Practical No. 8

Date :

Aim : Develop Python program for normal curves.

Theory :

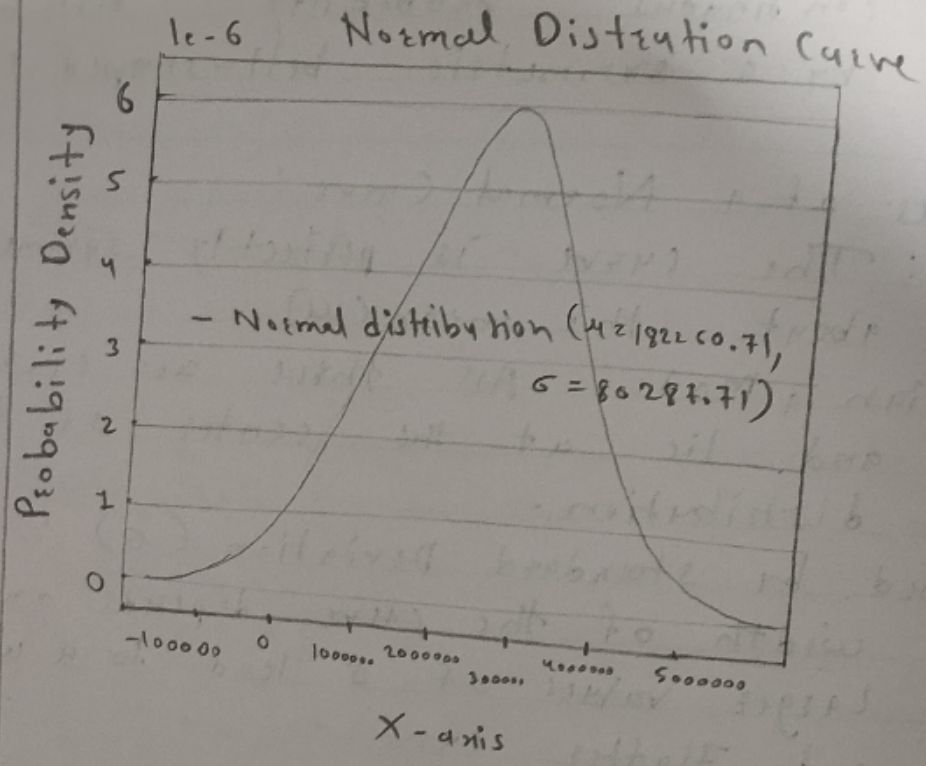
The normal curve, also known as the Gaussian distribution or ~~bell~~ bell curve, is a fundamental concept in statistics. It represents a continuous probability distribution characterized by a symmetric bell-shaped curve.

Key properties of a Normal Curve :

1. Symmetry : The curve is perfectly symmetrical about the mean (μ).
2. Mean, Median, Mode : All three are equal and lie at the center of the distribution.
3. Spread Defined by standard Deviation (σ) : The width of the curve depends on σ. Larger values of σ lead to a wider and flatter curve.
4. Empirical Rule (68-95-99.7 Rule) :
   - 68% of data falls within one standard deviation from the mean.
   - 95% within two standard deviations.
   - 99.7% within three standard deviations.

Program :

```
import numpy as np
import pandas as pd
```



Result: Hence a Program executed Successfu

Date :

```
import matplotlib.pyplot as plt
from scipy.stats import norm

df = pd.read_csv("train.csv")
data = df['SalesPrice'].dropna()

mu = np.mean(data)
sigma = np.std(data)

x = np.linspace(mu - 4*sigma, mu + 4*sigma, 1000)
y = norm.pdf(x, mu, sigma)

plt.plot(x, y, label = f'Normal Distribution (μ = {mu:
    .2f}, σ = {sigma:.2f})', color = 'blue')
plt.xlabel('x-axis')
plt.ylabel('Probability Density')
plt.title('Normal Distribution Curve ~~from CSV Data~~')
plt.legend()
plt.grid()
plt.show()
```

Result : Hence a Program executed Successfully.

Practical No. 9

Date :

Aim: Develop python program for simple linear regression.

Theory:

Simple Linear Regression is a fundamental statistical method used to model the relationship between a dependent variable (Y) and an independent variable (X). It assumes a linear correlation between the variables and is expressed mathematically as:

$$Y = mx + b$$

where;

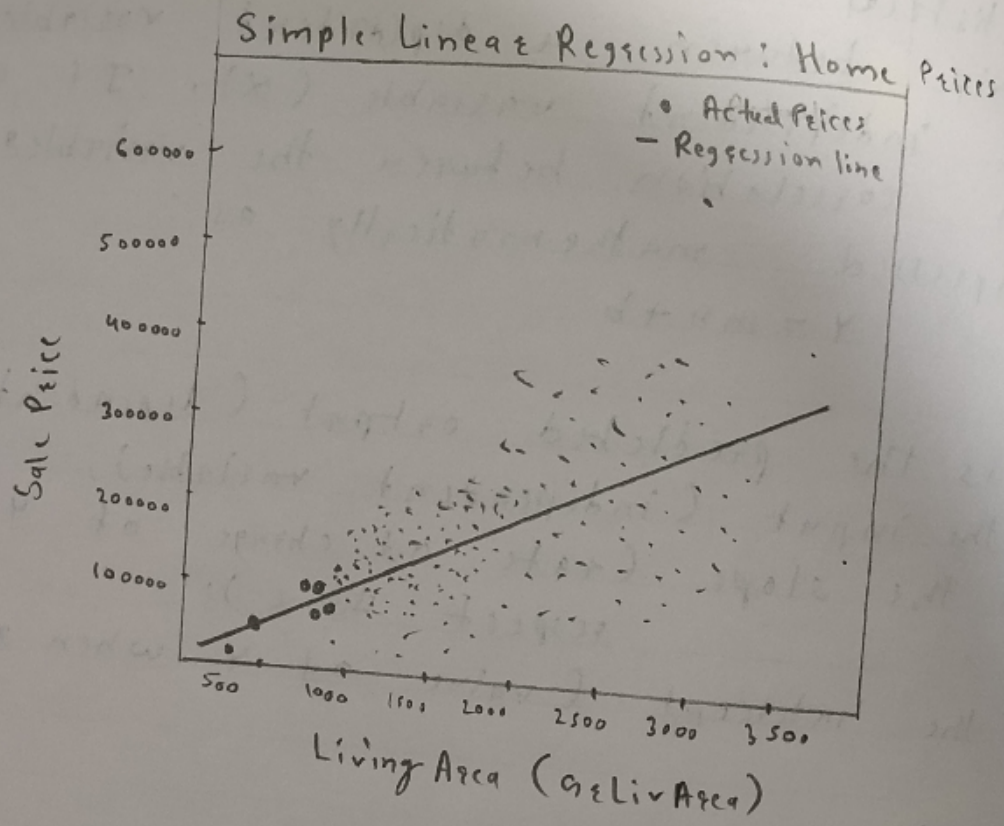
- Y is the predicted output (dependent variable),
- x is the input (independent variable),
- m is the slope (rate of change of Y with respect to x);
- b is the intercept (value of Y when $x = 0$).

Program:

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

data = pd.read_csv('train.csv')

features = 'GrLivArea'
target = 'SalePrice'
```


Simple Linear Regression : Home Prices
Actual Prices
Regression line
600000
500000
400000
300000
200000
100000
Sale Price
500
1000
1500
2000
2500
3000
3500
Living Area (sqLivArea)

Date :

```
X = data [[ feature ]]
y = data [ target ]

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

model = LinearRegression ()
model =
model.fit (X_train , y_train)

y_pred = model.predict (X_test)

print ("coefficient (Slope):", model.coef_[0])
print ("Intercept:", model.intercept_)
print ("R² Score (Accuracy):", model.score (X_test,
                                            y_test))

plt.scatter (X_test , y_test , color = 'blue' , label = 'Actual
                                                Prices')
plt.plot (X_test , y_pred , color = 'red' , label =
                                        'Regression Line")
plt.xlabel ('Living Area (GrLivArea)')
plt.ylabel ('Sale Price')
plt.title ('Simple Linear Regression : Home Prices')
plt.legend ()
plt.show ()
```

Output:

```
Coefficient (Slope): 115.806
Intercept : 7515.372
```

Date :

R^2 Score (Accuracy) : 0.4528

Result : Hence a Program executed successfully.

Practical No. 10

Date :

Aim: Develop python program for logistic regression.

Theory:

Logistic Regression is a statistical and machine learning used for binary classification problems. It predicts the probability that an observation belongs to a particular class and is commonly used when the output variable has two categories, such as Yes / No, Pass / Fail, Spam / not Spam.

$$P(Y=1|x) = \frac{1}{1+e^{-(B_0+B_1x)}}$$

where,
- $P(Y=1|x)$ is the probability of the positive class.
- B_0 is the intercept.
- $B_1 x$ represents the weighted sum of input features.
- e is Euler's number (~2.718).

Program:

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score
```

Date :

```
df = pd.read_csv('train.csv')
df['HighPrice'] = (df['SalePrice'] > 2000000).astype(int)

df.drop(['Id', 'SalesPrice'], axis=1, inplace=True)
df.fillna(df.mode().iloc[0], inplace=True)

label_encoders = {}
for column in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le

scaler = StandardScaler()
X = scaler.fit_transform(df.drop('HighPrice', axis=1))
y = df['HighPrice']

X_train, X_test, y_test = train_test_split(X, y, test_
size=0.2, random_state=42)

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test,
      y_pred))
print("Classification Report:\n", classification_report
      (y_test, y_pred))
```

Result: Hence program executed successfully.

Date :

Output :

Accuracy : 0.93
Confusion Matrix :
[[137 9]
[5 49]]
Classification Report :

	Precision	recall	f1-score	support
0	0.96	0.94	0.95	146
1	0.84	0.91	0.88	54
accuracy			0.93	200
macro avg	0.90	0.92	0.91	200
weighted avg	0.93	0.93	0.93	200

Result : Hence a ~~Executed~~ Program executed successfully.