# Data Analyst Nanodegree

# Project 6

# Data Visualization

(September 2015)

References:

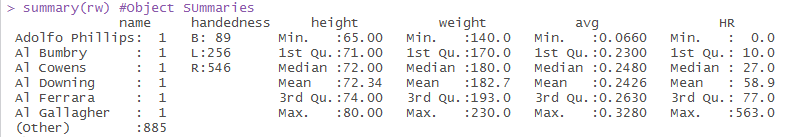
* DataSets: <https://docs.google.com/document/d/1w7KhqotVi5eoKE3I_AZHbsxdr-NmcWsLTIiZrpxWx4w/pub?embedded=true>
* Evaluations: <https://docs.google.com/document/d/1zRVs73M7P5ACKB0n3Di4k0AskId3pc6lIpMBmmydETk/pub>
* Code:
  + US Airlines: <https://github.com/allanbreyes/udacity-data-science/tree/master/p5>

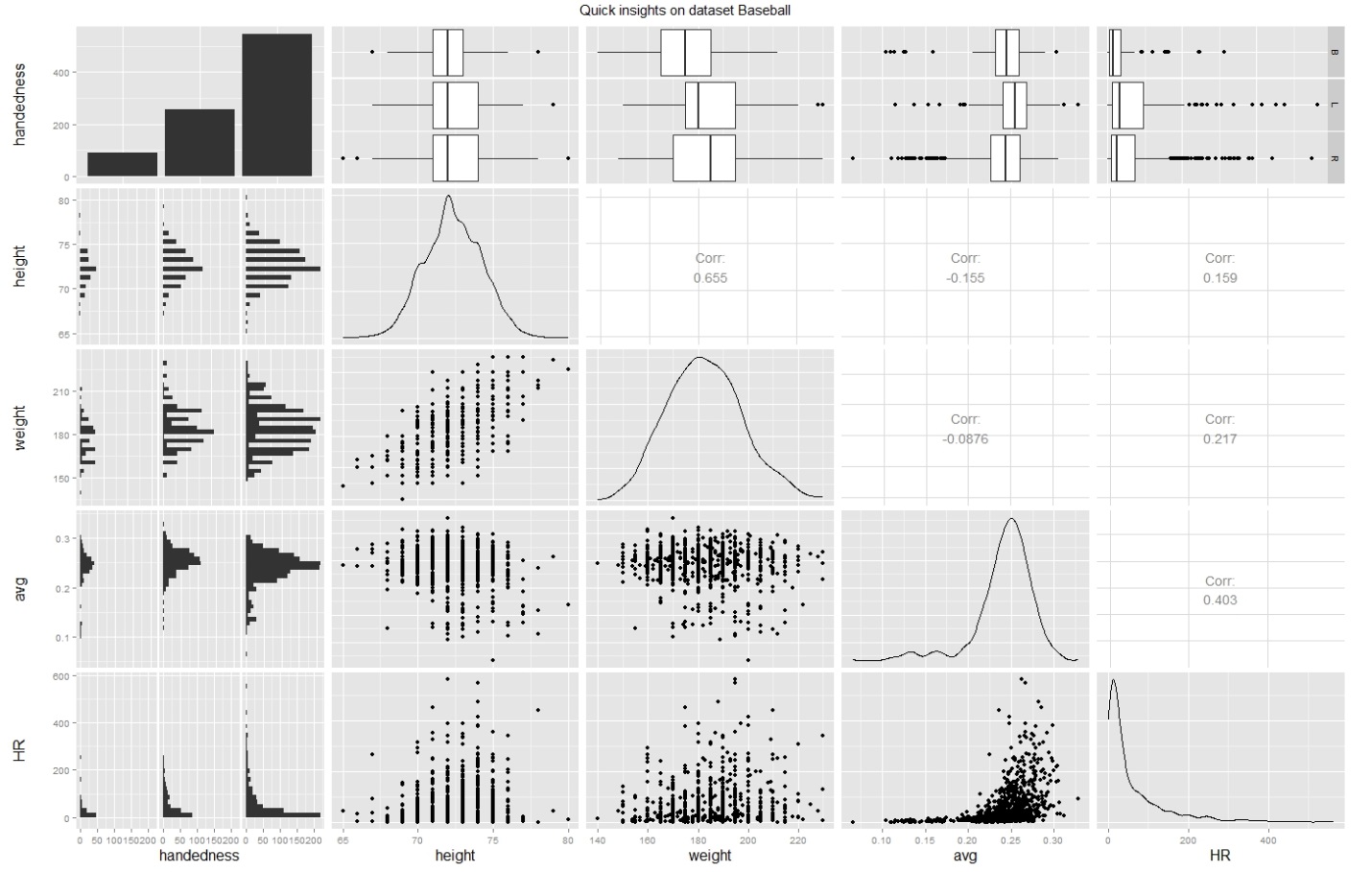
## Data:

|  |  |  |  |
| --- | --- | --- | --- |
| **Beginner** | [**Baseball Data**](https://www.google.com/url?q=https://s3.amazonaws.com/udacity-hosted-downloads/ud507/baseball_data.csv&sa=D&usg=AFQjCNEjepW24JPVyIpheLuF7wHlFJd2rg) | A data set containing 1,157 baseball players including their handedness (right or left handed), height, weight, batting average, and home runs. | Create a visualization that shows differences among the performance of the baseball players. |

What is Batting Average - AVG? A statistic that measures a **player's batting ability** by comparing the number of hits he has to his number of at bats. A player's batting average is used to evaluate his individual skill at batting. Batting Average does not include walks or hit by pitches. It also does not specify the difference between a single, double, triple, or home run - each are considered to be a single hit. Batting average is calculated as follows: hits divided by at bats. For example, during the 2011 MLB season Jose Bautista had 155 hits over 513 at bats for a .302 batting average.

Analyzing the data using R:





Data Cleaning

* Remove data sets with 0 for HR and avg
* We had to player with the same name, so I added an initial to be able to differiate them in the reporting (L for left handed)

## How do I Complete this Project?

This project is connected to the [**Data Visualization**](https://www.udacity.com/course/viewer#!/c-ud507-nd) course, but depending on your background knowledge of data visualization, [**dimple.js**](http://dimplejs.org/), and [**d3.js**](http://d3js.org/) you may not need to take the whole course to complete this project.

After completing Lesson 2 and Problem Set 2 of the course, you will be able to complete this project since you will have learned about dimple.js.

If you want to become more technical and expand your skill set, you can continue to Lesson 3 and Lesson 4, in which you will learn more about narrative structures and how to create graphics using d3.js. The d3.js library has a steeper learning curve, and we encourage you to take on the challenge if you desire.

The process for evaluating your project is not affected by your choice of using [**dimple.js**](http://dimplejs.org/) or [**d3.js**](http://d3js.org/).

### Introduction

For the final project, you will create an **explanatory** data visualization from a data set that communicates a clear finding or that highlights relationships or patterns in a data set. Your work should be a reflection of the theory and practice of data visualization, and you must use either [**dimple.js**](http://dimplejs.org/) or [**d3.js**](http://d3js.org/).

We will provide some options of data sets to explore; however, you may choose to explore an entirely different data set. You should be aware that finding your own data set and cleaning it using Python, R, or some other language can take considerable time and effort. This can add as much as a day, a week, or even months to your project so embark on the adventure to find and clean a data set if you are truly prepared with programming and data wrangling skills.

You have three options for this project. You should pick an option based on your prior experience with data munging and exploratory data analysis. The option you choose will not affect the evaluation of the project.

* **Option 1**   
  Select one of the beginner data sets, which already has a summary of findings, from the [**Data Set Options**](https://docs.google.com/document/d/1w7KhqotVi5eoKE3I_AZHbsxdr-NmcWsLTIiZrpxWx4w/pub?embedded=true) document. Then, create a visualization that communicates the findings.
* **Option 2**   
  Select one of the intermediate data sets from the [**Data Set Options**](https://docs.google.com/document/d/1w7KhqotVi5eoKE3I_AZHbsxdr-NmcWsLTIiZrpxWx4w/pub?embedded=true) document. You will investigate the data set to share a story or message about the data and then create a suitable visualization.
* **Option 3**   
  Find a data set, investigate it, and share your findings in a visualization. Your final graphic should primarily be explanatory, but it may also contain exploratory components. You can find a list of recommended websites to find data sets in the [**Data Set Options**](https://docs.google.com/document/d/1w7KhqotVi5eoKE3I_AZHbsxdr-NmcWsLTIiZrpxWx4w/pub?embedded=true) document. You should be aware that finding your own data set, cleaning the data set, and analyzing it (using R, iPython Notebook, or another tool) can take considerable time and effort. This can lengthen the time you spend on your project by days, weeks, or even months. Choose the option only if you feel prepared for a challenge!

Now, on to the details!

### Step One - Choose a Data Set

First, you will choose a data set from the [**Data Set Options**](https://docs.google.com/document/d/1w7KhqotVi5eoKE3I_AZHbsxdr-NmcWsLTIiZrpxWx4w/pub?embedded=true) document or find a data set to explore and visualize. You should choose a data set based on your prior experiences in programming and working with data. The data set you choose will not increase or decrease your chances of passing this project.

### Step Two - Get Organized

Eventually you’ll want to submit your project and share it. If you are familiar with [**GitHub**](https://github.com/), we encourage you to create a public repository or a public [**Gist**](https://gist.github.com/) for your project to track changes. Otherwise, you need to create the following files.

* an **index.html** file containing the code to create your visualization (you may include the JavaScript and CSS in this file or separate them in other files)
* a **README.md** file that includes four sections...
  + **Summary** - in no more than 4 sentences, briefly introduce your data visualization and add any context that can help readers understand it
  + **Design** - explain any design choices you made including changes to the visualization after collecting feedback
  + **Feedback** - include all feedback you received from others on your visualization from the first sketch to the final visualization
  + **Resources** - list any sources you consulted to create your visualization
* **data files**
  + the final data set used to create the visualization (usually .csv, .tsv, or .json file)
  + a codebook or other files related to the data set (description, readme, license)
* OPTIONAL FOLDERS IF YOU USE [**GITHUB**](https://github.com/)
  + **data** folder to include all the data related files
  + **js** folder to include .js files (not needed if javascript is in the index.html file)
  + **css** folder to include .css files (not needed if CSS is in the index.html file)

### Step Three - Find a Data Story

Explore your data set and craft a message or story around your data! Think about the overall message you want to convey and think about the comparison(s) or relationship(s) you want your readers to see.

### Step Four - Create Your Visualization

First, sketch ideas for your visualization. Once you settle on a sketch, explain any design choices in that sketch, such as chart type, visual encodings, and layout, in the **Design** section of the **README.md** file. Then, write code to create your visualization using either [**dimple.js**](http://dimplejs.org/) or [**d3.js**](http://d3js.org/). The visualization must include animation, interaction, or both. See the [**Project Rubric**](https://docs.google.com/document/d/1zRVs73M7P5ACKB0n3Di4k0AskId3pc6lIpMBmmydETk/pub) for more information.

### Step Five - Get Feedback

Share your visualization with **at least 3 other people** and document their feedback. There are many ways to get feedback, and more feedback is generally better! Here are some options. - Share your visualization with others in person and have them think aloud as they read and explore the graphic so you can document what stands out to them and how they interpret the graphic. - Share a link to your repository in the discussions and ask others to share constructive criticisms. Be sure to offer advice to others who are seeking feedback too!

* Create and share a [**Gist**](https://gist.github.com/), which contains an **index.html** file, data file, and any .js or .css files). Directions for creating and sharing a Gist can be found at [**http://bl.ocks.org/**](http://bl.ocks.org/).
  + **Box Plots Gist EXAMPLE**:
    - <https://gist.github.com/mbostock/4061502>
    - <http://bl.ocks.org/mbostock/4061502>

You might need to ask specific questions to prompt the reader. Here are some questions to help you. You can, of course, ask others.

* What do you notice in the visualization?
* What questions do you have about the data?
* What relationships do you notice?
* What do you think is the main takeaway from this visualization?
* Is there something you don’t understand in the graphic?

### Step Six - Document Feedback and Improve the Visualization

For each person that gives you feedback, add the person’s feedback to your **README.md** file in the Feedbacksection. As you improve and iterate on your visualization, update your code **AND** describe any changes in theDesign section of the **README.md file**.   
  
You should save multiple versions of your data visualization after you make changes to it. You can do this using GitHub or a Gist by making commits to your project, or you can simply save multiple version of you data visualization such as index1.html, index2.html, … , index\_final.html. Remember to save related files with similar numbers…

* main1.js, main2.js, … , main\_final.js (if you separate your Javascript from the HTML file)
* style1.css, style2.css, … , style\_final.css (if you separate your styling from the HTML file)

When should you save your files? You should save your files whenever you have a working version of your data visualization. If you get feedback and make changes, then wait to save the file until you have the data visualization working as you want it. Your goal is to build evidence that you have shared your visualization, received feedback, and responded to that feedback. You will need to submit the different versions of your visualization. **At a minimum, you need to submit an initial version of your data visualization (either as a sketch or as code) and the final index.html file and related files.**

### Step Seven - Review

Use the [**Project Rubric**](https://docs.google.com/document/d/1zRVs73M7P5ACKB0n3Di4k0AskId3pc6lIpMBmmydETk/pub) to review your project. If you are happy with your submission, then you’re ready to submit your project. If you see room for improvement, keep working to improve your project!

**Evaluation**

Use the [**Project Rubric**](https://docs.google.com/document/d/1zRVs73M7P5ACKB0n3Di4k0AskId3pc6lIpMBmmydETk/pub?embedded=true) to review your project. If you are happy with your submission, then you are ready to submit! If you see room for improvement in **any** category in which you do not meet specifications, keep working!

Your project will be evaluated by a Udacity reviewer according to the same [**Project Rubric**](https://docs.google.com/document/d/1zRVs73M7P5ACKB0n3Di4k0AskId3pc6lIpMBmmydETk/pub?embedded=true). Your project must "meet specifications" or "exceed specifications" in each category in order for your submission to pass.

**Submission**

Ready to submit your project? Go back to your Udacity Home, click on the project, and follow the instructions to submit!

* You can either send us a GitHub link of the files or upload a compressed directory (zip file).
* Inside the zip folder include a text file with a list of Web sites, books, forums, blog posts, GitHub repositories etc that you referred to or used in this submission (Add N/A if you did not use such resources).

It can take us up to 2 weeks to grade the project so keep checking back for updates.

If you are having any problems submitting your project or wish to check on the status of your submission, please email us at dataanalyst-project@udacity.com.

**What to include in your submission?**

1. the original index.html file for the first version of your graphic
2. the final index.html file for the final version of your graphic
3. the README.md file with the sections Summary, Design, Feedback, and Resources
4. the final data set file used for the graphic (usually .csv, .tsv, or .json)
5. A list of Web sites, books, forums, blog posts, github repositories, etc. that you referred to or used in creating your submission (add N/A if you did not use any such resources).
6. OPTIONAL: additional versions of your index.html as you iterated on your visualization based on feedback (index1.html, index2.html, index3.html, ... , index\_final.html)

## Introduction

Enron Corporation was an American energy, commodities and services company based in Houston, Texas. Enron Corporation bankrupted on December 2, 2001 because of corporate fraud and corruption. Enron is one of the largest corporate fraud cases in American history. The purpose of this project is to identify persons of interest (POI) in the fraud case from 146 executives at Enron. A (POI) is defined as a person who indicted for fraud, settled with the government, or testified in exchange for immunity. This report will use supervised machine learning techniques to construct a POI identifier with given information.

## The Enron Data

The Enron dataset includes 146 executives in the dataset. Each data point contains 14 financial features, 5 email features and 1 target label. The target label flags if this specific person is a POI or not. However, the target labels in the Enron data is unbalance where there are only 18 POIs in the dataset. The first step is to screen the available data points in each feature (14 financial features, 5 email features and 1 target label). Table 1 shows all available data points in each feature.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Email features** | | **Financial features** | | **Target Label** | |
| **Feature** | **# data** | **Feature** | **# data** | **Feature** | **# data** |
| to\_messages | 86 | total\_stock\_value | 126 | PoI | 146 |
| from\_poi\_to\_this\_person | 86 | total\_payments | 125 |  |  |
| from\_messages | 86 | restricted\_stock | 110 |  |  |
| from\_this\_person\_to\_poi | 86 | exercised\_stock\_options | 102 |  |  |
| shared\_receipt\_with\_poi | 86 | salary | 95 |  |  |
|  |  | expenses | 95 |  |  |
|  |  | other | 93 |  |  |
|  |  | bonus | 82 |  |  |
|  |  | long\_term\_incentive | 72 |  |  |
|  |  | deferred\_income | 49 |  |  |
|  |  | deferral\_payments | 39 |  |  |
|  |  | restricted\_stock\_deferred | 36 |  |  |
|  |  | director\_fees | 17 |  |  |
|  |  | loan\_advances | 4 |  |  |

Table 1

### Outlier Investigation

The next step is to remove any outliers or obvious mistakes in the provided dataset. After a manual inspection of the 146 financial data sets two outliers were removed which are “TOTAL” and “THE TRAVEL AGENCY IN THE PARK”. These two data entries are not suspects or part of our investigation in the Enron Company. Other than that, all the data points are reserved for further analysis, 144 in total.

### Optimize feature selection

After cleaning the data I went through the next step of feature selection. For that matter I used univariate feature selection by selecting the best features based on univariate statistical tests. Since the datasets are interval data points, the select k best method uses ANOVA to compute the features. It is worth noting that the best results were achieved with maximum feature set using the Linear Regression Algorithm. The results met already the given requirements of recall and precision greater than 0.3.

As a next step I added 3 new features to further boost the performance. The goal was to increase precision and recall values and reducing features needed to increase the performance of the algorithm.

Feature\_1: Adding “total\_stock\_value” and “exercise\_stock\_options” together. Why? – my hypothesis is that people with high Stock value have a big stake in this scandal and might have tried to exercise (insider trade)

Feature\_2: Adding “total\_stock\_value” and “salary” together: Why? My assumption is that persons with high compensation might have a higher likelihood to be involved

Featue\_3: Multiplying “shared\_receipt\_with\_poi” with “total\_payments”. Why? – I assume that people with high compensation and being part of the inner circle of “poi’s” might have a higher likelihood to be involved

Since I used “total\_stock\_value” in 2 new features I removed it from the data set. The above mentioned goals were met at k=4. Attached the used features:

* 'bonus'
* 'exercised\_stock\_options'
* 'new\_features\_1'
* 'new\_feature\_2'

In a last step I used MinMaxScaler (for algorithm where appropriated) to scale all features between 0 and 1. The features needed to be scaled since they were on vastly different scales, ranging from hundreds of e-mails to millions of dollars.

After selecting and scaling the available features I also introduced transformation techniques. My focus was on Principal Component Analysis (PCA). My intent was to further reduce features to the most important principal components to further tune the ML algorithms.

## Algorithm selection and Tuning

For the analysis of the Enron data I focused on the following classifiers:

* Gaussian Naive Bayes
* Linear Support Vector Classifier
* K Nearest Neighbor Classifier
* KMeans  
  (note - unsupervised learning algorithm, but wanted to test it for 2 cluster POI, non POI)
* AdaBoost with PCA
* Logistic Regression

If appropriate I followed for each algorithm the following evaluation framework:

1. If appropriated for the algorithm, scales the features between 0-1
2. Selects the KBest features using Anova F-value scoring for classification, try different values for parameter k = [1, max # features]
3. Uses KBest features to reduce dimensionality further by applying PCA
4. Using the resulting PCA components in the classifier. Optimize the output by using GridSearchCV() with a set of predefined parameters for each algorithm
5. Print best result for k, features, recall, precision

### Gaussian Naive Bayes

This was the algorithm preselected by udacity and I used it as the benchmark. GaussianNB().

|  |  |  |  |
| --- | --- | --- | --- |
| GaussianNB() – no added features | | | |
| K | **20** | Recall | **0.19** |
| Precision | **0.27** | Features | **No added features** |

|  |  |  |  |
| --- | --- | --- | --- |
| GaussianNB() – added 3 features | | | |
| K | **22** | Recall | **0.26** |
| Precision | **0.35** | Features | **Added features** |

### Linear Support Vector Classifier

To evaluate the impact of my added features and the kBest feature selection for different k = [2, len(features\_list) I have added a table to show the precision and recall values for different k measures and for the original set of features and with the added features.

Parameters = {'C': [1000.0, 5000.0, 10000.0, 50000.0, 100000.0]}

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **no added features** | | **added features** | |  |  | **no added features** | | **added features** | |
| **k** | **prec.** | **recall** | **prec.** | **recall** |  | **k** | **prec.** | **recall** | **prec.** | **recall** |
| **2** | 0.149 | 0.338 | 0.151 | 0.358 |  | **13** | 0.183 | 0.302 | 0.188 | 0.315 |
| **3** | 0.143 | 0.267 | 0.127 | 0.205 |  | **14** | 0.183 | 0.303 | 0.181 | 0.292 |
| **4** | 0.175 | 0.288 | 0.178 | 0.275 |  | **15** | 0.202 | 0.358 | 0.204 | 0.348 |
| **5** | 0.164 | 0.290 | 0.157 | 0.260 |  | **16** | 0.221 | 0.365 | 0.203 | 0.335 |
| **6** | 0.132 | 0.268 | 0.141 | 0.263 |  | **17** | 0.223 | 0.373 | 0.193 | 0.298 |
| **7** | 0.129 | 0.215 | 0.137 | 0.272 |  | **18** | 0.213 | 0.357 | 0.184 | 0.305 |
| **8** | 0.118 | 0.237 | 0.187 | 0.388 |  | **19** | 0.211 | 0.367 | 0.177 | 0.282 |
| **9** | 0.120 | 0.217 | 0.210 | 0.427 |  | **20** | 0.227 | 0.370 | 0.186 | 0.303 |
| **10** | 0.131 | 0.240 | 0.186 | 0.310 |  | **21** | - | - | 0.182 | 0.327 |
| **11** | 0.121 | 0.222 | 0.193 | 0.328 |  | **22** | - | - | 0.177 | 0.323 |
| **12** | 0.133 | 0.247 | 0.195 | 0.337 |  |  |  |  |  |  |

For the LSCV classifier I achieved the best results for k=9 with added features. It is worth noting that the new features did not only increase the performance but also decreased the amount of features needed which has an impact on performance of the algorithm. For the test set without added features I reach the best output with maximum set of features.

### K Nearest Neighbor Classifier

To evaluate the impact of my added features and the kBest feature selection for different k = [2, len(features\_list) I have added a table to show the precision and recall values for different k measures and for the original set of features and with the added features.

Parameter = {'n\_neighbors': [2, 5], "p":[2,3]}

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **no added features** | | **added features** | |  |  | **no added features** | | **added features** | |
| **k** | **prec.** | **recall** | **prec.** | **recall** |  | **k** | **prec.** | **recall** | **prec.** | **recall** |
| **2** | 0.549 | 0.130 | 0.558 | 0.128 |  | **13** | 0.586 | 0.137 | - | - |
| **3** | 0.649 | 0.210 | 0.598 | 0.132 |  | **14** | 0.586 | 0.137 | - | - |
| **4** | **0.647** | **0.220** | 0.608 | 0.168 |  | **15** | 0.586 | 0.137 | - | - |
| **5** | 0.591 | 0.173 | **0.614** | **0.170** |  | **16** | 0.586 | 0.137 | - | - |
| **6** | 0.479 | 0.133 | 0.545 | 0.130 |  | **17** | 0.569 | 0.137 | - | - |
| **7** | 0.565 | 0.175 | 0.513 | 0.130 |  | **18** | 0.569 | 0.137 | - | - |
| **8** | 0.563 | 0.157 | 0.014 | 0.002 |  | **19** | 0.579 | 0.140 | - | - |
| **9** | 0.563 | 0.157 | 0.014 | 0.002 |  | **20** | 0.579 | 0.140 | - | - |
| **10** | 0.563 | 0.157 | - | - |  | **21** |  |  | - | - |
| **11** | 0.563 | 0.157 | - | - |  | **22** |  |  | - | - |
| **12** | 0.563 | 0.157 | - | - |  |  |  |  |  |  |

For the KNN classifier I achieved the best results for k=4 with no added features. Here the added features did not meet my goal. The values for precision and recall declined and the best performance was achieved with a higher k value.

### KMeans

Parameters = {‘n\_clusters=2’}

This is an unsupervised learning algorithm, but I wanted to test it if I reduce it to 2 clusters, meaning either POI or non POI. Precision was OK, at least above the minimum requirement of 0.3, but very poor on the recall results 0.07

|  |  |  |  |
| --- | --- | --- | --- |
| KMeans() | | | |
| K=22 | **0.86** | Recall | **0.07** |
| Precision | **0.32** | features | **Added features** |

### AdaBoost (with PCA)

To evaluate the impact of my added features and the kBest feature selection for different k = [2, len(features\_list) I have added a table to show the precision and recall values for different k measures and for the original set of features and with the added features.

Parameters = {'base\_estimator':[None, DecisionTreeClassifier(min\_samples\_split=5, max\_features = None), RandomForestClassifier(min\_samples\_split=3, max\_features = None)], 'n\_estimators':[20, 50, 110]}

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **no added features** | | **added features** | |  |  | **no added features** | | **added features** | |
| **k** | **prec.** | **recall** | **prec.** | **recall** |  | **k** | **prec.** | **recall** | **prec.** | **recall** |
| **2** | 0.535 | 0.167 | 0.410 | 0.200 |  | **13** | 0.306 | 0.183 | 0.339 | 0.175 |
| **3** | **0.399** | **0.262** | **0.443** | **0.263** |  | **14** | 0.302 | 0.187 | 0.314 | 0.185 |
| **4** | 0.399 | 0.232 | 0.408 | 0.240 |  | **15** | 0.327 | 0.198 | 0.327 | 0.182 |
| **5** | 0.328 | 0.198 | 0.353 | 0.223 |  | **16** | 0.334 | 0.215 | 0.326 | 0.172 |
| **6** | 0.292 | 0.148 | 0.258 | 0.142 |  | **17** | 0.321 | 0.187 | 0.303 | 0.168 |
| **7** | 0.379 | 0.222 | 0.294 | 0.140 |  | **18** | 0.353 | 0.202 | 0.274 | 0.157 |
| **8** | 0.323 | 0.188 | 0.292 | 0.158 |  | **19** | 0.257 | 0.147 | 0.318 | 0.182 |
| **9** | 0.310 | 0.190 | 0.301 | 0.143 |  | **20** | 0.257 | 0.147 | 0.260 | 0.152 |
| **10** | 0.370 | 0.187 | 0.255 | 0.130 |  | **21** | - | - | 0.260 | 0.152 |
| **11** | 0.340 | 0.203 | 0.365 | 0.187 |  | **22** | - | - | 0.260 | 0.152 |
| **12** | 0.330 | 0.185 | 0.338 | 0.188 |  |  |  |  |  |  |

For the AdaBoost classifier I achieved the best results for k=3 with added features. The best result with the original set of features was also achieved with k=3, but precision was better with added features, meaning fewer false alarms for POIs.

### Logistic Regression (with PCA) – the winner

To evaluate the impact of my added features and the kBest feature selection for different k = [2, len(features\_list) I have added a table to show the precision and recall values for different k measures and for the original set of features and with the added features.

Parameters = {"C":[0.05, 0.5, 1, 10, 10\*\*2,10\*\*5,10\*\*10, 10\*\*20], "tol":[10\*\*-1, 10\*\*-5, 10\*\*-10], "class\_weight":['auto']}

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **no added features** | | **added features** | |  |  | **no added features** | | **added features** | |
| **k** | **prec.** | **recall** | **prec.** | **recall** |  | **k** | **prec.** | **recall** | **prec.** | **recall** |
| **2** | 0.252 | 0.368 | 0.247 | 0.360 |  | **13** | 0.228 | 0.438 | 0.203 | 0.297 |
| **3** | 0.297 | 0.445 | 0.247 | 0.395 |  | **14** | 0.227 | 0.437 | 0.205 | 0.297 |
| **4** | 0.309 | 0.535 | **0.335** | **0.597** |  | **15** | 0.224 | 0.427 | 0.205 | 0.305 |
| **5** | 0.281 | 0.540 | 0.311 | 0.575 |  | **16** | 0.284 | 0.458 | 0.206 | 0.318 |
| **6** | 0.260 | 0.517 | 0.287 | 0.567 |  | **17** | 0.274 | 0.442 | 0.202 | 0.307 |
| **7** | 0.259 | 0.570 | 0.264 | 0.528 |  | **18** | 0.322 | 0.523 | 0.257 | 0.450 |
| **8** | 0.243 | 0.577 | 0.226 | 0.290 |  | **19** | **0.323** | **0.532** | 0.243 | 0.420 |
| **9** | 0.236 | 0.538 | 0.225 | 0.292 |  | **20** | 0.323 | 0.532 | 0.234 | 0.398 |
| **10** | 0.239 | 0.442 | 0.211 | 0.272 |  | **21** | - | - | 0.220 | 0.370 |
| **11** | 0.246 | 0.460 | 0.211 | 0.272 |  | **22** | - | - | 0.220 | 0.370 |
| **12** | 0.243 | 0.467 | 0.211 | 0.292 |  |  |  |  |  |  |

For the Logistic Regression classifier I achieved the best results for k=4 with added features. It is worth noting that the new features did not only increase the performance but also decreased the amount of features needed which has an impact on performance of the algorithm. For the test set without added features I reach the best output with k=19 set of features.

## Validation and Performance

Algorithms may perform differently using different parameters depending on the structure of the underlying dataset. This step needs a lot of attention since this can lead to overturning your algorithm, meaning the algorithm will do extremely well with the training data, but fail miserably on unseen test data.

For each algorithm I used grid search (GridSearchCV) over any major tune-able parameters, over 300 randomized stratified cross-validation stratified splits. The parameters which gave the highest average score were selected for the final model.

Validation is the processed of checking to see how your model performs on unseen data. A common mistake would be tuning your model be able to predict your training data very well , but then having it perform poorly on test data. This is called over-fitting. One of the major goals in validation is to avoid over-fitting, which can be accomplished through a process called cross-validation.

Cross-validation is the process of randomly splitting the data into training and testing data. Then the model can train on the training data, and be validated on the testing data. The model selection process was validated using 300 randomized stratified cross-validation splits and selecting the parameters which performed best. Based on the small amount of data points an explicit hold-out set was not used. A stratified hold-out set of 20% would leave only around 3 POI points to do a one-time final test on. This would also not give much confidence in the precision of the performance metrics on such a small hold-out set, while also negatively impacting the ability to create the model.

**The winner of my analysis** is the Logistic Regression classifier based on my validation KPIs (precision, recall). I could achieve the best performance by using SelectKBest (applied to the extended feature list) and use the top 4 (k=4) features to optimize further with PCA.

|  |  |  |  |
| --- | --- | --- | --- |
| k | **4** | Recall | **0.60** |
| Precision | **0.34** | Features | **Added features** |

Summary of the top score:

## Conclusion

For my model, accuracy would be a sub-optimal evaluation metric due to the sparseness of POI’s being predicted. If I just guessed ‘Not a POI’ for everyone, I would attain 87.5% accuracy while not finding any perpetrators of fraud.

The winner was selected based on 2 KPIS:

1. **Precision** is the likelihood that a person at Enron who is identified as a POI is actually a true POI. The selected algorithm for the winner in this project has 0.34 which means that if the algorithm flags a POI, there would be a 66% chance that this person might not be POI (false alarms)
2. **Recall** defines the probability to predict a POI correctly. I have a 0.60 for our recall measurement, leading to the fact that as a potential POI, there is 60% chance that this algorithm would flag this person as a POI

I had better algorithms with hire score on precision, e.g. KNN with a precision score of 0.6, but with a significant downside on the recall rate. For this project, I would argue that in the context of searching for suspects of fraud in one of the largest cases of corporate fraud, recall is the more important criteria. I would like to find all people who were involved, even if it means I have to investigate and clear more extra innocent people.

Even if I declared a winner for this project, I feel that the performance of the classifier is still far from the practical application. Although the classifier has a 60% to identify a POI correctly it still has a very high % for false alarm. A good classifier should have a higher precision and recall value (closer to 0.8), which means the identifier is able to detect a potential POI and the possibility of being false detection is low.

The challenge in this project is with the given data. I only have 18 POIs in the dataset. This project clearly highlighted the challenges for dealing with smaller, but still complex, datasets.

To further improve the results I would have investigated into:

* More intelligent way of creating new features, and bringing them back in the K-best selection process
* Explore more complex pipelines for the data or even testing Deep Learning algorithm to understand if the results would increase