0.) Import the Credit Card Fraud Data From CCLE

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
df = pd.read_csv("fraudTest.csv")
```

In [3]:
```
df_select = df[["trans_date_trans_time", "category", "amt", "city_pop", "is_fraud"]]

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

X = pd.get_dummies(df_select, ["category"]).drop(["trans_date_trans_time", "is_fraud"], axis = 1)
y = df["is_fraud"]
```

```
/var/folders/rf/r80tr2w97dq_m411k8p1924w0000gn/T/ipykernel_37597/2282180580.py:3: 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_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_select["trans_date_trans_time"] = pd.to_datetime(df_select["trans_date_trans_time"])
/var/folders/rf/r80tr2w97dq_m411k8p1924w0000gn/T/ipykernel_37597/2282180580.py:4: 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_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df_select["time_var"] = [i.second for i in df_select["trans_date_trans_time"]]
```

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

In [4]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3)
```

In [6]:
```
X_test, X_holdout, y_test, y_holdout = train_test_split(X_test, y_test, test_size = .5)
```

In [7]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_holdout = scaler.transform(X_holdout)
```

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

In [8]:
```
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTE
```

In [32]:
```
ros = RandomOverSampler()
over_X, over_y = ros.fit_resample(X_train, y_train)

rus = RandomUnderSampler()
under_X, under_y = rus.fit_resample(X_train, y_train)

smote = SMOTE()
smote_X, smote_y = smote.fit_resample(X_train, y_train)
```

3.) Train three logistic regression models

In [10]:
```
from sklearn.linear_model import LogisticRegression
```

In [11]:
```
over_log = LogisticRegression().fit(over_X, over_y)

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

smote_log = LogisticRegression().fit(smote_X, smote_y)
```

4.) Test the three models

In [12]:
```
over_log.score(X_test, y_test)
```

Out[12]:
```
0.9216271983492886
```

In [13]:
```
under_log.score(X_test, y_test)
```
Out[13]:
```
0.926281820581108
```
In [14]:
```
smote_log.score(X_test, y_test)
```
Out[14]:
```
0.9196597807049114
```

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

In [16]:
```
from sklearn.metrics import confusion_matrix
```
In [17]:
```
y_true = y_test
```
In [18]:
```
y_pred = over_log.predict(X_test)
cm = confusion_matrix(y_true, y_pred)
cm
```
Out[18]:
```
array([[76579,  6450],
       [   83,   246]])
```
In [19]:
```
print("Over Sample Sensitivity : ", cm[1,1] /( cm[1,0] + cm[1,1]))
```
```
Over Sample Sensitivity :  0.7477203647416414
```
In [20]:
```
y_pred = under_log.predict(X_test)
cm = confusion_matrix(y_true, y_pred)
cm
```
Out[20]:
```
array([[76966,  6063],
       [   82,   247]])
```
In [21]:
```
print("Under Sample Sensitivity : ", cm[1,1] /( cm[1,0] + cm[1,1]))
```
```
Under Sample Sensitivity :  0.7507598784194529
```
In [22]:
```
y_pred = smote_log.predict(X_test)
cm = confusion_matrix(y_true, y_pred)
cm
```
Out[22]:
```
array([[76415,  6614],
       [   83,   246]])
```

In [23]:
```
print("SMOTE Sample Sensitivity : ", cm[1,1] /( cm[1,0] + cm[1,1]))
```

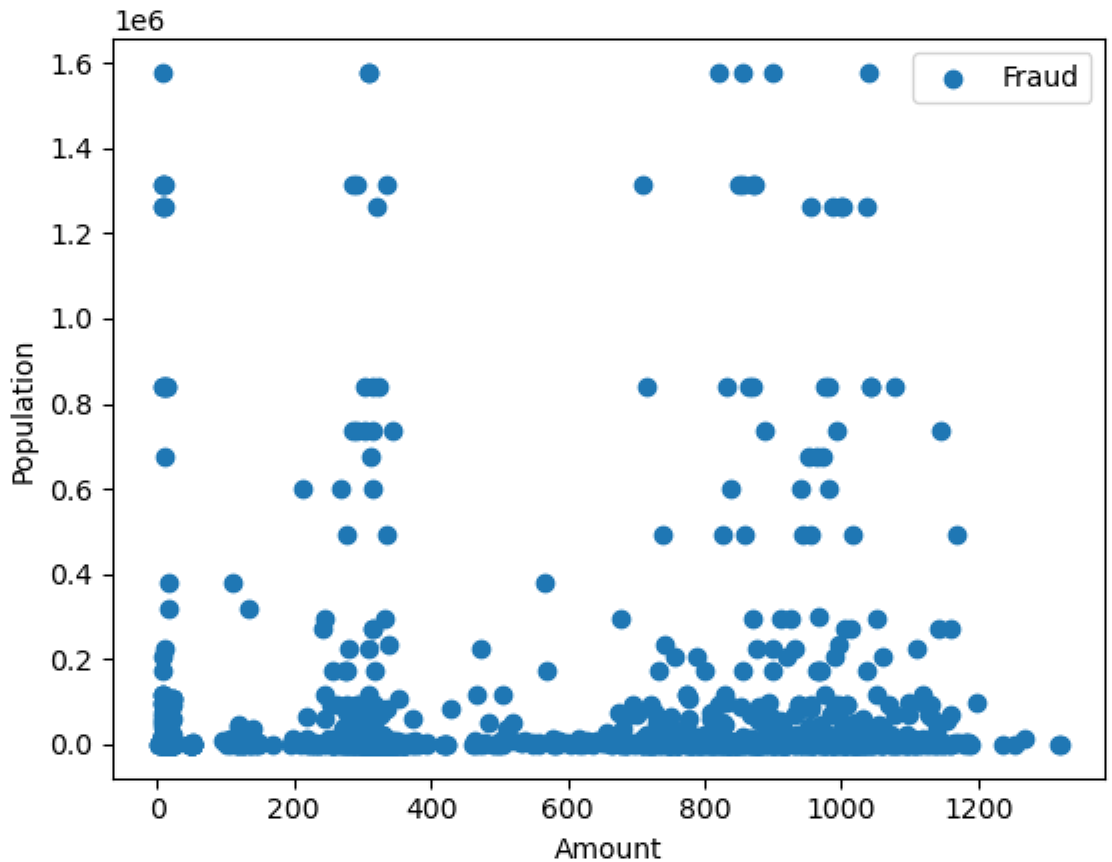
```
SMOTE Sample Sensitivity :  0.7477203647416414
```

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

In [30]:
```
raw_temp = pd.concat([X_train, y_train], axis =1)
```

In [31]:
```
#plt.scatter(raw_temp[raw_temp["is_fraud"] == 0]["amt"], raw_temp[raw_temp["is_fraud"] == 0]["city_pop"])
plt.scatter(raw_temp[raw_temp["is_fraud"] == 1]["amt"], raw_temp[raw_temp["is_fraud"] == 1]["city_pop"])
plt.legend(["Fraud", "Not Fraud"])
plt.xlabel("Amount")
plt.ylabel("Population")

plt.show()
```

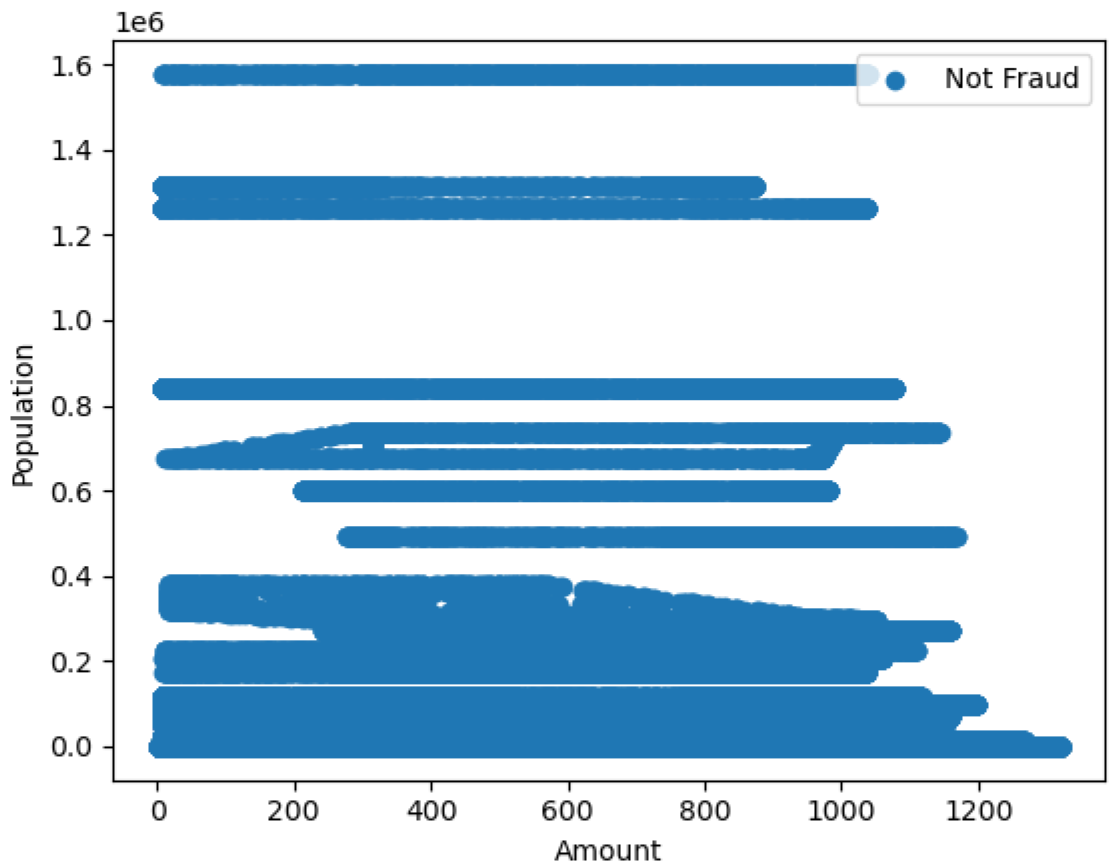


In [33]:
```
raw_temp = pd.concat([smote_X, smote_y], axis =1)
```

In [34]:
```
#plt.scatter(raw_temp[raw_temp["is_fraud"] == 0]["amt"], raw_temp[raw_temp["is_fraud"] == 0]["city_pop"])

plt.scatter(raw_temp[raw_temp["is_fraud"] == 1]["amt"], raw_temp[raw_temp["is_fraud"] == 1]["city_pop"])
plt.legend([ "Not Fraud", "Fraud"])
plt.xlabel("Amount")
plt.ylabel("Population")

plt.show()
```



7.) We want to compare oversampling, Undersampling and SMOTE across our 3 models (Logistic Regression, Logistic Regression Lasso and Decision Trees).

Make a dataframe that has a dual index and 9 Rows.

Calculate: Sensitivity, Specificity, Precision, Recall and F1 score. for out of sample data.

Notice any patterns across perfomance for this model. Does one totally out perform the others IE. over/under/smote or does a model perform better DT, Lasso, LR?

Choose what you think is the best model and why. test on Holdout

In [35]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score
import pandas as pd
```

In [36]:
```
resampling_methods = {
    "over" : RandomOverSampler(),
    "under" : RandomUnderSampler(),
    "smote" : SMOTE()
}

model_configs = {
    "LOG" : LogisticRegression(),
    "LASSO" : LogisticRegression(penalty = "l1", C = 2., solver = "liblinear"),
    "DTREE" : DecisionTreeClassifier()
}
```

In [37]:
```
trained_models = {}
```

In [38]:
```
def calc_perf_metrics(y_true, y_pred):


    tn, fn, fp, tp = confusion_matrix(y_true, y_pred).ravel()
    precision = tp / (tp + fp)
```

In [39]:
```
for resample_key, resampler in resampling_methods.items():
    resample_X, resample_y = resampler.fit_resample(X_train, y_train)

    for model_name, model in model_configs.items():
        combined_key = f"{resample_key}_{model_name}"
        trained_models[combined_key] = model.fit(resample_X, resample_y)


        df = pd.DataFrame()
        df['model'] = [combined_key]
        df['accuracy'] = [trained_models[combined_key].score(X_test, y_test)]
        y_pred = trained_models[combined_key].predict(X_test)
        y_true = y_test



        df['precision'] = [precision_score(y_true, y_pred)]
        df['recall'] = [recall_score(y_true, y_pred)]
        df['f1'] = [f1_score(y_true, y_pred)]


        print(df.to_string(index=False))
        print("-----")
```

```
   model  accuracy  precision   recall        f1
over_LOG  0.818134   0.015288 0.727838 0.029946
-----
     model  accuracy  precision  recall       f1
over_LASSO  0.907177   0.030038 0.73717 0.057724
-----
     model  accuracy  precision   recall     f1
over_DTREE  0.996725   0.580165 0.545879 0.5625
-----
    model  accuracy  precision   recall        f1
under_LOG  0.823382    0.01564 0.723173 0.030617
-----
      model  accuracy  precision  recall       f1
under_LASSO  0.902961   0.028757 0.73717 0.055354
-----
      model  accuracy  precision   recall        f1
under_DTREE  0.947683   0.065224 0.942457 0.122005
-----
    model  accuracy  precision   recall        f1
smote_LOG  0.824582    0.01568 0.720062 0.030692
-----
      model  accuracy  precision   recall        f1
smote_LASSO  0.978562   0.118657 0.709176 0.203299
-----
      model  accuracy  precision   recall        f1
smote_DTREE  0.995993   0.485779 0.664075 0.561104
-----
```

The best model is the one with the highest f1 score : Decision Tree with over sampling. Regardless of the sampling method, the best one is the Decision Tree. And regardless of model ,the best one is smote.