Technological Educational Institute of Crete
School of Engineering
Department of Informatics Engineering

Gene expression and gene regulatory network analysis with statistical methods and machine learning algorithms

Student: Droumalia Fotini TP4766

Supervisor: Koumakis Lefteris

Objectives of the Research

- Implement and analyze available pathway scoring techniques in order to confirm that non-binary expression data are more efficient
- Discover any non-binary pathway analysis tools that could be used for real-time/online predictions
- Comparing statistical and machine learning approaches to pathway analysis

Molecular Biology

- The field of molecular biology is concerned with the molecular foundations of biological activity
- It is focused on genes and proteins, and the interactions among these molecules
- It includes a variety of techniques designed to help researchers comprehend how molecules operate.

DNA

- It contains all the genetic materials required for an organism to develop and function
- Its design resembles a double helix, which is a twisted ladder formed by two connected strings that spiral around one another
- The central dogma of molecular biology is the process of generating functional cells in compliance with DNA instructions

Gene

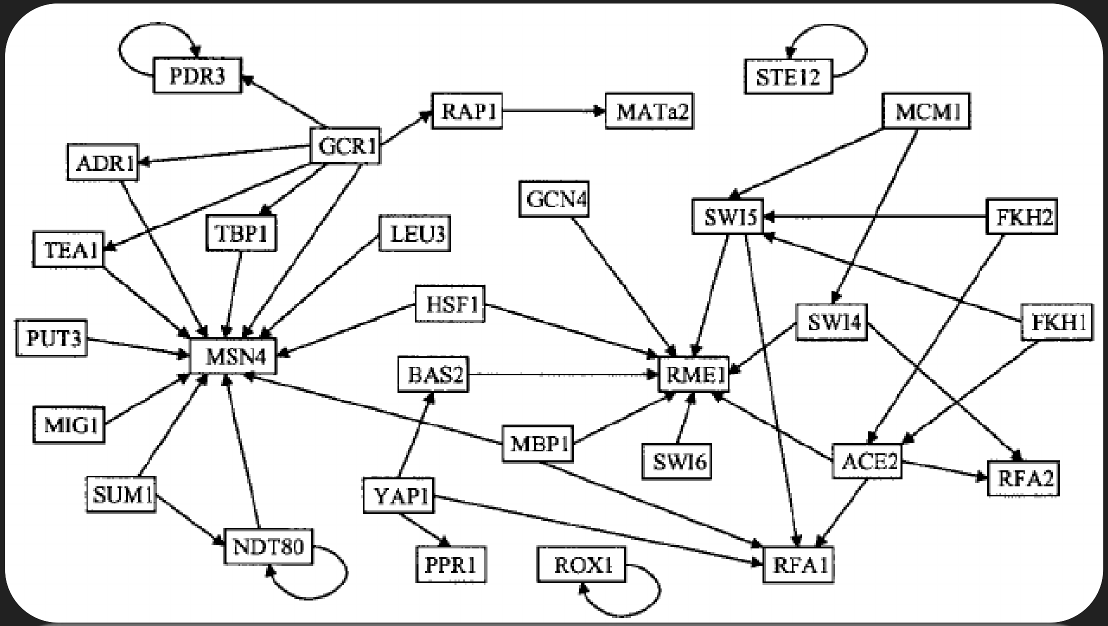
- It is made up of DNA and serves as the basic structural and functional unit of inheritance
- It is important for the synthesis of proteins, which are vital for the development, function, and regulation of the body's tissues and organs
- The entire genetic code of an organism is known as genome
- The entire genome was successfully sequenced by the Human Genome Project in 2022

Gene Expression

- The procedure by which genes control protein synthesis, that is when and where the RNA molecules and proteins are produced, to achieve the desirable results
- Genes are typically classified as "expressed" or "non-expressed" depending on whether a cell generates the protein that a gene encodes or not

Gene Regulatory Networks I

- They are sets of genes that interact with one another and can be described with the usage of graphs
- Every gene is represented by a node and a pathway's edge denotes any connection or other kind of interaction between the nodes



Gene Regulatory Networks II

- To represent these networks, mathematical and computational models can be used to explain the mechanisms behind the regulatory activities performed by genes
- They are useful for:
  1. Identifying circuits that can be used for a certain task
  2. Understand the flow of information in biological systems
  3. Predict alterations in gene expression under various conditions

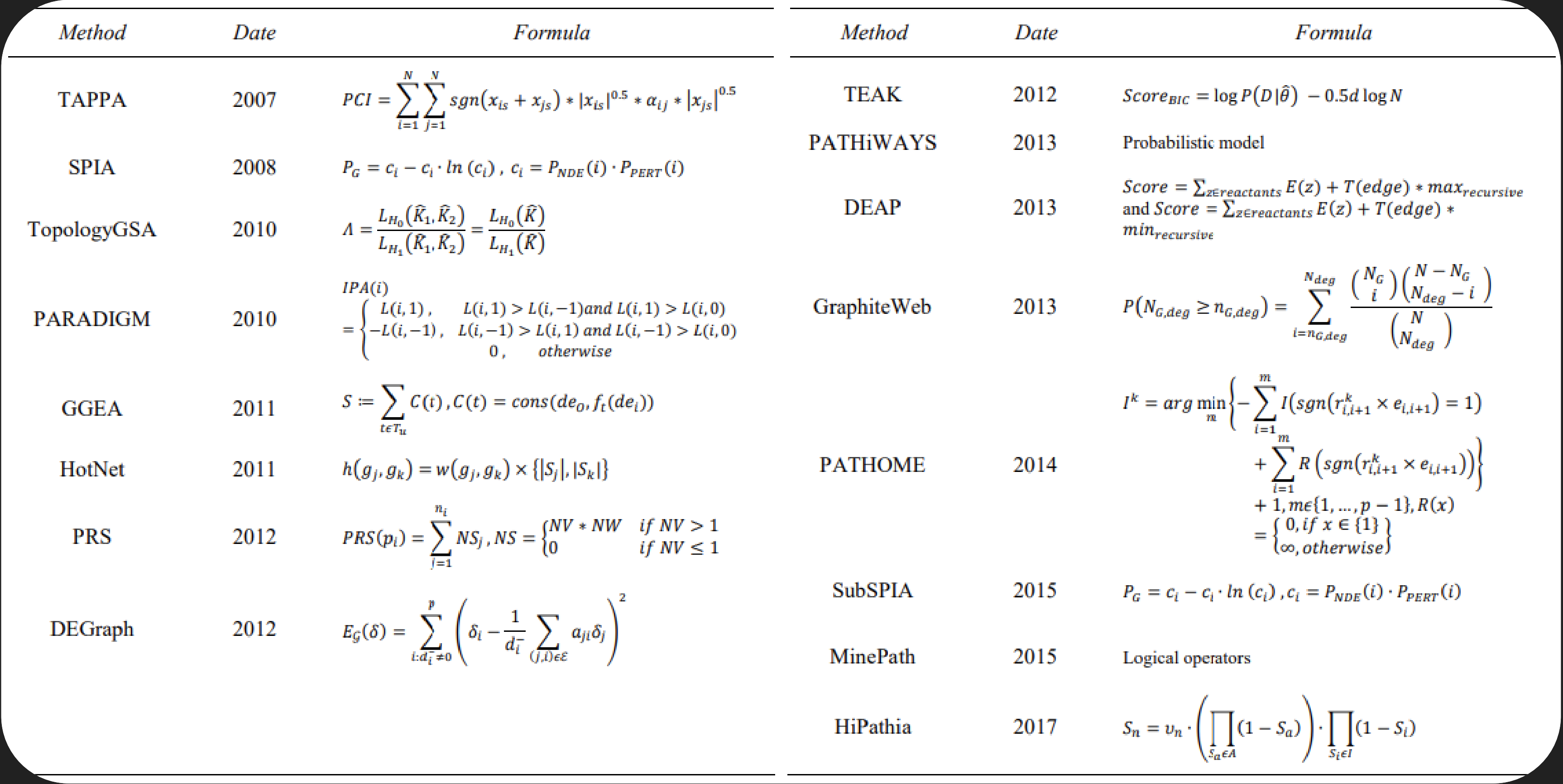
Pathway Analysis

- It utilizes data from both molecular pathway networks and gene expression analysis to determine significantly affected pathways in a particular scenario and to decipher the biological significance of differentially expressed genes and proteins
- The word "Pathway" describes the network of molecular interactions, reactions, and relationships as it appears graphically
- The score of a pathway indicates the extent to which a pathway differed between the two phenotypes
- Two main approaches for pathway analysis:
    1. Non-topology-based: perceive the pathways as plain collections of genes
    2. Topology-based: take into consideration the arrangement of genes

Pathway Analysis Tools

- The main object of study of this research was the pathway scoring methodology of several pathway analysis tools
- A total of 16 tools' scoring techniques were studied, however only 7 were selected for implementation due to their simplicity

Pathway Analysis Tools – Scoring Formulas



Method	*Date*	*Formula*
TAPPA	2007	$PCI = \sum_{i=1}^{N} \sum_{j=1}^{N} sgn(x_{is} + x_{js}) * \lvert x_{is} \rvert^{0.5} * \alpha_{ij} * \lvert x_{js} \rvert^{0.5}$
SPIA	2008	$P_G = c_i - c_i \cdot ln\,(c_i)\,,\ c_i = P_{NDE}(i) \cdot P_{PERT}(i)$
TopologyGSA	2010	$\Lambda = \frac{L_{H_0}(\widehat{K}_1, \widehat{K}_2)}{L_{H_1}(\widehat{K}_1, \widehat{K}_2)} = \frac{L_{H_0}(\widehat{K})}{L_{H_1}(\widehat{K})}$
PARADIGM	2010	$IPA(i) = \begin{cases} L(i,1)\,, & L(i,1) > L(i,-1) and\ L(i,1) > L(i,0) \\ -L(i,-1)\,, & L(i,-1) > L(i,1)\ and\ L(i,-1) > L(i,0) \\ 0\,, & otherwise \end{cases}$
GGEA	2011	$S := \sum_{t \epsilon T_u} C(t)\,, C(t) = cons(de_O, f_t(de_i))$
HotNet	2011	$h(g_j, g_k) = w(g_j, g_k) \times \{\lvert S_j \rvert, \lvert S_k \rvert\}$
PRS	2012	$PRS(p_i) = \sum_{j=1}^{n_i} NS_j\,, NS = \begin{cases} NV * NW & if\ NV > 1 \\ 0 & if\ NV \leq 1 \end{cases}$
DEGraph	2012	$E_G(\delta) = \sum_{i:d_i^- \neq 0}^{p} \left(\delta_i - \frac{1}{d_i^-} \sum_{(j,i) \epsilon \mathcal{E}} a_{ji} \delta_j \right)^2$
TEAK	2012	$Score_{BIC} = \log P(D \mid \hat{\theta}) - 0.5d \log N$
PATHiWAYS	2013	Probabilistic model
DEAP	2013	$Score = \sum_{z \in reactants} E(z) + T(edge) * max_{recursive}$ and $Score = \sum_{z \in reactants} E(z) + T(edge) * min_{recursive}$
GraphiteWeb	2013	$P(N_{G,deg} \geq n_{G,deg}) = \sum_{i=n_{G,deg}}^{N_{deg}} \frac{\binom{N_G}{i}\binom{N - N_G}{N_{deg} - i}}{\binom{N}{N_{deg}}}$
PATHOME	2014	$I^k = arg \min_m \left\{ -\sum_{i=1}^{m} I(sgn(r_{i,i+1}^k \times e_{i,i+1}) = 1) + \sum_{i=1}^{m} R\left(sgn(r_{i,i+1}^k \times e_{i,i+1})\right) \right\} + 1, m \epsilon \{1, \ldots, p-1\}, R(x) = \begin{cases} 0, if\ x \in \{1\} \\ \infty, otherwise \end{cases}$
SubSPIA	2015	$P_G = c_i - c_i \cdot ln\,(c_i)\,, c_i = P_{NDE}(i) \cdot P_{PERT}(i)$
MinePath	2015	Logical operators
HiPathia	2017	$S_n = v_n \cdot \left(\prod_{S_a \epsilon A} (1 - S_a) \right) \cdot \prod_{S_i \epsilon I} (1 - S_i)$

Datasets

1. The **GSE2034 gene expression dataset** that contains 286 breast cancer samples, of which 209 are ER-positive and 77 are ER-negative (https://ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=gse2034)
2. A **collection of 47 sub-paths** in total of which 15 relate to cellular activities, 24 to signal propagation and 8 to cancer in general (https://www.genome.jp/kegg-bin/show_organism?menu_type=pathway_maps&org=hsa)

Data Preprocessing – GSE2034

- Since a gene might be mapped to multiple Entrez identifiers, each Entrez identifier is translated to the corresponding gene
- Pathway genes and gene IDs that don't match the dataset GSE2034 entries are assigned as 'no-Probe'
- The average expression value of each KEGG ID is determined by combining all of the KEGG IDs' expression values into a single gene

Data Preprocessing – Subpathways

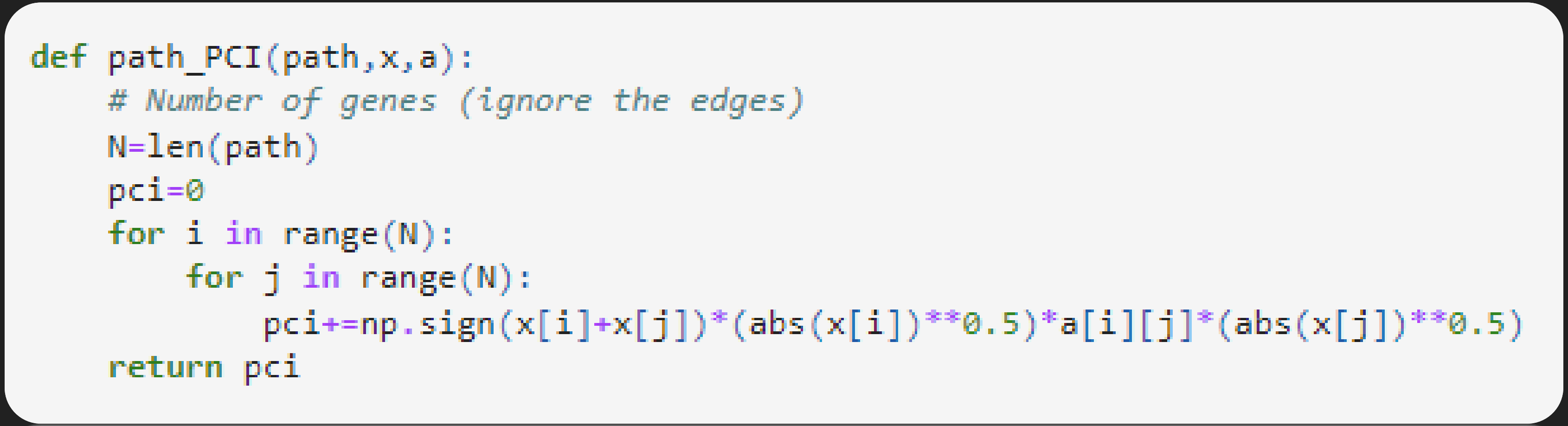
- Each sub-path of the dataset is divided by the nodes and edges
- Since each node may contain more than one genes, the average value is determined and assigned
- Data from the GSE2034 dataset are combined with the sub-paths’ dataset and create new collective data structures to simplify the analysis process

Score Calculation

- TAPPA, PRS, TEAK, GraphiteWeb, MinePath and HiPathia are the methods implemented in this work according to each tool's respective papers using the Python programming language
- All tools, except MinePath and GraphiteWeb, handle non-binary data
- Pathway topology is taken into account by all methodologies, however there are two approaches to the problem:
    1. TAPPA, GraphiteWeb, TEAK and PRS are based on probability theories and comprehend pathway topology as the influence of nodes upon each another
    2. HiPathia, DEAP and MinePath rely on the interaction type among genes and how it affects the outcome
- A two-dimensional matrix with rows denoting samples and columns indicating sub-paths formed for each tool after computing the score of each sample and each sub-path in accordance with the methodology of the associated tool
- The source code is available at https://github.com/fotinidrouma/Pathway-Analysis.git

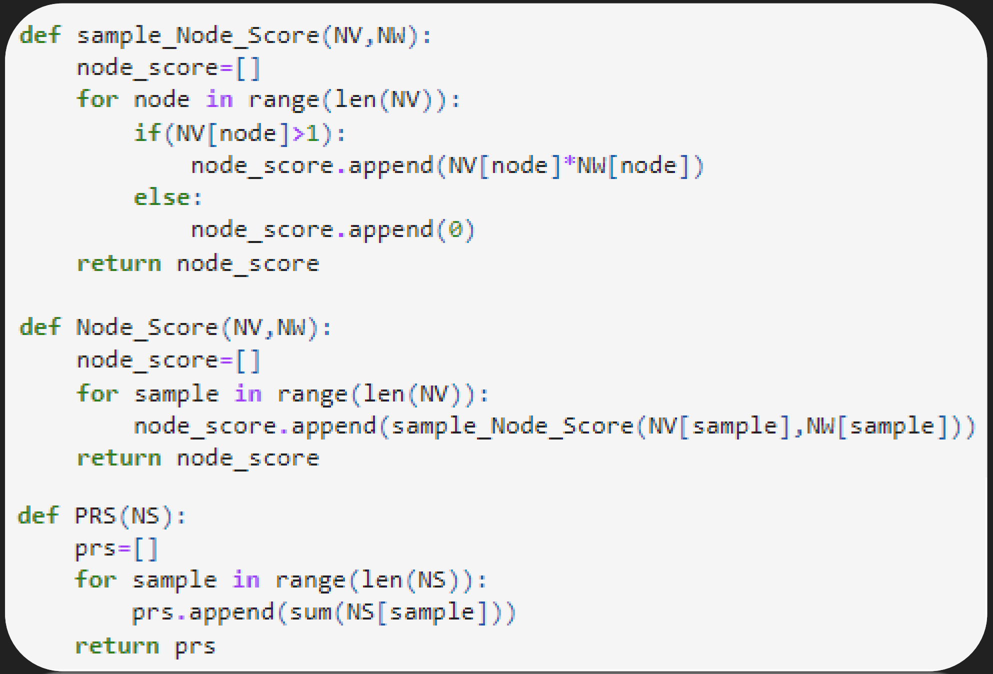
Score Calculation - TAPPA

For a given sub-path and each sample, the variable x represents the standard log expression estimate of the genes, and the respective adjacency matrices are denoted as a.



```python
def path_PCI(path,x,a):
    # Number of genes (ignore the edges)
    N=len(path)
    pci=0
    for i in range(N):
        for j in range(N):
            pci+=np.sign(x[i]+x[j])*(abs(x[i])**0.5)*a[i][j]*(abs(x[j])**0.5)
    return pci
```

