CSE17040 - Distance Measures

July 28, 2020

1 MLDM Lab 2

- CB.EN.U4CSE17040

[1]:
```
from scipy.spatial.distance import hamming
from scipy.spatial.distance import euclidean
from scipy.spatial.distance import cityblock
from scipy.spatial import minkowski_distance as minkowski
from math import *

import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from random import sample
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
```

1.1 Hamming Distance

[2]:
```
def hamming_distance(a, b):
    return sum(abs(item1 - item2) for item1, item2 in zip(a, b)) / len(a)
```

[3]:
```
row1 = [0, 0, 0, 0, 0, 1]
row2 = [0, 0, 0, 0, 1, 0]
hamming_distance(row1, row2)
```

[3]: 0.3333333333333333

[4]:
```
hamming(row1, row2)
```

[4]: 0.3333333333333333

1.2 Euclidean Distance

[5]:
```
def euclidean_distance(a, b):
    return sqrt(sum((item1-item2)**2 for item1, item2 in zip(a,b)))
```

[6]:
```
row1 = [10, 20, 15, 10, 5]
row2 = [12, 24, 18, 8, 7]
euclidean_distance(row1, row2)
```

[6]: 6.082762530298219

[7]:
```
euclidean(row1, row2)
```

[7]: 6.082762530298219

1.3 Manhattan Distance

[8]:
```
def manhattan_distance(a, b):
    return sum(abs(item1-item2) for item1, item2 in zip(a,b))
```

[9]:
```
manhattan_distance(row1, row2)
```

[9]: 13

[10]:
```
cityblock(row1, row2)
```

[10]: 13

1.4 Minkowski Distance

[11]:
```
def minkowski_distance(a, b, p):
    return sum(abs(e1-e2)**p for e1, e2 in zip(a,b))**(1/p)
```

[12]:
```
minkowski_distance(row1, row2, 1)
```

[12]: 13.0

[13]:
```
minkowski_distance(row1, row2, 2)
```

[13]: 6.082762530298219

[14]:
```
minkowski(row1, row2, 1)
```

[14]: 13.0

[15]:
```python
minkowski(row1, row2, 2)
```

[15]: 6.082762530298219

1.5 Cosine Similarity

[16]:
```python
def square_rooted(x):
    return round(sqrt(sum([a*a for a in x])),3)
```

[17]:
```python
def cosine_similarity(x,y):
    numerator = sum(a*b for a,b in zip(x,y))
    denominator = square_rooted(x)*square_rooted(y)
    return round(numerator/float(denominator),4)
```

[18]:
```python
cosine_similarity(row1, row2)
```

[18]: 0.9932

1.6 Jaccard Similarity

[19]:
```python
def jaccard_similarity(x,y):
    intersection_cardinality = len(set.intersection(*[set(x), set(y)]))
    union_cardinality = len(set.union(*[set(x), set(y)]))
    return intersection_cardinality/float(union_cardinality)
```

[20]:
```python
jaccard_similarity(row1, row2)
```

[20]: 0.0

[21]:
```python
jaccard_similarity([0,1,2,5,6],[0,2,3,5,7,9])
```

[21]: 0.375

2 Loan Status

[22]:
```python
data = pd.read_csv("loan_status.csv")
data.head()
```

[22]:

	grade	sub_grade	loan_status	purpose
0	B	B2	Fully Paid	credit_card
1	C	C4	Charged Off	car
2	C	C5	Fully Paid	small_business

```
3     C        C1    Fully Paid           other
4     B        B5    Fully Paid           other
```

[23]:
```
data.describe()
```

[23]:
```
       grade sub_grade loan_status             purpose
count     50        50          50                  50
unique     6        19           2                  10
top        B        B3  Fully Paid  debt_consolidation
freq      21         6          39                  22
```

[24]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   grade        50 non-null     object
 1   sub_grade    50 non-null     object
 2   loan_status  50 non-null     object
 3   purpose      50 non-null     object
dtypes: object(4)
memory usage: 1.7+ KB
```

[25]:
```
data.dtypes
```

[25]:
```
grade          object
sub_grade      object
loan_status    object
purpose        object
dtype: object
```

[26]:
```
data_crosstab = pd.crosstab(data['grade'],data['loan_status'], margins = False)
data_crosstab
```

[26]:
```
loan_status  Charged Off  Fully Paid
grade
A                      1          11
B                      5          16
C                      3           8
D                      1           3
E                      0           1
F                      1           0
```

[27]:
```
data_crosstab = pd.crosstab(data['purpose'],data['loan_status'], margins =␣
 ↪False)
```

```
data_crosstab
```

[27]:

```
loan_status         Charged Off  Fully Paid
purpose
car                           1           1
credit_card                   0           8
debt_consolidation            4          18
home_improvement              0           1
major_purchase                1           1
medical                       0           1
moving                        0           1
other                         4           5
small_business                1           2
wedding                       0           1
```

[28]:

```
data_crosstab = pd.crosstab([data['grade'],␣
↪data['purpose']],data['loan_status'], margins = False)
data_crosstab
```

[28]:

```
loan_status               Charged Off  Fully Paid
grade purpose
A     credit_card                   0           1
      debt_consolidation            1           7
      major_purchase                0           1
      other                         0           1
      wedding                       0           1
B     credit_card                   0           6
      debt_consolidation            1           5
      major_purchase                1           0
      medical                       0           1
      moving                        0           1
      other                         3           2
      small_business                0           1
C     car                           1           0
      credit_card                   0           1
      debt_consolidation            2           4
      home_improvement              0           1
      other                         0           1
      small_business                0           1
D     debt_consolidation            0           2
      other                         1           1
E     car                           0           1
F     small_business                1           0
```

[29]:

```
le = LabelEncoder()
df = data.copy()
```

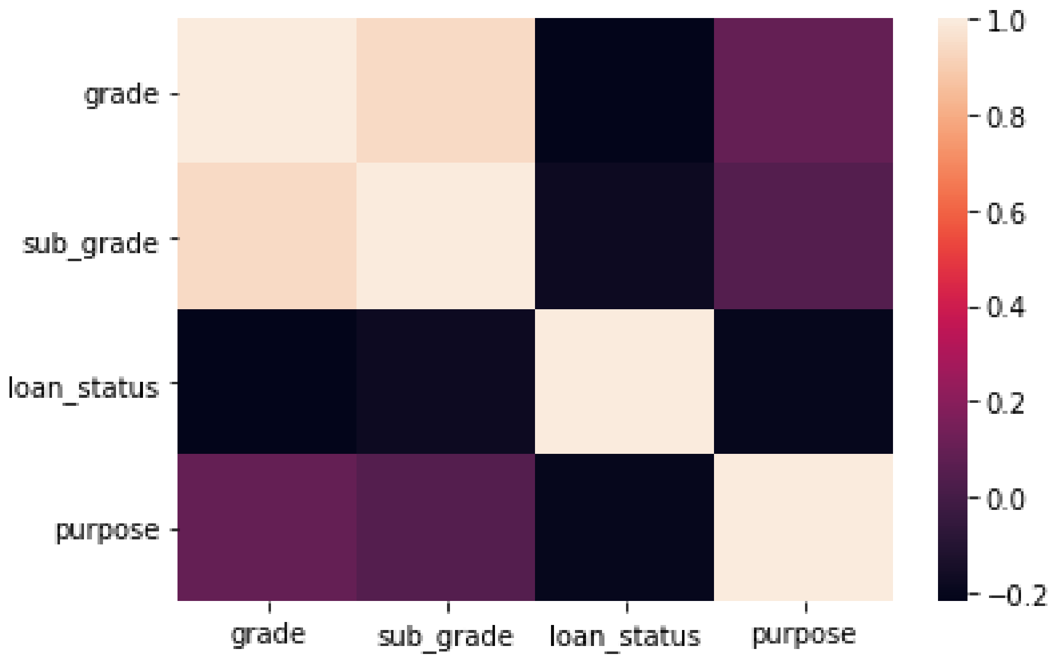
```
df['grade'] = le.fit_transform(data['grade'])#, 'sub_grade', 'purpose'␣
 ↪,'loan_status']])
df['sub_grade'] = le.fit_transform(data['sub_grade'])
df['purpose'] = le.fit_transform(data['purpose'])
df['loan_status'] = le.fit_transform(data['loan_status'])
df.head()
```

[29]:

	grade	sub_grade	loan_status	purpose
0	1	5	1	1
1	2	12	0	0
2	2	13	1	8
3	2	9	1	7
4	1	8	1	7

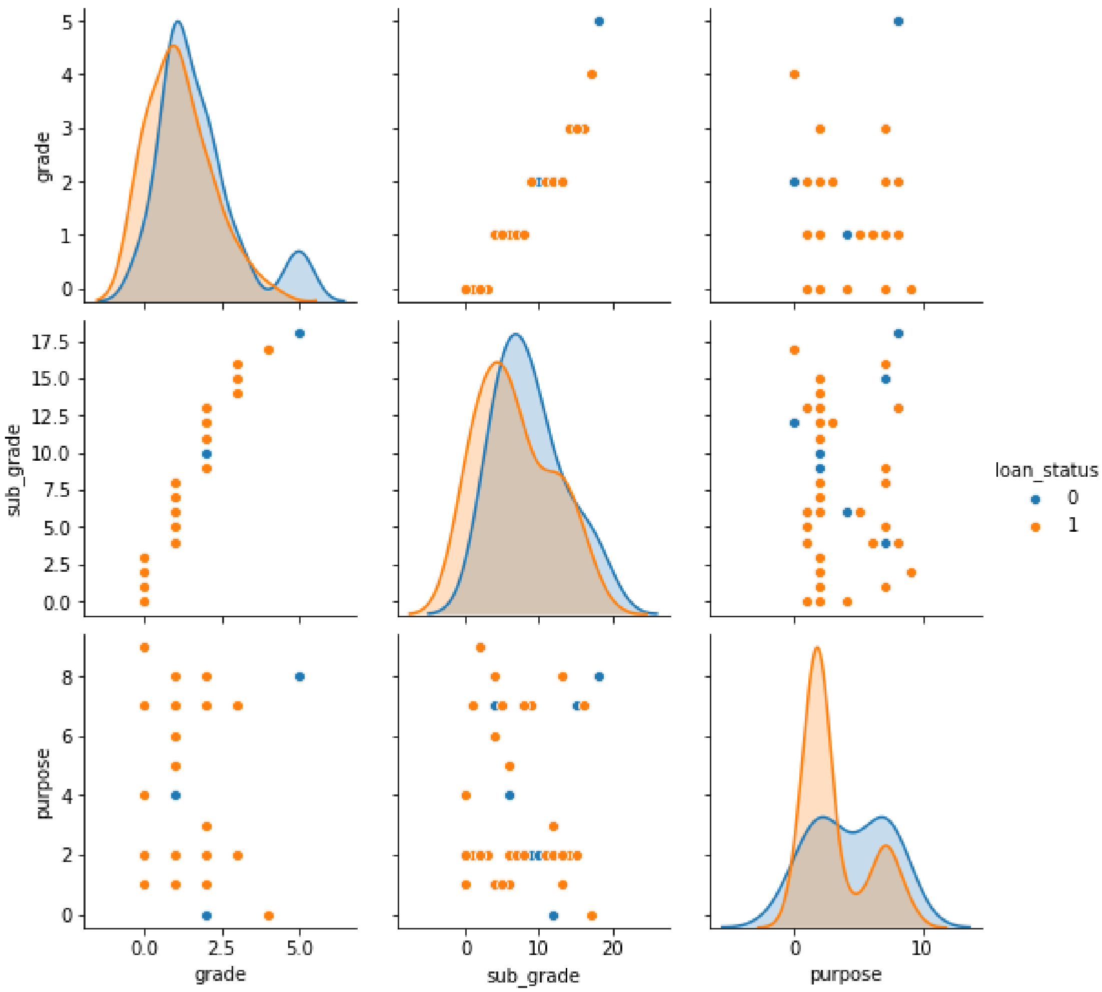
[30]:
```
sns.heatmap(df.corr())
```

[30]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcf2f18b250>



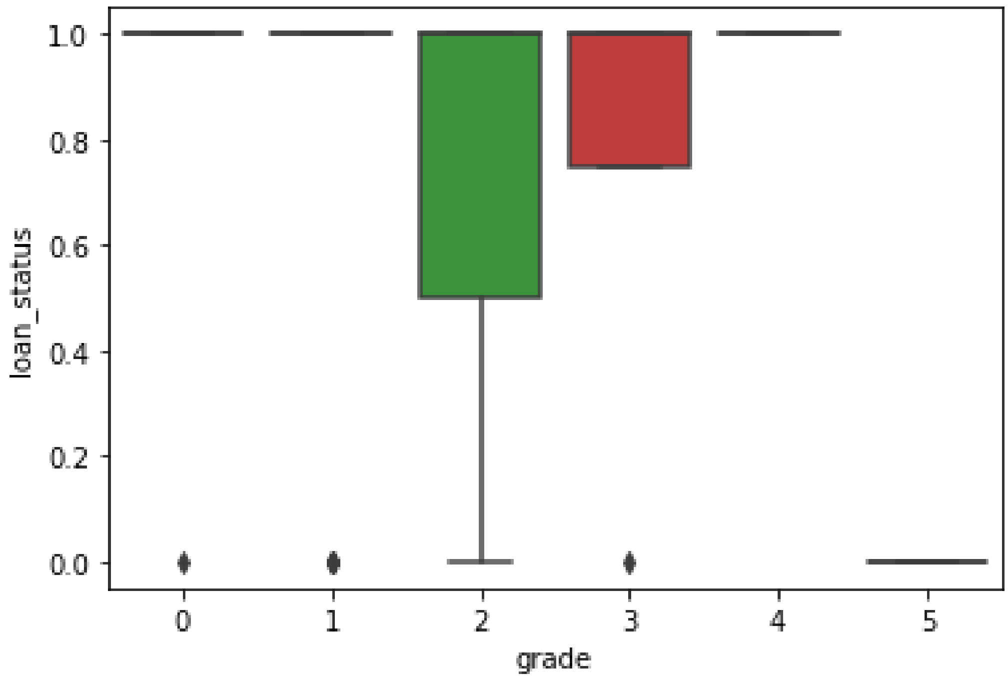
[31]:
```
sns.pairplot(df, hue='loan_status')
```

[31]: <seaborn.axisgrid.PairGrid at 0x7fcf2c9cec90>



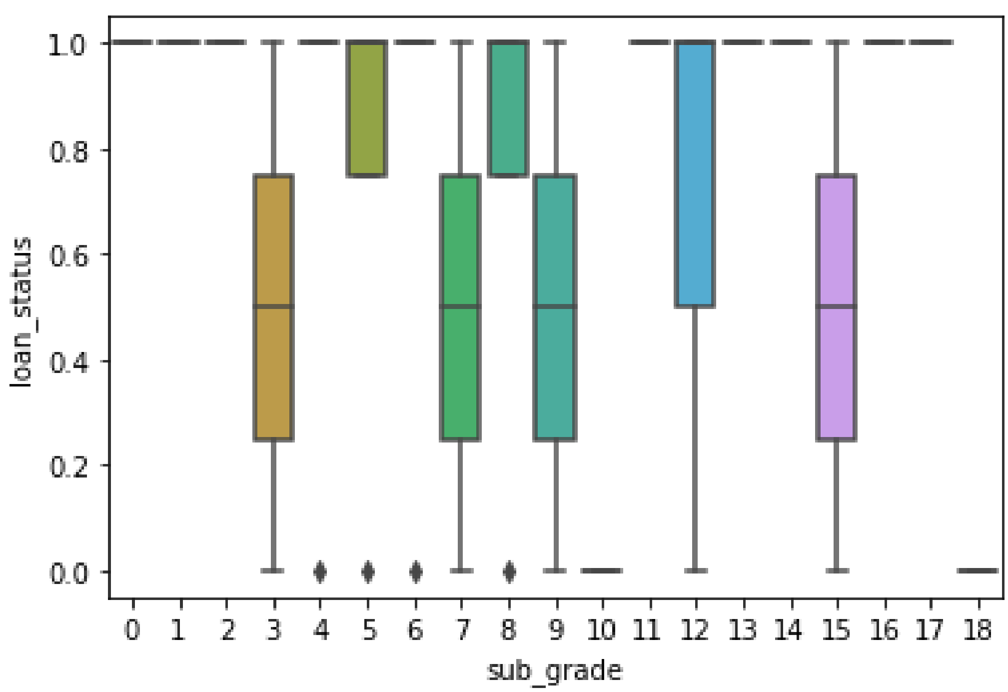
[32]:
```
sns.boxplot(df['grade'], df['loan_status'])
```

[32]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcf5df5ab50>



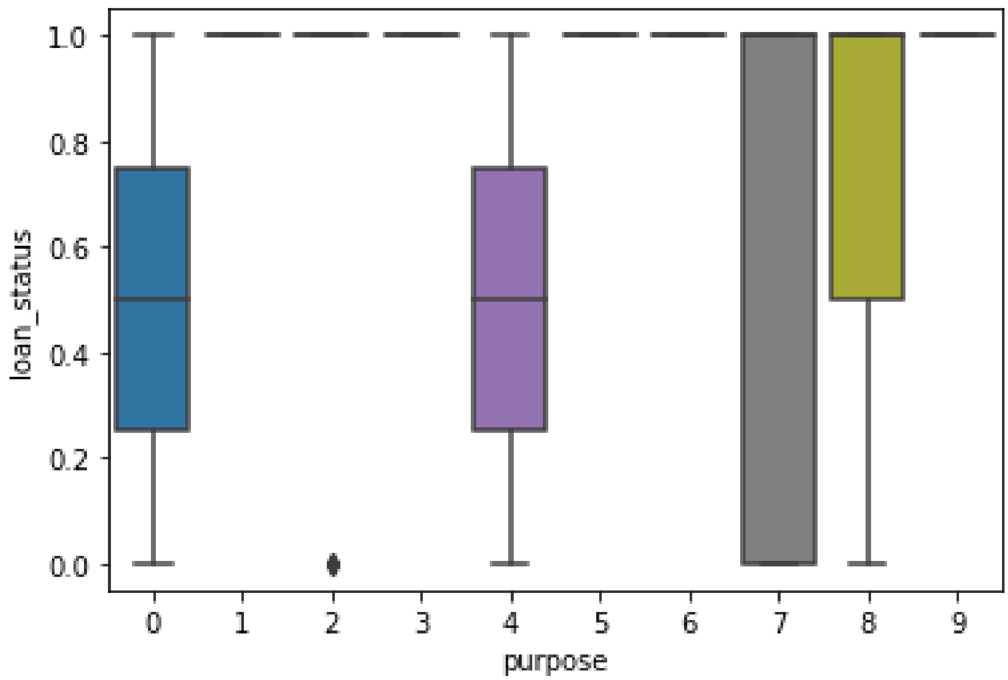
[33]:
```
sns.boxplot(df['sub_grade'], df['loan_status'])
```

[33]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcf2c2a1d10>



[34]:
```
sns.boxplot(df['purpose'], df['loan_status'])
```

[34]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcf2c076b50>



3 Data1.csv

[35]:
```
data = pd.read_csv('Data-1.csv')
data.head()
```

[35]:

	Country	Age	Salary	Purchased	Color
0	France	44.0	72000	No	Red
1	Spain	27.0	48000	Yes	Yellow
2	Germany	30.0	54000	No	Green
3	Spain	38.0	61000	No	Green
4	Germany	40.0	na	Yes	Yellow

[36]:
```
data.describe()
```

[36]:

	Age
count	114.000000
mean	39.842105

```
std      11.831845
min      27.000000
25%      31.000000
50%      38.000000
75%      44.000000
max      89.000000
```

[37]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 120 entries, 0 to 119
Data columns (total 5 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Country    120 non-null    object
 1   Age        114 non-null    float64
 2   Salary     120 non-null    object
 3   Purchased  108 non-null    object
 4   Color      120 non-null    object
dtypes: float64(1), object(4)
memory usage: 4.8+ KB
```

[38]:
```python
data.Country.unique()
```

[38]:
```
array(['France', 'Spain', 'Germany'], dtype=object)
```

[39]:
```python
data.isnull().sum()
```

[39]:
```
Country       0
Age           6
Salary        0
Purchased    12
Color         0
dtype: int64
```

[40]:
```python
data['Age'].fillna(method='ffill', inplace=True)
data['Purchased'].fillna(method='ffill', inplace=True)
```

[41]:
```python
int(data['Salary'][0])
```

[41]:
```
72000
```

[42]:
```python
data.isnull().sum()
```

[42]:
```
Country      0
Age          0
Salary       0
```

```
Purchased    0
Color        0
dtype: int64
```

[43]:
```
for i in range(len(data['Salary'])):
    try:
        data['Salary'][i] = int(data['Salary'][i])
    except:
        data['Salary'][i] = 0
```

```
/home/vkmanojk/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
/home/vkmanojk/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

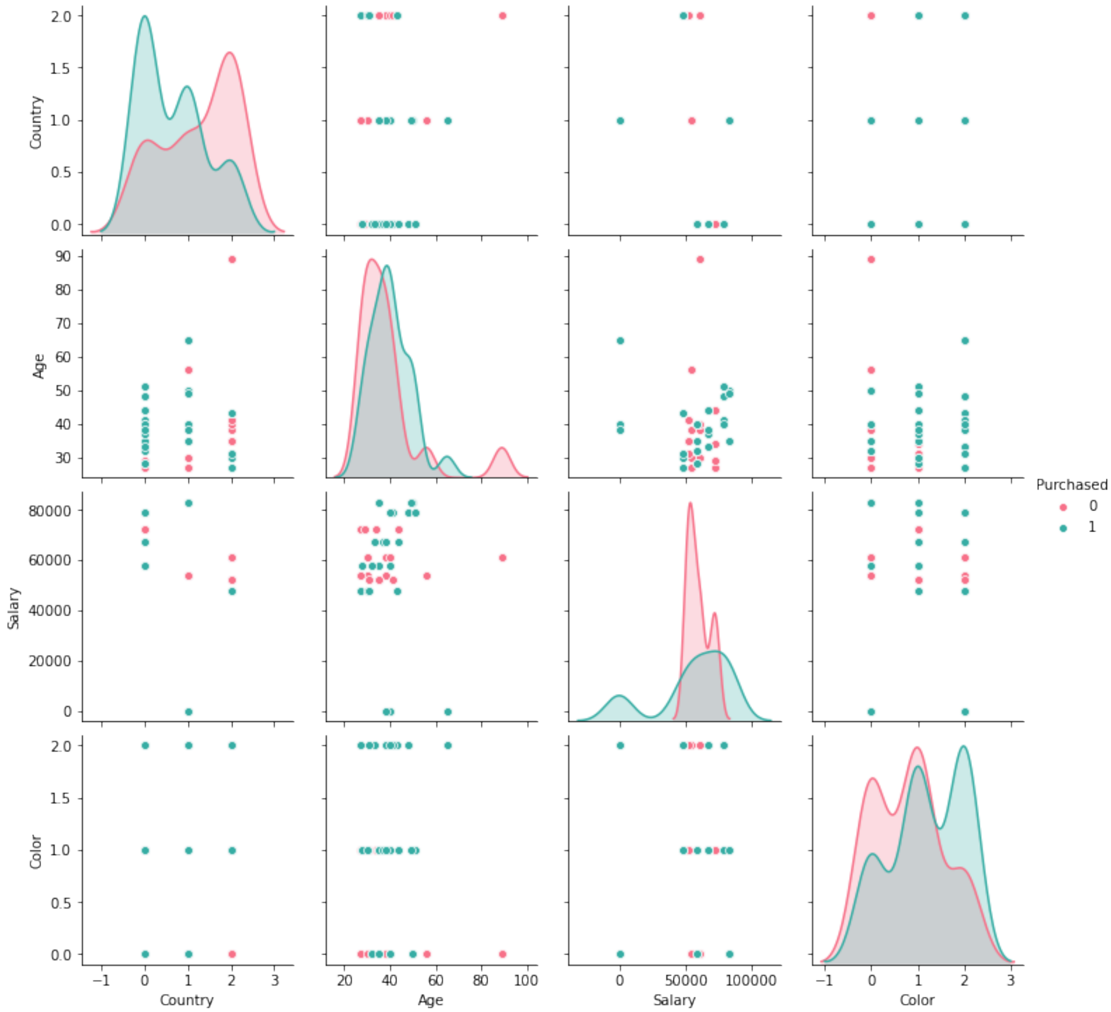
[44]:
```
df = data.copy()
df['Country'] = le.fit_transform(data['Country'])
df['Purchased'] = le.fit_transform(data['Purchased'])
df['Color'] = le.fit_transform(data['Color'])
df.head()
```

[44]:

	Country	Age	Salary	Purchased	Color
0	0	44.0	72000	0	1
1	2	27.0	48000	1	2
2	1	30.0	54000	0	0
3	2	38.0	61000	0	0
4	1	40.0	0	1	2

[45]:
```
sns.pairplot(df, hue = 'Purchased',palette="husl")
```

[45]: <seaborn.axisgrid.PairGrid at 0x7fcf2bf2ddd0>



3.1 Cross tab

[46]:
```
pd.crosstab(data['Country'], data['Purchased'])
```

[46]:
```
Purchased  No  Yes
Country
France     12   36
Germany    12   24
Spain      24   12
```

[47]:
```
pd.crosstab(data['Age'], data['Purchased'])
```

[47]:
```
Purchased  No  Yes
Age
```

```
27.0          6    3
28.0          0    3
29.0          3    0
30.0          6    3
31.0          3    3
32.0          0    3
33.0          0    3
34.0          3    0
35.0          6    6
37.0          0    3
38.0          6    6
40.0          3   12
41.0          3    3
43.0          0    3
44.0          3    3
48.0          0    3
49.0          0    3
50.0          0    6
51.0          0    3
56.0          3    0
65.0          0    3
89.0          3    0
```

[48]:
```
pd.crosstab(data['Salary'], data['Purchased'])
```

[48]:
```
Purchased  No  Yes
Salary
0           0   12
48000       0   12
52000      12    0
54000      12    0
58000       0   12
61000      12    0
67000       0   12
72000      12    0
79000       0   12
83000       0   12
```

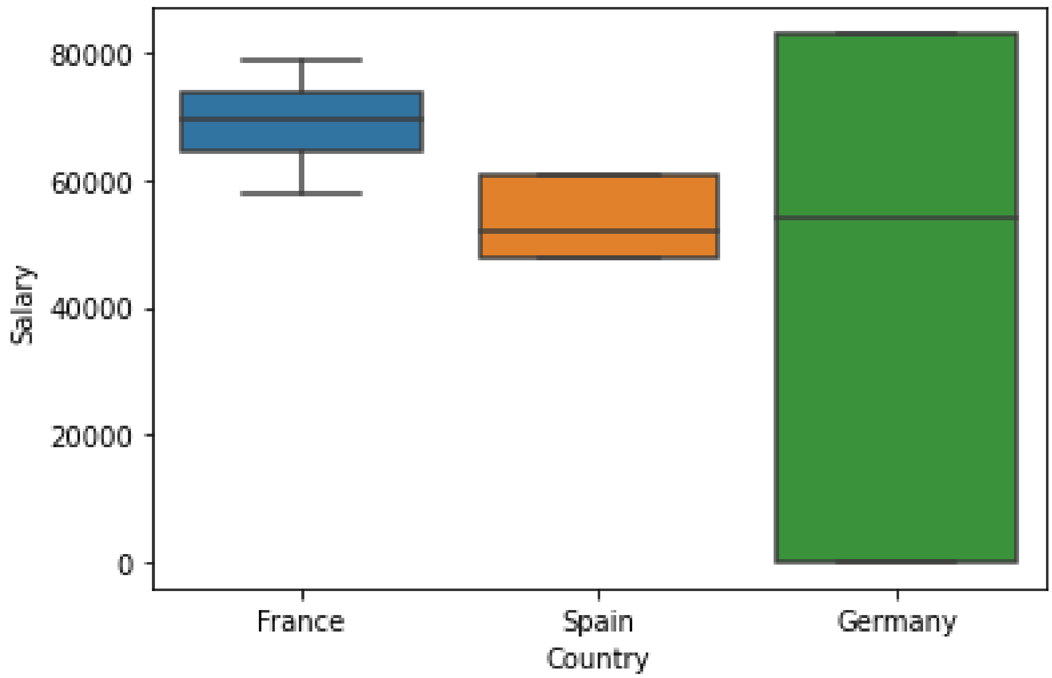
[49]:
```
pd.crosstab(data['Color'], data['Purchased'])
```

[49]:
```
Purchased  No  Yes
Color
Green      18   15
Red        21   27
Yellow      9   30
```

3.2 Box plot

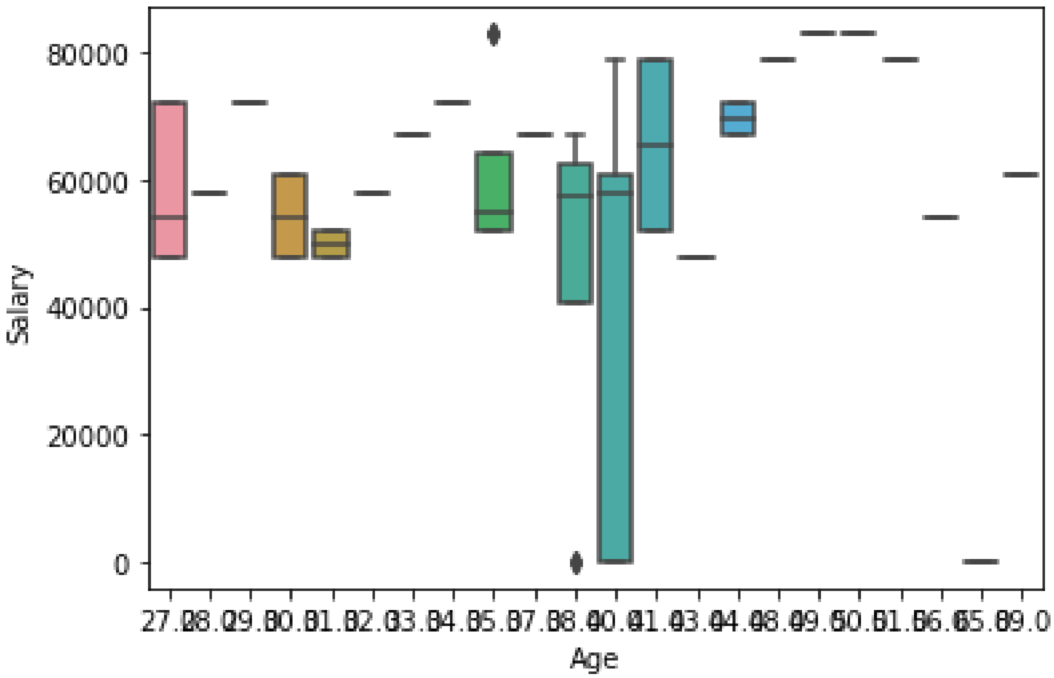
[50]:
```
sns.boxplot(data['Country'], data['Salary'])
```

[50]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcf2bf31550>



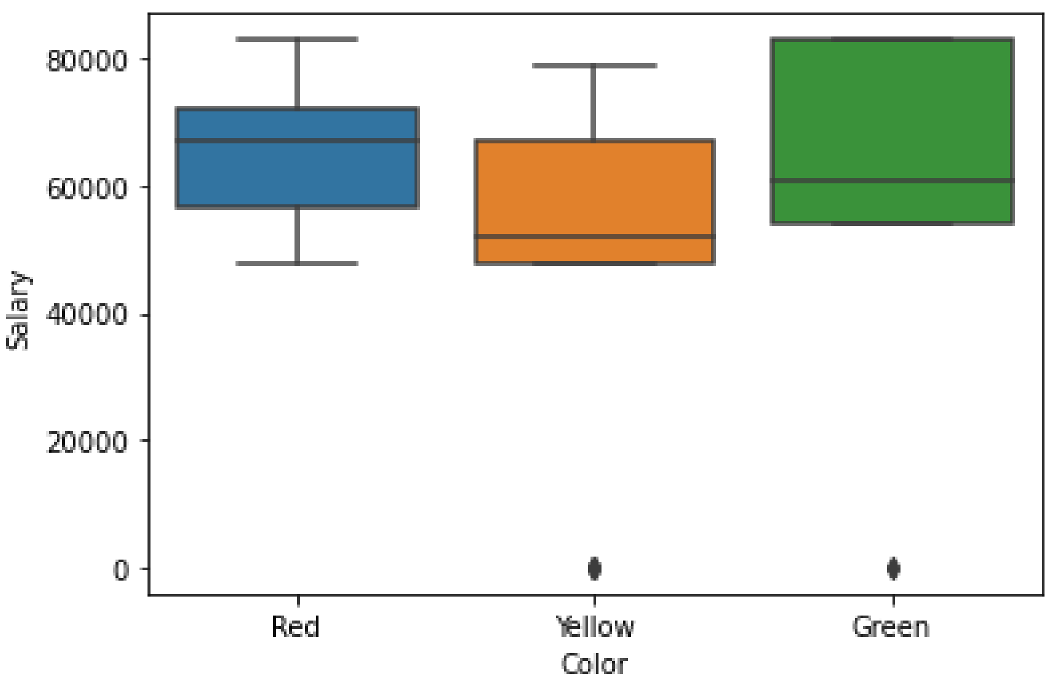
[51]:
```
sns.boxplot(data['Age'], data['Salary'])
```

[51]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcf298d8bd0>



[52]:
```
sns.boxplot(data['Color'], data['Salary'])
```

[52]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcf298d8550>



[53]:
```
scaler = MinMaxScaler()
data[['Age', 'Salary']] = scaler.fit_transform(data[['Age','Salary']])
```

[54]:
```
colors = [color for color in data['Color'].unique()]
colors
```

[54]: ['Red', 'Yellow', 'Green']

[55]:
```
transformedColors = pd.get_dummies(colors)
transformedColors
```

[55]:

	Green	Red	Yellow
0	0	1	0
1	0	0	1
2	1	0	0

[56]:
```
countries = [country for country in data['Country'].unique()]
countries
```

[56]: ['France', 'Spain', 'Germany']

[57]:
```
transformedCountries = pd.get_dummies(countries)
transformedCountries
```

[57]:

	France	Germany	Spain
0	1	0	0
1	0	0	1
2	0	1	0

3.3 Hamming Distance

[58]:
```
print('Hamming distance between Green and Red')
hamming(transformedColors.Green.values,transformedColors.Red.values)
```

