Klassifikation Blutspenden

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv("../_projectdata/blood-transfusion-service-center.csv",sep=',',doublequote=True)
print("Dataframe mit Datensätzen",df.shape[0])
print("Dataframe mit den Attributen",list(df.columns))
```

```
Dataframe mit Datensätzen 748
Dataframe mit den Attributen ['V1', 'V2', 'V3', 'V4', 'Class']
```

In [3]:
```
df.info()
# df.head(5)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 748 entries, 0 to 747
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   V1      748 non-null    int64
 1   V2      748 non-null    int64
 2   V3      748 non-null    int64
 3   V4      748 non-null    int64
 4   Class   748 non-null    int64
dtypes: int64(5)
memory usage: 29.3 KB
```

Säubern

In [4]:
```
#Nullwerte
df.isna().sum()
```

Out[4]:
```
V1       0
V2       0
V3       0
V4       0
Class    0
dtype: int64
```

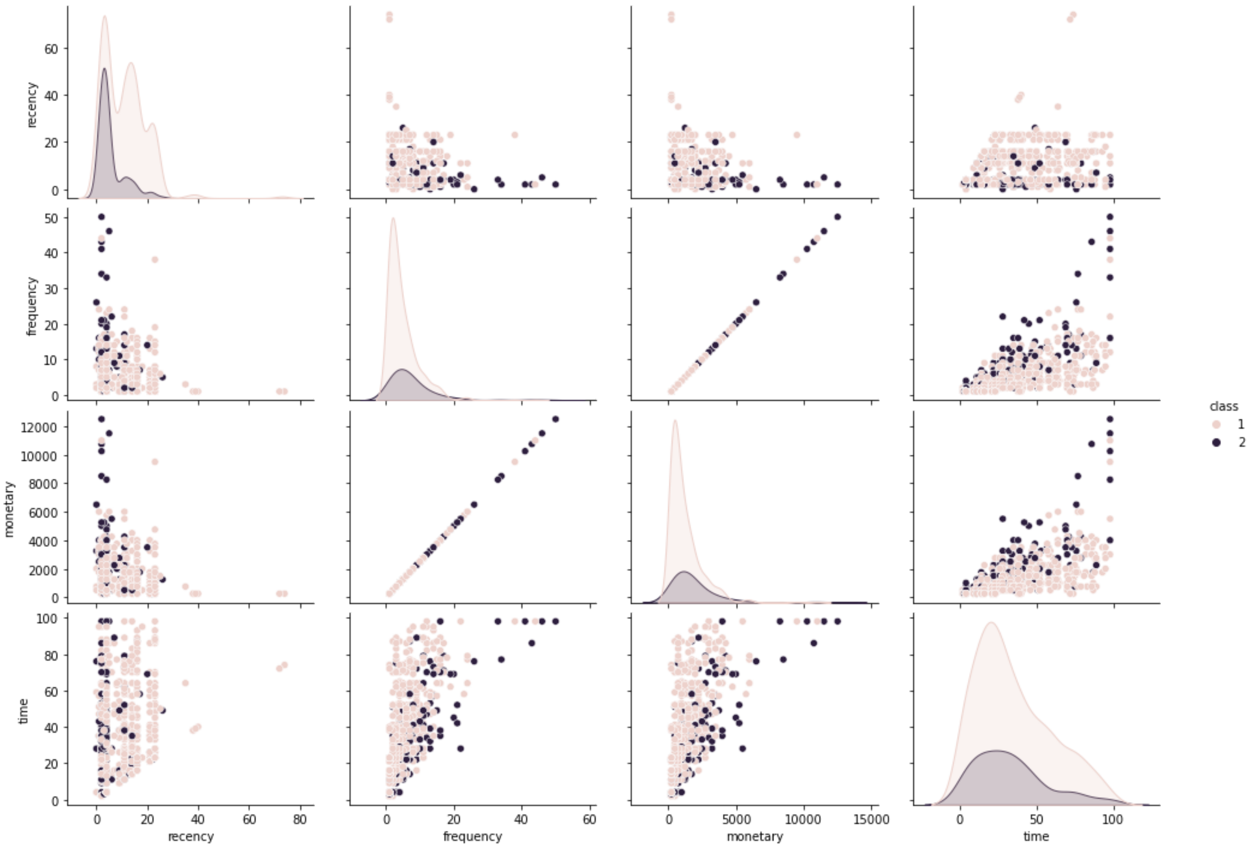
In [5]:
```
# Spalten benennen
df = df.rename(columns={"V1": "recency", "V2": "frequency", "V3": "monetary", "V4": "time", "Class": "class"})
df.columns
```

Out[5]:
```
Index(['recency', 'frequency', 'monetary', 'time', 'class'], dtype='object')
```

In [6]:
```
#Übersicht Klassen
df["class"]
```

Out[6]:
```
0      2
1      2
2      2
3      2
4      1
      ..
743    1
744    1
745    1
746    1
747    1
Name: class, Length: 748, dtype: int64
```

In [7]:
```
#Daten darstellen
import seaborn as sns
#kind{'scatter', 'kde', 'hist', 'reg'}
pairplot_figure = sns.pairplot(df, hue="class",kind="scatter")
pairplot_figure.fig.set_size_inches(15,10)
```



In [8]:
```
feature_columns = ['recency', 'frequency', 'monetary', 'time']
target_column = 'class'
df
```

Out[8]:

	recency	frequency	monetary	time	class
0	2	50	12500	98	2
1	0	13	3250	28	2
2	1	16	4000	35	2
3	2	20	5000	45	2
4	1	24	6000	77	1
...	...	...	...	...	...
743	23	2	500	38	1
744	21	2	500	52	1
745	23	3	750	62	1
746	39	1	250	39	1
747	72	1	250	72	1

748 rows × 5 columns

In [9]:
```
#Feature Datensätze und Target Datensätze trennen, dabei werden Attribute ausgeschlossen
features_df = df.iloc[:,0:4]  #

target_df = df.iloc[:,4]      #Die CLASS soll bestimmt werden
target_df
#features_df
```

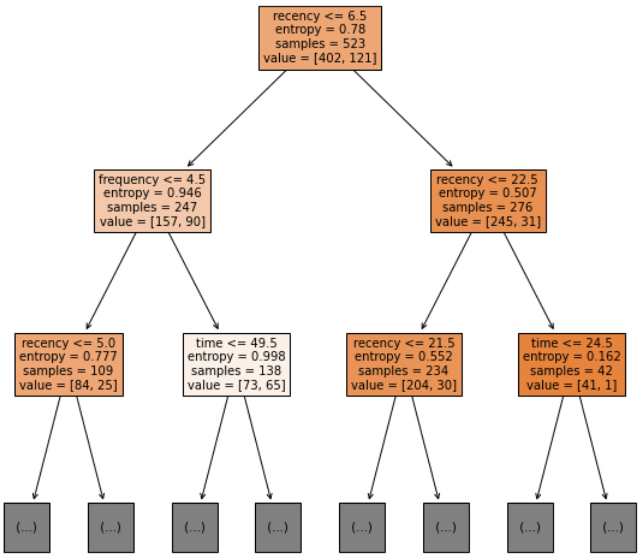
Out[9]:
```
0      2
1      2
2      2
3      2
4      1
      ..
743    1
744    1
745    1
746    1
747    1
Name: class, Length: 748, dtype: int64
```

In [10]:
```
#Daten zufällig in Trainingsdaten und Testdaten aufteilen
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(features_df, target_df, train_size = 0.7, random_state = 4)
#random_state = 1 für wiederholbaren Zufall

print("Trainingsdatensätze:",x_train.shape[0],"Zeilen")
print("Testdatensätze zur späteren Bewertung:",x_test.shape[0],"Zeilen")
```

```
Trainingsdatensätze: 523 Zeilen
Testdatensätze zur späteren Bewertung: 225 Zeilen
```

In [11]:
```
from sklearn import tree
clf = tree.DecisionTreeClassifier(criterion = "entropy") #Neuer Decisiontree
#criterion{“gini”, “entropy”}, default=”gini”

clf = clf.fit(x_train,y_train)        #Training
#print(clf.__doc__)
```

In [12]:
```
#Classifier anzeigen
plt.figure(figsize=(10,10))
tree.plot_tree(clf, fontsize=10, feature_names = x_test.columns,max_depth = 2,filled=True);
```



In [13]:
```
#Vorraussagen
y_pred = clf.predict(x_test)
```

In [14]:
```
#Vorraussage mit den Testdaten vergleichen
korrV = 0
anzV = 0
for (yt,yp) in list(zip(y_test,y_pred)):
    anzV += 1

    if yt == yp:
        print("        Testdaten:",yt,"Voraussage:",yp)
        korrV += 1
    else:
        print("FALSCH: Testdaten:",yt,"Voraussage:",yp,"FALSCH")


print("Von {0} Voraussagen treffen {1} zu. Das sind {2:3.0f} Prozent!".format(anzV, korrV,100*korrV/anzV))
```

```
        Testdaten: 2 Voraussage: 2
FALSCH: Testdaten: 1 Voraussage: 2 FALSCH
        Testdaten: 1 Voraussage: 1
        Testdaten: 2 Voraussage: 2
FALSCH: Testdaten: 2 Voraussage: 1 FALSCH
        Testdaten: 1 Voraussage: 1
FALSCH: Testdaten: 1 Voraussage: 2 FALSCH
        Testdaten: 1 Voraussage: 1
        Testdaten: 1 Voraussage: 1
        Testdaten: 2 Voraussage: 2
        Testdaten: 1 Voraussage: 1
        Testdaten: 2 Voraussage: 2
        Testdaten: 1 Voraussage: 1
FALSCH: Testdaten: 2 Voraussage: 1 FALSCH
        Testdaten: 1 Voraussage: 1
        Testdaten: 1 Voraussage: 1
        Testdaten: 1 Voraussage: 1
FALSCH: Testdaten: 2 Voraussage: 1 FALSCH
        Testdaten: 1 Voraussage: 1
        Testdaten: 1 Voraussage: 1
```

In [15]:
```
from sklearn import metrics
# Finding accuracy by comparing actual response values(y_test)with predicted response value(y_pred)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
# Providing sample data and the model will make prediction out of that data
```

```
Accuracy: 0.6977777777777778
```

Mehrere Durchläufe

In [16]:
```
from sklearn import metrics, tree
from sklearn.model_selection import train_test_split

trainpercent = 0.7
acc_list=[]
made = 4
n=100
for i in range(n):
    x_train, x_test, y_train, y_test = train_test_split(features_df, target_df, train_size = trainpercent, random_state = i)
    clf = tree.DecisionTreeClassifier(criterion = "entropy",max_depth = made ) #Neuer Decisiontree
    clf = clf.fit(x_train,y_train)        #Training
    y_pred = clf.predict(x_test)
    acc_list.append(metrics.accuracy_score(y_test, y_pred))

print("Accuracy (n=",n,"), Traindata: ",trainpercent*100,"%")
print("min:",min(acc_list))
print("max:",max(acc_list))
print("avg:",sum(acc_list)/n)
```

```
Accuracy (n= 100 ), Traindata:  70.0 %
min: 0.6933333333333334
max: 0.8311111111111111
avg: 0.7718666666666664
```

In [ ]: