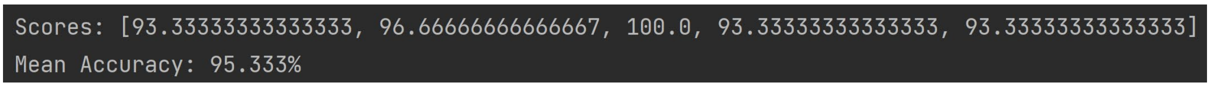
Homework 5 Report

Brian King

Question 1

Using 5 folds with the Naïve Bayes algorithm obtains the following accuracies for the validation sets:

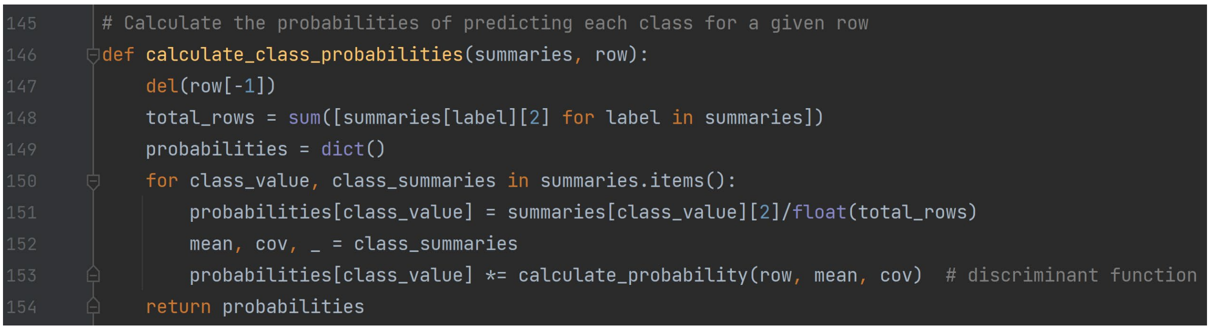


```
Scores: [93.33333333333333, 96.66666666666667, 100.0, 93.33333333333333, 93.33333333333333]
Mean Accuracy: 95.333%
```

Question 2

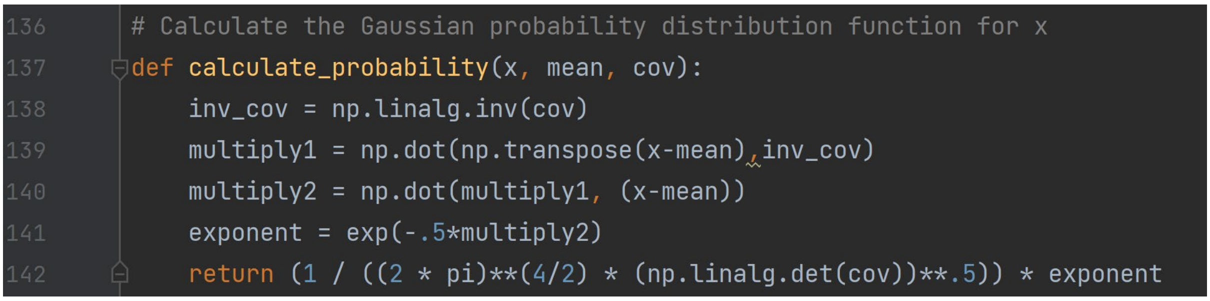
The following functions were modified in order to implement a Multivariate Bayes classifier in place of the Naive Bayes classifier:

Calculate_class_probabilities function



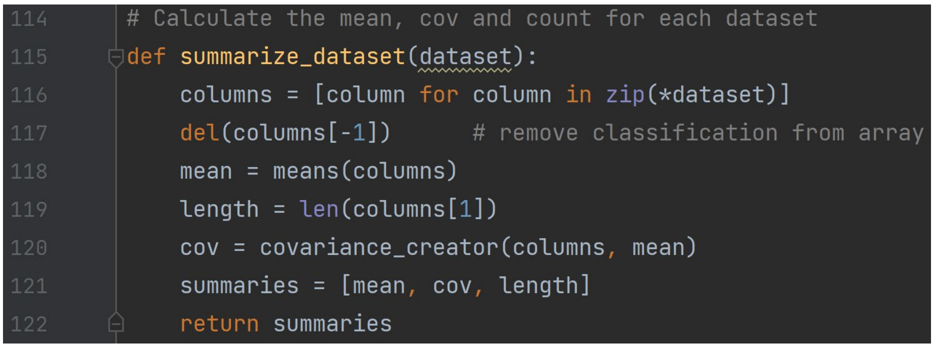
```python
# Calculate the probabilities of predicting each class for a given row
def calculate_class_probabilities(summaries, row):
    del(row[-1])
    total_rows = sum([summaries[label][2] for label in summaries])
    probabilities = dict()
    for class_value, class_summaries in summaries.items():
        probabilities[class_value] = summaries[class_value][2]/float(total_rows)
        mean, cov, _ = class_summaries
        probabilities[class_value] *= calculate_probability(row, mean, cov)  # discriminant function
    return probabilities
```

Calculate_probability function



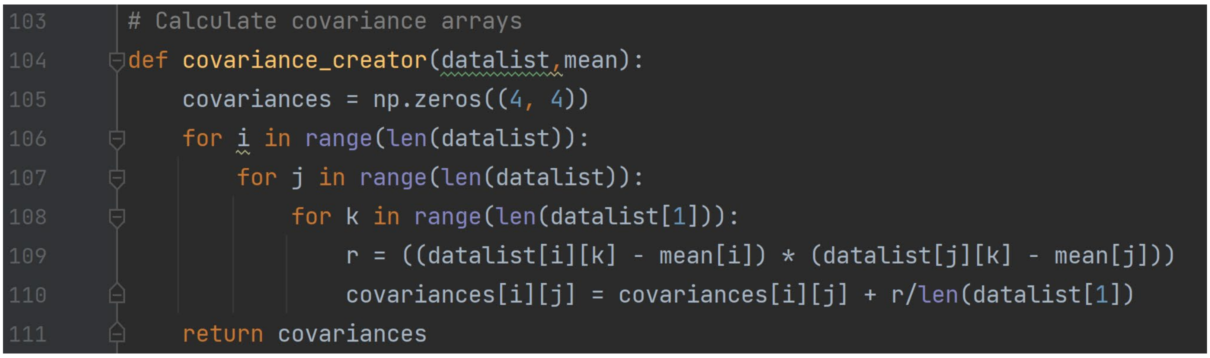
```python
# Calculate the Gaussian probability distribution function for x
def calculate_probability(x, mean, cov):
    inv_cov = np.linalg.inv(cov)
    multiply1 = np.dot(np.transpose(x-mean),inv_cov)
    multiply2 = np.dot(multiply1, (x-mean))
    exponent = exp(-.5*multiply2)
    return (1 / ((2 * pi)**(4/2) * (np.linalg.det(cov))**.5)) * exponent
```

Summarize_dataset function



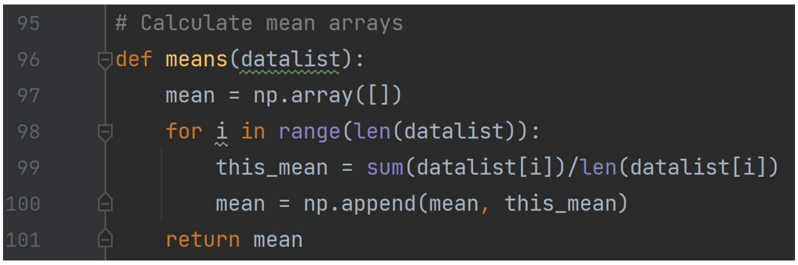
```python
# Calculate the mean, cov and count for each dataset
def summarize_dataset(dataset):
    columns = [column for column in zip(*dataset)]
    del(columns[-1])      # remove classification from array
    mean = means(columns)
    length = len(columns[1])
    cov = covariance_creator(columns, mean)
    summaries = [mean, cov, length]
    return summaries
```

Covariance function



```python
# Calculate covariance arrays
def covariance_creator(datalist,mean):
    covariances = np.zeros((4, 4))
    for i in range(len(datalist)):
        for j in range(len(datalist)):
            for k in range(len(datalist[1])):
                r = ((datalist[i][k] - mean[i]) * (datalist[j][k] - mean[j]))
                covariances[i][j] = covariances[i][j] + r/len(datalist[1])
    return covariances
```

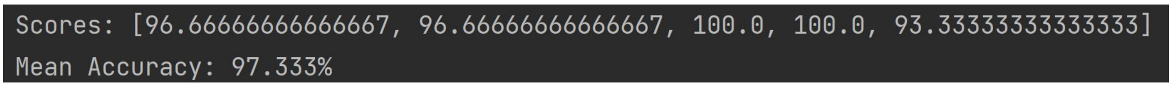
Mean function



```python
# Calculate mean arrays
def means(datalist):
    mean = np.array([])
    for i in range(len(datalist)):
        this_mean = sum(datalist[i])/len(datalist[i])
        mean = np.append(mean, this_mean)
    return mean
```

Question 3

Utilizing the aforementioned Multivariate Bayes classifier, the following accuracies were obtained.



```
Scores: [96.66666666666667, 96.66666666666667, 100.0, 100.0, 93.33333333333333]
Mean Accuracy: 97.333%
```

This is an improvement of 2% over the Naïve Bayes classifier. However, with large amounts of data (greater than the previous 150 data points used in the report), it is more efficient to use the Naïve Bayes classifier as it computationally less intensive.