LABORATORY CONTINUOUS EVALUATION			
S.No.	PROGRAM	Max.Marks	Marks Scored
1.	Preparedness	5	
2.	Coding	5	
3.	Testing	5	
4.	Viva	5	
	TOTAL	20	

Faculty Signature with Date

Experiment-05.

Dimensionality reduction and Impact Analysis.

problem statement:- Evaluate the affect of dimensionality reduction on classification accuracy.

Dataset:- winequality(Red+white)

[import pandas as pd
import]

Loading the Dataset:-

```
from google.colab import files
uploaded = files.upload().
```

Choosing file:- wine quality.csv.

Reading the file :-

```
import pandas as pd.
df = pd.read_csv('wine quality.csv')
print(df).
```

Output:-

	fixed acidity	volatile acidity -----	quality	Id
0	7.4	0.700	5	0
1	7.8	0.880	5	1
2	7.8	0.760	5	2
3	11.2	0.280 ----------	6	3
⋮	⋮	⋮	⋮	⋮
1142	5.9	0.645	5	1597

checking null values:

```
print(df.isnull().sum()).
```

output:-

fixed acidity	0
volatile acidity	0
citric acid	0
residual sugar.	0
chlorides.	0
free sulphor dioxide.	0
total sulphor dioxide.	0
density	0
pH	0
sulphates.	0
alcohol	0
quality	0
Id.	0

Register No. :

Experiment No. :

Date :

LABORATORY CONTINUOUS EVALUATION			
S.No.	PROGRAM	Max.Marks	Marks Obtained
1.	Preparedness	5	
2.	Coding	5	
3.	Testing	5	
4.	Viva	5	
	TOTAL	20	

Faculty Signature with Date

Load and preprocess dataset

```
import pandas as pd.
from sklearn.model_selection
    import train_test_split.
from sklearn.preprocessing import
    standardscalar.
df = pd.read_csv("wine equality:csv")
df = df.drop(columns=["Id"]).
x = df.drop("quality", axis=1)
y = df["quality"]
print("Datashape:", df.shape)
print("class distribution:\n", y.value_counts()).
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.2, random_state=42,
    stratefy=y).
scalar = Standardscalar()
x_train_scaled = scalar.fit_transform(x_train)
x_test_scaled = scalar.transform(x_test).
```

Output:-

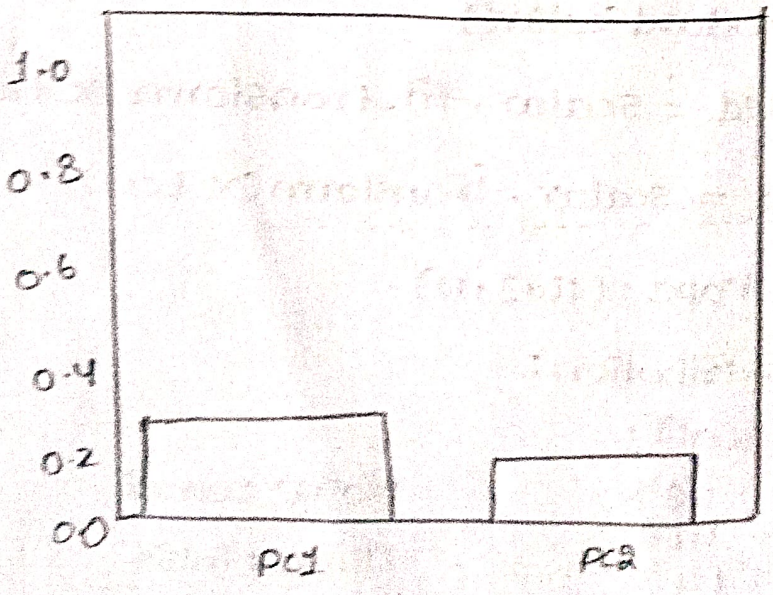
```
Dataset shape : (1143, 12)
class distribution:
    quality.
5    483
6    462
7    143
4     33
8     16
3      6
Name : count
dtype : int64
```

(1) Apply pca and LDA for reducing dimensionality.

code: (PCA)

```
import matplotlib.pyplot as plt.
from sklearn.decomposition import PCA.
pca = PCA(n_components =2)
x_train_pca =pca.fit_transform (x_train_scaled)
x_test_pca = pca.fit.transform (x_test_scaled).
plt.figure (figsize =(6,4))
plt.bar([1,2] , pca.explained_variance_ratio_, tick_labe
    ["PC1","PC2"] , color = ["#1f776b","#ff7f0e"])63
plt. title ('PCA - variance explained")
plt. ylabel ('Explained variance Ratio')
plt. xlabel ("principle components")
plt.ylim (0,1)
plt.show().
```

PCA variance explained.



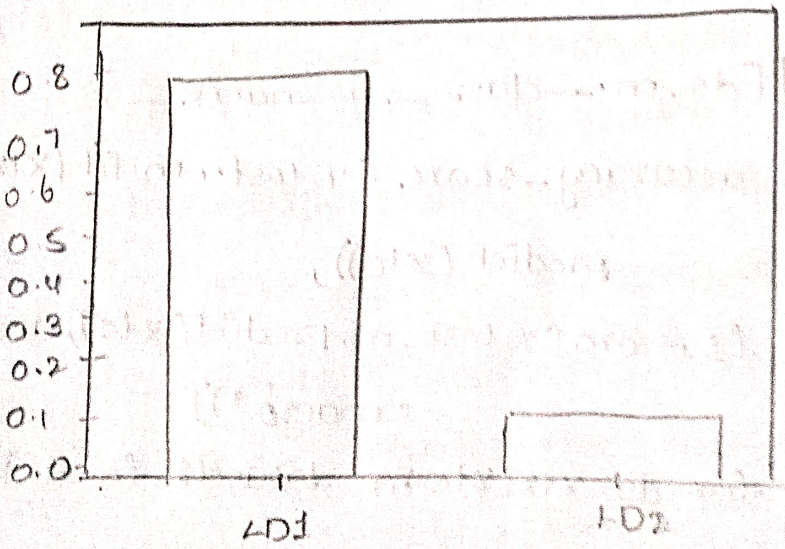
Register No. : Experiment No. : Date :

LABORATORY CONTINUOUS EVALUATION			
S.No.	PROGRAM	Max.Marks	Marks Scored
1.	Preparedness	5	
2.	Coding	5	
3.	Testing	5	
4.	Viva	5	
	TOTAL	20	

Faculty Signature with Date

code (LDA).

```
lda = LDA (n_components = 2)
lda = fit(X_train_scaled, y_train)
explained_var_ratio = lda.explained_
                      variance_ratio_)*
plt.figure (figsize = (6,4)).
plt.bar(["LD1","LD2"], explained_var_ratio, color =
        ["#1f77b4","#ff7f0e"])
plt.title("LDA - explained variance Ratio")
plt.ylabel(" proportion of variance ")
plt.show()
print(" explained variance ratio (LD1, LD2):", explained_var_
                                                ratio).
```



explained variance ratio (LD1, LD2): [0.832755 , 0.08835].

2) (Testing) Train Logistic regression and Decision-tree on :

- original dataset
- PCA - reduced dataset
- LDA - reduced dataset

code :

```
from sklearn . linear_model import LogisticRegression.
from sklearn.tree import DecisionTreeclassifier.
from sklearn.metrics import accuracy_score, f1_score.
models = [LogisticRegression (max_iter = 2000,
          random_state = 42),
          DecisionTreeclassifier (random_state = 42)]
datasets = [('original", X_train_scaled, X_test_scaled)
            ("PCA", X_train_pca, X_test_pca),
            ("LDA", X_train_lda, X_test_lda)].

results = [[ds, m.__class__.__name__,
            accuracy_score (y_test, m.fit (Xtr, y_train)
                predict (Xte)),
            f1_score (y_test, m.predict(Xte), average=
                "macro")]
            for ds, Xtr, Xte in datasets for m in models]

for r in results:
    print(r).
```

output :

```
[original', LogisticRegression, 0.6244, 0.2847)
[original', "DecisionTreeclassifier, 0.633, 0.3501]
```

LABORATORY CONTINUOUS EVALUATION			
S.No.	PROGRAM	Max.Marks	Marks Scored
1.	Preparedness	5	
2.	Coding	5	
3.	Testing	5	
4.	Viva	5	
	TOTAL	20	

Faculty Signature with Date

[PCA, LogisticRegression, 0.550, 0.2013]
[PCA, DecisionTreeclassifier, 0.55, 0.26]
[LDA, LogisticRegression, 0.628, 0.2818]
[LDA, DecisionTreeclassifier, 0.585, 0.301]

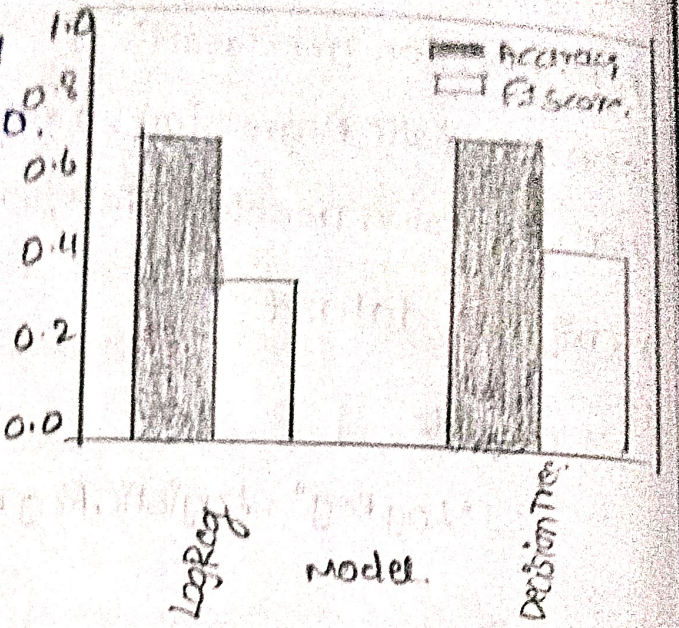
(i) original dataset:

```
models = {
    "LogReg": LogisticRegression(max_iter=2000, random_
                                 state=42),
    "DecisionTree": DecisionTreeclassifier(random_state=42)
}
results = []
for name, model in models.items():
    ypred = model.fit(x_train_scaled, y_train).predict
                                 (x_test_scaled)
    results.append([name, accuracy_score(y_test, y_pred),
        f1_score(y_score, y_pred, average="macro")])
    sns.heatmap(confusion_matrix(y_test, y_pred),
                annot=True, fmt="d", cmap="Blues")
    plt.title(f"matrix={name}"); plt.show().
df = pd.Dataframe(results, columns=["model", "Accuracy",
                                    "F1 score"])
print(df)
df.set_index("Model")[["Accuracy", "F1 score"]].plot(kind="bar",
    ylim=(0,1), figsize=(8,4), title="originalDataset") plt.show().
```

Output:

Original Dataset

	model	accuracy	fscore
0	LogReg	0.824	0.084
1	DecisionTree	0.833	0.850



PCA Reduced Dataset.

```
code: results = []
    for name, model in models.items():
        model.fit(x_train_pca, y_train)
        ypred = model.predict(x_test_pca).
        results.append([name, accuracy_score(y_test,
            y_pred), f1_score(y_test, ypred, average="macro")])
df = pd.DataFrame(results, columns = ["model", "Accuracy", "F1 score"])
print(df)
df.set_index("model")[["Accuracy", "F1 score"]].plot(kind=
    "bar", ylim = (0,1), figsize = (6,4), title = "PCA Dataset")
plt.show().
```

Decision Boundaries.

```
def plot_db(x, y, model, title):
    xx, yy = np.meshgrid(np.linspace(X[:, 0].min()-1,
        X[:,0].max()+1, 200), np.linspace(X[:, 1].min()-1,
        X[:,1].max()+1, 200)).
```

Register No. : | Experiment No. : | Date :

LABORATORY CONTINUOUS EVALUATION			
S.No.	PROGRAM	Max Marks	Marks Scored
1.	Preparedness	5	
2.	Coding	5	
3.	Testing	5	
4.	Viva	5	
	TOTAL	20	

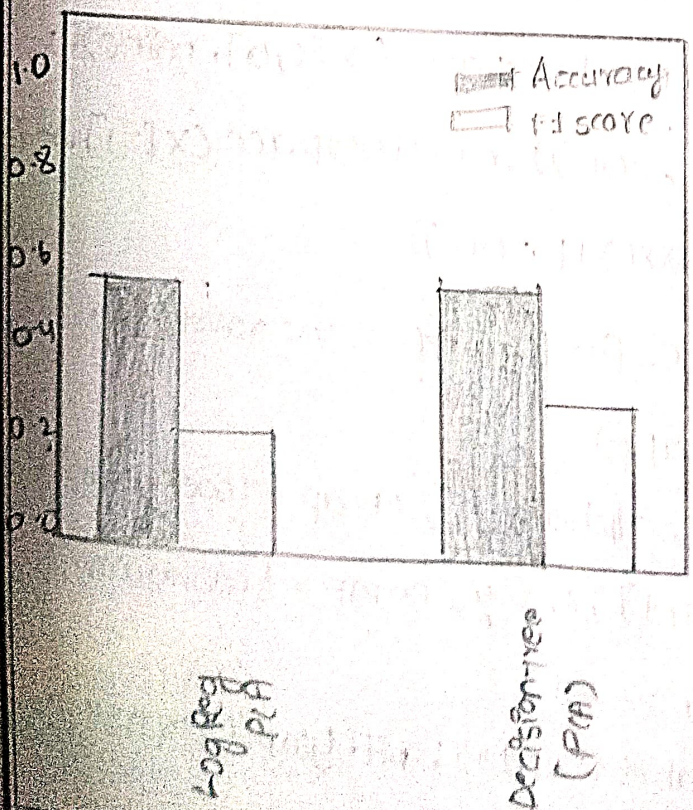
Faculty Signature with Date

```
z = model.predict(np.c_[xx.travel(), yy.travel()]).reshape(xx.shape)
plt.scatter(X[:,0], X[:,1], c=y, cmap="coolwarm", edgecolor="k", s=20)
plt.title(title); plt.xlabel("PC1"); plt.ylabel("PC2"); plt.show()
for name, model in models.items():
    plot_db(X_train_pca, y_train, model, f"PCA - {name}")
```

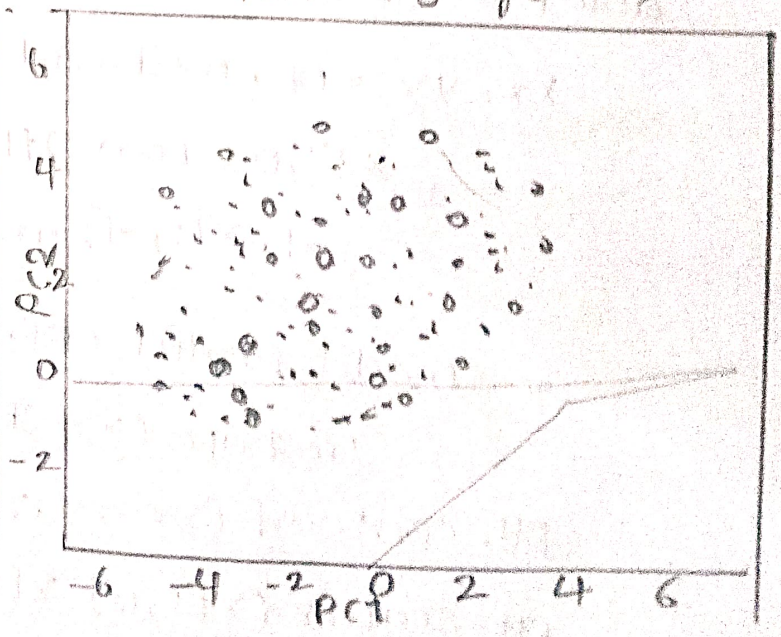
Output:

	model	Accuracy	F1 score
0	LogReg(PCA)	0.550	0.201
1	DecisionTree(PCA)	0.550	0.261

PCA Dataset -



PCA - LogReg (PCA)



LDA Reduced Dataset

```
results=[]
for name, model in models.items():
    model.fit(X_train_lda, y_train)
    y_pred = model.predict(X_test_lda)
    results.append([name, accuracy_score(y_test, y_pred),
        f1_score(y_test, ypred, average="macro")])
df = pd.DataFrame(results, columns=["model", "Accuracy",
    "f1 score"])
print(df)
df.set_index("model")[["Accuracy", "f1 score"]].plot(
    kind="bar", ylim=(0,1), figsize=(6,4), title=
    "LDA Dataset").
plt.show()
def Decision Boundary.
def plot_db(X, y, model, title):
    xx, yy = np.meshgrid(np.linespace(X[:,0].min()-1,
        X[:,0].max()+1, 200)), np.linespace(X[:,1].min
        ()-1, X[:,1].max()+1, 200)).
    z = model.predict(np.c_[xx.ravel(), yy.ravel()]).
        reshape(xx.shape)
    plt.contourf(xx, yy, z, alpha=0.3, cmap="coolwarm")
    plt.scatter(X[:,0], X[:,1], c=y, cmap="coolwarm",
        edgecolor="k", s=20)
    plt.title(title); plt.xlabel("LD1"); plt.ylabel("LD2");
```

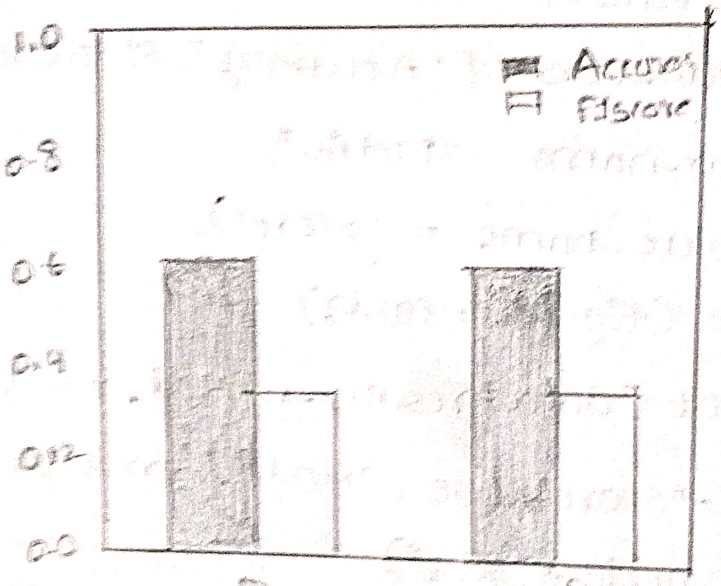
Register No. : Experiment No. : Date :

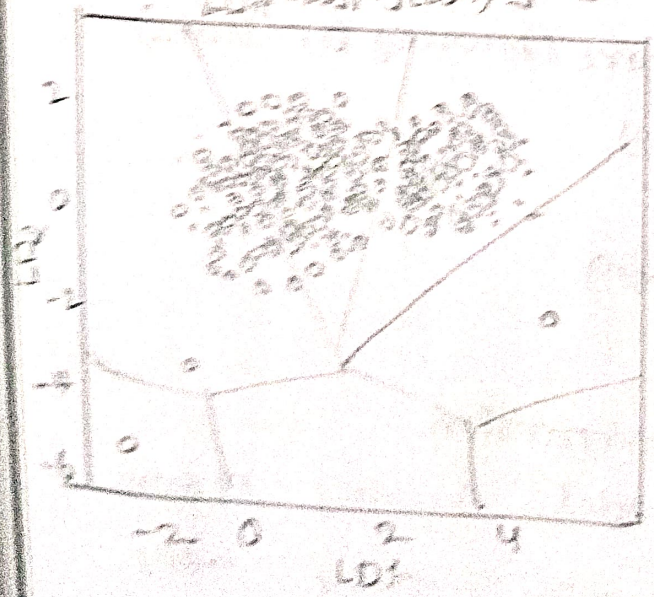
LABORATORY CONTINUOUS EVALUATION			
S.No.	PROGRAM	Max.Marks	Marks Scored
1.	Preparedness	5	
2.	Coding	5	
3.	Testing	5	
4.	Viva	5	
	TOTAL	20	
			Faculty Signature with Date

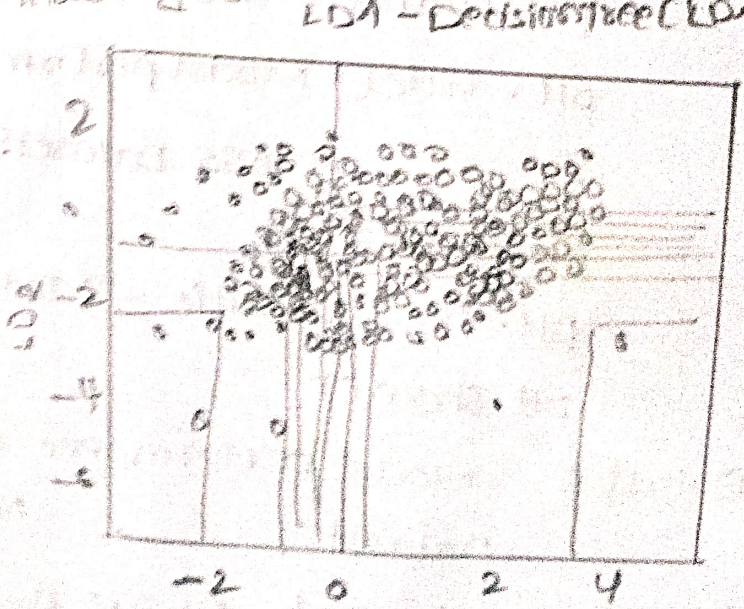
```
plt.show()
for name, model in models.items():
    plot_db(X_train_lda, y_train,
            model, f"LDA - {name}").
```

output :

	model	Accuracy	f1 score
0	LogReg (LDA)	0.628	0.287
1	DecisionTree(LDA)	0.585	0.3018







Model performance comparision

```
import seaborn as sns.
results_df = pd.DataFrame(results, columns=
        ["Datasets", "model", "Accuracy", "F1 score"])
print("\n Model performance comparision".)
print(results_df).
results_melted = results_df.melt(
        id_vars = ["Dataset", "Model"],
        value_vars = ["Accuracy", "F1 score"],
        var_name = "Metric",
        value_name = "score").
plt.figure(figsize = (10,6)).
sns.barplot(data = results_melted, x = "Dataset",
        y = "score", hue = "Model", errorbar=none,
        palette = "set2").
plt.ylabel("score")
plt.title("Model performance (Accuracy & F1 score)
        Across Datasets").
plt.ylim(0,1)
plt.legend(title = "Model")
plt.show().
```

Output :- Model performance comparision.

	Dataset	Model	Accuracy	F1 score.
0	original	Logistic Regression	0.624	0284.

Register No. : Experiment No. : Date :

LABORATORY CONTINUOUS EVALUATION			
S.No.	PROGRAM	Max.Marks	Marks Scored
1.	Preparedness	5	
2.	Coding	5	
3.	Testing	5	
4.	Viva	5	
	TOTAL	20	
			Faculty Signature with Date

1. Original DecisionTreecla 0.633 0.350
2. PCA LogisticRegressin 0.55 0.20
3. PCA DecisionTreecla 0.55 0.26
4. LDA LogisticRegression 0.62 0.28
5. LDA DecisionTreeclass 0.58 0.30.

6 Model performance (Accuracy & F1 score) Across Datasets