classmate
Date 4-3-25
Page 1

Lab 4:

(I) Dataset: housing.csv

```
import pandas as pd
```

(i) Load csv file:

```
df1=pd.read_csv('housing.csv')
```

(ii) Display all columns:

```
df1.columns
```

(iii) ~~Statistical~~

```
Index(['longitude', 'latitude', 'housing_median_age',
       'total_rooms', ... ], dtype='object')
```

(iii) Statistical information:

```
df1.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bed
count	20640	20640	20640	20640	20640

	population	households	median_income	median_value
cant	20640	20640	20640	20640

(iv) Count of unique labels for "Ocean proximity" column:

```
cnt=len(df1['ocean_proximity'].unique())
print(cnt)
```

⇒ 5

(v) display which ~~attributes~~ attributes having missing values cant greater than 0.

```
print(df1.columns[df1.isnull().any()])
```

⇒ Index(['total_bedroom'], dtype='object')

(II) Dataset: diabetes.csv

```
df2 = pd.read_csv('diabetes.csv')
```

(i) which columns have missing values:

```
print(df2.columns[df2.isnull().any()])
```

⇒ Index([], dtype='object')

(ii) Categorical columns:

```
cat1 = df2.select_dtypes(exclude=['Number']).columns
print(cat1)
```

⇒ Index(['Gender', 'CLASS'], dtype='object')

(III) Dataset = adult.csv

```
df3 = pd.read_csv('adult.csv')
```

(i) Missing values:

```
print(df3.columns[df3.isnull().any()])
```

⇒ Index([], dtype='object')

(ii) Categorical columns:

```
cat2 = df3.select_dtypes(exclude=['number']).columns
cat2
```

⇒ Index(['workclass', 'education', 'marital-status', 'occupation',
'relationship', 'race', 'gender', 'native-country', 'income'],
dtype='object')

Lab-1:

```
tickers = ["HDFCBANK.NS", "ICICIBANK.NS", "KOTAKBANK.NS"]
data = yf.download(tickers, start="2024-01-01", end="2024-12-30",
group_by='ticker')
import yfinance as yf import pandas as pd import matplotlib.
pyplot as plt
hdfc_data = data['HDFCBANK.NS']
hdfc_data['Daily Return'] = hdfc_data['Close'].pct_change()
print("\n Daily Returns for HDFC Industries:")
print(hdfc_data['Daily Return'].head(10))
plt.figure(figsize=(12,6))
plt.subplot(2,1,1)
hdfc_data['Close'].plot(title='HDFC Industries - Closing Price')
plt.subplot(2,1,2)
hdfc_data['Daily Return'].plot(title="HDFC ",
color="orange")
plt.tight_layout()
plt.show()
```

→ Daily Returns to HDFC Industries:

Date	Data
2024-01-01	NAN
2024-01-02	0.000584
2024-01-03	-0.015420
2024-01-04	0.018770
2024-01-05	-0.005116

```
import pandas as pd

1. data = {'USN': [1,2,3,4,5], 'Name': ["A","B","C","D",
"E"], 'Marks': [60,20,70,40,50]}
df = pd.DataFrame(data)
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	West

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

classmate
Date 14-3-25
Page 5

Lab-4:

Linear Regression and Multiple Linear Regression:

Find linear regression of data of week and profit.

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

$$\beta = ((X^T X)^{-1} X^T) y$$

$$X^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix} \qquad X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}$$

$$y^T = \begin{bmatrix} 2 & 4 & 5 & 9 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}$$

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

$$(X^T X)^{-1} = \begin{bmatrix} 1.2115 & -0.192 \\ -0.192 & 0.0384 \end{bmatrix}$$

$$((X^T X)^{-1}) X^T = \begin{bmatrix} 1.2115 & -0.192 \\ -0.192 & 0.0384 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 4 & 5 & 9 \end{bmatrix}$$

$2 \times 2 \qquad 2 \times 4$

$$= \begin{bmatrix} 0.8271 & 0.4435 & 0.2515 & -0.5165 \\ -0.1152 & -0.0384 & 0 & 0.1536 \end{bmatrix}$$

