In [1]:

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

In [ ]:

```

```

In [2]:

```
iris = datasets.load_iris()
species = iris.target


data = pd.DataFrame ( np.c_[ iris.data, species.reshape((species.shape[0],1))], columns
= iris.feature_names + ['species'])
data.head()
```

Out[2]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

In [3]:

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

Out[3]:

```
0.0    50
2.0    50
1.0    50
Name: species, dtype: int64
```

In [4]:

```
from sklearn.model_selection import train_test_split
train, test = train_test_split( data, test_size = 0.2, random_state = 123)
```

In [12]:

```
test.shape,train.shape
```

Out[12]:

```
((30, 5), (120, 5))
```

In [5]:

```
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'] == target]
.mean()[col]
                self.s[target,col,"std"] = self.train[ self.train['species'] == target].
std()[col]

        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 [6]:

```
Y_train = train['species']
X_train = train.drop( 'species' , axis = 1)
```

In [7]:

```
clf = NB(train)
clf.fit()
```

In [8]:

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

In [9]:

```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test, predictions)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-9-e398b8727b23> in <module>
      1 from sklearn.metrics import accuracy_score
----> 2 accuracy_score(Y_test, predictions)

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64 
     65             # extra_args > 0

~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py in accuracy_score(y_true
, y_pred, normalize, sample_weight)
    200
    200
    201     # Compute accuracy for each possible representation
--> 202     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    203     check_consistent_length(y_true, y_pred, sample_weight)
    204     if y_type.startswith('multilabel'):

~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py in _check_targets(y_true
, y_pred)
     81     y_pred : array or indicator matrix
     82     """
---> 83     check_consistent_length(y_true, y_pred)
     84     type_true = type_of_target(y_true)
     85     type_pred = type_of_target(y_pred)

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_consistent_length(*arr
ays)
    260     uniques = np.unique(lengths)
    261     if len(uniques) > 1:
--> 262         raise ValueError("Found input variables with inconsistent numbers of"
    263                          " samples: %r" % [int(l) for l in lengths])
    264

ValueError: Found input variables with inconsistent numbers of samples: [30, 50]
```

In [10]:

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
sk_predictions = gnb.fit(X_train, Y_train).predict(X_test)
accuracy_score(Y_test, predictions)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-10-1671406cef1b> in <module>
      2 gnb = GaussianNB()
      3 sk_predictions = gnb.fit(X_train, Y_train).predict(X_test)
----> 4 accuracy_score(Y_test, predictions)

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in inner_f(*args, **kwargs)
     61             extra_args = len(args) - len(all_args)
     62             if extra_args <= 0:
---> 63                 return f(*args, **kwargs)
     64
     65             # extra_args > 0

~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py in accuracy_score(y_true
, y_pred, normalize, sample_weight)
    200
    201     # Compute accuracy for each possible representation
--> 202     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    203     check_consistent_length(y_true, y_pred, sample_weight)
    204     if y_type.startswith('multilabel'):

~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py in _check_targets(y_true
, y_pred)
     81     y_pred : array or indicator matrix
     82     """
---> 83     check_consistent_length(y_true, y_pred)
     84     type_true = type_of_target(y_true)
     85     type_pred = type_of_target(y_pred)

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_consistent_length(*arr
ays)
    260     uniques = np.unique(lengths)
    261     if len(uniques) > 1:
--> 262         raise ValueError("Found input variables with inconsistent numbers of"
    263                          " samples: %r" % [int(l) for l in lengths])
    264

ValueError: Found input variables with inconsistent numbers of samples: [30, 50]
```

Naive bayes works the same as our model

In [ ]: