chapter-2-2

January 23, 2022

Run in Google Colab

1 Introduction to Machine Learning

1.1 Understanding the scikit-learn estimator API

Per completare questa parte vedere i notebooks di Flavio per il workshop

1.1.1 Transformers

The so-called ***transformer*** classes in scikit-learn, which are used for data transformation, have two essential methods: **fit** and **transform**. The fit method is used to learn the parameters from the training data, and the transform method uses those parameters to transform the data. Any data array that is to be transformed needs to have the same number of features as the data array that was used to fit the model.

1.1.2 Estimators

The ***estimators*** classes in scikit-learn, have an API that is conceptually very similar to the transformer class. Estimators have a **predict** method but can also have a transform method. We also used the fit method to learn the parameters of a model when we trained those estimators for classification. However, in supervised learning tasks, we additionally provide the class labels for fitting the model, which can then be used to make predictions about new, unlabeled data examples via the predict method, as illustrated in the following figure:

1.2 Validation and Testing

When data is used for forecasting there is a danger that the machine learning model will work very well for data, but will not generalize well to other data. An obvious point is that it is important that the data used in a machine learning model be representative of the situations to which the model is to be applied. It is also important to test a model out-of-sample, by this we mean that the model should be tested on data that is different from the sample data used to determine the parameters of the model.

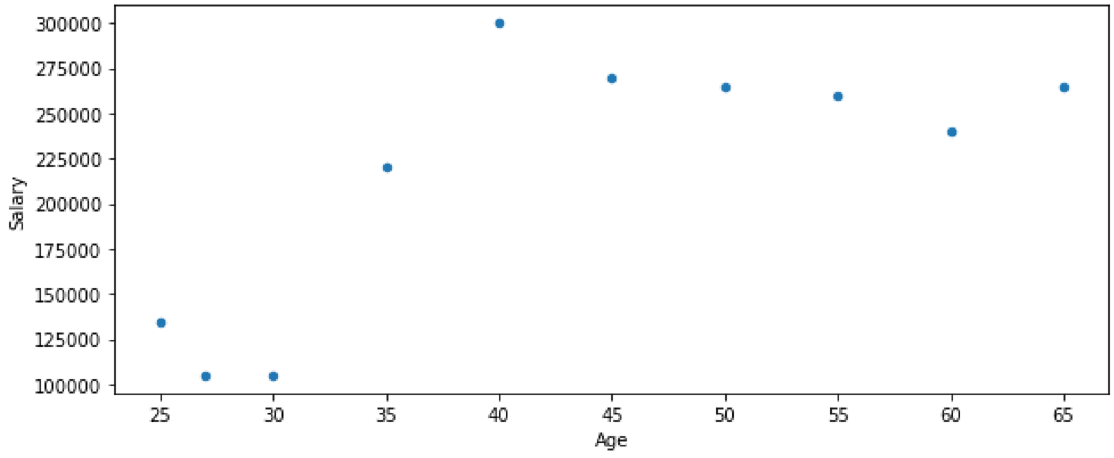
Data scientist refer to the sample data as the **training set** and the data used to determine the accuracy of the model as the **test set**, often a **validation set** is used as well as we explain later;

[1]:
```
if 'google.colab' in str(get_ipython()):
    from google.colab import files
    uploaded = files.upload()
    path = ''
else:
    path = './data/'
```

[2]:
```
# Load the Pandas libraries with alias 'pd'
import pandas as pd
# Read data from file 'salary_vs_age_1.csv'
# (in the same directory that your python process is based)
# Control delimiters, with read_table
df1 = pd.read_table(path + "salary_vs_age_1.csv", sep=";")
# Preview the first 5 lines of the loaded data
print(df1.head())
```

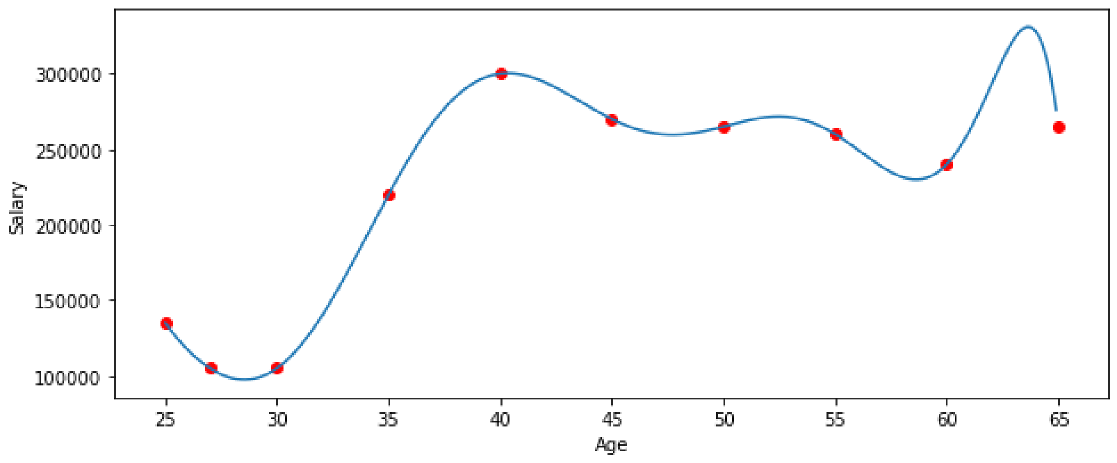
```
   Age  Salary
0   25  135000
1   27  105000
2   30  105000
3   35  220000
4   40  300000
```

[3]:
```
import matplotlib.pyplot as plt

plt.rcParams['figure.figsize'] = [10, 4]
ax=plt.gca()

df1.plot(x ='Age', y='Salary', kind = 'scatter', ax=ax)
plt.show()
```



polynomial fitting with pandas

[4]:
```
import numpy as np

x1 = df1['Age']
y1 = df1['Salary']

n = len(x1)

degree = 9

weights = np.polyfit(x1, y1, degree)
model   = np.poly1d(weights)

xx1 = np.arange(x1[0], x1[n-1], 0.1)
plt.plot(xx1, model(xx1))
plt.xlabel("Age")
plt.ylabel("Salary")
plt.scatter(x1,y1, color='red')
plt.show()
```



[5]:
```
y1  = np.array(y1)
yy1 = np.array(model(x1))

rmse = np.sqrt(np.sum((y1-yy1)**2)/(n-1))

print('Root Mean Square Error:')
print(rmse)
```

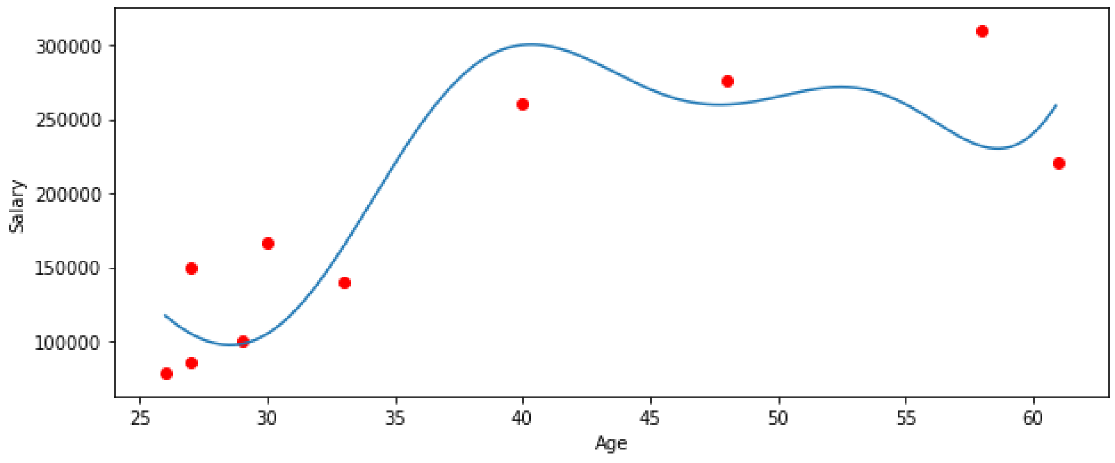
```
Root Mean Square Error:
0.0007184405154295555
```

```
[6]: if 'google.colab' in str(get_ipython()):
         from google.colab import files
         uploaded = files.upload()
         path = ''
     else:
         path = './data/'
```

```
[7]: df2 = pd.read_table(path + "salary_vs_age_2.csv", sep=";")
     x2 = df2['Age']
     y2 = df2['Salary']
     n  = len(x2)

     y2  = np.array(y2)
     yy2 = np.array(model(x2))

     rmse = np.sqrt(np.sum((y2-yy2)**2)/(n-1))

     print('Root Mean Square Error:')
     print(rmse)
```

```
Root Mean Square Error:
44726.74305949611
```

```
[8]: xx2 = np.arange(x2[0], x2[n-1], 0.1)
     plt.plot(xx2, model(xx2))
     plt.xlabel("Age")
     plt.ylabel("Salary")
     plt.scatter(x2,y2, color='red')
     plt.show()
```



- The root mean squared error (rmse) for the training data set is $12,902
- The rmse for the test data set is $38,794

We conclude that the model overfits the data. The complexity of the model should be increased only until out-of-sample tests indicate that it does not generalize well.

1.3 Bias and Variance

Suppose there is a relationship between an independent variable x and a dependent variable y:

$$y = f(x) + \epsilon \tag{1}$$

Where ϵ is an error term with mean zero and variance σ^2. The error term captures either genuine randomness in the data or noise due to measurement error.

Suppose we find a deterministic model for this relationship:

$$y = \hat{f}(x) \tag{2}$$

Now it comes a new data point x' not in the training set and we want to predict the corresponding y'. The error we will observe in our model at point x' is going to be

$$\hat{f}(x') - f(x') - \epsilon \tag{3}$$

There are two different sources of error in this equation. The first one is included in the factor ϵ, the second one, more interesting, is due to what is in our training set. A robust model should give us the same prediction whatever data we used for training out model. Let's look at the average error:

$$E\left[\hat{f}(x')\right] - f(x') \tag{4}$$

where the expectation is taken over random samples of training data (having the same distributio as the training data).

This is the definition of the **bias**

$$\text{Bias}\left[\hat{f}(x')\right] = E\left[\hat{f}(x')\right] - f(x') \tag{5}$$

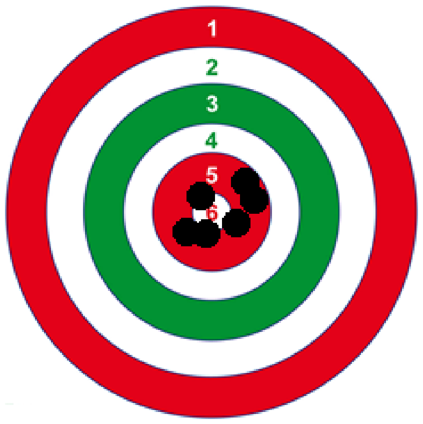
We can also look at the mean square error

$$E\left[\left(\hat{f}(x') - f(x') - \epsilon\right)^2\right] = \left[\text{Bias}\left(\hat{f}(x')\right)\right]^2 + \text{Var}\left[\hat{f}(x')\right] + \sigma^2 \tag{6}$$

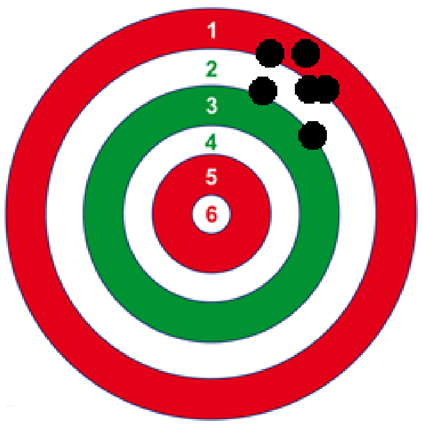
Where we remember that $\hat{f}(x')$ and ϵ are independent.

This show us that there are two important quantities, the **bias** and the **variance** that will affect our results and that we can control to some extent.

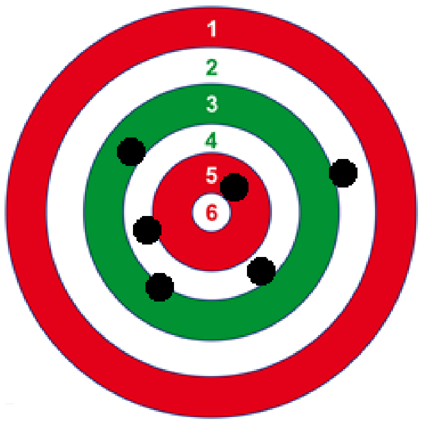
FIGURE 1.1 - A good model should have low bias and low variance



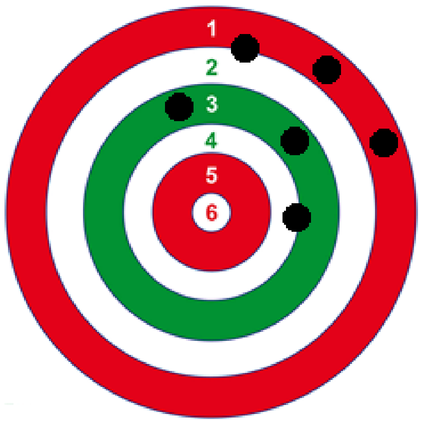
Low Bias and Low Variance



High Bias and Low Variance



Low Bias and High Variance

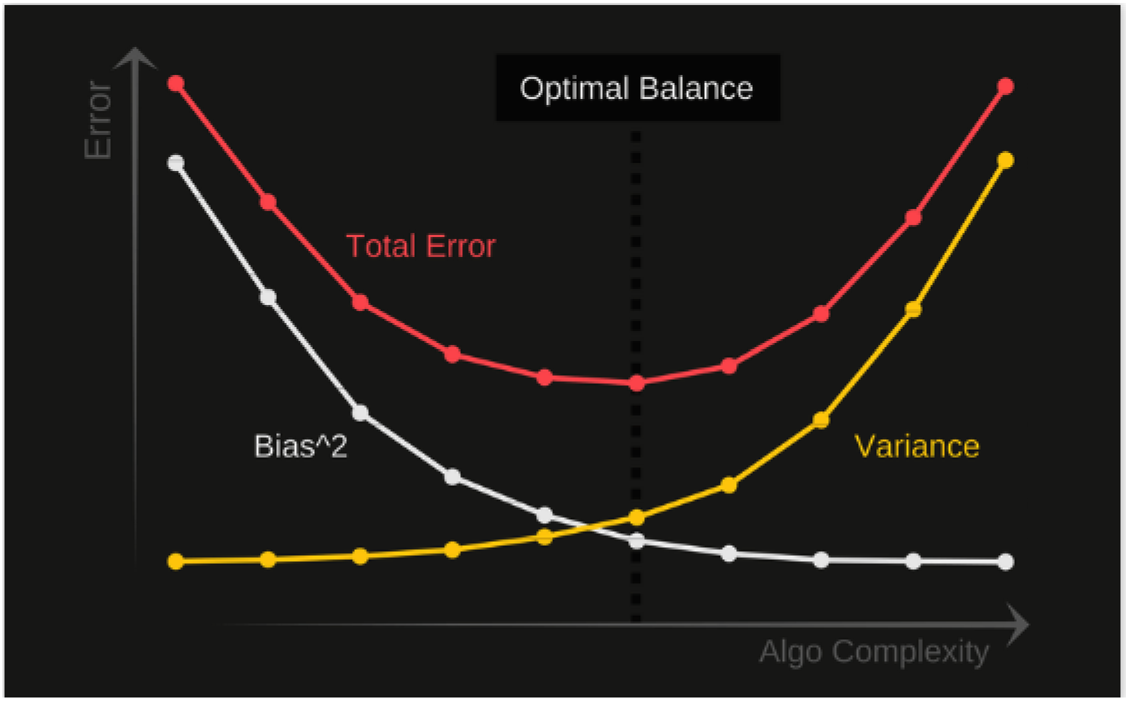


High Bias and High Variance

Bias is how far away the trained model is from the correct result on average. Where *on average* means over many goes at training the model using different data. And **Variance is a measure of the magnitude of that error**.

Unfortunately, we often find that there is a trade-off between bias and variance. As one is reduced, the other is increased. This is the matter of over- and under-fitting.

Overfitting is when we train our algorithm too well on training data, perhaps having too many parameters for fitting.



1.4 Partitioning a dataset into training and test datasets

```
[3]: import pandas as pd
```

```
[4]: if 'google.colab' in str(get_ipython()):
         from google.colab import files
         uploaded = files.upload()
         path = ''
     else:
         path = './data/'
```

```
[7]: # Both features and target have already been scaled: mean = 0; SD = 1
     # See Chapter 4-0 for a descrition of this dataset
     df = pd.read_csv(path + 'Houseprice_data_scaled.csv')
     df.head()
```

```
[7]:     LotArea  OverallQual  OverallCond  YearBuilt  YearRemodAdd  BsmtFinSF1  \
     0 -0.199572     0.652747    -0.512407   1.038851      0.875754    0.597837
     1 -0.072005    -0.072527     2.189741   0.136810     -0.432225    1.218528
     2  0.111026     0.652747    -0.512407   0.972033      0.827310    0.095808
     3 -0.077551     0.652747    -0.512407  -1.901135     -0.722887   -0.520319
     4  0.444919     1.378022    -0.512407   0.938624      0.730423    0.481458

       BsmtUnfSF  TotalBsmtSF  1stFlrSF  2ndFlrSF  ...   OLDTown     SWISU  \
     0 -0.937245    -0.482464 -0.808820  1.203988  ... -0.286942 -0.136621
     1 -0.635042     0.490326  0.276358 -0.789421  ... -0.286942 -0.136621
     2 -0.296754    -0.329118 -0.637758  1.231999  ... -0.286942 -0.136621
```

```
3  -0.057698    -0.722067 -0.528171  0.975236  ... -0.286942 -0.136621
4  -0.170461     0.209990 -0.036366  1.668495  ... -0.286942 -0.136621

   Sawyer   SawyerW   Somerst   StoneBr    Timber    Veenker  Bsmt Qual  \
0 -0.2253 -0.214192 -0.268378 -0.127929 -0.152629  -0.091644   0.584308
1 -0.2253 -0.214192 -0.268378 -0.127929 -0.152629  10.905682   0.584308
2 -0.2253 -0.214192 -0.268378 -0.127929 -0.152629  -0.091644   0.584308
3 -0.2253 -0.214192 -0.268378 -0.127929 -0.152629  -0.091644  -0.577852
4 -0.2253 -0.214192 -0.268378 -0.127929 -0.152629  -0.091644   0.584308

   Sale Price
0    0.358489
1    0.008849
2    0.552733
3   -0.528560
4    0.895898

[5 rows x 48 columns]
```

A convenient way to randomly partition this dataset into separate test and training datasets is to use the train_test_split function from scikit-learn's model_selection submodule:

[20]:
```
ncol = df.shape[1]

X = df.iloc[:, :ncol-1].values
y = df.iloc[:, ncol-1].values
```

[20]:
```
array([ 0.35848897,  0.00884914,  0.55273332, ..., -0.64510684,
       -0.63215721,  0.09302169])
```

[27]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =\
    train_test_split(X, y,
    test_size=0.3,
    random_state=0)
```

First, we assigned the NumPy array representation of the feature columns from 0 to ncol-1 to the variable X and we assigned the class labels from the last column to the variable y. Then, we used the train_test_split function to randomly split X and y into separate training and test datasets. By setting test_size=0.3, we assigned 30 percent of the wine examples to X_test and y_test, and the remaining 70 percent of the examples were assigned to X_train and y_train, respectively.

1.5 References

A. Géron, "*Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*", 2nd Edition. O'Reilly Media, 2019

S. Raschka and V. Mirjalili, "*Python Machine Learning: Machine Learning and Deep Learning with*

Python, scikit-learn, and TensorFlow 2", 3rd Edition. Packt Publishing Ltd, 2019.

S. Raschka, "*Model Evaluation, Model Selection and Algorithm Selection in Machine Learning*", downlodable here

Scikit-Learn web site

```
[ ]:
```