**Анализ и предобработка**

Из описания на сайте мы определяем основные ключи в объекте sklearn.utils.Bunch:

# Импорт необходимых модулей

import matplotlib

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

# Настройки для визуализации

# Если используется темная тема - лучше текст сделать белым

TEXT\_COLOR = 'black'

matplotlib.rcParams['figure.figsize'] = (15, 10)

matplotlib.rcParams['text.color'] = 'black'

matplotlib.rcParams['font.size'] = 14

matplotlib.rcParams['axes.labelcolor'] = TEXT\_COLOR

matplotlib.rcParams['xtick.color'] = TEXT\_COLOR

matplotlib.rcParams['ytick.color'] = TEXT\_COLOR

# Зафиксируем состояние случайных чисел

RANDOM\_STATE = 42

np.random.seed(RANDOM\_STATE)

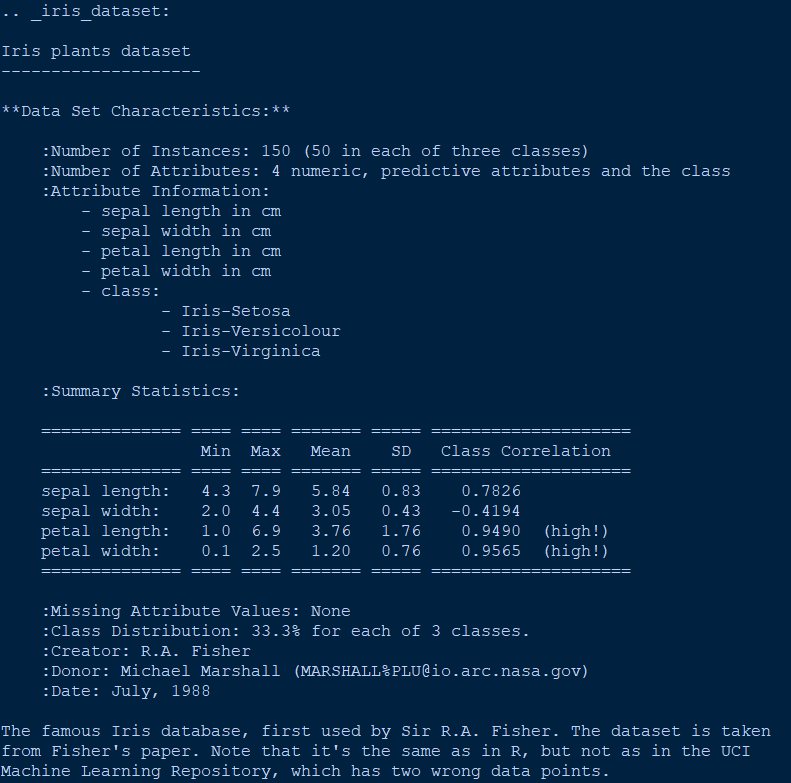
from sklearn.datasets import load\_iris

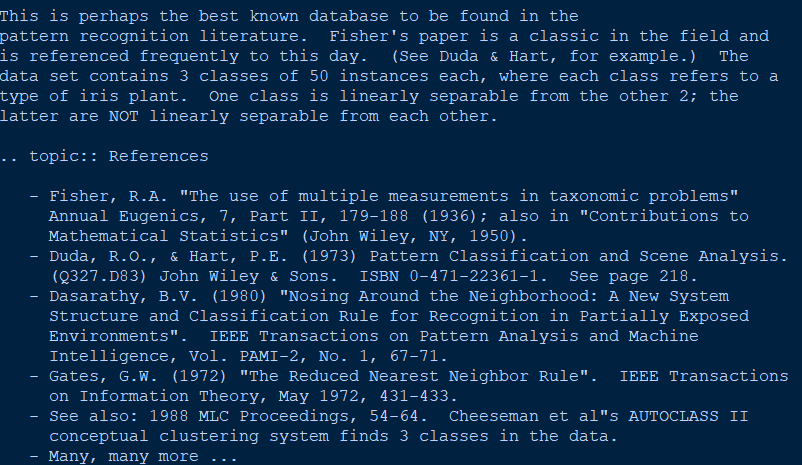
iris\_data = load\_iris()

print(iris\_data['DESCR'])

print(iris\_data['target\_names'])

Результат







Начнем анализ данных с загрузки данных в pandas в формате таблицы, в которую целевые классы запишем названиями в колонку species.

feature\_names = iris\_data['feature\_names']

df = pd.DataFrame(iris\_data['data'], columns=feature\_names)

species\_names = iris\_data['target\_names']

target\_idxs = iris\_data['target']

df['species'] = species\_names[target\_idxs]

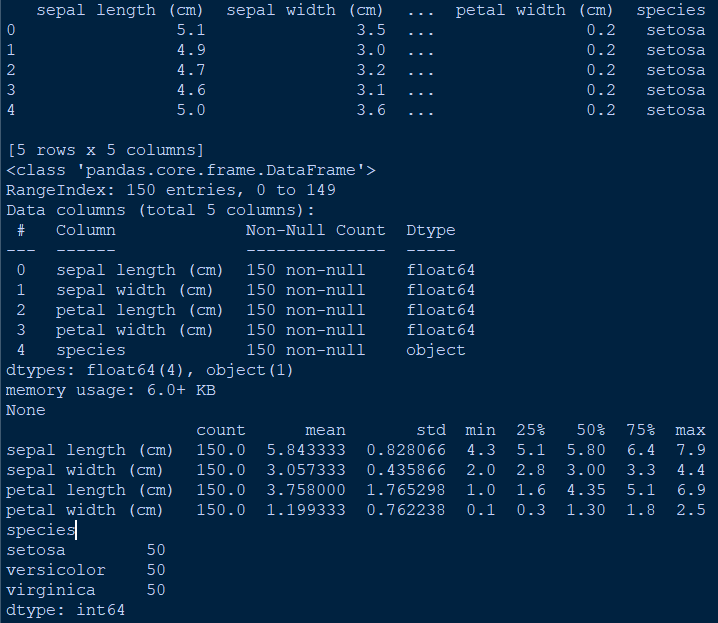
print(df.head())

print(df.info())

print(df.describe().T)

print(df.groupby('species').size())

Результат



fig, axs = plt.subplots(2, 2)

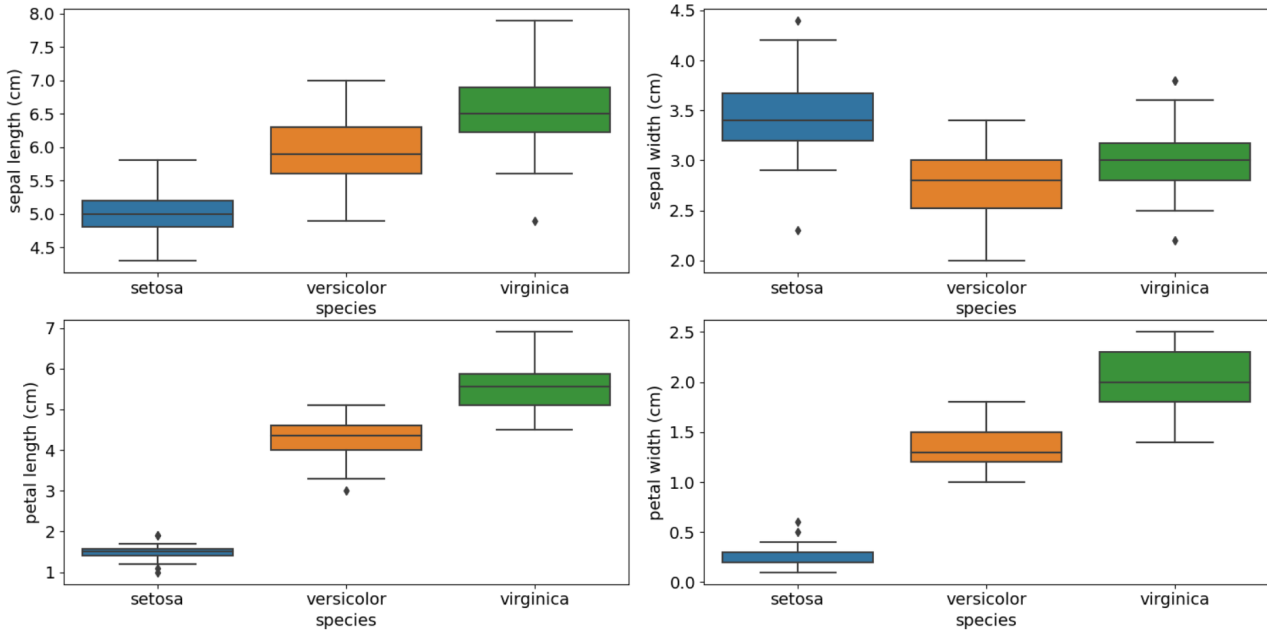
for i, feat\_name in enumerate(feature\_names):

row = i//2

col = i%2

sns.boxplot(x = 'species', y = feat\_name, data = df, order = species\_names, ax = axs[row, col]);

fig.tight\_layout();



fig, axs = plt.subplots(2, 2)

for i, feat\_name in enumerate(feature\_names):

row = i//2

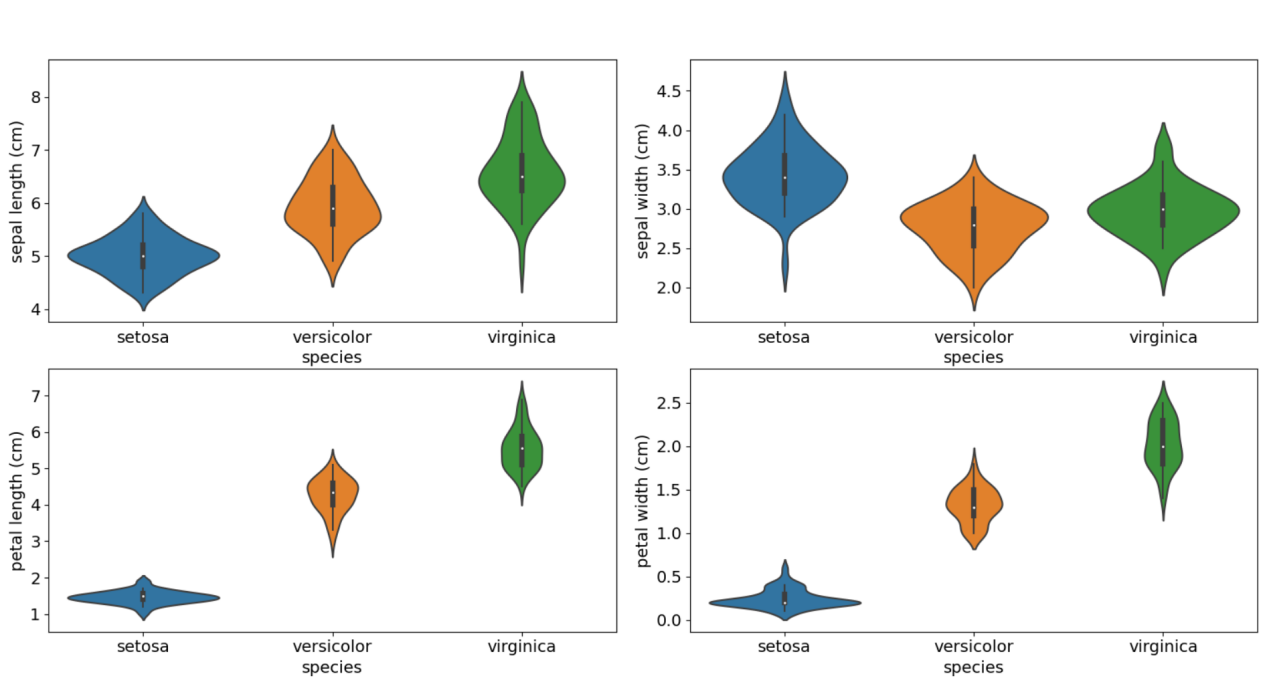
col = i%2

sns.violinplot(x = 'species', y = feat\_name, data = df, order = species\_names, ax = axs[row, col]);

fig.tight\_layout()

plt.show()

Результат

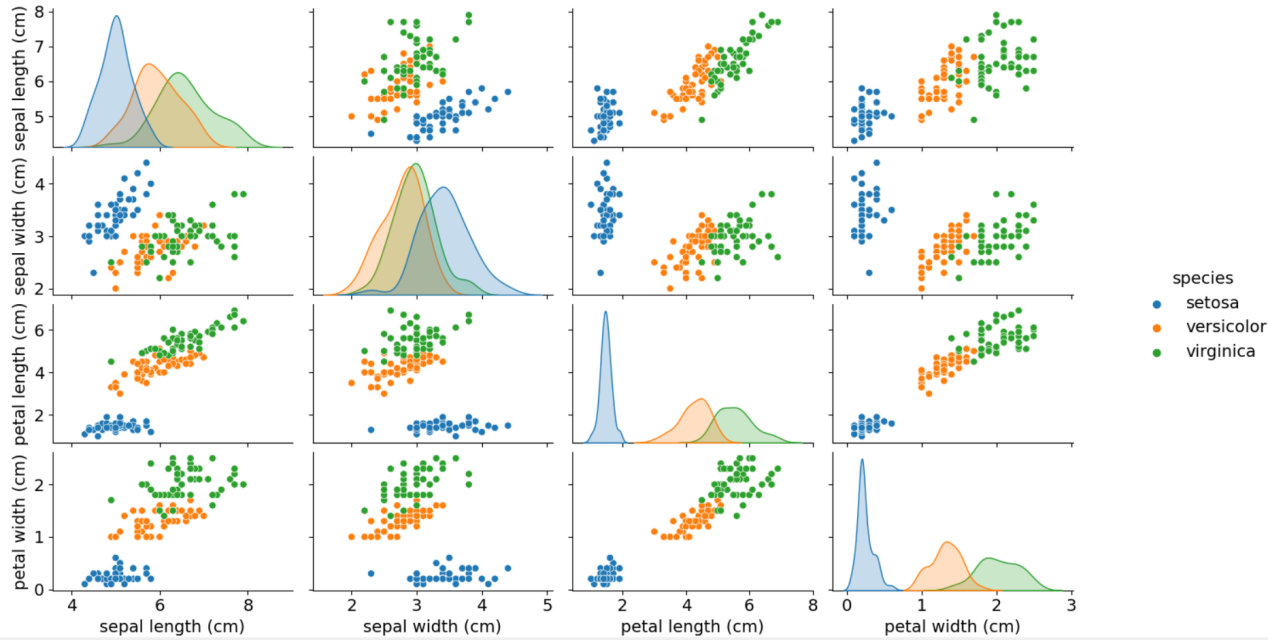


В данном случае небольшое количество признаков позволяет использовать попарное отображение признаков с помощью функции seaborn.pairplot().

sns.pairplot(df, hue='species')

plt.show()

Результат



# **Разработка модели**

Теперь пора делать модель для мультиклассовой логистической регрессии.

from sklearn.model\_selection import train\_test\_split

TRAIN\_RATIO = 0.7

X = df[feature\_names]

y = target\_idxs

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y,

train\_size=TRAIN\_RATIO,

random\_state=RANDOM\_STATE,

stratify=y

)

print(X\_train.shape, y\_train.shape)

print(X\_test.shape, y\_test.shape)

Результат



from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression(

random\_state=RANDOM\_STATE,

max\_iter=200,

multi\_class='multinomial'

)

print(logreg.fit(X\_train, y\_train))

Результат



Для ознакомления с результатами работы модели возьмем первый пример из тестовой выборки и проверим работу доступных у классификационной модели методов:

sample = X\_test.iloc[0]

prediction = logreg.predict([sample])

predict\_proba = logreg.predict\_proba([sample])

predict\_log\_proba = logreg.predict\_log\_proba([sample])

print(f' Sample:\n{sample}')

print(f' Prediction proba:\n{predict\_proba[0]}')

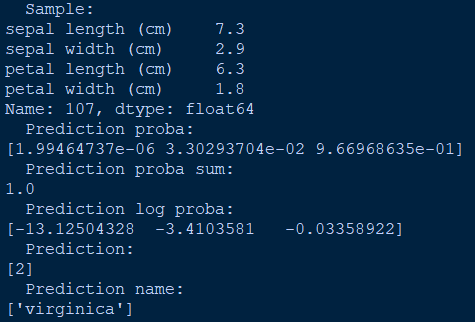
print(f' Prediction proba sum:\n{sum(predict\_proba[0])}')

print(f' Prediction log proba:\n{predict\_log\_proba[0]}')

print(f' Prediction:\n{prediction}')

print(f' Prediction name:\n{species\_names[prediction]}')

Результат



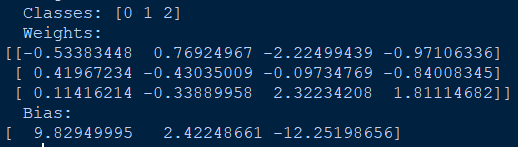
Помимо основных методов, объект логистической регрессии также имеет аттрибуты, хранящие значение весов и смещения.

print(f' Classes: {logreg.classes\_}')

print(f' Weights:\n{logreg.coef\_}')

print(f' Bias:\n{logreg.intercept\_}')

Результат



После ознакомления с базовыми методами и аттрибутами самое время оценить работу классификатора. Для этого воспользуемся базовыми методами визуализации и числовой оценки.

from sklearn.metrics import plot\_confusion\_matrix

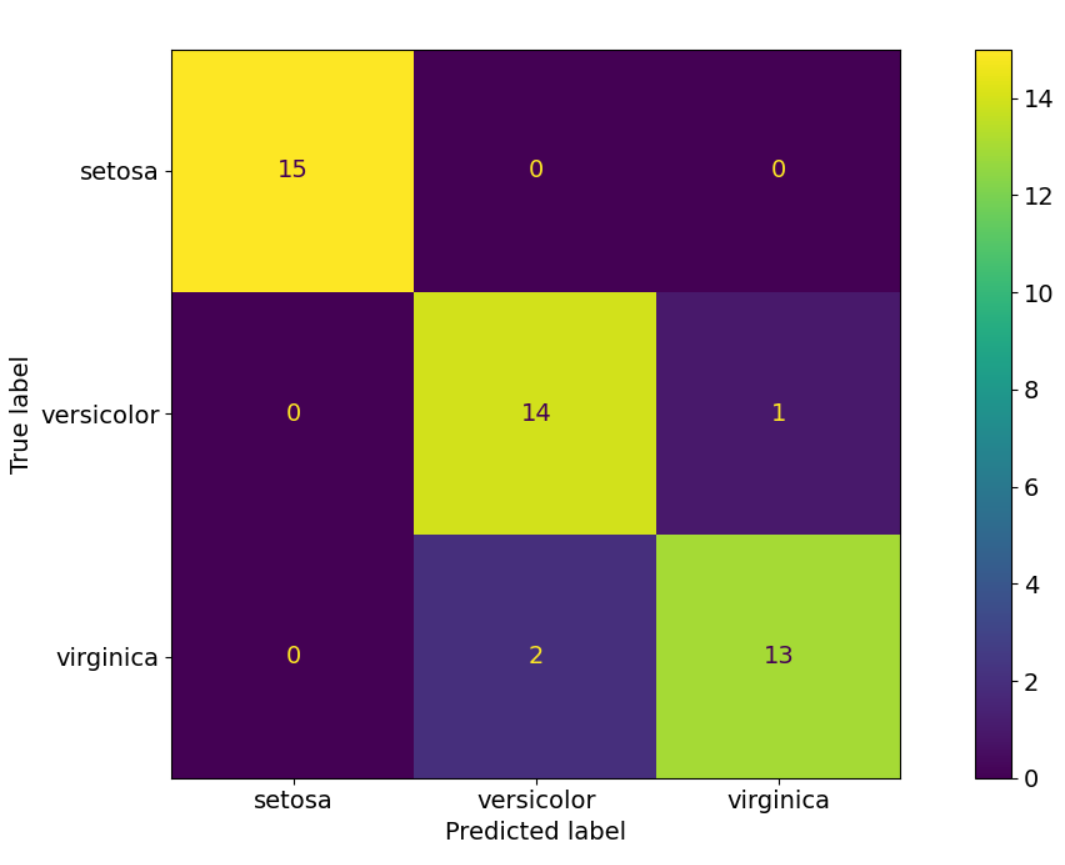
disp = plot\_confusion\_matrix(

logreg, X\_test, y\_test,

display\_labels=species\_names)

plt.show()

Результат

  
Помимо матричного представления полезно также оценить основные числовые показатели.

from sklearn.metrics import classification\_report

y\_pred = logreg.predict(X\_test)

report = classification\_report(

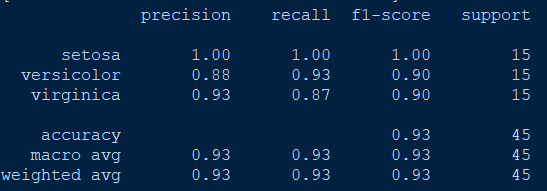
y\_test, y\_pred,

target\_names=species\_names

)

print(report)

Результат



# **Кросс-валидация**

Кросс-валидация очень полезна в случае малого набора данных - наш случай! Для этого sklearn имеет ряд функций - одна из них функция обучения + оценки sklearn.model\_selection.cross\_val\_score().

from sklearn.model\_selection import cross\_val\_score

X = df[feature\_names]

y = target\_idxs

# Данная функция используется только для оценки

# (она обучает модель внутри, но не возвращает ее)

scores = cross\_val\_score(

logreg, # Модель для оценки

X, # Данные для обучения

y, # Разметка для обучения

cv=5, # Количество фолдов

scoring='f1\_macro' # Желаемая метрика

)

print(f'Scores: {scores}')

print(f'F1 (macro): {scores.mean(): 0.2f} (+/- {scores.std() \* 2: 0.2f})')

Результат



# **Задание**

* Проведите базовый анализ данных и разработайте базовую модель (не забудьте оценить работу модели);
* Разберитесь и отобразите ROC-кривую с помощью sklearn.metrics.roc\_curve;
* Оцените работу модели при использовании StandartScaler;
* Исследуйте работу модели LogisticRegression с изменением аргумента (3 изменения / различных значения) iter - объяснить, что происходит при установке значения по-умолчанию (значение взять из справки);
* Оцените работу модели при обучении на двух признаках (попробовать 2 пары признаков):
  + sepal length (cm) + sepal width (cm);
  + petal length (cm) + petal width (cm);
* Найдите лучшую пару признаков методом кросс-валидации на обучающих данных (перебрать все возможные пары);
* Освойте и изучите работу подхода классификации **[К ближайших соседей (KNN)](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)**, продемонстрируйте работу и основную суть метода, сравните с показателями логистической регрессии;
* Оцените влияние аргумента n\_neighbors на работу модели KNN (7 различных значений); постройте график или таблицу основных показателей метрик от значения количества соседей;
* Отобразите ****плоскость решений**** при использовании двух признаков как лучшей модели логистической регрессии, так и лучшей модели KNN;
* (Для особо интересующихся) Почитайте про принцип работы метода [SVM](https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html) (Support Vector Machine), сравните [линейный тип SVM](https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html) модели с остальными.

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

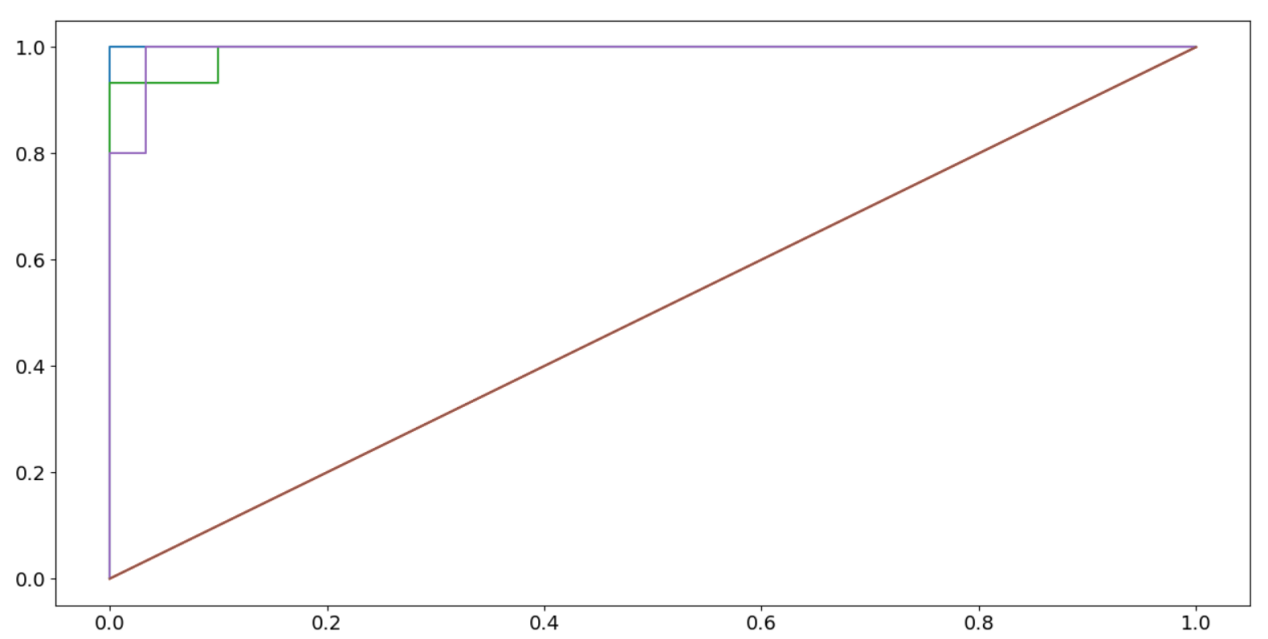
scaler.fit(X\_train)

X\_train\_scaled = scaler.transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

logreg.fit(X\_train\_scaled, y\_train)

Результат



from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X\_train)

X\_train\_scaled = scaler.transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

logreg.fit(X\_train\_scaled, y\_train)

y\_pred=logreg.predict(X\_test\_scaled)

report = classification\_report(

y\_test, y\_pred,

target\_names=species\_names

)

print(report)

scores = cross\_val\_score(

logreg, # Модель для оценки

X\_train\_scaled, # Данные для обучения

y\_train, # Разметка для обучения

cv=5, # Количество фолдов

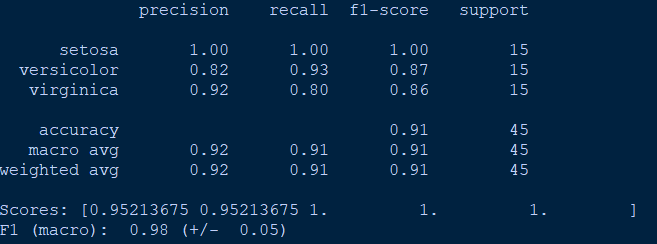
scoring='f1\_macro' # Желаемая метрика

)

print(f'Scores: {scores}')

print(f'F1 (macro): {scores.mean(): 0.2f} (+/- {scores.std() \* 2: 0.2f})')

Результат



logreg = LogisticRegression(

random\_state=RANDOM\_STATE,

max\_iter=100,

multi\_class='multinomial'

)

logreg.fit(X\_train, y\_train)

y\_pred=logreg.predict(X\_test)

report = classification\_report(

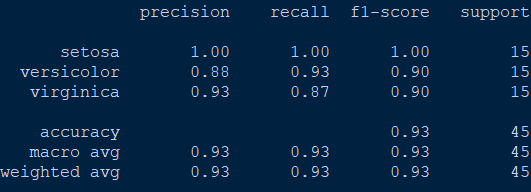
y\_test, y\_pred,

target\_names=species\_names

)

print(report)

Результат



logreg=LogisticRegression(

random\_state=RANDOM\_STATE,

max\_iter=50,

multi\_class='multinomial'

)

logreg.fit(X\_train, y\_train)

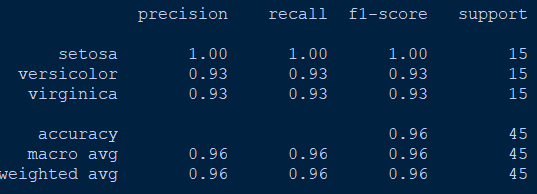
y\_pred=logreg.predict(X\_test)

report = classification\_report(

y\_test, y\_pred,

target\_names=species\_names

Результат



logreg = LogisticRegression(

random\_state=RANDOM\_STATE,

max\_iter=1000,

multi\_class='multinomial'

)

logreg.fit(X\_train, y\_train)

y\_pred=logreg.predict(X\_test)

report = classification\_report(

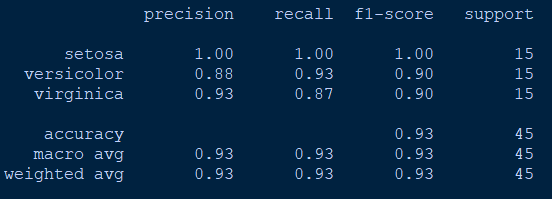
y\_test, y\_pred,

target\_names=species\_names

)

print(report)

Результат



X = df[[feature\_names[0],feature\_names[1]]]

y = target\_idxs

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y,

train\_size=TRAIN\_RATIO,

random\_state=RANDOM\_STATE,

stratify=y

)

logreg.fit(X\_train, y\_train)

y\_pred=logreg.predict(X\_test)

report = classification\_report(

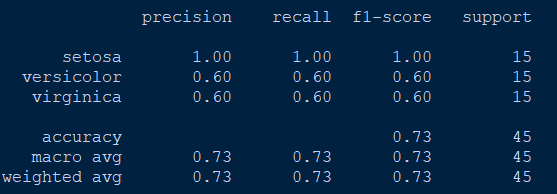
y\_test, y\_pred,

target\_names=species\_names

)

print(report)

Результат



X = df[[feature\_names[2],feature\_names[3]]]

y = target\_idxs

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y,

train\_size=TRAIN\_RATIO,

random\_state=RANDOM\_STATE,

stratify=y

)

logreg.fit(X\_train, y\_train)

y\_pred=logreg.predict(X\_test)

report = classification\_report(

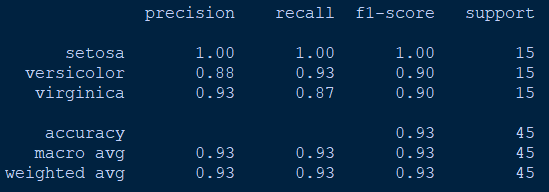
y\_test, y\_pred,

target\_names=species\_names

)

print(report)

Результат



for i in range(len(feature\_names)):

for j in range(len(feature\_names)):

if (i<=j):

continue

X = df[[feature\_names[i],feature\_names[j]]]

y = target\_idxs

scores = cross\_val\_score(

logreg, # Модель для оценки

X, # Данные для обучения

y, # Разметка для обучения

cv=5, # Количество фолдов

scoring='f1\_macro' # Желаемая метрика

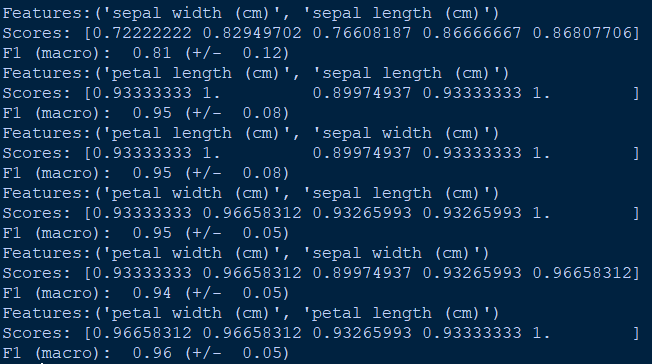
)

print(f'Features:{feature\_names[i],feature\_names[j]}')

print(f'Scores: {scores}')

print(f'F1 (macro): {scores.mean(): 0.2f} (+/- {scores.std() \* 2: 0.2f})')

Результат



from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=3)

scores = cross\_val\_score(

knn, # Модель для оценки

X, # Данные для обучения

y, # Разметка для обучения

cv=5, # Количество фолдов

scoring='f1\_macro' # Желаемая метрика

)

print(f'Features:{feature\_names[i],feature\_names[j]}')

print(f'Scores: {scores}')

print(f'F1 (macro): {scores.mean(): 0.2f} (+/- {scores.std() \* 2: 0.2f})')

Результат



scr=[]

for K\_n in range(1,8):

knn = KNeighborsClassifier(n\_neighbors=K\_n)

scores = cross\_val\_score(

knn, # Модель для оценки

X, # Данные для обучения

y, # Разметка для обучения

cv=5, # Количество фолдов

scoring='f1\_macro' # Желаемая метрика

)

scr.append(scores.mean())

plt.plot(range(1,8),scr)

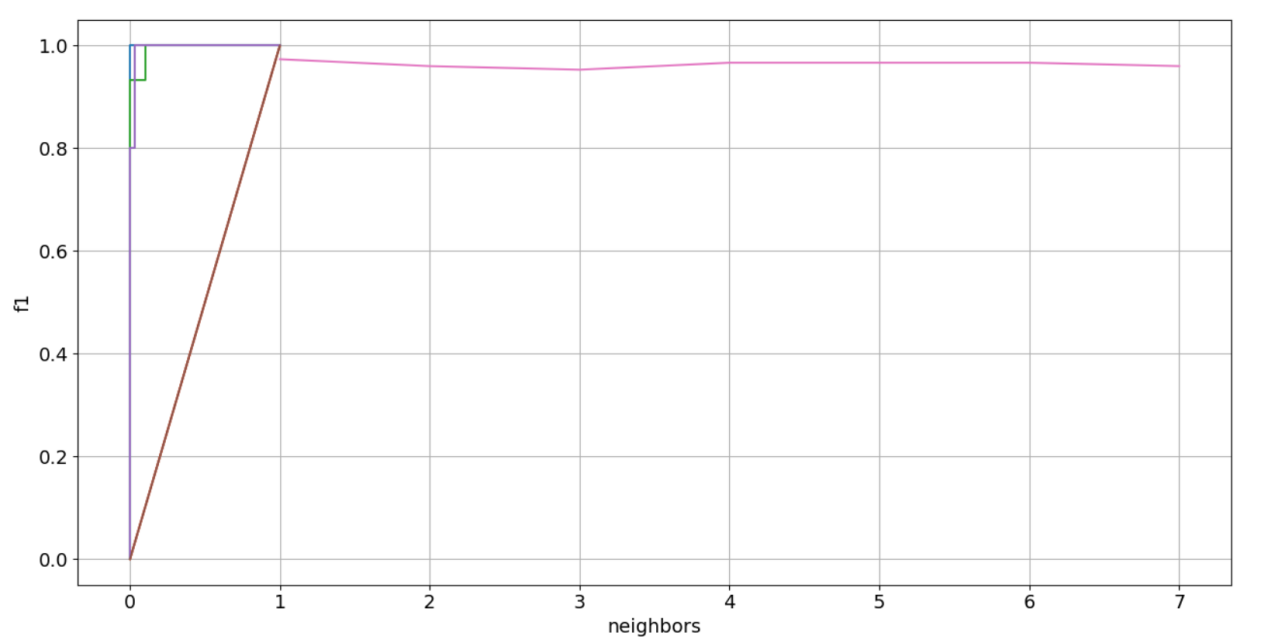
plt.grid(True)

plt.ylabel('f1')

plt.xlabel('neighbors')

plt.show()

Результат



from matplotlib.colors import ListedColormap

h = .02

cmap\_light = ListedColormap(['orange', 'cyan', 'cornflowerblue'])

cmap\_bold = ListedColormap(['darkorange', 'c', 'darkblue'])

X\_scaled=iris\_data.data[:, 0:2]

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(X\_scaled,y)

x\_min, x\_max = X\_scaled[:, 0].min() - 1, X\_scaled[:, 0].max() + 1

y\_min, y\_max = X\_scaled[:, 1].min() - 1, X\_scaled[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, 0.1),

np.arange(y\_min, y\_max, 0.1))

Z = knn.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.figure()

plt.pcolormesh(xx, yy, Z, cmap=cmap\_light)

plt.scatter(X\_scaled[:, 0], X\_scaled[:, 1], c=y, cmap=cmap\_bold,

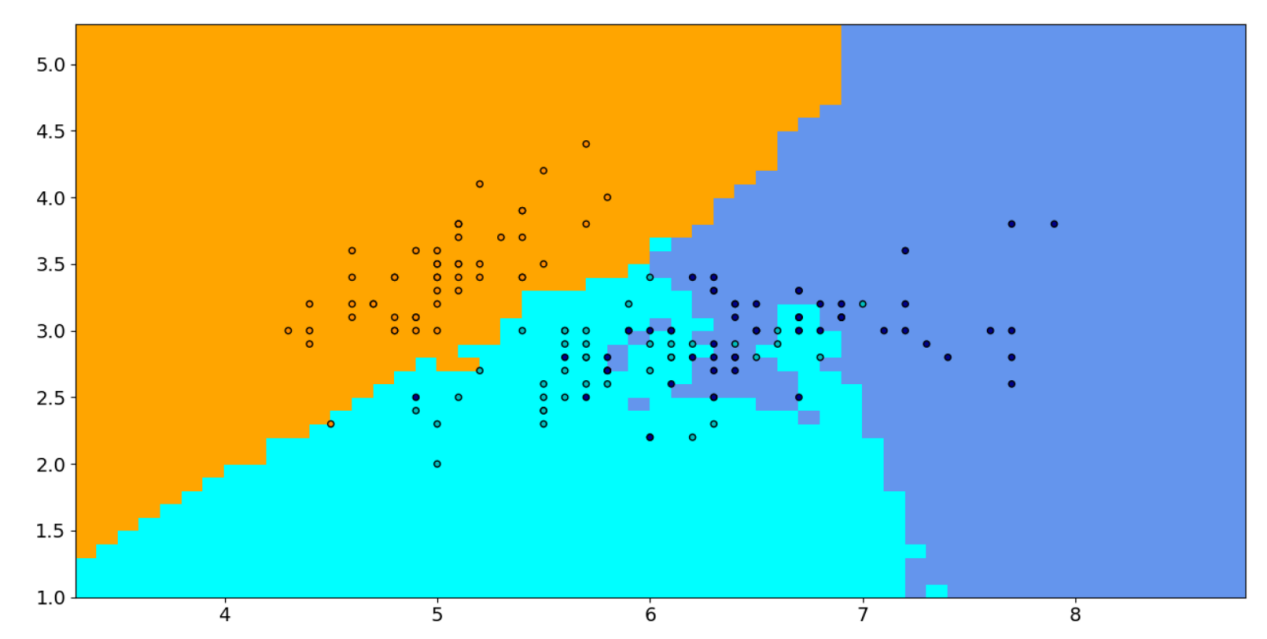
edgecolor='k', s=20)

plt.xlim(xx.min(), xx.max())

plt.ylim(yy.min(), yy.max())

plt.show()

Результат



logreg = LogisticRegression(

random\_state=RANDOM\_STATE,

max\_iter=200,

multi\_class='multinomial'

)

logreg.fit(X\_scaled,y)

x\_min, x\_max = X\_scaled[:, 0].min() - 1, X\_scaled[:, 0].max() + 1

y\_min, y\_max = X\_scaled[:, 1].min() - 1, X\_scaled[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h),

np.arange(y\_min, y\_max, h))

Z = logreg.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.figure()

plt.pcolormesh(xx, yy, Z, cmap=cmap\_light)

plt.scatter(X\_scaled[:, 0], X\_scaled[:, 1], c=y, cmap=cmap\_bold,

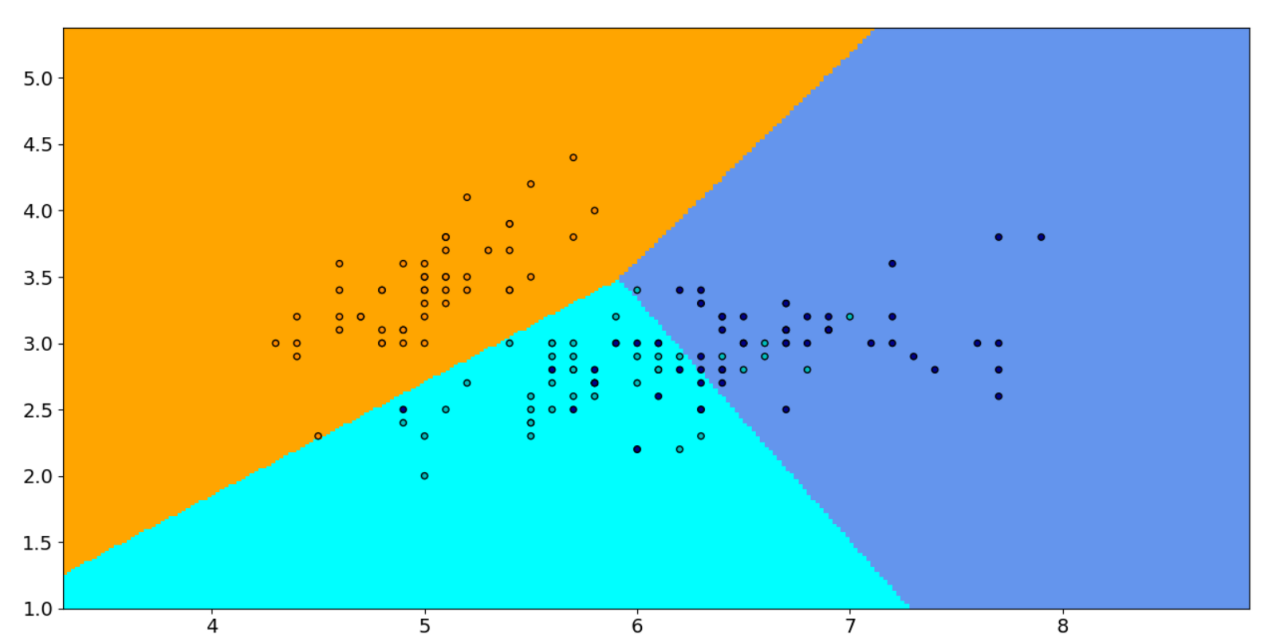
edgecolor='k', s=20)

plt.xlim(xx.min(), xx.max())

plt.ylim(yy.min(), yy.max())

plt.show()

Результат



from sklearn.svm import LinearSVC

TRAIN\_RATIO = 0.7

X = df[feature\_names]

y = target\_idxs

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y,

train\_size=TRAIN\_RATIO,

random\_state=RANDOM\_STATE,

stratify=y

)

svm=LinearSVC(

tol=1e-10,

C=10,

random\_state=RANDOM\_STATE,

max\_iter=5000

)

svm.fit(X\_train,y\_train)

Результат



scores = cross\_val\_score(

svm, # Модель для оценки

X, # Данные для обучения

y, # Разметка для обучения

cv=5, # Количество фолдов

scoring='f1\_macro' # Желаемая метрика

)

print(f'Scores: {scores}')

print(f'F1 (macro): {scores.mean(): 0.2f} (+/- {scores.std() \* 2: 0.2f})')

Результат

