HW7 report

Note that: when I run the code there are some warning:

1. ConvergenceWarning: Liblinear failed to converge, increase the number of iterations
2. DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

I changed the max_iter to 10000 and alpha to 1e-5, the warning still exists. I don't how to remove these warning, and because the warning didn't ruin the result, so I ignore these warning.

a

```
import numpy as np
import pandas as pd
import warnings
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from imblearn.over_sampling import SMOTE
from sklearn.multiclass import OneVsRestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split


# question a
file_data = pd.read_csv('Frogs_MFCCs.csv', header=None)
data_set = file_data.values[1:]
features = file_data.values[0]

training_data, test_data = train_test_split(data_set, test_size=0.3)
training_set = pd.DataFrame(data=training_data, columns=features, index=None)
test_set = pd.DataFrame(data=test_data, columns=features, index=None)
```

b(i)

```
# question b(i) to calculate exact match loss
def exact_match_loss(family_prediction, genus_prediction, species_prediction,
family_label, genus_label, species_label):
   if not (len(family_prediction) == len(genus_prediction) ==
len(species_prediction)
          == len(family_label) == len(genus_label) == len(species_label)):
       print('error exist')
```

```
        return 0
    right_count = 0
    data_numbers = len(family_prediction)
    for i in range(data_numbers):
        if family_prediction[i] == family_label[i] \
                and genus_prediction[i] == genus_label[i] \
                and species_prediction[i] == species_label[i]:
            right_count += 1
    exact_loss = 1 - right_count / data_numbers
    return exact_loss
```

```
# question b(i) to calculate humming score loss
def hamming_score_loss(family_prediction, genus_prediction, species_prediction,
family_label, genus_label,
                       species_label):
    if not (len(family_prediction) == len(genus_prediction) ==
len(species_prediction)
            == len(family_label) == len(genus_label) == len(species_label)):
        print('error exist')
        return 0
    error_count = 0
    data_numbers = len(family_prediction)
    for i in range(data_numbers):
        if family_prediction[i] != family_label[i]:
            error_count += 1
        if genus_prediction[i] != genus_label[i]:
            error_count += 1
        if species_prediction[i] != species_label[i]:
            error_count += 1
    hamming_loss = error_count / (3 * data_numbers)
    return hamming_loss
```

b(ii)

```
# question b(ii)
print('In question b(ii), using Gaussian kernels and ova classifier to train a
SVM, and the best parameters:')
parameters_c = np.array([10 ** value for value in range(-3, 7)])
kernel_variance = (np.array([value for value in range(1, 21)]) / 10)
parameters_gamma = 1 / (2 * (kernel_variance ** 2))
kernel_parameters = {'estimator__C': parameters_c, 'estimator__gamma':
parameters_gamma}

training_set_x = training_set.drop(['Family', 'Genus', 'Species', 'RecordID'],
axis=1)
training_set_y_family = pd.DataFrame(training_set['Family'])
training_set_y_genus = pd.DataFrame(training_set['Genus'])
training_set_y_species = pd.DataFrame(training_set['Species'])

test_set_x = test_set.drop(['Family', 'Genus', 'Species', 'RecordID'], axis=1)
test_set_y_family = pd.DataFrame(test_set['Family'])
test_set_y_genus = pd.DataFrame(test_set['Genus'])
test_set_y_species = pd.DataFrame(test_set['Species'])

ova_classifier = OneVsRestClassifier(SVC())
cross_validation_family = GridSearchCV(ova_classifier, kernel_parameters,
cv=10, n_jobs=-1)
cross_validation_genus = GridSearchCV(ova_classifier, kernel_parameters, cv=10,
n_jobs=-1)
cross_validation_species = GridSearchCV(ova_classifier, kernel_parameters,
cv=10, n_jobs=-1)

family_predictor = cross_validation_family.fit(training_set_x,
training_set_y_family)
genus_predictor = cross_validation_genus.fit(training_set_x,
training_set_y_genus)
species_predictor = cross_validation_species.fit(training_set_x,
training_set_y_species)

best_parameters_of_family = family_predictor.best_params_
best_parameters_of_genus = genus_predictor.best_params_
best_parameters_of_species = species_predictor.best_params_

best_c_family = best_parameters_of_family['estimator__C']
best_variance_family = (1 / best_parameters_of_family['estimator__gamma']) **
```

```
0.5
best_c_genus = best_parameters_of_genus['estimator__C']
best_variance_genus = (1 / best_parameters_of_genus['estimator__gamma']) ** 0.5
best_c_species = best_parameters_of_species['estimator__C']
best_variance_species = (1 / best_parameters_of_species['estimator__gamma']) **
0.5

print('the best weight of the SVM penalty in family model is', best_c_family)
print('the best width of the Gaussian Kernel in family model is',
best_variance_family)
print('the best weight of the SVM penalty in genus model is', best_c_genus)
print('the best width of the Gaussian Kernel in genus model is',
best_variance_genus)
print('the best weight of the SVM penalty in species model is', best_c_species)
print('the best width of the Gaussian Kernel in species model is',
best_variance_species)

family_result = family_predictor.predict(test_set_x)
genus_result = genus_predictor.predict(test_set_x)
species_result = species_predictor.predict(test_set_x)

exact_match_loss_score = exact_match_loss(family_result, genus_result,
species_result, test_set_y_family.values,
                                    test_set_y_genus.values,
test_set_y_species.values)
hamming_loss_score = hamming_score_loss(family_result, genus_result,
species_result, test_set_y_family.values,
                                   test_set_y_genus.values,
test_set_y_species.values)

print('the exact match loss is', exact_match_loss_score)
print('the hamming loss is', hamming_loss_score, '\n\n')
```

Result:

In question b(ii), using Gaussian kernels and ova classifier to train a SVM, and the best parameters:

the best weight of the SVM penalty in family model is 10.0

the best width of the Gaussian Kernel in family model is 0.7071067811865476

the best weight of the SVM penalty in genus model is 10.0

the best width of the Gaussian Kernel in genus model is 0.7071067811865476

the best weight of the SVM penalty in species model is 10.0
the best width of the Gaussian Kernel in species model is 0.7071067811865476
the exact match loss is 0.008800370541917513
the hamming loss is 0.006175698625907056

b(iii)

```
# question b(iii)
print('In question b(iii), using Gaussian kernels and ova classifier to train a
L1-penalized SVM (linear):')
kernel_parameters_l1 = {'C': parameters_c}
linearSVC_classifier = LinearSVC(penalty='l1', dual=False)
cross_validation_family_l1 = GridSearchCV(linearSVC_classifier,
kernel_parameters_l1, cv=10, n_jobs=-1)
cross_validation_genus_l1 = GridSearchCV(linearSVC_classifier,
kernel_parameters_l1, cv=10, n_jobs=-1)
cross_validation_species_l1 = GridSearchCV(linearSVC_classifier,
kernel_parameters_l1, cv=10, n_jobs=-1)

family_predictor_l1 = cross_validation_family_l1.fit(training_set_x,
training_set_y_family)
genus_predictor_l1 = cross_validation_genus_l1.fit(training_set_x,
training_set_y_genus)
species_predictor_l1 = cross_validation_species_l1.fit(training_set_x,
training_set_y_species)

best_c_family_l1 = family_predictor_l1.best_params_['C']
best_c_genus_l1 = genus_predictor_l1.best_params_['C']
best_c_species_l1 = species_predictor_l1.best_params_['C']

print('the best weight of the SVM penalty in family model is',
best_c_family_l1)
print('the best weight of the SVM penalty in genus model is', best_c_genus_l1)
print('the best weight of the SVM penalty in species model is',
best_c_species_l1)

family_result_l1 = family_predictor_l1.predict(test_set_x)
genus_result_l1 = genus_predictor_l1.predict(test_set_x)
species_result_l1 = species_predictor_l1.predict(test_set_x)
```

```
exact_match_loss_score_l1 = exact_match_loss(family_result_l1, genus_result_l1,
species_result_l1, test_set_y_family.values,test_set_y_genus.values,
test_set_y_species.values)
hamming_loss_score_l1 = hamming_score_loss(family_result_l1, genus_result_l1,
species_result_l1, test_set_y_family.values, test_set_y_genus.values,
test_set_y_species.values)

print('the exact match loss is', exact_match_loss_score_l1)
print('the hamming loss is', hamming_loss_score_l1)
```

Result:

In question b(iii), using Gaussian kernels and ova classifier to train a L1-penalized SVM (linear):
the best weight of the SVM penalty in family model is 100.0
the best weight of the SVM penalty in genus model is 10.0
the best weight of the SVM penalty in species model is 100.0
the exact match loss is 0.0889300602130616
the hamming loss is 0.05511811023622047

b(iv)

```
# question b(iv)
print('In question b(iv), after using SMOTE, the result is')
smote = SMOTE(ratio='minority')
training_set_x_family_smote, training_set_y_family_smote =
smote.fit_sample(training_set_x, training_set_y_family)
training_set_x_genus_smote, training_set_y_genus_smote =
smote.fit_sample(training_set_x, training_set_y_genus)
training_set_x_species_smote, training_set_y_species_smote =
smote.fit_sample(training_set_x,

training_set_y_species)

kernel_parameters_l1_smote = {'C': parameters_c}
linearSVC_classifier_smote = LinearSVC(penalty='l1', dual=False)
cross_validation_family_l1_smote = GridSearchCV(linearSVC_classifier_smote,
kernel_parameters_l1_smote, cv=10,
                                        n_jobs=-1)
```

```
cross_validation_genus_l1_smote = GridSearchCV(linearSVC_classifier_smote,
kernel_parameters_l1_smote, cv=10,
                                       n_jobs=-1)
cross_validation_species_l1_smote = GridSearchCV(linearSVC_classifier_smote,
kernel_parameters_l1_smote, cv=10,
                                        n_jobs=-1)

family_predictor_l1_smote =
cross_validation_family_l1_smote.fit(training_set_x_family_smote,

training_set_y_family_smote)
genus_predictor_l1_smote =
cross_validation_genus_l1_smote.fit(training_set_x_genus_smote,
                                                    training_set_y_genus_smote)
species_predictor_l1_smote =
cross_validation_species_l1_smote.fit(training_set_x_species_smote,

training_set_y_species_smote)

best_c_family_l1_smote = family_predictor_l1_smote.best_params_['C']
best_c_genus_l1_smote = genus_predictor_l1_smote.best_params_['C']
best_c_species_l1_smote = species_predictor_l1_smote.best_params_['C']

print('the best weight of the SVM penalty in family model is',
best_c_family_l1_smote)
print('the best weight of the SVM penalty in genus model is',
best_c_genus_l1_smote)
print('the best weight of the SVM penalty in species model is',
best_c_species_l1_smote)

family_result_l1_smote = family_predictor_l1_smote.predict(test_set_x)
genus_result_l1_smote = genus_predictor_l1_smote.predict(test_set_x)
species_result_l1_smote = species_predictor_l1_smote.predict(test_set_x)

exact_match_loss_score_l1_smote = exact_match_loss(family_result_l1_smote,
genus_result_l1_smote,
                                        species_result_l1_smote,
                                        test_set_y_family.values,
                                        test_set_y_genus.values,
test_set_y_species.values)
hamming_loss_score_l1_smote = hamming_score_loss(family_result_l1_smote,
```

```
genus_result_l1_smote,
                                        species_result_l1_smote,
                                        test_set_y_family.values,
                                        test_set_y_genus.values,
test_set_y_species.values)

print('the exact match loss is', exact_match_loss_score_l1_smote)
print('the hamming loss is', hamming_loss_score_l1_smote)
```

Result:

In question b(iv), after using SMOTE, the result is

the best weight of the SVM penalty in family model is 10.0

the best weight of the SVM penalty in genus model is 10.0

the best weight of the SVM penalty in species model is 10.0

the exact match loss is 0.09634089856415007

the hamming loss is 0.06206577119036591