Import library

```
import pandas as pd
import numpy as np
from sklearn.datasets import make_classification
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import seaborn as sn
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
#from sklearn.svm import SVM
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.metrics import roc_auc_score, roc_curve
```

Memanggil dataset

```
ds = pd.read_csv("D:/STATISTIKA/klasifikasi__UKT_komplit.csv")
ds.head(10)
```

```
   No.  StatusOrtu  Penghasilan  Status_Rumah  JMotor  Jmobil  DayaLis
KIPK
0    1           1      4000000             1       1       0        2
0
1    2           1      2500000             0       1       0        3
0
2    3           1      6000000             1       2       0        2
0
3    4           1      5440500             1       2       0        2
0
4    5           1     10000000             0       1       1        3
0
5    6           1      1000000             0       1       0        3
1
6    7           1     20000000             1       2       1        3
0
7    8           1     15000000             1       1       0        3
0
8    9           4      4000000             1       1       1        3
0
9   10           1            0             1       2       0        1
0
```

```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1707 entries, 0 to 1706
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   No.           1707 non-null   int64
 1   StatusOrtu    1707 non-null   int64
 2   Penghasilan   1707 non-null   int64
 3   Status_Rumah  1707 non-null   int64
 4   JMotor        1707 non-null   int64
 5   Jmobil        1707 non-null   int64
 6   DayaLis       1707 non-null   int64
 7   KIPK          1707 non-null   int64
dtypes: int64(8)
memory usage: 106.8 KB
```

```
ds.describe()
```

