In [27]:
```
import pandas as pds
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns #for plotting
from sklearn.ensemble import RandomForestClassifier #for the model
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz #plot tree
from sklearn.metrics import roc_curve, auc #for model evaluation
from sklearn.metrics import classification_report #for model evaluation
from sklearn.metrics import confusion_matrix #for model evaluation
from sklearn.model_selection import train_test_split #for data splitting
import eli5 #for purmutation importance
from eli5.sklearn import PermutationImportance
import shap #for SHAP values
from pdpbox import pdp, info_plots #for partial plots

import warnings
warnings.filterwarnings('ignore')

print("> Reading from data file and putting headers")
dataframe = pds.read_csv("E:\\New_Big _data\\Program\\processed.cleveland.csv"
,
                         header=None,
                         names=['age', 'sex', 'cp', 'trestbps','chol','fbs','r
estecg','thalach','exang','oldpeak','slope','ca','thal','num'])
print("dataframe >>")
dataframe
```

```
> Reading from data file and putting headers
dataframe >>
```

Out[27]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	nu
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	45.0	1.0	1.0	110.0	264.0	0.0	0.0	132.0	0.0	1.2	2.0	0.0	7.0	
299	68.0	1.0	4.0	144.0	193.0	1.0	0.0	141.0	0.0	3.4	2.0	2.0	7.0	
300	57.0	1.0	4.0	130.0	131.0	0.0	0.0	115.0	1.0	1.2	2.0	1.0	7.0	
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0	
302	38.0	1.0	3.0	138.0	175.0	0.0	0.0	173.0	0.0	0.0	1.0	?	3.0	

303 rows × 14 columns

In [28]:
```
print("> Replace empty values with NaN and meaningful value")
dataframe_re = dataframe.replace(to_replace ="?",value ="NaN")
print("dataframe_re >>")

dataframe_re
```

```
> Replace empty values with NaN and meaningful value
dataframe_re >>
```

Out[28]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	r
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	45.0	1.0	1.0	110.0	264.0	0.0	0.0	132.0	0.0	1.2	2.0	0.0	7.0	
299	68.0	1.0	4.0	144.0	193.0	1.0	0.0	141.0	0.0	3.4	2.0	2.0	7.0	
300	57.0	1.0	4.0	130.0	131.0	0.0	0.0	115.0	1.0	1.2	2.0	1.0	7.0	
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0	
302	38.0	1.0	3.0	138.0	175.0	0.0	0.0	173.0	0.0	0.0	1.0	NaN	3.0	

303 rows × 14 columns

In [29]:
```
from sklearn.preprocessing import Imputer
imp = Imputer(missing_values='NaN', strategy="median",axis=0)
imp = imp.fit(dataframe_re)
imp_df = imp.transform(dataframe_re)
print("imp_df >>")
print(type(imp_df))
```

```
imp_df >>
<class 'numpy.ndarray'>
```

In [30]:
```
new_df = pds.DataFrame(imp_df, columns =['age', 'sex', 'cp', 'trestbps','chol'
,'fbs','restecg','thalach','exang','oldpeak','slope','ca','thal','num'])
print("new_df >>")
new_df
```

```
new_df >>
```

Out[30]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	nu
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
298	45.0	1.0	1.0	110.0	264.0	0.0	0.0	132.0	0.0	1.2	2.0	0.0	7.0	1
299	68.0	1.0	4.0	144.0	193.0	1.0	0.0	141.0	0.0	3.4	2.0	2.0	7.0	2
300	57.0	1.0	4.0	130.0	131.0	0.0	0.0	115.0	1.0	1.2	2.0	1.0	7.0	3
301	57.0	0.0	2.0	130.0	236.0	0.0	2.0	174.0	0.0	0.0	2.0	1.0	3.0	1
302	38.0	1.0	3.0	138.0	175.0	0.0	0.0	173.0	0.0	0.0	1.0	0.0	3.0	0

303 rows × 14 columns

In [31]:
```
print("> Outliner detection for each column")
def detect_outlier(data_1):
    outliers=[]
    threshold=3
    mean_1 = np.mean(data_1)
    std_1 =np.std(data_1)
    ourliers_row=[]
    i = 0

    for y in data_1:
        i += 1
        z_score= (y - mean_1)/std_1
        if np.abs(z_score) > threshold:
            outliers.append(y)
            ourliers_row.append(i)
    return ourliers_row
out_list=[]
for col in list(new_df.columns):
    print(col)
    out_list.extend(detect_outlier(new_df[col].values))
    print(out_list)
```

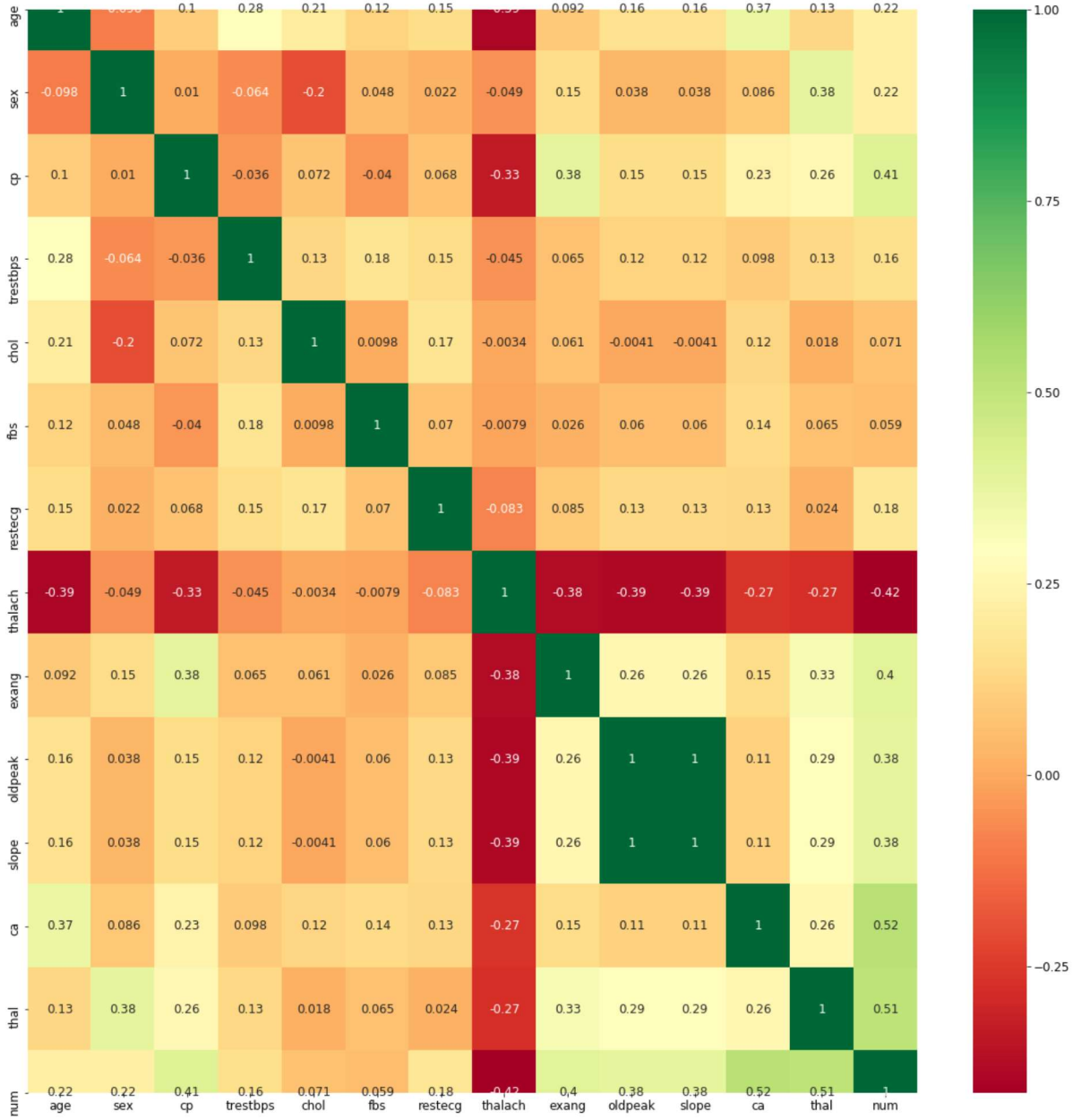
```
> Outliner detection for each column
age
[]
sex
[]
cp
[]
trestbps
[127, 189]
chol
[127, 189, 49, 122, 153, 182]
fbs
[127, 189, 49, 122, 153, 182]
restecg
[127, 189, 49, 122, 153, 182]
thalach
[127, 189, 49, 122, 153, 182, 246]
exang
[127, 189, 49, 122, 153, 182, 246]
oldpeak
[127, 189, 49, 122, 153, 182, 246, 92, 124]
slope
[127, 189, 49, 122, 153, 182, 246, 92, 124]
ca
[127, 189, 49, 122, 153, 182, 246, 92, 124]
thal
[127, 189, 49, 122, 153, 182, 246, 92, 124]
num
[127, 189, 49, 122, 153, 182, 246, 92, 124]
```

In [32]:
```
print("> Check and use common datatype")
print(new_df.dtypes)
new_df['age'] = new_df['age'].astype('int64')
new_df['sex'] = new_df['sex'].astype('int64')
new_df['cp'] = new_df['cp'].astype('int64')
new_df['trestbps'] = new_df['trestbps'].astype('int64')
new_df['chol'] = new_df['chol'].astype('int64')
new_df['cp'] = new_df['cp'].astype('int64')
new_df['fbs'] = new_df['fbs'].astype('int64')
new_df['restecg'] = new_df['restecg'].astype('int64')
new_df['thalach'] = new_df['thalach'].astype('int64')
new_df['exang'] = new_df['exang'].astype('int64')
new_df['oldpeak'] = new_df['slope'].astype('int64')
new_df['slope'] = new_df['slope'].astype('int64')
new_df['ca'] = new_df['ca'].astype('int64')
new_df['thal'] = new_df['thal'].astype('int64')
new_df['num'] = new_df['num'].astype('int64')
print(new_df.dtypes)
```

```
> Check and use common datatype
age         float64
sex         float64
cp          float64
trestbps    float64
chol        float64
fbs         float64
restecg     float64
thalach     float64
exang       float64
oldpeak     float64
slope       float64
ca          float64
thal        float64
num         float64
dtype: object
age         int64
sex         int64
cp          int64
trestbps    int64
chol        int64
fbs         int64
restecg     int64
thalach     int64
exang       int64
oldpeak     int64
slope       int64
ca          int64
thal        int64
num         int64
dtype: object
```

In [33]:
```
print("Draw correlation matrix for feature selection")
corrmat = new_df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
heatmapResult=sns.heatmap(new_df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
plt.savefig('heatmapResult.png')
plt.show()
```

Draw correlation matrix for feature selection



In [34]:
```
print("Need to drop thalach due to feature engineering")
df = new_df.drop(['thalach'],axis=1)
df.head(10)
```

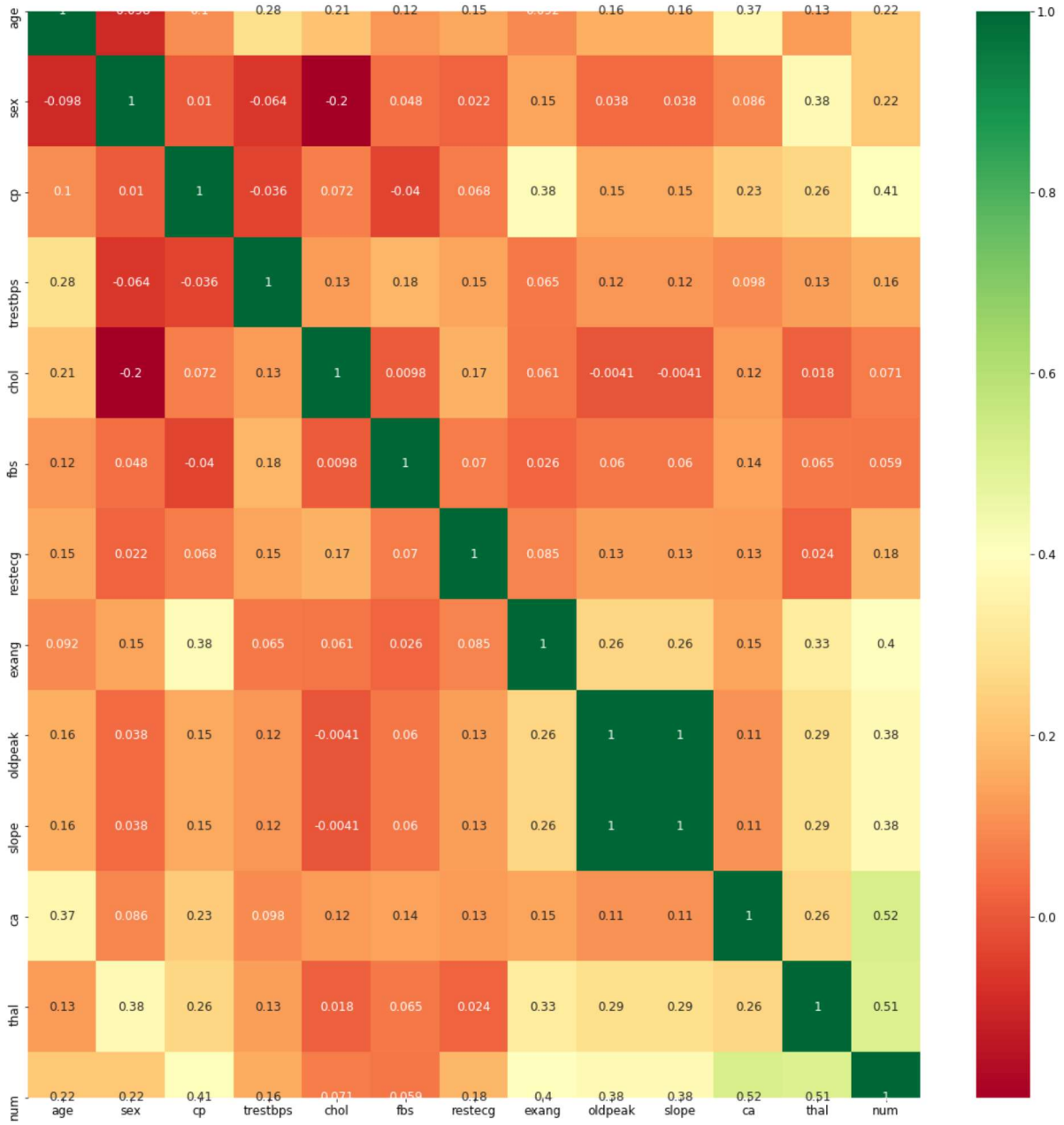
```
Need to drop thalach due to feature engineering
```

Out[34]:

	age	sex	cp	trestbps	chol	fbs	restecg	exang	oldpeak	slope	ca	thal	num
0	63	1	1	145	233	1	2	0	3	3	0	6	0
1	67	1	4	160	286	0	2	1	2	2	3	3	2
2	67	1	4	120	229	0	2	1	2	2	2	7	1
3	37	1	3	130	250	0	0	0	3	3	0	3	0
4	41	0	2	130	204	0	2	0	1	1	0	3	0
5	56	1	2	120	236	0	0	0	1	1	0	3	0
6	62	0	4	140	268	0	2	0	3	3	2	3	3
7	57	0	4	120	354	0	0	1	1	1	0	3	0
8	63	1	4	130	254	0	2	0	2	2	1	7	2
9	53	1	4	140	203	1	2	1	3	3	0	7	1

In [35]:
```
print("New correlation matrix for feature selection")
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
heatmapResultNew=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
plt.savefig('heatmapResultNew.png')
plt.show()
```

New correlation matrix for feature selection



In [36]:
```
print("Convert and categorise num into 0 or 1 and remove num")
heartdisease_map = {0:0,1:1,2:1,3:1,4:1}
df['heartdisease'] = df['num'].map(heartdisease_map)
df = df.drop(['num'],axis=1)
df.head(10)
```

```
Convert and categorise num into 0 or 1 and remove num
```

Out[36]:

	age	sex	cp	trestbps	chol	fbs	restecg	exang	oldpeak	slope	ca	thal	heartdisease
0	63	1	1	145	233	1	2	0	3	3	0	6	0
1	67	1	4	160	286	0	2	1	2	2	3	3	1
2	67	1	4	120	229	0	2	1	2	2	2	7	1
3	37	1	3	130	250	0	0	0	3	3	0	3	0
4	41	0	2	130	204	0	2	0	1	1	0	3	0
5	56	1	2	120	236	0	0	0	1	1	0	3	0
6	62	0	4	140	268	0	2	0	3	3	2	3	1
7	57	0	4	120	354	0	0	1	1	1	0	3	0
8	63	1	4	130	254	0	2	0	2	2	1	7	1
9	53	1	4	140	203	1	2	1	3	3	0	7	1

In [37]:
```
print("Splitting the data in training and testing")
x_data = df.drop('heartdisease', 1)
y_data = df['heartdisease']
x_train, x_test, y_train, y_test = train_test_split(x_data , y_data , test_siz
e = .2, random_state=10)
```

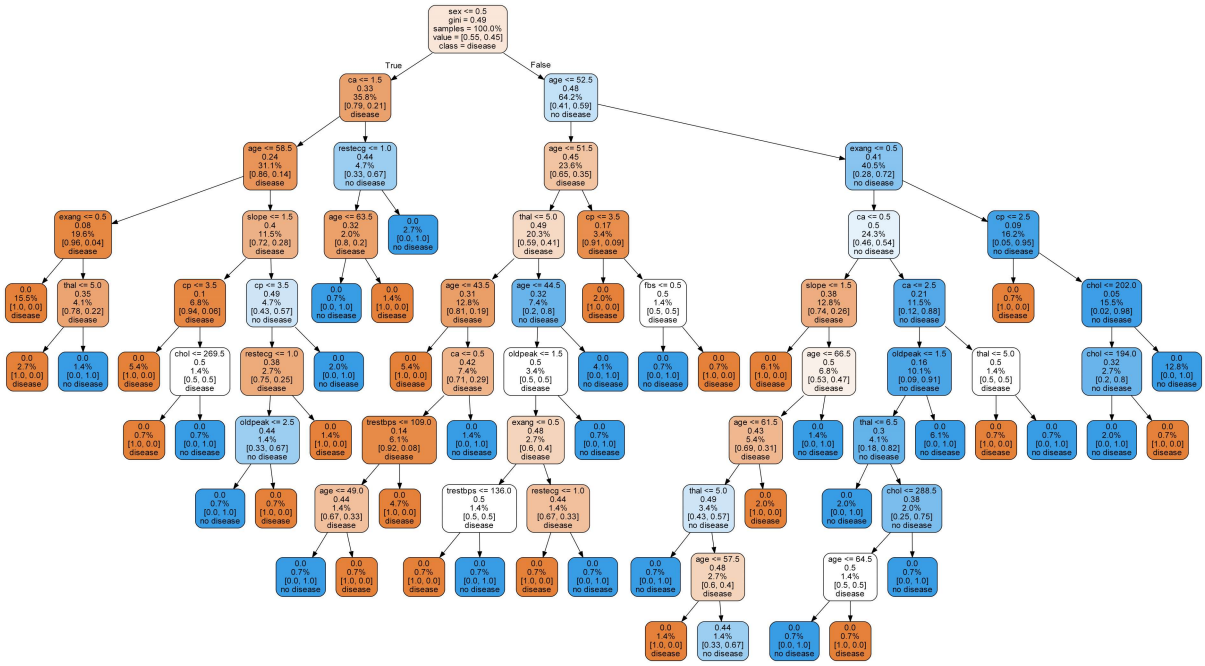
```
Splitting the data in training and testing
```

In [38]:
```
print("Training Random Forest Classification")
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators = 1000, random_state = 1,max_depth=9)
rf.fit(x_train, y_train)
accuracy = rf.score(x_test,y_test)*100
print("Random Forest Algorithm Accuracy Score : {:.2f}%".format(accuracy))
```

```
Training Random Forest Classification
Random Forest Algorithm Accuracy Score : 86.89%
```

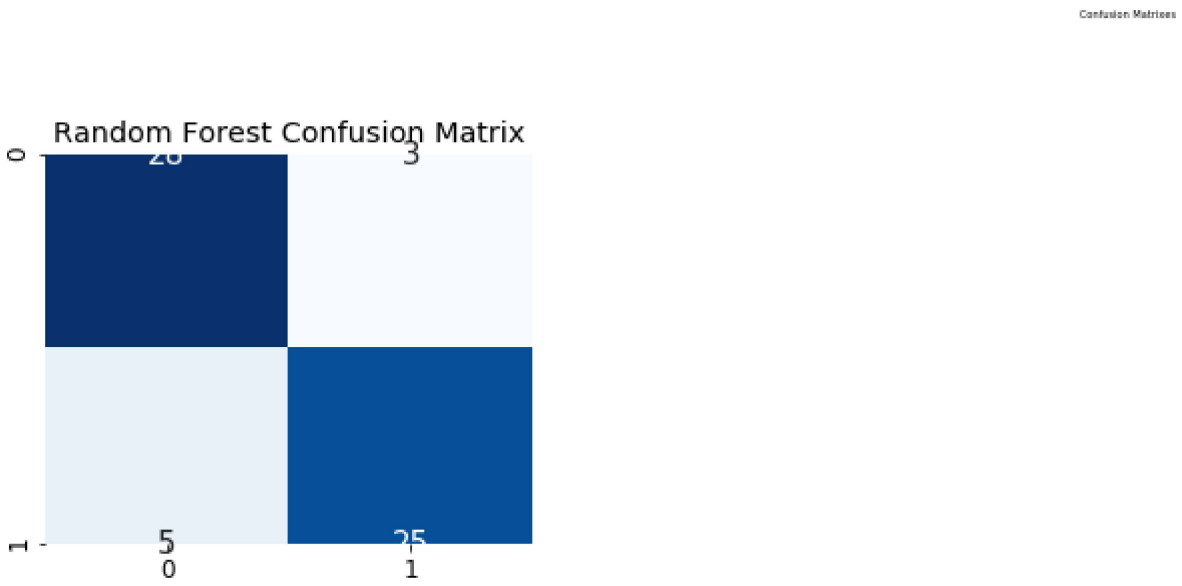
In [40]:
```
estimator = rf.estimators_[1]
feature_names = [i for i in x_train.columns]

y_train_str = y_train.astype('str')
y_train_str[y_train_str == '0'] = 'no disease'
y_train_str[y_train_str == '1'] = 'disease'
y_train_str = y_train_str.values

#code from https://towardsdatascience.com/how-to-visualize-a-decision-tree-fro
m-a-random-forest-in-python-using-scikit-learn-38ad2d75f21c

export_graphviz(estimator, out_file='tree.dot',
                feature_names = feature_names,
                class_names = y_train_str,
                rounded = True, proportion = True,
                label='root',
                precision = 2, filled = True)

from subprocess import call
call(['dot', '-Tpng', 'tree.dot', '-o', 'tree.png', '-Gdpi=600'])

from IPython.display import Image
Image(filename = 'tree.png')
```

Out[40]:



In [53]:
```
print("Plotting the confusion matrix")
y_head_rf = rf.predict(x_test)
cm_rf = confusion_matrix(y_test,y_head_rf)
y_pred_quant = rf.predict_proba(x_test)[:, 1]
print(cm_rf)
plt.figure(figsize=(20,10))
plt.suptitle("Confusion Matrixes",fontsize=5)
plt.subplots_adjust(wspace = 0.8, hspace= 0.8)
plt.subplot(2,3,1)
plt.title("Random Forest Confusion Matrix")
sns.heatmap(cm_rf,annot=True,cmap="Blues",fmt="d",cbar=False, annot_kws={"siz
e": 15})
plt.savefig('confusionMatrix.png')
plt.show()
```

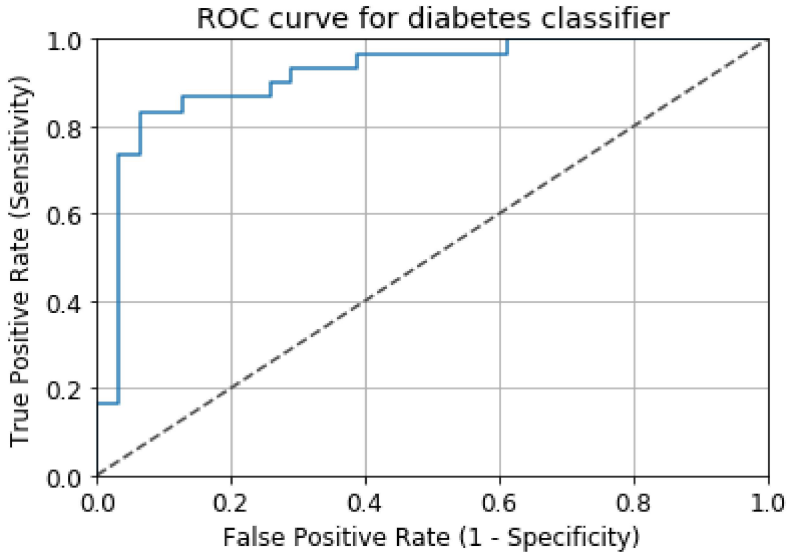
```
Plotting the confusion matrix
[[28  3]
 [ 5 25]]
```



In [42]:
```
print("Calculating the sensitivity and Specificity")
total=sum(sum(cm_rf))
sensitivity = cm_rf[0,0]/(cm_rf[0,0]+cm_rf[1,0])
print('Sensitivity : ', sensitivity )
specificity = cm_rf[1,1]/(cm_rf[1,1]+cm_rf[0,1])
print('Specificity : ', specificity)
```

```
Calculating the sensitivity and Specificity
Sensitivity :  0.8484848484848485
Specificity :  0.8928571428571429
```

In [43]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_quant)

fig, ax = plt.subplots()
ax.plot(fpr, tpr)
ax.plot([0, 1], [0, 1], transform=ax.transAxes, ls="--", c=".3")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.rcParams['font.size'] = 12
plt.title('ROC curve for diabetes classifier')
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
plt.grid(True)
```



In [44]:
```
auc(fpr, tpr)
```

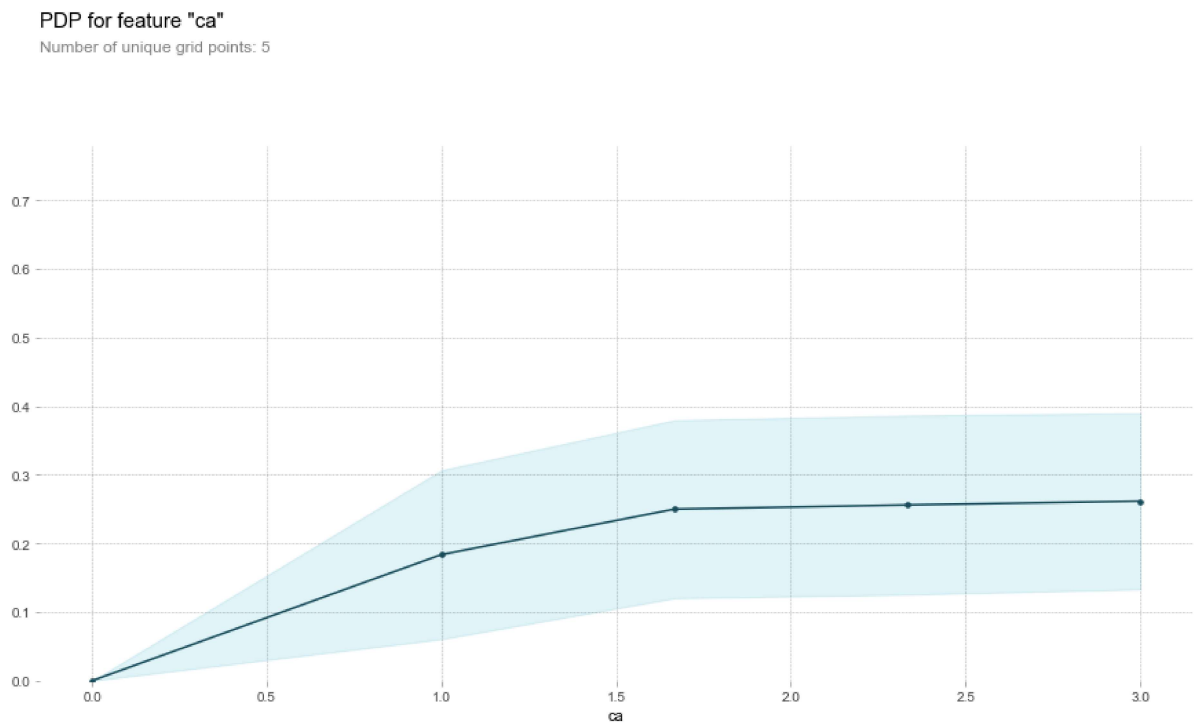
Out[44]:
```
0.9193548387096774
```

In [45]:
```
perm = PermutationImportance(rf, random_state=1).fit(x_test, y_test)
eli5.show_weights(perm, feature_names = x_test.columns.tolist())
```

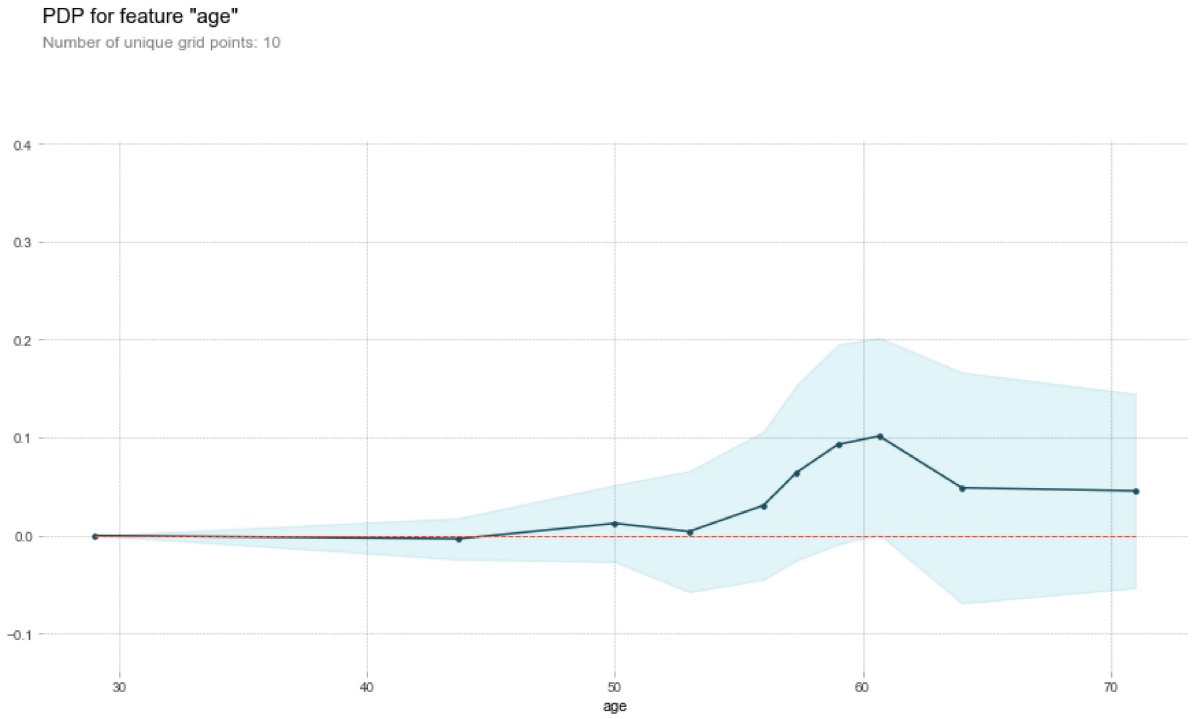
Out[45]:

Weight	Feature
0.0951 ± 0.0382	ca
0.0525 ± 0.0564	cp
0.0295 ± 0.0321	exang
0.0262 ± 0.0161	sex
0.0230 ± 0.0393	thal
0.0197 ± 0.0131	restecg
0.0197 ± 0.0435	age
0.0164 ± 0.0000	fbs
0.0000 ± 0.0293	oldpeak
-0.0000 ± 0.0587	trestbps
-0.0098 ± 0.0393	slope
-0.0131 ± 0.0245	chol

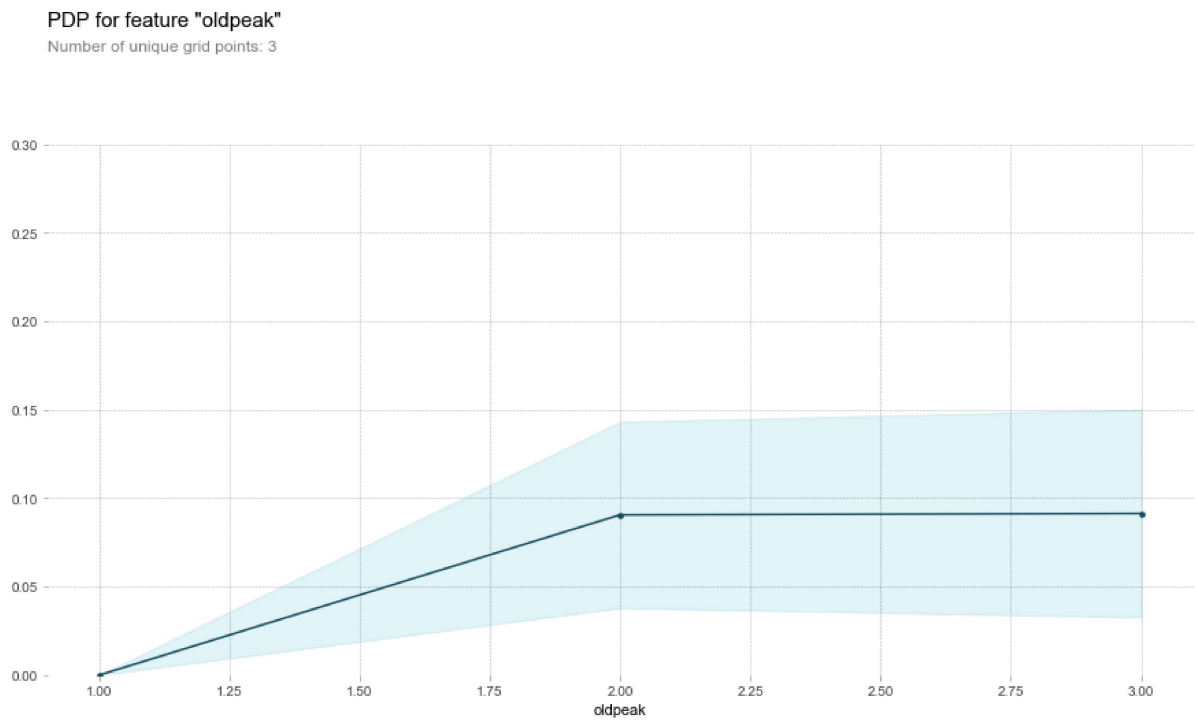
In [46]:
```
base_features = df.columns.values.tolist()
base_features.remove('heartdisease')

feat_name = 'ca'
pdp_dist = pdp.pdp_isolate(model=rf, dataset=x_test, model_features=base_featu
res, feature=feat_name)

pdp.pdp_plot(pdp_dist, feat_name)
plt.show()
```



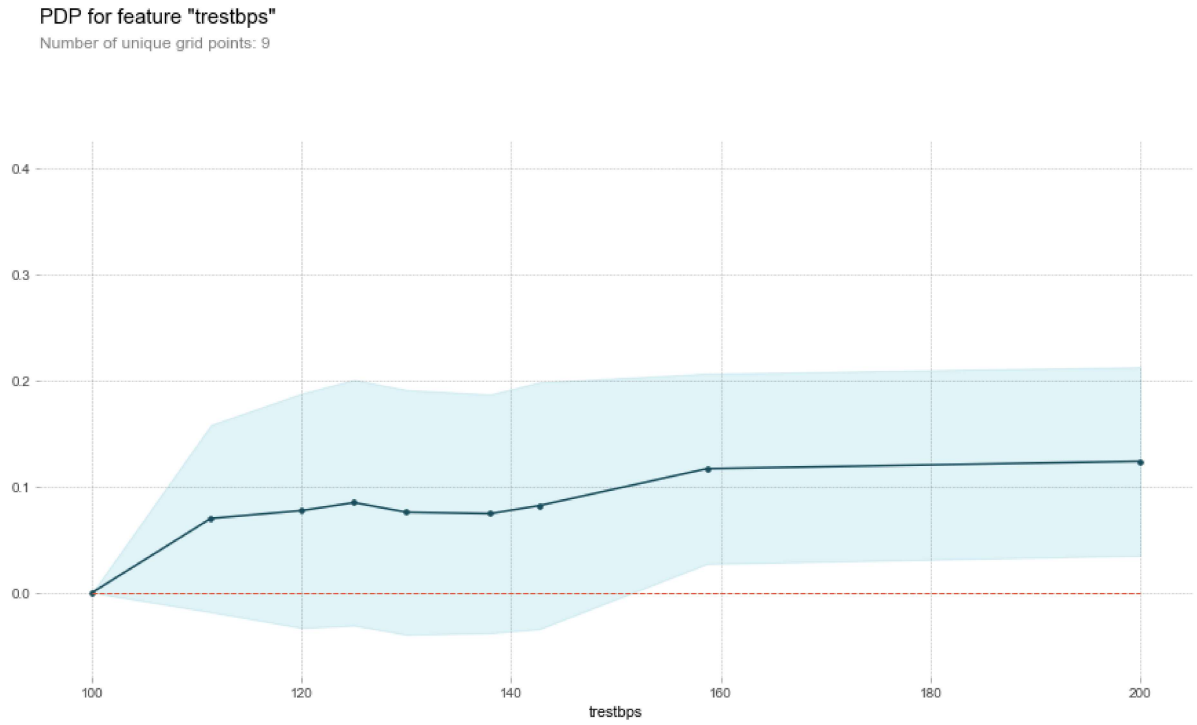
In [47]:
```
feat_name = 'age'
pdp_dist = pdp.pdp_isolate(model=rf, dataset=x_test, model_features=base_featu
res, feature=feat_name)

pdp.pdp_plot(pdp_dist, feat_name)
plt.show()
```



In [48]:
```
feat_name = 'oldpeak'
pdp_dist = pdp.pdp_isolate(model=rf, dataset=x_test, model_features=base_featu
res, feature=feat_name)

pdp.pdp_plot(pdp_dist, feat_name)
plt.show()
```



In [49]:
```
feat_name = 'trestbps'
pdp_dist = pdp.pdp_isolate(model=rf, dataset=x_test, model_features=base_featu
res, feature=feat_name)

pdp.pdp_plot(pdp_dist, feat_name)
plt.show()
```



In [ ]:
```

```