

Pattern & Anomaly Detection Lab

Experiment 8

Nested Cross Validation with Hyperparameter Tuning

Submitted By:

Dhruv Singhal

500075346

R177219074

AIML B3

Submitted To:

Dr. Gopal Phartiyal

Asst. Professor

SOCS

UPES

Hyperparameter Tuning and Nested Cross validation on Linear Regression

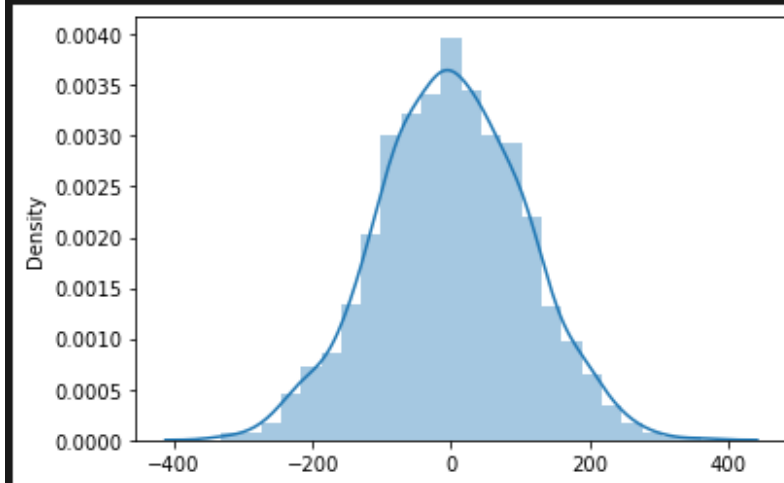
CODE:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 import warnings
5 warnings.filterwarnings('ignore')
6
7 from sklearn.datasets import make_regression
8 x,y=make_regression(n_samples=1000,n_features=5,noise=20)
9 sns.distplot(y)
10
11
12 plt.hist(x)
13
14 x_train = x
15 y_train =y
16
17
18 from sklearn.model_selection import GridSearchCV, cross_val_score
19 from sklearn.model_selection import KFold
20 from sklearn.linear_model import Ridge
21 NUM_TRIALS = 30
22
23
24 tuned_parameters = [{'solver' : ['svd', 'lsqr'],'fit_intercept': ['True'],'normalize': ['False']],
25                     {'solver' : ['sag', 'cholesky'],'fit_intercept': ['False'],'normalize': ['true']}]
26
27 score = 'r2'
28 non_nested_scores = np.zeros(NUM_TRIALS)
29 nested_scores = np.zeros(NUM_TRIALS)
30
31 # Loop for each trial
32 for i in range(NUM_TRIALS):
33
34     # model= GridSearchCV(linear_model.LinearRegression(), tuned_parameters, scoring= score)
35     inner_cv = KFold(n_splits=4, shuffle=True, random_state=i)
36     outer_cv = KFold(n_splits=4, shuffle=True, random_state=i)
37     model= GridSearchCV(estimator = Ridge(), param_grid = tuned_parameters, scoring = score)
38     model.fit(x_train, y_train)
39     non_nested_scores[i] = model.best_score_
40
41
42     # Nested CV with parameter optimization
43     model = GridSearchCV(estimator= Ridge(), param_grid = tuned_parameters, cv=inner_cv, scoring= score)
44     nested_score = cross_val_score(model, X=x_train, y=y_train, cv=outer_cv)
45     nested_scores[i] = nested_score.mean()
46
47
48 score_difference = non_nested_scores - nested_scores
49
50 print("Average difference of {:.6f} with std. dev. of {:.6f}."
51       .format(score_difference.mean(), score_difference.std()))
52
53
54
```

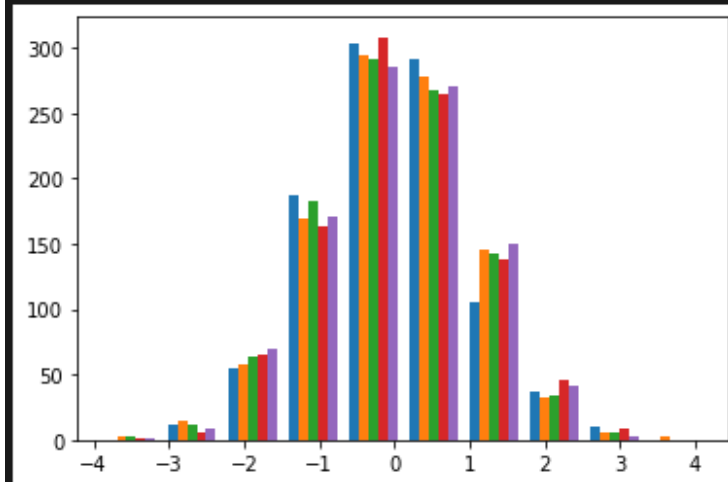
OUTPUT:

```
In [1]: runcell(0, 'B:/3rd year/5th sem/P&AD/exp9.py')
```

```
In [2]: runcell(1, 'B:/3rd year/5th sem/P&AD/exp9.py')
```



```
In [3]: runcell(2, 'B:/3rd year/5th sem/P&AD/exp9.py')
```



```
In [4]: runcell(3, 'B:/3rd year/5th sem/P&AD/exp9.py')
```

```
In [5]: runcell(4, 'B:/3rd year/5th sem/P&AD/exp9.py')
Average difference of -0.003284 with std. dev. of 0.001567.
```

Hyperparameter Tuning and Nested Cross validation on Logistic Regression

CODE:

```
1 import numpy as np
2 from sklearn.datasets import make_classification
3 from sklearn import linear_model
4 #from matplotlib import pyplot as plt
5 from sklearn.model_selection import GridSearchCV
6 from sklearn.model_selection import KFold
7 from sklearn.model_selection import cross_val_score
8 import warnings
9 warnings.filterwarnings('ignore')
10 #%%
11 # Generating Data
12 X, y = make_classification(n_samples = 1000, n_features = 5, n_classes = 2)
13 x_train = X
14 y_train = y
15 #%%
16 NUM_TRIALS = 30
17
18 tuned_parameters = [{ 'penalty' : ['l1', 'l2', 'elasticnet', 'none'],
19   'C' : np.logspace(-4, 4, 20),
20   'solver' : ['lbfgs', 'newton-cg', 'liblinear', 'sag', 'saga']
21   }]
22
23 score = 'accuracy'
24 non_nested_scores = np.zeros(NUM_TRIALS)
25 nested_scores = np.zeros(NUM_TRIALS)
26 # Loop for each trial
27 for i in range(NUM_TRIALS):
28
29     # model= GridSearchCV(linear_model.LogisticRegression(), tuned_parameters, scoring= score)
30     inner_cv = KFold(n_splits=4, shuffle=True, random_state=i)
31     outer_cv = KFold(n_splits=4, shuffle=True, random_state=i)
32     model= GridSearchCV(estimator = linear_model.LogisticRegression(), param_grid = tuned_parameters, scoring = score)
33     model.fit(x_train, y_train)
34     print(model.best_params_)
35     non_nested_scores[i] = model.best_score_
36
37
38     # Nested CV with parameter optimization
39     model = GridSearchCV(estimator= linear_model.LogisticRegression(), param_grid = tuned_parameters, cv=inner_cv, scoring= score)
40     nested_score = cross_val_score(model, X=x_train, y=y_train, cv=outer_cv)
41     nested_scores[i] = nested_score.mean()
42 score_difference = non_nested_scores - nested_scores
43 print("Average difference of {:.6f} with std. dev. of {:.6f}.".format(score_difference.mean(), score_difference.std()))
44
45 #%%
```

OUTPUT:

```
In [1]: runcell(0, 'B:/3rd year/5th sem/P&AD/LogisticTuning.py')
```

```
In [2]: runcell(1, 'B:/3rd year/5th sem/P&AD/LogisticTuning.py')
```

```
In [3]: runcell(2, 'B:/3rd year/5th sem/P&AD/LogisticTuning.py')
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
{'C': 0.23357214690901212, 'penalty': 'l1', 'solver': 'liblinear'}
Average difference of 0.001900 with std. dev. of 0.001719.
```

Hyperparameter Tuning and Nested Cross validation on Logistic Regression (Breast Cancer Dataset)

CODE:

```
1 import numpy as np
2 #from sklearn.datasets import make_classification
3 from sklearn import linear_model
4 #from matplotlib import pyplot as plt
5 from sklearn.model_selection import GridSearchCV
6 from sklearn.model_selection import KFold
7 from sklearn.model_selection import cross_val_score
8 import warnings
9 warnings.filterwarnings('ignore')
10 ###
11 # Generating Data
12 from sklearn.datasets import load_breast_cancer
13
14 data = load_breast_cancer()
15
16 X = data.data
17
18 y = data.target
19 x_train = X
20 y_train = y
21 ###
22 NUM_TRIALS = 30
23
24
25 tuned_parameters = [{ 'penalty' : ['l1', 'l2', 'elasticnet', 'none'],
26                        'solver' : ['lbfgs', 'newton-cg', 'liblinear', 'sag', 'saga']
27                      }]
28 score = 'accuracy'
29 non_nested_scores = np.zeros(NUM_TRIALS)
30 nested_scores = np.zeros(NUM_TRIALS)
31 # Loop for each trial
32 for i in range(NUM_TRIALS):
33
34     # model= GridSearchCV(linear_model.LogisticRegression(), tuned_parameters, scoring= score)
35     inner_cv = KFold(n_splits=4, shuffle=True, random_state=i)
36     outer_cv = KFold(n_splits=4, shuffle=True, random_state=i)
37     model= GridSearchCV(estimator = linear_model.LogisticRegression(), param_grid = tuned_parameters, scoring = score)
38     model.fit(x_train, y_train)
39     print(model.best_params_)
40     non_nested_scores[i] = model.best_score_
41
42
43     # Nested CV with parameter optimization
44     model = GridSearchCV(estimator= linear_model.LogisticRegression(), param_grid = tuned_parameters, cv=inner_cv, scoring= score)
45     nested_score = cross_val_score(model, X=x_train, y=y_train, cv=outer_cv)
46     nested_scores[i] = nested_score.mean()
47 score_difference = non_nested_scores - nested_scores
48 print("Average difference of {:.6f} with std. dev. of {:.6f}.".format(score_difference.mean(), score_difference.std()))
49
50
```

OUTPUT:

```
In [1]: runcell(0, 'B:/3rd year/5th sem/P&AD/Dataset(Logistic_Hyper_Tuning).py')
```

```
In [2]: runcell(1, 'B:/3rd year/5th sem/P&AD/Dataset(Logistic_Hyper_Tuning).py')
```

```
In [3]: runcell(2, 'B:/3rd year/5th sem/P&AD/Dataset(Logistic_Hyper_Tuning).py')
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

```
{'penalty': 'none', 'solver': 'newton-cg'}
```

Average difference of 0.009804 with std. dev. of 0.006161.