Ваганов Д.Д. ИУ5-62Б

Рубежный контроль №2 (4 вариант)

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.

Метод 1 - Метод опорных векторов

Метод 2 - Случайный лес

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.ensemble import RandomForestClassifier
```

In [2]:
```
df = pd.read_csv("toy_dataset.csv", delimiter=",")
```

In [3]:
```
#Проверям датасет на наличие пустых значений
for col in df.columns:
    missing = df[col].isnull()
    num_missing = np.sum(missing)
    if num_missing > 0:
        print('detected {}row null in {}'.format(num_missing,col))
    else:
        print('column {} is okay'.format(col))
```

```
column Number is okay
column City is okay
column Gender is okay
column Age is okay
column Income is okay
column Illness is okay
```

Пропуски не обнаружены

In [4]:
```
#Первые пять строк датасета
df.head()
```

Out[4]:

	Number	City	Gender	Age	Income	Illness
0	1	Dallas	Male	41	40367.0	No
1	2	Dallas	Male	54	45084.0	No
2	3	Dallas	Male	42	52483.0	No
3	4	Dallas	Male	40	40941.0	No
4	5	Dallas	Male	46	50289.0	No

In [5]:
```
#Так как Number является порядковым номером строки, удалим его
df = df.drop(columns=['Number'], axis=1)
df.head()
```

Out[5]:

	City	Gender	Age	Income	Illness
0	Dallas	Male	41	40367.0	No
1	Dallas	Male	54	45084.0	No
2	Dallas	Male	42	52483.0	No
3	Dallas	Male	40	40941.0	No
4	Dallas	Male	46	50289.0	No

Преобразование категориальных признаков в числовые

In [6]:
```
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder
```

Кодируем столбец 'City'

In [7]:
```
df['City'].unique()
```

Out[7]:
```
array(['Dallas', 'New York City', 'Los Angeles', 'Mountain View',
       'Boston', 'Washington D.C.', 'San Diego', 'Austin'], dtype=object)
```

In [8]:
```
ord_enc = OrdinalEncoder(dtype=np.int64)
```

In [9]:
```
df['City'] = ord_enc.fit_transform(df[['City']])
```

In [10]:
```
df['City'].unique()
```

Out[10]:
```
array([2, 5, 3, 4, 1, 7, 6, 0])
```

Кодируем столбец 'Gender'

In [11]:
```
df['Gender'].unique()
```

Out[11]:
```
array(['Male', 'Female'], dtype=object)
```

In [12]:
```
legender = LabelEncoder()
legender_arr = legender.fit_transform(df['Gender'])
df['Gender'] = legender_arr
df['Gender'].unique()
```

Out[12]:
```
array([1, 0])
```

Кодируем столбец 'Illness'

In [13]:
```
df['Illness'].unique()
```

Out[13]:
```
array(['No', 'Yes'], dtype=object)
```

In [14]:
```
leill = LabelEncoder()
df['Illness'] = leill.fit_transform(df['Illness'])
df['Illness'].unique()
```

Out[14]:
```
array([0, 1])
```

In [15]:
```
df.head()
```

Out[15]:

	City	Gender	Age	Income	Illness
0	2	1	41	40367.0	0
1	2	1	54	45084.0	0
2	2	1	42	52483.0	0
3	2	1	40	40941.0	0
4	2	1	46	50289.0	0

Возьмем 10000 строк для построение модели

In [16]:
```
from sklearn.utils import shuffle
```

In [17]:
```
df.shape
```

Out[17]:
```
(150000, 5)
```

In [18]:
```
df_cut = shuffle(df, random_state=1).reset_index(drop=True)
df_cut = df.iloc[0:10000, :]
```

In [19]:
```
df_cut.shape
```

Out[19]:
```
(10000, 5)
```

В качестве целевого признака выбрано наличие болезни ('Illness')

In [20]:
```
x_train, x_test, y_train, y_test = train_test_split(df_cut.drop('Illness', axis=1), df_cut['Illness'], test_size=0.5,
random_state=4)
```

Выбор метрик

Так как выполняется задача небинарной классификации и в тестовой выборке возможен дисбаланс классов, были выбраны следующие метрики:

-precision;
-recall;
-f1-score.
Всем метрикам был задан уровень детализации average='weighted'.

In [21]:
```
def print_metrics(y_test, y_pred):
    rep = classification_report(y_test, y_pred, output_dict=True, zero_division=0)
    print("weighted precision:", rep['weighted avg']['precision'])
    print("weighted recall:", rep['weighted avg']['recall'])
    print("weighted f1-score:", rep['weighted avg']['f1-score'])
```

Метод опорных векторов

Базовая модель

In [22]:
```
svm_model = SVC()
svm_model.fit(x_train, y_train)
y_pred_svm = svm_model.predict(x_test)
print_metrics(y_test, y_pred_svm)
```

```
weighted precision: 0.85414564
weighted recall: 0.9242
weighted f1-score: 0.8877929944912172
```

Подбор гиперпараметров

In [23]:
```
params = {'C': np.concatenate([np.arange(0.1, 2, 0.03), np.arange(2, 20, 1)])}
grid_cv = GridSearchCV(estimator=svm_model, param_grid=params, cv=10, n_jobs=-1, scoring='f1_macro')
grid_cv.fit(x_train, y_train)
print(grid_cv.best_params_)
```

```
{'C': 0.1}
```

Лучшая модель

In [24]:
```
best_svm_model = grid_cv.best_estimator_
best_svm_model.fit(x_train, y_train)
y_pred_svm = best_svm_model.predict(x_test)
print_metrics(y_test, y_pred_svm)
```

```
weighted precision: 0.85414564
weighted recall: 0.9242
weighted f1-score: 0.8877929944912172
```

Случайный лес

In [25]:
```
rfc_model = RandomForestClassifier()
rfc_model.fit(x_train, y_train)
y_pred_rfc = rfc_model.predict(x_test)
print_metrics(y_test, y_pred_rfc)
```

```
weighted precision: 0.8629238282771686
weighted recall: 0.8932
weighted f1-score: 0.8771246233451695
```

Подбор гиперпараметров

In [26]:
```
params = {'n_estimators': [5, 10, 50, 100], 'max_features': [2, 3, 4], 'criterion': ['gini', 'entropy'], 'min_samples_
leaf': [1, 2, 3, 4, 5]}
grid_cv = GridSearchCV(estimator=rfc_model, param_grid=params, cv=10, n_jobs=-1, scoring='f1_weighted')
grid_cv.fit(x_train, y_train)
print(grid_cv.best_params_)
```

```
{'criterion': 'gini', 'max_features': 3, 'min_samples_leaf': 5, 'n_estimators': 5}
```

In [27]:
```
best_rfc_model = grid_cv.best_estimator_
best_rfc_model.fit(x_train, y_train)
y_pred_rfc = best_rfc_model.predict(x_test)
print_metrics(y_test, y_pred_rfc)
```

```
weighted precision: 0.8707478711821672
weighted recall: 0.9226
weighted f1-score: 0.8881275860397595
```

Сравнение результатов

In [28]:
```
print("SVC result\n")
print_metrics(y_test, y_pred_svm)
```

```
SVC result

weighted precision: 0.85414564
weighted recall: 0.9242
weighted f1-score: 0.8877929944912172
```

In [29]:
```
print("RandomForestClassifier result\n")
print_metrics(y_test, y_pred_rfc)
```

```
RandomForestClassifier result

weighted precision: 0.8707478711821672
weighted recall: 0.9226
weighted f1-score: 0.8881275860397595
```

Вывод

Модели с подобранными гиперпараметрами оказались лучше базовых моделей. Метрики показывают, что качество модели SVM немного выше, чем модели "случайного леса".

In [ ]:
```

```