

LIST OF PUBLICATIONS

Papers Published in International Conference Proceedings

- “Decision Tree Based Model for the Classification of Pathogenic Gene Sequences Causing ASD”, International Conference on Smart Trends for Information Technology and Computer Communications, SmartCom 2017 conducted during 18-19 August 2017, Communications in Computer and Information Science, vol. 876. pp 201-212, Springer, Singapore (**Scopus indexed**)
- “Machine Learning-Based Model for Identification of Syndromic Autism Spectrum Disorder”, International Conference on Integrated Intelligent Computing, Communication and Security, conducted during 24 - 25 Jan 2018, Studies in Computational Intelligence, vol. 771. pp. 141-148, Springer, Singapore (**Scopus indexed**)

Papers Published / Accepted in International Journals

- “Identification of Autism Spectrum Disorder using a Multi-Label Approach”, Journal of Advanced Research and Dynamical Control Systems, Vol. 11, Issue – 2, pp. 134-141, 2019, (**Scopus indexed**)
- “Bidirectional Recurrent Neural Network based Model to identify ASD” – Accepted for publication in the Journal of Advances in Modelling and Analysis A. (**Scopus indexed**)

Papers in Review - International Journals

- “Codon Encoding of Gene Sequences for Deep Learning Based Prediction of Autism Spectrum Disorder” in the Journal of Computer Science
- “Long Short Term Memory Recurrent Neural Networks for Autism Spectrum Disorder Prediction using Codon Encoding” in the Turkish Journal of Electrical Engineering and Computer Sciences
- “Gated Recurrent Neural Network for ASD Gene Prediction using Codon encoding” in the Periodicals of Engineering and Natural Science
- “Efficient Prediction of Gene Susceptibility to Autism Spectrum Disorder using Deep Architectures” in the International Journal of Advanced Intelligence Paradigms (IJAIIP)
- “Recurrent Neural Network Based Models to predict Autism Spectrum Disorder Causative Genes” in the Arabian Journal for Science and Engineering

APPENDIX A - DATASETS

cDNA Sequence of MECP2 Gene

>NM_004992.3 Homo sapiens methyl-CpG binding protein 2 (MECP2), transcript variant 1, mRNA

```

ATGGTAGCTGGGATGTTAGGGGCTCAGGGAAGAAAAGTCAGAAGACCAGGACCTCCAGGGCCTCAAGG
ACAAACCCCTCAAGTTTAAAAAGGTGAAGAAAGATAAGAAAGAAGAGAAAGAGGGCAAGCATGAGCC
CGTGCAGCCATCAGCCCACCACTCTGCTGAGCCCCGAGAGGCAGGCAAAGCAGAGACATCAGAAGGGT
CAGGCTCCGCCCCGGCTGTGCCGGAAGCTTCTGCCTCCCCAAACAGCGGCGCTCCATCATCCGTGACC
GGGGACCCATGTATGATGACCCACCTGCCTGAAGGCTGGACACGGAAGCTTAAGCAAAGGAAATCT
GGCCGCTCTGCTGGGAAGTATGATGTGATTTGATCAATCCCAGGGAAAAGCCTTTAGCTCTAAAGTG
GAGTTGATTGCGTACTTCGAAAAGGTAGGCGACACATCCCTGGACCCTAATGATTTTGACTTCACGGTAA
CTGGGAGAGGGAGCCCCTCCGGCGAGAGCAGAAACCACCTAAGAAGCCCAAATCTCCCAAAGCTCCA
GGAAGTGGCAGAGGCCGGGGACGCCCAAAGGGAGCGGCACCACGAGACCCAAGGCGGCCACGTGAG
AGGGTGTGCAGGTGAAAAGGGTCTGGAGAAAAGTCTGGGAAGCTCCTTGCAAGATGCCTTTTCAA
ACTTCGCCAGGGGGCAAGGCTGAGGGGGGTGGGGCCACCACATCCACCCAGGTCATGGTGATCAAACG
CCCCGGCAGGAAGCGAAAAGCTGAGGCCGACCCTCAGGCCATTCCCAAGAAACGGGGCCGAAAGCCG
GGGAGTGTGGTGGCAGCCGCTGCCGCCGAGGCCAAAAGAAAGCCGTGAAGGAGTCTTCTATCCGATC
TGTGCAGGAGACCGTACTCCCCATCAAGAAGCGCAAGACCCGGGAGACGGTCAGCATCGAGGTCAAGG
AAGTGGTGAAGCCCCTGCTGGTGTCCACCTCGGTGAGAAGAGCGGGAAAGGACTGAAGACCTGTAAG
AGCCCTGGGCGGAAAAGCAAGGAGAGCAGCCCCAAGGGGCGCAGCAGCAGCGCCTCCTCACCCCCAA
    
```

Mutational information for MECP2 Gene from HGMD

The screenshot shows the HGMD website interface. At the top, there are navigation links and a search bar. The main content area displays the gene symbol **MECP2** and a summary of mutation types:

Misense/nonsense	Splicing	Regulatory	Small deletions	Small insertions	Small indels	Gross deletions	Gross insertions	Complex	Repeats
236 mutations in HGMD (professional 2019.1)	19 mutations in HGMD (professional 2019.1)	1 mutation in HGMD (professional 2019.1)	127 mutations in HGMD (professional 2019.1)	65 mutations in HGMD (professional 2019.1)	12 mutations in HGMD (professional 2019.1)	297 mutations in HGMD (professional 2019.1)	278 mutations in HGMD (professional 2019.1)	*4 mutations in HGMD (professional 2019.1)	No mutations

Further options available in [HGMD \(professional 2019.1\)](#)

Accession Number	Codon change	Amino acid change	Codon number	Genomic coordinates & HGVS nomenclature	Phenotype	Reference	Comments
CM058034	RTG-TTG	Met-Leu	1	Available to subscribers GACGTCAGGACCTCCAGGGCCTCAAGG	Rett syndrome	Gauthier (2005) Can J Neurol Sci 32, 321 Additional phenotype report available to subscribers	
CM032595	GAA-CAA	Glu-Gln	10	Available to subscribers GAGAGCAGAAACCACCTAAGAAGCCCAAATCTCCCAAAGCTCCA	Rett syndrome ?	Smeets (2003) Am J Med Genet A 122A, 227	
CM060329	GAA-TAA	Glu-Term	10	Available to subscribers GAGAGCAGAAACCACCTAAGAAGCCCAAATCTCCCAAAGCTCCA	Rett syndrome	Philippe (2006) Eur J Med Genet 49, 9	
CM023409	CAG-TAG	Gln-Term	16	Available to subscribers GAGGCGACACATCCCTGGACCCTAATGATTTTGACTTCACGGTAA	Rett syndrome	RSRF (2002) MECP2 UP	
HM071529	CAG-TAG	Gln-Term	19	Available to subscribers GAGGCGACACATCCCTGGACCCTAATGATTTTGACTTCACGGTAA	Rett syndrome	Cook (2000) Hum Genet 106, 238 Additional phenotype report available to subscribers	
CM010332	AAG-TAG	Lys-Term	22	Available to subscribers GAGGCGACACATCCCTGGACCCTAATGATTTTGACTTCACGGTAA	Rett syndrome	Vacca (2001) J Mol Med (Berl) 78, 648	
CM057720	CAG-TAG	Gln-Term	47	Available to subscribers GAGGCGACACATCCCTGGACCCTAATGATTTTGACTTCACGGTAA	Rett syndrome	Charman (2005) Eur J Hum Genet 13, 1121	suppl. table...
CM010333	TCA-TAA	Ser-Term	49	Available to subscribers GAGGCGACACATCCCTGGACCCTAATGATTTTGACTTCACGGTAA	Rett syndrome	Boardman (2001) Hum Genet 108, 43	

Transferring data from www.hgmd.cf.ac.uk

Missense Mutated MECP2 Gene

>NM_004992.3 Homo sapiens methyl-CpG binding protein 2 (MECP2), transcript variant 1, mRNA

TTGGTAGCTGGGATGTTAGGGCTCAGGGAAGAAAAGTCAGAAGACCAGGACCTCCAGGGCCTCAAGGA
CAAACCCCTCAAGTTTAAAAAGGTGAAGAAAGATAAGAAAAGAAGAGAAAAGAGGGCAAGCATGAGCCC
GTGCAGCCATCAGCCCACCACTGCTGAGCCCGCAGAGGCAGGCAAAGCAGAGACATCAGAAGGGTC
AGGCTCCGCCCCGGCTGTGCCGAAGCTTCTGCCTCCCCAACAGCGGCGCTCCATCATCCGTGACCG
GGGACCCATGTATGATGACCCACCCCTGCCTGAAGGCTGGACACGGAAGCTTAAGCAAAGGAAATCTG
GCCGCTCTGCTGGGAAGTATGATGTGTATTTGATCAATCCCCAGGGAAAAGCCTTTAGCTCTAAAGTGG
AGTTGATTGCGTACTTCGAAAAGGTAGGGCGACACATCCCTGGACCCTAATGATTTTGACTTCACGGTAAC
TGGGAGAGGGAGCCCCTCCCGGCGAGAGCAGAAACCACTAAGAAGCCCAAATCTCCCAAAGCTCCAG
GAACTGGCAGAGGCCGGGGACGCCCCAAAGGGAGCGGCACCACGAGACCCAAGGCGGCCACGTCAGA
GGGTGTGCAGGTGAAAAGGGTCCTGGAGAAAAGTCCTGGGAAGCTCCTTGTCAAGATGCCTTTTCAA
CTTCGCCAGGGGGCAAGGCTGAGGGGGGTGGGGCCACCACATCCACCCAGGTCATGGTGATCAAACGC
CCCGGCAGGAAGCGAAAAGCTGAGGCCGACCCCTCAGGCCATTCCCAAGAAACGGGGCCGAAAGCCGG
GGAGTGTGGTGGCAGCCGCTGCCGCCGAGGCCAAAAGAAAGCCGTGAAGGAGTCTTCTATCCGATCT
GTGCAGGAGACCGTACTCCCATCAAGAAGCGCAAGACCCGGGAGACGGTCAGCATCGAGGTCAAGGA
AGTGGTGAAGCCCCTGCTGGTGTCCACCCTCGGTGAGAAGAGCGGGAAAAGGACTGAAGACCTGTAAGA
GCCCTGGGCGGAAAAGCAAGGAGAGCAGCCCCAAGGGGCGCAGCAGCAGCGCCTCCTCACCCCCAA

SAMPLE OF CMDS DATASET

1256	2521	2211	1125	0.67	1.04	0.84	1.27	0.78	1.14	1.11	0.68	1.21
1.06	1	1.07	0.79	0.51	0.92	1.28	1.17	0.75	-4.75	7.1	-3.72	4.16
5.27	-13.4	5.8	1.5	0.19	2.54	-5.1	-8.32	-2.15	6.91	2.77	13	130
2	15	13	0.149	1.003								
1258	2521	2211	1125	0.67	1.06	0.92	1.26	0.8	1.16	1.11	0.69	1.27
1.06	1.01	1.07	0.79	0.51	0.92	1.28	1.17	0.75	-4.75	7.1	-3.72	4.16
5.27	-13.4	5.8	1.5	0.19	2.54	-5.1	-8.32	-2.15	6.91	2.77	13	130
2	15	12	0.149	1.004								
1079	1030	988	889	0.76	2.04	0.84	2.27	0.78	2.15	2.5	0.72	2.20
1.04	1.02	1.06	0.77	0.51	0.92	1.28	1.17	0.75	-1.71	7.4	-3.72	4.16
5.25	-13.4	5.8	1.4	0.2	2.54	-5.3	-8.32	-2.15	6.91	2.75	7	374
1	15	13	0.15	1.002								
393	437	439	192	0.55	1.15	0.6	0.84	1.39	0.7	1.01	1.3	1.13
0.72	2.20	1.04	1.02	1.06	0.77	0.51	0.92	1.28	1.17	0.75	-4.75	7.4
-3.72	4.16	5.25	-13.4	5.8	1.4	0.2	2.54	-8.32	-2.15	6.91	2	693
1	15	13	0.15	1.002								
2211	1866	1940	1729	0.33	1.03	0.8	0.4	0.9	0.47	0.1	1.3	1.13
0.2	2.20	2.04	1.02	1.06	0.78	0.51	0.2	1.23	1.18	0.65	-2.75	7.4
1.72	4.16	5.25	-13.4	5.8	1.4	0.2	2.54	-5.3	-8.32	-2.15	6.91	
14299	1	11	17	0.12	1.02							
2211	1867	1940	1729	0.36	1.07	0.9	0.5	0.6	0.67	0.1	1.7	1.13
0.2	2.20	2.04	1.02	1.06	0.78	0.51	0.8	1.23	1.18	0.65	-2.75	7.4
1.72	4.16	5.25	-13.4	5.8	1.4	0.2	2.54	-8.32	-2.15	6.91	2.7	14299
1	11	17	0.12	1.02								
686	557	467	438	0.49	1.04	0.98	0.2	1.39	1.27	0.41	0.3	1.6
0.72	2.24	1.04	1.02	1.06	0.77	0.51	0.92	1.28	1.17	0.75	-4.75	7.4
-3.72	4.16	5.25	-13.4	5.8	1.4	0.2	2.54	-5.3	-8.32	-2.15	2.7	2621
1	1	1	0.15	1.002								

SAMPLE OF MDS DATASET

898	898	1	7113	0	-1	-6	-0.4	-3	-4	-2.2	2	5
7.23	-2.76											
3032	3032	1	7113	0	-2	-6	-2.3	-5	-2	-3.3	8	20
4.21	-2.9											
4328	4328	1	7113	0	0	-1	0.6	-1	-1	-0.6	13	17
5.23	-2.7											
203	203	1	7113	0	0	-5	-0.2	-3	-3	-2.3	11	15
7.22	-2.7											
1214	1214	1	1461	0	0	-5	-0.2	-3	-3	-2.3	15	11
1.97	1.67											
410	410	1	1461	0	-2	-4	0.5	-2	-1	-0.8	7	8
1.98	4.78											
1339	1340	2	7114	1	-1	0	0.8	0	1	0.5	1	8
7.56	1.13											
925	926	2	7111	2	-1	-4	-0.1	-3	-4	-1	2	8
9.08	-4.1											
2971	2986	16	7130	3	0	-3	-1.3	-2	-1	-0.9	15	2
7.76	-5.03											
3728	3728	1	7114	3	4	7	2.5	5	3	2.4	1	1
7.88	1.51											

SAMPLE OF PMDS DATASET

393	437	439	192	0.55	1.15	0.6	0.84	1.39	0.7	1.01	1.3	1.13	
0.72	2.20	1.04	1.02	1.06	0.77	0.51	0.92	1.28	1.17	0.75	-4.75	7.4	
-3.72	4.16	5.25	-13.4	5.8	1.4	0.2	2.54	-8.32	-2.15	6.91	2693	1	
5	13	0.15	1.002	925	926	2	4310	2	-1	-4	-0.1	-3	
-4	-1	2	8	9.08	-4.1								
1079	1030	988	889	0.76	2.04	0.84	2.27	0.78	2.15	2.5	0.72	2.20	1.04
1.02	1.06	0.77	0.51	0.92	1.28	1.17	0.75	-4.75	7.4	-3.72	4.16	5.25	
-13.4	5.8	1.4	0.2	2.54	-5.3	-8.32	-2.15	6.91	2.75	7374	1	15	
13	0.15	1.002	410	410	1	2861	0	-2	-4	0.5	-2	-1	
-0.8	7	8	1.98	4.78									
1256	2521	2211	1125	0.67	1.04	0.84	1.27	0.78	1.14	1.11	0.68	1.21	
1.06	1	1.07	0.79	0.51	0.92	1.28	1.17	0.75	-4.75	7.1	-3.72		
4.16	5.27	-13.4	5.8	1.5	0.19	2.54	-5.1	-8.32	-2.15	6.91	2.77		
13130	2	15	13	0.149	1.003	898	898	1	2861	0	-1	-6	
-0.4	-3	-4	-2.2	2	5	7.23	-2.76						
2211	1866	1940	1729	0.33	1.03	0.8	0.4	0.9	0.47	0.1	1.3	1.13	
0.2	2.20	2.04	1.02	1.06	0.78	0.51	0.2	1.23	1.18	0.65	-2.75	7.4	
1.72	4.16	5.25	-13.4	5.8	1.4	0.2	2.54	-5.3	-8.32	-2.15	6.91		
14299	1	11	17	0.12	1.02	1339	1340	2	7114	1	-1	0	
0.8	0	1	0.5	1									
1258	2520	2210	1125	0.67	1.06	0.92	1.26	0.8	1.16	1.11	0.69	1.27	
1.06	1.01	1.07	0.79	0.51	0.92	1.28	1.17	0.75	-4.75	7.1	-3.72	4.16	
5.27	-13.4	5.8	1.5	0.19	2.54	-5.1	-8.32	-2.15	6.91	2.77	13130		
2	15	12	0.149	1.004	3032	3032	1	2861	0	-2	-6	-2.3	-
5-2	-3.3	8	20	7.21	-2.9								
2210	1867	1940	1729	0.36	1.03	0.9	0.7	1.4	0.55	0.1	1.3	2.13	
0.2	2.20	2.04	1.02	1.06	0.78	0.51	0.2	1.23	1.18	0.65	-2.75	7.4	
1.72	4.16	5.25	-13.4	5.8	1.4	0.2	2.54	-5.3	-8.32	-2.15	6.91		
14299	1	11	17	0.12	1.02	1339	1340	2	7114	1	-1	0	
0.8	0	1	0.5	1									

SAMPLE OF GSDS DATASET

122	2	1731	1	8	20	1	2	3	-1	-6	-0.4
-3	-4	-2.2	5.66E-08		2.94E-04		4	1	1	1	1
0	1	1	1883.4								
85	5	1130	2	5	8	1	3	2	-1	-4	1.1
-3	-4	-2.2	5.66E-08		0	3	1	0	1	1	0
1	0	1234.9									
43	1	770	2	6	10	1	2	2	-1	-3	-0.6
-3	-4	-2.2	0	5.71E-04		3	0	1	0	1	0
1	1	830.20									
66	2	131	1	5	7	1	1	1	-1	-1	-0.13
-3	-4	-2.2	0	5.71E-04		4	1	1	1	1	0
1	1	213.6									
110	1	1731	1	6	8	1	1	3	-1	-6	-0.2
-3	-4	-2.2	5.66E-08		2.94E-04		4	1	1	1	1
0	1	0	1854.6								
122	5	1731	1	8	10	1	2	2	-1	-2	0.17
-3	-4	-2.2	5.66E-08		2.94E-04		4	1	1	1	1
0	0	1	1878.97								
21	2	1731	1	5	11	1	2	1	-1	-4	0.5
-3	-4	-2.2	5.66E-08		2.94E-04		4	0	1	1	1
0	0	1	1769.3								

SAMPLE OF CODON ENCODED GENE SEQUENCE

10	46	5	41	6	12	11	54	3	22	18
35	21	44	39	10	15	17	21	14	11	12
4	41	60	24	28	13	19	11	8	35	41
9	26	6	41	6	38	11	54	26	22	18
15	20	4	39	10	15	18	21	8	11	24
4	57	48	25	39	12	20	16	62	35	49
10	46	5	41	6	15	11	54	3	22	18
35	21	44	39	10	15	17	21	14	11	2
4	41	36	24	28	12	19	11	8	35	41
3	26	16	41	6	24	11	54	26	22	18
15	20	23	39	10	15	18	21	8	11	24
4	57	60	25	39	12	20	16	62	35	4
12	26	32	8	6	17	11	54	26	22	18
15	20	23	39	10	15	19	21	8	11	24
4	57	60	25	39	12	20	16	61	35	60

APPENDIX B – CODING

R Code for Inducing Mutations and Extracting Features

```
require(stringi)
require(seqinr)
lines <- readLines('SHANK3.txt')
bdy2 <- paste(lines, collapse=",")
x<-stri_replace_all_charclass(bdy2, "\\p{WHITE_SPACE}", "")
x2<-gsub('[0-9]+', '', x)
x3<- gsub('[[:punct:]]', '', x2)
// Replace character in codon and convert to data frame
Posn= (Codon_number-1)* i +1 // i is position of character in a codon to be replaced
x4<-subcharT(x3, Posn)
x5<-data.frame(x4)
write.fasta(sequences=x5,names="",file.out="E://out1.fasta")
// Function select according to variation
subcharT <- function(string, pos, char="T") {
  for(i in pos) { string <- gsub(paste("^(.{", i-1, "}).", sep=""), "\\1T", string, perl=TRUE) }
  string }
subcharG <- function(string, pos, char="G") {
  for(i in pos) { string <- gsub(paste("^(.{", i-1, "}).", sep=""), "\\1G", string, perl=TRUE) }
  string }
subcharA <- function(string, pos, char="A") {
  for(i in pos) { string <- gsub(paste("^(.{", i-1, "}).", sep=""), "\\1A", string, perl=TRUE) }
  string }
subcharC<- function(string, pos, char="C") {
  for(i in pos) { string <- gsub(paste("^(.{", i-1, "}).", sep=""), "\\1C", string, perl=TRUE) }
  string }
// Extract Features From Modified Fasta
require(seqinr)
sh <- read.fasta(file = "E://out1.fasta")
shseq<-sh[[1]]
extract_feature(shseq)
// Function to extract features
extract_feature<-function(sequence)
{
  T<-table(shseq)
  G<-GC(shseq)
  R<-round(rho(shseq, wordsize=2),2)
```

```

Z<-round(zscore(shseq, modele='base'),2)
Y<- count(shseq, wordsize=3)
p<-c2s(shseq)
globalAligns1s2 <- pairwiseAlignment(x3, x4, substitutionMatrix = sigma, gapOpening = -2,
gapExtension = -8, scoreOnly = FALSE)
print(T)
print(G)
print(R)
print(Z)
print(Y)
print(globalAligns1s2)
}
// To normalize
sample=read.csv("E:\\out2.csv")
// formula is normalized = (x-min(x))/(max(x)-min(x))
normalized = (sample-min(sample))/(max(sample)-min(sample))
write.csv(normalized, "E:\\out3.csv")

```

Code to Predict ASD Gene Type

```

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import accuracy_score
df=pd.read_csv('E:\cmds.csv')
print(df)
x=df.values[:,0:43]
y=df.values[:,43]
x_train,x_test,y_train,y_test=train_test_split(x,y)
scaler=StandardScaler()
scaler.fit(x_train)
StandardScaler(copy=True,with_mean=True,with_std=True)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
mlp=MLPClassifier(hidden_layer_sizes=(8,8,8),max_iter=500)
mlp.fit(x_train,y_train)
# BUILD MODEL
MLPClassifier()

```

```

predictions=mlp.predict(x_test)
print(confusion_matrix(y_test,predictions))
print(classification_report(y_test,predictions))
# ACCURACY_SCORE
print ("Accuracy is",accuracy_score(y_test,predictions)*100)

```

Code to Predict ASD Causing Mutations

```

from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
from sklearn.cross_validation import train_test_split
from sklearn.metrics import accuracy_score
df=pd.read_csv('E:\mds.csv')
print(df)
x=df.values[:,0:15]
y=df.values[:,15]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=3,random_state=100)
model=DecisionTreeClassifier()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred
# ACCURACY
print "Accuracy using DecisionTreeClassifier is",accuracy_score(y_test,y_pred)*100
print(metrics.classification_report(expected,predicted))
print(metrics.classification_report(y_test,y_pred))

```

Code for Gene Susceptibility Prediction

```

import numpy as np
import matplotlib.pyplot as plt
from itertools import cycle
import pandas as pd
from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from scipy import interp
# Import some data to play with
dd=pd.read_csv("E:/genesuscept.csv")
X=dd.values[:,0:25]

```

```

y=dd.values[:,25]
# Binarize the output
y = label_binarize(y, classes=[0, 1, 2])
n_classes = y.shape[1]
# shuffle and split training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.5,
random_state = 0)
# Learn to predict each class against the other
classifier = OneVsRestClassifier(svm.SVC(kernel='linear', probability=True,
random_state=random_state))
y_score = classifier.fit(X_train, y_train).decision_function(X_test)
Y_pred =model.predict(X_test)
Y_pred
#ACCURACY SCORE
print "Accuracy is",accuracy_score(Y_test,Y_pred)*100
#CONFUSION MATRIX
print "confusion matrix",confusion_matrix(y_test,Y_pred)
#CLASSIFICATION REPORT
print "report:",classification_report(y_test,Y_pred)
# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))
lw=2
# Then interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])
# Finally average it and compute AUC
mean_tpr /= n_classes
fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

```

```

# Plot all ROC curves
plt.figure()
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
         ".format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)
plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
         ".format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)
colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=lw,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ".format(i, roc_auc[i]))
plt.plot([0, 1], [0, 1], 'k--', lw=lw)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc="lower right")
plt.show()

```

Codon Encoding

```

rep_val={"AAA":1, "ATA":2, "AGA":3, "ACA":4, "AAT":5, "ATT":6, "AGT":7, "ACT":8,
"AAG":9, "ATG":10, "AGG":11, "ACG":12, "AAC":13, "ATC":14, "AGC":15, "ACC":16,
"TAA":17, "TTA":18, "TGA":19, "TCA":20, "TAT":21, "TTT":22, "TGT":23, "TCT":24,
"TAG":25, "TTG":26, "TGG":27, "TCG":28, "TAC":29, "TTC":30, "TGC":31, "TCC":32,
"GAA":33, "GTA":34, "GGA":35, "GCA":36, "GAT":37, "GTT":38, "GGT":39, "GCT":40,
"GAG":41, "GTG":42, "GGG":43, "GCG":44, "GAC":45, "GTC":46, "GGC":47, "GCC":48,
"CAA":49, "CTA":50, "CGA":51, "CCA":52, "CAT":53, "CTT":54, "CGT":55, "CCT":56,
"CAG":57, "CTG":58, "CGG":59, "CCG":60, "CAC":61, "CTC":62, "CGC":63, "CCC":64}
dataframe = pandas.read_csv("D:\\New folder\\New folder\\sample\\data1.csv", header=None)
dataframe.replace(rep_val, inplace=True)
dataframe.head()
dataframe.to_csv("D:\\out3.csv")

```

Code for Bidirectional RNN Based Gene Prediction

```
import keras
```

```
import tensorflow as tf
from keras.models import Sequential
from keras.layers.wrappers import Bidirectional
from keras.layers import Dense
from keras.layers import SimpleRNN
from keras.layers import Dropout
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from keras.wrappers.scikit_learn import KerasClassifier
import numpy as np
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils
import pandas
data_dim = 1
timesteps = 2582
num_classes = 10
np.random.seed=7
opt = tf.train.AdamOptimizer(learning_rate=0.01)
dataframe = pandas.read_csv("D:\\geneid.csv", header=None)
dataset = dataframe.values
X = dataset[:,0:2582].astype(float)
Y = dataset[:,2582]
encoder = LabelEncoder()
```

```

encoder.fit(Y)

encoded_Y = encoder.transform(Y)

# convert integers to dummy variables (i.e. one hot encoded)

dummy_y = np_utils.to_categorical(encoded_Y)

model = Sequential()

model.add(Masking(mask_value=0., input_shape=(timesteps,data_dim)))

model.add(Bidirectional(SimpleRNN(8, return_sequences=False,input_shape=(timesteps,
data_dim))))

model.add(Dropout(0.3))

model.add(Dense(10, activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, dummy_y, test_size = 0.2, random_state = 0)

from sklearn.preprocessing import StandardScaler

X_train = X_train.reshape((len(X_train), timesteps, data_dim))

X_test = X_test.reshape((len(X_test), timesteps, data_dim))

model.fit(X_train, y_train, epochs=50, batch_size=200)

loss, accuracy = model.evaluate(X_test,y_test, verbose=0)

print('Accuracy: %f % (accuracy))

print('Loss: %f % (loss))

y_pred = model.predict_classes(X_test)

#print(y_pred)

ycat_pred = np_utils.to_categorical(y_pred, num_classes=10)

#print(ycat_pred)

from sklearn.metrics import precision_score

```

```

print(precision_score(y_test, ycat_pred, average='micro'))

from sklearn.metrics import recall_score

print(recall_score(y_test, ycat_pred, average='micro'))

from sklearn.metrics import classification_report

target_names = ['class 0', 'class 1', 'class 2','class 3', 'class 4', 'class 5','class 6', 'class 7', 'class
8','class 9']

print(classification_report(y_test, ycat_pred, target_names=target_names))

```

Code for LSTM Based Gene Prediction

```

import keras
from keras.models import Sequential
from keras.layers import LSTM, Dense
import numpy as np
data_dim = 1
timesteps = 2582
num_classes = 10
model = Sequential()
model.add(LSTM(8, return_sequences=True,
              input_shape=(timesteps, data_dim))) # returns a sequence of vectors of dimension 32
model.add(LSTM(8)) # return a single vector of dimension 32
model.add(Dense(11, activation='softmax'))
import pandas
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils
model.compile(loss='categorical_crossentropy', optimizer='adam',
             metrics=['accuracy'])
dataframe = pandas.read_csv("D:\\RESULTS\\out2.csv", header=None)
dataset = dataframe.values
X = dataset[:,0:2582].astype(float)
Y = dataset[:,2582]
X = X.reshape((len(X), timesteps, data_dim))
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
# convert integers to dummy variables (i.e. one hot encoded)
dummy_y = np_utils.to_categorical(encoded_Y)

```



```
y_new = dummy_y.reshape((len(X), num_classes))
estimator= model.fit(X, y_new, epochs=1, batch_size=2)
```

Code for GRU Based Gene Prediction

```
from keras.utils import plot_model
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
from keras.layers.recurrent import LSTM
from keras.layers.merge import concatenate
# define input
visible = Input(shape=(2582,1))
# feature extraction
extract1 = GRU(8)(visible)
# first interpretation model
interp1 = Dense(8, activation='relu')(extract1)
# second interpretation model
interp11 = Dense(8, activation='relu')(extract1)
interp12 = Dense(8, activation='relu')(interp11)
interp13 = Dense(8, activation='relu')(interp12)
# merge interpretation
merge = concatenate([interp1, interp13])
# output
output = Dense(10, activation='sigmoid')(merge)
model = Model(inputs=visible, outputs=output)
# summarize layers
import keras
import tensorflow as tf
from keras.models import Sequential
from keras.layers import GRU, Dense
from keras.layers import SimpleRNN
from keras.layers import Masking
from keras.layers import Dropout
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from keras.wrappers.scikit_learn import KerasClassifier
import numpy as np
from sklearn.preprocessing import LabelEncoder
```

```

from keras.utils import np_utils
import pandas
data_dim = 1
timesteps = 2582
num_classes = 10
np.random.seed=7
opt = tf.train.AdamOptimizer(learning_rate=0.01)
dataframe = pandas.read_csv("D:\\new1.csv", header=None)
dataset = dataframe.values
X = dataset[:,0:2582].astype(float)
Y = dataset[:,2582]
encoder = LabelEncoder()
encoder.fit(Y)
encoded_Y = encoder.transform(Y)
# convert integers to dummy variables (i.e. one hot encoded)
dummy_y = np_utils.to_categorical(encoded_Y)
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, dummy_y, test_size = 0.2, random_state =
0)
from sklearn.preprocessing import StandardScaler
X_train = X_train.reshape((len(X_train), timesteps, data_dim))
X_test = X_test.reshape((len(X_test), timesteps, data_dim))
model.fit(X_train, y_train, epochs=60, batch_size=64)

```

APPENDIX C – SCREENSHOTS

CDNA Gene Sequence for MECP2 gene

methy1-CpG binding protein 2
 Gene Symbol : [MECP2](#)
 Location : Xq28
 Based on accession number : [NM_004992.3](#)

Switch view

.....HGMD CARDIFF.....

```

1  ATG GTA GCT GGG ATG TTA GGG CTC AGG GAA GAA AAG TCA GAA GAC 15
16 CAG GAC CTC CAG GGC CTC AAG GAC AAA CCC CTC AAG TTT AAA AAG 30
31 GTG AAG AAA GAT AAG AAA GAA GAG AAA GAG GGC AAG CAT GAG CCC 45
46 GTG CAG CCA TCA GCC CAC CAC TCT GCT GAG CCC GCA GAG GCA GGC 60
61 AAA GCA GAG ACA TCA GAA GGG TCA GGC TCC GCC CCG GCT GTG CCG 75
76 GAA GCT TCT GCC TCC CCC AAA GAG GGG GGC TCC ATC ATC GGT GAC 90
91 CGG GGA CCC ATG TAT GAT GAC CCC ACC CCG CCG GAA GGC TGG ACA 105
106 CGG AAG CTT AAG CAA AGG AAA TCT GGC CCG TCT GCT GGG AAG TAT 120
121 GAT GTG TAT TTG ATC AAT CCC CAG GGA AAA GCC TTT CGC TCT AAA 135
136 GTG GAG TTG ATT GCG TAC TTC GAA AAG GTA GGC GAC ACA TCC CTG 150
151 GAC CCT AAT GAT TTT GAC TTC ACG GTA ACT GGG AGA GGG AGC CCC 165
166 TCC CGG CGA GAG CAG AAA CCA CCT AAG AAG CCC AAA TCT CCC AAA 180
181 GCT CCA GGA ACT GGC AGA GGC CGG GGA CCG CCC AAA GGG AGC GGC 195
196 ACC ACG AGA CCC AAG GCG GCC ACG TCA GAG GGT GTG CAG GTG AAA 210
211 AGG GTC CTG GAG AAA AGT CCT GGG AAG CTC CTT GTC AAG ATG CCT 225
226 TTT CAA ACT TCG CCA GGG GGC AAG GCT GAG GGG GGT GGG GCC ACC 240
  
```

Mutational information from SFARI gene database for MECP2 gene

Human Gene	Variant	Protein Change	Description	Origin	Penetrance	Frequency	Reference
MECP2	c.1035A>G	p.L=	synonymous_variant	-	-	11309367	Couvert P., [2001]
MECP2	c.1071A>G	p.L=	synonymous_variant	Unknown	Unknown	23055267	Cukier HN., [2012]
MECP2	c.1108G>A	p.Ala370Thr	missense_variant	De novo	-	30564305	Guo H., et al., [2012]
MECP2	c.1108G>A	p.Ala370Thr	missense_variant	Familial	Maternal	23055267	Cukier HN., [2012]
MECP2	c.1127C>G	p.Pro376Arg	missense_variant	-	-	15211631	Shibayama [2004]
MECP2	c.1138_1144del	p.Val380CysfsTer27	frameshift_variant	Familial	Maternal	30842647	Boonsawat [2019]
MECP2	c.1145C>T	p.L=	synonymous_variant	-	-	11007980	Orrico A., et al., [2000]

BLAST Alignment Results

Query ID: lcl|Query_116243
 Description: None
 Molecule type: dna
 Query Length: 3996
 Other reports: [Distance tree of results](#) [MSA viewer](#)

Percent Identity: [] to []
 E value: [] to []
 Filter Reset

Descriptions | Graphic Summary | Alignments | Taxonomy

Sequences producing significant alignments Download Manage Columns Show 100

select all 100 sequences selected GenBank Graphics Distance tree of results

Description	Max Score	Total Score	Query Cover	E value	Per. Ident	Accession
<input checked="" type="checkbox"/> Homo sapiens contactin associated protein like 2 (CNTNAP2), mRNA	7374	7374	100%	0.0	99.97%	NM_014141.6
<input checked="" type="checkbox"/> Homo sapiens contactin associated protein-like 2, mRNA (cDNA clone MGC:141933 IMAGE:8322425), complete cds	7369	7369	100%	0.0	99.95%	BC113373.1
<input checked="" type="checkbox"/> Homo sapiens contactin-associated protein 2 (CNTNAP2), mRNA, complete cds	7369	7369	100%	0.0	99.95%	AF319045.1
<input checked="" type="checkbox"/> Homo sapiens cell recognition molecule Caspr2 (CASPR2), mRNA, complete cds	7369	7369	100%	0.0	99.95%	AF193613.1
<input checked="" type="checkbox"/> Homo sapiens cDNA, FLJ96073, highly similar to Homo sapiens contactin associated protein-like 2 (CNTNAP2), mRNA	7363	7363	100%	0.0	99.92%	AK315113.1
<input checked="" type="checkbox"/> Homo sapiens contactin associated protein-like 2, mRNA (cDNA clone MGC:120815 IMAGE:7939625), complete cds	7363	7363	100%	0.0	99.92%	BC093780.1
<input checked="" type="checkbox"/> Homo sapiens KIAA0868, mRNA for KIAA0868 protein	7363	7363	100%	0.0	99.92%	AB020675.1
<input checked="" type="checkbox"/> Synthetic construct DNA, clone: pFKA0068, Homo sapiens CNTNAP2 gene for contactin associated protein-like 2, without stop codon, in Fl	7352	7352	99%	0.0	99.90%	AB462990.1
<input checked="" type="checkbox"/> PREDICTED: Pan paniscus contactin associated protein like 2 (CNTNAP2), mRNA	7197	7197	100%	0.0	99.17%	XM
<input checked="" type="checkbox"/> PREDICTED: Pan troglodytes contactin associated protein like 2 (CNTNAP2), transcript variant X2, mRNA	7191	7191	100%	0.0	99.15%	XM

Taskbar: Microsoft E..., appendix, NCBI Blast..., sample dat..., screenshot..., Sample cod..., Jupyter Not..., 6:39 AM 8/21/2019

MEKA Environment

MEKA GUI Chooser
 Program Tools Visualization Help
MEKA
 A Multi-label Extension to WEKA
 1.9.0

MEKA Explorer
 File Edit
 Preprocess Classify Visualize Log
 Classifier: Choose NSR -P 0 -N 0 -S 0 -W weka.classifiers.trees.J48 --C 0.25 -M 2
 Evaluation: Train/test split Start Stop History

meka.gui.goe.GenericObjectEditor
 Choose meka.classifiers.multitarget.NSR
 About: The Nearest Set Replacement (NSR) method. More Capabilities
 classifier: Choose J48 -C 0.25 -M 2
 debug: False
 doNotCheckCapabilities: False
 n: 0
 p: 0
 seed: 0
 Open... Save... OK Cancel

Taskbar: run, Windows Batch File, Date modified: 10/30/2015 8:49 AM, Date created: 2/2/2017 2:58 PM, 5:40 AM 8/21/2019

Input for Gene – Mutation Prediction in MEKA

The screenshot shows the MEKA Explorer interface. The main window displays a data table with columns labeled I, J, K, L, Zcc, Zcg, Zct, Zga, and Zc. The data rows contain numerical values. The MEKA Explorer window is open, showing the 'Filter' tab. The 'Current data set' is 'Id_genes_consol_res2' with 45 attributes and 1000 instances. The 'Attributes' list includes GC, GG, TA, TC, TT, Zaa, Zbc, Zcg, Zct, Zga, and Zgc. The 'Classes' list includes ALIGN, exons, donor, acceptor, percentCpG, CpGbyexp, avg length exons, gene_class, and mutation_class. The 'Selected attribute' section shows 'Name: None', 'Missing: None', 'Distinct: None', and 'Unique: None'.

Gene Susceptibility Prediction using Sci-kit learn

The screenshot shows a Jupyter Notebook interface with the following Python code:

```
In [ ]: import numpy as np
import matplotlib.pyplot as plt
from itertools import cycle
import pandas as pd
from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from scipy import interp
# Import some data to play with
dd=pd.read_csv("E:/genesuscept.csv")
X=dd.values[:,0:25]
y=dd.values[:,25]
# Binarize the output
y = label_binarize(y, classes=[0, 1, 2])
n_classes = y.shape[1]
# shuffle and split training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state = 0)
# Learn to predict each class against the other
classifier = OneVsRestClassifier(svm.SVC(kernel='linear', probability=True, random_state=random_state))
y_score = classifier.fit(X_train, y_train).decision_function(X_test)
Y_pred =model.predict(X_test)
Y_pred
#ACCURACY SCORE
print "Accuracy is",accuracy_score(Y_test,Y_pred)*100
#CONFUSION MATRIX
print "confusion matrix",confusion_matrix(y_test,Y_pred)
#CLASSIFICATION REPORT
print "report:",classification_report(y_test,Y_pred)
```

Output from Sci-kit learn Environment

