EXP1: SCIKIT LEARN

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn.linear_model
def prepare_country_stats(oecd_bli,gdp_per_capita):
    oecd_bli=oecd_bli[oecd_bli["INEQUALITY"]=="TOT"]
    oecd_bli=oecd_bli.pivot(index="Country",columns="Indicator",values="Value")
    gdp_per_capita.rename(columns={"2015":"GDP per capita"},inplace=True)
    gdp_per_capita.set_index("Country",inplace=True)

full_country_stats=pd.merge(left=oecd_bli,right=gdp_per_capita,left_index=True,right_index=Tr
ue)
    full_country_stats.sort_values(by="GDP per capita",inplace=True)
    remove_indices=[0,1,6,8,33,34,35]
    keep_indices=list(set(range(36))-set(remove_indices))
    return full_country_stats[["GDP per capita",'Life satisfaction']].iloc[keep_indices]
oecd_bli=pd.read_csv("C:/Users/CC-220/Desktop/datasets/oecd_bli_2015.csv",thousands=",")
gdp_per_capita=pd.read_csv("C:/Users/CC-220/Desktop/datasets/gdp_per_capita.csv",thousan
ds=",",delimiter='\t',encoding='latin1',na_values="n/a")
country_stats=prepare_country_stats(oecd_bli,gdp_per_capita)
x=np.c_[country_stats["GDP per capita"]]
y=np.c_[country_stats["Life satisfaction"]]
country_stats.plot(kind='scatter',x="GDP per capita",y="Life satisfaction")
plt.show()
linear_reg_model=sklearn.linear_model.LinearRegression()
linear_reg_model.fit(x,y)
x_new=[[22587]]
print(linear_reg_model.predict(x_new))
```

OUTPUT:----------
<IMG>
[[5.96242338]]

EXP2:K-NEAREST NEIGHBOR

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

```
import sklearn.neighbors
def prepare_country_stats(oecd_bli,gdp_per_capita):
    oecd_bli=oecd_bli[oecd_bli["INEQUALITY"]=="TOT"]
    oecd_bli=oecd_bli.pivot(index="Country",columns="Indicator",values="Value")
    gdp_per_capita.rename(columns={"2015":"GDP per capita"},inplace=True)
    gdp_per_capita.set_index("Country",inplace=True)

full_country_stats=pd.merge(left=oecd_bli,right=gdp_per_capita,left_index=True,right_index=Tr
ue)
    full_country_stats.sort_values(by="GDP per capita",inplace=True)
    remove_indices=[0,1,6,8,33,34,35]
    keep_indices=list(set(range(36))-set(remove_indices))
    return full_country_stats[["GDP per capita",'Life satisfaction']].iloc[keep_indices]
oecd_bli=pd.read_csv("C:/Users/CC-220/Desktop/datasets/oecd_bli_2015.csv",thousands=",")
gdp_per_capita=pd.read_csv("C:/Users/CC-220/Desktop/datasets/gdp_per_capita.csv",thousan
ds=",",delimiter='\t',encoding='latin1',na_values="n/a")
country_stats=prepare_country_stats(oecd_bli,gdp_per_capita)
x=np.c_[country_stats["GDP per capita"]]
y=np.c_[country_stats["Life satisfaction"]]
country_stats.plot(kind='scatter',x="GDP per capita",y="Life satisfaction")
plt.show()
knn=sklearn.neighbors.KNeighborsRegressor(n_neighbors=3)
knn.fit(x,y)
x_new=[[22587]]
print(knn.predict(x_new))
```

OUTPUT:----------

<IMG>

```
[[5.76666667]]
```

EXP3:DIGIT IMAGE CLASSIFIER

```
from sklearn.datasets import fetch_openml
mnist=fetch_openml('mnist_784',version=1,cache=True)
x,y=mnist["data"],mnist["target"]
print(x.shape)
#% matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
some_digit=x[26000]
some_digit_image=some_digit.reshape(28,28)
plt.imshow(some_digit_image,cmap=matplotlib.cm.binary,interpolation="nearest")
```

```
plt.axis("off")
plt.show()
x_train,x_test,y_train,y_test=x[:60000],x[60000:],y[:60000],y[60000:]
print(y_train)
import numpy as np
y_train=y_train.astype(np.int8)
print(y_train)
y_train_4=(y_train==4)
print(y_train_4)
y_test_4=(y_test==4)
from sklearn.linear_model import SGDClassifier
sgd_clf=SGDClassifier(random_state=42)
sgd_clf.fit(x_train,y_train_4)
sgd_clf.predict([some_digit])
```

OUTPUT:----------

```
(70000, 784)
<IMG>
['5' '0' '4' ... '5' '6' '8']
[5 0 4 ... 5 6 8]
[False False  True ... False False False]
array([ True])
```

EXP4:LINEAR REGRESSION

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets,linear_model
from sklearn.metrics import mean_squared_error,r2_score
diabetes=datasets.load_diabetes()
diabetes_x=diabetes.data[:,np.newaxis,2]
diabetes_x_train=diabetes_x[:-20]
diabetes_x_test=diabetes_x[-20:]
diabetes_y_train=diabetes.target[:-20]
diabetes_y_test=diabetes.target[-20:]
regr=linear_model.LinearRegression()
regr.fit(diabetes_x_train,diabetes_y_train)
diabetes_y_pred=regr.predict(diabetes_x_test)
print('Coefficients:\n',regr.coef_)
print('Intercept:\n',regr.intercept_)
print("Mean Squared Error:%.2f"%mean_squared_error(diabetes_y_test,diabetes_y_pred))
print("Variance score:%.2f"%r2_score(diabetes_y_test,diabetes_y_pred))
```

```
plt.scatter(diabetes_x_test,diabetes_y_test,color='black')
plt.plot(diabetes_x_test,diabetes_y_pred,color='blue',linewidth=3)
plt.show()
```

OUTPUT:----------
Coefficients:
 [938.23786125]
Intercept:
 152.91886182616167
Mean Squared Error:2548.07
Variance score:0.47
<IMG>

EXP5:K-MEANS CLUSTER

```
from pandas import DataFrame
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
Data={'x':[25,34,22,27,33,33,31,22,35,34,67,54,57,43,50,57,59,52,65,47,49,48,35,33,44,45,38,43,51,46],

'y':[79,51,53,78,59,74,73,57,69,75,51,32,40,47,53,36,35,58,59,50,25,20,14,12,20,5,29,27,8,7]}
df=DataFrame(Data,columns=['x','y'])
kmeans=KMeans(n_clusters=3).fit(df)
centroids=kmeans.cluster_centers_
print(centroids)
plt.scatter(df['x'],df['y'],c=kmeans.labels_.astype(float))
plt.scatter(centroids[:,0],centroids[:,1],c='red')
```

OUTPUT:----------
[[55.1 46.1]
 [43.2 16.7]
 [29.6 66.8]]
<matplotlib.collections.PathCollection at 0x1ed4290ea48>

?<IMG>

EXP6:SVM

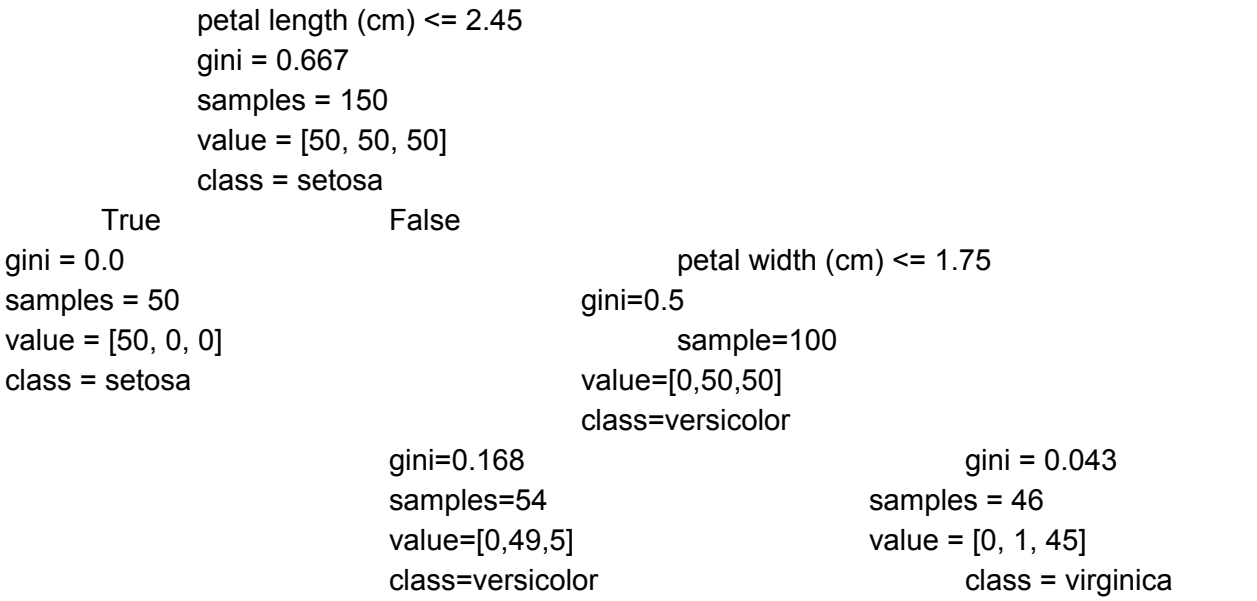
```
import numpy as np
from sklearn import datasets
```

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
iris=datasets.load_iris()
x=iris["data"][:,(2,3)]
y=(iris["target"]==2).astype(np.float64)
svm_clf=Pipeline((
    ("scaler",StandardScaler()),
    ("linear_svc",LinearSVC(C=1,loss="hinge")),
    ))
svm_clf.fit(x,y)
svm_clf.predict([[5.5,1.7]])

OUTPUT:----------
array([1.])

----------------------------------------------------------------------------------------------------------------------------
EXP7:DECISION TREE

from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
iris=load_iris()
x=iris.data[:,2:]
y=iris.target
tree_clf= DecisionTreeClassifier(max_depth=2)
tree_clf.fit(x,y)
from sklearn.tree import export_graphviz
export_graphviz(

tree_clf,out_file="C:/Users/CC-220/Desktop/iris_tree.dot",feature_names=iris.feature_names[2:],
class_names=iris.target_names,rounded=True,
filled=True)
print(tree_clf.predict_proba([[5,1.5]]))
print(tree_clf.predict([[5,1.5]]))

OUTPUT:----------
[[0.         0.90740741 0.09259259]]
[1]
                              ==============================
IRIS_TREE_DOT:::::::

digraph Tree {
node [shape=box, style="filled, rounded", color="black", fontname=helvetica] ;
```

```
edge [fontname=helvetica] ;
0 [label="petal length (cm) <= 2.45\ngini = 0.667\nsamples = 150\nvalue = [50, 50, 50]\nclass =
setosa", fillcolor="#ffffff"] ;
1 [label="gini = 0.0\nsamples = 50\nvalue = [50, 0, 0]\nclass = setosa", fillcolor="#e58139"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="petal width (cm) <= 1.75\ngini = 0.5\nsamples = 100\nvalue = [0, 50, 50]\nclass =
versicolor", fillcolor="#ffffff"] ;
0 -> 2 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
3 [label="gini = 0.168\nsamples = 54\nvalue = [0, 49, 5]\nclass = versicolor", fillcolor="#4de88e"]
;
2 -> 3 ;
4 [label="gini = 0.043\nsamples = 46\nvalue = [0, 1, 45]\nclass = virginica", fillcolor="#843de6"] ;
2 -> 4 ;
}
```

COPY PASTE IN WEBGRAPHVIZ.COM

OUTPUT----------



EXP8:ENSEMBLING LEARNING

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_moons
x,y=make_moons(n_samples=500,noise=0.30,random_state=42)
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42)
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
log_clf=LogisticRegression(solver="liblinear",random_state=42)
rnd_clf=RandomForestClassifier(n_estimators=10,random_state=42)
svm_clf=SVC(gamma="auto",random_state=42)
voting_clf=VotingClassifier(estimators=[('lr',log_clf),('rf',rnd_clf),('svc',svm_clf)],voting='hard')
from sklearn.metrics import accuracy_score
for clf in(log_clf,rnd_clf,svm_clf,voting_clf):
    clf.fit(x_train,y_train)
    y_pred=clf.predict(x_test)
    print(clf.__class__.__name__,accuracy_score(y_test,y_pred))
```

OUTPUT:----------

```
LogisticRegression 0.864
RandomForestClassifier 0.872
SVC 0.888
VotingClassifier 0.896
```

EXP9:RANDOM FORESTS

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_moons
import numpy as np
x,y=make_moons(n_samples=500,noise=0.30,random_state=42)
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42)
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
bag_clf=BaggingClassifier(
        DecisionTreeClassifier(splitter="random",max_leaf_nodes=16,random_state=42),
        n_estimators=500,max_samples=100,bootstrap=True,n_jobs=-1,random_state=42)
bag_clf.fit(x_train,y_train)
y_pred=bag_clf.predict(x_test)
from sklearn.ensemble import RandomForestClassifier
rnd_clf=RandomForestClassifier(n_estimators=500,max_leaf_nodes=16,n_jobs=-1,random_stat
e=42)
rnd_clf.fit(x_train,y_train)
y_pred_rf=rnd_clf.predict(x_test)
np.sum(y_pred==y_pred_rf)/len(y_pred)
```

OUTPUT:----------

0.976

EXP10:PRINCIPAL COMPONENT ANALYSIS

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_openml
mnist=fetch_openml('mnist_784',version=1,cache=True)
x,y=mnist["data"],mnist["target"]
x_train,x_test,y_train,y_test=train_test_split(x,y)
from sklearn.decomposition import PCA
import numpy as np
pca=PCA()
pca.fit(x_train)
cumsum=np.cumsum(pca.explained_variance_ratio_)
d=np.argmax(cumsum>=0.95)+1
print(d)
print(np.sum(pca.explained_variance_ratio_))
pca=PCA(n_components=154)
x_reduced=pca.fit_transform(x_train)
x_recovered=pca.inverse_transform(x_reduced)
print(x_reduced)
print(x_recovered)
```

OUTPUT:----------

```
154
1.0


[[ 171.00488262 1239.20257787   26.69374341 ...  -34.39209948
   -50.03492705   14.97357795]
 [1469.41314734 -233.51890486  608.55056023 ...   58.90450709
    14.97604655  -30.25324641]
 [-302.78376958  543.14969473 -414.74979418 ...  -25.62959597
   -25.84069167    3.864527  ]
 ...
 [-697.52419736  531.51598096 -155.40705833 ...  -11.41043906
    -9.34867903   27.5779096 ]
 [-436.26921333 -459.43820068  498.38198557 ...   -6.88277724
    52.00848603   46.07475921]
```

```
 [-473.24820757   21.35682881 -393.05617554 ...   11.6697899
    3.70974614   32.95969374]]
[[ 5.28181540e-14  5.93722329e-15 -2.19772507e-13 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [-7.47749030e-14  1.90324165e-14 -2.71639523e-13 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 2.40144141e-14  1.27721788e-14 -1.37859584e-13 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 ...
 [ 8.53480091e-14 -8.86963005e-15 -8.16765290e-14 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [-7.13047570e-14 -5.90244528e-14 -5.52995828e-14 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 1.90662717e-14  9.22878074e-14  4.25094694e-14 ...  0.00000000e+00
   0.00000000e+00  0.00000000e+00]]
```

----------ADDITIONAL ----------

```
class Animal:
    def speak(self):
        print("Animal Speaking")
class Dog(Animal):
    def bark(self):
        print("dog barking")
class DogChild(Dog):
    def eat(self):
        print("Eating")
d=DogChild()
d.bark()
d.speak()
d.eat()
```

OUTPUT:----------

```
dog barking
Animal Speaking
Eating
```

```
import matplotlib.pyplot as plt
```

```
x=[1,2.0,2.25,3.0,4.0]
y=[8,8.5,9.5,10.5,11.5]
x1=[9,10.5,11.5,12.5,13.5]
y1=[9.5,10,11,12.5,14]
plt.scatter(x,y,label='high in c,low save')
plt.scatter(x1,y1,label='low in c,low save',color='b')
plt.xlabel("sav")
plt.ylabel("inc")
plt.legend()
plt.show()
```

OUTPUT:----------

<IMG>

```
import matplotlib.pyplot as plt
pop=[22,55,67,45,21,42]
bins=[0,10,20,30,40,50,60]
plt.hist(pop,bins,histtype='bar',rwidth=0.8)
plt.xlabel("agegroup")
plt.ylabel("noofpeople")
plt.title("histogram")
plt.show()
```

OUTPUT:----------

<IMG>

```
import pandas as pd
df=pd.read_csv("http://rcs.bu.edu/examples/python/data_analysis/Salaries.csv")
print(df[10:12])
df_rank=df.groupby(['rank'])
print(df_rank.mean())
df_sorted=df.sort_values(by='service')
print(df_sorted)
```

OUTPUT:----------

```
   rank discipline  phd  service   sex  salary
10  Prof          B   39       33  Male  128250
11  Prof          B   23       23  Male  134778
              phd    service         salary
```

```
rank
AssocProf  15.076923  11.307692   91786.230769
AsstProf    5.052632   2.210526   81362.789474
Prof       27.065217  21.413043  123624.804348
       rank discipline  phd  service     sex  salary
55  AsstProf          A    2        0  Female   72500
23  AsstProf          A    2        0    Male   85000
43  AsstProf          B    5        0  Female   77000
17  AsstProf          B    4        0    Male   92000
12  AsstProf          B    1        0    Male   88000
..       ...        ...  ...      ...     ...     ...
40      Prof          A   39       36  Female  137000
27      Prof          A   45       43    Male  155865
36      Prof          B   45       45    Male  146856
0       Prof          B   56       49    Male  186960
9       Prof          A   51       51    Male   57800

[78 rows x 6 columns]

----------------------------------------------------------------

import pandas as pd
df=pd.read_csv("http://rcs.bu.edu/examples/python/data_analysis/Salaries.csv")
print(df.head())
df['salary'].dtype
print(df['salary'])

OUTPUT:----------

 rank discipline  phd  service   sex  salary
0  Prof          B   56       49  Male  186960
1  Prof          A   12        6  Male   93000
2  Prof          A   23       20  Male  110515
3  Prof          A   40       31  Male  131205
4  Prof          B   20       18  Male  104800
0     186960
1      93000
2     110515
3     131205
4     104800
       ...
73    105450
74    104542
```

```
75    124312
76    109954
77    109646
Name: salary, Length: 78, dtype: int64
```

```
import numpy as np
import matplotlib.pyplot as plt
x=np.arange(0,3*np.pi,0.1)
y=np.sin(x)
#plt.plot(x,y)
yy=np.sin(x)
y1=np.cos(x)
plt.plot(x,yy)
plt.plot(x,y1)
plt.show()
```

OUTPUT:----------

<IMG>

```
import numpy as np
from scipy.spatial.distance import pdist,squareform
x=np.array([(0,1),(1,0),(2,0)])
print(x)
d=squareform(pdist(x,'euclidean'))
print(d)
```

OUTPUT:----------

```
[[0 1]
 [1 0]
 [2 0]]
[[0.         1.41421356 2.23606798]
 [1.41421356 0.         1.        ]
 [2.23606798 1.         0.        ]]
```

```
class A:
    def __init__(self,a,b,c):
        self.a=a
```

```
        self.b=b
        self.c=c
    def summ(self):
        self.x=self.a+self.b+self.c
        print("sum is:",self.x)
    def area(self):
        self.s=self.x**2
        print("area %d"%self.s)

b=A(2,3,4)
b.summ()
b.area()
```

OUTPUT:----------

```
sum is: 9
area 81
```

```
m=[]
n=int(input("enter n"))
for i in range (n):
    m.append(int(input()))
s=int(input("enter search"))
t=0
for i in range(n):
    if(s==m[i]):
        print("element {} found at {} position".format(s,i))
        t=1
if(t==0):

    print("element not found")
```

OUTPUT:----------

```
enter n5
3
4
5
5
6
enter search6
```

element 6 found at 4 position

```
n=int(input())
for i in range(1,n+1):
    j=1
    c=0
    while(j<=i):
        if(i%j==0):
            c=c+1
        j=j+1
    if(c==2):
        print(i)
```

OUTPUT:----------

9
2
3
5
7

```
def rev(n):
    r=0
    c=0
    while(n>0):
        c=(n%10)
        r=(10*r+c)
        n=int(n/10)
    print(r)
rev(1234)
```

OUTPUT:----------

4321