Pandas, seaborn, обучение модели, метрики качества

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

In [2]:
```
# максимальное кол-во отображаемых столбцов
pd.set_option('display.max_columns', 13)
# максимальное кол-во отображаемых строк
pd.set_option('display.max_rows', 10)
# максимальная ширина столбца
pd.set_option('display.max_colwidth', 45)
# максимальная ширина отображения
pd.set_option('display.width', 80)
```

In [3]:
```
data = pd.read_csv('titanic.csv', index_col='passenger_id')
```

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 791 entries, 100 to 890
Data columns (total 11 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   survived  791 non-null    int64
 1   pclass    791 non-null    int64
 2   name      791 non-null    object
 3   gender    791 non-null    object
 4   age       636 non-null    float64
 5   sibsp     791 non-null    int64
 6   parch     791 non-null    int64
 7   ticket    791 non-null    object
 8   fare      791 non-null    float64
 9   cabin     184 non-null    object
 10  embarked  790 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 74.2+ KB
```

In [5]:
```
# pclass — класс пассажира (1 — высший, 2 — средний, 3 — низший);
# name — имя;
# gender — пол;
# age — возраст;
# sibsp — количество братьев, сестер, сводных братьев, сводных сестер, супругов на борту
# parch — количество родителей, детей (в том числе приемных) на борту титаника;
# ticket — номер билета;
# fare — плата за проезд;
# cabin — каюта;
# embarked — порт посадки (C — Шербур; Q — Квинстаун; S — Саутгемптон).
```

In [6]:
```
data.head()
```

Out[6]:

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	cab
100	0	3	Petranec, Miss. Matilda	female	28.0	0	0	349245	7.8958	N

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	cab
101	0	3	Petroff, Mr. Pastcho ("Pentcho")	male	NaN	0	0	349215	7.8958	N
102	0	1	White, Mr. Richard Frasar	male	21.0	0	1	35281	77.2875	D
103	0	3	Johansson, Mr. Gustaf Joel	male	33.0	0	0	7540	8.6542	N
104	0	3	Gustafsson, Mr. Anders Vilhelm	male	37.0	2	0	3101276	7.9250	N

In [7]:
```
data.head(10)
```

Out[7]:

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	cab
100	0	3	Petranec, Miss. Matilda	female	28.0	0	0	349245	7.8958	N
101	0	3	Petroff, Mr. Pastcho ("Pentcho")	male	NaN	0	0	349215	7.8958	N
102	0	1	White, Mr. Richard Frasar	male	21.0	0	1	35281	77.2875	D
103	0	3	Johansson, Mr. Gustaf Joel	male	33.0	0	0	7540	8.6542	N
104	0	3	Gustafsson, Mr. Anders Vilhelm	male	37.0	2	0	3101276	7.9250	N
105	0	3	Mionoff, Mr. Stoytcho	male	28.0	0	0	349207	7.8958	N
106	1	3	Salkjelsvik, Miss. Anna Kristine	female	21.0	0	0	343120	7.6500	N
107	1	3	Moss, Mr. Albert Johan	male	NaN	0	0	312991	7.7750	N
108	0	3	Rekic, Mr. Tido	male	38.0	0	0	349249	7.8958	N
109	1	3	Moran, Miss. Bertha	female	NaN	1	0	371110	24.1500	N

In [8]:
```
data.tail()
```

Out[8]:

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	cabin

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	cabin
886	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN
887	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42
888	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN
889	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148
890	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN

In [9]:
```
data.sample(n=5)
```

Out[9]:

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	cab
126	0	3	McMahon, Mr. Martin	male	NaN	0	0	370372	7.7500	Na
441	0	3	Hampe, Mr. Leon	male	20.0	0	0	345769	9.5000	Na
696	0	3	Kelly, Mr. James	male	44.0	0	0	363592	8.0500	Na
779	1	1	Robert, Mrs. Edward Scott (Elisabeth Walt...	female	43.0	0	1	24160	211.3375	
536	0	1	Butt, Major. Archibald Willingham	male	45.0	0	0	113050	26.5500	B3

In [10]:
```
data.shape
```

Out[10]:
```
(791, 11)
```

In [11]:
```
data.columns
```

Out[11]:
```
Index(['survived', 'pclass', 'name', 'gender', 'age', 'sibsp', 'parch',
       'ticket', 'fare', 'cabin', 'embarked'],
      dtype='object')
```

Выборки из таблицы

In [12]:
```
data.loc[:, 'name']
```

Out[12]:
```
passenger_id
100                       Petranec, Miss. Matilda
101              Petroff, Mr. Pastcho ("Pentcho")
102                     White, Mr. Richard Frasar
103                   Johansson, Mr. Gustaf Joel
104                Gustafsson, Mr. Anders Vilhelm
                             ...
886                      Montvila, Rev. Juozas
887                  Graham, Miss. Margaret Edith
888    Johnston, Miss. Catherine Helen "Carrie"
889                        Behr, Mr. Karl Howell
890                          Dooley, Mr. Patrick
Name: name, Length: 791, dtype: object
```

In [13]:
```
data.loc[:, ['name', 'gender']]
```

Out[13]:

passenger_id	name	gender
100	Petranec, Miss. Matilda	female
101	Petroff, Mr. Pastcho ("Pentcho")	male
102	White, Mr. Richard Frasar	male
103	Johansson, Mr. Gustaf Joel	male
104	Gustafsson, Mr. Anders Vilhelm	male
...	...	...
886	Montvila, Rev. Juozas	male
887	Graham, Miss. Margaret Edith	female
888	Johnston, Miss. Catherine Helen "Carrie"	female
889	Behr, Mr. Karl Howell	male
890	Dooley, Mr. Patrick	male

791 rows × 2 columns

In [14]:
```
data.loc[708, 'name']
```

Out[14]:
```
'Cleaver, Miss. Alice'
```

In [16]:
```
data.loc[100]
```

Out[16]:
```
survived                          0
pclass                            3
name        Petranec, Miss. Matilda
gender                       female
age                              28
                    ...
parch                             0
ticket                       349245
fare                         7.8958
cabin                           NaN
embarked                          S
Name: 100, Length: 11, dtype: object
```

In [17]:
```
data.loc[102:105]
```

Out[17]:

	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	cal

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	cab
passenger_id										
102	0	1	White, Mr. Richard Frasar	male	21.0	0	1	35281	77.2875	D
103	0	3	Johansson, Mr. Gustaf Joel	male	33.0	0	0	7540	8.6542	N
104	0	3	Gustafsson, Mr. Anders Vilhelm	male	37.0	2	0	3101276	7.9250	N
105	0	3	Mionoff, Mr. Stoytcho	male	28.0	0	0	349207	7.8958	N

In [18]:
```
data[data['gender'] == 'male']
#короткая запись того же loc
```

Out[18]:

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare
101	0	3	Petroff, Mr. Pastcho ("Pentcho")	male	NaN	0	0	349215	7.8958
102	0	1	White, Mr. Richard Frasar	male	21.0	0	1	35281	77.2875
103	0	3	Johansson, Mr. Gustaf Joel	male	33.0	0	0	7540	8.6542
104	0	3	Gustafsson, Mr. Anders Vilhelm	male	37.0	2	0	3101276	7.9250
105	0	3	Mionoff, Mr. Stoytcho	male	28.0	0	0	349207	7.8958
...	...	...	...	...	...	...	...	...	...
883	0	2	Banfield, Mr. Frederick James	male	28.0	0	0	C.A./SOTON 34068	10.5000
884	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500
886	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000
889	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000
890	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500

516 rows × 11 columns

In [19]:
```
data[(data['gender'] == 'male') & (data['pclass'] == 3)]
```

Out[19]:

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	c
101	0	3	Petroff, Mr. Pastcho ("Pentcho")	male	NaN	0	0	349215	7.8958	
103	0	3	Johansson, Mr. Gustaf Joel	male	33.0	0	0	7540	8.6542	
104	0	3	Gustafsson, Mr. Anders Vilhelm	male	37.0	2	0	3101276	7.9250	
105	0	3	Mionoff, Mr. Stoytcho	male	28.0	0	0	349207	7.8958	
107	1	3	Moss, Mr. Albert Johan	male	NaN	0	0	312991	7.7750	
...	...	...	...	...	...	...	...	...	...	...
877	0	3	Petroff, Mr. Nedelio	male	19.0	0	0	349212	7.8958	
878	0	3	Laleff, Mr. Kristo	male	NaN	0	0	349217	7.8958	
881	0	3	Markun, Mr. Johann	male	33.0	0	0	349257	7.8958	
884	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	
890	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	

308 rows × 11 columns

In [20]:
```
data[data['gender'] == 'male']['age']
```

Out[20]:
```
passenger_id
101     NaN
102    21.0
103    33.0
104    37.0
105    28.0
       ...
883    28.0
884    25.0
886    27.0
889    26.0
890    32.0
Name: age, Length: 516, dtype: float64
```

Функции таблицы

In [21]:
```
data['survived'].mean()
```

Out[21]:
```
0.3805309734513274
```

In [22]:
```
data['survived'].value_counts()
```

```
0    490
```

Out[22]:
```
1    301
Name: survived, dtype: int64
```

In [23]:
```
(data['gender'] == 'female').mean()
```

Out[23]:
```
0.347661188369153
```

In [24]:
```
data[data['gender'] == 'female']['survived'].mean()
```

Out[24]:
```
0.7345454545454545
```

In [25]:
```
data[data['gender'] == 'male']['survived'].mean()
```

Out[25]:
```
0.19186046511627908
```

In [26]:
```
data[data['gender'] == 'female']['survived'].sum()
```

Out[26]:
```
202
```

In [27]:
```
data[data['gender'] == 'female']['survived'].count()
#число заполненных строк!
```

Out[27]:
```
275
```

In [28]:
```
data[data['gender'] == 'female']['survived'].shape
```

Out[28]:
```
(275,)
```

In [29]:
```
data['age'].min()
```

Out[29]:
```
0.42
```

In [30]:
```
data['age'].max()
```

Out[30]:
```
80.0
```

In [31]:
```
data['age'].median()
```

Out[31]:
```
28.75
```

In [32]:
```
data['age'].mean()
```

Out[32]:
```
29.97301886792453
```

Пропуски данных

In [33]:
```
data.isnull().sum()
```

Out[33]:
```
survived      0
pclass        0
name          0
gender        0
age         155
           ...
parch         0
ticket        0
fare          0
cabin       607
embarked      1
Length: 11, dtype: int64
```

In [34]:
```
data.isnull().mean().sort_values(ascending=False)
```

Out[34]:
```
cabin        0.767383
age          0.195954
embarked     0.001264
fare         0.000000
ticket       0.000000
               ...
sibsp        0.000000
gender       0.000000
name         0.000000
pclass       0.000000
survived     0.000000
Length: 11, dtype: float64
```

In [35]:
```
data.dropna()
#у этой функции есть inplace
# удаление всех строк, где есть хотя бы одно пропущенное значение
```

Out[35]:

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	ca
102	0	1	White, Mr. Richard Frasar	male	21.0	0	1	35281	77.2875	
110	0	1	Porter, Mr. Walter Chamberlain	male	47.0	0	0	110465	52.0000	C
118	0	1	Baxter, Mr. Quigg Edmond	male	24.0	0	1	PC 17558	247.5208	
123	1	2	Webber, Miss. Susan	female	32.5	0	0	27267	13.0000	E
124	0	1	White, Mr. Percival Wayland	male	54.0	0	1	35281	77.2875	
...	...	...	...	...	...	...	...	...	...	
871	1	1	Beckwith, Mrs. Richard Leonard (Sallie Mo...	female	47.0	1	1	11751	52.5542	
872	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000	
879	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wil...	female	56.0	0	1	11767	83.1583	
887	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	
889	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C

166 rows × 11 columns

In [36]:
```
data.dropna(subset=['cabin', 'age', 'embarked'])
#пропуски будут искаться только в этих столбцах
```

Out[36]:

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	ca
102	0	1	White, Mr. Richard Frasar	male	21.0	0	1	35281	77.2875	
110	0	1	Porter, Mr. Walter Chamberlain	male	47.0	0	0	110465	52.0000	C
118	0	1	Baxter, Mr. Quigg Edmond	male	24.0	0	1	PC 17558	247.5208	
123	1	2	Webber, Miss. Susan	female	32.5	0	0	27267	13.0000	E
124	0	1	White, Mr. Percival Wayland	male	54.0	0	1	35281	77.2875	
...	...	...	...	...	...	...	...	...	...	
871	1	1	Beckwith, Mrs. Richard Leonard (Sallie Mo...	female	47.0	1	1	11751	52.5542	
872	0	1	Carlsson, Mr. Frans Olof	male	33.0	0	0	695	5.0000	
879	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wil...	female	56.0	0	1	11767	83.1583	
887	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	
889	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C

166 rows × 11 columns

In [37]:
```
data['age'].fillna(value=-999)
```

Out[37]:
```
passenger_id
100     28.0
101   -999.0
102     21.0
103     33.0
104     37.0
       ...
886     27.0
887     19.0
888   -999.0
889     26.0
890     32.0
Name: age, Length: 791, dtype: float64
```

In [38]:
```
data['age'].fillna(value=0)
```

Out[38]:
```
passenger_id
100    28.0
101     0.0
102    21.0
103    33.0
104    37.0
       ...
886    27.0
887    19.0
888     0.0
889    26.0
890    32.0
Name: age, Length: 791, dtype: float64
```

In [39]:
```
data['age'].fillna(value=data['age'].mean())
```

Out[39]:
```
passenger_id
100    28.000000
101    29.973019
102    21.000000
103    33.000000
104    37.000000
         ...
886    27.000000
887    19.000000
888    29.973019
889    26.000000
890    32.000000
Name: age, Length: 791, dtype: float64
```

In [42]:
```
data.isnull().mean().sort_values(ascending=False)
```

Out[42]:
```
embarked    0.0
fare        0.0
ticket      0.0
parch       0.0
sibsp       0.0
age         0.0
gender      0.0
name        0.0
pclass      0.0
survived    0.0
dtype: float64
```

In [43]:
```
data.head()
```

Out[43]:

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	e
100	0	3	Petranec, Miss. Matilda	female	28.00	0	0	349245	7.8958	
101	0	3	Petroff, Mr. Pastcho ("Pentcho")	male	28.75	0	0	349215	7.8958	
102	0	1	White, Mr. Richard Frasar	male	21.00	0	1	35281	77.2875	
103	0	3	Johansson, Mr. Gustaf Joel	male	33.00	0	0	7540	8.6542	

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	ei
104	0	3	Gustafsson, Mr. Anders Vilhelm	male	37.00	2	0	3101276	7.9250	

Дубликаты данных

In [44]:
```
data.duplicated().sum()
```

Out[44]: 0

In [45]:
```
data.drop_duplicates(inplace=True)
```

Apply

In [46]:
```
def age_group(age):
    if age < 18:
        return 0
    if age < 35:
        return 1
    else:
        return 2
```

In [47]:
```
data['age_group'] = data['age'].apply(age_group)
#можно передавать сразу две колонки, например data[['col1', 'col2']].apply(func, ax
```

In [48]:
```
data.head()
```

Out[48]:

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	ei
100	0	3	Petranec, Miss. Matilda	female	28.00	0	0	349245	7.8958	
101	0	3	Petroff, Mr. Pastcho ("Pentcho")	male	28.75	0	0	349215	7.8958	
102	0	1	White, Mr. Richard Frasar	male	21.00	0	1	35281	77.2875	
103	0	3	Johansson, Mr. Gustaf Joel	male	33.00	0	0	7540	8.6542	
104	0	3	Gustafsson, Mr. Anders Vilhelm	male	37.00	2	0	3101276	7.9250	

In [49]:
```
data.drop(['age_group'], 1, inplace = True)
```

In [50]:
```
data['embarked'] = data['embarked'].map({'C': 'Cherbourg', 'Q': 'Queenstown',
```

In [51]:
```
data.head()
```

Out[51]:

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	
100	0	3	Petranec, Miss. Matilda	female	28.00	0	0	349245	7.8958	S
101	0	3	Petroff, Mr. Pastcho ("Pentcho")	male	28.75	0	0	349215	7.8958	S
102	0	1	White, Mr. Richard Frasar	male	21.00	0	1	35281	77.2875	S
103	0	3	Johansson, Mr. Gustaf Joel	male	33.00	0	0	7540	8.6542	S
104	0	3	Gustafsson, Mr. Anders Vilhelm	male	37.00	2	0	3101276	7.9250	S

Группировки

In [52]:
```
data.groupby('gender').mean()
```

Out[52]:

gender	survived	pclass	age	sibsp	parch	fare
female	0.734545	2.127273	28.342727	0.665455	0.669091	46.271046
male	0.191860	2.387597	30.474496	0.406977	0.217054	25.228002

In [53]:
```
data.groupby('gender')['survived'].mean()
```

Out[53]:
```
gender
female    0.734545
male      0.191860
Name: survived, dtype: float64
```

In [54]:
```
splits = data.groupby('gender')
splits
```

Out[54]:
```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fb4df7aa4f0>
```

In [55]:
```
splits.get_group('female')  # тот же результат, что и data.loc[data['gender'] =
```

Out[55]:

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	
100	0	3	Petranec, Miss. Matilda	female	28.00	0	0	349245	7.8958	So
106	1	3	Salkjelsvik, Miss. Anna Kristine	female	21.00	0	0	343120	7.6500	So
109	1	3	Moran, Miss. Bertha	female	28.75	1	0	371110	24.1500	Q

passenger_id	survived	pclass	name	gender	age	sibsp	parch	ticket	fare	
111	0	3	Zabour, Miss. Hileni	female	14.50	1	0	2665	14.4542	
113	0	3	Jussila, Miss. Katriina	female	20.00	1	0	4136	9.8250	So
...	...	...	...	...	...	...	...	...	...	
880	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.00	0	1	230433	26.0000	So
882	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.00	0	0	7552	10.5167	So
885	0	3	Rice, Mrs. William (Margaret Norton)	female	39.00	0	5	382652	29.1250	Q
887	1	1	Graham, Miss. Margaret Edith	female	19.00	0	0	112053	30.0000	So
888	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	28.75	1	2	W./C. 6607	23.4500	So

275 rows × 10 columns

In [56]:
```
splits.count()
```

Out[56]:

gender	survived	pclass	name	age	sibsp	parch	ticket	fare	embarked
female	275	275	275	275	275	275	275	275	275
male	516	516	516	516	516	516	516	516	516

In [57]:
```
splits.mean()
```

Out[57]:

gender	survived	pclass	age	sibsp	parch	fare
female	0.734545	2.127273	28.342727	0.665455	0.669091	46.271046
male	0.191860	2.387597	30.474496	0.406977	0.217054	25.228002

In [58]:
```
splits['survived'].mean()
```

Out[58]:
```
gender
```

