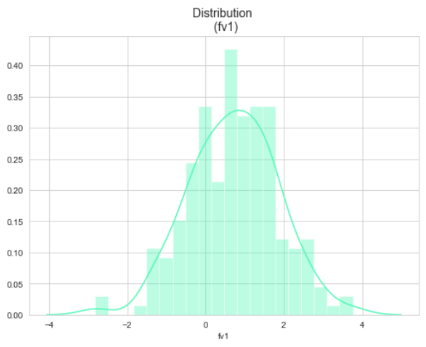
Homework 2

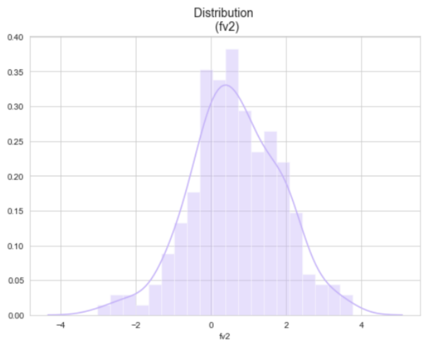
Abhishek Reddy Palle - 128003556

Problem 1)

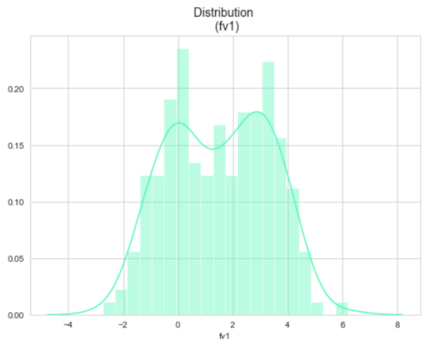
Exploratory Data Analysis

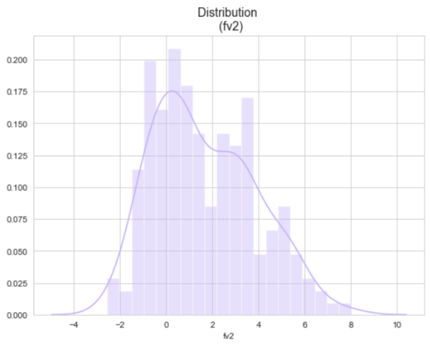
Feature Distribution for part A



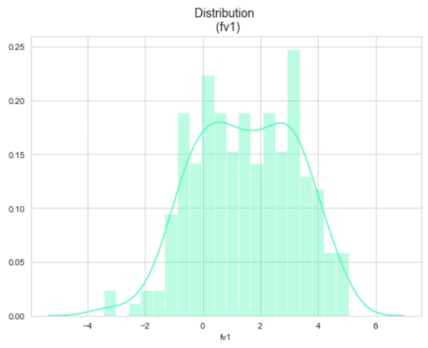


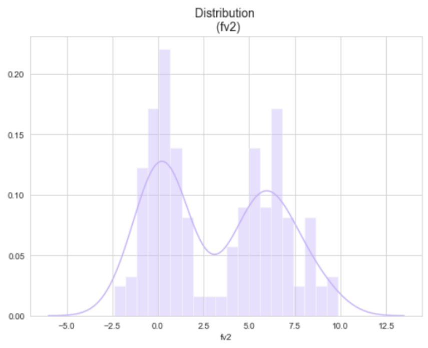
Feature Distribution for part B



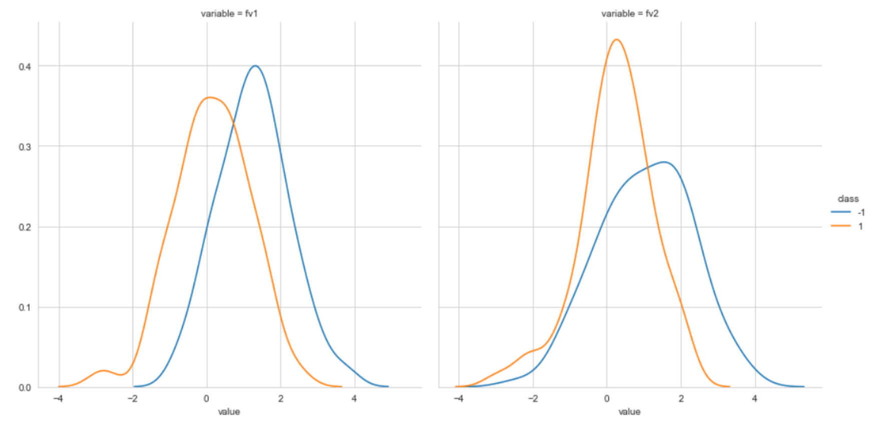


Feature Distribution for part C

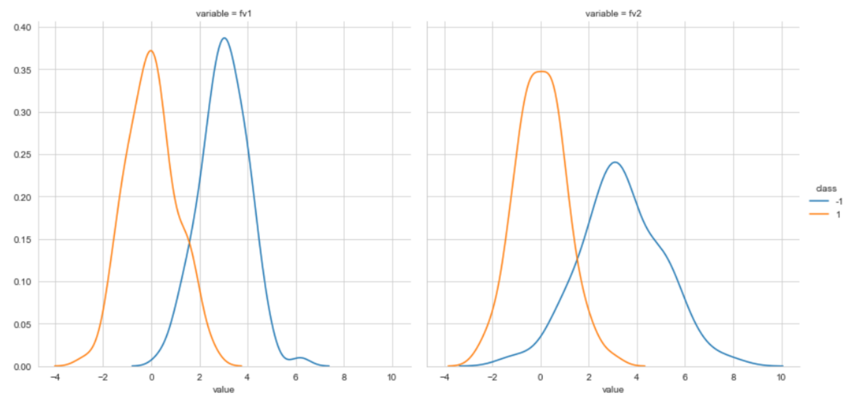




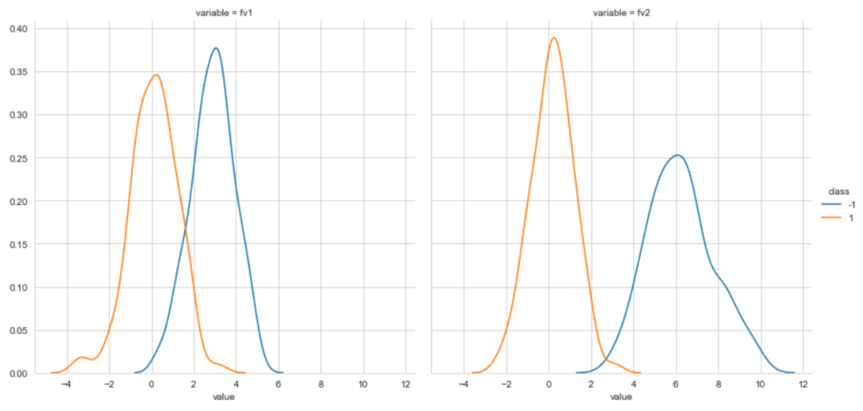
Class Conditioned Probability Distribution for part A



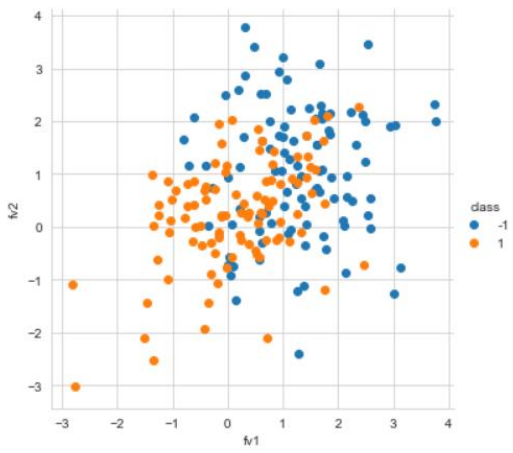
Class Conditioned Probability Distribution for part B



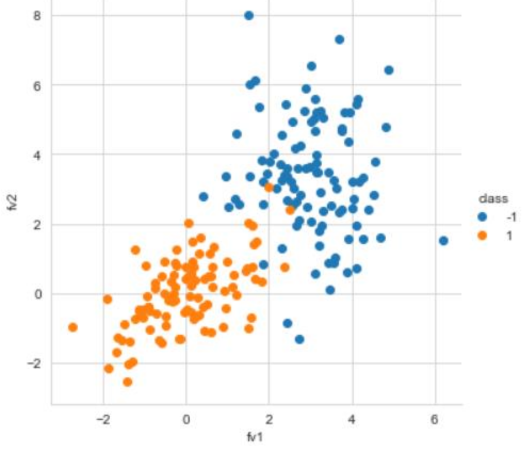
Class Conditioned Probability Distribution for part C



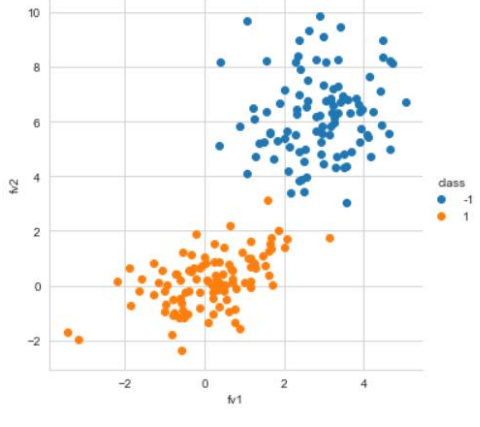
Scatter Plot between features for part A



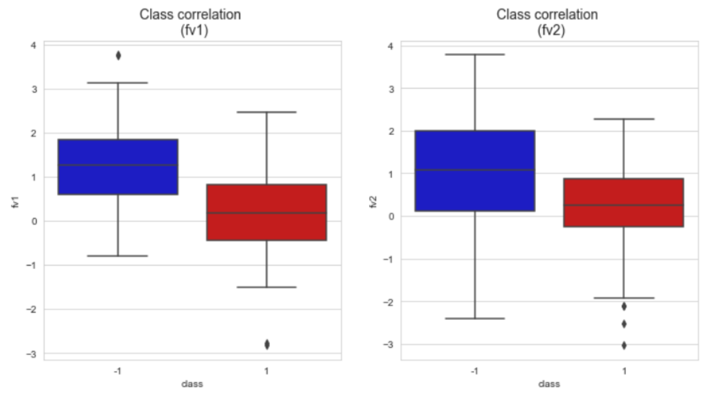
Scatter Plot between features for part B



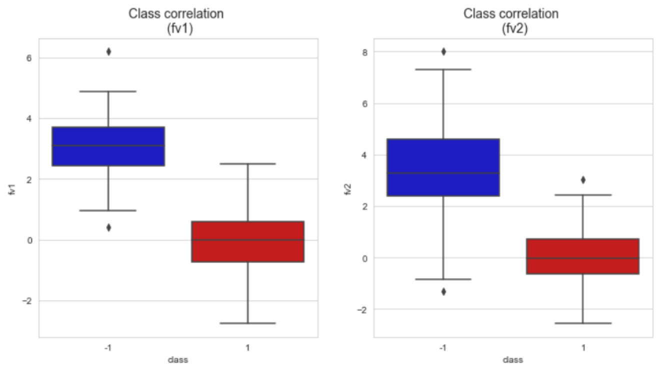
Scatter Plot between features for part C



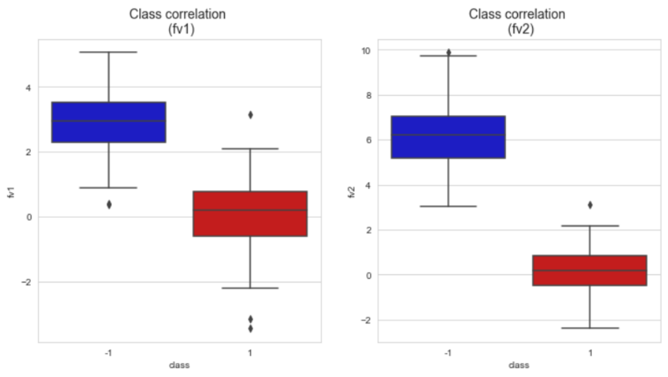
Box Plot to analyze Class Separability for part A



Box Plot to analyze Class Separability for part B



Box Plot to analyze Class Separability for part C

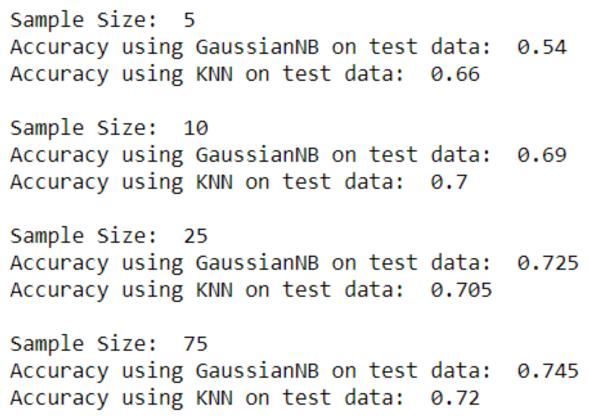


Model Analysis

Sub-problem A)

We compare the accuracy of the Bayes classifier with that of nearest neighbor classifier. In each case, we use the same set of training examples (as used for estimation). We get the following insights:

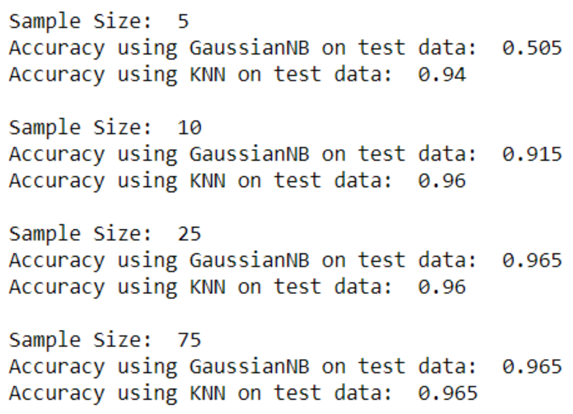
- We observe that accuracy increases as the training sample size increases.
- We also observe that Bayes classifier performs better than nearest neighbor classifier for higher training sample sizes.
- We also observe that accuracy achieved isn't too high since the data is highly non-separable as seen in the exploratory data analysis part.



```
Sample Size:  5
Accuracy using GaussianNB on test data:  0.54
Accuracy using KNN on test data:  0.66

Sample Size:  10
Accuracy using GaussianNB on test data:  0.69
Accuracy using KNN on test data:  0.7

Sample Size:  25
Accuracy using GaussianNB on test data:  0.725
Accuracy using KNN on test data:  0.705

Sample Size:  75
Accuracy using GaussianNB on test data:  0.745
Accuracy using KNN on test data:  0.72
```

Sub-problem B)

Here we observe better accuracies since the data is linearly separable. Also, the accuracies with increase in training sample size.



```
Sample Size:  5
Accuracy using GaussianNB on test data:  0.505
Accuracy using KNN on test data:  0.94

Sample Size:  10
Accuracy using GaussianNB on test data:  0.915
Accuracy using KNN on test data:  0.96

Sample Size:  25
Accuracy using GaussianNB on test data:  0.965
Accuracy using KNN on test data:  0.96

Sample Size:  75
Accuracy using GaussianNB on test data:  0.965
Accuracy using KNN on test data:  0.965
```

Now we ignore class labels, and only use the features in the training set to learn a two-component mixture density using EM algorithm. We then use the learnt class conditional densities and implement Bayes classifier and compare against the Bayes classifier learnt using the class labels. For two normal distributions A and B with means μA and μB and variances σA and σB, the mean and variance of this mixture would be:

$$\mu_{AB} = (p \times \mu_A) + (q \times \mu_B).$$

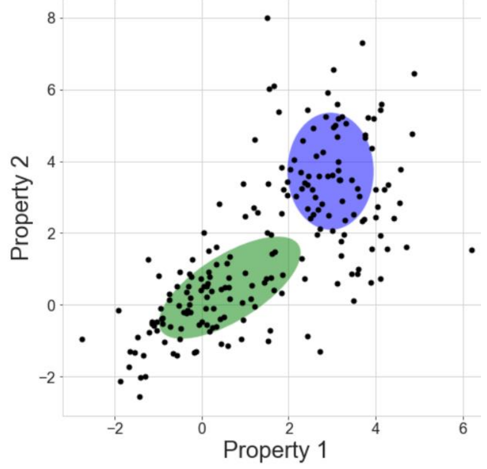
$$p_A\sigma_A^2 + p_B\sigma_B^2 + \left[p_A\mu_A^2 + p_B\mu_B^2 - (p_A\mu_A + p_B\mu_B)^2\right].$$

We get the following insights:

- Accuracy achieved in this unsupervised learning scenario is better than the one where we use 75 training samples to estimate parameters.

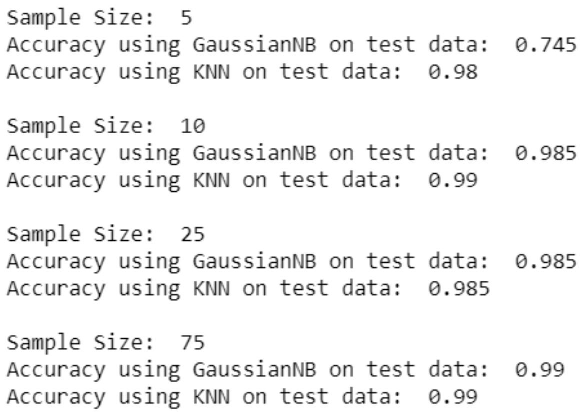
```
Accuracy using Gaussian Mixture Model on test data:  0.975

Mixing proportion: [0.52477346 0.47522654]

mu:  [[ 2.96081634  3.24353579]
 [-0.10012241 -0.05290842]]

cov:  [[[ 0.99009353 -0.02215773]
  [-0.02215773  2.9617381 ]]

 [[ 0.91739142  0.48950588]
  [ 0.48950588  0.96065441]]]
```

- Clusters observed

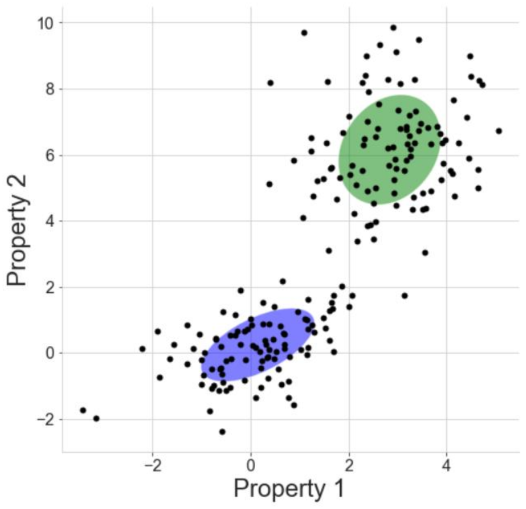


Sub-problem C)

Here for this scenario we observe best accuracies (99%) since the data is highly linearly separable. Also, the accuracies with increase in training sample size.



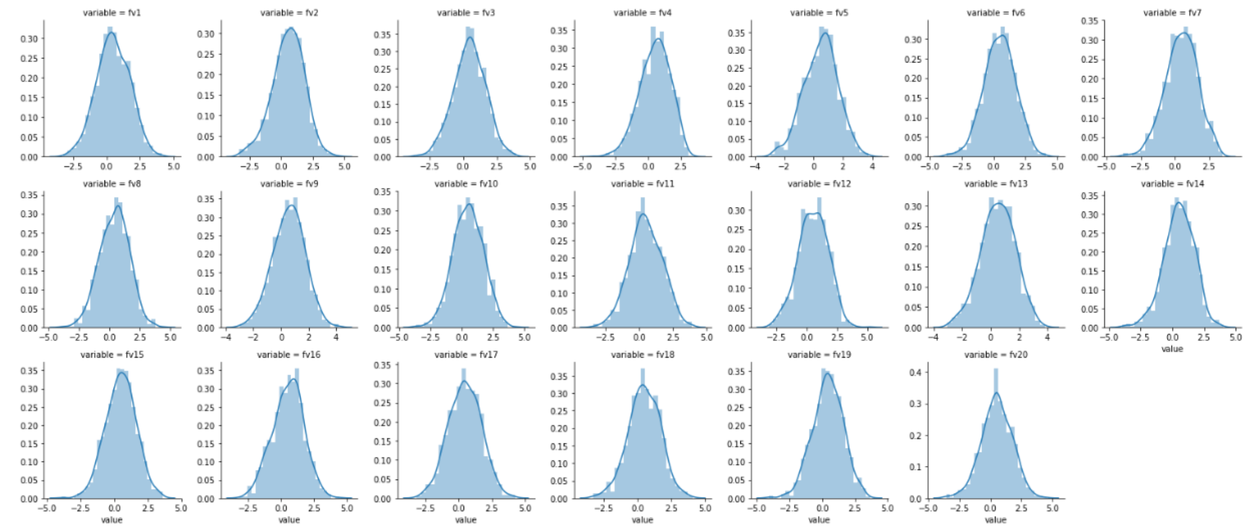
```
Sample Size:  5
Accuracy using GaussianNB on test data:  0.745
Accuracy using KNN on test data:  0.98

Sample Size:  10
Accuracy using GaussianNB on test data:  0.985
Accuracy using KNN on test data:  0.99

Sample Size:  25
Accuracy using GaussianNB on test data:  0.985
Accuracy using KNN on test data:  0.985

Sample Size:  75
Accuracy using GaussianNB on test data:  0.99
Accuracy using KNN on test data:  0.99
```



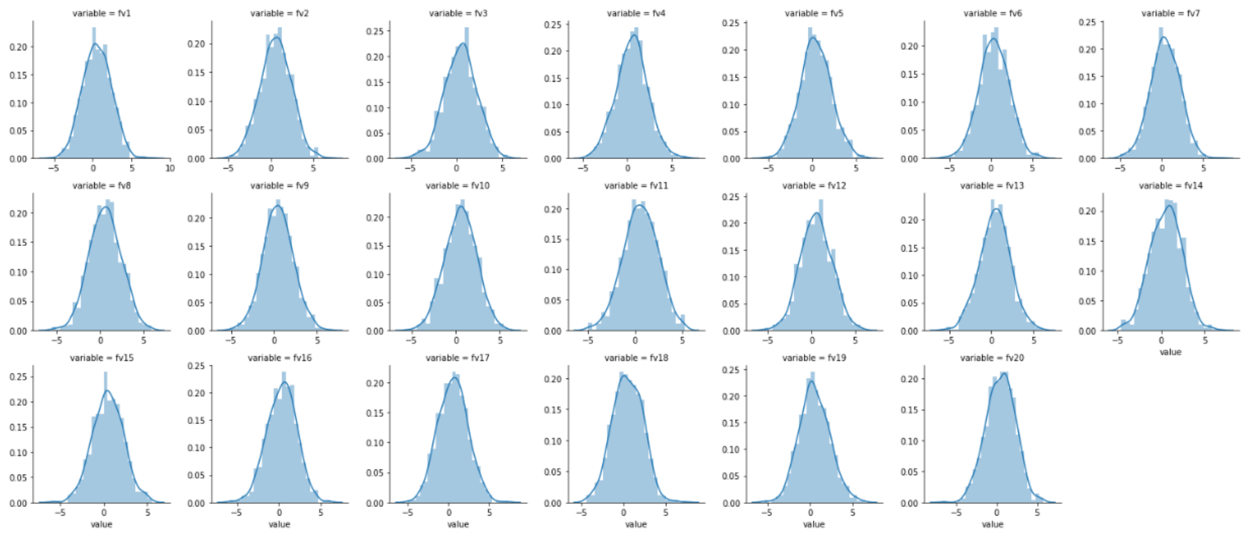
Problem 2)

Exploratory Data Analysis

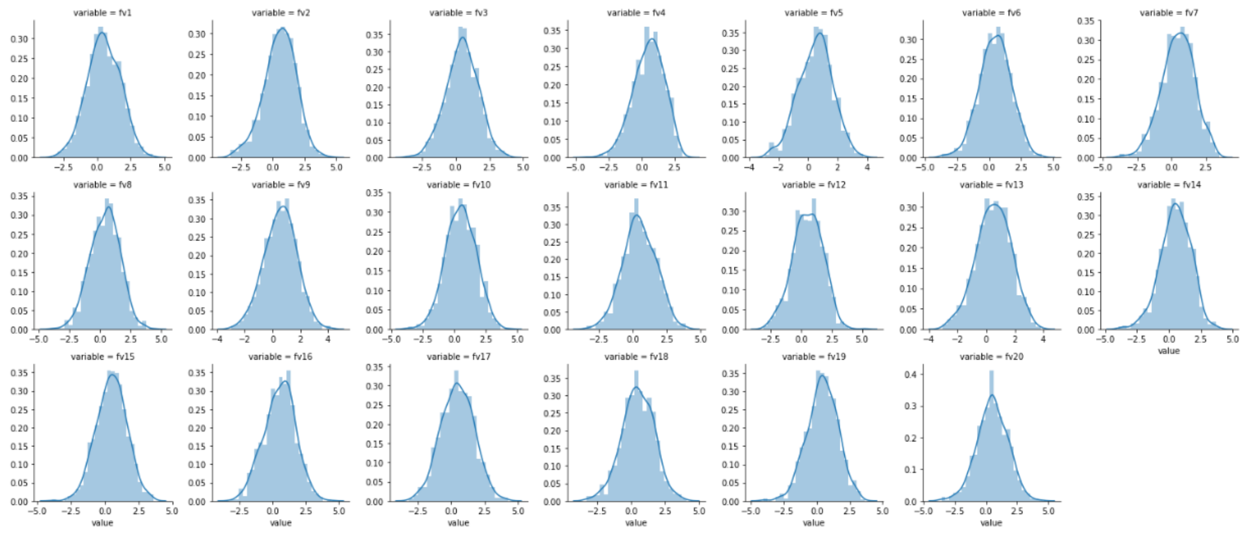
Kernel Density Estimation plot for part A



Kernel Density Estimation plot for part B



Kernel Density Estimation plot for part C

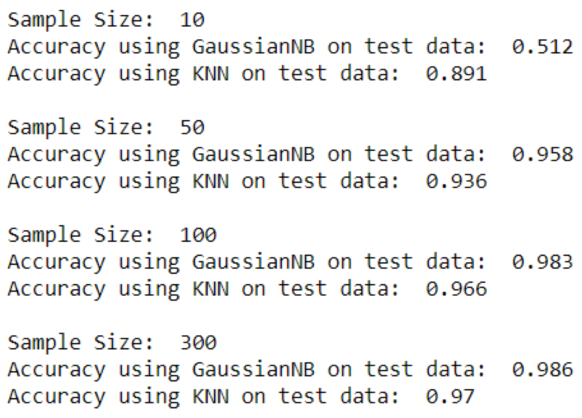


Model Analysis

For sub-problem A, B, & C – We observe

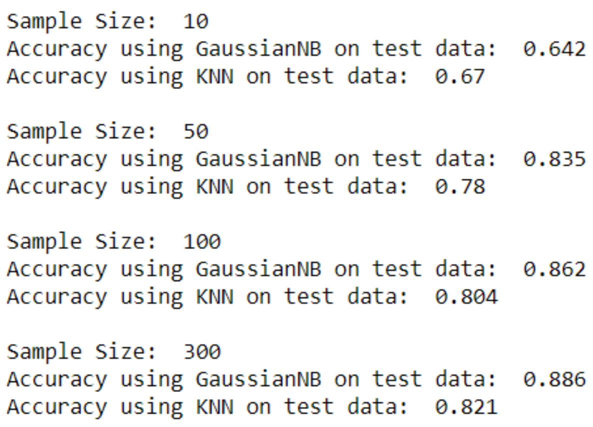
- We observe that accuracy increases as the training sample size increases.
- We also observe that Bayes classifier performs better than nearest neighbor classifier for higher training sample sizes.

Sub-problem A)



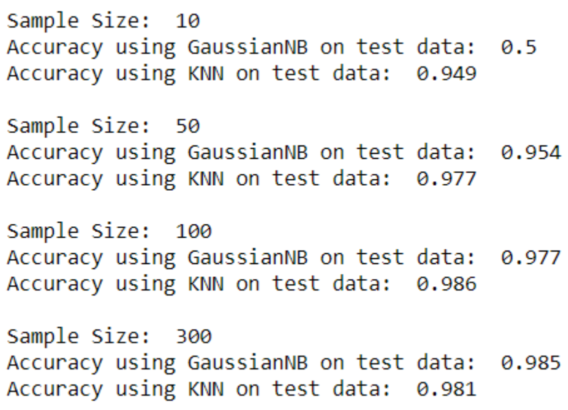
```
Sample Size:  10
Accuracy using GaussianNB on test data:  0.512
Accuracy using KNN on test data:  0.891

Sample Size:  50
Accuracy using GaussianNB on test data:  0.958
Accuracy using KNN on test data:  0.936

Sample Size:  100
Accuracy using GaussianNB on test data:  0.983
Accuracy using KNN on test data:  0.966

Sample Size:  300
Accuracy using GaussianNB on test data:  0.986
Accuracy using KNN on test data:  0.97
```

Sub-problem B)



```
Sample Size:  10
Accuracy using GaussianNB on test data:  0.642
Accuracy using KNN on test data:  0.67

Sample Size:  50
Accuracy using GaussianNB on test data:  0.835
Accuracy using KNN on test data:  0.78

Sample Size:  100
Accuracy using GaussianNB on test data:  0.862
Accuracy using KNN on test data:  0.804

Sample Size:  300
Accuracy using GaussianNB on test data:  0.886
Accuracy using KNN on test data:  0.821
```

Sub-problem C)

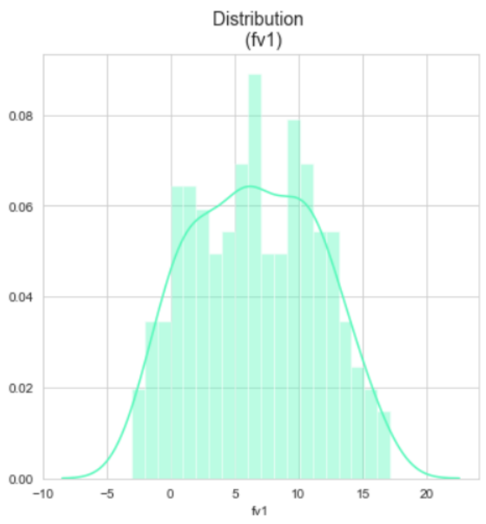


```
Sample Size:  10
Accuracy using GaussianNB on test data:  0.5
Accuracy using KNN on test data:  0.949

Sample Size:  50
Accuracy using GaussianNB on test data:  0.954
Accuracy using KNN on test data:  0.977

Sample Size:  100
Accuracy using GaussianNB on test data:  0.977
Accuracy using KNN on test data:  0.986

Sample Size:  300
Accuracy using GaussianNB on test data:  0.985
Accuracy using KNN on test data:  0.981
```

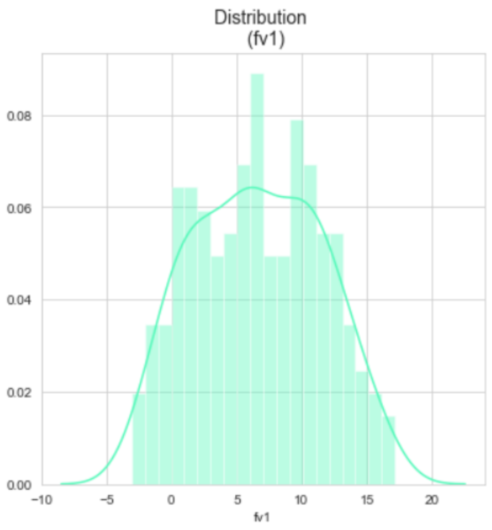
Problem 3)

Exploratory Data Analysis

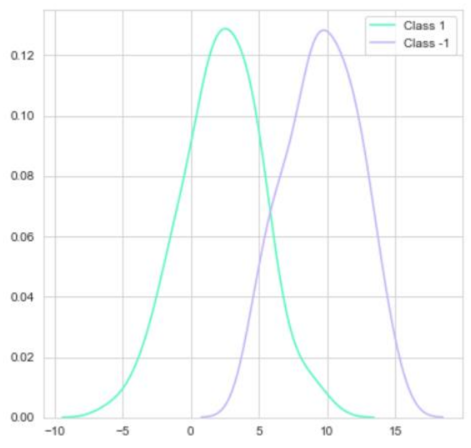
Feature Vector Distribution for part A



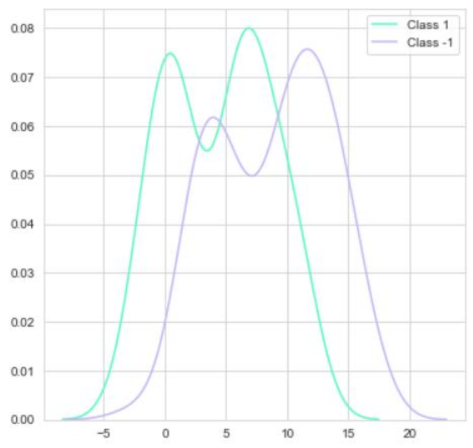
Feature Vector Distribution for part B



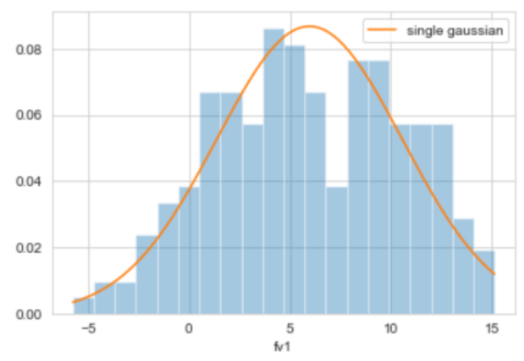
Class Conditioned Probability Distribution for part A



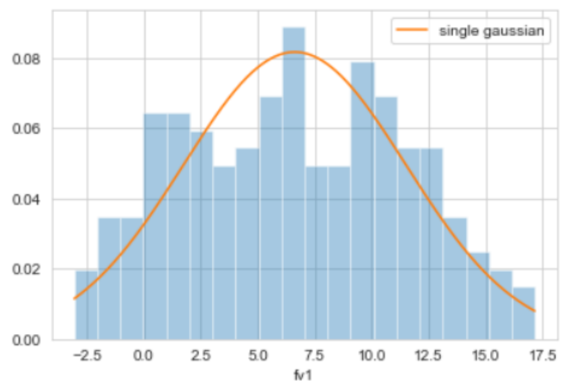
Class Conditioned Probability Distribution for part B



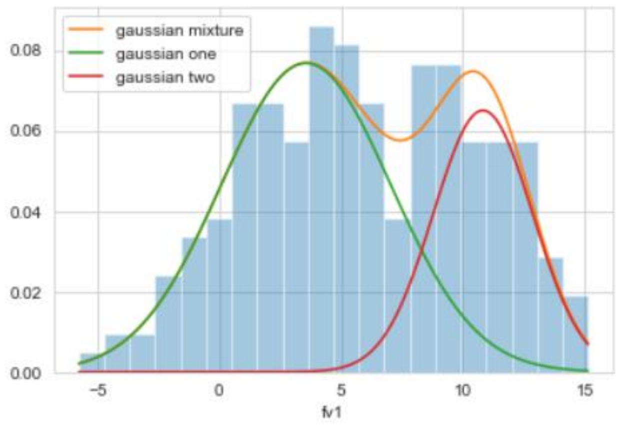
Gaussian fit for part A



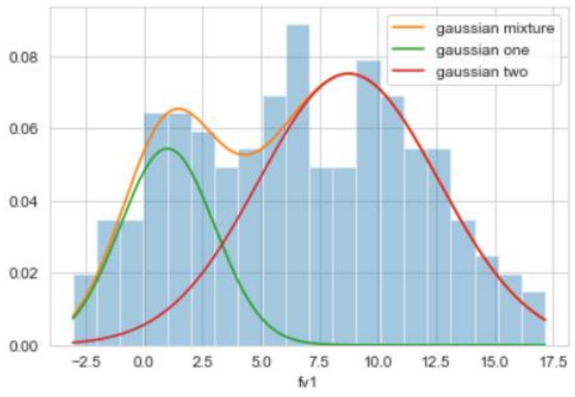
Gaussian fit for part B



Gaussian Mixture Model for part A



Gaussian Mixture Model for part B



Model Analysis

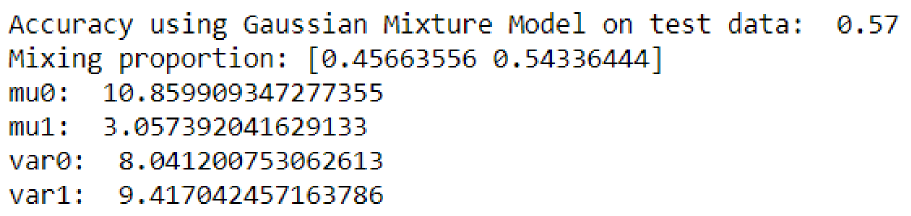
Sub-problem A)

```
Accuracy using Gaussian Mixture Model on test data:  0.88
Mixing proportion: [0.45637323 0.54362677]
mu0:  9.960475304909583
mu1:  2.6120687654509913
var0:  6.293499685869545
var1:  8.927416733677557
```

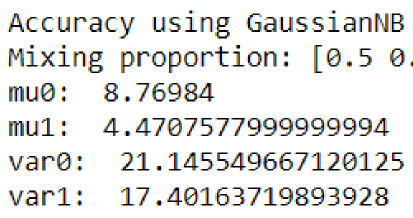
```
Accuracy using GaussianNB on test data:  0.925
Mixing proportion: [0.5 0.5]
mu0:  9.6330929
mu1:  2.2982767
var0:  6.9814808112329505
var1:  8.36347487215047
```

```
Accuracy using KNN on test data:  0.905
```

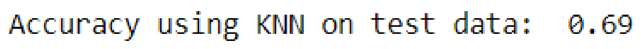
Sub-problem B)



```
Accuracy using Gaussian Mixture Model on test data:  0.57
Mixing proportion: [0.45663556 0.54336444]
mu0:  10.859909347277355
mu1:  3.057392041629133
var0:  8.041200753062613
var1:  9.417042457163786
```



```
Accuracy using GaussianNB on test data:  0.57
Mixing proportion: [0.5 0.5]
mu0:  8.76984
mu1:  4.4707577999999994
var0:  21.145549667120125
var1:  17.40163719893928
```

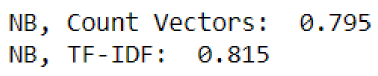


```
Accuracy using KNN on test data:  0.69
```

Problem 4)

Here we fit the Multinomial Naïve Bayes on ‘bag-of-word’ feature vectors and TF-IDF based feature vectors. Best accuracy is obtained with TF-IDF based feature vectors.

Accuracy:



```
NB, Count Vectors:  0.795
NB, TF-IDF:  0.815
```

Code:

```python
def getOnlyWords(wordsList):
    pattern = re.compile('[a-zA-Z]+')
    result = []
    for word in wordsList:
        if pattern.match(word) != None:
          result.append(word)
    return result
```

```python
def getWordsWithoutStop(sentence):
    sentence = sentence.lower()
    words = word_tokenize(sentence)
    withoutStopWords = [word for word in words if not word in stopWords]
    return getOnlyWords(withoutStopWords)
```

```python
# split the dataset into training and validation datasets
train_x, test_x, train_y, test_y = train_test_split(df['text'], df['class'], test_size=.2, random_state=1)

# Now we label encode our target column so that it can be used in machine learning models
encoder = LabelEncoder()
train_y = encoder.fit_transform(train_y)
test_y = encoder.fit_transform(test_y)
```

```python
# create a count vectorizer object and transform the training and validation data using count vectorizer object
count_vectorizer = CountVectorizer(tokenizer = getWordsWithoutStop)
xtrain_count = count_vectorizer.fit_transform(train_x)
xtest_count = count_vectorizer.transform(test_x)
```

```python
# create a tfid vectorizer object and transform the training and validation data using tfid vectorizer object
tfid_vectorizer = TfidfVectorizer(tokenizer = getWordsWithoutStop)
xtrain_tfid = tfid_vectorizer.fit_transform(train_x)
xtest_tfid = tfid_vectorizer.transform(test_x)
tfid_vectorizer.get_feature_names()
print(xtrain_tfid.shape)
print(xtrain_tfid.toarray())
```

```python
def train_model(classifier, feature_vector_train, label, feature_vector_valid):
    # fit the training dataset on the classifier
    classifier.fit(feature_vector_train, label)

    # predict the labels on validation dataset
    predictions = classifier.predict(feature_vector_valid)

    return metrics.accuracy_score(predictions, test_y)
```

```python
# Naive Bayes on Count Vectors
accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_count, train_y, xtest_count)
print ("NB, Count Vectors: ", accuracy)

# Naive Bayes on TF IDF Vectors
accuracy = train_model(naive_bayes.MultinomialNB(), xtrain_tfid, train_y, xtest_tfid)
print ("NB, TF-IDF: ", accuracy)
```