#1.A) BASIC PRACTICE EXPERIMENTS 1-4

#230701004

#ABHINAYA LAKSHMI.S

#30/07/24

import pandas as pd

import matplotlib.pyplot as plt

data = {‘Year’: list(range(2010, 2021)),

‘Job Postings’: [150, 300, 450, 600, 800, 1200, 1600, 2100, 2700, 3400, 4200]}

df = pd.DataFrame(data)

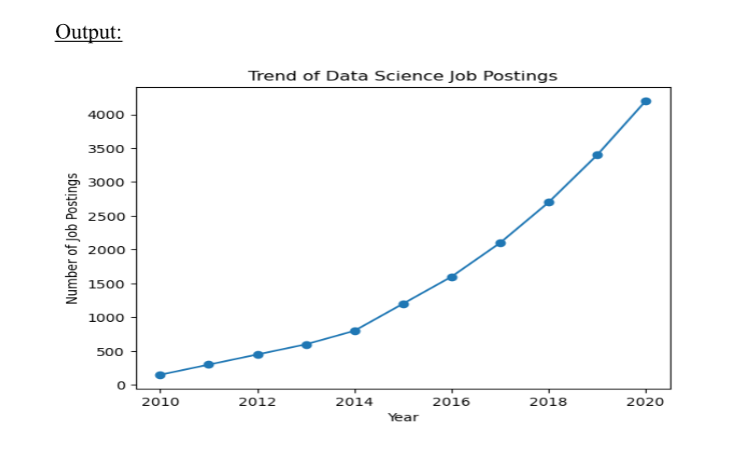
plt.plot(df[‘Year’], df[‘job Postings rob Postings’], marker=‘o’)

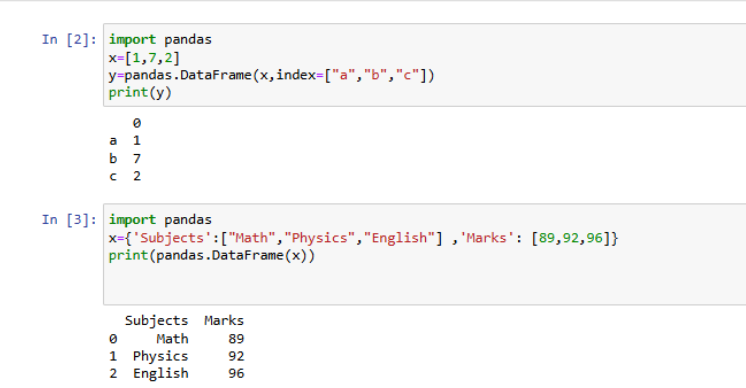
plt.title(‘Trend of Data Science Job Postings’)

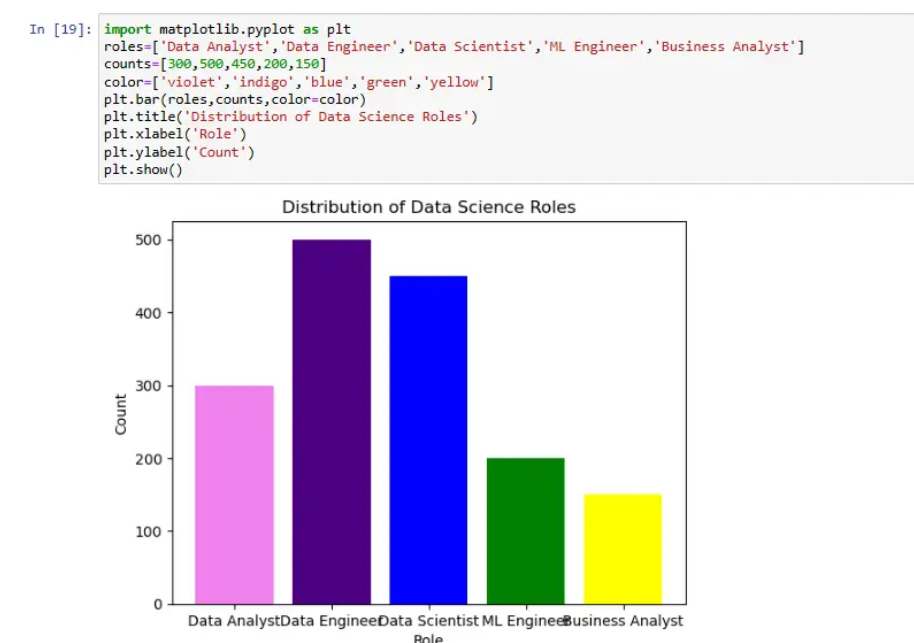
plt.xlabel(‘Year’)

plt.ylabel(‘Number of Job Postings’)

plt.show()







#1.B) PANDAS BUIT IN FUNCTION; NUMPY BUIT IN FUCTION- ARRAY SLICING, RAVEL,RESHAPE,NDIM

#230701004

#ABHINAYA LAKSHMI.S

#06.08.2024

#NUMPY FUNCTIONS

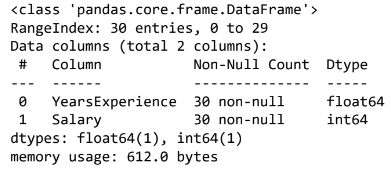
import numpy as np

import pandas as pd

df=pd.read\_csv('Salary\_data.csv')

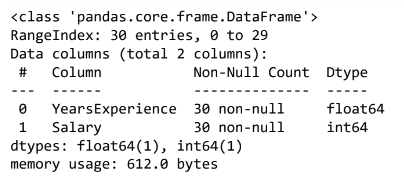
df

df.info()

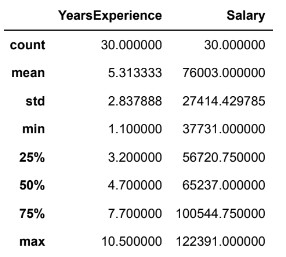


df.dropna(inplace=True)

df.info()



df.describe()



features=df.iloc[:,[0]].values

label=df.iloc[:,[1]].values

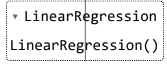
from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(features,label,test\_size=0.2,random\_st

from sklearn.linear\_model import LinearRegression

model=LinearRegression()

model.fit(x\_train,y\_train)



model.score(x\_train,y\_train)



model.score(x\_test,y\_test)



model.coef\_



model.intercept\_



import pickle

pickle.dump(model,open('SalaryPred.model','wb'))

model=pickle.load(open('SalaryPred.model','rb'))

yr\_of\_exp=float(input("Enter Years of Experience: "))

yr\_of\_exp\_NP=np.array([[yr\_of\_exp]])

Salary=model.predict(yr\_of\_exp\_NP)



print("Estimated Salary for {} years of experience is {}: " .format(yr\_of\_exp,Salary)



#PANDAS FUNCTIONS

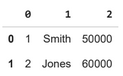
import numpy as np

import pandas as pd

list=[[1,'Smith',50000],[2,'Jones',60000]]

df=pd.DataFrame(list)

df

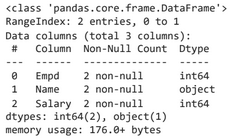


df.columns=['Empd','Name','Salary']

df

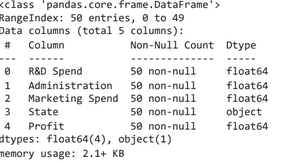


df.info()



df=pd.read\_csv("/content/50\_Startups.csv")

df.info()



df.head()

df.tail()

import numpy as np

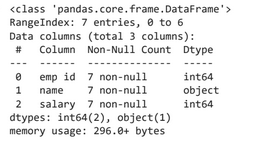
import pandas as pd

df = pd.read\_csv(“/content/employee.csv”)

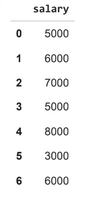
df.head()

df.tail()

df.info()

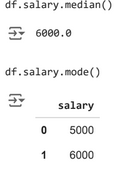


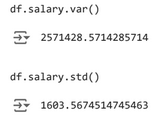
df.salary()



type(df.salary)

df.salary.mean()





empCol=df.columns

empCol



emparray=df.values

employee\_DF=pd.DataFrame(emparray,columns=empCol)

#OUTLIER DETECTION

#230701004

#ABHINAYA LAKSHMI.S

#13.08.2024

#sample calculation for low range(lr) , upper range (ur),percentile

import numpy as np

array=np.random.randint(1,100,16) # randomly generate 16 numbers between 1 to 100

array

#array([21, 72, 69, 45, 61, 43, 43, 59, 62, 42, 90, 25, 54, 86, 80, 13], dtype=int32)

array.mean()

np.percentile(array,25)

np.percentile(array,50)

np.percentile(array,75)

np.percentile(array,100)

#outliers detection

def outDetection(array):

    sorted(array)

    Q1,Q3=np.percentile(array,[25,75])

    IQR=Q3-Q1

    lr=Q1-(1.5\*IQR)

    ur=Q3+(1.5\*IQR)

    return lr,ur

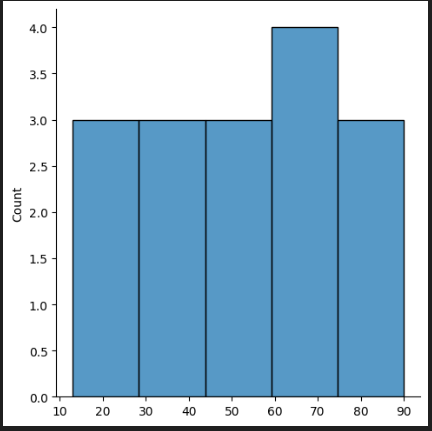
lr,ur=outDetection(array)

lr,ur

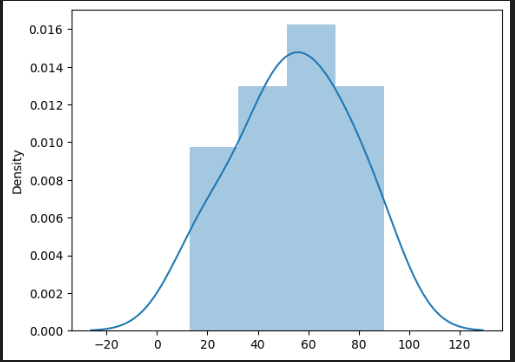
import seaborn as sns

%matplotlib inline

sns.displot(array)



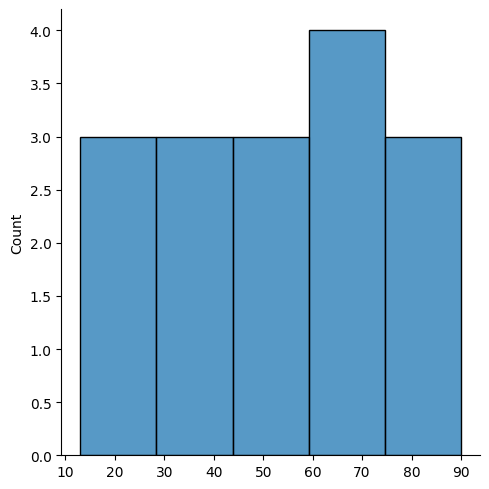
sns.distplot(array)



new\_array=array[(array>lr) & (array<ur)]

new\_array

sns.displot(new\_array)



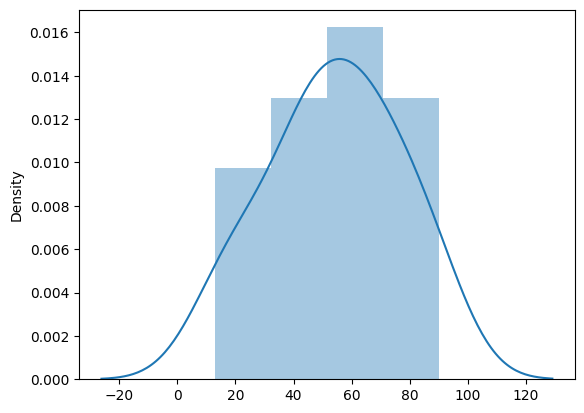
lr1,ur1=outDetection(new\_array)

lr1,ur1

final\_array=new\_array[(new\_array>lr1) & (new\_array<ur1)]

final\_array

sns.distplot(final\_array)



#3) Missing and inappropriate data

#230701004

#ABHINAYA LAKSHMI.S

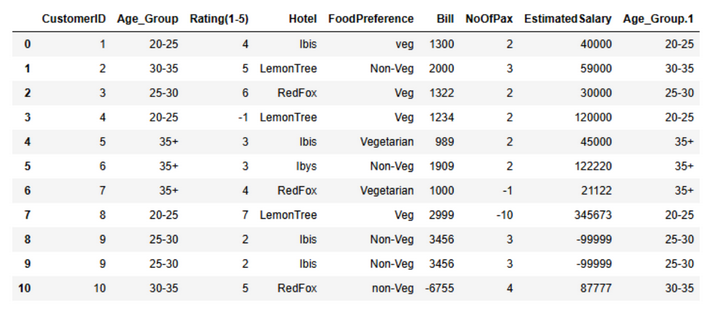
#20.08.2024

import numpy as np

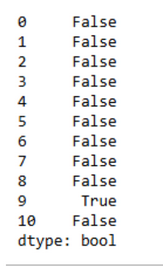
import pandas as pd

df=pd.read\_csv("Hotel\_Dataset.csv")

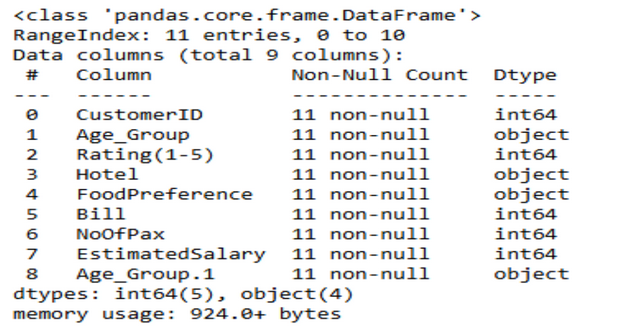
df



df.duplicated()

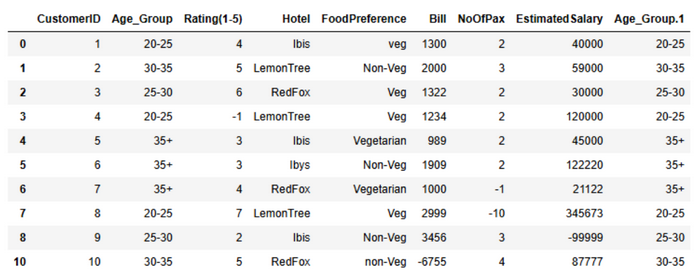


df.info()



df.drop\_duplicates(inplace=True)

df



len(df)

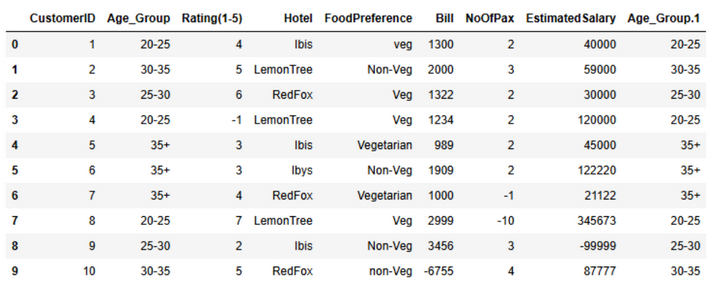
index=np.array(list(range(0,len(df))))

df.set\_index(index,inplace=True)

index

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

df



df.drop(['Age\_Group.1'],axis=1,inplace=True)

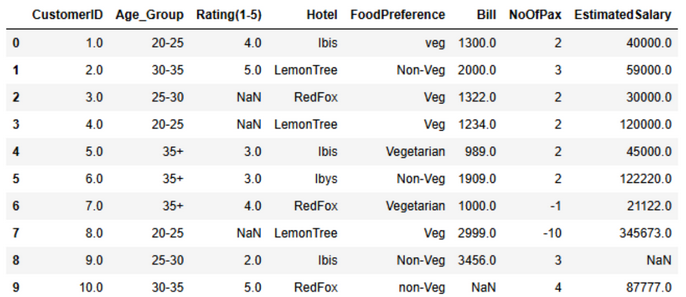
df

df.CustomerID.loc[df.CustomerID<0]=np.nan

df.Bill.loc[df.Bill<0]=np.nan

df.EstimatedSalary.loc[df.EstimatedSalary<0]=np.nan

df



df['NoOfPax'].loc[(df['NoOfPax']<1) | (df['NoOfPax']>20)]=np.nan

df

df.Age\_Group.unique()

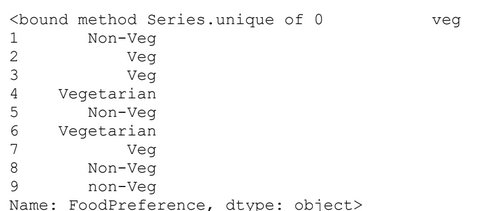


df.Hotel.unique()



df.Hotel.replace(['Ibys'],'Ibis',inplace=True)

df.FoodPreference.unique



df.FoodPreference.replace(['Vegetarian','veg'],'Veg',inplace=True)

df.FoodPreference.replace(['non-Veg'],'Non-Veg',inplace=True)

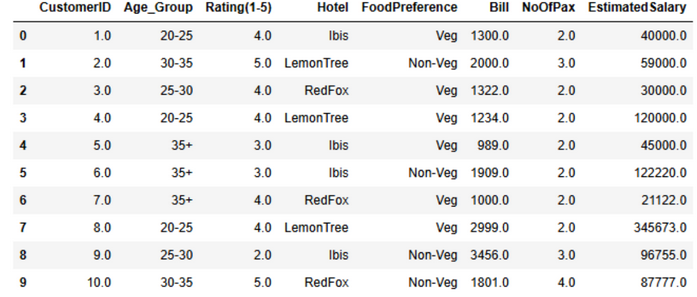
df.EstimatedSalary.fillna(round(df.EstimatedSalary.mean()),inplace=True)

df.NoOfPax.fillna(round(df.NoOfPax.median()),inplace=True)

df['Rating(1-5)'].fillna(round(df['Rating(1-5)'].median()), inplace=True)

df.Bill.fillna(round(df.Bill.mean()),inplace=True)

df



#4)Data Preprocessing

#230701004

#ABHINAYA LAKSHMI.S

#27.08.2024

import numpy as np

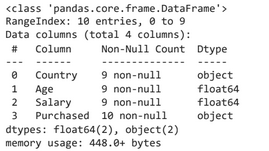
import pandas as pd

df=pd.read\_csv("/content/pre-process\_datasample.csv")

df



df.info()



df.Country.mode()



df.Country.mode()[0]

type(df.Country.mode())

df.Country.fillna(df.Country.mode()[0],inplace=True)

df.Age.fillna(df.Age.median(),inplace=True)

df.Salary.fillna(round(df.Salary.mean()),inplace=True)

df



pd.get\_dummies(df.Country)

updated\_dataset=pd.concat([pd.get\_dummies(df.Country),df.iloc[:,[1,2,3]]],axis=1)

France Germany Spain Age Salary Purchased

0 True False False 44.0 72000.0 No

1 False False True 27.0 48000.0 Yes

2 False True False 30.0 54000.0 No

3 False False True 38.0 61000.0 No

4 False True False 40.0 63778.0 Yes

5 True False False 35.0 58000.0 Yes

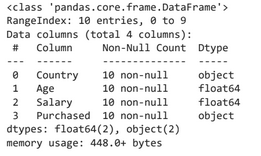
6 False False True 38.0 52000.0 No

7 True False False 48.0 79000.0 Yes

8 True False False 50.0 83000.0 No

9 True False False 37 0 67000 0 Yes

df.info()



updated\_dataset.Purchased.replace(['No','Yes'],[0,1],inplace=True)

updated\_dataset

France Germany Spain Age Salary Purchased

0 True False False 44.0 72000.0 0

1 False False True 27.0 48000.0 1

2 False True False 30.0 54000.0 0

3 False False True 38.0 61000.0 0

4 False True False 40.0 63778.0 1

5 True False False 35.0 58000.0 1

6 False False True 38.0 52000.0 0

7 True False False 48.0 79000.0 1

8 True False False 50.0 83000.0 0

9 True False False 37.0 67000.0 1

# EDA-Quantitative and Qualitative plots - Experiments 1

#230701004

#ABHINAYA LAKSHMI.S

# 03.09.2024

import seaborn as sns

import pandas as pd

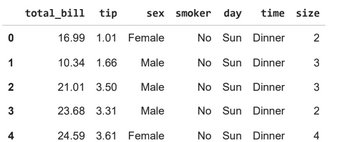
import numpy as np

import matplotlib.pyplot as plt

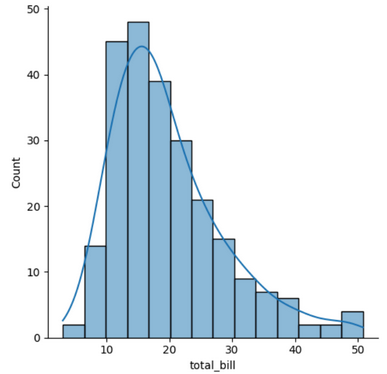
%matplotlib inline

tips=sns.load\_dataset('tips')

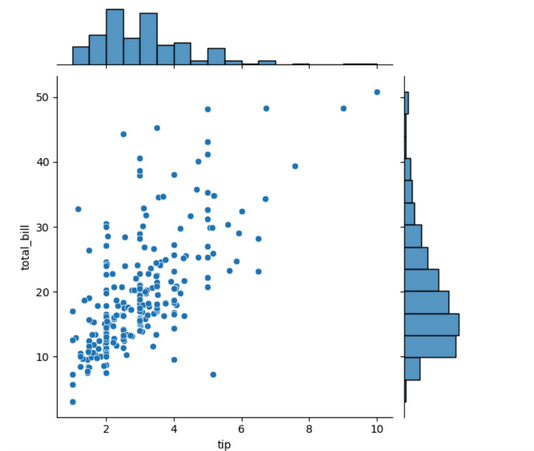
tips.head()



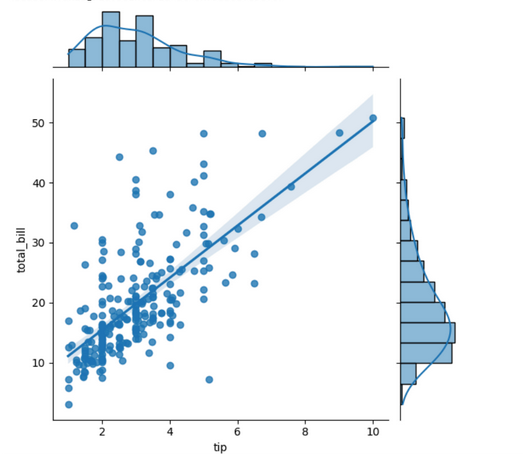
sns.displot(tips.total\_bill,kde=True)



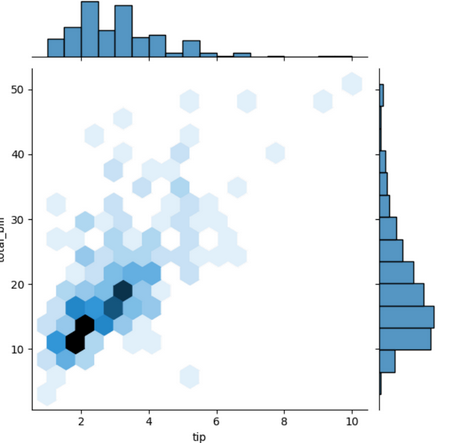
sns.jointplot(x=tips.tip,y=tips.total\_bill)



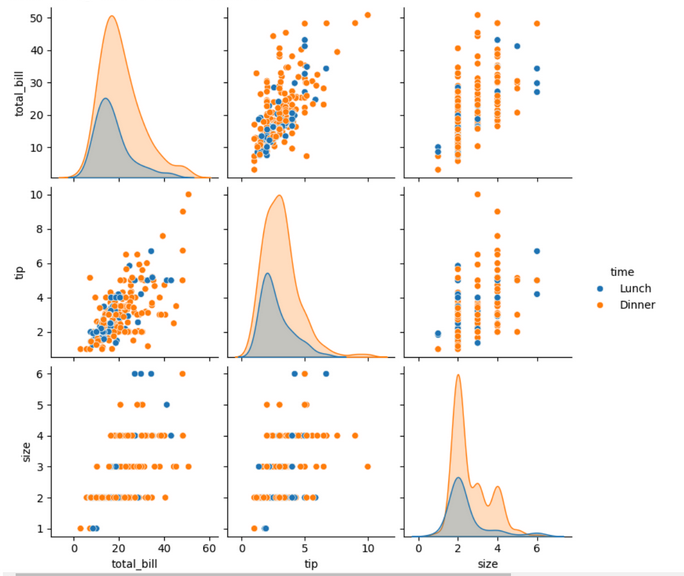
sns.jointplot(x=tips.tip,y=tips.total\_bill,kind="reg")



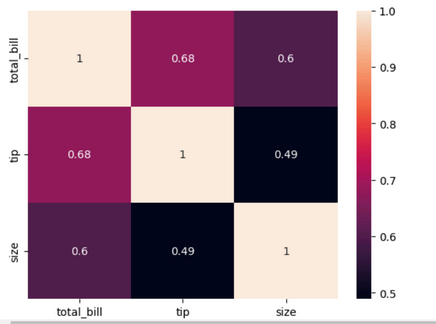
sns.jointplot(x=tips.tip,y=tips.total\_bill,kind="hex")



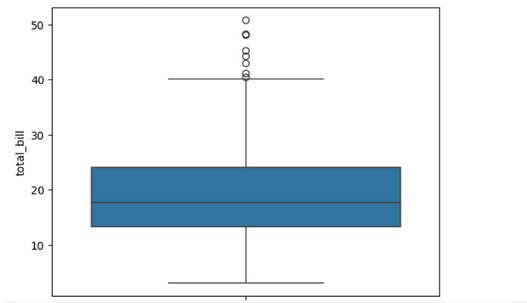
sns.pairplot(tips,hue=’time’)



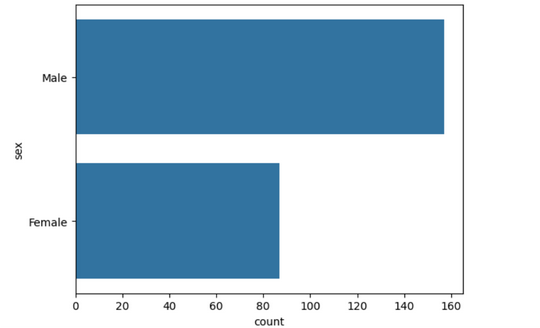
sns.heatmap(tips.corr(numeric\_only=True),annot=True)



sns.boxplot(tips.total\_bill)

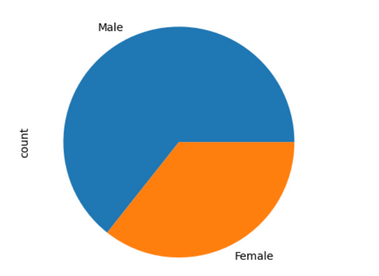


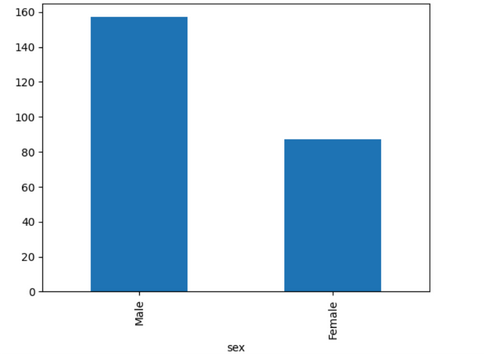
sns.countplot(tips.sex)



tips.sex.value\_counts().plot(kind='pie')

tips.sex.value\_counts().plot(kind='bar')





#Random Sampling and Sampling Distribution

#230701004

#ABHINAYA LAKSHMI.S

# 10.09.2024

import numpy as np

import matplotlib.pyplot as plt

population\_mean = 50

population\_std = 10

population\_size = 100000

population = np.random.normal(population\_mean, population\_std, population\_size)

plt.figure(figsize=(8, 5))

plt.hist(population, bins=50, color='skyblue', edgecolor='black', alpha=0.7)

plt.title('Population Distribution')

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.axvline(population\_mean, color='red', linestyle='dashed', linewidth=1.5, label='Population Mean')

plt.legend()

plt.show()

sample\_sizes = [30, 50, 100]

num\_samples = 1000

sample\_means = {}

for size in sample\_sizes:

sample\_means[size] = []

for \_ in range(num\_samples):

sample = np.random.choice(population, size=size, replace=False)

sample\_means[size].append(np.mean(sample))

plt.figure(figsize=(12, 8))

for i, size in enumerate(sample\_sizes):

plt.subplot(len(sample\_sizes), 1, i + 1)

plt.hist(sample\_means[size], bins=30, alpha=0.7, color='orange', edgecolor='black',

label=f'Sample Size {size}')

plt.axvline(np.mean(population), color='red', linestyle='dashed', linewidth=1.5, label='Population Mean')

plt.title(f'Sampling Distribution of the Sample Mean (Sample Size {size})')

plt.xlabel('Sample Mean')

plt.ylabel('Frequency')

plt.legend()

plt.tight\_layout()

plt.show()

plt.figure(figsize=(12, 8))

for i, size in enumerate(sample\_sizes):

plt.subplot(len(sample\_sizes), 1, i + 1)

plt.hist(sample\_means[size], bins=30, alpha=0.7, color='purple', edgecolor='black',

label=f'Sample Size {size}', density=True)

plt.axvline(np.mean(population), color='red', linestyle='dashed', linewidth=1.5, label='Population Mean')

plt.title(f'Sampling Distribution (Sample Size {size}) - CLT Demonstration')

plt.xlabel('Sample Mean')

plt.ylabel('Density')

plt.legend()

plt.tight\_layout()

plt.show()

# Z-Test  
# 230701004  
# ABHINAYA LAKSHMI.S  
# 10.09.2024

import numpy as np  
import scipy.stats as stats  
sample\_data = np.array([  
152, 148, 151, 149, 147, 153, 150, 148, 152, 149,  
151, 150, 149, 152, 151, 148, 150, 152, 149, 150,  
148, 153, 151, 150, 149, 152, 148, 151, 150, 153  
])  
population\_mean = 150  
sample\_mean = np.mean(sample\_data)  
sample\_std = np.std(sample\_data, ddof=1)  
n = len(sample\_data)  
z\_statistic = (sample\_mean - population\_mean) / (sample\_std / np.sqrt(n))  
p\_value = 2 \* (1 - stats.norm.cdf(np.abs(z\_statistic)))  
print(f"Sample Mean: {sample\_mean:.2f}")  
print(f"Z-Statistic: {z\_statistic:.4f}")  
print(f"P-Value: {p\_value:.4f}")  
alpha = 0.05  
if p\_value < alpha:  
print("Reject the null hypothesis: The average weight is significantly different from 150 grams.")  
else:  
print("Fail to reject the null hypothesis: There is no significant difference in average weight from 150  
grams.")

# T-Test

# 230701004

# ABHINAYA LAKSHMI.S

# 08.10.2024

import numpy as np

import scipy.stats as stats

np.random.seed(42)

sample\_size = 25

sample\_data = np.random.normal(loc=102, scale=15, size=sample\_size)

population\_mean = 100

sample\_mean = np.mean(sample\_data)

sample\_std = np.std(sample\_data, ddof=1)

n = len(sample\_data)

t\_statistic, p\_value = stats.ttest\_1samp(sample\_data, population\_mean)

print(f"Sample Mean: {sample\_mean:.2f}")

print(f"T-Statistic: {t\_statistic:.4f}")

print(f"P-Value: {p\_value:.4f}")

alpha = 0.05

if p\_value < alpha:

print("Reject the null hypothesis: The average IQ score is significantly different from 100.")

else:

print("Fail to reject the null hypothesis: There is no significant difference in average IQ score from 100."

# Anova TEST  
# 230701004  
# ABHINAYA LAKSHMI.S  
# 08.10.2024

import numpy as np  
import scipy.stats as stats  
np.random.seed(42)  
n\_plants = 25  
growth\_A = np.random.normal(loc=10, scale=2, size=n\_plants)  
growth\_B = np.random.normal(loc=12, scale=3, size=n\_plants)  
growth\_C = np.random.normal(loc=15, scale=2.5, size=n\_plants)  
f\_statistic, p\_value = stats.f\_oneway(growth\_A, growth\_B, growth\_C)  
print("Treatment A Mean Growth:", np.mean(growth\_A))  
print("Treatment B Mean Growth:", np.mean(growth\_B))  
print("Treatment C Mean Growth:", np.mean(growth\_C))  
print()  
print(f"F-Statistic: {f\_statistic:.4f}")  
print(f"P-Value: {p\_value:.4f}")  
alpha = 0.05  
if p\_value < alpha:  
print("Reject the null hypothesis: There is a significant difference in mean growth rates among the three  
treatments.")  
else:  
print("Fail to reject the null hypothesis: There is no significant difference in mean growth rates among the  
three treatments.")  
if p\_value < alpha:  
all\_data = np.concatenate([growth\_A, growth\_B, growth\_C])  
treatment\_labels = ['A'] \* n\_plants + ['B'] \* n\_plants + ['C'] \* n\_plants  
tukey\_results = pairwise\_tukeyhsd(all\_data, treatment\_labels, alpha=0.05)  
print("\nTukey's HSD Post-hoc Test:")  
print(tukey\_results)

# Feature Scaling

# 230701004

# ABHINAYA LAKSHMI.S

# 22.10.2024

import numpy as np

import pandas as pd

df = pd.read\_csv('/content/pre-process\_datasample.csv')

print("Original Data:")

print(df)

df['Country'].fillna(df['Country'].mode()[0], inplace=True)

features = df.iloc[:, :-1].values

label = df.iloc[:, -1].values

from sklearn.impute import SimpleImputer

age\_imputer = SimpleImputer(strategy="mean")

salary\_imputer = SimpleImputer(strategy="mean")

age\_imputer.fit(features[:, [1]])

salary\_imputer.fit(features[:, [2]])

features[:, [1]] = age\_imputer.transform(features[:, [1]])

features[:, [2]] = salary\_imputer.transform(features[:, [2]])

print("Features after handling missing values:")

print(features)

from sklearn.preprocessing import OneHotEncoder

oh = OneHotEncoder(sparse\_output=False)

Country = oh.fit\_transform(features[:, [0]])

print("OneHotEncoded 'Country' column:")

print(Country)

final\_set = np.concatenate((Country, features[:, [1, 2]]), axis=1)

print("Final dataset with OneHotEncoded 'Country' and other features:")

print(final\_set)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

sc.fit(final\_set)

feat\_standard\_scaler = sc.transform(final\_set)

print("Standardized features:")

print(feat\_standard\_scaler)

from sklearn.preprocessing import MinMaxScaler

mms = MinMaxScaler(feature\_range=(0, 1))

mms.fit(final\_set)

feat\_minmax\_scaler = mms.transform(final\_set)

print("Normalized features:")

print(feat\_minmax\_scaler)

# Linear Regression

# 230701004

# ABHINAYA LAKSHMI.S

# 29.10.2024

import numpy as np

import pandas as pd

df=pd.read\_csv('Salary\_data.csv')

df

df.info()

df.dropna(inplace=True)

df.info()

df.describe()

features=df.iloc[:,[0]].values

label=df.iloc[:,[1]].values

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(features,label,test\_size=0.2,random\_state=23)

from sklearn.linear\_model import LinearRegression

model=LinearRegression()

model.fit(x\_train,y\_train)

model.score(x\_train,y\_train)

model.score(x\_test,y\_test)

model.coef\_

model.intercept\_

import pickle

pickle.dump(model,open('SalaryPred.model','wb'))

model=pickle.load(open('SalaryPred.model','rb'))

yr\_of\_exp=float(input("Enter Years of Experience: "))

yr\_of\_exp\_NP=np.array([[yr\_of\_exp]])

Salary=model.predict(yr\_of\_exp\_NP)

print("Estimated Salary for {} years of experience is {}: " .format(yr\_of\_exp,Salary)

# Logistic Regression

# 230701004

#ABHINAYA LAKSHMI.S

# 05.11.2024

import numpy as np

import pandas as pd

df=pd.read\_csv('Social\_Network\_Ads.csv')

df

df.head()

features=df.iloc[:,[2,3]].values

label=df.iloc[:,4].values

features

label

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

from sklearn.linear\_model import LogisticRegression

for i in range(1,401):

x\_train,x\_test,y\_train,y\_test=train\_test\_split(features,label,test\_size=0.2,random\_state=i)

model=LogisticRegression()

model.fit(x\_train,y\_train)

train\_score=model.score(x\_train,y\_train)

test\_score=model.score(x\_test,y\_test)

if test\_score>train\_score:

print("Test {} Train{} Random State {}".format(test\_score,train\_score,i)

x\_train,x\_test,y\_train,y\_test=train\_test\_split(features,label,test\_size=0.2,random\_state=314)

finalModel=LogisticRegression()

finalModel.fit(x\_train,y\_train)

print(finalModel.score(x\_train,y\_train))

print(finalModel.score(x\_test,y\_test))

from sklearn.metrics import classification\_report

print(classification\_report(label,finalModel.predict(features)))