0.) Import the Credit Card Fraud Data From CCLE

In [ ]:
```
import pandas as pd
# from google.colab import drive
import matplotlib.pyplot as plt
import numpy as np
```

In [ ]:
```
# drive.mount('/content/gdrive/', force_remount = True)
```

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

In [ ]:
```
import warnings
warnings.simplefilter(action="ignore", category=FutureWarning)
```

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

In [ ]:
```
df.head()
```

Out[ ]:

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	...	lat	l
0	0	2020-06-21 12:14:25	2291163933867244	fraud_Kirlin and Sons	personal_care	2.86	Jeff	Elliott	M	351 Darlene Green	...	33.9659	-80.9
1	1	2020-06-21 12:14:33	3573030041201292	fraud_Sporer-Keebler	personal_care	29.84	Joanne	Williams	F	3638 Marsh Union	...	40.3207	-110.4
2	2	2020-06-21 12:14:53	3598215285024754	fraud_Swaniawski, Nitzsche and Welch	health_fitness	41.28	Ashley	Lopez	F	9333 Valentine Point	...	40.6729	-73.5
3	3	2020-06-21 12:15:15	3591919803438423	fraud_Haley Group	misc_pos	60.05	Brian	Williams	M	32941 Krystal Mill Apt. 552	...	28.5697	-80.8
4	4	2020-06-21 12:15:17	3526826139003047	fraud_Johnston-Casper	travel	3.19	Nathan	Massey	M	5783 Evan Roads Apt. 465	...	44.2529	-85.0

5 rows × 23 columns

In [ ]:
```
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"]
```

```
C:\Users\thinkpad\AppData\Local\Temp\ipykernel_22496\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
  df_select["trans_date_trans_time"] = pd.to_datetime(df_select["trans_date_trans_time"])
C:\Users\thinkpad\AppData\Local\Temp\ipykernel_22496\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
  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 [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

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

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

In [ ]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)
X_holdout = scaler.transform(X_holdout)
```

```
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:767: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:767: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:767: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:767: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if not hasattr(array, "sparse") and array.dtypes.apply(is_sparse).any():
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

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

In [ ]:

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

In [ ]:

```
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)
```

```
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

3.) Train three logistic regression models

In [ ]:

```
from sklearn.linear_model import LogisticRegression
```

In [ ]:

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

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

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

```
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

4.) Test the three models

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

```
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

Out[ ]:
```
0.9227428681110391
```

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

```
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

Out[ ]:
```
0.9094268096643393
```

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

```
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

Out[ ]:
```
0.9210633652438878
```

In [ ]:
```
# We see SMOTE performing with higher accuracy but is ACCURACY really the best measure?
```

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

In [ ]:
```
# Sensitivity here in credit fraud is more important as seen from last class
```

In [ ]:
```
from sklearn.metrics import confusion_matrix
```

In [ ]:
```
y_true = y_test
```

In [ ]:
```
y_pred = over_log.predict(X_test)
cm = confusion_matrix(y_true, y_pred)
cm
```

```
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

Out[ ]:
```
array([[76692,  6360],
       [   80,   226]], dtype=int64)
```

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

```
Over Sample Sensitivity :  0.738562091503268
```

In [ ]:
```
y_pred = under_log.predict(X_test)
cm = confusion_matrix(y_true, y_pred)
cm
```

```
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

Out[ ]:
```
array([[75582,  7470],
       [   80,   226]], dtype=int64)
```

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

```
Under Sample Sensitivity :  0.738562091503268
```

In [ ]:
```
y_pred = smote_log.predict(X_test)
cm = confusion_matrix(y_true, y_pred)
cm
```

```
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:605: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype):
c:\ProgramData\anaconda3\lib\site-packages\sklearn\utils\validation.py:614: FutureWarning: is_sparse is deprecated and will be r
emoved in a future version. Check `isinstance(dtype, pd.SparseDtype)` instead.
  if is_sparse(pd_dtype) or not is_extension_array_dtype(pd_dtype):
```

Out[ ]:
```
array([[76552,  6500],
       [   80,   226]], dtype=int64)
```

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

```
SMOTE Sample Sensitivity :  0.738562091503268
```

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

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

In [ ]:
```
raw_temp.columns = list(X.columns) + ["is_fraud"]
```

In [ ]:
```
raw_temp
```

Out[ ]:

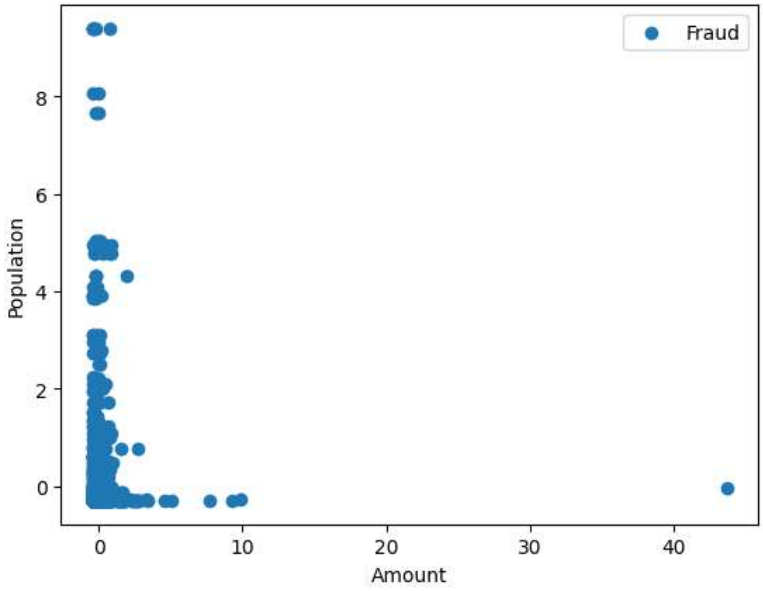
	amt	city_pop	time_var	category_entertainment	category_food_dining	category_gas_transport	category_grocery_net	category_groce
0	0.061851	-0.292731	-0.432628	-0.279008	-0.275853	-0.335807	-0.190082	-0.3
1	-0.269132	-0.269654	0.260668	-0.279008	-0.275853	-0.335807	-0.190082	-0.3
2	-0.198467	-0.291611	1.589484	-0.279008	-0.275853	-0.335807	-0.190082	-0.3
3	-0.416047	-0.263393	-1.357022	-0.279008	-0.275853	-0.335807	-0.190082	-0.3
4	-0.242460	-0.288789	-0.952599	3.584130	-0.275853	-0.335807	-0.190082	-0.3
...	...	...	...	...	...	...	...	
469263	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
452506	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
502059	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
441293	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
521559	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

505744 rows × 18 columns

In [ ]:
```
# 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 [ ]:
```
# raw_temp = pd.concat([smote_X, smote_y], axis =1)
raw_temp = pd.concat([pd.DataFrame(raw_temp, columns = X.columns), pd.DataFrame(smote_y, columns = ["is_fraud"])], axis = 1)
```

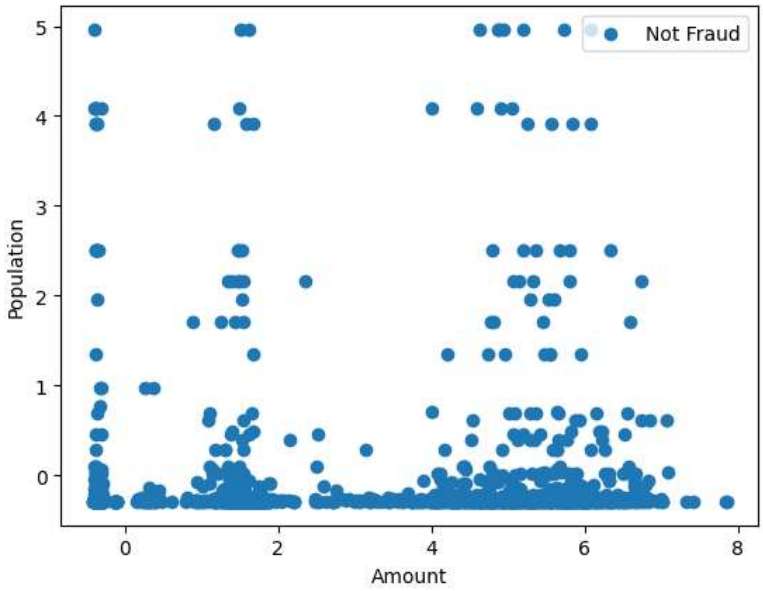
In [ ]:
```
raw_temp.columns = list(X.columns) + ["is_fraud"]
```

In [ ]:
```
#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()
```

```
c:\ProgramData\anaconda3\lib\site-packages\IPython\core\pylabtools.py:152: UserWarning: Creating legend with loc="best" can be s
low with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



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.

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

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

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

model_configs = {
    "LOG" : LogisticRegression(),
    "LASSO" : LogisticRegression(penalty = "l1", C = .5, solver = "liblinear"), # C is the inverse of the regularization strengt
    "DicisionTree" : DecisionTreeClassifier()
}
```

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

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


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

In [ ]:
```
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)

        # turn the performance metrics of different models into a dataframe
        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

        # tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
        # df['precision'] = tp / (tp + fp)
        # df['recall'] = tp / (tp + fn)
        # df['f1'] = 2 * (df['precision'] * df['recall']) / (df['precision'] + df['recall'])

        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, index = False)
        # print the dataframe without index
        print(df.to_string(index=False))
        print("-----")
```

```
    model  accuracy  precision  recall      f1
over_LOG  0.922251   0.034103  0.738562  0.065195
-----
     model  accuracy  precision  recall      f1
over_LASSO  0.922263   0.034108  0.738562  0.065205
-----
              model  accuracy  precision  recall      f1
over_DicisionTree  0.997049   0.600671  0.584967  0.592715
-----
    model  accuracy  precision  recall      f1
under_LOG  0.908815   0.029173  0.738562  0.056128
-----
      model  accuracy  precision  recall      f1
under_LASSO  0.913038   0.030561  0.738562  0.058694
-----
                model  accuracy  precision  recall      f1
under_DicisionTree  0.944109   0.059502  0.960784  0.112064
-----
    model  accuracy  precision  recall      f1
smote_LOG  0.919924   0.033133  0.738562  0.063421
-----
      model  accuracy  precision  recall      f1
smote_LASSO  0.919888   0.033118  0.738562  0.063394
-----
              model  accuracy  precision  recall      f1
smote_DicisionTree  0.991854   0.272838  0.732026  0.397516
-----
```

- The best model is the one with the highest f1 score : Decision Tree with over sampling
- Regardless of the sampling method, the best model is the Decision Tree
- Regardless of the model, the best sampling method is the over sampling