Exercise Set 5c

Hina Arora

(5c)
Grid-Search with Cross-Validation

https://scikit-learn.org/stable/modules/grid_search.html

Grid-Search

“It is possible and recommended to search the hyper-parameter space for the best cross validation score.

Any parameter provided when constructing an estimator may be optimized in this manner.

A search consists of:

- an estimator (regressor or classifier such as sklearn.svm.SVC());
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme; and
- a score function.

GridSearchCV exhaustively considers all parameter combinations.”

Copy

```
df = pd.read_csv('data/kaggleTitanic/train.csv’)

df['Deck'] = df['Cabin'].apply(lambda x: x[0] if pd.notna(x) else np.nan)

X = df.drop(['Survived'], axis=1)
y = df['Survived’]

Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.2, random_state=1)

Xtrain = Xtrain.copy()
Xtest = Xtest.copy()
ytrain = ytrain.copy()
ytest = ytest.copy()
```

Copy

```
numeric_features = ['Age']
numeric_transformer = Pipeline(steps=[
    ('si', SimpleImputer(missing_values=np.nan, strategy='median'))])

categorical_features = ['Pclass', 'Sex', 'Deck']
categorical_transformer = Pipeline(steps=[
    ('si', SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='X')),
    ('ohe', OneHotEncoder(sparse=False, dtype=int, handle_unknown='ignore'))])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)],
    remainder='drop’)

clf = Pipeline(steps=[('pp', preprocessor),
                      ('lr', LogisticRegression(solver='liblinear'))])
```

Copy

```
# Set up grid search

from sklearn.model_selection import GridSearchCV

param_grid = {
   'lr__penalty': ['l1', 'l2’]
}

gscv = GridSearchCV(
                    clf,
                    param_grid,
                    cv=5,
                    return_train_score=False)
```

Copy

```
# Search for best params

gscv.fit(Xtrain, ytrain)

print(gscv.best_estimator_, "\n")
print(gscv.best_score_, "\n")
print(gscv.best_params_, "\n")
print(gscv.cv_results_, "\n")
```

Copy

```
# Predict and Evaluate best_estimator_ on test data

ypred = gscv.best_estimator_.predict(Xtest)

from sklearn import metrics
print (metrics.accuracy_score(ytest, ypred))
print (metrics.confusion_matrix(ytest, ypred))
print (metrics.classification_report(ytest, ypred))
```

Question

Update param_grid to search for

- lr penalty l1, l2
- si strategy mean and median

Answer

 # Step 1: make following change and rerun this step

```
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer

numeric_features = ['Age’]

numeric_transformer = Pipeline(steps=[
    ('si', SimpleImputer(missing_values=np.nan, strategy='median'))])
```

```
# Step 2: make the following change, and rerun this step
#             and all the remaining steps that followed this step

from sklearn.model_selection import GridSearchCV

param_grid = {
    'lr__penalty': ['l1', 'l2’],
    'pp__num__si__strategy':['median','mean']
}

gscv = GridSearchCV(
                    clf,
                    param_grid,
                    cv=5,
                    return_train_score=False)
```