Exercise 1

Sergei Rogov

Assignment kNN & SVM

X	Class
0.5	B
3.0	B
4.5	A
4.6	A
4.9	A
5.2	B
5.3	B
5.5	A
7.0	B
9.5	B

X	Class
5.0	???

Distances

$(5.0-0.5)^2 = 4.5$
$(5.0-3.0)^2 = 2.0$ ✓
$(5.0-4.5)^2 = 0.5$ ✓
$(5.0-4.6)^2 = 0.4$ ✓
$(5.0-4.9)^2 = 0.1$ ✓
$(5.0-5.2)^2 = 0.2$ ✓
$(5.0-5.3)^2 = 0.3$ ✓
$(5.0-5.5)^2 = 0.5$ ✓
$(5.0-7.0)^2 = 2.0$ ✓
$(5.0-9.5)^2 = 4.5$

Sorted distances:

	X	Class	Distance
1)	4.9	A	0.1
2)	5.2	B	0.2
3)	5.3	B	0.3
4)	4.6	A	0.4
5)	4.5	A	0.5
6)	5.5	A	0.5
7)	3.0	B	2.0
8)	7.0	B	2.0
9)	0.5	B	4.5
	9.5	B	4.5

a) 1-nearest neighbour (to 5.0):

4.9 A 0.1

=> class = A

A 1x

3-nearest:

4.9 A 0.1
5.2 B 0.2
5.3 B 0.3

A 1x
B 2x

=> class = B

9-nearest:

4.9 A 0.1
5.2 B 0.2
5.3 B 0.3
4.6 A 0.4
4.5 A 0.5
5.5 A 0.5
3.0 B 2.0
7.0 B 2.0
0.5 B 4.5

A 4x
B 5x

=> class = B

b) Weighted approach:

1-nearest - remains the same

=> Class = A

3-nearest:

A: $\frac{1}{0.1} = 10$

B: $\frac{1}{0.2} + \frac{1}{0.3} = 8.333$

A > B

=> Class = A

9-nearest:

A: $\frac{1}{0.1} + \frac{1}{0.4} + \frac{1}{0.5} + \frac{1}{0.5} = 16.5$

B: $\frac{1}{0.2} + \frac{1}{0.3} + \frac{1}{2.0} + \frac{1}{2.0} + \frac{1}{4.5} = 9.555$

A > B

=> class = A

Assuming Inverse proportionality

Answer:

a) ① A
③ B
⑨ B

b) Weighted distance:
① A
③ A
⑨ A

Exercise 2

Python Code

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.datasets import load_wine
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_validate
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import make_scorer


# A function to perform 10-fold validation
# Prints out the mean values of 4 metrics
def train_and_test_model(dataset, model):
    x, y = dataset.data, dataset.target
    scoring = {'accuracy': make_scorer(accuracy_score),
               'precision': make_scorer(precision_score, average='macro',
zero_division=1),
               'recall': make_scorer(recall_score, average='macro'),
               'f1_score': make_scorer(f1_score, average='macro')}
    scores = cross_validate(model, x, y, cv=10, scoring=scoring)
    # 10-Fold Validation metrics
    print(f"Accuracy = {scores['test_accuracy'].mean():.3f}")
    print(f"Precision = {scores['test_precision'].mean():.3f}")
    print(f"Recall = {scores['test_recall'].mean():.3f}")
    print(f"F_score = {scores['test_f1_score'].mean():.3f}\n")


# loading iris and wine dataset
iris = load_iris()
wine = load_wine()

# creating knn and svm models
knn_model = KNeighborsClassifier(n_neighbors=13, algorithm='kd_tree')
svm_model = SVC(kernel='rbf', degree=3, gamma='scale')

print("iris, kNN")
train_and_test_model(iris, knn_model)  # Accuracy = 0.980
print("iris, SVM")
train_and_test_model(iris, svm_model)  # Accuracy = 0.973
print("wine, kNN")
train_and_test_model(wine, knn_model)  # Accuracy = 0.692
print("wine, SVM")
train_and_test_model(wine, svm_model)  # Accuracy = 0.681

# kNN performs slightly better than SVM on these datasets
```

Report on accuracy, Precision, Recall and F score

iris, kNN
Accuracy = 0.980
Precision = 0.983
Recall = 0.980
F_score = 0.980

iris, SVM
Accuracy = 0.973
Precision = 0.978
Recall = 0.973
F_score = 0.973

wine, kNN
Accuracy = 0.692
Precision = 0.712
Recall = 0.686
F_score = 0.680

wine, SVM
Accuracy = 0.681
Precision = 0.702
Recall = 0.656
F_score = 0.647

a) After doing 10-fold validation and measuring the mean of accuracy metrics I found out that kNN performed slightly better than SVM on these datasets.

b) I would use kNN algorithm here because it has higher accuracy metrics. However, with different k (for example k=7) SVM algorithm was better. Parameter k always requires investigation.

I experimented with algorithms parameters as well. The best performance for kNN algorithm happened to be with parameter k = 13;
When changing SVM parameter gamma from ‘scale’ to ‘auto’, SVM accuracy on wine dataset drastically fell from 0.681 to 0.439, so I kept ‘scale’.