```
In [1]: import pandas as pd
```

```
In [2]: data = pd.read_csv("/Users/anchalbhondekar/Desktop/academics/PP/credit card fraud detection using ml/creditca
```

```
In [3]: pd.options.display.max_columns = None
```

```
In [4]: data.head()
```

Out[4]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852

```
In [5]: data.tail()
```

Out[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	4.356170	-1.593105	2.711941
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	-0.975926	-0.150189	0.915802
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	-0.484782	0.411614	0.063119
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	-0.399126	-1.933849	-0.962886
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	-0.915427	-1.040458	-0.031513

```
In [7]: data.shape
```

```
Out[7]: (284807, 31)
```

In [8]:

```
print("Number of Rows",data.shape[0])
print("Number of Columns",data.shape[1])
```

```
Number of Rows 284807
Number of Columns 31
```

In [9]:

```
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 [10]:

```
data.isnull().sum()
```

Out[10]:

```
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 [11]:

```
data.head()
```

Out[11]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852

In [12]:

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

In [13]:

```
sc = StandardScaler()
data['Amount']=sc.fit_transform(pd.DataFrame(data['Amount']))
```

In [14]:

```
data.head()
```

Out[14]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852

In [15]:

```
data = data.drop(['Time'],axis=1)
```

In [16]:
```
data.head()
```

Out[16]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.31
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.14
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.16
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.28
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.11

In [17]:
```
data.shape
```

Out[17]:
```
(284807, 30)
```

In [18]:
```
data.duplicated().any()
```

Out[18]:
```
True
```

In [19]:
```
data = data.drop_duplicates()
```

In [20]:
```
data.shape
```

Out[20]:
```
(275663, 30)
```

In [21]:
```
284807- 275663
```

Out[21]:
```
9144
```

In [22]:
```
data['Class'].value_counts()
```

Out[22]:
```
Class
0    275190
1       473
Name: count, dtype: int64
```

In [23]:

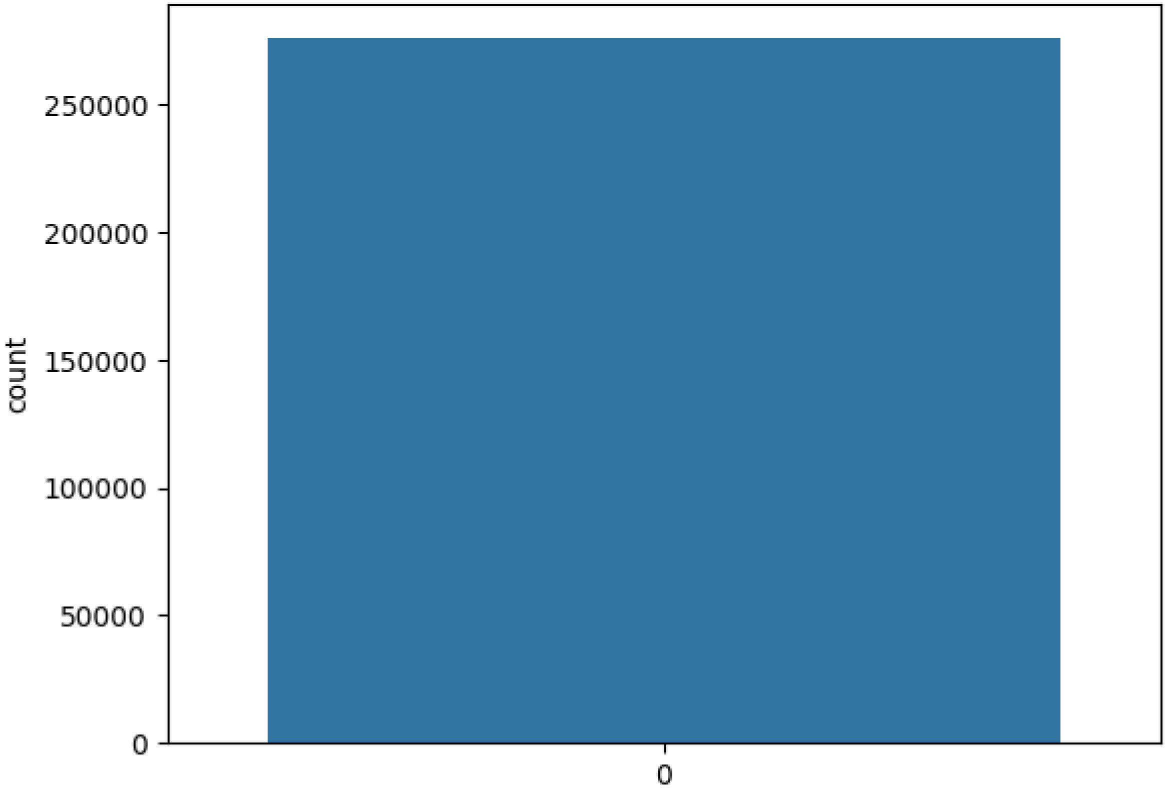
```
import seaborn as sns
```

In [24]:

```
sns.countplot(data['Class'])
```

Out[24]:

```
<Axes: ylabel='count'>
```



In [25]:

```
X = data.drop('Class',axis=1)
y = data['Class']
```

In [26]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,
                                                 random_state=42)
```

In [27]:
```
normal = data[data['Class']==0]
fraud = data[data['Class']==1]
```

In [28]:
```
normal.shape
```

Out[28]:
```
(275190, 30)
```

In [29]:
```
fraud.shape
```

Out[29]:
```
(473, 30)
```

In [30]:
```
normal_sample=normal.sample(n=473)
```

In [31]:
```
normal_sample.shape
```

Out[31]:
```
(473, 30)
```

In [32]:
```
new_data = pd.concat([normal_sample,fraud],ignore_index=True)
```

In [33]:
```
new_data['Class'].value_counts()
```

Out[33]:
```
Class
0    473
1    473
Name: count, dtype: int64
```

In [34]:
```
new_data.head()
```

Out[34]:

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	
0	0.071267	0.775378	-0.037550	-0.745492	0.948862	-0.076055	0.682721	0.043386	1.292418	-0.575466	0.525696	-2.863063	0.169523	1.9
1	-0.330704	-2.650395	-1.198694	0.216937	-0.727551	0.036969	1.112971	-0.120108	-0.265318	-0.386703	1.748991	0.383126	-1.206414	1.1
2	-0.852683	-0.136712	2.711335	-0.127106	-0.433043	1.026078	-0.574382	0.593640	1.369413	-1.164227	-2.112374	-0.457780	-1.540242	-0.9
3	-1.533500	1.229813	1.329231	-1.082743	-1.108438	-0.619590	0.100569	0.806861	-0.912606	-1.178890	1.276330	1.474004	0.962727	0.5
4	1.375659	-0.860802	-1.800710	1.193754	0.761542	0.963869	0.367476	0.114641	0.182964	0.054994	-0.203155	0.410284	0.116984	0.6

In [35]:

```
X = new_data.drop('Class',axis=1)
y = new_data['Class']
```

In [36]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,
                                                  random_state=42)
```

In [37]:

```
from sklearn.linear_model import LogisticRegression
log = LogisticRegression()
log.fit(X_train,y_train)
```

Out[37]:

```
LogisticRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [38]:

```
y_pred1 = log.predict(X_test)
```

In [39]:

```
from sklearn.metrics import accuracy_score
```

In [40]:

```
accuracy_score(y_test,y_pred1)
```

Out[40]:

```
0.9526315789473684
```

In [41]:

```
accuracy_score(y_test,y_pred1)
```

Out[41]:

```
0.9526315789473684
```

In [42]:

```
from sklearn.metrics import precision_score,recall_score,f1_score
```

In [43]:

```
precision_score(y_test,y_pred1)
```

Out[43]:

```
0.9894736842105263
```

In [44]:

```
precision_score(y_test,y_pred1)
```

Out[44]:

```
0.9894736842105263
```

```
In [45]: recall_score(y_test,y_pred1)

Out[45]: 0.9215686274509803

In [46]: recall_score(y_test,y_pred1)

Out[46]: 0.9215686274509803

In [47]: f1_score(y_test,y_pred1)

Out[47]: 0.9543147208121827

In [48]: f1_score(y_test,y_pred1)

Out[48]: 0.9543147208121827

In [49]: from sklearn.tree import DecisionTreeClassifier
         dt = DecisionTreeClassifier()
         dt.fit(X_train,y_train)

Out[49]: DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [50]: y_pred2 = dt.predict(X_test)

In [51]: accuracy_score(y_test,y_pred2)

Out[51]: 0.9105263157894737

In [52]: precision_score(y_test,y_pred2)

Out[52]: 0.912621359223301
```

```
In [53]: recall_score(y_test,y_pred2)

Out[53]: 0.9215686274509803
```

```
In [54]: f1_score(y_test,y_pred2)

Out[54]: 0.9170731707317074
```

```
In [55]: from sklearn.ensemble import RandomForestClassifier
         rf = RandomForestClassifier()
         rf.fit(X_train,y_train)

Out[55]: RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [56]: y_pred3 = rf.predict(X_test)
```

```
In [57]: accuracy_score(y_test,y_pred3)

Out[57]: 0.9421052631578948
```

```
In [58]: precision_score(y_test,y_pred3)

Out[58]: 0.9690721649484536
```

```
In [59]: recall_score(y_test,y_pred3)

Out[59]: 0.9215686274509803
```

```
In [60]: f1_score(y_test,y_pred3)

Out[60]: 0.9447236180904522
```

In [61]:
```
final_data = pd.DataFrame({'Models':['LR','DT','RF'],
              "ACC":[accuracy_score(y_test,y_pred1)*100,
                     accuracy_score(y_test,y_pred2)*100,
                     accuracy_score(y_test,y_pred3)*100
                    ]})
```

In [62]:
```
final_data
```

Out[62]:

	Models	ACC
0	LR	95.263158
1	DT	91.052632
2	RF	94.210526

In [69]:
```
X = data.drop('Class',axis=1)
y = data['Class']
```

In [70]:
```
X.shape
```

Out[70]:
```
(275663, 29)
```

In [71]:
```
y.shape
```

Out[71]:
```
(275663,)
```

In [72]:
```
from imblearn.over_sampling import SMOTE
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[72], line 1
----> 1 from imblearn.over_sampling import SMOTE

ModuleNotFoundError: No module named 'imblearn'
```

In [73]:

```
pip install imblearn
```

```
DEPRECATION: Loading egg at /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-package
s/jupyter-1.0.0-py3.11.egg is deprecated. pip 23.3 will enforce this behaviour change. A possible replaceme
nt is to use pip for package installation..
Collecting imblearn
  Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Collecting imbalanced-learn (from imblearn)
  Obtaining dependency information for imbalanced-learn from https://files.pythonhosted.org/packages/a3/9e/
fbe60a768502af54563dcb59ca7856f5a8833b3ad5ada658922e1ab09b7f/imbalanced_learn-0.11.0-py3-none-any.whl.metad
ata (https://files.pythonhosted.org/packages/a3/9e/fbe60a768502af54563dcb59ca7856f5a8833b3ad5ada658922e1ab0
9b7f/imbalanced_learn-0.11.0-py3-none-any.whl.metadata)
  Downloading imbalanced_learn-0.11.0-py3-none-any.whl.metadata (8.3 kB)
Requirement already satisfied: numpy>=1.17.3 in /Library/Frameworks/Python.framework/Versions/3.11/lib/pyth
on3.11/site-packages (from imbalanced-learn->imblearn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in /Library/Frameworks/Python.framework/Versions/3.11/lib/pytho
n3.11/site-packages (from imbalanced-learn->imblearn) (1.10.1)
Requirement already satisfied: scikit-learn>=1.0.2 in /Library/Frameworks/Python.framework/Versions/3.11/li
b/python3.11/site-packages (from imbalanced-learn->imblearn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in /Library/Frameworks/Python.framework/Versions/3.11/lib/pyth
on3.11/site-packages (from imbalanced-learn->imblearn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Library/Frameworks/Python.framework/Versions/3.11/l
ib/python3.11/site-packages (from imbalanced-learn->imblearn) (3.1.0)
Downloading imbalanced_learn-0.11.0-py3-none-any.whl (235 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 235.6/235.6 kB 339.3 kB/s eta 0:00:00a 0:00:01
Installing collected packages: imbalanced-learn, imblearn
Successfully installed imbalanced-learn-0.11.0 imblearn-0.0

[notice] A new release of pip is available: 23.2.1 -> 23.3.2
[notice] To update, run: pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

In [74]:

```
from imblearn.over_sampling import SMOTE
```

In [75]:

```
X_res,y_res = SMOTE().fit_resample(X,y)
```

In [76]:
```
y_res.value_counts()
```

Out[76]:
```
Class
0    275190
1    275190
Name: count, dtype: int64
```

In [90]:
```
import joblib
```

In [91]:
```
joblib.dump(rf1,"credit_card_model")
```

Out[91]:
```
['credit_card_model']
```

In [92]:
```
model = joblib.load("credit_card_model")
```

In [93]:
```
pred = model.predict([[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1]])
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/sklearn/base.py:439: UserWa
rning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  warnings.warn(
```

In [94]:
```
if pred == 0:
    print("Normal Transcation")
else:
    print("Fraudulent Transcation")
```

```
Normal Transcation
```

In [7]:

```
from tkinter import *
import joblib

def show_entry_fields():
    v1=float(e1.get())
    v2=float(e2.get())
    v3=float(e3.get())
    v4=float(e4.get())
    v5=float(e5.get())
    v6=float(e6.get())

    v7=float(e7.get())
    v8=float(e8.get())
    v9=float(e9.get())
    v10=float(e10.get())
    v11=float(e11.get())
    v12=float(e12.get())

    v13=float(e13.get())
    v14=float(e14.get())
    v15=float(e15.get())
    v16=float(e16.get())
    v17=float(e17.get())
    v18=float(e18.get())


    v19=float(e19.get())
    v20=float(e20.get())
    v21=float(e21.get())
    v22=float(e22.get())
    v23=float(e23.get())
    v24=float(e24.get())


    v25=float(e25.get())
    v26=float(e26.get())
    v27=float(e27.get())
    v28=float(e28.get())
    v29=float(e29.get())
    model = joblib.load('model_credit.pkl')
    y_pred = model.predict([[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,v29]])
    list1=[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,v29]

    result = []
    if y_pred ==0:
        result.append("Normal Transcation")
    else:
        result.append("Fraudulent Transcation")
        print("######################################")
        print("Credit Card Fraud Detection System", result)
        print("######################################")



    Label(master, text="Final Prediction from the model - credit card fraud detection").grid(row=31)
    Label(master, text=result).grid(row=32)



master = Tk()
master.title("Credit Card Fraud Detection System")


label = Label(master, text = "Credit Card Fraud Detection System", bg = "black", fg = "white",width = 30).gri


Label(master, text="Enter value of V1").grid(row=1)
Label(master, text="Enter value of V2").grid(row=2)
Label(master, text="Enter value of V3").grid(row=3)
Label(master, text="Enter value of V4").grid(row=4)
Label(master, text="Enter value of V5").grid(row=5)
Label(master, text="Enter value of V6").grid(row=6)

Label(master, text="Enter value of V7").grid(row=7)
Label(master, text="Enter value of V8").grid(row=8)
Label(master, text="Enter value of V9").grid(row=9)
Label(master, text="Enter value of V10").grid(row=10)
Label(master, text="Enter value of V11").grid(row=11)
Label(master, text="Enter value of V12").grid(row=12)
Label(master, text="Enter value of V13").grid(row=13)
Label(master, text="Enter value of V14").grid(row=14)
Label(master, text="Enter value of V15").grid(row=15)
Label(master, text="Enter value of V16").grid(row=16)
Label(master, text="Enter value of V17").grid(row=17)
Label(master, text="Enter value of V18").grid(row=18)

Label(master, text="Enter value of V19").grid(row=19)
Label(master, text="Enter value of V20").grid(row=20)
Label(master, text="Enter value of V21").grid(row=21)
Label(master, text="Enter value of V22").grid(row=22)
Label(master, text="Enter value of V23").grid(row=23)
Label(master, text="Enter value of V24").grid(row=24)

Label(master, text="Enter value of V25").grid(row=25)
Label(master, text="Enter value of V26").grid(row=26)
Label(master, text="Enter value of V27").grid(row=27)
Label(master, text="Enter value of V28").grid(row=28)
Label(master, text="Enter value of V29").grid(row=29)

e1 = Entry(master)
e2 = Entry(master)
e3 = Entry(master)
e4 = Entry(master)
e5 = Entry(master)
e6 = Entry(master)

e7 = Entry(master)
e8 = Entry(master)
e9 = Entry(master)
e10 = Entry(master)
e11 = Entry(master)
e12 = Entry(master)

e13 = Entry(master)
e14 = Entry(master)
e15 = Entry(master)
e16 = Entry(master)
e17 = Entry(master)
e18= Entry(master)
```

```
e19 = Entry(master)
e20 = Entry(master)
e21 = Entry(master)
e22 = Entry(master)
e23= Entry(master)
e24 = Entry(master)


e25 = Entry(master)
e26= Entry(master)
e27 = Entry(master)
e28 = Entry(master)
e29= Entry(master)

e1.grid(row=1, column=1)
e2.grid(row=2, column=1)
e3.grid(row=3, column=1)
e4.grid(row=4, column=1)
e5.grid(row=5, column=1)
e6.grid(row=6, column=1)

e7.grid(row=7, column=1)
e8.grid(row=8, column=1)
e9.grid(row=9, column=1)
e10.grid(row=10, column=1)
e11.grid(row=11, column=1)
e12.grid(row=12, column=1)


e13.grid(row=13, column=1)
e14.grid(row=14, column=1)
e15.grid(row=15, column=1)
e16.grid(row=16, column=1)
e17.grid(row=17, column=1)
e18.grid(row=18, column=1)


e19.grid(row=19, column=1)
e20.grid(row=20, column=1)
e21.grid(row=21, column=1)
e22.grid(row=22, column=1)
e23.grid(row=23, column=1)
```

```
e24.grid(row=24, column=1)

e25.grid(row=25, column=1)
e26.grid(row=26, column=1)
e27.grid(row=27, column=1)
e28.grid(row=28, column=1)
e29.grid(row=29, column=1)

Button(master, text='Predict', command=show_entry_fields).grid(row=30, column=1, sticky=W, pady=4)

mainloop( )
```

In [ ]:

In [ ]:

In [ ]: