DMML LAB 02 – CREDIT CARD FRAUD DETECTION USING DECISION TREE CLASSIFIER

1. Import all required libraries.

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,plot_tree
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix,precision_score
```

2. Load the data set and understands its schema.

```
data=pd.read_csv('credit_card.csv')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

3. Divide the data into two subsets – fraudulent transactions and non-fraudulent transactions.

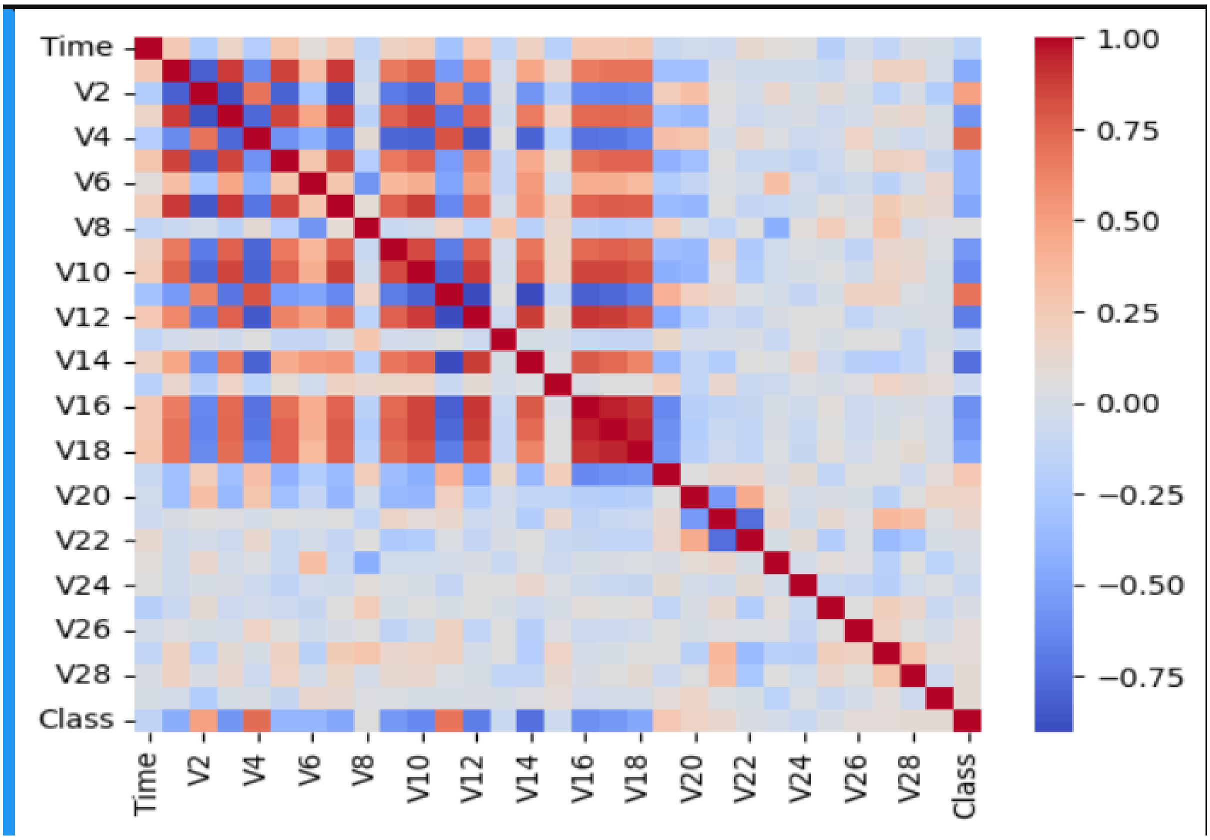
```
fraud=data[data['Class']==1]
n_fraud=data[data['Class']==0]
```

4. To make the data set balanced, perform down-sampling on non-fraudulent transactions. And create a balanced data.

```
down_sample=resample(n_fraud, replace=False, n_samples=len(fraud),random_state=42)
new_data=pd.concat([fraud,down_sample])
```

5. Create and display correlation matrix.

```python
matrix=new_data.corr()
sns.heatmap(matrix, annot=False, cmap="coolwarm")
plt.show()
print(matrix)
```




```
            Time        V1        V2        V3        V4        V5        V6  \
Time    1.000000  0.243263 -0.217137  0.155027 -0.208862  0.283542  0.075980
V1      0.243263  1.000000 -0.816943  0.882471 -0.619965  0.865225  0.329310
V2     -0.217137 -0.816943  1.000000 -0.867410  0.681796 -0.805321 -0.281813
V3      0.155027  0.882471 -0.867410  1.000000 -0.771848  0.857642  0.461689
V4     -0.208862 -0.619965  0.681796 -0.771848  1.000000 -0.581868 -0.439029
V5      0.283542  0.865225 -0.805321  0.857642 -0.581868  1.000000  0.293905
V6      0.075980  0.329310 -0.281813  0.461689 -0.439029  0.293905  1.000000
V7      0.217504  0.889875 -0.847130  0.891757 -0.710074  0.847129  0.275501
V8     -0.139904 -0.084553 -0.020836 -0.172597  0.101289 -0.206569 -0.572640
V9      0.190114  0.663959 -0.695144  0.757439 -0.787982  0.667176  0.365317
V10     0.215099  0.745482 -0.768706  0.854791 -0.794519  0.762153  0.417039
V11    -0.297619 -0.540378  0.621377 -0.719165  0.799422 -0.531985 -0.492832
V12     0.267066  0.603148 -0.669763  0.761258 -0.836754  0.624626  0.500273
V13    -0.141242 -0.036857  0.018820 -0.051889  0.041384 -0.111508 -0.115065
V14     0.186692  0.454024 -0.572220  0.654745 -0.795649  0.436951  0.520913
V15    -0.184182  0.140417 -0.190239  0.170439 -0.162090  0.086882 -0.043440
V16     0.250951  0.646476 -0.638585  0.728870 -0.731729  0.699029  0.427490
V17     0.244780  0.682528 -0.645158  0.737454 -0.713665  0.751720  0.417419
V18     0.272445  0.682592 -0.623291  0.703330 -0.645725  0.753525  0.353185
V19    -0.096996 -0.317552  0.215366 -0.319798  0.312209 -0.406675 -0.225027
V20    -0.043393 -0.316667  0.322719 -0.361543  0.281846 -0.325511 -0.115811
V21    -0.059941  0.011112  0.044077  0.028572 -0.020670  0.041081  0.025501
V22     0.116083 -0.049498 -0.014353 -0.062479  0.125225 -0.096883 -0.018369
V23     0.068847 -0.051885  0.132845 -0.028820  0.024547 -0.079087  0.317320
V24     0.055238 -0.054373 -0.001788  0.005521 -0.069553 -0.145552 -0.039262
V25    -0.199059 -0.078195  0.100363 -0.074373 -0.033853 -0.073384 -0.100485
V26    -0.018455  0.033892  0.001802 -0.025529  0.173673  0.048972 -0.054662
V27    -0.127087  0.183946 -0.162699  0.103928 -0.009021  0.194412 -0.167903
V28     0.013042  0.196646  0.008552  0.133026 -0.073229  0.172476 -0.032547
Amount -0.004057 -0.019517 -0.215022 -0.004810  0.008232 -0.106674  0.145229
Class  -0.149122 -0.446093  0.491904 -0.564371  0.712610 -0.382234 -0.389257
```

	V7	V8	V9	...	V21	V22	V23	\
Time	0.217504	-0.139904	0.190114	...	-0.059941	0.116083	0.068847	
V1	0.889875	-0.084553	0.663959	...	0.011112	-0.049498	-0.051885	
V2	-0.847130	-0.020836	-0.695144	...	0.044077	-0.014353	0.132845	
V3	0.891757	-0.172597	0.757439	...	0.028572	-0.062479	-0.028820	
V4	-0.710074	0.101289	-0.787982	...	-0.020670	0.125225	0.024547	
V5	0.847129	-0.206569	0.667176	...	0.041081	-0.096883	-0.079087	
V6	0.275501	-0.572640	0.365317	...	0.025501	-0.018369	0.317320	
V7	1.000000	0.087144	0.762076	...	0.037729	-0.119967	-0.095147	
V8	0.087144	1.000000	-0.079754	...	-0.125856	0.043338	-0.429852	
V9	0.762076	-0.079754	1.000000	...	0.156117	-0.246357	-0.042880	
V10	0.868716	-0.051761	0.842800	...	0.081223	-0.216097	-0.041899	
V11	-0.639992	0.170697	-0.692296	...	0.137390	0.019800	-0.029654	
V12	0.722655	-0.164852	0.761492	...	-0.074418	-0.114697	0.012807	
V13	-0.013158	0.272042	-0.033293	...	-0.014986	0.009377	-0.090450	
V14	0.544175	-0.185757	0.675375	...	-0.219396	0.061082	0.020356	
V15	0.193745	0.144394	0.147172	...	0.137953	-0.087289	-0.067742	
V16	0.749241	-0.172327	0.728924	...	-0.148757	-0.101342	0.003108	
V17	0.771584	-0.220965	0.756190	...	-0.092037	-0.125722	0.017807	
V18	0.762815	-0.181500	0.715396	...	-0.076985	-0.124196	0.022858	
V19	-0.354669	0.214137	-0.333228	...	0.114394	0.142355	0.010677	
V20	-0.378906	-0.030163	-0.365775	...	-0.541566	0.438776	0.092596	
V21	0.037729	-0.125856	0.156117	...	1.000000	-0.747431	0.125101	
V22	-0.119967	0.043338	-0.246357	...	-0.747431	1.000000	-0.000288	
V23	-0.095147	-0.429852	-0.042880	...	0.125101	-0.000288	1.000000	
V24	-0.053630	0.075672	0.021354	...	-0.052439	0.096116	-0.025635	
V25	0.060336	0.218897	0.000276	...	0.122580	-0.219157	0.073039	
V26	0.006195	0.053551	-0.149008	...	0.030305	0.021470	0.033188	
V27	0.238916	0.289669	0.139241	...	0.368388	-0.356527	-0.174672	
V28	0.151972	-0.015029	0.123195	...	0.317970	-0.270020	0.045112	
Amount	0.113331	0.028820	0.023240	...	0.024282	-0.002257	-0.160020	
Class	-0.477730	0.057207	-0.559589	...	0.124708	0.014889	-0.022999	

	V24	V25	V26	V27	V28	Amount	Class
Time	0.055238	-0.199059	-0.018455	-0.127087	0.013042	-0.004057	-0.149122
V1	-0.054373	-0.078195	0.033892	0.183946	0.196646	-0.019517	-0.446093
V2	-0.001788	0.100363	0.001802	-0.162699	0.008552	-0.215022	0.491904
V3	0.005521	-0.074373	-0.025529	0.103928	0.133026	-0.004810	-0.564371
V4	-0.069553	-0.033853	0.173673	-0.009021	-0.073229	0.008232	0.712610
V5	-0.145552	-0.073384	0.048972	0.194412	0.172476	-0.106674	-0.382234
V6	-0.039262	-0.100485	-0.054662	-0.167903	-0.032547	0.145229	-0.389257
V7	-0.053630	0.060336	0.006195	0.238916	0.151972	0.113331	-0.477730
V8	0.075672	0.218897	0.053551	0.289669	-0.015029	0.028820	0.057207
V9	0.021354	0.000276	-0.149008	0.139241	0.123195	0.023240	-0.559589
V10	-0.002696	0.038062	-0.052838	0.160527	0.127668	-0.003164	-0.628327
V11	-0.114634	0.003144	0.177269	0.175903	0.024516	-0.006541	0.685056
V12	0.039597	0.047919	-0.134242	-0.024698	0.002087	0.012669	-0.682039
V13	0.034578	0.000560	0.035840	0.039390	-0.124276	-0.004075	-0.078165
V14	0.137823	-0.070959	-0.199163	-0.208011	-0.128201	0.033345	-0.749228
V15	0.023227	-0.002986	0.027795	0.174894	0.115138	0.085339	-0.057636
V16	-0.049963	0.064583	-0.075298	-0.024822	0.012990	-0.033034	-0.597790
V17	-0.081907	0.042383	-0.059971	0.002905	0.054013	-0.029415	-0.559169
V18	-0.099776	0.070970	-0.037535	0.056283	0.096423	-0.023958	-0.464857
V19	0.106375	-0.140880	0.053805	0.055043	-0.058013	0.080091	0.267809
V20	-0.018396	0.009129	0.004026	-0.171003	0.035600	0.159680	0.169025
V21	-0.052439	0.122580	0.030305	0.368388	0.317970	0.024282	0.124708
V22	0.096116	-0.219157	0.021470	-0.356527	-0.270020	-0.002257	0.014889
V23	-0.025635	0.073039	0.033188	-0.174672	0.045112	-0.160020	-0.022999
V24	1.000000	-0.084664	-0.119194	-0.195946	-0.041418	0.023094	-0.084566
V25	-0.084664	1.000000	0.052024	0.209411	0.134466	-0.090211	0.012415
V26	-0.119194	0.052024	1.000000	0.191205	0.038860	-0.030079	0.084471
V27	-0.195946	0.209411	0.191205	1.000000	0.282197	0.074503	0.079840
V28	-0.041418	0.134466	0.038860	0.282197	1.000000	-0.063467	0.093970
Amount	0.023094	-0.090211	-0.030079	0.074503	-0.063467	1.000000	0.094434
Class	-0.084566	0.012415	0.084471	0.079840	0.093970	0.094434	1.000000

[31 rows x 31 columns]

5. Filter the relevant features based on threshold value.

```
related_features=matrix['Class'][abs(matrix['Class'])> 0.1].index.tolist()
related_features.remove('Class')
print(f"Selected features:{related_features}")
```

```
Selected features:['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V9', 'V10', 'V11', 'V12', 'V1
4', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21']
```

6. Create 2 sets for features and targets and train the model using decision tree algorithm.

```
x=new_data[related_features]
y=new_data['Class']
```

```
[138]:
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42,stratify=y)
```

```
[139]:
```

```
model=DecisionTreeClassifier(random_state=42,max_depth=5)
model.fit(x_train,y_train)
```

```
[139]:
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, random_state=42)
```

7. Predict value for test data and evaluate the model.

```
y_pred=model.predict(x_test)
```

```
[141]:
```

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label=1)
conf_matrix = confusion_matrix(y_test, y_pred)
```

8. Display the results.

```
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.92
Precision: 0.90

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.89      0.92        99
           1       0.90      0.96      0.93        98

    accuracy                           0.92       197
   macro avg       0.93      0.92      0.92       197
weighted avg       0.93      0.92      0.92       197
```

9. Create and display confusion matrix.

```
Confusion Matrix:
[[88 11]
 [ 4 94]]
```