Internship Report On Data Analytics at Ernst & Young, Bangalore

Palak Jain
Computer Science, University of Southampton
July-August 2019

Contents

1.1 Summary

During my internship period of 2 months, I worked with a team that specialises in Data Analytics. Python is a core language used in the department and initially, I had no experience with the programming language. As a result, I spent the first 2 weeks learning and practicing basic python techniques. I progressed onto using python to modify dataframes from external files. In week 4, I worked on utilising machine learning algorithms to identify anomalies in the datasets. As I got even more comfortable with machine learning algorithms, I worked on using convolutional neural networks to identify objects from images and NLP techniques to develop a reviews classification and a simple spell checker program.

1.2 Description of Internship

1.2.1 Data Extraction and Manipulation

Initially, I started off by looking at python fundamentals: lists, arrays, tuple, sets, functions, dictionaries, and practised using them in the Jupyter Notebook. After getting comfortable with the basics, I went into learning about handling datasets through the pandas and numpy libraries and representing results using seaborn and matplotlib. I worked on a number of tasks in which I had to read the data from external files, extract specific information from them or modify them. In all the tasks, I used various features such as count, sum, maximum, minimum, grouping, joins etc. Before writing the code for any one of these tasks, I made sure to first plan it out on paper as it would help me clarify my logic.

For my first task, I was given a table of constraints and my goal was to filter the data from dataframes to match the constraints and deliver the results in a specific output format. My approach was to use a for loop through the entire file and then use if/else or switch statements to filter out the unnecessary data. However, I realised that a more efficient way of approaching this problem would be to define functions with parameters. In this way, the constraints (numerical values or strings) could be passed onto the function directly and the function would apply the constraints the same way every time. This approach would avoid code duplication and hardcoded values. Another approach that was highlighted to me in my presentation was that I could use a dictionary mapping so that for each property, the constraint on it can be mapped to directly.

In addition to applying filters to extract data, I was given other tasks such as creating summaries and joining dataframes. These helped strengthen my understanding of what I learnt on my own and which python libraries to use when working with the datasets. Most importantly, figuring out the logic on my own has made me a more confident programmer.

1.2.2 Anomaly detection

I used a dataset from one of the previous tasks to train my model and detect anomalies. First, I split the dataset into training and test data, such that 80% of it was training data. Then, I dropped columns that didn't have any influence on the classification of the dataset as an anomaly. As the number of null values in the dataset was not significant, I dropped any null values in any of the columns. Since the remaining columns of the dataframe only had qualitative features, I assigned each possible qualitative value to a numerical value through mapping. In other words, if there were six technologies present in the technology column, I assigned the technologies each, a number from 1-6. Numerical values have to be assigned to qualitative values in order to apply machine learning models to them. I then applied the training set to a machine learning model (from sklearn/scikit-learn library) and predicted the anomalies in the test data using the applied model. Finally, I created a comparison chart of different machine learning models and their accuracy scores.

The accuracy score reflects the percentage of test data that is correctly identified as an anomaly. The highest accuracy (68.2%) was reached by the Random Forest Classifier and Decision tree algorithms, whereas the perceptron algorithm had the lowest score of 44.9%. Even though both are classification algorithms,the perceptron is much more complex in nature as it is a neural network, and as a result, it requires a much greater data size. That is why I suppose, it gave the least accurate score.

1.2.3 Object classification from images

After completing the Anomaly Detection task, I familiarised myself with how several machine learning algorithms work. As a result, I felt comfortable working in one of the areas which the team specialises in - developing object identification programs. I found a clothing images dataset in Kaggle and worked on building a convolutional neural network so that I could train my program to identify the clothing upon loading a test image.

The process I followed is as follows: Firstly, I prepared the data. This included resizing the images to 28*28 pixels and splitting the training data and test data. Then, I defined a sequential model as I intended to apply multiple layers of kernels. I applied the Convolution2D layer, defining the number of filters, kernel size, activation function and the input shape. I then applied the Max Pooling function, where I defined the pool size as 2 by 2 and finally the Dropout function. The Maxpool function takes the greatest number of pixels in the defined pool size and significantly reduces space. The Dropout function reduces overfitting. Similarly, I applied another layer of convolution with more filters to increase the accuracy of the model. I used the flatten function to integrate the two convolutional layers and convert it into a single dimension vector. I used the softmax function to output the accuracy levels in all categories of clothing, and compiled the model. Finally, I fit the model with the processed data, defined the

batch size and the epochs- number of steps. The resulting accuracy rate of testing the test data was 88.6 %.

Similar to detecting clothing from ots images, I worked on detecting handwritten digits. The approach I took was similar but I looked for techniques to make my model even stronger. As a result, I used data augmentation which included tilting images by small angle, zooming in and zooming out, and whitening the edges of the images. All these methods increase the size of the existing data and make the model stronger. In addition, I set a learning rate annealer which decreased the learning rate factor by 0.5 every time. Decreasing the learning rate overtime reduces the chance of overfitting. Both these techniques raised the accuracy of the model from 96 to 98% accuracy.

1.2.4 NLP techniques

I spent the last two weeks understanding Natural Language Processing techniques. NLP is a very broad subject which is based on programming computers to process and understand large amounts of natural language data. I worked on two small projects using NLP techniques- Reviews classifier and a Spellchecker. I found a restaurant's reviews dataset in Kaggle containing 1500 reviews and classification of which were positive reviews and which were not.

Following the basic steps, I first loaded the reviews dataset, removed all the punctuation between the words, converted them to lower case, and removed the stop words. Stop words are words like 'the' , 'an' etc. - they are necessary for the completion of a sentence but on their own, they have no significance. Then, I used the PorterStemmer() object to only keep root/stem words- this makes it easier for the computer to recognize the words. Lastly, I split all the reviews and put them in an empty array. Now,the data was ready to be split into train and test data. Lastly, I fit in the Random Forest Classifier model and printed the confusion matrix to get the accuracy of the model. Accuracy was calculated by the formula shown below (TP = True positives, FN = false negatives). The model gave a 70% accuracy rate.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

In order to build my spell checker, I initially looked into existing libraries and functions that I could use. Upon research I found out that I could use difflib.get_close_matches(word, dictionary) and the SpellChecker().correction(word) functions. The first compares the input word(incorrect word) with similar words in a dictionary file. The dictionary file I used was linuxwords.txt, which is freely available online. It is an English word dictionary. The latter function is an adaptation of Norwig's algorithm which tries to find the best correction by using

the Levenstein distance. First, I just used the SpellChecker() function which simply outputs the result of Norwig's algorithm. However, to understand the inner working of the function I implemented it on my own, by writing code to calculate the Levenshtein distance. This edit distance calculates how far the input word is from the dictionary. A word can be misspelt in 4 possible ways: through deletes, transposes, inserts and deletes. A levenshtein distance of 1 would mean that any one of these 4 ways would be applied once to reach the correct spelling of the word. For example, let's say a user inputs appel instead of the word apple(levenshtein distance = 1). There is a transpose error as position of l and e are switched. Similar to this, the algorithm computes all possible words with a distance of 1 and 2 and matches them with a set of known words. The words that appear in this set are the possible candidates. The closest known word from the list of candidates is then outputted.

1.3 Reflection

Overall this internship has been very beneficial to me. I was guided throughout on how to deepen my understanding and the tasks I was given complemented well with what I was learning. The internship helped me understand the significance of Data Analytics and being able to work with some datasets used by EY made it a solid work experience. My colleagues too, were very supportive when I needed assistance in comprehending a few subjects. I would sincerely like to thank EY for giving me the opportunity to intern here. If I am given the opportunity again, I would like to come back and work for EY.