Joh Eun Kang
CS-GY 6133
Project 1

Project Title: Help Your Robot

This project is the first project for Artificial Intelligence of CS-GY 6133.
Using the modified data from Kaggle, find a model that can help a robot to detect surfaces.

Algorithm Used:

Random Forest
Why Random Forest?

There are several advantages of using Random forest.

1. Random Forest algorithm can be used for both classification and regression task.
2. Random Forest is powerful method of handle large data set with higher dimensionality. It can handle a lot of features and identity most significant variables. (good for dimensionality reduction)
3. Random Forest classifier can handle the missing values and maintain the high accuracy.
4. It will not allow overfitting trees in the model

Downside of the Random Forest.

1. Need to choose the number of trees.

Getting Started:

Before loading the .ipynb file, please download the new data distributed from the CS-GY6133 on Slack. (X_train_new, X_train_test, y_train_new, y_test_new). Then, copy the location of the files and paste those to the cell below the "2) Load the Data sets".

Procedure and observations:

Importing Library

Load the given data sets

Understanding of the Data

a. Surface Count

According to the surface count of the y_train data, there are large number of data that indicates concrete, soft_pvc, wood,,, etc., but the hard tiles has very small number of data. So, we need to think about what this graph tells us. This graph can be interpreted as

1. If hard tiles do not have significance in its data, the model will be hard to classify them.
2. If the hard tiles do have significance, the small number of data would not be important.

b. Surface in the y_test set

We found that y_test data does not have the carpet, hard tiles, and soft tiles. If we assume that we already know the test data set, we can simply remove that three information in data set. However, in the real world, the "y_test data" is usually unknown. (there was no y_test in the Kaggle competition as well) Therefore, we keep these three surfaces.

c. Correlations:
As we can see from the correlation graph, orientation_X and orientation_W, orientation_Y and orientation_Z, and angular_velocity_Y and angular_velocity Z are most correlated features. So, we may think the top three pairs features can be useful for better classification.

Understanding of X_train, X_test

From the distribution of X_train and X_test data sets, we found that all those data are normally distributed and have same width and height, except the orientation_X and orientation_Y.

Most Supervised machine learning techniques are built on the assumption that data at the training and testing follow the same distribution. In our case, we can do a bootstrap sampling or look for standardization/normalization and feature engineering.

a. Class distribution
From the class distribution graph, we have some more information for classifying the training data set. As we can see from the graphs, there are some class differences in orientation_X, orientation_Y, and linear_acceleration_X, Y, and Z. Therefore, one can think that some statistical analysis such as mean, min, max, difference between min and max of each surface would help for better classification.

b. Scaling of the data
Since these three kinds of features have different unit scale, we should do the scaling. As we can see from the class distribution graphs with standard scaling, we detected differences that we could barely find from the non-scaled data. All the graphs show that every feature contains some significance among them. Also, from the shape of the graph, we are more confidence for doing feature engineering on orientation data. Therefore, adding some features related to the orientations such as standardized range of height, max height, min height, quantiles will be helpful for our model.

Feature Engineering

Based on the understanding of the data, we add additional features that could give us better accuracy.

a) Part 0:
Since the orientation distribution has some differences in test and training data, we try to normalize the orientation angles. The graphs show that there is no big difference between normalized and original data. (normalization may not be required, but we keep those features for now)
b) Part 1:
Add some statistically important features based on the previous steps.
c) Part 2:
Since we detected that the orientations have some significances, we add some more extra features on them.

From the feature engineering we have total of **232 features**.

Model Fitting
For the RandomForestClassifier function, we have to choose some parameters such as n_estimators, n_jobs, and random state. (there are some more variables one can choose)

n_esitmators: The number of trees in the forest
n_jobs: number of jobs to run in parallel.
Random state: controls both the randomness of the bootstrapping of the samples used when building trees

* The Radom Forest model cannot be trained with categorical data. So, we use label encoder to make categorical to numeric variables.

With **232 features**, **random state = 50**

n_estimators	5	10	50	200	1000	3000	10000
n_jobs	-1	-1	-1	-1	-1	-1	-1
Random state	50	50	50	50	50	50	50
Training time	532ms	724ms	3.53s	13.9s	1m 6s	3m 19s	11m 4s
Testing time	6.93ms	8.74ms	21.2ms	47.1ms	302ms	893ms	2.96 s
Confused	fine_concrete VS concrete, tiled	fine_concrete VS concrete, tiled	fine_concrete VS concrete, tiled wood	fine_concrete VS concrete, tiled, wood	fine_concrete VS concrete, tiled, wood	fine_concrete VS concrete, tiled, wood	fine_concrete VS concrete, tiled, wood
Accuracy	**85%**	85%	80%	80%	80%	80%	80%

* In this chart, the random state is set as 50 but even if the random state changes it does not affect to the accuracy. As we use higher number of estimators, running time of both training and testing have increased. From this chart, we observe that we have better accuracy when we have only 5 or 10 of n_estimators. So, can say that this model does not require a lot of decision trees (sub trees) for the better model.

Also, we may think that if the number of estimators is too high, the accuracy goes down because of the overfitting. However, the accuracy of the model remains as 80 from about 100 or higher.
This is because the random forest algorithm has a characteristic that

"Random forests has tendency to overfitting but as the number of trees increases this tendency decreases. That's why as you add more trees the testing performance does not fluctuates and tend to stay in a minima, hence adding more trees does not lead to overfitting but actually decreases it."

https://en.wikipedia.org/wiki/Talk%3ARandom_forest

From the chart's "confused" row, we can see the model has a problem with classifying finite concrete and concrete, and finite concrete and tiled, finite concrete and wood. This result is intuitively understandable. This result implies that we have chosen the features correctly.

Feature Importance

We have trained with **232 features** but some of those features may not be useful. So, we need to check the importance of the features. If some of those do not affect to the accuracy of the model, we can simply remove it.

n_estimators	5	10	50	200	1000	3000	10000
n_jobs	-1	-1	-1	-1	-1	-1	-1
Random state	50	50	50	50	50	50	50
Training time	212ms	416ms	2.34s	9.13s	45.6s	2m 16s	9m 4s
Testing time	7.61ms	8.74ms	21.1ms	56.7ms	291ms	301ms	330ms
# features used	81	88	104	109	115	117	127
Confused	fine_concrete V.S concrete, tiled,	fine_concrete VS concrete, tiled wood tiled VS wood	fine_concrete VS concrete, tiled, wood	fine_concrete VS concrete, tiled, wood	fine_concrete VS concrete, tiled, wood	fine_concrete VS concrete, tiled, wood	fine_concrete VS concrete, tiled, wood
Accuracy	**85%**	80%	80%	80%	80%	80%	80%

As we increased the n_estimators in reduced model, it requires more features to fit the model. This can cause the overfitting, but the model has same accuracy due to the same reason above. Interestingly, the reduced features with n_estimators = 10 got 5% lower accuracy than before. Also, the model could not classify the difference of tiled and wood. This implies that even though some of the features have lower importance than the mean importance, but those effect on the accuracy of the model.

After the feature reduction using mean importance accuracy, estimators = 5 is still the best model with the least number of features.

Result:

As a result, we chose the model using **estimators = 5, n_jobs = -1, 81 number of features** that gives us **85%** of accuracy as a best model.