Lab Course Machine Learning

Exercise 4

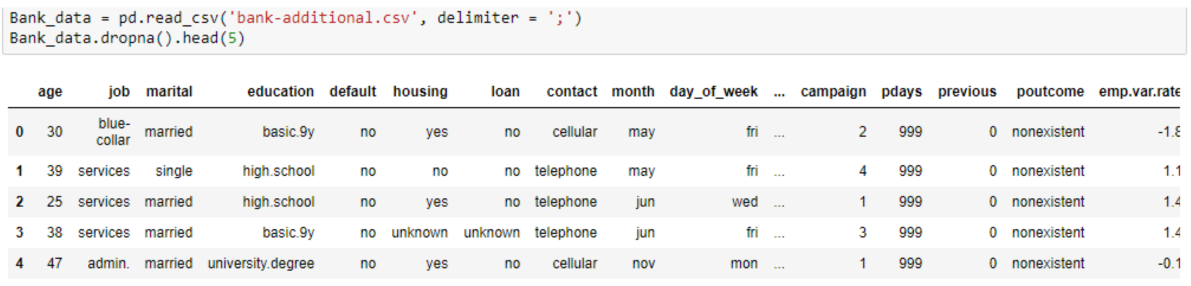
Kalaiselvan Panneerselvam

278920

1 Data Preprocessing

– Load the data using pandas & dropping NA rows

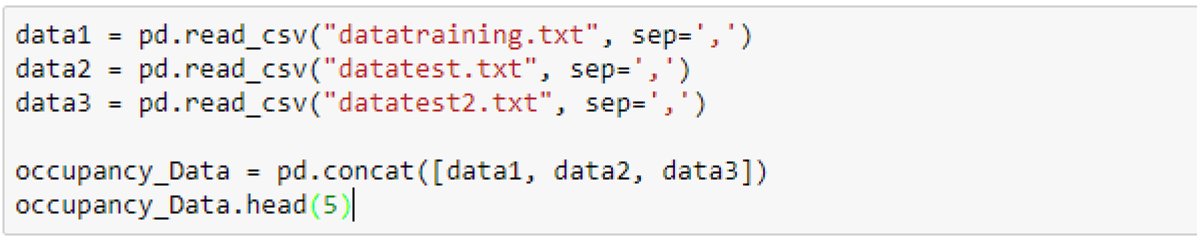
Bank Marketing Dataset



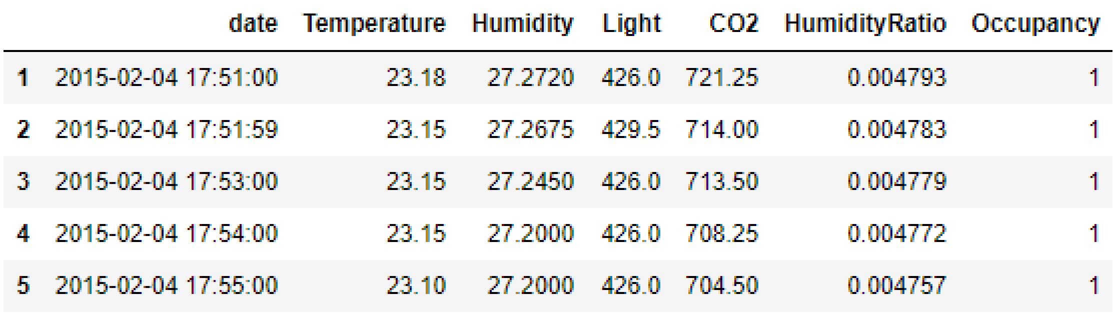
```
Bank_data = pd.read_csv('bank-additional.csv', delimiter = ';')
Bank_data.dropna().head(5)
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	fri	...	2	999	0	nonexistent	-1.8
1	39	services	single	high.school	no	no	no	telephone	may	fri	...	4	999	0	nonexistent	1.1
2	25	services	married	high.school	no	yes	no	telephone	jun	wed	...	1	999	0	nonexistent	1.4
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	fri	...	3	999	0	nonexistent	1.4
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	mon	...	1	999	0	nonexistent	-0.1

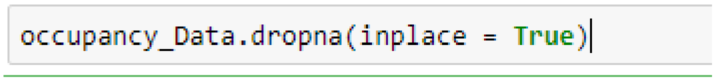
Occupancy Detection Dataset



```
data1 = pd.read_csv("datatraining.txt", sep=',')
data2 = pd.read_csv("datatest.txt", sep=',')
data3 = pd.read_csv("datatest2.txt", sep=',')

occupancy_Data = pd.concat([data1, data2, data3])
occupancy_Data.head(5)
```



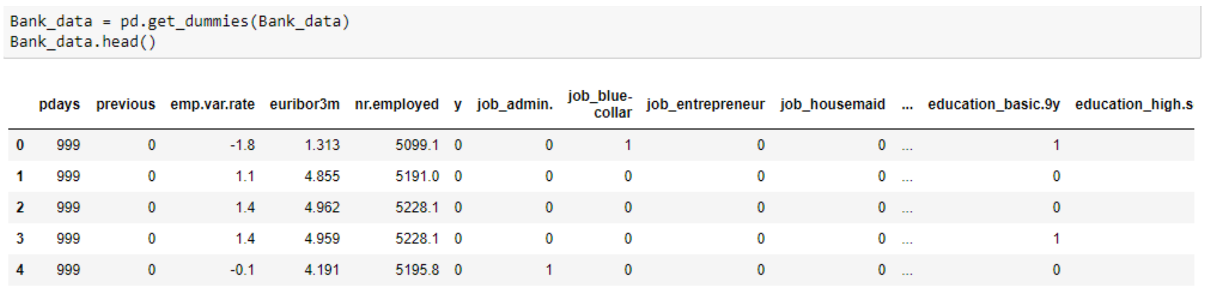
	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
1	2015-02-04 17:51:00	23.18	27.2720	426.0	721.25	0.004793	1
2	2015-02-04 17:51:59	23.15	27.2675	429.5	714.00	0.004783	1
3	2015-02-04 17:53:00	23.15	27.2450	426.0	713.50	0.004779	1
4	2015-02-04 17:54:00	23.15	27.2000	426.0	708.25	0.004772	1
5	2015-02-04 17:55:00	23.10	27.2000	426.0	704.50	0.004757	1



```
occupancy_Data.dropna(inplace = True)
```

- Convert any non-numeric values to numeric values

Bank Marketing Dataset

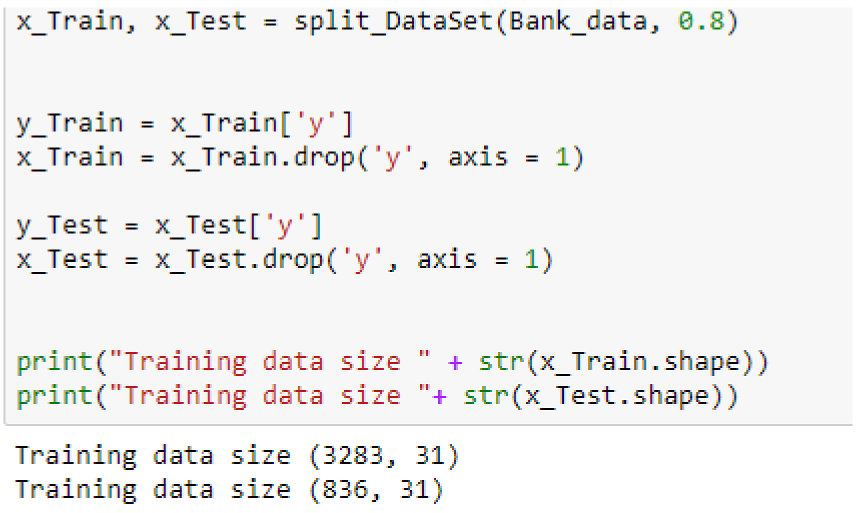


```
Bank_data = pd.get_dummies(Bank_data)
Bank_data.head()
```

	pdays	previous	emp.var.rate	euribor3m	nr.employed	y	job_admin.	job_blue-collar	job_entrepreneur	job_housemaid	...	education_basic.9y	education_high.s
0	999	0	-1.8	1.313	5099.1	0	0	1	0	0	...	1	
1	999	0	1.1	4.855	5191.0	0	0	0	0	0	...	0	
2	999	0	1.4	4.962	5228.1	0	0	0	0	0	...	0	
3	999	0	1.4	4.959	5228.1	0	0	0	0	0	...	1	
4	999	0	-0.1	4.191	5195.8	0	1	0	0	0	...	0	

- Split the data into a train(80%) and test(20%)

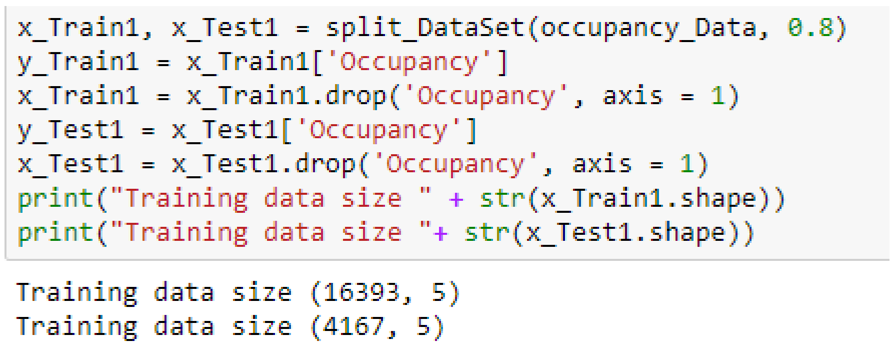
Bank Marketing Dataset



```
x_Train, x_Test = split_DataSet(Bank_data, 0.8)


y_Train = x_Train['y']
x_Train = x_Train.drop('y', axis = 1)

y_Test = x_Test['y']
x_Test = x_Test.drop('y', axis = 1)


print("Training data size " + str(x_Train.shape))
print("Training data size "+ str(x_Test.shape))
```

```
Training data size (3283, 31)
Training data size (836, 31)
```

Occupancy Detection Dataset



```
x_Train1, x_Test1 = split_DataSet(occupancy_Data, 0.8)
y_Train1 = x_Train1['Occupancy']
x_Train1 = x_Train1.drop('Occupancy', axis = 1)
y_Test1 = x_Test1['Occupancy']
x_Test1 = x_Test1.drop('Occupancy', axis = 1)
print("Training data size " + str(x_Train1.shape))
print("Training data size "+ str(x_Test1.shape))
```

