PROJECT PROPOSAL:
BREAST CANCER – MACHINE LEARNING

Source : https://www.kaggle.com/datasets/reihanenamdari/breast-cancer

Goal:

- To achieve the best accuracy in predicting Status

The Dataset:

```python
df = pd.read_csv('Breast_Cancer.csv')
print(df.shape)
df.head()
```

✓ 0.4s Python

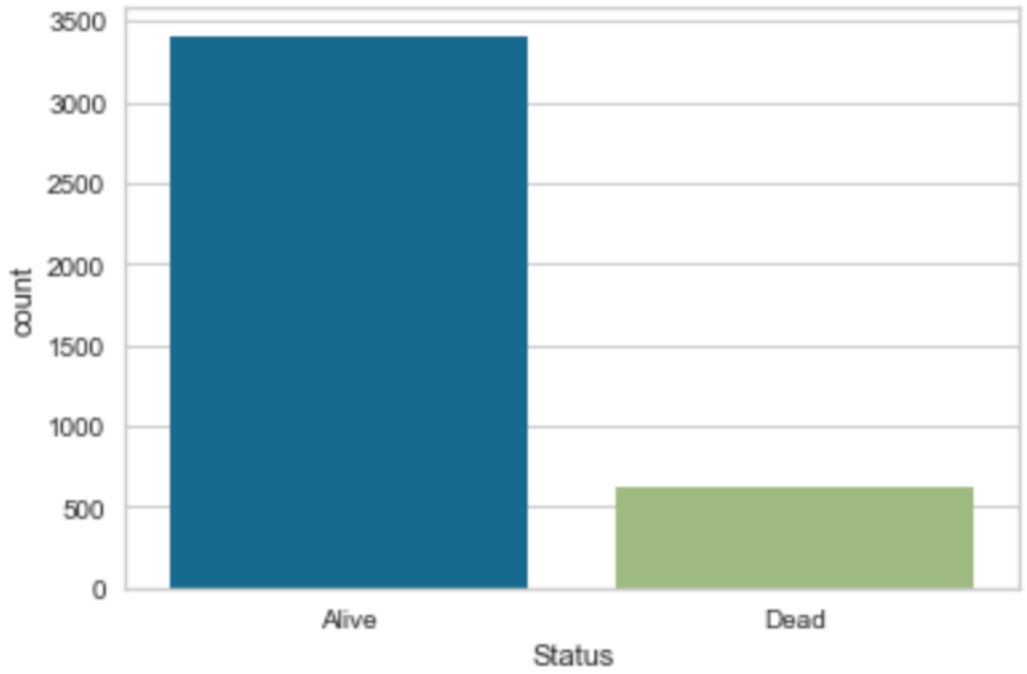
```
(4024, 16)
```

	Age	Race	Marital Status	T Stage	N Stage	6th Stage	differentiate	Grade	A Stage	Tumor Size	Estrogen Status	Progesterone Status	Regional Node Examined	Reginol Node Positive	Survival Months	Status
0	68	White	Married	T1	N1	IIA	Poorly differentiated	3	Regional	4	Positive	Positive	24	1	60	Alive
1	50	White	Married	T2	N2	IIIA	Moderately differentiated	2	Regional	35	Positive	Positive	14	5	62	Alive
2	58	White	Divorced	T3	N3	IIIC	Moderately differentiated	2	Regional	63	Positive	Positive	14	7	75	Alive
3	58	White	Married	T1	N1	IIA	Poorly differentiated	3	Regional	18	Positive	Positive	2	1	84	Alive
4	47	White	Married	T2	N1	IIB	Poorly differentiated	3	Regional	41	Positive	Positive	3	1	50	Alive

- Classification - Binary

Status	
Alive	Dead

```
Alive    3408
Dead      616
Name: Status, dtype: int64
```



T Stage	N Stage
Tumor Size	Lymph Node Status

```
1  df['T Stage '].value_counts()
```

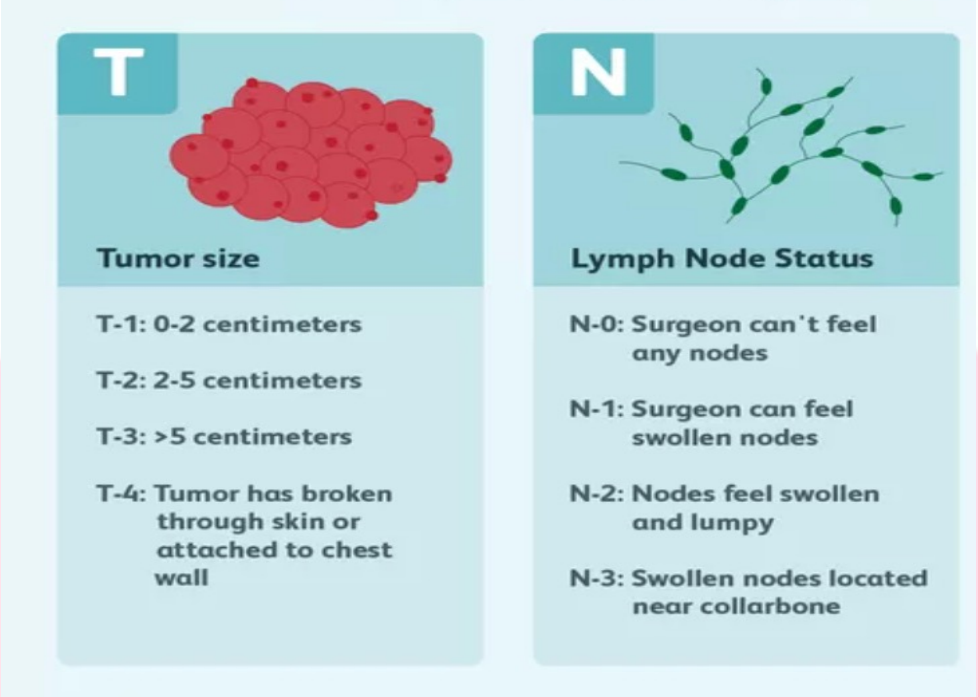
✓ 0.2s

```
T2    1786
T1    1603
T3     533
T4     102
Name: T Stage , dtype: int64
```

```
1  df['N Stage'].value_counts()
```

✓ 0.2s

```
N1    2732
N2     820
N3     472
Name: N Stage, dtype: int64
```

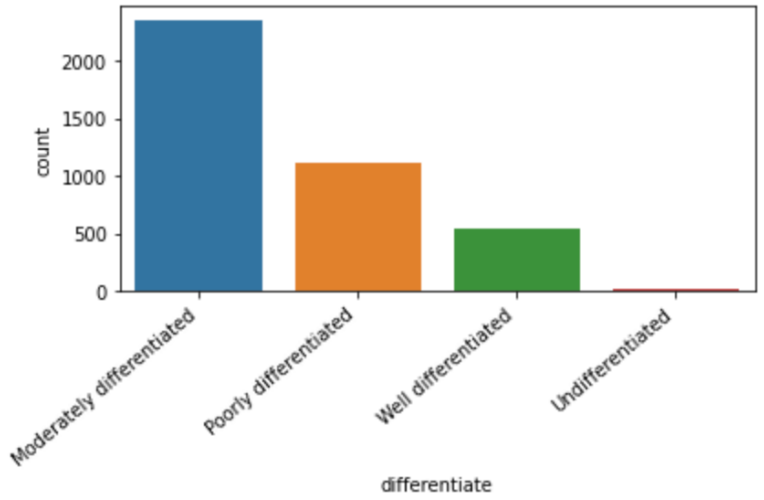


Differentiate

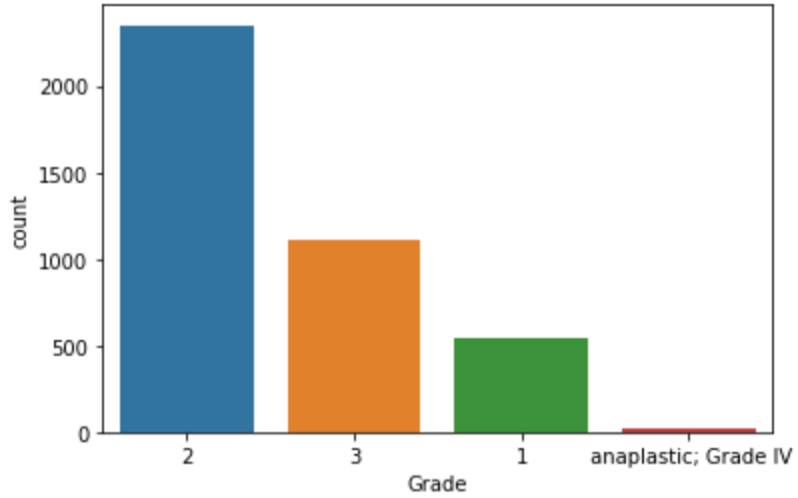
Grade

How Active is the Cancer

```
Moderately differentiated    2351
Poorly differentiated        1111
Well differentiated           543
Undifferentiated               19
Name: differentiate, dtype: int64
```




```
2                          2351
3                          1111
1                           543
 anaplastic; Grade IV       19
Name: Grade, dtype: int64
```

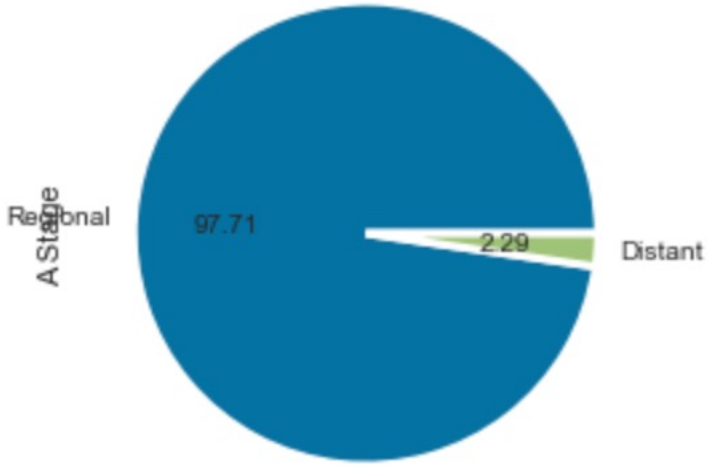


Grading invasive breast cancer cells

Three features of the invasive breast cancer cell are studied and each is given a score. The scores are then added to get a number between 3 and 9 that is used to get a grade of 1, 2, or 3, which is noted on your pathology report. Sometimes the terms *well differentiated*, *moderately differentiated*, and *poorly differentiated* are used to describe the grade instead of numbers:

- **Grade 1 or well differentiated** (score 3, 4, or 5). The cells are slower-growing, and look more like normal breast cells.
- **Grade 2 or moderately differentiated** (score 6, 7). The cells are growing at a speed of and look like cells somewhere between grades 1 and 3.
- **Grade 3 or poorly differentiated** (score 8, 9). The cancer cells look very different from normal cells and will probably grow and spread faster.

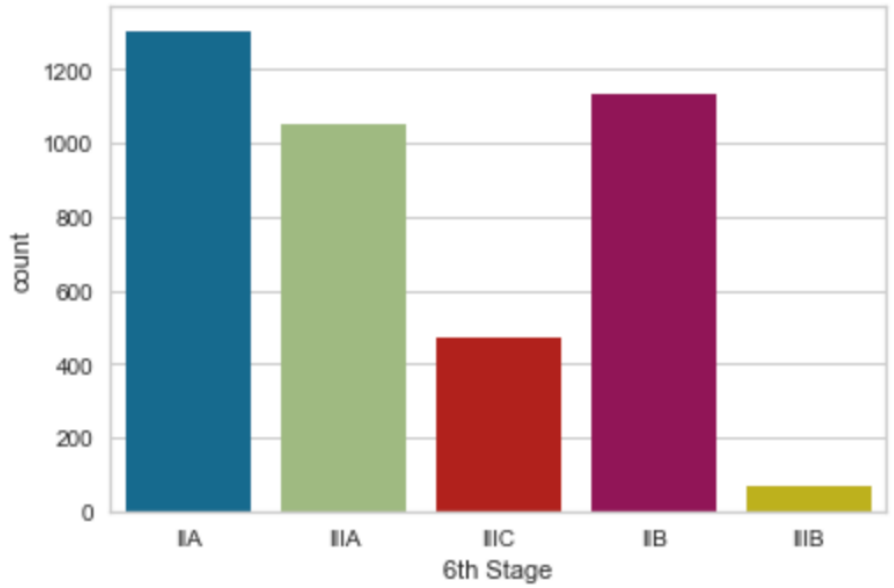
A Stage	6th Stage
State of Cancer [0 = Regional / 1 = Distant] Regional = Cancer has spread to nearby lymph nodes, tissues Distant = Cancer has spread to distant parts of the body such as the lungs, liver or bones	Stage of Cancer https://www.cancer.net/cancer-types/breast-cancer/stages




```
1  sns.countplot(x = '6th Stage', data = df)
```

✓ 0.1s

```
<AxesSubplot:xlabel='6th Stage', ylabel='count'>
```



Estrogen	Progesterone
If both positive , cancer is less aggressive	

```
1  pd.unique(df['Estrogen Status'])
```

✓ 0.4s

```
array(['Positive', 'Negative'], dtype=object)
```

```
1  pd.unique(df['Progesterone Status'])
```

✓ 0.2s

```
array(['Positive', 'Negative'], dtype=object)
```

No empty data cells:

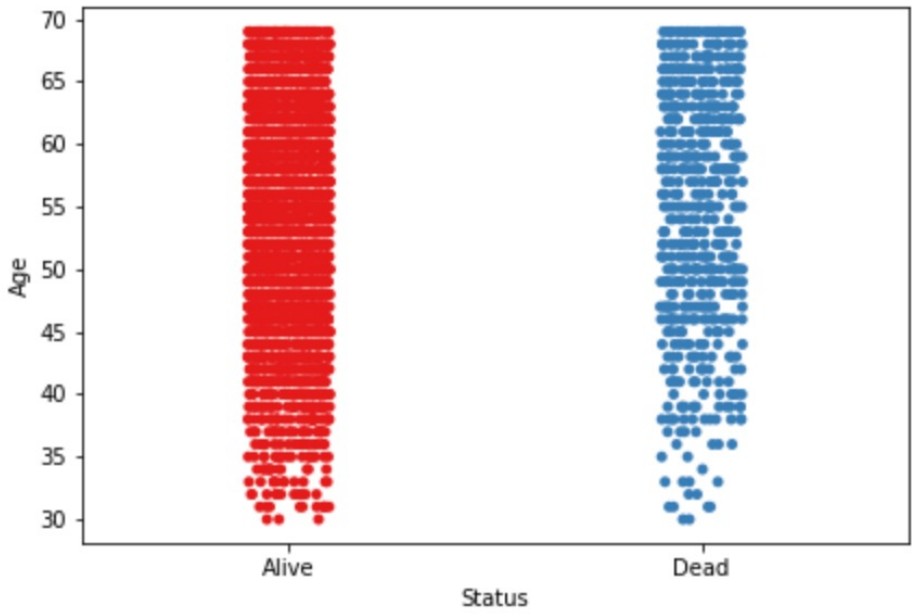
```
1  df.isnull().sum()
```

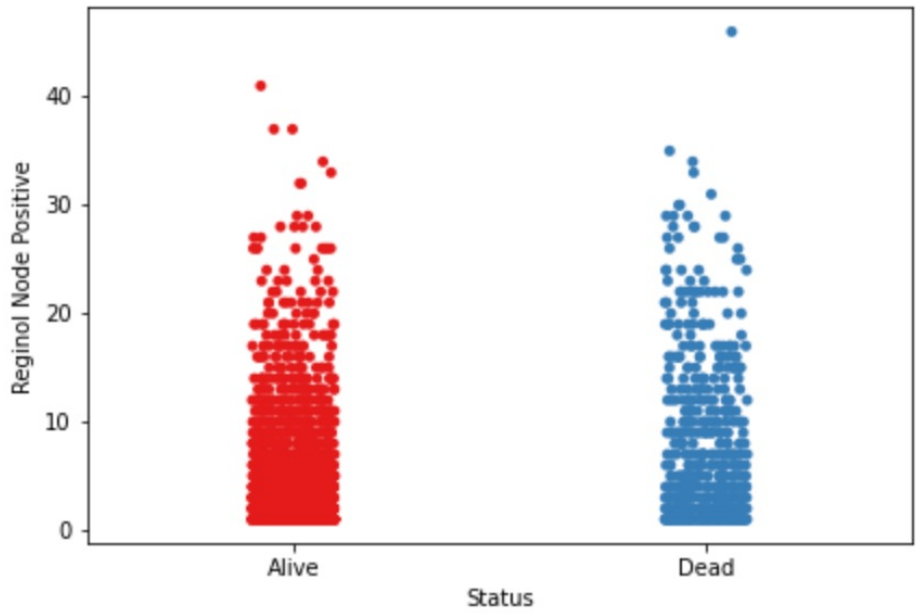
✓ 0.3s

```
Age                          0
Race                         0
Marital Status               0
T Stage                      0
N Stage                      0
6th Stage                    0
differentiate                0
Grade                        0
A Stage                      0
Tumor Size                   0
Estrogen Status              0
Progesterone Status          0
Regional Node Examined       0
Reginol Node Positive        0
Survival Months              0
Status                       0
dtype: int64
```

Correlation at first Glance:

Numerical Variables:

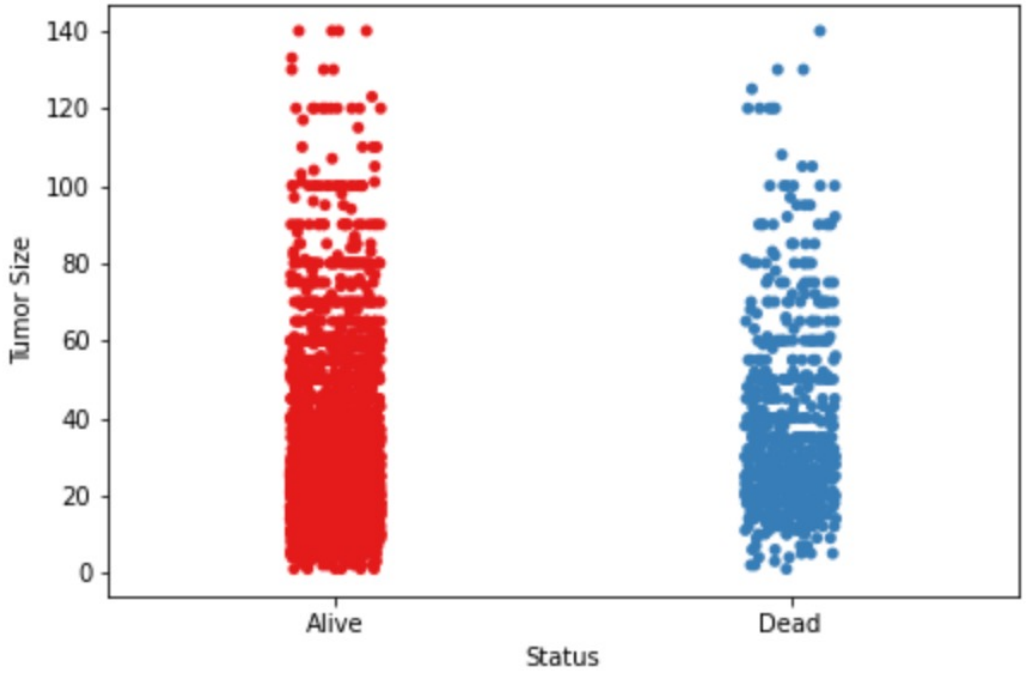


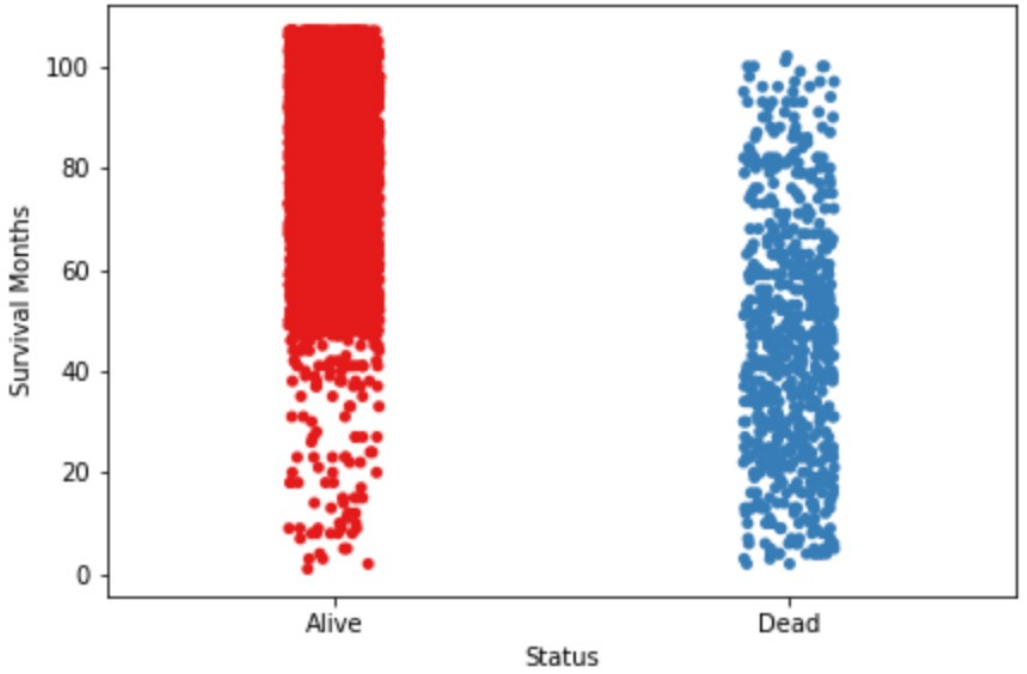


- Here we are checking if Age & Regional Node Positive affects/influence the Status.

<u>Age:</u>
It seems that there is no specific range of Age affecting the Status.

<u>Regional Node Positive:</u>
Higher Node positive tend to die

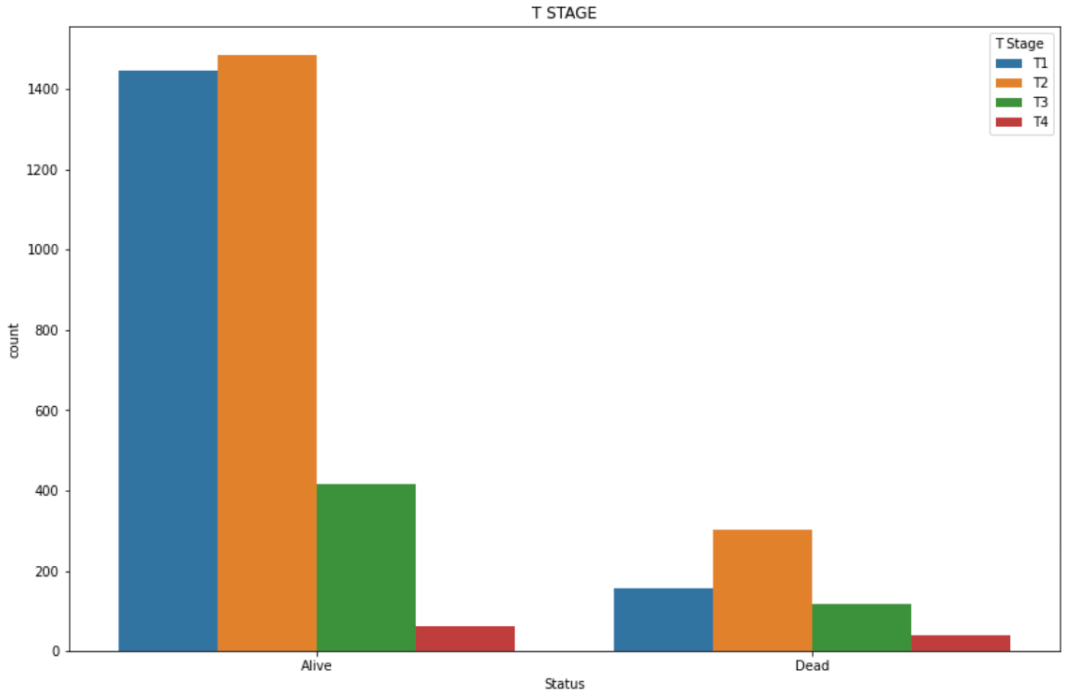


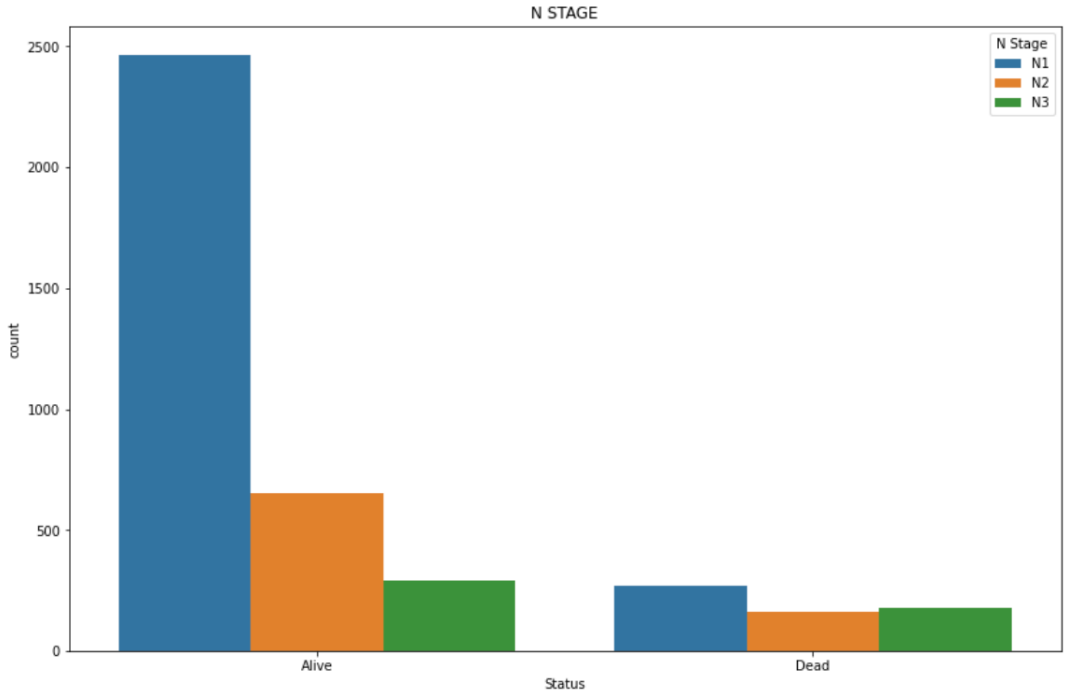


- Here we are checking if Tumor Size & Survival Months affects/influence the Status.

<u>Tumor Size:</u>
Larger tumor Size tend to die , but seems unclear from this plot

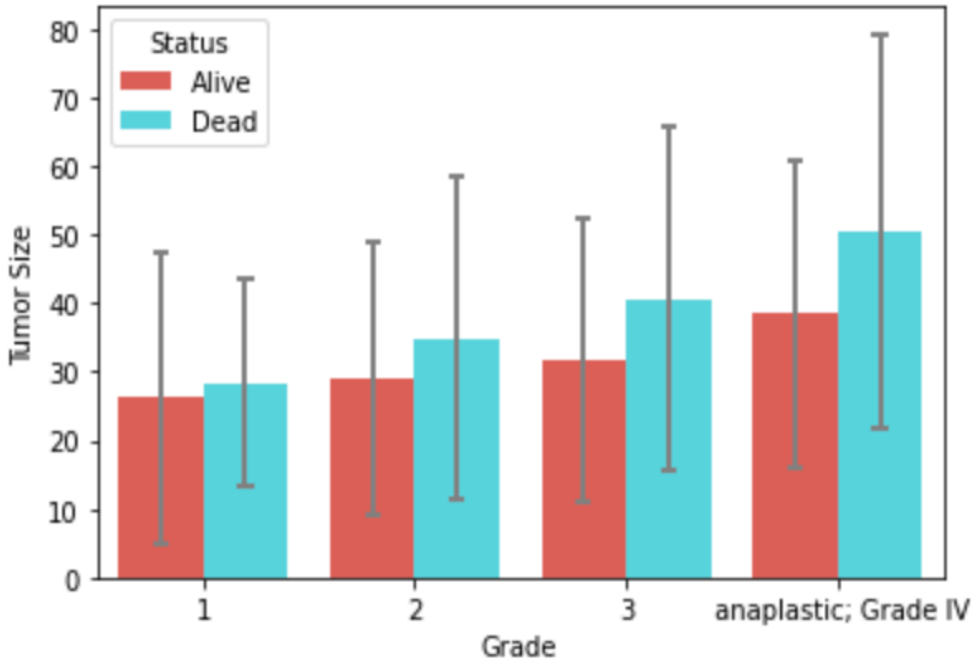
<u>Survival Months:</u>
Higher Survival Months tend to survive

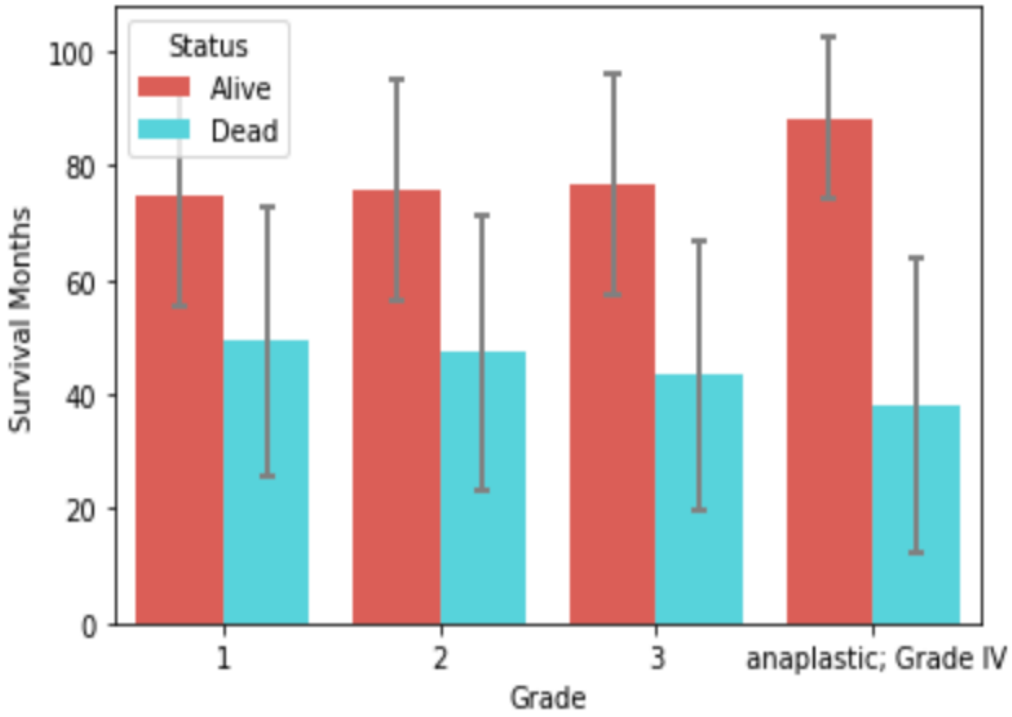




- From The T Stage , we can deduce that early stage has higher chance of Survival

Comparing Against Grade:





- The higher the grade the larger the tumor size
- Survival Months and Grade has little correlation ; Alive goes higher similarly Dead is decreasing

Label Encoding:

	Age	Race	Marital Status	T Stage	N Stage	6th Stage	differentiate	Grade	A Stage	Tumor Size	Estrogen Status	Progesterone Status	Regional Node Examined	Reginol Node Positive	Survival Months	Status
0	38	2	1	0	0	0	1	3	1	3	1	1	23	0	59	0
1	20	2	1	1	1	2	0	2	1	34	1	1	13	4	61	0
2	28	2	0	2	2	4	0	2	1	62	1	1	13	6	74	0
3	28	2	1	0	0	0	1	3	1	17	1	1	1	0	83	0
4	17	2	1	1	0	1	1	3	1	40	1	1	2	0	49	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4019	32	1	1	0	0	0	0	2	1	8	1	1	0	0	48	0
4020	26	2	0	1	1	2	0	2	1	45	1	1	13	7	68	0
4021	38	2	1	1	0	1	0	2	1	21	1	0	10	2	68	0
4022	28	0	0	1	0	1	0	2	1	43	1	1	10	0	71	0
4023	16	2	1	1	0	1	0	2	1	29	1	1	6	1	99	0

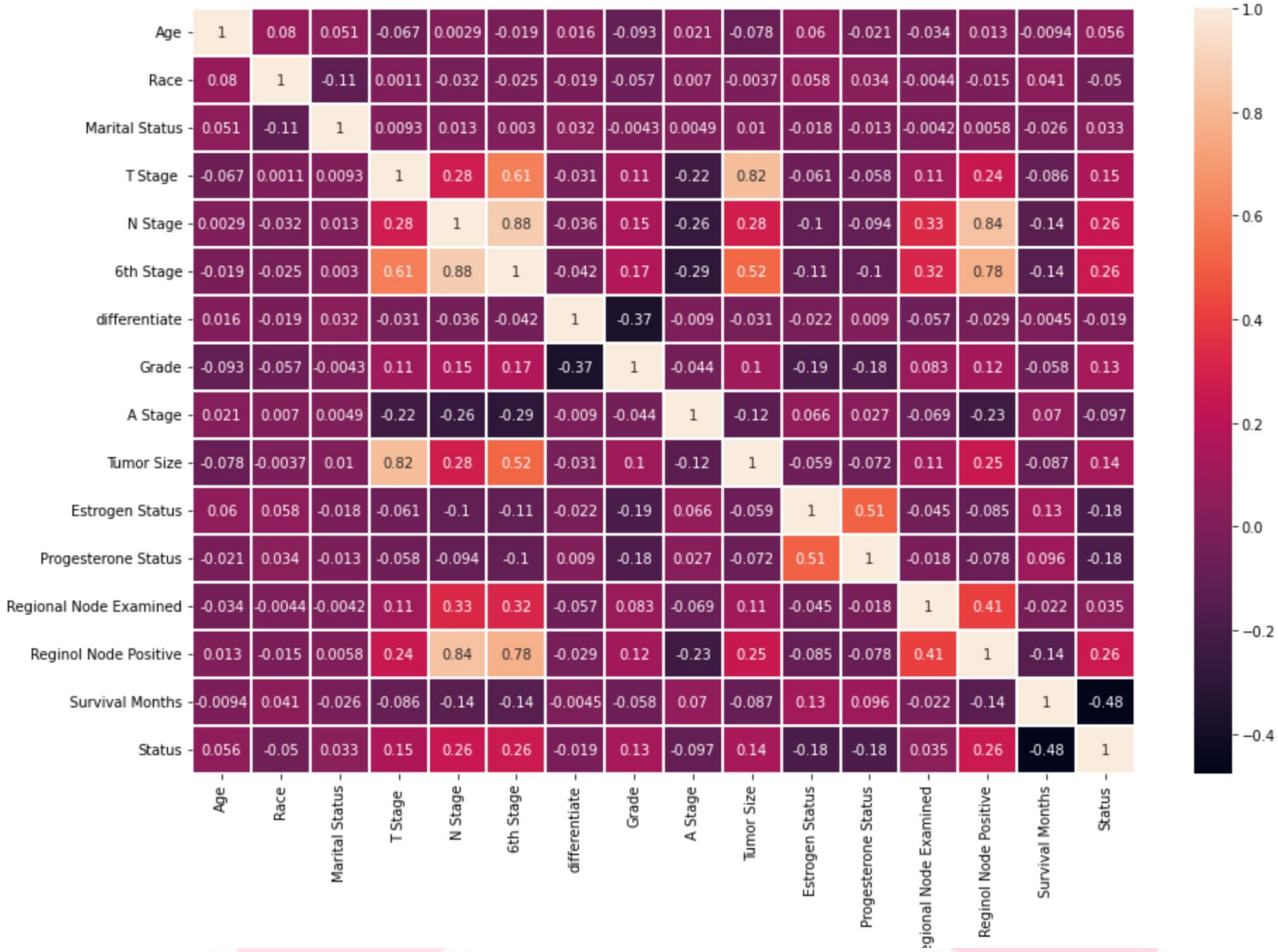
```python
from sklearn.preprocessing import LabelEncoder
```

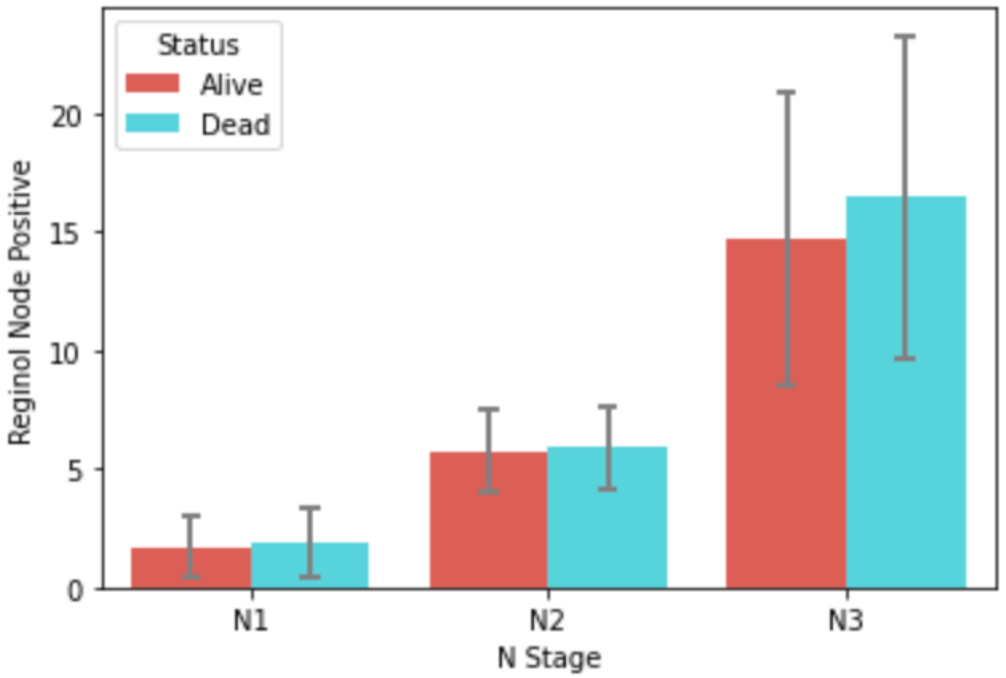
✓ 0.2s Python

```python
for column in df.columns:
    if df[column].dtype == np.number:
        continue
    df[column] = LabelEncoder().fit_transform(df[column]).__array__()
```

✓ 0.3s Python

Correlation:





- Higher N stage corresponds to higher number of regional nodes positive

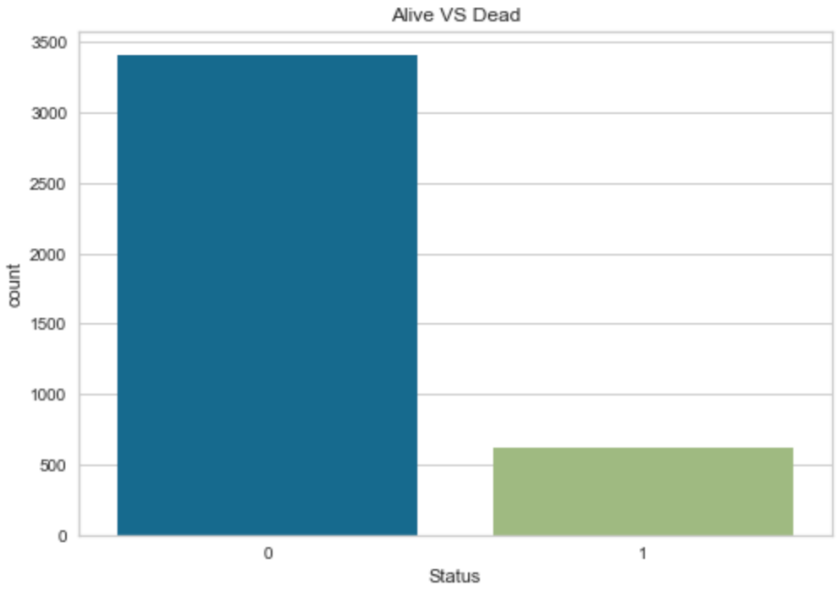
Handling imbalance Dataset: Using SMOTE

```
1  smote = SMOTE(random_state=25)
2  smote_X_train, smote_Y_train = smote.fit_resample(X,y)
```

✓ 0.3s

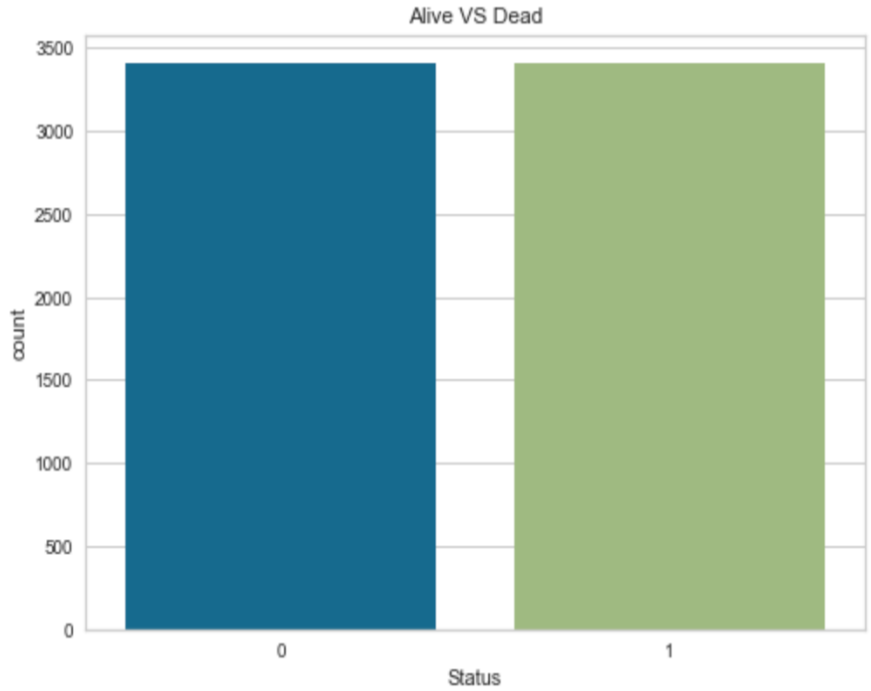
```
1  sns.countplot(x=y, data=df)
2  plt.title('Alive VS Dead')
3  plt.show()
```

✓ 0.1s




```
1  sns.countplot(x=smote_Y_train, data=df)
2  plt.title('Alive VS Dead')
3  plt.show()
```

✓ 0.7s




```
1  print("Before sampling class distribution:", Counter(y))
2  print("After sampling class distribution:", Counter(smote_Y_train))
```

✓ 0.2s

```
Before sampling class distribution: Counter({0: 3408, 1: 616})
After sampling class distribution: Counter({0: 3408, 1: 3408})
```

Train Test Split:

```
1  from sklearn.model_selection import train_test_split
```
✓ 0.1s

```
1  X_train, X_test, y_train, y_test = train_test_split(smote_X_train, smote_Y_train, test_size=0.30, random_state=0)
```
✓ 0.2s

70 - 30

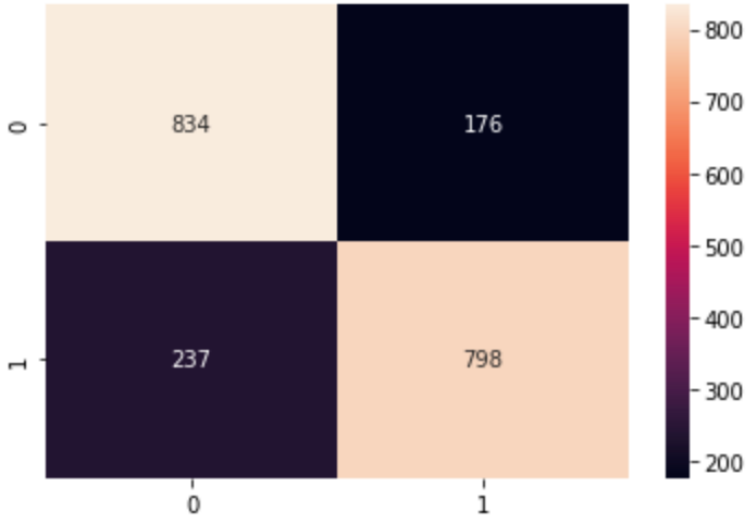
Standard Scaler:

```
1  from sklearn.preprocessing import StandardScaler
2
3  sc = StandardScaler()
4
5  X_train = sc.fit_transform(X_train)
6  X_test = sc.transform(X_test)
```
✓ 0.7s Python

Algorithm used:

- Logistic Regression
- Naïve Bayes
- KNN
- SVC

LOGISTIC REGRESSION:



	precision	recall	f1-score	support
0	0.78	0.83	0.80	1010
1	0.82	0.77	0.79	1035
accuracy			0.80	2045
macro avg	0.80	0.80	0.80	2045
weighted avg	0.80	0.80	0.80	2045

Accuracy is: 79.804%

```
parameters = {
    'penalty' : ['l1','l2'],
    'C'       : np.logspace(-3,3,7),
    'solver'  : ['newton-cg', 'lbfgs', 'liblinear'],
}
```

✓ 0.2s

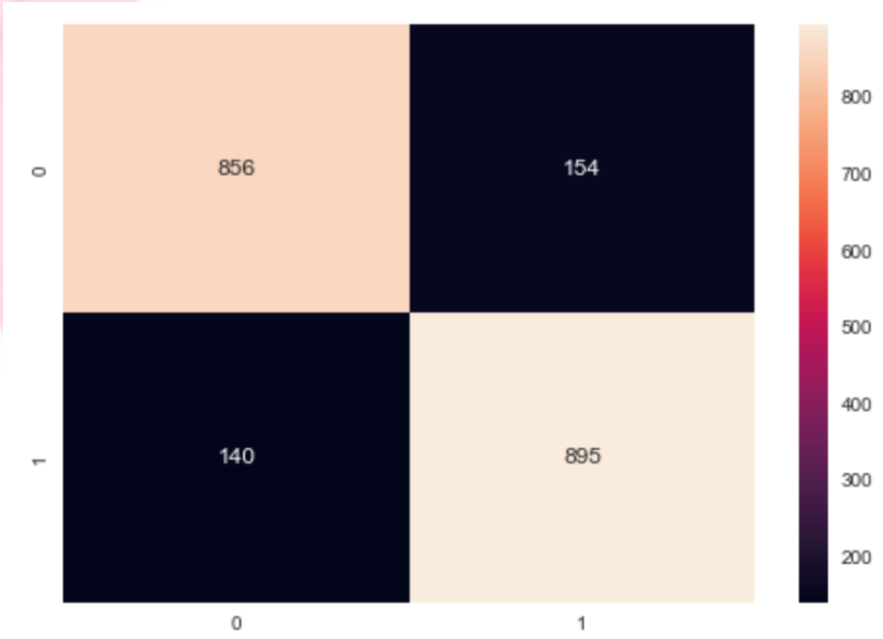
```
logistic_regression = LogisticRegression()
clf = GridSearchCV(logistic_regression,                    # model
                   param_grid = parameters,   # hyperparameters
                   scoring='accuracy',        # metric for scoring
                   cv=10)                     # number of folds
```

✓ 0.2s

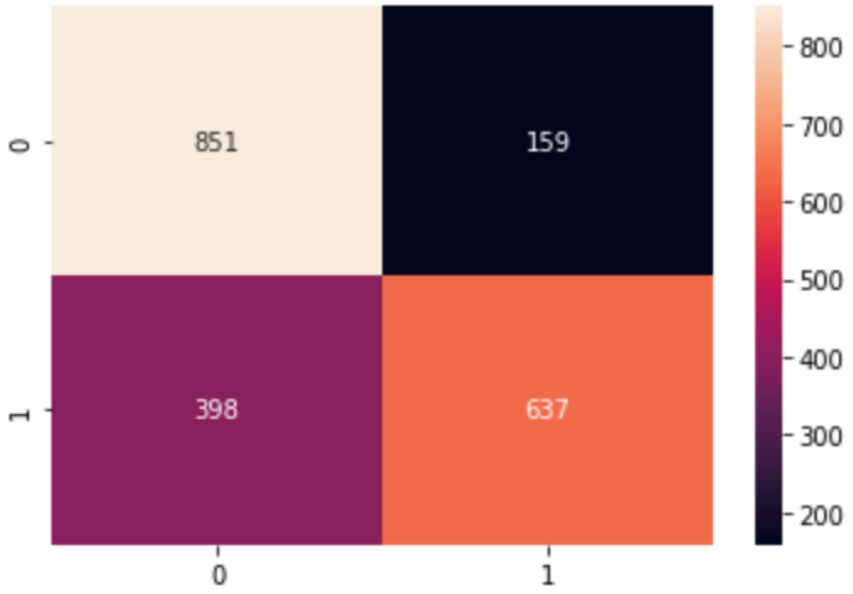
```
clf.fit(X_train,y_train)
print("Tuned Hyperparameters :", clf.best_params_)
print('Accuracy is: {:.3f}%'.format(clf.best_score_*100))
```

✓ 7.1s

```
Tuned Hyperparameters : {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
Accuracy is: 80.507%
```

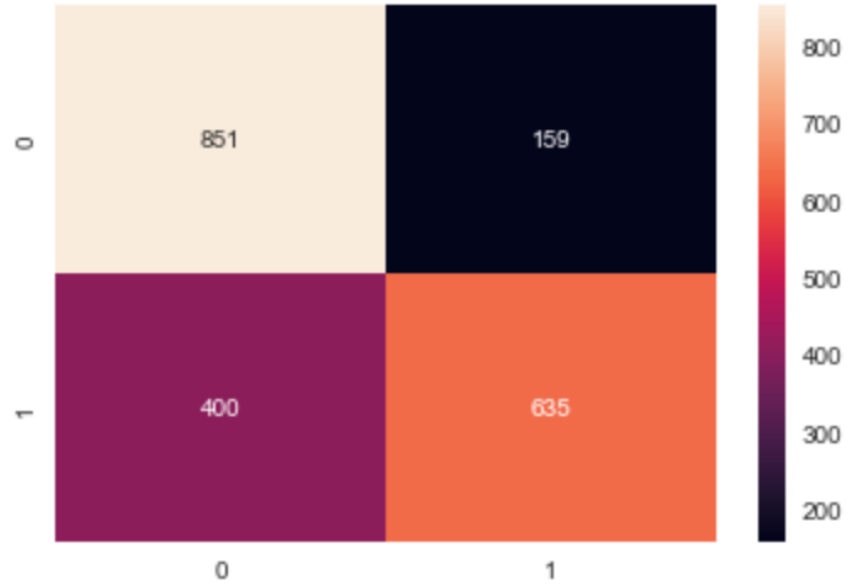


NAIVE BAYES:



	precision	recall	f1-score	support
0	0.68	0.84	0.75	1010
1	0.80	0.62	0.70	1035
accuracy			0.73	2045
macro avg	0.74	0.73	0.72	2045
weighted avg	0.74	0.73	0.72	2045

Accuracy is: 72.763%




```
param_grid_nb = {
    'var_smoothing': np.logspace(0,-9, num=100)
}
```

✓ 0.1s

```
nbModel_grid = GridSearchCV(estimator=GaussianNB(), param_grid=param_grid_nb, verbose=1, cv=10, n_jobs=-1, scoring='accuracy')
nbModel_grid.fit(X_train, y_train)
print("Tuned Hyperparameters :", nbModel_grid.best_params_)
print('Accuracy is: {:.3f}%'.format(nbModel_grid.best_score_*100))
```

✓ 0.5s

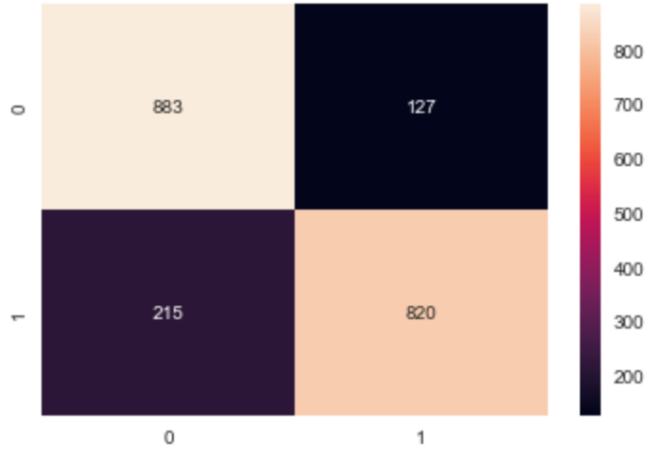
```
Fitting 10 folds for each of 100 candidates, totalling 1000 fits
Tuned Hyperparameters : {'var_smoothing': 0.006579332246575682}
Accuracy is: 73.821%
```

	precision	recall	f1-score	support
0	0.68	0.84	0.75	1010
1	0.80	0.61	0.69	1035
accuracy			0.73	2045
macro avg	0.74	0.73	0.72	2045
weighted avg	0.74	0.73	0.72	2045

KNN:

```
Accuracy is: 83.28%
```

```
<AxesSubplot:>
```

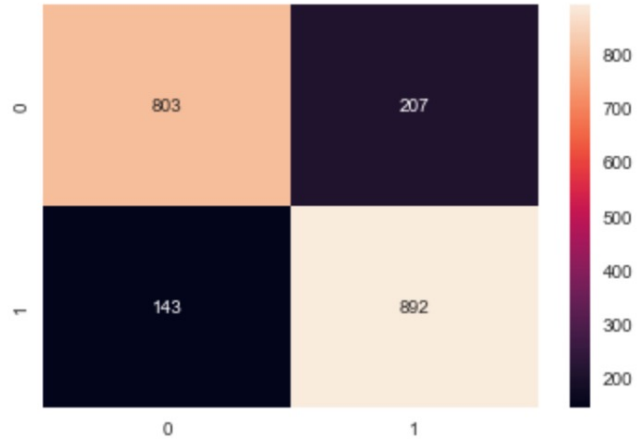



```
1  print(classification_report(y_test, y_pred))
```
✓ 0.2s

	precision	recall	f1-score	support
0	0.85	0.80	0.82	1010
1	0.81	0.86	0.84	1035
accuracy			0.83	2045
macro avg	0.83	0.83	0.83	2045
weighted avg	0.83	0.83	0.83	2045

```
2  sns.heatmap(cm, annot=True, fmt="g")
```
✓ 0.1s

```
<AxesSubplot:>
```

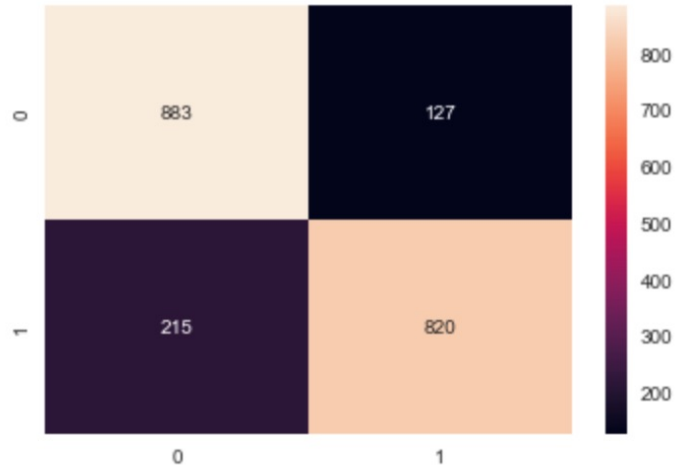



```
1  print(classification_report(y_test, grid_pre))
```
✓ 0.6s

	precision	recall	f1-score	support
0	0.85	0.80	0.82	1010
1	0.81	0.86	0.84	1035
accuracy			0.83	2045
macro avg	0.83	0.83	0.83	2045
weighted avg	0.83	0.83	0.83	2045

Accuracy for our training dataset with tuning is : 86.59%

SVC:




```
1  print(classification_report(y_test, y_pred))
```
✓ 0.2s

	precision	recall	f1-score	support
0	0.80	0.87	0.84	1010
1	0.87	0.79	0.83	1035
accuracy			0.83	2045
macro avg	0.84	0.83	0.83	2045
weighted avg	0.84	0.83	0.83	2045

Accuracy is: 83.276%

Tuned Hyperparameters : {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}

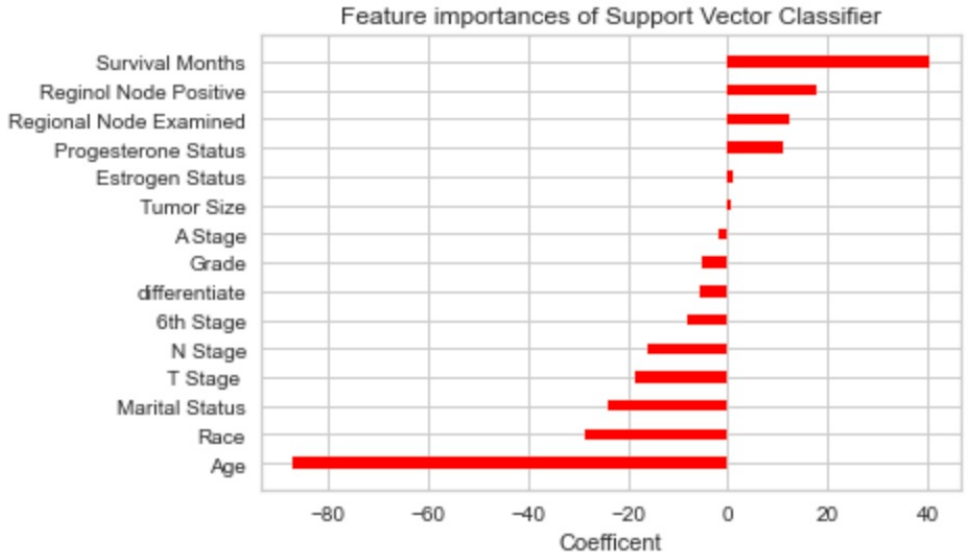
SVC(C=100, gamma=0.1)

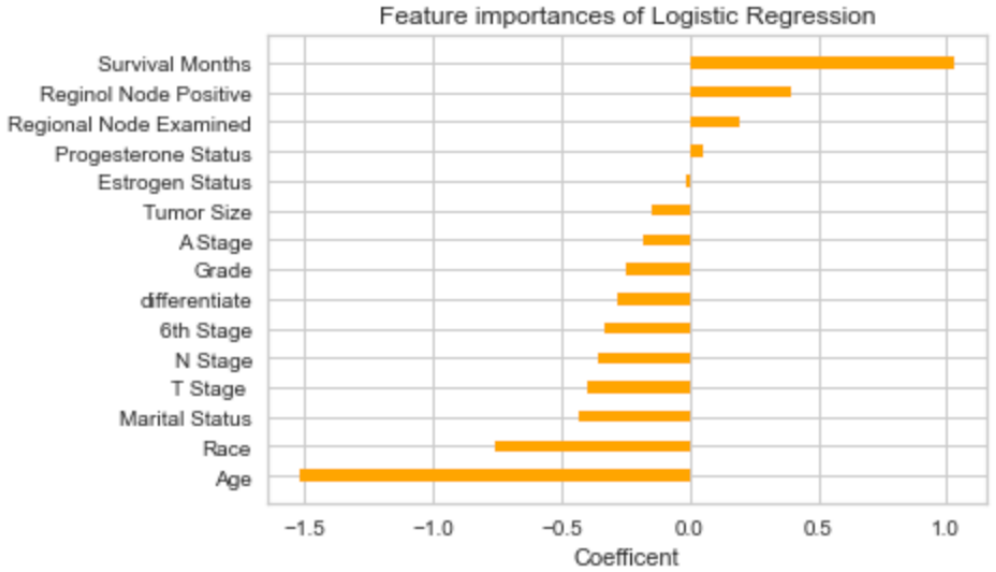
Accuracy is: 85.914%

```
1  grid_predictions = svc_grid.predict(X_test)
2
3  # print classification report
4  print(classification_report(y_test, grid_predictions))
```
36] ✓ 0.2s

	precision	recall	f1-score	support
0	0.86	0.85	0.85	1010
1	0.85	0.86	0.86	1035
accuracy			0.86	2045
macro avg	0.86	0.86	0.86	2045
weighted avg	0.86	0.86	0.86	2045

Feature Importance:





Conclusion:

- Logistic Regression , KNN & SVC are the algorithms to consider

- No pattern detected during EDA

- Our Data might not be a high Quality
    - Need a larger Data