene. Here, the important metric is p-value. If the p-value is below the 0.05 boundary, it can be said that the gene is a potential survival marker (biomarker). The result of the Survival Analysis is a plot which shows the graph of survival probability versus OS time in days and the p-value of the gene.

5.2 Design Of Database

NCBI GEO database was used to obtain the dataset. This database stores curated gene expression datasets, as well as original series and platform records in the Gene Expression Omnibus (GEO) repository [20].

Datasets including the platform with code "GPL570" was used because it is widespread. The codes of selected CRC datasets were "GSE13067", "GSE13294", "GSE18088", "GSE26682", "GSE35896", "GSE39084", "GSE39582" and "GSE75316". From these datasets, only expression values and and MSI/MMR statuses from metadata were obtained and merged into a single dataset.

There were three implemented preprocesses. The first one changes the affymetrix human genome names with external names, which is more common in daily use. The second one removes the batch effect, this step was necessary to equalize different datasets. The third one decreases the amount of genes with DGE, which finds the upregulations and downregulations.

For last, the merged dataset was exported as two CSV files, one for expressions and the other for metadata. CSV was prefered because it has an easy syntax to store and edit.

5.3 Design Of Input-Output

The project is in general a script project, which does not include an interface. All controls are made with terminal or IDE.

Each module has a configuration part at the start of its script. These configurations are required as input and must be changed with the help of an IDE or a text editor.

Every module can be started with a terminal input or the interface of the used IDE. To understand if the in-process module is working or not, terminal outputs were added to every module. To meet the requirement, `message()` command was used for R modules and `print()` command was used for Python modules. Error messages were also created for possible exceptions during processes.

For feature selection module, the inputs are NCBI GEO dataset names. These input can be changed from the configuration part of the R script. The outputs are merged expression and metadata tables formatted as CSV file.

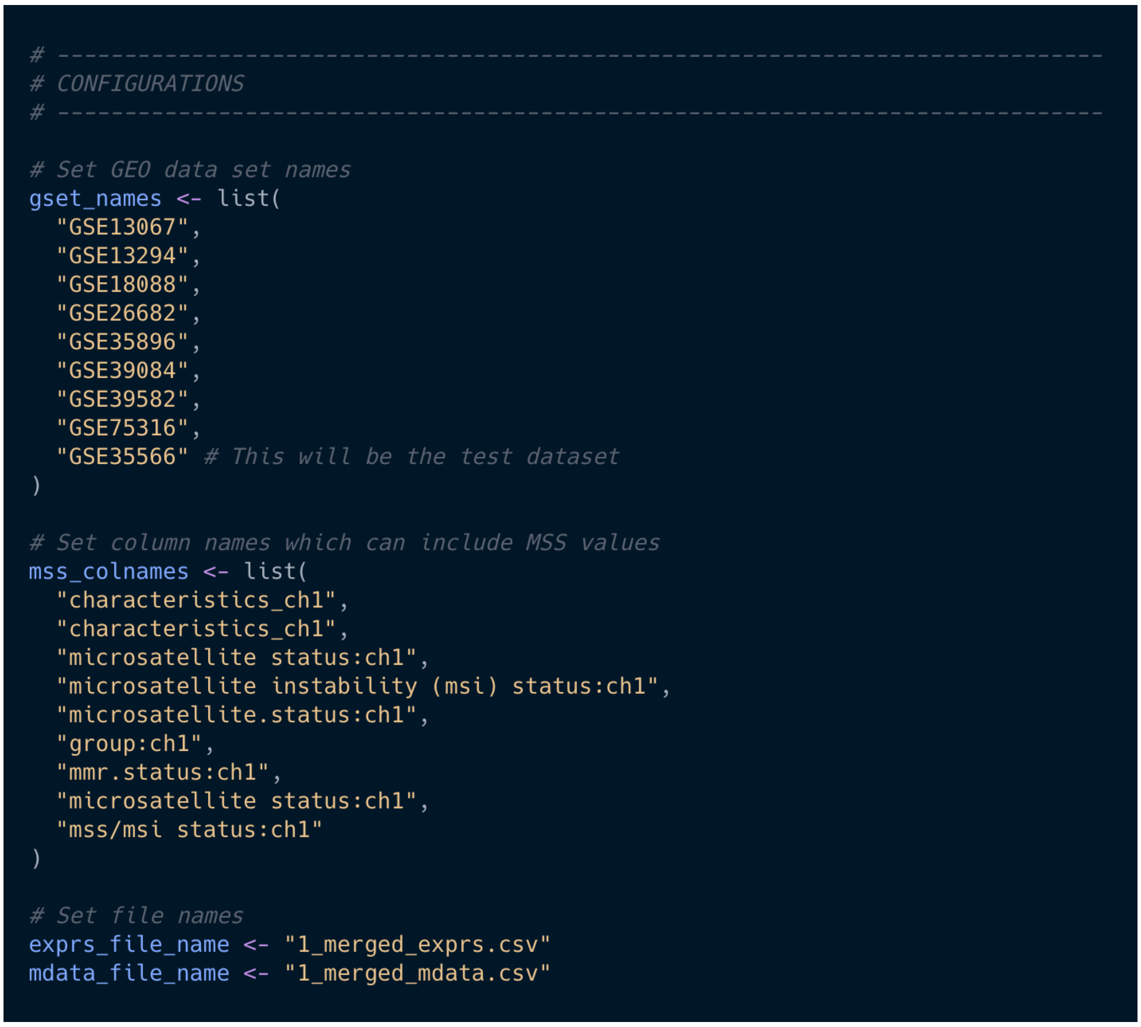
For training and testing module, this CSV files are used as the inputs. The desired input names can be changed from the configuration part of the Python script. The outputs will be the scores of the models for each classification method and the models themselves.

For survival analysis module, the inputs are selected NCBI GEO dataset name and CSV files. These inputs can be changed from the configuration part of the R script. The outputs are selected expression, metadata and gene tables as CSV and survival analysis plots formatted as SVG.

6 Implementation

6.1 Preprocessing

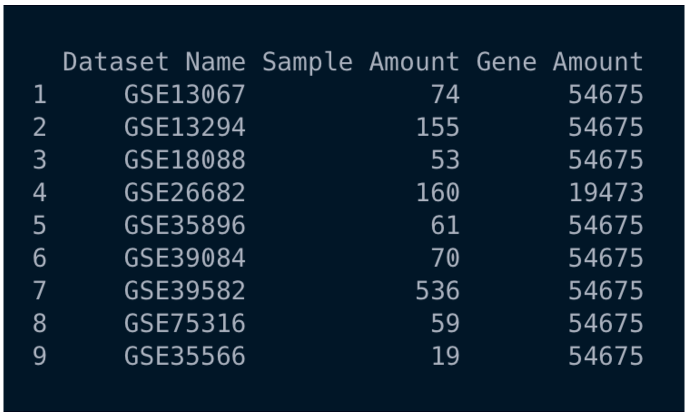
The names of the selected datasets, possible column names for MSI/MMR status and output file names were given in `1_get_and_merge_datasets.R` script's configuration as shown in 6.1.



```r
# -----------------------------------------------------------------------------
# CONFIGURATIONS
# -----------------------------------------------------------------------------

# Set GEO data set names
gset_names <- list(
  "GSE13067",
  "GSE13294",
  "GSE18088",
  "GSE26682",
  "GSE35896",
  "GSE39084",
  "GSE39582",
  "GSE75316",
  "GSE35566" # This will be the test dataset
)

# Set column names which can include MSS values
mss_colnames <- list(
  "characteristics_ch1",
  "characteristics_ch1",
  "microsatellite status:ch1",
  "microsatellite instability (msi) status:ch1",
  "microsatellite.status:ch1",
  "group:ch1",
  "mmr.status:ch1",
  "microsatellite status:ch1",
  "mss/msi status:ch1"
)

# Set file names
exprs_file_name <- "1_merged_exprs.csv"
mdata_file_name <- "1_merged_mdata.csv"
```

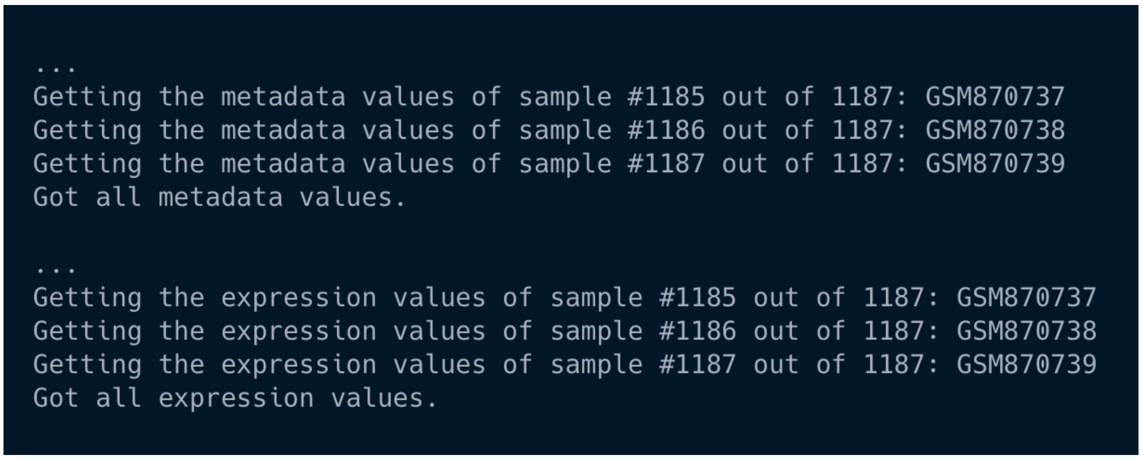
Figure 6.1 Configurations For Merging Datasets

Each dataset's size can be seen in 6.2. Genes present in each sample were selected and samples with null values were eliminated. At last, total sample count was 1187 and total gene count was 12809. As mentioned in 5.3, the progress outputs can be seen in 6.3. The merged datasets were saved as CSV, as shown in 6.4 and 6.5.



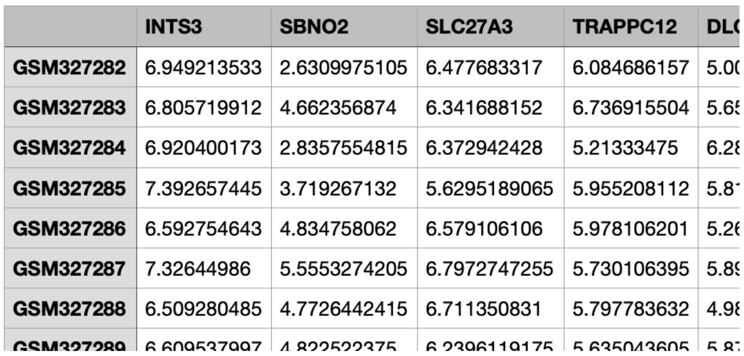
```
  Dataset Name Sample Amount Gene Amount
1     GSE13067            74       54675
2     GSE13294           155       54675
3     GSE18088            53       54675
4     GSE26682           160       19473
5     GSE35896            61       54675
6     GSE39084            70       54675
7     GSE39582           536       54675
8     GSE75316            59       54675
9     GSE35566            19       54675
```

Figure 6.2 The Sizes Of The Datasets As Output



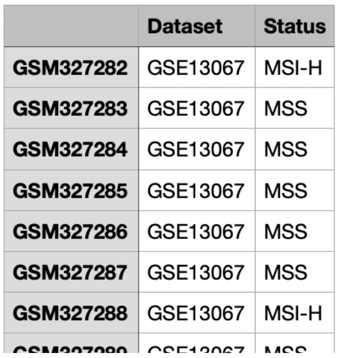
```
...
Getting the metadata values of sample #1185 out of 1187: GSM870737
Getting the metadata values of sample #1186 out of 1187: GSM870738
Getting the metadata values of sample #1187 out of 1187: GSM870739
Got all metadata values.

...
Getting the expression values of sample #1185 out of 1187: GSM870737
Getting the expression values of sample #1186 out of 1187: GSM870738
Getting the expression values of sample #1187 out of 1187: GSM870739
Got all expression values.
```

Figure 6.3 Outputs During Merging Process



	INTS3	SBNO2	SLC27A3	TRAPPC12	DL
GSM327282	6.949213533	2.6309975105	6.477683317	6.084686157	5.0
GSM327283	6.805719912	4.662356874	6.341688152	6.736915504	5.6
GSM327284	6.920400173	2.8357554815	6.372942428	5.21333475	6.2
GSM327285	7.392657445	3.719267132	5.6295189065	5.955208112	5.8
GSM327286	6.592754643	4.834758062	6.579106106	5.978106201	5.2
GSM327287	7.32644986	5.5553274205	6.7972747255	5.730106395	5.8
GSM327288	6.509280485	4.7726442415	6.711350831	5.797783632	4.9
GSM327289	6.609537997	4.822522375	6.2396119175	5.635043605	5.8

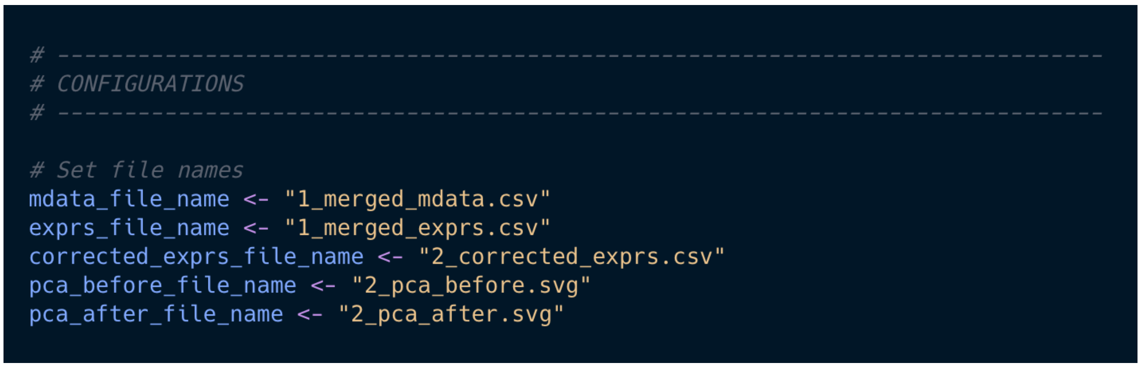
Figure 6.4 Merged Expressions



	Dataset	Status
GSM327282	GSE13067	MSI-H
GSM327283	GSE13067	MSS
GSM327284	GSE13067	MSS
GSM327285	GSE13067	MSS
GSM327286	GSE13067	MSS
GSM327287	GSE13067	MSS
GSM327288	GSE13067	MSI-H
GSM327289	GSE13067	MSS

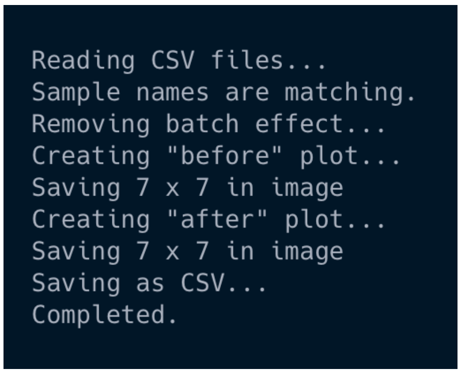
Figure 6.5 Merged Metadata

Output tables were used as input in `2_remove_batch_effect.R` script's configuration as shown in 6.6. The progress outputs can be seen in 6.7. The removing batch effect result can easily be seen comparing 6.8 and 6.9.



```r
# --------------------------------------------------------------------------------
# CONFIGURATIONS
# --------------------------------------------------------------------------------

# Set file names
mdata_file_name <- "1_merged_mdata.csv"
exprs_file_name <- "1_merged_exprs.csv"
corrected_exprs_file_name <- "2_corrected_exprs.csv"
pca_before_file_name <- "2_pca_before.svg"
pca_after_file_name <- "2_pca_after.svg"
```

Figure 6.6 Configurations For Removing Batch Effect



```
Reading CSV files...
Sample names are matching.
Removing batch effect...
Creating "before" plot...
Saving 7 x 7 in image
Creating "after" plot...
Saving 7 x 7 in image
Saving as CSV...
Completed.
```

Figure 6.7 Outputs During Removing Batch Effect

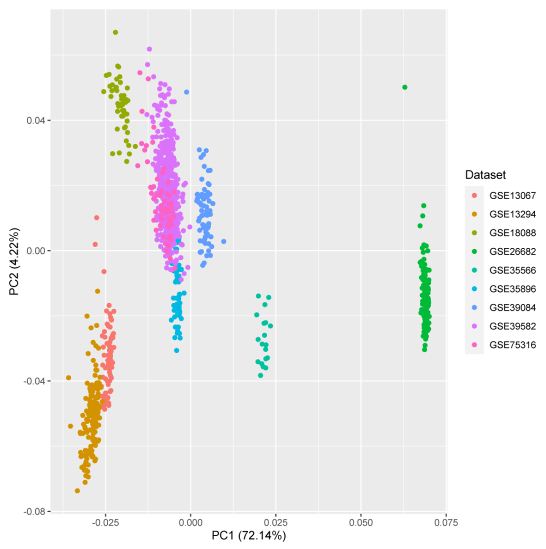


Figure 6.8 Expression Values Before Removing Batch Effect

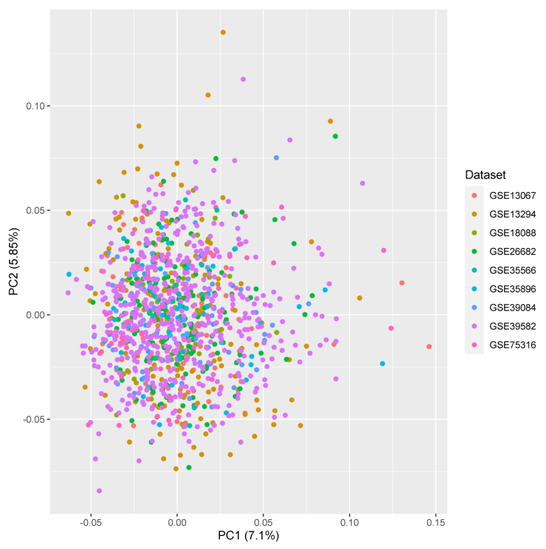
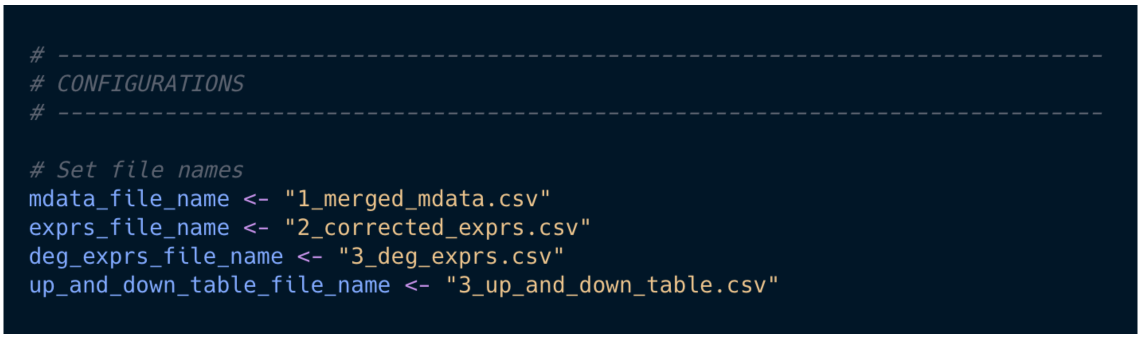


Figure 6.9 Expression Values After Removing Batch Effect

Corrected table was used as input in `3_differential_gene_expression.R` script's configuration as shown in 6.10.

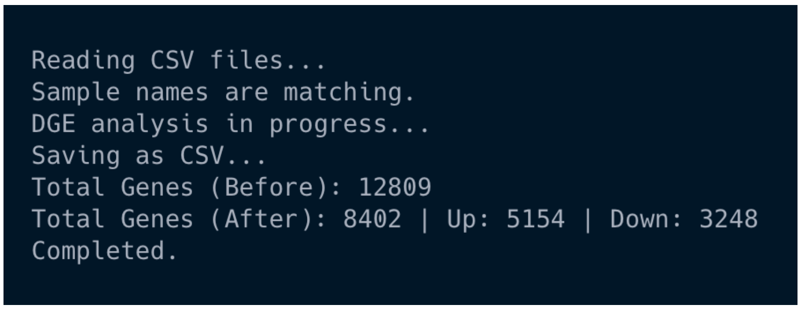


```
# ---------------------------------------------------------------------------
# CONFIGURATIONS
# ---------------------------------------------------------------------------

# Set file names
mdata_file_name <- "1_merged_mdata.csv"
exprs_file_name <- "2_corrected_exprs.csv"
deg_exprs_file_name <- "3_deg_exprs.csv"
up_and_down_table_file_name <- "3_up_and_down_table.csv"
```

Figure 6.10 Configurations For DGE Analysis

As seen in 6.11, upregulated and downregulated genes were found. These means:

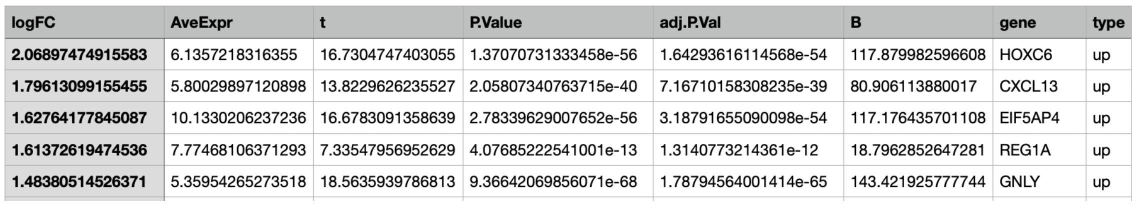
- 5154 genes are probably has effect on MSI-H.
- 3248 genes are probably has effect on MSS.
- The changes on the other 4407 genes expression values do not significantly affect the MSI/MMR status.



```
Reading CSV files...
Sample names are matching.
DGE analysis in progress...
Saving as CSV...
Total Genes (Before): 12809
Total Genes (After): 8402 | Up: 5154 | Down: 3248
Completed.
```

Figure 6.11 Outputs During DGE

The up/downregulations table was saved as CSV, as shown in 6.12. The table was used as input in Survival Analysis.

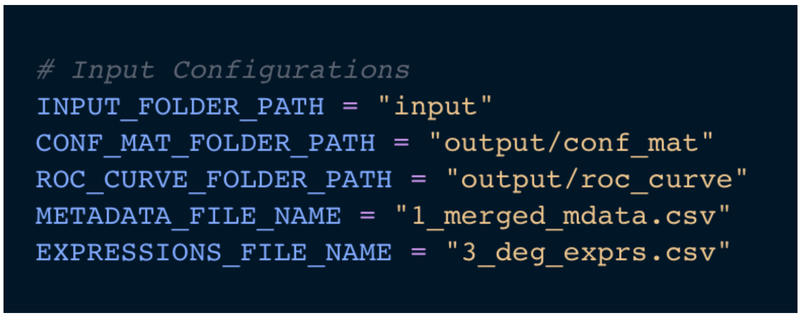


logFC	AveExpr	t	P.Value	adj.P.Val	B	gene	type
2.06897474915583	6.1357218316355	16.7304747403055	1.37070731333458e-56	1.64293616114568e-54	117.879982596608	HOXC6	up
1.79613099155455	5.80029897120898	13.8229626235527	2.05807340763715e-40	7.16710158308235e-39	80.906113880017	CXCL13	up
1.62764177845087	10.1330206237236	16.6783091358639	2.78339629007652e-56	3.18791655090098e-54	117.176435701108	EIF5AP4	up
1.61372619474536	7.77468106371293	7.33547956952629	4.07685222541001e-13	1.3140773214361e-12	18.7962852647281	REG1A	up
1.48380514526371	5.35954265273518	18.5635939786813	9.36642069856071e-68	1.78794564001414e-65	143.421925777744	GNLY	up

Figure 6.12 Upregulations & Downregulations Table

6.2 Training & Testing

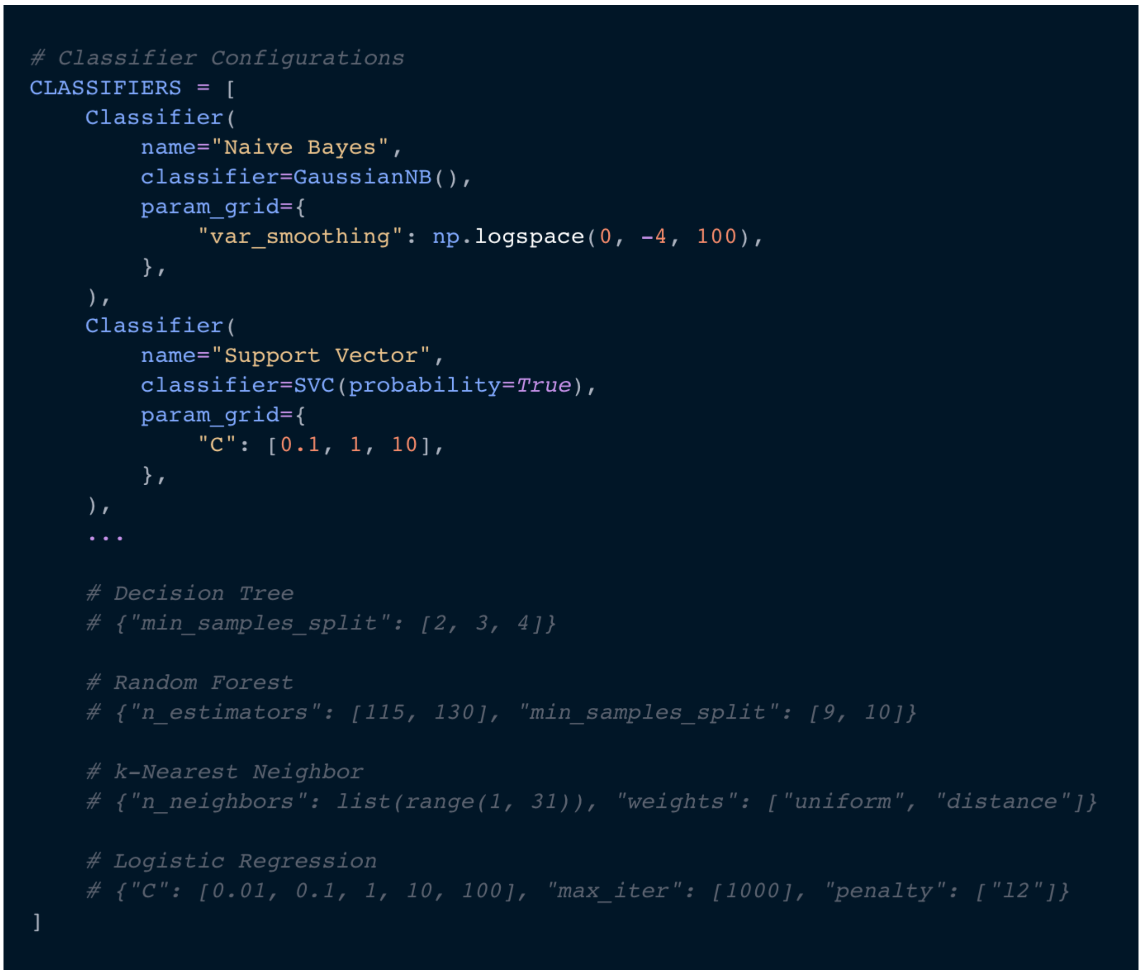
Expressions and metadata tables were used as input in training and testing module as shown in 6.13.



```python
# Input Configurations
INPUT_FOLDER_PATH = "input"
CONF_MAT_FOLDER_PATH = "output/conf_mat"
ROC_CURVE_FOLDER_PATH = "output/roc_curve"
METADATA_FILE_NAME = "1_merged_mdata.csv"
EXPRESSIONS_FILE_NAME = "3_deg_exprs.csv"
```

