Artificial Intelligence and Machine Learning Fundamentals

Activity 11: Random Forest Classification for Your Car Rental Company

1. This section will optimize your classifier to satisfy your clients better when selecting future cars for your car fleet. We will be performing random forest and extreme random forest classification on your car dealership data set you worked on in Activity 1 of this lesson. Suggest further improvements to the model to improve the performance of the classifier.
   We can reuse Steps 1 – 5 of Activity 1. The end of Step 5 looks as follows:
   ```
   from sklearn import model_selection
   features_train, features_test, label_train, label_test =
   model_
   selection.train_test_split(
   features,
   label,
   test_size=0.1
   )
   ```
   If you are using IPython, your variables may already be accessible in your console.
2. Let's create a Random Forest and an Extremely Randomized Trees classifier and train the models.
   ```
   from sklearn.ensemble import
   RandomForestClassifier,ExtraTreesClassifier
   random_forest_classifier =
   RandomForestClassifier(n_estimators=100, max_
   depth=6)
   random_forest_classifier.fit(features_train, label_train)
   extra_trees_classifier =ExtraTreesClassifier(
   n_estimators=100, max_depth=6
   )
   extra_trees_classifier.fit(features_train, label_train)
   ```
3. Let's estimate how well the two models perform on the test data:
   ```
   from sklearn.metrics import classification_report
   print(
   classification_report(
   label_test,
   random_forest_classifier.predict(features_test)
   )
   )
   ```

The output for model 1 is as follows:

	precision	recall	f1-score	support
0	0.78	0.78	0.78	36
1	0.00	0.00	0.00	5
2	0.94	0.98	0.96	127
3	0.75	0.60	0.67	5
avg / total	0.87	0.90	0.89	173

The output for model 1 is as follows:

```
print(
classification_report(
label_test,
extra_trees_classifier.predict(features_test)
)
)
```

	Precision	recall	f1-score	support
0	0.72	0.72	0.72	36
1	0.00	0.00	0.00	5
2	0.93	1.00	0.96	127
3	0.00	0.00	0.00	5
avg / total	0.83	0.88	0.86	173

4. We can also calculate the accuracy scores:

```
random_forest_classifier.score(features_test, label_test)
```

The output is as follows:

0.9017341040462428

The output for **extraTreesClassifier** is as follows:

```
extra_trees_classifier.score(features_test, label_test)
```

The output is as follows:

0.884393063583815

We can see that the random forest classifier is performing slightly better than the extra trees classifier.

5. As a first optimization technique, let's see which features are more important and which features are less important. Due to randomization, removing the least important features may reduce the random noise in the model.

```
random_forest_classifier.feature_importances_
```

The output is as follows:

array([0.12656512, 0.09934031, 0.02073233, 0.35550329, 0.05411809, 0.34374086])

The output for **extra_trees_classifier** is as follows:

```
extra_trees_classifier.feature_importances_
```

The output is as follows:

array([0.08699494, 0.07557066, 0.01221275, 0.38035005, 0.05879822, 0.38607338])

Both classifiers treats the third and the fifth attributes quite unimportant. We may not be sure about the fifth attribute, as the importance score is more than 5% in both models. However, we are quite certain that the third attribute is the least significant attribute in the decision. Let's see the feature names once again.

```
data_frame_encoded.head()
```

The output is as follows:

Buying	Maintenance	Doors	Persons	LuggageBoot	Safety	Class
0	3	3	0	0	2	1
1	3	3	0	0	2	2
2	3	3	0	0	2	0
3	3	3	0	0	1	1
4	3	3	0	0	1	2

The least important feature is Doors. It is quite evident in hindsight: the number of doors doesn't have as big of an influence in the car's rating than the safety rating for instance.

6. Remove the third feature from the model and retrain the classifier.

```
features2 =
np.array(data_frame_encoded.drop(['Class', 'Doors'],
1))
label2 = np.array(data_frame_encoded['Class'])
features_train2,
features_test2,
label_train2,
label_test2 = model_selection.train_test_split(
features2,
label2,
test_size=0.1
)
random_forest_classifier2 = RandomForestClassifier(
n_estimators=100, max_depth=6
)
random_forest_classifier2.fit(features_train2,
label_train2)
extra_trees_classifier2 = ExtraTreesClassifier(
n_estimators=100, max_depth=6
)
extra_trees_classifier2.fit(features_train2,
label_train2)
```

7. Let's compare how well the new models fare compared to the original ones:

```
print(
classification_report(
label_test2,
random_forest_classifier2.predict(features_test2)
)
)
```

The output is as follows:

	Precision	recall	f1-score	support
0	0.89	0.85	0.87	40
1	0.00	0.00	0.00	3
2	0.95	0.98	0.96	125
3	1.00	1.00	1.00	5
avg / total	0.92	0.93	0.93	173

8. Second Model:

```
print(
classification_report(
label_test2,
```

```
extra_trees_classifier2.predict(features_test2)
)
)
```

The output is as follows:

	Precision	recall	f1-score	support
0	0.78	0.78	0.78	40
1	0.00	0.00	0.00	3
2	0.93	0.98	0.95	125
3	1.00	0.40	0.57	5
avg / total	0.88	0.90	0.88	173

Although we did improve a few percentage points, note that a direct comparison is not possible, because of following reasons. First, the train-test split selects different data for training and testing. A few badly selected data points may easily cause a few percentage point increase or decrease in the scores. Second, the way how we train the classifiers also has random elements.

This randomization may also shift the performance of the classifiers a bit. Always use best judgement when interpreting results and measure your results multiple times on different train-test splits if needed.

9. Let's tweak the parametrization of the classifiers a bit more. The following set of parameters increase the F1 Score of the Random Forest Classifier to 97%:

```
random_forest_classifier2 = RandomForestClassifier(
n_estimators=150,
max_ depth=8,
criterion='entropy',
max_features=5
)
random_forest_classifier2.fit(features_train2,
label_train2)
print(
classification_report(
label_test2,
random_forest_classifier2.predict(features_test2)
)
)
```

The output is as follows:

	Precision	recall	f1-score	support
0	0.95	0.95	0.95	40
1	0.50	1.00	0.67	3
2	1.00	0.97	0.98	125
3	0.83	1.00	0.91	5
avg / total	0.97	0.97	0.97	173

10. Using the same parameters on the Extra Trees Classifier, we also get surprisingly good results:

```
extra_trees_classifier2 = ExtraTreesClassifier(
n_estimators=150,
max_depth=8,
criterion='entropy',
max_features=5
```

```
)
extra_trees_classifier2.fit(features_train2,
label_train2)
print(
classification_report(
label_test2,
extra_trees_classifier2.predict(features_test2)
)
)
```

The output is as follows:

	Precision	recall	f1-score	support
0	0.92	0.88	0.90	40
1	0.40	0.67	0.50	3
2	0.98	0.97	0.97	125
3	0.83	1.00	0.91	5
avg / total	0.95	0.94	0.94	173