DATA ANALYSIS ON HUMAN ACTIVITY RECOGNITION USING SMARTPHONES DATASET

SOEN 691 - Big Data Analytics (Winter 2020)

NAME	STUDENT ID
Girish Kumar Kadapa	40083533
Naren Morabagal Somasekhar	40082567
Adarsh Aravind	40082585
Liangzhao Lin	40085480

AGENDA

1. INTRODUCTION
2. DATASET
3. TECHNOLOGIES
4. DATA PREPROCESSING
5. MODEL TRAINING
6. OUTCOME/EVALUATION METRICS
7. CONCLUSION AND FUTURE WORK

INTRODUCTION

Dataset analysis project.

The goal of this project is to analyze the data, train the model, interpret and discuss the results of the data analysis with various metrics such as accuracy rate, F1 score, precision and recall.

The project is to predict the human activity labels such as "Walking, Walking_Upstairs, Walking_Downstairs, Sitting, Standing, Laying" from the data collected from the smartphone sensors.

DATASET

The dataset has captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz, containing 561-feature vector with time and frequency domain variables and labels namely six activities (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone on the waist performed by a person.

Data Set Characteristics	Multivariate, Time-Series
Associated Tasks	Classification, Clustering
Number of Instances	10299
Number of Attributes	561

DATASET (CONT...)

For each record in the dataset it is provided:

- Triaxial acceleration from the accelerometer (total acceleration) and the estimated body acceleration.
- Triaxial Angular velocity from the gyroscope.
- A 561-feature vector with time and frequency domain variables.
- Its activity labels.
- An identifier of the subject who carried out the experiment.

The movement data recorded was the x, y, and z accelerometer data (linear acceleration) and gyroscopic data (angular velocity) from the smartphone.

TECHNOLOGIES

1. Supervised Machine Learning Algorithms
   1.1. Random Forest
   1.2. K-nearest Neighbours
   1.3. Logistic Regression
2. Scikit-learn
3. Python Libraries - Matplotlib, seaborn, and plotly for plotting, GridSearch, Numpy for array vectorization, Pandas dataframes, precision, recall, f1_score, SimpleImputer, MinMaxScaler, confusion_matrix

DATA PRE-PROCESSING

1. Eliminate duplicate values
2. Remove/Impute null values
3. Scaling the features between the range [-1, 1]
4. Feature selection
5. Removing outliers
6. Data visualization
7. Class labels balance check

DATA PRE-PROCESSING (CONT…)

Check if any duplicate rows are available

```
In [214]: train_features[train_features.duplicated()].count().sum()
Out[214]: 0
```

Check if any null values are present and impute them with column mean (if the count is insignificant)

```
In [215]: Imputer = SimpleImputer(missing_values=np.nan, strategy="mean")
```

```
In [216]: train_features = Imputer.fit_transform(train_features.values)
          train_features = pd.DataFrame(train_features)
```

```
In [217]: train_features.isnull().sum().sum()
Out[217]: 0
```

Transform features by scaling each feature between [-1 1]

```
In [218]: scaler = MinMaxScaler(feature_range=(-1,1))
```

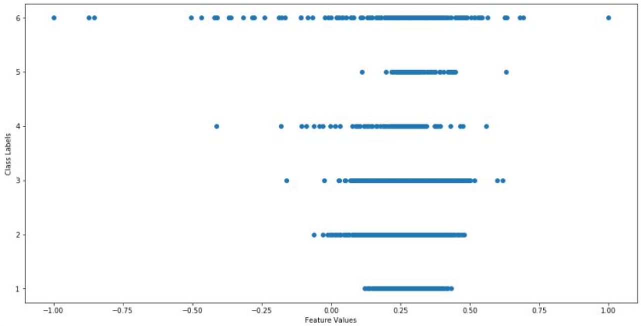
```
In [219]: train_features = pd.DataFrame(scaler.fit_transform(train_features.values))
```

```
In [220]: train_features.describe()
Out[220]:
```

