ML SUBMISSION 11

IMPLEMENTATION OF NAIVE BAYES FROM SCRATCH
J042 AVANI NARVEKAR

In [2]:

```
import pandas as pd
import numpy as np
from sklearn import datasets
from collections import Counter
```

In [4]:

```
data = pd.read_csv('Downloads/archive/Iris.csv')
data.head()
```

Out[4]:

	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

In [6]:

```
data['species'].value_counts()
```

Out[6]:

```
setosa        50
virginica     50
versicolor    50
Name: species, dtype: int64
```

In [7]:

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data, test_size = 0.3, random_state = 7)
```

In [10]:
```
class NB():
    def __init__(self,train):
        self.train = train
        self.X_train = train.drop('species', axis = 1)
        self.Y_train = train['species']
        self.s = {}

    def fit(self):
        self.result = Counter(self.Y_train)

        for target in self.result.keys():
            for col in self.X_train.columns:
                self.s[target,col,"mean"] = self.train[self.train['species'] == t
                self.s[target,col,"std"] = self.train[self.train['species'] == ta

        for i in self.result:
            self.result[i] = round(self.result[i]/len(self.X_train.index),8)

    def predict(self,X_test):
        count = 0
        prediction = []
        for i in X_test.index:
            prob_index = {}
            for target in self.result:
                prob = self.result[target]
                for col in self.X_train:
                    a = 1/(((2*np.pi)**0.5)*self.s[target,col,"std"])
                    b = -((X_test[col][i] - self.s[target,col,"mean"])**2)
                    c = 2*(self.s[target,col,"std"]**2)
                    prob = prob * a * np.exp(b/c)
                prob_index[target] = prob

            probability = 0
            for target in prob_index:
                if prob_index[target] > probability:
                    pred = target
                    probability = prob_index[target]
            prediction.append(pred)

        return prediction
```

In [11]:
```
clf = NB(train)
clf.fit()
```

In [12]:
```
Y_test = test['species']
X_test = test.drop('species', axis = 1)
predictions = clf.predict(X_test)
```

In [13]:
```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test, predictions)
```

Out[13]:
```
0.9666666666666667
```

In [14]:

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
mod = gnb.fit(data.iloc[:,:4], data.iloc[:,4])
predictions1 = clf.predict(data.iloc[:,:4])
accuracy_score(data.iloc[:,4], predictions1)
```

Out[14]:

```
0.9666666666666667
```