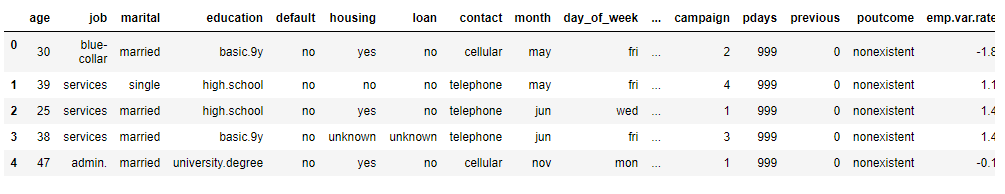
1. import pandas as pd
2. import numpy as np
3. import matplotlib.pyplot as plt
4. from sklearn.cross\_validation import train\_test\_split

**import CSV File into Data frame**

ds = pd.read\_csv('C:\\Users\\Kshitij\\Desktop\\bank-additional.csv',sep=';')

Read first five rows from the dataset

ds.head() 

**unique values from the column from the dataset**

ds.job.unique()

**Map column values with the number to make ML algorithm to learn**

ds['job'] = ds.job.map({'blue-collar':0,'services':1,'admin.':2, 'entrepreneur':3,'self-employed':4, 'technician':5, 'management':6, 'student':7, 'retired':8,'housemaid':9, 'unemployed':10, 'unknown':11})

**similarly we can map other columns values with numeric :-**

ds.marital = ds.marital.map({'married':0, 'single':1, 'divorced':2, 'unknown':3})

ds.education = ds.education.map({'basic.9y':0, 'high.school':1, 'university.degree':2, 'professional.course':3,'basic.6y':4,'basic.4y':5,'unknown':6,'illiterate

ds.default = ds.default.map({'no':0, 'unknown':2, 'yes':1})

ds.housing = ds.housing.map({'no':0, 'unknown':2, 'yes':1})

ds.loan = ds.loan.map({'no':0, 'unknown':2, 'yes':1})

ds.contact = ds.contact.map({'cellular':0, 'telephone':1})

s.month = ds.month.map({'may':5, 'jun':6, 'nov':11, 'sep':9, 'jul':7, 'aug':8, 'mar':3, 'oct':10, 'apr':4, 'dec':12})

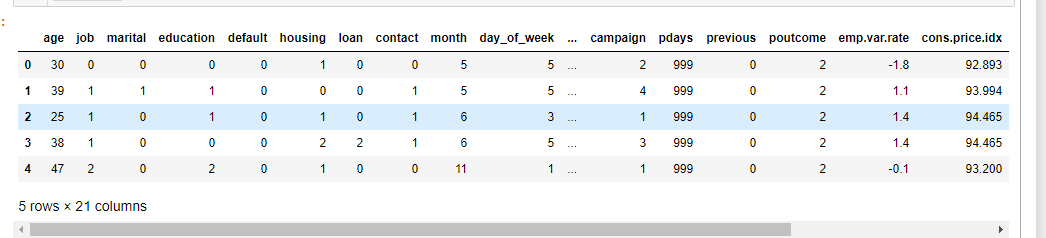
ds.day\_of\_week = ds.day\_of\_week.map({'fri':5, 'wed':3, 'mon':1, 'thu':4, 'tue':2})

ds.poutcome = ds.poutcome.map({'nonexistent':2, 'failure':0, 'success':1})

ds.y = ds.y.map({'yes':1, 'no':0})

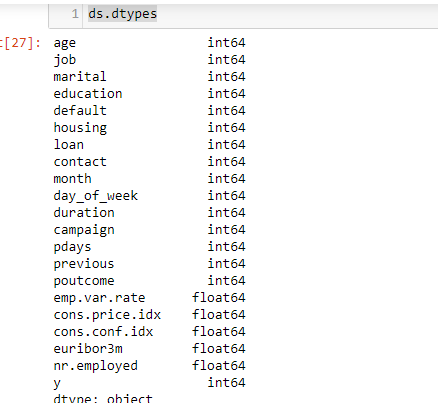
**Verifying data set completely changed to number format**

ds.head()