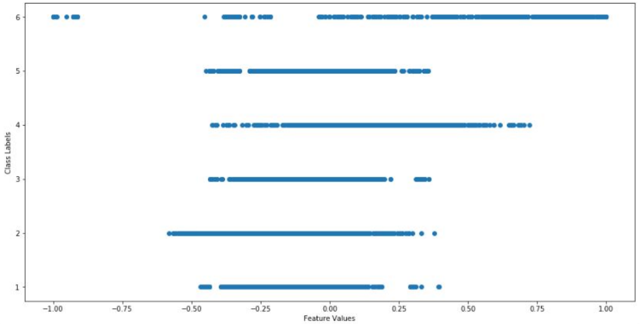
	0	1	2	3	4	5	6	7	8
count	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.000000	7352.0
mean	0.274488	-0.017695	-0.109141	-0.605438	-0.489659	-0.604754	-0.630512	-0.519133	-0.6061
std	0.070261	0.040811	0.056635	0.448734	0.524651	0.418687	0.424073	0.493927	0.4141
min	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.0000
25%	0.262975	-0.024863	-0.120993	-0.992754	-0.977304	-0.980233	-0.993591	-0.977803	-0.9802
50%	0.277193	-0.017219	-0.108676	-0.946196	-0.845545	-0.859365	-0.950709	-0.854984	-0.8571
75%	0.288461	-0.010783	-0.097794	-0.242813	0.007919	-0.262415	-0.292680	-0.051363	-0.2656
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000

DATA PRE-PROCESSING (CONT...)

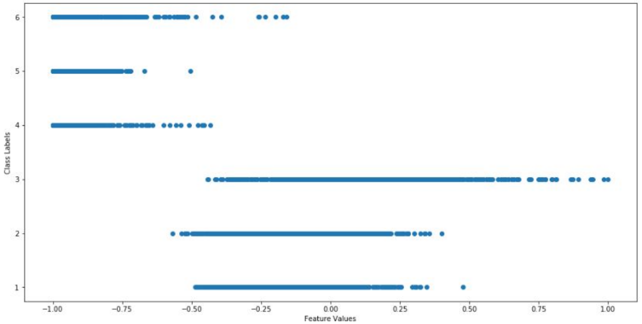
In [223]:
```
fig, ax = plt.subplots(figsize=(16,8))
ax.scatter(train_features[0],train_labels)
plt.xlabel(" Feature Values ")
plt.ylabel(" Class Labels ")
plt.show()
```



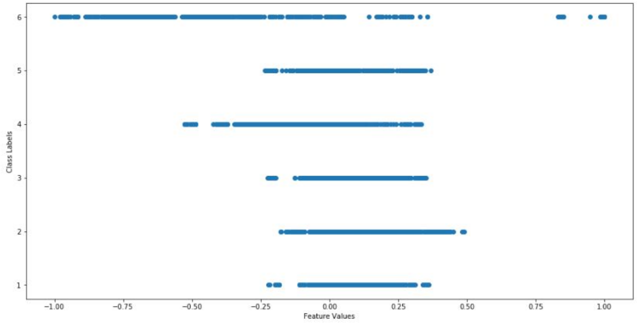
In [224]:
```
fig, ax = plt.subplots(figsize=(16,8))
ax.scatter(train_features[51],train_labels)
plt.xlabel(" Feature Values ")
plt.ylabel(" Class Labels ")
plt.show()
```



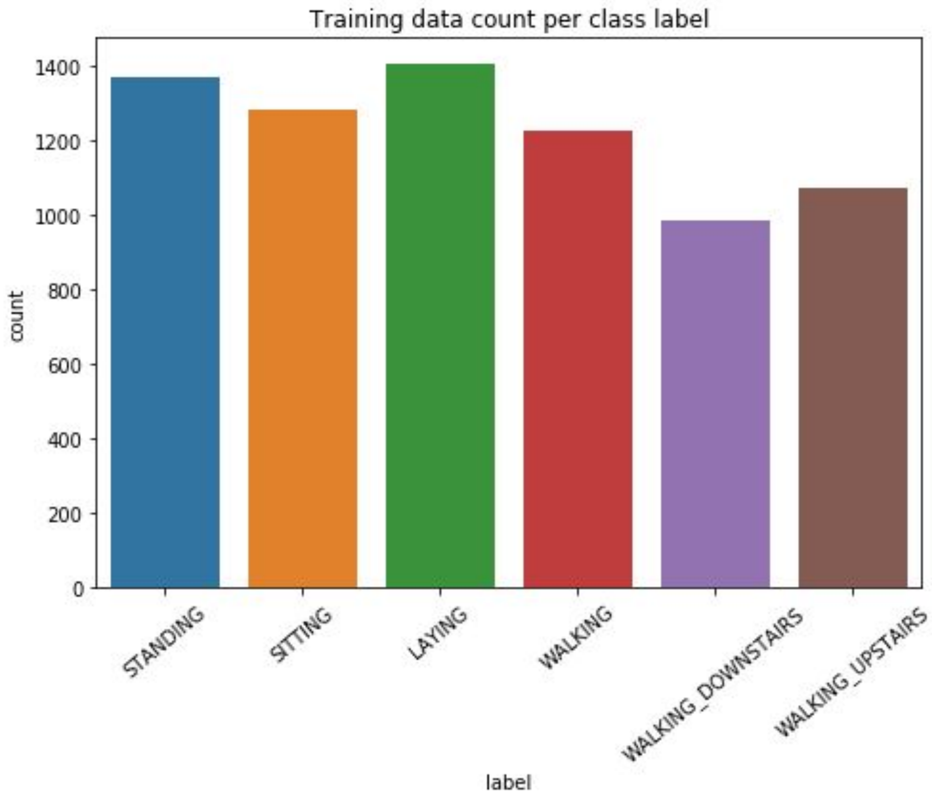
In [227]:
```
fig, ax = plt.subplots(figsize=(16,8))
ax.scatter(train_features[280],train_labels)
plt.xlabel(" Feature Values ")
plt.ylabel(" Class Labels ")
plt.show()
```



In [228]:
```
fig, ax = plt.subplots(figsize=(16,8))
ax.scatter(train_features[560],train_labels)
plt.xlabel(" Feature Values ")
plt.ylabel(" Class Labels ")
plt.show()
```



DATA BALANCE



MODEL TRAINING - GRIDSEARCH

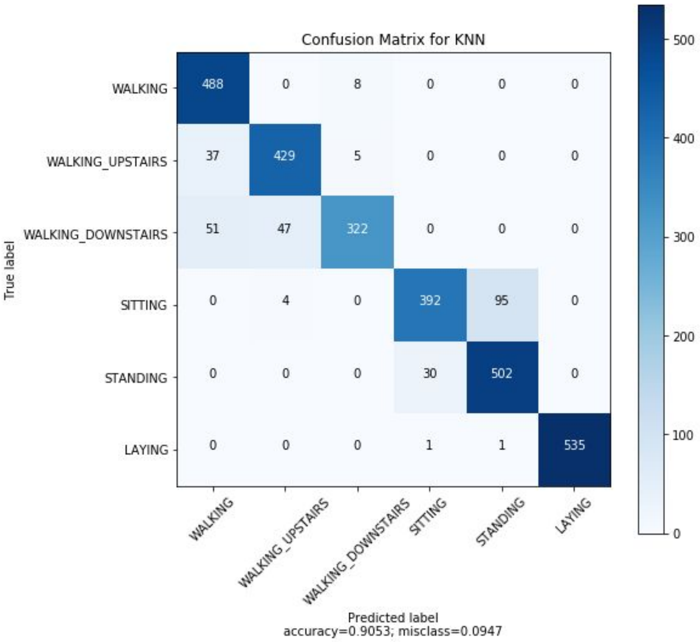
```
In [215]: knn_params = {'n_neighbors':np.array(range(9,18))}
          knn = KNeighborsClassifier()
          gridcv_knn = GridSearchCV(knn, knn_params, verbose=False, cv=3)

In [216]: gridcv_knn.fit(train_features,train_labels['label'].ravel())

Out[216]: GridSearchCV(cv=3, error_score='raise-deprecating',
                       estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                                      metric='minkowski',
                                                      metric_params=None, n_jobs=None,
                                                      n_neighbors=5, p=2,
                                                      weights='uniform'),
                       iid='warn', n_jobs=None,
                       param_grid={'n_neighbors': array([ 9, 10, 11, 12, 13, 14, 15, 16, 17])},
                       pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                       scoring=None, verbose=False)

In [217]: gridcv_knn.best_params_

Out[217]: {'n_neighbors': 17}
```

