Problem 6.1

(a) Decision region for 1-NN and 3-NN

In [569]:

```
#import libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

In [570]:

```
df =pd.DataFrame()
```

In [571]:

```
#Given points x,y
x = [1,0,0,-1,0,0,-2]
y = [0,1,-1,0,2,-2,0]
```

In [572]:

```
df = pd.DataFrame({'X': [1,0,0,-1,0,0,-2],
                   'Y': [0,1,-1,0,2,-2,0]})
```

In [573]:

```
#labels
Y = [-1,-1,-1,-1,1,1,1]
```

In [574]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [575]:

```
#for KNN with neighbours 1
knn = KNeighborsClassifier(n_neighbors=1)
```

In [576]:

```
knn.fit(df,Y)
```

Out[576]:

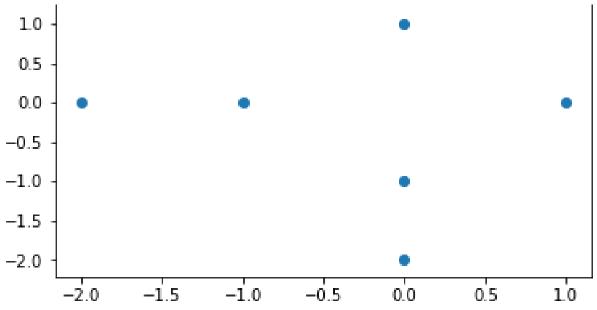
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')
```

In [577]:

```
plt.scatter(x,y)
```

Out[577]:

```
<matplotlib.collections.PathCollection at 0x1ed81021d30>
```



In [578]:

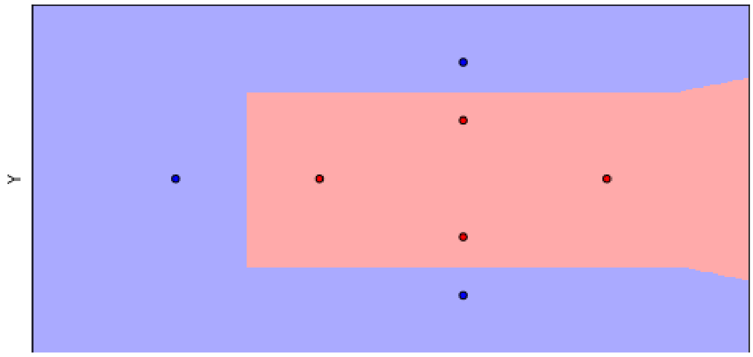
```
df
```

Out[578]:

	X	Y
0	1	0
1	0	1
2	0	-1
3	-1	0
4	0	2
5	0	-2
6	-2	0

In [579]:

```
#1-NN ecision regions
h = .02
x_min, x_max = df['X'].min() - 1, df['X'].max() + 1
y_min, y_max = df['Y'].min() - 1, df['Y'].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = knn.predict(np.c_[xx.ravel(), yy.ravel()])


Z = Z.reshape(xx.shape)
plt.figure(1, figsize=(8, 4))
plt.set_cmap(plt.cm.Paired)
plt.pcolormesh(xx, yy, Z,cmap=cmap_light)


plt.scatter(df['X'], df['Y'],c=Y,cmap=cmap_bold,edgecolor='k', s=20)
plt.xlabel('X')
plt.ylabel('Y')

plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xticks(())
plt.yticks(())

plt.show()
```



In [580]:

```
#For KNN with neighbours 3
knn = KNeighborsClassifier(n_neighbors=3)
```

In [581]:

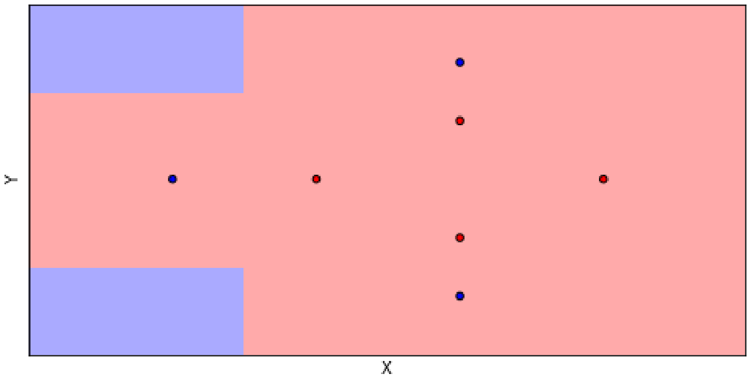
```
knn.fit(df,Y)
```

Out[581]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=3, p=2,
           weights='uniform')
```

In [582]:

```
h = .02

x_min, x_max = df['X'].min() - 1, df['X'].max() + 1
y_min, y_max = df['Y'].min() - 1, df['Y'].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = knn.predict(np.c_[xx.ravel(), yy.ravel()])


Z = Z.reshape(xx.shape)
plt.figure(1, figsize=(8, 4))
plt.set_cmap(plt.cm.Paired)
plt.pcolormesh(xx, yy, Z,cmap=cmap_light)


plt.scatter(df['X'], df['Y'],c=Y,cmap=cmap_bold,edgecolor='k', s=20 )
plt.xlabel('X')
plt.ylabel('Y')

plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xticks(())
plt.yticks(())

plt.show()
```



(b) Transform to non-linear and plot the decision regions

In [583]:

```
#transformation
X1 =[]
X2=[]
for index in df.values:
    x1 = np.sqrt(index[0]**2 + index[1]**2)
    x2 = np.arctan(index[1]/index[0])
    X1.append(x1)
    X1.append(x1)
    X2.append(x2)
```

```
C:\Users\kashi\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: RuntimeWarning: divide by zero
encountered in longlong_scalars
```

In [584]:

```
print(X1)
```

```
[1.0, 1.0, 1.0, 1.0, 2.0, 2.0, 2.0]
```

In [585]:

```
print(X2)
```

```
[0.0, 1.5707963267948966, -1.5707963267948966, -0.0, 1.5707963267948966, -1.5707963267948966, -0.0
]
```

In [586]:

```
new_df = pd.DataFrame()
new_df = pd.DataFrame({
                    'X': X1,
                   'Y': X2})
Y = [-1,-1,-1,-1,1,1,1]
```

In [587]:

```
new_df
```

Out[587]:

	X	Y
0	1.0	0.000000
1	1.0	1.570796
2	1.0	-1.570796
3	1.0	-0.000000
4	2.0	1.570796
5	2.0	-1.570796
6	2.0	-0.000000

In [588]:

```
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(new_df,Y)
```

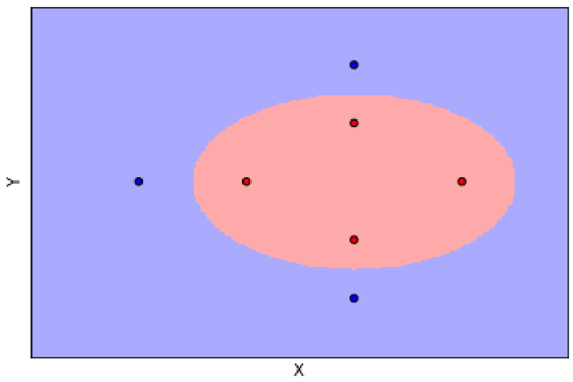
Out[588]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=1, p=2,
           weights='uniform')
```

In [589]:

```
x = [1,0,0,-1,0,0,-2]
y = [0,1,-1,0,2,-2,0]

h = .02

x_min, x_max = df['X'].min() - 1, df['X'].max() + 1
```

```
y_min, y_max = df['Y'].min() - 1, df['Y'].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))


Z11 = np.sqrt(np.square(x) + np.square(y))
Z22 = np.arctan(df['Y']/df['X'])
k = [np.array([Z11[i],Z22[i]]) for i in range(7)]

Z1 = np.sqrt(np.square(xx) + np.square(yy))
Z2 = np.arctan(yy/xx)
Z = knn.predict(np.c_[Z1.ravel(), Z2.ravel()])


Z = Z.reshape(xx.shape)

plt.set_cmap(plt.cm.Paired)
plt.pcolormesh(xx, yy, Z,cmap=cmap_light)

plt.scatter(df['X'],df['Y'] ,c=Y,cmap=cmap_bold,edgecolor='k', s=20   )
plt.xlabel('X')
plt.ylabel('Y')

plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xticks(())
plt.yticks(())

plt.show()
```



In [590]:

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(new_df,Y)
```

Out[590]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=3, p=2,
           weights='uniform')
```

In [592]:

```
h = .02

x_min, x_max = df['X'].min() - 1, df['X'].max() + 1
y_min, y_max = df['Y'].min() - 1, df['Y'].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))


Z11 = np.sqrt(np.square(x) + np.square(y))
Z22 = np.arctan(df['Y']/df['X'])
k = [np.array([Z11[i],Z22[i]]) for i in range(7)]

Z1 = np.sqrt(np.square(xx) + np.square(yy))
Z2 = np.arctan(yy/xx)
Z = knn.predict(np.c_[Z1.ravel(), Z2.ravel()])


Z = Z.reshape(xx.shape)
```

```
plt.set_cmap(plt.cm.Paired)
plt.pcolormesh(xx, yy, Z,cmap=cmap_light)


plt.scatter(df['X'],df['Y'] ,c=Y,cmap=cmap_bold,edgecolor='k', s=20  )
plt.xlabel('X')
plt.ylabel('Y')

plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xticks(())
plt.yticks(())

plt.show()
```