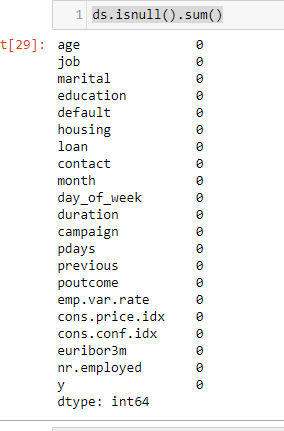


Determining data types of the columns

ds.dtypes

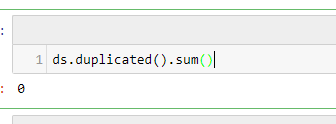


**Verifying dataset contains any null values or not**

****

**Calculating duplicates of the dataset**

ds.duplicated().sum()

****

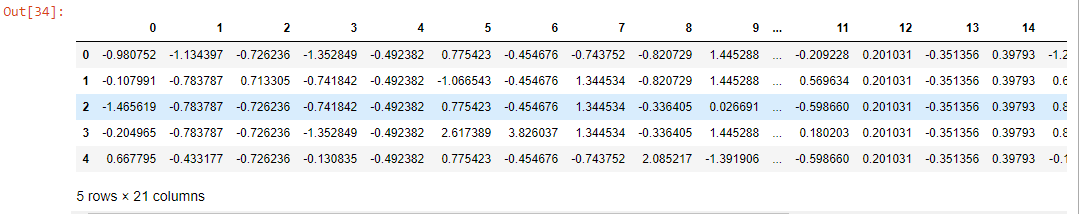
**converting dataset values into same scale**

from sklearn.preprocessing import StandardScaler

ds = StandardScaler().fit transform(ds)

ds = pd.DataFrame(ds)

ds.head()

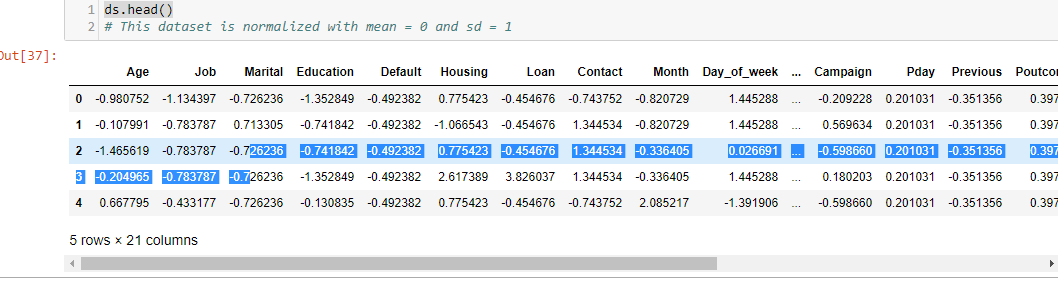


**Mapping again to the column names**

ds.rename(columns={0:'Age',1:'Job',2:'Marital',3:'Education',4:'Default',5:'Housing',6:'Loan',7:'Contact',8:'Month',9:'Day\_of\_week',10:'Duration',11:'Campaign',12:'Pday',13:'Previous', 14:'Poutcome',15:'Emp.var.rate',16:'Cons.price.idx',17:'Cons.conf.idx',18:'Euribor3m',19:'Nr.employed',20:'Y'},inplace=True)

**Converting into dataframe**

x = pd.DataFrame(ds)



**# Determine Class distribution**

ds.groupby('y').size()

y

0.0 3668

1.0 451

dtype: int64

This shows not allocated loans (zero) is more than allocated

as per class distribution.

**# Class distribution in terms of percentage**

ds.y.value\_counts(normalize=True)

0.0 0.890507

1.0 0.109493

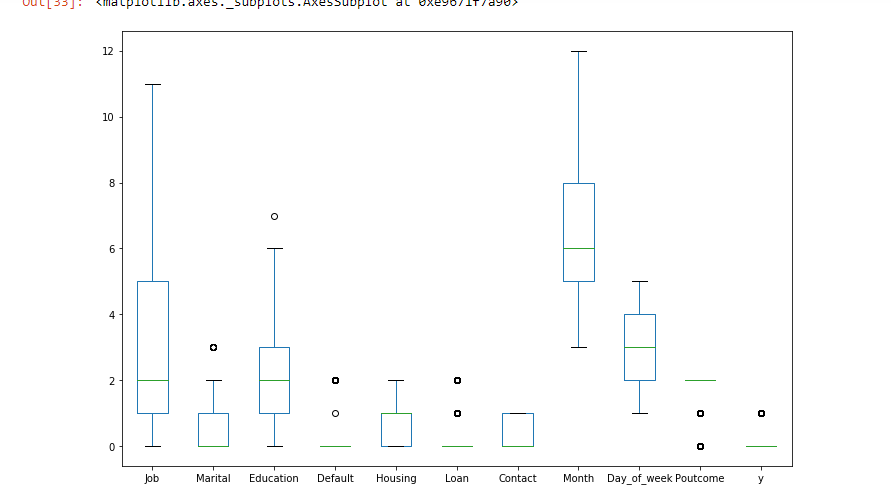
Name: y, dtype: float64

so above analysis dominated class in the dataset is not credited (i.e 0.0 = 0.890507 ) This is null accuracy.

%matplotlib inline

ds.plot(kind='box',figsize=(12,8))

//this shows distribution of data along with outliers

dot mentioned in the dataset are the outliers in the columns : Default, Loan, Poutcome,Martia, education. Columns do not contains outliers are : Job, Housing,Contact,Month

**Determine subbox plots**

ds.plot(kind='box',subplots=True,figsize=(12,12))

below is the subplots draws from boxplot above

Job Axes(0.125,0.125;0.0596154x0.755)

Marital Axes(0.196538,0.125;0.0596154x0.755)

Education Axes(0.268077,0.125;0.0596154x0.755)

Default Axes(0.339615,0.125;0.0596154x0.755)

Housing Axes(0.411154,0.125;0.0596154x0.755)

Loan Axes(0.482692,0.125;0.0596154x0.755)

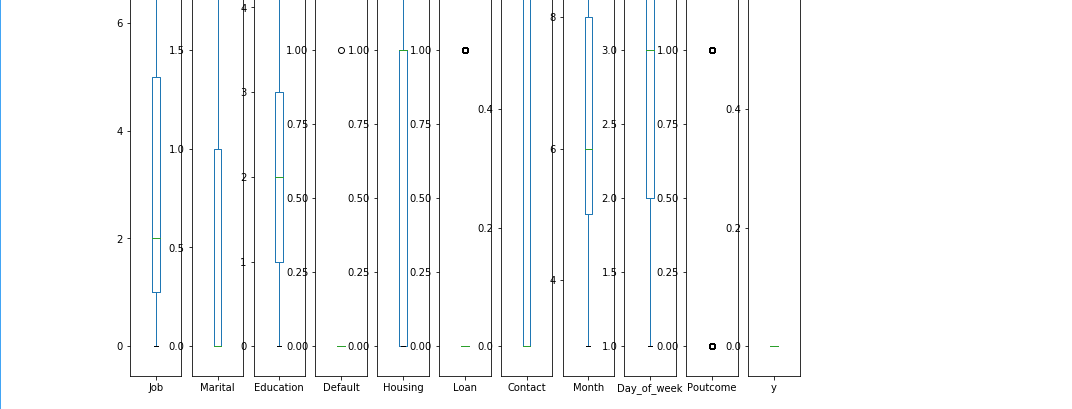
Contact Axes(0.554231,0.125;0.0596154x0.755)

Month Axes(0.625769,0.125;0.0596154x0.755)

Day\_of\_week Axes(0.697308,0.125;0.0596154x0.755)

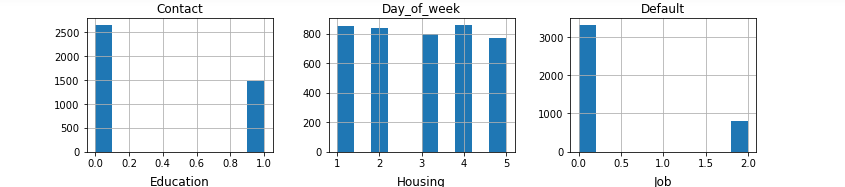
Poutcome Axes(0.768846,0.125;0.0596154x0.755)

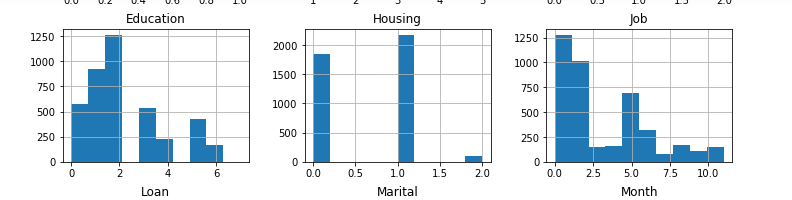
y Axes(0.840385,0.125;0.0596154x0.755)

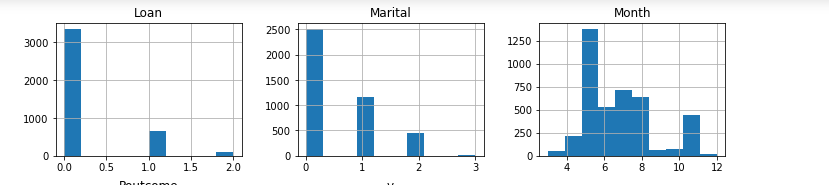


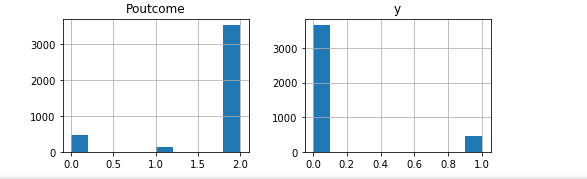
ds.hist(figsize=(12,12))

**Above draws histogram from the dataset**

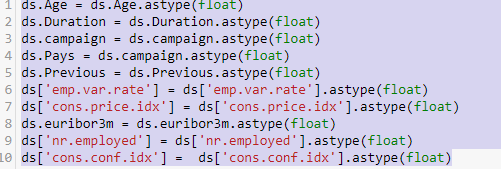








Convert Data types of the columns into float



**Changing dataset into array type**

array = ds.values

Taking all predictive variables to x dataset

x = array[:,0:19]

x.shape

y = array[:,20]

Taking all responsive variables to y dataset

y.shape

**# Standardized of the dataset into one scale**

from sklearn.preprocessing import StandardScaler

x = StandardScaler().fit\_transform(x)

x = pd.DataFrame(x)

x.head()

x.rename(columns={0:'Age',1:'Job',2:'Marital',3:'Education',4:'Default',5:'Housing',6:'Loan',7:'Contact',8:'Month',9:'Day\_of\_week',10:'Duration',11:'Campaign',12:'Pday',13:'Previous', 14:'Poutcome',15:'Emp.var.rate',16:'Cons.price.idx',17:'Cons.conf.idx',18:'Euribor3m',19:'Nr.employed'},inplace=True)

x\_train,x\_validation,y\_train,Y\_validation = model\_selection.train\_test\_split(x,y,test\_size=.10,random\_state=7)

**Above changing dataset into training and testing dataset**

x\_train.shape

(3707, 19)

x\_validation.shape

(412, 19)

y\_train.shape

(3707,)

Y\_validation.shape

(412,)

**Declaring array of models below**

models = []

models.append(('LR', LogisticRegression(solver='liblinear', multi\_class='ovr')))

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

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

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

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

models.append(('SVM', SVC(gamma='auto')))

print(models)

('LR', LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True,

intercept\_scaling=1, max\_iter=100, multi\_class='ovr', n\_jobs=1,

penalty='l2', random\_state=None, solver='liblinear', tol=0.0001,

verbose=0, warm\_start=False)), ('LDA', LinearDiscriminantAnalysis(n\_components=None, priors=None, shrinkage=None,

solver='svd', store\_covariance=False, tol=0.0001)), ('KNN', KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=1, n\_neighbors=5, p=2,

weights='uniform')), ('CART', DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=None,

max\_features=None, max\_leaf\_nodes=None,

min\_impurity\_split=1e-07, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

presort=False, random\_state=None, splitter='best')), ('NB', GaussianNB(priors=None)), ('SVM', SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,

decision\_function\_shape=None, degree=3, gamma='auto', kernel='rbf',

max\_iter=-1, probability=False, random\_state=None, shrinking=True,

tol=0.001, verbose=False))]

**Running cross validation for the training dataset**

results = []

names = []

for name, model in models:

kfold = model\_selection.KFold(n\_splits=10, random\_state=7)

cv\_results = model\_selection.cross\_val\_score(model, x\_train, y\_train, cv=kfold, scoring='accuracy')

results.append(cv\_results)

names.append(name)

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

print(msg)

LR: 0.910441 (0.013565)

LDA: 0.906657 (0.014475)

KNN: 0.898035 (0.011211)

CART: 0.886427 (0.011819)

NB: 0.817909 (0.014329)

SVM: 0.910175 (0.010480)

knn = KNeighborsClassifier()

knn.fit(x\_train,y\_train)

predictions = knn.predict(x\_validation)

print(accuracy\_score(Y\_validation,predictions))

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

**printing all classification parameters in a Go**

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

0.902912621359

[[357 11]

[ 29 15]]

precision recall f1-score support

0.0 0.92 0.97 0.95 368

1.0 0.58 0.34 0.43 44

avg / total 0.89 0.90 0.89 412

**Looking probabilities distribution**

x\_predicted\_probabilies = knn.predict\_proba(x\_validation)[:,1]

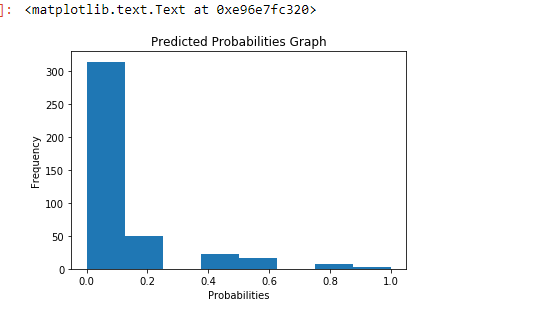
**# Visualize predicted probabilities of the class**

plt.hist(x\_predicted\_probabilies,bins=8)

plt.ylabel("Frequency")

plt.xlabel("Probabilities")

plt.title("Predicted Probabilities Graph")



**# AUC/ROC Curv**

from sklearn import metrics

t1,t2,threshold = metrics.roc\_curve(Y\_validation,x\_predicted\_probabilies)

plt.plot(t1,t2)

plt.xlim([0.0,1.0])

plt.ylabel([0.0,1.0])

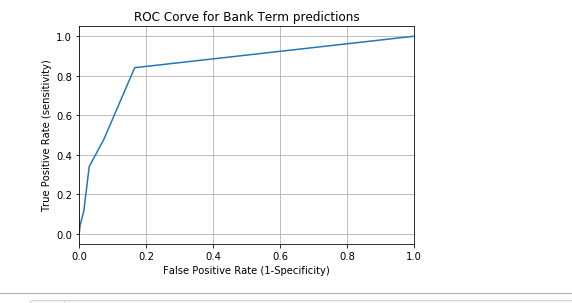
plt.title("ROC Corve for Bank Term predictions")

plt.xlabel("False Positive Rate (1-Specificity)")

plt.ylabel("True Positive Rate (sensitivity)")

plt.grid(True)

plt.show()



**Learning Curve**

The graph between training /destining dataset and model score (Accuracy) is called learning curve with varying training dataset.

import numpy as np

from sklearn.model\_selection import learning\_curve

logrego = LogisticRegression()

logrego.fit(x,y)

train\_sizes,train\_scores, test\_scores = learning\_curve(logrego,x,y, cv=kfold, scoring='accuracy',n\_jobs=-1,train\_sizes=np.linspace(0.01, 1.0, 50))

train\_mean = np.mean(train\_scores,axis=1)

train\_std = np.std(train\_scores,axis=1)

X\_validation\_mean =np.mean(test\_scores,axis=1)

x\_validation\_std = np.std(test\_scores,axis=1)

plt.figure()

plt.legend(loc="best")

plt.grid()

plt.fill\_between(train\_sizes,train\_mean - train\_std,train\_mean + train\_std,alpha=0.1,color="r")

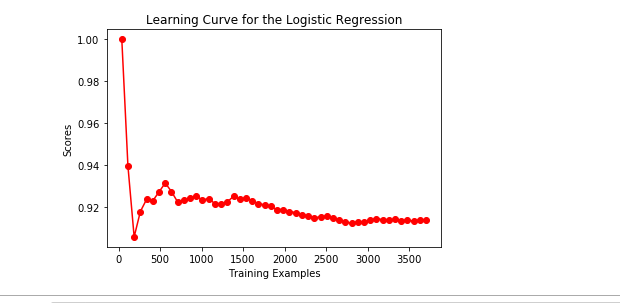
plt.fill\_between(train\_sizes,train\_mean - train\_std,train\_mean + train\_std,alpha=0.1,color="g")

plt.plot(train\_sizes,train\_mean,'o-',color="r",label= "Training Score")

plt.title("Learning Curve for the Logistic Regression")

plt.xlabel("Training Examples")

plt.ylabel("Scores")



plt.plot(train\_sizes,X\_validation\_mean,'o-',color="r",label= "Cross Validation Score")

plt.title("Learning Curve for the Logistic Regression with Cross Validation")

plt.xlabel("Training Examples")

plt.ylabel("Scores")

