CREDIT_SCORE CLUSTERING

The dataset was obtained from Kaggle platform where the main aim was to cluster customer credit scoring based on,Credit Utilization Ratio,Payment History,Number of Credit Accounts,Loan Amount, Interest Rate,Loan Term and Type of Loan.K_means clustering and Decision trees was used in analysis.

In [1]:
```python
#importing relevant packages
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
#loading dataset
data=pd.read_csv('Desktop/credit_scoring.csv')
```

In [3]:
```python
#head of data
data.head()
```

Out[3]:

	Age	Gender	Marital Status	Education Level	Employment Status	Credit Utilization Ratio	Payment History	Number of Credit Accounts	Loan Amount	Interest Rate	Loan Term	Type of Loan
0	60	Male	Married	Master	Employed	0.22	2685.0	2	4675000	2.65	48	Personal Loan
1	25	Male	Married	High School	Unemployed	0.20	2371.0	9	3619000	5.19	60	Auto Loan
2	30	Female	Single	Master	Employed	0.22	2771.0	6	957000	2.76	12	Auto Loan
3	58	Female	Married	PhD	Unemployed	0.12	1371.0	2	4731000	6.57	60	Auto Loan
4	32	Male	Married	Bachelor	Self-Employed	0.99	828.0	2	3289000	6.28	36	Personal Loan

In [4]:
```python
#shape of data
data.shape
```

Out[4]:
```
(1000, 12)
```

In [5]:
```python
#info on data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Age                        1000 non-null   int64
 1   Gender                     1000 non-null   object
 2   Marital Status             1000 non-null   object
 3   Education Level            1000 non-null   object
 4   Employment Status          1000 non-null   object
 5   Credit Utilization Ratio   1000 non-null   float64
 6   Payment History            1000 non-null   float64
 7   Number of Credit Accounts  1000 non-null   int64
 8   Loan Amount                1000 non-null   int64
 9   Interest Rate              1000 non-null   float64
 10  Loan Term                  1000 non-null   int64
 11  Type of Loan               1000 non-null   object
dtypes: float64(3), int64(4), object(5)
memory usage: 93.9+ KB
```

In [6]:
```python
#columns in data
data.columns
```

Out[6]:
```
Index(['Age', 'Gender', 'Marital Status', 'Education Level',
       'Employment Status', 'Credit Utilization Ratio', 'Payment History',
       'Number of Credit Accounts', 'Loan Amount', 'Interest Rate',
       'Loan Term', 'Type of Loan'],
      dtype='object')
```

In [7]:
```python
#desriptive statistics
data.describe()
```

Out[7]:

	Age	Credit Utilization Ratio	Payment History	Number of Credit Accounts	Loan Amount	Interest Rate	Loan Term
count	1000.000000	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	1000.000000
mean	42.702000	0.509950	1452.814000	5.580000	2.471401e+06	10.686600	37.128000
std	13.266771	0.291057	827.934146	2.933634	1.387047e+06	5.479058	17.436274
min	20.000000	0.000000	0.000000	1.000000	1.080000e+05	1.010000	12.000000
25%	31.000000	0.250000	763.750000	3.000000	1.298000e+06	6.022500	24.000000
50%	42.000000	0.530000	1428.000000	6.000000	2.437500e+06	10.705000	36.000000
75%	54.000000	0.750000	2142.000000	8.000000	3.653250e+06	15.440000	48.000000
max	65.000000	1.000000	2857.000000	10.000000	4.996000e+06	19.990000	60.000000

In [8]:
```python
#checking if there are null values
data.isnull().sum()
```

Out[8]:
```
Age                          0
Gender                       0
Marital Status               0
Education Level              0
Employment Status            0
Credit Utilization Ratio     0
Payment History              0
Number of Credit Accounts    0
Loan Amount                  0
Interest Rate                0
Loan Term                    0
Type of Loan                 0
dtype: int64
```

In [9]:
```python
## label encoding string_columns
```

In [10]:
```python
from sklearn.preprocessing import LabelEncoder
```

In [11]:
```python
encoder=LabelEncoder()
```

In [12]:
```python
string_columns = data.select_dtypes(include=['object']).columns.tolist()

# Apply LabelEncoder to all string columns
label_encoder = LabelEncoder()
for col in string_columns:
    data[col] = label_encoder.fit_transform(data[col])
```

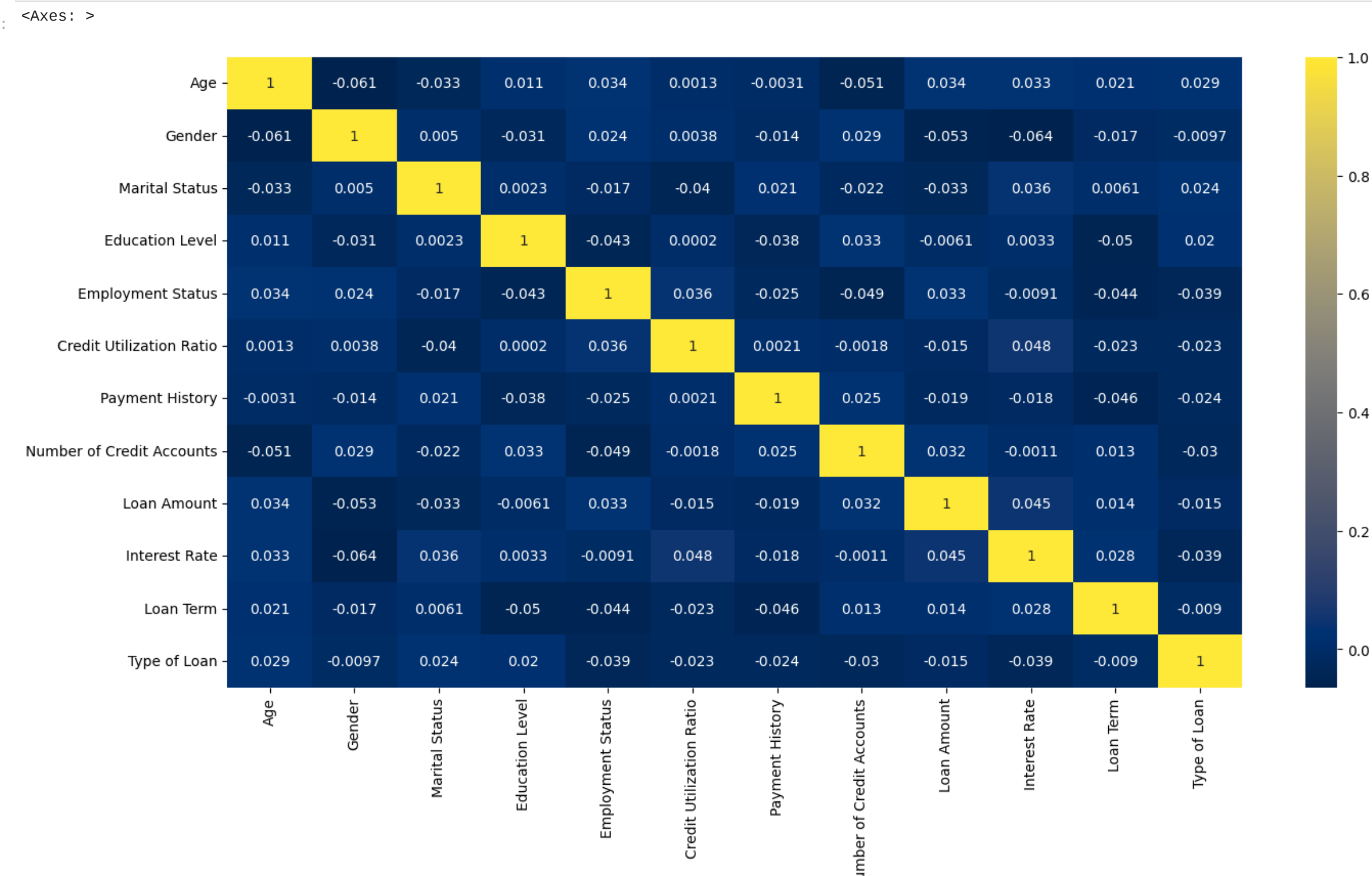
In [13]:
```python
data.corr()
```

Out[13]:

	Age	Gender	Marital Status	Education Level	Employment Status	Credit Utilization Ratio	Payment History	Number of Credit Accounts	Loan Amount	Interest Rate	Loan Term	Type of Loan
Age	1.000000	-0.060739	-0.033169	0.010536	0.034260	0.001308	-0.003114	-0.050672	0.034114	0.032867	0.021395	0.029437
Gender	-0.060739	1.000000	0.004966	-0.030724	0.023520	0.003816	-0.013561	0.028934	-0.052705	-0.064458	-0.016655	-0.009705
Marital Status	-0.033169	0.004966	1.000000	0.002261	-0.016894	-0.040207	0.020510	-0.021929	-0.033427	0.035683	0.006139	0.024147
Education Level	0.010536	-0.030724	0.002261	1.000000	-0.043000	0.000203	-0.037921	0.032890	-0.006067	0.003318	-0.049655	0.020078
Employment Status	0.034260	0.023520	-0.016894	-0.043000	1.000000	0.036413	-0.024883	-0.049489	0.032971	-0.009093	-0.044069	-0.039375
Credit Utilization Ratio	0.001308	0.003816	-0.040207	0.000203	0.036413	1.000000	0.002114	-0.001842	-0.014918	0.048237	-0.022924	-0.023498
Payment History	-0.003114	-0.013561	0.020510	-0.037921	-0.024883	0.002114	1.000000	0.025022	-0.019084	-0.017618	-0.045532	-0.024461
Number of Credit Accounts	-0.050672	0.028934	-0.021929	0.032890	-0.049489	-0.001842	0.025022	1.000000	0.032232	-0.001083	0.012794	-0.030000
Loan Amount	0.034114	-0.052705	-0.033427	-0.006067	0.032971	-0.014918	-0.019084	0.032232	1.000000	0.045288	0.014283	-0.015463
Interest Rate	0.032867	-0.064458	0.035683	0.003318	-0.009093	0.048237	-0.017618	-0.001083	0.045288	1.000000	0.028190	-0.038669
Loan Term	0.021395	-0.016655	0.006139	-0.049655	-0.044069	-0.022924	-0.045532	0.012794	0.014283	0.028190	1.000000	-0.009028
Type of Loan	0.029437	-0.009705	0.024147	0.020078	-0.039375	-0.023498	-0.024461	-0.030000	-0.015463	-0.038669	-0.009028	1.000000

In [14]:
```python
plt.figure(figsize=(16,8))
sns.heatmap(data.corr(),annot=True,cmap='cividis')
```

Out[14]:
```
<Axes: >
```



In [15]:
```python
data.head()
```

Out[15]:

	Age	Gender	Marital Status	Education Level	Employment Status	Credit Utilization Ratio	Payment History	Number of Credit Accounts	Loan Amount	Interest Rate	Loan Term	Type of Loan
0	60	1	1	2	0	0.22	2685.0	2	4675000	2.65	48	2
1	25	1	1	1	2	0.20	2371.0	9	3619000	5.19	60	0
2	30	0	2	2	0	0.22	2771.0	6	957000	2.76	12	0
3	58	0	1	3	2	0.12	1371.0	2	4731000	6.57	60	0
4	32	1	1	0	1	0.99	828.0	2	3289000	6.28	36	2

In [16]:
```python
#declaring x variable
x=data.iloc[:,5:12]
```

In [17]:
```python
#standardization
```

In [18]:
```python
from sklearn.preprocessing import StandardScaler
```

In [19]:
```python
scaler=StandardScaler()
```

In [20]:
```python
scaler.fit(x)
```

Out[20]:
```
▾ StandardScaler
StandardScaler()
```

In [21]:
```python
x_scaled=scaler.transform(x)
```

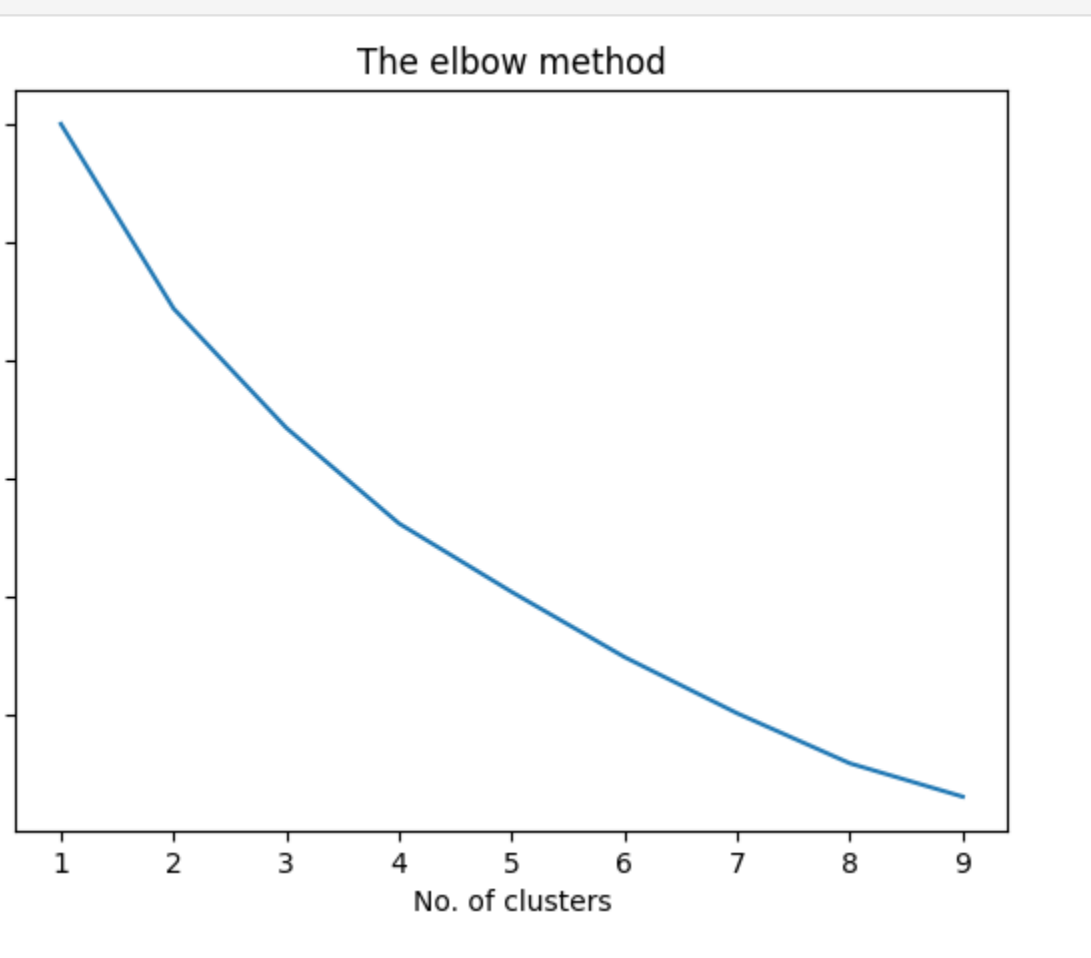
K_MEANS CLUSTERING

In [22]:
```python
from sklearn.cluster import KMeans
```

In [23]:
```python
wcss=[]
```

In [24]:
```python
for i in range(1, 10):
    kmeans=KMeans(i)
    kmeans.fit(x_scaled)
    wcss.append(kmeans.inertia_)
```

In [25]:
```python
#Elbow method plot
plt.plot(range(1,10),wcss)
plt.title('The elbow method')
plt.xlabel('No. of clusters')
plt.ylabel('wcss values')
plt.show()
```



In [26]:
```python
#predicting using n_clusters
kmeans=KMeans(n_clusters=3)
kmeans.fit(x_scaled)
clusters=x.copy()
clusters['cluster_pred']=kmeans.fit_predict(x_scaled)
```

In [27]:
```python
clusters.head()
```

Out[27]:

	Credit Utilization Ratio	Payment History	Number of Credit Accounts	Loan Amount	Interest Rate	Loan Term	Type of Loan	cluster_pred
0	0.22	2685.0	2	4675000	2.65	48	2	0
1	0.20	2371.0	9	3619000	5.19	60	0	1
2	0.22	2771.0	6	957000	2.76	12	0	1
3	0.12	1371.0	2	4731000	6.57	60	0	1
4	0.99	828.0	2	3289000	6.28	36	2	0

In [28]:
```python
#predicted clusters for each variable
clusters['cluster_pred']
```

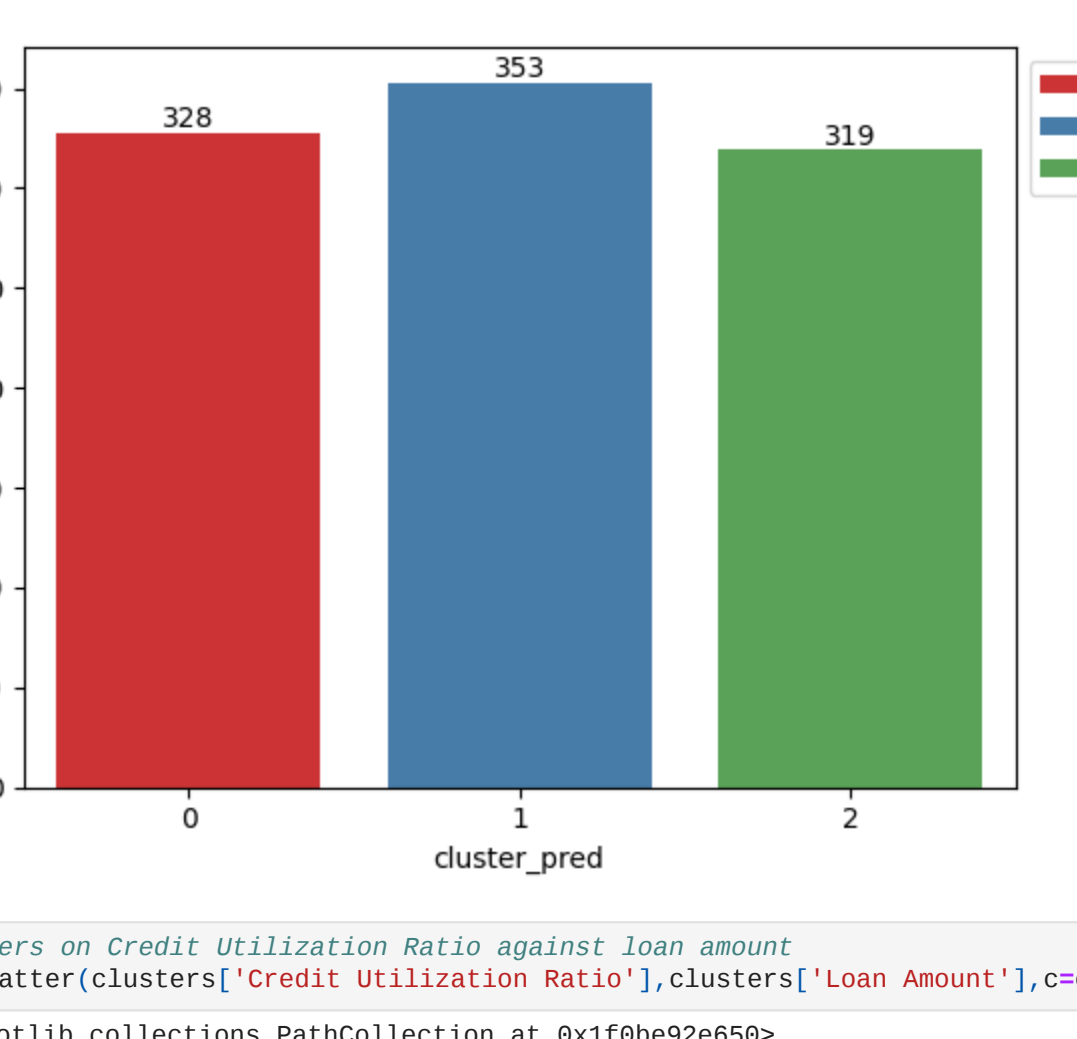
Out[28]:
```
0      0
1      1
2      1
3      1
4      0
      ..
995    1
996    2
997    2
998    0
999    1
Name: cluster_pred, Length: 1000, dtype: int32
```

In [29]:
```python
#Countplot on Status of various countries
ax=sns.countplot(x=clusters['cluster_pred'],data=clusters,hue=clusters['cluster_pred'],palette='Set1')
for i in ax.containers:
    ax.bar_label(i,)
plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
```

Out[29]:
```
<matplotlib.legend.Legend at 0x1f0c9f24f90>
```

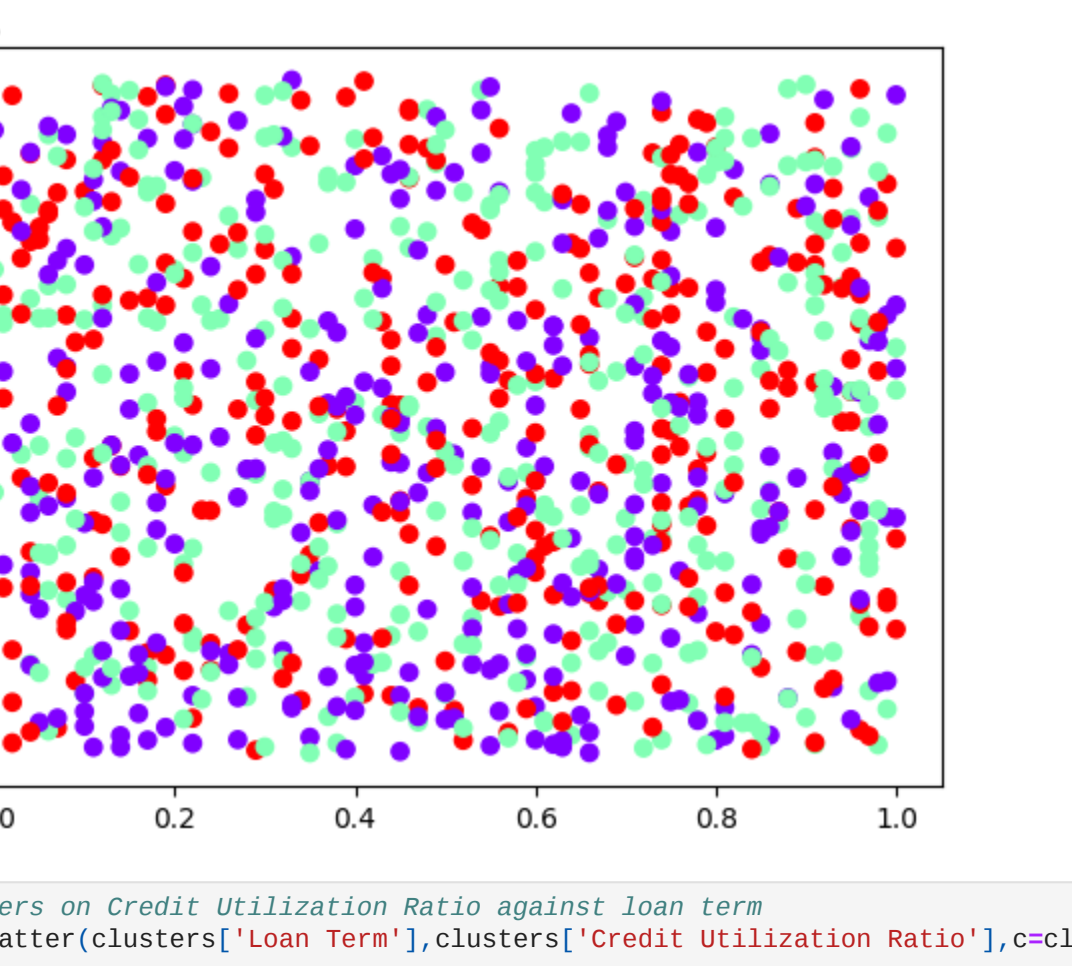
In [30]:
```python
#clusters on Credit Utilization Ratio against loan amount
plt.scatter(clusters['Credit Utilization Ratio'],clusters['Loan Amount'],c=clusters['cluster_pred'],cmap='rainbow')
```

Out[30]:
```
<matplotlib.collections.PathCollection at 0x1f0be92e650>
```



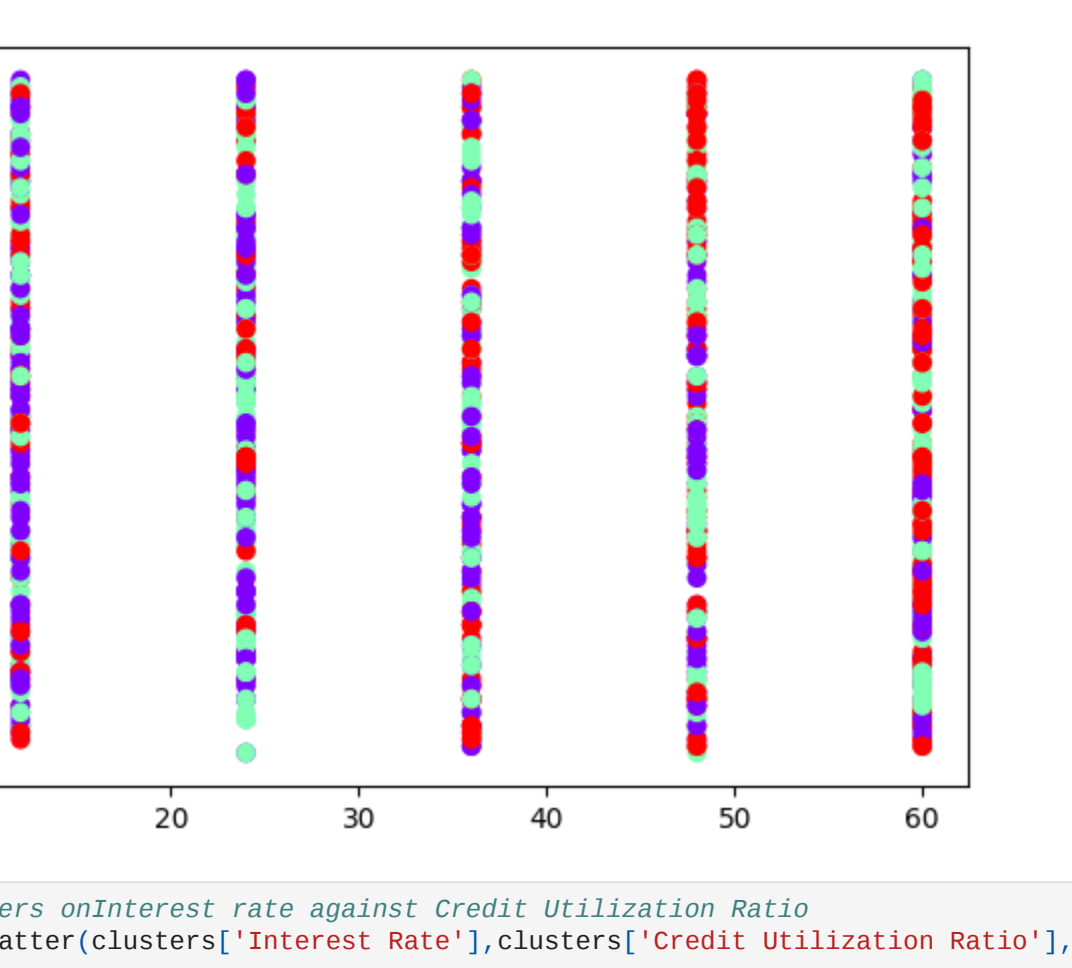
In [31]:
```python
#clusters on Credit Utilization Ratio against loan term
plt.scatter(clusters['Loan Term'],clusters['Credit Utilization Ratio'],c=clusters['cluster_pred'],cmap='rainbow')
```

Out[31]:
```
<matplotlib.collections.PathCollection at 0x1f0c189e450>
```



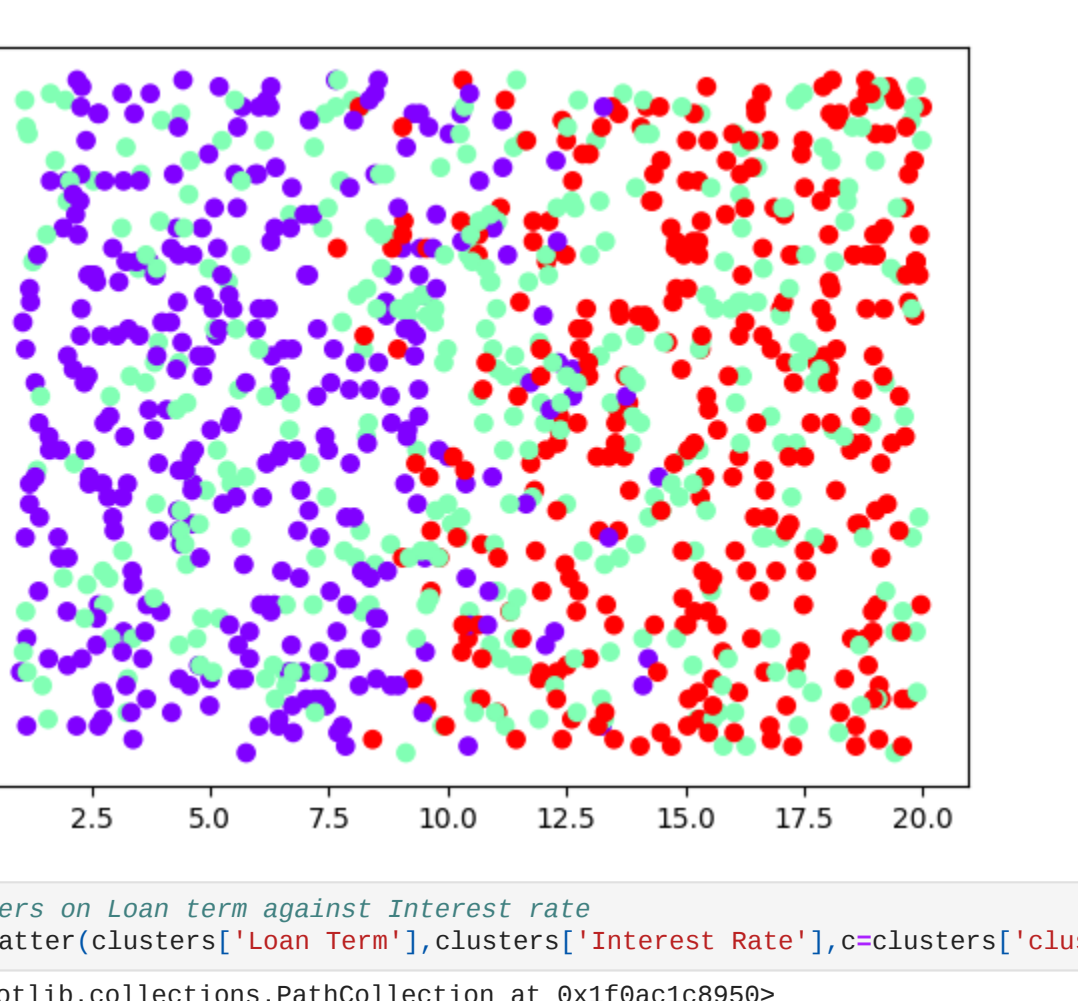
In [32]:
```python
#clusters onInterest rate against Credit Utilization Ratio
plt.scatter(clusters['Interest Rate'],clusters['Credit Utilization Ratio'],c=clusters['cluster_pred'],cmap='rainbow')
```

Out[32]:
```
<matplotlib.collections.PathCollection at 0x1f0ac1d4990>
```



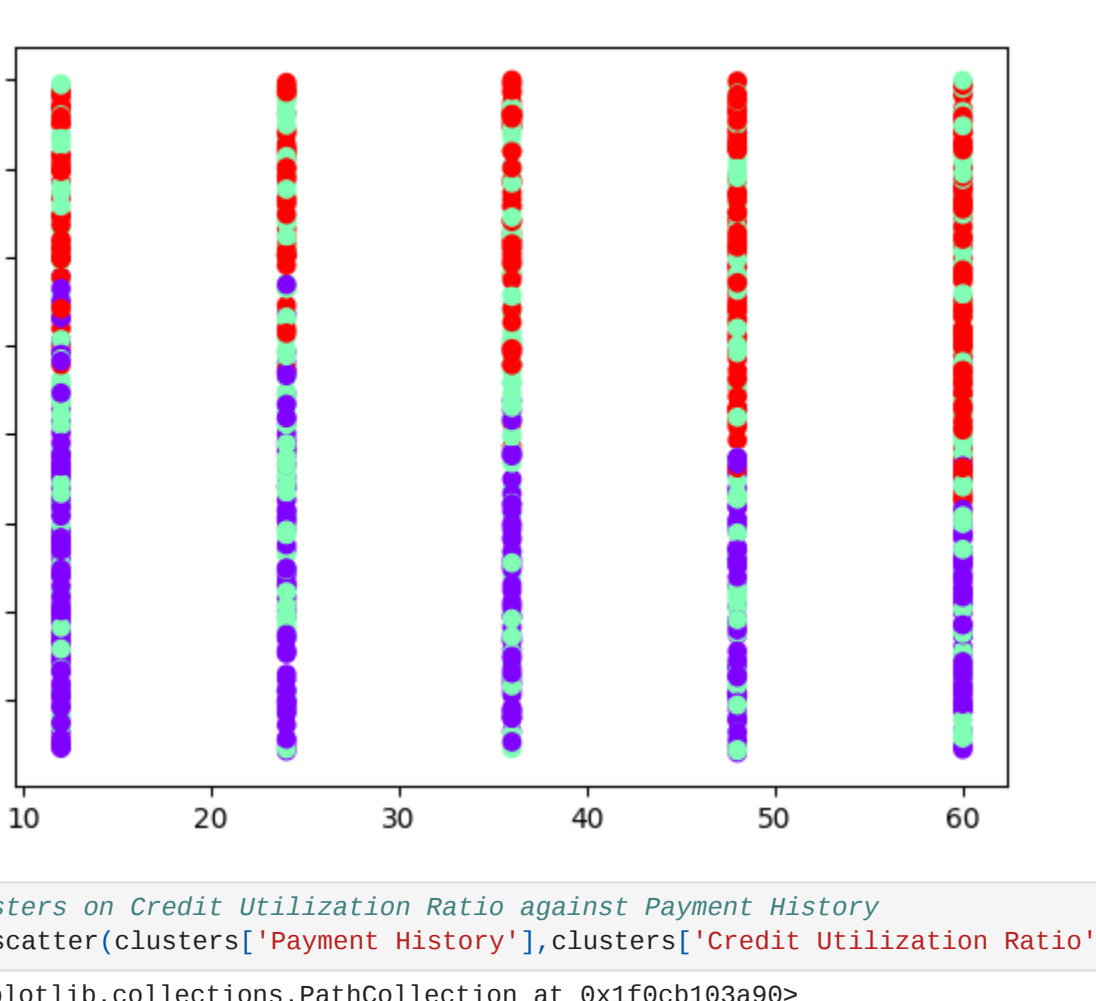
In [33]:
```python
#clusters on Loan term against Interest rate
plt.scatter(clusters['Loan Term'],clusters['Interest Rate'],c=clusters['cluster_pred'],cmap='rainbow')
```

Out[33]:
```
<matplotlib.collections.PathCollection at 0x1f0ac1c8950>
```



In [34]:
```python
#clusters on Credit Utilization Ratio against Payment History
plt.scatter(clusters['Payment History'],clusters['Credit Utilization Ratio'],c=clusters['cluster_pred'],cmap='rainbow')
```

Out[34]:
```
<matplotlib.collections.PathCollection at 0x1f0cb103a90>
```



TRAINING MODELS

DECISION TREE CLASIFFIER

In [35]:
```python
from sklearn.tree import DecisionTreeClassifier
```

In [36]:
```python
dtc= DecisionTreeClassifier()
```

In [37]:
```python
from sklearn.model_selection import train_test_split
```

In [38]:
```python
X=clusters.drop('cluster_pred',axis=1)
y=clusters['cluster_pred']
```

In [39]:
```python
#train_test split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=42)
```

In [40]:
```python
#fitting the model
dtc.fit(X_train,y_train)
```

Out[40]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [41]:
```python
#predicting x_test
y_pred_test = dtc.predict(X_test)
```

In [42]:
```python
from sklearn.metrics import accuracy_score
```

In [43]:
```python
accuracy_score(y_test,y_pred_test)
```

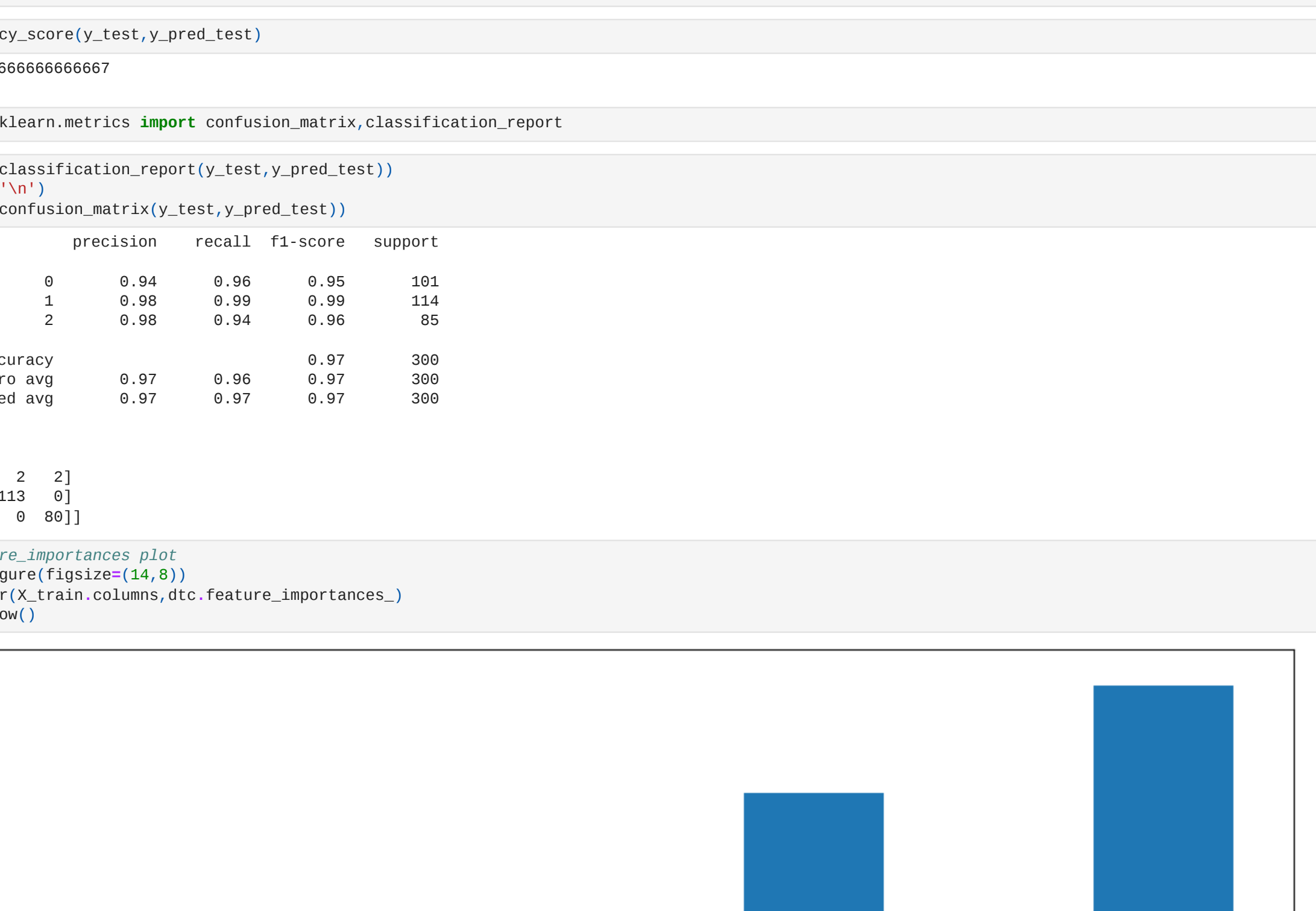
Out[43]:
```
0.9666666666666667
```

In [44]:
```python
from sklearn.metrics import confusion_matrix,classification_report
```

In [45]:
```python
print(classification_report(y_test,y_pred_test))
print('\n')
print(confusion_matrix(y_test,y_pred_test))
```

```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       101
           1       0.98      0.99      0.99       114
           2       0.98      0.94      0.96        85

    accuracy                           0.97       300
   macro avg       0.97      0.96      0.97       300
weighted avg       0.97      0.97      0.97       300



[[ 97   2   2]
 [  1 113   0]
 [  5   0  80]]
```

In [46]:
```python
#feature_importances plot
plt.figure(figsize=(14,8))
plt.bar(X_train.columns,dtc.feature_importances_)
plt.show()
```



In [47]:
```python
#DECISION TREE DIAGRAM
```

In [48]:
```python
from sklearn import tree
import graphviz
```

In [49]:
```python
data.feature_names = ['Credit Utilization Ratio', 'Payment History',
       'Number of Credit Accounts', 'Loan Amount', 'Interest Rate',
       'Loan Term', 'Type of Loan', 'cluster_pred']
fig, axis = plt.subplots(nrows=1, ncols=1, figsize=(30,30))
tree_data=tree.plot_tree(dtc,feature_names=data.feature_names, class_names = ['0','1','2'], filled=True);
graph = graphviz.Source(tree_data)
```