Machine Learning with Python

1 Part1

Most of the time, it's also possible to **convert a supervised dataset to unsupervised**

Firstly, first and foremost, let's start with importing required libraries.

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

from sklearn import datasets
from sklearn import manifold
```

1.1 Cross-Validation

- k-fold cross-validation
- stratified k-fold cross-validation
- hold-out based validation
- leave-one-out cross-validation
- group k-fold cross-validation

Occam's razor in simple words states that one should not try to complicate things that can be solved in a much simpler manner. In other words, the simplest solutions are the most generalizable solutions. In general, whenever your model does not obey Occam's razor, it is probably overfitting.

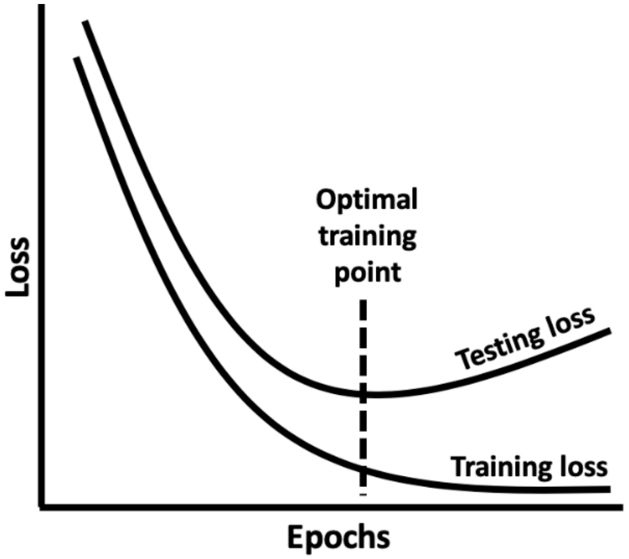


Figure 1: General Understanding for Over-fitting

Regarding stratified k-fold cross-validation, the important technique is that you will have k-different models which have different parameters, and these k-different model will be used for creating a inference which will be created by merging or ensembling from k-different inference results.

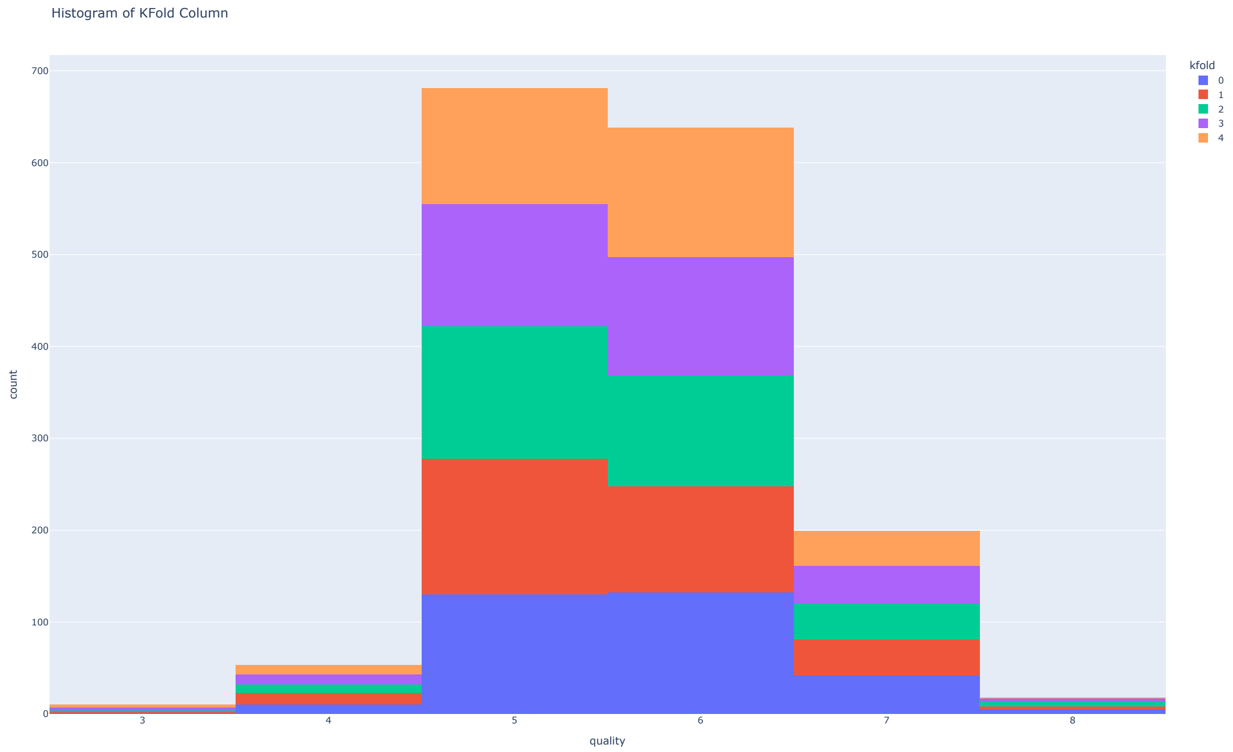
1.1.1 k-fold cross-validation

```
from sklearn.model_selection import KFold
from tabulate import tabulate
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/win
df = pd.read_csv(url, sep=";")

kf = KFold(n_splits=5, shuffle=True, random_state=42)

df['kfold'] = -1

# Split df into 5 folds
for fold, (train_index, test_index) in enumerate(kf.split(df)):
    df.loc[test_index, 'kfold'] = fold

df.to_csv('../data/wine-quality-data.csv')
print(df)
```

```
      fixed acidity  volatile acidity  citric acid  ...  alcohol  quality  kfold
0               7.4             0.700         0.00  ...      9.4        5      2
1               7.8             0.880         0.00  ...      9.8        5      4
2               7.8             0.760         0.04  ...      9.8        5      2
3              11.2             0.280         0.56  ...      9.8        6      2
4               7.4             0.700         0.00  ...      9.4        5      3
...             ...               ...          ...  ...      ...      ...    ...
1594            6.2             0.600         0.08  ...     10.5        5      2
1595            5.9             0.550         0.10  ...     11.2        6      4
1596            6.3             0.510         0.13  ...     11.0        6      3
1597            5.9             0.645         0.12  ...     10.2        5      1
1598            6.0             0.310         0.47  ...     11.0        6      3

[1599 rows x 13 columns]
```

Would be much better to show as a graph as well

```
import plotly.express as px
df = df.sort_values(by="kfold")
fig = px.histogram(df, x="quality", color="kfold", title="Histogram of KFold Colu
fig.show()
```

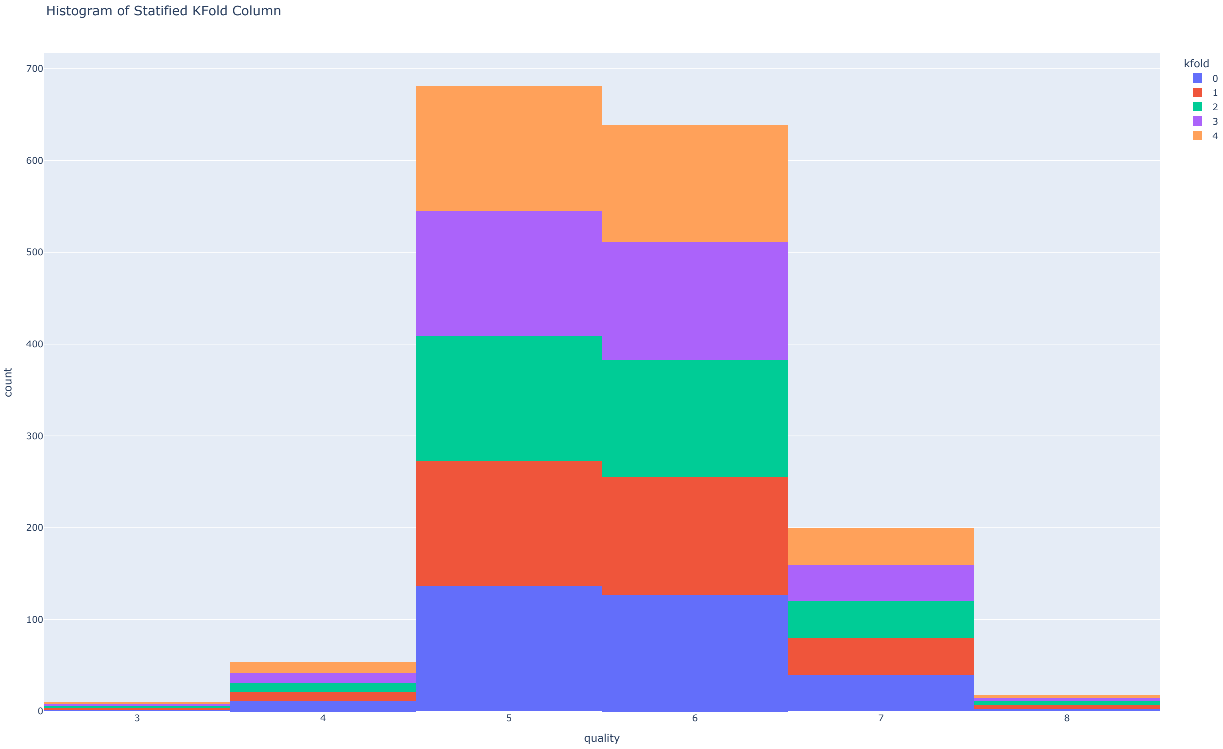


Loading [MathJax]/extensions/MathMenu.js

1.1.2 stratified k-fold cross-validation

```
from sklearn.model_selection import StratifiedKFold
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/win
df = pd.read_csv(url, sep=";")
# Create a StratifiedKFold object
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

df['kfold'] = -1

# Split df into 5 folds
for fold, (train_index, test_index) in enumerate(skf.split(df, df['quality'])):
    df.loc[test_index, 'kfold'] = fold
```

```
import plotly.express as px

df = df.sort_values(by="kfold")
fig = px.histogram(df, x="quality", color="kfold", title="Histogram of Statified
fig.show()
```



1.1.3 hold-out based validation

In some cases, stratified k-fold cross-validation is quite demanding for computing. For this kinds of case, just one fold is used for validation set. And it is recommended to split into higher number of k-fold if the number of samples is high such as 1 million.

1.2 Regression

- Mostly, simple k-fold cross-validation works for any regression problem
- If you see that the distribution of targets is not consistent, you can use stratified k-fold
- If you have a lot of samples($>$ 10k, $>$ 100k), then you don't need to care about the number of bins.