A3: ADAPT TO CHANGE

AI 539 ML CHALLENGES

SUBMITTED BY : PUSHPAK VIJAY KATKHEDE

DUE DATE – 02/23/2023

1) Classifiers used in the assignment –
We have used 4 classifiers as follows :

1. Random Forest – The core unit of random forest classifiers is the decision tree. the bootstrapping technique helps the development of random forest with a set of required number of decision trees to improve classification accuracy.

Hyper parameter settings –
RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=3, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=0, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None)

2. Gaussian Process - It is based on the framework of the Gaussian process, a probabilistic model that specifies a distribution over functions. The decision boundary that divides various classes in the input space is being modeled in this instance. The decision boundary is a function deduced from a Gaussian process prior, and the Gaussian process classifier assumes that the input points follow a Gaussian distribution. Using the observed data, the model then applies Bayes' theorem to generate the posterior distribution over the decision boundary.
Hyper parameter settings –
GaussianProcessClassifier(kernel=None, *, optimizer='fmin_l_bfgs_b', n_restarts_optimizer=0, max_iter_predict=100, warm_start=False, copy_X_train=True, random_state=None, multi_class='one_vs_rest', n_jobs=None)

3. 3-Nearest Neighbor – KNeighborsClassifier is a supervised learning algorithm that makes classifications based on data neighbors. With KNN we can have a certain set of data and from it draw patterns that can classify or group our data.
Hyper parameter settings –
KNeighborsClassifier(n_neighbors=3, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)

4. 9-Nearest Neighbor – KNeighborsClassifier is a supervised learning algorithm that makes ` classifications based on data neighbors. With KNN we can have a certain set of data and from it draw patterns that can classify or group our data.
Hyper parameter settings –
KNeighborsClassifier(n_neighbors=9, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)

Baseline :

1. DummyClassifier with strategy most_frequent –

DummyClassifier is a machine learning classifier that offers a quick approach to build a basic model to assess how well more intricate models perform. It enables users to select a number of prediction strategies, including the "most frequent" strategy. The DummyClassifier simply predicts the most prevalent class in the training set for each input instance when the strategy parameter is set to "most frequent." When the dataset is unbalanced and one class is significantly more prevalent than the others or when the cost of misclassification is asymmetric, this approach can be helpful.

Hyper parameter settings –

DummyClassifier(*, *strategy='most_frequent'*, *random_state=None*, *constant=None*)

2. DummyClassifier with strategy stratified–

When the "strategy" parameter is set to "stratified", the DummyClassifier class will generate predictions by randomly guessing classes according to the training data's class distribution. This means that if the training data has a 60% proportion of class A and a 40% proportion of class B, the DummyClassifier will predict class A 60% of the time and class B 40% of the time. The "stratified" strategy is useful when dealing with imbalanced datasets, where one class has significantly fewer samples than the other classes. In such cases, using the "stratified" strategy can prevent the model from predicting only the majority class and ignoring the minority class.

Hyper parameter settings –

DummyClassifier(*, *strategy=stratified*, *random_state=None*, *constant=None*)

2) Tables –

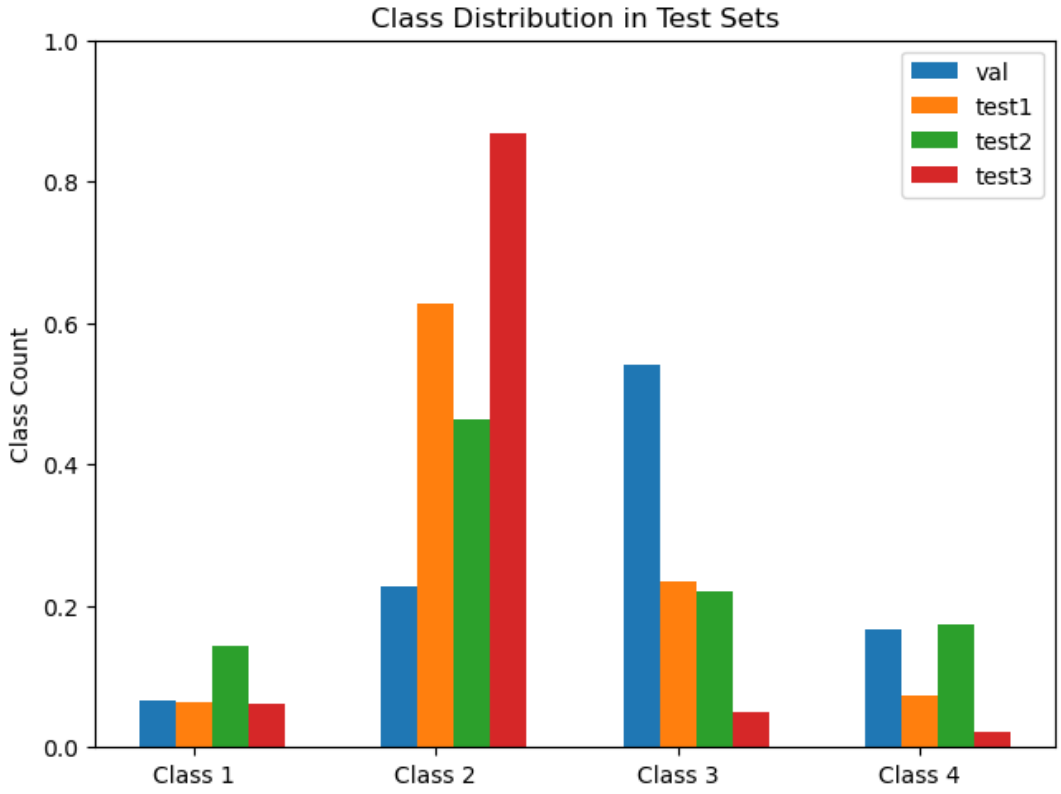
1. Accuracy comparison table before and after the BBSC

Accuracy	Val-TX	Test1-TX	Test2-FL	Test3-FL
Baseline: most_frequent	0.54	0.23	0.22	0.05
Baseline: stratified	0.39	0.29	0.26	0.24
RandomForest	0.61	['0.31', '0.15']	['0.22', '0.17']	['0.05', '0.02']
GaussianProcess	0.56	['0.33', '0.07']	['0.28', '0.17']	['0.14', '0.02']
3-NearestNeighbor	0.51	['0.38', '0.33']	['0.29', '0.31']	['0.25', '0.28']
9-NearestNeighbor	0.58	['0.37', '0.48']	['0.29', '0.38']	['0.23', '0.36']

2. Adaptation weights calculated through BBSC -

Accuracy	Test1-TX	Test2-FL	Test3-FL
RandomForest	['1.19', '2.24', '0.00', '6.50']	['0.07', '0.72', '0.00', '17.22']	['0.15', '0.87', '0.00', '23.37']
GaussianProcess	['0.16', '2.00', '0.00', '3.86']	['0.00', '1.54', '0.47', '2.51']	['0.00', '0.10', '0.00', '7.32']
3-NearestNeighbor	['0.38', '6.07', '0.00', '9.70']	['0.00', '4.98', '0.00', '8.54']	['0.00', '7.06', '0.00', '18.59']
9-NearestNeighbor	['0.20', '5.13', '0.00', '4.02']	['0.00', '3.60', '0.00', '2.76']	['0.00', '5.43', '0.00', '11.78']

Q8 Class Distribution Plot –



Q9 Answers –

A. In your own words, explain "label shift" and give an example.

Ans : - Label shift is a phenomenon that occurs when the distribution of labels in a dataset change's between the training and testing phases of a machine learning model. In other words, the proportion of different classes in the testing data is different from the proportion of classes in the training data. In other words, Label shift occurs when the distribution in the source data does not change however, the target distribution may change. Conceptually, the example of medical diagnosis for a disease can be hold as an example to explain. Label shift will occur when we train over a sample space and predict the outcome on test data where only few let's say for an instance 2% have the disease. In real time during the pandemic situation the actual number of people having disease may be 20% or say more. So here the distribution of classes in the target set i.e. real world is changed drastically. However, in label shift the input feature does not change with period and they remain constant.

Another example can be of email classification. The bulk of the emails in the training batch are spam, which helps the model learn how to categorize emails as spam or not. But when spammers adapt their strategies over time, reputable emails begin to resemble spam more and more. This results in a change in the labels' distribution, so there are fewer valid emails and more spam emails. As a result of not being trained on the new class of emails that are now considered spam, the model that was developed for the old distribution would perform worse on the new distribution.

B. In your own words, explain how BBSC works.

Ans - BBSC is based on the concept of BBSE which is used to identify the label shift in the data. Its is based on the fundamental idea of reducing the dimensionality and then performing the shift correct by using the shift corrections. BBSC leverages the confusion matrix to identify the label shift and the train / test distributions. The weights are initially calculated for each class which are the ratio of the newer predicted distribution versus the predictions made over previous state of data or mostly the validation data. These weights are identifiers for the balance of classes in the predicted label. Now, these adaptation weights can be used to update the posterior probabilities of the classifier which update the black box shift estimator by averaging method and hence, the classifier achiever a better generalization over the data. One important step here is the normalizing of the data distribution according to the newer calculated weights and helps us achieve a better classification accuracy after BBSC is applied.

C. In your experiments, in which cases did BBSC adaptation improve test set accuracy? How was better accuracy achieved?

Ans – In my experiments, I observed that BBSC adaptations improved the accuracy of the classifier's based on K – Nearest Neighbor system. The reason is that the adaptation weights which are calculated based on the confusion which has a better representation of Falsely predicted values as this are critical for calculating the adaptation weights. The opportunity to balance the posterior probabilities through the adapting weights is higher when there is comparable number of predictions for all the classes. The ideal case is when there are no zero values in the confusion which we can see for 3-KNNC and only one zero field in the 9-KNNC confusion matrix.

D. Did BBSC ever yield lower accuracy for a classifier after adaptation? If so, why do you think this happened?

Ans – Yes, I saw that BBSC yielded lower accuracies after adaption for Random Forest and gaussian process classifiers. For these classifiers the results calculated for the adaption weights are not distributed as they should be w.r.t the validation confusion matrix true labels. For and ideal condition to achieve the highest accuracy, they should be inversely proportional to the distribution of the True positives i.e., the diagonals of the confusion matrix of the classifiers. However, in other two classifiers that yield better accuracy as they had a closer distribution with the normalized inverse of these true positives. These observations are clear from the below confusion matrices calculated on the validation data. Here one more observation is for those two classifiers we can see that there are many cells which had zero values which has reduced the accuracy at the end.

CF for Random Forest

```
[ 23   0   0   0]
[  3   2  73   0]
[  1   0 185   0]
[  0   0  56   1]
```

CF for KNNC3

```
[ 20   2   0   1]
[  2  31  41   4]
[  2  50 118  16]
[  1  11  37   8]
```

CF for Gaussian Process

```
[ 19   2   1   1]
[  2  29  39   8]
[  1  38 127  20]
[  1   8  29  19]
```

CF for KNNC9

```
[ 17   3   1   2]
[  2  26  49   1]
[  0  27 152   7]
[  1   4  47   5]
```

E. What do you learn by looking at the distribution of true class labels in the three test sets? How did car accident severity change in each data set (vs. validation)

Ans – In the test 1 data set, which is same space and, but different time fraction has shown a significant hike severity 2 of accidents whereas other remain fairly same. The reason might be due to more important features i.e. humidity and pressure decrease in the summer than that of the winter where validation dataset is based on.

In the test 2 data set, which is same time fraction and, but different location has again shown a smaller hike in severity 2 of accidents whereas other remain same.

In the test 3 data set, which is both different time fraction and, different location has again shown an abnormal hike in severity 2 of accidents whereas other remain same.

From all of this we can speculate that the overall severity of the accidents is decreased (tends to class 2) as we move from Texas to Florida in space but also in time fraction i.e. from winter to summer. Due to valued decrease in more signification features.

F. What is one reason that would cause all BBSC adaptation weights to be 0.0?

Ans – As per my understanding and from the experiments provided in the file, I understood that if there exist a class/ or classes that are too infrequent (or in other words tends to be having nearly zero representation in a larger dataset) will cause all the BBSC adaptation weights to be zero. In other words, the confusion matrices generated for such cases will be degenerate. The major reason for this is that BBSC calculates the adaptation weights by performing the averaging on the inverse of confusion matrix. But, in this case the confusion matrix becomes uninvertible and hence, the weights become zero.

G. Reflection: What did you learn from completing this assignment? How long did it take (in hours)? What was the hardest part? What might you use in the future?

Ans – One of the most important things that I understood from the assignment is there needs to be a proper analysis of the data and possible shifts as they can be very substantial in the performance of the classifier. Also, while doing the background research for completing the assignment I came across many articles which suggested multiple methods to deal and address label shifts and how to identify them. The assignment took me a little longer this time approximating nearly 8-9 hours as I was doing more of a reading on the backend to understand it better. Fortunately, this time I did not feel any part to be hard as you had already provided the BBSC implementation which was supposed to be the hardest. I would be using BBSC in my project also as the dataset I am using does show the label shift and the covariate shift.

Q10) Extra Credit – Feature Importance-

- Ranked Table as per the feature Importance for classifier Random Forest used above.

Pressure(in)	0.361516
Humidity(%)	0.172828
Traffic_Signal	0.157766
Distance(mi)	0.146292
Wind_Chill(F)	0.0472482
Temperature(F)	0.034827
Wind_Speed(mph)	0.0344878
Visibility(mi)	0.0146668
Amenity	0.0065259
Junction	0.00575199
Crossing	0.00547306
Precipitation(in)	0.00456229
Railway	0.00415314
Stop	0.00168904
Give_Way	0.0015052
Station	0.000365977
Traffic_Calming	0.000341357
Bump	0
No_Exit	0
Roundabout	0
Turning_Loop	0

As per my initial analysis of the feature descriptions I suspected that the features visibility, junction, crossing, and traffic calming should be of the highest importance. However, from the results it is evident that they are ranked quite down the list. This itself was a surprise for me. Also, the feature bump has zero importance also surprised me as I was thinking there might be at least some importance to this feature as in real scenario there are accidents that happens due to bumps (I am myself a victim of such one case).

Q11) Extra Credit

(a) Which of the three test sets do you think exhibit covariate shift?

Ans – Amongst the given test sets I think that the Test3-FL exhibit covariate shift as there exist the spatiotemporal change, but the accuracies are most near to the validation dataset.

(B) What is one method for detecting covariate shift?

Ans - We can use the method importance weighting for detecting covariate shift. Importance weighting involves reweighting the training data so that the distribution of the training data matches the distribution of the test data. The idea is to assign higher weights to instances in the training data that are like the test data, and lower weights to instances that are dissimilar. When using importance weighting, each training instance is assigned a weight so that the total weight of all training instances equals the size of the training set. The density of the test data at the instance's feature values divided by the density of the training data at those same feature values is commonly used to calculate an instance's weight. Once the weights have been calculated, they can be used to reweight the training set so that its distribution matches that of the test set. We can identify and measure the degree of the covariate shift by contrasting the model's performance on the weighted and unweighted training data.

Sometimes, BBSC could also be used to detect covariate shift.

(C) What is one method for addressing covariate shift (to correct predictions and improve accuracy)?

Ans – Also, the same method of importance weighting can be used to correct predictions and improving the accuracy of the classifier. A machine learning model can be trained on the weighted data from previous experiment and assessed on the test data after reweighting the training data. The model's performance on the test data will typically be worse than its performance on the unweighted training data if the covariate shift is significant. We can identify and measure the degree of the covariate shift by contrasting the model's performance on the weighted and unweighted training data.