# Check the versions of libraries

# Python version

import sys

print('Python: {}'.format(sys.version))

# scipy

import scipy

print('scipy: {}'.format(scipy.\_\_version\_\_))

# numpy

import numpy

print('numpy: {}'.format(numpy.\_\_version\_\_))

# matplotlib

import matplotlib

print('matplotlib: {}'.format(matplotlib.\_\_version\_\_))

# pandas

import pandas

print('pandas: {}'.format(pandas.\_\_version\_\_))

# scikit-learn

import sklearn

print('sklearn: {}'.format(sklearn.\_\_version\_\_))

# Load libraries

import pandas as pd

#from pandas.t

dataset = pd.read\_csv('C:\\Users\\vartika\\Desktop\\bankr2.csv')

print(dataset.shape)

#head

print(dataset.head(20))

#statistical summary

print(dataset.describe())

print(dataset.groupby('y').size())

import matplotlib.pyplot as plt

#data visualization

#univariate plots

#box and whisker plots

dataset.plot(kind='box', subplots=True, layout=(10,10), sharex=False, sharey=False)

plt.show()

#histograms

dataset.hist()

plt.show()

#scatter plot matrix

#scatter\_matrix(dataset)

#plt.show()

from sklearn import model\_selection

from sklearn.model\_selection import KFold

#create a validation dataset

# split-out validation dataset

array = dataset.values

X = array[:,0:3]

X=X.astype('int')

Y = array[:,3]

Y=Y.astype('int')

validation\_size = 0.20

seed = 7

X\_train, X\_validation, Y\_train, Y\_validation = model\_selection.train\_test\_split(X, Y, test\_size=validation\_size, random\_state=seed)

print(X\_train.shape, Y\_train.shape)

print(X\_validation.shape, Y\_validation.shape)

from pandas.plotting import scatter\_matrix

import matplotlib.pyplot as plt

from sklearn import model\_selection

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

# Test options and evaluation metr

num\_folds = 10

num\_instances = len(X\_train)

seed = 7

scoring = 'accuracy'

# Spot Check Algorithms

models = []

models.append(('LR', LogisticRegression()))

models.append(('LDA', LinearDiscriminantAnalysis()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('DT', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC()))

# evaluate each model in turn

results = []

names = []

for name, model in models:

kfold = model\_selection.KFold(n\_splits=10, random\_state=seed)

cv\_results = model\_selection.cross\_val\_score(model, X\_train, Y\_train, cv=kfold, scoring=scoring)

results.append(cv\_results)

names.append(name)

msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())

print(msg)

# Compare Algorithms

fig = plt.figure()

fig.suptitle('Algorithm Comparison')

ax = fig.add\_subplot(111)

plt.boxplot(results)

ax.set\_xticklabels(names)

plt.show()

# Make predictions on validation dataset

knn = KNeighborsClassifier()

knn.fit(X\_train, Y\_train)

predictions = knn.predict(X\_validation)

print('Accuracy score knn',accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

#prediction of logistic regression

model=LogisticRegression(penalty='l2', max\_iter=1000)

model.fit(X\_train, Y\_train)

predictions=model.predict(X\_validation)

print('Accuracy score LR',accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

#prediction of svm

svm = SVC()

print('SVM')

svm.fit(X\_train, Y\_train)

predictions=svm.predict(X\_validation)

print('Accuracy score SVM',accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

#prediction of decision tree

dt=DecisionTreeClassifier()

dt.fit(X\_train, Y\_train)

predictions = dt.predict(X\_validation)

print('Accuracy score DT',accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

#prediction of gaussian nb

nb=GaussianNB()

nb.fit(X\_train, Y\_train)

predictions = nb.predict(X\_validation)

print('Accuracy score NB',accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

#prediction of LDA

lda=LinearDiscriminantAnalysis()

lda.fit(X\_train, Y\_train)

predictions = lda.predict(X\_validation)

print('Accuracy score LDA',accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

# Check the versions of libraries

# Python version

import sys

print('Python: {}'.format(sys.version))

# scipy

import scipy

print('scipy: {}'.format(scipy.\_\_version\_\_))

# numpy

import numpy

print('numpy: {}'.format(numpy.\_\_version\_\_))

# matplotlib

import matplotlib

print('matplotlib: {}'.format(matplotlib.\_\_version\_\_))

# pandas

import pandas

print('pandas: {}'.format(pandas.\_\_version\_\_))

# scikit-learn

import sklearn

print('sklearn: {}'.format(sklearn.\_\_version\_\_))

# Load libraries

import pandas as pd

#from pandas.t

dataset = pd.read\_csv('C:\\Users\\vartika\\Desktop\\bankr1.csv')

print(dataset.shape)

#head

print(dataset.head(20))

#statistical summary

print(dataset.describe())

print(dataset.groupby('y').size())

import matplotlib.pyplot as plt

#data visualization

#univariate plots

#box and whisker plots

dataset.plot(kind='box', subplots=True, layout=(10,10), sharex=False, sharey=False)

plt.show()

#histograms

dataset.hist()

plt.show()

#scatter plot matrix

#scatter\_matrix(dataset)

#plt.show()

from sklearn import model\_selection

from sklearn.model\_selection import KFold

#create a validation dataset

# split-out validation dataset

array = dataset.values

X = array[:,0:3]

X=X.astype('int')

Y = array[:,3]

Y=Y.astype('int')

validation\_size = 0.20

seed = 7

X\_train, X\_validation, Y\_train, Y\_validation = model\_selection.train\_test\_split(X, Y, test\_size=validation\_size, random\_state=seed)

print(X\_train.shape, Y\_train.shape)

print(X\_validation.shape, Y\_validation.shape)

from pandas.plotting import scatter\_matrix

import matplotlib.pyplot as plt

from sklearn import model\_selection

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

# Test options and evaluation metr

num\_folds = 10

num\_instances = len(X\_train)

seed = 7

scoring = 'accuracy'

# Spot Check Algorithms

models = []

models.append(('LR', LogisticRegression()))

models.append(('LDA', LinearDiscriminantAnalysis()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('DT', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC()))

# evaluate each model in turn

results = []

names = []

for name, model in models:

kfold = model\_selection.KFold(n\_splits=10, random\_state=seed)

cv\_results = model\_selection.cross\_val\_score(model, X\_train, Y\_train, cv=kfold, scoring=scoring)

results.append(cv\_results)

names.append(name)

msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())

print(msg)

# Compare Algorithms

fig = plt.figure()

fig.suptitle('Algorithm Comparison')

ax = fig.add\_subplot(111)

plt.boxplot(results)

ax.set\_xticklabels(names)

plt.show()

# Make predictions on validation dataset

knn = KNeighborsClassifier()

knn.fit(X\_train, Y\_train)

predictions = knn.predict(X\_validation)

print('Accuracy score knn',accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

#prediction of logistic regression

model=LogisticRegression(penalty='l2', max\_iter=1000)

model.fit(X\_train, Y\_train)

predictions=model.predict(X\_validation)

print('Accuracy score LR',accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

#prediction of svm

svm = SVC()

print('SVM')

svm.fit(X\_train, Y\_train)

predictions=svm.predict(X\_validation)

print('Accuracy score SVM',accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

#prediction of decision tree

dt=DecisionTreeClassifier()

dt.fit(X\_train, Y\_train)

predictions = dt.predict(X\_validation)

print('Accuracy score DT',accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

#prediction of gaussian nb

nb=GaussianNB()

nb.fit(X\_train, Y\_train)

predictions = nb.predict(X\_validation)

print('Accuracy score NB',accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

#prediction of LDA

lda=LinearDiscriminantAnalysis()

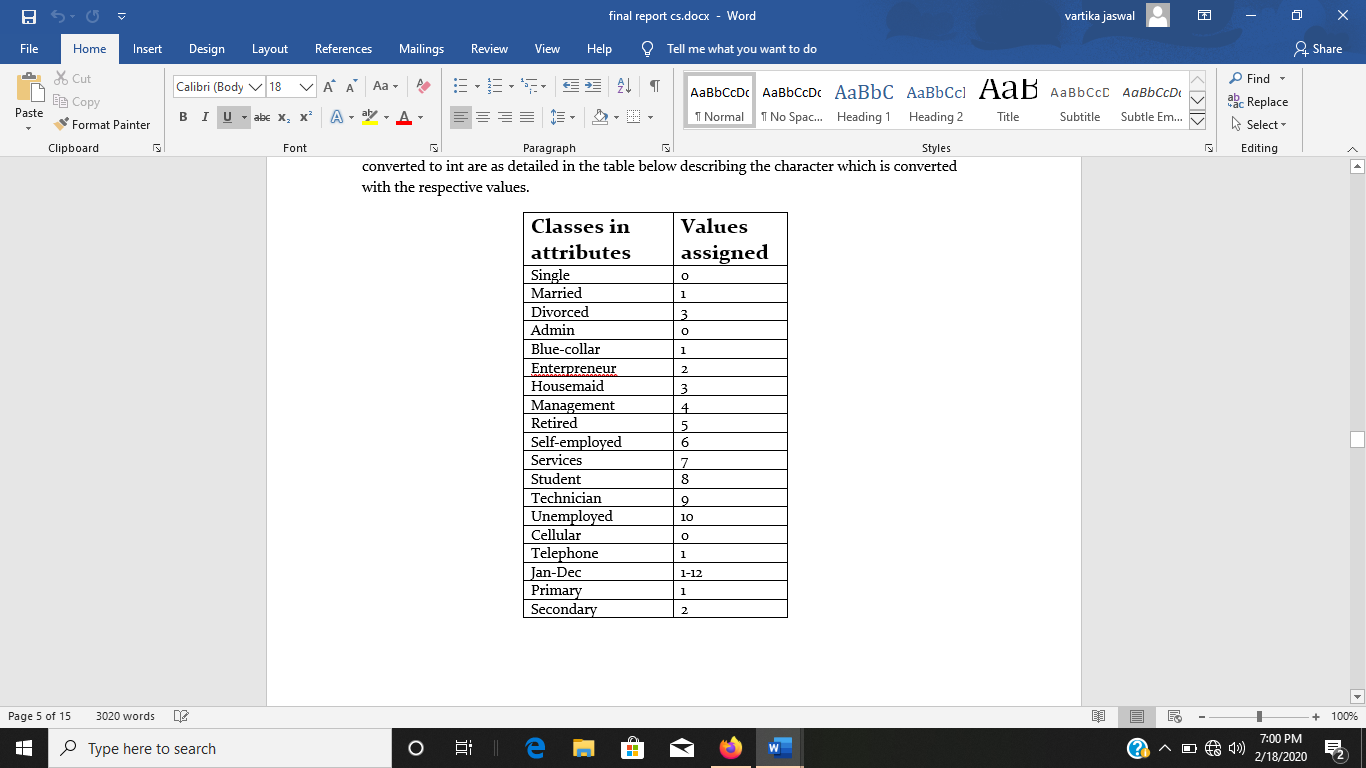
lda.fit(X\_train, Y\_train)

predictions = lda.predict(X\_validation)

print('Accuracy score LDA',accuracy\_score(Y\_validation, predictions))

print(confusion\_matrix(Y\_validation, predictions))

print(classification\_report(Y\_validation, predictions))

changed values in bankr1 from the downloaded dataset