Date 4/3/25
Page 1

LAB-0

→ tickets - ["SBI BANK.NS", "ICICI BANK.NS", "KOTAK BANK.NS"]
data = yf.download(tickets, start = "2021-01-01", end = "2024-12-30", group by = 'ticket')

import yfinance as yf import pandas as pd import matplotlib.pyplot as plt

→ SBI - data = data['SBI BANK.NS']
SBI - data['Daily Return'] = SBI - data['close'].pct change()

Print("\n Daily returns for SBI Industries:")
Print(SBI_data['Daily Return'].head())
plt.figure(Fig Size = (12,6))
plt.subplot(2,1,1)
SBI_data['close'].plot(title = 'SBI Industries - closing price")
plt.subplot(2,1,2)
SBI - data['Daily return:].plot(title = "SBI Industries - Daily Returns").
plt.tight_layout()
plt.show()

→ Daily Returns for SBI Industries.

Date:	
2024-01-01	NaN
2024-01-02	0.000589
2024-01-03	-0:015420
2024-01-04	0.016730
2024-01-05	-0:005116

import pandas as pd

1. df = {'USN': [1, 2, 3, 4, 5], 'Name': ['A', 'B', 'C', 'D', 'E', 'F'], 'Marks': [10, 20, 30, 40, 50]}

df = pd.DataFrame(df)
Print (df)

	USN	Name	Marks
0	1	A	10
1	2	B	20
2	3	C	30
3	4	D	40
4	5	E	50

2. From sklearn.datasets import load_diabetes
diabetes = load_diabetes()
df = pd.DataFrame(diabetes.data, columns = diabetes.feature_names)
df['target'] = diabetes.target
df.head()

3. file_path = 'Sample_Sales_data.csv'
df = pd.read_csv(file_path)
Print (Sample_data)

	Product	Quantity	Price	Sales	Region
0	Laptop	5	1000	5000	North
1	Mouse	15	20	300	east

4. df.to_csv('output.csv', index = False)
Print ("Data saved to output.csv")

LAB-1

(I) Data Set considered : housing.csv

(i) To load .csv file into data frame.

```
import pandas as pd.
df = pd.read_csv('housing.csv')
```

(ii) To display information of all columns

⇒ Print (df.columns)

⇒ Index ('longitude', 'latitude', 'housing_median_age' 'total_rooms', 'total_bedrooms', 'population', 'household', 'median_house_value')

(iii) To display Statistical information of all numerical columns

⇒ df.describe()

~~longitude latitude housing_median_range.~~

	longitude	latitude	housing median range	totalRoom	population
count	20640	20640	20640	20640	20640

totalbedroom	household	median_income	house value
20433	20640	20640	20640

(iv) To display the count of unique tickets for "ocean_proximity" column.

```
l = df.ocean_boo proximity.unique()
print(l)
Print (len(l))
```

['NEAR BAY' 'OCEAN' 'INLAND' 'NEAR OCEAN' 'ISLAND']
5

(v) To display which columns in a data set having missing values count greater than 0

Print (df.columns[df.isnull().any()])
Index(['total_bedrooms'], dtype='object')

(II) data sets considered: diabetes.csv, income.csv
import pandas as pd.
df1 = pd.read_csv('diabetes.csv')
df2 = pd.read_csv('income.csv')

(1) which columns in the data set had missing values? How did you handle them?
Print (df.columns[df.isnull().any()])
Index([], dtype='object')

(2) which categorical columns did you identify in data set?
How did you encode them?
categorical_col = df.select_dtypes(exclude=['number']) columns.

diabetes: Gender / class.

adult: workclass, education, marital-status, occupation, relationship, race, gender, native-country income.

Date 11/3/25
Page 5

Linear Regression and Multiple linear Regression

Find linear Regression of data of week and product

x_i (week)	y_i (Sales in thousands)
1	2
2	4
3	5
4	9

Sol:

$$x = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \qquad y = \begin{bmatrix} 2 \\ 4 \\ 5 \\ 9 \end{bmatrix}$$

$$x^T x = \begin{bmatrix} 4 & 10 \\ 10 & 30 \end{bmatrix}$$

$$(x^T x)^{-1} = \begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 0.2 \end{bmatrix}$$

$$(x^T x)^{-1} x^T = \begin{bmatrix} 1.5 & 0.5 \\ -0.5 & 0.2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.5 \end{bmatrix}$$

$$\beta = \left((x^T x)^{-1} x x^T \right) y = \begin{bmatrix} 1 & 0.5 & 0 & 0.5 \\ -0.3 & -01 & 0.1 & 0.5 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \\ 5 \\ 9 \end{bmatrix}$$

$$= \begin{bmatrix} -0.5 \\ 2.2 \end{bmatrix}$$

$y = a_0 + a_1 x$

$y = -0.5 + 2.2x$

for $x = 5$

$y = -0.5 + (2.5)5 = 10.5$

*

1. import libraries
2. import data - distribution
3. Analyze data - distribution
4. Distribution plot - visualization.
5. Relationship b/w variables
6. Split the data.
7. Train the model
8. Predict the result.
9. Visualize prediction
10. check values of co-efficient & intercept

11/2/25

Date __/__/__
Page 7

* Linear Regression.

```
import Pandas as Pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import linear regression
from sklearn.impute import SimpleImputer.

file_Path = "canada_per_capita_income.csv"
data = Pd.read_csv ( file path)

data.fillna(data.mean(numeric_only=True),
inplace = True)

x = data [[ 'year']]
y = data [' per capita income (US$)']

model = linear Regression()
model.fit (X, Y)

year_2020 = np.array([[ 2020]])
income_2020 = model.predict ( year_2020)

plt.scatter(X, y, color = 'blue', label = "Actual Data")
plt.plot (X, model.predict (X), color = 'red',
linewidth = 2, label = 'Regression line')
plt.xlabel ('year')
plt.ylabel (" Per capita Income (US$)")
plt.title ('canada per capita income prediction')
plt.legend()
plt
plt.show()
```

OUTPUT

Predicted per capita income in 2020 = $ 41288.69

* Multiple linear Regression.

```
From sklearn.preprocessing import LabelEncoder
file_path = "1000_companies.csv"
data = pd.read_csv("/content/1000_companies.csv");
data = fillna(data.mean(numeric_only=True), inplace=True)
label_encoder = LabelEncoder()
data['State'] = label_encoder.fit_transform
(data['State'])

x = data.iloc[:, :-1]
y = data.iloc[:, -1]

model = LinearRegression()
model.fit(X, Y)

test_data = np.array([[91694.48, 15841.3, 11931.2
, label_encoder.transform(['Florida'])[0]]
Predicted_profit = model.predict(test_data)
Print(f"Predicted profit: ${predicted_profit[0]:.2f}
```

OUTPUT

Predicted Profit = $ -2153.01

error

```
y_pred = model.predict(X)

mae = mean_absolute_error(y, y_pred)

mse = mean_squared_error(y, y_pred)
rmse = np.sqrt(mse)

r2 = r2_score(y, y_pred)

Printf(f"Salary prediction (Experience) - Error Metrics")

Printf(f"MAE : {mae:.2f}")

Printf("mse : {mse:.2f}")

Printf("Rmse : {rmse:.2f}")

Printf("R^2 score : {r^2:.4f}")
```

OUTPUT

```
mae : 13202.74
mse : 571205.771.52
RMSE : 23899.91
R^2 score : 0.9424
```

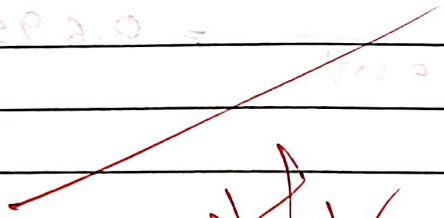
Date __/__/__
Page 11

```
Printf("Slope (Experience): %f", model.coef_[0]);
Printf("Intercept (c): %f", model.intercept_);
```

OUTPUT

Slope = 0.55

Intercept = -70 214.44



AL
11/3/25

LAB-3

Date / / Page 18/3/25

(1) $a_0 = -5$

$a_1 = 0.8$

Logistic regression $p(x) = \frac{1}{1 + e^{-(a_0 + a_1 x)}}$

$= \frac{1}{1 + e^{-(-5 + 0.8x)}}$

(2) $x = 7$

$h_\theta(7) = \frac{1}{1 + e^{-(-5 \times 0.8 \times 7)}} = 0.6957$

3. Threshold of 0.5

$h(x) = 0.6457$

$h(x) \geq 0.5$

Thus $y = 1$ (pass)

$y = \begin{cases} 1 & \text{if } h(x) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$

(4) $z = [2, 1, 0]$ → classes

Softmax $(z_x) = \frac{e^{z_k}}{\sum_{j=1} e^{z_j}}$

Softmax $(z_1) = \frac{e^2}{e^2 + e^1 + e^0} = 0.665$

Date 18/3/25
Page 13

$$\text{Softmax}(z_2) = \frac{e^1}{e^2 + e^1 + e^0} = 0.244$$

$$\text{Softmax}(z_3) = \frac{e^0}{e^2 + e^1 + e^0} = 0.092$$

∴ Probability of 3 classes are

66.8 % 24.4 % 9.1 %

* Logestic Regression Binary.

```
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="+",
color='red')

from sklearn.model_selection import train_test
split.

X_train, X_test, y_train, y_test = train_test_
split(df[['age']], df.bought_insurance,
train_size = 0.9, random_state = 10)
X_train.shape
X_test

from sklearn.linear_model import
logistic Regression
model = logistic Regression()

model.fit(X_train, y_train)
y_predict = model.predict(x_test)
model.score(x_test, y_test)

y_predicted = model.predict([[60]])
y_predicted.
```

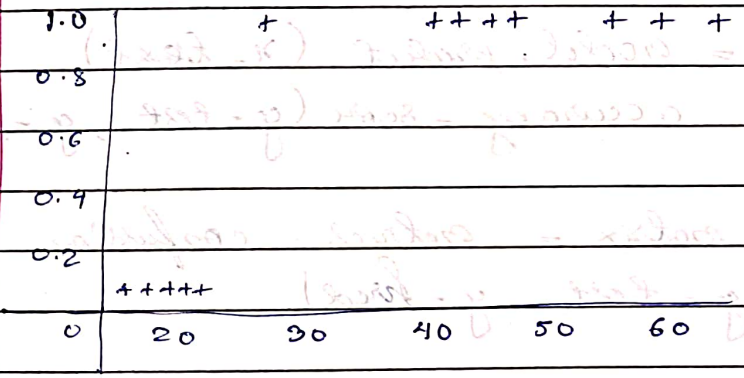
```
import math
def sigmoid(x):
    return 1/(1+math.exp(-x))

def prediction_func(age)
    z = 0.127 * age - 4.973.
    y = sigmoid(z)
    return z

age=35
prediction = func(age)
```

output :-

0.37098347



* Logestic Regression - Multiclass

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import
train_test_split
from sklearn.linear_model import
logestic Regression
```

```
from sklearn.metrics import accuracy_score
from sklearn import metrics.

iris = pd.read_csv("iris.csv")
iris.head()

x = iris.drop('Species', axis='columns')
y = iris.Species.

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

model = LogesticRegression(multiclass="multinomial")

model.fit = model.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)

confussion_matrix = metrics.confussion_matrix(y_test, y_pred)

cm_display = metrics.confussion_matrix_display(confussion_matrix = confussion_matrix, display_labels=["Setosa", "versicolor", "virginica"])
cm = display.plot()
plt.show()
```

Date 18/3/25
Page 17

OUTPUT

Accuracy of multinomial logistic regression model on the test set : 1.00

Setosa	10	0	0
versicolor	0	9	0
virginica	0	0	11

Setosa versicolor virginica.

Predicted label.

Date 1/4/25
Page 18

LAB-4 Decision trees

instance	a_2	a_3	classification
1	Hot	High	No
2	Hot	High	No
6	cool	High	No
7	Hot	High	No
8	Hot	Normal	Yes

$$\text{Entropy}(S) = \frac{-4}{5}\log_2\left(\frac{4}{5}\right) - \frac{1}{5}\log_2\left(\frac{1}{5}\right)$$

$$= 0.7219$$

for a_2 :

$$S_{hot}[1+, 3-] = \frac{1}{4}\log_2\left(\frac{1}{4}\right) - \left(\frac{3}{4}\right)\log_2\left(\frac{3}{4}\right)$$

$$= 0.8113$$

$$S_{cool}[0+, 1-] = 0$$

$$\text{Gain}(S, a_2) = 0.7219 - \frac{4}{5} \times 0.8113 - 0 = 0.7286$$

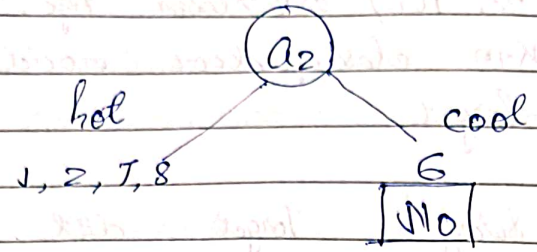
for a_3 :

$$S_{high}[0+, 4-] = 0$$

$$S_{normal}[1+, 0-] = 0$$

$$\text{Gain}(S, a_3) = 0.7219 - 0 - 0 = 0.7219$$

∴ a_2 is the root node since Gain (S, a_2) is high



instance	a_2	a_3	classification.
1	Hot	High	No
2	Hot	High	No
7	Hot	High	No
8	Hot	Normal	Yes

$$\text{Entropy}(S) = -\frac{1}{4}\log_2\left(\frac{1}{4}\right) - \frac{3}{4}\log_2\left(\frac{3}{4}\right)$$

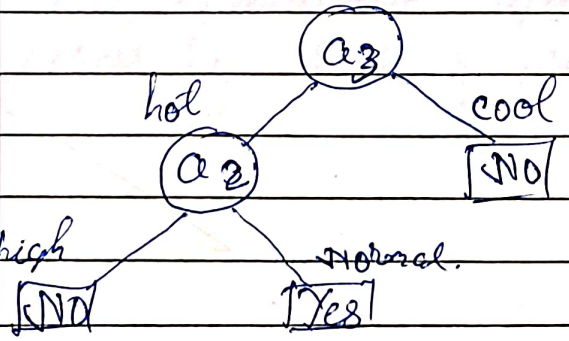
$$= 0.8113$$

for a_3:

$S_{high}[0+, 3-] = 0$

$S_{normal}[1+, 0-] = 0$

$$\text{Gain}(S, a_3) = 0.8113 - 0 - 0 = 8.113$$



Date 1/4/25
Page

LAB - 6

* Consider the following dataset for K=3 and test data (x, 35, 100) as (Person, Age, Salary)x Solving using Knn classification model & predict the target.

Person	Age	Salary K	Target	dist	Rank
A	18	50	N	52.81	5
B	23	55	N	46.57	9
C	24	70	N	31.95	2
D	41	60	Y	40.49	3
E	43	70	Y	31.64	1
F	38	40	Y	60.07	6
X	35	100	?		

K = B.
Rank 1 = Y.
Rank 2 = N
Rank 3 = Y

The target predicted is Y

Date __/__/____
Page ____

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

iris = pd.read_csv("iris.csv")
diabetes = pd.csv("diabetes")
heart = pd.read.csv("heart")

X_iris = iris.iloc[:, :-1]
y_iris = iris.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test
_split(X_iris, y_iris, test_size=0.2,
random_state=42)

Knn = KNeighbourclassifier(n_neighbour=3)
Knn.fit(X_train, y_train)
y_pred = Knn.predict(X_test)


Print("KNN classification for IRIS dataset:")
Print("Accuracy:" accuracy_score(ytest, ypred)
Print("confusion matrix", confusion_matrix(y_test, ypred)
Print("classification Report:\n", classification_
report(y_test, y_pred)
```

Date __/__/__
Page

```
X_diabetes = diabetes.iloc[:, :-1]
y_diabetes = diabetes.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X_diabetes, y_diabetes, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

OUTPUT

KNN classification of Iris data set

Accuracy : 1.0

confusion matrix :

```
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

classification report

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	10
versicolor	1.00	1.00	1.00	9
virginica	1.00	1.00	1.00	11

code 1

Iris.csv

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

iris = pd.read_csv("iris.csv")

x_iris = iris.iloc[:, :-1]
y_iris = iris.iloc[:, :-1]

x_train, x_test, y_train, y_test = split(x_iris
y_iris, test_size = 0.2, random_state(42)

dtc = DecisionTreeClassifier()
dtc.fit(x_train, y_train)
y_pred = dtc.predict(x_test)

print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred)
```

OUTPUT

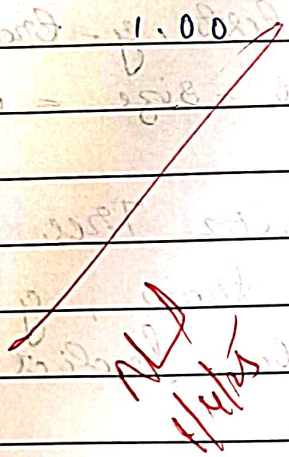
Accuracy : 1.0

confusion matrix :

```
[[10  0  0
 [ 0  9  0
 [ 0  0 11]]
```

classification Report:

	Precision	recall	f1-scor.	Suppo
Setosa	1.00	1.00	1.00	10
versicolor	1.00	1.00	1.00	9
virginica	1.00	1.00	1.00	11
accuracy	1		1.00	30
macro avg.	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

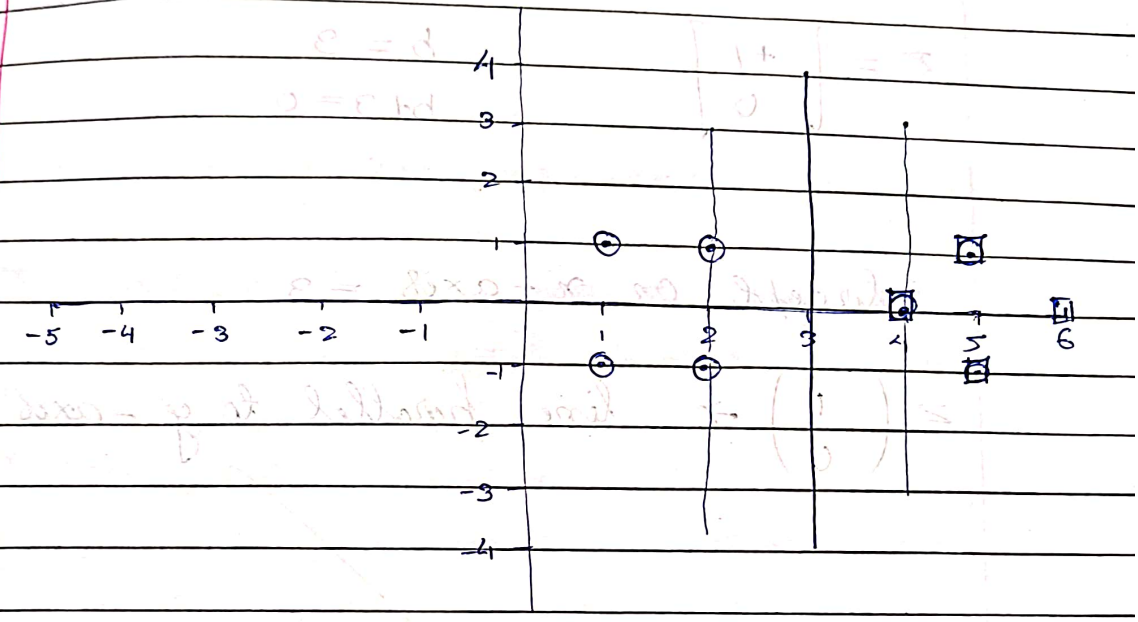


Date __/__/__
Page ____ __

* SVM

Draw an optimal hyperplane using linear svm to classify the following points

{(1,1), (2,1), (1,-1), (2,-1)} - +ve labelled
{(4,0), (5,1), (5,-1), (6,0)} - -ve labelled.



$$S_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad S_2 = \begin{bmatrix} 2 \\ -1 \end{bmatrix} \quad S_3 = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$$

$$\tilde{S}_1 = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \quad \tilde{S}_2 = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} \quad \tilde{S}_3 = \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix}$$

$$\alpha_1 \tilde{S}_1 \tilde{S}_1 + \alpha_2 \tilde{S}_2 \tilde{S}_1 + \alpha_3 \tilde{S}_3 \tilde{S}_1 = +1$$
$$\alpha_1 \tilde{S}_1 \tilde{S}_2 + \alpha_2 \tilde{S}_2 \tilde{S}_2 + \alpha_3 \tilde{S}_3 \tilde{S}_2 = +1$$
$$\alpha_1 \tilde{S}_1 \tilde{S}_3 + \alpha_2 \tilde{S}_2 \tilde{S}_3 + \alpha_3 \tilde{S}_3 \tilde{S}_3 = -1$$

$$\alpha_1(6) + \alpha_2(4) + \alpha_3(9) = +1$$
$$\alpha_1(4) + \alpha_2(6) + \alpha_3(4) = +1$$
$$\alpha_1(9) + \alpha_2(9) + \alpha_3(17) = -1$$

$$\alpha_1 = 13/4$$
$$\alpha_2 = 13/4$$
$$\alpha_3 = -7/2$$

Date __/__/__
Page

$$w = \alpha_1 \tilde{s_1} + \alpha_2 \tilde{s_2} + \alpha_3 \tilde{s_2}$$

$$= 3/4 \begin{bmatrix} 2 \\ 1 \end{bmatrix} + 13/4 \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} + -7/2 \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 3 \end{bmatrix} = -\begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}$$

$$z = \begin{bmatrix} +1 \\ 0 \end{bmatrix} \qquad b = 3$$

$$b + 3 = 0$$

intercept on x-axis = 3

$z \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ → line parallel to y-axis

8/4/2

Date __/__/__
Page _____

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score,
confusion_matrix.


iris_df = pd.read_csv("/content/iris (1) (3).csv")

X = iris_df.drop("Species", axis=1)
y = LabelEncoder().fit_transform(iris_df["Species"]

X_train, X_test, y_train, y_test = train_test_spli
(X, y, test_size=0.2, random_state=42)

Svm_linear = SVC(kernel='linear')
Svm_linear.fit(X_train, y_train)
y_pred_linear = Svm_linear.predict(X_test

print("linear Kernel:")
print("Accuracy:", accuracy_score(y_test, y_
_linear))
print("confusion Matrix:\n", confusion_matrix(y_tes
, y_pred_linear))

Svm_rbf = SVC(kernel='rbf')
Svm_rbf = Svm_rbf.predict(X_test)
```

```
Print ("\nRBF Kernel:")
Print("Accuracy:" accuracy_score(y_test,
y_pred_rbf))

Print("confusion matrix:\n", confusion_matrix
(y_test, y_pred_rbf))
```

OUTPUT

linear Kernel:

Accuracy : 1.0

confusion Matrix :-

```
[[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]]
```

RBF Kernel:

Accuracy : 1.0

confusion Matrix:

```
[[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]]
```

Date __/__/____
Page _____

Q. what is the accuracy score of the classifier using the linear kernel on RBF kernel.

Ans linear Kernel Accuracy = ~ 0.98

RBF Kernel Accuracy = ~ 1.00

Q. which kernel gave better performance on the IRIS dataset

Ans. RBF Kernel.

LAB-8

Date 15/4/21
Page

1. Difference B/w Decision Tree And Random Forest

Feature	Decision Tree	Random Forest
Definition	A single tree-based model that splits data based on features	An ensemble of decision tr (usually trained with bootstrapping and features randomness)
Bias-variance	Low bias, high-variance (overfits easily)	low variance and better generalization
Accuracy	Less accurate than RF for large datasets	Generally more accurate due to ensemble learni
overfitting	Prone to overfitting on noisy data	less prone due to averaging across multiple trees.
Interpretability	Easy to interpret	Harder to interpret
Training Speed	Faster (single model)	Slower. (multiple mod
Prediction Speed	Faster (one tree)	Slower (aggregates prediction from many trees)

Date __/__/__
Page _____

* Parameters of Random Forest classifier()

(1) n_estimators
(2) criterion
(3) max_depth
(4) min_sample_split
(5) min_weight_fraction_leaf.
(6) min_sample_leaf.
(7) max_features.
(8) max_leaf_nodes
(9) min_impurity_decrease
(10) bootstrap.
(11) oob_score.
(12) n_jobs
(13) random_state
(14) verbose
(15) class_weight
(16) warm_start

* Algorithm of Random Forest.

(1) Input: Dataset with features and target

2. For i=1 to n_estimators:-

- Draw a bootstrap sample from training set
- Train a Decision Tree on this sample

Date __/__/__
Page

- At each node, select a random subset of features.
- use the best split among the selected features.
- Grow the tree to maximum depth.

3. Aggregate the predictions:

- For classification : use majority voting from all trees.
- For regression : use average prediction from all trees.

4. OUTPUT :- Final Prediction.

Date __/__/__
Page .

code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv("/content/train-train.csv")

Print("columns:\n", df.columns)

label_encoders = {}
for column in df.columns:
  if df[column].dtype == 'object':
  le = LabelEncoder()
  df[column] = le.fit_transform(df[column].astype
  (str))
  label_encoders[column] = le

X = df.iloc[:, :-1]
y = df.iloc[:, -1]

X_train, X_test, y_train, y_test = train_
test_split(X, y, test size = 0.2, random
state = 42)

rf_model = RandomForestClassifier(n_estimators = 100,
random_state = 42)
rf_model.fit(X_train, y_train)

y_pred = rf_model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

Print(f"\n Accuracy Score: {accuracy:.4f}")
Print("\n Confusion Matrix:")
Print(conf_matrix)
```

OUTPUT

```
Columns :-
Index(['PassengerId', 'Survived', 'Pclass',
       'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin',
       'Embarked'], dtype = object

Accuracy Score : 0.8659

Confusion Matrix :-

[[[26   0  17]
  [ 0  12   5]
  [ 2   0 117]]]
```

LAB-10 : K means.

Date __/__/__
Page _____

(i) write an algorithm of K means?
(ii) choose K initial centroids
(iii) Assign each point to the nearest centroid.
(iii) Recalculate centroids.
(iv) Repeat until convergence.

(2) How to determine no. of clusters?
- Elbow method:
- Silhouse the method:

Sum of Squared errors (SSE):

$$SSE = \sum_{i=1}^{K} \sum_{x_j \in C_i} (x_j - \mu_i)^2$$

where c_i = cluster
μ_i = centroid.

(3) what is the formula of Sum of Squared errors? Plot SSE vs no. of cluster

(4) Describe elbows tehnique.
= Plot SSE against K. The elbow is there SSE start decreasing at a slower rate. This is the optimal K.

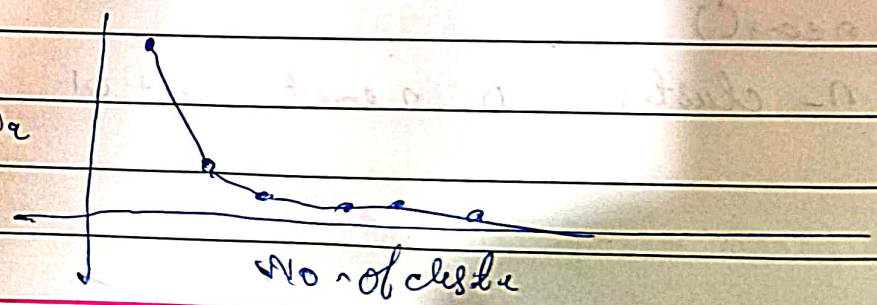
(5) Discuss all the parameters used in Kmeans()
=> n_cluster, n_init, max_iter

```
import pandas as pd
import matplotlib.pyplot as plt
From sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

iris = load_iris()
data = pd.DataFrame(iris.data, columns = iris.feature_
X = data[['Petal length (cm)', 'Petal width (cm)']

Scaler = StandardScaler()
X_Scaled = Scaler.fit_transform(X)
inertia = []
K_range = range(1,11)
For K is in K_range:
KMeans = KMeans(n_cluster = K, random_state = 42)
KMeans.Fit(X_Scaled)
inertia.append(KMeans.inertia_)

Plt.figure(fig size(8,5))
Plt.Plot(K_range, inertia, marker = 'o')
Plt.title("elbow method for optimal K")
Plt.xlabel("Number of clusters (K)")
Plt.ylabel("inertia)
Plt.grid(True)
Plt.Show()
```



Date __/__/__
Page _____

LAB-11 :- PCA

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score.

digits = load_digits()
X = digits.data
y = digits.target

Scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
Pca = PCA( n_components =2)
X_Pca = PCA.fit_transform (X_scaled)

X_train, X_test, y_train, y_test = train_test_split(
(X_PCA, y, test_size =0.2, random_state=42)

model = logistic Regression()
model.fit ( X_train, y_train)

y_pred = model.Predict (X_test)
accuracy = accuracy_score (y_test, y_pred)
Score = model.score (X_test, y_test)

Print ("model Score (accuracy using score:")
                    round (score, y))

Print ("Accuracy using PCA with 2 compont, roun
```

⇒ OUTPUT

Model Score :- 0.5389

Accuracy :- 0.5319