CS5014
170025298 03/03/2021

Section 1

Q1

Data are loaded with Pandas, whereas we have to do the following for cleaning those data:

- Remove missing data: remove rows of data that have '?' inside.
- Remove duplicates: Duplicate data is redundant for training
- Ensure the data are in the right format: prevent training our models with wrong data
- Fix encoding and file errors: Identify string objects (exclude numerical elements) and convert those objects to numerical (For example, if there are three elements in a column [a, b, c], we would use [0, 1, 2] to represent those objects)

Q2

For making the most of data for training, we take 20% data as its testing set. Testing data and training data will be split randomly for prevent the testing data causes the prying error. Splitting on the whole set uses the sklearn package for separating training and learning sets directly. Splitting on the whole set use tools provided by sklearn.

Q3

For data encoding and conversion, we converse all alphabetical elements numerically as what is described in question a, and uses one-hot encoding to encode all characteristic values. We remove all unnecessary rows and columns in order to clean the data and normalize the data with sklearn. The method of scaling depends on the type of our data. Since the output of the model does not represents extreme high or small values as well as we do not do the feature selection, we would use Z-score standardization for scaling.

Q4

If we do have the filed knowledge, we can prevent which data may cause the data leakage. However, the label for data are shown to be meaningless. We would plotting those features otherwise to check the similarities between our results and classification rate. We can also use the validation set for verifying the data leakage.

Section 2

Q1

The worst case for classification accuracy for predicting using the model is that the model has the lowest generalization. The prediction may trains quite well, but will become incorrect once predicting with data using the testing set, which is the issue for 'overfitting'. The best case for classification is that the question is being calculated as what the model describes, and the output is actually correct.

Q2

- penalty: Penalty considers and evaluates error that may happen to incidences. The use of penalty function increases the ability of model handling errors and prevents the overfitting.
- tol: Parameter "tol" represents the tolerance of the model. Tolerance represents as the ability of an object fight against noises. Setting tolerance of the model improves the ability of our model to handle noises. However, the tolerance may decrease the accuracy of the model.
- max_iter: The maximum iteration for calculating the model. The number of iteration based on the use of gradient, whereas no matter which gradient method we would use, the decrement of the gradient follows step by step, and the max iteration number defines when should we force the iteration stop.

Q3

According to the calculation process given by sklearn:

```
class_weight : dict or 'balanced', default=None
        Weights associated with classes in the form
``{class_label: weight}``.
        If not given, all classes are supposed to have weight
one.

        The "balanced" mode uses the values of y to
automatically adjust
        weights inversely proportional to class frequencies in
the input data
        as ``n_samples / (n_classes * np.bincount(y))``.

        Note that these weights will be multiplied with
sample_weight (passed
        through the fit method) if sample_weight is specified.
```

The balanced option of sklearn calculate the weight of classes in a balanced way in total. The option is useful as it solves the issue of sample unbalancing: each class would be distributed in a balanced way and solve the issue of sample balancing.

Q4

According to the internal built-in properties, classes for model includes ['+', '-']

- predict_proba(): returns a vector represents the probability of the output for each class, such as:

```
>>[[0.6612548, 0.1214128]]
Represents:
    [[P('+')    P('-')]]
```

- predict(): returns the predicted output according to its maximum probability.
- decision_function(): According to the document of sklearn, the decision function evaluates the reliability of the model. In my opinion, since we already know that we have get two output classes, the machine would choose only one classifier to represent the result. However, if more classifiers are chosen, such as we would return [0, +, -] as our output, we need two binary classifiers for evaluating the output of the function. The decision function represents the reliability for each binary classification.

Section 3

Q1

The classification accuracy of my model represents as 0.9083969465648855, where the data is being calculated according to the formula shows as following:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{RC + RN}{FC + FN + RC + RN}$$

Whereas symbols represents as following:

- RC: Real confidential: The predicted result and the real result are all represents as '+'
- RN: Real not confidential: The predicted result and the real result are all represents as '-'
- FC: False confidential: The predicted result represents as '+' but the real result represents as '-'

- FN: False not confidential: The predicted result represents as '-' but the real result represents as '+'

Q2

The balanced accuracy score of my model represents as 0.9093983208955223, where the data is being calculated according to the formula shows as following:

$$BAS = \frac{1}{2} \times (\frac{TP}{TP + FN} + \frac{TN}{TN + FP})$$

whereas symbols represents as following:

- TP: True Positive
- TN: True Negative
- FN: False Negative
- FP: False Positive

Q3

The return value for confusion matrix is:

```
metrics.confusion_matrix(Y_res, Y)
>>[[61  3]
 [ 9 58]]
>>[[TP, FN]
  [FP, TN]]
```

Confusion matrix evaluates the accuracy of the classification, whereas the confusion matrix represents the number of true positive, true negative, false positive and false negative.

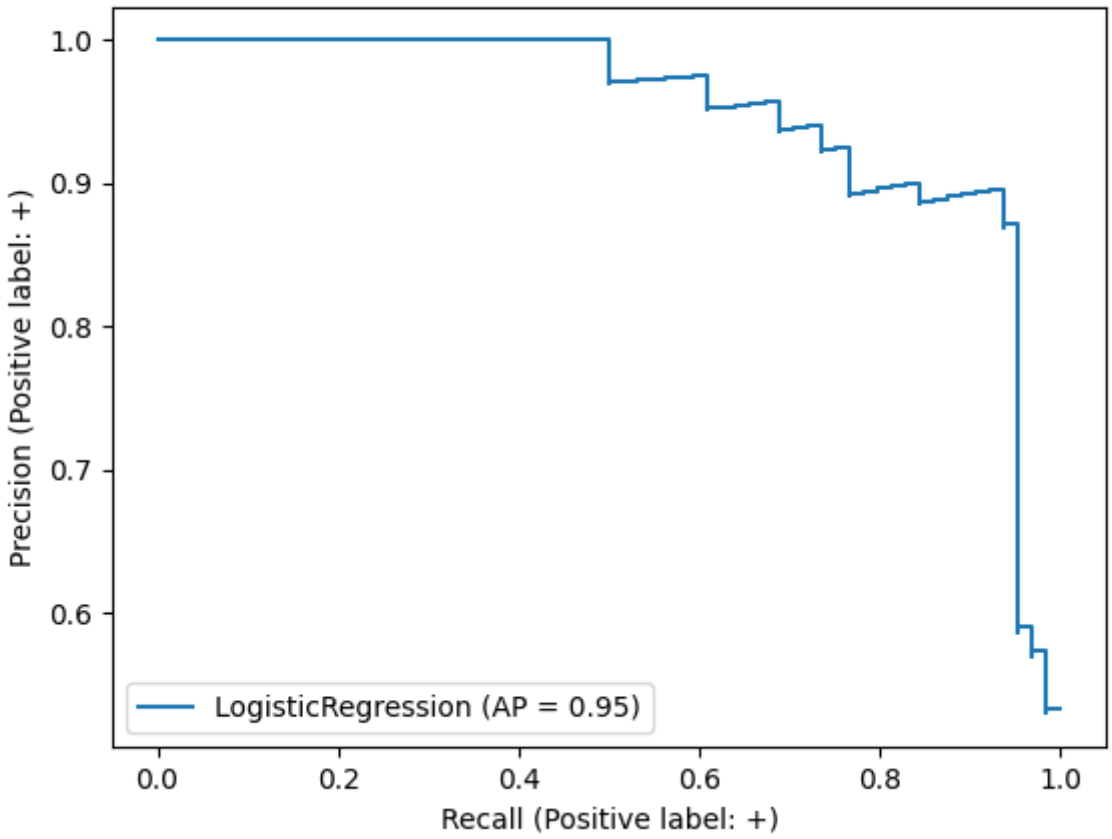
For unbalanced 2a, since for P and N are unbalanced, we may find the large differences between TP and TN / FP and FN. The Recall function represents as:

$$recall = \frac{TP}{TP + FN}$$

if the data is unbalanced, assume we have a fixed accuracy α, recall rate are to be affected if negative data is large enough, and the recall rate tend to get lower. Unbalanced dataset tend to makes the result unreliable.

For balanced 2b, since for P and N are balanced, we do have TP and TN / FP and FN normally distributed. Precision and recall of the data tend to be more stable.

Q4



The precision-recall curve represents as the figure showing above, introduces relation between precision and recall.

The positive average precision score of the model represents as the value of 0.9256587978304862. By the introduction from the function document, the score is the weight mean of precisions achieved at each threshold, which can be represented as

$$AP = \sum_n (R_n - R_{n-1}) P_n$$

Whereas P_n and R_n represents as precision and recall at the nth threshold.

Section 4

Q1

L2 represents as the algorithm likes the following:

$$||X||_2 = \sqrt{\sum_{i=1}^{n} X_i^2}$$

Logistic regression uses logarithmic loss function, whereas the axiom represents as follows:

$$L(Y, P\ \ (Y|X)) = -logP(Y|X)$$

Our gradient of the function may represents as:

$$Gradient = -logP(Y|X) + \sqrt{\sum_{i=1}^{n} X_i^2}$$

Q2

Q3

Q4