Tyler Wulff

Hands-On With Data

There are many places to find datasets with real-word data useful to experiment. Some of these repos are UC Irvine, Kaggle, and AWS. When dealing with data, pipelines are commonly used. Pipelines are sequences of data processing components and can be very useful when dealing with data. There are components in the pipeline that run asynchronously, pulling in data and processing it. This allows for broken components to not ruin everything and for different people to focus on different components. On the downside, broken components may be difficult to detect if they aren't being monitored at a level it needs to be. It is important to first come up with an objective or problem when starting off building a model. It can also be useful to look at what the current solutions look like for the same or similar problem. This allows you to see the downfalls of previous solutions and how yours could be better. Then, framing the problem as a type of learning or what type of task it is very important so you know where to really get started. Looking at the type of data and what you are trying to accomplish will help when deciding these factors. After making these decisions, you must select a performance measure. One that is commonly used for regressions problems is the Root Mean Square Error. This measure gives high weight to large errors and can show how much error is being made in the predictions. There is also the Mean Absolute Error, which is more useful in situations where there are many outliers. Computing the RMSE corresponds to the Euclidian norm, while the MAE corresponds to the Manhattan norm. Since the higher the norm index, the larger values are focused on more, so the RMSE is more likely to be affected by outliers. When outliers aren't greatly present, the RMSE is the way to go. After going through all of this, it is important to list the assumptions that have been made so far in order to prevent future errors.

It is important to download the data and then download python properly with different modules that will be used throughout the project (NumPy, Pandas, Matplotlib, pip, etc.) along with making a clean work directory. Creating an isolated environment can also be useful to keep everything together. When starting a Jupyter notebook, a list of cells will be available allowing for code to be run. Creating a function to decompress/pull in your data can be very useful if there is regularly changing data. Many functions can be used on a dataframe to help you look at the data structure that you have. These consist of head() (look at top 5 rows), info() (description of data), value_counts() (categories), and describe() (gives a summary of numerical attributes). Matplotlib has a hist() method used to create histograms, but you need to use the command *%matplotlib inline* to specify the backend. Looking at the histogram can give you a lot of information about the data and gives you a preliminary look at patterns.

After getting the data, you need to set aside a test set, maybe around 20% of the dataset, avoiding looking at the test set. You need to make sure the test set is consistent across many runs, so using a unique identifier can be helpful. Scikit-Learn also has a train_test_split function that can also be used. Scikit-Learn also has a StratifiedShuffleSplit that can be used to split the dataset so that there is not significant bias, comparing to the dataset as a whole. A random test set may present significant bias, especially in smaller datasets.

Scatter plots can also be useful in visualizing data. The plot() function can be used with an alpha parameter to look at where the high density areas consist of in a scatter plot. You can also use the cmap option to color the values to give a better visualization of the data. When dealing with smaller datasets, the standard correlation coefficient can be calculated using the corr() method. This is a value between -1 and 1 showing either positive or negative and weak or strong correlation. 1 means strong and positive, -1 means strong and negative correlation. This method only looks at linear correlations though, ignoring nonlinear relationships. The scatter_matrix function will plot the numerical attributes against one another, another way to look at correlation.

Functions can be written to prepare the data for algorithms. Allowing for reproduction, future use and to try different things. It is important to first clean the data by dealing with missing values (dropna(), drop(), fillna(), or Imputer). Scilkit-Learn's API focusses on consistency, inspection, nonproliferation of classes, composition and sensible defaults, making it clean in useful. It may also be useful to convert test attribute to numbers, making it easier to work with. A OneHotEncoder can be used to convert integer categorical values int one-hot vectors, which can be useful when dealing with converting text data to numerical data to Boolean data. LabelBinarizer may also be useful with sparse_output=true to do these transformations in one. Custom transformers can be created by creating a class and implementing a fit(), transorm(), and fit_transform(). Feature scaling is available by using min-max scaling (normalization) or standardization (always have a zero mean). The Pipeline class can be used with sequences of transformations. Pipeline ahs fit(), fit_transform() to call on transformers.

Selecting and training a model is the next step. Sklearn has linear regression, RandomForestRegressor, and DecisionTreeRegressor available to train a model on the training set. Checking the error can show you good information, but you don't want to test on the test set until you are confident. Scikit-Learn's cross-validation allows to split the training set into subsets and then runs the model on each of the subsets. The cross-validation wants a utility function instead of a cost function. The RandomForestRegressor train many Decision Trees and average them out. Ensemble learning consists of building models on top of each other. After working with a model, you should save it so you can come back and look at what you havStarie so far.

Scikit-Learn has a GridSearchCV that allows you to look at different hyperparameters to work with. Consecutive powers of 10 is a good place to start when working with hyperparameters. When the search space for the hyperparamer is large, RandomizedSearchCV may be more useful. After the models have been looked at in depth, evaluating the final model on the test set is the best way to test the your model.

In regards to my semester project, I can use a lot of the information from this chapter. We have already started to frame our problem and look at the performance measures we will use. I believe that we will use RMSE to measure the performance, because I don't think that outliers will be a huge issue for us. We have also already started setting up a workspace in google collab instead of jupyter notebook, but it is a similar setup. The data we have so far has not been downloading, so I think that using a similar approach to how the chapter showed the downloading of the data could be very useful. We are already working on mounting the data in google collab. The visualization tips will be very useful for our project. I want to try to get a look at both player and team data before really digging deep into the model. I think that scatter plots comparing different player attributes will give a very good look into what the data looks like and what we will be dealing with. It will help us in regards to how the data has been distributed and what we need to do to the data. Our test data will be split at about what the chapter recommends (~20%), but the Scikit_Learn functions to split the data will be useful (StratifiedShuffleSplit). Looking at the correlation between different player data and team data will be very interesting. It will help us look at what may lead to more points scored or less points allowed. Looking at all of the data we have and using the scatter_matrix function could really help us when looking at the different attributes and how they correlate to one another. When it comes to players and injuries, cleaning the data will be very important. There will more than likely be missing values in many datasets, we need to make sure that we clean the data properly to avoid these missing values causing any issues. When it comes to handling text attributes, I don’t think we are going to have to worry too much about it, since most of the data we are dealing with deal with numerical values. We will use ensemble learning when working with our model to try and use multiple models to develop a better algorithm. We are going to have to look at the type of models more in depth in order to see which will be the best. The information on how to get the models started and tested will be very useful when testing and training our data to make sure we have the best model.