MACHINE LEARNING

HEART DISEASE PREDICTION

Team Members:

NAVADEEP REDDY (2010030313)

VENKATESH (2010030359)

SHASHIKANTH (2010030494)

MANOJ PERAVALI (2010030503)

AKHIL (2010030513)

CODE:

```
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
import matplotlib.pyplot as plt
filePath = 'C:\\Users\\manoj\\Downloads\\archive.zip'
data = pd.read_csv(filePath)


data.head(5)
print("(Rows, columns): " + str(data.shape))
data.columns
data.nunique(axis=0) #it returns unique values
data.describe() #to describe data
print(data.isna().sum()) #it displays missing values
```

```
data['target'].value_counts()

corr = data.corr()

plt.subplots(figsize=(15,10))

sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, cmap=sns.diverging_palette(220, 20, as_cmap=True))

sns.heatmap(corr, xticklabels=corr.columns,

        yticklabels=corr.columns,

        annot=True,

        cmap=sns.diverging_palette(220, 20, as_cmap=True)) #correlation matrix shows postive or negative relation to target



neg_data = data[data['target']==0] #filters data by negative heart disease

neg_data.describe()

pos_data = data[data['target']==1] #filters data by positive heart disease

pos_data.describe() #the above two are used to calculate thalach(max heart rate)

print("(Positive Patients ST depression): " + str(pos_data['oldpeak'].mean()))

print("(Negative Patients ST depression): " + str(neg_data['oldpeak'].mean()))

print("(Positive Patients thalach): " + str(pos_data['thalach'].mean()))

print("(Negative Patients thalach): " + str(neg_data['thalach'].mean()))

X = data.iloc[:, :-1].values #here we assign 13 features to X and 1 target feature to y

y = data.iloc[:, -1].values

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X,y,test_size = 0.32, random_state = 1) #we split train and test sample

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

x_train = sc.fit_transform(x_train)
```

```
x_test = sc.transform(x_test)
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression

model1 = LogisticRegression(random_state=1) # get instance of model
model1.fit(x_train, y_train) #to Train model

y_pred1 = model1.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred1)) # output accuracy

from sklearn.metrics import classification_report
from sklearn.svm import SVC

model2 = SVC(random_state=1) # get instance of model
model2.fit(x_train, y_train) # Train/Fit model

y_pred2 = model2.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred2)) # output accuracy
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier

model3 = RandomForestClassifier(random_state=1)# get instance of model
model3.fit(x_train, y_train) # Train/Fit model

y_pred3 = model3.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred3)) # output accuracy
```

```
#testing
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size = 0.32, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

x_train = sc.fit_transform(x_train)
x_test[0,10]
for dd in x_test:
    print(dd)
x_test = sc.transform(x_test)
from sklearn.metrics import classification_report



model6 = LogisticRegression(random_state=1) # get instance of model

model6.fit(x_train, y_train) # Train/Fit model

y_pred6 = model6.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred6)) # output accuracy



for i in range(10):
    print('rows',i)
    print(x_test[i:i+1])
```

```
    y_pred6 = model6.predict(x_test[i:i+1])
    print(y_pred6)
from sklearn.metrics import classification_report
from sklearn.svm import SVC

model7 = SVC(random_state=1) # get instance of model
model7.fit(x_train, y_train) # Train/Fit model

y_pred7 = model7.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred7)) # output accuracy

from sklearn.metrics import classification_report
from sklearn.svm import SVC

model7 = SVC(random_state=1) # get instance of model
model7.fit(x_train, y_train) # Train/Fit model
for i in range(10):
    print('rows',i)
    print(x_test[i:i+1])
    y_pred7= model7.predict(x_test[i:i+1])
    print(y_pred7)
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier

model9 = RandomForestClassifier(random_state=1)# get instance of model
model9.fit(x_train, y_train) # Train/Fit model
```

```
y_pred9 = model9.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred9)) # output accuracy


for i in range(10):
    print('rows',i)
    print(x_test[i:i+1])
    y_pred9 = model9.predict(x_test[i:i+1])
    print(y_pred9)
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
model8 = KNeighborsClassifier()# get instance of model
model8.fit(x_train, y_train) # Train/Fit model

y_pred8 = model8.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred8)) # output accuracy


for i in range(10):
    print('rows',i)
    print(x_test[i:i+1])
    y_pred8 = model8.predict(x_test[i:i+1])
    print(y_pred8)
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred3)
print(cm)
accuracy_score(y_test, y_pred3)
```

```
# get importance
importance = model3.feature_importances_

# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
index= data.columns[:-1]
importance = pd.Series(model3.feature_importances_, index=index)
importance.nlargest(13).plot(kind='barh', colormap='winter')
```

OUTPUT:

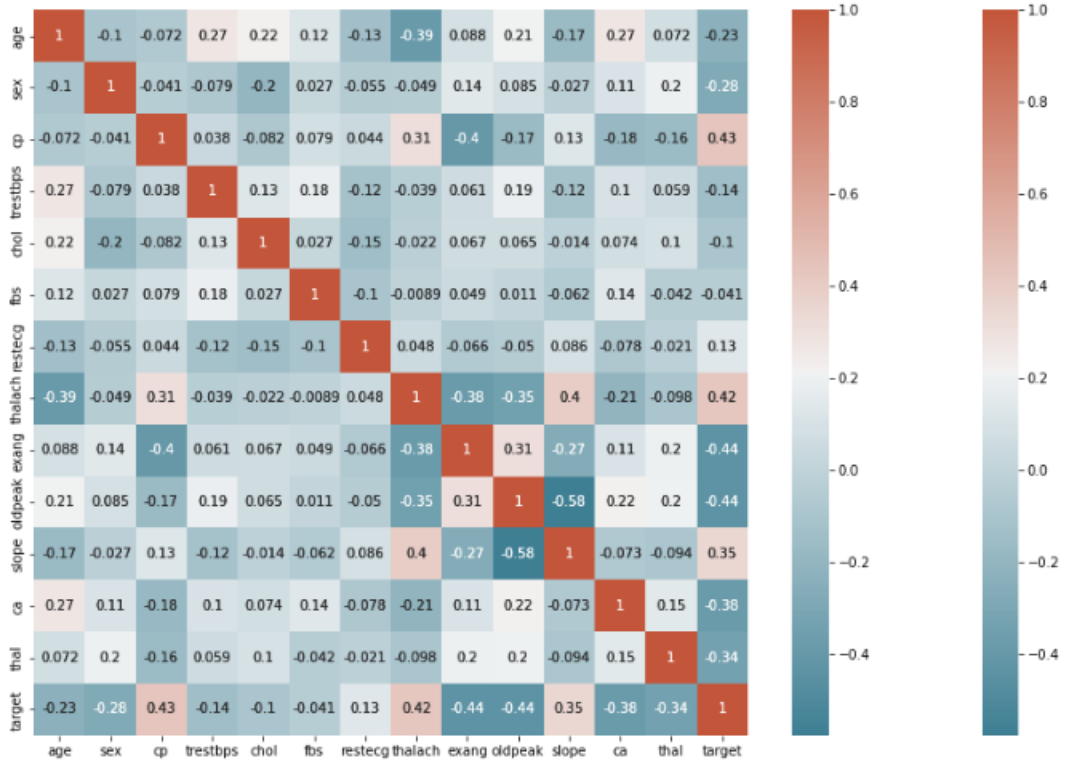
In [1]:

```
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
import matplotlib.pyplot as plt
filePath = 'C:\\Users\\manoj\\Downloads\\archive.zip'
data = pd.read_csv(filePath)

data.head(5)
print("(Rows, columns): " + str(data.shape))
data.columns
data.nunique(axis=0) #it returns unique values
data.describe() #to describe data
print(data.isna().sum()) #it displays missing values
data['target'].value_counts()
corr = data.corr()
plt.subplots(figsize=(15,10))
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, cmap=sns.diverging_palette(220, 20, as_cmap=Tru
sns.heatmap(corr, xticklabels=corr.columns,
            yticklabels=corr.columns,
            annot=True,
            cmap=sns.diverging_palette(220, 20, as_cmap=True)) #correlation matrix shows postive or negative relation to target


neg_data = data[data['target']==0] #filters data by negative heart disease
neg_data.describe()
pos_data = data[data['target']==1] #filters data by positive heart disease
pos_data.describe() #the above two are used to calculate thalach(max heart rate)
```

```
(Rows, columns): (1025, 14)
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Out[1]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	526.000000	526.000000	526.000000	526.000000	526.000000	526.000000	526.000000	526.000000	526.000000	526.000000	526.000000	526.000000	526.0
mean	52.408745	0.570342	1.378327	129.245247	240.979087	0.134981	0.598859	158.585551	0.134981	0.569962	1.593156	0.370722	2.1
std	9.631804	0.495498	0.945881	16.112188	53.010345	0.342029	0.502109	19.096928	0.342029	0.771079	0.590295	0.871462	0.4
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	96.000000	0.000000	0.000000	0.000000	0.000000	0.0



```
print("(Positive Patients ST depression): " + str(pos_data['oldpeak'].mean()))
print("(Negative Patients ST depression): " + str(neg_data['oldpeak'].mean()))
print("(Positive Patients thalach): " + str(pos_data['thalach'].mean()))
print("(Negative Patients thalach): " + str(neg_data['thalach'].mean()))
```

```
(Positive Patients ST depression): 0.5699619771863115
(Negative Patients ST depression): 1.6002004008016042
(Positive Patients thalach): 158.58555133079847
(Negative Patients thalach): 139.1302605210421
```

```
X = data.iloc[:, :-1].values #here we assign 13 features to X and 1 target feature to y
y = data.iloc[:, -1].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size = 0.32, random_state = 1) #we split train and test sample
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression

model1 = LogisticRegression(random_state=1) # get instance of model
model1.fit(x_train, y_train) #to Train model

y_pred1 = model1.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred1)) # output accuracy
```

```
              precision    recall  f1-score   support

           0       0.90      0.72      0.80       169
           1       0.75      0.92      0.83       159

    accuracy                           0.81       328
   macro avg       0.83      0.82      0.81       328
weighted avg       0.83      0.81      0.81       328
```

In [4]:
```
from sklearn.metrics import classification_report
from sklearn.svm import SVC

model2 = SVC(random_state=1) # get instance of model
model2.fit(x_train, y_train) # Train/Fit model

y_pred2 = model2.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred2)) # output accuracy
```

```
              precision    recall  f1-score   support

           0       0.97      0.84      0.90       169
           1       0.85      0.97      0.91       159

    accuracy                           0.90       328
   macro avg       0.91      0.90      0.90       328
weighted avg       0.91      0.90      0.90       328
```

In [5]:
```
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier

model3 = RandomForestClassifier(random_state=1)# get instance of model
model3.fit(x_train, y_train) # Train/Fit model

y_pred3 = model3.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred3)) # output accuracy
```

```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       169
           1       0.98      1.00      0.99       159

    accuracy                           0.99       328
   macro avg       0.99      0.99      0.99       328
weighted avg       0.99      0.99      0.99       328
```

In [6]:
```
#testing
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y,test_size = 0.32, random_state = 0)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

x_train = sc.fit_transform(x_train)
x_test[0,10]
for dd in x_test:
    print(dd)
x_test = sc.transform(x_test)
```

