An Analysis of Job Status and Compensation in San Francisco Municipal Employees

Zac Taylor

4/26/16

Introduction

Businesses from all sectors struggle with the issue of incomplete data when conducting research. Incomplete data has the potential to occur in every type of dataset. In most statistical processes the easiest method is to remove the entire incomplete case whenever missing data is encountered. However, if enough cases are removed there is the possibility that there will not be enough data left to get an unbiased, accurate prediction. Even with enough data left over there is the possibility of removing valuable information without even knowing. For instance, if a company is conducting internal salary research and every employee of a particular department did not include their gender, the data from an entire department would be excluded from your research if you were only considering complete observation cases. This could have a serious impact on the accuracy and validity of any type of predictive model and requires careful attention.

The purpose of this project was to analyze an incomplete dataset and develop a highly accurate predictive model capable of predicting as many incomplete variables as possible. The goal was to use these models to predict missing variables with a very high probability. If a probability threshold was met then the predictions for each observation could be added to the incomplete dataset. This would reduce the number of incomplete cases and thus the chances of making inaccurate predictions.

The intended audience would be large corporation managers or researchers that are interested in employee compensation or customer survey analysis. Any large successful corporation will need to conduct research on several different aspects of its business almost constantly. The analysis and reports from their research heavily influence how the corporation is run and which decisions it makes. It is absolutely in their best interest to make sure that their research is as accurate as possible. It would be beneficial if they could use a general algorithm or methodology to allow them to reduce the number of missing cases and improve the quality of their dataset.

Background

There are a few technical terms that need to be clarified for this research. Employee compensation is all pay that an employee receives. This includes other pay, overtime pay, benefits, base pay and total pay. Job status refers to whether or not an employee is full time or part time. The terms cases and observations are used interchangeably in this paper. They refer to a specific employee at a specific row from the dataset (some employees work more than one year). A classification algorithm is used to predict a certain class based on the observations it receives. Training sets are used to train the algorithm to see how well it can make predictions while test sets provide a way to validate those results.

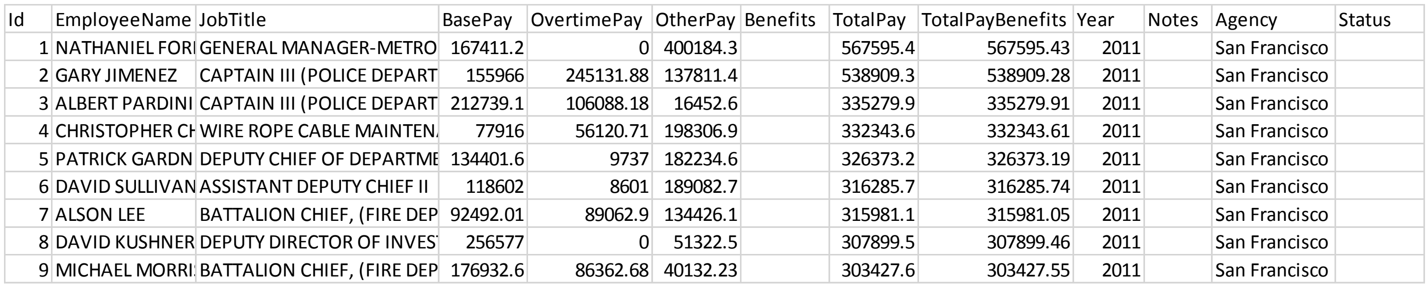
I relied heavily on the skills I learned in my data mining course where I was introduced to a number of different classification algorithms. Knowing how to read API's for packages and where to find answers was one of my biggest assets. This provided me with the framework to conduct my own experiments. Linear algebra also played a very helpful role in helping me understand some concepts, especially PCA. My statistical learning class also helped me understand how to correctly set parameters for several of the packages I was using in addition to a more mathematical understanding of the models. I also had a basic understanding of the R language however it took significantly longer than expected to become proficient.

There were a number of new skills I needed to learn to complete this research. Data cleanup was something that had always been glossed over in tutorials but had to be dealt with directly in this dataset. Based on my experiences, Excel is far easier to format data with than R, so I had to brush up on my Excel skills. I also needed to become much more familiar with the syntax and error reporting of the R language and several different packages. Syntax aside, the ability to implement the majority of the classification and analysis functions required additional outside research for successful implementation.

The two main programming tools used in this project were Excel and R. Excel was used for data cleanup and R was used for all machine learning algorithms. Within R, I utilized a number of different packages. Nnet, neuralnet and NeuralNetTools were used for neural network implementation. Rpart, randomForest and e1071 were used for decision tree and random forest calculations. Rattle, rpart.plot, ggplot and corrplot were used for various plotting functions and visuals.

Formative analysis of dataset and problem area

Every year since 2011 the city of San Francisco has released information on their municipal employees. A very well-known data science website, kaggle.com, compiled the each of the years from 2011-2014 and released them for public use and analysis. The data was released in two forms, CSV and an RSQLite database. This is where the dataset was obtained for this research and the CSV file was used for this research. There are over 148,000 total observations in this dataset that are fairly evenly distributed over each of the years, with the amount of employees steadily increases each year. The year 2011 contains 36,159 observations, 2012 had 36,766, 2013 had 37,606 and 2014 had 38,119. Each observation includes employee name, job title, job status, employment year (2011-2014) and compensation, which includes base pay, total pay, benefits, total pay and benefits, other pay and overtime pay.



Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Agency	Status
1	NATHANIEL FORI	GENERAL MANAGER-METRO	167411.2	0	400184.3		567595.4	567595.43	2011		San Francisco	
2	GARY JIMENEZ	CAPTAIN III (POLICE DEPART	155966	245131.88	137811.4		538909.3	538909.28	2011		San Francisco	
3	ALBERT PARDINI	CAPTAIN III (POLICE DEPART	212739.1	106088.18	16452.6		335279.9	335279.91	2011		San Francisco	
4	CHRISTOPHER CH	WIRE ROPE CABLE MAINTEN/	77916	56120.71	198306.9		332343.6	332343.61	2011		San Francisco	
5	PATRICK GARDN	DEPUTY CHIEF OF DEPARTME	134401.6	9737	182234.6		326373.2	326373.19	2011		San Francisco	
6	DAVID SULLIVAN	ASSISTANT DEPUTY CHIEF II	118602	8601	189082.7		316285.7	316285.74	2011		San Francisco	
7	ALSON LEE	BATTALION CHIEF, (FIRE DEP	92492.01	89062.9	134426.1		315981.1	315981.05	2011		San Francisco	
8	DAVID KUSHNER	DEPUTY DIRECTOR OF INVES	256577	0	51322.5		307899.5	307899.46	2011		San Francisco	
9	MICHAEL MORRI	BATTALION CHIEF, (FIRE DEP	176932.6	86362.68	40132.23		303427.6	303427.55	2011		San Francisco	

The table above shows the first 9 observations from the Salaries dataset. All observations were from the San Francisco Agency, so that variable provided no value and was removed. There were no values in the Notes column for any observation so that column was also removed. The Employee Name column was analyzed in an attempt to derive a gender for each observation based on the name. This would provide another synthetic variable to the dataset. Exploratory research was conducted on the subject and potential packages, however this avenue did not appear promising. One kaggle.com forum poster reported having less than 22% accuracy with gender prediction based on first name. After this decision the Employee Name column was removed and this doubled to keep the dataset anonymous by only referring to observations by their ID.

It is important to note that only observations from 2014 have a value for the job status variable. In addition, there are several thousand observations which have null benefits. These two discoveries were a major hurdle to producing an accurate, unbiased model. The R language has a function to deal with incomplete cases called complete.cases(). This function omits any observations that have missing data from the data frame. In normal situations where only a few variables are missing, this is a quick and simple fix for null data.

In the San Francisco Salaries dataset, tens thousands of observations have missing values, potentially across multiple variables. To compound the problem a large majority of the missing values correspond to different years of the dataset. The majority of missing values for benefits occur in the year 2011 and the only year with any job status observations recorded is 2014. If complete.cases() is used now the majority of the data, specifically any year that is not 2014, will be omitted.

The first option was to use years 2012 – 2014 as the dataset and remove the job status variable. This would greatly reduce the number of incomplete observations but would also remove a potentially significant variable. The observations from 2011 would not be added because setting the benefits of all 2011 observations to 0 would incorrectly label anyone who received benefits that year that were not recorded.

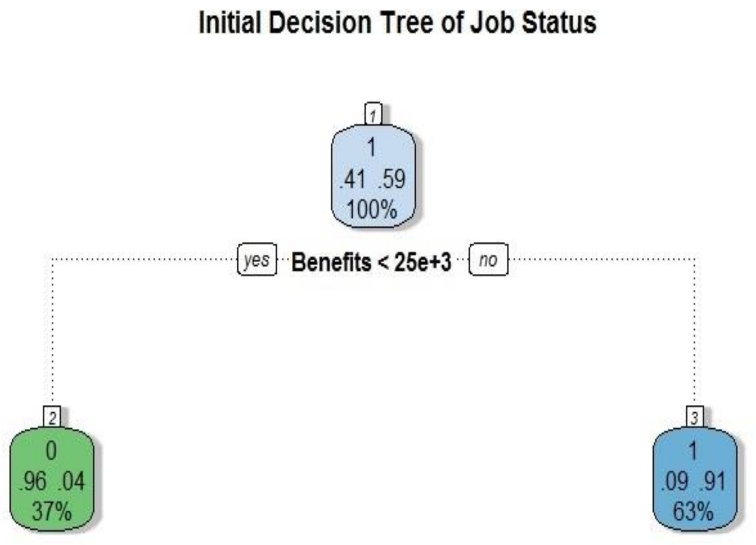
The second option was to only use data from 2014 and keep the job status variable in the model. The majority of this research was focused on retaining this variable and 2014 provides enough observations to generate an accurate general model. The goal was to develop a classifier using supervised learning to predict job status (full or part time) with an accuracy of over 99%. Then apply this model to unsupervised learning for the years 2011-2013. If a high enough accuracy could be achieved on the supervised dataset, there is the potential to apply predictions to the unsupervised data and apply job status labels to years other than 2014.

Methodology

Four different classifiers were chosen to predict job status: neural networks, decision trees, random forests and support vector machines. I began with a neural network but ran into a substantial amount of implementation issues. So I decided to go with a very basic classifier, the decision tree.

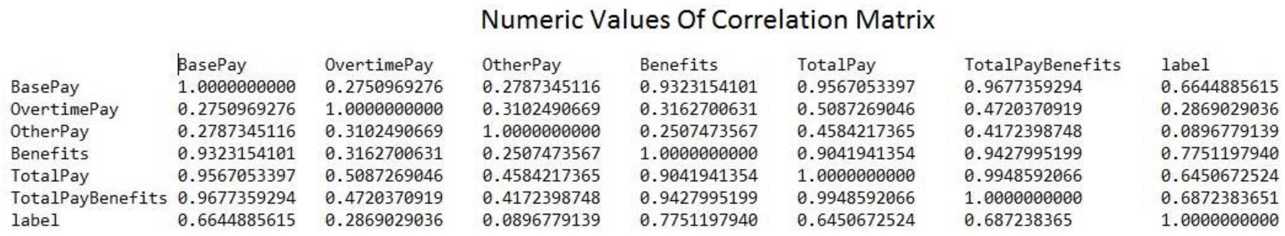
A decision tree is a tree structure with a root node, branches and leaf nodes. Each internal node represents a test on a variable, each branch represents the outcome of said test and each leaf node holds a class label. This model was a fairly simple model but served as a solid baseline function to test the more complex algorithms. I implemented the decision tree using the rpart package and was surprised at the initial results. At the default settings it only needed to branch one time to get an accuracy over 90%. This branch was based off employee benefits being above or below $25,000. Altering the complexity parameter, or the amount by which branching a node improved the relative error, allowed for a higher accuracy and a more complex tree.

In the figure below we have a plot of the initial decision tree representing the accuracies for each label. 96% of the part time employees were classified correctly along with 91% of full time employees.



Although an accurate prediction was attained with the decision initial decision tree, I found it really odd that it could get that accurate of a prediction based on one variable. I decided to do a more in depth analysis of my variables. I began by creating a correlation matrix to determine how positively or negatively correlated the variables were with each other. This yielded very interesting results.

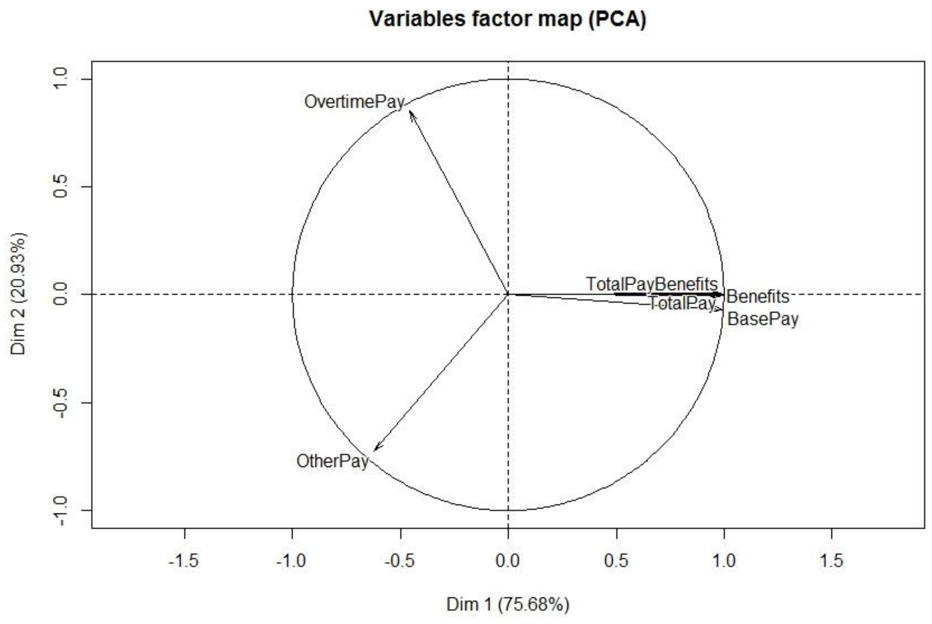
The table below shows that out of seven variables (including the label) four of them show over 90% positive correlation between each other. These include TotalPayBenefits, TotalPay, Benefits and BasePay. The correlation values in the matrix represent how much of the data in one variable is linked to another. If the variables are highly correlated, keeping all of them in the algorithm will provide little additional information because most of the information can be understood from one of the variables.


Numeric Values Of Correlation Matrix

	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	label
BasePay	1.0000000000	0.2750969276	0.2787345116	0.9323154101	0.9567053397	0.9677359294	0.6644885615
OvertimePay	0.2750969276	1.0000000000	0.3102490669	0.3162700631	0.5087269046	0.4720370919	0.2869029036
OtherPay	0.2787345116	0.3102490669	1.0000000000	0.2507473567	0.4584217365	0.4172398748	0.0896779139
Benefits	0.9323154101	0.3162700631	0.2507473567	1.0000000000	0.9041941354	0.9427995199	0.7751197940
TotalPay	0.9567053397	0.5087269046	0.4584217365	0.9041941354	1.0000000000	0.9948592066	0.6450672524
TotalPayBenefits	0.9677359294	0.4720370919	0.4172398748	0.9427995199	0.9948592066	1.0000000000	0.6872383651
label	0.6644885615	0.2869029036	0.0896779139	0.7751197940	0.6450672524	0.687238365	1.0000000000

I wanted to see if I could find another way to achieve these same results. I decided to use Principal Component Analysis to achieve this. Principal component analysis is a method that converts a set of observations of potentially correlated variables into a set of values of linearly uncorrelated variables called principal components. The first principal component captures the majority of the variance in the data with each one after capturing less and less. PCA analysis yielded similar results to the correlation matrix.

In the figure below we can see that the same variables are all clustered together indicating their obvious correlation. Here 75.68% of the data is captured by the first dimension, with 20.93% captured by the second.



This realization led me to create a separate minimal dataset where all of the correlated variables were removed except for benefits. I did this to determine the change in accuracy and to see how drastically computation time could be reduced for each algorithm.

The last steps in my methodology were to implement the more complex algorithms and get results. Random forests were the next step from decision trees. It is an ensemble learning method for classification that constructs a large amount of decision trees and takes the mode of the classifications. Random forests tend to correct the decision tree's habit of overfitting the training set.

The next algorithm used was a support vector machine (SVM). This is another supervised learning algorithm that can be used in classification. Support vector machines work by splitting the data with a line into two halves with each class observation on a separate side. It then determines the maximum margin hyperplane, or the maximum distance between the closest points to the line from both classes. The points closest to the line are our support vectors. SVM's are unique from other classifiers in that they only value the support vector points while essentially ignoring the rest of the data points.

To come full circle I ended my research by re-implementing neural networks. This was much more achievable the second time around because of a much better understanding of the dataset and variables. Neural networks attempt to simulate neurons in the brain that you can teach things to like pattern recognition and decision making. Neural nets consist of an input layer, hidden layer and an output layer with interconnected neurons making up the network. In this research, inputs are the variables and the output is the job status label.

Results

The results of my classifiers were fairly straight forward. I used the caret package to help predict the accuracies of all models. My initial results were achieved using the original dataset containing all variables. These were then compared with the algorithm's results on the minimal dataset where the three highly correlated variables, TotalPay, TotalPayBenfits and BasePay were omitted. The minimal dataset was far easier to work with and the computation time for all algorithms was significantly reduced. The neural networks received the most drastic computational time reduction.

In the table below are the results from both datasets. In every model except SVM, full time accuracy prediction is consistently higher than part time. The baseline model, decision tree, was the least accurate of the four models with 92.9%. The best model was the neural network at 98.16%.

Initial Results

	Overall Accuracy	PT Accuracy	FT Accuracy
Decision Tree	0.9290099	0.9120269	0.9571767
Random Forest	0.9663964	0.9546928	0.9663964
SVM	0.9515714	0.9688549	0.9404823
Neural Net	0.9816030	0.9789056	0.9852908

Results on minimal Dataset

	Overall Accuracy	PT Accuracy	FT Accuracy
Decision Tree	0.9267550	0.9094957	0.9559446
Random Forest	0.9301480	0.9168121	0.9520421
SVM	0.9258106	0.9031567	0.9655870
Neural Net	0.9657292	0.9547920	0.9723041

Decision trees were the least affected by the removal of the variables, noting only a 0.3% change. This is attributed to the simplicity of the model. Random forests were the most affected with an accuracy difference of around 3.6%.

Although a 98% accuracy was achieved, it is not high enough to artificially insert those predictions into the unlabeled data. I was hoping for above 99% accuracy to be able to implement this idea. More work will be needed to tune the models to achieve this level of precision.

Lessons Learned

There were a number of experiences that were more difficult than expected. Neural nets provided a significant challenge until I had a better grasp of my data. Data cleanup was not overly difficult but it was tedious. I was surprised at how much planning and time it took to get my data into a usable state. Job title classification was also a major hurdle. There were hundreds of different job titles, several with different job levels as well. I tried to find ways to break them into classes and keep record of it all but I could not find a good method for it. The only real thing that surprised me was how easily the decision tree made its classification. I knew benefits were a significant factor in job status but I had no idea it could predict it with over 90% accuracy based on that alone.

My advice for a future student is to know where and how to look for answers. Reading package websites or API's and utilizing Stack overflow are invaluable tools. If they are using R, the help function will save a lot of time. The first thing to focus on is getting your useable data together. Don't worry about which algorithms you are going to implement until you have a solid understanding of your data. It is more important to know which algorithm will perform best given the data you have rather than which algorithm you think will be the most successful. My final bit of advice is to try not to do too many things at once. Segment out the research and focus on completing different parts of it as you go instead of trying to dabble in every cool idea you can think of.

Annotated Bibliography

ALICE, M.

Fitting a neural network in R; neuralnet package

Alice, M., 2015. Fitting a neural network in R; neuralnet package. *R-bloggers*.
http://www.r-bloggers.com/fitting-a-neural-network-in-r-neuralnet-package/

This blog provided a nice tutorial that explained several aspects of the neuralnet package in R. It also showed me how to plot the neural network and gain a better overall understanding of neural networks.

CORRELATION MATRIX: A QUICK START GUIDE TO ANALYZE, FORMAT AND VISUALIZE A CORRELATION MATRIX USING R SOFTWARE - DOCUMENTATION - STHDA

2016. Correlation matrix: A quick start guide to analyze, format and visualize a correlation matrix using R software - Documentation - STHDA. *Sthda.com*.
http://www.sthda.com/english/wiki/correlation-matrix-a-quick-start-guide-to-analyze-format-and-visualize-a-correlation-matrix-using-r-software

This website provided an in depth analysis of correlation matrices and how to implement them in the R language. It also provided information on how to use the corrplot package to create correlation matrix visuals.

CREATING, VALIDATING AND PRUNING THE DECISION TREE IN R | EDUREKA BLOG

2015. Creating, Validating and Pruning the Decision Tree in R | Edureka Blog. *Edureka Blog*.
http://www.edureka.co/blog/implementation-of-decision-tree/

This blog provided a tutorial on implementing decision trees using the rpart package. It also provided examples of decision tree plots using the fancyRpart package.

KAGGLE: THE HOME OF DATA SCIENCE

2016. Kaggle: The Home of Data Science. *Kaggle.com*.
https://www.kaggle.com/kaggle/sf-salaries

Kaggle is where I found my dataset for this research. It also provided a helpful forum and information other data scientists had created on exploratory data analysis. This provided a basis for starting my research.

PRINCIPAL COMPONENT ANALYSIS IN R : PRCOMP() VS. PRINCOMP() - R SOFTWARE AND DATA MINING - DOCUMENTATION - STHDA

2016. Principal component analysis in R: prcomp() vs. princomp() - R software and data mining - Documentation - STHDA. *Sthda.com*.

http://www.sthda.com/english/wiki/principal-component-analysis-in-r-prcomp-vs-princomp-r-software-and-data-mining

This website provided an analysis of two different packages for conducting Principal Component Analysis. It helped me choose which package was best for my research and how to plot the PCA.

THE CARET PACKAGE

2016. The caret Package. *Topepo.github.io*.

http://topepo.github.io/caret/index.html

The caret package allowed me to get accurate predictions for my models and helped with data splitting and pre-processing. This website provided everything I needed to know about using and implementing the caret package.

THE COMPREHENSIVE R ARCHIVE NETWORK

2016. The Comprehensive R Archive Network. *Cran.r-project.org*.

https://cran.r-project.org/

The CRAN website provided documentation on every R package. It also included several tutorials that helped me in the beginning of my research getting reacquainted with the R language.