Survival Prediction using Autoencoder and AFT model

salehinnazmus672@gmail.com

January 2021

1 Introduction

The goal of this project is to predict survival times of patients using gene expression features. For the 1st part of our projcet, we are aiming to reduce the dimensionality of our gene-expression dataset which has been extracted from http://www.linkedomics.org/. The RNA-Seq data has 528 samples and 4571 features. For dimensionality reduction we will be using standard, denoisng and variational Autoencoder, which are a special type of neural network. After the feature reduction, we will use reduced feature in our clinical dataset and evaluate the effectiveness of predictions.

2 Dimensionality Reduction with Autoencoder

Autoencoder is an unsupervised artificial neural network that learns how to efficiently compress and encode data then learns how to reconstruct the data back from the reduced encoded representation to a representation that is as close to the original input as possible. For higher dimensional data, autoencoders are capable of learning a complex representation of the data (manifold) which can be used to describe observations in a lower dimensionality and correspondingly decoded into the original input space. An standard autoencoder has following parts:
Encoder: In this part, the input is reduced to a lower dimension. It's typical form looks like:

$$f_\theta(x) = s(Wx + b)$$

where f_θ is the transformation of input x to hidden layer, the parameter set is, $\theta = (W, b)$
Decoder: The resulting hidden representation y is mapped back to a reconstructed vector, $z = g_{\theta'}(y)$. The mapping $g_{\theta'}(y)$ is called the decoder.It's typical form is:

$$g_{\theta'}(y) = s(W'y + b')$$

where parameters are $\theta' = (W', b')$
Loss Function: The loss function typically used in these architectures is mean squared error, $J(x, z) = |x - z|^2$, which measures how close the reconstructed input z is to the original input x.Depending on the types of autoencoder, different types of loss functions are used.

For our project, we will basically use Denoising Autoencoder and Variational Autoencoder, which has shown promising results with dimensionality reduction tasks previosly.

Denoising Autoencoder In denoising autoencoder, the input is intentionally corrupted with noise in order to avoid overfitting and extract more meaningful information. This is done by first corrupting the initial input x into $\hat{x}$ by means of a stochastic mapping $\hat{x} \sim q_D(x|\hat{x})$. Corrupted input $\hat{x}$ is then mapped, as with the basic autoencoder, to a hidden representation

$$y = f_\theta(\hat{x}) = s(W\hat{x} + b)$$

A reconstruction of $z = g'_\theta(y)$ is done from that. Denoising autoencoders (DAE) can be stacked to form a deep network by using the hidden representation of one as input for the next autoencoder. These architectures are called stacked denoising autoencoders (SDAE)

Variational Autoencoder In Variational autoencoder, rather than building an encoder which outputs a single value to describe each latent state attribute, the encoder will output a value from the probability distribution of the latent attribute. VAE tries to maximize the probability of each X in the training set under the entire generative process

$$P(x) = \int P(x|z;\theta)P(z)\, dx$$

Here $P(z)$ is the prior on the latent distribution z, We assume that we have a set of latent vector z in a high-dimensional space Z which we can easily sample according to some probability density function (PDF) P(z) defined over Z. We want to optimize θ so that we can sample z from $P(z)$ with high probability such that $f(z;\theta)$ will be like our data. Here is how the VAE works

- Some latent representations of z are sampled from P(z)
- Based on the sample, the actual representation x is created which is a stochastic process $P(x|z)$ x is the training data and z is the latent variable.

The input is encoded as distribution over the latent space. Next a point from the latent space is sampled from that distribution. The sampled point is decoded and the reconstruction error can be computed. Finally, the reconstruction error is backpropagated through the network. the encoded distributions are chosen to be normal so that the encoder can be trained to return the mean and the covariance matrix that describe these Gaussians.

Our goal is to find good values of latent variables given the dataset which is $P(z|x)$. We can use Bayes rule to find out $P(z|x) = \frac{P(x|z)P(z)}{P(x)}$. In VAE, it is assumed that $P(z)$ can be sampled from normal distribution. So, we use $q_\lambda(z|x)$ to approximate $P(z|x)$. In this case, λ represents the parameters of normal distribution. Kullback-Leibler divergence is used to measure how well q is approximating p:

$$KLq_\lambda(z|x)||p(z|x) = E_q[log(q_\lambda(z|x))] - E_q[log(p(x,z))) + logp(x)$$

The first two terms can be written as what we call Evidence lower bound (ELBO).

$$ELBO(\lambda) = E_q[log(p(x,z))) - E_q[log(q_\lambda(z|x)$$

Our goal is to find variational parameters λ that minimizes this variance.

$$q^*_\lambda(z|x) = argmin_\lambda KL(q_\lambda(z|x)||p(z|x))$$

The evidence can be rewritten as:

$$log[p(x)] = ELBO(\lambda) + KLq_\lambda(z|x)||p(z|x)$$

We can minimize the KL-divergence, by maximizing ELBO function. The neural network will try to learn the paramters (μ and σ) of q that maximizes $P(x|z)$. The ELBO can be expressed as:

$$-ELBO = KL[q(z|x)||p(z)] - E_{Z\sim(q(z|x))}log(p(x|z))$$

Assuming independence of $x_1, x_2, ...x_j$ the KL term can be written as:

$$-ELBO = \sum_j KL[q(z_j|x_j)||p(z_j)] - E_{Z\sim(q(z_j|x_j))}log(p(x_j|z_j))$$

When we build autoencoders for dimensionality reduction these are the couple of things to consider:

- Activation Function: There are different types of non-linear activation functions widely used in autoencoders. Here is a brief summary of them: RELU: The Relu function is given as:

  $$A(x) = max(0,x)$$

  It gives an output if x is positive, and 0 otherwise. Relu is less computationally expesive because of it's 0 output as x is negative. The disadvantage is that neurons that will go into negative x region will stop responding to variations and gradient will be 0 at that region. This is called a dying Relu problem.

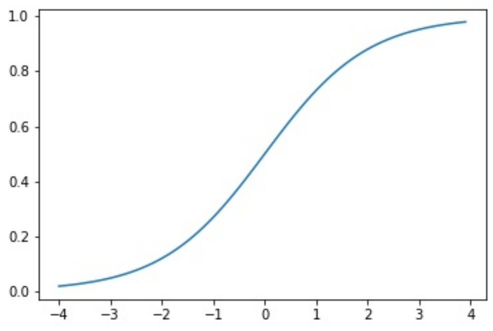


Figure 1: Sigmoid Function

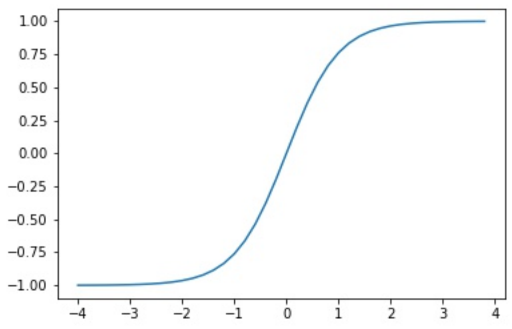


Figure 2: tanh Function

Sigmoid: The sigmoid function is given as :

$$A = 1/(1 + e^-x)$$

We can see that between -2 and 2, y values changes abruptly, making sigmoid function a good candidate for classification tasks. But toward aither end of x, the gradient change will be small, giving rise to ’vanishing gradient’ problem.

tanh function: tanh function is given as:

$$A = 2/(1 + e^-2x) - 1$$

The gradients are steeper in the tanh function than sigmoid. But the tanh function also has the ’vanishing gradient’ problem like the sigmoid.

We will be comparing the output of our model with different activation functions.

- Number of hidden units: The number of features for the final representation is another parameter that we will be focusing on. For this project we will restrict reduced dimension size to 5/10 features.

2.1 Predicting Survival of patients

For prediting survival of patients, we will use AFT regression method. Let T be the survival time. Suppose we have a random sample of size n from a target population. For a subject $i(i = 1, 2, ..., n)$, we have observed values of covariates $xi1, xi2, ..., xip$and possibly censored survival time t_i. We can express the AFT model as:

$$log(t_i) = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + W_i$$

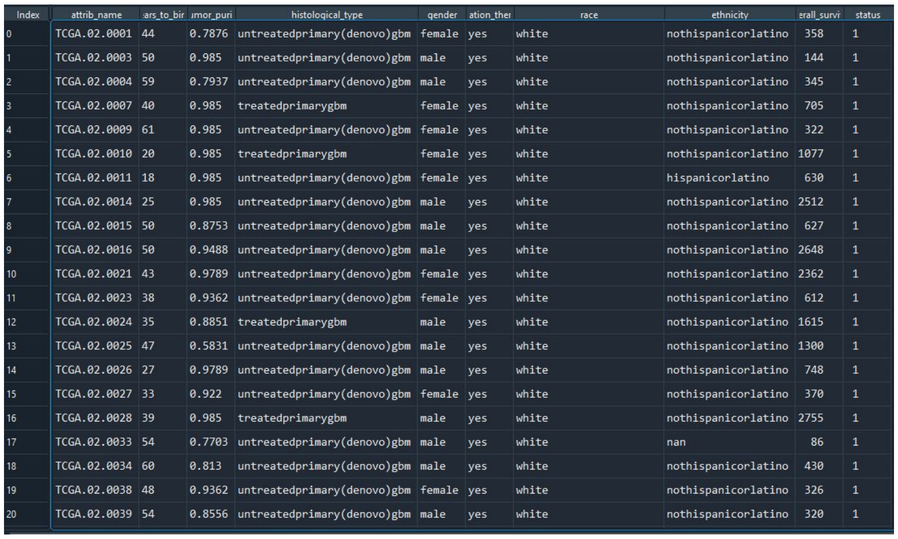
.We can select the errors to have different distribution like log-normal, weibull, exponential etc.

where $\beta = (\beta_0, \beta_1, ...)$are the regression coeffiecients. $W_i \sim$ i.i.d are the error

3 Methodologies

3.1 Overview of Dataset

We extracted 2 different dataset for our project. Clinical Data contains features such as gender, Tumor Purity, Years after birth, censoring information, so on. Here is the original clinical dataset after basic preprocessing:



Index	attrib_name	ars_to_bir	ımor_puri	histological_type	gender	ation_ther	race	ethnicity	erall_survi	status
0	TCGA.02.0001	44	0.7876	untreatedprimary(denovo)gbm	female	yes	white	nothispanicorlatino	358	1
1	TCGA.02.0003	50	0.985	untreatedprimary(denovo)gbm	male	yes	white	nothispanicorlatino	144	1
2	TCGA.02.0004	59	0.7937	untreatedprimary(denovo)gbm	male	yes	white	nothispanicorlatino	345	1
3	TCGA.02.0007	40	0.985	treatedprimarygbm	female	yes	white	nothispanicorlatino	705	1
4	TCGA.02.0009	61	0.985	untreatedprimary(denovo)gbm	female	yes	white	nothispanicorlatino	322	1
5	TCGA.02.0010	20	0.985	treatedprimarygbm	female	yes	white	nothispanicorlatino	1077	1
6	TCGA.02.0011	18	0.985	untreatedprimary(denovo)gbm	female	yes	white	hispanicorlatino	630	1
7	TCGA.02.0014	25	0.985	untreatedprimary(denovo)gbm	male	yes	white	nothispanicorlatino	2512	1
8	TCGA.02.0015	50	0.8753	untreatedprimary(denovo)gbm	male	yes	white	nothispanicorlatino	627	1
9	TCGA.02.0016	50	0.9488	untreatedprimary(denovo)gbm	male	yes	white	nothispanicorlatino	2648	1
10	TCGA.02.0021	43	0.9789	untreatedprimary(denovo)gbm	female	yes	white	nothispanicorlatino	2362	1
11	TCGA.02.0023	38	0.9362	untreatedprimary(denovo)gbm	female	yes	white	nothispanicorlatino	612	1
12	TCGA.02.0024	35	0.8851	treatedprimarygbm	male	yes	white	nothispanicorlatino	1615	1
13	TCGA.02.0025	47	0.5831	untreatedprimary(denovo)gbm	male	yes	white	nothispanicorlatino	1300	1
14	TCGA.02.0026	27	0.9789	untreatedprimary(denovo)gbm	male	yes	white	nothispanicorlatino	748	1
15	TCGA.02.0027	33	0.922	untreatedprimary(denovo)gbm	female	yes	white	nothispanicorlatino	370	1
16	TCGA.02.0028	39	0.985	treatedprimarygbm	male	yes	white	nothispanicorlatino	2755	1
17	TCGA.02.0033	54	0.7703	untreatedprimary(denovo)gbm	male	yes	white	nan	86	1
18	TCGA.02.0034	60	0.813	untreatedprimary(denovo)gbm	male	yes	white	nothispanicorlatino	430	1
19	TCGA.02.0038	48	0.9362	untreatedprimary(denovo)gbm	female	yes	white	nothispanicorlatino	326	1
20	TCGA.02.0039	54	0.8556	untreatedprimary(denovo)gbm	male	yes	white	nothispanicorlatino	320	1

Figure 3: Clinical Data

We deleted the rows which had 'NAN' values as the status. Race, radiation therapy and ethnicity column are discarded because we don't think these features contribution is not significant in overall survival. We also extracted tissue samples for this patients which is our RNA-seq dataset. Each row represents gene features for each patients. The total gene expression dataset has 528 rows and 4570 columns. After matching this with the clinical data, we were left with 500 samples.

3.2 Data Preprocessing

The RNA-seq data has 528 samples and 4571 gene values. There are 191 missing values in our entire dataset, all of which comes from ”RAC1” genes. We discarded the entire gene column and feed the data to the autoencoder.

We divided the data into 75% train and 10% test set. A validation set of 15% data were used when training.

3.3 Results with different Autoencoder

We will be building a basic autoencoder 1st, and then gradually try to improve the performance with different parameters and variations. We trained a basic autoencoder, a denoised autoencoder and a variational autoencoder for this project. For now, we restricted the output to 5 feature for each autoencoder. Sigmoid activations were used at each layer. For loss functions we used Mean squared error. Stochastic Gradient Descent was used as optimization algorithm with learning rate set to 0.1. Number of epochs were set to 100. Each autoencoder uses a hidden layer size of 2000,1000,500,100 and 5 respectively. Here is loss plot for the basic autoencoder:

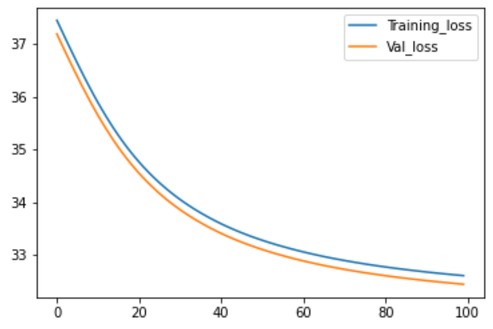


Figure 4: Loss plot for the basic autoencoder

The denoising autoencoder was trained with the same parameter. Only difference is we introduced gaussian noise at the input of the encoder. The amount of noise was set to 0.25.Here is the loss plot for Denoising Autoencoder:

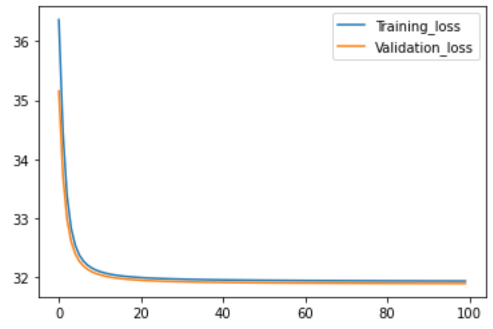


Figure 5: Loss plot for the denoised autoencoder

The variational autoencoder has technical differences compared to the previous 2 autoencoders.

- Prior Distribution We considered a mixture distribution of 2 gaussian distribution.
- An encoder with 5 dimensional output was used. The encoder converts the input into 100 dimension and finally outputs a 5 dimensional gaussian.
- The decoder takes the latent variables from encoder and feeds outputs an independent normal variable of size 4570.
- The loss function calculates mean of the negative log distribution of the decoder.

Here is the loss function for our variational autoencoder:

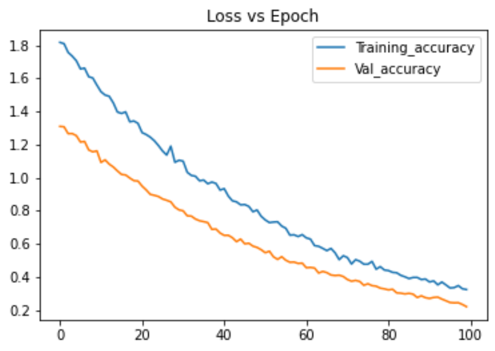
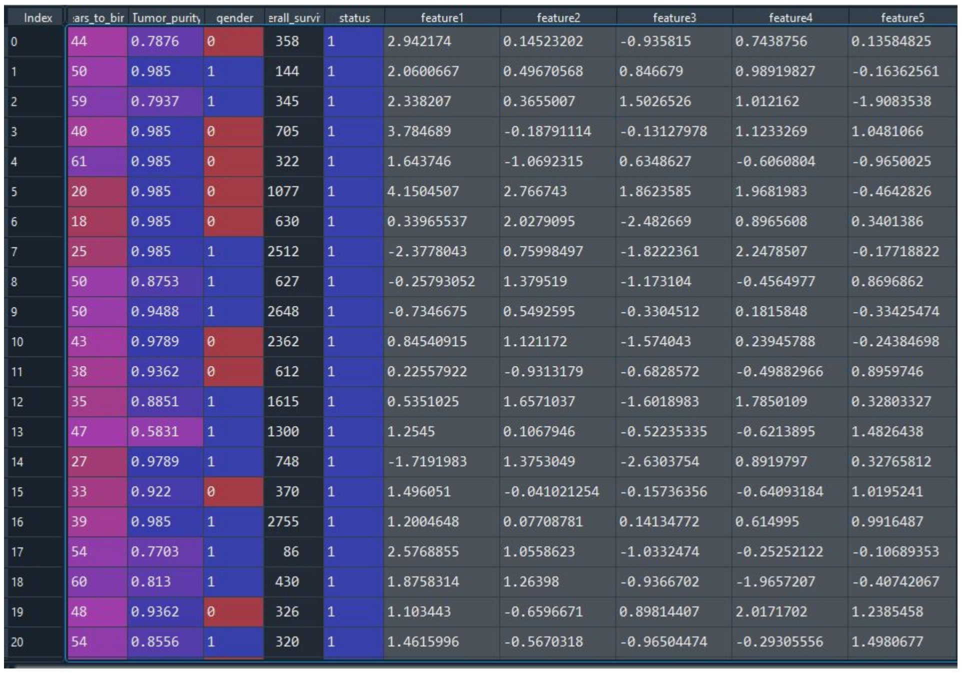


Figure 6: Loss function of the variational autoencoder

After the training, we extracted and merged the prediction from our encoder from each of the autoencoder to the cilinical dataset. Because of very low



Index	ars_to_bir	Tumor_purity	gender	erall_survi	status	feature1	feature2	feature3	feature4	feature5
0	44	0.7876	0	358	1	2.942174	0.14523202	-0.935815	0.7438756	0.13584825
1	50	0.985	1	144	1	2.0600667	0.49670568	0.846679	0.98919827	-0.16362561
2	59	0.7937	1	345	1	2.338207	0.3655007	1.5026526	1.012162	-1.9083538
3	40	0.985	0	705	1	3.784689	-0.18791114	-0.13127978	1.1233269	1.0481066
4	61	0.985	0	322	1	1.643746	-1.0692315	0.6348627	-0.6060804	-0.9650025
5	20	0.985	0	1077	1	4.1504507	2.766743	1.8623585	1.9681983	-0.4642826
6	18	0.985	0	630	1	0.33965537	2.0279095	-2.482669	0.8965608	0.3401386
7	25	0.985	1	2512	1	-2.3778043	0.75998497	-1.8222361	2.2478507	-0.17718822
8	50	0.8753	1	627	1	-0.25793052	1.379519	-1.173104	-0.4564977	0.8696862
9	50	0.9488	1	2648	1	-0.7346675	0.5492595	-0.3304512	0.1815848	-0.33425474
10	43	0.9789	0	2362	1	0.84540915	1.121172	-1.574043	0.23945788	-0.24384698
11	38	0.9362	0	612	1	0.22557922	-0.9313179	-0.6828572	-0.49882966	0.8959746
12	35	0.8851	1	1615	1	0.5351025	1.6571037	-1.6018983	1.7850109	0.32803327
13	47	0.5831	1	1300	1	1.2545	0.1067946	-0.52235335	-0.6213895	1.4826438
14	27	0.9789	1	748	1	-1.7191983	1.3753049	-2.6303754	0.8919797	0.32765812
15	33	0.922	0	370	1	1.496051	-0.041021254	-0.15736356	-0.64093184	1.0195241
16	39	0.985	1	2755	1	1.2004648	0.07708781	0.14134772	0.614995	0.9916487
17	54	0.7703	1	86	1	2.5768855	1.0558623	-1.0332474	-0.25252122	-0.10689353
18	60	0.813	1	430	1	1.8758314	1.26398	-0.9366702	-1.9657207	-0.40742067
19	48	0.9362	0	326	1	1.103443	-0.6596671	0.89814407	2.0171702	1.2385458
20	54	0.8556	1	320	1	1.4615996	-0.5670318	-0.96504474	-0.29305556	1.4980677

Figure 7: Final Dataset

variance of the output from encoder, we standerdized each feature columns. Here's how the final dataset looks like: Our final dataset was divided into 75 % training and 25 % test set

4 Evaluation

We use concordance index or c-index which is a metric to evaluate the predictions made by our model. It is a generalization of the area under the ROC curve (AUC) that can take into account censored data. It represents the global assessment of the model discrimination power: this is the model's ability to correctly provide a reliable ranking of the survival times based on the individual risk scores. It can be computed with the following formula:

$$Cindex = \frac{\sum_{i,j} 1_{T_j<T_i}.1_{n_j>n_i}.\delta_j}{\sum_{i,j} 1_{T_j<T_i}.\delta_j}$$

where, n_j is the risk score of a unit i
$1_{T_j<T_i} = 1$ if $T_j < T_i$, else 0.
$1_{n_j>n_i} = 1$ if $n_j > n_i$, else 0.

We used concordance index function of python which is under pysurvival package to evaluate results from different models. Next we use AFT model to fit this data. For that we used Weibull regression model. Here is training summary for the basic autoencoder:

```
                                coef    exp(coef)  ...              p    -log2(p)
param   covariate                                 ...
lambda_ Tumor_purity    0.458624     1.581896  ...   2.766812e-02    5.175631
        feature1       -0.008752     0.991286  ...   8.208847e-01    0.284748
        feature2        0.121640     1.129348  ...   2.647767e-03    8.561008
        feature3        0.035431     1.036066  ...   3.703594e-01    1.433002
        feature4       -0.056879     0.944708  ...   1.491432e-01    2.745230
        feature5        0.044207     1.045198  ...   2.995832e-01    1.738971
        gender         -0.080894     0.922292  ...   2.940841e-01    1.765699
        years_to_birth -0.027490     0.972885  ...   2.564293e-23   75.045785
        Intercept       7.659919  2121.584559  ...  1.981027e-207  686.652867
rho_    Intercept       0.270212     1.310243  ...   9.193076e-15   46.628374
Concordance = 0.65
AIC = 6294.48
log-likelihood ratio test = 111.81 on 8 df
-log2(p) of ll-ratio test = 65.75
```

Concordance index from test set is 0.6749

Next we fit the model with the features from denoising autoencoder. Here is the training summary:

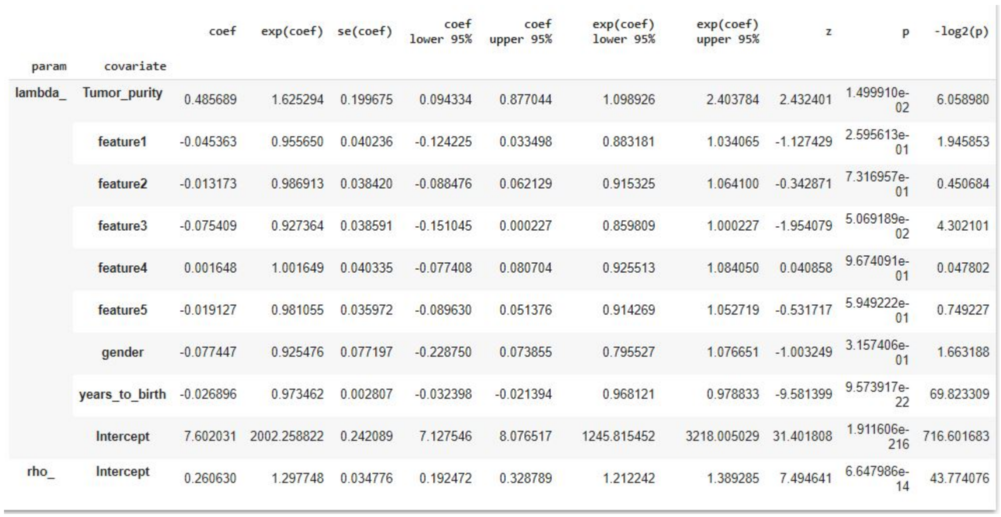
```
                            coef    exp(coef)  ...              p    -log2(p)
param   covariate                             ...
lambda_ Tumor_purity    0.384080     1.468263  ...   7.983587e-02    3.646819
        feature1       -0.091135     0.912894  ...   4.457189e-02    4.487722
        feature2        0.063202     1.065242  ...   1.796724e-01    2.476559
        feature3        0.038187     1.038925  ...   3.959889e-01    1.336468
        feature4        0.040105     1.040920  ...   3.179979e-01    1.652911
        feature5       -0.049310     0.951886  ...   2.584917e-01    1.951810
        gender         -0.082736     0.920594  ...   2.965496e-01    1.753655
        years_to_birth -0.026309     0.974034  ...   1.977760e-19   62.132766
        Intercept       7.655046  2111.271753  ...  4.962089e-194  642.143103
rho_    Intercept       0.267767     1.307042  ...   1.538325e-14   45.885633

Concordance = 0.65
AIC = 6295.13
log-likelihood ratio test = 111.17 on 8 df
-log2(p) of ll-ratio test = 65.31
```

Concordance index for the test set was found to be 0.6531274131274132

Here is the training summary for the variational autoencoder:

```
model lifelines.WeibullAFTFitter
duration col 'overall_survival'
```



param	covariate	coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	z	p	-log2(p)
lambda_	Tumor_purity	0.485689	1.625294	0.199675	0.094334	0.877044	1.098926	2.403784	2.432401	1.499910e-02	6.058980
	feature1	-0.045363	0.955650	0.040236	-0.124225	0.033498	0.883181	1.034065	-1.127429	2.595613e-01	1.945853
	feature2	-0.013173	0.986913	0.038420	-0.088476	0.062129	0.915325	1.064100	-0.342871	7.316957e-01	0.450684
	feature3	-0.075409	0.927364	0.038591	-0.151045	0.000227	0.859809	1.000227	-1.954079	5.069189e-02	4.302101
	feature4	0.001648	1.001649	0.040335	-0.077408	0.080704	0.925513	1.084050	0.040858	9.674091e-01	0.047802
	feature5	-0.019127	0.981055	0.035972	-0.089630	0.051376	0.914269	1.052719	-0.531717	5.949222e-01	0.749227
	gender	-0.077447	0.925476	0.077197	-0.228750	0.073855	0.795527	1.076651	-1.003249	3.157406e-01	1.663188
	years_to_birth	-0.026896	0.973462	0.002807	-0.032398	-0.021394	0.968121	0.978833	-9.581399	9.573917e-22	69.823309
	Intercept	7.602031	2002.258822	0.242089	7.127546	8.076517	1245.815452	3218.005029	31.401808	1.911606e-216	716.601683
rho_	Intercept	0.260630	1.297748	0.034776	0.192472	0.328789	1.212242	1.389285	7.494641	6.647986e-14	43.774076

Figure 8: Training results for variational autoencoder

```
event col 'status'
number of observations 500
number of events observed 431
log-likelihood -3140.26
Concordance 0.65
AIC 6300.53
log-likelihood ratio test 105.76 on 8 df
-log2(p) of ll-ratio test 61.62
```

Concordance index for the test set was found to be 0.6654826254826255.

Next step was to find out how the c index changes with increasing dimensions of our autoencoders. This time we changed output dimension to 10, leading us to have total 19 dimensional feature. Here is the result for standard autpoencoder for training data:

```
                            coef    exp(coef)  ...             p    -log2(p)
param   covariate                              ...
lambda_ Tumor_purity       0.341046     1.406418  ...  3.489032e-01    1.519101
        ethnicity          0.281114     1.324604  ...  2.751910e-03    8.505351
        features1         -0.105084     0.900249  ...  6.556745e-02    3.930876
        features10         0.036089     1.036748  ...  4.204201e-01    1.250097
        features2         -0.001258     0.998743  ...  9.732213e-01    0.039160
        features3          0.014634     1.014742  ...  7.242813e-01    0.465378
        features4          0.067125     1.069429  ...  1.413518e-01    2.822638
        features5         -0.057512     0.944111  ...  2.603407e-01    1.941527
        features6          0.121715     1.129432  ...  5.575925e-03    7.486573
        features7          0.002368     1.002371  ...  9.552147e-01    0.066103
        features8          0.095418     1.100119  ...  2.090719e-02    5.579857
        features9          0.159459     1.172877  ...  2.976067e-04   11.714306
        gender            -0.137637     0.871415  ...  9.765770e-02    3.356122
        histological_type -0.218823     0.803464  ...  1.913390e-01    2.385797
        race              -0.131312     0.876944  ...  4.267893e-02    4.550332
        radiation_therapy  0.314359     1.369382  ...  2.526768e-04   11.950419
        years_to_birth    -0.021582     0.978649  ...  3.791452e-11   34.618458
        Intercept          7.137726  1258.563133  ...  5.820887e-44  143.623597
rho_    Intercept          0.358897     1.431750  ...  2.503915e-19   61.792448
---
Concordance = 0.67
AIC = 5008.86
log-likelihood ratio test = 147.74 on 17 df
-log2(p) of ll-ratio test = 73.64
```

The test c index is 0.62. Next, we use denoised autoencoder for 10 dimensional output. Here are the results:

```
                               coef   exp(coef)  ...             p    -log2(p)
param   covariate                                ...
lambda_ Tumor_purity       0.717372    2.049041  ...  5.563441e-02    4.167879
        ethnicity          0.273259    1.314240  ...  3.436825e-03    8.184708
        features1         -0.015048    0.985065  ...  7.570747e-01    0.401492
        features10         0.022014    1.022258  ...  7.336125e-01    0.446910
        features2         -0.013435    0.986655  ...  7.670319e-01    0.382642
        features3         -0.001270    0.998731  ...  9.784189e-01    0.031476
        features4         -0.006006    0.994012  ...  8.983836e-01    0.154597
        features5          0.097043    1.101907  ...  6.991738e-02    3.838205
        features6         -0.062011    0.939873  ...  1.489144e-01    2.747445
        features7          0.063927    1.066015  ...  2.268782e-01    2.140010
        features8         -0.073212    0.929404  ...  1.711524e-01    2.546646
        features9          0.014610    1.014717  ...  7.747294e-01    0.368236
        gender            -0.101649    0.903346  ...  2.210244e-01    2.177723
        histological_type -0.194701    0.823081  ...  2.211612e-01    2.176830
        race              -0.127125    0.880624  ...  5.616745e-02    4.154122
        radiation_therapy  0.334465    1.397193  ...  7.131173e-05   13.775501
        years_to_birth    -0.021663    0.978570  ...  2.877761e-11   35.016262
        Intercept          6.731376  838.299588  ...  1.775666e-38  125.404907
rho_    Intercept          0.329523    1.390305  ...  1.466538e-16   52.598435
Concordance = 0.66
AIC = 5031.16
log-likelihood ratio test = 125.43 on 17 df
-log2(p) of ll-ratio test = 59.29
```

Test c index is found to be 0.65. Lastly, we will see how the variational autoencoder performs:

```
    model lifelines.WeibullAFTFitter
```

```
duration col 'overall_survival'
event col 'status'
number of observations 375
number of events observed 324
log-likelihood -2350.49
time fit was run 2021-07-01 22:11:43 UTC
coef exp(coef) se(coef) coef lower 95% coef upper 95% exp(coef) lower 95%
exp(coef) upper 95% z p -log2(p)
lambda_ Tumor_purity 0.67 1.96 0.23 0.22 1.13 1.25 3.09 2.91 <0.005
8.12
feature1 -0.06 0.94 0.04 -0.15 0.02 0.86 1.02 -1.41 0.16 2.66
feature10 -0.09 0.91 0.04 -0.16 -0.02 0.85 0.98 -2.51 0.01 6.36
feature2 -0.02 0.98 0.04 -0.10 0.07 0.90 1.07 -0.40 0.69 0.53
feature3 -0.08 0.92 0.04 -0.17 0.00 0.85 1.00 -1.92 0.05 4.19
feature4 -0.06 0.94 0.04 -0.15 0.03 0.86 1.03 -1.39 0.17 2.60
feature5 -0.04 0.96 0.04 -0.12 0.04 0.89 1.04 -0.90 0.37 1.45
feature6 -0.02 0.98 0.05 -0.11 0.07 0.90 1.07 -0.45 0.65 0.62
feature7 0.03 1.03 0.04 -0.06 0.11 0.95 1.11 0.62 0.54 0.89
feature8 0.07 1.07 0.05 -0.02 0.16 0.98 1.17 1.50 0.13 2.90
feature9 0.01 1.01 0.04 -0.07 0.09 0.93 1.09 0.26 0.80 0.33
gender -0.01 0.99 0.09 -0.18 0.16 0.84 1.18 -0.07 0.94 0.09
years_to_birth -0.03 0.97 0.00 -0.03 -0.02 0.97 0.98 -8.19 <0.005 51.73
Intercept 7.32 1511.01 0.29 6.75 7.89 856.22 2666.55 25.26 <0.005 465.26
rho_ Intercept 0.30 1.35 0.04 0.22 0.38 1.25 1.47 7.33 <0.005 41.97
Concordance 0.63
AIC 4730.98
log-likelihood ratio test 92.63 on 13 df
-log2(p) of ll-ratio test 44.38
```

The test c-index was found to be 0.64

5 Discussion

A c-index score of 0.5 indicates a fully random score, whereas a score of 1 indicate a perfect match. In that sense, we can see our model performed reasonably. All of the models scored around 0.65, which is not very bad. There are many areas to imporve. For example, the number of autoencoder features for prediction was selected to be 5. We can change the size and see how it affects the c-index. Moreover, we can tune the autoencoder with grid-search to improve our model. Also, we can use another evaluation metric to see how it is affected by reduced features.

References

[1] Pascal Vincent , Hugo Larochelle, and Isabelle Lajoie. *Learning Useful Representations in a Deep Network with a Local Denoising Criterion.* Journal of Machine Learning Research 11 (2010) 3371-3408

[2] Hajime Uno, Tianxi Cai, Michael J.Pencina, Ralph Augostino. the C-statistics for Evaluating Overall Adequacy of Risk Prediction Procedures with Censored Survival Data Stat Med. 2011 May 10; 30(10): 1105–1117. doi:10.1002/sim.4154

[3] CARL DOERSCH. *Tutorial on Variational Autoencoders.* August 16, 2016.

[4] *Avinash Sharma V.* [*Understanding Activation Functions in Neural Networks*]. Medium.com, March 30, 2017.

[5] Knuth: Computers and Typesetting,
`http://www-cs-faculty.stanford.edu/~uno/abcde.html`