Figure 6.13 Input Configurations For Training & Testing

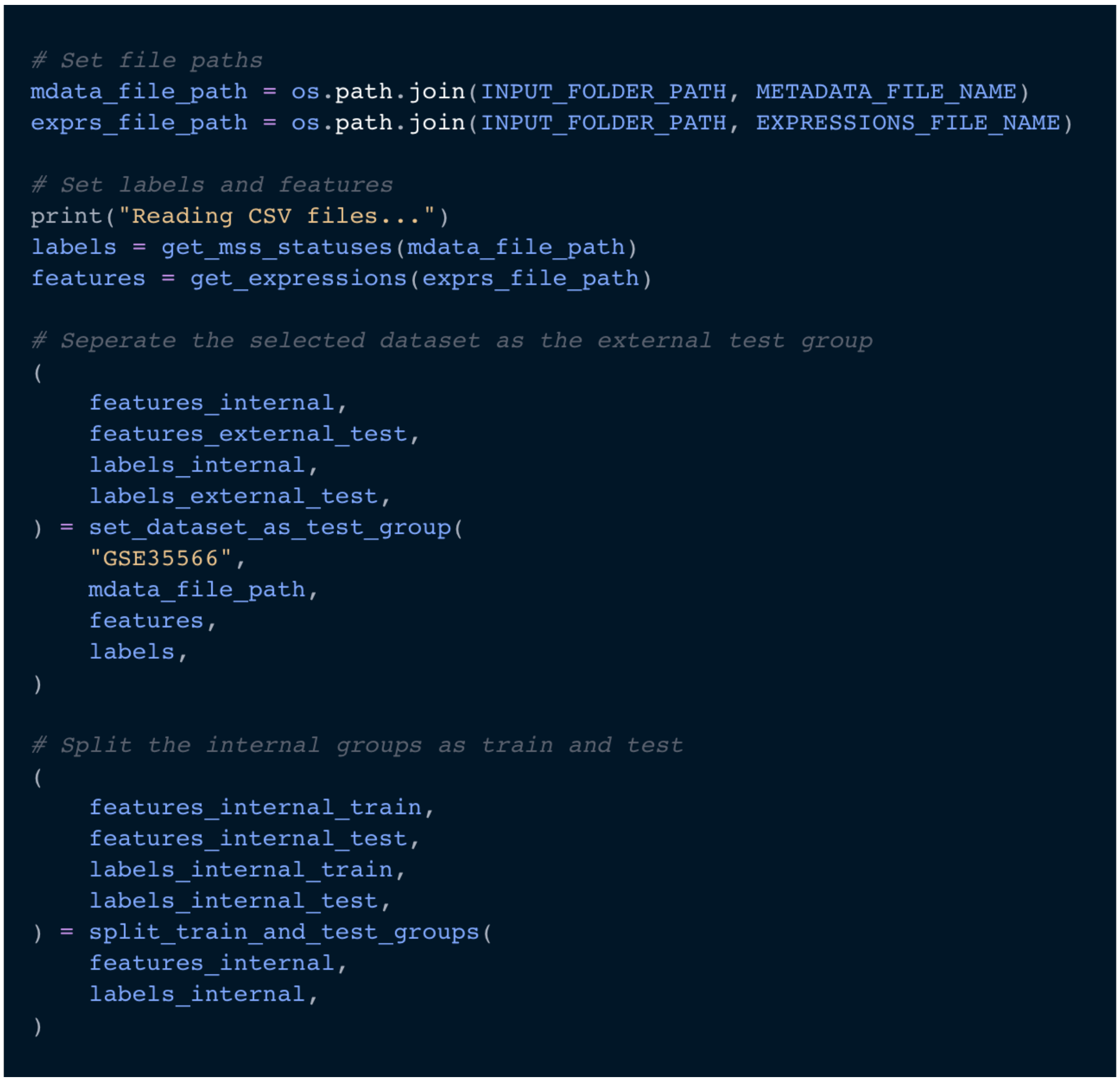
A common Python library scikit-learn was used to develop the module. All classifiers shown in 6.14 are from this library. Each classifier's possible parameters are defined as `param_grid` in the custom `Classifier` class.



```python
# Classifier Configurations
CLASSIFIERS = [
    Classifier(
        name="Naive Bayes",
        classifier=GaussianNB(),
        param_grid={
            "var_smoothing": np.logspace(0, -4, 100),
        },
    ),
    Classifier(
        name="Support Vector",
        classifier=SVC(probability=True),
        param_grid={
            "C": [0.1, 1, 10],
        },
    ),
    ...

    # Decision Tree
    # {"min_samples_split": [2, 3, 4]}

    # Random Forest
    # {"n_estimators": [115, 130], "min_samples_split": [9, 10]}

    # k-Nearest Neighbor
    # {"n_neighbors": list(range(1, 31)), "weights": ["uniform", "distance"]}

    # Logistic Regression
    # {"C": [0.01, 0.1, 1, 10, 100], "max_iter": [1000], "penalty": ["l2"]}
]
```

Figure 6.14 Classifier Configurations

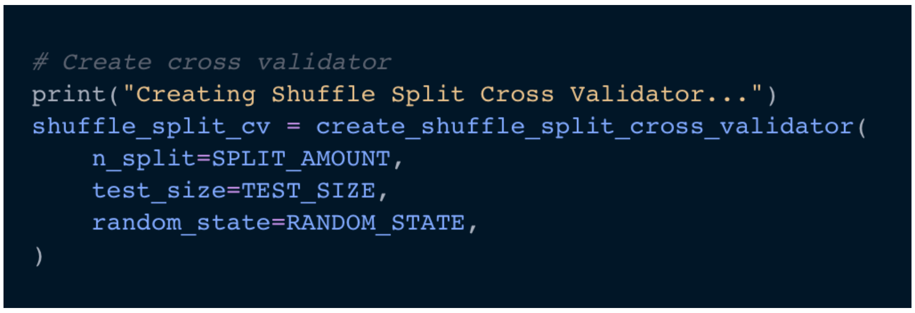
After reading CSV datasets, "GSE35566" dataset was seperated to use later as external dataset as shown in 6.15. It was merged before to be sure the genes are the same in all datasets. The rest of the merged dataset was splitted into train and test groups.



```python
# Set file paths
mdata_file_path = os.path.join(INPUT_FOLDER_PATH, METADATA_FILE_NAME)
exprs_file_path = os.path.join(INPUT_FOLDER_PATH, EXPRESSIONS_FILE_NAME)

# Set labels and features
print("Reading CSV files...")
labels = get_mss_statuses(mdata_file_path)
features = get_expressions(exprs_file_path)

# Seperate the selected dataset as the external test group
(
    features_internal,
    features_external_test,
    labels_internal,
    labels_external_test,
) = set_dataset_as_test_group(
    "GSE35566",
    mdata_file_path,
    features,
    labels,
)

# Split the internal groups as train and test
(
    features_internal_train,
    features_internal_test,
    labels_internal_train,
    labels_internal_test,
) = split_train_and_test_groups(
    features_internal,
    labels_internal,
)
```

Figure 6.15 Separating & Splitting Datasets

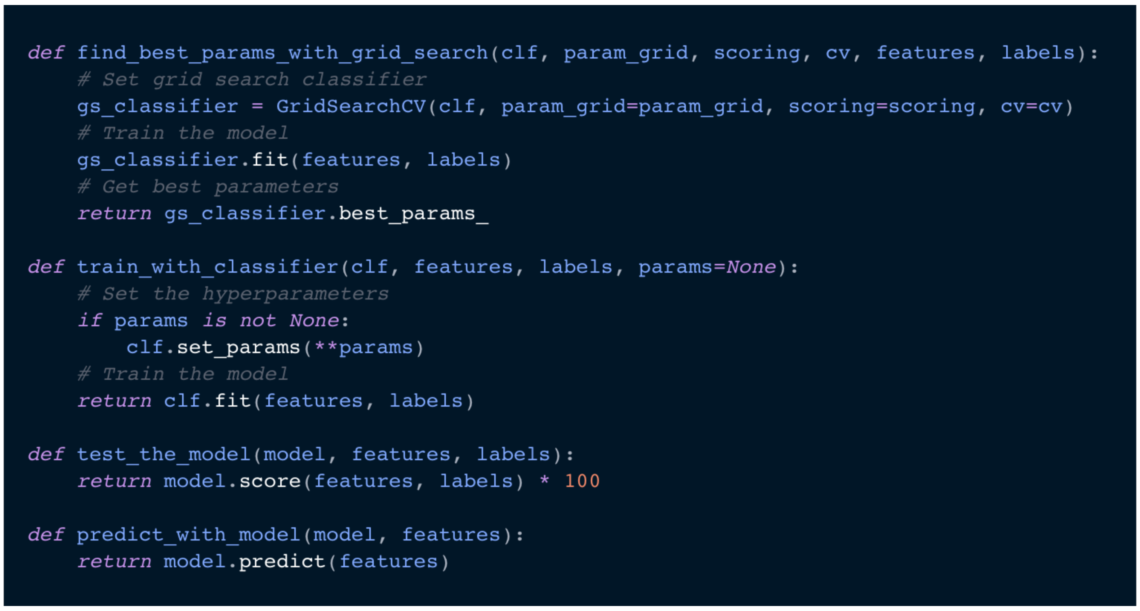
Shuffle Split was added as cross validator to be used during Grid Search in 6.16.



```python
# Create cross validator
print("Creating Shuffle Split Cross Validator...")
shuffle_split_cv = create_shuffle_split_cross_validator(
    n_split=SPLIT_AMOUNT,
    test_size=TEST_SIZE,
    random_state=RANDOM_STATE,
)
```

Figure 6.16 Creating Cross Validator

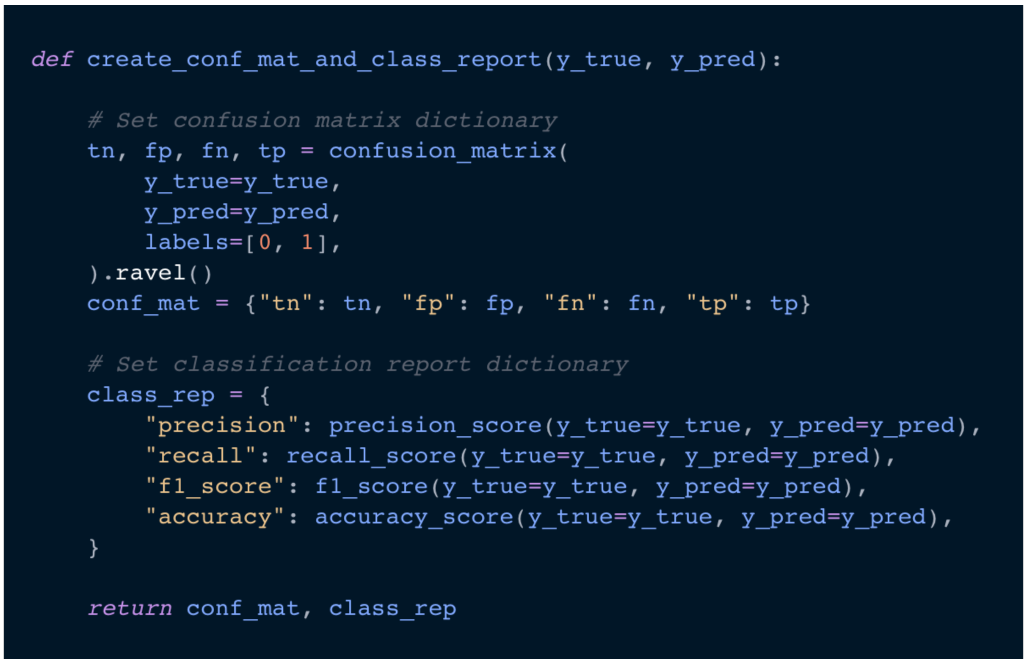
For each dataset, best parameters were found in Grid Search, a model was trained and tested with internal and external data, with the functions in 6.17.



```python
def find_best_params_with_grid_search(clf, param_grid, scoring, cv, features, labels):
    # Set grid search classifier
    gs_classifier = GridSearchCV(clf, param_grid=param_grid, scoring=scoring, cv=cv)
    # Train the model
    gs_classifier.fit(features, labels)
    # Get best parameters
    return gs_classifier.best_params_

def train_with_classifier(clf, features, labels, params=None):
    # Set the hyperparameters
    if params is not None:
        clf.set_params(**params)
    # Train the model
    return clf.fit(features, labels)

def test_the_model(model, features, labels):
    return model.score(features, labels) * 100

def predict_with_model(model, features):
    return model.predict(features)
```

Figure 6.17 Grid Search, Training & Testing Functions

After testing and predicting, the confusion matrix and classification report results were calculated, with the functions in 6.18.



```python
def create_conf_mat_and_class_report(y_true, y_pred):

    # Set confusion matrix dictionary
    tn, fp, fn, tp = confusion_matrix(
        y_true=y_true,
        y_pred=y_pred,
        labels=[0, 1],
    ).ravel()
    conf_mat = {"tn": tn, "fp": fp, "fn": fn, "tp": tp}

    # Set classification report dictionary
    class_rep = {
        "precision": precision_score(y_true=y_true, y_pred=y_pred),
        "recall": recall_score(y_true=y_true, y_pred=y_pred),
        "f1_score": f1_score(y_true=y_true, y_pred=y_pred),
        "accuracy": accuracy_score(y_true=y_true, y_pred=y_pred),
    }

    return conf_mat, class_rep
```

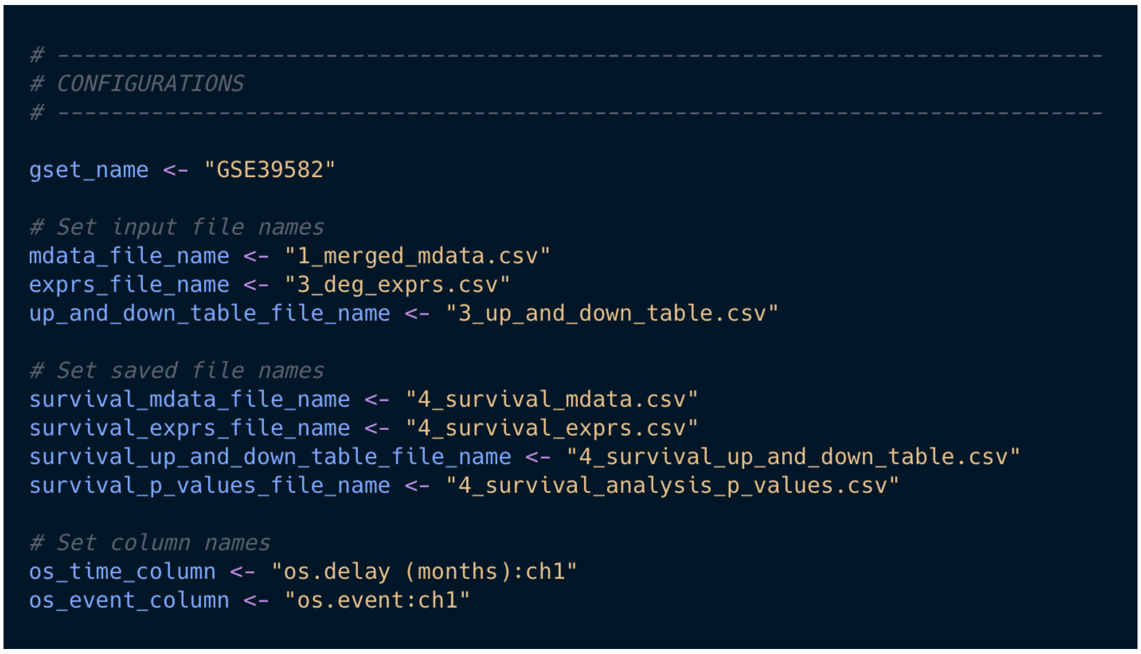
Figure 6.18 Confusion Matrix & Classification Report Function

For last, ROC Curve plots were created and all results were printed out or saved to output folder.

6.3 Survival Analysis

From all datasets, only "GSE39582" dataset has the requested metadata, which is overall survival (OS) data. That is why this dataset was used in Survival Analysis.

The name of the dataset, expressions and metadata tables were used as input in training and testing module as shown in 6.19.

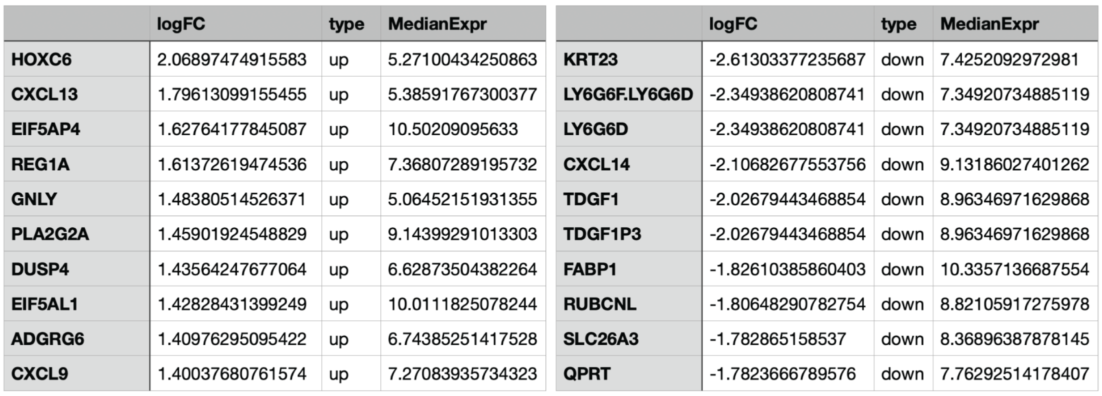


```
# ----------------------------------------------------------------------------
# CONFIGURATIONS
# ----------------------------------------------------------------------------

gset_name <- "GSE39582"

# Set input file names
mdata_file_name <- "1_merged_mdata.csv"
exprs_file_name <- "3_deg_exprs.csv"
up_and_down_table_file_name <- "3_up_and_down_table.csv"

# Set saved file names
survival_mdata_file_name <- "4_survival_mdata.csv"
survival_exprs_file_name <- "4_survival_exprs.csv"
survival_up_and_down_table_file_name <- "4_survival_up_and_down_table.csv"
survival_p_values_file_name <- "4_survival_analysis_p_values.csv"

# Set column names
os_time_column <- "os.delay (months):ch1"
os_event_column <- "os.event:ch1"
```

Figure 6.19 Survival Analysis Configurations

From up/downregulations table, 10 genes from each regulation group (total of 20 genes) were selected. The selection metric was logFC values of each gene.

For selected genes, medians of expression values were added to survival genes table as shown in 6.20.



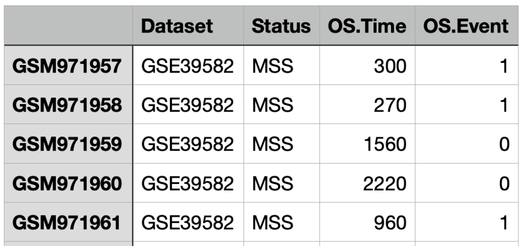
	logFC	type	MedianExpr
HOXC6	2.06897474915583	up	5.27100434250863
CXCL13	1.79613099155455	up	5.38591767300377
EIF5AP4	1.62764177845087	up	10.50209095633
REG1A	1.61372619474536	up	7.36807289195732
GNLY	1.48380514526371	up	5.06452151931355
PLA2G2A	1.45901924548829	up	9.14399291013303
DUSP4	1.43564247677064	up	6.62873504382264
EIF5AL1	1.42828431399249	up	10.0111825078244
ADGRG6	1.40976295095422	up	6.74385251417528
CXCL9	1.40037680761574	up	7.27083935734323

	logFC	type	MedianExpr
KRT23	-2.61303377235687	down	7.4252092972981
LY6G6F.LY6G6D	-2.34938620808741	down	7.34920734885119
LY6G6D	-2.34938620808741	down	7.34920734885119
CXCL14	-2.10682677553756	down	9.13186027401262
TDGF1	-2.02679443468854	down	8.96346971629868
TDGF1P3	-2.02679443468854	down	8.96346971629868
FABP1	-1.82610385860403	down	10.3357136687554
RUBCNL	-1.80648290782754	down	8.82105917275978
SLC26A3	-1.782865158537	down	8.36896387878145
QPRT	-1.7823666789576	down	7.76292514178407

Figure 6.20 Survival Genes Table

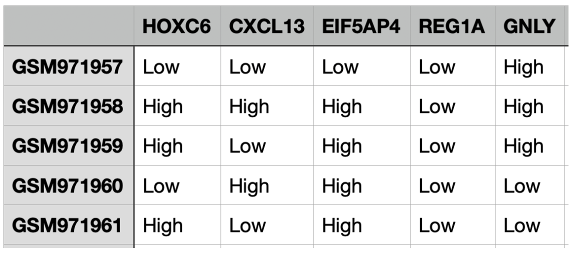
The expressions and metadata tables were reduced and minimized. The values of expressions table were changed with "High" or "Low" labels due to their correlation with their median value like in 6.21. For every sample in the dataset, OS times and OS events were added to the metadata table as in 6.22.

