HIGH SCHOOL EDITION
Journal of Student Research
Volume 13 Issue 3 (2024)

GENIE: Genetic Evaluation and Naive Inference for Early Diagnosis of Fabry Disease

Ishaan Ghosh[1] and Srivarun Kankanala[1]

[1]Edison Academy Magnet School, USA

ABSTRACT

Fabry Disease is a rare lysosomal disorder that reduces the body's ability to decompose glycosphingolipids that naturally accumulate in lysosomes. Specifically, the disease involves mutations in the galactosidase alpha (GLA) gene, preventing adequate production of the enzyme α-galactosidase (α-Gal). This enzyme is responsible for the breakdown of glycosphingolipids, which if not metabolized, can harm the involuntary functions of the nervous and cardiovascular systems, eyes, and kidneys. To date, the only known Fabry Disease treatment is 1-deoxygalactonojirimycin, otherwise known as oral migalastat. Fabry Disease typically falls under two categories: Classic and Late-Onset. The former develops during childhood or adolescence, while the latter is not evident until early adulthood. Current diagnosis methods are tedious and time-consuming, and the disease may spread tremendously before being identified and treated. To address this inefficiency, two machine-learning classifiers were developed: one Linear model and one Naive Bayes model. Given a mutated α-GLA nucleotide sequence, each model was to determine the mutation's Fabry variant and its amenability to oral migalastat. The Linear classifier achieved an accuracy of 59%, while the Naive Bayes classifier reached an accuracy of 96%. Thus, a correlation was established between the independent features of a Fabry-affected genotype and the patient's phenotype—an observation that will tremendously improve Fabry Disease diagnosis and treatment.

Introduction

Fabry Disease is a rare lysosomal X-linked disorder attributed to a reduction in the body's ability to decompose lipids naturally accumulating in lysosomes. Specifically, the disease involves a mutation in the galactosidase α (GLA) gene (located on loci Xq 22.1), preventing sufficient production of the enzyme α-galactosidase A (α-Gal A). This enzyme is responsible for the breakdown of glycosphingolipids, which, when deposited in the lysosomes in excessive amounts, can harm the involuntary functions of the nervous and cardiovascular systems, eyes, and kidneys (3).

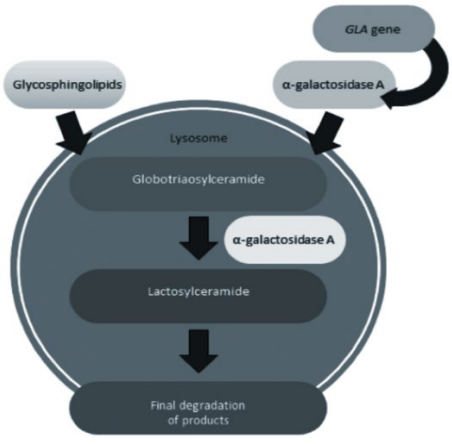

Figure 1. Lysosomal operation in unaffected cell. Figure reproduced from Amodio et al (1).

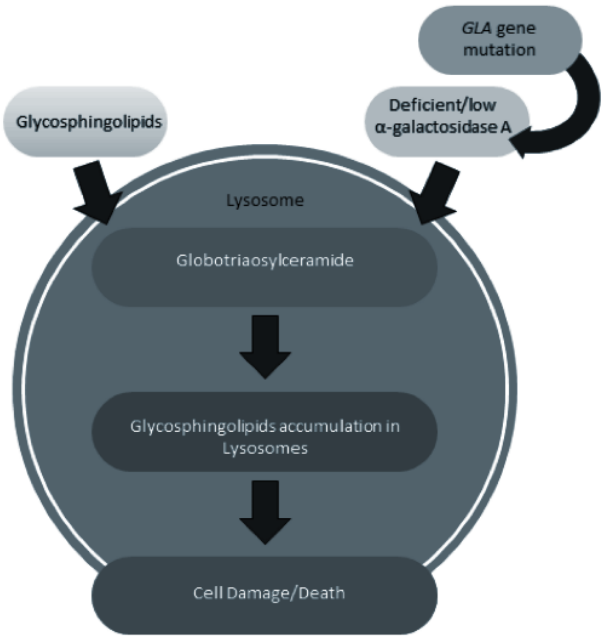

Figure 2. Lysosomal operation in Fabry-affected cell. Figure reproduced from Amodio et al (1).

Over 1,200 unique nucleotide sequences have been identified as mutations associated with the GLA gene (5). Over 60% of these mutations are missense mutations, manifesting themselves in heterogeneous phenotypes from Classic to Late-Onset. The former variant of Fabry Disease develops during childhood or adolescence, while the latter is not evident until early adulthood.

Current Diagnosis

To date, the only known treatment that has been developed to counter Fabry Disease is 1-deoxygalactonojirimycin, AT1001, or oral migalastat, a drug developed by Amicus Therapeutics. Certain patients with α-Gal A mutations are amenable to oral migalastat, showing increased catalytic activity of α-Gal A after treatment (4). This effect is most commonly found among missense mutations; mutations in the GLA gene that severely affect the protein structure (e.g. frameshift mutations, large deletions, and insertions) were not amenable to α-Gal A (3, 4).

Diagnosis of Fabry Disease typically involves an enzyme assay, looking for decreased α-Gal activity using the biomarker plasma lyso-Gb3 (7). Furthermore, for pregnant women, an amniocentesis test, which analyzes a fetus's genome for mutations in the α-GLA gene can be used. These methods, however, are tedious and time-consuming and yield worse results for women than men, especially when analyzing for left ventricular hypertrophy (4); the prognosis may worsen tremendously before being identified and treated. It should be noted that it takes 15 years upon the onset of symptoms to diagnose this disease (3). A neural network was thus implemented to address the inefficiency of diagnosis.

Data Compilation

Genetic Data

Mutations that cause Fabry Disease, although they have comparable effects, appear in the human genome in various forms. The majority of these are point substitutions, although many large-scale insertions, deletions, and complex mutations also cause Fabry Disease. Upon their discovery, most mutations are uploaded to the Human Gene Mutation Database, the Shire Human Genetic Therapies Fabry Outcome Survey registry, and other public documentation.

Creating Dataset

The dataset used for this study is based on two data compilations from Amicus Therapeutics, the developers of migalastat, one of the only known modern cures to Fabry Disease. The Supplemental Appendix list provides labels of numerous genetic mutations, along with their corresponding Fabry Disease types (Classic or Late-Onset). The Validation of Pharmacogenetics Supplement, similarly, reports several genetic mutations and their amenability to migalastat.

By combining data from these tables, one large dataset of 1,264 mutations was constructed. The data is divided into four classes of Fabry mutations: Classic and Not Amenable (Class 1), Classic and Amenable (Class 2), Late-Onset and Amenable (Class 3), and Late Onset and Not Amenable (Class 4). Each data point is defined by a full GLA gene sequence of 1,290 nucleotides; these mutated sequences are generated by a Python script, which replaces the codons specified in the mutation name for each mutation (A31V changes Codon 31 from Alanine to Valine). To balance the dataset, entries were removed until all classes had an equal number of data points (80 per class, 320 total). This data is used to train predictive machine-learning models, seeking to classify a genetic mutation by its phenotype and amenability.

```
def mutate_codon_by_letter(
    dna_sequence, codon_position,
    new_codon_letter):

  codon_dict = {
      'A': ['GCT', 'GCC', 'GCA', 'GCG'
  ],
      'R': ['CGT', 'CGC', 'CGA', 'CGG'
  , 'AGA', 'AGG'],
      ... # codons for amino acids
      'V': ['GTT', 'GTC', 'GTA', 'GTG'
  ],
      '_': ['TAA', 'TAG', 'TGA'],  #
  stop codons
  }

  possible_codons = codon_dict.get(
   new_codon_letter.upper())

  mutated_sequences = []
  for codon in possible_codons:
      mutated_sequence = dna_sequence
   [:3 * (codon_position - 1)] + codon
    + dna_sequence[3 * codon_position
   :]
      mutated_sequences.append(
   mutated_sequence)

  return mutated_sequences
```

Figure 3. Python script for mutated sequence generation.

Results

Linear Regression Model

The first model attempting to establish a connection between a Fabry mutation, and its manifestation is a Linear Regression model, which assumes a linear relationship between these two variables. To derive features from each data point, a preprocessing script is run on the dataset, extracting seven characteristics for each mutation. A score is assigned to each nucleotide base (A = 10, G = 20, T = 30, C = 40), allowing these features to be analyzed numerically. Features include:

- Type of mutation
- Average score of nucleotides removed
- Average score of nucleotides added
- Location of mutation
- Number of transitions
- Number of transversions
- Length of mutation

The model's architecture consists of seven input features (one for each characteristic) and 12 hidden layers, which start at 110 nodes and narrowing to 5 nodes. Through these hidden layers, the relative significance, or weight, of each input feature is determined. The final output layer consists of four nodes, one for each of the class predictions. Each layer is accompanied by a rectified linear unit activation function, which alters the output to fit the desired range of positive numbers.

```
class Model(nn.Module):
  def __init__(self, inputFeatures =
   7, hiddenLayer1 = 110, hiddenLayer2
    = 100, hiddenLayer3 = 90,
   hiddenLayer4 = 80,
               hiddenLayer5 = 70,
   hiddenLayer6 = 60, hiddenLayer7 =
   50, hiddenLayer8 = 40,
               hiddenLayer9 = 30,
   hiddenLayer10 = 20, hiddenLayer11 =
    10, hiddenLayer12 = 5, output = 4)
   :
    super().__init__()
    self.fc1 = nn.Linear(inputFeatures
   , hiddenLayer1)
    self.fc2 = nn.Linear(hiddenLayer1,
    hiddenLayer2)
    # self.fc3 to self.fc10
    self.fc11 = nn.Linear(
   hiddenLayer10, hiddenLayer11)
    self.fc12 = nn.Linear(
   hiddenLayer11, hiddenLayer12)
    self.out = nn.Linear(hiddenLayer10
   , output)
  def forward(self, x):
    x = F.relu(self.fc1(x))
    x = F.relu(self.fc2(x))
    # F.relu(self.fc3(x)) to F.relu(
   self.fc10(x))
    x = F.relu(self.fc11(x))
    x = F.relu(self.fc12(x))
    x = self.out(x)

    return x
```

Figure 4. Linear model architecture.

Using an 80/20 training/testing split, a random set of mutations is chosen to train the model. Employing the Adam optimizer, the model minimizes loss (becomes more "fit") by adjusting the relative weight of each feature. By iterating through 200 epochs at a learning rate of 0.001, the model approaches a global minima loss value, reaching its maximum accuracy.

The Adam optimizer employs the Stochastic Gradient Descent Algorithm to minimize the loss function. Consider a function $f(x)$ with 7 unique input features that is said to predict the class of the genetic phenotype.

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_7 x_7 = \sum_{j=0}^{7} \theta_j x_j$$

Let m be the number of training samples. The goal of the Adam optimizer is to minimize the cost function j represented as a least-squares difference of the model and the true value. Let θ be a parameter such that $f_\theta(x) \approx y$.

$$\mathrm{j}(\theta) = \frac{1}{2}\alpha \sum_{\mathrm{i}=1}^{\mathrm{m}} (f_\theta(x_i) - y_i)^2$$

Minimizing this uses stochastic gradient descent by the following formula. Note that α represents the learning rate, which in this model is 0.001. The $\coloneqq$ represents assignment, rather than equality.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} j(\theta) \quad (j = 0, 1, \ldots 7)$$

The partial derivative for each feature is simply $f(x)$, as no other terms in $f(x)$ depend on that specific θ_j and thus the equation can be simplified to:

$$\theta_j := \theta_j - \alpha \left[\sum_{i=1}^{m} \frac{1}{2} (f_\theta(x_i) - y_i) \frac{2\partial}{\partial \theta_j} \left(\sum_{j=0}^{7} \theta_j x_j \right) \right]$$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (f_\theta(x_i) - y_i) x_j$$

The Adam Optimizer builds on Gradient Descent, by helping the model converge to the minima faster but works on the same fundamental principles.

```
X = my_df.drop('Disease Type', axis =
    1)
y = my_df['Disease Type']
X = X.values
y = y.values

X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size =
    0.2, random_state = 100)

X_train = torch.FloatTensor(X_train)
X_test = torch.FloatTensor(X_test)
y_train = torch.LongTensor(y_train)
y_test = torch.LongTensor(y_test)

criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.
    parameters(), lr = 0.001)
```

Figure 5. Linear model training and testing.

This algorithm, despite having optimized hyperparameters, has achieved an accuracy of no greater than 59%. This suggests that, given a 1,290-nucleotide sequence, the relationship between a genetic mutation and its Fabry behavior cannot be determined via a linear model.

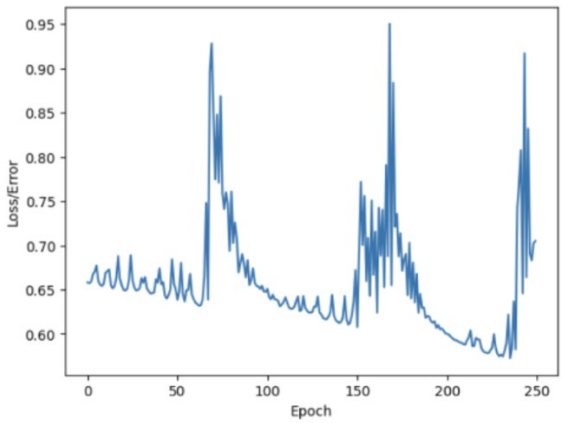

Figure 6. Loss function of linear model.

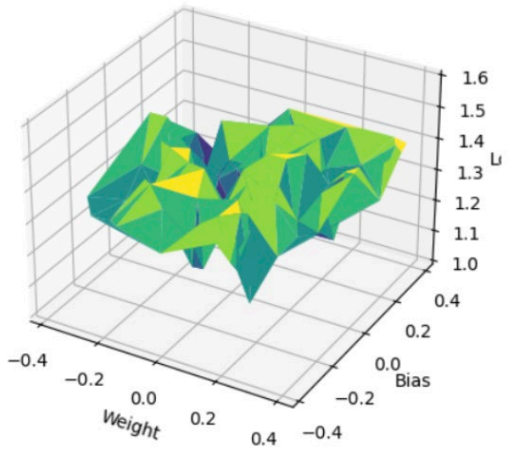

Figure 7. Gradient Descent function of linear model.

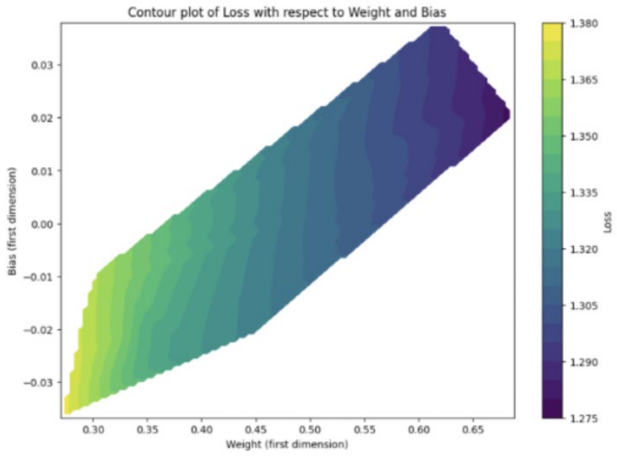

Figure 8. Contour plot of linear model.

The visualizations of the linear model further prove that it is unfit to classify nucleotide sequences based on their phenotype and amenability to oral migalastat. The loss function of this model (Figure 6) supports the low accuracy, as the loss of the function is high with a greater number of training epochs. The 3D representation of the loss, as shown in the gradient descent (Figure 7), is convoluted and unclear. The lack of a clear global minima displays that the linear model is not a good representation of this classifier.

The contour plot (Figure 8) of the loss function shows that as the loss is decreased, the bias of the model is increased, adding to the evidence that a linear model does not fit the relationship between the nucleotide sequence and phenotypic variant and amenability to oral migalastat to a Fabry patient. Thus, a deeper, more intricate analysis of the data is necessary to determine the expression of a specific genetic mutation.

Naive Bayes Model

To address this concern, a probabilistic approach was taken to connect Fabry Disease with its phenotypic variant and amenability. Under the assumption of the Bayes' Theorem, the Multinomial Naive Bayes model naively assumes that all features in the input data are independent of each other. With this assumption, it calculates the posterior probability of a genetic sequence belonging to a certain class given its features and sorts it into a class based on the highest posterior probability.

Let dataset D $= \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ drawn from a population $P(y, x)$ where y is the classes to be organized into and x is the test data and we are given n samples are in **D** for testing. Since we do not know the distribution of the training data, we can approximate it with some parameter θ that we understand. To make a prediction for a specific test point, we can average out every possible model over all parameters.

$$P(y|x) = \int_0^{\theta} P(y|\theta)\ P(\theta|\mathbf{D})\, d\theta$$

$P(y|\theta)$ measures the probability of a class appearing with a given parameter and $P(\theta|\mathbf{D})$ represents how applicable the parameter is to the dataset. However, since getting these exact probabilities are nearly impossible for each state, we must make the Naive Bayes Assumption, which states that each feature is distributed independently among the class (6).

$$P(x|y) = \prod_{\alpha=1}^{n} P(x_\alpha|y)$$

where x_α represents each input feature of the testpoint. Now, using Bayes theorem as well as the Naive Bayes' Assumption, the probability of a certain class given a set of n features, or the posterior probability was calculated for each class and compared. The model sorted the sample data points into the class with the highest posterior probability (6). Calling the model $f(x)$:

$$P(y|x) = \frac{P(y)P(x|y)}{P(x)}$$

$$\mathrm{f}(x) = argmax\ P(y|x)$$

$$f(x) = argmax\ \frac{P(y)P(x|y)}{P(y)}$$

$$f(x) = \underset{y}{argmax}\, P(y|x)P(y)$$

$$f(x) = \text{argmax} \prod_{\alpha=1}^{n} P(x_\alpha|y)P(y)$$

$$f(x) = \text{argmax} \sum_{\alpha=1}^{n} log\big(P(x_\alpha|y)\, P(y)\big) + log\big(P(y)\big)$$

The model began by separating the nucleotide sequence into 6-mers. These small sections of coding sequences were analyzed independently to examine subtle features from the larger sequence. These sequences were vectorized, allowing the model to extract features from the data.

```
# k-mers function
def getKmers(sequence, size=6):
    return [sequence[x:x+size].lower()
    for x in range(len(sequence) -
   size + 1)]

# finding sequences
sequences['words'] = sequences.apply(
   lambda x: getKmers(x['sequences']),
    axis=1)
sequences = sequences.drop('sequences'
   , axis=1)
sequences.head()

# getting classes
sequence_texts = list(sequences['words
   '])
for item in range(len(sequence_texts))
   :
    sequence_texts[item] = ' '.join(
   sequence_texts[item])
y_data = sequences.iloc[:, 0].values

# vectorization of nucleotide
   sequences
from sklearn.feature_extraction.text
   import CountVectorizer
cv = CountVectorizer(ngram_range=(8,8)
   )
X = cv.fit_transform(sequence_texts)
```

Figure 9. Naive Bayes model data vectorization.

85% of the total data was then used to train the total data with a learning rate of 0.01, while the other 15% was used for testing.

```
# splitting data
from sklearn.model_selection import
   train_test_split
X_train, X_test, y_train, y_test =
   train_test_split(X, y_data,

                 test_size = 0.15,

                 random_state = 42)

# defining classifier
from sklearn.naive_bayes import
   MultinomialNB
classifier = MultinomialNB(alpha=.01)
classifier.fit(X_train, y_train)

# making predictions
y_pred = classifier.predict(X_test)
```

Figure 10. Naive Bayes model training and testing.

The model showed an accuracy of 95.8%, with its confusion matrix displaying an incorrect prediction for only 2 of the 48 test points. The model's precision, at 95.8%, measures the proportion of its classifications that were accurate. The model's recall, also at 95.8% measures the proportion of true classifications that it was able to make. These values, overall, suggest that the Naive Bayes model is able to classify a genetic Fabry mutation by its phenotype and amenability with an accuracy of 95.8%.

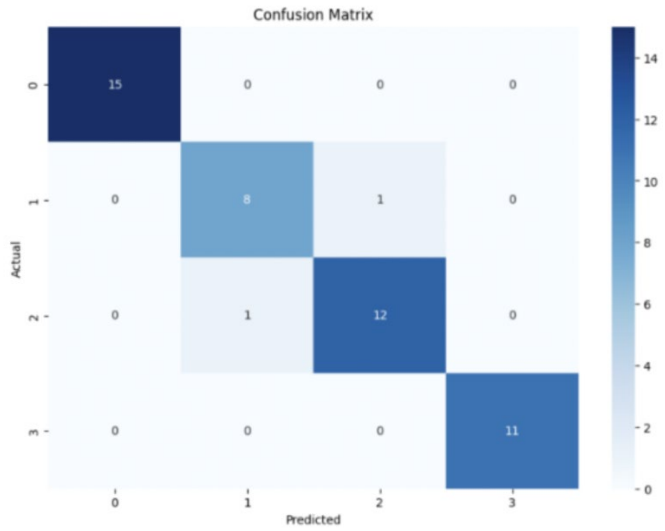

Figure 11. Confusion matrix of Naive Bayes model.

Reinforcement Analysis

To gain a deeper understanding of the model's predictive power for individual characteristics, the dataset was altered to exclude certain classes, subclasses, and features. For example, by muting all Late-Onset genetic mutations, the model's ability to discern between amenable and non-amenable Classic mutations was determined. Similarly, by muting all amenable mutations, the model's ability to determine Classic and Late-Onset non-amenable mutations was determined.

Table 1. Results of Reinforcement Analyses

Alteration	Dataset Size	Accuracy (%)
Full Dataset	320	95.8
Only Classic	160	95.8
Only Late-Onset	160	100
Only Amenable	160	91.7
Only Not Amenable	160	91.7
Class 0 Muted	240	94.4
Class 1 Muted	240	100
Class 2 Muted	240	97.2
Class 3 Muted	240	88.9

The results from these data-alteration experiments show that, when given a genetic sequence, the model's ability to determine a Fabry variant is comparable to its ability to determine migalastat-amenability. The most inaccurate trial is Class 3 (Late-Onset, Non-Amenable) was removed, resulting in an accuracy below 90%. This suggests that, using 80% of the given data for training, the model is proficient in identifying Class 3 mutations. The consistency in accuracy among various experiments supports that the model effectively discerns between each of the four classes with relatively equal strength.

Conclusion

Overall, the Naive Bayes Classifier was significantly more effective than the Linear Classifier at diagnosing the Fabry-affected sequences. Due to its high accuracy, this model provides evidence that the genetic features may be analyzed to predict Fabry expression and amenability to oral migalastat in a person.

By employing such a model, the time needed for the diagnosis of Fabry Disease can be reduced, allowing for more immediate treatments to be administered. Furthermore, medical examiners will be aware of the effectiveness of oral migalastat on a particular patient and may administer treatment methods accordingly. This will, overall, decrease the mortality rates of the disease, especially in the more heavily affected male population. The success of this model also gives valuable insight into the biological workings of not only Fabry Disease, but all genetic disorders; certain mutation features can be linked with certain disease behaviors. A deeper understanding of gene expression will allow for more time-efficient diagnosis and treatment, ultimately improving the efficiency of healthcare systems.

To improve the universality of this model, the number of output classifications may be expanded to account for amenability to other Fabry treatment drugs. With such developments, the Naive Bayes classifier will grow to be an effective diagnosis tool and therapeutic examiner for Fabry Disease, and the knowledge gleaned from its development may set the foundation for revolutionary optimizations in the healthcare industry.

For future steps, it is crucial that one delve into the confirmational shape of each of these protein sequences and analyze how the mutated GLA gene sequence affects the enzyme's ability to fold into its final confirmation shape.

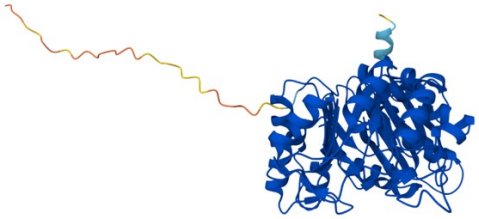

Figure 12. Appropriate confirmational shape of α-Gal A. Figure reproduced from Alphafold (2).

Acknowledgments

I would like to thank my advisor for the valuable insight provided to me on this topic.

References

1. Amodio, F., Caiazza, M., Monda, E., Rubino, M., Capodicasa, L., Chiosi, F., Simonelli, V., Dongiglio, F., Fimiani, F., Pepe, N., Chimenti, C., Calabrò, P., & Limongelli, G. (2022, October 12). An Overview of Molecular Mechanisms in Fabry Disease. MDPI. https://doi.org/10.3390/biom12101460

2. Benjamin, E. R., Della Valle, M. C., Wu, X., Katz, E., Pruthi, F., Bond, S., Brofin, B., Williams, H., Yu, J., Bichet, D. G., Germain, D. P., Giugliani, R., Hughes, D., Schiffmann, R., Wilcox, W. R., Desnick, R. J., Kirk, J., Barth, J., Barlow, C., . . . Lockhart, D. J. (2016, September 22). The validation of pharmacogenetics for the identification of Fabry patients to be treated with migalastat. PubMed. https://pubmed.ncbi.nlm.nih.gov/27657681/

3. Bokhari, S. R. A. (2023, July 4). Fabry Disease. StatPearls. https://www.ncbi.nlm.nih.gov/books/NBK435996/.

4. EMBL-EBI. (n.d.). Alpha-galactosidase A. Alphafold Protein Structure Database. https://alphafold.ebi.ac.uk/entry/P06280

5. Filoni, C., Caciotti, A., Carraresi, L., Cavicchi, C., Parini, R., Antuzzi, D., Zampetti, A., Feriozzi, S., Poisetti, P., Garman, S. C., Guerrini, R., Zammarchi, E., Donati, M. A., & Morrone, A. (2009, November 24). Functional studies of new GLA gene mutations leading to conformational Fabry disease. PubMed. https://pubmed.ncbi.nlm.nih.gov/19941952/

6. Vardarli, I., Rischpler, C., Herrmann, K., & Weidemann, F. (2020, June 22). Diagnosis and screening of patients with Fabry disease. PubMed. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7319521/

7. Weinberger, K. (2018, July 11). Machine Learning Lecture 10 “Naive Bayes continued” -Cornell CS4780 SP17. YouTube https://www.youtube.com/watch?v=rqB0XWoMreU&list=PLl8OlHZGYOQ7bkVbuRthEsaLr7bONzbXS&index=10