Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 3 з дисципліни
«Прикладні задачі машинного навчаннях»

«Класифікація, регресія і кластеризація з використанням бібліотеки scikit-learn»

Варіант 7

Виконав студент: ІП-12 Васильєв Єгор Костянтинович

Перевірив: Нестерук Андрій Олександрович

Київ 2023

Лабораторна робота №3

Тема: Класифікація, регресія і кластеризація з використанням бібліотеки scikit-learn

Постановка завдання:

1) Первинна обробка завантажених даних середньо січневих температур.
2) Пошук регресійної прямої та тестування моделі.
3) Прогнозування температур.
4) Візуалізація набору даних та регресійної прямої.
5) Порівняння результату з минулою лабораторною роботою.
6) Класифікація згенерованого набору даних класифікатором SVC.
7) Порівняння класифікаторів KNeighborsClassifier, GaussianNB та SVC.

Хід роботи:

- Первинна обробка даних

	Contiguous U.S.	Average Temperature	January
0	Units: Degrees Fahrenheit	nan	nan
1	Base Period: 1901-2000	nan	nan
2	Missing: -99	nan	nan
3	Date	Value	Anomaly
4	189501	26.69	-3.43
5	189601	31.48	1.36
6	189701	28.17	-1.95
7	189801	30.67	0.55
8	189901	29.68	-0.44
9	190001	34.11	3.99
10	190101	31.62	1.50
11	190201	30.06	-0.06
12	190301	30.88	0.76
13	190401	27.55	-2.57

	Date	Temperature	Anomaly
0	1895	-2.95000	-3.43
1	1896	-0.28889	1.36
2	1897	-2.12778	-1.95
3	1898	-0.73889	0.55
4	1899	-1.28889	-0.44
5	1900	1.17222	3.99
6	1901	-0.21111	1.50
7	1902	-1.07778	-0.06
8	1903	-0.62222	0.76
9	1904	-2.47222	-2.57
10	1905	-2.72222	-3.02
11	1906	0.48889	2.76

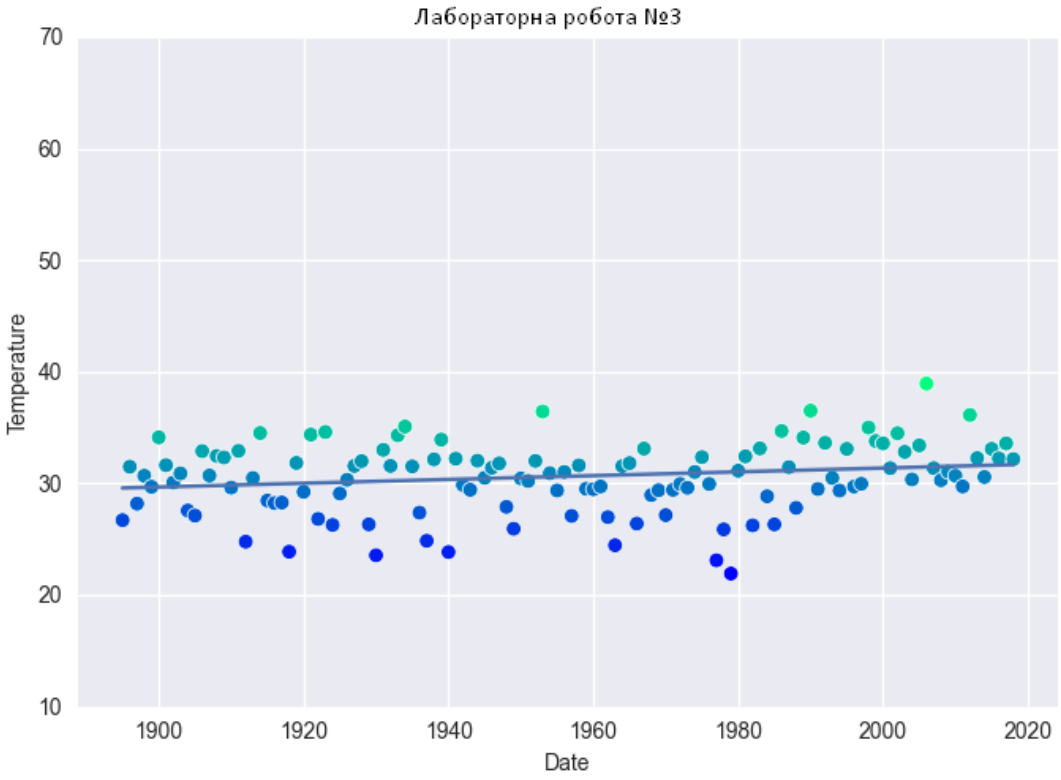
- Порівняння прогнозованих температур та фактичних

```
predicted: 30.82, expected: 28.94, difference: -6.09%
predicted: 31.55, expected: 29.71, difference: -5.82%
predicted: 30.07, expected: 26.26, difference: -12.66%
predicted: 30.29, expected: 24.84, difference: -17.99%
predicted: 30.99, expected: 25.86, difference: -16.54%
predicted: 30.63, expected: 27.07, difference: -11.62%
predicted: 30.02, expected: 34.36, difference: 14.47%
```

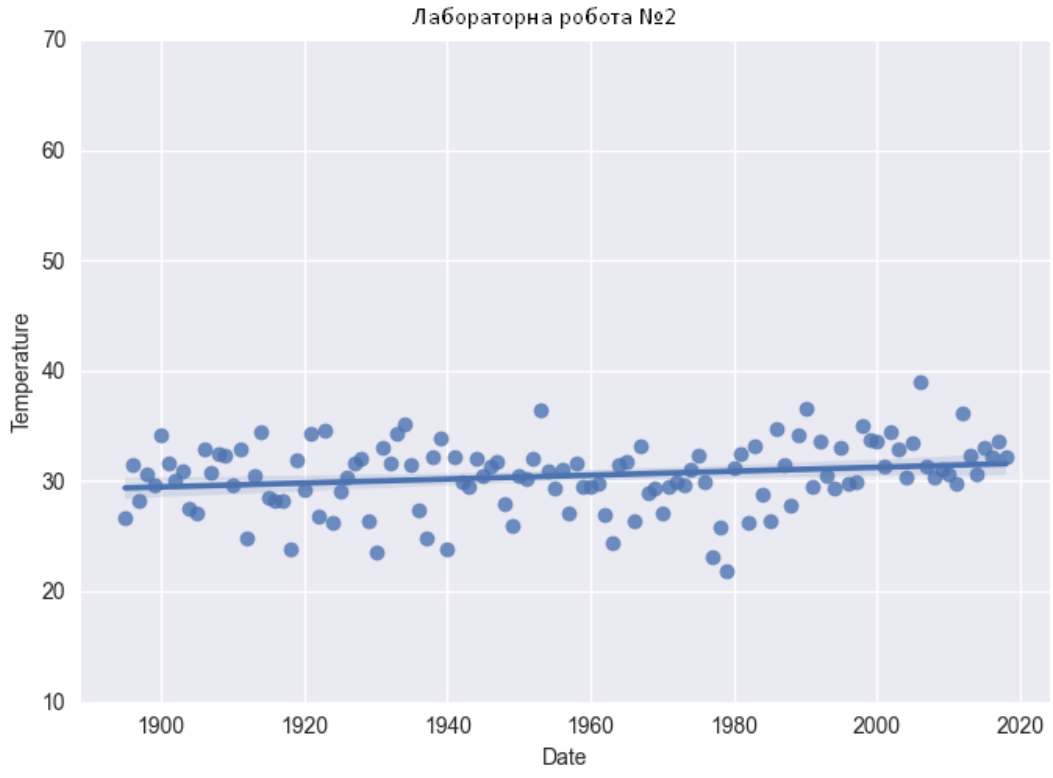
- Передбачення температур на 2019 та 1890 роки

```
Predicted temperature in 2019: [31.68369771]
Predicted temperature in 1890: [29.48934931]
```

- Візуалізація даних

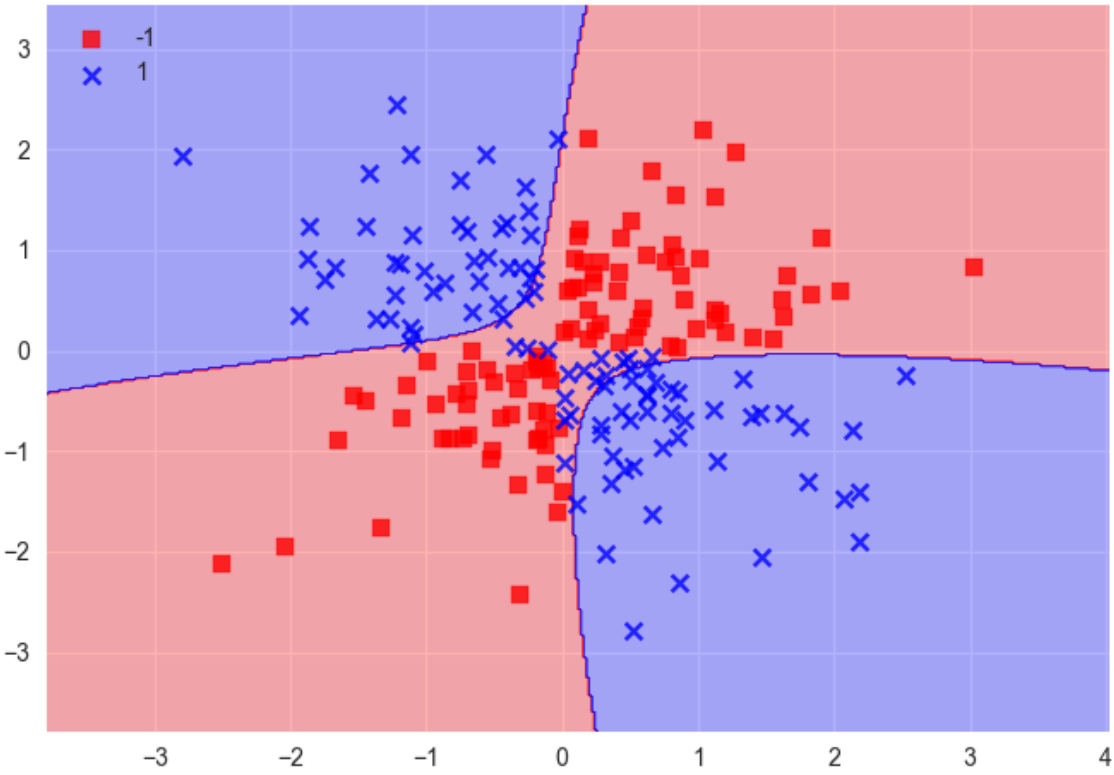


- Порівняння з регресійної прямою минулої лабораторної роботи

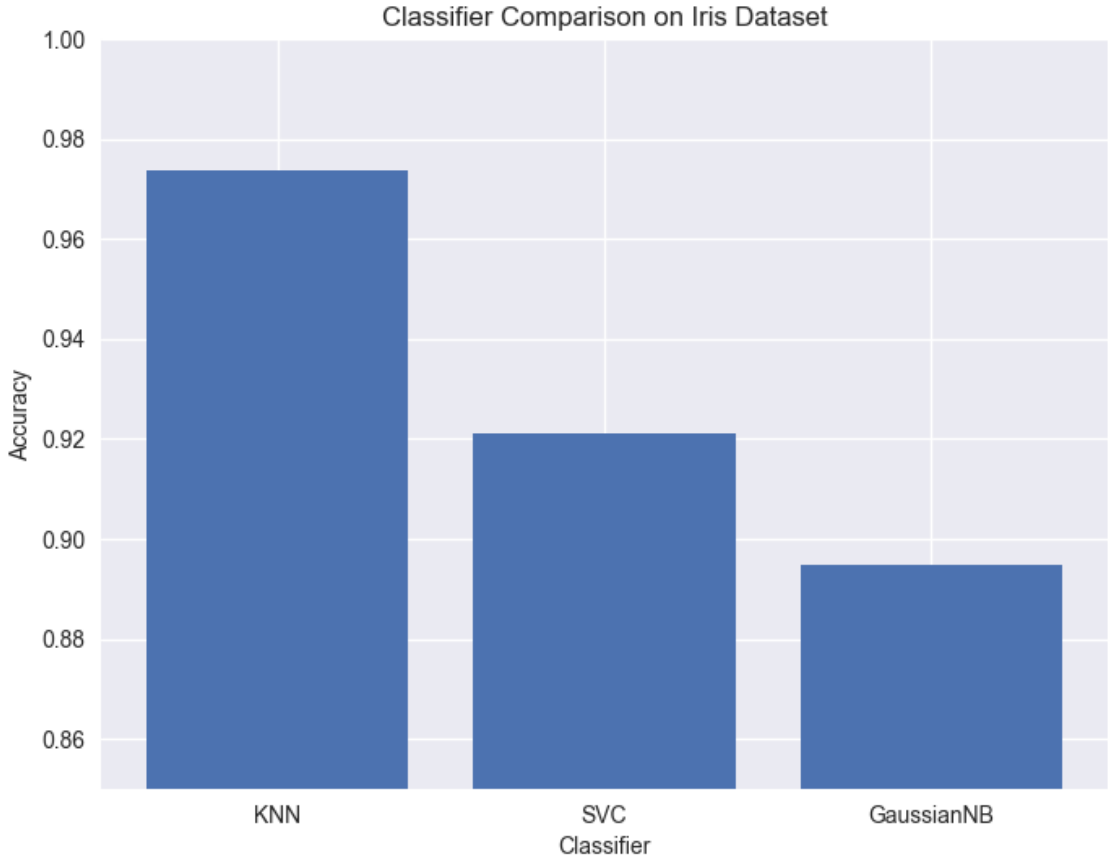


Як бачимо, лінійні регресії ідентичні, оскільки в обох бібліотеках використовується той самій метод пошуку прямої.

- Візуалізація класифікації опорних векторів згенерованих даних



- Порівняння класифікаторів на вбудованому наборі даних



Використаний алгоритм: KNeighborsClassifier використовує алгоритм k-найближчих сусідів для класифікації точок даних, SVC використовує машинний алгоритм опорних векторів, а GaussianNB використовує наївний Баєсів алгоритм. Ці алгоритми мають різні сильні та слабкі сторони та можуть по-різному працювати з різними типами наборів даних.

Швидкість навчання: KNeighborsClassifier і GaussianNB відносно швидкі для навчання, оскільки вони не потребують обчислення великої кількості параметрів. SVC, з іншого боку, може бути повільним для навчання, особливо на великих наборах даних.

Точність прогнозування: точність прогнозування кожного класифікатора може відрізнятися залежно від характеру набору даних, що аналізується. Загалом SVC добре працює на широкому діапазоні наборів даних і часто вважається хорошим універсальним вибором. KNeighborsClassifier також може добре працювати, особливо на менших наборах даних, але може бути більш чутливим до шуму в даних. GaussianNB часто використовується для класифікації тексту та інших програм, що включають розріджені дані.

Параметри налаштування: кожен класифікатор має інший набір параметрів налаштування, які можна налаштувати для оптимізації його продуктивності на певному наборі даних. Наприклад, KNeighborsClassifier має параметр n_neighbors, який керує кількістю сусідів, які використовуються в класифікації, тоді як SVC має параметр C, який керує штрафом за неправильну класифікацію. Налаштування цих параметрів може бути важливим кроком в оптимізації продуктивності кожного класифікатора.

Висновок

Було поглиблено знання бібліотек pandas, matplotlib, seaborn, numpy та досліджено різні типи класифікаторів Scikit-Learn; було побудовано лінійну регресію та візуалізовано зміну середньої температури січня континентальної Америки; було досліджено класифікатор на базі методу опорних векторів для згенерованого набору даних; було порівняно декілька класифікаційних оцінювачів: KNeighborsClassifier, SVC та GaussianNB для вбудованого в scikit-learn Iris датасету; таким чином було вивчену тему «Класифікація, регресія і кластеризація з використанням бібліотеки scikit-learn»

```python
import ...

pd.set_option('display.precision', 2)
plt.style.use('seaborn-v0_8')

initial_df = pd.read_csv(r'F:\Egor\Уроки\Машинне навчання\Ла62\1895-2018.csv')
nyc = initial_df.copy()
nyc.columns = ['Date', 'Temperature', 'Anomaly']
nyc.drop(index=range(4), inplace=True)
nyc.reset_index(drop=True, inplace=True)
nyc = nyc.astype({'Date': 'intб4', 'Temperature': 'float'})
nyc.loc[:, 'Date'] = nyc.loc[:, 'Date'].floordiv(100)
print(nyc.head())
dates = nyc.loc[:, 'Date'].values
temps = nyc.loc[:, 'Temperature'].values

X_train, X_test, Y_train, Y_test = train_test_split(nyc.loc[:, 'Date'].values.reshape(-1, 1),
                                                    nyc.loc[:, 'Temperature'].values, random_state=11)
linear_regression = LinearRegression()
linear_regression.fit(X=X_train, y=Y_train)
k = linear_regression.coef_
b = linear_regression.intercept_
predicted = linear_regression.predict(X_test)
expected = Y_test
for p, e in zip(predicted[::5], expected[::5]):
    print(f'predicted: {p:.2f}, expected: {e:.2f}, difference: {(e-p)/p*100:.2f}%')
predict = (lambda x: linear_regression.coef_ * x + linear_regression.intercept_)
print('Predicted temperature in 2019:', predict(2019))
print('Predicted temperature in 1890:', predict(1890))
axes = sns.scatterplot(data=nyc, x='Date', y='Temperature', hue='Temperature', palette='winter', legend=False)
axes.set_ylim(10, 70)
x = np.array([min(nyc.loc[:, 'Date'].values), max(nyc.loc[:, 'Date'].values)])
y = predict(x)
line = plt.plot(x, y)
plt.show()
# part 2
np.random.seed(1)
X_xor = np.random.randn(200, 2)
Y_xor = np.logical_xor(X_xor[:, 0] > 0, X_xor[:, 1] > 0)
Y_xor = np.where(Y_xor, 1, -1)
plt.scatter(X_xor[Y_xor == 1, 0], X_xor[Y_xor == 1, 1], c='b', marker='x', label='1')
plt.scatter(X_xor[Y_xor == -1, 0], X_xor[Y_xor == -1, 1], c='r', marker='s', label='-1')
plt.xlim(-3, 3)
plt.ylim(-3, 3)
plt.legend(loc='best')
plt.show()
```

Вихідний код

```python
def plot_decision_regions(X, Y, classifier, test_idx=None, resolution=0.02):
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(Y))])
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution), np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    for idx, c1 in enumerate(np.unique(Y)):
        plt.scatter(x=X[Y == c1, 0], y=X[Y == c1, 1], alpha=0.8, c=colors[idx], marker=markers[idx], label=c1)

    if test_idx:
        X_test, Y_test = X[test_idx, :], Y[test_idx]
        plt.scatter(X_test[:, 0], X_test[:, 1], c='', edgecolor='black', alpha=1.0, linewidth=1, marker='0', s=100)


svm = SVC(kernel='rbf', random_state=1, gamma=0.10, C=10.0)
svm.fit(X_xor, Y_xor)
plot_decision_regions(X_xor, Y_xor, classifier=svm)
plt.legend(loc='upper left')
plt.show()

# part 3

iris = load_iris()

# Split dataset into training and testing data
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, random_state=11)

# Initialize classifiers
knn = KNeighborsClassifier()
svc = SVC()
gnb = GaussianNB()

# Train classifiers
knn.fit(X_train, y_train)
svc.fit(X_train, y_train)
gnb.fit(X_train, y_train)

# Make predictions on testing data
knn_prediction = knn.predict(X_test)
svc_prediction = svc.predict(X_test)
gnb_prediction = gnb.predict(X_test)

# Calculate accuracy scores
knn_accuracy = accuracy_score(y_test, knn_prediction)
svc_accuracy = accuracy_score(y_test, svc_prediction)
gnb_accuracy = accuracy_score(y_test, gnb_prediction)

# Print accuracy scores
print(f'KNN Accuracy: {knn_accuracy:.3f}')
print(f'SVC Accuracy: {svc_accuracy:.3f}')
print(f'GaussianNB Accuracy: {gnb_accuracy:.3f}')

# Plot results
labels = ['KNN', 'SVC', 'GaussianNB']
accuracies = [knn_accuracy, svc_accuracy, gnb_accuracy]
plt.figure(figsize=(8, 6))
plt.bar(labels, accuracies)
plt.title('Classifier Comparison on Iris Dataset')
plt.xlabel('Classifier')
plt.ylabel('Accuracy')
plt.ylim(0.85, 1)
plt.show()
```