Normally, samples with MSS and MSI must be separated. But in this dataset, there are way more MSS values than MSI, so they are not seperated and used as a whole.



	Dataset	Status	OS.Time	OS.Event
GSM971957	GSE39582	MSS	300	1
GSM971958	GSE39582	MSS	270	1
GSM971959	GSE39582	MSS	1560	0
GSM971960	GSE39582	MSS	2220	0
GSM971961	GSE39582	MSS	960	1

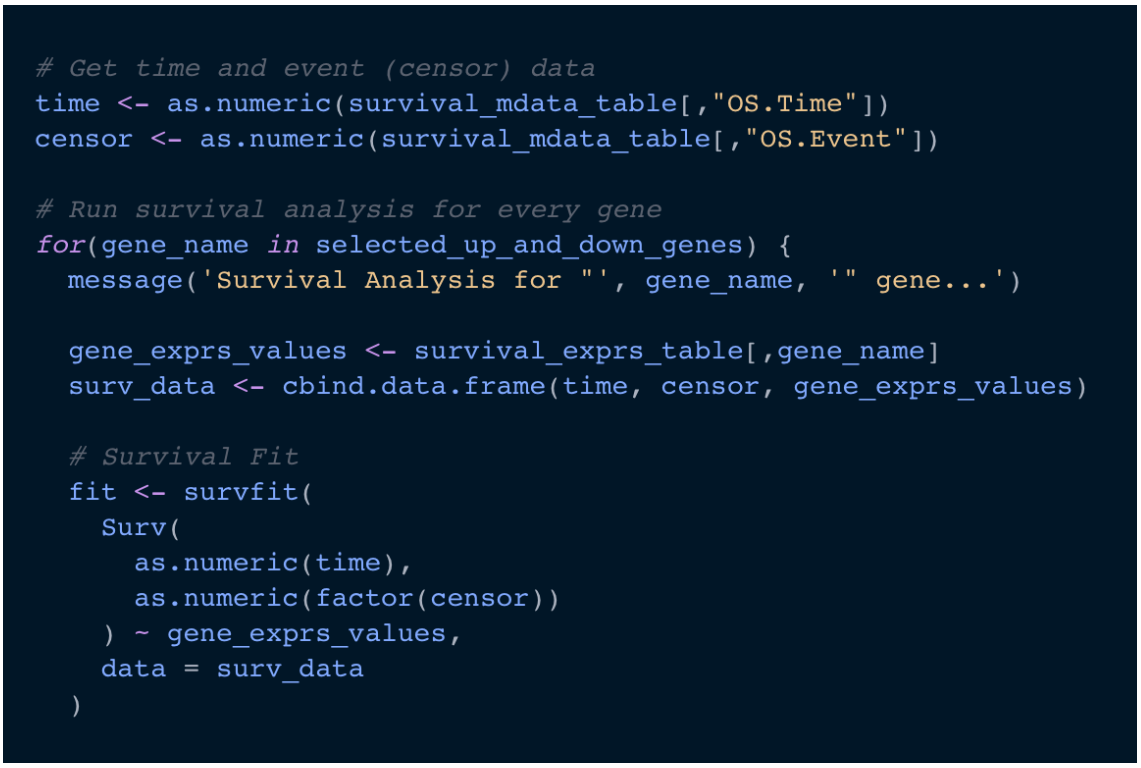
Figure 6.21 Survival Expressions Table



	HOXC6	CXCL13	EIF5AP4	REG1A	GNLY
GSM971957	Low	Low	Low	Low	High
GSM971958	High	High	High	Low	High
GSM971959	High	Low	High	Low	High
GSM971960	Low	High	High	Low	Low
GSM971961	High	Low	High	Low	Low

Figure 6.22 Survival Metadata Table

For every gene, an estimate of a survival curve for censored data was computed in 6.23. Each curve was saved as a plot and calculated p-values were printed out and analyzed.



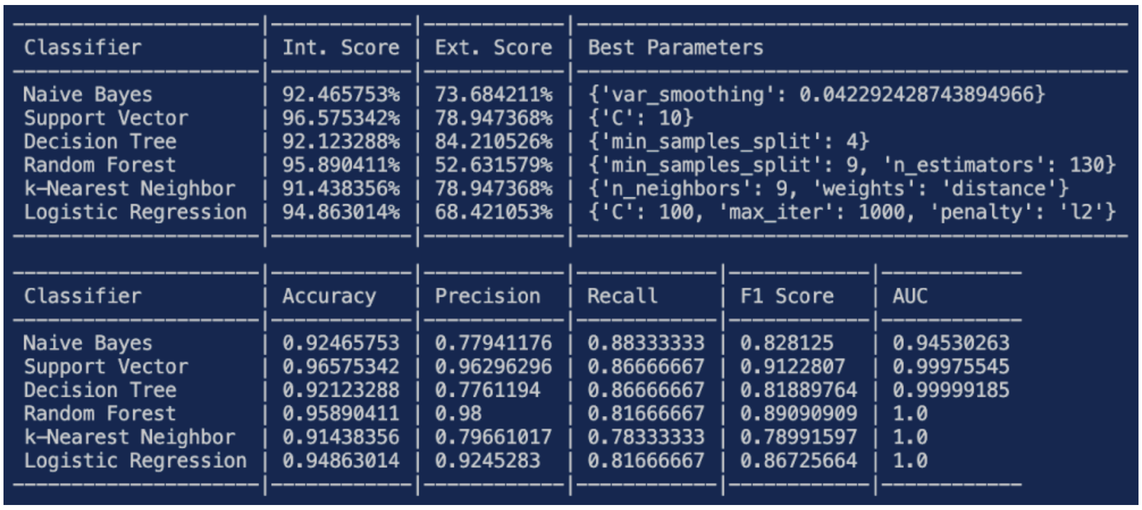
```r
# Get time and event (censor) data
time <- as.numeric(survival_mdata_table[,"OS.Time"])
censor <- as.numeric(survival_mdata_table[,"OS.Event"])

# Run survival analysis for every gene
for(gene_name in selected_up_and_down_genes) {
  message('Survival Analysis for "', gene_name, '" gene...')

  gene_exprs_values <- survival_exprs_table[,gene_name]
  surv_data <- cbind.data.frame(time, censor, gene_exprs_values)

  # Survival Fit
  fit <- survfit(
    Surv(
      as.numeric(time),
      as.numeric(factor(censor))
    ) ~ gene_exprs_values,
    data = surv_data
  )
```

Figure 6.23 Survival Fit

7
Experimental Results

The whole project was designed as modular as possible. To try different parameters or inputs, it is enough to change the configuration parts of each module. To test different cases, the used configurations were changed and improved.

For each input or in each run in training and testing, best parameters can change. To deal with this, multiple parameters with Grid Search were used. The best parameters which were selected by Grid Search were shown in 7.1.



```
---------------------|------------|------------|------------------------------------------------
 Classifier          | Int. Score | Ext. Score | Best Parameters
---------------------|------------|------------|------------------------------------------------
 Naive Bayes         | 92.465753% | 73.684211% | {'var_smoothing': 0.042292428743894966}
 Support Vector      | 96.575342% | 78.947368% | {'C': 10}
 Decision Tree       | 92.123288% | 84.210526% | {'min_samples_split': 4}
 Random Forest       | 95.890411% | 52.631579% | {'min_samples_split': 9, 'n_estimators': 130}
 k-Nearest Neighbor  | 91.438356% | 78.947368% | {'n_neighbors': 9, 'weights': 'distance'}
 Logistic Regression | 94.863014% | 68.421053% | {'C': 100, 'max_iter': 1000, 'penalty': 'l2'}
---------------------|------------|------------|------------------------------------------------

---------------------|------------|------------|------------|------------|------------
 Classifier          | Accuracy   | Precision  | Recall     | F1 Score   | AUC
---------------------|------------|------------|------------|------------|------------
 Naive Bayes         | 0.92465753 | 0.77941176 | 0.88333333 | 0.828125   | 0.94530263
 Support Vector      | 0.96575342 | 0.96296296 | 0.86666667 | 0.9122807  | 0.99975545
 Decision Tree       | 0.92123288 | 0.7761194  | 0.86666667 | 0.81889764 | 0.99999185
 Random Forest       | 0.95890411 | 0.98       | 0.81666667 | 0.89090909 | 1.0
 k-Nearest Neighbor  | 0.91438356 | 0.79661017 | 0.78333333 | 0.78991597 | 1.0
 Logistic Regression | 0.94863014 | 0.9245283  | 0.81666667 | 0.86725664 | 1.0
---------------------|------------|------------|------------|------------|------------
```

Figure 7.1 The Results Of Performance/Success Metrics

As shown in 7.1, the test was repeated with internal and external data, to see if the success vary with a completely new sample.

The confusion matrices of all classifiers were shown in 7.2. In this matrix, a 0 to 1 match or a 1 to 0 match means the prediction is wrong (false). The sum of these two matches (top-right and bottom-left parts) gives the total wrong predictions.

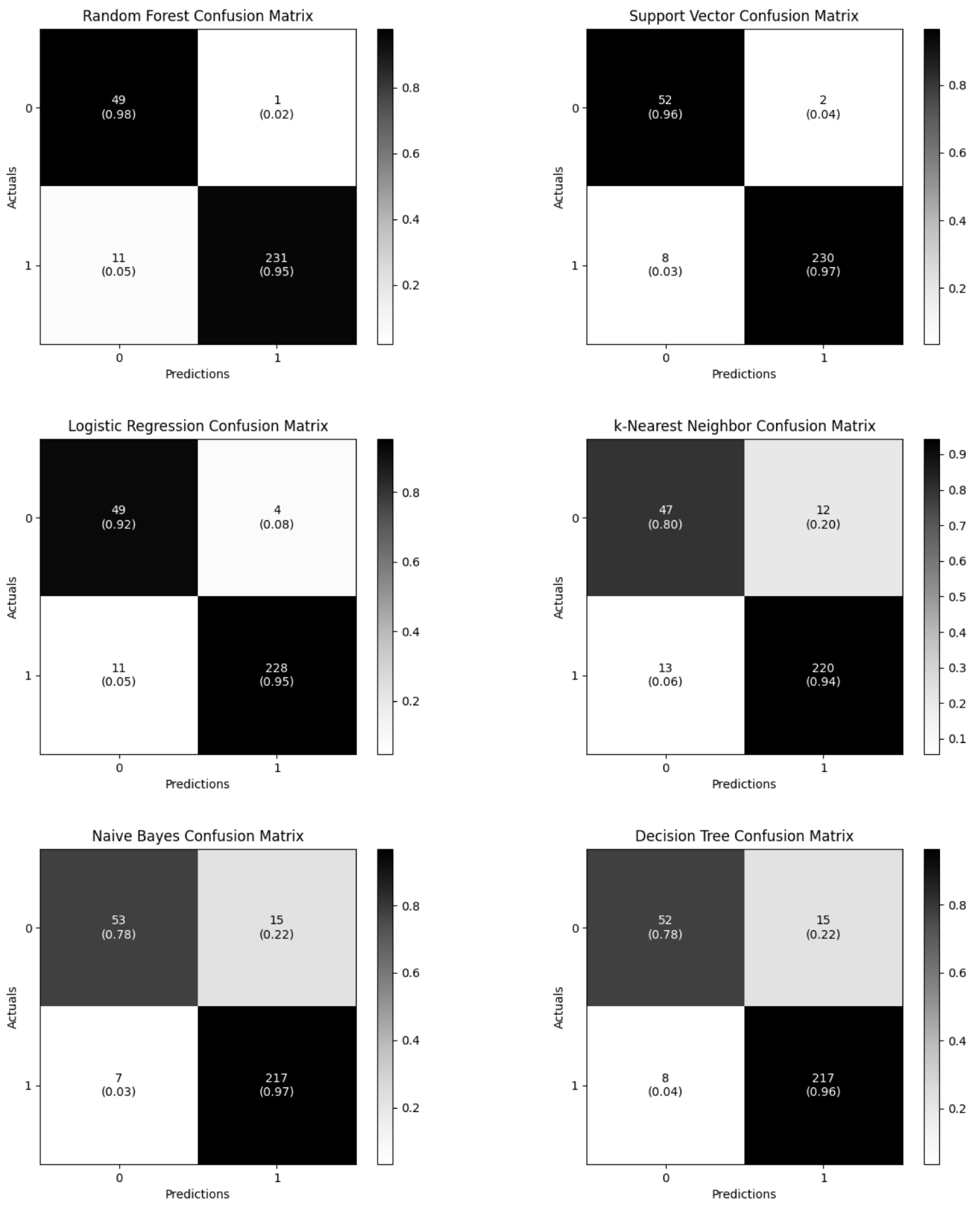


Figure 7.2 Confusion Matrices

Another metric is AUC (from ROC Curve). The score is higher as close as the curve is to top-left corner, and AUC gets higher too. The lowest results were shown in 7.3.

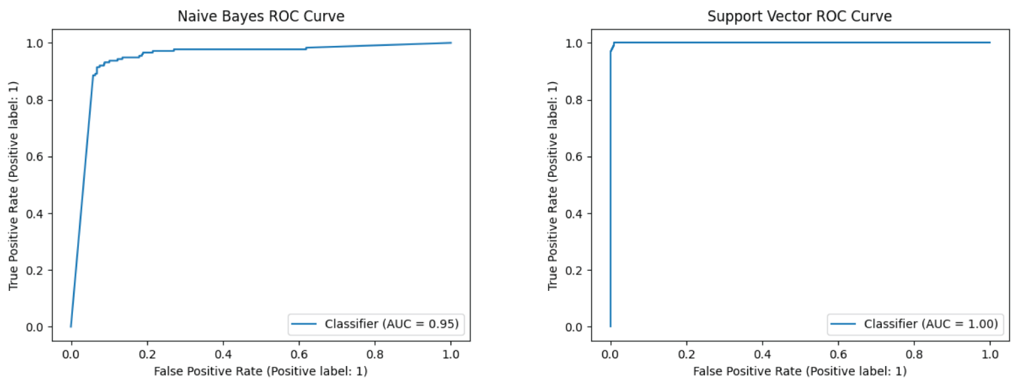
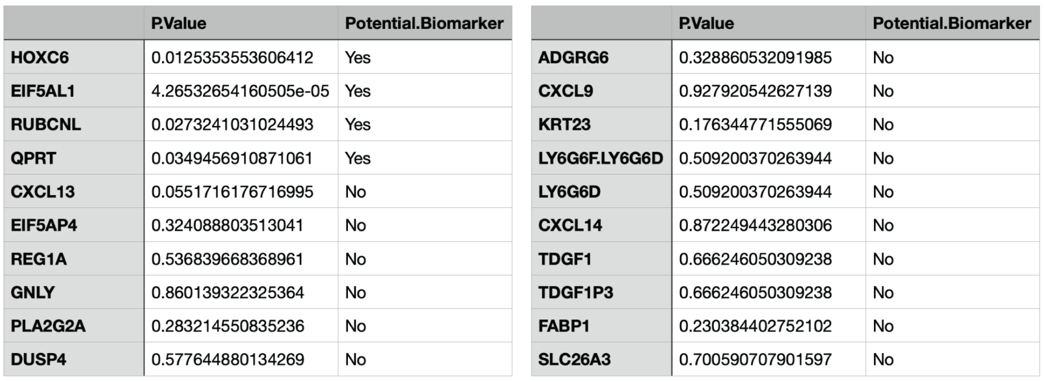


Figure 7.3 ROC Curves Of The Least Successful Classifiers

For Survival Analysis, the key metric is p-value. If the calculated p-value for a gene is less than 0.05, it can be considered as a survival marker. Four of tested genes were qualified as survival marker as shown in 7.4.



	P.Value	Potential.Biomarker
HOXC6	0.0125353553606412	Yes
EIF5AL1	4.26532654160505e-05	Yes
RUBCNL	0.0273241031024493	Yes
QPRT	0.0349456910871061	Yes
CXCL13	0.0551716176716995	No
EIF5AP4	0.324088803513041	No
REG1A	0.536839668368961	No
GNLY	0.860139322325364	No
PLA2G2A	0.283214550835236	No
DUSP4	0.577644880134269	No

	P.Value	Potential.Biomarker
ADGRG6	0.328860532091985	No
CXCL9	0.927920542627139	No
KRT23	0.176344771555069	No
LY6G6F.LY6G6D	0.509200370263944	No
LY6G6D	0.509200370263944	No
CXCL14	0.872249443280306	No
TDGF1	0.666246050309238	No
TDGF1P3	0.666246050309238	No
FABP1	0.230384402752102	No
SLC26A3	0.700590707901597	No

Figure 7.4 Survival Analysis Results

The detailed plot of potential survival marker genes survival analysis results were shown in 7.5.

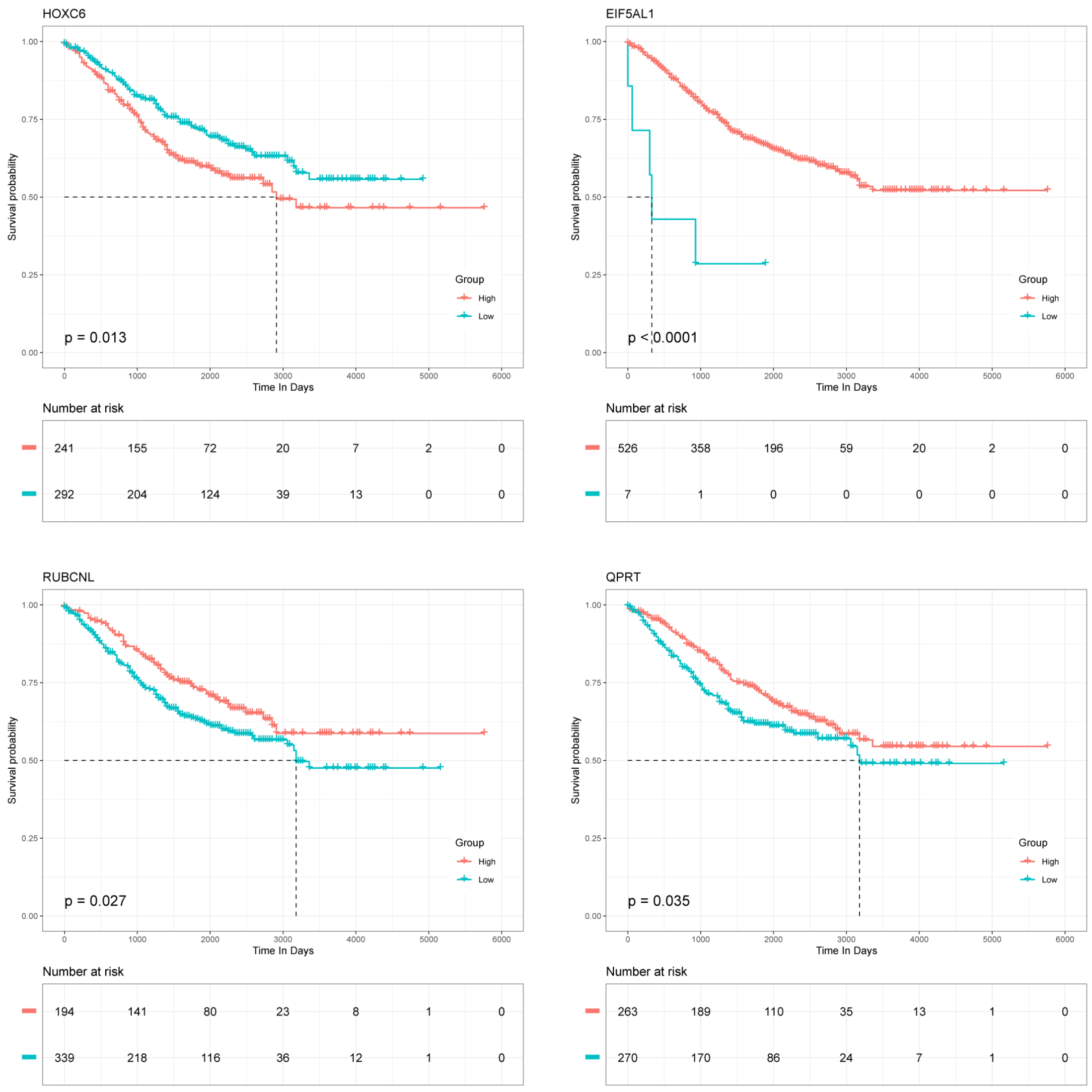


Figure 7.5 Survival Analysis Plots

8
Performance Analysis

As seen in 7.1, tests were repeated for internal and external test groups. The score of internal test group is higher as expected, because its train group has similar data.

Even with external test group, the score is rising up to 84% in Decision Tree, which is a really great result for external data. The dataset size and merged dataset structure has a great impact on this score, beacuse a big dataset has more example and a merged dataset has different types of samples to examine.

When comparing internal and external scores, the highest change is in Random Forest. As in other classifiers, the best parameters for Random Forest vary in every run. But variations are more in Random Forest, due to its randomly growing trees. This is why the internal score of Random Forest changed in all runs, up to 89%. Another reason can be parameter selection in configuration, a change on ranges for each parameter can lead to a more stable result.

As shown in 7.4, four potential survival genes were found. These are HOXC6, EIF5AL1, RUBCNL and QPRT. A search was done on DisGeNET platform for each gene and the results were shared below.

Here, GDA score is a method to show how well the de-identification protects individual user privacy and how much analytic value remains after anonymization [21].

- **HOXC6:** It has the GDA score of 0.07 for Prostate Carcinoma and 0.02 for Cervix Carcinoma [22]. Both diseases have been shown to cause CRC [23] [24].
- **EIF5AL1:** It has the GDA score of 0.01 for Ovarian Carcinoma, Colorectal Carcinoma and Lung Neoplasm [25].
- **RUBCNL:** It has the GDA score of 0.03 for Cervix Cancer and Cervix Carcinoma and 0.01 for High Grade Cervical Intraepithelial Neoplasia [26]. As said before, Cervix Cancer/Carcinoma has been shown to cause CRC [23].
- **QPRT:** It has the GDA score of 0.2 for Huntington Disease, which is an autosomal dominant progressive neurodegenerative disorder [27] [28].

9 Conclusion

After the long process of development, three modules were completed. Multiple datasets were merged during preprocesses, 6 different models were trained and tested, and 4 genes were found as survival markers.

9.1 Evaluation Of Experiment Results

The goal of minimum 80% score was achieved for testing with both internal and external data. Decision Tree was found to be the most successful method with 92% score with internal data and 84% score with external data.

Also, the goal of finding at least one new survival marker was achieved with 4 potential survival markers. The marker EIF5AL1 was found to be directly related to CRC, and the markers HOXC6 and RUBCNL were found to be related to cervix/prostate cancer, which can cause CRC.

9.2 Advantages and Deficiencies

The biggest advantage of the project was being a modular project. Each step can be skipped or repeated. Besides, it is easy to change the configurations for each module. It allows to try different datasets from NCBI, which makes the project flexible.

An important deficiency is finding best parameters during fine-tuning. Due to the time limit and hardware limitations, trying many different hyperparameters while training was not feasible. As the number of parameters increases, the computation time increases too. That is why many parameters could not tried.

9.3 Suggestions

Two things can be suggested, make your project modular and save every work as a new version and never delete a piece of code.

The modularity of the project was made the development process very easy. For example, every module has its own configurations, adding a new classifier is easy as copy and paste it to the list of classifiers in the configurations.

Besides, every new piece of code was stored as a new version. Sometimes, the previous version of a code is needed to add a new feature. Version controlling helped this process a lot.

References

[1] A. C. Society. "Key statistics for colorectal cancer." (), [Online]. Available: https://www.cancer.org/cancer/colon-rectal-cancer/about/key-statistics.html (visited on 04/17/2023).

[2] F. Ponzio, E. Macii, E. Ficarra, and S. D. Cataldo. "Colorectal cancer classification using deep convolutional networks - an experimental study." (2018), [Online]. Available: https://www.researchgate.net/publication/322881743_Colorectal_Cancer_Classification_using_Deep_Convolutional_Networks_-_An_Experimental_Study (visited on 04/17/2023).

[3] H. Li *et al.* "Colorectal cancer detected by machine learning models using conventional laboratory test data." (2021), [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8606732/ (visited on 04/17/2023).

[4] L. Li, Q. Feng, and X. Wang. "Premsim: An r package for predicting microsatellite instability from the expression profiling of a gene panel in cancer." (2020), [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/32257050/ (visited on 04/17/2023).

[5] A. Brooks. "How much does an mri cost without insurance in 2023?" (2023), [Online]. Available: https://www.talktomira.com/post/how-much-does-an-mri-cost-without-insurance-in-2021 (visited on 04/20/2023).

[6] NCBI. "Ncbi geo gpl570 platform." (2003), [Online]. Available: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GPL570 (visited on 04/18/2023).

[7] R. C. Team. "What is r?" (), [Online]. Available: https://www.r-project.org/about.html (visited on 04/18/2023).

[8] "Kvkk." (), [Online]. Available: https://www.kvkk.gov.tr/en/ (visited on 04/18/2023).

[9] A. C. Society. "Tests to diagnose and stage colorectal cancer." (), [Online]. Available: https://www.cancer.org/cancer/colon-rectal-cancer/detection-diagnosis-staging/how-diagnosed.html (visited on 04/19/2023).

[10] T. Onega *et al.* "Costs of diagnostic and preoperative workup with and without breast mri in older women with a breast cancer diagnosis." (2016), [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4769533/ (visited on 04/19/2023).

[11] A. Anjum, S. Jaggi, E. Varghese, S. Lall, A. Bhowmik, and A. Rai. “Identification of differentially expressed genes in rna-seq data of arabidopsis thaliana: A compound distribution approach.” (2016), [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4827276/ (visited on 04/30/2023).

[12] “Shuffle split.” (), [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html#random-permutations-cross-validation-a-k-a-shuffle-split (visited on 04/30/2023).

[13] “Exhaustive grid search.” (), [Online]. Available: https://scikit-learn.org/stable/modules/grid_search.html#exhaustive-grid-search (visited on 05/31/2023).

[14] “What is logistic regression?” (), [Online]. Available: https://www.ibm.com/topics/logistic-regression (visited on 04/30/2023).

[15] “What is the k-nearest neighbors algorithm?” (), [Online]. Available: https://www.ibm.com/topics/knn (visited on 04/30/2023).

[16] “What is a decision tree?” (), [Online]. Available: https://www.ibm.com/topics/decision-trees (visited on 04/30/2023).

[17] “How svm works.” (), [Online]. Available: https://www.ibm.com/docs/en/spss-modeler/saas?topic=models-how-svm-works (visited on 04/30/2023).

[18] “What are naive bayes classifiers?” (), [Online]. Available: https://www.ibm.com/topics/naive-bayes (visited on 04/30/2023).

[19] “What is random forest?” (), [Online]. Available: https://www.ibm.com/topics/random-forest (visited on 04/30/2023).

[20] “Geo datasets.” (), [Online]. Available: https://www.ncbi.nlm.nih.gov/gds (visited on 04/29/2023).

[21] “Gda score overview.” (), [Online]. Available: https://www.gda-score.org/wp-content/uploads/2019/01/gda-score-overview-v1.pdf (visited on 06/02/2023).

[22] “Disgenet - hoxc6.” (), [Online]. Available: https://www.disgenet.org/browser/1/1/0/3223/ (visited on 06/02/2023).

[23] S.-C. Liao, H.-Z. Yeh, C.-S. Chang, W.-C. Chen, C.-H. Muo, and F.-C. Sung. “Radiation therapy for cervical cancer increases risk for colorectal cancer.” (2021), [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8303695/ (visited on 06/02/2023).

[24] S.-H. Ko, M. K. Baeg, W. J. Bae, P. Kim, and M.-G. Choi. “Prostate cancer patients may have an increased risk of coexisting advanced colorectal neoplasms.” (2016), [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5024772/ (visited on 06/02/2023).

[25] “Disgenet - eif5al1.” (), [Online]. Available: https://www.disgenet.org/browser/1/1/0/143244/ (visited on 06/02/2023).

[26] “Disgenet - rubcnl.” (), [Online]. Available: https://www.disgenet.org/browser/1/1/0/80183/ (visited on 06/02/2023).

[27] "Disgenet - qprt." (), [Online]. Available: https://www.disgenet.org/browser/1/1/0/23475/ (visited on 06/02/2023).

[28] "Huntington disease." (), [Online]. Available: https://www.ncbi.nlm.nih.gov/medgen/C0020179 (visited on 06/02/2023).

Curriculum Vitae

FIRST MEMBER

Name-Surname: Ali Emre Nebiler
Birthdate and Place of Birth: 10.10.2000, Manisa
E-mail: emre.nebiler@std.yildiz.edu.tr
Phone: 0533 254 41 12
Practical Training:
- SabancıDx, Software Developer Intern
- Nokia, IP Network Automation Intern

Project System Informations

System and Software: Windows/MacOS/Linux, R and Python
Required RAM: 8GB
Required Disk: 5GB