```
Hamming distance between Green and Red
```

[58]: 0.6666666666666666

[59]:
```
print('Hamming distance between Green and Yellow')
hamming(transformedColors.Green.values,transformedColors.Yellow.values)
```

```
Hamming distance between Green and Yellow
```

[59]: 0.6666666666666666

[60]:
```
print('Hamming distance between Yellow and Red')
hamming(transformedColors.Yellow.values,transformedColors.Red.values)
```

```
Hamming distance between Yellow and Red
```

[60]:
```
0.6666666666666666
```

[61]:
```
print('Hamming distance between France and Germany')
hamming(transformedCountries.France.values,transformedCountries.Germany.values)
```

```
Hamming distance between France and Germany
```

[61]:
```
0.6666666666666666
```

[62]:
```
print('Hamming distance between France and Spain')
hamming(transformedCountries.France.values,transformedCountries.Spain.values)
```

```
Hamming distance between France and Spain
```

[62]:
```
0.6666666666666666
```

[63]:
```
print('Hamming distance between Spain and Germany')
hamming(transformedCountries.Spain.values,transformedCountries.Germany.values)
```

```
Hamming distance between Spain and Germany
```

[63]:
```
0.6666666666666666
```

3.4 Euclidean Distance

[64]:
```
purchased = data['Salary'][data['Purchased']=='Yes']
notPurchased = data['Salary'][data['Purchased']=='No']
```

[65]:
```
print("Euclidean distance between salaries of those who purchased and those who␣
↪didn't")
euclidean_distance(purchased.values[:len(notPurchased)], notPurchased.values)
```

```
Euclidean distance between salaries of those who purchased and those who didn't
```

[65]:
```
2.2761839138049376
```

[66]:
```
purchased = data['Age'][data['Purchased']=='Yes']
notPurchased = data['Age'][data['Purchased']=='No']
```

[67]:
```
print("Euclidean distance between ages of those who purchased and those who␣
↪didn't")
euclidean_distance(purchased.values[:len(notPurchased)], notPurchased.values)
```

```
Euclidean distance between ages of those who purchased and those who didn't
```

[67]:
```
2.007270862723103
```

[68]:
```python
print("Euclidean distance between age and salary")
euclidean_distance(data['Salary'].values, data['Age'].values)
```

```
Euclidean distance between age and salary
```

[68]:
```
6.421416331269408
```

3.5 Manhattan Distance

[69]:
```python
purchased = data['Salary'][data['Purchased']=='Yes']
notPurchased = data['Salary'][data['Purchased']=='No']
print("Manhattan distance between salaries of those who purchased and those who␣
↪didn't")
manhattan_distance(purchased.values[:len(notPurchased)], notPurchased.values)
```

```
Manhattan distance between salaries of those who purchased and those who didn't
```

[69]:
```
13.1566265060241
```

[70]:
```python
purchased = data['Age'][data['Purchased']=='Yes']
notPurchased = data['Age'][data['Purchased']=='No']
print("Manhattan distance between ages of those who purchased and those who␣
↪didn't")
manhattan_distance(purchased.values[:len(notPurchased)], notPurchased.values)
```

```
Manhattan distance between ages of those who purchased and those who didn't
```

[70]:
```
9.870967741935482
```

[71]:
```python
print("Manhattan distance between age and salary")
manhattan_distance(data['Salary'].values, data['Age'].values)
```

```
Manhattan distance between age and salary
```

[71]:
```
67.20734551107658
```

3.6 Minkowski Distance

[72]:
```python
purchased = data['Salary'][data['Purchased']=='Yes']
notPurchased = data['Salary'][data['Purchased']=='No']
print("Minkowski distance between salaries of those who purchased and those who␣
↪didn't with power 50")
```

```
minkowski_distance(purchased.values[:len(notPurchased)], notPurchased.values,␣
↪50)
```

```
Minkowski distance between salaries of those who purchased and those who didn't
with power 50
```

[72]: 0.6707834228068674

[73]:
```
purchased = data['Age'][data['Purchased']=='Yes']
notPurchased = data['Age'][data['Purchased']=='No']
print("Minkowski distance between ages of those who purchased and those who␣
↪didn't with power 50")
minkowski_distance(purchased.values[:len(notPurchased)], notPurchased.values,␣
↪50)
```

```
Minkowski distance between ages of those who purchased and those who didn't with
power 50
```

[73]: 0.9205480696375034

[74]:
```
print("Minkowski distance between age and salary with power 50")
minkowski_distance(data['Salary'].values, data['Age'].values, 50)
```

```
Minkowski distance between age and salary with power 50
```

[74]: 0.9022134220635781

3.7 Cosine Similarity

[75]:
```
purchased = data['Salary'][data['Purchased']=='Yes']
notPurchased = data['Salary'][data['Purchased']=='No']
print("Cosine Similarity between salaries of those who purchased and those who␣
↪didn't")
cosine_similarity(purchased.values[:len(notPurchased)], notPurchased.values)
```

```
Cosine Similarity between salaries of those who purchased and those who didn't
```

[75]: 0.9016

[76]:
```
purchased = data['Age'][data['Purchased']=='Yes']
notPurchased = data['Age'][data['Purchased']=='No']
print("Cosine Similarity between ages of those who purchased and those who␣
↪didn't")
cosine_similarity(purchased.values[:len(notPurchased)], notPurchased.values)
```

```
Cosine Similarity between ages of those who purchased and those who didn't
```

[76]: 0.4806

[77]:
```python
print("Cosine similarity between age and salary")
cosine_similarity(data['Salary'].values, data['Age'].values)
```

```
Cosine similarity between age and salary
```

[77]: 0.6874

3.8 Jaccard Similarity

[78]:
```python
purchased = data['Salary'][data['Purchased']=='Yes']
notPurchased = data['Salary'][data['Purchased']=='No']
print("Jaccard Similarity between salaries of those who purchased and those who␣
 ↪didn't")
jaccard_similarity(purchased.values[:len(notPurchased)], notPurchased.values)
```

```
Jaccard Similarity between salaries of those who purchased and those who didn't
```

[78]: 0.0

[79]:
```python
purchased = data['Age'][data['Purchased']=='Yes']
notPurchased = data['Age'][data['Purchased']=='No']
print("Jaccard Similarity between ages of those who purchased and those who␣
 ↪didn't")
jaccard_similarity(purchased.values[:len(notPurchased)], notPurchased.values)
```

```
Jaccard Similarity between ages of those who purchased and those who didn't
```

[79]: 0.36363636363636365

[80]:
```python
print("Jaccard similarity between age and salary")
jaccard_similarity(data['Salary'].values, data['Age'].values)
```

```
Jaccard similarity between age and salary
```

[80]: 0.06666666666666667

[ ]:
```python

```