In [1]:

```
import os
os.chdir("D:\Data Science\Python")
```

In [2]:

```
os.getcwd()
```

Out[2]:

```
'D:\\Data Science\\Python'
```

In [71]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics
%matplotlib inline
```

In [78]:

```
data = pd.read_csv("bank-loan.csv")
```

In [79]:

```
data.head()
```

Out[79]:

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
0	41	3	17	12	176	9.3	11.359392	5.008608	1.0
1	27	1	10	6	31	17.3	1.362202	4.000798	0.0
2	40	1	15	14	55	5.5	0.856075	2.168925	0.0
3	41	1	15	14	120	2.9	2.658720	0.821280	0.0
4	24	2	2	0	28	17.3	1.787436	3.056564	1.0

In [80]:

```
data.isnull().sum()
```

Out[80]:

```
age           0
ed            0
employ        0
address       0
income        0
debtinc       0
creddebt      0
othdebt       0
default     150
dtype: int64
```

In [81]:

```
data['default'].isnull().value_counts()
```

Out[81]:

```
False    700
True     150
Name: default, dtype: int64
```

In [82]:

```
data['default'].fillna(data['default'].median(),inplace=True)
```

In [85]:

```
data.head()
```

Out[85]:

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
0	41	3	17	12	176	9.3	11.359392	5.008608	1.0
1	27	1	10	6	31	17.3	1.362202	4.000798	0.0
2	40	1	15	14	55	5.5	0.856075	2.168925	0.0
3	41	1	15	14	120	2.9	2.658720	0.821280	0.0
4	24	2	2	0	28	17.3	1.787436	3.056564	1.0

In [91]:

```
data.isnull().any()
```

Out[91]:

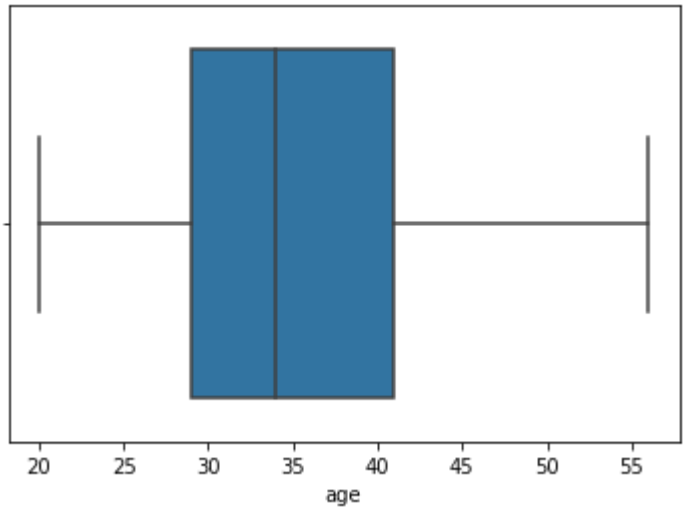
```
age         False
ed          False
employ      False
address     False
income      False
debtinc     False
creddebt    False
othdebt     False
default     False
dtype: bool
```

In [111]:

```
sns.boxplot(data['age'])
```

Out[111]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bca56e3688>
```

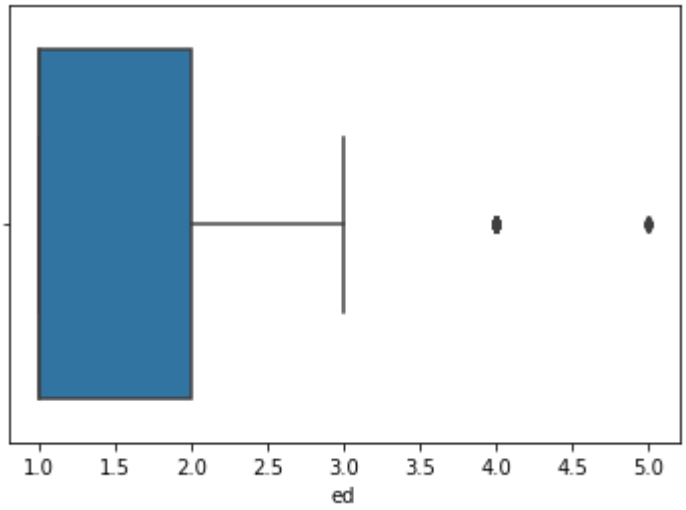


In [32]:

```
sns.boxplot(data['ed'])
```

Out[32]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bc9ddc23c8>
```

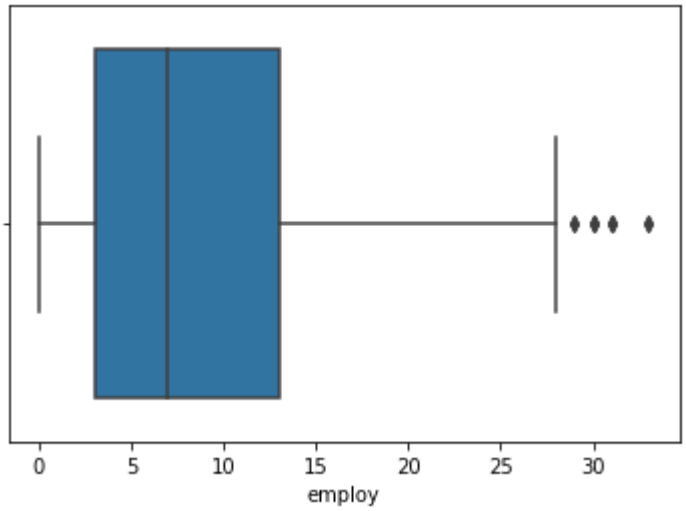


In [33]:

```
sns.boxplot(data['employ'])
```

Out[33]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bc9f133b48>
```

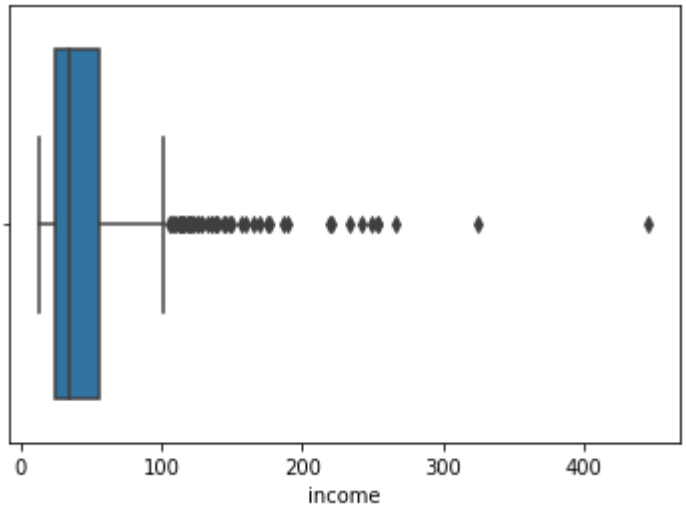


In [34]:

```
sns.boxplot(data['income'])
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bc9f1b1948>
```

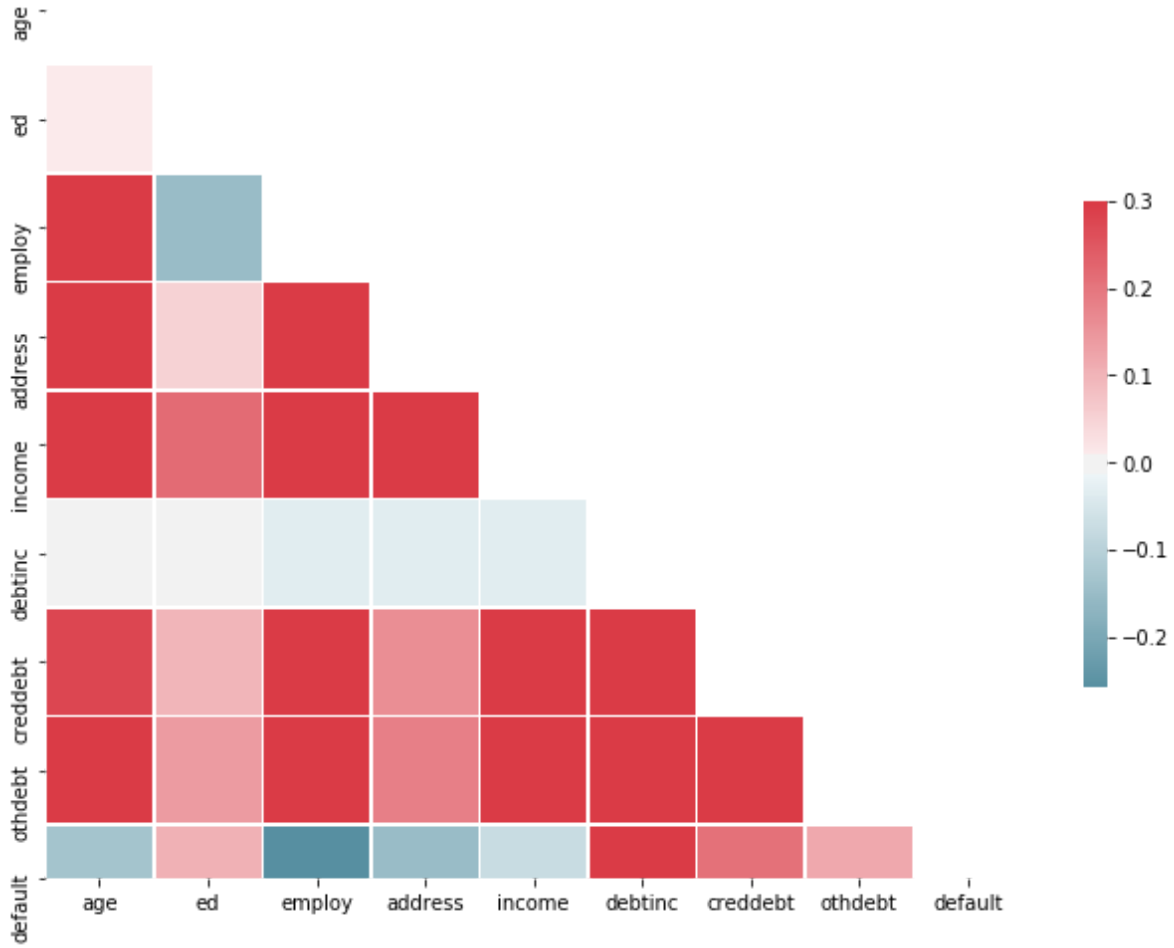


In [41]:

```
corr = data.corr()

# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True

f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

Out[41]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bca2bcb7c8>
```



In [42]:

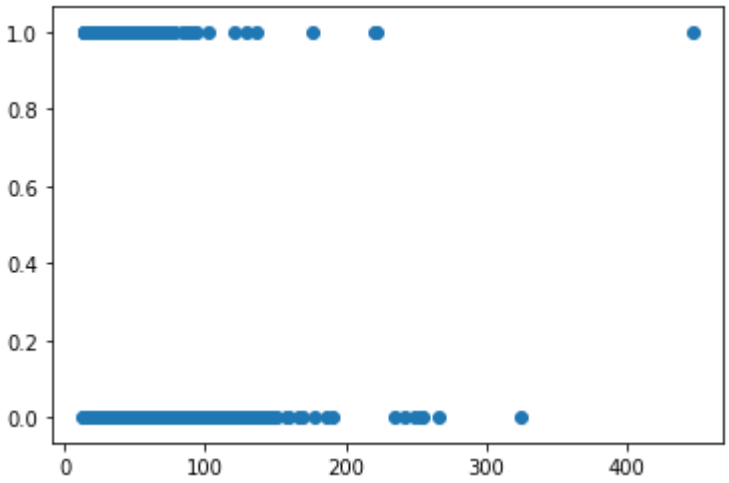
```
X = data[['income', 'debtinc','creddebt','othdebt']]
y = data['default']
```

