Register No. : 231FA04651 Experiment No. : 4 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

Experiment-4

Feature selection on a Breast Cancer Dataset

Dataset :- Breast Cancer wisconsin Data set -

problem Statement :- select the most informative features to predict cancer diagnosis.

Preprocessing :-

Loading the Data.

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

file : breast cancer.csv.

importing the data.

```
import pandas as pd
df = pd.read_csv('breast cancer.csv')
print(df).
```

output :

	id	diagnosis	radius_mean	----	fractal_dimen
0	842302	M	17.99		0.11890
1	842517	M	20.57		0.08902
2	84300903	M	19.69		0.08758
:	:	:	:	----	:
568	92751	B	7.76		0.07039.

Missing values :-

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

id	— 0
diagnosis	— 0
radius_mean	— 0
texture_mean	— 0
perimeter_mean	— 0
area_mean	— 0
smoothness_mean	— 0
compactness_mean	— 0
concavity_mean	— 0
concave points_mean	— 0
symmetry_mean	— 0
fractal_dimension_mean	— 0
radius_se	— 0
texture_se	— 0
perimeter_se	— 0
area_se	— 0
smoothness_se	— 0
compactness_se	— 0
concavity_se	— 0
concave points_se	— 0
symmetry_se	— 0
fractional_dimension_se	— 0
radius_worst	— 0

Register No. : Experiment No. : Date :

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

Faculty Signature with Date

texture_worst - 0
perimeter_worst - 0
area worst - 0
smoothness_worst - 0
compactness_worst - 0
concavity_worst - 0
concave points_worst - 0
symmetry_worst - 0
fractional_dimension_worst - 0

Label encoding:-

```
from sklearn.preprocessing import LabelEncoder.
label_encoder = LabelEncoder()
df['diagnosis'] = label_encoder.fit_transform
            (df["diagnosis"])
print(df).
```

output:-

	id	diagnosis	radius_mean	...	worst-fractional_dimension
0	842302	1	17.99		0.11890
1	842517	1	20.57		0.08902
2	84300903	1	19.69		0.08758
⋮	⋮	⋮	⋮	...	⋮
568	92751	0	7.76		0.07039.

Tasks:-

(1) Apply FilterMethod : chi-square test.

Code :-
```
from sklearn.feature_selection import selectKBest, chi2
from sklearn.preprocessing import MinMaxScalar.
Scalar = MinMaxScalar()
X_scaled = scalar.fit_transform(X)
chi2_selector = SelectKBest(chi2, k=10)
X_chi2 = chi2_selector.fit_transform(X_scaled, y)
Selected_chi2 = X.columns[chi2_Selector.get_support()].
                tolist()
print("chi-square Selected features :", Selected_chi2)
```

Output:-

chi-square selected Features :

['mean radius', 'mean perimeter', 'mean area', 'mean concavity',
'mean concave points', 'worst radius', 'worst perimeter',
'worst area', 'worst concavity', 'worst concave points'].

X, Y, X_scaled missed one code -Google colab

(2) Apply wrapper Method : Forward and Backward selection

Code:-
```
from mlxtend.feature_selection import Sequential
                 Feature selector as SFS
from sklearn.Linear_model import Logistic Regression
lr = LogisticRegression(max_iter = 500, solver = 'lib
                                       linear').
```

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

```
sfs_forward = SFS(lr, k_features=10,
          forward = True,
          floating = False,
          scoring = 'accuracy',
          cv = 5).
sfs_forward = sfs_forward.fit(x_scaled ; y)
print ("Forward selection Features:\n", list(x.columns[list(
          sfs.forward.k_features_idx)]]))

sfs_backward = SFS (lr, K_features = 10, Forward = False,
          floating = False, scoring = 'accuracy', cv=5)
sfs_backward = sfs_backward.fit (x_Scaled , y)
print (" Backward selection Features :\n", List(x.columns[
          list (sfs_backward.k_feature_idx_)]]).
```

Output: Forward selection Features:-

['id', 'smoothness_mean', 'concativity_mean', 'Symmetry_mean', 'smoothness_se', 'concavity_se', 'fractal_dimension_se', 'texture_worst', 'perimeter_worst', 'smoothness_worst']

Backward selection Features:-

['concavity_mean', 'concave points_mean', 'radius_se', 'texture_se', 'symmetry_se', 'fractional_dimension_se', 'texture_worst', 'area_worst', 'smoothness_worst', 'symmetry_worst'].

(3) Apply Embeded Method: Elastic Net Regularization.

code:-

```
from sklearn.preprocessing import StandardScalar
from sklearn.Linear-model import LogisticRegression
scalar = Standardscalar()
x_scaled_std = scalar.fit_transform(X)
elastic_net = LogisticRegressionCV(
        Cs =10,
        cv= 5,
        penalty= "elasticnet",
        solver = "saga",
        l1_ratios = [0.5],
    max_iter = 5000,
        Scoring= "accuracy"
    )
elastic_net.fit(x_scaled_std, y)
coef = np.mean (abs(elastic_net.coef_), axis = 0)
selected_embeded = X.columns [coef > np.percentile
                (coef, 75)].tolist()
print ("Elastic Net Selected Features:", selected_
                embedded)
```

output:- Elastic Net selected Features:

['concave points_mean', 'radius_se', 'radius_worst', 'texture_worst', 'perimeter_worst', 'area-worst', 'smoothness_worst', 'concave. points_worst'].

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH Department of

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

(4) Evaluate model performance with and without feature selection using Logistic Regression.

```
from sklearn.model_selection import
        cross_val_score.
def evaluate_model (x,y, model):
    Scores = cross_val_score (model, x, y, cv=5, scoring="
                accuracy")
    return scores.mean().
    [result]
result[]
results.append (["Full Feature Set", "All", evaluate_model
            (x, y, base_model)])
results.append (["Filter (chi2)", selected_chi2, evaluate_mo
        -del (x[selected_chi2], y, base_model)])
results.append (["wrapper (forward)", selected_forward,
        evaluate_model (x['selected_forward'], y,
            base_model)])
results.append (["wrapper (Backend)", selected_backward,
    evaluate_model (x[selected_backward], y, base
    model)])
results.append (["embeded (elastic Net)", selected_embedded,
    evaluate_model (x[selected_embedded], y,
```

base_model)])

results_df = pd.DataFrame(results, columns = ["Method", "Selected Features", "Accuracy"]).

print(result_df)

Output:-

	Method	Selected Features	Accuracy
0	Full Feature set	All	
1	Filter (chi2)	[mean radius, mean perimeter, mean ar	
2	wrapper (Forward)	[mean smoothness, mean compactness	
3	wrapper (Backward)	[mean radius, mean texture	
4	Embedded (ElasticNet)	[mean concave points, radius	

cross

	Accuracy
0	0.950815
1	0.943782
2	0.952554
3	0.956078
4	0.945552

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,5))
plt.bar(results_df["Method"], results_df["Accuracy"]
        , color="skyblue")
plt.ylabel("Accuracy")
plt.title("Logistic Regression performance with
```

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

```
Feature Selection")
plt.ylabel ("Accuracy")
plt.title ("Logistic Regression
  performance with Feature selection")
plt.xticks (rotation=20, ha='right')
for i,v in enumerate(results_df["Accuracy"]):
    plt.text (i,v+0.005, f"{v:.3f}", ha="center"
              fontweight="bold")

plt.right_layout()
plt.show().
```