Score Calculation - PRS

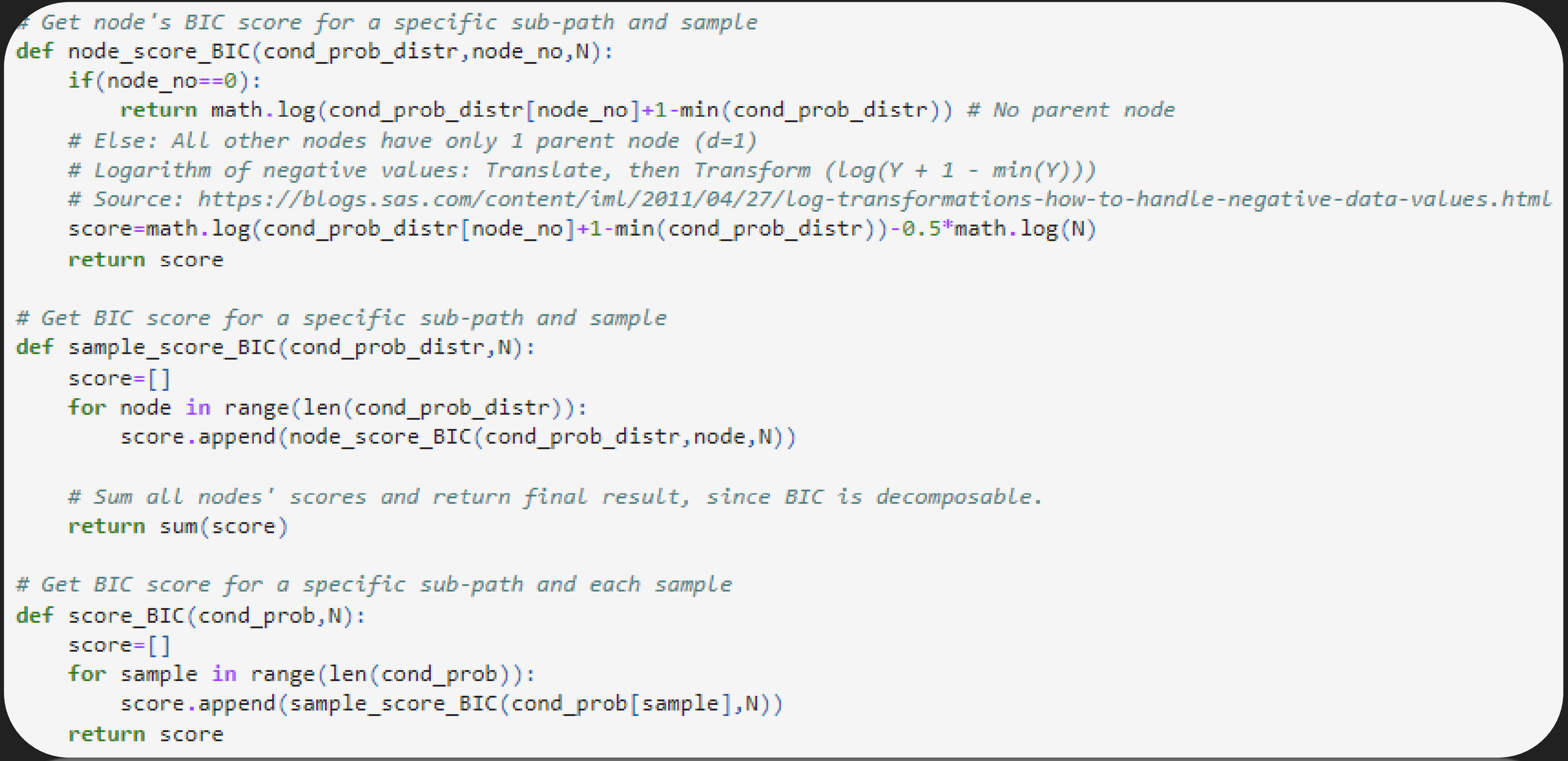


```
def sample_Node_Score(NV,NW):
    node_score=[]
    for node in range(len(NV)):
        if(NV[node]>1):
            node_score.append(NV[node]*NW[node])
        else:
            node_score.append(0)
    return node_score

def Node_Score(NV,NW):
    node_score=[]
    for sample in range(len(NV)):
        node_score.append(sample_Node_Score(NV[sample],NW[sample]))
    return node_score

def PRS(NS):
    prs=[]
    for sample in range(len(NS)):
        prs.append(sum(NS[sample]))
    return prs
```

Using the corresponding node values and weights, the pathway score is produced by summing the scores of each node. The non-parametric permutation method of the second normalization step of PRS was replaced with FDR correction.

Score Calculation - TEAK

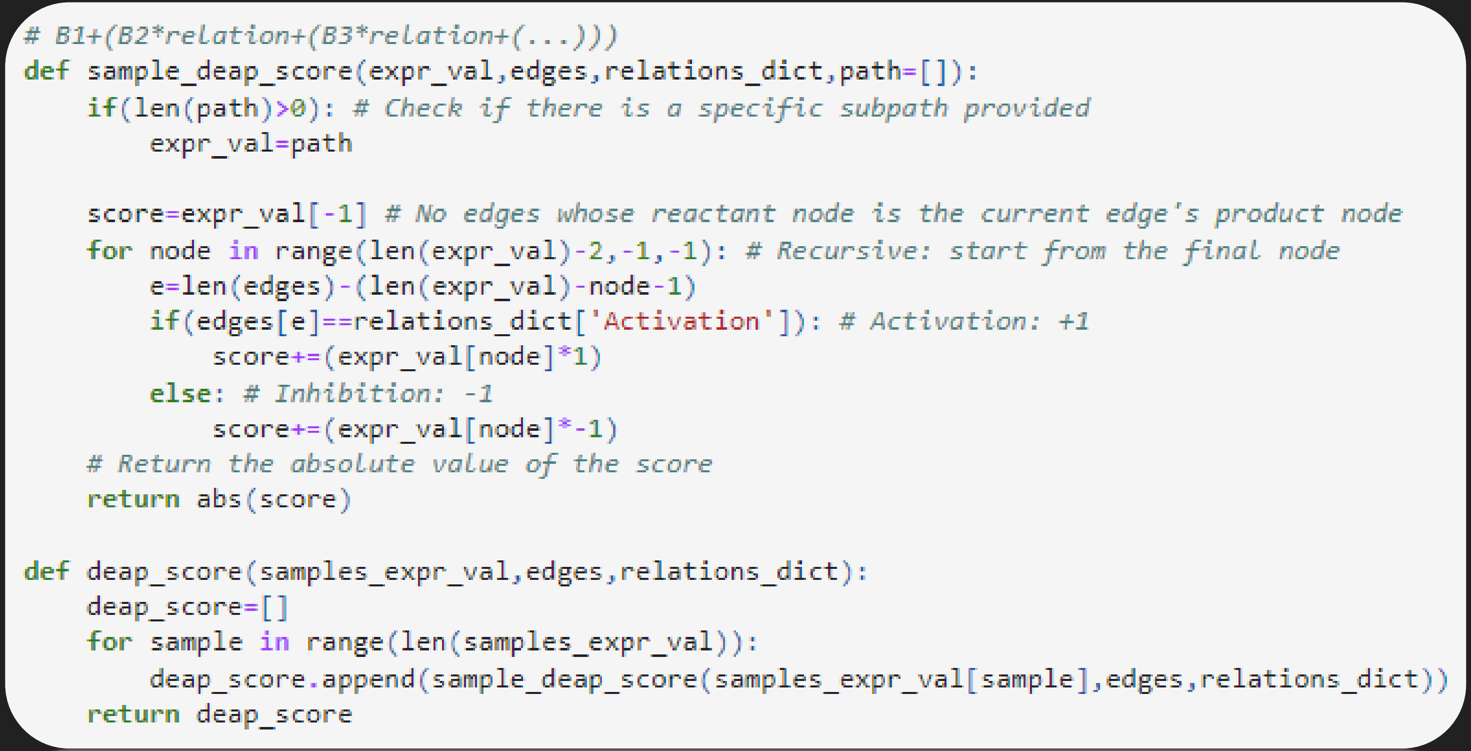
N refers to the number of samples in the expression data, while cond_prob_distr stands for Conditional Probability Distribution.



```python
# Get node's BIC score for a specific sub-path and sample
def node_score_BIC(cond_prob_distr,node_no,N):
    if(node_no==0):
        return math.log(cond_prob_distr[node_no]+1-min(cond_prob_distr)) # No parent node
    # Else: All other nodes have only 1 parent node (d=1)
    # Logarithm of negative values: Translate, then Transform (log(Y + 1 - min(Y)))
    # Source: https://blogs.sas.com/content/iml/2011/04/27/log-transformations-how-to-handle-negative-data-values.html
    score=math.log(cond_prob_distr[node_no]+1-min(cond_prob_distr))-0.5*math.log(N)
    return score

# Get BIC score for a specific sub-path and sample
def sample_score_BIC(cond_prob_distr,N):
    score=[]
    for node in range(len(cond_prob_distr)):
        score.append(node_score_BIC(cond_prob_distr,node,N))

    # Sum all nodes' scores and return final result, since BIC is decomposable.
    return sum(score)

# Get BIC score for a specific sub-path and each sample
def score_BIC(cond_prob,N):
    score=[]
    for sample in range(len(cond_prob)):
        score.append(sample_score_BIC(cond_prob[sample],N))
    return score
```

Score Calculation - DEAP

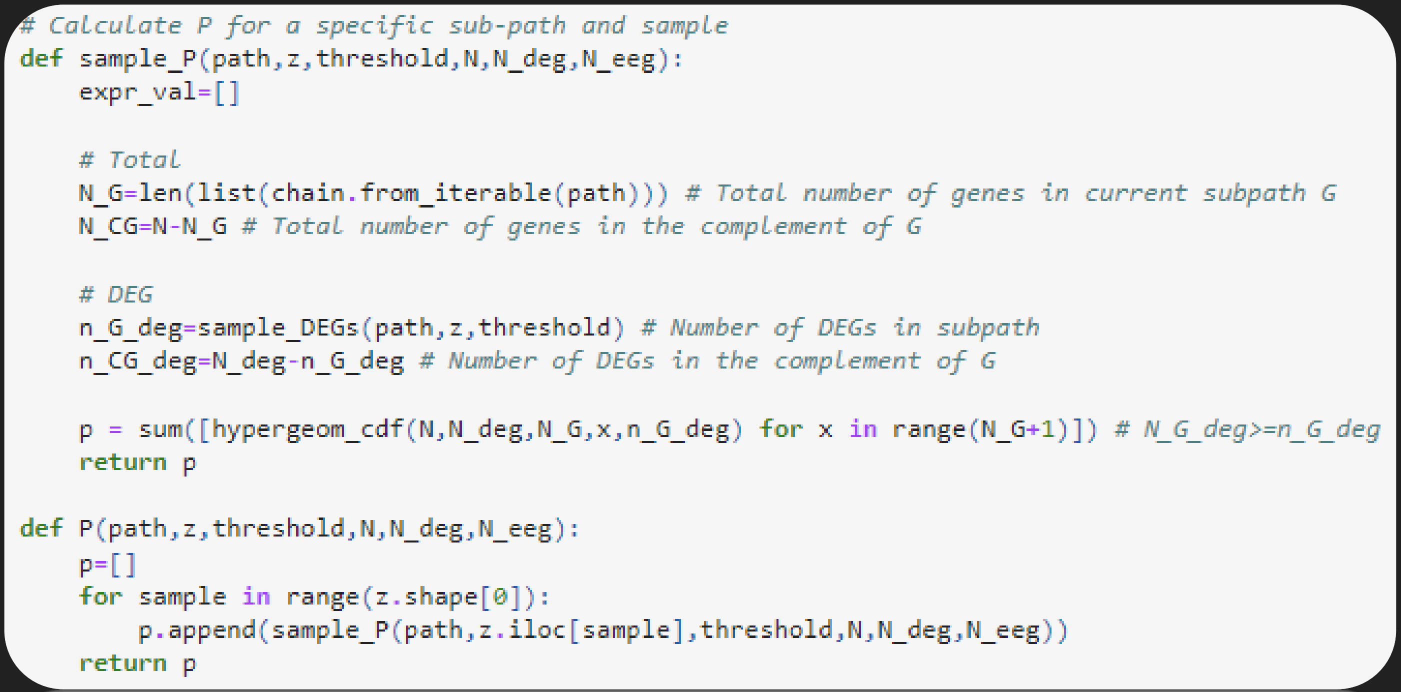
Through a recursive function, the type of gene interaction is considered. Instead of employing DEAP's random rotation method, FDR correction was employed.



```python
# B1+(B2*relation+(B3*relation+(...)))
def sample_deap_score(expr_val,edges,relations_dict,path=[]):
    if(len(path)>0): # Check if there is a specific subpath provided
        expr_val=path

    score=expr_val[-1] # No edges whose reactant node is the current edge's product node
    for node in range(len(expr_val)-2,-1,-1): # Recursive: start from the final node
        e=len(edges)-(len(expr_val)-node-1)
        if(edges[e]==relations_dict['Activation']): # Activation: +1
            score+=(expr_val[node]*1)
        else: # Inhibition: -1
            score+=(expr_val[node]*-1)
    # Return the absolute value of the score
    return abs(score)

def deap_score(samples_expr_val,edges,relations_dict):
    deap_score=[]
    for sample in range(len(samples_expr_val)):
        deap_score.append(sample_deap_score(samples_expr_val[sample],edges,relations_dict))
    return deap_score
```

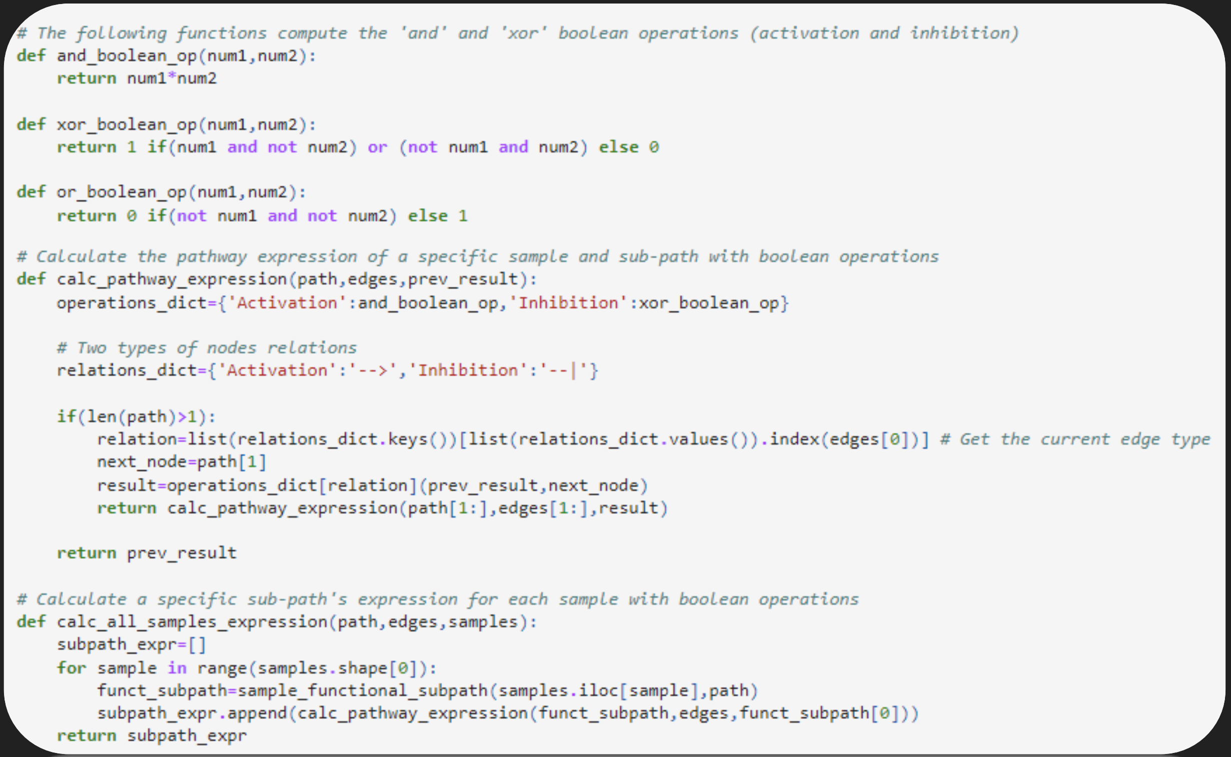
Score Calculation - GraphiteWeb

N stands for the overall number of genes screened, N_deg refers to the amount of differentially expressed genes, and N_eeg indicates the total amount of equally expressed genes.



```python
# Calculate P for a specific sub-path and sample
def sample_P(path,z,threshold,N,N_deg,N_eeg):
    expr_val=[]

    # Total
    N_G=len(list(chain.from_iterable(path))) # Total number of genes in current subpath G
    N_CG=N-N_G # Total number of genes in the complement of G

    # DEG
    n_G_deg=sample_DEGs(path,z,threshold) # Number of DEGs in subpath
    n_CG_deg=N_deg-n_G_deg # Number of DEGs in the complement of G

    p = sum([hypergeom_cdf(N,N_deg,N_G,x,n_G_deg) for x in range(N_G+1)]) # N_G_deg>=n_G_deg
    return p

def P(path,z,threshold,N,N_deg,N_eeg):
    p=[]
    for sample in range(z.shape[0]):
        p.append(sample_P(path,z.iloc[sample],threshold,N,N_deg,N_eeg))
    return p
```

Score Calculation – MinePath

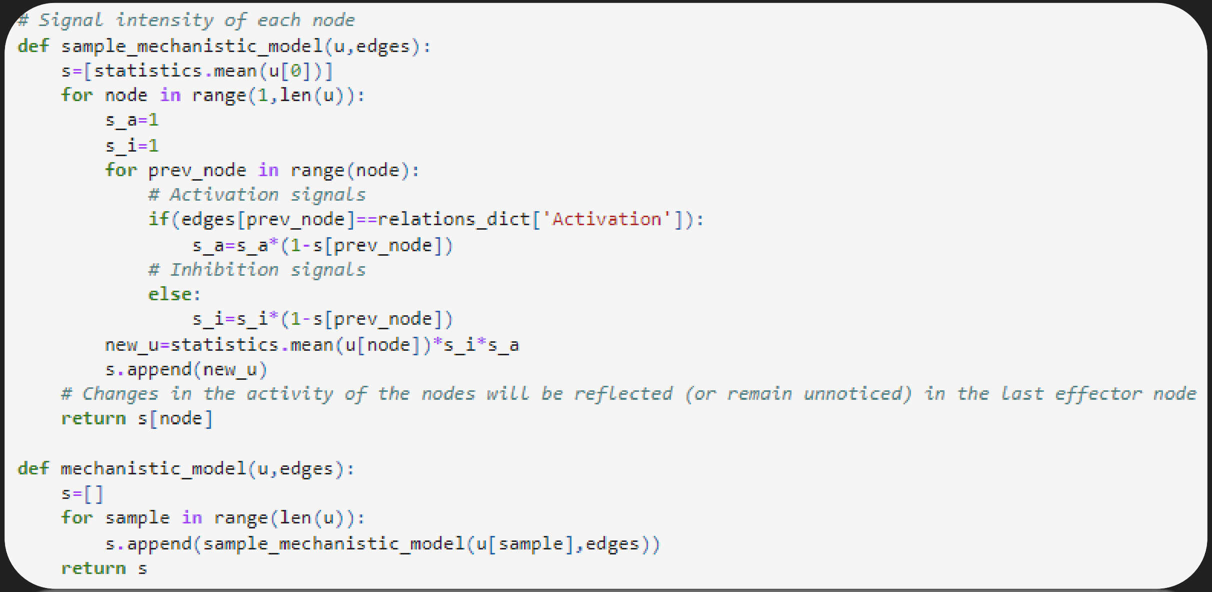


```python
# The following functions compute the 'and' and 'xor' boolean operations (activation and inhibition)
def and_boolean_op(num1,num2):
    return num1*num2

def xor_boolean_op(num1,num2):
    return 1 if(num1 and not num2) or (not num1 and num2) else 0

def or_boolean_op(num1,num2):
    return 0 if(not num1 and not num2) else 1

# Calculate the pathway expression of a specific sample and sub-path with boolean operations
def calc_pathway_expression(path,edges,prev_result):
    operations_dict={'Activation':and_boolean_op,'Inhibition':xor_boolean_op}

    # Two types of nodes relations
    relations_dict={'Activation':'-->','Inhibition':'--|'}

    if(len(path)>1):
        relation=list(relations_dict.keys())[list(relations_dict.values()).index(edges[0])] # Get the current edge type
        next_node=path[1]
        result=operations_dict[relation](prev_result,next_node)
        return calc_pathway_expression(path[1:],edges[1:],result)

    return prev_result

# Calculate a specific sub-path's expression for each sample with boolean operations
def calc_all_samples_expression(path,edges,samples):
    subpath_expr=[]
    for sample in range(samples.shape[0]):
        funct_subpath=sample_functional_subpath(samples.iloc[sample],path)
        subpath_expr.append(calc_pathway_expression(funct_subpath,edges,funct_subpath[0]))
    return subpath_expr
```

Depending on the interactions between genes, MinePath uses Boolean operators to handle binary expression values.

Score Calculation – HiPathia

The normalized gene expression values for a specific sub-path and sample are indicated by the variable u, which is necessary for calculating the score.



```
# Signal intensity of each node
def sample_mechanistic_model(u,edges):
    s=[statistics.mean(u[0])]
    for node in range(1,len(u)):
        s_a=1
        s_i=1
        for prev_node in range(node):
            # Activation signals
            if(edges[prev_node]==relations_dict['Activation']):
                s_a=s_a*(1-s[prev_node])
            # Inhibition signals
            else:
                s_i=s_i*(1-s[prev_node])
        new_u=statistics.mean(u[node])*s_i*s_a
        s.append(new_u)
    # Changes in the activity of the nodes will be reflected (or remain unnoticed) in the last effector node
    return s[node]

def mechanistic_model(u,edges):
    s=[]
    for sample in range(len(u)):
        s.append(sample_mechanistic_model(u[sample],edges))
    return s
```

Decision Tree Algorithm

- In order to compare the results of each methodology, the **Decision Tree algorithm** was utilized to train and assess each approach‘s performance and recognize the most significant sub-paths
- A Decision Tree algorithm’s basis is the continuous division of the data by a specified criterion
- Decision nodes and leaves illustrate the tree’s structure
    - Decision nodes: separate the data
    - Leaves: represent the options or results

Subpathway Ranking

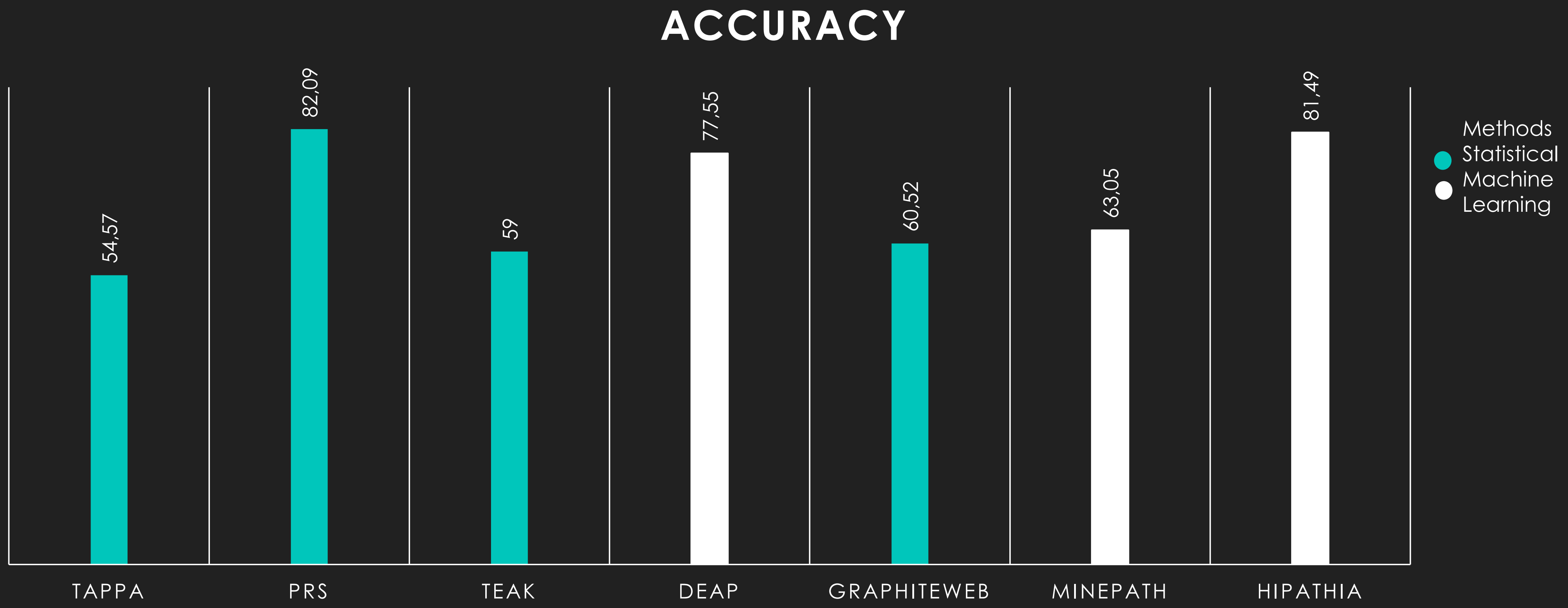
1. Because machine learning algorithms can only handle numerical values, the categorical classes of the samples of the datasets were converted into numerical variables
2. The updated datasets were split into training and testing sets in a proportion of 70% to 30%
3. Using the pre-built decision tree functions that the scikit-learn package provides for the Python programming language, the training sets that resulted from the prior stage were used to train each machine learning model, and the respective testing sets were used to evaluate each model

Predictive Performance I

- In the case of GraphiteWeb, the classification model's accuracy decreased when the results were adjusted using the Benjamini and Hochberg method. Such reduction can be explained by an increase in the ability to adjust to novel, previously unseen data
- In terms of the significance of each methodology's feature, no common sub-paths could be found among the tools

Tools	Accuracy
TAPPA	54,57%
PRS	**82,09%**
TEAK	59,00%
DEAP	77,55%
GraphiteWeb	60,52%
MinePath	63,05%
HiPathia	81,49%

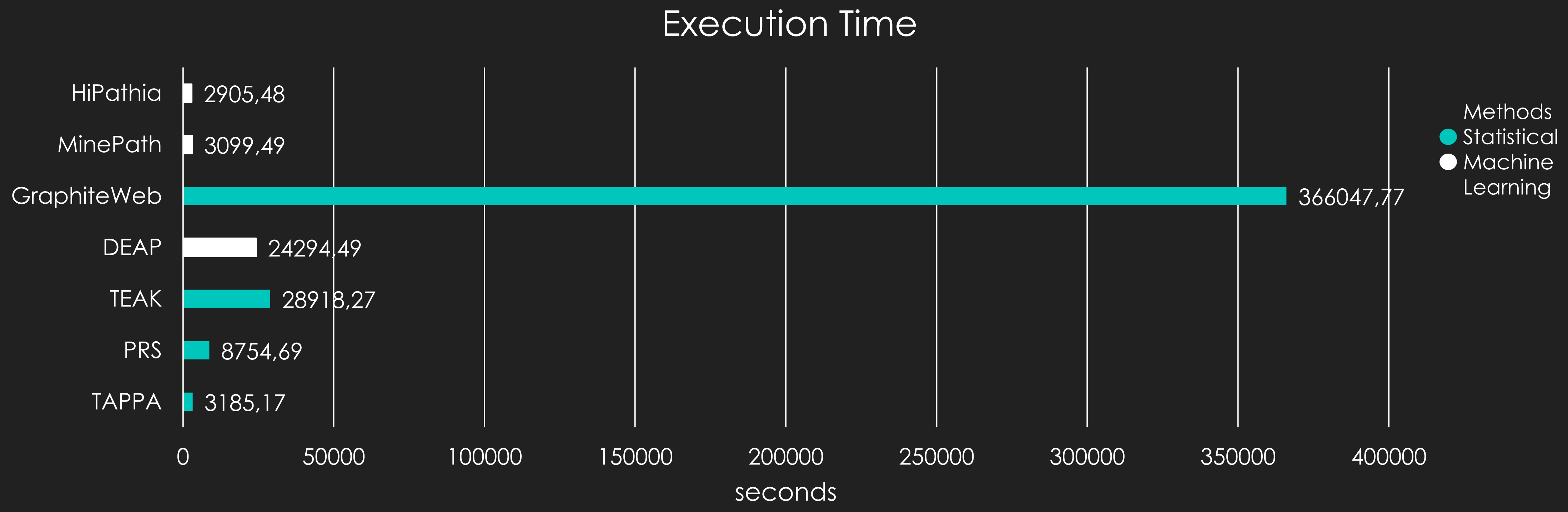
Predictive Performance II



Execution Time I

Tools	Execution Time (sec)
TAPPA	3185,17
PRS	8754,69
TEAK	28918,27
DEAP	24294,49
GraphiteWeb	366047,77
MinePath	3099,49
HiPathia	**2905,48**

Execution Time II



Conclusions

- HiPathia which displayed high accuracy rate and the lowest execution time, is the best option for online analysis
- The findings suggest that machine learning-based techniques outperform statistical methods in terms of performance and speed, since they can handle huge amounts of data better
- Most of the studied tools handle non-binary gene expression levels, confirming their preference for continuous values, because they retain information that may be valuable

Limitations

- The abundance of genomic platforms provide inconsistent results due to differences in platform design, synthesis, and probe annotation
- Machine learning in healthcare presents numerous issues, like poor data quality and ethical dilemmas
- The quantity and variety of tools employed in this study are insufficient to fully support the findings

Future Work

- Conduct further research while taking into consideration proportionately additional tools from each approach
- Use the scripts presented in this paper to analyze gene expression data from various genomic platforms
- Focus on non-binary gene expression data and primarily employ machine learning to study gene expression and gene regulatory networks
- Integrate GraphiteWeb's methodology with HiPathia's scoring algorithm to accelerate execution

Thank You!

For Your Attention