Capstone Project

Help navigate the robots

Udacity Data Scientist Nanodegree
Melanija Gerasimovska
6/19/2021

Definition

Project Overview

The dataset used for the purpose of this project named '*Help navigate robots'* is from competition: CareerCon 2019 on Kaggle. In this project, we will help robots to recognize the floor surface they are standing on using data collected from Inertial Measurement Units (IMU sensors).
We all know that robots are already part of our daily lives and we know how important is for them to understand the environment we live in. For us it is easy to go from one place to another and we know the information where we stand and on what we stand. But, for robots that is a huge process. We can contribute to that process with solving problems with real data and we can help navigate robots better.
The IMU sensor data is collected while driving a small mobile robot over different floor surfaces on the university premises.
**The data for this competition has been collected by Heikki Huttunen and Francesco Lomio from the Department of Signal Processing and Damoon Mohamadi, Kaan Celikbilek, Pedram Ghazi and Reza Ghabcheloo from the Department of Automation and Mechanical Engineering both from Tampere University, Finland.*

Problem Statement

The main task is to predict which one of the nine floor types (carpet, tiles, concrete, etc.) the robot is on. This can be done by using sensor data such as acceleration and velocity. This way, we can improve the navigation of robots without assistance across different surfaces.
The data is separated into X_train, X_test and Y_train.

So, in this project we used Stratified K-Folds cross-validator (for stratified sampling) and RandomForestClassifier to solve the problem. This is a classification problem, and we expect to have a high accuracy and in most of the time, to predict correctly on what the robot is standing on.

There are many algorithms that can be used to solve this problem, and maybe some of them will give better results than the one we got here. It is a matter of choice and time.

Metrics

The metric used here is accuracy, one of the most common metric for classification. Accuracy takes equaly into account true positives and true negatives.

$$accuracy = \frac{true\ positives + true\ negatives}{size\ of\ data}$$

Also we are using confusion matrix, a technique for summarizing the performance of a classification algorithm. Classification accuracy alone can be misleading if we have an unequal number of observations in each class or if we have more than two classes in our dataset.

A confusion matrix gives an insight not only into the errors being made by our classifier but more importantly the types of errors that are being made.

Analysis

Data Exploration

The dataset we have is pretty much clear data. As we mentioned, the data is separated into X_train, X_test, y_train. There is also a sample_sumbission file for data not seen (and for data we do not know anything).

Explanation of the data:

X_[train/test].csv - the input data, covering 10 sensor channels and 128 measurements per time series plus three ID columns:

row_id: The ID for this row.

series_id: ID number for the measurement series. Foreign key to y_train/sample_submission.

measurement_number: Measurement number within the series.

The orientation channels encode the current angles how the robot is oriented as a quaternion (see here). Angular velocity describes the angle and speed of motion, and linear acceleration components describe how the speed is changing at different times. The 10 sensor channels are:

orientation_X

orientation_Y

orientation_Z

orientation_W

angular_velocity_X

angular_velocity_Y

angular_velocity_Z

linear_acceleration_X

linear_acceleration_Y

linear_acceleration_Z

y_train.csv - the surfaces for training set.

-series_id: ID number for the measurement series.

-group_id: ID number for all of the measurements taken in a recording session. Provided for the training set only, to enable more cross validation strategies.

-surface: the target for this competition.

So, in X_train we have 13 columns and 487680 rows. In X_test we have 13 columns and 488448 rows. That is because we have 6 more series in X_test that are not in X_train. For each series we have 128 measurements.

In the dataset, series_id and measurement_number are int64, row_id is object (string) and the other columns are float64.

Data Visualization

In order to find out more about data, the best way to do that is to plot some parts of it, in order to get more sense of it.

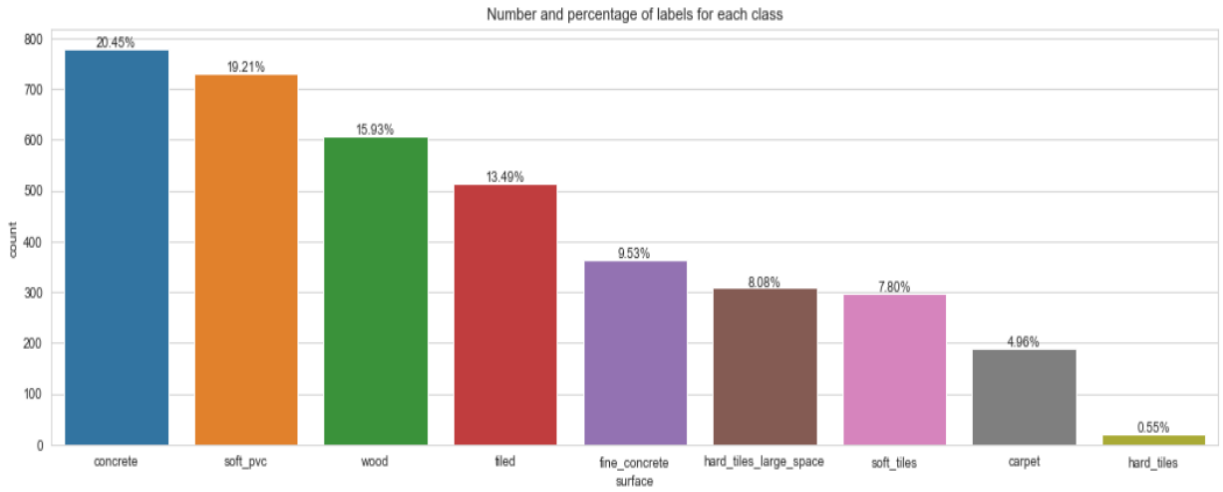

Figure 1 Distribution of surface in target data

In figure 1, we can see the distribution of surfaces in target data. We can see that in most of the time, the robot was on concrete – type of floor, and more rarely on hard_tiles.

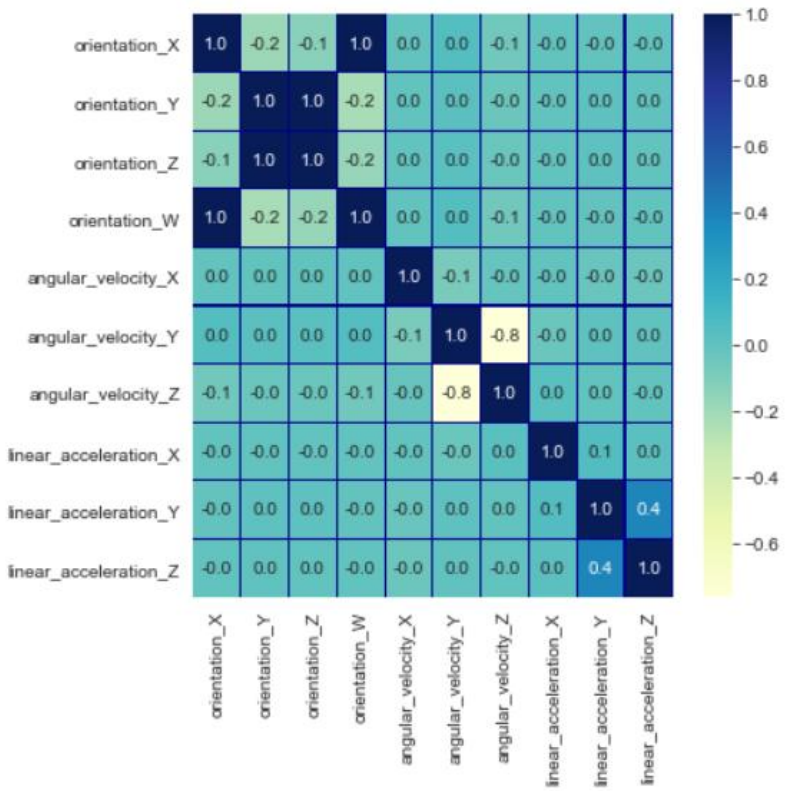

Figure 2 Correlation map of X_train

In the correlation map of X_train, we can see that there is a very strong correlation (1.0) is between orientation_X and orientation_W and between orientation_Z and orientation_Y. There is a strong inverse correlation (-0.8) between angular_velocity_Z and angular_velocity_Y. Also, there is a medium positive correlation (0.4) between linear_acceleration_Y and linear_acceleration_Z.

Now, we are going to see the orientations for series_id = 0.

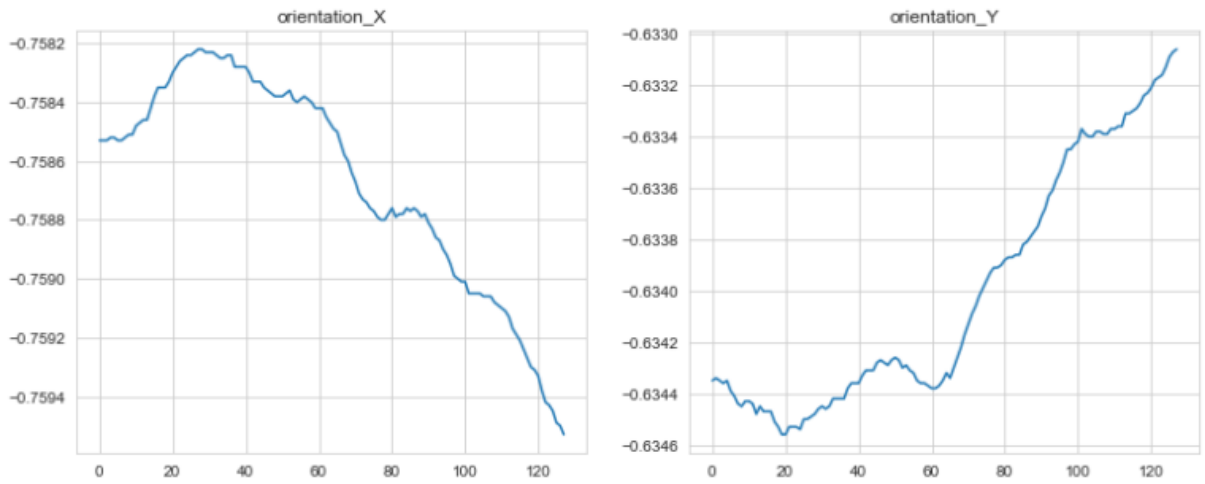

Figure 3 Orientation_X and orientation_Y for series_0

We can see that orientations X and Y are going in different directions.

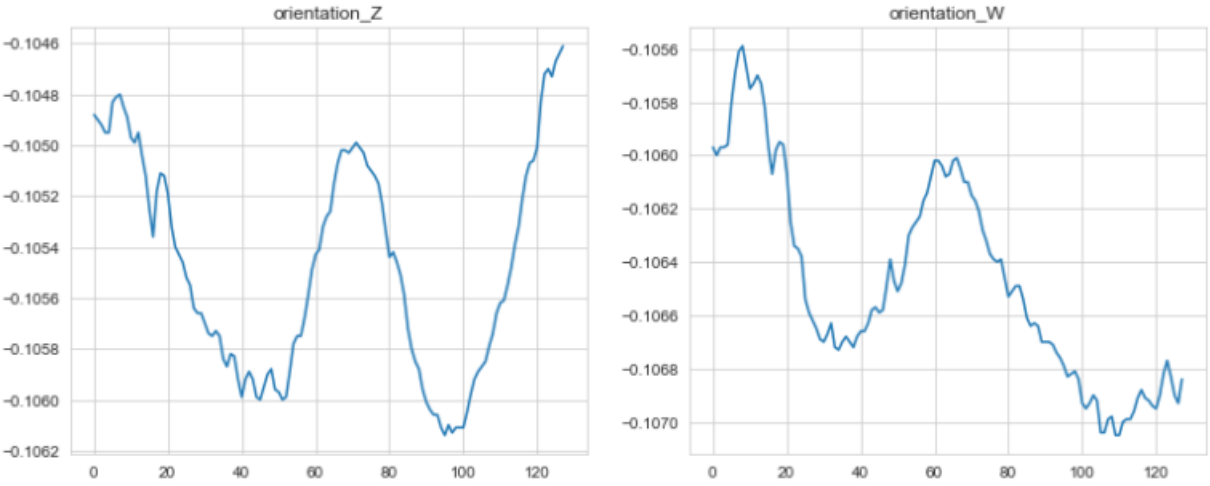

Figure 4 Orientation Z and W for series_0

Orientations Z and W are very different.

From the data visualizations of the linear acceleration in the notebook, there were some patterns in linear acceleration Y that were repeated.

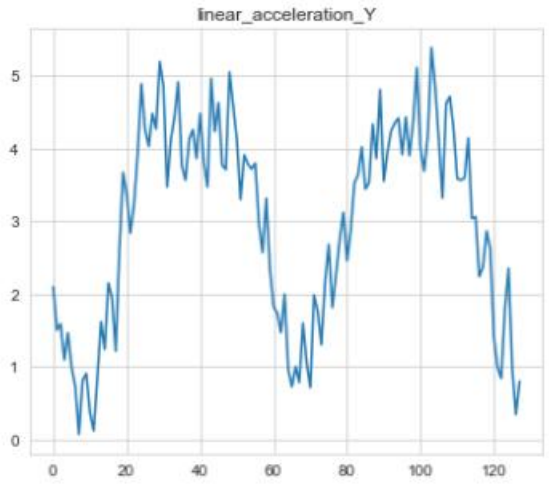

Figure 5 Linear acceleration Y of series_id = 0

There are no abnormalities in the data and no missing values. As we mentioned, the data is clean and we just have to find out the patterns for every surfaces, or better said, the model should do that for us.

Methodology

Data Preprocessing

In this part, we have the feature engineering.

The four coordinates X,Y,Z and W should be converted into Euler Angles, where we get the three angles that the describe orientation of the robot. Euler angles are limited by a phenomenon called "gimbal lock", which prevents them from measuring orientation when the pitch angle approaches +/- 90 degrees. From other side, quaternions provide an alternative measurement technique that is not limited by gimbal lock. But, quaternions are less intuitive than Euler Angles and are more complicated.

After the conversion, we added more features, like median, mean, max, min, standard deviation, total angular velocity, total linear acceleration, kurtosis, skewness, etc. There are no rules for

this part that has to be followed. It all depends on your thoughts about the problem and the expertise in this field.

We used LabelEncoder to give labels on every surface we have, since it is a categorical variable.

We used Standard Scaler to scale the data, because the data is in different ranges.

So, there are three types of scalers: Standard, MinMax and Robust scaler. StandardScaler follows Standard Normal Distribution (SND). Therefore, it makes mean = 0 and scales the data to unit variance. MinMaxScaler scales all the data features in the range [0, 1] or else in the range [-1, 1] if there are negative values in the dataset.

For our purposes, I think Standard Scaler is doing a great job, but in the future, the other types of scaler can be tried to see the differences.

Implementation

For this project and this type of dataset, we used Stratified K Fold with number of splits 20, random state 59, and shuffle equals True.

As an algorithm, Random Forest Classifier was choosen with number of estimators equals 500. These are the results for the 20 folds:

Fold: 0 score: 0.8900523560209425

Fold: 1 score: 0.9162303664921466

Fold: 2 score: 0.9528795811518325

Fold: 3 score: 0.93717277486911

Fold: 4 score: 0.9214659685863874

Fold: 5 score: 0.9476439790575916

Fold: 6 score: 0.9214659685863874

Fold: 7 score: 0.8952879581151832

Fold: 8 score: 0.8952879581151832

Fold: 9 score: 0.9109947643979057

Fold: 10 score: 0.8894736842105263

Fold: 11 score: 0.9421052631578948

Fold: 12 score: 0.868421052631579

Fold: 13 score: 0.9

Fold: 14 score: 0.8947368421052632

Fold: 15 score: 0.9210526315789473

Fold: 16 score: 0.9210526315789473

Fold: 17 score: 0.9421052631578948

Fold: 18 score: 0.9368421052631579

Fold: 19 score: 0.9315789473684211

Avg Accuracy 0.9167925048222652

So, the best accuracy is for fold 2, where we got ~95% accuracy. We are satisfied with this result so far. There were no complications that occured during this process.

With the default number of estimators = 100, we got the following results:

Fold: 0 score: 0.8795811518324608

Fold: 1 score: 0.9057591623036649

Fold: 2 score: 0.9476439790575916

Fold: 3 score: 0.9214659685863874

Fold: 4 score: 0.9109947643979057

Fold: 5 score: 0.9424083769633508

Fold: 6 score: 0.9057591623036649

Fold: 7 score: 0.8795811518324608

Fold: 8 score: 0.8900523560209425

Fold: 9 score: 0.9109947643979057

Fold: 10 score: 0.8894736842105263

Fold: 11 score: 0.9368421052631579

Fold: 12 score: 0.8789473684210526

Fold: 13 score: 0.9

Fold: 14 score: 0.8947368421052632

Fold: 15 score: 0.9157894736842105

Fold: 16 score: 0.9210526315789473

Fold: 17 score: 0.9473684210526315

Fold: 18 score: 0.9421052631578948

Fold: 19 score: 0.9263157894736842

Avg Accuracy 0.9123436208321852

The best accuracy is ~94%.

We increased the estimators for 400, and we still got not so far good results than the ones with the default number of estimators.

Still, the 500 estimators are choosen, since 95% is better than 94% accuracy.

Refinement

So, the feature engineering and scaling the data with Standard Scaler, and using Random Forest Clasifier was successful. It is important to mention that scaling the data is not improving our model, because RandomForest model is completely insensitive to the scale of the dataset features.

We tried Random Forest with default parameters (n_estimators = 100) and with n_estimators = 500.

With the default number of estimators we got the best accuracy ~94%.

We got very good results with n_estimators = 500, accuracy ~95% with a lot of features, and only in few steps. Still, the results are not drastically changed. But, 95% is better than 94%, and with increasing the number of estimators from 100 to 500, we improve the accuracy from 94 to 95%. It is not a big improvement, but it is still an improvement.

It was a great experience to work on this project and understand what I should do with this type of data.

Results

Model Evaluation and Validaition

To obtain a deterministic behaviour during fitting, random_state was fixed. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. And we can see that in all 20 folds, we got accuracy around 90%, which is a good indicator for the performance of the algorithm each time.

Justification

Since we have only one algorithm used, we can not compare Machine Learning algorithms. As mention before, the final results are very high, with number of estimators = 500.

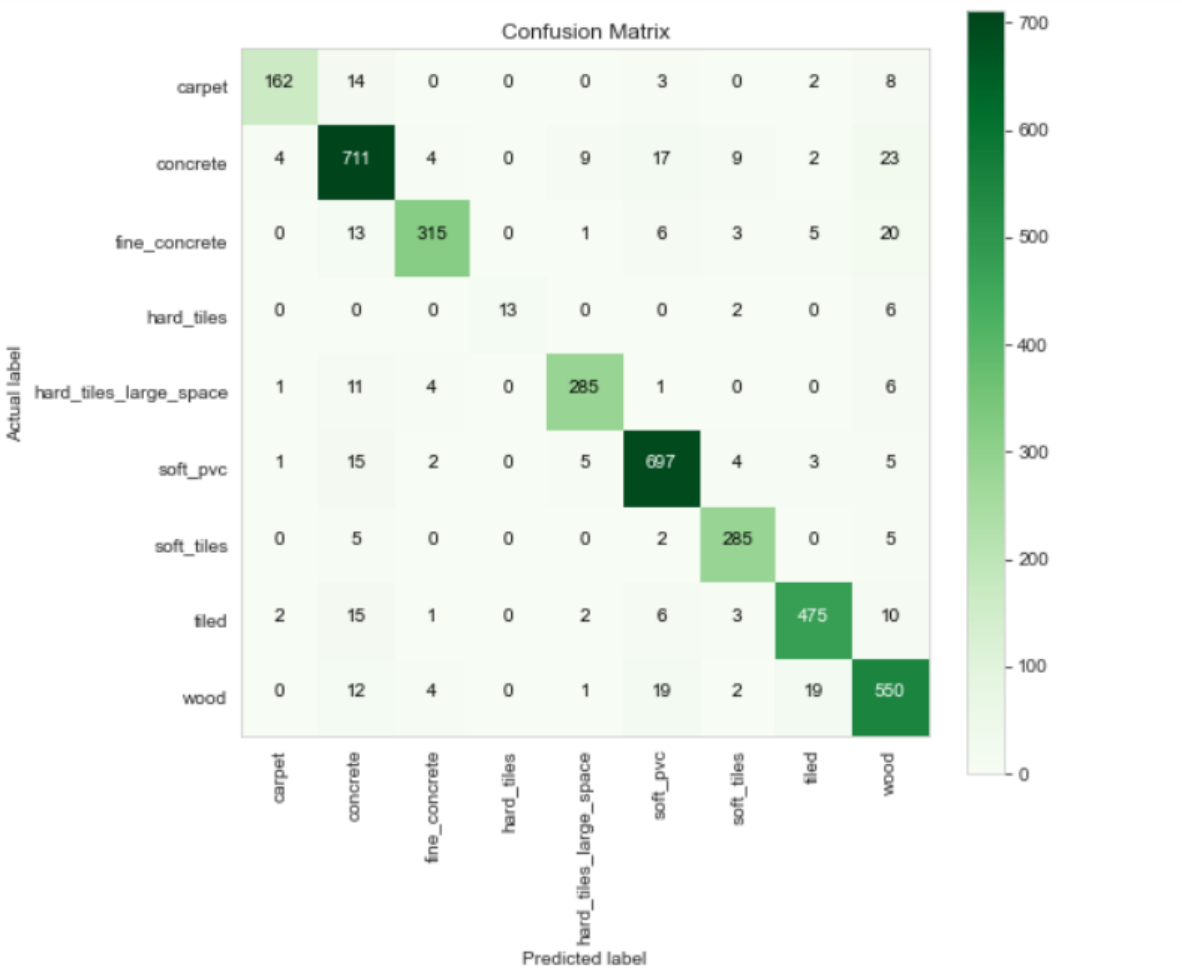

Figure 6 Confusion matrix of the final results

From figure 6, we can see the confusion matrix of the actual and predicted labels. So, it is obvious that the model predicted concrete the most, since most of the data is for surface concrete, but it is good that it predicted hard_tiles a lot, because we do not have a lot of data for it. But this is because the data for every surface is different, and the model can see those differences and learn better.

The current solution to the problem is adequate, since using a Machine learning algorithm as Random Forest is pretty simple, fast and easy to use. This algorithm learned very good (accuracy 95%), and in 95% of the time, will guess correctly on what the robot stands, and we will use this information from the model to navigate the robot better.

We found the answer we are looking for very easily. One simple algorithm solved our problem and gave us amazing results. If we didn't know what the data is, then surely we should go with unsupervised learning, but in this case, Random Forest helped us navigate robots better, so far.

Conclusion

Reflection

I have studied Robotics, but never worked on a project like this, where you also need Data Science skills to solve the problem. During the research, I learned the importance of the Euler angles when we have navigation of a robot, and maybe we can use only them to predict and to navigate the robot better. That can be addressed as a part of improvements. This project is not that tough for solving, since the data was pretty much clear. Choosing features needed a research. In any case, the project was interesting and I was able to get acquainted with a different project and learn how to work with this type of data, as well as develop a model for working in real time.

Improvements

And of course, there are steps that can be done to improve the algorithm.

In further analysis, we should take a closer look on how to control the orientation Z better. Also, we can implement different algorithms, such as SVM, neural networks, etc., that maybe will give us better results.

Random Forest gave us results very fast, and which is good, because we need to know in real time on what the robot stands and to use that information to control the movements of the robot better.

We can try to extract more features, or maybe try with less, to see what we will get.

There are many steps that can be included, and that depends on your choice.