Sanjana Niranjan

1BM22CS418

Sl no	Date	Index	Marks	Sign
1	21/3/24	import & export data	10	21/3/24
2	28/3/24	Lab 2 - End to End project	10	28/3/24
3	4/4/24	Simple Linear Regression Multiple Linear Regression	10	4/4/24
4	18/4/24	Week 4	8	
5	25/4/24	Logistic Regression	8	
6	9/5/24	KNN	9	23/5/24
7	9/5/24	SVM		
8	23/5/24	ANN using BP	10	
9	23/5/24	Random forest		
10	23/5/24	ADA boost		
11	30/5/24	K-Means	10	30/5/24
12	30/5/24	PCA		

21/03/24

Write a python program to import and export data using pandas library function

```
import pandas as pd
airbnb-data = pd.read-csv ("data/listings-austin.csv"
airbnb-data.head ()
```

Output :

id	name	host-id	host-name	neighbourhood -group	latitude
2265	Zen-east	2466	Paddy	null	78702 97.812
5245	Austin	2466	Paddy	null	78729 -98.98

Reading data from URL

```
url = "http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
col_names = ["sepal-length-in-cm",
             "sepal-width-in-cm",
             "petal-length-in-cm",
             "petal-width-in-cm",
             "class"]

iris_data = pd.read_csv(url, names = col_names)
iris_data.head()
```

Output

Sepal-len-in-cm	Sepal-width	petal-length	petal-width
5.1	6.0	1.4	0.1
5.9	7.0	1.3	0.2
2.8	4.8	1.4	0.1

Exporting dataframe to a CSV file

```
iris_data.to_csv("cleaned_iris_data.csv")
```

Output

	Sepal-length	Sepal-width	petal-length	petal-width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.3	0.2
2	4.7	3.2	1.3	0.2
	4.4	3.1		

28/03/24

Get the Data

```
import os
import tarfile
import urllib

DOWNLOAD_ROOT = "https://raw.github.com/ageron/handson/"
HOUSING_PATH = os.path.join("data", "01")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz")

fetch_housing_data()

def load_housing_data(housing_path = HOUSING_PATH):
data_path = os.path.join(housing_path, "housing.csv")
return pd.read_csv(data_path)

housing.head()
```

Discover and Visualize the Data

```
strat_train_set.shape ; strat_test_set.shape

strat_test_set.reset_index().to_feather(fname='data/01/
                                         strat_test_set.f')

housing = strat_train_set.copy(); housing.shape

housing.plot(kind='scatter', x='longitude', y='latitude')
plt.show()
```

Prepare the Data for Machine Learning Algorith

```
housing = strat_train_set.drop("median_house_value", axis
housing_labels = strat_train_set["median_house_value"].co
housing.shape, housing_labels.shape.

imputer = SimpleImputer(strategy="median")
imputer.fit(housing_num)
imputer.statistics_
```

Select and Train a model

```
lin_reg = LinearRegression()
lin_reg.fit(X=housing_prepared, y=housing_labels)
some_data = housing.iloc[:5]
housing_predictions = lin_reg.predict(housing_prepared)
lin_rmse = np.sqrt(lin_mse)
lin_rmse
```

Fine-Tune Your Model

```
from sklearn.model_selection import GridSearchCV
param_grid = [
    'n_estimators': [3,10,30], 'max_features': [2 4 6 8]
forest_reg = RandomForestRegressor()
```

```
grid_search.fit(X=housing_prepared, y=housing_labels)

cvres = grid_search.cv_results

X_test = strat_test_set.drop(labels='median_house_value')
y_test = strat_test_set['median_house_value'].copy()
```

Sneha
28/12/24

4/4/24 Simple Linear Regression

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection
```

Import Data

```
df-sal = pd.read-csv('/content/Salary_data.csv
```

Analyze Data

Describe

```
df-sal.describe()
```

Distribution

```
sns.distplot(df-sal['Salary']
```

Relationship between Salary and Experience

```
plt.scatter(df-sal['yearofExperience'], df-sal['Salary
plt.title('Salary vs Experience')
plt.show()
```

Split Data

Split into Independent/dependent variables

```
x = df-sal.iloc[:, :1]
y = df_sal.iloc[:1:]
```

Visualize Predictions

Prediction on training Sets

```
plt.scatter (x_train , y_train , color = 'light coral')
plt.plot ( x_train , y_pred_train ')
plt.xlabel ('Years of Experience')
plt.ylabel ('Salary')
```

Prediction on Test Set

```
plt.scatter (x_test , y_test)
```

Co-efficient and Intercept

```
print (f'co-efficient : { regressor.coef_ }')
```

4/4/24 Multiple Linear Regression

Import Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
```

Import Data

```
df_start = pd.read_csv('/content/50_Startup.csv')
```

Analyze Data

Describe

```
df_data_describe()
```

Distribution

```
plt.title('Profit Distribution Plot')
plt.show()
```

Split into Independent | Dependent

```
x = df_start.iloc[:, :-1]
y = df_start.iloc[:, :1]
```

Train Model

```
regressor = LinearRegression()
regressor.fit(x_train, y_train)
```

Predict Results

y_pred = regressor.predict(x_test)

Compare Prediction

np.set_printoptions(precision=2)

Sneha
4/4/24

18/4/24 Week 4

Use an appropriate data set for building the decision tree and apply this knowledge to classify

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.tree import DecisionClassification
from sklearn import tree
iris = load_iris()
y = iris_data
y = iris_length
from sklearn.model_selection import train
y_train, y_test = train_test_split(y, y, test_0.43)
tree model = decision classifier
tree model = tree model.fit (y_train, y_train)
plt.figure (figsize = (15, 20)
tree.plot_tree (tree model, filled = True,
                name = iris_feature_name);
```

<u>Output</u>

Plot width = 0.8
gini = 0.663
Sample = 85
Values = 24, 30, 8

gini = 0.0
Sample = 24
Values = 24, 0, 0

Petal Width ≤ 1.4
gini 0.5
Sample 61
Value = 0, 30, 31

```
y_pred = tree.Model.predict(x_test)
from sklearn metrics import accuracy--
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy", accuracy)
```

OUTPUT

Accuracy = 0.96923

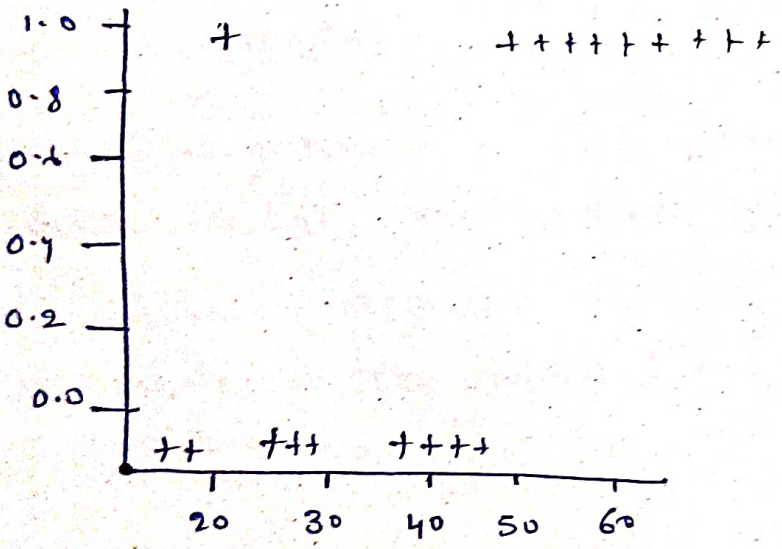
22/5/24

25/4/24

Logistic Regression

Week 5

```
import pandas as pd
from matplotlib import pyplot as plt
df = pd.read_csv("insurance_data.csv")
df.head()
plt.scatter(df.age, df.bought_insurance, marker=+
            colour= red)
from sklearn.model_selection
```




```
from sklearn.model_selection import train_
x = df[['age']]
y = df['bough_insurance']

# Perform the split
x_train, x_test, y_train, y_test = train_test_
        split(x, y, train_line =0.8)

# Display x_test
```

print (x-test)

	age
2	47
4	61
5	56
21	26
22	40
17	58

```
from sklearn.linear_model import logistic_requirement
model = logistic_requirement()
model.fit (x_train, y_train)
Print (x_test)
y_predicted = model.predict (x_test)
probability = model.predic (x_test, y_test)
print (probability)
print (score)
```

predicted values [1 1 1 1 1 0]

probabilities [[0.09251522, 0.90218477]
[0.40313939, 0.67676061]
[0.37595921 0.624013]
[0.2760642, 0.7239358]]

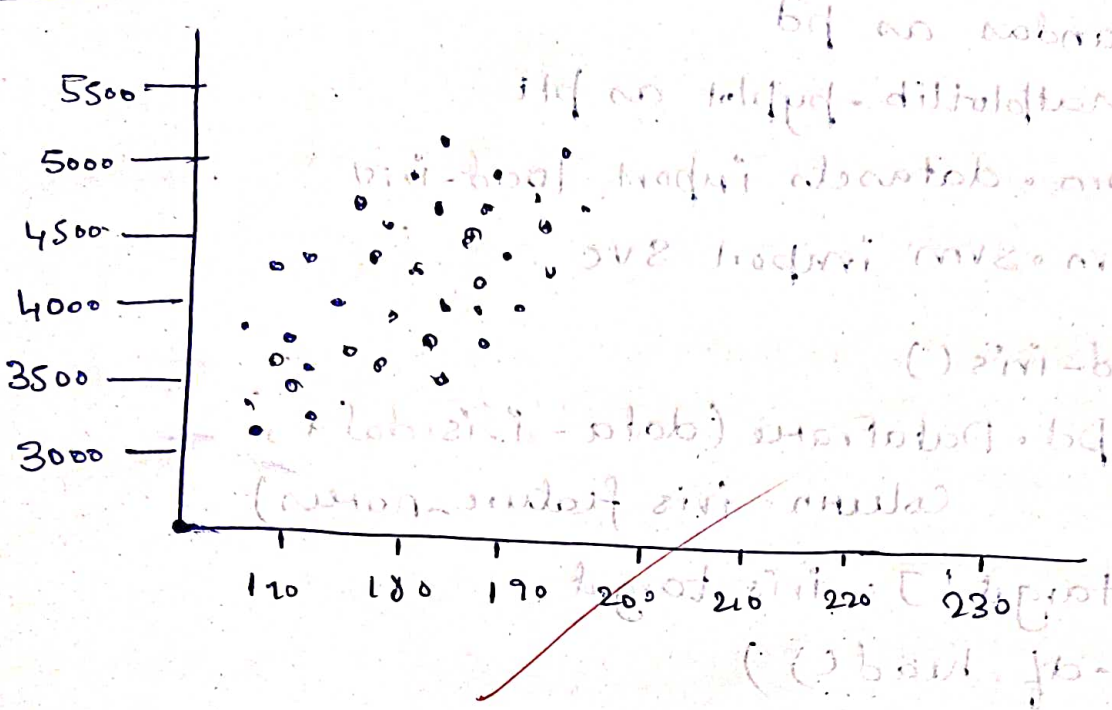
Accuracy 1.0

9/5/24

KNN

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("https://raw.githubusercontent/penguins.csv")

print(df.head())
sns.scatterplot(x="flipper_length_mm",
                y="body_mass_g", hue="species
plt.title("Penguin species classification")
plt.xlabel("Flipper length")
plt.ylabel("Body Mass(g)")
plt.show()

df.dropna(inplace=True)
x = df[['flipper_length', 'body_mass_g']]
y = df['species']
X_train, X_test, y_train = test_size=0.2.
knn = KNeighborClassifier(n_neighbor=3)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)
print("Accuracy", accuracy)
```

Output

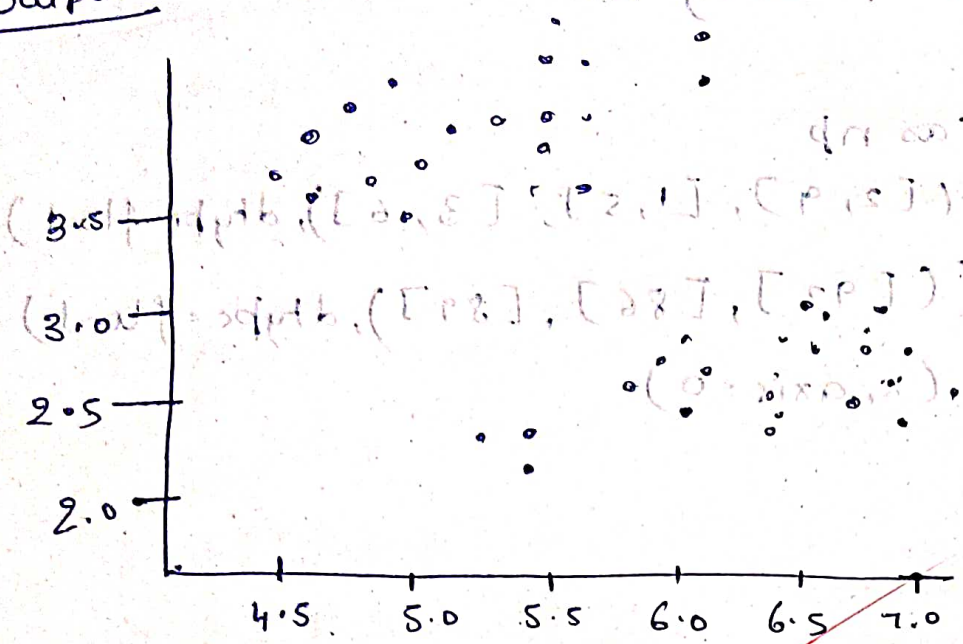


9/5/24

SVM

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.svm import SVC

iris = load_iris()
iris_df = pd.DataFrame(data = iris.data,
                       column = iris.feature_names)
iris_df['target'] = iris.target
print(iris_df.head())

plt.figure(figsize = (10, 6))
plt.scatter(iris_df['sepal length (cm)'], iris_df
 ['sepal width (cm)'], c = iris_df['target'],
 cmap = 'viridis')
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.title('Iris dataset')

X_train, X_test, y_train, y_test = train_test_split(iris.data,
iris.target, test_size = 0.3,
random_state = 42

svm_classifier = SVC(kernel = 'linear')
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)

print("Accuracy of SVM classifier", accuracy)
```

Output



23/5/24

23/5/24

ANN using BP

```
import numpy as np
x = np.array([[2, 9], [1, 5], [3, 6]], dtype=float)
y = np.array([[92], [86], [89]], dtype=float)
x = x / np.max(x, axis=0)
y = y / 100
epoch = 5000
lr = 0.1
inputlayer_neurons = 2
hiddenlayer_neurons = 3
output_neurons = 1

wh = np.random.uniform(size=(inputlayer_neurons,
                             hiddenlayer_neurons))
bh = np.random.uniform(size=(1, hiddenlayer_neurons))
wout = np.random.uniform(size=(hiddenlayer_neurons,
                               output_neurons))
bout = np.random.uniform(size=(1, output_neurons))

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

def derivatives_sigmoid(x):
    return x*(1-x)

for i in range(epoch):
    hinpl = np.dot(x, wh)
```

```
hinp = hinp1 + bh
hlayer_act = sigmoid(hinp)
outinp1 = np.dot(hlayer_act, wout)
outinp = outinp1 + bout
output = sigmoid(outinp)

hiddengrad = derivatives_sigmoid(hlayer_act)
d_hiddenlayer = Eh * hiddengrad

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

<u>Output</u>

Input

```
[[0.66667  1]
 [0.3333   0.55555]
 [1        0.66667]]
```

Actual output

```
[[0.92]
 [0.86]
 [0.89]]
```

Predicted Output

```
[[0.784187 18]
 [0.765399 93]
 [0.788927287]]
```

23/5/24

Random Forest Algorithm

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test
from sklearn.metrics import accuracy_score

iris = load_iris()
x = iris.data
y = iris.target

x_train, x_test, y_train, y_test = train_test_split
    (x, y, test_size = 0.5, random_state = 32)

rf_classifier = RandomForestClassifier()
rf_classifier.fit(x_train, y_train)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: " accuracy)
```

Accuracy: 0.93333

```
classification_rep = classification_report(y_test, y_pred
print("\n Classification Report \n", classification_rep)
```

Output

Classification Report

	precision	recall	f1-score	support
0	1.00	1.00	1.00	30
1	0.81	1.00	0.89	21
2	1.00	0.79	0.88	24
Accuracy			0.93	75
macro avg	0.94	0.93	0.93	75
Weighted avg	0.95	0.93	0.93	75

23/5/24

ADA Boost

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_split
from sklearn.metrics.

iris = load_iris()
x = iris.data
y = iris.target

x_train, x_test, y_train, y_test = train_test_split
                                   (x, y, size=0.4)

adaboost_clf = AdaBoostClassifier (n_estimators = 30,
learning_rate = 1.0, random_state = 42)
adaboost_clf.fit (X_train, y_train)

accuracy = accuracy_score(y_test, y_pred)
print ("Accuracy": accuracy)
```

<u>Output</u>

Accuracy : 0.96666

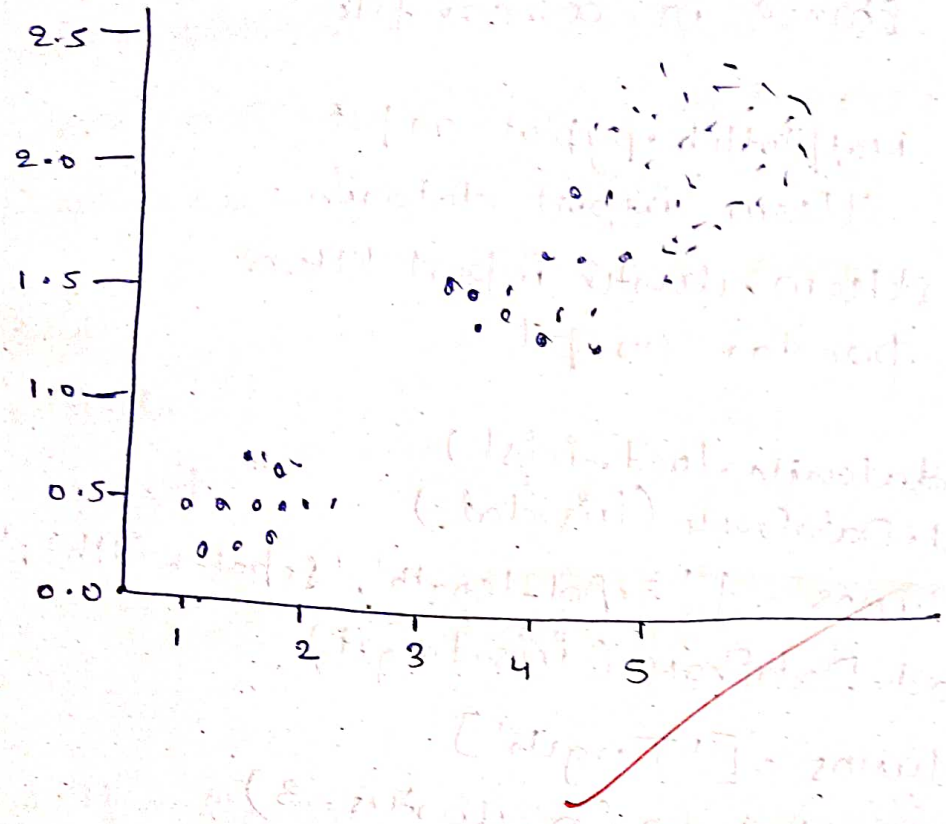
23/5/24

30/5/24

Build k-means algorithm to cluster a set of data stored in a .csv file

```
import matplotlib.pyplot as plt
import sklearn import datasets
from sklearn.cluster import KMeans
import pandas as pd

iris = datasets.load_iris()
x = pd.DataFrame (iris.data)
x.columns = ['sepal-length', 'sepal-width', 'petal-length']
y = pd.DataFrame (iris.target)
y.columns = ['Targets']
model = KMeans (n-clusters = 3)
model.fit(x)
plt.figure (figsize = (14, 14))
colormap = np.array (['red', 'lime', 'black'])
plt.subplot (2, 2, 1)
plt.scatter (x.petal-length, x.petal-width,
            c = colormap [y.targets], s = 40)

plt.title ('Real clusters')
plt.xlabel ('Petal length')
plt.ylabel ('Petal Width')
```

Output

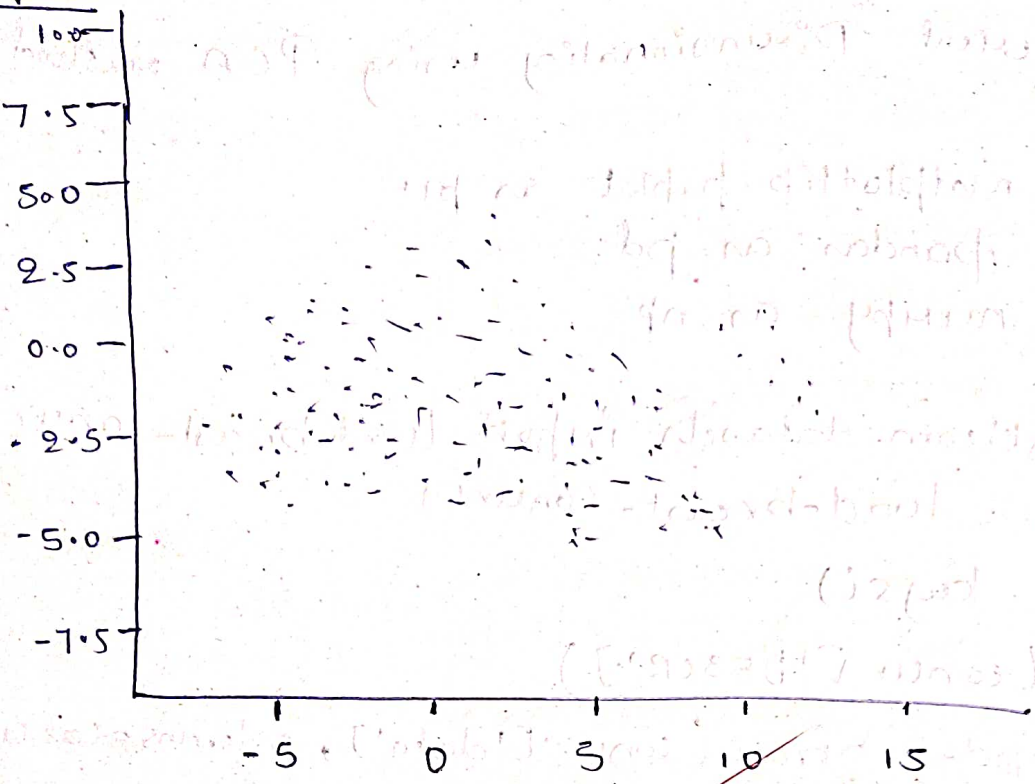


30/5/24

Implement Dimensionality using PCA method

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

from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
cancer.keys()
print(cancer['DESCR'])
df = pd.DataFrame(cancer['data'], columns=cancer
        ['feature_names'])
df.head()

pca = PCA(n_components=2)
pca.fit(scaled_data)
x_pca = pca.transform(scaled_data)
scaled_data.shape

x_pca.shape

plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0], x_pca[:,1],
            c=cancer['target'], cmap='plasma')
```

Output



Sneha
30/5/24