W205 Final Project

How Secure is Your Job?

Arvin Sahni, Shih Yu Chang, Madison J Myers

14 August, 2016

Introduction: Age of Job Insecurity

Though the upturn of personal freedoms, increased education and developing economies worldwide may lead to the illusion of a secure hiring climate, reality is in great contrast. In fact, the current climate for jobs can be unsettling, particularly depending on the sector in which you seek employment and the subject which you have studied. Global trends show that more and more people are investing in education as it is “believed to deliver opportunity“[1], yet most recent grads struggle to find employment unless they are in the technology or medical fields.[2] Almost all workers now “face greater job insecurity than in the past, due to increases in the practice of downsizing, layoffs and other expressions of employers’ willingness to treat labour as a variable cost of production”.[3] This change is due to rapidly changing organizations who must adapt to new global trends or fail. “The unpredictable economic situation and the tougher competitive standards have resulted in downsizing, mergers, acquisitions, and other types of structural change”.[4] This insecurity is found across the globe where employment is shifting from product- based to knowledge-based and from full employment to contract or part-time work, leading to a change in skill demand, as well as a change in job availability. As of 2009, the service sector provided 42.7 % of jobs “compared to agriculture (34.9%) and industry (22.4%) and in developed economies the service sector is even larger; for instance representing 71.5% of all EU jobs”,[5] which is a dramatic change from previous decades. As a result, job seekers and current employees are now combating an environment saturated with increased global mobility, increased competition and increased insecurity, leading to a very difficult job market to navigate.

The Solution Our Project Proposes to Have:

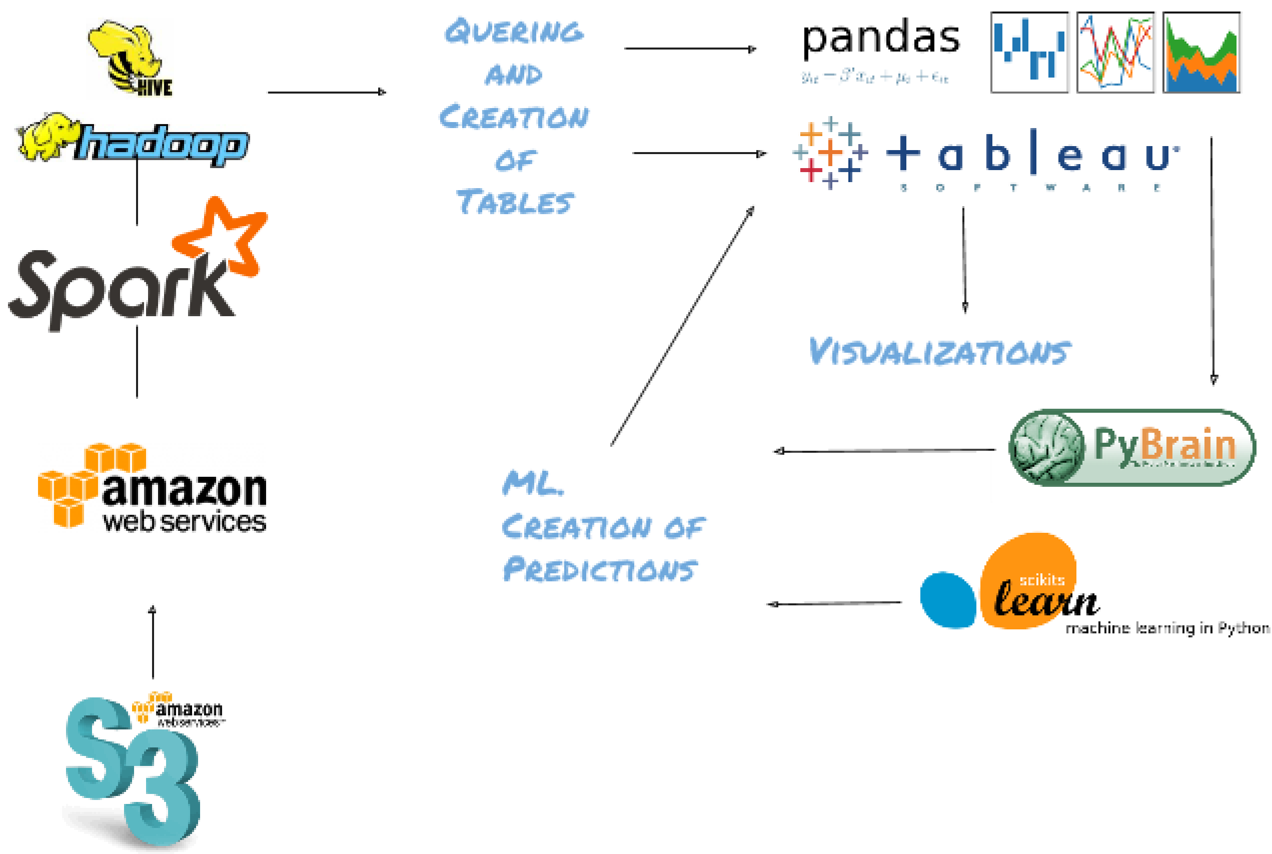
Due to the insecure employment environment, our team has decided to analyze the job market in order to help people make more informed job choices. We have broken the environment down into several factors: location, industry sector, income, expected savings, local standard of living, unemployment rate, hiring trends, retention trends, and layoff trends. Looking at these factors, we hope to not only demonstrate the factors one should consider when choosing a

---

[1] Brown, Phillip. The Opportunity Trap: Education and Employment in a Global Economy. Copyright Cardiff University, School of Social Sciences.
[2] Careerbuilder. “These are the Most In-Demand Jobs for 2016”. http://advice.careerbuilder.com/posts/these-are-the-most-indemand-jobs-for-2016. Copyright 2016, Careerbuilder, LLC.
[3] Kalleberg, Arne L. “Nonstandard Employment Relations and Labour Market Inequality: Cross-national Patterns,” in *Inequalities of the World*. Copyright 2006 by Verso.
[4] Magnus Sverke, Johnny Hellgren, Katharina Naswall. “No Security: A Meta-Analysis and Review of Job Insecurity and Its Consequences”. Copyright by Educational Publishing Foundation.
[5] Ian C. Woodward, Schon Beechler. “The Global ‘War for Talent’”. Copyright 2009 by Elsevier Inc.

field of study or a field of work, but we hope to also help predict where hiring and layoffs will take place.

The Architecture



Our System Architecture is divided into four phases: Data cleaning, initial querying and analysis, machine learning and visualizations. We initially stored our data in S3 and then launched an EC2 instance, the "ucbw205_complete_plus_postgres" AMI on AWS . We then used Hive on top of Spark within our EC2 instance on AWS. Within Hive we explored our data, conducted queries, formed tables and transformed our data on the connection between our beforementioned factors: location, industry sector, income, expected savings, local standard of living, unemployment rate, hiring trends, retention trends, and layoff trends. This gave us a better idea of what our outlooks were and the trends across different sectors and countries. These outputs were visualized using Tableau. The next step was to build a model to determine how long selected sector employees could survive if given a layoff. The next analysis, using PyBrain and Scikit, was focused on analyzing the global job market and its relation to the related country economy. The outputs were stored again in Amazon Storage Services (S3) and were visualized once again by Tableau.

Data Source & Acquisition

In an effort to use the skills we learned in class, we combined datasets from varying countries, so that we had a diversity in data as well as a substantial amount of data to work with.

Singapore:

Department of Statistics, Singapore. http://www.singstat.gov.sg

https://stats.mom.gov.sg

United States:

Bureau of Labor Statistics, United States Department of Labor. http://www.bls.gov/

Global:

Federal Reserve Bank of St. Louis, Economic Research, FRED Economic Data.

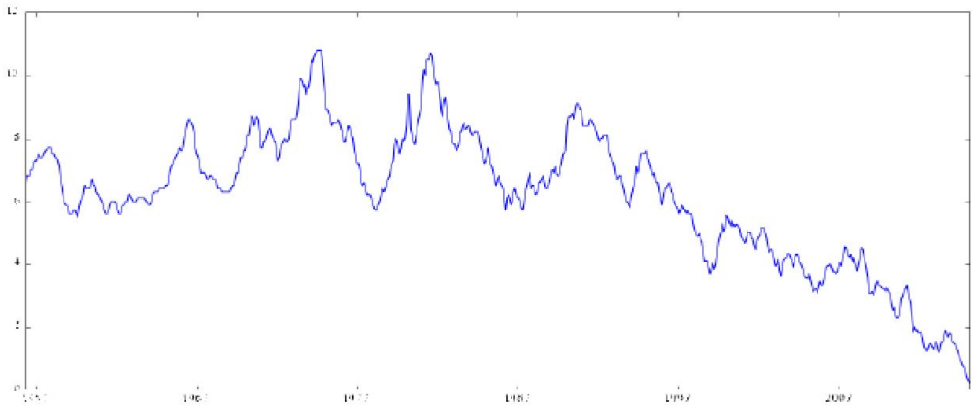
http://research.stlouisfed.org/fred2/

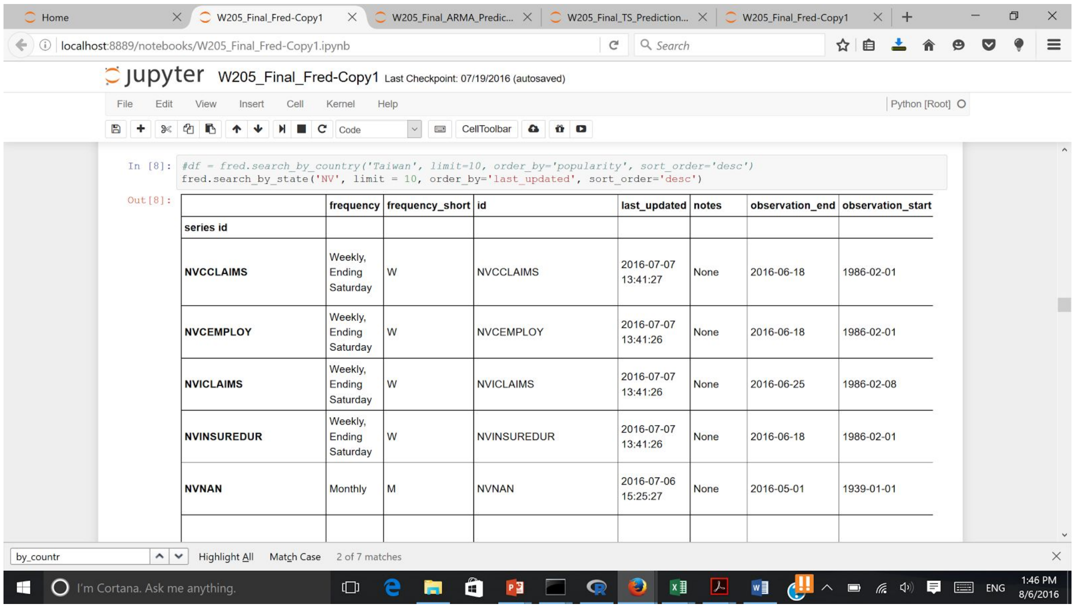
All data provided by this database are presented by time-series format and all data can be downloaded as .csv, excel, pdf, and figure files. Since there are 390,000 time-series from various economical indices contributed by 79 data sources, the data size is about 39 GB. Moreover, this database will update day-by-day. Hence, this FRED database has 3V properties as other Big Data. The FRED website provides an access KEY (need registration) for developers to write code to acquire data from the cloud. We have developed a python API to access this data from the FRED website. Different from current available APIs, we implement API by building new functions which are capable to search time series data by locations, e.g., states, countries.

Accessing Data

We stored our data in S3, as mentioned above, however please see how we attained our FRED API below:

```
In [2]: y = fred.get_series('IRLTLT01DEM156N')
        plt.plot(y)

Out[2]: [<matplotlib.lines.Line2D at 0x95a0da0>]
```

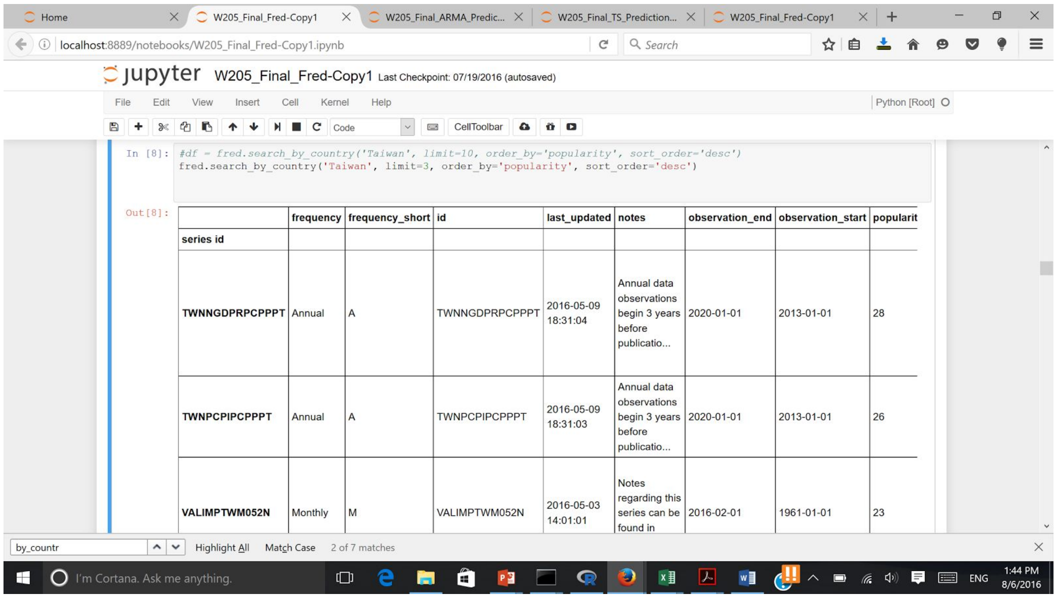




```
In [8]: #df = fred.search_by_country('Taiwan', limit=10, order_by='popularity', sort_order='desc')
        fred.search_by_state('NV', limit = 10, order_by='last_updated', sort_order='desc')
```

Out[8]:

	frequency	frequency_short	id	last_updated	notes	observation_end	observation_start
series id							
NVCCLAIMS	Weekly, Ending Saturday	W	NVCCLAIMS	2016-07-07 13:41:27	None	2016-06-18	1986-02-01
NVCEMPLOY	Weekly, Ending Saturday	W	NVCEMPLOY	2016-07-07 13:41:26	None	2016-06-18	1986-02-01
NVICLAIMS	Weekly, Ending Saturday	W	NVICLAIMS	2016-07-07 13:41:26	None	2016-06-25	1986-02-08
NVINSUREDUR	Weekly, Ending Saturday	W	NVINSUREDUR	2016-07-07 13:41:26	None	2016-06-18	1986-02-01
NVNAN	Monthly	M	NVNAN	2016-07-06 15:25:27	None	2016-05-01	1939-01-01



```
In [8]: #df = fred.search_by_country('Taiwan', limit=10, order_by='popularity', sort_order='desc')
        fred.search_by_country('Taiwan', limit=3, order_by='popularity', sort_order='desc')
```

Out[8]:

	frequency	frequency_short	id	last_updated	notes	observation_end	observation_start	popularit
series id								
TWNNGDPRPCPPPT	Annual	A	TWNNGDPRPCPPPT	2016-05-09 18:31:04	Annual data observations begin 3 years before publicatio...	2020-01-01	2013-01-01	28
TWNPCPIPCPPPT	Annual	A	TWNPCPIPCPPPT	2016-05-09 18:31:03	Annual data observations begin 3 years before publicatio...	2020-01-01	2013-01-01	26
VALIMPTWM052N	Monthly	M	VALIMPTWM052N	2016-05-03 14:01:01	Notes regarding this series can be found in	2016-02-01	1961-01-01	23

```
In [2]: %matplotlib inline
        from fredapi import Fred
        fred = Fred(api_key='48602a9ca88b2b0eefabd0904eaeb6be')
        #data = fred.get_series('SP500')
        import pandas as pd
        pd.options.display.max_colwidth = 60
        import numpy as np
        import matplotlib.pyplot as plt
        import urllib3
        import urllib
        from IPython.core.pylabtools import figsize
        figsize(20, 5)
        from matplotlib.pylab import rcParams
        rcParams['figure.figsize'] = 15, 6
        from statsmodels.tsa.stattools import adfuller

In [8]: fred.get_series_info('PCEPILFE')

Out[8]: frequency                                                              Monthly
        frequency_short                                                              M
        id                                                                    PCEPILFE
        last_updated                                            2016-06-29 07:51:07-05
        notes                        BEA Account Code: DPCCRG3  A Guide to the National Incom...
        observation_end                                                     2016-05-01
        observation_start                                                   1959-01-01
        popularity                                                                  70
        realtime_end                                                        2016-07-07
        realtime_start                                                      2016-07-07
        seasonal_adjustment                                        Seasonally Adjusted
        seasonal_adjustment_short                                                   SA
        title                        Personal Consumption Expenditures Excluding Food and Ene...
        units                                                           Index 2009=100
        units_short                                                     Index 2009=100
        dtype: object
```

Initial Queries and Results

As stated above, we used Hive SQL on top of Apache Spark to query our datasets and create tables of our analysis. A sample of our work can be seen below:

First we uploaded our data to Hive and saved it.

Example:

```
wget https://s3.amazonaws.com/w205akssg/sg/Avg_Home_ownership_by_dwelling.csv
tail -n +2 Avg_Home_ownership_by_dwelling.csv > sg1.csv
hdfs dfs -mkdir /user/w205/project/sg1
hdfs dfs -put sg1.csv /user/w205/project/sg1
```

We then created tables with the saved data.

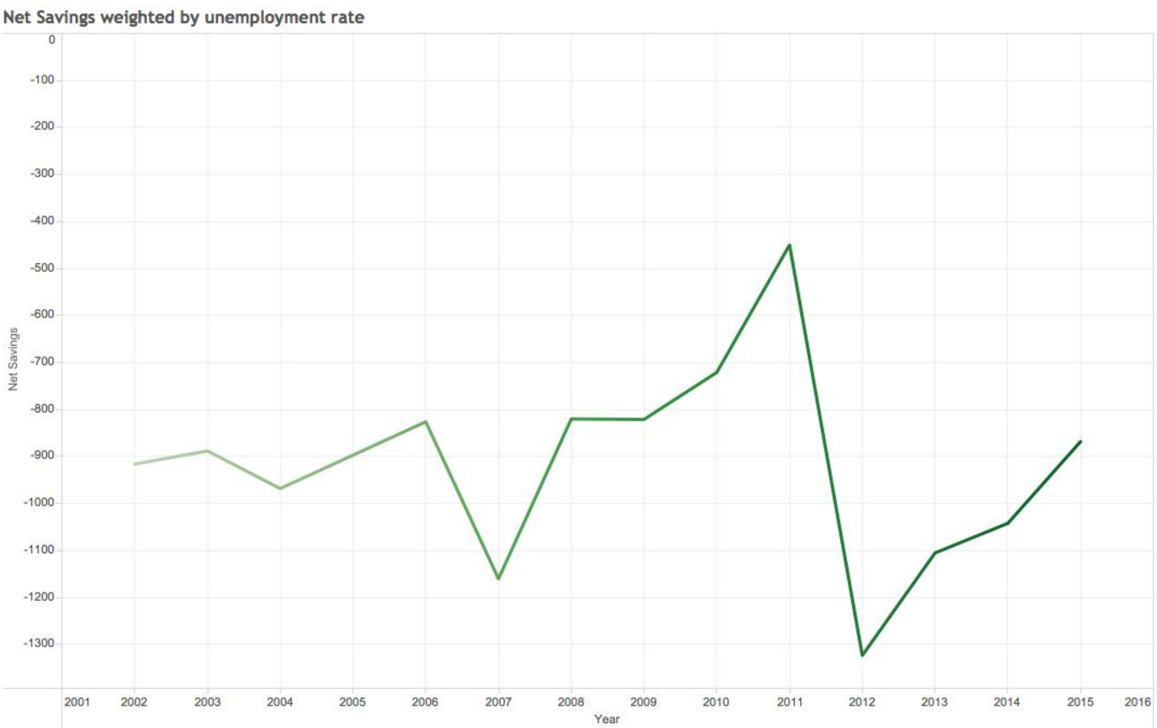
Example:

Avg Home ownership by dwelling.csv

```
DROP TABLE  avg_hdb_ownership_dwelling;
CREATE EXTERNAL TABLE avg_hdb_ownership_dwelling
(year string,
table_type string,
hdb_type string,
ownership_rate string)
ROW FORMAT SERDE ‘org.apache.hadoop.hive.serde2.OpenCSVSerde’
WITH SERDEPROPERTIES (
“separatorChar”=“,”,
“quoteChar” =‘“‘,
“escapeChar” =‘\\'
)
STORED AS TEXTFILE
LOCATION ‘/user/w205/project/sg1/‘;
```

We then created visualizations with our data by connecting to Tableau with Cloudera.

Examples:

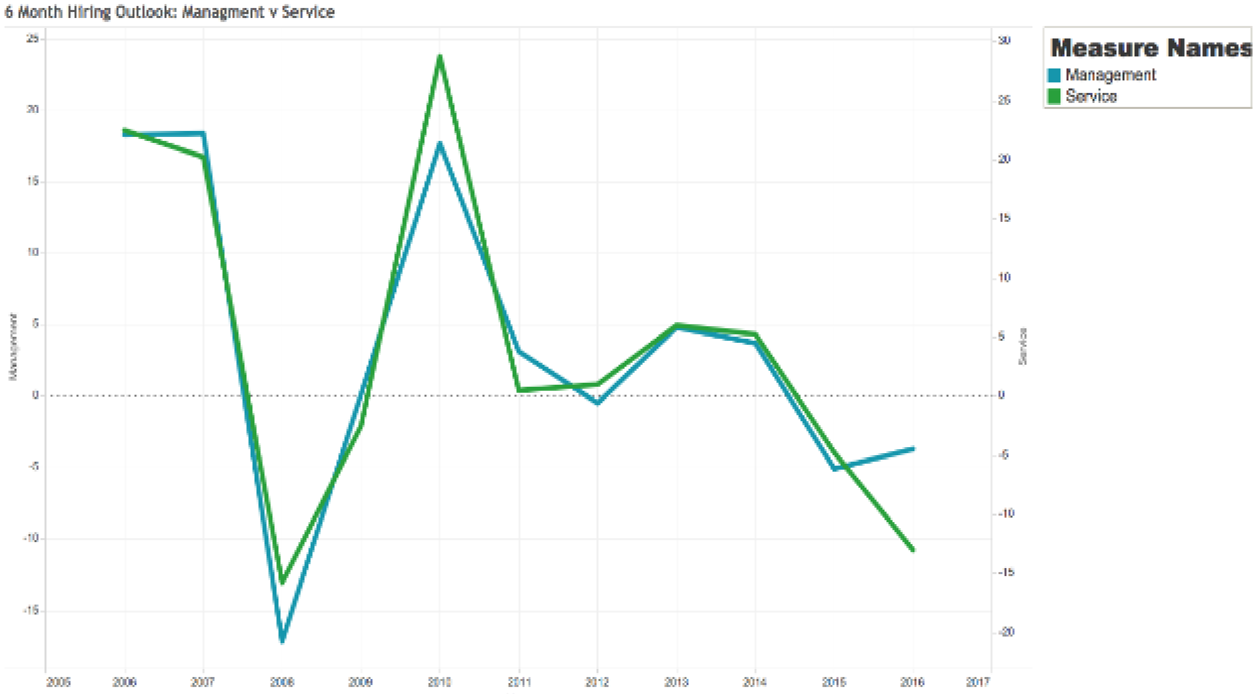


The trend of sum of Net Savings for Year. Color shows details about Year.

Net flow of employees by Sector

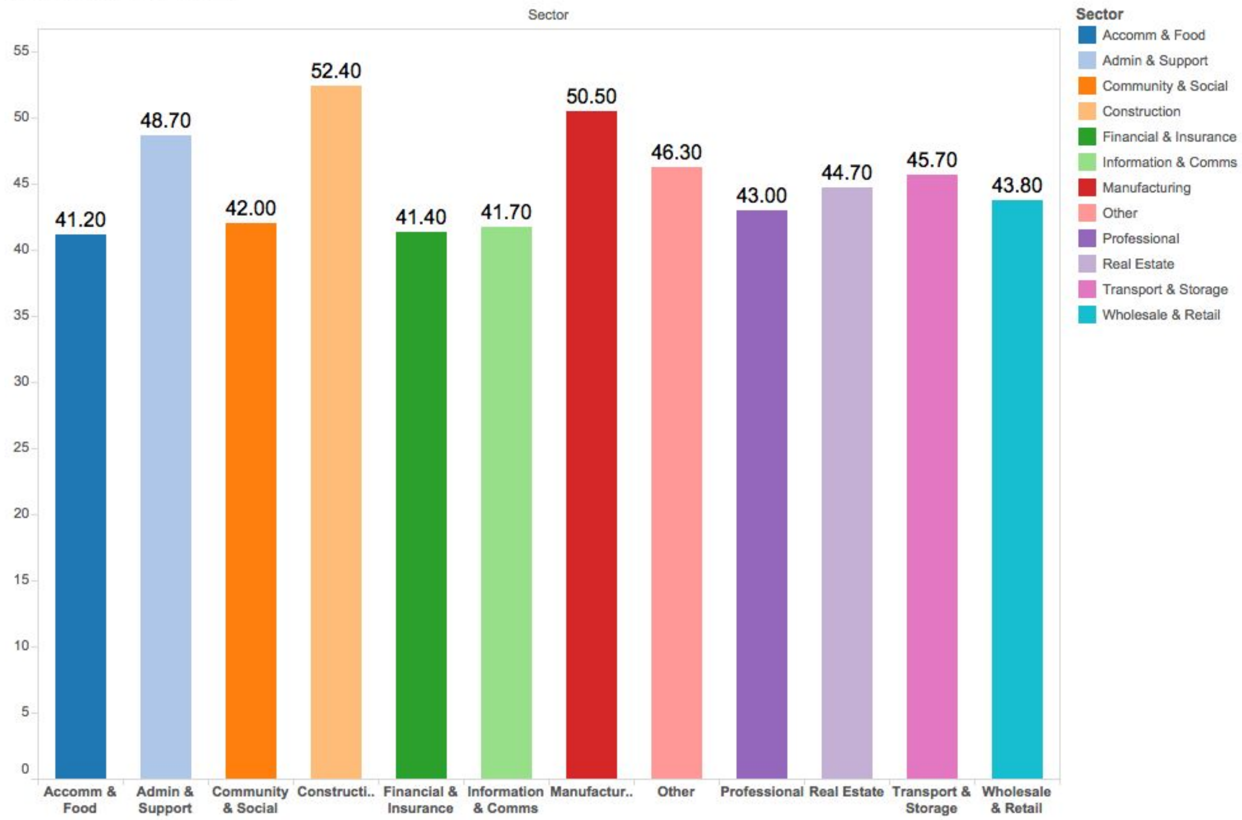
The trend of sum of Value for Year. Color shows details about Service.

6 Month Hiring Outlook: Managment v Service



The trends of Management and Service for Year. Color shows details about Management and Service.

Average Hours by Sector



Sum of Avg Weekly Hours for each Sector. Color shows details about Sector. The view is filtered on sum of Avg Weekly Hours, which ranges from 30.00 to 52.40.

In our analysis, we unsurprisingly saw a connection between times of unemployment and job market fluctuation to the recession. We also saw that most job sectors behaved in the same general pattern, though the degree seemed related to the current economic climate. Exactly during and directly after the economic crash that started in September 2008, you see all sectors drop dramatically. Thereafter it is easy to see that the Information and Communication sector has stayed on top, even though it has not been at a consistent hiring rate. Initial analysis like this has helped us forecast hiring trends and job stability.

Looking at the 2011, Global economic crisis, we can see that there is a clear slowdown in hiring within all sectors. This year also marks the absolute lows for both services and manufacturing sectors in terms of our 6 month forecast. On the other hand however, when we view the ‘changes in sector’ the manufacturing sector has managed to overcome the crisis better than the services sector. The economic crisis can also be seen to have had a major impact on the net median earnings as they have declined substantially after 2011, however recover does seem to be on the horizon.

For our study, it is also interesting to look at the value contribution attributed to employees per sector as an indicator of how valuable employees are. Some key highlights are that the Construction sector has a consistent rating through all analyzed years, the Business service sector shows the highest decline, the Finance as well as the Wholesale & Retail Trade sector show the most promising recoveries.

ARIMA Prediction Model

In statistics and econometrics, and specially in time series analysis, an autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. These models are fitted to time series data either to better understand the data or to forecast future points in the series (predicting). They are applied in prediction contexts where data show evidence of non-stationarity, where an initial differencing step can be applied to reduce the non-stationarity.

By applying the ARIMA prediction mechanism, we can get the range of duration in months for each country and state. This is based on each unique combination of asset values at the time of a layoff.. For example, X and the living cost of a particular area after some period of time, say 1 year. Then, time duration range can be determined as MIN: X / (MAX living cost in prediction), and MAX: X/(MIN living cost in prediction). For example, the following code shows that living in Los Angeles, after a layoff, the maximum duration in months one could live on savings is 4.29937748343 months and the minimum duration is 4.06629570388 months, where FRED time-series is CUURA421SEHF02. Following codes implemented our approach.

```
### Given P, Q, predictions_ARIMA by residual ### CUURA422SA0, CUURA210SA0L2 (monthly)

series = fred.get_series('CUURA421SEHF02')
series = series[np.logical_not(np.isnan(series))]
#res = sm.tsa.ARMA(series, (3, 4)).fit()
#res_error = sum(res.resid.values**2)
#print(res_error)

def pred_ARIMA(p, q, series):
```

```
    ts_log = np.log(series)
    model = ARIMA(ts_log, order=(p, 1, q))
    results_ARIMA = model.fit(disp = -1, method = 'css')
    predictions_ARIMA_diff = pd.Series(results_ARIMA.fittedvalues, copy=True)
    predictions_ARIMA_diff_cumsum = predictions_ARIMA_diff.cumsum()
    predictions_ARIMA_log = pd.Series(ts_log.ix[0], index=ts_log.index)
                                            predictions_ARIMA_log          =
predictions_ARIMA_log.add(predictions_ARIMA_diff_cumsum,fill_value=0)
    predictions_ARIMA = np.exp(predictions_ARIMA_log)
    return predictions_ARIMA


UpperLimit = 4
p = range(0, UpperLimit + 1)
q = range(0, UpperLimit + 1)

## perform model order optimization##

def opt_paras(series):

    res = sm.tsa.ARMA(series, (0, 0)).fit()
    res_error = sum(res.resid.values**2)
    opt_p = 0
    opt_q = 0

    for i in range(0, UpperLimit):
        for j in range(0, UpperLimit):
            #if i==0 and j==0:
            #    continue
            res = sm.tsa.ARMA(series, (p[i], q[j])).fit()
            new_res_error = sum(res.resid.values**2)
            if (new_res_error < res_error):
                res_error = new_res_error
                opt_p = p[i]
                opt_q = q[j]

    series_pred = pred_ARIMA(opt_p, opt_q, series)
    return opt_p, opt_q, mean_squared_error(series, series_pred)

result = opt_paras(series)
print(result)
plt.plot(series)
plt.plot(pred_ARIMA(result[0], result[1], series), color='green')

# get what you need for predicting future ahead
res = sm.tsa.ARMA(series, order=(result[0], result[1])).fit()
params = res.params
```

```
residuals = res.resid
p = res.k_ar
q = res.k_ma
k_exog = res.k_exog
k_trend = res.k_trend
steps = 12
CPI = _arma_predict_out_of_sample(params, steps, residuals, p, q, k_trend, k_exog, endog=series, exog=None, start=len(series))
print(CPI)
#### Calculate survival period ####

# initial assest of an employee get layoff
initial_assets = 20000
#CPI = [300, 400, 500, 600]

# Housing price moneth pay 1982 as basis
CA_month = 750

print("Living in Los Angeles, the maximum duration is ", (initial_assets/(CA_month * 3 * min(CPI)/100)), "months; "
    "the minimum duration is ", (initial_assets/(CA_month * 3 * max(CPI)/100)), "months." )
```

Exponential Smoothing (ES) Prediction Model

Exponential smoothing is another method to perform prediction. A basic technique for smoothing time series data, particularly for recursively applying as many as three low-pass filters with exponential window functions. Such techniques have been applied in various domains. It is an easily calculated procedure for approximately calculating or recalling some value, or for making some determination based on prior assumptions from historical observation, such as seasonality. Like any application of repeated low-pass filtering, the observed phenomenon may be an essentially random process, or it may be an orderly process with noise. The most basic ES model is the simple moving average that the past observations are weighted equally. Exponential smoothing assigns exponentially decreasing weights as the observation get older. In other words, recent observations are given relatively more weight in forecasting than the older observations.

We implemented ES as following:

```
### Implementing Exponential Smoothing###

### Test series
y = fred.get_series('CUURA311SEHA')
series = y.values
```

```
plt.plot(series)

def ini_trend(series, slen):
    sum = 0.0
    for i in range(slen):
        sum += float(series[i+slen] - series[i]) / slen
    return sum / slen

def ini_seasonal_components(series, slen):
    seasonals = {}
    season_averages = []
    n_seasons = len(series)//slen
    # compute season averages
    for j in range(n_seasons):
        season_averages.append(sum(series[slen*j:slen*j+slen])/float(slen))
    # compute initial values
    for i in range(slen):
        sum_of_vals_over_avg = 0.0
        for j in range(n_seasons):
            sum_of_vals_over_avg += series[slen*j+i]-season_averages[j]
        seasonals[i] = sum_of_vals_over_avg/n_seasons
    return seasonals

def Exponential_Smoothing(series, slen, alpha, beta, gamma, n_preds):
    result = []
    seasonals = ini_seasonal_components(series, slen)
    for i in range(len(series)+n_preds):
        if i == 0: # initial values
            smooth = series[0]
            trend = ini_trend(series, slen)
            result.append(series[0])
            continue
        if i >= len(series): # we are forecasting
            m = len(series) - i + 1
            result.append((smooth + m*trend) + seasonals[i%slen])
        else:
            val = series[i]
            # Additive ES
                          last_smooth, smooth = smooth, alpha*(val-seasonals[i%slen]) +
(1-alpha)*(smooth+trend)
            trend = beta * (smooth-last_smooth) + (1-beta)*trend
            seasonals[i%slen] = gamma*(val-smooth) + (1-gamma)*seasonals[i%slen]
            result.append(smooth+trend+seasonals[i%slen])
    return result


future_windows = 36
```

```
series_pred = Exponential_Smoothing(series, 12, opt_results[0], opt_results[1], opt_results[2], future_windows)
#plt.plot(series_pred)
#print(mean_squared_error(series, series_pred))
print(series_pred[len(series) + 1])
print(series_pred[len(series) + future_windows - 1])
```

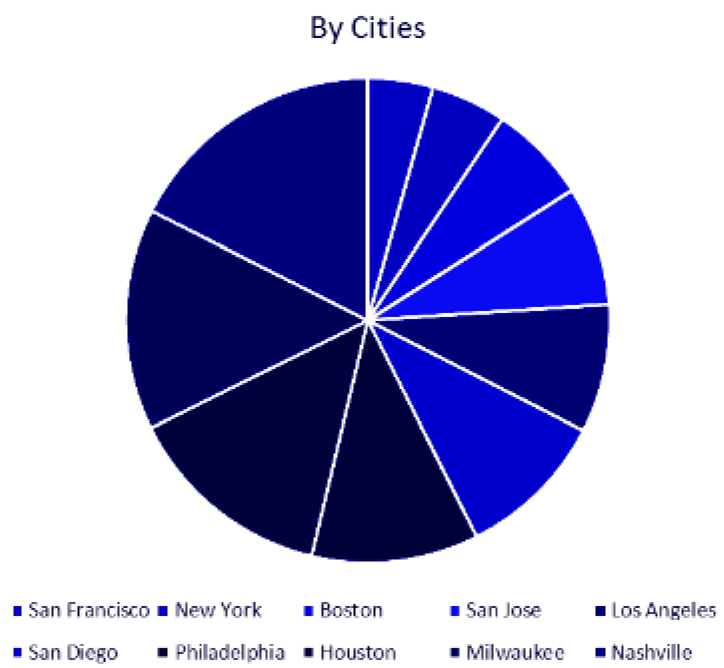
Because ES prediction errors are sensitive for model weights selections, we also provide following codes to get optimal ES prediction model by minimizing mean-squared-error of past historical data.

```
### Optimizing Exponential Smoothing Parameters ###
series = y.values
#print(mean_squared_error(series, series_pred))
# alpha, beta, gamma

resolution = 20
alpha = np.linspace(0.0, 1.0, resolution)
beta = np.linspace(0.0, 1.0, resolution)
gamma = np.linspace(0.0, 1.0, resolution)

def opt_paras(series, slen):
    Min_MSE = mean_squared_error(series, Exponential_Smoothing(series, slen, 0.001, 0.001, 0.001, 0.001, 0))
    opt_alpha = 0.001
    opt_beta = 0.001
    opt_gamma = 0.001
    for i in range(0, resolution):
        for j in range(0, resolution):
            for k in range(0, resolution):
                new_MSE = mean_squared_error(series, Exponential_Smoothing(series, slen, alpha[i], beta[j], gamma[k], 0))
                if (new_MSE < Min_MSE):
                    Min_MSE = new_MSE
                    opt_alpha = alpha[i]
                    opt_beta = beta[j]
                    opt_gamma = gamma[k]
    series_pred = Exponential_Smoothing(series, slen, opt_alpha, opt_beta, opt_gamma, 0)
    return opt_alpha, opt_beta, opt_gamma, mean_squared_error(series, series_pred)

opt_results = opt_paras(series, 12)
print(opt_results)


future_windows = 36
series_pred = Exponential_Smoothing(series, 12, opt_results[0], opt_results[1], opt_results[2], future_windows)
```

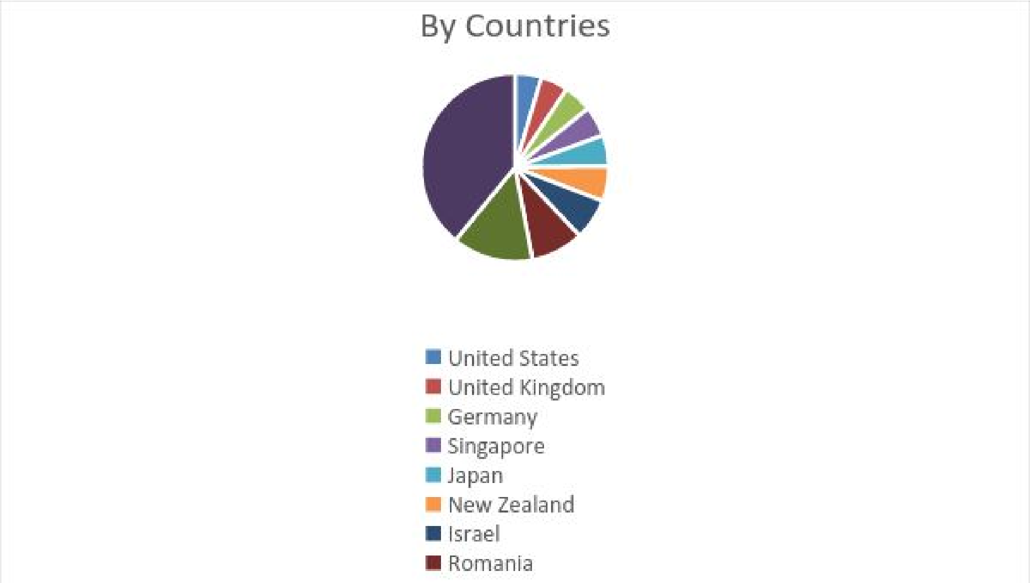
```
#plt.plot(series_pred)
#print(mean_squared_error(series, series_pred))
print(series_pred[len(series) + 1])
print(series_pred[len(series) + future_windows - 1])
```

More details codes can be reviewed at W205_Final_ARMA_Prediction-Copy1.ipynb and W205_Final_TS_Prediction-Copy1.ipynb

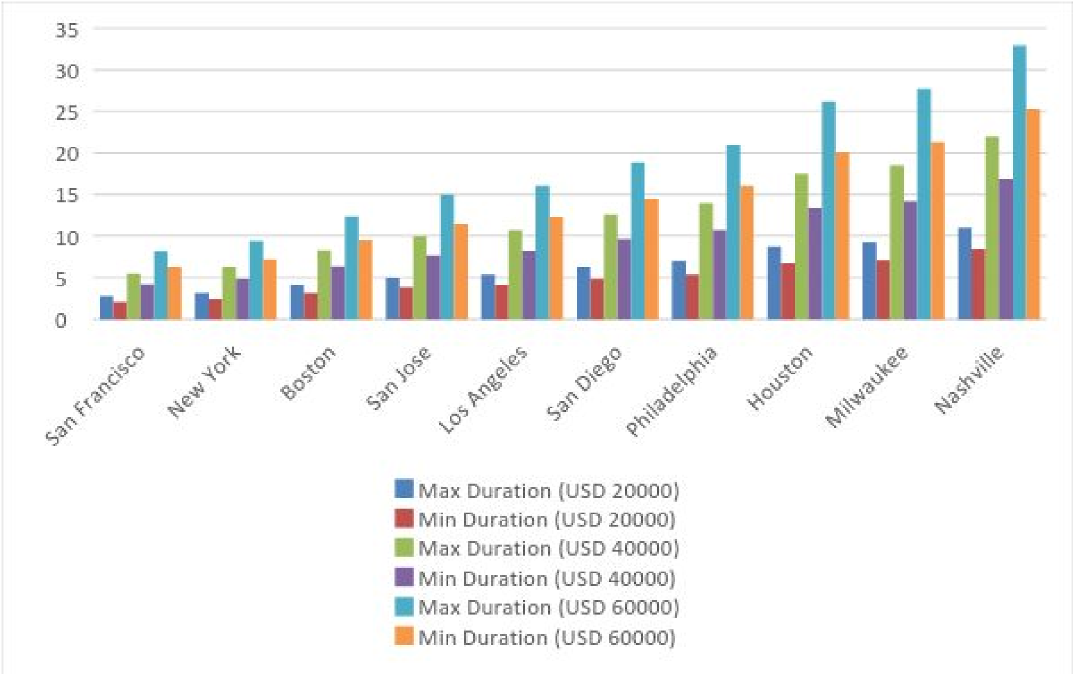
The following pie-chart shows the compared duration of months one can last on savings by city:



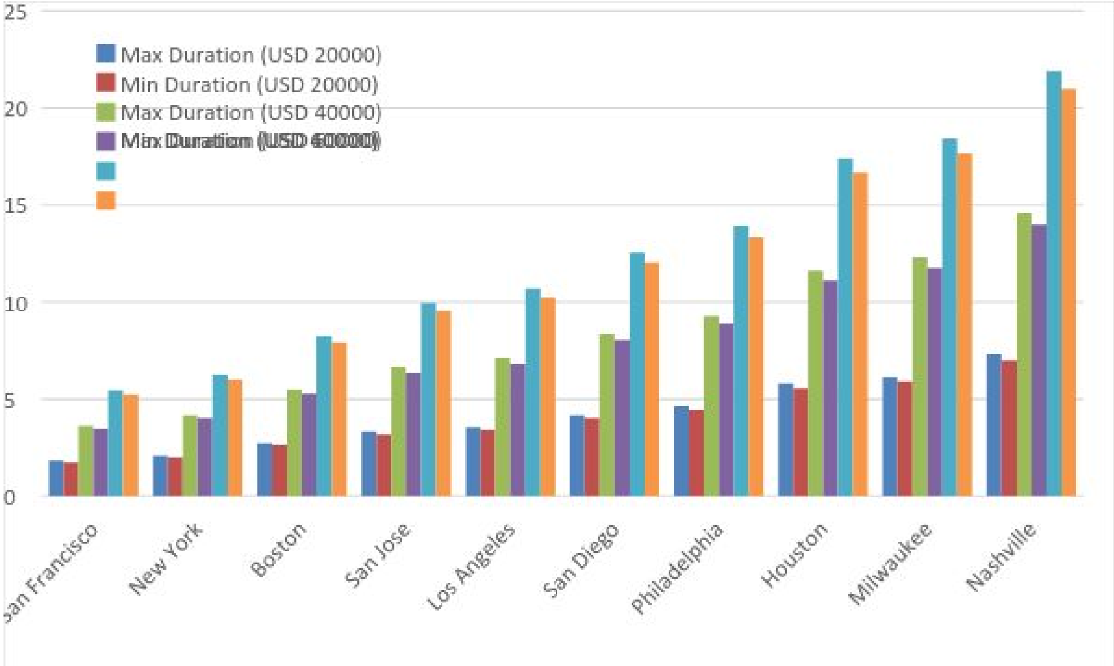
The following pie-chart shows the proportion of time can survive after a layoff by country:



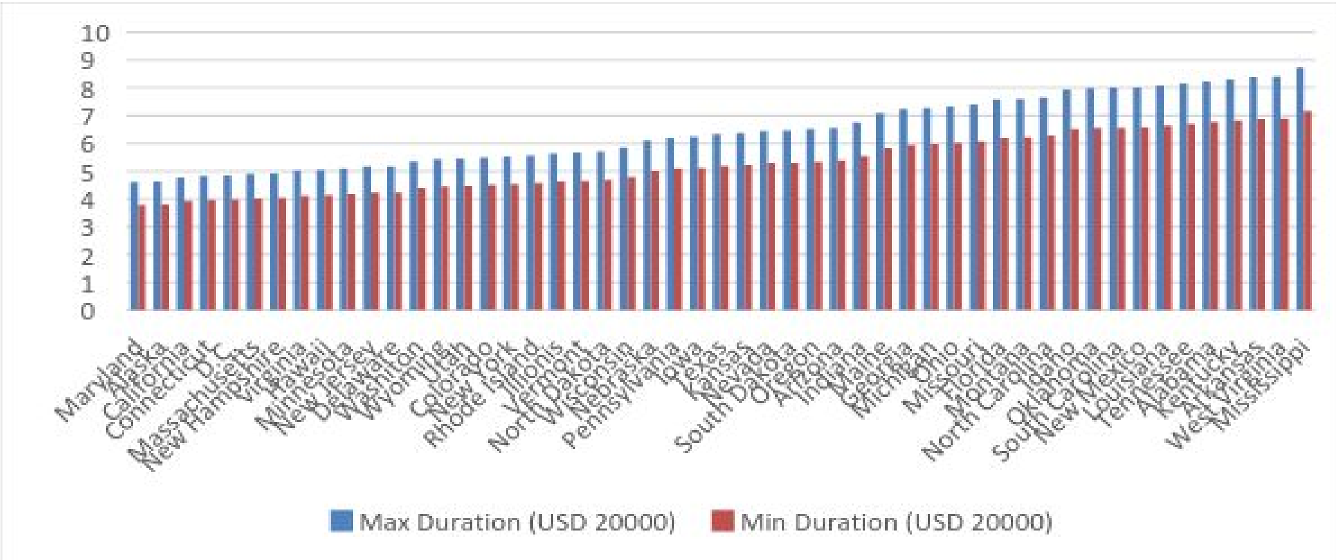
The following bar-chart shows the survival months after layoff by cities predicted by ARIMA model:



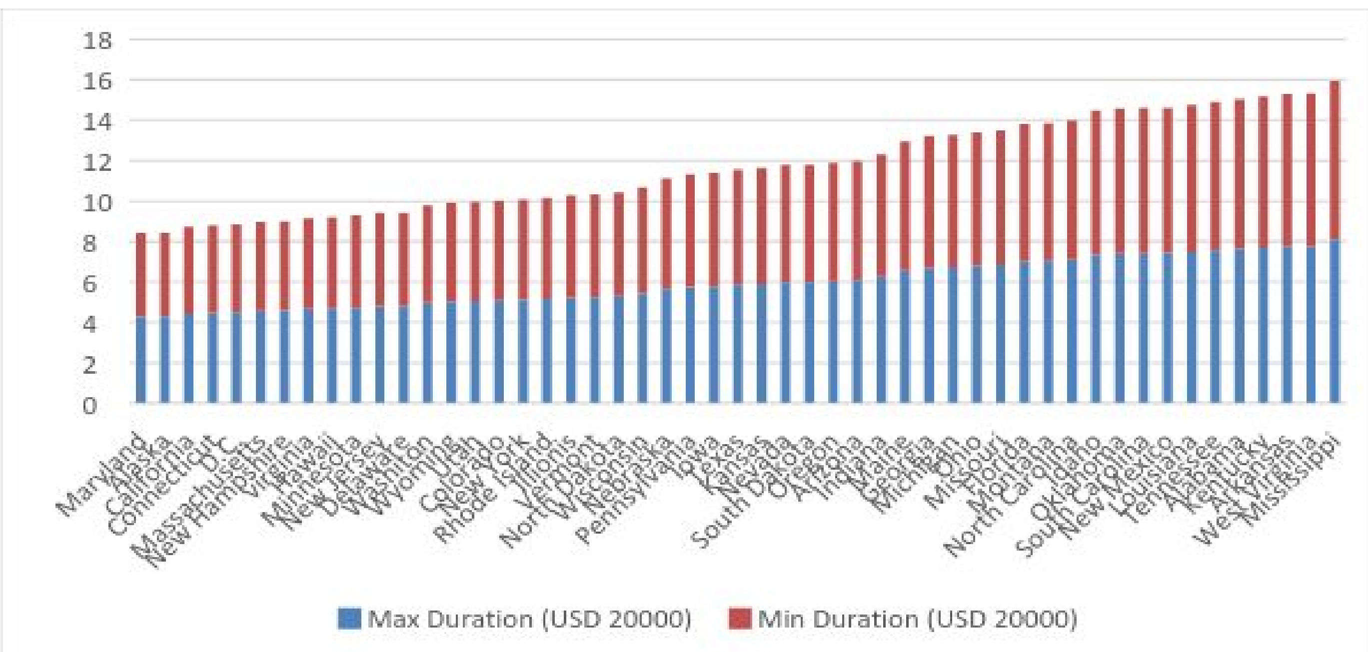
The following bar-chart shows the survival months after layoff by cities predicted by exponential smooth model:



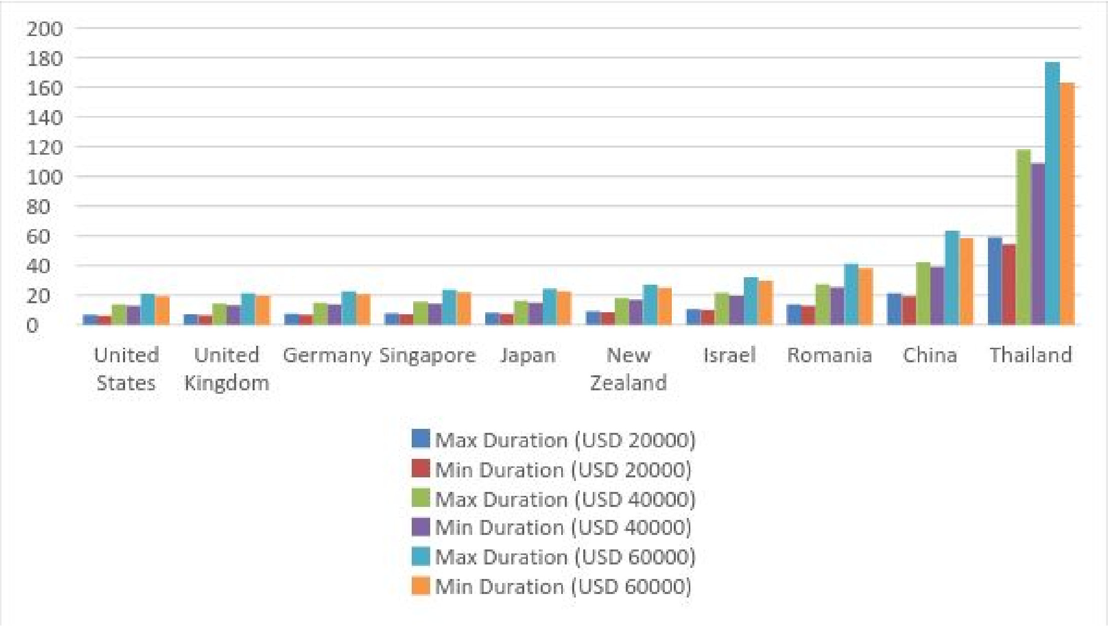
The following bar-chart shows the survival months after layoff by states predicted by ARIMA model if initial individual asset value is US 20000:



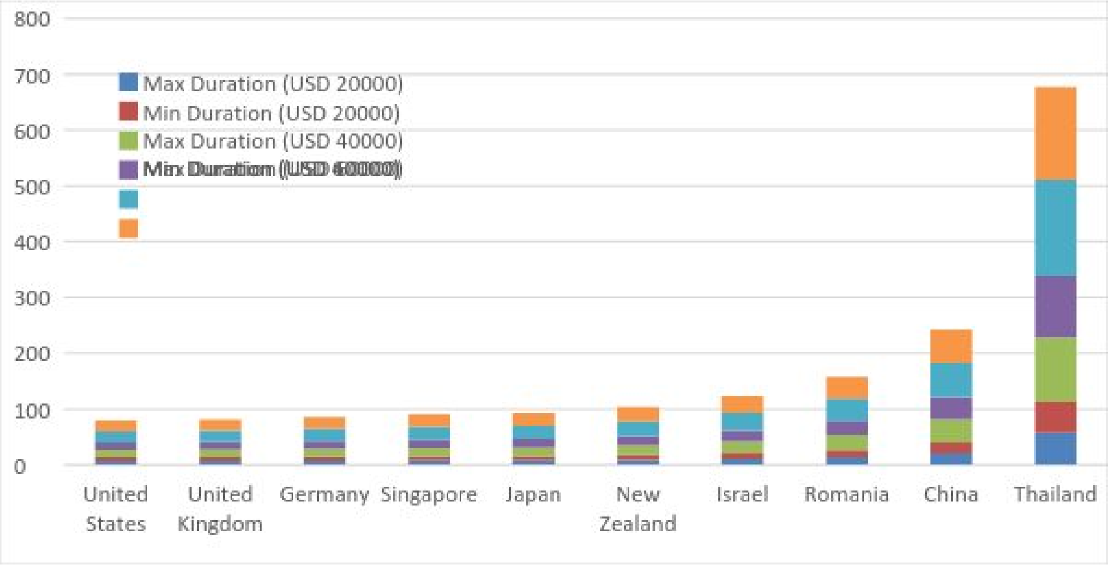
The following bar-chart shows the survival months after layoff by states predicted by ES model given initial individual asset value is US 20000:



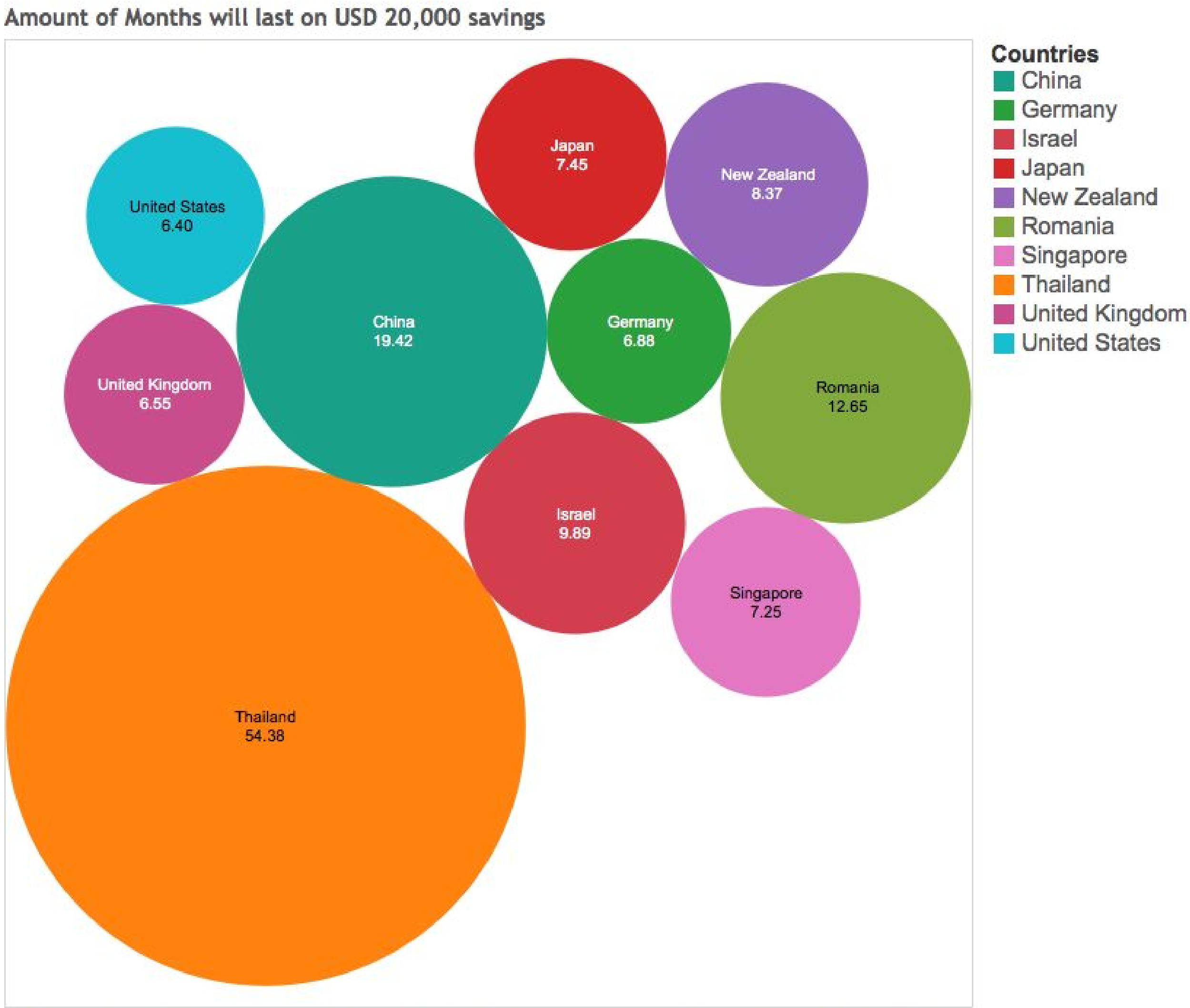
The following bar-chart shows the survival months after layoff by countries predicted by ARIMA model.



The following bar-chart shows the survival months after layoff by countries predicted by ES model.



Global Perspective:



Countries and sum of Predicted minimum months (USD 20000 in assets). Color shows details about Countries. Size shows sum of Predicted minimum months (USD 20000 in assets). The marks are labeled by Countries and sum of Predicted minimum months (USD 20000 in assets). The view is filtered on Countries, which keeps 10 of 10 members.

Minimum Wage and Unemployment

This part aims to model and quantify the relationship between minimum wage and unemployment rate. This economic relationship is studied based on the data provided by FRED for the United States from 2002 to 2013. This time period is broken up into three phases: before-recession (2002 Jan. to 2006 Dec.), recession (2007 Jan. to 2009 Dec.), and after-recession (2010 Jan. to 2013 Dec.). For the multiple regression models, it was concluded that minimum wage had a significant effect on unemployment when the economy was during the recession phase. R code (LinearRegresUnemployment.R) implemented in our study about the effect of minimum wage to unemployment, and education level vs minimum wage relationship.

Multiple Regression Model

The multiple regressions for each time period are defined below as following equation.

$unemployment$ rate $= \beta 0 + \beta 1\ minWage + \beta 2\ EduLevel + \beta 3\ GDP$

The multiple regression model for the before-recession and after-recession showed an insignificance for the minimum wage and unemployment. It also can be seen from the table that the coefficients for GDP per capita are very small and seem almost insignificant and yet that is not exactly the case for GDP per capita; consider the fact that GDP per capita is in tens of thousands of dollars. As a result, even an extremely small coefficient, $\beta 3$, can still have some effect on the unemployment level. The percentage of high school graduates percentage, education level, is shown to be significant at every phases.

For the recession model, minimum wage was significant at the 15% level and education level once again was significant at all levels. GDP per capita was not significant during this period of time. The minimum wage increased significance for the recession model as compared to the pre-recession model can be explained by the booming of the economy. During the recession, the unemployment rate greatly increased which can be attributed to the poor economic structure at the time. This affected on the model and made it seem like the increase in unemployment rate was correlated to minimum wage. To assess whether this model is appropriate the F-test was completed again, resulting in joint significance at the low level. Consequently, this model is adequate and minimum wage is critical. However, the recession could have also made the significance between unemployment rate and minimum wage more apparent. When firms are faced with an economic hardship, the higher minimum wages makes it harder to recruit labor. Economic reasoning would be used to explain this causation as opposed to recession causing bias in the model. Table below is about multiple regression model results summarized from following R-code summary at each economic phase.

Table : Multiple Regression Model Results

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Ind. Variables	Before-Recession	Recession	After-Recession
Min_Wage	NA	0.164	NA
Hgih_School_Percentage	0.647 ***	0.815 ***	0.442 ***
G_D_P	~0 ***	~0	~0 ***
Intercept	4.857 ***	2.496	13.872 ***
R-square	0.908	0.992	0.956

Before-regression (Jan. 2002 to Dec. 2006):

```
lm(formula = pre_Unemployment_Rate ~ pre_Min_Wage + pre_Hgih_School_Percentage +
    pre_G_D_P, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.29278 -0.14961  0.00555  0.09804  0.32181

Coefficients: (1 not defined because of singularities)
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)                4.857e+00  9.576e-01   5.071 4.48e-06 ***
pre_Min_Wage                      NA         NA      NA       NA
pre_Hgih_School_Percentage  6.473e-01  9.729e-02   6.653 1.20e-08 ***
pre_G_D_P                  -2.149e-04  4.119e-05  -5.218 2.63e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1611 on 57 degrees of freedom
Multiple R-squared:  0.9079,    Adjusted R-squared:  0.9046
F-statistic: 280.9 on 2 and 57 DF,  p-value: < 2.2e-16
```

Within-regression (Jan. 2007 to Dec. 2009):

```
lm(formula = inn_Unemployment_Rate ~ inn_Min_Wage + inn_Hgih_School_Percentage +
    inn_G_D_P, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-0.3452 -0.1519 -0.0241  0.1265  0.3770

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)                2.4963806  4.6388070   0.538    0.594
inn_Min_Wage               0.1638688  0.1264321   1.296    0.204
inn_Hgih_School_Percentage 0.8147887  0.0388150  20.992   <2e-16 ***
inn_G_D_P                 -0.0001597  0.0003353  -0.476    0.637
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.197 on 32 degrees of freedom
Multiple R-squared:  0.9919,    Adjusted R-squared:  0.9912
F-statistic:  1309 on 3 and 32 DF,  p-value: < 2.2e-16
```

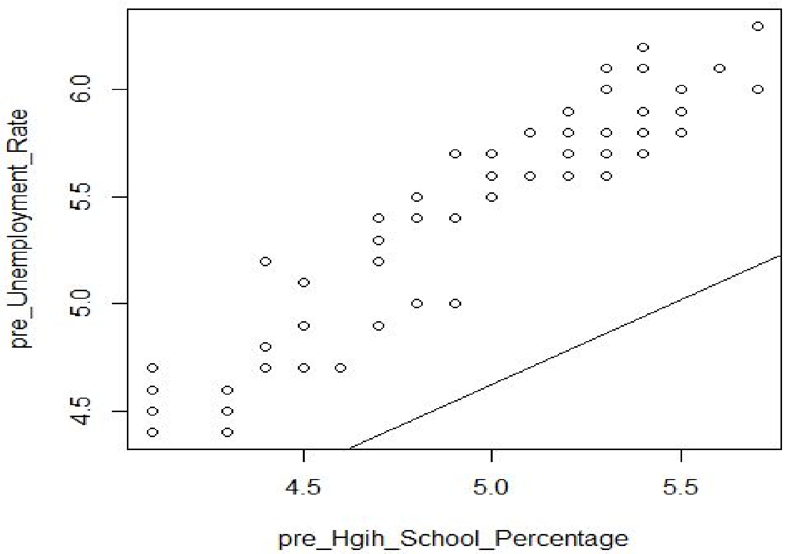
After-regression (Jan. 2010 to Dec. 2013):

```
lm(formula = aft_Unemployment_Rate ~ aft_Min_Wage + aft_Hgih_School_Percentage +
    aft_G_D_P, data = df)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.47352 -0.09843 -0.00910  0.13919  0.34008 

Coefficients: (1 not defined because of singularities)
                            Estimate Std. Error t value Pr(>|t|)    
(Intercept)               13.8722351  2.8354133   4.892 1.31e-05 ***
aft_Min_Wage                      NA         NA      NA       NA    
aft_Hgih_School_Percentage  0.4421829  0.0786007   5.626 1.12e-06 ***
aft_G_D_P                  -0.0005878  0.0001368  -4.298 9.12e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1913 on 45 degrees of freedom
Multiple R-squared:  0.9557,	Adjusted R-squared:  0.9537 
F-statistic: 484.8 on 2 and 45 DF,  p-value: < 2.2e-16
```

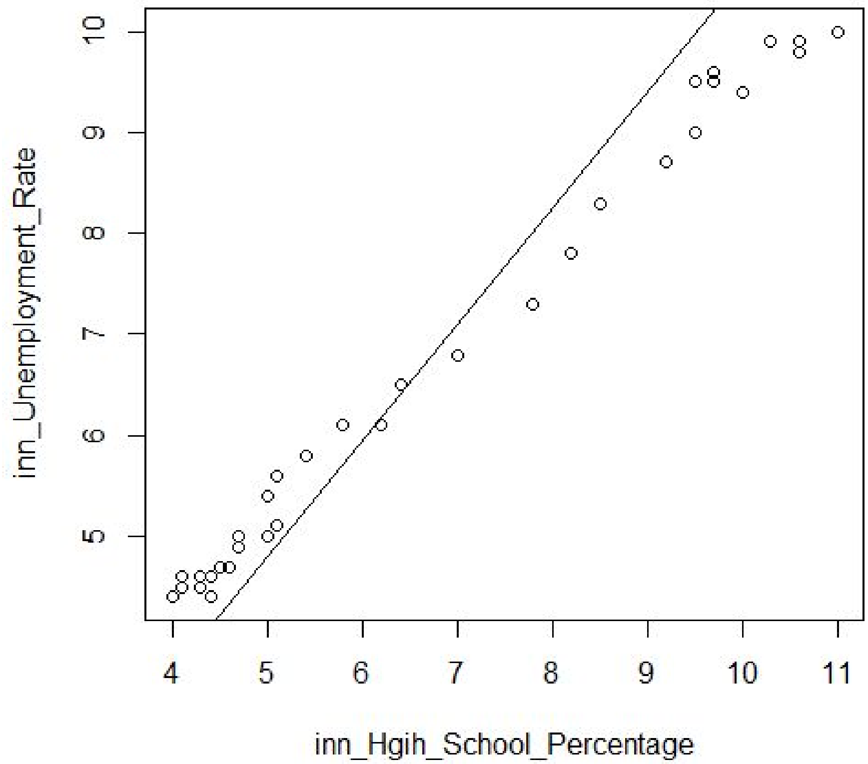
Education and Unemployment

We generated education level and unemployment regression plot for three phases, before-regression, within-regression, and after-regression. According to following figures, there is a highly correlated relationship between education level (percentage of people get high school degree) and unemployment. But, surprisingly, we observed that higher percentage of high-school educated people in USA will have higher unemployment rate. This quite different from general sense.

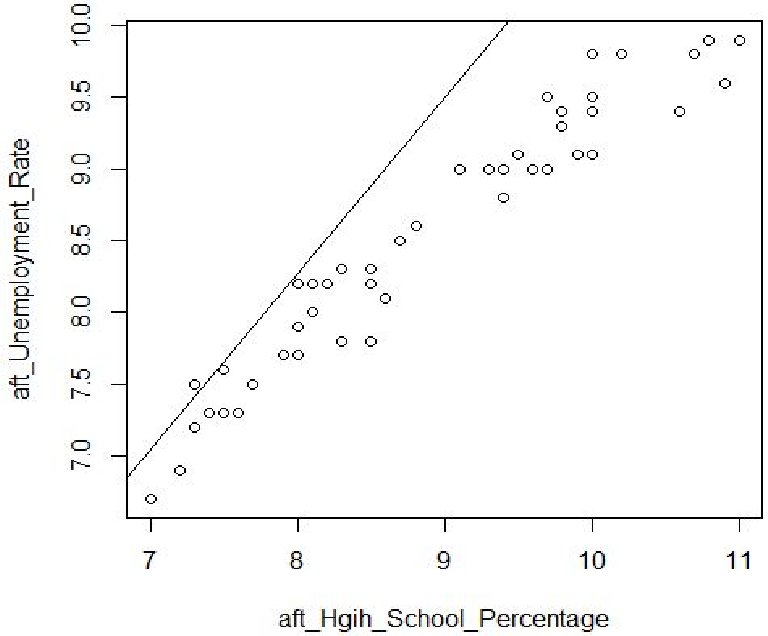
Before-regression (Jan. 2002 to Dec. 2006):



Within-regression (Jan. 2007 to Dec. 2009):



After-regression (Jan. 2010 to Dec. 2013):



Skills Applied From W205 Class

- Used an AWS EC2 Instance.
- Attached a volume.
- Used many types of datasets.
- Used Spark.
- Used Hive to:
    - Query.
    - Make tables.
- Developed an API (Required by FRED).
- Connected to Tableau using Cloudera.
- Created Visualizations.

Limitations and Future Extensions:

As data continues to be created and as we bring more data in to analyze changing economies, changing job competition levels, education, politics and more, we will need to scale out. We can do this by continuing to work in Spark, while adding storage on AWS.

Future extensions on this project would be to incorporate many more factors that have impact on the job market. We would need to look at where interest is for the job seeker as well as for the employers. Not only will we need to better forecast what job is in demand, but we will need to consider what will happen to people who are already employed in certain fields. Whether it be how they would fare laid off, or the likelihood they will keep their jobs given their education, skill level and experience or what might happen given social, political and economic circumstances will add greatly to this project. Given that job insecurity is high across the globe, this app could be high in demand.

Find our project on Github:
https://github.com/MadisonJMyers/UCB_Projects/tree/master/W205_FinalProject