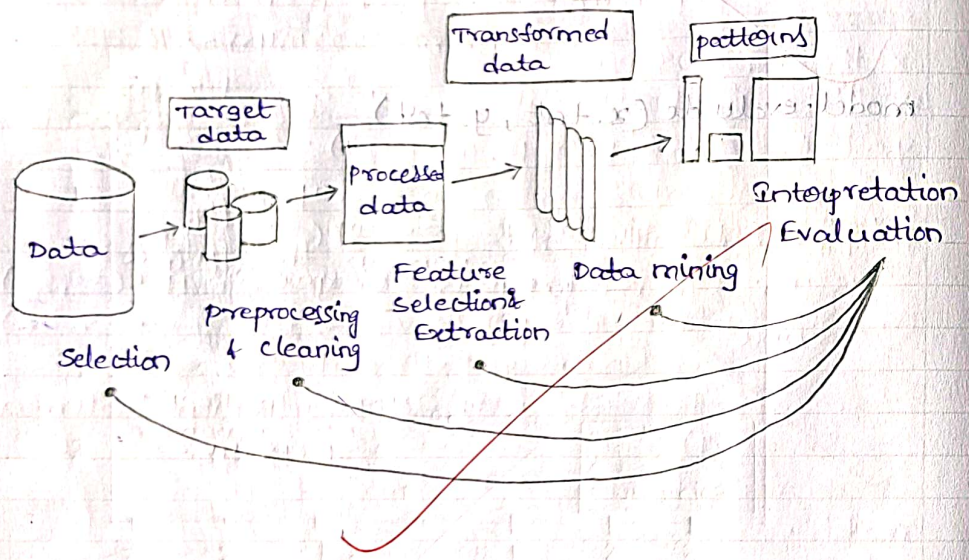


BVC
GROUP OF INSTITUTIONS
ODALAREVU · AMALAPURAM · RAJAHMUNDRY

Experiment - 11

Aim: Write a program to implement principal component Analysis.

Description:

principle component Analysis (PCA):

→ PCA is an unsupervised learning algorithm that is used for the dimensionality reduction in Machine learning.

→ It is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformation.

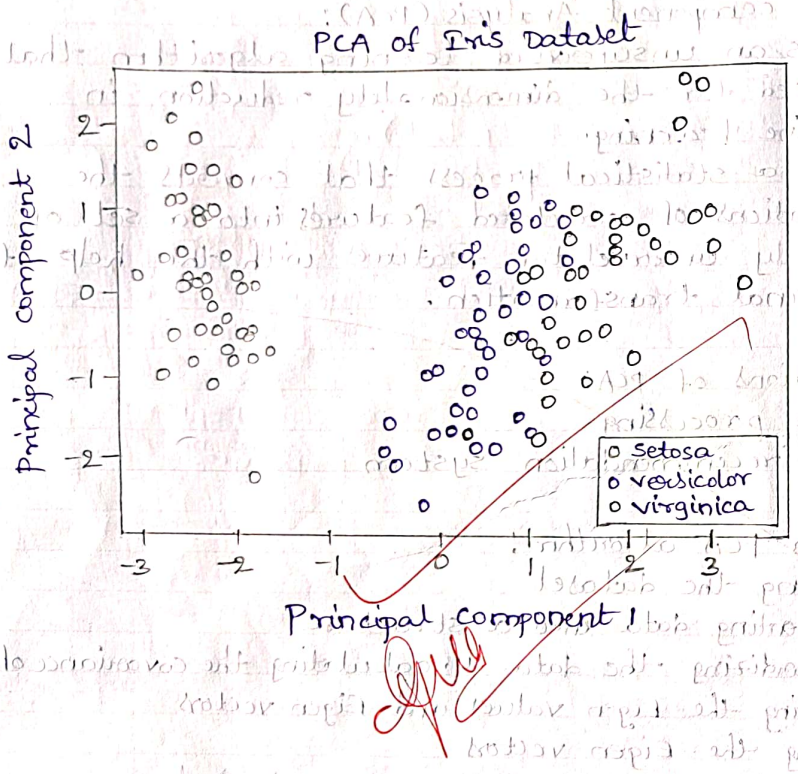
Applications of PCA:

→ Image processing

→ Movie recommendation system

steps for PCA algorithm:

1. Getting the dataset
2. Representing data into a structure
3. Standardizing the data 4. calculating the covariance of Z.
5. calculating the Eigen values and Eigen vectors
6. Sorting the Eigen vectors
7. Calculating the new features or principal components.
8. Remove less or unimportant features from the new dataset.



Program:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler

data = load_iris()
X = data.data
y = data.target

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(8,6))
for target in np.unique(y):
    indices = y == target
    plt.scatter(X_pca[indices,0], X_pca[indices,1], label=
data.target_names[target])
plt.xlabel('Principal component 1')
plt.ylabel('Principal component 2')
plt.title('PCA of Iris Dataset')
plt.legend()
plt.grid()
plt.show()
```

BVC GROUP OF INSTITUTIONS
ODALAREVU • AMALAPURAM • RAJAHMUNDRY

Experiment - 12

Aim: Exploratory Data Analysis for classification using pandas or matplotlib.

Description:
EDA: It is an important step in data analysis process. It involves training and visualizing the data to gain insights, identify patterns and understanding the characteristics of datasets. EDA helps in discovering between variables, detecting outliers and preparing the data for further analysis or modeling.
→ EDA often uses statistical graphics and other data visualization methods. It includes barcharts, line charts, piecharts, map charts.

Process:
→ understand the data
→ Determine which variable are important
→ prepare your data
→ Improve your machine learning models
→ Determine if statistical techniques are appropriate

program:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Describe:

	total_bill	tip	size
count	244.00	244.00	244.00
mean	19.7	2.99	2.56
std	8.9	1.00	1.00
min	13.34	2.0	2.0
25%	13.34	2.9	2.0
50%	17.79	3.5	3.0
75%	24.12	10.0	6.0
max	50.8	210.0	6.0

Missing values:

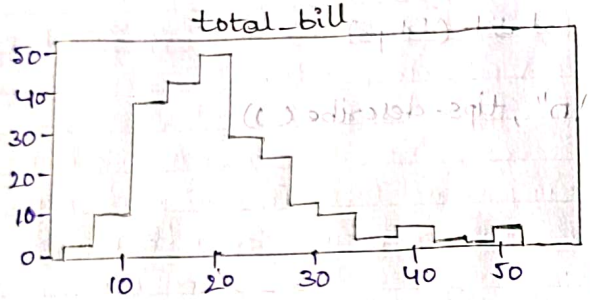
total_bill	0
tip	0
sex	0
smoker	0
day	0
time	0
size	0

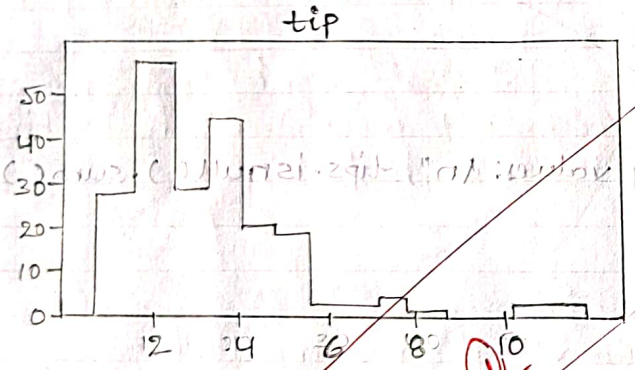
dtype: int64

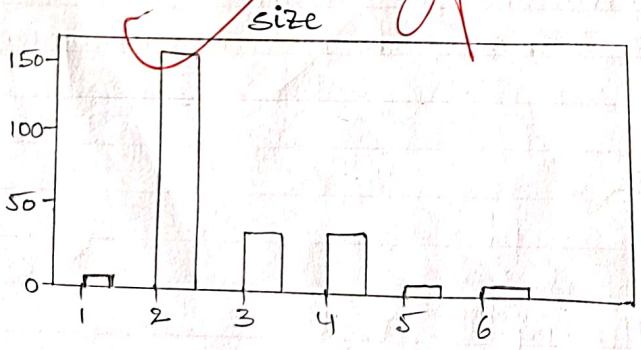

```
tips = sns.load_dataset('tips')

print("Describe:\n", tips.describe())

print("missing values:\n", tips.isnull().sum())
```








```
tips[['total_bill','tip','size']].hist(bins=15,figsize=
                                        (10,5))
plt.tight_layout()
plt.show()
```

Experiment - 13

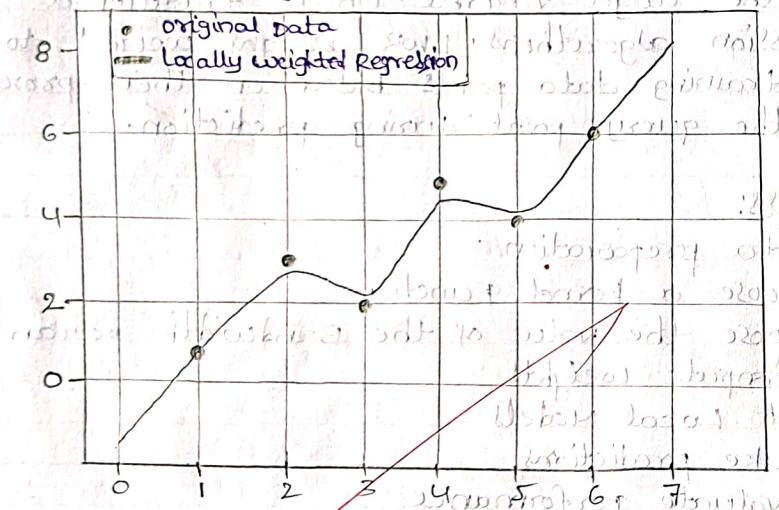
Aim: Implement the non-parametric locally weighted Regression algorithm in order to fit data points, select appropriate data set for your experiment and draw graphs.

Description:

The Locally weighted Regression (LWR) algorithm is a non-parametric regression method that aims to model the relationship between the input features and the target variable. Unlike traditional regression algorithms, LWR assigns weights to the training data points based on their proximity to the query point during prediction.

Process:

1. Data preparation
2. Choose a kernel Function.
3. Choose the value of the Bandwidth parameter
4. Compute weights
5. Fit Local Models
6. Make predictions
7. Evaluate performance



Program:

```
import numpy as np
import matplotlib.pyplot as plt
def gaussian_kernel(x, xi, tau):
    return np.exp((x - xi) ** 2 / (-2 * tau ** 2))
def locally_weighted_regression(x_train, y_train, x_query,
    m = len(x_train)                                    tau):
    X = np.column_stack((np.ones(m), x_train))
    W = np.diag([gaussian_kernel(x_query, xi, tau) for
xi in x_train])
    theta = np.linalg.inv(X.T @ W @ X) @ X.T @ W @
y_train
    return theta
x_train = np.array([1, 2, 3, 4, 5, 6])
y_train = np.array([1, 3, 2, 5, 4, 6])
x_query = np.linspace(0, 7, 100)
tau = 0.5
y_pred = []
for x in x_query:
    theta = locally_weighted_regression(x_train, y_train, x, tau)
    y_pred.append(theta[0] + theta[1] * x)
plt.scatter(x_train, y_train, color='blue', label='original data')
plt.plot(x_query, y_pred, color='red', label='Locally weighted Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Locally weighted Regression')
plt.legend()
plt.grid(True)
plt.show()
```

BVC
GROUP OF INSTITUTIONS
ODALAREVU · AMALAPURAM · RAJAHMUNDRY

Experiment - 14

Aim: Write a program to implement categorical Encoding, One-hot Encoding

Description:

categorical encoding: categorical encoding is a process of converting variable into numerical representation. the machine learning algorithm can understand. categorical variable are represent discret categories or Groups such as product type.

There are several common methods for categorical learning.

- → Label Encoding
- → One Hot Encoding
- → ordinal Encoding

Program:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder,
OneHotEncoder

data = {'color': ['Red', 'Blue', 'Green', 'Red', 'Blue']}
df = pd.DataFrame(data)

label_encoder = LabelEncoder()
df['color_Encoded'] = label_encoder.fit_transform
                      (df['color'])
```

Original Dataframe:

	Color	Color_Encoded
0	Red	2
1	Blue	0
2	Green	1
3	Red	2
4	Blue	0

Categorical Encoded Dataframe:

	Color	Color_Encoded
0	Red	2
1	Blue	0
2	Green	1
3	Red	2
4	Blue	0

One Hot Encoded Dataframe:

	Blue	Green	Red
0	1.0	0.0	1.0
1	1.0	0.0	0.0
2	0.0	1.0	0.0
3	0.0	0.0	1.0
4	1.0	0.0	0.0


```
onehot_encoder = OneHotEncoder(sparse=False)
onehot_encoded = onehot_encoder.fit_transform(df[[
'Color_Encoded']])
onehot_df = pd.DataFrame(onehot_encoded, columns=label_
encoder.classes_)

print("original dataframe:")
print(df)
print("\ncategorical Encoded DataFrame:")
print(df[['Color','Color_Encoded']])
print("\nOne-Hot Encoded DataFrame:")
print(onehot_df)
```

Experiment - 15

Aim: Write a python program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data set.

Description:

A Bayesian network also known as a Bayesian belief network or probabilistic graphical model, is a graphical representation of probabilistic relationships among variables.

1. Conditional probability Distributions (CPDs)
2. Inference
3. Learning

Program:

```
import numpy as np
import pandas as pd
import csv
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianModel
from pgmpy.inference import VariableElimination
import os
os.chdir("C://Users//2ND2AN//AppData//Roaming//Microsoft//Windows//Start Menu//Programs//Python 3.7")

heartDisease = pd.read_csv("new dataset.csv")
```

Sample instances from the dataset are given below:

	age	sex	cp	treatbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	Heart disease
0	63	1	1	145	233	1	2	150	0	2.3	3	0	6	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3	3	2
2	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
3	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0

Attributes and datatypes

```
age          int64
sex          int64
cp           int64
treatbps     int64
chol         int64
fbs          int64
restecg      int64
thalach      int64
exang        int64
oldpeak    float64
slope        int64
ca          object
thal        object
heardisease  int64
dtype: object
```



```
heartDisease = heartDisease.replace('?', np.nan)
print('Sample instances from the dataset are given below')
print(heartDisease.head())

print('\n Attributes and datatypes')
print(heartDisease.dtypes)

model = BayesianModel([('age', 'heartdisease'), ('sex', 'heartdisease'),
('exang', 'heartdisease'), ('cp', 'heartdisease'), ('heartdisease',
'restecg'), ('heartdisease', 'chol')])
print('\nLearning CPD using Maximum likelihood estimators')
model.fit(heartDisease, estimator=MaximumLikelihoodEstimator)

print('\n Inferencing with Bayesian Network:')
HeartDiseasetest_infer.query(variables=['heartdisease'],
evidence={'restecg': 1})
print(q1)

print('\n 2. Probability of HeartDisease given evidence=cp')
q2 = HeartDiseasetest.infer.query(variables=['heartdisease'],
evidence = {'cp': 2})
print(q2)
```

Learning CPD using Maximum likelihood estimators.
Inferencing with Bayesian Network:

1. probability of HeartDisease given evidence=restecg

heartdisease	phi(heartdisease)
heartdisease(0)	0.1012
heartdisease(1)	0.0000
heartdisease(2)	0.2392
heartdisease(3)	0.2015
heartdisease(4)	0.4581

2. probability of HeartDisease given evidence=cp

heartdisease	phi(heartdisease)
heartdisease(0)	0.3610
heartdisease(1)	0.2159
heartdisease(2)	0.1373
heartdisease(3)	0.1537
heartdisease(4)	0.1321