In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split

# Importing dataset
df=pd.read_csv('E:/Machine Learning/Loan_RandomForest/train_loan.csv')
df.head()
```

Out[2]:

)ependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_A
0	Graduate	No	5849	0.0	NaN	
1	Graduate	No	4583	1508.0	128.0	
0	Graduate	Yes	3000	0.0	66.0	
0	Not Graduate	No	2583	2358.0	120.0	
0	Graduate	No	6000	0.0	141.0	

In [3]:

```
# Data Preprocessing and null values imputation
# Label Encoding
df['Gender']=df['Gender'].map({'Male':1,'Female':0})
df['Married']=df['Married'].map({'Yes':1,'No':0})
df['Education']=df['Education'].map({'Graduate':1,'Not Graduate':0})
df['Dependents'].replace('3+',3,inplace=True)
df['Self_Employed']=df['Self_Employed'].map({'Yes':1,'No':0})
df['Property_Area']=df['Property_Area'].map({'Semiurban':1,'Urban':2,'Rural':3})
df['Loan_Status']=df['Loan_Status'].map({'Y':1,'N':0})

#Null Value Imputation
rev_null=['Gender','Married','Dependents','Self_Employed','Credit_History','LoanAmount','Lo
df[rev_null]=df[rev_null].replace({np.nan:df['Gender'].mode(),
                                   np.nan:df['Married'].mode(),
                                   np.nan:df['Dependents'].mode(),
                                   np.nan:df['Self_Employed'].mode(),
                                   np.nan:df['Credit_History'].mode(),
                                   np.nan:df['LoanAmount'].mean(),
                                   np.nan:df['Loan_Amount_Term'].mean()})
```

In [4]:

```
X=df.drop(columns=['Loan_ID','Loan_Status']).values
Y=df['Loan_Status'].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 4
```

In [5]:

```
a=list(df.columns)
a.pop()
a.remove('Loan_ID')
a
```

Out[5]:

```
['Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'ApplicantIncome',
 'CoapplicantIncome',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History',
 'Property_Area']
```

In [6]:

```
feature_name=a
```

In [7]:

```
class_name=list(df['Loan_Status'].unique())
class_name
```

Out[7]:

```
[1, 0]
```

In [8]:

```
print('Shape of X_train=>',X_train.shape)
print('Shape of X_test=>',X_test.shape)
print('Shape of Y_train=>',Y_train.shape)
print('Shape of Y_test=>',Y_test.shape)# Building Decision Tree
```

```
Shape of X_train=> (491, 11)
Shape of X_test=> (123, 11)
Shape of Y_train=> (491,)
Shape of Y_test=> (123,)
```

In [17]:

```
# Building Decision Tree
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion = 'entropy', random_state = 42)
dt.fit(X_train, Y_train)
```

Out[17]:

```
DecisionTreeClassifier(criterion='entropy', random_state=42)
```

In [18]:

```
# Evaluation on Training set
dt_pred_train = dt.predict(X_train)
print('Training Set Evaluation F1-Score=>',f1_score(Y_train,dt_pred_train))
```

```
Training Set Evaluation F1-Score=> 1.0
```

In [19]:

```
# Evaluating on Test set
dt_pred_test = dt.predict(X_test)
print('Testing Set Evaluation F1-Score=>',f1_score(Y_test,dt_pred_test))
```

```
Testing Set Evaluation F1-Score=> 0.7953216374269005
```

In [ ]:

```

```

Random Forest

Random Forest is a tree-based machine learning algorithm that leverages the power of multiple decision trees for making decisions. As the name suggests, it is a “forest” of trees!

But why do we call it a “random” forest? That’s because it is a forest of randomly created decision trees. Each node in the decision tree works on a random subset of features to calculate the output. The random forest then combines the output of individual decision trees to generate the final output.

In [20]:

```
# Building  Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(criterion = 'entropy', random_state = 42)
rfc.fit(X_train, Y_train)

# Evaluating on Training set
rfc_pred_train = rfc.predict(X_train)
print('Training Set Evaluation F1-Score=>',f1_score(Y_train,rfc_pred_train))
```

```
Training Set Evaluation F1-Score=> 1.0
```

In [21]:

```
# Evaluating on Test set
rfc_pred_test = rfc.predict(X_test)
print('Testing Set Evaluation F1-Score=>',f1_score(Y_test,rfc_pred_test))
```

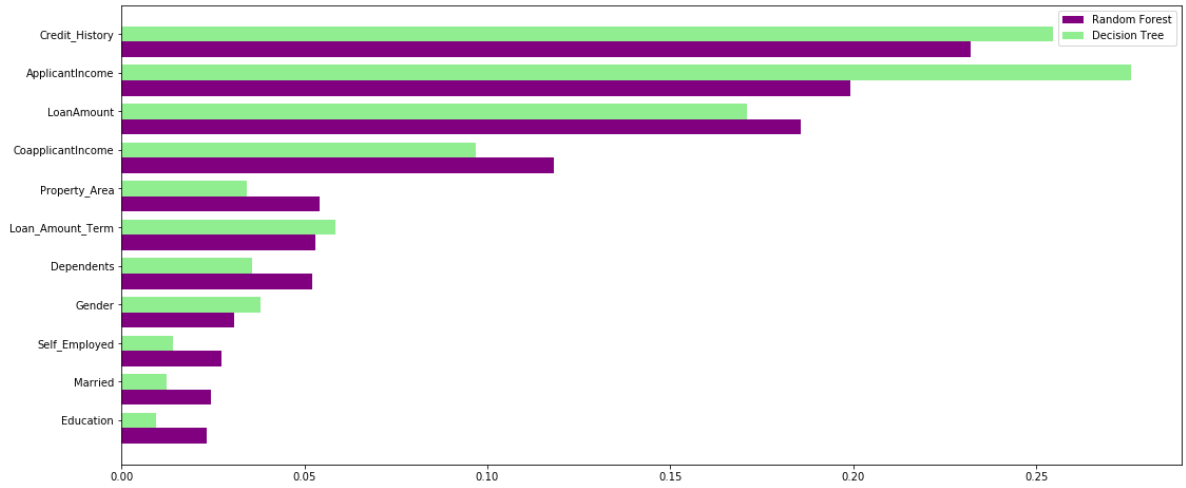
```
Testing Set Evaluation F1-Score=> 0.8461538461538461
```

In [27]:

```
feature_importance=pd.DataFrame({
    'rfc':rfc.feature_importances_,
    'dt':dt.feature_importances_
},index=df.drop(columns=['Loan_ID','Loan_Status']).columns)
feature_importance.sort_values(by='rfc',ascending=True,inplace=True)

index = np.arange(len(feature_importance))
fig, ax = plt.subplots(figsize=(18,8))
rfc_feature=ax.barh(index,feature_importance['rfc'],0.4,color='purple',label='Random Forest
dt_feature=ax.barh(index+0.4,feature_importance['dt'],0.4,color='lightgreen',label='Decisio
ax.set(yticks=index+0.4,yticklabels=feature_importance.index)

ax.legend()
plt.show()
```



In [12]:

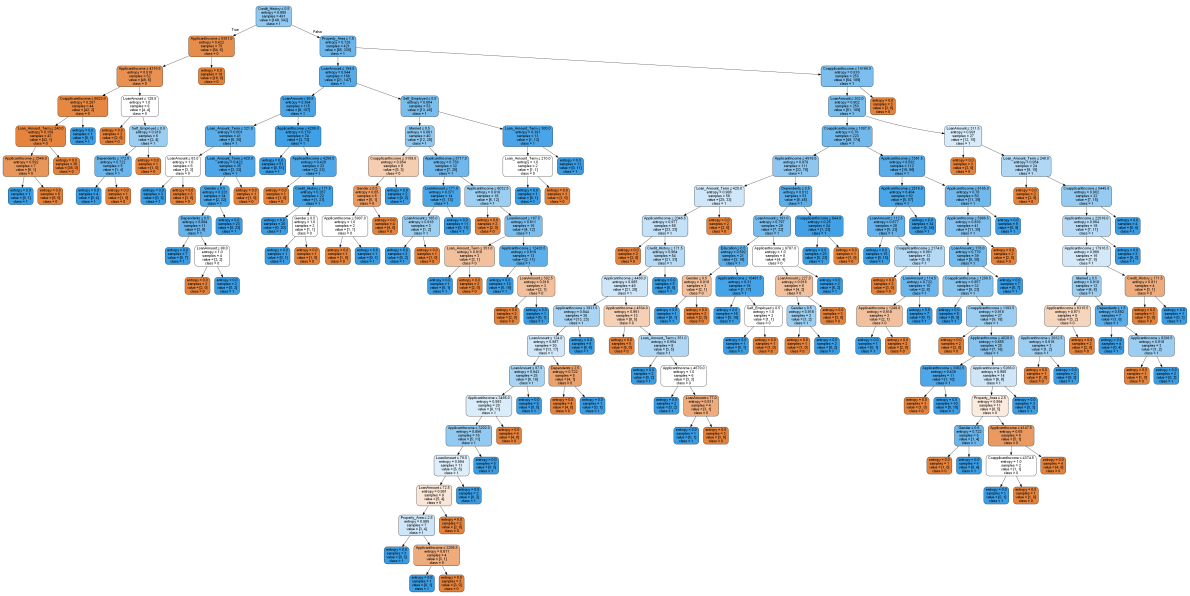
```
import six
import sys
sys.modules['sklearn.externals.six'] = six
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

from six import StringIO
```

In [51]:

```
dot_data = StringIO()
export_graphviz(dt, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True, feature_names = feature_name,class_names=['0','1']
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('loan.png')
Image(graph.create_png())
```

Out[51]:



In [ ]: