ML

DAY2_1

1. Prepare an ML model using KNN Classifier to predict the Species information for a given iris flower using Sepal Length, Sepal Width, Petal Length & Petal Width. Use the complete iris dataset for training. Use it to predict the species of an iris flower.

```
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
iris=load_iris()
X=iris.data
y=iris.target

print(iris.feature_names)
print(X.shape)
print(y.shape)

knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(X,y)

print(knn.predict([[3,5,4,2]]))
print(knn.predict(X))
print(y)
```

2. Print the Accuracy Score and Confusion matrix for KNN Classifier using iris data. (Split iris dataset to train and test sets.)

```
#!/usr/bin/env python
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,accuracy_score

iris_df=load_iris()

X=iris_df.data
y=iris_df.target

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)

knn=KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train,y_train)

predictd_vals_of_y=knn.predict(X_test) #Apply knn on X_test and predict value of y_test


print("")
print("Predicted values of y:")
print(predictd_vals_of_y) #Predicted value of y_test

print("")
print("Actual values of y:")
```

```
print(y_test) #Actual value of y_test

print("")
print("Confusion_matrix :To compare values of Actual(y_test) and Predicted values")
print(confusion_matrix(y_test,predictd_vals_of_y)) #Compare actual values (y_test) and predicted values(prediction)

print("")
print("Accuracy_score")
print(accuracy_score(y_test,predictd_vals_of_y)) #Compare the accuracy score of actual values(y_test) and predicted values(prediction)
print("")
```

day3_1 Prepare an ML model using KMeans algorithm to cluster some sample input generated using make_blob function. Plot the clusters.

```
from sklearn.datasets.samples_generator import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
X,y_true=make_blobs(n_samples=300,centers=4,cluster_std=0.6)
print(X)
plt.scatter(X[:,0],X[:,1])
plt.show()

kmeans=KMeans(n_clusters=4)
kmeans.fit(X)
y_kmeans=kmeans.predict(X)
print(y_kmeans)
plt.scatter(X[:,0],X[:,1],c=y_kmeans,cmap='viridis')
plt.show()
print(confusion_matrix(y_true,y_kmeans))
```

day3_2 Cluster IRIS dataset on the basis of Sepal Length and Sepal Width and visualise them.

```
from sklearn.datasets.samples_generator import make_blobs
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
iris =load_iris()

x=iris.data
print x
plt.scatter(x[:,0],x[:,1])
plt.show()
```

```
kmeans=KMeans(n_clusters=10)
kmeans.fit(x)
y=kmeans.predict(x)
print y
plt.scatter(x[:,0],x[:,1],c=y,cmap='rainbow')
plt.show()
```

mlday3_3 Cluster IRIS dataset on the basis of Petal.Length and Petal.Width and visualise them.

```
from sklearn.datasets.samples_generator import make_blobs
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
iris =load_iris()

x=iris.data
print x
plt.scatter(x[:,2],x[:,3])
plt.show()

kmeans=KMeans(n_clusters=5)
kmeans.fit(x)
y=kmeans.predict(x)
print y
plt.scatter(x[:,2],x[:,3],c=y,cmap='viridis')
plt.show()
```

mlday3_4

Try clustering with some suitable datasets in the link

http://cs.joensuu.fi/sipu/datasets/

(Try with datasets having more than one input columns.)

```
from sklearn.datasets.samples_generator import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_table("http://cs.joensuu.fi/sipu/datasets/s1.txt", delimiter = '\s+')
X = data.as_matrix()
plt.scatter(X[:,0],X[:,1])
plt.show()
kmeans = KMeans(n_clusters = 18)
kmeans.fit(X)
y_kmeans =kmeans.predict(X)
print(y_kmeans)
center =kmeans.cluster_centers_
plt.scatter(X[:,0],X[:,1],c=y_kmeans,cmap = 'viridis', s=50)
plt.scatter(center[:,0],center[:,1], s=200, c='red')
plt.show()
```

assign_1 day5

dlday5_1) Generate a sample dataset using make_moon function and visualize clusters obtain using
a. Kmeans algorithm
b. spectral clustering

```
#!usr/bin/env python
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
from sklearn.cluster import SpectralClustering
X,y_true=make_moons(n_samples=300,noise=0.05,random_state=0)
print(X)

model=SpectralClustering(n_clusters=2, affinity='nearest_neighbors')

#model.fit(X)
y_kmeans=model.fit_predict(X)
print(y_kmeans)
plt.scatter(X[:,0],X[:,1],c=y_kmeans,s=50,cmap='viridis')
#centers=kmeans.cluster_centers_
#plt.scatter(centers[:,0],centers[:,1],s=200,c='red')
plt.show()
```

assign_1 day5
2)Develop a linear regressor for Advertising.csv dataset and print the importatnt metrics for performance evaluation (MSE,RMSE, etc.)

```
#!/usr/bin/env python
import pandas as pd
from sklearn.linear_model import LinearRegression

from sklearn.cross_validation import train_test_split
from sklearn import metrics
import numpy as np

data=pd.read_csv("/home/vimala/mypython/Advertising.csv",index_col=0)
print(data.head())
feature_cols1 = ['TV','radio','newspaper']
feature_cols2 = ['TV','radio']
X1=data[feature_cols1]

X2=data[feature_cols2]

#print(X.head())

y=data['sales']
#print(y)

X_train1,X_test1,y_train1,y_test1=train_test_split(X1,y,test_size=0.4,random_state=4)

X_train2,X_test2,y_train2,y_test2=train_test_split(X2,y,test_size=0.4,random_state=4)
linreg1=LinearRegression()
linreg2=LinearRegression()
linreg1.fit(X_train1,y_train1)
```

```
linreg2.fit(X_train2,y_train2)
#print(linreg.coef_)
#print(linreg.intercept_)



y_pred1=linreg1.predict(X_test1)
y_pred2=linreg2.predict(X_test2)
#print(y_pred)
#print(y_test)


#print(y_pred.shape)
#print(y_test.shape)




#scores.append(metrics.accuracy_score(y_test,y_pred))

#print("Accuracy")
print("Absolute Error")
print("Mean Squared Error")
print("Root Mean Squared Error")

#print( metrics.accuracy_score(y_test,y_pred))
print( metrics.mean_absolute_error(y_test1,y_pred1))
print( metrics.mean_absolute_error(y_test2,y_pred2))
print( metrics.mean_squared_error(y_test1,y_pred1))
print( metrics.mean_squared_error(y_test2,y_pred2))
print( np.sqrt(metrics.mean_squared_error(y_test1,y_pred1)))
print( np.sqrt(metrics.mean_squared_error(y_test2,y_pred2)))
```

assign_2day5

1) Suppose you are the CEO of a restaurant franchise and are considering different cities for opening a new outlet. The chain already has outlets in various cities and you have data for profits and populations from the cities. You would like to use this data to help you select which city to expand to next. The file ex1.txt contains data for the problem. The first column is population of a city and second column is profit. Both values are in 10,000s.

A negative value of profit indicates a loss.
a) Create a scatterplot between population and profits
b) Develop an ML model to predict profit for a given city (by providing population)

```
#!/usr/bin/env python
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

df = pd.read_csv('ex1.txt', header=None)
df.columns = ['pop', 'profit']
df.head()

sns.regplot(x=df['pop'], y=df['profit'], data=df,fit_reg=False);

X = df[['pop']]
y = df['profit']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

lr=LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)

print(y_pred[:5])
print(y_test[:5])
print(mean_squared_error(y_pred, y_test))
plt.scatter(X,y,color="green")
plt.show()
```

set2
mlday5_2.Suppose you are selling your house and you want to see what a good market price would be. The ex2.txt contains a training set of housing prices in India.The first column is the size of the house (in square feet). The second column is the number of bedrooms and the third column is the price of the house.
a) Apply suitable scaling to standardize the data. (Try this later)
b) Use scatter plots to visualize the data
c) Develop an ML model to predict the house price

```
#!/usr/bin/env python

import numpy as np
```

```
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error

df = pd.read_csv('ex2.txt')
df.head()

# w/o scaling

X = df[['area', 'bedrooms']]
y = df['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

lr=LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)

print(y_pred[:5])
print(y_test[:5])
print(mean_absolute_error(y_pred, y_test))
print(mean_squared_error(y_pred, y_test))

# w/ scaling

X = df[['area', 'bedrooms']]
y = df['price']

sc = StandardScaler()
X = sc.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

lr=LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)

print(y_pred[:5])
print(y_test[:5])
print(mean_absolute_error(y_pred, y_test))
print(mean_squared_error(y_pred, y_test))
```

mlday5_3. Suppose that you are the administrator of a university department and you want to determine applicants' chance to get admitted to a course. Admission is based on the scores of two exams. You have the historical data of earlier students. The data contains the marks of two exams and their admission status. Data is available in the file ex3.txt.
a)Visualize the data using scatterplot (score1 in X axis , score2 in Y axis and two different colours for admitted and non admitted students)
b)Develop an ML model to predict the admission status of a new set of students.

```
#!/usr/bin/env python
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

df = pd.read_csv('ex3.txt')
df.head()

#sns.scatterplot(x='score1', y='score2', hue='admission', data=df )

X = df[['score1', 'score2']]
y = df['admission']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

knn=KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

print(y_pred[:5])
print(y_test[:5])
print(accuracy_score(y_pred, y_test))
print(confusion_matrix(y_pred, y_test))

#print(score1)
plt.scatter(X_train["score1"][1:],X_train["score2"][1:])

plt.show()
```

4) The sinking of the Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.
One of the reasons that the shipwreck led to such a loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.
Develop an ML model to predict the survival of passengers.
(Use titanic.csv file)

```
#!/usr/bin/env python
import numpy as np
```

```
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
df = pd.read_csv('titanic.csv')

df =df[['Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']]
df.head()

df['Sex'] = df['Sex'].astype('category')
df['Embarked'] = df['Embarked'].astype('category')
df.info()

df.isnull().sum()

mean_age = df['Age'].mean()
df['Age'].fillna(mean_age, inplace=True)

most_embarked = df['Embarked'].value_counts().idxmax()
df['Embarked'].fillna(most_embarked, inplace=True)

df['Embarked'].value_counts()

d1 = {'male': 1, 'female': 0}
d2 = {'S': 1, 'C': 2, 'Q': 3}

df['Sex'].replace(d1, inplace=True)
df['Embarked'].replace(d2, inplace=True)

df.head()

X = df[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']]
y = df['Survived']

sc = StandardScaler()
X = sc.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

knn=KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

print(y_pred[:5])
print(y_test[:5])
print(accuracy_score(y_pred, y_test))
print(confusion_matrix(y_pred, y_test))
```

set2

mlday5_5) Predicting rock facies (classes of rocks) from well log data

Well log data is recorded either during drilling operations or after the drilling via
tools either on the drill string or wireline tools descended into the well.
Typically, geoscientists would take the logs and make correlations by hand. They

would have to draw lines from log to log to get a holistic view of the rock type/facies, their characteristics, and their content. This can get tedious in mature fields and is prone to likely misinterpretation in new fields.
Train a machine learning model that is able to predict the facies for wells not in the training set.
Data set (mining.csv)
The data set we will use comes from University of Kansas. This dataset was taken
from nine wells with 3232 examples, consisting of a set of seven predictor variables and a rock facies (class).
Features:
GR: gamma ray (GR)
ILD_log10: resistivity logging (ILD_log10)PE: photoelectric effect (PE)
DeltaPHI: neutron-density porosity difference
PHIND: average neutron-density porosity
NM_M: nonmarine-marine indicator
RELPOS: relative position
Rock facies:
Nonmarine sandstone (SS)
Nonmarine coarse siltstone (CSiS)
Nonmarine fine siltstone (FSiS)
Marine siltstone and shale (Sish)
Mudstone (limestone) (MS)
Wackestone (limestone) (WS)
Dolomite (D)
Packstone-grainstone (limestone) (PS)
Phylloid-algal bafflestone (limestone) (BS)

```
#!/usr/bin/env python
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

df = pd.read_csv('mining.csv')
df.head()

df.isnull().sum()

df['Formation'].value_counts()
```

```
df['Well Name'].value_counts()

dummies = pd.get_dummies(df['Formation'])
df = pd.concat([df, dummies], axis=1)
df.drop(['Formation'], inplace=True, axis=1)

dummies = pd.get_dummies(df['Well Name'])
df = pd.concat([df, dummies], axis=1)
df.drop(['Well Name'], inplace=True, axis=1)

df.head()

X = df.drop('Facies', axis=1)
y = df['Facies']

sc = StandardScaler()
X = sc.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

knn=KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

print(y_pred[:5])
print(y_test[:5])
print(accuracy_score(y_pred, y_test))
print(confusion_matrix(y_pred, y_test))
```

mlday6_1.Develop an ML model for iris dataset using DecisionTree Classifier. Print Confusion matrix & Accuracy. Also compute cross val score of accuracy for 20 cross validations with Kfold

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
from sklearn.model_selection import KFold
from sklearn.cross_validation import cross_val_score,KFold

iris=load_iris()
X=iris.data
y=iris.target
#kfold=KFold(n_splits=20,random_state=7)
tree1=DecisionTreeClassifier()
#a=cross_val_score(tree1,X,y,cv=kfold)
#print(a)
tree1.fit(X,y)
p=tree1.predict(X)
```

```
print(confusion_matrix(y,p))
print(accuracy_score(y,p))
print(classification_report(y,p))
```

mlday6_2. Develop an ML model to predict the house price for the boston housing dataset included in sklearn. Find out the RMSE values for models developed using KNNRegressor & DecionTreeRegressor and select the best one. Also plot the scatter diagram showing the actual and predicted values while using DecisionTreeRegressor

```
from sklearn.datasets import load_boston
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt
iris=load_boston()
print(iris.feature_names)
X=iris.data
y=iris.target
#KNeighborsRegressor
tree= KNeighborsRegressor()
tree.fit(X,y)
p=tree.predict(X)
plt.scatter(y,p)
print("KNeighborsRegressor MSE:" , mean_squared_error(y,p))
#DecisionTreeRegressor
tree1=DecisionTreeRegressor()
tree1.fit(X,y)
p=tree1.predict(X)
print("DecisionTreeRegressor MSE:",mean_squared_error(y,p))

plt.scatter(y,p)
```

mlday6_3. Apply DecisionTreeRegressor to the data in https://people.sc.fsu.edu/~jburkardt/datasets/regression/x15.txt Compute the RMSE value, by splitting the data to train set and test set.

```
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
import pandas as pd
df = pd.read_csv('new.csv', delimiter = '\t')
X = df[['tax', 'avg_income', 'paved_highway', 'pop_prop']]
y = df['consumption']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
print(y_test[:6])
knn = KNeighborsRegressor()
knn.fit(X_train, y_train)
p = knn.predict(X_test)
print(p[:6])
print(mean_squared_error(y_test, p))
dt = DecisionTreeRegressor()
```

```
dt.fit(X_train, y_train)
p = dt.predict(X_test)
print(p[:6])
print(mean_squared_error(y_test, p))
```

mlday7_1.Implement a Classifier using Random Forest Classifier for the pimaindians dataset. Last column is the target column.

```
#!/usr/bin/env python
import pandas as pd
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
df=pd.read_csv('/home/ai34/pimaindians.csv',delimiter=',')
print(df.head())
X=df.iloc[:,0:7].values
y=df.iloc[:,8].values
rf=RandomForestClassifier()
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
rf.fit(X_train,y_train)
p=rf.predict(X_test)
print (accuracy_score(y_test,p))
print(confusion_matrix(y_test,p))
```

mlday7_2.Implement a Classifier using Random Forest Classifier for the banking.csv dataset. Last column is the target column.

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

data=pd.read_csv("/home/ai34/banking.csv",delimiter=",")
le =LabelEncoder()

for col in data.columns.values:
        if data[col].dtypes=='object':
                le.fit(data[col].values)
                data[col]=le.transform(data[col])
X=data.iloc[:, 0:19].values
y=data.iloc[:, 20].values

X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.33,)

rf=RandomForestClassifier()
rf.fit(X_train,y_train)
p=rf.predict(X_test)
print(metrics.confusion_matrix(p,y_test))
print("Accuracy:",metrics.accuracy_score(y_test,p))
```

mlday7_3.Take second column from the banking dataset (job) to a variable. Apply Label Encoding & print the result

```
#!usr/bin/env python
import pandas as pd
df=pd.read_csv('/home/ai2/banking.csv',delimiter=',')
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
secondCol=df['job']
le.fit(secondCol)
newCol=le.transform(secondCol)
print(newCol)
```

mlday7_4.Develop an ML model for classification, in the pima indians diabetes dataset using logistic Regression.

```
#!/usr/bin/env python
#accuracy score
#confusion matrix
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
df=pd.read_csv('/home/ai34/mlDay7/pimaindians.csv',delimiter=',')
en=LabelEncoder()
for i in df.columns:
 if(df[i].dtype=='object'):
  df[i]=en.fit_transform(df[i])
X=df.iloc[:,0:7].values
y=df.iloc[:,8].values
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
lr=LogisticRegression()
lr.fit(X_train,y_train)
p=lr.predict(X_test)

print (accuracy_score(y_test,p))
print(confusion_matrix(y_test,p))
```

mlday7_5.Implement a KNN Classifier for the banking.csv dataset. Use only the numeric columns from the dataset as input.

```
import pandas as pd
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
df=pd.read_csv('/home/ai34/mlDay7/pimaindians.csv',delimiter=',')
en=LabelEncoder()
for i in df.columns:
```

```
 if(df[i].dtype=='object'):
  df[i]=en.fit_transform(df[i])
print(df.head())
X= df.drop('y',axis=1)
y= df['y']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
p=knn.predict(X_test)
print(accuracy_score(p,y_test))
print(confusion_matrix(p,y_test))
```

mlday7_6 Implement a KNN Classifier for the banking.csv dataset. Convert the categorical columns to numeric using Label Encoding and use those columns also in the input

```
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd

f1=pd.read_csv('/home/ai34/banking.csv')
le=LabelEncoder()
for col in f1.columns.values:
        if f1[col].dtypes=='object':
                le.fit(f1[col].values)
                f1[col]=le.transform(f1[col])

X=f1.drop('y',axis=1)
y=f1['y']

rf=KNeighborsClassifier()
rf.fit(X,y)
p=rf.predict(X)
print(confusion_matrix(p,y))
print(accuracy_score(p,y))
```

mlday8_1. Naïve Bayes Classifier
1. Apply Naïve Bayes Classifier for Iris dataset

```
from sklearn import datasets
from sklearn.metrics import confusion_matrix
iris = datasets.load_iris()
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
y_pred = gnb.fit(iris.data, iris.target).predict(iris.data)
print(confusion_matrix(iris.target,y_pred))
```

mlday8_2. Apply Naïve Bayes Classifier for the banking dataset

```
#!/usr/bin/env python
import pandas as pd
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
df=pd.read_csv('/home/ai34/banking.csv',delimiter=',')
en= LabelEncoder()
print(df.head())
for i in df.columns:
 if(df[i].dtype=='object'):
  df[i]=en.fit_transform(df[i])
X=df.drop('y',axis=1)
y=df['y']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
rf=GaussianNB()
rf.fit(X_train,y_train)
p=rf.predict(X_test)
print (accuracy_score(p,y_test))
print(confusion_matrix(p,y_test))
```

mlday8_3.Data Preprocessing
Implement an ML model for the bikeshare.csv dataset Regression Problem (https://www.kaggle.com/c/bike-sharing-demand/data)
Evaluate the model by splitting the data using train_test_split function.
Compute Mean Squared Error, also plot the actual values Vs predictions graph
Apply scaling and observe the performance
Apply normalization and observe the performance
Apply MinMaxScaler and observe the performance

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing

df = pd.read_csv('train.csv')

X = df.drop(['datetime','count'], axis=1)
y = df['count']

#Without normalising----------------------------------------------------------------------

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

rf =LinearRegression()
rf.fit(X_train ,y_train)
p = rf.predict(X_test)

print(mean_squared_error(p, y_test))
```

```
plt.scatter(y_test,p)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()

#With normalising----------------------------------------------------------------------------

normalized_X = preprocessing.normalize(X)

X_train, X_test, y_train, y_test = train_test_split(normalized_X, y, test_size=0.3)

rf =LinearRegression()
rf.fit(X_train ,y_train)
p = rf.predict(X_test)

print(mean_squared_error(p, y_test))

plt.scatter(y_test,p)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()

#With scaling--------------------------------------------------------------------------------

X_scaled = preprocessing.scale(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3)

rf =LinearRegression()
rf.fit(X_train ,y_train)
p = rf.predict(X_test)

print(mean_squared_error(p, y_test))

plt.scatter(y_test,p)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()

#With minmaxscaling--------------------------------------------------------------

min_max_scaler = preprocessing.MinMaxScaler()
X_minmax = min_max_scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_minmax, y, test_size=0.3)

rf =LinearRegression()
rf.fit(X_train ,y_train)
p = rf.predict(X_test)

print(mean_squared_error(p, y_test))

plt.scatter(y_test,p)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```

mlday8_4.) Implement an ML model for the Immunotherapy.csv dataset. Evaluate the model by splitting the data using train_test_split function Compute accuracy score and confusion matrix Apply scaling and observe the performance Apply normalization and observe the performance Apply MinMaxScaler and observe the performance (You may not always get better performance)

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.model_selection import train_test_split
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
df = pd.read_csv('Immunotherapy.csv')
X = df.drop('Result_of_Treatment', axis=1)
y = df['Result_of_Treatment']
#Without normalising----------------------------------------------------------------------
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
rf =LogisticRegression()
rf.fit(X_train ,y_train)
p = rf.predict(X_test)
print(accuracy_score(p, y_test))
print(confusion_matrix(p, y_test))
#With normalising-------------------------------------------------------------------------------
normalized_X = preprocessing.normalize(X)
X_train, X_test, y_train, y_test = train_test_split(normalized_X, y, test_size=0.3)
rf =LogisticRegression()
rf.fit(X_train ,y_train)
p = rf.predict(X_test)
print(accuracy_score(p, y_test))
print(confusion_matrix(p, y_test))
#With scaling-----------------------------------------------------------------------------------
X_scaled = preprocessing.scale(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3)
rf =LogisticRegression()
rf.fit(X_train ,y_train)
p = rf.predict(X_test)
print(accuracy_score(p, y_test))
print(confusion_matrix(p, y_test))



#With minmaxscaling-----------------------------------------------------------------
min_max_scaler = preprocessing.MinMaxScaler()
X_minmax = min_max_scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_minmax, y, test_size=0.3)
rf =LogisticRegression()
rf.fit(X_train ,y_train)
p = rf.predict(X_test)
print(accuracy_score(p, y_test))
print(confusion_matrix(p, y_test))
```

Algorithm Comparison using box plot

mlday8_5) generate the box plot showing the comparison of cross validation accuracies of iris dataset (for the following algorithms)

1.KNN
2.Logistic regression
3.Naive Bayes
4.DecisionTree
5.Random Forest

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score,train_test_split,KFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from matplotlib import pyplot as plt
import pandas as pd
url="https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
data=pd.read_csv(url)
data=data.as_matrix()
X=data[:,0:8]
y=data[:,8]
kfold=KFold(10,random_state=7)
models=[]
models.append(("KNN",KNeighborsClassifier()))
models.append(("NB",GaussianNB()))
models.append(("LG",LogisticRegression()))
models.append(("CART",DecisionTreeClassifier()))
models.append(("RFC",RandomForestClassifier()))
results=[]
names=[]
scoring='accuracy'
for name,model in models:
        kfold=KFold(n_splits=10,random_state=7)
        v=cross_val_score(model,X,y,cv=kfold,scoring=scoring)
        results.append(v)
        names.append(name)
        print(name)
        print(v)
fig=plt.figure()
fig.suptitle('Algorithm Comparison')
ax=fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

mlday8_6.generate the box plot showing the comparison of cross validated RMSE values for the boston dataset. Apply any 5 regressors including Ridge and Lasso

```
from sklearn.datasets import load_boston
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import cross_val_score,KFold
from matplotlib import pyplot as plt
data=load_boston()
X=data.data
y=data.target
models=[]
models.append(("KNN",KNeighborsRegressor()))
models.append(("LR",LinearRegression()))
models.append(("DT",DecisionTreeRegressor()))
models.append(("Ridge",Ridge()))
models.append(("Lasso",Lasso()))
results=[]
names=[]
for name,model in models:
        kfold=KFold(n_splits=10,random_state=7)
        v=cross_val_score(model,X,y,cv=kfold,scoring='neg_mean_squared_error')
        results.append(v)
        names.append(name)
        print(name)
        print(v)
fig=plt.figure()
fig.suptitle('Algorithm Comparison')
ax=fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

mlday9 . Develop an ML model for predicting the digits in the digits image dataset in sklearn using SVM . Plot some sample input images together with its class information. Also plot some predicted outputs together with its actual images.

```
#!/usr/bin/env python
from sklearn.datasets import load_digits
from sklearn.metrics import confusion_matrix
from sklearn.svm import SVC
digits=load_digits()
X=digits.data
y=digits.target
print(X.shape)
print(y.shape)
print(digits.images.shape)
#print(y[0])
#print(digits.images[0])
svm=SVC()
svm.fit(X,y)
p=svm.predict(X)
print(confusion_matrix(y,p))
print(y)
print(p)
```

```
import matplotlib.pyplot as plt
i_g=list(zip(digits.images,digits.target))
for index,(image,label) in enumerate(i_g[:8]):
        plt.subplot(2,4,index+1)
        plt.axis('off')
        plt.imshow(image,cmap=plt.cm.gray_r)
        plt.title(label)
plt.show()
```

mlday10_1 a)Apply PCA for the breast_cancer dataset in sklearn to reduce the number of columns to 2 and find out the accuracy score and confusion matrix. Split the data using train_test_split function for training and testing. Use Support Vector Classifier.

```
from sklearn.datasets import load_breast_cancer
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.model_selection import train_test_split
data = load_breast_cancer()
X=data.data
y=data.target
pca = PCA(2)
X_transformed = pca.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_transformed, y, test_size=0.33)
classifier = SVC()
classifier.fit(X_train, y_train)
pred_labels = classifier.predict(X_test)
print(confusion_matrix(y_test,pred_labels))
print(accuracy_score(y_test,pred_labels))
```

mlday10_1 b) do the same for the wine dataset.

```
from sklearn.datasets import load_wine
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.model_selection import train_test_split
data = load_wine()
X=data.data
y=data.target
pca = PCA(2)
X_transformed = pca.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_transformed, y, test_size=0.33)
```

```
classifier = SVC()
classifier.fit(X_train, y_train)
pred_labels = classifier.predict(X_test)
print(confusion_matrix(y_test,pred_labels))
print(accuracy_score(y_test,pred_labels))
```

mlday10_2 a) Compute the RMSE values of the ML model using SVR after applying PCA to reduce the number of columns to 3 in the diabetes dataset. Observe the effect of varying the number of principal components in the model development.

```
from sklearn.datasets import load_diabetes
from sklearn.decomposition import PCA
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from math import sqrt
data = load_diabetes()
X=data.data
y=data.target
pca = PCA(3)
X_transformed = pca.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_transformed, y, test_size=0.33)
classifier = SVR()
classifier.fit(X_train, y_train)
pred_labels = classifier.predict(X_test)
print(sqrt(mean_squared_error(y_test,pred_labels)))
```

mlday10_2 b) do the same for boston dataset.

```
from sklearn.datasets import load_boston
from sklearn.decomposition import PCA
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from math import sqrt
data = load_boston()
X=data.data
y=data.target
pca = PCA(3)
X_transformed = pca.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_transformed, y, test_size=0.33)
classifier = SVR()
classifier.fit(X_train, y_train)
pred_labels = classifier.predict(X_test)
print(sqrt(mean_squared_error(y_test,pred_labels)))
```

3) Develop an ML model for the breast_cancer dataset in sklearn using LDA. (from sklearn.discriminant_analysis import LinearDiscriminantAnalysis) Print the accuracy and confusion_matrix.

```
from sklearn.datasets import load_breast_cancer
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
data = load_breast_cancer()
X=data.data
y=data.target
pca = PCA(2)
```

```
X_transformed = pca.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_transformed, y, test_size=0.33)
classifier = LinearDiscriminantAnalysis()
classifier.fit(X_train, y_train)
pred_labels = classifier.predict(X_test)
print(confusion_matrix(y_test,pred_labels))
print(accuracy_score(y_test,pred_labels))
```

open cv

opencvday1_1) Create a black image of 512X512 dimension and fill region 150X150 with green color. Display the image.

```
import numpy as np
import cv2 as cv
img=np.zeros((512,512,3))
img[1:100,1:100]=(0,0,0)
print(img)
cv.imshow('d2',img)
cv.waitKey(0)
```

opencvday1_2) Create a black image of 512X512 dimension.
a. Add a line from the point (10,10) to (500,10) in red color
b. Add a line from the point (20,20) to (500,20) in green
c. Draw a rectangle with corner points at (30,30) and (100,100)
d. Add a circle at any position with radius 30
e. Add an ellipse at (150,300) with axes 100 &75 and fill the ellipse with green color
f. Add a Text “Computer Vision” at 10,400 position with font size 2

```
import numpy as np
import cv2 as cv
img=np.zeros((512,512,3))
img[1:100,1:100]=(0,0,0)
cv.line(img,(10,10),(500,10),(0,0,255),5)
cv.line(img,(20,20),(500,20),(0,255,0),5)
cv.rectangle(img,(30,30),(100,100),(255,158,255),3)
cv.circle(img,(300,300),30,(0,255,255),-1)
cv.ellipse(img,(150,300),(100,75),0,0,360,(0,255,0),-1)
font=cv.FONT_HERSHEY_SIMPLEX
cv.putText(img,'Computer Vision',(10,400),font,2,(255,0,255),2)
print(img)
cv.imshow('d2',img)
cv.waitKey(0)
```

opencvday1_3) Read an image file and write to another file after doing not operation on all points.

```
import pandas as pd
import cv2 as cv
img=cv.imread("/home/ai34/walking.jpg")
print(img)
img=cv.bitwise_not(img)
#cv.imshow('image'.img)
#cv.waitKey(0)
cv.imwrite("invertedwalk.jpg",img)
```

opencvday2_1. Write a program to apply translation with translation factors tx=200 and ty=100 on the image of apple.

```
#!/usr/bin/env python
import numpy as np
import cv2 as cv
img=cv.imread('apple.jpg')
M=np.float32([[1,0,200],[0,1,100]])
rows,cols,c=img.shape
dst=cv.warpAffine(img,M,(cols,rows))
cv.imshow('dst',dst)
cv.waitKey(0)
cv.imshow('img',img)
cv.waitKey(0)
```

opencvday2_2. Write a program apply scaling on an image with scaling factor 2 on both x and y axes. Try the same on both color and black and white images.

```
#!/usr/bin/env python
import numpy as np
import cv2 as cv
img=cv.imread('messi.jpg',0)
rows,cols=img.shape
print(img.shape)
#dst=cv.warpAffine(img,M,(cols,rows))
dst=cv.resize(img,None,fx=2,fy=2,interpolation=cv.INTER_CUBIC)
#dst=img.resize(342,548)
cv.imshow('img',img)
cv.waitKey(0)
cv.imshow('dst',dst)
cv.waitKey(0)
```

opencvday2_3. Write a program to rotate an image 45 degrees.

```
import pandas as pd
import cv2 as cv
img= cv.imread('messi.jpg')
rows,cols,c=img.shape
M=cv.getRotationMatrix2D(((cols-1)/2,(rows-1)/2),45,1)
dst=cv.warpAffine(img,M,(cols,rows))
print(rows,cols)
cv.imshow('img',img)
cv.waitKey(0)
cv.imshow('dst',dst)
cv.waitKey(0)
```

opencvday2_4.Write a program to rotate an image 45 degrees and apply scaling of 1.5 in both axes.

```
import cv2 as cv
import numpy as np
```

```
img = cv.imread("cat.jpg")
rows,cols,c = img.shape

M = cv.getRotationMatrix2D(((cols-1)/2,(rows-1)/2),45,1.5)
dest = cv.warpAffine(img,M,(cols,rows))
pic = np.hstack((img,dest))
cv.imshow('output',pic)
cv.waitKey(0)
```

Opencvday2_5. Write a program to apply various thresholding operations on an image gradient.jpg. Apply the same with other similar images and analyze the results.

THRESH_BINARY
THRESH_BINARY_INV
THESH_TRUNC
THRESH_TOZERO
THRESH_TOZERO_INV

```
#!/usr/bin/env python
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('gradient.jpg',0)
ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO_INV)
titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]
for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
print(ret)
```

Opencvday2_6. Apply Edge detection using Canny in any two images

```
import cv2 as cv
import pandas as pd
img1=cv.imread('face.jpg')
print(img1.shape)
img2=cv.imread('cat1.jpg')
gst=cv.Canny(img2,0,70)
dst=cv.Canny(img1,0,100)
cv.imshow('face',img1)
cv.waitKey(0)
cv.imshow('dst',dst)
cv.waitKey(0)
cv.imshow('cat1',img2)
cv.waitKey(0)
```

```
cv.imshow('gst',gst)
cv.waitKey(0)

cv.destroyAllWindows()
```

Opencvday2_7. Apply Perspective Transformation on the Sudoku/ or any other image to improve clarity.

```
#!usr/bin/env python
import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread('sudoku.png')
rows,cols,ch = img.shape
print(rows)
print(cols)
pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]])
pts2 = np.float32([[0,0],[500,0],[0,500],[500,500]])
M = cv2.getPerspectiveTransform(pts1,pts2)
dst = cv2.warpPerspective(img,M,(500,500))
plt.subplot(121),plt.imshow(img),plt.title('Input')
plt.subplot(122),plt.imshow(dst),plt.title('Output')
plt.show()
```

Opencvday3_1.Apply face and eyes detection on team.jpg image

```
import numpy as np
import cv2
faces=cv2.CascadeClassifier('faces.xml')
img=cv2.imread('team.jpg')
gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

dfaces=faces.detectMultiScale(gray)
print(len(dfaces))
for(x,y,w,h) in dfaces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
        eyes=cv2.CascadeClassifier('eyes.xml')
        roi_gray=gray[y:y+h,x:x+w]
        roi_color=img[y:y+h,x:x+w]
        deyes=eyes.detectMultiScale(roi_gray)
        for (ex,ey,ew,eh) in deyes:
                cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)


cv2.imshow('dst',img)
cv2.waitKey(0)
```

Opencvday3_2.Apply car detection on car.jpg image

```
import numpy as np
import cv2
cars=cv2.CascadeClassifier('cars.xml')
img=cv2.imread('car.jpg')
img1=cv2.imread('car1.jpg')
gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

dfaces =cars.detectMultiScale(gray)
print(len(dfaces))
for (x,y,w,h) in dfaces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    cv2.rectangle(img1,(x,y),(x+w,y+h),(255,0,0),2)

cv2.imshow('dst',img)
```

```
cv2.waitKey(0)
cv2.imshow('car',img1)
print(gray.shape)
cv2.waitKey(0)
```

Opencvday3_3.Apply cat detection on cats.jpg file

```
import numpy as np
import cv2
face_cascade = cv2.CascadeClassifier('cats.xml')

#eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread('cats.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray)

print(len(faces))
for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

    #roi_gray = gray[y:y+h, x:x+w]
    #roi_color = img[y:y+h, x:x+w]
    #eyes = eye_cascade.detectMultiScale(roi_gray)
    #for (ex,ey,ew,eh) in eyes:
     #   cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
#            print(ex,ey)
cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Opencvday3_4.Apply erosion and dilation on Alphabet Image, Chessboard

```
import cv2
import numpy as np
# Reading the input image
img = cv2.imread('j.png', 0)
# Taking a matrix of size 5 as the kernel
kernel = np.ones((5,5), np.uint8)
img_erosion = cv2.erode(img, kernel, iterations=1)
img_dilation = cv2.dilate(img, kernel, iterations=1)
cv2.imshow('Input', img)
cv2.imshow('Erosion', img_erosion)
cv2.imshow('Dilation', img_dilation)
cv2.waitKey(0)
img1 = cv2.imread('chessboard.png',0)
kernel = np.ones((5,5), np.uint8)
img_erosion1 = cv2.erode(img1, kernel, iterations=1)
img_dilation1 = cv2.dilate(img1, kernel, iterations=1)
cv2.imshow('Input1', img1)
cv2.imshow('Erosion1', img_erosion1)
cv2.imshow('Dilation1', img_dilation1)
cv2.waitKey(0)
```

Opencvday3_5.Do noise removal in the girl.jpg image using blur functions

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread('girl.jpg')
blur = cv2.blur(img,(5,5))
#cv2.imshow('dst',blur)
#cv2.waitKey(0)
blur2 = cv2.GaussianBlur(img,(5,5),0)

median=cv2.medianBlur(img,5)
#blur = cv2.bilateralFilter(img,9,75,75)
h=np.hstack((img,blur,blur2
cv2.imshow('dst',h)
cv2.waitKey(0)
```

Opencvday3_6.Apply sobel feature detection functions on the ‘building.png’ picture.

```
import  cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread('building.jpg',0)
sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=1)
sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=1)
dst= np.hstack((sobelx,sobely))
cv2.imshow('dsti1',img)
cv2.waitKey(0)
cv2.imshow('dst',dst)
cv2.waitKey(0)
```

Opencvday3_7.Try to improve the readability of the song (sonnet.gif) by applying suitable image manipulation method.

Opencvday3_8.Improve the readability of the sudoku2.png image by applying suitable image manipulation methods. A sample answer is given in sudokuclear.png image

```
#!/usr/bin/env python
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('sudoku.png',0)

ret,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,11,2)
th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v = 127)',
          'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
images = [img, th1, th2, th3]

for i in range(4):
    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

Opencvday3_9.Apply edge detection on the coins.jpg image

```
#!/usr/bin/env python
import cv2

img = cv2.imread('coins.jpg')

edges = cv2.Canny(img, 100, 200)

cv2.imshow('img', edges)
cv2.waitKey(0) # waits until a key is pressed
cv2.destroyAllWindows()
```