**ЛАБОРАТОРНА РОБОТА № 2**

**ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ**

**Мета роботи:**

використовуючи спеціалізовані бібліотеки та мову

програмування Python дослідити різні методи класифікації даних та

навчитися їх порівнювати.

**Хід роботи:**

**Завдання 1.**

1. Age - (числова) вік.
2. Workclass - (категоріальна) робочий клас: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
3. Fnlwgt - (числова) final weight, кількість людей, яку представляє запис.
4. Education - (категоріальна) Рівень освіти: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
5. Education-num - (числова) кількість освіт.
6. marital-status - (категоріальна) сімейний стан: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
7. occupation - (категоріальна) Професія: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
8. relationship - (категоріальна) Роль у сім’ї: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
9. race - (категоріальна) – Раса: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
10. sex – (бінарна) Стать.
11. capital-gain - (числова) Дохід.
12. capital-loss - (числова) Витрати.
13. hours-per-week - (числова) К-ть робочих годин на тиждень.
14. native-country - (категоріальна) Рідна країна.
15. income – (бінарна) Дохід >50K або <=50K.

Лістинг LR\_2\_task\_1.py:

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

*from* sklearn *import* preprocessing

*from* sklearn.svm *import* LinearSVC

*from* sklearn.multiclass *import* OneVsOneClassifier

*from* sklearn.model\_selection *import* train\_test\_split, cross\_val\_score

input\_file = 'income\_data.txt'

X = []

y = []

count\_class1 = 0

count\_class2 = 0

max\_datapoints = 25000

*with* open(input\_file, 'r') *as* f:

*for* line *in* f.readlines():

*if* count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:

*break*

*if* '?' in line:

*continue*

        data = line[:-1].split(', ')

        income\_class = data[-1]

*if* income\_class == '<=50K' and count\_class1 < max\_datapoints:

            X.append(data)

            count\_class1 += 1

*if* income\_class == '>50K' and count\_class2 < max\_datapoints:

            X.append(data)

            count\_class2 += 1

X = np.array(X)

label\_encoder = []

X\_encoded = np.empty(X.shape)

*for* i, item *in* enumerate(X[0]):

*if* item.isdigit():

        X\_encoded[:, i] = X[:, i]

*else*:

        current\_label\_encoder = preprocessing.LabelEncoder()

        label\_encoder.append(current\_label\_encoder)

        X\_encoded[:, i] = current\_label\_encoder.fit\_transform(X[:, i])

X = X\_encoded[:, :-1].astype(int)

y = X\_encoded[:, -1].astype(int)

classifier = OneVsOneClassifier(LinearSVC(random\_state=0))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)

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

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

accuracy = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)

print("Accuracy score: " + str(round(100 \* accuracy.mean(), 2)) + "%")

precision = cross\_val\_score(classifier, X, y, scoring='precision', cv=3)

print("Precision score: " + str(round(100 \* precision.mean(), 2)) + "%")

recall = cross\_val\_score(classifier, X, y, scoring='recall', cv=3)

print("Recall score: " + str(round(100 \* recall.mean(), 2)) + "%")

f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)

print("F1 score: " + str(round(100 \* f1.mean(), 2)) + "%")

input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White',

              'Male',

              '0', '0', '40', 'United-States']

input\_data\_encoded = [-1] \* len(input\_data)

count = 0

*for* i, item *in* enumerate(input\_data):

*if* item.isdigit():

        input\_data\_encoded[i] = int(input\_data[i])

*else*:

        encoder = label\_encoder[count]

        input\_data\_encoded[i] = int(encoder.transform([(input\_data[i])])[-1])

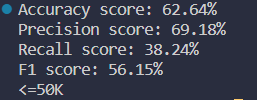
        count += 1

input\_data\_encoded = np.array(input\_data\_encoded)

predicted\_class = classifier.predict([input\_data\_encoded])

print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])

Результат виконання програми:



Висновок роботи програми: тестова точка належить до класу <=50K

<=50K

**Завдання 2.2. Порівняння якості класифікаторів SVM з нелінійними**

**ядрами**

Лістинг програми LR\_2\_task\_2\_1.py (Поліноміальне):

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

*from* sklearn *import* preprocessing

*from* sklearn.svm *import* SVC

*from* sklearn.multiclass *import* OneVsOneClassifier

*from* sklearn.model\_selection *import* train\_test\_split, cross\_val\_score

input\_file = 'income\_data.txt'

X = []

y = []

count\_class1 = 0

count\_class2 = 0

max\_datapoints = 25000

*with* open(input\_file, 'r') *as* f:

*for* line *in* f.readlines():

*if* count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:

*break*

*if* '?' in line:

*continue*

        data = line[:-1].split(', ')

        income\_class = data[-1]

*if* income\_class == '<=50K' and count\_class1 < max\_datapoints:

            X.append(data)

            count\_class1 += 1

*if* income\_class == '>50K' and count\_class2 < max\_datapoints:

            X.append(data)

            count\_class2 += 1

X = np.array(X)

label\_encoder = []

X\_encoded = np.empty(X.shape)

*for* i, item *in* enumerate(X[0]):

*if* item.isdigit():

        X\_encoded[:, i] = X[:, i]

*else*:

        current\_label\_encoder = preprocessing.LabelEncoder()

        label\_encoder.append(current\_label\_encoder)

        X\_encoded[:, i] = current\_label\_encoder.fit\_transform(X[:, i])

X = X\_encoded[:, :-1].astype(int)

y = X\_encoded[:, -1].astype(int)

classifier = OneVsOneClassifier(SVC(kernel='poly', degree=8, max\_iter=10000))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)

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

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

accuracy = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)

print("Accuracy score: " + str(round(100 \* accuracy.mean(), 2)) + "%")

precision = cross\_val\_score(classifier, X, y, scoring='precision', cv=3)

print("Precision score: " + str(round(100 \* precision.mean(), 2)) + "%")

recall = cross\_val\_score(classifier, X, y, scoring='recall', cv=3)

print("Recall score: " + str(round(100 \* recall.mean(), 2)) + "%")

f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)

print("F1 score: " + str(round(100 \* f1.mean(), 2)) + "%")

input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White',

              'Male',

              '0', '0', '40', 'United-States']

input\_data\_encoded = [-1] \* len(input\_data)

count = 0

*for* i, item *in* enumerate(input\_data):

*if* item.isdigit():

        input\_data\_encoded[i] = int(input\_data[i])

*else*:

        encoder = label\_encoder[count]

        input\_data\_encoded[i] = int(encoder.transform([(input\_data[i])])[-1])

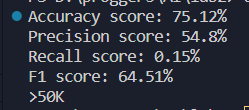
        count += 1

input\_data\_encoded = np.array(input\_data\_encoded)

predicted\_class = classifier.predict([input\_data\_encoded])

print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])

Результат виконання poly:



Лістинг програми LR\_2\_task\_2\_2.py (Гаусове):

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

*from* sklearn *import* preprocessing

*from* sklearn.svm *import* SVC

*from* sklearn.multiclass *import* OneVsOneClassifier

*from* sklearn.model\_selection *import* train\_test\_split, cross\_val\_score

input\_file = 'income\_data.txt'

X = []

y = []

count\_class1 = 0

count\_class2 = 0

max\_datapoints = 25000

*with* open(input\_file, 'r') *as* f:

*for* line *in* f.readlines():

*if* count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:

*break*

*if* '?' in line:

*continue*

        data = line[:-1].split(', ')

        income\_class = data[-1]

*if* income\_class == '<=50K' and count\_class1 < max\_datapoints:

            X.append(data)

            count\_class1 += 1

*if* income\_class == '>50K' and count\_class2 < max\_datapoints:

            X.append(data)

            count\_class2 += 1

X = np.array(X)

label\_encoder = []

X\_encoded = np.empty(X.shape)

*for* i, item *in* enumerate(X[0]):

*if* item.isdigit():

        X\_encoded[:, i] = X[:, i]

*else*:

        current\_label\_encoder = preprocessing.LabelEncoder()

        label\_encoder.append(current\_label\_encoder)

        X\_encoded[:, i] = current\_label\_encoder.fit\_transform(X[:, i])

X = X\_encoded[:, :-1].astype(int)

y = X\_encoded[:, -1].astype(int)

classifier = OneVsOneClassifier(SVC(kernel='rbf', max\_iter=10000))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)

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

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

accuracy = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)

print("Accuracy score: " + str(round(100 \* accuracy.mean(), 2)) + "%")

precision = cross\_val\_score(classifier, X, y, scoring='precision', cv=3)

print("Precision score: " + str(round(100 \* precision.mean(), 2)) + "%")

recall = cross\_val\_score(classifier, X, y, scoring='recall', cv=3)

print("Recall score: " + str(round(100 \* recall.mean(), 2)) + "%")

f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)

print("F1 score: " + str(round(100 \* f1.mean(), 2)) + "%")

input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White',

              'Male',

              '0', '0', '40', 'United-States']

input\_data\_encoded = [-1] \* len(input\_data)

count = 0

*for* i, item *in* enumerate(input\_data):

*if* item.isdigit():

        input\_data\_encoded[i] = int(input\_data[i])

*else*:

        encoder = label\_encoder[count]

        input\_data\_encoded[i] = int(encoder.transform([(input\_data[i])])[-1])

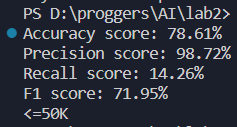
        count += 1

input\_data\_encoded = np.array(input\_data\_encoded)

predicted\_class = classifier.predict([input\_data\_encoded])

print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])

Результат виконання rbf:



Лістинг програми LR\_2\_task\_2\_3.py (Сигмоїдальне):

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

*from* sklearn *import* preprocessing

*from* sklearn.svm *import* SVC

*from* sklearn.multiclass *import* OneVsOneClassifier

*from* sklearn.model\_selection *import* train\_test\_split, cross\_val\_score

input\_file = 'income\_data.txt'

X = []

y = []

count\_class1 = 0

count\_class2 = 0

max\_datapoints = 25000

*with* open(input\_file, 'r') *as* f:

*for* line *in* f.readlines():

*if* count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:

*break*

*if* '?' in line:

*continue*

        data = line[:-1].split(', ')

        income\_class = data[-1]

*if* income\_class == '<=50K' and count\_class1 < max\_datapoints:

            X.append(data)

            count\_class1 += 1

*if* income\_class == '>50K' and count\_class2 < max\_datapoints:

            X.append(data)

            count\_class2 += 1

X = np.array(X)

label\_encoder = []

X\_encoded = np.empty(X.shape)

*for* i, item *in* enumerate(X[0]):

*if* item.isdigit():

        X\_encoded[:, i] = X[:, i]

*else*:

        current\_label\_encoder = preprocessing.LabelEncoder()

        label\_encoder.append(current\_label\_encoder)

        X\_encoded[:, i] = current\_label\_encoder.fit\_transform(X[:, i])

X = X\_encoded[:, :-1].astype(int)

y = X\_encoded[:, -1].astype(int)

classifier = OneVsOneClassifier(SVC(kernel='sigmoid', max\_iter=10000))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=5)

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

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

accuracy = cross\_val\_score(classifier, X, y, scoring='accuracy', cv=3)

print("Accuracy score: " + str(round(100 \* accuracy.mean(), 2)) + "%")

precision = cross\_val\_score(classifier, X, y, scoring='precision', cv=3)

print("Precision score: " + str(round(100 \* precision.mean(), 2)) + "%")

recall = cross\_val\_score(classifier, X, y, scoring='recall', cv=3)

print("Recall score: " + str(round(100 \* recall.mean(), 2)) + "%")

f1 = cross\_val\_score(classifier, X, y, scoring='f1\_weighted', cv=3)

print("F1 score: " + str(round(100 \* f1.mean(), 2)) + "%")

input\_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White',

              'Male',

              '0', '0', '40', 'United-States']

input\_data\_encoded = [-1] \* len(input\_data)

count = 0

*for* i, item *in* enumerate(input\_data):

*if* item.isdigit():

        input\_data\_encoded[i] = int(input\_data[i])

*else*:

        encoder = label\_encoder[count]

        input\_data\_encoded[i] = int(encoder.transform([(input\_data[i])])[-1])

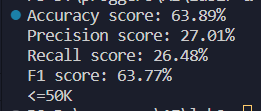
        count += 1

input\_data\_encoded = np.array(input\_data\_encoded)

predicted\_class = classifier.predict([input\_data\_encoded])

print(label\_encoder[-1].inverse\_transform(predicted\_class)[0])

Результат виконання sigmoid:



Для усіх трьох видів SVM було задано пеараметр max-iter=10000.

Висновок: Гаусове (RBF) ядро має найвищий показник точності (Accuracy) серед усіх ядер і найвищий показник точності класифікації. Він також має високий показник Precision, що означає, що модель добре виділяє позитивні класи. Однак, Recall у цього ядра не є дуже високим, що може означати, що воно може пропускати деякі позитивні приклади. F1-score також вищий, ніж у лінійного ядра.

Поліноміальне ядро має високий показник Accuracy, але відмінності в Precision та Recall є дуже великими, що може свідчити про дуже нерівномірну класифікацію. Висока точність може бути обумовлена великою кількістю правильно класифікованих негативних прикладів.

Сигмоїдальне ядро має низький показник точності (Accuracy) та Precision, що може означати, що це ядро слабко підходить для даного завдання класифікації.

З усього цього можна зробити висновок, що гаусове (RBF) ядро має найкращі результати.

**Завдання 2.3. Порівняння якості класифікаторів на прикладі**

**класифікації сортів ірисів**

Лістинг коду ознайомлення зі структурою даних:

*from* sklearn.datasets *import* load\_iris

iris\_dataset = load\_iris()

print("Ключі iris\_dataset: \n{}".format(iris\_dataset.keys()))

print(iris\_dataset['DESCR'][:193] + "\n...")

print("Назви відповідей:{}".format(iris\_dataset['target\_names']))

print("Назва ознак: \n{}".format(iris\_dataset['feature\_names']))

print("Тип масиву data: {}".format(type(iris\_dataset['data'])))

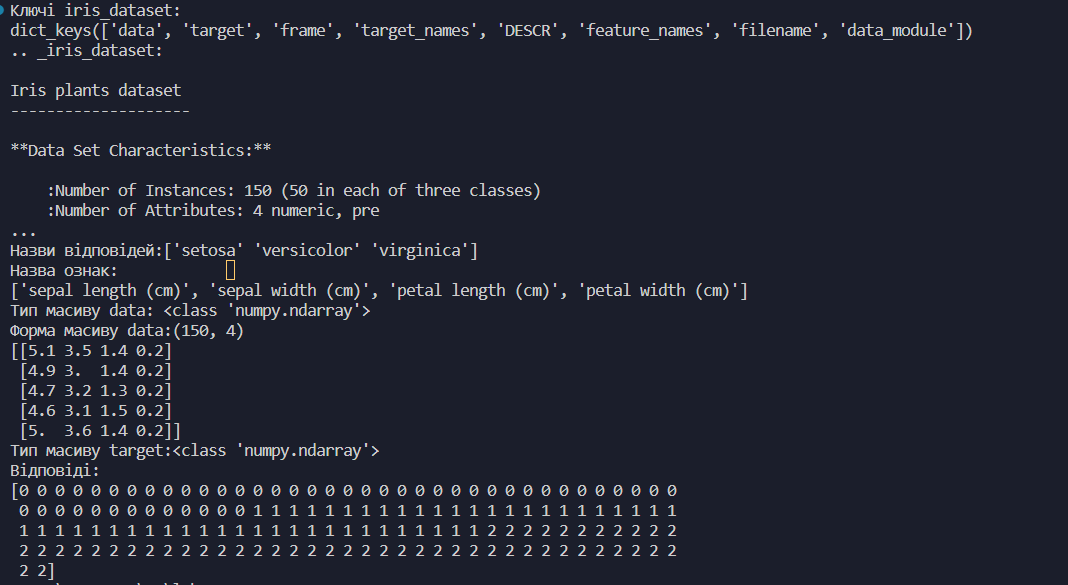
print("Форма масиву data:{}".format(iris\_dataset['data'].shape))

print(iris\_dataset['data'][:5])

print("Тип масиву target:{}".format(type(iris\_dataset['target'])))

print("Відповіді:\n{}".format(iris\_dataset['target']))

Результат**:**

****

Код візуалізації графіків:

*from* pandas *import* read\_csv

*from* pandas.plotting *import* scatter\_matrix

*from* matplotlib *import* pyplot

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

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

*from* sklearn.model\_selection *import* StratifiedKFold

*from* sklearn.metrics *import* classification\_report

*from* sklearn.metrics *import* confusion\_matrix

*from* sklearn.metrics *import* accuracy\_score

*from* sklearn.linear\_model *import* LogisticRegression

*from* sklearn.tree *import* DecisionTreeClassifier

*from* sklearn.neighbors *import* KNeighborsClassifier

*from* sklearn.discriminant\_analysis *import* LinearDiscriminantAnalysis

*from* sklearn.naive\_bayes *import* GaussianNB

*from* sklearn.svm *import* SVC

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

dataset = read\_csv(url, names=names)

print(dataset.shape)

print(dataset.head(20))

print(dataset.describe())

print(dataset.groupby('class').size())

dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)

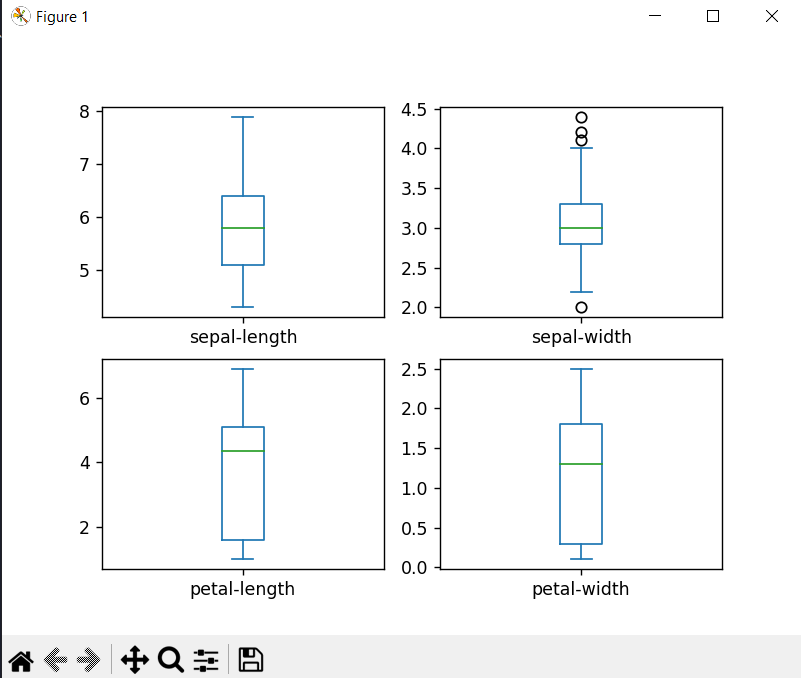
pyplot.show()

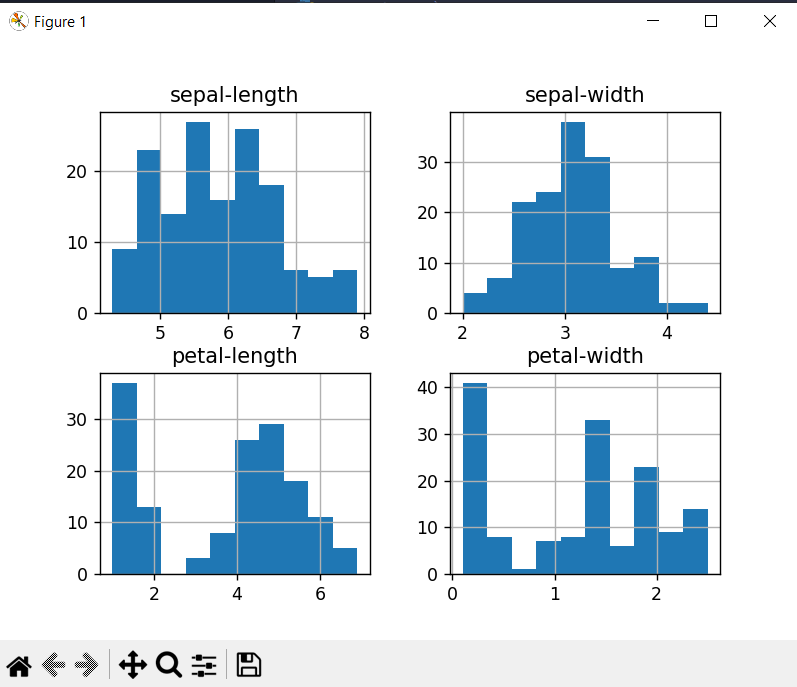
dataset.hist()

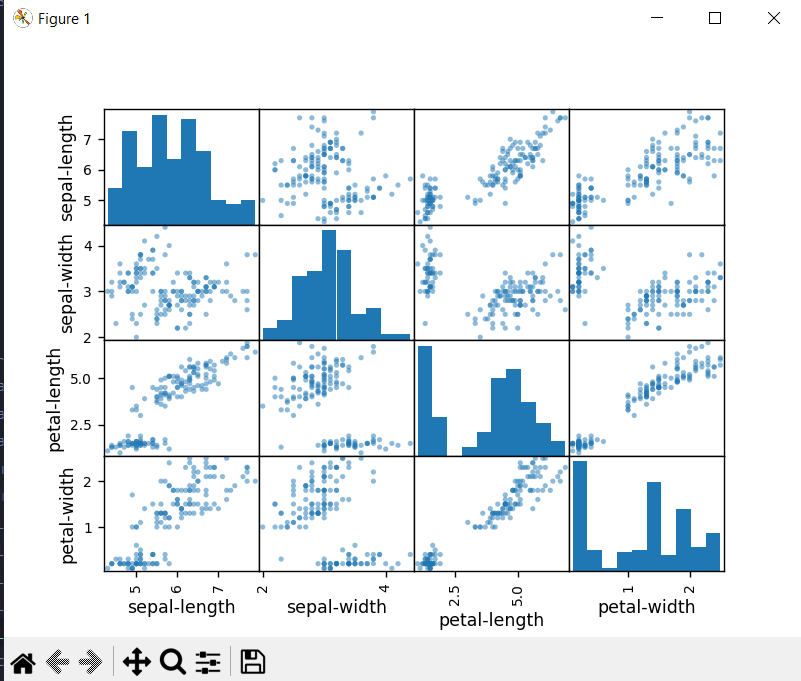
pyplot.show()

scatter\_matrix(dataset)

pyplot.show()







Лістинг програми:

*from* pandas *import* read\_csv

*from* pandas.plotting *import* scatter\_matrix

*from* matplotlib *import* pyplot

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

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

*from* sklearn.model\_selection *import* StratifiedKFold

*from* sklearn.metrics *import* classification\_report

*from* sklearn.metrics *import* confusion\_matrix

*from* sklearn.metrics *import* accuracy\_score

*from* sklearn.linear\_model *import* LogisticRegression

*from* sklearn.tree *import* DecisionTreeClassifier

*from* sklearn.neighbors *import* KNeighborsClassifier

*from* sklearn.discriminant\_analysis *import* LinearDiscriminantAnalysis

*from* sklearn.naive\_bayes *import* GaussianNB

*from* sklearn.svm *import* SVC

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

dataset = read\_csv(url, names=names)

print(dataset.shape)

print(dataset.head(20))

print(dataset.describe())

print(dataset.groupby('class').size())

dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)

pyplot.show()

dataset.hist()

pyplot.show()

scatter\_matrix(dataset)

pyplot.show()

*# step 3*

array = dataset.values

X = array[:, 0:4]

y = array[:, 4]

X\_train, X\_validation, y\_train, y\_validation = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

*# Завантажуємо алгоритми моделі*

models = []

models.append(('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')))

models.append(('LDA', LinearDiscriminantAnalysis()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('CART', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC(gamma='auto')))

*# оцінюємо модель на кожній ітерації*

results = []

names = []

*for* name, model *in* models:

    kfold = StratifiedKFold(n\_splits=10, random\_state=1, shuffle=True)

    cv\_results = cross\_val\_score(model, X\_train, y\_train, cv=kfold, scoring='accuracy')

    results.append(cv\_results)

    names.append(name)

    print('%s: %f (%f)' % (name, cv\_results.mean(), cv\_results.std()))

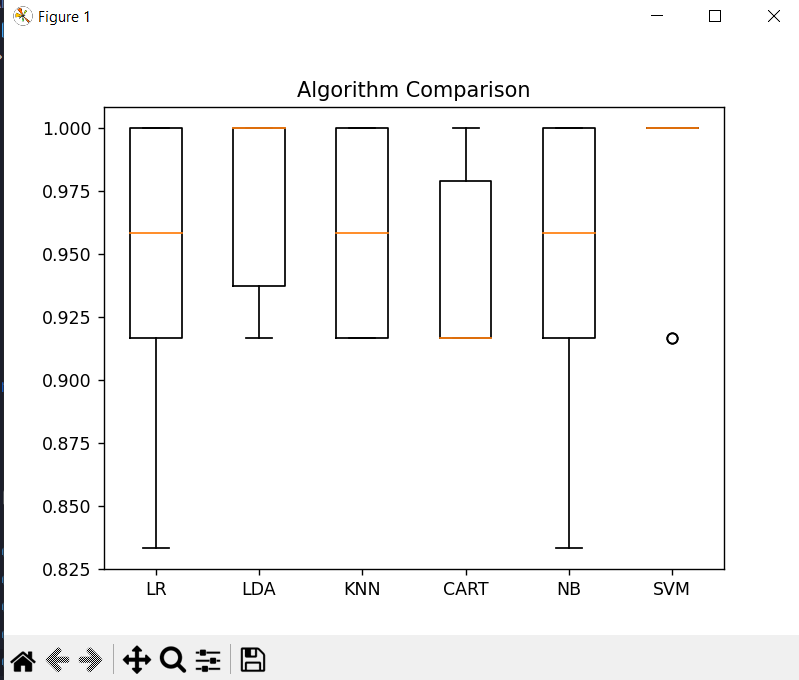
*# Порівняння алгоритмів*

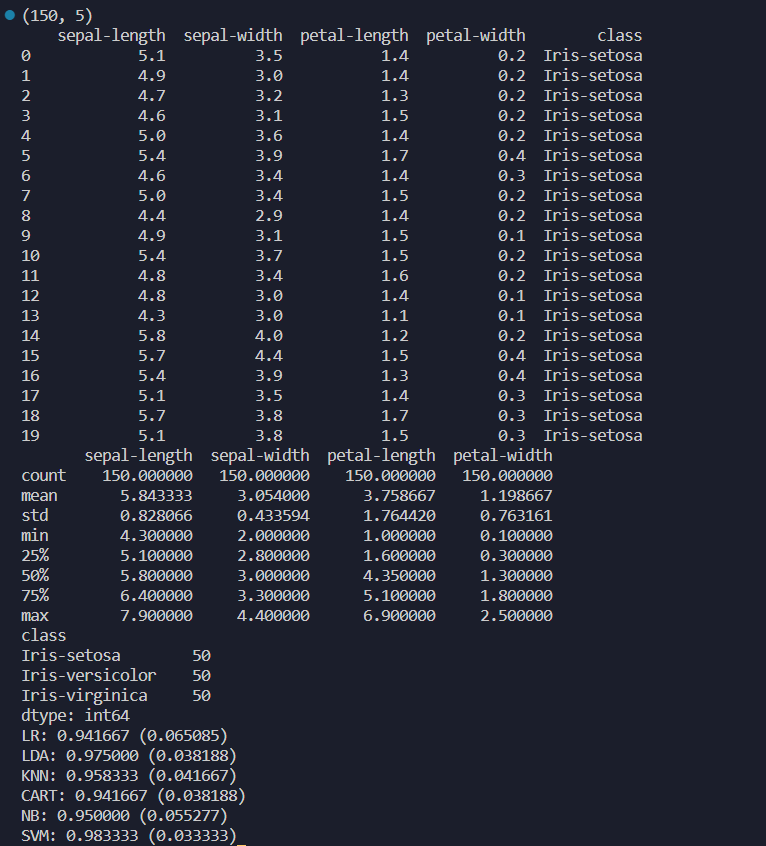
pyplot.boxplot(results, labels=names)

pyplot.title('Algorithm Comparison')

pyplot.show()

Результат:





SVM має найвищий показник точності (Accuracy) - 98.33%. Таким чином, на основі цих результатів можна вважати, що метод опорних векторів (SVM) найкраще впорався з завданням класифікації набору даних.аними.

Лістинг програми LR\_2\_task\_3.py:

*from* pandas *import* read\_csv

*from* pandas.plotting *import* scatter\_matrix

*from* matplotlib *import* pyplot

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

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

*from* sklearn.model\_selection *import* StratifiedKFold

*from* sklearn.metrics *import* classification\_report

*from* sklearn.metrics *import* confusion\_matrix

*from* sklearn.metrics *import* accuracy\_score

*from* sklearn.linear\_model *import* LogisticRegression

*from* sklearn.tree *import* DecisionTreeClassifier

*from* sklearn.neighbors *import* KNeighborsClassifier

*from* sklearn.discriminant\_analysis *import* LinearDiscriminantAnalysis

*from* sklearn.naive\_bayes *import* GaussianNB

*from* sklearn.svm *import* SVC

*import* numpy *as* np

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"

names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

dataset = read\_csv(url, names=names)

print(dataset.shape)

print(dataset.head(20))

print(dataset.describe())

print(dataset.groupby('class').size())

dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)

pyplot.show()

dataset.hist()

pyplot.show()

scatter\_matrix(dataset)

pyplot.show()

*# step 3*

array = dataset.values

X = array[:, 0:4]

y = array[:, 4]

X\_train, X\_validation, y\_train, y\_validation = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

*# Завантажуємо алгоритми моделі*

models = []

models.append(('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')))

models.append(('LDA', LinearDiscriminantAnalysis()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('CART', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC(gamma='auto')))

*# оцінюємо модель на кожній ітерації*

results = []

names = []

*for* name, model *in* models:

    kfold = StratifiedKFold(n\_splits=10, random\_state=1, shuffle=True)

    cv\_results = cross\_val\_score(model, X\_train, y\_train, cv=kfold, scoring='accuracy')

    results.append(cv\_results)

    names.append(name)

    print('%s: %f (%f)' % (name, cv\_results.mean(), cv\_results.std()))

*# Порівняння алгоритмів*

pyplot.boxplot(results, labels=names)

pyplot.title('Algorithm Comparison')

pyplot.show()

*# Створюємо прогноз на контрольній вибірці*

model = SVC(gamma='auto')

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

predictions = model.predict(X\_validation)

*# Оцінюємо прогноз*

print(accuracy\_score(y\_validation, predictions))

print(confusion\_matrix(y\_validation, predictions))

print(classification\_report(y\_validation, predictions))

X\_new = np.array([[5, 2.9, 1, 0.2]])

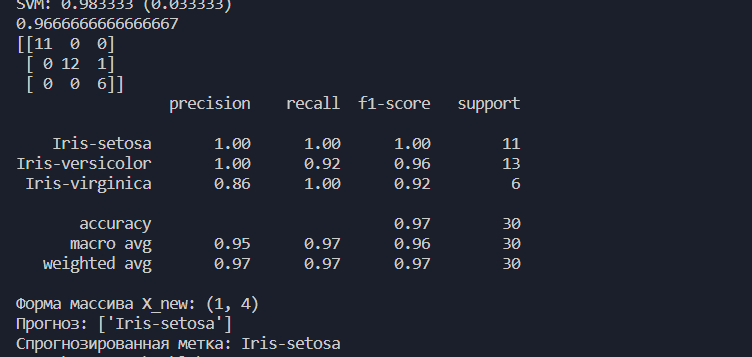
print("Форма массива X\_new: {}".format(X\_new.shape))

prediction = model.predict(X\_new)

print("Прогноз: {}".format(prediction))

print("Спрогнозированная метка: {}".format(prediction[0]))

Результа виконання програми:



Висновок: Оцінка точності (accuracy) дял обраного методу SVM – 96.67%. Квітка з кроку 8 ([5, 2.9, 1, 0.2]) належить до класу Iris-setosa.

**Завдання 2.4 Порівняння якості класифікаторів для набору даних завдання 2.1**

Лістинг програми LR\_2\_task\_4.py:

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

*from* sklearn *import* preprocessing

*from* sklearn.svm *import* LinearSVC

*from* sklearn.multiclass *import* OneVsOneClassifier

*from* pandas *import* read\_csv

*from* pandas.plotting *import* scatter\_matrix

*from* matplotlib *import* pyplot

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

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

*from* sklearn.model\_selection *import* StratifiedKFold

*from* sklearn.metrics *import* classification\_report

*from* sklearn.metrics *import* confusion\_matrix

*from* sklearn.metrics *import* accuracy\_score

*from* sklearn.linear\_model *import* LogisticRegression

*from* sklearn.tree *import* DecisionTreeClassifier

*from* sklearn.neighbors *import* KNeighborsClassifier

*from* sklearn.discriminant\_analysis *import* LinearDiscriminantAnalysis

*from* sklearn.naive\_bayes *import* GaussianNB

*from* sklearn.svm *import* SVC

input\_file = 'income\_data.txt'

X = []

y = []

count\_class1 = 0

count\_class2 = 0

max\_datapoints = 25000

*with* open(input\_file, 'r') *as* f:

*for* line *in* f.readlines():

*if* count\_class1 >= max\_datapoints and count\_class2 >= max\_datapoints:

*break*

*if* '?' in line:

*continue*

        data = line[:-1].split(', ')

        income\_class = data[-1]

*if* income\_class == '<=50K' and count\_class1 < max\_datapoints:

            X.append(data)

            count\_class1 += 1

*if* income\_class == '>50K' and count\_class2 < max\_datapoints:

            X.append(data)

            count\_class2 += 1

X = np.array(X)

label\_encoder = []

X\_encoded = np.empty(X.shape)

*for* i, item *in* enumerate(X[0]):

*if* item.isdigit():

        X\_encoded[:, i] = X[:, i]

*else*:

        current\_label\_encoder = preprocessing.LabelEncoder()

        label\_encoder.append(current\_label\_encoder)

        X\_encoded[:, i] = current\_label\_encoder.fit\_transform(X[:, i])

X = X\_encoded[:, :-1].astype(int)

y = X\_encoded[:, -1].astype(int)

X\_train, X\_validation, y\_train, y\_validation = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

models = []

models.append(('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')))

models.append(('LDA', LinearDiscriminantAnalysis()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('CART', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC(gamma='auto', max\_iter=10000)))

results = []

names = []

*for* name, model *in* models:

    kfold = StratifiedKFold(n\_splits=10, random\_state=1, shuffle=True)

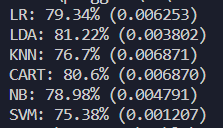
    cv\_results = cross\_val\_score(model, X\_train, y\_train, cv=kfold, scoring='accuracy')

    results.append(cv\_results)

    names.append(name)

    print('%s: %s (%f)' % (name,str(round(100 \* cv\_results.mean(), 2)) + "%",cv\_results.std()))

Результат виконання:

******

**Завдання 2.5. Класифікація даних лінійним класифікатором Ridge**

Лістинг програми LR\_2\_task\_5.py:

*import* numpy *as* np

*from* sklearn.datasets *import* load\_iris

*from* sklearn.linear\_model *import* RidgeClassifier

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

*from* sklearn *import* metrics

*from* sklearn.metrics *import* confusion\_matrix

*from* io *import* BytesIO

*import* seaborn *as* sns

*import* matplotlib.pyplot *as* plt

iris = load\_iris()

X, y = iris.data, iris.target

Xtrain, Xtest, ytrain, ytest = train\_test\_split(X, y, test\_size=0.3,random\_state=0)

clf = RidgeClassifier(tol=1e-2, solver="sag")

clf.fit(Xtrain, ytrain)

ypred = clf.predict(Xtest)

print('Accuracy:', np.round(metrics.accuracy\_score(ytest, ypred), 4))

print('Precision:', np.round(metrics.precision\_score(ytest, ypred,average='weighted'), 4))

print('Recall:', np.round(metrics.recall\_score(ytest, ypred, average='weighted'),4))

print('F1 Score:', np.round(metrics.f1\_score(ytest, ypred, average='weighted'),4))

print('Cohen Kappa Score:', np.round(metrics.cohen\_kappa\_score(ytest, ypred), 4))

print('Matthews Corrcoef:', np.round(metrics.matthews\_corrcoef(ytest, ypred), 4))

print('\t\tClassification Report:\n', metrics.classification\_report(ypred, ytest))

mat = confusion\_matrix(ytest, ypred)

sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)

plt.xlabel('True label')

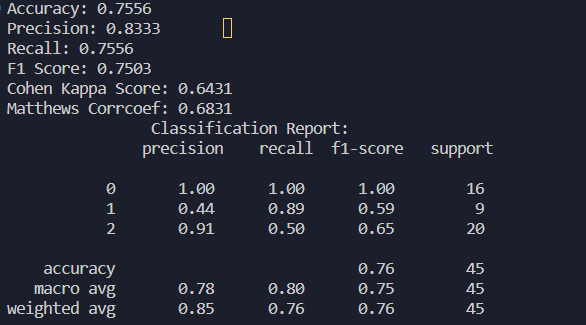
plt.ylabel('Predicted label')

plt.savefig("Confusion.jpg")

f = BytesIO()

plt.savefig(f, format="svg")

Результат виконання:



У класифікаторі Ridge були використані налаштування tol=1e-2 – точність, solver="sag" – розв'язник Stochastic Average Gradient.

Показники якості:

Акуратність – 75.56%

Точність – 83.33%

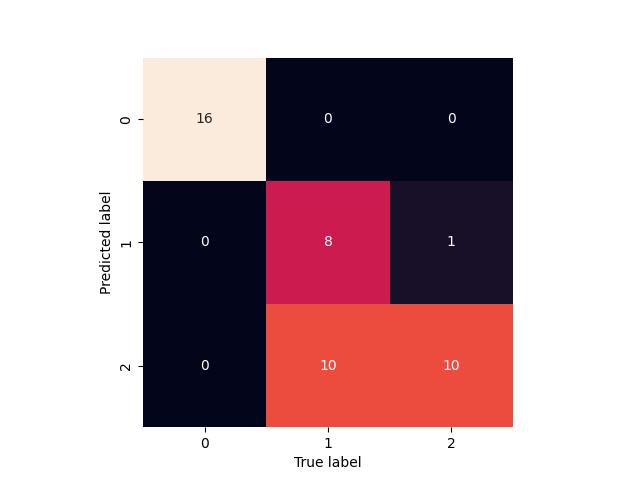
Повнота – 75.56%

f-міра – 75.03%

коефіцієнт Коена Каппа – 64.31%

коефіцієнт кореляції Метьюза – 68.31%

Confusion.jpg:



Це матриця плутанини (confusion matrix) – візуалізація результату класифікації моделі машинного навчання. На осі x відображаються значення "True label" (правильних міток), а на осі y відображаються значення "Predicted label" (прогнозованих міток)

Коефіцієнт Коена Каппа вимірює ступінь узгодженості між прогнозованими класифікаціями і дійсними класами у випадку, коли існує випадкова вірогідність вибору класу.

Коефіцієнт кореляції Метьюза вимірює якість класифікаційної моделі, якщо дані мають баланс або дисбаланс класів.

***Github:*** [***https://github.com/mtvi/ipz202\_Shcherback\_lab2***](https://github.com/mtvi/ipz202_Shcherback_lab2)

***Висновки:*** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідили різні методи класифікації даних та навчилися їх порівнювати.