Wrapper Methods

May 15, 2021

[7]:
```
# Forward Feature Selection
from mlxtend.feature_selection import SequentialFeatureSelector
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression


dataset = pd.read_csv('dataset.csv')
X= dataset.drop(columns='Result')
Y= dataset['Result']
# X.head()
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2)

lr = LogisticRegression(class_weight='balanced', solver='lbfgs',␣
 ↪random_state=42, n_jobs=-1, max_iter=500)
ffs = SequentialFeatureSelector(lr, k_features='best', forward=True, n_jobs=-1)
ffs.fit(X,Y)
features = list(ffs.k_feature_names_)
features = list(map(str, features))

print(features)

# lr.fit(X_train[features], Y_train)
# y_pred = lr.predict(X_train[features])
# print(y_pred)
```

```
['Links_in_tags', 'SFH', 'Submitting_to_email', 'Abnormal_URL', 'on_mouseover',
'popUpWidnow', 'Iframe']
```

[8]:
```
# Backward Feature Selection
from mlxtend.feature_selection import SequentialFeatureSelector
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression


dataset = pd.read_csv('dataset.csv')
X= dataset.drop(columns='Result')
```

```
Y= dataset['Result']
# X.head()
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2)

lr = LogisticRegression(class_weight='balanced', solver='lbfgs',␣
 ↪random_state=42, n_jobs=-1, max_iter=500)
bfs = SequentialFeatureSelector(lr, k_features='best', forward=False, n_jobs=-1)
bfs.fit(X,Y)
features = list(bfs.k_feature_names_)
features = list(map(str, features))

print(features)

# lr.fit(X_train[features], Y_train)
# y_pred = lr.predict(X_train[features])
# print(y_pred)
```

```
['Links_in_tags', 'SFH', 'Submitting_to_email', 'Abnormal_URL', 'on_mouseover',
'popUpWidnow', 'Iframe']
```

[9]:
```
# Exhaustive Feature Selection
from mlxtend.feature_selection import ExhaustiveFeatureSelector

#import algorythm u want to evaluate on your features
from sklearn.ensemble import RandomForestClassifier

# create the exhaustive feature selector object
efs = ExhaustiveFeatureSelector(RandomForestClassifier(),
                               min_features=4,
                               max_features=8,
                               scoring='roc_auc',
                               cv=2)

efs = efs.fit(X,Y)

selected_features = X_train.columns[list(efs.best_idx_)]
print(selected_features)
```

```
Features: 381/381

Index(['Links_in_tags', 'Submitting_to_email', 'Abnormal_URL', 'Redirect',
       'on_mouseover', 'RightClick', 'popUpWidnow', 'Iframe'],
      dtype='object')
```

[10]:
```
print(efs.best_score_)
```

```
0.7341994884162122
```

```
[19]: #Recursive Feature Elimination
      from sklearn.feature_selection import RFE

      rfe = RFE(lr, n_features_to_select=7) #choose number of columns to be selected
      rfe = rfe.fit(X_train,Y_train)

      # summarize all features
      for i in range(X.shape[1]):
              print('Column: %d, Selected %s, Rank: %.3f col_name: %s' % (i, rfe.
       ↪support_[i], rfe.ranking_[i], X_train.columns[i]))
```

```
Column: 0, Selected True, Rank: 1.000 col_name: Links_in_tags
Column: 1, Selected True, Rank: 1.000 col_name: SFH
Column: 2, Selected False, Rank: 2.000 col_name: Submitting_to_email
Column: 3, Selected True, Rank: 1.000 col_name: Abnormal_URL
Column: 4, Selected True, Rank: 1.000 col_name: Redirect
Column: 5, Selected True, Rank: 1.000 col_name: on_mouseover
Column: 6, Selected True, Rank: 1.000 col_name: RightClick
Column: 7, Selected False, Rank: 3.000 col_name: popUpWidnow
Column: 8, Selected True, Rank: 1.000 col_name: Iframe
```

```
[ ]:
```