Project Name: San Francisco Employee Compensation Dataset

OVERVIEW

The San Francisco Controller's Office maintains a database of the salary and benefits paid to City employees since fiscal year 2013. This data is summarized and presented on the Employee Compensation report hosted at http://openbook.sfgov.org, and is also available in this dataset in CSV format. New data is added on a bi-annual basis when available for each fiscal and calendar year.

This is a dataset hosted by the city of San Francisco. The organization has an open data platform found here and they update their information according to the amount of data that is brought in. Update Frequency: This dataset is updated annually.

Data Dictionary

Field Name	Data Type	Definition	Notes (optional)
Year Type	Plain Text	Fiscal (July through June) or Calendar (January through December)	
Year	Plain Text	An accounting period of 12 months. The City and County of San Francisco operates on a fiscal year that begins on July 1 and ends on June 30 the following year. The Fiscal Year ending June 30, 2012 is represented as FY2011-2012.	
Organization Group Code	Plain Text	Org Group is a group of Departments. For example, the Public Protection Org Group includes departments such as the Police, Fire, Adult Probation, District Attorney, and Sheriff.	
Organization Group	Plain Text	Org Group is a group of Departments. For example, the Public Protection Org Group includes departments such as the Police, Fire, Adult Probation, District Attorney, and Sheriff.	
Department Code	Plain Text	Departments are the primary organizational unit used by the City and County of San Francisco. Examples include Recreation and Parks, Public Works, and the Police Department.	

Department Code	Plain Text	Departments are the primary organizational unit used by the City and County of San Francisco. Examples include Recreation and Parks, Public Works, and the Police Department.	
Union Code	Plain Text	Unions represent employees in collective bargaining agreements. A job belongs to one union, although some jobs are unrepresented (usually temporarily).	
Union	Plain Text	Unions represent employees in collective bargaining agreements. A job belongs to one union, although some jobs are unrepresented (usually temporarily).	
Job Family Code	Plain Text	Job Family combines similar Jobs into meaningful groups.	
Job Family	Plain Text	Job Family combines similar Jobs into meaningful groups.	
Employee Identifier	Plain Text	Each distinct number in the “Employee Identifier” column represents one employee. These identifying numbers are not meaningful but rather are randomly assigned for the purpose of building this dataset. The column does not appear on the Employee Compensation report hosted on openbook.sfgov.org, but that report does show one row for each employee. Employee ID has been included here to allow users to reconstruct the original report. Note that each employee’s identifier will change each time this dataset is updated, so comparisons by employee across multiple versions of the dataset are not possible.	
Salaries	Number	Normal salaries paid to permanent or temporary City employees.	
Overtime	Number	Amounts paid to City employees working in excess of 40 hours per week.	
Other Salaries	Number	Various irregular payments made to City employees including premium pay, incentive pay, or other one-time payments.	
Total Salary	Number	The sum of all salaries paid to City employees.	
Retirement	Number	City contributions to employee retirement plans.	
Health/Dental	Number	City-paid premiums to health and dental insurance plans covering City employees. To protect confidentiality as legally required, pro-rated citywide averages are presented in lieu of employee-specific health and dental benefits.	
Other Benefits	Number	Mandatory benefits paid on behalf of employees, such as Social Security (FICA and Medicare) contributions, unemployment insurance premiums, and minor discretionary benefits not included in the above categories.	
Total Benefits	Number	The sum of all benefits paid to City employees.	

Total Compensation	Number	The sum of all salaries and benefits paid to City employees.	

Data Exploration & Visualization

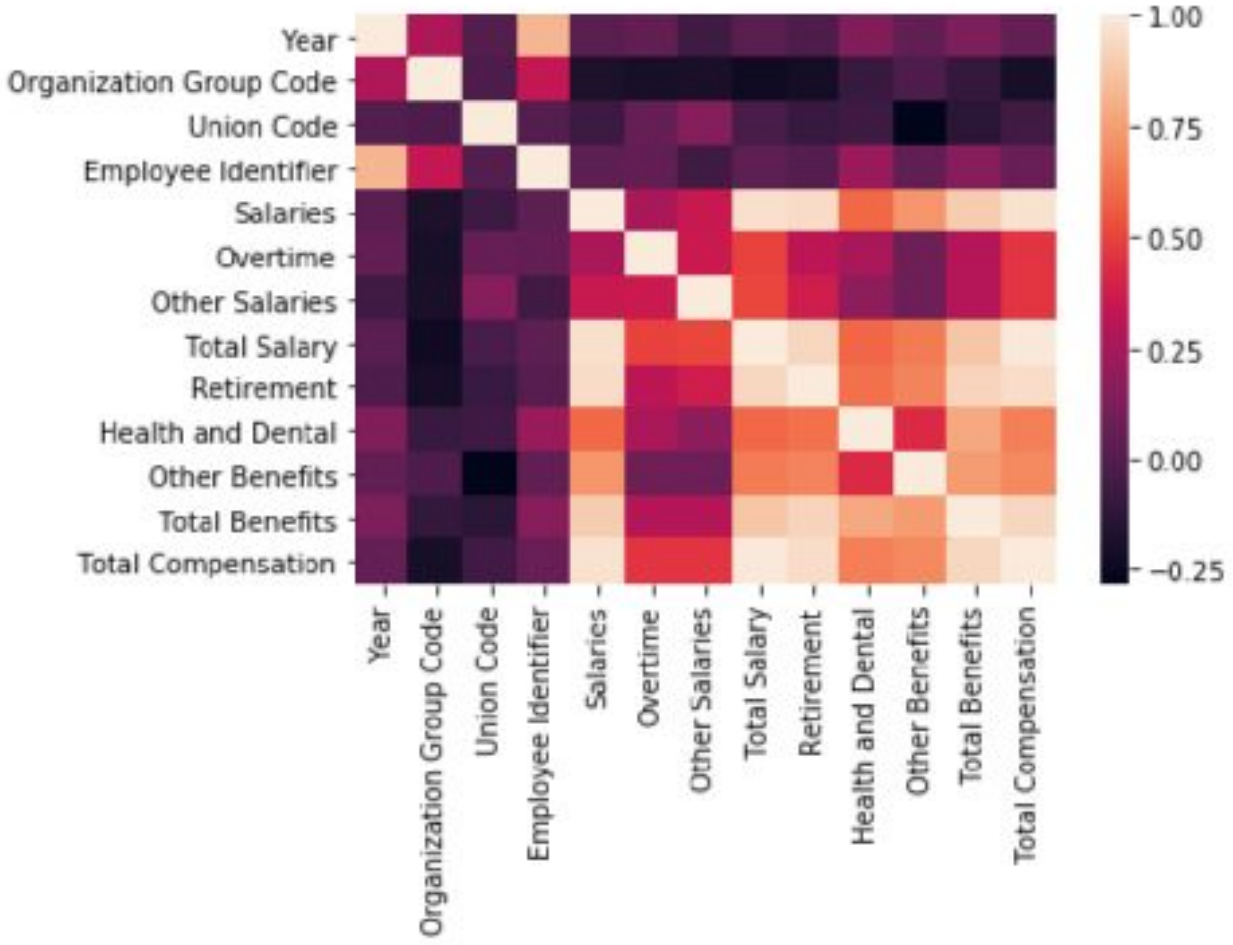
Highlights

1. There are **835307** rows and 22 columns
2. Mix data column with int and string and special chars
3. Quite a bit of bad data

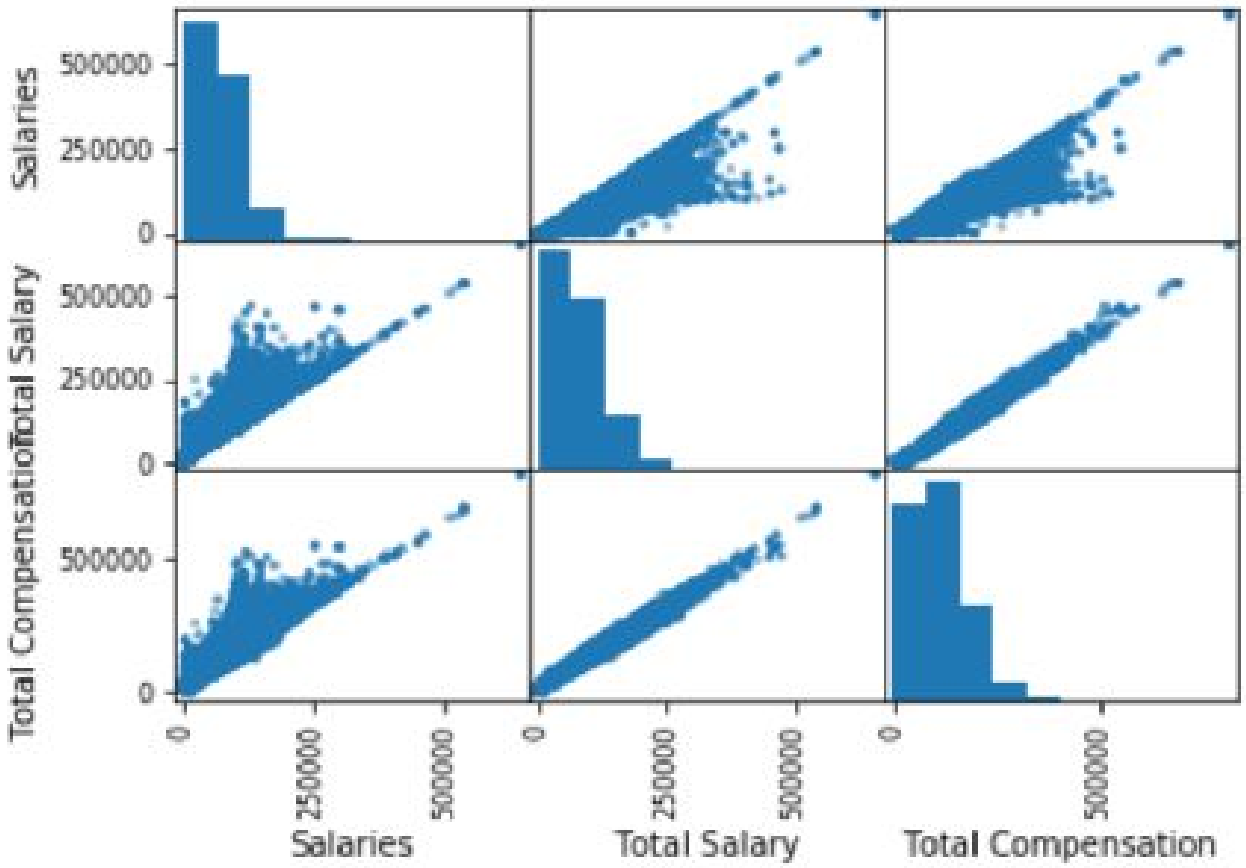
Below image depicts the different columns and their data types.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 835307 entries, 0 to 835306
Data columns (total 22 columns):
Year Type                    835307 non-null object
Year                         835307 non-null int64
Organization Group Code      835307 non-null int64
Organization Group           835307 non-null object
Department Code              806544 non-null object
Department                   434809 non-null object
Union Code                   834746 non-null float64
Union                        834746 non-null object
Job Family Code              835307 non-null object
Job Family                   835307 non-null object
Job Code                     835307 non-null object
Job                          835304 non-null object
Employee Identifier          835307 non-null int64
Salaries                     835307 non-null float64
Overtime                     835307 non-null float64
Other Salaries               835307 non-null float64
Total Salary                 835307 non-null float64
Retirement                   835307 non-null float64
Health and Dental            835307 non-null float64
Other Benefits               835307 non-null float64
Total Benefits               835307 non-null float64
Total Compensation           835307 non-null float64
dtypes: float64(10), int64(3), object(9)
memory usage: 140.2+ MB
```

Below image depicts the heatmap for different features and their correlation. This helps us understand what features are closely related and what impact they will have on our model and data analysis.

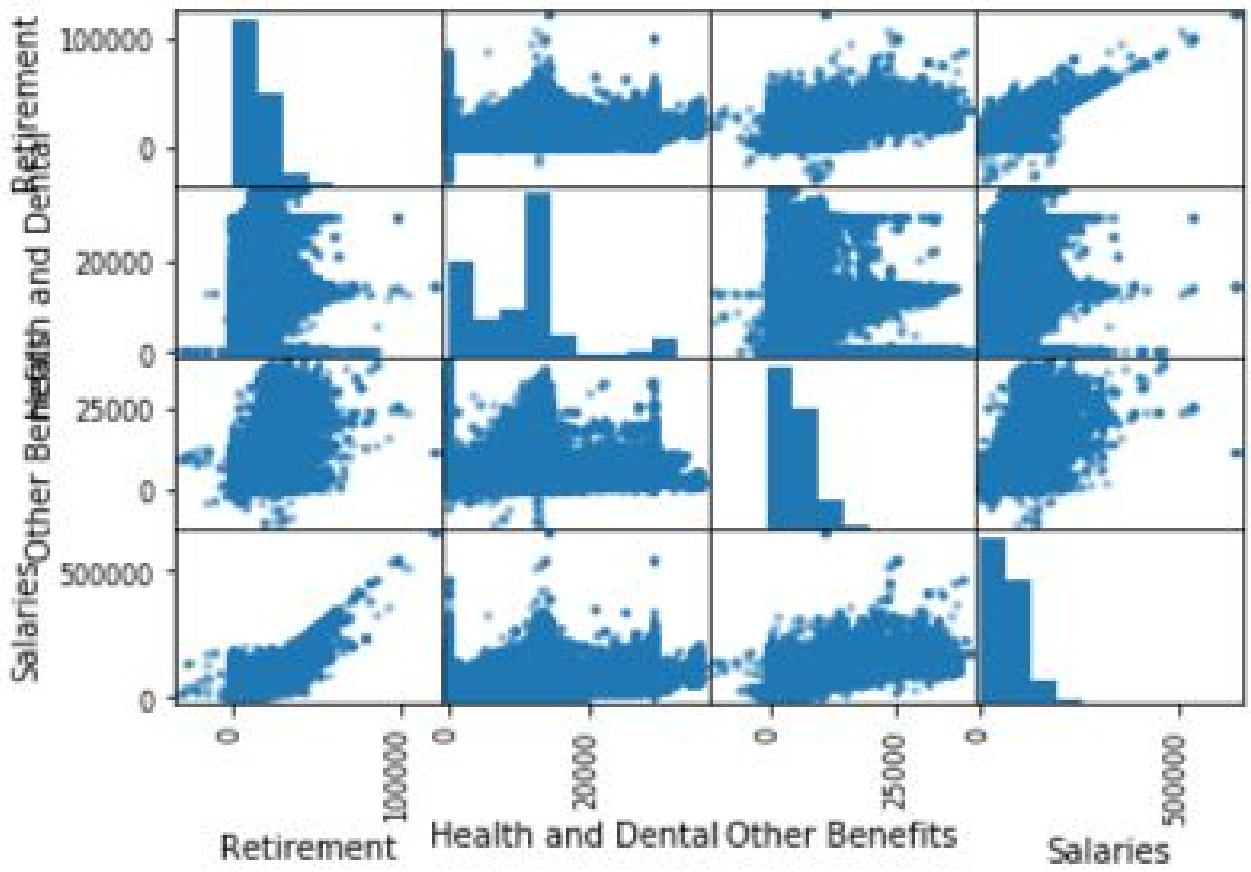


Next, we will select some closely related feature set to analyze further correlations



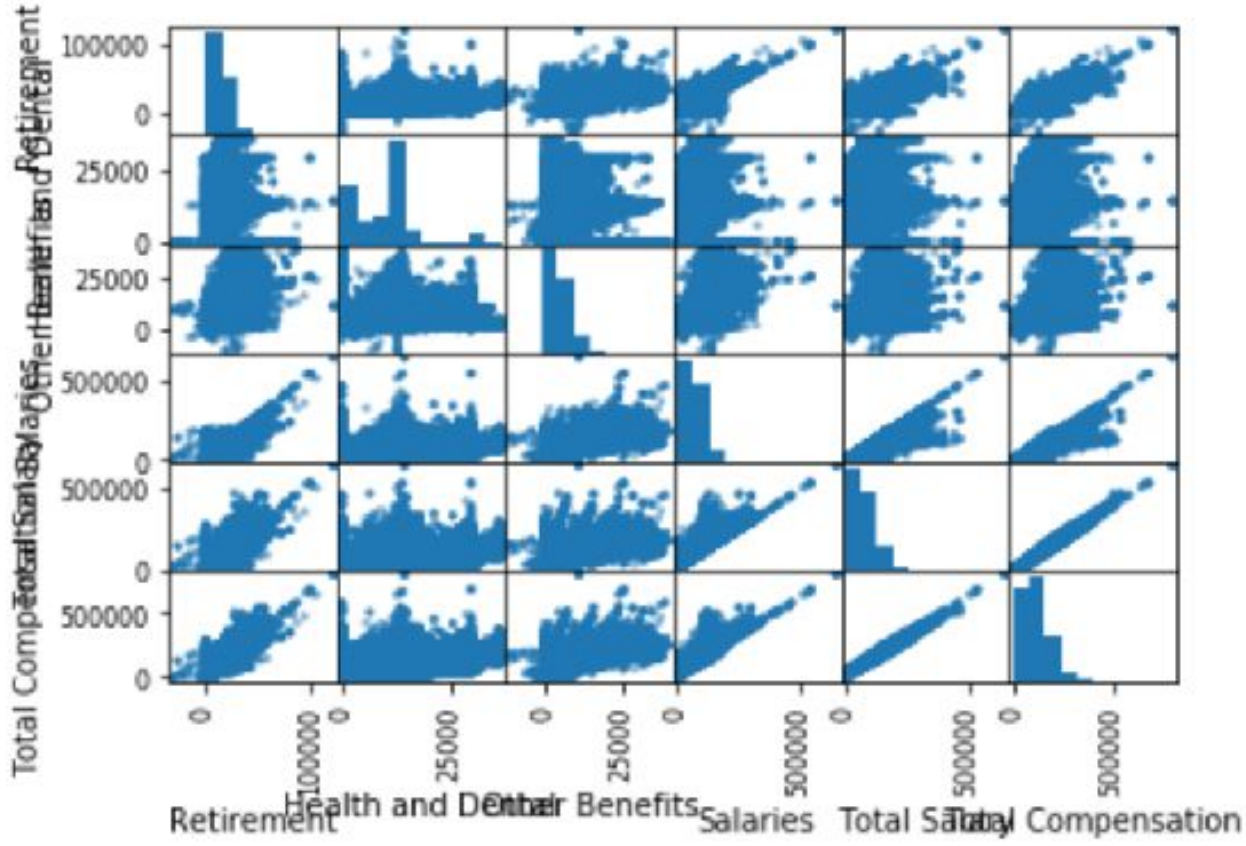
Almost all the features above are highly correlated, ~85% or more.

Next we will analyze the different benefits and see if they show the same level of correlation as the salary components

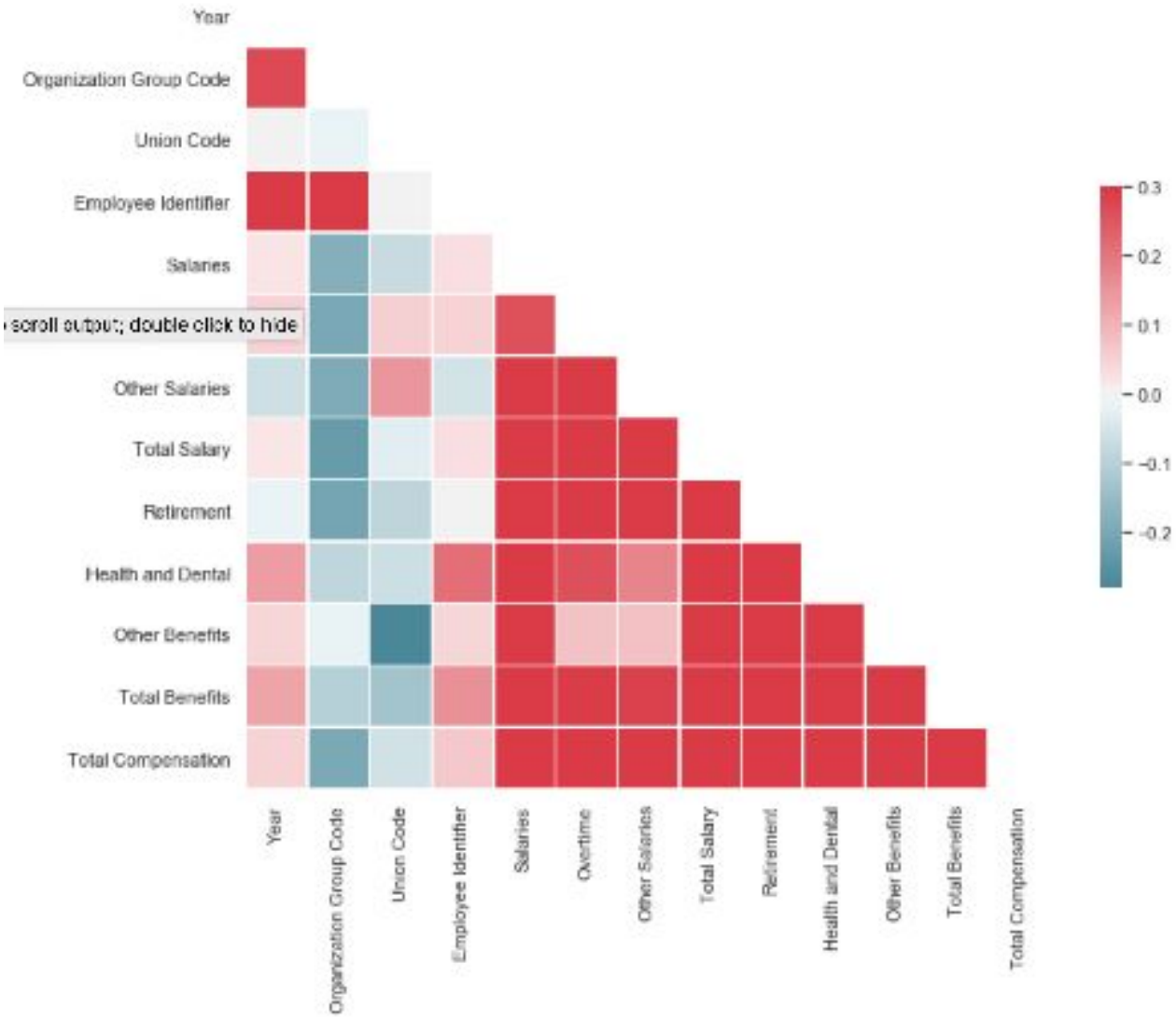


Based on above scatter matrix, not all benefits show high correlation.

Now, let's try to plot a scatter matrix for all Salary and Benefits components of compensation



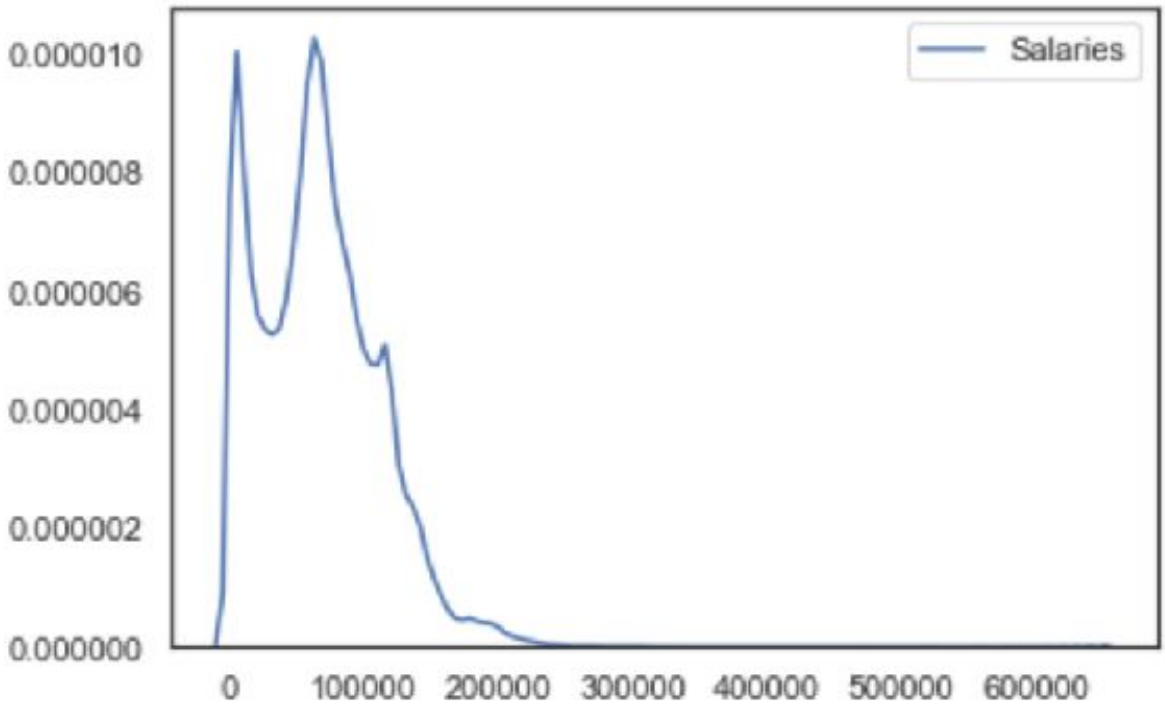
A different subplot to visualize the same correlation data in a different format using SNS library

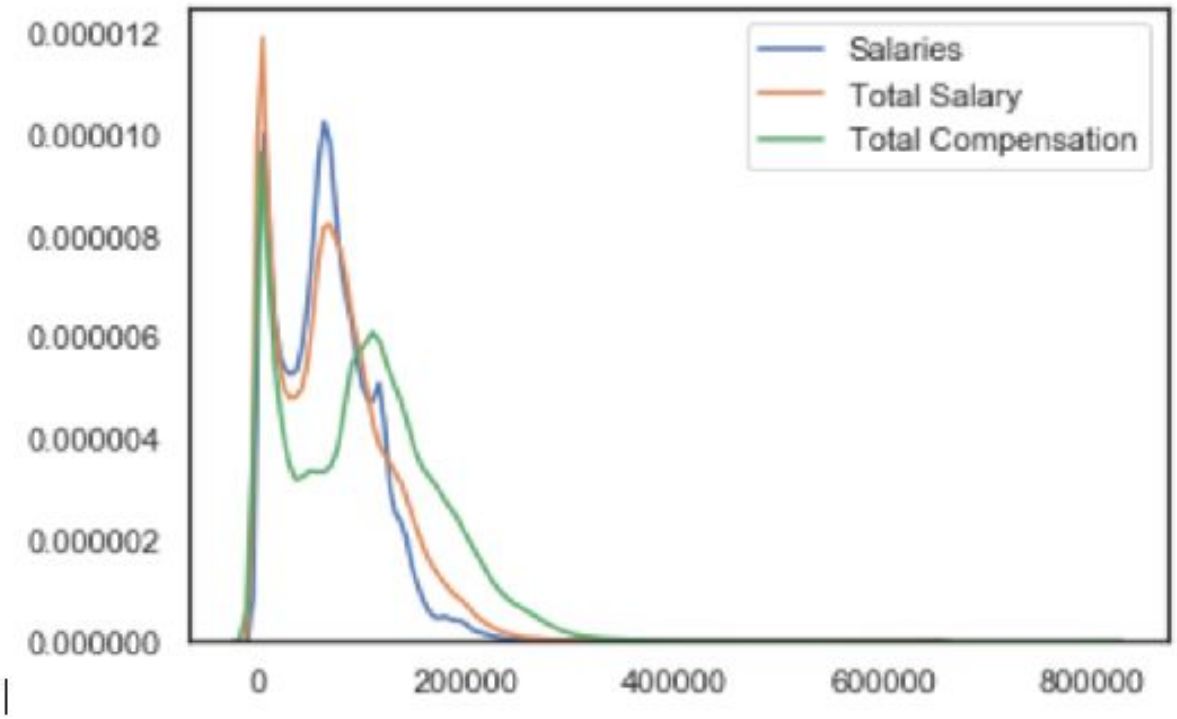


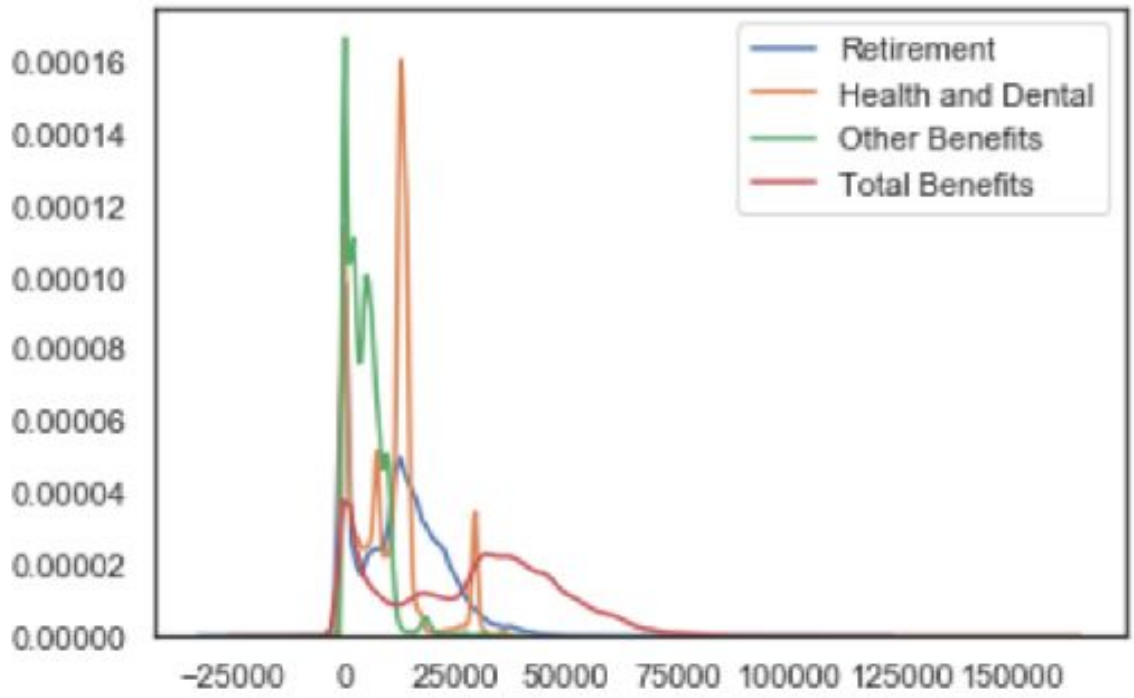
Let's do some analysis about the label data, which is our target for prediction “Salaries”

```
total_salary_description = df['Salaries'].describe()
print(total_salary_description)
```

```
count     835307.000000
mean       54096.806795
std        47845.576491
min       -68771.780000
25%         5734.080000
50%        52664.000000
75%        85601.040000
max       645739.460000
Name: Salaries, dtype: float64
```







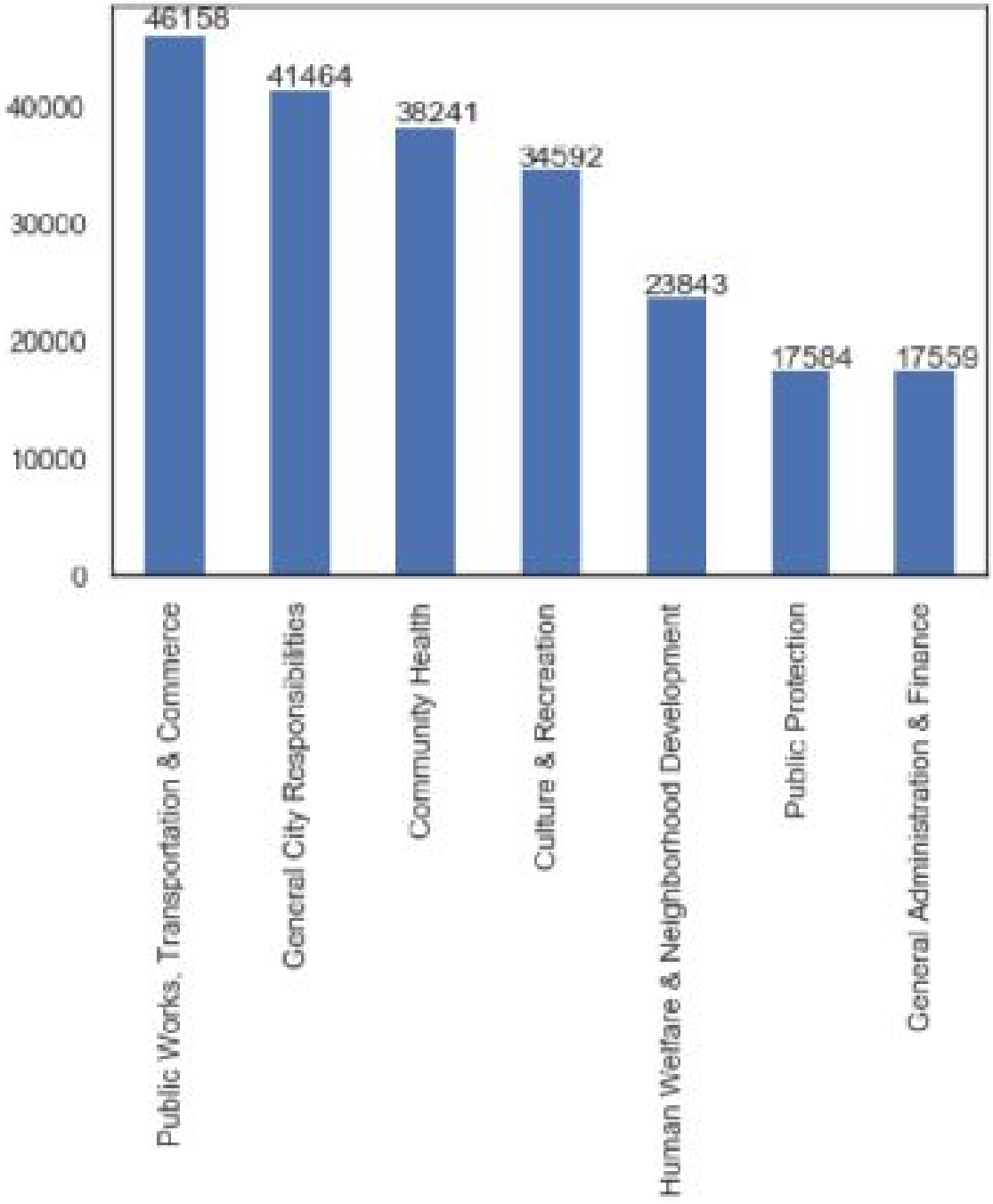
What we learn from the above plots.

1) Salary is not a perfect bell distributed curve, its positively skewed curved.
2) When converted to equal bin classes, it will create a highly imbalanced classes
3) There are extreme outliers, we cannot ignore them so we have to use different technique for handling imbalanced classes

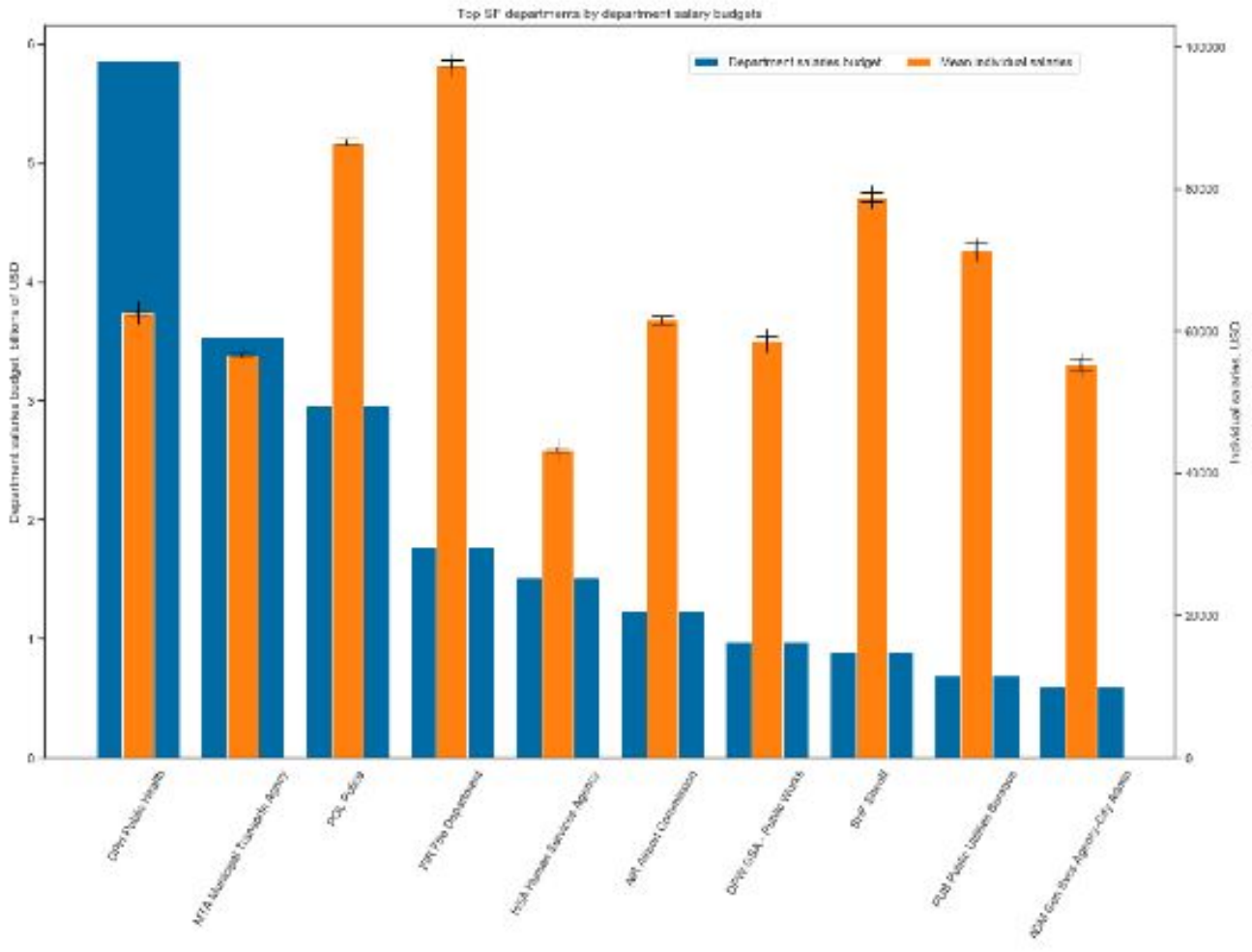
Let's do some more analysis around the mean salary

1) How many organizations we are there when we filter the data to salaries <35,000

	Organization Count	Organization %
Public Works, Transportation & Commerce	46158	0.238702
General City Responsibilities	41464	0.424254
Community Health	38241	0.281805
Culture & Recreation	34592	0.614980
Human Welfare & Neighborhood Development	23843	0.425054
Public Protection	17584	0.148515
General Administration & Finance	17559	0.312605

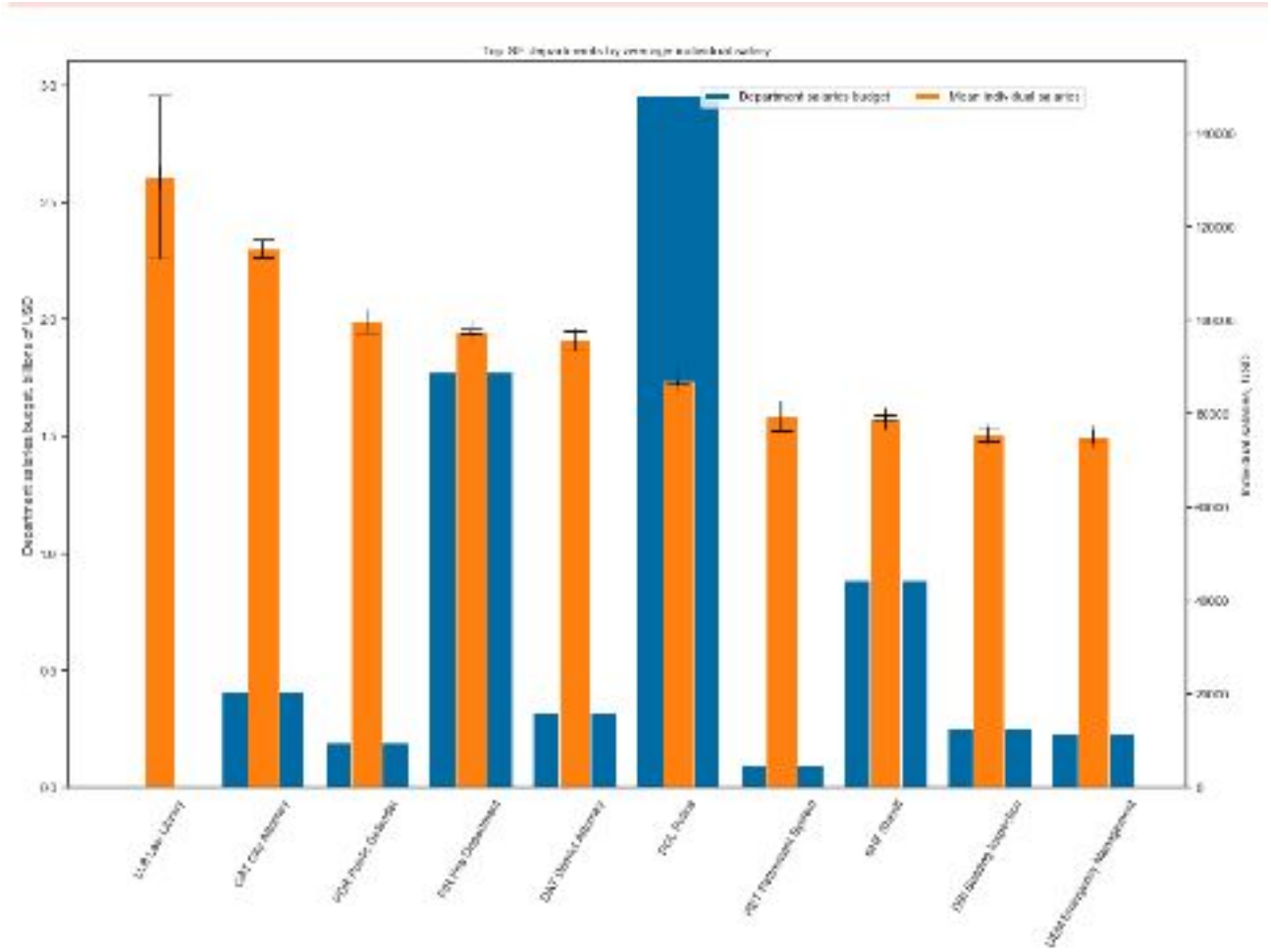


2) How many departments we have if we filter the data to salaries <35,000
3) How many jobs we have if we filter the data to salaries <35,000



Let's explore top departments by average individual salaries and by department salaries budgets.

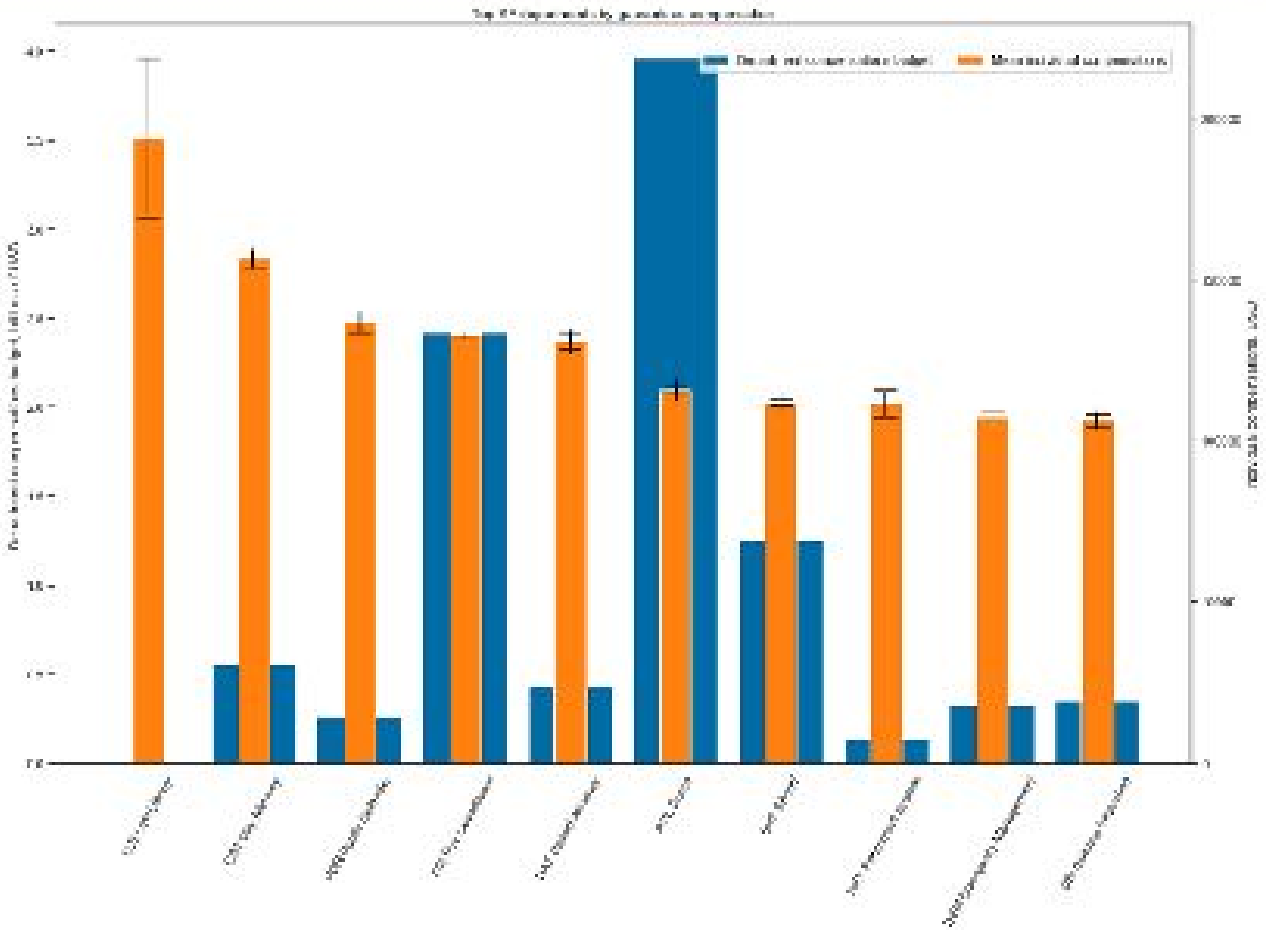
a) Top SF departments by department salary budgets



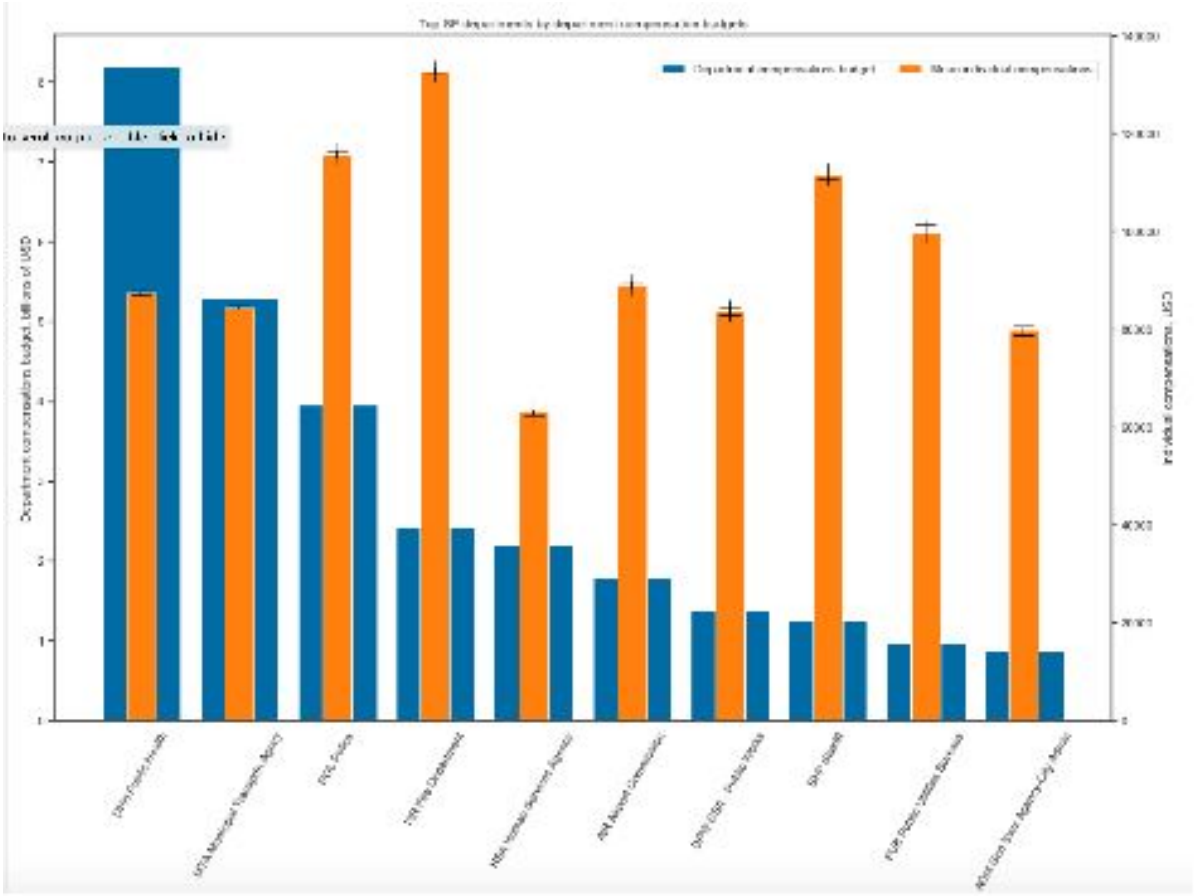
b) Top SF departments by average individual salary

Guaranteed compensations

a) Top SF departments by department compensation budgets



b) Top SF departments by guaranteed compensation

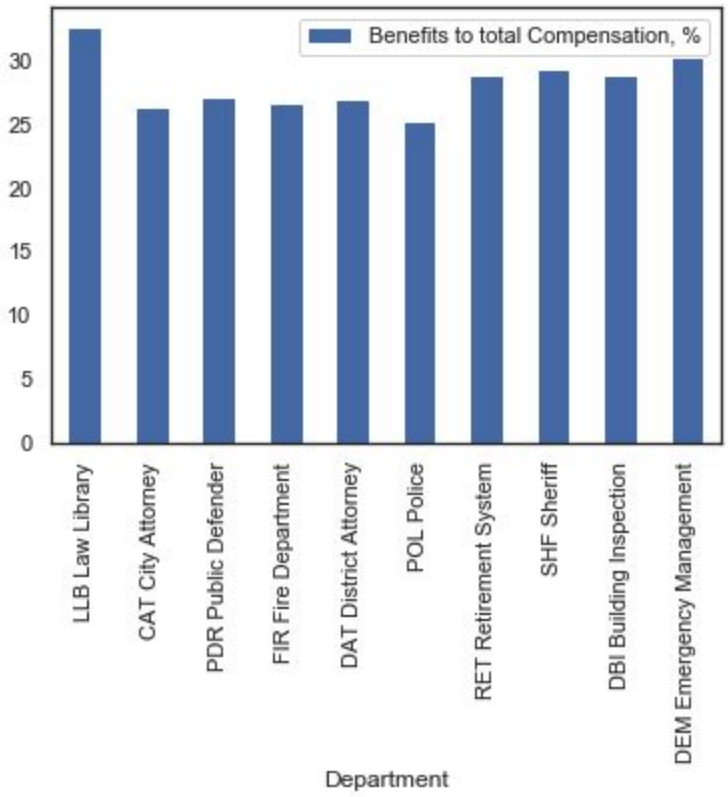


Conclusions:

1) Top lists for department budgets in both categories are identical.
2) Top lists for individual salaries and compensations are different

	Department by salary	Salary	Department by compensation	Compensation
0	LLB Law Library	130550.191053	LLB Law Library	193656.042632
1	CAT City Attorney	115187.183780	CAT City Attorney	156404.225238
2	PDR Public Defender	99710.752017	PDR Public Defender	136885.705695
3	FIR Fire Department	97532.573128	FIR Fire Department	132874.166907
4	DAT District Attorney	95727.394895	DAT District Attorney	130993.454040
5	POL Police	86626.460841	POL Police	115795.588324
6	RET Retirement System	79365.535892	SHF Sheriff	111688.977321
7	SHF Sheriff	78877.634304	RET Retirement System	111596.429520
8	DBI Building Inspection	75490.752176	DEM Emergency Management	107527.742032
9	DEM Emergency Management	75029.822082	DBI Building Inspection	106208.651973

3) Benefits to total Compensation



We can see from the plot that benefits are from ~25 to ~33% of total compensation for the top 10 departments.

Multiclass Classification

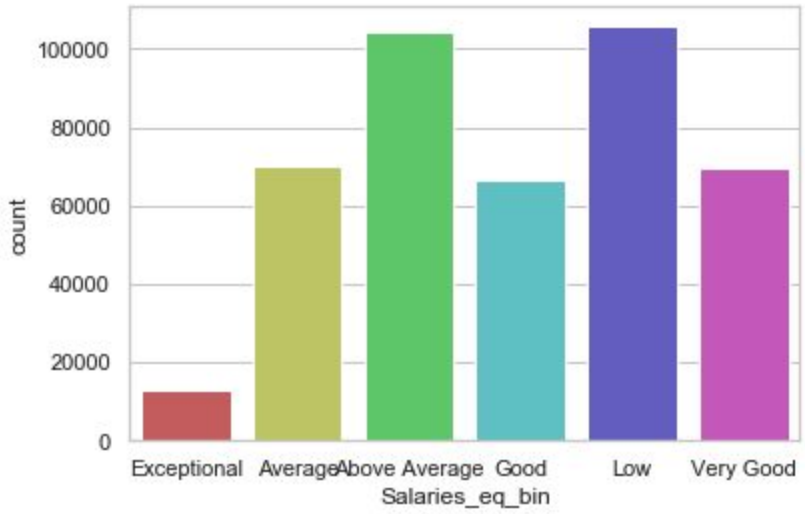
Multiclass classification on Low, Medium, Average, Above Average and High salary using different Feature selection, encoding techniques combined with various models.

Data-Preparation

1) Identify Null, Missing values and replace or drop them as necessary
2) There are compensation data which are beyond 2019. Which is a bad data and needs to be dropped.
3) There are Number of Nulls: 27300 in Department Code, but the Department name is not Null, so we will find the corresponding DepartName and fetch their code and then replace missing values.
4) We are dropping 284384 where the Department field is Null. In an ideal case, I would like to process them and have them some values which best represents the rest of data. That way we can avoid any data loss. Right now I am losing around 20% data due to dropping of the Null from Department. It's

not much of a concern for my analysis as my processing power cannot process even 400,000 odd rows.

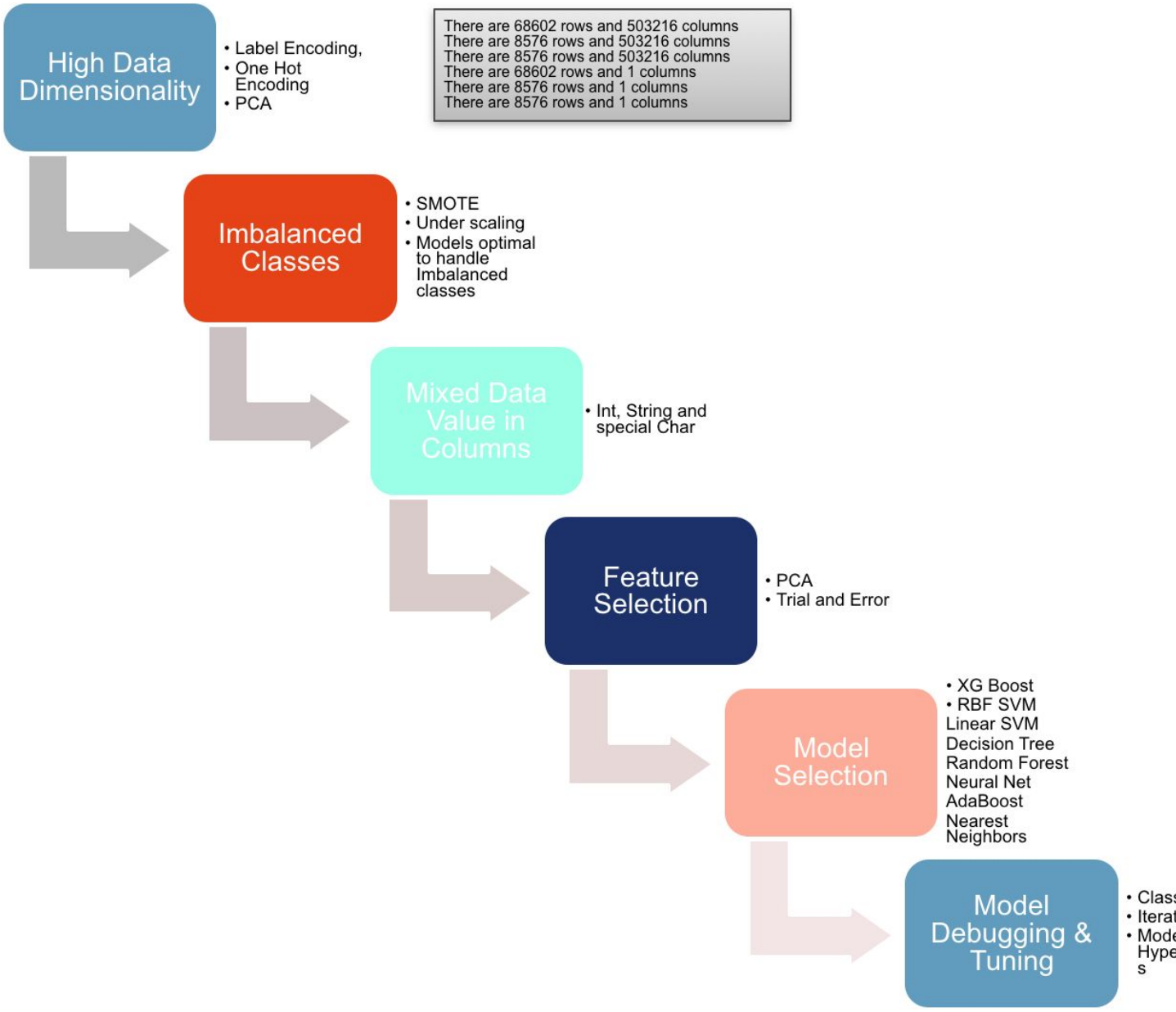
5) Convert the "Total Compensation" to a Label Column with 10 classes and then use other features to classify the test data.



6) Use **LabelEncoder** label encoder for encoding all categorical columns.
7) Then use OneHotEncoder for encoding values to convert the Ordinal values to Nominal Values

Architecture

High level flow of what we did for identifying right tools, processing and model.



We started two different codebase,

1) **Approach One**: Which uses Label encoder followed by One Hot Encoder and then uses a variety of Models with different hyper parameters to explore if certain models work better than others. Below are key summary and measurement of the solution approach.
    a) Initially I had applied OneHot encoding to all columns in Dataset and I was left with Training data with shape of 87000 X 320,000. So I applied oneHot encoding on selected columns and optimized the input data to

```
display_shape(X_train_ohe)
display_shape(X_test_ohe)
display_shape(X_val_ohe)
display_shape(y_train)
display_shape(y_test)
display_shape(y_val)
```

```
There are 85752 rows and 2498 columns
There are 10720 rows and 2498 columns
There are 10720 rows and 2498 columns
There are 85752 rows and 1 columns
There are 10720 rows and 1 columns
There are 10720 rows and 1 columns
```

    b) I applied XGBoost which is inherently designed to take care of Imbalanced dataset

```
import random
#XGBClassifier
#https://www.kaggle.com/stuarthallows/using-xgboost-with-scikit-learn
#!pip install xgboost
from xgboost import XGBClassifier as xgb
names = ["XGBClassifier"]
xgb_params = {'learning_rate': 0.25,
              'max_depth': 15,
              'subsample': 0.5,
              'colsample_bytree': 0.9,
              'objective': 'multi:softmax',
              'silent': 1,
              'n_estimators':190,
              'scale_pos_weight': 0.8,
              'gamma':6,
              'min_child_weight':8,
              "subsample": random.uniform(0.6, 0.4),
              'eta':1
             }

classifiers = [xgb(**xgb_params, seed = 10, num_boost_round=10, nfold=5,
               metrics={'error'})]
execute_model(names,classifiers)
```

    c) My training, test and cross validation and test accuracy was excellent

```
XGBClassifier
{'fit_time': array([20.56941414, 20.42533398]), 'score_time': array([0.22425604, 0.17624426]), 'test_precision_
o': array([0.95488035, 0.96120974]), 'test_recall_macro': array([0.95261916, 0.96301077])}
executing: XGBClassifier
```

	Train Time	Test Time	Cross validation Time	Train Acc	Test Acc
XGBClassifier	297.717277	0.212527	43.112667	0.995464	0.991325

d) Next I applied different variety of algorithms
   i) "Decision Tree",
   ii) "Random Forest",
   iii) "Neural Net",
   iv) "AdaBoost",
   v) "Nearest Neighbors",
   vi) "RBF SVM",
   vii) "Linear SVM"

e) Below are their performance data

	Train Time	Test Time	Cross validation Time	Train Acc	Test Acc
Decision Tree	0.655684	0.004436	0.132986	0.957365	0.950653
Random Forest	0.157632	0.016901	0.216051	0.256146	0.234049
Neural Net	183.240459	0.071461	12.702081	0.991487	0.993937
AdaBoost	8.279554	0.119834	2.657209	0.537503	0.549907
Nearest Neighbors	0.027761	49.486523	410.157930	0.994437	0.986474
RBF SVM	3730.064405	78.545982	639.264606	1.000000	0.252146
Linear SVM	6.696111	0.092639	1.287315	0.999883	0.999907

2) **Approach Two:** Which uses Labelencoder followed by Standard Scaler transformation and then apply PCA to identify number of features matching 95% variance. And then run same set of Models
   a) PCA helped identify and reduce the feature dimensionality form 2498 to 12

```
1  pca.explained_variance_ratio_
```

```
array([0.2537441 , 0.18897699, 0.08635808, 0.07704082, 0.07468936,
       0.05299173, 0.05166655, 0.04591835, 0.03952677, 0.03249153,
       0.02905687, 0.02149257])
```

```
1  X1_train = pca.transform(X1_train)
2  display_shape(X1_train)
3  X1_test = pca.transform(X1_test)
4  display_shape(X1_test)
```

```
There are 85753 rows and 12 columns
There are 21439 rows and 12 columns
```

b) Execute XGBoost

```
executing: XGBClassifier
```

	Train Time	Test Time	Train Acc	Test Acc
XGBClassifier	401.574578	1.446255	0.938883	0.915761

c) Using rest of algorithms

```
executing: RBF SVM
executing: Linear SVM
executing: Decision Tree
executing: Random Forest
executing: Neural Net
executing: AdaBoost
executing: Nearest Neighbors
```

	Train Time	Test Time	Train Acc	Test Acc
RBF SVM	820.977278	36.792579	0.952643	0.903260
Linear SVM	107.239194	17.579831	0.799132	0.805028
Decision Tree	0.799893	0.003158	0.741047	0.747050
Random Forest	0.560919	0.023653	0.536518	0.539671
Neural Net	68.885626	0.051792	0.855632	0.856803
AdaBoost	16.896211	0.211870	0.597635	0.603806
Nearest Neighbors	0.136507	3.145154	0.946194	0.891833

Conclusion:

1) PCA help reduce the feature dimensions to great extent, almost 98% of features were dropped.
2) Certain Models like XGBoost, SVC, MLPClassifier are better at handling Imbalanced datasets / classes.
3) Without PCA SVM (RBF and linear Kernel) did not completed their execution on my machine
4) I excluded few approach I tried but removed from final close like SMOTE and Undersampling I did to boost the model accuracy when I was using fewer feature sets.But Model accuracy was very low. Around 50-55 %. I was primarily trying to predict using only department, year and job data and discard all benefits and different compensation components.
5) Certain models work with same accuracy with or without PCA (may be just on my dataset, and this conclusion cannot be generalized like MLPClassifier and XGBoost

If given more time what else I would do

a) While data clean up, I dropped almost 25% of data because department, union code were null. Based on my analysis, the Departments which were null, did have department code, which for a different observation had department. So it would take quite a bit of code and processing to extract missing department form other similar observation and replace missing values. Same can be done for Union and Job
b) Due to processing limitations, I only used 50% of the available data, which was 35% of total data. I would like to use 100% of data
c) I would like to create a model which uses Unon, Job, Department, Year to make predictions and see if I can go above 80% accuracy and then predict both benefits and salary individually.

Multi Linear Regression

Using MultiLinear Regression, we're predict Total Compensation based on Salary, Health & Dental and Other Benefits.

Data-Preparation

1) Identify Null, Missing values and replace or drop them as necessary
2) Even though there was compensation data beyond 2019 – we decided not to drop them as we're already building one model having dropped them in Multi Class Classification. Since we're predicting- we could leverage the built model to predict what the Total compensation would be

3) There are Number of Nulls: 27300 in Department Code, but the Department name is not Null, so we will find the corresponding DepartName and fetch their code and then replace missing values.
4) Dropping ~284000 rows as the Department is null- but considering that the total dataset was ~780000, this isn't too much of a concern. A logical way would have been to populate some values, but considering that we still have a big dataset on which we could run our model, we chose this approach.
5) Choose Salaries, Medical and Dental, and other Benefits based on the correlation of data with Total Compensation.

ut[158]:

	Year	Organization Group Code	Union Code	Employee Identifier	Salaries	Overtime	Other Salaries	Total Salary	Retirement	Health and Dental	Other Benefits	Total Benefits	Com
Year	1.000000	0.272428	0.000267	0.820786	0.021324	0.047006	-0.063583	0.021030	-0.015807	0.135772	0.044373	0.125144	
Organization Group Code	0.272428	1.000000	-0.016097	0.337910	-0.182923	-0.199745	-0.192342	-0.226231	-0.208970	-0.086495	-0.018694	-0.103981	
Union Code	0.000267	-0.016097	1.000000	-0.000050	-0.074664	0.056852	0.150896	-0.032227	-0.089034	-0.065639	-0.280619	-0.132510	
Employee Identifier	0.820786	0.337910	-0.000050	1.000000	0.032070	0.049268	-0.057906	0.031463	0.008650	0.209840	0.043867	0.158409	
Salaries	0.021324	-0.182923	-0.074664	0.032070	1.000000	0.263628	0.350111	0.960856	0.948719	0.595185	0.721336	0.899072	
Overtime	0.047006	-0.199745	0.056852	0.049268	0.263628	1.000000	0.356168	0.497833	0.310390	0.260118	0.078670	0.294249	
Other Salaries	-0.063583	-0.192342	0.150896	-0.057906	0.350111	0.356168	1.000000	0.505904	0.376403	0.177665	0.080782	0.284654	
Total Salary	0.021030	-0.226231	-0.032227	0.031463	0.960856	0.497833	0.505904	1.000000	0.930811	0.591519	0.646309	0.873130	
Retirement	-0.015807	-0.208970	-0.089034	0.008650	0.948719	0.310390	0.376403	0.930811	1.000000	0.618693	0.672923	0.923645	
Health and Dental	0.135772	-0.086495	-0.065639	0.209840	0.595185	0.260118	0.177665	0.591519	0.618693	1.000000	0.422480	0.783721	
Other Benefits	0.044373	-0.018694	-0.280619	0.043867	0.721336	0.078670	0.080782	0.646309	0.672923	0.422480	1.000000	0.744509	
Total Benefits	0.125144	-0.103981	-0.132510	0.158409	0.899072	0.294249	0.284654	0.873130	0.923645	0.783721	0.744509	1.000000	
Total Compensation	0.049914	-0.198763	-0.060356	0.066831	0.968723	0.455202	0.458670	0.991155	0.952776	0.659041	0.689648	0.930102	
Guaranteed Compensations	0.053241	-0.162984	-0.093814	0.071043	0.991165	0.278637	0.337892	0.955169	0.961749	0.665245	0.744040	0.949197	

6) We also wanted to do something ambitious. We wanted to predict how much would be the Total compensation of every employee. However, considering that this dataset easily had close to a ~100,000 unique employees – we disbanded the idea after 5 to 6 hours of additional effort. One thing we realized during this was the RMSE and the efficiency score/R2_score were less than 10% and we had to do a lot of data cleanup that what we'd bargained for. We also came to the conclusion that's its best to pick features which have around 25-30 unique values and we shouldn't concentrate on columns which have a huge number of unique values

```
df_linearReg['Employee Identifier'].nunique()
```

```
100198
```

7) We deliberately picked 3 measures/features correlated to Total Compensation- one in the high 90s and others in the 60s and we were able to achieve a pretty decent predictions.
8) We normalized the data by dividing the numbers by 100,000
9) As followed in standard m/c learning models – we selected 80% of the data as training data and the remaining 20% as the testing data

10) The slope/coefficient and the y-intercept values are as below. Assuming that my features were zero, then the constant value/intercept would have been "[-0.02209658]

```
In [21]: print(reg_eidentifier.coef_) # co-effecient/slope
         print(reg_eidentifier.intercept_)  # constant value/Y-Intercept
         print(reg_eidentifier.score(x_train, y_train))
         print(reg_eidentifier.score(x_test, y_test))

         [[ 1.34940543  1.91321068 -0.58442695]]
         [-0.02209658]
         0.9538962761393598
         0.9548621362298281
```

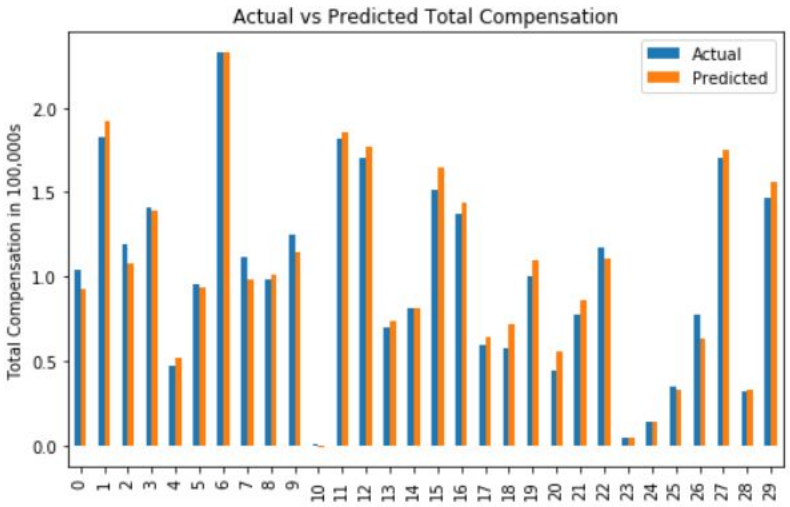
11) The predicted values also turned out to be very close to the Actual values.

```
In [24]: df_graph = pd.DataFrame({'Actual':y_test.flatten(), 'Predicted':y_pred_compensation.flatten()})
         print(df_graph)

                Actual  Predicted
         0     1.039961   0.928117
         1     1.831466   1.923939
         2     1.193484   1.075463
         3     1.408170   1.387170
         4     0.473508   0.515319
         ...        ...        ...
         85749 0.154004   0.190057
         85750 1.031555   0.814901
         85751 0.045936   0.028842
         85752 1.663983   1.813181
         85753 1.739922   1.712103

         [85754 rows x 2 columns]
```

12) We then decided to plot the first 30 records in a graph to see how close the actual and predicted values were –



Conclusion:

- The dependent variable 'y' moves in close coordination with the independent variable **x**'. Higher the value of 'x'- the 'y' changes accordingly
- There's a 'positive' linear relationship between the features chosen by us and the Total Compensation column
- We were able to achieve an accuracy of ~96%

- Linear regression as a model works best when we pick columns/features which have limited standard values. An employee_key/identifier may be a wrong value, but the Employee Department may be a better pick as there are limited numbers of them

Project Execution

The project contains three different Python Notebook file.

1) SF Employee Compensations.ipynb
2) Multiclass classification the Employee Total Compensation-V1.ipynb
3) Linear Regression-Employee Total Compensation-Final.ipynb

How to execute the project.

1) First execute the “SF Employee Compensations.ipynb” which will create a “my_df.pickle” on local system which will have dataset which is EDA complete for next steps.
2) Now the next two Analysis can be executed in any order
    a) Multiclass classification the Employee Total Compensation-V1.ipynb - This file may take around 30-50 min to execute on a machine with i7 processor and 16 GB of memory.
    b) Linear Regression-Employee Total Compensation-Final.ipynb

Final Project Requirements

You are expected to work on an original idea that is substantial. You can solve any deep learning problem of your choice. If you would like to discuss the details of your project with me, feel free to stop by before or after class.

You can form teams up to a maximum of 4 people to work on the project. Only one person in the group need to upload the groups work. All members will receive the same grade.

In the last class, each team will present their work. There will be NO extension of this deadline.

You need to prepare a report in pdf format that will explain the work that you have done including any relevant screen-shot and Python program(s) that you have written. Please see the section "Project report details" (see below) for more information.

You will be evaluated based on the following:

1. Data preparation
2. Data visualization
3. Visualizing the learning process

4. Appropriate choice of architecture

Project report details:

Please format the final project report according to the instructions below.

Introduction – In this section, you should describe the problem that you are solving, any background information that will help the instructors to understand the program

Requirements - List all the Python modules that need to be installed. If some of these modules need a specific version, please indicate so. You can also list any other conditions that are needed to run the program.

Description of the Python program. You need to describe the programs that you wrote.

Screenshots of the program output - If you are using a specific hardware and cannot obtain screenshot, please enclose appropriate photographs

Conclusion - Describe in brief the problem you solved, the program you wrote and obtained output.

Python program - If the program is one file, please add it as one of the pages in the report. If the code is large and spans more than one file, enclose a separate zip file.

Please make sure that the final report is in PDF format.

Libraries used

- import pandas as pd
- import numpy as np
- import matplotlib.pyplot as plt
- %matplotlib inline
- import os
- ## print(os.listdir("../input"))
- import seaborn as sns
- from pandas.plotting import scatter_matrix
- from mpl_toolkits.mplot3d import Axes3D
- from sklearn.preprocessing import StandardScaler, MinMaxScaler, FunctionTransformer, OneHotEncoder, KBinsDiscretizer, MaxAbsScaler, LabelEncoder
- from sklearn.pipeline import Pipeline, FeatureUnion
- from sklearn.impute import SimpleImputer
- from sklearn.model_selection import GridSearchCV, cross_val_score, train_test_split as split
- from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

- from sklearn.linear_model import LinearRegression
- from sklearn.tree import DecisionTreeRegressor
- from sklearn.ensemble import RandomForestRegressor
- from sklearn.mixture import GaussianMixture
- from time import time
- import pickle
- import numpy as np
- import matplotlib.pyplot as plt
- from matplotlib.colors import ListedColormap
- from sklearn.model_selection import train_test_split
- from sklearn.preprocessing import StandardScaler
- from sklearn.datasets import fetch_20newsgroups_vectorized
- from sklearn.neural_network import MLPClassifier
- from sklearn.neighbors import KNeighborsClassifier
- from sklearn.svm import LinearSVC, SVC
- from sklearn.gaussian_process import GaussianProcessClassifier
- from sklearn.gaussian_process.kernels import RBF
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
- from sklearn.naive_bayes import GaussianNB
- from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
- import pandas as pd
- from sklearn.model_selection import cross_validate
- from sklearn.metrics import recall_score
- from sklearn.utils import class_weight
- h = .02 # step size in the mesh