Human Activity Recognition and Smartphone dataset

I decided to use Human Activity Recognition and Smartphone dataset for this project because it comes with the features and target variable that will be useful in applying supervised classification model. The dataset is available in this course's first lab and the useful information from the author who prepared the dataset is available at UC Irvine Machine Learning Repository
https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

Main Objective: Classification

I will be using supervised logistic regression model in classifying activities into one of the six activities (walking, walking upstairs, walking downstairs, sitting, standing, and laying) using the dataset.

Dataset Summary

Here are some attributes of the dataset:

Number of rows	10299
Number of columns	562
Number of columns with object data types	1
Number of columns with float data types	561
Number of columns with int data types	0

Here is the list of columns:

```
data.columns
```

```
Index(['tBodyAcc-mean()-X', 'tBodyAcc-mean()-Y', 'tBodyAcc-mean()-Z',
       'tBodyAcc-std()-X', 'tBodyAcc-std()-Y', 'tBodyAcc-std()-Z',
       'tBodyAcc-mad()-X', 'tBodyAcc-mad()-Y', 'tBodyAcc-mad()-Z',
       'tBodyAcc-max()-X',
       ...
       'fBodyBodyGyroJerkMag-skewness()', 'fBodyBodyGyroJerkMag-kurtosis()',
       'angle(tBodyAccMean,gravity)', 'angle(tBodyAccJerkMean),gravityMean)',
       'angle(tBodyGyroMean,gravityMean)',
       'angle(tBodyGyroJerkMean,gravityMean)', 'angle(X,gravityMean)',
       'angle(Y,gravityMean)', 'angle(Z,gravityMean)', 'Activity'],
      dtype='object', length=562)
```

'Activity' is the target column/label.

Data Exploration

Breakdown of activities:

```
data.Activity.value_counts()
```

```
LAYING                1944
STANDING              1906
SITTING               1777
WALKING               1722
WALKING_UPSTAIRS      1544
WALKING_DOWNSTAIRS    1406
Name: Activity, dtype: int64
```

Data Cleaning

The Human Activity Recognition dataset is already clean and well maintained by the author and does not contain significant number of missing or duplicate data that would require data cleansing steps.

Feature Engineering

Label encoding for Activity column:

Using the below code, the Activity column was encoded.

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
data['Activity'] = le.fit_transform(data.Activity)
data['Activity'].sample(5)
### END SOLUTION
```

After transforming the activities:

```
data.Activity.value_counts()
```

```
0    1944
2    1906
1    1777
3    1722
5    1544
4    1406
```

Calculate correlations and dropping the values:

Simplified by emptying all the data below diagonal and then stacking the data and converting into a dataframe to see how dependent variables are correlated to each other.

	feature1	feature2	correlation	abs_correlation
0	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	0.128037	0.128037
1	tBodyAcc-mean()-X	tBodyAcc-mean()-Z	-0.230302	0.230302
2	tBodyAcc-mean()-X	tBodyAcc-std()-X	0.004590	0.004590
3	tBodyAcc-mean()-X	tBodyAcc-std()-Y	-0.016785	0.016785
4	tBodyAcc-mean()-X	tBodyAcc-std()-Z	-0.036071	0.036071
...	...	...	...	...
157075	angle(tBodyGyroJerkMean,gravityMean)	angle(Y,gravityMean)	-0.004582	0.004582
157076	angle(tBodyGyroJerkMean,gravityMean)	angle(Z,gravityMean)	-0.012549	0.012549
157077	angle(X,gravityMean)	angle(Y,gravityMean)	-0.748249	0.748249
157078	angle(X,gravityMean)	angle(Z,gravityMean)	-0.635231	0.635231
157079	angle(Y,gravityMean)	angle(Z,gravityMean)	0.545614	0.545614

157080 rows × 4 columns

Finding most highly correlated values:

```
# The most highly correlated values
corr_values.sort_values('correlation', ascending=False).query('abs_correlation>0.8')
### END SOLUTION
```

	feature1	feature2	correlation	abs_correlation
156894	fBodyBodyGyroJerkMag-mean()	fBodyBodyGyroJerkMag-sma()	1.000000	1.000000
93902	tBodyAccMag-sma()	tGravityAccMag-sma()	1.000000	1.000000
101139	tBodyAccJerkMag-mean()	tBodyAccJerkMag-sma()	1.000000	1.000000
96706	tGravityAccMag-mean()	tGravityAccMag-sma()	1.000000	1.000000
94257	tBodyAccMag-energy()	tGravityAccMag-energy()	1.000000	1.000000
...	...	...	...	...
22657	tGravityAcc-mean()-Y	angle(Y,gravityMean)	-0.993425	0.993425
39225	tGravityAcc-arCoeff()-Z,3	tGravityAcc-arCoeff()-Z,4	-0.994267	0.994267
38739	tGravityAcc-arCoeff()-Z,2	tGravityAcc-arCoeff()-Z,3	-0.994628	0.994628
23176	tGravityAcc-mean()-Z	angle(Z,gravityMean)	-0.994764	0.994764
38252	tGravityAcc-arCoeff()-Z,1	tGravityAcc-arCoeff()-Z,2	-0.995195	0.995195

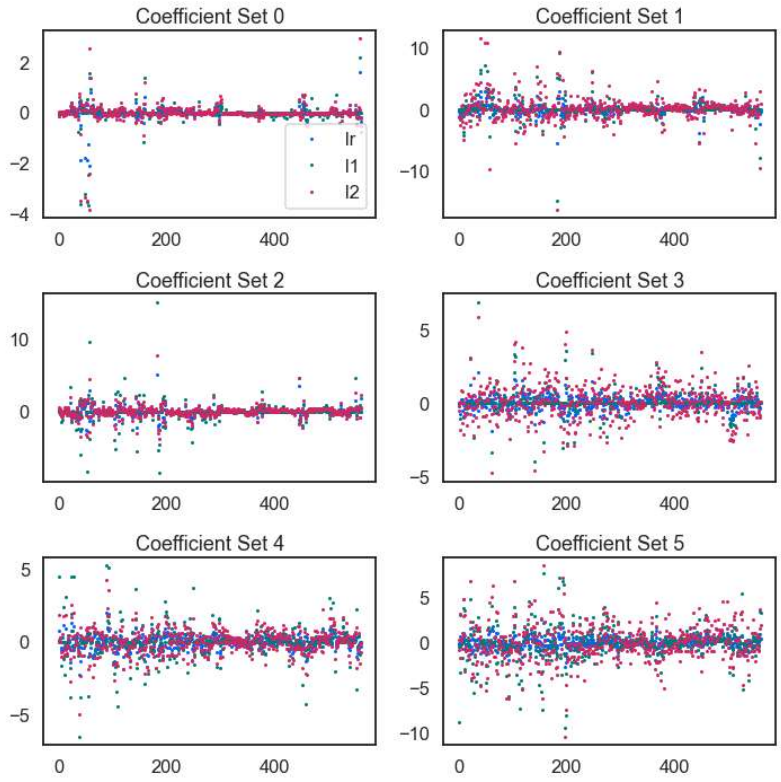
22815 rows × 4 columns

Summary of Training

Logistic Regression

The target Activity column was label encoded in the dataset before using that for fitting to logistic regression model. Before fitting the model, the dataset was split into training and testing datasets with testing dataset size of 30%.

After fitting logistic regression model to both datasets without regularization, the L1 and L2 regularizations were applied to the dataset. All three models were stored for coefficient magnitude comparison.



Predicting class for each model:

	lr	l1	l2
0	3	3	3
1	5	5	5
2	3	3	3
3	1	1	1
4	0	0	0

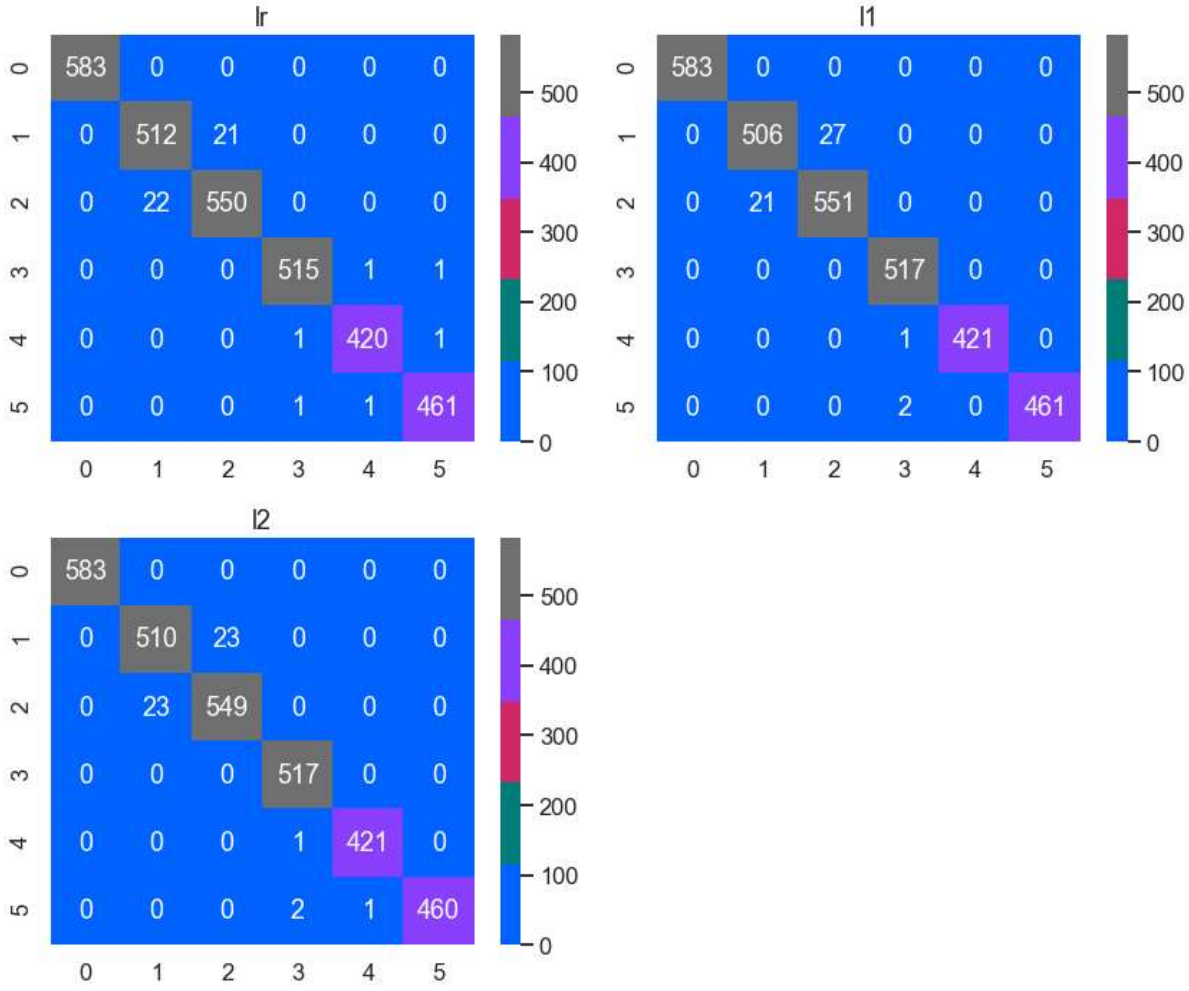
Probability of each model:

	lr	l1	l2
0	0.998939	0.999010	0.999758
1	0.988165	0.999838	0.999477
2	0.987592	0.995565	0.999696
3	0.981381	0.999189	0.994338
4	0.998277	0.999921	0.999997

Calculating error metrics for each model:

	lr	l1	l2
precision	0.984144	0.983514	0.983824
recall	0.984142	0.983495	0.983819
fscore	0.984143	0.983492	0.983819
accuracy	0.984142	0.983495	0.983819
auc	0.990384	0.989949	0.990165

Plotting confusion matrix:



Key Findings and Insights

When applying the L1 regularization with the penalty, the processing took a while to finish to fit the model. L2 regularization tend to work faster than L1. Also, upon comparing the coefficients for 6 classifications in all 3 models (simple logistic regression, LR with L1 and LR with L2), we noticed that the coefficients were closely matched for activity 0 and they were sparsely scattered for activity 3,4 and 5.

Next Steps

The next step would be to cover all the topics discussed in the course and create a one python notebook to provide end to end solution for classifying activities using K-Nearest Neighbor (K-N-N) and Support Vector Machine (SVM). The dataset used for this exercise was small. Another next step is to find a dataset that can be used to fit models using different classifiers.