DTSA_5509_final_exam

July 31, 2023

0.0.1 1. Project Topic

1. For my final project I've decided to use genetic data from cancer patients with gliomas. Gliomas is the type of brain cancer and for clinicians it's really important to distinguish between two main types of this tumor - LGG (Lower-Grade Glioma) and GBM (Glioblastoma Multiforme). Dataset has information about 20 genes (mutated or not, binary value) and 3 clinical features. And the main point of this project is to find optimal subset of mutation genes and clinical features for the glioma types.
2. We can consider this problem as specific subset of supervised learning where we would need to find best set of predictors that will give us the best results
3. So the main goal would be to build classifier for types of glioblastoma and then find best subset of features for this classifier

0.0.2 2. Data source

1. I got dataset from the study in International Journal of Molecular Sciences [1]

And here is a link to download the data - https://archive.ics.uci.edu/static/public/759/glioma+grading+clinical+a

2. Dataset is in csv format and has data about 839 patients (rows) and 23 attributes (columns). Clinical attributes are: Race (Categorical), Gender (Categorical) and Age_at_diagnosis (Continious). Also data has information about mutations in 20 genes, all these variables are Categorical with only 2 values: NOT_MUTATED and MUTATED. All data is within the single table

[1]. Tasci, E., Zhuge, Y., Kaur, H., Camphausen, K., & Krauze, A. V. (2022). Hierarchical Voting-Based Feature Selection and Ensemble Learning Model Scheme for Glioma Grading with Clinical and Molecular Characteristics. International Journal of Molecular Sciences, 23(22), 14155.

[2]:
```
import re
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

0.0.3 3. Data cleaning

[3]:
```
dataset = pd.read_csv("/Users/alexey/Downloads/
↪glioma+grading+clinical+and+mutation+features+dataset/
↪TCGA_GBM_LGG_Mutations_all.csv")
```

[4]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 862 entries, 0 to 861
Data columns (total 27 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Grade              862 non-null    object
 1   Project            862 non-null    object
 2   Case_ID            862 non-null    object
 3   Gender             862 non-null    object
 4   Age_at_diagnosis   862 non-null    object
 5   Primary_Diagnosis  862 non-null    object
 6   Race               862 non-null    object
 7   IDH1               862 non-null    object
 8   TP53               862 non-null    object
 9   ATRX               862 non-null    object
 10  PTEN               862 non-null    object
 11  EGFR               862 non-null    object
 12  CIC                862 non-null    object
 13  MUC16              862 non-null    object
 14  PIK3CA             862 non-null    object
 15  NF1                862 non-null    object
 16  PIK3R1             862 non-null    object
 17  FUBP1              862 non-null    object
 18  RB1                862 non-null    object
 19  NOTCH1             862 non-null    object
 20  BCOR               862 non-null    object
 21  CSMD3              862 non-null    object
 22  SMARCA4            862 non-null    object
 23  GRIN2A             862 non-null    object
 24  IDH2               862 non-null    object
 25  FAT4               862 non-null    object
 26  PDGFRA             862 non-null    object
dtypes: object(27)
memory usage: 182.0+ KB
```

0.0.4 3.1 First let's explore how many nas we have in the dataset and then decide what we can do with these values

[5]:
```
dataset.isna().any()
```

[5]:
```
Grade                False
Project              False
Case_ID              False
Gender               False
Age_at_diagnosis     False
Primary_Diagnosis    False
```

```
Race                     False
IDH1                     False
TP53                     False
ATRX                     False
PTEN                     False
EGFR                     False
CIC                      False
MUC16                    False
PIK3CA                   False
NF1                      False
PIK3R1                   False
FUBP1                    False
RB1                      False
NOTCH1                   False
BCOR                     False
CSMD3                    False
SMARCA4                  False
GRIN2A                   False
IDH2                     False
FAT4                     False
PDGFRA                   False
dtype: bool
```

standard approach doesn't show any missing values, but these values might just be encoded differently; let's print out all unique values for every columns excluding Project and Case_ID, as these columns won't be used in the learning

[6]:
```
for column_name in dataset.columns:
    if column_name not in ['Project', 'Case_ID']:
        print(f"{column_name}\t{','.join(dataset[column_name].unique())}")
```

```
Grade   LGG,GBM
Gender  Male,Female,--
Age_at_diagnosis        51 years 108 days,38 years 261 days,35 years 62 days,32
years 283 days,31 years 187 days,33 years 78 days,35 years 68 days,44 years 239
days,33 years 350 days,87 years,51 years 328 days,54 years 95 days,52 years 214
days,47 years 123 days,34 years 132 days,40 years 192 days,53 years 352 days,41
years 70 days,43 years 161 days,37 years 159 days,47 years 173 days,31 years 8
days,25 years 191 days,66 years 305 days,56 years 250 days,35 years 362 days,51
years 363 days,37 years 32 days,54 years 183 days,32 years 76 days,65 years 28
days,43 years 131 days,51 years 59 days,43 years 221 days,25 years 214 days,45
years 24 days,50 years 153 days,27 years 166 days,53 years 252 days,46 years 144
days,24 years 239 days,--,34 years 70 days,29 years 198 days,45 years 124
days,62 years 90 days,46 years 224 days,36 years 247 days,62 years 202 days,70
years 159 days,53 years 41 days,48 years 124 days,40 years 69 days,40 years 7
days,20 years 359 days,57 years 200 days,38 years 322 days,52 years 192 days,56
years 104 days,59 years 275 days,67 years 107 days,48 years 346 days,59 years
254 days,58 years 147 days,27 years 247 days,51 years 230 days,74 years 11
```

days,52 years 230 days,61 years 62 days,66 years 146 days,42 years 32 days,31 years 344 days,48 years 268 days,33 years 340 days,34 years 213 days,24 years 54 days,55 years 48 days,27 years 323 days,29 years 32 days,39 years 131 days,70 years 3 days,30 years 338 days,39 years 178 days,25 years 41 days,48 years 160 days,57 years 17 days,34 years 307 days,58 years 137 days,55 years 208 days,60 years 49 days,38 years 13 days,56 years 113 days,54 years 180 days,31 years 152 days,54 years 146 days,52 years 241 days,33 years 54 days,42 years 173 days,20 years 75 days,64 years 106 days,29 years 232 days,36 years 222 days,31 years 362 days,36 years 67 days,32 years 173 days,38 years 292 days,29 years 298 days,41 years 76 days,35 years 189 days,33 years 39 days,32 years 50 days,49 years 276 days,69 years 217 days,14 years 154 days,33 years 349 days,47 years 92 days,33 years 58 days,63 years 364 days,62 years 205 days,37 years 83 days,53 years 183 days,53 years 74 days,45 years 76 days,33 years 331 days,53 years 359 days,52 years 156 days,64 years 304 days,71 years 70 days,40 years 300 days,36 years 354 days,25 years 249 days,63 years 170 days,40 years 310 days,47 years 215 days,73 years 241 days,24 years 51 days,48 years 342 days,41 years 325 days,36 years 189 days,45 years 334 days,44 years 204 days,27 years 6 days,50 years 181 days,42 years 141 days,40 years 248 days,62 years 192 days,33 years 195 days,46 years 42 days,41 years 117 days,47 years 333 days,56 years 159 days,38 years 64 days,60 years 64 days,29 years 150 days,59 years 196 days,44 years 66 days,69 years 60 days,63 years 253 days,43 years 142 days,61 years 316 days,39 years 66 days,31 years 315 days,51 years 164 days,41 years 203 days,60 years 360 days,74 years 158 days,63 years 81 days,66 years 170 days,51 years 312 days,39 years 62 days,38 years 111 days,37 years 92 days,30 years 32 days,54 years 143 days,36 years 357 days,48 years 197 days,29 years 81 days,32 years 135 days,44 years 35 days,30 years 329 days,57 years 287 days,19 years 55 days,70 years 242 days,63 years 67 days,41 years 255 days,29 years 291 days,42 years 318 days,51 years 172 days,38 years 25 days,28 years 344 days,32 years 166 days,26 years 43 days,67 years 329 days,40 years 326 days,59 years 295 days,58 years 16 days,56 years 171 days,20 years 276 days,49 years 296 days,37 years 31 days,48 years 243 days,71 years 227 days,43 years 316 days,27 years 309 days,25 years 78 days,30 years 126 days,47 years 335 days,31 years 262 days,30 years 295 days,30 years 276 days,24 years 146 days,41 years 66 days,34 years 189 days,56 years 155 days,39 years 348 days,23 years 3 days,31 years 273 days,17 years 271 days,64 years 110 days,27 years 221 days,69 years 64 days,37 years 51 days,44 years 92 days,25 years 97 days,29 years 235 days,26 years 209 days,47 years 294 days,39 years 252 days,35 years 26 days,33 years 287 days,74 years 204 days,50 years 167 days,65 years 314 days,60 years 116 days,34 years 346 days,44 years 229 days,21 years 243 days,55 years 87 days,43 years 361 days,36 years 49 days,70 years 100 days,23 years 352 days,43 years 195 days,30 years 264 days,38 years 140 days,34 years 188 days,55 years 92 days,30 years 238 days,61 years 68 days,44 years 344 days,35 years 238 days,68 years 129 days,44 years 206 days,38 years 89 days,36 years 304 days,52 years 32 days,36 years 183 days,33 years 176 days,60 years 119 days,44 years 90 days,58 years 157 days,33 years 364 days,62 years 235 days,44 years 226 days,36 years 311 days,29 years 132 days,25 years 282 days,20 years 307 days,61 years 305 days,41 years 222 days,47 years 224 days,59 years 212 days,48 years 311 days,38 years 24 days,27 years 58 days,58 years 15 days,26 years 118 days,44 years 209 days,32 years 196 days,22 years 305 days,25 years 316 days,34 years 27 days,24 years 278 days,30 years 348 days,39 years 280 days,22 years 236 days,53 years 286 days,30 years 355 days,37 years 239 days,36 years 73 days,41 years 314 days,57 years 201 days,52 years 243 days,31 years 346 days,32 years 167 days,44 years 131 days,40 years 106 days,44 years 263 days,39 years 5 days,62 years 93 days,45 years 296 days,48 years 131 days,49 years 187 days,38 years 194 days,28 years 115 days,38 years 273 days,37 years 81 days,41 years 179 days,60 years 15 days,50 years 232 days,75 years 292 days,43 years 143 days,52 years 63 days,38 years 333 days,66 years 344 days,30 years 57 days,59 years 262 days,35 years 37 days,38 years 203 days,42 years 310 days,59 years 166 days,26 years 94 days,58 years 266 days,31 years 10 days,30 years 161 days,45 years 231 days,37 years 166 days,34 years 162 days,34 years 339 days,32 years 53 days,66 years 176 days,46 years 89 days,38 years 326 days,33 years 213 days,40 years 61 days,52 years 145 days,59 years 359 days,58 years 55 days,43 years 281 days,33 years 192 days,55 years 40 days,43 years 226 days,60 years 123 days,33 years 332 days,48 years 80 days,70 years 205 days,53 years 197 days,66 years 136 days,36 years 275 days,40 years 49 days,73 years 103 days,36 years 165 days,74 years 56 days,43 years 145 days,40 years 323 days,38 years 128 days,54 years 291 days,63 years 54 days,62 years 303 days,35 years 53 days,39 years 174 days,29 years 341 days,46 years 333 days,46 years 110 days,51 years 329 days,54 years 221 days,42 years 91 days,64 years 108 days,29 years 220 days,65 years 195 days,41 years 91 days,31 years 206 days,58 years 5 days,54 years 318 days,28 years 26 days,20 years 116 days,34 years 160 days,54 years 198 days,45 years 259 days,48 years 325 days,29 years 86 days,22 years 8 days,32 years 161 days,40 years 50 days,61 years 47 days,57 years 276 days,54 years 100 days,56 years 317 days,34 years 139 days,27 years 289 days,37 years 149 days,43 years 332 days,41 years 10 days,33 years 104 days,39 years 35 days,31 years 300 days,67 years 219 days,30 years 140 days,48 years 212 days,50 years 302 days,45 years 34 days,47 years 299 days,32 years 158 days,26 years 171 days,29 years 213 days,28 years 79 days,66 years 16 days,39 years 304 days,33 years 238 days,60 years 75 days,35 years 65 days,35 years 45 days,58 years 168 days,26 years 50 days,30 years 113 days,38 years 82 days,35 years 325 days,39 years 291 days,55 years 306 days,49 years 337 days,50 years 59 days,52 years 163 days,51 years 121 days,61 years 285 days,52 years 149 days,35 years 66 days,47 years 266 days,35 years 70 days,31 years 105 days,31 years 35 days,37 years 50 days,37 years 97 days,42 years 249 days,32 years 116 days,61 years 22 days,48 years 78 days,34 years 42 days,30 years 252 days,30 years 86 days,28 years 195 days,47 years 226 days,35 years 91 days,23 years 171 days,49 years 264 days,36 years 78 days,54 years 268 days,22 years 300 days,42 years 5 days,28 years 213 days,25 years 232 days,66 years 3 days,46 years 181 days,57 years 310 days,43 years 22 days,49 years 309 days,33 years 99 days,27 years 306 days,62 years 166 days,55 years 153 days,37 years 285 days,37 years 107 days,32 years 122 days,38 years 143 days,42 years 289 days,47 years 165 days,54 years 227 days,29 years 49 days,39 years 359 days,28 years 249 days,23 years 108 days,21 years 276 days,38 years 229 days,49 years 302 days,38 years 308 days,22 years 205 days,67 years 215 days,73 years 329 days,62 years 100 days,57 years 314 days,30 years 334 days,24 years 181 days,20 years 84 days,41 years 116 days,53 years 258 days,34 years 237 days,51 years 350 days,57 years 364 days,31 years 176 days,57 years 141 days,49 years 263 days,60 years 106 days,73 years 250 days,64 years 298 days,73 years 164 days,33 years 239 days,67 years 151 days,72 years 74 days,69 years 124 days,79 years 39 days,68 years 7 days,76 years 5 days,82 years 14 days,58 years 9 days,52 years 251 days,81 years 217 days,74 years 172 days,76 years 87 days,57 years 110 days,67 years 121 days,63 years 313 days,64 years 143 days,69 years 50 days,66 years 28 days,54 years 347 days,60 years 114 days,45 years 230 days,36 years 219 days,49 years 174 days,69 years 219 days,54 years 340 days,58 years 327 days,61 years 183 days,64 years 191 days,65 years 309 days,60 years 298 days,53 years 309 days,39 years 193 days,58 years 227 days,56 years 172 days,65 years 344 days,36 years 59 days,72 years 192 days,48 years 348 days,39 years 157 days,54 years 176 days,63 years 333 days,66 years 159 days,52 years 97 days,89 years 105 days,54 years 242 days,60 years 262 days,77 years 116 days,60 years 264 days,80 years 61 days,51 years 264 days,53 years 145 days,43 years 313 days,74 years 144 days,76 years 118 days,49 years 228 days,48 years 333 days,68 years 334 days,74 years 359 days,69 years 211 days,64 years 192 days,58 years 144 days,47 years 88 days,61 years 282 days,75 years 194 days,52 years 246 days,47 years 199 days,21 years 288 days,72 years 169 days,68 years 108 days,67 years,52 years 238 days,59 years 11 days,59 years 149 days,48 years 362 days,76 years 58 days,72 years 193 days,51 years 113 days,67 years 187 days,82 years 115 days,64 years 209 days,75 years 265 days,85 years 221 days,73 years 105 days,78 years 273 days,45 years 136 days,42 years 225 days,53 years 299 days,68 years 290 days,43 years 245 days,59 years 289 days,52 years 137 days,61 years 244 days,40 years 207 days,63 years 263 days,63 years 307 days,70 years 98 days,59 years 362 days,56 years 101 days,62 years 190 days,24 years 83 days,72 years 345 days,67 years 161 days,54 years 200 days,86 years 155 days,62 years 222 days,63 years 97 days,58 years 200 days,60 years 5 days,67 years 17 days,23 years 310 days,58 years 59 days,64 years 18 days,65 years 37 days,61 years 32 days,52 years 161 days,77 years 139 days,58 years 280 days,78 years 264 days,63 years 122 days,63 years 86 days,23 years 133 days,65 years 94 days,50 years 79 days,55 years 209 days,60 years 258 days,62 years 220 days,69 years 95 days,56 years 187 days,76 years 72 days,76 years 252 days,86 years 8 days,69 years 80 days,59 years 219 days,75 years 337 days,57 years 202 days,69 years 257 days,45 years 226 days,47 years 130 days,74 years 57 days,78 years 93 days,59 years 64 days,21 years 266 days,81 years 34 days,70 years 60 days,62 years 153 days,75 years 191 days,60 years 136 days,63 years 53 days,61 years 52 days,47 years 71 days,76 years 150 days,84 years 287 days,77 years 153 days,48 years 182 days,69 years 223 days,56 years 12 days,76 years 171 days,70 years 117 days,34 years 267 days,68 years 341 days,54 years 216 days,74 years 313 days,73 years 193 days,57 years 112 days,44 years 230 days,83 years 265 days,74 years 291 days,51 years 275 days,52 years 60 days,73 years 102 days,81 years 307 days,78 years 94 days,57 years 312 days,53 years 147 days,50 years 330 days,78 years 56 days,76 years 236 days,57 years 118 days,36 years 108 days,72 years 284 days,31 years 11 days,65 years 66 days,63 years 201 days,49 years 351 days,59 years 311 days,69 years 232 days,62 years 304 days,52 years 270 days,56 years 294 days,81 years 168 days,40 years 293 days,81 years 319 days,50 years 171 days,38 years 8 days,48 years 183 days,51 years 86 days,66 years 341 days,71 years 315 days,24 years 156 days,63 years 195 days,69 years 116 days,78 years 7 days,30 years 155 days,52 years 288 days,65 years 57 days,78 years 271 days,61 years 37 days,50 years 173 days,39 years 57 days,66 years 81 days,49 years 6 days,54 years 78 days,58 years 48 days,73 years 4 days,53 years

303 days,63 years 292 days,66 years 191 days,49 years 322 days,59 years 335 days,54 years 327 days,57 years 274 days,59 years 157 days,44 years 55 days,58 years 14 days,65 years 18 days,67 years 34 days,75 years 172 days,88 years 209 days,53 years 85 days,75 years 126 days,66 years 269 days,51 years 51 days,40 years 360 days,58 years 6 days,57 years 29 days,55 years 144 days,40 years 268 days,40 years 206 days,54 years 89 days,53 years 136 days,56 years 9 days,59 years 306 days,75 years 118 days,59 years 10 days,44 years 135 days,46 years 236 days,68 years 248 days,30 years 340 days,36 years 302 days,78 years 270 days,83 years 235 days,51 years 201 days,63 years 118 days,59 years 227 days,53 years 233 days,74 years 83 days,60 years 246 days,54 years 295 days,63 years 9 days,58 years 308 days,60 years 274 days,71 years 257 days,75 years 78 days,50 years 309 days,67 years 6 days,60 years 358 days,65 years 143 days,51 years 90 days,45 years 301 days,39 years 316 days,70 years 245 days,73 years 133 days,65 years 360 days,72 years 251 days,61 years 177 days,65 years 22 days,30 years 327 days,64 years 50 days,55 years 267 days,79 years 123 days,74 years 294 days,63 years 24 days,81 years 21 days,53 years 86 days,69 years 343 days,60 years 342 days,55 years 214 days,75 years 342 days,47 years 349 days,71 years 148 days,25 years 82 days,61 years 247 days,47 years 302 days,50 years 129 days,72 years 97 days,31 years 7 days,86 years 216 days,38 years 214 days,72 years 302 days,55 years 46 days,66 years 320 days,55 years 361 days,72 years 57 days,33 years 7 days,83 years 114 days,43 years 259 days,68 years 224 days,79 years 183 days,67 years 326 days,49 years 298 days,66 years 256 days,68 years 221 days,47 years 52 days,61 years 218 days,49 years 157 days,67 years 120 days,32 years 268 days,65 years 41 days,66 years 213 days,62 years 281 days,50 years 16 days,34 years 111 days,61 years 11 days,77 years 353 days,54 years 101 days,51 years 32 days,53 years 8 days,78 years 253 days,46 years 337 days,52 years 244 days,48 years 249 days,51 years 205 days,62 years 41 days,58 years 20 days,61 years 112 days,66 years 111 days,64 years 43 days,56 years 114 days,77 years 325 days,85 years 65 days,77 years 178 days,63 years 121 days,76 years 221 days
Primary_Diagnosis Oligodendroglioma, NOS,Mixed glioma,Astrocytoma, NOS,Astrocytoma, anaplastic,Oligodendroglioma, anaplastic,--,Glioblastoma
Race white,asian,black or african american,--,not reported,american indian or alaska native
IDH1 MUTATED,NOT_MUTATED
TP53 NOT_MUTATED,MUTATED
ATRX NOT_MUTATED,MUTATED
PTEN NOT_MUTATED,MUTATED
EGFR NOT_MUTATED,MUTATED
CIC NOT_MUTATED,MUTATED
MUC16 NOT_MUTATED,MUTATED
PIK3CA MUTATED,NOT_MUTATED
NF1 NOT_MUTATED,MUTATED
PIK3R1 NOT_MUTATED,MUTATED
FUBP1 MUTATED,NOT_MUTATED
RB1 NOT_MUTATED,MUTATED
NOTCH1 NOT_MUTATED,MUTATED
BCOR NOT_MUTATED,MUTATED
CSMD3 NOT_MUTATED,MUTATED

```
SMARCA4 NOT_MUTATED,MUTATED
GRIN2A  NOT_MUTATED,MUTATED
IDH2    NOT_MUTATED,MUTATED
FAT4    NOT_MUTATED,MUTATED
PDGFRA  NOT_MUTATED,MUTATED
```

from the cell above I can see that:

1. missing values are encoded as '–' and as 'not reported'.
2. Age is in string format, so we would need to convert it to some continious form. I would suggest to convert years to days and add these value to the remaining number of days. I won't take into accout leap years as I woulnd't expect this precision to add any value to our model
3. Other categorical variables should be converted to 0s or 1s

so first let's cound how many rows with missing values we have. For this I will need to explore just three columns: Gender, Age_at_diagnosis and Race. I don't need to explore Primary_Diagnosis column as Grade will be used instead for classification

[7]:
```python
# Find indices of missing values for Gender column
gender_missing_indices = dataset.index[dataset.Gender.isin(['--', 'not␣
↪reported'])].tolist()
```

[8]:
```python
# Find indices of missing values for Age_at_diagnosis column
age_missing_indices = dataset.index[dataset.Age_at_diagnosis.isin(['--', 'not␣
↪reported'])].tolist()
```

[9]:
```python
# Find indices of missing values for Race column
race_missing_indices = dataset.index[dataset.Race.isin(['--', 'not reported'])].
↪tolist()
```

[10]:
```python
# Merge all indices into one list
missing_values_indices = list()
missing_values_indices.extend(gender_missing_indices)
missing_values_indices.extend(age_missing_indices)
missing_values_indices.extend(race_missing_indices)
```

[11]:
```python
# Use set to remove non-unique indices
print(f"Number of missing values: {len(set(missing_values_indices))}")
print(f"Missing values indices: {set(missing_values_indices)}")
```

```
Number of missing values: 23
Missing values indices: {256, 268, 396, 525, 794, 671, 163, 41, 437, 706, 71,
455, 583, 846, 208, 341, 608, 231, 490, 747, 622, 623, 504}
```

As we can see there are only 23 rows with missing values so I think it will be easier to just remove these rows as it won't affect anyhow our future learning model. Also let's remove 'Primary_Diagnosis', 'Project' and 'Case_ID' columns as we won't use them in the analysis

[12]:
```
# let's remove 'Primary_Diagnosis', 'Project' and 'Case_ID' columns from the␣
↪dataset
dataset = dataset.drop(['Primary_Diagnosis', 'Project', 'Case_ID'], axis=1)
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 862 entries, 0 to 861
Data columns (total 24 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Grade              862 non-null    object
 1   Gender             862 non-null    object
 2   Age_at_diagnosis   862 non-null    object
 3   Race               862 non-null    object
 4   IDH1               862 non-null    object
 5   TP53               862 non-null    object
 6   ATRX               862 non-null    object
 7   PTEN               862 non-null    object
 8   EGFR               862 non-null    object
 9   CIC                862 non-null    object
 10  MUC16              862 non-null    object
 11  PIK3CA             862 non-null    object
 12  NF1                862 non-null    object
 13  PIK3R1             862 non-null    object
 14  FUBP1              862 non-null    object
 15  RB1                862 non-null    object
 16  NOTCH1             862 non-null    object
 17  BCOR               862 non-null    object
 18  CSMD3              862 non-null    object
 19  SMARCA4            862 non-null    object
 20  GRIN2A             862 non-null    object
 21  IDH2               862 non-null    object
 22  FAT4               862 non-null    object
 23  PDGFRA             862 non-null    object
dtypes: object(24)
memory usage: 161.8+ KB
```

[13]:
```
# let's remove rows with missing values
dataset = dataset.drop(set(missing_values_indices), axis=0)
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 839 entries, 0 to 861
Data columns (total 24 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Grade              839 non-null    object
 1   Gender             839 non-null    object
```

```
 2   Age_at_diagnosis  839 non-null    object
 3   Race              839 non-null    object
 4   IDH1              839 non-null    object
 5   TP53              839 non-null    object
 6   ATRX              839 non-null    object
 7   PTEN              839 non-null    object
 8   EGFR              839 non-null    object
 9   CIC               839 non-null    object
 10  MUC16             839 non-null    object
 11  PIK3CA            839 non-null    object
 12  NF1               839 non-null    object
 13  PIK3R1            839 non-null    object
 14  FUBP1             839 non-null    object
 15  RB1               839 non-null    object
 16  NOTCH1            839 non-null    object
 17  BCOR              839 non-null    object
 18  CSMD3             839 non-null    object
 19  SMARCA4           839 non-null    object
 20  GRIN2A            839 non-null    object
 21  IDH2              839 non-null    object
 22  FAT4              839 non-null    object
 23  PDGFRA            839 non-null    object
dtypes: object(24)
memory usage: 163.9+ KB
```

[14]:

```
# Convert years to days
for row in dataset.iterrows():
    numbers = re.findall(r'\d+', row[1]['Age_at_diagnosis'])
    age = 0
    years_to_day = int(numbers[0]) * 365
    if len(numbers) > 1:
        age = years_to_day + int(numbers[1])
    else:
        age = years_to_day
    row[1]['Age_at_diagnosis'] = years_to_day
```

[15]:

```
# Convert Gender to number, Male = 0, Female = 1
for row in dataset.iterrows():
    if row[1]['Gender'] == 'Male':
        row[1]['Gender'] = 0
    else:
        row[1]['Gender'] = 1
```

[16]:

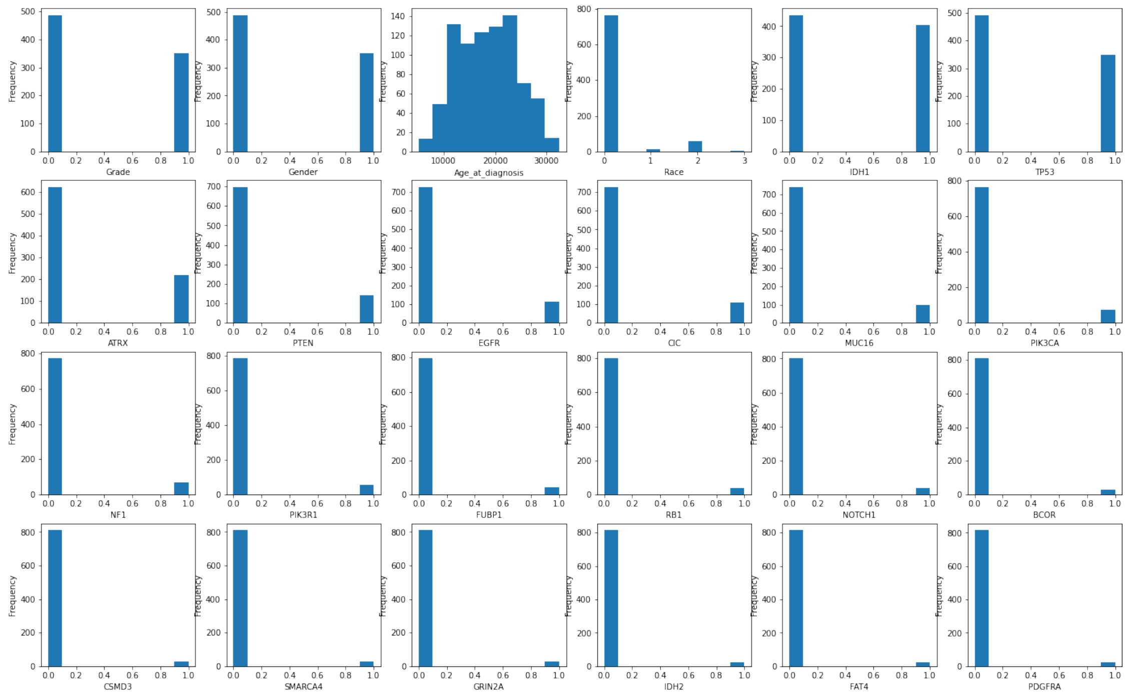
```
# Convert Grade to number, LGG = 0, GBM = 1
for row in dataset.iterrows():
    if row[1]['Grade'] == 'LGG':
        row[1]['Grade'] = 0
```

```
    else:
        row[1]['Grade'] = 1
```

[17]:
```
# Convert Race to humber, white = 0, asian = 1, black or african american = 2,␣
↪american indian or alaska native = 3
for row in dataset.iterrows():
    if row[1]['Race'] == 'white':
        row[1]['Race'] = 0
    elif row[1]['Race'] == 'asian':
        row[1]['Race'] = 1
    elif row[1]['Race'] == 'black or african american':
        row[1]['Race'] = 2
    else:
        row[1]['Race'] = 3
```

[18]:
```
# Convert genes mutation status to number, MUTATED = 1, NOT_MUTATED = 0
gene_names = [gene_name for gene_name in dataset.columns if gene_name not in␣
↪['Grade', 'Gender', 'Age_at_diagnosis', 'Race']]
for row in dataset.iterrows():
    for gene_name in gene_names:
        if row[1][gene_name] == 'NOT_MUTATED':
            row[1][gene_name] = 0
        else:
            row[1][gene_name] = 1
```

[235]:
```
# let's plot histogram of existin values just to check that cleaning was OK
fig = plt.figure(figsize=(24, 15))
i = 0
for column in dataset:
    sub = fig.add_subplot(4, 6, i+1)
    sub.set_xlabel(column)
    dataset[column].plot(kind='hist')
    i += 1
```



0.0.5 3.2 Cleaning stage conclusion:

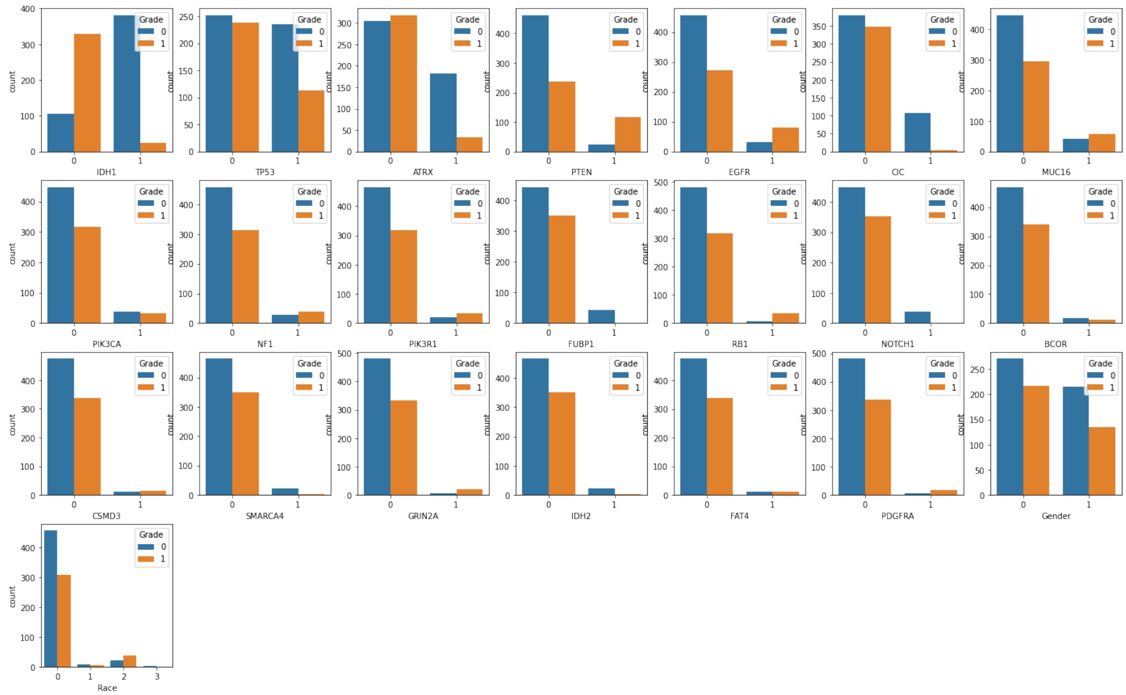
1. We've removed all missing values
2. We've converted Age to the number of days
3. We've convert all other categorical variables to 0s, 1s, etc…

As I can see from the plot above, Grade, Gender and IDH1 variables looks balanced. Age column more or less similar to normal distribution as it should be and other variables, including Race and most of the genes are quite unbalanced (there are more values of one type than another)

0.0.6 4. Exploratory Data Analysis

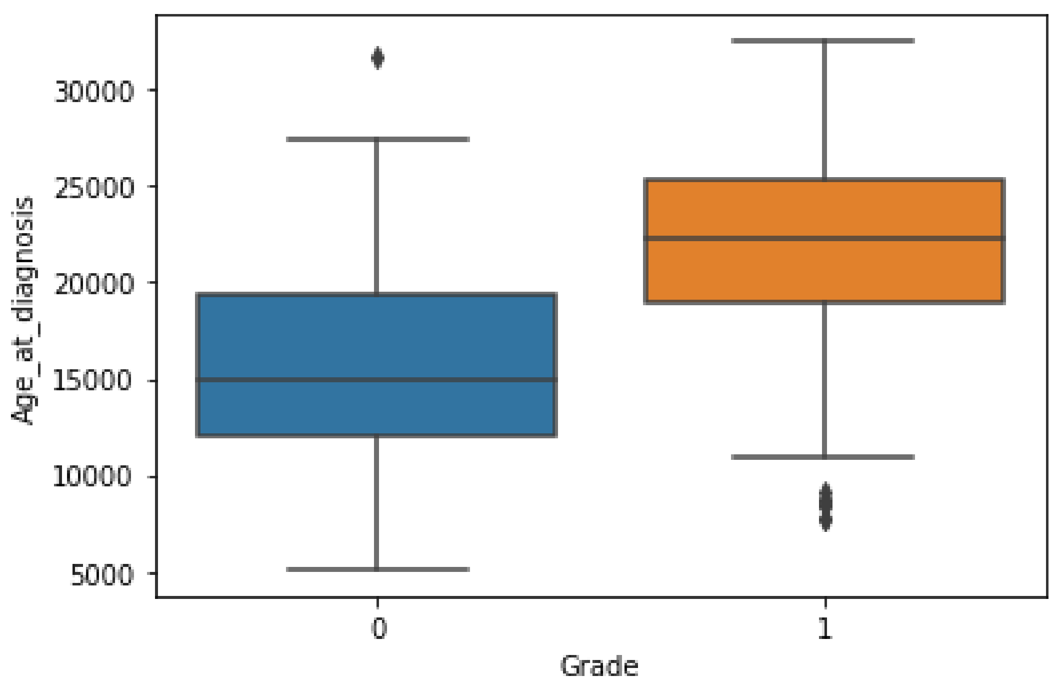
[236]:
```python
# I've already built histogram of features above and now it would be great to check feature vs target distribution
list_of_columns = gene_names + ['Gender', 'Race']
fig = plt.figure(figsize=(24, 15))

for i in range(len(list_of_columns)):
    column = list_of_columns[i]
    sub = fig.add_subplot(4, 7, i+1)
    chart = sns.countplot(data=dataset, x=column, hue='Grade')
```



[237]:
```python
# And finally I would like to build similar plot but for Age_at_diagnosis column
sns.boxplot(x='Grade', y='Age_at_diagnosis', data=dataset)
```

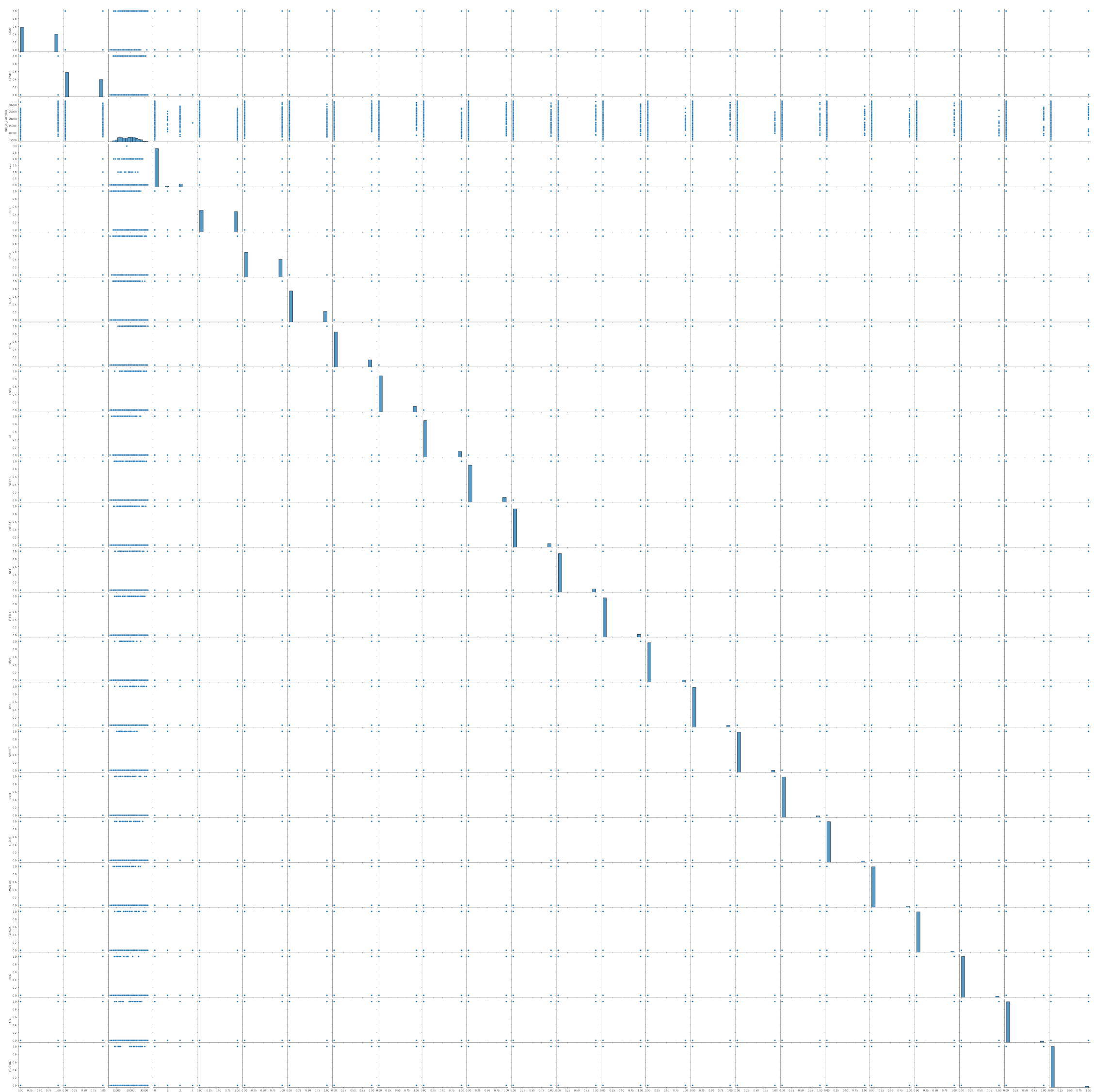
[237]: <Axes: xlabel='Grade', ylabel='Age_at_diagnosis'>



[238]:
```python
# And finally let's build correlation plot just to check if any variables are
  ↪correlated
sns.pairplot(dataset)
```

```
/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-
packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to
tight
  self._figure.tight_layout(*args, **kwargs)
```

[238]: <seaborn.axisgrid.PairGrid at 0x2b82c76d0>



0.0.7 4.1 EDA summary:

1. Many genes represent unbalanced features - it mean that probably they won't give any advantage for a future model
2. For other variables it's possible to see that Grade target is correlated with state of the predictor, it means these feature will be useful for future model
3. Looks like Age_at_diagnosis in generate is linked with GBM

0.0.8 5. Models

there are many classification algorithms that we can use to predict grade of cancer in this case. I would suggest to try main classification algorithms with default parameters (logistic regression, naive bayes, SVM, K-nearest neighbours, decision tree, random forest and gradient boosting as one of ensemble methods), choose the best one and then try to find the best parameters for best algorithm

[239]:
```
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix
```

[240]:
```
# now we need to split our dataset into traning and test datasets
# remove Grade from dataset
X = dataset.drop(['Grade'], axis=1)
# and save it to y varibale
y = dataset['Grade']

# split dataset into train and test with 70% to 30% ratio
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
y_train_transformed = y_train.astype('int')
y_test_transformed = y_test.astype('int')
```

[241]:
```
pipeline = list()
pipeline.append(LogisticRegression(solver='liblinear'))
pipeline.append(GaussianNB())
pipeline.append(SVC())
# I've tried to use KNeighborsClassifier as well but looks like it has some bug␣
↪in library thas was fixed only a couple
# of weeks ago and maybe didn't get to the production yet
# pipeline.append(KNeighborsClassifier())
pipeline.append(DecisionTreeClassifier())
```

```
pipeline.append(RandomForestClassifier())
pipeline.append(GradientBoostingClassifier())
```

[242]:
```
accuracy_list = list()
auc_list = list()
cm_list = list()
for model in pipeline:
    print(model)
    model.fit(X_train, y_train_transformed)
    y_pred = model.predict(X_test)
    accuracy_list.append(metrics.accuracy_score(y_test_transformed, y_pred))
    fpr, tpr, _tresholds = metrics.roc_curve(y_test_transformed, y_pred)
    auc_list.append(round(metrics.auc(fpr, tpr), 2))
    cm_list.append(confusion_matrix(y_test_transformed, y_pred))
```

```
LogisticRegression(solver='liblinear')
GaussianNB()
SVC()
DecisionTreeClassifier()
RandomForestClassifier()
GradientBoostingClassifier()
```

[243]:
```
models = ['LogisticRegression', 'GaussianNB', 'SVC', 'DecisionTreeClassifier',␣
↪'RandomForestClassifier', 'GradientBoostingClassifier']

results = pd.DataFrame({'Model': models, 'Accuracy': accuracy_list, 'AUC':␣
↪auc_list})
results
```

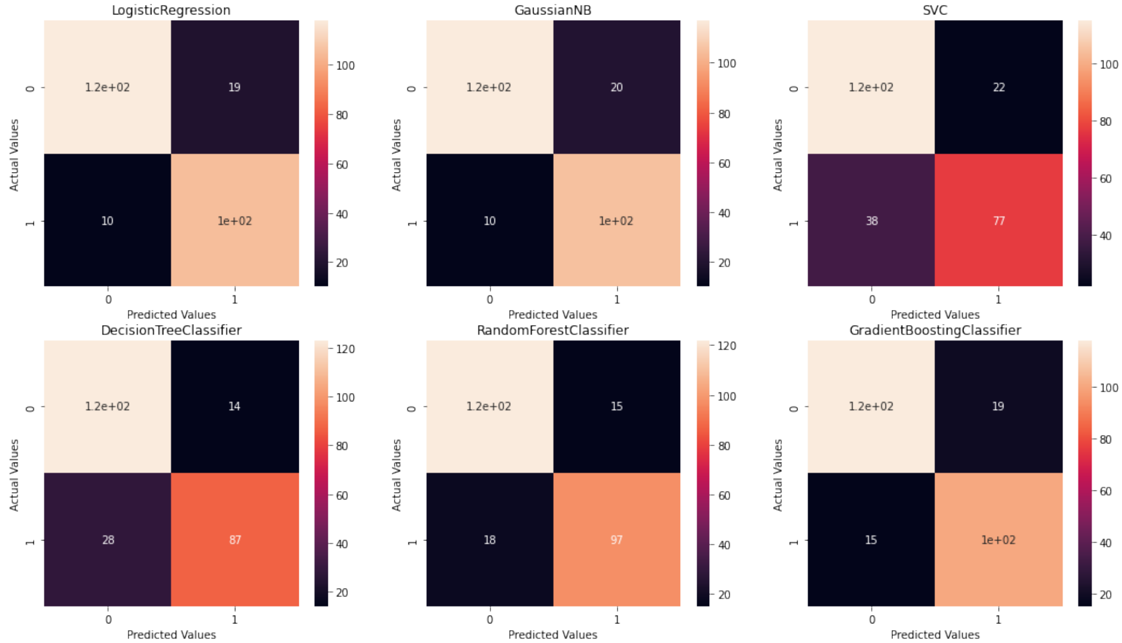
[243]:

	Model	Accuracy	AUC
0	LogisticRegression	0.884921	0.89
1	GaussianNB	0.880952	0.88
2	SVC	0.761905	0.75
3	DecisionTreeClassifier	0.833333	0.83
4	RandomForestClassifier	0.869048	0.87
5	GradientBoostingClassifier	0.865079	0.87

Suprisingly but simple LogisticRegression analysis gives the best results with default parameters

[244]:
```
# Now let's looks into confustion matrices
fig = plt.figure(figsize=(18, 10))
for i in range(len(cm_list)):
    confussion_matrix = cm_list[i]
    model = models[i]
    sub = fig.add_subplot(2, 3, i+1).set_title(model)
    plot = sns.heatmap(confussion_matrix, annot=True)
    plot.set_xlabel('Predicted Values')
```

```
    plot.set_ylabel('Actual Values')
```



Here we can see again that Logistic Regression gives the best results - only 10+21=31 missclassified samples and I will try to improve this score now

0.0.9 6. Results and analysis

[245]:
```
# To explore logistic regression in more details I would use statsmodel library␣
↪instead
import statsmodels.api as sm


tmp = {'Age_at_diagnosis': list(X_train['Age_at_diagnosis']),
                   'Gender': list(X_train['Gender']),
                   'Race': list(X_train['Race']),
                   'grade': list(y_train)}



for gene_name in gene_names:
    tmp[gene_name] = list(X_train[gene_name])

df = pd.DataFrame(tmp)


#define response variable
```

```python
y = df['grade']

#define predictor variables
x = df[['Age_at_diagnosis', 'Gender', 'Race'] + gene_names]

#add constant to predictor variables
x = sm.add_constant(x)

#fit regression model
model = sm.OLS(y, x).fit()

#view summary of model fit
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  grade   R-squared:                       0.563
Model:                            OLS   Adj. R-squared:                  0.545
Method:                 Least Squares   F-statistic:                     31.52
Date:                Sun, 30 Jul 2023   Prob (F-statistic):           8.98e-86
Time:                        17:34:52   Log-Likelihood:                -172.06
No. Observations:                 587   AIC:                             392.1
Df Residuals:                     563   BIC:                             497.1
Df Model:                          23
Covariance Type:            nonrobust
================================================================================
====
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
const                0.4633      0.076      6.135      0.000       0.315
0.612
Age_at_diagnosis  1.229e-05   3.09e-06      3.982      0.000    6.23e-06
1.83e-05
Gender              -0.0297      0.028     -1.047      0.296      -0.085
0.026
Race                 0.0264      0.026      1.028      0.305      -0.024
0.077
IDH1                -0.6137      0.048    -12.788      0.000      -0.708
-0.519
TP53                 0.1087      0.038      2.886      0.004       0.035
0.183
ATRX                -0.0373      0.041     -0.904      0.366      -0.118
0.044
PTEN                 0.0820      0.041      1.979      0.048       0.001
0.163
EGFR                -0.0681      0.044     -1.532      0.126      -0.156
```

```
0.019
CIC               -0.0185      0.054     -0.340      0.734      -0.125
0.088
MUC16              0.0567      0.043      1.304      0.193      -0.029
0.142
PIK3CA             0.0075      0.053      0.141      0.888      -0.097
0.112
NF1               -0.1694      0.053     -3.205      0.001      -0.273
-0.066
PIK3R1             0.1138      0.062      1.834      0.067      -0.008
0.236
FUBP1             -0.0255      0.070     -0.367      0.713      -0.162
0.111
RB1                0.0046      0.070      0.066      0.948      -0.132
0.141
NOTCH1            -0.1077      0.068     -1.577      0.115      -0.242
0.026
BCOR              -0.0240      0.077     -0.311      0.756      -0.176
0.128
CSMD3              0.0440      0.083      0.531      0.595      -0.119
0.206
SMARCA4           -0.0911      0.077     -1.190      0.235      -0.241
0.059
GRIN2A             0.2455      0.080      3.067      0.002       0.088
0.403
IDH2              -0.5501      0.090     -6.091      0.000      -0.727
-0.373
FAT4               0.0357      0.081      0.439      0.661      -0.124
0.196
PDGFRA             0.0702      0.081      0.870      0.385      -0.088
0.229
==============================================================================
Omnibus:                       30.631   Durbin-Watson:                   2.031
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               65.358
Skew:                          -0.292   Prob(JB):                     6.42e-15
Kurtosis:                       4.527   Cond. No.                     1.47e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.47e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[246]:

```python
# Next let's remove predictors that don't have statisticaly significant p-value
# (0.05) and now we only have 8 predictors
```

```python
updated_columns = ['Age_at_diagnosis', 'IDH1', 'TP53', 'PTEN', 'EGFR', 'NF1',␣
 ↪'GRIN2A', 'IDH2']

#define predictor variables
x = df[updated_columns]

#add constant to predictor variables
x = sm.add_constant(x)

#fit regression model
model = sm.OLS(y, x).fit()

#view summary of model fit
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  grade   R-squared:                       0.551
Model:                            OLS   Adj. R-squared:                  0.545
Method:                 Least Squares   F-statistic:                     88.76
Date:                Sun, 30 Jul 2023   Prob (F-statistic):           1.86e-95
Time:                        17:34:52   Log-Likelihood:                -179.77
No. Observations:                 587   AIC:                             377.5
Df Residuals:                     578   BIC:                             416.9
Df Model:                           8
Covariance Type:            nonrobust
================================================================================
====
                       coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----
const                0.4821      0.073      6.590      0.000       0.338
0.626
Age_at_diagnosis  1.235e-05   3.03e-06      4.078      0.000     6.4e-06
1.83e-05
IDH1                -0.6677      0.039    -16.961      0.000      -0.745
-0.590
TP53                 0.1040      0.030      3.442      0.001       0.045
0.163
PTEN                 0.0737      0.041      1.809      0.071      -0.006
0.154
EGFR                -0.0675      0.044     -1.545      0.123      -0.153
0.018
NF1                 -0.1876      0.052     -3.611      0.000      -0.290
-0.086
GRIN2A               0.2558      0.078      3.276      0.001       0.102
0.409
```

```
IDH2                  -0.5792      0.087     -6.644      0.000      -0.750
-0.408
==============================================================================
Omnibus:                       30.076   Durbin-Watson:                   2.024
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               71.190
Skew:                          -0.242   Prob(JB):                     3.48e-16
Kurtosis:                       4.636   Cond. No.                     1.38e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.38e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[247]:

```
# now let's get updated accuracy, auc and confussion matrix for a new model
X_train_updated = X_train.drop(['Gender', 'Race', 'ATRX', 'CIC', 'MUC16',␣
 ↪'PIK3CA', 'PIK3R1', 'FUBP1', 'RB1', 'NOTCH1',
                                'BCOR', 'CSMD3', 'SMARCA4', 'FAT4', 'PDGFRA'],␣
 ↪axis=1)
X_test_updated = X_test.drop(['Gender', 'Race', 'ATRX', 'CIC', 'MUC16',␣
 ↪'PIK3CA', 'PIK3R1', 'FUBP1', 'RB1', 'NOTCH1',
                                'BCOR', 'CSMD3', 'SMARCA4', 'FAT4', 'PDGFRA'],␣
 ↪axis=1)

model = LogisticRegression(solver='liblinear')
model.fit(X_train_updated, y_train_transformed)
y_pred = model.predict(X_test_updated)
accuracy_list.append(metrics.accuracy_score(y_test_transformed, y_pred))
fpr, tpr, _tresholds = metrics.roc_curve(y_test_transformed, y_pred)
auc_list.append(round(metrics.auc(fpr, tpr), 2))
cm_list.append(confusion_matrix(y_test_transformed, y_pred))
```

[248]:

```
# also let's try to use different penalties instead, here will use l2 penalty
model = LogisticRegression(solver='liblinear', penalty='l2')
model.fit(X_train, y_train_transformed)
y_pred = model.predict(X_test)
accuracy_list.append(metrics.accuracy_score(y_test_transformed, y_pred))
fpr, tpr, _tresholds = metrics.roc_curve(y_test_transformed, y_pred)
auc_list.append(round(metrics.auc(fpr, tpr), 2))
cm_list.append(confusion_matrix(y_test_transformed, y_pred))
```

[249]:

```
# model for l1 penalty
model = LogisticRegression(solver='liblinear', penalty='l1')
model.fit(X_train, y_train_transformed)
y_pred = model.predict(X_test)
accuracy_list.append(metrics.accuracy_score(y_test_transformed, y_pred))
```

```python
fpr, tpr, _tresholds = metrics.roc_curve(y_test_transformed, y_pred)
auc_list.append(round(metrics.auc(fpr, tpr), 2))
cm_list.append(confusion_matrix(y_test_transformed, y_pred))
```

[ ]:

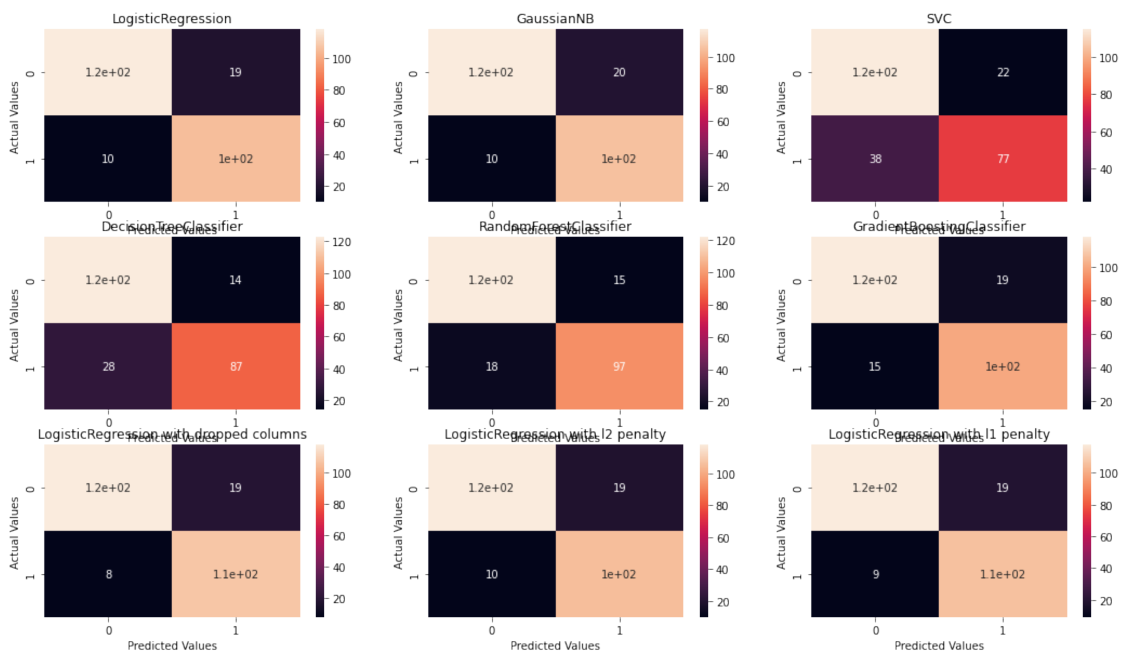
```python

```

0.0.10 7. Discussion and Conclusion

[250]:

```python
fig = plt.figure(figsize=(18, 10))
models.append('LogisticRegression with dropped columns')
models.append('LogisticRegression with l2 penalty')
models.append('LogisticRegression with l1 penalty')
for i in range(len(cm_list)):
    confussion_matrix = cm_list[i]
    model = models[i]
    sub = fig.add_subplot(3, 3, i+1).set_title(model)
    plot = sns.heatmap(confussion_matrix, annot=True)
    plot.set_xlabel('Predicted Values')
    plot.set_ylabel('Actual Values')
```



[251]:

```python
results = pd.DataFrame({'Model': models, 'Accuracy': accuracy_list, 'AUC':␣
↪auc_list})
results
```

[251]:

	Model	Accuracy	AUC
0	LogisticRegression	0.884921	0.89

1	GaussianNB	0.880952	0.88
2	SVC	0.761905	0.75
3	DecisionTreeClassifier	0.833333	0.83
4	RandomForestClassifier	0.869048	0.87
5	GradientBoostingClassifier	0.865079	0.87
6	LogisticRegression with dropped columns	0.892857	0.90
7	LogisticRegression with l2 penalty	0.884921	0.89
8	LogisticRegression with l1 penalty	0.888889	0.89

0.0.11 Summary:

I found that only subset of predictors are statistically significant: only 1 clinical predictor - Age_at_diagnosis (so Race and Gender don't affect the type of cancer), and only 7 genes out of 20 - 'IDH1', 'TP53', 'PTEN', 'EGFR', 'NF1', 'GRIN2A', 'IDH2'

These results seem to be logical as with age patients can develop cancer and we saw this in the boxplot above as well. And the same for genes: most of the genes were in mutated or non_mutated state so logically this information didn't add any value to the classification. Considering plot, I think IDH1 gene can have the highest correlation with cancer grade.

Using only statistically significant predictor I was able to improve accuracy and auc of the model (from 0.88 to 0.89 and from 0.89 to 0.9 respectively). That might not be that big difference but what is more important that with updated set of predictors it will be easier in the future to collect more data and get better results

GitHub Repo for this project can be found here:

https://github.com/also9275/dtsa_5509_final_project

[ ]:

```
```