Fraud_Detection

Importing necessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.tree
import DecisionTree
from sklearn.metrics import precision_score,recall_score
from sklearn.metrics import f1_score,roc_auc_score,roc_curve

df=pd.read_csv("creditcard.csv")
```

```
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
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	0.128539	-0.189115	0.133558	-0.021053	149.62	0
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	0.167170	0.125980	-0.008983	0.014724	2.69	0
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	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
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	0.647376	-0.221929	0.062723	0.061458	123.50	0
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	-0.206010	0.502292	0.219422	0.215153	69.99	0

5 rows × 31 columns

```
df.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
dtype: float64
memory usage: 67.4 MB
```

Checking null values

```
df.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
```

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

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

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

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

```
(284315, 31)
(492, 31)
```

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

Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount
0	94838.202258	0.008258	-0.006771	0.012171	-0.007860	0.005453	0.002419	0.009637	-0.000987	0.004467	...	0.000844	-0.001225	-0.000548	0.000226	-0.000305	0.000744	0.000548	-0.000093	-0.000269	88.291022

1 rows × 30 columns

Prediction with unbalanced data

```
target=['Class']
X=df.loc[:,df.columns!=target]
Y=df.loc[:,df.columns==target]
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.33,random_state=1)
# X_train,X_test,Y_train,Y_test=train_test
# X_train_predict,predict_test
```

```
logisticgrid=RegressionC
log_fit=LogisticRegression()
log_fit.fit(X_train,Y_train)
```

```
C:\Users\Pritt\ jupyter\profile\lib\site-packages\sklearn\utilities\validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Pritt\ jupyter\profile\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
LogisticRegression()
```

```
Y_predtrain=log_fit.fit(X_train)
```

```
accuracy_score(Y_test,Y_pred)
```

```
0.9994511413108967
```

```
precision_score(Y_test,Y_pred)
```

```
0.8648782166094842
```

```
recall_score(Y_test,Y_pred)
```

```
0.6043307086614173
```

```
f1_score(Y_test,Y_pred)
```

```
0.6944444444444445
```

Undersampling

```
target='Class'
X=df.loc[:,df.columns!=target]
Y=df.loc[:,df.columns==target]
```

```
minority_class_indices=fraud.index
print(minority_class_len)
```

```
492
```

```
majority_class_indices=legit.index
print(majority_class_indices)
```

```
Int64Index([     0,      1,      2,      3,      4,      5,      6,      7,
            ...
            284306, 284307, 284308, 284309, 284310, 284311, 284312, 284313,
            284314, 284315,
            dtype='int64', length=284315)
```

```
random_majority_indices=np.random.choice(majority_class_indices,minority_class_len,replace=False)
print(len(random_majority_indices))
```

```
492
```

```
minority_class_indices=fraud.index
print(minority_class_indices)
```

```
Int64Index([ 541,  623,  4920,  6108,  6329,  6331,  6334,  6335,
            6337,  6338,
            ...
            279863, 280143, 280149, 280635, 281144, 281674, 281674, 281674,
            281674, 281674,
            dtype='int64', length=492)
```

```
under_sample_indices=np.concatenate([minority_class_indices,random_majority_indices])
```

```
under_sample=df.loc[under_sample_indices]
```

```
sns.countplot(x=target,data=under_sample)
plt.show()
```

```
X_under_sample=under_sample.loc[:,under_sample.columns!=target]
Y_under_sample=under_sample.loc[:,under_sample.columns==target]
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.33,random_state=1)
# X_train,X_test,Y_train,Y_test=train_test
# X_train_predict,predict_test
```

```
logisticgrid=RegressionC
log_fit=LogisticRegression()
log_fit.fit(X_train,Y_train)
```

```
C:\Users\Pritt\ jupyter\profile\lib\site-packages\sklearn\utilities\validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
LogisticRegression()
```

```
Y_predtrain=log_fit.fit(X_train)
```

```
accuracy_score(Y_test,Y_pred)
```

```
0.9390114114114114
```

```
precision_score(Y_test,Y_pred)
```

```
0.9747899159663866
```

```
recall_score(Y_test,Y_pred)
```

```
0.9016666666666667
```

```
f1_score(Y_test,Y_pred)
```

```
0.9367965367965367
```

after undersampling model prediction

Decision tree classifier

```
from sklearn.tree import DecisionTreeClassifier
dtmodelcreate=DecisionTreeClassifier()
dt.fit(X_train,Y_train)
```

```
DecisionTreeClassifier()
```

```
Y_predtrain=dt.fit(X_train)
```

```
accuracy_score(Y_test,Y_pred)
```

```
0.8922967207967207
```

```
precision_score(Y_test,Y_pred)
```

```
0.8888888888888888
```

```
recall_score(Y_test,Y_pred)
```

```
0.8866666666666667
```

```
f1_score(Y_test,Y_pred)
```

```
0.8866666666666667
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
rfmodelcreate=RandomForestClassifier()
rf.fit(X_train,Y_train)
```

```
C:\Users\Pritt\ jupyter\profile\lib\site-packages\sklearn\ensemble\_forest.py:245: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  model = cloned_estimator.fit(X, y, sample_weight=fit_idx[i])
RandomForestClassifier()
```

```
Y_predtrain=rf.fit(X_train)
```

```
accuracy_score(Y_test,Y_pred)
```

```
0.9394922094922094
```

```
precision_score(Y_test,Y_pred)
```

```
0.9666666666666667
```

```
recall_score(Y_test,Y_pred)
```

```
0.9066666666666667
```

```
f1_score(Y_test,Y_pred)
```

```
0.9356666666666667
```

```
Y_pred=rf.predict(X_test)
```

```
Y_test=Y_test.reset_index(drop=True)
Y_test.head()
```

	Class
0	0
1	0
2	0
3	0
4	0

```
Y_pred=Y_pred.reset_index(drop=True)
Y_pred.head()
```

	0
0	0
1	0
2	0
3	0
4	0

```
fraud_detect=pd.DataFrame({'Model':['LR','DT','RF'], 'ACC':[accuracy_score(Y_test,Y_pred)*100,
                                                            accuracy_score(Y_test,Y_pred)*100,
                                                            accuracy_score(Y_test,Y_pred)*100]})
```

```
fraud_detect
```

	Model	ACC
0	LR	93.939114
1	DT	93.939114
2	RF	93.939114

```
sns.barplot(fraud_detect['Model'],fraud_detect['ACC'])
```

```
C:\Users\Pritt\ jupyter\profile\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
<AxesSubplot:xlabel='Model', ylabel'ACC'>
```

Here LogisticRegression turned out to be the best method

Oversampling

```
target='Class'
X=df.loc[:,df.columns!=target]
Y=df.loc[:,df.columns==target]
```

```
Y.shape
```

```
(284807, 30)
```

```
Y.shape
```

```
(284807, 1)
```

```
pip install imblearn
```

```
Requirement already satisfied: imblearn in c:\users\pritt\ jupyter\profile\lib\site-packages (0.0)
Requirement already satisfied: imbalanced-learn in c:\users\pritt\ jupyter\profile\lib\site-packages (from imblearn) (0.10.1)
Requirement already satisfied: numpy>=1.17.3 in c:\users\pritt\ jupyter\profile\lib\site-packages (from imbalanced-learn->imblearn) (1.21.5)
Requirement already satisfied: scipy>=1.3.2 in c:\users\pritt\ jupyter\profile\lib\site-packages (from imbalanced-learn->imblearn) (1.7.3)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\pritt\ jupyter\profile\lib\site-packages (from imbalanced-learn->imblearn) (1.0)
Requirement already satisfied: joblib>=1.1.1 in c:\users\pritt\ jupyter\profile\lib\site-packages (from imbalanced-learn->imblearn) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\pritt\ jupyter\profile\lib\site-packages (from imbalanced-learn->imblearn) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

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

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

```
y_res.shape
```

```
(568630, 30)
```

```
y_res.shape
```

```
(568630, 1)
```

```
y_res.value_counts()
```

```
Class
0    284315
1    284315
dtype: int64
```

```
X_train,X_test,Y_train,Y_test=train_test_split(X_res,y_res,test_size=0.33,random_state=1)
# X_train,X_test,Y_train,Y_test=train_test
# X_train_predict,predict_test
```

```
logisticgrid=RegressionC
log_fit=LogisticRegression()
log_fit.fit(X_train,Y_train)
```

```
C:\Users\Pritt\ jupyter\profile\lib\site-packages\sklearn\utilities\validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
LogisticRegression()
```

```
Y_predtrain=log_fit.fit(X_train)
```

```
accuracy_score(Y_test,Y_pred)
```

```
0.9723819267826744
```

```
precision_score(Y_test,Y_pred)
```

```
0.9705882352941176
```

```
recall_score(Y_test,Y_pred)
```

```
0.9741666666666666
```

```
f1_score(Y_test,Y_pred)
```

```
0.9723529411764705
```

From this after oversampling our accuracy is increased in detecting credit card fraud detection

Decision tree classifier

```
from sklearn.tree import DecisionTreeClassifier
dtmodelcreate=DecisionTreeClassifier()
dt.fit(X_train,Y_train)
```

```
DecisionTreeClassifier()
```

```
Y_predtrain=dt.fit(X_train)
```

```
accuracy_score(Y_test,Y_pred)
```

```
0.9994666666666666
```

```
precision_score(Y_test,Y_pred)
```

```
0.9994666666666666
```

```
recall_score(Y_test,Y_pred)
```

```
0.9994666666666666
```

```
f1_score(Y_test,Y_pred)
```

```
0.9994666666666666
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
rfmodelcreate=RandomForestClassifier()
rf.fit(X_train,Y_train)
```

```
C:\Users\Pritt\ jupyter\profile\lib\site-packages\sklearn\ensemble\_forest.py:245: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  model = cloned_estimator.fit(X, y, sample_weight=fit_idx[i])
RandomForestClassifier()
```

```
Y_predtrain=rf.fit(X_train)
```

```
accuracy_score(Y_test,Y_pred)
```

```
0.9999947368421053
```

```
precision_score(Y_test,Y_pred)
```

```
0.9999947368421053
```

```
recall_score(Y_test,Y_pred)
```

```
0.9999947368421053
```

```
f1_score(Y_test,Y_pred)
```

```
0.9999947368421053
```

```
fraud_detect=pd.DataFrame({'Model':['LR','DT','RF'], 'ACC':[accuracy_score(Y_test,Y_pred)*100,
                                                            accuracy_score(Y_test,Y_pred)*100,
                                                            accuracy_score(Y_test,Y_pred)*100]})
```

```
fraud_detect
```

	Model	ACC
0	LR	97.238267
1	DT	99.946667
2	RF	99.999474

```
sns.barplot(fraud_detect['Model'],fraud_detect['ACC'])
```

```
C:\Users\Pritt\ jupyter\profile\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
<AxesSubplot:xlabel='Model', ylabel'ACC'>
```

Save the model

```
rfmodelcreate=RandomForestClassifier()
rfmodelcreate.fit(X,Y)
```

```
C:\Users\Pritt\ jupyter\profile\lib\site-packages\sklearn\ensemble\_forest.py:2:: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  model = cloned_estimator.fit(X, y, sample_weight=fit_idx[i])
RandomForestClassifier()
```

```
import joblib
```

```
joblib.dump(rfmodelcreate,"credit_card_model")
```

```
['credit_card_model']
```

```
joblib.dump(rfmodelcreate,"credit_card_model")
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       
```