Random Forest

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import gridspec
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import  accuracy_score, precision_score
```

```
data = pd.read_csv('/content/drive/MyDrive/creditcard.csv')
data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267

5 rows × 31 columns

```
data.dropna(inplace=True)
```

```
data.shape
```

```
(284807, 31)
```

```
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Time	284807.0	9.481386e+04	47488.145955	0.000000	54201.500000	84692.000000	139320.500000	172792.000000
V1	284807.0	1.168375e-15	1.958696	-56.407510	-0.920373	0.018109	1.315642	2.454930
V2	284807.0	3.416908e-16	1.651309	-72.715728	-0.598550	0.065486	0.803724	22.057729
V3	284807.0	-1.379537e-15	1.516255	-48.325589	-0.890365	0.179846	1.027196	9.382558
V4	284807.0	2.074095e-15	1.415869	-5.683171	-0.848640	-0.019847	0.743341	16.875344
V5	284807.0	9.604066e-16	1.380247	-113.743307	-0.691597	-0.054336	0.611926	34.801666
V6	284807.0	1.487313e-15	1.332271	-26.160506	-0.768296	-0.274187	0.398565	73.301626
V7	284807.0	-5.556467e-16	1.237094	-43.557242	-0.554076	0.040103	0.570436	120.589494
V8	284807.0	1.213481e-16	1.194353	-73.216718	-0.208630	0.022358	0.327346	20.007208
V9	284807.0	-2.406331e-15	1.098632	-13.434066	-0.643098	-0.051429	0.597139	15.594995
V10	284807.0	2.239053e-15	1.088850	-24.588262	-0.535426	-0.092917	0.453923	23.745136
V11	284807.0	1.673327e-15	1.020713	-4.797473	-0.762494	-0.032757	0.739593	12.018913
V12	284807.0	-1.247012e-15	0.999201	-18.683715	-0.405571	0.140033	0.618238	7.848392
V13	284807.0	8.190001e-16	0.995274	-5.791881	-0.648539	-0.013568	0.662505	7.126883
V14	284807.0	1.207294e-15	0.958596	-19.214325	-0.425574	0.050601	0.493150	10.526766
V15	284807.0	4.887456e-15	0.915316	-4.498945	-0.582884	0.048072	0.648821	8.877742
V16	284807.0	1.437716e-15	0.876253	-14.129855	-0.468037	0.066413	0.523296	17.315112
V17	284807.0	-3.772171e-16	0.849337	-25.162799	-0.483748	-0.065676	0.399675	9.253526
V18	284807.0	9.564149e-16	0.838176	-9.498746	-0.498850	-0.003636	0.500807	5.041069
V19	284807.0	1.039917e-15	0.814041	-7.213527	-0.456299	0.003735	0.458949	5.591971
V20	284807.0	6.406204e-16	0.770925	-54.497720	-0.211721	-0.062481	0.133041	39.420904
V21	284807.0	1.654067e-16	0.734524	-34.830382	-0.228395	-0.029450	0.186377	27.202839
V22	284807.0	-3.568593e-16	0.725702	-10.933144	-0.542350	0.006782	0.528554	10.503090
V23	284807.0	2.578648e-16	0.624460	-44.807735	-0.161846	-0.011193	0.147642	22.528412

V24	284807.0	4.473266e-15	0.605647	-2.836627	-0.354586	0.040976	0.439527	4.584549
V25	284807.0	5.340915e-16	0.521278	-10.295397	-0.317145	0.016594	0.350716	7.519589
V26	284807.0	1.683437e-15	0.482227	-2.604551	-0.326984	-0.052139	0.240952	3.517346
V27	284807.0	-3.660091e-16	0.403632	-22.565679	-0.070840	0.001342	0.091045	31.612198
V28	284807.0	-1.227390e-16	0.330083	-15.430084	-0.052960	0.011244	0.078280	33.847808
Amount	284807.0	8.834962e+01	250.120109	0.000000	5.600000	22.000000	77.165000	25691.160000
Class	284807.0	1.727486e-03	0.041527	0.000000	0.000000	0.000000	0.000000	1.000000

```
fraud = data[data.Class == 1]
valid = data[data.Class == 0]
```

```
fraud.Amount.describe()
```

```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

```
valid.Amount.describe()
```

```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

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

```
((284807, 30), (284807,))
```

```
X_data = X.values
y_data = y.values
```

```
X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, test_size = .2, random_state = 42)
```

```
rfc = RandomForestClassifier()
```

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

```
▾ RandomForestClassifier
RandomForestClassifier()
```

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

```
acc = accuracy_score(y_test, pred)
acc
```

```
0.9995962220427653
```

```
prec = precision_score(y_test, pred)
prec
```

```
0.9746835443037974
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Logistic Regression

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score
```

```
data = pd.read_csv('/content/drive/MyDrive/creditcard.csv')
```

```
data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.1
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.1
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.9
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.1
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.1

5 rows × 31 columns

```
data.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.84807
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.2134
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.1943
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.3216
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.0862
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.2358
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.2734
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.0007

8 rows × 31 columns

```
data['Class'].value_counts()
```

```
0    284315
1       492
Name: Class, dtype: int64
```

```
legit=data[data.Class==0]
fraud=data[data.Class==1]
```

```
legit.Amount.describe()
```

```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

```
fraud.Amount.describe()
```

```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

```
legit_sample = legit.sample(n=492)
```

```
new_dataset = pd.concat([legit_sample, fraud], axis=0)
new_dataset
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	
68689	53100.0	-1.036107	1.310444	1.694569	0.549866	0.086610	-0.122477	0.517944	-0.104989	-0.337299	...	-0.319768	-0.56
178756	123768.0	0.307645	-0.134362	-0.201855	-2.154790	0.181265	-0.016183	0.824177	-0.290879	-1.049938	...	-0.036491	-0.16
44885	42125.0	-0.560224	0.867220	1.805506	0.439807	0.185886	-0.102419	0.738651	-0.081385	-0.524958	...	-0.175373	-0.34
240047	150407.0	2.044278	0.089607	-1.811933	0.222810	0.652724	-0.297039	0.061380	-0.043791	0.254553	...	-0.327021	-0.86
120399	75815.0	1.121923	-0.776216	0.379933	0.287319	-0.639505	0.645173	-0.646367	0.094758	-0.794018	...	-0.188615	-0.12
...	...	...	...	...	...	...	...	...	...	...	...	...	
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.064945	...	0.778584	-0.31
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.127396	...	0.370612	0.02
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.652250	...	0.751826	0.83
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.632333	...	0.583276	-0.26
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.577829	...	-0.164350	-0.29

984 rows × 31 columns

```
new_dataset['Class'].value_counts()
```

```
0    492
1    492
Name: Class, dtype: int64
```

```
X = new_dataset.drop(columns='Class', axis=1)
Y = new_dataset['Class']
```

```
print(X)
```

```
            Time        V1        V2        V3        V4        V5        V6  \
68689    53100.0 -1.036107  1.310444  1.694569  0.549866  0.086610 -0.122477
178756  123768.0  0.307645 -0.134362 -0.201855 -2.154790  0.181265 -0.016183
44885    42125.0 -0.560224  0.867220  1.805506  0.439807  0.185886 -0.102419
240047  150407.0  2.044278  0.089607 -1.811933  0.222810  0.652724 -0.297039
120399   75815.0  1.121923 -0.776216  0.379933  0.287319 -0.639505  0.645173
...          ...       ...       ...       ...       ...       ...       ...
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494
280143  169347.0  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.326536
280149  169351.0 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.003346
281144  169966.0 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.943548
281674  170348.0  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.096695

              V7        V8        V9  ...       V20       V21       V22  \
68689   0.517944 -0.104989 -0.337299  ...  0.645328 -0.319768 -0.566693
178756  0.824177 -0.290879 -1.049938  ...  0.042822 -0.036491 -0.164773
44885   0.738651 -0.081385 -0.524958  ...  0.223098 -0.175373 -0.343294
240047  0.061380 -0.043791  0.254553  ... -0.141209 -0.327021 -0.865309
120399 -0.646367  0.094758 -0.794018  ... -0.202461 -0.188615 -0.129553
...          ...       ...       ...  ...       ...       ...       ...
279863 -0.882850  0.697211 -2.064945  ...  1.252967  0.778584 -0.319189
280143 -1.413170  0.248525 -1.127396  ...  0.226138  0.370612  0.028234
280149 -2.234739  1.210158 -0.652250  ...  0.247968  0.751826  0.834108
281144 -2.208002  1.058733 -1.632333  ...  0.306271  0.583276 -0.269209
281674  0.223050 -0.068384  0.577829  ... -0.017652 -0.164350 -0.295135

             V23       V24       V25       V26       V27       V28  Amount
68689  -0.199955 -0.007223  0.044895  0.369299  0.201383  0.067522    3.99
178756  0.052353  0.144043 -0.422244 -0.556323 -0.269563 -0.259981   96.80
44885  -0.280159 -0.045975  0.399565  0.428836  0.033619  0.067141   28.99
```

```
240047  0.296283  0.126098 -0.262881  0.178221 -0.063821 -0.044256    1.98
120399 -0.319534 -0.824112  0.645396 -0.182539  0.058705  0.033676  119.50
...          ...       ...       ...       ...       ...       ...      ...
279863  0.639419 -0.294885  0.537503  0.788395  0.292680  0.147968  390.00
280143 -0.145640 -0.081049  0.521875  0.739467  0.389152  0.186637    0.76
280149  0.190944  0.032070 -0.739695  0.471111  0.385107  0.194361   77.89
281144 -0.456108 -0.183659 -0.328168  0.606116  0.884876 -0.253700  245.00
281674 -0.072173 -0.450261  0.313267 -0.289617  0.002988 -0.015309   42.53

[984 rows x 30 columns]
```

```
print(Y)
```

```
68689     0
178756    0
44885     0
240047    0
120399    0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

```
model = LogisticRegression()
```

```
model.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (sta
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

```
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

```
print('Accuracy score on Test Data : ', test_data_accuracy)
```

```
Accuracy score on Test Data :  0.9390862944162437
```

```
precis=precision_score(X_test_prediction, Y_test)
```

```
print(precis)
```

```
0.9081632653061225
```

Decision Tree

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```