Рубежный контроль №2

Тема: Методы построения моделей машинного обучения

Ширшов А.С. ИУ5-65Б Вариант-19

Загрузка необходимых библиотек:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn import preprocessing
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, 
classification_report

from sklearn.ensemble import AdaBoostClassifier
```

Считываем датасет и делаем первичный анализ данных

```
data = pd.read_csv('./investments_VC.csv', encoding='latin1', sep=",")
target_col = 'status'

data.head()
```

```
                         permalink                 name  \
0           /organization/waywire             #waywire   
1  /organization/tv-communications  &TV Communications   
2    /organization/rock-your-paper     'Rock' Your Paper   
3   /organization/in-touch-network    (In)Touch Network   
4   /organization/r-ranch-and-mine    -R- Ranch and Mine   

                 homepage_url  \
0        http://www.waywire.com   
1         http://enjoyandtv.com   
2   http://www.rockyourpaper.org   
3  http://www.InTouchNetwork.com   
4                          NaN   

                                       category_list        market   \
0         |Entertainment|Politics|Social Media|News|          News   
1                                           |Games|         Games   
2                            |Publishing|Education|    Publishing   
3  |Electronics|Guides|Coffee|Restaurants|Music|i...   Electronics   
```

```
4                      |Tourism|Entertainment|Games|       Tourism

   funding_total_usd      status country_code state_code
region  ...  \
0          17,50,000    acquired          USA         NY  New York
City  ...
1          40,00,000   operating          USA         CA    Los
Angeles  ...
2             40,000   operating          EST        NaN
Tallinn  ...
3          15,00,000   operating          GBR        NaN
London  ...
4             60,000   operating          USA         TX
Dallas  ...

  secondary_market  product_crowdfunding round_A round_B round_C
round_D  \
0              0.0                   0.0     0.0     0.0     0.0
0.0
1              0.0                   0.0     0.0     0.0     0.0
0.0
2              0.0                   0.0     0.0     0.0     0.0
0.0
3              0.0                   0.0     0.0     0.0     0.0
0.0
4              0.0                   0.0     0.0     0.0     0.0
0.0

  round_E round_F  round_G  round_H
0     0.0     0.0      0.0      0.0
1     0.0     0.0      0.0      0.0
2     0.0     0.0      0.0      0.0
3     0.0     0.0      0.0      0.0
4     0.0     0.0      0.0      0.0

[5 rows x 39 columns]
```

```
data.shape
```

```
(54294, 39)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54294 entries, 0 to 54293
Data columns (total 39 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   permalink             49438 non-null  object
 1   name                  49437 non-null  object
 2   homepage_url          45989 non-null  object
```

```
 3   category_list         45477 non-null  object
 4    market               45470 non-null  object
 5    funding_total_usd    49438 non-null  object
 6   status                48124 non-null  object
 7   country_code          44165 non-null  object
 8   state_code            30161 non-null  object
 9   region                44165 non-null  object
 10  city                  43322 non-null  object
 11  funding_rounds        49438 non-null  float64
 12  founded_at            38554 non-null  object
 13  founded_month         38482 non-null  object
 14  founded_quarter       38482 non-null  object
 15  founded_year          38482 non-null  float64
 16  first_funding_at      49438 non-null  object
 17  last_funding_at       49438 non-null  object
 18  seed                  49438 non-null  float64
 19  venture               49438 non-null  float64
 20  equity_crowdfunding   49438 non-null  float64
 21  undisclosed           49438 non-null  float64
 22  convertible_note      49438 non-null  float64
 23  debt_financing        49438 non-null  float64
 24  angel                 49438 non-null  float64
 25  grant                 49438 non-null  float64
 26  private_equity        49438 non-null  float64
 27  post_ipo_equity       49438 non-null  float64
 28  post_ipo_debt         49438 non-null  float64
 29  secondary_market      49438 non-null  float64
 30  product_crowdfunding  49438 non-null  float64
 31  round_A               49438 non-null  float64
 32  round_B               49438 non-null  float64
 33  round_C               49438 non-null  float64
 34  round_D               49438 non-null  float64
 35  round_E               49438 non-null  float64
 36  round_F               49438 non-null  float64
 37  round_G               49438 non-null  float64
 38  round_H               49438 non-null  float64
dtypes: float64(23), object(16)
memory usage: 16.2+ MB
```

Очистка данных

Проверим датасет на пустые значения, уберём данные не влияющие на целевой признак, очистим данные от лишних символов.

```
data.isnull().mean()
```

```
permalink               0.089439
name                    0.089457
homepage_url            0.152963
category_list           0.162394
 market                 0.162523
```

```
 funding_total_usd      0.089439
status                 0.113641
country_code           0.186558
state_code             0.444487
region                 0.186558
city                   0.202085
funding_rounds         0.089439
founded_at             0.289903
founded_month          0.291229
founded_quarter        0.291229
founded_year           0.291229
first_funding_at       0.089439
last_funding_at        0.089439
seed                   0.089439
venture                0.089439
equity_crowdfunding    0.089439
undisclosed            0.089439
convertible_note       0.089439
debt_financing         0.089439
angel                  0.089439
grant                  0.089439
private_equity         0.089439
post_ipo_equity        0.089439
post_ipo_debt          0.089439
secondary_market       0.089439
product_crowdfunding   0.089439
round_A                0.089439
round_B                0.089439
round_C                0.089439
round_D                0.089439
round_E                0.089439
round_F                0.089439
round_G                0.089439
round_H                0.089439
dtype: float64
```

```
data=data.drop(['permalink','category_list','founded_at',
'founded_month',
       'founded_quarter',
       'first_funding_at', 'last_funding_at'],axis=1)
```

Смотрим на количество нулевых значений. По результату ниже видно, что часто повторяется число 4856, это как окажется пустые строки в нашем наборе данных. Их надо убрать.

```
data.isnull().sum()
```

```
name                    4857
homepage_url            8305
 market                 8824
 funding_total_usd      4856
```

```
status                    6170
country_code             10129
state_code               24133
region                   10129
city                     10972
funding_rounds            4856
founded_year             15812
seed                      4856
venture                   4856
equity_crowdfunding       4856
undisclosed               4856
convertible_note          4856
debt_financing            4856
angel                     4856
grant                     4856
private_equity            4856
post_ipo_equity           4856
post_ipo_debt             4856
secondary_market          4856
product_crowdfunding      4856
round_A                   4856
round_B                   4856
round_C                   4856
round_D                   4856
round_E                   4856
round_F                   4856
round_G                   4856
round_H                   4856
dtype: int64
```

```
data=data.dropna(how="all")
```

SimpleImputer

При помощи SimpleImputer вставим пропущенные данные

```
imp = SimpleImputer(missing_values=np.nan, strategy='most_frequent')

imputed = {}

for col in data:
    contains_nan = data[col].isnull().sum() != 0
    if contains_nan:
        data_imp = data[[col]]
        data_imp = imp.fit_transform(data_imp)
        imputed[col] = data_imp

for col_name in imputed:
    df = pd.DataFrame({col_name:imputed[col_name].T[0]})
    data[col_name] = df.copy()
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 49438 entries, 0 to 49437
Data columns (total 32 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   name                   49438 non-null  object 
 1   homepage_url           49438 non-null  object 
 2    market                49438 non-null  object 
 3    funding_total_usd     49438 non-null  object 
 4   status                 49438 non-null  object 
 5   country_code           49438 non-null  object 
 6   state_code             49438 non-null  object 
 7   region                 49438 non-null  object 
 8   city                   49438 non-null  object 
 9   funding_rounds         49438 non-null  float64
 10  founded_year           49438 non-null  float64
 11  seed                   49438 non-null  float64
 12  venture                49438 non-null  float64
 13  equity_crowdfunding    49438 non-null  float64
 14  undisclosed            49438 non-null  float64
 15  convertible_note       49438 non-null  float64
 16  debt_financing         49438 non-null  float64
 17  angel                  49438 non-null  float64
 18  grant                  49438 non-null  float64
 19  private_equity         49438 non-null  float64
 20  post_ipo_equity        49438 non-null  float64
 21  post_ipo_debt          49438 non-null  float64
 22  secondary_market       49438 non-null  float64
 23  product_crowdfunding   49438 non-null  float64
 24  round_A                49438 non-null  float64
 25  round_B                49438 non-null  float64
 26  round_C                49438 non-null  float64
 27  round_D                49438 non-null  float64
 28  round_E                49438 non-null  float64
 29  round_F                49438 non-null  float64
 30  round_G                49438 non-null  float64
 31  round_H                49438 non-null  float64
dtypes: float64(23), object(9)
memory usage: 12.4+ MB
```

Уберём лишние пробелы, также обработаем столбец funding_total_usd так как там встречаются значения вида 11,11,231

```
data.columns=data.columns.str.strip()

data['funding_total_usd']=data['funding_total_usd'].str.replace(",",""
)
```

```
data["funding_total_usd"]=pd.to_numeric(data["funding_total_usd"],erro
rs="coerce").convert_dtypes()
funding_mode=data['funding_total_usd'].mode()[0]
data['funding_total_usd']=data["funding_total_usd"].fillna(funding_mod
e)
```

```
data.head()
```

```
                 name                     homepage_url         market  \
0            #waywire         http://www.waywire.com          News
1  &TV Communications          http://enjoyandtv.com         Games
2   'Rock' Your Paper   http://www.rockyourpaper.org    Publishing
3   (In)Touch Network  http://www.InTouchNetwork.com   Electronics
4  -R- Ranch and Mine           http://app.thotz.co/       Tourism

   funding_total_usd     status country_code state_code         region
\
0            1750000   acquired          USA         NY  New York City

1            4000000  operating          USA         CA    Los Angeles

2              40000  operating          EST         CA        Tallinn

3            1500000  operating          GBR         CA         London

4              60000  operating          USA         TX         Dallas


          city  funding_rounds  ...  secondary_market
product_crowdfunding  \
0     New York             1.0  ...               0.0
0.0
1  Los Angeles             2.0  ...               0.0
0.0
2      Tallinn             1.0  ...               0.0
0.0
3       London             1.0  ...               0.0
0.0
4   Fort Worth             2.0  ...               0.0
0.0

   round_A  round_B  round_C  round_D  round_E  round_F  round_G
round_H
0      0.0      0.0      0.0      0.0      0.0      0.0      0.0
0.0
1      0.0      0.0      0.0      0.0      0.0      0.0      0.0
0.0
2      0.0      0.0      0.0      0.0      0.0      0.0      0.0
0.0
3      0.0      0.0      0.0      0.0      0.0      0.0      0.0
```

```
0.0
4      0.0      0.0      0.0      0.0      0.0      0.0      0.0
0.0

[5 rows x 32 columns]
```

```
data.shape
```

```
(49438, 32)
```

```
data.dtypes
```

```
name                       object
homepage_url               object
market                     object
funding_total_usd           Int64
status                     object
country_code               object
state_code                 object
region                     object
city                       object
funding_rounds            float64
founded_year              float64
seed                      float64
venture                   float64
equity_crowdfunding       float64
undisclosed               float64
convertible_note          float64
debt_financing            float64
angel                     float64
grant                     float64
private_equity            float64
post_ipo_equity           float64
post_ipo_debt             float64
secondary_market          float64
product_crowdfunding      float64
round_A                   float64
round_B                   float64
round_C                   float64
round_D                   float64
round_E                   float64
round_F                   float64
round_G                   float64
round_H                   float64
dtype: object
```

```
data.select_dtypes('O').describe()
```

LabelEncoding

Закодируем строковые признаки при помощи LabelEncoder

```
encoder_name=LabelEncoder()
encoder_market=LabelEncoder()
encoder_country_code=LabelEncoder()
encoder_homepage_url=LabelEncoder()
encoder_status=LabelEncoder()
encoder_state_code=LabelEncoder()
encoder_city=LabelEncoder()
encoder_region=LabelEncoder()

data['name']=encoder_name.fit_transform(data['name'])
data['market']=encoder_market.fit_transform(data['market'])
data['country_code']=encoder_country_code.fit_transform(data['country_
code'])
data['status']=encoder_homepage_url.fit_transform(data['status'])
data['homepage_url']=encoder_homepage_url.fit_transform(data['homepage
_url'])
data['state_code']=encoder_state_code.fit_transform(data['state_code']
)
data['city']=encoder_city.fit_transform(data['city'])
data['region']=encoder_region.fit_transform(data['region'])

data.head()
```

```
   name  homepage_url  market  funding_total_usd  status  country_code
\
0     0         43610     465            1750000       0           110

1     1          4422     277            4000000       2           110

2     2         37197     543              40000       2            35

3     3         15435     211            1500000       2            38

4     4          1124     683              60000       2           110


   state_code  region  city  funding_rounds  ...  secondary_market  \
0          40     699  2547             1.0  ...               0.0
1           6     570  2098             2.0  ...               0.0
2           6     956  3645             1.0  ...               0.0
3           6     568  2085             1.0  ...               0.0
4          53     251  1234             2.0  ...               0.0

   product_crowdfunding  round_A  round_B  round_C  round_D  round_E
round_F  \
0                   0.0      0.0      0.0      0.0      0.0      0.0
0.0
1                   0.0      0.0      0.0      0.0      0.0      0.0
0.0
2                   0.0      0.0      0.0      0.0      0.0      0.0
```

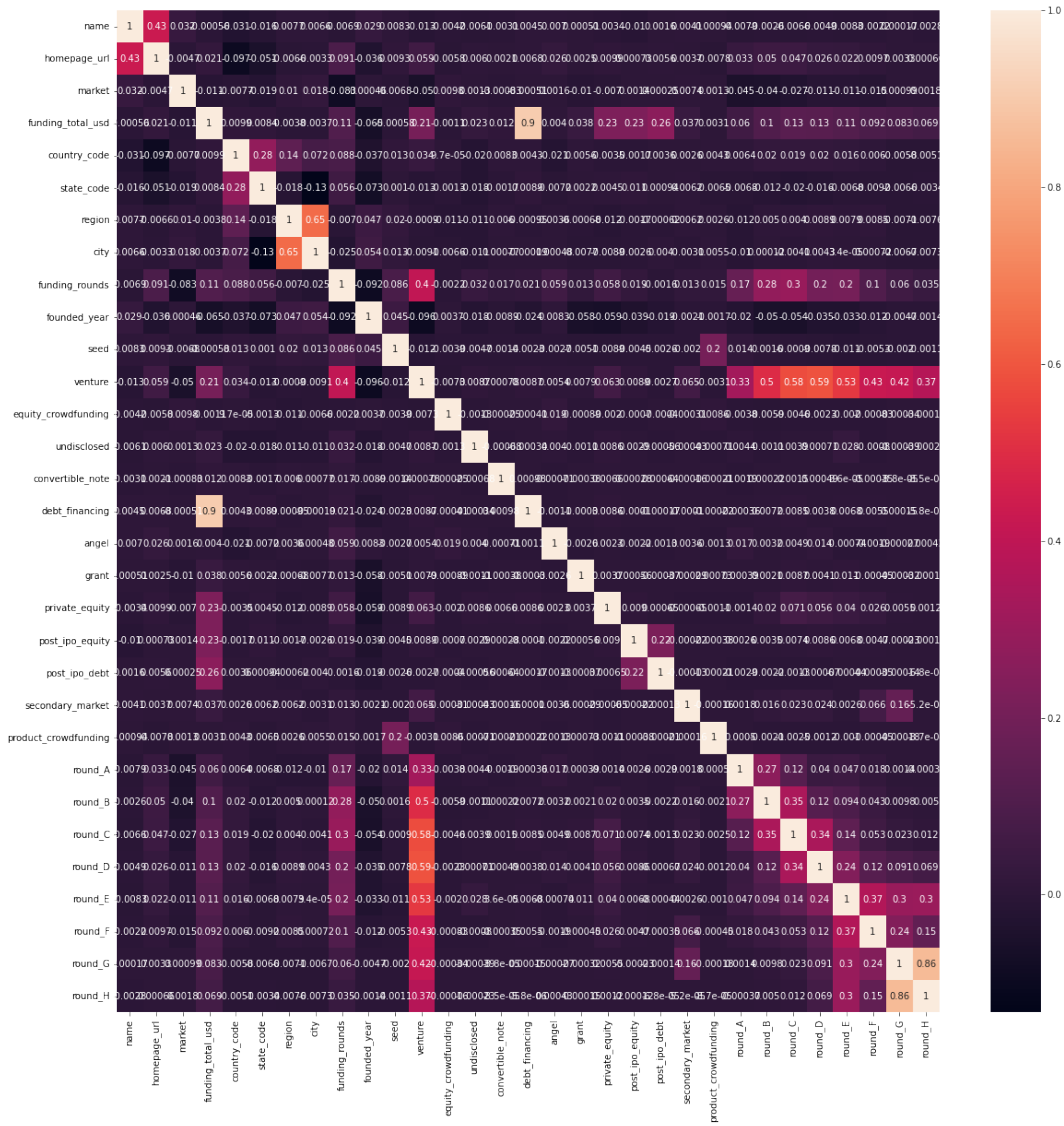
```
0.0
3                    0.0      0.0      0.0      0.0      0.0      0.0
0.0
4                    0.0      0.0      0.0      0.0      0.0      0.0
0.0

   round_G  round_H
0      0.0      0.0
1      0.0      0.0
2      0.0      0.0
3      0.0      0.0
4      0.0      0.0

[5 rows x 32 columns]
```

Построим тепловую карту корреляций и отбросим те признаки, которые линейно корреллируют между собой.

```
data_corr=data.drop("status",axis=1).corr()
plt.figure(figsize=(20,20))
sns.heatmap(data_corr,annot=True)
plt.show()
```



```python
threshold=0.5
def correlation_fun(ds,threshold):
    corr_col=set()
    corr_mat=ds.corr()
    for i in range(len(corr_mat.columns)):
        for j in range(i):
            if abs(corr_mat.iloc[i,j])>threshold:
                colname=corr_mat.columns[i]
                corr_col.add(colname)
    return corr_col

correlation_fun(data.drop("status",axis=1),threshold)
```

```
{'city', 'debt_financing', 'round_C', 'round_D', 'round_E', 'round_H'}
```

```
data=data.drop(['city', 'round_C', 'round_D', 'round_E',
'round_H'],axis=1)
```

Делим выборку на тестовую и обучающую

```
target = data[target_col]
data_X_train, data_X_test, data_y_train, data_y_test =
train_test_split(data, target, test_size=0.2, random_state=1)
```

```
data_X_train.shape
```

```
(39550, 27)
```

```
data_X_test.shape
```

```
(9888, 27)
```

```
np.unique(target)
```

```
array([0, 1, 2])
```

Метод опорных векторов

```
svr_1 = svm.LinearSVC()
svr_1.fit(data_X_train, data_y_train)
```

```
/home/artyom/.local/lib/python3.8/site-packages/sklearn/svm/
_base.py:1206: ConvergenceWarning: Liblinear failed to converge,
increase the number of iterations.
  warnings.warn(
```

```
LinearSVC()
```

```
data_y_pred_1 = svr_1.predict(data_X_test)
accuracy_score(data_y_test, data_y_pred_1)
```

```
0.8080501618122977
```

```
f1_score(data_y_test, data_y_pred_1, average='micro')
```

```
0.8080501618122977
```

```
f1_score(data_y_test, data_y_pred_1, average='macro')
```

```
0.3295056626437861
```

```
f1_score(data_y_test, data_y_pred_1, average='weighted')
```

```
0.7868994820528529
```

```
svr_2 = svm.LinearSVC(C=1.0, max_iter=10000)
svr_2.fit(data_X_train, data_y_train)
```

```
/home/artyom/.local/lib/python3.8/site-packages/sklearn/svm/
_base.py:1206: ConvergenceWarning: Liblinear failed to converge,
```

```
increase the number of iterations.
  warnings.warn(
```

```
LinearSVC(max_iter=10000)
```

```
data_y_pred_2 = svr_2.predict(data_X_test)
accuracy_score(data_y_test, data_y_pred_2)
```

```
0.6844660194174758
```

```
f1_score(data_y_test, data_y_pred_2, average='micro')
```

```
0.6844660194174758
```

```
f1_score(data_y_test, data_y_pred_2, average='macro')
```

```
0.35340656910368473
```

```
f1_score(data_y_test, data_y_pred_2, average='weighted')
```

```
0.7253240978043891
```

```
svr_3 = svm.LinearSVC(C=1.0, penalty='l1', dual=False, max_iter=10000)
svr_3.fit(data_X_train, data_y_train)
```

```
/home/artyom/.local/lib/python3.8/site-packages/sklearn/svm/
_base.py:1206: ConvergenceWarning: Liblinear failed to converge,
increase the number of iterations.
  warnings.warn(
```

```
LinearSVC(dual=False, max_iter=10000, penalty='l1')
```

```
data_y_pred_3_0 = svr_3.predict(data_X_train)
accuracy_score(data_y_train, data_y_pred_3_0)
```

```
0.9997218710493047
```

```
data_y_pred_3 = svr_3.predict(data_X_test)
accuracy_score(data_y_test, data_y_pred_3)
```

```
0.9998988673139159
```

```
f1_score(data_y_test, data_y_pred_3, average='micro')
```

```
0.9998988673139159
```

```
f1_score(data_y_test, data_y_pred_3, average='weighted')
```

```
0.9998988213894379
```

Градиентный бустинг

```
ab1 = AdaBoostClassifier()
ab1.fit(data_X_train, data_y_train)
data_y_pred_1 = ab1.predict(data_X_test)
```

```
data_y_pred_1_0 = ab1.predict(data_X_train)
accuracy_score(data_y_train, data_y_pred_1_0)
```

```
1.0
```

```
accuracy_score(data_y_test, data_y_pred_1)
```

```
1.0
```

```
f1_score(data_y_test, data_y_pred_1, average='micro')
```

```
1.0
```

```
f1_score(data_y_test, data_y_pred_1, average='macro')
```

```
1.0
```

```
f1_score(data_y_test, data_y_pred_1, average='weighted')
```

```
1.0
```

Выводы

При использовании обоих методов удалось получить практический 100% процентную точность. При использовании метода опорных векторов наилучший результат показала модель с параметрами: svr_3 = svm.LinearSVC(C=1.0, penalty='l1', dual=False, max_iter=10000)

При использовании градиентного бустинга модель показывает 100% результат. На практике такой результат очень маловероятен, в данной работе он вероятно связан с тем, что были опущены некоторые параметры, которые по моему мнению не могли влиять итоговый результат. Это и показывала корреляционная таблица, приведённая выше.

При наших результатах, можно использовать обе модели, однако градиентный бустинг будет справляться несколько точнее.