Lab Assignment One: Exploring Table Data

Ahmet Ata Ersoy

Yasin Cagatay Duygu

1. Business Understanding

This dataset of breast cancer patients was obtained from the SEER Program of the NCI, shows the 2017 November update. It provides information on population-based cancer statistics. The dataset involves female patients with infiltrating duct and lobular carcinoma breast cancer. Patients were diagnosed between 2006-2010. 4024 patients were ultimately included. This dataset is found from Google Dataset Search. We wanted to work on health care data. This data set is open-source and can be found in the following links:

https://ieee-dataport.org/open-access/seer-breast-cancer-data

https://zenodo.org/record/5120960#.YxVhSnbMJPZ

This data set has 15 features including age, race, marital status, N-stage, 6th Stage, Grade, A Stage, Tumor Size, Estrogen Status, Progesterone Status, regional node examined, regional node positive, survival months and survival status. We did an analysis to show the relationship between survival status and other features.

This analysis can be used by the administration of the hospitals. We experienced a pandemic in which hospitals reach their capacity. Thus, this analysis is important for hospital's source utilization. This analysis is also critical to understand which patient should be screened aggressively. It is crucial for hospitals and doctors to predict which patients will receive what type of treatment (neo-adjuvant, adjuvant chemotherapy and/or radiation). Besides, hospitals can plan increasing screening outreach in certain population groups that have higher mortality rate.

Unfortunately, we could not found a cut-off value for how well our prediction algorithm should perform to be useful in medicine.

These following questions will be investigated

1. Can we predict survival status of patients?
2. Which attributes correlate with each other?
3. How does breast cancer affect different races?

2. Data Understanding

2.1 Data Description

In [ ]:
```
# Load the dataset
import pandas as pd
import numpy as np
from statistics import stdev
import os
import warnings
warnings.filterwarnings("ignore")

print('Pandas:', pd.__version__)
print('Numpy:',np.__version__)

import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('SEER Breast Cancer Dataset_2.csv') # read in the csv file

#Making sure that the variable are callable as objects using their name. Therefore we had to replace category names.
df = df.rename(columns={'T Stage ': 'T_stage', 'N Stage':'N_Stage', '6th Stage': 'SixStage', 'Race ': 'Race', 'A Stage': 'A_Stage', 'Tumor Size': 'Tumor_Size', 'Estrogen Status' : 'Estrogen

df.head()
```

```
Pandas: 1.3.5
Numpy: 1.21.6
```

Out[ ]:

	Age	Race	Marital Status	T_stage	N_Stage	SixStage	Grade	A_Stage	Tumor_Size	Estrogen_Status	Progesterone_Status	Regional_Node_Examined	Reginol_Node_Positive	Survival_Months	Status
0	43	Other (American Indian/AK Native, Asian/Pacifi...	Married (including common law)	T2	N3	IIIC	Moderately differentiated; Grade II	Regional	40	Positive	Positive	19	11	1	Alive
1	47	Other (American Indian/AK Native, Asian/Pacifi...	Married (including common law)	T2	N2	IIIA	Moderately differentiated; Grade II	Regional	45	Positive	Positive	25	9	2	Alive
2	67	White	Married (including common law)	T2	N1	IIB	Poorly differentiated; Grade III	Regional	25	Positive	Positive	4	1	2	Dead
3	46	White	Divorced	T1	N1	IIA	Moderately differentiated; Grade II	Regional	19	Positive	Positive	26	1	2	Dead
4	63	White	Married (including common law)	T2	N2	IIIA	Moderately differentiated; Grade II	Regional	35	Positive	Positive	21	5	3	Dead

In [ ]:
```
# to convert ordinal data to numerical form
df_2=df.replace(to_replace = 'T1', value = 1)
df_2=df_2.replace(to_replace = 'T2', value = 2)
df_2=df_2.replace(to_replace = 'T3', value = 3)
df_2=df_2.replace(to_replace = 'T4', value = 4)
df_2=df_2.replace(to_replace = 'N2', value = 2)
df_2=df_2.replace(to_replace = 'N1', value = 1)
df_2=df_2.replace(to_replace = 'N3', value = 3)
df_2=df_2.replace(to_replace = 'Regional', value = 0)
df_2=df_2.replace(to_replace = 'Distant', value = 1)
df_2=df_2.replace(to_replace = 'Positive', value = 1)
df_2=df_2.replace(to_replace = 'Negative', value = 0)
df_2=df_2.replace(to_replace = 'Alive', value = 1)
df_2=df_2.replace(to_replace = 'Dead', value = 0)
df_2=df_2.replace(to_replace = 'IIA', value = 1)
df_2=df_2.replace(to_replace = 'IIB', value = 2)
df_2=df_2.replace(to_replace = 'IIIA', value = 3)
df_2=df_2.replace(to_replace = 'IIIB', value = 4)
df_2=df_2.replace(to_replace = 'IIIC', value = 5)
df_2=df_2.replace(to_replace = 'Well differentiated; Grade I', value = 1)
df_2=df_2.replace(to_replace = 'Moderately differentiated; Grade II', value = 2)
df_2=df_2.replace(to_replace = 'Poorly differentiated; Grade III', value = 3)
df_2=df_2.replace(to_replace = 'Undifferentiated; anaplastic; Grade IV', value = 4)
```

In [ ]:
```
# Check data types
df_2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4024 entries, 0 to 4023
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype 
---  ------                  --------------  ----- 
 0   Age                     4024 non-null   int64 
 1   Race                    4024 non-null   object
 2   Marital Status          4024 non-null   object
 3   T_stage                 4024 non-null   int64 
 4   N_Stage                 4024 non-null   int64 
 5   SixStage                4024 non-null   int64 
 6   Grade                   4024 non-null   int64 
 7   A_Stage                 4024 non-null   int64 
 8   Tumor_Size              4024 non-null   int64 
 9   Estrogen_Status         4024 non-null   int64 
 10  Progesterone_Status     4024 non-null   int64 
 11  Regional_Node_Examined  4024 non-null   int64 
 12  Reginol_Node_Positive   4024 non-null   int64 
 13  Survival_Months         4024 non-null   int64 
 14  Status                  4024 non-null   int64 
dtypes: int64(13), object(2)
memory usage: 471.7+ KB
```

Based on the dataframe information, there are no missing values in the dataset as there are 4024 entries in each feature with all non-null elements (There are 4024 rows in the data).

If there were missing values in the dataset, we could try split-impute-combine or K-Nearest Neighbors Imputation methods.

In [ ]:
```
# describe function was used to generate descriptive statistical summary of the dataset.
df_2.describe()
```

Out[ ]:

	Age	T_stage	N_Stage	SixStage	Grade	A_Stage	Tumor_Size	Estrogen_Status	Progesterone_Status	Regional_Node_Examined	Reginol_Node_Positive	Survival_Months	Status
count	4024.000000	4024.000000	4024.000000	4024.000000	4024.000000	4024.000000	4024.000000	4024.000000	4024.000000	4024.000000	4024.000000	4024.000000	4024.000000
mean	53.972167	1.784791	1.438370	2.321819	2.150596	0.022863	30.473658	0.933151	0.826541	14.357107	4.158052	71.297962	0.846918
std	8.963134	0.765531	0.693479	1.266624	0.638234	0.149485	21.119696	0.249791	0.378691	8.099675	5.109331	22.921430	0.360111
min	30.000000	1.000000	1.000000	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	1.000000	1.000000	1.000000	0.000000
25%	47.000000	1.000000	1.000000	1.000000	2.000000	0.000000	16.000000	1.000000	1.000000	9.000000	1.000000	56.000000	1.000000
50%	54.000000	2.000000	1.000000	2.000000	2.000000	0.000000	25.000000	1.000000	1.000000	14.000000	2.000000	73.000000	1.000000
75%	61.000000	2.000000	2.000000	3.000000	3.000000	0.000000	38.000000	1.000000	1.000000	19.000000	5.000000	90.000000	1.000000
max	69.000000	4.000000	3.000000	5.000000	4.000000	1.000000	140.000000	1.000000	1.000000	61.000000	46.000000	107.000000	1.000000

In [ ]:
```
# create a data description table
data_des = pd.DataFrame()

data_des['Features'] = df.columns
data_des['Description'] = ['Age of patient',
                           'Race of patient',
                          'Marital Status',
                          'T followed by a number from 0 to 4 describes the main (primary) tumor`s size and if it has spread to the skin or to the chest wall under the breast.',
                          'N followed by a number from 0 to 3 indicates whether the cancer has spread to lymph nodes near the breast and, if so, how many lymph nodes are involved.',
                          'Cancer stage defined based on 6th Edition of the American Communite of Cancer staging manual',
                          'Grade',
                          'Indicates whether the cancer is locally advanced or metastasised',
                           'Tumor Size',
                           'Estrogen Status',
                           'Progesterone_Status',
                           'Records the total number of regional lymph nodes that were removed and examined by the pathologist.',
                           'Records the exact number of regional lymph nodes examined by the pathologist that were found to contain metastases.',
                           'How many months the patient survived after diagnosis',
                           'Any patient that dies after the follow-up cut-off date is recoded to alive as of the cut-off date.',
                           ]

data_des['Scales'] = ['ratio', 'nominal', 'nominal', 'ordinal','ordinal','ordinal','ordinal', 'ordinal','ratio', 'ratio', 'nominal', 'ratio','ratio','ratio', 'nominal']
data_des['Discrete\Continuous'] = ['continuous', 'discrete', 'discrete', 'discrete', 'discrete', 'discrete', 'discrete', 'discrete','continuous','discrete','discrete','continuous','continuo
data_des['Range'] = ['30 - 69',
                     '{Black, Other (American Indian/AK Native, Asian/Pacific Islander), White}',
                     'Married (including common law), Divorced, Single, Widowed',
                     '1 : T1, 2 : T2, 3 : T3, 4 : T4',
                     '1 : N1, 2 : N2, 3 : N3',
                     '1 : IIA, 2 : IIB, 3 : IIIA, 4 : IIIB, 5 : IIIC',
                     '1 : Well differentiated; Grade I, 2 : Moderately differentiated; Grade II, 3 : Poorly differentiated; Grade III, 4 : Undifferentiated; anaplastic; Grade IV',
                     '0 : Distance, 1 : Regional',
                     '1 - 140',
                     '0 : Negative, 1 : Positive',
                     '0 : Negative, 1 : Positive',
                     '1 - 61',
                     '1 - 46',
                     '1 - 107',
                     '0 : Alive, 1 : Dead']

data_des
```

Out[ ]:

	Features	Description	Scales	Discrete\Continuous	Range
0	Age	Age of patient	ratio	continuous	30 - 69
1	Race	Race of patient	nominal	discrete	{Black, Other (American Indian/AK Native, Asia...
2	Marital Status	Marital Status	nominal	discrete	Married (including common law), Divorced, Sing...
3	T_stage	T followed by a number from 0 to 4 describes t...	ordinal	discrete	1 : T1, 2 : T2, 3 : T3, 4 : T4
4	N_Stage	N followed by a number from 0 to 3 indicates w...	ordinal	discrete	1 : N1, 2 : N2, 3 : N3
5	SixStage	Cancer stage defined based on 6th Edition of t...	ordinal	discrete	1 : IIA, 2 : IIB, 3 : IIIA, 4 : IIIB, 5 : IIIC
6	Grade	Grade	ordinal	discrete	1 : Well differentiated; Grade I, 2 : Moderate...
7	A_Stage	Indicates whether the cancer is locally advanc...	ordinal	discrete	0 : Distance, 1 : Regional
8	Tumor_Size	Tumor Size	ratio	continuous	1 - 140
9	Estrogen_Status	Estrogen Status	ratio	discrete	0 : Negative, 1 : Positive
10	Progesterone_Status	Progesterone_Status	nominal	discrete	0 : Negative, 1 : Positive
11	Regional_Node_Examined	Records the total number of regional lymph nod...	ratio	continuous	1 - 61
12	Reginol_Node_Positive	Records the exact number of regional lymph nod...	ratio	continuous	1 - 46
13	Survival_Months	How many months the patient survived after dia...	ratio	continuous	1 - 107
14	Status	Any patient that dies after the follow-up cut-...	nominal	discrete	0 : Alive, 1 : Dead

2.2 Data Quality

Let's clean the dataset a little before moving on.

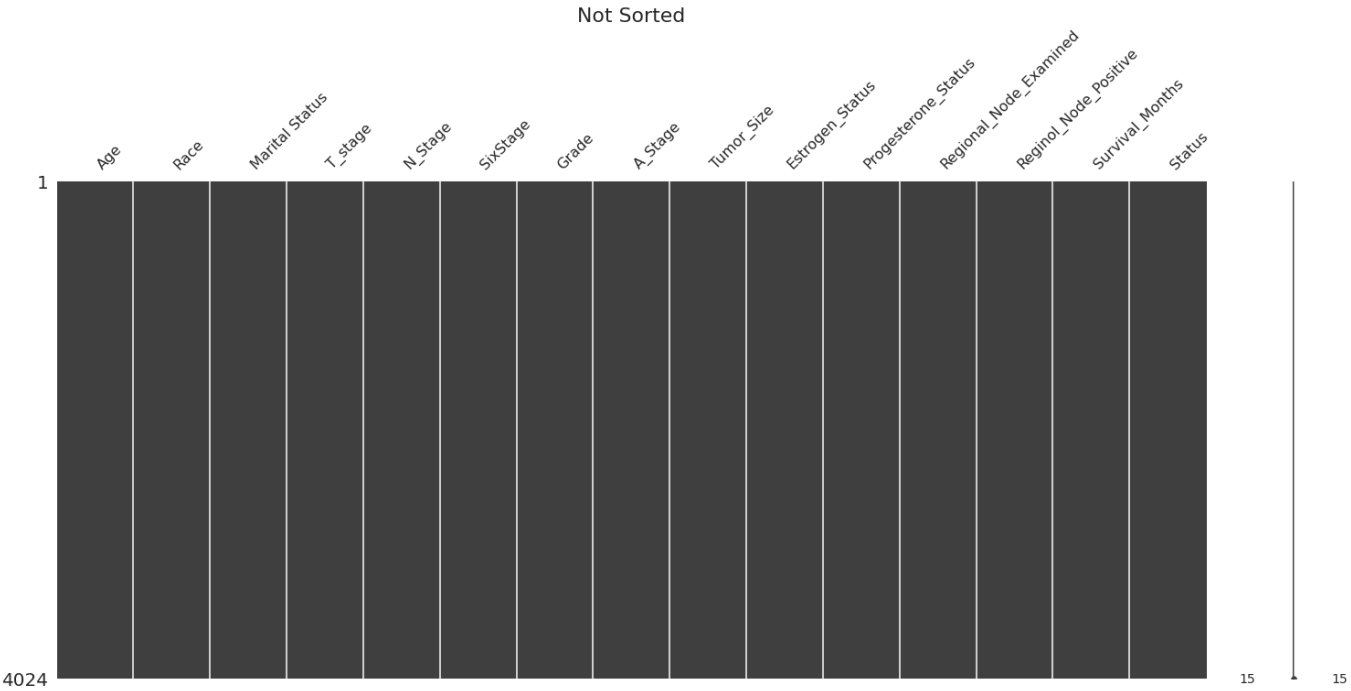
All attributes are useful for the analysi and as stated before, there is no missing values in our chosen dataset.

First, we start by checking if we have any duplicates in our data. Our data doesn't have any duplicates as can be seen below.

In [ ]:
```
# checking for duplicate instances
idx = df_2.duplicated()

# find the number of duplicate (not first show)
len(df_2[idx])
```

Out[ ]: 1

In [ ]:
```
# this python magics will allow plot to be embedded into the notebook
import matplotlib
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter('ignore', DeprecationWarning)
%matplotlib inline

# External package: conda install missingno
import missingno as mn

mn.matrix(df)
plt.title("Not Sorted",fontsize=22)

plt.figure()

plt.show()
```



<Figure size 432x288 with 0 Axes>

3. Data Visualization

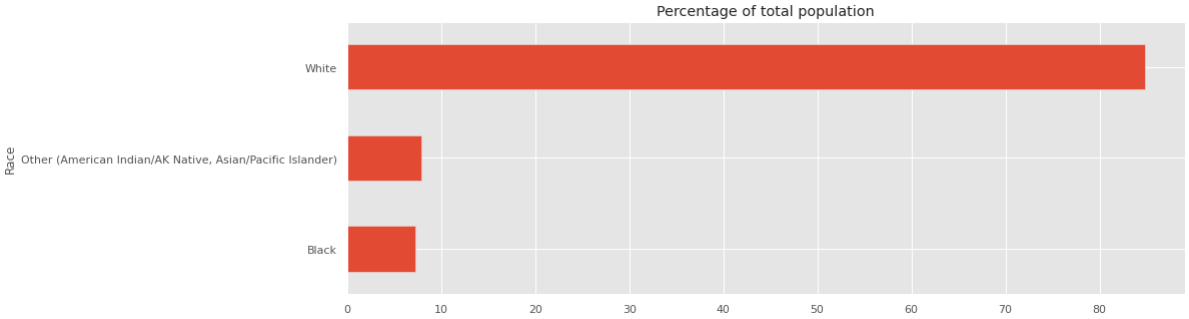
3.1 Data Exploration

In [ ]:
```
# Lets aggregate by department and count the resignation rate
df_grouped_race = df_2.groupby(by='Race')
```

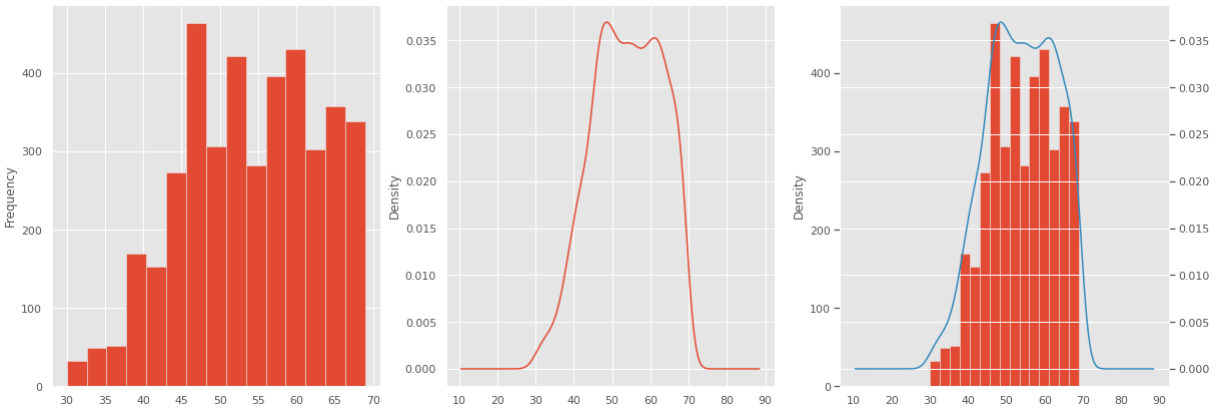
```
for val,grp in df_grouped_race:
    print('The race of',len(grp),'people is',val +'.')

print('----------------------------------------')
print('Representation of each population (%):')
print(df_grouped_race.Age.count()/df_2.Age.count()*100,)

print('========================================')

# Start by just plotting what we previsously grouped!
plt.style.use('ggplot')

fig = plt.figure(figsize=(15,5))

repr_rate = df_grouped_race.Age.count()/df_2.Age.count()*100
ax = repr_rate.plot(kind='barh')
plt.title('Percentage of total population')
```

```
The race of 291 people is Black.
The race of 320 people is Other (American Indian/AK Native, Asian/Pacific Islander).
The race of 3413 people is White.
----------------------------------------
Representation of each population (%):
Race
Black                                                 7.231610
Other (American Indian/AK Native, Asian/Pacific Islander)     7.952286
White                                                84.816103
Name: Age, dtype: float64
========================================
```

Out[ ]: Text(0.5, 1.0, 'Percentage of total population')



In [ ]:
```
# Start by just plotting what we previously grouped!
plt.style.use('ggplot')

fig = plt.figure(figsize=(20,7))

plt.subplot(1,3,1)
df_2.Age.plot.hist(bins=15)

plt.subplot(1,3,2)
df_2.Age.plot.kde(bw_method=0.2)

plt.subplot(1,3,3)
df_2.Age.plot.hist(bins=15)
df_2.Age.plot.kde(bw_method=0.2, secondary_y=True)

# remember that visualization is interpretted, it supports evidence.
# plt.ylim([0, 0.06])

plt.show()
```



5 year is an important milestone for cancer patients. In literature, 5 year survival rate is commonly investigated. Therefore, we filtered our data based on the patients that survived the 5 year threshold or died within the first 5 years.

In [ ]:
```
# To study the 5-year survival rates, data was reduced into two groups
df_5y_Survived = df_2[(df_2.Survival_Months > 59) & (df_2.Status == 1)]
df_5y_NotSurvived = df_2[(df_2.Survival_Months < 59) & (df_2.Status == 0)]
frames = [df_5y_Survived, df_5y_NotSurvived]
df_5y= pd.concat(frames)
#df_5y = df_5y_Survived + df_5y_NotSurvived
df_5y.info()
```

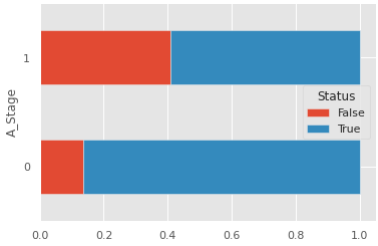
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3089 entries, 1203 to 1124
Data columns (total 15 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     3089 non-null   int64
 1   Race                    3089 non-null   object
 2   Marital Status          3089 non-null   object
 3   T_stage                 3089 non-null   int64
 4   N_Stage                 3089 non-null   int64
 5   SixStage                3089 non-null   int64
 6   Grade                   3089 non-null   int64
 7   A_Stage                 3089 non-null   int64
 8   Tumor_Size              3089 non-null   int64
 9   Estrogen_Status         3089 non-null   int64
 10  Progesterone_Status     3089 non-null   int64
 11  Regional_Node_Examined  3089 non-null   int64
 12  Reginol_Node_Positive   3089 non-null   int64
 13  Survival_Months         3089 non-null   int64
 14  Status                  3089 non-null   int64
dtypes: int64(13), object(2)
memory usage: 386.1+ KB
```

3.1.1 Cross tab

Cross tabs shows 5 year survival percentage between different attributes of the dataset.

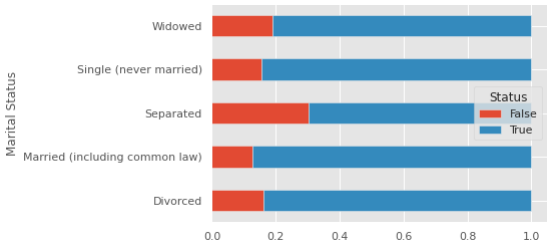
In [ ]:
```
# the cross tab operator provides an easy way to get these numbers

print(df_5y.groupby(by = 'A_Stage').Age.count())
print('====================================')
survival = pd.crosstab(df_5y['A_Stage'], df_5y.Status.astype(bool),normalize='index') # how to group
print(survival)

survival.plot(kind='barh', stacked=True)
plt.show()
```

```
A_Stage
0    3018
1      71
Name: Age, dtype: int64
====================================
Status      False      True
A_Stage
0        0.134526  0.865474
1        0.408451  0.591549
```



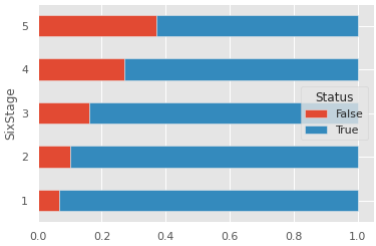
In [ ]:
```
# the cross tab operator provides an easy way to get these numbers

print(df_5y.groupby(by = 'Marital Status').Age.count())
print('====================================')
survival = pd.crosstab(df_5y['Marital Status'], df_5y.Status.astype(bool),normalize='index') # how to group
print(survival)

survival.plot(kind='barh', stacked=True)
plt.show()
```

```
Marital Status
Divorced                          386
Married (including common law)    2025
Separated                           33
Single (never married)             472
Widowed                            173
Name: Age, dtype: int64
====================================
Status                            False      True
Marital Status
Divorced                       0.163212  0.836788
Married (including common law)  0.126420  0.873580
Separated                      0.303030  0.696970
Single (never married)         0.154661  0.845339
Widowed                        0.190751  0.809249
```



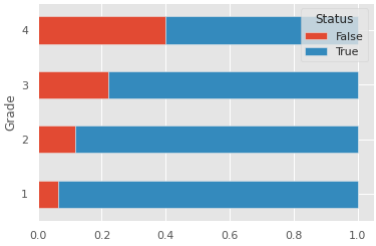
In [ ]:
```
# the cross tab operator provides an easy way to get these numbers

print(df_5y.groupby(by = 'SixStage').Age.count())
print('====================================')
survival = pd.crosstab(df_5y['SixStage'], df_5y.Status.astype(bool),normalize='index') # how to group
print(survival)

survival.plot(kind='barh', stacked=True)
plt.show()
```

```
SixStage
1    1022
2     846
3     797
4      52
5     372
Name: Age, dtype: int64
====================================
Status       False      True
SixStage
1         0.066536  0.933464
2         0.102837  0.897163
3         0.160602  0.839398
4         0.269231  0.730769
5         0.370968  0.629032
```



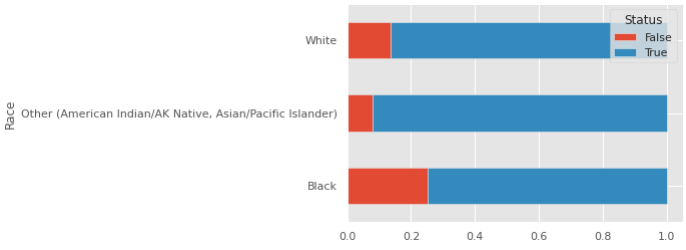
In [ ]:
```
# the cross tab operator provides an easy way to get these numbers

print(df_5y.groupby(by = 'Grade').Age.count())
print('====================================')
survival = pd.crosstab(df_5y['Grade'], df_5y.Status.astype(bool),normalize='index') # how to group
print(survival)

survival.plot(kind='barh', stacked=True)
plt.show()
```

```
Grade
1     406
2    1810
3     858
4      15
Name: Age, dtype: int64
====================================
Status     False      True
Grade
1       0.064039  0.935961
2       0.117680  0.882320
3       0.221445  0.778555
4       0.400000  0.600000
```



In [ ]:
```
# the cross tab operator provides an easy way to get these numbers

print(df_5y.groupby(by = 'Race').Age.count())
print('====================================')
survival = pd.crosstab(df_5y['Race'], df_5y.Status.astype(bool),normalize='index') # how to group
print(survival)

survival.plot(kind='barh', stacked=True)
plt.show()
```

```
Race
Black                                                227
Other (American Indian/AK Native, Asian/Pacific Islander)     241
White                                               2621
Name: Age, dtype: int64
====================================
Status                                             False      True
Race
Black                                           0.251101  0.748899
Other (American Indian/AK Native, Asian/Pacific...  0.078838  0.921162
White                                           0.136971  0.863029
```



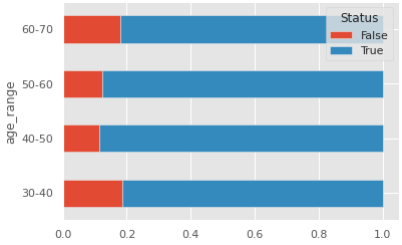
2. Which attributes correlate with each other?

This is an answer for the second question. 5-year survival percentage shows difference between races.

In [ ]:
```
df_5y['age_range'] = pd.cut(df_5y.Age,[30,40,50,60,1e6], labels=['30-40','40-50','50-60','60-70']) # this creates a new variable
#print(df_5y_gr_race[df_5y_gr_race.Status == 1].Age.count()/df_5y_gr_race.Age.count())
```

```
# the cross tab operator provides an easy way to get these numbers

print(df_5y.age_range.count())
print('==================================')
survival = pd.crosstab(df_5y['age_range'], df_5y.Status.astype(bool),normalize='index') # how to group
print(survival)

survival.plot(kind='barh', stacked=True)
plt.show()
```

```
3085
==================================
Status        False      True
age_range
30-40      0.185185  0.814815
40-50      0.113636  0.886364
50-60      0.123020  0.876980
60-70      0.180046  0.819954
```



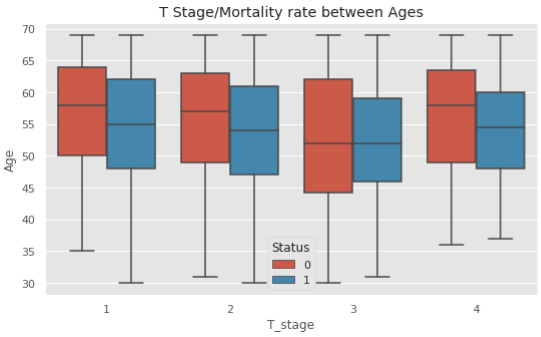
In [ ]:
```
# Let's break up the age variable
df['age_range'] = pd.cut(df.Age,[30,35,40,45,50,55,60,65,1e6], labels=['30-35','35-40','40-45','45-50','50-55','55-60','60-65','65+']) # this creates a new variable
df['surv_month_range'] = pd.cut(df.Survival_Months,[0,12,24,36,48,60,72,84,96,1e6], labels=['1','2','3','4','5', '6','7','8','9']) # this creates a new variable
df['tumor_size_range'] = pd.cut(df.Tumor_Size,[0,20,40,60,80,100,1e6], labels=['0-20','20-40','40-60','60-80','80-100','100']) # this creates a new variable

df.age_range.describe()
#df_grouped = df.groupby(by=['Race','age_range','Status'])
#df_grouped.Status.count()
df_grouped = df.groupby(by=['tumor_size_range','Status'])
df_grouped.tumor_size_range
```

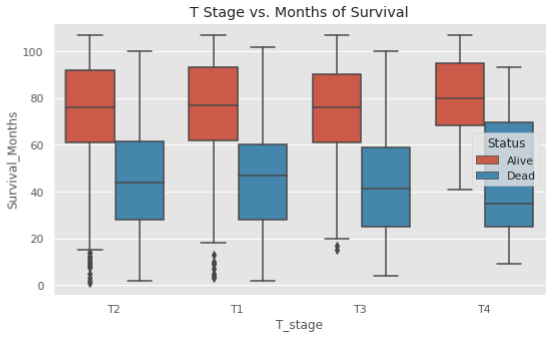
Out[ ]:
```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x7fdf9a103390>
```

3.1.2 Box Plots

The filtered dataset (5-year survival) was investigated grouped by their T-stage status and their 5-year survival status versus. age of patients. Non-filtered dataset was investigated by their Survival time.

In [ ]:
```
plt.subplots(figsize=(30,5))
plt.subplot(1,3,1)
sns.boxplot(x="T_stage", y="Age", hue="Status", data=df_5y)
plt.title('T Stage/Mortality rate between Ages')

plt.subplot(1,3,2)
sns.boxplot(x="T_stage", y="Survival_Months", hue="Status", data=df)
plt.title('T Stage vs. Months of Survival')

plt.show()
```



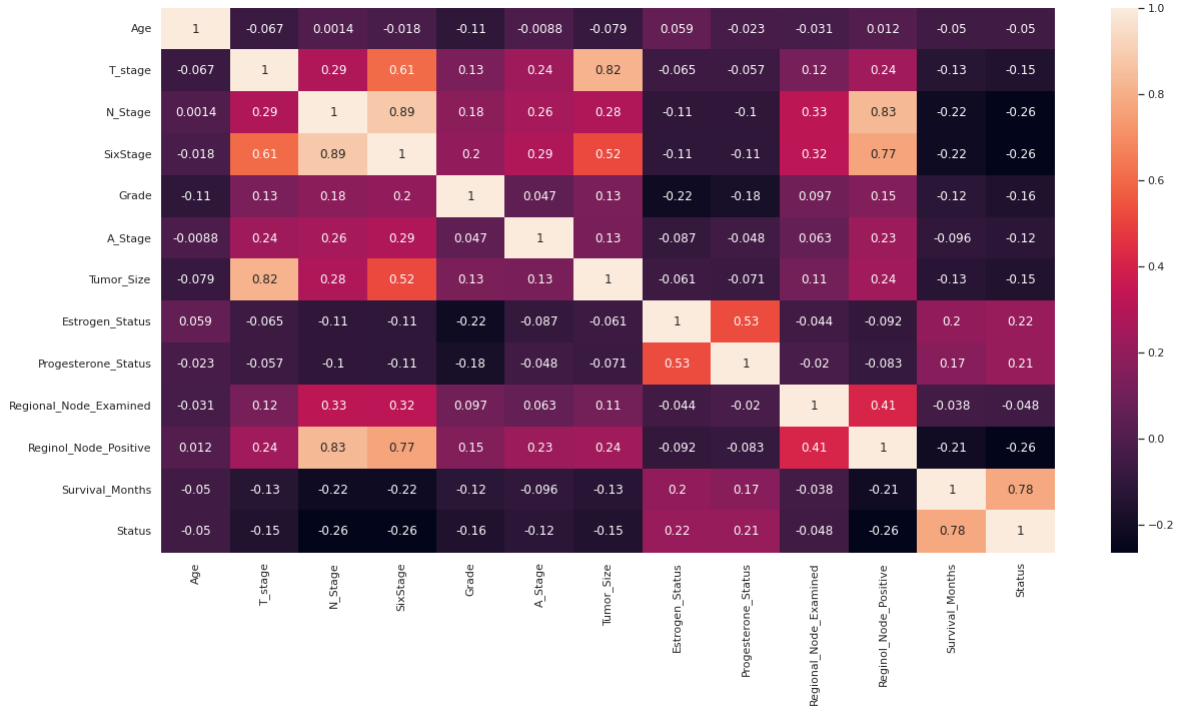


3.2 Data Relationship Exploration

3.2.1 Correlation Matrix

In [ ]:
```
import matplotlib.pyplot as plt
cmap = sns.set(style="darkgrid") # one of the many styles to plot using

f, ax = plt.subplots(figsize=(20, 10))
sns.heatmap(df_5y.corr(), cmap=cmap, annot=True)
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf99cbc390>
```



2. Which attributes correlate with each other?

As T stage is indicator for tumor size. They are highly corralated as expected. This applies to correlation between N stage and regional node positive.

Status and Survival Month highly correlates (positively), this indicates the patients that survive longer has higher chance to stay alive.

T stage positively corralated with N stage meaning that larger tumors tend to have more positive lymph nodes.

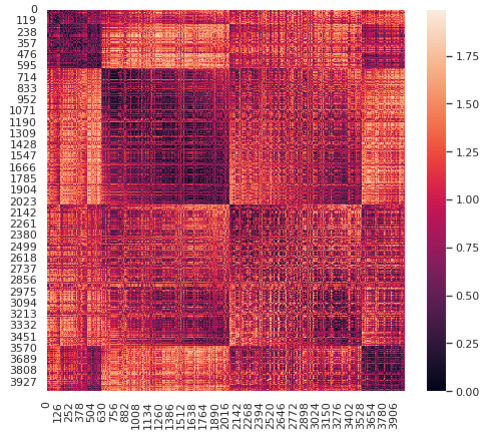
Having at least one hormone receptor positive (estrogen or progesterone) increases the chance of survival.

As T stage, N stage, sixth stage and grade are direct indicators of cancer progression, they all negatively correlate with the Survival Status.

In [ ]:
```
# Let's break up the age variable
df_5y['age_range'] = pd.cut(df_5y.Age,[30,40,50,60,1e6], labels=['30-40','40-50','50-60','60-70']) # this creates a new variable
df_5y['surv_month_range'] = pd.cut(df_5y.Age,[0,12,24,36,48,60,72,84,96,1e6], labels=['1 year','2 years','3 years','4 year','5 year', '6 year','7 years','8 year','9 year']) # this creates a
df_5y['tumor_size_range'] = pd.cut(df_5y.Tumor_Size,[0,20,40,60,80,100,1e6], labels=['0-20','20-40','40-60','60-80','80-100','100']) # this creates a new variable
df_5y['regional_node_positive_range'] = pd.cut(df_5y.Reginol_Node_Positive,[0,10,20,30,1e6], labels=['0-10','10-20','20-30','30-46']) # this creates a new variable
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.metrics.pairwise import pairwise_distances

f, ax = plt.subplots(figsize=(8, 7))
# Lets scale the data to be zero mean, unit variance
std = StandardScaler()
# and lets also sort the data. And copy + sort
df_2= df_2.copy().sort_values(by=['Status' , 'T_stage'])
# Transform etmek gerekiyor

vars_to_use = ['Status', 'Age', 'Tumor_Size', 'N_Stage' , 'T_stage' ,'Estrogen_Status', 'Progesterone_Status', 'Regional_Node_Examined', 'Reginol_Node_Positive', 'Survival_Months'] # pick v


xdata = pairwise_distances(std.fit_transform(df_2[vars_to_use].to_numpy()),
                           metric='correlation')
sns.heatmap(xdata, cmap=cmap, annot=False)
```

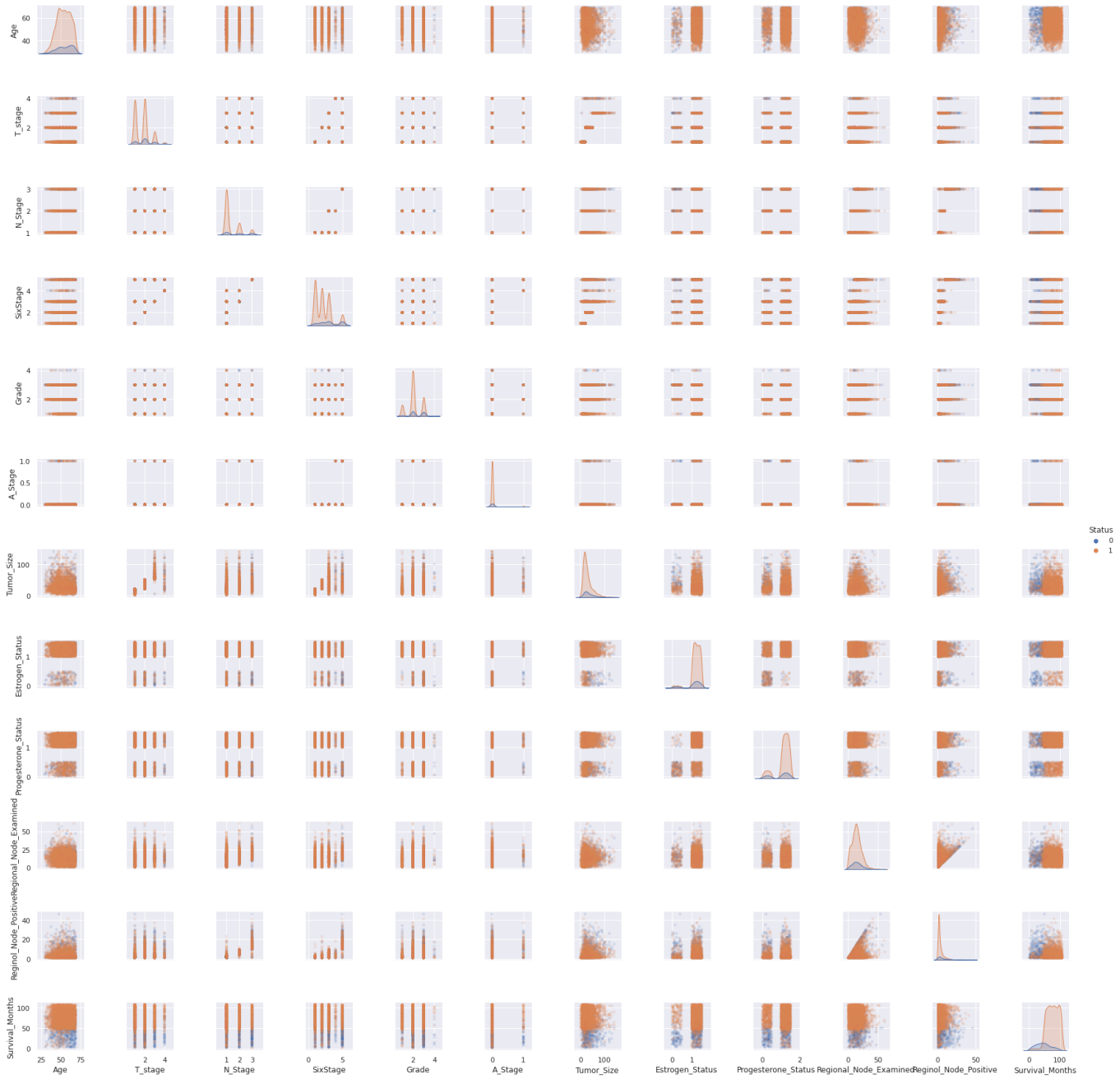
Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf99690a50>
```



3.2.2 Scatter plots except for department

In [ ]:
```
# Lets make a pretty plot of the scatter matrix
df_2_jitter = df_2.copy()
df_2_jitter[['Tumor_Size','Estrogen_Status','Progesterone_Status']] += np.random.rand(len(df_2_jitter),3)/2
sns.pairplot(df_2_jitter, hue="Status", height=2,
```

```
            plot_kws=dict(s=20, alpha=0.15, linewidth=0))
plt.show()
```



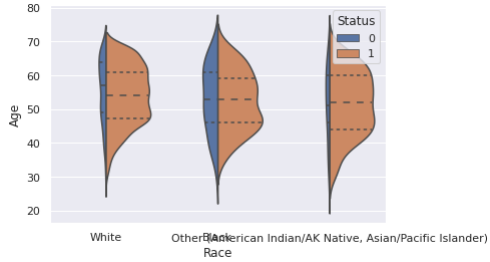
In [ ]:
```
df_5y.age_range.describe()
#df_grouped = df.groupby(by=['Race','age_range','Status'])
#df_grouped.Status.count()
df_grouped = df_5y.groupby(by=['tumor_size_range','Status'])
df_grouped.tumor_size_range
```

Out[ ]:
```
<pandas.core.groupby.generic.SeriesGroupBy object at 0x7fdf976a1e10>
```

3.2.3 Violin Plots

Violin plots was used to compare the five year survival status between different parameters.

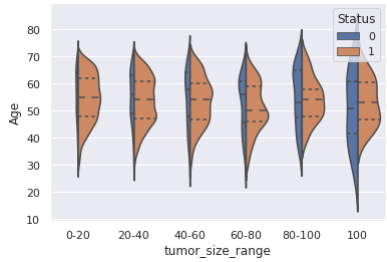
In [ ]:
```
sns.violinplot(x="Race", y="Age", hue="Status", data=df_5y,
               split=True, # split across violins
               inner="quart", # show innner stats like mena, IQR,
               scale="count") # scale the size of the plot by the count within each group
plt.show()
```



Violin plots indicates the mortality rates are significantly higher for black patients. Otherwise, the age distribution doesn't change much between the patients.

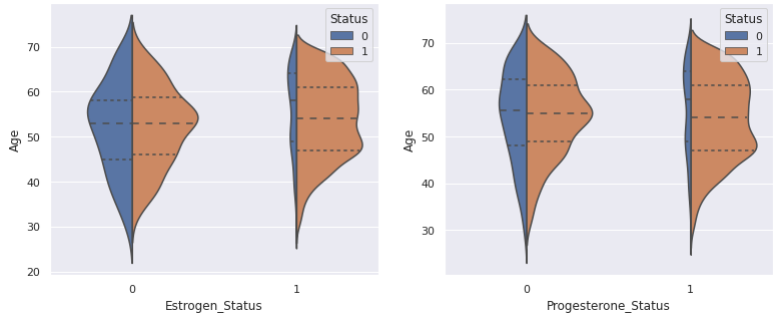
In [ ]:
```
sns.violinplot(x="tumor_size_range", y="Age", hue="Status", data=df_5y,
               split=True, # split across violins
               inner="quart", # show innner stats like mena, IQR,
               scale="count") # scale the size of the plot by the count within each group
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdf98046f50>
```



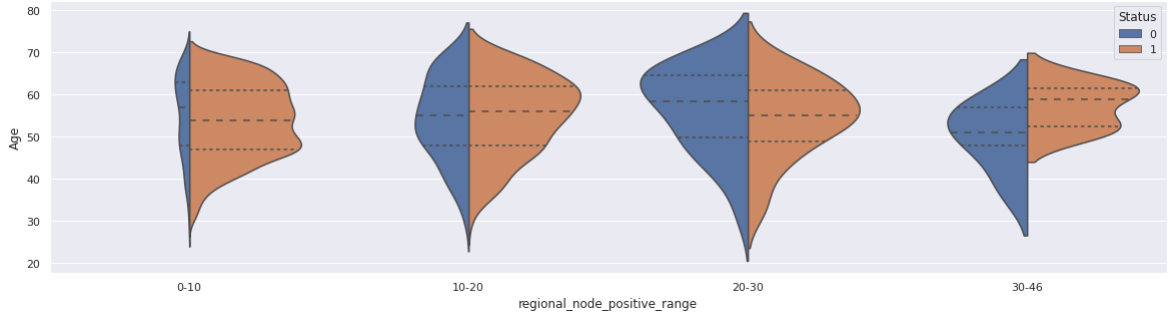
It can be seen that as the tumor size increases, for all ages in general, the mortality rates increase.

In [ ]:
```
fig = plt.figure(figsize=(20,5))

plt.subplot(1,3,1)
sns.violinplot(x="Estrogen_Status", y="Age", hue="Status", data=df_5y,
               split=True, # split across violins
               inner="quart", # show innner stats like mena, IQR,
               scale="count") # scale the size of the plot by the count within each group

plt.subplot(1,3,2)
sns.violinplot(x="Progesterone_Status", y="Age", hue="Status", data=df_5y,
               split=True, # split across violins
               inner="quart", # show innner stats like mena, IQR,
               scale="count") # scale the size of the plot by the count within each group
plt.show()
```



Estrogen and Progestrone positivity highly increases 5 year survival rates.

In [ ]:
```
fig = plt.figure(figsize=(20,5))

sns.violinplot(x="regional_node_positive_range", y="Age", hue="Status", data=df_5y,
               split=True, # split across violins
               inner="quart", # show innner stats like mena, IQR,
               scale="count") # scale the size of the plot by the count within each group

plt.show()
```



As expected, the number of positive regional nodes increases mortality.

4. Dimensionality Reduction

4.1 UMAP - Uniform Manifold Approximation and Projection for Dimension Reduction

Installing the UMAP module:

In [ ]:
```
!pip install umap-learn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: umap-learn in /usr/local/lib/python3.7/dist-packages (0.5.3)
Requirement already satisfied: numba>=0.49 in /usr/local/lib/python3.7/dist-packages (from umap-learn) (0.56.0)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.7/dist-packages (from umap-learn) (1.0.2)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from umap-learn) (4.64.0)
Requirement already satisfied: pynndescent>=0.5 in /usr/local/lib/python3.7/dist-packages (from umap-learn) (0.5.7)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from umap-learn) (1.21.6)
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.7/dist-packages (from umap-learn) (1.7.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from numba>=0.49->umap-learn) (57.4.0)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from numba>=0.49->umap-learn) (4.12.0)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.7/dist-packages (from numba>=0.49->umap-learn) (0.39.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from pynndescent>=0.5->umap-learn) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.22->umap-learn) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->numba>=0.49->umap-learn) (4.1.1)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->numba>=0.49->umap-learn) (3.8.1)
```

In [ ]:
```
#Choosing the attributes to feed the UMAP algorithm.
df_5y_int = df_5y[['Age', 'SixStage', 'Grade', 'Tumor_Size',
                   'Estrogen_Status', 'Progesterone_Status', 'Survival_Months', 'Status']]
df_5y_int.info()

## UMAP reducer
reducer = umap.UMAP(
    n_components=2, n_neighbors=20, random_state=42, transform_seed=42, verbose=False
)
reducer.fit(df_5y_int)
reduced_dy_5y = reducer.transform(df_5y_int)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3089 entries, 1203 to 1124
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Age                  3089 non-null   int64
 1   SixStage             3089 non-null   int64
 2   Grade                3089 non-null   int64
 3   Tumor_Size           3089 non-null   int64
 4   Estrogen_Status      3089 non-null   int64
 5   Progesterone_Status  3089 non-null   int64
 6   Survival_Months      3089 non-null   int64
 7   Status               3089 non-null   int64
dtypes: int64(8)
memory usage: 281.7 KB
```

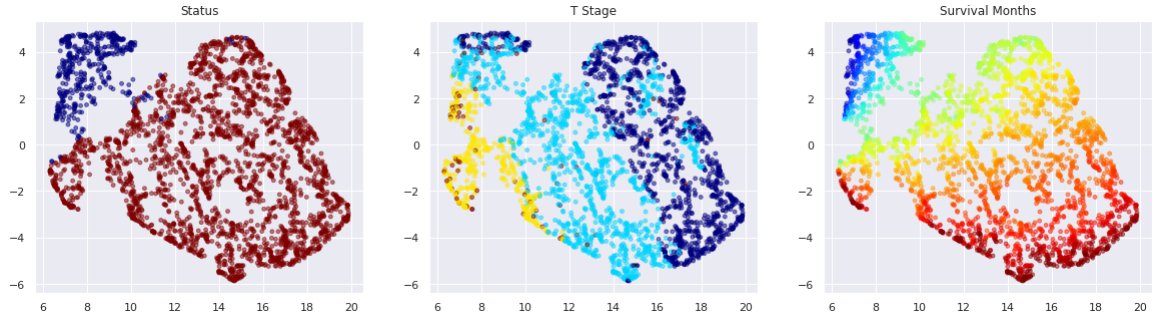
UMAP algorithm was used to reduce the 8 dimensional dataset to 2.

In [ ]:
```
# dimension was reduced to 2.
reduced_dy_5y.shape[1]
```

Out[ ]:
```
2
```

Plotting UMAP 2D reduced data:

In [ ]:
```
#plt.scatter(reduced_dy_5y[:, 0], reduced_dy_5y[:, 1], s=15, c=df_5y.T_stage, alpha=0.5)
#plt.scatter(reduced_dy_5y[:, 0], reduced_dy_5y[:, 1], s=15, c=df_5y.T_stage, alpha=0.5)
#plt.show()

# 2D UMAP plots
plt.subplots(figsize=(20, 5))

plt.subplot(1,3,1)
plt.scatter(reduced_dy_5y[:, 0], reduced_dy_5y[:, 1], s=15, c=df_5y.Status, cmap = 'jet', alpha=0.5)
plt.title('Status')

plt.subplot(1,3,2)
plt.scatter(reduced_dy_5y[:, 0], reduced_dy_5y[:, 1], s=15, c=df_5y.T_stage, cmap = 'jet', alpha=0.5)
plt.title('T Stage')

plt.subplot(1,3,3)
plt.scatter(reduced_dy_5y[:, 0], reduced_dy_5y[:, 1], s=15, c=df_5y.Survival_Months, cmap = 'jet', alpha=0.5)
plt.title('Survival Months')

plt.show()
```



- We observe 2 discrete clusters, especially capturing the Status (Dead/Alive) of the patients. Top left region clusters the patients didn't survive 5 years perfectly.
- T stage is an indicator of the tumor size and was seen to advance with the increasing UMAP dimensions.
- Gradual change in survival months can be seen on the UMAP reduced dataset.

1. Can we predict survival status of patients?

- UMAP clustering model shows we could manage to cluster patients based on their survival status without feeding the model with status information.

In [ ]: