```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive
drive.mount('/content/drive')
from matplotlib import style
style.use('ggplot')

import warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

```
Mounted at /content/drive
```

```
train = pd.read_csv("/content/drive/MyDrive/credit-data/fraudTrain.csv")
test = pd.read_csv("/content/drive/MyDrive/credit-data/fraudTest.csv")
```

```
train.head() # top 5 rows are displayed
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	a
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.

5 rows × 23 columns

```
test.head()
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	l
0	0	2020-06-21 12:14:25	2291163933867244	fraud_Kirlin and Sons	personal_care	2.86	Jeff	E
1	1	2020-06-21 12:14:33	3573030041201292	fraud_Sporer-Keebler	personal_care	29.84	Joanne	Willi
2	2	2020-06-21 12:14:53	3598215285024754	fraud_Swaniawski, Nitzsche and Welch	health_fitness	41.28	Ashley	Lo
3	3	2020-06-21 12:15:15	3591919803438423	fraud_Haley Group	misc_pos	60.05	Brian	Willi
4	4	2020-06-21 12:15:17	3526826139003047	fraud_Johnston-Casper	travel	3.19	Nathan	Mas

5 rows × 23 columns

```
#combined for data cleaning and visualization
data = pd.concat([train,test], axis = 0) #axis = 0 means row wise(stacked vertically)
data.head()
```

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	l
0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Ba
1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	
2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanc
3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	W
4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Ga

5 rows × 23 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1852394 entries, 0 to 555718
Data columns (total 23 columns):
 #   Column                 Dtype  
---  ------                 -----  
 0   Unnamed: 0             int64  
 1   trans_date_trans_time  object 
 2   cc_num                 int64  
 3   merchant               object 
 4   category               object 
 5   amt                    float64
 6   first                  object 
 7   last                   object 
 8   gender                 object 
 9   street                 object 
 10  city                   object 
 11  state                  object 
 12  zip                    int64  
 13  lat                    float64
 14  long                   float64
 15  city_pop               int64  
 16  job                    object 
 17  dob                    object 
 18  trans_num              object 
 19  unix_time              int64  
 20  merch_lat              float64
 21  merch_long             float64
 22  is_fraud               int64  
dtypes: float64(5), int64(6), object(12)
memory usage: 339.2+ MB
```

```
data.reset_index(inplace = True)
data.head(10)
```

	index	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	fir
0	0	0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennif
1	1	1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephan
2	2	2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edwa
3	3	3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeren
4	4	4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyl
5	5	5	2019-01-01 00:04:08	4767265376804500	fraud_Stroman, Hudson and Erdman	gas_transport	94.63	Jennif
					fraud_Rowe-			

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1852394 entries, 0 to 1852393
Data columns (total 24 columns):
 #   Column                 Dtype  
---  ------                 -----  
 0   index                  int64  
 1   Unnamed: 0             int64  
 2   trans_date_trans_time  object 
 3   cc_num                 int64  
 4   merchant               object 
 5   category               object 
 6   amt                    float64
 7   first                  object 
 8   last                   object 
 9   gender                 object 
 10  street                 object 
 11  city                   object 
 12  state                  object 
 13  zip                    int64  
 14  lat                    float64
 15  long                   float64
 16  city_pop               int64  
 17  job                    object 
 18  dob                    object 
 19  trans_num              object 
 20  unix_time              int64  
 21  merch_lat              float64
 22  merch_long             float64
 23  is_fraud               int64  
dtypes: float64(5), int64(7), object(12)
memory usage: 339.2+ MB
```

```
data.duplicated().sum() #for checking duplicate values
```

```
0
```

```
data.isnull().sum() # checking for null values
```

```
index                    0
Unnamed: 0               0
trans_date_trans_time    0
cc_num                   0
merchant                 0
category                 0
amt                      0
first                    0
last                     0
gender                   0
street                   0
city                     0
state                    0
zip                      0
lat                      0
long                     0
city_pop                 0
job                      0
dob                      0
trans_num                0
unix_time                0
```

```
merch_lat                0
merch_long               0
is_fraud                 0
dtype: int64
```

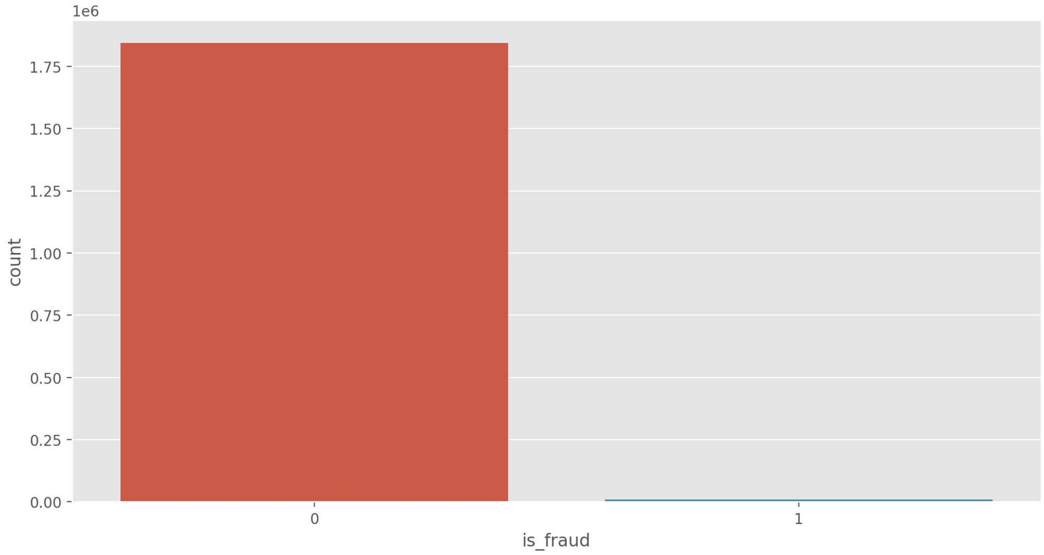
```
data = data.drop(['index', 'Unnamed: 0'], axis = 1)  #for removing columns
```

```
data.describe()
```

	cc_num	amt	zip	lat	long	city_pop	unix_time
count	1.852394e+06	1.852394e+06	1.852394e+06	1.852394e+06	1.852394e+06	1.852394e+06	1.852394e+06
mean	4.173860e+17	7.006357e+01	4.881326e+04	3.853931e+01	-9.022783e+01	8.864367e+04	1.358674e+09
std	1.309115e+18	1.592540e+02	2.688185e+04	5.071470e+00	1.374789e+01	3.014876e+05	1.819508e+07
min	6.041621e+10	1.000000e+00	1.257000e+03	2.002710e+01	-1.656723e+02	2.300000e+01	1.325376e+09
25%	1.800429e+14	9.640000e+00	2.623700e+04	3.466890e+01	-9.679800e+01	7.410000e+02	1.343017e+09
50%	3.521417e+15	4.745000e+01	4.817400e+04	3.935430e+01	-8.747690e+01	2.443000e+03	1.357089e+09
75%	4.642255e+15	8.310000e+01	7.204200e+04	4.194040e+01	-8.015800e+01	2.032800e+04	1.374581e+09
max	4.992346e+18	2.894890e+04	9.992100e+04	6.669330e+01	-6.795030e+01	2.906700e+06	1.388534e+09

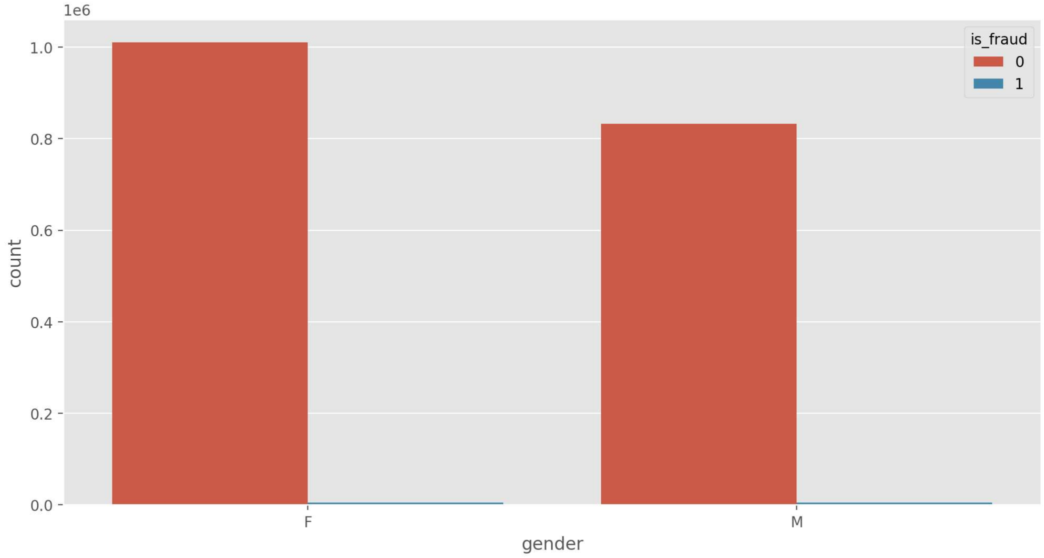
```
plt.figure(figsize = (12,6), dpi = 200)
sns.countplot(x = data['is_fraud'])
```

<Axes: xlabel='is_fraud', ylabel='count'>




```
plt.figure(figsize = (12,6), dpi = 200)
sns.countplot(x = 'gender', hue = 'is_fraud', data = data) #fraud w.r.t gender plot
```

```
<Axes: xlabel='gender', ylabel='count'>
```




```
plt.figure(figsize = (16,8), dpi = 200)
sns.countplot(x = 'category', hue = 'is_fraud', data = data)
plt.xticks(rotation = 60)
plt.show()
```

```
data.head()
```

	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gende
0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	
1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	
2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	I
3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	I
4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	I

5 rows × 22 columns

```
X = data.drop(['is_fraud'], axis = 1) # Creating dependant and independant features dataset
Y = data['is_fraud']
```

```
# the encoding class encodes categorical data into quantifiable values, Eg. : low = 0, med = 1, high = 2
from sklearn.preprocessing import OrdinalEncoder
cols = ['trans_date_trans_time', 'merchant', 'category', 'first', 'last', 'gender', 'street', 'city', 'state', 'job', 'dob', 'trans_
encoder = OrdinalEncoder()
X[cols] = encoder.fit_transform(X[cols])
```

```
# Scaling means adjusting the values to similar proportional range and preventing some features from dominating others due to their
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
```

```
Y = data[['is_fraud']].values
```

```
print('Independant Feautres Shape: ', X.shape)
print('Dependant Feautres Shape: ', Y.shape)
```

```
Independant Feautres Shape:  (1852394, 21)
Dependant Feautres Shape:  (1852394, 1)
```

```
data['is_fraud'].value_counts() # Resampling as dataset is highly unbalanced (gives count of unique values)
```

```
0    1842743
1       9651
Name: is_fraud, dtype: int64
```

```
#Using Oversampling using SMOTE( Synthetic Minority Oversampling Technique )
from imblearn.over_sampling import SMOTE
smote_sampler = SMOTE()
x_sampled, y_sampled = smote_sampler.fit_resample(X, Y)

print('Data : ', x_sampled.shape)
print('Labels : ', y_sampled.shape)
```

```
Data :  (3685486, 21)
Labels :  (3685486,)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_sampled, y_sampled, test_size = 0.2, random_state = 2)

print('Training Data Shape   : ', x_train.shape)
print('Training Labels Shape : ', y_train.shape)
```

```
print('Testing Data Shape    : ', x_test.shape)
print('Testing Labels Shape  : ', y_test.shape)
```

```
Training Data Shape   :  (2948388, 21)
Training Labels Shape :  (2948388,)
Testing Data Shape    :  (737098, 21)
Testing Labels Shape  :  (737098,)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
lr_classifier = LogisticRegression()
lr_classifier.fit(x_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
pred_train = lr_classifier.predict(x_train)
pred_test = lr_classifier.predict(x_test)

print('Training Accuracy: ', accuracy_score(y_train, pred_train)) #computes how many predictions are correct in pred_train w.r.t y_train
print('Testing Accuracy: ', accuracy_score(y_test, pred_test)) #computes how many predictions are correct in pred_test w.r.t y_test
```

```
Training Accuracy:  0.8753339112762635
Testing Accuracy:  0.8755959180461756
```

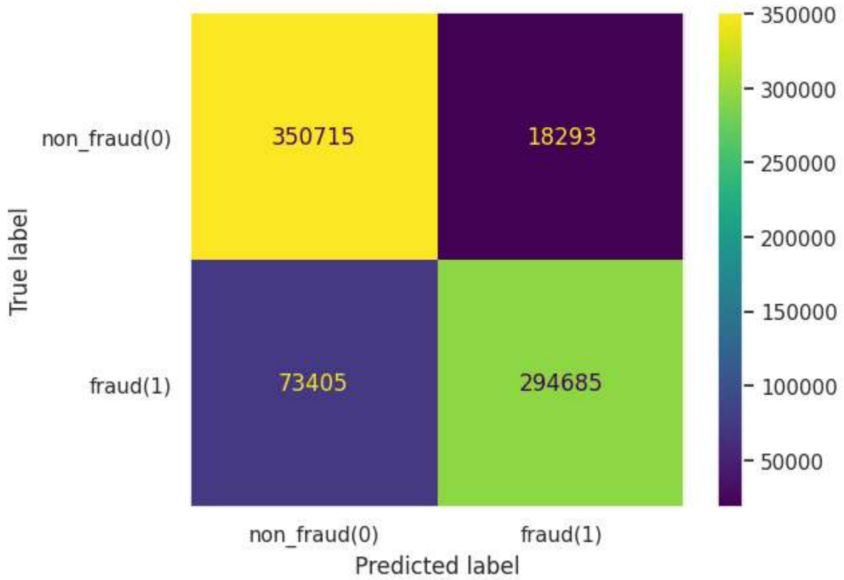
```
print('Training Set f1 score: ', f1_score(y_train, pred_train))
print('Testing Set f1 score: ', f1_score(y_test, pred_test))

print('Test Set precision: ', precision_score(y_test, pred_test))
print('Test Set recall: ', recall_score(y_test, pred_test))
```

```
Training Set f1 score:  0.8652351942181848
Testing Set f1 score:  0.8653614617042645
Test Set precision:  0.9415518023631054
Test Set recall:  0.8005786628270259
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(y_test, pred_test)
display_labels = ["non_fraud(0)", "fraud(1)"]
```

```
plt.figure(figsize = (6,3), dpi = 100)
sns.set(rc = {'axes.grid' : False}) #for turning off grid lines in the plot
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = display_labels)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7dd828c9ada0>
<Figure size 600x300 with 0 Axes>
```



Decision Tree Classification

```
from sklearn.tree import DecisionTreeClassifier
dt_classifier = DecisionTreeClassifier(max_depth = 50, random_state = 100)
dt_classifier.fit(x_train, y_train)
```

```
▾              DecisionTreeClassifier
DecisionTreeClassifier(max_depth=50, random_state=100)
```

```
pred_train = dt_classifier.predict(x_train)
pred_test = dt_classifier.predict(x_test)

print('Training Accuracy: ', accuracy_score(y_train, pred_train))
print('Testing Accuracy: ', accuracy_score(y_test, pred_test))
```

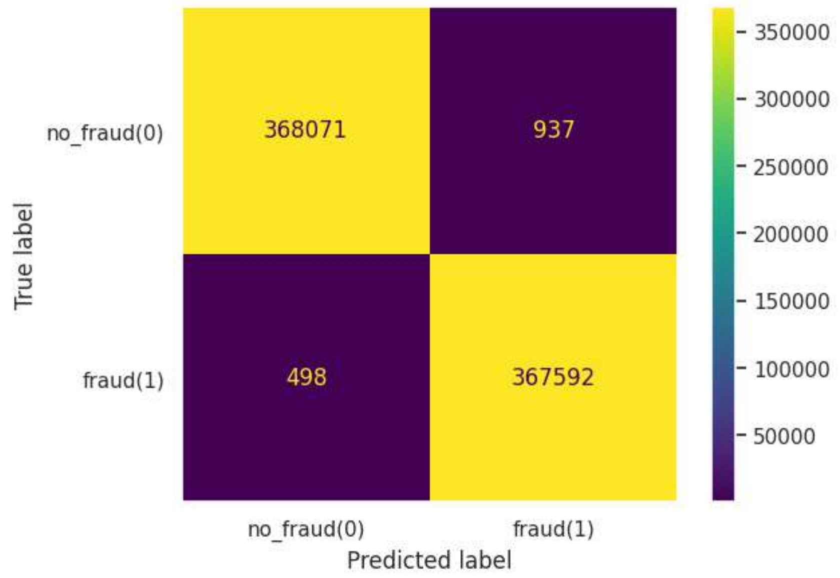
```
Training Accuracy:  0.9999124945563475
Testing Accuracy:  0.9980531761041272
```

```
print('Training f1 score: ', f1_score(y_train, pred_train))
print('Testing f1 score: ', f1_score(y_test, pred_test))

print('Test Set precision: ', precision_score(y_test, pred_test))
print('Test Set recall: ', recall_score(y_test, pred_test))
```

```
Training f1 score:  0.9999125286739822
Testing f1 score:  0.9980519101462222
Test Set precision:  0.9974574592501539
Test Set recall:  0.9986470700100519
```

```
cm = confusion_matrix(y_test, pred_test)
display_labels = ['no_fraud(0)', 'fraud(1)']

plt.figure(figsize = (6,3), dpi = 100)
sns.set(rc = {'axes.grid' : False})
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = display_labels)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7dd80b96cc10>
<Figure size 600x300 with 0 Axes>
```



Random Forest Classification

```
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier(n_estimators = 20, random_state = 2)
rf_classifier.fit(x_train, y_train)
```

```
▾              RandomForestClassifier
RandomForestClassifier(n_estimators=20, random_state=2)
```

```
pred_train = rf_classifier.predict(x_train)
pred_test = rf_classifier.predict(x_test)
```

```
print('Training Accuracy: ', accuracy_score(y_train, pred_train))
print('Testing Accuracy: ', accuracy_score(y_test, pred_test))
```

```
Training Accuracy:  0.9999935558006613
Testing Accuracy:  0.9990191263576892
```

```
print('Training f1 score: ', f1_score(y_train, pred_train))
print('Testing f1 score: ', f1_score(y_test, pred_test))

print('Test Set Precision: ', precision_score(y_test, pred_test))
print('Test Set recall: ', recall_score(y_test, pred_test))
```

```
Training f1 score:  0.999993557808665
Testing f1 score:  0.9990184938564232
Test Set Precision:  0.998418052114736
Test Set recall:  0.9996196582357576
```

```
cm = confusion_matrix(y_test, pred_test)
display_labels = ['no_fraud(0)', 'fraud(1)']

plt.figure(figsize = (6,3), dpi = 100)
sns.set(rc = {'axes.grid' : False})
disp = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = display_labels)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7dd828d9ddb0>
<Figure size 600x300 with 0 Axes>
```