In [1]:

```python
import os
```

In [2]:

```python
import os
os.chdir("D:/Certificates/Data Science/Edwisor/Project 1")
```

In [3]:

```python
# check current working directory
os.getcwd()
```

Out[3]:

```
'D:\\Certificates\\Data Science\\Edwisor\\Project 1'
```

In [4]:

```python
print(os.listdir(os.getcwd()))
```

```
['.RData', '.Rhistory', 'bank-loan.csv', 'DataN0108 (1).pdf', 'In.doc
x', 'Project 1.csv', 'project report_.pdf', 'Project.docx', 'Python cod
ing.docx', 'R Coding.docx', 'R.R']
```

In [5]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
%matplotlib inline
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn import svm
from sklearn.preprocessing import scale
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import auc
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.decomposition import PCA
from sklearn.ensemble import GradientBoostingClassifier
```

In [6]:
```
bank =pd.read_csv("bank-loan.csv")
```

In [7]:
```
bank.columns
```

Out[7]:
```
Index(['age', 'ed', 'employ', 'address', 'income', 'debtinc', 'creddeb
t',
       'othdebt', 'default'],
      dtype='object')
```

In [8]:
```
bank.head(5)
```

Out[8]:

	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 [9]:
```
bank.tail(5)
```

Out[9]:

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
845	34	1	12	15	32	2.7	0.239328	0.624672	NaN
846	32	2	12	11	116	5.7	4.026708	2.585292	NaN
847	48	1	13	11	38	10.8	0.722304	3.381696	NaN
848	35	2	1	11	24	7.8	0.417456	1.454544	NaN

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
849	37	1	20	13	41	12.9	0.899130	4.389870	NaN

In [10]:

```
bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 850 entries, 0 to 849
Data columns (total 9 columns):
age         850 non-null int64
ed          850 non-null int64
employ      850 non-null int64
address     850 non-null int64
income      850 non-null int64
debtinc     850 non-null float64
creddebt    850 non-null float64
othdebt     850 non-null float64
default     700 non-null float64
dtypes: float64(4), int64(5)
memory usage: 59.9 KB
```

In [11]:

```
bank.shape
```

Out[11]:

```
(850, 9)
```

In [12]:

```
bank.describe()
```

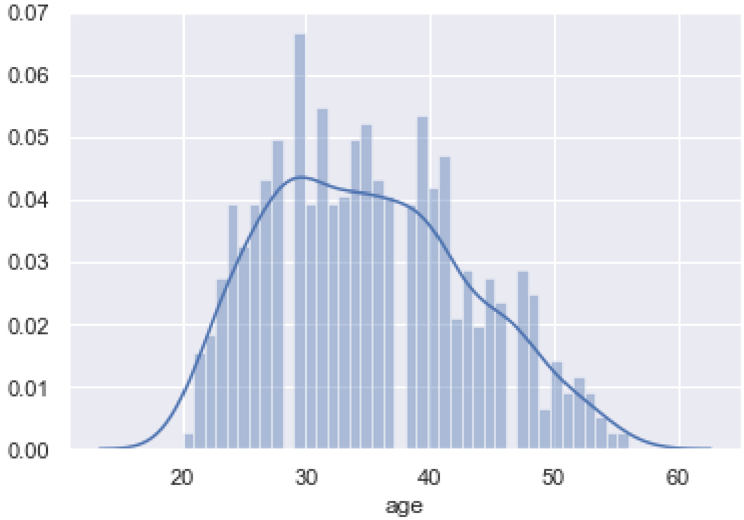
Out[12]:

	age	ed	employ	address	income	debtinc	creddebt	o
count	850.000000	850.000000	850.000000	850.000000	850.000000	850.000000	850.000000	850.0
mean	35.029412	1.710588	8.565882	8.371765	46.675294	10.171647	1.576805	3.0
std	8.041432	0.927784	6.777884	6.895016	38.543054	6.719441	2.125840	3.3
min	20.000000	1.000000	0.000000	0.000000	13.000000	0.100000	0.011696	0.0
25%	29.000000	1.000000	3.000000	3.000000	24.000000	5.100000	0.382176	1.0
50%	34.000000	1.000000	7.000000	7.000000	35.000000	8.700000	0.885091	2.0

	age	ed	employ	address	income	debtinc	creddebt	o
75%	41.000000	2.000000	13.000000	12.000000	55.750000	13.800000	1.898440	3.9
max	56.000000	5.000000	33.000000	34.000000	446.000000	41.300000	20.561310	35.1

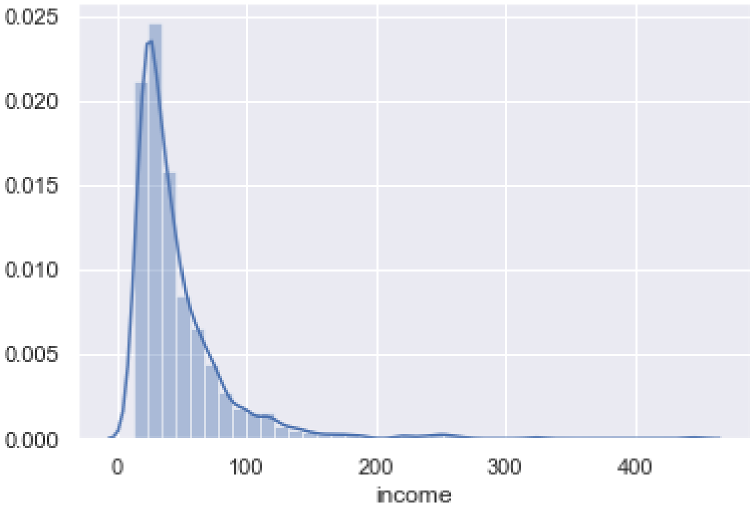
In [13]:
```
############ univariate analysis and bivariate analysis ###############
###########
# analysis for single variable in the dataset and relation between 2 va
riables.
sns.distplot(bank["age"], kde = True , bins = 40)
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2797f155a88>
```



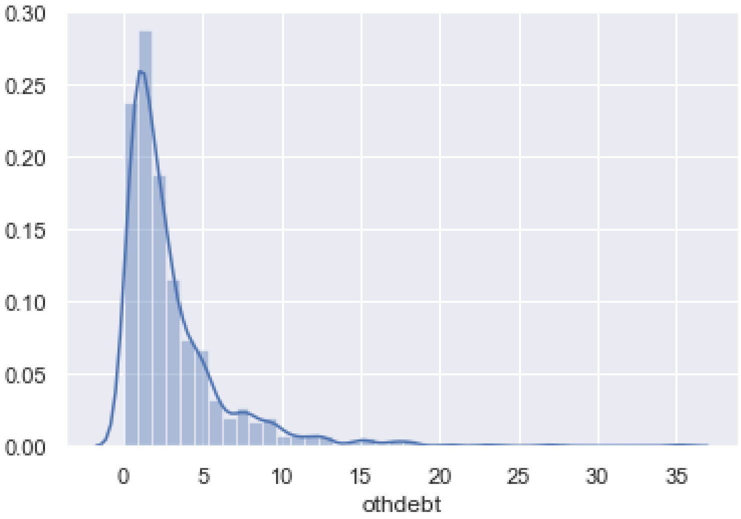
In [14]:
```
sns.distplot(bank["income"], kde = True , bins = 40)
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2797f4e8e48>
```



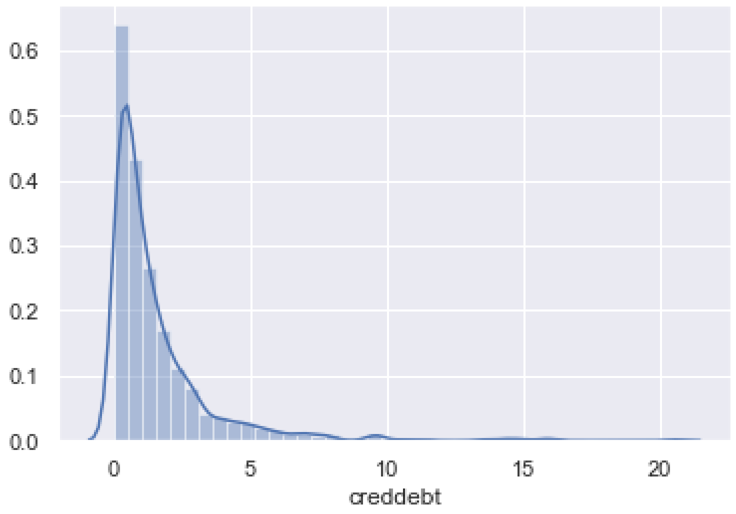
```
In [15]: sns.distplot(bank["othdebt"], kde = True , bins = 40)
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x2797f5c2288>
```



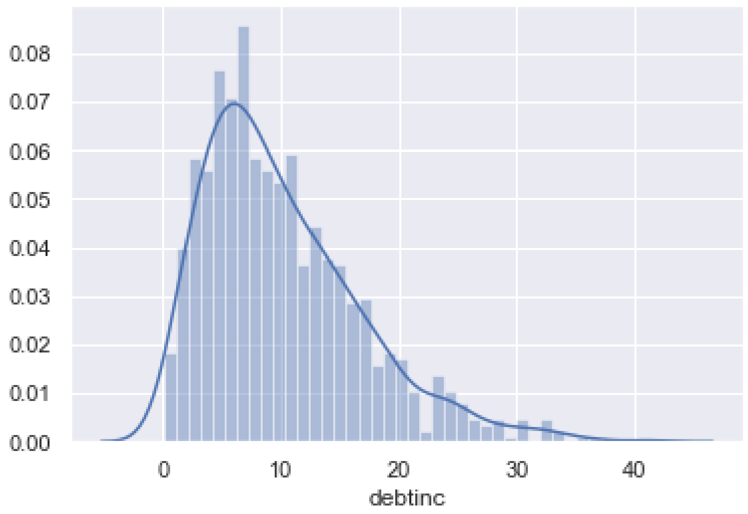
```
In [16]: sns.distplot(bank["creddebt"], kde = True , bins = 40)
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x2797f5dd548>
```



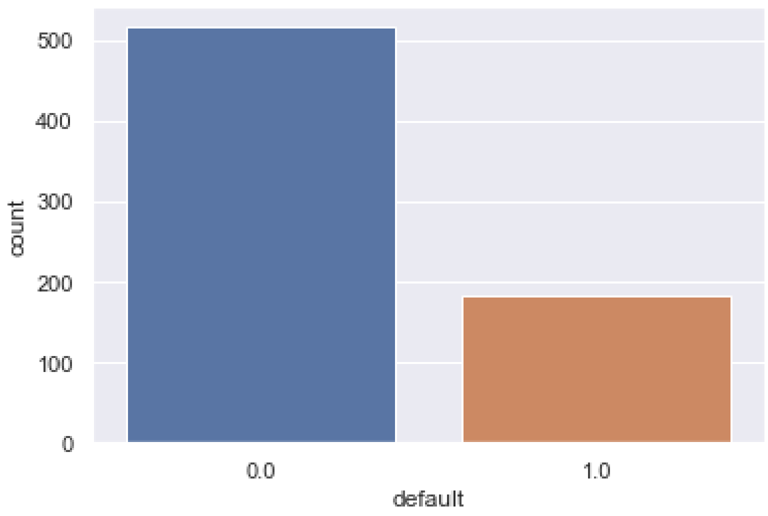
```
In [17]: sns.distplot(bank["debtinc"], kde = True , bins = 40)
```

```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x2797f75e0c8>
```



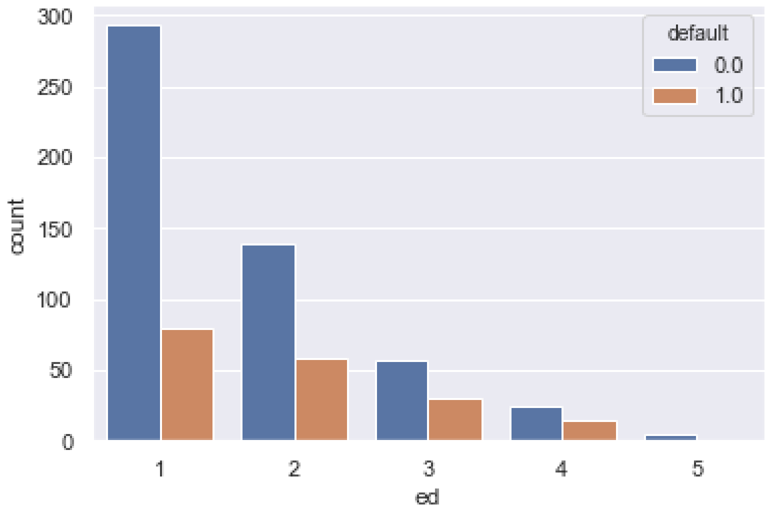
In [18]:
```
sns.countplot( x = "default" , data = bank)
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2797f845a48>
```



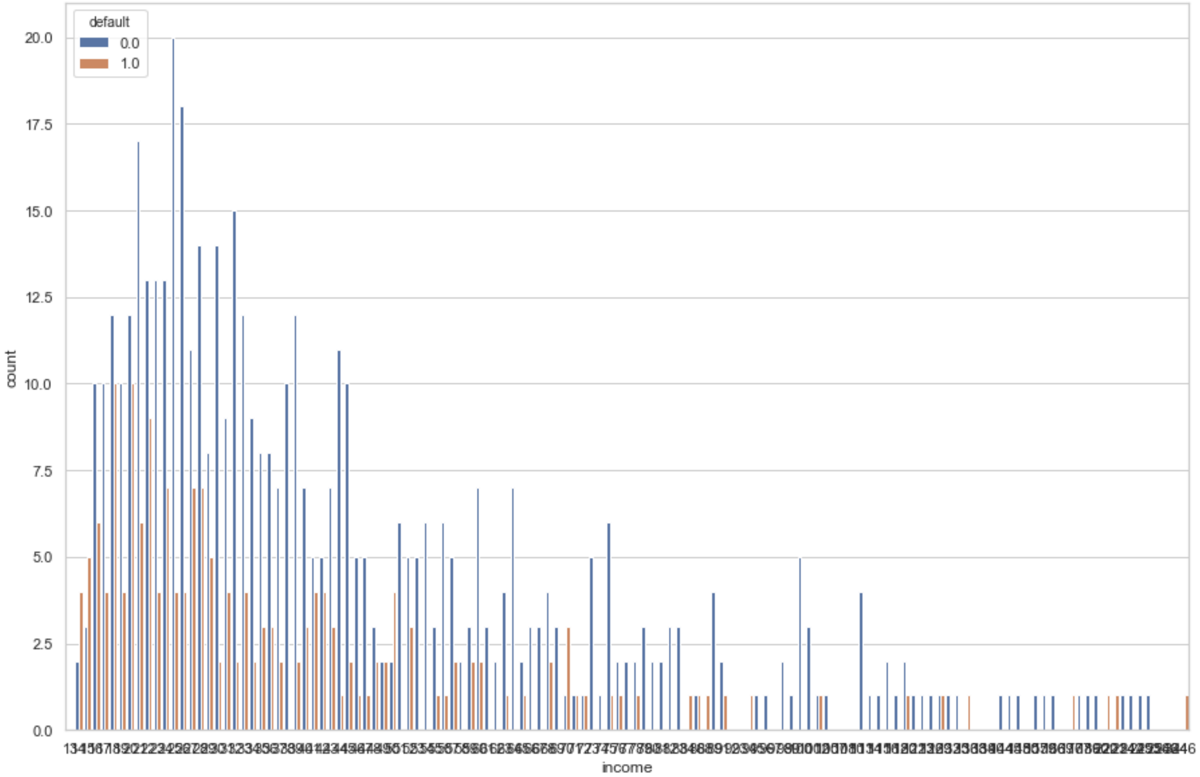
In [19]:
```
sns.countplot( x = "ed" , data = bank, hue = "default")
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2797f8b9308>
```



In [20]:
```
sns.set_style("whitegrid")
plt.figure(figsize = (15,10))
sns.countplot( x = "income" , data = bank, hue = "default")
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2797f94ed88>
```



In [21]:
```
############# missing value analysis ################

#Calculating the null values in the dataframe
missing_value = pd.DataFrame(bank.isnull().sum())
missing_value = (missing_value/len(bank))*100
missing_value.reset_index()

missing_value = missing_value.rename(columns = {'index': 'Variables', 0
: 'Missing_percentage'})
#Arranging Missing Values in Decreasing Order
missing_value = missing_value.sort_values('Missing_percentage', ascendi
ng = False)
#save output results
missing_value.to_csv("Missing_perc.csv", index = False)
missing_value
```

Out[21]:

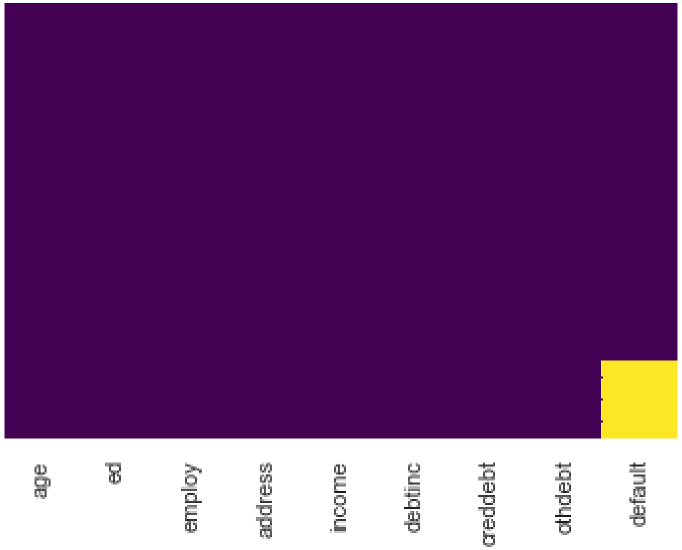
	Missing_percentage
default	17.647059
age	0.000000
ed	0.000000
employ	0.000000
address	0.000000
income	0.000000
debtinc	0.000000
creddebt	0.000000
othdebt	0.000000

In [22]:

```
sns.heatmap(bank.isnull(),yticklabels=False,cbar=False,cmap="viridis")
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2797fd57508>
```

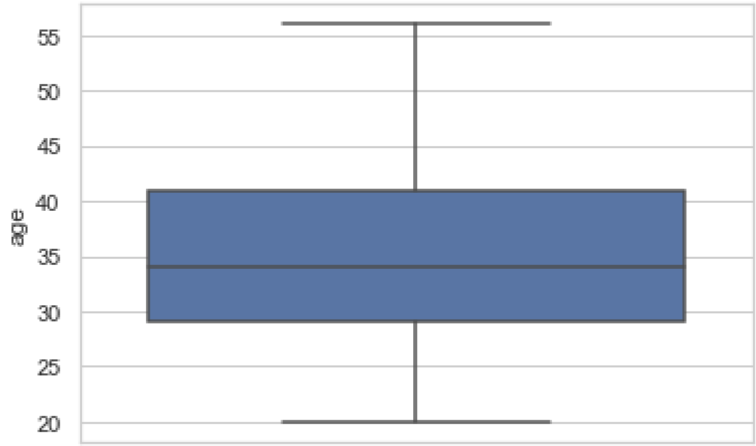


In [23]:
```
bank.isnull().sum()
```

Out[23]:
```
age           0
ed            0
employ        0
address       0
income        0
debtinc       0
creddebt      0
othdebt       0
default     150
dtype: int64
```

In [24]:
```
sns.boxplot( x = "age" , data = bank , orient = "v")
```

Out[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2790005b488>
```



In [25]:

```
bank.default=bank.default.fillna(2)
```

In [26]:

```
bank.default=bank.default.astype(int)
```

In [27]:

```
bank.head(2)
```

Out[27]:

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
0	41	3	17	12	176	9.3	11.359392	5.008608	1
1	27	1	10	6	31	17.3	1.362202	4.000798	0

In [28]:

```
bank.tail(2)
```

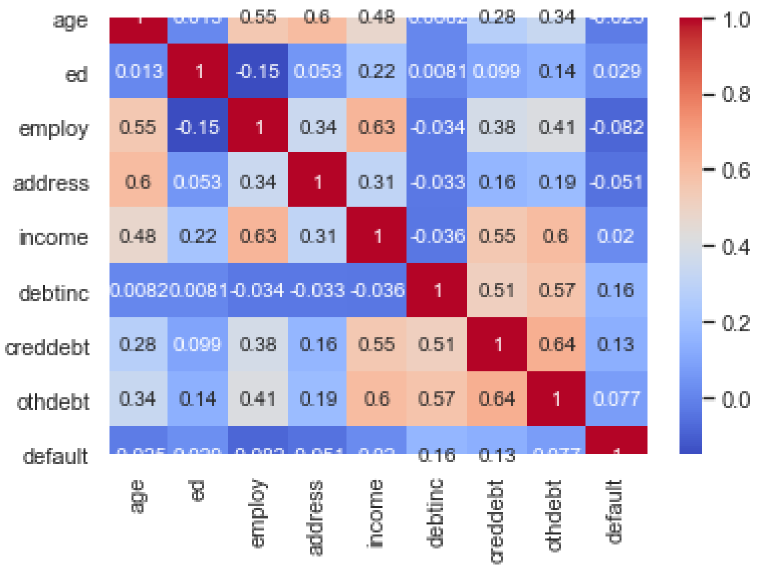
Out[28]:

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
848	35	2	1	11	24	7.8	0.417456	1.454544	2
849	37	1	20	13	41	12.9	0.899130	4.389870	2

In [29]:

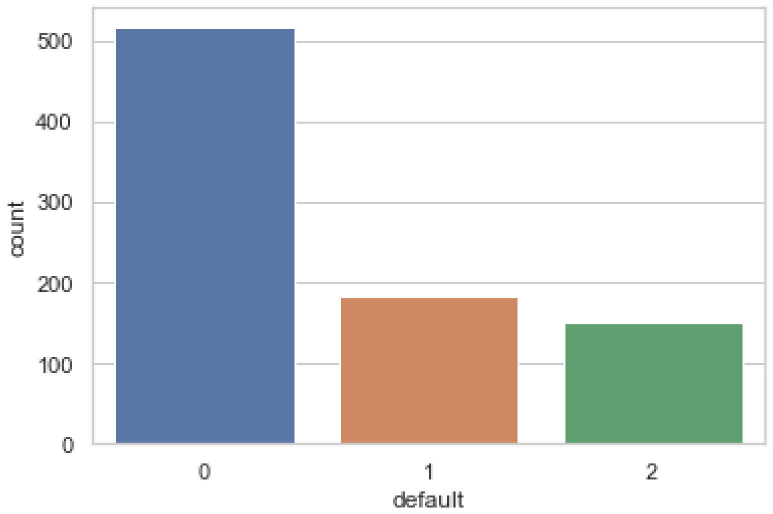
```
df_1 = bank.corr()
sns.heatmap(df_1 , annot = True , cmap = "coolwarm")
```

```
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x279000d3e08>
```



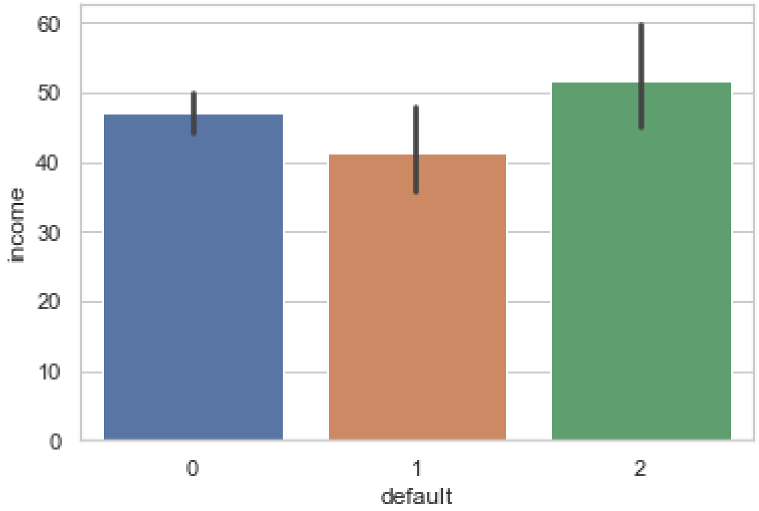
```
In [30]: sns.countplot(bank.default)
```

```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x2790020d188>
```



In [31]:
```
sns.barplot(x='default',y='income',data=bank)
```

Out[31]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2790026fc08>
```



In [32]:
```
#### selecting all missing values from dataset and we will predict thos
e default case with best accurate
##machine learning algorithm.
train=bank.loc[bank['default']!=2]
print(train.head(5))
print(train.tail(5))
```

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

```
     age  ed  employ  address  income  debtinc  creddebt   othdebt  def
ault
695   36   2       6       15      27      4.6  0.262062  0.979938
   1
696   29   2       6        4      21     11.5  0.369495  2.045505
   0
697   33   1      15        3      32      7.6  0.491264  1.940736
   0
698   45   1      19       22      77      8.4  2.302608  4.165392
   0
699   37   1      12       14      44     14.7  2.994684  3.473316
   0
```

In [33]:
```
train.default.unique()
```

Out[33]:
```
array([1, 0], dtype=int64)
```

In [34]:
```
test=bank.loc[bank.default==2]
test=test.iloc[:,0:8]
print(test.head(2))
print(test.tail(2))
```

```
     age  ed  employ  address  income  debtinc  creddebt   othdebt
700   36   1      16       13      32     10.9  0.544128  2.943872
701   50   1       6       27      21     12.9  1.316574  1.392426
     age  ed  employ  address  income  debtinc  creddebt   othdebt
848   35   2       1       11      24      7.8  0.417456  1.454544
849   37   1      20       13      41     12.9  0.899130  4.389870
```

In [35]:
```
X =train[['age', 'ed', 'employ', 'address', 'income', 'debtinc', 'credd
ebt',
       'othdebt']]
y = train['default']
```

In [36]:
```
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, random_state=42,
test_size=0.25)
```

```
In [37]: from sklearn.tree import DecisionTreeClassifier
```

```
In [38]: clf_gini = DecisionTreeClassifier(criterion = "gini", random_state = 10
         ,
                                         max_depth=5, min_samples_leaf=7)
         clf_gini.fit(Xtrain, ytrain)
```

```
Out[38]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=
         5,
                                max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=No
         ne,
                                min_samples_leaf=7, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort=False,
                                random_state=10, splitter='best')
```

```
In [39]: dt_pred = clf_gini.predict(Xtest)
```

```
In [40]: dt_pred1 = clf_gini.predict_proba(Xtest)[:,1]
```

```
In [41]: accuracy_score(ytest, dt_pred)
```

```
Out[41]: 0.7828571428571428
```

```
In [42]: print(classification_report(ytest, dt_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.87      0.86       132
           1       0.56      0.51      0.54        43

    accuracy                           0.78       175
   macro avg       0.70      0.69      0.70       175
weighted avg       0.78      0.78      0.78       175
```

```
In [43]: precision_dc, recall_dc, thresholds_dc = precision_recall_curve(ytest,
dt_pred1)
```

In [44]:
```
fpr_dc, tpr_dc, thresholds_dc = roc_curve(ytest, dt_pred1)
```

In [45]:
```
############ Random Forest#####
clf_rf = RandomForestClassifier(random_state=42)
```

In [46]:
```
clf_rf.fit(Xtrain, ytrain)
```

```
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\ensemble\fores
t.py:245: FutureWarning: The default value of n_estimators will change
from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[46]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gi
ni',
                       max_depth=None, max_features='auto', max_leaf_no
des=None,
                       min_impurity_decrease=0.0, min_impurity_split=No
ne,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=42, v
erbose=0,
                       warm_start=False)
```

In [47]:
```
y_pred_rf = clf_rf.predict(Xtest)
```

In [48]:
```
cv_scores = cross_val_score(clf_rf, Xtrain, ytrain, cv = 5)
print("Average 5-Fold CV Score: {}".format(np.mean(cv_scores)))
```

```
Average 5-Fold CV Score: 0.7714285714285715
```

In [49]:
```
cv_scores = cross_val_score(clf_rf, Xtrain, ytrain, cv = 5, scoring =
'roc_auc')
print("Average 5-Fold CV Score using ROC scoring: {}".format(np.mean(cv
_scores)))
```

```
Average 5-Fold CV Score using ROC scoring: 0.7528293135435994
```

In [50]:
```
accuracy_score(ytest, y_pred_rf)
```

Out[50]:
```
0.7542857142857143
```

In [51]:
```
n_space = np.array([5, 6, 10, 12, 15, 50, 100, 200, 500])
criterion_vals = ['gini', 'entropy']
max_features_vals = ['auto', 'sqrt', 'log2']
min_samples_leaf_sp = [1,5,10,25,50]
bootstrap_sp = [True, False]


param_grid = {'n_estimators': n_space, 'criterion' : criterion_vals,
              'max_features':max_features_vals, 'min_samples_leaf': min
_samples_leaf_sp,
              'bootstrap': bootstrap_sp}
```

In [52]:
```
rf_clf_tuning = GridSearchCV(clf_rf, param_grid, cv=5)
```

In [53]:
```
rf_clf_tuning.fit(Xtrain, ytrain)
```

Out[53]:
```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=RandomForestClassifier(bootstrap=True, class_wei
ght=None,
                                              criterion='gini', max_dep
th=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.
0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=
0.0,
                                              n_estimators=10, n_jobs=N
one,
                                                   oob_score=False, random_s
tate=42,
                                                   verbose=0, warm_start=Fal
se),
             iid='warn', n_jobs=None,
             param_grid={'bootstrap': [True, False],
                         'criterion': ['gini', 'entropy'],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [1, 5, 10, 25, 50],
                         'n_estimators': array([  5,   6,  10,  12,  1
5,  50, 100, 200, 500])},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=Fa
lse,
             scoring=None, verbose=0)
```

In [54]:

```
print("Tuned RF Parameters: {}".format(rf_clf_tuning.best_params_))
print("Best score is {}".format(rf_clf_tuning.best_score_))
```

```
Tuned RF Parameters: {'bootstrap': True, 'criterion': 'gini', 'max_feat
ures': 'log2', 'min_samples_leaf': 5, 'n_estimators': 200}
Best score is 0.8019047619047619
```

In [64]:

```
best_rf_clf = RandomForestClassifier(criterion = 'gini', bootstrap = Tr
ue,
                                     max_features = 'log2', min_samples
_leaf = 5, n_estimators = 200)
```

In [65]:

```
best_rf_clf.fit(Xtrain, ytrain)
```

Out[65]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gi
ni',
                       max_depth=None, max_features='log2', max_leaf_no
des=None,
                       min_impurity_decrease=0.0, min_impurity_split=No
ne,
                       min_samples_leaf=5, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=200,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
In [66]: y_best_rf_preds = best_rf_clf.predict(Xtest)
```

```
In [67]: cv_scores = cross_val_score(best_rf_clf, Xtrain, ytrain, cv = 5)
         print("Average 5-Fold CV Score: {}".format(np.mean(cv_scores)))
```

```
Average 5-Fold CV Score: 0.7885714285714286
```

```
In [68]: accuracy_score(ytest, y_best_rf_preds)
```

```
Out[68]: 0.8114285714285714
```

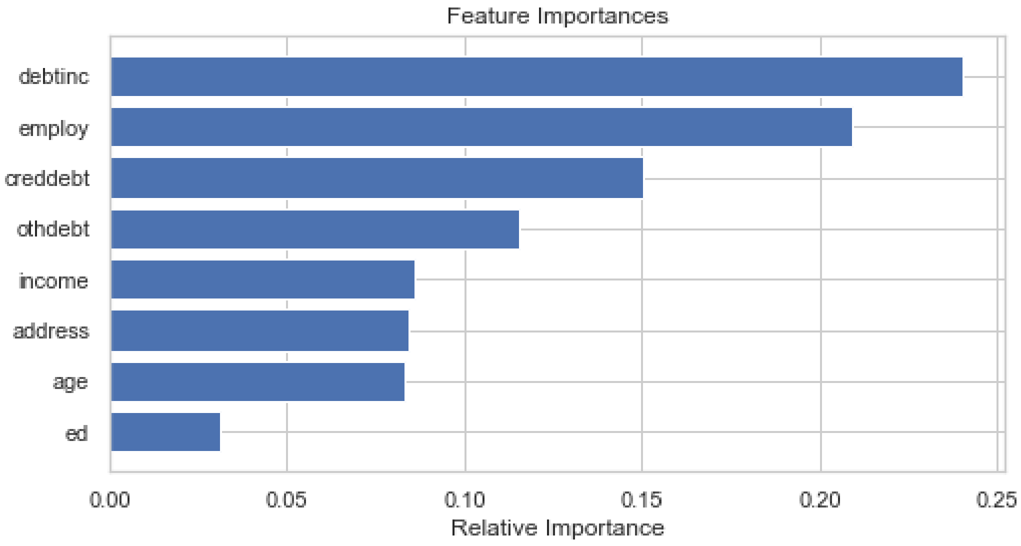
```
In [69]: y_best_rf_probas = best_rf_clf.predict_proba(Xtest)[:,1]
```

```
In [70]: print(classification_report(ytest, (y_best_rf_probas > 0.5).astype(int
         )))
```

```
              precision    recall  f1-score   support

           0       0.84      0.93      0.88       132
           1       0.68      0.44      0.54        43

    accuracy                           0.81       175
   macro avg       0.76      0.69      0.71       175
weighted avg       0.80      0.81      0.80       175
```

```
In [71]: fig, ax = plt.subplots(figsize=(8,4))
         features = train.columns
         importances = best_rf_clf.feature_importances_
         indices = np.argsort(importances)

         plt.title('Feature Importances')
         plt.barh(range(len(indices)), importances[indices], color='b', align='c
         enter')
```

```
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



In [72]:
```
y_rf_probs = clf_rf.predict_proba(Xtest)
```

In [73]:
```
precision_rf, recall_rf, thresholds_rf = precision_recall_curve(ytest,
y_best_rf_probas)
```

In [74]:
```
fpr_rf, tpr_rf, thresholds_rf = roc_curve(ytest, y_best_rf_probas)
```

In [ ]:
```
##################logistic regression#################
basic approach to classification in supervised learning. Assumptions =
that data has no outliers, there are two classes to be predicted, and t
hat no two independent variables are highly correlated to each other.
```

In [75]:
```
clf_log = LogisticRegression()
```

In [76]:
```
train_scale = scale(train)
```

In [77]:
```
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, random_state=42,
test_size=0.25)
```

In [78]:
```
clf_log.fit(Xtrain, ytrain)
```

```
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[78]:
```
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, verbos
e=0,
                   warm_start=False)
```

In [79]:
```
y_log_pred = clf_log.predict(Xtest)
```

In [80]:
```
accuracy_score(ytest, y_log_pred)
```

Out[80]:
```
0.8571428571428571
```

In [81]:
```
C_space = np.array([0.0001, 0.001, 0.1, 1])
```

In [82]:
```
param_grid = {'C': C_space}
```

In [83]:
```
clf_log_tuning = GridSearchCV(clf_log, param_grid, cv=5)
```

In [84]:
```
clf_log_tuning.fit(Xtrain, ytrain)
```

```
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

```
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
```

```
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
```

```
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[84]:

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=LogisticRegression(C=1.0, class_weight=None, dua
l=False,
                                          fit_intercept=True,
                                          intercept_scaling=1, l1_ratio
=None,
                                          max_iter=100, multi_class='wa
rn',
                                          n_jobs=None, penalty='l2',
                                          random_state=None, solver='wa
rn',
                                          tol=0.0001, verbose=0,
                                          warm_start=False),
             iid='warn', n_jobs=None,
             param_grid={'C': array([1.e-04, 1.e-03, 1.e-01, 1.e+00])},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=Fa
lse,
             scoring=None, verbose=0)
```

In [85]:

```
print("Tuned Logistic Regression Parameters: {}".format(clf_log_tuning
.best_params_))
print("Best score is {}".format(clf_log_tuning .best_score_))
```

```
Tuned Logistic Regression Parameters: {'C': 1.0}
Best score is 0.7904761904761904
```

In [86]:

```
clf_log = LogisticRegression(C = 1.0)
```

In [87]:

```
clf_log.fit(Xtrain, ytrain)
```

```
C:\Users\chinnababu\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[87]:

```
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, verbos
e=0,
                                   warm_start=False)
```

```
In [88]: y_preds = clf_log.predict(Xtest)
```

```
In [89]: p_clf_log_ba = clf_log.predict_proba(Xtest)
```

```
In [90]: accuracy_score(ytest, y_preds)
```

```
Out[90]: 0.8571428571428571
```

```
In [91]: print(classification_report(ytest, y_preds))
```

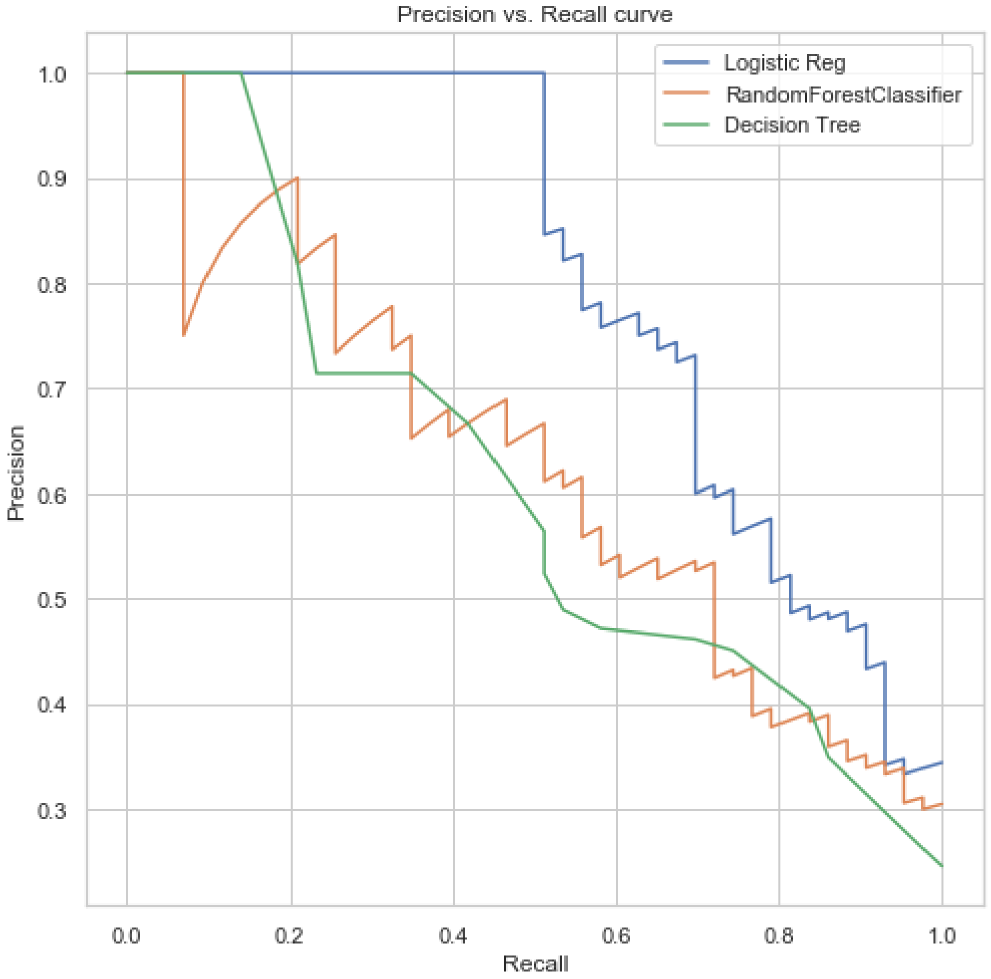
```
              precision    recall  f1-score   support

           0       0.87      0.95      0.91       132
           1       0.78      0.58      0.67        43

    accuracy                           0.86       175
   macro avg       0.83      0.76      0.79       175
weighted avg       0.85      0.86      0.85       175
```

```
In [92]: precision_lg, recall_lg, thresholds_lg = precision_recall_curve(ytest,
         p_clf_log_ba[:, 1])
```

```
In [93]: fpr_lg, tpr_lg, thresholds_lg = roc_curve(ytest, p_clf_log_ba[:, 1])
```

```
In [94]: fig, ax = plt.subplots(figsize=(8,8))
         plt.plot(recall_lg, precision_lg)
         plt.plot(recall_rf, precision_rf)
         plt.plot(recall_dc, precision_dc)
         plt.legend(('Logistic Reg', 'RandomForestClassifier', 'Decision Tree'))
         plt.xlabel('Recall')
         plt.ylabel('Precision')
         plt.title('Precision vs. Recall curve')
```

Out[94]:
```
Text(0.5, 1.0, 'Precision vs. Recall curve')
```



In [95]:
```
area_log_reg = auc(recall_lg, precision_lg)
print(area_log_reg)
area_rf = auc(recall_rf, precision_rf)
```

```
print(area_rf)
area_dc = auc(recall_dc, precision_dc)
print(area_dc)
```

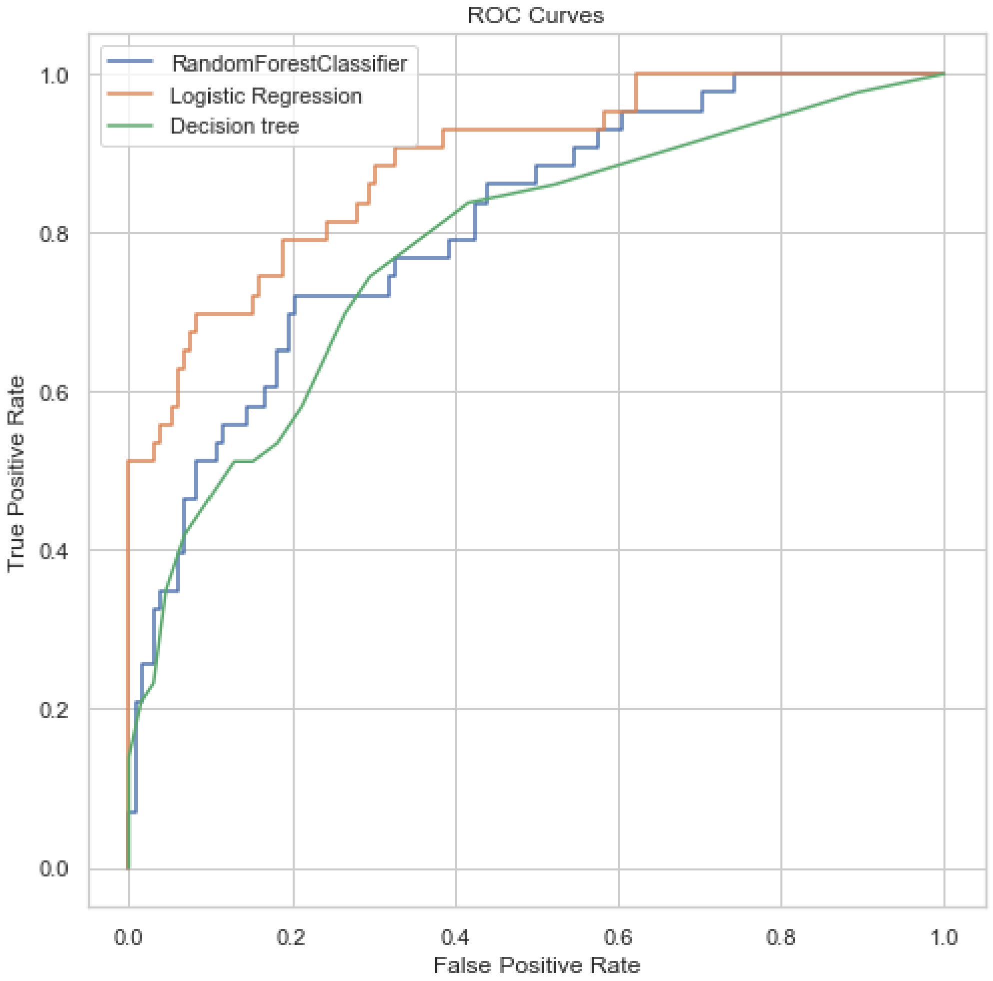
```
0.8011770322280233
0.6231740234944497
0.6083448382291768
```

In [96]:

```
fig, ax = plt.subplots(figsize=(8,8))
plt.plot(fpr_rf, tpr_rf)
plt.plot(fpr_lg, tpr_lg)
plt.plot(fpr_dc, tpr_dc)
plt.legend(('RandomForestClassifier','Logistic Regression' , 'Decision tree'))
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves')
```

Out[96]:

```
Text(0.5, 1.0, 'ROC Curves')
```



In [97]:

```
Areas_ROC_decision = roc_auc_score(ytest, dt_pred1)
Areas_ROC_logistic  = roc_auc_score(ytest, p_clf_log_ba[:, 1])
Areas_ROC_randomforest = roc_auc_score(ytest, y_best_rf_probas)
print(Areas_ROC_decision)
print(Areas_ROC_logistic)
print(Areas_ROC_randomforest)
```

```
0.7791578576462297
```

```
0.8879492600422833
0.8095489781536294
```

In [ ]: