InterpretableML.with.LIME.TabularData

January 16, 2024

1 Use Case: Interpretable Machine Learning with LIME for tabular data

- Course: AML
- Lecturer: Gernot Heisenberg
- Author of notebook: Gernot Heisenberg
- Date: 28.08.2023

1.0.1 Description

This is one implementation example for LIME interpreting a random forest ensemble model that predicts tablua data. PLease try to understand the implementation.

1.0.2 Imports

Import all necessary python utilities for loading, preprocessing and predicting the data.

NOTE In some cases you need to downgrade to scikit-learn-1.2.2 version to get it running

[1]:
```
from utils import DataLoader
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score, accuracy_score
from interpret.blackbox import LimeTabular
from interpret import show

# get rid of those LIME warnings
import warnings
warnings.filterwarnings('ignore')
```

1.0.3 Load and preprocess the data

1.1 load the data

```
[2]: data_loader = DataLoader()
     data_loader.load_dataset()
     data_loader.preprocess_data()
```

1.2 Split the data for evaluation

```
[3]: X_train, X_test, y_train, y_test = data_loader.get_data_split()
```

1.3 Oversample the train data

```
[4]: X_train, y_train = data_loader.oversample(X_train, y_train)
     print("X_train.shape", X_train.shape)
     print("X_test.shape", X_test.shape)
```

```
X_train.shape (7776, 21)
X_test.shape (1022, 21)
```

1.3.1 Fit the blackbox model (Random Forest Classifier) to the stroke data

For the case of image explanations, perturbations will be generated by turning on and off some of the superpixels in the image.

```
[5]: rf = RandomForestClassifier()
     rf.fit(X_train, y_train)
     y_pred = rf.predict(X_test)

     print(f"F1 Score {f1_score(y_test, y_pred, average='macro')}")
     print(f"Accuracy {accuracy_score(y_test, y_pred)}")
```

```
F1 Score 0.5275288937809577
Accuracy 0.9383561643835616
```

1.3.2 Apply LIME

Initilize Lime for being used with tabular data

```
[6]: X_train = X_train.astype(float)
```

```
[7]: lime = LimeTabular(model=rf, data=X_train, random_state=1)
```

1.3.3 Get local explanations and interpret the results

[8]:
```
lime_local = lime.explain_local(X_test[-10:], y_test[-10:], name='LIME')
show(lime_local)
```

[ ]:
```
```