**Multi Omic model Creation**

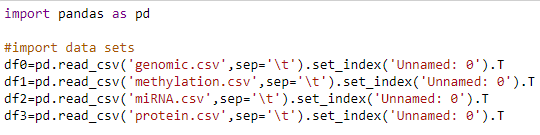
**Dataset Preparation**

After finalizing the model for single omics procedure for all omics types (4), we able to find the most appropriate features for each omics considering the model performance. Then we reduced the data tuples in initial data sets by only taking the tuples related to selected features.

**Import data**

**01.**

Here we imported those datasets into python 3 environment.



Reason for using set\_index is, if we did not define the index, by default Pandas will create an index for the DataFrame. It makes things a little more confusing, because by default the “index” is just the range of numbers starting at 0.

Here we swap the rows and columns using *transpose()* method to get the features and label into columns and patients into rows.

**02.**

Here are the shapes of each dataset.



**Data Preprocessing**

**Data Integration**

**01.**

Now we need to combine these 4 datasets. Since the patients are not exactly same in 4 datasets, we could not use merge method. So we used *pandas.concat* method for combine the datasets.



We transposed the dataset to grab the columns named as “subtype”.

**02.**

Here is the shape of new data frame



Here are 832 columns (data tuples). It’s just the combination of df0, df1, df2 and df3 (291+61+309+171=832).

Here they show 1183 rows(features). Since in 4 datasets there were more similar columns and some different columns for each other, number of columns are differed from df0, df1, df2 and df3. We actually need to remove non similar items for further process. Since those non similar items have missing values in combined datasets.

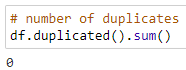
**03.**

Then we divided columns into 2 datasets. In one dataset we kept the features. And in other we kept labels. There were actually 4 labels since we used 4 datasets. We isolated them and created a 1 label column which contain fair combination of those 4 columns.

**Data Cleaning**

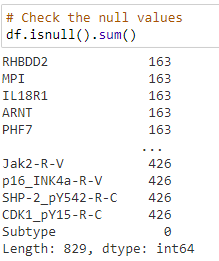
**01.**

First, check for duplicates.



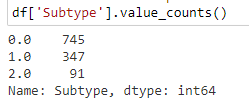
**02.**

Then we check for missing values.

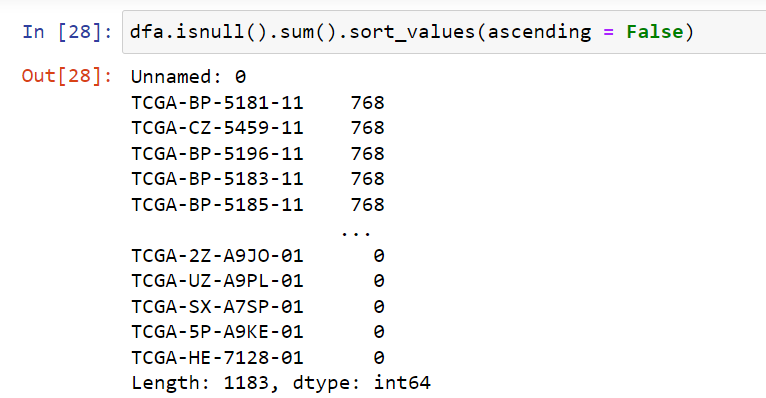


We found more null valued rows with this. And this can affect the model very badly. And we cannot remove this much of features from the dataset. So we try to do the further process in patient’s side.

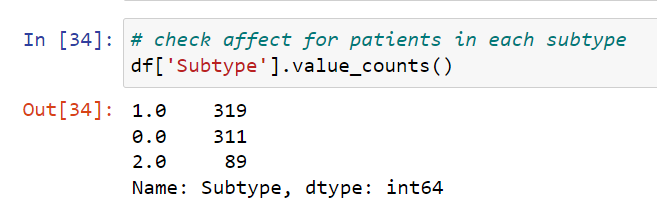
We checked how much patients in each subtype



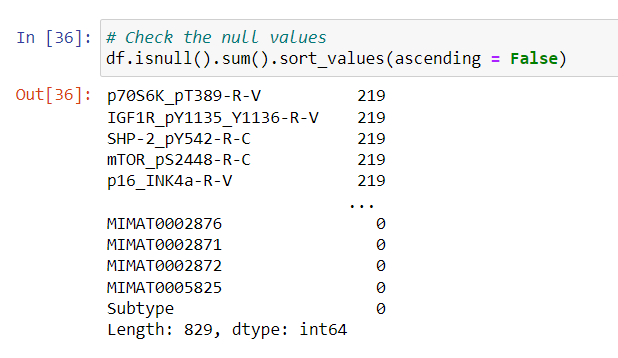
For further process we got the transpose of the dataset and check null values for each patient.



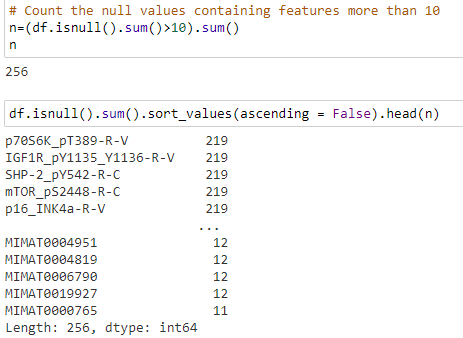
We count the patients who have more than 300 null values. There were 464 patients. We removed those patients and check the number of patients for each omic type.



Then we check the number of null values for each omic type.



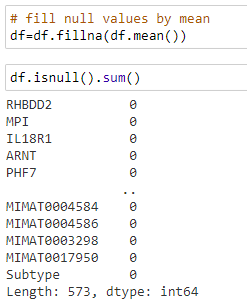
Here we check the number features which have more than 10 null values. We got 256 features now.



We removed those features are here is the size of the main data frame after removing both patients and features.



And we filled remaining null vales by the mean.



**03.**

Then we checked the rows which include zero values.

Count of zeros

IL18R1 : 1

C6orf150 : 1

PNCK : 15

DIRC3 : 13

ARHGAP8 : 1

FAM186B : 1

FOXL1 : 6

CXorf57 : 1

CAMK4 : 11

LMTK3 : 2

NMUR1 : 1

C5orf36 : 1

KLC3 : 1

BIK : 5

C19orf26 : 13

ALPK2 : 8

C22orf45 : 8

GPR85 : 1

FBXO2 : 1

BCO2 : 1

DOK7 : 7

DIRAS2 : 2

RBMS3 : 1

C15orf59 : 3

HAGHL : 1

KCNK3 : 4

SLC2A5 : 2

C19orf73 : 5

TMEM26 : 2

ENPP3 : 9

C1orf53 : 1

SYNGR3 : 1

FAM155B : 3

CYP2J2 : 17

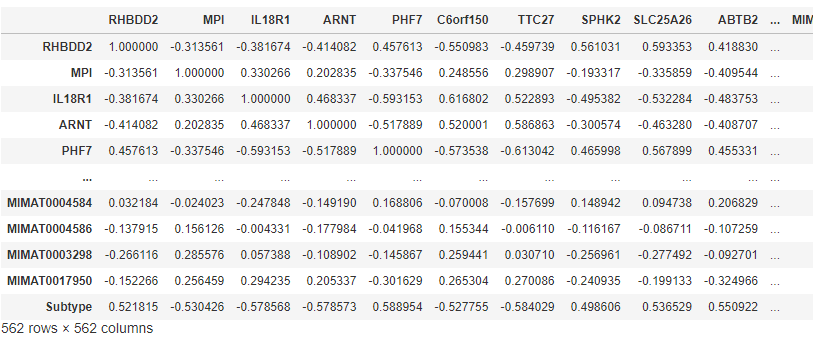
CP : 7

Very few zero valued containing column were there. So, we didn’t handle that few zero values because zero also can be a valid reading.

**Feature Selection - Pearson correlation coefficient**

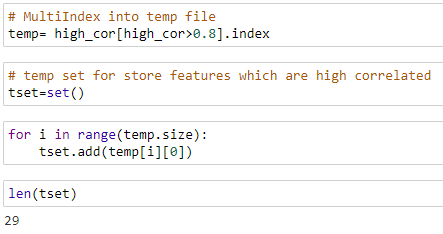
**01.**

Getting the correlation coefficients in dataset



**02.**

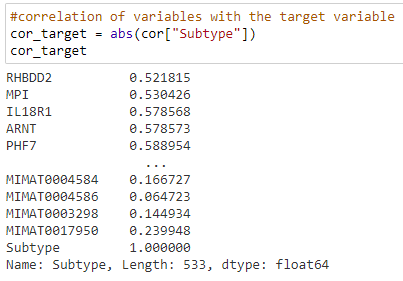
In this step we removed the features which overly corelated. For that we take a threshold value of 0.80 and removed relevant features. Since we assumed those values are high correlated. It means these are somewhat linearly dependent with other features. Hence those effect on the dependent variable is almost same.



Removed feature count: 29

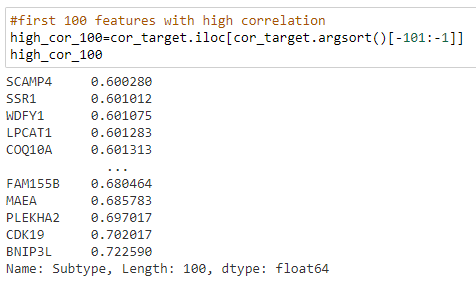
**2.**

Taking the absolute correlation of features with the target variable.

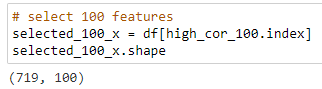


**3.**

Initially we got the best 100 features for further process.

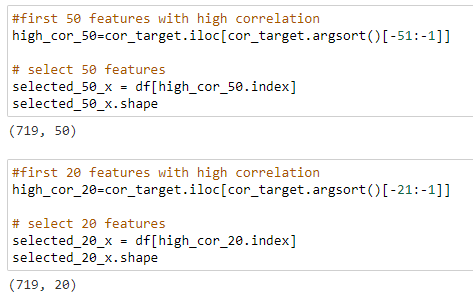


Selected 100 features.



**4.**

Likewise in the step 3, We build models which is getting different feature amounts.

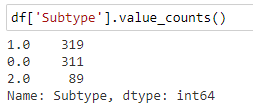


**Dataset balancing**

**01.**

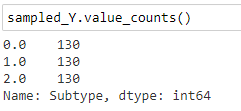
Then we viewed the value count of each subtype.

* Initial feature count from each class (Before sampling)



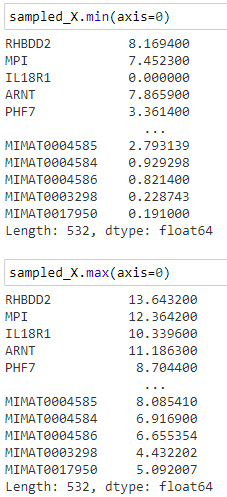
As it shows the data frame is highly imbalanced. So, We decided to handle these imbalancing. For that we used both Oversampling and Undersampling.

(After sampling)



**Feature Scaling**

In here we got an idea about the value range in features.



Here the value range looks small. Even though we hope to try the transformations also in the following processes.

**Test SVM**

**01.**

First, we try to find the best kernel to use for further process. Here we used cross validation for this test.

* Accuracy SVM with rbf kernel: 0.95 (+/- 0.03)
* Accuracy SVM with poly kernel: 0.98 (+/- 0.03)
* Accuracy SVM with linear kernel: 0.98 (+/- 0.03)
* Accuracy SVM with sigmoid kernel: 0.61 (+/- 0.07)

Both poly and linear kernels performed well.

**02.**

Then, we find the accuracy without data transformation. Then we tried both standardization and normalization. So, we observed that without data transformation performed well with SVM. Here also used cross validation for this test.

For “poly” kernel;

* Accuracy SVM without transformation: 0.98 (+/- 0.03)
* Accuracy SVM with Standardization: 0.82 (+/- 0.10)
* Accuracy SVM with Normalization: 0.98 (+/- 0.03)

For poly kernel no transformation needs

For “linear” kernel;

* Accuracy SVM without transformation: 0.98 (+/- 0.03)
* Accuracy SVM with Standardization: 0.98 (+/- 0.03)
* Accuracy SVM with Normalization: 0.98 (+/- 0.03)

For linear kernel no transformation needs

For further process we choose liner kernel.

**03.**

Then we considered other datasets, which we created with different number of features (20,50,100) and evaluate those models using SVM by cross validations.

* Accuracy SVM using best 20 features: 0.89 (+/- 0.05)
* Accuracy SVM using best 50 features: 0.92 (+/- 0.06)
* Accuracy SVM using best 100 features: 0.94 (+/- 0.04)

We tried more than 100 number of features for a better performance.

* Accuracy SVM using best 120 features: 0.95 (+/- 0.05)
* Accuracy SVM using best 140 features: 0.95 (+/- 0.05)
* Accuracy SVM using best 150 features: 0.94 (+/- 0.04)
* Accuracy SVM using best 200 features: 0.96 (+/- 0.03)
* Accuracy SVM using best 250 features: 0.97 (+/- 0.04)
* Accuracy SVM using best 280 features: 0.97 (+/- 0.04)
* Accuracy SVM using best 300 features: 0.97 (+/- 0.04)
* Accuracy SVM using best 350 features: 0.97 (+/- 0.04)
* Accuracy SVM using best 380 features: 0.97 (+/- 0.03)
* **Accuracy SVM using best 400 features: 0.98 (+/- 0.03)**

So, number of features with 400 shows the best performance.

**Test RF**

**01.**

First, we try different combinations of parameters with the help of GridSearchCV. Here we tested best 3 parameters in RF and took the best combination.

* Best parameters are:
  + 'criterion': 'entropy'
  + 'max\_depth': 8,
  + 'n\_estimators': 500

02.

Then we did the cross validation.

* Accuracy RF with best\_300 params: 0.97 (+/- 0.03)

**03.**

Then we test the model for other datasets, which we created with different number of features (20, 50, 100) and evaluate those models’ using RF by cross validations.

* Accuracy RF using best 20 features: 0.91 (+/- 0.04)
* Accuracy RF using best 50 features: 0.95 (+/- 0.04)
* Accuracy RF using best 100 features: 0.95 (+/- 0.03)

Then we selected the feature amount larger than 100

* Accuracy RF using best 120 features: 0.95 (+/- 0.03)
* Accuracy RF using best 140 features: 0.94 (+/- 0.03)
* Accuracy RF using best 150 features: 0.95 (+/- 0.02)
* Accuracy RF using best 200 features: 0.94 (+/- 0.02)
* Accuracy RF using best 250 features: 0.96 (+/- 0.03)
* Accuracy RF using best 300 features: 0.95 (+/- 0.02)
* Accuracy RF using best 350 features: 0.96 (+/- 0.03)
* Accuracy RF using best 400 features: 0.96 (+/- 0.03)
* **Accuracy RF using best 450 features: 0.97 (+/- 0.03)**

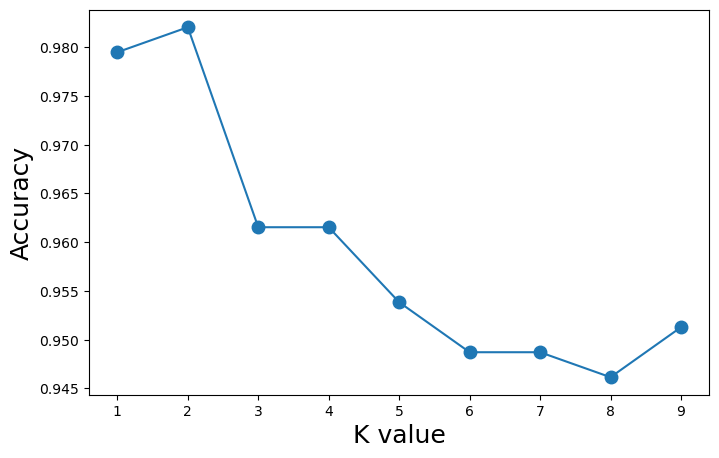
450 features give the accuracy same as with all features.

**Test KNN**

**01.**

First, we scaled the features to use KNN. Here we tested with Standardization as well as Normalization. We tried to find the best k value with both scaling and took the accuracies of them.

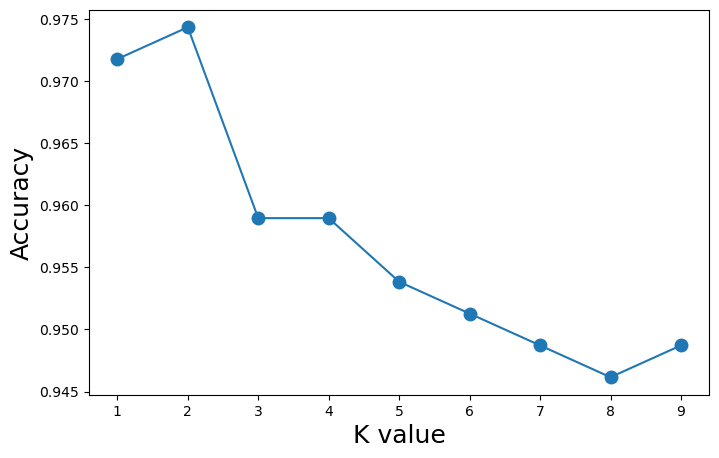
**For Standardization:**



It shows best accuracy when K=2.

Accuracy KNN with Standardization: 0.98 (+/- 0.03)

**For Standardization:**



It shows best accuracy when K=2.

Accuracy KNN with Normalization: 0.97 (+/- 0.02)

Considering above performance, we choose **Standardization** with **K=2** for the further process.

**02.**

Then we tried for other datasets, which we created with different number of features (20,50,100) and evaluate those models using KNN by cross validations.

* Accuracy KNN using best 20 features: 0.91 (+/- 0.04)
* Accuracy KNN using best 50 features: 0.92 (+/- 0.03)
* Accuracy KNN using best 100 features: 0.93 (+/- 0.04)

Then we tried with feature amount larger than 100

* Accuracy KNN using best 120 features: 0.93 (+/- 0.06)
* Accuracy KNN using best 140 features: 0.92 (+/- 0.05)
* Accuracy KNN using best 150 features: 0.93 (+/- 0.05)
* Accuracy KNN using best 200 features: 0.92 (+/- 0.04)
* Accuracy KNN using best 250 features: 0.92 (+/- 0.05)
* Accuracy KNN using best 300 features: 0.94 (+/- 0.05)
* Accuracy KNN using best 350 features: 0.96 (+/- 0.05)
* Accuracy KNN using best 400 features: 0.95 (+/- 0.06)
* Accuracy KNN using best 450 features: 0.96 (+/- 0.05)
* **Accuracy KNN using best 500 features: 0.98 (+/- 0.03)**

Here 500 features give the best performance

**03.**

Finally, we checked the performance of all 3 algorithms. Here below are the obtained results

* **Accuracy SVM using 400 features: 0.98 (+/- 0.03)**
* Accuracy RF using 450 features: 0.97 (+/- 0.03)
* Accuracy KNN using 500 features: 0.98 (+/- 0.03)

**Overall Results – Multi Omic Data**

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of features** | **Cross Validation Score** | | |
| **SVM (Without transformation)** | **RF** | **KNN (Standardized)** |
| 532(All) | 0.98 (+/- 0.03) | 0.97 (+/- 0.03) | 0.98 (+/- 0.03) |
| 500 |  |  | 0.98 (+/- 0.03) |
| 450 |  | 0.97 (+/- 0.03) | 0.96 (+/- 0.05) |
| 400 | 0.98 (+/- 0.03) | 0.96 (+/- 0.03) | 0.95 (+/- 0.06) |
| 380 | 0.97 (+/- 0.03) |  | 0.95 (+/- 0.06) |
| 350 | 0.97 (+/- 0.04) | 0.96 (+/- 0.03) | 0.96 (+/- 0.05) |
| 300 | 0.97 (+/- 0.04) | 0.95 (+/- 0.02) | 0.96 (+/- 0.05) |
| 280 | 0.97 (+/- 0.04) |  | 0.93 (+/- 0.06) |
| 250 | 0.97 (+/- 0.04) | 0.96 (+/- 0.03) | 0.92 (+/- 0.05) |
| 200 | 0.96 (+/- 0.03) | 0.94 (+/- 0.02) | 0.92 (+/- 0.05) |
| 150 | 0.94 (+/- 0.04) | 0.95 (+/- 0.02) | 0.93 (+/- 0.05) |
| 140 | 0.95 (+/- 0.05) | 0.94 (+/- 0.03) | 0.93 (+/- 0.05) |
| 120 | 0.95 (+/- 0.05) | 0.95 (+/- 0.03) | 0.93 (+/- 0.06) |
| 100 | 0.94 (+/- 0.04) | 0.95 (+/- 0.03) | 0.93 (+/- 0.06) |
| 50 | 0.92 (+/- 0.06) | 0.95 (+/- 0.04) | 0.92 (+/- 0.03) |
| 20 | 0.89 (+/- 0.05) | 0.91 (+/- 0.04) | 0.91 (+/- 0.04) |

Highest accuracy we obtained is around 98% and SVM abled to provide that. And the number of features we used is 400. For this we didn’t used any transformation method.

* Here below are the parameters we used for SVM
* kernel=' linear’
* Selected features (400):
  1. MIMAT0004688
  2. MIMAT0000448
  3. MIMAT0000426
  4. MIMAT0002872
  5. MIMAT0004927
  6. MIMAT0004614
  7. MIMAT0001080
  8. MIMAT0004680
  9. MIMAT0005825
  10. MIMAT0000265
  11. MIMAT0000079
  12. MIMAT0003283
  13. MIMAT0000274
  14. MIMAT0004588
  15. MIMAT0016847
  16. MIMAT0004485
  17. MIMAT0000451
  18. MIMAT0000254
  19. MIMAT0004956
  20. MIMAT0000244
  21. MIMAT0000073
  22. MIMAT0004501
  23. MIMAT0004673
  24. MIMAT0000091
  25. MIMAT0000764
  26. MIMAT0000270
  27. MIMAT0026738
  28. MIMAT0019746
  29. MIMAT0001620
  30. MIMAT0000226
  31. MIMAT0004553
  32. MIMAT0019761
  33. MIMAT0000415
  34. MIMAT0002809
  35. MIMAT0000458
  36. MIMAT0026477
  37. MIMAT0004571
  38. MIMAT0000455
  39. MIMAT0004482
  40. MIMAT0018073
  41. MIMAT0019208
  42. MIMAT0004550
  43. MIMAT0017985
  44. MIMAT0000424
  45. MIMAT0003328
  46. MIMAT0002876
  47. MIMAT0000703
  48. MIMAT0000077
  49. MIMAT0001343
  50. MIMAT0000099
  51. MIMAT0000071
  52. MIMAT0004585
  53. MIMAT0004911
  54. MIMAT0000416
  55. MIMAT0000434
  56. MIMAT0000689
  57. MIMAT0003241
  58. MIMAT0005797
  59. MIMAT0004693
  60. MIMAT0000069
  61. MIMAT0000460
  62. MIMAT0000432
  63. MIMAT0000435
  64. MIMAT0000750
  65. MIMAT0000065
  66. MIMAT0000710
  67. MIMAT0015045
  68. MIMAT0000688
  69. MIMAT0017990
  70. MIMAT0010214
  71. MIMAT0000459
  72. MIMAT0000759
  73. MIMAT0025848
  74. MIMAT0000756
  75. MIMAT0004697
  76. MIMAT0004502
  77. MIMAT0004799
  78. MIMAT0003150
  79. MIMAT0005792
  80. MIMAT0016895
  81. MIMAT0022696
  82. MIMAT0004569
  83. MIMAT0000253
  84. MIMAT0000096
  85. MIMAT0004589
  86. MIMAT0003239
  87. MIMAT0000086
  88. MIMAT0000275
  89. MIMAT0000272
  90. MIMAT0021020
  91. MIMAT0004611
  92. MIMAT0000425
  93. MIMAT0002888
  94. MIMAT0004509
  95. MIMAT0003885
  96. MIMAT0000763
  97. MIMAT0000444
  98. MIMAT0000281
  99. MIMAT0004551
  100. RBM22
  101. ALDOC
  102. MIMAT0002874
  103. MIMAT0022692
  104. MIMAT0000431
  105. SPHK2
  106. MIMAT0000267
  107. MIMAT0002820
  108. MIMAT0004497
  109. SMPD1
  110. SLC2A1
  111. ARHGEF18
  112. KCNJ8
  113. MIMAT0000227
  114. HPS3
  115. MIMAT0005796
  116. TMEM22
  117. TPD52L1
  118. LOC100128822
  119. MIMAT0000646
  120. CCND1
  121. AP3S1
  122. RHBDD2
  123. RAB11FIP4
  124. MIMAT0000261
  125. BCL7A
  126. DDIT4
  127. ARHGAP39
  128. C6orf150
  129. ACAD10
  130. SLC15A4
  131. MPI
  132. DUSP1
  133. IL10RB
  134. MIMAT0004515
  135. MIMAT0004500
  136. GPR85
  137. KTELC1
  138. UBD
  139. SLC25A26
  140. PPM1H
  141. SCIN
  142. 7-Sep
  143. MYO1B
  144. LMTK3
  145. CUX1
  146. BCAT2
  147. MIMAT0005924
  148. PDGFD
  149. PGAM5
  150. KLC3
  151. TLCD1
  152. C3orf64
  153. CAPN1
  154. CMPK2
  155. IL1RAP
  156. NRL
  157. EHD2
  158. OGDHL
  159. C3orf1
  160. RLF
  161. MIMAT0004776
  162. BIRC3
  163. SLC25A12
  164. PARD6A
  165. ABTB2
  166. MN1
  167. BIK
  168. PWWP2B
  169. MIMAT0004784
  170. TMEM53
  171. PXMP4
  172. POLR3C
  173. NMUR1
  174. 7-Mar
  175. FBXO2
  176. DCLRE1B
  177. APLP2
  178. HIP1R
  179. C9orf21
  180. FAM115C
  181. SERPINB9
  182. ZNF629
  183. AKT2
  184. ALS2CL
  185. OSBPL2
  186. ATP6V0E2
  187. ARHGAP8
  188. PLEKHG2
  189. RIPK1
  190. ANO10
  191. CRLF3
  192. HNMT
  193. MIMAT0000510
  194. TIPRL
  195. RAB11B
  196. INSIG2
  197. MIMAT0004748
  198. C10orf41
  199. MGAT3
  200. ATXN1
  201. FAM186B
  202. DNAJC5
  203. CADM4
  204. MIMAT0002174
  205. SLC9A9
  206. SNX33
  207. GDI2
  208. PDP1
  209. PAG1
  210. CHST15
  211. KIAA1522
  212. LDHD
  213. TAF5
  214. GORASP1
  215. DIRAS2
  216. STRBP
  217. DTNB
  218. CHCHD7
  219. DNAJA3
  220. GIT2
  221. JAG1
  222. ProSAPiP1
  223. HTRA1
  224. FYN
  225. KIAA0284
  226. IL18R1
  227. ARNT
  228. PLOD1
  229. BCAR1
  230. HMGB3
  231. AFF3
  232. SLC29A2
  233. DCUN1D3
  234. TRDMT1
  235. CRTC1
  236. NEDD4
  237. ABLIM3
  238. LOX
  239. C3orf18
  240. CD302
  241. KCTD1
  242. ACTR3B
  243. GTPBP5
  244. SGCB
  245. DDX28
  246. PDDC1
  247. TTC27
  248. TM4SF18
  249. LOC644538
  250. TSSK6
  251. TOM1
  252. KHDRBS1
  253. C5orf36
  254. RAB40B
  255. RBMS3
  256. IFIT5
  257. PPIP5K1
  258. PPFIA4
  259. CMC1
  260. MIMAT0000245
  261. TMEM116
  262. PLCB1
  263. SRD5A3
  264. PPT1
  265. ANKRD32
  266. ZNF792
  267. PHF7
  268. RAB8B
  269. MID1IP1
  270. BNIP2
  271. SIRT5
  272. PRPF40B
  273. DDR1
  274. OSMR
  275. RB1
  276. PAIP2
  277. HSD17B11
  278. AK3L1
  279. ANTXR2
  280. FOXJ3
  281. ADRA1B
  282. CUEDC1
  283. AFAP1
  284. AIDA
  285. SRGAP2
  286. RSU1
  287. SFT2D2
  288. ZNF777
  289. CCDC88A
  290. RNF219
  291. MIMAT0004493
  292. C15orf52
  293. FOXRED1
  294. TROVE2
  295. KCTD3
  296. AHSA1
  297. DEF8
  298. BCO2
  299. LLGL2
  300. FAM110B
  301. SCAMP4
  302. SSR1
  303. WDFY1
  304. LPCAT1
  305. COQ10A
  306. PISD
  307. EIF2C1
  308. MBNL2
  309. C3orf58
  310. C5orf15
  311. NAP1L4
  312. ADPRH
  313. DARS
  314. AAGAB
  315. TMEM45A
  316. TMEM51
  317. MIMAT0000083
  318. WIPI2
  319. FBXO17
  320. NR3C1
  321. PCBP4
  322. ANP32E
  323. TRPM4
  324. C15orf59
  325. MSH6
  326. PSKH1
  327. ARRDC3
  328. CALCOCO2
  329. PLSCR4
  330. SLC25A38
  331. CKB
  332. ACAD11
  333. ZNF25
  334. FAM160B1
  335. TMEM8B
  336. KCNK3
  337. SLC39A14
  338. REPIN1
  339. ITGB1
  340. TMEM26
  341. C14orf159
  342. UBAC1
  343. DHCR7
  344. PPP3R1
  345. CASP10
  346. METRNL
  347. CXorf57
  348. ST14
  349. ADCK1
  350. RALB
  351. MYO19
  352. PARP3
  353. ICK
  354. CLEC16A
  355. ABL2
  356. STARD13
  357. SLC25A35
  358. TMEM37
  359. C3orf31
  360. SDR39U1
  361. CD109
  362. PYGB
  363. C19orf52
  364. SERTAD2
  365. TCP11L1
  366. TMEM55B
  367. C19orf73
  368. F8
  369. TM7SF2
  370. KATNAL1
  371. SLC2A5
  372. BNIP3
  373. TGFA
  374. SHROOM4
  375. ZDHHC23
  376. HAGHL
  377. RIT1
  378. DYRK3
  379. DBN1
  380. MIMAT0004809
  381. FTO
  382. EDN1
  383. PDIA5
  384. ACLY
  385. MXI1
  386. DDX58
  387. GDE1
  388. C1orf53
  389. EHBP1
  390. PAK4
  391. TLR3
  392. KCNJ11
  393. SYNGR3
  394. SSPN
  395. MSN
  396. FAM155B
  397. MAEA
  398. PLEKHA2
  399. CDK19
  400. BNIP3L