```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,confusion_matrix,precision_score,recall_score,ConfusionMatrixDisplay,
```

```
data=pd.read_csv('creditcard.csv')
```

```
data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V2
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.11047
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.10128
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.90941
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.19032
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.13745

5 rows × 31 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9965 entries, 0 to 9964
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Time    9965 non-null   int64  
 1   V1      9965 non-null   float64
 2   V2      9965 non-null   float64
 3   V3      9964 non-null   float64
 4   V4      9964 non-null   float64
 5   V5      9964 non-null   float64
 6   V6      9964 non-null   float64
 7   V7      9964 non-null   float64
 8   V8      9964 non-null   float64
 9   V9      9964 non-null   float64
 10  V10     9964 non-null   float64
 11  V11     9964 non-null   float64
 12  V12     9964 non-null   float64
 13  V13     9964 non-null   float64
 14  V14     9964 non-null   float64
 15  V15     9964 non-null   float64
 16  V16     9964 non-null   float64
 17  V17     9964 non-null   float64
 18  V18     9964 non-null   float64
 19  V19     9964 non-null   float64
 20  V20     9964 non-null   float64
 21  V21     9964 non-null   float64
 22  V22     9964 non-null   float64
 23  V23     9964 non-null   float64
 24  V24     9964 non-null   float64
 25  V25     9964 non-null   float64
 26  V26     9964 non-null   float64
 27  V27     9964 non-null   float64
 28  V28     9964 non-null   float64
 29  Amount  9964 non-null   float64
 30  Class   9964 non-null   float64
dtypes: float64(30), int64(1)
memory usage: 2.4 MB
```

```
data.isnull()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22
0	False	False	False	False	False	False	False	False	False	False	...	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	..
9960	False	False	False	False	False	False	False	False	False	False	...	False	False
9961	False	False	False	False	False	False	False	False	False	False	...	False	False
9962	False	False	False	False	False	False	False	False	False	False	...	False	False
9963	False	False	False	False	False	False	False	False	False	False	...	False	False
9964	False	False	False	True	True	True	True	True	True	True	...	True	True

9965 rows × 31 columns

```
data.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	
9960	14837	1.286884	-0.124610	0.148283	-0.259343	0.248357	0.896718	-0.626627	0.227693	1.618678	...	-0.381864	-0.904515	-0.02
9961	14854	1.318742	0.496408	0.114876	0.695262	0.170133	-0.537180	0.025492	-0.272931	1.267298	...	-0.484943	-1.111176	0.02
9962	14857	1.241757	0.419587	0.806183	0.894811	-0.507886	-1.118126	0.018908	-0.343335	1.210781	...	-0.379396	-0.817785	0.18
9963	14861	1.304800	-0.052885	0.415235	-0.081725	-0.223525	0.097752	-0.561240	0.067228	1.617203	...	-0.379597	-0.929204	0.02
9964	14864	-1.747939	3.712444	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	

5 rows × 31 columns

Handling Null Values

```
credit_card_data=data.fillna(value=0)
credit_card_data
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	
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	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	-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	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.1
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9960	14837	1.286884	-0.124610	0.148283	-0.259343	0.248357	0.896718	-0.626627	0.227693	1.618678	...	-0.381864	-0.904515	-0.0
9961	14854	1.318742	0.496408	0.114876	0.695262	0.170133	-0.537180	0.025492	-0.272931	1.267298	...	-0.484943	-1.111176	0.0
9962	14857	1.241757	0.419587	0.806183	0.894811	-0.507886	-1.118126	0.018908	-0.343335	1.210781	...	-0.379396	-0.817785	0.1
9963	14861	1.304800	-0.052885	0.415235	-0.081725	-0.223525	0.097752	-0.561240	0.067228	1.617203	...	-0.379597	-0.929204	0.0
9964	14864	-1.747939	3.712444	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.0

9965 rows × 31 columns

```
credit_card_data.isnull().sum()
```

```
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
```

```
credit_card_data.duplicated().sum()
credit_card_data.drop_duplicates()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	
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	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	-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	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.1
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9960	14837	1.286884	-0.124610	0.148283	-0.259343	0.248357	0.896718	-0.626627	0.227693	1.618678	...	-0.381864	-0.904515	-0.0
9961	14854	1.318742	0.496408	0.114876	0.695262	0.170133	-0.537180	0.025492	-0.272931	1.267298	...	-0.484943	-1.111176	0.0
9962	14857	1.241757	0.419587	0.806183	0.894811	-0.507886	-1.118126	0.018908	-0.343335	1.210781	...	-0.379396	-0.817785	0.1
9963	14861	1.304800	-0.052885	0.415235	-0.081725	-0.223525	0.097752	-0.561240	0.067228	1.617203	...	-0.379597	-0.929204	0.0
9964	14864	-1.747939	3.712444	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.0

9923 rows × 31 columns

```
#checking the distribution of data among the legitimate and fraud transactions
credit_card_data['Class'].value_counts()
```

```
0.0    9927
1.0      38
Name: Class, dtype: int64
```

```
#THIS IS HIGHLY UNBALANCED DATA
#0-NORMAL TRANSACTION
#1-FRAUD TRANSACTION
```

Exploratory Data Analysis

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

```
print(legit.shape)
print(fraud.shape)
```

```
(19813, 31)
(85, 31)
```

```
#statistical measures
legit.Amount.describe()
```

```
count    9927.000000
mean       62.913163
std       184.033110
min         0.000000
```

```
25%         5.155000
50%        15.950000
75%        51.045000
max      7712.430000
Name: Amount, dtype: float64
```

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

```
count      38.000000
mean       75.730526
std       304.521215
min         0.000000
25%         1.000000
50%         1.000000
75%         1.000000
max      1809.680000
Name: Amount, dtype: float64
```

```
credit_card_data.groupby('Class').mean()
```

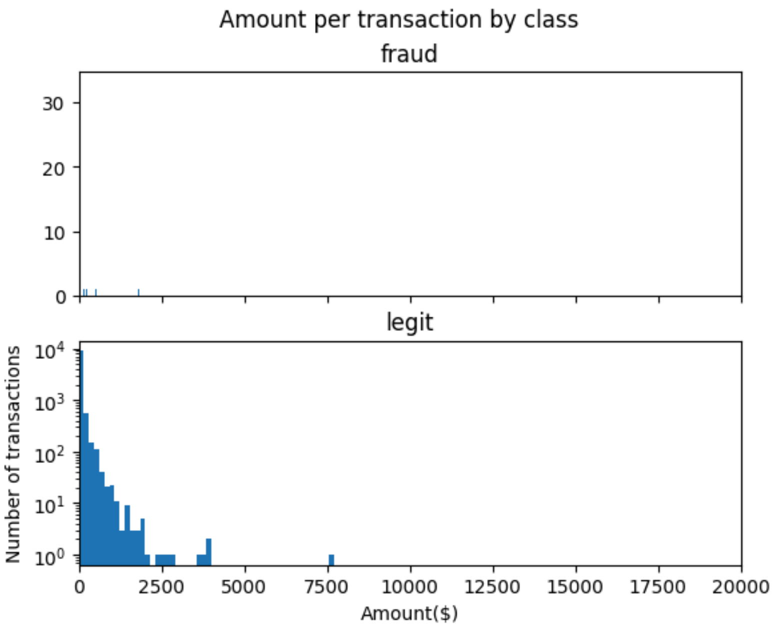
Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21
0.0	5922.508512	-0.235729	0.267180	0.934294	0.239711	-0.041739	0.140911	-0.057655	-0.069964	0.812921	...	0.0261	-0.055133
1.0	9063.157895	-1.796662	3.810809	-6.415255	5.618146	-1.247563	-2.111328	-3.777261	1.150469	-2.276505	...	0.4677	0.741934

