Predicting Survival Rate

Task 1:

1. Identify the dataset columns into nominal, categorical, continues etc. categories

As identified using the description provided with the dataset, I figured that following are the Numerical and Nominal features in the dataset

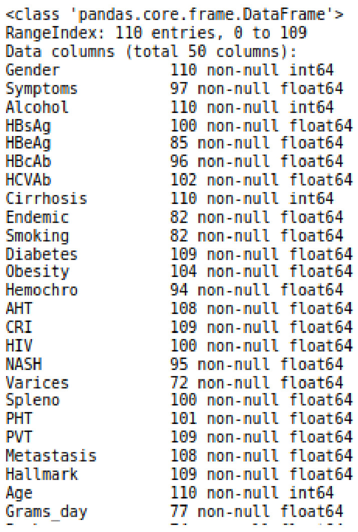
Nominal:
Gender, Symptoms, Alcohol, HbsAg, HbcAb, HCVAb, Cirrhosis, Endemic, Smoking, Diabetes, Obesity, Hemochro, AHT, CRI, HIV, NASH, Varices, Spleno, PHT, PVT, Metastasis, Hallmark

Numerical:
Age, Grams_day, Packs_year, PS, Encephalopathy, Ascites, INR, AFP, Hemoglobin, MCV, Leucocytes, Platelets, Albumin, Total_Bil, ALT, AST, GGT, ALP, TP, Creatinine, Nodule, Major_Dim, Dir_BilIron, Sat, Ferritin

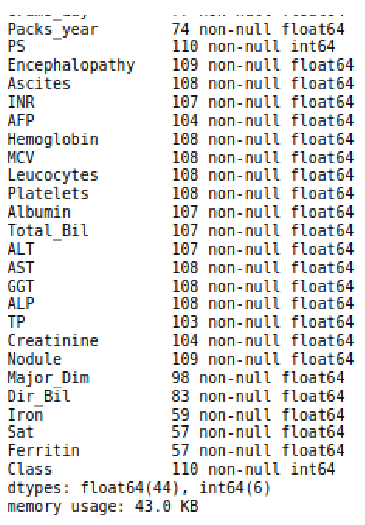
Out of the above numerical features, there are some that are ordinal like Ascites degree, Encephalopathy degree and Performance Status. I will be treating them the same as numerical data in this particular problem.

One feature – HbeAg has mostly NaN values or zero values so I will be dropping that column.

2. Use dataframe.info and dataframe.describe to get the insights about the data



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 110 entries, 0 to 109
Data columns (total 50 columns):
Gender            110 non-null int64
Symptoms          97 non-null float64
Alcohol           110 non-null int64
HBsAg             100 non-null float64
HBeAg             85 non-null float64
HBcAb             96 non-null float64
HCVAb             102 non-null float64
Cirrhosis         110 non-null int64
Endemic           82 non-null float64
Smoking           82 non-null float64
Diabetes          109 non-null float64
Obesity           104 non-null float64
Hemochro          94 non-null float64
AHT               108 non-null float64
CRI               109 non-null float64
HIV               100 non-null float64
NASH              95 non-null float64
Varices           72 non-null float64
Spleno            100 non-null float64
PHT               101 non-null float64
PVT               109 non-null float64
Metastasis        108 non-null float64
Hallmark          109 non-null float64
Age               110 non-null int64
Grams_day         77 non-null float64
```



```
Packs_year        74 non-null float64
PS                110 non-null int64
Encephalopathy    109 non-null float64
Ascites           108 non-null float64
INR               107 non-null float64
AFP               104 non-null float64
Hemoglobin        108 non-null float64
MCV               108 non-null float64
Leucocytes        108 non-null float64
Platelets         108 non-null float64
Albumin           107 non-null float64
Total_Bil         107 non-null float64
ALT               107 non-null float64
AST               108 non-null float64
GGT               108 non-null float64
ALP               108 non-null float64
TP                103 non-null float64
Creatinine        104 non-null float64
Nodule            109 non-null float64
Major_Dim         98 non-null float64
Dir_Bil           83 non-null float64
Iron              59 non-null float64
Sat               57 non-null float64
Ferritin          57 non-null float64
Class             110 non-null int64
dtypes: float64(44), int64(6)
memory usage: 43.0 KB
```

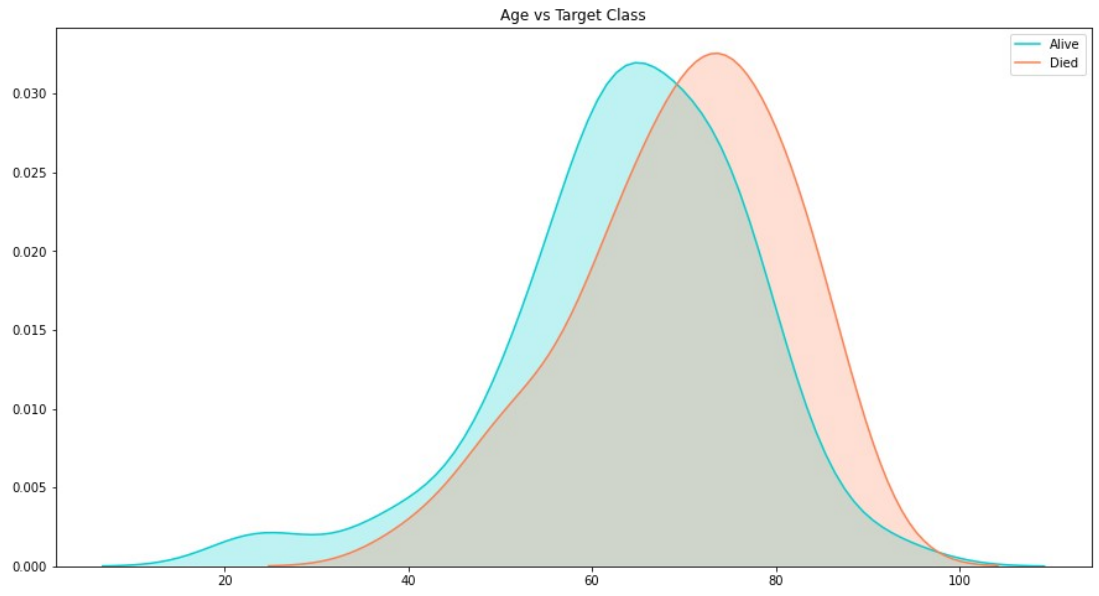
3. Find the number of null values for each columns

```
Gender             0
Symptoms          13
Alcohol            0
HBsAg             10
HBcAb             14
HCVAb              8
Cirrhosis          0
Endemic           28
Smoking           28
Diabetes           1
Obesity            6
Hemochro          16
AHT                2
CRI                1
HIV               10
NASH              15
Varices           38
Spleno            10
PHT                9
PVT                1
Metastasis         2
Hallmark           1
Age                0
Grams_day         33
Packs_year        36
PS                 0
Encephalopathy     1
Ascites            2
INR                3
AFP                6
Hemoglobin         2
MCV                2
Leucocytes         2
Platelets          2
Albumin            3
Total_Bil          3
ALT                3
AST                2
GGT                2
ALP                2
TP                 7
Creatinine         6
Nodule             1
Major_Dim         12
Dir_Bil           27
Iron              51
Sat               53
Ferritin          53
Class              0
```

4. Know about the patients

a. Find the oldest person: Person with Age: 93 years
b. Find the youngest person: Person with Age: 23 years
c. Find the average age group: 50-75 years
d. Find median age: 67.0
e. Find the relationship between the deaths and ages

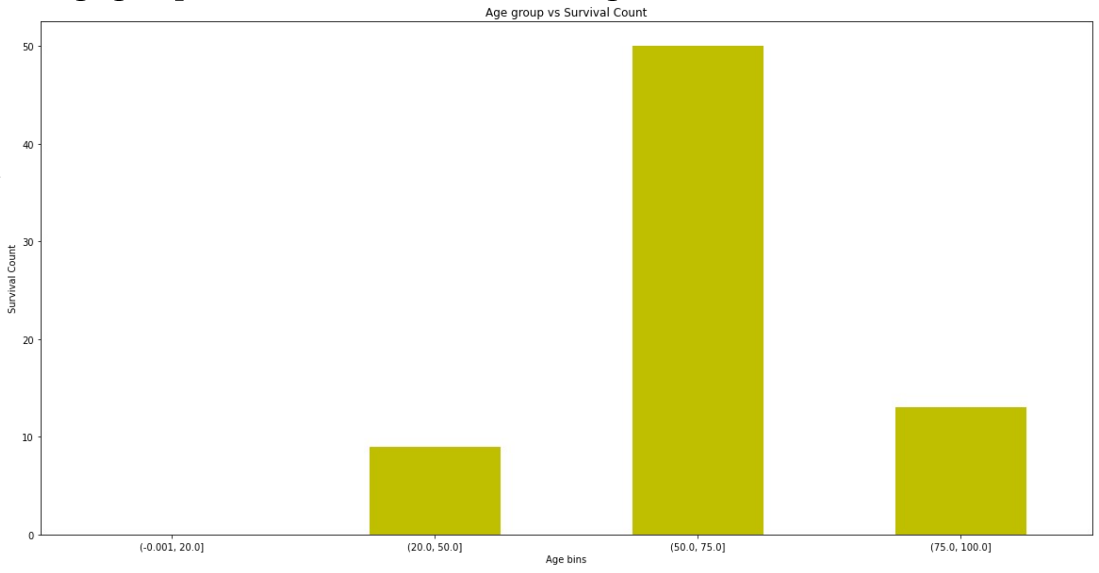
We can observe most



datapoints in dataset are from ages 50 to 75 and older people are more likely to not survive compared.

f. Find the age groups whose survival rate is the largest

Age group with highest



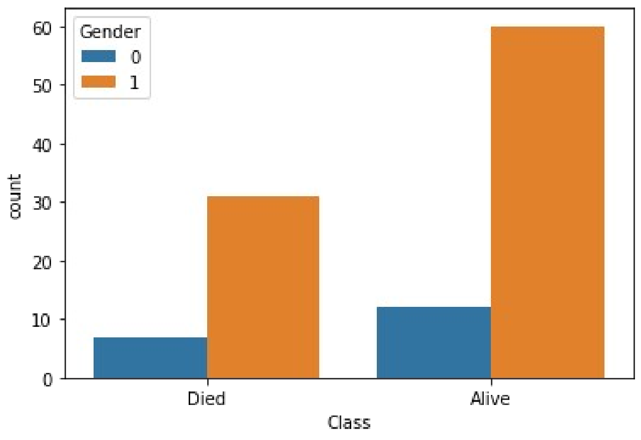
survival rate is age 50 – 75 years

g. Find similar relationships for at least 3-4 columns that you think can play a role in

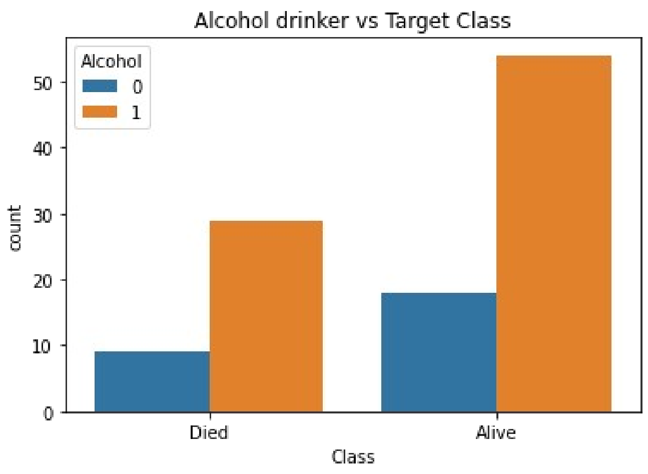
prediction

Gender:

We observe that Gender=1 which is Male has more number of cases and in turn has significantly more male persons alive and dead compared to Female gender.
Thus from this dataset we can say that men have more probability of dying of hcc

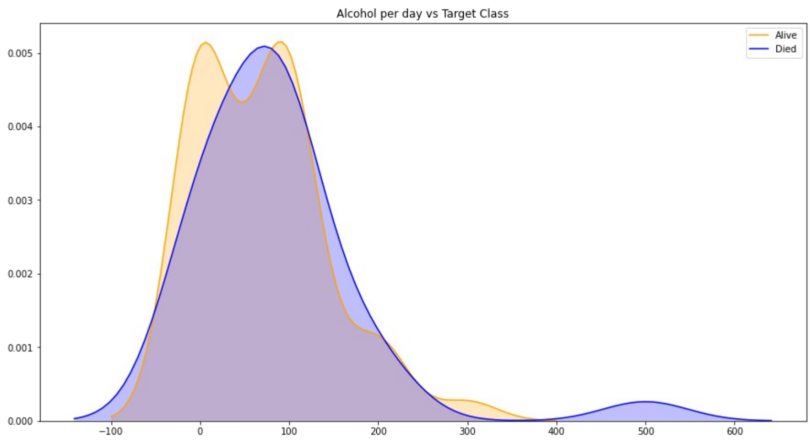


Alcohol Drinker or not:

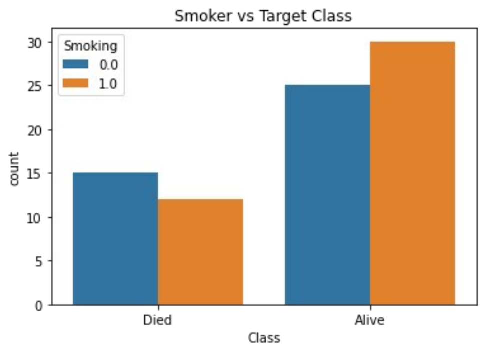


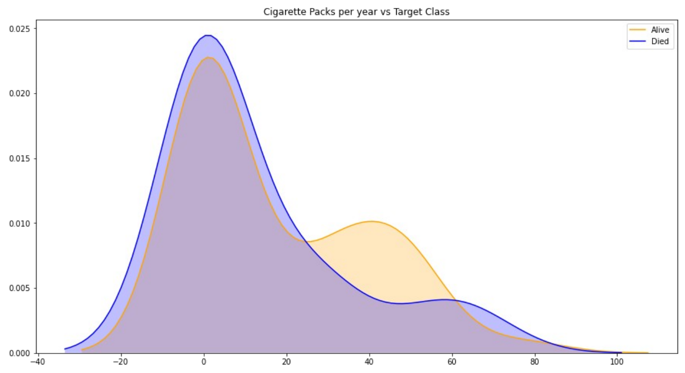
Alcohol consumption per day:

We can observe that alcohol consumption per day is directly related to deaths

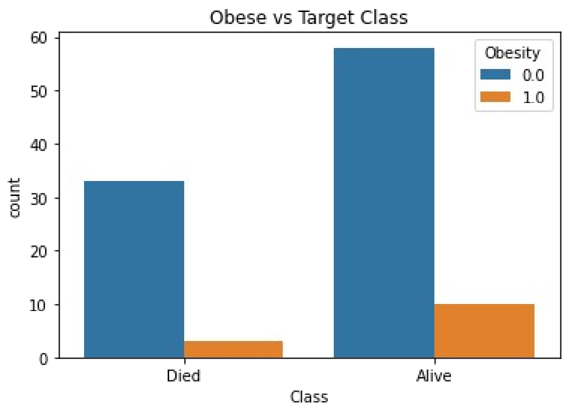


Smoking:
We can clearly observe that smoking habits are directly proportional to number of cases of HCC and number of deaths as well



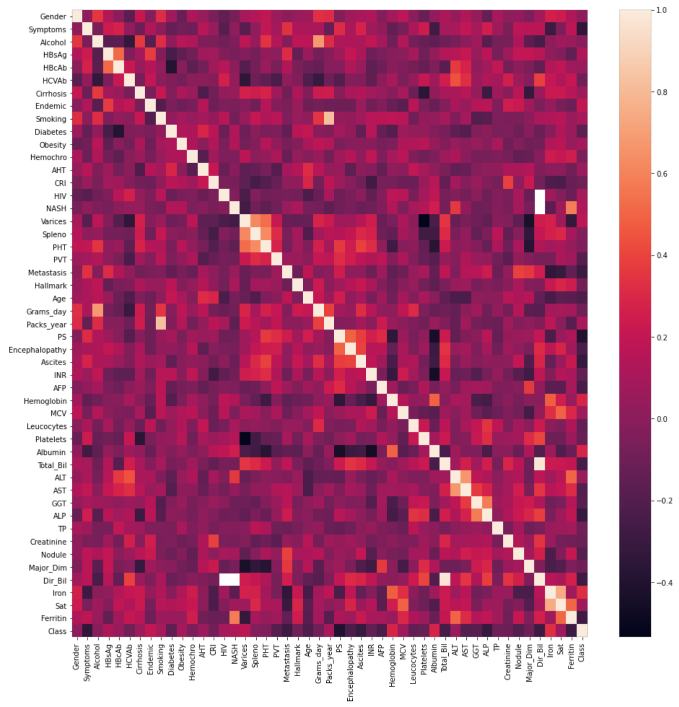


<u>Obesity:</u>

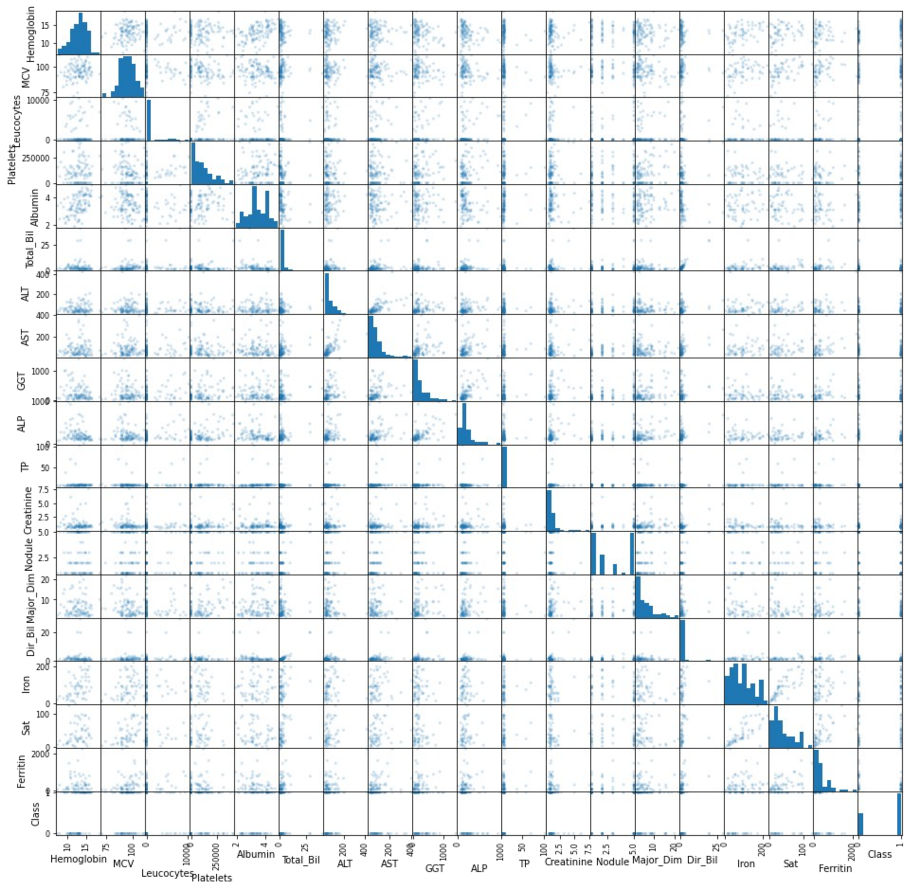


h. Get more visuals on data distributions
Use plotCorrelationMatrix

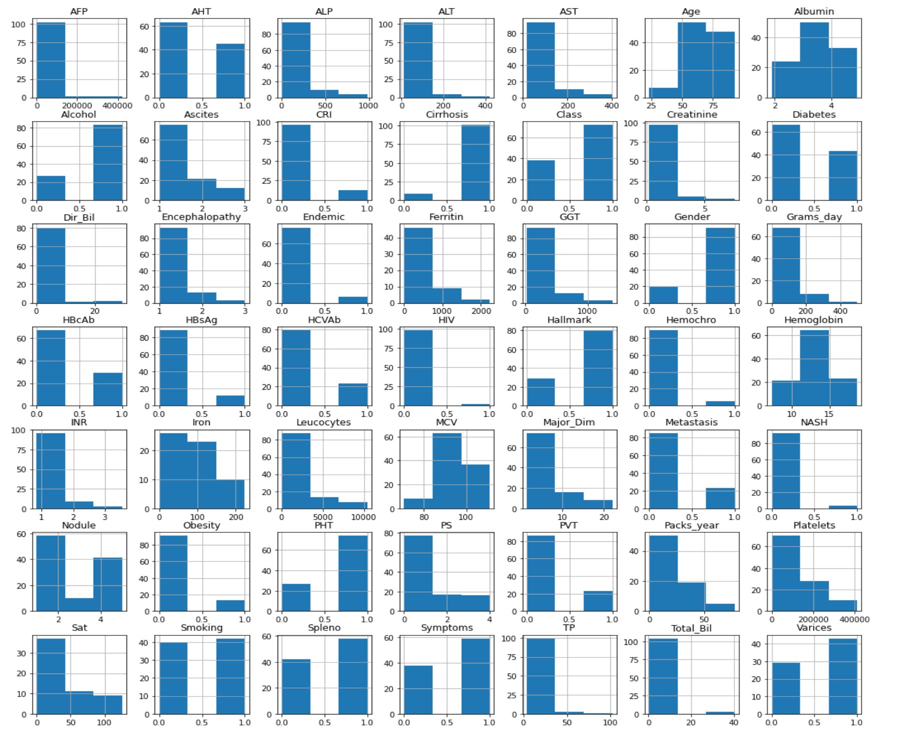
We can observe that the following are the columns with least correlation with Class column:
['Ascites','Encephalopathy','PS','Nodule','Age']



ii. plotScatterMatrix



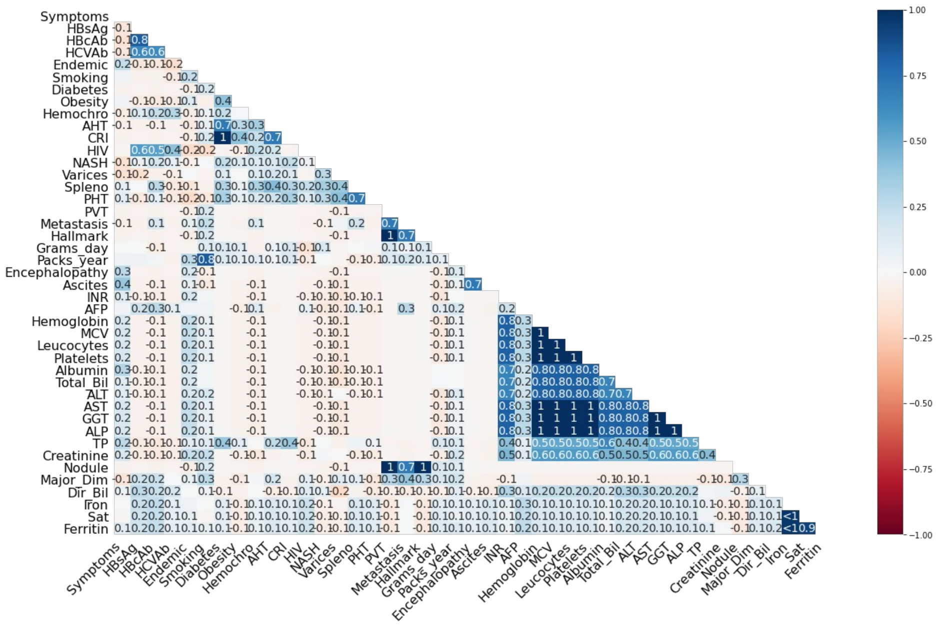
iii. PlotPerColumnDistribution



i. Find missing values

Get the count of missing values: 527 total missing values

Plot a heat map for missing values



j. Applying a different technique to handle missing values

As an additional technique, I also tested Imputing missing values using KNN. KNN is an algorithm that is useful for matching a point with its closest k neighbors in a multi-dimensional space. It can be used for data that are continuous, discrete, ordinal and categorical which makes it particularly useful for dealing with all kind of missing data.
The following table shows the results for each missing value replacement technique:

	Accuracy	**F1 score**	**Precision**	**Recall**
Drop NaN	0.7143	0.75	0.6429	0.9
Replace with zero	0.909	0.933	0.933	0.933
Replace with mean	0.7619	0.7826	0.9	0.4879
KNN Imputed	0.7142	0.8	0.8	0.8

k. Applying the feature scaling technique if you think it is required.
I will be using feature scaling technique- MinMaxScaler that scales and translates each feature individually in the given range on the training set, e.g. between zero and one as SVM will assume that all features are in the same range.
Other steps taken:

1. Dropped rows having null values for more than 10 features
2. Dropped columns having more than 90 rows as null values
3. Replaced NaN values for Nominal features with the Mode of those features

4. Replaced NaN values for Numerical features with the Mean of those features
5. Scaled x values using MinMaxScaler from range -1 to 1
6. Split data into training and testing dataset with test size as 20 %

l. Applying the regression models that you think is most suited for this problem.

I believe that SVM will be most suitable for this particular machine learning problem of classification of patients and predicting their survival.
I will be using sklearn.svm.SVC from scikit learn library. The objective of SVC (Support Vector Classifier) is to fit to the data provided and then returning a "best fit" hyperplane that divides, or categorizes our data. Once the hyperplane is ready, we can then feed more data to the classifier to predict the target class.
There are several hyperparameters used in creating the SVC model. Some of them are mentioned below:

1. C: Regularization parameter. The strength of the regularization is inversely proportional to C.
2. Kernel: Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable.
3. Degree: Degree of the polynomial kernel function ('poly').
4. Gamma: Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.
5. Shrinking: Whether to use the shrinking heuristic.
6. Probability: Whether to enable probability estimates.
7. Class_weight: Set the parameter C of class i to class_weight[i]*C for SVC.

With having so many parameters and their effect being so different, I was very confused whether to use a particular hyperparameter or not. If yes, what value should that parameter have.
I had to run my model 5-10 times before I realised that the possibilities are too many to experiment. I then discovered GridSearchCV.

sklearn.model_selection.GridSearchCV

GridSearchCV is used to make hyperparameter tuning easier. It takes in a dictionary that describes the parameters that could be tried on a model to train it. It then tries out all the combinations of parameters provided by us in the dictionary and returns the best parameter and what its value should be.
My parameter dictionary was:

```
 param_grid = [
  {'C': [0.001,0.01,0.1,1,10,100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [,0.01,0.001, 0.0001], 'kernel': ['rbf', 'poly']},
 ]
```

Result: {'C': 0.1, 'kernel': 'linear'}
With this, I got my best parameter for kernel as 'linear' and c as 0.1.
I used the following parameters finally:
kernel='linear'
probability=True (to use probability estimates while deciding on target class value)
C=0.1
class_weight='balanced' (to balance out the 0 and 1 values in target class attribute)

With the above model I got the following results:

Accuracy	**F1 Score**	**Precision**	**Recall**	**RMSE**

0.7619	0.8275	0.8	0.8571	0.4879

Here after I used this model to predict for the hcc-test.csv dataset and uploaded it to Kaggle to get a score of 0.50452.

m. At least one of the models used to compute should be your own implementation using NumPy.

For this, I implemented Logistic Regression as it can be used for classification problems as well.

Logistic regression uses an equation as the representation shown below. Input values (x) are combined linearly using weights or coefficient values to predict an output value (y).
Below is logistic regression equation:

y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient that must be learned from your training data.

Please refer to the code in Jupyter Notebook for the code for Logistic regression.

With the code I implemented myself, I only got an accuracy score of 0.3214.

Task 2:
Using hcc-data-complete-balance.csv

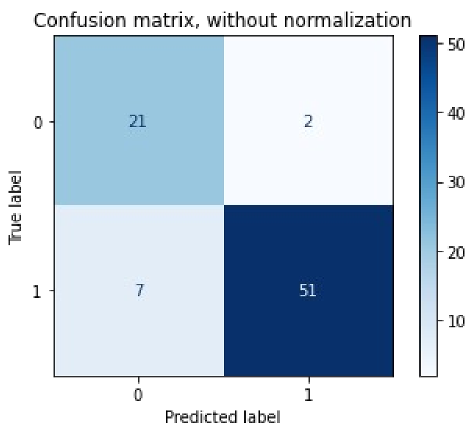
For this dataset, I repeated some of the steps I had implemented for Task 1 such as:

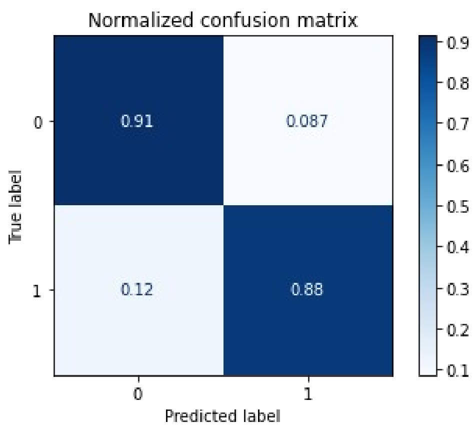
7. Dropped rows having null values for more than 10 features
8. Dropped columns having more than 90 rows as null values
9. Replaced NaN values for Nominal features with the Mode of those features
10. Replaced NaN values for Numerical features with the Mean of those features
11. Scaled x values using MinMaxScaler from range -1 to 1
12. Split data into training and testing dataset with test size as 20 %
13. Created an SVC classifier with following hyperparameters: kernel='linear', probability=True,C=0.1, class_weight='balanced'
14. Fit the training dataset on to the model
15. Predicted on test dataset
16. Calculated metrics

The following is the result I got:

Accuracy	**F1 Score**	**Precision**	**Recall**	**RMSE**
0.6666	0.6666	0.6363	0.7	0.5773

Confusion Matrix:





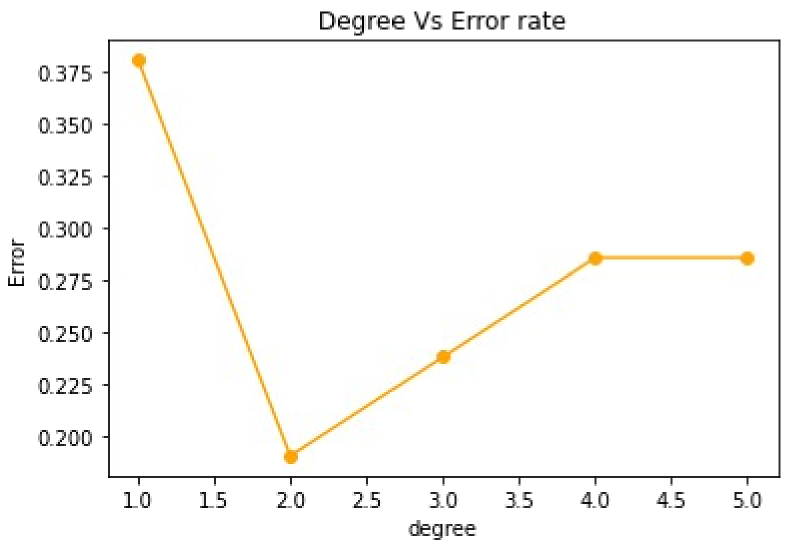
When compared to the results in we can observe that the metrics were better in Task 1 as compared to Task 2.

Task 3:
Applying Polynomial Feature Transform

a. Apply feature transform on the features used in task 1

After applying feature transform using PolynomialFeatures, I calculated accuracy for degrees from 1-5. After plotting the error rate which is 1-Accuracy, this is the graph I got.

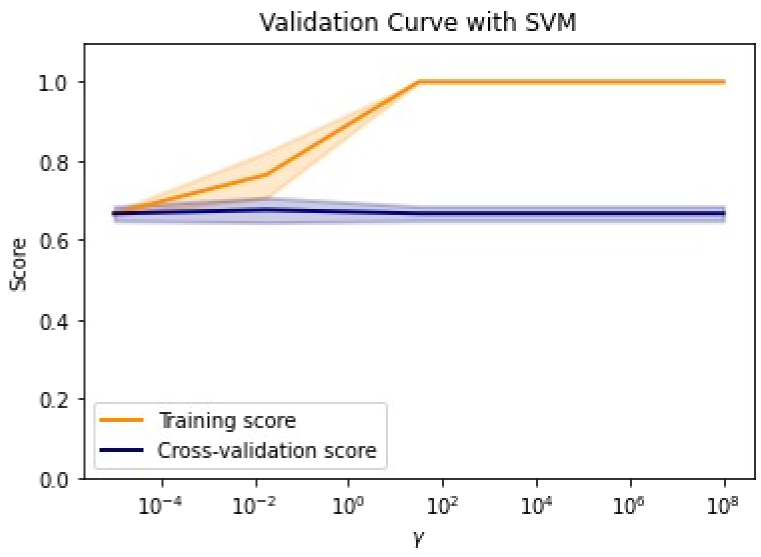
As observed, error rate is lowest, that is, accuracy is highest with Degree = 2



Applying feature transform increased the accuracy as compared to Task1 and Task2.

b. Can you identify if your model is underfitting or overfitting?

For very low values of gamma, we can observe that both the training score and the validation score are at 0.6. This is not necessarily underfitting. Medium values of gamma will result in high values for only Training score while the cross validation score remains plateaued. If gamma is too high, the classifier will overfit, which means that the training score is good but the validation score is lower and same as with a lower gamma.


Validation Curve with SVM
Score
Training score
Cross-validation score
10^{-4}
10^{-2}
10^{0}
10^{2}
10^{4}
10^{6}
10^{8}
γ