```
Training data size (16393, 5)
Training data size (4167, 5)
```

1 Linear Classification with Gradient Ascent

Feature selection

Bank Marketing Dataset

The Bank Marketing dataset used is related with direct marketing campaigns of a banking institution. The marketing campaigns were based primarily on calls made via phone or landline(telephone). Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be (or not) subscribed. This project aims to measure the success of the bank's marketing campaign by predict whether a customer will be subscribe for a product.

Bank Marketing Data:

- age (numeric)
- job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
- marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
- education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
- default: has credit in default? (categorical: 'no','yes','unknown')
- housing: has housing loan? (categorical: 'no','yes','unknown')
- loan: has personal loan? (categorical: 'no','yes','unknown')

Related with the last contact of the current campaign:

- contact: contact communication type (categorical: 'cellular','telephone')
- month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
- day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
- duration: last contact duration, in seconds (numeric)

Other Attributes:

- campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
- previous: number of contacts performed before this campaign and for this client (numeric)
- poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

Social and Economic Context Attributes

- emp.var.rate: employment variation rate - quarterly indicator (numeric)

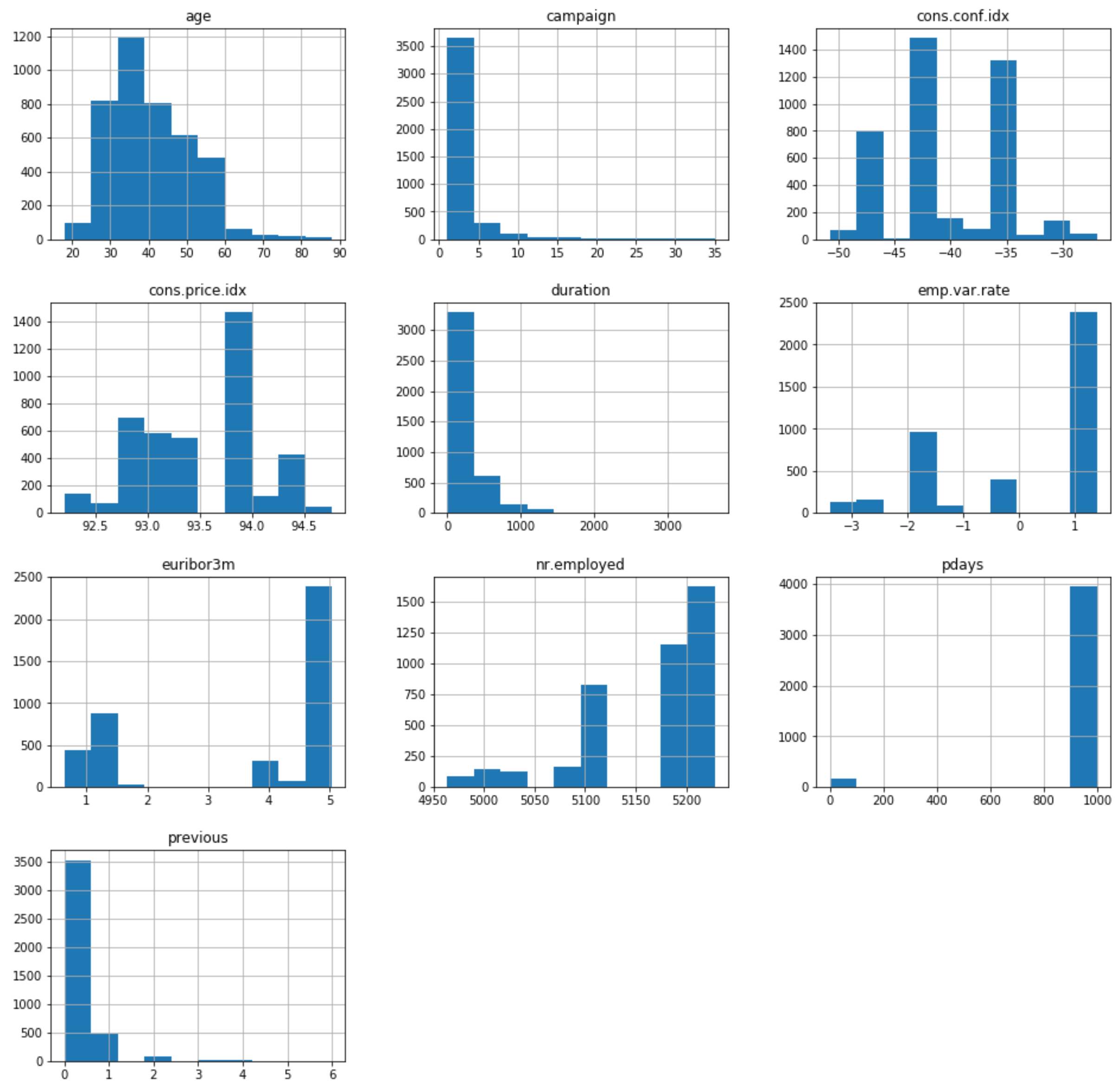
- cons.price.idx: consumer price index - monthly indicator (numeric)
- cons.conf.idx: consumer confidence index - monthly indicator (numeric)
- euribor3m: euribor 3 month rate - daily indicator (numeric)
- nr.employed: number of employees - quarterly indicator (numeric)

Output variable (target):

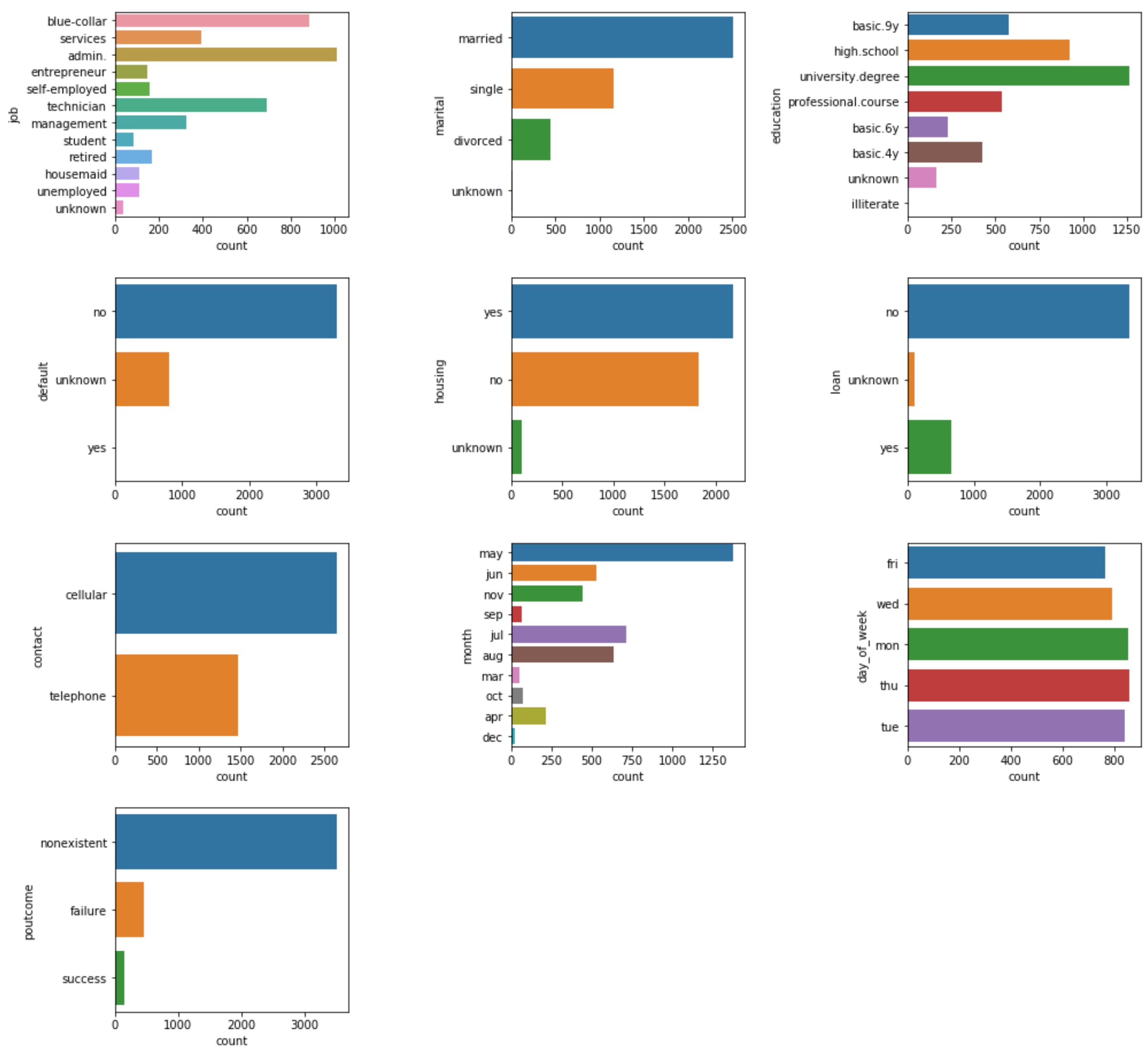
- y: has the client subscribed a term deposit? (binary: 'yes','no')

Visualising continuous and categorical columns

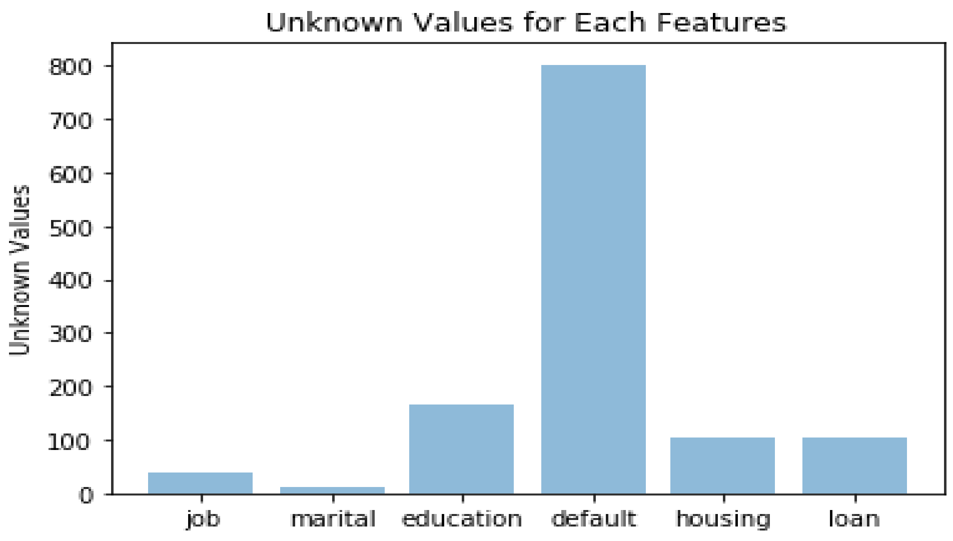
-Histogram of Continuous Variables



-Histogram of Categorical Variables



Unknown Values in each features:



The column ‘default’ has more number of unknown values and therefore I drop this feature for the prediction. After exploring the data and analyzing all the visualization, we conclude to drop the features such as 'contact', 'month', 'duration', 'default', and all others that would have less impact on the chances of customer subscription.

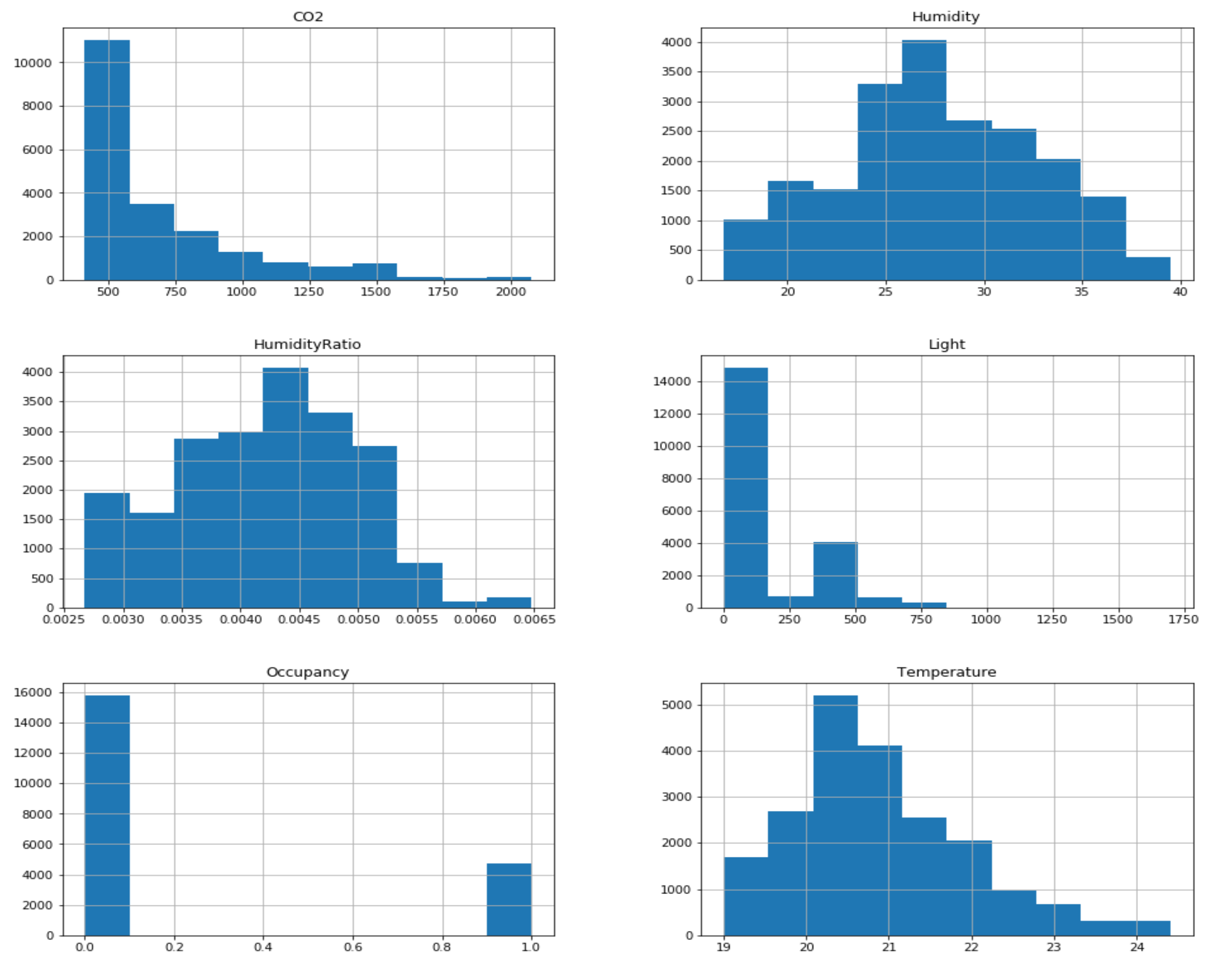
Occupancy Detection Dataset :

The goal of the Occupancy Detection Dataset is correctly detect the presence of users given information on time of day, humidity, CO2 levels, etc.

Attribute Information:

- date time year-month-day hour:minute:second
- Temperature, in Celsius
- Relative Humidity, %
- Light, in Lux
- CO2, in ppm
- Humidity Ratio, Derived quantity from temperature and relative humidity, in kgwater-vapor/kg-air
- Occupancy, 0 or 1, 0 for not occupied, 1 for occupied status

- Histogram of Variables



I dropped the feature ‘date’ in the occupancy dataset since it has less impact on the target y.

- Linear Regression model is given as $\hat{y}^n = \sigma(\beta^T \mathbf{x}^n)$ where σ is a logistic function $\frac{1}{1+e^{-\beta^T \mathbf{x}^n}}$

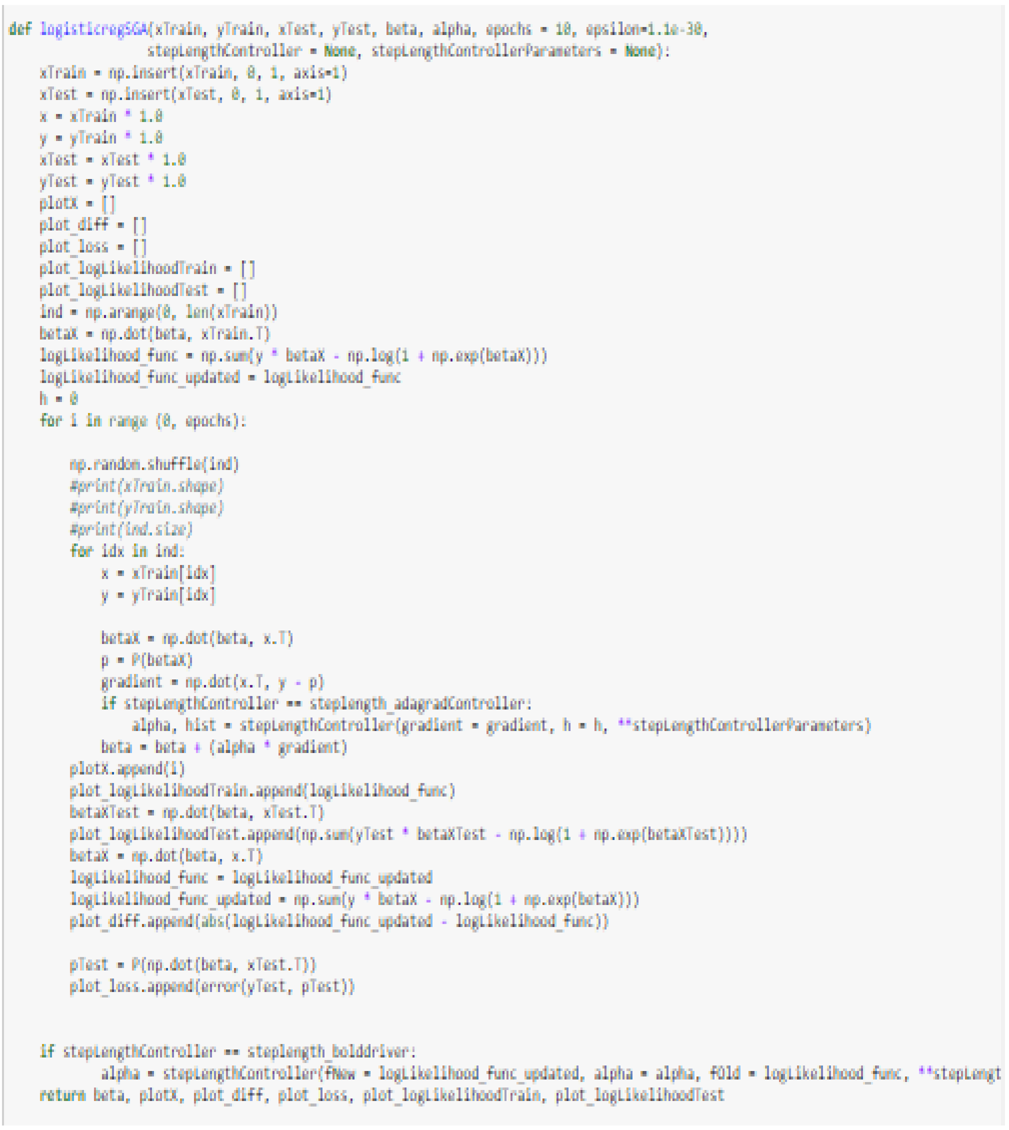
The logistic function is given by P(x) and the logloss function is given by error(y,p).

```python
def P(x):
    return 1.0/(1.0 + np.exp(-x))
def error(y, p):
    return -np.sum(y * np.log(p) + (1 - y) * np.log(1 - p))
```

- Optimize the loglikelihood function $l(x, y)$ using Gradient Descent algorithm. Implement **(log-regSGA/SGD and SGA/SGD algorithms)**. Choose i_{max} between 100 to 1000.

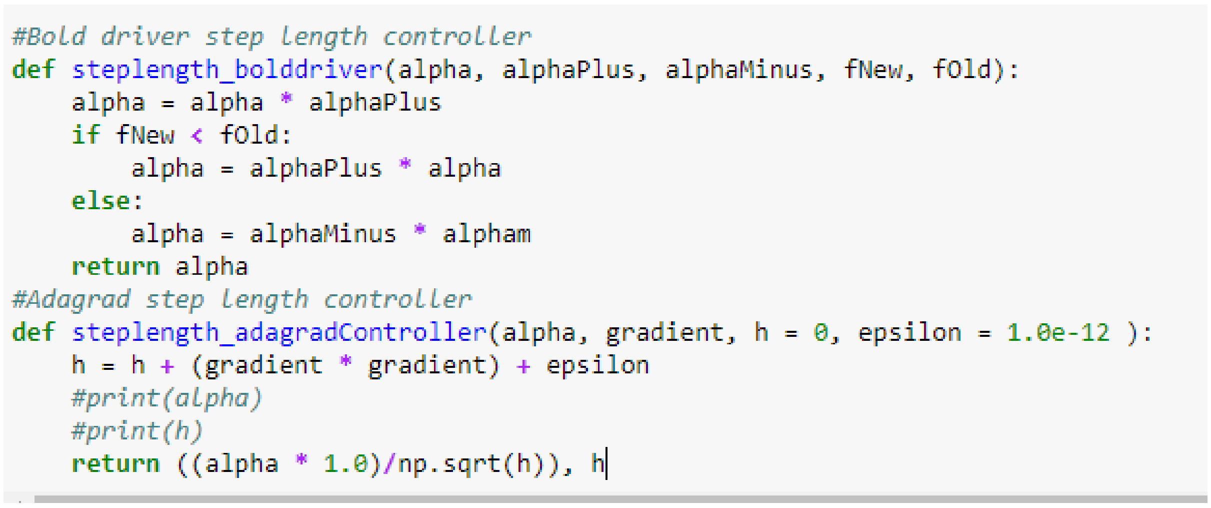
The Gradient Ascent is given by the function logisticregGA() and the Stochastic Gradient Ascent is given by function logisticregSGA().

```python
def logisticregGA(x, y, xTest, yTest, beta, alpha , max_itr=1000, epsilon=1.1e-20):
    x = np.insert(x, 0, 1, axis=1)
    xTest = np.insert(xTest, 0, 1, axis=1)
    x = x * 1.0
    y = y * 1.0
    xTest = xTest * 1.0
    yTest = yTest * 1.0
    xTranspose = x.T
    plotX = []
    plot_diff = []
    plot_loss = []
    plot_logLikelihoodTrain = []
    plot_logLikelihoodTest = []
    betaX = np.dot(beta, xTranspose)
    logLikelihood_func = np.sum(y * betaX - np.log(1 + np.exp(betaX)))
    for i in range(0, max_itr):
        p = P(betaX)
        gradient = np.dot(x.T , y - p)
        beta = beta + (alpha * gradient)
        betaX = np.dot(beta, x.T)
        logLikelihood_func_updated = np.sum(y * betaX - np.log(1 + np.exp(betaX)))
        plot_diff.append(abs(logLikelihood_func_updated - logLikelihood_func))
        plotX.append(i)
        pTest = P(np.dot(beta, xTest.T))
        plot_loss.append(error(yTest, pTest))
        plot_logLikelihoodTrain.append(logLikelihood_func_updated)
        betaXTest = np.dot(beta, xTest.T)
        plot_logLikelihoodTest.append(np.sum(yTest * betaXTest - np.log(1 + np.exp(betaXTest))))
        if abs(logLikelihood_func_updated - logLikelihood_func) < epsilon:
            print("Converged in " + str(i) + " iterations")
            return beta, plotX, plot_diff, plotY_RMSE, lossfunction, rmse
        logLikelihood_func = logLikelihood_func_updated
    print("Algorithm does not converge in " + str(max_itr) + " interations")
    return beta, plotX, plot_diff, plot_loss, plot_logLikelihoodTrain, plot_logLikelihoodTest
```



```python
def logisticregSGA(xTrain, yTrain, xTest, yTest, beta, alpha, epochs = 10, epsilon=1.1e-30,
                   stepLengthController = None, stepLengthControllerParameters = None):
    xTrain = np.insert(xTrain, 0, 1, axis=1)
    xTest = np.insert(xTest, 0, 1, axis=1)
    x = xTrain * 1.0
    y = yTrain * 1.0
    xTest = xTest * 1.0
    yTest = yTest * 1.0
    plotX = []
    plot_diff = []
    plot_loss = []
    plot_logLikelihoodTrain = []
    plot_logLikelihoodTest = []
    ind = np.arange(0, len(xTrain))
    betaX = np.dot(beta, xTrain.T)
    logLikelihood_func = np.sum(y * betaX - np.log(1 + np.exp(betaX)))
    logLikelihood_func_updated = logLikelihood_func
    h = 0
    for i in range (0, epochs):

        np.random.shuffle(ind)
        #print(xTrain.shape)
        #print(yTrain.shape)
        #print(ind.size)
        for idx in ind:
            x = xTrain[idx]
            y = yTrain[idx]

            betaX = np.dot(beta, x.T)
            p = P(betaX)
            gradient = np.dot(x.T, y - p)
            if stepLengthController == steplength_adagradController:
                alpha, hist = stepLengthController(gradient = gradient, h = h, **stepLengthControllerParameters)
            beta = beta + (alpha * gradient)
        plotX.append(i)
        plot_logLikelihoodTrain.append(logLikelihood_func)
        betaXTest = np.dot(beta, xTest.T)
        plot_logLikelihoodTest.append(np.sum(yTest * betaXTest - np.log(1 + np.exp(betaXTest))))
        betaX = np.dot(beta, x.T)
        logLikelihood_func = logLikelihood_func_updated
        logLikelihood_func_updated = np.sum(y * betaX - np.log(1 + np.exp(betaX)))
        plot_diff.append(abs(logLikelihood_func_updated - logLikelihood_func))

        pTest = P(np.dot(beta, xTest.T))
        plot_loss.append(error(yTest, pTest))


    if stepLengthController == steplength_bolddriver:
            alpha = stepLengthController(fNew = logLikelihood_func_updated, alpha = alpha, fOld = logLikelihood_func, **stepLengt
    return beta, plotX, plot_diff, plot_loss, plot_logLikelihoodTrain, plot_logLikelihoodTest
```

- Implement AdaGrad for adaptive step length and Bold Driver steplength

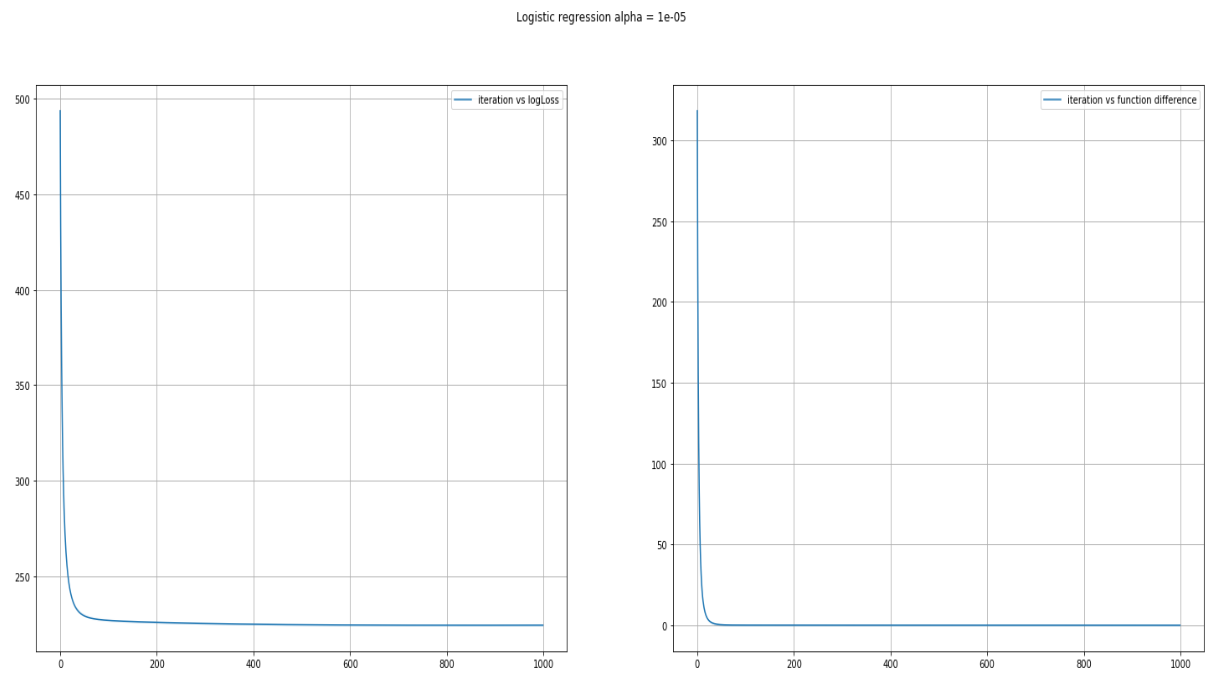


```python
#Bold driver step length controller
def steplength_bolddriver(alpha, alphaPlus, alphaMinus, fNew, fOld):
    alpha = alpha * alphaPlus
    if fNew < fOld:
        alpha = alphaPlus * alpha
    else:
        alpha = alphaMinus * alpham
    return alpha
#Adagrad step length controller
def steplength_adagradController(alpha, gradient, h = 0, epsilon = 1.0e-12 ):
    h = h + (gradient * gradient) + epsilon
    #print(alpha)
    #print(h)
    return ((alpha * 1.0)/np.sqrt(h)), h
```

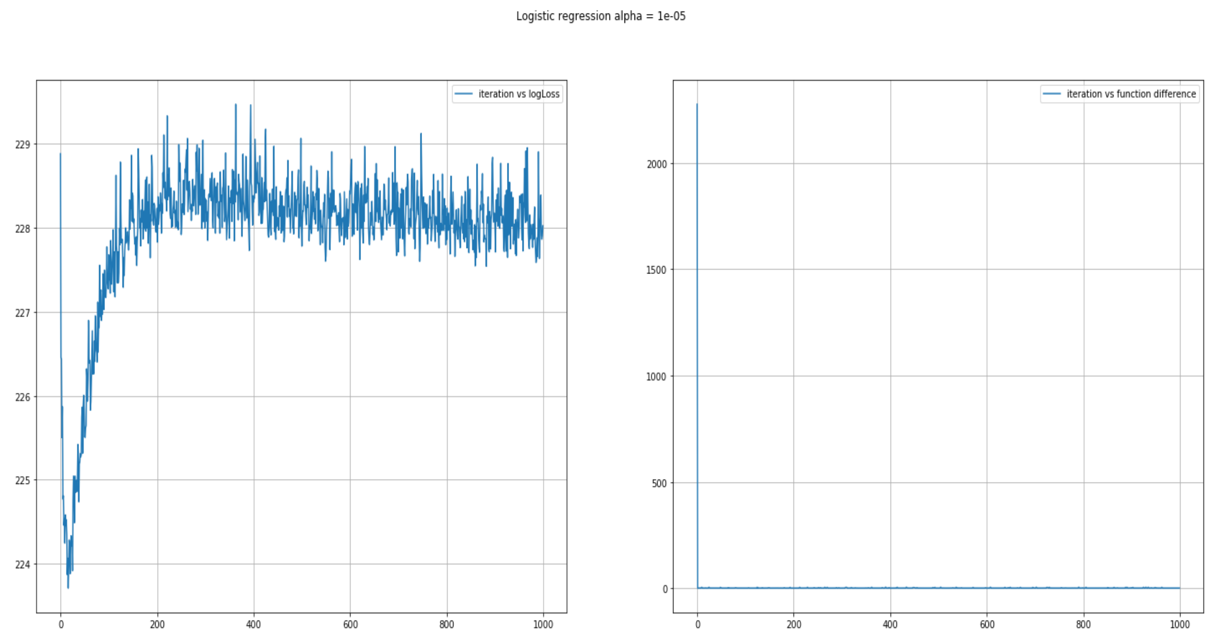
Plots of logloss and function difference of GA, SGA , SGA with BD and SGA with Adagrad

Bank Marketing Dataset

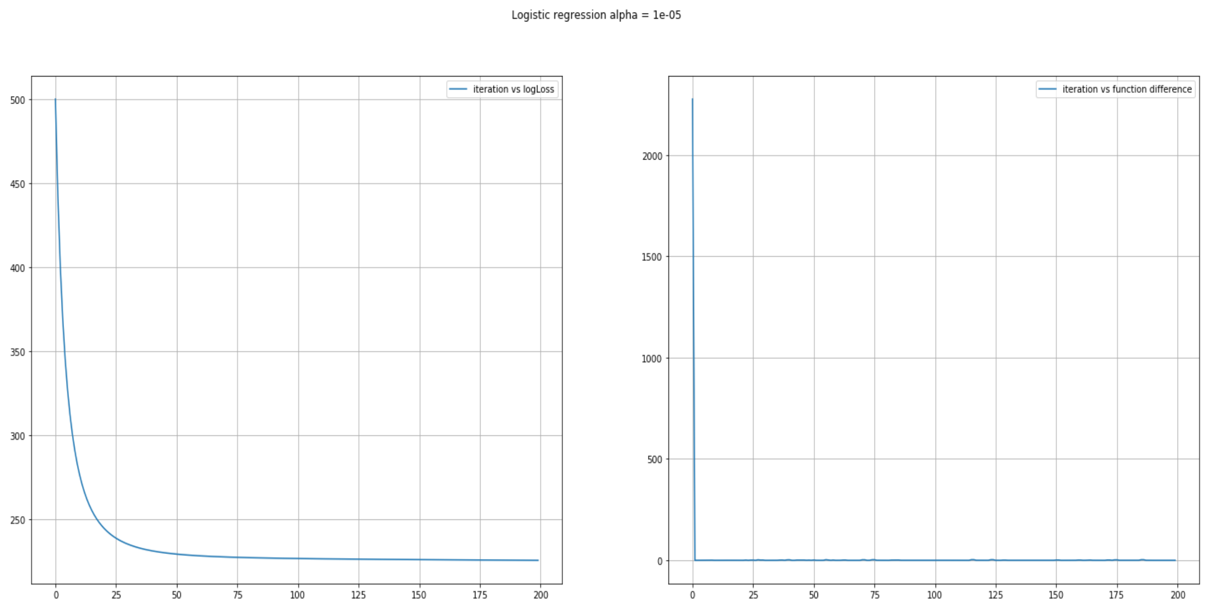
-Plots of logloss and function difference of Gradient Ascent



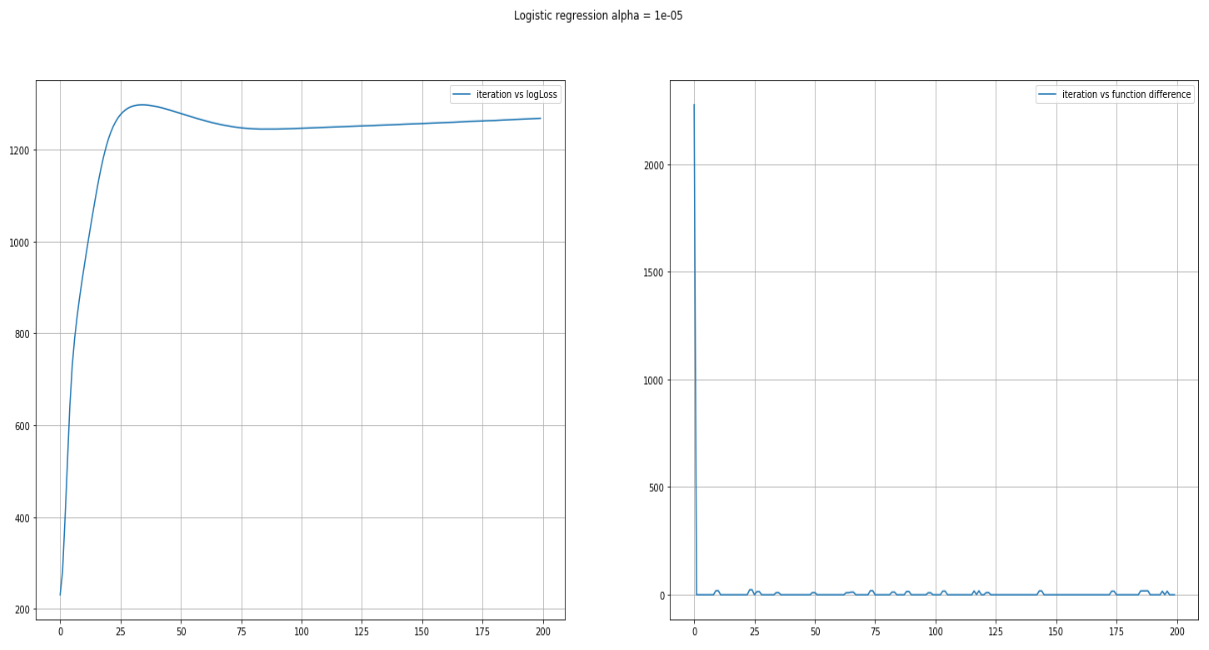
-Plots of logloss and function difference of Stochastic Gradient Ascent



-Plots of logloss and function difference of Stochastic Gradient Ascent with BoldDriver step length Controller

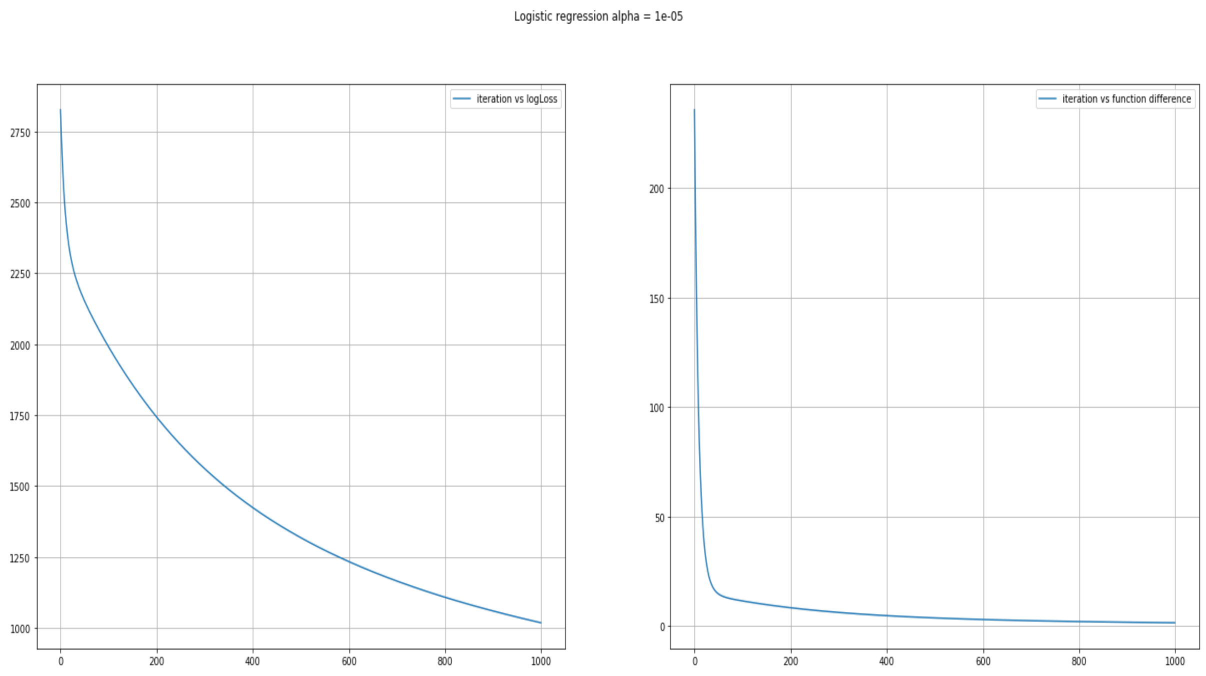


-Plots of logloss and function difference of Stochastic Gradient Ascent with Adagrad step length Controller

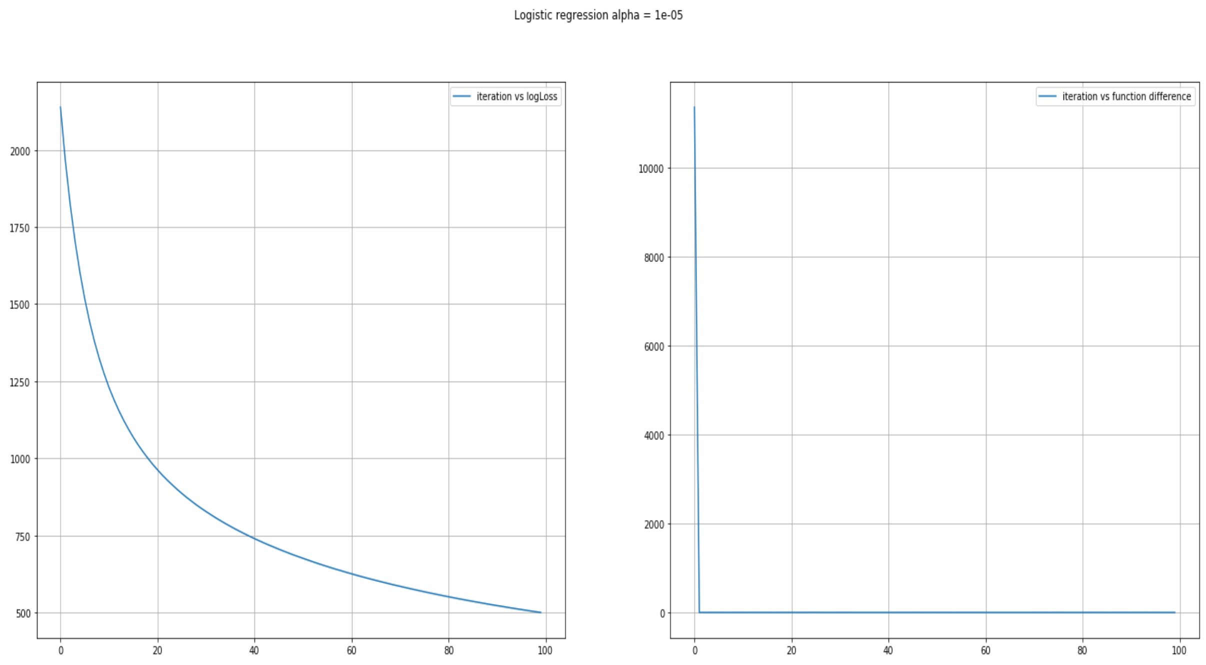


Occupancy detection Dataset

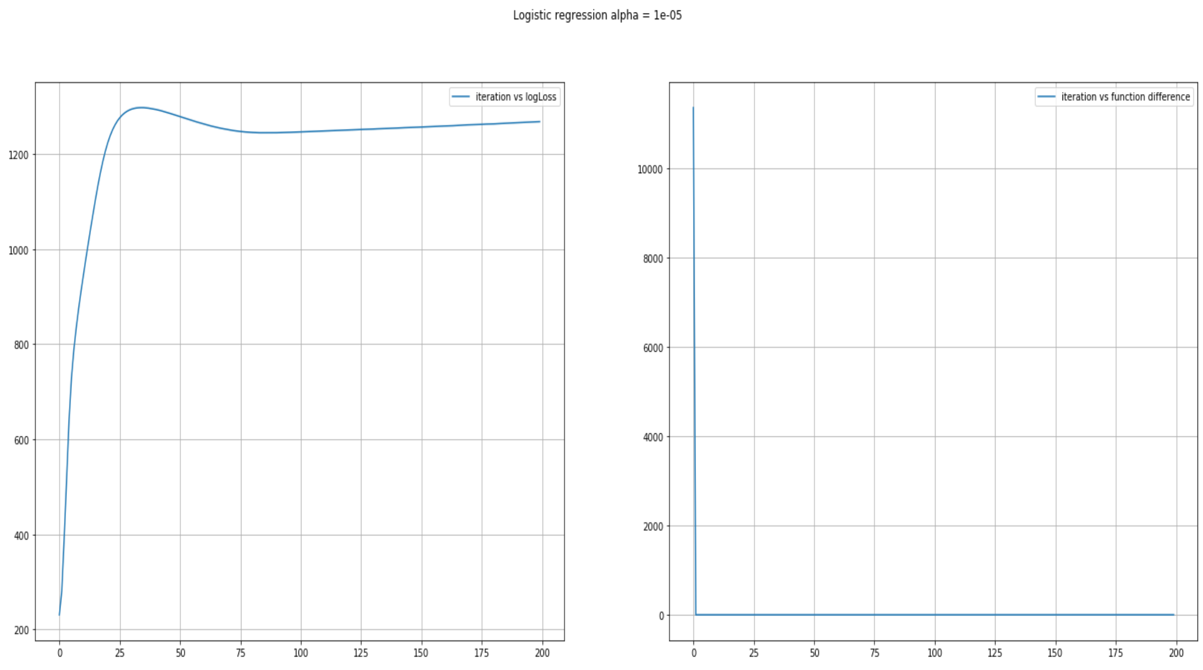
Plots of logloss and function difference of Gradient Ascent



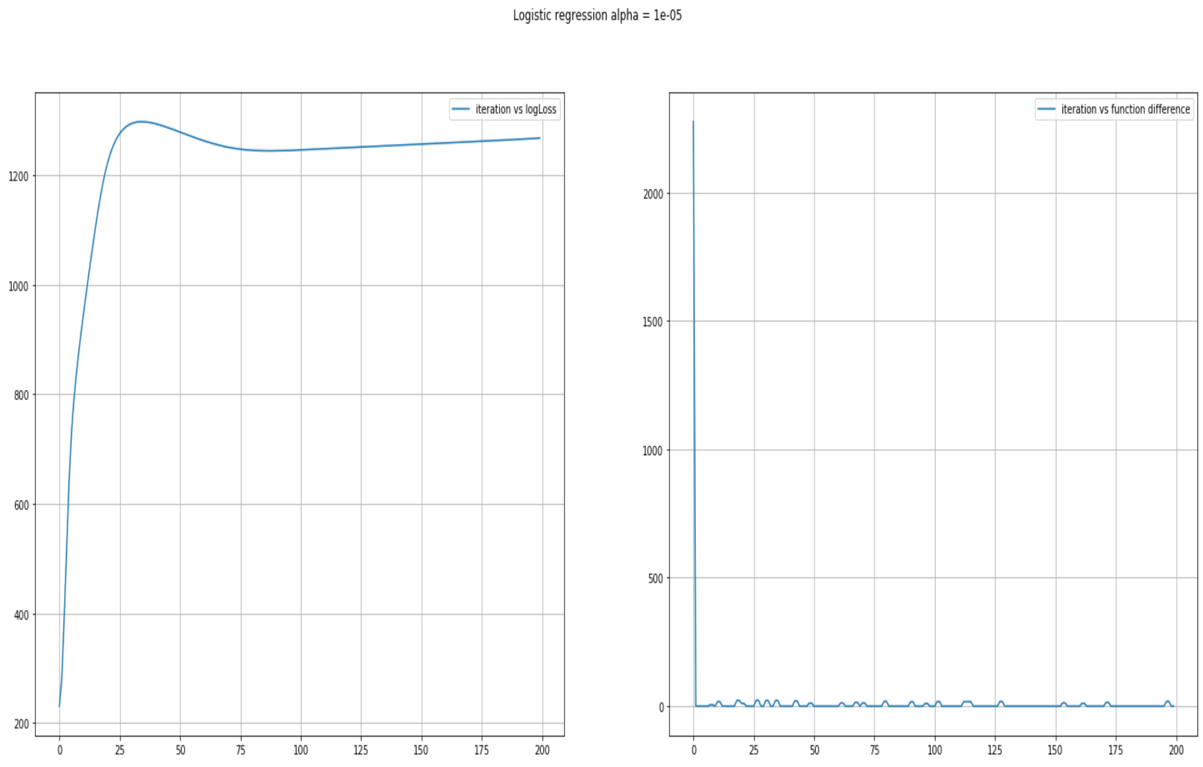
-Plots of logloss and function difference of Stochastic Gradient Ascent



-Plots of logloss and function difference of Stochastic Gradient Ascent with BoldDriver step length Controller



-Plots of logloss and function difference of Stochastic Gradient Ascent with Adagrad step length Controller


Logistic regression alpha = 1e-05
iteration vs logLoss
iteration vs function difference