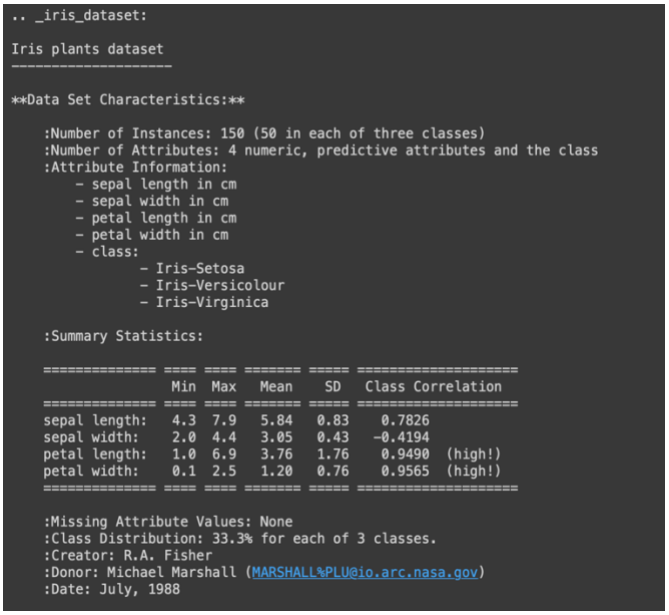
[CMPUT 466/566, Fall 2021] Machine learning

Course Project Report

Dataset Introduction

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken from Fisher's paper. Note that it's the same as in R, but not as in the UCI Machine Learning Repository, which has two wrong data points.



```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988
```

This is perhaps the best-known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Project Introduction

In this project, we use five different machine learning algorithms to classify the Iris dataset that comes with the sklearn library. The five machine learning algorithms are Baseline, SVM, Logistic Regression, Naive Bayes, and KNN. We use a systematic hyperparameter adjustment method to implement a training-verification-test infrastructure. First, the cross-validation method (by cross_val_score method of sklearn) finds the best hyperparameters that achieve the highest cross-validation accuracy for each model, and then refits the model on the entire training set according to the best hyperparameters and evaluates the performance on the test set. Finally, we get the conclusion through the Cross Validation Accuracy, Negative Mean Squared Error, and Loss figures of the models corresponding to the four algorithms. When drawing, perform log10 on hyperparameter.

Baseline

Construct a baseline classifier based on a random prediction algorithm. When evaluating other machine learning algorithms, the accuracy of the baseline classifier provides a reference result for comparison. Random prediction algorithms predict random results observed in the training data. The baseline test accuracy is 0.63333333.

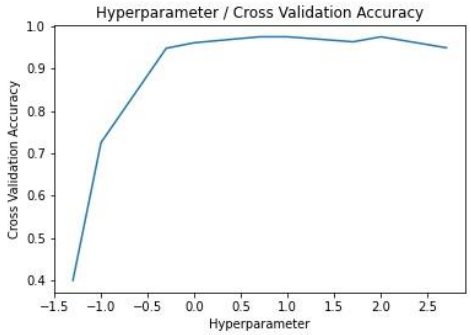
SVM

SVM is a machine learning classification model that maps the feature vector of an instance to some points in the space. The purpose of SVM is to draw a line to distinguish it "best" These two types of points, even if there are new points in the future, this line can also make a good classification. SVM is suitable

for small and medium-sized data samples, non-linear, high-dimensional classification problems. SVM was first proposed by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963. The current version (soft margin) was proposed by Corinna Cortes and Vapnik in 1993 and published in 1995. Before the advent of deep learning (2012), SVM was considered the most successful and best performing algorithm in machine learning in the past ten years. In sklearn.svm.SVC(), C is the regularization parameter and also hyperparameter. The intensity of regularization is inversely proportional to C. The penalty is the squared L2 penalty. Adjusting the value of C to construct different SVM models to provide data for cross-validation step and the drawing of Cross Validation Accuracy figure, Negative Mean Squared Error figure and Loss figure. Then using the best hyperparameter to refit the model and make predictions on the data of test set.

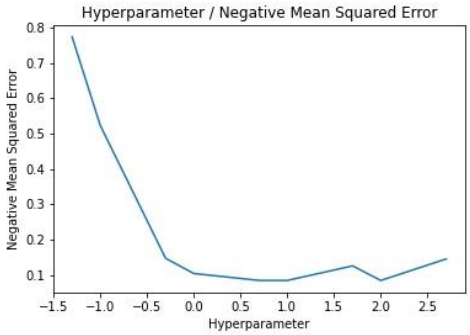
Run the program, we get the best hyperparameter is when C = 5, and SVM test accuracy is 0.96666667.

Figure of Hyperparameter / Cross Validation Accuracy by SVM classifier:



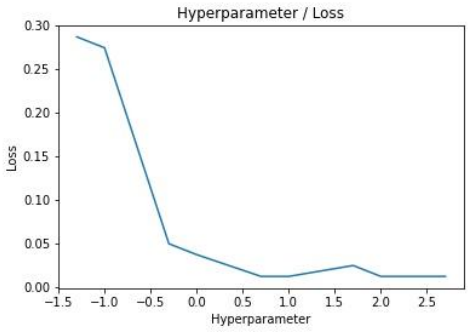
As hyperparameter C becomes larger, the cross validation accuracy increases, and accuracy converges near 0.96.

Figure of Hyperparameter / Negative Mean Squared Error by SVM classifier:



As hyperparameter C becomes larger, the negative mean squared error decreases, and it converges near 0.1.

Figure of Hyperparameter / Loss by SVM classifier:



As hyperparameter C becomes larger, the loss decreases, and it converges near around 0.01.

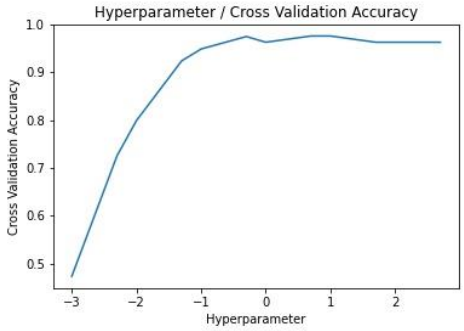
Logistic Regression

Logistic regression is a commonly used machine learning method in the industry, used to estimate the possibility of a certain thing, and used to classify. In the case of classification, the Logistic regression classifier after learning is a set of weights $w_1x_1,\ w_2x_2, ...,\ w_mx_m$. When the test data in the test sample

set is input, the weight of this group is linearly added to the test data to obtain a z value: $z = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_mx_m + b = z = w^Tx + b$. Then it is calculated in the form of the sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$. In sklearn.linear_model.LogisticRegression(), C is the inverse of regularization strength. Like in SVM, smaller values specify stronger regularization. Suppose C is hyperparameter of logistic regression, adjusting the value of C to construct different logistic regression models to provide data for cross-validation step and the drawing of Cross Validation Accuracy figure, Negative Mean Squared Error figure and Loss figure. Then using the best hyperparameter to refit the model and make predictions on the data of test set.

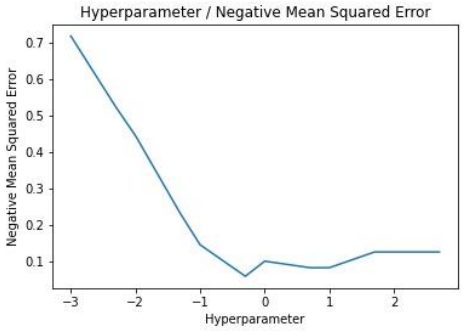
Run the program, we get the best hyperparameter is when C = 5, and logistic regression test accuracy is 0.96666667.

Figure of Hyperparameter / Cross Validation Accuracy by logistic regression *classifier:*



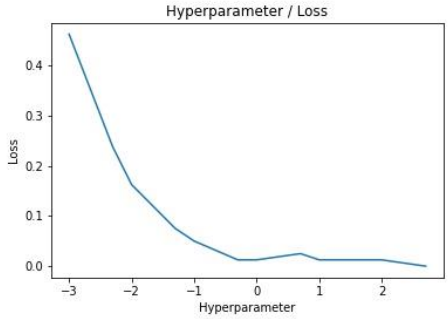
As hyperparameter C becomes larger, the cross validation accuracy increases, and it converges near 0.96.

Figure of Hyperparameter / Negative Mean Squared Error by logistic regression *classifier:*



As hyperparameter C becomes larger, the negative mean squared error decreases in total, the lowest negative MSE is below 0.1.

Figure of Hyperparameter / Loss by logistic regression *classifier:*



As hyperparameter C becomes larger, the loss decreases, and it converges near almost 0.0.

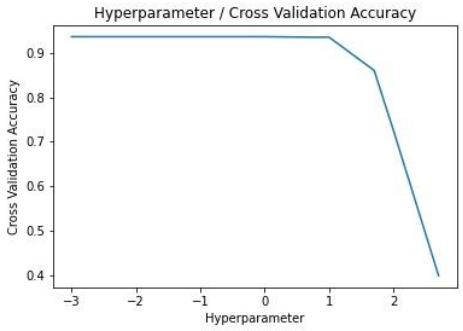
Naive Bayes

Naive Bayes is a kind of Bayesian classification algorithm, which is an improvement of Bayesian. This algorithm can be applied to large databases, and the method is simple, the classification accuracy rate is high, and the speed is fast. The significant difference between Naive Bayes and Bayes is that Naive Bayes

makes the assumption of independence, assuming that each feature is independent and uncorrelated. The Bayesian formula is $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$. In sklearn.naive_bayes.CategoricalNB(), alpha is the additive (Laplace) smoothing parameter which is consider to be hyperparameter. Adjusting the value of alpha to construct different NB models to provide data for cross-validation step and the drawing of Cross Validation Accuracy figure, Negative Mean Squared Error figure and Loss figure. Then using the best hyperparameter to refit the model and make predictions on the data of test set.

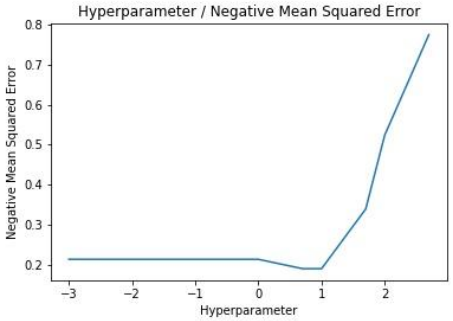
Run the program, we get the best hyperparameter is when C = 0.001, and naive bayes test accuracy is 0.86666667.

Figure of Hyperparameter / Cross Validation Accuracy by Naive Bayes classifier:



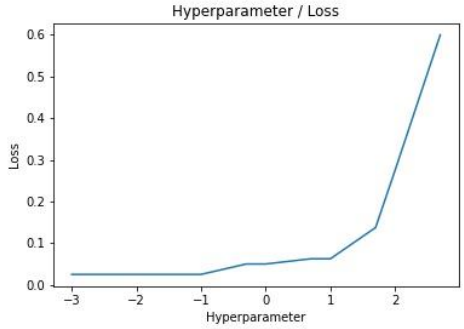
As hyperparameter C becomes larger, the cross validation accuracy decreases, from around 0.94 to below 0.4.

Figure of Hyperparameter / Negative Mean Squared Error by Naive Bayes classifier:



As hyperparameter C becomes larger, the negative mean squared error increases, from around 0.21 to around 0.8.

Figure of Hyperparameter / Loss by Naive Bayes classifier:



As hyperparameter C becomes larger, the loss increases, from around 0.02 to around 0.6.

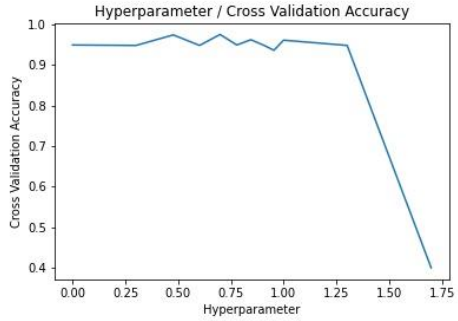
KNN

The KNN nearest neighbor classification algorithm is also a commonly used machine learning algorithm. In order to judge the category of the unknown sample, take all the samples of the known category as a reference, calculate the distance between the unknown sample and all the known samples, and select the K known samples with the closest distance to the unknown sample, according to the voting rule that the minority obeys the majority, the unknown sample and the K nearest samples are classified into one

category. In sklearn.neighbors.KneighborsClassifier(), n_neighbors is the number of neighbors to use by default for kneighbors queries which is hyperparameter. Adjusting the value of n_neighbors to construct different KNN models to provide data for cross-validation step and the drawing of Cross Validation Accuracy figure, Negative Mean Squared Error figure and Loss figure. Then using the best hyperparameter to refit the model and make predictions on the data of test set.

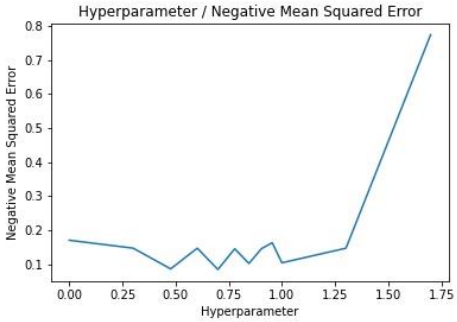
Run the program, we get the best hyperparameter is when C = 5, and naive bayes test accuracy is 0.96666667.

Figure of Hyperparameter / Cross Validation Accuracy by KNN classifier:



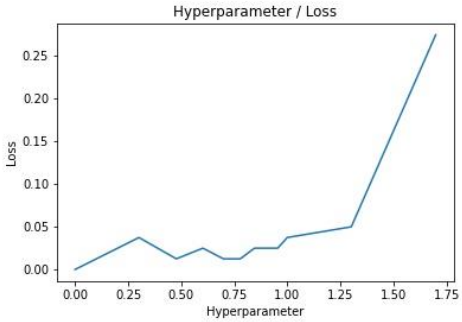
As hyperparameter C becomes larger, the cross validation accuracy decreases, from around 0.96 to below 0.4.

Figure of Hyperparameter / Negative Mean Squared Error by KNN classifier:



As hyperparameter C becomes larger, the negative mean squared error increases in total, from around 0.15 to around 0.8.

Figure of Hyperparameter / Loss by KNN classifier:



As hyperparameter C becomes larger, the loss increases in total, from around 0.03 to over 0.25.

Result

The test accuracys of the latter four models are all higher than that of the baseline model. It can be seen from the figures of the four models that the cross validation accuracy of SVM model and logistic regression model increase as hyperparameter increases; the negative mean squared error and the loss of SVM model and logistic regression model decrease as hyperparameter increases. Naive bayes model and KNN model are opposite. Among these models, SVM, logistic regression, and KNN have the same test accuracy which is 0.96666667, and the hyperparameter is 5. The test accuracy of naive bayes which is 0.86666667 is slightly smaller than the previous models' accuracy, the hyperparameter is 0.001. The test

accuracy of naive bayes is low because this is because NB uses a prior and data to determine the posterior probability to determine the classification, so there is a certain error rate in the classification decision; NB assumes that the attributes are between mutual independence, this assumption is often not true in practical applications. When the number of attributes is large or the correlation between attributes is large, the classification effect is not good; and NB is very sensitive to the expression of input data.