RAJALAKSHMI ENGINEERING COLLEGE

Approved by AICTE | Affiliated to Anna University | Accredited by NAAC

Department of Computer Science and Engineering

CS23334 Fundamentals of Data Science Lab

III semester II Year (2023R)

Name of the Student : BA MADHUJA

Register Number :240701296

jupyter madhujads Last Checkpoint: 14 minutes ago

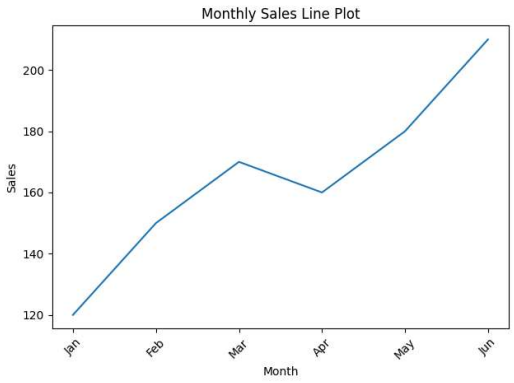
File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[5]:
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("sample_sales.csv")

plt.plot(df["Month"], df["Sales"])
plt.xlabel("Month")
plt.ylabel("Sales")
plt.title("Monthly Sales Line Plot")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[ ]:

jupyter madhujads Last Checkpoint: 22 minutes ago

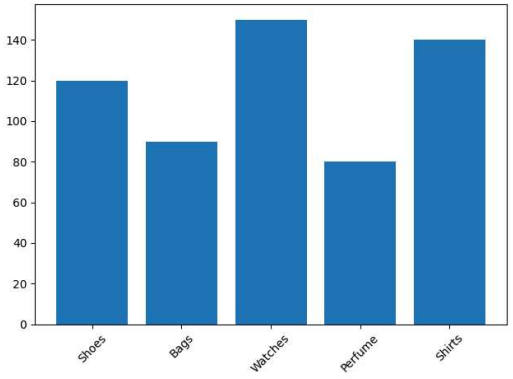
File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[10]:
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("product_sales.csv")

plt.bar(df["Product"], df["Sales"])
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[ ]:

jupyter madhujads Last Checkpoint: 24 minutes ago

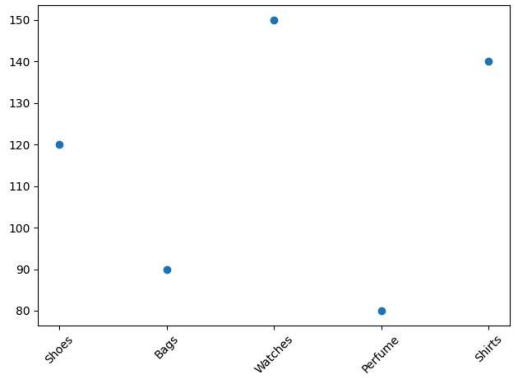
File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[11]:
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("product_sales.csv")

plt.scatter(df["Product"], df["Sales"])
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[ ]:

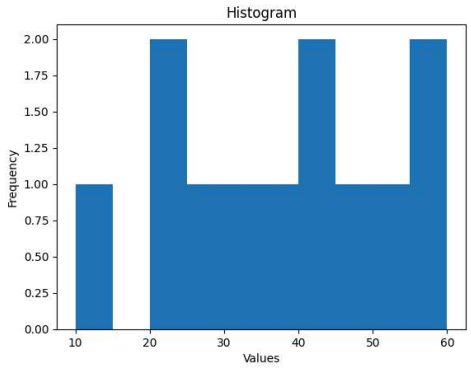
jupyter madhujads Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Trusted

Code JupyterLab Python 3 (ipykernel)

[27]:
```
import matplotlib.pyplot as plt

data = [10, 20, 22, 25, 30, 35, 40, 42, 45, 50, 55, 60]

plt.hist(data)
plt.xlabel("Values")
plt.ylabel("Frequency")
plt.title("Histogram")
plt.show()
```



[ ]:

jupyter madhujads Last Checkpoint: 25 minutes ago

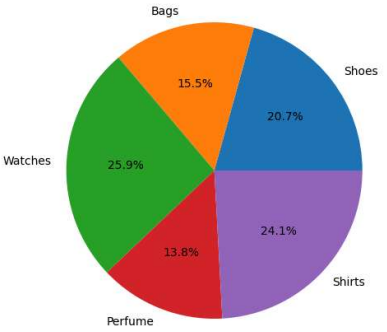
File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[12]:
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("product_sales.csv")

plt.pie(df["Sales"], labels=df["Product"], autopct="%1.1f%%")
plt.tight_layout()
plt.show()
```



[ ]:

jupyter madhujads Last Checkpoint: 30 minutes ago

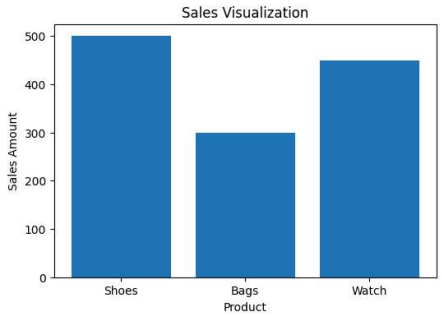
File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[13]:
```
import pandas as pd
import matplotlib.pyplot as plt

data = {
    "Product": ["Shoes", "Bags", "Watch", "Shoes", "Bags"],
    "Sales": [500, 300, 450, None, 250]
}

df = pd.DataFrame(data)
df.to_csv("sample_sales.csv", index=False)

df = pd.read_csv("sample_sales.csv")
df["Sales"] = df["Sales"].fillna(df["Sales"].mean())
df["Product"] = df["Product"].astype(str)

plt.figure(figsize=(6,4))
plt.bar(df["Product"], df["Sales"])
plt.xlabel("Product")
plt.ylabel("Sales Amount")
plt.title("Sales Visualization")
plt.show()
```



jupyter madhujads Last Checkpoint: 33 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[15]:
```
import pandas as pd
import numpy as np

df = pd.DataFrame({
    "Height": [150, 160, 170, 180, 190],
    "Weight": [55, 60, 72, 80, 90]
})

# Min-Max scaling to [0,1]
def minmax_scale(df):
    res = (df - df.min()) / (df.max() - df.min())
    return res

# Standardization (Z-score)
def standard_scale(df):
    res = (df - df.mean()) / df.std(ddof=0)   # population std (ddof=0)
    return res

df_minmax = minmax_scale(df)
df_standard = standard_scale(df)

print("Min-Max:\n", df_minmax)
print("\nStandardized:\n", df_standard)
```

```
Min-Max:
    Height    Weight
0    0.00  0.000000
1    0.25  0.142857
2    0.50  0.485714
3    0.75  0.714286
4    1.00  1.000000

Standardized:
      Height    Weight
0 -1.414214 -1.281250
1 -0.707107 -0.890625
2  0.000000  0.046875
3  0.707107  0.671875
4  1.414214  1.453125
```

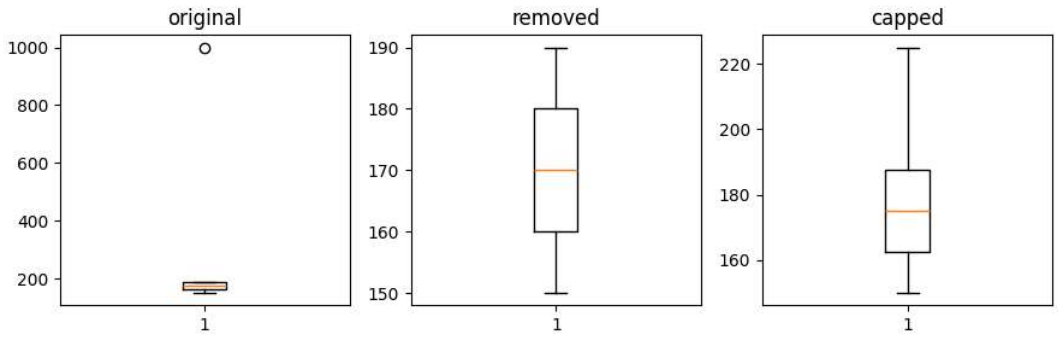
[ ]:

jupyter madhujads Last Checkpoint: 35 minutes ago

File Edit View Run Kernel Settings Help

Code JupyterLab Python 3

[16]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

try:
    df = pd.read_csv("your_file.csv")
except:
    df = pd.DataFrame({"Height":[150,160,170,180,190,1000],"Weight":[55,60,72,80,90,300]})

num = df.select_dtypes(include=[np.number]).columns.tolist()

def iqr_mask(s,k=1.5):
    q1,q3 = s.quantile(0.25), s.quantile(0.75)
    i = q3 - q1
    return (s < q1 - k*i) | (s > q3 + k*i)

mask = df[num].apply(iqr_mask)
df_removed = df.loc[~mask.any(axis=1)].reset_index(drop=True)
df_capped = df.copy()
for c in num:
    q1,q3 = df[c].quantile(0.25), df[c].quantile(0.75)
    i = q3 - q1
    low,up = q1 - 1.5*i, q3 + 1.5*i
    df_capped[c] = df[c].clip(low,up)

if num:
    col = num[0]
    plt.figure(figsize=(9,3))
    plt.subplot(1,3,1); plt.boxplot(df[col].dropna()); plt.title("original")
    plt.subplot(1,3,2); plt.boxplot(df_removed[col].dropna()); plt.title("removed")
    plt.subplot(1,3,3); plt.boxplot(df_capped[col].dropna()); plt.title("capped")
    plt.tight_layout(); plt.show()
```



[ ]:

jupyter madhujads Last Checkpoint: 37 minutes ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

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

try:
    df = pd.read_csv("your_file.csv")
except:
    df = pd.DataFrame({
        "Age":[23,45,31,35,28,120],
        "Salary":[40000,80000,54000,60000,45000,999999],
        "Dept":["HR","IT","IT","Sales","HR","IT"]
    })

print("ROWS, COLS:", df.shape)
print("\nHEAD:\n", df.head())
print("\nINFO:")
df.info()
print("\nDESCRIPTION:\n", df.describe(include="all").T)
print("\nMISSING VALUES:\n", df.isnull().sum())
print("\nUNIQUE VALUES (sample):")
for c in df.columns:
    print(c, "->", df[c].nunique(), "unique")

num = df.select_dtypes(include=[np.number]).columns.tolist()
cat = df.select_dtypes(include=["object","category"]).columns.tolist()

if num:
    print("\nCORRELATION:\n", df[num].corr())
    df[num].hist(figsize=(8, 4*len(num)))
    plt.tight_layout()
    plt.show()

for c in cat:
    vc = df[c].value_counts().head(10)
    print(f"\nVALUE COUNTS for {c}:\n", vc)
    vc.plot(kind="bar")
    plt.title(c)
    plt.tight_layout()
    plt.show()

if num:
    plt.figure(figsize=(6,6))
    plt.imshow(df[num].corr(), interpolation="nearest")
    plt.xticks(range(len(num)), num, rotation=45)
    plt.yticks(range(len(num)), num)
    plt.colorbar()
    plt.title("Correlation matrix")
    plt.tight_layout()
    plt.show()
```

ROWS, COLS: (6, 3)

HEAD:

```
   Age  Salary   Dept
0   23   40000     HR
1   45   80000     IT
2   31   54000     IT
3   35   60000  Sales
4   28   45000     HR
```

INFO:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Age     6 non-null      int64
 1   Salary  6 non-null      int64
 2   Dept    6 non-null      object
dtypes: int64(2), object(1)
memory usage: 276.0+ bytes
```

DESCRIPTION:

```
       count unique  top freq      mean          std      min      25% \
Age      6.0    NaN  NaN  NaN     47.0     36.523965     23.0    28.75
Salary   6.0    NaN  NaN  NaN  213166.5  385719.913841  40000.0  47250.0
Dept       6      3   IT    3      NaN          NaN      NaN      NaN

          50%      75%       max
Age       33.0     42.5     120.0
Salary  57000.0  75000.0  999999.0
```

Dept NaN NaN NaN

MISSING VALUES:

Age 0

Salary 0

Dept 0

dtype: int64

UNIQUE VALUES (sample):

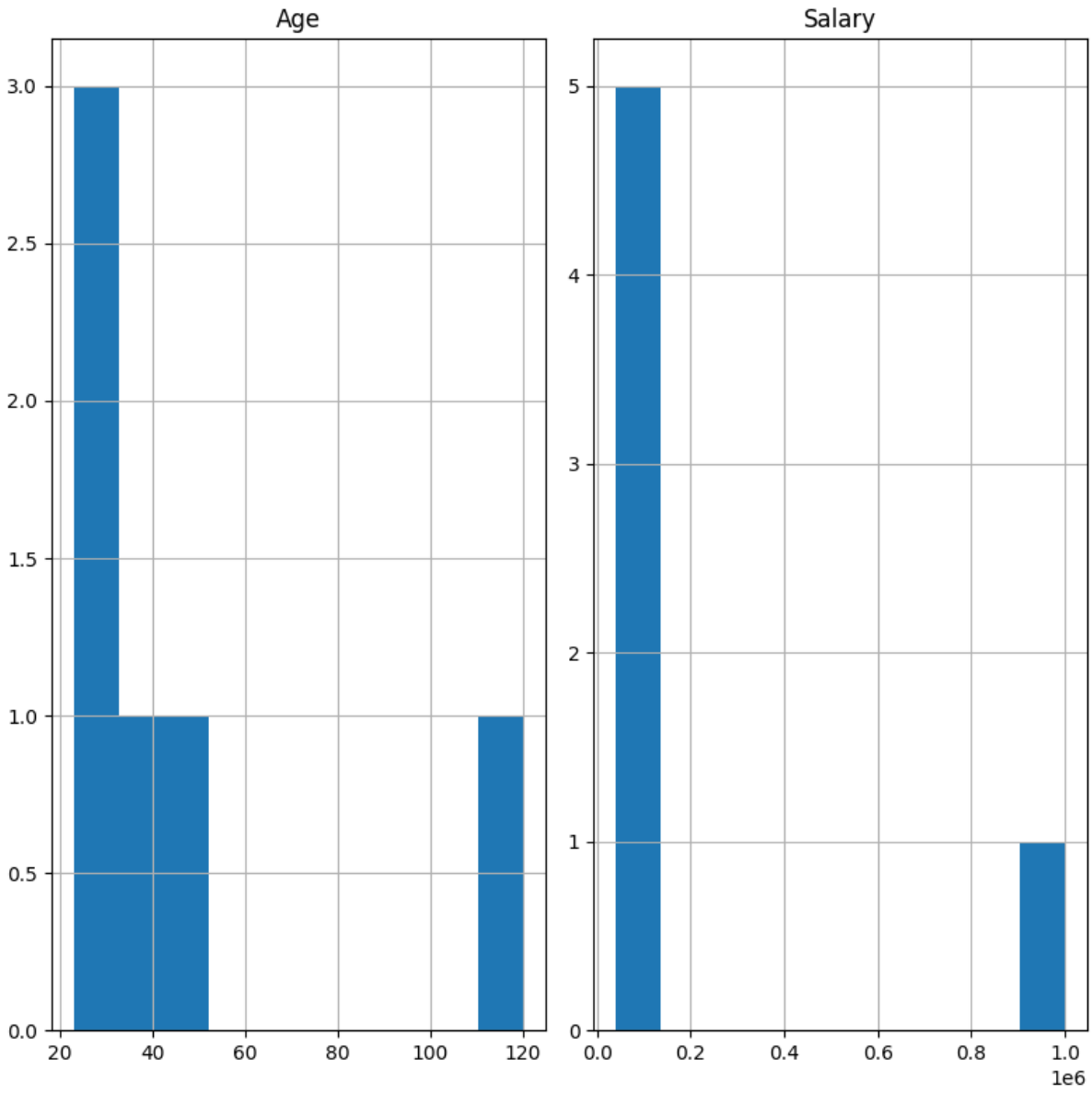
Age -> 6 unique

Salary -> 6 unique

Dept -> 3 unique

CORRELATION:

	Age	Salary
Age	1.000000	0.985815
Salary	0.985815	1.000000



VALUE COUNTS for Dept:

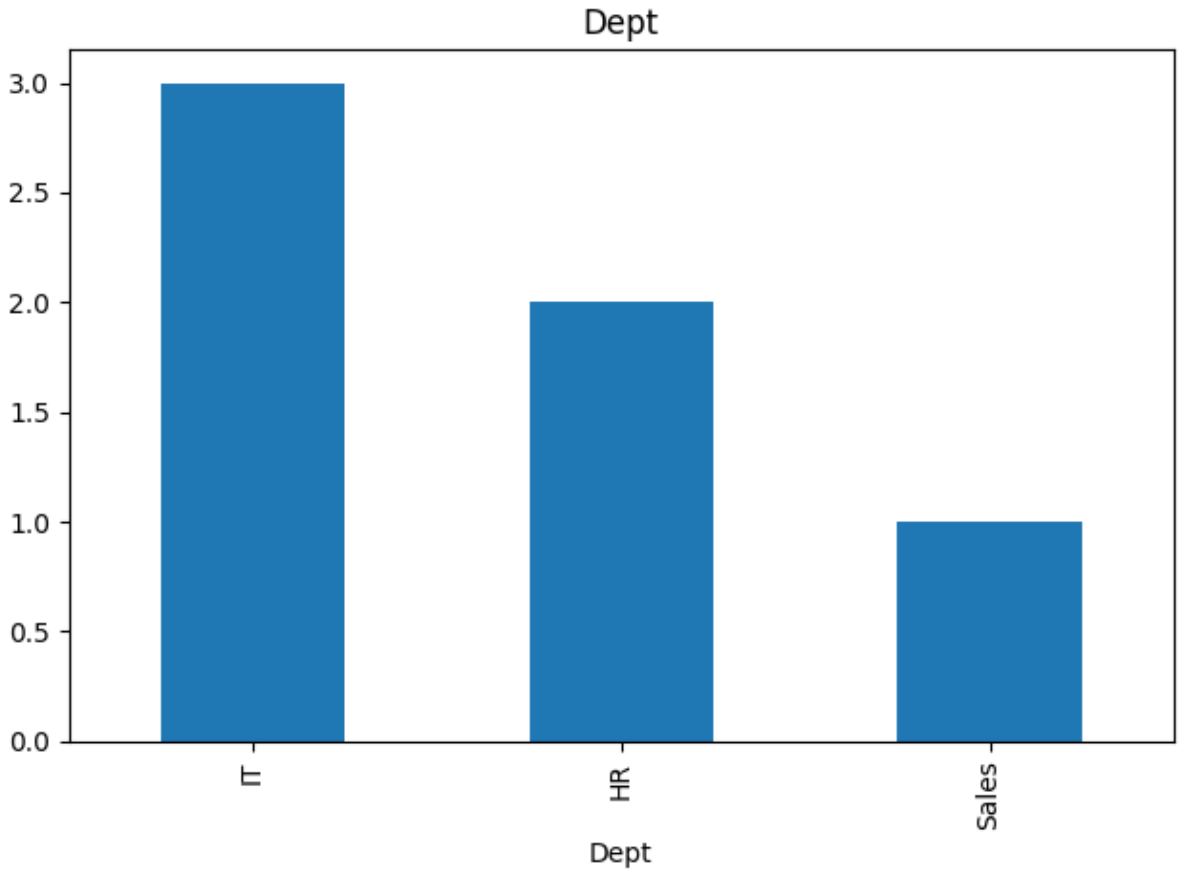
Dept

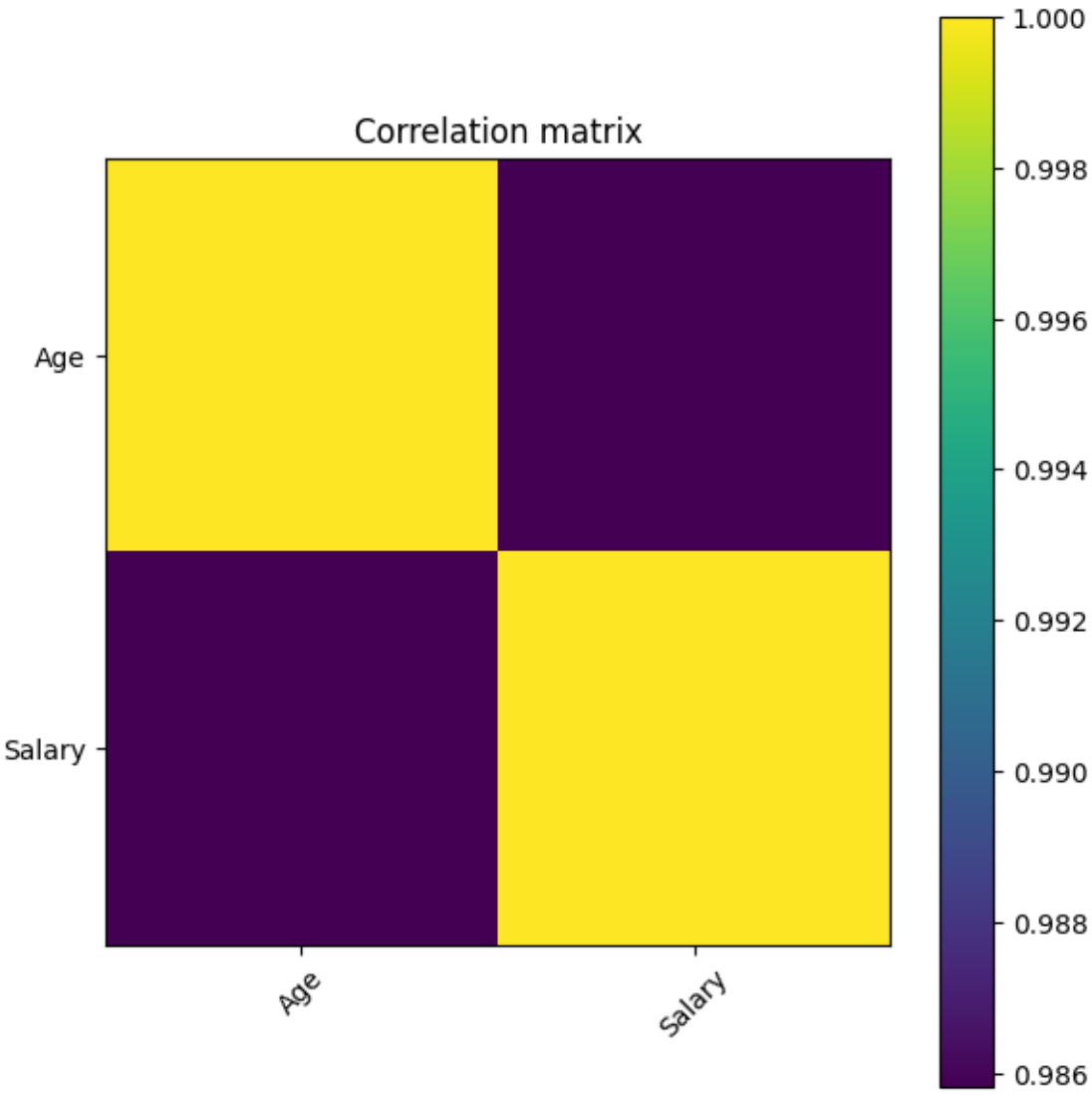
IT 3

HR 2

Sales 1

Name: count, dtype: int64


Dept
3.0
2.5
2.0
1.5
1.0
0.5
0.0
IT
HR
Sales
Dept


Correlation matrix
Age
Salary
Age
Salary
1.000
0.998
0.996
0.994
0.992
0.990
0.988
0.986

localhost:8888/notebooks/madhujads.ipynb

jupyter madhujads Last Checkpoint: 42 minutes ago

File Edit View Run Kernel Settings Help

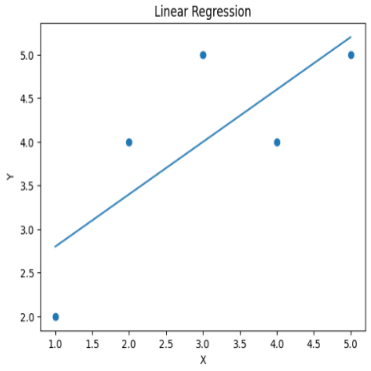
Trusted

Code

JupyterLab Python 3 (ipykernel)

[18]:
```
import numpy as np
import matplotlib.pyplot as plt

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

mx = np.mean(x)
my = np.mean(y)

b1 = np.sum((x - mx)*(y - my)) / np.sum((x - mx)**2)
b0 = my - b1*mx

y_pred = b0 + b1*x

print("Slope:", b1)
print("Intercept:", b0)
print("Predicted:", y_pred)

plt.scatter(x, y)
plt.plot(x, y_pred)
plt.xlabel("X")
plt.ylabel("Y")
plt.title("Linear Regression")
plt.show()
```

```
Slope: 0.6
Intercept: 2.2
Predicted: [2.8 3.4 4.  4.6 5.2]
```



[ ]:

jupyter madhujads Last Checkpoint: 46 minutes ago

File Edit View Run Kernel Settings Help

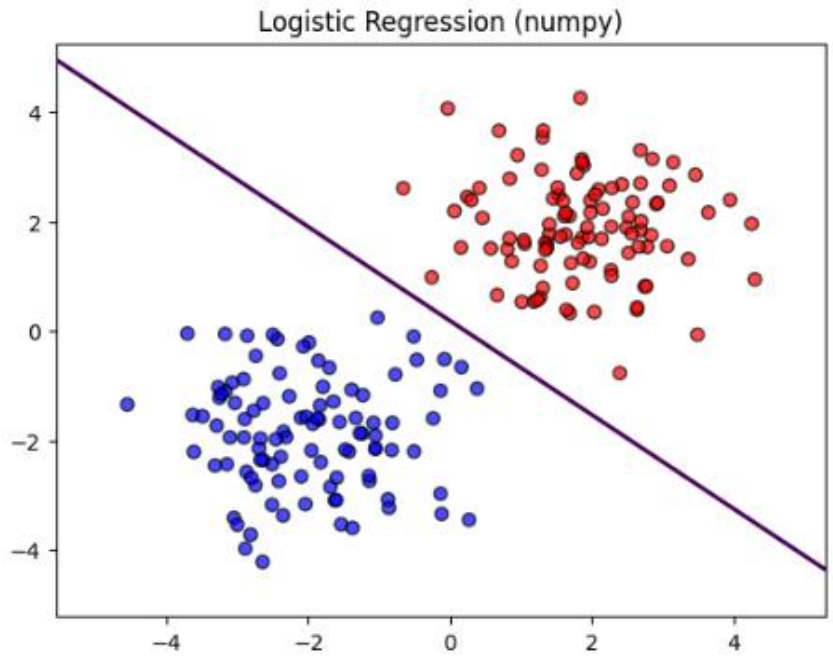
Trusted

Code

JupyterLab Python 3 (ipykernel)

[19]:
```
import numpy as np
import matplotlib.pyplot as plt

def sigmoid(z):
    return 1 / (1 + np.exp(-z))

def add_bias(X):
    return np.c_[np.ones(X.shape[0]), X]

def train_logistic(X, y, lr=0.1, epochs=10000):
    Xb = add_bias(X)
    w = np.zeros(Xb.shape[1])
    for i in range(epochs):
        z = Xb.dot(w)
        preds = sigmoid(z)
        grad = Xb.T.dot(preds - y) / Xb.shape[0]
        w -= lr * grad
    return w

def predict_proba(X, w):
    Xb = add_bias(X)
    return sigmoid(Xb.dot(w))

def predict(X, w, threshold=0.5):
    return (predict_proba(X, w) >= threshold).astype(int)

np.random.seed(0)
N = 100
X0 = np.random.randn(N,2) + np.array([-2, -2])
X1 = np.random.randn(N,2) + np.array([2, 2])
X = np.vstack([X0, X1])
y = np.hstack([np.zeros(N), np.ones(N)])

w = train_logistic(X, y, lr=0.2, epochs=2000)
y_pred = predict(X, w)
acc = (y_pred == y).mean()
print("Accuracy:", acc)

plt.scatter(X[:,0], X[:,1], c=y, cmap='bwr', edgecolor='k', alpha=0.7)
xmin, xmax = X[:,0].min()-1, X[:,0].max()+1
ymin, ymax = X[:,1].min()-1, X[:,1].max()+1
xx, yy = np.meshgrid(np.linspace(xmin,xmax,200), np.linspace(ymin,ymax,200))
grid = np.c_[xx.ravel(), yy.ravel()]
probs = predict_proba(grid, w).reshape(xx.shape)
plt.contour(xx, yy, probs, levels=[0.5], linewidths=2)
plt.title("Logistic Regression (numpy)")
plt.show()
```

```
Accuracy: 1.0
```



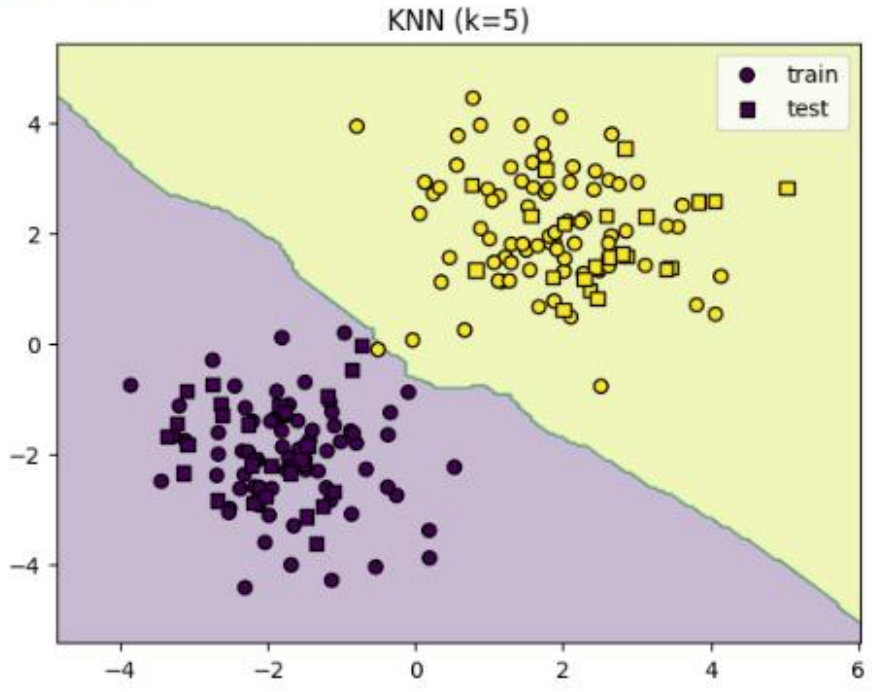
jupyter madhujads Last Checkpoint: 49 minutes ago

File Edit View Run Kernel Settings Help

Code JupyterLab

[21]:
```
import numpy as np
import matplotlib.pyplot as plt

def knn_predict(X_train, y_train, X_test, k=3):
    y_pred = []
    for x in X_test:
        dists = np.sqrt(((X_train - x) ** 2).sum(axis=1))
        idx = np.argsort(dists)[:k]
        vals, counts = np.unique(y_train[idx], return_counts=True)
        y_pred.append(vals[np.argmax(counts)])
    return np.array(y_pred)

np.random.seed(1)
N = 100
X0 = np.random.randn(N,2) + np.array([-2, -2])
X1 = np.random.randn(N,2) + np.array([2, 2])
X = np.vstack([X0, X1])
y = np.hstack([np.zeros(N), np.ones(N)])

perm = np.random.permutation(len(X))
train_idx = perm[:150]
test_idx = perm[150:]
X_train, y_train = X[train_idx], y[train_idx]
X_test, y_test = X[test_idx], y[test_idx]

k = 5
y_pred = knn_predict(X_train, y_train, X_test, k=k)
acc = (y_pred == y_test).mean()
print("Accuracy:", acc)

xmin, xmax = X[:,0].min()-1, X[:,0].max()+1
ymin, ymax = X[:,1].min()-1, X[:,1].max()+1
xx, yy = np.meshgrid(np.linspace(xmin,xmax,200), np.linspace(ymin,ymax,200))
grid = np.c_[xx.ravel(), yy.ravel()]
grid_pred = knn_predict(X_train, y_train, grid, k=k).reshape(xx.shape)

plt.contourf(xx, yy, grid_pred, alpha=0.3)
plt.scatter(X_train[:,0], X_train[:,1], c=y_train, edgecolor='k', label='train')
plt.scatter(X_test[:,0], X_test[:,1], c=y_test, marker='s', edgecolor='k', label='test')
plt.title(f'KNN (k={k})')
plt.legend()
plt.show()
```

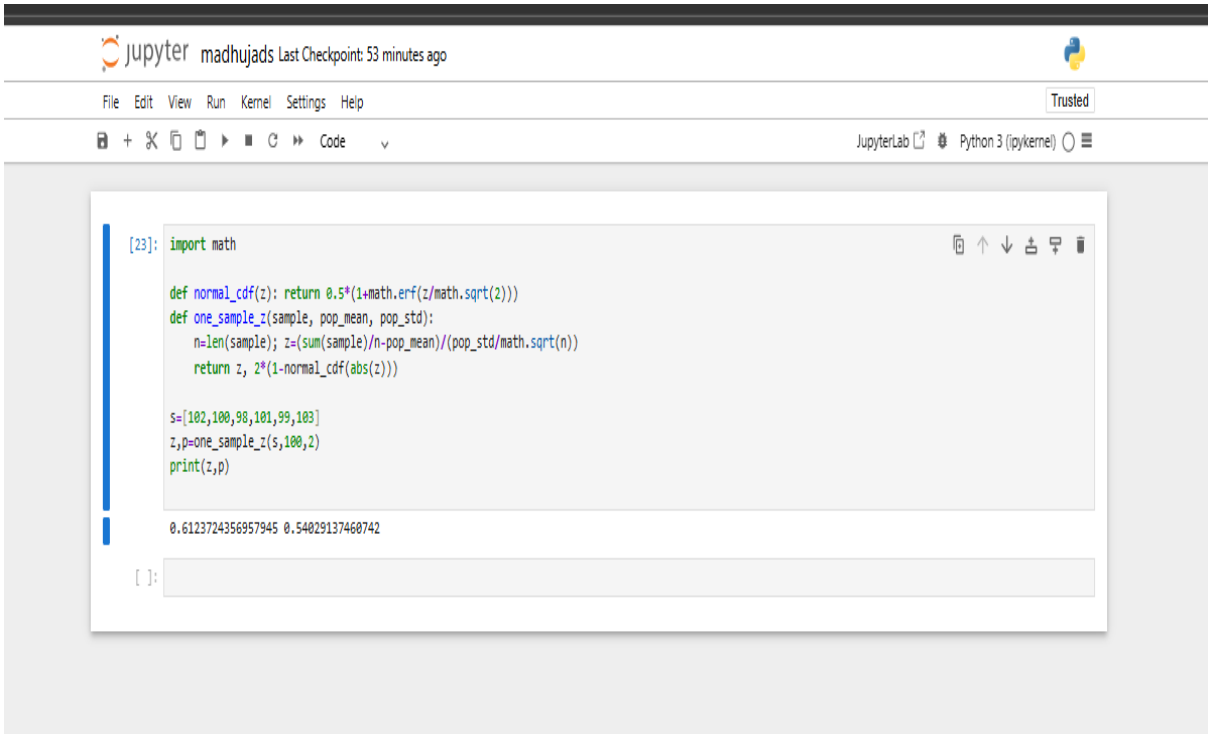
```
Accuracy: 1.0
```



[22]:
```
import math

x1 = [12, 14, 15, 16, 18]
x2 = [10, 11, 13, 14, 15]

m1 = sum(x1)/len(x1)
m2 = sum(x2)/len(x2)

v1 = sum((i - m1)**2 for i in x1) / (len(x1)-1)
v2 = sum((i - m2)**2 for i in x2) / (len(x2)-1)

n1, n2 = len(x1), len(x2)
sp = math.sqrt(((n1-1)*v1 + (n2-1)*v2) / (n1+n2-2))

t = (m1 - m2) / (sp * math.sqrt(1/n1 + 1/n2))

print("t-value:", t)
```

```
t-value: 1.7597653802562394
```

[ ]:

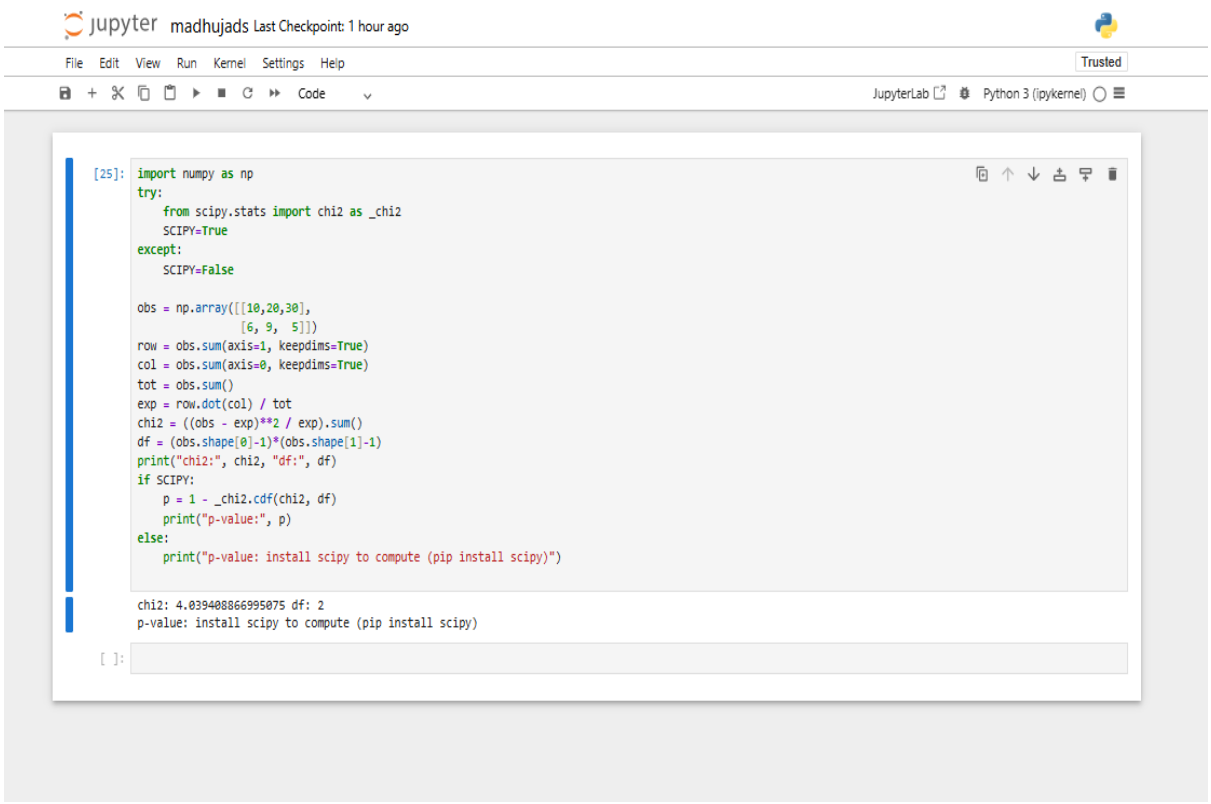

jupyter madhujads Last Checkpoint: 53 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[23]:
import math
def normal_cdf(z): return 0.5*(1+math.erf(z/math.sqrt(2)))
def one_sample_z(sample, pop_mean, pop_std):
n=len(sample); z=(sum(sample)/n-pop_mean)/(pop_std/math.sqrt(n))
return z, 2*(1-normal_cdf(abs(z)))
s=[102,100,98,101,99,103]
z,p=one_sample_z(s,100,2)
print(z,p)
0.6123724356957945 0.54029137460742
[ ]:

Code JupyterLab Python 3 (ipykernel)

[24]:
```
import numpy as np

g1 = [12, 14, 15, 16]
g2 = [18, 17, 16, 19]
g3 = [22, 21, 20, 23]

groups = [g1, g2, g3]
means = [np.mean(g) for g in groups]
overall = np.mean(np.concatenate(groups))

ssb = sum([len(g)*(m-overall)**2 for g,m in zip(groups,means)])
ssw = sum([sum((x-m)**2 for x in g) for g,m in zip(groups,means)])

k = len(groups)
n = sum(len(g) for g in groups)
dfb = k-1
dfw = n-k

msb = ssb/dfb
msw = ssw/dfw

F = msb/msw
print("F-value:", F)
```

```
F-value: 25.319999999999997
```

[ ]:


jupyter madhujads Last Checkpoint: 1 hour ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[25]:
import numpy as np
try:
from scipy.stats import chi2 as _chi2
SCIPY=True
except:
SCIPY=False
obs = np.array([[10,20,30],
[6, 9, 5]])
row = obs.sum(axis=1, keepdims=True)
col = obs.sum(axis=0, keepdims=True)
tot = obs.sum()
exp = row.dot(col) / tot
chi2 = ((obs - exp)**2 / exp).sum()
df = (obs.shape[0]-1)*(obs.shape[1]-1)
print("chi2:", chi2, "df:", df)
if SCIPY:
p = 1 - _chi2.cdf(chi2, df)
print("p-value:", p)
else:
print("p-value: install scipy to compute (pip install scipy)")
chi2: 4.039408866995075 df: 2
p-value: install scipy to compute (pip install scipy)
[ ]: