DATA SCIENCE LAB PROGRAMS

1. Demonstrate all the basic plots using Matplotlib package and python programming. import matplotlib.pyplot as plt

import numpy as np

# Generate some data for plotting x = np.linspace(0, 10, 100)

y = np.sin(x) plt.figure() plt.plot(x,y) plt.title("Line Chart")

categories=['A','B','C','D'] values=[20,35,30,25] plt.figure() plt.bar(categories,values) plt.title("Bar Chart")

x=np.random.randn(100) y=np.random.randn(100) colors=np.random.rand(100) sizes=100\*np.random.rand(100) plt.figure()

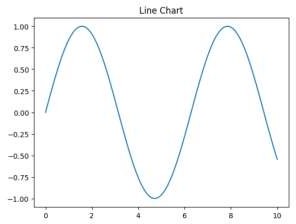
plt.scatter(x,y,c=colors, s=sizes, alpha=0.5) plt.title("Scatter Plot")

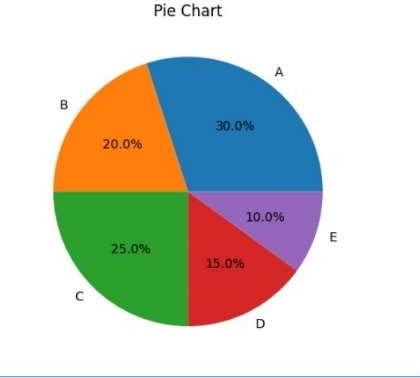
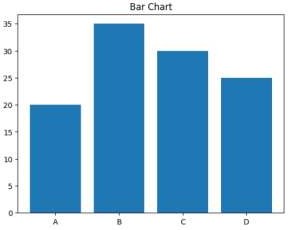
sizes = [30, 20, 25, 15, 10] labels = ['A', 'B', 'C', 'D', 'E']

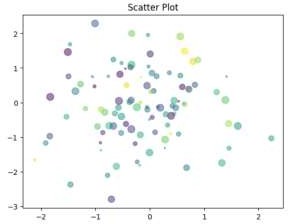
plt.figure()

plt.pie(sizes, labels=labels, autopct="%1.1f%%") plt.title("Pie Chart")

plt.show()







1. Implement a python program to perform File Operations on Excel Dataset.

import pandas as pd df=pd.read\_excel('data.xlsx') print("First few rows") print(df.head())

print("\n Summary statistics:") print(df.describe()) filtered\_data=df[df['Age']>30] print("\n Filtered data(Age>30):") print(filtered\_data)

sorted\_data=df.sort\_values(by='salary',ascending=False) print("\nSorted data(by Salary):")

print(sorted\_data) df['Bonus']=df['salary']\*0.1

print("\n Data with new column(Bonus)") print(df) df.to\_excel('Output.xlsx',index=False) print("\n Data written to output.xlsx")

OUTPUT:

First few rows

Name Age salary 0 Jones 25 11000

1 Kivell 46 56000

2 Jardine 54 35000

3 Gill import pandas as pd df=pd.read\_excel('data.xlsx') print("First few rows") print(df.head())

print("\n Summary statistics:") print(df.describe()) filtered\_data=df[df['Age']>30] print("\n Filtered data(Age>30):") print(filtered\_data)

sorted\_data=df.sort\_values(by='salary',ascending=False) print("\nSorted data(by Salary):")

print(sorted\_data) df['Bonus']=df['salary']\*0.1

print("\n Data with new column(Bonus)") print(df) df.to\_excel('Output.xlsx',index=False) print("\n Data written to output.xlsx")

28 67000

4 Sorvino 45 33000

|  |  |  |
| --- | --- | --- |
| Summary statisti  Age | | cs:  salary |
| count | 21.000000 | 21.000000 |
| mean | 39.285714 | 53404.761905 |
| std | 11.895978 | 50922.396607 |
| min | 23.000000 | 9000.000000 |
| 25% | 30.000000 | 30500.000000 |
| 50% | 38.000000 | 35000.000000 |
| 75% | 45.000000 | 62000.000000 |
| max | 67.000000 | 220000.000000 |

|  |  |  |  |
| --- | --- | --- | --- |
| Fi | ltered dat  Name | a(Age>30):  Age salary | |
| 1 | Kivell | 46 | 56000 |
| 2 | Jardine | 54 | 35000 |
| 4 | Sorvino | 45 | 33000 |
| 5 | Jones | 34 | 21000 |
| 6 | Andrews | 67 | 54000 |
| 8 | Thompson | 33 | 65000 |
| 11 | Howard | 38 | 23000 |
| 12 | Parent | 56 | 9000 |
| 13 | Jones | 45 | 34000 |
| 14 | Smith | 36 | 31000 |
| 15 | Jones | 56 | 170000 |
| 17 | Jones | 45 | 65000 |
| 18 | Parent | 33 | 62000 |
| 19 | Kivell | 38 | 41000 |
| 20 | Smith | 40 | 30500 |

|  |  |  |  |
| --- | --- | --- | --- |
| Sor | ted data(b  Name | y Salary): Age salary | |
| 7 | Jardine | 23 | 220000 |
| 15 | Jones | 56 | 170000 |
| 3 | Gill | 28 | 67000 |
| 8 | Thompson | 33 | 65000 |
| 17 | Jones | 45 | 65000 |
| 18 | Parent | 33 | 62000 |
| 1 | Kivell | 46 | 56000 |
| 6 | Andrews | 67 | 54000 |
| 9 | Jones | 29 | 45000 |
| 19 | Kivell | 38 | 41000 |
| 2 | Jardine | 54 | 35000 |
| 10 | Morgan | 30 | 34000 |
| 13 | Jones | 45 | 34000 |
| 4 | Sorvino | 45 | 33000 |
| 14 | Smith | 36 | 31000 |
| 20 | Smith | 40 | 30500 |
| 11 | Howard | 38 | 23000 |
| 5 | Jones | 34 | 21000 |
| 16 | Morgan | 24 | 15000 |
| 0 | Jones | 25 | 11000 |
| 12 | Parent | 56 | 9000 |

Data with new column(Bonus)

Name Age

Jones 25

Kivell 46

Jardine 54

Gill 28

Sorvino 45

Jones 34

Andrews 67

Jardine 23

Thompson 33

Jones 29

Morgan 30

Howard 38

Parent 56

Jones 45

Smith 36

Jones 56

Morgan 24

Jones 45

Parent 33

Kivell 38

Smith 40

salary 11000

56000

35000

67000

33000

21000

54000

220000

65000

45000

34000

23000

9000

34000

31000

170000

15000

65000

62000

41000

30500

Bonus

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

1100.0

5600.0

3500.0

6700.0

3300.0

2100.0

5400.0

22000.0

6500.0

4500.0

3400.0

2300.0

900.0

3400.0

3100.0

17000.0

1500.0

6500.0

6200.0

4100.0

3050.0

Data written to output.xlsx

1. Write a python program to perform Array operations using the Numpy package.

import numpy as np # Create arrays

a = np.array([1, 2, 3, 4, 5])

b = np.array([6, 7, 8, 9, 10])

print("Array a", a) print("Array b", b)

print("Sum of array a and b", np.add(a,b)) print("Difference of array a and b", np.subtract(a,b)) print("Product of arrays a and b", np.multiply(a,b)) print("Division of arrays a and b", np.divide(a,b)) print("Square root of array a:",np.sqrt(a)) print("Exponential of array a:",np.exp(a)) print("Minimum value of array a:",np.min(a)) print("Maximum value of array b:",np.max(b)) print("Mean of array a:",np.mean(a)) print("Standard deviation of array b:",np.std(b)) print("Sum of elements in array a:",np.sum(a)) c=np.array([[1,2],[3,4],[5,6]])

print("Array c:") print(c)

print("Reshaped array c:") print(np.reshape(c,(2,3)))

|  |  |  |
| --- | --- | --- |
| d=np.array([[1,2,3],[4,5,6]]) |  | |
| print("Array d:") |
| print(d) |
| print("Transposed array d:") |
| print(np.transpose(d)) |
| **Output:** |
| Array a [1 2 3 4 5] |
| Array b [ 6 7 8 9 10]  Sum of array a and b [ 7 9 11 13 15] |
| Difference of array a and b [-5 -5 -5 -5 -5]  Product of arrays a and b [ 6 14 24 36 50] |
| Division of arrays a and b [0.16666667 0.28571429 0.375 | 0.44444444 0.5 | ] |
| Square root of array a: [1. 1.41421356 1.73205081 2. | 2.23606798] |  |

Exponential of array a: [ 2.71828183 7.3890561 20.08553692 54.59815003 148.4131591 ]

Minimum value of array a: 1 Maximum value of array b: 10 Mean of array a: 3.0

Standard deviation of array b: 1.4142135623730951 Sum of elements in array a: 15

Array c:

[[1 2]

[3 4]

[5 6]]

Reshaped array c: [[1 2 3]

[4 5 6]]

Array d:

[[1 2 3]

[4 5 6]]

Transposed array d: [[1 4]

[2 5]

[3 6]]

1. Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.

import numpy as np x=np.array(([2,9],[1,9],[3,6]),dtype=float)

y=np.array(([92],[86],[89]),dtype=float) x=x/np.amax(x,axis=0)

y=y/100

def sigmoid(x):

return 1/(1+np.exp(-x))

def derivation\_sigmoid(x): return x\*(1-x)

epoch=5000 lr=0.1

inputlayer\_neurons=2 hiddenlayer\_neurons=3 outputlayer\_neurons=1

wb=np.random.uniform(size=(inputlayer\_neurons,hiddenlayer\_neurons)) bb=np.random.uniform(size=(1,hiddenlayer\_neurons)) wout=np.random.uniform(size=(hiddenlayer\_neurons,outputlayer\_neurons)) bout=np.random.uniform(size=(1,outputlayer\_neurons))

for i in range(epoch): hinp1=np.dot(x,wb)

hinp=hinp1+bb hlayer\_act=sigmoid(hinp) outinp1=np.dot(hlayer\_act,wout) outinp=outinp1+bout output=sigmoid(outinp)

EO=y-output outgrad=derivation\_sigmoid(output) d\_output=EO\*outgrad EH=d\_output.dot(wout.T) hiddengrad=derivation\_sigmoid(hlayer\_act) d\_hiddenlayer=EH\*hiddengrad wout+=hlayer\_act.T.dot(d\_output)\*lr wb+=x.T.dot(d\_output)\*lr

print("Inpput:\n" +str(x)) print("Actual:\n" +str(y)) print("Predicted:\n",output)

OUTPUT:-

Inpput:

[[0.66666667 1. ]

[0.33333333 1. ]

[1. 0.66666667]]

Actual: [[0.92]

[0.86]

[0.89]]

Predicted: [[0.89184048]

[0.88433366]

[0.89399225]]

1. Demonstrate Linear Regression operation using python programming. import pandas as pd

import numpy as np

import matplotlib.pyplot as plt import seaborn as sns

dataset = pd.read\_csv('advertising.csv') dataset.head(10)

dataset.shape dataset.isna().sum() dataset.duplicated().any()

fig, axs = plt.subplots(3, figsize = (5,5))

plt1 = sns.boxplot(dataset['TV'], ax = axs[0])

plt2 = sns.boxplot(dataset['Newspaper'], ax = axs[1]) plt3 = sns.boxplot(dataset['Radio'], ax = axs[2]) plt.tight\_layout()

sns.distplot(dataset['Sales']);

sns.pairplot(dataset, x\_vars=['TV', 'Radio', 'Newspaper'], y\_vars='Sales', height=4, aspect=1, kind='scatter')

plt.show()

sns.heatmap(dataset.corr(), annot = True) plt.show()

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression from sklearn import metrics

x = dataset[['TV']] y = dataset['Sales']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.3, random\_state = 100) slr= LinearRegression()

slr.fit(x\_train, y\_train) print('Intercept: ', slr.intercept\_) print('Coefficient:', slr.coef\_)

print('Regression Equation: Sales = 6.948 + 0.054 \* TV') plt.scatter(x\_train, y\_train)

plt.plot(x\_train, 6.948 + 0.054\*x\_train, 'r') plt.show()

#Prediction of Test and Training set result y\_pred\_slr= slr.predict(x\_test) x\_pred\_slr= slr.predict(x\_train)

print("Prediction for test set: {}".format(y\_pred\_slr))

slr\_diff = pd.DataFrame({'Actual value': y\_test, 'Predicted value': y\_pred\_slr}) slr\_diff

#Predict for any value slr.predict([[56]])

# print the R-squared value for the model from sklearn.metrics import accuracy\_score

print('R squared value of the model: {:.2f}'.format(slr.score(x,y)\*100)) OUTPUT:

[https://classroom.google.com/c/NTc4MTM5NzE3NDYx/m/NjQ4ODAwOTUyMTg2/det](https://classroom.google.com/c/NTc4MTM5NzE3NDYx/m/NjQ4ODAwOTUyMTg2/details) [ails](https://classroom.google.com/c/NTc4MTM5NzE3NDYx/m/NjQ4ODAwOTUyMTg2/details)

1. Train a regularized logistic regression classifier on the in-build iris dataset using scikit- learns. Train the model and report the best classification accuracy.

# Importing the necessary libraries import numpy as np

import matplotlib.pyplot as plt import pandas as pd

# Importing the dataset

dataset = pd.read\_csv('iris.csv') dataset.describe() dataset.info()

# Splitting the dataset into the Training set and Test set X = dataset.iloc[:, [0,1,2, 3]].values

y = dataset.iloc[:, 4].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0) # Feature Scaling

from sklearn.preprocessing import StandardScaler sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train) X\_test = sc.transform(X\_test)

# Fitting Logistic Regression to the Training set

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 0, solver='lbfgs', multi\_class='auto')

classifier.fit(X\_train, y\_train) # Predicting the Test set results

y\_pred = classifier.predict(X\_test) # Predict probabilities

probs\_y=classifier.predict\_proba(X\_test) probs\_y = np.round(probs\_y, 2)

res = "{:<10} | {:<10} | {:<10} | {:<13} | {:<5}".format("y\_test", "y\_pred", "Setosa(%)", "versicolor(%)", "virginica(%)\n")

res += "-"\*65+"\n"

res += "\n".join("{:<10} | {:<10} | {:<10} | {:<13} | {:<10}".format(x, y, a, b, c) for x, y, a, b, c in zip(y\_test, y\_pred, probs\_y[:,0], probs\_y[:,1], probs\_y[:,2]))

res += "\n"+"-"\*65+"\n" print(res)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix cm = confusion\_matrix(y\_test, y\_pred) print(cm)

# Plot confusion matrix import seaborn as sns import pandas as pd

# confusion matrix sns heatmap

## https:/[/www.kaggle.com/](http://www.kaggle.com/agungor2/various-confusion-matrix-plots)a[gungor2/various-confusion-matrix-plots](http://www.kaggle.com/agungor2/various-confusion-matrix-plots) ax = plt.axes()

df\_cm = cm

sns.heatmap(df\_cm, annot=True, annot\_kws={"size": 30}, fmt='d',cmap="Blues", ax = ax ) ax.set\_title('Confusion Matrix')

plt.show() OUTPUT:

<https://classroom.google.com/c/NTc4MTM5NzE3NDYx/m/NjY1Mjg0NzYwOTkz/details>

1. Write a python program to perform Data Manipulation operations using Pandas package. import pandas as pd

data={

'Name':['John','Emma','Sant','Lisa','Tom'], 'Age':[25,30,28,32,27],

'Country':['USA','Canada','India','UK','Australia'], 'Salary':[50000,60000,70000,80000,65000]

}

df=pd.DataFrame(data) print("Original DataFrame") print(df)

name\_age=df[['Name','Age']] print("Original DataFrame") print(df) name\_age=df[['Name','Age']] print("Name and Age columns") print(name\_age)

filtered\_df=df[df['Country']=='USA'] print("\nfiltered DataFrame(Country='USA')") print(filtered\_df)

sorted\_df=df.sort\_values("Salary",ascending=False) print("\nsorted DataFrame(by ssalary in descending order)") print(sorted\_df)

average\_Salary=df['Salary'].mean()

print("\nAverage salary",average\_Salary)

df['Experience']=[3,6,4,8,5] print("\nDataFrame with added experience") print(df) df.loc[df['Name']=='Emma','Salary']=65000

print("\nDataFrame with updating emma salary") print(df)

df.drop('Experience',axis=1) print("\nDataFrame after deleting the column ") print(df)

OUTPUT:-

Original DataFrame

Name Age Country Salary 0 John 25 USA 50000

* 1. Emma 30 Canada 60000
  2. Sant 28 India 70000 3 Lisa 32 UK 80000

4 Tom 27 Australia 65000 Original DataFrame

Name Age Country Salary 0 John 25 USA 50000

1. Emma 30 Canada 60000
2. Sant 28 India 70000 3 Lisa 32 UK 80000

4 Tom 27 Australia 65000 Name and Age columns

Name Age

1. John 25
2. Emma 30
3. Sant 28
4. Lisa 32
5. Tom 27

filtered DataFrame(Country='USA') Name Age Country Salary

0 John 25 USA 50000

sorted DataFrame(by ssalary in descending order)

|  |  |  |
| --- | --- | --- |
|  | Name Age | Country Salary |
| 3 | Lisa 32 | UK 80000 |
| 2 | Sant 28 | India 70000 |
| 4 | Tom 27 | Australia 65000 |
| 1 | Emma 30 | Canada 60000 |
| 0 | John 25 | USA 50000 |

Average salary 65000.0

DataFrame with added experience

Name Age Country Salary Experience

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 John 25 | USA 50000 | 3 |  | |
| 1 Emma 30 | Canada 60000 |  |  | 6 |
| 2 Sant 28 | India 70000 | 4 |  |  |
| 3 Lisa 32 | UK 80000 | 8 |  |  |
| 4 Tom 27 | Australia 65000 |  | 5 |  |

DataFrame with updating emma salary

Name Age Country Salary Experience

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 John 25 | USA 50000 | 3 |  |  |
| 1 Emma 30 | Canada 65000 |  |  | 6 |
| 2 Sant 28 | India 70000 | 4 |  |  |
| 3 Lisa 32 | UK 80000 | 8 |  |  |
| 4 Tom 27 | Australia 65000 |  | 5 |  |

DataFrame after deleting the column

Name Age Country Salary Experience

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 John 25 | USA 50000 | 3 |  | |
| 1 Emma 30 | Canada 65000 |  |  | 6 |
| 2 Sant 28 | India 70000 | 4 |  |  |
| 3 Lisa 32 | UK 80000 | 8 |  |  |
| 4 Tom 27 | Australia 65000 |  | 5 |  |