Sipna College of Engineering & Technology, Amravati.
Department of Computer Science & Engineering
Session 2022-2023

Branch :- Computer Sci. & Engg. **Class :- Final Year**
Subject :-Artificial Intelligence and Machine Learning **Sem :- VIII**

Teacher Manual

PRACTICAL NO 2

AIM: To Understand and implement the concept of supervise learning

S/W REQUIRED: Python
DATA SET USED: iris.csv

Gaussian

Gaussian process models are one of the few machine learning models that can be solved analytically while still being able to model relatively complex systems. Gaussian process models have found various uses in areas such as geostatistics and the optimization of expensive-to-evaluate functions, for example the hyper parameter searches for deep learning neural networks, or the optimization of lasers.

Gaussian Processes Step by Step
Gaussian process models assume that the value of an observed target y_n has the form:

$$y_n = f(\mathbf{x}_n) + e_n,$$

where $f(\mathbf{x}_n)$ is some function giving rise to the observed targets, $\mathbf{x}_n$ is the *n*th row of a set of φ inputs $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_\varphi]^T$, and e_n is independent Gaussian noise. The conditional probability of observing y_n given $f(\mathbf{x}_n)$ is the normal distribution:

$p(y_n|f(x_n)) = N(y_n|f(x_n), \sigma)$,

where σ is the standard deviation of e_n. As the noise is assumed to be independent for each sample, the joint probability distribution of φ observed target values $\mathbf{y} = [y_1, y_2, \ldots y_\varphi]^T$ conditioned on φ values of $\boldsymbol{f}(\mathbf{x})=[f(\mathbf{x}_1), f(\mathbf{x}_2), \ldots f(\mathbf{x}_\varphi)]^T$ is defined to be the normal distribution:

$p(y|f(x)) = N(y|f(x), \sigma)$,

where $\sigma = \sigma I$ is a diagonal matrix of size $\varphi \times \varphi$.

In order to make predictions on y, we need to determine the marginal probability distribution p(y). This probability distribution can be obtained by marginalizing the conditional distribution p(y|f(x)) over the distribution p(f(x)) using the integral:

$p(y) = \int p(y|f(x)) \bullet p(f(x))\, df(x)$.

The distribution p(f(x)) is defined to be a Gaussian distribution with a mean of 0 and covariance kernel matrix K of size $\varphi \times \varphi$:

$p(f(x)) = N(f(x)|0, K)$.

The covariance matrix K is composed of distances between two rows in x, and assumes that similar inputs should give rise to similar target values in y. Each element in the matrix K is computed as:

$K[n, m] = k(x_n, x_m)$,

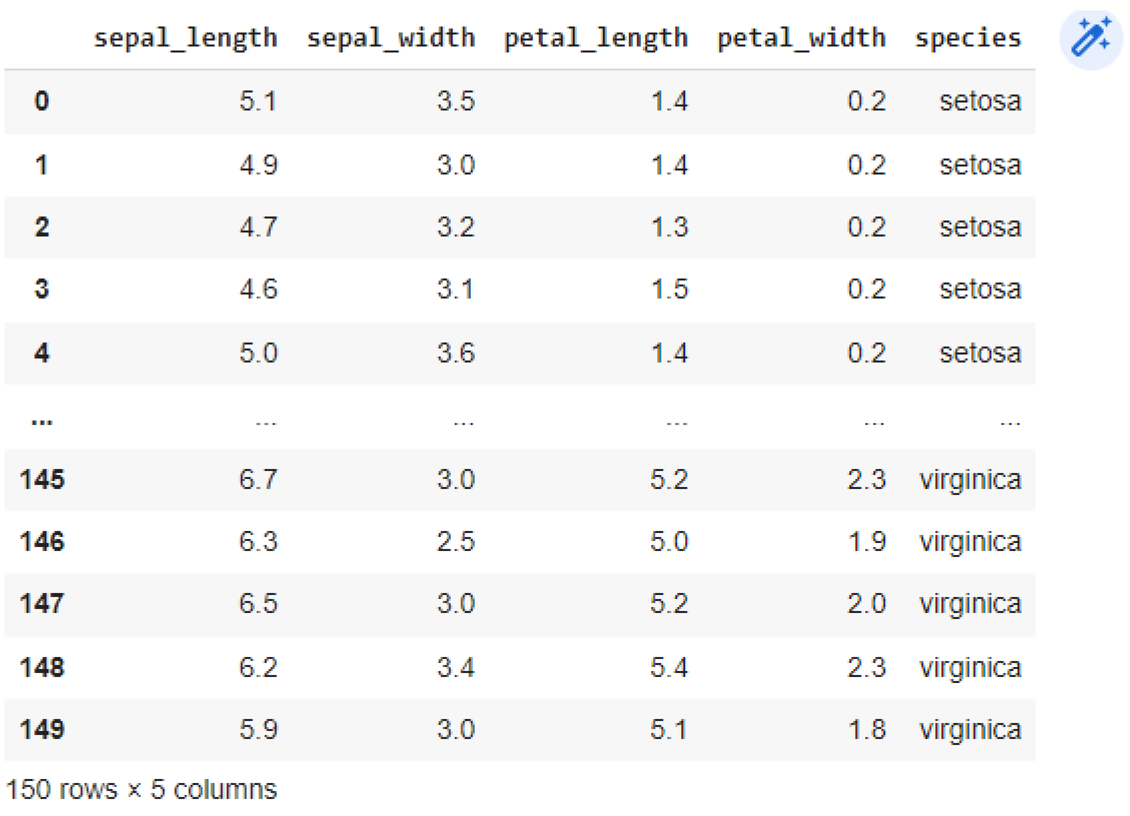
where k is some function to be defined later. Using the equation for p(f(x)) above, we can perform the integral involved in p(y) to obtain the solution:

$p(y) = \int p(y|f(x)) \bullet p(f(x))\, df(x)$
$= \int N(y|f(x),\ \sigma) \bullet N(f(x)|0, K)\, df(x)$
$= N(y|0, C)$.

where the resulting covariance matrix has the form: $C = K+\sigma = K+\sigma I$. Therefore, each element in C can be written as: $C[n, m] = k(x_n, x_m) + \sigma\delta_{nm}$.

Implementation:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

df=pd.read_csv("/content/sample_data/iris.csv")
df
```

Output



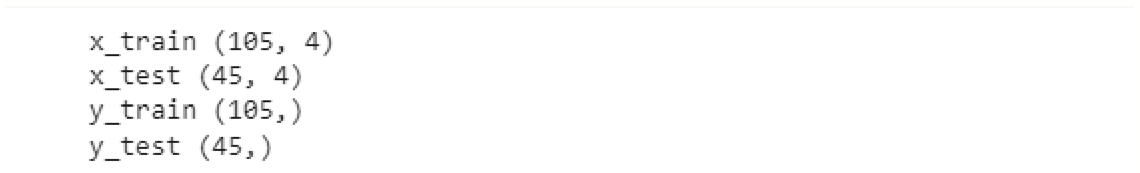
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

```
#split the data
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,stratify=y)


print("x_train",x_train.shape)
print("x_test",x_test.shape)
print("y_train",y_train.shape)
print("y_test",y_test.shape)
```

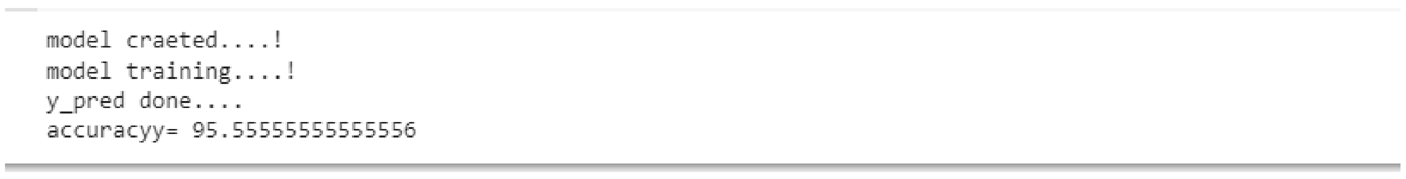
Output



```
x_train (105, 4)
x_test (45, 4)
y_train (105,)
y_test (45,)
```

```
from sklearn.naive_bayes import GaussianNB

#model creation
model=GaussianNB()
print("model craeted....!")

#model training
model.fit(x_train,y_train)
print("model training....!")

#finding pred on test data
y_pred=model.predict(x_test)
print("y_pred done....")

from sklearn.metrics import accuracy_score
acc=accuracy_score(y_pred,y_test)
print("accuracyy=",acc*100)
```

Output



```
model craeted....!
model training....!
y_pred done....
accuracyy= 95.55555555555556
```

CONCLUSION: Thus we have implemented the concept of supervise learning.