$(X^T X)^{-1} X^T Y$

$$= \begin{bmatrix} 0.825 & 0.4425 & 0.2575 & -0.3625 \\ -0.1152 & -0.0384 & 0 & 0.1536 \end{bmatrix}_{2\times 4} \begin{bmatrix} 2 \\ 4 \\ 5 \\ 9 \end{bmatrix}_{4\times 1}$$

$$= \begin{bmatrix} 0.038 \\ 0.984 \end{bmatrix}$$

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

$$(X^T X)^{-1} X^T = \begin{bmatrix} 1 & 0.5 & 0 & -0.5 \\ -0.3 & -0.1 & 0.1 & 0.3 \end{bmatrix}$$

(2)

$$2Y = 2\begin{bmatrix} 2 \\ 4 \\ 5 \\ 9 \end{bmatrix} = \begin{bmatrix} -0.5 & 2.2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$$

$Y = \beta_0 + \beta_1 X$

Q1. Predict Canada's per capita income in year 2020.

```
import pandas as pd
from sklearn import linear_model

df = pd.read_csv('canada_per_capita_income.csv')

df.columns = ['year', 'per_capita_income']
X = df[['year']]
y = df['per_capita_income']

reg = linear_model.LinearRegression()
reg.fit(X, y)
predicted = reg.predict([[2020]])
print(predicted)
```

⇒ 41288.694

Q2. Predict salary of employee with 12 years of experience. (salary.csv)

⇒ 139574.041

Q3. Predict salaries for following candidates:

1) 2 year experience, 9 test score, 6 interview score
2) 12 year experience, 10 test score, 10 interview score

using dataset hiring.csv

```
from word2number import w2n
import pandas as pd
from sklearn import linear_model

df = pd.read_csv('hiring.csv')

df.experience = df.experience.fillna("zero")
df.experience = df.experience.apply(lambda x: w2n.word_to_num(x))

df['test_score(out of 10)'] = df['test_score'].fillna(
        df['test_score'].median())

reg = linear_model.LinearRegression()
reg.fit(df[['experience','test_score','interview_score']],
        df['salary'])

print(reg.predict([[2,9,6]]))
print(reg.predict([[12,10,10]]))
```

⇒ [53205.967]
[92002.18]

Q4. Predict profit using 1000_companies.csv for following.
91694.48, R&D spend, 515841.3 Admin,
119321.24 Marketing spend, Florida state

⇒ [51017.346]

Lab-5: Logistic Regression

Q1) Binary Classification (Sigmoid Func^n)

$a_0 = -5,\ a_1 = 0.8$

Linear eq^n $\Rightarrow$ $z = a_0 + a_1 x$

$= -5 + 0.8x$

$p(x)$ for $x = 7$

$y = p(x) = \frac{1}{1+e^{-(a_0+a_1x)}} = \frac{1}{1+e^{-(-5+0.8(7))}} = 0.64$

$0.64 \geq 0.5 \Rightarrow 1$ pass

∴ Student who studies for 7 hrs will pass.
The predicted class is pass.

Q2. Multiclass Regression (Softmax func^n)

$z = [2, 1, 0]$ ∴ find probability values of the 3 classes.

$$\text{softmax}(z_k) = \frac{e^{z_k}}{\sum_{j=1}^{3} e^{z_j}}$$

$$\text{Softmax}(z_1) = \frac{e^2}{e^2+e^1+e^0} = 0.665 = 66.5\%$$

$$\text{Softmax}(z_2) = \frac{e^1}{e^2+e^1+e^0} = 0.244 = 24.4\%$$

$$\text{Softmax}(z_3) = \frac{e^0}{e^2+e^1+e^0} = 0.091 = 9.1\%$$

Lab 5
18/3

1) insurance_data.csv :

```
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 LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
X_test
y_test
y_predicted = model.predict(X_test)
y_predicted
model.score(X_test, y_test)
model.predict_proba(X_test)
y_predicted = model.predict([[60]])
y_predicted
model.coef_
model.intercept_
import math
def sigmoid(x):
    return 1 / (1+math.exp(-x))
def prediction_function(x):
    z = 0.042 * age - 1.523
    y = sigmoid(z)
    return y
age = 35
```

prediction_function(age)

⇒ 0.3709839769552725

2) iris.csv:

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn import metrics
import matplotlib.pyplot as plt

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 = LogisticRegression(multi_class='multinomial')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

⇒ 1.00

1. HR Comma_sep. csv :

(i) Salary has a direct impact on employee retention because as per the data, employees who have less salary have a lower retention rate and vice versa.

(ii) The accuracy of our ML model is 79.63%. The accuracy is decent :(neither good nor bad).

2. Zoo dataset :

(i) I have dropped the columns 'animal_name' and 'class_type' as they had categorical values.

(ii) There weren't any missing values in dataset.

(iii) The confusion matrix tell us that there are higher number of true positive values.

(iv) Reptile class has been misclassified as fish class.

Done
19-5-23

Lab - 03

Instance	a2	a3	Classification
1	Hot	High	No
2	Hot	High	No
6	Cool	High	No
7	Hot	High	No
8	Hot	Normal	Yes

Attribute: a2

values(a2) = Hot, Cool

$S_{a2} = [1+, 4-]$ $\quad Entropy(S_{a2}) = -\frac{1}{5}\log_2(\frac{1}{5}) - \frac{4}{5}\log_2(\frac{4}{5})$

$S_{Hot} = [1+, 3-]$ $\quad Entropy(S_{Hot}) = -\frac{1}{4}\log_2(\frac{1}{4}) - \frac{3}{4}\log_2(\frac{3}{4}) = 0.8112$

$S_{Cool} = [0+, 1-]$ $\quad Entropy(S_{Cool}) = 0.0$

$Gain(S, a2) = Entropy(S) - \sum_{v \in (Hot, Cool)} \frac{|S_v|}{|S|} Entropy(S_v)$

$= 0.9709 - \frac{4}{5}(0.8112) - (\frac{1}{5})0 = 0.3219$

Attribute: a3

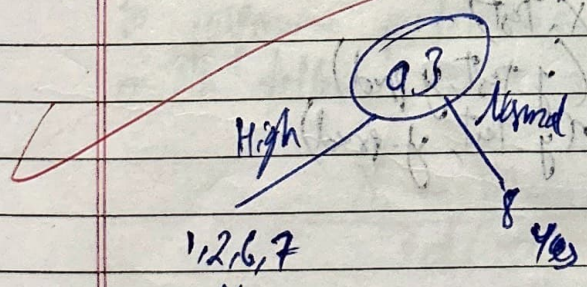
values(a3) = High, Normal

$S_{all} = 0.7219$

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

$S_{Normal} = 0$

$Gain(S, a3) = 0.9709 - (\frac{4}{5})0 - (\frac{1}{5})0 = 0.7219$



1. The accuracy achieved is 1.0 (100%). Confusion matrix will show that all values alay diagonal and zeroes elsewhere. There are no misclassifications.

2. We need to visualize the tree, analyze the nodes and identify important features. In decision tree classifier discrete class labels are predicted for each instance. In regression tree continuous value for each instance. It aims to create split that minimize the variance of target variable.

Petrol_consumption.csv:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, mean_squared
df = pd.read_csv('petrol_consumption.csv')
X = df.drop('Petrol_Consumption', axis=1)
y = df['Petrol_Consumption']
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size=0.2, random_state=42)
regressor = DecisionTreeRegressor()
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mse ** 0.5
print(mae, mse, rmse)
```

output:

MAE: 95.6
MSE: 17837.8
RMSE: 133.5357

Lab-6:

Q. Consider the following dataset. for K=3 and test data: (x, 35, 100) as (Person, Age, Salary K) . solve using KNN classifier model & predict the ~~target~~ target.

Person	Age	Salary K	Target	dist.	Rank
A	18	150	N	82.81	5
B	~~20~~ 23	~~50~~ 55	N	96.57	9
C	24	70	N	31.95	2
D	41	60	Y	40.49	3
E	43	70	Y	31.89	1
F	38	40	Y	60.79	6
X	35	100	?		

K=3

Rank 1 = Y , Rank 2 = N , Rank 3 = Y

The target predicted is Y.

Q.1. To find value of K, perform iteration with range of K value. Perform evaluation for each K and finally select the one with highest accuracy and lowest rate.

Q2. Feature Scaling is used for standardization and to improvise performance. To perform standard scaler we the following formula:

$$\frac{X_i - \text{mean}(x)}{sd(x)}$$

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.datasets import load_iris 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=0.2,
                                                    random_state=42)
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
print(accuracy)
print(conf_matrix)
print(class_report)
```

Output:

Accuracy : 1.0

Confusion Matrix :
$$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 11 \end{bmatrix}$$

Classification Report:

	Precision	recall	f1-score	support
0	1.00	1	1	10
1	1.00	1	1	9
2	1.00	1	1	11

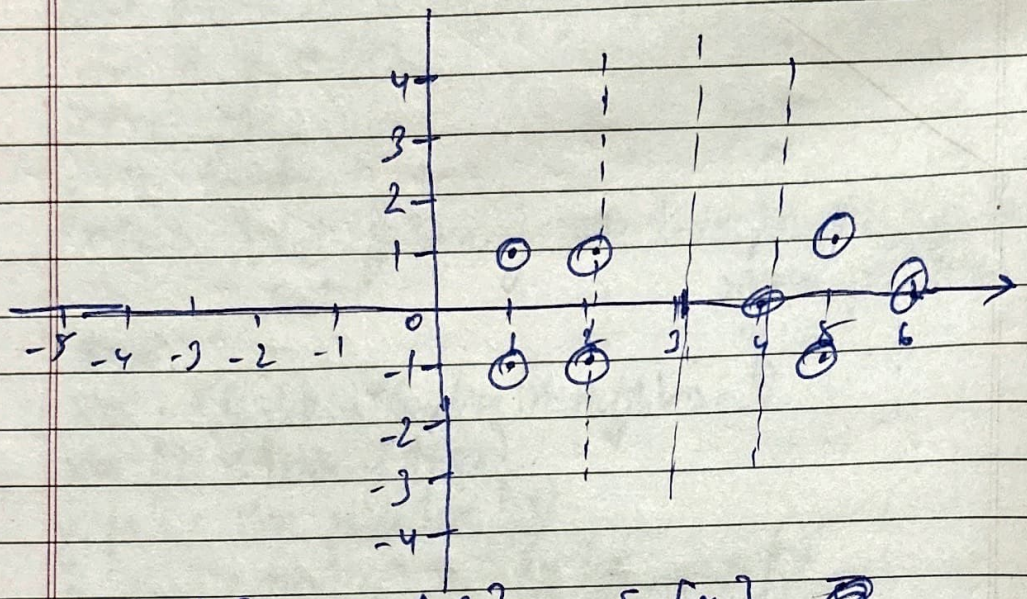
		Precision	recall	f1-score	Support
accuracy	= 1.0				
macro avg		1	1	1	30
weighted avg		1	1	1	30

classmate
Date 8-4-25
Page 18

Lab-7:

1. Draw an optimal hyperplane using linear SVM to classify the following points:

(1,1) (2,1) (1,-1) (2,-1) (4,0) (5,1) (5,-1) (6,0)



$s_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ $s_2 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$ $s_3 = \begin{bmatrix} 4 \\ 0 \end{bmatrix}$

$\bar{s}_1 = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$ $\bar{s}_2 = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}$ $\bar{s}_3 = \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix}$

$\alpha_1 \bar{s}_1 \bar{s}_1 + \alpha_2 \bar{s}_2 \bar{s}_1 + \alpha_3 \bar{s}_3 \bar{s}_1 = +1$
$\alpha_1 \bar{s}_1 \bar{s}_2 + \alpha_2 \bar{s}_2 \bar{s}_2 + \alpha_3 \bar{s}_3 \bar{s}_2 = +1$
$\alpha_1 \bar{s}_1 \bar{s}_3 + \alpha_2 \bar{s}_2 \bar{s}_3 + \alpha_3 \bar{s}_3 \bar{s}_3 = -1$

$\alpha_1(6) + \alpha_2(9) + \alpha_3(9) = +1$ $\alpha_1 = 13/4$
$\alpha_1(9) + \alpha_2(6) + \alpha_3(9) = +1$ $\alpha_2 = 13/4$
$\alpha_1(9) + \alpha_2(9) + \alpha_3(17) = -1$ $\alpha_3 = -7/2$

$w = \alpha_1 \bar{s}_1 + \alpha_2 \bar{s}_2 + \alpha_3 \bar{s}_3$

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

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

$b + 3 = 0$

intercept on x-axis = 3

$2\begin{pmatrix} 1 \\ 0 \end{pmatrix} \Rightarrow$ line parallel to y-axis

iris.csv:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

iris = pd.read_csv('iris.csv')
X = iris.drop('species', axis=1)
y = iris['species']
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size=0.2, random_state=42)
svm_rbf = OneVsRestClassifier(SVC(kernel='rbf', random_state=42,
                                  probability=True))
svm_rbf.fit(X_train, y_train)
y_pred = svm_rbf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True)
fpr = {}
tpr = {}
roc_auc = {}
for i in range(26):
    fpr[i], tpr[i], _ = roc_curve(y_test_bitor[:, i],
                                  y_pred_prob[:, i])
```

roc_auc[i] = auc(fpr[i], tpr[i])

for i in range(26):
    plt.plot(fpr[i], tpr[i])

fpr i i

Kernels = ['linear', 'rbf']
for K in Kernels:
    svm_classifier = SVC(K=Kernel)
    svm_classifier.fit(X_train, y_train)
    y_pred = svm_classifier.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(K, ":", accuracy)
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True)
    plt.show()

Accuracy of SVM classifier with linear Kernel: 1.0
Confusion Matrix:

$$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 11 \end{bmatrix}$$

Accuracy of SVM classifier with rbf Kernel: 1.0
Confusion matrix:

$$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 11 \end{bmatrix}$$

Q1. Accuracy of svm classifier with linear Kernel : 1.0
with rbf Kernel : 1.0

Since both Kernels perform equally well, using the one with simplicity and computational efficiency could be better choice. In this case, linear Kernel would be a more practical choice due to its simplicity and efficiency.

Q2. The confusion matrix shows the model's prediction breakdown highlighting true positives on the diagonal and misclassifications. The AUC score measures model performance in binary classification. For multi-class, AUC is calculated for each class against the others. AUC of 1 indicates perfect performance. The combination of accuracy and AUC provides insight into the models ability to differentiate b/w classes. The letter-recognition dataset is more complex than the iris dataset, so SVM performance is expected to be lower.

classmate
Date 15-9-25
Page 22

Lab 8 - Random Forest:

Decision Tree	Random Forest
→ A decision tree is a tree like model of decisions along with possible outcomes in a diagram.	→ A classification algorithm consisting of many decisions combined to a single tree.
→ There is always a scope for overfitting.	→ Random forest avoids and prevents overfitting.
→ Results are not accurate	→ Gives accurate and precise results.
→ Require low computation.	→ Require more computation
→ Easy to visualize	→ Complex visualization

Parameters of random forest:

→ Max features: Max. no. of features RF is allowed to try in an individual tree. Features auto/None, sqrt, 0.2

→ n-estimators: Number of trees you want to build before taking max. voting or averaging of prediction.

→ min-sample-leaf:

→ n-jobs: no. of processors allowed.

→ random_state:

→ cv: cross-validation method

Algorithm:

1) Select random samples from a given data or training set.

2) The algorithm will construct a decision tree for every training data.

3) Voting will take place by averaging the decision tree.

4) Finally, elect the most voted prediction result as the final prediction result.

Accuracy with 10 n-estimators = 10 = 1.0
" with 50 " = 50 = 1.0
" 100 " = 1.0

classmate
Date 29-4-25
Page 24

Lab-11:

PCA:

1. Calculate mean
2. Calculation of covariance matrix
3. Eigen value of covariance matrix
4. Computation of the eigen vectors - unit eigen vectors
5. Computation of first principal components
6. Geometrical meaning of first principal components.

Q. Given the table, reduce dimension from 2 to 1 using PCA:

Feature	Eg 1	Eg 2	Eg 3	Eg 4
x_1	4	8	13	7
x_2	11	4	5	14

Mean:

1) $\bar{x}_1 = \frac{1}{4}(4+8+13+7) = 8$

$\bar{x}_2 = \frac{1}{4}(11+4+5+14) = 8.5$

2) Covariance matrix:

$$Cov(x_1, x_2) = \frac{1}{N-1} \sum_{k=1}^{N} (x_{1k} - \bar{x}_1)^2$$

$$= \frac{1}{3}\left[(4-8)^2 + (8-8)^2 + (13-8)^2 + (7-8)^2\right]$$

$$= 14$$

$$Cov(x_1, x_2) = \frac{1}{N-1} \sum_{k=1}^{N} (x_{2k} - \bar{x}_2)^2$$

$$= \frac{1}{3}\left((4-8)(11-8.5) + (8-8)(4-8.5) + (13-8)(5-8.5) + (7-8)(14-8.5)\right]$$

$$= -11$$

$$cov(X_2, X_1) = cov(X_1, X_2)$$
$$= -11$$

$$cov(X_2, X_2) = \frac{1}{N-1} \sum_{k=1}^{N} (X_{2k} - \bar{X}_2)^2$$

$$= \frac{1}{3}\left[(11-8.5)^2 + (4-8.5)^2 + (5-8.5)^2 + (14-8.5)^2\right]$$

$$= 23$$

2) $$S = \begin{bmatrix} cov(X_1, X_1) & cov(X_1, X_2) \\ cov(X_2, X_1) & cov(X_2, X_2) \end{bmatrix} = \begin{bmatrix} 14 & -11 \\ -11 & 23 \end{bmatrix}$$

3) $$\det(S - \lambda I) = \begin{bmatrix} 14-\lambda & -11 \\ -11 & 23-\lambda \end{bmatrix}$$

$$= (14-\lambda)(23-\lambda) - 11^2$$

$$= \lambda^2 - 37\lambda + 201$$

$$\lambda = \frac{1}{2}(37 \pm \sqrt{565}) = 30.3849,\ 6.6151$$

$$= \lambda_1, \lambda_2$$

4) $$U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = (S - \lambda I) X$$

$$= \begin{bmatrix} 14-\lambda_1 & -11 \\ -11 & 23-\lambda_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$= \begin{bmatrix} (14-\lambda_1)u_1 - 11u_2 \\ -11u_1 + (23-\lambda_1)u_2 \end{bmatrix} = 0$$

$$(14-\lambda_1)u_1 - 11u_2 = 0$$
$$-11u_1 + (23-\lambda_1)u_2 = 0$$

$$\frac{u_1}{11} = \frac{u_2}{14-\lambda_1} = t_1$$

$$u_1 = 11t$$
$$u_2 = (14-\lambda_1)t$$

$$v_1 = \begin{bmatrix} 11 \\ 14-\lambda_1 \end{bmatrix}$$

$$\|v_1\| = \sqrt{11^2 + (14-\lambda)^2} = \sqrt{11^2 + (14-30.7849)^2}$$
$$= 19.7348$$

$$e_1 = \begin{bmatrix} 11/\|v_1\| \\ (14-\lambda_1)/\|v_1\| \end{bmatrix} = \begin{bmatrix} 0.5574 \\ -0.8303 \end{bmatrix}$$

$$e_2 = \begin{bmatrix} 0.8303 \\ 0.5574 \end{bmatrix}$$

5) $\begin{bmatrix} x_{1K} \\ x_{2K} \end{bmatrix}$

$$e_1^T \begin{bmatrix} x_{1K} - \bar{x}_1 \\ x_{2K} - \bar{x}_2 \end{bmatrix} = \begin{bmatrix} 0.5574 & -0.8303 \end{bmatrix} \begin{bmatrix} x_{1K} - \bar{x}_1 \\ x_{2K} - \bar{x}_2 \end{bmatrix}$$

$$= 0.5574(x_{1K} - \bar{x}_1) - 0.8303(x_{2K} - \bar{x}_2)$$

Code for Randomforest:

```
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
X, y = iris.data, iris.target
y_binarized = label_binarize(y, classes=[0,1,2])
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size=0.2, random_state=42)
n_estimators_list = [10, 20, 30]
all_scores = []
rf_model_default = RandomForestClassifier(n_estimators=10, random_state
                                          =42)
~~n_estimators_default = RandomForestClassifier()~~
rf_model_default.fit(X_train, y_train)
y_pred_default = rf_model_default(X_test)
accuracy_default = accuracy_score(y_test, y_pred_default)
print(accuracy_default)
best_accuracy = 0
best_n_estimators = 10
for n_estimators in [10, 50, 100, 200, 500]
        rf_model = RandomForestClassifier(n_estimators, random_state=42)
        rf_model.fit(X_train, y_train)
        y_pred = rf_model.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        if accuracy > best_accuracy:
                best_accuracy = accuracy
                best_n_estimators = n_estimators
print(best_accuracy)
```

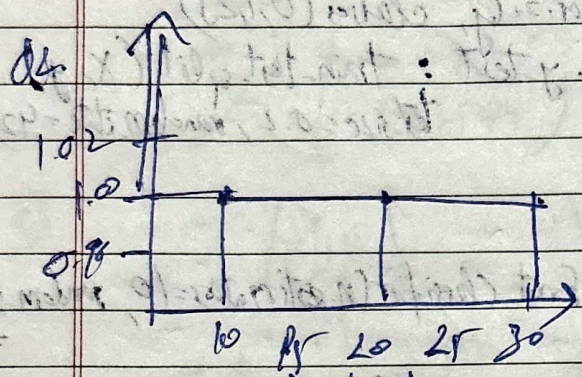
~~Output:~~

Output:

Accuracy with default = 100%
Best accuracy = 100%

Q1. Accuracy remains same for n = 10, 20, 30
accuracy = 1.00

Q2.



classmate
Date 15-9-25
Page 29

Lab-9 : Boosting:

Q1. Boosting is an ensemble learning technique that combines multiple weak learners to create a long strong learner. It works by sequentially training models, where each new model focuses on mistakes made by the previous ones. The models' predictions are weighted based on their performance.

Q2. Parameters of AdaBoost Classifiers.

~~→ Total estimators (int)~~

→ base_estimator (object, default = None) = base model to be used for boosting.

→ n_estimators (int) - no of boosting rounds

→ learning_rate - scaling factor

→ algorithm ({'SAMME', 'SAMME.R'}, default = 'SAMME.R')

→ random_state - controls randomness

→ loss ({'linear', 'square', 'exponential'}) - loss function

Q3. Algorithm:

1) Initialize: set the weights of all training samples equally.
2) For each initialization:
   - Train a base learner on weighted training set
   - Calculate error of the model on training data.
   - Compute model weight based on error.
   - Update sample weights: increase weight of misclassified samples
3) Combine: The final model is the weighted sum of the base learners, where learners with lower errors have higher weights.

classmate
Date
Page 30

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
data = pd.read_csv('income.csv')
X = data.drop(columns=['income_level'])
y = data['income_level']
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size=0.2, random_state=42)
ada_model = fit(X_train, y_train)
y_pred = ada_model(y_test, y_pred) (X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
print(accuracy * 100)
print(conf_matrix)
```

Output:

Accuracy: 87.27%

Confusion matrix:

$$\begin{bmatrix} 7003 & 411 \\ 1223 & 1192 \end{bmatrix}$$

classmate
Date 24-4-25
Page 71

Lab-10 : K-Means:

1. Algorithm:

1) Initialize: Randomly select K points as initial cluster centroids.
2) Assign: Assign each data point to the closest centroid.
3) Update: Compute new centroids as the ~~means~~ mean of the points in each cluster.
4) Repeate step 2 and 3 until centroids do not change.

2. How to determine no. of clusters (K)

1) Elbow Method
2) Silhoutte Analysis
3) Gap statistic
4) Domain Knowledge or business requirements

3. SSE =

$$\sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2$$

C_i = cluster i
μ_i = centroid of cluster i
x = data point in cluster C_i
$\|x - \mu_i\|^2$ = squared euclidean distance

4. We run K-means for a range of K (1 to 10)

5.

1) Plot SSE vs K
2) Observe the point where SSE dramatically drops and levels off
3) It suggests diminishing returns by adding more clusters.

6. Parameters:

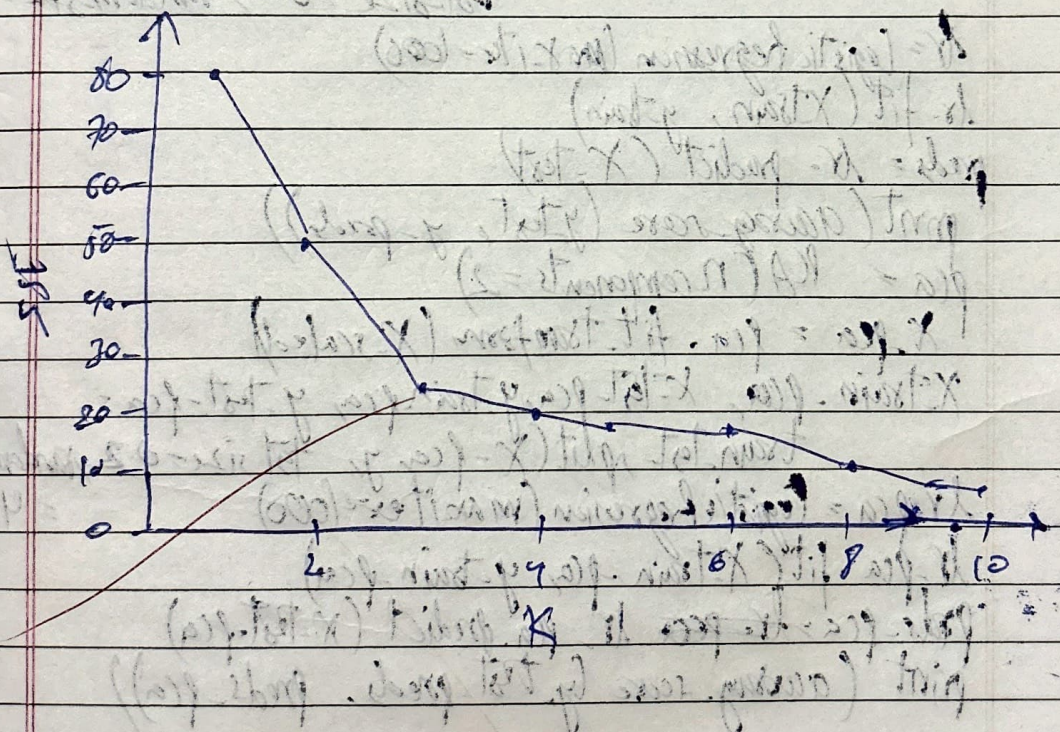
- n_clusters: no of clusters (K)
- init - method to initialize centroids
- n_init - times algo is run
- max_iter - max iteration per run
- tol - convergence tolerance
- algorithm - auto, full
- random_state - seed for reproducibility.

Code:

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
import random
import matplotlib.pyplot as plt

np.random.seed(42)
Names = [f'Person_{i}' for i in range(1, 51)]
ages = np.random.randint(20, 60, size=50)
incomes = np.random.randint(20000, 120000, size=50)
income_df = pd.DataFrame({
        'Name': Names,
        'Age': ages,
        'Income': incomes})
df = pd.read_csv('income.csv')
df_numeric = df.drop('Name', axis=1)
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_numeric)
X_train, X_test, y_train, y_test (df_scaled, test_size=0.2,
                                  random_state=42)
```

```
sse = []
K_range = range(1,11)
for K in K_range:
    Km = KMeans(n_clusters = K, random_state = 42)
    Km.fit(X_train)
    sse.append(Km.inertia_)
plt.plot(K_range, sse, marker = 'o')
plt.show()
optimal_K = 3
Kmeans = KMeans(n_clusters = optimal_K, random_state = 42)
Kmeans.fit(X_train)
y_pred = Kmeans.predict(X_test)
print(y_pred)
```



Code for PCA:

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA

digits = load_digits()
X, y = digits.data, digits.target
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
                                   test_size=0.2, random_state=42)
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
preds = lr.predict(X_test)
print(accuracy_score(y_test, y_preds))
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
X_train_pca, X_test_pca, y_train_pca, y_test_pca =
        train_test_split(X_pca, y, test_size=0.2, random_state
                                                    =42)
lr_pca = LogisticRegression(max_iter=1000)
lr_pca.fit(X_train_pca, y_train_pca)
preds_pca = lr_pca.predict(X_test_pca)
print(accuracy_score(y_test, preds, preds_pca))
```

Accuracy without pca = 0.9722
Accuracy with pca (2 components) = 0.5389