KNN

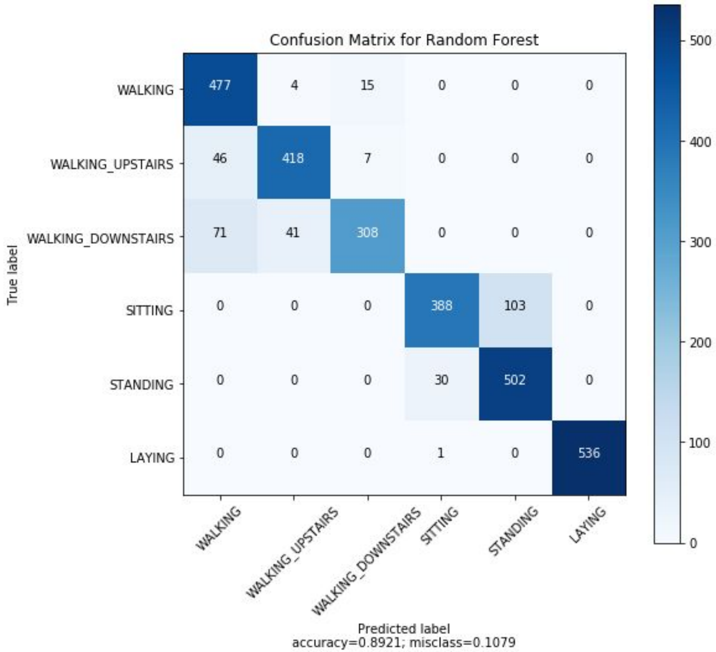
```
In [222]: rf_params = {'n_estimators' : np.arange(10,30,10),'max_depth' : np.arange(1,6,2)}
          rf = RandomForestClassifier(random_state=0)
          gridcv_rf = GridSearchCV(rf, rf_params, verbose=False, cv=3)

In [223]: gridcv_rf.fit(train_features,train_labels['label'].ravel())

Out[223]: GridSearchCV(cv=3, error_score='raise-deprecating',
                       estimator=RandomForestClassifier(bootstrap=True, class_weight=None,
                                                        criterion='gini', max_depth=None,
                                                        max_features='auto',
                                                        max_leaf_nodes=None,
                                                        min_impurity_decrease=0.0,
                                                        min_impurity_split=None,
                                                        min_samples_leaf=1,
                                                        min_samples_split=2,
                                                        min_weight_fraction_leaf=0.0,
                                                        n_estimators='warn', n_jobs=None,
                                                        oob_score=False, random_state=0,
                                                        verbose=0, warm_start=False),
                       iid='warn', n_jobs=None,
                       param_grid={'max_depth': array([1, 3, 5]),
                                   'n_estimators': array([10, 20])},
                       pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                       scoring=None, verbose=False)

In [224]: gridcv_rf.best_params_

Out[224]: {'max_depth': 5, 'n_estimators': 20}
```

Random Forest

OUTCOME/EVALUATION METRICS - CONFUSION MATRIX



KNN



Random Forest

OUTCOME/EVALUATION METRICS (CONT...)

```
knn_predictions = gridcv_knn.predict(test_features)
```

```
print(classification_report(test_labels,knn_predictions))
```

	precision	recall	f1-score	support
1	0.85	0.98	0.91	496
2	0.89	0.91	0.90	471
3	0.96	0.77	0.85	420
4	0.93	0.80	0.86	491
5	0.84	0.94	0.89	532
6	1.00	1.00	1.00	537
accuracy			0.91	2947
macro avg	0.91	0.90	0.90	2947
weighted avg	0.91	0.91	0.90	2947

KNN

```
rf_predictions = gridcv_rf.predict(test_features)
```

```
print(classification_report(test_labels,rf_predictions))
```

	precision	recall	f1-score	support
1	0.80	0.96	0.88	496
2	0.90	0.89	0.90	471
3	0.93	0.73	0.82	420
4	0.93	0.79	0.85	491
5	0.83	0.94	0.88	532
6	1.00	1.00	1.00	537
accuracy			0.89	2947
macro avg	0.90	0.89	0.89	2947
weighted avg	0.90	0.89	0.89	2947

Random Forest

OUTCOME/EVALUATION METRICS (CONT...)

We have analysed the accuracy, precision, recall, and F1 score using three machine learning algorithms which is shown in the below table.

		Train Accuracy	Test Accuracy	Precision	Recall	F1 score
1	**KNN**	0.967492	0.905327	0.910584	0.905327	0.904179
2	**Logistic Regression**	0.994423	0.961995	0.963746	0.961995	0.961805
3	**Random Forest**	0.937296	0.892094	0.898754	0.892094	0.890947

CONCLUSION AND FUTURE WORK

Takeaways/Learning from the project:

We got the opportunity to implement end to end machine learning models.

Future work:

We would like to integrate trained models into hardware systems and get the live results and see the model in action.

THANK YOU