In [43]:

```
plt.scatter(data['income'],data['default'] )
```

Out[43]:

```
<matplotlib.collections.PathCollection at 0x2bca2cc6588>
```

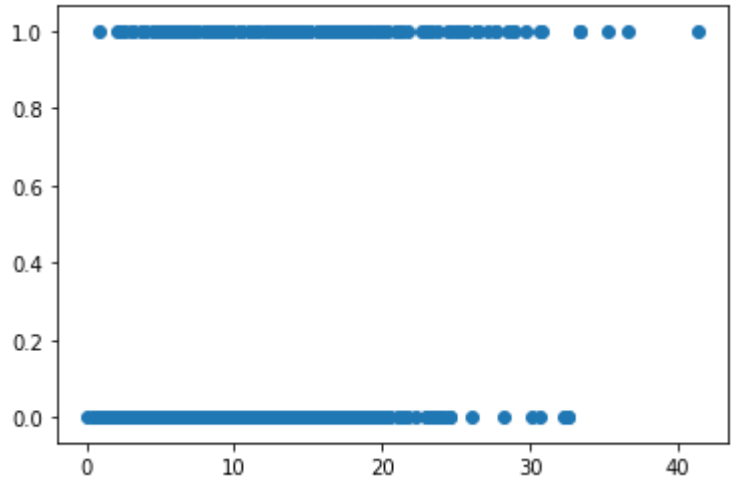


In [44]:

```
plt.scatter(data['debtinc'],data['default'] )
```

Out[44]:

```
<matplotlib.collections.PathCollection at 0x2bca2d5d748>
```

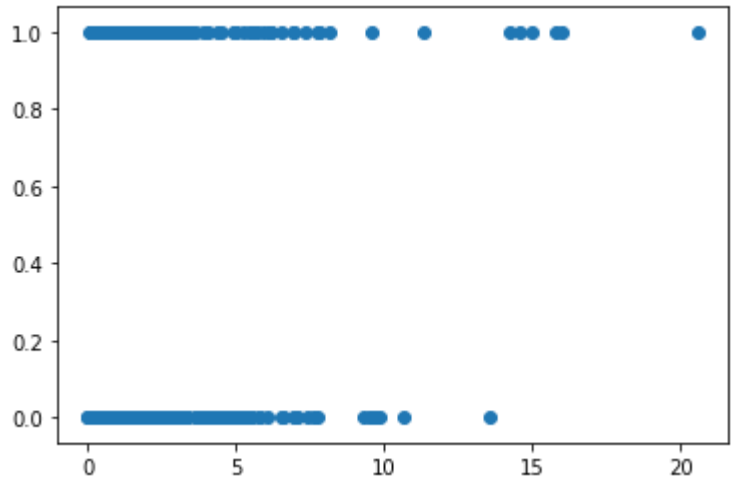


In [45]:

```
plt.scatter(data['creddebt'],data['default'] )
```

Out[45]:

```
<matplotlib.collections.PathCollection at 0x2bca2dbe648>
```

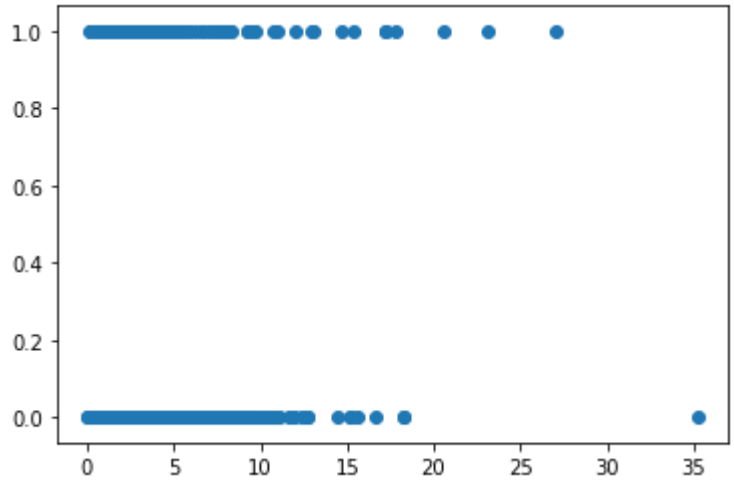


In [46]:

```
plt.scatter(data['othdebt'],data['default'] )
```

Out[46]:

```
<matplotlib.collections.PathCollection at 0x2bca2e1d808>
```



In [132]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2,random_state =10)
```

In [48]:

```
X_train.head()
```

Out[48]:

	income	debtinc	creddebt	othdebt
476	18	16.3	0.352080	2.581920
411	91	11.7	1.256346	9.390654
377	21	4.9	0.608139	0.420861
649	28	18.7	2.125816	3.110184
585	33	4.8	0.693792	0.890208

In [49]:

```
X_test.head()
```

Out[49]:

	income	debtinc	creddebt	othdebt
780	23	9.8	0.861028	1.392972
233	33	2.8	0.103488	0.820512
68	31	8.2	1.492154	1.049846
810	64	23.3	7.754240	7.157760
121	19	8.4	0.279300	1.316700

In [133]:

```
y_train.shape
```

Out[133]:

```
(680,)
```

In [87]:

```
X_test.shape
```

Out[87]:

```
(170, 4)
```

In [ ]:

```
#Logistic Regression Model Fitting
```

In [55]:

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
C:\Users\Userr\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:
432: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Speci
fy a solver to silence this warning.
  FutureWarning)
```

Out[55]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [ ]:

```
#Predicting the test set results and calculating the accuracy
```

In [155]:

```
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(
```

```
Accuracy of logistic regression classifier on test set: 0.78
```

In [56]:

```
#Confusion Matrix
```

In [152]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

```
[[123   8]
 [ 30   9]]
```

In [153]:

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

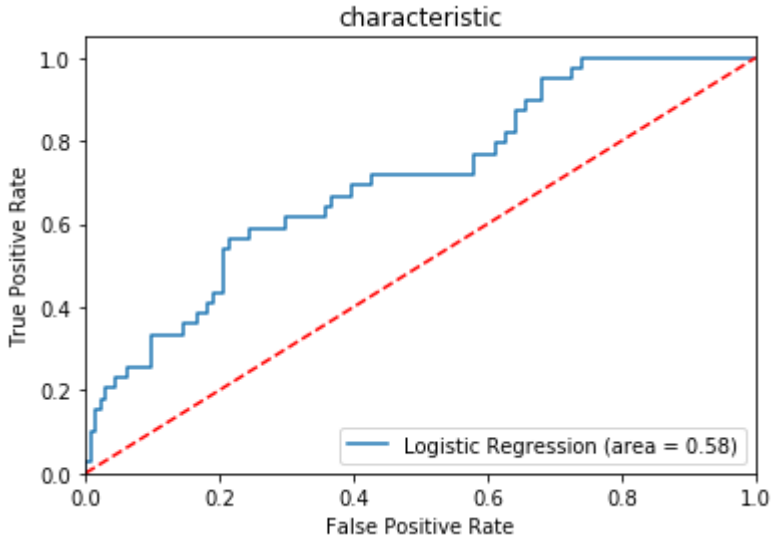
```
              precision    recall  f1-score   support

         0.0       0.80      0.94      0.87       131
         1.0       0.53      0.23      0.32        39

    accuracy                           0.78       170
   macro avg       0.67      0.58      0.59       170
weighted avg       0.74      0.78      0.74       170
```

In [ ]:

```
#ROC Curve
```

In [140]:

```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_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('characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```



In [ ]:

```
#Random Forest Classifier
```

In [144]:

```
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=250)

print('n_estimators is the number of trees to be used in the forest. Since Random Forest is

rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
n_estimators is the number of trees to be used in the forest. Since Random F
orest is an ensemble method comprising of creating multiple decision trees,
this parameter is used to control the number of trees to be used in the proc
ess
              precision    recall  f1-score   support

         0.0       0.83      0.88      0.85       131
         1.0       0.48      0.38      0.43        39

    accuracy                           0.76       170
   macro avg       0.66      0.63      0.64       170
weighted avg       0.75      0.76      0.75       170
```

In [ ]: