Практическая работа 4.

Лазарев Александр. КМБО-03-22.

Вариант 16.

Установка данных.

```
import pandas as pd

# Загрузка данных
data = pd.read_csv('StudentsPerformance.csv')

print(data)
```

```
     gender race/ethnicity parental level of education
lunch  \
0    female        group B           bachelor's degree      standard

1    female        group C                some college      standard

2    female        group B             master's degree      standard

3      male        group A          associate's degree  free/reduced

4      male        group C                some college      standard

..      ...            ...                         ...           ...

995  female        group E             master's degree      standard

996    male        group C                 high school  free/reduced

997  female        group C                 high school  free/reduced

998  female        group D                some college      standard

999  female        group D                some college  free/reduced


    test preparation course  math score  reading score  writing score
0                      none          72             72             74

1                 completed          69             90             88

2                      none          90             95             93

3                      none          47             57             44
```

```
4                      none          76             78             75
..                      ...         ...            ...            ...
995               completed          88             99             95
996                    none          62             55             55
997               completed          59             71             65
998               completed          68             78             77
999                    none          77             86             86

[1000 rows x 8 columns]
```

Данные установлены корректно. Переименуем столбцы для удобной работы и проверим на наличие дубликатов.

```
# Переименование.
data.rename(columns={'race/ethnicity': 'ethnicity',
                           'parental level of education':
'parental_level_education',
                            'test preparation course': 'test_course',
'math score': 'math_score',
                            'reading score': 'reading_score', 'writing
score': 'writing_score'}, inplace=True)
# Проверка на дубликаты.
data.duplicated().value_counts()
```

```
False    1000
dtype: int64
```

Дубликаты отсутствуют, данные готовы к работе.

Ознакомимся с возможными значениями.

```
print('gender:',data['gender'].unique())
print('ethnicity:',data['ethnicity'].unique())
print('parental level education:',
data['parental_level_education'].unique())
print('lunch:', data['lunch'].unique())
print('test course:', data['test_course'].unique())
```

```
gender: ['female' 'male']
ethnicity: ['group B' 'group C' 'group A' 'group D' 'group E']
parental level education: ["bachelor's degree" 'some college'
"master's degree" "associate's degree"
 'high school' 'some high school']
```

```
lunch: ['standard' 'free/reduced']
test course: ['none' 'completed']
```

Таким образом:

1. Пол можно сделать дамми-переменной (мужчины - 1, женщины - 0).
2. Этнической принадлежности будут присвоены соответственно значения от 1 до 5.
3. Родительский уровень образования также будет пронумерован от 1 до 6 (чем лучше образование, тем выше значение).
4. Переменная, отвечающая за вид получаемого обеда - дамии-переменная (стандартный - 1, со скидкой или беспланый - 0).
5. Наличие подготовительных курсов - дамми-переменная (1 - проходил, 0 - не проходил).

Отмечу, что для построения классификатора необходимо, чтобы целевая переменная была бинарной. Таким образом, переменная writing_score будет иметь класс 0, если ее значение выше, и 1, если ее значение ниже или совпадает со средним.

Приступим к нормализации данных для осуществения анализа.

```
data['gender'] = data['gender'].replace({'female': 0, 'male': 1})
data['ethnicity'] = data['ethnicity'].replace({'group A': 1, 'group
B': 2, 'group C': 3, 'group D': 4, 'group E': 5})
data['parental_level_education'] =
data['parental_level_education'].replace({'some high school': 1, 'high
school': 2,
                              'some college': 3, "associate's degree":
4, "bachelor's degree": 5, "master's degree": 6})
data['lunch'] = data['gender'].replace({'free/reduced': 0, 'standard':
1})
data['test_course'] = data['test_course'].replace({'none': 0,
'completed': 1})
# Обработка целевого признака:
data['writing_score'] = data['writing_score'].astype(int) #
преобразуем столбец к целому типу
data['writing_score'] = data['writing_score'].apply(lambda x: 1 if x
<= data['writing_score'].mean() else 0)

data['math_score'] = data['math_score'].astype(int)  #все оставшиеся
исходные данные следует преобразовать к целому типу
data['reading_score'] = data['reading_score'].astype(int)
print(data) # ознакомимся с результатами
```

```
     gender  ethnicity  parental_level_education  lunch
test_course  \
0         0          2                         5      0            0

1         0          3                         3      0            1
```

```
2         0          2                          6      0            0

3         1          1                          4      1            0

4         1          3                          3      1            0

..      ...        ...                        ...    ...          ...

995       0          5                          6      0            1

996       1          3                          2      1            0

997       0          3                          2      0            1

998       0          4                          3      0            1

999       0          4                          3      0            0


     math_score  reading_score  writing_score
0            72             72              0
1            69             90              0
2            90             95              0
3            47             57              1
4            76             78              0
..          ...            ...            ...
995          88             99              0
996          62             55              1
997          59             71              1
998          68             78              0
999          77             86              0

[1000 rows x 8 columns]
```

Выделим целевой признак и удалим его из данных. Для этого используем метод drop():

```
# Выделение целевого признака
target = data['writing_score']

# Удаление целевого признака из данных
data.drop(columns=['writing_score'], inplace=True)
```

Разделим набор данных на обучающую и тестовую выборки. Для этого используем функцию train_test_split() из библиотеки scikit-learn:

```
from sklearn.model_selection import train_test_split
```

```
# Разделение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(data, target,
test_size=0.25, random_state=16)
```

Тестовая выборка составит четверть от всех данных. Также указан параметр random_state, для того чтобы данные не менялись при каждом запуске программы.

Создадим классификатор методом опорных векторов, проведем его обучение и предсказание ответов для тестовой выборки. Для этого используем класс SVM из библиотеки scikit-learn:

```
from sklearn.svm import SVC

# Создание классификатора
clf = SVC()

# Обучение классификатора на обучающей выборке
clf.fit(X_train, y_train)

# Предсказание классов на тестовой выборке
y_pred = clf.predict(X_test)
```

Оценим точность построенного классификатора с помощью метрик precision, recall и F1 с помощью значений y_test тестовой выборки. Будем использовать методы из уже упомянутой библиотеки scikit-learn.

```
from sklearn.metrics import precision_score, recall_score, f1_score

# Оценка качества модели
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print('Precision:', precision)
print('Recall:', recall)
print('F1-score:', f1)
```

```
Precision: 0.921875
Recall: 0.9365079365079365
F1-score: 0.9291338582677166
```

Значения метрик высоки, что говорит о хорошем качестве классификатора и о возможности проведения качественных предсказаний.

ЧАСТЬ 2.

Построим классификатор типа Случайный Лес (Random Forest) для решения вышеописанной задачи классификации. Для этого создадим сам

классификатор и зададим возможные значения гиперпараметра n_estimators, необходимые для выявления лучшего классификатора (первый перебор шагом 50 от 50 до 1000):

```
from sklearn.ensemble import RandomForestClassifier

# Создание классификатора
clf = RandomForestClassifier()

# Определение сетки гиперпараметров
param_grid = {'n_estimators': list(range(50, 1001, 50))} # число
деревьев в лесу.
```

Далее определим лучший набор гиперпараметров для построения классификатора, используя GridSearch. Сравнивать будем по уже указаным метрикам precision, recall и f1.

```
from sklearn.model_selection import GridSearchCV

grid_search = GridSearchCV(clf, param_grid, scoring=['precision',
'recall', 'f1'], refit='f1')
grid_search.fit(X_train, y_train)
print('Число деревьев для лучших значений метрик в первом приближении:
', grid_search.best_params_)
```

```
Число деревьев для лучших значений метрик в первом приближении:
{'n_estimators': 300}
```

Далее определим лучшее число деревьев во втором приближении:

```
param_grid = {'n_estimators': list(range(250, 351, 10))} # число
деревьев в лесу.
grid_search = GridSearchCV(clf, param_grid, scoring=['precision',
'recall', 'f1'], refit='f1')
grid_search.fit(X_train, y_train)
print('Число деревьев для лучших значений метрик во втором
приближении: ', grid_search.best_params_)
```

```
Число деревьев для лучших значений метрик во втором приближении:
{'n_estimators': 340}
```

Таким образом определен лучший классификатор. Проведем его анализ по аналогии с тем, что проводили для SVM классификатора:

```
# Обучение классификатора с лучшими гиперпараметрами на той же
обучающей выборке для сравнения классификаторов
best_clf = grid_search.best_estimator_
best_clf.fit(X_train, y_train)

# Предсказание классов на той же тестовой выборке
y_pred = best_clf.predict(X_test)
```

```
# Оценка качества модели с помощью метрик precision, recall и F1
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Вывод результатов
print('Precision:', precision)
print('Recall:', recall)
print('F1-score:', f1)
```

```
Precision: 0.944
Recall: 0.9365079365079365
F1-score: 0.9402390438247011
```

Заметен прирост точности предсказаний. Это говорит о том, что классификатор случайного дерева (собранный из лучших значений гиперпараметров) предсказывает данные точнее классификатора метода опорных векторов. Таким образом лучший классификатор для данного исследования это:

clf = RandomForestClassifier(n_estimators=340)