```
#Import Library
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Download Dataset
data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-ML241EN-
SkillsNetwork/labs/datasets/Human_Activity_Recognition_Using_Smartphones_Data.csv")

#########################################################################

#Examine the data type
data.dtypes.value_counts()   #It will show you the count value of the types of data (ex. object, int, float etc.)

#Search the column that the type is Object
s = 0
for i in data.columns:
    if data[i].dtype == 'O' :
        s+=1
        print(i)
print(s)

#Check the min and max value in the data set except the last column due to the type of this column is object.
data.iloc[:,:-1].min().value_counts()
data.iloc[:,:-1].max().value_counts()

#Examine the value of the colume that the type is object
data['Activity'].value_counts()

#Encoder the value into the integer and add in new column name Activity2
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data['Activity2'] = le.fit_transform(data.Activity)

#Compare the encoded and existing colume sort by Activity2
data[['Activity','Activity2']].value_counts(sort=False)

#Create a new dataset with encoded column and drop the existing column which the type is object
data2 = data.drop(columns=['Activity'], inplace=False)
```

```
#############################################################################

#Calculate the correlation between the dependent variable

#Correlation หรือ ค่าสหสัมพันธ์ เป็นการดูทิศทางความสัมพันธ์ระหว่างตัวแปร 2 ตัว
#โดยมี Correlation Coefficient (r) หรือ ค่าสัมประสิทธิ์สหสัมพันธ์ เป็นตัวบ่งชี้ถึงความสัมพันธ์นี้
#ซึ่งค่าสัมประสิทธิ์สหสัมพันธ์นี้จะมีค่าอยู่ระหว่าง -1.0 ถึง +1.0 ซึ่งหากมีค่าใกล้ -1.0 นั้นหมายความว่าตัวแปรทั้งสองตัวมีความสัมพันธ์กันอย่างมากในเชิงตรงกันข้าม
#หากมีค่าใกล้ +1.0 นั้นหมายความว่า ตัวแปรทั้งสองมีความสัมพันธ์กันโดยตรงอย่างมาก และหากมีค่าเป็น 0 นั้นหมายความว่า ตัวแปรทั้งสองตัวไม่มีความสัมพันธ์ต่อกัน

#Feature columns except encoded column
feature_cols = data2.columns[:-1]

#Calculate corr values
corr_values = data2[feature_cols].corr() #return correlation matrix

#Create array of index that is a lower triagonal where the first array is rows and the second array is columns
tril_index = np.tril_indices_from(corr_values)

#Create tuple for the matrix coordinate(ex. (row,col) )
Coord = zip(*tril_index) #return tuples of coordinate (ex. (0,0), (1,0), (1,1) etc.)

#Change the existing lower triagonal value to NaN due to this existing value is unused( ค่าที่ซ้ำกัน และ ค่าที่เกิดจากการเปลี่ยนเทียบของตัวเอง )
for x in Coord:
    corr_values.iloc[x[0], x[1]] = np.NaN

#Stack a dataframe (return all columns that show for each row and also show tha value)
corr_values = corr_values.stack()

#Change to dataframe
corr_values = corr_values.to_frame()

#Reset the index name, it will show name of columnn are 'level_0', 'level_1', 0
corr_values = corr_values.reset_index()

#Rename the column name
corr_values = corr_values.rename(columns={'level_0':'feature1', 'level_1':'feature2', 0:'correlation'})

#Get the absolute values for sorting
corr_values['abs_correlation'] = corr_values.correlation.abs()

#Create a histrogram to see the absolute correlation value
sns.set_context('talk')
sns.set_style('white')

ax = corr_values.abs_correlation.hist(bins=50, figsize=(12, 8))
ax.set(xlabel='Absolute Correlation', ylabel='Frequency')

# save the plot as a PNG file
plt.savefig('histogram.png')

# The most highly correlated values
corr_values.sort_values('correlation', ascending=False).query('abs_correlation>0.8')  # ascending is True mean sort from
min to max and us
### END SOLUTION
```

```
#########################################################################

#Split data into train and test data by using StratifiedShuffleSplit
from sklearn.model_selection import StratifiedShuffleSplit
Split = StratifiedShuffleSplit(n_splits=1, test_size=0.3, random_state=42) # n_split = Number of re-shuffling & splitting
iterations, test_size is portion of test data set

Index = Split.split(data2[feature_cols], data2['Activity2']) # .split(X, y) return index of train and test dataset

for Train_index, Test_index in Index:
    X_train = data2.loc[Train_index, feature_cols]
    y_train = data2.loc[Train_index, 'Activity2']  # .loc[name_row, name_col] but .iloc[number_row, number_col]
    X_test = data2.loc[Test_index, feature_cols]
    y_test = data2.loc[Test_index, 'Activity2']

#########################################################################

#Fit Logistic Regression
from sklearn.linear_model import LogisticRegression

#To choose a solver, you might want to consider the following aspects:
#1.For small datasets, ‘liblinear’ is a good choice, whereas ‘sag’ and ‘saga’ are faster for large ones;
#2.For multiclass problems, only ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ handle multinomial loss;
#3.‘liblinear’ and is limited to one-versus-rest schemes.
#4.‘newton-cholesky’ is a good choice for n_samples >> n_features, especially with one-hot encoded categorical features
with rare categories.
# Note that it is limited to binary classification and the one-versus-rest reduction for multiclass classification.
# Be aware that the memory usage of this solver has a quadratic dependency on n_features because it explicitly computes the
Hessian matrix.

LR = LogisticRegression(solver= 'liblinear').fit(X_train, y_train)  # return equation that fit the X_train and y_train
(y_train = coeff*X_train +inception)

# Using cross validation to determine the hyperparameters and fit models using L1 and L2 regularization.
from sklearn.linear_model import LogisticRegressionCV

# L1 regularized logistic regression
LR_L1 = LogisticRegressionCV(Cs=10, cv=4, penalty='l1', solver='liblinear').fit(X_train, y_train)

# L2 regularized logistic regression
LR_L2 = LogisticRegressionCV(Cs=10, cv=4, penalty='l2', solver='liblinear').fit(X_train, y_train)
```

```
#########################################################################

#Compare the magnitudes of the coefficients for each of the models.

# Combine all the coefficients into a dataframe
coefficients = list()

coeff_labels = ['LR', 'L1', 'L2']
coeff_models = [LR, LR_L1, LR_L2]


for lab,mod in zip(coeff_labels, coeff_models):  # from zip lab is in coeff_labels and mod is is coeff_models
    coeffs = mod.coef_                           # the shape of coeffs is (1, n_features)
    coeff_label = pd.MultiIndex(levels=[[lab], [0,1,2,3,4,5]],
                                codes=[[0,0,0,0,0,0], [0,1,2,3,4,5]]) # levels is index name and codes is ตำแหน่ง ของ index ตย ใน goodnotes
    coefficients.append(pd.DataFrame(coeffs.T, columns=coeff_label)) # coeff.T for chanege the shape from (1, n_features) to (n_features, 1)

#coefficients ที่ได้จะมี 3 dataframe เนื่องจาก loop โดย loop1 is LR, loop2 is LR_L1 and loop3 is LR_L2

coefficients = pd.concat(coefficients, axis=1) # นำข้อมูลของแต่ละ dataframe รวมให้เป็น 1 dataframe โดยที่ไม่แยก index กัน ระหว่าง dataframe axis=0 for index , axis=1 for column ตย ใน goodnotes

# Visualize the dataframe to see relation
fig, axList = plt.subplots(nrows=3, ncols=2) #แสดงกราฟทั้งหมด 6 กราฟ โดยแบ้งเป็น 3 rows 2 columns โดยตำแหน่งจะเก็บใน axList เช่น รูปแถวที่ 1 คอลัมน์ที่ 2 จะได้ว่า ax.List[0, 1]
axList = axList.flatten()                    #เปลี่ยนจาก array 2D เป็น 1D โดยตำแหน่งการเรียกใช้จะต่างไป โดยจะนับจากแถวก่อน แล้วไล่ไปตามจำนวนคอลัมน์ เช่น รูปแถวที่ 2 คอลัมน์ที่ 2 จะได้ว่า ax.List[3]
fig.set_size_inches(10,10)                   # set ขนาดของ fig โดยจะกำหนดค่าของ ความกว้าง(แกน x) และ ความสูง(แกน y)

# the enumerate() function adds a counter to an iterable. It returns an iterator that produces tuples containing (index, element) for each element in the input iterable.
for ax in enumerate(axList):  # ax[0] จะมี 0, 1, 2, 3, 4, 5  ax[1] จะมี 0, 1, 2, 3, 4, 5
    loc = ax[0]
    ax = ax[1]

    # .xs function แสดงค่าเฉพาะ label ตามที่กำหนด  axis = 0 หมายถึง row , axis = 1 หมายถึง column
    data = coefficients.xs(loc, level=1, axis=1)
    data.plot(marker='o', ls='', ms=2.0, ax=ax, legend=False)

    if ax is axList[0]:
        ax.legend(loc=4)

    ax.set(title='Coefficient Set '+str(loc))

plt.tight_layout()
```

```
###########################################################################

# Predict the class and the probability for each
y_pred = list()
y_prob = list()

coeff_labels = ['lr', 'l1', 'l2']
coeff_models = [lr, lr_l1, lr_l2]

for lab,mod in zip(coeff_labels, coeff_models):
    y_pred.append(pd.Series(mod.predict(X_test), name=lab))
    y_prob.append(pd.Series(mod.predict_proba(X_test).max(axis=1), name=lab))

y_pred = pd.concat(y_pred, axis=1)
y_prob = pd.concat(y_prob, axis=1)

###########################################################################

# Calculate error metrics
### BEGIN SOLUTION
from sklearn.metrics import precision_recall_fscore_support as score
from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score
from sklearn.preprocessing import label_binarize

metrics = list()
cm = dict()

for lab in coeff_labels:

    # Preciision, recall, f-score from the multi-class support function
    precision, recall, fscore, _ = score(y_test, y_pred[lab], average='weighted')

    # The usual way to calculate accuracy
    accuracy = accuracy_score(y_test, y_pred[lab])

    # ROC-AUC scores can be calculated by binarizing the data
    auc = roc_auc_score(label_binarize(y_test, classes=[0,1,2,3,4,5]),
              label_binarize(y_pred[lab], classes=[0,1,2,3,4,5]),
              average='weighted')

    # Last, the confusion matrix
    cm[lab] = confusion_matrix(y_test, y_pred[lab])

    metrics.append(pd.Series({'precision':precision, 'recall':recall,
                              'fscore':fscore, 'accuracy':accuracy,
                              'auc':auc},
                             name=lab))

metrics = pd.concat(metrics, axis=1)
```

```
# Display the confusion metric
### BEGIN SOLUTION
fig, axList = plt.subplots(nrows=2, ncols=2)
axList = axList.flatten()
fig.set_size_inches(12, 10)

axList[-1].axis('off')

for ax,lab in zip(axList[:-1], coeff_labels):
    sns.heatmap(cm[lab], ax=ax, annot=True, fmt='d');
    ax.set(title=lab);

plt.tight_layout()
### END SOLUTION
```