```
[ 44.    1.    2.  130.  233.    0.    1.  179.    1.    0.4   2.    0.
   2. ]
[ 58.    0.    1. 136. 319.    1.    0. 152.    0.    0.    2.    2.    2.]
[ 63.    1.    0. 140. 187.    0.    0. 144.    1.    4.    2.    2.    3.]
[ 58.    1.    2. 140. 211.    1.    0. 165.    0.    0.    2.    0.    2.]
[ 61.    1.    0.  120.  260.    0.    1.  140.    1.    3.6   1.    1.
   3. ]
[ 53.    1.    0.  140.  203.    1.    0.  155.    1.    3.1   0.    0.
   3. ]
[ 58.    1.    0. 125. 300.    0.    0. 171.    0.    0.    2.    2.    3.]
[ 63.    0.    0. 150. 407.    0.    0. 154.    0.    4.    1.    3.    3.]
[ 57.    1.    0.  130.  131.    0.    1.  115.    1.    1.2   1.    1.
   3. ]
[ 52.    1.    0. 112. 230.    0.    1. 160.    0.    0.    2.    1.    2.]
[ 57.    1.    2.  128.  229.    0.    0.  150.    0.    0.4   1.    1.
   3. ]
[ 51.    1.    0.  140.  298.    0.    1.  122.    1.    4.2   1.    3.
   3. ]
[ 64.    1.    3.  110.  211.    0.    0.  144.    1.    1.8   1.    0.
```

In [7]:
```
from sklearn.metrics import classification_report
from sklearn.svm import SVC

model7 = SVC(random_state=1) # get instance of model
model7.fit(x_train, y_train) # Train/Fit model

y_pred7 = model7.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred7)) # output accuracy

from sklearn.metrics import classification_report
from sklearn.svm import SVC

model7 = SVC(random_state=1) # get instance of model
model7.fit(x_train, y_train) # Train/Fit model
for i in range(10):
    print('rows',i)
    print(x_test[i:i+1])
    y_pred7= model7.predict(x_test[i:i+1])
    print(y_pred7)
```

```
              precision    recall  f1-score   support

           0       0.96      0.93      0.95       152
           1       0.94      0.97      0.96       176

    accuracy                           0.95       328
   macro avg       0.95      0.95      0.95       328
weighted avg       0.95      0.95      0.95       328

rows 0
[[-1.17250599  0.64996075  1.03075659 -0.09487623 -0.2521747  -0.42350563
   0.94034933  1.29398006  1.42493326 -0.5757849   0.96862314 -0.74507628
  -0.51634564]]
[1]
rows 1
[[ 0.38646845 -1.53855444  0.05858157  0.24570093  1.4410992   2.36124368
  -0.95393819  0.12019182 -0.70178726 -0.92759036  0.96862314  1.1890696
  -0.51634564]]
[1]
rows 2
[[ 0.94324504  0.64996075 -0.91359346  0.47275238 -1.15787935 -0.42350563
  -0.95393819 -0.22759728  1.42493326  2.59046423  0.96862314  1.1890696
   1.09031915]]
[0]
rows 3
[[ 0.38646845  0.64996075  1.03075659  0.47275238 -0.68533779  2.36124368
  -0.95393819  0.68534912 -0.70178726 -0.92759036  0.96862314 -0.74507628
  -0.51634564]]
[1]
rows 4
[[ 0.72053441  0.64996075 -0.91359346 -0.66250484  0.27943454 -0.42350563
   0.94034933 -0.40149184  1.42493326  2.23865877 -0.66212645  0.22199666
   1.09031915]]
[0]
rows 5
[[-0.17030813  0.64996075 -0.91359346  0.47275238 -0.84285165  2.36124368
  -0.95393819  0.25061274  1.42493326  1.79890195 -2.29287603 -0.74507628
   1.09031915]]
```

In [8]:
```
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier

model9 = RandomForestClassifier(random_state=1)# get instance of model
model9.fit(x_train, y_train) # Train/Fit model

y_pred9 = model9.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred9)) # output accuracy


for i in range(10):
    print('rows',i)
    print(x_test[i:i+1])
    y_pred9 = model9.predict(x_test[i:i+1])
    print(y_pred9)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       152
           1       1.00      1.00      1.00       176

    accuracy                           1.00       328
   macro avg       1.00      1.00      1.00       328
weighted avg       1.00      1.00      1.00       328

rows 0
[[-1.17250599  0.64996075  1.03075659 -0.09487623 -0.2521747  -0.42350563
   0.94034933  1.29398006  1.42493326 -0.5757849   0.96862314 -0.74507628
  -0.51634564]]
[1]
rows 1
[[ 0.38646845 -1.53855444  0.05858157  0.24570093  1.4410992   2.36124368
  -0.95393819  0.12019182 -0.70178726 -0.92759036  0.96862314  1.1890696
  -0.51634564]]
[0]
rows 2
[[ 0.94324504  0.64996075 -0.91359346  0.47275238 -1.15787935 -0.42350563
  -0.95393819 -0.22759728  1.42493326  2.59046423  0.96862314  1.1890696
   1.09031915]]
[0]
rows 3
[[ 0.38646845  0.64996075  1.03075659  0.47275238 -0.68533779  2.36124368
  -0.95393819  0.68534912 -0.70178726 -0.92759036  0.96862314 -0.74507628
  -0.51634564]]
[1]
rows 4
[[ 0.72053441  0.64996075 -0.91359346 -0.66250484  0.27943454 -0.42350563
   0.94034933 -0.40149184  1.42493326  2.23865877 -0.66212645  0.22199666
   1.09031915]]
[0]
rows 5
[[-0.17030813  0.64996075 -0.91359346  0.47275238 -0.84285165  2.36124368
  -0.95393819  0.25061274  1.42493326  1.79890195 -2.29287603 -0.74507628
   1.09031915]]
[0]
rows 6
[[ 0.38646845  0.64996075 -0.91359346 -0.37869054  1.0670038  -0.42350563
  -0.95393819  0.94619096 -0.70178726 -0.92759036  0.96862314  1.1890696
```

In [9]:
```
from sklearn.metrics import classification_report


model6 = LogisticRegression(random_state=1) # get instance of model

model6.fit(x_train, y_train) # Train/Fit model

y_pred6 = model6.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred6)) # output accuracy


for i in range(10):
    print('rows',i)
    print(x_test[i:i+1])
    y_pred6 = model6.predict(x_test[i:i+1])
    print(y_pred6)
```

```
              precision    recall  f1-score   support

           0       0.90      0.81      0.85       152
           1       0.85      0.92      0.88       176

    accuracy                           0.87       328
   macro avg       0.87      0.86      0.87       328
weighted avg       0.87      0.87      0.87       328

rows 0
[[-1.17250599  0.64996075  1.03075659 -0.09487623 -0.2521747  -0.42350563
   0.94034933  1.29398006  1.42493326 -0.5757849   0.96862314 -0.74507628
  -0.51634564]]
[1]
rows 1
[[ 0.38646845 -1.53855444  0.05858157  0.24570093  1.4410992   2.36124368
  -0.95393819  0.12019182 -0.70178726 -0.92759036  0.96862314  1.1890696
  -0.51634564]]
[1]
rows 2
[[ 0.94324504  0.64996075 -0.91359346  0.47275238 -1.15787935 -0.42350563
  -0.95393819 -0.22759728  1.42493326  2.59046423  0.96862314  1.1890696
   1.09031915]]
[0]
rows 3
[[ 0.38646845  0.64996075  1.03075659  0.47275238 -0.68533779  2.36124368
  -0.95393819  0.68534912 -0.70178726 -0.92759036  0.96862314 -0.74507628
  -0.51634564]]
[1]
rows 4
[[ 0.72053441  0.64996075 -0.91359346 -0.66250484  0.27943454 -0.42350563
   0.94034933 -0.40149184  1.42493326  2.23865877 -0.66212645  0.22199666
   1.09031915]]
[0]
rows 5
[[-0.17030813  0.64996075 -0.91359346  0.47275238 -0.84285165  2.36124368
  -0.95393819  0.25061274  1.42493326  1.79890195 -2.29287603 -0.74507628
   1.09031915]]
[0]
rows 6
```

In [10]:
```
from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier
model8 = KNeighborsClassifier()# get instance of model
model8.fit(x_train, y_train) # Train/Fit model

y_pred8 = model8.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred8)) # output accuracy


for i in range(10):
    print('rows',i)
    print(x_test[i:i+1])
    y_pred8 = model8.predict(x_test[i:i+1])
    print(y_pred8)
```

```
              precision    recall  f1-score   support

           0       0.87      0.87      0.87       152
           1       0.89      0.89      0.89       176

    accuracy                           0.88       328
   macro avg       0.88      0.88      0.88       328
weighted avg       0.88      0.88      0.88       328

rows 0
[[-1.17250599  0.64996075  1.03075659 -0.09487623 -0.2521747  -0.42350563
   0.94034933  1.29398006  1.42493326 -0.5757849   0.96862314 -0.74507628
  -0.51634564]]
[1]
rows 1
[[ 0.38646845 -1.53855444  0.05858157  0.24570093  1.4410992   2.36124368
  -0.95393819  0.12019182 -0.70178726 -0.92759036  0.96862314  1.1890696
  -0.51634564]]
[1]
rows 2
[[ 0.94324504  0.64996075 -0.91359346  0.47275238 -1.15787935 -0.42350563
  -0.95393819 -0.22759728  1.42493326  2.59046423  0.96862314  1.1890696
   1.09031915]]
[0]
rows 3
[[ 0.38646845  0.64996075  1.03075659  0.47275238 -0.68533779  2.36124368
  -0.95393819  0.68534912 -0.70178726 -0.92759036  0.96862314 -0.74507628
  -0.51634564]]
[1]
rows 4
[[ 0.72053441  0.64996075 -0.91359346 -0.66250484  0.27943454 -0.42350563
   0.94034933 -0.40149184  1.42493326  2.23865877 -0.66212645  0.22199666
   1.09031915]]
[0]
rows 5
[[-0.17030813  0.64996075 -0.91359346  0.47275238 -0.84285165  2.36124368
  -0.95393819  0.25061274  1.42493326  1.79890195 -2.29287603 -0.74507628
   1.09031915]]
[0]
rows 6
[[ 0.38646845  0.64996075 -0.91359346 -0.37869054  1.0670038  -0.42350563
  -0.95393819  0.94619096 -0.70178726 -0.92759036  0.96862314  1.1890696
   1.09031915]]
[0]
```

In [11]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred3)
print(cm)
accuracy_score(y_test, y_pred3)
```

```
[[71 81]
 [94 82]]
```

Out[11]: 0.46646341463414637

In [12]:
```
# get importance
importance = model3.feature_importances_

# summarize feature importance
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
```

```
Feature: 0, Score: 0.08525
Feature: 1, Score: 0.02915
Feature: 2, Score: 0.13056
Feature: 3, Score: 0.06158
Feature: 4, Score: 0.08428
Feature: 5, Score: 0.00913
Feature: 6, Score: 0.02352
Feature: 7, Score: 0.11836
Feature: 8, Score: 0.07039
Feature: 9, Score: 0.11075
Feature: 10, Score: 0.06251
Feature: 11, Score: 0.09574
Feature: 12, Score: 0.11878
```

In [13]:
```
index= data.columns[:-1]
importance = pd.Series(model3.feature_importances_, index=index)
importance.nlargest(13).plot(kind='barh', colormap='winter')
```

Out[13]: <AxesSubplot:>