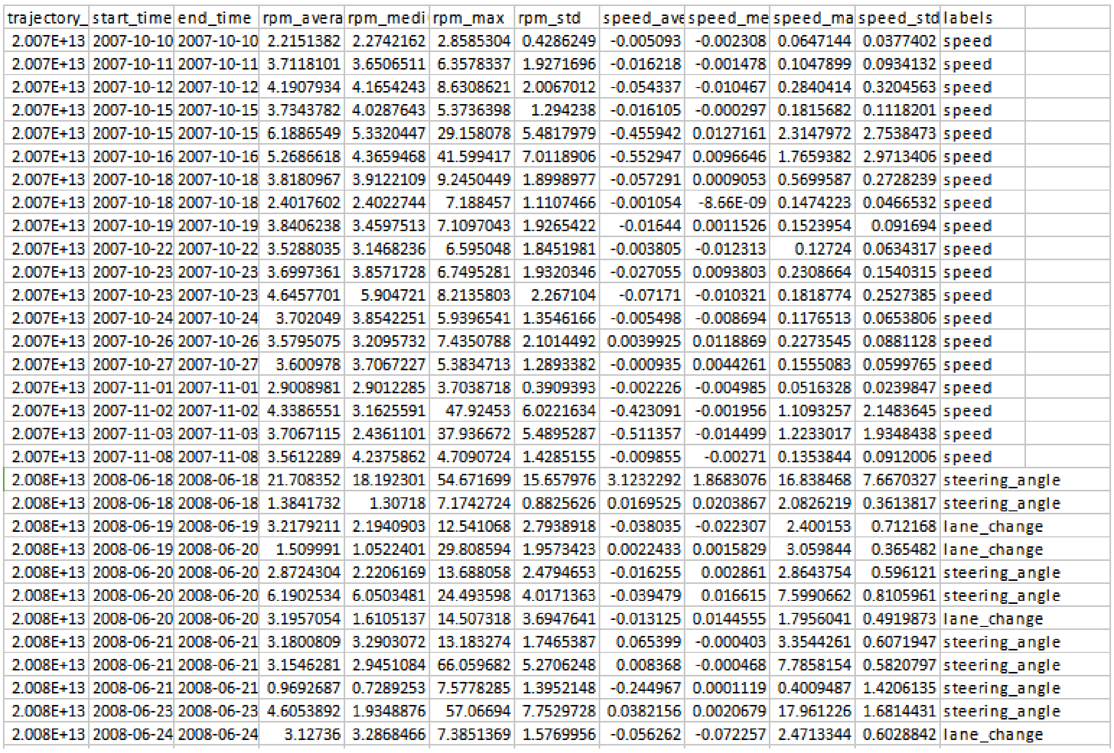
A Driving Decision Strategy (DDS) Based on Machine learning for an autonomous vehicle

Dataset discription:



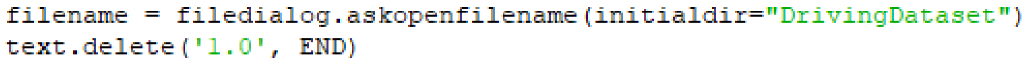
trajectory_	start_time	end_time	rpm_avera	rpm_medi	rpm_max	rpm_std	speed_ave	speed_me	speed_ma	speed_std	labels
2.007E+13	2007-10-10	2007-10-10	2.2151382	2.2742162	2.8585304	0.4286249	-0.005093	-0.002308	0.0647144	0.0377402	speed
2.007E+13	2007-10-11	2007-10-11	3.7118101	3.6506511	6.3578337	1.9271696	-0.016218	-0.001478	0.1047899	0.0934132	speed
2.007E+13	2007-10-12	2007-10-12	4.1907934	4.1654243	8.6308621	2.0067012	-0.054337	-0.010467	0.2840414	0.3204563	speed
2.007E+13	2007-10-15	2007-10-15	3.7343782	4.0287643	5.3736398	1.294238	-0.016105	-0.000297	0.1815682	0.1118201	speed
2.007E+13	2007-10-15	2007-10-15	6.1886549	5.3320447	29.158078	5.4817979	-0.455942	0.0127161	2.3147972	2.7538473	speed
2.007E+13	2007-10-16	2007-10-16	5.2686618	4.3659468	41.599417	7.0118906	-0.552947	0.0096646	1.7659382	2.9713406	speed
2.007E+13	2007-10-18	2007-10-18	3.8180967	3.9122109	9.2450449	1.8998977	-0.057291	0.0009053	0.5699587	0.2728239	speed
2.007E+13	2007-10-18	2007-10-18	2.4017602	2.4022744	7.188457	1.1107466	-0.001054	-8.66E-09	0.1474223	0.0466532	speed
2.007E+13	2007-10-19	2007-10-19	3.8406238	3.4597513	7.1097043	1.9265422	-0.01644	0.0011526	0.1523954	0.091694	speed
2.007E+13	2007-10-22	2007-10-22	3.5288035	3.1468236	6.595048	1.8451981	-0.003805	-0.012313	0.12724	0.0634317	speed
2.007E+13	2007-10-23	2007-10-23	3.6997361	3.8571728	6.7495281	1.9320346	-0.027055	0.0093803	0.2308664	0.1540315	speed
2.007E+13	2007-10-23	2007-10-23	4.6457701	5.904721	8.2135803	2.267104	-0.07171	-0.010321	0.1818774	0.2527385	speed
2.007E+13	2007-10-24	2007-10-24	3.702049	3.8542251	5.9396541	1.3546166	-0.005498	-0.008694	0.1176513	0.0653806	speed
2.007E+13	2007-10-26	2007-10-26	3.5795075	3.2095732	7.4350788	2.1014492	0.0039925	0.0118869	0.2273545	0.0881128	speed
2.007E+13	2007-10-27	2007-10-27	3.600978	3.7067227	5.3834713	1.2893382	-0.000935	0.0044261	0.1555083	0.0599765	speed
2.007E+13	2007-11-01	2007-11-01	2.9008981	2.9012285	3.7038718	0.3909393	-0.002226	-0.004985	0.0516328	0.0239847	speed
2.007E+13	2007-11-02	2007-11-02	4.3386551	3.1625591	47.92453	6.0221634	-0.423091	-0.001956	1.1093257	2.1483645	speed
2.007E+13	2007-11-03	2007-11-03	3.7067115	2.4361101	37.936672	5.4895287	-0.511357	-0.014499	1.2233017	1.9348438	speed
2.007E+13	2007-11-08	2007-11-08	3.5612289	4.2375862	4.7090724	1.4285155	-0.009855	-0.00271	0.1353844	0.0912006	speed
2.008E+13	2008-06-18	2008-06-18	21.708352	18.192301	54.671699	15.657976	3.1232292	1.8683076	16.838468	7.6670327	steering_angle
2.008E+13	2008-06-18	2008-06-18	1.3841732	1.30718	7.1742724	0.8825626	0.0169525	0.0203867	2.0826219	0.3613817	steering_angle
2.008E+13	2008-06-19	2008-06-19	3.2179211	2.1940903	12.541068	2.7938918	-0.038035	-0.022307	2.400153	0.712168	lane_change
2.008E+13	2008-06-19	2008-06-20	1.509991	1.0522401	29.808594	1.9573423	0.0022433	0.0015829	3.059844	0.365482	lane_change
2.008E+13	2008-06-20	2008-06-20	2.8724304	2.2206169	13.688058	2.4794653	-0.016255	0.002861	2.8643754	0.596121	steering_angle
2.008E+13	2008-06-20	2008-06-20	6.1902534	6.0503481	24.493598	4.0171363	-0.039479	0.016615	7.5990662	0.8105961	steering_angle
2.008E+13	2008-06-20	2008-06-20	3.1957054	1.6105137	14.507318	3.6947641	-0.013125	0.0144555	1.7956041	0.4919873	lane_change
2.008E+13	2008-06-21	2008-06-21	3.1800809	3.2903072	13.183274	1.7465387	0.065399	-0.000403	3.3544261	0.6071947	steering_angle
2.008E+13	2008-06-21	2008-06-21	3.1546281	2.9451084	66.059682	5.2706248	0.008368	-0.000468	7.7858154	0.5820797	steering_angle
2.008E+13	2008-06-21	2008-06-21	0.9692687	0.7289253	7.5778285	1.3952148	-0.244967	0.0001119	0.4009487	1.4206135	steering_angle
2.008E+13	2008-06-23	2008-06-23	4.6053892	1.9348876	57.06694	7.7529728	0.0382156	0.0020679	17.961226	1.6814431	steering_angle
2.008E+13	2008-06-24	2008-06-24	3.12736	3.2868466	7.3851369	1.5769956	-0.056262	-0.072257	2.4713344	0.6028842	lane_change

The datase consists of 978 records and twelve columns. Out of twelve columns 11 columns are DDS dataset features and one column is class label. Class label consists either speed or steering_angle or lane_change.

Importing required packages

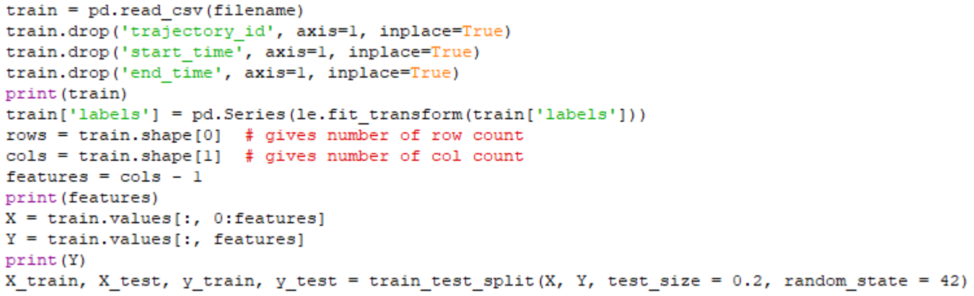
```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
import numpy as np
from tkinter.filedialog import askopenfilename
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from genetic_selection import GeneticSelectionCV
```

Upload dataset



```
filename = filedialog.askopenfilename(initialdir="DrivingDataset")
text.delete('1.0', END)
```

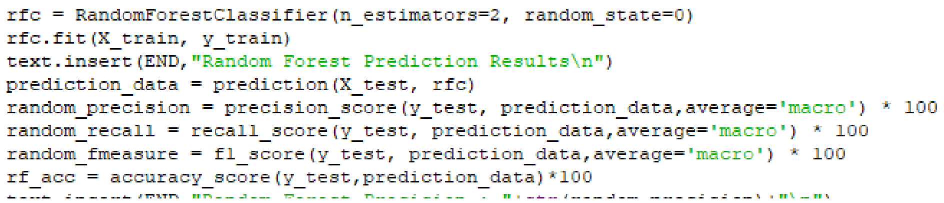
Generate model



```
train = pd.read_csv(filename)
train.drop('trajectory_id', axis=1, inplace=True)
train.drop('start_time', axis=1, inplace=True)
train.drop('end_time', axis=1, inplace=True)
print(train)
train['labels'] = pd.Series(le.fit_transform(train['labels']))
rows = train.shape[0]  # gives number of row count
cols = train.shape[1]  # gives number of col count
features = cols - 1
print(features)
X = train.values[:, 0:features]
Y = train.values[:, features]
print(Y)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)
```

Applying machine learning algorithms

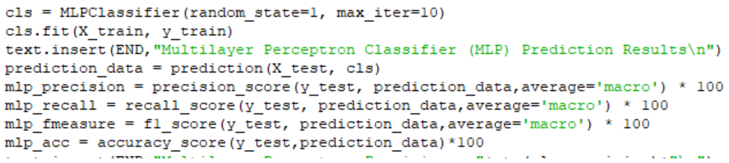
Random forest



```
rfc = RandomForestClassifier(n_estimators=2, random_state=0)
rfc.fit(X_train, y_train)
text.insert(END,"Random Forest Prediction Results\n")
prediction_data = prediction(X_test, rfc)
random_precision = precision_score(y_test, prediction_data,average='macro') * 100
random_recall = recall_score(y_test, prediction_data,average='macro') * 100
random_fmeasure = f1_score(y_test, prediction_data,average='macro') * 100
rf_acc = accuracy_score(y_test,prediction_data)*100
```

RF accuracy:67.3469387755102

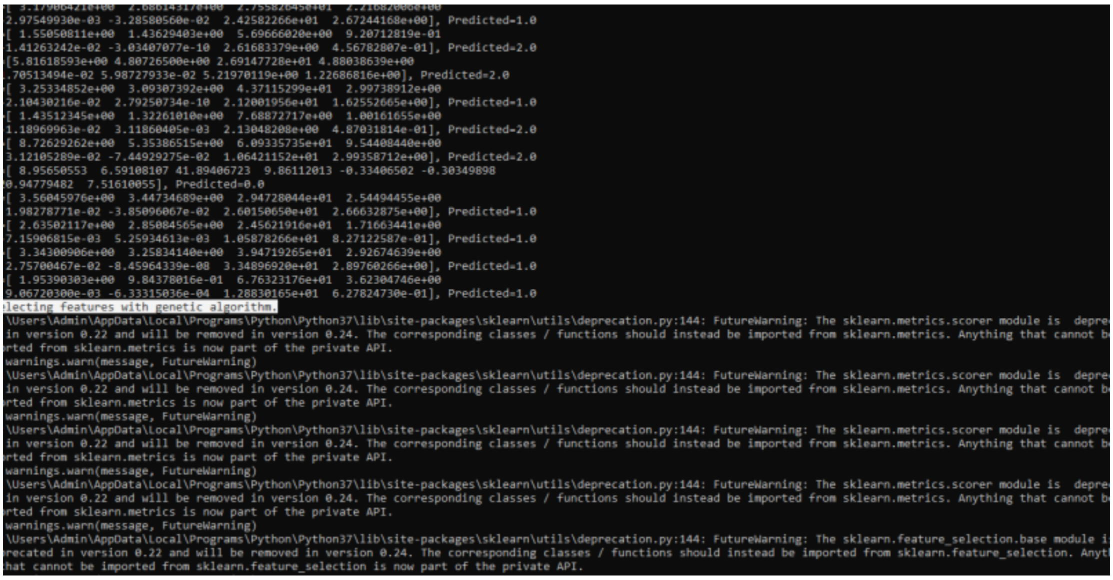
MLP:



```
cls = MLPClassifier(random_state=1, max_iter=10)
cls.fit(X_train, y_train)
text.insert(END,"Multilayer Perceptron Classifier (MLP) Prediction Results\n")
prediction_data = prediction(X_test, cls)
mlp_precision = precision_score(y_test, prediction_data,average='macro') * 100
mlp_recall = recall_score(y_test, prediction_data,average='macro') * 100
mlp_fmeasure = f1_score(y_test, prediction_data,average='macro') * 100
mlp_acc = accuracy_score(y_test,prediction_data)*100
```

MLP accuracy:48.97959183673469

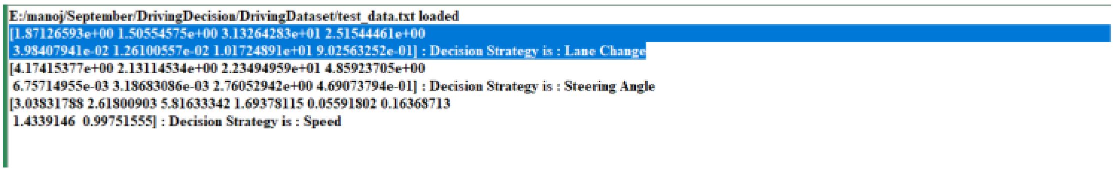
```
dds = RandomForestClassifier(n_estimators=45, random_state=42)
selector = GeneticSelectionCV(dds,  #algorithm name
                              cv=5,
                              verbose=1,
                              scoring="accuracy",
                              max_features=5,
                              n_population=10, #population
                              crossover_proba=0.5, #cross over
                              mutation_proba=0.2,
                              n_generations=50,
                              crossover_independent_proba=0.5,
                              mutation_independent_proba=0.05, #mutation
                              tournament_size=3,
                              n_gen_no_change=5,
                              caching=True,
                              n_jobs=-1)
selector = selector.fit(X_train, y_train)
text.insert(END,"DDS Prediction Results\n")
prediction_data = prediction(X_test, selector)
dds_precision = precision_score(y_test, prediction_data,average='macro') * 100
dds_recall = recall_score(y_test, prediction_data,average='macro') * 100
dds_fmeasure = f1_score(y_test, prediction_data,average='macro') * 100
dds_acc = accuracy_score(y_test,prediction_data)*100
```



```
[ 3.17906421e+00  2.68614317e+00  2.75582645e+01  2.21682006e+00
2.97549930e-03 -3.28580560e-02  2.42582266e+01  2.67244168e+00], Predicted=1.0
[ 1.55050811e+00  1.43629403e+00  5.69666020e+00  9.20712819e-01
1.41263242e-02 -3.03407077e-10  2.61683379e+00  4.56782807e-01], Predicted=2.0
[5.81618593e+00 4.80726500e+00 2.69147728e+01 4.88038639e+00
.70513494e-02 5.98727933e-02 5.21970119e+00 1.22686816e+00], Predicted=2.0
[ 3.25334852e+00  3.09307392e+00  4.37115299e+01  2.99738912e+00
2.10430216e-02  2.79250734e-10  2.12001956e+01  1.62552665e+00], Predicted=1.0
[ 1.43512345e+00  1.32261010e+00  7.68872717e+00  1.00161655e+00
1.18969963e-02  3.11860405e-03  2.13048208e+00  4.87031814e-01], Predicted=2.0
[ 8.72629262e+00  5.35386515e+00  6.09335735e+01  9.54408440e+00
3.12105289e-02 -7.44929275e-02  1.06421152e+01  2.99358712e+00], Predicted=2.0
[ 8.95650553  6.59108107 41.89406723  9.86112013 -0.33406502 -0.30349898
0.94779482  7.51610055], Predicted=0.0
[ 3.56045976e+00  3.44734689e+00  2.94728044e+01  2.54494455e+00
1.98278771e-02 -3.85096067e-02  2.60150650e+01  2.66632875e+00], Predicted=1.0
[ 2.63502117e+00  2.85084565e+00  2.45621916e+01  1.71663441e+00
7.15906815e-03  5.25934613e-03  1.05878266e+01  8.27122587e-01], Predicted=1.0
[ 3.34300906e+00  3.25834140e+00  3.94719265e+01  2.92674639e+00
2.75700467e-02 -8.45964339e-08  3.34896920e+01  2.89760266e+00], Predicted=1.0
[ 1.95390303e+00  9.84378016e-01  6.76323176e+01  3.62304746e+00
9.06720300e-03 -6.33315036e-04  1.28830165e+01  6.27824730e-01], Predicted=1.0
lecting features with genetic algorithm.
\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.metrics.scorer module is  depre
in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.metrics. Anything that cannot b
rted from sklearn.metrics is now part of the private API.
warnings.warn(message, FutureWarning)
\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.metrics.scorer module is  depre
in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.metrics. Anything that cannot b
rted from sklearn.metrics is now part of the private API.
warnings.warn(message, FutureWarning)
\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.metrics.scorer module is  depre
in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.metrics. Anything that cannot b
rted from sklearn.metrics is now part of the private API.
warnings.warn(message, FutureWarning)
\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.metrics.scorer module is  depre
in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.metrics. Anything that cannot b
rted from sklearn.metrics is now part of the private API.
warnings.warn(message, FutureWarning)
\Users\Admin\AppData\Local\Programs\Python\Python37\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.feature_selection.base module i
recated in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.feature_selection. Anyt
hat cannot be imported from sklearn.feature_selection is now part of the private API.
```

Upload DDS dataset for prediction

```
text.insert(END,filename+" loaded\n");
test = pd.read_csv(filename)
test.drop('trajectory_id', axis=1, inplace=True)
test.drop('start_time', axis=1, inplace=True)
test.drop('end_time', axis=1, inplace=True)
cols = test.shape[1]
test = test.values[:, 0:cols]
predict = classifier.predict(test)
print(predict)
for i in range(len(test)):
    if predict[i] == 0:
        text.insert(END,str(test[i])+" : Decision Strategy is : Lane Change\n")
    if predict[i] == 1:
        text.insert(END,str(test[i])+" : Decision Strategy is : Speed\n")
    if predict[i] == 2:
        text.insert(END,str(test[i])+" : Decision Strategy is : Steering Angle\n")
```



```
E:/manoj/September/DrivingDecision/DrivingDataset/test_data.txt loaded
[1.87126593e+00 1.50554575e+00 3.13264283e+01 2.51544461e+00
 3.98407941e-02 1.26100557e-02 1.01724891e+01 9.02563252e-01] : Decision Strategy is : Lane Change
[4.17415377e+00 2.13114534e+00 2.23494959e+01 4.85923705e+00
 6.75714955e-03 3.18683086e-03 2.76052942e+00 4.69073794e-01] : Decision Strategy is : Steering Angle
[3.03831788 2.61800903 5.81633342 1.69378115 0.05591802 0.16368713
 1.4339146  0.99751555] : Decision Strategy is : Speed
```