In [17]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [18]:
```
col_names = ['sepal-length','sepal-width','petal-length','petal-width','Class']
df = pd.read_csv('iris.csv',names = col_names)
df
```

Out[18]:

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [19]:
```
from math import sqrt

# calculate the Euclidean distance between two vectors
def euclidean_distance(row1, row2):
    distance = 0.0
    for i in range(len(row1)-1):
        distance += (row1[i] - row2[i])**2
    return sqrt(distance)

# Test distance function
dataset = [[5.1,3.5,1.4,0.2],
[4.9,3.0,1.4,0.2],
[4.7,3.2,1.3,0.2],
[4.6,3.1,1.5,0.2],
[5.0,3.6,1.4,0.2],
[5.4,3.9,1.7,0.4],
[4.6,3.4,1.4,0.3],
[5.0,3.4,1.5,0.2],
[4.4,2.9,1.4,0.2],
[4.9,3.1,1.5,0.1]]
row0 = dataset[0]
for row in dataset:
    distance = euclidean_distance(row0, row)
    print(distance)
```

```
0.0
0.5385164807134502
0.509901951359278
0.648074069840786
0.1414213562373093
0.5830951894845303
0.5099019513592785
0.17320508075688762
0.9219544457292882
0.4582575694955836
```

In [20]:
```
# Locate the most similar neighbors
def get_neighbors(train, test_row, num_neighbors):
    distances = list()
    for train_row in train:
        dist = euclidean_distance(test_row, train_row)
        distances.append((train_row, dist))
    distances.sort(key=lambda tup: tup[1])
    neighbors = list()
    for i in range(num_neighbors):
        neighbors.append(distances[i][0])
    return neighbors

neighbors = get_neighbors(dataset, dataset[0], 3)
for neighbor in neighbors:
    print(neighbor)
```

```
[5.1, 3.5, 1.4, 0.2]
[5.0, 3.6, 1.4, 0.2]
[5.0, 3.4, 1.5, 0.2]
```

In [21]:
```
# Make a classification prediction with neighbors
def predict_classification(train, test_row, num_neighbors):
    neighbors = get_neighbors(train, test_row, num_neighbors)
    output_values = [row[-1] for row in neighbors]
    prediction = max(set(output_values), key=output_values.count)
    return prediction

prediction = predict_classification(dataset, dataset[0], 3)
print('Expected %d, Got %d.' % (dataset[0][-1], prediction))
```

```
Expected 0, Got 0.
```

In [22]:
```
from sklearn.model_selection import train_test_split
X = df[['sepal-length','sepal-width','petal-length','petal-width']]
y = df[['Class']]
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.4)
```

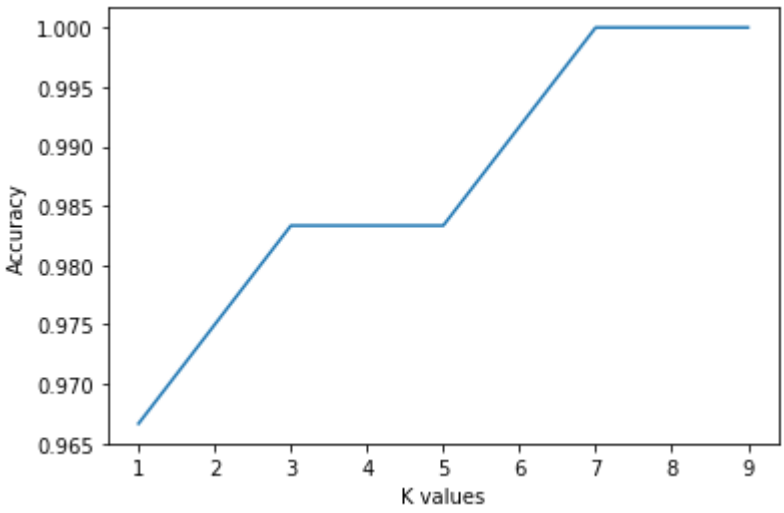
In [111]:
```
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
k_range = range(1,10,2)
scores=[]
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    scores.append(metrics.accuracy_score(y_test, y_pred))

print(scores)
```

```
[0.9666666666666667, 0.9833333333333333, 0.9833333333333333, 1.0, 1.0]

C:\Users\Nupur goel\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:1
79: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
C:\Users\Nupur goel\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:1
79: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
C:\Users\Nupur goel\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:1
79: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
C:\Users\Nupur goel\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:1
79: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
C:\Users\Nupur goel\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:1
79: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

In [112]:
```
plt.plot(k_range, scores)
plt.xlabel('K values')
plt.ylabel('Accuracy')
plt.show()
```



In [ ]:
```
#This graph shows that we are getting accuracy of 0.983 when k is between 3 to 5 but whe
#gives the best accuracy so knee point must be floor then k must be 7
```

In [23]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import average_precision_score
from sklearn.naive_bayes import GaussianNB
df = pd.read_csv('wine.csv')
X = df.drop('Proline', axis=1)
y = df['Proline']

Accuracy = []
prec_mat = []
split_size = [0.5, 0.4, 0.3, 0.2, 0.1]
for i in range(0,5):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = split_size[i],
    gnb = GaussianNB()
    gnb.fit(X_train, y_train)

    y_pred = gnb.predict(X_test)

    Accuracy.append(metrics.accuracy_score(y_test, y_pred)*100)
    y_true = y_test.values
    y_true = y_true.reshape(-1,1)
    prec_mat.insert(i,metrics.classification_report(y_true, y_pred, digits = 3))
```

```
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Nupur goel\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:
1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set t
o 0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Nupur goel\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:
1245: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to
0.0 in labels with no true samples. Use `zero_division` parameter to control this b
ehavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Nupur goel\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:
1245: UndefinedMetricWarning: Precision and F-score are ill-defined and being set t
o 0.0 in labels with no predicted samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Nupur goel\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:
1245: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to
0.0 in labels with no true samples. Use `zero_division` parameter to control this b
ehavior.
```

In [29]:
```
for i in range(0,5):
    print(prec_mat[i])
```

```
              precision    recall  f1-score   support

         278       0.000     0.000     0.000         1
         290       0.000     0.000     0.000         1
         312       0.000     0.000     0.000         1
         325       0.000     0.000     0.000         1
         345       0.000     0.000     0.000         1
         352       0.000     0.000     0.000         1
         355       0.000     0.000     0.000         1
         365       0.000     0.000     0.000         1
         372       0.000     0.000     0.000         1
         378       0.000     0.000     0.000         1
         380       0.000     0.000     0.000         2
         410       0.000     0.000     0.000         1
         415       0.000     0.000     0.000         1
         420       0.000     0.000     0.000         1
         428       0.000     0.000     0.000         1
         434       0.000     0.000     0.000         1
         450       0.000     0.000     0.000         1
```

In [25]:
```
plt.show(prec_mat[i])
```

In [28]:
```
import pandas as pd
df = pd.read_csv('train.csv', delimiter =',')
df
```

Out[28]:

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2
...	...	...	...	...
156055	156056	8544	Hearst 's	2
156056	156057	8544	forced avuncular chortles	1
156057	156058	8544	avuncular chortles	3
156058	156059	8544	avuncular	2
156059	156060	8544	chortles	2

156060 rows × 4 columns

In [29]:
```
df.head()
```

Out[29]:

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2

In [31]:
```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer
#tokenizer to remove unwanted elements from out data like symbols and numbers
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
cv = CountVectorizer(lowercase=True,stop_words='english',ngram_range = (1,1),tokenizer =
text_counts= cv.fit_transform(df['Phrase'])
```

In [ ]:
```
Accuracy = []
prec_mat = []
split_size = [0.5, 0.4, 0.3, 0.2, 0.1]
for i in range(0,5):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = split_size[i],
    gnb = GaussianNB()
    gnb.fit(X_train, y_train)

    y_pred = gnb.predict(X_test)

    Accuracy.append(metrics.accuracy_score(y_test, y_pred)*100)
    y_true = y_test.values
    y_true = y_true.reshape(-1,1)
    prec_mat.insert(i,metrics.classification_report(y_true, y_pred, digits = 3))
```

In [32]:
```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
split_size = [0.5, 0.4, 0.3, 0.2, 0.1]
for i in range(0,5):
    X_train, X_test, y_train, y_test = train_test_split(
    text_counts, df['Sentiment'], test_size = split_size[i], random_state=0)
    clf = MultinomialNB().fit(X_train, y_train)
    predicted= clf.predict(X_test)
    print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predicted))
```

```
MultinomialNB Accuracy: 0.5956427015250545
MultinomialNB Accuracy: 0.6035979751377675
MultinomialNB Accuracy: 0.6095305224486308
MultinomialNB Accuracy: 0.6106946046392413
MultinomialNB Accuracy: 0.6144431628860695
```

In [39]:
```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
df = pd.read_csv('train.csv')
def func(text):
    tokens = word_tokenize(text)
    return tokens
for i in df['Phrase'].index:
    df['Phrase'][i] = func(df['Phrase'][i])
df
```

```
[nltk_data] Downloading package punkt to C:\Users\Nupur
[nltk_data]     goel\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
<ipython-input-39-2bcdcdb94322>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['Phrase'][i] = func(df['Phrase'][i])
```

Out[39]:

	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	[A, series, of, escapades, demonstrating, the,...	1
1	2	1	[A, series, of, escapades, demonstrating, the,...	2
2	3	1	[A, series]	2
3	4	1	[A]	2
4	5	1	[series]	2
...	...	...	...	...
156055	156056	8544	[Hearst, 's]	2
156056	156057	8544	[forced, avuncular, chortles]	1
156057	156058	8544	[avuncular, chortles]	3
156058	156059	8544	[avuncular]	2
156059	156060	8544	[chortles]	2

156060 rows × 4 columns

In [ ]: