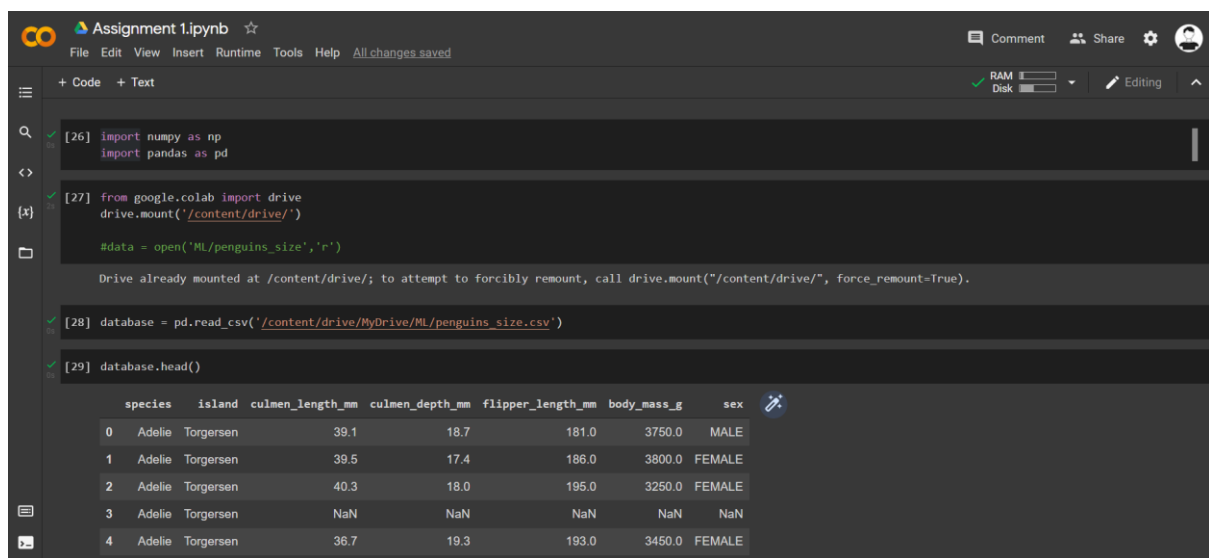


Name : Kshitij V Darwhekar

Roll No: TETB19

Sub : Soft Computing

Q1. Perform following task on any dataset



The screenshot shows a Jupyter Notebook titled "Assignment 1.ipynb". The code cells are as follows:

```
[26] import numpy as np
import pandas as pd

[27] from google.colab import drive
drive.mount('/content/drive/')

#data = open('ML/penguins_size','r')

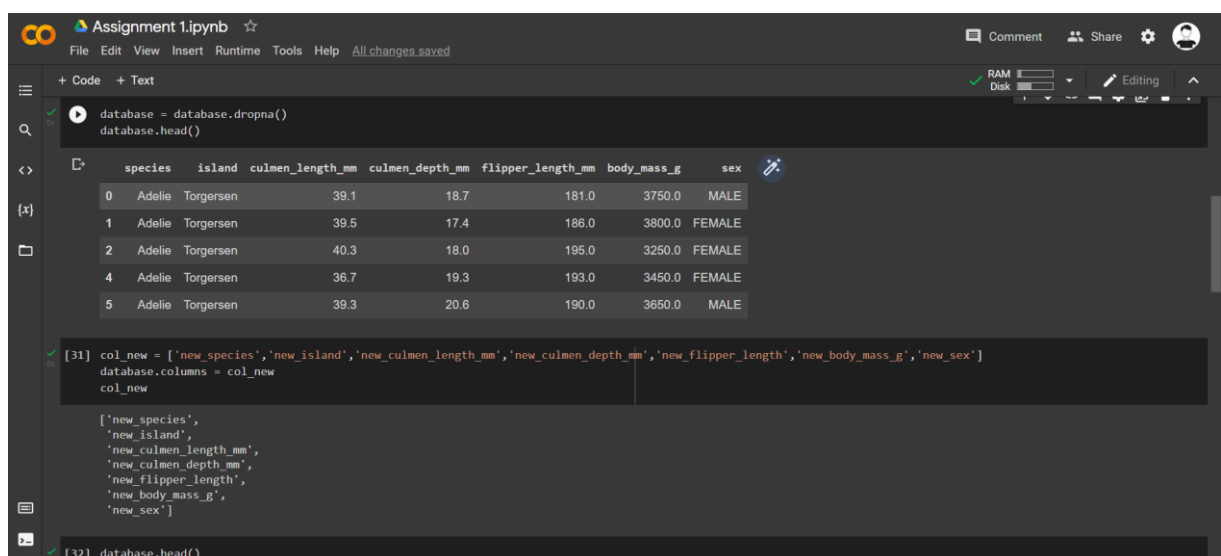
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force_remount=True).

[28] database = pd.read_csv('/content/drive/MyDrive/ML/penguins_size.csv')

[29] database.head()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
3	Adelie	Torgersen	NaN	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE

1. Change column name present in dataset



The screenshot shows the same Jupyter Notebook with additional code cells:

```
database = database.dropna()
database.head()
```

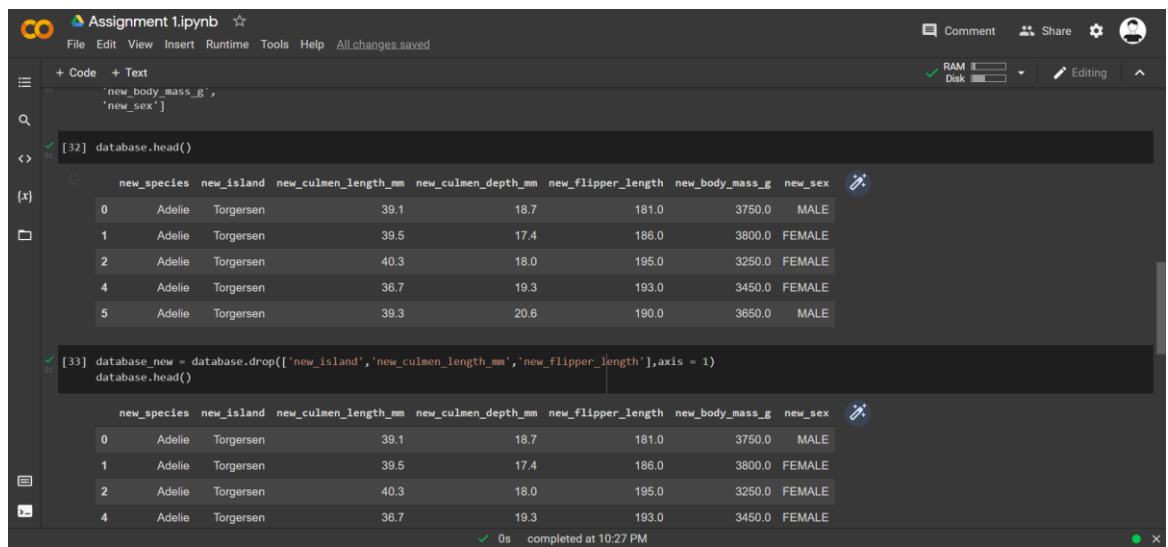
	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE

```
[31] col_new = ['new_species', 'new_island', 'new_culmen_length_mm', 'new_culmen_depth_mm', 'new_flipper_length', 'new_body_mass_g', 'new_sex']
database.columns = col_new
col_new

['new_species',
 'new_island',
 'new_culmen_length_mm',
 'new_culmen_depth_mm',
 'new_flipper_length',
 'new_body_mass_g',
 'new_sex']

[32] database.head()
```

2. Drop unessential column



The screenshot shows a Jupyter Notebook titled "Assignment 1.ipynb". The code cell [32] contains the command `database.head()`, which displays a table with 7 columns: `new_species`, `new_island`, `new_culmen_length_mm`, `new_culmen_depth_mm`, `new_flipper_length`, `new_body_mass_g`, and `new_sex`. The code cell [33] contains the command `database_new = database.drop(['new_island', 'new_culmen_length_mm', 'new_flipper_length'], axis = 1)` followed by `database_new.head()`. The resulting table has 5 columns: `new_species`, `new_culmen_depth_mm`, `new_body_mass_g`, `new_sex`, and `location`.

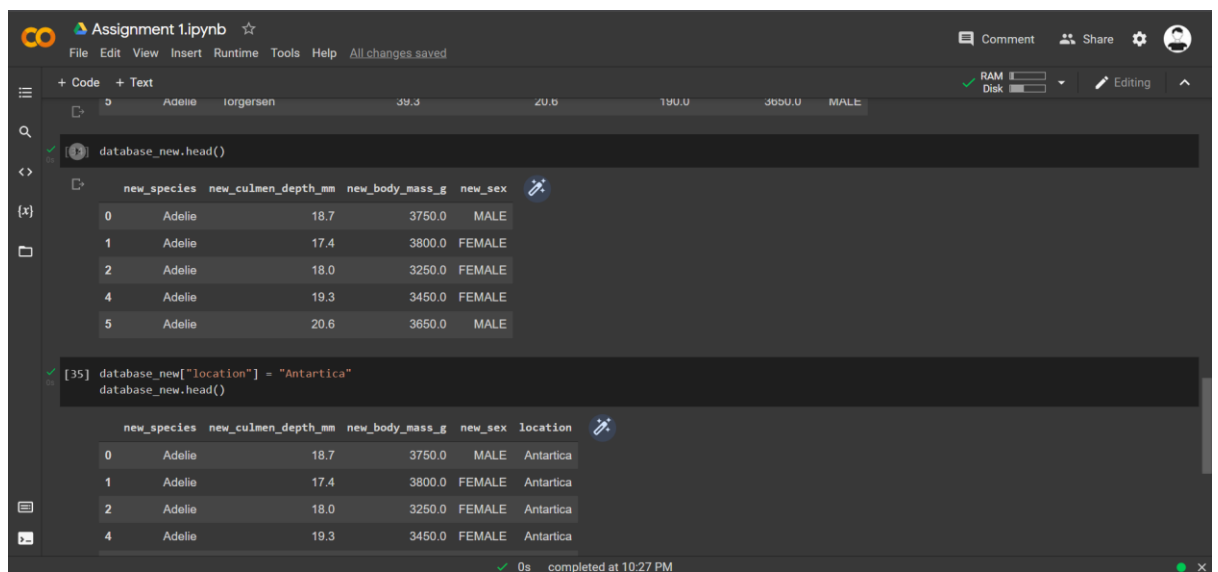
```
[32] database.head()
```

	new_species	new_island	new_culmen_length_mm	new_culmen_depth_mm	new_flipper_length	new_body_mass_g	new_sex
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0	MALE
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0	FEMALE
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0	FEMALE
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0	FEMALE
5	Adelie	Torgersen	39.3	20.6	190.0	3650.0	MALE

```
[33] database_new = database.drop(['new_island', 'new_culmen_length_mm', 'new_flipper_length'], axis = 1)
database_new.head()
```

	new_species	new_culmen_depth_mm	new_body_mass_g	new_sex
0	Adelie	18.7	3750.0	MALE
1	Adelie	17.4	3800.0	FEMALE
2	Adelie	18.0	3250.0	FEMALE
4	Adelie	19.3	3450.0	FEMALE

3. Add new columns in dataset



The screenshot shows a Jupyter Notebook titled "Assignment 1.ipynb". The code cell [34] contains the command `database_new.head()`, which displays a table with 5 columns: `new_species`, `new_culmen_depth_mm`, `new_body_mass_g`, `new_sex`, and `location`. The code cell [35] contains the command `database_new["location"] = "Antartica"` followed by `database_new.head()`. The resulting table has 5 columns: `new_species`, `new_culmen_depth_mm`, `new_body_mass_g`, `new_sex`, and `location`.

```
[34] database_new.head()
```

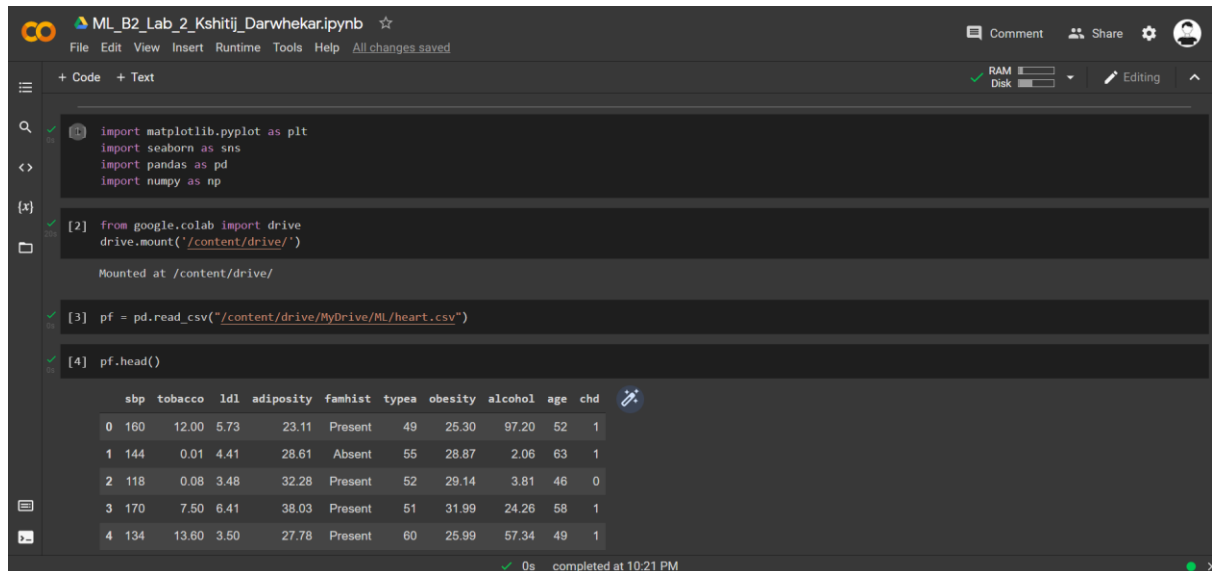
	new_species	new_culmen_depth_mm	new_body_mass_g	new_sex
0	Adelie	18.7	3750.0	MALE
1	Adelie	17.4	3800.0	FEMALE
2	Adelie	18.0	3250.0	FEMALE
4	Adelie	19.3	3450.0	FEMALE
5	Adelie	20.6	3650.0	MALE

```
[35] database_new["location"] = "Antartica"
database_new.head()
```

	new_species	new_culmen_depth_mm	new_body_mass_g	new_sex	location
0	Adelie	18.7	3750.0	MALE	Antartica
1	Adelie	17.4	3800.0	FEMALE	Antartica
2	Adelie	18.0	3250.0	FEMALE	Antartica
4	Adelie	19.3	3450.0	FEMALE	Antartica

Q2. Explain overfitting, underfitting and confusion matrix with example

1. Select appropriate dataset

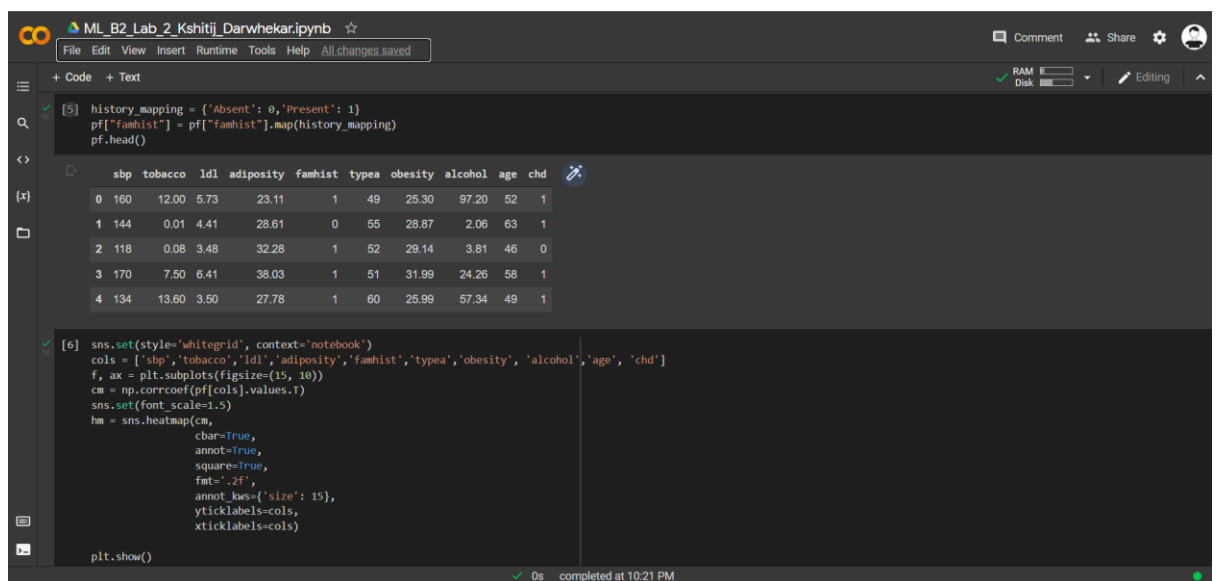


The first screenshot shows a Google Colab notebook titled "ML_B2_Lab_2_Kshitij_Darwhekar.ipynb". The code cells show the following steps:

- Importing libraries: `import matplotlib.pyplot as plt`, `import seaborn as sns`, `import pandas as pd`, and `import numpy as np`.
- Mounting the Google Drive: `from google.colab import drive`, `drive.mount('/content/drive/')`. The output shows "Mounted at /content/drive/".
- Reading the CSV file: `pf = pd.read_csv("/content/drive/MyDrive/ML/heart.csv")`.
- Viewing the first few rows: `pf.head()`.

The output of `pf.head()` is a table with 10 columns: `sbp`, `tobacco`, `ldl`, `adiposity`, `famhist`, `typea`, `obesity`, `alcohol`, `age`, and `chd`. The first five rows of data are shown.

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	160	12.00	5.73	23.11	Present	49	25.30	97.20	52	1
1	144	0.01	4.41	28.61	Absent	55	28.87	2.06	63	1
2	118	0.08	3.48	32.28	Present	52	29.14	3.81	46	0
3	170	7.50	6.41	38.03	Present	51	31.99	24.26	58	1
4	134	13.60	3.50	27.78	Present	60	25.99	57.34	49	1

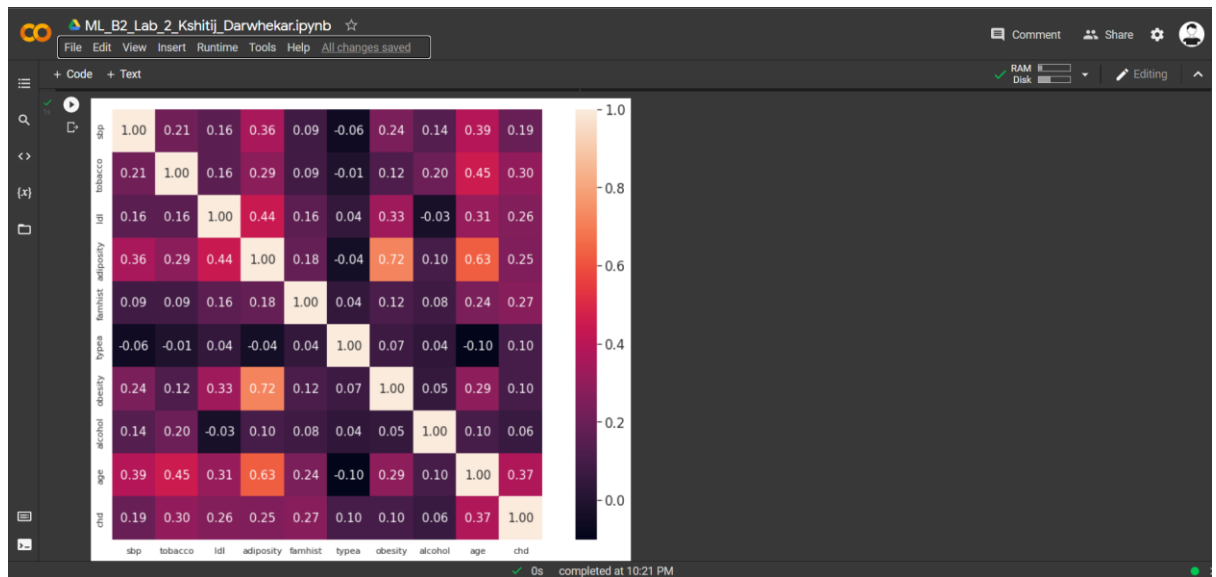


The second screenshot shows the continuation of the Colab notebook. The code cells show the following steps:

- Mapping the `famhist` column: `history_mapping = {'Absent': 0, 'Present': 1}`, `pf["famhist"] = pf["famhist"].map(history_mapping)`, and `pf.head()`.
- Viewing the first few rows: `pf.head()`.
- Creating a correlation heatmap: `sns.set(style='whitegrid', context='notebook')`, `cols = ['sbp', 'tobacco', 'ldl', 'adiposity', 'famhist', 'typea', 'obesity', 'alcohol', 'age', 'chd']`, `f, ax = plt.subplots(figsize=(15, 10))`, `cm = np.corrcoef(pf[cols].values.T)`, `sns.set(font_scale=1.5)`, `hm = sns.heatmap(cm, char=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 15}, yticklabels=cols, xticklabels=cols)`, and `plt.show()`.

The output of `pf.head()` is a table with 10 columns: `sbp`, `tobacco`, `ldl`, `adiposity`, `famhist`, `typea`, `obesity`, `alcohol`, `age`, and `chd`. The first five rows of data are shown.

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	160	12.00	5.73	23.11	1	49	25.30	97.20	52	1
1	144	0.01	4.41	28.61	0	55	28.87	2.06	63	1
2	118	0.08	3.48	32.28	1	52	29.14	3.81	46	0
3	170	7.50	6.41	38.03	1	51	31.99	24.26	58	1
4	134	13.60	3.50	27.78	1	60	25.99	57.34	49	1



2. Find confusion matrix present analysis

```

x = pf[['tobacco','ldl','adiposity','famhist','typea','obesity','alcohol','age']].values
y = pf[['chd']].values

[8] from sklearn.model_selection import train_test_split

[9] x_train , x_test , y_train,y_test = train_test_split(x,y,train_size = 0.9)

[10] # Apply logistic regression

from sklearn.linear_model import LogisticRegression

[18] model = LogisticRegression(C=1,penalty='l2')
model.fit(x_train,y_train)
y_pred=model.predict(x_test)

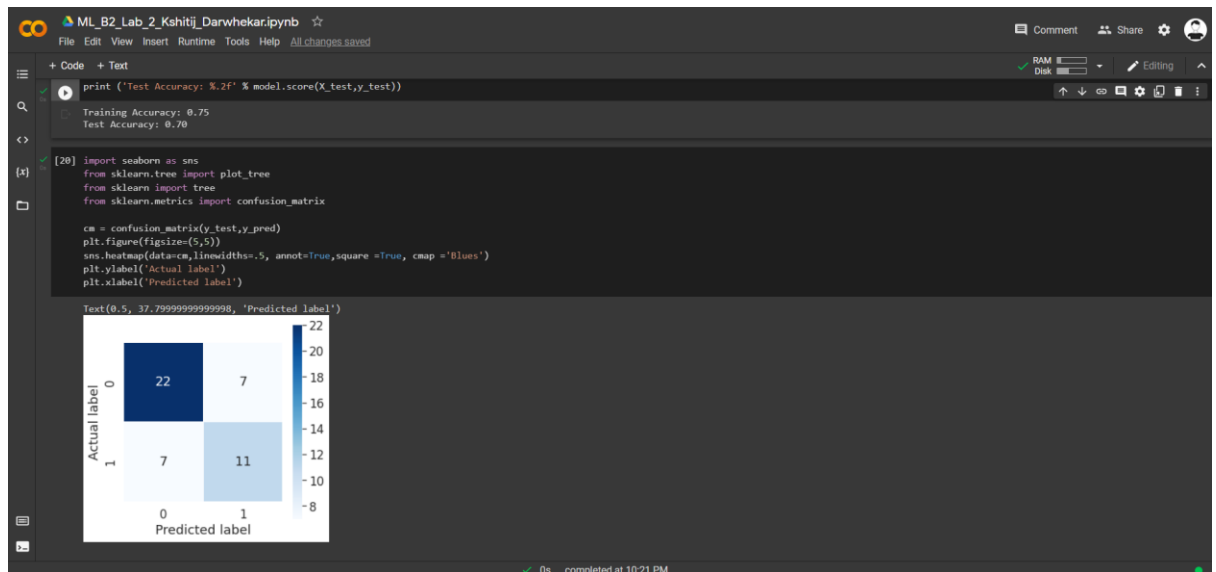
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to
y = column_or_1d(y, warn=True)

[19] print ('Training Accuracy: %.2f' % model.score(x_train,y_train))

print ('Test Accuracy: %.2f' % model.score(x_test,y_test))

Training Accuracy: 0.75
Test Accuracy: 0.70

```



3. Analyze overfitting of model

Name:- Kshitij V. Darwhekar
Roll no:- TETB19
Sub:- Soft Computing

Q2.
⇒

3. Overfitting

Overfitting occurs when our machine learning model tries to cover all the data points or more than the required data points present in given dataset. Because of this, the model ~~has~~ starts caching noises & inaccurate values present in the dataset, and all these factors reduce the efficiency & accuracy of the model. The overfitted model has low bias & high variance.

we can avoid overfitting in our model by

- cross-validation
- Training with more data
- Removing features
- Early stopping the training

4. Analyze underfitting of model

4. Underfitting

Underfitting occurs when our machine learning model is not able to capture the underlying trend of the data. To avoid the overfitting in the model, the fed training data can be stopped at early stage, due to which the model may not learn enough from training data.

In case of underfitting the model is not able to learn enough from training dataset & hence it reduces the accuracy and produce unreliable predictions. An underfitted model has high bias & low variance.

we can avoid underfitting:-

- By increasing the training time of the model
- By increasing no. of features