

Python Programming and Machine Learning for Economists – Exercises

Michael E. Rose, PhD

Course at ifo Institute, Jan/Feb 2022


Work only in the GitHub repo that you shared with me!

For each of the mandatory exercises below, create *one script* (no jupyter Notebook) that contains both code and answers (as comments) to the open questions in appropriate order.

Save the script in the main folder, properly named in correspondence to the name of the exercise. Avoid colons, dots and blanks in the filename. All scripts need to adhere to PEP8 and must be readable to someone that knows Python. Above all, they must execute without error on my machine.

Apart from the script, there should be one folder named "output" to store output such as figures and tables.

1 Optional Exercises for Pandas and Plotting

 Sketch for today: [French Taunting](#)

1. Tips

- Load seaborn's tips dataset using `seaborn.load_dataset("tips")` into a DataFrame.
- Convert the short weekday names to their long version (e.g., "Thursday" instead of "Thur") using `.replace()`.
- Create a scatterplot of "tip" vs. "total_bill" colored by "day" and facets (either by row or by column) by variable "sex". Label the axis so that the unit becomes apparent. Save the figure as `./output/tips.pdf`

2. Occupations

- Import the pipe-separated dataset from <https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user> into a DataFrame. The data is on occupations and demographic information.
- Print the last 10 entries and then the first 25 entries.
- What is the type of each column?
- Count how often each occupation is present! Store the information in a new object.
- How many different occupations are there in the dataset? What is the most frequent occupation? (🔗 Try to use a programmatic solution for these questions using the new object!)
- Sort the new object by index. Then create a figure and an axis. Plot a histogram for occupations on that axis (🔗 Do not use `.hist()`). Add an appropriate label to the x-axis. How does the figure look like if you don't sort by index beforehand? Save the figure as `./output/occupations.pdf`

3. Iris


- Read the Iris dataset from <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data> as DataFrame. The data is on measures of flowers and the values are on "sepal_length (in cm)", "sepal_width (in cm)", "petal_length (in cm)", "petal_width (in cm)" and "class". Since the data doesn't provide the column names, add the column names after reading in, or alternatively provide while reading in.
- Set the values of the rows 10 to 29 of the column 'petal_length (in cm)' to missing.
- Replace missing values with 1.0.
- Save the data as comma-separated file named `./output/iris.csv` without index.
- Visualize the distribution of all of the continuous variables by "class" with a catplot of your choice. Optionally, try to tilt/rotate the labels using `.set_xticklabels()`, which accepts a rotation parameter). Save the figure as `./output/iris.pdf`.

4. Memory

- Load the comma-separated data from <https://query.data.world/s/wsjbxdqhw6z6izgdxiyv5p2lfqh7gx> into a DataFrame (large file!).

- (b) Inspect the DataFrame using `.info()` and with `.info(memory_usage="deep")`. What is the difference between the two calls? How much space does the DataFrame require in memory?
- (c) Create a copy of the object with only columns of type object by using `.select_dtypes(include=['object'])`.
- (d) Look at the summary of this object new (using `.describe()`). Which columns have very few unique values compared to the number of observations?
- (e) Does it make sense to convert a column of type object to type category if more than 50% of the observations contain unique values? Why/Why not?
- (f) Convert all columns of type object of the original dataset to type category where you deem this appropriate.
- (g) What is the final size in memory?
- (h) Could above routine have sped up somewhere? (🔗 Look at the documentation for `.read_csv()`).
- (i) Subset the DataFrame to all all the numeric columns only. then store the file twice in folder `./output/`, namely as CSV file using `.to_csv()` and secondly as feather file using `.to_feather()` (Do not attempt to check in & push these files, as this they would be blocked by GitHub!). By how much do the file sizes differ approximately, and why is that? (🔗 Check out the [corresponding documentation](#))

2 Exercises for Unsupervised Machine Learning

 Sketch for today: [Village Witch](#)

1. Principal Component Analysis

- (a) Read the textfile `./data/olympics.csv` (in your git repository) into a DataFrame using the first column as index. The data lists the individual performances of 33 male athletes during the [Decathlon of the 1988 Olympic summer games](#) (100m sprint, running long, (broad) jump, shot put, high jump, 400m run, 110m hurdles, discus throw, pole vault, javelin throw, 1.500m run). Print summary statistics for each of the variables and decide (and act accordingly): Does it make sense to drop variable "score" before proceeding?
- (b) Scale the data such that all variables have unit variance. Which pandas DataFrame method can you use to assert that all variables have unit variance?
- (c) Fit a plain vanilla PCA model. Store the components in a DataFrame to display the loadings of each variable. Which variables load most prominently on the first component? Which ones on the second? Which ones on the third? How would you thus interpret those components?
- (d) How many components do you need to explain at least 90% of the data? (🔗 Use `numpy.cumsum()` for this.)


2. Clustering

- (a) Load the iris dataset using `sklearn.datasets.load_iris()`. The data is on classifying flowers.
- (b) Scale the data such that each variable has unit variance.
- (c) Assume there are three clusters. Fit a K-Means model, an Agglomerative Model and a DBSCAN model (with `min_sample` equal to 2 and ϵ equal to 1) with Euclidean distance. Store only the cluster assignments in a new DataFrame.
- (d) Compute the silhouette scores using `sklearn.metrics.silhouette_score()` for each cluster algorithm from c). Why do you have to treat noise assignments from DBSCAN differently? Which model has the highest Silhouette score?
- (e) Add variables "sepal width" and "petal length" including the corresponding column names to the DataFrame that contains the cluster assignments. (Beware of the dimensionality!)
- (f) Rename noise assignments to "Noise".
- (g) Plot a three-fold scatter plot using "sepal width" as x-variable and "petal length" as y-variable, with dots colored by the cluster assignment and facets by cluster algorithm. (🔗 Melt the DataFrame with above variables as ID variables.) Save the plot as `./output/cluster_petal.pdf`. Does the noise assignment make sense intuitively?

3 Exercises for Supervised Machine Learning

 Sketch for today: [Machine that goes ping](#)


1. Feature Engineering

- (a) Load the Boston House Price dataset using `sklearn.datasets.load_boston()` as per usual.
- (b) Extract polynomial features (without bias!) and interactions up to a degree of 2 using `PolynomialFeatures()`. How many features do you end up with?
- (c) Create a pandas DataFrame using the polynomials. Use the originally provided feature names to generate names for the polynomials ( `.get_feature_names()` accepts a parameter) and use them as column names. Also add the dependent variable to the DataFrame and name the column "y". Finally save the DataFrame as comma-separated textfile named `./output/polynomials.csv`.

2. Regularization

- (a) Read the data from exercise "Feature Engineering" (`./output/polynomials.csv`) into a DataFrame.
- (b) Use column "y" as target variable and all other columns as predicting variables (named X in class) and split them as usual.
- (c) Learn an ordinary OLS model, a Ridge model and a Lasso model using the provided data with penalty parameter equal to 0.3. Using the the R^2 scores, which model yields the best prediction?
- (d) Create a DataFrame containing the learned coefficients of all models and the feature names as index. In how many rows are the Lasso coefficients equal to 0 while the Ridge coefficients are not?
- (e) Using `matplotlib.pyplot`, create a horizontal bar plot of dimension 10x30 showing the coefficient sizes. Save the figure as `./output/polynomials.pdf`.

3. Neural Network Regression

- (a) Load the diabetes dataset using `sklearn.datasets.load_diabetes()`. The data is on health and diabetes of 442 patients. Split the data as usual.
- (b) Learn a Neural Network Regressor with identity-activation after Standard-Scaling with in total nine parameter combinations of your choice. Use the best solver for weight optimization for this dataset according to the documentation! To keep computational burden low you may use a 3-fold Cross-validation and at most 1,000 iterations.
- (c) What are your best parameters? How well do they perform in the training set? How well does your model generalize?
- (d) Plot a heatmap for the first coefficients matrix of the best model ( Access the model via `.best_estimator_`. One of its attributes is `_final_estimator`, which behaves like a normal model object.). Be sure to label the correct axis with the feature names. Save the heatmap as `./output/nn_diabetes_importances.pdf`.

4. Neural Networks Classification

- (a) Load the breast cancer dataset using `sklearn.datasets.load_breast_cancer()`. As usual, split the data into test and training set.
- (b) Read up about the Area Under the Curve of the Receiver Operating Characteristic Curve, the so-called ROC-AUC-metric, e.g. at towardsdatascience.com.
- (c) Learn a a Neural Network Classifier after MinMax-Scaling with in total four parameter combinations (and 1000 iterations) of your choice using 5-fold Cross-Validation. To keep computation burden low, stop after 1,000 iterations and use the best solver for this dataset. Using the ROC-AuC-score metric to pick the best model, what are the best parameter combinations, which is its ROC-AuC-score, and how well does it generalize in terms of the ROC-AuC-score?
- (d) Plot the confusion matrix as a heatmap for the best model and save it as `./output/nn_breast_confusion.pdf`.

4 Exercises for Natural Language Processing

 Sketch for today: [Argument Clinic](#)

1. Regular Expressions

- (a) Solve the first eight regex practice problems on https://regexone.com/problem/matching_html (hover over "Interactive Tutorial" to see them).

2. Speeches I

- (a) Read up about `pathlib.Path().glob()`. Use it to read the text files in `./data/speeches` into a corpus (i.e. a list of strings), but only those that start with "R0". The files represent a non-random selection of speeches of central bankers, which have already been stripped off meta information. The files are encoded in UTF8, except for two broken ones. Use a try-except-statement to skip reading them and print the filename instead.
- (b) Vectorize the speeches using tfidf using 1-grams, 2-grams and 3-grams while removing English stopwords and proper tokenization (i.e., you create a tfidf matrix).
- (c) Pickle the resulting sparse matrix using `pickle.dump()` as `./output/speech_matrix.pk`. Save the terms as well as `./output/terms.csv`

3. Speeches II

- (a) Read the count-matrix from exercise "Speeches I" (`./output/speech_matrix.pk`) using `pickle.load()`.
- (b) Using the matrix, create a dendrogram of hierarchical clustering using the cosine distance and the complete linkage method. Remove the x-ticks from the plot. Optionally, set the color threshold such that three clusters are shown.
- (c) Save the dendrogram as `./output/speeches_dendrogram.pdf`.

4. Job Ads

- (a) Read the text file `./data/Stellenanzeigen.txt` and parse the lines such that you obtain a DataFrame with three columns: "Newspaper", "Date", "Job Ad". Make sure to set the Date column as Datetime type.
- (b) Create a new column counting the number of words per job ad. Plot the average job ad length by decade.
- (c) Create a second DataFrame that aggregates the job ads by decade, keeping just this information. Which are the most often used terms used by decade after appropriate cleaning?