50. Cross-Validation in Machine Learning (Practical)

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
dataset = pd.read_csv(r'Data/placement.csv')
dataset.head(3)
```

Out[2]:

	cgpa	package
0	6.89	3.26
1	5.12	1.98
2	7.82	3.25

In [ ]:

```

```

In [3]:

```
x = dataset.iloc[:,:-1]
y = dataset['package']
```

50.1 Check how much accuracy this data can have

We will use cross-validation

In [14]:

```
# Below model will ask for estimator (on which model you want to train it on)
from sklearn.linear_model import LinearRegression
```

In [16]:

```
from sklearn.model_selection import cross_val_score
```

In [18]:

```
# cv: cross-validation: number or LeaveOneOut, LeavePOut, KFold, Stratified, KFold
p = cross_val_score(LinearRegression(), x,y,cv=5)
p
```

Out[18]:

```
array([0.75398043, 0.79051763, 0.75683837, 0.78086775, 0.70887127])
```

In [20]:

```
p.sort()
p*100
```

Out[20]:

```
array([70.88712673, 75.39804264, 75.68383749, 78.0867752 , 79.05176315])
```

min_accuracy: 70% and max_accuracy: 79%

In [ ]:

```

```

In [24]:

```
# cv: cross-validation: number or LeaveOneOut, LeavePOut, KFold, StratifiedKFold
p1 = cross_val_score(LinearRegression(), x,y,cv=KFold(n_splits=10))
p1.sort()
p1*100
```

Out[24]:

```
array([60.48000765, 65.67540106, 67.20523867, 69.890411  , 73.50599138,
       74.37616704, 80.3181025 , 82.0986355 , 82.64799643, 83.96333567])
```

min_accuracy: 60% and max_accuracy: 83%

In [ ]:

```

```

50.2 Cross-Validation Methods

In [4]:

```
new_data = dataset.head(10)
```

In [9]:

```
x_new = new_data.iloc[:,:-1]
y_new = new_data['package']
```

In [6]:

```
from sklearn.model_selection import LeaveOneOut, LeavePOut, KFold, StratifiedKFold
```

In [10]:

```
lo = LeaveOneOut()

for train, test in lo.split(x_new,y_new):
    print(train, test)
```

```
[1 2 3 4 5 6 7 8 9] [0]
[0 2 3 4 5 6 7 8 9] [1]
[0 1 3 4 5 6 7 8 9] [2]
[0 1 2 4 5 6 7 8 9] [3]
[0 1 2 3 5 6 7 8 9] [4]
[0 1 2 3 4 6 7 8 9] [5]
[0 1 2 3 4 5 7 8 9] [6]
[0 1 2 3 4 5 6 8 9] [7]
[0 1 2 3 4 5 6 7 9] [8]
[0 1 2 3 4 5 6 7 8] [9]
```

In [12]:

```
lp = LeavePOut(p=2)

for train, test in lp.split(x_new,y_new):
    print(train, test)
```

```
[2 3 4 5 6 7 8 9] [0 1]
[1 3 4 5 6 7 8 9] [0 2]
[1 2 4 5 6 7 8 9] [0 3]
[1 2 3 5 6 7 8 9] [0 4]
[1 2 3 4 6 7 8 9] [0 5]
[1 2 3 4 5 7 8 9] [0 6]
[1 2 3 4 5 6 8 9] [0 7]
[1 2 3 4 5 6 7 9] [0 8]
[1 2 3 4 5 6 7 8] [0 9]
[0 3 4 5 6 7 8 9] [1 2]
[0 2 4 5 6 7 8 9] [1 3]
[0 2 3 5 6 7 8 9] [1 4]
[0 2 3 4 6 7 8 9] [1 5]
[0 2 3 4 5 7 8 9] [1 6]
[0 2 3 4 5 6 8 9] [1 7]
[0 2 3 4 5 6 7 9] [1 8]
[0 2 3 4 5 6 7 8] [1 9]
[0 1 4 5 6 7 8 9] [2 3]
[0 1 3 5 6 7 8 9] [2 4]
[0 1 3 4 6 7 8 9] [2 5]
[0 1 3 4 5 7 8 9] [2 6]
[0 1 3 4 5 6 8 9] [2 7]
[0 1 3 4 5 6 7 9] [2 8]
[0 1 3 4 5 6 7 8] [2 9]
[0 1 2 5 6 7 8 9] [3 4]
[0 1 2 4 6 7 8 9] [3 5]
[0 1 2 4 5 7 8 9] [3 6]
[0 1 2 4 5 6 8 9] [3 7]
[0 1 2 4 5 6 7 9] [3 8]
[0 1 2 4 5 6 7 8] [3 9]
[0 1 2 3 6 7 8 9] [4 5]
[0 1 2 3 5 7 8 9] [4 6]
[0 1 2 3 5 6 8 9] [4 7]
[0 1 2 3 5 6 7 9] [4 8]
[0 1 2 3 5 6 7 8] [4 9]
[0 1 2 3 4 7 8 9] [5 6]
[0 1 2 3 4 6 8 9] [5 7]
[0 1 2 3 4 6 7 9] [5 8]
[0 1 2 3 4 6 7 8] [5 9]
[0 1 2 3 4 5 8 9] [6 7]
[0 1 2 3 4 5 7 9] [6 8]
[0 1 2 3 4 5 7 8] [6 9]
[0 1 2 3 4 5 6 9] [7 8]
[0 1 2 3 4 5 6 8] [7 9]
[0 1 2 3 4 5 6 7] [8 9]
```

In [13]:

```
kf = KFold(n_splits=5)

for train, test in kf.split(x_new,y_new):
    print(train, test)
```

```
[2 3 4 5 6 7 8 9] [0 1]
[0 1 4 5 6 7 8 9] [2 3]
[0 1 2 3 6 7 8 9] [4 5]
[0 1 2 3 4 5 8 9] [6 7]
[0 1 2 3 4 5 6 7] [8 9]
```

In [ ]:

```
sf = StratifiedFold(n_splits=5)

for train, test in kf.split(x_new,y_new):
    print(train, test)

# It will generate error, b/c it works only in classification analysis, and don't w
```

In [ ]: