Задача 4

Условие

Набор данных: <https://www.kaggle.com/arashnic/hr-analytics-job-change-of-data-scientists?select=aug_test.csv>

Тип классификатора: *LogisticRegression (логистическая регрессия)*

Классификация по столбцу: *Education level (Masters – класс 0, остальные уровни – класс 1)*

1. Обработайте набор данных набор данных, указанный во втором столбце таблицы 4.1, подготовив его к решению задачи классификации. Выделите целевой признак, указанный в последнем столбце таблицы, и удалите его из данных, на основе которых будет обучаться классификатор. Разделите набор данных на тестовую и обучающую выборку. Постройте классификатор типа, указанного в третьем столбце, для задачи классификации по параметру, указанному в последнем столбце. Оцените точность построенного классификатора с помощью метрик precision, recall и F1 на тестовой выборке.
2. Постройте классификатор типа Случайный Лес (Random Forest) для решения той же задачи классификации. Оцените его качество с помощью метрик precision, recall и F1 на тестовой выборке. Какой из классификаторов оказывается лучше?

Решение:

1. Опишем данные в столбцах:
   1. enrollee\_id : уникальный идентификатор кандидата
   2. city: Код города
   3. city\_ development \_index : индекс развития города (в масштабе)
   4. gender: Пол кандидата
   5. relevent\_experience: Соответствующий опыт кандидата
   6. enrolled\_university: Тип зачисленных университетских курсов, если таковые имеются.
   7. education\_level: Уровень образования кандидата
   8. major\_discipline :Обучение основной дисциплине кандидата
   9. experience: Кандидатский общий стаж в годах
   10. company\_size: Количество сотрудников в компании текущего работодателя
   11. company\_type : Тип текущего работодателя
   12. lastnewjob: разница в годах между предыдущей работой и текущей работой
   13. training\_hours: завершенные часы обучения
   14. target: 0 - Не ищу смены работы, 1 - Ищу смену работы
2. Смотрим информацию о датасете:

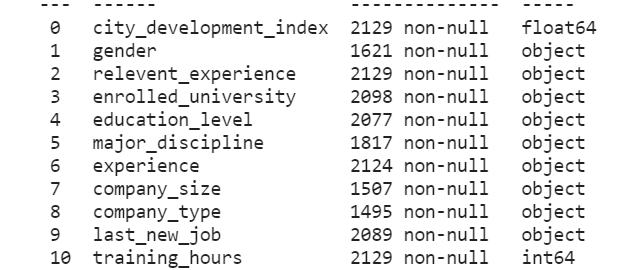


Рисунок 1. Информация о датасете.

Из Рисунка 1 видим, что некоторые признаки имеют пропуски и являются типами “object”, а не численными.

1. Обработаем данные для решения задачи классификации.
   1. Кодируем столбцы из категориального признака в численный. Смотри приложение 1 строки 28-61.
   2. Будем заполнять пропуски в данных значением по умолчанию – индексом 0. Смотри приложение 1 строки 62-66.
2. Выделяем целевой признак *- education\_level.* И удаляем его из данных, на основе которых будет обучаться классификатор. Смотри приложение 1 строки 67-68.
3. Разделяем набор данных на тестовую и обучающую выборки. Смотри приложение 1 строки 71-74.
4. Построим классификатор *LogisticRegression* сначала для обучающей выборки, а после для тестового набора данных*, используя оценщик GridSearchCV.* Смотри приложение 1 строки 76-83.



Рисунок 2. Оптимальные параметры, подобранные GridSearchCV.

* 1. Оценим точность построенного классификатора с помощью метрик precision, recall и F1 для тестового набора данных.

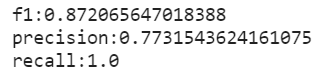


Рисунок 3. Результаты метрик для обучающей выборки.

* 1. Оценим точность построенного классификатора с помощью метрик precision, recall и F1 для обучающей выборки.

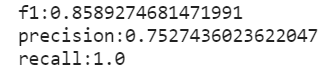


Рисунок 4. Результат метрик для тестового набора данных.

1. Построим классификатор типа Случайный Лес (Random Forest) для решения той же задачи классификации. Смотри приложение 1.
   1. Оценим качество с помощью метрик precision, recall и F1 на обучающей выборкe.

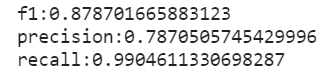


Рисунок 5. Результат метрик для обучающей выборки, классификатором Random Forest.

* 1. Оценим качество с помощью метрик precision, recall и F1 на тестовой выборкe.

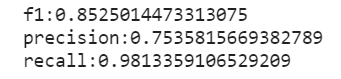


Рисунок 6. Результат метрик для тестовой выборки, классификатором Random Forest.

Полный код решения задачи приведён в Приложении 1.

Вывод:

На основе всех полученных данных я оцениваю классификатор Логистической регрессии, как довольно точный классификатор, так как он имеет довольно хорошую точность в 75%, полноту 100%, и f-меру в 86%.

Сравнивая результаты метрик классификаторов: Логистической регрессии и Рандомного леса, видим, что классификатор Логистической регрессии лучше, чем классификатор Рандомного леса. Смотри таблицу 1.

|  |  |  |
| --- | --- | --- |
| Тип метрики: | LogisticRegression | Random Forest |
| f1 | 85,9% | 85,2% |
| precision | 75,2% | 75,3 |
| recall | 100% | 98% |

Таблица 1. Сравнение метрик классификаторов Логистической регрессии и Рандомного леса.

Приложение 1

Устанавливаем необходимые библиотеки:

1. !pip install pandas
2. !pip install sklearn
3. import pandas as pd
4. import numpy as np
5. import warnings
6. warnings.filterwarnings('ignore')
7. data = pd.read\_csv('aug\_test.csv', index\_col='city')
8. #исключим столбец с уникальными значениями: enrollee\_id
9. data = data[data.columns.drop(['enrollee\_id'])]
10. data.head()
11. Описание столбцов:
12. enrollee\_id : уникальный идентификатор кандидата
13. city: Код города
14. city\_ development \_index : индекс развития города (в масштабе)
15. gender: Пол кандидата
16. relevent\_experience: Соответствующий опыт кандидата
17. enrolled\_university: Тип зачисленных университетских курсов, если таковые имеются.
18. education\_level: Уровень образования кандидата
19. major\_discipline :Обучение основной дисциплине кандидата
20. experience: Кандидатский общий стаж в годах
21. company\_size: Количество сотрудников в компании текущего работодателя
22. company\_type : Тип текущего работодателя
23. lastnewjob: разница в годах между предыдущей работой и текущей работой
24. training\_hours: завершенные часы обучения
25. target: 0 - Не ищу смены работы, 1 - Ищу смену работы
26. data.info()
27. #Кодируем столбец из категориального признака
28. data['gender'] = np.where(data['gender'] == 'Male', 1, 0)
29. data['education\_level'] = np.where(data['education\_level'] == 'Masters', 0, 1)
30. data['relevent\_experience'] = np.where(data['relevent\_experience'] == 'No relevent experience',0,1)
31. my\_set = set(data.enrolled\_university)
32. i = 0
33. for item in my\_set:
34. data['enrolled\_university'] = data['enrolled\_university'].replace(item, i)
35. i = i + 1
36. my\_set = set(data.last\_new\_job)
37. i = 0
38. for item in my\_set:
39. data['last\_new\_job'] = data['last\_new\_job'].replace(item, i)
40. i = i + 1
41. my\_set = set(data.major\_discipline)
42. i = 0
43. for item in my\_set:
44. data['major\_discipline'] = data['major\_discipline'].replace(item, i)
45. i = i + 1
47. my\_set = set(data.company\_type)
48. i = 0
49. for item in my\_set:
50. data['company\_type'] = data['company\_type'].replace(item, i)
51. i = i + 1
52. my\_set = set(data.company\_size)
53. i = 0
54. for item in my\_set:
55. data['company\_size'] = data['company\_size'].replace(item, i)
56. i = i + 1
57. my\_set = set(data.experience)
58. i = 0
59. for item in my\_set:
60. data['experience'] = data['experience'].replace(item, i)
61. i = i + 1
62. data.loc[data.experience == 'NaN' , 'experience'] = 0
63. data.loc[data.last\_new\_job == 'never' , 'last\_new\_job'] = 0
64. data.loc[data.last\_new\_job == 'NaN' , 'last\_new\_job'] = 0
65. data.loc[data.last\_new\_job == '>4' , 'last\_new\_job'] = 5
66. data.loc[data.gender == 'NaN' , 'gender'] = 0
67. target = data.education\_level
68. train = data.drop('education\_level', axis = 1)
69. train.info()
70. #y - целевая переменнная (target)
71. from sklearn.model\_selection import train\_test\_split
72. X\_train, X\_test, y\_train, y\_test = train\_test\_split(train, target, test\_size = 0.3, random\_state = 42)
73. N\_train, \_ = X\_train.shape
74. N\_test, \_ = X\_test.shape
75. print (N\_train, N\_test)
76. from sklearn.linear\_model import LogisticRegression
77. from sklearn.model\_selection import GridSearchCV
78. from sklearn.model\_selection import StratifiedKFold
79. grid={"C":np.logspace(-5,5,11), "penalty":["l1","l2"]}
80. logreg=LogisticRegression(solver='liblinear')
81. logreg\_cv=GridSearchCV(logreg,grid,cv=10)
82. logreg\_cv.fit(X\_train,y\_train)
83. logreg\_cv.best\_estimator\_
84. from sklearn.model\_selection import cross\_val\_score
85. print("f1:"+str(np.average(cross\_val\_score(logreg\_cv, X\_train, y\_train, scoring='f1'))))
86. print("precision:"+str(np.average(cross\_val\_score(logreg\_cv, X\_train, y\_train, scoring='precision'))))
87. print("recall:"+str(np.average(cross\_val\_score(logreg\_cv, X\_train, y\_train, scoring='recall'))))
88. from sklearn.model\_selection import cross\_val\_score
89. print("f1:"+str(np.average(cross\_val\_score(logreg\_cv, X\_test, y\_test, scoring='f1'))))
90. print("precision:"+str(np.average(cross\_val\_score(logreg\_cv, X\_test, y\_test, scoring='precision'))))
91. print("recall:"+str(np.average(cross\_val\_score(logreg\_cv, X\_test, y\_test, scoring='recall'))))
92. from sklearn.ensemble import RandomForestClassifier
93. param\_grid = { 'n\_estimators': [50, 100, 150],'max\_features': ['auto'],'max\_depth' : list(range(1, 10)), 'criterion' :['gini']}
94. RandForCrit = GridSearchCV(estimator = RandomForestClassifier(), param\_grid = param\_grid, cv = 5, refit = True)
95. RandForCrit.fit(X\_train, y\_train)
96. RandForCrit.predict(X\_test)
97. print("f1:"+str(np.average(cross\_val\_score(RandForCrit.best\_estimator\_, X\_train, y\_train, scoring='f1'))))
98. print("precision:"+str(np.average(cross\_val\_score(RandForCrit.best\_estimator\_, X\_train, y\_train, scoring='precision'))))
99. print("recall:"+str(np.average(cross\_val\_score(RandForCrit.best\_estimator\_, X\_train, y\_train, scoring='recall'))))
100. print("f1:"+str(np.average(cross\_val\_score(RandForCrit.best\_estimator\_, X\_test, y\_test, scoring='f1'))))
101. print("precision:"+str(np.average(cross\_val\_score(RandForCrit.best\_estimator\_, X\_test, y\_test, scoring='precision'))))
102. print("recall:"+str(np.average(cross\_val\_score(RandForCrit.best\_estimator\_, X\_test, y\_test, scoring='recall'))))