Siddharth Patel

19BCP130

MACHINE LEARNING LAB - 1 (August 2022, Week -1)

Python Practice

Q1. Write a function that converts a decimal number to binary number.

```
# your code here
def dec_to_bin(x):
    lst = []
    while x>=1:
        lst.append(x%2)
        x = x//2
    lst = lst[::-1]
    return ''.join(str(i) for i in lst)
dec_to_bin(10)
```

```
'1010'
```

```
5%2
```

```
1
```

Q2. Write a function to compute the sigmoid of a vector of values.
A sigmoid of a real number x is defind as $\sigma(x) = \frac{1}{1+e^{-}x}$
For example: Input: $[x_1, x_2, x_3, x_4]$
Output: $[\sigma(x_1), \sigma(x_2), \sigma(x_3), \sigma(x_4)]$

```
# Your code here.
import math

def sigmoid_cal(x):
    result = []
    for i in range(len(x)):
        result.append(1/(1+math.exp(-(x[i]))))
    return result

input_x = [13.56, 23.45, 16, 2, 4]
sigmoid_cal(input_x)
```

```
[0.9999987088810224,
 0.9999999999345675,
 0.9999998874648379,
 0.8807970779778823,
 0.9820137900379085]
```

Q3. Write a function to compute the derivative of the sigmoid function with respect to its input x. Here, the x is a vector.
The derivative of a sigmoid is defined as: $\sigma(x)(1-\sigma(x))$
TRY TO PROVE THIS IN YOUR COPY

```
# Your code here
from sympy import *
from sympy.abc import x
def my_function(equation):
    equation = equation
    var = 'x'
    x = symbols("x")
    y = symbols("y")
    return diff(equation, var)
ans = my_function('1/(1+exp(-(x)))')
print(ans)

inp = 16
rslt_1 = sigmoid_cal([inp])[0]*(1-sigmoid_cal([inp])[0])
rslt_2 = ans.evalf(subs={x:inp})
rslt_1, rslt_2
```

```
exp(-x)/(1 + exp(-x))**2
(1.1253514941371313e-07, 1.12535149390932e-7)
```

Q4. Write a function to normalize the rows of a matrix. After applying this function to an input matrix x of size $m \times n$, each row of x should be a vector of unit length.

```
# Your code here.
import numpy as np
m = int(input('Enter number of rows: \n'))
n = int(input('Enter number of columns: \n'))
mat = []
for i in range(m):
    mat.append([])
    for j in range(n):
        z = int(input(f'Enter value at ({m, n})'))
        mat[i].append(z)

npA = np.array(mat)
ans = np.linalg.norm(npA)
npA/ans
```

```
array([[0.40824829, 0.40824829, 0.40824829],
       [0.40824829, 0.40824829, 0.40824829]])
```

Q5. Write a program which can map() and filter() to make a list whose elements are cube of even numbers in a list.

```
# Your code here
def cube(x):
```

```
    return x**3
def even_cube(lst):
    return list(map(cube, filter(lambda x:x%2==0, lst)))

even_cube([1,2,3,4,5,6,7,8,9])
```

```
[8, 64, 216, 512]
```

Q6. consider the marks list of class students given two lists
Students =
['student1','student2','student3','student4','student5','student6','student7','student8','student9','student10']
Marks = [45, 78, 12, 14, 48, 43, 45, 98, 35, 80]
from the above two lists the Student[0] got Marks[0], Student[1] got Marks[1] and so on

your task is to print the name of students
a. Who got top 5 ranks, in the descending order of marks
b. Who got least 5 ranks, in the increasing order of marks
d. Who got marks between >25th percentile <75th percentile, in the increasing order of marks

```
Ex 1:
Students=['student1','student2','student3','student4','student5','student6','student7','student8',
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]
a.
student8  98
student10 80
student2  78
student5  48
student7  47
b.
student3 12
student4 14
student9 35
student6 43
student1 45
c.
student9 35
student6 43
student1 45
student7 47
student5 48
```

```
# your code here
Students = ['student1','student2','student3','student4','student5','student6','student7','student8',
Marks = [45, 78, 12, 14, 48, 43, 47, 98, 35, 80]

listz = [list(i) for i in zip(Students, Marks)]
listz.sort(key=lambda x:x[1], reverse=True)
```

```
listz
print('Top 5:')
for i in range(5):
    print(listz[-(len(listz)-i)])
print('\n')
print('Bottom 5:')
for i in range(5):
    print(listz[-i-1])

print('\n')
print('Between 25 to 75 percentile:')
listz.sort(key=lambda x:x[1])
min_marks = listz[0][1]
max_marks = listz[-1][1]
difference = int(max_marks-min_marks)
per_25 = difference*0.25
per_75 = difference*0.75

for i in range(len(listz)):
        if int(listz[i][1])<per_75 and int(listz[i][1])>per_25:
            print(listz[i])
```

```
Top 5:
['student8', 98]
['student10', 80]
['student2', 78]
['student5', 48]
['student7', 47]


Bottom 5:
['student3', 12]
['student4', 14]
['student9', 35]
['student6', 43]
['student1', 45]


Between 25 to 75 percentile:
['student9', 35]
['student6', 43]
['student1', 45]
['student7', 47]
['student5', 48]
```

Q7. consider you have given n data points in the form of list of tuples like S=[(x1,y1),(x2,y2),(x3,y3),(x4,y4),(x5,y5),..,(xn,yn)] and a point P=(p,q)
your task is to find 5 closest points(based on cosine distance) in S from P <brs (x,y) and (p,q) is defind as $cos^{-1}(\frac{(x\cdot p+y\cdot q)}{\sqrt{(x^2+y^2)}\cdot\sqrt{(p^2+q^2)}})$>cosine distance between two point

```
Ex:

S= [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1)(6,0),(1,-1)]
P= (3,-4)

Output:
(6,-7)
```

```
(1,-1)
(6,0)
(-5,-8)
(-1,-1)
```

```
# your code here
import math

def getClosestPoints(S, P):
  distances = []

  p = P[0]
  q = P[1]

  for s in S:
    x = s[0]
    y = s[1]

    distance = math.acos( (x * p + y * q) / ( math.sqrt( x ** 2 + y ** 2 ) * math.sqrt( p ** 2 + q *
    distances.append([distance, (x, y)])

  distances.sort(key = lambda x: x[0])

  for i in range(5):
    print(distances[i][1])


SList = [(1,2),(3,4),(-1,1),(6,-7),(0, 6),(-5,-8),(-1,-1),(6,0),(1,-1)]
PPoint = (3,-4)

getClosestPoints(SList, PPoint)
```

```
(6, -7)
(1, -1)
(6, 0)
(-5, -8)
(-1, -1)
```

Q8: Given two sentences S1, S2 You will be given a list of lists, each sublist will be of length 2 i.e. [[x,y],[p,q], [l,m]..[r,s]] consider its like a matrix of n rows and two columns

a. the first column Y will contain interger values

b. the second column Y_{score} will be having float values

Your task is to find the value of

$f(Y, Y_{score}) = -1 * \frac{1}{n}\Sigma_{foreachY,Y_{score}pair}(Ylog10(Y_{score}) + (1-Y)log10(1-Y_{score}))$ here n is the number of rows in the matrix

```
Ex:
[[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1, 0.8]]
output:
0.4243099
```

$\frac{-1}{8} \cdot ((1 \cdot log_{10}(0.4) + 0 \cdot log_{10}(0.6)) + (0 \cdot log_{10}(0.5) + 1 \cdot log_{10}(0.5)) + \ldots + (1 \cdot log_{10}(0.8) + 0$

```
import math

def calc(inList):
  Y = [x[0] for x in inList]
  Yscore = [x[1] for x in inList]

  innerSummation = 0

  for i in range(len(inList)):
    innerSummation += (Y[i] * math.log10(Yscore[i])) + (1 - Y[i]) * math.log10(1 - Yscore[i])

  ans = (-1) * ( innerSummation ) / len(Y)

  return ans

inputList = [[1, 0.4], [0, 0.5], [0, 0.9], [0, 0.3], [0, 0.6], [1, 0.1], [1, 0.9], [1, 0.8]]

calc(inputList)
```

```
0.42430993457031635
```

Assignment on Exploratory data analysis

Download Haberman Cancer Survival dataset from Kaggle. You may have to create a Kaggle account to donwload data. (https://www.kaggle.com/gilsousa/habermans-survival-data-set) or you can also run the below cell and load the data directly.

```
import pandas as pd
df = pd.read_csv('haberman.csv')
df.head()
```

	age	year	nodes	status
0	30	64	1	1
1	30	62	3	1
2	30	65	0	1
3	31	59	2	1
4	31	65	4	1

1.1 Analyze high level statistics of the dataset: number of points, numer of features, number of classes, data-points per class.

- You have to write all of your observations in Markdown cell with proper formatting.
- Do not write your observations as comments in code cells.

```
import numpy as np
```

```
print(f"Number of points: {len(df)}")
```

```
Number of points: 306
```

Here, 306 is number of rows in dataset which implies number of data points.

```
print(f"Number of features: {len(df.columns)-1}")
```

```
Number of features: 3
```

As target column is not counted as feature, we remove that from length to get number of features.

```
print(f"Number of classes: {df['status'].nunique()}")
```

```
Number of classes: 2
```

Number of classes in `status` are 2.

```
print(df['status'].value_counts())
```

```
1    225
2     81
Name: status, dtype: int64
```

Here, we have 225 values of status code 1 and 81 values of status code 2 which means dataset is not balanced.

1.2 - Explain the objective of the problem.

(The objective for a problem can be defined as a brief explanation of problem that you are trying to solve using the given dataset)

As per Kaggle: The dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer.

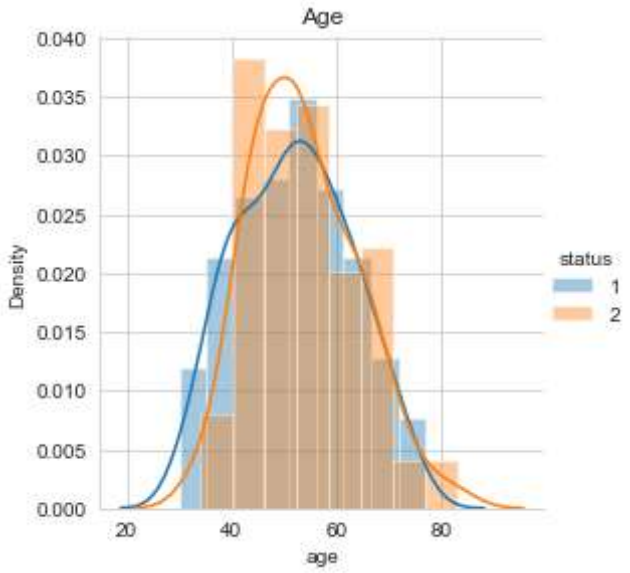
Objective: It is classification type of problem and can be solved with supervised methods. Target is to identify if the person will survive or not based on given parameters about person.

1.3 Perform Univariate analysis - Plot PDF, CDF, Boxplot.

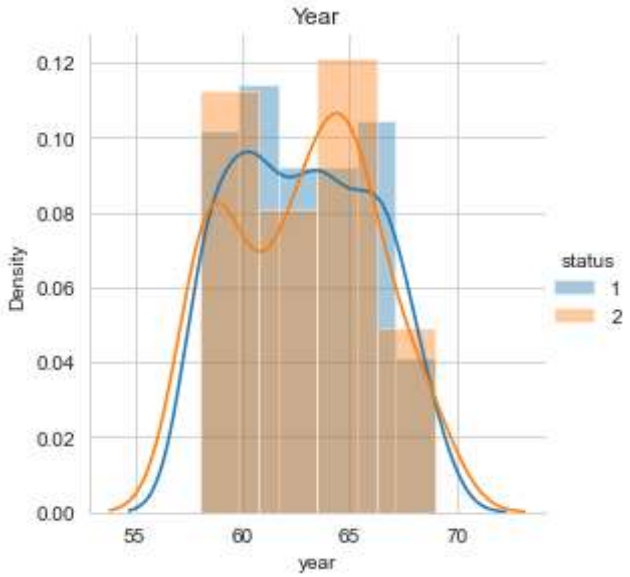
- Plot the required charts to understand which feature are important for classification.
- Make sure that you add titles, legends and labels for each and every plots.
- Do write observations/inference for each plot.

To ignore warnings of versions of libraries and possibility of function depriciation, we use warnings.

```
import warnings
warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt
import seaborn as sns
sns.FacetGrid(df, hue="status", height=4) \
   .map(sns.distplot, "age") \
   .add_legend().set(title='Age')
plt.show()
```

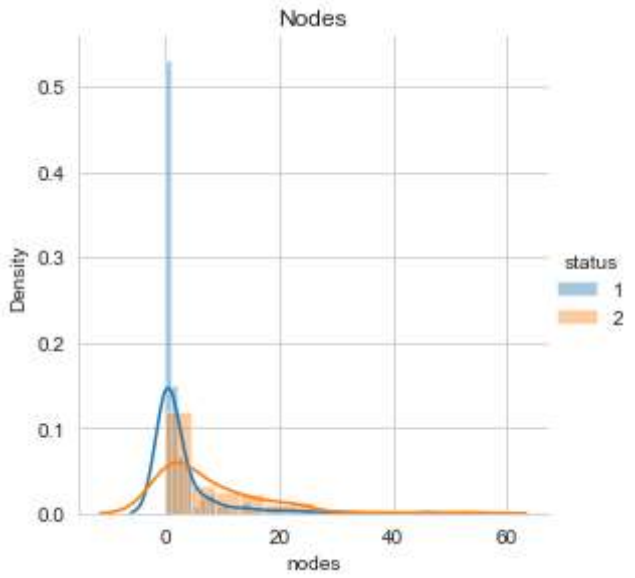



```
from turtle import title
import matplotlib.pyplot as plt
import seaborn as sns
sns.FacetGrid(df, hue="status", height=4) \
   .map(sns.distplot, "year") \
   .add_legend().set(title='Year')
plt.show()
```

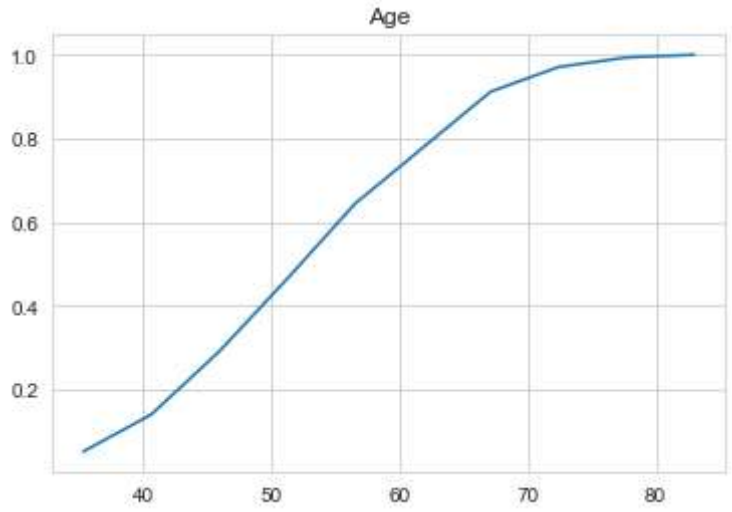



```
import matplotlib.pyplot as plt
import seaborn as sns
sns.FacetGrid(df, hue="status", height=4) \
   .map(sns.distplot, "nodes") \
```

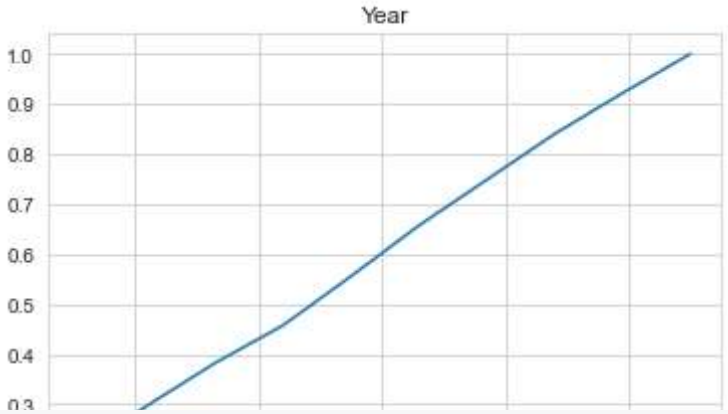
```
   .add_legend().set(title='Nodes')
plt.show()
```



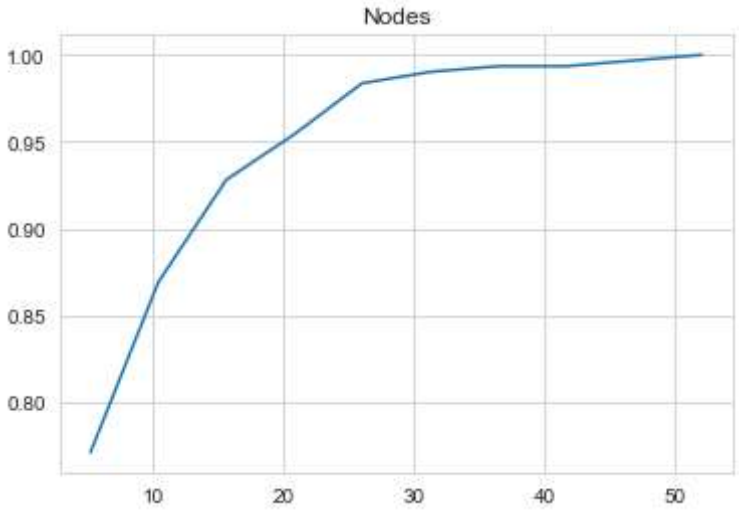

```
counts, bin_edges = np.histogram(df['age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
cdf = np.cumsum(pdf)
plt.title("Age")
plt.plot(bin_edges[1:], cdf)

plt.show()
```



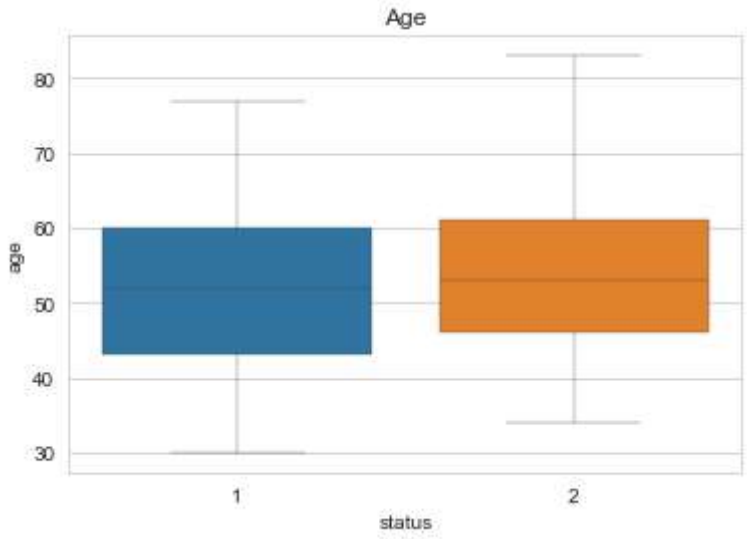

```
counts, bin_edges = np.histogram(df['year'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
cdf = np.cumsum(pdf)
plt.title("Year")
plt.plot(bin_edges[1:], cdf)

plt.show()
```



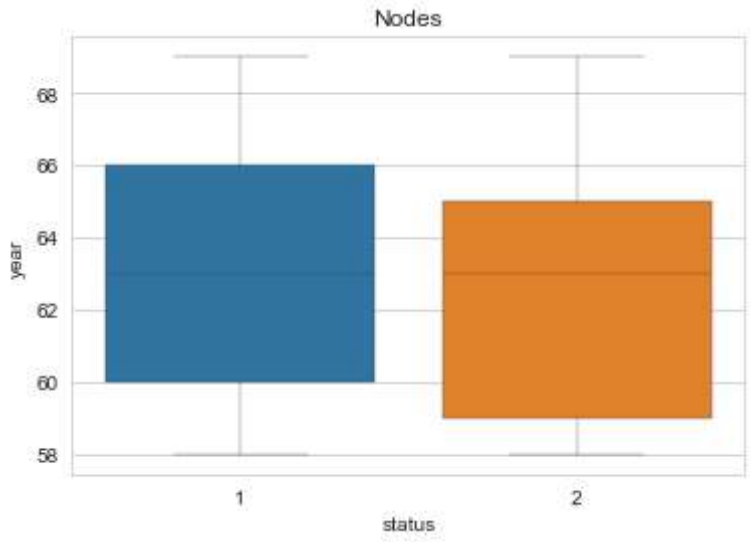

```
counts, bin_edges = np.histogram(df['nodes'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
cdf = np.cumsum(pdf)
plt.title("Nodes")
plt.plot(bin_edges[1:], cdf)

plt.show()
```



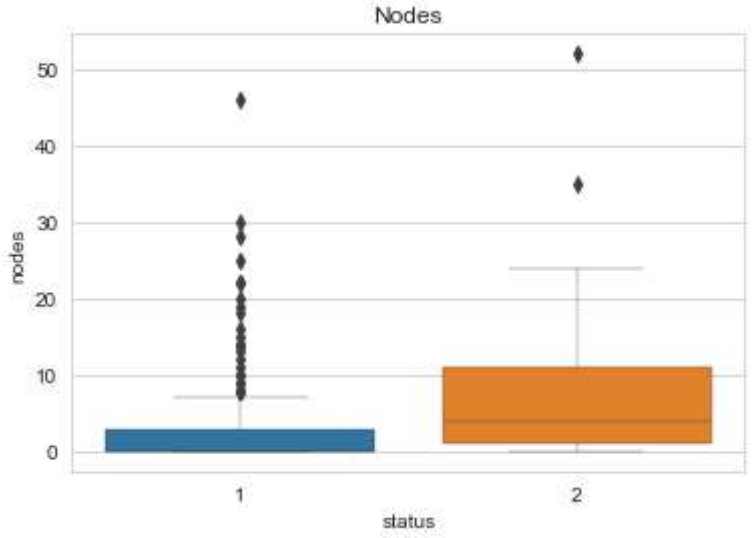

```
sns.boxplot(x='status',y='age', data=df, linewidth=0.35).set(title='Age')
plt.show()
```




```
sns.boxplot(x='status',y='year', data=df, linewidth=0.35).set(title='Nodes')
plt.show()
```




```
sns.boxplot(x='status',y='nodes', data=df, linewidth=0.35).set(title='Nodes')
plt.show()
```

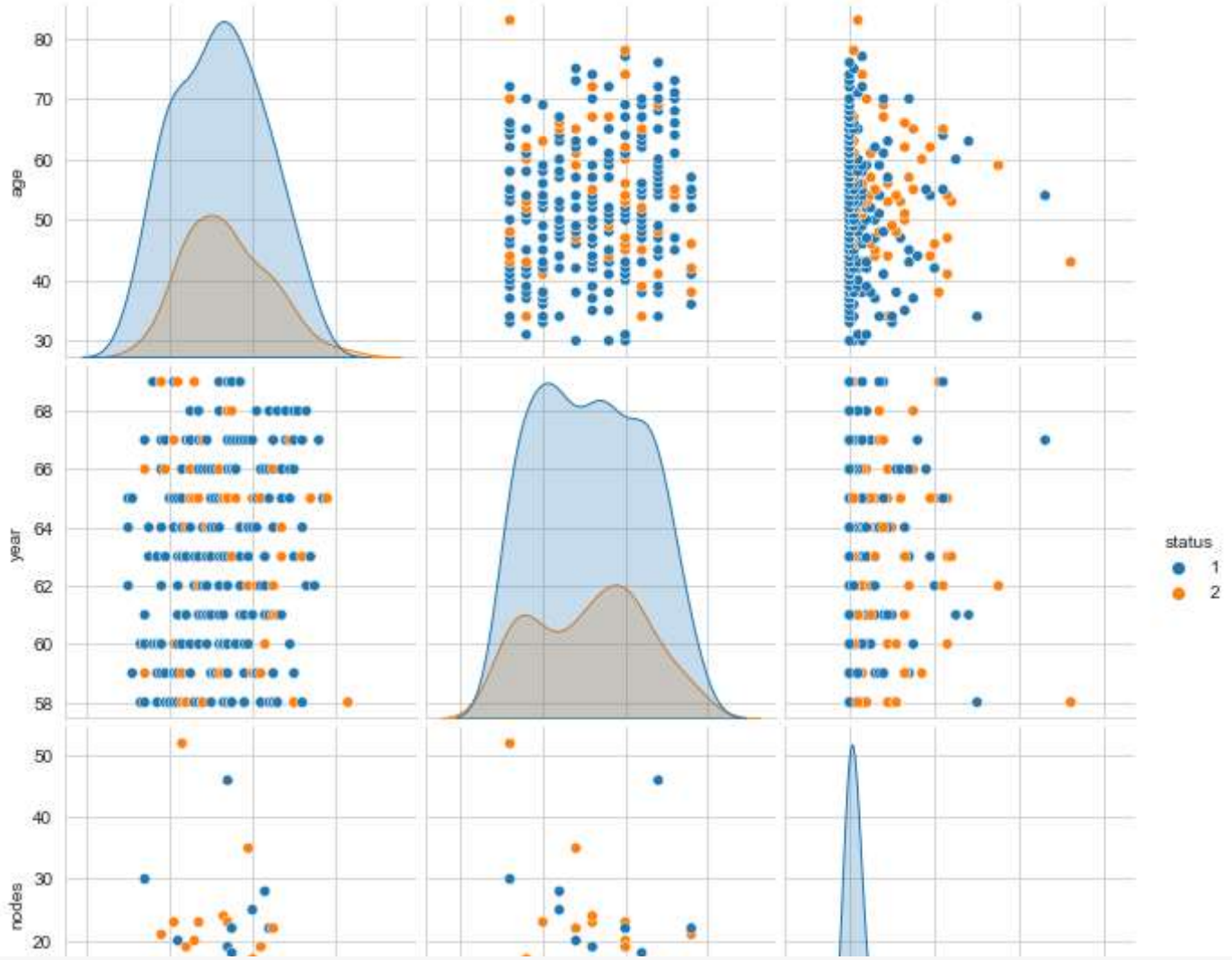


It is visible that classes are overlapping and very difficult to saperate linearly. Status and Nodes having lots of outliers as per Box plot.Status vs Year is evenly distributed.

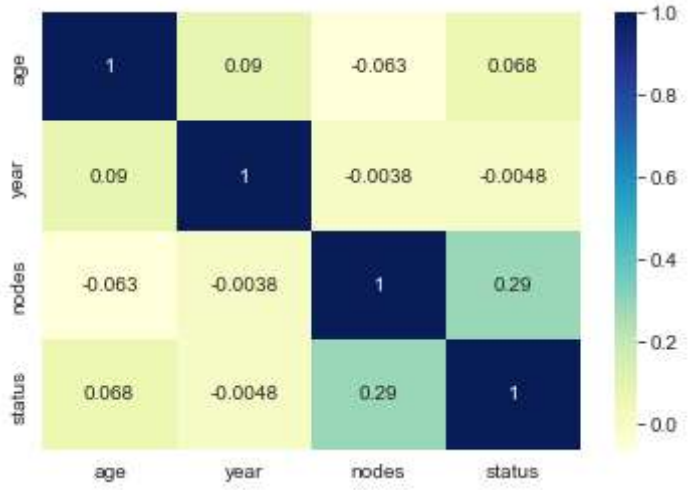
1.4 Perform Bivariate analysis - Plot 2D Scatter plots and Pair plots

- Plot the required Scatter plots and Pair plots of different features to see which combination of features are useful for clasification task
- Make sure that you add titles, legends and labels for each and every plots.
- Do write observations/inference for each plot.

```
sns.set_style("whitegrid")
sns.pairplot(df, hue="status", size=3, palette='tab10')
plt.show()
```




```
dataplot = sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)
plt.show()
```



This pairplot contains all required and possible scatter plots. It is visible that features are extremely related to each other and hence I plotted correlation heatmap to check the same.
Features are not very correlated and mixed so very difficult to put conventional lines to seperate classes. If one wants to go for classification, it is possible to test with trying to generate new features with given one.

1.5 Summarize your final conclusions of the Exploration

- You can desrcibe the key features that are important for the Classification task.
- Try to quantify your results i.e. while writing observations include numbers,percentages, fractions etc.

- Write a brief of your exploratory analysis in 3-5 points

1. In the classification task, the first thing we need to check is that type of classification (binary or categorical). According to this dataset the problem is type of binary classification. Here, types are 1 and 2 which means we need to create binary classifier if we want to go for classification.
2. Another thing which is important for classification problem is whether the dataset is biased or not. If the dataset is biased then the problem is that our model will also biased with the class which has more number of samples. Here, dataset is pretty biased and needs to be balanced before feeding to model.
3. If the dataset contains large amount of outliers then it will affect your model's prediction. You can findout the number of outliers by visualization and you have to deal with these outliers. As we know Nodes is having many outliers, we need to handel them.
4. As per assignment's instruction we perform all visualization tasks and noticed that:
    - This dataset is biased with class `1`
    - Graphs showing classes are very overlapping and hence it is difficult to create simple hyperplane to seperate classes.
    - Data features are not very correlated to target.
    - Usually when dataset features are not visibally saperable, one can go for XGBoost or Random Forest etc. algorithms as they perform very good with tabular data.