Implementation of Naive Bayes from Scratch

Name: Ankita Dasgupta

Roll No: J014

In [76]:
```
#from sklearn.datasets import load_iris
import pandas as pd
df=pd.read_csv("C:\\Users\\ankit\\Anaconda3\\lib\\site-
packages\\sklearn\\datasets\\data\\iris.csv")
df.columns = ['sepal_length','sepal_width','petal_length','petal_width',
'iris']
classes = ['setosa', 'versicolor', 'virginica']

for i in range(0,3):
    df.loc[df.iris==i, 'iris'] = classes[i]
df.head()
```

Out[76]:

	sepal_length	sepal_width	petal_length	petal_width	iris
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

In [24]:
```
prob_classes=dict()
for unique_class in classes:
    prob_classes[unique_class] = None
    for key in prob_classes.keys() :
        prob_classes[key] = 50/150
prob_classes
```

Out[24]:
```
{'setosa': 0.3333333333333333,
 'versicolor': 0.3333333333333333,
 'virginica': 0.3333333333333333}
```

In [79]:
```
data = dict()
for c in classes:
    data[c] = dict()
    data[c]["mean"] = dict()
    data[c]["std_dev"] = dict()
    for col in df.columns[:4]:
        data[c]["mean"][col]=df.loc[df["iris"] == c].mean()[col]
```

```
        data[c]["std_dev"][col]=df.loc[df["iris"] == c].std()[col]
data
```

Out[79]:
```
{'setosa': {'mean': {'sepal_length': 5.005999999999999,
   'sepal_width': 3.428000000000001,
   'petal_length': 1.4620000000000002,
   'petal_width': 0.2459999999999999},
  'std_dev': {'sepal_length': 0.3524896872134512,
   'sepal_width': 0.3790643690962886,
   'petal_length': 0.1736639964801841,
   'petal_width': 0.10538558938004569}},
 'versicolor': {'mean': {'sepal_length': 5.936,
   'sepal_width': 2.7700000000000005,
   'petal_length': 4.26,
   'petal_width': 1.3259999999999998},
  'std_dev': {'sepal_length': 0.5161711470638635,
   'sepal_width': 0.3137983233784114,
   'petal_length': 0.4699109772399580 6,
   'petal_width': 0.197752680004544}},
 'virginica': {'mean': {'sepal_length': 6.587999999999998,
   'sepal_width': 2.9739999999999998,
   'petal_length': 5.552,
   'petal_width': 2.026},
  'std_dev': {'sepal_length': 0.635879593274432,
   'sepal_width': 0.3224966381726376,
   'petal_length': 0.5518946956639835,
   'petal_width': 0.2746500556366673 3}}}
```

In [86]:
```
import numpy as np

prediction = pd.DataFrame(columns=classes , index=range(0,len(df),1))

for c in classes:
    for idx, r in df.iterrows():
        prob = 1
        prob = prob_classes[c]
        for col in df.columns[:4]:
            t1 = 1/(data[c]["std_dev"][col]*((2*np.pi)**0.5))
            a = ((-((df.loc[idx,col]-data[c]["mean"][col])**2)))
            b = (2*(data[c]["std_dev"][col]**2))
            t2 = np.exp(a/b)
            prob = prob*t1*t2
        prediction.loc[idx , c] = prob

prediction
```

Out[86]:

	setosa	versicolor	virginica
0	2.791534	0.0	0.0
1	1.488164	0.0	0.0
2	1.163145	0.0	0.0
3	1.085765	0.0	0.0
4	2.656738	0.0	0.0

	setosa	versicolor	virginica
...	...	...	...
145	0.0	0.0	0.132245
146	0.0	0.001295	0.045437
147	0.0	0.000096	0.217838
148	0.0	0.0	0.055163
149	0.0	0.004895	0.076857

150 rows × 3 columns

In [95]:
```
prediction["Prediction"] = [0]*150
for idx, r in prediction.iterrows():
    i = 0
    for c in r[:-1]:
        if max(r[:-1]) == c:
            prediction.loc[idx , 'Prediction'] =prediction.columns[i]
        i += 1

prediction
```

Out[95]:

	setosa	versicolor	virginica	Prediction
0	2.791534	0.0	0.0	setosa
1	1.488164	0.0	0.0	setosa
2	1.163145	0.0	0.0	setosa
3	1.085765	0.0	0.0	setosa
4	2.656738	0.0	0.0	setosa
...	...	...	...	...
145	0.0	0.0	0.132245	virginica
146	0.0	0.001295	0.045437	virginica
147	0.0	0.000096	0.217838	virginica
148	0.0	0.0	0.055163	virginica
149	0.0	0.004895	0.076857	virginica

150 rows × 4 columns

In [96]:
```
correct = 0
for i in range(0,150):
    if prediction.Prediction[i] == df.iris[i]:
        correct += 1
print("Accuracy: ", (correct*100)/150)
```

```
Accuracy:  96.0
```

In [98]:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

clf = GaussianNB()
clf.fit(df.iloc[:,:4], df.iris)

predictions = clf.predict(df.iloc[:,:4])
print(accuracy_score(df.iloc[:,4] , predictions))
```

```
0.96
```

In [ ]:

```

```