Рубежный контроль №2

Тема: Методы построения моделей машинного обучения

Стрихар П.А. ИУ5-61Б

Загрузка необходимых библиотек:

In [1]:

```
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
from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, cla
from sklearn.metrics import confusion_matrix
from sklearn.metrics import mean_absolute_error, mean_squared_error, mea
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.linear_model import LogisticRegression, LogisticRegressionC
from sklearn.ensemble import RandomForestClassifier

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

In [2]:

```
data = pd.read_csv('./HRDataset_v14.csv', sep=",")
TARGET_COL_NAME = 'RecruitmentSource'
TARGET_IS_NUMERIC = data[TARGET_COL_NAME].dtype != 'O'
TARGET_IS_NUMERIC
```

Out[2]:

```
False
```

In [3]:

```
data
```

Out[3]:

	Employee_Name	EmpID	MarriedID	MaritalStatusID	GenderID	EmpStatusID	DeptID
0	Adinolfi, Wilson K	10026	0	0	1	1	5
1	Ait Sidi, Karthikeyan	10084	1	1	1	5	3
2	Akinkuolie, Sarah	10196	1	1	0	5	5
3	Alagbe,Trina	10088	1	1	0	1	5
4	Anderson, Carol	10069	0	2	0	5	5
...	...	...	...	...	...	...	...
306	Woodson, Jason	10135	0	0	1	1	5

307	Ybarra, Catherine	10301	0	0	0	5	5
308	Zamora, Jennifer	10010	0	0	0	1	3
309	Zhou, Julia	10043	0	0	0	1	3
310	Zima, Colleen	10271	0	4	0	1	5

311 rows × 36 columns

In [4]:

```
data.shape
```

Out[4]:

```
(311, 36)
```

In [5]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 311 entries, 0 to 310
Data columns (total 36 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Employee_Name               311 non-null    object 
 1   EmpID                       311 non-null    int64  
 2   MarriedID                   311 non-null    int64  
 3   MaritalStatusID             311 non-null    int64  
 4   GenderID                    311 non-null    int64  
 5   EmpStatusID                 311 non-null    int64  
 6   DeptID                      311 non-null    int64  
 7   PerfScoreID                 311 non-null    int64  
 8   FromDiversityJobFairID      311 non-null    int64  
 9   Salary                      311 non-null    int64  
 10  Termd                       311 non-null    int64  
 11  PositionID                  311 non-null    int64  
 12  Position                    311 non-null    object 
 13  State                       311 non-null    object 
 14  Zip                         311 non-null    int64  
 15  DOB                         311 non-null    object 
 16  Sex                         311 non-null    object 
 17  MaritalDesc                 311 non-null    object 
 18  CitizenDesc                 311 non-null    object 
 19  HispanicLatino              311 non-null    object 
 20  RaceDesc                    311 non-null    object 
 21  DateofHire                  311 non-null    object 
 22  DateofTermination           104 non-null    object 
 23  TermReason                  311 non-null    object 
 24  EmploymentStatus            311 non-null    object 
 25  Department                  311 non-null    object 
 26  ManagerName                 311 non-null    object 
 27  ManagerID                   303 non-null    float64
 28  RecruitmentSource           311 non-null    object 
 29  PerformanceScore            311 non-null    object 
 30  EngagementSurvey            311 non-null    float64
 31  EmpSatisfaction             311 non-null    int64  
 32  SpecialProjectsCount        311 non-null    int64  
 33  LastPerformanceReview_Date  311 non-null    object 
 34  DaysLateLast30              311 non-null    int64  
 35  Absences                    311 non-null    int64  
dtypes: float64(2), int64(16), object(18)
memory usage: 87.6+ KB
```

In [6]:

```
# проверим есть ли пропущенные значения
data.isnull().sum()
```

Out[6]:

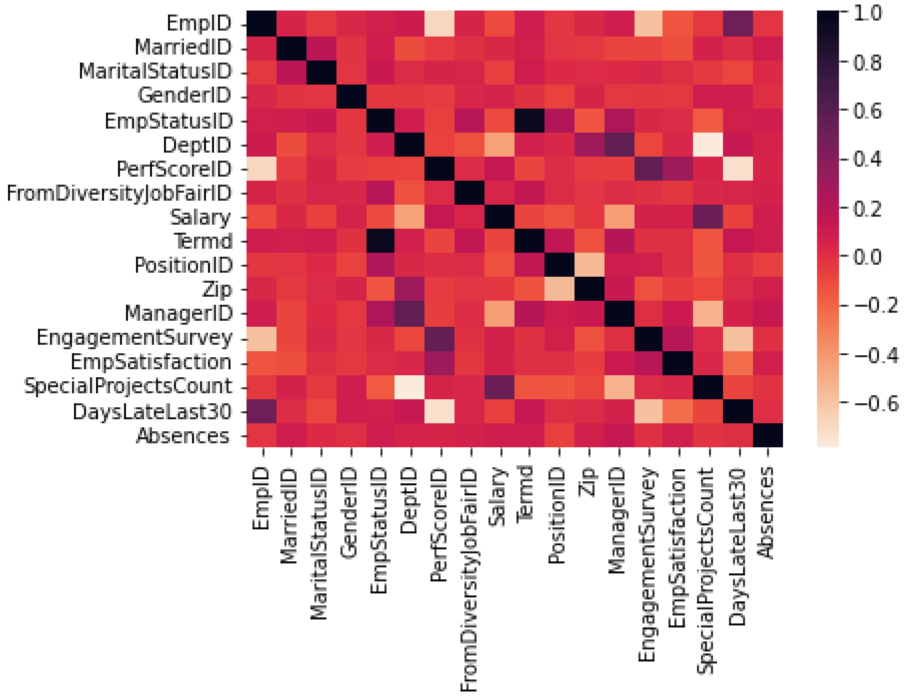
```
Employee_Name                  0
EmpID                          0
MarriedID                      0
MaritalStatusID                0
GenderID                       0
EmpStatusID                    0
DeptID                         0
PerfScoreID                    0
FromDiversityJobFairID         0
Salary                         0
Termd                          0
PositionID                     0
Position                       0
State                          0
Zip                            0
DOB                            0
Sex                            0
MaritalDesc                    0
CitizenDesc                    0
HispanicLatino                 0
RaceDesc                       0
DateofHire                     0
DateofTermination            207
TermReason                     0
EmploymentStatus               0
Department                     0
ManagerName                    0
ManagerID                      8
RecruitmentSource              0
PerformanceScore               0
EngagementSurvey               0
EmpSatisfaction                0
SpecialProjectsCount           0
LastPerformanceReview_Date     0
DaysLateLast30                 0
Absences                       0
dtype: int64
```

Удалим колонки, которые не влияют на целевой признак

Построим heatmap для лучшего визуального представления всез корреляций

In [7]:

```
cmap = sns.cm.rocket_r
ax = sns.heatmap(data.corr(), cmap=cmap)
```



In [8]:

```
data = data.drop(columns=['Employee_Name', 'EmpID', 'DateofTermination',
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 311 entries, 0 to 310
Data columns (total 32 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   MarriedID                   311 non-null    int64
 1   MaritalStatusID             311 non-null    int64
 2   GenderID                    311 non-null    int64
 3   EmpStatusID                 311 non-null    int64
 4   DeptID                      311 non-null    int64
 5   PerfScoreID                 311 non-null    int64
 6   FromDiversityJobFairID      311 non-null    int64
 7   Salary                      311 non-null    int64
 8   Termd                       311 non-null    int64
 9   PositionID                  311 non-null    int64
 10  Position                    311 non-null    object
 11  State                       311 non-null    object
 12  Zip                         311 non-null    int64
 13  DOB                         311 non-null    object
 14  Sex                         311 non-null    object
 15  MaritalDesc                 311 non-null    object
 16  CitizenDesc                 311 non-null    object
 17  HispanicLatino              311 non-null    object
 18  RaceDesc                    311 non-null    object
 19  DateofHire                  311 non-null    object
 20  TermReason                  311 non-null    object
 21  EmploymentStatus            311 non-null    object
 22  Department                  311 non-null    object
 23  ManagerName                 311 non-null    object
 24  RecruitmentSource           311 non-null    object
 25  PerformanceScore            311 non-null    object
 26  EngagementSurvey            311 non-null    float64
 27  EmpSatisfaction             311 non-null    int64
 28  SpecialProjectsCount        311 non-null    int64
 29  LastPerformanceReview_Date  311 non-null    object
 30  DaysLateLast30              311 non-null    int64
```

```
 31   Absences                     311 non-null    int64
dtypes: float64(1), int64(15), object(16)
memory usage: 77.9+ KB
```

Обработка пропусков

In [9]:

```
# Импьютация наиболее частыми значениями
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'>
RangeIndex: 311 entries, 0 to 310
Data columns (total 32 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   MarriedID                    311 non-null    int64
 1   MaritalStatusID              311 non-null    int64
 2   GenderID                     311 non-null    int64
 3   EmpStatusID                  311 non-null    int64
 4   DeptID                       311 non-null    int64
 5   PerfScoreID                  311 non-null    int64
 6   FromDiversityJobFairID       311 non-null    int64
 7   Salary                       311 non-null    int64
 8   Termd                        311 non-null    int64
 9   PositionID                   311 non-null    int64
 10  Position                     311 non-null    object
 11  State                        311 non-null    object
 12  Zip                          311 non-null    int64
 13  DOB                          311 non-null    object
 14  Sex                          311 non-null    object
 15  MaritalDesc                  311 non-null    object
 16  CitizenDesc                  311 non-null    object
 17  HispanicLatino               311 non-null    object
 18  RaceDesc                     311 non-null    object
 19  DateofHire                   311 non-null    object
 20  TermReason                   311 non-null    object
 21  EmploymentStatus             311 non-null    object
 22  Department                   311 non-null    object
 23  ManagerName                  311 non-null    object
 24  RecruitmentSource            311 non-null    object
 25  PerformanceScore             311 non-null    object
 26  EngagementSurvey             311 non-null    float64
 27  EmpSatisfaction              311 non-null    int64
 28  SpecialProjectsCount         311 non-null    int64
 29  LastPerformanceReview_Date   311 non-null    object
 30  DaysLateLast30               311 non-null    int64
 31  Absences                     311 non-null    int64
```

```
dtypes: float64(1), int64(15), object(16)
memory usage: 77.9+ KB
```

Кодирование строковых признаков (LabelEncoding)

In [10]:

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

In [11]:

```
le = preprocessing.LabelEncoder()

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

data
```

Out[11]:

	MarriedID	MaritalStatusID	GenderID	EmpStatusID	DeptID	PerfScoreID	FromDiversity
0	0	0	1	1	5	4	
1	1	1	1	5	3	3	
2	1	1	0	5	5	3	
3	1	1	0	1	5	3	
4	0	2	0	5	5	3	
...	...	...	...	...	...	...	
306	0	0	1	1	5	3	
307	0	0	0	5	5	1	
308	0	0	0	1	3	4	
309	0	0	0	1	3	3	
310	0	4	0	1	5	3	

311 rows × 32 columns

Масштабируем числовые данные

In [12]:

```
scaler = preprocessing.MinMaxScaler()

number_fields_source = number_cols.loc[:, number_cols.columns!=TARGET_CO

for col_name in number_fields_source:
    data[col_name] = scaler.fit_transform(data[[col_name]])

data
```

Out[12]:

	MarriedID	MaritalStatusID	GenderID	EmpStatusID	DeptID	PerfScoreID	FromDiversity
0	0.0	0.00	1.0	0.0	0.8	1.000000	
1	1.0	0.25	1.0	1.0	0.4	0.666667	
2	1.0	0.25	0.0	1.0	0.8	0.666667	
3	1.0	0.25	0.0	0.0	0.8	0.666667	
4	0.0	0.50	0.0	1.0	0.8	0.666667	

...	...	...	...	...	...	...
306	0.0	0.00	1.0	0.0	0.8	0.666667
307	0.0	0.00	0.0	1.0	0.8	0.000000
308	0.0	0.00	0.0	0.0	0.4	1.000000
309	0.0	0.00	0.0	0.0	0.4	0.666667
310	0.0	1.00	0.0	0.0	0.8	0.666667

311 rows × 32 columns

Делим выборку на обучающую и тестовую

In [13]:

```
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 [14]:

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

Out[14]:

```
((248, 32), (248,))
```

In [15]:

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

Out[15]:

```
((63, 32), (63,))
```

In [16]:

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

Out[16]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

Логистическая регрессия

In [17]:

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

```
C:\Users\pstri\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9
_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sklearn\l
inear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    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-re
gression
  n_iter_i = _check_optimize_result(
```

Out[17]:

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

In [18]:

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

Out[18]: 0.7619047619047619

In [19]:
```
f1_score(data_y_test, data_y_pred_1, average='micro')
```

Out[19]: 0.7619047619047619

In [20]:
```
f1_score(data_y_test, data_y_pred_1, average='macro')
```

Out[20]: 0.7208312792201521

In [21]:
```
f1_score(data_y_test, data_y_pred_1, average='weighted')
```

Out[21]: 0.7563589699202566

In [22]:
```
svr_2 = LogisticRegression(solver='lbfgs', max_iter=10000)
svr_2.fit(data_X_train, data_y_train)
```

```
C:\Users\pstri\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9
_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sklearn\l
inear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converg
e (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown i
n:
    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-re
gression
  n_iter_i = _check_optimize_result(
```

Out[22]: LogisticRegression(max_iter=10000)

In [23]:
```
data_y_pred_2 = svr_2.predict(data_X_test)
accuracy_score(data_y_test, data_y_pred_2)
```

Out[23]: 0.8095238095238095

In [24]:
```
f1_score(data_y_test, data_y_pred_2, average='micro')
```

Out[24]: 0.8095238095238095

In [25]:
```
f1_score(data_y_test, data_y_pred_2, average='macro')
```

Out[25]: 0.5976659982174688

In [26]:
```
f1_score(data_y_test, data_y_pred_2, average='weighted')
```

Out[26]: 0.7990160710748947

Случайный лес

In [27]:
```
RT = RandomForestClassifier(n_estimators=15, random_state=123)
RT.fit(data_X_train, data_y_train)
```

Out[27]:
```
RandomForestClassifier(n_estimators=15, random_state=123)
```

In [28]:
```
accuracy_score(data_y_test, RT.predict(data_X_test))
```

Out[28]:
```
0.7777777777777778
```

In [29]:
```
f1_score(data_y_test, data_y_pred_1, average='micro')
```

Out[29]:
```
0.7619047619047619
```

In [30]:
```
f1_score(data_y_test, data_y_pred_1, average='macro')
```

Out[30]:
```
0.7208312792201521
```

In [31]:
```
f1_score(data_y_test, data_y_pred_1, average='weighted')
```

Out[31]:
```
0.7563589699202566
```

In [32]:
```
RT = RandomForestClassifier(n_estimators=30, random_state=123)
RT.fit(data_X_train, data_y_train)
```

Out[32]:
```
RandomForestClassifier(n_estimators=30, random_state=123)
```

In [33]:
```
accuracy_score(data_y_test, RT.predict(data_X_test))
```

Out[33]:
```
0.873015873015873
```

In [34]:
```
f1_score(data_y_test, data_y_pred_1, average='micro')
```

Out[34]:
```
0.7619047619047619
```

In [35]:
```
f1_score(data_y_test, data_y_pred_1, average='macro')
```

Out[35]:
```
0.7208312792201521
```

In [36]:
```
f1_score(data_y_test, data_y_pred_1, average='weighted')
```

Out[36]:
```
0.7563589699202566
```

Выводы

При использовании логистической регрессии наилучшую точность (0.809) показала модель с параметром `max_iter=10000`. При использовании метода "Случайный лес" получилось добиться более высокого показателя точности (0.873), поэтому в целом предпочтительнее использовать его.