Федеральное агентство связи

Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Лабораторная работа №1

Вариант №9

Выполнил: студент IV курса ИВТ,

гр. ИП-713

Михеев Н.А.

Проверила: ассистент кафедры ПМиК

Морозова К.И.

Новосибирск, 2020 г.

Цель

Суть лабораторной работы заключается в написании классификатора на основе метода k ближайших соседей. Данные из файла необходимо разбить на две выборки, обучающую и тестовую, согласно общепринятым правилам разбиения. На основе этих данных необходимо обучить разработанный классификатор и протестировать его на обеих выборках. В качестве отчёта требуется представить работающую программу и таблицу с результатами тестирования для каждого из 10 разбиений. Разбиение выборки необходимо выполнять программно, случайным образом, при этом, не нарушая информативности обучающей выборки. Разбивать рекомендуется по следующему правилу: делим выборку на 3 равных части, 2 части используем в качестве обучающей, одну в качестве тестовой. Кроме того, обучающая выборка должна быть сгенерирована таким образом, чтобы минимизировать разницу между количеством представленных в ней объектов разных классов, т.е. $abs\left(\left|\{(x_i, y_i) \in X^l \middle| y_i = -1\}\right| - \left|\{(x_i, y_i) \in X^l | y_i = 1\}\right|\right) \to min$.

Результат работы

В результате работы был реализован класс kwnn и функция скользящего контроля LOO(leave-one-out). Для проверки работоспособности программы сначала выполняется считывание данных, разбиение их на обучающую и тестовую выборку с помощью метода из библиотеки sklearn. Далее с помощью скользящего контроля LOO идет поиск наилучшего количества соседей k. Идет инициализация класса KWNN в который передается наше число соседей k и обучающая выборка – X и Y train. Далее идет уже предсказание по нашей тестовой выборке – X_test с весом неубывающей прогрессии. По полученному результату идет сравнение с истинными данными – Y_test и подсчет ошибок, и вывод результатов работы программы. При различном random_state разбиения точность алгоритма всегда >90%.

```
Searching the best k:
1: 0.0731
2: 0.0918
3: 0.0884
4: 0.0979
5: 0.1019
Best chosen k = 1
```

Рис.1 – поиск наилучшего кол-ва ближайших соседей методом скользящего контроля leave-one-out.

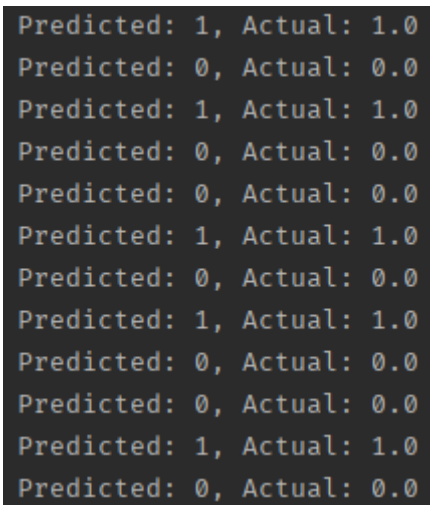

Рис.2 – работа программы, проверка предсказанного результата и настоящего из Y_test.

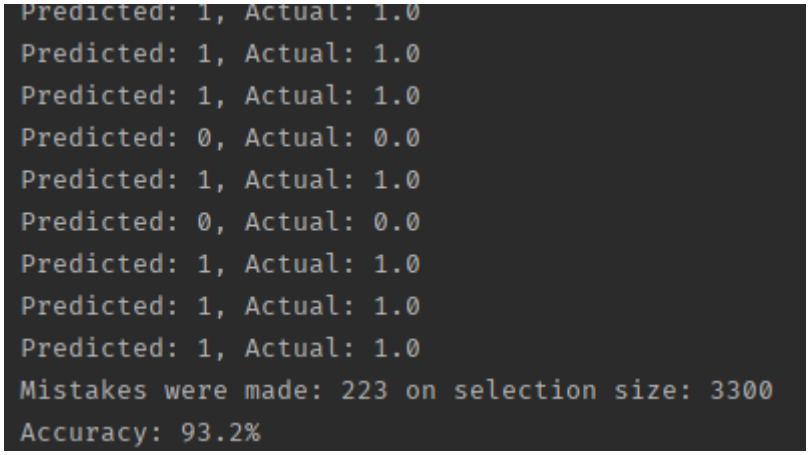

Рис.3 – вывод результатов работы о кол-ве ошибок, точность работы алгоритма.

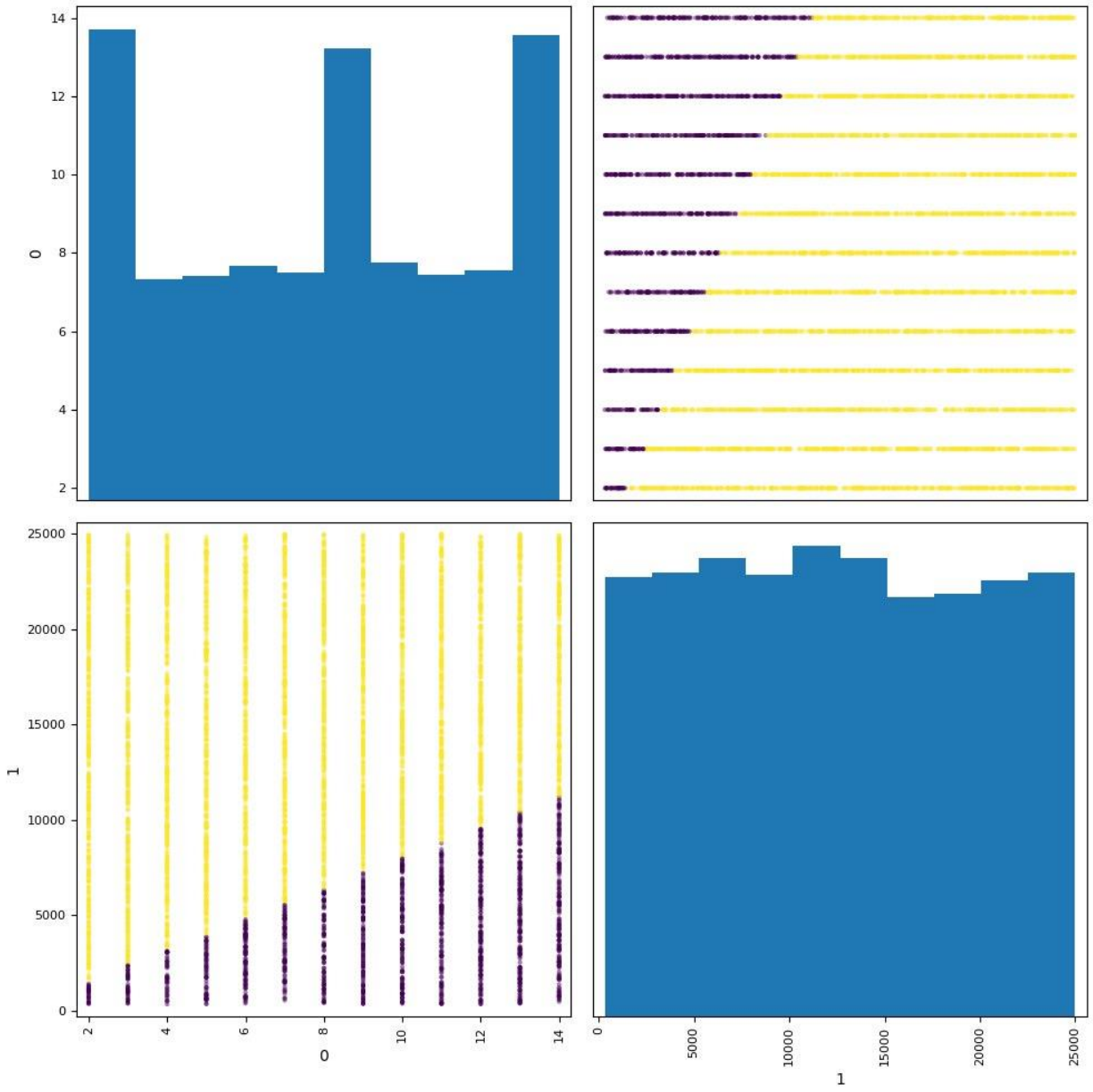

Рис.3 – графическое представление входных данных X_train с помощью pandas dataframe – диаграмма рассеяния.

Листинг программы

```
import numpy as np
from sklearn.model_selection import train_test_split


class KWNN:
    def __init__(self, k, x, y):
        self.k = int(k)
        self.x = x
        self.y = y

    def predict(self, x):
        y = []
        if x.ndim > 1:
```

```
            for obs in x:
                y.append(self._predict(obs))
        else:
            y.append(self._predict(x))
        return y

    def _predict(self, obs):
        distance = np.sqrt(np.sum((self.x - obs) ** 2, axis=1))
        idx = np.argsort(distance)[:self.k]
        counts = np.bincount(self.y[idx].astype(int))

        if len(counts):
            prediction = np.argmax(counts)
        else:
            prediction = -1
        return prediction


def LOO(x, y, param=1):
    scores = []

    for i in range(len(x)):
        test_x, test_y = x[i], y[i]
        knn = KWNN(param, np.delete(x, i, axis=0), np.delete(y, i))
        scores.append(knn.predict(test_x)[0] != test_y)
    return sum(scores) / len(x)


def main():
    raw_data = np.genfromtxt('data2.csv', delimiter=',', skip_header=True)
    X_data = np.stack((raw_data[:, 0], raw_data[:, 1]), axis=1)
    Y_data = raw_data[:, 2]
    X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data,
stratify=Y_data,
                                                        test_size=0.33,
random_state=12345)

    print("Searching the best k:")
    scores = {}
    for i in range(1, 6):
        score = LOO(X_train, Y_train, i)
        scores[i] = score
        print(f'{i}: {score:.4f}')
    k = min(scores, key=scores.get)
    print(f'Best chosen k = {k}')

    kwnn = KWNN(k, X_train, Y_train)
    predictions = kwnn.predict(X_test)
    mistakes = 0
    for i in range(len(Y_test)):
        print(f"Predicted: {predictions[i]}, Actual: {Y_test[i]}")
        if predictions[i] != Y_test[i]:
            mistakes += 1
    print(f"Mistakes were made: {mistakes} on selection size:
{len(Y_test)}")
    print(f"Accuracy: {100 - (mistakes / len(Y_test) * 100):.3}%")
```

```
if __name__ == '__main__':
    exit(main())
```