# 0.) Import the Credit Card Fraud Data From CCLE

```
import pandas as pd
from google.colab import drive
import matplotlib.pyplot as plt
import numpy as np
```

```
drive.mount('/content/gdrive/', force_remount = True)
```

```
Mounted at /content/gdrive/
```

```
df = pd.read_csv("/content/gdrive/MyDrive/Econ441B/fraudTest.csv")
```

```
df.head()
```

| | Unnamed: 0 | trans_date_trans_time | cc_num | merchant | category | a |
|---|---|---|---|---|---|---|
| **0** | 0 | 2020-06-21 12:14:25 | 2291163933867244 | fraud_Kirlin and Sons | personal_care | 2. |
| **1** | 1 | 2020-06-21 12:14:33 | 3573030041201292 | fraud_Sporer-Keebler | personal_care | 29. |
| **2** | 2 | 2020-06-21 12:14:53 | 3598215285024754 | fraud_Swaniawski, Nitzsche and Welch | health_fitness | 41. |
| **3** | 3 | 2020-06-21 12:15:15 | 3591919803438423 | fraud_Haley Group | misc_pos | 60. |
| **4** | 4 | 2020-06-21 12:15:17 | 3526826139003047 | fraud_Johnston-Casper | travel | 3. |

5 rows × 23 columns

```
df['dob'] = pd.to_datetime(df['dob'])
df['year_birth'] = df['dob'].dt.year
```

```
df['customer_classification'] = df['year_birth'].copy()
# ya = 0, midd = 1, eld = 2
df.loc[df['customer_classification'] < 1960, 'customer_classification'] = 2
df.loc[(df['customer_classification'] <= 1985) & (df['customer_classification']>= 1960), 'cus
df.loc[(df['customer_classification'] <= 2005) & (df['customer_classification']>= 1986), 'cus
df.loc[df['customer_classification'] == 2, 'customer_classification'] = 'elderly'
df.loc[df['customer_classification'] == 1, 'customer_classification'] = 'middle_aged'
df.loc[df['customer_classification'] == 0, 'customer_classification'] = 'young_adult'
```

```
df_select = df[["trans_date_trans_time", "customer_classification", "amt", "city_pop", "is_fr
df_select.columns
```

```
Index(['trans_date_trans_time', 'customer_classification', 'amt', 'city_pop',
       'is_fraud'],
      dtype='object')
```

```
df_select["trans_date_trans_time"] = pd.to_datetime(df_select["trans_date_trans_time"])
df_select["time_var"] = [i.hour for i in df_select["trans_date_trans_time"]]
```

```
<ipython-input-7-2b0517f78e66>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  df_select["trans_date_trans_time"] = pd.to_datetime(df_select["trans_date_trans_time"]
<ipython-input-7-2b0517f78e66>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  df_select["time_var"] = [i.hour for i in df_select["trans_date_trans_time"]]
```

```
X = pd.get_dummies(df_select, ["customer_classification"]).drop(["trans_date_trans_time", "is
y = df["is_fraud"]
```

```
X.head()
```

| | amt | city_pop | time_var | customer_classification_elderly | customer_classification_m |
|---|---|---|---|---|---|
| **0** | 2.86 | 333497 | 12 | 0 | |
| **1** | 29.84 | 302 | 12 | 0 | |
| **2** | 41.28 | 34496 | 12 | 0 | |
| **3** | 60.05 | 54767 | 12 | 0 | |
| **4** | 3.19 | 1126 | 12 | 1 | |

```
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: is_fraud, dtype: int64
```

# 1.) Use scikit learn preprocessing to split the data into 70/30 in out of sample

```
from sklearn.model_selection import train_test_split


X_train, X_test, y_train,y_test = train_test_split(X,y, test_size = .3)
```

# 2.) Make three sets of training data (Oversample, Undersample and SMOTE)

```
import imblearn
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTE


ros = RandomOverSampler(random_state=0)
over_X, over_y = ros.fit_resample(X,y)


rus = RandomUnderSampler(random_state=0)
under_X, under_y = rus.fit_resample(X,y)


oversample = SMOTE()
smote_x, smote_y = oversample.fit_resample(X,y)


'''import imblearn
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTE
ros = RandomOverSampler(random_state=0)
over_X, over_y = ros.fit_resample(X,y)
rus = RandomUnderSampler(random_state=0)
```

```
under_X, under_y = rus.fit_resample(X,y)
oversample = SMOTE()
smote_x, smote_y = oversample.fit_resample(X,y)'''
```

## 3.) Train three logistic regression models

```
from sklearn.linear_model import LogisticRegression


clf_over = LogisticRegression()
over_log = clf_over.fit(over_X, over_y)


clf_under = LogisticRegression()
under_log = clf_under.fit(under_X, under_y)


clf_smote = LogisticRegression()
smote_log = clf_smote.fit(smote_x, smote_y)
```

## 4.) Test the three models

```
over_y_pred = over_log.predict(X_test)


under_y_pred = under_log.predict(X_test)


smote_y_pred = smote_log.predict(X_test)
```

## 5.) Which performed best in Out of Sample metrics?

SMOTE outperformed the other models based on accuracy, precision, and F1-score. Based on these three out of sample metrics, I believe that SMOTE performed the best.

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


over_acc = accuracy_score(y_test, over_y_pred)
over_prec = precision_score(y_test, over_y_pred)
over_rec = recall_score(y_test, over_y_pred)
over_f1 = f1_score(y_test, over_y_pred)
```

```
print('Over:\n')
print("Accuracy: ", over_acc)
print("Precision: ", over_prec)
print("Recall: ", over_rec)
print("F1-score: ", over_f1)
```

```
Over:

Accuracy:  0.810300151155258
Precision:  0.015387983611171926
Recall:  0.7723704866562009
F1-score:  0.03017479300827967
```

```
under_acc = accuracy_score(y_test, under_y_pred)
under_prec = precision_score(y_test, under_y_pred)
under_rec = recall_score(y_test, under_y_pred)
under_f1 = f1_score(y_test, under_y_pred)
print('Under:\n')
print("Accuracy: ", under_acc)
print("Precision: ", under_prec)
print("Recall: ", under_rec)
print("F1-score: ", under_f1)
```

```
Under:

Accuracy:  0.8127294320881019
Precision:  0.015493314745580128
Recall:  0.7676609105180534
F1-score:  0.03037361408739402
```

```
smote_acc = accuracy_score(y_test, smote_y_pred)
smote_prec = precision_score(y_test, smote_y_pred)
smote_rec = recall_score(y_test, smote_y_pred)
smote_f1 = f1_score(y_test, smote_y_pred)
print('SMOTE:\n')
print("Accuracy: ", smote_acc)
print("Precision: ", smote_prec)
print("Recall: ", smote_rec)
print("F1-score: ", smote_f1)
```

```
SMOTE:

Accuracy:  0.8211089517502819
Precision:  0.01608250157509036
Recall:  0.7613814756671899
F1-score:  0.031499642787556015
```

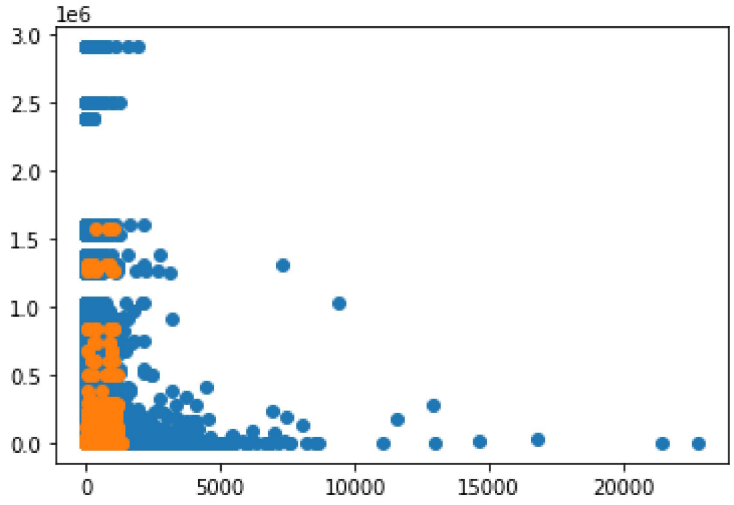
# 6.) Pick two features and plot the two classes before and after SMOTE.

```
import matplotlib.pyplot as plt
```

```
OG_data_temp = pd.concat([X_train, y_train], axis=1)
```

Before SMOTE

```
# Orange
plt.scatter(OG_data_temp[OG_data_temp['is_fraud'] == 0]['amt'],OG_data_temp[OG_data_temp['is_
# Blue
plt.scatter(OG_data_temp[OG_data_temp['is_fraud'] == 1]['amt'],OG_data_temp[OG_data_temp['is_
plt.show()
```




After SMOTE

```
smote_y_df = pd.DataFrame(smote_y)
```

```
new_data_temp = pd.concat([smote_x, smote_y_df], axis=1)
```

```
# Orange
plt.scatter(new_data_temp[new_data_temp['is_fraud'] == 0]['amt'],new_data_temp[new_data_temp[
# Blue
plt.scatter(new_data_temp[new_data_temp['is_fraud'] == 1]['amt'],new_data_temp[new_data_temp[
plt.show()
```