In [1]:
```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
%matplotlib inline
```

Data Preprocessing

In [2]:
```
raw_data = pd.read_csv('wine_modified.csv')
```

In [3]:
```
# Question 1
raw_data = raw_data[raw_data.isnull().sum(axis=1)<8]
raw_data = raw_data[raw_data['class'].notnull()]
raw_data.shape[0]
```

Out[3]:
```
154
```

In [4]:
```
# Question 2
raw_data.isnull().sum(axis=0)
```

Out[4]:
```
class                   0
Alcohol                 0
Malic acid              0
Ash                    95
Alcalinity of ash       0
Magnesium               9
Total phenols           0
Flavanoids             35
Nonflavanoid phenols    0
Proanthocyanins         0
Color intensity         0
Hue                     0
OD280/OD315             0
Proline                 0
dtype: int64
```

In [5]:
```
del raw_data['Ash']
```

In [6]:
```
avg = raw_data['Magnesium'].mean()
raw_data['Magnesium'] = raw_data['Magnesium'].fillna(avg)
```

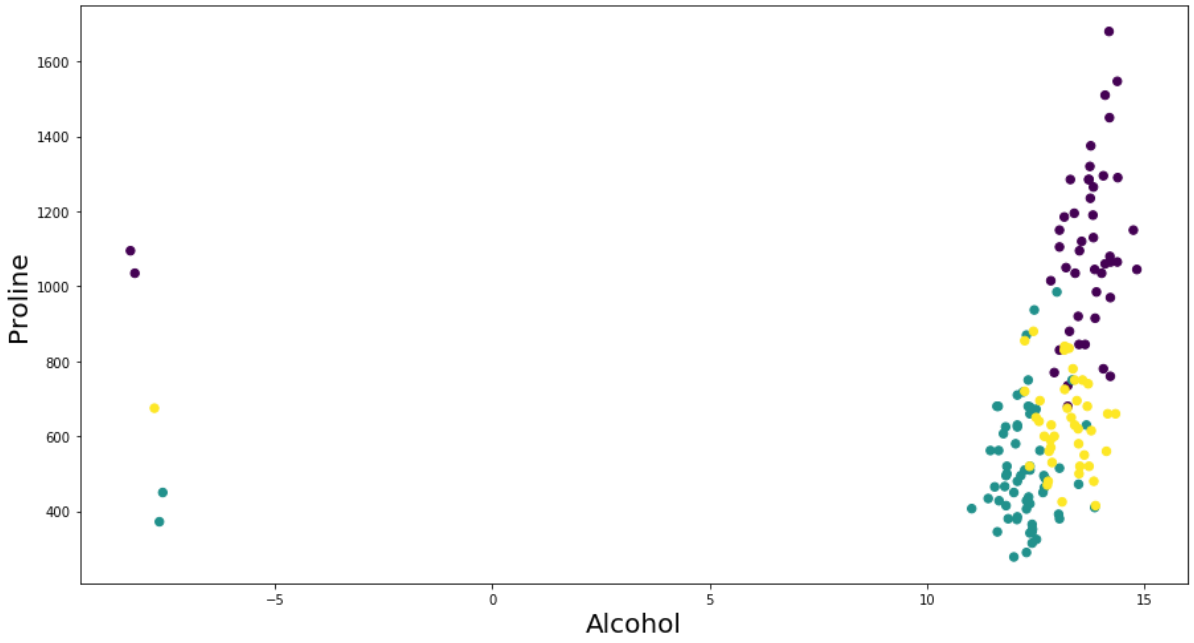
In [7]:
```
avg = raw_data['Flavanoids'].mean()
raw_data['Flavanoids'] = raw_data['Flavanoids'].fillna(avg)
```

In [8]:
```
raw_data.std(axis=0)
```

Out[8]:
```
class                         0.766522
Alcohol                       3.804067
Malic acid                    1.116005
Alcalinity of ash             3.456794
Magnesium                    14.440377
Total phenols                 0.617237
Flavanoids                    0.873573
Nonflavanoid phenols          0.127083
Proanthocyanins               0.587671
Color intensity               2.325204
Hue                           0.229412
OD280/OD315                   0.723261
Proline                     303.033368
dtype: float64
```

In [9]:
```
#Question 3
plt.figure(figsize=(15,8))
plt.xlabel('Alcohol', fontsize=20)
plt.ylabel('Proline', fontsize=20)
plt.scatter(raw_data['Alcohol'], raw_data['Proline'], s=40,
c=raw_data['class'])
```

Out[9]:
```
<matplotlib.collections.PathCollection at 0x1006c5cf8>
```



In [10]:
```
# Outliers/Incorrect values Alcohol
raw_data = raw_data[raw_data['Alcohol'] > 0]
```

In [11]:
```
# Outliers/Incorrect values Proline
mean = raw_data['Proline'].mean()
std = raw_data['Proline'].std()
raw_data = raw_data[raw_data['Proline'].between(mean-4*std,mean+4*std)]
```

In [12]:
```
raw_data
```

Out[12]:

	class	Alcohol	Malic acid	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Pro
0	1.0	14.23	1.71	15.6	127.000000	2.80	3.060000	0.28	2.29
1	1.0	13.20	1.78	11.2	100.000000	2.65	2.760000	0.26	1.28
2	1.0	13.16	2.36	18.6	101.000000	2.80	3.240000	0.30	2.81
4	1.0	13.24	2.59	21.0	118.000000	2.80	2.690000	0.39	1.82
5	1.0	14.20	1.76	15.2	112.000000	3.27	1.937983	0.34	1.97
6	1.0	14.39	1.87	14.6	96.000000	2.50	2.520000	0.30	1.98
7	1.0	14.06	2.15	17.6	121.000000	2.60	2.510000	0.31	1.25
8	1.0	14.83	1.64	14.0	97.000000	2.80	2.980000	0.29	1.98
9	1.0	13.86	1.35	16.0	98.000000	2.98	1.937983	0.22	1.85
10	1.0	14.10	2.16	18.0	99.496552	2.95	1.937983	0.22	2.38
12	1.0	13.75	1.73	16.0	89.000000	2.60	2.760000	0.29	1.81
13	1.0	14.75	1.73	11.4	91.000000	3.10	3.690000	0.43	2.81
14	1.0	14.38	1.87	12.0	102.000000	3.30	1.937983	0.29	2.96
17	1.0	13.83	1.57	20.0	115.000000	2.95	3.400000	0.40	1.72
18	1.0	14.19	1.59	16.5	108.000000	3.30	1.937983	0.32	1.86
19	1.0	13.64	3.10	15.2	116.000000	2.70	1.937983	0.17	1.66
20	1.0	14.06	1.63	16.0	126.000000	3.00	3.170000	0.24	2.10
21	1.0	12.93	3.80	18.6	102.000000	2.41	2.410000	0.25	1.98
23	1.0	12.85	1.60	17.8	95.000000	2.48	2.370000	0.26	1.46
24	1.0	13.50	1.81	20.0	96.000000	2.53	2.610000	0.28	1.66
25	1.0	13.05	2.05	25.0	124.000000	2.63	2.680000	0.47	1.92
26	1.0	13.39	1.77	16.1	93.000000	2.85	1.937983	0.34	1.45
27	1.0	13.30	1.72	17.0	94.000000	2.40	2.190000	0.27	1.35
28	1.0	13.87	1.90	19.4	107.000000	2.95	2.970000	0.37	1.76
29	1.0	14.02	1.68	16.0	96.000000	2.65	2.330000	0.26	1.98
30	1.0	13.73	1.50	22.5	101.000000	3.00	3.250000	0.29	2.38
33	1.0	13.76	1.53	19.5	132.000000	2.95	2.740000	0.50	1.35
34	1.0	13.51	1.80	19.0	110.000000	2.35	2.530000	0.29	1.54
35	1.0	13.48	1.81	20.5	100.000000	2.70	1.937983	0.26	1.86
36	1.0	13.28	1.64	15.5	110.000000	2.60	2.680000	0.34	1.36
...	...	...	...	...	...	...	...	...	...

	class	Alcohol	Malic acid	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Pro
143	3.0	13.62	4.95	20.0	92.000000	2.00	1.937983	0.47	1.02
144	3.0	12.25	3.88	18.5	112.000000	1.38	1.937983	0.29	1.14
145	3.0	13.16	3.57	21.0	102.000000	1.50	0.550000	0.43	1.30
146	3.0	13.88	5.04	20.0	80.000000	0.98	0.340000	0.40	0.68
148	3.0	13.32	3.24	21.5	92.000000	1.93	0.760000	0.45	1.25
150	3.0	13.50	3.12	24.0	123.000000	1.40	1.570000	0.22	1.25
151	3.0	12.79	2.67	22.0	112.000000	1.48	1.937983	0.24	1.26
152	3.0	13.11	1.90	25.5	99.496552	2.20	1.280000	0.26	1.56
153	3.0	13.23	3.30	18.5	98.000000	1.80	0.830000	0.61	1.87
154	3.0	12.58	1.29	20.0	103.000000	1.48	0.580000	0.53	1.40
155	3.0	13.17	5.19	22.0	93.000000	1.74	0.630000	0.61	1.55
156	3.0	13.84	4.12	19.5	89.000000	1.80	0.830000	0.48	1.56
157	3.0	12.45	3.03	27.0	97.000000	1.90	0.580000	0.63	1.14
158	3.0	14.34	1.68	25.0	98.000000	2.80	1.310000	0.53	2.70
159	3.0	13.48	1.67	22.5	89.000000	2.60	1.937983	0.52	2.29
160	3.0	12.36	3.83	21.0	88.000000	2.30	0.920000	0.50	1.04
161	3.0	13.69	3.26	20.0	107.000000	1.83	0.560000	0.50	0.80
162	3.0	12.85	3.27	22.0	106.000000	1.65	1.937983	0.60	0.96
164	3.0	13.78	2.76	22.0	99.496552	1.35	0.680000	0.41	1.03
165	3.0	13.73	4.36	22.5	88.000000	1.28	0.470000	0.52	1.15
166	3.0	13.45	3.70	23.0	111.000000	1.70	0.920000	0.43	1.46
168	3.0	13.58	2.58	24.5	105.000000	1.55	1.937983	0.39	1.54
169	3.0	13.40	4.60	25.0	112.000000	1.98	0.960000	0.27	1.11
171	3.0	12.77	2.39	19.5	86.000000	1.39	0.510000	0.48	0.64
172	3.0	14.16	2.51	20.0	91.000000	1.68	0.700000	0.44	1.24
173	3.0	13.71	5.65	20.5	95.000000	1.68	0.610000	0.52	1.06
174	3.0	13.40	3.91	23.0	102.000000	1.80	1.937983	0.43	1.41
175	3.0	13.27	4.28	20.0	120.000000	1.59	0.690000	0.43	1.35
176	3.0	13.17	2.59	20.0	120.000000	1.65	0.680000	0.53	1.46
177	3.0	14.13	4.10	24.5	96.000000	2.05	1.937983	0.56	1.35

149 rows × 13 columns

Decision Tree

In [13]:
```
train_data = pd.read_csv('wine_train_data.csv')
train_labels = pd.read_csv('wine_train_labels.csv')
val_data =  pd.read_csv('wine_val_data.csv')
val_labels = pd.read_csv('wine_val_labels.csv')
test_data = pd.read_csv('wine_test_data.csv')
test_labels = pd.read_csv('wine_test_labels.csv')
```

In [14]:
```
total_data = pd.concat([train_data, val_data], ignore_index=True)
total_labels = pd.concat([train_labels, val_labels], ignore_index=True)
```

In [15]:
```
#Question 4
for crit in ['gini', 'entropy']:
    model = DecisionTreeClassifier(criterion=crit)
    model.fit(train_data, train_labels)
    pred = model.predict(val_data)
    print ('Validation accuracy for criterion ' + crit + ' = ' + str(acc
uracy_score(val_labels, pred)))
```

```
Validation accuracy for criterion gini = 0.897435897436
Validation accuracy for criterion entropy = 0.948717948718
```

In [16]:
```
model = DecisionTreeClassifier(criterion='entropy')
model.fit(total_data, total_labels)
pred = model.predict(test_data)
print ('Test accuracy for criterion ' + 'entropy' + ' = ' + str(accuracy
_score(test_labels, pred)))
```

```
Test accuracy for criterion entropy = 0.794871794872
```

In [17]:
```
# Question 5
for sample_split_size in [2,5,10,20]:
    model = DecisionTreeClassifier(criterion='entropy', min_samples_spli
t=sample_split_size)
    model.fit(train_data, train_labels)
    pred = model.predict(val_data)
    print ('Validation accuracy for min_sample_split ' + str(sample_spli
t_size) + ' = ' + str(accuracy_score(val_labels, pred)))
```

```
Validation accuracy for min_sample_split 2 = 0.948717948718
Validation accuracy for min_sample_split 5 = 0.974358974359
Validation accuracy for min_sample_split 10 = 0.923076923077
Validation accuracy for min_sample_split 20 = 0.948717948718
```

In [18]:
```
model = DecisionTreeClassifier(criterion='entropy', min_samples_split=5)
model.fit(total_data, total_labels)
pred = model.predict(test_data)
print ('Test accuracy for criterion ' + 'entropy' + ' = ' + str(accuracy
_score(test_labels, pred)))
```

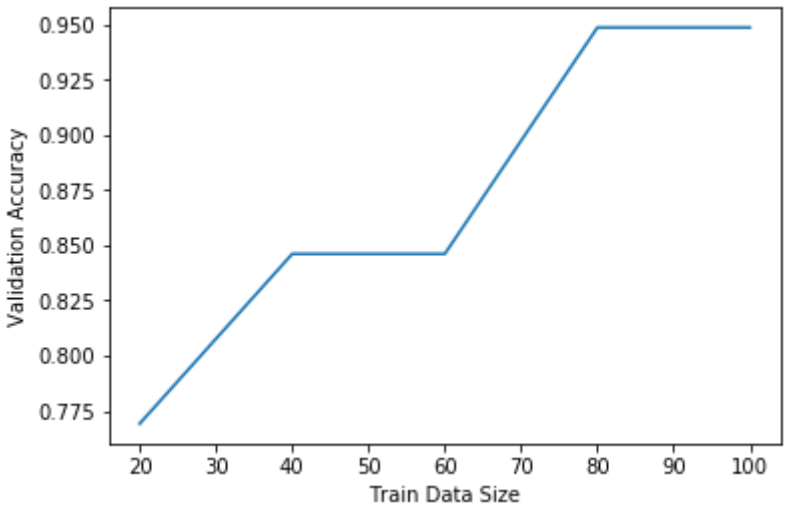
```
Test accuracy for criterion entropy = 0.820512820513
```

In [19]:
```
# Question 6
result=[]
for data_size in [20,40,60,80, 100]:
    model = DecisionTreeClassifier(criterion='entropy', min_samples_spli
t=5)
    model.fit(train_data[:data_size], train_labels[:data_size])
    pred = model.predict(val_data)
    result.append(accuracy_score(val_labels, pred))
    print ('Validation accuracy for data size ' + str(data_size) + ' = '
 + str(accuracy_score(val_labels, pred)))
```

```
Validation accuracy for data size 20 = 0.769230769231
Validation accuracy for data size 40 = 0.846153846154
Validation accuracy for data size 60 = 0.846153846154
Validation accuracy for data size 80 = 0.948717948718
Validation accuracy for data size 100 = 0.948717948718
```

In [20]:
```
plt.ylabel('Validation Accuracy')
plt.xlabel('Train Data Size')
plt.plot([20,40,60,80, 100], result)
```

Out[20]:
```
[<matplotlib.lines.Line2D at 0x112c31eb8>]
```



Nearest Neighbor

In [492]:
```
# Normalize Data
mean = total_data.mean(axis=0)
std = total_data.std(axis=0)
total_data = (total_data - mean)/std
train_data = (train_data - mean)/std
val_data = (val_data - mean)/std
test_data = (test_data-mean)/std
```

In [23]:
```
total_data
```

Out[23]:

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proa
0	12.53	5.51	2.64	25.0	96	1.79	0.60	0.63	1.10
1	13.49	1.66	2.24	24.0	87	1.88	1.84	0.27	1.03
2	13.75	1.73	2.41	16.0	89	2.60	2.76	0.29	1.81
3	12.29	2.83	2.22	18.0	88	2.45	2.25	0.25	1.99
4	14.10	2.16	2.30	18.0	105	2.95	3.32	0.22	2.38
5	11.66	1.88	1.92	16.0	97	1.61	1.57	0.34	1.15
6	13.45	3.70	2.60	23.0	111	1.70	0.92	0.43	1.46
7	13.73	4.36	2.26	22.5	88	1.28	0.47	0.52	1.15
8	12.69	1.53	2.26	20.7	80	1.38	1.46	0.58	1.62
9	12.42	4.43	2.73	26.5	102	2.20	2.13	0.43	1.71
10	14.22	1.70	2.30	16.3	118	3.20	3.00	0.26	2.03
11	13.05	1.73	2.04	12.4	92	2.72	3.27	0.17	2.91
12	13.17	5.19	2.32	22.0	93	1.74	0.63	0.61	1.55
13	13.11	1.01	1.70	15.0	78	2.98	3.18	0.26	2.28
14	12.37	0.94	1.36	10.6	88	1.98	0.57	0.28	0.42
15	11.87	4.31	2.39	21.0	82	2.86	3.03	0.21	2.91
16	11.84	2.89	2.23	18.0	112	1.72	1.32	0.43	0.95
17	12.42	1.61	2.19	22.5	108	2.00	2.09	0.34	1.61
18	14.38	1.87	2.38	12.0	102	3.30	3.64	0.29	2.96
19	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35
20	13.83	1.57	2.62	20.0	115	2.95	3.40	0.40	1.72
21	12.34	2.45	2.46	21.0	98	2.56	2.11	0.34	1.31
22	11.82	1.47	1.99	20.8	86	1.98	1.60	0.30	1.53
23	12.08	1.83	2.32	18.5	81	1.60	1.50	0.52	1.64
24	13.73	1.50	2.70	22.5	101	3.00	3.25	0.29	2.38
25	14.39	1.87	2.45	14.6	96	2.50	2.52	0.30	1.98
26	12.77	2.39	2.28	19.5	86	1.39	0.51	0.48	0.64
27	13.05	1.77	2.10	17.0	107	3.00	3.00	0.28	2.03
28	13.56	1.71	2.31	16.2	117	3.15	3.29	0.34	2.34
29	13.82	1.75	2.42	14.0	111	3.88	3.74	0.32	1.87
...	...	...	...	...	...	...	...	...	...

	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proa
109	13.08	3.90	2.36	21.5	113	1.41	1.39	0.34	1.14
110	12.29	1.41	1.98	16.0	85	2.55	2.50	0.29	1.77
111	11.84	0.89	2.58	18.0	94	2.20	2.21	0.22	2.35
112	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46
113	12.25	4.72	2.54	21.0	89	1.38	0.47	0.53	0.80
114	13.64	3.10	2.56	15.2	116	2.70	3.03	0.17	1.66
115	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81
116	14.19	1.59	2.48	16.5	108	3.30	3.93	0.32	1.86
117	11.82	1.72	1.88	19.5	86	2.50	1.64	0.37	1.42
118	13.48	1.67	2.64	22.5	89	2.60	1.10	0.52	2.29
119	13.86	1.51	2.67	25.0	86	2.95	2.86	0.21	1.87
120	14.30	1.92	2.72	20.0	120	2.80	3.14	0.33	1.97
121	12.43	1.53	2.29	21.5	86	2.74	3.15	0.39	1.77
122	11.62	1.99	2.28	18.0	98	3.02	2.26	0.17	1.35
123	13.41	3.84	2.12	18.8	90	2.45	2.68	0.27	1.48
124	13.51	1.80	2.65	19.0	110	2.35	2.53	0.29	1.54
125	11.96	1.09	2.30	21.0	101	3.38	2.14	0.13	1.65
126	12.51	1.73	1.98	20.5	85	2.20	1.92	0.32	1.48
127	13.34	0.94	2.36	17.0	110	2.53	1.30	0.55	0.42
128	13.39	1.77	2.62	16.1	93	2.85	2.94	0.34	1.45
129	12.60	1.34	1.90	18.5	88	1.45	1.36	0.29	1.35
130	11.79	2.13	2.78	28.5	92	2.13	2.24	0.58	1.76
131	14.38	3.59	2.28	16.0	102	3.25	3.17	0.27	2.19
132	12.85	1.60	2.52	17.8	95	2.48	2.37	0.26	1.46
133	13.72	1.43	2.50	16.7	108	3.40	3.67	0.19	2.04
134	14.10	2.02	2.40	18.8	103	2.75	2.92	0.32	2.38
135	12.70	3.55	2.36	21.5	106	1.70	1.20	0.17	0.84
136	12.00	1.51	2.42	22.0	86	1.45	1.25	0.50	1.63
137	12.88	2.99	2.40	20.0	104	1.30	1.22	0.24	0.83
138	13.84	4.12	2.38	19.5	89	1.80	0.83	0.48	1.56

139 rows × 13 columns

In [493]:
```
#Question 7
model = KNeighborsClassifier(n_neighbors=3)
model.fit(total_data, total_labels)
pred = model.predict(test_data)
print('Test Accuracy = ' + str(accuracy_score(test_labels, pred)))
```

```
Test Accuracy = 0.871794871795

//anaconda/envs/py35/lib/python3.5/site-packages/ipykernel/__main__.py:
3: DataConversionWarning: A column-vector y was passed when a 1d array
 was expected. Please change the shape of y to (n_samples, ), for examp
le using ravel().
  app.launch_new_instance()
```

In [494]:
```
# Question 8
dist_set = ['manhattan','euclidean', 'chebyshev']
for dist in dist_set:
    model = KNeighborsClassifier(n_neighbors=3, metric=dist)
    model.fit(train_data, train_labels)
    pred = model.predict(val_data)
    print('Validation Accuracy for ' + dist + ' = ' +
str(accuracy_score(val_labels, pred)))
```

```
Validation Accuracy for manhattan = 0.948717948718
Validation Accuracy for euclidean = 0.923076923077
Validation Accuracy for chebyshev = 0.923076923077

//anaconda/envs/py35/lib/python3.5/site-packages/ipykernel/__main__.py:
5: DataConversionWarning: A column-vector y was passed when a 1d array
 was expected. Please change the shape of y to (n_samples, ), for examp
le using ravel().
```

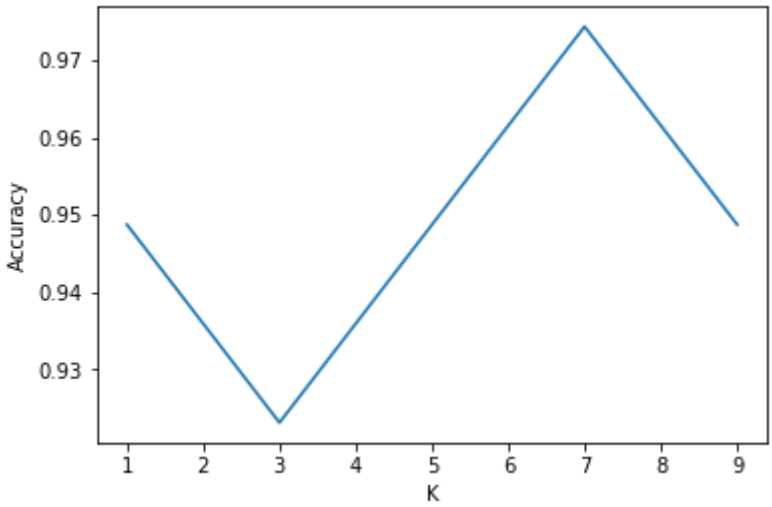
In [495]:
```
model = KNeighborsClassifier(n_neighbors=3, metric='manhattan')
model.fit(total_data, total_labels)
pred = model.predict(test_data)
print('Test Accuracy = ' + str(accuracy_score(test_labels, pred)))
```

```
Test Accuracy = 0.974358974359

//anaconda/envs/py35/lib/python3.5/site-packages/ipykernel/__main__.py:
2: DataConversionWarning: A column-vector y was passed when a 1d array
 was expected. Please change the shape of y to (n_samples, ), for examp
le using ravel().
  from ipykernel import kernelapp as app
```

In [496]:
```
#Question 9
result = []
k_set = [1,3,5,7,9]
for k in k_set:
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(train_data, train_labels)
    pred = model.predict(val_data)
    result.append(accuracy_score(val_labels, pred))
print(result)
plt.ylabel('Accuracy')
plt.xlabel('K')
plt.plot(k_set, result)
```

```
[0.94871794871794868, 0.92307692307692313, 0.94871794871794868, 0.97435
897435897434, 0.94871794871794868]

//anaconda/envs/py35/lib/python3.5/site-packages/ipykernel/__main__.py:
8: DataConversionWarning: A column-vector y was passed when a 1d array
 was expected. Please change the shape of y to (n_samples, ), for examp
le using ravel().
```

Out[496]:
```
[<matplotlib.lines.Line2D at 0x11c93b358>]
```



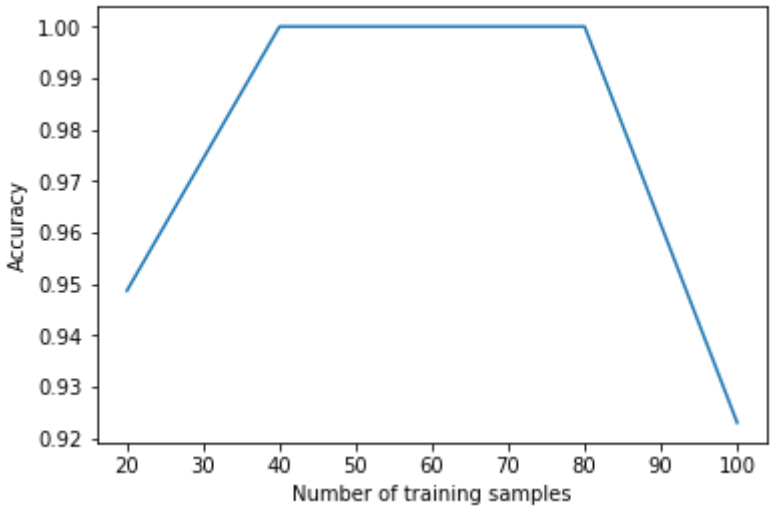
In [497]:
```
model = KNeighborsClassifier(n_neighbors=7)
model.fit(total_data, total_labels)
pred = model.predict(test_data)
print('Test Accuracy = ' + str(accuracy_score(test_labels, pred)))
```

```
Test Accuracy = 0.923076923077

//anaconda/envs/py35/lib/python3.5/site-packages/ipykernel/__main__.py:
2: DataConversionWarning: A column-vector y was passed when a 1d array
 was expected. Please change the shape of y to (n_samples, ), for examp
le using ravel().
  from ipykernel import kernelapp as app
```

In [498]:
```
#Question 10
model = KNeighborsClassifier(n_neighbors=3)
result = []
train_size = [20,40,60,80,100]
for s in train_size:
    model.fit(train_data[:s], train_labels[:s])
    pred = model.predict(val_data)
    result.append(accuracy_score(val_labels, pred))
print(result)
plt.ylabel('Accuracy')
plt.xlabel('Number of training samples')
plt.plot(train_size, result)
```

```
[0.94871794871794868, 1.0, 1.0, 1.0, 0.92307692307692313]

//anaconda/envs/py35/lib/python3.5/site-packages/ipykernel/__main__.py:
8: DataConversionWarning: A column-vector y was passed when a 1d array
 was expected. Please change the shape of y to (n_samples, ), for examp
le using ravel().
```

Out[498]:
```
[<matplotlib.lines.Line2D at 0x11c246c50>]
```



In [ ]:
```

```