|  |  |  |
| --- | --- | --- |
|  | |  |
| МИНОБРНАУКИ РОССИИ | |  |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** | |  |
|  | Институт информационных технологий (ИТ) | |
|  | Кафедра прикладной математики | |

|  |  |
| --- | --- |
| **Отчет по практической работе №5.** | |
| **по дисциплине** | |
| **«Технологии и инструментарий анализа больших данных»** | |
|  | |
| Выполнил студент группы ИКБО- 22-20 | Гологузов К.А. |
| Принял | Парамонов А.А. |

Москва 2023

Решение

1. Найти данные для классификации. Данные в группе повторяться не должны. Предобработать данные, если это необходимо.



Рис.1 – Датасет

2. Изобразить гистограмму, которая показывает баланс классов. Сделать выводы.

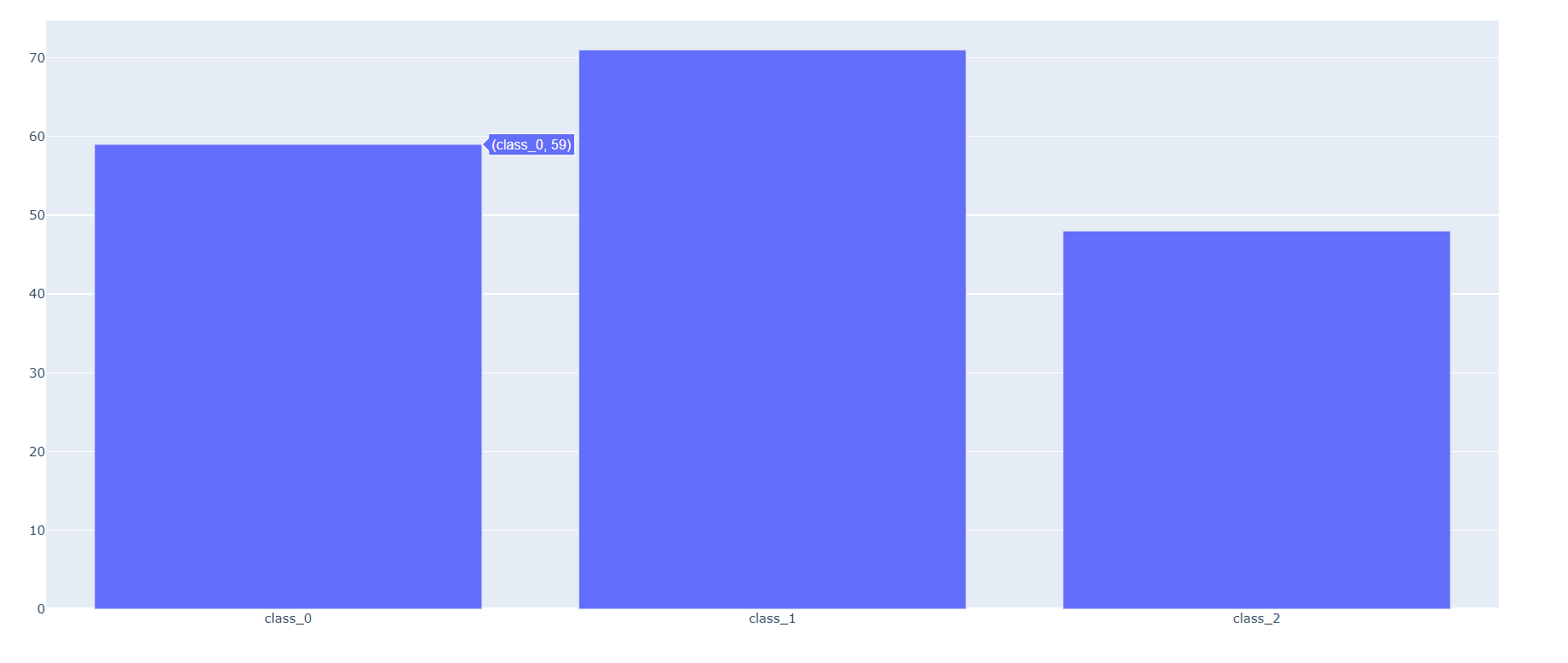


Рис. 2 – Гистограммы классов

На графике видно, что данные достаточно сбалансированы. К классу 0 относится 59 объектов, к классу 1 относится 71 объект, к классу 2 относится 48 объектов.

3. Разбить выборку на тренировочную и тестовую. Тренировочная для обучения модели, тестовая для проверки ее качества.



Рис. 3 – Разбиение выборки

4. Применить алгоритмы классификации: логистическая регрессия, SVM, KNN. Построить матрицу ошибок по результатам работы моделей (использовать confusion\_matrix из sklearn.metrics).

Логистическая регрессия:

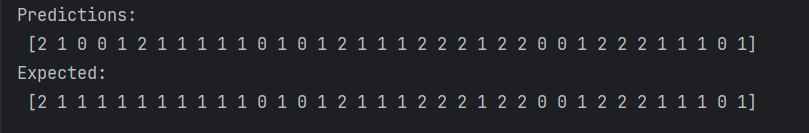


Рис. 4 – Результат работы

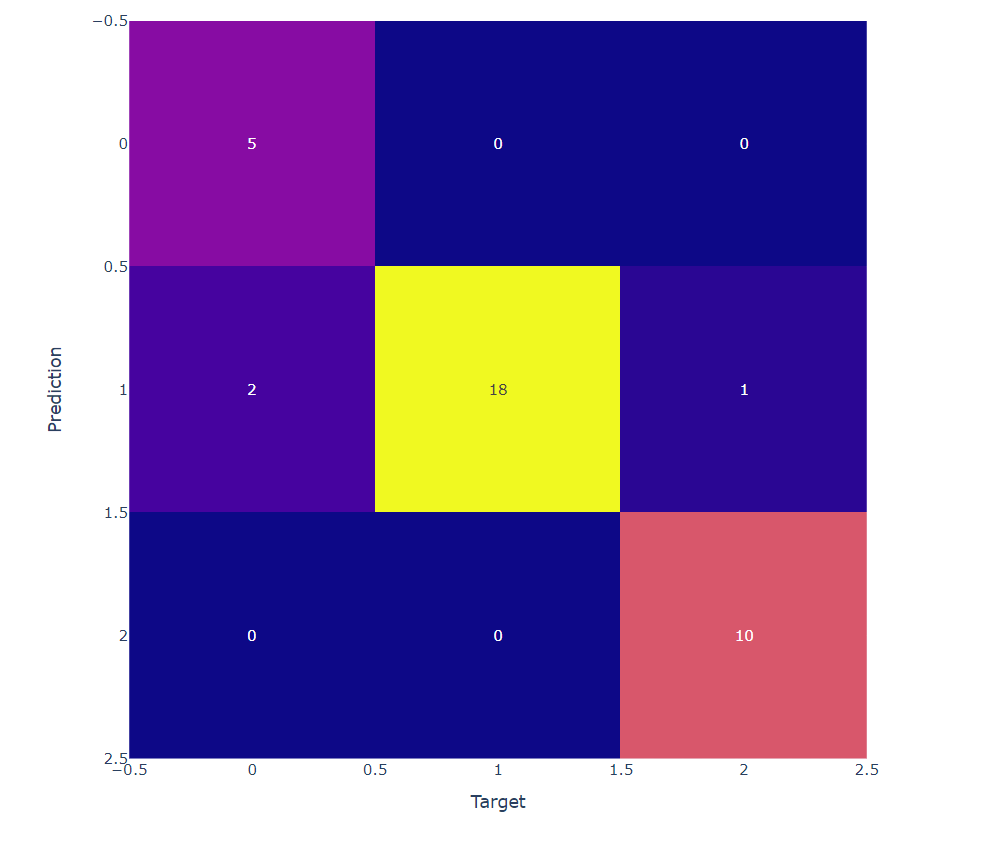


Рис. 5 – Матрица ошибок логистической регрессии

SVM (метод опорных векторов):

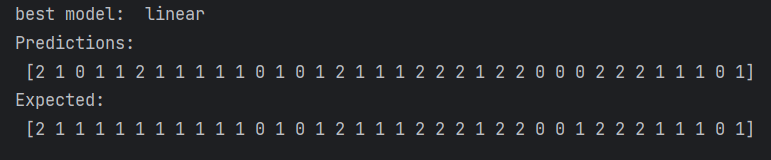


Рис. 6 – Результат работы

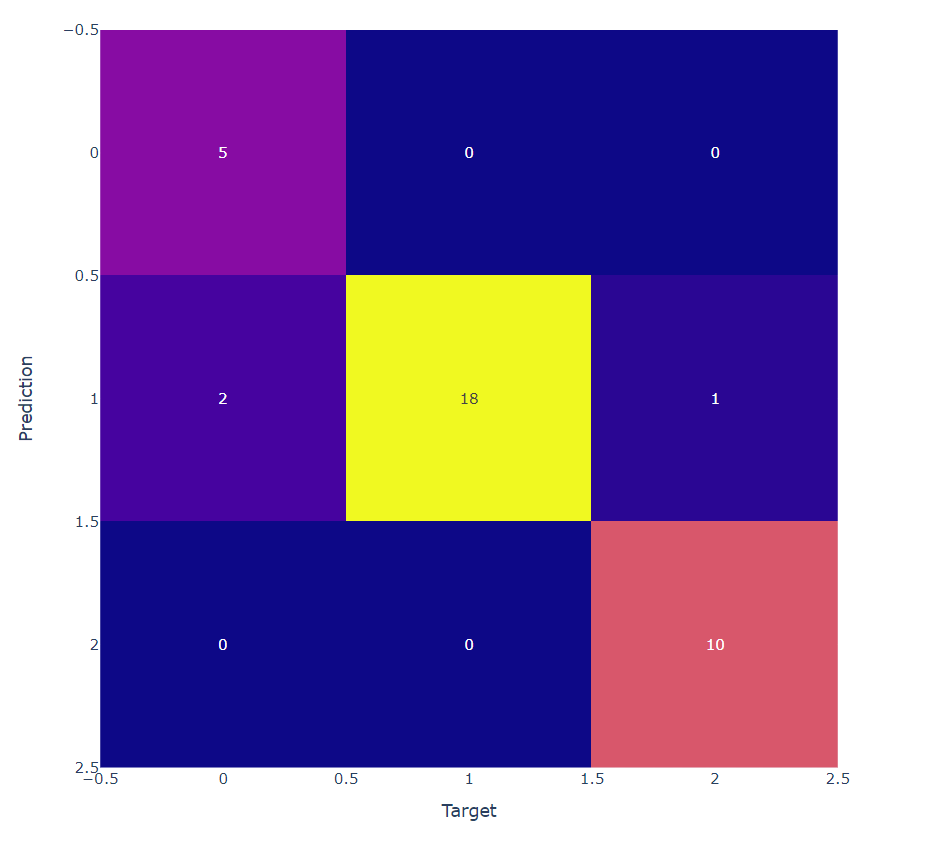


Рис. 7 – Матрица ошибок SVM

KNN:

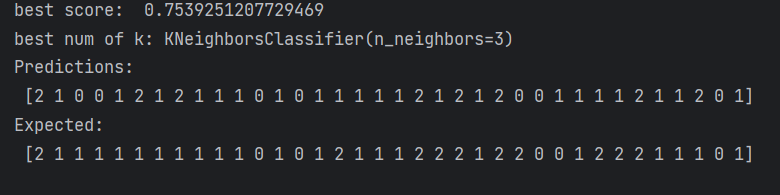


Рис. 8 – Результат работы

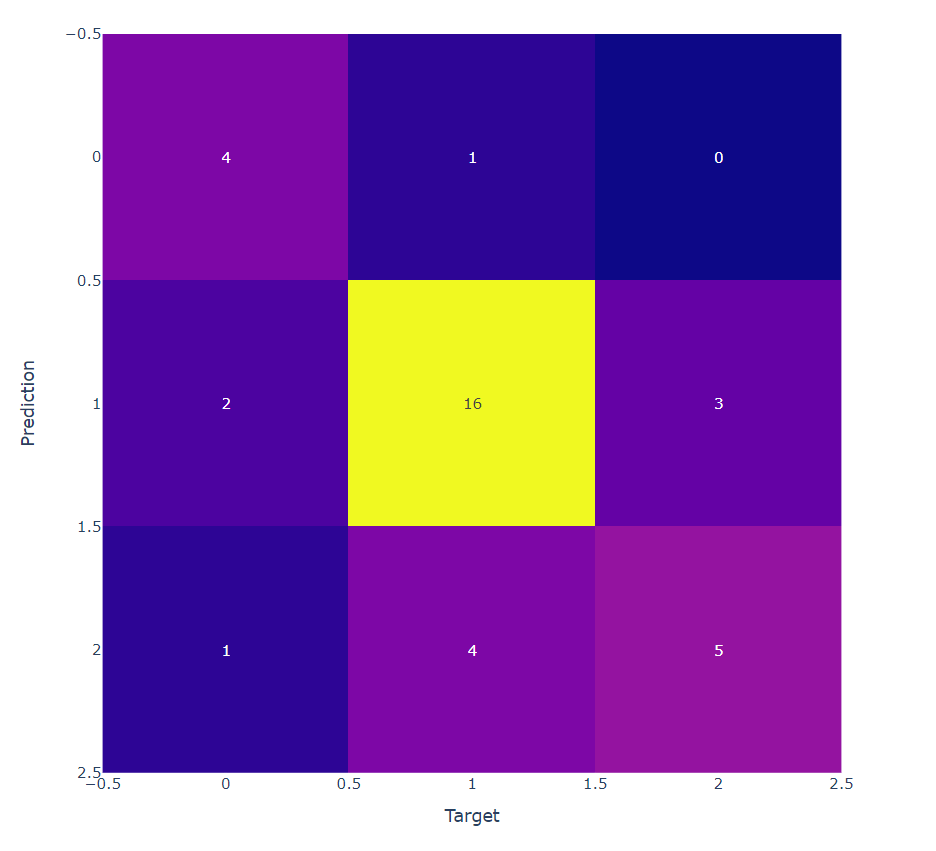


Рис. 9 – Матрица ошибок KNN

5. Сравнить результаты классификации, используя accuracy, precision, recall и f1-меру (можно использовать classification\_report из sklearn.metrics). Сделать выводы.

Логистическая регрессия:

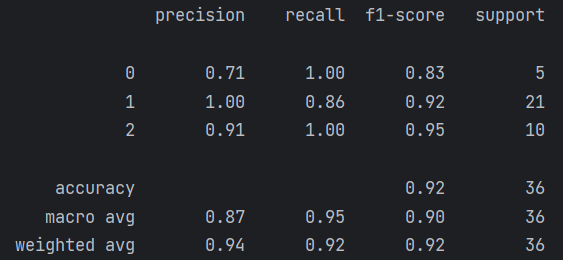


Рис. 10 – Результат классификации логистической регрессии

SVM:

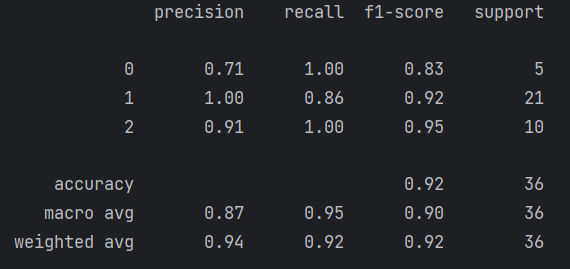


Рис. 11 – Результат классификации SVM

KNN:

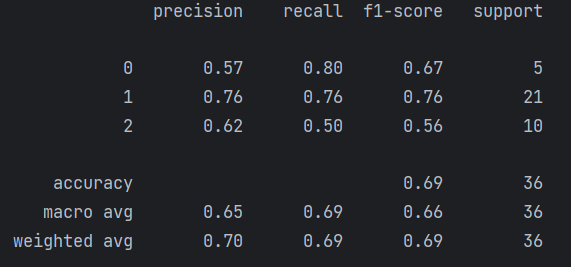


Рис. 12 – Результат классификации KNN

Исходя из результатов, мы можем сделать вывод, что наименее точным алгоритмом классификации для данного датасета является KNN.

ПРИЛОЖЕНИЕ

Приложение 1- Код программы

import numpy as np  
import sklearn  
from sklearn.datasets import load\_wine  
from sklearn.model\_selection import train\_test\_split, GridSearchCV  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.svm import SVC  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import confusion\_matrix, classification\_report  
import plotly.graph\_objs as go  
import plotly.express as px  
  
  
data = load\_wine(as\_frame=True)  
predictords = data.data  
target = data.target  
target\_names = data.target\_names  
  
fig = go.Figure(data=[go.Bar(x = target\_names, y = [len(target.loc[target == 0]), len(target.loc[target == 1]),  
 len(target.loc[target == 2])])])  
fig.show()  
  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(predictords, target, train\_size=0.8,  
 shuffle=True, random\_state=271)  
print("x train", x\_train.shape, "x test", x\_test.shape, "y train", y\_train.shape, "y test", y\_test.shape)  
# REGRESSION  
model = LogisticRegression(random\_state=271)  
model.fit(x\_train, y\_train)  
y\_predict = model.predict(x\_test)  
print("Predictions:\n", y\_predict)  
print("Expected:\n", np.array(y\_test))  
print(classification\_report(y\_test, y\_predict))  
fig = px.imshow(confusion\_matrix(y\_test, y\_predict), text\_auto=True)  
fig.update\_layout(xaxis\_title = "Target", yaxis\_title = "Prediction")  
fig.show()  
  
# SVM  
param\_kernel = ("linear", "rbf", "poly", "sigmoid")  
parameters = {"kernel": param\_kernel}  
model = SVC()  
grid\_search\_svm = GridSearchCV(estimator=model, param\_grid=parameters, cv=6)  
grid\_search\_svm.fit(x\_train, y\_train)  
best\_model = grid\_search\_svm.best\_estimator\_  
print("best model: ", best\_model.kernel)  
svm\_preds = best\_model.predict(x\_test)  
print("Predictions:\n", svm\_preds)  
print("Expected:\n", np.array(y\_test))  
print(classification\_report(y\_test, svm\_preds))  
fig = px.imshow(confusion\_matrix(y\_test, svm\_preds), text\_auto=True)  
fig.update\_layout(xaxis\_title = "Target", yaxis\_title = "Prediction")  
fig.show()  
  
# KNN  
number\_of\_neighbours = np.arange(3, 10)  
model\_KNN = KNeighborsClassifier()  
params = {"n\_neighbors": number\_of\_neighbours}

Продолжение – Приложение 1

grid\_search = GridSearchCV(estimator=model\_KNN, param\_grid=params, cv = 6)  
grid\_search.fit(x\_train, y\_train)  
print("best score: ", grid\_search.best\_score\_)  
print("best num of k:", grid\_search.best\_estimator\_)  
knn\_preds = grid\_search.predict(x\_test)  
print("Predictions:\n", knn\_preds)  
print("Expected:\n", np.array(y\_test))  
print(classification\_report(y\_test, knn\_preds))  
fig = px.imshow(confusion\_matrix(y\_test, knn\_preds), text\_auto=True)  
fig.update\_layout(xaxis\_title = "Target", yaxis\_title = "Prediction")  
fig.show()