Unabridged Review of Supervised Machine Learning Regression and Classification Technique with Practical Task

Adrash Kumar, Pratibha Yadav

Department of Computer Science and Technology, Galgotias University, Greater Noida, U.P.

Abstract *-- Machine Learning is a field of computer science provides the ability to learn without programming and the program explicitly. The computer analyzes by using the data and predicts the next task to perform it was provided some data it is a label or unable. There is a various way to solve the supervised and unsupervised problem. Some problem with Rule-based techniques, Logic-based techniques, Instance-based techniques, Stochastic techniques. This paper presents a complete review of supervised learning. Supervised learning is defined as given a dataset are already know what our correct output is should look like, an idea is that there is a relationship between input and output. We use python to write any code because python is most suitable for Machine Learning also we use more practical view of coding in these paper for better understanding any algorithm.*

I. Introduction

Before know about machine learning and supervised learning techniques important to know the basic introduction of artificial intelligence because in recent days machine learning is mostly used in artificial intelligence for creating very powerful applications or systems because artificial intelligence will shape our future in a more powerful way comparative any other technology in this century. Whenever you don't understand machine learning and artificial intelligence then you will feel like being left behind some technologies, waking up in a world full of technology which you feel more and more like magic. Machine learning is an important part of artificial intelligence because most probably part of artificial intelligence depends on machine learning. machine learning is useful areas for deploying explicitly written algorithms with high-speed performance are unfeasible. Challenges are considered very difficult to train any machine. The machine is actually used for learning many applications that can't be considered only as artificial intelligence.Find the patterns in any data by "machines learn" then analysis about the data information that you get from the world. If provide more data to a machine, then machine analysis more accurate output . But not all data are the same, Imagine you're a student and you want to clear gate exam(one of the toughest exam in India) by the first rank. In order to get the first rank in gate exam, you're going to need sufficient amount of information. Like data, this information can either lead you in the right direction or the wrong direction.Depending on what you want to predict, supervised learning can use to solve any problems by two technique: regression or classification.

Regression technique work to predicting a continuous quantity (continuous output). Continuous means there aren't gaps in the value that Y (output of given data). Classification technique work to predicting a discrete class label output (0 or 1).discrete means there are gaps in the value that Y(Y is the output of given data) can take on it is only in form of true or false otherwise it is 0 or 1.

Few Applications of Machine Learning: -

- **Speech Recognition [1]**
- **Mobile SMS Spam/Not Spam [2]**
- **Detection of heartbeats [3]**
- **Text-to-speech based on NLP [4]**

2. Learning Task for supervised learning

A. Regression:

I) Practically Implement Linear Regression Algorithms (by using Python)

#Import Library like pandas, numpy, other library used in the specific problem.

from sklearn import linear_model

#Identification of different variables/feature in data i.e. textual,numerical,categorial etc.

#Convert textual or categorical variables into numerical in order to make data into a suitable format so ML have applied to it.

#Splitting data into different parts i.e. training set (approx 75% data) & testing set (approx 25% data).

trainingSet = file[:,130]#store 0 to 130

testingSet =file[130,:]#store 130 to n values

```
#seprate training data

trainingX =trainingSet[:,[0,1,2,3]]#first 4 columns #separate features/observation

trainingX = trainingX.astype(float)#convert integer value in float typecast in float

trainingY =trainingSet[:,[4]]#seprate target variable

#seprate testing data

testingX =testingSet[:,[0,1,2,3]]

testingX = testingX.astype(float)#typecast in float

testingY =testingSet[:,[4]]

# Create linear regression object

linear = linear_model.LinearRegression()

# Train  model

linear.fit(trainingX,trainingY)#fit is regalization

linear.score(testingX,testingY)

#Analyzed Output

predicted= linear.predict(testingX)
```

ii) Practically Implement Support Vector Machine Algorithms (by using Python)

```
#Import Library like pandas, numpy, other library used in the specific problem.

from sklearn.svm import SVR

Total_Test , Total_Attribute = 10, 5

np.random.seed(0)

y = np.random.randn(Total_Test)

X = np.random.randn(Total_Test, Total_Attribute)

data = SVR(C=1.0, epsilon=0.2)

data.fit(X, y)

SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.2, gamma='auto',
  kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

iii) Practically Implement Decision Tree Regression Algorithms (by using Python)

```
#import matplotlib,pandas,numpy.

from sklearn.tree import DecisionTreeRegressor

#You get, X (predictor) and Y (target) from given dataset

decision = np.random.RandomState(1)

regression.fit(X, y)
```

iv) Practically Implement Random Forest Regression Algorithms (by using Python)

```
#Import Library like pandas, numpy, other library used in specific problem.

from sklearn.ensemble import RandomForestRegressor

# Create Random Forest object

from sklearn.datasets import make_regression

M, n = make_regression(Total_Features=4,Total_Informative=2,Random_State=0, Shuffle=False)

# Train the model using the training sets and check the score

regression = RandomForestRegressor(max_depth=2, random_state=0)

regression.fit(M, n)

RandomForestRegressor(bootstrap=True,criterion='mse', max_depth=2,max_features='auto',max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2,min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,oob_score=False, random_state=0,verbose=0, warm_start=False)

print(regression.feature_importances_)

print(regression.predict([[0, 0, 0, 0]]))
```

v) Practically Implement Lasso Algorithms (by using Python)

```
#Import Library like pandas, numpy, other library used in the specific problem.

from sklearn.linear_model import Lasso

lassoRegression = Lasso(alpha=0.3, normalize=True)

lassoRegression.fit(X_training,Y_training)

pred = lassoReg.predict(X_CV)

# calculating meanvalue

meanvalue = np.mean((PRED_CV - Y_CV)**2)

meanvalue

#output

lassoRegression.score(X_CV,Y_CV)
```

Note->Now you completed regression techniques.

B. Classification:

i.) Practically Implement Logistic Regression Algorithms (by using Python)

```
#Import Library

from sklearn.linear_model import LogisticRegression

#suppose you get, M (predictor) and n (target) for training data set and M_test(predictor) of test_dataset

# Create logistic regression object

logisticmodel = LogisticRegression()

# Train the model using the training sets and check the score

logisticmodel.fit(M, n)

logisticmodel.score(M, n)

#analysis output

analysis= logisticmodel.predict(M_test)
```

ii) Practically Implement K Nearest Neighbors Algorithms (by using Python)

```
#Import Library

from sklearn.neighbors import KNeighborsClassifier

#suppose you get, X (predictor) and Y (target) data set and x_testing(predictor) of testing_dataset

# Create KNeighbors classifier object model

KNeighborsClassifier(n_neighbors=6) # default value is 5

# Train the model using the training sets and check the score

model.fit(X, y)

#analised Output

analised= model.predict(x_testing)
```

iii) Practically Implement Naïve Bayesian Algorithms (by using Python)

```
#Import Library

from sklearn.naive_bayes import GaussianNB

# there is another distribution for multinomial classes like Bernoulli Naive Bayes, Refer link

model = GaussianNB()

# Train your model using the training sets and check the score

model.fit(M, n)

#Analysied Output

analysed= model.predict(m_testing)
```

iv) Practically Implement Decision Tree (DT) Classification Algorithms (by using Python)

```
#Import Library like pandas, numpy.

from sklearn import tree

#You get, X (predictor) and Y (target) from given dataset

# Create Decision tree object

decisionmodel= tree.DecisionTreeClassifier(criterion='gini')
```

#In classification, you add more technique for more acuracy

Decisionmodel = tree.DecisionTreeRegressor() for regression

decisionmodel.fit(X, y)

decisionmodel.score(X, y)

#analized prediction

analized= model.predict(x_test)

v) Practically Implement SVM classification Algorithms (by using Python)

#Import Library as matplotlib,pandas,numpy.

from sklearn import svm

#we get, X (predictor) and Y (target) in given dataset

#Now model SVM classification object

vectormodel = svm.svc()

there is various option associated with it, this is simple for classification. You can refer link, for mo# re detail.

#We train the model using the training set and testing set

vectormodel.fit(X, y)

vectormodel.score(X, y)

#analised prediction

analised= vectormodel.predict(x_test)

3) Parameters used for evaluating tool performance:

It is not the right way to only consider the value of the accuracy achieved by any classification techniques as a measure of evaluation of the performance of classification algorithms. The correctness or accuracy of the classification is only for the examples related to their actual class. It does not offer other specialties of classified; Relations between data attributes, the right distribution of data examples to each class, the number of positive results from all received positive results and many others. Parameters are described which are evaluated in the evaluation process[8].

		Predicted	
		0	1
Actual	0	TN	FP
	1	FN	TP

CONFUSION MATRIX

Here, TN is True Negative, TP is True Negative, FN is False Negative, FP is False Positive. It shows the number of correct predictions made by the classified.

And it is calculated as: TN+TP/TN+FP+FN+TP

4) Supervised (Regression) practical create a model for housing dataset based on existing features:

You can download USA_Housing data from Kaggle.com or another website. Suppose that problem is that your friend is real estate broker. He was provide you some data it is in form of extension csv txt hologram else any formate. He wants your help to predict housing price in some specific region. Your work is that to predict what is a price of the house using data provided by your friend. Your friend gives you complete data of house amd given data contain 7 column :- Avg. Area Income, Avg. Area House Age, Avg. Area Number of Rooms, Avg. Area Number of Bedrooms, Area Population, Price, Address

Let us start important part as coding part using python

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

housedata = pd.read_csv('USA_Housing.csv')

housedata.head()

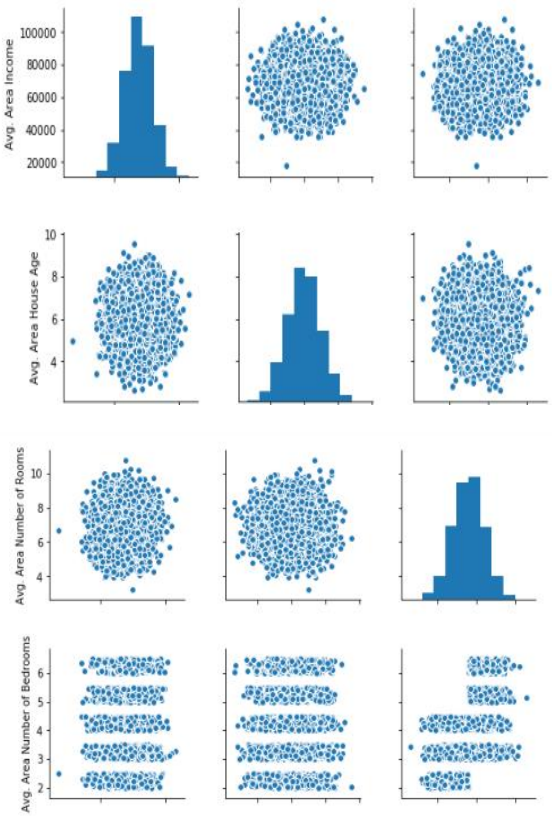
housedata.info()

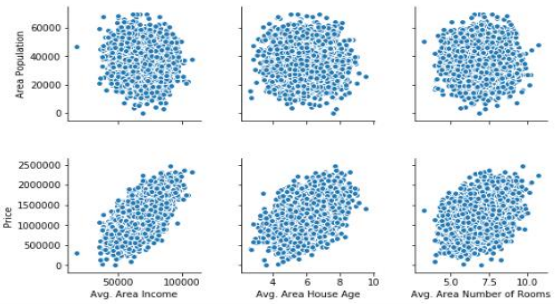
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
Avg. Area Income              5000 non-null float64
Avg. Area House Age           5000 non-null float64
Avg. Area Number of Rooms     5000 non-null float64
Avg. Area Number of Bedrooms  5000 non-null float64
Area Population               5000 non-null float64
Price                         5000 non-null float64
Address                       5000 non-null object
dtypes: float64(6), object(1)
memory usage: 273.5+ KB
```

housedata.columns

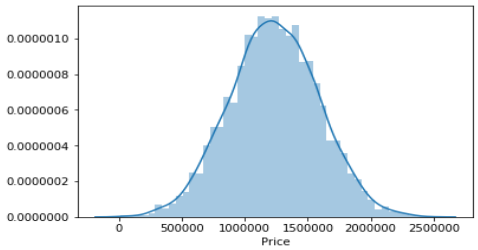
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

sns.pairplot(housedata)

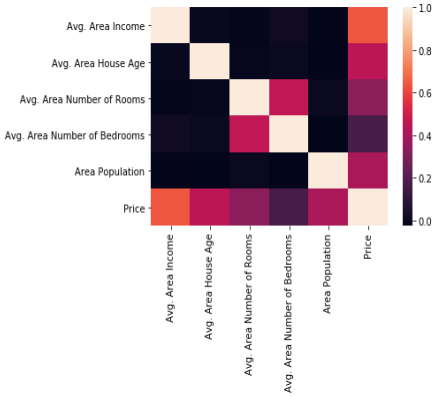




sns.distplot(housedata['Price'])



sns.heatmap(housedata.corr())



#Now impostant task start training your Model

#visulization part completed seprate training and testing set and also seprate X (predictor) and y (target)

X = housedata[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',

'Avg. Area Number of Bedrooms', 'Area Population']]

y = housedata['Price']

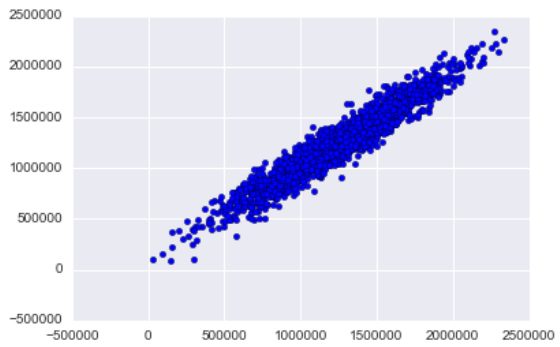
from sklearn.linear_model import LinearRegression

lm = LinearRegression()

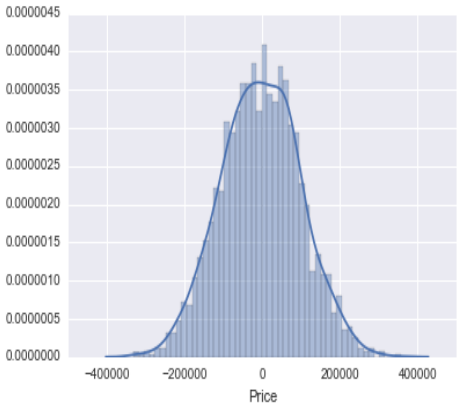
```
lm.fit(X_train,y_train)

#model evaluation

print(lm.intercept_)

coeff_df = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient'])

coeff_df
```

	Coefficient
Avg. Area Income	21.528276
Avg. Area House Age	164883.282027
Avg. Area Number of Rooms	122368.678027
Avg. Area Number of Bedrooms	2233.801864
Area Population	15.150420

```
predictions = lm.predict(X_test)

plt.scatter(y_test,predictions)
```




```
sns.distplot((y_test-predictions),bins=50);
```




```
from sklearn import metrics

print('Mean Absolute Error (MAE) :', metrics.mean_absolute_error(y_test, predictions))

print('Mean Squared Error (MSE):', metrics.mean_squared_error(y_test, predictions))

print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
Mean Absolute Error (MAE) : 82288.22251914957
Mean Squared Error (MSE): 10460958907.209501
Root Mean Squared Error: 102278.82922291153
```

5) Supervised (Classification) practical create a model for iris dataset based on existing features:

You can download iris dataset from Kaggle.com or another website. Suppose that problem is that you want to predict type of species by using features (SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm) .

Let us start important part as coding part using python

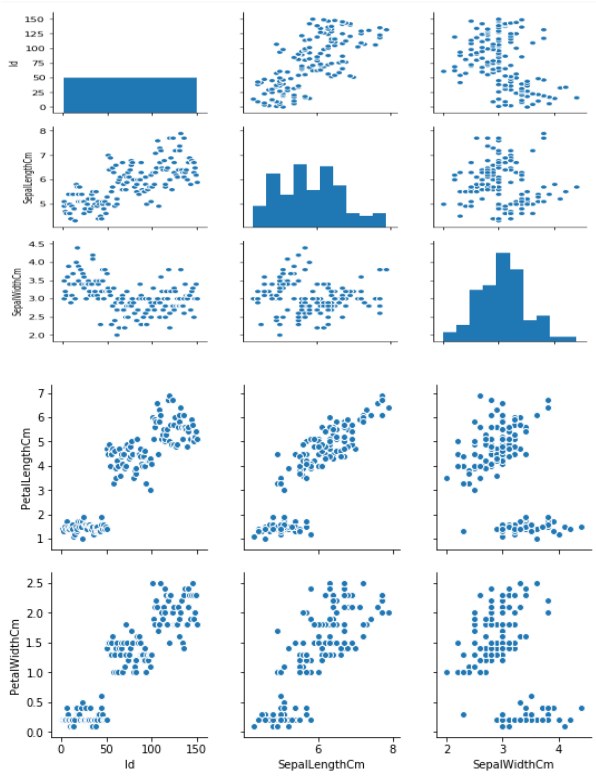
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

irisdata = pd.read_csv("Iris.csv")

irisdata.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

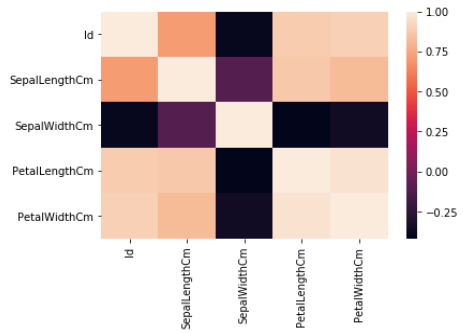
irisdata.describe()

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

sns.pairplot(irisdata)



sns.heatmap(irisdata.corr())



#Now impostant task start training a Logistic Regression Model

Visulization part completed seprate training and testing set and also seprate X (predictor) and y (target)

```
import numpy as np

from numpy import genfromtxt

from sklearn import linear_model

file = genfromtxt("iris.csv",delimiter=",",dtype="str")

dic={}

count = 0

for val in file:

    if val[5] not in dic:

        dic[val[5]]=count

        count +=1

for val in file:

    val[5]=dic[val[5]]

trainingSet = file[:130]

testingset =file[130:]

trainingX =trainingSet[:,[1,2,3,4]]

trainingY =trainingSet[:,[5]]

testingX =testingset[:,[1,2,3,4]]

testingY =testingset[:,[5]]

lr=linear_model.LogisticRegression()

lr.fit(trainingX,trainingY)

lr.score(testingX,testingY)*100
```

#These model get 95 accuracy you also try and do better accuracy score.

6) Conclusion

The primary goal was to prepare an Unabridged Review of Supervised Machine Learning detail ideas and present

different techniques for every supervised learning method classification & Regression technique with practically implemented. This paper makes it a clear vision that every algorithm of supervised machine learning with practically implemented code which help us how to solve real life problem or industrial problem. The choice of an algorithm should be made depending on the type of problem whose data available to you. The accuracy can be increased by using two or more algorithm together in suitable conditions. It also contains a practical model for housing price & Iris dataset which help you the great extent to solve any other machine learning problem by yourself.

Refrences

1) Machine Learning Paradigms for Speech Recognition: An Overview Li Deng, Fellow, IEEE, and Xiao Li, Member, IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING

2) A Review on Mobile SMS Spam Filtering Technique Received January 11, 2017, accepted February 7, 2017, date of publication February 13, 2017

3) Detection of Heartbeats Based on the Bayesian Framework Wen-Long Chin, Jong-Hun Yu, and Cheng-Lung Tseng Department of Engineering Science.

4) An NLP based text-to-speech synthesizer for Moroccan Arabic Rajae Moumen IEEE.

5) Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the lasso". Journal of the Royal Statistical Society. Series B (methodological), Wiley.

6) Santosa, Fadil; Symes, William W. (1986). "Linear inversion of band-limited reflection seismograms". SIAM Journal on Scientific and Statistical Computing.

7) Bicego, M. and Loog, M, “Weighted K-Nearest Neighbor revisited”, 23rd International Conference on Pattern Recognition (ICPR), 2016, pp. 1642-1647

8) Powers, David M W (2011). "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation" (PDF) Journal of Machine Learning Technologies.