Machine Learning with Python

Machine Learning Workshop @ MPSTME

Instructor: Santosh Chapaneri

Model Comparison with ROC Curves

In [1]:

```
import numpy as np
import pandas as pd
import sklearn.model_selection as cv
```

In [2]:

```
# Load the dataset with Pandas
train = pd.read_csv('Data/titanic_train.csv')

data = train[['Sex', 'Age', 'Pclass', 'Survived']].copy()
data['Sex'] = data['Sex'] == 'female'
data = data.dropna()
data.head()
```

Out[2]:

	Sex	Age	Pclass	Survived
0	False	22.0	3	0
1	True	38.0	1	1
2	True	26.0	3	1
3	True	35.0	1	1
4	False	35.0	3	0

In [3]:

```
# Create X and Y
data_np = data.astype(np.int32).values
X = data_np[:,:-1] # Features
y = data_np[:,-1] # Target (survived or not)
```

In [4]:

```
# Split the dataset
X_train, X_test, y_train, y_test = cv.train_test_split(X, y, test_size=0.25)
```

In [5]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn import svm
from sklearn.linear_model import Perceptron
from sklearn.neural_network import MLPClassifier

from sklearn import metrics
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
%matplotlib inline
```

Logistic Regression Classifier

In [6]:

```
### Logistic Regression
logreg = LogisticRegression(solver='lbfgs')
logreg.fit(X_train, y_train)

y_pred_log = logreg.predict(X_test)

confmat = confusion_matrix(y_test, y_pred_log)
confmat
```

Out[6]:

```
array([[78, 22],
       [25, 54]], dtype=int64)
```

In [7]:

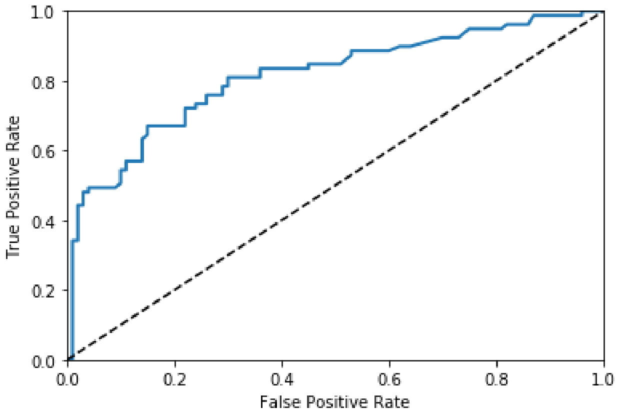
```
accuracy_LOG = accuracy_score(y_test, y_pred_log)
print('Accuracy of LogReg:', accuracy_LOG)
print()

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_log))
```

```
Accuracy of LogReg: 0.7374301675977654

             precision    recall  f1-score   support

          0       0.76      0.78      0.77       100
          1       0.71      0.68      0.70        79

avg / total       0.74      0.74      0.74       179
```

In [8]:

```
### ROC Curve for Logistic Regression
LOG_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
print('AUC of LogReg:', LOG_roc_auc)

fpr_LOG, tpr_LOG, thresholds_LOG = roc_curve(y_test,
                                             logreg.predict_proba(X_test)[:,1])

def plot_roc_curve(fpr, tpr, label=None):
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0,1],[0,1],'k--')
    plt.axis([0,1,0,1])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')

plot_roc_curve(fpr_LOG,tpr_LOG)
plt.show()
```

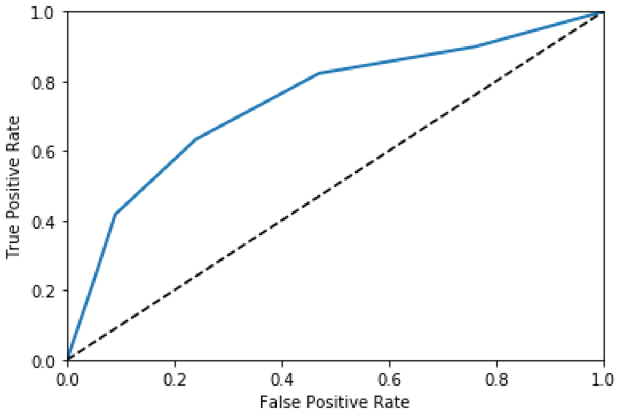
```
AUC of LogReg: 0.7317721518987342
```



KNN Classifier

```
In [9]: knn_clf = KNeighborsClassifier(n_neighbors=5)
        knn_clf.fit(X_train, y_train)
        y_pred_KNN = knn_clf.predict(X_test)
        print(classification_report(y_test, y_pred_KNN))

        accuracy_KNN = accuracy_score(y_test, y_pred_KNN)
        print('Accuracy of KNN:', accuracy_KNN)

        KNN_roc_auc = roc_auc_score(y_test, knn_clf.predict(X_test))
        print('AUC of KNN:', KNN_roc_auc)

        fpr_KNN, tpr_KNN, thresholds_KNN = roc_curve(y_test,
                                                      knn_clf.predict_proba(X_test)[:,1])
        plot_roc_curve(fpr_KNN, tpr_KNN)
        plt.show()
```

```
             precision    recall  f1-score   support

          0       0.72      0.76      0.74       100
          1       0.68      0.63      0.65        79

avg / total       0.70      0.70      0.70       179

Accuracy of KNN: 0.7039106145251397
AUC of KNN: 0.6964556962025317
```

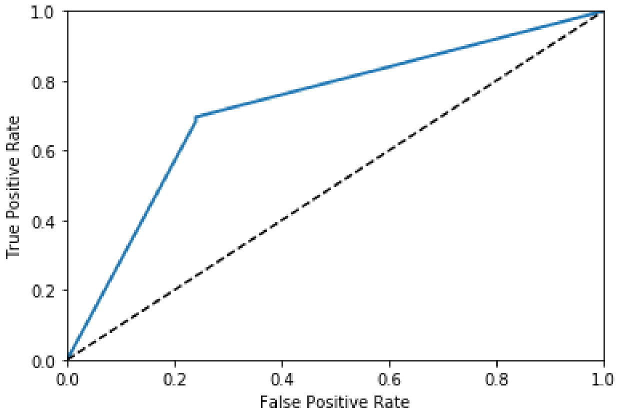


Naive Bayes Classifier

In [10]:

```
### Gussian Naive Bayes Algorithm
bnb = BernoulliNB()
bnb.fit(X_train, y_train)
y_pred_bnb = bnb.predict(X_test)
print(classification_report(y_test, y_pred_bnb))

accuracy_BNB = accuracy_score(y_test, y_pred_bnb)
print('Accuracy of Naive Bayes:', accuracy_BNB)

BNB_roc_auc = roc_auc_score(y_test, bnb.predict(X_test))
print('AUC of Naive Bayes:', BNB_roc_auc)

fpr_BNB, tpr_BNB, thresholds_BNB = roc_curve(y_test,
                                             bnb.predict_proba(X_test)[:,1])
plot_roc_curve(fpr_BNB, tpr_BNB)
plt.show()
```

```
             precision    recall  f1-score   support

          0       0.76      0.76      0.76       100
          1       0.70      0.70      0.70        79

avg / total       0.73      0.73      0.73       179

Accuracy of Naive Bayes: 0.7318435754189944
AUC of Naive Bayes: 0.7281012658227849
```

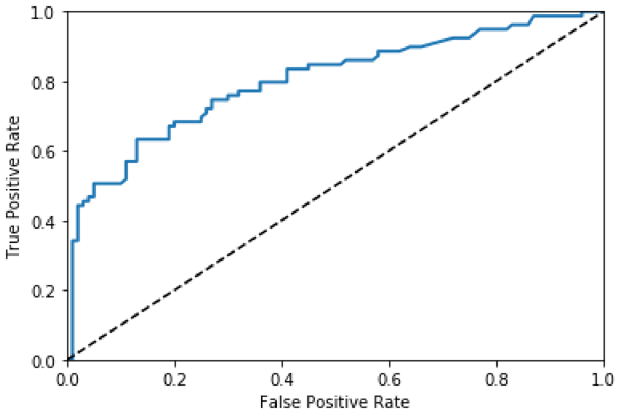


SVM

In [11]:

```
SVM_clf = svm.SVC(kernel='linear',probability=True)
SVM_clf.fit(X_train, y_train)
y_pred_SVM = SVM_clf.predict(X_test)
print(classification_report(y_test, y_pred_SVM))

accuracy_SVM = accuracy_score(y_test, y_pred_SVM)
print('Accuracy of SVM:', accuracy_SVM)

SVM_roc_auc = roc_auc_score(y_test, SVM_clf.predict(X_test))
print('AUC of SVM:', SVM_roc_auc)

fpr_SVM, tpr_SVM, thresholds_SVM = roc_curve(y_test,
                                             SVM_clf.predict_proba(X_test)[:,1])
plot_roc_curve(fpr_SVM, tpr_SVM)
plt.show()
```

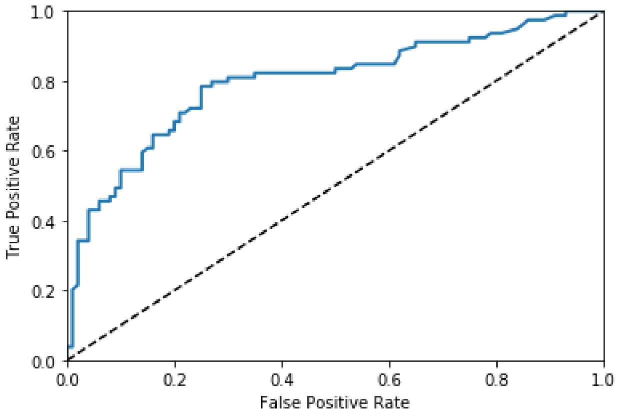
```
             precision    recall  f1-score   support

          0       0.75      0.76      0.76       100
          1       0.69      0.68      0.69        79

avg / total       0.73      0.73      0.73       179

Accuracy of SVM: 0.7262569832402235
AUC of SVM: 0.7217721518987342
```



In [12]:

```
# SVM with hyper-paramete tuning
# Set the parameters by cross-validation
parameters = [{'kernel': ['rbf'],
               'gamma': [1e-4, 1e-3, 0.01, 0.1, 0.2, 0.5],
                'C': [1, 10, 100, 1000]},
              {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]

# Perform Grid Search
print("# Tuning hyper-parameters")
from sklearn.model_selection import GridSearchCV
clf = GridSearchCV(svm.SVC(probability=True), parameters, cv=5, iid=True)

# Fit
clf.fit(X_train, y_train)

# Predict on test data
y_pred_SVM_CV = clf.predict(X_test)

# Show the classification report
print(classification_report(y_test, y_pred_SVM_CV))

accuracy_SVM_CV = accuracy_score(y_test, y_pred_SVM_CV)
print('Accuracy of SVM:', accuracy_SVM_CV)

SVM_CV_roc_auc = roc_auc_score(y_test, clf.predict(X_test))
print('AUC of SVM:', SVM_CV_roc_auc)

fpr_SVM_CV, tpr_SVM_CV, thresholds_SVM_CV = roc_curve(y_test,
                                                clf.predict_proba(X_test)[:,1])
plot_roc_curve(fpr_SVM_CV, tpr_SVM_CV)
plt.show()
```

```
# Tuning hyper-parameters
             precision    recall  f1-score   support

          0       0.74      0.81      0.78       100
          1       0.73      0.65      0.68        79

avg / total       0.74      0.74      0.74       179

Accuracy of SVM: 0.7374301675977654
AUC of SVM: 0.7277848101265822
```



Perceptron

In [17]:

```
perc_clf = Perceptron(alpha=0.0001, class_weight=None, eta0=1.0,
                      fit_intercept=True, max_iter=1000, n_jobs=1,
                      penalty=None, random_state=0, shuffle=True,
                      verbose=0, warm_start=False)

perc_clf.fit(X_train, y_train)
y_pred_PERC = perc_clf.predict(X_test)
print(classification_report(y_test, y_pred_PERC))

accuracy_PERC = accuracy_score(y_test, y_pred_PERC)
print('Accuracy of Perceptron:', accuracy_PERC)
```

```
             precision    recall  f1-score   support

          0       0.71      0.05      0.09       100
          1       0.45      0.97      0.61        79

avg / total       0.60      0.46      0.32       179

Accuracy of Perceptron: 0.4581005586592179
```

Multi-layer Perceptron

In [18]:

```
mlp = MLPClassifier(hidden_layer_sizes=(13,13,13),max_iter=500)
mlp.fit(X_train,y_train)
y_pred_MLP = mlp.predict(X_test)
print(classification_report(y_test, y_pred_MLP))

accuracy_MLP = accuracy_score(y_test, y_pred_MLP)
print('Accuracy of MLP:', accuracy_MLP)
```

```
             precision    recall  f1-score   support

          0       0.57      0.99      0.73       100
          1       0.83      0.06      0.12        79

avg / total       0.69      0.58      0.46       179

Accuracy of MLP: 0.5810055865921788
```

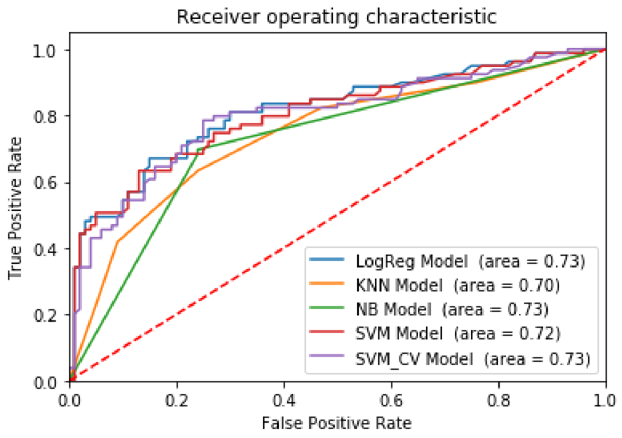
In [15]:

```
# Snapshot
print ("Logistic Regression Accuracy : " ,accuracy_LOG)
print ("KNN Accuracy :                 " ,accuracy_KNN)
print ("Naive Bayes Accuracy :         " ,accuracy_BNB)
print ("SVM Accuracy :                 " ,accuracy_SVM)
print ("SVM_CV Accuracy :              " ,accuracy_SVM_CV)
print ("Perceptron Accuracy :          " ,accuracy_PERC)
print ("MLP Accuracy :                 " ,accuracy_MLP)
```

```
Logistic Regression Accuracy :  0.7374301675977654
KNN Accuracy :                  0.7039106145251397
Naive Bayes Accuracy :          0.7318435754189944
SVM Accuracy :                  0.7262569832402235
SVM_CV Accuracy :               0.7374301675977654
Perceptron Accuracy :           0.4581005586592179
MLP Accuracy :                  0.5083798882681564
```

In [19]:

```
# Plot All ROC curves in one plot

plt.figure()
plt.plot(fpr_LOG, tpr_LOG,
         label='LogReg Model  (area = %0.2f)' % LOG_roc_auc)
plt.plot(fpr_KNN, tpr_KNN,
         label='KNN Model  (area = %0.2f)' % KNN_roc_auc)
plt.plot(fpr_BNB, tpr_BNB,
         label='NB Model  (area = %0.2f)' % BNB_roc_auc)
plt.plot(fpr_SVM, tpr_SVM,
         label='SVM Model  (area = %0.2f)' % SVM_roc_auc)
plt.plot(fpr_SVM_CV, tpr_SVM_CV,
         label='SVM_CV Model  (area = %0.2f)' % SVM_CV_roc_auc)

plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```


Receiver operating characteristic
True Positive Rate
False Positive Rate
LogReg Model (area = 0.73)
KNN Model (area = 0.70)
NB Model (area = 0.73)
SVM Model (area = 0.72)
SVM_CV Model (area = 0.73)