In [1]:

```
import numpy as np
import pandas as pd
```

In [10]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import recall_score
from sklearn.metrics import auc
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import RocCurveDisplay
from sklearn.metrics import \
PrecisionRecallDisplay
import seaborn as sns
import matplotlib.pyplot as plt
```

In [11]:

```
divorce= pd.read_csv('divorce.csv')
```

In [12]:

```
divorce
```

Out[12]:

	Sorry_end	Ignore_diff	begin_correct	Contact	Special_time	No_home_time	2_strangers	e
0	2	2	4	1	0	0	0	
1	4	4	4	4	4	0	0	
2	2	2	2	2	1	3	2	
3	3	2	3	2	3	3	3	
4	2	2	1	1	1	1	0	
...	...	...	...	...	...	...	...	
165	0	0	0	0	0	0	0	
166	0	0	0	0	0	0	0	
167	1	1	0	0	0	0	0	
168	0	0	0	0	0	0	0	
169	0	0	0	0	0	0	0	

170 rows × 55 columns

In [13]:

```
divorce.head()
```

Out[13]:

	Sorry_end	Ignore_diff	begin_correct	Contact	Special_time	No_home_time	2_strangers	enj
0	2	2	4	1	0	0	0	
1	4	4	4	4	4	0	0	
2	2	2	2	2	1	3	2	
3	3	2	3	2	3	3	3	
4	2	2	1	1	1	1	0	

5 rows × 55 columns

In [14]:

```
divorce.tail()
```

Out[14]:

	Sorry_end	Ignore_diff	begin_correct	Contact	Special_time	No_home_time	2_strangers	e
165	0	0	0	0	0	0	0	
166	0	0	0	0	0	0	0	
167	1	1	0	0	0	0	0	
168	0	0	0	0	0	0	0	
169	0	0	0	0	0	0	0	

5 rows × 55 columns

In [15]:

```
divorce.describe()
```

Out[15]:

	Sorry_end	Ignore_diff	begin_correct	Contact	Special_time	No_home_time	2_strar
count	170.000000	170.000000	170.000000	170.000000	170.000000	170.000000	170.00
mean	1.776471	1.652941	1.764706	1.482353	1.541176	0.747059	0.49
std	1.627257	1.468654	1.415444	1.504327	1.632169	0.904046	0.89
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	2.000000	2.000000	2.000000	1.000000	1.000000	0.000000	0.00
75%	3.000000	3.000000	3.000000	3.000000	3.000000	1.000000	1.00
max	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.00

8 rows × 55 columns

In [16]:

```
divorce.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170 entries, 0 to 169
Data columns (total 55 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Sorry_end                       170 non-null    int64
 1   Ignore_diff                     170 non-null    int64
 2   begin_correct                   170 non-null    int64
 3   Contact                         170 non-null    int64
 4   Special_time                    170 non-null    int64
 5   No_home_time                    170 non-null    int64
 6   2_strangers                     170 non-null    int64
 7   enjoy_holiday                   170 non-null    int64
 8   enjoy_travel                    170 non-null    int64
 9   common_goals                    170 non-null    int64
 10  harmony                         170 non-null    int64
 11  freeom_value                    170 non-null    int64
 12  entertain                       170 non-null    int64
 13  people_goals                    170 non-null    int64
 14  dreams                          170 non-null    int64
 15  love                            170 non-null    int64
 16  happy                           170 non-null    int64
 17  marriage                        170 non-null    int64
 18  roles                           170 non-null    int64
 19  trust                           170 non-null    int64
 20  likes                           170 non-null    int64
 21  care_sick                       170 non-null    int64
 22  fav_food                        170 non-null    int64
 23  stresses                        170 non-null    int64
 24  inner_world                     170 non-null    int64
 25  anxieties                       170 non-null    int64
 26  current_stress                  170 non-null    int64
 27  hopes_wishes                    170 non-null    int64
 28  know_well                       170 non-null    int64
 29  friends_social                  170 non-null    int64
 30  Aggro_argue                     170 non-null    int64
 31  Always_never                    170 non-null    int64
 32  negative_personality            170 non-null    int64
 33  offensive_expressions           170 non-null    int64
 34  insult                          170 non-null    int64
 35  humiliate                       170 non-null    int64
 36  not_calm                        170 non-null    int64
 37  hate_subjects                   170 non-null    int64
 38  sudden_discussion               170 non-null    int64
 39  idk_what's_going_on             170 non-null    int64
 40  calm_breaks                     170 non-null    int64
 41  argue_then_leave                170 non-null    int64
 42  silent_for_calm                 170 non-null    int64
 43  good_to_leave_home              170 non-null    int64
 44  silence_instead_of_discussion   170 non-null    int64
 45  silence_for_harm                170 non-null    int64
 46  silence_fear_anger              170 non-null    int64
 47  I'm_right                       170 non-null    int64
 48  accusations                     170 non-null    int64
 49  I'm_not_guilty                  170 non-null    int64
 50  I'm_not_wrong                   170 non-null    int64
```

```
 51  no_hesitancy_inadequate        170 non-null    int64
 52  you're_inadequate              170 non-null    int64
 53  incompetence                   170 non-null    int64
 54  Divorce_Y_N                    170 non-null    int64
dtypes: int64(55)
memory usage: 73.2 KB
```

In [17]:

```
divorce.isna().sum()
```

Out[17]:

```
Sorry_end                          0
Ignore_diff                        0
begin_correct                      0
Contact                            0
Special_time                       0
No_home_time                       0
2_strangers                        0
enjoy_holiday                      0
enjoy_travel                       0
common_goals                       0
harmony                            0
freeom_value                       0
entertain                          0
people_goals                       0
dreams                             0
love                               0
happy                              0
marriage                           0
roles                              0
trust                              0
likes                              0
care_sick                          0
fav_food                           0
stresses                           0
inner_world                        0
anxieties                          0
current_stress                     0
hopes_wishes                       0
know_well                          0
friends_social                     0
Aggro_argue                        0
Always_never                       0
negative_personality               0
offensive_expressions              0
insult                             0
humiliate                          0
not_calm                           0
hate_subjects                      0
sudden_discussion                  0
idk_what's_going_on                0
calm_breaks                        0
argue_then_leave                   0
silent_for_calm                    0
good_to_leave_home                 0
silence_instead_of_discussion      0
silence_for_harm                   0
silence_fear_anger                 0
I'm_right                          0
accusations                        0
I'm_not_guilty                     0
I'm_not_wrong                      0
no_hesitancy_inadequate            0
you're_inadequate                  0
incompetence                       0
```

```
Divorce_Y_N                    0
dtype: int64
```

In [19]:

```
X = divorce.drop(['Divorce_Y_N'], axis=1)
```

In [20]:

```
X
```

Out[20]:

	Sorry_end	Ignore_diff	begin_correct	Contact	Special_time	No_home_time	2_strangers	e
0	2	2	4	1	0	0	0	
1	4	4	4	4	4	0	0	
2	2	2	2	2	1	3	2	
3	3	2	3	2	3	3	3	
4	2	2	1	1	1	1	0	
...	...	...	...	...	...	...	...	
165	0	0	0	0	0	0	0	
166	0	0	0	0	0	0	0	
167	1	1	0	0	0	0	0	
168	0	0	0	0	0	0	0	
169	0	0	0	0	0	0	0	

170 rows × 54 columns

In [21]:

```
y = divorce['Divorce_Y_N']
```

In [22]:

```
y
```

Out[22]:

```
0      1
1      1
2      1
3      1
4      1
      ..
165    0
166    0
167    0
168    0
169    0
Name: Divorce_Y_N, Length: 170, dtype: int64
```

In [23]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,  random_state=

clf_rf = RandomForestClassifier()

parametrs = {'n_estimators': range(10, 50, 10),
             'max_depth': range(1, 12, 2),
             'min_samples_leaf': range(1, 7),
             'min_samples_split': range(2, 9, 2)}

grid_search_cv_clf = GridSearchCV(clf_rf, parametrs, cv=5, n_jobs=-1, verbose=1)
grid_search_cv_clf.fit(X_train, y_train)
grid_search_cv_clf.best_params_
```

```
Fitting 5 folds for each of 576 candidates, totalling 2880 fits
```

Out[23]:

```
{'max_depth': 1,
 'min_samples_leaf': 1,
 'min_samples_split': 4,
 'n_estimators': 10}
```

In [24]:

```
best_clf = grid_search_cv_clf.best_estimator_
best_clf.score(X_test, y_test)
```

Out[24]:

```
0.9824561403508771
```

In [25]:

```
feature_importance = best_clf.feature_importances_
feature_importance_df = pd.DataFrame({'features': list(X_train),
                                      'feature_importance': feature_importance})
feature_importance_df = feature_importance_df.sort_values('feature_importance', ascendir
feature_importance_df
```

Out[25]:

	features	feature_importance
37	hate_subjects	0.3
25	anxieties	0.2
14	dreams	0.1
15	love	0.1
39	idk_what's_going_on	0.1
17	marriage	0.1
29	friends_social	0.1
31	Always_never	0.0
32	negative_personality	0.0
33	offensive_expressions	0.0
34	insult	0.0
35	humiliate	0.0
36	not_calm	0.0
38	sudden_discussion	0.0
40	calm_breaks	0.0
0	Sorry_end	0.0
41	argue_then_leave	0.0
42	silent_for_calm	0.0
43	good_to_leave_home	0.0
44	silence_instead_of_discussion	0.0
45	silence_for_harm	0.0
46	silence_fear_anger	0.0
47	I'm_right	0.0
48	accusations	0.0
49	I'm_not_guilty	0.0
50	I'm_not_wrong	0.0
51	no_hesitancy_inadequate	0.0
52	you're_inadequate	0.0
30	Aggro_argue	0.0
27	hopes_wishes	0.0

	features	feature_importance
28	know_well	0.0
1	Ignore_diff	0.0
2	begin_correct	0.0
3	Contact	0.0
4	Special_time	0.0
5	No_home_time	0.0
6	2_strangers	0.0
7	enjoy_holiday	0.0
8	enjoy_travel	0.0
9	common_goals	0.0
10	harmony	0.0
11	freeom_value	0.0
12	entertain	0.0
13	people_goals	0.0
16	happy	0.0
18	roles	0.0
19	trust	0.0
20	likes	0.0
21	care_sick	0.0
22	fav_food	0.0
23	stresses	0.0
24	inner_world	0.0
26	current_stress	0.0
53	incompetence	0.0

In [26]:

```
X_short = X[['hate_subjects', 'roles', 'dreams', 'humiliate', 'likes',
             'happy', 'marriage', 'love','freeom_value', 'anxieties',
             'enjoy_travel', 'Ignore_diff', 'Special_time', 'care_sick']]
```

In [27]:

```
X_short
```

Out[27]:

	hate_subjects	roles	dreams	humiliate	likes	happy	marriage	love	freeom_value	anxiet
0	1	0	0	1	0	0	0	1	0	
1	4	3	4	2	1	4	4	4	3	
2	1	3	3	1	1	3	3	3	4	
3	3	3	3	1	1	3	3	3	3	
4	0	2	1	0	1	1	1	1	1	
...	...	...	...	...	...	...	...	...	...	
165	0	0	0	0	0	0	0	0	0	
166	1	0	0	1	0	0	0	0	0	
167	1	0	0	0	0	0	0	1	1	
168	0	0	0	0	0	0	0	0	1	
169	0	0	0	0	1	0	0	0	1	

170 rows × 14 columns

In [28]:

```
X_train_sh, X_test_sh, y_train_sh, y_test_sh = train_test_split(X_short, y, test_size=0.

clf_rf = RandomForestClassifier()

parametrs = {'n_estimators': range(10, 50, 10),
             'max_depth': range(1, 12, 2),
             'min_samples_leaf': range(1, 7),
             'min_samples_split': range(2, 9, 2)}

grid_search_cv_clf = GridSearchCV(clf_rf, parametrs, cv=5, n_jobs=-1, verbose=1)
grid_search_cv_clf.fit(X_train_sh, y_train_sh)
grid_search_cv_clf.best_params_
```

```
Fitting 5 folds for each of 576 candidates, totalling 2880 fits
```

Out[28]:

```
{'max_depth': 1,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 10}
```

In [29]:

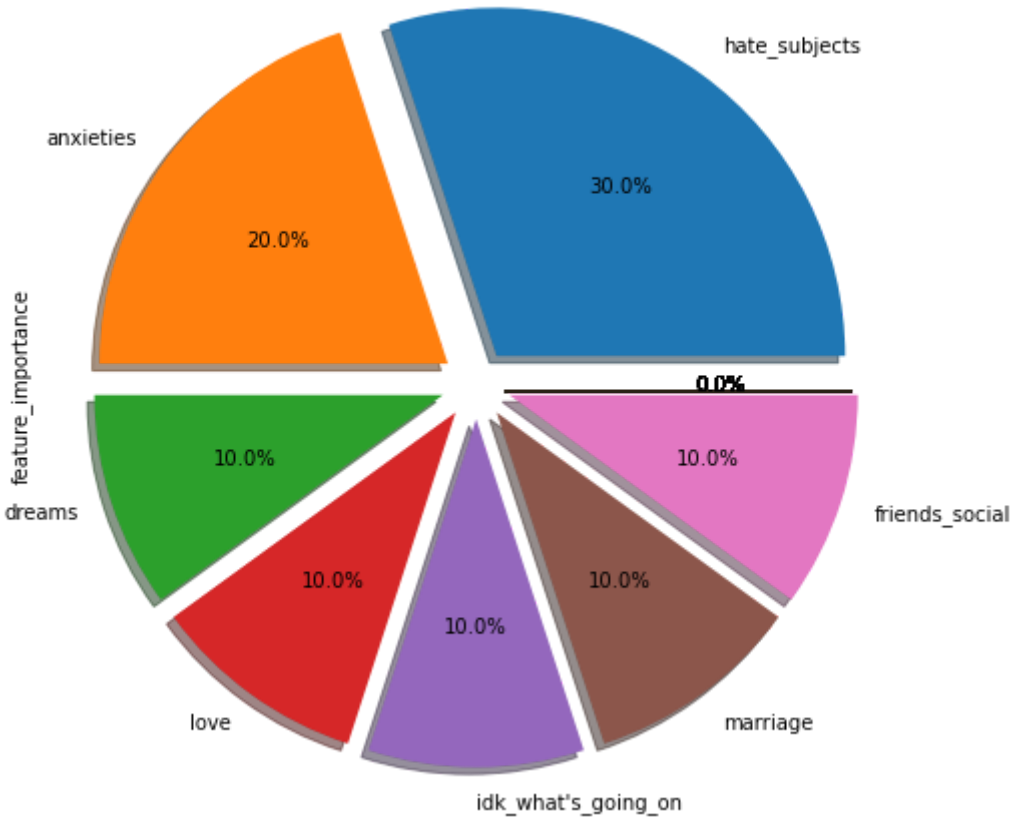
```
best_clf = grid_search_cv_clf.best_estimator_

best_clf.score(X_test_sh, y_test_sh)
```

Out[29]:

```
0.9824561403508771
```

In [30]:

```
feature_importance_df.plot.pie(
                        explode=[0.1]*len(X_train.columns),
                        labels = feature_importance_df.features,
                        y = 'feature_importance',
                        autopct='%1.1f%%',
                        shadow=True,
                        legend=False,
                        figsize=(8, 8));
```

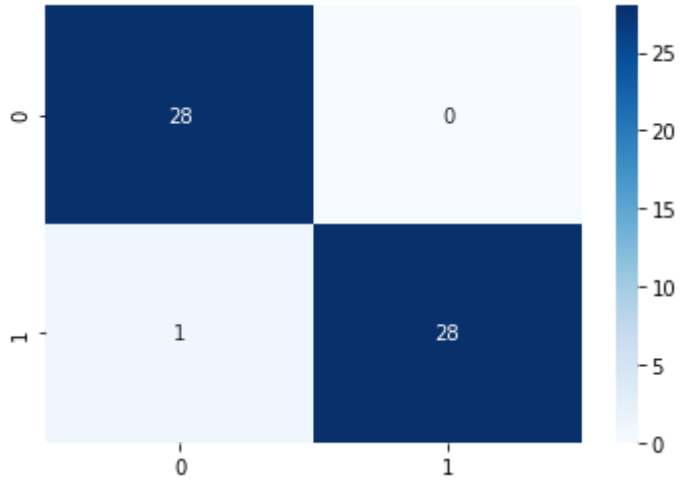


In [31]:

```
y_pred = best_clf.predict(X_test_sh)
sns.heatmap(confusion_matrix(y_test_sh, y_pred), annot=True, cmap="Blues")
```

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xd77c48c070>
```



In [32]:

```
precision = precision_score(y_test_sh, y_pred)
recall = recall_score(y_test_sh, y_pred)
```

In [33]:

```
probability = best_clf.predict_proba(X_test_sh)
probability
```

Out[33]:

```
array([[0.87078262, 0.12921738],
       [0.00850575, 0.99149425],
       [0.67575617, 0.32424383],
       [0.00850575, 0.99149425],
       [0.67435405, 0.32564595],
       [0.68030643, 0.31969357],
       [0.96893077, 0.03106923],
       [0.00850575, 0.99149425],
       [0.96893077, 0.03106923],
       [0.00850575, 0.99149425],
       [0.00850575, 0.99149425],
       [0.67575617, 0.32424383],
       [0.96893077, 0.03106923],
       [0.00850575, 0.99149425],
       [0.00850575, 0.99149425],
       [0.96893077, 0.03106923],
       [0.00850575, 0.99149425],
       [0.96893077, 0.03106923],
       [0.1066539 , 0.8933461 ],
       [0.67575617, 0.32424383],
       [0.00850575, 0.99149425],
       [0.00850575, 0.99149425],
       [0.00850575, 0.99149425],
       [0.00850575, 0.99149425],
       [0.00850575, 0.99149425],
       [0.78673326, 0.21326674],
       [0.00850575, 0.99149425],
       [0.96893077, 0.03106923],
       [0.1031486 , 0.8968514 ],
       [0.00850575, 0.99149425],
       [0.00850575, 0.99149425],
       [0.00850575, 0.99149425],
       [0.96893077, 0.03106923],
       [0.78673326, 0.21326674],
       [0.1066539 , 0.8933461 ],
       [0.87792448, 0.12207552],
       [0.1066539 , 0.8933461 ],
       [0.87078262, 0.12921738],
       [0.96893077, 0.03106923],
       [0.67575617, 0.32424383],
       [0.96893077, 0.03106923],
       [0.96893077, 0.03106923],
       [0.00850575, 0.99149425],
       [0.96893077, 0.03106923],
       [0.87792448, 0.12207552],
       [0.96893077, 0.03106923],
       [0.00850575, 0.99149425],
       [0.00850575, 0.99149425],
       [0.96893077, 0.03106923],
       [0.00850575, 0.99149425],
       [0.96893077, 0.03106923],
       [0.6920904 , 0.3079096 ],
       [0.96893077, 0.03106923],
```

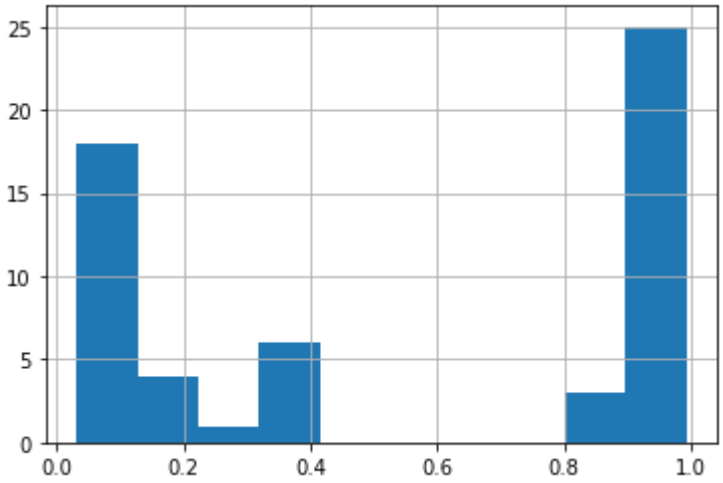
```
       [0.00850575, 0.99149425],
       [0.00850575, 0.99149425],
       [0.96893077, 0.03106923],
       [0.00850575, 0.99149425]])
```

In [34]:

```
true_probability = probability[:, 1]
pd.Series(true_probability).hist()
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xd77c35ea00>
```



In [35]:

```
fpr, tpr, thresholds = roc_curve(y_test_sh, true_probability)
roc_auc= auc(fpr, tpr)
lw=2
plt.figure()
plt.plot(fpr, tpr, color='darkorange',
         lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```