ED5340 LAB 11 Explanations | Roll No: ED21S014

Qn 2: (paperwork) Briefly write about the data used, parameters tuned and the interpretation of the results from each of the methods.

A) Implementation of Linear Regression using Scikit Learn

DATASET:

Name of dataset used is sklearn.datasets.load_diabetes with following properties:

Samples total	442
Dimensionality	10
Features	real, -.2 < x < .2
Targets	integer 25 - 346

Only first feature is used for model fitting.

features: ['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
X dim: 2
X rows/samples: 442
X cols/features: 10
y dim: 1
y rows/samples value: 442

PARAMETERES TUNED:

Model fitting was implemented with and without initial random weights

INTERPRETATION OF RESULT:

a) results of model fitting without initial weights (base case)

Coefficient of feature after training: [941.43097333]
Mean squared error: 3035.06
Coefficient of determination: 0.41

Coefficient of feature after training represents the weight assigned for feature after training.
Mean squared error regression loss: measure of error between trained and observed target values.
Coefficient of determination: determines model performance, best possible score is 1.0 and it can be negative (when model performance is worse).

b) results of model fitting with initial random weights

Coefficient of feature after training: [884.12091353]
Mean squared error: 3045.35
Coefficient of determination: 0.41

Conclusion: By considering initial random weights, the model fitting is better but MSE increases while model performance (coefficient of determination remains same).

B) Implementation of Logistic Regression using Scikit Learn

DATASET:

Name of dataset used is sklearn.datasets.load_iris with following properties:

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

names of dataset columns: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
target names: ['setosa', 'versicolor', 'virginica']
X dim: 2
X rows/samples: 150
X cols/features: 4
y dim: 1
y rows/samples value: 150

PARAMETERES TUNED:

fit_intercept
tol=0.01
multi_class='multinomial'
class_weight='balanced'
sample_weight
C
combination of all above parameters a) with strong and weak regularization

INTERPRETATION OF RESULT:

a) using default initialisation parameters (base case, tol = 0.0001)
log reg prediction score: 0.9733333333333334
coefficient of features in decision function: [[-0.42316549 0.96688516 -2.51669376 -1.08078412]
[0.53431322 -0.32159716 -0.20666272 -0.94355211]
[-0.11114773 -0.645288 2.72335648 2.02433623]]
bias: [9.84902418 2.23817992 -12.0872041]
number of iterations accross all classes: [115]

score: Return the mean accuracy on the given test data and labels.

b) using fit_intercept=False
log reg prediction score: 0.9666666666666667
coefficient of features in decision function: [[0.77433982 1.74702228 -2.34860274 -1.11227851]
[0.6234361 -0.09428645 -0.04476242 -1.02178783]
[-1.39777592 -1.65273583 2.39336516 2.13406633]]
bias: [0. 0. 0.]
number of iterations accross all classes: [50]

Conclusion: excluding bias in model fitting reduces number of iterations as bias values need not be optimised by this exclusion reduces accuracy of model.

c) using tol=0.1
log reg prediction score: 0.9733333333333334
coefficient of features in decision function: [[-0.42331915 0.96662855 -2.51650116 -1.08076895]
[0.53417397 -0.32126057 -0.20653936 -0.94363811]
[-0.11085481 -0.64536799 2.72304052 2.02440706]]
bias: [9.8496201 2.237713 -12.0873331]
number of iterations accross all classes: [111]

Conclusion: **reducing tolerance from 0.0001 to 0.1 yield same model performance and fitting, change in number of iterations is not significant.**

d) using multi_class='multinomial' or cross entropy loss function

log reg prediction score: 0.9733333333333334
coefficient of features in decision function: [[-0.42316549 0.96688516 -2.51669376 -1.08078412]
[0.53431322 -0.32159716 -0.20666272 -0.94355211]
[-0.11114773 -0.645288 2.72335648 2.02433623]]
bias: [9.84902418 2.23817992 -12.0872041]
number of iterations accross all classes: [115]

Conclusion: **changing training algorithm from one-vs-rest (OvR) scheme (default) to multinomial (cross entropy loss) does not affect model performance.**

e) using class_weight='balanced'

log reg prediction score: 0.9733333333333334
coefficient of features in decision function: [[-0.42316549 0.96688516 -2.51669376 -1.08078412]
[0.53431322 -0.32159716 -0.20666272 -0.94355211]
[-0.11114773 -0.645288 2.72335648 2.02433623]]
bias: [9.84902418 2.23817992 -12.0872041]
number of iterations accross all classes: [115]

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.

Conclusion: **no change in model performance with ref. to default model fitting case.**

f) using random initial sample weight

log reg prediction score: 0.96
coefficient of features in decision function: [[-0.45889117 0.66377428 -2.08894527 -0.85273276]
[0.43251561 -0.2952155 -0.0943191 -0.76583455]
[0.02637556 -0.36855879 2.18326437 1.61856731]]
bias: [9.05326679 1.69995271 -10.7532195]
number of iterations accross all classes: [74]

Conclusion: **with random initialized weights, number of iterations reduce but model accuracy score reduces indicated poorly fit model as compared to base model fitting.**

g) using stronger regularisation strength

log reg prediction score: 0.96
coefficient of features in decision function: [[-0.30245855 0.34626876 -1.1811841 -0.48002144]
[0.0709291 -0.33152656 0.07289077 -0.23053665]
[0.23152945 -0.0147422 1.10829333 0.71055808]]
bias: [5.32612723 1.59006332 -6.91619054]
number of iterations accross all classes: [89]

Low C value indicates stronger regularization, high C value indicates weaker regularization

Conclusion: **stronger regularization leads to lower model accuracy score while number of iterations decrease indicating model fitting is not at par with base model due to higher penalizing of weights during training.**

h) using all above combinations
log reg prediction score: 0.9133333333333333
coefficient of features in decision function: [[-0.24373896 0.23341934 -0.87173006 -0.36275391]
 [-0.01985738 -0.22886547 0.09325691 -0.0961044]
 [0.26359634 -0.00455387 0.77847316 0.45885831]]
bias: [4.07337858 1.17000796 -5.24338654]
number of iterations accross all classes: [47]

Conclusion: number of iterations reduces while model accuracy decreases considerably due to cascaded effect of multiple hyper parameter tuning, attempt should be made to determine optimum tuned values of hyper parameters for best fit.

i) tuning C using all above combinations for stronger regularisation
log reg prediction score: 0.8333333333333334
coefficient of features in decision function: [[-0.1045784   0.06228971 -0.31362175 -0.12972586]
 [0.00209185 -0.05910011 0.07291293 0.00850995]
 [0.10248655 -0.0031896   0.24070882 0.12121591]]
bias: [1.6916924 -0.01200964 -1.67968275]
number of iterations accross all classes: [37]

Conclusion: number of iterations reduces further while model accuracy decreases further considerably due to cascaded effect of multiple hyper parameter tuning, attempt should be made to determine optimum tuned values of hyper parameters for best fit.

C) Implementation of SVM using Scikit Learn

DATASET:
X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
y = np.array([1, 1, 2, 2])

PARAMETERES TUNED:
C, kernel function

INTERPRETATION OF RESULT:
a) default parameters (base case)
mean accuracy such that each label set be correctly predicted: 1.0
0 if correctly fitted, 1 otherwise: 0
intercept_: [1.13245812e-05]
get support vectors: [[-1. -1.]
 [-2. -1.]
 [1. 1.]
 [2. 1.]]
get indices of support vectors: [0 1 2 3]
get number of support vectors for each class: [2 2]

b) strong regularisation
mean accuracy such that each label set be correctly predicted: 1.0
0 if correctly fitted, 1 otherwise: 0
intercept_: [-0.]
get support vectors: [[-1. -1.]
 [-2. -1.]
 [1. 1.]
 [2. 1.]]
get indices of support vectors: [0 1 2 3]
get number of support vectors for each class: [2 2]

Conclusion: strong regularisation does not significantly affect model fitting for given dataset.

c) change kernel function to linear
mean accuracy such that each label set be correctly predicted: 1.0
Weights assigned to the features when kernel=linear: [[0.5 0.5]]
0 if correctly fitted, 1 otherwise: 0
intercept_: [-0.]
get support vectors: [[-1. -1.]
 [1. 1.]]
get indices of support vectors: [0 2]
get number of support vectors for each class: [1 1]

Conclusion: due to linear kernel function, number of support vectors for each class reduces to 1

d) change kernel function to polynomial of degree 3
mean accuracy such that each label set be correctly predicted: 1.0
0 if correctly fitted, 1 otherwise: 0
intercept_: [-0.]
get support vectors: [[-1. -1.]
 [1. 1.]]
get indices of support vectors: [0 2]
get number of support vectors for each class: [1 1]

Conclusion: polynomial based SVM yields similar results as that of linear kernel function, indicating model fitting in polynomial of degree 1 is best fit

D) Implementation of PCA using Scikit Learn

DATASET:
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])

PARAMETERES TUNED:
number of components to keep after implementation of PCA

INTERPRETATION OF RESULT:
a) pca model keeping all components (base case)

1.pca without whitening
Average log-likelihood of the samples under the current model: -2.3042258122025783
Percentage of variance explained by each of the selected components: [0.99244289 0.00755711]
2-norms of the n_components variables: [6.30061232 0.54980396]
2.pca with whitening
Average log-likelihood of the samples under the current model: -8.048352335770117
Percentage of variance explained by each of the selected components: [0.99244289 0.00755711]
2-norms of the n_components variables: [6.30061232 0.54980396]

explained_variance_ratio_: Percentage of variance explained by each of the selected components.

whiten: When True (False by default) the components_ vectors are multiplied by the square root of n_samples and then divided by the singular values to ensure uncorrelated outputs with unit component-wise variances.
Whitening will remove some information from the transformed signal (the relative variance scales of the components) but can sometime improve the predictive accuracy of the downstream estimators by making their data respect some hard-wired assumptions.

b) pca model keeping 1 component

1.pca without whitening
Average log-likelihood of the samples under the current model: -2.304225812202576
Percentage of variance explained by each of the selected components: [0.99244289]
2-norms of the n_components variables: [6.30061232]

2.pca with whitening
Average log-likelihood of the samples under the current model: -2.972979654859703
Percentage of variance explained by each of the selected components: [0.99244289]
2-norms of the n_components variables: [6.30061232]

Conclusion: whitening reduces average log-likelihood of the samples indicating reduced information content in PCA output.
keeping one component in PCA output increases information content with similar variance (with and without whitening)

c) pca model keeping 2 component

1.pca without whitening
Average log-likelihood of the samples under the current model: -2.3042258122025783
Percentage of variance explained by each of the selected components: [0.99244289 0.00755711]
2-norms of the n_components variables: [6.30061232 0.54980396]

2.pca with whitening
Average log-likelihood of the samples under the current model: -8.048352335770117
Percentage of variance explained by each of the selected components: [0.99244289 0.00755711]
2-norms of the n_components variables: [6.30061232 0.54980396]

Conclusion: n_components == min(n_samples, n_features) = min(6,2) = 2
hence above case is same as base case