Code documentation

Note

This homework is submitted with a delay of 2 days with allowance from the Teaching Assistant in charge Ronen Nir.

Abstract

The python code implements full automation in preparing the features, loading them into a dataset, performing model evaluation, choosing the best model according to user-defined metrics, and making a classification on the desired test dataset and other unseen datasets.

The user must define:

- The features extractions dataset
- The tasks
- The models to test
- Validation metrics

The code includes explicit annotations and comments for better understanding

The general process

1. Feature extraction

The pipeline from the HW no. 2 was used for this purpose. Additional option to use different pipelines. As explained in HW2, the pipeline is made of stages, each performs a certain operation on some feature/features in the dataframe, and returns the dataframe with modified features. This way, the dataframe at the output from the pipeline contains dataframe, which is ready to be input to the classifier.

2. Defining the tasks

Since we have multiple tasks, we use the Class to define those. Each task definition required target specification (column name), and the metrics, upon which the best classifier should be chosen.

3. Defining the models

User defines the list of models and dictionary of parameters he wishes to test. If no parameters are passed, the single model is used. If parameters dictionary includes numerous parameters, the GridSearch Cross Validation is used to find the best parameters for this model (according to the metrics for a specific task that the learning is made upon).

4. Defining the validation metrics

User defines the metrics, which he wants to be tested on the validation dataset. For each task, all the models (all the best models after the CV process, if numerous parameters were specified) are later tested on all those metrics and the information is displayed in a table.

5. Choosing the best model

For each task, the loop over all models occurs. If numerous parameters were specified, the GridSearchCV from sklearn is used, according to the metrics specified for the specific task. The scored are saved for each model, and the best is chosen.

6. Evaluating on the test dataset

After the best models were chosen (they may be different for each task), they are evaluated on the given dataset. The results are then printed in the .csv file with predictions.

7. Logging

The code contains explicit print statements, which are saved in the log file for further examination. It is possible to run the whole script in a shell mode without losing valuable information.

1. Chosen models for the Virus task

1.1 Spreader

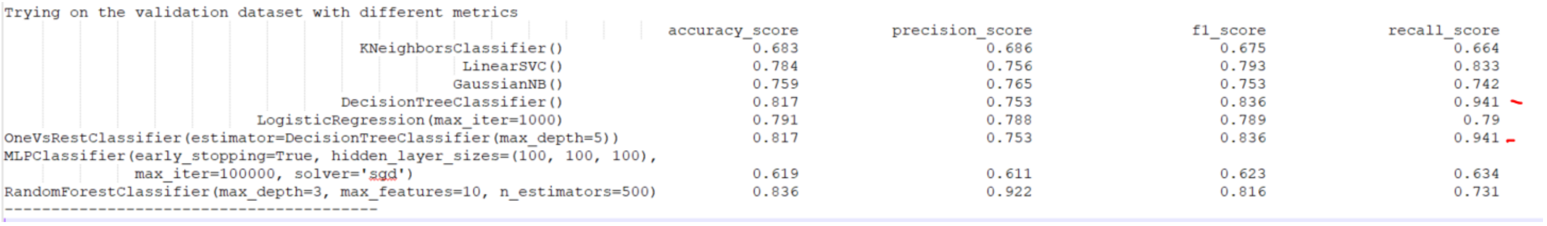
1.1.1 Process

Initially, we disregarded the pcrResult features, because they are empirical results of a test to check for existence of a virus and not to ascertain whether the person transmits the virus. Indicating transmission would rather be symptoms and physical health. This expert knowledge observation was confirmed once we used multiple classifiers, k-Nearest Neighbors, Decision Tree and Gaussian Naïve Bayes, and found out the f1, accuracy, and recall scores were not negatively affected.

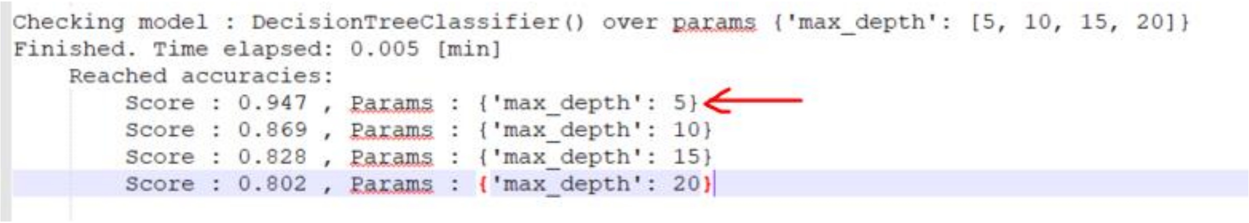
Secondly, we considered all the features from the original dataset and discarded only the ones that are redundant for any classification type, such as PatientID, location, and Address.

Thirdly, we used Sequential Forward Selection to find best features out of the remaining ones after the above first two elimination stages. We found out Sex and BloodType were least helpful in classification then we removed them.

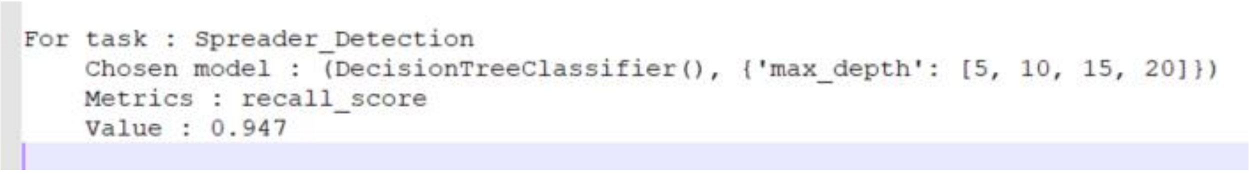
We tried out numerous models. The best one was Decision Tree Classifier, as can be seen from the output log:



```
Trying on the validation dataset with different metrics
                                                                  accuracy_score      precision_score          f1_score      recall_score
                                       KNeighborsClassifier()              0.683                0.686             0.675             0.664
                                                  LinearSVC()              0.784                0.756             0.793             0.833
                                                 GaussianNB()              0.759                0.765             0.753             0.742
                                     DecisionTreeClassifier()              0.817                0.753             0.836             0.941
                          LogisticRegression(max_iter=1000)                0.791                0.788             0.789              0.79
OneVsRestClassifier(estimator=DecisionTreeClassifier(max_depth=5))         0.817                0.753             0.836             0.941
MLPClassifier(early_stopping=True, hidden_layer_sizes=(100, 100, 100),
              max_iter=100000, solver='sgd')                               0.619                0.611             0.623             0.634
RandomForestClassifier(max_depth=3, max_features=10, n_estimators=500)     0.836                0.922             0.816             0.731
----------------------------------------
```



```
Checking model : DecisionTreeClassifier() over params {'max_depth': [5, 10, 15, 20]}
Finished. Time elapsed: 0.005 [min]
    Reached accuracies:
        Score : 0.947 , Params : {'max_depth': 5}
        Score : 0.869 , Params : {'max_depth': 10}
        Score : 0.828 , Params : {'max_depth': 15}
        Score : 0.802 , Params : {'max_depth': 20}
```



```
For task : Spreader_Detection
    Chosen model : (DecisionTreeClassifier(), {'max_depth': [5, 10, 15, 20]})
    Metrics : recall_score
    Value : 0.947
```

1.1.2 Scoring

Spreading the virus has grave consequences for the environment and the exponentiality of the infiltration of the virus in society makes it extremely important that it is stopped. Therefore, we used recall, i.e. TP / TP+FN, which penalizes heavily classifying someone who is spreading the virus as someone who is not.

1.1.3 Result

The chosen model for this task is Decision Tree with recall 94.7%.

1.2 Risk

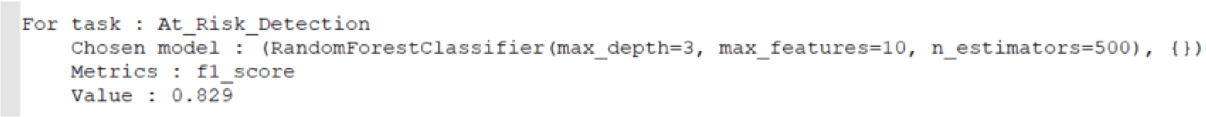
1.2.1 Process

Initially, we disregarded the pcrResult features, because they are empirical results of a test to check for existence of a virus and not to ascertain the physical capacity of the person to survive a virus. This expert knowledge observation was confirmed once we used multiple classifiers, k-Nearest Neighbors, Decision Tree and Gaussian Naïve Bayes, and found out the f1, accuracy, and recall scores were not negatively affected.

Secondly, we considered all the features from the original dataset and discarded only the ones that are redundant for any classification type, such as PatientID, location, and Address.

Thirdly, we used Sequential Forward Selection to find best features out of the remaining ones after the above first two elimination stages. We found out Sex and BloodType were least helpful in classification then we removed them.

We tried out numerous models as previously. The best one was Random Forest Classifier. The log is attached and this can be verified:



```
For task : At_Risk_Detection
    Chosen model : (RandomForestClassifier(max_depth=3, max_features=10, n_estimators=500), {})
    Metrics : f1_score
    Value : 0.829
```

1.2.2 Scoring

Low penalty for assessing someone who isn't at risk for disease as someone who is at risk (False Positive). On the other hand, high penalty for assessing someone who is at risk as being safe (False Negative). This implies high recall is crucial. Yet, the viruses aren't life-threatening in nearly all cases, then accuracy should be taken into account as well, because wrongly thinking he/she is at risk can still be a burden on someone's life.

Therefore, F1 score is ideal, because it can be thought of as a weighted average of precision and recall.

2.2.3 Result

The chosen model for this task is Decision Tree with 82.8% accuracy.

1.3 Disease

1.3.1 Process

Initially, we tested whether seemingly unrelated features to the presence of a virus affect classification. We performed that using Sequential Forward Selection with all of the features in the dataset, except the redundant ones, such as PatientID, Location, and Address. The unrelated features are :

```
“BMI", "DisciplineScore", "TimeOnSocialActivities","AgeGroup",'AvgMinSportsPerDay
','AvgHouseholdExpenseOnPresents','HappinessScore','StepsPerYear','NrCousins'
```

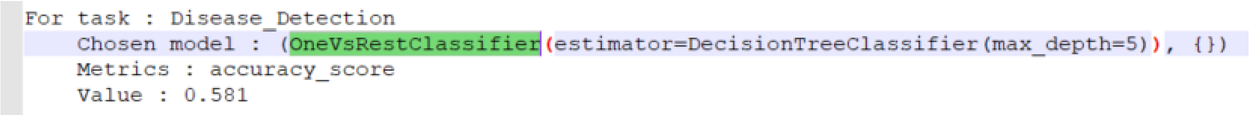
We found out they were detrimental to the classification task by the low accuracy score.

Secondly, we used SFS to check if all features were necessary, and indeed they were. Best classification score was with all the remaining features from the previous stages.

The binary classification of virus detected/not detected was unsurprisingly more accurate then classifying with a OneVsAll approach.

We have tried to avoid using the PCA and use the raw PCR results scores, scaled. This had lead to strong overfitting (we have reached around 0.85 accuracy on the training dataset and 0.3 on the validation dataset in the leading performance models). So, we came back into using the PCA results. We have tried different number of components, different scaling techniques, but didn’t succeed to improve the performance. If we had more time, we would have for sure found the way to improve the performance.

We tried out numerous models. Models like Logistic Regression and SVM did not converge even for high iterations number. The best one was Decision Tree Classifier by using the OneVsRest classifier technique generously supplied by sklearn.



```
For task : Disease_Detection
    Chosen model :  (OneVsRestClassifier(estimator=DecisionTreeClassifier(max_depth=5)), {})
    Metrics : accuracy_score
    Value : 0.581
```

1.3.2 Result

The chosen model for this task is Decision Tree with accuracy score of 0.581 on the validation dataset.

2. Best Features

RISK	SPREADER	DISEASE
SyndromeClass	SyndromeClass	TimeOnSocialActivities
BloodTypeInt	BloodTypeInt	DisciplineScore
SexInt	SexInt	BMI
NrCousins	NrCousins	pcrResult1,…..,pcrResult16
StepsPerYear	StepsPerYear	SexInt
HappinessScore	HappinessScore	BloodType
AvgHouseholdExpenseOnPresents	AvgHouseholdExpenseOnPresents	SyndromeClass
AvgMinSportsPerDay	AvgMinSportsPerDay	
AgeGroup	AgeGroup	
TimeOnSocialActivities	TimeOnSocialActivities	
DisciplineScore	DisciplineScore	
BMI	BMI	

<u>Analysis</u>

The disease classification is clearly largely determined by pcrResult's. pcr stands for Polymerase chain reaction and is a test that very accurately looks for virus presence. Yet, this test does not affect neither whether the person is at risk or not, because risk arises from physical factos, nor whether the person is a "super-spreader" , which is more determined by the symptoms of the person.

We do see that physical factors such as BMI and gender play a role in propensity to spreading, being at risk, and catching viruses.

Factors that are not purely physical affect spread and risk. These factors include time on social activities, which clearly impacts how much a person spreads a virus to other. Age has vital importance to risk of serious health repercussions. Amount of sports per day is clearly indicative of physical resilience, which means a person has milder symptoms and therefore transmits to less people and is at lower risk.

4.Conclusion

The classification task involves three distinct classification sub-tasks.

Disease Type requires accurately predicting whether there is a virus present and which type it is.

Risk stresses both accuracy and recall as a balance of not making the population overly cautious, yet taking the disease seriously.

Spread of the virus is of vital importance and requires high recall score for the virus not to spread and cause havoc.

We used Principal Component Analysis and Forward Feature Selection as preprocessing steps. Both were used for the purpose of dimensionality reduction. Scaling and filling nan values occurred during the preprocessing stage as well, in order to not outweigh any specific feature during training, which leads to approximation error.

Then using hyperparameter optimization we parsed through hyperparameters that were sparse yet few in order to test out possible tuning. Some models didn’t converge such as SVM. Overall the best model among the classification tasks was Decision Tree.

The evaluation on the previously unseen data is also attached, so is the whole code.

By running the ‘automatic_classification_main.py’ file you will automatically generate all the results that we have received.