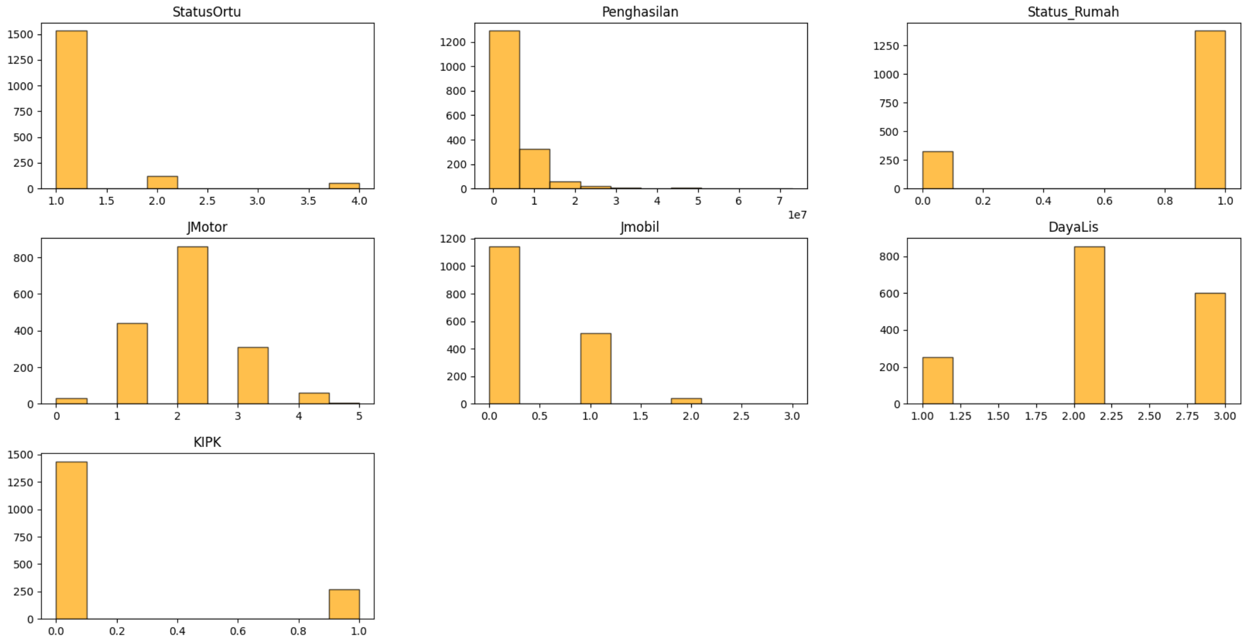
```
               No.   StatusOrtu   Penghasilan  Status_Rumah
JMotor  \
count  1707.000000  1707.000000  1.707000e+03   1707.000000
1707.000000
mean    854.000000     1.164030  5.195012e+06      0.807264
1.968366
std     492.912771     0.565764  5.552922e+06      0.394563
0.823274
min       1.000000     1.000000 -1.000000e+06      0.000000
0.000000
25%     427.500000     1.000000  2.000000e+06      1.000000
1.000000
50%     854.000000     1.000000  4.000000e+06      1.000000
2.000000
75%    1280.500000     1.000000  6.131916e+06      1.000000
2.000000
max    1707.000000     4.000000  7.300000e+07      1.000000
5.000000

            Jmobil      DayaLis         KIPK
count  1707.000000  1707.000000  1707.000000
mean      0.357938     2.205038     0.157586
std       0.541534     0.676276     0.364460
min       0.000000     1.000000     0.000000
25%       0.000000     2.000000     0.000000
50%       0.000000     2.000000     0.000000
75%       1.000000     3.000000     0.000000
max       3.000000     3.000000     1.000000
```

```
del(ds["No."])
ds.describe()
```

```
         StatusOrtu   Penghasilan  Status_Rumah       JMotor
Jmobil  \
count  1707.000000  1.707000e+03   1707.000000  1707.000000
1707.000000
mean      1.164030  5.195012e+06      0.807264     1.968366
0.357938
std       0.565764  5.552922e+06      0.394563     0.823274
0.541534
min       1.000000 -1.000000e+06      0.000000     0.000000
0.000000
25%       1.000000  2.000000e+06      1.000000     1.000000
0.000000
50%       1.000000  4.000000e+06      1.000000     2.000000
0.000000
75%       1.000000  6.131916e+06      1.000000     2.000000
1.000000
max       4.000000  7.300000e+07      1.000000     5.000000
3.000000

          DayaLis         KIPK
count  1707.000000  1707.000000
mean      2.205038     0.157586
std       0.676276     0.364460
min       1.000000     0.000000
25%       2.000000     0.000000
50%       2.000000     0.000000
75%       3.000000     0.000000
max       3.000000     1.000000
```

```
histogram = ds
histogram.hist(figsize=(20,10),alpha = 0.7, color =
'orange',edgecolor ='black',grid=False)
```

```
array([[<Axes: title={'center': 'StatusOrtu'}>,
        <Axes: title={'center': 'Penghasilan'}>,
        <Axes: title={'center': 'Status_Rumah'}>],
       [<Axes: title={'center': 'JMotor'}>,
        <Axes: title={'center': 'Jmobil'}>,
        <Axes: title={'center': 'DayaLis'}>],
       [<Axes: title={'center': 'KIPK'}>, <Axes: >, <Axes: >]],
      dtype=object)
```



Output di atas adalah hasil plot untuk distribusi tiap fitur dataset, Disini didapat 4 fitur merupakan categorical variable dan sisanya berupa numerical variable

```
from sklearn.preprocessing import MinMaxScaler
array = ds.values
x = array[:,1:6] #slicing dataframe kedalam array
y = array[:,6]

scaler = MinMaxScaler()
#transformasi data
x = scaler.fit_transform(x)
x

array([[0.06756757, 1.        , 0.2       , 0.        , 0.5       ],
       [0.0472973 , 0.        , 0.2       , 0.        , 1.        ],
       [0.09459459, 1.        , 0.4       , 0.        , 0.5       ],
       ...,
       [0.02567568, 1.        , 0.6       , 0.33333333, 0.5       ],
       [0.08108108, 0.        , 0.4       , 0.        , 0.        ],
       [0.12162162, 1.        , 0.6       , 0.        , 0.5       ]])
```

Disini saya terapkan normalisasi data yaitu minmaxscaler

```
y = y.astype('int')
y

array([0, 0, 0, ..., 0, 0, 0])

Counter(y)
Counter({0: 1438, 1: 269})
```

Hasil dari target berupa logit biner dengan memanggil Counter(y), maka didapat jumlah tiap target

```
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2,
random_state=2)
# instantiating the random over sampler
ros = RandomOverSampler()
# resampling x_train, y_train
X_ros, y_ros = ros.fit_resample(x_train, y_train)
# new class distribution
print(Counter(y_ros))

Counter({0: 1156, 1: 1156})
```

untuk mengatasi dataset yang tidak seimbang. Disini digunakan function RandomOverSample untuk. towardds the majority class, untuk memprediksi random, Data yang telah di pisah data train dan data test kemudian

```
logmodel = LogisticRegression()

#1 model dengan dataset asli
model1=logmodel.fit(x_train, y_train)
predictions1a = model1.predict(x_train)
predictions1b = model1.predict(x_test)
predictions1c = model1.predict_proba(x_test)[:,1]
print("-------Model-1: Logit Biner dengan Dataset Asli-------")
print("Kinerja Data Training:")
print(classification_report(y_train, predictions1a))
print(confusion_matrix(y_train, predictions1a))
print(accuracy_score(y_train, predictions1a))
print("Kinerja Data Testing:")
print(classification_report(y_test, predictions1b))
print(confusion_matrix(y_test, predictions1b))
print(accuracy_score(y_test, predictions1b))

-------Model-1: Logit Biner dengan Dataset Asli-------
Kinerja Data Training:
              precision    recall  f1-score   support

           0       0.85      0.99      0.92      1156
           1       0.48      0.06      0.11       209

    accuracy                           0.85      1365
   macro avg       0.67      0.53      0.51      1365
```

```
weighted avg       0.80      0.85      0.79      1365

[[1142   14]
 [ 196   13]]
0.8461538461538461
Kinerja Data Testing:
              precision    recall  f1-score   support

           0       0.83      0.98      0.90       282
           1       0.33      0.05      0.09        60

    accuracy                           0.82       342
   macro avg       0.58      0.51      0.49       342
weighted avg       0.74      0.82      0.76       342

[[276   6]
 [ 57   3]]
0.8157894736842105
```

Klasifikasi report dan confusion matrix dari data train dan data test menghasilkan kinerja dari data testing dan data train yang tidak beda jauh, yang menunjukkan bahwa model bekerja dengan sangat baik

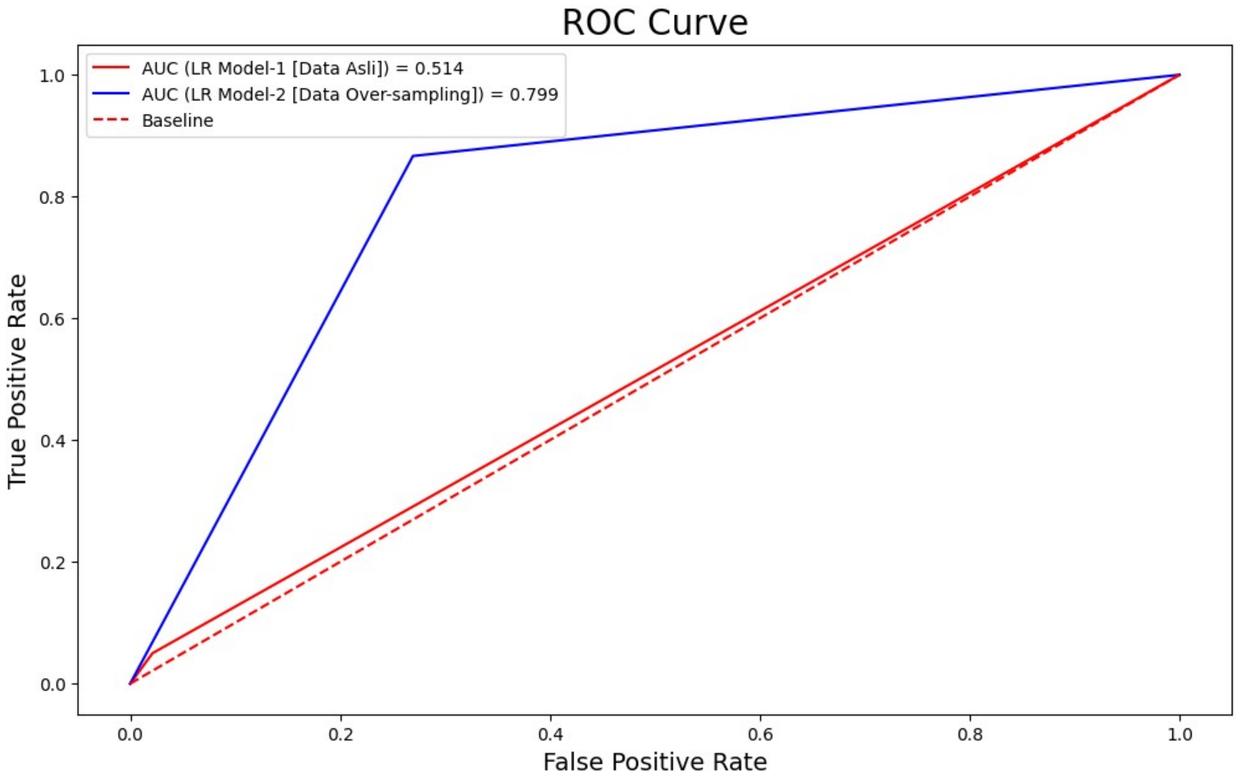
```
model2=logmodel.fit(X_ros, y_ros)
predictions2a = model2.predict(X_ros)
predictions2b = model2.predict(x_test)
predictions2c = model2.predict_proba(x_test)[:,1]
print("-------Model-2: Logit Biner dengan Dataset Over-sampling-------")
print("Kinerja Data Training:")
print(classification_report(y_ros, predictions2a))
print(confusion_matrix(y_ros, predictions2a))
print(accuracy_score(y_ros, predictions2a))
print("Kinerja Data Testing:")
print(classification_report(y_test, predictions2b))
print(confusion_matrix(y_test, predictions2b))
print(accuracy_score(y_test, predictions2b))
```

```
-------Model-2: Logit Biner dengan Dataset Over-sampling-------
Kinerja Data Training:
              precision    recall  f1-score   support

           0       0.82      0.68      0.74      1156
           1       0.73      0.85      0.78      1156

    accuracy                           0.76      2312
   macro avg       0.77      0.76      0.76      2312
weighted avg       0.77      0.76      0.76      2312
```

```
[[787 369]
 [178 978]]
0.7634083044982699
Kinerja Data Testing:
              precision    recall  f1-score   support

           0       0.96      0.73      0.83       282
           1       0.41      0.87      0.55        60

    accuracy                           0.75       342
   macro avg       0.68      0.80      0.69       342
weighted avg       0.87      0.75      0.78       342

[[206  76]
 [  8  52]]
0.7543859649122807
```

Klasifikasi report dan confusion matrix dari data train dan data test yang telah diresampling, menghasilkan kinerja dari data testing dan data train yang tidak beda jauh, yang menunjukkan bahwa model bekerja dengan sangat baik. meski besgitu nilai yang dihasilkan sedikit lebih kecil dari data tanpa diresampling.

```
#y_test_int = y_test.replace({'Good': 1, 'Bad': 0})
auc1 = roc_auc_score(y_test, predictions1b)
fpr1, tpr1, thresholds1 = roc_curve(y_test, predictions1b)
auc2 = roc_auc_score(y_test, predictions2b)
fpr2, tpr2, thresholds2 = roc_curve(y_test, predictions2b)
plt.figure(figsize=(12, 7))
plt.plot(fpr1, tpr1, label=f'AUC (LR Model-1 [Data Asli]) = 
{auc1:.3f}',color='red')
plt.plot(fpr2, tpr2, label=f'AUC (LR Model-2 [Data Over-sampling]) = 
{auc2:.3f}',color='blue')
plt.plot([0, 1], [0, 1], color='red', linestyle='--', 
label='Baseline')
plt.title('ROC Curve', size=20)
plt.xlabel('False Positive Rate', size=14)
plt.ylabel('True Positive Rate', size=14)
plt.legend()

<matplotlib.legend.Legend at 0x1ac5ed80450>
```



Hasil output menampilkan perbandingan hasil data train dan test dengan data resampling dan data asli.