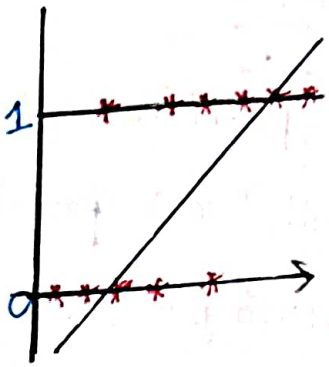
19/04/22 10:00pm

Logistic Regression

*** Que:- Why it is called Logistic Regression? it is classificatio

Ans :-

* First, Apply The Linear Regression



$$y = b_0 + b_1 * x$$

⇓

Convert it to "Probability"

⇒ Sigmoid function:- $P = \frac{1}{1+e^{-y}}$

$$= \frac{1}{1+\frac{1}{e^y}} = \frac{1}{\frac{e^y+1}{e^y}} \Rightarrow \frac{e^y}{1+e^y}$$

⇒ < 1 (+ve) (0 to 1)

When denominator is higher than the Newmonator, any(-)value can occur

Ex:- 5^{40}

$5^{-40} = \frac{1}{5^{40}} =$

powers will not give negative value

Answer :- < 1

A:- The Output we are predicting is

* probability = Continous → Regression

So, it called as Logestic Regression

Actual y

Predicting Probability

Predicted y

Ex :-

Age	y	$\hat{y}$	Prob	Pred y
20	1	120.3	0.6	1
21	1	–	0.8	1
22	1	–	0.7	1
24	0	–	0.4	0
26	0 x	–	0.54	1 x
21	1	–	0.6	1
28	0 x	–	0.72	1 x

Logistic Regression

class

Ex:

$$P = \frac{1}{1+e^{-y}}$$

$$P = \frac{1}{1+e^{-120.3}}$$

$$P = \frac{e^{y}}{e^{y}+1}$$

$$P = \frac{e^{-120.3}}{e^{-120.3}+1}$$

$\therefore e = 2.71$ → Constant

$$\hat{P} = \frac{2.71^{-120.3}}{2.71^{-120.3}+1}$$

$$y = b_0 + b_1 x$$

(Ex:-)

$= 2.3 + 5.9 x$

$= 2.3 + 5.9[20]$

$= 2.3 + 118$

$= 120.3$

Convert this value in probability

We apply "Discritization"

* Convert continuous variable To discrete Variable

$P \leq 0.5 \Rightarrow 0.0$ to $0.5 \Rightarrow 0$

$P > 0.5 \Rightarrow 0.5$ to $1 \Rightarrow 1$

→ Sigmoid Function

$$P = \frac{1}{1+e^{-y}}$$

$$1+e^{-y} = \frac{1}{P}$$

$$e^{-y} = \frac{1}{P} - 1$$

$$e^{-y} = \frac{1-P}{P}$$

applying both sides −1, opposite

$$\left\{ e^{y} = \frac{P}{1-P} \right\}$$

* Apply log on both sides

$$\log_e e^{y} = \log_e \left[\frac{P}{1-P}\right]$$

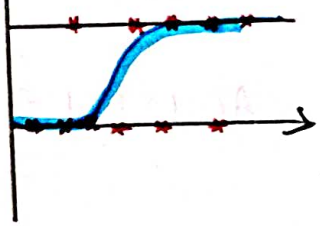
$$y = \log_e \left[\frac{P}{1-P}\right]$$

$$y = \ln\left[\frac{P}{1-P}\right]$$

$$y = b_0 + b_1 * x$$

$$\ln\left[\frac{P}{1-P}\right] = b_0 + b_1 * x$$

$$\ln\left[\frac{P}{1-P}\right] = b_0 + b_1 * x$$

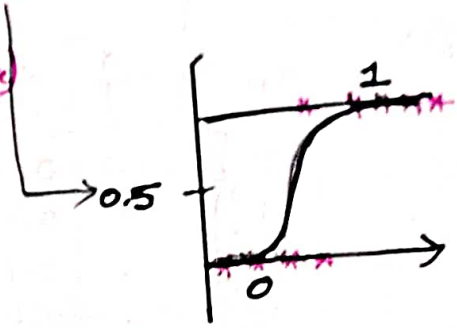


$\therefore \log_{10} = \log$

$\log_e = \ln$

Que :- In logistic Regression, threshold of 0.5 is Fixed?

Ans :- Higher The AUC (Area Under Curve) is Best Model



Ex :-

y	$\hat{p}$	Thre 0.5	Thre 0.6	Thre 0.4	Thre 0.8
1	0.65	1 ✓	1 ✓	1 ✓	0 ✗
1	0.55	1 ✓	0 ✗	1 ✓	0 ✗
1	0.35	0 ✗	0 ✗	0 ✗	0 ✗
0	0.28	0 ✓	0 ✓	0 ✓	0 ✓
0	0.44	0 ✓	0 ✓	1 ✗	0 ✓

1. Threshold = 0.5

Accuracy = $\frac{4}{5}$ = 80%

2. Threshold = 0.6

Accuracy = $\frac{3}{5}$ = 60%

3. Threshold = 0.4

Accuracy = $\frac{3}{5}$ = 60%

4. Threshold = 0.8

Accuracy = $\frac{2}{5}$ = 40%

* Confusion matrix also changes along with the Threshold setting.

Practical

CODE

Data

An Experiment was conducted on 5000 participants to study the effects of age and physical health on hearing loss. Specially the ability to hear high pitched tones. this data displays the result of study in which participants were Evaluated

and scored for physical ability and then had to take an audio test (pass / no pass) which Evaluated their ability to hear high frequencies. The age of the user was also noted. it is possible to build a model that would predict Someone's likelihood to hear the high Frequency Sound based solely on their Features (age and physical score?

* Features
  - ⇒ age = Age of participants in Years
  - ⇒ physical score = Score acheived during physical Exam.
* Label / Target
  - ⇒ test_result = 0 if no pass
    1 test passsed

```
# imports
# import numpy as np
# import pandas as pd
# import matplotlib.pyplot as plt
# import seaborn as sns

# Load
# df = pd.read_csv("hearing_test.csv")
# df.head()
```

df.head()

Out :

	Age	Physical_score	test_result
0	33.0	40.7	1
1	50.0	37.2	1
2	52.0	24.7	0
3	56.0	31.0	0
4	35.0	42.9	1

Hearing

df.shape

Out : [5000, 3]

df.isnull().sum()

Out : age 0
physical_score 0
test_result 0

20/4/22 12.11 Am
Dt:- 20/4/22
1:00 Pm

df.["Test_results"].value_counts()

Out : 1 3000
0 2000

⇒ Exploratory Data Analysis and Visualization

Logistic Regression

Step 1 : $y = \beta_0 + \beta_p x$

Step 2: Convert it Probability (p)

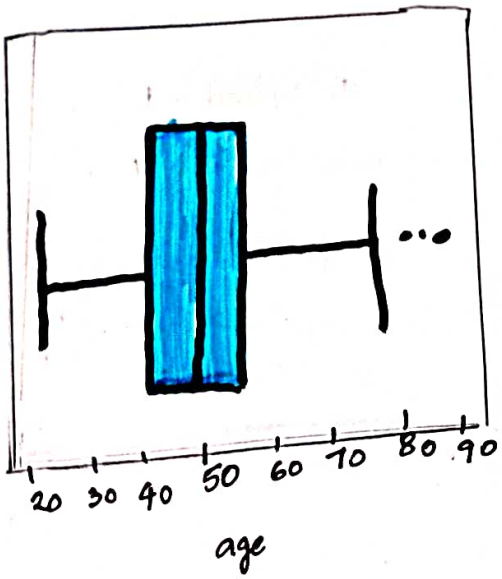
$$p = \frac{e^y}{e^y + 1}$$

Step 3: different threshold values

$p > 0.5 = 1$
$p \leq 0.5 = 0$

```
# sns.boxplot(df["age"])
# plt.show()
```

Out



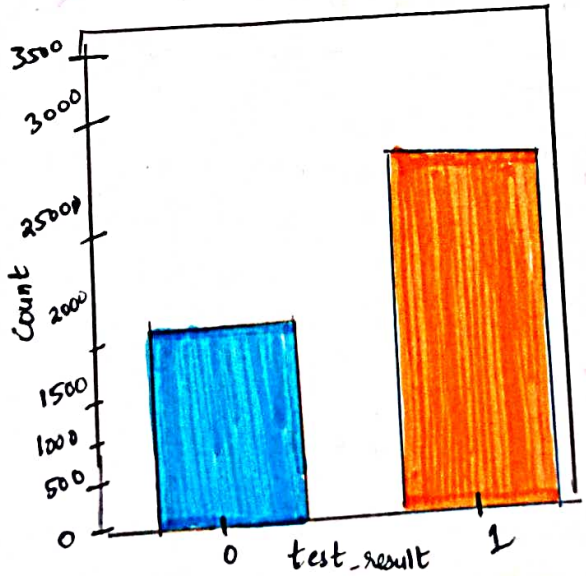

```
# df.describe()
```

Out:

	age	Phy_sc	test_result
count	5000.000000	5000.000000	5000.000000
mean	51.609000	32.760260	0.600000
std	11.287001	8.169802	0.489947
min	18.000000	-0.000000	0.000000
25%	43.000000	26.700000	0.000000
50%	51.000000	35.300000	1.000000
75%	60.000000	38.900000	1.000000
Max%	90.000000	50.000000	1.000000

```
# sns.countplot(data=df, x="test_result")
```

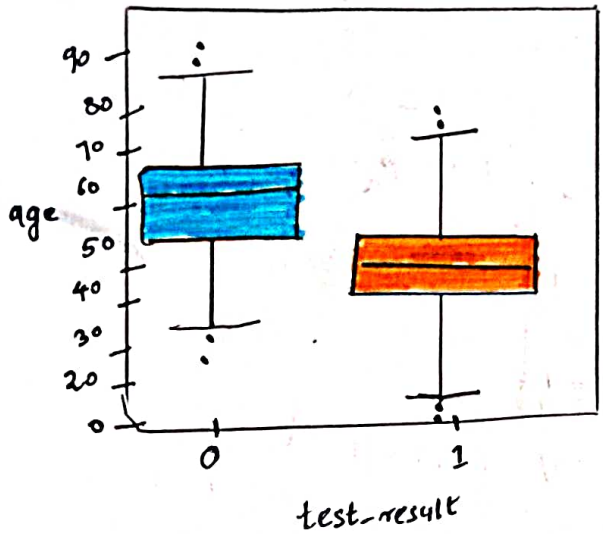
Out:



Compare with different columns.

sns.boxplot (x = "test_result", y = "age", data = df)

Out :

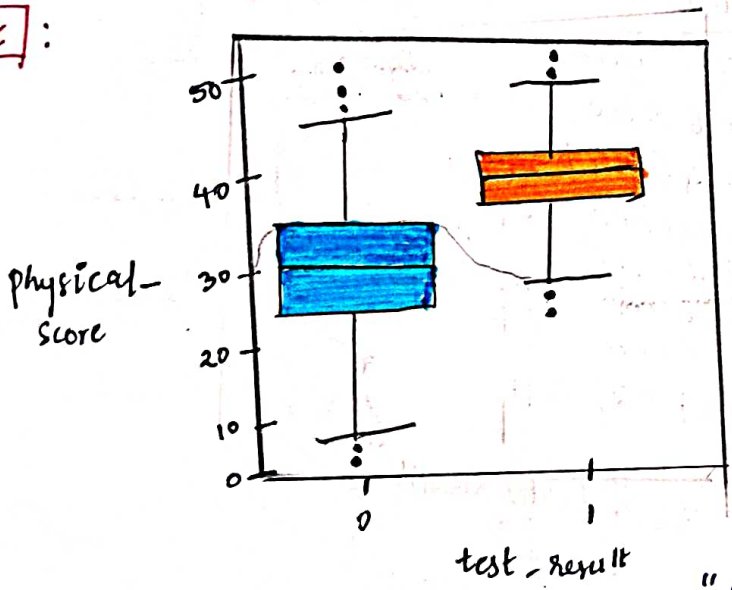


passed - 1
More less age people

Failed - 0
Age above 50 are failed.

sns.boxplot (x = "test_result", y = "Physical_score", data = df)

Out :



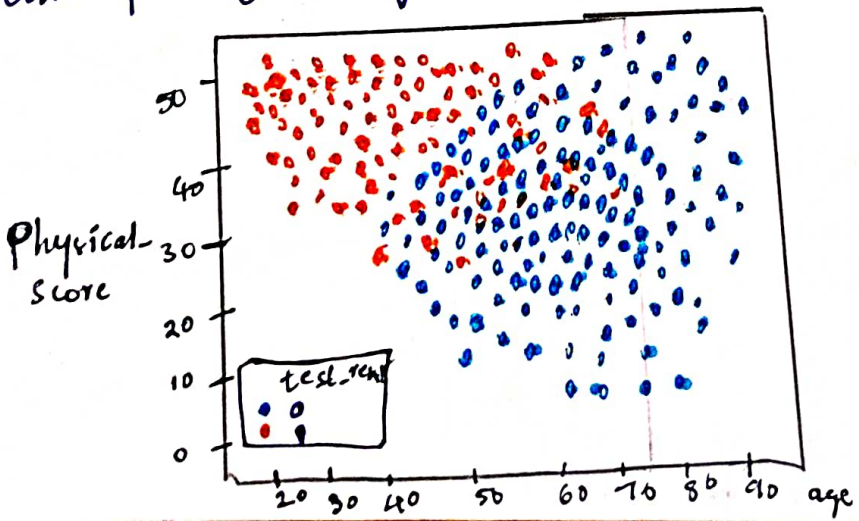
# Lower the physical score
Test_results = 0

# Higher the physical score
Test_results = 1

seperation

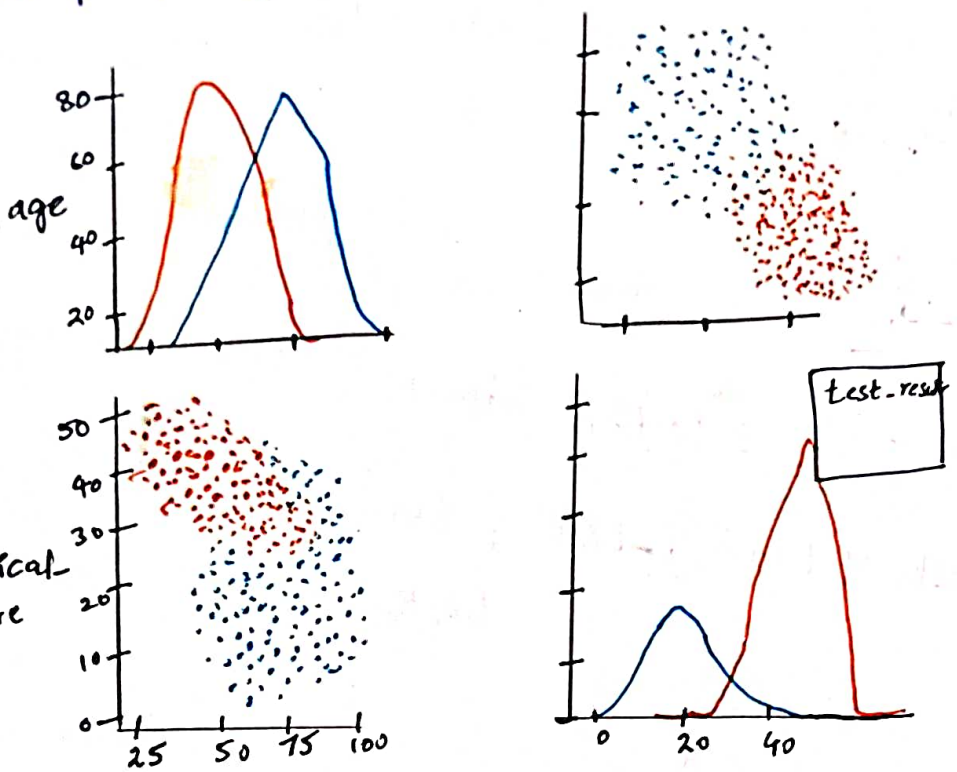
sns.scatterplot (x = "age", y = "physical_score", data = df, hue = "Test_result")

Out :



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

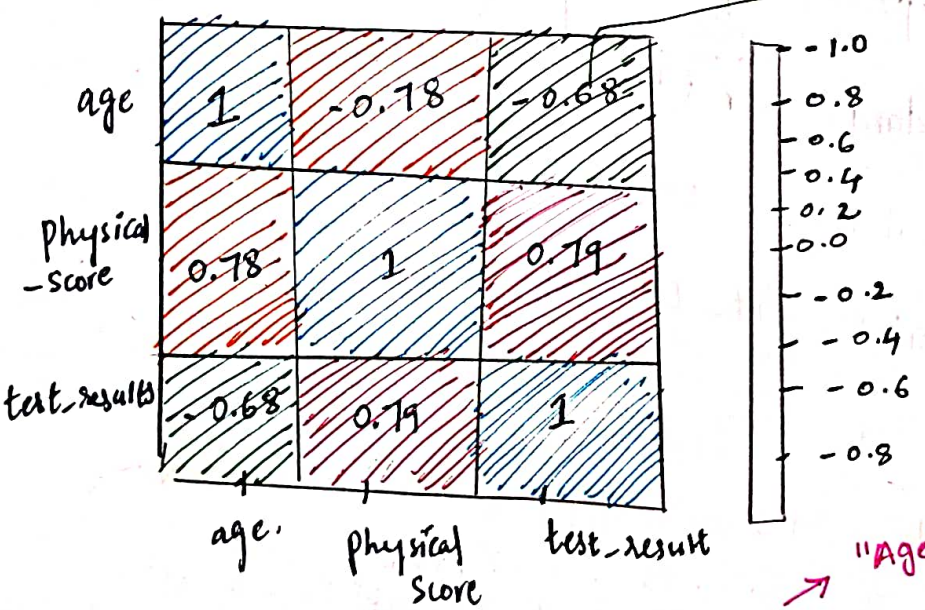
Out:



sns.heatmap(df.corr(), annot = True)

→ displaying numbers on graph

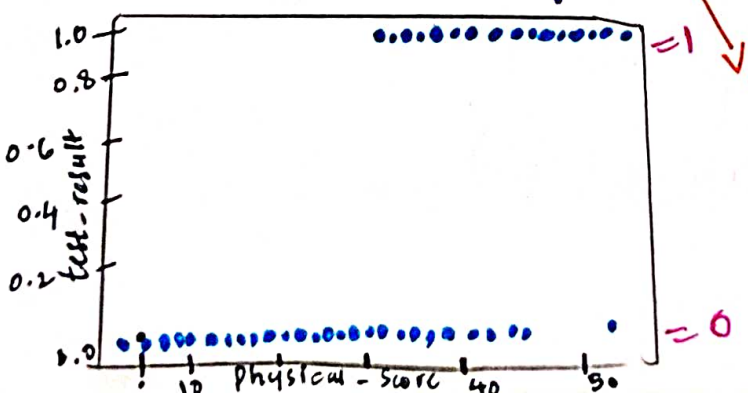
Out:



sns.scatterplot(x = "physical_score", y = "test_result", data_df)

"Age" ⇒ Same

Out



#X and Y

```
# x = df.drop("test_result", axis=2)
# y = df["test_result"]
```

⇒ **Train_test_split**

```
from sklearn.model_selection import train_test_split
# x_train, x_test, y_train, y_test = train_test_split(x, y,
                                     test_size=0.3, random_state=29)
```

⇒ **Scaling**

```
from sklearn.preprocessing import StandardScaler
# scaler = StandardScaler()
# x_train = scaler.fit_transform(x_train)
# x_test = scaler.fit_transform(x_test)
```

⇒ **MODELLING** (Basline (or) Raw model) ⇒ default parameter.

```
from sklearn.linear_model import LogisticRegression
# log_model = LogisticRegression()
# log_model.fit(x_train, y_train)
```

out: LogisticRegression()

⇒ Prediction

```
# Predicting on Train data
# ypred_train = log_model.Predict(x_train)
# Predicting on Test data
# ypred_test = log_model.Predict(x_test)
```

⇒ Evaluation

```
from sklearn.metrics import accuracy_score.
# Train Accuracy
# accuracy_score(y_train, ypred_train)
```

out : 0.919

```
# Test Accuracy
# accuracy_score(y_test, ypred_test)
```

out : 0.905

⇒ Cross Validation

```
from sklearn.model_selection import cross_val_score
# scores = cross_val_score(log_model, x, y, cv=5)
# print(scores)
# scores.mean()
```

out [0.933, 0.915, 0.908, 0.91, 0.91]

0.916 (Test accuracy, CV] should be equal +,- 5%)

```
from sklearn.metrics import Confusion_matrix
```

Edey Rayali, Ela ? ⇒ {Help} ⇒ yTrue = y
y_pred = ypred

```
# Confusion_matrix [y_test, ypred_test)
```

5000 data
3500 Train | 1500 Test

Out :

```
array([[ 515 , 82],
       [ 37 , 866 ]],
```

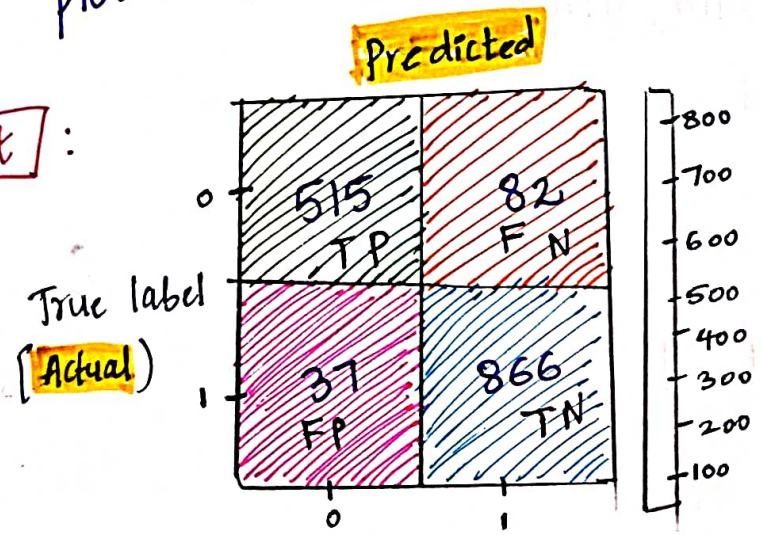
(0 = Actual, 1 = Actual; 0 = Predicted, 1 = Predicted)

```
from sklearn.metrics import plot_Confusion_matrix
```

{Help} Arguments.

```
# plot_Confusion_matrix (log_model, x_test, y_test)
```

Out :



random_state
29

```
#   513   76
    66   845
```

* Accuracy = $\frac{TP+TN}{TP+TN+FP+FN} = \frac{515+866}{515+866+37+82} = \frac{1381}{1500} = 0.91$

```
from sklearn.metrics import classification_report
```

```
# print (classification_report (y_test, ypred_test)
```

Out :

	Precision	recall	f1 score	support
0	0.93	0.86	0.90	597
1	0.91	0.96	0.94	903
accuracy			0.92	1500
macro . avg	0.92	0.91	0.92	1500
Weighted. avg	0.92	0.92	0.92	1500

→ Calculation

⇒ Precision (0) = $\frac{TP}{TP+FP}$

$= \frac{515}{515+37} = \frac{515}{552} = 0.93$

⇒ Precision (1) = $\frac{TN}{TN+FP}$

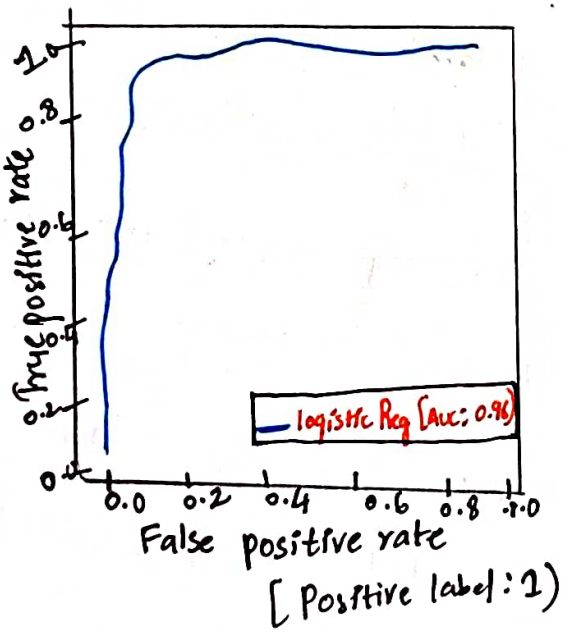
$= \frac{866}{866+82} = \frac{866}{948} = 0.91$

⇒ AUC

```
from sklearn.metrics import plot_roc_curve

# plot_roc_curve (log_model, X_test, y_test)
```

Out:



[Positive label: 1)

20/4/22
4:30pm