Machine Learning for Economics and Finance

Python Exercises

Ole Wilms

April 24, 2024

Contents

Important Instructions

- The purpose of these exercises is to get to know Python by solving some basic programming exercises
- In case you struggle with some problems, please post your questions on the OpenOlat Forum.
- Particularly difficult questions are marked by (D). Don't worry if you cannot solve these questions right away. Throughout the course, these programming concepts will become easier to understand.
- Sample solutions to the exercises will be provided next week. However, I strongly encourage all students to work on the exercises beforehand.

Task 1: Constructing a dataset

1. Create different kinds of vectors with 6 entries each:
    - vector a: a vector with only ones (hint: you can use the `np.repeat()` function)
    - vector b: a vector of integers that goes from 1 to 6 (hint: you can use the `np.arange()` function)
    - vector c: a vector where each entry is drawn from a normal distribution with mean 2 standard deviation 5.
    - vector d: a vector where each entry consists of one of the words in "*Machine Learning for Economics and Finance*".

[ ]:

2. Stack vector b into a matrix $M1$ of dimension 2 x 3 where you fill in by column. Stack the same vector into a matrix $M2$ of dimension 3 x 2 where you fill in by row.

[ ]:

3. Add the two matrices. You will obtain an error message. What's going wrong? Solve the problem using the transpose function `np.transpose()`.

[ ]:

4. Create a vector *train_sample* with 4 entries by randomly sampling 4 values from vector b without replacement (that is, you cannot draw the same number twice). For this you can use the function `np.random.choice()`. Run the code that creates the vector multiple times. Explain what's happening. Fix the issue by using the function `np.random.seed()`.

[ ]:

5. Put vectors a, b, c and d together in a dataframe called *df*.

[ ]:

6. Name the columns of *df* *'Ones'*, *'Seq'*, *'Normal'* and *'Coursename'* respectively (hint: you can use the function `pd.DataFrame()`). Provide a summary of the dataframe using the `describe()`function.

[ ]:

7. (D) Add a column called *'Int'* to the dataframe which checks whether column *'Normal'* is larger than 0. If that is the case *'Int'* should contain a *TRUE*, if that is not the case *'Int'* should contain a FALSE. Proceed as follows:
    - Create a new column named *'Int'* in the DataFrame, initializing all elements to True. Use a loop to iterate through each row of the DataFrame. For each row, check if the corresponding value in the *'Normal'* column is greater than 0. If it is, retain the *TRUE* value in the *'Int'* column; otherwise, replace it with *FALSE*."

[ ]:

8. (D) Can you think of an easier way to construct the column *'Int'* instead of the loop described above? If yes, add this column and call it *'Int2'*

```
[ ]:
```

9. (D) Now we use our vector *train_sample* to construct two distinct datasets from *df*. The numbers in *train_sample* refer to the rows of our dataframe *df* that we want to use for the first dataset while all other rows can be used for the second dataset. Construct a new dataframe called *df_train* that only contains the rows in *train_sample*. Note that you can simply use square brackets to extract rows from a dataframe. Make sure that you extract all columns but only the rows that are in *train_sample*. Your object *df_train* should have 4 rows and as many columns as *df*.

```
[ ]:
```

10. (D) Construct another dataframe called *df_test* which contains the other two rows of *df* that are not in *df_train*. Note that you can use `~df.index.isin()` to select all rows that are *NOT* in *train_sample*.

```
[ ]:
```

Task 2: Working data from the ***ISLR2*** library

1. Install and load the library *ISLP*.

```
[ ]:
```

2. Load the dataset *Auto* and save it into an object called *Auto*. Use the help function to obtain information about the variables in *Auto*.

```
[ ]:
```

3. Provide a summary of *Auto* using the `describe()` function. Do you think all the variables in *Auto* could be readily used for a linear regression model?

```
[ ]:
```

4. The goal of the following exercises is to understand the relation between the variable *'mpg'* and *'horsepower'*:
   - Provide a histogram of *'mpg'* using the function `hist()`. Hint: For creating plots and visualizations, the `matplotlib` package is a common choice.
   - Compute the pearson correlation between *'mpg'* and *'horsepower'*. For this, first select the two respective columns using `Auto["mpg","horsepower"]` and then use the function `corr()`. Is there a positive or negative relationship between the two variables?
   - Provide a plot with *'horsepower'* on the x-axis and *'mpg'* on the y-axis. Do you think a linear regression model is well suited to predict *'mpg'* using *'horsepower'* ?

```
[ ]:
```

Task 3: Working with external data

1. Load the dataset `'return_data.csv'` which contains historical returns of Apple (*'ret_apple'*), the index return of the *S&P500* which is a broad portfolio of stocks in the US (*'ret_index'*), as well as the return of a riskless investment in government bonds (*'rf'*). Make sure that you set the right working director when you try to load in the data. In the dataset, a number of 0.1 corresponds to a return of 10%.

[ ]:

2. To get to know the data, construct three plots each having the date on the x-axis and the respective return time series on the y-axis.

[ ]:

3. Compute the means and the standard deviations of the three time series and interpret the results.

[ ]:

4. What was the maximum loss in a single month when holding Apple stocks? What are the maximum losses for the *S&P500* and the risk-free rate? Interpret.

[ ]:

5. Compute the pearson correlation between *'ret_apple'* and *'ret_index'* using the function `cor()`. Interpret the result.

[ ]: