APPENDIX A – CODE SNIPPET OF DATASET COLLECTION

```
# import data from csv
dataframe = pandas.read_csv('Scoring.csv',usecols = range(0,19), engine='python')
X = dataframe.values

# Initialize Dictionaries and Lists
players2008 = {}
players2009 = {}
players2010 = {}
players2011 = {}
players_2008 = []
players_2009 = []
players_2010 = []
players_2011 = []

#Find players who have played in 2008-2011 seasons at least half of the season
for i in X:
    if (i[5] == 'C') and (i[4] == 'NHL') and (i[1] == 2008) and (i[6]>40 and i[6] != 'nan'):
        if i[0] in players2008:
            players2008[i[0]] += 1
        else:
            players2008[i[0]] = 1
    if (i[5] == 'C') and (i[4] == 'NHL') and (i[1] == 2009) and (i[6]>40 and i[6] != 'nan'):
        if i[0] in players2009:
            players2009[i[0]] += 1
        else:
            players2009[i[0]] = 1
    if (i[5] == 'C') and (i[4] == 'NHL') and (i[1] == 2010) and (i[6]>40 and i[6] != 'nan'):
        if i[0] in players2010:
            players2010[i[0]] += 1
        else:
            players2010[i[0]] = 1
    if (i[5] == 'C') and (i[4] == 'NHL') and (i[1] == 2011) and (i[6]>40 and i[6] != 'nan'):
        if i[0] in players2011:
            players2011[i[0]] += 1
        else:
            players2011[i[0]] = 1


# Remove players who have multiple team appearances in a season
for i in players2008.keys():
    if players2008[i] > 1:
        del players2008[i]
for i in players2009.keys():
    if players2009[i] > 1:
        del players2009[i]
for i in players2010.keys():
    if players2010[i] > 1:
        del players2010[i]
for i in players2011.keys():
    if players2011[i] > 1:
        del players2011[i]

#Add players to lists if they have played in all three seasons
for i in X:
    if ((i[0] in players2008) and (i[0] in players2009) and (i[0] in players2010) and (i[0] in players2011)) and (i[6]
        if (i[1] == 2008):
            players_2008.append(i)
        if (i[1] == 2009):
            players_2009.append(i)
        if (i[1] == 2010):
            players_2010.append(i)
        if (i[1] == 2011):
            players_2011.append(i)
```

APPENDIX B - CODE SNIPPET OF REGRESSION MODELS

```
## Linear Regression - Using feature data from 2008 and target data 2009
x_2008 = [X1_2008, X2_2008, X3_2008, X4_2008]
X_2008 = np.column_stack(x_2008+[[1]*len(x_2008[0])])
beta_hat = np.linalg.lstsq(X_2008,Y_2009)[0]
train_predicted_2009 = np.dot(X_2008,beta_hat)
```

```
## Calculating the Error for Train

error_train = (abs(train_predicted_2009 - Y_2009)/Y_2009) * 100

total = sum(error_train)/len(error_train)
```

```
## Predicting player performances for 2011 season using 2010 data


x_2010 = [X1_2010, X2_2010, X3_2010, X4_2010]
X_2010 = np.column_stack(x_2010+[[1]*len(x_2010[0])])
predicted_2011 = np.dot(X_2010,beta_hat)
```

```
## Calculating the Error for Test

# This is the error
error = (abs(predicted_2011 - Y_2011)/Y_2011) * 100

total = sum(error)/len(error)
```

```
## Linear regression using avg of past two years

X1_avg1 = [x + y for x, y in zip(X1_2008, X1_2009)]
X1_avg1 = [x / 2 for x in X1_avg1]
X2_avg1 = [x + y for x, y in zip(X2_2008, X2_2009)]
X2_avg1 = [x / 2 for x in X2_avg1]
X3_avg1 = [x + y for x, y in zip(X3_2008, X3_2009)]
X3_avg1 = [x / 2 for x in X3_avg1]
X4_avg1 = [x + y for x, y in zip(X4_2008, X4_2009)]
X4_avg1 = [x / 2 for x in X4_avg1]

# Linear Regression - Using feature data from 2008/2009 and target data 2010
x_avg1 = [X1_avg1, X2_avg1, X3_avg1, X4_avg1]
X_avg1 = np.column_stack(x_avg1+[[1]*len(x_avg1[0])])
beta_hat_avg = np.linalg.lstsq(X_avg1,Y_2010)[0]
train_avg = np.dot(X_2008,beta_hat)

X1_avg2 = [x + y for x, y in zip(X1_2009, X1_2010)]
X1_avg2 = [x / 2 for x in X1_avg2]
X2_avg2 = [x + y for x, y in zip(X2_2009, X2_2010)]
X2_avg2 = [x / 2 for x in X2_avg2]
X3_avg2 = [x + y for x, y in zip(X3_2009, X3_2010)]
X3_avg2 = [x / 2 for x in X3_avg2]
X4_avg2 = [x + y for x, y in zip(X4_2009, X4_2010)]
X4_avg2 = [x / 2 for x in X4_avg2]

x_avg2 = [X1_2010, X2_2010, X3_2010, X4_2010]
X_avg2 = np.column_stack(x_avg2+[[1]*len(x_avg2[0])])
predicted_2011_avg = np.dot(X_avg2,beta_hat_avg)
```

```
## Calculating the Error

#This is the train error
error_train_avg = (abs(train_avg - Y_2010)/Y_2010) * 100
total_train_avg = sum(error_train_avg)/len(error_train_avg)
print total_train_avg

# This is the test error
error_avg = (abs(predicted_2011_avg - Y_2011)/Y_2011) * 100
total_test_avg = sum(error_avg)/len(error_avg)
print total_test_avg
```

APPENDIX C – CODE SNIPPET OF K-FOLD VALIDATION

```
######################### 2nd fold
train_X_1 = train_X_2
train_X_2 = test_X
test_X = train_X_1

train_Y_1 = train_Y_2
train_Y_2 = test_Y
test_Y = train_Y_1

training_X = []
for i in train_X_1:
    training_X.append(i)
for i in train_X_2:
    training_X.append(i)

training_Y = []
for i in train_Y_1:
    training_Y.append(i)
for i in train_Y_2:
    training_Y.append(i)


b_hat = np.linalg.lstsq(training_X,training_Y)[0]
pred_train = np.dot(training_X,b_hat)
pred_test = np.dot(test_X,b_hat)

# Training error
err_train = (abs(pred_train - training_Y)/training_Y) * 100

ttl_train = sum(err_train)/len(err_train)

Training_Error.append(ttl_train)
print ttl_train

# Testing error
err_test = (abs(pred_test - test_Y)/test_Y) * 100

ttl_test = sum(err_test)/len(err_test)

Testing_Error.append(ttl_test)
print ttl_test
```

APPENDIX D – CODE SNIPPET OF SVM

```
# SVM
from sklearn.svm import SVC

model_SVM = SVC()
model_SVM.fit(Stats,cluster_train_Y)

predict_train_Y = model_SVM.predict(Stats)
predict_test_Y = model_SVM.predict(Stats_test)

print predict_train_Y
print predict_test_Y

print calError(predict_train_Y,cluster_train_Y) # Train Error
print calError(predict_test_Y,cluster_test_Y)  # Test Error
```