BANA 620 PROJECT REPORT: DESCRIPTIVE AND PREDICTIVE ANALYTICS APPLIED TO THE SKILLED NURSING FACILITY COST REPORTS

Joshua Cabal

California State University, Northridge

April 14, 2024

Contents

1 Executive Summary

An overview of the project, including key findings, recommendations, and a brief summary of the analysis conducted. This section should be concise and geared toward readers who may not delve into the full details of the report.

2 Introduction

Context and background information about the opportunity the project addresses. This section should outline the objectives of the project and the significance of the analysis being conducted.

3 Methodology

A detailed description of the analytical methods and tools used in the project. This should include data collection processes, data analysis techniques (e.g., statistical methods, machine learning algorithms), and any software or programming languages utilized.

4 Data Description

An overview of the data set(s) used, including sources, size, and characteristics of the data. Highlight any data cleaning or preprocessing steps undertaken to prepare the data for analysis.

4.1 Skilled Nusing Facility Cost Report

Medicare-certified institutional providers are required to submit annual cost reports. These data files contain the highest level of cost report status for cost reports in each reported fiscal years. The cost report contains provider information such as facility characteristics, utilization data, cost and charges by cost center (in total and for Medicare), Medicare settlement data, and financial statement data. CMS maintains the cost report data in the Healthcare Provider Cost Reporting Information System (HCRIS). Skilled Nursing Facilities (SNF) submit their cost report data to HCRIS using form CMS-2540-2010. The reports used are from the years 2015 through 2021.

4.1.1 Preprocessing: Data Loading and Handling Missing Values

Since each year has their own .csv file, the several skilled nursing facility cost reports needed to be concatenated into a single DataFrame. In order to do this, the headers fom years 2020 and 2021 had to be made consistent with the headers from the other years. This was manually completed in Microsoft Excel, and the similar columns were renamed accordingly. Any columns that were unable to be matched were dropped and a Year attribute was added to all records during the import step. Finally, the headers were changed to only lowercase for consistency.

After import, the next step was to handle the null values. I began with column leveling cleaning and defined two different lists which contain the columns that will be dropped: dropNull, dropRedundancy. As the list names demonstrate, these attributes will only hinder model performance, if incldued. The first list, dropNull, contains the columns which contain an overwhelming majority, over 90%, of null values across all records. The second list, dropRedundancy, contains the columns which contain data that are captured within other attributers. Because the submitted financial form requires as breakdown of some of the reported numbers, these columns were removed and only the highest level feature was kept.

To handle the remaining null values, I proceeded with record level cleaning by simple imputation. In this case, I used mean imputation and rounded to the nearest integer. I rounded to the nearest integer because a few of the attributes attain values only belonging to the integers. After all loading, 55 columns remained from the initial 100.

4.1.2 Preprocessing: Handling Outliers

The first method used to handle outliers was by z-score threshold. As the name implies, this method works by choosing a z-score threshold and removing all records which attain values outside this window.

In my case, I used the value of 3 as my threshold and chose the target columns net income and accounts payable. These values were chosen because, after testing against multiple different attributes and with varying thresholds, using these as the target columns handled many of the outliers.

```python
# dealing with a subset of outliers by z-score threshold
def removeOutliers(input_df, column_name, zScoreThreshold):
    mean = input_df[column_name].mean()
    std = input_df[column_name].std()
    z_scores = (input_df[column_name] - mean) / std

    return input_df[(z_scores > -zScoreThreshold) & (z_scores < zScoreThreshold)]

numerical_columns = ['net_income', 'accounts_payable']
for each in numerical_columns:
   df = removeOutliers(fullCostReportdf, each, 3)
```

Listing 1: Function to clean by z-score threshold

After this was completed, the data was manually brushed through for more outliers and the only attribute which contained suspect outliers to be removed was the number of beds. Eleven nursing facilities had reported having over 21000 beds, and therefore, we defined a variable named bedThreshold and removed all the records which had more than this threshold.

5 Analysis and Findings

Presentation of the analysis conducted, including data visualization (charts, graphs, tables) and statistical outputs. This section should detail the insights gained from the analysis, interpreting the results in the context of the business problem.

6 Discussion

Interpretation of the findings, discussing how they address the project objectives and their implications for the business. This section should also cover any limitations of the analysis and considerations for future research.

7 Recommendations

Based on the analysis and findings, provide actionable recommendations for the business. Clearly articulate the expected impact of these recommendations and suggest a plan for implementation.

8 Conclusion

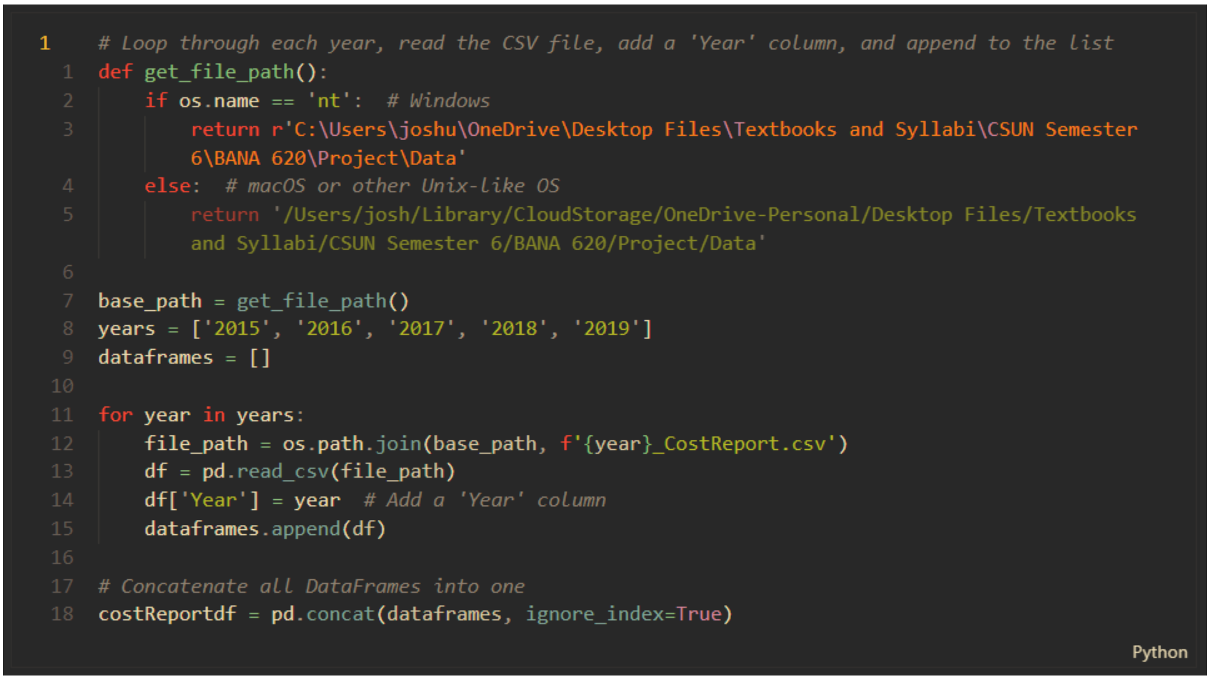
Summarize the key points from the report, reinforcing the value of the findings and recommendations.

9 Appendices

Include any additional material that supports the analysis, such as detailed data tables, code snippets, or extended methodology descriptions.

10 References

List all sources cited in the report, including data sources, literature, and any external references used in the analysis.



```python
# Loop through each year, read the CSV file, add a 'Year' column, and append to the list
def get_file_path():
    if os.name == 'nt':  # Windows
        return r'C:\Users\joshu\OneDrive\Desktop Files\Textbooks and Syllabi\CSUN Semester 6\BANA 620\Project\Data'
    else:  # macOS or other Unix-like OS
        return '/Users/josh/Library/CloudStorage/OneDrive-Personal/Desktop Files/Textbooks and Syllabi/CSUN Semester 6/BANA 620/Project/Data'

base_path = get_file_path()
years = ['2015', '2016', '2017', '2018', '2019']
dataframes = []

for year in years:
    file_path = os.path.join(base_path, f'{year}_CostReport.csv')
    df = pd.read_csv(file_path)
    df['Year'] = year  # Add a 'Year' column
    dataframes.append(df)

# Concatenate all DataFrames into one
costReportdf = pd.concat(dataframes, ignore_index=True)
```

Figure 1: Data loading code snippet