Machine Learning - AS2 By Tang Xu

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
```

Q1: Data Processing

In [2]:

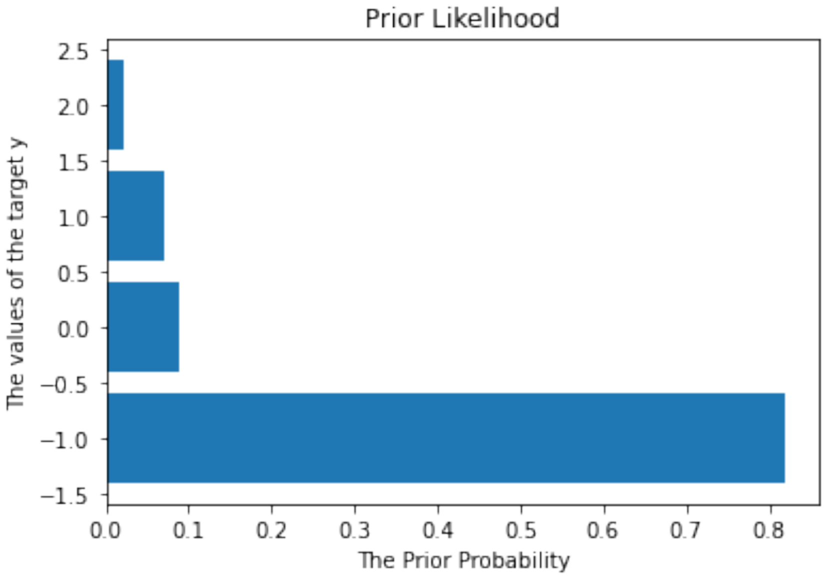
```
df_all=pd.read_csv('/Users/xutang/Desktop/Data Science 2/hw2_data.csv')
df_all=df_all.query('label == [-1,0,1,2]')
print('The shape of the total df is: {}'.format(df_all.shape))
for year in [2018,2019,2020]:
    locals()['df_'+str(year)] = df_all[(df_all['fyear']==year)]
    continue

y_train=df_2018['label']
y_valid=df_2019['label']
y_test=df_2020['label']

X_train=df_2018.drop('label',1)
X_valid=df_2019.drop('label',1)
X_test=df_2020.drop('label',1)

label_list=[-1,0,1,2]
```

```
The shape of the total df is: (2211, 56)
```

Q2: Prior Probabilities in the Training Set

In [3]:

```
pri_prob=np.array(y_train.value_counts())/len(y_train)
plt.barh(label_list,pri_prob)
plt.xlabel('The Prior Probability')
plt.ylabel('The values of the target y')
plt.title('Prior Likelihood')
plt.show()
print('The prior prob of different ys are:', pri_prob.round(4))
```



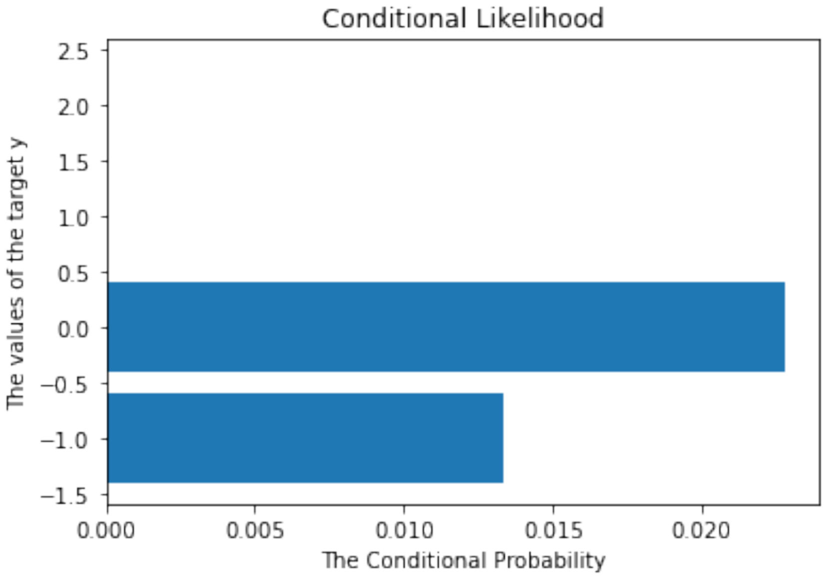

```
The prior prob of different ys are: [0.8204 0.0889 0.0702 0.0206]
```

Q3: Calculate the conditional probability

- ret=0.1 is extremely strict and I assume the number in the range [0.095,0.105] can all be seen as 0.1

In [4]:

```
df_train_y0=df_2018.query('label==-1')
df_train_y1=df_2018.query('label==0')
df_train_y2=df_2018.query('label==1')
df_train_y3=df_2018.query('label==2')

cond0=df_train_y0[(df_train_y0['ret']>=0.095) & (df_train_y0['ret']<=0.105)
cond1=df_train_y1[(df_train_y1['ret']>=0.095) & (df_train_y1['ret']<=0.105)
cond2=df_train_y2[(df_train_y2['ret']>=0.095) & (df_train_y2['ret']<=0.105)
cond3=df_train_y3[(df_train_y3['ret']>=0.095) & (df_train_y3['ret']<=0.105)

cond_prob=[cond0,cond1,cond2,cond3]

plt.barh(label_list,cond_prob)
plt.xlabel('The Conditional Probability')
plt.ylabel('The values of the target y')
plt.title('Conditional Likelihood')
plt.show()

print('The conditional probability when ret=0.1 is: ', np.array(cond_prob).
```




```
The conditional probability when ret=0.1 is:  [0.0133 0.0228 0.     0.    ]
```

Q4: Guassian Naive Bayes on Train and Valid

In [5]:

```
from sklearn.naive_bayes import GaussianNB
X_tv=pd.concat([X_train,X_valid])
y_tv=pd.concat([y_train,y_valid])
gnb=GaussianNB()
gnb.fit(X_tv,y_tv)
test_score = gnb.score(X_test,y_test)
print('The accuracy score of Gaussian Naive Bayes from train+valid is:', te
```

```
The accuracy score of Gaussian Naive Bayes from train+valid is: 0.2887
```

Q5: Output the confusion matrix

- I'm not so sure cause there are two pairs with the same absolute number 7

In [6]:

```
from sklearn.metrics import confusion_matrix
y_pred=gnb.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred, labels=[-1,0,1,2])
print(conf_matrix)
print('Two pairs are: regarding true 0 as 2 and mistaking -1 with 2')
```

```
[[ 1  0  0  7]
 [ 1 22  2 52]
 [ 0  0  1  7]
 [ 0  0  0  4]]
Two pairs are: regarding true 0 as 2 and mistaking -1 with 2
```

Q6: Implement Gaussian Mixture model

In [7]:

```
from sklearn.mixture import GaussianMixture
from sklearn.metrics import accuracy_score
cov_type=['full','tied','diag','spherical']
train_score=[]
test_score=[]

for i,t in enumerate(cov_type):
    gmm = GaussianMixture(n_components=4, init_params='kmeans', random_stat
    gmm.fit(X_tv, y_tv)
    pred_tv=gmm.predict(X_tv)
    pred_test=gmm.predict(X_test)
    train_score.append(accuracy_score(y_tv,pred_tv))
    test_score.append(accuracy_score(y_test,pred_test))
    continue


x = np.arange(len(cov_type))  # the label locations
width = 0.35  # the width of the bars
fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, train_score, width, label='train')
rects2 = ax.bar(x + width/2, test_score, width, label='test')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('Scores')
ax.set_title('Scores of train and test')
ax.set_xticks(x)
ax.set_xticklabels(cov_type)
ax.legend()
fig.tight_layout()

plt.show()

print('The params of the first and the second model will be: tied and sphe
print('The test scores when covariance_type equals Tied and Spherical are:
```


```
The params of the first and the second model will be: tied and spherical.
The test scores when covariance_type equals Tied and Spherical are: 0.4639
and 0.3299 respectively.
```

Q7: LDA Model

In [8]:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
solver_list=['svd', 'lsqr', 'eigen']
n_comp=[1,2,3]
score_matrix=np.ones((3,3))
for i,sol in enumerate(solver_list):
    for j,n in enumerate(n_comp):
        lda=LinearDiscriminantAnalysis(n_components=n, solver=sol)
        lda.fit(X_tv,y_tv)
        y_pred=lda.predict(X_test)
        score=accuracy_score(y_test,y_pred).round(4)
        score_matrix[i,j]=score
        continue
print(score_matrix)
```

```
[[0.7732 0.7732 0.7732]
 [0.7732 0.7732 0.7732]
 [0.7732 0.7732 0.7732]]
```

In [9]:

```
lda=LinearDiscriminantAnalysis(n_components=3, solver='svd')
lda.fit(X_tv,y_tv)
y_pred=lda.predict(X_test)
score=accuracy_score(y_test,y_pred).round(4)
print(score)
print(lda.coef_)
print(lda.intercept_)
```

```
0.7732
[[ 1.74259968e-06  4.97113283e-02  9.83729121e-01 -1.18396414e-01
  -2.48238518e+00 -6.36330269e+00  7.18693599e+00  5.14366430e+00
  -5.07526587e-02  2.01911636e-01 -1.71041911e+00 -5.70156185e-01
  -1.62866823e+00  1.13901686e-04  7.22339921e-01  4.06095283e-01
  -1.19973969e-02  3.01696187e-01  4.90008591e-01  9.53428147e-01
  -8.69759327e+01 -2.63544452e-01  2.09244303e-01 -5.84875977e-01
   6.07283939e-01 -3.38505945e-06  2.10499471e+00 -2.46832109e-01
   5.58574280e+00 -2.92714057e+00 -3.99678590e+00 -2.38240978e-02
   1.41708951e-01 -1.93183698e-03  3.01529578e-01 -1.61953836e-02
   2.86107208e-02  1.13416239e-02  9.13062248e-03 -1.04654804e-01
   5.58076642e-01  1.10143943e-04 -1.50293687e-05  2.42162137e-05
   4.53816631e-02 -6.27676693e-01  5.19097757e-01 -2.99390185e-01
  -4.10725529e+00  1.35947716e-01 -5.33681511e+00  4.70172970e-01
  -5.71705071e-01  1.05956620e+00 -2.77205136e-02]
 [-3.61401454e-07 -4.27830269e-03  8.81270160e-02  6.70110039e-03
   1.99474549e-01  1.23185052e-02 -1.56627025e-02 -4.18684663e-01
  -1.62943740e-02  5.50834029e-02 -2.87843782e-01 -2.25719902e-04
   1.52519695e-01 -6.44110273e-06 -6.00025808e-02 -2.49329957e-02
   7.29087036e-03 -7.18527943e-02 -5.99675621e-02 -6.84930675e-02
   7.82284803e+00  3.36383042e-03 -1.96828439e-02  2.30934540e-02
   1.70178283e-01  2.83743451e-07 -2.39404464e-01 -1.22647324e-01
  -4.14196556e-01  2.50901019e-01  7.76035026e-01  3.70424926e-04
  -9.44227457e-03  8.60218528e-05  1.64429447e-02  7.78669246e-03
  -2.26912726e-03 -3.32013499e-04 -1.53547945e-02 -4.68829637e-02
   2.60101805e-02 -1.12086980e-05 -5.01947662e-04 -4.59932146e-05
   1.12729039e-02  2.61656405e-01  3.05646639e-02  4.35394359e-02
   2.09110249e-01 -2.46907085e-02  4.99894512e-01  1.79496374e-02
   1.62163712e-02 -2.93291308e-02  6.62179303e-04]
 [ 7.29904202e-07  1.62190329e-02 -1.03845664e+00  9.58582636e-02
   7.36182022e-02  1.12790073e+00 -1.88593430e+00 -9.44721964e-02
   1.29084897e-01 -8.51833738e-01  2.44382873e+00  2.58103274e-01
  -1.66673483e-01 -3.02317536e-05  9.92138131e-02  7.76142174e-02
  -8.37648888e-02  3.63949850e-01  1.94031381e-01 -1.64881350e-01
   2.10037197e+00  1.79371270e-01  2.74183647e-02  9.60940059e-01
  -3.23115245e+00  1.63060222e-06  4.38294261e-01  1.11578754e+00
  -1.61808761e+00 -4.53109801e-02 -1.61824225e+00  7.40859968e-03
  -3.51907325e-03  1.33415289e-03 -9.28574040e-01 -3.50162795e-02
   6.28106310e-04 -6.00146004e-03  1.13854706e-01  4.15962937e-01
  -5.43879323e-01  2.58065297e-05  2.49175094e-03  3.96516937e-04
  -1.13034095e-01 -1.60879441e+00 -2.36898105e-01 -1.83267012e-01
   1.77727890e+00  1.25377816e-01 -2.32642743e-01 -4.51748931e-01
   3.39402832e-01 -4.05202779e-01  2.10492312e-02]
 [ 5.16512229e-06 -6.66144179e-02 -2.45578451e+00 -2.53682453e-01
   2.79951778e-01  1.61489921e+01 -1.56827549e+01 -6.09436840e-01
   2.72831021e-01  6.66089806e-01  6.86753365e+00  8.59608407e-01
   2.22828693e-01 -8.24639266e-06 -4.96503015e-01 -7.07422528e-01
   9.93696274e-02  2.86204452e-01 -1.09007633e-01  1.34734004e-01
  -2.04098299e+01  1.92692369e-02 -4.96619795e-02 -2.84913045e+00
   4.55064018e+00 -6.29377598e-06  4.36646539e-01  1.01669892e+00
   3.93986299e+00  2.62196153e-01 -1.00342353e+01  3.49951836e-02
  -9.37903007e-02 -2.30463038e-03  2.13677024e+00 -1.04009143e-01
  -1.03854812e-02 -5.91132478e-04  9.89093050e-02  4.65907293e-01
  -6.51894521e-01 -3.96863688e-05  9.28100278e-03  8.10930879e-05
  -1.26483804e-01 -1.45943641e+00 -1.94605704e+00  6.35984491e-02
  -1.60375858e+00 -1.05161696e-02 -5.49355188e-01 -4.22588038e-01
  -1.00146180e-01 -7.52610007e-01 -1.87439009e-02]]
[-105.04892604    8.67068748  -37.77897431  125.59681264]
```

In [ ]:

In [ ]: