Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Лабораторная работа №2

по дисциплине «Машинное обучение»

Выплолнил студент
гр. 33534/5 Стойкоски Н.С.

Руководитель И.А. Селин

Санкт-Петербург
2019 г.

Оглавление

Постановка задачи

1. Исследуйте, как объем обучающей выборки и количество тестовых данных, влияет на точность классификации или на вероятность ошибочной классификации в датасетах про крестики-нолики и о спаме e-mail сообщений.

2. Постройте классификатор для обучающего множества Glass (glass.csv), данные которого характеризуются 10-ю признаками:

1. Id number: 1 to 214; 2. RI: показатель преломления; 3. Na: сода (процент содержания в соответствующем оксиде); 4. Mg; 5. Al; 6. Si; 7. K; 8. Ca; 9. Ba; 10. Fe.

Классы характеризуют тип стекла:

(1) окна зданий, плавильная обработка
(2) окна зданий, не плавильная обработка
(3) автомобильные окна, плавильная обработка
(4) автомобильные окна, не плавильная обработка (нет в базе)
(5) контейнеры
(6) посуда
(7) фары

Посмотрите заголовки признаков и классов. Перед построением классификатора необходимо также удалить первый признак Id number, который не несет никакой информационной нагрузки.

Постройте графики зависимости ошибки классификации от значения n_neighbors.

Определите подходящие метрики из класса DistanceMetric и исследуйте, как тип метрики расстояния влияет на точность классификации.

Определите, к какому типу стекла относится экземпляр с характеристиками

RI =1.516 Na =11.7 Mg =1.01 Al =1.19 Si =72.59 K=0.43 Ca =11.44 Ba =0.02 Fe =0.1

Определите, какой из признаков оказывает наименьшее влияние на определение класса путем последовательного исключения каждого признака.

3. Для построения классификатора используйте заранее сгенерированные обучающие и тестовые выборки, хранящиеся в файлах svmdata4.txt, svmdata4test.txt. Найдите оптимальное значение n_neighbors, обеспечивающее наименьшую ошибку классификации. Посмотрите, как выглядят данные на графике.

Ход работы

1. Была создана функция которая на входе принимает наборы признаков и классов, а так же коэффициент - отношение объёма обучающей выборки к общее число данных. С использованием функции sklearn.model_selection.train_test_split() входные массивы случайным образом перемешиваются, и разделяются на тестовой

и обучающей выборки в соответствие с входного коэффициента. Строится классификатор К-ближайших соседей (Sklearn.neighbors.KNeighborsClassifier) с параметрами n_neighbors=5, metric='manhattan', классифицируются выборки и на выходе подаются точность классификации на тестовой и на обучающей выборки соответственно. Эта функция далее используется для двух разных наборах данных (крестики-нолики, спам) с многократное варирование объёма обучающей выборки. Результаты выводятся в виде графиков, построени с использованием библиотеки matplotlib.

2. На обучающего множества Glass, при использование разных метрики из класса DistanceMetric (euclidean, manhattan, chebyshev), и при многократного варирования параметра n_neigbors, строются требуемые графики зависимостей, при этом точность классификации оценивается с использованием метода перекрестного контроля (sklearn.model_selection.cross_val_score).

Исходя из построенных графиков, на данном наборе найболее лучшая метрика расстояния – это метрика типа 'manhattan', с лучших результатов при n_neighbors=1, 5, 14.

Данный экземпляр:

RI =1.516 Na =11.7 Mg =1.01 Al =1.19 Si =72.59 K=0.43 Ca =11.44 Ba =0.02 Fe =0.1
```
[5]
[[0.  0.2  0.  0.8  0.  0. ]]
```
Отклассифицирован к типу стекла – (5) контейнеры с вероятность 0.80.

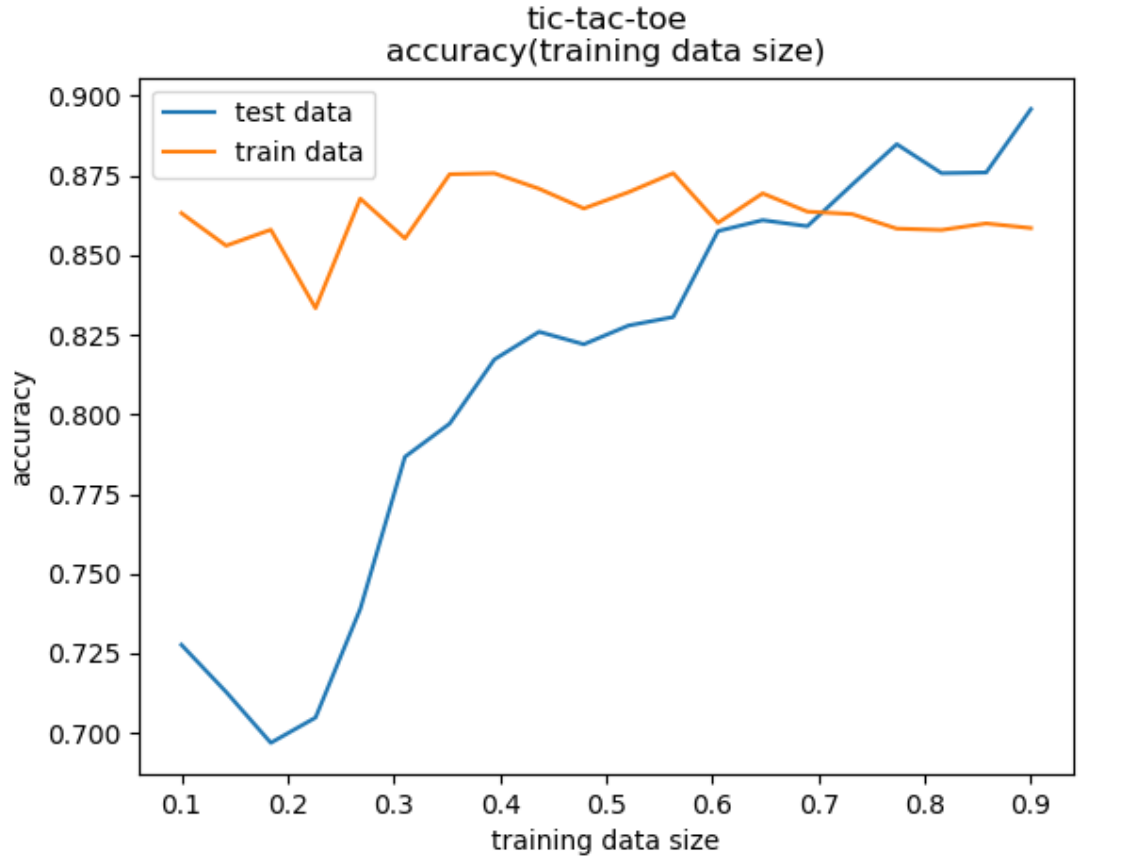
При последовательного исключения каждого из признаков, было найдено их влияние на определение класса (разница в точности классификации с использованием всех признаков для обучения, и с исключением одного из признаков)

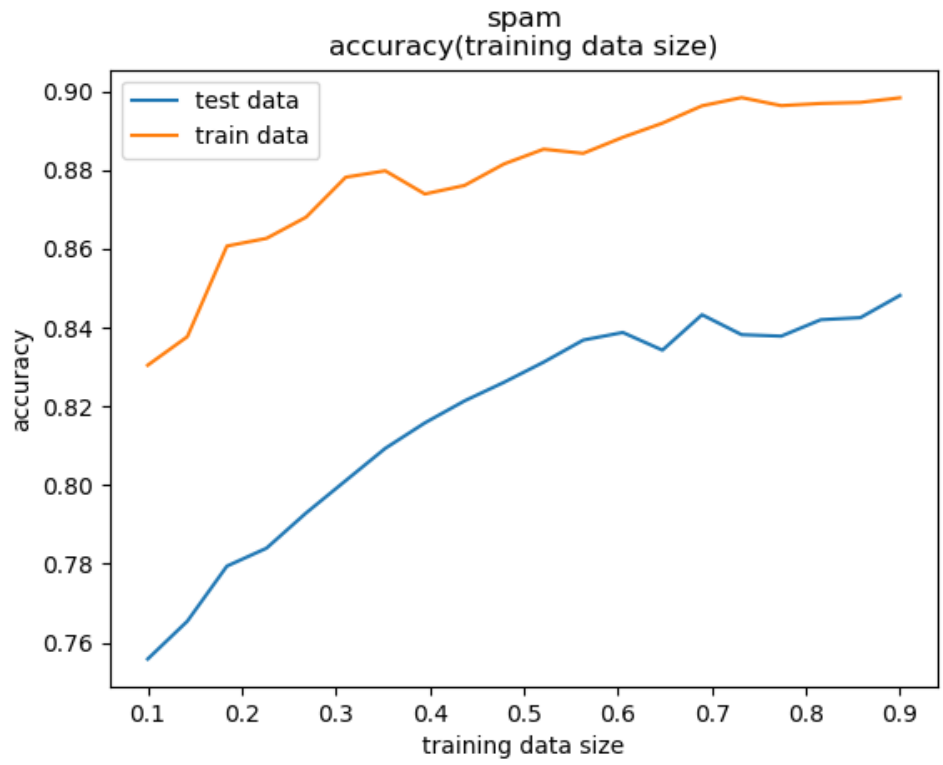
```
Change in accuracy by excluding column:
RI :  0.0
Na :  -0.05617101245008216
Mg :  -0.05498053625960597
Al :  0.0007788851974898092
Si :  -0.041630591630591574
K :  -0.015228363367898234
Ca :  -0.055855733413872954
Ba :  0.004793281653746817
Fe :  0.0035714285714284477
```

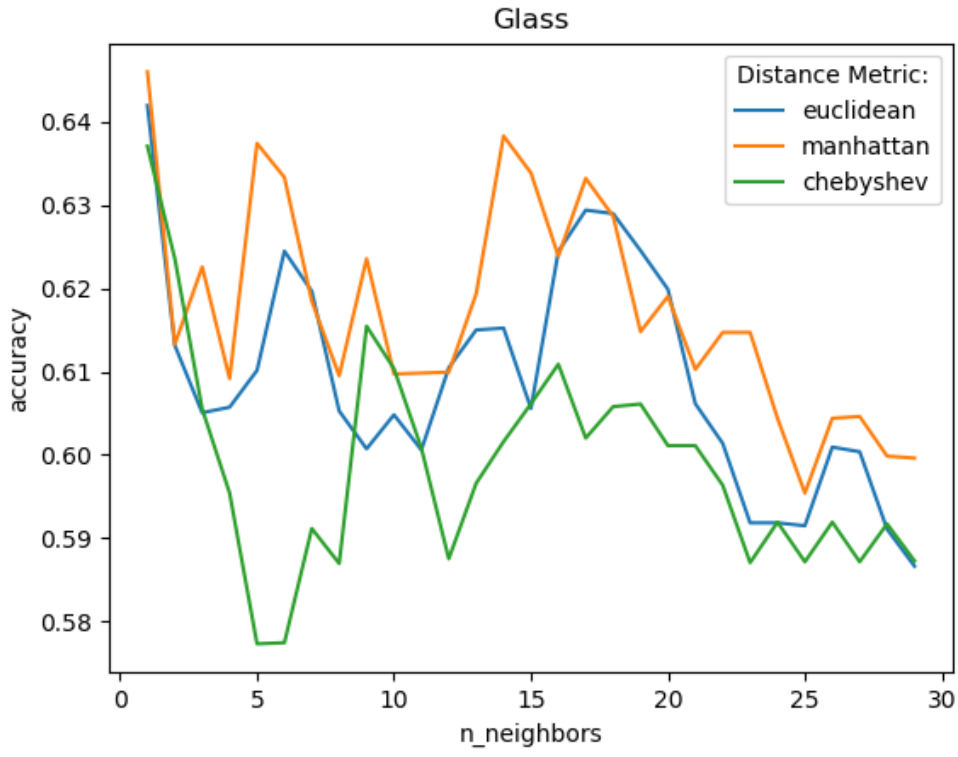
Следовательно можно сделать вывод что признак RI никак не влияет на точность классификации.

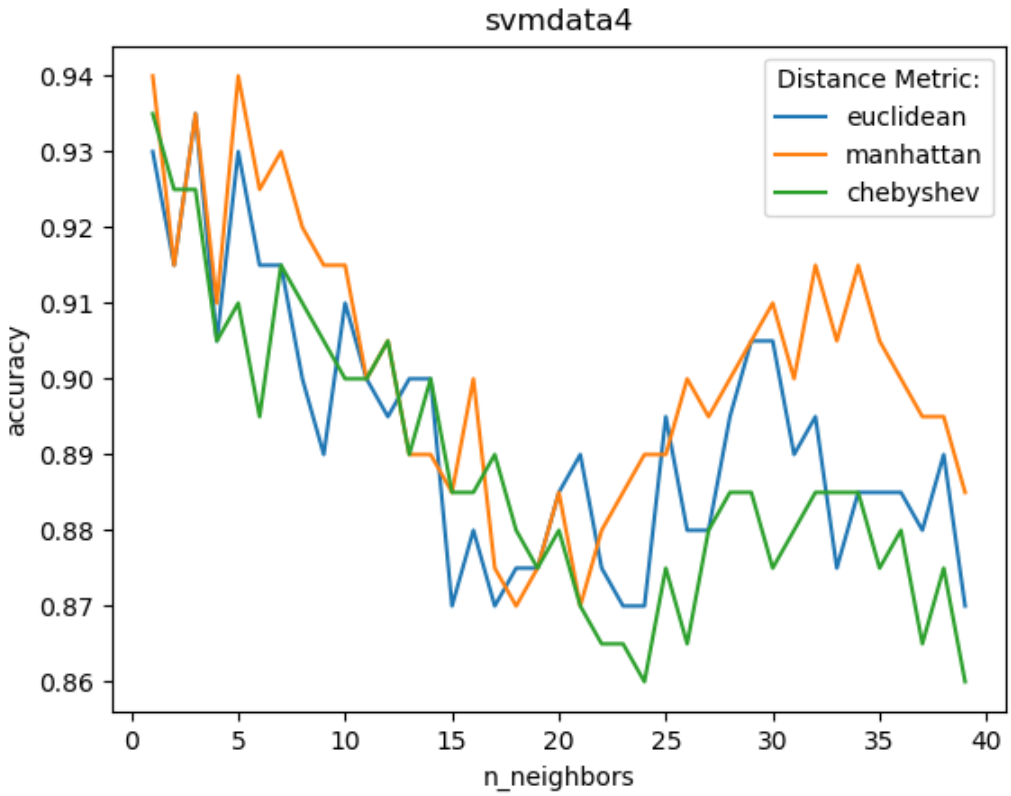
3. На наборах svmdata4.txt, svmdata4test.txt при построение график зависимостей точности от количество соседей, и разные метрики расстояния, найдено оптимальное значение n_neigbors = 1, при метрика расстояния типа ‘manhattan’. Построен график содержащий раскращенных точек в зависимости от их класс (зеленые и красные) при этом крестики отображают неправильно отклассифицированные точки (красный крестик – точка действительно является красная, но неправильно откласифицирована как зеленая).

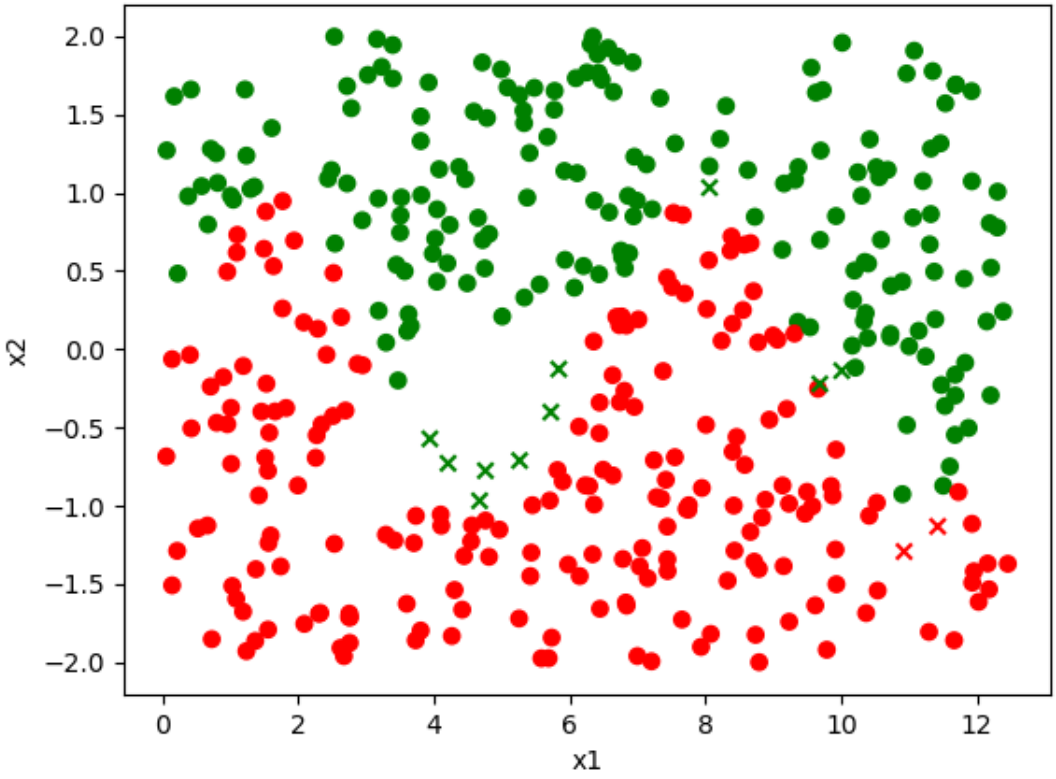
Результаты




spam
accuracy(training data size)
test data
train data
accuracy
training data size
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0.76
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9


Glass
Distance Metric:
euclidean
manhattan
chebyshev
accuracy
n_neighbors
0.64
0.63
0.62
0.61
0.60
0.59
0.58
0
5
10
15
20
25
30


svmdata4
Distance Metric:
euclidean
manhattan
chebyshev
accuracy
n_neighbors
0.94
0.93
0.92
0.91
0.90
0.89
0.88
0.87
0.86
0
5
10
15
20
25
30
35
40


x2
x1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
2
4
6
8
10
12

Вывод

В ходе данной лабораторной работы было исследовано как обьем обучюшей выборки и количество тестовых данных, влияет на точность классификации на два разных датасетах.
Были построени графики зависимости точности классификации от количества рассматриваемых соседей при использованием разных метрики расстояния на наборе glass.csv и были найдени влияния признаков на определение класса путем последовательного исключюения каждого из признаков при обучений.
Так же был построен классификатор на наборе svmdata4.txt, и в свою очередь проведено тестирование на наборе svmdata4test.txt, при этом все данные и результаты классификации были представлены представлены на точечном графике.

Текст программы

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn import preprocessing
from sklearn import metrics
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np

def calculate_accuracy(features, targets, train_size):
    test_size = 1 - train_size
    x_train, x_test, y_train, y_test = \
        train_test_split(features, targets, test_size=test_size, random_state=1)

    neigh = KNeighborsClassifier(n_neighbors=5, n_jobs=-1, metric='manhattan')
    neigh.fit(x_train, y_train)

    return (metrics.accuracy_score(y_test, neigh.predict(x_test)),
            metrics.accuracy_score(y_train, neigh.predict(x_train)))

def make_plot(ratios, accuracies, title):
    plt.figure()
    plt.plot(ratios, [acc[0] for acc in accuracies], label='test data')
    plt.plot(ratios, [acc[1] for acc in accuracies], label='train data')
    plt.xlabel('training data size')
    plt.ylabel('accuracy')
    plt.title(f'{title}\naccuracy(training data size)')
    plt.legend()
    plt.savefig(f'{title}.png')

def tic_tac_toe():
    features, targets = [], []
```

```
    with open("Tic_tac_toe.txt") as inp:
        for line in inp:
            features.append(line.split(',')[0:9])
            targets.append(line.split(',')[9].strip())

    le = preprocessing.LabelEncoder()
    features_encoded = [le.fit_transform(sample) for sample in features]
    targets_encoded = le.fit_transform(targets)

    ratios = np.linspace(0.1, 0.9, 20)
    accuracies = [calculate_accuracy(features_encoded, targets_encoded, ratio) for ratio
in ratios]
    make_plot(ratios, accuracies, 'tic-tac-toe')

def spam():
    df = pd.read_csv('spam.csv', sep=',')
    features = df.iloc[:, 1:58].values
    targets = df['type'].values
    targets_encoded = preprocessing.LabelEncoder().fit_transform(targets)

    ratios = np.linspace(0.1, 0.9, 20)
    accuracies = [calculate_accuracy(features, targets_encoded, ratio) for ratio in
ratios]
    make_plot(ratios, accuracies, 'spam')

def glass():
    df = pd.read_csv('glass.csv', sep=',')
    features = df.iloc[:, 2:-1].values
    targets = df['Type'].values

    plt.figure()
    metrics = ['euclidean', 'manhattan', 'chebyshev'] #, 'minkowski']
    max_neighbors = 30
    for metric in metrics:
        scores = []
        for n_neigbors in range (1, max_neighbors):
            neigh = KNeighborsClassifier(n_neighbors=n_neigbors, n_jobs=1, metric=metric)
            scores.append(cross_val_score(neigh, features, targets, cv=5,
n_jobs=1).mean())
        plt.plot(range(1, max_neighbors, 1), scores, label=metric)
    plt.title('Glass')
    plt.xlabel('n_neighbors')
    plt.ylabel('accuracy')
    plt.legend(title='Distance Metric:');


    neigh = KNeighborsClassifier(n_neighbors=5, n_jobs=1, metric='manhattan')
    neigh.fit(features, targets)
    print('prediction for RI=1.516 Na=11.7 Mg=1.01 Al=1.19 Si=72.59 K=0.43 Ca=11.44
Ba=0.02 Fe=0.1:')
    sample = [1.516, 11.7, 1.01, 1.19, 72.59, 0.43, 11.44, 0.02, 0.1]
    print(neigh.predict([sample]))
    print(neigh.predict_proba([sample]))


    neigh = KNeighborsClassifier(n_neighbors=5, n_jobs=1, metric='manhattan')
    accuracy = cross_val_score(neigh, features, targets, cv=5, n_jobs=1).mean()
    print('Change in accuracy by excluding column:')
```

```
    columns = ['RI','Na','Mg','Al','Si','K','Ca','Ba','Fe']
    for col in columns:
        features = df.sub(df[col], axis=0).iloc[:, 2:-1].values
        neigh = KNeighborsClassifier(n_neighbors=5, n_jobs=1, metric='manhattan')
        new_accuracy = cross_val_score(neigh, features, targets, cv=5, n_jobs=1).mean()
        print(col,': ', new_accuracy - accuracy)

def svmdata4():
    df_train = pd.read_csv('svmdata4.txt', delim_whitespace=True)
    df_test = pd.read_csv('svmdata4test.txt', delim_whitespace=True)

    dict = {'green': 1, 'red': 2}
    x_train = df_train[['X1', 'X2']].values
    x_test = df_test[['X1', 'X2']].values
    y_train = [dict[val] for val in df_train['Colors'].values]
    y_test = [dict[val] for val in df_test['Colors'].values]

    plt.figure()
    knn_metrics = ['euclidean', 'manhattan', 'chebyshev']
    max_neighbors = 40
    for metric in knn_metrics:
        scores = []
        for n_neigbors in range (1, max_neighbors):
            neigh = KNeighborsClassifier(n_neighbors=n_neigbors, n_jobs=1, metric=metric)
            neigh.fit(x_train, y_train)
            scores.append(metrics.accuracy_score(y_test, neigh.predict(x_test)))
        plt.plot(range(1, max_neighbors, 1), scores, label = metric)
    plt.title('svmdata4')
    plt.xlabel('n_neighbors')
    plt.ylabel('accuracy')
    plt.legend(title='Distance Metric:')


    neigh = KNeighborsClassifier(n_neighbors=1, n_jobs=1, metric='manhattan')
    neigh.fit(x_train, y_train)
    x_merged, y_merged = np.array([*x_train, *x_test]), np.array([*y_train, *y_test])
    pred_correct = np.isclose(neigh.predict(x_merged), y_merged)

    inv_dict = {v: k for k, v in dict.items()}
    x1, x2 = np.array([v[0] for v in x_merged]), np.array([v[1] for v in x_merged])
    plt.figure()
    plt.scatter(x1[pred_correct==True], x2[pred_correct==True],
                color=[inv_dict[v] for v in y_merged[pred_correct==True]], marker='o')
    plt.scatter(x1[pred_correct==False], x2[pred_correct==False],
                color=[inv_dict[v] for v in y_merged[pred_correct==False]], marker='x')
    plt.xlabel('x1')
    plt.ylabel('x2')

tic_tac_toe()
spam()
glass()
svmdata4()
plt.show()
```