"بسمه تعالی"

گزارش تمرین عملی چهارم یادگیری ماشین – زهرا لطیفی – ۹۹۲۳۰۶۹

سوال اول

الف) SVM خطی

ابتدا کتابخانه‌های لازم ایمپورت شده‌اند.

```
# Import the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split, KFold
```

سپس مجموعه داده Prediction of students performance.csv لود شده و ۵ ردیف اول آن نمایش داده شده است. پس از آن با دستور زیر، ستون Target را کد کردیم چرا که از جنس عددی نبود.

```
for i in dataset.select_dtypes(include='object').columns:
    dataset[i] = dataset[i].astype('category').cat.codes
```

x و y داده‌ها را از هم جدا کرده، داده‌ها را با روش mean/std نرمالایز کردیم چرا که فیچرها محدوده‌های متفاوتی داشتند. سپس ۳۰ درصد داده‌ها را برای تست، ۲۰ درصد باقی را برای ولید و بقیه را برای آموزش جدا کردیم.

مدل SVM خطی را با دستور LinearSVC(random_state = 0, tol = 0.1, max_iter = 10000) ایجاد کردیم. پارامتر max_iter را تا جایی بالا بردیم که warning عدم همگرایی حذف شود. مدل را به داده‌های train فیت کرده و با اهمال آن بر داده‌های آموزش و تست accuracy را محاسبه و گزارش کردیم. نهایتا با دستور زیر، تعداد ساپورت وکتورها را محاسبه کردیم.

```
# Find Support Vectors
decision_function = svm.decision_function(x_train)
support_vector_indices = np.unique(np.where(np.abs(decision_function) <= 1+1e-5)[0])
support_vectors = x_train.iloc[support_vector_indices]
```

نتیجه به شرح زیر است:

```
Train acuuracy: 77.423
Test acuuracy: 76.130

Number of support vectors = 2287
```

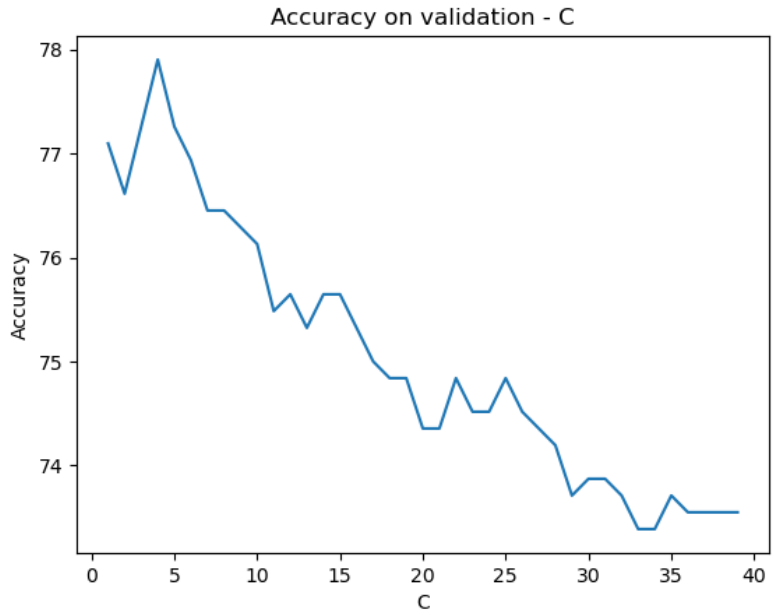
ب) Soft – SVM

این بار از الگوریتم SVM غیرخطی در حالت soft SVM استفاده کردیم. برای یافتن مقدار بهینه ابرپارامتر C یک حلقه for نوشته و به ازای مقادیر مختلف آن، accuracy را برروی داده‌های ولید بررسی کرده و حداکثر مقدار آن را به همراه C بهینه گزارش کردیم.

```
max_acc = 0
opt_C = 0
acc_list = []
# Find the optimal C according to validation accuracy
for i in range(1,40):
    svm = SVC(C = i, tol = 1e-4, max_iter = -1)
    svm.fit(x_train, y_train)
    y_valid_pred = svm.predict(x_valid)
    valid_acc = accuracy_score(y_valid, y_valid_pred)
    acc_list.append(valid_acc*100)
    if (valid_acc > max_acc):
        max_acc = valid_acc
        opt_C = i
print(f"Highest accuracy on validation set is {max_acc*100:.3f} for C = {opt_C}")
```

```
Highest accuracy on validation set is 77.903 for C = 4
```

