Credit Card Fraud Detection - K-Nearest Neighbor(KNN)

Importing the Dependencies

In [1]:

```
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pylab as plt

%matplotlib inline
```

In [2]:

```
# loading the dataset to a Pandas DataFrame
credit_card_data = pd.read_csv('creditcard.csv')
```

In [3]:

```
# first 5 rows of the dataset
credit_card_data.head()
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.09869
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.08510
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.24767
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.37743
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.27053

5 rows × 31 columns

In [4]:

```
credit_card_data.describe().transpose()
```

Out[4]:

	count	mean	std	min	25%	50%	
Time	284807.0	9.481386e+04	47488.145955	0.000000	54201.500000	84692.000000	13
V1	284807.0	1.168375e-15	1.958696	-56.407510	-0.920373	0.018109	
V2	284807.0	3.416908e-16	1.651309	-72.715728	-0.598550	0.065486	
V3	284807.0	-1.379537e-15	1.516255	-48.325589	-0.890365	0.179846	
V4	284807.0	2.074095e-15	1.415869	-5.683171	-0.848640	-0.019847	
V5	284807.0	9.604066e-16	1.380247	-113.743307	-0.691597	-0.054336	
V6	284807.0	1.487313e-15	1.332271	-26.160506	-0.768296	-0.274187	
V7	284807.0	-5.556467e-16	1.237094	-43.557242	-0.554076	0.040103	
V8	284807.0	1.213481e-16	1.194353	-73.216718	-0.208630	0.022358	
V9	284807.0	-2.406331e-15	1.098632	-13.434066	-0.643098	-0.051429	
V10	284807.0	2.239053e-15	1.088850	-24.588262	-0.535426	-0.092917	
V11	284807.0	1.673327e-15	1.020713	-4.797473	-0.762494	-0.032757	
V12	284807.0	-1.247012e-15	0.999201	-18.683715	-0.405571	0.140033	
V13	284807.0	8.190001e-16	0.995274	-5.791881	-0.648539	-0.013568	
V14	284807.0	1.207294e-15	0.958596	-19.214325	-0.425574	0.050601	
V15	284807.0	4.887456e-15	0.915316	-4.498945	-0.582884	0.048072	
V16	284807.0	1.437716e-15	0.876253	-14.129855	-0.468037	0.066413	
V17	284807.0	-3.772171e-16	0.849337	-25.162799	-0.483748	-0.065676	
V18	284807.0	9.564149e-16	0.838176	-9.498746	-0.498850	-0.003636	
V19	284807.0	1.039917e-15	0.814041	-7.213527	-0.456299	0.003735	
V20	284807.0	6.406204e-16	0.770925	-54.497720	-0.211721	-0.062481	
V21	284807.0	1.654067e-16	0.734524	-34.830382	-0.228395	-0.029450	
V22	284807.0	-3.568593e-16	0.725702	-10.933144	-0.542350	0.006782	
V23	284807.0	2.578648e-16	0.624460	-44.807735	-0.161846	-0.011193	
V24	284807.0	4.473266e-15	0.605647	-2.836627	-0.354586	0.040976	
V25	284807.0	5.340915e-16	0.521278	-10.295397	-0.317145	0.016594	
V26	284807.0	1.683437e-15	0.482227	-2.604551	-0.326984	-0.052139	
V27	284807.0	-3.660091e-16	0.403632	-22.565679	-0.070840	0.001342	
V28	284807.0	-1.227390e-16	0.330083	-15.430084	-0.052960	0.011244	
Amount	284807.0	8.834962e+01	250.120109	0.000000	5.600000	22.000000	
Class	284807.0	1.727486e-03	0.041527	0.000000	0.000000	0.000000	

In [5]:

```
# dataset informations
credit_card_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
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

In [6]:

```
# checking the number of missing values in each column
credit_card_data.isnull().sum()
```

Out[6]:

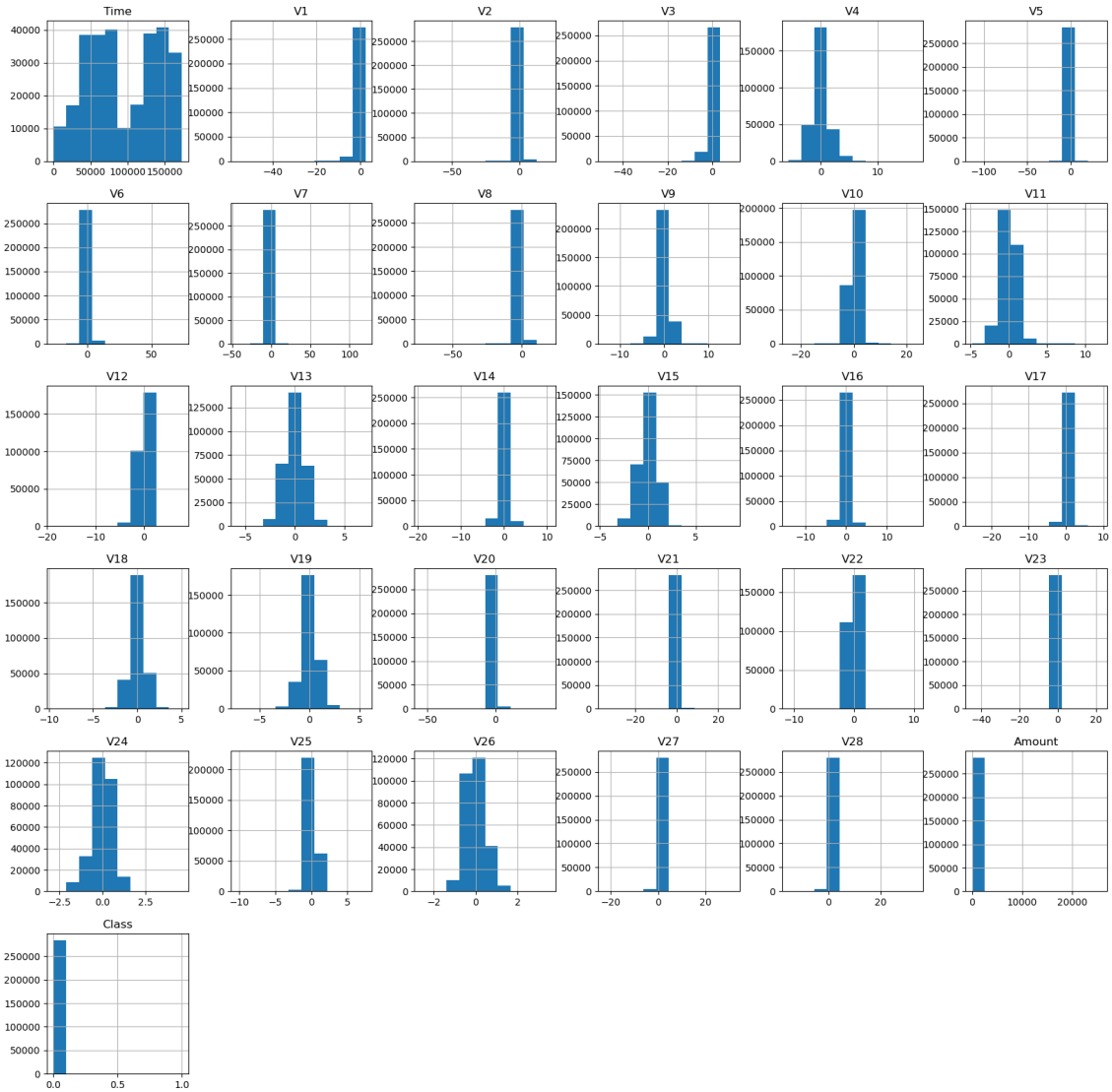
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

In [7]:

```
#
credit_card_data.hist(figsize=(20,20))
```

Out[7]:

```
array([[<Axes: title={'center': 'Time'}>, <Axes: title={'center': 'V1'}>,
        <Axes: title={'center': 'V2'}>, <Axes: title={'center': 'V3'}>,
        <Axes: title={'center': 'V4'}>, <Axes: title={'center': 'V5'}>],
       [<Axes: title={'center': 'V6'}>, <Axes: title={'center': 'V7'}>,
        <Axes: title={'center': 'V8'}>, <Axes: title={'center': 'V9'}>,
        <Axes: title={'center': 'V10'}>, <Axes: title={'center': 'V11'}
>],
       [<Axes: title={'center': 'V12'}>, <Axes: title={'center': 'V13'}>,
        <Axes: title={'center': 'V14'}>, <Axes: title={'center': 'V15'}>,
        <Axes: title={'center': 'V16'}>, <Axes: title={'center': 'V17'}
>],
       [<Axes: title={'center': 'V18'}>, <Axes: title={'center': 'V19'}>,
        <Axes: title={'center': 'V20'}>, <Axes: title={'center': 'V21'}>,
        <Axes: title={'center': 'V22'}>, <Axes: title={'center': 'V23'}
>],
       [<Axes: title={'center': 'V24'}>, <Axes: title={'center': 'V25'}>,
        <Axes: title={'center': 'V26'}>, <Axes: title={'center': 'V27'}>,
        <Axes: title={'center': 'V28'}>,
        <Axes: title={'center': 'Amount'}>],
       [<Axes: title={'center': 'Class'}>, <Axes: >, <Axes: >, <Axes: >,
        <Axes: >, <Axes: >]], dtype=object)
```


Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class

In [8]:

```
sns.pairplot(credit_card_data, hue='Time')
```

```
---------------------------------------------------------------------
--
KeyError                                  Traceback (most recent call las
t)
File c:\Users\USER\AppData\Local\Programs\Python\Python39\lib\site-packag
es\pandas\core\indexes\base.py:3803, in Index.get_loc(self, key, method,
tolerance)
   3802 try:
-> 3803     return self._engine.get_loc(casted_key)
   3804 except KeyError as err:

File c:\Users\USER\AppData\Local\Programs\Python\Python39\lib\site-packag
es\pandas\_libs\index.pyx:138, in pandas._libs.index.IndexEngine.get_loc
()

File c:\Users\USER\AppData\Local\Programs\Python\Python39\lib\site-packag
es\pandas\_libs\index.pyx:165, in pandas._libs.index.IndexEngine.get_loc
()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.hashta
ble.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.hashta
ble.PyObjectHashTable.get_item()

KeyError: 'Type'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call las
t)
Cell In[8], line 1
----> 1 sns.pairplot(credit_card_data, hue='Type')

File c:\Users\USER\AppData\Local\Programs\Python\Python39\lib\site-packag
es\seaborn\axisgrid.py:2114, in pairplot(data, hue, hue_order, palette, v
ars, x_vars, y_vars, kind, diag_kind, markers, height, aspect, corner, dr
opna, plot_kws, diag_kws, grid_kws, size)
   2112 # Set up the PairGrid
   2113 grid_kws.setdefault("diag_sharey", diag_kind == "hist")
-> 2114 grid = PairGrid(data, vars=vars, x_vars=x_vars, y_vars=y_vars, hu
e=hue,
   2115                 hue_order=hue_order, palette=palette, corner=corn
er,
   2116                 height=height, aspect=aspect, dropna=dropna, **gr
id_kws)
   2118 # Add the markers here as PairGrid has figured out how many level
s of the
   2119 # hue variable are needed and we don't want to duplicate that pro
cess
   2120 if markers is not None:

File c:\Users\USER\AppData\Local\Programs\Python\Python39\lib\site-packag
es\seaborn\axisgrid.py:1321, in PairGrid.__init__(self, data, hue, vars,
x_vars, y_vars, hue_order, palette, hue_kws, corner, diag_sharey, height,
aspect, layout_pad, despine, dropna)
   1310     self.hue_vals = pd.Series(["_nolegend_"] * len(data),
   1311                               index=data.index)
   1312 else:
   1313     # We need hue_order and hue_names because the former is used
to control
```

```
   1314     # the order of drawing and the latter is used to control the
order of
   (...)
   1319     # to the axes-level functions, while always handling legend c
reation.
   1320     # See GH2307
-> 1321     hue_names = hue_order = categorical_order(data[hue], hue_orde
r)
   1322     if dropna:
   1323         # Filter NA from the list of unique hue names
   1324         hue_names = list(filter(pd.notnull, hue_names))

File c:\Users\USER\AppData\Local\Programs\Python\Python39\lib\site-packag
es\pandas\core\frame.py:3804, in DataFrame.__getitem__(self, key)
   3802 if self.columns.nlevels > 1:
   3803     return self._getitem_multilevel(key)
-> 3804 indexer = self.columns.get_loc(key)
   3805 if is_integer(indexer):
   3806     indexer = [indexer]

File c:\Users\USER\AppData\Local\Programs\Python\Python39\lib\site-packag
es\pandas\core\indexes\base.py:3805, in Index.get_loc(self, key, method,
tolerance)
   3803     return self._engine.get_loc(casted_key)
   3804 except KeyError as err:
-> 3805     raise KeyError(key) from err
   3806 except TypeError:
   3807     # If we have a listlike key, _check_indexing_error will raise
   3808     #  InvalidIndexError. Otherwise we fall through and re-raise
   3809     #  the TypeError.
   3810     self._check_indexing_error(key)

KeyError: 'Type'
```

standardize the variables

In [8]:

```
from sklearn.preprocessing import StandardScaler
```

In [9]:

```
scaler = StandardScaler()
```

In [10]:

```
X = pd.DataFrame(scaler.fit_transform(credit_card_data.drop(["Class"],axis = 1)))
y = credit_card_data.Class
```

In [11]:

```
X.head()
```

Out[11]:

	0	1	2	3	4	5	6	7	
0	-1.996583	-0.694242	-0.044075	1.672773	0.973366	-0.245117	0.347068	0.193679	0.08
1	-1.996583	0.608496	0.161176	0.109797	0.316523	0.043483	-0.061820	-0.063700	0.07
2	-1.996562	-0.693500	-0.811578	1.169468	0.268231	-0.364572	1.351454	0.639776	0.2(
3	-1.996562	-0.493325	-0.112169	1.182516	-0.609727	-0.007469	0.936150	0.192071	0.3'
4	-1.996541	-0.591330	0.531541	1.021412	0.284655	-0.295015	0.071999	0.479302	-0.22

5 rows × 30 columns

Train Test Split

In [12]:

```
from sklearn.model_selection import train_test_split
```

In [13]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.30)
```

Using KNN

In [14]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [15]:

```
knn = KNeighborsClassifier(n_neighbors=1)
```

In [16]:

```
knn.fit(X_train,y_train)
```

Out[16]:

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
```

In [17]:

```
pred = knn.predict(X_test)
```

Predictions and Evaluations

In [18]:

```
from sklearn.metrics import classification_report,confusion_matrix
```

In [19]:

```
print(confusion_matrix(y_test,pred))
```

```
[[85279    11]
 [   30   123]]
```

In [20]:

```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85290
           1       0.92      0.80      0.86       153

    accuracy                           1.00     85443
   macro avg       0.96      0.90      0.93     85443
weighted avg       1.00      1.00      1.00     85443
```

In [21]:

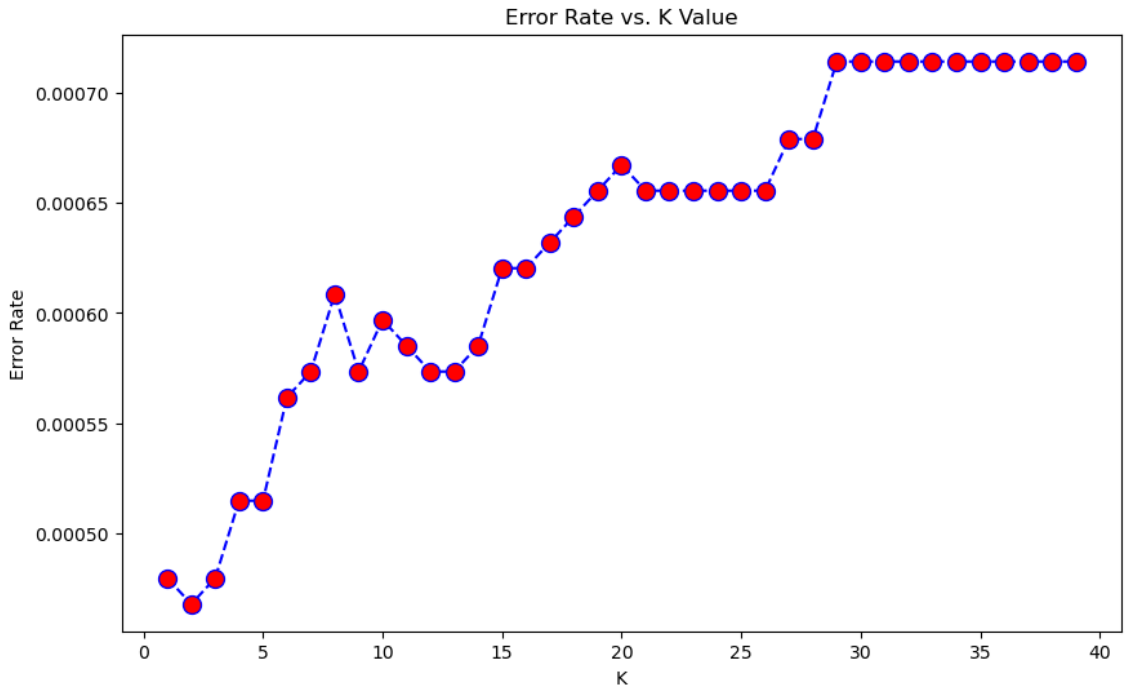
```
error_rate = []

# Will take some time
for i in range(1,40):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

In [22]:

```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
```

Out[22]:

```
Text(0, 0.5, 'Error Rate')
```



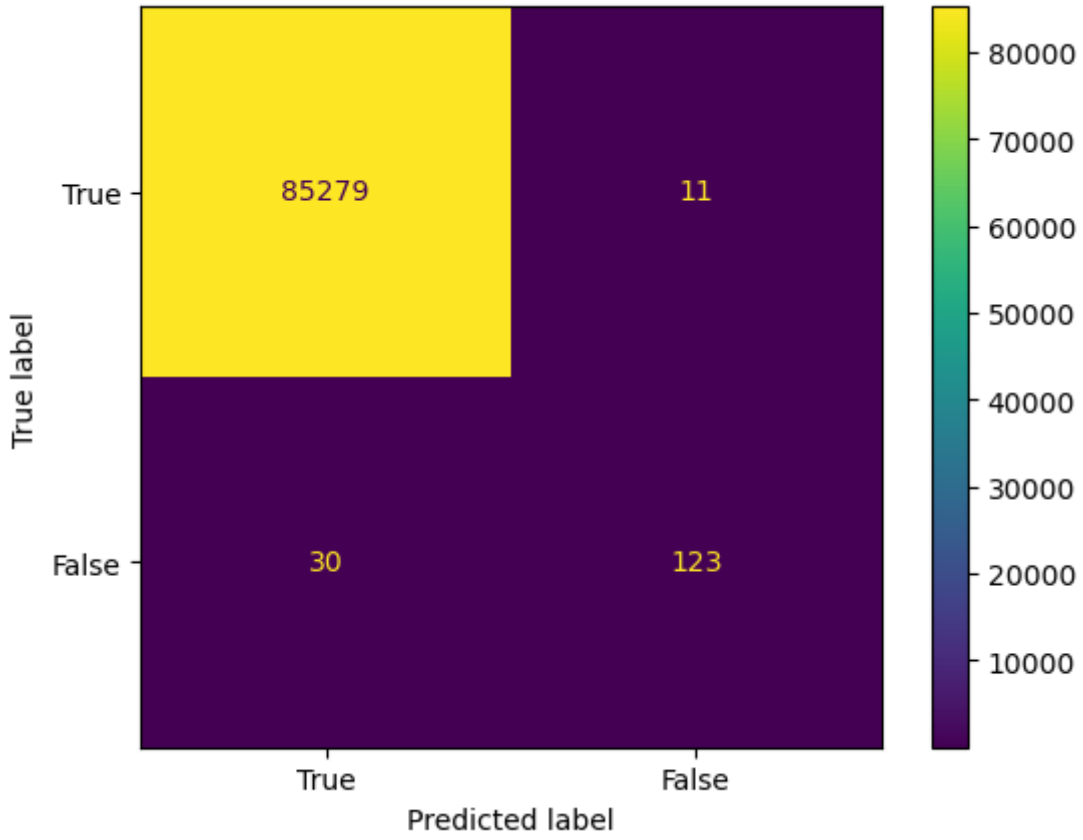
In [23]:

```
#Orginal K=1
knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(X_train,y_train)
pred = knn.predict(X_test)

print('WITH k=1')
print('\n')
print(confusion_matrix(y_test,pred))
print('\n')
print(classification_report(y_test,pred))
```

```
WITH k=1


[[85279    11]
 [   30   123]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85290
           1       0.92      0.80      0.86       153

    accuracy                           1.00     85443
   macro avg       0.96      0.90      0.93     85443
weighted avg       1.00      1.00      1.00     85443
```

In [24]:

```
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt

conf_matrix = confusion_matrix(y_test, pred)
vis = ConfusionMatrixDisplay(confusion_matrix = conf_matrix,display_labels = [True,False
vis.plot()
plt.grid(False)
plt.show()
```



In [ ]:

```
```