2 rows × 30 columns

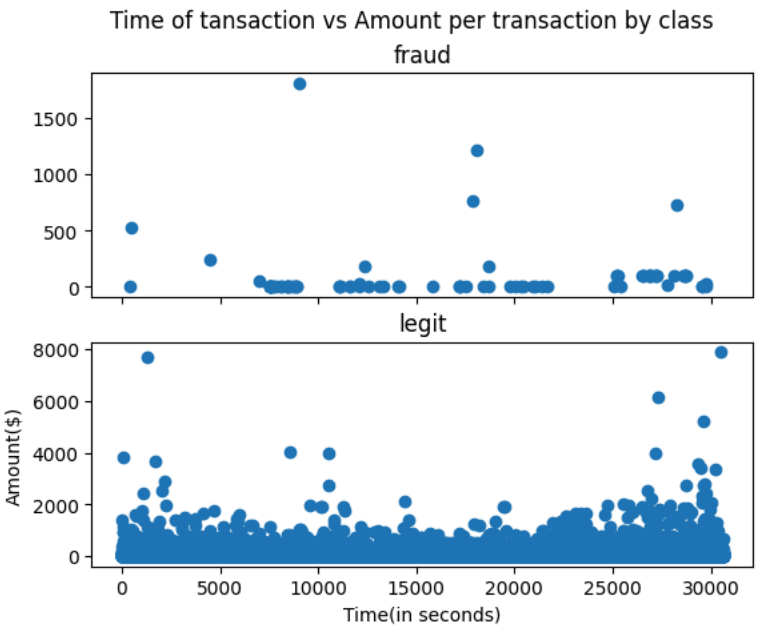
```
count_classes=pd.value_counts(credit_card_data['Class'],sort=True)
count_classes.plot(kind='bar',rot=0)
plt.title("Transaction class distribution")
plt.xlabel('Class')
plt.ylabel('Frequency')
```

```
Text(0, 0.5, 'Frequency')
```


```
f,(ax1,ax2)=plt.subplots(2,1,sharex=True)
f.suptitle('Amount per transaction by class')
bins=50
ax1.hist(fraud.Amount,bins=bins)
ax1.set_title('fraud')
ax2.hist(legit.Amount,bins=bins)
ax2.set_title('legit')
plt.xlabel('Amount($)')
plt.ylabel('Number of transactions')
plt.xlim(0,20000)
plt.yscale('log')
plt.show()
```

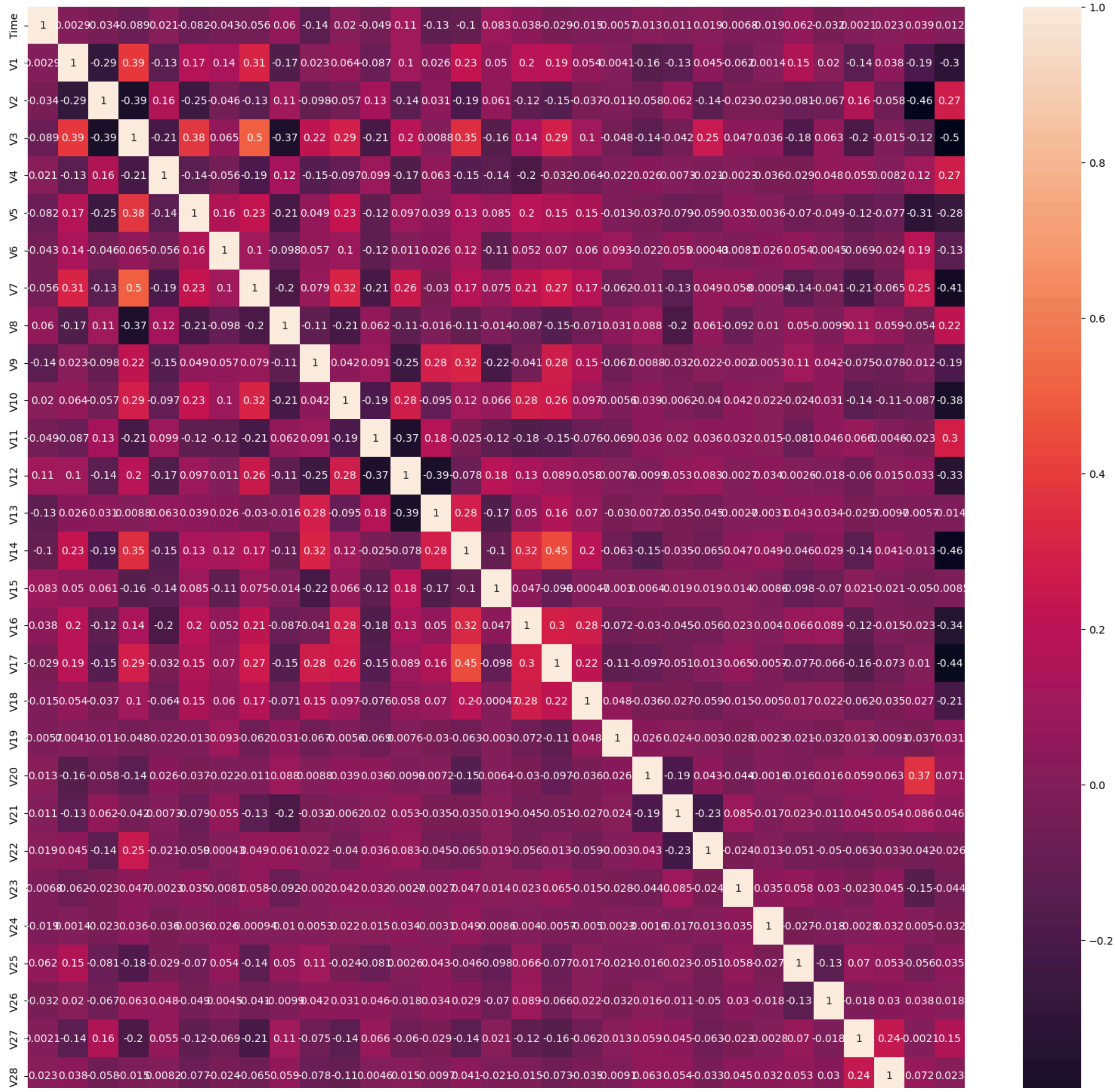



```
#we will be checking the time frame from the data
f,(ax1,ax2)=plt.subplots(2,1,sharex=True)
f.suptitle('Time of tansaction vs Amount per transaction by class')
ax1.scatter(fraud.Time,fraud.Amount)
ax1.set_title('fraud')
ax2.scatter(legit.Time,legit.Amount)
ax2.set_title('legit')
plt.ylabel('Amount($)')
plt.xlabel('Time(in seconds)')

plt.show()
```




```
#Correlation
import seaborn as sns
cor=credit_card_data.corr()
top_corr_features=cor.index
plt.figure(figsize=(20,20))

sns.heatmap(credit_card_data[top_corr_features].corr(),annot=True)
```

```
<Axes: >
```



Model Training

```
columns=credit_card_data.columns.tolist()
columns=[c for c in columns if c not in ['Class']]
target='Class'
state=np.random.RandomState(2)
x=credit_card_data[columns]
y=credit_card_data[target]
print(x.shape)
print(y.shape)
```

```
(9965, 30)
(9965,)
```

```
print(x)
```

```
       Time        V1        V2        V3        V4        V5        V6  \
0         0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388
1         0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361
2         1 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499
3         1 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203
4         2 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921
...     ...       ...       ...       ...       ...       ...       ...
9960  14837  1.286884 -0.124610  0.148283 -0.259343  0.248357  0.896718
9961  14854  1.318742  0.496408  0.114876  0.695262  0.170133 -0.537180
9962  14857  1.241757  0.419587  0.806183  0.894811 -0.507886 -1.118126
9963  14861  1.304800 -0.052885  0.415235 -0.081725 -0.223525  0.097752
```

```
9964  14864 -1.747939  3.712444  0.000000  0.000000  0.000000  0.000000

            V7        V8        V9  ...       V20       V21       V22  \
0     0.239599  0.098698  0.363787  ...  0.251412 -0.018307  0.277838
1    -0.078803  0.085102 -0.255425  ... -0.069083 -0.225775 -0.638672
2     0.791461  0.247676 -1.514654  ...  0.524980  0.247998  0.771679
3     0.237609  0.377436 -1.387024  ... -0.208038 -0.108300  0.005274
4     0.592941 -0.270533  0.817739  ...  0.408542 -0.009431  0.798278
...        ...       ...       ...  ...       ...       ...       ...
9960 -0.626627  0.227693  1.618678  ... -0.093459 -0.381864 -0.904515
9961  0.025492 -0.272931  1.267298  ... -0.051795 -0.484943 -1.111176
9962  0.018908 -0.343335  1.210781  ... -0.107163 -0.379396 -0.817785
9963 -0.561240  0.067228  1.617203  ... -0.108758 -0.379597 -0.929204
9964  0.000000  0.000000  0.000000  ...  0.000000  0.000000  0.000000

           V23       V24       V25       V26       V27       V28  Amount
0    -0.110474  0.066928  0.128539 -0.189115  0.133558 -0.021053  149.62
1     0.101288 -0.339846  0.167170  0.125895 -0.008983  0.014724    2.69
2     0.909412 -0.689281 -0.327642 -0.139097 -0.055353 -0.059752  378.66
3    -0.190321 -1.175575  0.647376 -0.221929  0.062723  0.061458  123.50
4    -0.137458  0.141267 -0.206010  0.502292  0.219422  0.215153   69.99
...        ...       ...       ...       ...       ...       ...     ...
9960 -0.027985 -1.743540  0.090885  0.870425 -0.084116 -0.022744   12.18
9961  0.028259 -0.549934  0.328634  0.106061 -0.046154  0.017304    1.78
9962  0.181425  0.662879  0.172535  0.033636 -0.051084  0.017208    1.29
9963  0.020955 -0.877006  0.084384  0.807465 -0.099851 -0.015404    4.72
9964  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000    0.00

[9965 rows x 30 columns]
```

```
print(y)
```

```
0       0.0
1       0.0
2       0.0
3       0.0
4       0.0
       ...
9960    0.0
9961    0.0
9962    0.0
9963    0.0
9964    0.0
Name: Class, Length: 9965, dtype: float64
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,stratify=y,random_state=2)
```

```
print(x.shape,x_train.shape,x_test.shape)
```

```
(9965, 30) (7972, 30) (1993, 30)
```

```
#model fitting
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
x_train_pred=rf.predict(x_train)
```

```
model = LogisticRegression(solver='saga')
# training the Logistic Regression Model with Training Data
model.fit(x_train, y_train)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
 intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='auto', n_jobs=None, penalty='l2',
 random_state=None, solver='saga', tol=0.0001, verbose=0,
warm_start=False)
```

Model Evaluation

```
#train data accuracy
accuracy=accuracy_score(x_train_pred,y_train)
print("train_Accuracy:",accuracy)
```

```
train_Accuracy: 1.0
```

```
#accuracy on test datatest
x_test_pred=rf.predict(x_test)
test_data_accuracy=accuracy_score(x_test_pred,y_test)
print("Test_data_accuracy",test_data_accuracy)
```

```
Test_data_accuracy 0.9989949748743718
```

```
precision=precision_score(y_test,x_test_pred)
print("Precision:",precision)
```

```
Precision: 0.9333333333333333
```

```
f1score=f1_score(y_test,x_test_pred)
print("f1_score:",f1score)
```

```
f1_score: 0.8749999999999999
```

```
recall=recall_score(x_test_pred,x_test_pred)
print('recall:',recall)
```

```
recall: 1.0
```

```
confusion_matrix = metrics.confusion_matrix(y_test, x_test_pred)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.show()
```