Hands-On Machine Learning with Scikit-Learn, Keras, and

TensorFlow

After exploring the machine learning landscape (chapter I), chapter 2 dives into the practical side of the machine learning project essentially describing the end-to-end process. This report aims to deliver the full experience while maintaining simplicity.

Most Projects fall into these main steps sequentially:

- Look at the big picture.
- Get the data.
- Explore and visualize the data to gain insights.
- Prepare the data for machine learning algorithms.
- Select a model and train it.
- Fine-tune your model.
- Present your solution.
- Launch, monitor, and maintain your system

At first, it is worth noting that when practicing machine learning, a practitioner is advised to search for real data to understand the practical machine learning project management process. **"Kaggle.com"** and **"UC Irvine Machine Learning Repository"** are famous sites to get started but there are certainly more to visit.

An End-to-End Machine Learning Project:

1. Looking at the big picture

    - You typically need to understand the **main objective, Frame the problem, and understand what the current solution looks like.** Having acquired all the information required, **system designing** takes place. You would look at the training supervision (e.g. supervised, unsupervised ..etc), the task's output for determining classification or regression, and the type of learning technique (Batch or Online).

- You would need to select a **fitting performance measurement** for model evaluation (e.g. regression tasks can evaluated through mean squared error, root mean squared error, and other types of evaluation metrics)

- You would need to check for assumptions before going through with the actual project to eliminate any major issues later on. A popular case would be checking the assumption of the task's output, the difference between a price prediction (numeric, hence regression) and a price range prediction (categorical, therefore classification) would require a complete system change from regression models to classification models, after months of work, if the assumptions have not been verified correctly!

2. Getting the Data

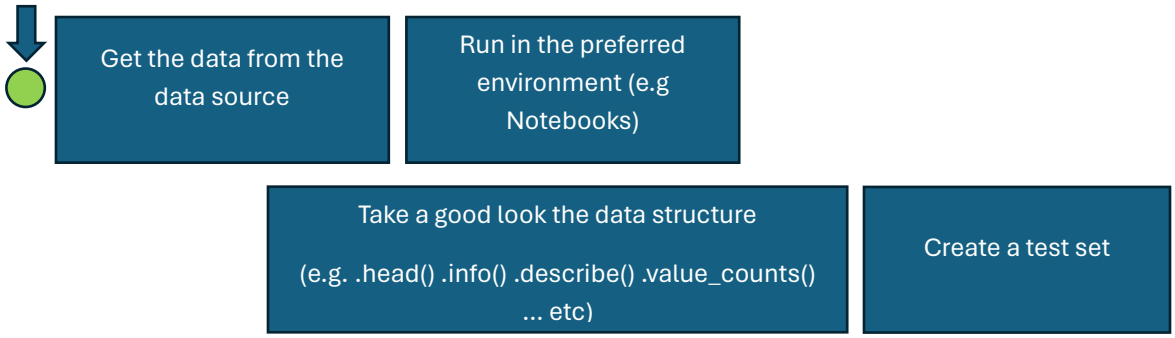

Figure 1

- To elaborate on Figure 1, you would pull the data, run it in any environment of choice, and then inspect the structure of the data using many methods. There are famous methods, including head(n) for examining a portion of any size of the dataset, Info() method explores the basic information on the dataset pulled (columns, non-null counts, data types, ... etc), describe() "describes" the summary statistics of numeric variables/features, value_counts() counts the unique instances in the variable and returns their frequencies, plotting a simple histogram is another possible way to understand numeric values in terms of spread, skewness, and to verify the need for scaling. Lastly, create the test set (and the train set) either with "train_test_split" or using the "StratifiedShuffleSplit", the latter preserves the proportions of variables in the test data to ensure full representation.

3. **Exploring and visualizing the data to gain insights**

- A glance at the the data has been secured so far, the goal now is to go more in-depth.

- Firstly, copying the training data into another data frame is a safer option in case reverting is needed after experimentation.

- This part typically gains insights from, visualizing data of interest (such as geographical data) and looking at the correlations between attributes either numerically (Pearson correlation using .corr( )) or visually using the scatter_matrix. Additionally, you may want to attempt different attribute combinations in case newer attributes correlate more with the target label.

4. **Preparing the data for machine learning algorithms**

- At this stage the data is ready to be cleaned as part of the preparation. Writing functions can be more beneficial encompassing transformation reproduction, library building/reuse, and generally make it easier to try out various transformations and see which combinations work best.

  Typically you would follow the figure below:

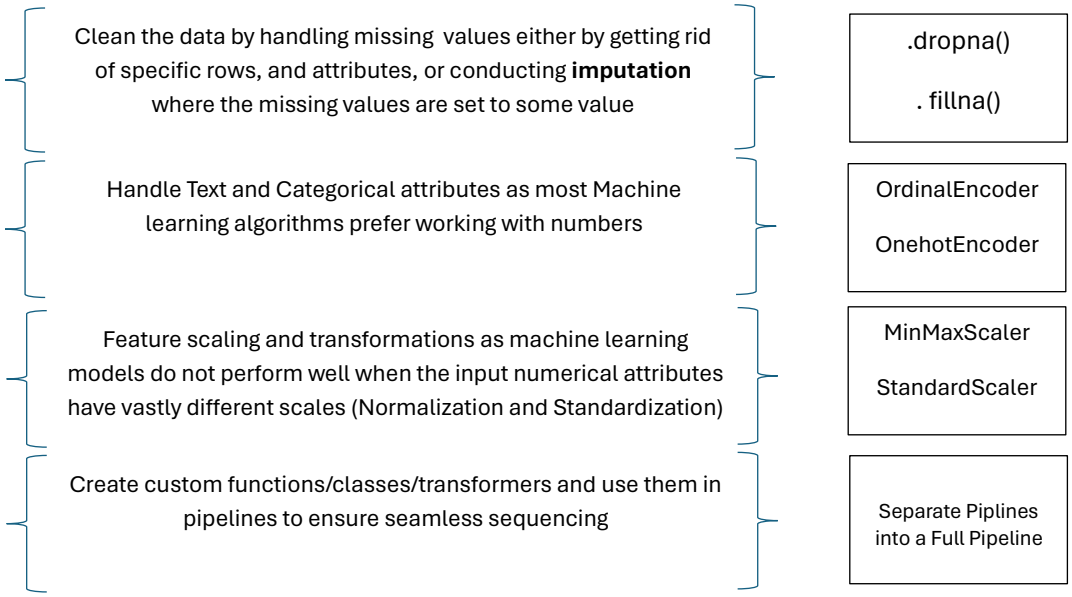

Figure 2

5. **Selecting and Training a Model**

- All in all, the problem has been framed, the data is in our hands and has been explored, a training set and a testing set have been sampled, and a pipeline has been written for automatic data preprocessing in preparation for the machine learning model.

- The stage is very simple you will choose a model, train the model, evaluate it through the chosen evaluation metric to decide if the model is sufficient, or try out different models in comparison.

- LinearRegression from sklearn.linear_model and DecisionTreeRegressor from sklearn.tree are examples of models provided by the library

- The mean squared error as a basic example can acquired through mean_squared_error from sklearn.metrics

- Evaluation instead can be done by splitting the already existing training set into a training set and a validation set, using train_test_split(), as mentioned previously in Chapter 1, thus training the model and evaluating it against the validation set.

- A better alternative is done using **Cross-Validation.** It is done using Scikit-Learn's k_fold cross-validation feature (cross_val_score from sklearn.model_selection). The feature randomly splits the training set into nonoverlapping subsets called **folds**, and then it trains and evaluates the decision tree model (**n**) times, picking a different fold for evaluation every time and using the other (**n-1**) folds for training. The result is an array containing the evaluation scores for an average feel on the performance of the model.

6. Fine-Tuning your Model

    - After shortlisting different models, you can tweak hyperparameters manually to fine-tune them, but using the **Grid Search (GridSearchCV from Scikit-Learn)** the process is automated for you as it searches for the best parameter value using cross-validation. However, when a lot of combinations exist using the (RandomizedSearchCV) or (HalvingRandomSearchCV) for efficient usage of the computational resources.

    - Combining models is another way of fine-tuning a system this group or **ensemble,** hence the name "Ensemble methods", will often perform better than the best individual model. Analyzing the best models and their corresponding errors can be great for insight gathering (e.g. features selection after inspection), and finally evaluating the system on the test set that should perform sufficiently well.

- The project prelaunch phase contains:

    - A solution presentation containing learning outcomes, what worked and what didn't, the assumptions that were made, and the system's limitations, with clear visuals and statements.

    - A documentation for the whole project.

    - It might still be a good idea to launch the project to free up time for other tasks as well.

7. Launching, Monitoring, and Maintaining Your System

- **Launch and Deployment**

    - After obtaining the approval, the solution is prepared by polishing the code, writing documentation and testing, and deploying the model into the production environment.

    - **Joblib (joblib)** is a library that makes model transfer and loading into production environments easy. It is also advisable to save all models, cross-validation scores, and predictions for comparison and selection purposes**.**

    - The models can be integrated directly into a web app or deployed as dedicated web services, accessible via a REST API, or even using cloud services (e.g. on Google's Vertex AI) after saving the model using joblib and uploading it to the cloud.

- **Monitoring the code is a must to maintain the system**

    - Domain knowledge metrics and human analysis are very important regularly to keep the system in check.

    - Evaluation of input data quality could affect the system's performance down the line if not updated and monitored steadily, thus the need for continuous quality checking.

    - Keeping backups of models and datasets ensures reverting to previous versions if new updates perform poorly or data corruption occurs, in addition to the ability to make comparisons between new and previous models.