**Predict Overall Survival of Breast Cancer Patients by combining Multi-Omics Data**

Contents

[**1.** **Introduction** 2](#_Toc26675587)

[**2.** **Related Work** 2](#_Toc26675588)

[**3.** **Dataset** 3](#_Toc26675589)

[**4.** **Data exploratory analysis and Feature selection/extraction** 4](#_Toc26675590)

[**5.** **Methods** 9](#_Toc26675591)

[5.1 **Models applied** 10](#_Toc26675592)

[5.2 **Modelling architecture** 11](#_Toc26675594)

[5.1.1 Combined dataset architecture with PCA 11](#_Toc26675595)

[5.1.2 Modelling on individual datasets with PCA 12](#_Toc26675597)

[5.1.3 Neural Network architecture 12](#_Toc26675598)

[6 **Experiments and Results** 13](#_Toc26675599)

[6.1 **Case1: Predict the survival time of the patient** 13](#_Toc26675600)

[6.1.1 Performance and error analysis on individual dataset 13](#_Toc26675601)

[6.1.2 Performance and error analysis on combined dataset 14](#_Toc26675602)

[6.2 **Case2: Predict the survival status of the patient** 22](#_Toc26675604)

[6.2.1 **Performance and error analysis on individual dataset** 23](#_Toc26675605)

[6.2.2 **Performance and error analysis on combined dataset** 24](#_Toc26675606)

[7 **Conclusion** 31](#_Toc26675608)

[8 **Future Work** 32](#_Toc26675609)

[9 **References** 32](#_Toc26675610)

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Document Version** | **Author** | **Comments** |
|  |  |  |  |
| 12/04/2019 | 0.1 | Sriparna Mondal | 1st draft of project report |

**Abstract**

Precision medicine, an emerging field of medicine, has enabled prospects of customization of healthcare, medical decisions and treatments tailored to individual patients. The use of genomic and transcriptomic biomarkers as well as other multi-omics data has played a major role in precision oncology. Concurrent with the explosion of clinically relevant molecular data, the application of machine learning methods to multi-omics datasets has become more commonplace. In this paper, I tried to present a novel method of combining multi-omics datasets to predict breast cancer patients’ overall survival.

# **Introduction**

Cancer is a notoriously heterogeneous disease which makes tracking its progression and severity incredibly difficult. The most common cancer staging system is the TNM (tumor, nodes, metastasis) system which is based primarily on clinical information like tumor size, extent of spread, etc. [1]. Combining phenotypic and molecular data from cancer patients can lead to more detailed descriptions of disease progression and severity. I experimented with three molecular datasets (DNA methylation, RNASeq and miRNASeq data) along with Clinical dataset to predict the overall survival of breast cancer patients. I experimented with linear models (SVM, KNN, PCA with SVM) on individual datasets along with Neural nets followed by applying models on combined datasets which I will explain in subsequent sections.

# **Related Work**

Sathipati and Ho [12] used an optimized SVM regression to identify miRNA signatures associated with survival time in patients with lung adenocarcinoma. They used a novel feature selection algorithm called IBCGA [13] and these features were then fed into traditional SVR. Although their custom SVR outperformed other regression methods, it did not generalize well to unseen validation data. Another issue with this paper was the size of datasets.

In another instance, Zhu et al. [6] incorporated patient profiles of somatic mutations, DNA copy number, DNA methylation, mRNA expression, miRNA expression, protein expression and their combinations to predict overall survival (3,382 samples across 14 cancer types). They created similarity/kernel matrices for all molecular datasets for all patients to understand the underlying biology across cancers that was leading to disease progression. They showed that predictive power depended on the cancer type and varied across molecular profiles; no single modality of data was superior across all cancers.

The work of Chaudhary et al. [9] is the most comparable to mine where they use RNASeq, miRNA,and methylation data from patients with hepatocellular carcinoma to identify survival subpopulations using an autoencoder, a Cox-PH feature selection strategy, and k-means clustering. The validity of the resulting clusters/subpopulations was confirmed using non-TCGA molecular datasets. In another example, Kwon et al. [8] combined RNASeq and miRNA data with an SVM classifier to identify prognostic biomarkers for pancreatic ductal adenocarcinoma. They similarly validated their findings by using external datasets and report 705 multi-markers for 27 miRNAs and 289 genes as promising potential biomarkers. Lastly, given the diversity of problems and machine learning approaches, the study by Lin and Lane [5] offered a detailed overview of how to approach multi-omics data integration. Based off their work, I opted to use model-based integration using RNASeq, miRNA, and methylation data for individual classifiers.

# **Dataset**

I have taken 4 datasets from the Linked Omics database [2] which has organized and compiled data from The Cancer Genome Atlas (TCGA) project. In particular, I obtained datasets of TCGA patients with invasive breast carcinoma [3] (designated TCGA-BRCA, all datasets had patients as rows and features as columns):

1. The clinical dataset : this has details of the patients including gender, age, tumor purity, ER status, PR status, overall survival days and survival status ( whether dead or alive).
2. The gene-level miRNA dataset : MicroRNAs (miRNAs) are responsible for the regulation of target genes involved in various biological processes and may play oncogenic or tumor suppressive roles. Integrated miRNA and related gene analyses in different types of cancers have been the focus of many studies.
3. The HiSeq gene-level RNASeq dataset: RNA-Seq with next-generation sequencing (NGS) is increasingly the method of choice for researchers studying the transcriptome. RNA-Seq allows researchers to detect both known and novel features in a single assay, enabling the detection of transcript isoforms, gene fusions, single nucleotide variants, and other features without the limitation of prior knowledge.
4. The gene-level DNA Methylation dataset: DNA methylation, an important epigenetic mark, is well known for its regulatory role in gene expression, especially the negative correlation in the promoter region. However, its correlation with gene expression across genome at human population level has not been well studied.

Regarding units, the RNASeq and miRNA datasets were provided in log-2 normalized RSEM and RPM, respectively; the methylation dataset was reported in centered beta values.

Below table shows the no. of records and features present in the datasets.

|  |  |  |
| --- | --- | --- |
|  | **Total patients** | **Features** |
| Clinical | 1098 | 21 |
| Methylation | 784 | 20107 |
| RNASeq | 1094 | 20155 |
| miRNA | 756 | 824 |

**Observations:**

* All the molecular datasets do not have all the patients’ data enlisted in clinical dataset. In fact, out of 1098 patients present in Clinical dataset, **only 612** records are common across all datasets.
* Almost all the datasets have more numbers of features compared to no. of records/rows. So, we can say all the datasets suffer from ‘curse of dimensionality’ which is very common for multi-omics datasets.
* Data exploratory analysis should be done carefully.
* Before splitting up data for training and testing, it needs to be ensured that all datasets should have records for common patients.

# **Data exploratory analysis and Feature selection/extraction**

Machine learning and deep learning algorithms learn from data, which consists of different types of features. The training time and performance of a machine learning algorithm depends heavily on the features in the dataset. Ideally, we should only retain those features in the dataset that help our machine learning model learn something. Unnecessary and redundant features not only slow down the training time of an algorithm, but they also affect the performance of the algorithm.

For feature selection, both filter methods and wrapper methods have been tried.

1. **Handling missing values:**

Clinical dataset – Patients with missing values in the target columns ‘overall\_survival’ and ‘status’ have been dropped. Since these two are the target columns, there is no point in retaining those records if the corresponding column values are missing.

Methylation dataset – There were around 2079 features which have missing records in this dataset. Below steps were carried out to reduce it.

1. At first, identified which features in the dataset have missing values and how many. Embedding the result here for reference.

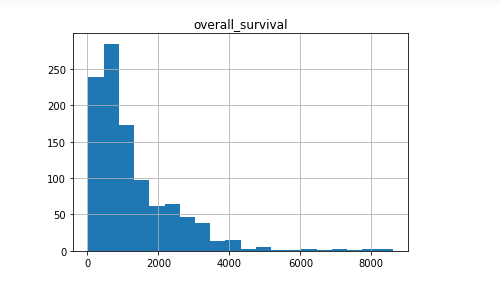


1. Analyzing the results, it seems there are some features which have very few missing records. So instead of discarding all the missing values, altogether better to go step by step.
2. Dropped columns which have more than 10 missing values. Now row numbers remain intact. Column numbers dropped to 19661.
3. Next, dropped columns which have more than 1 missing value. Now, there is big drop in row numbers. There are only 612 records which are common across all datasets, so any reduction further in that, might put additional constraints on the experiments whereas there are more than 20k features. So, decided to drop all features which have missing values.

RNASeq and miRNA datasets do not have missing records.

1. **Survival data analysis:**

The dataset had P(T > t), very few patients with high overall survival. Gene and molecular biomarkers cannot provide such precise survival time in days. The initial attempts to predict survival time as regression failed due to (1) non-uniform distribution of samples (2) granular survival time. Most literature I surveyed, used classification to predict short versus long term survival. Hence, the problem has been reframed as a classification problem.



The overall survival time has been splited into three buckets : short (<1.5yrs), medium term(1.5-3.5yrs), high (>3.5yrs) survival time.

1. **Check features with low variance:**

Constant features are the type of features that contain only one value for all the outputs in the dataset. Constant features provide no information that can help in classification of the record at hand. Eventually, it might be better to remove those features.

To do so will use Variance Threshold function that we imported earlier. The function requires a value for its threshold parameter. Passing a value of zero for the parameter will filter all the features with zero variance. This method was applied for all molecular datasets.

* RNA dataset: With threshold =0, its output was 50 .
* Methylation dataset: With threshold=0, its output was 0.
* miRNA dataset: With threshold=0, its output was 0.

Now, will check for Quasi constant features setting threshold = 0.01.

* RNA dataset: With threshold =0.01, its output was 438.
* Methylation dataset: With threshold =0.01, its output was 8814. This has good number of Quasi constant features.
* MiRNA dataset: It has approx. 142.

Will check classifier performance before and after removing low variance features.

1. **Check correlated Features using corr() Method:**

Two or more than two features are correlated if they are close to each other in the linear space. Will create correlation matrix for the columns in the dataset and an empty set that will contain all the correlated features. Next, will loop through all the columns in the correlation\_matrix and will add the columns with a correlation value of 0.95/0.98 to the correlated\_features.

* RNA dataset: Setting correlation value 0.98 yielded 7 rows while 0.95 gave 62.
* Methylation dataset: Setting to 0.98 , output turns out to be 441.
* Mi RNA : did not get significant number.

Will check classifier performance before and after removing the correlated features.

1. **Check duplicate records:**

Duplicate features are the features that have similar values. Duplicate features do not add any value to algorithm training, rather they add overhead and unnecessary delay to the training time. Therefore, it is always recommended to remove the duplicate features from the dataset before training.

This will be done taking the transpose of training data followed by using duplicated() method to identify duplicate rows and finally apply drop\_duplicates() to drop the duplicate features retaining first copy.

* RNA dataset: This does not have any duplicate features.
* Methylation dataset : This has approx. 100, but chose not to drop , it looks like space in some column names did not yield accurate results.

1. **Univariate feature selection:**

This is one “**filter**” method-based feature selection technique. Univariate feature selection works by selecting the best features based on univariate statistical tests. It can be seen as a preprocessing step to an estimator. For this experiment, ‘SelectKBest’ function and ‘f\_classif’ scoring function has been used for e.g. SelectKBest(f\_classif, k=20)

For individual datasets, top 20 features and corresponding scores have been embedded in the excel.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **RNA dataset** | |  | **Methylation Dataset** | |  | **miRNA dataset** | |
|  |  |  |  |  |  |  |  |
| **Feature Name** | **Score** |  | **Features** | **Score** |  | **Features** | **Score** |
| hsa-mir-1231 | 17.22293 |  | CLCA3P | 16.776734 |  | hsa-mir-26a-2 | 13.94132 |
| hsa-mir-3150b | 16.05684 |  | OR10A6 | 14.3803 |  | hsa-mir-150 | 13.21052 |
| hsa-mir-1257 | 13.66043 |  | COQ2 | 10.617566 |  | hsa-mir-29a | 12.14908 |
| hsa-mir-151 | 13.51595 |  | OGN | 9.971466 |  | hsa-mir-195 | 12.14565 |
| hsa-mir-605 | 13.33366 |  | ENTPD8 | 9.606852 |  | hsa-mir-146a | 11.48632 |
| hsa-mir-24-2 | 12.60432 |  | ZNF347 | 9.369038 |  | hsa-mir-3687 | 11.45853 |
| hsa-mir-3616 | 11.82344 |  | VNN3 | 8.994012 |  | hsa-mir-101-1 | 10.71934 |
| hsa-mir-1915 | 11.30421 |  | GPRC6A | 8.798931 |  | hsa-mir-424 | 10.58952 |
| hsa-mir-421 | 11.29383 |  | TMC4 | 8.485852 |  | hsa-mir-3676 | 10.28741 |
| hsa-mir-23a | 10.53366 |  | KAZALD1 | 8.446548 |  | hsa-mir-99a | 9.804673 |
| hsa-mir-320c-1 | 10.49233 |  | AMBN | 8.434302 |  | hsa-mir-1247 | 9.737114 |
| hsa-mir-3927 | 10.41992 |  | SGCG | 8.348616 |  | hsa-mir-202 | 9.493321 |
| hsa-mir-23b | 10.14045 |  | RGS21 | 8.273399 |  | hsa-mir-125b-2 | 9.408153 |
| hsa-mir-937 | 9.894804 |  | OR10A5 | 8.054077 |  | hsa-mir-3157 | 9.017903 |
| hsa-mir-2115 | 9.565763 |  | ALDH1A2 | 7.883902 |  | hsa-mir-3130-1 | 8.693305 |
| hsa-mir-3155 | 9.329302 |  | SUSD3 | 7.78008 |  | hsa-mir-139 | 8.488334 |
| hsa-mir-340 | 9.155176 |  | SUCLG2 | 7.73767 |  | hsa-mir-497 | 7.902064 |
| hsa-mir-3914-1 | 9.006405 |  | PON2 | 7.637967 |  | hsa-mir-551b | 7.898551 |
| hsa-mir-573 | 8.571417 |  | KLRC1 | 7.415051 |  | hsa-mir-411 | 7.876713 |
| hsa-mir-3654 | 8.53218 |  | AKT1 | 7.396328 |  | hsa-mir-24-2 | 7.279065 |

One of the major disadvantages of univariate filter methods is that they may select redundant features because the relationship between individual features is not taken into account while making decisions. So, did not drop any feature based on this output. But, this list shows features with high scoring which might be useful for further bio-analysis.

1. **Step forward sequential feature selector:**

It is a wrapper method in which marry the feature selection process to the type of model being built, evaluating feature subsets in order to detect the model performance between features, and subsequently select the best performing subset.

Sebastian Raschka's ‘mlxtend’ library includes an implementation (Sequential Feature Selector).I tried this approach, but it was taking very long even on high end server. Even after running continuously for 3 days, there were no significant output. So, I thought about testing this. Since most of my datasets have huge number of number columns (>20k)

1. **Principal Component Analysis (PCA):**

Principal component analysis (PCA) is a technique for dimensionality reduction, which is the process of reducing the number of predictor variables in a dataset. More specifically, PCA is an unsupervised type of feature extraction, where original variables are combined and reduced to their most important and descriptive components.

Multi-omics dataset suffers from the curse of dimensionality p >> n . Based on prior studies [10-11], Supervised PCA has proven to be effective in selecting subset of features. Let X be a (n × p) feature data matrix and Y be an n dimensional overall survival target vector. For Supervised PCA, we (i) compute standard regression coefficients for each feature



(ii) form a reduced matrix of size k consisting of ||Xj || features with highest Sk regression coefficients, (iii) compute principal components of this reduced matrix and use these as input features for classification.

I tried different values of k on for each dataset ,will share before PCA and after PCA classifier results in subsequent sections. Dimensionality reduction is a crucial technique in my project.

1. **Autoencoder:**

Autoencoders (AE) are neural networks that aims to copy their inputs to their outputs. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation. Today data denoising and dimensionality reduction for data visualization are considered as two main interesting practical applications of autoencoders. With appropriate dimensionality and sparsity constraints, autoencoders can learn data projections that are more interesting than PCA or other basic techniques.

Will employ both Vanilla and Multi-layer auto encoder and test. Autoencoder encoded compressed output will be fed to Multi-layer perceptron.

# **Methods**

In this section will discuss various methods used for selecting a framework for multi-omics-based survival prediction. Did a baseline evaluation of performance on each dataset using standard classification methods (SVM, Random Forest, Gradient Boosted decision trees, K-NN) and Neural Net. Based on prior work, later combined three datasets to train a model that generalized better as compared to models trained on individual datasets.

I have two target variables to predict.

1. **Predict the survival time of the patient** – whether the patient survived short (<1.5yrs), medium term(1.5-3.5yrs), long (>3.5yrs) term. At first, as mentioned in the previous section the problem was converted to Classification problem by altering the overall survival days in Clinical dataset. Then the dataset was joined with three molecular datasets individually based on attribute ID which is patient ID.
2. **Predict the survival status of the patient** – whether the patient is alive or passed away. Here, the ‘status’ column of the clinical dataset is to be predicted.

I applied two methodologies, at first baselining on each molecular dataset and later applying models on combined datasets.

## 5.1 **Models applied**

### Here three molecular datasets – Methylation ,RNA seq, miRNA datasets were combined with Clinical dataset. Data exploratory analysis was done, and feature selection methodologies were carried out as described in section 4.Then following models were applied.

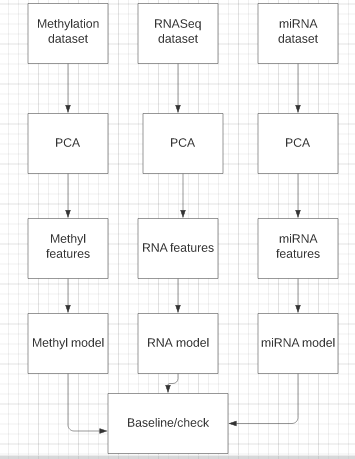
1. KNN : K-nearest neighbor (KNN) is a non-parametric, lazy learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point. By experimenting with K which signifies the numbers of nearest neighbors , we can optimize the model. Here , the model has been tested for K=3 to 30.
2. SVM: A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side. Here SVM was tested with ‘Linear’ kernel with varying values of penalty parameter C and gamma.
3. Random Forest: Random forest, like its name implies, consists of many individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction. There are quite a few hyperparameters to tune to optimize the model. It turned out that by tuning n\_estimator, max\_depth, min\_samples\_split, min\_samples\_leaf it is possible to get better performance from Random Forest compared to other models.
4. XGBoost : XGBoost stands for Extreme Gradient Boosting; it is a specific implementation of the Gradient Boosting method which uses more accurate approximations to find the best tree model. It computes second-order gradients, i.e. second partial derivatives of the loss function and employs advanced regularization (L1 & L2), which improves model generalization. By tuning hyperparameters like max\_depth , n\_estimators,min\_sample\_split, learning\_rate experimented with the model performance.
5. Creating an ensemble model : Some literature paper recommended ensemble model for breast cancer data. We have tested with ensemble of SVM, K-NN and Gradient Boosted Decision Tree.
6. Neural net : A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Recent literature suggested promising results in survival prediction with Deep Neural Networks and multi-omics data. A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. So Neural net model has been designed which includes an autoencoder for reducing and capturing non-linear relationships between attributes, and a multilayer perceptron for the prediction task.

## 5.2 **Modelling architecture**

### Combined dataset architecture with PCA

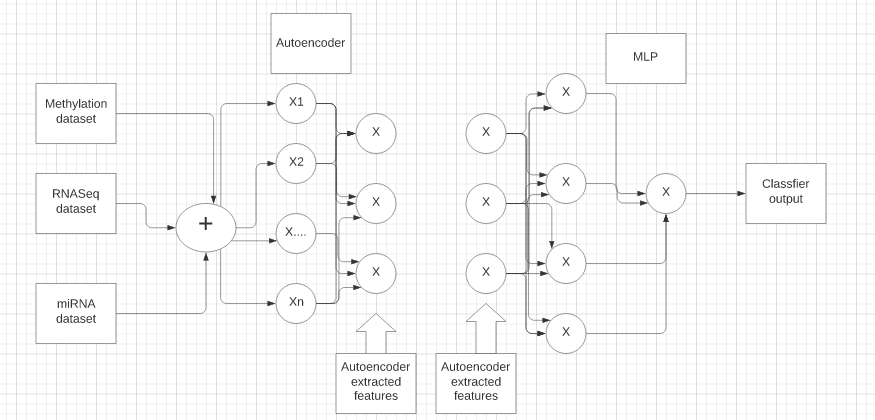
### 

### Modelling on individual datasets with PCA



### Neural Network architecture

Here showing the architecture for Autoencoder with MLP.



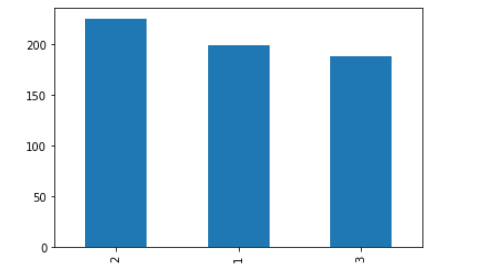
This is for illustration purpose. The number of layers and neurons in each layer will be different for the problem here.

# 6 **Experiments and Results**

The primary challenge with these datasets was high dimensionality. Putting below the performance reported by the models on individual datasets and combined datasets. Will report the results separately for before applying Feature extraction(FE) and after applying FE.

## 6.1 **Case1: Predict the survival time of the patient**

Will look at the results for the case of predicting the overall survival time of the patient. Following is the distribution of the classes – 1 (short term) , 2 (medium term) and 3 (long term) in the dataset.



We can see Class 2 = 36.7% , Class 1 = 32.5% and Class 3=30.72% approximately, so there is no major class imbalance. Though accuracy score might be a good indication of the model performance here but will evaluate in detail in terms of confusion matrix ,precision and recall.

### 6.1.1 Performance and error analysis on individual dataset

Before applying any feature extraction technique like PCA, the performance for various models on the individual datasets stands like below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **SVM** | **KNN** | **Random Forest** | **Gradient Boosting** | **XGBoost** |
| **Methylation** | 34.14% | 34.14% | 34.95% | **38%** | 34.57% |
| **miRNA** | 47.10% | **48.78% (n=20)** | 39% | 43.90% | 42.30% |
| **rnaSeq** | 33.17% | **38.3% (n=25)** | 35% | 37% | 36.70% |

After applying PCA, for different values of n (number of components), the performance looks like below –

RnaSeq Dataset:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **PCA** | | | |  |
| **Dataset** | **Models** | n=10 | n=20 | n=50 | n=100 | n=200 |
|  |  |  |  |  |  |  |
| RnaSeq | KNN | 0.3270142 | 0.3791469 | **0.42654028** | 0.4265403 | 0.402844 |
| SVM | 0.3317536 | 0.3507109 | 0.38388626 | 0.3601896 | 0.345972 |
| Random Forest | 0.3222749 | 0.3981043 | 0.37440758 | 0.3507109 | 0.331754 |
| GBM | 0.3601 | 0.37 | 0.36492891 | 0.3696682 | 0.42654 |

MiRNA dataset:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **Model** | **n=10** | **n=20** | **N=30** | **n=50** | **n=100** |
| miRNA | KNN | 40.6% (At n=40) | 41.4% at n=20 |  | 47.96% at n=20 | 43.9% at n=20 |
| SVM | 37.39% | 41.40% | **47.96%** | 46.30% | 46.30% |
| Random Forest | 35.70% | 34.14% | 39.83% | 47.15% | 38.21% |

Methylation dataset:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **Models** | n=10 | n=20 | n=50 | n=100 | n=200 |
| **Methylation** | KNN | 0.3252033 | 0.3495935 | 0.37398374 | 0.3333333 | 0.349593 |
| SVM | 0.3495935 | 0**.398374** | 0.39837398 | 0.3739837 | 0.357724 |
| Random Forest | 0.3821138 | 0.3821138 | 0.33333333 | 0.3739837 | 0.373984 |
| GBM | 0.3414634 | 0.3495935 | 0.3495935 | 0.3902439 | 0.341463 |

There has been improvement after applying PCA. Amongst the datasets, ‘miRNA’ has better correlation with the target survival column. Methylation and RNASeq datasets have high curse of dimensionality. SVM and KNN models have performed better compared to other models.

## 6.1.2 Performance and error analysis on combined dataset

First, will evaluate the models without applying any feature extraction method and later, applying feature extraction techniques. The combined dataset which is amalgamation of Clinical dataset and three molecular datasets (Methylation, miRNA , RNASeq) has **612 rows and 39005 features.**

#### 6.1.2.1 Performance evaluation prior Feature extraction

### KNN

Below table is the output before significant pre-processing, right after removing only the missing values .Looks like maximum is accuracy at nn=5.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| KNN | **Number of nearest neighbors(nn)** | | | | |
| 3 | 5 | 7 | 10 | 20 |
| 43.90% | **45.52%** | 43% | 37.39% | 39% |

#### SVM

Below table is the output before significant pre-processing. Average accuracy is at **37.39%.**

|  |  |  |  |
| --- | --- | --- | --- |
| SVM | **C** | **Gamma** | **Accuracy** |
| default | default | 36.50% |
| 1 | 0.001 | 37.39% |
| 1 | 0.005 | 37.39% |
| 10 | 0.005 | 37.39% |
| 50 | 0.005 | 37.39% |
| 100 | 0.005 | 37.39% |

#### Random Forest

Tested with below hyperparameters using GridSearchCV.

*n\_estimators = [100,150,200, 300, 500]*

*max\_depth = [5,6,7, 8, 15, 25, 30]*

*min\_samples\_split = [2, 5,6,7, 10, 15, 100]*

*min\_samples\_leaf = [1, 2,4, 5,6,7, 10,20]*

Maximum accuracy turns out to be **43.55%** for *{'max\_depth': 30,*

*'min\_samples\_leaf': 1,*

*'min\_samples\_split': 5,*

*'n\_estimators': 200}*

#### XGBoost

Tested with below hyperparameters.

*'max\_depth': range (2, 5, 10),*

*'n\_estimators': range(50, 100, 200),*

*'min\_sample\_split': range(5,10,20),*

*'learning\_rate':[0.1, 0.01, 0.05]*

Maximum accuracy obtained at **43%** for {*'learning\_rate': 0.05,*

*'max\_depth': 2,*

*'min\_sample\_split': 5,*

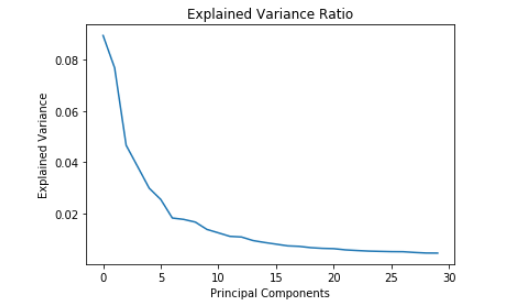
*'n\_estimators': 50}*

#### 6.1.2.2 Performance after feature extraction using PCA

Curse of dimensionality is a major problem in all datasets. Will look at the performance of the models after applying dimensionality reduction technique PCA. Checked by varying number of PCA components from 10 to 200.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Combined dataset** |  | n=10 | n=20 | n=30 | n=50 | n=100 | n=200 |
| KNN | 39.83% | 40.69% | 40.65% | 43.90% | 40.65% | 43% |
| SVM | 38.21% | 37.39% | 37.39% | 37.39% | 37.39% | 37.39% |
| Random Forest | 34.14% | 34.14% | 35.77% | 35.77% | 43.90% | 32.50% |

If I plot, PCA Explained Variance Ratio for n=30, it seems there is very little change after 10.So it will be better to test the models for 10 < n < 25.



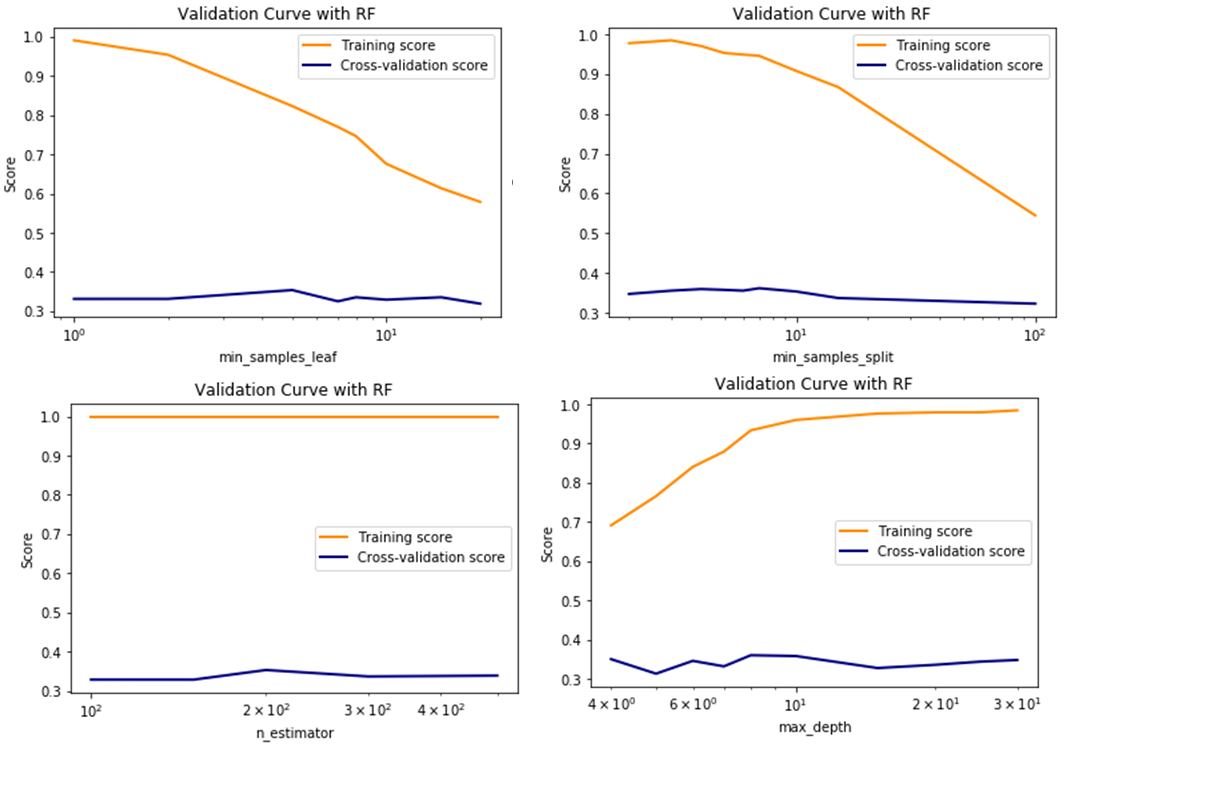
Next, will tune the hyperparameters and maximize the performance of the models.

Will look at Random Forest , XGBoost and other Ensemble models as these models gave better performance.

After trying with various combinations of **PCA number of components**, it turns out **n=10** works well. Will showcase the results here.

1. **Random Forest:**

* Plotted validation curves for various hyperparameters to get idea about optimal values.

- Then choose to test the model with below hyperparameters .

*Number of estimators = [100,150,200, 300, 500]*

*max\_depth = [4,5,6,7,8,10,15,20,25,30]*

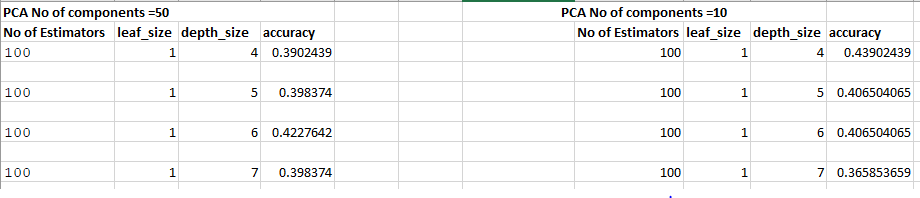
*min\_samples\_split = [2,3,4, 5,6,7, 10, 15, 100]*

*min\_samples\_leaf = [1, 2, 5,7,8, 10,15,20]*

* Did the testing exhaustively in manual mode and tried ‘GridSearchCV’ as well with 5-fold cross validation. Noted log loss function as well.
* Embedding here the detailed results for PCA number of components = 10 and 50 against the various hyperparameters mentioned above.



Copying small snapshot from the results excel -



It is seen that Random Forest performs best for below combination of hypermeters:

**PCA number of components = 10**

**Number of estimators = 150**

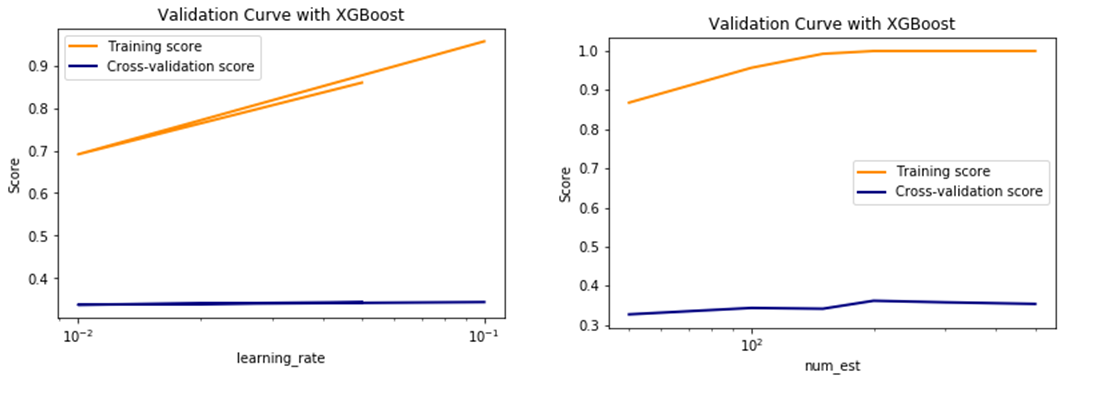
**Min\_samples\_leaf = 7**

**Maximum depth = 8**

**Accuracy is coming around 47.96%.**

1. **XGBoost:**

* Plotted some validation curves first.



* Did exhaustive search for best hyperparameters. Mostly narrowed down to below search conditions.

*num\_est = [50,100,150,200, 300, 500]*

*max\_depth = [4,5,6,7,8,10,15,20,25,30]*

*min\_samples\_split = [2,3,4, 5,6,7, 10, 15,20, 100]*

*min\_samples\_leaf = [1, 2, 5,7,8, 10,15,20]*

*learning\_rate=[0.1, 0.01, 0.05]*

* Embedding here the results of various combinations of hyperparameters.



**-** Did extensive grid search with 10-fold cross validation, when plotted against log loss function following combination came out as optimal.



* On further manual testing, the best combination of hyperparameters turns out to be below.

***n\_estimators = 300, n\_jobs = -1,min\_samples\_leaf = 1,max\_depth=6,learning\_rate=0.1***

**with accuracy of 44.71%.**

1. **Ensemble method with SVM, KNN and Gradient Boosting Classifier:**

Though ensemble model with weak classifiers can give good output but experimenting with this combination of models could not outdo other ensembles like Random Forest or XGBoost. Maximum accuracy obtained was around 41%.

Following was the hyperparameters chosen after experiments-

1. SVC – Kernel = linear, C = 10, gamma=0.001
2. KNN - metric='Euclidean', n\_neighbors=5
3. Gradient Boosting Classifier - n\_estimators = 150, min\_samples\_leaf = 6,min\_samples\_split =6,max\_depth=8
4. **Neural Net :**

Two approaches have been tried here. One is fully connected sequential model with input/hidden and output layer in which no dimensional reduction was done on the input. In another approach, autoencoder was used in conjunction with Multi-layer Perceptron (MLP).

1. **MLP without autoencoder:**

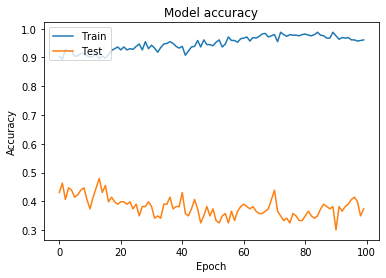
Since the dataset has few rows and high number of features, there was high chance of quick overfitting with Neural Net, so tested exhaustively and carefully the hyperparameters.

* After few experiments, Keras sequential model was chosen.
* Activation function was ReLU for hidden layers and Softmax for output layer since this has 3 class classification.
* Loss function was ‘categorical\_crossentropy’ ( since it is multi class problem) and optimizer was ‘adam’. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm.
* Epoch value was mostly 100 and batch size was 50. Increasing these values were overfitting the model quickly.
* Number of neurons in output layer is 3 , since the problem has 3 classes.

Next major part is to experiment with number of layers and number of neurons at each layer. Several combinations were tried. It was seen architecture with 3 hidden layers was working better. Illustrating results of few scenarios –

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No of neurons in layer 1** | **Any dropout?** | **No of neurons in layer 2** | **No of neurons in layer 3** | **Accuracy** |
|  |  |  |  |  |
| 200 |  | 100 | 200 | 30.08% |
| 500 |  | 50 | 100 | 37% |
| 500 | 0.5 | 500 | 200 | 40% |
| 500 | 0.5 | 100 | 50 | 41% |
| 500 | 0.5 | 100 | 80 | 45% |
| 500 | 0.5 | 200 | 100 | 41.5% |

Dropout was introduced to curb overfitting. The row highlighted above shows most apt configuration. Corresponding accuracy plot is below. In some cases, although the accuracy was little high, but were not considered because of overfitting issue.



1. **MLP with Autoencoder:**

Tested with Vanilla and Multi-layer autoencoder with various number of layers and encoding dimensions. Accuracies are close to the one reported by MLP without Autoencoder. Here also, ReLU activation functions were applied at hidden layers where Softmax at output layer.

Sharing some results –

* At encoding dimension =10, best accuracy was 43% , with 500 neurons at first hidden layer and 300 in the 2nd hidden layer. Accuracies dropped drastically if the number of neurons are reduced.
* At encoding dimension =50, best accuracy was 40% , with 500 neurons at first hidden layer and 100 in the 2nd hidden layer.
* At encoding dimension =100, best accuracy was 45% , with 500 neurons at first hidden layer and 300 in the 2nd hidden layer.

**So, here the max accuracy obtained was 45%.**

**Summary:**

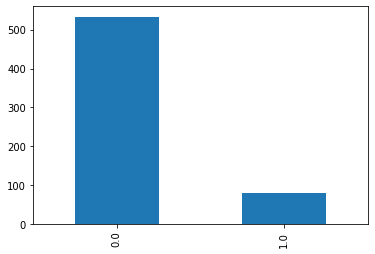
To predict the overall survival time, it seems Random Forest will be most apt one.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Precision** | **Recall** | **F1-Score** | **Accuracy** |
|  |  |  |  |  |
| Random Forest | 0.49 | 0.47 | 0.48 | 0.48 |

Though XGBoost and Neural Net also offers close by accuracies. Even after exhaustive hyperparameter tuning, the maximal accuracy went to around 48%. Main challenge here was the curse of dimensionality problem. To boost the accuracy, need to have more records and preferably other genomics datasets as well. I have taken only three molecular datasets.

## 6.2 **Case2: Predict the survival status of the patient**

Will look at the results for the case of predicting the overall survival status of the patient. Following is the distribution of the classes in the dataset.



There is major class imbalance, there are very few people ( status =1) who passed away. Will apply **SMOTE** sampling technique to balance the class.

### 6.2.1 **Performance and error analysis on individual dataset**

Following is the performance of various models in terms of accuracy scores on individual datasets before applying any feature extraction technique.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **SVM** | **KNN** | **Random Forest** | **Gradient Boosting** |
|  |  |  |  |  |
| Methylation | 85% | 84.6% (n=10) | 83.90% | 81.3% |
|  |  |  |  |  |
| RNASeq | 84.60% | 84.6% (n=10) | 84.60% | 82.30% |
|  |  |  |  |  |
| miRNA | 84.60% | 84.60% | 84.30% |  |

After applying PCA, taking different values of **n** on different datasets the performance looks like below,

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **PCA** | | | | | |
| **Dataset** | **Models** | n=10 | n=20 | n=50 | n=100 | n=200 | n=300 |
|  |  |  |  |  |  |  |  |
| **Methylation** | KNN | 83.98% | 83.00% | 83.90% | 83.98% | 84.30% | 83.60% |
| SVM | 84.60% | 84.60% | 84.60% | 84.60% | 84.60% | 84.60% |
| Random Forest | 84.30% | 84.30% | 84.60% | 84.30% | 84.60% | 84.60% |

Checked, how SVD works on Methylation dataset as well.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **Models** | **SVD** | | | | | |
|  |  | n=10 | n=20 | n=50 | n=100 | n=200 | n=300 |
| **Methylation** | KNN | 83.00% | 83.90% | 83.90% | 84.30% | 84.30% | 83.60% |
| SVM | 84.60% | 84.60% | 84.60% | 84.60% | 84.60% | 84.60% |
| Random Forest | 84.30% | 83.90% | 84.30% | 84.60% | 84.60% | 84.60% |

Looks like , the output is close by for both PCA and SVD. Decided to check PCA on other datasets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **PCA** | | | |
| **Dataset** | **Models** | n=10 | n=20 | n=50 |
| **RNASeq** | KNN | 84.60% | 84.60% | 84.60% |
| SVM | 84.60% | 84.60% | 84.60% |
| Random Forest | 83.30% | 83.60% | 84.60% |

It seems the accuracy score is approximately 84.6% for n=10 for both KNN and SVM. Same score has been given by Random Forest at n=20.

### 6.2.2 **Performance and error analysis on combined dataset**

#### 6.2.2.1 Performance evaluation prior Feature extraction

### KNN

Below table is the output before significant pre-processing, right after removing only the missing values .Looks like maximum is accuracy at nn=7.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| KNN | **Number of nearest neighbors(nn)** | | | | |
| 3 | 5 | 7 | 10 | 20 |
| 82.11% | 82.92% | **83.73%** | 83.73% | 83.4% |

#### SVM

Below table is the output before significant pre-processing. Average accuracy is at **83.73%.**

|  |  |  |  |
| --- | --- | --- | --- |
| SVM | **C** | **Gamma** | **Accuracy** |
|  |  |  |
| 1 | 0.001 | 83.73% |
| 1 | 0.005 | 83.73% |
| 10 | 0.001 | 83.73% |
| 50 | 0.001 | 83.73% |
| 100 | 0.005 | 83.73% |

#### Random Forest

Tested with below hyperparameters using GridSearchCV.with 3-fold cross validation

*n\_estimators = [100,150,200, 300, 500]*

*max\_depth = [5,6,7, 8, 15, 25, 30]*

*min\_samples\_split = [2, 5,6,7, 10, 15, 100]*

*min\_samples\_leaf = [1, 2,4, 5,6,7, 10,20]*

Maximum accuracy turns out to be **88%** for *{'max\_depth': 15,*

*'min\_samples\_leaf': 1,*

*'min\_samples\_split': 6,*

*'n\_estimators': 300}*

#### XGBoost

Tested with below hyperparameters.

sample\_leaf\_options = [1,2,3,4,5,7,10,20,50]

n\_estimators = [100,150,200,300]

max\_depth= [4,5,6,7,8,9]

Maximum accuracy obtained at **83.73%** for {*'learning\_rate': 0.05,*

*'max\_depth': 2,*

*'min\_sample\_split': 5,*

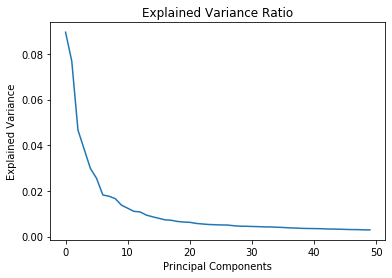
*'n\_estimators': 50}*

#### 6.2.2.2 Performance evaluation after Feature extraction

Curse of dimensionality is a major problem in all datasets. Will look at the performance of the models in terms of accuracies after applying dimensionality reduction technique PCA. Checked by varying number of PCA components from 10 to 200.

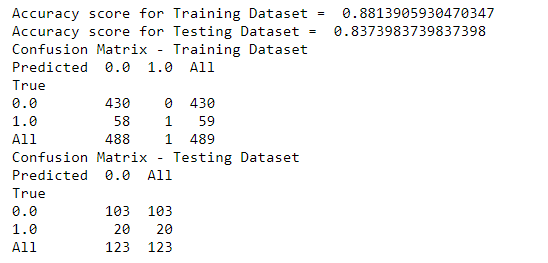
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Combined dataset** | **Models** | **n=10** | **n=20** | **n=30** | **n=50** | **n=100** | **n=200** |
| KNN | 83.73984 | 83.73984 | 83.73984 | 83.73984 | 83.73984 | 83.73984 |
| SVM | 83.73984 | 83.73984 | 83.73984 | 83.73984 | 83.73984 | 83.73984 |
| Random Forest | 82.92683 | 83.73984 | 82.92683 | 83.73984 | 83.73984 | 83.73984 |

If I plot, PCA Explained Variance Ratio for n=50, it seems there is very little change after 20.So it will be better to test the models for 15 < n < 30.



* Next, will tune the hyperparameters and maximize the performance of the models.
* Will look at Random Forest , XGBoost and other Ensemble models as these models gave better performance.
* It has been observed that at PCA number of components =20, the models performed better.
* Since, the class is imbalanced , accuracy will not be solo good measure. Will look at confusion matrix as well. In previous section, we saw mostly in all cases, the accuracy is on an average 88%, need to analyze in detail before starting to tune the models.

Checking for SVM model with kernel – linear , C = 1 and gamma = 0.001, the accuracy on training and test set and confusion matrix looks like below –



It clearly shows, though accuracy is good, but it is picking up majority classes. Same happens when I checked other models as well. We have very few 1’s in the status column in the dataset, it’s mostly 0’s.

There are two ways to fix it.

1. Oversampling the minority classes using SMOTE technique.
2. Run the model only on the majority class, which is target output 0, if the test data happens to be otherwise, it can generate anomaly.

Considering its genomics data and there is risk associated with wrong prediction, and both approaches have its positive and negative sides, will explore the first approach here.

1. **Random Forest:**

* Did exhaustive search to fine tune the hyperparameters. For e.g. checked the performance for the below range of parameters

*sample\_leaf\_options = [1,2,3,4,5,7,10,20,50]*

*n\_estimators = [100,150,200,300]*

*max\_depth= [4,5,6,7,8,9]*

* Also, plotted validation curves to cross verify and Grid search as well.
* Embedding below the excel containing results for various combinations of hyperparameters.



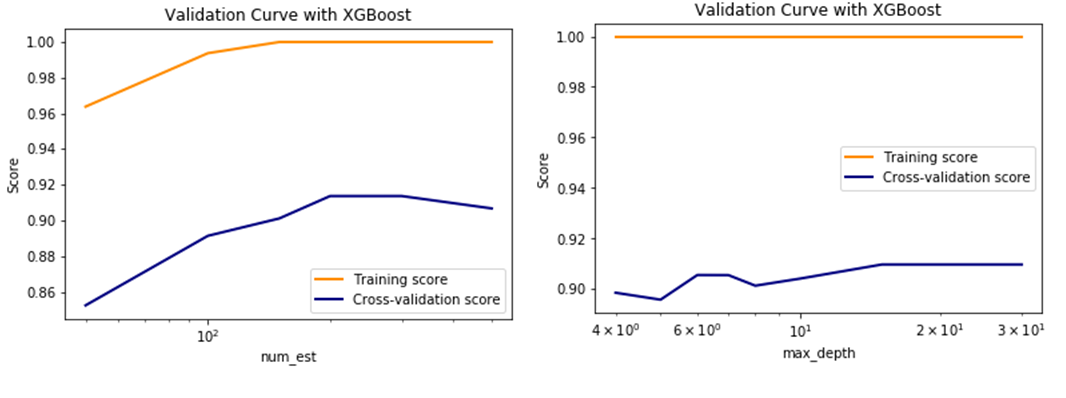
* Maximal accuracy obtained as **81%** for this combination of hyperparameters *n\_estimators = 150, n\_jobs = -1,random\_state =0, min\_samples\_leaf = 1,min\_samples\_split =4,max\_depth=6*

1. **SVM :**

Tried various hyperparameters, but maximal accuracy is low - **less than 75%.**

1. **XGBoost:**

* Plotted some validation curves first.



* Did exhaustive search for best hyperparameters. Mostly narrowed down to below search conditions.

*num\_est = [50,100,150,200, 300, 500]*

*max\_depth = [4,5,6,7,8,10,15,20,25,30]*

*min\_samples\_split = [2,3,4, 5,6,7, 10, 15,20, 100]*

*min\_samples\_leaf = [1, 2, 5,7,8, 10,15,20]*

*learning\_rate=[0.1, 0.01, 0.05]*

* Embedding here the results of various combinations of hyperparameters.



**-** Did extensive grid search with 10-fold cross validation.

On further manual testing, the best combination of hyperparameters turns out to be below.

***n\_estimators = 100, n\_jobs = -1,random\_state =0, min\_samples\_leaf = 1,min\_samples\_split =4,max\_depth=6,learning\_rate=0.1***

**with accuracy of 78%.**

1. **Ensemble model:**

Tried few ensemble model ( voting based) only with SVMs and with KNN,SVM and Boosting trees in combination. But the accuracies reported were less than Random Forest and XGBoost.

1. **Neural Net :**

The approaches here are same as that of case1.

1. **MLP without autoencoder:**

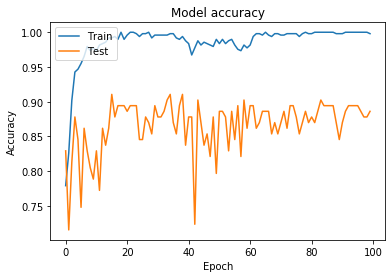
Since the dataset has few rows and high number of features, there was high chance of quick overfitting with Neural Net, so tested exhaustively and carefully the hyperparameters.

* The loss function, epoch and batch values are same as that in the case 1.
* Number of neurons in output layer is 2 , since the problem has 2 classes.

Next major part is to experiment with number of layers and number of neurons at each layer. Several combinations were tried. It was seen architecture with 3 hidden layers was working better. Illustrating results of few scenarios –

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No of neurons in layer 1** | **Any dropout?** | **No of neurons in layer 2** | **No of neurons in layer 3** | **Accuracy** |
| 500 |  | 50 |  | 85% |
| 500 | 0.5 | 50 |  | 89.40% |
| 500 | 0.5 | 50 | 200 | 85.30% |
| 500 | 0.5 | 50 | 100 | 90.20% |
| 200 | 0.5 | 20 | 200 | 87% |
| 500 | 0.5 | 100 | 500 | 87% |
| 500 | 0.2 | 50 | 200 | 87% |

The row highlighted above shows most apt configuration. Corresponding accuracy plot is below.



1. **MLP with Autoencoder:**

Tested with Vanilla and Multi-layer autoencoder with various number of layers and encoding dimensions. Accuracies are close to the one reported by MLP without Autoencoder. Here also, ReLU activation functions were applied at hidden layers where Softmax at output layer.

Sharing some results –

|  |  |  |  |
| --- | --- | --- | --- |
| **Encoded dimension** | **No of neurons in layer 1** | **No of neurons in layer 2** | **Accuracy** |
| 10 | 500 | 300 | 84.50% |
| 500 | 200 | 86% |
| 100 | 20 | 83.70% |
| 50 | 100 | 20 | 83.70% |
| 500 | 100 | 88.60% |
| 500 | 200 | 86% |
| 100 | 500 | 200 | 90.24% |
| 200 | 100 | 89.40% |
| 300 | 100 | 83.70% |
| 600 | 200 | 80.40% |

**So, here the max accuracy obtained was 90.24%.**

1. **Summary:**

Copying below the accuracy, precision, recall and F1 score of the models which gave better results for case 2 of predicting survival status of the patients.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Precision** | **Recall** | **F1-Score** | **Accuracy** |
|  |  |  |  |  |
| Random Forest | 0.8 | 0.81 | 0.8 | 0.81 |
| XGBoost | 0.77 | 0.78 | 0.78 | 0.78 |
| Neural net | 0.87 | 0.89 | 0.88 | 0.9 |
| MLP with autoencoder | 0.88 | 0.9 | 0.88 | 0.9 |

# 7 **Conclusion**

* Primary challenge here was high dimensionality of the datasets. Most of the datasets have >20k features and less records which resulted in overfitting.
* For such high dimensional data, it is better approach to apply dimensional reduction technique before feeding to the classifier. Feature extraction methods are preferred than filtering techniques, as the first one creates summarized version of the original features from a combination of the original set and therefore less lossy.
* For case 1, to predict the overall survival time of the patients , it seems PCA will be good Feature extraction method. Post that Random Forest classifier gave best performance with accuracy around 48%. Though additional patient records and correlation with more molecular datasets would have boosted the performance. Autoencoder with MLP also tried, but it turned out that the combination of PCA and Random Forest works better for this use case.
* Regarding case 2, to predict the survival status of the patients, it was seen that Autoencoder works better as Feature extraction technique. With autoencoder compressed output fed to Multi-layer perceptron, it yielded around 90% accuracy.

# 8 **Future Work**

I propose to explore the following approaches to improve performance: (1) Use class-balanced loss function to address imbalances in clinical dataset with natural split of overall survival, (2) incorporate phenotype data from clinical dataset as additional features, (3) incorporate clinically-relevant metrics like Inverse Probability Weighting into models, (4) deeper exploration using NN architecture, (5) augment clinical data from other sources to increase sample size and diversity, (6) use additional molecular data from the same clinical data source, (7) extend the method to predict overall for other cancer types.

# 9 **References**

[1] https://www.cancer.gov/about-cancer/diagnosis-staging/staging

[2] Vasaikar, S., Straub, P., Wang, J. and Zhang, B. (2017). LinkedOmics: analyzing multi-omics data within and across 32 cancer types. Nucleic Acids Research , 46(D1), pp.D956-D963.

[3] http://linkedomics.org/data\_download/TCGA-BRCA/

[4] A Framework for Implementing Machine Learning on Omics Data

[5] Lin, E. and Lane, H. (2017). Machine learning and systems genomics approaches for multi-omics data. Biomarker Research , 5(1).

[6] Zhu, B., Song, N., Shen, R., Arora, A., Machiela, M., Song, L., Landi, M., Ghosh, D., Chatterjee, N., Baladandayuthapani, V. and Zhao, H. (2017). Integrating Clinical and Multiple Omics Data for Prognostic Assessment across Human Cancers. Scientific Reports , 7(1).

[7] Mirza, B., Wang, W., Wang, J., Choi, H., Chung, N. and Ping, P. (2019). Machine Learning and Integrative Analysis of Biomedical Big Data. Genes , 10(2), p.87.

[8] Kwon, M., Kim, Y., Lee, S., Namkung, J., Yun, T., Yi, S., Han, S., Kang, M., Kim, S., Jang, J. and Park, T. (2015). Integrative analysis of multi-omics data for identifying multi-markers for diagnosing pancreatic cancer.

[9] Chaudhary, K., Poirion, O., Lu, L. and Garmire, L. (2017). Deep Learning–Based Multi-Omics Integration Robustly Predicts Survival in Liver Cancer. Clinical Cancer Research , 24(6), pp.1248-1259.

[14] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, Serge Belongie, Class-Balanced Loss Based on Effective Number of Samples, CVPR 2019.

[15] Ye, L., Li, F., Song, Y., Yu, D., Xiong, Z., Li, Y., Shi, T., Yuan, Z., Lin, C., Wu, X., Ren, L., Li, X. and Song, L. (2018). Overexpression of CDCA7 predicts poor prognosis and induces EZH2-mediated progression of triple-negative breast cancer. International Journal of Cancer , 143(10), pp.2602-2613.