گزارش ۷: مقایسه ی روش های مختلف طبقه بندی روی یک مسئله benchmark

نگارش

نیکا شهابی۹۷۱۳۰۲۳

استاد

دکتر مهدی قطعی

خرداد ۱۴۰۰

لینک پروژه در گیتاب:

https://github.com/nikashahabi/supervised-classification-on-iris-dataset

توضیح صورت مسئله:

برای یک مجموعه داده‌ی استاندارد که لیبل دارند الگوریتم های classification و مدل های مختلف را train کنید و مدل های خود را ارزیابی کنید. در این پروژه از مجموعه داده ی iris استفاده شد و supervised learning توسط مدل‌های زیر انجام شد.

1. SVM
2. Decision Tree
3. K-nearest neighbors
4. Logistic Regression

توضیح درمورد دیتاست استفاده شده و visualization آن:

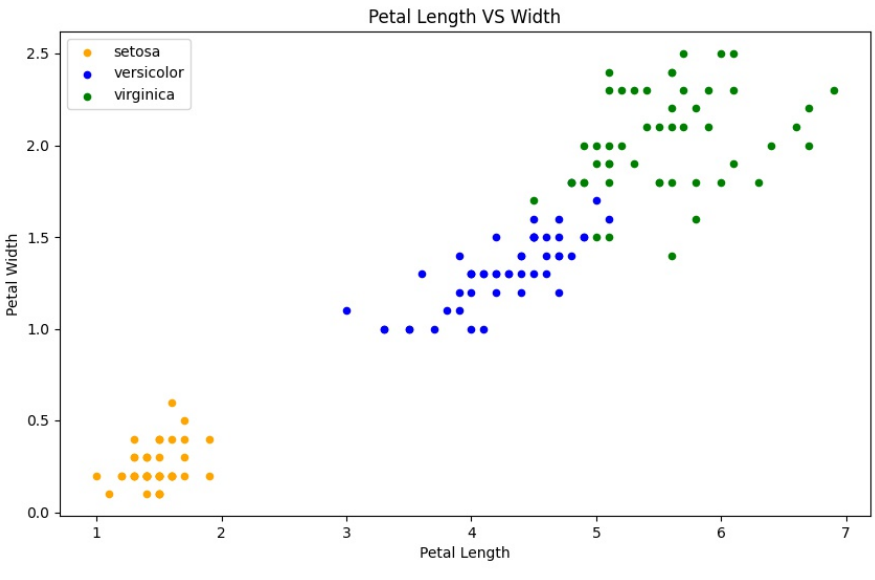
در این پروژه از دیتاست iris استفاده شده است. این دیتاست ۱۵۰ سطر دارد که هر کدام ۵ یا ۶ ستون دارند. (بستگی دارد دیتاست از کجا دانلود شود. ممکن است ستون id حذف شده باشد و ۵ سطر موجود باشد.) هر یک از این ۱۵۰ سطر ویژگی های یک گل و گونه ی را نشان میدهند. یک گل ۴ ویژگی دارد و جزو یکی از سه دسته ی موجود در دیتابیس(setosa, versicolor, virginica) است. هر یک از این گونه/دسته ها ۵۰ تا نمونه در دیتابیس دارند. یکی از گونه ها به طور خطی جدایی پذیر از دو گونه ی دیگر است ولی دو گونه ی دیگر خطی جدایی پذیر نیستند. به علاوه هیچ null ای در جدول وجود ندارد و خیالمان راحت است که همه‌ی خانه‌ها پر هستند. با دستورات زیر اطلاعاتی از جدول کسب میکنیم:

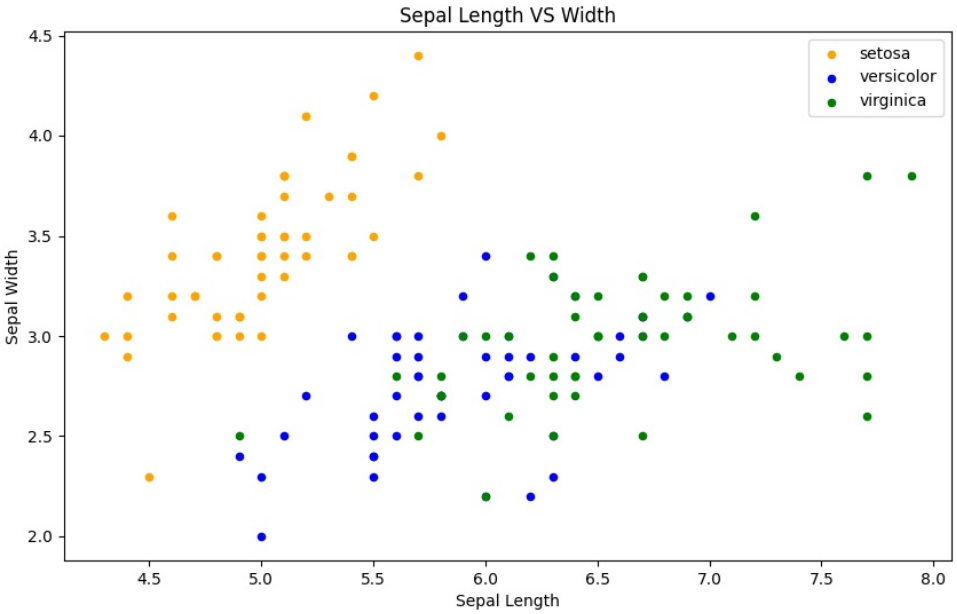
```
print(iris.info())
print(iris['species'].value_counts())
print(iris.describe())
```

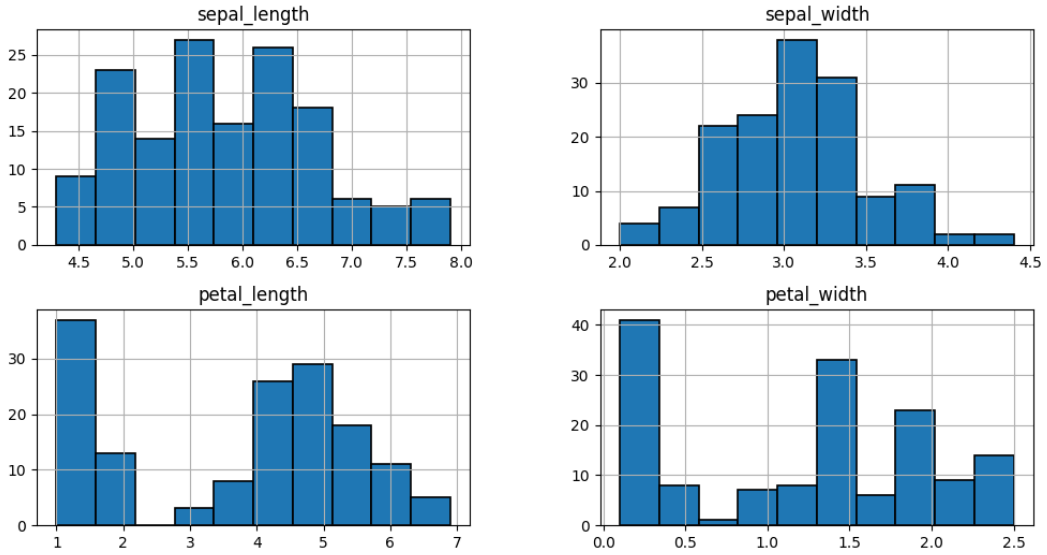
```
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
```

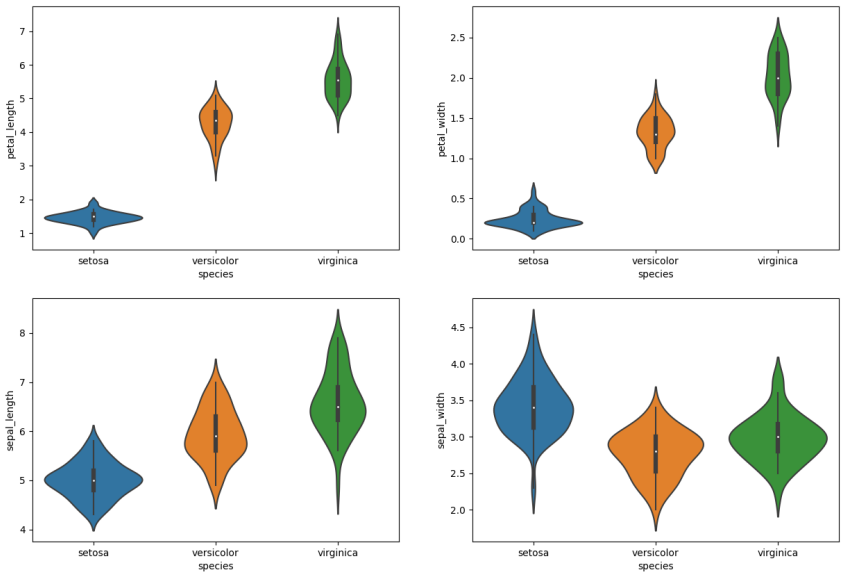
```
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

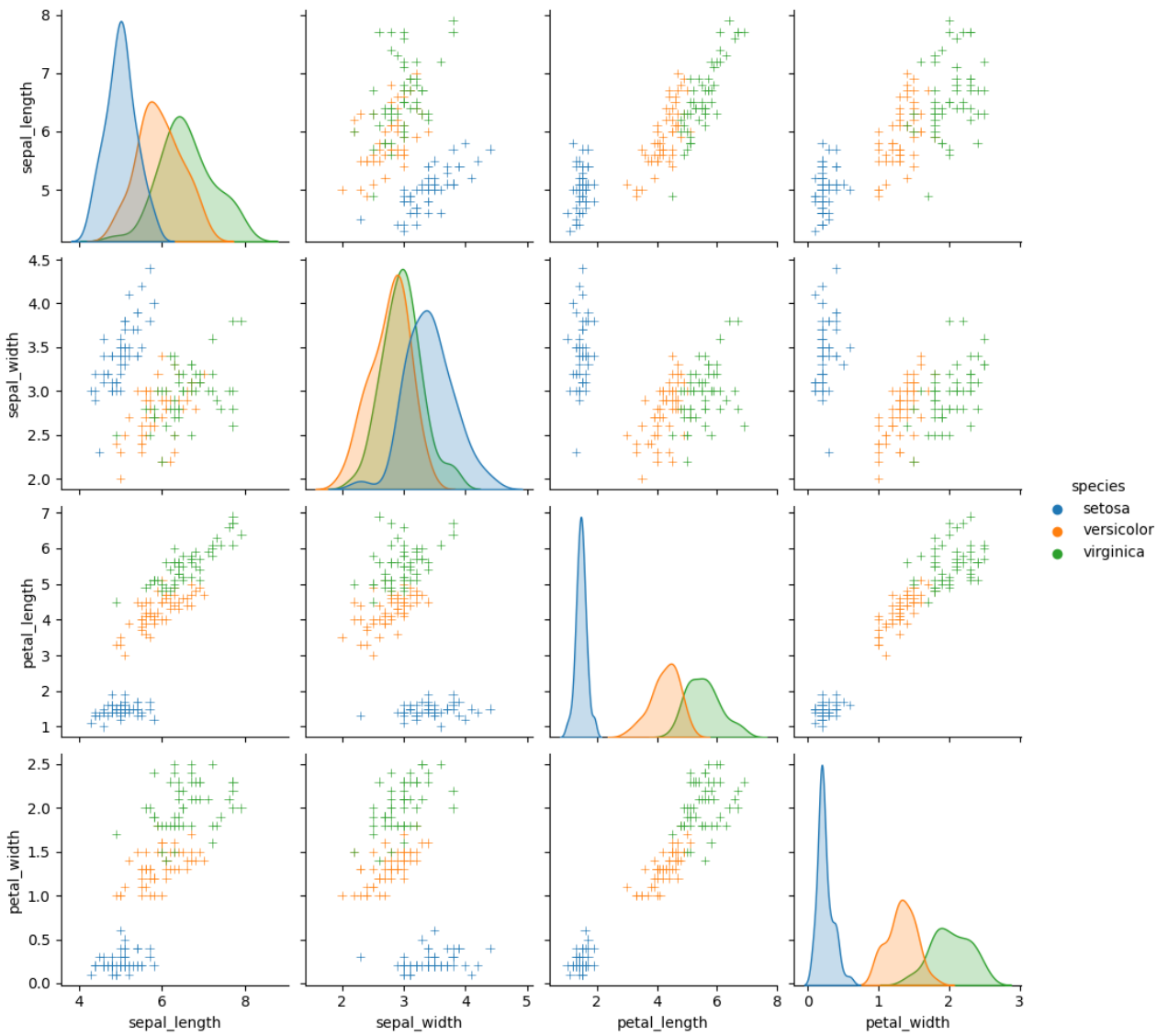
حال به visualization داده ها برای استخراج اطلاعات میپردازیم. تابع استفاده شده در اینجا plotData(iris)است. ما فقط خروجی figure ها را در اینجا می‌آوریم و به تحلیل آن ها میپردازیم.




Sepal Length VS Width
setosa
versicolor
virginica
Sepal Width
Sepal Length
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


sepal_length
sepal_width
petal_length
petal_width


petal_length
petal_width
sepal_length
sepal_width
setosa
versicolor
virginica
species


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica

از شکل آخر متوجه میشویم که با ویژگی petal (داشتن طول و عرض Petel) سه گونه تقریبا خطی جدایی پذیر هستند. پس احتمال میرود که logistic regression با داشتن این دو ویژگی خوب کار کند. (در شکل اول به طور واضح تر این خطی جدایی پذیر بودن مشخص است) به علاوه تحلیل های واضحی را میتوانیم از روی شکل های دیگر انجام دهیم. مثلا اینکه بیشتر گل های یک گونه چهpetal length ای دارند...

توضیح درمورد مدل های استفاده شده:

اینکه train و test با هم دیگر اشتراک داشته باشند کار درستی نیست چون از ویژگی های خود test برای train کردن مدل استفاده شده و ممکن است باعث overfit شدن و واریانس زیاد بشود. پس در ابتدا داده را به دو قسمت train set و test set تقسیم کردیم. داده ی تست در این پروژه ۳۰٪ کل داده را شامل میشود.

```
train, test = sc.train_test_split(iris, test_size=0.3)
print(train.shape, test.shape)
train_X = train[['sepal_length','sepal_width', 'petal_length', 'petal_width']]
train_y = train.species
test_X = test[['sepal_length','sepal_width', 'petal_length', 'petal_width']]
test_y = test.species
```

خروجی این تکه کد که به ترتیب train shape و test shape را نشان میدهد:

(5 ,45) (5 ,105)

همان طور که در توضیح صورت مسئله هم آورده شد در اینجا با ۴ مدل کار کردیم.

یکی یکی آن ها را توضیح میدهیم و میگوییم برای هر یک از آن ها چه کتاب خانه ای استفاده شده است.

1. SVM

در این روش ابرصفحه ای که ارائه میشود به طور شهودی ماکسیمم فاصله را از داده های نزدیک به خود دارد. (maximum margin separator) به علاوه SVM قادر است که داده هایی که linearly seperable نیستند را به بعدهای بالاتر ببرد و احتمال اینکه در آن بعد ها خطی جدا شوند بیشتر است.

در این پروژه برای پیاده سازی SVM از تابع زیر استفاده شده.

```
SVM(train_X, train_y, test_X, test_Y):
```

که خود از کتابخانه و مدل زیر استفاده کرده است:

```
from sklearn import svm
```

2. Decision Tree

از درخت تصمیم گیری که داده ها را بر حسب feature ها وارد node های زیرین میکند در مسئله های طبقه بندی استفاده میشود. در این درخت اینکه از چه feature ای در ابتدا برای قسمت کردن درخت استفاده شود مسئله است. همچنین اینکه terminology برای بیشتر ادامه ندادن درخت هم مهم است. به هر حال در این پروژه از توابع آماده‌ی کتاب خانه ها استفاده شده.

```
decisionTree(train_X, train_y, test_X, test_Y):
```

```
from sklearn.tree import DecisionTreeClassifier
```

3. K-nearest Neighbors

در این مدل این طور نیست که صرفا parameter هایی train بشوند و دیگر با خود داده کاری نداشته باشیم. بلکه به تعدادی (n) تا از نزدیک ترین همسایه های یک داده نگاه میشود و بر حسب رای اکثریت آن ها طبقه بندی برای آن داده مشخص میشود. برای اینکه n بهینه را پیدا کنیم ابتدا تابع

```
testkNearestNeighbors(train_X, train_y, test_X, test_y, nmax):
```

را صدا میزنیم تا از ۱ تا nmax بهترین n را پیدا کنیم. در بخش خروجی کد، نمودار کشیده شده توسط این تابع آورده شده است که از روی آن n بهینه مشخص است. سپس n بهینه به تابع زیر داده میشود تا الگوریتم K-nearest neighbors پیاده شود.

```
kNearestNeighbors(train_X, train_y, test_X, test_y, k=3):
```

```
from sklearn.neighbors import KNeighborsClassifier
```

4. Logistic Regression

یک شبکه ی عصبی ساده است که ترکیبی از یک تابع خطی و یک تابع sigmoid است و پارامتری به اسم w را train میکند. logistic regression به طور کلی برای binary classification است. ولی با تغییر هایی درون آن میتوان آن را روی دیگر مسائل classification استفاده کرد. یکی از تغییرهای ممکن استفاده از ایده ی one-vs-rest است که هر دفعه یک کلاس را در مقابل بقیه میسنجد. در توضیحات کتاب خانه ی استفاده شده میبینیم استفاده از این option وجود دارد:

```
Logistic Regression (aka logit, MaxEnt) classifier.
```

```
In the multiclass case, the training algorithm uses the one-vs-rest (OvR)
scheme if the 'multi_class' option is set to 'ovr', and uses the
cross-entropy loss if the 'multi_class' option is set to 'multinomial'.
```

کد استفاده شده برای پیاده سازی این مدل در تابع زیر وجود دارد.

```
logisticRegression(train_X, train_y, test_X, test_y):
```

```
from sklearn.linear_model import LogisticRegression
```

توضیح درباره‌ی نحوه‌ی ارزیابی مدل‌های استفاده شده:

برای ارزیابی مدل ها از

```
metrics.accuracy_score(prediction, test_y)
```

استفاده شده است که prediction بردار تولید شده توسط مدل train شده برای test_X است.

```
from sklearn import metrics
```

این تابع درصد داده های test ای که به طور کاملا درست پیش بینی شده‌اند را پیدا میکند.

در ادامه این درصد برای مدل های گفته شده آورده میشود و تحلیل روی آن ها انجام میشود.

توضیح توابع مسئله:

در بین کد به صورت کامنت آورده شده است.

نمونه خروجی مسئله:

ورودی:

```
if __name__ == "__main__":
    iris = loadData()
    train, test = sc.train_test_split(iris, test_size=0.3)
    train_X = train[['sepal_length','sepal_width', 'petal_length',
'petal_width']]
    train_y = train.species
    test_X = test[['sepal_length','sepal_width', 'petal_length',
'petal_width']]
    test_y = test.species
    SVM(train_X, train_y, test_X, test_y)
    decisionTree(train_X, train_y, test_X, test_y)
    n = testkNearestNeighbors(train_X, train_y, test_X, test_y, nmax=25)
    kNearestNeighbors(train_X, train_y, test_X, test_y, n)
    logisticRegression(train_X, train_y, test_X, test_y)
```

خروجی: علاوه بر ارزیابی مدل پیش بینی مدل برای هر یک از داده های test را هم نشان میدهد. قسمت های مهم تر هایلایت شده است.

:SVM

The accuracy of the SVM on train set is 0.9619047619047619

The accuracy of the SVM on the test set is: 0.9111111111111111

[1.6 ,4.5 ,3.4 ,6.0]

versicolor

prediction: versicolor

[2.1 ,5.6 ,2.8 ,6.4]

virginica

prediction: virginica

[0.4 ,1.6 ,3.4 ,5.0]

setosa

prediction: setosa

[1.6 ,5.8 ,3.0 ,7.2]

virginica

prediction: virginica

[1.3 ,4.3 ,2.9 ,6.4]

versicolor

prediction: versicolor

[0.2 ,1.6 ,3.0 ,5.0]

setosa

prediction: setosa

[0.2 ,1.6 ,3.4 ,4.8]

setosa

prediction: setosa

[2.0 ,5.1 ,3.2 ,6.5]

virginica

prediction: virginica

[1.0 ,4.0 ,2.2 ,6.0]

versicolor

prediction: versicolor

[2.3 ,5.1 ,3.1 ,6.9]

virginica

prediction: virginica

[2.3 ,5.4 ,3.4 ,6.2]

virginica

prediction: virginica

[1.7 ,4.5 ,2.5 ,4.9]

virginica

prediction: versicolor

[2.2 ,5.8 ,3.0 ,6.5]

virginica

prediction: virginica

[1.1 ,3.9 ,2.5 ,5.6]

versicolor

prediction: versicolor

[0.2 ,1.2 ,4.0 ,5.8]

setosa

prediction: setosa

[2.1 ,6.6 ,3.0 ,7.6]

virginica

prediction: virginica

[1.0 ,3.7 ,2.4 ,5.5]

versicolor

prediction: versicolor

[1.5 ,4.2 ,3.0 ,5.9]

versicolor

prediction: versicolor

[0.4 ,1.5 ,4.4 ,5.7]

setosa

prediction: setosa

[0.2 ,1.4 ,3.2 ,4.6]

setosa

prediction: setosa

[0.2 ,1.6 ,3.2 ,4.7]

setosa

prediction: setosa

[1.8 ,6.0 ,3.2 ,7.2]

virginica

prediction: virginica

[2.3 ,5.2 ,3.0 ,6.7]

virginica

prediction: virginica

[1.3 ,4.0 ,2.3 ,5.5]

versicolor

prediction: versicolor

[1.8 ,4.8 ,3.0 ,6.0]

virginica

prediction: versicolor

[1.3 ,4.2 ,2.7 ,5.6]

versicolor

prediction: versicolor

[0.1 ,1.4 ,3.0 ,4.8]

setosa

prediction: setosa

[1.9 ,5.1 ,2.7 ,5.8]

virginica

prediction: virginica

[1.0 ,4.1 ,2.7 ,5.8]

versicolor

prediction: versicolor

[1.3 ,4.3 ,2.9 ,6.2]

versicolor

prediction: versicolor

[0.2 ,1.4 ,3.5 ,5.1]

setosa

prediction: setosa

[1.4 ,4.4 ,3.0 ,6.6]

versicolor

prediction: versicolor

[1.4 ,3.9 ,2.7 ,5.2]

versicolor

prediction: versicolor

[0.2 ,1.5 ,3.7 ,5.4]

setosa

prediction: setosa

[1.2 ,4.7 ,2.8 ,6.1]

versicolor

prediction: versicolor

[1.8 ,4.8 ,2.8 ,6.2]

virginica

prediction: versicolor

[0.4 ,1.7 ,3.9 ,5.4]

setosa

prediction: setosa

[0.4 ,1.3 ,3.9 ,5.4]

setosa

prediction: setosa

[0.2 ,1.5 ,3.4 ,5.0]

setosa

prediction: setosa

[1.5 ,4.9 ,2.5 ,6.3]

versicolor

prediction: versicolor

[1.0 ,3.5 ,2.6 ,5.7]

versicolor

prediction: versicolor

[1.8 ,4.9 ,2.7 ,6.3]

virginica

prediction: versicolor

[1.3 ,4.1 ,2.8 ,5.7]

versicolor

prediction: versicolor

[2.5 ,6.1 ,3.6 ,7.2]

virginica

prediction: virginica

[1.4 ,4.7 ,2.9 ,6.1]

versicolor

prediction: versicolor

:DECISION TREE

The accuracy of the Decision Tree on train set is 1.0

The accuracy of the Decision Tree on test set is 0.8888888888888888

[1.6 ,4.5 ,3.4 ,6.0]

versicolor

prediction: versicolor

[2.1 ,5.6 ,2.8 ,6.4]

virginica

prediction: virginica

[0.4 ,1.6 ,3.4 ,5.0]

setosa

prediction: setosa

[1.6 ,5.8 ,3.0 ,7.2]

virginica

prediction: versicolor

[1.3 ,4.3 ,2.9 ,6.4]

versicolor

prediction: versicolor

[0.2 ,1.6 ,3.0 ,5.0]

setosa

prediction: setosa

[0.2 ,1.6 ,3.4 ,4.8]

setosa

prediction: setosa

[2.0 ,5.1 ,3.2 ,6.5]

virginica

prediction: virginica

[1.0 ,4.0 ,2.2 ,6.0]

versicolor

prediction: versicolor

[2.3 ,5.1 ,3.1 ,6.9]

virginica

prediction: virginica

[2.3 ,5.4 ,3.4 ,6.2]

virginica

prediction: virginica

[1.7 ,4.5 ,2.5 ,4.9]

virginica

prediction: versicolor

[2.2 ,5.8 ,3.0 ,6.5]

virginica

prediction: virginica

[1.1 ,3.9 ,2.5 ,5.6]

versicolor

prediction: versicolor

[0.2 ,1.2 ,4.0 ,5.8]

setosa

prediction: setosa

[2.1 ,6.6 ,3.0 ,7.6]

virginica

prediction: virginica

[1.0 ,3.7 ,2.4 ,5.5]

versicolor

prediction: versicolor

[1.5 ,4.2 ,3.0 ,5.9]

versicolor

prediction: versicolor

[0.4 ,1.5 ,4.4 ,5.7]

setosa

prediction: setosa

[0.2 ,1.4 ,3.2 ,4.6]

setosa

prediction: setosa

[0.2 ,1.6 ,3.2 ,4.7]

setosa

prediction: setosa

[1.8 ,6.0 ,3.2 ,7.2]

virginica

prediction: virginica

[2.3 ,5.2 ,3.0 ,6.7]

virginica

prediction: virginica

[1.3 ,4.0 ,2.3 ,5.5]

versicolor

prediction: versicolor

[1.8 ,4.8 ,3.0 ,6.0]

virginica

prediction: versicolor

[1.3 ,4.2 ,2.7 ,5.6]

versicolor

prediction: versicolor

[0.1 ,1.4 ,3.0 ,4.8]

setosa

prediction: setosa

[1.9 ,5.1 ,2.7 ,5.8]

virginica

prediction: virginica

[1.0 ,4.1 ,2.7 ,5.8]

versicolor

prediction: versicolor

[1.3 ,4.3 ,2.9 ,6.2]

versicolor

prediction: versicolor

[0.2 ,1.4 ,3.5 ,5.1]

setosa

prediction: setosa

[1.4 ,4.4 ,3.0 ,6.6]

versicolor

prediction: versicolor

[1.4 ,3.9 ,2.7 ,5.2]

versicolor

prediction: versicolor

[0.2 ,1.5 ,3.7 ,5.4]

setosa

prediction: setosa

[1.2 ,4.7 ,2.8 ,6.1]

versicolor

prediction: versicolor

[1.8 ,4.8 ,2.8 ,6.2]

virginica

prediction: versicolor

[0.4 ,1.7 ,3.9 ,5.4]

setosa

prediction: setosa

[0.4 ,1.3 ,3.9 ,5.4]

setosa

prediction: setosa

[0.2 ,1.5 ,3.4 ,5.0]

setosa

prediction: setosa

[1.5 ,4.9 ,2.5 ,6.3]

versicolor

prediction: virginica

[1.0 ,3.5 ,2.6 ,5.7]

versicolor

prediction: versicolor

[1.8 ,4.9 ,2.7 ,6.3]

virginica

prediction: virginica

[1.3 ,4.1 ,2.8 ,5.7]

versicolor

prediction: versicolor

[2.5 ,6.1 ,3.6 ,7.2]

virginica

prediction: virginica

[1.4 ,4.7 ,2.9 ,6.1]

versicolor

prediction: versicolor

:K-NEAREST NEIGHBORS n= 1

The accuracy of KNN on train set is 1.0

The accuracy of KNN on test set is 0.9555555555555556

[1.6 ,4.5 ,3.4 ,6.0]

versicolor

prediction: versicolor

[2.1 ,5.6 ,2.8 ,6.4]

virginica

prediction: virginica

[0.4 ,1.6 ,3.4 ,5.0]

setosa

prediction: setosa

[1.6 ,5.8 ,3.0 ,7.2]

virginica

prediction: virginica

[1.3 ,4.3 ,2.9 ,6.4]

versicolor

prediction: versicolor

[0.2 ,1.6 ,3.0 ,5.0]

setosa

prediction: setosa

[0.2 ,1.6 ,3.4 ,4.8]

setosa

prediction: setosa

[2.0 ,5.1 ,3.2 ,6.5]

virginica

prediction: virginica

[1.0 ,4.0 ,2.2 ,6.0]

versicolor

prediction: versicolor

[2.3 ,5.1 ,3.1 ,6.9]

virginica

prediction: virginica

[2.3 ,5.4 ,3.4 ,6.2]

virginica

prediction: virginica

[1.7 ,4.5 ,2.5 ,4.9]

virginica

prediction: versicolor

[2.2 ,5.8 ,3.0 ,6.5]

virginica

prediction: virginica

[1.1 ,3.9 ,2.5 ,5.6]

versicolor

prediction: versicolor

[0.2 ,1.2 ,4.0 ,5.8]

setosa

prediction: setosa

[2.1 ,6.6 ,3.0 ,7.6]

virginica

prediction: virginica

[1.0 ,3.7 ,2.4 ,5.5]

versicolor

prediction: versicolor

[1.5 ,4.2 ,3.0 ,5.9]

versicolor

prediction: versicolor

[0.4 ,1.5 ,4.4 ,5.7]

setosa

prediction: setosa

[0.2 ,1.4 ,3.2 ,4.6]

setosa

prediction: setosa

[0.2 ,1.6 ,3.2 ,4.7]

setosa

prediction: setosa

[1.8 ,6.0 ,3.2 ,7.2]

virginica

prediction: virginica

[2.3 ,5.2 ,3.0 ,6.7]

virginica

prediction: virginica

[1.3 ,4.0 ,2.3 ,5.5]

versicolor

prediction: versicolor

[1.8 ,4.8 ,3.0 ,6.0]

virginica

prediction: virginica

[1.3 ,4.2 ,2.7 ,5.6]

versicolor

prediction: versicolor

[0.1 ,1.4 ,3.0 ,4.8]

setosa

prediction: setosa

[1.9 ,5.1 ,2.7 ,5.8]

virginica

prediction: virginica

[1.0 ,4.1 ,2.7 ,5.8]

versicolor

prediction: versicolor

[1.3 ,4.3 ,2.9 ,6.2]

versicolor

prediction: versicolor

[0.2 ,1.4 ,3.5 ,5.1]

setosa

prediction: setosa

[1.4 ,4.4 ,3.0 ,6.6]

versicolor

prediction: versicolor

[1.4 ,3.9 ,2.7 ,5.2]

versicolor

prediction: versicolor

[0.2 ,1.5 ,3.7 ,5.4]

setosa

prediction: setosa

[1.2 ,4.7 ,2.8 ,6.1]

versicolor

prediction: versicolor

[1.8 ,4.8 ,2.8 ,6.2]

virginica

prediction: virginica

[0.4 ,1.7 ,3.9 ,5.4]

setosa

prediction: setosa

[0.4 ,1.3 ,3.9 ,5.4]

setosa

prediction: setosa

[0.2 ,1.5 ,3.4 ,5.0]

setosa

prediction: setosa

[1.5 ,4.9 ,2.5 ,6.3]

versicolor

prediction: virginica

[1.0 ,3.5 ,2.6 ,5.7]

versicolor

prediction: versicolor

[1.8 ,4.9 ,2.7 ,6.3]

virginica

prediction: virginica

[1.3 ,4.1 ,2.8 ,5.7]

versicolor

prediction: versicolor

[2.5 ,6.1 ,3.6 ,7.2]

virginica

prediction: virginica

[1.4 ,4.7 ,2.9 ,6.1]

versicolor

prediction: versicolor

:LOGISTIC REGRESSION

The accuracy of the Logistic Regression on train set is 0.9904761904761905

The accuracy of the Logistic Regression on test set is 0.9333333333333333

[1.6 ,4.5 ,3.4 ,6.0]

versicolor

prediction: versicolor

[2.1 ,5.6 ,2.8 ,6.4]

virginica

prediction: virginica

[0.4 ,1.6 ,3.4 ,5.0]

setosa

prediction: setosa

[1.6 ,5.8 ,3.0 ,7.2]

virginica

prediction: virginica

[1.3 ,4.3 ,2.9 ,6.4]

versicolor

prediction: versicolor

[0.2 ,1.6 ,3.0 ,5.0]

setosa

prediction: setosa

[0.2 ,1.6 ,3.4 ,4.8]

setosa

prediction: setosa

[2.0 ,5.1 ,3.2 ,6.5]

virginica

prediction: virginica

[1.0 ,4.0 ,2.2 ,6.0]

versicolor

prediction: versicolor

[2.3 ,5.1 ,3.1 ,6.9]

virginica

prediction: virginica

[2.3 ,5.4 ,3.4 ,6.2]

virginica

prediction: virginica

[1.7 ,4.5 ,2.5 ,4.9]

virginica

prediction: versicolor

[2.2 ,5.8 ,3.0 ,6.5]

virginica

prediction: virginica

[1.1 ,3.9 ,2.5 ,5.6]

versicolor

prediction: versicolor

[0.2 ,1.2 ,4.0 ,5.8]

setosa

prediction: setosa

[2.1 ,6.6 ,3.0 ,7.6]

virginica

prediction: virginica

[1.0 ,3.7 ,2.4 ,5.5]

versicolor

prediction: versicolor

[1.5 ,4.2 ,3.0 ,5.9]

versicolor

prediction: versicolor

[0.4 ,1.5 ,4.4 ,5.7]

setosa

prediction: setosa

[0.2 ,1.4 ,3.2 ,4.6]

setosa

prediction: setosa

[0.2 ,1.6 ,3.2 ,4.7]

setosa

prediction: setosa

[1.8 ,6.0 ,3.2 ,7.2]

virginica

prediction: virginica

[2.3 ,5.2 ,3.0 ,6.7]

virginica

prediction: virginica

[1.3 ,4.0 ,2.3 ,5.5]

versicolor

prediction: versicolor

[1.8 ,4.8 ,3.0 ,6.0]

virginica

prediction: versicolor

[1.3 ,4.2 ,2.7 ,5.6]

versicolor

prediction: versicolor

[0.1 ,1.4 ,3.0 ,4.8]

setosa

prediction: setosa

[1.9 ,5.1 ,2.7 ,5.8]

virginica

prediction: virginica

[1.0 ,4.1 ,2.7 ,5.8]

versicolor

prediction: versicolor

[1.3 ,4.3 ,2.9 ,6.2]

versicolor

prediction: versicolor

[0.2 ,1.4 ,3.5 ,5.1]

setosa

prediction: setosa

[1.4 ,4.4 ,3.0 ,6.6]

versicolor

prediction: versicolor

[1.4 ,3.9 ,2.7 ,5.2]

versicolor

prediction: versicolor

[0.2 ,1.5 ,3.7 ,5.4]

setosa

prediction: setosa

[1.2 ,4.7 ,2.8 ,6.1]

versicolor

prediction: versicolor

[1.8 ,4.8 ,2.8 ,6.2]

virginica

prediction: versicolor

[0.4 ,1.7 ,3.9 ,5.4]

setosa

prediction: setosa

[0.4 ,1.3 ,3.9 ,5.4]

setosa

prediction: setosa

[0.2 ,1.5 ,3.4 ,5.0]

setosa

prediction: setosa

[1.5 ,4.9 ,2.5 ,6.3]

versicolor

prediction: versicolor

[1.0 ,3.5 ,2.6 ,5.7]

versicolor

prediction: versicolor

[1.8 ,4.9 ,2.7 ,6.3]

virginica

prediction: virginica

[1.3 ,4.1 ,2.8 ,5.7]

versicolor

prediction: versicolor

[2.5 ,6.1 ,3.6 ,7.2]

virginica

prediction: virginica

[1.4 ,4.7 ,2.9 ,6.1]

versicolor

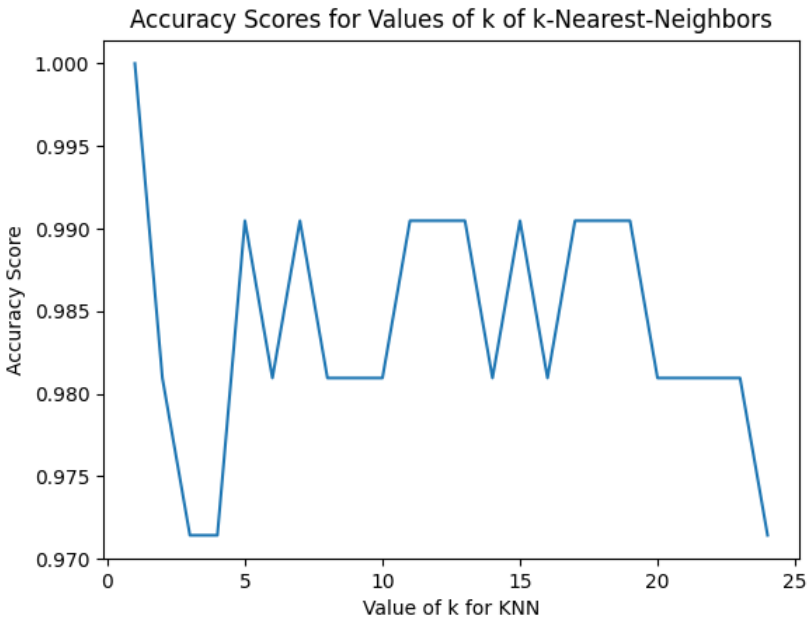
prediction: versicolor

Process finished with exit code 0

تحلیل خروجی (مقایسه‌ی ارزیابی مدل‌ها):

SVM	K- nearest neighbors	Decision Tree	Logistic Regression	Model/ accuracy
0.961904761904762	1.0	1.0	0.990476190476191	**Accuracy on train set**
0.911111111111111	0.955555555555556	0.888888888888889	0.933333333333333	**Accuracy on test set**

لازم به ذکر است که k nearest neighbors در این اجرا n=1 انتخاب شده چون:



به طور کلی همه ی مدل ها بسیار خوب نتیجه داده اند و چون روی داده ی train خوب عمل کرده اند Bias کم و چون اختلاف عملکرد رو train و Test کم است Variance کم دارند و خوب Fit شده اند. (نه Overfit و نه underfit) بین آنها هم اگر بخواهیم نظر بدهیم k-nearest neighbors بهتر از بقیه عمل کرده.

پیشنهاد برای بهتر شدن پروژه:

۱. از آنجا که در بخش Visualization داده ها دو ویژگی هایی را مشاهده کردیم که به وسیله‌ی آنها گونه ها به طور خطی جدایی پذیر بودند میتوانیم فقط از آن دو ویژگی برای train کردن مدل هایمان استفاده کنیم.

۲. میتوانیم برای مشاهده ی کارکرد بهتر خود مدل ها learning curve بکشیم.

۳. میتوانیم برای اینکه دقیق تر بفهمیم variance و bias مدل هایمان چطوراست هم نمودار بکشیم.

منابع:

https://www.kaggle.com/jchen2186/machine-learning-with-iris-dataset
https://www.kaggle.com/ranjeetjain3/visualization-machine-learning-deep-learning
https://www.kaggle.com/uciml/iris
https://www.kaggle.com/jchen2186/machine-learning-with-iris-dataset
https://www.kaggle.com/ash316/ml-from-scratch-with-iris/comments