Breast Cancer Survival Prediction

Introduction

The most important information that usually patients diagnosed with breast cancer want to know is how long they will survive. Patients with different criteria when they have been digones can have very different survival rates. The intention of this study is to predict survival rate for patients with different diagnostic criteria.

Dataset

Dataset that has been used here is Seer Breast Cancer Data, This dataset of breast cancer patients was obtained from the 2017 November update of the SEER Program of the NCI, which provides information on population-based cancer statistics. The dataset involved female patients with infiltrating duct and lobular carcinoma breast cancer diagnosed in 2006-2010.

Project Steps:

1. **Data Wrangling** - This is where we try to get a raw understanding of what the data looks like and perform transformations and operations on it to bring it to a more usable state
2. **Exploratory Data Analysis** - Here, we try to understand the nature and distribution of our data by plotting visualizations of different features in the dataset. This can give useful insights prior to the modeling phase.
3. **Preprocessing** - The stage where we perform any other operations on the data, like scaling or normalizing, before it is ready to be fed into a machine learning model.
4. **Modeling** - The most important part of course, where we actually try fitting different models onto the preprocessed dataset and evaluate which model performs the best prediction.

Data Wrangling

The good thing is, this dataset is clean but still needs some extra clean up and preprocessing before it's ready to be fitted to a model.

Each sample of data consists of age, race, marital status, t stage, n stage,a stage, tumor size, estrogen status, progesterone status,regional nodes examined, regional nodes positive and status of the patient.

So clearly, this is a very rich dataset and there is plenty of relevant information our models can use while training. However, the challenge is The dataset has a few features that are continuous and the rest are categorical variables.

1.Data Cleaning

As investigated there are 10 categorical columns and 5 numerical columns. Also there seems to be no null value for all the records.

Another step to clean the data is removing the unnecessary text from the records. For example, the “Grade” feature has the below records:

1. 'Moderately differentiated; Grade II'
2. 'Poorly differentiated; Grade III'
3. 'Well differentiated; Grade I'
4. 'Undifferentiated; anaplastic; Grade IV'

We get enough information by just having the grades as below:

1. Grade II
2. Grade III
3. Grade I
4. Grade IV

To be able to do proper statistical analysis we are going to convert the categorical columns to numeric data

2.Convert Data to Dummy Data

Used two methods to convert the data to numeric in this step. For the features “one hot encoding” was used as below with dropped one column to avoid Dummy Variable Trap and multicollinearity.

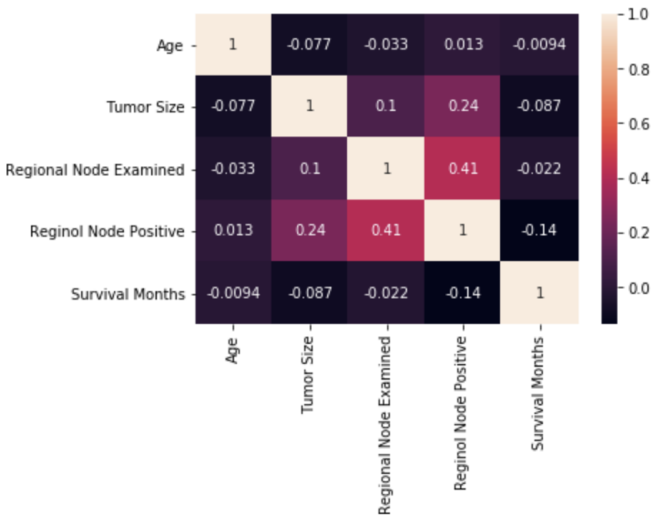
- features = pd.get_dummies(features,drop_first = True)

Encoded the target column using LabelEncoder as below:

- target = pd.DataFrame(le.fit_transform(target),columns = ['Status'])

3.Correlation between features

To visualize the correlation between the continuous features, Heatmap was used as below:

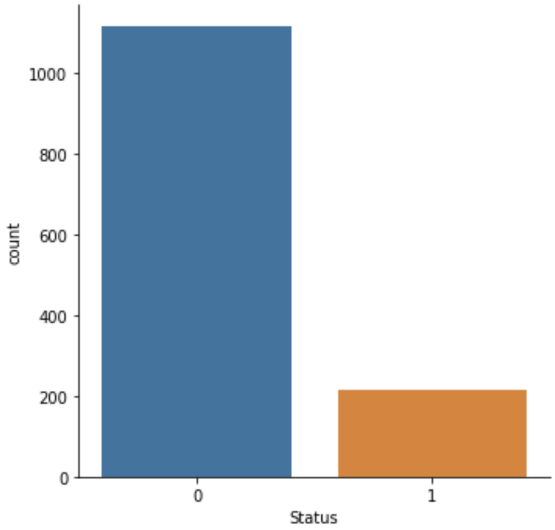


As seen in the above map there are some correlations between the features such as Regional Node Examined with Regional Node Positive and Tumor size. There will be deeper investigation in the next section.

4.Balancing the Data

Imbalanced classification is the problem of classification when there is an unequal distribution of classes in the training dataset. Let's investigate how our target data look:

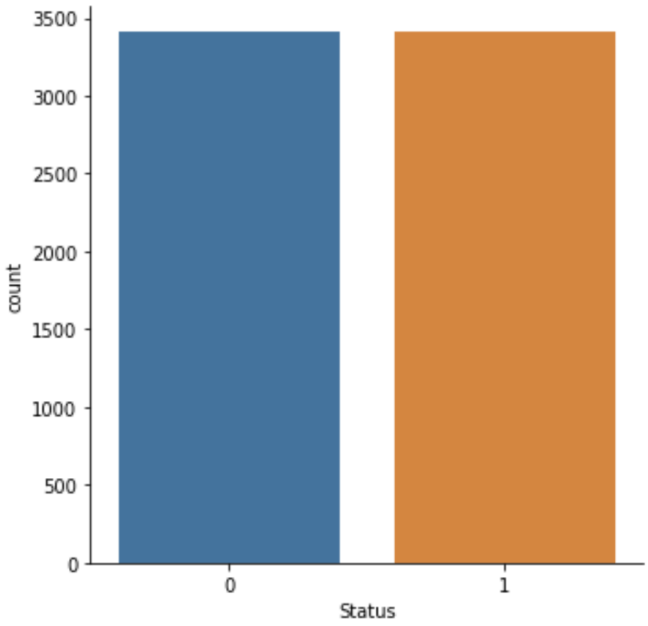
- sns.catplot(x='Status',data=y_test,datakind="count");



Obviously our target data is not balanced and the number of alive records are almost 5th times the records with dead status. To balance the data SMOTE technique was used:

- oversample = SMOTE()
- X_smote, y_smote = oversample.fit_resample(X, y)
- sns.catplot(x='Status',data=y_smote,kind="count");

After resampling(over sample) our data, records are balanced and the targets are equally disputed between Live and Dead:

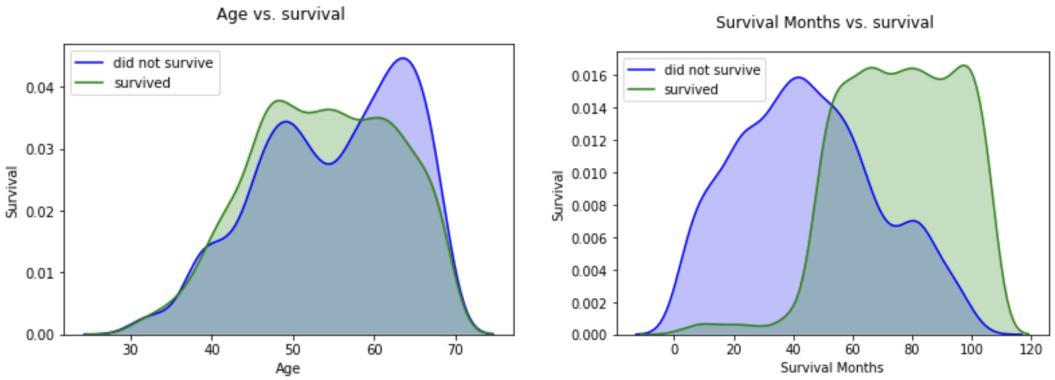


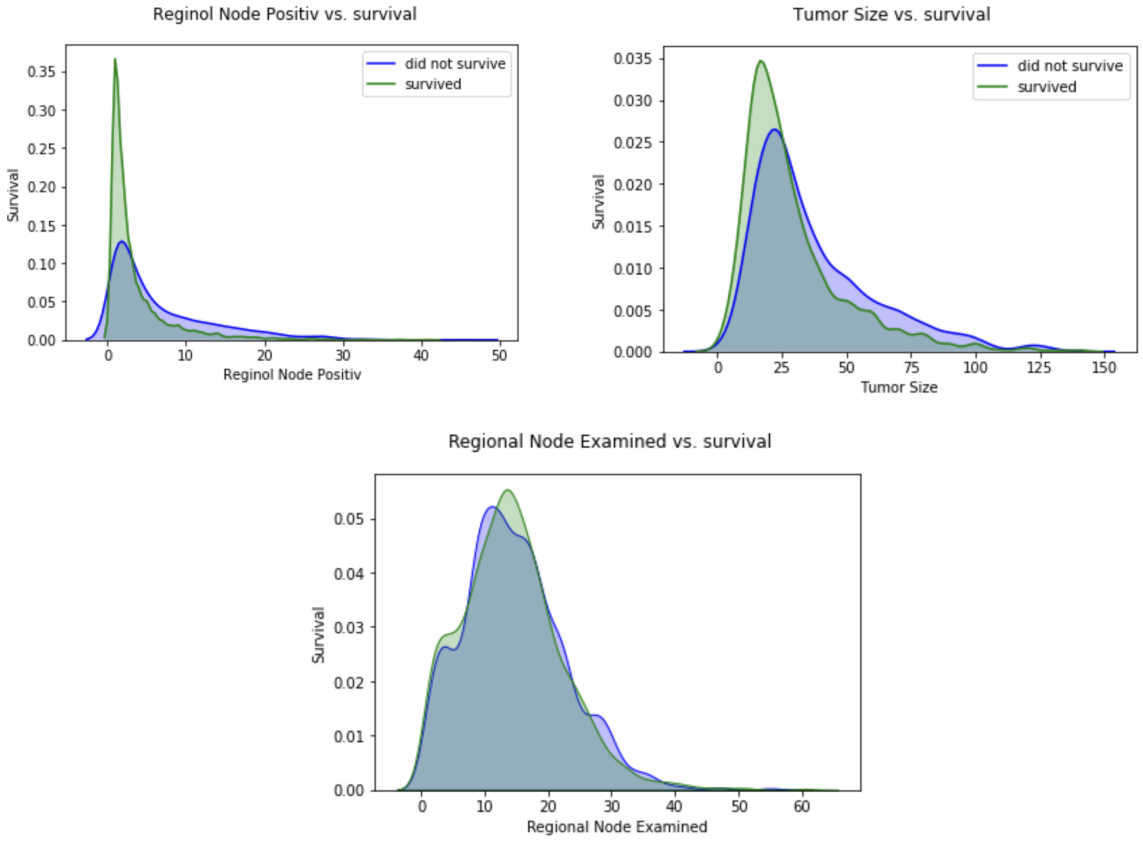
Exploratory Data Analysis

In this section we will dive deeper to explore the data. We will explore the correlation of continuous features and the relationship of the categorial features with the target.

1. Continuous data

There are 5 continuous features including "Tumor Size","Regional Node Positive"'Regional Node Examined',"Age" and ”Survival Months” . As the target is still categorical we used sns.kdeplot to plot each feature against the values of the target.

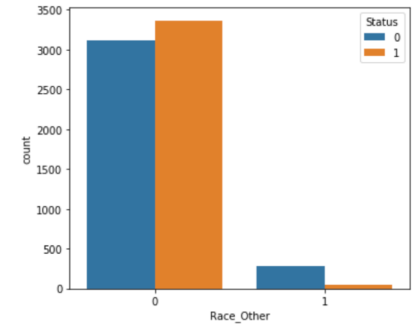


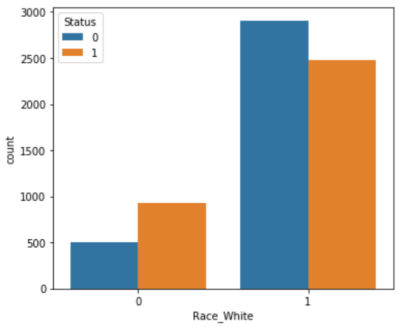


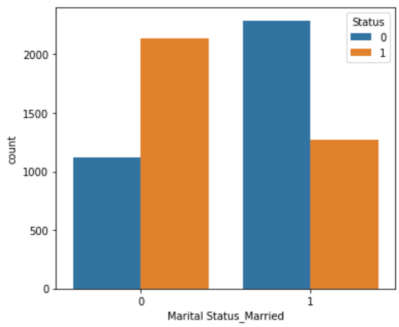
As seen in the plots, there is not an obvious correlation or relation between the features and the target except for “Survival Months”, which obviously is longer for surviving patients and much less for the dead patients.

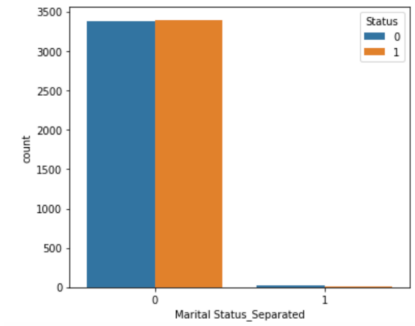
2. Categorical data

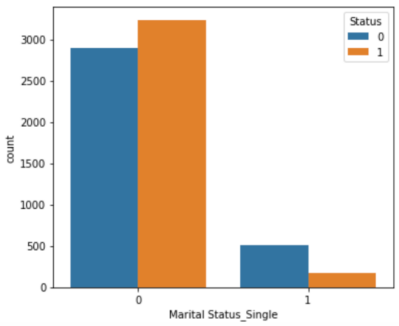
The rest of the features are categorical, to compare the relationship we plot the count bars for each target value.

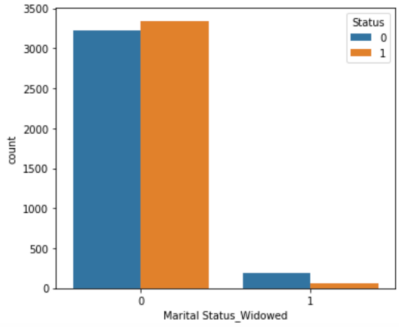

Status
0
1
count
Race_Other

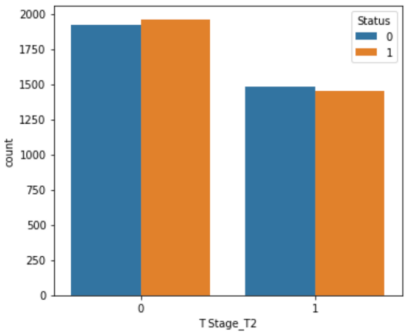

Status
0
1
count
Race_White

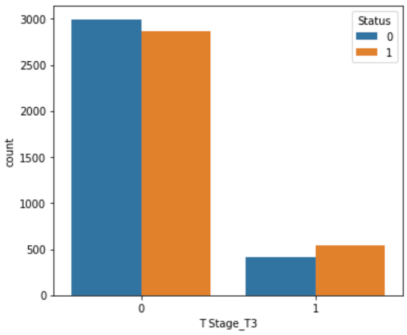

Status
0
1
count
Marital Status_Married

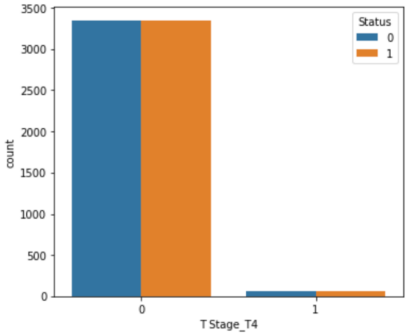

Status
0
1
count
Marital Status_Separated

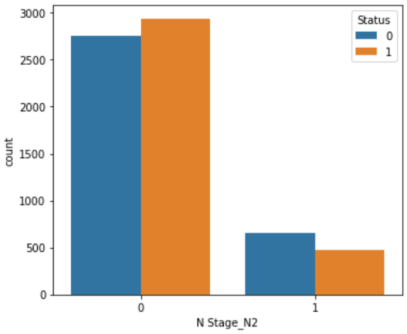

Status
0
1
count
Marital Status_Single

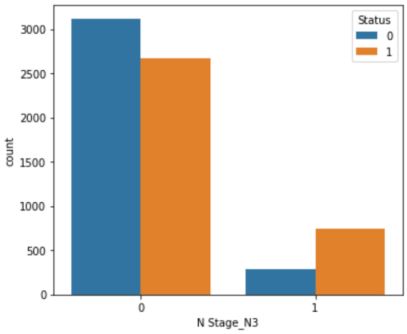

Status
0
1
count
Marital Status_Widowed

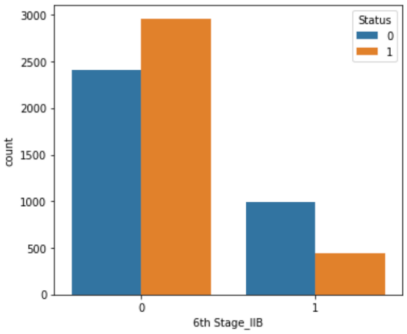

Status
0
1
count
T Stage_T2

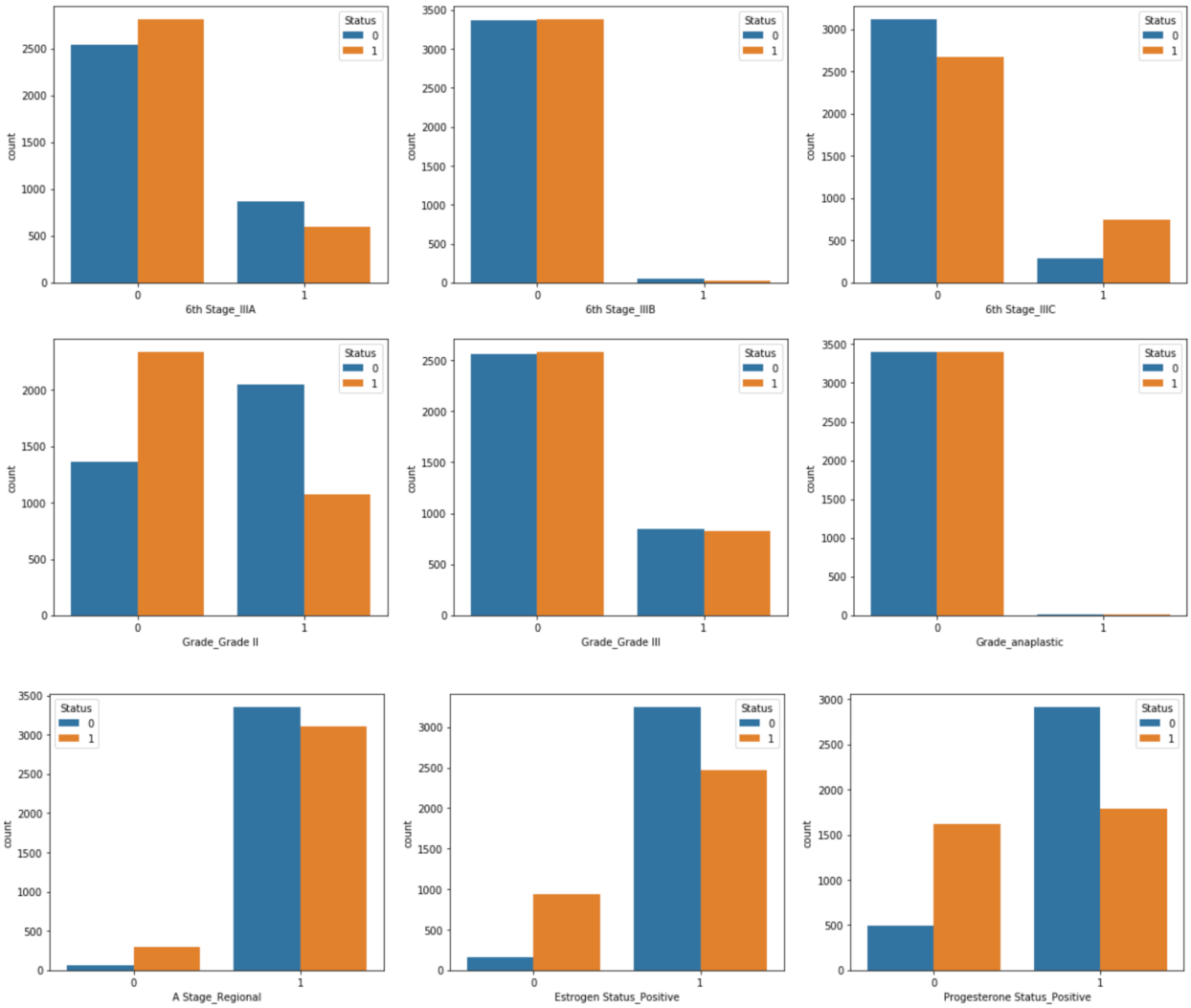

Status
0
1
count
T Stage_T3


Status
0
1
count
T Stage_T4


Status
0
1
count
N Stage_N2


Status
0
1
count
N Stage_N3


Status
0
1
count
6th Stage_IIB



As seen in the above graphs some features show some contributions to the target classification. For example, married people could live longer, or people with grade 2 cancer live longer and that could be that their grade is higher and they will not survive so long. Also Being Estrogen or Progesterone positive could lead to a longer life.

Preprocessing

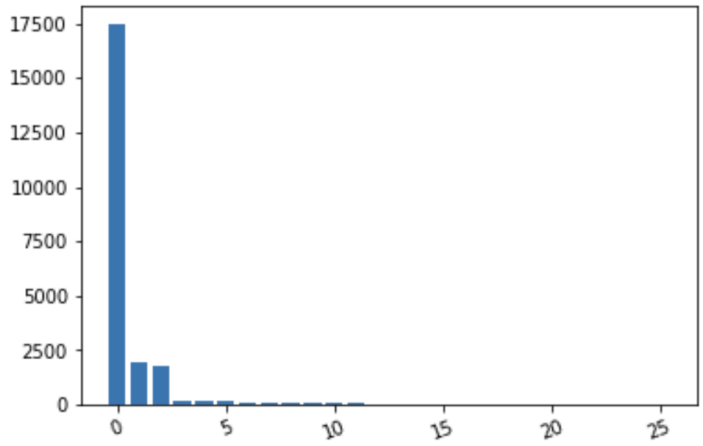
1.Feature Selection

There are two popular feature selection techniques that can be used for categorical input data and a categorical target variable, which are Chi-Squared Statistic and Mutual Information Statistic.

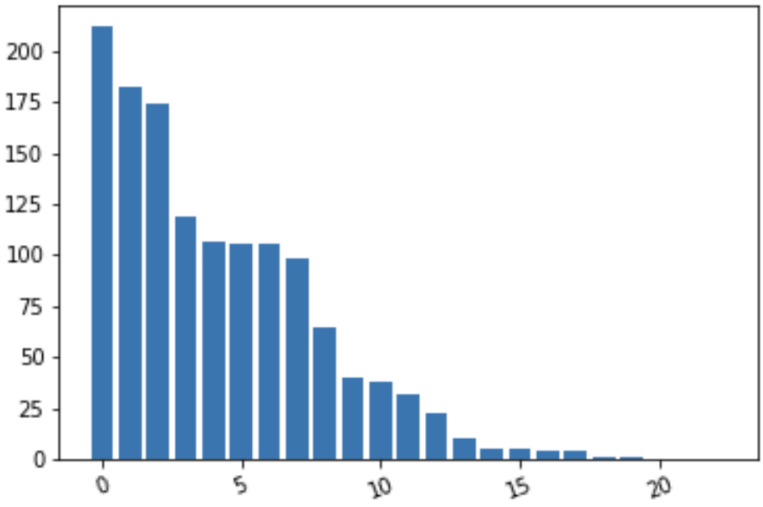
We will use the Chi-Squared Statistic.

- fs = SelectKBest(score_func=chi2, k='all')
- fs.fit(X_train, y_train)

	Name	Score
0	Survival Months	17478.803015
1	Tumor Size	1967.014049
2	Reginol Node Positive	1750.902463
3	Grade_Grade II	212.033694
4	Marital Status_Married	182.786460
5	Progesterone Status_Positive	173.888837
6	6th Stage_IIB	118.987770
7	Marital Status_Single	106.100394
8	N Stage_N3	105.947388
9	6th Stage_IIIC	105.947388
10	Race_Other	98.157095
11	Estrogen Status_Positive	64.603892
12	Age	40.265232
13	Marital Status_Widowed	37.917861
14	6th Stage_IIIA	31.855896
15	Race_White	22.984368
16	N Stage_N2	10.651684
17	A Stage_Regional	5.579573
18	T Stage_T3	5.100155
19	Marital Status_Separated	4.193361
20	Regional Node Examined	3.945062
21	6th Stage_IIIB	1.389694
22	T Stage_T4	1.146433
23	Grade_anaplastic	0.243042
24	T Stage_T2	0.155767
25	Grade_Grade III	0.131553



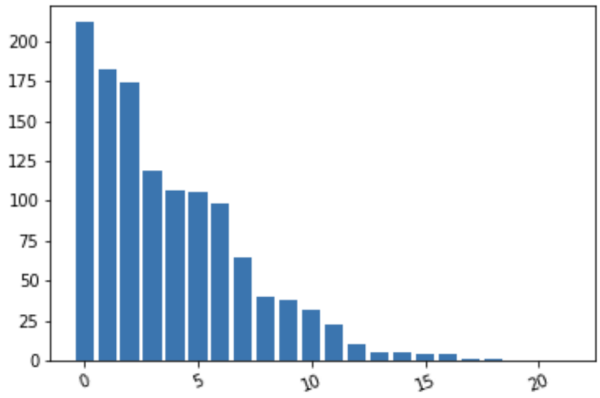
Let's drop survival month and tumor size and positive node as they have very high impact on the target data and explore the rest features



As seen above these features are the most important ones according to our KBest model:

1. Grade_Grade II
2. Progesterone Status_Positive
3. N Stage_N3
4. 6th Stage_IIIC
5. Marital Status_Married

As discovered in data wrangling T stage,N stage and 6th stage had correlation with each other. Therefore we can just select either N Stage_N3, 6th Stage_IIIC and drop one. I will drop N Stage_N3.



As seen above these features are the most important ones according to our new KBest model:

1. Survival Months
2. Tumor Size

3. Regional Node Positive
4. Grade_Grade II
5. Progesterone Status_Positive
6. 6th Stage_IIIC
7. Marital Status_Married
8. Marital Status_Single
9. Race_Other
10. Estrogen Status_Positive

2. Split the data to test and train

For the step which is fitting the data to the models, we need to split the data to train and test as below:

- X_train, X_test, y_train, y_test = prepare_train_test(df,'Status')
- print('X train shape is:{}'.format(X_train.shape))
- print('y train shape is:{}'.format(y_train.shape))
- print('X test shape is:{}'.format(X_test.shape))
- print('y test shape is:{}'.format(y_test.shape))

Also we need to scale the continuous data before applying the model.

3. Scale the data

We use the MinMaxScaler to scale our data as below:

- scaler = MinMaxScaler(feature_range=(0, 5))
- X_train_scaled = scaler.fit_transform(X_train)
- X_test_scaled = scaler.transform(X_test)

After scaling the data , we will apply the same steps to the selected data, which is data with top selected data,

- df_feature_selection = df[['Survival Months','Tumor Size','Reginol Node Positive','Grade_Grade II','Progesterone Status_Positive','6th Stage_IIIC','Marital Status_Married','Estrogen Status_Positive','Status']]
- X_train, X_test, y_train, y_test = prepare_train_test(df_feature_selection,'Status')
- scaler = MinMaxScaler(feature_range=(0, 5))
- X_train_scaled = scaler.fit_transform(X_train)
- X_test_scaled = scaler.transform(X_test)

Modeling

1.Explore different models with default values

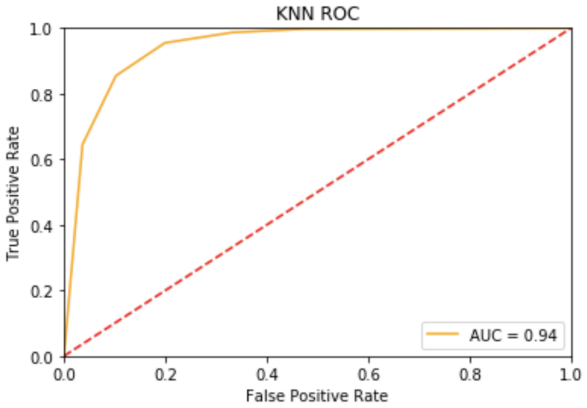
There are 5 models that we will investigate

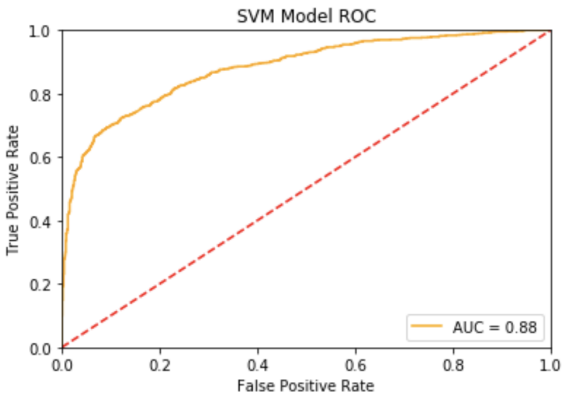
1. Logistic Regression
2. Decision Tree Classifier
3. Random Forest
4. KNeighborsClassifier
5. Support Vector Machine

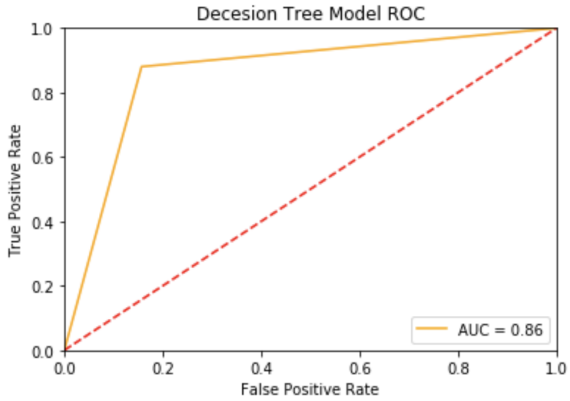
1.1 With all available data

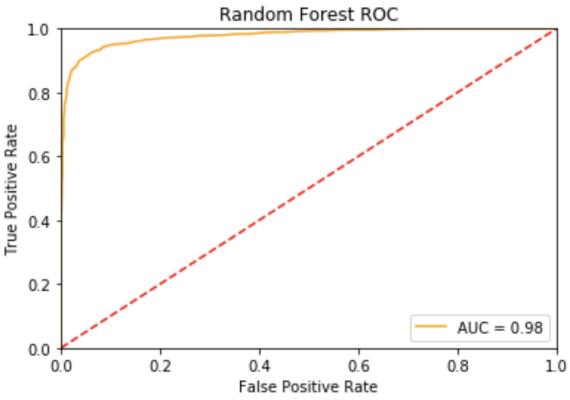
We will use the default models just to get the idea how they work and then go deeper to each model. Also in the section below we will use all the available data.

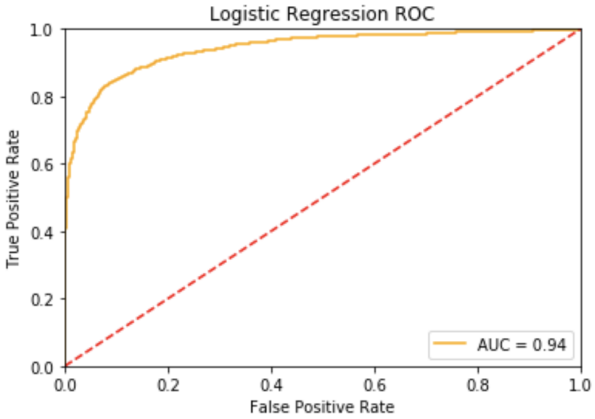
```
log_reg_model = LogisticRegression()
apply_mode(bc_data_X_train,bc_data_y_train,bc_data_X_test,bc_data_y_test,log_reg_model,'Logistic Regression')

rf_model = RandomForestClassifier()
apply_mode(bc_data_X_train,bc_data_y_train,bc_data_X_test,bc_data_y_test,rf_model,'Random Forest')

knn_model = KNeighborsClassifier()
apply_mode(bc_data_X_train,bc_data_y_train,bc_data_X_test,bc_data_y_test,knn_model,'KNN')

dt_model = DecisionTreeClassifier()
apply_mode(bc_data_X_train,bc_data_y_train,bc_data_X_test,bc_data_y_test,dt_model,'Decesion Tree Model')

SVM_model = SVC(probability=True)
apply_mode(bc_data_X_train,bc_data_y_train,bc_data_X_test,bc_data_y_test,SVM_model,'SVM Model')
```





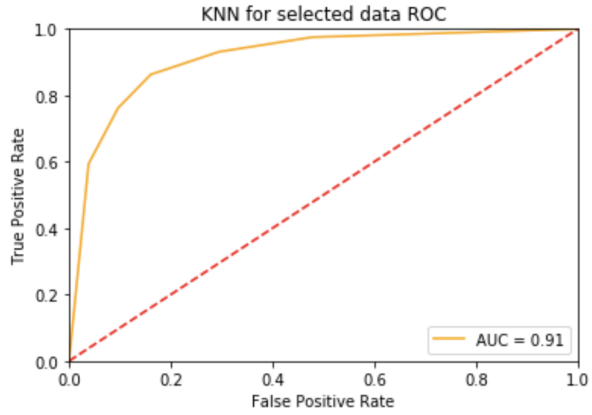


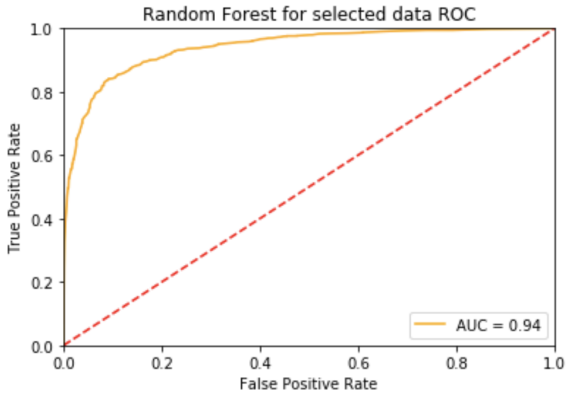


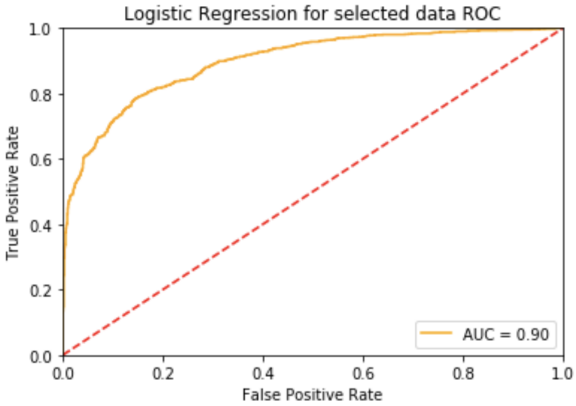


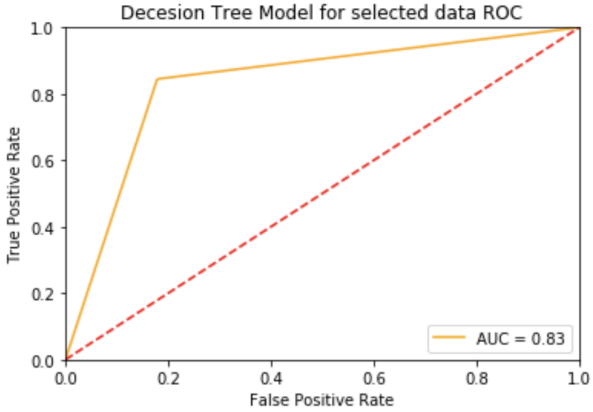
1.2 With selected features

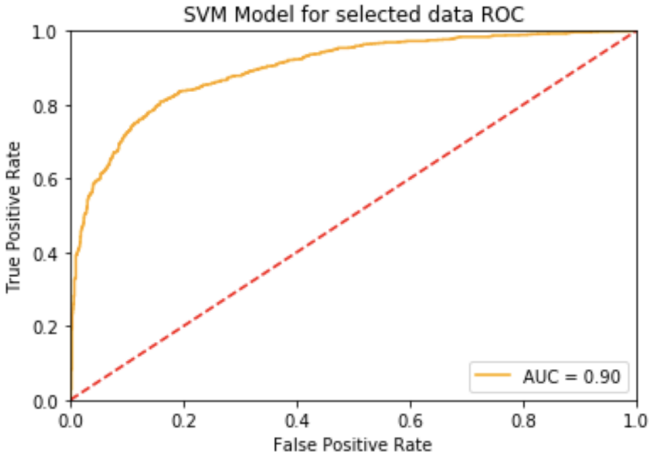
Now lets run the same classifiers on selected data with less features as below: ('Grade_Grade II', 'Progesterone Status_Positive', '6th Stage_IIIC', 'Marital Status_Married', 'Estrogen Status_Positive')


KNN for selected data ROC
True Positive Rate
False Positive Rate
AUC = 0.91


Random Forest for selected data ROC
True Positive Rate
False Positive Rate
AUC = 0.94


Logistic Regression for selected data ROC
True Positive Rate
False Positive Rate
AUC = 0.90


Decesion Tree Model for selected data ROC
True Positive Rate
False Positive Rate
AUC = 0.83


SVM Model for selected data ROC
True Positive Rate
False Positive Rate
AUC = 0.90

1.3 Summary of the results

Now lets see the result for all the available data

Model	Deafualt values	Accuracy	F1-Score	Elapsed time(s)
Logistic regression	Solver:lbfgs, Penalty:l2	0.87	0.87	0.0413
Random Forest	n_estimators=100,criterion=gini	0.93	0.93	0.0366
KNN	n_neighbors=5 ,algorithm='auto', leaf_size=30,metric='minkowski'	0.88	0.89	0.0412
Decesion Tree	criterion='gini'	0.86	0.86	0.0105
SVM	C=1.0, kernel='rbf', degree=3, gamma='scale'	0.80	0.78	2.3361

Now lets see the result for fewer selected data

Model	Deafualt values	Accuracy	F1-Score	Elapsed time(s)
Logistic regression	Solver:lbfgs, Penalty:l2	0.82	081	0.0110
Random Forest	n_estimators=100,criterion=gini	0.87	0.87	0.0141
KNN	n_neighbors=5 ,algorithm='auto', leaf_size=30,metric='minkowski'	0.85	0.85	0.0420
Decesion Tree	criterion='gini'	0.82	0.83	0.0015
SVM	C=1.0, kernel='rbf', degree=3, gamma='scale'	0.82	0.81	1.1410

Different classification models with default values, while using all the data or selected data gave us different results. As seen all models have very close Accuracy and F1-score when using all the data. Elapsed time is much higher in the SVM model, around 2.33 seconds.

When decreasing the columns and used some specific features('Grade_Grade II', 'Progesterone Status_Positive', '6th Stage_IIIC', 'Marital Status_Married', 'Estrogen Status_Positive') that have most of the data's property, the accuracy declined in all models, but as expected the elapsed time for running and predicting has decreased a lot.

In conclusion, as the data is not so huge and the elapsed time does not seem to be a problem for the amount of the data we have, and the accuracy is better with all the data, we will use all the data for deeper analysis.

2.Explore different models with hyper parameter

2.1 Model Exploration

Model 1: Logistic Regression

```
param_grid = [{'C':[100, 10, 1.0, 0.1, 0.01]}]
model_1 = LogisticRegression(max_iter = 5000)
apply_gridSearch(model_1,param_grid,bc_data_X_train,bc_data_y_train,'Logistic Regression')
```

Model 2: Decision Tree Classifier

```
param_grid = {'criterion':['gini','entropy'],'max_depth':np.arange(10,100,10),'max_features': ['auto', 'sqrt', 'log2'
model_2 = DecisionTreeClassifier()
apply_gridSearch(model_2,param_grid,bc_data_X_train,bc_data_y_train,'Decision Tree')
```

Model 3: Random Forest Classifier

```
param_grid = {'criterion':['gini','entropy'],'n_estimators':np.arange(10,200,50),'max_features': ['auto', 'sqrt', 'log2'
model_3 = RandomForestClassifier()
apply_gridSearch(model_3,param_grid,bc_data_X_train,bc_data_y_train,'Random Forest Classifier')
```

Model 4: KNeighbors Classifier

```
param_grid = {'n_neighbors':np.arange(3,50),'algorithm': ['auto','ball_tree','kd_tree','brute']}
model_4 = KNeighborsClassifier()
apply_gridSearch(model_4,param_grid,bc_data_X_train,bc_data_y_train,'KNeighbors Classifier')
```

Model 5: Support Vector Machine

```
param_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel': ['rbf', 'sigmoid']}
model_5 = SVC()
apply_gridSearch(model_5,param_grid,bc_data_X_train,bc_data_y_train,'SVM')
```

2.2 Results summary:

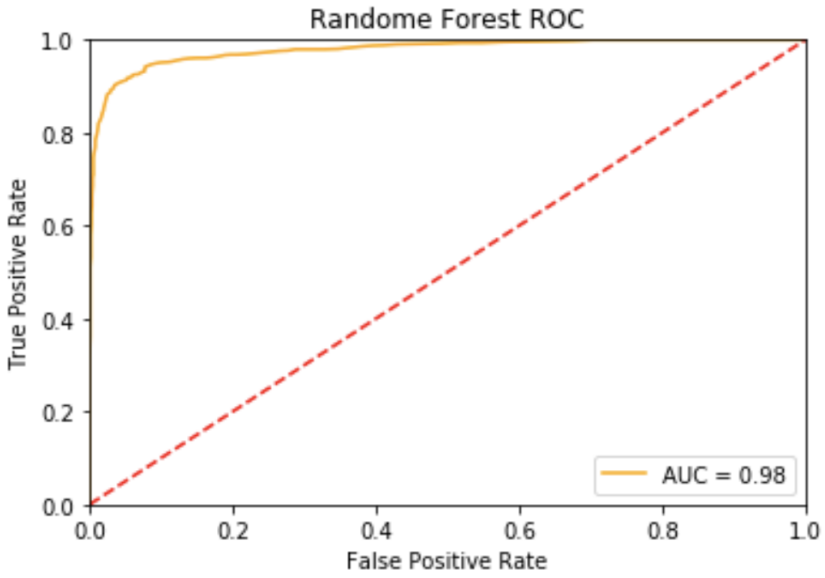
Model	Best paramter	Classifier Score	Grid Search Elapsed time(s)
Logistic regression	{'C': '10'}	0.85	82.92
Decesion Tree	{'criterion': 'entropy', 'max_depth': 80, 'max_features': 'sqrt'}	0.85	0.90
Random Forest	{'criterion': 'gini', 'max_features': 'log2', 'n_estimators': 110}	0.92	9.3
KNN	{'algorithm': 'auto', 'n_neighbors': 3}	0.86	151
SVM	{'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}	0.91	93.9

According to the different models' score result, Random Forest and SVM have the highest score equal to 92% AND 91%. Let's compare the performance of these models with the selected parameter.

3. Selected Models

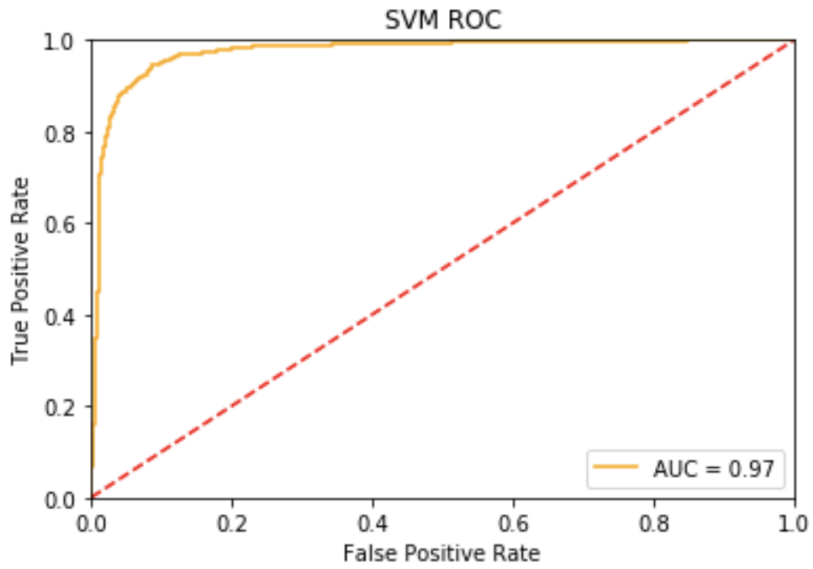
3.1 RandomForestClassifier:

```
rf_model = RandomForestClassifier(criterion = 'gini', n_estimators= 110,max_features = 'log2')
apply_mode(bc_data_X_train,bc_data_y_train,bc_data_X_test,bc_data_y_test,rf_model,'Randome Forest')
```



3.2 SupportVectorClassification.:

```
svm_model = SVC(probability=True, C = 10, gamma =0.01, kernel ='rbf')
apply_mode(bc_data_X_train,bc_data_y_train,bc_data_X_test,bc_data_y_test,svm_model,'SVM')
```



3.3 Summary of the results and Conclusion

Model	F1-score	AUC	Overal Elapsed time	Fitting Time	Predicting time
Random Forest	93%	98%	0.31	0.28	0.03
SVM	93%	97%	2.91	2.79	0.11

After running the models and comparing the result for predicting the test data we see slightly better AUC (98%) for RF compared to SVM which is 97%. They both have an equal (93%) score. Also if we consider the performance of the models, Decision Tree took much much less time to fit (0.2877) compared to SVM fitting time (2.79), also much less time to predict the test data (0.03) compared to the SVM predicting time which is (0.11 s).

For our purpose, which is to predict the survival rate of cancer patients, we select Random Forest as our winning model.