```
import numpy as np
import pandas as pd
from typing import Dict, Tuple
from scipy import stats
from IPython.display import Image
from IPython.display import Image
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.datasets import load_iris, load_boston
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.ensemble import ExtraTreesClassifier, ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_error,
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.metrics.pairwise import cosine_similarity, euclidean_distances, manhattan_distan
from collections import defaultdict
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib_venn import venn2
%matplotlib inline
sns.set(style="ticks")
```

## Чтение и обработка данных

```
data = pd.read_csv('winemag-data_first150k.csv')
data.head()
```

| | Unnamed: 0 | country | description | designation | points | price | province | region_1 | reg |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | US | This tremendous 100% varietal wine hails from ... | Martha's Vineyard | 96 | 235.0 | California | Napa Valley | |
| | | | Ripe aromas of fig | Carodorum | | | | | |

```
data.shape
```

```
(150930, 11)
```

```
description_data = data[data['description'].notnull()]
description_data.shape
```

```
(150930, 11)
```

```
title = description_data['designation'].values
title[0:5]
```

```
array(["Martha's Vineyard", 'Carodorum Selección Especial Reserva',
       'Special Selected Late Harvest', 'Reserve', 'La Brûlade'],
      dtype=object)
```

```
descriptions = description_data['description'].values
descriptions[0:5]
```

```
array(['This tremendous 100% varietal wine hails from Oakville and was aged over three y
       'Ripe aromas of fig, blackberry and cassis are softened and sweetened by a slathe
       'Mac Watson honors the memory of a wine once made by his mother in this tremendou
       "This spent 20 months in 30% new French oak, and incorporates fruit from Ponzi's
       'This is the top wine from La Bégude, named after the highest point in the vineya
      dtype=object)
```

```
description_data.keys()
```

```
Index(['Unnamed: 0', 'country', 'description', 'designation', 'points',
       'price', 'province', 'region_1', 'region_2', 'variety', 'winery'],
      dtype='object')
```

```
wine_ids = description_data['Unnamed: 0'].values
wine_ids
```

```
array([     0,      1,      2, ..., 150927, 150928, 150929])
```

```
%%time
tfidf = TfidfVectorizer()
description_matrix = tfidf.fit_transform(descriptions)
description_matrix
```

```
CPU times: user 4.41 s, sys: 30.3 ms, total: 4.44 s
Wall time: 4.45 s
```

```
description_matrix
```

```
<150930x30748 sparse matrix of type '<class 'numpy.float64'>'
	with 5162508 stored elements in Compressed Sparse Row format>
```

## Фильтрация на основе содержания. Метод k-ближайших соседей

```
class SimplerKnnRecomender:
  def __init__(self, X_matrix, X_ids, X_title, X_overview):
        """
        Входные параметры:
        X_matrix - обучающая выборка (матрица объект-признак)
        X_ids - массив идентификаторов объектов
        X_description - массив описаний объектов
        X_overview - массив описаний объектов
        """
        #Сохраняем параметры в переменных объекта
        self._X_matrix = X_matrix
        self.df = pd.DataFrame(
            {'id': pd.Series(X_ids, dtype='int'),
            'description': pd.Series(X_title, dtype='str'),
            'overview': pd.Series(X_overview, dtype='str'),
            'dist': pd.Series([], dtype='float')})

  def recommend_for_single_object(self, K: int, \
              X_matrix_object, cos_flag = True, manh_flag = False):
        """
        Метод формирования рекомендаций для одного объекта.
        Входные параметры:
        K - количество рекомендуемых соседей
        X_matrix_object - строка матрицы объект-признак, соответствующая объекту
        cos_flag - флаг вычисления косинусного расстояния
        manh_flag - флаг вычисления манхэттэнского расстояния
        Возвращаемое значение: K найденных соседей
        """

        scale = 1000000
        # Вычисляем косинусную близость
        if cos_flag:
            dist = cosine_similarity(self._X_matrix, X_matrix_object)
            self.df['dist'] = dist * scale
            res = self.df.sort_values(by='dist', ascending=False)
            # Не учитываем рекомендации с единичным расстоянием,
            # так как это искомый объект
            res = res[res['dist'] < scale]

        else:
            if manh_flag:
                dist = manhattan_distances(self._X_matrix, X_matrix_object)
```

```
        else:
            dist = euclidean_distances(self._X_matrix, X_matrix_object)
        self.df['dist'] = dist * scale
        res = self.df.sort_values(by='dist', ascending=True)
        # Не учитываем рекомендации с единичным расстоянием,
        # так как это искомый объект
        res = res[res['dist'] > 0.0]

    # Оставляем K первых рекомендаций
    res = res.head(K)
    return res
```

```
test_id = 11
print(title[test_id])
print(descriptions[test_id])
```

```
Estate Vineyard Wadensvil Block
From 18-year-old vines, this supple well-balanced effort blends flavors of mocha, cherry
```

```
test_matrix = description_matrix[test_id]
test_matrix
```

```
<1x30748 sparse matrix of type '<class 'numpy.float64'>'
	with 38 stored elements in Compressed Sparse Row format>
```

```
skr1 = SimplerKnnRecomender(description_matrix, wine_ids, title, descriptions)
```

```
# 15 вин, наиболее похожих на Estate Vineyard Wadensvil Block
# в порядке убывания схожести на основе косинусного сходства
rec1 = skr1.recommend_for_single_object(15, test_matrix)
rec1
```

| | id | description | overview | dist |
|---|---|---|---|---|
| **314** | 314 | Durant Vineyard Bishop Block | This gorgeous wine deftly integrates a lively ... | 240817.816706 |
| **98668** | 98668 | Bucher Vineyard | Forward and delicious, a vibrant wine that's f... | 239677.127659 |
| **122788** | 122788 | Bucher Vineyard | Forward and delicious, a vibrant wine that's f... | 239677.127659 |
| **66340** | 66340 | NaN | A soft, herbaceous wine that's destined for ea... | 235219.103194 |

```
# При поиске с помощью Евклидова расстояния получаем такой же результат
rec2 = skr1.recommend_for_single_object(15, test_matrix, cos_flag = False)
rec2
```

| | id | description | overview | dist |
|---|---|---|---|---|
| **314** | 314 | Durant Vineyard Bishop Block | This gorgeous wine deftly integrates a lively ... | 1.232219e+06 |
| **98668** | 98668 | Bucher Vineyard | Forward and delicious, a vibrant wine that's f... | 1.233145e+06 |
| **122788** | 122788 | Bucher Vineyard | Forward and delicious, a vibrant wine that's f... | 1.233145e+06 |
| **149980** | 149980 | NaN | A soft, herbaceous wine that's destined for ea... | 1.236755e+06 |
| **66340** | 66340 | NaN | A soft, herbaceous wine that's destined for ea... | 1.236755e+06 |
| **137724** | 137724 | Viña Pedro Gonzalez | Ripe and smooth, with hints of molasses, toast... | 1.249305e+06 |
| **115644** | 115644 | Viña Pedro Gonzalez | Ripe and smooth, with hints of molasses, toast... | 1.249305e+06 |
| **112936** | 112936 | Halkidiki Vineyards | This simple, supple, modern Merlot blends blac... | 1.251913e+06 |
| **15136** | 15136 | Halkidiki Vineyards | This simple, supple, modern Merlot blends blac... | 1.251913e+06 |
| **6417** | 6417 | NaN | Old vines help give smoothness and concentrati... | 1.253660e+06 |

```
# Манхэттэнское расстояние дает абсолютно иные результаты поиска
rec3 = skr1.recommend_for_single_object(15, test_matrix,
                                        cos_flag = False, manh_flag = True)
rec3
```

| | id | description | overview | dist |
|---|---|---|---|---|
| **128156** | 128156 | NaN | Sweet and fruity. | 7.097305e+06 |
| **116396** | 116396 | NaN | Sweet and fruity. | 7.097305e+06 |
| **144597** | 144597 | NaN | Sulfury, soft and sweet. | 7.111033e+06 |
| **144862** | 144862 | Picnic Hill Vineyard Old Vines | Hot, sweet and Porty. | 7.264079e+06 |
| **144858** | 144858 | NaN | Sweet, overripe and rough. | 7.284878e+06 |
| **103568** | 103568 | NaN | Thin, green and tannic. | 7.338500e+06 |
| **36008** | 36008 | NaN | Thin, green and tannic. | 7.338500e+06 |
| **63757** | 63757 | NaN | Unripe, with feline aromas and flavors. | 7.355518e+06 |
| **130747** | 130747 | NaN | Unripe, with feline aromas and flavors. | 7.355518e+06 |
| **107797** | 107797 | NaN | Unripe, with feline aromas and flavors. | 7.355518e+06 |
| **128458** | 128458 | NaN | Very tannic, rough. | 7.392086e+06 |
| **116483** | 116483 | NaN | Very tannic, rough. | 7.392086e+06 |

## Коллаборативная фильтрация. Метод на основе сингулярного разложения

```
data.head()
```

| | Unnamed: 0 | country | description | designation | points | price | province | region_1 | reg |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | US | This tremendous 100% varietal wine hails from ... | Martha's Vineyard | 96 | 235.0 | California | Napa Valley | |
| | | | Ripe aromas of fig | Carodorum | | | | | |

```
data3 = data[30000:55000]
```

```
# Количество уникальных виноделен
len(data3['winery'].unique())
```

```
7407
```

```
# Количество уникальных вин
len(data3['designation'].unique())
```

```
11196
```

```
# Сформируем матрицу взаимодействий на основе рейтингов
# Используется идея из статьи - https://towardsdatascience.com/beginners-guide-to-creating-an
def create_utility_matrix(data):
    itemField = 'designation'
    userField = 'winery'
    valueField = 'points'

    userList = data[userField].tolist()
    itemList = data[itemField].tolist()
    valueList = data[valueField].tolist()

    users = list(set(userList))
    items = list(set(itemList))

    users_index = {users[i]: i for i in range(len(users))}
    pd_dict = {item: [0.0 for i in range(len(users))] for item in items}

    for i in range(0,data.shape[0]):
        item = itemList[i]
        user = userList[i]
        value = valueList[i]
        pd_dict[item][users_index[user]] = value

    X = pd.DataFrame(pd_dict)
    X.index = users

    itemcols = list(X.columns)
    items_index = {itemcols[i]: i for i in range(len(itemcols))}

    return X, users_index, items_index
```

```
%%time
user_item_matrix, users_index, items_index = create_utility_matrix(data3)
```

```
CPU times: user 11.9 s, sys: 1.16 s, total: 13.1 s
Wall time: 12.6 s
```

```
user_item_matrix
```

| | NaN | Yarrabank Cuvée | Le Franette | Vigne Vecchie della Cappelletta | Avila Beach Sunset Dry Rosé of | Jules Réserve | Second Generation | Satrape |
|---|---|---|---|---|---|---|---|---|
| **Château La Croix-Davids** | 90.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ( |
| **Quadrant** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ( |
| **Château Salitis** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ( |
| **Dark Horse** | 85.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ( |
| **San Giuseppe** | 85.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ( |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **Comtesse Thérèse** | 87.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ( |
| **Freeman** | 88.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ( |

```
# Выделение тестовой строки
user_item_matrix__test = user_item_matrix.loc[['San Giuseppe']]
user_item_matrix__test
```

| | NaN | Yarrabank Cuvée | Le Franette | Vigne Vecchie della Cappelletta | Avila Beach Sunset Dry Rosé of | Jules Réserve | Second Generation | Satrapezo |
|---|---|---|---|---|---|---|---|---|
| **San Giuseppe** | 85.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

```
#taster_names = description_data['taster_name'].unique()
taster_names = np.delete(data3['winery'].unique(), 0)
taster_names = np.delete(taster_names, 7)
taster_names
```

```
array(['Louis Max', 'Luigi Pira', 'Mauro Veglio', ..., 'Rendition',
       'Keermont', 'Lagoon Hill'], dtype=object)
```

```
# Оставшаяся часть матрицы для обучения
user_item_matrix__train = user_item_matrix.loc[taster_names]
user_item_matrix__train
```

| | NaN | Yarrabank Cuvée | Le Franette | Vigne Vecchie della Cappelletta | Avila Beach Sunset Dry Rosé of | Jules Réserve | Second Generation | Satrapez |
|---|---|---|---|---|---|---|---|---|
| **Louis Max** | 86.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0. |
| **Luigi Pira** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0. |
| **Mauro Veglio** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0. |
| **Mounts** | 88.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0. |
| **Oddero** | 90.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0. |
| **...** | ... | ... | ... | ... | ... | ... | ... | . |
| **Château Clos Haut-Peyraguey** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0. |
| **Ronco delle Betulle** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0. |
| **Rendition** | 90.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0. |
| **Keermont** | 90.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0. |

```
%%time
U, S, VT = np.linalg.svd(user_item_matrix__train.T)
V = VT.T
```

```
CPU times: user 9min 35s, sys: 12.6 s, total: 9min 48s
Wall time: 5min 1s
```

```
# Матрица соотношения между дегустаторами и латентными факторами
U.shape
```

```
(11196, 11196)
```

```
# Матрица соотношения между объектами и латентными факторами
V.shape
```

```
(7405, 7405)
```

```
S.shape
```

```
(7405,)
```

```
Sigma = np.diag(S)
Sigma.shape
```

```
(7405, 7405)
```

```
# Диагональная матрица сингулярных значений
Sigma
```

```
array([[5.33232100e+03, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 1.25945475e+03, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 1.18046642e+03, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        1.58212049e-14, 0.00000000e+00, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 1.03628788e-14, 0.00000000e+00],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 8.51763513e-15]])
```

```
# Используем 3 первых сингулярных значения
r=3
Ur = U[:, :r]
Sr = Sigma[:r, :r]
Vr = V[:, :r]
# Матрица соотношения между виноделом и латентными факторами
test_winery = np.mat(user_item_matrix__test.values)
test_winery.shape, test_winery
```

```
((1, 11196), matrix([[85.,  0.,  0., ...,  0.,  0.,  0.]]))
```

```
tmp = test_winery * Ur * np.linalg.inv(Sr)
tmp
```

```
matrix([[-0.01590941, -0.00227796,  0.00225625]])
```

```
test_winery_result = np.array([tmp[0,0], tmp[0,1], tmp[0,2]])
test_winery_result
```

```
array([-0.01590941, -0.00227796,  0.00225625])
```

```
# Вычисляем косинусную близость между текущим виноделом
# и остальными виноделами
cos_sim = cosine_similarity(Vr, test_winery_result.reshape(1, -1))
cos_sim[:10]
```

```
array([[ 9.99564026e-01],
       [-2.79442277e-16],
       [ 3.10425969e-17],
       [ 9.96784967e-01],
       [ 9.99999535e-01],
       [ 1.00000000e+00],
       [ 9.99928466e-01],
       [-4.33543932e-02],
       [ 9.99997669e-01],
       [ 4.28218464e-01]])
```

```
# Преобразуем размерность массива
cos_sim_list = cos_sim.reshape(-1, cos_sim.shape[0])[0]
cos_sim_list[:10]
```

```
array([ 9.99564026e-01, -2.79442277e-16,  3.10425969e-17,  9.96784967e-01,
        9.99999535e-01,  1.00000000e+00,  9.99928466e-01, -4.33543932e-02,
        9.99997669e-01,  4.28218464e-01])
```

```
# Находим наиболее близкого винодела
recommended_winery_id = np.argsort(-cos_sim_list)[0]
recommended_winery_id
```

```
3171
```

```
test_winery
```

```
matrix([[85.,  0.,  0., ...,  0.,  0.,  0.]])
```

```
# Получение названия вина
wine_list = list(user_item_matrix.columns)
def film_name_by_movieid(ind):
    try:
        wine = wine_list[ind]
        #print(wineId)
        #flt_links = data3[data['movieId'] == wineId]
        #tmdbId = int(flt_links['tmdbId'].values[0])
        #md_links = df_md[df_md['id'] == tmdbId]
        #res = md_links['title'].values[0]
        return wine
    except:
        return ''
```

```
# Вина, текущей винодельни:
i=1
for idx, item in enumerate(np.ndarray.flatten(np.array(test_winery))):
    if item > 0:
        wine_title = film_name_by_movieid(idx)
        print('{} - {} - {}'.format(idx, wine_title, item))
```

```
        if i==20:
            break
        else:
            i+=1
```

```
 0 - nan - 85.0
```

```
# Вина, наиболее схожие с винодельней:
i=1
recommended_user_item_matrix = user_item_matrix.loc[['Oddero']]
for idx, item in enumerate(np.ndarray.flatten(np.array(recommended_user_item_matrix))):
    if item > 0:
        wine_title = film_name_by_movieid(idx)
        print('{} - {} - {}'.format(idx, wine_title, item))
        if i==20:
            break
        else:
            i+=1
```

```
 0 - nan - 90.0
 5224 - Vinchio - 88.0
```

Как видно, фильтрация на основе содержания и коллаборативная фильтрация показывают различные результаты работы в рамках рекомендательных систем