Aryaman Gautam

J001

In [1]:

```
import numpy as np
import pandas as pd
import scipy.stats as stats
```

In [2]:

```
class NaiveBayesClassifier():

    def calc_prior(self, features, target):
        self.prior = (features.groupby(target).apply(lambda x: len(x)) / self.rows).to_numpy()
        return self.prior

    def calc_statistics(self, features, target):
        self.mean = features.groupby(target).apply(np.mean).to_numpy()
        self.var = features.groupby(target).apply(np.var).to_numpy()
        return self.mean, self.var

    def gaussian_density(self, class_idx, x):
        mean = self.mean[class_idx]
        var = self.var[class_idx]
        numerator = np.exp((-1/2)*((x-mean)**2) / (2 * var))
        denominator = np.sqrt(2 * np.pi * var)
        prob = numerator / denominator
        return prob

    def calc_posterior(self, x):
        posteriors = []
        for i in range(self.count):
            prior = np.log(self.prior[i])
            conditional = np.sum(np.log(self.gaussian_density(i, x)))
            posterior = prior + conditional
            posteriors.append(posterior)
        return self.classes[np.argmax(posteriors)]
    def fit(self, features, target):
        self.classes = np.unique(target)
        self.count = len(self.classes)
        self.feature_nums = features.shape[1]
        self.rows = features.shape[0]
        self.calc_statistics(features, target)
        self.calc_prior(features, target)

    def predict(self, features):
        preds = [self.calc_posterior(f) for f in features.to_numpy()]
        return preds

    def accuracy(self, y_test, y_pred):
        accuracy = np.sum(y_test == y_pred) / len(y_test)
        return accuracy
```

In [3]:

```
df=pd.read_csv('irisn.csv')
df.head()
```

Out[3]:

	sepal.length	sepal.width	petal.length	petal.width	variety
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 [4]:

```
df = df.sample(frac=1, random_state=1).reset_index(drop=True)


print(df.shape)


X, y = df.iloc[:, :-1], df.iloc[:, -1]
```

```
X_train, X_test, y_train, y_test = X[:100], X[100:], y[:100], y[100:]


print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
(150, 5)
(100, 4) (100,)
(50, 4) (50,)
```

In [8]:

```
x.classes, x.feature_nums, x.rows, x.count
```

Out[8]:

```
(array(['Setosa', 'Versicolor', 'Virginica'], dtype=object), 4, 100, 3)
```

In [9]:

```
x.calc_prior(X_train, y_train)
x.prior
x.calc_statistics(X_train, y_train)
```

Out[9]:

```
(array([[5.08387097, 3.50322581, 1.46129032, 0.24516129],
        [5.9125    , 2.790625  , 4.275     , 1.33125   ],
        [6.71891892, 2.98918919, 5.63243243, 2.05675676]]),
 array([[0.11361082, 0.10934443, 0.02430801, 0.0089282 ],
        [0.21296875, 0.08272461, 0.185625  , 0.03214844],
        [0.3566691 , 0.11339664, 0.32867787, 0.0592111 ]]))
```

In [10]:

```
x.mean
```

Out[10]:

```
array([[5.08387097, 3.50322581, 1.46129032, 0.24516129],
       [5.9125    , 2.790625  , 4.275     , 1.33125   ],
       [6.71891892, 2.98918919, 5.63243243, 2.05675676]])
```

In [11]:

```
x.var
```

Out[11]:

```
array([[0.11361082, 0.10934443, 0.02430801, 0.0089282 ],
       [0.21296875, 0.08272461, 0.185625  , 0.03214844],
       [0.3566691 , 0.11339664, 0.32867787, 0.0592111 ]])
```

In [5]:

```
x = NaiveBayesClassifier()


x.fit(X_train, y_train)
```

In [6]:

```
predictions = x.predict(X_test)
```

In [7]:

```
x.accuracy(y_test, predictions)
```

Out[7]:

```
0.92
```

In [ ]: