**DATA PRE-PROCESSING – COMP 541**

**STOCK MARKET PREDICTION (**In progress)

**SARKIS MIKAELIAN**

**CALIFORNIA STATE UNIVERSITY, NORTHRIDGE**

**Data cleaning:** Missing values, noise, and inconsistencies contribute to inaccurate data. We have applied the following techniques and methods to address the problems and clean our data. **1. Missing Values:**

I loaded the data from a CSV file into a Pandas DataFrame, to read a comma-separated values (csv) file into DataFrame and optionally iterating or breaking the file into chunks did the following:

1. # Load the Pandas libraries with alias 'pd'
2. **import** pandas **as** pd
3. # Read data from file 'filename.csv'
4. # (in the same directory that your python process is based)
5. # Control delimiters, rows, column names with read\_csv
6. data = pd.read\_csv("oracle\_data.csv")

Ran isnull:

data.isnull()

Which returns boolean values to see which values are missing/NA.

* No values were missing, there is no need to use a global constant to fill in the missing value or replace all missing attribute values by the same constant such as a label like “Unknown”.

Once we get more data, we will run it through the same procedure. If we are missing any data, we will either label it as unknown and ignore it by using dropna, or use a measure of central tendency for the attribute (the mean or median) to fill in the missing value by using fillna.

Applied the 3 following rules to the data for Discrepancy Detection :

* Unique rule: Which resulted in all data being unique to their fields.
* Consecutive rule: Which resulted in no missing values between the lowest and highest values for the attribute.
* Null rule: Which resulted in the date field having multiple “#” symbols in some of the fields. Eliminated that by replacing it using data.replace by the same sequence date of that year.

**Noisy Data:**

We are going to use different techniques to eliminate random errors or variance in a measured variable.

Outlier Analysis:

We are using data visualization to identify outliers, which may represent noise. We are using **scatter plot** to display a **collection of points**, each having the value of **one variable** determining the position on the **horizontal**axis and the value of the **other variable** determining the position on the **vertical** axis.

fig, ax = plt.subplots(figsize=(16,8))

We are using Pandas to bin values into discrete intervals.

pandas.**cut**(*x*, *bins*, *right=True*, *labels=None*, *retbins=False*, *precision=3*, *include\_lowest=False*, *duplicates='raise'*)

We are cutting it when we need to segment and sort data values into bins. Especially when dealing with continuous variable to a categorical variable. Binning categories into an equal number of bins, or a pre-specified array of bins.

pandas.**cut**(*x*, *bins*, *right=True*, *labels=None*, *retbins=False*, *precision=3*, *include\_lowest=False*, *duplicates='raise'*)

Bin values into discrete intervals.

Use cut when you need to segment and sort data values into bins. This function is also useful for going from a continuous variable to a categorical variable. For example, cut could convert ages to groups of age ranges. Supports binning into an equal number of bins, or a pre-specified array of bins.

Regression:

We are using the following technique to calculate a regression line using SciPy:

**scipy.stats.linregress(***x***,***y=None***)**

This computes a least-squares regression for two sets of measurements, where both arrays should have the same length. If only x is given (and y=None), then it must be a two-dimensional array where one dimension has length 2. The two sets of measurements are then found by splitting the array along the length-2 dimension. We have to insert the data into arrays and then run the following technique to get the desired calculation.

ETL:  
For ETL (extraction/transformation/loading) tools we are using Bonobo which is integrated with Python. The goal here is to capture price from each site and then transform them into a standardized format for later usage. In our case it’s to convert price string into float after cleaning and save it into a text file.

import bonobo  
import requests  
from bs4 import BeautifulSoup

def scrape\_yahoo():  
 price = ''  
 status = ''  
 url = '<https://finance.yahoo.com/quote/ORCL/'>  
 r = requests.get(url, headers=headers)  
 if r.status\_code == 200:  
 html = r.text.strip()  
 soup = BeautifulSoup(html, 'lxml')  
 price\_status\_section = soup.select('.home-summary-row')  
 if len(price\_status\_section) > 1:  
 price = price\_status\_section[1].text.strip()  
 return price

def scrape\_nasdaq():  
 price = ''  
 status = ''  
 url = '[https://www.nasdaq.com/symbol/orcl'](https://www.redfin.com/TX/Dallas/2619-Colby-St-75204/unit-B/home/32251730%27)  
 r = requests.get(url, headers=headers)  
 if r.status\_code == 200:  
 html = r.text.strip()  
 soup = BeautifulSoup(html, 'lxml')  
 price\_section = soup.find('span', {'itemprop': 'price'})  
 if price\_section:  
 price = price\_section.text.strip()  
 return price

def extract():  
 yield scrape\_yahoo()  
 yield scrape\_nasdaq()

def transform(price: str):  
 t\_price = price.replace(',', '').lstrip('$')  
 return float(t\_price)

def load(price: float):  
 with open('pricing.txt', 'a+', encoding='utf8') as f:  
 f.write((str(price) + '\n'))

if \_\_name\_\_ == '\_\_main\_\_':  
 headers = {  
 'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_11\_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36',  
 'referrer': 'https://google.com'  
 }  
 # scrape\_nasdaq()  
 graph = bonobo.Graph(  
 extract,  
 transform,  
 load,  
 )  
 bonobo.run(graph)

Data scrubbing: Using numPy, Pandas in Python, by

1. Dropping unnecessary columns in a DataFrame.
2. Changing the index of a DataFrame.
3. Using .str() methods to clean columns.
4. Using the DataFrame.applymap() function to clean the entire dataset, element-wise.
5. Renaming columns to a more recognizable set of labels.
6. Skipping unnecessary rows in a CSV file.