In [119]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier  #Import Decision Tree  Classifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LinearRegression
```

In [120]:

```
df = pd.read_csv("D:/PGDAI - lec/Machine learning/Dataset/train_loan.csv")
df.head(15)
```

Out[120]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coap
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	
5	LP001011	Male	Yes	2	Graduate	Yes	5417	
6	LP001013	Male	Yes	0	Not Graduate	No	2333	
7	LP001014	Male	Yes	3+	Graduate	No	3036	
8	LP001018	Male	Yes	2	Graduate	No	4006	
9	LP001020	Male	Yes	1	Graduate	No	12841	
10	LP001024	Male	Yes	2	Graduate	No	3200	
11	LP001027	Male	Yes	2	Graduate	NaN	2500	
12	LP001028	Male	Yes	2	Graduate	No	3073	
13	LP001029	Male	No	0	Graduate	No	1853	
14	LP001030	Male	Yes	2	Graduate	No	1299	

In [121]:

```
df.drop('Loan_ID',inplace=True, axis=1)
df
```

Out[121]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInc
0	Male	No	0	Graduate	No	5849	
1	Male	Yes	1	Graduate	No	4583	15
2	Male	Yes	0	Graduate	Yes	3000	
3	Male	Yes	0	Not Graduate	No	2583	23
4	Male	No	0	Graduate	No	6000	
...	...	...	...	...	...	...	
609	Female	No	0	Graduate	No	2900	
610	Male	Yes	3+	Graduate	No	4106	
611	Male	Yes	1	Graduate	No	8072	2
612	Male	Yes	2	Graduate	No	7583	
613	Female	No	0	Graduate	Yes	4583	

614 rows × 12 columns

In [122]:

```
df.shape
```

Out[122]:

```
(614, 12)
```

In [123]:

```
df.describe()
```

Out[123]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In [124]:

```
df.isnull().sum()
```

Out[124]:

```
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [126]:

```
df['Gender'].fillna(df['Gender'].mode()[0],inplace=True)
df['Married'].fillna(df['Married'].mode()[0],inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0],inplace=True)
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0],inplace=True)
df['LoanAmount'].fillna(df['LoanAmount'].mean(),inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0],inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0],inplace=True)

df.head(20)
```

12841	10968.0	349.000000	360.0	1.0	Semiurban	N
3200	700.0	70.000000	360.0	1.0	Urban	Y
2500	1840.0	109.000000	360.0	1.0	Urban	Y
3073	8106.0	200.000000	360.0	1.0	Urban	Y
1853	2840.0	114.000000	360.0	1.0	Rural	N
1299	1086.0	17.000000	120.0	1.0	Urban	Y
4950	0.0	125.000000	360.0	1.0	Urban	Y
3596	0.0	100.000000	240.0	1.0	Urban	Y
3510	0.0	76.000000	360.0	0.0	Urban	N
4887	0.0	133.000000	360.0	1.0	Rural	N
2600	3500.0	115.000000	360.0	1.0	Urban	Y

In [127]:

```
df.isnull().sum()
```

Out[127]:

```
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

In [134]:

```
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})
```

In [135]:

```
df
```

cantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
5849	0.0	146.412162	360.0	1.0	2	1
4583	1508.0	128.000000	360.0	1.0	3	0
3000	0.0	66.000000	360.0	1.0	2	1
2583	2358.0	120.000000	360.0	1.0	2	1
6000	0.0	141.000000	360.0	1.0	2	1
...	...	...	...	...	...	...
2900	0.0	71.000000	360.0	1.0	3	1
4106	0.0	40.000000	180.0	1.0	3	1
8072	240.0	253.000000	360.0	1.0	2	1
7583	0.0	187.000000	360.0	1.0	2	1
4583	0.0	133.000000	360.0	0.0	1	0

Decision Tree

In [136]:

```
print(df.groupby('Loan_Status').size())
```

```
Loan_Status
0    192
1    422
dtype: int64
```

In [137]:

```
columns != 'Loan_Status'], df['Loan_Status'], stratify = df['Loan_Status'],random_state=42)
```

In [138]:

```
print(y_train.value_counts())
print(y_test.value_counts())
```

```
1    316
0    144
Name: Loan_Status, dtype: int64
1    106
0     48
Name: Loan_Status, dtype: int64
```

In [139]:

```
feature_name = list(X_train.columns)
class_name = list(y_train.unique())
feature_name
```

Out[139]:

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

In [140]:

```
class_name
```

Out[140]:

```
[0, 1]
```

In [141]:

```
clf = DecisionTreeClassifier()

clf = clf.fit(X_train,y_train)
```

In [143]:

```
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7532467532467533
```

In [145]:

```
y_pred = clf.predict(X_train)
print("Accuracy:",metrics.accuracy_score(y_train, y_pred))
```

```
Accuracy: 1.0
```

Random Forest

In [146]:

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=20)
model = model.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = model.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7987012987012987
```

In [147]:

```
y_pred = model.predict(X_train)
print("Accuracy:",metrics.accuracy_score(y_train, y_pred))
```

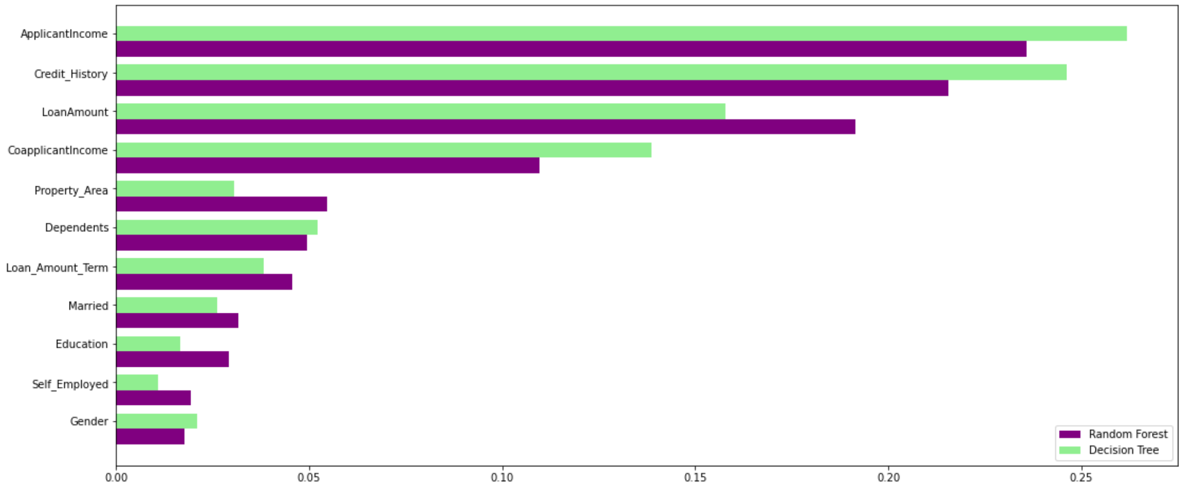
```
Accuracy: 0.9956521739130435
```

In [152]:

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

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

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



In [156]:

```
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 [ ]: