In [ ]:

```python
# Imports
from pyspark.sql import SparkSession
from pyspark.sql import Row
from pyspark.sql.functions import lit
from pyspark.sql.types import StringType
from pyspark.sql.types import DoubleType
from pyspark.sql.functions import year
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import MinMaxScaler
from pyspark.sql.functions import col
from pyspark.sql.functions import concat
import pyspark.sql.functions as fn
from scipy import stats
import pandas as pd
import numpy as np
import shapefile
import six
```

In [10]:

```python
# Linear regression analysis to find the relationship between the given two variable

# Inbuilt linear regression
def linear_regression_inbuilt(data, dependent_vars):
    vectorAssembler = VectorAssembler(inputCols = dependent_vars, outputCol = 'features')
    vector_df = vectorAssembler.transform(data)
    vector_df = vector_df.select(['features', 'crime_count'])
#     glr = GeneralizedLinearRegression(family="binomial", link="logit", maxIter=10, regParam=0.0)
#     lr_model = glr.fit(data)
    lr = LinearRegression(featuresCol = 'features', labelCol='crime_count', maxIter=10, regParam=0.3, el
asticNetParam=0.8)
    lr_model = lr.fit(vector_df)
    return lr_model

# Custom multivariate linear regression
def linear_regression(data, indexes):

    # Gathering the data for linear regression
    x = []
    y = []
    for each in data:
        x_s = [ float(each[index]) for index in indexes if each[index] != None]
        if len(x_s) == len(indexes):
            x.append(x_s)
            y.append(float(each[1]))

    # Preparing the data for linear regression
    N = len(x)
    M = len(indexes)
    df = N - (1+M)
    X = np.reshape(x,(N,M))
    Y = np.reshape(y,(len(y),1))

    if df <= 0:
        return -1

    if N > 1:
        X = (X - np.mean(X,axis=0))/np.std(X,axis=0)
        Y = (Y - np.mean(Y))/ np.std(Y)

    # Linear regression
    X = np.hstack((X,np.ones((N,1))))
    X_t = np.transpose(X)
    X_inv = np.linalg.pinv(np.dot(X_t,X))
    weights = np.dot( np.dot(X_inv,X_t) ,Y)

    # Finding the p-value
    rss = np.sum(np.power((Y - np.dot(X, weights)), 2))
    s_squared = rss / df
    se = np.sum(np.power((X[:, 0]), 2))
    tt = (weights[0, 0] / np.sqrt(s_squared / se))
    pval = stats.t.sf(np.abs(tt), df)

    # Returning the betas and pvalue
    return weights[0][0],pval
```

In [7]:

```
# Creating the spark context
spark = SparkSession.builder.master("local[*]").getOrCreate()

# Reading the data
crime_data = spark.read.json("hdfs://home/udit_gupta_1/processed_data/")
demographics = spark.read.option("header", "true").csv("../Data/borough_demographics.csv")

# Preparing data for linear regression
# Finding borough level aggregate
complaints = crime_data.rdd.filter(lambda row: row['RECORD_TYPE'] == 'C').toDF()
complaints_df = complaints.filter(col("Year").isin(keep_column))
complaints_df = complaints_df.filter(col("BORO_NM").isin(list(boro_dict.values())))
complaints_df = complaints_df.withColumn("Key",fn.concat(fn.col("BORO_NM"),fn.col("Year")))
crime_count = complaints_df.groupby("Key").agg(fn.count(col('CMPLNT_NUM')).alias('crime_count'))

# Casting the data to suitable types
demographics = demographics.withColumn("unemployment_rate", demographics["unemployment_rate"].cast(Doubl
eType()))
demographics = demographics.withColumn("racial_diversity", demographics["racial_diversity"].cast(DoubleT
ype()))
demographics = demographics.withColumn("income_diversity", demographics["income_diversity"].cast(DoubleT
ype()))
demographics = demographics.withColumn("proverty_rate", demographics["proverty_rate"].cast(DoubleType())
)
demographics = demographics.withColumn("population", demographics["population"].cast(DoubleType()))

# Combining Data
final_data = crime_count.join(demographics, crime_count.Key == demographics.Demo_Key)
final_data = final_data.drop('Demo_Key')
demo_crime_data = final_data.rdd.map(list).collect()
```

In [16]:

```
# Correlation between the crime_count and other demographics data
for i in final_data.columns:
    if not( isinstance(final_data.select(i).take(1)[0][0], six.string_types)):
        print( "Correlation between crime_count and", i, final_data.stat.corr('crime_count',i))
```

```
Correlation between crime_count and crime_count 1.0
Correlation between crime_count and proverty_rate 0.1895440726873476
Correlation between crime_count and income_diversity -0.19683924650147086
Correlation between crime_count and racial_diversity 0.2940662008177049
Correlation between crime_count and unemployment_rate -0.20042337200923163
Correlation between crime_count and population 0.3918328051254336
```

In [11]:

```
# Inbuilt linear regression
lr_model = linear_regression_inbuilt(final_data, ["unemployment_rate"])
model_summary = lr_model.summary
print("r2: %f" % model_summary.r2)
```

```
r2: 0.040170
```

Hyothesis Testing

- Level of Significance is 0.05
- The null hypthesis states that demographic factor don't effect the crime count.
- The null hypothesis is rejected if the pvalue is less than 0.05.
- The null hypothesis is not rejected if the pvalue is greater than 0.05.

Below we are printing the pvalues for linear regression between crime rate and various demographic factors.

In [19]:
```
print("Significance of various demographic factors with crime_rate")
print("crime_rate vs proverty_rate "+str(linear_regression(demo_crime_data, [4])[1]))
print("crime_rate vs income_diversity "+ str(linear_regression(demo_crime_data, [5])[1]))
print("crime_rate vs racial_diversity "+str(linear_regression(demo_crime_data, [6])[1]))
print("crime_rate vs unemployment_rate "+str(linear_regression(demo_crime_data, [7])[1]))

print("\nSignificance of various demographic factors with crime_rate controlled by population")
print("crime_rate vs proverty_rate controlled by population "+str(linear_regression(demo_crime_data, [4,
8])[1]))
print("crime_rate vs income_diversity controlled by population "+ str(linear_regression(demo_crime_data,
[5,8])[1]))
print("crime_rate vs racial_diversity controlled by population "+str(linear_regression(demo_crime_data,
[6,8])[1]))
print("crime_rate vs unemployment_rate controlled by population  "+str(linear_regression(demo_crime_data
, [7,8])[1]))
```

```
Significance of various demographic factors with crime_rate
crime_rate vs proverty_rate 0.06523546120848842
crime_rate vs income_diversity 0.03606532612160858
crime_rate vs racial_diversity 0.008711275823851098
crime_rate vs unemployment_rate 0.05471047391610688

Significance of various demographic factors with crime_rate controlled by population
crime_rate vs proverty_rate controlled by population 0.1829580286423675
crime_rate vs income_diversity controlled by population 0.04857684388345955
crime_rate vs racial_diversity controlled by population 0.00022642815530583585
crime_rate vs unemployment_rate controlled by population  0.012688870210739571
```