Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Ордена Трудового Красного Знамени федеральное государственное
бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»
Кафедра «МКиИТ»

Лабораторная работа №3

по дисциплине «Data mining»

Москва 2023

ЛР3. NB, LR, SVM

In [1]:

```
import numpy as np
import pandas as pd

from sklearn import preprocessing
import matplotlib.pyplot as plt
plt.rc("font", size=14)
import seaborn as sns
sns.set(style="white") #white background style for seaborn plots
sns.set(style="whitegrid", color_codes=True)

import warnings
warnings.simplefilter(action='ignore')
```

In [2]:

```
# Создадим DataFrame train_df из CSV train.csv
train_df = pd.read_csv("./train.csv")

# Создадим DataFrame test_df из CSV test.csv
test_df = pd.read_csv("./test.csv")

train_df.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

In [3]:

```
#Посчитайте количество объектов в DF train и test
train_df.count()
```

Out[3]:

```
PassengerId    891
Survived       891
Pclass         891
Name           891
Sex            891
Age            714
SibSp          891
Parch          891
Ticket         891
Fare           891
Cabin          204
Embarked       889
dtype: int64
```

In [4]:

```
test_df.count()
```

Out[4]:

```
PassengerId    418
Pclass         418
Name           418
Sex            418
Age            332
SibSp          418
Parch          418
Ticket         418
Fare           417
Cabin           91
Embarked       418
dtype: int64
```

Примечание. В тестовых данных нет целевой переменной (т. е. столбец «Survival» отсутствует), поэтому цель состоит в том, чтобы предсказать эту переменную с использованием различных алгоритмов машинного обучения, таких как логистическая регрессия.

In [5]:

```
# Проверьте, есть ли в данных train_df пропущенные значения
train_df.isnull().sum ()
```

Out[5]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

Давайте разберемся с графиой "Age" Ответьте на вопросы и сделайте следующие дейтсвия

Сколько процентов значений пропущено? Найдите медиану и среднее значение переменной. Постойте гистаграмму. Какое значение больше: медиана или среднее?

In [6]:

```
train_df.isnull().sum ()/ len(train_df)* 100
```

Out[6]:

```
PassengerId     0.000000
Survived        0.000000
Pclass          0.000000
Name            0.000000
Sex             0.000000
Age            19.865320
SibSp           0.000000
Parch           0.000000
Ticket          0.000000
Fare            0.000000
Cabin          77.104377
Embarked        0.224467
dtype: float64
```

In [7]:

```
train_df.groupby(train_df['Age'].isnull()).mean()
```

Out[7]:

Age	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
False	448.582633	0.406162	2.236695	29.699118	0.512605	0.431373	34.694514
True	435.581921	0.293785	2.598870	NaN	0.564972	0.180791	22.158567

In [8]:

```
train_df.groupby(train_df['Age'].isnull()).median()
```

Out[8]:

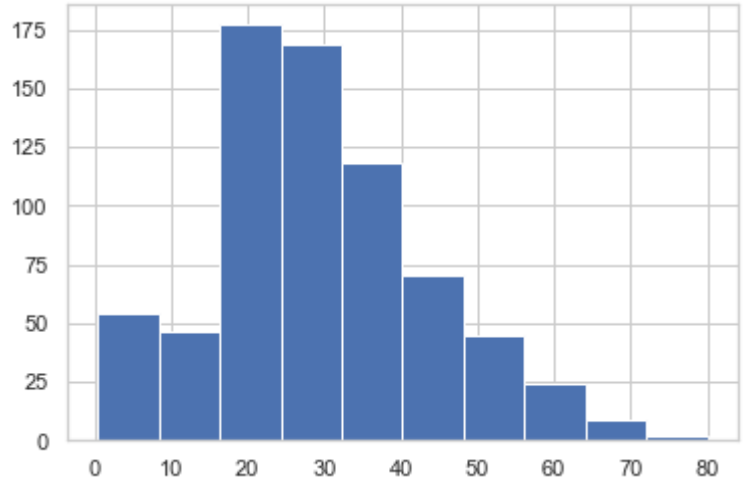
Age	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
False	445	0	2	28.0	0	0	15.7417
True	452	0	3	NaN	0	0	8.0500

In [53]:

```
train_df['Age'].hist()
```

Out[53]:

```
<AxesSubplot:>
```



Так как график распределения смещён вправа, использование среднего значения может дать нам необъективные результаты из-за заполнения возрастов, которые старше желаемого. Чтобы справиться с этим, мы будем использовать медиану для вменения пропущенных значений.

Давайте разберемся с графиой "Cabin"

Сколько процентов значений пропущено?

In [9]:

```
train_df['Cabin'].isnull().sum ()/ len(train_df)* 100
```

Out[9]:

```
77.10437710437711
```

Если в столбце больше половины пропусков, то он не информативен для нас. В дальнейшем мы его уберем

Давайте разберемся с графиой "Embarked"

Сколько процентов значений пропущено?

In [10]:

```
train_df['Embarked'].isnull().sum ()/ len(train_df)* 100
```

Out[10]:

```
0.22446689113355783
```

In [11]:

```
train_data = train_df.copy()
```

Основываясь на оценке отсутствующих значений в наборе данных, внесите в данные следующие изменения:

Если в строке отсутствует «Age», вставьте средний возраст. Если отсутствует «Embarked», замените его на наиболее распространенный порт посадки. Удалите столбец "Cabin"

In [12]:

```
train_data['Age'].fillna(train_data['Age'].mean(), inplace = True)
train_data.drop(['Cabin'], axis = 1, inplace = True)
train_data['Embarked'].replace(np.nan, 'S', inplace = True)
train_data.head()
```

Out[12]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

Выполните проверку на пустые значения. Таковых остаться не должно

In [13]:

```
train_data.isnull().sum ()/ len(train_df)* 100
```

Out[13]:

```
PassengerId    0.0
Survived       0.0
Pclass         0.0
Name           0.0
Sex            0.0
Age            0.0
SibSp          0.0
Parch          0.0
Ticket         0.0
Fare           0.0
Embarked       0.0
dtype: float64
```

Согласно словарю данных Kaggle, и SibSp, и Parch относятся к путешествиям с семьей. Для простоты (и для учета возможной мультиколлинеарности) я объединим влияние этих переменных в один категориальный предиктор: путешествовал ли этот человек один или нет (0 или 1). Не забудьте удалить SibSp и Parch (Пока работаем с train_data)

In [14]:

```
train_data['TravelAlone'] = np.where(train_data['SibSp'] + train_data['Parch']> 0,  1,0)
train_data.drop(['SibSp'], axis = 1, inplace = True)
train_data.drop(['Parch'], axis = 1, inplace = True)
train_data.head()
```

Out[14]:

	PassengerId	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Embarked	Tr
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	A/5 21171	7.2500	S	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	PC 17599	71.2833	C	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	STON/O2. 3101282	7.9250	S	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	113803	53.1000	S	
4	5	0	3	Allen, Mr. William Henry	male	35.0	373450	8.0500	S	

также создадим категориальные переменные для класса пассажира ("Pclass"), пола ("Sex") и порта посадки ("Embarked"). Используем функцию pd.get_dummies Не забудьте удалить колонки, из которых делаете dummie-переменные

In [15]:

```
training=pd.get_dummies(train_data, columns=['Pclass','Sex','Embarked'], drop_first= Fal

train_data.drop(['Pclass', 'Sex', 'Embarked'], axis= 1 , inplace= True )

final_train = training
final_train.head()
```

Out[15]:

	PassengerId	Survived	Name	Age	Ticket	Fare	TravelAlone	Pclass_1	Pclass
0	1	0	Braund, Mr. Owen Harris	22.0	A/5 21171	7.2500	1	0	
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	PC 17599	71.2833	1	1	
2	3	1	Heikkinen, Miss. Laina	26.0	STON/O2. 3101282	7.9250	0	0	
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	113803	53.1000	1	1	
4	5	0	Allen, Mr. William Henry	35.0	373450	8.0500	0	0	

Теперь примените те же изменения к тестовым данным.

Применим то же значение для «Возраст» в тестовых данных, что и для моих данных обучения (если отсутствует, возраст = 28). Уберем «Cabin» из тестовых данных В переменной порта "Embarked" не было пропущенных значений. Добавим dummie переменные. Наконец, заполним 1 пропущенное значение для «Fare» с медианой 14,45.

In [16]:

```
final_test=train_df.copy()
final_test.drop(['Cabin'], axis= 1 , inplace= True )
final_test['Age'].replace(np.nan, '28', inplace = True)
final_test['Fare'].replace(np.nan, '14,45', inplace = True)
final_test = pd.get_dummies(train_df, columns=['Pclass','Sex','Embarked'], drop_first= F


final_test.head()
```

Out[16]:

	PassengerId	Survived	Name	Age	SibSp	Parch	Ticket	Fare	Cabin	Pclass_
0	1	0	Braund, Mr. Owen Harris	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	Heikkinen, Miss. Laina	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	1	0	113803	53.1000	C123	
4	5	0	Allen, Mr. William Henry	35.0	0	0	373450	8.0500	NaN	

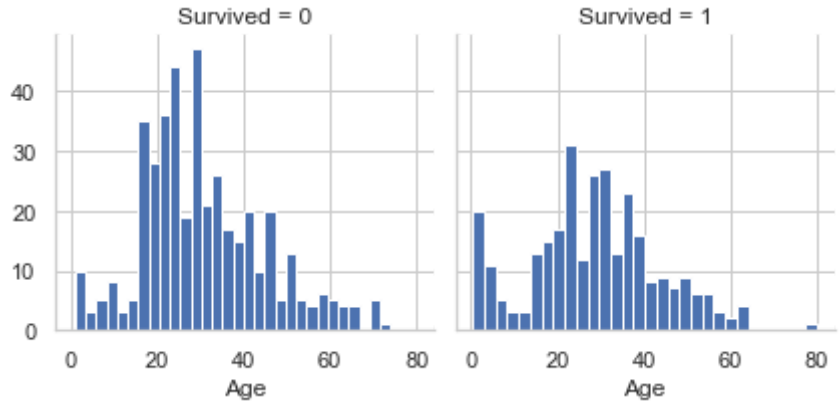
Оцените выживаемость Пассажиров до 16 лет

In [17]:

```
sns.set_style('whitegrid')
%matplotlib inline

g = sns.FacetGrid(final_test, col='Survived')
g.map(plt.hist, 'Age', bins=30)
```

Out[17]:

```
<seaborn.axisgrid.FacetGrid at 0x251f4fbfbb0>
```

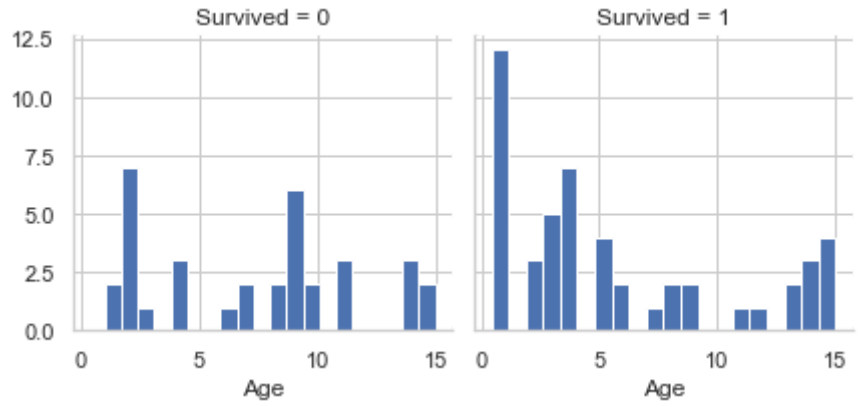


In [18]:

```
age16=final_test.loc[lambda final_test: final_test['Age'] < 16, :]
sns.set_style('whitegrid')
%matplotlib inline

g = sns.FacetGrid(age16, col='Survived')
g.map(plt.hist, 'Age', bins=20)
```

Out[18]:

```
<seaborn.axisgrid.FacetGrid at 0x251f33c62b0>
```



Учитывая выживаемость пассажиров моложе 16 лет,включим в свой набор данных еще одну категориальную переменную: «IsMinor». Значение 1 - если меньше 16 лет, 0 - если больше

In [19]:

```
final_train['IsMinor']=np.where(train_df['Age'] < 16,  1,0)
final_test['IsMinor']=np.where(train_df['Age'] < 16,  1,0)
final_train.head()
```

Out[19]:

	PassengerId	Survived	Name	Age	Ticket	Fare	TravelAlone	Pclass_1	Pclass
0	1	0	Braund, Mr. Owen Harris	22.0	A/5 21171	7.2500	1	0	
1	2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0	PC 17599	71.2833	1	1	
2	3	1	Heikkinen, Miss. Laina	26.0	STON/O2. 3101282	7.9250	0	0	
3	4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0	113803	53.1000	1	1	
4	5	0	Allen, Mr. William Henry	35.0	373450	8.0500	0	0	

Определите самый безопасный класс Определите, кому удаолсь выжить с большей вероятностью: кто путешествовал в одиночку или нет

In [20]:

```
final_train.groupby(['IsMinor']) ['Survived'].value_counts(normalize=True)
```

Out[20]:

```
IsMinor  Survived
0        0           0.637376
         1           0.362624
1        1           0.590361
         0           0.409639
Name: Survived, dtype: float64
```

Выбор признаков для анализа

Рекурсивное устранение признаков

рекурсивное исключение функций (RFE) заключается в выборе функций путем рекурсивного рассмотрения все меньших и меньших наборов функций. Во-первых, оценщик обучается на начальном наборе признаков, и важность каждого признака определяется либо с помощью атрибута «coef_», либо с помощью атрибута «feature_importances_». Затем наименее важные функции

удаляются из текущего набора функций. Эта процедура рекурсивно повторяется для сокращенного набора до тех пор, пока в конечном итоге не будет достигнуто желаемое количество функций для выбора.

Изучите материалы ниже

https://www.helenkapatsa.ru/kross-validatsiia/
https://www.codecamp.ru/blog/cross-validation-k-fold/
http://scikit-learn.org/stable/modules/feature_selection.html

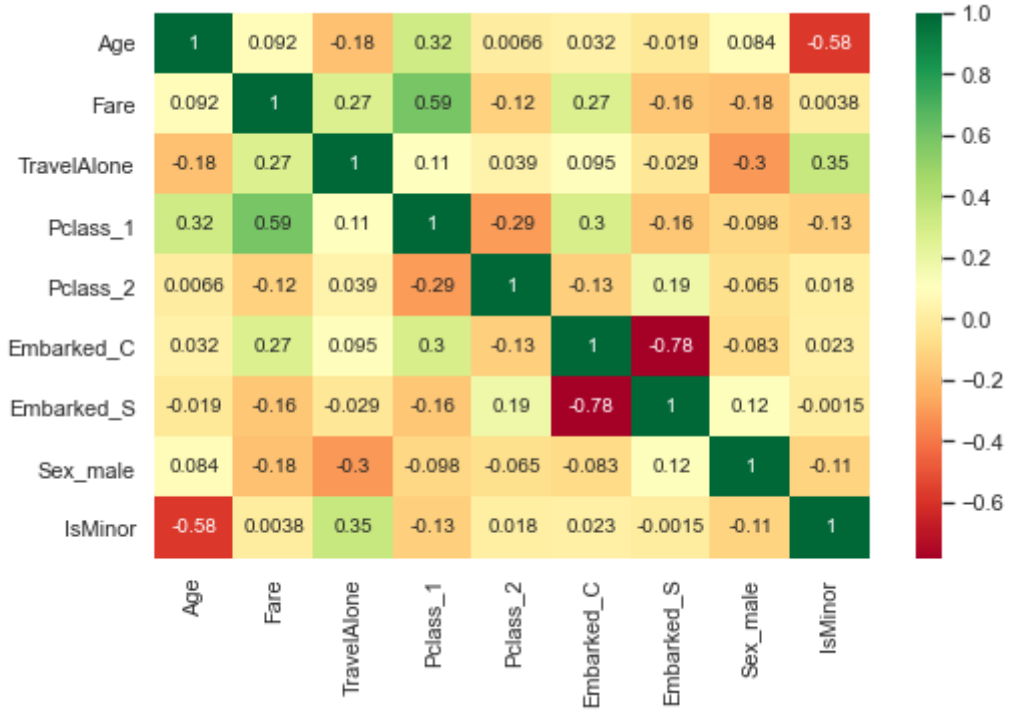
In [44]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

cols = ["Age","Fare","TravelAlone","Pclass_1","Pclass_2","Embarked_C","Embarked_S","Sex_
X = final_train[cols]
y = final_train['Survived']
# Создайте logreg и вычислите важность функций
model = LogisticRegression()
# создайте модель RFE и выберите 8 атрибутов
rfe = RFE(model, 8)
rfe = rfe.fit(X, y)
# резюмируем  выбор атрибутов
print('Selected features: %s' % list(X.columns[rfe.support_]))
```

```
Selected features: ['Age', 'TravelAlone', 'Pclass_1', 'Pclass_2', 'Embarke
d_C', 'Embarked_S', 'Sex_male', 'IsMinor']
```

In [45]:

```
Selected_features = ["Age","Fare","TravelAlone","Pclass_1","Pclass_2","Embarked_C","Emba
X = final_train[Selected_features]

plt.subplots(figsize=(8, 5))
sns.heatmap(X.corr(), annot=True, cmap="RdYlGn")
plt.show()
```



представьте, перед вами стоит задача оценки качества работы модели машинного обучения и сравнения таких моделей межд собой

идея: разделение датасета на выборку для обучения и тестирования

Оценка модели на основе простого разделения train/test с использованием функции `train_test_split()`

In [46]:

```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, classification_report, precision_score, reca
from sklearn.metrics import confusion_matrix, precision_recall_curve, roc_curve, auc, lo

# create X (features) and y (response)
X = final_train[Selected_features]
y = final_train['Survived']

# можно использовать разделение обучения/тестирования с разными значениями random_state
# мы можем изменить значения random_state, которые изменят показатели точности
# результаты сильно меняются, поэтому результаты тестирования являются оценкой с высокой
# test_size разделяет выборку на тестовую и убучающую в соотношении 20/80
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)

# check classification scores of logistic regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
y_pred_proba = logreg.predict_proba(X_test)[:, 1]
[fpr, tpr, thr] = roc_curve(y_test, y_pred_proba)
print('Train/Test split results:')
print(logreg.__class__.__name__+" accuracy is %2.3f" % accuracy_score(y_test, y_pred))
print(logreg.__class__.__name__+" log_loss is %2.3f" % log_loss(y_test, y_pred_proba))
print(logreg.__class__.__name__+" auc is %2.3f" % auc(fpr, tpr))
```

```
Train/Test split results:
LogisticRegression accuracy is 0.788
LogisticRegression log_loss is 0.504
LogisticRegression auc is 0.840
```

Оценка модели на основе K-fold cross-validation с использованием функции cross_val_score()

In [47]:

```
# 10-fold cross-validation logistic regression
logreg = LogisticRegression()
# Используем cross_val_score function
# Мы передаём полные X and y, а не X_train и y_train, функция сама разбивает данные
# cv=10 for 10 folds
# scoring = {'accuracy', 'neg_log_loss', 'roc_auc'} в качестве метрик оценивания результа
scores_accuracy = cross_val_score(logreg, X, y, cv=10, scoring='accuracy')
scores_log_loss = cross_val_score(logreg, X, y, cv=10, scoring='neg_log_loss')
scores_auc = cross_val_score(logreg, X, y, cv=10, scoring='roc_auc')
print('K-fold cross-validation results:')
print(logreg.__class__.__name__+" average accuracy is %2.3f" % scores_accuracy.mean())
print(logreg.__class__.__name__+" average log_loss is %2.3f" % -scores_log_loss.mean())
print(logreg.__class__.__name__+" average auc is %2.3f" % scores_auc.mean())
```

```
K-fold cross-validation results:
LogisticRegression average accuracy is 0.799
LogisticRegression average log_loss is 0.455
LogisticRegression average auc is 0.850
```

Оценка модели на основе K-fold cross-validation с использованием функции cross_validate()

In [48]:

```
from sklearn.model_selection import cross_validate

scoring = {'accuracy': 'accuracy', 'log_loss': 'neg_log_loss', 'auc': 'roc_auc'}

modelCV = LogisticRegression()

results = cross_validate(modelCV, X, y, cv=10, scoring=list(scoring.values()),
                         return_train_score=False)

print('K-fold cross-validation results:')
for sc in range(len(scoring)):
    print(modelCV.__class__.__name__+" average %s: %.3f (+/-%.3f)" % (list(scoring.keys(
                             if list(scoring.values())[sc]=='neg_log_loss'
                             else results['test_%s' % list(scoring.values())[sc]].mean
                             results['test_%s' % list(scoring.values())[sc]].std()))
```

```
K-fold cross-validation results:
LogisticRegression average accuracy: 0.799 (+/-0.025)
LogisticRegression average log_loss: 0.455 (+/-0.039)
LogisticRegression average auc: 0.850 (+/-0.029)
```

Формула Байеса

$$P(A|B) = \frac{P(B \mid A)P(A)}{P(B)}$$

где

P (A) P(A) – априорная вероятность гипотезы A (смысл такой терминологии см. ниже);

P (A | B) – вероятность гипотезы A при наступлении события B (апостериорная вероятность);

P (B | A) – вероятность наступления события B при истинности гипотезы A;

P (B) P(B) – полная вероятность наступления события B.

In [49]:

```
import matplotlib.pyplot as plt
from scipy.stats.stats import pearsonr
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, precision_score
```

In [50]:

```
classifier = GaussianNB()
classifier.fit(X_train, y_train)

#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2
```

Out[50]:

```
GaussianNB()
```

In [51]:

```
def cross_validate(estimator, train, validation):
    X_train = train[0]
    Y_train = train[1]
    X_val = validation[0]
    Y_val = validation[1]
    train_predictions = classifier.predict(X_train)
    train_accuracy = accuracy_score(train_predictions, Y_train)
    train_recall = recall_score(train_predictions, Y_train)
    train_precision = precision_score(train_predictions, Y_train)

    val_predictions = classifier.predict(X_val)
    val_accuracy = accuracy_score(val_predictions, Y_val)
    val_recall = recall_score(val_predictions, Y_val)
    val_precision = precision_score(val_predictions, Y_val)

    print('Model metrics')
    print('Accuracy  Train: %.2f, Validation: %.2f' % (train_accuracy, val_accuracy))
    print('Recall    Train: %.2f, Validation: %.2f' % (train_recall, val_recall))
    print('Precision Train: %.2f, Validation: %.2f' % (train_precision, val_precision))

cross_validate(classifier, (X_train, y_train), (X_test, y_test))
```

```
Model metrics
Accuracy  Train: 0.76, Validation: 0.72
Recall    Train: 0.71, Validation: 0.76
Precision Train: 0.59, Validation: 0.53
```

SVM

In [52]:

```
from sklearn.svm import SVC
# Declaring the SVC with no tunning
classifier = SVC()

# Fitting the data. This is where the SVM will learn
classifier.fit(X_train, y_train)

# Predicting the result and giving the accuracy
score = classifier.score(X_test, y_test)

print(score)
```

```
0.6312849162011173
```

In [30]:

```
#Посчитайте score, если train set будет состоять только из 3 переменных: ['Sex', 'Age',
```

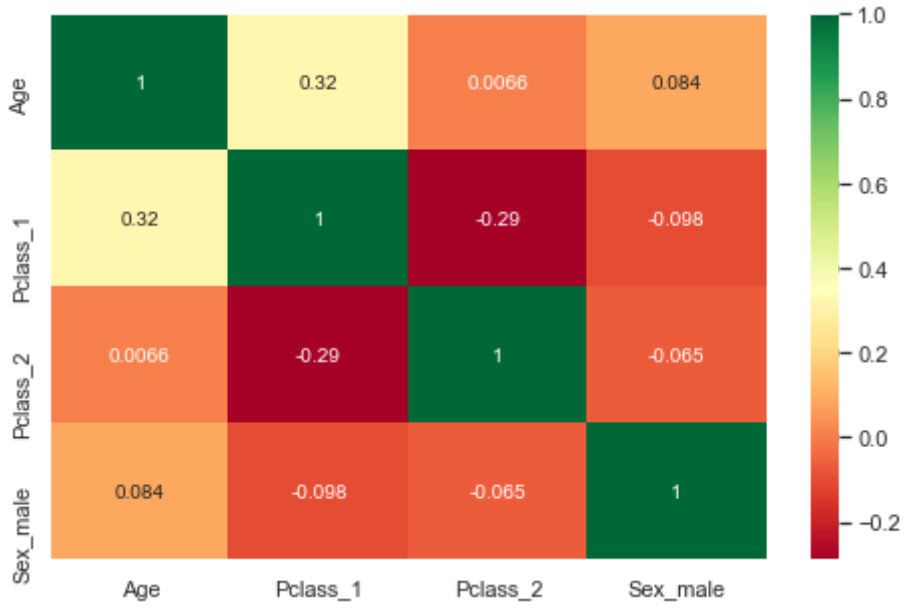
In [41]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

cols = ["Age","Pclass_1","Pclass_2","Sex_male"]
X = final_train[cols]
y = final_train['Survived']
# Создайте logreg и вычислите важность функций
model = LogisticRegression()
# создайте модель RFE и выберите 8 атрибутов
rfe = RFE(model, 8)
rfe = rfe.fit(X, y)
# резюмируем  выбор атрибутов
print('Selected features: %s' % list(X.columns[rfe.support_]))
```

```
Selected features: ['Age', 'Pclass_1', 'Pclass_2', 'Sex_male']
```

In [22]:

```
Selected_features = ['Age', 'Pclass_1', 'Pclass_2', 'Sex_male']
X = final_test[Selected_features]

plt.subplots(figsize=(8, 5))
sns.heatmap(X.corr(), annot=True, cmap="RdYlGn")
plt.show()
```



In [23]:

```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, classification_report, precision_score, reca
from sklearn.metrics import confusion_matrix, precision_recall_curve, roc_curve, auc, lo

# create X (features) and y (response)
X = final_train[Selected_features]
y = final_train['Survived']

# можно использовать разделение обучения/тестирования с разными значениями random_state
# мы можем изменить значения random_state, которые изменят показатели точности
# результаты сильно меняются, поэтому результаты тестирования являются оценкой с высокой
# test_size разделяет выборку на тестовую и убучающую в соотношении 20/80
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)

# check classification scores of logistic regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
y_pred_proba = logreg.predict_proba(X_test)[:, 1]
[fpr, tpr, thr] = roc_curve(y_test, y_pred_proba)
print('Train/Test split results:')
print(logreg.__class__.__name__+" accuracy is %2.3f" % accuracy_score(y_test, y_pred))
print(logreg.__class__.__name__+" log_loss is %2.3f" % log_loss(y_test, y_pred_proba))
print(logreg.__class__.__name__+" auc is %2.3f" % auc(fpr, tpr))
```

```
Train/Test split results:
LogisticRegression accuracy is 0.765
LogisticRegression log_loss is 0.511
LogisticRegression auc is 0.820
```

In [24]:

```
# 10-fold cross-validation logistic regression
logreg = LogisticRegression()
# Используем cross_val_score function
# Мы передаём полные X and y, а не X_train и y_train, функция сама разбивает данные
# cv=10 for 10 folds
# scoring = {'accuracy', 'neg_log_loss', 'roc_auc'} в качестве метрик оценивания результа
scores_accuracy = cross_val_score(logreg, X, y, cv=10, scoring='accuracy')
scores_log_loss = cross_val_score(logreg, X, y, cv=10, scoring='neg_log_loss')
scores_auc = cross_val_score(logreg, X, y, cv=10, scoring='roc_auc')
print('K-fold cross-validation results:')
print(logreg.__class__.__name__+" average accuracy is %2.3f" % scores_accuracy.mean())
print(logreg.__class__.__name__+" average log_loss is %2.3f" % -scores_log_loss.mean())
print(logreg.__class__.__name__+" average auc is %2.3f" % scores_auc.mean())
```

```
K-fold cross-validation results:
LogisticRegression average accuracy is 0.791
LogisticRegression average log_loss is 0.457
LogisticRegression average auc is 0.845
```

In [25]:

```
from sklearn.model_selection import cross_validate

scoring = {'accuracy': 'accuracy', 'log_loss': 'neg_log_loss', 'auc': 'roc_auc'}

modelCV = LogisticRegression()

results = cross_validate(modelCV, X, y, cv=10, scoring=list(scoring.values()),
                         return_train_score=False)

print('K-fold cross-validation results:')
for sc in range(len(scoring)):
    print(modelCV.__class__.__name__+" average %s: %.3f (+/-%.3f)" % (list(scoring.keys(
                                if list(scoring.values())[sc]=='neg_log_loss'
                                else results['test_%s' % list(scoring.values())[sc]].mean
                                results['test_%s' % list(scoring.values())[sc]].std()))
```

```
K-fold cross-validation results:
LogisticRegression average accuracy: 0.791 (+/-0.020)
LogisticRegression average log_loss: 0.457 (+/-0.034)
LogisticRegression average auc: 0.845 (+/-0.025)
```

In [26]:

```
import matplotlib.pyplot as plt
from scipy.stats.stats import pearsonr
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, recall_score, precision_score
```

In [27]:

```
classifier = GaussianNB()
classifier.fit(X_train, y_train)

#X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2
```

Out[27]:

```
GaussianNB()
```

In [28]:

```
def cross_validate(estimator, train, validation):
    X_train = train[0]
    Y_train = train[1]
    X_val = validation[0]
    Y_val = validation[1]
    train_predictions = classifier.predict(X_train)
    train_accuracy = accuracy_score(train_predictions, Y_train)
    train_recall = recall_score(train_predictions, Y_train)
    train_precision = precision_score(train_predictions, Y_train)

    val_predictions = classifier.predict(X_val)
    val_accuracy = accuracy_score(val_predictions, Y_val)
    val_recall = recall_score(val_predictions, Y_val)
    val_precision = precision_score(val_predictions, Y_val)

    print('Model metrics')
    print('Accuracy  Train: %.2f, Validation: %.2f' % (train_accuracy, val_accuracy))
    print('Recall    Train: %.2f, Validation: %.2f' % (train_recall, val_recall))
    print('Precision Train: %.2f, Validation: %.2f' % (train_precision, val_precision))

cross_validate(classifier, (X_train, y_train), (X_test, y_test))
```

```
Model metrics
Accuracy  Train: 0.79, Validation: 0.78
Recall    Train: 0.72, Validation: 0.81
Precision Train: 0.71, Validation: 0.65
```

In [29]:

```
from sklearn.svm import SVC
# Declaring the SVC with no tunning
classifier = SVC()

# Fitting the data. This is where the SVM will learn
classifier.fit(X_train, y_train)

# Predicting the result and giving the accuracy
score = classifier.score(X_test, y_test)

print(score)
```

```
0.5698324022346368
```