تمرین HW13 هوش – محمد اصولیان 99521073

train کردن با داده های خام و گزارش نتایج

در این قسمت 80 درصد داده ها از iris به صورت رندوم برای تست انتخاب شد. از خود داده ها بدون تغییری برای ترین کردن استفاده شد. برای classification از همان فرمول گفته شده در اسلاید 25 استفاده شد

```
def classify(feats, record):
    probs = {}
    for cl in feats[0][0].keys():
        probs[cl] = 1
        for i in range(4):
            probs[cl] *= feats[i][record[i]][cl]
    return max(probs, key=probs.get)
```

```
train with raw values:
Correct:  26
Total:  30
```

نتایج به دست آمده تقریبا درست حدس میزند و مناسب است. اما در قدم بعد سعی کردم که این نتاج را بیشتر بهبود دهم.

train کردن با داده های تغییر داده شده و گزارش نتایج

در این قسمت به جای استفاده از داده های خام، داده ها را رند کردم و تمام مقادیر را به اعداد صحیح 0 تا 10 تبدیل کردم.

```
def round_records(records):
    rounded_records = []
    for record in records:
        rounded_record = []
        for value in record[:4]:
            rounded_record.append(round(value))
        rounded_record.append(record[4])
        rounded_records.append(rounded_record)
    return rounded_records
```

پس از انجام این عمل نتایج دقیق تر شد:

```
train with rounded values:
Correct:  28
Total:  30
```

فکر میکنم علت بهتر شدن نتایج این است که زمانی که اعداد را رند نمیکنیم، تعداد حالت ها زیاد میشود و از آنجایی که دیتابیس خیلی بزرگ نیست، تعداد حالت ها کم میشود و مدل به خوبی train نمیشود.

اما وقتی که اعداد را رند میکنیم تعداد دسته ها را به تنها 10 دسته کاهش میدهیم. با انجام این کار از هر دسته به اندازه کافی داده وجود خواهد داشت و مدل پیشبینی های بهتری انجام میدهد.

اجرای توابع اماده از کتاب خانه scikit-learn و مقایسه نتایج

دستورات مشخص شده در سایت کتابخانه را به ترتیب وارد کردم و نتایج به این صورت بود:

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.naive_bayes import GaussianNB
>>> X, y = load_iris(return_X_y=True)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_si
>>> gnb = GaussianNB()
>>> y_pred = gnb.fit(X_train, y_train).predict(X_test)
>>> print("Number of mislabeled points out of a total %d points : %d'
...       % (X_test.shape[0], (y_test != y_pred).sum()))
Number of mislabeled points out of a total 75 points : 4
```

نتجه 4 خطا از 75 رکورد بود. از نتایج من در حالت دوم که 2 خطا از 30 رکورد بود بهتر است اما تفاوت خیلی زیادی ندارد و نشان میدهد که مدل ما هم توانسه تا حد خوبی به تابع موجود در کتابخانه scikit-learn نزدیک شود.