# dsbda 1:data wrangling

df=pd.read\_csv(‘\_ \_ \_’)

df.head()

df.tail()

df.shape

df.info()

df.dtypes

df.isnull()

df.isnull().sum()

df.replace(np.nan, df[‘bore’.mean], inplace=True)

df.iloc [ 5 ]

df.iloc [ :5,:3 ]

df[‘stroke’]=df[‘stroke’].astype(‘int’)

from sklearn.preprocessing import LabelEncoder

data=df[‘body-style’]

label\_encoder=LabelEncoder()

encoded\_data=label\_encoder.fit\_transform(data)

df[‘HB’]=df[‘body-style’].str[0]

label\_encoder=LabelEncoder()

df[HB encoded]=label\_encoder.fit\_transform(data)

# dsbda 2

df=pd.read\_csv(‘\_ \_ \_’)

df.head()

df.info()

df.describe()

df.dtypes

df.shape

df.isnull()

df.isnull().sum()

df.replace(np.nan, df[‘\_ \_ \_’.mean], inplace=True)

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

sns.boxplot(y=df[‘\_ \_ \_’]

Q1 = df[‘math score’].quantile(0.25)

Q3 = df[‘math score’].quantile(0.75)

IQR= Q3 - Q1

df=df[ (df[‘math score’] >= Q1-1.5 x IQR) & (df[‘math score’] <= Q3+1.5 x IQR)]

sns.boxplot(y=df[‘\_ \_ \_’]

df[‘math score\_log’]=np.log(df[‘math score’])

print(df[‘math score\_log’].head())

# dsbda 3

df=pd.read\_csv(‘\_ \_ \_’)

df.info()

df.dropna(inplace=True)

df.drop(columns=['Purchased'],inplace=True)

def categorize\_age(age):

if age<30:

return 'Young'

elif 30<=age<60:

return 'Middle-aged'

else:

return 'Old'

df['Age\_Group']=df['Age'].apply(categorize\_age)

summary\_stats=df.groupby('Age\_Group')['EstimatedSalary'].describe()

def calculate\_mean(group):

return group['EstimatedSalary'].mean()

df.groupby('Age\_Group').apply(calculate\_mean)

def calculate\_median(group):

return group['EstimatedSalary'].median()

df.groupby('Age\_Group').apply(calculate\_median)

def calculate\_mode(group):

return group['EstimatedSalary'].mode()

df.groupby('Age\_Group').apply(calculate\_mode)

def calculate\_min(group):

return group['EstimatedSalary'].min()

df.groupby('Age\_Group').apply(calculate\_min)

#2ndpart

df=pd.read\_csv('/content/Iris.csv')

setosa=df[df['Species']=='Iris-setosa']

versicolor=df[df['Species']=='Iris-versicolor']

virginica=df[df['Species']=='Iris-virginica']

def describe\_species(df, species\_name):

species\_data = df[df['Species'] == species\_name]

print(f"{species\_name} statistics:\*\*")

print(species\_data.describe())

describe\_species(df, 'Iris-setosa')

describe\_species(df, 'Iris-versicolor')

# dsbda 4

data = pd.read\_csv('/content/boston.csv')

data

head tail info dtypes isnull.sum

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

X= data.iloc[:,0:13]

y= data.iloc[:,-1]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.20, random\_state=4)

print(X\_train.shape)

print(X\_test.shape)

print(y\_train.shape)

print(y\_test.shape)

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import make\_pipeline

model = make\_pipeline(StandardScaler(), LinearRegression())

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

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

y\_pre=model.predict(X\_test)

num\_samples = len(y\_test)

for i in range(num\_samples):

pred\_price=y\_pre[i].item()

actual\_price=y\_test.iloc[i].item()

print(f"Sample {i+1}: Predicted Price: ${pred\_price:.2f}, Actual Price: ${actual\_price}")

# dsbda 5

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score

data=pd.read\_csv('/content/Social\_Network\_Ads.csv')

features=['Age','EstimatedSalary']

target='Purchased'

X\_train,X\_test,y\_train,y\_test=train\_test\_split

(data[features],data[target],test\_size=0.2,random\_state=42)

model = LogisticRegression()

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

y\_predicted = model.predict(X\_test)

confusion\_matrix\_result = confusion\_matrix(y\_test, y\_predicted)

print("Confusion Matrix:\n", confusion\_matrix\_result)

accuracy = accuracy\_score(y\_test, y\_predicted)

error\_rate = 1 - accuracy

precision = precision\_score(y\_test, y\_predicted)

recall = recall\_score(y\_test, y\_predicted)

print("\nAccuracy:", accuracy)

print("Error Rate:", error\_rate)

print("Precision:", precision)

print("Recall:", recall)

# dsbda 6

df = pd.read\_csv('/content/Iris.csv')

df

shape describe isnull isnull.sum

x = df.drop(["Species"],axis=1)

y = df["Species"]

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

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.2, random\_state=0)

print(x\_train.shape)

print(y\_train.shape)

print(x\_test.shape)

print(y\_test.shape)

from sklearn.naive\_bayes import MultinomialNB

classifier = MultinomialNB()

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

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

y\_pred = classifier.predict(x\_test)

y\_pred

y\_test

import sklearn.metrics

lbs = ['Iris-versicolor','Iris-setosa','Iris-virginica']

print(sklearn.metrics.confusion\_matrix(y\_test, y\_pred, labels = lbs))

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

error\_rate = 1 - accuracy

print("Error rate:", error\_rate)

precision = precision\_score(y\_test, y\_pred, average="macro")

print("Precision:", precision)

recall = recall\_score(y\_test, y\_pred, average="macro")

print("Recall:", recall)

# dsbda 7

import nltk

nltk.download('punkt')

nltk.download('wordnet')

nltk.download('averaged\_perceptron\_tagger')

nltk.download('stopwords')

from nltk.tokenize import word\_tokenize

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer, WordNetLemmatizer

from nltk import pos\_tag

document = "This is an example document that we will use to demonstrate document preprocessing."

tokens = word\_tokenize(document)

pos\_tags = pos\_tag(tokens)

stop\_words = set(stopwords.words('english'))

filtered\_tokens = [word for word in tokens if not word.lower() in stop\_words]

ps = PorterStemmer()

stemmed\_tokens = [ps.stem(word) for word in filtered\_tokens]

wnl = WordNetLemmatizer()

lemmatized\_tokens = [wnl.lemmatize(word) for word in filtered\_tokens]

#2ndpart

import math

from collections import Counter

corpus = [

'The quick brown fox jumps over the lazy dog',

'The brown fox is quick',

'The lazy dog is sleeping'

]

tokenized\_docs = [doc.lower().split() for doc in corpus]

tf\_docs = [Counter(tokens) for tokens in tokenized\_docs]

n\_docs = len(corpus)

idf = {}

for tokens in tokenized\_docs:

for token in set(tokens):

idf[token] = idf.get(token, 0) + 1

for token in idf:

idf[token] = math.log(n\_docs / idf[token])

tfidf\_docs = [ ]

for tf\_doc in tf\_docs:

tfidf\_doc = {}

for token, freq in tf\_doc.items():

tfidf\_doc[token] = freq \* idf[token]

tfidf\_docs.append(tfidf\_doc)

for i, tfidf\_doc in enumerate(tfidf\_docs):

print(f"Document {i+1}: {tfidf\_doc}")

# dsbda 8

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

dataset = sns.load\_dataset('titanic')

sns.displot(x = df['age'], bins = 10,kde=True)

sns.displot(df['age'], bins = 10,kde=False)

sns.jointplot(x = df['age'], y = df['fare'], kind ='scatter')

sns.jointplot(x = df['age'], y = df['fare'], kind = 'hex')

sns.rugplot(df['fare'])

sns.barplot(x='sex', y='age', data=df)

sns.barplot(x='sex', y='age', data=df, estimator=np.std)

sns.countplot(x='sex', data=df)

sns.boxplot(x='sex', y='age', data=df)

sns.boxplot(x='sex', y='age', data=df, hue="survived")

sns.violinplot(x='sex', y='age', data=df)

sns.violinplot(x='sex', y='age', data=df, hue='survived')

sns.stripplot(x='sex', y='age', data=df, jitter=False)

sns.stripplot(x='sex', y='age', data=df, jitter=True)

sns.stripplot(x='sex', y='age', data=df, jitter=True, hue='survived')

sns.swarmplot(x='sex', y='age', data=df)

sns.swarmplot(x='sex', y='age', data=df, hue='survived')

sns.pairplot(df, hue="pclass")

required\_columns = ['survived', 'pclass', 'age','sibsp', 'parch','fare']

data = df[required\_columns].copy()

sns.heatmap(data.corr(), cmap='Wistia')

sns.heatmap(data.corr(), annot=True,cmap='Wistia')

#2ndpart

plt.figure(figsize=(10, 6))

sns.histplot(dataset['fare'], bins=30 )

plt.title('Distribution of Ticket Prices on Titanic')

plt.xlabel('Fare')

plt.ylabel('Frequency')

plt.show()

# dsbda 9

df = sns.load\_dataset('titanic')

df.head()

sns.boxplot(data=df, x="age", y="sex",hue='survived')

sns.boxplot(data=df, x="sex", y="age",hue='survived')

# dsbda 10

from sklearn.datasets import load\_iris

import pandas as pd

iris = load\_iris()

iris\_df = pd.DataFrame(iris.data, columns=iris.feature\_names)

iris\_df['target'] = iris.target

#1

iris\_df.dtypes

#2

import matplotlib.pyplot as plt

iris\_df.hist()

plt.show()

#3

iris\_df.boxplot()

plt.show()

#4

import seaborn as sns

sns.boxplot(data=iris\_df, orient="h")

plt.show()