Brooke Hanson

Final Report:
Genetics and Genome Analysis and Modeling

Problem Statement:

With the exponential increase in world population, there is a correlating increase in genetic mutations that lead to a range of genetic defects in a range from mild to fatal. Hereditary illnesses are becoming more common due to a lack of understanding about the need for genetic testing. Often kids die as a result of these illnesses, thus genetic testing is critical for individuals interested in becoming parents to understand risks of genetic disorders that are present in their own DNA and that can be passed on. This therefore leads to the question, can these hereditary illnesses be predicted?

Using a dataset from Kaggle.com, which included demographic data such as the patient's age, mother's age, father's age, as well as health data such as blood cell count, white blood cell count, respiratory rate, and heart rate. I used this data to create multiple neural networks and tune them with learning rate schedulers to determine if these conditions can be predicted, and with what accuracy they can be predicted.

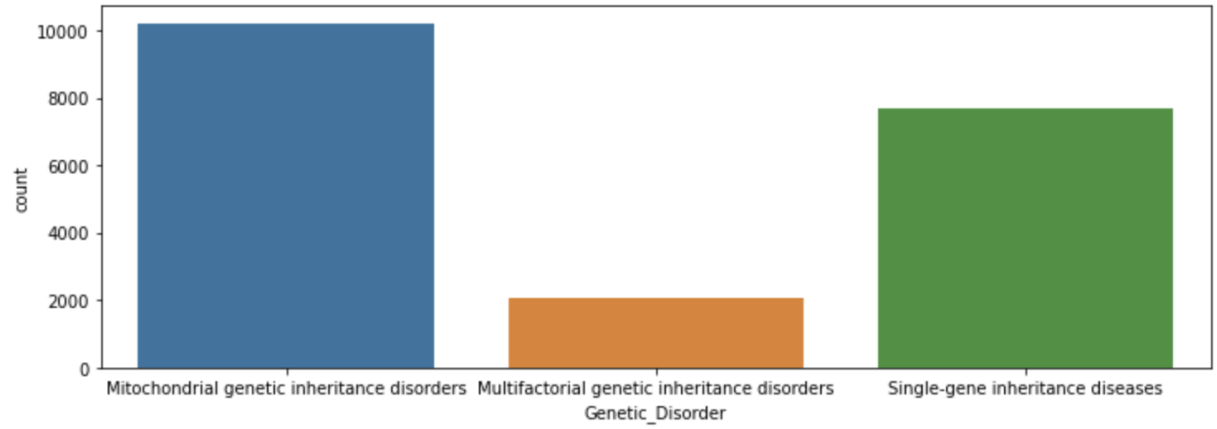
Data Wrangling and Cleaning:

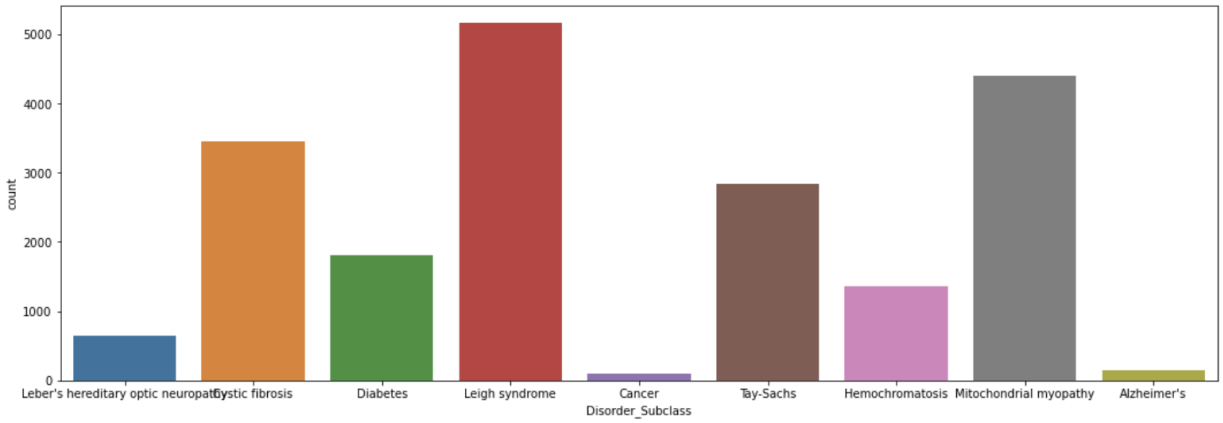
For this dataset there were many columns that were unnecessary for a fruitful analysis such as patient name, family name, patient id, location of institute, institute name, father's name, and place of birth. As these variables were not pertinent to a patient's health or hereditary they were unnecessary for analysis. There were also a plethora of missing values in the data. I chose to fill the numeric missing values with the median, and for the categorical variables, I filled the majority of categorical variables with a backfill method to keep as much continuity in the data as possible. There were also missing values in the Genetic Disorder and Subclass Disorder columns and since those are the target variables, I dropped the NA values in those columns in order to maintain the data integrity. The final data that I analyzed for the Genetic Disorder neural network consisted of 18,047 data points and 29 columns with Genetic Disorder being the target variable. The final data I analyzed for the Subclass Disorder neural network consisted of 18,047 data points and 30 columns with Subclass Disorder being the target variable.

Exploratory Data Analysis:

For the Exploratory Data Analysis, I began by plotting the age data of the patient, mother and father to understand the spread of the age data. I then was interested in the

count of each genetic disorder as well the count of each subclass disorder. The results of such are depicted below:



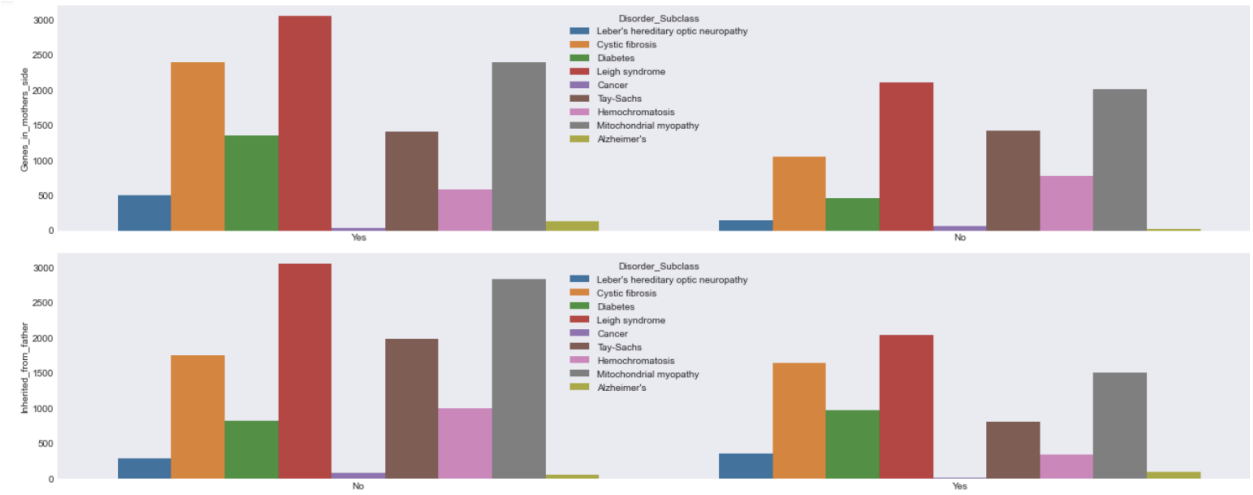


From these bar plots, it is clear there are some Genetic Disorders and Subclass Disorders are more common than others. This may be something to keep in mind when making the neural network.

I then wanted to look into the relationship some of the variables had with the target variable, I plotted the categorical variables against the genetic disorders to see if any patterns emerged. Below is an example of such a plot. The example categorical variables I used are ‘Genes_in_mothers_side’ and ‘Inherited_from_father’. While only two plots are present, it appears that across all the categorical factors there is a consistency still seen in the occurrence of each genetic disorder. This is slightly disconcerting as it may indicate there is not a strong relationship with the variables and the genetic disorder.

Brooke Hanson

The next variables I wanted to get an idea of how the categorical variables had an effect on the subclass disorder. I used the same categorical variables 'Genes_in_mothers_side' and 'Inherited_from_father' to understand how the same categorical variables affect the different response variables.



While the general shape of the bar plot stays the same, there seems to be a difference between Alzheimer's, Cancer, and Tay-Sachs between the graphs. This may be an early indication that the Subclass Disorder may be more influenced by the variables than the genetics disorder.

Finally before moving into the modeling phase, I wanted to test independence of each of the variables in relation to the target variables. I ran chi2 tests for each of the variables and filtered out any that were not significant. Below are the significance results for the Genetic Disorder variable:

```
Genes_in_mothers_side  p value for chi2 test:  1.1467655663681908e-38
Inherited_from_father  p value for chi2 test:  3.001679792644031e-38
Maternal_gene  p value for chi2 test:  1.2927887155021174e-25
Paternal_gene  p value for chi2 test:  3.452529030466087e-25
Blood_test_result  p value for chi2 test:  0.001679257943707886
Symptom_1  p value for chi2 test:  2.247982145630757e-46
Symptom_2  p value for chi2 test:  9.215941215610407e-85
Symptom_3  p value for chi2 test:  2.999850462213498e-128
Symptom_4  p value for chi2 test:  1.0122534825925479e-142
Symptom_5  p value for chi2 test:  3.465776098290755e-218
```

I then did a similar process for the Subclass Disorder with the results below:

```
Genes_in_mothers_side  p value for chi2 test:  8.008984833300492e-167
Inherited_from_father  p value for chi2 test:  1.0456427733813996e-146
Maternal_gene  p value for chi2 test:  1.2214538777261223e-128
Paternal_gene  p value for chi2 test:  3.7713333352110493e-115
Blood_test_result  p value for chi2 test:  0.046058976788885529
Symptom_1  p value for chi2 test:  1.5267241061175745e-233
Symptom_2  p value for chi2 test:  0.0
Symptom_3  p value for chi2 test:  0.0
Symptom_4  p value for chi2 test:  0.0
Symptom_5  p value for chi2 test:  0.0
```

What is interesting to note, is that the significance associated with the variables is higher for almost all of the variables when looking at the Subclass Disorder test. This may be an early indication that the Subclass Disorder model will be more informed and perform better than the Genetic Disorder model.

In Depth Analysis and Modeling: Genetic Disorders

To begin with the model and preprocessing of the data, I again fill the necessary NA values, and ensure the data set only has the variables that are necessary for analysis. Before analysis I must standardize the numeric values in the dataset and for neural networks, the best standardization practice for neural networks is for the numeric data to be standardized between 0 and 1. To perform the standardization I used the equation (X - min(X))/(max(X) - min(X)).

The next step in the modeling process is splitting up the data into the explanatory and response variables. For the first model the response variable is the Genetic Disorder, and for the second model the response variable is Subclass Disorder and Genetic Disorder then become part of the explanatory variables. After the data has been split up into X, y, X2, and y2 datasets, I created dummy variables for the categorical values as well as dummy variables for the response variables y1 and y2.

Now it is time to move onto the development of the neural network. I begin by defining the Sequential() model and start my baseline model off with an Adam optimizer and a learning rate of 0.00001. In that model I also included Dropout layers between each dense layer, as well as BatchNormalization() between each layer as well. These additions help deal with the unbalanced categorical data, as well as ensure the model does not get overfitted to the training data. I ran the model through a KerasClassifier and used kfold = 10 to run the model multiple times and get an understanding of the average accuracy of the model.

When I ran the baseline model, and took an average of the accuracy of the model with the result below:

```
Baseline: 51.29% (5.47%)
```

This result is an early indication of how the model will perform. A neural network makes guesses about the outcome and learns from these guesses in each layer of the network. For the output layer in this model there are 2 outcomes and if the model is not learning then the accuracy would stay around 50%. Something to consider as moving forward and trying to tune the model.

The second iteration of my model I took the same initial model, but specifically fit the model to the data and used the predict() and evaluate() functions of keras to get an idea of how the training and test sets are performing with the model. Below are the results:

```
Accuracy on training data: 0.5503221154212952%
 Error on training data: 0.44967788457870483
Accuracy on test data: 0.5479224324226379%
 Error on test data: 0.45207756757736206
```

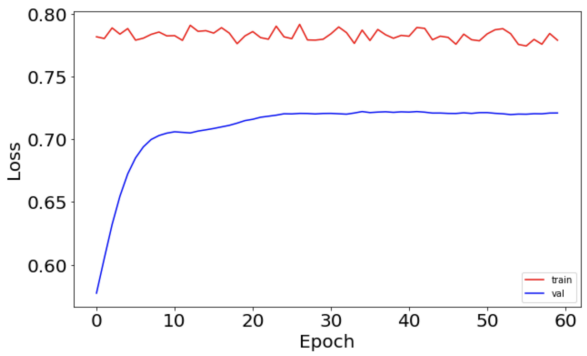
As seen from the increased accuracy scores, the fitted model has a slightly better performance and performed similarly on the training and test sets which is encouraging that the model isn't overfitted to the training data.

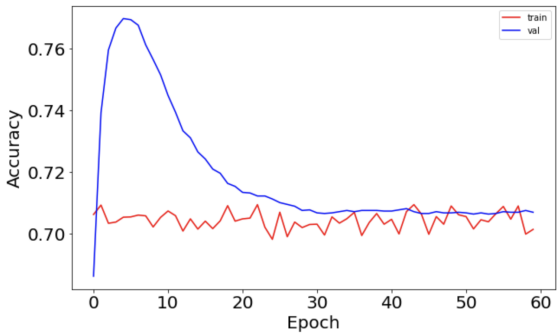
After I have created a working model I want to be able to tune the model to see if I can achieve a higher accuracy score. I will start trying to tune the model using Stochastic Gradient Descent, Gradient Descent is one of the main ways to tune a neural network. The SGD function has a built in learning rate scheduler that will decrease the learning rate as the epochs progress. The learning rate decreases according to this function: $lr=lr\times1/(1+decay*epoch)$. I then compared the previous results to the results of the SGD.

```
Accuracy on training data: 0.5084851384162903%
 Error on training data: 0.4915148615837097
Accuracy on test data: 0.5013850331306458%
 Error on test data: 0.4986149686935425
```

Brooke Hanson

The accuracy on the training and test data for the SGD model turns out to be worse than the original model. I also plotted the history of the loss and accuracy to get a visual representation of the model.

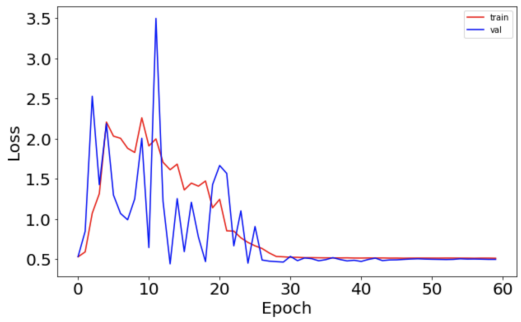


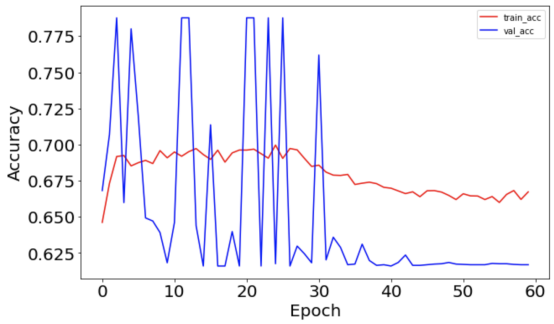


While it is encouraging that the loss is stable for the training set, it is not a sign of a good model that the loss increases on the test set. For the accuracy graph it is good that the test set accuracy is higher than the training, so the model isn't overfit on the training data, but the curve is not the ideal.

The next model I will experiment with is the exponential decay model to experiment with a different learning rate function: $lr=lr_0 \times e^{\wedge}(-kt)$. Below is the test loss and accuracy associated with the exponential decay model. It seems to be the lowest accuracy score seen so far with the model. The graphs for the model are shown below as well.

```
Test loss: 0.2490587681531906
Test accuracy: 0.3806094229221344
```





These are the plots associated with the accuracy and loss of the exponential decay model. From both plots it is clear that the test set has a lot of noise and the test accuracy recorded is the lowest seen from any of the models.

These models for hypertuning the genetic disorder model have only shown that the data to predict the genetic disorder may not be relevant enough to make a well trained model.

In Depth Analysis and Modeling: Disorder Subclass

As I have already preprocessed the data and split the data into appropriate training and test sets, it is time to make a second set of neural networks to investigate the relationship data has with the response variable of Disorder Subclass. Now it is time to move onto the development of the neural network. I again begin by defining the Sequential() model and start my baseline model off with an Adam optimizer and a learning rate of 0.00001. In that model I also included Dropout layers between each dense layer, as well as BatchNormalization() between each layer as well. These additions help deal with the unbalanced categorical data, as well as ensure the model does not get overfitted to the training data. I ran the model through a KerasClassifier and used kfold = 10 to run the model multiple times and get an understanding of the average accuracy of the model.

```
Baseline: 21.40% (3.69%)
```

This low accuracy may be related to there being 8 final classes that the model must predict to and as I stated previously the baseline model is meant to help inform about the baseline predictive power of the model. The second iteration of my model I took the same initial model, but specifically fit the model to the data and used the predict() and evaluate() functions of keras to get an idea of how the training and test sets are performing with the model. Below are the results:

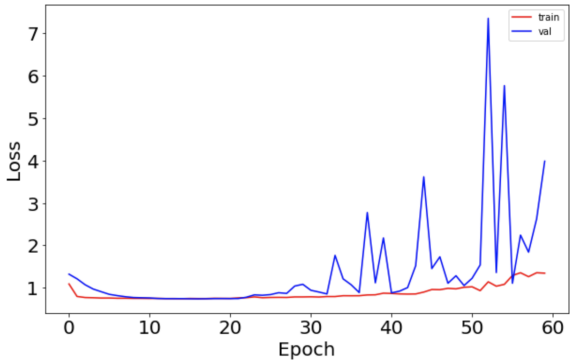
```
Accuracy on training data: 0.24416430294513702%
 Error on training data: 0.755835697054863
Accuracy on test data: 0.23490305244922638%
 Error on test data: 0.7650969475507736
```

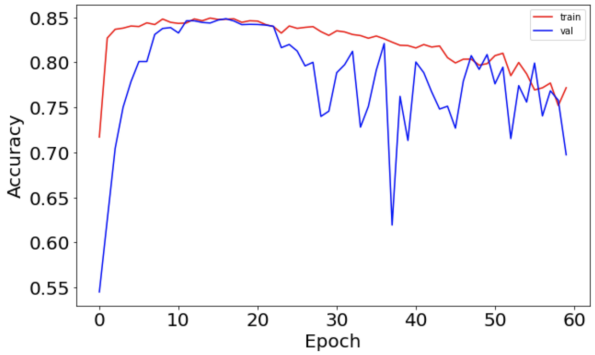
It is encouraging that there is an increase in accuracy with the training data on a model that is fit to the data, as well as the fact that the training and test accuracy are similar for the model lets me know that the model has not been overfitted. Though the low accuracy in general may be again attributed to the multiple classes.

After I have created a working model I want to again be able to tune the model to see if I can achieve a higher accuracy score. I will again start by trying to tune the model using Stochastic Gradient Descent, Gradient Descent is one of the main ways to tune a neural network. The SGD function has a built in learning rate scheduler that will decrease the learning rate as the epochs progress. The learning rate decreases according to this function: $lr=lr\times1/(1+decay*epoch)$. I then compared the previous results to the results of the SGD.

```
Test loss: 15.857719421386719
Test accuracy: 0.4861495792865753
```

From the scores seen here, there is a much higher accuracy score for this model than has been observed before, but there is also a much higher loss value than before. Therefore the model has not been fully optimized but there are signs of improvement. Below are the graphs associated with the accuracy and loss functions of the SGD model.

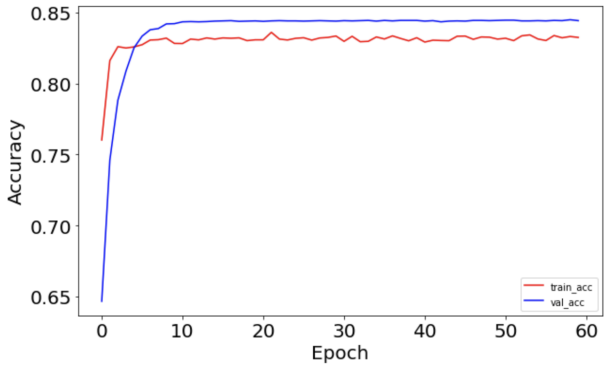


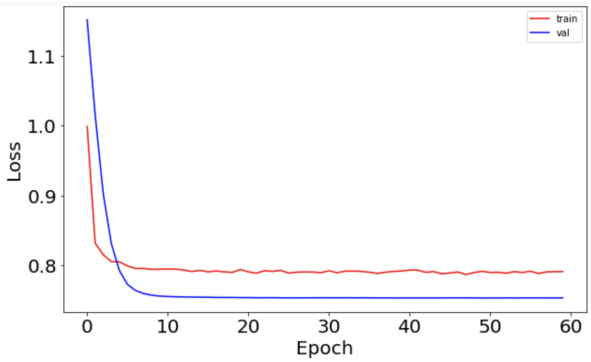


From these graphs, it is clear there are some issues with the validation set being quite noisy for both the accuracy and the loss. It is also something to note that the loss goes up as the epochs go, and the accuracy goes down in the same way. It is clear the SGD model is not the most efficient for this data.

The next model I will experiment with is the exponential decay model to experiment with a different learning rate function: $lr=lr_0 \times e^{\wedge}(-kt)$. Below is the test loss and accuracy associated with the exponential decay model. It seems to be the lowest accuracy score seen so far with the model. The graphs for the model are shown below as well.

```
Test loss: 0.5671818852424622
Test accuracy: 0.7127423882484436
```





Brooke Hanson

The model accuracy score and graphs for the exponential decay model are clearly superior to any of the previous models. The accuracy score is higher for the test set than the training set, and the loss function decreases exponentially as it should, and the test loss is lower than the training loss. All of which indicates that the best model for the Disorder Subclass data.

Takeaways and Improvements:

This investigation into neural networks and this genetic data have been very educational. The best optimizer I found for the Genetic Disorder model was the Adam optimizer, and the best optimizer found for the Disorder Subclass was the Stochastic Gradient Descent with the Scheduled Learning Rate function of exponential decay. The highest accuracy achieved with the Genetic Disorder model was 0.55 for the training set and 0.54 for the test set. The highest accuracy for the Disorder Subclass model was 0.71 for the test set.

The improvements that can be made to this model consist of data improvements. The majority of the data included is demographic data, as well as categorical data that is not as informative as true genetic data could be. If I was able to include gene sequencing data from patients as well as parents I believe that both models would be able to be trained and predict at a much higher level.