аНАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра інформатики та програмної інженерії

Звіт до лабораторної роботи №4

з курсу

«Машине навчання»

студента 2 курсу
групи ІТ-02
Макарова Іллі Сергійовича

Викладач:
Оніщенко В.

Київ – 2022

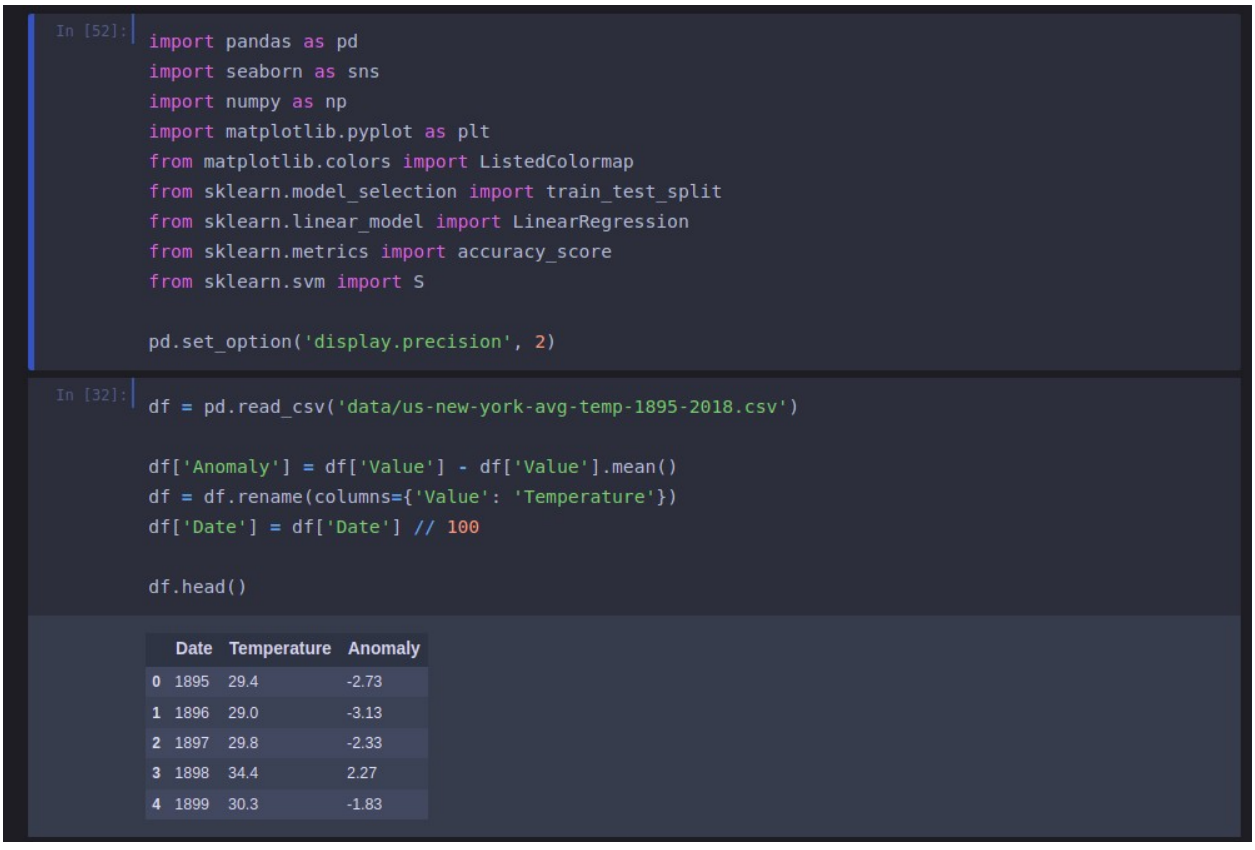
Тема: класифікація, регресія і кластеризація з використанням бібліотеки scikit-learn

Виконання:

Итак, у задания было две части. Первая, на 3 балла, где нужно было просто переписать уже готовый код и запустить его. И вторая, более интересная, где нужно было выполнить класификацию объектов тремя моделями, и посмотреть что получилось.

Для начала я вставлю скрины задания на 3 балла, особено комментировать не буду, так как вряд-ли там что то можно добавить. Далле же буде вторая часть работы, уже с моими комментариями.

ПЕРВАЯ ЧАСТЬ



In [52]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
from sklearn.svm import S

pd.set_option('display.precision', 2)
```

In [32]:
```
df = pd.read_csv('data/us-new-york-avg-temp-1895-2018.csv')

df['Anomaly'] = df['Value'] - df['Value'].mean()
df = df.rename(columns={'Value': 'Temperature'})
df['Date'] = df['Date'] // 100

df.head()
```

	Date	Temperature	Anomaly
0	1895	29.4	-2.73
1	1896	29.0	-3.13
2	1897	29.8	-2.33
3	1898	34.4	2.27
4	1899	30.3	-1.83

In [14]:
```
x_train, x_test, y_train, y_test = train_test_split(
    df['Date'].values.reshape(-1, 1), df['Temperature'], random_state=5
)
x_train.shape
```

```
(93, 1)
```

In [33]:
```
regression = LinearRegression()
regression.fit(x_train, y_train)

y_predicted = regression.predict(x_test)

for predicted_value, actual_value in list(zip(y_predicted, y_test))[::5]:
    print(f'Actual value: {actual_value} and predicted: {round(predicted_value, 1)}')
```

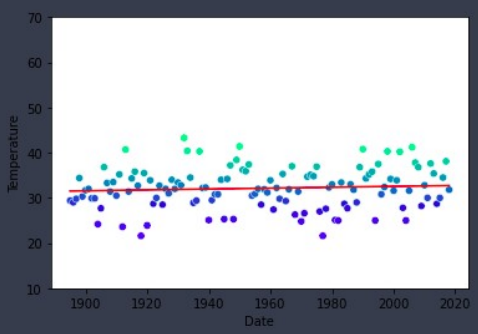
```
Actual value: 35.9 and predicted: 32.1
Actual value: 40.3 and predicted: 31.9
Actual value: 40.2 and predicted: 32.6
Actual value: 31.2 and predicted: 32.2
Actual value: 21.6 and predicted: 31.7
Actual value: 34.0 and predicted: 32.0
Actual value: 30.8 and predicted: 32.0
```

In [34]:
```
predict = lambda x: regression.coef_ * x + regression.intercept_
```

In [38]:
```
round(predict(2019)[0], 2), round(predict(1890)[0], 2)
```

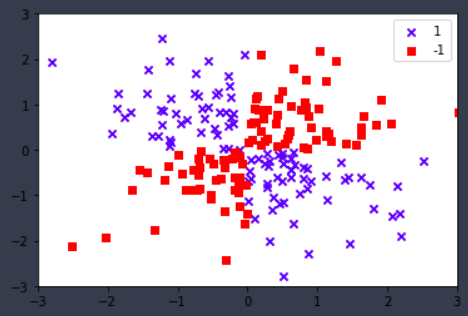
```
(32.74, 31.47)
```

In [53]:
```
axes = sns.scatterplot(
    data=df,
    x='Date',
    y='Temperature',
    hue='Temperature',
    palette='winter',
    legend=False
)

axes.set_ylim(10, 70)

x = np.array([min(df['Date'].values), max(df['Date'].values)])
y = predict(x)

plt.plot(x, y, color='red')
plt.show()
```



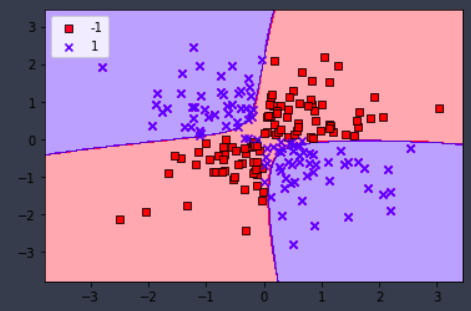
Clasification

In [58]:
```
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()
```



In [65]:
```
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.ylim(xx2.min(), xx2.max())
    plt.xlim(xx1.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,
            edgecolor='black'
        )
```

```
            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='o',
                    s=100,
                    label='Something'
                )

svm = SVC(kernel='rbf', random_state=1, gamma=0.1, 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()
```



ЧАСТЬ ВТОРАЯ

Ну а теперь часть вторая, уже более интересно.

В качестве датасета для класификации я выбрал класический датасет ирисов. Кароче цветочки будем класифицировать.

Итак, импортируем все что нам понадобиться, и скачиваем датасет

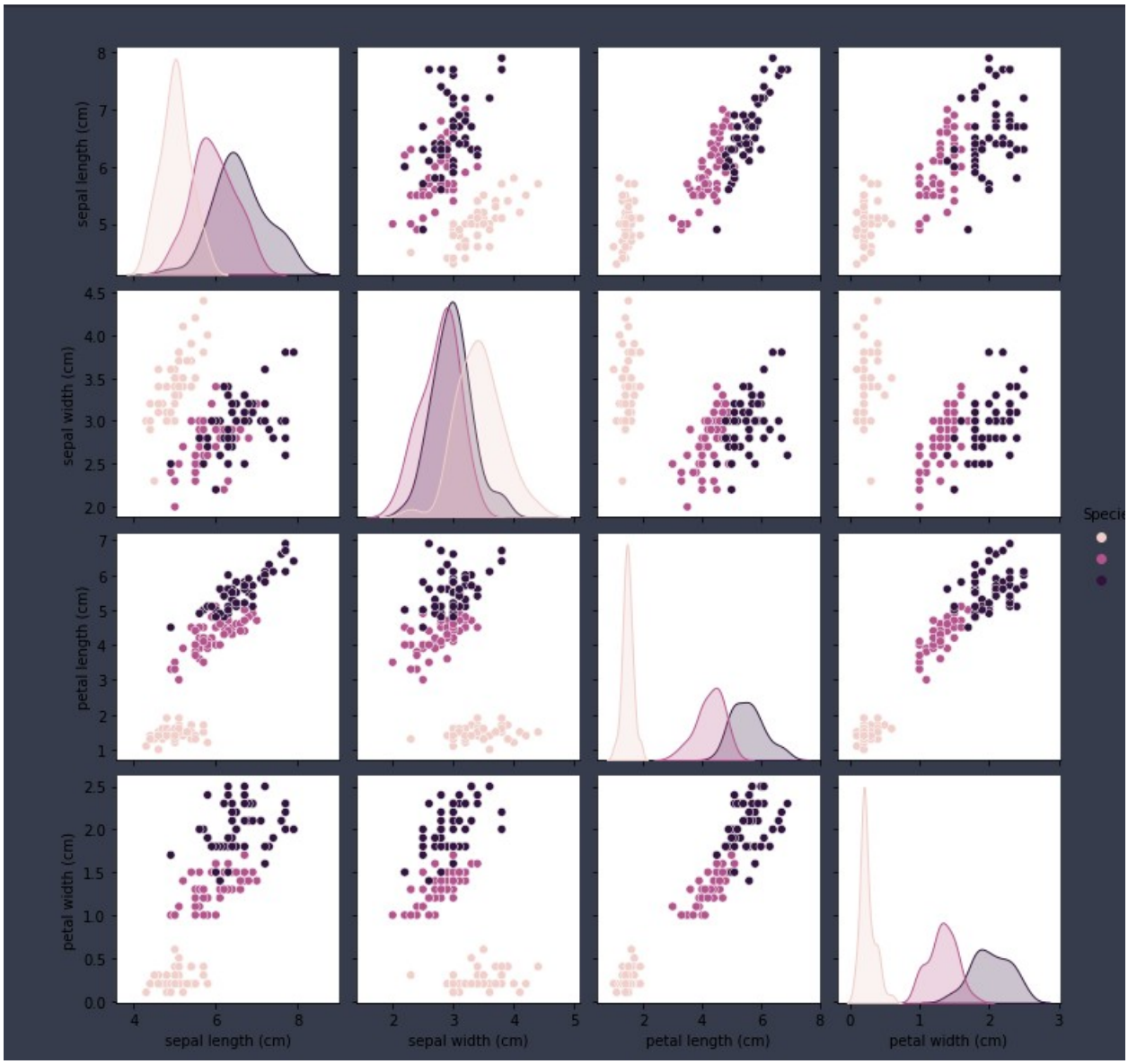
```
In [38]: import pandas as pd
         import matplotlib.pyplot as plt
         from seaborn import pairplot
         from sklearn.datasets import load_iris
         from sklearn.model_selection import train_test_split
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.metrics import accuracy_score
         from sklearn.svm import SVC
         from sklearn.naive_bayes import GaussianNB
```

```
PREPARE THE DATA
```

```
In [39]: iris_data = load_iris()
```

Такс, теперь визуализируем данные:

```
In [40]: df_iris_data = pd.DataFrame(iris_data.data, columns=iris_data.feature_names)
         df_iris_target = pd.DataFrame(iris_data.target, columns=['Species'])

         df_iris_data['Species'] = iris_data.target
         pairplot(df_iris_data, hue='Species')
         plt.show()
```



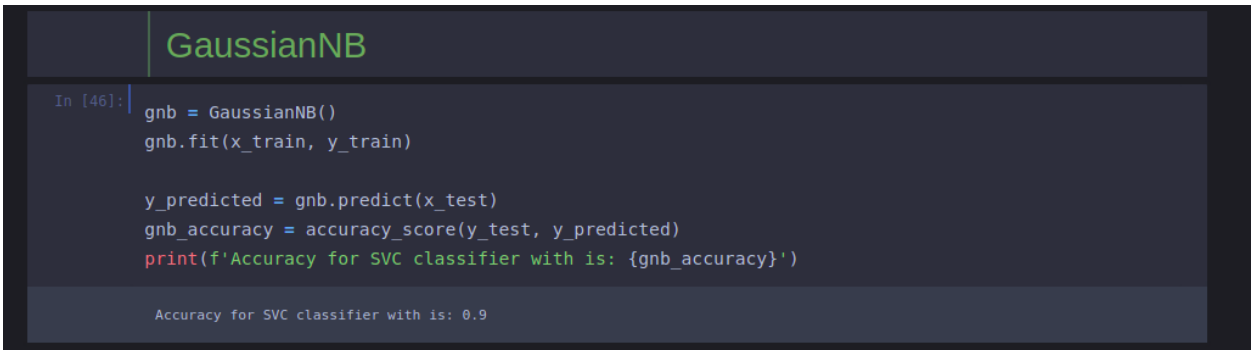
Разобъем данные на тесторые и тренировочные:

```
In [41]: iris_data.target_names, iris_data.feature_names

(array(['setosa', 'versicolor', 'virginica'], dtype='<U10'),
 ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'])

In [42]: x_train, x_test, y_train, y_test = train_test_split(
             iris_data.data,
             iris_data.target,
             test_size=0.2,
             random_state=5
         )

In [43]: x_train.shape, x_test.shape

((120, 4), (30, 4))
```

Ну и дело за малым, создаем три модели, и сравниваем их точность

```
KNN

In [44]: knn = KNeighborsClassifier(n_neighbors=6)
         knn.fit(x_train, y_train)

         y_predicted = knn.predict(x_test)
         knn_accuracy = accuracy_score(y_test, y_predicted)

         print(f'Accuracy for KNN classifier with k={6} is: {knn_accuracy}')

Accuracy for KNN classifier with k=6 is: 0.9666666666666667

SVC
```

```
SVC

In [45]: svc = SVC(kernel='linear')
         svc.fit(x_train, y_train)

         y_predicted = svc.predict(x_test)
         svc_accuracy = accuracy_score(y_test, y_predicted)
         print(f'Accuracy for SVC classifier with is: {svc_accuracy}')

Accuracy for SVC classifier with is: 0.9333333333333333
```



Итого KNN показал себя лучше всех :)

В заключение могу сказать, что работа была довольно не плохой, я познакомился с тремя популярными алгоритмами класификации, не много потыкал их создание, и тестирование качества модели.