РК2 Бахрамов Никита Андреевич ИУ5-63Б

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.

Датасет https://www.kaggle.com/carlolepelaars/toy-dataset

Методы:

1. Случайный лес
2. Дерево решений

```
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.metrics import classification_report, confusion_matrix
from sklearn.ensemble import RandomForestClassifier

data = pd.read_csv("toy_dataset.csv", delimiter=",")

print('Количество пропущенных значений')
data.isnull().sum()
```

```
Количество пропущенных значений

Number     0
City       0
Gender     0
Age        0
Income     0
Illness    0
dtype: int64
```

```
data.head(5)
```

```
   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
```

```
data = data.drop(columns=['Number'], axis=1)
data.head()
```

```
     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

```
from sklearn.preprocessing import LabelEncoder, OrdinalEncoder

data['City'].unique()
```

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

```
ord_enc = OrdinalEncoder(dtype=np.int64)
data['City'] = ord_enc.fit_transform(data[['City']])
data['City'].unique()
```

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

По аналогии кодируем другие столбцы

```
legender = LabelEncoder()
legender_arr = legender.fit_transform(data['Gender'])
data['Gender'] = legender_arr
data['Gender'].unique()
```

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

```
leill = LabelEncoder()
data['Illness'] = leill.fit_transform(data['Illness'])
data['Illness'].unique()
```

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

```
data.head()
```

```
   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 1000 строк для построения модели

```
from sklearn.utils import shuffle

data_cut = shuffle(data, random_state=1).reset_index(drop=True)
data_cut = data.iloc[0:1000, :]

data_cut.shape
```

```
(1000, 5)
```

Целевой признак: illness

```
x_train, x_test, y_train, y_test =
train_test_split(data_cut.drop('Illness', axis=1),
data_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'.

```
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 случайного Леса

```
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: 0.8559183673469388
weighted recall: 0.9
weighted f1-score: 0.8697639225181599
```

Подбор гиперпараметров

```
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': 2, 'min_samples_leaf': 1,
'n_estimators': 50}
```

```
best_rfc_model = grid_cv.best_estimator_
best_rfc_model.fit(x_train, y_train)
y_pred_tree = best_rfc_model.predict(x_test)
print_metrics(y_test, y_pred_tree)
```

```
weighted precision: 0.85799968409414
weighted recall: 0.898
weighted f1-score: 0.8712630991192204
```

метод дерева решений

```
tree = DecisionTreeClassifier()
tree.fit(x_train,y_train)
y_pred_tree = tree.predict(x_test)
print_metrics(y_test, y_pred_tree)
```

```
weighted precision: 0.848463804820396
weighted recall: 0.838
weighted f1-score: 0.8430979719503582
```

```
from sklearn.tree import DecisionTreeClassifier

params = {'min_samples_leaf': range(3, 30)}
tree = DecisionTreeClassifier(random_state=3)
grid_cv_tree = GridSearchCV(estimator=tree, cv=10, param_grid=params,
n_jobs=-1, scoring='f1_weighted')
grid_cv_tree.fit(x_train, y_train)
print(grid_cv_tree.best_params_)
```

```
{'min_samples_leaf': 10}
```

```
best_tree = grid_cv_tree.best_estimator_
best_tree.fit(x_train, y_train)
y_pred_tree = best_tree.predict(x_test)
print_metrics(y_test, y_pred_tree)
```

```
weighted precision: 0.8244640000000001
weighted recall: 0.908
weighted f1-score: 0.8642180293501048
```

Вывод:

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