```
female    0.734545
male      0.191860
Name: survived, dtype: float64
```

In [59]:
```
splits.first()
```

Out[59]:

gender	survived	pclass	name	age	sibsp	parch	ticket	fare	embarked
female	0	3	Petranec, Miss. Matilda	28.00	0	0	349245	7.8958	Southampton
male	0	3	Petroff, Mr. Pastcho ("Pentcho")	28.75	0	0	349215	7.8958	Southampton

In [60]:
```
splits['survived'].agg(['mean', 'std', 'count'])
```

Out[60]:

gender	mean	std	count
female	0.734545	0.442380	275
male	0.191860	0.394146	516

Seaborn

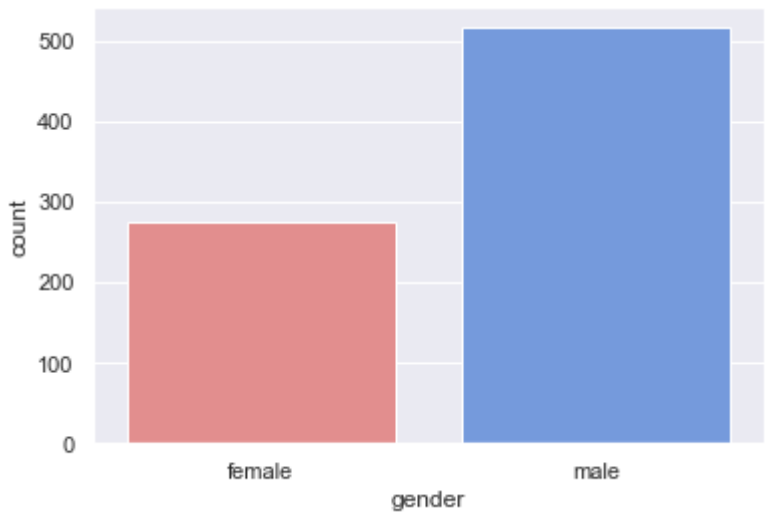
In [61]:
```
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
sns.set(style="darkgrid")
```

In [62]:
```
pal = dict(male="#6495ED", female="#F08080")
```

In [63]:
```
data['gender'].value_counts()
```

Out[63]:
```
male      516
female    275
Name: gender, dtype: int64
```

In [64]:
```
plot = sns.countplot(x='gender', data=data, palette=pal)
plot.figure.savefig('1.png')
```

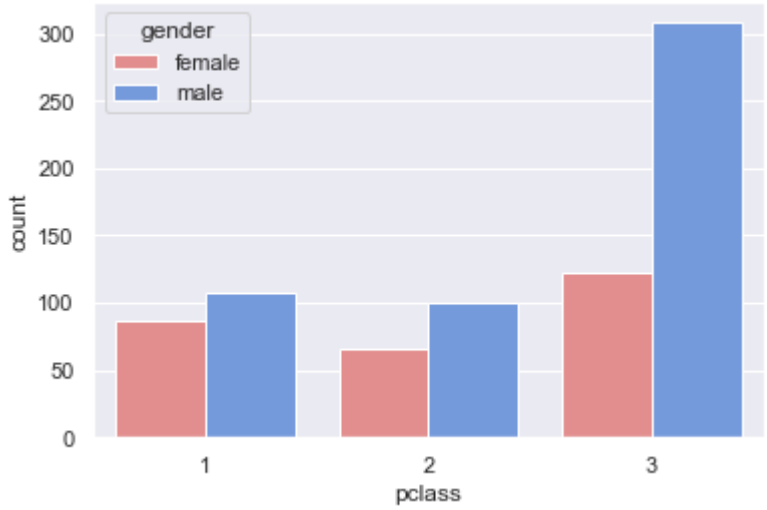


In [65]:
```
pd.crosstab(data['gender'], data['pclass'])
```

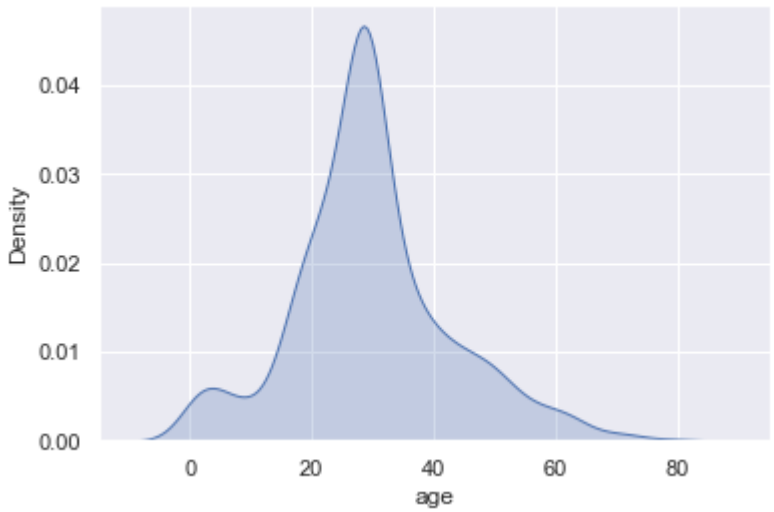
Out[65]:

pclass	1	2	3
gender			
female	87	66	122
male	108	100	308

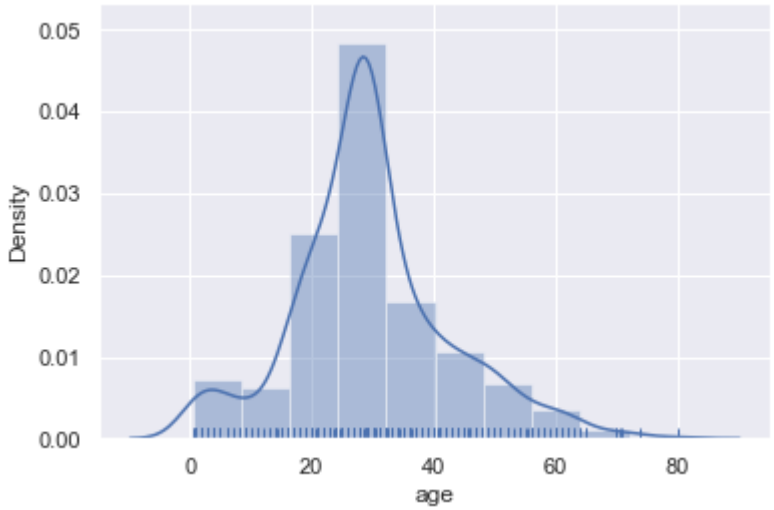
In [66]:
```
sns.countplot(x='pclass', data=data, hue='gender', palette=pal);
```



In [67]:
```
sns.kdeplot(data['age'], shade=True);
```



In [68]:
```
sns.distplot(data['age'], bins=10, rug=True);
```



Обработка категориальных признаков

In [69]:
```
for col in data:
    if data[col].dtype == 'object':
        print(col, len(list(data[col].unique())))
```

```
name 791
gender 2
ticket 619
embarked 3
```

In [70]:
```
data['gender'].unique()
```

Out[70]:
```
array(['female', 'male'], dtype=object)
```

In [71]:
```
data['embarked'].unique()
```

Out[71]:
```
array(['Southampton', 'Queenstown', 'Cherbourg'], dtype=object)
```

In [72]:
```
data.drop(['name', 'ticket'], 1, inplace=True)
```

In [73]:
```
data['gender'] = data['gender'].map({'male': 0, 'female': 1})
```

In [74]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le.fit(data['embarked'])
le.transform(data['embarked']);
```

In [75]:
```
from sklearn.preprocessing import LabelBinarizer

lb = LabelBinarizer()
lb.fit(data['embarked'])
cat_lb = lb.transform(data['embarked'])
cat_lb
```

Out[75]:
```
array([[0, 0, 1],
       [0, 0, 1],
       [0, 0, 1],
       ...,
       [0, 0, 1],
       [1, 0, 0],
       [0, 1, 0]])
```

In [76]:
```
a = pd.DataFrame(cat_lb, index=data.index)
a.rename(columns={0: 'From_Cherbourg', 1: 'From_Queenstown', 2: 'From_Southam
```

In [77]:
```
data = pd.concat([data, a], axis=1)
```

In [78]:
```
data.drop(['embarked'], 1, inplace=True)
```

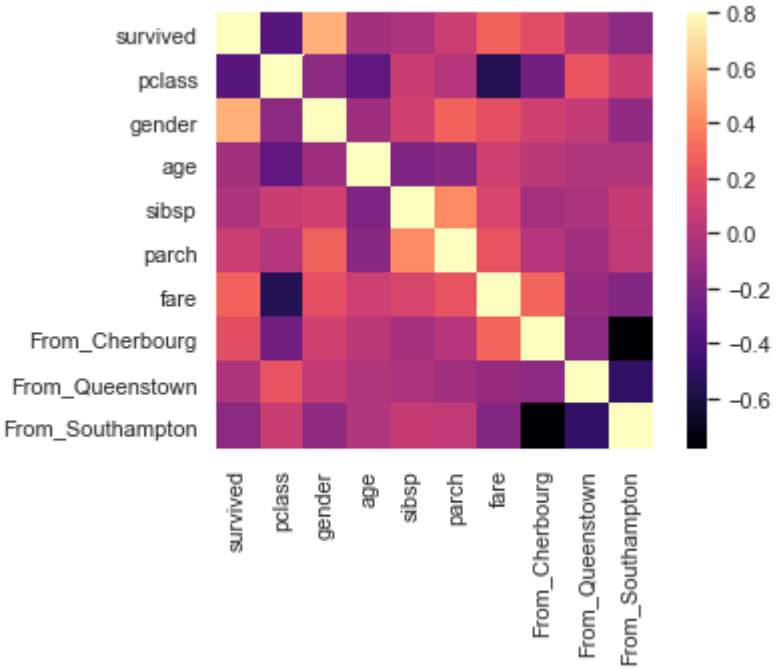
In [79]:
```
data.head()
```

Out[79]:

passenger_id	survived	pclass	gender	age	sibsp	parch	fare	From_Cherbourg	From_
100	0	3	1	28.00	0	0	7.8958	0	
101	0	3	0	28.75	0	0	7.8958	0	
102	0	1	0	21.00	0	1	77.2875	0	
103	0	3	0	33.00	0	0	8.6542	0	
104	0	3	0	37.00	2	0	7.9250	0	

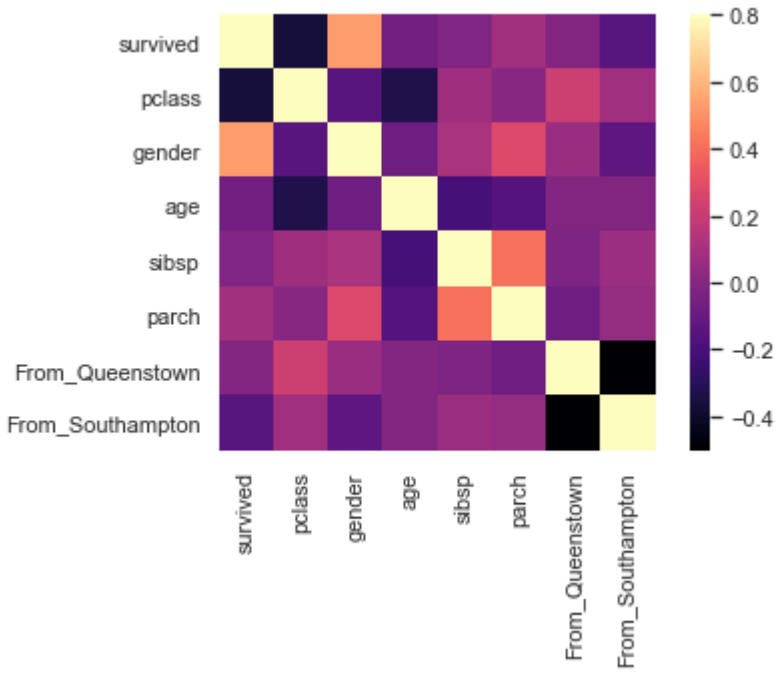
Корреляция

In [80]:
```
corr = data.corr()
sns.heatmap(corr, vmax=.8, square=True, cmap='magma');
```



In [81]:
```
data.drop(['fare', 'From_Cherbourg'], 1, inplace=True)
```

In [82]:
```
corr = data.corr()
sns.heatmap(corr, vmax=.8, square=True, cmap='magma');
```



Обучение модели

In [83]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

In [84]:
```
X = data.loc[:, data.columns != 'survived']
y = data.loc[:, data.columns == 'survived']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rand
```

In [85]:
```
forest = RandomForestClassifier(n_estimators = 50, max_depth = 3)
forest.fit(X_train, y_train)
y_pred = forest.predict(X_test)
```

Метрики качества

In [86]:
```
from sklearn.metrics import accuracy_score, f1_score

print("Accuracy test: ", accuracy_score(y_pred, y_test))
print("f1 test: ", f1_score(y_pred, y_test))
```

```
Accuracy test:  0.7647058823529411
f1 test:  0.631578947368421
```

In [87]:
```
y_pred_train = forest.predict(X_train)

print("Accuracy train: ", accuracy_score(y_pred_train, y_train))
print("f1 train: ", f1_score(y_pred_train, y_train))
```

```
Accuracy train:  0.8300180831826401
f1 train:  0.7025316455696201
```

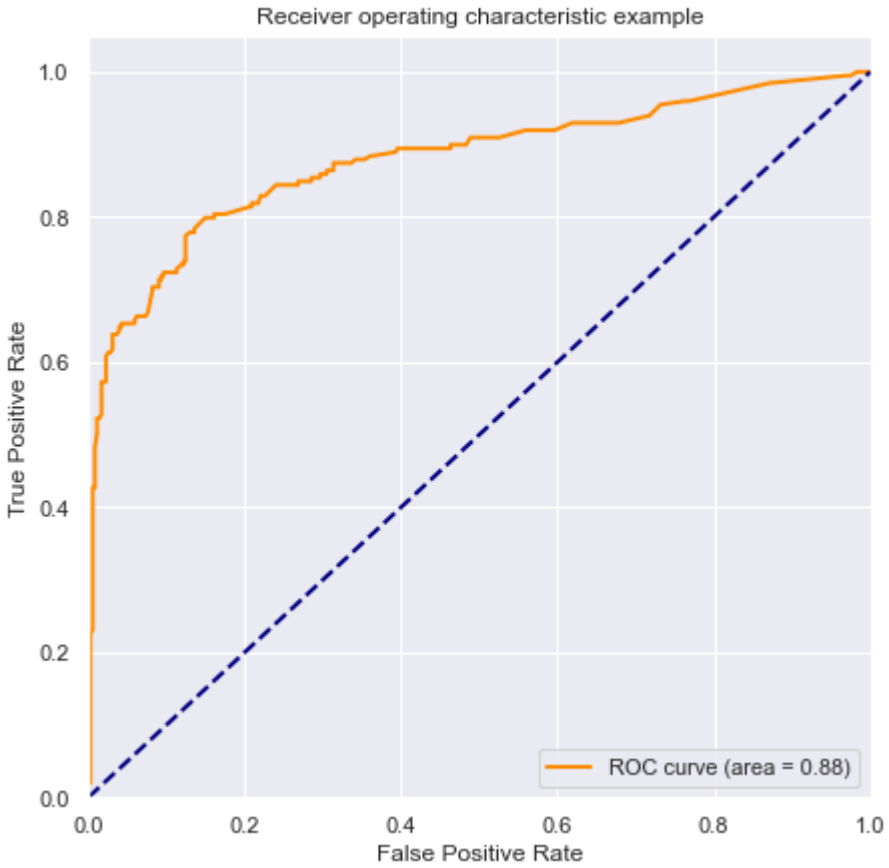
In [88]:
```
y_pred_proba_train = forest.predict_proba(X_train)
y_pred_proba_test = forest.predict_proba(X_test)
```

In [89]:
```
from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve, auc
from sklearn import metrics

fpr, tpr, thresholds = metrics.roc_curve(y_train, y_pred_proba_train[:, 1])
```

```
roc_auc = auc(fpr, tpr)
print("auc train: ", roc_auc)
```

```
auc train:  0.877927774465548
```

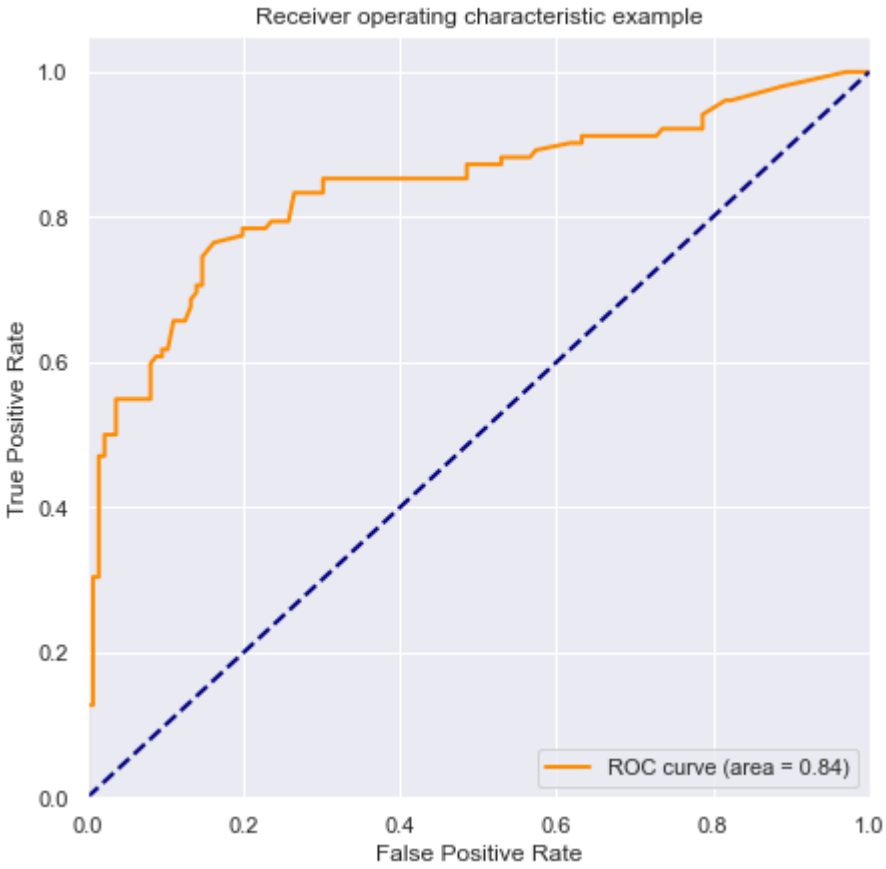
In [90]:
```
plt.figure(figsize = (7,7))
lw = 2
plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example')
plt.legend(loc="lower right")
plt.show()
```



In [91]:
```
fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred_proba_test[:, 1])
roc_auc = auc(fpr, tpr)
print("auc test: ", roc_auc)
```

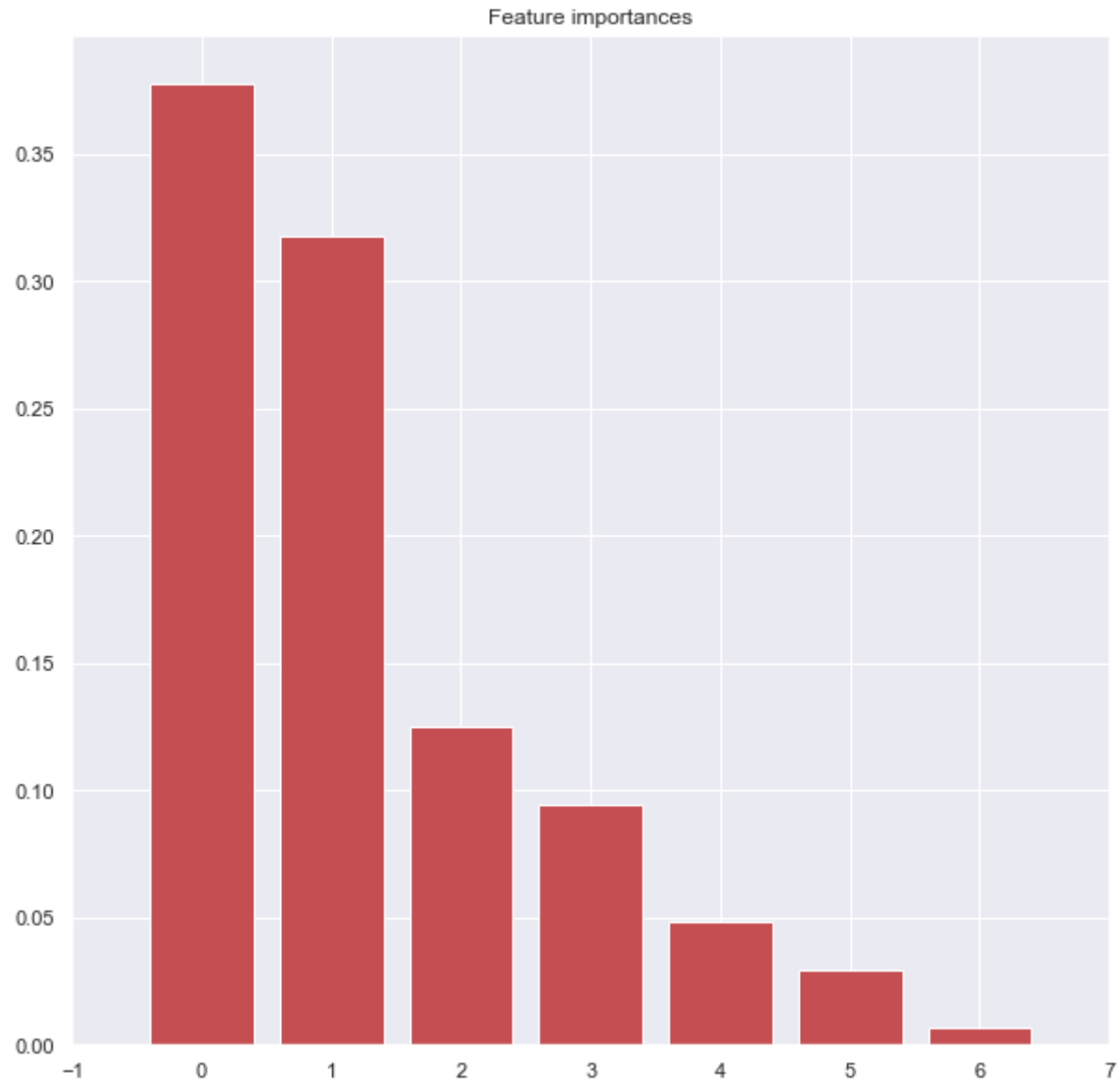
```
auc test:  0.8407583621683967
```

In [92]:
```
plt.figure(figsize = (7,7))
lw = 2
plt.plot(fpr, tpr, color='darkorange', lw=lw, label='ROC curve (area = %0.2f)
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()
```



In [93]:
```
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_], axis=
indices = np.argsort(importances)[::-1]

print("Feature ranking:")

for f in range(X.shape[1]):
    print("%d. %s (%f)" % (f + 1, X.columns[indices[f]], importances[indices[

plt.figure(figsize=(10,10))
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices], color="r", align="center")

plt.xlim([-1, X.shape[1]])
plt.show()
```

```
Feature ranking:
1. pclass (0.377847)
2. gender (0.317338)
3. age (0.124779)
4. sibsp (0.094595)
5. parch (0.048629)
6. From_Southampton (0.029693)
7. From_Queenstown (0.007119)
```


Feature importances
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−1
0
1
2
3
4
5
6
7

In [ ]: