Лабораторная работа № 3

Классификация

Вариант № 4

Цель работы

Произвести классификацию методом KNN.

Задание

Необходимо модель действий сотрудника банка, принимающего решение о выдаче кредита. Данные, с которыми оперирует сотрудник банка - 15 переменных, содержание которых не раскрывается. В файле содержится также два столбца, показывающие, была удовлетворена заявка, или она была отвергнута. Если в первом столбце стоит единица, то заявка была отвергнута, если во втором столбце стоит единица, то заявка была удовлетворена. Таким образом, столбцы дублируют друг друга.

Число наблюдений – 655.

Число переменных – 15, из них 6 измерены в количественной (непрерывной) шкале, 9 – в шкале наименований (номинальной шкале).

Этапы работы:

1. Выделить обучающую и тестовую выборки.
2. Определить наилучшее значение k.
3. Оценить качество прогноза на тестовой выборке с помощью таблицы сопряженности.
4. Выдать процент ошибок, допущенных классификатором на тестовой выборке.

Решение

Для решения задачи были использованы библиотеки:

— Matplotlib - библиотека для графического представления данных.

— Pandas - программная библиотека на языке Python для обработки и анализа данных. Она представляет собой специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами.

— Scikit-learn - бесплатная библиотека программного обеспечения для машинного обучения для языка программирования Python. Она включает различные алгоритмы классификации, регрессии и кластеризации, включая методы опорных векторов, случайные леса, повышение градиента, k-средние и DBSCAN, и предназначена для взаимодействия с числовыми и научными библиотеками Python numpy и scipy.

Импортируем необходимые библиотеки и считываем данные из файла в датафрейм используя функцию read_csv:

```
1  import pandas as pd
2  from sklearn.model_selection import train_test_split
3  from sklearn.neighbors import KNeighborsClassifier
4  from sklearn.metrics import confusion_matrix, accuracy_score
5
6  data = pd.read_csv('Credit_Screening.dat', delimiter=';')
```

Рисунок 1 – Фрагмент кода. Импорт библиотек и чтение файла Credit_Screening.dat

Выполним над датаферймом преобразования:

```
8  X = data.iloc[:, :-2]  # Исключаем последние два столбца
9  y = data['desired1']  # Предполагаем, что 'desired1' - целевая переменная
10
11 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Рисунок 2 – Фрагмент кода. Преобразование данных.

Определяем наилучшее значение k:

```
best_k = None
best_accuracy = 0

for k in range(1, 10):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)

    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_k = k

print(f"Наилучшее значение k: {best_k}")
```

Рисунок 3 – Фрагмент кода. Определение лучшего значения k.

Затем необходимо оценить качество прогноза на тестовой выборке с помощью таблицы сопряженности. Для начала обучаем модель с лучшим k:

```
best_knn = KNeighborsClassifier(n_neighbors=best_k)
best_knn.fit(X_train, y_train)
y_pred = best_knn.predict(X_test)
```

Рисунок 4 – Фрагмент кода. Обучение модели с лучшем k

Затем выведем таблицу сопряженности и процент ошибок:

```
conf_matrix = confusion_matrix(y_test, y_pred)
print("Contingency table:")
print(conf_matrix)

error_rate = 100 * (conf_matrix[0, 1] + conf_matrix[1, 0]) / len(y_test)
print(f"Error rate: {error_rate}%")
```

Рисунок 5 – Фрагмент кода. Вывод таблицы сопряженности и процента ошибок

Запустив готовый код, получим данные в консоль:

```
Best k value: 7
Contingency table:
[[30 25]
 [13 63]]
Error rate in %: 29.00763358778626%
```

Рисунок 6 – Результат работы программы