## Понижение размерности

### Цель работы

Применить методы понижения размерности для решения задач машинного обучения.

### Задания для выполнения

1. Загрузите датасет credit\_data.
2. Проверьте датасет на наличие текстовых атрибутов. Замените текстовые атрибуты на числовые без потери качества данных.
3. Выведите информацию о количественных параметрах датасета;
4. Разделите эти данные на тестовую и обучающую выборки;
5. Обучите модель случайных лесов на обучающей выборке. Проверьте точность предсказаний.
6. Оцените полученную модель с помощью метрик.
7. Понизьте размерность данных с помощью метода главных компонент.
8. Обучите заново модель случайных лесов и оцените ее эффективность с помощью метрик.
9. Постройте график зависимости точности модели от размерности данных.
10. Сделайте вывод о применимости модели.

### Методические указания

Для начала работы нам потребуется импортировать необходимые библиотеки:

%matplotlib inline

import matplotlib.pyplot as plt

import seaborn as sns; sns.set()

import numpy as np

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

from sklearn import metrics

В первую очередь загрузим датасет, выведем количество пустых значений в каждом столбце:

data = pd.read\_csv(r'.../credit\_data.csv',delimiter=',')

print(data.isna().sum())

Удалим столбцы не несущие какой-либо полезной информации, а также вынесем метки строк в отдельный массив target:

target = data['Risk']

data = data.drop(['Risk','Unnamed: 0', 'Purpose'], axis=1)

Заменим текстовые категориальные признаки на числовые с помощью функции map. Пустые значения в столбцах заменим на 0.

data['Saving accounts'] = data['Saving accounts'].map({"little":1,"moderate":2,"quite rich":3 ,"rich":4 });

data['Checking account'] = data['Checking account'].map({"little":1,"moderate":2,"rich":3 });

target = target.map({"good":1,"bad":0});

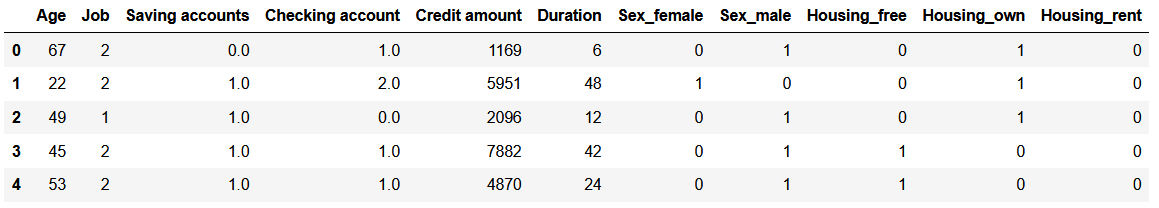
data['Saving accounts'] = data['Saving accounts'].fillna(0)

data['Checking account'] = data['Checking account'].fillna(0)

Заменим полученные категориальные признаки на индикаторы с помощью метода get\_dummies. Это необходимо поскольку модель случайных лесов плохо работает с категориальными признаками, но неплохо обучается с индикаторами.

new\_data = pd.get\_dummies(data)

new\_data.head()



Нормализуем данные и понизим размерность данных до 2-х атрибутов.

from sklearn.cluster import KMeans;

from sklearn.decomposition import PCA;

from sklearn.preprocessing import normalize;

y = KMeans().fit\_predict(new\_data)

X = normalize(new\_data);

x\_PCA = PCA(n\_components=2).fit\_transform(X,2);

print(x\_PCA.shape)

Построим график на основе полученных атрибутов:

plt.scatter(x\_PCA[:,0], x\_PCA[:,1], c=target, cmap='Spectral')

plt.figure()

