Dataset Analysis

Supervised Learning by Oleg Lastocichin

06/01/2022

Questions and Tasks

1. Load and explore the dataset.
    a. The dataset had no missing values/ duplicates, I changed the dtypes for the data, and encoded it using sklearn's LaberBinarizer for Yes/No categorical columns and LabelEncoder for columns with more values, I didn't encode the gormiti_land for clearance, it will be encoded for the ensembly though, and I will make a choice to handle outliers if they're going to impact the models later.

2. Softmax Regression
    a. I further pre-processing was not necessary.
    b. Trained a logistic regression model as indicated through GridSearch with crossvalidation, I input these C as parameter grid

```
Parameter grid:
{'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100]}
```

the more the C increases the more the accuracy settles, and and it increases less and less while having big jumps of C value.
The best C was 100

c. The model is not overfitting the training and the test set have a score of 0.75.

```
softgrid = GridSearchCV(softmax_reg, param_grid, cv=5)
```

3. DecisionTree

a. Not necessary

b.

```
params = {'max_depth': [2, 5, 7, 10, 20, 25, 30, 40, 50, 100, 125, 150, 200], 'criterion': ['gini','entropy'],
'min_samples_split': [1, 3, 5], 'min_samples_leaf': [1, 3, 5], 'max_features': list(range(0,13)), 'max_leaf_nodes': [10, 20, 30]}
decisiongrid = GridSearchCV(DecisionTreeClassifier(random_state=24), params, n_jobs=-1, verbose=1, cv=3)

decisiongrid.fit(X_train, y_train)
```

The best parameters were:

- `{'criterion': 'gini', 'max_depth': 10, 'max_features': 11, 'max_leaf_nodes': 30, 'min_samples_leaf': 2, 'min_samples_split': 1}`

I researched good hyperparameters through gridsearch, and provided those parameters:

i. max_depth is how much the tree is allowed to grow, a too high value might overfit the training data and have errors in the testing set, likewise if its too little it will be problematic since it'll not be able to adapt well to our data.
ii. Criterion is the quality of the split
iii. Min_samples_split is how much we want to split the samples in the nodes, higher splits can reduce overfitting.
iv. Min_samples_leaf is a important hyperparameter since it's really relevant to reduce overfitting by limiting the splitting of nodes.
v. Max_features is useful when looking for the best split by reducing the amount of features to consider

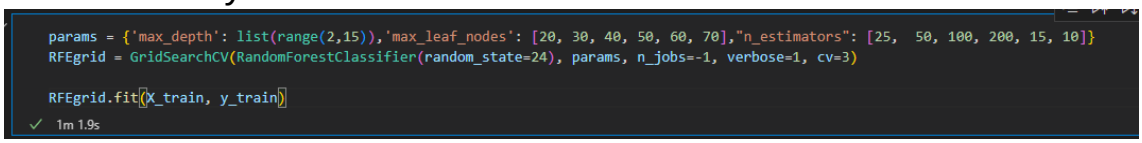
vi. Max_leaf_node is the maximum number of leaves that the tree is going to contain.

c. This model is slightly overfitting, with a difference in accuracy of 0.02 (train 0.789, test 0.7715) between the sets, decision trees are very prone to overfit the data but in this case it is almost negligible.

4. Random Forest

a. Not necessary

b.



The best parameters were:

- `{'max_depth': 11, 'max_leaf_nodes': 70, 'n_estimators': 200}`

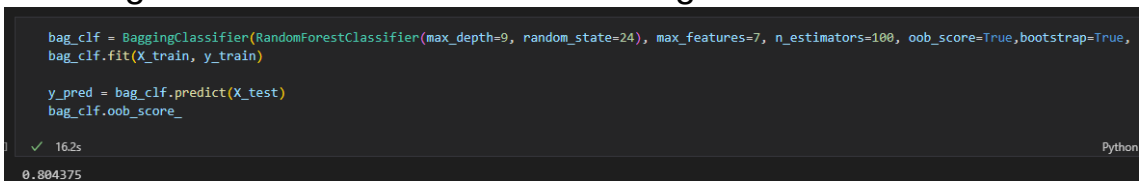
i. Max depth has the same use

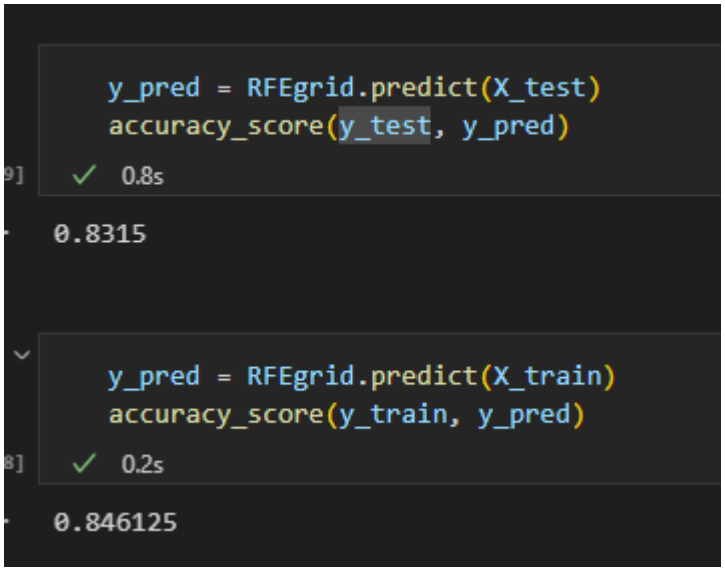
ii. Max_lead_nodes has the same use

iii. N_estimators is the amount of trees each group is, increasing this number doesn't always cause overfitting but increases complexity

c. The 2 most important features are Against_Lava and Against_Rock/Against_wind with a lot of weight on them while the categorical didn't have much impact on the model, this makes sense since where does a gormiti lives might have a higher impact on its resistances thus affecting its scores more than some categorical values.

d.

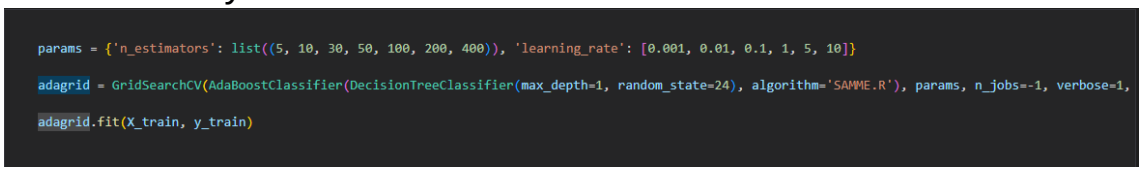


e. 

the model is slightly overfitting but it's almost negligible.

5. AdaBoost

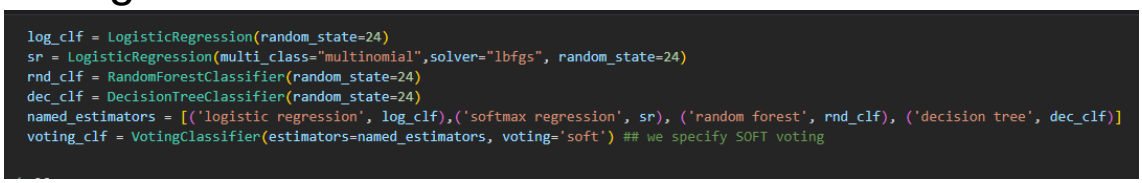
a. Not necessary

b. 

the parameters that need a tuning procedure are those specified, learning rate is the step size of each iteration, it can help to find the best accuracy.

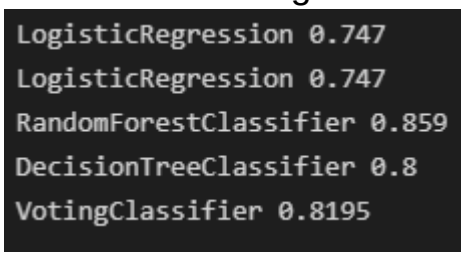
c. The model is not overfitting the difference is score is very little (less than 0.005)

6. Soft voting

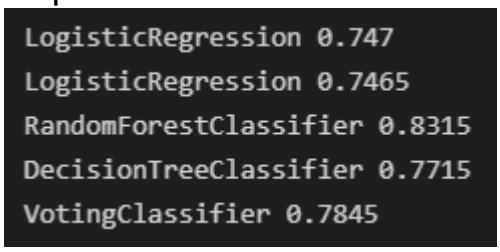
a. 

Did the models without the best parameters and

included a LogisticRegression, which performs the same as the softmax regression

b. 

c. The voting classifier performed slightly better than most except the random forest

d. # 

this one instead was using the best models out of each gridsearch, the voting classifier performed slightly worse

7. Blender

a. Some label encoding for the gormiti_land such that the code works.

```
sr = softgrid.best_estimator_
dec_clf = decisiongrid.best_estimator_
rnd_clf = RFEgrid.best_estimator_
ada_clf = adagrid.best_estimator_

estimators = [sr, dec_clf, rnd_clf, ada_clf]
```
✓ 0.2s

```
le = prepro.LabelEncoder()
blender_y = le.fit_transform(y_train)
```
✓ 0.2s

```
X_train_predictions = np.empty((len(X_train), len(estimators)), dtype=np.float32)

for index, estimator in enumerate(estimators):
    X_train_predictions[:, index] = le.fit_transform(estimator.predict(X_train))
```
✓ 0.4s

```
rnd_forest_blender = RandomForestClassifier(n_estimators=200, oob_score=True, random_state=24)
rnd_forest_blender.fit(X_train_predictions, blender_y)
```
✓ 0.7s

b. This model slightly overfit with a difference of 0.02 in the score (train 0.85, test 0.83)

8. Compare all

a. The worst model was AdaBoost with an accuracy of 0.7415 , while the best was Random Forest with an accuracy of 0.8315 the second best was the Ensemble with 0.83 they came very close in this case. Overall all the models had comparably good accuracy (not really high since we're aiming for a 100% accuracy on both sets but I think we need more data to gain it)