Integrated Engineering Capstone– ES 4499

Distracted Driving – Individual Prototype Assignment

Joerg Oxborough

250849310

Date Submitted: December 6th, 2018

Problem:

Distracted driving is a growing epidemic effecting many young and elderly Canadians. Recently many provinces have enacted laws prohibiting the usage of cellular devices while operating a vehicle however enforcement of these recent laws has proven to be troublesome as it is very difficult to detect when a driver is distracted.

Proposed Solution:

Through the usage of data acquired from the drivers input an algorithm detects sequences of inputs that correlate to distracted driving and notify the user to refocus their attention.

Tools Used:

Python is a scripting programming language and the de-facto tool to use for machine learning due to its extensive libraries in relation to data manipulation, data processing, and machine learning.

Example from Live_Prediction.py:

```
import winsound #Output audio on windows devices
import matplotlib.animation as animation #Create updating graphs
import inputs #Get Xbox controller input
import re #Reformat incoming data
import matplotlib.pyplot as plt #Plot and manipulate data
import numpy as np #Mathematical manipulation with arrays
import pandas as pd #Creating Data frames
import pickle #Saving, and loading models
from sklearn import model_selection #Selecting models
from sklearn import preprocessing   #Preprocessing data for models
from sklearn.metrics import classification_report #Outputing model validiaty
from sklearn.metrics import confusion_matrix #Which data is trained/tested
from sklearn.metrics import accuracy_score #Model accuracy
from sklearn.linear_model import LogisticRegression #Logistic Regression model
from sklearn.tree import DecisionTreeClassifier #Decision Tree model
from sklearn.neighbors import KNeighborsClassifier #Nearest Neighbours model
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis #LDA model
from sklearn.naive_bayes import GaussianNB #NB model
from sklearn.svm import SVC #Support Vector Machine (Takes too long)
from sklearn.ensemble import RandomForestClassifier #Random Forest model
from sklearn.ensemble import ExtraTreesClassifier #Extreme Random Forest model
from sklearn.ensemble import AdaBoostClassifier #ADA boost for Decision Tree
```

Data Collection:

Due to the lack of a physical car and the ethical limitations on obtaining distracted driving data a driving simulation was used. The game “Track Mania” was used for data acquisition purposes. Eleven tracks were traversed while recording the users input firstly non-distracted and then the tracks were completed a second time with the user texting and driving simultaneously. In total 33120 samples were recorded.

Example of data:

```
[[ 255  320 2853    1] #The first column shows the speed input
 [ 255  320 3090    1] #The second and third show the X/Y steering inputs
 [ 255  320 3239    1] #The fourth shows distraction or non-distraction
 ...
 [   0  377 2193    0]
 [   0  377 2193    0]
 [   0  377 2193    0]]
```

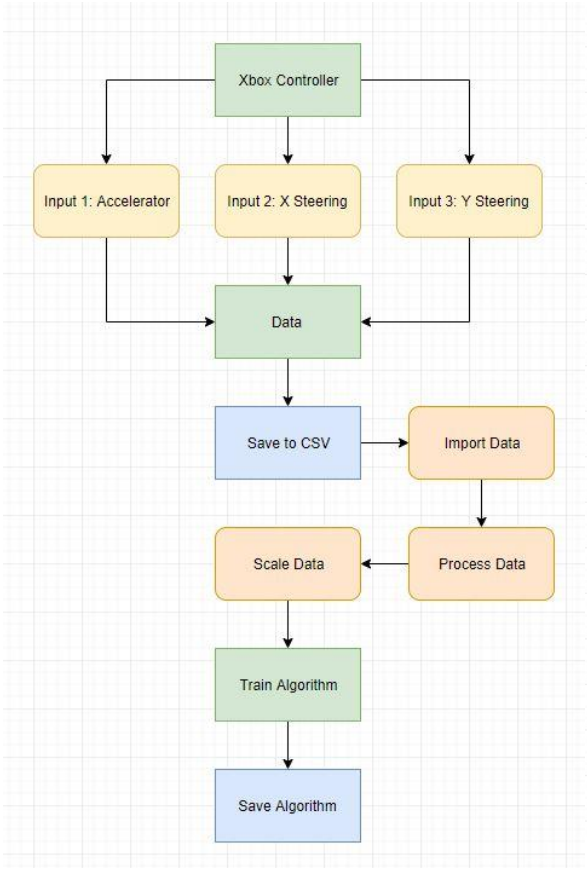
System:

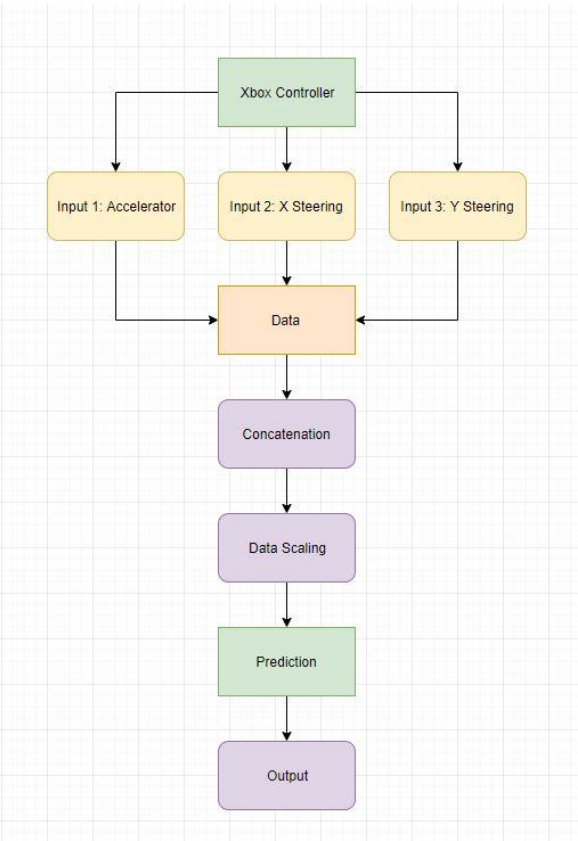
The program ‘InputData.py’ gathers data from an Xbox or similar controller which is then broken down into each component and a new sample is created very time an adjustment is made by the user. Once finalized the data is individually saved in the comma separated values format due to ease of manipulation in the following procedures.

The program ‘Machine learning.py’ then loads all of the samples in the csv format, the samples are first labeled as either ‘Distracted’ or ‘Non-Distracted’ and then concatenated into one list. The data is then converted to a Pandas data frame and transformed into a NumPy array this process is needed to scale the data in order for all the values to be in the range of 0 – 1 as the models function better with a simple input range. The data is then separated into a training set and a validation set after an iterative process the optimal validation size for this data set is 18%. Once the model has been trained with a suitable level of accuracy the model is saved using the Python library Pickle which allows the model to be used in other programs while skipping the training step.

Once the model has been trained and saved it can be used for real time prediction of distracted behaviour ‘Live_Prediction.py’ uses code from ‘InputData.py’ to get the input data and ‘Machine learning.py’ to configure the data and run it through the model.

System Flowchart:





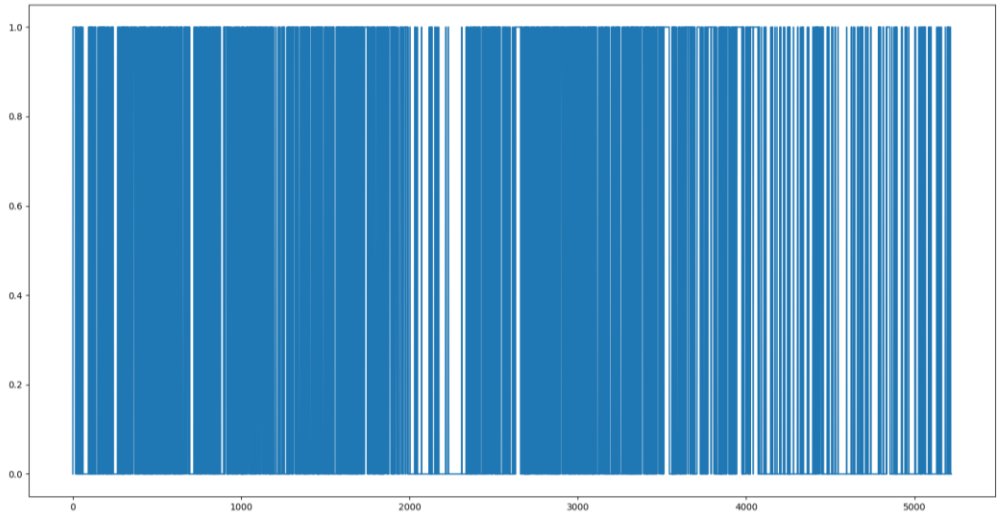
The Model:

Model	**Accuracy (%)**
Logistic Regression	64.55
Linear Discrimination Analysis	64.51
KNeighbour Classifier	65.41
Decision Tree Classifier	69.89
Gaussian NB	64.48
Random Forest Classifier	71.31
Extra Trees Classifier	**72.45**
ADA Boost Classifier	71.15

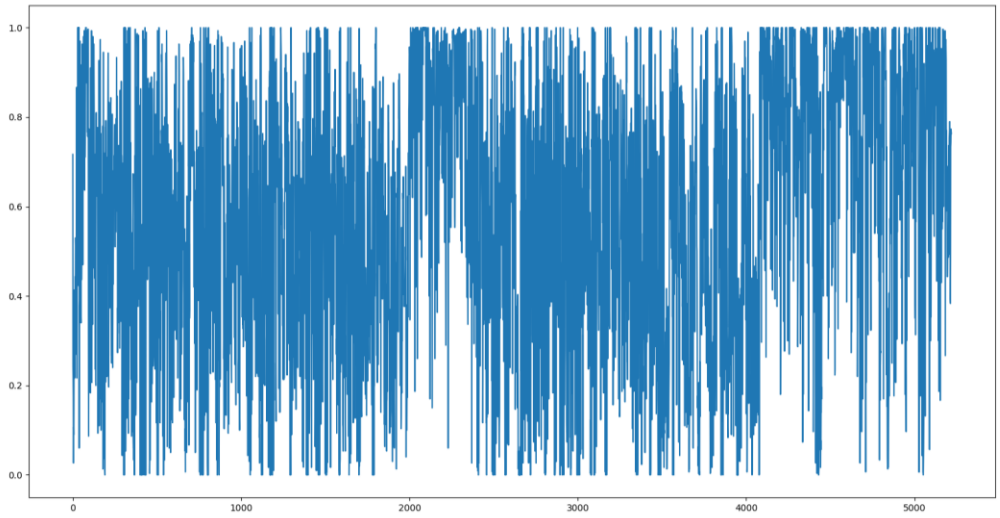
After extensive testing and many iterations the model with the best fit is the Extra-Trees Classifier (Extremely Randomized Trees) this model is well attuned to data that is continuously changing. Although the 72.45% accuracy of the model is somewhat lacking, due to the data gathering process where a test run was labeled either distracted or non-distracted to improve simplicity rather than continuously monitor the user's distraction state. This causes the model to be confused because some data labeled distracted is actually focused and some focused data is distracted. To counteract this issue instead of using a binary distracted or non-distracted state the probability of each state was calculated and the distraction notification only triggers when the model is above 90% confident in a state.

Example Graphs:

The first graph is showing the binary outputs of the model on a 5200 sample dataset.
In this example 1 is non-distracted and 0 is distracted.



The second graph is the probability output of the same model and dataset.



It is much more useful to use the probability as it gives a greater level of confidence and will lead to less false positives.

The graphs were generated using the matplotlib library.

Example code:

```
distracted = ERF.predict_proba(X_Predict)
DisProb, NonProb = zip(*distracted)
plt.plot(DisProb)
plt.show()
```

Full code:

All the project files are included in the following GitHub repository, please note that the device you're using will have to support Python 3.6 (64 bit) you will also have to have pip installed and configure each library.

https://github.com/Joerg-ffs/Capstone-Project