ИУ5-61Б Плотников Ф.С.

Вариант 19

Тема: Методы построения моделей машинного обучения

Задание

Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

При решении задач можно выбирать любое подмножество признаков из приведенного набора данных.

Для сокращения времени построения моделей можно использовать фрагмент набора данных (например, первые 200-500 строк).

Методы 1 и 2 для ИУ5-61Б: Линейная/логистическая регрессия, Случайный лес

Набор данных: https://www.kaggle.com/datasets/arindam235/startup-investments-crunchbase

In [1]:

```
# Загрузка необходимых библиотек
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn import preprocessing
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
#from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_repor
t
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_err
or, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import AdaBoostClassifier
from sklearn.impute import SimpleImputer, MissingIndicator
```

In [2]:

```
data = pd.read_csv('investments_VC.csv', encoding = "iso-8859-1")
TARGET_COL_NAME = 'status'
```

```
TARGET_IS_NUMERIC = data[TARGET_COL_NAME].dtype != 'O'
TARGET_IS_NUMERIC
```

Out[2]:

```
False
```

In [3]:

```
data = data.head(1009)
data
```

Out[3]:

	permalink	name	homepage_url	category_li
0	/organization/waywire	#waywire	http://www.waywire.com	|Entertainment|Politics|Social Media|New
1	/organization/tv-communications	&TV Communications	http://enjoyandtv.com	|Game
2	/organization/rock-your-paper	'Rock' Your Paper	http://www.rockyourpaper.org	|Publishing|Educatio
3	/organization/in-touch-network	(In)Touch Network	http://www.InTouchNetwork.com	|Electronics|Guides|Coffee|Restaurants|Musicli
4	/organization/r-ranch-and-mine	-R- Ranch and Mine	NaN	|Tourism|Entertainment|Game
...	...	...	...	
1004	/organization/addfleet	AddFleet	http://www.addfleet.com	|Transportation|Mobility|Mobil
1005	/organization/addiction-campuses-of-america	Addiction Campuses of America	http://addictioncampus.com/	|Health Car
1006	/organization/addictive	Addictive	http://www.pitchtarget.com	Na
1007	/organization/arkli	AddIn Social	http://www.addinsocial.com	|Email Marketing|Sales and Marketing|Internet
1008	/organization/additech	Additech	http://www.additech.com	|Automotiv

1009 rows × 39 columns

In [4]:

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

Out[4]:

```
permalink                0
name                     0
homepage_url            72
category_list           78
 market                 78
 funding_total_usd       0
status                  19
country_code           105
state_code             417
region                 105
city                   129
funding_rounds           0
founded_at             214
founded_month          214
founded_quarter        214
founded_year           214
first_funding_at         0
last_funding_at          0
seed                     0
venture                  0
equity_crowdfunding      0
undisclosed              0
```

```
convertible_note          0
debt_financing            0
angel                     0
grant                     0
private_equity            0
post_ipo_equity           0
post_ipo_debt             0
secondary_market          0
product_crowdfunding      0
round_A                   0
round_B                   0
round_C                   0
round_D                   0
round_E                   0
round_F                   0
round_G                   0
round_H                   0
dtype: int64
```

In [5]:

```
data.status = data.status.dropna(axis=0)
```

In [6]:

```
data
```

Out[6]:

	permalink	name	homepage_url	category_li
0	/organization/waywire	#waywire	http://www.waywire.com	|Entertainment|Politics|Social Media|New
1	/organization/tv-communications	&TV Communications	http://enjoyandtv.com	|Game
2	/organization/rock-your-paper	'Rock' Your Paper	http://www.rockyourpaper.org	|Publishing|Educatio
3	/organization/in-touch-network	(In)Touch Network	http://www.InTouchNetwork.com	|Electronics|Guides|Coffee|Restaurants|Music|i
4	/organization/r-ranch-and-mine	-R- Ranch and Mine	NaN	|Tourism|Entertainment|Game
...	...	...	...	
1004	/organization/addfleet	AddFleet	http://www.addfleet.com	|Transportation|Mobility|Mobil
1005	/organization/addiction-campuses-of-america	Addiction Campuses of America	http://addictioncampus.com/	|Health Car
1006	/organization/addictive	Addictive	http://www.pitchtarget.com	Na
1007	/organization/arkli	AddIn Social	http://www.addinsocial.com	|Email Marketing|Sales and Marketing|Internet
1008	/organization/additech	Additech	http://www.additech.com	|Automotiv

1009 rows × 39 columns

Удалим колонки, которые не влияют на целевой признак **status**

In [7]:

```
data.drop(columns=['permalink'], inplace=True)
data.drop(columns=['homepage_url'], inplace=True)
data.drop(columns=['name'], inplace=True)
data.shape
```

Out[7]:

```
(1009, 36)
```

Закодируем категориальные признаки

In [8]:

```
not_number_cols = data.select_dtypes(include=['object'])
number_cols = data.select_dtypes(exclude=['object'])
```

In [9]:

```
le = preprocessing.LabelEncoder()

for col_name in not_number_cols:
    data[col_name] = le.fit_transform(data[col_name])

data
```

Out[9]:

	category_list	market	funding_total_usd	status	country_code	state_code	region	city	funding_rounds	founded_at	...
0	122	119	161	0	44	30	141	243	1.0	196	...
1	145	71	356	2	44	5	112	203	2.0	268	...
2	275	135	357	2	16	46	204	356	1.0	206	...
3	108	56	136	2	19	46	110	200	1.0	159	...
4	369	166	444	2	44	40	53	120	2.0	249	...
...	...	...	...	...	...	...	...	...	...	...	...
1004	374	107	364	1	15	46	19	21	2.0	179	...
1005	154	74	325	2	44	39	136	45	2.0	268	...
1006	410	183	0	2	46	46	228	409	1.0	247	...
1007	110	152	417	1	6	33	150	256	2.0	147	...
1008	27	13	13	2	44	40	85	150	3.0	20	...

1009 rows × 36 columns

Обработка пропусков

In [10]:

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

Out[10]:

```
category_list             0
 market                   0
 funding_total_usd        0
status                    0
country_code              0
state_code                0
region                    0
city                      0
funding_rounds            0
founded_at                0
founded_month             0
founded_quarter           0
founded_year            214
first_funding_at          0
last_funding_at           0
seed                      0
venture                   0
```

```
equity_crowdfunding       0
undisclosed               0
convertible_note          0
debt_financing            0
angel                     0
grant                     0
private_equity            0
post_ipo_equity           0
post_ipo_debt             0
secondary_market          0
product_crowdfunding      0
round_A                   0
round_B                   0
round_C                   0
round_D                   0
round_E                   0
round_F                   0
round_G                   0
round_H                   0
dtype: int64
```

In [11]:

```
def test_num_impute_col(dataset, column, strategy_param):
    temp_data = dataset[[column]]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(temp_data)

    filled_data = data_num_imp[mask_missing_values_only]

    return column, strategy_param, filled_data.size, filled_data[0], filled_data[filled_
data.size-1]

strategies=['mean', 'median', 'most_frequent']
data.founded_year.fillna(test_num_impute_col(data, 'founded_year', strategies[1])[3], in
place=True)
```

In [12]:

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

Out[12]:

```
category_list           0
 market                 0
 funding_total_usd      0
status                  0
country_code            0
state_code              0
region                  0
city                    0
funding_rounds          0
founded_at              0
founded_month           0
founded_quarter         0
founded_year            0
first_funding_at        0
last_funding_at         0
seed                    0
venture                 0
equity_crowdfunding     0
undisclosed             0
convertible_note        0
debt_financing          0
angel                   0
grant                   0
private_equity          0
post_ipo_equity         0
```

```
post_ipo_debt            0
secondary_market         0
product_crowdfunding     0
round_A                  0
round_B                  0
round_C                  0
round_D                  0
round_E                  0
round_F                  0
round_G                  0
round_H                  0
dtype: int64
```

In [13]:

```
# Я возьму первые 9 столбцов
data = data.iloc[:, 0:9]
```

In [14]:

```
data
```

Out[14]:

	category_list	market	funding_total_usd	status	country_code	state_code	region	city	funding_rounds
0	122	119	161	0	44	30	141	243	1.0
1	145	71	356	2	44	5	112	203	2.0
2	275	135	357	2	16	46	204	356	1.0
3	108	56	136	2	19	46	110	200	1.0
4	369	166	444	2	44	40	53	120	2.0
...	...	...	...	...	...	...	...	...	...
1004	374	107	364	1	15	46	19	21	2.0
1005	154	74	325	2	44	39	136	45	2.0
1006	410	183	0	2	46	46	228	409	1.0
1007	110	152	417	1	6	33	150	256	2.0
1008	27	13	13	2	44	40	85	150	3.0

1009 rows × 9 columns

Делим выборку на обучающую и тренировочную

In [15]:

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

In [16]:

```
data_X_train.shape, data_y_train.shape
```

Out[16]:

```
((807, 9), (807,))
```

In [17]:

```
data_X_test.shape, data_y_test.shape
```

Out[17]:

```
((202, 9), (202,))
```

In [18]:

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

Out[18]:

```
array([0, 1, 2, 3])
```

Логистическая регрессия

In [19]:

```
svr_1 = LogisticRegression(solver='lbfgs', max_iter=1000)
svr_1.fit(data_X_train, data_y_train)
```

```
C:\Users\user\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:444: Converge
nceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[19]:

```
▾ LogisticRegression
LogisticRegression(max_iter=1000)
```

In [20]:

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

Out[20]:

```
0.9653465346534653
```

In [21]:

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

Out[21]:

```
0.9653465346534653
```

In [22]:

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

Out[22]:

```
0.7258117236870778
```

In [23]:

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

Out[23]:

```
0.9506482354489535
```

In [24]:

```
svr_2 = LogisticRegression(solver='lbfgs', max_iter=10000)
svr_2.fit(data_X_train, data_y_train)
```

Out[24]:

```
▾ LogisticRegression
```

```
LogisticRegression(max_iter=10000)
```

In [25]:

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

Out[25]:

```
0.9801980198019802
```

In [26]:

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

Out[26]:

```
0.9801980198019802
```

In [27]:

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

Out[27]:

```
0.8721428571428571
```

In [28]:

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

Out[28]:

```
0.9753606789250353
```

Случайный лес

In [29]:

```
RT = RandomForestClassifier(n_estimators=15, random_state=123)
RT.fit(data_X_train, data_y_train)
```

Out[29]:

```
▾ RandomForestClassifier
RandomForestClassifier(n_estimators=15, random_state=123)
```

In [30]:

```
accuracy_score(data_y_test, RT.predict(data_X_test))
```

Out[30]:

```
0.9900990099009901
```

In [31]:

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

Out[31]:

```
0.9653465346534653
```

In [32]:

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

Out[32]:

```
0.7258117236870778
```

In [33]:

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

Out[33]:

```
0.9506482354489535
```

In [34]:

```
RT = RandomForestClassifier(n_estimators=30, random_state=123)
RT.fit(data_X_train, data_y_train)
```

Out[34]:

```
RandomForestClassifier(n_estimators=30, random_state=123)
```

In [35]:

```
accuracy_score(data_y_test, RT.predict(data_X_test))
```

Out[35]:

```
0.9851485148514851
```

In [36]:

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

Out[36]:

```
0.9653465346534653
```

In [37]:

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

Out[37]:

```
0.7258117236870778
```

In [38]:

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

Out[38]:

```
0.9506482354489535
```

Выводы

При использовании логистической регрессии наилучшую точность **(0.980)** показала модель с параметром **max_iter=10000.** При использовании метода "Случайный лес" получилось добиться более высокого показателя точности **(0.990),** хотя и разница незначительная**,** но предпочтительней использовать второй метод**.**