سپس نمودار دقت برروی مجموعه ولید بر حسب C را رسم کرده و با استفاده از C بهینه، مدلی را بر داده‌های train فیت کرده و دقت برروی مجموعه تست و آموزش و تعداد ساپورت وکتورها را گزارش کردیم.




```
Training accuracy:  88.611
Test accuracy:  76.280
Number of support vectors = 1434
```

شاهدیم که دقت افزایش پیدا کرده و تعداد ساپورت وکتورها به میزان قابل توجهی کاهش یافت.

پ) کرنل‌های چندجمله‌ای و RBF

در این مرحله، همان فرآیند سوال قبل تکرار شد با این تفاوت که دقت برروی مجموعه ولید به ازای دو کرنل چندجمله‌ای و rbf بررسی شد. سپس عملکرد مدل با C بهینه پیدا شده و این دو کرنل بر روی مجموعه‌های تست و آموزش گزارش شد.

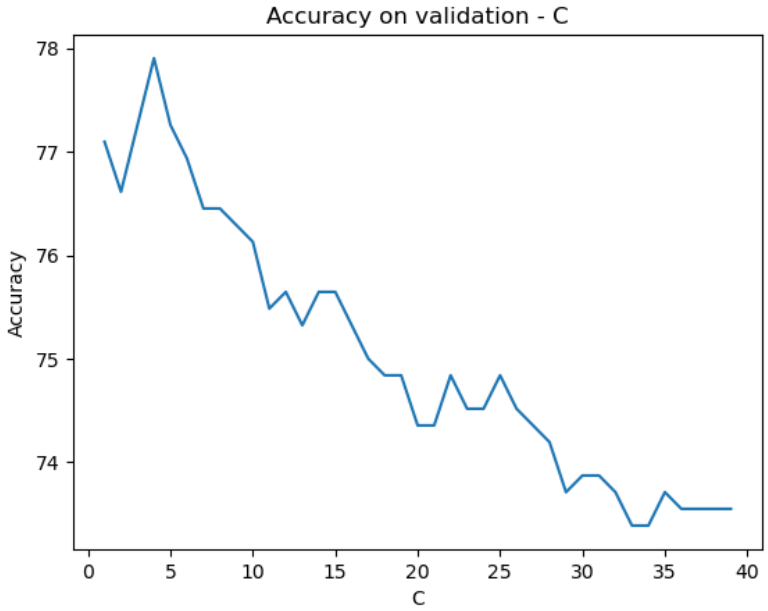
```
Highest accuracy on validation set for rbf kernel is 77.903 with C = 4
Highest accuracy on validation set for poly kernel is 77.903 with C = 4
Train accuracy is: 88.611 for rbf kernel
Test accuracy is 74.473 for rbf kernel
Number of support vectors for rbf kernel = 1434
Train accuracy is: 86.470 for poly kernel
Test accuracy is 71.160 for poly kernel
Number of support vectors for poly kernel = 1473
```

شاهدیم که دقت کرنل RBF از چندجمله‌ای مقداری بهتر بود و تعداد ساپورت وکتورهایش هم کمتر بود.

ت) ترکیب کرنل و soft SVM

با بهترین کرنل بخش قبل که RBF بود، C بهینه را یافته و بیشترین مقدار دقت بر روی داده‌های ولید گزارش شد و بار دیگر نمودار Valid – C رسم شد.

```
Highest accuracy on validation set is  77.903 for C = 4
```



مطابق انتظار، نتیجه همانند بخش ب شد. چرا که کرنل پیش فرض SVM غیرخطی همان RBF است.

```
Train accuracy:  88.611
Test accuracy:  74.473
Number of support vectors = 1434
```

ث) 3-fold cross validation

در این بخش هم از روش k-fold استفاده کردیم. که نتیجه تقریبا مشابهی با RBF به ما داد.

```
Train accuracy:  88.611
Test accuracy:  74.473
Number of support vectors = 1434
```

سوال دوم

الف) SVR خطی

ابتدا کتابخانه‌های لازم ایمپورت شده‌اند.

```
import numpy as np
import pandas as pd
from sklearn.svm import LinearSVR, SVR
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, KFold
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

سپس مجموعه داده Fuel Consumption.csv لود شده و ۵ ردیف اول آن نمایش داده شده است. x و y داده‌ها را از هم جدا کردیم. (توجه داریم که تنها از فیچرهای عددی استفاده کردیم.) داده‌ها را با دستور StandardScaler() نرمالایز کردیم چرا که فیچرها محدوده‌های متفاوتی داشتند. سپس ۲۰ درصد داده‌ها را برای تست و بقیه را برای آموزش جدا کردیم.

برای validation از روش Kfold استفاده کردیم که برای اینکه ۲۰ درصد داده‌های کل ولید باشد، لازم بود ۲۵ درصد داده‌های مانده پس از جدا شدن مجموعه تست، برای ولید انتخاب شوند. پس k را برابر 4 گذاشتیم.

```
# Create the KFold object
k = 4 # %25 of train set for validation set
kf = KFold(n_splits=k, shuffle=True, random_state=42)
```

مدل LinearSVR را ساخته و به داده‌های آموزش فیت کردیم.

```
model = LinearSVR(epsilon = 0.1, C = 1.0, loss = 'epsilon_insensitive',
random_state = 42, max_iter = 10000)
```

سپس kfold را اجرا کرده و سه معیار mean_squared_error، mean_absolute_error، r2_score را در هر مرحله بر روی داده‌های ولید محاسبه کرده و میانگین آن‌ها را گزارش کردیم.
سپس این معیارها را برای داده‌های آموزش و تست هم محاسبه و گزارش کردیم.

```
Train set:
MSE: 0.0018
MAE: 0.0331
R2: 0.9982

Validation set:
Average MSE: 0.0016
Average MAE: 0.0305
Average R2: 0.9984

Test set:
MSE: 0.0018
MAE: 0.0331
R2: 0.9983
```

```python
# Loop over the train and valid indices
for train_index, valid_index in kf.split(X_train):
    # Split the data into train and valid sets
    x_train, x_valid = X_train[train_index], X_train[valid_index]
    y_train, y_valid = Y_train[train_index], Y_train[valid_index]

    # Fit the model on the train set
    model.fit(x_train, y_train)

    # Predict the target values for the valid set
    y_valid_pred = model.predict(x_valid)

    # Compute the accuracy metrics for the valid set
    mse_valid = mean_squared_error(y_valid, y_valid_pred)
    mae_valid = mean_absolute_error(y_valid, y_valid_pred)
    r2_valid = r2_score(y_valid, y_valid_pred)

    # Append the metrics to the lists
    mse_list.append(mse_valid)
    mae_list.append(mae_valid)
    r2_list.append(r2_valid)

# Calculate the average of the metrics
mse_mean = np.mean(mse_valid)
mae_mean = np.mean(mae_valid)
r2_mean = np.mean(r2_valid)
```

ب) Nonlinear SVR

در این بخش دقیقا همان مراحل بخش قبل را یک بار با کرنل RBF و یک بار با کرنل چند جمله‌ای تکرار کردیم. دستور هم به شکل زیر تغییر یافت.

```
model = SVR(kernel = 'poly', epsilon = 0.1, C = 1.0, max_iter = 100000)

Train set:
MSE: 0.1899
MAE: 0.2911
R2: 0.8105

Validation set:
Average MSE: 0.1554
Average MAE: 0.2814
Average R2: 0.8417

Test set:
MSE: 0.2735
MAE: 0.3190
R2: 0.7370
```

```
model = SVR(kernel = 'rbf', epsilon = 0.1, C = 1.0, max_iter = 10000)
Train set:
MSE: 0.0048
MAE: 0.0526
R2: 0.9952

Validation set:
Average MSE: 0.0068
Average MAE: 0.0519
Average R2: 0.9930

Test set:
MSE: 0.0109
MAE: 0.0556
R2: 0.9895
```

شاهدیم که نتایج SVR خطی از همه بهتر بود و نتایج با کرنل RBF کمی از آن پایین‌تر و کرنل چند جمله‌ای بسیار بدتر بود.