**Section 2:**

With the course’s focus on classification and the encouragement in the instructions to do a project involving classification, we decided to create a classification model that would predict a CEO (Chief Executive Officer) versus a CFO (Chief Financial Officer). One of our biggest challenges involved how to come to our question. This is not a dataset that we found on Kaggle or a site that specifically curated it for the purposes of using and discovering classification techniques. Rather, it is a completely unedited data scrub from the SEC API. The process to discover our question was iterative, and we relied on exploratory data analysis and combining our intuitions in order to carve our path. Our overall goal was simple: demonstrate knowledge gained in ADS502 by using key learnings from the course to apply to “wild data.”

With this in mind, we started with two datasets, one for executive pay, and the other for company financial data. We extracted only the companies listed in the S and P 500 over the last 10 years. For each company’s financial records, we turned excel workbook files into CSV files in order to more easily manipulate the data. Thereafter, we combined the CSV files of the S and P 500 companies into a single Pandas data frame. We then grouped the data into average values for every year based on its ticker. For example, Apple will have average values specific to its financials for each year over the last 10 years.

Next, we moved to the executive pay dataset, and extracted the executive information for those individuals within the S and P 500 ticker list over the last 10 years. After this, we combined the two separate data frames into one by joining them via their tickers. Now that we had our main data frame ready, we performed cleaning and pre-processing before sub-setting the data frames by CEO and CFO. We did this both so that we could calculate a sum of the earnings based on the ticker grouping over the last 10 years separately for CEOs and CFOs, and also to remove the other executives included who were not either a CEO or CFO. Once this was done, we concatenated them together, essentially pasting one on top of the other. Once this data frame was ready, we partitioned it to be 40% testing and 60% training. This was from the guidance of the textbook, where it states, “for smaller or less complex data sets, one should retain sufficient records for accurate assessment, so that the training sets would contain only 50–67% of the original data" (Tan et al, 2019). We felt that this was a small enough dataset to keep 40% of the records in the test set.

At this point, we were almost ready to run the classification models and select the best one. Before this, however, we first scaled the data because of the large numbers involved when summing over ten years. Then we ran our first classification model as a CART model. We ran the complete cell with the setup, then we decided to create a function so that the other models could use the function to create a confusion matrix based on the model’s train/test datasets. This made running the other models more space-efficient, as only one line of code was required for each one. We proceeded to run CART, C5.0, random forest, logistic regression, naïve bayes, K Neighbors, and support vector machine models.

**Section 4**

1. All the data mining steps should be explained, such as in data preparation and data cleaning steps: what steps do you take to get it into a format amenable to analysis? Are there missing variables or outliers? What methods do you perform for EDA and what are your findings?

The first step was to get the data we needed from the SEC API for executive pay, and from Stockrow for the company financials. After gathering the data from both sources, we decided we wanted to use only S&P 500 data and so we also loaded a data source containing the 503 tickers. We matched the data from the company financials to only return data for each ticker in the list of 503. For each ticker in the list, we created CSV files and then combined all the CSV files into a Pandas data frame to perform manipulations on. Now that our data frame was ready, we narrowed the fields into just ones we wanted: Ticker, Date, Revenue, Gross Profit, Operating Income, Income after Tax, and Net Income Common. Additionally, we created a datetime object from the Date feature and extracted the year out of it. This let us ultimately break down the information by yearly basis so that we could establish yearly trends.

Next, we grouped the data by ticker (representative of company) and year and got the mean values for each ticker per year. To make sure this was done correctly, we sanity checked “AAPL,” Apple, and found that the numbers matched when we did an independent search online. From here, we were ready to join in the executive pay data set and did so based on the ticker; only executive pay information for the S&P 500 companies were included. Note that not all companies were in the executive pay dataset, and after merging them together only 485 companies remained with 18 not present in the executive pay from the SEC API.

For data cleaning, we noted some duplicate records and performed a standard drop\_duplicates() method was used to eliminate these duplicates. The company Agilent had duplicate records that were duplicates in reality but had only one slight difference that caused them to get past the drop\_duplicates method. We eliminated these duplicates by deleting the column that was supposed to contain full names but instead contained “human resources” and “group.” We noted about 200 outliers in the executive pay data and dropped these as well. One such outlier was a salary of 18 billion US dollars in one year. There were also some missing values in the company financials. For instance, 73 of the 485 companies did not have revenue data in 2012 and 14 did not have revenue data in 2021. We decided not to impute values or drop these companies because our classification model was classifying executives as either CEOs or CFOs. The more important metrics, in our eyes, were individual pay statistics.